Oracle® Database
データベース管理者ガイド
19c

F16141-12(原本部品番号:E96348-17)

2023年8月

タイトルおよび著作権情報

Oracle Databaseデータベース管理者ガイド, 19c

F16141-12

原本著者: Mark Doran、Padmaja Potineni、Rajesh Bhatiya

原本協力者: A.Agrawal、L.Ashdown、P.Avril、D.Austin、T.Baby、H.Baer、S.Ball、S.Battula、M.Bauer、T.Bednar、E.Belden、J.Byun、L.Carpenter、A.Chaudhry、C.Chang、B.Cheng、H.Chien、T.Chien、G.Christman、C. C. Chui、L.Clarke、D.Colello、C.Colrain、K.Cook、J.Creighton、A.Dadhich、S.Datta、S.Davidson、M.Dilman、S.Doraiswamy、J.Draaijer、M.Fallen、M.Fuller、D.Gagne、A.Ganesh、GP Gongloor、J.Gonzalez、V.Goorah、S.Gopalan、S.Gupta、B.Habeck、S.Hase、W.Hu、P.Huey、K.Inoue、M.Ito、C.Iyer、K.Itikarlapalli、P.Jaganath、S.Jain、C.Jones、S.Joshi、B.Khaladkar、F.Kobylanski、B.Krishnan、V.Krishnaswamy、A.Kruglikov、B.Kuchibhotla、V.Kuhr、R.Kumar、S.Kumar、V.Kumar、H.Lakshmanan、A.Lee、B.Lee、J.Lee、S. K. Lee、T.Lee、C.Lei、B.Leung、Y.Li、I.Listvinsky、B.Llewellyn、H.Lombera、B.Lundhild、S.Lynn、R.Mani、V.Marwah、C.McGregor、J.McDonnell、J.McHugh、B.McGuirk、J.Meeks、K.Mensah、M.Minhas、K.Mohan、H.Mohankumar、A.Munnolimath、G.Mulagund、P.Murguia、P.Murthy、A.Mylavarapu、V.Moore、N.Muthukrishnan、S.Muthulingam、L.Nim、S.Panchumarthy、R.Pang、V.Panteleenko、R.Pingte、K.Rajamani、A.Raghavan、M.Ramacher、R.Ramkissoon、S.Ravindhran、G.Ravipati、A.Ray、W.Ren、K.Rich、J.Rivera、C. A. L.Rueda、R.Rungta、S.Sahu、P.Shanthaveerappa、S.Sonawane、Y.Sarig、M.Savanur、S.Shankar、D.Sharma、A.Shen、B.Sinha、J.Spiller、D.Steiner、J.Stern、M.Stewart、S.Stoian、R.Swonger、M.Subramaniam、N.Sundarappa、M.Susairaj、A.Tran、A.Tsukerman、C.Tuzla、T.Ueda、K.Umamageswaran、D.Utzig、E.Voss、N.Wagner、X.Wang、M.Wei、S.Wertheimer、P.Wheeler、D.Williams、A.Witkowski、S.Wolicki、D. M. Wong、Z.Yang、T. F. Yu、W.Zhang

目次

はじめに

このマニュアルでは、Oracle Databaseを作成、構成および管理する方法について説明します。

- 対象読者
- ドキュメントのアクセシビリティ
- 関連ドキュメント
- 表記規則

対象読者

このマニュアルは、次のタスクを実行するデータベース管理者を対象にしています。

- 1つ以上のOracle Databaseの作成および構成
- Oracle Databaseの監視およびチューニング
- Oracle Databaseに対する日常的なメンテナンス操作の監視
- 表、索引、ビューなどのスキーマ・オブジェクトの作成およびメンテナンス
- システム・ジョブおよびユーザー・ジョブのスケジュール
- 問題の診断、修復およびレポート

このマニュアルを使用するにあたって、リレーショナル・データベースの概念を理解しておく必要があります。また、Oracle Databaseを実行するオペレーティング・システム環境についても知っておく必要があります。

親トピック: はじめに

ドキュメントのアクセシビリティ

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

親トピック: はじめに

関連ドキュメント

詳細は、次のOracleドキュメントを参照してください。

- Oracle Database 2日でデータベース管理者
- Oracle Database概要
- Oracle Database SQL言語リファレンス
- Oracle Databaseリファレンス

- Oracle Database PL/SQLパッケージおよびタイプ・リファレンス
- Oracle Automatic Storage Management管理者ガイド
- Oracle Database VLDBおよびパーティショニング・ガイド
- Oracle Databaseエラー・メッセージ
- Oracle Database Net Services管理者ガイド
- Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド
- Oracle Databaseパフォーマンス・チューニング・ガイド
- Oracle Database SQLチューニング・ガイド
- Oracle Database開発ガイド
- Oracle Database PL/SQL言語リファレンス
- SQL*Plus ユーザーズ・ガイドおよびリファレンス
- Oracle Multitenant管理者ガイド
- Oracle Database In-Memoryガイド
- Oracle Database Oracle Shardingの使用

このマニュアルの多くの例では、サンプル・スキーマを使用しています。これらのスキーマの詳細は、『Oracle Databaseサンプル・スキーマ』を参照してください。

親トピック: はじめに

表記規則

このドキュメントでは、次のテキスト表記規則が使用されます:

規則	意味
boldface	太字体は、アクションに関連付けられたグラフィカル・ユーザー・インタフェース要素や、本文または用語集で定義されている用語を示します。
italic	イタリック体は、ブック・タイトル、強調、またはユーザーが特定の値を指定するプレースホルダー変数を示します。
`monospace`	等幅体は、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、またはユーザーが入力するテキストを示します。

親トピック: はじめに

このリリースでの『Oracle Database管理者ガイド』の変更点

このドキュメントは、Oracle Databaseの最新リリース用に変更されています。

- Oracle Databaseリリース19c, バージョン19.7での変更点
- Oracle Databaseリリース19c, バージョン19.1での変更点
- Oracle Databaseリリース18cバージョン18.1での変更点
- Oracle Database 12cリリース2 (12.2)での変更点
- Oracle Database 12cリリース1 (12.1.0.2)の変更
- Oracle Database 12cリリース1 (12.1.0.1)の変更

Oracle Databaseリリース19c, バージョン19.7での変更点

Oracle Databaseリリース19c、バージョン19.7での『Oracle Database管理者ガイド』の変更点は次のとおりです。

新機能

- Standard Edition高可用性

  Standard Edition高可用性機能により、Oracle Clusterwareを使用するStandard Edition 2データベースにクラスタ・ベースのフェイルオーバーが提供されます。

  Oracle DatabaseのStandard Edition高可用性の管理を参照してください。

親トピック: このリリースでの『Oracle Database管理者ガイド』の変更点

Oracle Databaseリリース19c, バージョン19.1での変更点

Oracle Databaseリリース19c、バージョン19.1での『*Oracle Database*管理者ガイド』の変更点は次のとおりです。

- 新機能
- サポート対象外となった機能

親トピック: このリリースでの『Oracle Database管理者ガイド』の変更点

新機能

このリリースの新機能は次のとおりです。

- Oracle Databaseで提供されるユーザー・アカウントは、スキーマ専用アカウントになりました。

  Oracle Databaseで提供されるユーザー・アカウントのほとんどは、SYSおよびサンプル・スキーマを除き、スキーマ専用アカウント(つまり、これらのアカウントはパスワードなしで作成される)です。これにより、悪質なユーザーがこれらのアカウントにログインできなくなります。これらのアカウントには、認証する必要が生じたらいつでもパスワードを割り当てることができます。アカウントのステータスを確認するには、DBA_USERSデータ・ディクショナリ・ビューのACCOUNT_STATUS列を問い合せます。アカウントがスキーマ専用の場合、ステータスはNONEです。

  「事前定義のユーザー・アカウント」を参照してください。

- 自動索引作成

  自動索引作成機能では、アプリケーション・ワークロードの変化に基づいてOracleデータベース内の索引の作成、再

作成、削除などの索引管理タスクが自動化されます。この機能では、Oracleデータベース内で索引を自動的に管理することで、データベースのパフォーマンスが向上します。

「自動索引の管理」を参照してください。

- 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

  CPUやI/Oなどのシステム・リソースの過剰消費が原因でリソース・マネージャによって終了されたSQL文の実行計画に対して、隔離を構成できるようになりました。SQL文の隔離された実行計画は再実行できなくなるため、データベースのパフォーマンスの低下を防ぐことができます。

  「過剰なシステム・リソースを使用するSQL文の実行計画の隔離」を参照してください。

- Memoptimized Rowstore – 高速インジェスト

  Memoptimized Rowstoreの高速インジェスト機能では、IoT (モノのインターネット)アプリケーションなど、アプリケーションからの頻度の高い単一行データ挿入の処理が最適化されます。

  Memoptimized Rowstoreを使用する高パフォーマンス・データ・ストリーミングの有効化を参照してください。

- ハイブリッド・パーティション表

  Oracle Databaseでは、ハイブリッド・パーティション表がサポートされるようになりました。ハイブリッド・パーティション表は、一部のパーティションがデータベース内にあり一部のパーティションがデータベース外部の外部ファイル(オペレーティング・システム・ファイルやHadoop Distributed File System (HDFS)ファイルなど)内にあるパーティション表です。

  『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

  「作成する表のタイプの指定」も参照してください。

- DBCAサイレント・モードの新機能

  次に、DBCAのサイレント・モードの新機能を示します。

  - リモートPDBのクローニングによってPDBを作成する機能

    DBCAコマンド`createPluggableDatabase`の`createFromRemotePDB`パラメータを使用すると、リモートPDBをクローニングしてPDBを作成できます。

    「createPluggableDatabase」を参照してください。

  - 別のCDBにPDBを再配置する機能

    DBCAコマンド`relocatePDB`を使用すると、PDBを別のCDBに再配置できます。

    「relocatePDB」を参照してください。

  - Oracleデータベースの複製を作成する機能

    DBCAコマンド`createDuplicateDB`を使用すると、Oracleデータベースの複製を作成できます。

    「createDuplicateDB」を参照してください。

- SQL文診断可能性に関連する新機能

  SQLテスト・ケース・ビルダーやSQL修復アドバイザなどのSQL診断および修復ツールは、問題のあるSQL文を管理するための診断機能と修復機能を向上させるために拡張されました。

  - SQL修復アドバイザの新しい関数`DBMS_SQLDIAG.SQL_DIAGNOSE_AND_REPAIR`

SQL修復アドバイザの新しい関数DBMS_SQLDIAG.SQL_DIAGNOSE_AND_REPAIRは、診断タスクを作成して実行し、クリティカル・エラーを発生させているSQL文についてSQLパッチ推奨を受け入れます。

『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

「DBMS_SQLDIAGパッケージのサブプログラムを使用したSQL修復アドバイザの実行」も参照してください。

- SQLテスト・ケース・ビルダー出力を制御する新しいオプション

  DBMS_SQLDIAG.EXPORT_SQL_TESTCASEプロシージャに新しいオプション(compress、diag_eventおよびproblem_type)を指定することで、SQLテスト・ケース・ビルダーの出力を制御できるようになりました。

  「SQLテスト・ケース・ビルダーの出力」を参照してください。

- 新しいビューV$SQL_TESTCASES

  V$SQL_TESTCASESビューを問い合せることで、SQLテスト・ケース・ビルダーによって生成されたすべてのテスト・ケースに関する情報を表示できます。

  『Oracle Databaseリファレンス』を参照してください。

  「SQLテスト・ケース・ビルダーの出力」も参照してください。

- データ・ディクショナリを作成するための新しいSQLスクリプトcatpcat.sql

  CREATE DATABASE文を使用してデータベースを作成した後、データ・ディクショナリを作成するために、スクリプトcatalog.sqlおよびcatproc.sqlを実行するかわりに、新しいスクリプトcatpcat.sqlを使用してcatctl.plを実行できます。スクリプトcatpcat.sqlは、パラレル・プロセスでスクリプトcatalog.sqlおよびcatproc.sqlを実行し、それによってデータ・ディクショナリの手動作成のパフォーマンスを向上させます。

  「ステップ11: スクリプトの実行によるデータ・ディクショナリ・ビューの作成」を参照してください。

- 新しいALTER SYSTEM文のFLUSH PASSWORDFILE_METADATA_CACHE句

  データベース・パスワード・ファイルの名前または場所を変更した場合は、変更内容を有効にするために、FLUSH PASSWORDFILE_METADATA_CACHE句を指定してALTER SYSTEM文を実行する必要があります。

  「ORAPWDを使用したデータベース・パスワード・ファイルの作成」を参照してください。

  『Oracle Database SQL言語リファレンス』も参照してください。

- Oracle Database In-Memoryに対して自動的に有効になっているOracle Database Resource Manager

  INMEMORY_SIZE初期化パラメータを0より大きい値に設定することでOracle Database In-Memoryが有効になっている場合は、リソース・マネージャは自動的に有効になります。

  「Oracle Database Resource Managerの有効化とプランの切替え」を参照してください。

- 新しいビューDBA_REGISTRY_BACKPORTS

  新しいビューDBA_REGISTRY_BACKPORTSには、データベースに適用したパッチによって修正された不具合が示されます。データ・ディクショナリを変更した不具合修正のみがこのビューに示されます。

  『Oracle Databaseリファレンス』を参照してください。

親トピック: Oracle Database 19cリリース19.1での変更点

サポート対象外となった機能

次の機能は、このリリースではサポートされなくなりました。

- Oracle Multimedia

  Oracle Database 19c以降では、Oracle Multimediaはサポートされなくなりました。SecureFiles LOBにマルチメディア・コンテンツを格納し、イメージの処理および変換にサード・パーティ製品を使用することをお薦めします。

- Oracle Streams

  Oracle Database 19c以降では、Oracle Streams機能はサポートされなくなりました。Oracle GoldenGateを使用して、Oracle Streamsのすべてのレプリケーション機能を置き換えます。

親トピック: Oracle Database 19cリリース19.1での変更点

Oracle Databaseリリース18c、バージョン18.1における変更

Oracle Databaseリリース18c、バージョン18.1での『*Oracle Database*管理者ガイド』の変更点は次のとおりです。

- 新機能
- その他の変更点

親トピック: このリリースでの『Oracle Database管理者ガイド』の変更点

新機能

このリリースの新機能は次のとおりです。

- シャドウ消失書込み保護

  シャドウ消失書込み保護は、消失書込みを高速に検出して即座に応答します。シャドウ消失書込み保護を使用すれば、データ消失を最小限に抑え、データベース修復に必要な時間を短縮できます。Oracle Data Guardスタンバイ・データベースを必要とせずに、データベース、表領域またはデータファイルのシャドウ消失書込み保護を有効化できます。

  「シャドウ表領域を使用した消失書込み保護の管理」を参照してください。

- Oracle Databaseパスワード・ファイルの新しいデフォルトの場所

  Oracle Databaseのパスワード・ファイルのデフォルトの場所は、ORACLE_HOMEディレクトリではなく、ORACLE_BASEディレクトリです。この変更により、読取り専用Oracleホームがサポートされます。

  「ORAPWD構文およびコマンドライン引数の説明」を参照してください。

  「読取り専用モードのOracleホームの構成について」を参照してください。

- プライベート一時表

  プライベート一時表はメモリーに格納され、その表を作成したセッションでのみ参照できます。

  「一時表の作成」を参照してください。

- インライン外部表

  インライン外部表により、SQL文の一部として外部表の実行時定義が可能になり、データ・ディクショナリに永続オブジェクトとして外部表を作成する必要がなくなります。

「インライン外部表の使用」を参照してください。

- パラレル・キューのタイムアウト

  タイムアウトになったパラレルSQL文に対する処理は、PQ_TIMEOUT_ACTIONリソース・マネージャ・ディレクティブを設定することで指定できます。

  「パラレル・キューのタイムアウト」を参照してください。

- セッション内のSQL文の取消し

  セッション内のSQL文は、ALTER SYSTEM CANCEL SQL文を使用して取消しできます。

  「セッション内のSQL文の取消し」を参照してください。

- スケーラブルな順序

  スケーラブルな順序は、Oracle RACデータベースでデータのロード操作のパフォーマンスを向上するために使用できます。

  「順序をスケーラブルにする方法」を参照してください。

- 順序のリセット

  新しいRESTART句は、順序の採番をリセットするために、ALTER SEQUENCE文で使用できます。

  「順序の変更」を参照してください。

- Memoptimized Rowstore

  Memoptimized Rowstoreにより、主に主キー列に基づいて問合せが実行される表のデータの高速参照が可能になります。この機能は、IoT (モノのインターネット)のようなアプリケーションで特に役立ちます。

  Memoptimized Rowstoreを使用する高パフォーマンス・データ・ストリーミングの有効化を参照してください。

- 読取り専用Oracleホーム

  読取り専用モードでOracleホームを構成すると、Oracleホーム・ディレクトリORACLE_HOME内でのファイルの作成や変更を防止できます。読取り専用Oracleホームは、複数のデータベース・サーバー間で共有可能なソフトウェア・イメージとして使用できます。複数のデータベース・サーバーにパッチを配布する際に更新する必要のあるOracleホーム・イメージは1つのみになるため、パッチ適用と一括ロールアウトが簡単になります。

  「読取り専用モードのOracleホームの構成について」を参照してください。

- Oracle Automatic Storage Management (Oracle ASM)によるマルチテナント・プラガブル・データベース(PDB)のクローニング

  Oracle ASMは、PDBのクローンの作成に使用できます。

  『Oracle Multitenant管理者ガイド』を参照してください。

- プロキシ常駐接続プーリング

  プロキシ常駐接続プーリングにより、データベース・クライアントの高可用性、セキュリティおよびパフォーマンスが向上します。プロキシ常駐接続プーリングには、Traffic DirectorモードのOracle Connection Managerを使用して構成できるプロキシ常駐接続プールを使用します。

  「プロキシ常駐接続プーリングについて」を参照してください。

- Oracle DatabaseとMicrosoft Active Directoryサービスの統合

Oracle Databaseは、Microsoft Active Directoryに直接接続して、ユーザーを認証および認可できます。これにより、Oracle Enterprise User Security (EUS)とOracle Directory Serviceのフル機能の統合よりも簡単で単純なオプションが提供されます。

「ユーザーと権限の集中管理について」を参照してください。

- Active Data Guardの新しいnologging句

  データベースのnologgingは、REDOログの生成量が大幅に増加しないようにして、Active Data Guard環境で使用する際のサポートを向上するために拡張されています。次の2つの新しいnologgingモードは、既存のnologgingモードNOLOGGINGの代替として導入されました。

```
STANDBY NOLOGGING FOR DATA AVAILABILITY
STANDBY NOLOGGING FOR LOAD PERFORMANCE
```

  「FORCE LOGGINGモードのパフォーマンスに関する考慮点」を参照してください。

- DBCAサイレント・モードの新機能
  - マルチテナント・コンテナ・データベース(CDB)のコピーの作成

    CDBのコピーは、-createDuplicateDBコマンドを使用することで作成できます。

    「createDuplicateDB」を参照してください。
  - Oracle RACデータベースのコピーの作成

    Oracle RACデータベースのコピーは、-createDuplicateDBコマンドのオプション-databaseConfigTypeに値RACまたはRACONENODEを使用することで作成できます。

    「createDuplicateDB」を参照してください。
  - 既存のデータベースからのスタンバイ・データベースの作成

    スタンバイ・データベースは、-createDuplicateDBコマンドのオプション-createAsStandbyを使用することで、既存のデータベースから作成できます。

    「createDuplicateDB」を参照してください。
  - データベースの作成前のハードウェア前提条件とソフトウェア前提条件の検証

    データベースを作成する際のハードウェア前提条件とソフトウェア前提条件は、データベースを作成する必要のあるシステムで-executePrereqsコマンドを実行することで検証できます。

    「executePrereqs」を参照してください。
  - データベース・テンプレートのコピーの作成

    データベース・テンプレートのコピーは、-createTemplateFromTemplateコマンドを使用することで作成できます。

    「createTemplateFromTemplate」を参照してください。
- プラガブル・データベース(PDB)のコピーの作成

  PDBのコピーは、-createPluggableDatabaseコマンドのオプション-createPDBFromに値PDBを使用することで作成できます。

  「createPluggableDatabase」を参照してください。

親トピック: Oracle Database 18cリリース18.1での変更点

その他の変更

このリリースでの追加変更は次のとおりです。

- 新しいOracle Multitenantのマニュアル

  Oracle Multitenantのすべてのトピックは、『Oracle Database管理者ガイド』から新しいマニュアルの『Oracle Multitenant管理者ガイド』に移動されました。

- 新しいOracle Shardingのマニュアル

  Oracle Shardingのすべてのトピックは、『Oracle Database管理者ガイド』から新しいマニュアルの『Oracle Database Oracle Shardingの使用』に移動されました。

親トピック: Oracle Database 18cリリース18.1での変更点

Oracle Database 12cリリース2 (12.2)での変更点

Oracle Database 12cリリース2 (12.2)での『Oracle Database管理者ガイド』の変更点は次のとおりです。

- 新機能

親トピック: このリリースでの『Oracle Database管理者ガイド』の変更点

新機能

このリリースの新機能は次のとおりです。

- 表のオンライン再定義の改善:
  - 以前のリリースでは、BFILE列を含む表は、オンラインで再定義できませんでした。Oracle Database 12cリリース2 (12.2)では、BFILE列を含む表をオンラインで再定義できます。
  - 表の形状を変更していない表のオンライン再定義の場合は、表のロールバックを有効して、元の定義に表を戻し、表に対して行われたDMLの変更を保存できます。

    「表のオンライン再定義のロールバック」を参照してください。
  - 表のオンライン再定義中に、高速リフレッシュ可能な依存マテリアライズド・ビューをリフレッシュするには、REDEF_TABLEプロシージャまたはSTART_REDEF_TABLEプロシージャで、refresh_dep_mviewsパラメータをYに設定します。

    「表のオンライン再定義中に依存マテリアライズド・ビューをリフレッシュする方法」を参照してください。
  - DBMS_REDEFINITIONパッケージのEXECUTE_UPDATEプロシージャにより、表の一括更新のパフォーマンスを最適化します。REDOログに更新が記録されないため、パフォーマンスが最適化されます。

    「バルク更新のパフォーマンス最適化」を参照してください。
  - V$ONLINE_REDEFビューを問い合せて、表のオンライン再定義操作の進行状況を監視できます。

    「表のオンライン再定義の進行状況の監視」を参照してください。
  - 表のオンライン再定義が失敗したときは、多くの場合、失敗の原因となった問題を修正し、最後に停止したところからオンライン再定義プロセスを再開できます。

「失敗後の表のオンライン再定義の再開」を参照してください。

- リソース・マネージャによるPGA使用量の制限

  リソース・マネージャは、特定のコンシューマ・グループの各セッションに割当て可能なPGAメモリー量を制限できます。

  「プログラム・グローバル領域(PGA)」を参照してください。

- 索引圧縮の改善

  以前のリリースで使用可能な低レベルに加えて、高レベルの拡張索引圧縮を指定できます。高レベルの拡張索引圧縮は、低レベルよりも多くの領域を節約します。

  「拡張索引圧縮を使用した索引の作成」を参照してください。

- 配列の挿入のために使用可能なハイブリッド列圧縮

  配列の挿入によって挿入された行を、ハイブリッド列圧縮を使用して圧縮できます。以前のリリースでは、ダイレクト・パスINSERTにより挿入された行のみをハイブリッド列圧縮を使用して圧縮できました。

  「表圧縮について」を参照してください。

- 表の移動操作の改善

  ALTER TABLE MOVE文にONLINEキーワードが含まれている場合、移動操作中にデータ操作言語(DML)の操作がサポートされます。また、ONLINEキーワードおよびUPDATE INDEXES句が含まれている場合は、移動操作中に索引を使用できます。

  「新規セグメントまたは表領域への表の移動」を参照してください。

- 業務の分離のための新しいSYSRAC管理権限

  Oracle Databaseでは、Oracle Real Application Clusters (Oracle RAC)の操作に関連するタスクのための新しい管理権限を提供するようになりました。

  「管理権限」を参照してください。

- 実行時間の長いトランザクションのための新しいデータベース常駐接続プール・パラメータ

  MAX_THINK_TIMEパラメータに指定された制限によって、実行時間の長いトランザクションがロールバックされるのを防ぐために、DBMS_CONNECTION_POOLパッケージ内のサブプログラムのための新しいMAX_TXN_THINK_TIMEパラメータは、進行中のトランザクションを持つ任意のセッションの最大時間(秒単位)を指定します。

  「データベース常駐接続プーリングの構成パラメータ」を参照してください。

- データベース常駐接続プールの各接続の状態に関する追加情報の提供

  V$CPOOL_CONN_INFOビューに追加された新しい列は、接続プール内の各接続の現在の状態に関する詳細を提供します。

  「接続プールの接続の状態の調査」を参照してください。

- データベース・リンクの監視の向上

  新しいビューおよび提供されたPL/SQLファンクションにより、送信データベース・リンクのホスト名の判別、受信データベース・リンクの詳細の表示、および高いシステム変更番号(SCN)のアクティビティ・ソースの判別が可能になります。

  「送信データベース・リンクのホストの判断」、「受信データベース・リンクの情報の判断」および「受信データベース・リンクの高SCNアクティビティ・ソースの判断」を参照してください。

- オブジェクトの隔離

  オブジェクトの隔離は、エラーが発生したオブジェクトを隔離し、システムへの影響についてオブジェクトを監視します。

  「隔離されたオブジェクトの監視」を参照してください。

- インスタンスの中断の遅延

  INSTANCE_ABORT_DELAY_TIME初期化パラメータは、エラーが発生してインスタンスが中断されるときの遅延時間を指定します。

  「インスタンス中断の遅延」を参照してください。

- 事前作成されたプロセス

  Oracle Databaseでは、プロセスを事前に作成してクライアント接続パフォーマンスを向上できます。

  「事前作成されたプロセスの管理」を参照してください。

- パーティション化された外部表

  大量のデータがある場合は、外部表をパーティション化して、問合せのパフォーマンスを高速化し、データのメンテナンスを強化できます。

  外部表のパーティション化を参照してください。

親トピック: Oracle Database 12cリリース2 (12.2)での変更点

Oracle Database 12cリリース1 (12.1.0.2)での変更点

Oracle Database 12cリリース1 (12.1.0.2)での『Oracle Database管理者ガイド』の変更を次に示します。

- 新機能

親トピック: このリリースでの『Oracle Database管理者ガイド』の変更点

新機能

このリリースの新機能は次のとおりです。

- インメモリー列ストア

  インメモリー列ストア(IM列ストア)は、SGAのオプション領域で、表全体、表パーティション、個々の列およびマテリアライズド・ビューが圧縮列形式で格納されます。データベースでは、特殊な技法を使用して列データを超高速でスキャンします。IM列ストアはデータベースバッファ・キャッシュに代わるものではなく、補完するものです。

  「Oracle Database In-Memoryによる問合せパフォーマンスの向上」を参照してください。

- データ・ポンプによるインメモリー列ストアのサポート

  データ・ポンプを使用すると、インポートされるデータベース・オブジェクトについてインメモリー句を維持、上書き、削除できます。

  Oracle Database In-Memoryガイドを参照してください。

- 強制全データベース・キャッシュ・モード

  パフォーマンスを改善するために、強制的にインスタンスでデータベースをバッファ・キャッシュに格納させることができます。

「強制フル・データベース・キャッシュ・モードの使用方法」を参照してください。

- 大規模表キャッシュ

  自動大規模表キャッシュ機能を使用すると、パラレル問合せでバッファ・キャッシュを使用できるようになります。

  「メモリー・アーキテクチャの概要」を参照してください。

- 属性クラスタ表

  属性クラスタリングは、ディスク上に近接近でデータを格納するヒープ構成表のディレクティブを指定し、パフォーマンスおよびデータ・ストレージを向上させます。このディレクティブは、一括挿入や移動操作と同様、ダイレクト・パス操作に対してのみ適用できます。

  「属性クラスタ表の使用」を参照してください。

- ゾーン・マップ

  ゾーンとは、ディスク上の連続したデータ・ブロックのセットです。ゾーン・マップでは、個々のゾーンすべてについて、指定された列の最小値および最大値を追跡管理します。ゾーン・マップの最大の利点は、表スキャンに関するI/Oを削減することにあります。

  「ゾーン・マップの使用」を参照してください。

- 拡張索引圧縮

  拡張索引圧縮により、圧縮率が非常に高くなりますが、索引には依然として効率的にアクセスできます。拡張索引圧縮は、ブロック・レベルに作用し、各ブロックを最適に圧縮できます。つまり、ユーザーはデータの特性を認識する必要がありません。拡張索引圧縮がブロックごとに適した圧縮を自動的に選択します。

  「拡張索引圧縮を使用した索引の作成」を参照してください。

- Oracle Clusterwareによる診断フレームワークのサポート

  Oracle Clusterwareでは、診断トレース・データおよびClusterwareアラート・ログの記録に診断フレームワークおよびADRを使用します。

  「Oracle Clusterware環境でのADR」を参照してください。

- READオブジェクト権限およびREAD ANY TABLEシステム権限

  オブジェクトに対するREAD権限により、他の権限を付与しなくても、ユーザーはオブジェクトから選択できるようになります。

  詳細は、「外部表のシステム権限およびオブジェクト権限」および『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: Oracle Database 12cリリース1 (12.1.0.2)での変更点

Oracle Database 12cリリース1 (12.1.0.1)での変更点

Oracle Database 12cリリース1 (12.1.0.1)での『Oracle Database管理者ガイド』の変更を次に示します。

- 新機能
- 非推奨となった機能

親トピック: このリリースでの『Oracle Database管理者ガイド』の変更点

新機能

このリリースの新機能は次のとおりです。

- フル・トランスポータブル・エクスポート/インポート

  フル・トランスポータブル・エクスポート/インポートでは、データベースを、あるデータベース・インスタンスから別のデータベース・インスタンスに移動できます。データベースのトランスポートは、全データベースのエクスポート/インポートなどの、データベースを移動する他の方法よりはるかに高速です。また、フル・トランスポータブル・エクスポート/インポートを使用して、非CDB (またはOracle Database 11g リリース2 (11.2.0.3)データベース)をCDBに含まれるPDBに移動できます。

  「データのトランスポート」を参照してください。

- 業務の分離のための新しい管理権限

  Oracle DatabaseはOracle Recovery Manager (Oracle RMAN)、Oracle Data Guardおよび透過的データ暗号化に関連するタスクの管理権限を提供するようになりました。それぞれの新しい管理権限によって、管理の各領域におけるタスクを完了するのに必要な最小限の権限が付与されます。新しい管理権限を使用すると、数多くの一般的なタスクに対してSYSDBA管理権限を付与することを回避できます。

  「管理権限」を参照してください

- 複数のフラッシュ・デバイスのためのDatabase Smart Flash Cacheのサポート

  データベース・インスタンスは、ボリューム・マネージャを必要とすることなく、Database Smart Flash Cache用の複数のフラッシュ・デバイスにアクセスし、組み合せることができます。

  「Database Smart Flash Cacheの初期化パラメータ」を参照してください。

- 一時UNDO

  一時オブジェクトに対するUNDOは、UNDO表領域ではなく、一時表領域に格納されます。一時UNDOを使用すると、UNDO表領域に格納されるUNDOの量およびREDOログのサイズが減少します。また、Oracle Active Data Guardオプションを使用するフィジカル・スタンバイ・データベースの一時表で、データ操作言語(DML)を操作することもできます。

  「一時UNDOの管理」を参照してください。また、Oracle Data Guard環境での一時UNDOの利点の詳細は、『Oracle Data Guard概要および管理』を参照してください。

- オンラインでのデータファイルの移動

  オンラインの、アクセスされているデータファイルを移動できます。この機能により、異なる記憶デバイスへのデータの移動などのメンテナンス操作が容易になります。

  「オンライン・データファイルの名前変更と再配置」を参照してください。

- 同じ列セットに対する複数の索引

  既存の索引を削除して異なる属性を使用して再作成することなく、アプリケーションの移行を実行するために、同じ列セットに対して複数の索引を作成できます。

  「同じ列セットに対する複数の索引の作成について」を参照してください。

- オンラインのパーティションまたはサブパーティションの移動

  表のオンライン再定義を使用せずに、移動されているパーティションまたはサブパーティションでDML操作を中断すること

なく実行し続けることができます。

「新規セグメントまたは表領域への表の移動」を参照してください。

- 1つのステップでの表のオンライン再定義

  DBMS_REDEFINITIONパッケージのREDEF_TABLEプロシージャを使用して、プロシージャへの単一コールで表の記憶域プロパティのオンライン再定義を実行できます。

  「REDEF_TABLEプロシージャを使用したオンライン再定義の実行」を参照してください。

- 複数のパーティションを含む表のオンライン再定義

  表の複数のパーティションを再定義するときに停止時間を最小限に抑えるために、それらのパーティションを、オンラインの単一のセッションで再定義できます。

  「1つ以上のパーティションのオンライン再定義」を参照してください。

- 仮想プライベート・データベース(VPD)ポリシーを含む表のオンライン再定義

  停止時間を最小限に抑えるために、VPDポリシーを含む表をオンラインで再定義できます。

  「オンライン再定義時の仮想プライベート・データベース(VPD)ポリシーの処理」を参照してください。

- FINISH_REDEF_TABLEプロシージャの新しいtime limitパラメータ

  DBMS_REDEFINITIONパッケージのFINISH_REDEF_TABLEプロシージャのdml_lock_timeoutパラメータで、保留中のDMLのコミットをプロシージャが待機する時間を指定できます。

  「DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行」のステップ8を参照してください。

- 非表示の列

  表の個々の列を不可視にできます。表の一般的なアクセスでは、表の不可視の列は表示されません。

  「不可視の列の理解」を参照してください。

- NULL値可能列のデフォルト値で最適化されたALTER TABLE...ADD COLUMN

  NULL値可能列は、NOT NULL制約を使用せずに作成された列です。特定のタイプの表では、デフォルト値を持つNULL値可能列を追加するときに、データベースは操作のリソース使用率および記憶域要件を最適化できます。このことは、既存のすべてのレコードの値を格納する必要がないように、新しい列のデフォルト値を表メタデータとして格納することによって行われます。

  「表の列の追加」を参照してください。

- CloneDBを使用したデータベースのcopy-on-writeクローニング

  CloneDBを使用してデータベースをクローニングする場合、Oracle Databaseはcopy-on-writeテクノロジに基づいてCloneDBデータベースにファイルを作成できるため、ディスク上に追加の記憶域が必要になるのは、CloneDBデータベースで変更されたブロックに対してのみになります。

  「CloneDBを使用したデータベースのクローニング」を参照してください。

- DDLログ

  DDL文のロギングが有効な場合、DDL文はアラート・ログではなく別のDDLログに記録されます。

  「DDLログ」を参照してください。

- デバッグ・ログ

  問題のデバッグに使用できる情報は、アラート・ログではなく別のデバッグ・ログに記録されます。

  「デバッグ・ログ」を参照してください。

- サーバー制御(SRVCTL)ユーティリティの完全単語オプション

  ユーザビリティを向上させるため、SRVCTLユーティリティの各オプションが、単一文字ではなく完全な単語になりました。

  「Oracle RestartのSRVCTLコマンド・リファレンス」を参照してください。

- トランザクション・ガードおよびアプリケーション・コンティニュイティ

  アプリケーションを重複するトランザクションの発行および関連する論理エラーから保護するため、トランザクション・ガードによってトランザクションの最大1回実行が保証されます。トランザクション・ガードにより、リカバリ可能な通信エラーの後にトランザクションをリプレイして処理を続行する機能であるアプリケーション・コンティニュイティが使用可能になります。

  「トランザクション・ガードおよびアプリケーション・コンティニュイティ」を参照してください。

- 文のキューイングの拡張

  重要な文はパラレル文のキューを無視できます。優先度の高いコンシューマ・グループに対してリソース・プラン・ディレクティブPARALLEL_STMT_CRITICALをBYPASS_QUEUEに設定し、そのコンシューマ・グループのパラレル・ステートメントがパラレル・ステートメント・キューを無視するように設定できます。

  「リソース・プラン・ディレクティブの作成」を参照してください。

- 新しいジョブ・タイプ

  SQL*Plus、RMANインタプリタまたはコンピュータ・プラットフォームのコマンド・シェルを使用してカスタム・ユーザー・スクリプトの実行を許可する複数の新しいスクリプト・ジョブが追加されました。

  「スクリプト・ジョブ」を参照してください。

親トピック: Oracle Database 12cリリース1 (12.1.0.1)での変更点

非推奨となった機能

次の機能は、今回のリリースで非推奨であり、将来のリリースでサポートされなくなる可能性があります。

- ORAPWDのIGNORECASE引数

  厳密認証をサポートするために、IGNORECASEをnに設定するか、IGNORECASEを省略することをお薦めします。このオプションのORAPWD引数のデフォルト値はnです。

  詳細は、「ORAPWDを使用したデータベース・パスワード・ファイルの作成」を参照してください。

- サーバー制御(SRVCTL)ユーティリティ・コマンドを使用した単一文字オプション

  すべてのSRVCTLコマンドは、単一文字オプションのかわりに完全単語オプションを受け入れるように拡張されました。このリリースで追加されたすべての新規SRVCTLコマンド・オプションは完全単語オプションのみをサポートしており、等価の単一文字はありません。SRVCTLコマンドでの単一文字オプションの使用は、将来のリリースではサポートされなくなる可能性があります。

  詳細は、「Oracle RestartのSRVCTLコマンド・リファレンス」を参照してください。

- FILE_MAPPING初期化パラメータ

FILE_MAPPING初期化パラメータは非推奨です。これは、下位互換性を保つためにのみサポートされています。

FILE_MAPPING初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

- *_SCHEDULER_CREDENTIALS

  このビューは、下位互換性のために引き続き使用可能です。

  詳細は、「スケジューラ・ジョブ資格証明の指定」を参照してください。

関連項目:

Oracle Databaseアップグレード・ガイド

親トピック: Oracle Database 12cリリース1 (12.1.0.1)での変更点

第I部 基本データベース管理

データベース管理者は特定の職責を持ち、データベース管理タスクの実行方法を理解する必要があります。

- データベース管理スタート・ガイド
  データベース管理を開始するには、データベース・ユーザーのタイプ、データベース・セキュリティ、権限など、データベースの基本概念を理解する必要があります。データベースへのコマンドとSQLの送信やパスワード・ファイルの作成などの基本タスクを実行できる必要もあります。
- Oracle Databaseの作成および構成
  データベースを計画した後で、グラフィカル・ツールまたはSQLコマンドを使用してデータベースを作成できます。
- 起動と停止
  データベースの起動時には、そのデータベースのインスタンスを作成し、データベースの状態を確認します。現在実行中のOracle Databaseインスタンスの停止では、必要に応じて、データベースをクローズおよびディスマウントできます。
- Oracle Databaseの自動再起動の構成
  Oracle Restart機能を使用してOracleデータベースを構成すると、ハードウェアまたはソフトウェアに障害が発生した後やデータベース・ホスト・コンピュータが再起動した場合は常に、データベース、リスナーおよびその他のOracleコンポーネントを自動的に再起動できます。
- プロセスの管理
  Oracle Databaseは、いくつかのプロセスを使用して、複数のユーザーとアプリケーションが単一のデータベース・インスタンスに同時に接続できるようにします。
- メモリーの管理
  メモリー管理には、データベースの変更に応じたOracle Databaseインスタンス・メモリー構造の最適なサイズのメンテナンスが含まれます。
- ユーザーの管理とデータベースのセキュリティ保護
  すべてのデータベースのセキュリティ・ポリシーを設定します。
- データベースの監視
  データベースの動作を定期的に監視することは重要です。監視することによって、気付いていないエラーの情報を取得できるのみでなく、データベースの正常な動作について理解を深めることもできます。正常な動作を理解しておくことで、なんらかの誤りがある場合に状況を容易に認識できるようになります。
- 問題の診断と解決
  Oracle Databaseには、データベースの問題を診断して解決するために、診断データの収集と管理ための高度な障害診断インフラストラクチャが含まれています。診断データには、以前のリリースにも含まれていたトレース・ファイル、ダンプおよびコア・ファイルに加えて、顧客やOracleサポート・サービスが問題を迅速かつ効率的に識別、調査、追跡、解決できる新しいタイプの診断データが含まれています。

1 データベース管理スタート・ガイド

データベース管理を開始するには、データベース・ユーザーのタイプ、データベース・セキュリティ、権限など、データベースの基本概念を理解する必要があります。データベースへのコマンドとSQLの送信やパスワード・ファイルの作成などの基本タスクを実行できる必要もあります。

- Oracle Databaseユーザーのタイプ
  ユーザーのタイプとその役割および責任は、データベース・サイトによって異なります。小規模のサイトでは、1名のデータベース管理者を配置して、アプリケーション開発者およびユーザー向けのデータベースを管理できます。大規模なサイトでは、データベース管理者の役割を複数の人および複数の専門グループに分割する必要があります。
- データベース管理者のタスク
  Oracle Databaseを設計、実装およびメンテナンスするには、いくつかの特定のタスクを完了する必要があります。
- SQL文
  Oracle Databaseとのやり取りは主にSQL文を発行して行います。
- Oracle Databaseソフトウェアのリリースの識別
  リリースを完全に識別するには、5つもの番号が必要な場合があります。
- データベース管理者のセキュリティと権限について
  Oracle Database管理者の管理タスクを実行するには、データベース内部、およびお場合によってはそのデータベースが稼働するサーバーのオペレーティング・システムで特定の権限が必要です。データベース管理者のアカウントへのアクセスは厳しく管理する必要があります。
- データベース管理者の認証
  DBAは、データベースの起動や停止などの特別な操作を実行します。これらの操作はDBAのみが実行するため、DBAのユーザー名には安全性の高い認証方式が必要です。
- データベース・パスワード・ファイルの作成とメンテナンス
  パスワード・ファイル作成ユーティリティORAPWDを使用して、データベース・パスワード・ファイルを作成できます。一部のオペレーティング・システムでは、標準インストール時にこのファイルを作成できます。
- データ・ユーティリティ
  Oracle Databaseのデータのメンテナンスに役立つOracleユーティリティが用意されています。

親トピック: 基本データベース管理

1.1 Oracle Databaseユーザーのタイプ

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

- データベース管理者
  各データベースには、少なくとも1名のデータベース管理者(DBA)が必要です。Oracle Databaseシステムは規模が大きく、多数のユーザーによって使用される可能性があります。したがって、1名の担当者ではデータベースを管理できないため、複数のDBAでグループを編成して役割を分担します。
- セキュリティ管理者
  サイトによっては、データベースに1名以上のセキュリティ管理者が必要です。セキュリティ管理者は、ユーザーの登録、データベースに対するユーザー・アクセスの制御と監視およびシステム・セキュリティのメンテナンスを実施します。
- ネットワーク管理者

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

- アプリケーション開発者
  アプリケーション開発者は、データベース・アプリケーションを設計し、実装します。
- アプリケーション管理者
  Oracle Databaseのサイトでは、特定のアプリケーションを管理するために1名以上のアプリケーション管理者が必要な場合があります。アプリケーションごとに専門の管理者を置く場合があります。
- データベース・ユーザー
  データベース・ユーザーは、アプリケーションまたはユーティリティを介してデータベースと対話します。

親トピック: データベース管理スタート・ガイド

1.1.1 データベース管理者

各データベースには、少なくとも1名のDBAが必要です。Oracle Databaseシステムは規模が大きく、多数のユーザーによって使用される可能性があります。したがって、1名の担当者ではデータベースを管理できないため、複数のDBAでグループを編成して役割を分担します。

データベース管理者が担当するタスクは次のとおりです。

- Oracle Databaseサーバーとアプリケーション・ツールをインストールおよびアップグレードします。
- データベース・システムにシステム記憶域を割り当て、将来の記憶域要件を計画します。
- アプリケーション開発者がアプリケーションを設計した後、プライマリ・データベースの記憶域構造(表領域)を作成します。
- アプリケーション開発者がアプリケーションを設計した後、プライマリ・オブジェクト(表、ビュー、索引)を作成します。
- アプリケーション開発者から得た情報に基づき、必要に応じてデータベース構造を修正します。
- ユーザーを登録し、システム・セキュリティをメンテナンスします。
- Oracleのライセンス契約に従っていることを確認します。
- データベースに対するユーザー・アクセスを制御し、監視します。
- データベースのパフォーマンスを監視し、最適化します。
- データベース情報のバックアップおよびリカバリの計画を立てます。
- テープ上のアーカイブ済データをメンテナンスします。
- データベースをバックアップおよびリストアします。
- 技術サポートについてOracleサポート・サービスに連絡します。

親トピック: Oracle Databaseユーザーのタイプ

1.1.2 セキュリティ管理者

サイトによっては、データベースに1名以上のセキュリティ管理者が必要です。セキュリティ管理者は、ユーザーの登録、データベースに対するユーザー・アクセスの制御と監視およびシステム・セキュリティのメンテナンスを実施します。

したがって、サイトにセキュリティ管理者が別にいる場合、DBAはこれらの業務に対する役割を持ちません。

セキュリティ管理者の業務の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: Oracle Databaseユーザーのタイプ

1.1.3 ネットワーク管理者

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

ネットワーク管理者の業務の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

関連項目:

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

親トピック: Oracle Databaseユーザーのタイプ

1.1.4 アプリケーション開発者

アプリケーション開発者は、データベース・アプリケーションを設計し、実装します。

アプリケーション開発者が担当するタスクは、次のとおりです。

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

アプリケーション開発者は、これらのタスクの一部をDBAと協力して実施する場合があります。アプリケーション開発タスクの詳細は、『Oracle Database開発ガイド』を参照してください。

親トピック: Oracle Databaseユーザーのタイプ

1.1.5 アプリケーション管理者

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

親トピック: Oracle Databaseユーザーのタイプ

1.1.6 データベース・ユーザー

データベース・ユーザーは、アプリケーションまたはユーティリティを介してデータベースと対話します。

一般ユーザーが担当するタスクは、次のとおりです。

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

1.2 データベース管理者のタスク

Oracle Databaseを設計、実装およびメンテナンスするには、いくつかの特定のタスクを完了する必要があります。

ノート:

新しいリリースにアップグレードする場合は、既存の本番環境(ソフトウェアとデータベースの両方)のバックアップを作成してからインストールしてください。既存の本番データベースの保存方法については、『Oracle Database アップグレード・ガイド』を参照してください。

- タスク1: データベース・サーバー・ハードウェアの評価
  使用可能なコンピュータ・リソースを、Oracle Databaseとそのアプリケーションで最大限に活用するための評価を行います。
- タスク2: Oracle Databaseソフトウェアのインストール
  データベース管理者は、Oracle Databaseサーバーのソフトウェアとすべてのフロントエンド・ツール、およびデータベースにアクセスするデータベース・アプリケーションをインストールします。
- タスク3: データベースの計画
  データベース管理者は、データベースの論理記憶域構造、全体的なデータベース設計、データベースのバックアップ戦略を計画する必要があります。
- タスク4: データベースの作成とオープン
  データベース設計が完了した後、データベースを作成し、通常使用のためにオープンできます。
- タスク5: データベースのバックアップ
  データベース構造の作成後、データベースに対して予定していたバックアップ計画を実行します。
- タスク6: システム・ユーザーの登録
  データベース構造のバックアップが完了した後、Oracleのライセンス契約に従ってデータベースのユーザーを登録し、登録したユーザーに対して適切な権限とロールを付与できます。
- タスク7: データベース設計の実装
  データベースを作成して起動し、システム・ユーザーを登録した後、必要なすべての表領域を作成して、計画したデータベースの論理構造を実装できます。表領域の作成を完了すると、データベース・オブジェクトを作成できます。
- タスク8: 実行データベースのバックアップ
  データベースがすべて実装されたときに、再度バックアップを作成します。定期的にバックアップを作成するようにスケジュールし、また、データベース構造の変更を実装した直後にも必ずデータベースのバックアップを作成するようにします。
- タスク9: データベースのパフォーマンス・チューニング
  データベースのパフォーマンスの最適化は、DBAの日常的な作業の1つです。Oracle Databaseでは、各種ユーザー・グループへのリソースの割当てを制御できるよう、データベース・リソース管理機能が提供されています。
- タスク10: リリース更新とリリース更新リビジョンのダウンロードとインストール
  データベースのインストール後、定期的に、Oracleソフトウェアのリリース更新(Update)とリリース更新リビジョン(Revision)をダウンロードしてインストールします。
- タスク11: 追加ホストへのロール・アウト
  Oracle Databaseのインストールを正しく構成し、チューニングし、パッチを適用してテストした後、そのインストールを他のホストにロール・アウトする必要がある場合があります。

親トピック: データベース管理スタート・ガイド

1.2.1 タスク1: データベース・サーバー・ハードウェアの評価

使用可能なコンピュータ・リソースを、Oracle Databaseとそのアプリケーションで最大限に活用するために評価します。

この評価では、次のような情報を明らかにする必要があります。

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

親トピック: データベース管理者のタスク

1.2.2 タスク2: Oracle Databaseソフトウェアのインストール

データベース管理者は、Oracle Databaseサーバーのソフトウェアとすべてのフロントエンド・ツール、およびデータベースにアクセスするデータベース・アプリケーションをインストールします。

分散処理環境インストールでは、データベースが中核になるコンピュータ(データベース)によって制御され、データベース・ツールとアプリケーションがリモート・マシン(クライアント)で実行される場合があります。このような場合は、Oracle Databaseを実行するコンピュータに、リモート・システムを接続するために必要なOracle Netコンポーネントもインストールする必要があります。

インストールするソフトウェアの詳細は、「Oracle Databaseソフトウェアのリリースの識別」を参照してください。

関連項目:

インストールの特定の要件や指示の詳細は、次のマニュアルを参照してください。

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

親トピック: データベース管理者のタスク

1.2.3 タスク3: データベースの計画

データベース管理者は、データベースの論理記憶域構造、全体的なデータベース設計、データベースのバックアップ戦略を計画する必要があります。

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

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

データベース・オブジェクトのリレーショナル設計と、各オブジェクトの記憶特性について計画します。オブジェクトを作成する前に、

オブジェクトと各オブジェクトの物理記憶域間の関連について計画を立てることによって、1つの単位としてのデータベースのパフォーマンスを直接制御できます。データベースの拡張計画についても必ず検討してください。

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

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

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

データベースの論理記憶域構造、オブジェクトおよび整合性制約の作成方法は、「Oracle Databaseの構造と記憶域」および「スキーマ・オブジェクト」を参照してください。

親トピック: データベース管理者のタスク

1.2.4 タスク4: データベースの作成とオープン

データベース設計が完了すると、データベースを作成し、オープンできます。

データベースを作成するには、Database Configuration Assistantを使用してインストール時に作成する方法と、データベースを作成するためのスクリプトを用意する方法があります。

データベースの作成方法については「Oracle Databaseの作成および構成」を、データベース起動時のガイドラインについては「起動と停止」を参照してください。

親トピック: データベース管理者のタスク

1.2.5 タスク5: データベースのバックアップ

データベース構造の作成後、データベースに対して予定していたバックアップ計画を実行します。

追加のREDOログ・ファイルを作成し、データベース全体の最初のバックアップ(オンラインまたはオフライン)を作成して、その後の定期的なデータベース・バックアップをスケジュールします。

関連項目:

Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド

親トピック: データベース管理者のタスク

1.2.6 タスク6: システム・ユーザーの登録

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

このタスクのガイドラインについては、「ユーザーの管理とデータベースのセキュリティ保護」を参照してください。

親トピック: データベース管理者のタスク

1.2.7 タスク7: データベース設計の実装

データベースを作成して起動し、システム・ユーザーを登録した後、必要なすべての表領域を作成して、計画したデータベースの

論理構造を実装できます。表領域の作成を完了すると、データベース・オブジェクトを作成できます。

データベースの論理記憶域構造とオブジェクトの作成方法は、「Oracle Databaseの構造と記憶域」および「スキーマ・オブジェクト」を参照してください。

親トピック: データベース管理者のタスク

1.2.8 タスク8: 実行データベースのバックアップ

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

親トピック: データベース管理者のタスク

1.2.9 タスク9: データベースのパフォーマンス・チューニング

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

データベース・リソース・マネージャについては、「Oracle Database Resource Managerを使用したリソースの管理」を参照してください。

関連項目:

データベースとアプリケーションのチューニング方法の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: データベース管理者のタスク

1.2.10 タスク10: リリース更新とリリース更新リビジョンのダウンロードとインストール

データベースのインストール後、定期的に、Oracleソフトウェアのリリース更新(Update)とリリース更新リビジョン(Revision)をダウンロードしてインストールします。

Oracle Database18c以降では、リリース更新(Update)およびリリース更新リビジョン(Revision)の形式で四半期ごとに更新が提供されています。パッチ・セットはリリースされなくなりました。インストールに必要な更新は、My Oracle SupportのWebサイトで確認してください。

関連項目:

- リリース更新(Update)とリリース更新リビジョン(Revision)をダウンロードしてインストールする手順は、使用しているプラットフォームのOracle Databaseインストレーション・ガイドを参照してください。
- My Oracle Supportノート2285040.1

親トピック: データベース管理者のタスク

1.2.11 タスク11: 追加ホストへのロール・アウト

Oracle Databaseのインストールを正しく構成し、チューニングし、パッチを適用してテストした後、そのインストールを他のホストにロール・アウトする必要がある場合があります。

ロール・アウトは、次のような場合に行います。

- 複数の本番データベース・システムがある場合。
- 本番システムと同じ開発システムやテスト・システムを作成する場合。

追加の各ホストでインストール、チューニングおよびパッチ適用を実行するかわりに、テスト済のOracle Databaseインストールを他のホストにクローニングすると、時間を節約でき、不整合をなくすことができます。次の2種類のクローニングを使用できます。

- Oracleホームのクローニング—Oracleホーム・ディレクトリとサブディレクトリの構成済およびパッチ適用済バイナリのみが宛先のホストにコピーされ、新しい環境にあわせて調整されます。その後、このクローニングしたホームを使用してインスタンスを起動し、データベースを作成できます。

  Oracle Enterprise Manager Cloud Controlを使用すると、Oracleホームを1つ以上の宛先ホストにクローニングできます。提供されているスクリプトのセットとOracle Universal Installerを使用して、Oracleホームを手動でクローニングできます。

- データベースのクローニング—データベース・ファイルや初期化パラメータなどのチューニング済のデータベースが、既存のOracleホーム(おそらくクローニングしたホーム)にクローニングされます。

  Cloud Controlを使用すると、Oracle Databaseインスタンスを既存のOracleホームにクローニングできます。

関連項目:

- Oracle Enterprise Manager Cloud管理ガイド
- Oracle Enterprise Managerライフサイクル管理ガイド
- Cloud Controlオンライン・ヘルプ
- 「CloneDBを使用したデータベースのクローニング」
- Oracle Multitenant管理者ガイド

親トピック: データベース管理者のタスク

1.3 SQL文

Oracle Databaseとのやり取りは主にSQL文を発行して行います。

- データベースに対するコマンドとSQLの発行
  SQL文やコマンドをOracle Databaseに発行する方法は複数あります。
- SQL*Plusについて
  SQL*PlusはOracle Databaseへの主要なコマンドライン・インタフェースです。SQL*Plusは、データベースの起動と停止、データベース初期化パラメータの設定、ユーザーの作成と管理、データベース・オブジェクト(表や索引など)の作成と変更、データの挿入と更新、SQL問合せの実行などを行うために使用します。
- SQL*Plusを使用したデータベースへの接続
  SQL*Plusを使用してOracle Databaseインスタンスに接続します。

親トピック: データベース管理スタート・ガイド

1.3.1 データベースに対するコマンドとSQLの発行

これらのSQL文やコマンドをOracle Databaseに発行する方法は複数あります。

- SQL*Plusのコマンドライン・インタフェースを使用する直接的な方法
- Oracle Enterprise Manager Database Express (EM Express)やOracle Enterprise Manager Cloud Control (Cloud Control)などのグラフィカル・ユーザー・インタフェース使用する間接的な方法

  これらのツールを使用すると、使いやすいグラフィカル・インタフェースを使用してデータベースを管理できます(SQL文やコマンドは、ツールが内部的に発行します)。

  詳細は、『Oracle Database 2日でデータベース管理者』およびツールのオンライン・ヘルプを参照してください。
- SQL Developerを使用する直接的な方法

  開発者はSQL Developerを使用してデータベース・スキーマとアプリケーションの作成およびテストをしますが、SQL Developerはデータベースの管理作業にも使用できます。

  詳細は、『Oracle SQL Developerユーザーズ・ガイド』を参照してください。

Oracle Databaseでは、データベースの起動と停止およびデータベース設定の変更などのコマンドを含む、SQLのスーパーセットもサポートされています。

親トピック: SQL文

1.3.2 SQL*Plusについて

SQL*PlusはOracle Databaseへの主要なコマンドライン・インタフェースです。SQL*Plusは、データベースの起動と停止、データベース初期化パラメータの設定、ユーザーの作成と管理、データベース・オブジェクト(表や索引など)の作成と変更、データの挿入と更新、SQL問合せの実行などを行うために使用します。

SQL文とコマンドを発行する前に、データベースに接続する必要があります。SQL*Plusでは、ローカル接続またはリモート接続が可能です。ローカル接続とは、SQL*Plusを実行しているのと同じコンピュータで動作しているOracle Databaseに接続することです。リモート接続とは、リモート・コンピュータで動作しているOracle Databaseにネットワークを介して接続することです。そのようなデータベースはリモート・データベースと呼ばれます。Oracle Databaseの完全インストール、Oracle Clientのインストールまたは Instant Clientのインストールを行うと、ローカル・コンピュータにSQL*Plus実行可能ファイルが提供されます。

関連項目:

SQL*Plusユーザーズ・ガイドおよびリファレンス

親トピック: SQL文

1.3.3 SQL*Plusを使用したデータベースへの接続

SQL*Plusを使用してOracle Databaseインスタンスに接続します。

- SQL*Plusを使用したデータベースへの接続について
  Oracle Databaseには、プロセスとメモリーのコレクションであるOracle Databaseインスタンスと、ユーザー・データとシステム・データを含むディスク・ファイルのセットのコンポーネントが含まれます。
- ステップ1: コマンド・ウィンドウのオープン
  プラットフォームで必要な処理を実行して、オペレーティング・システムのコマンドを入力できるウィンドウを開きます。
- ステップ2: オペレーティング・システムの環境変数の設定
  プラットフォームによっては、SQL*Plusを起動する前に、環境変数を設定するか少なくとも正しく設定されていることを確認することが必要な場合があります。

- ステップ3: SQL*Plusの起動
  Oracle Databaseに接続するには、次のいずれかのオプションを使用してSQL*Plusを起動します。
- ステップ4: SQL*PlusのCONNECTコマンドの発行
  最初にOracle Databaseインスタンスに接続する、または任意の時点で別のユーザーとして再接続するには、SQL*PlusのCONNECTコマンドを発行します。

親トピック: SQL文

1.3.3.1 SQL*Plusを使用したデータベースへの接続について

Oracle Databaseには次のコンポーネントが含まれています。プロセスとメモリーの集まりであるOracle Databaseインスタンス、およびユーザー・データとシステム・データを含むディスク・ファイルのセットです。

それぞれのインスタンスには、システムID(SID)とも呼ばれるインスタンスIDがあります。ホスト・コンピュータ上に複数のOracleインスタンスが存在し、各インスタンスがそれぞれのデータファイル・セットを持つ可能性があるため、接続先のインスタンスを特定する必要があります。ローカル接続の場合は、オペレーティング・システムの環境変数を設定してインスタンスを識別します。リモート接続では、ネットワーク・アドレスとデータベース・サービス名を指定して、インスタンスを識別します。ローカル接続とリモート接続のいずれの場合も、環境変数を設定して、SQL*Plusの実行可能ファイルをオペレーティング・システムが検出できるようにし、実行可能ファイルにサポート・ファイルとスクリプトへのパスを設定する必要があります。

関連項目:

Oracleインスタンスに関するバックグラウンド情報は、『Oracle Database概要』を参照してください。

親トピック: SQL*Plusを使用したデータベースへの接続

1.3.3.2 ステップ1: コマンド・ウィンドウのオープン

プラットフォームで必要な処理を実行して、オペレーティング・システムのコマンドを入力できるウィンドウを開きます。

- コマンド・ウィンドウを開きます。

親トピック: SQL*Plusを使用したデータベースへの接続

1.3.3.3 ステップ2: オペレーティング・システムの環境変数の設定

プラットフォームによっては、SQL*Plusを起動する前に、環境変数を設定するか少なくとも正しく設定されていることを確認することが必要な場合があります。

たとえば、ほとんどのプラットフォームでは、環境変数ORACLE_SIDおよびORACLE_HOMEを設定する必要があります。さらに、ORACLE_HOME/binディレクトリを含めるようにPATH環境変数を構成する必要があります。一部のプラットフォームでは、さらに環境変数の設定が必要な場合があります。

- UnixおよびLinuxでは、必要に応じてオペレーティング・システムのコマンドを入力して環境変数を設定します。
- Microsoft Windowsでは、インストーラはWindowsレジストリ内のORACLE_HOMEとORACLE_SIDに値を自動的に割り当てます。LD_LIBRARY_PATHを変更します

インストール時にデータベースを作成しなかった場合、インストーラはレジストリ内のORACLE_SIDを設定しないため、後でデータベースを作成するときに、ORACLE_SID環境変数をコマンド・ウィンドウから設定する必要があります。

UnixおよびLinuxのインストールには、oraenvとcoraenvの2つのスクリプトが含まれています。これらのスクリプトを使用すると、環境変数を容易に設定できます。

いずれのプラットフォームでも、異なるOracleホームを使用するインスタンス間で切り替える場合は、ORACLE_HOME環境変数を変更する必要があります。同じOracleホームを複数のインスタンスで共有している場合は、インスタンスを切り替えるときにORACLE_SIDのみを変更する必要があります。

例1-1 Unixの環境変数の設定(Cシェル)

```
setenv ORACLE_SID orcl
setenv ORACLE_HOME /u01/app/oracle/product/database_release_number/dbhome_1
setenv LD_LIBRARY_PATH
$ORACLE_HOME/lib:/usr/lib:/usr/dt/lib:/usr/openwin/lib:/usr/ccs/lib
```

例1-2 Linuxの環境変数の設定(Bashシェル)

```
export ORACLE_SID=orcl
export ORACLE_HOME=/u01/app/oracle/product/database_release_number/dbhome_1
export
LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib:/usr/dt/lib:/usr/openwin/lib:/usr/ccs/lib
```

例1-3 Microsoft Windowsの環境変数の設定

```
SET ORACLE_SID=orawin2
setenv LD_LIBRARY_PATH
$ORACLE_HOME/lib:/usr/lib:/usr/dt/lib:/usr/openwin/lib:/usr/ccs/lib:$LD_LIBRARY_PATH
```

このMicrosoft Windowsの例では、ORACLE_HOMEおよびORACLE_SIDがレジストリに設定されているが、別のインスタンスに接続するにはORACLE_SIDのレジストリ値を上書きする必要があると仮定しています。

Microsoft Windowsでは、コマンド・プロンプト・ウィンドウで設定した環境変数値でレジストリの値が上書きされます。

親トピック: SQL*Plusを使用したデータベースへの接続

1.3.3.4 ステップ3: SQL*Plusの起動

Oracle Databaseに接続するには、次のいずれかのオプションを使用してSQL*Plusを起動します。

1. 次のいずれかを行います:
   - PATH環境変数に$ORACLE_HOME/binが含まれていることを確認します。
   - ディレクトリを$ORACLE_HOME/binに変更します。PATH環境変数にドット(".")が含まれていることを確認します。
2. 次のコマンドを入力します(UnixとLinuxでは大/小文字が区別されます)。

   ```
   sqlplus /nolog
   ```

   完全なパスを指定してsqlplusコマンドを実行することもできます。

   ```
   $ORACLE_HOME/bin/sqlplus /nolog
   ```

親トピック: SQL*Plusを使用したデータベースへの接続

1.3.3.5 ステップ4: SQL*PlusのCONNECTコマンドの発行

最初にOracle Databaseインスタンスに接続する、または任意の時点で別のユーザーとして再接続するには、SQL*PlusのCONNECTコマンドを発行します。

- SQL*Plusで、CONNECTコマンドを発行します。

例1-4 ローカル・データベース・ユーザーへの接続

次の簡単な例では、ユーザーSYSTEMでローカル・データベースに接続します。SQL*Plusによって、ユーザーSYSTEMのパスワードの入力が求められます。

```
connect system
```

例1-5 SYSDBA権限でのローカル・データベース・ユーザーへの接続

次の例では、SYSDBA権限のユーザーSYSでローカル・データベースに接続します。SQL*Plusによって、ユーザーSYSのパスワードの入力が求められます。

```
connect sys as sysdba
```

ユーザーSYSとして接続するときには、AS SYSDBAを指定して接続する必要があります。

例1-6 SYSBACKUP権限でのローカル・データベース・ユーザーへの接続

次の例では、SYSBACKUP権限のユーザーSYSBACKUPでローカル・データベースに接続します。SQL*Plusによって、ユーザーSYSBACKUPのパスワードの入力が求められます。

```
connect sysbackup as sysbackup
```

ユーザーSYSBACKUPとして接続するときには、AS SYSBACKUPを指定して接続する必要があります。

例1-7 オペレーティング・システム認証によるSYSDBA権限でのローカルな接続

次の例では、オペレーティング・システム認証を使用してSYSDBA権限でローカルに接続します。

```
connect / as sysdba
```

例1-8 簡易接続構文による接続

この例では、簡易接続の構文を使用し、ユーザーsalesadminとして、ホストdbhost.example.comで動作するリモート・データベースに接続します。Oracle Netリスナー(リスナー)は、デフォルト・ポート(1521)でリスニングしています。データベース・サービスは、sales.example.comです。SQL*Plusによって、ユーザーsalesadminのパスワードの入力が求められます。

```
connect salesadmin@"dbhost.example.com/sales.example.com"
```

例1-9 サービス・ハンドラ・タイプを指定した簡易接続構文による接続

この例は、サービス・ハンドラ・タイプが指定されていることを除き、簡易接続構文によるデータベースへの接続と同じです。

```
connect salesadmin@"dbhost.example.com/sales.example.com:dedicated"
```

例1-10 デフォルト以外のリスナー・ポートを使用した簡易接続構文による接続

この例は、デフォルト以外のポート番号1522でリスナーがリスニングしていることを除き、簡易接続構文によるデータベースへの接続と同じです。

```
connect salesadmin@"dbhost.example.com:1522/sales.example.com"
```

例1-11 ホストIPアドレスを使用した簡易接続構文による接続

この例は、ホスト名のかわりにホストのIPアドレスが使用されている点を除き、簡易接続構文によるデータベースへの接続と同じです。

```
connect salesadmin@"192.0.2.5/sales.example.com"
```

例1-12 IPv6アドレスによる接続

この例では、IPv6アドレスを使用して接続します。角括弧で囲まれている点に注意してください。

```
connect salesadmin@"[2001:0DB8:0:0::200C:417A]/sales.example.com"
```

例1-13 インスタンスの指定による接続

次の例では、接続先のインスタンスを指定し、データベース・サービス名を省略します。インスタンスのみを指定する場合はサービス・ハンドラのタイプを指定できないことに注意してください。

```
connect salesadmin@"dbhost.example.com//orcl"
```

例1-14 ネット・サービス名による接続

次の例では、ユーザーsalesadminとして、ネット・サービス名sales1によって指定されたデータベース・サービスにリモート接続します。SQL*Plusによって、ユーザーsalesadminのパスワードの入力が求められます。

```
connect salesadmin@sales1
```

例1-15 外部認証による接続

次の例では、外部認証を使用して、ネット・サービス名sales1によって指定されたデータベース・サービスにリモート接続します。

```
connect /@sales1
```

例1-16 SYSDBA権限と外部認証による接続

次の例では、外部認証を使用してSYSDBA権限で、ネット・サービス名sales1によって指定されたデータベース・サービスにリモート接続します。

```
connect /@sales1 as sysdba
```

例1-17 サービス名を使用したユーザーとしての接続

次の例では、ユーザーsalesadminとして、ネット・サービス名sales1によって指定されたデータベース・サービスにリモート接続します。データベース・セッションは、rev21エディションで開始されます。SQL*Plusによって、ユーザーsalesadminのパスワードの入力が求められます。

```
connect salesadmin@sales1 edition=rev21
```

ノート:

SYSDBA 権限を持つユーザーとしてデータベースに接続しているときに問題が発生した場合は、My Oracle Support ノート 69642.1、233223.1、18089.1 および 747456.1 を参照してください。

- SQL*PlusのCONNECTコマンドの構文
  Oracleインスタンスに初めて接続するか、Oracleインスタンスに再接続するには、SQL*Plus CONNECTコマンドを使用します。

親トピック: SQL*Plusを使用したデータベースへの接続

1.3.3.5.1 SQL*PlusのCONNECTコマンドの構文

Oracleインスタンスに初めて接続するか、Oracleインスタンスに再接続するには、SQL*Plus CONNECTコマンドを使用します。

構文

```
CONN[ECT] [logon] [AS {SYSOPER | SYSDBA | SYSBACKUP | SYSDG | SYSKM | SYSRAC}]
```

logonの構文は、次のとおりです。

```
{username | /}[@connect_identifier] [edition={edition_name | DATABASE_DEFAULT}]
```

usernameを入力すると、パスワードの入力を求めるプロンプトがSQL*Plusによって表示されます。パスワードは入力しても画面には表示されません。

次の表で、CONNECTコマンドの構文の構成要素を説明します。

構文の構成要素	説明
/	接続要求の外部認証を呼び出します。このタイプの認証では、データベース・パスワードは使用しません。最も一般的な外部認証方式はオペレーティング・システム認証で、この方式では、特定のホスト・ユーザー・アカウントを使用してホスト・オペレーティング・システムにログインすることで、データベース・ユーザーの認証を行います。また、Oracle ウォレットまたはネットワーク・サービスを使用して、外部認証を実行することもできます。詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。「オペレーティング・システム認証の使用」も参照してください。
AS {SYSOPER | SYSDBA | SYSBACKUP | SYSDG | SYSKM | SYSRAC}	データベース・ユーザーが管理権限で接続していることを示します。この権限で接続できるのは、事前定義された特定の管理ユーザーまたはパスワード・ファイルに追加されているユーザーのみです。詳細は、「管理権限」を参照してください。
username	有効なデータベース・ユーザー名。username をデータ・ディクショナリと一致させて、ユーザー・パスワードの入力を求めることによって、データベースが接続要求を認証します。
connect_identifier (1)	リモート接続用の Oracle Net 接続識別子。正確な構文は Oracle Net の構成によって異なります。省略すると、SQL*Plus はローカル・インスタンスへの接続を試みます。 一般的な接続識別子は net service name です。この識別子は、Oracle Net 接続記述子(ネットワーク・アドレスとデータベース・サービス名)の別名です。通常、この別名はローカル・コンピュータの tnsnames.ora ファイルで解決されますが、他の方法でも解決できます。 接続識別子の詳細は、『Oracle Database Net Services 管理者ガイド』を参照してください。
connect_identifier (2)	別の方法として、接続識別子で簡易接続の構文も使用できます。簡易接続では、クライアント(ローカル)コンピュータで Oracle Net Services を設定しなくても、すぐに TCP/IP 接続が可能になります。 接続識別子用の簡易接続の構文は次のとおりです(前後の二重引用符も必要です)。 "host[:port][/service_name][:server][/instance_name]" ここで:

構文の構成要素	説明

- host は、リモート・データベースがあるコンピュータのホスト名または IP アドレスです。

  IP バージョン 4(IPv4)および IP バージョン 6(IPv6)の両方のアドレスに対応しています。IPv6 アドレスは大カッコで囲む必要があります。IPv6 アドレスの詳細は、『Oracle Database Net Services 管理者ガイド』を参照してください。

- port は host の Oracle Net リスナーがデータベース接続をリスニングする TCP ポートです。省略すると 1521 になります。

- service_name は、接続先のデータベース・サービス名です。リモート・ホスト上の Net Services のリスナー構成でデフォルトのサービスが指定されている場合は省略可能です。デフォルトのサービスが構成されていない場合は、service_name を指定する必要があります。通常、各データベースではグローバル・データベース名と同じ名前の標準サービスが提供され、グローバル・データベース名は DB_NAME と DB_DOMAIN 初期化パラメータで次のように構成されます。

  ```
  DB_NAME.DB_DOMAIN
  ```

  DB_DOMAIN が NULL の場合は、標準サービス名は DB_NAME のみです。たとえば、DB_NAME が orcl で DB_DOMAIN が us.example.com の場合、標準サービス名は orcl.us.example.com です。

  詳細は、「データベース・サービスでのアプリケーション・ワークロードの管理」を参照してください。

- server はサービス・ハンドラのタイプです。有効な値は、dedicated、shared および pooled です。省略されている場合、サーバーのデフォルトのタイプはリスナーによって選択され、構成されている場合は共有サーバー、それ以外の場合は専用サーバーになります。

- instance_name は、接続先のインスタンスです。サービス名とインスタンス名の両方を指定することもできますが、通常、このような指定は Oracle Real Application Clusters (Oracle RAC)環境でのみ行います。Oracle RAC 環境または単一インスタンス環境で、インスタンス名のみを指定した場合は、デフォルトのデータベース・サービスに接続されます。listener.ora ファイルにデフォルト・サービスが構成されていない場合は、エラーが生成されます。インスタンス名は、INSTANCE_NAME 初期化パラメータから取得できます。

簡易接続の詳細は、『Oracle Database Net Services 管理者ガイド』を参照してください。

構文の構成要素	説明
`edition={edition_name | DATABASE_DEFAULT}`	新規のデータベース・セッションを開始するエディションを指定します。エディションを指定した場合はそのエディションが必ず存在する必要があり、また、エディションに対する USE 権限が必要になります。この句が指定されていない場合は、セッションではデータベースのデフォルトのエディションが使用されます。 エディションおよびエディションに基づく再定義の詳細は、『Oracle Database 開発ガイド』を参照してください。

関連項目:

- 「オペレーティング・システム認証の使用」
- データベース・サービスの詳細は、「データベース・サービスでのアプリケーション・ワークロードの管理」を参照してください
- CONNECTコマンドの詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。
- ネット・サービス名の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。
- listener.oraでのデフォルトのサービスの定義方法は、『Oracle Database Net Servicesリファレンス』を参照してください。

親トピック: ステップ4: SQL*PlusのCONNECTコマンドの発行

1.4 Oracle Databaseソフトウェアのリリースの識別

特定のリリースを完全に識別するには、5つの番号が必要です。

Oracle Databaseは継続的に開発が進められ、メンテナンスが必要なため、定期的に新リリースが作成されています。すべての顧客が、新リリースを最初に利用したり、既存のリリースに対する特定のメンテナンスを必要とするわけではありません。この結果、複数の製品リリースが同時に存在することになります。

- Oracle Databaseのリリース番号について
  Oracle Databaseリリースは、リリース情報を示す5つの数値セグメントによって分類されます。
- 現行のリリース番号のチェック
  現在インストールされているOracle Databaseのリリースを識別し、使用している他のデータベース・コンポーネントのリリース・レベルを確認するには、データ・ディクショナリ・ビューPRODUCT_COMPONENT_VERSIONを問い合せます。

親トピック: データベース管理スタート・ガイド

1.4.1 Oracle Databaseのリリース番号について

Oracle Databaseリリースは、リリース情報を示す5つの数値セグメントによって分類されます。

ノート:

四半期ごとの更新は、リリース更新(Update または RU)およびリリース更新リビジョン(Revision または RUR)の形式で提供されています。パッチ・セットまたはバンドル・パッチ・セットはリリースされなくなりました。詳細は、My

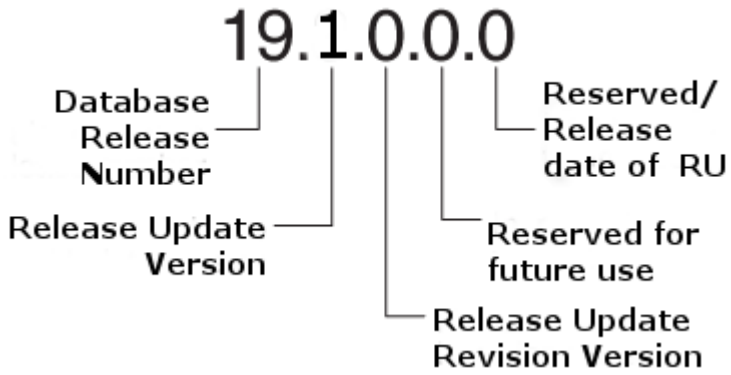
Oracle Support の Note 2285040.1 を参照してください。

Oracle Databaseリリースは、versionとversion_fullのリリースでリリースされます。

versionリリースは、メジャー・リリース・バージョン.0.0.0.0の形式で指定されます。メジャー・リリース・バージョンは、Oracle Databaseのバージョンがリリースされた最初の年の最後の2桁に基づいています。たとえば、2018年に最初にリリースされたOracle Databaseバージョンのメジャー・リリース・バージョンは18で、そのversionリリースは18.0.0.0.0になります。

version_fullリリースはversionリリースの更新で、メジャー・リリース・バージョン、四半期ごとのリリース更新バージョン(Update)、および四半期ごとのリリース更新リビジョン(Revision)に基づいて指定されます。version_fullリリースは、次の例に示すように、ピリオドで区切られた5つの数値セグメントで分類されます。

図1-1 Oracle Databaseのリリース番号の例



- 1番目の数字: この数字はメジャー・リリース・バージョンを示します。これは、Oracle Databaseのバージョンがリリースされた最初の年の最後の2桁でもあります。
- 2番目の数字: この数字はリリース更新バージョン(UpdateまたはRU)を示します。
- 3番目の数字: この数字はリリース更新リビジョン・バージョン(RevisionまたはRUR)を示します。
- 4番目の数字: この数字は将来使用するために予約されています。現在は常に0に設定されています。
- 5番目の数字: 最初の3つのフィールドのみが一般的に使用されますが、5番目のフィールドには19.7.0.0.200414など、リリース更新(RU)のリリース日を冗長に明確にする数値を表示できます。

ノート:

最初の 3 つの数字は、主に Oracle Database のリリースを識別します。

注意:

アップグレードを開始する前およびダウングレードを開始する前に、ターゲット・データベースに最新のリリース更新を適用することをお薦めします。可能な場合は、ソース環境にパッチが適用されていることも確認します。リリースの更新は累積されます。Oracle Grid Infrastructure 環境も更新する場合は、Oracle Database Oracle ホームを更新する前に、必ず最新のリリース更新を Grid 環境に適用します。更新の詳細は、My Oracle Support ノート 2118136.2 を参照してください。

関連トピック

- My Oracle Supportノート2285040.1
- My Oracle Supportノート2118136.2

親トピック: Oracle Databaseソフトウェアのリリースの識別

1.4.2 現行のリリース番号のチェック

現在インストールされているOracle Databaseのリリースを識別し、使用している他のデータベース・コンポーネントのリリース・レベルを確認するには、データ・ディクショナリ・ビューPRODUCT_COMPONENT_VERSIONを問い合せます。

問合せの例を次に示します。他の製品のリリース・レベルは、データベース・サーバーと関係なく増加する場合があります。

```
COL PRODUCT FORMAT A38
COL VERSION FORMAT A10
COL VERSION_FULL FORMAT A12
COL STATUS FORMAT A12
SELECT * FROM PRODUCT_COMPONENT_VERSION;
PRODUCT                                  VERSION    VERSION_FULL STATUS
---------------------------------------- ---------- ------------ ------------
NLSRTL                                   19.0.0.0.0 19.2.0.0.0   Production
Oracle Database 19c Enterprise Edition 19.0.0.0.0 19.2.0.0.0   Production
PL/SQL                                   19.0.0.0.0 19.2.0.0.0   Production
...
```

オラクル社にソフトウェアの問題を報告する際は、この問合せの結果を伝えることが重要です。

ノート:

V$VERSION ビューを問い合せて、現在インストールされているすべての Oracle Database コンポーネントに関するコンポーネント・レベル情報を表示することもできます。

親トピック: Oracle Databaseソフトウェアのリリースの識別

1.5 データベース管理者のセキュリティと権限について

Oracle Database管理者の管理タスクを実行するには、データベースとそのデータベースが稼働するサーバーのオペレーティング・システムで特定の権限が必要です。データベース管理者のアカウントへのアクセスは厳しく管理する必要があります。

- データベース管理者のオペレーティング・システム・アカウント
  データベースに対する管理業務の多くを実行するには、オペレーティング・システム・コマンドを実行できる必要があります。
- 管理ユーザー・アカウント
  Oracle Databaseでは、管理権限に関連付けられている複数の管理ユーザー・アカウントが用意されています。

親トピック: データベース管理スタート・ガイド

1.5.1 データベース管理者のオペレーティング・システム・アカウント

データベースに対する管理業務の多くを実行するには、オペレーティング・システム・コマンドを実行できる必要があります。

Oracle Databaseが稼働するオペレーティング・システムによって異なりますが、オペレーティング・システムにアクセスするには、オペレーティング・システム・アカウント(ID)が必要になります。その場合、そのオペレーティング・システム・アカウントに対しては、他のデータベース・ユーザーには不要なオペレーティング・システム権限やアクセス権(Oracle Databaseソフトウェアのインストールの実行など)が必要になります。Oracle Databaseファイルを自分のアカウント内に格納する必要はありませんが、それらに

対するアクセス権は必要です。

関連項目:

使用しているオペレーティング・システム固有のOracleマニュアルを参照してください。データベース管理者のアカウントの作成方法は、オペレーティング・システムによって異なります。

親トピック: データベース管理者のセキュリティと権限について

1.5.2 管理ユーザー・アカウント

Oracle Databaseでは、管理権限に関連付けられている複数の管理ユーザー・アカウントが用意されています。

- 管理ユーザー・アカウントについて
  管理ユーザー・アカウントには、CREATE ANY TABLE権限やALTER SESSION権限、SYSスキーマが所有するパッケージのEXECUTE権限のような、データベースの領域の管理に必要となる特別な権限があります。
- SYS
  Oracleデータベースを作成すると、すべての権限を持つユーザーSYSが自動的に作成されます。
- SYSTEM
  Oracleデータベースを作成すると、ユーザーSYSTEMも自動的に作成され、DBAロールが付与されます。
- SYSBACKUP、SYSDG、SYSKMおよびSYSRAC
  Oracleデータベースを作成すると、データベース管理者の業務の分離を容易にするために、SYSBACKUP、SYSDG、SYSKMおよびSYSRACの各ユーザーが自動的に作成されます。
- DBAロール
  事前に定義されたDBAロールは、Oracle Databaseのインストールで自動的に作成されます。このロールには、ほとんどのデータベース・システム権限が含まれています。このため、DBAロールは実際のDBAにのみ付与してください。

親トピック: データベース管理者のセキュリティと権限について

1.5.2.1 管理ユーザー・アカウントについて

管理ユーザー・アカウントには、CREATE ANY TABLE権限やALTER SESSION権限、SYSスキーマが所有するパッケージのEXECUTE権限のような、データベースの領域の管理に必要になる特別な権限があります。

Oracle Databaseのインストール時には、次の管理ユーザー・アカウントが自動的に作成されます。

- SYS
- SYSTEM
- SYSBACKUP
- SYSDG
- SYSKM
- SYSRAC

少なくとも1人の管理ユーザーを追加作成し、日常的な管理タスクを実行するための適切な権限を付与することをお薦めします。これらの目的のためにSYSおよびSYSTEMを使用しないでください。

ノート:

Oracle Universal Installer (OUI)および Database Configuration Assistant (DBCA)では、SYS および SYSTEM のパスワードの入力が要求され、デフォルトのパスワードは受け入れられません。

関連項目:

- 「データベースの保護: ユーザーSYSおよびSYSTEMのパスワードの指定」
- データベースの構成のためのセキュリティ・チェックリストについては、Oracle Databaseセキュリティ・ガイドを参照してください。

親トピック: 管理ユーザー・アカウント

1.5.2.2 SYS

Oracle Databaseを作成すると、すべての権限を持つユーザーSYSが自動的に作成されます。

データベースのデータ・ディクショナリの実表とビューはすべて、SYSスキーマに格納されます。これらの実表とビューは、Oracle Databaseの操作にとって重要です。データ・ディクショナリの整合性を保持するには、SYSスキーマ内の表をデータベースのみで操作します。ユーザーやデータベース管理者はそれらの表を修正したり、ユーザーSYSのスキーマ内に表を作成しないでください。(ただし、必要に応じてデータ・ディクショナリ設定の記憶域パラメータを変更することは可能です。)

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

親トピック: 管理ユーザー・アカウント

1.5.2.3 SYSTEM

Oracle Databaseを作成すると、ユーザーSYSTEMも自動的に作成され、DBAロールが付与されます。

このユーザー名SYSTEMを使用して、管理情報を表示する追加表とビュー、および各種のOracle Databaseのオプションとツール製品で使用される内部表とビューが作成されます。管理ユーザー以外のユーザーに関係する表の格納に、SYSTEMスキーマを使用しないでください。

親トピック: 管理ユーザー・アカウント

1.5.2.4 SYSBACKUP、SYSDG、SYSKMおよびSYSRAC

Oracle Databaseを作成すると、データベース管理者の業務の分離を容易にするために、SYSBACKUP、SYSDG、SYSKMおよびSYSRACの各ユーザーが自動的に作成されます。

これらのユーザーは次の方法で業務を分離します。

- SYSBACKUPによって、RMANまたはSQL*PlusからのOracle Recovery Manager (RMAN)バックアップおよびリカバリ操作が容易になります。
- SYSDGによって、Data Guardの操作が容易になります。ユーザーは、Data Guard BrokerまたはDGMGRLコマンドライン・インタフェースを使用して操作を実行できます。
- SYSKMによって、透過的データ暗号化キーストア操作が容易になります。
- SYSRACによって、SRVCTLなどのOracle RACユーティリティにかわってClusterwareエージェントでデータベースに接続することで、Oracle Real Application Clusters (Oracle RAC)操作が容易になります

  SYSRAC管理権限はデータベース・ユーザーに付与できず、パスワード・ファイルでサポートされません。SYSRAC管理

権限は、オペレーティング・システム認証を使用してデータベースに接続するためにOracle ClusterwareのOracleエージェントによってのみ使用されます。

これらの各アカウントには、新しい管理権限のために指定された、同じ名前を持つ1つのユーザーがあります。具体的には、SYSBACKUPアカウントには、SYSBACKUP管理権限に対して指定されたユーザーがあります。SYSDGアカウントには、SYSDG管理権限に対して指定されたユーザーがあります。SYSKMアカウントには、SYSKM管理権限に対して指定されたユーザーがあります。

ユーザーを作成し、そのユーザーに、日常的な管理タスクを実施するときに使用する、適切な管理権限を付与します。これにより、各ユーザー・アカウントを個別に管理し、各ユーザー・アカウントに個別のパスワードを割り当てることができます。SYSBACKUP、SYSDGまたはSYSKMユーザー・アカウントはこれらの目的のために使用しないでください。

これらの管理権限のいずれかを使用するには、データベースに接続するときに、権限(たとえば、AS SYSBACKUP、AS SYSDGまたはAS SYSKM)を指定して権限を使用する必要があります。認証が成功すると、ユーザーは、管理権限が有効になっているセッションでデータベースに接続されます。この場合、セッション・ユーザーは対応する管理ユーザー・アカウントになります。たとえば、ユーザーbradminがAS SYSBACKUP管理権限で接続した場合、セッション・ユーザーはSYSBACKUPになります。

ノート:

- これらのユーザー・アカウントは削除できません。
- これらのユーザー・アカウントはスキーマ専用アカウントです(つまり、パスワードなしで作成される)。これらのユーザー・アカウントには、認証する必要が生じたらいつでもパスワードを割り当てることができます。

関連項目:

- 「管理権限」
- Oracle Databaseセキュリティ・ガイド

親トピック: 管理ユーザー・アカウント

1.5.2.5 DBAロール

事前に定義されたDBAロールは、Oracle Databaseのインストールで自動的に作成されます。このロールには、ほとんどのデータベース・システム権限が含まれています。このため、DBAロールは実際のDBAにのみ付与してください。

ノート:

DBA ロールには、SYSDBA、SYSOPER、SYSBACKUP、SYSDG または SYSKM システム権限は含まれていません。これらは、データベースとインスタンスの起動や停止など、基本的なデータベース管理タスクの実行を管理者に許可する特別な管理権限です。これらの管理権限の詳細は、「管理権限」を参照してください。

関連項目:

- 管理ユーザー・アカウントの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

- 「パスワード・ファイル認証の使用」

親トピック: 管理ユーザー・アカウント

1.6 データベース管理者の認証

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

- 管理権限
  管理者が基本的なデータベース操作を実行するために必要な管理権限は、特別なシステム権限によって付与されます。
- 管理権限で認可されている操作
  各管理権限は、操作の特定のセットを認可します。
- データベース管理者の認証方法
  データベース管理者は、アカウント・パスワード、オペレーティング・システム(OS)認証、パスワード・ファイル、またはOracle Internet Directoryなどのディレクトリベースの認証サービスを使用した強力な認証によって認証できます。
- オペレーティング・システム認証の使用
  特別なオペレーティング・システム・グループにメンバーシップがあると、DBAは、データベースのユーザー名とパスワードを使用するかわりに、オペレーティング・システムを経由してデータベースに対する認証を行うことができます。このことはオペレーティング・システム認証と呼ばれます。
- パスワード・ファイル認証の使用
  Oracle DatabaseインスタンスおよびOracle Automatic Storage Management (Oracle ASM)インスタンスのパスワード・ファイル認証を使用できます。Oracle Databaseのパスワード・ファイルはデータベース・パスワード・ファイルと呼ばれ、Oracle ASMのパスワード・ファイルはOracle ASMパスワード・ファイルと呼ばれます。

親トピック: データベース管理スタート・ガイド

1.6.1 管理権限

管理者が基本的なデータベース操作を実行するために必要な管理権限は、特別なシステム権限によって付与されます。

これらの権限を次に示します。

- SYSDBA
- SYSOPER
- SYSBACKUP
- SYSDG
- SYSKM
- SYSRAC

SYSRAC権限を除き、必要な認可レベルに応じてこれらの権限がユーザーに付与されます。SYSRAC権限は、オペレーティング・システム認証を使用してデータベースに接続するためにOracle ClusterwareのOracleエージェントによってのみ使用されるため、ユーザーに付与できません。

Oracle Database 12cリリース1 (12.1)からは、SYSBACKUP、SYSDGおよびSYSKM管理権限を使用できるようになりました。Oracle Database 12cリリース2 (12.2)からは、SYSRAC管理権限を使用できます。それぞれの新しい管理権限によって、管理の各領域におけるタスクを完了するのに必要な最小限の権限が付与されます。新しい管理権限を使用すると、数多くの一般的なタスクに対してSYSDBA管理権限を付与することを回避できます。

ノート:

これらの管理権限を使用すると、データベースがオープンしていなくてもデータベース・インスタンスにアクセスできます。これらの権限の制御は、完全にデータベース機能の範囲外にあります。これらの権限を持つデータベース管理者を認証する方式には、オペレーティング・システム(OS)認証、パスワード・ファイルおよびディレクトリベースの認証サービスによる強力な認証があります。

また、これらの権限は、他の方法では権限を付与できない特定のデータベース操作を実行可能にする接続のタイプと考えることもできます。たとえば、SYSDBA 権限がある場合は CONNECT コマンドの AS SYSDBA 句を指定してデータベースに接続し、STARTUP および SHUTDOWN 操作を実行できます。「データベース管理者の認証方法」を参照してください。

親トピック: データベース管理者の認証

1.6.2 管理権限で許可されている操作

各管理権限は、操作の特定のセットを認可します。

次の表に、各管理権限で許可されている操作を示します。

管理権限	許可される操作
SYSDBA	● STARTUP および SHUTDOWN 操作の実行 ● ALTER DATABASE: オープン、マウント、バックアップまたは文字セットの変更 ● CREATE DATABASE ● DROP DATABASE ● CREATE SPFILE ● ALTER DATABASE ARCHIVELOG ● ALTER DATABASE RECOVER ● RESTRICTED SESSION 権限を含む この管理権限を使用すると、ユーザー・データの表示を含め、ほとんどの操作を実行できます。これは最も強力な管理権限です。
SYSOPER	● STARTUP および SHUTDOWN 操作の実行 ● CREATE SPFILE ● ALTER DATABASE: オープン、マウントまたはバックアップ

管理権限	許可される操作
	● ALTER DATABASE ARCHIVELOG ● ALTER DATABASE RECOVER(完全リカバリのみ。UNTIL TIME|CHANGE|CANCEL|CONTROLFILE など、不完全リカバリの形式では、SYSDBA で接続する必要があります。) ● RESTRICTED SESSION 権限を含む この権限を使用すると、ユーザー・データの表示を除く基本操作を実行できます。
SYSBACKUP	この権限を使用すると、Oracle Recovery Manager (RMAN)または SQL*Plus からバックアップおよびリカバリ操作を実行できます。 この管理権限で許可される操作の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
SYSDG	この権限を使用すると、Data Guard 操作を実行できます。Data Guard Broker または DGMGRL コマンドライン・インタフェースでこの権限を使用できます。 この管理権限で許可される操作の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
SYSKM	この権限を使用すると、透過的データ暗号化キーストア操作を実行できます。 この管理権限で許可される操作の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
SYSRAC	この権限により、Oracle Clusterware の Oracle エージェントは Oracle Real Application Clusters (Oracle RAC)操作を実行できます。 この管理権限で許可される操作の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

これらの権限の使用を許可する方法は、使用する認証方式によって異なります。

管理権限で接続すると、通常ユーザー名に関連付けられていない現行スキーマで接続が確立されます。SYSDBAの場合、現行スキーマはSYSです。SYSOPERの場合、現行スキーマはPUBLICです。SYSBACKUP、SYSDGおよびSYSRACの場合、現行スキーマは名前解決のためのSYSです。ただし、SYSKMの現行スキーマはSYSKMです。

また、管理権限で接続すると、特定のセッション・ユーザーを使用して接続が確立されます。SYSDBAとして接続した場合、セッション・ユーザーはSYSとなります。SYSOPERの場合、セッション・ユーザーはPUBLICとなります。SYSBACKUP、SYSDG、SYSKM、SYSRACの場合、セッション・ユーザーはそれぞれSYSBACKUP、SYSDG、SYSKM、SYSRACとなります。

関連項目:

- 「管理ユーザー・アカウント」
- 「オペレーティング・システム認証の使用」
- 「パスワード・ファイル認証の使用」
- 現行スキーマおよびセッション・ユーザーの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- Oracle Databaseセキュリティ・ガイド

例1-18 AS SYSDBA接続時の現行スキーマ

この例では、ユーザーがSYSDBA管理権限で接続したときに、他のスキーマ(SYS)に割り当てられることを説明します。SYSDBA管理権限を付与されているサンプル・ユーザーmydbaが、次のコマンドおよび文を発行したと想定します。

```
CONNECT mydba
CREATE TABLE admin_test(name VARCHAR2(20));
```

後で、ユーザーmydbaが次のコマンドおよび文を発行します。

```
CONNECT mydba AS SYSDBA
SELECT * FROM admin_test;
```

ユーザーmydbaは次のエラー・メッセージを受け取ります。

```
ORA-00942: table or view does not exist
```

SYSDBAで接続しているため、ユーザーmydbaは現在SYSスキーマを参照していますが、表はmydbaスキーマに作成されています。

例1-19 AS SYSBACKUP接続時の現行スキーマおよびセッション・ユーザー

この例では、ユーザーがSYSBACKUP管理権限で接続したときに、他のスキーマ(SYS)および他のセッション・ユーザー(SYSBACKUP)に割り当てられることを説明します。SYSBACKUP管理権限を付与されているサンプル・ユーザーmydbaが、次のコマンドおよび文を発行したと想定します。

```
CONNECT mydba AS SYSBACKUP
SELECT SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA') FROM DUAL;
SYS_CONTEXT('USERENV','CURRENT_SCHEMA')
--------------------------------------------------------------------------------
SYS
SELECT SYS_CONTEXT('USERENV', 'SESSION_USER') FROM DUAL;
SYS_CONTEXT('USERENV','SESSION_USER')
--------------------------------------------------------------------------------
SYSBACKUP
```

親トピック: データベース管理者の認証

1.6.3 データベース管理者の認証方法

データベース管理者は、アカウント・パスワード、オペレーティング・システム(OS)認証、パスワード・ファイル、またはOracle Internet Directoryなどのディレクトリベースの認証サービスを使用した強力な認証によって認証できます。

- データベース管理者の認証方法について
  データベース管理者を認証する方法は複数あります。
- セキュリティで保護されていないリモート接続
  セキュリティで保護されていない接続で、権限を持つユーザーとしてOracle Databaseに接続するには、パスワード・

ファイルによる認証を受ける必要があります。

- ローカル接続およびセキュリティで保護されたリモート接続
  ローカル接続またはセキュリティで保護されたリモート接続で、権限を持つユーザーとしてOracle Databaseに接続できます。

親トピック: データベース管理者の認証

1.6.3.1 データベース管理者の認証方法について

データベース管理者を認証する方法は複数あります。

Oracleデータベースは、他のユーザーと同様に(アカウント・パスワードを使用して)、データ・ディクショナリを介してデータベース管理者を認証できます。データベース・パスワードでは、大/小文字が区別されることに注意してください。データベース・パスワードの大/小文字区別の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

通常のデータ・ディクショナリ認証に加えて、SYSDBA、SYSOPER、SYSBACKUP、SYSDGまたはSYSKM権限を持つデータベース管理者の認証には、次の方式を使用できます。

- オペレーティング・システム(OS)認証
- パスワード・ファイル(KerberosおよびSSL認証サービスを含む)
- Oracle Internet Directoryなどのディレクトリ・ベースの認証サービスによる強力な認証

ノート:

SYSRAC 権限は、Oracle Clusterware の Oracle エージェントによる OS 認証のみ許可します。パスワード・ファイルおよび強力な認証は、SYSRAC 権限では使用できません。

これらの方式は、データベースが起動していない場合や使用できない場合にデータベース管理者を認証する際に必要になります。(データベースが使用可能な場合にも使用できます。)

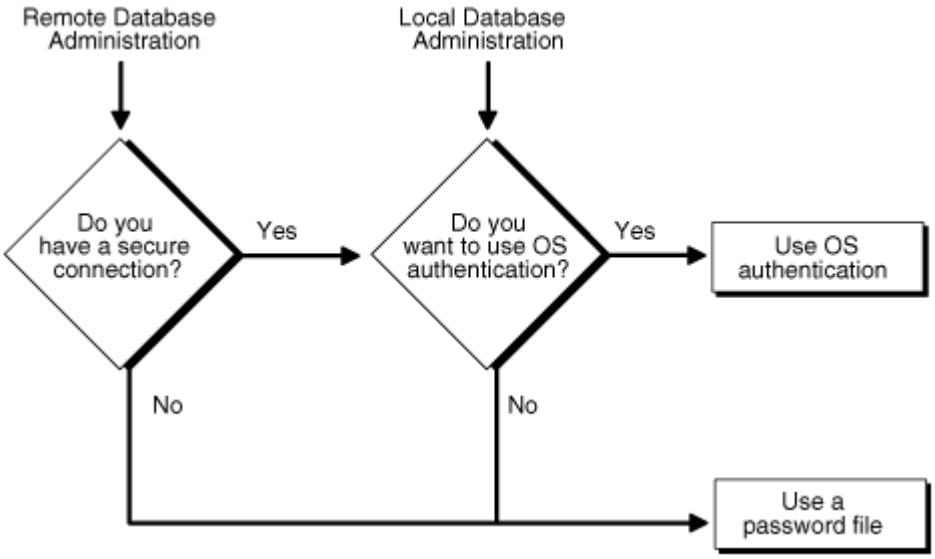
ここからは、オペレーティング・システム認証とパスワード・ファイル認証について説明します。ディレクトリベースの認証を使用してデータベース管理者を認証する方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

ノート:

オペレーティング・システム認証は、パスワード・ファイル認証より優先されます。オペレーティング・システム認証の要件を満たしている場合は、パスワード・ファイルを使用している場合でもオペレーティング・システム認証によって認証されます。

データベースと同じシステム上でデータベースをローカルで管理するか、または単一のリモート・クライアントから複数の異なるデータベースを管理するかによって、選択が決まります。次の図は、データベース管理者用に選択できる認証方式を示しています。

図1-2 データベース管理者の認証方式



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

関連項目:

- ディレクトリベースの認証を使用してデータベース管理者を認証する方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- Oracle Database Net Services管理者ガイド

親トピック: データベース管理者の認証方法

1.6.3.2 セキュリティで保護されていないリモート接続

セキュリティで保護されていない接続で、権限を持つユーザーとしてOracle Databaseに接続するには、パスワード・ファイルによる認証を受ける必要があります。

パスワード・ファイル認証を使用すると、SYSDBA、SYSOPER、SYSBACKUP、SYSDGまたはSYSKM管理権限を付与されたデータベース・ユーザー名を追跡管理するために、データベースでは、パスワード・ファイルを使用します。この認証方式については、「パスワード・ファイル認証の使用」を参照してください。

親トピック: データベース管理者の認証方法

1.6.3.3 ローカル接続およびセキュリティで保護されたリモート接続

ローカル接続またはセキュリティで保護されたリモート接続で、権限を持つユーザーとしてOracle Databaseに接続できます。

接続には、次の2つの方法があります。

- データベースにパスワード・ファイルがあり、ユーザーがシステム権限を付与されている場合は、パスワード・ファイルを使用して接続し、認証を受けることができます。
- サーバーでパスワード・ファイルを使用していない場合や、ユーザーにシステム権限が付与されていないためにパスワード・ファイルにアクセスできない場合は、オペレーティング・システム認証を使用できます。ほとんどのオペレーティング・システムでは、データベース管理者の認証には、データベース管理者のオペレーティング・システム・ユーザー名を特別なグループに配置する必要があります。

  たとえば、OSDBAグループのユーザーには、SYSDBA管理権限が付与されます。同様に、OSOPERグループはユーザーにSYSOPER管理権限を付与するために使用され、OSBACKUPDBAグループはユーザーにSYSBACKUP管理

権限を付与するために使用され、OSDGDBAグループはユーザーにSYSDG管理権限を付与するために使用され、OSKMDBAグループはユーザーにSYSKM管理権限を付与するために使用され、OSRACDBAグループはユーザーにSYSRAC管理権限を付与するために使用されます。

親トピック: データベース管理者の認証方法

1.6.4 オペレーティング・システム認証の使用

特別なオペレーティング・システム・グループにメンバーシップがあると、DBAは、データベースのユーザー名とパスワードを使用するかわりに、オペレーティング・システムを経由してデータベースに対する認証を行うことができます。このことはオペレーティング・システム認証と呼ばれます。

- オペレーティング・システム・グループ
  オペレーティング・システム・グループは、データベース・インストール・プロセスの一貫として作成され、特定の名前を割り当てられます。
- オペレーティング・システム認証を使用するための準備
  DBAは、データベース・ユーザー名およびパスワードを使用するのでなく、オペレーティング・システムを通じてデータベースに対して認証できます。
- オペレーティング・システム認証を使用した接続
  ユーザーは、オペレーティング・システム認証を使用してデータベースに接続できます。

親トピック: データベース管理者の認証

1.6.4.1 オペレーティング・システム・グループ

オペレーティング・システム・グループは、データベースのインストール・プロセスで作成され、特定の名前が割り当てられます。

オペレーティング・システム・グループのデフォルトの名前は、次の表に示すようにオペレーティング・システムによって異なります。

オペレーティング・システム・グループ	UNIXまたはLinuxユーザー・グループ	Windowsユーザー・グループ
OSDBA	`dba`	`ORA_DBA` (すべての Oracle ホーム) `ORA_HOMENAME_DBA` (特定の Oracle ホームごと)
OSOPER	`oper`	`ORA_OPER` (すべての Oracle ホーム) `ORA_HOMENAME_OPER` (特定の Oracle ホームごと)
OSBACKUPDBA	`backupdba`	`ORA_HOMENAME_SYSBACKUP`
OSDGDBA	`dgdba`	`ORA_HOMENAME_SYSDG`
OSKMDBA	`kmdba`	`ORA_HOMENAME_SYSKM`
OSRACDBA	`racdba`	`ORA_HOMENAME_SYSRAC`

Windowsユーザー・グループ名では、HOMENAMEをOracleホーム名で置き換えます。

Oracle Universal Installerではデフォルトの名前が使用されますが、UNIXまたはLinuxではこの名前は上書きできます。UNIXまたはLinuxで、上書きする理由の1つは、異なるOracleホームの同じホスト・コンピュータ上で複数のインスタンスが実行されている場合です。各インスタンスのプリンシパルDBAを異なるユーザーにする場合は、各インスタンスに対して異なるグループを作成することによって、各インスタンスのセキュリティを高めることができます。

たとえば、異なるOracleホームの同じUNIXまたはLinuxホスト上の2つのインスタンスについて、1つ目のインスタンスのOSDBAグループの名前をdba1とし、2つ目のインスタンスのOSDBAの名前をdba2にできます。1人目のDBAはdba1のみのメンバーとし、2人目のDBAはdba2のみのメンバーとします。これにより、オペレーティング・システム認証を使用すると、各DBAは割り当てられたインスタンスに対してのみ接続できます。

Windowsでは、デフォルトのユーザー・グループ名は変更できません。HOMENAMEプレースホルダを使用すると、同じホストWindowsコンピュータ上で複数のインスタンスが実行されている場合に異なるユーザー・グループ名を指定できます。

グループのメンバーシップは、データベースへの接続に次のように影響します。

- データベースへの接続時に、OSDBAグループに属するユーザーがAS SYSDBAを指定した場合、そのユーザーは、SYSDBA管理権限でデータベースに接続します。
- データベースへの接続時に、OSOPERグループに属するユーザーがAS SYSOPERを指定した場合、そのユーザーは、SYSOPER管理権限でデータベースに接続します。
- データベースへの接続時に、OSBACKUPDBAグループに属するユーザーがAS SYSBACKUPを指定した場合、そのユーザーは、SYSBACKUP管理権限でデータベースに接続します。
- データベースへの接続時に、OSDGDBAグループに属するユーザーがAS SYSDGを指定した場合、そのユーザーは、SYSDG管理権限でデータベースに接続します。
- データベースへの接続時に、OSKMDBAグループに属するユーザーがAS SYSKMを指定した場合、そのユーザーは、SYSKM管理権限でデータベースに接続します。
- データベースへの接続時に、OSRACDBAグループに属するユーザーがAS SYSRACを指定した場合、そのユーザーは、SYSRAC管理権限でデータベースに接続します。
- これらのオペレーティング・システム・グループのいずれにも属していないユーザーがSYSDBA、SYSOPER、SYSBACKUP、SYSDG、SYSKMまたはSYSRACとして接続しようとした場合、CONNECTコマンドを発行するとエラーが発生します。

関連項目:

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

親トピック: オペレーティング・システム認証の使用

1.6.4.2 オペレーティング・システム認証を使用するための準備

DBAは、データベース・ユーザー名およびパスワード以外のオペレーティング・システムを通じてデータベースに対して認証できます。

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

1. ユーザーのオペレーティング・システム・アカウントを作成します。
2. 適切なオペレーティング・システムで定義されたグループにアカウントを追加します。

親トピック: オペレーティング・システム認証の使用

1.6.4.3 オペレーティング・システム認証を使用した接続

ユーザーは、オペレーティング・システム認証を使用してデータベースに接続できます。

オペレーティング・システム認証は、次のいずれかのアクションを実行することで使用できます。

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

```
CONNECT / AS SYSDBA
CONNECT / AS SYSOPER
CONNECT / AS SYSBACKUP
CONNECT / AS SYSDG
CONNECT / AS SYSKM
```

- セキュリティで保護された接続を介したリモート・オペレーティング・システム認証がサポートされているのは、Windowsプラットフォームのみです。リモート・データベースのネット・サービス名を指定する必要があります。

```
CONNECT /@net_service_name AS SYSDBA
CONNECT /@net_service_name AS SYSOPER
CONNECT /@net_service_name AS SYSBACKUP
CONNECT /@net_service_name AS SYSDG
CONNECT /@net_service_name AS SYSKM
```

  クライアント・コンピュータとデータベース・ホスト・コンピュータの両方がWindowsドメイン上にあることが必要です。

ノート:

SYSRAC 管理権限は、オペレーティング・システム認証を使用してデータベースに接続するために Oracle Clusterware の Oracle エージェントによってのみ使用されます。

関連項目:

- 「SQL*Plusを使用したデータベースへの接続」
- CONNECTコマンドの構文は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください

親トピック: オペレーティング・システム認証の使用

1.6.5 パスワード・ファイル認証の使用

Oracle DatabaseインスタンスおよびOracle Automatic Storage Management(Oracle ASM)インスタンスのパスワード・ファイル認証を使用できます。Oracle Databaseのパスワード・ファイルはデータベース・パスワード・ファイルと呼ばれ、Oracle ASMのパスワード・ファイルはOracle ASMパスワード・ファイルと呼ばれます。

- パスワード・ファイル認証を使用するための準備
  パスワード・ファイル認証を準備するには、パスワード・ファイルを作成し、REMOTE_LOGIN_PASSWORDFILE初期化パラメータを設定し、権限を付与する必要があります。
- パスワード・ファイル認証を使用した接続
  パスワード・ファイル認証を使用すると、管理ユーザーは、SQL*PlusのCONNECTコマンドによりローカルまたはリモート・データベースに接続して認証を受けることができます。デフォルトでは、パスワードは大/小文字が区別されます。

関連項目:

Oracle ASMパスワード・ファイルの作成の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

親トピック: データベース管理者の認証

1.6.5.1 パスワード・ファイル認証を使用するための準備

パスワード・ファイル認証を準備するには、パスワード・ファイルを作成し、REMOTE_LOGIN_PASSWORDFILE初期化パラメータを設定し、権限を付与する必要があります。

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

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

   ```
   orapwd FILE=filename FORMAT=12.2
   ```

   詳細は、「データベース・パスワード・ファイルの作成とメンテナンス」を参照してください。

   ノート:

   - Oracle Database のインストール・プロセスで Database Configuration Assistant (DBCA)を起動した場合は、DBCA によってパスワード・ファイルが作成されます。
   - パスワード・ファイルを FORMAT=LEGACY 引数を使用して作成した場合、管理権限 SYSBACKUP、SYSDG および SYSKM はパスワード・ファイルでサポートされません。
   - FORMAT コマンドライン引数のデフォルトは 12.2 です。
   - 管理権限 SYSRAC は、パスワード・ファイルではサポートされません。
   - 管理権限は、ファイルが FORMAT=12.2 引数で作成された場合にのみ外部ユーザーに付与できます。FORMAT=12.2 では、管理ユーザーの SSL および Kerberos 認証も有効になります。
   - Oracle ASM ディスク・グループに格納されるデータベース・パスワード・ファイルを作成する場合、それを複数の Oracle RAC データベース・インスタンスで共有できます。パスワード・ファイルは、各 Oracle RAC データベース・インスタンスには複製されません。

2. REMOTE_LOGIN_PASSWORDFILE初期化パラメータをexclusiveに設定します。(この値がデフォルトです。)

   ノート:

   REMOTE_LOGIN_PASSWORDFILE は静的な初期化パラメータであるため、変更するにはデータベースを再起動する必要があります。

3. ユーザーSYS(または管理権限を持つ他のユーザー)としてデータベースに接続します。
4. データベース内にユーザーが存在しない場合は、ユーザーを作成し、パスワードを割り当てます。

データベース・パスワードでは、大/小文字が区別されることに注意してください。データベース・パスワードの大/小文字区別の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

5. ユーザーに、SYSDBA、SYSOPER、SYSBACKUP、SYSDGまたはSYSKM管理権限を付与します。たとえば:

```
GRANT SYSDBA to mydba;
```

この文はパスワード・ファイルにユーザーを追加し、これにより接続AS SYSDBA、AS SYSOPER、AS SYSBACKUP、AS SYSDGまたはAS SYSKMを可能にします。

関連項目:

パスワード・ファイルの作成とメンテナンスの方法は、「データベース・パスワード・ファイルの作成とメンテナンス」を参照してください

親トピック: パスワード・ファイル認証の使用

1.6.5.2 パスワード・ファイル認証を使用した接続

パスワード・ファイル認証を使用すると、SQL*PlusのCONNECTコマンドを使用することによって、管理ユーザーが認証され、ローカルまたはリモート・データベースに接続できます。デフォルトでは、パスワードは大/小文字が区別されます。

パスワード・ファイル認証を使用して接続するには:

- SQL*Plusでは、有効なユーザー名とパスワードおよびAS SYSDBA、AS SYSOPER、AS SYSBACKUP、AS SYSDGまたはAS SYSKM句を使用してCONNECTコマンドを実行します。

たとえば、ユーザーmydbaがSYSDBA権限を付与されている場合、次のコマンドで接続できます。

```
CONNECT mydba AS SYSDBA
```

しかし、ユーザーmydbaがSYSOPER権限を付与されていない場合、次のコマンドは失敗します。

```
CONNECT mydba AS SYSOPER
```

ノート:

オペレーティング・システム認証は、パスワード・ファイル認証より優先されます。具体的には、ユーザーが適切なオペレーティング・システム・グループ(OSDBA や OSOPER など)のメンバーであるときに、適切な句(AS SYSDBA など)を使用して接続すると、指定したユーザー名/パスワードにかかわらず、関連付けられている管理権限で接続されます。

ユーザーがいずれのオペレーティング・システム・グループにも属しておらず、パスワード・ファイルにも指定されていない場合、句を使用して接続しようとすると失敗します。

関連項目:

- SQL*Plusを使用したデータベースへの接続について
- ORAPWDを使用したデータベース・パスワード・ファイルの作成

- CONNECTコマンドの構文は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。
- 『Oracle Databaseセキュリティ・ガイド』

親トピック: パスワード・ファイル認証の使用

1.7 データベース・パスワード・ファイルの作成とメンテナンス

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

- ORAPWD構文およびコマンドライン引数の説明
  ORAPWDコマンドでは、パスワード・ファイルを作成およびメンテナンスします。
- ORAPWDを使用したデータベース・パスワード・ファイルの作成
  ORAPWDを使用してデータベース・パスワード・ファイルを作成できます。
- データベース・パスワード・ファイルの共有と無効化
  初期化パラメータREMOTE_LOGIN_PASSWORDFILEを使用して、データベース・パスワード・ファイルが複数のOracle Databaseインスタンス間で共有されるかどうかを制御します。このパラメータを使用して、パスワード・ファイル認証を無効にすることもできます。
- 管理者パスワードとデータ・ディクショナリとの同期の維持
  REMOTE_LOGIN_PASSWORDFILE初期化パラメータをnoneからexclusiveまたはsharedに変更した場合、データ・ディクショナリ内のパスワードとSYS以外(SYSDBA、SYSOPER、SYSBACKUP、SYSDGおよびSYSKM)の管理ユーザーのパスワード・ファイル内のパスワードを同じにする必要があります。
- データベース・パスワード・ファイルへのユーザーの追加
  ユーザーにSYSDBA、SYSOPER、SYSBACKUP、SYSDGまたはSYSKM管理権限を付与すると、そのユーザーの名前と権限情報がデータベース・パスワード・ファイルに追加されます。
- 管理権限の付与と取消し
  管理権限を付与するにはGRANT文を使用します。管理権限を取り消すにはREVOKE文を使用します。
- データベース・パスワード・ファイル・メンバーの表示
  V$PWFILE_USERSビューには、管理権限を付与されたユーザーに関する情報が含まれます。
- データベース・パスワード・ファイルの削除
  不要になった場合はデータベース・パスワードを削除できます。

関連項目:

- 「パスワード・ファイル認証の使用」
- 「データベース管理者の認証方法」
- Oracle ASMパスワード・ファイルの作成およびメンテナンスの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

親トピック: データベース管理スタート・ガイド

1.7.1 ORAPWD構文およびコマンドライン引数の説明

ORAPWDコマンドでは、パスワード・ファイルを作成およびメンテナンスします。

ORAPWDコマンドの構文は次のとおりです。

```
orapwd FILE=filename
[FORCE={y|n}]
[ASM={y|n}]
[DBUNIQUENAME=dbname]
[FORMAT={12.2|12}]
[SYS={y|n|password|external('sys-external-name')|global('sys-directory-DN')}]
[SYSBACKUP={y|n|password|external('sysbackup-external-name')|global('sysbackup-
directory-DN')}]
[SYSDG={y|n|password|external('sysdg-external-name')|global('sysdg-directory-DN')}]
[SYSKM={y|n|password|external('syskm-external-name')|global('syskm-directory-DN')}]
[DELETE={y|n}]
[INPUT_FILE=input-fname]

orapwd DESCRIBE FILE=filename
```

次の表にコマンドの引数がまとめられています。

引数	説明
FILE	DESCRIBE 引数が含まれていない場合は、新しいパスワード・ファイルに割り当てる名前を指定します。完全なパスを指定する必要があります。ファイル名のみを指定した場合、ファイルはカレント・ディレクトリに書き込まれます。 DESCRIBE 引数が含まれている場合は、既存のパスワード・ファイルの名前を指定します。
FORCE	(オプション)y の場合は、既存のパスワード・ファイルが上書きされます。
ASM	(オプション) y の場合は、Oracle ASM ディスク・グループに Oracle ASM パスワード・ファイルを作成します。 デフォルトの n の場合、オペレーティング・システムのファイル・システムにパスワード・ファイルを作成します。DBUNIQUENAME 引数を指定した場合、パスワード・ファイルはデータベース・パスワード・ファイルになります。DBUNIQUENAME 引数を指定しない場合、パスワード・ファイルはデータベース・パスワード・ファイルまたは Oracle ASM パスワード・ファイルにできます。
DBUNIQUENAME	ASM ディスク・グループのみに存在するデータベース・パスワード・ファイルを識別するために使用される一意のデータベース名です。データベース・パスワード・ファイルが Oracle ASM ディスク・グループに格納される場合、この引数は必須です。Oracle ASM パスワード・ファイルが ASM 引数を y に設定することによって作成される場合、この引数は無視されます。
FORMAT	(オプション)次のいずれかの値を指定します。 ● 12.2 (デフォルト)では、パスワード・ファイルが 12.2 形式で作成されます。この形式では、外部ユーザーへの管理権限の付与がサポートされ、管理ユーザーに対して SSL および Kerberos 認証が有効化されます。 ● 12 では、パスワード・ファイルが Oracle Database 12c 形式で作成されます。この形式では、SYSBACKUP、SYSDG および SYSKM 管理権限がサポートされます。

引数	説明
SYS	(オプション)この引数では、SYS ユーザーがパスワード、外部またはグローバルのいずれで認証されるかを指定します。 この引数は、y、n、password、external('sys-external-name')または global(sys-directory-DN)に設定できます。 パスワード・ファイル・エントリを移行するために SYS=y および INPUT_FILE が指定されている場合、SYS 管理ユーザーの新しいパスワードの入力を求められます。 password の場合は、SYS 管理ユーザーのパスワードの入力を求められます。 external('sys-external-name')の場合は、sys-external-name を SYS 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。 global(sys-directory-DN)の場合は、グローバル SYS ユーザーのディレクトリ・サービス名を指定します。
SYSBACKUP	(オプション) SYSBACKUP エントリを作成します。(オプション)この引数では、SYSBACKUP ユーザーが、パスワード、外部またはグローバルのいずれで認証されるかを指定します。 この引数は、y、n、password、external('sysbackup-external-name')または global(sysbackup-directory-DN)に設定できます。 password の場合は、SYSBACKUP 管理ユーザーのパスワードの入力を求められます。 external('sysbackup-external-name')の場合は、sysbackup-external-name を SYSBACKUP 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。 global(sysbackup-directory-DN)の場合は、グローバル SYSBACKUP ユーザーのディレクトリ・サービス名を指定します。
SYSDG	(オプション) SYSDG エントリを作成します。(オプション)この引数では、SYSDG ユーザーが、パスワード、外部またはグローバルのいずれで認証されるかを指定します。 この引数は、y、n、password、external('sysdg-external-name')または global(sysdg-directory-DN)に設定できます。 password の場合は、SYSDG 管理ユーザーのパスワードの入力を求められます。 external('sysdg-external-name')の場合は、sysdg-external-name を SYSDG 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。 global(sysdg-directory-DN)の場合は、グローバル SYSDG ユーザーのディレクトリ・サービ

引数	説明
	ス名を指定します。
SYSKM	(オプション) SYSKM エントリを作成します。(オプション)この引数では、SYSKM ユーザーが、パスワード、外部またはグローバルのいずれで認証されるかを指定します。 (オプション)この引数は、y、n、password、external('syskm-external-name')またはglobal(syskm-directory-DN)に設定できます。 password の場合は、SYSKM 管理ユーザーのパスワードの入力を求められます。 external('syskm-external-name')の場合は、syskm-external-name を SYSKM 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。 y の場合は、パスワード・ファイルに SYSKM エントリを作成します。パスワードの入力を求められます。パスワードは、作成されたパスワード・ファイルに保存されます。 n の場合、パスワード・ファイルに SYSKM エントリは作成されません。 ノート: SYSKM 引数の y および n 値は、Oracle Database 12c リリース 2 (12.2)で非推奨になり、将来のリリースでサポートされなくなる可能性があります。 global(syskm-directory-DN)の場合は、グローバル SYSKM ユーザーのディレクトリ・サービス名を指定します。
DELETE	(オプション) y の場合は、指定されたパスワード・ファイルを削除します。 n (デフォルト)の場合は、指定されたパスワード・ファイルを作成します。
INPUT_FILE	(オプション)入力パスワード・ファイルの名前。ORAPWD によって、入力ファイルのエントリが新しいパスワード・ファイルに移行されます。 この引数は、パスワード・ファイルをある形式から別の形式に変換するために使用できます。たとえば、12 形式から 12.2 形式に変換できます。 また、この引数を使用して、SYS 管理ユーザーのパスワードをリセットできます。 ORAPWD は、Oracle ASM ディスク・グループに格納されている入力パスワードを移行することはできません。
DESCRIBE	指定したパスワード・ファイルのプロパティを表示します(FORMAT 値(12.2 または 12)を含む)。

等号(=)文字の前後にスペースは使用できません。

ノート:

各外部名は一意にする必要があります。

次の項では、一部のORAPWDコマンドライン引数について詳しく説明します。

FILE

この引数には、作成するパスワード・ファイルの名前を設定します。これは必須の引数です。

Oracle ASM ディスク・グループにおける場所を指定した場合、データベース・パスワード・ファイルはクラスタ内のノード間で自動的に共有されます。Oracle ASM ディスク・グループを使用してパスワード・ファイルを格納し、Oracle Managed Files を使用しない場合は、フルパスを含むパスワード・ファイルの名前を指定する必要があります。Oracle Managed Files を使用する場合、フルパス名は必要ありません。

Oracle ASM ディスク・グループにおける場所を指定しない場合、パスワード・ファイルに必要なファイル名はオペレーティング・システムによって異なります。一部のオペレーティング・システムでは、パスワード・ファイルを特定の形式で作成して特定のディレクトリに格納する必要があります。また、環境変数を使用してパスワード・ファイルの名前と位置を指定できるオペレーティング・システムもあります。

次の表は、UNIX、Linux および Windows プラットフォームにおいてパスワード・ファイルに必要な名前と場所を示しています。その他のプラットフォームについては、プラットフォーム固有のマニュアルを参照してください。

プラットフォーム	必要な名前	必要な場所
UNIX および Linux	`orapwORACLE_SID`	`ORACLE_BASE/dbs`
Windows	`PWDORACLE_SID.ora`	`ORACLE_BASE¥database`

たとえば、SID が `orcldw` のデータベース・インスタンスの場合、パスワード・ファイルの名前は、Linux では `orapworcldw` となり、Windows では `PWDorcldw.ora` となります。

Oracle Real Application Clusters (Oracle RAC)環境で、環境変数がパスワード・ファイルのパスに設定される必要があるプラットフォームでは、各インスタンスの環境変数は同じパスワード・ファイルを指し示す必要があります。

ポリシー管理の Oracle RAC データベースまたは形式 `db_unique_name_n` (n は数字)の `ORACLE_SID` を使用した Oracle RAC One Node データベースの場合、`ORACLE_BASE/dbs/orapwsid_prefix` または

ORACLE_BASE¥database¥PWDsid_prefix.ora を使用して、パスワード・ファイルが最初に検索されます。パスワード・ファイルの検索には、sid_prefix (データベース名の最初の 8 文字)が使用されます。

ノート:

- パスワード・ファイルやパスワード・ファイルの位置を特定する環境変数の保護は、システムのセキュリティにとって非常に重要です。パスワード・ファイルや環境変数にアクセスできるユーザーは、接続に対するセキュリティを脅かす潜在的な可能性を持っています。
- Oracle Database 18c 以降では、デフォルト・ディレクトリにパスワード・ファイルが見つからない場合、データベースは、前のデータベース・リリースでデフォルト・ディレクトリであったディレクトリ内のパスワード・ファイルをチェックします。18c より前の Oracle Database リリースでは、パスワード・ファイルのデフォルト・ディレクトリは、UNIX および Linux プラットフォームの場合は ORACLE_HOME/dbs、Windows の場合は ORACLE_HOME¥database でした。

関連項目:

Oracle Managed Files の使用

FORCE

この引数を y に設定すると、既存のパスワード・ファイルを上書きできます。同じ名前のパスワード・ファイルがすでに存在している場合に、この引数を省略するかまたは n に設定すると、エラーが返されます。

ASM

この引数を y に設定すると、ORAPWD により Oracle ASM パスワード・ファイルが作成されます。FILE 引数には Oracle ASM ディスク・グループ内の場所を指定する必要があります。

この引数を n (デフォルト)に設定すると、ORAPWD によりパスワード・ファイルが作成されます。FILE 引数には Oracle ASM ディスク・グループ内の場所、またはオペレーティング・システムのファイル・システム内の場所を指定できます。DBUNIQUENAME 引数を指定した場合、パスワード・ファイルはデータベース・パスワード・ファイルになります。DBUNIQUENAME 引数を指定しない場合、パスワード・ファイルはデータベース・パスワード・ファイルまたは Oracle ASM パスワード・ファイルにできます。

関連項目:

Oracle ASM パスワード・ファイルの作成およびメンテナンスの詳細は、『Oracle Automatic Storage Management 管理者ガイド』を参照してください。

DBUNIQUENAME

この引数には、Oracle ASM ディスク・グループに作成するデータベース・パスワード・ファイルの一意のデータベース名を設定します。データベース・パスワード・ファイルの場所で更新するデータベース・リソースを識別します。

データベース・パスワード・ファイルがオペレーティング・システムのファイル・システムに作成される場合、この引数は必要ありません。

Oracle ASM パスワード・ファイルが ASM 引数を y に設定することによって作成される場合、この引数は無視されます。

FORMAT

この引数が 12.2 に設定されている場合(デフォルト)、ORAPWD は 12.2 形式のデータベース・パスワード・ファイルを作成します。パスワード・ファイルで外部ユーザーへの管理権限の付与と管理ユーザーに対する SSL および Kerberos 認証をサポートするには、12.2 形式が必要です。ユーザーに割り当てられるパスワード・プロファイルは、管理ユーザーにも強制されます。

この引数を 12 に設定すると、ORAPWD によりデータベース・パスワード・ファイルが Oracle Database 12c 形式で作成されます。Oracle Database 12c 形式は、パスワード・ファイルで SYSBACKUP、SYSDG および SYSKM 管理権限をサポートするために必要です。

この引数を legacy に設定すると、ORAPWD により Oracle Database 12c 以前の形式でデータベース・パスワード・ファイルが作成されます。パスワード・ファイルでは SYSDBA および SYSOPER 管理権限はサポートされますが、SYSBACKUP、SYSDG および SYSKM 管理権限はサポートされません。

SYS

パスワード・ファイル・エントリを移行するために SYS=Y および INPUT_FILE が指定されている場合、SYS 管理ユーザーの新しいパスワードの入力を求められます。

password の場合は、SYS 管理ユーザーのパスワードの入力を求められます。

external('sys-external-name')の場合は、sys-external-name を SYS 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。

global(sys-directory-DN)の場合は、グローバル SYS ユーザーのディレクトリ・サービス名

を指定します。

SYSBACKUP

password の場合は、SYSBACKUP 管理ユーザーのパスワードの入力を求められます。

external('sysbackup-external-name')の場合は、sysbackup-external-name を SYSDG 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。

global(sysbackup-directory-DN)の場合は、グローバル SYSBACKUP ユーザーのディレクトリ・サービス名を指定します。

SYSDG

password の場合は、SYSDG 管理ユーザーのパスワードの入力を求められます。

external('sysdg-external-name')の場合は、sysdg-external-name を SYSDG 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。

global(sysdg-directory-DN)の場合は、グローバル SYSDG ユーザーのディレクトリ・サービス名を指定します。

SYSKM

password の場合は、SYSKM 管理ユーザーのパスワードの入力を求められます。

external('syskm-external-name')の場合は、syskm-external-name を SYSKM 管理ユーザーの SSL または Kerberos 認証の外部名で置換します。

global(syskm-directory-DN)の場合は、グローバル SYSKM ユーザーのディレクトリ・サービス名を指定します。

DELETE

この引数を y に設定すると、ORAPWD により指定したパスワード・ファイルが削除されます。y を指定した場合は、FILE、ASM または DBUNIQUENAME を指定する必要があります。FILE を指定した場合は、ファイルを ASM ディスク・グループに格納する必要があります。

この引数を n (デフォルト)に設定すると、ORAPWD によりパスワード・ファイルが作成されます。

INPUT_FILE

この引数では入力パスワード・ファイルの名前を指定します。ORAPWD によって、入力ファイルのエント

リが新しいパスワード・ファイルに移行されます。この引数は、パスワード・ファイルをある形式から別の形式に変換できます。たとえば、12 形式から 12.2 形式に変換できます。

また、この引数を使用して、SYS 管理ユーザーのパスワードをリセットできます。

INPUT_FILE 引数を指定すると、ORAPWD により新しいエントリは作成されません。したがって、ORAPWD では次の引数が無視されます。

- PASSWORD
- SYSBACKUP
- SYSDG
- SYSKM

入力ファイルを指定し、入力ファイルが新しいパスワード・ファイルで置き換えられる場合は、FORCE を y に設定する必要があります。

ノート:

FORMAT 引数を指定しない場合、デフォルトでは、新しいパスワード・ファイルは入力ファイルから 12.2 形式で作成されます。

関連項目:

「管理権限」および「データベース・パスワード・ファイルへのユーザーの追加」

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.7.2 ORAPWDを使用したデータベース・パスワード・ファイルの作成

ORAPWDを使用してデータベース・パスワード・ファイルを作成できます。

データベース・パスワード・ファイルを作成するには:

- ORAPWDコマンドを実行します。

例1-20 Oracle ASMディスク・グループに格納されるデータベース・パスワード・ファイルの作成

次のコマンドでは、リリース12.2形式のorapworclという名前のデータベース・パスワード・ファイルが作成され、Oracle ASMディスク・グループに格納されます。データベース・パスワード・ファイルはOracle ASMディスク・グループに格納されるため、DBUNIQUENAME引数が必要です。

```
orapwd FILE='+DATA/orcl/orapworcl' DBUNIQUENAME='orcl' FORMAT=12.2
```

例1-21 SYSBACKUPエントリを使用したデータベース・パスワード・ファイルの作成

次の例は、データベース・パスワード・ファイルにSYSBACKUPエントリが作成されることを除き、例1-20に似ています。パスワー

ド・ファイルは、デフォルトでは12.2形式です。

```
orapwd FILE='+DATA/orcl/orapworcl' DBUNIQUENAME='orcl' SYSBACKUP=password FORMAT=12.2
```

例1-22 SYSおよびSYSKMの外部認証を使用したデータベース・パスワード・ファイルの作成

次の例は、SYSおよびSYSKM管理ユーザーに外部名を指定することを除いて例1-20と同様です。

```
orapwd FILE='+DATA/orcl/orapworcl' DBUNIQUENAME='orcl' FORMAT=12.2
sys=external('KerberosUserSYS@example.com')
syskm=external('KerberosUserSYSKM@example.com')
```

例1-23 ファイル・システムに格納されるデータベース・パスワード・ファイルの作成

次のコマンドでは、12.2形式のorapworclという名前のデータベース・パスワード・ファイルが作成され、オペレーティング・システムのファイル・システムのデフォルトの場所に格納されます。

```
orapwd FILE='/u01/oracle/dbs/orapworcl' FORMAT=12.2
```

例1-24 レガシー・データベース・パスワード・ファイルのOracle Database 12c形式への移行

次のコマンドでは、レガシー形式のデータベース・パスワード・ファイルが12.2形式に移行されます。パスワード・ファイルの名前はorapworclで、オペレーティング・システムのファイル・システムに格納されます。新しいデータベース・パスワード・ファイルで既存のデータベース・パスワード・ファイルが置き換えられます。このため、FORCEをyに設定する必要があります。

```
orapwd FILE='/u01/oracle/dbs/orapworcl' FORMAT=12.2
INPUT_FILE='/u01/oracle/dbs/orapworcl' FORCE=y
```

例1-25 SYS管理ユーザーのパスワードのリセット

次のコマンドは、SYS管理ユーザーのパスワードをリセットします。新しいデータベース・パスワード・ファイルで既存のデータベース・パスワード・ファイルが置き換えられます。このため、FORCEをyに設定する必要があります。

```
orapwd FILE='/u01/oracle/dbs/orapworcl' SYS=Y INPUT_FILE='/u01/oracle/dbs/orapworcl'
FORCE=y
```

SYS管理ユーザーの新しいパスワードを入力するように求められます。

例1-26 パスワード・ファイルの説明

次のコマンドは、orapworclパスワード・ファイルを説明します。

```
orapwd DESCRIBE FILE='orapworcl'
Password file Description : format=12.2
```

ノート:

データベースのパスワード・ファイルの名前または場所を変更した場合は、次のコマンドを実行して変更を有効にします。

```
SQL> ALTER SYSTEM FLUSH PASSWORDFILE_METADATA_CACHE;
```

このコマンドはメタデータ・キャッシュをフラッシュします。データベースへの後続のログインでは新しいパスワード・ファイルが使用されます。Oracle RAC 環境では、このコマンドによってすべての Oracle RAC データベース内のキャッシュがクリアされますが、変更がすべての Oracle RAC データベースに伝播されるまで、古いパスワード・ファイルを引き続き使

用できるデータベースが存在する可能性があります。

このコマンドを実行した後は、V$PASSWORDFILE_INFO ビューを問い合せることで変更を確認できます。

関連項目:

Oracle ASMディスク・グループでの共有パスワード・ファイルの管理の詳細は、Oracle Automatic Storage Management管理者ガイドを参照してください

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.7.3 データベース・パスワード・ファイルの共有と無効化

初期化パラメータREMOTE_LOGIN_PASSWORDFILEを使用して、データベース・パスワード・ファイルが複数のOracle Databaseインスタンス間で共有されるかどうかを制御します。このパラメータを使用して、パスワード・ファイル認証を無効にすることもできます。

パスワード・ファイルを共有またはパスワード・ファイル認証を無効にするには:

- REMOTE_LOGIN_PASSWORDFILE初期化パラメータを設定します。

REMOTE_LOGIN_PASSWORDFILE初期化パラメータを次のいずれかの値に設定できます。

- none: このパラメータをnoneに設定すると、Oracle Databaseはパスワード・ファイルが存在しない場合と同じように動作します。つまり、セキュリティで保護されていない接続では、権限付きの接続は許可されません。
- exclusive: (デフォルト) exclusiveのパスワード・ファイルは、1つのデータベースのみが使用できます。exclusiveのファイルのみを変更できます。exclusiveのパスワード・ファイルを使用すると、ユーザーの追加、変更および削除を行えます。また、ALTER USERコマンドを使用して、SYS、SYSBACKUP、SYSDGまたはSYSKMのパスワードを変更することもできます。

  exclusiveのパスワード・ファイルがOracle ASMディスク・グループに格納される場合、単一インスタンス・データベースまたはOracle Real Application Clusters (Oracle RAC)データベースの複数のインスタンスでパスワード・ファイルを使用できます。

  exclusiveのパスワード・ファイルは、オペレーティング・システムに格納されている場合、1つのデータベースの1つのインスタンスでのみ使用できます。
- shared: sharedパスワード・ファイルは、オペレーティング・システムに格納される場合にも、同じサーバーで稼働している複数のデータベース、またはOracle RACデータベースの複数のインスタンスで使用できます。sharedパスワード・ファイルは読取り専用であり、変更できません。したがって、sharedパスワード・ファイルにはユーザーを追加できません。ユーザーを追加しようとしたり、SYSのパスワードあるいは管理権限のあるユーザーのパスワードを変更しようとすると、エラーが生成されます。REMOTE_LOGIN_PASSWORDFILEがexclusiveに設定されている間に、管理権限を必要とするすべてのユーザーをパスワード・ファイルに追加する必要があります。すべてのユーザーを追加した後で、REMOTE_LOGIN_PASSWORDFILEをsharedに変更すると、ファイルを共有できます。

  このオプションは、1つのパスワード・ファイルで複数のデータベースを管理する場合に役立ちます。

  Oracle ASMパスワード・ファイルにはsharedを指定できません。

REMOTE_LOGIN_PASSWORDFILEがexclusiveまたはsharedに設定されているときに、パスワード・ファイルが欠落している場合は、REMOTE_LOGIN_PASSWORDFILEをnoneに設定した場合と同じです。

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.7.4 管理者パスワードとデータ・ディクショナリとの同期の維持

REMOTE_LOGIN_PASSWORDFILE初期化パラメータをnoneからexclusiveまたはsharedに変更した場合、データ・ディクショナリ内のパスワードとSYS以外(SYSDBA、SYSOPER、SYSBACKUP、SYSDGおよびSYSKM)の管理ユーザーのパスワード・ファイル内のパスワードを同じにする必要があります。

ノート:

Oracle Database 12c リリース 2 (12.2)以降、SYS ユーザーの認証は、データ・ディクショナリを使用せずにパスワード・ファイルのみを使用して発生します。

SYSDBA、SYSOPER、SYSBACKUP、SYSDGおよびSYSKMなどのSYS以外の管理ユーザーのパスワードを同期化するには、次に示すように、まずこれらのユーザーの権限を取り消して再度付与する必要があります。

1. SYSDBA権限を付与されているすべてのユーザーを特定します。

```
SELECT USERNAME FROM V$PWFILE_USERS WHERE USERNAME != 'SYS' AND SYSDBA='TRUE';
```

2. これらのユーザーのSYSDBA権限を取り消して再度付与します。

```
REVOKE SYSDBA FROM non-SYS-user;
GRANT SYSDBA TO non-SYS-user;
```

3. SYSOPER権限を付与されているすべてのユーザーを特定します。

```
SELECT USERNAME FROM V$PWFILE_USERS WHERE USERNAME != 'SYS' AND SYSOPER='TRUE';
```

4. これらのユーザーのSYSOPER権限を取り消して再度付与します。

```
REVOKE SYSOPER FROM non-SYS-user;
GRANT SYSOPER TO non-SYS-user;
```

5. SYSBACKUP権限を付与されているすべてのユーザーを特定します。

```
SELECT USERNAME FROM V$PWFILE_USERS WHERE USERNAME != 'SYS' AND SYSBACKUP
='TRUE';
```

6. これらのユーザーのSYSBACKUP権限を取り消して再度付与します。

```
REVOKE SYSBACKUP FROM non-SYS-user;
GRANT SYSBACKUP TO non-SYS-user;
```

7. SYSDG権限を付与されているすべてのユーザーを特定します。

```
SELECT USERNAME FROM V$PWFILE_USERS WHERE USERNAME != 'SYS' AND SYSDG='TRUE';
```

8. これらのユーザーのSYSDG権限を取り消して再度付与します。

```
REVOKE SYSDG FROM non-SYS-user;
GRANT SYSDG TO non-SYS-user;
```

9. SYSKM権限を付与されているすべてのユーザーを特定します。

```
SELECT USERNAME FROM V$PWFILE_USERS WHERE USERNAME != 'SYS' AND SYSKM='TRUE';
```

10. これらのユーザーのSYSKM権限を取り消して再度付与します。

```
REVOKE SYSKM FROM non-SYS-user;
GRANT SYSKM TO non-SYS-user;
```

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.7.5 データベース・パスワード・ファイルへのユーザーの追加

ユーザーにSYSDBA、SYSOPER、SYSBACKUP、SYSDGまたはSYSKM管理権限を付与すると、そのユーザーの名前と権限情報がデータベース・パスワード・ファイルに追加されます。

ユーザーがこれらの権限のうち1つでも持っている間は、そのユーザーの名前がパスワード・ファイルに残っています。これらのすべての権限を取り消すと、Oracle Databaseによって、そのユーザーはパスワード・ファイルから削除されます。

ノート:

SYSBACKUP、SYSDG または SYSKM 管理権限をサポートするには、FORMAT=12.2 または FORMAT=12 引数を指定してパスワード・ファイルを作成する必要があります。

パスワード・ファイルを作成した新規ユーザーの追加方法

パスワード・ファイルを作成して新規ユーザーを追加するには、次の手順を実行します。

1. 「ORAPWDを使用したデータベース・パスワードの作成」に記載されている指示に従って、パスワード・ファイルを作成します。
2. REMOTE_LOGIN_PASSWORDFILE初期化パラメータをexclusiveに設定します。(デフォルト値。)

   これらの権限を付与しようとしたときに初期化パラメータREMOTE_LOGIN_PASSWORDFILEが正しく設定されていない場合、Oracle Databaseによってエラーが発行されます。

ノート:

REMOTE_LOGIN_PASSWORDFILE は静的な初期化パラメータであるため、変更するにはデータベースを再起動する必要があります。

3. 次の例に示すように、SYSDBA権限で接続し、プロンプトが表示されたらSYSのパスワードを入力します。

```
CONNECT SYS AS SYSDBA
```

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

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.7.6 管理権限の付与と取消し

管理権限を付与するにはGRANT文を使用します。管理権限を取り消すにはREVOKE文を使用します。

ユーザーにSYSDBA、SYSOPER、SYSBACKUP、SYSDGまたはSYSKM管理権限を付与するには:

- GRANT文を実行します。

たとえば:

```
GRANT SYSDBA TO mydba;
```

ユーザーから管理権限を取り消すには:

- REVOKE文を実行します。

たとえば:

```
REVOKE SYSDBA FROM mydba;
```

管理権限を付与するGRANTに指定された場合、WITH ADMIN OPTIONは無視され、次のルールが適用されます。

- 現在SYSDBAとして接続しているユーザーは、別のユーザーに他の任意の管理権限を付与して、別のユーザーの管理権限を取り消すことができます。
- 現在SYSOPERとして接続しているユーザーは、別のユーザーに管理権限を付与できず、別のユーザーの管理権限を取り消すことができません。
- 現在SYSBACKUPとして接続しているユーザーは、別のユーザーのSYSBACKUP管理権限を取り消すことができます。
- 現在SYSDGとして接続しているユーザーは、別のユーザーのSYSDG管理権限を取り消すことができます。
- 現在SYSKMとして接続しているユーザーは、別のユーザーのSYSKM管理権限を取り消すことができます。

ロールはデータベースが起動してから使用できるようになるため、管理権限はロールには付与できません。データベース管理権限を、オペレーティング・システムのロールと混同しないでください。

関連項目:

管理権限の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.7.7 データベース・パスワード・ファイル・メンバーの表示

V$PWFILE_USERSビューには、管理権限を付与されたユーザーに関する情報が含まれます。

管理権限を付与されたユーザーを判別するには:

- V$PWFILE_USERSビューを問い合せます。

関連項目:

V$PWFILE_USERSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.7.8 データベース・パスワード・ファイルの削除

不要になった場合はデータベース・パスワードを削除できます。

ユーザーを認証するためのデータベース・パスワード・ファイルが不要になったと判断した場合は、それを削除します。

- データベース・パスワード・ファイルを削除し、オプションでREMOTE_LOGIN_PASSWORDFILE初期化パラメータをnoneにリセットします。

このファイルを削除した後は、オペレーティング・システムによって認証されたユーザーのみがSYSDBA、SYSOPER、SYSBACKUP、SYSDGまたはSYSKMデータベース管理の操作を実行できます。

親トピック: データベース・パスワード・ファイルの作成とメンテナンス

1.8 データ・ユーティリティ

Oracle Databaseのデータのメンテナンスに役立つOracleユーティリティが用意されています。

SQL*Loader

SQL*Loaderは、Oracle Databaseのデータベース管理者とその他のユーザーの両方が使用します。これは、標準的なオペレーティング・システム・ファイル(テキスト形式やCデータ形式のファイルなど)からデータベース表にデータをロードします。

エクスポート・ユーティリティとインポート・ユーティリティ

データ・ポンプ・ユーティリティを使用すると、データをアーカイブしたり、Oracle Database間でデータを移動できます。また、従来のインポート(IMP)およびエクスポート(EXP)ユーティリティを使用して、以前のリリースとの間でデータをインポートおよびエクスポートできます。

関連項目:

- SQL*Loaderの詳細は、『Oracle Databaseユーティリティ』を参照してください。
- データ・ポンプの詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: データベース管理スタート・ガイド

2 Oracle Databaseの作成および構成

データベースの計画を作成したら、グラフィカル・ツールまたはSQLコマンドを使用してデータベースを作成できます。

- Oracle Databaseの作成について
  通常は、Oracle Databaseソフトウェアのインストール時にデータベースを作成します。ただし、インストール後にもデータベースを作成できます。
- データベースを作成する前の考慮点
  データベースの作成では、いくつかのオペレーティング・システム・ファイルを準備することで、それらのファイルをOracle Databaseとしてまとめて動作できるようにします。データベースは、データファイルの数やアクセスするインスタンスの数にかかわらず、1回のみ作成します。データベースの作成では、既存のデータベース内の情報を消去し、同じ名前と物理構造を持つ新規データベースを作成することもできます。
- DBCAを使用したデータベースの作成
  Oracle Database Configuration Assistant (DBCA)は、Oracleデータベースを作成および構成するためのツールです。
- CREATE DATABASE文を使用したデータベースの作成
  CREATE DATABASE SQL文の使用は、Oracle Database Configuration Assistant (DBCA)を使用するよりも手動操作の多いデータベース作成アプローチです。DBCAのかわりにこの文を使用する利点は、スクリプト内からデータベースを作成できることです。
- CREATE DATABASE文の句の指定
  CREATE DATABASE文を実行すると、Oracle Databaseはいくつかの操作を実行します。実際の操作は、CREATE DATABASE文で指定した句、および初期化パラメータの設定に応じて実行されます。
- 初期化パラメータの指定
  新規データベースを作成する前に、基本初期化パラメータを追加または編集できます。
- サーバー・パラメータ・ファイルを使用した初期化パラメータの管理
  Oracle Databaseの初期化パラメータは、テキスト形式の初期化パラメータ・ファイルに格納されていました。管理性を向上させるために、データベースを起動および停止している間も持続するバイナリ形式のサーバー・パラメータ・ファイルによる初期化パラメータのメンテナンスを選択できます。
- データベース・サービスでのアプリケーション・ワークロードの管理
  データベース・サービスは、1つ以上のデータベース・インスタンスをある単位にまとめて表現したものです。サービスを使用すれば、データベースのワークロードをグループ化し、特定の作業リクエストを適切なインスタンスにルーティングできます。
- Oracle DatabaseのStandard Edition高可用性の管理
  Standard Edition高可用性機能は、Oracle Clusterwareを使用したOracle Database Standard Edition 2単一インスタンス・データベースに向けて計画外停止に対する保護を提供します。
- データベースを作成した後の考慮点
  データベースの作成後、インスタンスは実行されたままになり、データベースはオープンされて通常のデータベースの用途に使用できます。データベースの作成後に特定のアクションを実行することが必要な場合があります。
- データベースのクローニング
  この項では、Oracleデータベースの様々なクローニング方法について説明します。
- データベースの削除
  データベースの削除には、データファイル、オンラインREDOログ、制御ファイルおよび初期化パラメータ・ファイルの削除も含まれます。
- データベースのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューに対してデータベースのコンテンツと構造に関する情報を問い合せることができます。

- サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンス
  この項では、Database Configuration Assistant (DBCA)サイレント・モード・コマンドの構文とオプションに関して詳しく説明します。

関連項目:

- 基礎になるオペレーティング・システム・ファイルがOracle Databaseサーバーによって自動的に作成および管理されるデータベースの作成方法の詳細は、「Oracle Managed Filesの使用」を参照してください
- Oracle Real Application Clusters(Oracle RAC)環境でのデータベース作成に関する情報は、使用しているプラットフォーム固有のOracle Real Application Clustersインストレーション・ガイドを参照してください。
- フリート・パッチ適用およびプロビジョニング(以前のデータベース・リリースでは高速ホーム・プロビジョニングと呼ばれていた)を使用したデータベース作成の詳細は、『Oracle Clusterware管理およびデプロイメント・ガイド』を参照してください。

親トピック: 基本データベース管理

2.1 Oracle Databaseの作成について

通常は、Oracle Databaseソフトウェアのインストール時にデータベースを作成します。ただし、インストール後にもデータベースを作成できます。

インストール後にデータベースを作成する理由には、次のものがあります。

- Oracle Universal Installer(OUI)を使用してソフトウェアのみをインストールして、データベースを作成しなかったため。
- 既存のOracle Databaseと同じホスト・コンピュータに別のデータベース(およびデータベース・インスタンス)を作成するため。この場合、この章では、新しいデータベースが既存のデータベースと同じOracleホームを使用すると仮定します。OUIを再度実行すれば、新しいOracleホームにデータベースを作成することもできます。
- データベースのコピー(クローン)を作成するため。

データベースを作成する具体的な方法には、次の方法があります。

- Database Configuration Assistant(DBCA)グラフィカル・ツールを使用。

  「DBCAを使用したデータベースの作成」を参照してください
- CREATE DATABASE SQL文を使用

  「CREATE DATABASE文を使用したデータベースの作成」を参照してください

親トピック: Oracle Databaseの作成および構成

2.2 データベースを作成する前の考慮点

データベースの作成では、いくつかのオペレーティング・システム・ファイルを準備することで、それらのファイルをOracle Databaseとしてまとめて動作できるようにします。データベースは、データファイルの数やアクセスするインスタンスの数にかかわらず、1回のみ作成します。データベースの作成では、既存のデータベース内の情報を消去し、同じ名前と物理構造を持つ新規データベースを作成することもできます。

- データベース作成計画
  データベースの作成を準備するには、調査と綿密な計画が必要です。
- 文字セットの選択について
  データベースに対して適切な文字セットを選択することが重要です。データベース文字セットとして、AL32UTF8をお薦めします。
- 読取り専用モードのOracleホームの構成について
  Oracle Databaseリリース18c以降では、読取り専用Oracleホームをソフトウェア・イメージとしてデプロイすることによって、複数のデータベース・サーバー間でのソフトウェアのパッチ適用や一括ロールアウトを簡略化できます。
- データベース作成の前提条件
  Oracle Databaseが正常に作成されるようにするために、データベースの前提条件を確認してください。

親トピック: Oracle Databaseの作成および構成

2.2.1 データベース作成計画

データベースの作成を準備するには、調査と綿密な計画が必要です。

表2-1に、推奨する処理を示します。

表2-1 データベース計画タスク

処置	関連情報
データベース表と索引を計画し、それらが必要とする領域を見積ります。	Oracle Database の構造と記憶域 スキーマ・オブジェクト
データベースを構成する基礎になるオペレーティング・システム・ファイルのレイアウトを計画します。ファイルを適切に分散すると、ファイル・アクセス時の I/O が分散されるため、データベースのパフォーマンスを大幅に向上できます。Oracle ソフトウェアをインストールしてデータベースを作成するときに I/O を分散させるには、いくつかの方法があります。たとえば、REDO ログ・ファイルを異なるディスクに配置したり、ストライプ化を使用できます。競合が少なくなるようにデータファイルを配置できます。また、データの密度(データ・ブロック当たりの行数)を制御できます。高速リカバリ領域を作成する場合は、データファイルとは異なる記憶デバイスに配置することをお薦めします。 この計画タスクを大幅に簡素化するには、データベース記憶域を構成するオペレーティング・システム・ファイルの作成および管理に、Oracle Managed Files および自動ストレージ管理の使用を検討します。	Oracle Managed Files の使用 Oracle Automatic Storage Management 管理者ガイド Oracle Database パフォーマンス・チューニング・ガイド 該当の Oracle Database インストレーション・ガイドなど、使用しているオペレーティング・システム固有の Oracle マニュアル。
グローバル・データベース名を選択します。ネットワーク構造内のデータベースの名前と位置で構成されます。グローバル・データベース名を作成するには、DB_NAME と DB_DOMAIN の両方の初期化パラメータを設定します。	「グローバル・データベース名の決定」
初期化パラメータ・ファイルに含まれる初期化パラメータについて理解します。特にサー	「初期化パラメータと初期化パラメー

処置	関連情報
バー・パラメータ・ファイルの概要と操作について理解します。サーバー・パラメータ・ファイルを使用すると、初期化パラメータをサーバー側のディスク・ファイルに永続的に格納して管理できます。	タ・ファイルについて」 「サーバー・パラメータ・ファイルの概要」 Oracle Database リファレンス
データベース・キャラクタ・セットを選択します。 データ・ディクショナリ内のデータも含め、すべての文字データは、そのデータベースの文字セットに格納されます。データベースの作成時には、データベース文字セットを指定します。 詳細は、「文字セットの選択について」を参照してください。	Oracle Database グローバリゼーション・サポート・ガイド
データベースでサポートする必要のあるタイム・ゾーンを検討します。 Oracle Database では、2 つのタイム・ゾーン・ファイルのいずれかが、有効なタイム・ゾーンのソースとして使用されます。デフォルトのタイム・ゾーン・ファイルは `timezlrg_11.dat` です。これには、小さいタイム・ゾーン・ファイル `timezone_11.dat` より、多くのタイム・ゾーンが含まれています。	「データベースのタイム・ゾーン・ファイルの指定」
データベースの標準ブロック・サイズを選択します。このサイズはデータベースの作成時に `DB_BLOCK_SIZE` 初期化パラメータによって指定しますが、データベースの作成後は変更できません。 標準ブロック・サイズは、SYSTEM 表領域およびその他の大部分の表領域で使用されます。また、表領域の作成時に、最大 4 つの非標準ブロック・サイズを指定できます。	「データベース・ブロック・サイズの指定」
セクター・サイズが 4KB のディスク上にオンライン REDO ログ・ファイルを格納する場合は、REDO ログのブロック・サイズを手動で指定する必要があるかどうかを確認します。	「REDO ログ・ファイルのブロック・サイズの計画」
SYSAUX 表領域の適切な初期サイズを決定します。	「SYSAUX 表領域について」
SYSTEM 以外のユーザーに対してはデフォルトの表領域を使用し、データベース・オブジェクトが SYSTEM 表領域に誤って保存されないようにします。	「デフォルト永続表領域の作成」
UNDO データを管理するために、UNDO 表領域を使用します。	UNDO の管理
目的のデータベースに読取り専用 Oracle ホームを構成するか、読取り/書込み	読取り専用モードの Oracle ホーム

処置	関連情報
Oracle ホームを構成するかを検討します。	の構成について
バックアップおよびリカバリ計画を作成し、データベースを障害から保護します。重要な点として、多重化による制御ファイルの保護、適切なバックアップ・モードの選択、およびオンライン REDO ログ・ファイルとアーカイブ REDO ログ・ファイルの管理があげられます。	REDO ログの管理 アーカイブ REDO ログ・ファイルの管理 制御ファイルの管理 Oracle Database バックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド
インスタンスの起動と停止、データベースのマウントとオープンの原理およびオプションについて理解します。	起動と停止

親トピック: データベースを作成する前の考慮点

2.2.2 文字セットの選択について

データベースに対して適切な文字セットを選択することが重要です。データベース文字セットとして、AL32UTF8をお薦めします。

AL32UTF8とは、Unicode標準であるUTF-8エンコードに対してOracleで使用している名前です。Unicode標準は、現在世界で使用されている言語のほとんどをサポートする汎用文字セットです。Unicode標準を使用することは、データベース処理を含め、すべての多言語テクノロジにおいて不可欠です。

データベースが作成され、本番データが累積された後にデータベース文字セットを変更することは、時間がかかる複雑なプロジェクトになります。そのため、インストール時に適切な文字セットを選択することが重要です。データベースに現在多言語データが格納されていなくても、数年内に多言語データを格納する予定がある場合、通常は、データベース文字セットにAL32UTF8を選択することのみが、適切な判断となります。Unicodeの普遍性と柔軟性は、通常はシングルバイト文字セットと比較した場合のテキスト処理速度の若干の低下や非Unicode文字セットよりも高い非ASCIIテキストの記憶域要件などの関連する追加コストを上回る価値があります。

AL32UTF8を使用せず、ベンダー要件によって選択が制限されない場合は、データベースに推奨としてリストされている文字セットの1つを使用することをお薦めします。推奨される文字セットは、最新のクライアント・オペレーティング・システムの要件に基づいて選択されたものです。Oracle Universal Installer (OUI)は推奨されるリストのみを提示し、非推奨文字セットを選択するにはDatabase Configuration Assistant (DBCA)を別途使用する必要があります。また、DBCAのデフォルトのデータベース作成構成では、推奨される文字セットのみ選択できます。非推奨文字セットを選択するには、DBCAまたは CREATE DATABASE文の高度な構成モードを使用する必要があります。

文字セットの選択肢がOUIまたはDBCAインストール・モードで表示されない場合は、別の文字セットを持つカスタム・データベース・テンプレートが選択されていないかぎり、AL32UTF8がデータベース文字セットとして使用されます。


ノート:

- データベースの文字セットには、AL32UTF8 を使用することをお薦めします。AL32UTF8 は、Unicode の UTF-8 エンコーディングの適切な実装です。Oracle Database12c リリース 2 以降では、Oracle Universal Installer (OUI)および Oracle Database Configuration Assistant (DBCA)を使用してデータベースを作成する際、デフォルトのデータベース文字セットとして AL32UTF8 が使用されます。

- ASCII ベースのプラットフォーム上のデータベースには、ASCII ベースの文字セットのみ選択できます。

注意:

Oracle8i リリース 1 (8.1.7)以前の Oracle Database クライアントおよびサーバーとの互換性のために必要であるか、またはアプリケーション・ベンダーから明示的に要求された場合以外は、UTF8 をデータベース文字セットとして使用しないでください。名称は似ていますが、UTF8 は Unicode エンコーディング UTF-8 の適切な実装ではありません。UTF-8 処理が予期されている場合に UTF8 文字セットが使用されると、データの消失およびセキュリティの問題が発生する場合があります。このことは、XML や URL アドレスなどの Web 関連データに特に該当します。

AL32UTF8 および UTF8 文字セットは、最大文字幅が異なるため、相互に互換性がありません。AL32UTF8 の最大文字幅は 4 バイトで、UTF8 の最大文字幅は 3 バイトです。

関連項目:

データベースに推奨される文字セットの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください

親トピック: データベースを作成する前の考慮点

2.2.3 読取り専用モードのOracleホームの構成について

Oracle Databaseリリース18c以降では、読取り専用Oracleホームをソフトウェア・イメージとしてデプロイすることによって、複数のデータベース・サーバー間でのソフトウェアのパッチ適用や一括ロールアウトを簡略化できます。

読取り専用Oracleホーム(ORACLE_HOME)では、Oracleホーム・ディレクトリ内でのファイルの作成および変更が防止されます。Oracleホームを読取り専用モードで構成するには、最初にソフトウェアのみのデプロイメントを使用してOracle Databaseソフトウェアをインストールし、リスナーおよびデータベースを作成する前に読取り専用Oracleホームとして構成します。

従来の読取り/書込みのOracleホームにはインスタンス固有のファイルが含まれているため、それらのファイルにパッチを適用する場合は、各ファイルに個別にパッチを適用する必要があります。ただし、Oracleホームが読取り専用の場合は、インスタンス固有のファイルがOracleホームではなくOracleベース・ディレクトリ(ORACLE_BASE)に別個に保存されます。この構成によって、読取り専用Oracleホームは、静的ファイルのみが格納されるため、複数のデータベース・サーバー間で共有できるソフトウェア・イメージとして使用できます。このオプションを使用すると、複数のデータベース・サーバーにパッチを配布する際に更新する必要のあるOracleホーム・イメージは1つのみになるため、パッチ適用と一括ロールアウトが簡略化されます。

従来のORACLE_BASEディレクトリおよびORACLE_HOMEディレクトリとは異なり、読取り専用Oracleホームには次の追加のディレクトリが存在します。

- ORACLE_BASE_HOME: ORACLE_BASEディレクトリ内のサブディレクトリです。このディレクトリには、ユーザー固有のファイル、インスタンス固有のファイルおよびログ・ファイルが含まれています。

- ORACLE_BASE_CONFIG: このディレクトリは、ORACLE_BASEディレクトリと同じものです。このディレクトリには、イ

ンスタンス固有の動的ファイル(構成ファイルなど)が含まれています。

関連トピック

- Oracle Databaseのインストレーション・ガイド

関連項目:

読取り専用モードのOracleホームの構成の詳細は、目的のプラットフォームに固有のOracle Databaseインストレーション・ガイドで、「読取り専用Oracleホームの構成」を参照してください。

親トピック: データベースを作成する前の考慮点

2.2.4 データベース作成の前提条件

Oracle Databaseが正常に作成されるようにするために、データベースの前提条件を確認してください。

新しいデータベースを作成するには、次の前提条件を満たす必要があります。

- 必要なOracleソフトウェアがインストールされていること。これには、オペレーティング・システム固有の各環境変数の設定、およびソフトウェアとデータベース・ファイルのディレクトリ構造の設定が含まれます。
- Oracle Databaseインスタンスを起動するために十分なメモリーが使用可能であること。
- Oracle Databaseを実行するコンピュータ上に、設計したデータベースのための十分なディスク記憶域が使用可能であること。

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

親トピック: データベースを作成する前の考慮点

2.3 DBCAを使用したデータベースの作成

Oracle Database Configuration Assistant (DBCA)は、Oracleデータベースを作成および構成するためのツールです。

- DBCAを使用したデータベースの作成
  Database Configuration Assistant (DBCA)のほうが自動化されており、DBCAが完了するとデータベースが使用可能状態になるので、この方法でデータベースを作成することをお薦めします。
- 対話型DBCAを使用したデータベースの作成について
  データベースを最も簡単に作成する方法は、Database Configuration Assistant (DBCA)を使用することです。
- 非対話型(サイレント) DBCAを使用したデータベースの作成について
  Database Configuration Assistant (DBCA)の非対話型(サイレント)モードを使用してデータベースを作成できます。

親トピック: Oracle Databaseの作成および構成

2.3.1 DBCAを使用したデータベースの作成について

Database Configuration Assistant(DBCA)はほとんど自動化されており、DBCAが完了するとデータベースが使用可能状態になるので、DBCAを使用してデータベースを作成することをお薦めします。

選択したインストール・タイプによっては、Oracle Universal Installer(OUI)からDBCAを起動できます。また、Oracle Databaseをインストール後は、スタンドアロン・ツールとして、いつでもDBCAを起動できます。

DBCAは、対話型モードまたは非対話型(サイレント)モードで実行できます。対話型モードには、データベースを作成して構成するためのグラフィカル・インタフェースおよびガイド付きワークフローが用意されています。非対話型(サイレント)モードでは、データベース作成スクリプトを記述できます。DBCAを非対話型(サイレント)モードで実行するには、コマンドライン引数またはレスポンス・ファイル、あるいはその両方を指定します。

親トピック: DBCAを使用したデータベースの作成

2.3.2 対話型DBCAを使用したデータベースの作成について

データベースを最も簡単に作成する方法は、Database Configuration Assistant (DBCA)を使用することです。

対話型DBCAを使用したデータベースの作成方法の詳細は、『Oracle Database 2日でデータベース管理者』を参照してください。

親トピック: DBCAを使用したデータベースの作成

2.3.3 非対話型(サイレント) DBCAを使用したデータベースの作成について

Database Configuration Assistant (DBCA)の非対話型(サイレント)モードを使用してデータベースを作成できます。

関連項目:

非対話型(サイレント)モードのDBCAの使用方法の詳細は、サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンスを参照してください。

親トピック: DBCAを使用したデータベースの作成

2.4 CREATE DATABASE文を使用したデータベースの作成

CREATE DATABASE SQL文の使用は、Oracle Database Configuration Assistant (DBCA)を使用するよりも手動操作の多いデータベース作成アプローチです。DBCAのかわりにこの文を使用する利点は、スクリプト内からデータベースを作成できることです。

- CREATE DATABASE文を使用したデータベースの作成について
  CREATE DATABASE文を使用する場合は、実行可能なデータベースを作成する前に他の処理が必要です。この処理には、データ・ディクショナリ表のビューの作成、標準PL/SQLパッケージのインストールなどがあります。これらの処理には、提供されたスクリプトを実行します。
- ステップ1: インスタンス識別子(SID)の指定
  ORACLE_SID環境変数は、後で作成して同じホスト・コンピュータで同時に実行する他のOracle Databaseインスタンスと区別するために使用されます。
- ステップ2: 必要な環境変数が設定されていることの確認
  プラットフォームによっては、環境変数を設定するか少なくとも正しく設定されていることを確認するまで、SQL*Plus(後のステップで必要)を起動できない場合があります。
- ステップ3: データベース管理者の認証方式の選択
  データベースを作成するには、作成するユーザーが認証を受け、適切なシステム権限が付与されている必要があります。
- ステップ4: 初期化パラメータ・ファイルの作成

Oracleインスタンスの起動時に、初期化パラメータ・ファイルが読み込まれます。このファイルには、テキスト・エディタで作成して変更できるテキスト・ファイルか、データベースにより作成されて動的に変更されるバイナリ・ファイルを使用できます。バイナリ・ファイル(こちらの使用を推奨)は、サーバー・パラメータ・ファイルと呼ばれます。ここで示すステップでは、テキスト形式の初期化パラメータ・ファイルを作成します。後述のステップで、テキスト・ファイルからサーバー・パラメータ・ファイルを作成します。

- ステップ5: (Windowsの場合のみ)インスタンスの作成
  Windowsプラットフォームでは、インスタンスが存在しない場合には、インスタンスに接続する前に手動で作成する必要があります。ORADIMコマンドでは、新しいWindowsサービスを作成することでOracle Databaseインスタンスを作成します。
- ステップ6: インスタンスへの接続
  SQL*Plusを起動して、SYSDBA管理権限でOracle Databaseインスタンスに接続します。
- ステップ7: サーバー・パラメータ・ファイルの作成
  サーバー・パラメータ・ファイルを使用すると、ALTER SYSTEMコマンドを使用して初期化パラメータを変更でき、変更内容はデータベースを停止して起動した後も持続します。サーバー・パラメータ・ファイルは、編集済のテキスト形式の初期化ファイルから作成します。
- ステップ8: インスタンスの起動
  データベースをマウントせずにインスタンスを起動します。
- ステップ9: CREATE DATABASE文の発行
  新しいデータベースを作成するには、CREATE DATABASE文を使用します。
- ステップ10: 追加の表領域の作成
  データベースを稼働させるには、アプリケーション・データ用に追加の表領域を作成する必要があります。
- ステップ11: スクリプトの実行によるデータ・ディクショナリ・ビューの作成
  データ・ディクショナリ・ビュー、シノニムおよびPL/SQLパッケージの作成およびSQL*Plusの適切な稼働に必要なスクリプトを実行します。
- ステップ12: (オプション)スクリプトの実行による追加オプションのインストール
  必要に応じて、他のスクリプトも実行できます。実行するスクリプトは、使用またはインストール対象として選択する機能とオプションによって異なります。
- ステップ13: データベースのバックアップ
  メディア障害が発生した場合にリカバリするための完全なファイル・セットが確実に存在するように、データベースの全体バックアップを作成してください。
- ステップ14: (オプション)インスタンスの自動起動の有効化
  ホスト・コンピュータの再起動時に、Oracleインスタンスが自動的に起動されるように構成する必要がある場合があります。

親トピック: Oracle Databaseの作成および構成

2.4.1 CREATE DATABASE文を使用したデータベースの作成について

CREATE DATABASE文を使用する場合は、実行可能なデータベースを作成する前に他の処理が必要です。この処理には、データ・ディクショナリ表のビューの作成、標準PL/SQLパッケージのインストールなどがあります。これらの処理には、提供されたスクリプトを実行します。

データベース作成用の既存のスクリプトがある場合は、Oracle Databaseの新機能を活用するようにこれらのスクリプトを編集することを検討してください。

この項の手順が適用できるのは、単一インスタンスのインストールの場合のみです。Oracle RACデータベースの作成手順の詳

細は、使用しているプラットフォーム固有のOracle Real Application Clusters(Oracle RAC)インストレーション・ガイドを参照してください。

ノート:

- 単一インスタンスとは、単一のホスト・コンピュータに存在できる Oracle インスタンスが 1 つのみであるという意味ではありません。実際には、複数の Oracle インスタンス(および関連するデータベース)を、単一のホスト・コンピュータで実行できます。単一インスタンスのデータベースとは、一度に 1 つの Oracle インスタンスのみがアクセスするデータベースのことですが、これに対して、Oracle RAC データベースには、複数ノードの複数 Oracle インスタンスが同時にアクセスします。
- Oracle Database 12c リリース 2 (12.2)以降では、読取り専用インスタンスと読取り/書込みインスタンスが単一の Oracle RAC データベース内で共存できます。この構成はパラレル問合せのスケーラビリティのために役に立ちます。

ヒント:

Oracle Automatic Storage Management(Oracle ASM)を使用してディスク記憶域を管理している場合は、これらのステップを実行する前に、Oracle ASM インスタンスを起動してディスク・グループを構成する必要があります。『Oracle Automatic Storage Management 管理者ガイド』を参照してください。

関連項目:

- Oracle RACの詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照
- 単一のOracle RACデータベース内に共存する読取り専用インスタンスと読取り/書込みインスタンスの構成の詳細は、Oracle Clusterware管理およびデプロイメント・ガイドを参照してください

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.2 ステップ1: インスタンス識別子(SID)の指定

ORACLE_SID環境変数は、後で作成して同じホスト・コンピュータで同時に実行する他のOracle Databaseインスタンスと区別するために使用されます。

1. インスタンスの一意のOracleシステム識別子(SID)を決定します。
2. コマンド・ウィンドウを開きます。

ノート:

後続のステップでは、このコマンド・ウィンドウを使用します。

3. ORACLE_SID環境変数を設定します。

ORACLE_SIDでの有効な文字に関する制約はプラットフォーム固有です。一部のプラットフォームでは、SIDの大/小文字が区別されます。

ノート:

一般的には、SID はデータベース名と等しくなるように設定します。データベース名に指定できるのは最大 8 文字までです。詳細は、『Oracle Database リファレンス』の DB_NAME 初期化パラメータに関する説明を参照してください。

次のUNIXおよびLinuxオペレーティング・システムの例では、「ステップ6: インスタンスへの接続」で接続するインスタンスのSIDを設定しています。

- Bourne、BashまたはKornシェル:

```
ORACLE_SID=mynewdb
export ORACLE_SID
```

- Cシェルの場合:

```
setenv ORACLE_SID mynewdb
```

次の例では、Windowsオペレーティング・システム用のSIDを設定しています。

```
set ORACLE_SID=mynewdb
```

関連項目:

Oracleインスタンスに関するバックグラウンド情報は、『Oracle Database概要』を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.3 ステップ2: 必要な環境変数が設定されていることの確認

プラットフォームによっては、環境変数を設定するか少なくとも正しく設定されていることを確認するまで、(後のステップで必要な)SQL*Plusを起動できない場合があります。

- 必要な環境変数を設定します。

たとえば、ほとんどのプラットフォームでは、ORACLE_SIDとORACLE_HOMEを設定する必要があります。さらに、PATH変数にORACLE_HOME/binディレクトリを含めるように設定することもお薦めします。UNIXおよびLinuxプラットフォームでは、これらの環境変数を手動で設定する必要があります。Windowsプラットフォームでは、Windowsレジストリ内のORACLE_HOMEとORACLE_SIDにOUIが値を自動的に割り当てます。インストール時にデータベースを作成しなかった場合は、レジストリ内のORACLE_SIDをOUIが設定しないため、後でデータベースを作成するときにORACLE_SID環境変数を設定する必要があります。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.4 ステップ3: データベース管理者の認証方式の選択

データベースを作成するには、作成するユーザーが認証を受け、適切なシステム権限が付与されている必要があります。

- 認証方式を決定します。

必要な権限を付与された管理者として認証されるには、次の方法があります。

- パスワード・ファイルによる方法
- オペレーティング・システム認証による方法

パスワード・ファイルによる認証方法を使用するには、「データベース・パスワード・ファイルの作成とメンテナンス」の説明に従ってパスワード・ファイルを作成します。オペレーティング・システム認証による方法を使用するには、必ずオペレーティング・システムの適切なユーザー・グループのメンバーになっているユーザー・アカウントでホスト・コンピュータにログインします。たとえば、UNIXおよびLinuxプラットフォームでは、通常dbaユーザー・グループを使用します。Windowsプラットフォームでは、Oracleソフトウェアをインストールするユーザーが、必要なユーザー・グループに自動的に配置されます。

関連項目:

- 「データベース管理者のセキュリティと権限について」
- パスワード・ファイルとオペレーティング・システム認証の詳細は、「データベース管理者の認証」を参照してください

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.5 ステップ4: 初期化パラメータ・ファイルの作成

Oracleインスタンスの起動時に、初期化パラメータ・ファイルが読み込まれます。このファイルには、テキスト・エディタで作成して変更できるテキスト・ファイルか、データベースにより作成されて動的に変更されるバイナリ・ファイルを使用できます。バイナリ・ファイル(こちらの使用を推奨)は、サーバー・パラメータ・ファイルと呼ばれます。ここで示すステップでは、テキスト形式の初期化パラメータ・ファイルを作成します。後述のステップで、テキスト・ファイルからサーバー・パラメータ・ファイルを作成します。

- 初期化パラメータ・ファイルを作成します。

テキスト形式の初期化パラメータ・ファイルを作成する方法の1つは、「初期化パラメータ・ファイルのサンプル」に記載されているサンプルを編集する方法です。

初期化パラメータのファイルを手動で作成する場合は、少なくとも表2-2に記載されているパラメータが含まれているようにしてください。記載されていない他のすべてのパラメータにはデフォルトの値があります。

表2-2 推奨される最低限の初期化パラメータ

パラメータ名	必須	ノート
DB_NAME	はい	データベース識別子。CREATE DATABASE 文で使用される値に対応している必要があります。最大 8 文字です。
CONTROL_FILES	いいえ	推奨。提供されていない場合は、データベース・インスタンスにより初期化パラメータ・ファイルと同じ場所に制御ファイルが作成されます。このパラメータを提供することで制御ファイルを多重化できます。詳細は、「初期制御ファイルの作成」を参照してください。
MEMORY_TARGET	いいえ	インスタンスにより使用されるメモリーの合計容量を設定し、自動メモリー管理を有効にします。このパラメータのかわりに他の初期化パラメータを選択して、メモリーの使用をより詳細に手動で制御することもできます。「メモリーの手動構成」を参照してください。

操作を容易にするために、デフォルトのファイル名を使用して、Oracle Databaseのデフォルトの場所に初期化パラメータ・ファイルを配置します。このようにすることにより、Oracle Databaseが初期化パラメータ・ファイルのデフォルトの場所を自動的に参照するため、データベースの起動時にSTARTUPコマンドのPFILE句を指定する必要がなくなります。

使用しているプラットフォームの初期化パラメータ・ファイルのデフォルトのファイル名や場所など、初期化パラメータおよび初期化パラメータ・ファイルの詳細は、「初期化パラメータと初期化パラメータ・ファイルについて」を参照してください。

関連項目:

- 「初期化パラメータの指定」
- すべての初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.6 ステップ5: (Windowsの場合のみ)インスタンスの作成

Windowsプラットフォームでは、インスタンスが存在しない場合には、インスタンスに接続する前に手動で作成する必要があります。ORADIMコマンドでは、新しいWindowsサービスを作成することでOracle Databaseインスタンスを作成します。

インスタンスを作成するには:

- Windowsのコマンド・プロンプトで、次のコマンドを入力します。

```
oradim -NEW -SID sid -STARTMODE MANUAL -PFILE file
```

次のプレースホルダを適切な値に置き換えます。

- `sid` - 目的のSID (mynewdbなど)
- `file` - テキスト形式の初期化パラメータ・ファイルへのフルパス

> 注意:
>
> この時点では、-STARTMODE 引数を AUTO に設定しないでください。設定すると新しいインスタンスが起動され、まだ存在しないデータベースをマウントしようとします。必要に応じて、「ステップ 14: (オプション)インスタンスの自動起動の有効化」で、このパラメータを AUTO に変更できます。

ほとんどのOracle Databaseサービスは、Oracleホーム・ユーザーの権限を使用してシステムにログオンします。サービスは、このユーザーの権限を使用して実行されます。ORADIMコマンドは、このユーザー・アカウントのパスワードを要求します。ORADIMを使用して、他のオプションを指定できます。

関連項目:

- ORADIMコマンドおよびOracleホーム・ユーザーの詳細は、『Oracle Databaseプラットフォーム・ガイドfor Microsoft Windows』を参照してください。
- Oracleホーム・ユーザーの詳細は、『Oracle Databaseインストレーション・ガイドfor Microsoft Windows』を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.7 ステップ6: インスタンスへの接続

SQL*Plusを起動して、SYSDBA管理権限でOracle Databaseインスタンスに接続します。

- パスワード・ファイルによる認証を使用するには、次のコマンドを入力してプロンプトが表示された後、SYSパスワードを入力します。

```
$ sqlplus /nolog
SQL> CONNECT SYS AS SYSDBA
```

- オペレーティング・システム認証を使用するには、次のコマンドを入力します。

```
$ sqlplus /nolog
SQL> CONNECT / AS SYSDBA
```

SQL*Plusが次のメッセージを表示します。

```
Connected to an idle instance.
```

ノート:

SQL*Plus が次のようなメッセージを表示する場合があります。

```
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production Version 19.1.
```

その場合には、インスタンスがすでに起動されています。間違ったインスタンスに接続している可能性があります。EXIT コマンドして SQL*Plus を終了し、ORACLE_SID が正しく設定されていることを確認して、このステップを繰り返してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.8 ステップ7: サーバー・パラメータ・ファイルの作成

サーバー・パラメータ・ファイルを使用すると、ALTER SYSTEMコマンドを使用して初期化パラメータを変更でき、変更内容はデータベースを停止して起動した後も持続します。サーバー・パラメータ・ファイルは、編集済のテキスト形式の初期化ファイルから作成します。

- 次のSQL*Plusコマンドを実行します。

```
CREATE SPFILE FROM PFILE;
```

このSQL*Plusコマンドでは、テキスト形式の初期化パラメータ・ファイル(PFILE)がデフォルトの場所のデフォルトのファイル名で読み込まれ、テキスト形式の初期化パラメータ・ファイルからサーバー・パラメータ・ファイル(SPFILE)が作成されて、SPFILEがデフォルトのSPFILE名でデフォルトの場所に書き込まれます。

デフォルトのファイル名と場所を使用していない場合は、PFILEとSPFILEの両方のファイル名とパスワードを指定することもできます。

ヒント:

サーバー・パラメータ・ファイルを有効にするには、データベースを再起動する必要があります。

ノート:

この時点でのサーバー・パラメータ・ファイルの作成はオプションですが、作成することをお薦めします。サーバー・パラメータ・ファイルを作成しないと、インスタンスが起動されるたびにテキスト形式の初期化パラメータ・ファイルが引き続き読み込まれます。

重要—Oracle Managed Files を使用しており、初期化パラメータ・ファイルに CONTROL_FILES パラメータが指定されていない場合には、CREATE DATABASE 文でデータベースの制御ファイルが作成されるときに制御ファイルの名前と場所を保存できるように、この時点でサーバー・パラメータ・ファイルを作成しておく必要があります。詳細は、「データベース作成時の Oracle Managed Files の作成」を参照してください。

関連項目:

- 「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」
- CREATE SPFILEコマンドの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.9 ステップ8: インスタンスの起動

データベースをマウントせずにインスタンスを起動します。

- NOMOUNT句を指定してSTARTUPコマンドを実行します。

通常、この方法で起動するのはデータベースの作成時またはメンテナンス時のみです。この例では、初期化パラメータ・ファイルまたはサーバー・パラメータ・ファイルがデフォルトの場所に配置されているため、PFILE句の指定は不要です。

```
STARTUP NOMOUNT
```

この時点で、インスタンス・メモリーが割り当てられて、インスタンスのプロセスが起動されます。データベース自体はまだ存在しません。

関連項目:

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

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.10 ステップ9: CREATE DATABASE文の発行

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

- CREATE DATABASE文を実行します。

ノート:

マルチテナント・コンテナ・データベース(CDB)を作成する場合は、『Oracle Multitenant 管理者ガイド』の例を参照してください。

例1

次の文では、データベースmynewdbが作成されます。このデータベース名は、初期化パラメータ・ファイルのDB_NAMEパラメータと同じにする必要があります。この例では、次のことを想定しています。

- 初期化パラメータ・ファイルのCONTROL_FILESパラメータには、制御ファイルの数と場所が指定されています。
- ディレクトリ/u01/app/oracle/oradata/mynewdbが存在します。
- ディレクトリ/u01/logs/myおよび/u02/logs/myが存在します。

```
CREATE DATABASE mynewdb
   USER SYS IDENTIFIED BY sys_password
   USER SYSTEM IDENTIFIED BY system_password
   LOGFILE GROUP 1 ('/u01/logs/my/redo01a.log','/u02/logs/my/redo01b.log') SIZE 100M
BLOCKSIZE 512,
           GROUP 2 ('/u01/logs/my/redo02a.log','/u02/logs/my/redo02b.log') SIZE 100M
BLOCKSIZE 512,
           GROUP 3 ('/u01/logs/my/redo03a.log','/u02/logs/my/redo03b.log') SIZE 100M
BLOCKSIZE 512
   MAXLOGHISTORY 1
   MAXLOGFILES 16
   MAXLOGMEMBERS 3
   MAXDATAFILES 1024
   CHARACTER SET AL32UTF8
   NATIONAL CHARACTER SET AL16UTF16
   EXTENT MANAGEMENT LOCAL
   DATAFILE '/u01/app/oracle/oradata/mynewdb/system01.dbf'
     SIZE 700M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
   SYSAUX DATAFILE '/u01/app/oracle/oradata/mynewdb/sysaux01.dbf'
     SIZE 550M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
   DEFAULT TABLESPACE users
      DATAFILE '/u01/app/oracle/oradata/mynewdb/users01.dbf'
      SIZE 500M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED
   DEFAULT TEMPORARY TABLESPACE tempts1
      TEMPFILE '/u01/app/oracle/oradata/mynewdb/temp01.dbf'
      SIZE 20M REUSE AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED
   UNDO TABLESPACE undotbs1
      DATAFILE '/u01/app/oracle/oradata/mynewdb/undotbs01.dbf'
      SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED
   USER_DATA TABLESPACE usertbs
      DATAFILE '/u01/app/oracle/oradata/mynewdb/usertbs01.dbf'
      SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

次の特性を持つデータベースが作成されます。

- データベースにはmynewdbという名前が付けられます。グローバル・データベース名はmynewdb.us.example.comになり、ドメイン部分(us.example.com)は、初期化パラメータ・ファイルから取得されます。「グローバル・データベース名の決定」を参照してください。
- CONTROL_FILES初期化パラメータ(データベース作成前に初期化パラメータ・ファイルに設定)で指定された3つの制御ファイルが作成されます。「初期化パラメータ・ファイルのサンプル」および「制御ファイルの指定」を参照してください。
- ユーザー・アカウントSYSとSYSTEMのパスワードが指定した値に設定されます。パスワードでは、大/小文字が区別されます。SYSおよびSYSTEMのパスワードを指定する2つの句は、このリリースのOracle Databaseではオプションです。ただし、いずれか一方の句を指定した場合は、もう一方の句も指定する必要があります。これらの句の使用方法の詳細は、「データベースの保護：ユーザーSYSおよびSYSTEMのパスワードの指定」を参照してください。

- 新しいデータベースには、LOGFILE句で指定されているように、REDOログ・ファイルのグループが3つあり、各グループには2つのメンバーが含まれます。MAXLOGFILES、MAXLOGMEMBERSおよびMAXLOGHISTORYは、REDOログの制限を定義します。「適切なREDOログ・ファイル数の選択」を参照してください。REDOログ・ファイルのブロック・サイズは、ディスクの物理セクターと同じサイズである512バイトに設定されています。ブロック・サイズを物理セクター・サイズと同じ(デフォルト)にする場合には、BLOCKSIZE句はオプションです。通常のセクター・サイズおよびブロック・サイズは512です。BLOCKSIZEに設定可能な値は512、1024および4096です。新しいディスクでセクター・サイズが4KBの場合には、オプションでBLOCKSIZEを4096に指定します。詳細は、「REDOログ・ファイルのブロック・サイズの計画」を参照してください。
- MAXDATAFILESによって、このデータベースでオープンできるデータファイルの最大数が指定されます。この数は、制御ファイルの初期サイズに影響を及ぼします。

ノート:

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

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

- このデータベースにデータを格納する際は、AL32UTF8文字セットが使用されます。
- NATIONAL CHARACTER SETとしてAL16UTF16文字セットが指定され、特にNCHAR、NCLOBまたはNVARCHAR2として定義された列にデータを格納する際に使用されます。
- DATAFILE句で指定したとおりに、SYSTEM,表領域(オペレーティング・システム・ファイル/u01/app/oracle/oradata/mynewdb/system01.dbfからなる)が作成されます。指定した名前のファイルがすでに存在する場合は上書きされます。
- SYSTEM表領域はローカル管理の表領域として作成されます。「ローカル管理のSYSTEM表領域の作成」を参照してください。
- SYSAUX DATAFILE句で指定したとおりに、SYSAUX表領域(オペレーティング・システム・ファイル/u01/oracle/oradata/mynewdb/sysaux01.dbfからなる)が作成されます。「SYSAUX表領域について」を参照してください。
- DEFAULT TABLESPACE句により、指定した名前で、このデータベース用のデフォルト永続表領域が作成されます。
- DEFAULT TEMPORARY TABLESPACE句により、指定した名前で、このデータベース用のデフォルト一時表領域が作成されます。「デフォルト一時表領域の作成」を参照してください。
- UNDO TABLESPACE句によって、UNDO表領域が指定の名前で作成されます。このUNDO表領域には、初期化パラメータ・ファイルでUNDO_MANAGEMENT=AUTOを指定した場合に、このデータベース用のUNDOデータが格納されます。このパラメータを省略すると、デフォルトでAUTOに設定されます。「自動UNDO管理の使用: UNDO表領域の作成」を参照してください。

- USER_DATA TABLESPACE句により、指定した名前で、ユーザー・データおよびデータベース・オプション(Oracle XML DBなど)を格納するための表領域が作成されます。
- このCREATE DATABASE文ではARCHIVELOG句が指定されていないため、初期状態ではオンラインREDOログはアーカイブされません。これはデータベース作成時の慣例です。後でALTER DATABASE文を使用して、ARCHIVELOGモードに切り替えることができます。mynewdbのアーカイブに関連する初期化パラメータ・ファイル内の初期化パラメータは、LOG_ARCHIVE_DEST_1およびLOG_ARCHIVE_FORMATです。「アーカイブREDOログ・ファイルの管理」を参照してください。

ヒント:

- CREATE DATABASE 文で使用しているディレクトリがすべて存在することを確認します。CREATE DATABASE 文ではディレクトリは作成されません。
- Oracle Managed Files を使用していない場合は、すべての TABLESPACE 句に DATAFILE 句または TEMPFILE 句を指定する必要があります。
- データベースの作成に失敗した場合は、アラート・ログを参照すると、失敗の原因とその対処措置を判別できます。「アラート・ログの表示」を参照してください。プロセス番号を含むエラー・メッセージが表示される場合は、そのプロセスのトレース・ファイルを調べます。そのプロセス番号がトレース・ファイル名に含まれているトレース・ファイルを探します。詳細は、「トレース・ファイルの検索」を参照してください。
- CREATE DATABASE 文を失敗後に再発行するには、最初にインスタンスを停止してから、前の CREATE DATABASE 文で作成されたファイルを削除する必要があります。

例2

この例では、より簡単なCREATE DATABASE文を使用できる、Oracle Managed Filesを使用したデータベースの作成例を示します。Oracle Managed Filesを使用するには、初期化パラメータDB_CREATE_FILE_DESTを設定する必要があります。このパラメータは、データベースで作成されて自動的に名前が付けられる様々なデータベース・ファイル用のベース・ディレクトリを定義します。次の文は、初期化パラメータ・ファイルにこのパラメータを設定する例を示しています。

```
DB_CREATE_FILE_DEST='/u01/app/oracle/oradata'
```

Oracle Managed Filesおよび次のCREATE DATABASE文を使用すると、SYSTEM表領域とSYSAUX表領域および文に指定されている追加の表領域がデータベースによって作成され、すべてのデータファイル、制御ファイルおよびREDOログ・ファイルのデフォルト・サイズとプロパティが選択されます。この方法で設定されるこれらのプロパティおよび他のデフォルト・データベース・プロパティは本番環境には適さない場合があるため、生成された設定を調べて必要に応じて変更することをお薦めします。

```
CREATE DATABASE mynewdb
USER SYS IDENTIFIED BY sys_password
USER SYSTEM IDENTIFIED BY system_password
EXTENT MANAGEMENT LOCAL
DEFAULT TEMPORARY TABLESPACE temp
UNDO TABLESPACE undotbs1
DEFAULT TABLESPACE users;
```

ヒント:

CREATE DATABASE 文が失敗する場合でステップ 7 を完了していない場合は、予期しない方法で初期化パラメータが設定されている、このインスタンス用の既存のサーバー・パラメータ・ファイル(SPFILE)がないかを確認します。たとえば、すべての制御ファイルへの完全パスが SPFILE に設定されている場合は、それらの制御ファイルが存在しないと、CREATE DATABASE 文が失敗します。不必要な SPFILE を削除した後は、必ずインスタンスを停止して(STARTUP NOMOUNT を指定して)再起動します。詳細は、「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」を参照してください。

関連項目:

- 「CREATE DATABASE文の句の指定」
- 「データベース作成時のOracle Managed Filesの作成」
- Oracle Managed Filesの使用
- CREATE DATABASE文で指定できる句およびパラメータ値の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.11 ステップ10: 追加の表領域の作成

データベースを稼働させるには、アプリケーション・データ用に追加の表領域を作成する必要があります。

- CREATE TABLESPACE文を実行して、追加の表領域を作成します。

次のサンプル・スクリプトは、追加の表領域を作成します。

```
CREATE TABLESPACE apps_tbs LOGGING
     DATAFILE '/u01/app/oracle/oradata/mynewdb/apps01.dbf'
     SIZE 500M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
     EXTENT MANAGEMENT LOCAL;
-- create a tablespace for indexes, separate from user tablespace (optional)
CREATE TABLESPACE indx_tbs LOGGING
     DATAFILE '/u01/app/oracle/oradata/mynewdb/indx01.dbf'
     SIZE 100M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
     EXTENT MANAGEMENT LOCAL;
```

表領域の作成方法の詳細は、「表領域の管理」を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.12 ステップ11: スクリプトの実行によるデータ・ディクショナリ・ビューの作成

データ・ディクショナリ・ビュー、シノニムおよびPL/SQLパッケージの作成およびSQL*Plusの適切な稼働に必要なスクリプトを実行します。

1. 次のいずれかのステップを実行します。
   - SQL*Plusで、SYSDBA権限を持つユーザーとして次のスクリプトを実行します。

     ```
     @?/rdbms/admin/catalog.sql
     @?/rdbms/admin/catproc.sql
     ```

     または

- スクリプトcatpcat.sqlを指定してcatctl.plを実行します。

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catctl.pl -d
$ORACLE_HOME/rdbms/admin -n number_of_processes -l output_log_directory
catpcat.sql
```

number_of_processesの値には、システムで使用可能なプロセッサの数が反映されている必要があります。CDBと非CDBのこの最大値は8にできます。この値を指定しない場合、非CDBでのデフォルト値は4であり、CDBでのデフォルト値はCPU_COUNT初期化パラメータとなります。

ノート:

- マルチテナント・コンテナ・データベース(CDB)を作成する場合は、パラメータ-c 'CDB$ROOT PDB$SEED' (Windows システムでは二重引用符を使用)を追加で指定して、スクリプト catpcat.sql を最初にルート・コンテナ(CDB$ROOT)で実行し、次にシード PDB (PDB$SEED)で実行します。
- スクリプト catpcat.sql は、パラレル・プロセスでスクリプト catalog.sql および catproc.sql を実行し、それによってデータ・ディクショナリ作成のパフォーマンスを向上させます。

2. SQL*Plusで、SYSDBA権限を持つユーザーとして次のスクリプトを実行します。

```
@?/rdbms/admin/utlrp.sql
```

3. SQL*Plusで、SYSTEMユーザーとして次のスクリプトを実行します。

```
@?/sqlplus/admin/pupbld.sql
```

アットマーク(@)はSQL*Plusスクリプトを実行するコマンドの省略記述です。疑問符(?)はOracleホーム・ディレクトリを表すSQL*Plusの変数です。

次の表に、これらのスクリプトの説明を示します。

スクリプト	説明
catalog.sql	データ・ディクショナリ表のビュー、動的パフォーマンス・ビューおよび多くのビューに対するパブリック・シノニムを作成します。シノニムに PUBLIC アクセスを付与します。
catproc.sql	PL/SQL に必要なスクリプトや PL/SQL とともに使用されるスクリプトをすべて実行します。
utlrp.sql	パッケージ、プロシージャおよび型も含めて、無効な状態となっているすべての PL/SQL モジュールを再コンパイルします。
pupbld.sql	SQL*Plus に必要です。SQL*Plus がユーザーのコマンドを使用禁止にできるようにします。
catpcat.sql	データ・ディクショナリを作成します。このスクリプトは、catctl.pl プログラムを使用して動作し(SQL*Plus は使用しない)、パラレル・プロセスでスクリプト catalog.sql および

スクリプト	説明
	`catproc.sql` を内部的に実行し、それによってデータ・ディクショナリ作成のパフォーマンスを向上させます。

関連項目:

`catctl.pl`に対するすべてのパラメータのリストは、Oracle Databaseアップグレード・ガイドを参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.13 ステップ12: (オプション)スクリプトの実行による追加オプションのインストール

必要に応じて、他のスクリプトも実行できます。実行するスクリプトは、使用またはインストール対象として選択する機能とオプションによって異なります。

- スクリプトを実行して追加オプションをインストールします。

使用可能なスクリプトは、『Oracle Databaseリファレンス』を参照してください。

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

インストールを計画している製品のインストールと管理の方法は、製品固有のOracleマニュアルを参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.14 ステップ13: データベースのバックアップ

メディア障害が発生した場合にリカバリするための完全なファイル・セットが確実に存在するように、データベースの全体バックアップを作成してください。

- データベースをバックアップします。

データベースのバックアップの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.4.15 ステップ14: (オプション)インスタンスの自動起動の有効化

ホスト・コンピュータの再起動時に、Oracleインスタンスが自動的に起動されるように構成する必要がある場合があります。

- ホスト・コンピュータの再起動時に、Oracleインスタンスが自動的に起動されるように構成します。

詳細は、オペレーティング・システムのドキュメントを参照してください。たとえばWindowsでは、次のコマンドを使用して、コンピュータの再起動時にデータベース・サービスがインスタンスを起動するように構成します。

```
ORADIM -EDIT -SID sid -STARTMODE AUTO -SRVCSTART SYSTEM [-SPFILE]
```

自動再起動時にインスタンスがSPFILEを読み込むようにする場合は、`-SPFILE`引数を使用する必要があります。

関連項目:

- Oracle Databaseの自動再起動の構成
- ORADIMコマンドの詳細は、『Oracle Databaseプラットフォーム・ガイドfor Microsoft Windows』を参照してください。

親トピック: CREATE DATABASE文を使用したデータベースの作成

2.5 CREATE DATABASE文の句の指定

CREATE DATABASE文を実行すると、Oracle Databaseはいくつかの操作を実行します。実際の操作は、CREATE DATABASE文で指定した句、および初期化パラメータの設定に応じて実行されます。

- CREATE DATABASE文の句について
  CREATE DATABASE句を使用して、データベースの作成および管理を簡略化できます。
- データベースの保護: ユーザーSYSおよびSYSTEMのパスワードの指定
  データベースを保護するには、ユーザーSYSおよびSYSTEMのパスワードを指定します。
- ローカル管理のSYSTEM表領域の作成
  データベースの作成時に、ローカル管理のSYSTEM表領域を作成します。ローカル管理表領域では、各データファイルに格納されるビットマップを使用してエクステントを管理します。
- SYSAUX表領域のデータファイル属性の指定
  SYSAUX表領域はデフォルトで作成されますが、データベース作成時にデータファイル属性を指定できます。
- 自動UNDO管理の使用: UNDO表領域の作成
  自動UNDO管理ではUNDO表領域が使用されます。
- デフォルト永続表領域の作成
  デフォルトの永続表領域を作成することをお薦めします。この表領域には、Oracle Databaseによって、別の永続表領域が明示的に指定されていないSYSTEM以外のユーザーが割り当てられます。
- デフォルト一時表領域の作成
  デフォルト一時表領域を作成する場合、Oracle Databaseでは、一時表領域が明示的に割り当てられていないユーザーに対してこの表領域が一時表領域として割り当てられます。
- データベース作成時のOracle Managed Filesの指定
  Oracle Managed Files機能を使用すると、CREATE DATABASE文に指定する句とパラメータの数を最小限に抑えることができます。
- データベース作成時のbigfile表領域のサポート
  Oracle Databaseでは、bigfile表領域を作成できるようにすることで、表領域の管理を簡素化し、非常に大規模なデータベースのサポートを可能にしています。
- データベースのタイムゾーンとタイムゾーン・ファイルの指定
  Oracle Databaseの日時データ型、期間データ型およびタイムゾーン・サポートにより、イベントとトランザクションの時間に関して一貫性のある情報を格納できます。
- FORCE LOGGINGモードの指定
  一部のデータ定義言語文(CREATE TABLEなど)では、NOLOGGING句を使用でき、するとある種のデータベース操作からデータベースREDOログのREDOレコードが生成されなくなります。NOLOGGING句を設定すると、データベース・リカバリ・メカニズム外で容易にリカバリできる操作は高速化されますが、メディア・リカバリとスタンバイ・データベースに悪影響を与える可能性があります。

親トピック: Oracle Databaseの作成および構成

2.5.1 CREATE DATABASE文の句について

CREATE DATABASE句を使用して、データベースの作成および管理を簡略化できます。

CREATE DATABASE文を実行すると、Oracle Databaseは少なくとも次の操作を実行します。

- データベース用のデータファイルの作成
- データベース用の制御ファイルの作成
- データベース用のオンラインREDOログの作成とARCHIVELOGモードの設定
- SYSTEM表領域の作成
- SYSAUX表領域の作成
- データ・ディクショナリの作成
- データベースへのデータの格納に使用する文字セットの設定
- データベース・タイム・ゾーンの設定
- データベースのマウントとオープン

親トピック: CREATE DATABASE文の句の指定

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

データベースを保護するには、ユーザーSYSおよびSYSTEMのパスワードを指定します。

- CREATE DATABASE文に、ユーザーSYSおよびSYSTEMのパスワードを指定する句を含めます。

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

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

パスワードを選択するときは、パスワードでは大/小文字が区別されることに注意してください。また、使用中のデータベースにパスワード形式の要件が設定されている場合もあります。

関連項目:

セキュアなパスワードの作成についてのオラクル社のガイドラインについては、Oracle Databaseセキュリティ・ガイドを参照してください。

親トピック: CREATE DATABASE文の句の指定

2.5.3 ローカル管理のSYSTEM表領域の作成

データベースの作成時に、ローカル管理のSYSTEM表領域を作成します。ローカル管理表領域では、各データファイルに格納されるビットマップを使用してエクステントを管理します。

- CREATE DATABASE文にEXTENT MANAGEMENT LOCAL句を指定すると、ローカル管理のSYSTEM表領域を作成できます。

EXTENT MANAGEMENT LOCAL句を指定しない場合は、デフォルトでディクショナリ管理のSYSTEM表領域が作成されます。ディクショナリ管理表領域は非推奨です。

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

- DEFAULT TEMPORARY TABLESPACE句をCREATE DATABASE文に指定
- UNDO TABLESPACE句をCREATE DATABASE文に指定

関連項目:

- SYSTEM表領域に対してEXTENT MANAGEMENT LOCALを指定する場合のDEFAULT TEMPORARY TABLESPACE句およびUNDO TABLESPACE句の使用に関する詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 「ローカル管理表領域」
- 「ローカル管理表領域へのSYSTEM表領域の移行」

親トピック: CREATE DATABASE文の句の指定

2.5.4 SYSAUX表領域のデータファイル属性の指定

SYSAUX表領域はデフォルトで作成されますが、データベース作成時にデータファイル属性を指定できます。

SYSAUX表領域のデータファイル属性を指定するには:

- CREATE DATABASE文にSYSAUX DATAFILE句を含めます。

SYSTEM表領域に対してDATAFILE句を指定している場合は、SYSAUX DATAFILE句も指定する必要があり、指定しない場合は、CREATE DATABASE文が失敗します。この要件は、Oracle Managed Files機能が使用可能な場合は適用されません(「データベース作成時のOracle Managed Filesの作成」を参照してください)。

- SYSAUX表領域について
  SYSAUX表領域はデータベース作成時に必ず作成されます。SYSAUX表領域は、SYSTEM表領域の予備の表領域として機能します。これは、以前に固有の表領域を必要としていたOracle Databaseの多くの機能と製品に対するデフォルトの表領域であるため、データベースに必要な表領域の数が削減されます。また、SYSTEM表領域の負荷も軽減されます。

親トピック: CREATE DATABASE文の句の指定

2.5.4.1 SYSAUX表領域について

SYSAUX表領域はデータベース作成時に必ず作成されます。SYSAUX表領域は、SYSTEM表領域の予備の表領域として機能します。これは、以前に固有の表領域を必要としていたOracle Databaseの多くの機能と製品に対するデフォルトの表領域であるため、データベースに必要な表領域の数が削減されます。また、SYSTEM表領域の負荷も軽減されます。

SYSAUX表領域に対して指定できるのはデータファイルの属性のみで、CREATE DATABASE文でSYSAUX DATAFILE句を使用して指定します。SYSAUX表領域の必須属性は、Oracle Databaseによって設定され、次のような属性があります。

- PERMANENT
- READ WRITE
- EXTENT MANAGMENT LOCAL
- SEGMENT SPACE MANAGMENT AUTO

これらの属性は、ALTER TABLESPACE文で変更できず、変更しようとするとエラーが発生します。SYSAUX表領域の削除や

名前変更はできません。

SYSAUX表領域のサイズは、SYSAUXを使用するデータベース・コンポーネントのサイズにより決定します。V$SYSAUX_OCCUPANTSビューを問い合せることにより、これらのコンポーネントのリストを表示できます。これらのコンポーネントの初期サイズに基づいて、SYSAUX表領域には、データベース作成時に最低限400MB必要です。SYSAUX表領域の領域要件は、データベースが完全にデプロイされた後、その使用状況やワークロードによって増加します。SYSAUX表領域の領域消費を継続的に管理する方法の詳細は、「SYSAUX表領域の管理」を参照してください。

SYSAUX表領域のセキュリティ属性はSYSTEM表領域と同じです。

関連項目:

「SYSAUX表領域の管理」

親トピック: SYSAUX表領域のデータファイル属性の指定

2.5.5 自動UNDO管理の使用: UNDO表領域の作成

自動UNDO管理ではUNDO表領域が使用されます。

- 自動UNDO管理を使用可能にするには、初期化パラメータ・ファイルでUNDO_MANAGEMENT初期化パラメータをAUTOに設定します。このパラメータを省略すると、デフォルトでデータベースが自動UNDO管理になります。

このモードでは、UNDOデータがUNDO表領域に格納され、Oracle Databaseによって管理されます。UNDO表領域を定義して名前を付けるには、データベース作成時にCREATE DATABASE文にUNDO TABLESPACE句を指定する必要があります。この句を省略して自動UNDO管理を使用可能にすると、SYS_UNDOTBSという名前のデフォルトのUNDO表領域が作成されます。

ノート:

UNDO 表領域を自分で定義する場合は、ブロック・サイズがデータベースのデータファイルの最大ブロック・サイズと一致するようにします。

関連項目:

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

親トピック: CREATE DATABASE文の句の指定

2.5.6 デフォルト永続表領域の作成

デフォルトの永続表領域を作成することをお薦めします。この表領域には、Oracle Databaseによって、別の永続表領域が明示的に指定されていないSYSTEM以外のユーザーが割り当てられます。

データベースのデフォルトの永続表領域を指定するには:

- DEFAULT TABLESPACE句をCREATE DATABASE文に含めます

DEFAULT TABLESPACE句を指定しない場合、SYSTEM以外のユーザーに対するデフォルトの永続表領域はSYSTEM表領

域です。

関連項目:

CREATE DATABASEおよびALTER DATABASEのDEFAULT TABLESPACE句の構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: CREATE DATABASE文の句の指定

2.5.7 デフォルト一時表領域の作成

デフォルト一時表領域を作成する場合、Oracle Databaseでは、一時表領域が明示的に割り当てられていないユーザーに対してこの表領域が一時表領域として割り当てられます。

データベースのデフォルト一時表領域を作成するには:

- DEFAULT TEMPORARY TABLESPACE句をCREATE DATABASE文に含めます。

一時表領域または表領域グループは、CREATE USER文で明示的にユーザーに割り当てることができます。ただし、それを行わない場合は、デフォルトの一時表領域がデータベースに指定されていないと、これらのユーザーには一時表領域としてSYSTEM表領域がデフォルトで割り当てられます。一時データをSYSTEM表領域に格納するのはよい方法ではなく、すべてのユーザーに一時表領域を個別に割り当てるのは非効率的です。したがって、CREATE DATABASEのDEFAULT TEMPORARY TABLESPACE句の使用をお薦めします。

> ノート:
>
> ローカル管理の SYSTEM 表領域を指定すると、その SYSTEM 表領域は一時表領域として使用できません。この場合は、デフォルトの一時表領域を作成する必要があります。この動作については、「ローカル管理の SYSTEM 表領域の作成」を参照してください。

関連項目:

- CREATE DATABASEおよびALTER DATABASEのDEFAULT TEMPORARY TABLESPACE句の構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 一時表領域の作成と使用については、「一時表領域」を参照してください
- 一時表領域グループの作成と使用については、「複数の一時表領域: 表領域グループの使用」を参照してください

親トピック: CREATE DATABASE文の句の指定

2.5.8 データベース作成時のOracle Managed Filesの作成

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

- Oracle Databaseによってファイルが作成および管理されるディレクトリまたはOracle Automatic Storage Management(Oracle ASM)ディスク・グループのいずれかを指定します。

初期化パラメータ・ファイルにDB_CREATE_FILE_DEST、DB_CREATE_ONLINE_LOG_DEST_nまたは

DB_RECOVERY_FILE_DEST初期化パラメータを設定すると、データベースの基礎になるオペレーティング・システム・ファイルをOracle Databaseで作成および管理できます。指定した初期化パラメータおよびCREATE DATABASE文で指定した句に応じて、次のデータベース構造のオペレーティング・システム・ファイルがOracle Databaseによって自動的に作成および管理されます。

- 表領域とそのデータファイル
- 一時表領域とその一時ファイル
- 制御ファイル
- オンラインREDOログ
- アーカイブREDOログ・ファイル
- フラッシュバック・ログ
- ブロック・チェンジ・トラッキング・ファイル
- RMANバックアップ

関連項目:

高速リカバリ領域を作成する初期化パラメータの設定方法については、「高速リカバリ領域の指定」を参照してください

次のCREATE DATABASE文で、Oracle Managed Files機能の動作を示します。必要な初期化パラメータは指定されているとします。

```
CREATE DATABASE mynewdb
     USER SYS IDENTIFIED BY sys_password
     USER SYSTEM IDENTIFIED BY system_password
     EXTENT MANAGEMENT LOCAL
     UNDO TABLESPACE undotbs1
     DEFAULT TEMPORARY TABLESPACE tempts1
     DEFAULT TABLESPACE users;
```

- SYSTEM表領域はローカル管理の表領域として作成されます。EXTENT MANAGEMENT LOCAL句を指定しないと、SYSTEM表領域はディクショナリ管理の対象として作成されますが、この方法はお薦めしません。
- DATAFILE句が指定されていないため、SYSTEM表領域用のOracle Managed Filesのデータファイルが作成されます。
- LOGFILE句が指定されていないため、Oracleが管理するREDOログ・ファイルのグループが2つ作成されます。
- SYSAUX DATAFILEが指定されていないため、SYSAUX表領域用のOracle Managed Filesのデータファイルが作成されます。
- UNDO TABLESPACE句およびDEFAULT TABLESPACE句にDATAFILE副次句が指定されていないため、これらの各表領域用にOracle Managed Filesのデータファイルが作成されます。
- DEFAULT TEMPORARY TABLESPACE句にTEMPFILE副次句が指定されていないため、Oracle Managed Filesの一時ファイルが作成されます。
- 初期化パラメータ・ファイルにCONTROL_FILES初期化パラメータが指定されていない場合は、Oracle Managed Filesの制御ファイルも作成されます。
- サーバー・パラメータ・ファイル(「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」を参照)を使用してい

る場合は、適切な初期化パラメータが自動的に設定されます。

関連項目:

- Oracle Managed Files機能と使用方法の詳細は、「Oracle Managed Filesの使用」を参照してください
- 自動ストレージ管理の詳細は、Oracle Automatic Storage Management管理者ガイドを参照してください。

親トピック: CREATE DATABASE文の句の指定

2.5.9 データベース作成時のbigfile表領域のサポート

Oracle Databaseでは、bigfile表領域を作成できるようにすることで、表領域の管理を簡素化し、非常に大規模なデータベースのサポートを可能にしています。

bigfile表領域に含めることができるファイルは1つのみですが、そのファイルには最大40億ブロックまで設定できます。1つのOracle Databaseのデータファイルの最大数は制限されています(通常は64000ファイル)。したがって、bigfile表領域によって、Oracle Databaseの記憶域容量が大幅に増加します。

この項では、bigfile表領域のサポートを可能にするCREATE DATABASE文の句について説明します。

- デフォルトの表領域タイプの指定
  CREATE DATABASE文のSET DEFAULT...TABLESPACE句によって、後続のCREATE TABLESPACE文でこのデータベースに使用される表領域のデフォルト・タイプが決定します。
- デフォルトの表領域タイプの上書き
  SYSTEMおよびSYSAUX表領域は、常にデフォルトの表領域タイプで作成されます。ただし、UNDOおよびDEFAULT TEMPORARY表領域のデフォルトの表領域タイプは、CREATE DATABASE操作時に、オプションで明示的に上書きできます。

関連項目:

bigfile表領域の詳細は、「bigfile表領域」を参照してください

親トピック: CREATE DATABASE文の句の指定

2.5.9.1 デフォルトの表領域タイプの指定

CREATE DATABASE文のSET DEFAULT...TABLESPACE句によって、後続のCREATE TABLESPACE文でこのデータベースに使用される表領域のデフォルト・タイプが決定します。

- SET DEFAULT BIGFILE TABLESPACEまたはSET DEFAULT SMALLFILE TABLESPACEを指定します。

この句を省略すると、デフォルトでは、従来のタイプのOracle Database表領域であるsmallfile表領域が作成されます。smallfile表領域には最大1022ファイルを含めることができ、それぞれ400万ブロックまで設定できます。

bigfile表領域によってデータファイルがユーザーに対して完全に透過的になるため、bigfile表領域を使用するとOracle Managed Filesの機能がさらに強化されます。ALTER TABLESPACE文のSQL構文は、基礎になるデータファイルではなく表領域で操作を実行できるように拡張されています。

表領域のデフォルト・タイプがbigfile表領域であることを指定するには、「データベース作成時のOracle Managed Filesの作成」に示したCREATE DATABASE文を次のように変更します。

```
CREATE DATABASE mynewdb
     USER SYS IDENTIFIED BY sys_password
     USER SYSTEM IDENTIFIED BY system_password
     SET DEFAULT BIGFILE TABLESPACE
     UNDO TABLESPACE undotbs1
     DEFAULT TEMPORARY TABLESPACE tempts1;
```

デフォルトの表領域タイプをデータベース作成後に動的に変更するには、ALTER DATABASE文のSET DEFAULT TABLESPACE句を使用します。

```
ALTER DATABASE SET DEFAULT BIGFILE TABLESPACE;
```

データベースの現行のデフォルト一時表領域タイプを判断するには、DATABASE_PROPERTIESデータ・ディクショナリ・ビューを次のように問い合せます。

```
SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES
   WHERE PROPERTY_NAME = 'DEFAULT_TBS_TYPE';
```

親トピック: データベース作成時のbigfile表領域のサポート

2.5.9.2 デフォルトの表領域タイプの上書き

SYSTEMおよびSYSAUX表領域は、常にデフォルトの表領域タイプで作成されます。ただし、UNDOおよびDEFAULT TEMPORARY表領域のデフォルトの表領域タイプは、CREATE DATABASE操作時に、オプションで明示的に上書きできます。

- デフォルトの表領域タイプをオーバーライドするUNDO TABLESPACE句またはDEFAULT TEMPORARY TABLESPACE句を指定します。

たとえば、次のように指定すると、デフォルトの表領域タイプがsmallfileのデータベースで、bigfileのUNDO表領域を作成できます。

```
CREATE DATABASE mynewdb
...
     BIGFILE UNDO TABLESPACE undotbs1
        DATAFILE '/u01/oracle/oradata/mynewdb/undotbs01.dbf'
        SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

次のように指定すると、デフォルトの表領域タイプがbigfileのデータベースで、smallfileのDEFAULT TEMPORARY表領域を作成できます。

```
CREATE DATABASE mynewdb
   SET DEFAULT BIGFILE TABLESPACE
...
     SMALLFILE DEFAULT TEMPORARY TABLESPACE tempts1
        TEMPFILE '/u01/oracle/oradata/mynewdb/temp01.dbf'
        SIZE 20M REUSE
...
```

親トピック: データベース作成時のbigfile表領域のサポート

2.5.10 データベースのタイム・ゾーンとタイム・ゾーン・ファイルの指定

Oracle Databaseの日時データ型、期間データ型およびタイム・ゾーン・サポートにより、イベントとトランザクションの時間に関して一貫性のある情報を格納できます。

- データベースのタイムゾーンの設定
  CREATE DATABASE文のSET TIME_ZONE句を使用して、データベース・タイムゾーンを設定できます。
- データベースのタイムゾーン・ファイルについて
  Oracleホーム・ディレクトリのサブディレクトリには、2つのタイムゾーン・ファイルがあります。タイム・ゾーン・ファイルには有効なタイム・ゾーン名が含まれています。
- データベースのタイムゾーン・ファイルの指定
  情報を共有しているすべてのデータベースが同じタイムゾーン・データファイルを使用する必要があります。

親トピック: CREATE DATABASE文の句の指定

2.5.10.1 データベースのタイム・ゾーンの設定

CREATE DATABASE文のSET TIME_ZONE句を使用して、データベース・タイム・ゾーンを設定できます。

- データベースの作成時にデータベースのタイム・ゾーンを設定するには、CREATE DATABASE文でSET TIME_ZONE句を使用します。

データベースのタイム・ゾーンを設定しない場合は、デフォルトでホスト・オペレーティング・システムのタイム・ゾーンに設定されます。

セッションのデータベース・タイム・ゾーンを変更するには、ALTER SESSION文でSET TIME_ZONE句を使用します。

関連項目:

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

親トピック: データベースのタイム・ゾーンとタイム・ゾーン・ファイルの指定

2.5.10.2 データベースのタイム・ゾーン・ファイルについて

Oracleホーム・ディレクトリのサブディレクトリには、次の2つのタイム・ゾーン・ファイルがあります。タイム・ゾーン・ファイルには有効なタイム・ゾーン名が含まれています。

タイム・ゾーンごとに次の情報も含まれています。

- 協定世界時(UTC)からのオフセット。
- 夏時間への移行時間。
- 標準時間と夏時間の略称。

デフォルトのタイム・ゾーン・ファイルはORACLE_HOME/oracore/zoneinfo/timezlrg_11.datです。タイム・ゾーン数の少ないタイム・ゾーン・ファイルはORACLE_HOME/oracore/zoneinfo/timezone_11.datにあります。

ファイル内で、データベースで使用されているタイム・ゾーン名を表示するには、次の問合せを実行します。

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

関連項目:

タイム・ゾーン・ファイルの管理および選択の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

親トピック: データベースのタイム・ゾーンとタイム・ゾーン・ファイルの指定

2.5.10.3 データベースのタイム・ゾーン・ファイルの指定

情報を共有しているすべてのデータベースが同じタイム・ゾーン・データファイルを使用する必要があります。

データベース・サーバーでは、定義数が多いタイム・ゾーン・ファイルが常にデフォルトで使用されます。

クライアントで小さなタイム・ゾーン・ファイルを使用するには、すべてのデータが小さなファイル内の地域のみ参照することを認識します。

- クライアントのORA_TZFILE環境変数をクライアントのタイムゾーンversion.datファイルのフルパス名に設定します。versionは、データベース・サーバーで使用されるタイム・ゾーン・ファイル・バージョンと一致します。

デフォルトの定義数の多いタイム・ゾーン・ファイルをクライアントですでに使用している場合、定義数の少ないタイム・ゾーン・ファイルに変更するのは実用的ではありません。データベースに、定義数の少ないファイルには含まれていないタイム・ゾーンのデータが含まれている可能性があるためです。

親トピック: データベースのタイム・ゾーンとタイム・ゾーン・ファイルの指定

2.5.11 FORCE LOGGINGモードの指定

一部のデータ定義言語文(CREATE TABLEなど)では、NOLOGGING句を使用できますが、ある種のデータベース操作ではデータベースREDOログにREDOレコードが生成されません。NOLOGGING句を設定すると、データベース・リカバリ・メカニズム外で容易にリカバリできる操作は高速化されますが、メディア・リカバリとスタンバイ・データベースに悪影響を与える可能性があります。

Oracle Databaseでは、DDL文にNOLOGGINGが指定されていても、REDOレコードを強制的に書き込ませることができます。データベースでは、一時表領域と一時セグメントのREDOレコードは生成されないため、FORCE LOGGINGはオブジェクトに影響を与えません。

- FORCE LOGGING句の使用
  NOLOGGINGがDDL文で指定されている場合でも、REDOレコードを強制的に書き込むことができます。
- FORCE LOGGINGモードのパフォーマンスに関する考慮点
  FORCE LOGGINGモードは、ある程度のパフォーマンスの低下を伴います。

関連項目:

NOLOGGINGモードで実行できる操作の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: CREATE DATABASE文の句の指定

2.5.11.1 FORCE LOGGING句の使用

NOLOGGINGがDDL文で指定されている場合でも、REDOレコードを強制的に書き込むことができます。

データベースをFORCE LOGGINGモードにするには:

- CREATE DATABASE文にFORCE LOGGING句を含めます。

この句を指定しない場合、データベースはFORCE LOGGINGモードになりません。

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

次のSQL文を使用すると、FORCE LOGGINGモードを取り消すことができます。

```
ALTER DATABASE NO FORCE LOGGING;
```

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

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

関連項目:

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

親トピック: FORCE LOGGINGモードの指定

2.5.11.2 FORCE LOGGINGモードのパフォーマンスに関する考慮点

FORCE LOGGINGモードは、ある程度のパフォーマンスの低下を伴います。

主として完全メディア・リカバリを確実に行うためにFORCE LOGGINGを指定し、アクティブになっているスタンバイ・データベースがない場合は、次の点を考慮する必要があります。

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

データベースがNOARCHIVELOGモードで稼働している場合、通常はFORCE LOGGINGモードにする利点はありません。NOARCHIVELOGモードではメディア・リカバリが不可能であるため、FORCE LOGGINGと組み合せて使用する場合は、ほとんど利点がなく、パフォーマンスが低下する可能性があります。

Oracle Databaseリリース18c以降には、次の2つの新しいNOLOGGING句が導入されています。この句を使用すると、ログに記録されない操作の実行と、Active Data Guardスタンバイ・データベースによるすべてのデータの受信が可能になるため、FORCE LOGGINGモードで生成される巨大なREDOログによるパフォーマンスの低下を防止できます。

- STANDBY NOLOGGING FOR DATA AVAILABILITY
- STANDBY NOLOGGING FOR LOAD PERFORMANCE

関連項目:

STANDBY NOLOGGING句の詳細は、『Oracle Data Guard概要および管理』を参照してください

親トピック: FORCE LOGGINGモードの指定

2.6 初期化パラメータの指定

新しいデータベースを作成する前に、基本初期化パラメータを追加または編集できます。

- 初期化パラメータと初期化パラメータ・ファイルについて

Oracleインスタンスの起動時に、初期化パラメータ・ファイルから初期化パラメータが読み込まれます。このファイルでは、少なくともDB_NAMEパラメータを指定する必要があります。他のすべてのパラメータにはデフォルトの値があります。

- グローバル・データベース名の決定
  グローバル・データベース名は、ユーザー指定のローカル・データベース名と、ネットワーク構造内でのデータベースの位置で構成されます。
- 高速リカバリ領域の指定
  高速リカバリ領域とは、Oracle Databaseでバックアップおよびリカバリに関連するファイルを格納および管理できる場所のことです。これは、現在のデータベース・ファイル(データファイル、制御ファイルおよびオンラインREDOログ)の場所であるデータベース領域とは別です。
- 制御ファイルの指定
  すべてのデータベースには、データベースの構造(名前、作成時のタイムスタンプ、データ・ファイルおよびREDOファイルの名前や場所など)を記述するエントリを含む制御ファイルがあります。CONTROL_FILES初期化パラメータには、1つ以上の制御ファイル名をカンマで区切って指定します。
- データベース・ブロック・サイズの指定
  DB_BLOCK_SIZE初期化パラメータは、データベースの標準ブロック・サイズを指定します。
- 最大プロセス数の指定
  Oracle Databaseに同時に接続できるオペレーティング・システム・プロセスの最大数は、PROCESSES初期化パラメータによって決定します。
- DDLロック・タイムアウトの指定
  ブロッキングDDL文がロックを待機する時間を指定できます。
- UNDO領域管理方法の指定
  すべてのOracle Databaseには、データベースの変更を取り消すために使用する情報のメンテナンス方法が必要です。これらの情報は、主にコミットされる前のトランザクションの処理レコードから構成されます。これらのレコードを総称してUNDOデータと呼びます。
- データベース互換性レベルの指定
  データベースの互換レベルは、COMPATIBLE初期化パラメータによって制御されます。
- ライセンスに関するパラメータの設定
  指名ユーザー・ライセンスを使用している場合、Oracle Databaseではこの形式のライセンスを施行できます。データベース内に作成するユーザーの数に対して、制限を設定できます。この制限に達すると、それ以上のユーザーは作成できません。

関連項目:

- デフォルト設定など、すべての初期化パラメータに関する説明は、『Oracle Databaseリファレンス』を参照してください。
- メモリー管理に関係する初期化パラメータの説明は、「メモリーの管理」を参照してください

親トピック: Oracle Databaseの作成および構成

2.6.1 初期化パラメータと初期化パラメータ・ファイルについて

Oracleインスタンスの起動時に、初期化パラメータ・ファイルから初期化パラメータが読み込まれます。このファイルでは、少なくともDB_NAMEパラメータを指定する必要があります。他のすべてのパラメータにはデフォルトの値があります。

初期化パラメータ・ファイルは、読取り専用のテキスト・ファイル(PFILE)または読取り/書込みのバイナリ・ファイルです。

バイナリ・ファイルはサーバー・パラメータ・ファイルと呼ばれています。サーバー・パラメータ・ファイルを使用すると、ALTER

SYSTEMコマンドを使用して初期化パラメータを変更でき、変更内容は停止して起動した後も持続します。また、Oracle Databaseによる自己チューニングの基礎ともなります。このため、サーバー・パラメータ・ファイルを使用することをお薦めします。サーバー・パラメータ・ファイルは、編集済のテキスト形式の初期化ファイルから手動で作成するか、またはデータベースを作成するDatabase Configuration Assistant(DBCA)を使用して自動的に作成できます。

サーバー・パラメータ・ファイルを手動で作成する前に、テキスト形式の初期化パラメータ・ファイルを使用してインスタンスを起動できます。起動時に、Oracleインスタンスは最初にデフォルトの場所でサーバー・パラメータ・ファイルを検索し、見つからない場合は、テキスト形式の初期化パラメータ・ファイルを検索します。また、STARTUPコマンドの引数としてテキスト形式の初期化パラメータ・ファイルを指定すると、既存のサーバー・パラメータ・ファイルを上書きすることもできます。

テキスト形式の初期化パラメータ・ファイルのデフォルトのファイル名と場所は、次の表のとおりです。

プラットフォーム	デフォルト名	デフォルトの場所

UNIX および Linux	`initORACLE_SID.ora`
	
	たとえば、mynewdb データベースの初期化パラメータ・ファイル名は次のようになります。
	
	`initmynewdb.ora`
		ORACLE_HOME/dbs

Windows	`initORACLE_SID.ora`	ORACLE_HOME¥database

初めてOracle Databaseを作成する場合は、変更するパラメータ値の数を最小限にとどめておくことをお薦めします。データベースと環境に慣れてから、多数の初期化パラメータをALTER SYSTEM文で動的にチューニングします。テキスト形式の初期化パラメータ・ファイルを使用している場合、現行のインスタンスについてのみ変更できます。これを永続的にするには、初期化パラメータ・ファイル内で手動で更新する必要があり、更新しない場合は、次回データベースを停止して起動すると変更内容が失われます。サーバー・パラメータ・ファイルを使用している場合、ALTER SYSTEM文で行った初期化パラメータ・ファイルの変更内容は、停止して起動した後も持続します。

- 初期化パラメータ・ファイルのサンプル
  Oracle Databaseには通常、適切な値が設定されたテキスト形式の初期化パラメータのサンプルが用意されています。構成やオプション、およびデータベースのチューニング計画によっては、オラクル社が提供するこれらの初期化パラメータを編集し、他の初期化パラメータを追加することが可能です。
- テキスト形式の初期化パラメータ・ファイル
  テキスト形式の初期化パラメータ・ファイルでは、パラメータの値を名前/値ペアで指定します。

関連項目:

- DB_NAMEパラメータの詳細は、「グローバル・データベース名の決定」を参照してください
- 「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」
- 「初期化パラメータ・ファイルおよび起動について」

親トピック: 初期化パラメータの指定

2.6.1.1 初期化パラメータ・ファイルのサンプル

Oracle Databaseには通常、適切な値が設定されたテキスト形式の初期化パラメータのサンプルが用意されています。構成やオプション、およびデータベースのチューニング計画によっては、オラクル社が提供するこれらの初期化パラメータを編集し、他の初期化パラメータを追加することが可能です。

テキスト形式の初期化パラメータ・ファイルのサンプルは`init.ora`という名前で、ほとんどのプラットフォームの次の場所にあります。

```
ORACLE_HOME/dbs
```

サンプル・ファイルの内容は、次のとおりです。

```
###########################################################################
# Example INIT.ORA file
#
# This file is provided by Oracle Corporation to help you start by providing
# a starting point to customize your RDBMS installation for your site.
#
# NOTE: The values that are used in this file are only intended to be used
# as a starting point. You may want to adjust/tune those values to your
# specific hardware and needs. You may also consider using Database
# Configuration Assistant tool (DBCA) to create INIT file and to size your
# initial set of tablespaces based on the user input.
###########################################################################

# Change '<ORACLE_BASE>' to point to the oracle base (the one you specify at
# install time)

db_name='ORCL'
memory_target=1G
processes = 150
db_block_size=8192
db_domain=''
db_recovery_file_dest='<ORACLE_BASE>/flash_recovery_area'
db_recovery_file_dest_size=2G
diagnostic_dest='<ORACLE_BASE>'
dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
# You may want to ensure that control files are created on separate physical
# devices
control_files = (ora_control1, ora_control2)
compatible ='12.0.0'
```

親トピック: 初期化パラメータと初期化パラメータ・ファイルについて

2.6.1.2 テキスト形式の初期化パラメータ・ファイル

テキスト形式の初期化パラメータ・ファイルでは、パラメータの値を名前/値ペアで指定します。

テキスト形式の初期化パラメータ・ファイル(PFILE)には、次のいずれかの書式の名前/値ペアを含める必要があります。

- 値を1つのみ受け入れるパラメータの場合

  ```
  parameter_name=value
  ```

- 1つ以上の値を受け入れるパラメータの場合(`CONTROL_FILES`パラメータなど)

  ```
  parameter_name=(value[,value] ...)
  ```

文字列型のパラメータ値は、一重引用符(')で囲む必要があります。ファイル名の大/小文字区別が有効になるのは、ホスト・オ

ペレーティング・システムで有効な場合のみです。

複数の値を受け入れるパラメータの場合、アラート・ログから名前/値ペアを簡単にコピーして貼り付けられるように、複数行でパラメータを繰り返すことができます。この場合、各行には異なる値が含まれます。

```
control_files='/u01/app/oracle/oradata/orcl/control01.ctl'
control_files='/u01/app/oracle/oradata/orcl/control02.ctl'
control_files='/u01/app/oracle/oradata/orcl/control03.ctl'
```

複数の値を受け入れないパラメータを繰り返した場合、最後に指定した値のみが有効です。

関連項目:

- テキスト形式の初期化パラメータ・ファイルの内容と構文の詳細は、『Oracle Databaseリファレンス』を参照してください。
- 「アラート・ログ」

親トピック: 初期化パラメータと初期化パラメータ・ファイルについて

2.6.2 グローバル・データベース名の決定

グローバル・データベース名は、ユーザー指定のローカル・データベース名と、ネットワーク構造内でのデータベースの位置で構成されます。

- DB_NAMEおよびDB_DOMAIN初期化パラメータを設定します。

データベース名のローカル名コンポーネントはDB_NAME初期化パラメータによって決定し、ネットワーク構造内のドメイン(論理的な位置)はオプションで指定できるDB_DOMAINパラメータによって決定します。これら2つのパラメータの設定を組み合せて、ネットワーク内で一意になるデータベース名を形成する必要があります。

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

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

ALTER DATABASE RENAME GLOBAL_NAME文を使用すると、データベースのGLOBAL_NAMEを変更できます。ただし、最初にDB_NAMEおよびDB_DOMAIN初期化パラメータを変更し、制御ファイルを再作成した後に、データベースを停止して再起動する必要があります。制御ファイルの再作成は、ALTER DATABASE BACKUP CONTROLFILE TO TRACEコマンドで簡単に行えます。詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

- DB_NAME初期化パラメータ
  データベース識別子はDB_NAME初期化パラメータで指定します。
- DB_DOMAIN初期化パラメータ
  分散データベース・システムでは、DB_DOMAIN初期化パラメータには、ネットワーク構造内でのデータベースの論理上の位置を指定します。

関連項目:

データベース名の別の変更手段であるDBNEWIDユーティリティの使用方法は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: 初期化パラメータの指定

2.6.2.1 DB_NAME初期化パラメータ

データベース識別子はDB_NAME初期化パラメータで指定します。

DB_NAMEには、8文字以内のテキスト文字列を設定する必要があります。データベース名はアルファベットで開始する必要があります。DB_NAMEに指定した名前は、データベースの作成時に、データベースのデータファイル、REDOログ・ファイルおよび制御ファイルに記録されます。データベース・インスタンスの起動時に、(パラメータ・ファイル内の)DB_NAMEパラメータの値と制御ファイル内のデータベース名が一致しないと、データベースは起動しません。

親トピック: グローバル・データベース名の決定

2.6.2.2 DB_DOMAIN初期化パラメータ

分散データベース・システムでは、DB_DOMAIN初期化パラメータには、ネットワーク構造内でのデータベースの論理上の位置を指定します。

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

関連項目:

分散データベースの詳細は、「分散データベースの管理」を参照してください

親トピック: グローバル・データベース名の決定

2.6.3 高速リカバリ領域の指定

高速リカバリ領域とは、Oracle Databaseでバックアップおよびリカバリに関連するファイルを格納および管理できる場所のことです。これは、現在のデータベース・ファイル(データファイル、制御ファイルおよびオンラインREDOログ)の場所であるデータベース領域とは別です。

高速リカバリ領域を指定するには、次の初期化パラメータを使用します。

- DB_RECOVERY_FILE_DEST: 高速リカバリ領域の場所。ディレクトリ、ファイル・システムまたは自動ストレージ管理(Oracle ASM)のディスク・グループです。

  Oracle Real Application Clusters(Oracle RAC)環境では、この位置はクラスタ・ファイル・システム、Oracle ASMディスク・グループ、またはNFSを介して構成された共有ディレクトリであることが必要です。

- DB_RECOVERY_FILE_DEST_SIZE: 高速リカバリ領域で使用される最大総バイト数を指定します。この初期化パラメータは、DB_RECOVERY_FILE_DESTを使用可能にする前に指定する必要があります。

Oracle RAC環境では、これら2つのパラメータの設定がすべてのインスタンスで同じであることが必要です。

これらのパラメータは、LOG_ARCHIVE_DESTおよびLOG_ARCHIVE_DUPLEX_DESTパラメータの値を設定している場合は使用可能にできません。高速リカバリ領域を設定する前に、これらのパラメータを使用禁止にする必要があります。かわりに、LOG_ARCHIVE_DEST_nパラメータの値を設定できます。

高速リカバリ領域によってデータベースのバックアップ操作およびリカバリ操作が簡素化されるため、この領域を使用することをお薦めします。

関連項目:

高速リカバリ領域の作成方法と使用方法を学習するには、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: 初期化パラメータの指定

2.6.4 制御ファイルの指定

すべてのデータベースには、データベースの構造(名前、作成時のタイムスタンプ、データ・ファイルおよびREDOファイルの名前や場所など)を記述する制御ファイルがあります。CONTROL_FILES初期化パラメータには、1つ以上の制御ファイル名をカンマで区切って指定します。

- CONTROL_FILES初期化パラメータを設定します。

CREATE DATABASE文を実行すると、CONTROL_FILESパラメータに記述した制御ファイルが作成されます。

初期化パラメータ・ファイルにCONTROL_FILESを指定しないと、オペレーティング・システム固有のデフォルト・ファイル名を使用して、初期化パラメータ・ファイルと同じディレクトリにOracle Databaseが制御ファイルを作成します。Oracle Managed Filesが使用可能な場合は、Oracle Managed Filesの制御ファイルが作成されます。

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

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

関連項目:

- 「制御ファイルの管理」
- 「データベース作成時のOracle Managed Filesの作成」

親トピック: 初期化パラメータの指定

2.6.5 データベース・ブロック・サイズの指定

DB_BLOCK_SIZE初期化パラメータは、データベースの標準ブロック・サイズを指定します。

- DB_BLOCK_SIZE初期化パラメータを設定します。

このブロック・サイズはSYSTEM表領域に使用され、その他の表領域ではデフォルトで使用されます。Oracle Databaseは、最大4つの非標準ブロック・サイズをサポートします。

- DB_BLOCK_SIZE初期化パラメータ
  標準ブロック・サイズには、最も一般的に使用するブロック・サイズを選択します。多くの場合、設定が必要なブロック・サイズはこれのみです。
- 非標準ブロック・サイズ
  非標準ブロック・サイズの表領域を作成できます。

親トピック: 初期化パラメータの指定

2.6.5.1 DB_BLOCK_SIZE初期化パラメータ

標準ブロック・サイズには、最も一般的に使用するブロック・サイズを選択します。多くの場合、設定が必要なブロック・サイズはこれのみです。

- DB_BLOCK_SIZE初期化パラメータを設定します。

通常、DB_BLOCK_SIZEは4Kまたは8Kに設定します。このパラメータの値を指定しないと、オペレーティング・システム固有のデフォルト・データ・ブロック・サイズが使用されますが、大抵の場合、このデフォルト・ブロック・サイズで十分です。

データベースの作成後は、データベースを再作成する以外にブロック・サイズを変更する方法はありません。データベースのブロック・サイズがオペレーティング・システムのブロック・サイズと異なる場合は、データベースのブロック・サイズがオペレーティング・システムのブロック・サイズの倍数であることを確認してください。たとえば、使用しているオペレーティング・システムのブロック・サイズが2KB(2048バイト)の場合、次のDB_BLOCK_SIZE初期化パラメータの設定は有効です。

```
DB_BLOCK_SIZE=4096
```

データ・ブロック・サイズを大きくすると、ディスクとメモリーのI/O(データのアクセスと格納)の効率が向上します。したがって、次の条件に該当する場合は、オペレーティング・システムのブロック・サイズより大きいブロック・サイズの指定を考慮してください。

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

関連項目:

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

親トピック: データベース・ブロック・サイズの指定

2.6.5.2 非標準ブロック・サイズ

非標準ブロック・サイズの表領域を作成できます。

非標準ブロック・サイズの表領域を作成するには:

- CREATE TABLESPACE文でBLOCKSIZE句を指定します。

これらの非標準ブロック・サイズには、2の累乗である2KB、4KB、8KB、16KB、32KBのいずれかを指定します。最大ブロック・サイズに関するプラットフォーム固有の制限が適用されるので、プラットフォームによっては、これらのサイズの一部は指定できない場合があります。

非標準ブロック・サイズを使用する場合は、使用するすべての非標準ブロック・サイズについて、SGAメモリーのバッファ・キャッシュ領域内にサブキャッシュを構成する必要があります。これらのサブキャッシュの構成に使用する初期化パラメータについては、「自動共有メモリー管理の使用」を参照してください。

データベースに複数のブロック・サイズを指定できる機能は、特にデータベース間で表領域をトランスポートする場合に役立ちま

す。たとえば、OLTP環境から、8KBの標準ブロック・サイズを使用するデータ・ウェアハウス環境に、4KBのブロック・サイズを使用する表領域をトランスポートできます。

ノート:

32KB のブロック・サイズは、64 ビットのプラットフォームでのみ有効です。

注意:

セクター・サイズが 4KB のディスクを使用している場合、パフォーマンスが低下する可能性があるため、2KB のブロック・サイズを指定することはお薦めしません。詳細は、「REDO ログ・ファイルのブロック・サイズの計画」を参照してください。

関連項目:

- 「表領域の作成」
- 「データベース間での表領域のトランスポート」

親トピック: データベース・ブロック・サイズの指定

2.6.6 最大プロセス数の指定

Oracle Databaseに同時に接続できるオペレーティング・システム・プロセスの最大数は、PROCESSES初期化パラメータによって決定します。

- PROCESSES初期化パラメータを設定します。

このパラメータの値は、最低でも各バックグラウンド・プロセスごとに1つ、および各ユーザー・プロセスごとに1つです。バックグラウンド・プロセスの数は、使用しているデータベースの機能によって異なります。たとえば、アドバンスト・キューイングまたはファイル・マッピング機能を使用している場合は、追加のバックグラウンド・プロセスが必要です。自動ストレージ管理を使用している場合は、データベース・インスタンス用に追加プロセスを3つ追加します。

50のユーザー・プロセスを実行する予定の場合は、PROCESSES初期化パラメータの値を70に見積もって設定することをお薦めします。

親トピック: 初期化パラメータの指定

2.6.7 DDLロック・タイムアウトの指定

ブロッキングDDL文がロックを待機する時間を指定できます。

データ定義言語(DDL)文は非ブロッキングまたはブロッキングのいずれかで、どちらのタイプのDDL文にも内部構造の排他ロックが必要です。DDL文の実行時にこのようなロックが使用できない場合、非ブロッキングDDL文とブロッキングDDL文の動作は異なります。

- 非ブロッキングDDLは、DDLの影響を受けるオブジェクトを参照している同時DMLトランザクションがすべてコミットするかロール・バックするまで待機します。

- ブロッキングDDLは、ロックが使用できるようになった直後に実行していれば成功するような場合でも失敗します。

ブロッキングDDL文でロックを待機できるようにするには、DDLロック・タイムアウトを指定します。これは、DDLコマンドが必要なロックを待機する秒数であり、この秒数を超えるとDDL文は失敗します。

- DDLロック・タイムアウトを指定するには、DDL_LOCK_TIMEOUTパラメータを設定します。

設定可能なDDL_LOCK_TIMEOUTの値の範囲は、0から1,000,000です。デフォルトは0です。DDL_LOCK_TIMEOUTは、ALTER SESSION文を使用して、システム・レベルまたはセッション・レベルで設定できます。

ノート:

DDL_LOCK_TIMEOUT パラメータは、非ブロッキング DDL 文に作用しません。

関連項目:

- Oracle Databaseリファレンス
- Oracle Database開発ガイド
- Oracle Database SQL言語リファレンス

親トピック: 初期化パラメータの指定

2.6.8 UNDO領域管理方法の指定

すべてのOracle Databaseには、データベースの変更を取り消すために使用する情報の管理方法が必要です。これらの情報は、主にコミットされる前のトランザクションの処理レコードから構成されます。これらのレコードを総称してUNDOデータと呼びます。

UNDO表領域を使用する自動UNDO管理用の環境を設定する手順。

- UNDO_MANAGEMENT初期化パラメータをデフォルトのAUTOに設定します。

- UNDO_MANAGEMENT初期化パラメータ
  UNDO_MANAGEMENT初期化パラメータでは、インスタンスを自動UNDO管理モード(UNDOがUNDO表領域に格納されます)で起動するかどうかを指定します。自動UNDO管理モードを使用可能にするには、このパラメータをAUTOに設定します。パラメータを省略するか、NULLにすると、デフォルトでAUTOになります。
- UNDO_TABLESPACE初期化パラメータ
  UNDO_TABLESPACE初期化パラメータでは、インスタンスのデフォルトのUNDO表領域をオーバーライドできます。

関連項目:

UNDOの管理

親トピック: 初期化パラメータの指定

2.6.8.1 UNDO_MANAGEMENT初期化パラメータ

UNDO_MANAGEMENT初期化パラメータでは、インスタンスを自動UNDO管理モード(UNDOがUNDO表領域に格納されます)で起動するかどうかを指定します。自動UNDO管理モードを使用可能にするには、このパラメータをAUTOに設定します。パラメータを省略するか、NULLにすると、デフォルトでAUTOになります。

親トピック: UNDO領域管理方法の指定

2.6.8.2 UNDO_TABLESPACE初期化パラメータ

UNDO_TABLESPACE初期化パラメータでは、インスタンスのデフォルトのUNDO表領域をオーバーライドできます。

インスタンスを自動UNDO管理モードで起動すると、そのインスタンスは、UNDOデータを格納するためのUNDO表領域を選択しようとします。自動UNDO管理モードでデータベースが作成されている場合は、デフォルトのUNDO表領域(システム生成のSYS_UNDOTBS表領域またはユーザー指定のUNDO表領域)が、インスタンス起動時に使用されるUNDO表領域となります。インスタンスに対するこのデフォルトは、UNDO_TABLESPACE初期化パラメータに値を指定することで上書きできます。このパラメータは特に、Oracle Real Application Clusters環境のインスタンスに特定のUNDO表領域を割り当てる際に役立ちます。

UNDO表領域がUNDO_TABLESPACE初期化パラメータによって指定されていない場合は、データベース内で最初に使用可能なUNDO表領域が選択されます。使用可能なUNDO表領域がない場合、インスタンスはUNDO表領域のない状態で起動し、UNDOデータはSYSTEM表領域に書き込まれます。このモードでは実行しないようにしてください。

ノート:

CREATE DATABASE 文を使用してデータベースを作成するときには、UNDO_TABLESPACE パラメータを初期化パラメータ・ファイルに指定しないでください。かわりに、UNDO TABLESPACE 句を CREATE DATABASE 文に指定します。

親トピック: UNDO領域管理方法の指定

2.6.9 データベースの互換性レベルの指定

データベースの互換レベルは、COMPATIBLE初期化パラメータによって制御されます。

- COMPATIBLE初期化パラメータをリリース番号に設定します。
- COMPATIBLE初期化パラメータについて
  COMPATIBLE初期化パラメータによって、ディスクのファイル形式に影響を与える機能の使用をデータベースで使用可能または使用不可にできます。たとえば、Oracle Database 19cデータベースを作成しても、初期化パラメータ・ファイルにCOMPATIBLE=12.0.0を指定すると、Oracle Database 19cとの互換性が必要な機能の使用を試行した場合にエラーが生成されます。これは、データベースが12.0.0互換性レベルにあるとみなされているためです。

親トピック: 初期化パラメータの指定

2.6.9.1 COMPATIBLE初期化パラメータについて

COMPATIBLE初期化パラメータによって、ディスクのファイル形式に影響を与える機能の使用をデータベースで使用可能または使用不可にできます。たとえば、Oracle Database 19cデータベースを作成しても、初期化パラメータ・ファイルにCOMPATIBLE=12.0.0を指定すると、Oracle Database 19cとの互換性が必要な機能の使用を試行した場合にエラーが生成されます。これは、データベースが12.0.0互換性レベルにあるとみなされているためです。

COMPATIBLE初期化パラメータを変更することによって、データベースの互換性レベルを上げることができます。このようにすると、設定した互換性レベルよりも低いレベルではデータベースを起動できなくなりますが、互換性を上げる前の時点にPoint-in-Timeリカバリを実行することはできます。

COMPATIBLEパラメータのデフォルト値は、主要な最新リリースの番号です。

ノート:

- Oracle Database 19c の場合、COMPATIBLE パラメータのデフォルト値は 19.0.0 です。最小値は 12.0.0 です。デフォルト値を使用して Oracle Database を作成すると、このリリースのすべての新機能をすぐに使用できますが、データベースはダウングレードできなくなります。
- このパラメータを ALTER SYSTEM 文を使用してサーバー・パラメータ・ファイル(SPFILE)に設定する場合、SCOPE=SPFILE を指定し、データベースを再起動して変更を有効にする必要があります。
- COMPATIBLE 初期化パラメータは、19.0.0 のように、ドットで区切られた 3 桁以上の数字として指定する必要があります。

関連項目:

- データベースの互換性とCOMPATIBLE初期化パラメータの詳細は、『Oracle Databaseアップグレード・ガイド』を参照してください。
- Oracle Databaseリファレンス
- データベースのPoint-in-Timeリカバリの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: データベースの互換性レベルの指定

2.6.10 ライセンスに関するパラメータの設定

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

ノート:

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

データベースに作成するユーザー数を制限するには、そのデータベースの初期化パラメータ・ファイルにLICENSE_MAX_USERS初期化パラメータを設定します。

次の例は、LICENSE_MAX_USERS初期化パラメータを設定します。

```
LICENSE_MAX_USERS = 200
```

ノート:

Oracle では、同時セッションの数によるライセンス提供がなくなりました。したがって、LICENSE_MAX_SESSIONS および LICENSE_SESSIONS_WARNING 初期化パラメータは不要のため、非推奨になりました。

親トピック: 初期化パラメータの指定

2.7 サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

Oracle Databaseの初期化パラメータは、テキスト形式の初期化パラメータ・ファイルに格納されていました。管理性を向上させるために、データベースを起動および停止している間も持続するバイナリ形式のサーバー・パラメータ・ファイルによる初期化パラメータのメンテナンスを選択できます。

- サーバー・パラメータ・ファイルの概要
  サーバー・パラメータ・ファイルは、Oracle Databaseサーバーが稼働するシステムで管理される初期化パラメータのリポジトリと考えられます。これは、サーバー側の初期化パラメータ・ファイルとして設計されています。
- サーバー・パラメータ・ファイルへの移行
  現在、テキスト形式の初期化パラメータ・ファイルを使用している場合は、サーバー・パラメータ・ファイルに移行できます。
- サーバー・パラメータ・ファイルのデフォルトの名前と場所
  SPFILEの名前と格納場所には、データベースによるデフォルト設定を使用することをお薦めします。これにより、データベースの管理が容易になります。たとえば、STARTUPコマンドはこのデフォルトの場所を想定して、SPFILEを読み込みます。
- サーバー・パラメータ・ファイルの作成
  サーバー・パラメータ・ファイルを作成するには、CREATE SPFILE文を使用します。この文を実行するには、SYSDBA、SYSOPERまたはSYSBACKUP管理権限が必要です。
- SPFILE初期化パラメータ
  SPFILE初期化パラメータには、現在のサーバー・パラメータ・ファイルの名前を指定します。
- 初期化パラメータ値の変更
  データベース・インスタンスの操作に影響するように初期化パラメータ値を変更できます。
- 初期化パラメータ値のクリア
  ALTER SYSTEM RESET文を使用すると、初期化パラメータ値をクリアできます。これを行うと、初期化パラメータ値はデフォルト値または開始値に変更されます。
- サーバー・パラメータ・ファイルのエクスポート
  CREATE PFILE文を使用すると、サーバー・パラメータ・ファイル(SPFILE)をテキスト形式の初期化パラメータ・ファイルにエクスポートできます。
- サーバー・パラメータ・ファイルのバックアップの作成
  サーバー・パラメータ・ファイル(SPFILE)をエクスポートすることでバックアップを作成できます。データベースのバックアップおよびリカバリ計画がRecovery Manager(RMAN)を使用して実装されている場合は、RMANを使用してSPFILEのバックアップを作成できます。SPFILEのバックアップは、データベースのバックアップ作成時にRMANにより自動的に作成されますが、RMANを使用すると現在アクティブになっているSPFILEのバックアップも作成できます。
- 失われたまたは破損したサーバー・パラメータ・ファイルのリカバリ
  サーバー・パラメータ・ファイル(SPFILE)をリカバリできます。サーバー・パラメータ・ファイル(SPFILE)が失われるかまたは破損した場合は、現行インスタンスが失敗するか、またはデータベース・インスタンス起動時の次回の試行が失敗する可能性があります。
- パラメータ設定を表示する方法

いくつかの異なる方法を使用してパラメータ設定を表示できます。

親トピック: Oracle Databaseの作成および構成

2.7.1 サーバー・パラメータ・ファイルの概要

サーバー・パラメータ・ファイルは、Oracle Databaseサーバーが稼働するシステムで管理される初期化パラメータのリポジトリと考えられます。これは、サーバー側の初期化パラメータ・ファイルとして設計されています。

サーバー・パラメータ・ファイルに格納された初期化パラメータは持続性があり、インスタンスの実行中に加えられたパラメータへの変更はインスタンスの停止から起動までの間も持続します。この配置によって、ALTER SYSTEM文による変更を持続させるために、初期化パラメータを手動で更新する必要がなくなります。また、Oracle Databaseサーバーによる自己チューニングの基礎ともなります。

最初のサーバー・パラメータ・ファイルは、CREATE SPFILE文を使用して、テキスト形式の初期化パラメータ・ファイルから作成します。(Database Configuration Assistantで直接作成することもできます。)サーバー・パラメータ・ファイルは、テキスト・エディタで編集できないバイナリ・ファイルです。Oracle Databaseには、サーバー・パラメータ・ファイル内のパラメータの設定を表示および変更するために、他のインタフェースが用意されています。

ノート:

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

PFILE句を指定せずにSTARTUPコマンドを発行すると、Oracleインスタンスは、オペレーティング・システム固有のデフォルトの位置でサーバー・パラメータ・ファイルを検索し、そのファイルから初期化パラメータの設定を読み込みます。サーバー・パラメータ・ファイルが見つからない場合は、テキスト形式の初期化パラメータ・ファイルを検索します。サーバー・パラメータ・ファイルがあってもテキスト形式の初期化パラメータ・ファイルの設定を優先する場合は、STARTUPコマンドの発行時にPFILE句を指定する必要があります。サーバー・パラメータ・ファイルを使用してインスタンスを起動する方法の詳細は、「データベースの起動」を参照してください。

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.2 サーバー・パラメータ・ファイルへの移行

現在、テキスト形式の初期化パラメータ・ファイルを使用している場合は、サーバー・パラメータ・ファイルに移行できます。

サーバー・パラメータ・ファイルへ移行するには:

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

ノート:

Oracle Real Application Clusters 環境でサーバー・パラメータ・ファイルに移行する場合は、インスタンス固有のすべての初期化パラメータ・ファイルを、1 つの初期化パラメータ・ファイルに結合する必要があります。この操作の方法、および Oracle Real Application Clusters のインストールの一部であるインスタン

スでサーバー・パラメータ・ファイルを使用する際の固有の処理については、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』および使用しているプラットフォーム固有の Oracle Real Application Clusters インストレーション・ガイドを参照してください。

2. CREATE SPFILE FROM PFILE文を使用して、デフォルト位置にサーバー・パラメータ・ファイルを作成します。詳細は、「サーバー・パラメータ・ファイルの作成」を参照してください。

   この文を実行すると、テキスト形式の初期化パラメータ・ファイルが読み込まれて、サーバー・パラメータ・ファイルが作成されます。CREATE SPFILE文を発行するために、データベースを起動する必要はありません。

3. インスタンスを起動または再起動します。

   インスタンスはデフォルト位置にある新規SPFILEを検出し、そのファイルを使用して起動します。

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.3 サーバー・パラメータ・ファイルのデフォルトの名前と場所

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

次の表は、UNIX、LinuxおよびWindowsのプラットフォームごとに、テキスト形式の初期化パラメータ・ファイル(PFILE)とサーバー・パラメータ・ファイル(SPFILE)の両方のデフォルトの名前と場所を示しています(Oracle Automatic Storage Management(Oracle ASM)を使用する場合と使用しない場合について)。この表では、SPFILEをファイルと想定しています。

表2-3 UNIX、LinuxおよびWindowsにおけるPFILEおよびSPFILEのデフォルトの名前と場所

プラットフォーム	PFILEのデフォルト名	SPFILEのデフォルト名	PFILEのデフォルトの場所	SPFILEのデフォルトの場所
UNIX および Linux	initORACLE_SID.ora	spfileORACLE_SID.ora	Oracle_Home/dbs またはデータファイルと同じ場所	Oracle ASM を使用しない場合: Oracle_Home/dbs またはデータファイルと同じ場所 Oracle ASM が存在する場合: データファイルと同じディスク・グループ(データベースが DBCA によって作成されたことを想定)
Windows	initORACLE_SID.ora	spfileORACLE_SID.ora	Oracle_Home¥database	Oracle ASM を使用しない場合: OH¥database

プラットフォーム	PFILEのデフォルト名	SPFILEのデフォルト名	PFILEのデフォルトの場所	SPFILEのデフォルトの場所
				Oracle ASM が存在する場合: データファイルと同じディスク・グループ(データベースが DBCA によって作成されたことを想定)

ノート:

起動時に、インスタンスは最初に `spfileORACLE_SID.ora` という名前の SPFILE を検索し、見つからない場合は `spfile.ora` を検索します。`spfile.ora` を使用すると、すべての Real Application Clusters(Oracle RAC)インスタンスで同じサーバー・パラメータ・ファイルを使用できます。

いずれの SPFILE も見つからない場合、インスタンスはテキスト形式の初期化パラメータ・ファイル `initORACLE_SID.ora` を検索します。

デフォルトの場所以外の場所にSPFILEを作成する場合は、デフォルトのPFILEの場所に、そのサーバー・パラメータ・ファイルを指し示すスタブPFILEを作成する必要があります。詳細は、「データベースの起動」を参照してください。

Oracle ASMが存在する場合にDBCAを使用してデータベースを作成すると、DBCAによってSPFILEはOracle ASMディスク・グループ内に配置され、このスタブPFILEも作成されます。

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.4 サーバー・パラメータ・ファイルの作成

サーバー・パラメータ・ファイルを作成するには、`CREATE SPFILE`文を使用します。この文を実行するには、SYSDBA、SYSOPERまたはSYSBACKUP管理権限が必要です。

サーバー・パラメータ・ファイルを作成するには:

- `CREATE SPFILE`文を実行します。

ノート:

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

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

サーバー・パラメータ・ファイル(SPFILE)は、既存のテキスト形式の初期化パラメータ・ファイルまたはメモリーから作成できます。メモリーからのSPFILEの作成では、実行中のインスタンス内の初期化パラメータの現行値をSPFILEにコピーします。

次の例では、テキスト形式の初期化パラメータ・ファイル/u01/oracle/dbs/init.oraからサーバー・パラメータ・ファイルを作成しています。この例ではSPFILEの名前を指定していないため、ファイルは、表2-3に示したプラットフォーム固有のデフォルトの名前と場所で作成されます。

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

次の例では、名前と場所を指定してサーバー・パラメータ・ファイルを作成しています。

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

次の例では、メモリー内の初期化パラメータの現行値から、デフォルトの場所にサーバー・パラメータ・ファイルを作成しています。

```
CREATE SPFILE FROM MEMORY;
```

デフォルトのSPFILE名と場所を使用するか、SPFILE名と場所を指定するかどうかに関係なく、同じ名前のSPFILEがその場所にすでに存在する場合は、警告メッセージなしに上書きされます。

テキスト形式の初期化パラメータ・ファイルからSPFILEを作成すると、初期化パラメータ・ファイル内のパラメータ設定と同じ行に記述されているコメントもSPFILEで管理されます。他のコメントはすべて無視されます。

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.5 SPFILE初期化パラメータ

SPFILE初期化パラメータには、現在のサーバー・パラメータ・ファイルの名前を指定します。

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

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.6 初期化パラメータ値の変更

データベース・インスタンスの操作に影響する初期化パラメータ値を変更できます。

- 初期化パラメータ値の変更について
  ALTER SYSTEM文を使用すると、初期化パラメータ値を設定、変更またはデフォルトにリストアできます。テキスト形式の初期化パラメータ・ファイルを使用している場合、現行のインスタンスに対するパラメータの値のみALTER SYSTEM文で変更されます。これは、ディスク上のテキスト形式の初期化パラメータを自動的に更新するメカニズムがないためです。以後のインスタンスに渡すためには、ディスク上の初期化パラメータを手動で更新する必要があります。サーバー・パラメータ・ファイルを使用している場合は、このような制限はありません。
- 初期化パラメータ値の設定または変更
  サーバー・パラメータ・ファイルで、ALTER SYSTEM文のSET句を使用して初期化パラメータ値を設定または変更します。

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.6.1 初期化パラメータ値の変更について

ALTER SYSTEM文を使用すると、初期化パラメータ値を設定、変更またはデフォルトにリストアできます。テキスト形式の初期

化パラメータ・ファイルを使用している場合、現行のインスタンスに対するパラメータの値のみALTER SYSTEM文で変更されます。これは、ディスク上のテキスト形式の初期化パラメータを自動的に更新するメカニズムがないためです。以後のインスタンスに渡すためには、ディスク上の初期化パラメータを手動で更新する必要があります。サーバー・パラメータ・ファイルを使用している場合は、このような制限はありません。

初期化パラメータには、次の2種類があります。

- 動的な初期化パラメータ: 現行のOracle Databaseインスタンスに対して変更できます。変更は即時に有効になります。
- 静的な初期化パラメータ: 現行のインスタンスに対して変更できません。これらのパラメータはテキスト形式の初期化パラメータまたはサーバー・パラメータ・ファイルで変更し、変更を有効にするにはデータベースを再起動する必要があります。

親トピック: 初期化パラメータ値の変更

2.7.6.2 初期化パラメータ値の設定または変更

サーバー・パラメータ・ファイルで、ALTER SYSTEM文のSET句を使用して初期化パラメータ値を設定または変更します。

- ALTER SYSTEM SET文を実行します。

たとえば、次の文は、接続を中断するまでのログイン試行の最大失敗回数を変更します。ここにはコメントが含まれており、変更をサーバー・パラメータ・ファイルにのみ適用することを明示しています。

```
ALTER SYSTEM SET SEC_MAX_FAILED_LOGIN_ATTEMPTS=3
                 COMMENT='Reduce from 10 for tighter security.'
                 SCOPE=SPFILE;
```

次の例では、属性のリストをとる複雑な初期化パラメータを設定しています。値を設定するパラメータは、LOG_ARCHIVE_DEST_n初期化パラメータです。この文によって、このパラメータの既存の設定を変更するか、または新しいアーカイブ先を作成できます。

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

値がパラメータのリストで構成されている場合は、位置または序数によって個々の属性を編集することはできません。パラメータを更新するたびに、完全な値のリストを指定する必要があります。これによって、新しいリストで古いリストが完全に置換されます。

- ALTER SYSTEM SET文のSCOPE句
  ALTER SYSTEM SET文のオプションのSCOPE句では、初期化パラメータの変更のスコープを指定します。

親トピック: 初期化パラメータ値の変更

2.7.6.2.1 ALTER SYSTEM SET文のSCOPE句

ALTER SYSTEM SET文のオプションのSCOPE句では、初期化パラメータの変更のスコープを指定します。

SCOPE句	説明
SCOPE = SPFILE	サーバー・パラメータ・ファイルのみに変更が適用されます。その場合、それぞれ次のような結果になります。 ● 変更は現行インスタンスには適用されません。

SCOPE句	説明
	● 動的パラメータと静的パラメータの両方について、次の起動時に変更が有効になり、以後持続します。 静的パラメータでは、SCOPE 指定のみ使用できます。
SCOPE = MEMORY	メモリーのみに変更が適用されます。その場合、それぞれ次のような結果になります。 ● 変更は現行インスタンスに適用され、即時に有効になります。 ● 動的パラメータの場合は、変更が即時に有効になりますが、サーバー・パラメータ・ファイルは更新されないので、変更は持続しません。 静的パラメータでは、この指定は使用できません。
SCOPE = BOTH	サーバー・パラメータ・ファイルとメモリーの両方に変更が適用されます。その場合、それぞれ次のような結果になります。 ● 変更は現行インスタンスに適用され、即時に有効になります。 ● 動的パラメータの場合は、サーバー・パラメータ・ファイルが更新されるので、変更が持続します。 静的パラメータでは、この指定は使用できません。

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

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

SCOPEをSPFILEまたはBOTHに指定した場合は、オプションのCOMMENT句を使用すると、パラメータの更新にテキスト文字列を関連付けることができます。サーバー・パラメータ・ファイルにコメントが記述されます。

親トピック: 初期化パラメータ値の設定または変更

2.7.7 初期化パラメータ値のクリア

ALTER SYSTEM RESET文を使用すると、初期化パラメータ値をクリアできます。これを行うと、初期化パラメータ値はデフォルト値または開始値に変更されます。

ALTER SYSTEM RESET文にはSCOPE句が含まれています。ALTER SYSTEM RESET文およびSCOPE句を非CDBまた

はマルチテナント・コンテナ・データベース(CDB)ルートで実行した場合は、この文がプラガブル・データベース(PDB)、アプリケーション・ルートまたはアプリケーションPDBで実行された場合と異なる動作となります。

パラメータの開始値は、インスタンスの起動またはPDBのオープンが完了した後のメモリー内のパラメータの値です。この値は、起動直後にV$SYSTEM_PARAMETERビューのVALUE列およびDISPLAY_VALUE列に表示されます。パラメータの値は起動時に内部的に調整されることがあるため、開始値はspfileの値またはデフォルト値(spfileにパラメータが設定されていない場合)と異なる場合があります。

ALTER SYSTEM RESET文のSCOPE値は、非CDBおよびCDBのCDB$ROOTでは次のように動作します。

- SCOPE=SPFILE: インスタンスでspfileが使用されている場合は、spfileからパラメータを削除します。次のインスタンスの起動時に、デフォルト値が有効になります。
- SCOPE=MEMORY: 開始値がすぐに有効になります。ただし、変更はインスタンスのspfileに格納されず、インスタンスの再起動時に失われます。
- SCOPE=BOTH: インスタンスでspfileが使用されている場合は、spfileからパラメータを削除します。デフォルト値が即座に有効になり、この変更はインスタンスを再起動しても維持されます。

ノート:

SCOPE=BOTH は SCOPE=MEMORY の動作を変更します。SCOPE=BOTH を発行すると、その後の SCOPE=MEMORY では常にパラメータがデフォルト値にリセットされます。

ALTER SYSTEM RESET文のSCOPE値は、PDB、アプリケーション・ルートまたはアプリケーションPDBでは次のように動作します。

- SCOPE=SPFILE: コンテナのspfileからパラメータを削除します。コンテナは次にPDBがオープンされたときにルートからパラメータ値を継承します。
- SCOPE=MEMORY: 次に2つの例を示します。
  - パラメータはコンテナがオープンされたときにコンテナのspfileに存在します。パラメータ値はパラメータの開始値に更新されます。この変更はコンテナのspfileに格納されず、次にコンテナがオープンされたときに失われます。
  - パラメータはコンテナがオープンされたときにコンテナのspfileに存在しません。コンテナはルートからパラメータ値を継承して起動されます。
- SCOPE=BOTH: コンテナのspfileからパラメータを削除します。コンテナはルートからパラメータ値を継承します。

ノート:

- SCOPE=BOTH は SCOPE=MEMORY の動作を変更します。SCOPE=BOTH が発行された後に SCOPE=MEMORY が発行されると、コンテナは常にルートからパラメータ値を継承します。
- コンテナがルートからパラメータ値を継承する場合、PDB は CDB$ROOT から値を継承します。アプリケーション・コンテナでは、アプリケーション PDB はアプリケーション・ルートからパラメータ値を継承し、アプリケーション・ルートは CDB$ROOT からパラメータ値を継承します。

関連項目:

- ALTER SYSTEMコマンドの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- Oracle Multitenant管理者ガイド

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.8 サーバー・パラメータ・ファイルのエクスポート

CREATE PFILE文を使用すると、サーバー・パラメータ・ファイル(SPFILE)をテキスト形式の初期化パラメータ・ファイルにエクスポートできます。

- CREATE PFILE文を実行します。

いくつかの理由でサーバー・パラメータ・ファイルのエクスポートが必要な場合があります。

- 診断のために、現在インスタンスで使用しているすべてのパラメータのリストを作成する場合。これは、SQL*PlusのSHOW PARAMETERSコマンド、あるいはV$PARAMETERまたはV$PARAMETER2ビューの選択と同じです。
- 最初にエクスポートし、結果のテキスト・ファイルを編集してから、CREATE SPFILE文を使用して再作成するという手順で、サーバー・パラメータ・ファイルを変更する場合

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

CREATE PFILE文を実行するには、SYSDBA、SYSOPERまたはSYSBACKUP管理権限が必要です。エクスポート・ファイルは、データベース・サーバー・システム上に作成されます。そこには、パラメータ設定と同じ行に記述されているパラメータに関するコメントがすべて含まれます。

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

```
CREATE PFILE FROM SPFILE;
```

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

次の例では、サーバー・パラメータ・ファイルからテキスト形式の初期化パラメータ・ファイルを作成していますが、複数のファイル名が指定されています。

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

ノート:

メモリー内の初期化パラメータの現行値から PFILE を作成する方法もあります。次は、必要なコマンドの例です。

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

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.9 サーバー・パラメータ・ファイルのバックアップ

サーバー・パラメータ・ファイル(SPFILE)をエクスポートすることでバックアップを作成できます。データベースのバックアップおよびリカバリ計画がRecovery Manager(RMAN)を使用して実装されている場合は、RMANを使用してSPFILEのバックアップを作

成できます。SPFILEのバックアップは、データベースのバックアップ作成時にRMANにより自動的に作成されますが、RMANを使用すると現在アクティブになっているSPFILEのバックアップも作成できます。

- エクスポートするかRMANを使用してサーバー・パラメータ・ファイルをバックアップします。

関連項目:

「サーバー・パラメータ・ファイルのエクスポート」

Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.10 失われたまたは破損したサーバー・パラメータ・ファイルのリカバリ

サーバー・パラメータ・ファイル(SPFILE)をリカバリできます。サーバー・パラメータ・ファイル(SPFILE)が失われるかまたは破損した場合は、現行インスタンスが失敗するか、またはデータベース・インスタンス起動時の次回の試行が失敗する可能性があります。

SPFILEをリカバリする方法は複数あります。

- インスタンスが実行中の場合は、次のコマンドを発行して、メモリー内の初期化パラメータの現行値からSPFILEを再作成します。

  ```
  CREATE SPFILE FROM MEMORY;
  ```

  このコマンドでは、SPFILEはデフォルトの場所にデフォルトの名前で作成されます。新しい名前を使用して、または指定した場所にSPFILEを作成することもできます。例については、「サーバー・パラメータ・ファイルの作成」を参照してください。

- 有効なテキスト形式の初期化パラメータ・ファイル(PFILE)がある場合は、次の文を使用してPFILEからSPFILEを再作成します。

  ```
  CREATE SPFILE FROM PFILE;
  ```

  このコマンドは、PFILEがデフォルトの場所にあり、デフォルトの名前であると想定しています。PFILEがデフォルト以外の場所にあるか、デフォルト以外の名前の場合に使用するコマンド構文については、「サーバー・パラメータ・ファイルの作成」を参照してください。

- バックアップからSPFILEをリストアします。

  詳細は、「サーバー・パラメータ・ファイルのバックアップの作成」を参照してください。

- これらのいずれの方法も使用できない状況の場合は、次のステップを実行します。

  1. アラート・ログのパラメータ値のリストからテキスト形式の初期化パラメータ・ファイル(PFILE)を作成します。

     インスタンスの起動時に、起動に使用された初期化パラメータがアラート・ログに書き込まれます。このセクションを(XMLタグが含まれていない)テキスト・バージョンのアラート・ログからコピーして、新規PFILEに貼り付けることができます。

     詳細は、「アラート・ログの表示」を参照してください。

  2. PFILEからSPFILEを作成します。

     詳細は、「サーバー・パラメータ・ファイルの作成」を参照してください。

パラメータ更新時の読取り/書込みエラー

パラメータの更新時にサーバー・パラメータ・ファイルの読込みまたは書込みでエラーが発生した場合は、アラート・ログにエラーが記録されて、サーバー・パラメータ・ファイルに対する後続のパラメータ更新がすべて無視されます。この時点では、次の処理のいずれかを実行できます。

- インスタンスを停止し、この項の前述の説明に従ってサーバー・パラメータ・ファイルをリカバリした後、インスタンスを再起動します。
- 後続のパラメータ更新を持続的にするための処置を講じない場合は、データベースの実行を継続します。

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.7.11 パラメータ設定を表示する方法

いくつかの異なる方法を使用してパラメータ設定を表示できます。

方法	説明
SHOW PARAMETERS	この SQL*Plus コマンドを使用すると、現行セッションで有効な初期化パラメータの値が表示されます。
SHOW SPPARAMETERS	この SQL*Plus コマンドを使用すると、サーバー・パラメータ・ファイル(SPFILE)の初期化パラメータの値が表示されます。
CREATE PFILE	この SQL 文は、SPFILE または現在のインメモリー設定からテキスト形式の初期化パラメータ・ファイル(PFILE)を作成します。PFILE はテキスト・エディタで表示できます。
V$PARAMETER	このビューを使用すると、現行セッションで有効な初期化パラメータの値が表示されます。
V$PARAMETER2	このビューを使用すると、現行セッションで有効な初期化パラメータの値が表示されます。各リスト・パラメータ値が別々の行に出力されるため、このビューの方がリスト・パラメータ値を容易に識別できます。
V$SYSTEM_PARAMETER	このビューを使用すると、インスタンスで有効な初期化パラメータの値が表示されます。新しいセッションは、インスタンス全体の値からパラメータ値を継承します。
V$SYSTEM_PARAMETER2	このビューを使用すると、インスタンスで有効な初期化パラメータの値が表示されます。新しいセッションは、インスタンス全体の値からパラメータ値を継承します。各リスト・パラメータ値が別々の行に出力されるため、このビューの方がリスト・パラメータ値を容易に識別できます。
V$SPPARAMETER	このビューを使用すると、SPFILE の現在の内容が表示されます。インスタンスで SPFILE が使用されていない場合は、ISSPECIFIED 列に FALSE が返されます。

関連項目:

ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

2.8 データベース・サービスでのアプリケーション・ワークロードの管理

データベース・サービスは、1つ以上のデータベース・インスタンスをある単位にまとめて表現したものです。サービスを使用すれば、データベースのワークロードをグループ化し、特定の作業リクエストを適切なインスタンスにルーティングできます。

- データベース・サービス
  データベース・サービスは1つのデータベースを表します。このデータベースは、単一インスタンス・データベースか、または複数の同時データベース・インスタンスが含まれたOracle Real Application Clusters (Oracle RAC)データベースとなります。グローバル・データベース・サービスは、データ・レプリケーションを介して同期化された複数のデータベースが提供するサービスです。
- グローバル・データ・サービス
  Oracle Database 12cからは、複数のOracle Databaseが関連するワークロードを管理するために、グローバル・データ・サービス(GDS)を使用できます。GDSを使用すると、管理者は、共通のサービスを提供する複数のレプリケート・データベースにまたがるクライアント・ワークロードを自動的かつ透過的に管理できます。これらの共通サービスは、グローバル・サービスと呼ばれます。
- データベース・サービスのデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューを問い合せて、データベース・サービスに関する情報を検索できます。

親トピック: Oracle Databaseの作成および構成

2.8.1 データベース・サービス

データベース・サービスは1つのデータベースを表します。このデータベースは、単一インスタンス・データベースか、または複数の同時データベース・インスタンスが含まれたOracle Real Application Clusters (Oracle RAC)データベースとなります。グローバル・データベース・サービスは、データ・レプリケーションを介して同期化された複数のデータベースが提供するサービスです。

- データベース・サービスについて
  データベース・サービスは、1つのデータベースのワークロードを互いに共通の要素を持たないグループに分割します。
- データベース・サービスおよびパフォーマンス
  データベース・サービスは、パフォーマンス・チューニングに追加のディメンションを提供します。
- データベース・サービスを使用するOracle Databaseの機能
  Oracle Database機能のいくつかは、データベース・サービスをサポートしています。
- データベース・サービスの作成
  データベース・サービスの作成方法はデータベースの構成によっていくつかあります。

親トピック: データベース・サービスでのアプリケーション・ワークロードの管理

2.8.1.1 データベース・サービスについて

データベース・サービスは、1つのデータベースのワークロードを互いに共通の要素を持たないグループに分割します。

各データベース・サービスは、一般的な属性、サービス・レベルしきい値および優先度でワークロードを表します。グループ化は作業の属性に基づいて行われますが、これらの属性には、使用するアプリケーション機能、アプリケーション機能を実行する場合の

優先度、管理の対象になるジョブ・クラス、アプリケーション機能またはジョブ・クラスで使用するデータの範囲などが含まれていることがあります。たとえば、Oracle E-Business Suiteでは、総勘定元帳、売掛金勘定、受注など、職務ごとにサービスを定義します。データベース・サービスを構成するとき、一意の名前、関連するパフォーマンス目標および関連する重要性を各サービスに指定します。これらのデータベース・サービスは、Oracle Databaseと緊密に統合され、データ・ディクショナリに保持されます。

接続要求には、データベース・サービス名を指定できます。このように、中間層アプリケーションおよびクライアントとサーバーのアプリケーションでは、データベース・サービスをTNS接続データ内の接続の一部として指定することで、サービスを利用します。データベース・サービス名が指定されておらず、Net Servicesファイルlistener.oraにデフォルトのデータベース・サービスが指定されている場合は、デフォルトのデータベース・サービスを使用して接続されます。

データベース・サービスを使用すると、1つのデータベースのワークロードの構成、管理、有効化と無効化を実行でき、さらに単一エンティティとして測定できます。これは、Database Configuration Assistant (DBCA)、Oracle Net Configuration Assistant、Oracle Enterprise Manager Cloud Control (Cloud Control)などの標準的なツールを使用して実行できます。Cloud Controlは、表示と操作に関するサービスを全体としてサポートし、必要な場合はインスタンス・レベルへのドリルダウンをサポートしています。

Oracle Real Application Clusters(Oracle RAC)環境では、データベース・サービスは1つ以上のインスタンスにまたがり、トランザクション・パフォーマンスに基づいたワークロードの均衡化に役立ちます。この機能によって、エンドツーエンドの無人リカバリ、ワークロードによるロール変更、位置の完全な透過性が可能となります。また、Oracle RACを使用すると、Cloud Control、DBCAおよびServer Controlユーティリティ(SRVCTL)で複数のデータベース・サービス機能を管理できます。

データベース・サービスには、アプリケーション、アプリケーション機能およびデータの範囲が、機能サービスまたはデータ依存サービスとして記述されています。機能サービスは最も一般的なワークロードのマッピングです。特定の機能を使用する複数のセッションはまとめてグループ化されます。これに対して、データ依存ルーティングは、データ・キーに基づいてセッションをデータベース・サービスにルーティングします。作業要求のデータベース・サービスへのマッピングは、アプリケーション・サーバーとTPモニターのオブジェクト関連マッピング・レイヤーで発生します。たとえば、Oracle RACでは、データベースが共有されているため、これらの範囲は要求に基づいて完全に動的にできます。

アプリケーションにより使用されるデータベース・サービスに加えて、Oracle Databaseでは、2つの内部データベース・サービスもサポートされています。SYS$BACKGROUNDはバックグラウンド・プロセスのみで使用され、SYS$USERSはサービスに関連していないユーザー・セッションに対するデフォルトのデータベース・サービスです。

データベース・サービスを使用する場合にアプリケーション・コードを変更する必要はありません。クライアント側の作業は、指定したデータベース・サービスに接続できます。Oracle Scheduler、パラレル実行、Oracle Databaseアドバンスト・キューイングなどのサーバー側での作業では、ワークロード定義の一部としてデータベース・サービス名を設定します。データベース・サービス下で実行される作業要求は、そのサービスのパフォーマンスしきい値を継承し、サービスの一部として測定されます。

関連項目:

- 『Oracle Database概要』
- Oracle RAC環境でのサービスの利用の詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。
- サービスへの接続の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。
- Cloud Controlのオンライン・ヘルプ

親トピック: データベース・サービス

2.8.1.2 データベース・サービスおよびパフォーマンス

データベース・サービスは、パフォーマンス・チューニングに追加のディメンションを提供します。

すべてのセッションを匿名で共有している大部分のシステムでは、「サービスとSQL」によるチューニングで「セッションとSQL」によるチューニングを置換できます。データベース・サービスを使用することで、ワークロードが表示可能および測定可能となります。リソースの使用と待機は、アプリケーションがその起因となっています。また、データベース・サービスに割り当てたリソースは、ロードの増減にあわせて調整できます。この動的なリソース割当てによって、要求の発生に対応した費用効率の高いソリューションが可能となります。たとえば、データベース・サービスを自動的に測定し、そのパフォーマンスをサービス・レベルのしきい値と比較できます。パフォーマンス違反はCloud Controlにレポートされるため、自動ソリューションまたはスケジュールされたソリューションを実行できます。

親トピック: データベース・サービス

2.8.1.3 データベース・サービスを使用するOracle Databaseの機能

Oracle Database機能のいくつかは、データベース・サービスをサポートしています。

自動ワークロード・リポジトリ(AWR)は、サービスのパフォーマンスを管理します。AWRには、実行時間、待機クラスおよびサービスで使用されたリソースも含めて、データベース・サービスのパフォーマンスが記録されます。AWRは、データベース・サービス応答時間がしきい値を超えた場合、警告をアラートします。動的なビューには、現在のサービス・パフォーマンスのメトリックが時間の履歴単位でレポートされます。各データベース・サービスには、応答時間とCPU使用に関するサービス品質のしきい値があります。

さらに、データベース・リソース・マネージャでは、データベース・サービスをコンシューマ・グループにマッピングできます。これによって、データベース・サービスの優先度を他のサービスと関連させて自動的に管理できます。コンシューマ・グループを使用すると、比率またはリソース使用量の観点から相対的な優先順序を定義できます。

データベース・サービスのエディション属性を指定することもできます。エディションを使用すると、データベース内に同じオブジェクトの複数のバージョンを保持できます。データベース・サービスにエディションを指定すると、そのデータベース・サービスを指定するそれ以降のすべて接続で、初期セッション・エディションとしてこのエディションが使用されます。

エディションをデータベース・サービス属性として指定すると、リソース使用の管理が容易になります。たとえば、1つのエディションに関連付けられた複数のデータベース・サービスを1つのOracle RAC環境における個別のインスタンスに配置でき、データベース・リソース・マネージャは、リソース・プランを対応するデータベース・サービスに関連付けることによって、異なるエディションで使用されているリソースを管理できます。

Oracle Schedulerでは、ジョブ・クラスの作成時にオプションでデータベース・サービスを割り当てることができます。実行中に複数のジョブがジョブ・クラスに割り当てられ、データベース・サービス内で複数のジョブ・クラスを実行できます。ジョブ・クラスとともにデータベース・サービスを使用することで、ジョブ・スケジューラによって実行される作業が、ワークロード管理とパフォーマンス・チューニングに対して示されます。

パラレル問合せとパラレルDMLの場合、問合せコーディネータは他のクライアントと同じようにデータベース・サービスに接続します。パラレル問合せプロセスは、実行中そのデータベース・サービスを継承します。問合せが終了した時点で、パラレル実行プロセスはデフォルトのデータベース・サービスに戻ります。

関連項目:

- 「Oracle Database Resource Managerを使用したリソースの管理」
- 「コンシューマ・グループへのセッションのマッピング・ルールの指定」
- 「データベース・サービスのエディション属性の設定」

- 「Oracle Schedulerを使用したジョブのスケジューリング」

親トピック: データベース・サービス

2.8.1.4 データベース・サービスの作成

データベース・サービスの作成方法はデータベースの構成によっていくつかあります。

ノート:

Oracle Database 19c 以降、SERVICE_NAMES パラメータをお客様が使用することは非推奨になりました。今後のリリースでサポートが終了する可能性があります。サービスを管理するには、SRVCTL または GDSCTL コマンドライン・ユーティリティ、または DBMS_SERVICE パッケージを使用することをお薦めします。

ノート:

この項では、ローカルでのサービスの作成について説明します。サービスを作成してグローバルに操作することもできます。詳細は、「グローバル・データ・サービス」を参照してください。

データベース・サービスを作成するには:

- 単一インスタンスのデータベースをOracle Restartで管理している場合は、SRVCTLユーティリティを使用してデータベース・サービスを作成します。

```
srvctl add service -db db_unique_name -service service_name
```

- Oracle Restartで管理していない単一インスタンスのデータベースでは、次のいずれかを実行します。
  - SERVICE_NAMESパラメータに希望するデータベース・サービス名を追加します。
  - DBMS_SERVICE.CREATE_SERVICEパッケージ・プロシージャをコールします。
- (オプション)データベース・サービス属性をCloud ControlまたはDBMS_SERVICE.MODIFY_SERVICEで定義します。

Oracle Netリスナー(リスナー)はクライアントからの着信接続要求を受信し、データベース・サーバーへのこれらの要求のトラフィックを管理します。リスナーは、登録済サービスの接続を処理し、動的サービス登録をサポートしています。

関連項目:

- Oracle Restartの詳細は、「Oracle Databaseの自動再起動の構成」を参照してください
- DBMS_SERVICEパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- Oracle RAC環境でのサービスの作成の詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。
- Oracle Netリスナーおよびサービスの詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

親トピック: データベース・サービス

2.8.2 グローバル・データ・サービス

Oracle Database 12cからは、複数のOracle Databaseが関連するワークロードを管理するために、グローバル・データ・サービス(GDS)を使用できます。GDSを使用すると、管理者は、共通のサービスを提供する複数のレプリケート・データベースにまたがるクライアント・ワークロードを自動的かつ透過的に管理できます。これらの共通サービスは、グローバル・サービスと呼ばれます。

GDSにより、様々な場所に存在する複数のデータベースを、グローバル・クライアントが共有できるプライベートGDS構成に統合できます。次の利点があります。

- グローバル・リソースの集中管理が可能になります。
- グローバルなスケーラビリティ、可用性およびランタイム・ロード・バランシングを提供します。
- データベースをGDS構成に動的に追加したり、グローバル・サービスを動的に移行できます。
- Oracle Active Data Guard、Oracle GoldenGateなどの機能を使用するレプリケート・データベースの分散環境でのサービス管理、ロード・バランシングおよびフェイルオーバーの各機能を拡張します。
- (ローカルまたはグローバルの両方に配置された)複数のデータベースをまたいで、グローバル・サービスをシームレスにフェイルオーバーすることによって、高可用性を提供します。
- サービス、接続ロード・バランシングおよびランタイム・ロード・バランシングにより、データ・センター内およびデータ・センター間の両方でのワークロード・バランシングを提供します。
- サービス・クライアントの要求に応じて、GDS構成のリソースを効率的に使用できます。

関連項目:

- Oracle Database Global Data Services概要および管理ガイド
- 『Oracle Database概要』

親トピック: データベース・サービスでのアプリケーション・ワークロードの管理

2.8.3 データベース・サービスのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューを問い合せて、データベース・サービスに関する情報を検索できます。

データベース・サービスに関する情報は、次のビューで参照できます。

- DBA_SERVICES
- ALL_SERVICESまたはV$SERVICES
- V$ACTIVE_SERVICES
- V$SERVICE_STATS
- V$SERVICE_EVENT
- V$SERVICE_WAIT_CLASS
- V$SERV_MOD_ACT_STATS

- V$SERVICEMETRIC
- V$SERVICEMETRIC_HISTORY

次の追加ビューにも、データベース・サービスに関する情報が表示されます。

- V$SESSION
- V$ACTIVE_SESSION_HISTORY
- DBA_RSRC_GROUP_MAPPINGS
- DBA_SCHEDULER_JOB_CLASSES
- DBA_THRESHOLDS

ALL_SERVICESビューにはGLOBAL_SERVICE列が含まれ、V$SERVICESビューとV$ACTIVE_SERVICESビューにはGLOBAL列が含まれます。これらのビューおよび列により、データベース・サービスがグローバル・サービスであるかどうかを判断できます。

親トピック: データベース・サービスでのアプリケーション・ワークロードの管理

2.9 Oracle DatabaseのStandard Edition高可用性の管理

Standard Edition高可用性機能は、Oracle Clusterwareを使用したOracle Database Standard Edition 2単一インスタンス・データベースに向けて計画外停止に対する保護を提供します。

ノート:

Standard Edition 高可用性は、Oracle Database 19c リリース 19.7 以降でサポートされます。

ノート:

Standard Edition 高可用性で使用する `srvctl` コマンドは、Oracle Restart で使用するものとは異なります。Standard Edition 高可用性の場合、`srvctl` コマンドとは、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』で説明されているものを表します。

- Standard Edition高可用性について
  Oracle Database 19c以降では、高可用性モードでOracle Database Standard Edition 2をインストールできます。
- Oracle DatabaseによるStandard Edition高可用性を使用するための要件
  Standard Edition高可用性を使用する場合は、次に示す構成の要件に従ってOracle Database Standard Edition 2をデプロイします。
- Oracle DatabaseのStandard Edition高可用性の有効化
  Standard Edition高可用性を有効化することで、Oracle Database Standard Edition 2データベースにクラスタ・ベースのフェイルオーバーを実現します。
- Standard Edition高可用性データベースの別のノードへの再配置
  計画停止を管理するために、Standard Edition高可用性を使用するOracle Database Standard Edition 2データベースは、別の構成済ノードに再配置できます。
- Standard Edition高可用性データベースへのノードの追加

既存のStandard Edition高可用性構成に新しいノードを追加することで、Standard Edition 2データベースのフェイルオーバー機能が強化されます。

- Standard Edition高可用性データベースからの構成済ノードの削除
  srvctlコマンドを使用して、Standard Edition高可用性データベース用に構成されたノードのリストからノードを削除します。
- Standard Edition高可用性データベースの起動と停止
  srvctlコマンドを使用して、Standard Edition高可用性用に構成されたOracle Databaseを起動または停止します。
- Oracle DatabaseのStandard Edition高可用性の非アクティブ化
  単一インスタンスOracle DatabaseのStandard Edition高可用性を非アクティブ化すると、そのデータベースは高可用性フェイルオーバー構成に含まれなくなります。

関連トピック

- 『Oracle Databaseインストレーション・ガイド for Linux』
- 『Oracle Real Application Clusters管理およびデプロイメント・ガイド』

親トピック: Oracle Databaseの作成および構成

2.9.1 Standard Edition高可用性について

Oracle Database 19c以降では、高可用性モードでOracle Database Standard Edition 2をインストールできます。

Standard Edition高可用性では、Oracle Clusterwareを使用し、単一インスタンスのStandard Edition Oracle Databases用にクラスタベースのフェイルオーバーを提供します。

Oracle Standard Edition高可用性の利点は、Oracle Clusterware、Oracle Automatic Storage Management (Oracle ASM)およびOracle ASM Cluster File System (Oracle ACFS)など、Oracle Grid Infrastructureにすでに含まれているクラスタ機能およびストレージ・ソリューションです。

データベース・ファイル用および非構造化データ用のOracle ASMやOracle ACFSなどの、統合された、共有の、同時にマウントされた記憶域を使用することにより、Oracle Grid Infrastructureではフェイルオーバーのノード上でOracle Databaseを再起動でき、これは、フェイルオーバーおよび再マウントされたボリュームやファイル・システムに依存するクラスタ・ソリューションよりもはるかに高速です。

Oracle Database 19cリリース更新(19.7)以降、Standard Edition高可用性は、Linux x86-64、Oracle Solaris on SPARC (64ビット)およびMicrosoft Windowsでサポートされています。

Oracle Database 19cリリース更新(19.13)以降、Standard Edition高可用性は、IBM AIX on POWER Systems (64ビット)およびHP-UX Itaniumでサポートされています。

ノート:

この項では、Standard Edition Oracle Databases 19c にクラスタベースのデータベース・フェイルオーバーを提供する Standard Edition 高可用性について詳しく説明します。Oracle Database の高可用性オプションの詳細は、『Oracle Clusterware 管理およびデプロイメント・ガイド』を参照してください。

親トピック: Oracle DatabaseのStandard Edition高可用性の管理

2.9.2 Oracle DatabaseによるStandard Edition高可用性を使用するための要件

Standard Edition高可用性を使用する場合は、次に示す構成の要件に従ってOracle Database Standard Edition 2をデプロイします。

- データベースはスタンドアロン・クラスタ用のOracle Grid Infrastructureを実行しているクラスタ内に作成します。そのデータベース・ファイルは、Oracle Automatic Storage Management (Oracle ASM)またはOracle Automatic Storage Management Cluster File System (Oracle ACFS)に配置します。
- Database Configuration Assistantを使用しているときには、Standard Edition高可用性用に構成するOracle Database Standard Edition 2データベースの作成時にリスナーを作成しないでください。
- データベースは、リモート・リスナーとしての単一クライアント・アクセス名(SCAN)リスナーと、ローカル・リスナーとしてのノード・リスナーに登録します。
- データベース・サービスを作成します。このサービスは、デフォルトのデータベース・サービスのかわりに、アプリケーション・クライアントまたはデータベース・クライアントをデータベースに接続するときに使用します。
- サーバー・パラメータ・ファイル(`spfile`)とパスワード・ファイルが、Oracle ASMまたはOracle ACFSに配置されていることを確認します。`spfile`とパスワード・ファイルがデータベースの作成時または構成時にローカル・ファイル・システムに配置されていた場合は、それらのファイルをOracle ASMまたはOracle ACFSに移動します。

満たしておく必要のある追加の要件については、データベースのインストール・ドキュメントを参照してください。

関連トピック

- Oracle Databaseインストレーション・ガイドfor Linux

親トピック: Oracle DatabaseのStandard Edition高可用性の管理

2.9.3 Oracle DatabaseのStandard Edition高可用性の有効化

Standard Edition高可用性を有効化することで、Oracle Database Standard Edition 2データベースにクラスタ・ベースのフェイルオーバーを実現します。

ノート:

このトピックに示すステップは、Standard Edition 高可用性を構成するための Oracle Database ソフトウェア・バイナリをインストールして(『Oracle Database インストレーション・ガイド for Linux』を参照)、データベースを作成した後で実行する必要があります。1 つのクラスタ・ノードで実行する既存の Standard Edition 2 データベースがあるときに、このデータベースの Standard Edition 高可用性を有効化する場合は、その構成にノードを追加する必要があります。

前提条件

- 初期化パラメータ`local_listener`が設定されていないことを確認します。これにより、ノードのリスナーが自動的に使用されるようにして、データベース接続がノードの仮想IPアドレスを通じて確立されるようにします。

  次のコマンドを使用して、現在のリスナー構成を表示します。

  ```
  SQL> SHOW PARAMETER LOCAL_LISTENER;
  ```

  このコマンドの出力にローカル・リスナー名が表示された場合は、次のコマンドを使用してローカル・リスナーをリセットします。

  ```
  SQL> ALTER SYSTEM RESET LOCAL_LISTENER SCOPE = BOTH;
  ```

リスナー構成の変更が反映されるようにするために、データベースを再起動する必要があります。ただし、Standard Edition高可用性構成の検証の一環として、データベースが別のノードに再配置された場合は、データベースの再起動は不要です。

- Oracle ASM Cluster File System (Oracle ACFS)にデータベース・ファイルを格納する場合は、Oracle ClusterwareにOracle ACFSファイル・システムを登録し、対応するOracle ACFSリソースに対するデータベース・リソースの依存性を構成するために`srvctl`コマンドを使用する必要があります。たとえば:

```
$ srvctl modify database -db se2cdb2 -acfspath /u01/app/oradata
```

Oracle Database Standard Edition 2データベースのStandard Edition高可用性を有効化するには:

1. データベースの作成に`CREATE DATABASE`コマンドを使用した場合は、そのデータベースをOracle Clusterwareに登録します。

   `CREATE DATABASE`コマンドを使用して作成したデータベースは、Oracle Clusterwareに自動的には登録されません。`srvctl add database`コマンドを使用して、データベースを登録します。

   たとえば、次のコマンドでは、`se2cdb`という一意の名前が付いた単一インスタンス・データベースをノード`node3`および`node5`に登録します。

```
$ srvctl add database -db se2cdb -oraclehome $ORACLE_HOME
-dbtype SINGLE -spfile +DATA/SE2CDB1/PARAMETERFILE/spfile.276.1030845691
-node node3,node5
```

2. データベースがすでにOracle Clusterwareに登録されている場合は、`srvctl modify database`コマンドを使用してStandard Edition高可用性を有効化します。

   たとえば、次のコマンドでは、一意のデータベース名が`se2cdb`のStandard Edition 2データベースでStandard Edition高可用性を有効化します。

```
$ srvctl modify database -db se2cdb -node node3,node5
```

ノート:

固定単一インスタンス・データベースを作成するための`-fixed`オプションは、Standard Edition 高可用性ではサポートされていません。

データベースのStandard Edition高可用性を有効化すると、次のようになります。

- データベース・インスタンスを実行しているノードの計画外停止が発生すると、インスタンスは構成済ノードのリストにある使用可能なノードで再起動されます。
- データベース・インスタンスの計画外中断が発生すると、現在のノードでインスタンスの再起動が試行されます。再起動に失敗した場合は、構成済ノードのリストにある使用可能なノードへのフェイルオーバーが開始されます。
- データベース・インスタンスを実行しているノードのパブリック・ネットワークへの接続が完全に失われると、そのインスタンスは構成済ノードのリストにある使用可能なノードに再配置されます。

ノート:

ノード・リスト内のノードの順序によって、データベースを起動するノードが決まります。Oracle Clusterware は、ノード・リストの最初のノードでデータベースを起動しようとします。最初のノードが使用できない場合は、ノード・リスト

の次のノードに移動します。

また、Oracle Clusterwareは、現在のノードで障害が発生した場合、ノード・リスト内のノードの順序を使用してフェイルオーバー・ターゲットを決定します。フェイルオーバー・ターゲットが考慮されるのは、リスト内の最初のノードから始まって適切な候補が見つかるまでで、クラスタ内の他の状況によってこの順序でフェイルオーバー・ノードを決定できない場合を除きます。

Standard Edition高可用性の構成の検証

Standard Edition高可用性の構成(特に現在の構成済ノードのリスト)を検証する場合は、`srvctl config database`コマンドを使用できます。たとえば:

```
$ srvctl config database -db se2cdb
...
Type: SINGLE
...
Configured nodes:
        node3, node5
```

この出力では、データベースのタイプが単一でありながら、構成済ノードが複数存在していることに注目してください。これは、Standard Edition高可用性が有効になっていることを示しています。

さらに検証を進めるには、データベースを別の構成済ノードに再配置します。

関連トピック

- Oracle Databaseインストレーション・ガイドfor Linux
- Standard Edition高可用性データベースへのノードの追加

親トピック: Oracle DatabaseのStandard Edition高可用性の管理

2.9.4 Standard Edition高可用性データベースの別のノードへの再配置

計画停止を管理するために、Standard Edition高可用性を使用するOracle Database Standard Edition 2データベースは、別の構成済ノードに再配置できます。

データベースの再配置先ノードは、このデータベースの構成済ノードのリストに含まれている必要があります。

アクティブなOracle Database Standard Edition 2データベースを現在のノードから別の構成済ノードに再配置するには:

- `srvctl relocate`コマンドを使用します。

  このコマンドでは、オフライン再配置を実行します。既存のノードのデータベースを停止してから、新しいノードで起動します。

たとえば、次のコマンドでは、Standard Edition高可用性を使用しているse2cdbという名前のStandard Edition 2データベースをノードnode5に再配置します。

```
$ srvctl relocate database -db se2cdb -node node5
```

ノート:

`srvctl relocate database`コマンドのオプション`-abort`と`-revert`は、Standard Edition高可用性ではサポートされていません。

2.9.5 Standard Edition高可用性データベースへのノードの追加

既存のStandard Edition高可用性構成に新しいノードを追加することで、Standard Edition 2データベースのフェイルオーバー機能が強化されます。

ノードの追加は、特定のシナリオで必要になることがあります。たとえば、Standard Edition高可用性データベース用に2つのノードを構成したとします。その後、クラスタに新しいノードが追加され、その新しいノードをデータベース構成に組み込もうとしている場合です。これは、そのノードをStandard Edition高可用性構成に追加することで実現できます。別のシナリオとして、既存のStandard Edition 2データベースに対してStandard Edition高可用性を有効にしようとしている場合があります。

Standard Edition高可用性データベースにノードを追加するには:

1. 新しいノードにOracle DatabaseのOracleホームを拡張します。このステップは、使用している記憶域によって異なります。

   ローカル(非共有) Oracleホームの場合:

   a. 最初のクラスタ・ノード(Standard Edition高可用性を構成したノード)に、Oracleインストール所有者ユーザー・アカウント(`oracle`)としてログインします。

   b. `ORACLE_HOME/addnode`ディレクトリに移動して、`addnode.sh`スクリプトを実行します。

      たとえば、Oracleホームを`node3`に拡張するには、次のようにします。

      ```
      $ ./addnode.sh -silent "CLUSTER_NEW_NODES={node3}"
      ```

   Oracle ACFSを使用している共有Oracleホームの場合:

   c. `root`として`Grid_home/bin`ディレクトリから次のコマンドを実行して、新しいノードでOracle ACFSリソースを起動します。

      ```
      # srvctl start filesystem -device volume_device [-node node_name]
      ```

      ノート:

      Oracle ホームが格納されている Oracle ACFS レジストリ・リソースおよび Oracle ACFS ファイル・システム・リソースを含む Oracle ACFS リソースが、新しく追加されたノードでオンラインであることを確認します。

   d. 最初のクラスタ・ノード(Standard Edition高可用性を構成したノード)に、Oracleインストール所有者ユーザー・アカウント(`oracle`)としてログインします。

   e. 追加しようとしているノードに、最初のクラスタ・ノードのOracleホームをアタッチします。

      ```
      $ $ORACLE_HOME/addnode/addnode.sh -silent CLUSTER_NEW_NODES=new_node_name
      ```

2. 新しいノード(追加することになるノード)で、`root`ユーザーとして`ORACLE_HOME/root.sh`スクリプトを実行します。

   ```
   # /u01/app/oracle/product/19.0.0/dbhome_1/root.sh
   ```

3. いずれかの構成済ノード(追加するノードを含む)で、`oracle`ユーザーとして`srvctl modify database`コマンドを使用することで、新しいノード追加します。

-node引数には、既存の構成済ノードと追加する新しいノードをリストする必要があります。

たとえば、次のコマンドでは、ノードnode3をse2cdbという一意の名前のデータベースに追加します。既存の構成済ノードは、node1とnode5です。

```
$ srvctl modify database -db se2cdb -node node1,node3,node5
```

新しい構成を必要に応じて確認してください(「Oracle DatabaseのStandard Edition高可用性の有効化」を参照)。

関連トピック

- Oracle DatabaseのStandard Edition高可用性の有効化

親トピック: Oracle DatabaseのStandard Edition高可用性の管理

2.9.6 Standard Edition高可用性データベースからの構成済ノードの削除

srvctlコマンドを使用して、Standard Edition高可用性データベース用に構成されたノードのリストからノードを削除します。

Oracle Databaseホーム・ディレクトリに移動します。Linuxの場合は、Oracleインストール所有者ユーザー・アカウント(oracle)として、データベース・ホスト・コンピュータにログインします。Windowsの場合は、管理者としてデータベース・ホスト・コンピュータにログインします。

Standard Edition高可用性を使用するデータベースから構成済ノードを削除するには:

1. srvctl config databaseコマンドを使用して、既存の構成済ノードをリストします。
2. 削除しようとしているノードでデータベースが実行されている場合は、srvctl relocate databaseコマンドを使用して、そのデータベースを別の構成済ノードに再配置します。
3. -node引数を指定したsrvctl modify databaseコマンドを使用して、ノードを削除します。
   -node引数には、削除が必要なノード以外のすべての構成済ノードをリストする必要があります。

例2-1 Standard Edition高可用性データベースからの構成済ノードの削除

この例の前提として、sec2cdbという一意の名前のデータベースがStandard Edition高可用性を使用していて、構成済ノードのnode1、node2およびnode3があるとします。現在、データベースはnode3で実行されています。このデータベースの構成済ノードのリストからnode2を削除するには、Oracle Databaseホームをインストールしたユーザーとしてログインし、次のコマンドを実行します。

```
$ srvctl modify database -db sec2cdb -node node1,node3
```

関連トピック

- Standard Edition高可用性データベースの別のノードへの再配置

親トピック: Oracle DatabaseのStandard Edition高可用性の管理

2.9.7 Standard Edition高可用性データベースの起動と停止

srvctlコマンドを使用して、Standard Edition高可用性用に構成されたOracle Databaseを起動または停止します。

Oracle Databaseホーム・ディレクトリに移動します。Linuxの場合は、Oracleインストール所有者ユーザー・アカウント(oracle)として、データベース・ホスト・コンピュータにログインします。Windowsの場合は、管理者としてデータベース・ホスト・コンピュータにログインします。

Standard Edition高可用性データベースを起動するには:

- `srvctl start database`コマンドを使用します。

  オプションとして、データベースを起動する必要のあるノードを指定する`-node`引数を含めます。

Standard Edition高可用性データベースを停止するには:

- `srvctl stop database`コマンドを使用します

例2-2 指定したノードでのStandard Edition高可用性データベースの起動

この例では、一意の名前が`se2cdb`というデータベースを`node3`という名前のノードで起動します。

```
$ srvctl start database -db sec2cdb -node node3
```

例2-3 Standard Edition高可用性データベースの停止

この例では、Standard Edition高可用性を使用するように構成されたデータベース・インスタンスを停止します。このデータベースの一意の名前は`sec2cdb`です。

```
$ srvctl stop database -db sec2cdb
```

関連トピック

- Oracle Real Application Clusters管理およびデプロイメント・ガイド

親トピック: Oracle DatabaseのStandard Edition高可用性の管理

2.9.8 Oracle DatabaseのStandard Edition高可用性の非アクティブ化

単一インスタンスOracle DatabaseのStandard Edition高可用性を非アクティブ化すると、そのデータベースは高可用性フェイルオーバー構成に含まれなくなります。

Oracle DatabaseのStandard Edition高可用性を非アクティブ化するには:

- `srvctl modify`コマンドを使用して、`-node`引数に1つのノードのみを含めます。

例2-4 Oracle DatabaseのStandard Edition高可用性の使用の非アクティブ化

この例では、`se2cdb`という一意の名前が付いたデータベースのStandard Edition高可用性の使用を非アクティブ化して、このデータベース用に`node1`という1つのノードのみを構成します。

```
srvctl modify database -db se2cdb -node node1
```

以前のすべての構成済ノードは削除され、このデータベースはOracle Clusterwareに登録された単一インスタンス・データベースになります。

親トピック: Oracle DatabaseのStandard Edition高可用性の管理

2.10 データベースを作成した後の考慮点

データベースの作成後、インスタンスは実行されたままになり、データベースはオープンされて通常のデータベースの用途に使用できます。データベースの作成後に特定のアクションを実行することが必要な場合があります。

- データベース・セキュリティ
  デフォルトのOracle Database機能を使用すると、Oracleデータベースのいくつかの領域でセキュリティを構成できます。
- 透過的データ暗号化

透過的データ暗号化は、個々のデータベース列をデータファイルに格納する前に暗号化するか、表領域全体を暗号化する機能です。ユーザーが、オペレーティング・システムのツールを使用してデータファイルの内容を直接参照することによって、データベース・アクセス制御メカニズムを迂回しようとした場合でも、透過的データ暗号化によって、このようなユーザーが機密情報を参照できないようにします。

- 安全性の高い外部パスワード・ストア
  認証および資格証明の署名をネットワークに公開しないように、クライアント側のOracleウォレットの使用を検討してください。
- トランザクション・ガードおよびアプリケーション・コンティニュイティ
  トランザクション・ガードは、論理トランザクションIDを使用して、リカバリ可能なエラーの後にクライアント・アプリケーションが重複するトランザクションを送信する可能性を排除します。アプリケーション・コンティニュイティは、データベース・セッションを使用不可にするリカバリ可能なエラーの後、中断せずに迅速な方法でデータベースに対するリクエストの再実行を可能にします。
- データベースでのファイル・システム・サーバーのサポート
  Oracleデータベースは、ファイル・システム・オブジェクトを格納し、任意のNFSクライアントからそれらにアクセスするように構成できます。データベースは、ファイルとそのメタデータの両方を格納します。データベースは、オペレーティング・システム(OS)カーネルのNFSデーモン・プロセスからファイル・システム・リクエストに応答します。
- Oracle Databaseサンプル・スキーマ
  Oracle Databaseには、Oracle Database機能の理解に役立つサンプル・スキーマが含まれています。一部のOracle Databaseのドキュメントおよびトレーニング・マテリアルでは、サンプル・スキーマが例で使用されます。

親トピック: Oracle Databaseの作成および構成

2.10.1 データベース・セキュリティ

デフォルトのOracle Database機能を使用すると、Oracleデータベースのいくつかの領域でセキュリティを構成できます。

データベースのセキュリティを構成できる領域をいくつか次に示します。

- ユーザー・アカウント: 作成したユーザー・アカウントは様々な方法で保護できます。サイトのパスワード・ポリシーを強化するために、パスワード・プロファイルを作成することもできます。
- 認証方式: Oracle Databaseには、ユーザーおよびデータベース管理者用の認証を構成する方法がいくつかあります。たとえば、ユーザーは、データベース・レベル、オペレーティング・システムおよびネットワークで認証できます。
- 権限とロール: 権限とロールを使用すると、データに対するユーザー・アクセスを制限できます。

ノート:

- 新しく作成したデータベースには、データベースの管理にとって重要なユーザー・アカウントが少なくとも 3 つ(SYS、SYSTEM および SYSMAN)あります。認可されたユーザーのみが使用する追加の管理アカウントが提供されています。
- データベースへの不当なアクセスを防ぎ、データベースの整合性を保護するには、データベースの作成時に、ユーザーSYS に対して新しいパスワードを指定することが重要です。
- Oracle Database 19c 以上では、SYS およびサンプル・スキーマを除き、Oracle Database で提供されるユーザー・アカウントのほとんどはスキーマ専用アカウント(つまり、これらのアカウントはパスワードなしで作

成される)です。これらのアカウントには、認証する必要が生じたときにいつでもパスワードを割り当てることができますが、セキュリティを強化するために、認証が必要なくなった場合にはこれらのアカウントをスキーマ専用アカウントに戻すことをお薦めします。

アカウントのステータスを確認するには、DBA_USERS データ・ディクショナリ・ビューの ACCOUNT_STATUS 列を問い合せます。アカウントがスキーマ専用の場合、ステータスは NONE です。

関連項目:

- ユーザーSYSおよびSYSTEMの詳細は、管理ユーザー・アカウントを参照してください。
- 新規の各Oracle Databaseインストールで作成される事前定義済ユーザー・アカウントをすべて示すリストは、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- 新しいユーザーの追加方法とパスワードの変更方法を学習するには、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- データベース・ユーザー・アカウントのロック解除に使用するALTER USER文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。
- Oracle Identity Managementの詳細は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください
- データベースを構成するためのセキュリティ・ガイドラインについては、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: データベースを作成した後の考慮点

2.10.2 透過的データ暗号化

透過的データ暗号化は、個々のデータベース列をデータファイルに格納する前に暗号化するか、表領域全体を暗号化する機能です。ユーザーが、オペレーティング・システムのツールを使用してデータファイルの内容を直接参照することによって、データベース・アクセス制御メカニズムを迂回しようとした場合でも、透過的データ暗号化によって、このようなユーザーが機密情報を参照できないようにします。

CREATE TABLE権限のあるユーザーは、暗号化対象の表の列を1つ以上選択できます。データはデータファイルで暗号化されます。適切な権限のあるデータベース・ユーザーは、データを暗号化されていない形式で表示できます。透過的データ暗号化の有効化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

関連項目:

- 「機密データを格納する列の暗号化」
- 「暗号化された表領域」

親トピック: データベースを作成した後の考慮点

2.10.3 安全性の高い外部パスワード・ストア

認証および資格証明の署名をネットワークに公開しないように、クライアント側のOracleウォレットの使用を検討してください。

アプリケーションでデータベースに接続するためにパスワード資格証明が使用される大規模なデプロイメントでは、この資格証明をクライアント側のOracleウォレットに格納できます。Oracle Walletは、資格証明の認証および署名の格納に使用されるセキュアなソフトウェア・コンテナです。

クライアント側のOracleウォレットにデータベース・パスワード資格証明を格納することで、ユーザー名とパスワードをアプリケーション・コード、バッチ・ジョブまたはスクリプトに埋め込む必要がなくなります。クライアント側の記憶域により、スクリプトやアプリケーション・コードでパスワードをさらすリスクが軽減されます。また、ユーザー名とパスワードが変更されるたびにコードを変更する必要がないため、メンテナンスも簡素化されます。また、アプリケーション・コードを変更する必要がないため、これらのユーザー・アカウントのパスワード管理ポリシーをさらに簡単に規定できるようになります。

外部パスワード・ストアを使用するようにクライアントを構成すると、アプリケーションでは、次の構文を使用してパスワード認証を使用しているデータベースに接続できます。

```
CONNECT /@database_alias
```

このCONNECTコマンドでは、データベース・ログイン資格証明を指定する必要はありません。かわりに、データベース・ログイン資格証明はクライアントのウォレットで検索されます。

関連項目:

- Oracle Databaseセキュリティ・ガイド
- Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド

親トピック: データベースを作成した後の考慮点

2.10.4 トランザクション・ガードおよびアプリケーション・コンティニュイティ

トランザクション・ガードは、リカバリ可能なエラーの後、クライアント・アプリケーションが重複トランザクションを送信する可能性を防止するために論理トランザクションIDを使用します。アプリケーション・コンティニュイティは、データベース・セッションを使用不可にするリカバリ可能なエラーの後、中断せずに迅速な方法でデータベースに対するリクエストの再実行を可能にします。

トランザクション・ガードは、使用できなくなるデータベース・セッションで最後にオープンされたトランザクションについて既知の結果を提供するために、アプリケーション開発者が使用できる信頼性の高いプロトコルおよびAPIです。停止後、データベースからクライアントに送信されたコミット・メッセージは非永続です。アプリケーション(クライアント)とOracle Database (サーバー)の間の接続が切断されると、クライアントは通信が失敗したことを示すエラー・メッセージを受信します。このエラー・メッセージでは、コミット操作またはプロシージャ・コールの成功または失敗に関してはクライアントに通知しません。

トランザクション・ガードでは、アプリケーションの観点からトランザクションを識別するグローバル一意識別子である、論理トランザクション識別子(LTXID)と呼ばれる概念を使用します。リカバリ可能な停止が発生すると、アプリケーションではLTXIDを使用してトランザクションの結果を判別します。あいまいな通信エラーのかわりに、この結果をクライアントに返すことができます。ユーザーはトランザクションを再実行するかどうかを決定できます。また、状態が適切な場合にトランザクションが再実行されるように、アプリケーションをコーディングすることもできます。

アプリケーション・コンティニュイティは、リカバリ可能な停止(計画外停止と計画停止の両方)に続いて処理中のデータベース・セッションをリカバリすることで、エンド・ユーザーおよびアプリケーションから停止をマスキングします。リプレイが成功した後、アプリケーションは元のデータベース・セッションが停止した位置から新しいセッションを使用して続行できます。アプリケーション・コンティニュイティはこのリカバリを実行するため、アプリケーションにとっては、その停止が遅延された実行のように見えます。

アプリケーション・コンティニュイティはサービス・レベルで有効化され、リカバリ可能な停止に対して起動されます。通常、これらの

停止は基礎になるソフトウェア、フォアグラウンド、ハードウェア、通信、ネットワークまたは記憶域レイヤーに関連します。アプリケーション・コンティニュイティは、問合せ、ALTER SESSION文、JavaとOCIのAPI、PL/SQL、DDLおよび障害の前にコミットされなかった最後のトランザクションをサポートします。アプリケーション・コンティニュイティは、トランザクション・ガードを使用して、最後の処理中のトランザクションがコミットされたかどうか、および最後のユーザー・コールが完了したかどうかを判別します。

関連項目:

- トランザクション・ガードおよびアプリケーション・コンティニュイティの概念の詳細は、『Oracle Database概要』を参照してください。
- トランザクション・ガードおよびアプリケーション・コンティニュイティの詳細は、『Oracle Database開発ガイド』を参照してください。

親トピック: データベースを作成した後の考慮点

2.10.5 データベースでのファイル・システム・サーバーのサポート

Oracleデータベースは、ファイル・システム・オブジェクトを格納し、任意のNFSクライアントからそれらにアクセスするように構成できます。データベースは、ファイルとそのメタデータの両方を格納します。データベースは、オペレーティング・システム(OS)カーネルのNFSデーモン・プロセスからファイル・システム・リクエストに応答します。

Oracle File System (OFS)サーバーをデータベースに構成し、ファイル・システムを作成する場合は、電子メール、ビデオ、オーディオ・ファイル、クレジット・カード請求書、ドキュメント、写真イメージなどの非構造化データをデータベース内に格納できます。SQLを使用せずに、これらの非構造化オブジェクトを操作および管理できます。かわりに、NFSサポートのためのオペレーティング・システムのユーティリティを使用できます。

データベースでNFSアクセスを有効にするには、OFS_THREADS初期化パラメータを設定して、NFSリクエストを処理するための十分な数のOFSスレッドを構成します。最初のファイル・システムがデータベースでマウントされている場合、OFS_THREADS初期化パラメータは、作成するOFSスレッド数を制御します。OFS_THREADSパラメータで指定された数のスレッドが、データベース・インスタンスに対して1回のみ作成され、後続のファイル・システムではスレッドが追加作成されません。OFS_THREADS初期化パラメータのデフォルト値は4です。データベースの起動時に、OFSDバックグラウンド・プロセスは、データベース・サーバーによって起動される唯一のOFSプロセスです。

DBMS_FSパッケージを使用し、指定したデータベース・オブジェクトを使用してデータベースにファイル・システムを作成できます。このパッケージを使用して、指定したファイル・システムをマウントおよびアンマウントすることもできます。

関連項目:

- Oracle File System (OFS)の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください
- DBMS_FSパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

親トピック: データベースを作成した後の考慮点

2.10.6 Oracle Databaseサンプル・スキーマ

Oracle Databaseには、Oracle Database機能の理解に役立つサンプル・スキーマが含まれています。一部のOracle

Databaseのドキュメントおよびトレーニング・マテリアルでは、サンプル・スキーマが例で使用されます。

スキーマとインストール手順の詳細は、『Oracle Databaseサンプル・スキーマ』を参照してください。

ノート:

本番データベースにはサンプル・スキーマをインストールしないことをお薦めします。

親トピック: データベースを作成した後の考慮点

2.11 データベースのクローニング

この項では、Oracleデータベースの様々なクローニング方法について説明します。

- 非マルチテナント環境でのCloneDBを使用したデータベースのクローニング
  CloneDBを使用すると、複数の異なる場所にデータ・ファイルをコピーすることなく、非マルチテナント環境でデータベースを複数回クローニングできます。かわりに、CloneDBはcopy-on-writeテクノロジを使用します。そのため、ディスク上に追加の記憶域が必要になるのは、変更されているブロックのみになります。
- マルチテナント環境でのデータベースのクローニング
  マルチテナント環境でデータベースをクローニングできます。
- Oracle Automatic Storage Management (Oracle ASM)を使用したデータベースのクローニング
  Oracle Automatic Storage Management (Oracle ASM)では、マルチテナント・コンテナ・データベース(CDB)でのプラガブル・データベース(PDB)のクローニングがサポートされています。Oracle ASMでは、非CDBのクローニングはサポートされていません。

親トピック: Oracle Databaseの作成および構成

2.11.1 非マルチテナント環境でのCloneDBを使用したデータベースのクローニング

CloneDBを使用すると、複数の異なる場所にデータ・ファイルをコピーすることなく、非マルチテナント環境でデータベースを複数回クローニングできます。かわりに、CloneDBはcopy-on-writeテクノロジを使用します。そのため、ディスク上に追加の記憶域が必要になるのは、変更されているブロックのみになります。

- CloneDBを使用したデータベースのクローニングについて
  多くの場合、テスト目的またはその他の目的で本番データベースをクローニングすることが必要となります。
- CloneDBを使用したデータベースのクローニング
  CloneDBでデータベースをクローニングできます。
- CloneDBを使用したデータベースのクローニング後
  CloneDBデータベースを作成した後、本番データベースを使用するほとんどすべての方法でそのデータベースを使用できます。最初は、CloneDBデータベースは各データファイルに対して最小限の記憶域を使用します。CloneDBデータベースの行に対して変更を加えると、記憶域がオンデマンドで割り当てられます。

親トピック: データベースのクローニング

2.11.1.1 CloneDBを使用したデータベースのクローニングについて

多くの場合、テスト目的またはその他の目的で本番データベースをクローニングすることが必要となります。

本番データベースをクローニングする一般的な理由は、次のとおりです。

- データベースを使用する新規アプリケーションのデプロイ、または既存のアプリケーションの更新

- データベースを実行するシステムでの、オペレーティング・システムのアップグレード計画
- データベース・インストール用の新しい記憶域
- レポート
- 古いデータの分析

新しいアプリケーションをデプロイする前、オペレーティング・システムのアップグレードを実行する前、または新しい記憶域を使用する前に、新しい状況でデータベースが適切に動作することを確認するために徹底的にテストすることが必要です。クローニングは、本番データファイルのコピーを1つ以上のテスト環境に作成することで実行できますが、通常、これらのコピーを割り当てて管理するために、大量の記憶域が必要となります。

CloneDBを使用すると、データファイルを複数の異なる場所にコピーしなくても、データベースを複数回クローニングできます。Oracle Databaseでは、かわりにcopy-on-writeテクノロジを使用してCloneDBデータベースにファイルを作成するため、ディスク上に追加の記憶域が必要になるのは、CloneDBデータベースで変更されたブロックのみとなります。

この方法でデータベースをクローニングすると、次の利点があります。

- テスト目的で必要になる記憶域の量が削減されます。
- 様々な目的のために複数のデータベース・クローンを迅速に作成できます。

CloneDBデータベースは、データベース・バックアップのデータファイルを使用します。バックアップ・データファイルを使用すると、CloneDBインスタンスによる本番データファイルへのアクセスが行われず、CPUやI/Oリソースなどの本番データベースのリソースに対するCloneDBインスタンスによる競合が発生しません。

ノート:

- この項で説明する CloneDB を使用したデータベースのクローニングの手順は、マルチテナント環境のデータベースには適用できません。
- CloneDB 機能は、パフォーマンス・テスト向けではありません。

関連項目:

マルチテナント環境でのデータベースのクローニングの詳細は、「マルチテナント環境でのデータベースのクローニング」を参照してください

親トピック: 非マルチテナント環境でのCloneDBを使用したデータベースのクローニング

2.11.1.2 CloneDBを使用したデータベースのクローニング

CloneDBでデータベースをクローニングできます。

データベースをクローニングするには、次の前提条件を満たしている必要があります。

- 各CloneDBデータベースはDirect NFSクライアントを使用する必要があり、本番データベースのバックアップはNFSボリューム上に存在する必要があります。

  Direct NFSクライアントにより、Oracle Databaseは、オペレーティング・システム・カーネルのNFSクライアントを使用するかわりに、ネットワーク接続記憶域(NAS)デバイスに直接アクセスできます。このCloneDBデータベース機能は、

Direct NFSクライアントをサポートするプラットフォームで使用可能です。

Direct NFSクライアントの詳細は、使用しているオペレーティング・システムの『Oracle Grid Infrastructureインストレーション・ガイド』を参照してください。

- CloneDBデータベースの変更されたブロックを追跡するには、少なくとも2MBの追加システム・グローバル領域(SGA)メモリーが必要です。

  「メモリーの管理」を参照してください。

- データベース・バックアップ用および各CloneDBデータベースの変更されたブロック用の記憶域が必要となります。

  データベース・バックアップに必要な記憶域は、バックアップの実行に使用する方法によって異なります。単一の全体RMANバックアップでは、最も多くの記憶域が必要となります。ストレージ・アプライアンスの機能を使用して実行される記憶域スナップショットは、そのストレージ・アプライアンスの要件に従います。単一のバックアップで複数のCloneDBデータベースをサポートできます。

  各CloneDBデータベースで必要になる記憶域の量は、そのデータベースの書込みアクティビティによって異なります。変更される各ブロックには、記憶域の使用可能な1つのブロックが必要となります。したがって、必要になる合計記憶域は、一定期間にCloneDBデータベースで変更されるブロックの数によって異なります。

この項では、1つのCloneDBデータベースを作成するためのステップについて説明し、これらのサンプル・データベースおよびディレクトリを使用します。

- 本番データベースPROD1のOracleホームは/u01/prod1/oracleです。
- データベース・バックアップのファイルは/u02/oracle/backup/prod1にあります。
- CloneDBデータベースCLONE1のOracleホームは/u03/clone1/oracleです。

CloneDBを使用してデータベースをクローニングするには:

1. 本番データベースのバックアップを作成します。次のバックアップ・オプションがあります。

   - オンライン・バックアップ

     オンライン・バックアップを実行する場合は、本番データベースがARCHIVELOGモードになっており、必要なすべてのアーカイブREDOログ・ファイルが保存され、CloneDBデータベース環境にアクセスできることを確認します。

   - 全体オフライン・バックアップ

     全体オフライン・バックアップを実行する場合は、バックアップ・ファイルがCloneDBデータベース環境にアクセスできることを確認します。

   - データベース・ファイルをコピーするバックアップ

     BACKUP AS COPYをRMANで指定すると、RMANは各ファイルを、ディスクに作成されたデータベース・ファイルのビットごとのコピーであるイメージ・コピーとしてコピーします。イメージ・コピーは、LinuxのcpやWindowsのCOPYなどのオペレーティング・システム・コマンドで作成したコピーと同じですが、RMANのリポジトリに記録されるため、RMANで使用できます。RMANを使用すると、データベースがオープンしている間にイメージ・コピーを作成できます。コピーされたデータベース・ファイルがCloneDBデータベース環境にアクセスできることを確認します。

   データベースのバックアップの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

2. テキスト形式の初期化パラメータ・ファイル(PFILE)がない場合は、作成します。

   サーバー・パラメータ・ファイル(SPFILE)を使用する場合は、本番データベース上で次の文を実行して、PFILEを作成します。

   ```
   CREATE PFILE FROM SPFILE;
   ```

3. 本番データベースをクローニングするためのSQLスクリプトを作成します。

   後のステップで、CloneDBデータベースを作成するために、1つ以上のSQLスクリプトを使用します。SQLスクリプトを作成するには、オラクル社が提供するclonedb.plと呼ばれるPerlスクリプトを使用するか、またはSQLスクリプトを手動で作成できます。

   clonedb.pl Perlスクリプトを使用するには、次のステップを完了します:

   - オペレーティング・システムのプロンプトで、次の環境変数を設定します。

     `MASTER_COPY_DIR` - ステップ1で作成したバックアップが含まれるディレクトリを指定します。このディレクトリには、本番データベースのデータファイルのバックアップのみが含まれていることを確認します。

     `CLONE_FILE_CREATE_DEST` - データファイル、ログ・ファイル、制御ファイルなど、CloneDBデータベース・ファイルが作成されるディレクトリを指定します。

     `CLONEDB_NAME` - CloneDBデータベースの名前を指定します。

     `S7000_TARGET` - バックアップ用のファイル・システムを提供するNFSホストおよびCloneDBデータベースがSun Storage 7000である場合は、ホストの名前を指定します。それ以外の場合は、この環境変数を設定しないでください。この環境変数は、記憶域スナップショットを使用してクローニングを実行する必要がある場合にのみ設定します。この変数を設定せずに、Direct NFSクライアントにS7000ストレージ・アレイを使用できます。

   - clonedb.pl Perlスクリプトを実行します。

     このスクリプトは、$ORACLE_HOME/rdbms/installディレクトリにあり、次の構文が含まれます。

     ```
     $ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/install/clonedb.pl
     prod_db_pfile [sql_script1] [sql_script2]
     ```

     次のオプションを指定します。

     `prod_db_pfile` - 本番データベースのPFILEのフルパスを指定します。

     `sql_script1` - clonedb.plによって生成された最初のSQLスクリプトの名前を指定します。デフォルトはcrtdb.sqlです。

     `sql_script2` - clonedb.plによって生成された2番目のSQLスクリプトの名前を指定します。デフォルトはdbren.sqlです。

     clonedb.plスクリプトは、本番データベースのPFILEをCloneDBデータベースのディレクトリにコピーします。また、CloneDBデータベースの作成に使用する2つのSQLスクリプトも作成します。

   - clonedb.pl Perlスクリプトにより生成された2つのSQLスクリプトを確認し、必要に応じて変更します。

   - CloneDBデータベース環境の初期化パラメータを変更し、ファイルを保存します。

     CloneDBデータベース環境固有の初期化パラメータ(SGAサイズ、PGAターゲット、CPUの数などを制御するパラメータなど)を変更します。`CLONEDB`パラメータを`TRUE`に設定する必要があり、初期化パラメータ・ファイ

ルにこのパラメータが含まれています。初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

- SQL*Plusで、CloneDBデータベースにSYSDBA管理権限で接続します。
- clonedb.pl Perlスクリプトで生成されたSQLスクリプトを実行します。

  たとえば、スクリプトでデフォルト名を使用する場合は、SQLプロンプトで次のスクリプトを実行します。

```
crtdb.sql
dbren.sql
```

SQLスクリプトを手動で作成するには、次のステップを実行します:

- SYSDBAまたはSYSBACKUP管理権限でデータベースに接続します。

  「SQL*Plusを使用したデータベースへの接続」を参照してください。
- 次のステップを実行して、本番データベースからバックアップ制御ファイル・スクリプトを生成します。

  次のSQL文を実行します。

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

  この文により、制御ファイルを作成するSQL文を含むトレース・ファイルが生成されます。CREATE CONTROLFILE文を含むトレース・ファイルは、DIAGNOSTIC_DEST初期化パラメータで指定されたディレクトリに格納されます。データベース・アラート・ログでこのトレース・ファイルの名前と場所を確認します。
- ステップ3.bで生成したトレース・ファイルを開き、トレース・ファイル内のSTARTUP NOMOUNT文とCREATE CONTROLFILE文を新しいSQLスクリプトにコピーします。
- ステップ3.cで作成した新しいSQLスクリプトを次の方法で編集します。

  データベースの名前を、作成するCloneDBデータベースの名前に変更します。たとえば、PROD1をCLONE1に変更します。

  ログ・ファイルの場所を、CloneDBデータベース環境のディレクトリに変更します。たとえば、/u01/prod1/oracle/dbs/t_log1.fを/u03/clone1/oracle/dbs/t_log1.fに変更します。

  データファイルの場所を、バックアップの場所に変更します。たとえば、/u01/prod1/oracle/dbs/t_db1.fを/u02/oracle/backup/prod1/t_db1.fに変更します。

  次に、ALTER DATABASE BACKUP CONTROLFILE TO TRACE文で生成された元の文の例を示します。

```
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "PROD1" NORESETLOGS  ARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 '/u01/prod1/oracle/dbs/t_log1.f'  SIZE 25M BLOCKSIZE 512,
  GROUP 2 '/u01/prod1/oracle/dbs/t_log2.f'  SIZE 25M BLOCKSIZE 512
-- STANDBY LOGFILE
DATAFILE
  '/u01/prod1/oracle/dbs/t_db1.f',
  '/u01/prod1/oracle/dbs/t_ax1.f',
  '/u01/prod1/oracle/dbs/t_undo1.f',
  '/u01/prod1/oracle/dbs/t_xdb1.f',
  '/u01/prod1/oracle/dbs/undots.dbf'
CHARACTER SET WE8ISO8859P1
```

```
;
```

次に、新しいSQLスクリプトの変更された文の例を示します。

```
STARTUP NOMOUNT PFILE=/u03/clone1/oracle/dbs/clone1.ora
CREATE CONTROLFILE REUSE DATABASE "CLONE1" RESETLOGS  ARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 '/u03/clone1/oracle/dbs/t_log1.f'  SIZE 25M BLOCKSIZE 512,
  GROUP 2 '/u03/clone1/oracle/dbs/t_log2.f'  SIZE 25M BLOCKSIZE 512
-- STANDBY LOGFILE
DATAFILE
  '/u02/oracle/backup/prod1/t_db1.f',
  '/u02/oracle/backup/prod1/t_ax1.f',
  '/u02/oracle/backup/prod1/t_undo1.f',
  '/u02/oracle/backup/prod1/t_xdb1.f',
  '/u02/oracle/backup/prod1/undots.dbf'
CHARACTER SET WE8ISO8859P1
;
```

データファイルで取得した記憶域レベルのスナップショットがある場合は、RMANバックアップ・ファイル名を記憶域スナップショット名に置き換えることができます。

- SQLスクリプトを編集した後、そのスクリプトをCloneDBデータベース環境にアクセスできる場所に保存します。

  新しいSQLスクリプトの名前および保存場所をノートにとります。このスクリプトは、後続のステップで実行します。この例では、スクリプトの名前はcreate_clonedb1.sqlであると想定しています。

- テキスト形式の初期化パラメータ・ファイル(PFILE)を、本番データベース環境からCloneDBデータベース環境にコピーします。

  たとえば、テキスト形式の初期化パラメータ・ファイルを/u01/prod1/oracle/dbsから/u03/clone1/oracle/dbsにコピーします。ファイルの名前および場所は、変更したSQLスクリプトのSTARTUP NOMOUNTコマンドで指定した名前と場所に一致する必要があります。ステップ3.dの例では、ファイルは/u03/clone1/oracle/dbs/clone1.oraです。

- CloneDBデータベース環境の初期化パラメータを変更し、ファイルを保存します。

  CLONEDBパラメータを追加し、このパラメータがTRUEに設定されていることを確認します。CloneDBデータベース環境に固有の他のすべての初期化パラメータ(SGAサイズ、PGAターゲット、CPUの数などを制御するパラメータなど)を変更します。初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

- SQL*Plusで、CloneDBデータベースにSYSDBA管理権限で接続します。

- ステップ3.eで保存したSQLスクリプトを実行します。

  たとえば、SQL*Plusで次のように入力します。

  ```
  @create_clonedb1.sql
  ```

- バックアップの場所の各データファイルで、DBMS_DNFSパッケージのCLONEDB_RENAMEFILEプロシージャを実行し、CloneDBデータベース環境の適切な場所を指定します。

  たとえば、バックアップ・データファイルが/u02/oracle/backup/prod1/t_db1.fで、CloneDBデータベー

ス・データファイルが/u03/clone1/oracle/dbs/t_db1.fである場合は、次のプロシージャを実行します。

```
BEGIN
  DBMS_DNFS.CLONEDB_RENAMEFILE(
    srcfile  => '/u02/oracle/backup/prod1/t_db1.f',
    destfile => '/u03/clone1/oracle/dbs/t_db1.f');
END;
/
```

DBMS_DNFSパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

4. CloneDBデータベースをオンライン・バックアップから作成した場合は、CloneDBデータベースをリカバリします。全体オフライン・バックアップまたはBACKUP AS COPYバックアップを実行した場合、このステップは必要ありません。

   たとえば、次のSQL文をCloneDBデータベースで実行します。

   ```
   RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL;
   ```

   この文により、バックアップが実行されたときのアーカイブREDOログ・ファイルを求められます。

5. 次のSQL文を実行して、データベースをオープンします。

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

   CloneDBデータベースを使用する準備ができました。

本番データベースの追加のCloneDBデータベースを作成するには、ステップ3から5を各CloneDBデータベースに対して繰り返します。

親トピック: 非マルチテナント環境でのCloneDBを使用したデータベースのクローニング

2.11.1.3 CloneDBを使用したデータベースのクローニング後

CloneDBデータベースを作成した後、本番データベースを使用するほとんどすべての方法でそのデータベースを使用できます。最初は、CloneDBデータベースは各データファイルに対して最小限の記憶域を使用します。CloneDBデータベースの行に対して変更を加えると、記憶域がオンデマンドで割り当てられます。

同じバックアップ・ファイルを使用して、複数のCloneDBデータベースを作成できます。このバックアップは、RMANまたは記憶域レベルのスナップショットによって取得できます。データファイルで取得した記憶域レベルのスナップショットがある場合は、RMANバックアップ・ファイル名を記憶域スナップショット名に置き換えることができます。

V$CLONEDFILEビューを使用して、CloneDBデータベースの各データファイルに関する情報を表示できます。この情報には、バックアップのデータファイル名、CloneDBデータベースでの対応するデータファイル名、バックアップ・ファイルから読み取られたブロック数、バックアップ・ファイルに対して発行された要求の数が含まれます。

CloneDBデータベースはバックアップ・ファイルをバックエンド記憶域として使用するため、これらのバックアップ・ファイルは、実行するために各CloneDBデータベースで使用可能である必要があります。バックアップ・ファイルが使用不可能になると、CloneDBデータベースはエラーを返します。

CloneDBデータベースの使用が完了した後、CloneDBデータベース環境を破棄できます。CloneDBデータベース環境のすべてのファイルは、本番データベース環境またはバックアップ環境に影響を与えずに削除できます。

関連項目:

V$CLONEDFILEビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 非マルチテナント環境でのCloneDBを使用したデータベースのクローニング

2.11.2 マルチテナント環境でのデータベースのクローニング

マルチテナント環境でデータベースをクローニングできます。

マルチテナント環境でのデータベースのクローニングの詳細は、『Oracle Multitenant管理者ガイド』を参照してください。

親トピック: データベースのクローニング

2.11.3 Oracle Automatic Storage Management (Oracle ASM)を使用したデータベースのクローニング

Oracle Automatic Storage Management (Oracle ASM)では、マルチテナント・コンテナ・データベース(CDB)でのプラガブル・データベース(PDB)のクローニングがサポートされています。Oracle ASMでは、非CDBのクローニングはサポートされていません。

詳細は、次のガイドを参照してください。

- Oracle Automatic Storage Management管理者ガイド
- Oracle Multitenant管理者ガイド

親トピック: データベースのクローニング

2.12 データベースの削除

データベースの削除には、データファイル、オンラインREDOログ、制御ファイルおよび初期化パラメータ・ファイルの削除も含まれます。

警告:

データベースを削除すると、そのデータベースに含まれるすべてのデータが削除されます。

データベースを削除するには:

- 次の文を発行します。

```
DROP DATABASE;
```

DROP DATABASE文では、まず、すべての制御ファイルと制御ファイルに記述されている他のすべてのデータベース・ファイルが削除されます。その後、データベース・インスタンスが停止されます。

DROP DATABASE文の使用に成功するには、排他的に、制限モードでデータベースをマウントしておく必要があります。

DROP DATABASE文は、アーカイブREDOログ・ファイル、およびデータベースのコピーまたはバックアップには影響を与えません。これらのファイルを削除する場合は、RMANを使用することをお薦めします。

Database Configuration Assistantを使用してデータベースを作成した場合は、データベースおよびファイルの削除にこのツールを使用できます。

関連項目:

「データベースの可用性の変更」

親トピック: Oracle Databaseの作成および構成

2.13 データベースのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューに対してデータベースのコンテンツと構造に関する情報を問い合せることができます。

次のビューを使用して、データベース・コンテンツと構造に関する情報を表示できます。

ビュー	説明
DATABASE_PROPERTIES	永続的なデータベース・プロパティが表示されます。
GLOBAL_NAME	グローバル・データベース名が表示されます。
V$DATABASE	制御ファイル内のデータベース情報が表示されます。

関連トピック

- パラメータ設定を表示する方法

親トピック: Oracle Databaseの作成および構成

2.14 サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンス

この項では、Database Configuration Assistant (DBCA)サイレント・モードのコマンドの構文とオプションに関して詳しく説明します。

- DBCAコマンドライン構文の概要
  この項では、サイレント・モードのDBCAのコマンドライン構文の概要を示します。
- DBCAテンプレートについて
  DBCAを使用して、オラクル社提供のテンプレートまたは自分で作成したテンプレートからデータベースを作成できます。
- Oracleウォレットを使用したDBCAコマンドでのデータベース・ユーザー認証
  DBCAサイレント・モード・コマンドで、データベース・ユーザー認証用のセキュアな外部パスワード・ストアとしてOracleウォレットを使用できます。
- DBCAサイレント・モードのコマンド
  この項では、すべてのDBCAサイレント・モード・コマンドと、その構文およびパラメータの説明をリストします。
- DBCA終了のコード
  サイレント・モードでのDBCAコマンドの実行の結果は、終了コードとしてレポートされます。

関連項目:

対話モードでのDBCAの使用の詳細は、『Oracle Database 2日でデータベース管理者』を参照してください。

親トピック: Oracle Databaseの作成および構成

2.14.1 DBCAコマンドライン構文の概要

この項では、サイレント・モードでのDBCAのコマンドライン構文の概要を示します。

DBCAサイレント・モードのコマンド構文は、次のとおりです。

```
dbca  [-silent] [command [options]] [-h|-help]
```

次の表では、DBCAサイレント・モード・コマンドの構文について説明します。

表2-4 DBCAサイレント・モード・コマンドの構文の説明

オプション	説明
`-silent`	DBCA をサイレント・モードで実行する場合は、`-silent` を指定します。 サイレント・モードの DBCA は、コマンドライン・オプションとして指定された値を使用してデータベースを作成または変更します。
`command options`	DBCA コマンド、およびコマンドの有効なオプションを指定します。

`-h | -help`	DBCA のヘルプを表示します。
	
	特定のコマンドのヘルプを表示するには、次のように入力します。
	
	`dbca command -help`
	
	たとえば、`-createDatabase` コマンドのヘルプを表示するには、次のように入力します。
	
	`dbca -createDatabase -help`

次の例では、DBCAのサイレント・モードでデータベースを作成する方法を示します。

```
dbca -silent -createDatabase -templateName General_Purpose.dbc
                             -gdbname oradb.example.com
                             -sid oradb
                             -characterSet AL32UTF8
                             -memoryPercentage 30
                             -emConfiguration DBEXPRESS
Enter SYSTEM user password:
password
Enter SYS user password:
password
Copying database files
1% complete
3% complete
...
```

完全なサイレント操作が実行されるように、stdoutをファイルにリダイレクトできます。ただし、その場合は、コマンドライン引数またはレスポンス・ファイルに管理ユーザーのパスワードを指定する必要があることがあります。

ノート:

Oracle ウォレットを管理ユーザーのパスワードを格納するためのセキュアな外部パスワード・ストアとして使用する場合、これらのユーザーのパスワードをコマンドライン引数またはレスポンス・ファイルで指定する必要はありません。詳細は、Oracle ウォレットを使用した DBCA コマンドでのデータベース・ユーザー認証を参照してください。

DBCAコマンドライン引数の概要のヘルプを表示するには、次のコマンドを入力します。

```
dbca -help
```

デフォルトなど、引数の詳細は、配布媒体に同梱されているレスポンス・ファイル・テンプレートを参照してください。レスポンス・ファイル・テンプレートの名前と場所に関する情報は、ご使用のプラットフォームの『Oracle Databaseインストレーション・ガイド』を参照してください。

関連項目:

「DBCAサイレント・モードのコマンド」

親トピック: サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンス

2.14.2 DBCAテンプレートについて

DBCAを使用して、オラクル社提供のテンプレートまたは自分で作成したテンプレートからデータベースを作成できます。

DBCAテンプレートは、データベースの作成に必要な情報が含まれたXMLファイルです。次の2つのタイプのワークロードのテンプレートが提供されています。

- 汎用またはオンライン・トランザクション処理
- データ・ウェアハウス

使用するデータベースでサポートされるワークロードのタイプに適したテンプレートを選択します。どちらを選択するか不明な場合は、「汎用またはオンライン・トランザクション処理」テンプレートを使用します。特定のワークロード要件を満たすために、カスタム・テンプレートを作成することもできます。

ノート:

「汎用またはオンライン・トランザクション処理」テンプレートおよび「データ・ウェアハウス」テンプレートでは、COMPATIBLE 初期化パラメータを 12.1.0.2.0 に設定してデータベースを作成します。

親トピック: サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンス

2.14.3 Oracleウォレットを使用したDBCAコマンドでのデータベース・ユーザー認証

DBCAサイレント・モード・コマンドで、データベース・ユーザー認証用のセキュアな外部パスワード・ストアとしてOracleウォレットを使用できます。

Oracleウォレットは、Oracle Databaseの外部にあるセキュアなソフトウェア・コンテナであり、Oracle Databaseユーザーの認証資格証明の格納に使用できます。データベース・ユーザーの認証にOracleウォレットを使用するには、次のDBCAサイレント・モード・コマンド・パラメータを使用します。

- `useWalletForDBCredentials` : データベース・ユーザー認証にOracleウォレットを使用する場合は`true`を指定し、それ以外の場合は`false`を指定します。デフォルトは`false`です。

  `true`を指定する場合は、次のパラメータを追加指定します。

  - `dbCredentialsWalletLocation`: Oracleウォレット・ファイルを格納するディレクトリ。
  - (オプション) `dbCredentialsWalletPassword`: Oracleウォレット・アカウント・ユーザーのパスワード。Oracleウォレットで自動ログインが有効になっている場合、このパスワードを指定する必要はありません。

ユーザーを認証するために、サイレント・モードのDBCAで使用可能な、次のキーおよび関連パスワードをOracleウォレットに格納できます。

- `oracle.dbsecurity.sysPassword`: SYSユーザーのパスワード
- `oracle.dbsecurity.systemPassword`: SYSTEMユーザーのパスワード
- `oracle.dbsecurity.pdbAdminPassword`: プラガブル・データベース(PDB)管理者のパスワード
- `oracle.dbsecurity.dbsnmpPassword`: DBSNMPユーザーのパスワード
- `oracle.dbsecurity.asmsnmpPassword`: ASMSNMPユーザーのパスワード
- `oracle.dbsecurity.lbacsysPassword`: LBACSYSユーザーのパスワード
- `oracle.dbsecurity.sysdbaUserPassword`: 作成または構成しているデータベースのSYSDBAロール・ユーザーのパスワード
- `oracle.dbsecurity.oracleHomeUserPassword`: Oracleホーム・ユーザーのパスワード
- `oracle.dbsecurity.dvUserPassword`: Oracle Data Vaultユーザーのパスワード
- `oracle.dbsecurity.dvAccountManagerPassword`: Oracle Data Vaultアカウント・マネージャのパスワード
- `oracle.dbsecurity.emPassword`: Enterprise Manager管理者のパスワード
- `oracle.dbsecurity.asmPassword`: ASMユーザーのパスワード
- `oracle.dbsecurity.asmsysPassword`: :ASMSYSユーザーのパスワード
- `oracle.dbsecurity.walletPassword`: ディレクトリ・サービスでの認証用のOracleウォレット・アカウント・ユーザー・パスワード
- `oracle.dbsecurity.userDNPassword`: ディレクトリ・サービスのユーザーのパスワード
- `oracle.dbsecurity.srcDBsysdbaUserPassword`: データベースの複製などの特定の操作を実行するためにソースとして使用しているデータベースのSYSDBAロール・ユーザーのパスワード
- `oracle.dbsecurity.dbLinkUserPassword`: データベース・リンクのユーザーのパスワード

ノート:

Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用してOUDアカウントのパスワードをウォレットに格納する必要があります。

- `oracle.dbsecurity.walletPassword`
- `oracle.dbsecurity.userDNPassword`

関連項目:

mkstoreコマンドライン・ユーティリティを使用してセキュアな外部パスワード・ストアとしてOracleウォレットを構成する方法については、Oracle Databaseセキュリティ・ガイドを参照してください。

親トピック: サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンス

2.14.4 DBCAサイレント・モードのコマンド

この項では、すべてのDBCAサイレント・モード・コマンドと、その構文およびパラメータの説明をリストします。

- createDatabase
  createDatabaseコマンドにより、データベースが作成されます。
- createDuplicateDB
  createDuplicateDBコマンドは、Oracleデータベースの複製を作成します。
- configureDatabase
  configureDatabaseコマンドにより、データベースが構成されます。
- createTemplateFromDB
  createTemplateFromDBコマンドにより、既存のデータベースからデータベース・テンプレートが作成されます。
- createTemplateFromTemplate
  createTemplateFromTemplateコマンドにより、既存のデータベース・テンプレートからデータベース・テンプレートが作成されます。
- createCloneTemplate
  createCloneTemplateコマンドにより、既存のデータベースからクローン(シード)データベース・テンプレートが作成されます。
- deleteTemplate
  deleteTemplateコマンドにより、データベース・テンプレートが削除されます。
- generateScripts
  generateScriptsコマンドにより、データベースの作成に使用できるスクリプトが生成されます。
- deleteDatabase
  deleteDatabaseコマンドにより、データベースが削除されます。
- createPluggableDatabase
  createPluggableDatabaseコマンドにより、マルチテナント・コンテナ・データベース(CDB)内にプラガブル・データベース(PDB)が作成されます。
- unplugDatabase
  unplugDatabaseコマンドにより、マルチテナント・コンテナ・データベース(CDB)からプラガブル・データベース(PDB)が切断されます。
- deletePluggableDatabase
  deletePluggableDatabaseコマンドにより、PDBが削除されます。
- relocatePDB
  relocatePDBコマンドは、リモートCDBからローカルCDBにPDBを再配置します。
- configurePluggableDatabase
  configurePluggableDatabaseコマンドにより、プラガブル・データベース(PDB)が構成されます。
- addInstance
  addInstanceコマンドにより、データベース・インスタンスが管理者管理Oracle RACデータベースに追加されます。
- deleteInstance

deleteInstanceコマンドにより、データベース・インスタンスが管理者管理Oracle RACデータベースから削除されます。

- executePrereqs

  executePrereqsコマンドにより、前提条件チェックが実行され、その結果が報告されます。このコマンドは、データベースを作成するdbcaの実行前に環境をチェックするために使用できます。

親トピック: サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンス

2.14.4.1 createDatabase

createDatabaseコマンドにより、データベースが作成されます。

構文およびパラメータ

次の構文でdbca -createDatabaseコマンドを使用します。

```
dbca -createDatabase
    -responseFile | (-gdbName,-templateName)
    -responseFile response_file_directory
    -gdbName global_database_name
    -templateName database_template_name
    [-sid database_system_identifier]
    [-createAsContainerDatabase {true | false}
        [-numberOfPDBs number_of_pdbs]
        [-pdbName pdb_name]
        [-pdbStorageMAXSizeInMB maximum_storage_size_of_the_pdb]
        [-pdbStorageMAXTempSizeInMB maximum_temporary_storage_size_of_the_pdb]
        [-useLocalUndoForPDBs {true | false}]
        [-pdbAdminPassword pdb_administrator_password]
        [-pdbOptions pdb_options]
    [-sysPassword SYS_user_password]
    [-systemPassword SYSTEM_user_password]
    [-emConfiguration {DBEXPRESS | CENTRAL | BOTH | NONE}
        [-dbsnmpPassword DBSNMP_user_password]
        [-omsHost Oracle_Management_Server_host_name]
        [-omsPort Oracle_Management_Server_port_number]
        [-emUser EM_administrator_user_name]
        [-emPassword EM_administrator_user_password]
        [-emExpressPort EM_Express_port]
        [-emExpressPortAsGlobalPort EM_Express_global_port]]
    [-dvConfiguration {true | false}
        -dvUserName Database_Vault_owner_name
        -dvUserPassword Database_Vault_owner_password
        [-dvAccountManagerName Database_Vault_account_manager_name
        -dvAccountManagerPassword Database_Vault_account_manager_password]]
    [-olsConfiguration {true | false}
        [-configureWithOID configure_with_OID_flag]]
    [-datafileDestination data_files_directory]
    [-redoLogFileSize maximum_redo_log_file_size]
    [-recoveryAreaDestination recovery_files_directory
        [-recoveryAreaSize fast_recovery_area_size]]
    [-datafileJarLocation data_files_backup_directory]
    [-storageType {FS | ASM}
        [-asmsnmpPassword ASMSNMP_password]
        -datafileDestination database_files_directory]
    [-useWalletForDBCredentials {true | false}
        [-dbCredentialsWalletPassword wallet_account_password]
        -dbCredentialsWalletLocation wallet_files_directory]
    [-runCVUChecks {true | false}]
    [-nodelist database_nodes_list]
    [-oracleHomeUserName Oracle_Home_user_name]
        [-oracleHomeUserPassword Oracle_Home_user_password]
    [-enableArchive {true | false}
        [-archiveLogMode {AUTO | MANUAL}]
```

```
        [-archiveLogDest archive_log_files_directory]]
    [-memoryMgmtType {AUTO | AUTO_SGA | CUSTOM_SGA}]
    [-createListener new_database_listener]
    [-useOMF {true | false}]
    [-dbOptions database_options]
    [-customScripts list_of_custom_sql_scripts]
    [-policyManaged | -adminManaged]
    [-policyManaged
        -serverPoolName server_pool_names
        [-pqPoolName pq_pool_name]
        [-createServerPool new_server_pool_name]
            [-pqPoolName new_pq_pool_name]
            [-force]
            [-pqCardinality pq_cardinality_of_the_new_server_pool]
            [-cardinality cardinality_of_the_new_server_pool]]
    [-adminManaged]
    [-databaseConfigType {SINGLE | RAC | RACONENODE}
            [-RACOneNodeServiceName service_name_for_RAC_One_Node_database]]
    [-characterSet database_character_set]
    [-nationalCharacterSet database_national_character_set]
    [-registerWithDirService {true | false}
        [-dirServiceUserName directory_service_user_name]
        [-dirServicePassword directory_service_password]
        [-databaseCN database_common_name]
        [-dirServiceCertificatePath certificate_file_path]
        [-dirServiceUser directory_service_user_name]
        [-ldapDirectoryAccessType ldap_directory_access_type]
        [-useSYSAuthForLDAPAccess use_sys_user_for_ldap_access_flag]
        [-walletPassword wallet_password]]
    [-listeners listeners_list]
    [-variablesFile variables_file]
    [-variables variables_list]
    [-initParams initialization_parameters_list
        [-initParamsEscapeChar initialization_parameters_escape_character]]
    [-sampleSchema {true | false}]
    [-memoryPercentage | -totalMemory]
    [-memoryPercentage percentage_of_total_memory_to_assign_to_oracle_database]
    [-totalMemory total_memory_to_assign_to_oracle_database_in_MB]
    [-databaseType {MULTIPURPOSE | DATA_WAREHOUSING | OLTP}]
```

表2-5 createDatabaseパラメータ

パラメータ	必須/オプション	説明
`-responseFile response_file_directory`	必須	レスポンス・ファイルの絶対ディレクトリ・パス。
`-gdbName global_database_name`	必須	database_name.domain_name 形式のグローバル・データベース名。
`-templateName database_template_name`	必須	デフォルトの場所にある既存のデータベース・テンプレートの名前、またはデフォルトの場所にないデータベース・テンプレートへの完全なパス。
`-sid`	オプション	データベース・システム識別子(SID)

パラメータ	必須/オプション	説明
database_system_identifier		SID は、データベースを実行するインスタンスを一意に識別します。 指定しないと、デフォルトでデータベース名に設定されます。
-createAsContainerDatabase {true | false}	オプション	CDB を作成する場合は、true を指定します。非 CDB を作成す る場合は、false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータはオプションです。 ● -numberOfPDBs: 作成する PDB の数。デフォルトは 0(ゼロ)です。 ● -pdbName: 各 PDB の基本名。-numberOfPDBs 1 よりも大きい場合は、それぞれの名前に番号が追加され ます。このパラメータは、-numberOfPDBs が 0 (ゼロ)よ 大きい場合に指定する必要があります。 ● -pdbStorageMAXSizeInMB: PDB の最大記憶域 イズ(MB 単位)。 ● -pdbStorageMAXTempSizeInMB: PDB の最大一 時記憶域サイズ(MB 単位)。 ● -useLocalUndoForPDBs {true | false}: PDB にローカル UNDO を使用するかどうかを指定します ● -pdbAdminPassword: PDB 管理者のパスワード。 ● -pdbOptions: name:value 形式のカンマ区切りリ トとして PDB オプションを指定します。 例: JSERVER:true, DV:false
-sysPassword SYS_user_password	オプション	新しいデータベースの SYS ユーザー・パスワード。
-systemPassword SYSTEM_user_password	オプション	新しいデータベースの SYSTEM ユーザー・パスワード。
-emConfiguration {DBEXPRESS | CENTRAL | BOTH |	オプション	Enterprise Manager 構成の設定。 DBEXPRESS、CENTRAL または BOTH が指定されている場合は

パラメータ	必須/オプション	説明
NONE}		次の追加パラメータを指定します。 ● -dbsnmpPassword: DBSNMP ユーザー・パスワード ● -omsHost: Oracle Management Server のホスト名。 ● -omsPort: Oracle Management Server のポート番号。 ● -emUser: Enterprise Manager 管理者のユーザー名。 ● -emPassword: Enterprise Manager 管理者のパスワード。 ● -emExpressPort: Enterprise Manager Express のポート番号。 ● -emExpressPortAsGlobalPort: Enterprise Manager Express のグローバル・ポート番号。
-dvConfiguration {true | false}	オプション	Database Vault を有効化および構成する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加の Database Vault パラメータが必要です。 ● -dvUserName: Database Vault 所有者の名前を指定します。 ● -dvUserPassword: Database Vault 所有者のパスワードを指定します。 ● -dvAccountManagerName: Database Vault アカウント・マネージャの名前を指定します。 ● -dvAccountManagerPassword: Database Vault アカウント・マネージャのパスワードを指定します。

パラメータ	必須/オプション	説明
-olsConfiguration {true | false}	オプション	Oracle Label Security (OLS)を有効化および構成する場合は true を指定し、それ以外の場合は false を指定しますデフォルトは false です。 true を指定した場合は、-configureWithOID パラメータを追加で指定できます。この追加パラメータは、Oracle Internet Directory(OID)によって Oracle Label Security (OLS)を構成する場合に指定します。
-datafileDestination data_files_directory	オプション	データベースのデータ・ファイルの場所への完全なパス。
-redoLogFileSize maximum_size_of_redo_log_file	オプション	各オンライン REDO ログのサイズ(MB)。
-recoveryAreaDestination fast_recovery_area_directory	オプション	バックアップおよびリカバリ領域である、高速リカバリ領域の宛先ディレクトリ。高速リカバリ領域を無効にする場合は、NONE を指定します。 また、-recoveryAreaSize パラメータを使用すると、高速リカバリ領域のサイズを MB 単位で指定できます。このパラメータは省略可能です。
-datafileJarLocation data_files_backup_directory	オプション	RMAN バックアップの圧縮形式で格納されているデータベースのバックアップ・データ・ファイル(拡張子が.dfb のファイル)の絶対ディレクトリ・パス。
-storageType {FS | ASM}	オプション	FS または ASM のどちらかの記憶域タイプを指定します。 ● FS: ファイル・システム記憶域タイプ。 FS が指定されている場合、データベース・ファイルは、使用しているオペレーティング・システムのファイル・システムによって管理されます。データベース・ファイルを格納するディレクトリパスは、データベース・テンプレートまたは-datafileDestination パラメータを使用して指定できます。Oracle Database は、実際のファイルを作成および管理できます。 ● ASM: Oracle Automatic Storage Management

パラメータ	必須/オプション	説明
		(Oracle ASM)の記憶域のタイプ。 ASM が指定されている場合、データベース・ファイルは Oracle ASM ディスク・グループに配置されます。データベース・ファイルの配置とネーミングは Oracle Database によって自動的に管理されます。 ASM を指定した場合は、-asmsnmpPassword パラメータを使用して ASMSNMP パスワードも指定できます。このパラメータは省略可能です。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dbCredentialsWalletLocation: Oracle ウォレット・ファイルのディレクトリの場所。 ● -dbCredentialsWalletPassword: Oracle ウォレット・アカウントのパスワード。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● oracle.dbsecurity.walletPassword ● oracle.dbsecurity.userDNPassword
-runCVUChecks {true | false}	オプション	Oracle RAC データベースに定期的なクラスタ検証ユーティリティ・チェックを実行する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。
-nodelist database_nodes_list	オプション	データベース・ノードのカンマ区切りのリスト。
-enableArchive	オプション	ログ・ファイルのアーカイブを有効にする場合は true を指定し、そ

パラメータ	必須/オプション	説明
{true | false}		以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -archiveLogMode {AUTO | MANUAL}: 自動アーカイブ・モードまたは手動アーカイブ・モードのどちらかを定します。デフォルトは、自動アーカイブ・モードです。 ● -archiveLogDest: アーカイブ・ログ・ファイルを格納る場合のディレクトリ・パス。
-memoryMgmtType {AUTO | AUTO_SGA | CUSTOM_SGA}	オプション	次のいずれかのメモリー管理タイプを指定します。 ● AUTO: SGA および PGA の自動メモリー管理。 ● AUTO_SGA: SGA の自動共有メモリー管理。 ● CUSTOM_SGA: SGA の手動共有メモリー管理。 ノート: データベース・インスタンスの合計物理メモリーが 4GB より大きい場合は、データベースのインストール時および作成時に自動メモリー管理オプション AUTO を指定できません。このような環境の場合は、自動共有メモリー管理オプション AUTO_SGA を指定することをお薦めします。
-createListener new_database_listener	オプション	listener_name:port の形式でデータベースを登録するデータベース・リスナー。
-useOMF {true | false}	オプション	Oracle Managed Files (OMF)を使用する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。
-dbOptions database_options	オプション	name:value ペアのカンマ区切りリストとして、データベース・オプションを指定します。 例: JSERVER:true,DV:false
-customScripts custom_scripts_list	オプション	データベースの作成後に実行する必要がある SQL スクリプトのカンマ区切りリストを指定します。スクリプトはリストされている順序で実行されます。

パラメータ	必須/オプション	説明

`-oracleHomeUserName`
`Oracle_Home_user_name`
`-oracleHomeUserPassword`
`Oracle_Home_user_password`
	オプション	Oracle ホームのユーザー名とパスワード。

`-policyManaged`	オプション	ポリシー管理型データベース。 次の追加パラメータを指定できます。 ● `-serverPoolName`: 新しいサーバー・プールの作成時には 1 つのサーバー・プール名を指定します。それ以外の場合は、既存のサーバー・プールのカンマ区切りリストを指定します。 ● `-pqPoolName`: PQ プールの名前を指定します。 ● `-createServerPool`: 新しいサーバー・プールを作成する場合は、このパラメータを指定します。 : ● `-pqPoolName`: PQ プールの名前を指定します。 ● `-force`: 適切な空きサーバーが使用できないときに強制的にサーバー・プールを作成する場合は、このパラメータを指定します。 ● `-pqCardinality`: 新しいサーバー・プールの PQ カーディナリティを指定します。 ● `-cardinality`: 新しいサーバー・プールのカーディナリティを指定します。
`-adminManaged`	オプション	管理者管理データベース。
`-databaseConfigType` `{SINGLE | RAC | RACONENODE}`	オプション	次のいずれかのデータベース構成タイプを指定します。 ● `SINGLE`: 単一の個別のデータベース。 ● `RAC`: Oracle RAC データベース。

パラメータ	必須/オプション	説明
		● RACONENODE: Oracle RAC One Node データベース。 Oracle RAC One Node データベースの場合は、-RACOneNodeServiceName パラメータを使用してサービス名を指定できます。
-characterSet database_character_set	オプション	データベースの文字セット
-nationalCharacterSet database_national_character_set	オプション	データベースの各国語文字セット。
-registerWithDirService {true | false}	オプション	Lightweight Directory Access Protocol (LDAP)サービスに登録する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータが必要です。 ● -dirServiceUserName: LDAP サービスのユーザー名 ● -dirServicePassword: LDAP サービスのパスワー ● -databaseCN: データベースの共通名。 ● -dirServiceCertificatePath: データベースとディレクトリ・サービスの間の SSL を構成する際に使用する証明書ファイルのディレクトリ・パス。 ● -dirServiceUser: ディレクトリ・サービスのユーザー名。 ● -ldapDirectoryAccessType {PASSWORD | SSL}: LDAP ディレクトリのアクセス・タイプ。 ● -useSYSAuthForLDAPAccess {true | false}: LDAP アクセスに SYS ユーザー認証を使用するかどうかを指定します。 ● -walletPassword: データベース・ウォレットのパスワ

パラメータ	必須/オプション	説明
		ド
-listeners listeners_list	オプション	データベースのリスナーのカンマ区切りリスト。
-variablesFile variables_file	オプション	データベース・テンプレートの完全なディレクトリ・パスと変数ファイルの名前。
-variables variables_list	オプション	データベース・テンプレートの変数に対応する name=value ペアのカンマ区切りリスト。
-initParams initialization_parameters_list	オプション	データベースの初期化パラメータ値の name=value ペアのカンマ区切りリスト。 初期化パラメータの複数の値の間に特定のエスケープ文字を使用する場合は、-initParamsEscapeChar パラメータを追加指定できます。エスケープ文字を指定しない場合は、デフォルトのエスケープ文字としてバックスラッシュ(/)が使用されます。
-sampleSchema {true | false}	オプション	HR サンプル・スキーマ(EXAMPLE 表領域)をデータベースに含める場合は、true を指定します。オラクル社のガイドおよび入門資料には、サンプル・スキーマに基づいた例が含まれています。本番データベースにはサンプル・スキーマをインストールしないことをお薦めします。 HR サンプル・スキーマを使用せずにデータベースを作成する場合は false を指定します。デフォルトは false です。
-memoryPercentage percentage_of_total_memory_to_assign_to_oracle_database または -totalMemory total_memory_to_assign_to_oracle_database_in_MB	オプション	-memoryPercentage または-totalMemory のどちらかを指定します。 ● -memoryPercentage データベースが使用できる物理メモリーの割合。 ● -totalMemory。 データベースが使用できる物理メモリーの合計量(MB 単位)。

パラメータ	必須/オプション	説明

`-databaseType`
 	
`{MULTIPURPOSE | DATA_WAREHOUSING | OLTP}`
	オプション	データベースの目的が OLTP とデータ・ウェアハウスの両方である場合は、`MULTIPURPOSE` を指定します。
		
		データベースの主な目的がデータ・ウェアハウスである場合は、`DATA_WAREHOUSING` を指定します。
		
		データベースの主な目的がオンライン・トランザクション処理である場合は、`OLTP` を指定します。

関連項目:

Oracle Databaseサンプル・スキーマ

親トピック: DBCAサイレント・モードのコマンド

2.14.4.2 createDuplicateDB

`createDuplicateDB`コマンドは、Oracleデータベースの複製を作成します。

前提条件

次に、`createDuplicateDB`コマンドを使用するための前提条件を示します。

- 複製するデータベースがARCHIVELOGモードになっている。
- 複製するデータベースがリモート・サーバーにある場合は、DBCAが実行されているシステムからリモート・サーバーへの接続が必要。

構文およびパラメータ

次の構文で`dbca -createDuplicateDB`コマンドを使用します。

```
dbca -createDuplicateDB
    -gdbName global_database_name
    -primaryDBConnectionString easy_db_connection_string
    -sid database_system_identifier
    [-initParams initialization_parameters
        [-initParamsEscapeChar initialization_parameters_escape_character]]
    [-sysPassword SYS_user_password]
    [-policyManaged | -adminManaged]
    [-policyManaged
        -serverPoolName server_pool_names
        [-pqPoolName pq_pool_name]
        [-createServerPool new_server_pool_name
            [-pqPoolName new_pq_pool_name]
            [-force]
            [-pqCardinality pq_cardinality_of_the_new_server_pool]
            [-cardinality cardinality_of_the_new_server_pool]]]
    [-adminManaged]
    [-nodelist database_nodes_list]
    [-datafileDestination data_files_directory]
    [-recoveryAreaDestination recovery_files_directory
        [-recoveryAreaSize fast_recovery_area_size]]
    [-databaseConfigType {SINGLE | RAC | RACONENODE}
```

```
        [-RACOneNodeServiceName service_name_for_RAC_One_Node_database]]
    [-useOMF {true | false}]
    [-storageType {FS | ASM}
        [-asmsnmpPassword ASMSNMP_password]
        -datafileDestination database_files_directory]
    [-createListener new_database_listener]
    [-createAsStandby
        [-dbUniqueName db_unique_name_for_standby_database]]
    [-customScripts custom_sql_scripts_to_run_after_database_creation]
    [-useWalletForDBCredentials {true | false}
        -dbCredentialsWalletPassword wallet_account_password
        -dbCredentialsWalletLocation wallet_files_directory]
```

表2-6 createDuplicateDBパラメータ

パラメータ	必須/オプション	説明
-gdbName global_database_name	必須	database_name.domain_name 形式の重複するデータベースのグローバル・データベース名
-primaryDBConnectionString easy_db_connection_string	必須	複製するデータベースに接続するための簡易接続文字列。簡易接続文字列は次の形式で指定する必要があります。 "host[:port][/service_name][:server][/instance_name]" 詳細は、SQL*Plus の CONNECT コマンドの構文で「connect_identifier (2)」の説明を参照してください。
-sid database_system_identifier	必須	重複するデータベースのデータベース・システム識別子(SID)。 SID は、データベースを実行するインスタンスを一意に識別します。指定しないと、デフォルトでデータベース名に設定されます。
-initParams initialization_parameters_list	オプション	データベースの初期化パラメータ値の name=value ペアのカンマ切りリスト。 初期化パラメータの複数の値の間に特定のエスケープ文字を使用する場合は、-initParamsEscapeChar パラメータを追加指定できます。エスケープ文字を指定しない場合は、デフォルトのエスケープ文字としてバックスラッシュ(/)が使用されます。
-sysPassword SYS_user_password	オプション	SYS ユーザーのパスワード。
-policyManaged	オプション	ポリシー管理型データベース。 ノート: ポリシー管理型データベースまたは管理者管理データベー

パラメータ	必須/オプション	説明
		を指定できます。 次の追加パラメータを指定できます。 ● -serverPoolName: 新しいサーバー・プールの作成時には 1 つのサーバー・プール名を指定します。それ以外の場合は、既存のサーバー・プールのカンマ区切りリストを指定します。 ● -pqPoolName: PQ プールの名前を指定します。 ● -createServerPool: 新しいサーバー・プールを作成する場合は、このパラメータを指定します。 ● -pqPoolName: PQ プールの名前を指定します。 ● -force: 適切な空きサーバーが使用できないときに強制的にサーバー・プールを作成する場合は、このパラメータを指定します。 ● -pqCardinality: 新しいサーバー・プールの PQ カーディナリティを指定します。 ● -cardinality: 新しいサーバー・プールのカーディナリティを指定します。
-adminManaged	オプション	管理者管理データベース。 ノート: ポリシー管理型データベースまたは管理者管理データベースを指定できます。
-nodelist database_nodes_list	オプション	管理者管理データベースの場合は、カンマで区切ってデータベースノードを指定します。
-datafileDestination data_files_directory	オプション	データベースのデータ・ファイルの完全なディレクトリ・パス。
-recoveryAreaDestination fast_recovery_area_directory	オプション	バックアップおよびリカバリ領域である、高速リカバリ領域の宛先ディレクトリ。高速リカバリ領域を無効にする場合は、NONE を指定しま

パラメータ	必須/オプション	説明
		す。 また、-recoveryAreaSize パラメータを使用すると、高速リカ リ領域のサイズを MB 単位で指定できます。このパラメータは省略 能です。
-databaseConfigType {SINGLE | RAC | RACONENODE}	オプション	次のいずれかのデータベース構成タイプを指定します。 ● SINGLE: 単一の個別のデータベース。 ● RAC: Oracle RAC データベース。 ● RACONENODE: Oracle RAC One Node データベース。 Oracle RAC One Node データベースの場合は、-RACOneNodeServiceName パラメータを使用してサービス名を指定できます。
-useOMF {true | false}	オプション	Oracle Managed Files (OMF)を使用する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。
-storageType {FS | ASM}	オプション	FS または ASM のどちらかの記憶域タイプを指定します。 ● FS: ファイル・システム記憶域タイプ。 FS が指定されている場合、データベース・ファイルは、使用しているオペレーティング・システムのファイル・システムによって管理されます。データベース・ファイルを格納するディレクトリパスは、データベース・テンプレートまたは-datafileDestination パラメータを使用して指定できます。Oracle Database は、実際のファイルを作成および管理できます。 ● ASM: Oracle Automatic Storage Management (Oracle ASM)の記憶域のタイプ。 ASM が指定されている場合、データベース・ファイルは Oracle ASM ディスク・グループに配置されます。データベース・ファイルの配置とネーミングは Oracle Database によ

パラメータ	必須/オプション	説明
		て自動的に管理されます。 ASM を指定した場合は、-asmsnmpPassword パラメータを使用して ASMSNMP パスワードも指定できます。このパラメータは省略可能です。
-createListener new_database_listener	オプション	listener_name:port の形式でデータベースを登録するデータベース・リスナー。
-createAsStandby	オプション	重複するデータベースがプライマリ・データベースのスタンバイ・データベースであることを指定します オプションで、-dbUniqueName パラメータを使用して、スタンバイ・データベースの一意データベース名を設定します。-dbUniqueName パラメータが指定されていない場合は、DB_NAME 初期化パラメータの値が使用されます。
-customScripts custom_sql_scripts_to_run_after_database_creation	オプション	重複するデータベースの作成後に実行する必要のある SQL スクリプトのカンマ区切りリスト。スクリプトは、リストされている順に実行されます。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dbCredentialsWalletPassword: Oracle ウォレット・アカウントのパスワード。 ● -dbCredentialsWalletLocation: Oracle ウォレット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● oracle.dbsecurity.walletPassword

パラメータ	必須/オプション	説明
		● oracle.dbsecurity.userDNPassword

関連トピック

- 『Oracle Data Guard概要および管理』

親トピック: DBCAサイレント・モードのコマンド

2.14.4.3 configureDatabase

configureDatabaseコマンドにより、データベースが構成されます。

構文およびパラメータ

次の構文でdbca -configureDatabaseコマンドを使用します。

```
dbca -configureDatabase
   -sourceDB database_sid
   [-sysDBAUserName SYSDBA_user_name]
   [-sysDBAPassword SYSDBA_user_password]
   [-registerWithDirService {true | false}
      -dirServiceUserName directory_service_user_name
      [-databaseCN database_common_name]
      [-dirServiceCertificatePath certificate_file_path]
      [-dirServiceUser directory_service_user_name]
      [-dirServicePassword directory_service_password]
      [-ldapDirectoryAccessType ldap_directory_access_type]
      [-useSYSAuthForLDAPAccess use_sys_user_for_ldap_access_flag]
      [-walletPassword wallet_password]]
   [-unregisterWithDirService {true | false}
      -dirServiceUserName directory_service_user_name
      [-dirServicePassword directory_service_password]
      [-walletPassword wallet_password]]
   [-addDBOption database_options]
   [-dvConfiguration {true | false}
      -dvUserName Database_Vault_owner_name
      -dvUserPassword Database_Vault_owner_password
      [-dvAccountManagerName Database_Vault_account_manager_name]
      [-dvAccountManagerPassword Database_Vault_account_manager_password]]
   [-olsConfiguration {true | false}
      -configureWithOID configure_with_OID_flag]
   [-configureOracleR
      -oracleRConfigTablespace tablespace_for_Oracle_R_configuration]
   [-moveDatabaseFiles
      -datafileDestination data_files_directory
      -sourceDB database_sid
      [-initParams initialization_parameters_list
        [-initParamsEscapeChar initialization_parameters_escape_character]]
      [-recoveryAreaDestination fast_recovery_area_directory
        [-recoveryAreaSize fast_recovery_area_size]]
      [-useOMF {true | false}]]
   [-regenerateDBPassword {true | false}]
   [-useWalletForDBCredentials {true | false}
      -dbCredentialsWalletPassword wallet_account_password
      -dbCredentialsWalletLocation wallet_files_directory]
```

表2-7 configureDatabaseパラメータ

パラメータ	必須/オプション	説明
-sourceDB database_sid	必須	構成するデータベースのデータベース・システム識別子(SID)。
-sysDBAUserName SYSDBA_user_name	オプション	SYSDBA 権限を持つユーザーのユーザー名。
-sysDBAPassword SYSDBA_user_password	オプション	SYSDBA 権限を持つユーザーのパスワード。
-registerWithDirService {true | false}	オプション	Lightweight Directory Access Protocol (LDAP)サービス 登録する場合は true を指定し、それ以外の場合は false を 定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータが必要です。 ● -dirServiceUserName: LDAP サービスのユーザー 名。 ● -dirServicePassword: LDAP サービスのパスワー ● -databaseCN: データベースの共通名。 ● -dirServiceCertificatePath: ディレクトリ・サ ビスの証明書ファイルのパス。 ● -dirServiceUser: ディレクトリ・サービスのユーザー 名。 ● -ldapDirectoryAccessType {PASSWORD | SSL}: LDAP ディレクトリのアクセス・タイプ。 ● -useSYSAuthForLDAPAccess {true | false}: LDAP アクセスの SYS ユーザー認証を使用す るかどうかを指定します。 ● -walletPassword: データベース・ウォレットのパスワ ド
-unregisterWithDirService {true | false}	オプション	Lightweight Directory Access Protocol (LDAP)サービス 登録を解除する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータが必要です。

パラメータ	必須/オプション	説明
		● -dirServiceUserName: LDAP サービスのユーザー名。 ● -dirServicePassword: LDAP サービスのパスワー ● -walletPassword: データベース・ウォレットのパスワード
-addDBOption database_options	オプション	カンマ区切りリストの形式で、次の Oracle Database オプションの 1 つまたは複数を指定します。 ● JSERVER: Oracle JServer JAVA Virtual Machin ● ORACLE_TEXT: Oracle Text ● IMEDIA: Oracle Locator (完全にサポートされている および Oracle Multimedia (サポート対象外) ● CWMLITE: Oracle OLAP with Oracle Warehous Builder (OWB) ● SPATIAL: Oracle Spatial and Graph ● OMS: Oracle Management Server ● APEX: Oracle Application Express ● DV: Oracle Database Vault 例: -addDBOption JSERVER,ORACLE_TEXT,OMS
-dvConfiguration {true | false}	オプション	Database Vault を有効化および構成する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加の Database Vault パラメータが必要です。 ● -dvUserName: Database Vault 所有者のユーザー名を指定します。

パラメータ	必須/オプション	説明
		● -dvUserPassword: Database Vault 所有者のパスワードを指定します。 ● -dvAccountManagerName: 個別の Database Vault アカウント・マネージャを指定します。 ● -dvAccountManagerPassword: Database Vault アカウント・マネージャのパスワードを指定します。
-olsConfiguration {true | false}	オプション	Oracle Label Security を有効化および構成する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、Oracle Internet Directory (OID) によって Oracle Label Security を構成するための -configureWithOID パラメータを追加指定できます。このパラメータは省略可能です。
-configureOracleR	オプション	データベースの Oracle R を構成する場合は、このパラメータを指定します。 また、Oracle R 構成用の表領域(SYSAUX 表領域など)を割り当てるために、-oracleRConfigTablespace パラメータも指定できます。
-moveDatabaseFiles	オプション	このパラメータは、データベース・ファイルの記憶域の場所を別の記憶域の場所に移動する場合に指定します。たとえば、データベース・ファイルを ASM から FS に移動する場合や、FS から ASM に移動する場合です。 次に示す追加パラメータを指定します。 ● -datafileDestination: すべてのデータベース・ファイルの宛先ディレクトリ ● -sourceDB: 単一インスタンス・データベースのデータベース・システム識別子(SID)、または Oracle RAC データベースの一意のデータベース名。 ● -initParams: データベース初期化パラメータ

パラメータ	必須/オプション	説明
		(name=value ペアのカンマ区切りリストの形式) また、初期化パラメータの複数の値の間に特定のエスケー 文字を使用する場合は、-initParamsEscapeCha パラメータを指定できます。エスケープ文字を指定しない場 合は、デフォルトのエスケープ文字としてバックスラッシュ(/) 使用されます。 ● -recoveryAreaDestination: バックアップおよび カバリ領域である、高速リカバリ領域の宛先ディレクトリ。高 速リカバリ領域を無効にする場合は、NONE を指定します また、-recoveryAreaSize パラメータを使用すると、 高速リカバリ領域のサイズを MB 単位で指定できます。こ パラメータは省略可能です。 ● -useOMF: Oracle Managed Files (OMF)を使用す る場合は true を指定し、それ以外の場合は false を 指定します。
-regenerateDBPassword {true | false}	オプション	Oracle Internet Directory (OID)サーバーの登録パスワード 再生成する場合は true を指定し、それ以外の場合は false 指定します。デフォルトは false です。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は tru を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dbCredentialsWalletLocation: Oracle ウ レット・ファイルのディレクトリの場所。 ● -dbCredentialsWalletPassword: Oracle ウ レット・アカウントのパスワード。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次 キーを使用して OUD パスワードをウォレットに格納する必要があり す。

パラメータ	必須/オプション	説明
		• oracle.dbsecurity.walletPassword • oracle.dbsecurity.userDNPassword

親トピック: DBCAサイレント・モードのコマンド

2.14.4.4 createTemplateFromDB

createTemplateFromDBコマンドにより、既存のデータベースからデータベース・テンプレートが作成されます。

構文およびパラメータ

次の構文でdbca -createTemplateFromDBコマンドを使用します。

```
dbca -createTemplateFromDB
   -sourceDB source_database_sid
   -templateName new_database_template_name
   -sysDBAUserName SYSDBA_user_name
   -sysDBAPassword SYSDBA_user_password
   [-maintainFileLocations {true | false}]
   [-connectionString easy_connect_string]
   [-useWalletForDBCredentials {true | false}
       -dbCredentialsWalletPassword wallet_account_password
       -dbCredentialsWalletLocation wallet_files_directory]
```

表2-8 createTemplateFromDBパラメータ

パラメータ	必須/オプション	説明
-sourceDB source_database_sid	必須	ソース・データベースのシステム識別子(SID)。
-templateName new_database_template_name	必須	新しいデータベース・テンプレートの名前。
-sysDBAUserName SYSDBA_user_name	必須	SYSDBA 権限を持つユーザーのユーザー名。
-sysDBAPassword SYSDBA_user_password	必須	SYSDBA 権限を持つユーザーのパスワード。
-maintainFileLocations {true | false}	オプション	テンプレートのデータベースのファイル場所を使用する場合は true を指定します。 テンプレートのファイルの場所とは異なる場所を使用する場合は false (デフォルト)を指定します。ファイルの場所は、Oracle Flexible Architecture (OFA)によって決められます。
-connectionString	オプション	リモート・データベースに接続するための次の形式の簡単な接続文

パラメータ	必須/オプション	説明
`easy_connect_string`		列。 `"host[:port][/service_name][:server][/instance_name]"`

`-useWalletForDBCredentials`
 	
`{true | false}`
	オプション	データベース資格証明に Oracle Wallet を使用する場合は `true` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。
		
		`true` を指定した場合は、次の追加パラメータを指定できます。
		
		● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。
		
		● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。
		
		ノート:
		
		Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。
		
		● `oracle.dbsecurity.walletPassword`
		
		● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.5 createTemplateFromTemplate

`createTemplateFromTemplate`コマンドにより、既存のデータベース・テンプレートからデータベース・テンプレートが作成されます。

構文およびパラメータ

次の構文で`dbca -createTemplateFromTemplate`コマンドを使用します。

```
dbca -createTemplateFromTemplate
    -sourcetemplateName existing_template_name
    -templateName new_template_name
    [-variables variables_list]
    [-characterSet database_character_set]
    [-nationalCharacterSet database_national_character_set]
    [-recoveryAreaDestination fast_recovery_area_directory]
        -recoveryAreaSize fast_recovery_area_size]
    [-datafileDestination data_files_directory]
    [-useOMF {true | false}]
    [-datafileJarLocation database_backup_files_directory]
    [-memoryPercentage percentage_of_total_memory_to_assign_to_oracle_database]
```

```
    [-totalMemory total_memory_to_assign_to_oracle_database]
    [-dbOptions database_options]
    [-variablesFile variables_file]
    [-redoLogFileSize redo_log_file_size]
    [-initParams initialization_parameters_list]
        [-initParamsEscapeChar escape_character_for_initialization_parameters]
    [-storageType {FS | ASM}
        [-asmsnmpPassword ASMSNMP_password]
        -datafileDestination data_files_directory]
    [-enableArchive {true | false}
         -archiveLogMode {AUTO | MANUAL}
         -archiveLogDest archive_logs_directory
    [-memoryMgmtType {AUTO | AUTO_SGA | CUSTOM_SGA}]
    [-useWalletForDBCredentials {true | false}
         -dbCredentialsWalletPassword wallet_account_password
         -dbCredentialsWalletLocation wallet_files_directory]
```

表2-9 createTemplateFromTemplateのパラメータ

パラメータ	必須/オプション	説明
-sourceTemplateName existing_template_name	必須	デフォルトの場所にある既存のデータベース・テンプレートの名前、またはデフォルトの場所にないデータベース・テンプレートへの完全なパス。
-templateName new_template_name	必須	新しいデータベース・テンプレートの名前。
-variables variables_list	オプション	データベース・テンプレートの変数に対応する name=value ペアのカンマ区切りリスト。
-characterSet database_character_set	オプション	データベースの文字セット
-nationalCharacterSet database_national_character_set	オプション	データベースの各国語文字セット。
-recoveryAreaDestination fast_recovery_area_directory	オプション	バックアップおよびリカバリ領域である、高速リカバリ領域のディレクトリ・パス。
-datafileDestination data_files_directory	オプション	データ・ファイルのディレクトリ・パス。
-useOMF {true | false}	オプション	Oracle Managed Files (OMF)を使用する場合は true を指定し、それ以外の場合は false を指定します。

パラメータ	必須/オプション	説明
-datafileJarLocation database_backup_files_directory	オプション	データベース・オフライン・バックアップの場所(クローン・データベース作成の場合のみ)。 シード・データベースのデータファイルは、RMAN バックアップの圧縮形式で、拡張子が.dfb のファイルに格納されます。
-memoryPercentage percentage_of_total_memory_to_assign_to_oracle_database または -totalMemory total_memory_to_assign_to_oracle_database	オプション	-memoryPercentage または-totalMemory のどちらかを指定します。 ● -memoryPercentage データベースが使用できる物理メモリーの割合。 ● -totalMemory データベースが使用できる物理メモリーの量(MB 単位)。
-dbOptions database_options	オプション	name:value ペアのカンマ区切りリストとして、データベース・オプションを指定します。 例: JSERVER:true,DV:false
-variablesFile variables_file	オプション	データベース・テンプレートの変数とその値が含まれているファイルへの完全なディレクトリ・パスとファイル名。
-redoLogFileSize redo_log_file_size	オプション	各オンライン REDO ログ・ファイルのサイズ(MB 単位)。
-initParams initialization_parameters_list	オプション	データベース初期化パラメータとその値についての name=value ペアのカンマ区切りリスト。
-storageType {FS | ASM}	オプション	ファイル・システムの場合は FS、Oracle Automatic Storage Management (Oracle ASM)システムの場合は ASM を指定します。 FS が指定されている場合、データベース・ファイルは、使用しているオペレーティング・システムのファイル・システムによって管理されます。データベース・ファイルを格納するディレクトリ・パスは、-datafileDestination パラメータを使用して指定します。 ASM を指定した場合、データベース・ファイルは Oracle ASM ディ

パラメータ	必須/オプション	説明
		ク・グループに配置されます。データベース・ファイルの配置とネーミンは Oracle Database によって自動的に管理されます。-asmsnmpPassword パラメータを使用して、ASM 監視用の ASMSNMP パスワードも指定します。
-enableArchive {true | false}	オプション	ログ・ファイルのアーカイブを有効にするには、true を指定します。フォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -archiveLogMode {AUTO | MANUAL}: 自動アーカイブ・モード(AUTO)または手動アーカイブ・モード(MANUAL)のどちらかを指定します。デフォルトは、自動アカイブ・モード(AUTO)です。 ● -archiveLogDest: アーカイブ・ログ・ファイルを格納る場合のディレクトリ・パス。
-memoryMgmtType {AUTO | AUTO_SGA | CUSTOM_SGA}	オプション	次のいずれかのメモリー管理タイプを指定します。 ● AUTO: SGA および PGA の自動メモリー管理。 ● AUTO_SGA: SGA の自動共有メモリー管理。 ● CUSTOM_SGA: SGA の手動共有メモリー管理。 ノート: データベース・インスタンスの合計物理メモリーが 4GB より大きい場合は、データベースのインストール時および作成時に自動メモリー管理オプション AUTO を指定できません。このような環境の場合は、自動共有メモリー管理オプション AUTO_SGA を指定することをお薦めします。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dbCredentialsWalletPassword: Oracle ウォレット・アカウントのパスワード。 ● -dbCredentialsWalletLocation: Oracle ウ

パラメータ	必須/オプション	説明
		レット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次キーを使用して OUD パスワードをウォレットに格納する必要があります。 ● `oracle.dbsecurity.walletPassword` ● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.6 createCloneTemplate

`createCloneTemplate`コマンドにより、既存のデータベースからクローン(シード)データベース・テンプレートが作成されます。

構文およびパラメータ

次の構文で`dbca -createCloneTemplate`コマンドを使用します。

```
dbca -createCloneTemplate
   -sourceSID source_database_sid | -sourceDB source_database_name
   -templateName new_database_template_name
   [-promptForWalletPassword]
   [-rmanParallelism parallelism_integer_value]
   [-maxBackupSetSizeInMB maximum_backup_set_size_in_MB]
   [-dataFileBackup {true | false}]
   [-datafileJarLocation data_files_backup_directory]
   [-sysDBAUserName SYSDBA_user_name]
   [-sysDBAPassword SYSDBA_user_password]
   [-useWalletForDBCredentials {true | false}
       -dbCredentialsWalletPassword wallet_account_password
       -dbCredentialsWalletLocation wallet_files_directory]
```

表2-10 createCloneTemplateのパラメータ

パラメータ	必須/オプション	説明

`-sourceSID source_database_sid`
 	
または
 	
`-sourceDB source_database_name`
	必須	ソース・データベースのシステム識別子(SID)またはソース・データベースの名前を指定します。

`-templateName new_database_template_name`	必須	新しいデータベース・テンプレートの名前。

パラメータ	必須/オプション	説明
`-sysDBAUserName SYSDBA_user_name`	オプション	SYSDBA 権限を持つユーザーのユーザー名。
`-sysDBAPassword SYSDBA_user_password`	オプション	SYSDBA 権限を持つユーザーのパスワード。
`-maxBackupSetSizeInMB maximum_backup_set_size_in_MB`	オプション	バックアップ・セットの最大サイズ(MB)
`-rmanParallelism parallelism_integer_value`	オプション	RMAN 操作の並列度の整数値。
`-datafileJarLocation data_files_backup_directory`	オプション	データ・ファイルを圧縮形式でバックアップとして保存する完全なディクトリ・パス。
`-dataFileBackup {true | false}`	オプション	データ・ファイルのバックアップを作成する場合は `true` を指定し、それ以外の場合は `false` を指定します。
`-useWalletForDBCredentials {true | false}`	オプション	データベース資格証明に Oracle Wallet を使用する場合は `true` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。 `true` を指定した場合は、次の追加パラメータを指定できます。 ● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。 ● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● `oracle.dbsecurity.walletPassword` ● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.7 deleteTemplate

`deleteTemplate`コマンドにより、データベース・テンプレートが削除されます。

構文およびパラメータ

次の構文でdbca -deleteTemplateコマンドを使用します。

```
dbca -deleteTemplate
   -templateName name_of_an_existing_database_template
   [-useWalletForDBCredentials {true | false}
      -dbCredentialsWalletPassword wallet_account_password
      -dbCredentialsWalletLocation wallet_files_directory]
```

表2-11 deleteTemplateのパラメータ

パラメータ	必須/オプション	説明
-templateName name_of_an_existing_database_template	必須	削除する既存のデータベース・テンプレートの名前。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は tru を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dbCredentialsWalletPassword: Oracle ウォレット・アカウントのパスワード。 ● -dbCredentialsWalletLocation: Oracle ウォレット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● oracle.dbsecurity.walletPassword ● oracle.dbsecurity.userDNPassword

親トピック: DBCAサイレント・モードのコマンド

2.14.4.8 generateScripts

generateScriptsコマンドにより、データベースの作成に使用できるスクリプトが生成されます。

構文およびパラメータ

次の構文でdbca -generateScriptsコマンドを使用します。

```
dbca -generateScripts
    -templateName database_template_name
```

```
    -gdbName global_database_name
    [-sid database_system_identifier]
    [-scriptDest sql_scripts_directory]
    [-createAsContainerDatabase {true | false}
        [-numberOfPDBs number_of_pdbs_to_create]
        [-pdbName pdb_name]
        [-pdbStorageMAXSizeInMB maximum_storage_size_of_the_pdb]
        [-pdbStorageMAXTempSizeInMB maximum_temporary_storage_size_of_the_pdb]
        [-useLocalUndoForPDBs {true | false}]
        [-pdbAdminPassword pdb_administrator_password]
        [-pdbOptions pdb_options]
    [-sysPassword SYS_user_password]
    [-systemPassword SYSTEM_user_password]
    [-emConfiguration {DBEXPRESS | CENTRAL | BOTH | NONE}
        [-dbsnmpPassword DBSNMP_user_password]
        [-omsHost EM_Management_Server_host_name]
        [-omsPort EM_Management_Server_port_number]
        [-emUser EM_administrator_name]
        [-emPassword EM_administrator_password]
        [-emExpressPort EM_Express_port]
        [-emExpressPortAsGlobalPort EM_Express_global_port]]
    [-dvConfiguration {true | false}
        -dvUserName Database_Vault_owner_user_name
        -dvUserPassword Database_Vault_owner_user_password
        [-dvAccountManagerName Database_Vault_account_manager_name
        -dvAccountManagerPassword Database_Vault_account_manager_password]]
    [-olsConfiguration {true | false}
        [-configureWithOID configure_with_OID_flag]]
    [-datafileDestination data_files_directory]
    [-redoLogFileSize maximum_redo_log_file_size_in_MB]
    [-recoveryAreaDestination fast_recovery_area_directory
        [-recoveryAreaSize fast_recovery_area_size]]
    [-datafileJarLocation data_files_backup_directory]
    [-responseFile response_file_directory]
    [-storageType {FS | ASM}
        [-asmsnmpPassword ASMSNMP_password]
        -datafileDestination data_files_directory]
    [-runCVUChecks {true | false}]
    [-nodelist database_nodes_list]
    [-enableArchive {true | false}
        [-archiveLogMode {AUTO | MANUAL}]
        [-archiveLogDest archive_log_files_directory]]
    [-memoryMgmtType {AUTO | AUTO_SGA | CUSTOM_SGA}]
    [-createListener new_database_listener_to_register_the_database_with]
    [-useOMF {true | false}]
    [-dbOptions database_options]
    [-customScripts custom_sql_scripts_to_run_after_database_creation]
    [-policyManaged | -adminManaged]
    [-policyManaged
        -serverPoolName server_pool_names
        [-pqPoolName pq_pool_name]
        [-createServerPool new_server_pool_name]
            [-pqPoolName new_pq_pool_name]
            [-force]
            [-pqCardinality pq_cardinality_of_the_new_server_pool]
            [-cardinality cardinality_of_the_new_server_pool]]
    [-adminManaged]
    [-databaseConfigType {SINGLE | RAC | RACONENODE}
            [-RACOneNodeServiceName service_name_for_RAC_one_node_database]]
    [-characterSet database_character_set]
    [-nationalCharacterSet database_national_character_set]
    [-registerWithDirService {true | false}
        [-dirServiceUserName directory_service_user_name]
        [-dirServicePassword directory_service_user_password]
        [-databaseCN database_common_name]
        [-dirServiceCertificatePath certificate_file_path]
        [-dirServiceUser directory_service_user_name]
```

```
        [-ldapDirectoryAccessType ldap_directory_access_type]
        [-useSYSAuthForLDAPAccess use_sys_user_for_ldap_access_flag]
        [-walletPassword wallet_password]]
    [-listeners list_of_listeners_to_register_the_database_with]
    [-variablesFile variables_file]
    [-variables variables_list]
    [-initParams initialization_parameters_list
        [-initParamsEscapeChar initialization_parameters_escape_character]]
    [-sampleSchema {true | false}]
    [-memoryPercentage percentage_of_total_memory_to_assign_to_the_database]
    [-totalMemory total_memory_to_assign_to_the_database_in_MB]
    [-databaseType {MULTIPURPOSE | DATA_WAREHOUSING | OLTP}]
    [-useWalletForDBCredentials {true | false}
        -dbCredentialsWalletPassword wallet_account_password
        -dbCredentialsWalletLocation wallet_files_directory]
```

表2-12 generateScriptsのパラメータ

パラメータ	必須/オプション	説明
`-templateName database_template_name`	必須	デフォルトの場所にある既存のデータベース・テンプレートの名前、またはデフォルトの場所にないテンプレートへの完全パス
`-gdbName global_database_name`	必須	database_name.domain_name 形式のグローバル・データベース名。

`-sid`
 	
`database_system_identifier`
	オプション	データベース・システム識別子(SID)
		
		SID は、データベースを実行するインスタンスを一意に識別します。指定しないと、デフォルトでデータベース名に設定されます。

`-scriptDest scripts_directory`	オプション	スクリプトを格納する完全なディレクトリ・パス。

`-createAsContainerDatabase`
 	
`{true | false}`
	オプション	CDB を作成する場合は `true` を指定し、非 CDB を作成する場合は `false` を指定します。デフォルトは `false` です。
		
		`true` を指定した場合は、次のオプション・パラメータを指定できます。
		
		● `-numberOfPDBs`: 作成する PDB の数。デフォルトは(ゼロ)です。
		
		● `-pdbName`: 各 PDB の名前。`-numberOfPDBs` の値が 1 よりも大きい場合は、それぞれの PDB 名に番号が追加されます。このパラメータは、`-numberOfPDBs` 値が(ゼロ)より大きい場合に指定する必要があります。
		
		● `-pdbStorageMAXSizeInMB`: PDB の最大記憶域

パラメータ	必須/オプション	説明
		イズ(MB 単位)。 ● -pdbStorageMAXTempSizeInMB: PDB の最大一時記憶域サイズ(MB 単位)。 ● -useLocalUndoForPDBs {true | false}: PDB にローカル UNDO を使用する必要があるかどうかをすフラグ。 ● -pdbAdminPassword: PDB 管理者のパスワード。 ● -pdbOptions: カンマ区切りリストの形式での PDB のプション。各オプションは、name:value 形式で指定する必要があります。 例: JSERVER:true,DV:false
-sysPassword SYS_user_password	オプション	新しいデータベースの SYS ユーザー・パスワード。
-systemPassword SYSTEM_user_password	オプション	新しいデータベースの SYSTEM ユーザー・パスワード。
-emConfiguration {DBEXPRESS | CENTRAL | BOTH | NONE}	オプション	Enterprise Manager 構成の設定。 DBEXPRESS、CENTRAL または BOTH が指定されている場合は次の追加パラメータを指定します。 ● -dbsnmpPassword: DBSNMP ユーザー・パスワード ● -omsHost: Oracle Management Server のホスト名。 ● -omsPort: Oracle Management Server のポート番号。 ● -emUser: Enterprise Manager 管理者のユーザー名。 ● -emPassword: Enterprise Manager 管理者のパスワード。

パラメータ	必須/オプション	説明
		● -emExpressPort: Enterprise Manager Expres のポート番号。 ● -emExpressPortAsGlobalPort: Enterprise Manager Express のグローバル・ポート番号。
-dvConfiguration {true | false}	オプション	Database Vault を有効化および構成する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加の Database Vault パラメタが必要です。 ● -dvUserName: Database Vault 所有者の名前。 ● -dvUserPassword: Database Vault 所有者のパスワード。 ● -dvAccountManagerName: Database Vault アカウント・マネージャの名前。 ● -dvAccountManagerPassword: Database Vault アカウント・マネージャのパスワード。
-olsConfiguration {true | false}	オプション	Oracle Label Security (OLS)を有効化および構成する場合は true を指定し、それ以外の場合は false を指定しますデフォルトは false です。 true を指定した場合は、Oracle Internet Directory (OID) によって Oracle Label Security (OLS)を構成するための-configureWithOID パラメータを追加指定できます。このパラメータは省略可能です。
-datafileDestination data_files_directory	オプション	データベースのデータファイルの場所への完全なパス
-redoLogFileSize maximum_size_of_online_redo_log	オプション	各オンライン REDO ログ・ファイルのサイズ(MB 単位)。
-recoveryAreaDestination	オプション	バックアップおよびリカバリ領域である、高速リカバリ領域のディレクト

パラメータ	必須/オプション	説明
fast_recovery_area_directory		リ。高速リカバリ領域を無効にするには、NONE を指定します。 また、-recoveryAreaSize パラメータを使用すると、高速リカリ領域のサイズを MB 単位で指定できます。このパラメータは省略能です。
-datafileJarLocation data_files_backup_directory	オプション	RMAN バックアップの圧縮形式になっているデータベースのバックアップ・データ・ファイル(拡張子が.dfb のファイル)のディレクトリ。
-responseFile response_file_directory	オプション	レスポンス・ファイルのディレクトリ・パス。
-storageType {FS | ASM}	オプション	FS または ASM のどちらかの記憶域タイプを指定します。 ● FS: ファイル・システム記憶域タイプ。 FS が指定されている場合、データベース・ファイルは、使用しているオペレーティング・システムのファイル・システムによって管理されます。データベース・ファイルを格納するディレクトリパスは、データベース・テンプレートまたは-datafileDestination パラメータを使用して指定できます。Oracle Database は、実際のファイルを作成および管理できます。 ● ASM: Oracle Automatic Storage Management (Oracle ASM)の記憶域のタイプ。 ASM が指定されている場合、データベース・ファイルは Oracle ASM ディスク・グループに配置されます。データベース・ファイルの配置とネーミングは Oracle Database によって自動的に管理されます。 ASM を指定した場合は、-asmsnmpPassword パラメータを使用して ASMSNMP パスワードも指定できます。このパラメータは省略可能です。
-runCVUChecks {true | false}	オプション	Oracle RAC データベースに定期的なクラスタ検証ユーティリティ・チェックを実行する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。

パラメータ	必須/オプション	説明
-nodelist database_nodes_list	オプション	データベース・ノードのカンマ区切りのリスト。
-enableArchive {true | false}	オプション	ログ・ファイルのアーカイブを有効にする場合は true を指定し、そ 以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -archiveLogMode {AUTO | MANUAL}: 自動 アーカイブ・モードまたは手動アーカイブ・モードのどちらかを 定します。デフォルトは、自動アーカイブ・モードです。 ● -archiveLogDest: アーカイブ・ログ・ファイルを格納 る場合のディレクトリ。
-memoryMgmtType {AUTO | AUTO_SGA | CUSTOM_SGA}	オプション	次のいずれかのメモリー管理タイプを指定します。 ● AUTO: SGA および PGA の自動メモリー管理。 ● AUTO_SGA: SGA の自動共有メモリー管理。 ● CUSTOM_SGA: SGA の手動共有メモリー管理。 ノート: データベース・インスタンスの合計物理メモリーが 4GB より きい場合は、データベースのインストール時および作成時に自動メ リー管理オプション AUTO を指定できません。このような環境の場合 は、自動共有メモリー管理オプション AUTO_SGA を指定すること お薦めします。
-createListener new_database_listener	オプション	listener_name:port の形式で、データベースを登録するデータ ベース・リスナー。
-useOMF {true | false}	オプション	Oracle Managed Files (OMF)を使用する場合は true を指 定し、それ以外の場合は false を指定します。
-dbOptions database_options	オプション	name:value ペアのカンマ区切りリストとして、データベース・オプ ションを指定します。 例: JSERVER:true,DV:false

パラメータ	必須/オプション	説明

`-customScripts`
 	
`custom_sql_scripts_list`
	オプション	データベースの作成後に実行する必要のある SQL スクリプトのカン区切りリストを指定します。スクリプトはリストされている順序で実行れます。

`-policyManaged`	オプション	ポリシー管理型データベース。 次の追加パラメータを指定できます。 ● `-serverPoolName`: 新しいサーバー・プールの作成時には 1 つのサーバー・プール名を指定します。それ以外の場合は、既存のサーバー・プールのカンマ区切りリストを指定ます。 ● `-pqPoolName`: PQ プールの名前を指定します。 ● `-createServerPool`: 新しいサーバー・プールを作成する場合は、このパラメータを指定します。 　● `-pqPoolName`: PQ プールの名前を指定します。 　● `-force`: 適切な空きサーバーが使用できないときに強制的にサーバー・プールを作成する場合はこのパラメータを指定します。 　● `-pqCardinality`: 新しいサーバー・プールの PQ カーディナリティを指定します。 　● `-cardinality`: 新しいサーバー・プールのカーディナリティを指定します。
`-adminManaged`	オプション	管理者管理データベース。

`-databaseConfigType`
 	
`{SINGLE | RAC | RACONENODE}`
	オプション	次のいずれかのデータベース構成タイプを指定します。
		
		● `SINGLE`: 単一の個別のデータベース。
		
		● `RAC`: Oracle RAC データベース。
		
		● `RACONENODE`: Oracle RAC One Node データベース。

パラメータ	必須/オプション	説明
		Oracle RAC One Node データベースの場合は、-RACOneNodeServiceName パラメータを使用してサービス名を指定できます。
-characterSet database_character_set	オプション	データベースの文字セット
-nationalCharacterSet database_national_character_set	オプション	データベースの各国語文字セット。
-registerWithDirService {true | false}	オプション	Lightweight Directory Access Protocol (LDAP)サービスに登録する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータが必要です。 ● -dirServiceUserName: LDAP サービスのユーザー名。 ● -dirServicePassword: LDAP サービスのパスワー ● -databaseCN: データベースの共通名。 ● -dirServiceCertificatePath: ディレクトリ・サービスの証明書ファイルのパス。 ● -dirServiceUser: ディレクトリ・サービスのユーザー名。 ● -ldapDirectoryAccessType {PASSWORD | SSL}: LDAP ディレクトリのアクセス・タイプ。 ● -useSYSAuthForLDAPAccess {true | false}: LDAP アクセスに SYS ユーザー認証を使用するかどうかを指定します。 ● -walletPassword: データベース・ウォレットのパスワード
-listeners	オプション	データベースのリスナーのカンマ区切りリスト。

パラメータ	必須/オプション	説明
listeners_list		
-variablesFile variables_file	オプション	データベース・テンプレートの変数とその値が含まれているファイルへ ディレクトリ・パス。
-variables variables_list	オプション	データベース・テンプレートの変数に対応する name=value ペア カンマ区切りのリスト。
-initParams initialization_parameters_list	オプション	データベースの初期化パラメータ値の name=value ペアのカンマ 切りリスト。 初期化パラメータの複数の値の間に特定のエスケープ文字を使用 る場合は、-initParamsEscapeChar パラメータを追加指定 できます。エスケープ文字を指定しない場合は、デフォルトのエスケ プ文字としてバックスラッシュ(/)が使用されます。
-sampleSchema {true | false}	オプション	データベースに HR サンプル・スキーマ(EXAMPLE 表領域)を含める 場合は、true を指定します。それ以外の場合は false を指定 ます。デフォルトは false です。 オラクル社のガイドおよび入門資料には、サンプル・スキーマに基づ た例が含まれています。本番データベースにはサンプル・スキーマをイ ンストールしないことをお薦めします。
-memoryPercentage percentage_of_total_memory_assigned_to_the_database	オプション	データベースが使用できる物理メモリーの割合。
-totalMemory total_memory_assigned_to_the_database_in_MB	オプション	データベースが使用できる物理メモリーの合計量(MB 単位)。
-databaseType {MULTIPURPOSE | DATA_WAREHOUSING | OLTP}	オプション	データベースの目的が OLTP とデータ・ウェアハウスの両方である場 合は、MULTIPURPOSE を指定します。 データベースの主な目的がデータ・ウェアハウスである場合は、 DATA_WAREHOUSING を指定します。 データベースの主な目的がオンライン・トランザクション処理である場

パラメータ	必須/オプション	説明
		合は、OLTP を指定します。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は tru を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dbCredentialsWalletLocation: Oracle ウ レット・ファイルのディレクトリの場所。 ● -dbCredentialsWalletPassword: Oracle ウ レット・アカウントのパスワード。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次 キーを使用して OUD パスワードをウォレットに格納する必要があり す。 ● oracle.dbsecurity.walletPassword ● oracle.dbsecurity.userDNPassword

親トピック: DBCAサイレント・モードのコマンド

2.14.4.9 deleteDatabase

deleteDatabaseコマンドにより、データベースが削除されます。

構文およびパラメータ

次の構文でdbca -deleteDatabaseコマンドを使用します。

```
dbca -deleteDatabase
   -sourceDB database_name_or_sid
   [-sysDBAUserName SYSDBA_user_name]
   [-sysDBAPassword SYSDBA_user_password]
   [-forceArchiveLogDeletion]
   [-deRegisterEMCloudControl
     [-omsHost Oracle_Management_Server_host_name
      -omsPort Oracle_Management_Server_port_number
      -emUser EM_administrator_user_name
      -emPassword EM_administrator_password]]
   [-unregisterWithDirService {true | false}
      -dirServiceUserName directory_service_user_name
      [-dirServicePassword directory_service_user_password
      [-walletPassword wallet_password]]
   [-sid database_system_identifier]
   [-useWalletForDBCredentials {true | false}
       -dbCredentialsWalletPassword wallet_account_password
```

```
       -dbCredentialsWalletLocation wallet_files_directory]
```

表2-13 deleteDatabaseのパラメータ

パラメータ	必須/オプション	説明
`-sourceDB database_name_or_sid`	必須	Oracle RAC データベースの一意のデータベース名、または単一インスタンス・データベースのデータベース・システム識別子(SID)。
`-sysDBAUserName SYSDBA_user_name`	オプション	SYSDBA 権限を持つユーザーのユーザー名。
`-sysDBAPassword SYSDBA_password`	オプション	SYSDBA 権限を持つユーザーのパスワード。
`-forceArchiveLogDeletion`	オプション	データベース・アーカイブ・ログを削除する場合は、このパラメータを指定します。
`-deRegisterEMCloudControl`	オプション	このパラメータは、Enterprise Manager Cloud Control からデータベースの登録を解除するための次のパラメータとともに指定します。 ● `-omsHost`: Oracle Management Server のホスト名。 ● `-omsPort`: Oracle Management Server のポート番号。 ● `-emUser`: Enterprise Manager 管理者のユーザー名。 ● `-emPassword`: Enterprise Manager 管理者のパスワード。
`-unregisterWithDirService {true | false}`	オプション	このパラメータは、ディレクトリ・サービスからデータベースの登録を解除するための次のパラメータとともに指定します。 ● `-dirServiceUserName`: ディレクトリ・サービスのユーザー名。 ● `-dirServicePassword`: ディレクトリ・サービスのユーザー・パスワード。 ● `-walletPassword`: データベース・ウォレットのパスワード

パラメータ	必須/オプション	説明
`-sid database_system_identifier`	オプション	データベース・システム識別子(SID)
`-useWalletForDBCredentials {true | false}`	オプション	データベース資格証明に Oracle Wallet を使用する場合は `tru` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。 `true` を指定した場合は、次の追加パラメータを指定できます。 ● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。 ● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● `oracle.dbsecurity.walletPassword` ● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.10 createPluggableDatabase

`createPluggableDatabase`コマンドにより、マルチテナント・コンテナ・データベース(CDB)内にプラガブル・データベース(PDB)が作成されます。

構文およびパラメータ

次の構文で`dbca -createPluggableDatabase`コマンドを使用します。

```
dbca -createPluggableDatabase
   -sourceDB cdb_sid
   -pdbName name_of_the_pdb_to_create
   [-createAsClone {true | false}]
   [-createPDBFrom {DEFAULT | FILEARCHIVE | RMANBACKUP | USINGXML | PDB}
      [-pdbArchiveFile pdb_archive_file_name_with_directory_path]
      [-PDBBackUpfile pdb_backup_file_name_with_directory_path]
      [-PDBMetadataFile pdb_metadata_file_name_with_directory_path]
      [-pdbAdminUserName pdb_administrator_name]
      [-pdbAdminPassword pdb_administrator_password]
      [-createNewPDBAdminUser {true | false}]
      [-sourceFileNameConvert method_to_locate_pdb_files]
      [-fileNameConvert names_of_pdb_files]
      [-pdbStorageMAXSizeInMB maximum_storage_size_for_the_pdb_in_MB]
      [-pdbStorageMAXTempSizeInMB maximum_temporary_storage_size_for_the_pdb_in_MB]
      [-workArea directory_to_unzip_PDB_archive_files_for_FILEARCHIVE_option]
```

```
      [-copyPDBFiles  {true | false}]
      [-sourcePDB name_of_the_pdb_to_clone]
   [-createFromRemotePDB
      -pdbName name_of_the_local_pdb_to_create
      -sourceDB database_name_of_the_local_pdb
      -remotePDBName name_of_the_remote_pdb
      -remoteDBConnString db_connection_string_of_the_remote_pdb
      -sysDBAUserName name_of_the_sysdba_user
      -sysDBAPassword password_of_the_sysdba_user
      -dbLinkUsername name_of_the_dblink_user_of_the_remote_pdb
      -dbLinkUserPassword password_of_the_dblink_user_of_the_remote_pdb]
   [-pdbDatafileDestination pdb_data_files_directory]
   [-useMetaDataFileLocation {true | false}]
   [-registerWithDirService {true | false}
      -dirServiceUserName directory_service_user_name
      [-dirServicePassword directory_service_user_password]
      [-databaseCN directory_service_database_common_name]
      [-dirServiceCertificatePath certificate_file_directory_path]
      [-dirServiceUser active_directory_account_user_name]
      [-walletPassword wallet_password]]
   [-lbacsysPassword LBACSYS_user_password]
   [-createUserTableSpace {true | false)]
   [-pdbStorageMAXSizeInMB maximum_storage_size_for_the_pdb_in_MB]
   [-pdbStorageMAXTempSizeInMB maximum_temporary_storage_size_for_the_pdb_in_MB]
   [-customScripts custom_sql_scripts_to_run_after_PDB_creation]
   [-pdbUseMultipleBackup number_of_pdb_backups_to_create]
   [-dvConfiguration {true | false}
      -dvUserName Database_Vault_owner_name
      -dvUserPassword Database_Vault_owner_password
      [-dvAccountManagerName Database_Vault_account_manager_name]
      [-dvAccountManagerPassword Database_Vault_account_manager_password]]
   [-useWalletForDBCredentials {true | false}
      -dbCredentialsWalletPassword wallet_account_password
      -dbCredentialsWalletLocation wallet_files_directory]
```

表2-14 createPluggableDatabaseのパラメータ

パラメータ	必須/オプション	説明
-sourceDB cdb_sid	必須	CDB のデータベース・システム識別子(SID)。
-pdbName name_of_the_pdb_to_create	必須	新しい PDB の名前。 ノート: Oracle RAC データベースの場合、PDB 名はクラスタ内で 一意にする必要があります。
-createAsClone {true | false}	オプション	新規 PDB の作成に使用する予定のファイルが、既存の PDB の作 成に使用されたファイルと同じ場合は、true を指定します。true を指定すると、Oracle Database は一意の PDB DBID、GUI および新規 PDB で予期されるその他の識別子を生成します。 新規 PDB の作成に使用する予定のファイルが、既存の PDB の作 成に使用されたファイルと同じでない場合は、false (デフォルト)

パラメータ	必須/オプション	説明
		指定します。
-createPDBFrom {DEFAULT | FILEARCHIVE | RMANBACKUP | USINGXML | PDB}	オプション	CDB のシードから PDB を作成する場合は DEFAULT を指定します。DEFAULT を指定する場合は、次の追加パラメータが必要です。 ● -pdbAdminUserName: PDB のローカル管理者のユーザー名。 ● -pdbAdminPassword: PDB のローカル管理者のパスワード 切断された PDB のファイルから PDB を作成する場合は、FILEARCHIVE を指定します。FILEARCHIVE を指定する場合は、次のパラメータがさらに必要です。 ● -pdbArchiveFile: 切断された PDB のアーカイブ・ファイルの完全パスおよび名前 アーカイブ・ファイルには、PDB の XML メタデータ・ファイル、データファイルなど、PDB のすべてのファイルが含まれています。通常、アーカイブ・ファイルには .gz の拡張子が付いています。 ● -createNewPDBAdminUser: 新しい PDB 管理者を作成する場合は true を指定し、新しい PDB 管理者を作成しない場合は false を指定します。 ● -workArea: PDB アーカイブ・ファイルの解凍が必要なディレクトリの場所を指定します。 Recovery Manager (RMAN)バックアップから PDB を作成する場合は、RMANBACKUP を指定します。RMANBACKUP を指定する場合は、次の追加パラメータが必要です。 ● -pdbBackUpfile: PDB バックアップ・ファイルの完全なパスと名前。 ● -pdbMetadataFile: PDB の XML メタデータ・ファイルの完全なパスと名前。 切断された PDB の XML メタデータ・ファイルから PDB を作成する

パラメータ	必須/オプション	説明
		場合は、USINGXML を指定します。USINGXML を指定する場合は、次の追加パラメータが必要です。 ● -pdbMetadataFile: PDB の XML メタデータ・ファイの完全なパスと名前。 既存の PDB をクローニングして新しい PDB を作成する場合は PDB を指定します。PDB を指定する場合は、次の追加パラメータ必要になります。 ● -sourcePDB: クローニングする既存の PDB の名前。 次のオプション・パラメータを指定します(必要な場合)。 ● -sourceFileNameConvert: このパラメータでは、PDB XML メタデータ・ファイルにリストされた PDB のファイの検索方法を指定します。 『Oracle Multitenant 管理者ガイド』で説明されている CREATE PLUGGABLE DATABASE 文の SOURCE_FILE_NAME_CONVERT 句を参照してください。 ● -fileNameConvert: このパラメータでは、PDB のファイルの名前を指定します。 『Oracle Multitenant 管理者ガイド』で説明されている CREATE PLUGGABLE DATABASE 文の FILE_NAME_CONVERT 句を参照してください。 ● -pdbStorageMAXSizeInMB: PDB の最大記憶域サイズ(MB 単位)を指定します。 『Oracle Multitenant 管理者ガイド』で説明されている PDB の記憶域に関する情報を参照してください。 ● -pdbStorageMAXTempSizeInMB: PDB の最大一時記憶域サイズ(MB 単位)を指定します。 ● -copyPDBFiles {true | false}: PDB データファイルをコピーする必要がある場合は true を指定し、それ以外の場合は false を指定します。

パラメータ	必須/オプション	説明
-createFromRemotePDB	オプション	リモート PDB のクローニングによって PDB を作成します。 次のパラメータを指定します。 ● -pdbName: 作成するローカル PDB の名前。 ● -sourceDB: ローカル PDB のデータベース名。 ● -remotePDBName: クローニングするリモート PDB の名前。 ● -remoteDBConnString: リモート PDB のデータベース接続文字列。 ● -sysDBAUserName: SYSDBA ユーザーの名前。 ● -sysDBAPassword: SYSDBA ユーザーのパスワード ● -dbLinkUsername: リモート PDB のデータベース・リンク・ユーザーの名前。 ● -dbLinkUserPassword: リモート PDB のデータベース・リンク・ユーザーのパスワード。 ノート: ● ローカル CDB のデータベース・ユーザーには、ルート・コンテナの CREATE PLUGGABLE DATABASE 権限が必要です。 ● リモート CDB は、ローカル UNDO モードである必要があります。 ● リモート PDB は、ARCHIVELOG モードである必要があります。 ● データベース・リンクの接続先のリモート PDB のデータベース・ユーザーには、CREATE PLUGGABLE DATABASE 権限と CREATE SESSION 権限が必要です。
-pdbDatafileDestination pdb_data_files_directory	オプション	新しい PDB データ・ファイルへの完全なディレクトリ・パス。 パラメータが指定されていない場合は、Oracle Managed Files

パラメータ	必須/オプション	説明
		たは PDB_FILE_NAME_CONVERT 初期化パラメータで、ファイルの名前と場所の生成方法を指定します。Oracle Managed Files および PDB_FILE_NAME_CONVERT 初期化パラメータの両方を使用する場合は、Oracle Managed Files が優先されます。 パラメータが指定されていない場合は、Oracle Managed Files が有効になっておらず、PDB_FILE_NAME_CONVERT 初期化パラメータは設定されていません。デフォルトで、ルートのファイルのディレクトリ内の PDB 名を持つサブディレクトリへのパスが使用されます。
-useMetaDataFileLocation {true | false}	オプション	データファイルの抽出時に、PDB アーカイブ内の XML メタデータ・ファイルで定義されているデータファイルのパスを使用する場合は、true を指定します。 データファイルの抽出時に、PDB アーカイブ内の XML メタデータ・ファイルで定義されているデータファイルのパスを使用しない場合は、false (デフォルト)を指定します。
-registerWithDirService {true | false}	オプション	Lightweight Directory Access Protocol (LDAP)サービスに PDB を登録する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータが必要です。 ● -dirServiceUserName: LDAP サービスのユーザー名。 ● -dirServicePassword: LDAP サービスのパスワード ● -dirServiceUser: Active Directory アカウントのユーザー名。 ● -dirServiceCertificatePath: ディレクトリ・サービスの証明書ファイルのパス。 ● -databaseCN: ディレクトリ・サービス・データベースの共通名。 ● -walletPassword: データベース・ウォレットのパスワード

パラメータ	必須/オプション	説明

`-lbacsysPassword`
 	
`LBACSYS_user_password`
	オプション	ディレクトリ・サービスによって OLS を構成する必要がある場合は、LBACSYS ユーザー・パスワードを指定します。

`-createUserTableSpace`
 	
`{true | false}`
	オプション	新しい PDB にデフォルトのユーザー表領域を作成する必要がある場合は `true` を指定します。

`-pdbStorageMAXSizeInMB`
 	
`maximum_storage_size_for_the_pdb_in_MB`
	オプション	PDB の最大記憶域サイズ(MB 単位)を指定します。

`-pdbStorageMAXTempSizeInMB`
 	
`maximum_temporary_storage_size_for_the_pdb_in_MB`
	オプション	PDB の最大一時記憶域サイズ(MB 単位)を指定します。

`-pdbUseMultipleBackup`
 	
`number_of_pdb_backups_to_create`
	オプション	作成する PDB バックアップの数を指定します。

`-customScripts`
 	
`lcustom_sql_scripts_to_run_after_PDB_creation`
	オプション	PDB の作成後に実行するカスタム SQL スクリプトのリストを指定します。

`-dvConfiguration`
 	
`{true | false}`
	オプション	Database Vault を有効化および構成する場合は `true` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。
		
		`true` を指定した場合は、次の追加の Database Vault パラメータが必要です。
		
		● `-dvUserName`: Database Vault 所有者の名前を指定します。
		
		● `-dvUserPassword`: Database Vault 所有者のパスワードを指定します。
		
		● `-dvAccountManagerName`: 個別の Database Vault アカウント・マネージャの名前を指定します。
		
		● `-dvAccountManagerPassword`: Database Vault アカウント・マネージャのパスワードを指定します。

パラメータ	必須/オプション	説明

`-useWalletForDBCredentials`
 	
`{true | false}`
	オプション	データベース資格証明に Oracle Wallet を使用する場合は `tru` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。
		
		`true` を指定した場合は、次の追加パラメータを指定できます。
		
		● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。
		
		● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。
		
		ノート:
		
		Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。
		
		● `oracle.dbsecurity.walletPassword`
		
		● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.11 unplugDatabase

`unplugDatabase`コマンドにより、マルチテナント・コンテナ・データベース(CDB)からプラガブル・データベース(PDB)が切断されます。

構文およびパラメータ

次の構文で`dbca -unplugDatabase`コマンドを使用します。

```
dbca -unplugDatabase
   -sourceDB cdb_sid
   -pdbName pdb_name
   [-unregisterWithDirService {true | false}
      -dirServiceUserName directory_service_user_name
      -dirServicePassword directory_service_user_password
      -walletPassword wallet_password]
   [-archiveType {TAR | RMAN | NONE}
      [-rmanParallelism parallelism_integer_value]
      [-pdbArchiveFile pdb_archive_file_directory]
      [-PDBBackUpfile pdb_backup_file_directory]
      [-PDBMetadataFile pdb_metadata_file_directory]
      [-rmanParallelism parallelism_integer_value]]
   [-useWalletForDBCredentials {true | false}
      -dbCredentialsWalletPassword wallet_account_password
      -dbCredentialsWalletLocation wallet_files_directory]
```

表2-15 unplugDatabaseのパラメータ

パラメータ	必須/オプション	説明
-sourceDB cdb_sid	必須	CDB のデータベース・システム識別子(SID)。
-pdbName pdb_name	必須	PDB 名。
-archiveType {TAR | RMAN | NONE}	オプション	切断された PDB のファイルを tar ファイルに格納する場合は、TAR を指定します。 切断された PDB のファイルを RMAN バックアップに格納する場合は、RMAN を指定します。 tar ファイルまたは RMAN バックアップを使用しないで、切断された PDB のファイルを格納する場合は NONE を指定します。 次のいずれかのパラメータを指定します。 ● -pdbArchiveFile: PDB アーカイブ・ファイルの絶対ファイル・パスと名前を指定します。 ● -pdbBackUpfile: アーカイブ・タイプが RMAN の場合は PDB バックアップ・ファイルの絶対ファイル・パスと名前を定します。PDB の作成時に作成されるバックアップが複数存在する場合は、カンマ区切りのファイル・パスを指定します。 ● -pdbMetadataFile: アーカイブ・タイプが RMAN または NONE の場合は、PDB メタデータ・ファイルの絶対ファイル・パスと名前を指定します。 ● -rmanParallelism: RMAN 並列度の整数値を指定します。
-unregisterWithDirService {true | false}	オプション	LDAP サービスから PDB の登録を解除する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータが必要です。 ● -dirServiceUserName: LDAP サービスのユーザー名。 ● -dirServicePassword: LDAP サービス・ユーザー

パラメータ	必須/オプション	説明
		パスワード。 ● `-walletPassword`: データベース・ウォレットのパスワード

`-useWalletForDBCredentials`
 	
`{true | false}`
	オプション	データベース資格証明に Oracle Wallet を使用する場合は `tru` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。
		
		`true` を指定した場合は、次の追加パラメータを指定できます。
		
		● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。
		
		● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。
		
		ノート:
		
		Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。
		
		● `oracle.dbsecurity.walletPassword`
		
		● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.12 deletePluggableDatabase

`deletePluggableDatabase`コマンドにより、PDBが削除されます。

構文およびパラメータ

次の構文で`dbca -deletePluggableDatabase`コマンドを使用します。

```
dbca -deletePluggableDatabase
   -sourceDB cdb_sid
   -pdbName pdb_name
   [-useWalletForDBCredentials {true | false}
       -dbCredentialsWalletPassword wallet_account_password
       -dbCredentialsWalletLocation wallet_files_directory]
```

表2-16 deletePluggableDatabaseのパラメータ

パラメータ	必須/オプション	説明
`-sourceDB cdb_sid`	必須	CDB のデータベース・システム識別子(SID)。
`-pdbName pdb_name`	必須	削除する PDB の名前。
`-useWalletForDBCredentials {true | false}`	オプション	データベース資格証明に Oracle Wallet を使用する場合は `tru` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。 `true` を指定した場合は、次の追加パラメータを指定できます。 ● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。 ● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● `oracle.dbsecurity.walletPassword` ● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.13 relocatePDB

`relocatePDB`コマンドは、リモートCDBからローカルCDBにPDBを再配置します。

前提条件

次に、`relocatePDB`コマンドを実行するための前提条件を示します。

- ローカルPDB内のデータベース・ユーザーには、ローカルCDBルート・コンテナでの`CREATE PLUGGABLE DATABASE`権限が必要です。
- リモートCDBは、ローカルUNDOモードである必要があります。
- リモートPDBとローカルPDBは、ARCHIVELOGモードである必要があります。
- データベース・リンクの接続先のリモートPDB内のデータベース・ユーザーには、`CREATE PLUGGABLE DATABASE`、`SESSION`および`SYSOPER`権限が必要です。
- ローカルPDBとリモートPDBに同じオプションがインストールされているか、リモートPDBに、ローカルPDBにインストールされているオプションのサブセットがある必要があります。

構文およびパラメータ

次の構文でdbca -relocatePDBコマンドを使用します。

```
dbca -relocatePDB
   -pdbName name_of_the_local_pdb_to_create
   -sourceDB database_name_of_the_local_pdb
   -remotePDBName name_of_the_remote_pdb_to_relocate
   -remoteDBConnString db_connection_string_of_the_remote_pdb
   -sysDBAUserName name_of_the_sysdba_user
   -sysDBAPassword password_of_the_sysdba_user
   -dbLinkUsername name_of_the_dblink_user_of_the_remote_pdb
   -dbLinkUserPassword password_of_the_dblink_user_of_the_remote_pdb
```

表2-17 relocatePDBのパラメータ

パラメータ	必須/オプション	説明
-pdbName name_of_the_local_pdb_to_create	必須	リモート PDB の再配置後に作成するローカル PDB の名前。
-sourceDB database_name_of_the_local_pdb	必須	ローカル PDB のデータベース名。
-remotePDBName name_of_the_remote_pdb_to_relocate	必須	再配置するリモート PDB の名前。
-remoteDBConnString db_connection_string_of_the_remote_pdb	必須	リモート PDB のデータベース接続文字列。
-sysDBAUserName name_of_the_sysdba_user	必須	SYSDBA ユーザーの名前。
-sysDBAPassword password_of_the_sysdba_user	必須	SYSDBA ユーザーのパスワード。
-dbLinkUsername name_of_the_dblink_user_of_the_remote_pdb	必須	リモート PDB のデータベース・リンク・ユーザーの名前。
-dbLinkUserPassword password_of_the_dblink_user_of_the_remote_pdb	必須	リモート PDB のデータベース・リンク・ユーザーのパスワード。

親トピック: DBCAサイレント・モードのコマンド

2.14.4.14 configurePluggableDatabase

configurePluggableDatabaseコマンドにより、プラガブル・データベース(PDB)が構成されます。

構文およびパラメータ

次の構文でdbca -configurePluggableDatabaseコマンドを使用します。

```
dbca -configurePluggableDatabase
   -sourceDB cdb_sid
   -pdbName pdb_name
```

```
  [-dvConfiguration {true | false}
     -dvUserName Database_Vault_owner_name
     -dvUserPassword Database_Vault_owner_password
     [-dvAccountManagerName Database_Vault_account_manager_name]
     [-dvAccountManagerPassword Database_Vault_account_manager_password]]
  [-olsConfiguration {true | false}
     [-configureWithOID configure_with_OID_flag]]
  [-configureOracleR
     [-oracleRConfigTablespace tablespace_for_Oracle_R_configuration]]
  [-registerWithDirService {true | false}
     -dirServiceUserName directory_service_user_name
     [-dirServicePassword directory_service_user_password]
     [-walletPassword wallet_password]
     [-databaseCN database_common_name]
     [-dirServiceCertificatePath certificate_file_path]
     [-dirServiceUser active_directory_account_user_name]]
  [-unregisterWithDirService {true | false}
     -dirServiceUserName directory_service_user_name
     [-dirServicePassword directory_service_user_password]
     [-walletPassword wallet_password]]
  [-lbacsysPassword LBACSYS_user_password]]
  [-useWalletForDBCredentials {true | false}
     -dbCredentialsWalletPassword wallet_account_password
     -dbCredentialsWalletLocation wallet_files_directory]
```

表2-18 configurePluggableDatabaseのパラメータ

パラメータ	必須/オプション	説明
`-sourceDB cdb_sid`	必須	CDB のデータベース・システム識別子(SID)。
`-pdbName pdb_name`	必須	PDB 名。
`-dvConfiguration {true | false}`	オプション	PDB の Database Vault を有効化および構成する場合は `tru` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。 `true` を指定した場合は、次の追加の Database Vault パラメータが必要です。 ● `-dvUserName`: Database Vault 所有者のユーザー名を指定します。 ● `-dvUserPassword`: Database Vault 所有者のパスワードを指定します。 ● `-dvAccountManagerName`: 個別の Database Vault アカウント・マネージャを指定します。 ● `-dvAccountManagerPassword`: Database Vault アカウント・マネージャのパスワードを指定します。

パラメータ	必須/オプション	説明
-olsConfiguration {true | false}	オプション	PDB の Oracle Label Security (OLS)を有効化および構成する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、Oracle Internet Directory (OID)によって Oracle Label Security (OLS)を構成するための-configureWithOID パラメータを追加指定できます。このパラメータは省略可能です。
-lbacsysPassword	オプション	ディレクトリ・サービスによって OLS を構成する必要がある場合は、LBACSYS ユーザー・パスワードを指定します。
-configureOracleR	オプション	PDB の Oracle R を構成する場合は、このパラメータを指定します。 また、Oracle R 構成用の表領域(たとえば、SYSAUX 表領域)を割り当てるために、-oracleRConfigTablespace パラメータも指定できます。
-registerWithDirService{true | false}	オプション	Lightweight Directory Access Protocol (LDAP)サービスに PDB を登録する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dirServiceUserName: LDAP サービスのユーザー名。 ● -dirServicePassword: LDAP サービス・ユーザーのパスワード。 ● -walletPassword: データベース・ウォレットのパスワード ● -databaseCN: データベースの共通名。 ● -dirServiceCertificatePath: ディレクトリ・サービスの証明書ファイルのパス。 ● -dirServiceUser: Active Directory アカウントのユーザー名。

パラメータ	必須/オプション	説明
`unregisterWithDirService {true | false}`	オプション	Lightweight Directory Access Protocol (LDAP)サービスから PDB の登録を解除する場合は `true` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。 `true` を指定した場合は、次の追加パラメータを指定できます。 ● `-dirServiceUserName`: LDAP サービスのユーザー名。 ● `-dirServicePassword`: LDAP サービス・ユーザーパスワード。 ● `-walletPassword`: データベース・ウォレットのパスワード
`-useWalletForDBCredentials {true | false}`	オプション	データベース資格証明に Oracle Wallet を使用する場合は `tru` を指定し、それ以外の場合は `false` を指定します。デフォルトは `false` です。 `true` を指定した場合は、次の追加パラメータを指定できます。 ● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。 ● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次キーを使用して OUD パスワードをウォレットに格納する必要があります。 ● `oracle.dbsecurity.walletPassword` ● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.15 addInstance

`addInstance`コマンドにより、データベース・インスタンスが管理者管理Oracle RACデータベースに追加されます。

構文およびパラメータ

次の構文でdbca -addInstanceコマンドを使用します。

```
dbca -addInstance
   -gdbName global_database_name
   -nodeName database_instance_node_name
   [-updateDirService {true | false}
      -dirServiceUserName directory_service_user_name
      -dirServicePassword directory_service_user_password]
   [-instanceName database_instance_name]
   [-sysDBAUserName SYSDBA_user_name]
   [-sysDBAPassword SYSDBA_user_password]
   [-useWalletForDBCredentials {true | false}
      -dbCredentialsWalletPassword wallet_account_password
      -dbCredentialsWalletLocation wallet_files_directory]
```

表2-19 addInstanceのパラメータ

パラメータ	必須/オプション	説明
-gdbName global_database_name	必須	database_name.domain_name 形式のグローバル・データベース名。
-nodeName database_instance_node_name	必須	データベース・インスタンスのノード名。
-instanceName database_instance_name	オプション	データベース・インスタンス名。
-sysDBAUserName SYSDBA_user_name	オプション	SYSDBA 権限を持つデータベース・ユーザーのユーザー名。
-sysDBAPassword SYSDBA_user_password	オプション	SYSDBA 権限を持つデータベース・ユーザーのパスワード。
-updateDirService {true | false}	オプション	ディレクトリ・サービスにデータベースを登録する場合は true を指定し、それ以外の場合は false を指定します。 true を指定した場合は、次の追加パラメータが必要です。 ● -dirServiceUserName: ディレクトリ・サービスのユーザー名。 ● –dirServicePassword: ディレクトリ・サービス・ユーザーのパスワード。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。

パラメータ	必須/オプション	説明
		`true` を指定した場合は、次の追加パラメータを指定できます。 ● `-dbCredentialsWalletPassword`: Oracle ウォレット・アカウントのパスワード。 ● `-dbCredentialsWalletLocation`: Oracle ウォレット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● `oracle.dbsecurity.walletPassword` ● `oracle.dbsecurity.userDNPassword`

親トピック: DBCAサイレント・モードのコマンド

2.14.4.16 deleteInstance

`deleteInstance`コマンドにより、データベース・インスタンスが管理者管理Oracle RACデータベースから削除されます。

構文およびパラメータ

次の構文で`dbca -deleteInstance`コマンドを使用します。

```
dbca -deleteInstance
   -gdbName global_database_name
   -instanceName database_instance_name
   [-nodeName database_instance_node_name]
   [-updateDirService {true | false}
      -dirServiceUserName directory_service_user_name
      -dirServicePassword directory_service_user_password]
   [-sysDBAUserName SYSDBA_user_name]
   [-sysDBAPassword SYSDBA_user_password]
   [-useWalletForDBCredentials {true | false}
      -dbCredentialsWalletPassword wallet_account_password
      -dbCredentialsWalletLocation wallet_files_directory]
```

表2-20 deleteInstanceのパラメータ

パラメータ	必須/オプション	説明
`-gdbName global_database_name`	必須	database_name.domain_name 形式のグローバル・データベース名。

パラメータ	必須/オプション	説明
-instanceName database_instance_name	必須	データベース・インスタンス名。
-nodeName node_name_of_database_instance	オプション	データベース・インスタンスのノード名。
-sysDBAUserName SYSDBA_user_name	オプション	SYSDBA 権限を持つデータベース・ユーザーのユーザー名。
-sysDBAPassword SYSDBA_user_password	オプション	SYSDBA 権限を持つデータベース・ユーザーのパスワード。
-updateDirService {true | false}	オプション	ディレクトリ・サービスからデータベースの登録を解除する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータが必要です。 ● -dirServiceUserName: ディレクトリ・サービスのユーザー名。 ● –dirServicePassword: ディレクトリ・サービス・ユーザーのパスワード。
-useWalletForDBCredentials {true | false}	オプション	データベース資格証明に Oracle Wallet を使用する場合は true を指定し、それ以外の場合は false を指定します。デフォルトは false です。 true を指定した場合は、次の追加パラメータを指定できます。 ● -dbCredentialsWalletPassword: Oracle ウォレット・アカウントのパスワード。 ● -dbCredentialsWalletLocation: Oracle ウォレット・ファイルのディレクトリの場所。 ノート: Oracle Unified Directory (OUD)を使用している場合は、次のキーを使用して OUD パスワードをウォレットに格納する必要があります。 ● oracle.dbsecurity.walletPassword

パラメータ	必須/オプション	説明
		• oracle.dbsecurity.userDNPassword

親トピック: DBCAサイレント・モードのコマンド

2.14.4.17 executePrereqs

executePrereqsコマンドにより、前提条件チェックが実行され、その結果が報告されます。このコマンドは、データベースを作成するdbcaの実行前に環境をチェックするために使用できます。

構文およびパラメータ

次の構文でdbca -executePrereqsコマンドを使用します。

```
dbca -executePrereqs
   -databaseConfigType {SINGLE | RAC | RACONENODE}
        [-RACOneNodeServiceName RAC_node_service_name]
   [-nodelist database_nodes_list]
```

表2-21 executePrereqsのパラメータ

パラメータ	必須/オプション	説明
-databaseConfigType {SINGLE | RAC | RACONENODE}	必須	次のいずれかのデータベース構成タイプを指定します。 • SINGLE: 単一の個別のデータベース。 • RAC: Oracle RAC データベース。 • RACONENODE: Oracle RAC One Node データベース。 Oracle RAC One Node データベースの場合は、-RACOneNodeServiceName パラメータを使用してサービス名を指定できます。
-nodelist database_nodes_list	オプション	データベース・ノードのカンマ区切りのリスト。

親トピック: DBCAサイレント・モードのコマンド

2.14.5 DBCAの終了コード

サイレント・モードでのDBCAコマンドの実行の結果は、終了コードとしてレポートされます。

次の表では、DBCAからオペレーティング・システムに返される終了コードを示します。

表2-22 Database Configuration Assistantの終了コード

終了コード	説明
0	コマンドの実行に成功しました
6	コマンドの実行に成功しましたが、警告が発生しました
-1	コマンドの実行に失敗しました
-2	ユーザーからの入力が無効でした
-4	コマンドがユーザーに取り消されました

親トピック: サイレント・モード時のDatabase Configuration Assistantコマンド・リファレンス

3 起動と停止

データベースの起動時には、そのデータベースのインスタンスを作成し、データベースの状態を確認します。現在実行中のOracle Databaseインスタンスの停止では、必要に応じて、データベースをクローズおよびディスマウントできます。

- データベースの起動
  データベースの起動時には、そのデータベースのインスタンスを作成し、データベースの状態を確認します。インスタンスは通常、データベースをマウントおよびオープンすることで起動します。これによって、有効なユーザーはデータベースに接続して通常のデータ・アクセス操作を実行できます。
- データベースの可用性の変更
  データベースの可用性を変更できます。可用性の変更は、メンテナンス上の理由やデータベースを読取り専用にするために行う場合があります。
- データベースの停止
  SQL*PlusまたはOracle Restartでデータベースを停止できます。
- データベースの静止
  データベースの静止中は、DBAによるトランザクション、問合せ、フェッチまたはPL/SQL文のみ実行できます。
- データベースの一時停止と再開
  `ALTER SYSTEM SUSPEND`文は、データファイル(ファイル・ヘッダーとファイル・データ)および制御ファイルへの入出力(I/O)をすべて停止します。一時停止状態によって、I/Oに干渉されずにデータベースのバックアップを作成できます。データベースを一時停止すると、実行中のすべてのI/O操作の完了が許可され、新しいデータベース・アクセスはキューに待機した状態になります。通常のデータベース操作を再開するには、`ALTER SYSTEM RESUME`文を使用します。
- インスタンス中断の遅延
  `INSTANCE_ABORT_DELAY_TIME`初期化パラメータを使用して、エラーによりインスタンスが中断したときに、データベースの停止を遅延する時間を秒単位で指定します。

関連項目:

Oracle Real Application Clusters環境に固有の追加情報は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

親トピック: 基本データベース管理

3.1 データベースの起動

データベースの起動時には、そのデータベースのインスタンスを作成し、データベースの状態を確認します。インスタンスは通常、データベースをマウントおよびオープンすることで起動します。これによって、有効なユーザーはデータベースに接続して通常のデータ・アクセス操作を実行できます。

- データベースの起動オプションについて
  Oracle Restartを使用していない場合は、SQL*Plus、Recovery ManagerまたはOracle Enterprise Manager Cloud Control (Cloud Control)を使用してデータベース・インスタンスを起動できます。Oracle Restartでデータベースを管理している場合は、SRVCTLを使用してデータベースを起動する方法をお薦めします。
- 起動時における初期化パラメータの指定
  データベースは、インスタンスを起動するために、サーバー・パラメータ・ファイル(`SPFILE`)またはテキスト形式の初期化パラメータ・ファイル(PFILE)からインスタンス構成パラメータ(初期化パラメータ)を読み込む必要があります。データベー

スは、デフォルトの場所でこれらのファイルを検索します。これらのファイルにデフォルト以外の場所を指定することも可能で、その方法は、データベースの起動にSQL*Plusを使用するか(Oracle Restartを使用していない場合)、SRVCTLを使用するか(データベースをOracle Restartで管理している場合)によって異なります。

- データベース・サービスの自動起動について
  データベースがOracle Restartで管理されている場合は、個々のデータベース・サービス(サービス)ごとに起動オプションを構成できます。
- インスタンス起動の準備
  SQL*Plusを使用してデータベース・インスタンスを起動する前に、いくつかの準備ステップを実行する必要があります。
- インスタンスの起動
  SQL*PlusまたはOracle Restartを使用してインスタンスを起動できます。

親トピック: 起動と停止

3.1.1 データベースの起動オプションについて

Oracle Restartを使用していない場合は、SQL*Plus、Recovery ManagerまたはOracle Enterprise Manager Cloud Control (Cloud Control)を使用してデータベース・インスタンスを起動できます。Oracle Restartでデータベースを管理している場合は、SRVCTLを使用してデータベースを起動する方法をお薦めします。

Oracle Restartの詳細は、「Oracle Databaseの自動再起動の構成」を参照してください。

- SQL*Plusを使用したデータベースの起動
  SQL*Plusセッションを開始し、管理者権限でOracle Databaseに接続すると、STARTUPコマンドを発行できます。このマニュアルでは、SQL*Plusを使用するこの方法について詳細に説明します。
- Recovery Managerを使用したデータベースの起動
  Recovery Manager (RMAN)を使用して、STARTUPおよびSHUTDOWNコマンドを実行する方法もあります。この方法を選択するのは、RMAN環境でSQL*Plusを起動しない場合です。
- Cloud Controlを使用したデータベースの起動
  Cloud Controlは、起動や停止も含めたデータベースの管理に使用できます。Cloud Controlは、GUIコンソール、エージェント、共有サービスおよびOracleのツール製品を組み合せて、Oracle製品の管理のために統合された包括的なシステム管理プラットフォームを提供します。Cloud Controlを使用すると、コマンドライン操作のかわりにGUIインタフェースを使用して、このマニュアルで説明されている機能を実行できます。
- SRVCTLを使用したデータベースの起動
  Oracle Restartがインストールされてデータベース用に構成されている場合は、SRVCTLを使用してデータベースを起動することをお薦めします。

親トピック: データベースの起動

3.1.1.1 SQL*Plusを使用したデータベースの起動

SQL*Plusセッションを開始し、管理者権限でOracle Databaseに接続すると、STARTUPコマンドを発行できます。このマニュアルでは、SQL*Plusを使用するこの方法について詳細に説明します。

- SQL*PlusのSTARTUPコマンドを実行します。

関連トピック

- SQL*Plusユーザーズ・ガイドおよびリファレンス

親トピック: データベースの起動オプションについて

3.1.1.2 Recovery Managerを使用したデータベースの起動

Recovery Manager(RMAN)を使用して、STARTUPおよびSHUTDOWNコマンドを実行する方法もあります。この方法を選択するのは、RMAN環境でSQL*Plusを起動しない場合です。

- RMANのSTARTUPコマンドを実行します。

関連項目:

RMAN STARTUPコマンドの詳細は、『Oracle Databaseバックアップおよびリカバリ・リファレンス』を参照してください。

親トピック: データベースの起動オプションについて

3.1.1.3 Cloud Controlを使用したデータベースの起動

Cloud Controlは、起動や停止も含めたデータベースの管理に使用できます。Cloud Controlは、GUIコンソール、エージェント、共有サービスおよびOracleのツール製品を組み合せて、Oracle製品の管理のために統合された包括的なシステム管理プラットフォームを提供します。Cloud Controlを使用すると、コマンドライン操作のかわりにGUIインタフェースを使用して、このマニュアルで説明されている機能を実行できます。

- Cloud Controlでデータベース・インスタンスを起動します。

関連項目:

Cloud Controlのオンライン・ヘルプ

親トピック: データベースの起動オプションについて

3.1.1.4 SRVCTLを使用したデータベースの起動

Oracle Restartがインストールされてデータベース用に構成されている場合は、SRVCTLを使用してデータベースを起動することをお薦めします。

SRVCTL使用してデータベース・インスタンスを起動すると、次の項目が保証されます。

- データベースが依存する任意のコンポーネント(Oracle Automatic Storage ManagementやOracle Netリスナーなど)が、適切な順序で最初に自動起動します。
- データベースはOracle Restart構成の設定に従って起動します。そのような設定の例は、サーバー・パラメータ・ファイルの場所です。
- データベースのOracle Restart構成に格納された環境変数は、インスタンス起動前に設定されます。

SRVCTLを使用してデータベース・インスタンスを起動するには:

- `srvctl start database`コマンドを実行します。

詳細は、「srvctl start database」および「Oracle Restartで管理されているコンポーネントの起動と停止」を参照してください。

親トピック: データベースの起動オプションについて

3.1.2 起動時における初期化パラメータの指定

データベースは、インスタンスを起動するために、サーバー・パラメータ・ファイル(SPFILE)またはテキスト形式の初期化パラメータ・ファイル(PFILE)からインスタンス構成パラメータ(初期化パラメータ)を読み込む必要があります。データベースは、デフォルトの場所でこれらのファイルを検索します。これらのファイルにデフォルト以外の場所を指定することも可能で、その方法は、データベースの起動にSQL*Plusを使用するか(Oracle Restartを使用していない場合)、SRVCTLを使用するか(データベースをOracle Restartで管理している場合)によって異なります。

- 初期化パラメータ・ファイルおよび起動について
  データベース・インスタンスを起動すると、データベースは、プラットフォーム固有のデフォルトの場所にあるSPFILEから初期化パラメータの読込みを試行します。SPFILEが見つからない場合は、テキスト形式の初期化パラメータ・ファイルを検索します。
- デフォルト以外のサーバー・パラメータ・ファイルを使用したSQL*Plusでの起動
  SQL*Plusでは、PFILE句を使用して、デフォルト以外のサーバー・パラメータ・ファイルでインスタンスを起動できます。
- デフォルト以外のサーバー・パラメータ・ファイルを使用したSRVCTLでの起動
  データベースがOracle Restartで管理されている場合は、データベースのOracle Restart構成でSPFILEの場所オプションを設定または変更することにより、デフォルト以外のSPFILEの場所を指定できます。

関連項目:

初期化パラメータ、初期化パラメータ・ファイルおよびサーバー・パラメータ・ファイルの詳細は、「Oracle Databaseの作成および構成」を参照してください

親トピック: データベースの起動

3.1.2.1 初期化パラメータ・ファイルおよび起動について

データベース・インスタンスを起動すると、データベースは、プラットフォーム固有のデフォルトの場所にあるSPFILEから初期化パラメータの読込みを試行します。SPFILEが見つからない場合は、テキスト形式の初期化パラメータ・ファイルを検索します。

表2-3に、PFILEおよびSPFILEのデフォルトの名前と場所を示します。

プラットフォーム固有のデフォルトの場所では、Oracle Databaseは次の順序でファイル名を検査し、初期化パラメータ・ファイルを検索します。

1. SRVCTLコマンドの`srvctl add database`または`srvctl modify database`の`-spfile`オプションで指定した場所

   現在の設定は`srvctl config database`コマンドで確認できます。

2. `spfileORACLE_SID.ora`
3. `spfile.ora`
4. `initORACLE_SID.ora`

最初の3つのファイルはSPFILEであり、4つ目はテキスト形式の初期化パラメータ・ファイルです。DBCAによってOracle Automatic Storage Managementディスク・グループにSPFILEが作成されている場合、データベースではそのディスク・グループ内のSPFILEが検索されます。

`CREATE SPFILE`文で`AS COPY`が指定されず、データベースがOracle Clusterwareのリソースとして定義されている場合、`spfile_name`と`FROM PFILE`句の両方を指定すると、データベース・リソース内のSPFILEの名前と場所がこの文によって

自動的に更新されます。CREATE SPFILE文でAS COPYが指定されている場合は、SPFILEがコピーされ、データベース・リソースは更新されません。

ノート:

spfile.ora ファイルがこの検索パスに含まれているのは、Oracle Real Application Clusters 環境では、すべてのインスタンスの初期化パラメータ設定が 1 つのサーバー・パラメータ・ファイルに格納されるためです。サーバー・パラメータ・ファイルにはインスタンス固有の格納場所はありません。

Oracle Real Application Clusters 環境のサーバー・パラメータ・ファイルの詳細は、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。

ユーザー(またはDatabase Configuration Assistant)が作成したサーバー・パラメータ・ファイルをテキスト形式の初期化パラメータ・ファイルで上書きする場合は、SQL*Plusを使用してSTARTUPコマンドのPFILE句を指定すると、その初期化パラメータ・ファイルを識別できます。

```
STARTUP PFILE = /u01/oracle/dbs/init.ora
```

デフォルト以外のサーバー・パラメータ・ファイル

デフォルト以外のサーバー・パラメータ・ファイル(SPFILE)とは、デフォルトの場所以外の場所に存在するSPFILEです。通常は、インスタンスをデフォルト以外のSPFILEで起動する必要はありません。ただし、そうする必要がある場合は、SRVCTL (Oracle Restartを使用)およびSQL*Plusのいずれでも実行できます。これらについては、この項で後ほど説明します。

初期化ファイルおよびOracle Automatic Storage Management

Oracle Automatic Storage Management (Oracle ASM)を使用するデータベースの場合、通常はデフォルト以外のSPFILEがあります。Database Configuration Assistant(DBCA)を使用してOracle ASMを使用するようにデータベースを構成すると、DBCAによってOracle ASMディスク・グループ内にデータベース・インスタンスのSPFILEが作成され、ローカル・ファイル・システム内のデフォルトの場所に、そのSPFILEを指し示すテキスト形式の初期化パラメータ・ファイル(PFILE)が作成されます(次の項を参照)。

関連トピック

- Oracle RestartのSRVCTLコマンド・リファレンス

親トピック: 起動時における初期化パラメータの指定

3.1.2.2 デフォルト以外のサーバー・パラメータ・ファイルを使用したSQL*Plusでの起動

SQL*Plusでは、PFILE句を使用して、デフォルト以外のサーバー・パラメータ・ファイルでインスタンスを起動できます。

SQL*Plusでデフォルト以外のサーバー・パラメータ・ファイルを使用して起動するには:

1. SPFILEパラメータのみを記述した1行のテキスト形式の初期化パラメータ・ファイルを作成します。パラメータの値には、デフォルト以外のサーバー・パラメータ・ファイルの場所を指定します。

   たとえば、次のパラメータのみを記述したテキスト形式の初期化パラメータ・ファイル /u01/oracle/dbs/spf_init.oraを作成します。

   ```
   SPFILE = /u01/oracle/dbs/test_spfile.ora
   ```

ノート:

従来のテキスト形式の初期化パラメータ・ファイルで、サーバー・パラメータ・ファイルを設定するために IFILE 初期化パラメータを使用することはできません。この場合は、必ず SPFILE 初期化パラメータを使用してください。

2. この初期化パラメータ・ファイルを指定して、インスタンスを起動します。

```
STARTUP PFILE = /u01/oracle/dbs/spf_init.ora
```

SPFILEはデータベース・ホスト・コンピュータに存在している必要があります。したがって、前述の方法は、クライアント・システムからSPFILEを使用するデータベースを起動する手段にもなります。また、クライアント・システムがクライアント側で初期化パラメータ・ファイルを維持する必要もなくなります。クライアント・システムがSPFILEパラメータが含まれている初期化パラメータ・ファイルを読み込むと、その値をサーバーに渡し、そこで指定されたSPFILEが読み込まれます。

親トピック: 起動時における初期化パラメータの指定

3.1.2.3 デフォルト以外のサーバー・パラメータ・ファイルを使用したSRVCTLでの起動

データベースがOracle Restartで管理されている場合は、データベースのOracle Restart構成でSPFILEの場所オプションを設定または変更することにより、デフォルト以外のSPFILEの場所を指定できます。

SRVCTLでデフォルト以外のサーバー・パラメータ・ファイルを使用して起動するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

```
srvctl modify database -db db_unique_name -spfile spfile_path
```

db_unique_nameは、データベースのDB_UNIQUE_NAME初期化パラメータの設定と一致している必要があります。

3. 次のコマンドを入力します。

```
srvctl start database -db db_unique_name [options]
```

詳細は、「Oracle RestartのSRVCTLコマンド・リファレンス」を参照してください。

親トピック: 起動時における初期化パラメータの指定

3.1.3 データベース・サービスの自動起動について

データベースがOracle Restartで管理されている場合は、個々のデータベース・サービス(サービス)ごとに起動オプションを構成できます。

サービスの管理ポリシーをAUTOMATIC(デフォルト)に設定した場合は、SRVCTLを使用してデータベースを起動するとサービスが自動的に起動します。管理ポリシーをMANUALに設定した場合、サービスは自動起動されず、SRVCTLを使用して手動で起動する必要があります。MANUALを設定しても、サービスは実行中にOracle Restartで監視され、障害が発生すると再起動されます。

データベースがOracle Restartで管理されているOracle Data Guard(Data Guard)環境では、Oracle Restart構成でサービスにData Guardロールを割り当てることにより、サービスの自動起動をさらに制御できます。サービスの管理ポリシーがAUTOMATICであり、割り当てられているいずれかのロールがデータベースの現行のロールに一致する場合にのみ、手動でデータベースを起動するとサービスが自動的に起動されます。

サービスに対してData Guardロールの管理ポリシーを設定する構文は、「srvctl add service」および「srvctl modify service」を参照してください。

ノート:

Oracle Restart を使用する場合は、SRVCTL を使用してデータベース・サービスを作成することをお薦めします。

親トピック: データベースの起動

3.1.4 インスタンス起動の準備

SQL*Plusを使用してデータベース・インスタンスを起動する前に、次の準備ステップを実行する必要があります。

ノート:

次の手順は、Oracle Restart を使用していないインストール用です。データベースを Oracle Restart で管理している場合は、「Oracle Restart で管理されているコンポーネントの起動と停止」の手順に従ってください。

インスタンスを起動準備するには:

1. データベースが依存するOracleコンポーネントが起動されたことを確認します。

   たとえば、データベースのデータがOracle Automatic Storage Management(Oracle ASM)のディスク・グループに格納される場合は、Oracle ASMインスタンスが実行中であること、および必要なディスク・グループがマウントされていることを確認してください。また、データベースが起動する前にOracle Netリスナーを起動することをお薦めします。

2. オペレーティング・システム認証を使用する場合は、OSDBAグループのメンバーとしてデータベース・ホスト・コンピュータにログインします。
3. 環境変数が目的のOracleインスタンスに接続するように設定されていることを確認します。
4. データベースに接続せずに、SQL*Plusを起動します。

   ```
   SQLPLUS /NOLOG
   ```

5. SYSOPER、SYSDBA、SYSBACKUPまたはSYSDGとしてOracle Databaseに接続します。たとえば:

   ```
   CONNECT username AS SYSDBA
   –or–
   CONNECT / AS SYSDBA
   ```

データベースに接続され、データベース・インスタンスを起動する準備が完了します。

関連項目:

- 「オペレーティング・システム認証の使用」
- Oracleインスタンスに接続するための環境変数の設定の詳細は、「データベースに対するコマンドとSQLの発行」を参照してください
- CONNECT、STARTUPおよびSHUTDOWNコマンドの説明と構文は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』

を参照してください

親トピック: データベースの起動

3.1.5 インスタンスの起動

SQL*PlusまたはOracle Restartを使用してインスタンスを起動できます。

- インスタンスの起動について
  Oracle Restartを使用していない場合は、SQL*PlusのSTARTUPコマンドを使用してOracle Databaseインスタンスを起動します。Oracle Restartでデータベースを管理している場合は、`srvctl start database`コマンドを使用することをお薦めします。
- インスタンスを起動し、データベースをマウントしてオープンする方法
  通常のデータベース操作とは、インスタンスが起動しており、データベースがマウントおよびオープンされていることを意味します。このモードでは、有効なユーザーがデータベースに接続して、データ・アクセス操作を実行できます。
- インスタンスを起動するが、データベースをマウントしない方法
  インスタンスは、データベースをマウントしなくても起動できます。通常、この方法で起動するのはデータベースの作成時のみです。
- インスタンスを起動し、データベースをマウントする方法
  インスタンスを起動し、データベースをオープンしないでマウントして、特定のメンテナンス操作を実行できます。
- 起動時にインスタンスへのアクセスを制限する方法
  インスタンスの使用を管理担当者にのみ許可し、一般データベース・ユーザーの使用を禁止するには、制限モードでインスタンスを起動して、オプションでデータベースをマウントおよびオープンします。
- インスタンスを強制的に起動する方法
  通常と異なる状況では、データベース・インスタンスを起動しようとしたときに問題が発生することがありますが、データベース・インスタンスを強制的に起動できます。
- インスタンスを起動し、データベースをマウントして、完全メディア・リカバリを開始する方法
  メディア・リカバリが必要な場合は、インスタンスを起動し、データベースをインスタンスにマウントして、リカバリ処理を自動的に開始できます。
- オペレーティング・システム起動時にデータベースを自動的に起動する方法
  多くのサイトでは、システムの起動直後に1つ以上のOracle Databaseインスタンスとデータベースを自動的に起動する手順を使用しています。
- リモート・インスタンスを起動する方法
  ローカルのOracle Databaseサーバーが分散データベースの一部を構成している場合は、リモート・インスタンスとデータベースを起動できます。

親トピック: データベースの起動

3.1.5.1 インスタンスの起動について

Oracle Restartを使用していない場合は、SQL*PlusのSTARTUPコマンドを使用してOracle Databaseインスタンスを起動します。Oracle Restartでデータベースを管理している場合は、`srvctl start database`コマンドを使用することをお薦めします。

SQL*PlusおよびOracle Restartを使用して、次のような様々なモードでインスタンスを起動できます。

- `NOMOUNT`—データベースをマウントせずにインスタンスを起動します。この場合、データベースへのアクセスは禁止され、通常はデータベース作成または制御ファイルの再作成時にのみ使用されます。

- MOUNT—インスタンスを起動し、データベースをマウントするが、クローズしたままにするモード。この状態では、一部のDBAアクティビティは可能ですが、データベースへの汎用アクセスはできません。
- OPEN—インスタンスを起動し、データベースをマウントしてオープンするモード。この操作を非制限モードで実行してユーザー全員にアクセスを許可するか、制限モードで実行してデータベース管理者のみにアクセスを許可できます。
- FORCE—起動時または停止時に問題が発生した後にインスタンスを強制的に起動するモード。
- OPEN RECOVER—インスタンスを起動し、直後に完全メディア・リカバリを開始するモード。

ノート:

共有サーバー・プロセスを介してデータベースに接続してデータベースのインスタンスを起動することはできません。

次の例では、インスタンスを起動できるいくつかの状態を具体的に説明します。STARTUPコマンドの句の結合時やsrvctl start databaseコマンドの起動オプションの結合時は、いくつかの制限が適用されます。

ノート:

制御ファイル、データベース・ファイルまたはオンライン REDO ログが使用できない場合は、インスタンスの起動で問題が発生することがあります。データベースをマウントするときに CONTROL_FILES 初期化パラメータで指定された 1 つ以上のファイルが存在しない場合、またはオープンできない場合は、Oracle Database によって警告メッセージが返され、データベースはマウントされません。データベースをオープンするときに、1 つ以上のデータファイルまたはオンライン REDO ログを使用できない場合、またはオープンできない場合、データベースは警告メッセージを返し、データベースをオープンしません。

関連項目:

- STARTUPコマンドの構文の詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。
- Oracle Restartで管理されるデータベースの起動手順は、「Oracle Restartで管理されているコンポーネントの起動と停止」を参照してください。

親トピック: インスタンスの起動

3.1.5.2 インスタンスを起動し、データベースをマウントしてオープンする方法

通常のデータベース操作とは、インスタンスを起動し、データベースをマウントおよびオープンすることを意味します。このモードでは、有効なユーザーがデータベースに接続して、データ・アクセス操作を実行できます。

次のコマンドは、インスタンスを起動し、デフォルトの場所から初期化パラメータを読み込んで、データベースをマウントおよびオープンします。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
STARTUP	`srvctl start database -db db_unique_name`

db_unique_nameはDB_UNIQUE_NAME初期化パラメータと一致します。

親トピック: インスタンスの起動

3.1.5.3 インスタンスを起動するが、データベースをマウントしない方法

インスタンスは、データベースをマウントしなくても起動できます。通常、この方法で起動するのはデータベースの作成時のみです。

次のコマンドのいずれかを使用します。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`STARTUP NOMOUNT`	`srvctl start database -db db_unique_name -startoption nomount`

親トピック: インスタンスの起動

3.1.5.4 インスタンスを起動し、データベースをマウントする方法

インスタンスを起動し、データベースをオープンしないでマウントして、特定のメンテナンス操作を実行できます。

たとえば、次のようなタスクの実行時は、データベースのマウントは必要ですが、オープンはしないでください。

- Oracle Database 12cリリース1 (12.1.0.2)以上での、データベース・インスタンスを強制フル・データベース・キャッシュ・モードへの切替。詳細は、「強制フル・データベース・キャッシュ・モードの使用方法」を参照してください。
- REDOログ・アーカイブ・オプションの有効化および無効化。詳細は、「アーカイブREDOログ・ファイルの管理」を参照してください。
- 全データベース・リカバリの実行。詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

次のコマンドは、インスタンスを起動してデータベースをマウントしますが、データベースはクローズしたままです。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`STARTUP MOUNT`	`srvctl start database -db db_unique_name -startoption mount`

親トピック: インスタンスの起動

3.1.5.5 起動時にインスタンスへのアクセスを制限する方法

インスタンスの使用を管理担当者にのみ許可し、一般データベース・ユーザーの使用を禁止するには、制限モードでインスタンスを起動して、オプションでデータベースをマウントおよびオープンします。

次のいずれかのタスクを実行する必要があるときは、このインスタンス起動モードを使用してください。

- データのエクスポートまたはインポートを実行する場合
- SQL*Loaderを使用してデータをロードする場合
- 一時的に一般ユーザーがデータを使用できないようにする場合
- 特定の移行またはアップグレード操作を実行する場合

通常、`CREATE SESSION`システム権限を持つすべてのユーザーが、オープンしているデータベースに接続できます。データベースを制限モードでオープンすると、`CREATE SESSION`および`RESTRICTED SESSION`システム権限の両方を持つユーザーのみがデータベースへのアクセスを許可されます。データベース管理者のみに`RESTRICTED SESSION`システム権限が付与さ

れている必要があります。さらに、インスタンスが制限モードの場合、データベース管理者はOracle Netリスナーからリモートでインスタンスにアクセスすることはできませんが、インスタンスが実行されているシステムからローカルでインスタンスにアクセスすることのみ可能です。

次のコマンドは、制限モードでインスタンスを起動します(データベースをマウントしてオープンします)。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`STARTUP RESTRICT`	`srvctl start database -db db_unique_name -startoption restrict`

MOUNT、NOMOUNTおよびOPENモードと制限モードを組み合せて使用できます。

`RESTRICTED SESSION`機能を無効にするには、`ALTER SYSTEM`文を使用します。

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

データベースを非制限モードでオープンし、後でアクセス制限が必要であると判明した場合は、`ALTER SYSTEM`文を使用して制限できます。詳細は、「オープンしているデータベースへのアクセスを制限する方法」を参照してください。

関連項目:

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

親トピック: インスタンスの起動

3.1.5.6 インスタンスを強制的に起動する方法

通常と異なる状況では、データベース・インスタンスを起動しようとしたときに問題が発生することがありますが、データベース・インスタンスを強制的に起動できます。

次の問題が発生している場合以外は、データベースを強制的に起動しないでください。

- 現行インスタンスを`SHUTDOWN NORMAL`、`SHUTDOWN IMMEDIATE`または`SHUTDOWN TRANSACTIONAL`コマンドで停止できない場合
- インスタンス起動時に問題が発生した場合。

このような問題が発生した場合、次のいずれかのコマンドを指定して新しいインスタンスを起動(必要に応じて、データベースをマウントおよびオープン)すると、通常は問題を解決できます。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`STARTUP FORCE`	`srvctl start database -db db_unique_name -startoption force`

インスタンスの実行中にFORCEモードを使用すると、ABORTモードでインスタンスを停止した後に再起動します。この場合、メッセージ「インスタンス`"abort"`の停止中」に続いて「ORACLEインスタンス`"normal"`の起動中」がアラート・ログに表示されます。

関連項目:

現行インスタンスの強制終了による影響の詳細は、「ABORTモードによる停止」を参照してください

親トピック: インスタンスの起動

3.1.5.7 インスタンスを起動し、データベースをマウントして、完全メディア・リカバリを開始する方法

メディア・リカバリが必要な場合は、インスタンスを起動し、データベースをインスタンスにマウントして、リカバリ処理を自動的に開始できます。

これには、次のいずれかのコマンドを使用します。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`STARTUP OPEN RECOVER`	`srvctl start database -db db_unique_name -startoption "open,recover"`

必要ない場合にリカバリを実行しようとすると、Oracle Databaseによってエラー・メッセージが表示されます。

親トピック: インスタンスの起動

3.1.5.8 オペレーティング・システム起動時にデータベースを自動的に起動する方法

多くのサイトでは、システムの起動直後に1つ以上のOracle Databaseインスタンスとデータベースを自動的に起動する手順を使用しています。

そのための手順は、オペレーティング・システムによって異なります。自動起動の詳細は、オペレーティング・システム固有のOracleマニュアルを参照してください。

データベースの自動起動を構成する方法としてお薦めする(およびプラットフォームに依存しない)方法は、Oracle Restartです。詳細は、「Oracle Databaseの自動再起動の構成」を参照してください。

親トピック: インスタンスの起動

3.1.5.9 リモート・インスタンスを起動する方法

ローカルのOracle Databaseサーバーが分散データベースの一部を構成している場合は、リモート・インスタンスとデータベースを起動できます。

リモート・インスタンスの起動と停止の手順は、通信プロトコルとオペレーティング・システムによって大きく異なります。

親トピック: インスタンスの起動

3.2 データベースの可用性の変更

データベースの可用性を変更できます。可用性の変更は、メンテナンス上の理由やデータベースを読取り専用にするために行う場合があります。

- インスタンスにデータベースをマウントする方法
  特定の管理操作を実行する場合は、データベースを起動してインスタンスにマウントし、クローズしたままにする必要があります。そのためには、インスタンスを起動してデータベースをマウントします。
- クローズしているデータベースをオープンする方法
  データベースをオープンすることによって、マウントされ、クローズしているデータベースを一般的な用途のために使用可能にできます。
- データベースを読取り専用モードでオープンする方法
  データベースを読取り専用モードでオープンすると、オープンしたデータベースを問い合せることができますが、その間に

データの内容がオンラインで変更されることはありません。

- オープンしているデータベースへのアクセスを制限する方法
  データベースが制限モードの場合は、RESTRICTED SESSION権限を持つユーザーのみが新しい接続を開始できます。SYSDBAとして接続するユーザー、またはDBAロールを使用して接続するユーザーにはこの権限があります。

親トピック: 起動と停止

3.2.1 インスタンスにデータベースをマウントする方法

特定の管理操作を実行する場合は、データベースを起動してインスタンスにマウントし、クローズしたままにする必要があります。そのためには、インスタンスを起動してデータベースをマウントします。

- あらかじめ起動したインスタンスにデータベースをマウントするには、次のように、SQL文ALTER DATABASEでMOUNT句を指定します。

```
ALTER DATABASE MOUNT;
```

関連項目:

データベースをマウントし、クローズしておくことが必要な操作(およびインスタンスの起動とデータベースを1ステップでマウントする手順)の詳細は、「インスタンスを起動し、データベースをマウントする方法」を参照してください

親トピック: データベースの可用性の変更

3.2.2 クローズしているデータベースをオープンする方法

データベースをオープンすることによって、マウントされ、クローズしているデータベースを一般的な用途のために使用可能にできます。

- マウントされたデータベースをオープンするには、ALTER DATABASE SQL文でOPEN句を使用します。

```
ALTER DATABASE OPEN;
```

この文の実行後は、CREATE SESSIONシステム権限を持つ有効なOracle Databaseユーザーであれば、誰でもデータベースに接続できます。

親トピック: データベースの可用性の変更

3.2.3 データベースを読取り専用モードでオープンする方法

データベースを読取り専用モードでオープンすると、オープンしたデータベースを問い合せることができますが、その間にデータの内容がオンラインで変更されることはありません。

これにより、データファイルとREDOログ・ファイルにデータが書き込まれないことが保証されますが、データベース・リカバリや、REDOを生成せずにデータベースの状態を変更する操作が制限されることはありません。たとえば、データファイルをオフラインとオンラインの間で切り替えてもデータの内容には影響しないため、このような切替えは可能です。

ディスク・ソートの実行時など、読取り専用モードでデータベースを問い合せるときに一時表領域を使用する場合、問合せの発行者には、デフォルト一時表領域としてローカル管理表領域が割り当てられている必要があります。割り当てられていない場合は、問合せが失敗します。詳細は、「ローカル管理の一時表領域の作成」を参照してください。

次の文では、データベースが読取り専用モードでオープンします。

```
ALTER DATABASE OPEN READ ONLY;
```

また、次のように読取り/書込みモードでデータベースをオープンすることもできます。

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

ただし、読取り/書込みはデフォルトのモードです。

ノート:

RESETLOGS 句と READ ONLY 句は併用できません。

読取り専用データベースの制約

- 読取り専用データベースに対してアプリケーションを実行中は、データベース・オブジェクトへ書き込まないでください。たとえば、アプリケーションは、グローバルな一時表を含むデータベース表に対して行の挿入、削除、更新またはマージを実行する際に、データベース・オブジェクトに書き込みます。アプリケーションは、データベースの順序を操作する際にデータベース・オブジェクトに書き込みます。また、アプリケーションは行のロック、EXPLAIN PLANの実行またはDDLの実行の際にデータベース・オブジェクトに書き込みます。オラクル社が提供するPL/SQLパッケージのファンクションとプロシージャの多くは(DBMS_SCHEDULERなど)、データベース・オブジェクトに書き込みます。アプリケーションがこれらのファンクションとプロシージャのいずれかをコールする場合、または前述の操作のいずれかを実行する場合、アプリケーションからデータベース・オブジェクトへ書込みが実行されるため、読取り専用ではありません。
- 読取り専用データベースで実行する場合は、1つのデータベース・リンクを使用する進行中のトランザクションをコミットまたはロールバックしてから、別のデータベース・リンクを使用する必要があります。これは、最初のデータベース・リンクで汎用的なSELECT文を実行し、トランザクションが現在読取り専用である場合も当てはまります。
- 読取り専用データベースではPL/SQLストアド・プロシージャをコンパイルまたは再コンパイルできません。リモート・プロシージャ・コールが原因のPL/SQLの無効化を最小限に抑えるには、読取り専用データベースに対してリモート・プロシージャ・コールを実行するセッションではREMOTE_DEPENDENCIES_MODE=SIGNATUREを使用します。
- 読取り専用データベースで一度もコールされたことがないリモート・プロシージャ(読取り専用のリモート・プロシージャであっても)は、そのデータベースから起動できません。この制約は、無名PL/SQLブロックの中およびSQL文の中のリモート・プロシージャ・コールに当てはまります。ストアド・プロシージャ内にリモート・プロシージャ・コールを指定するか、またはデータベース内のリモート・プロシージャを読取り専用にする前に起動できます。

関連項目:

ALTER DATABASE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: データベースの可用性の変更

3.2.4 オープンしているデータベースへのアクセスを制限する方法

データベースが制限モードの場合は、RESTRICTED SESSION権限を持つユーザーのみが新しい接続を開始できます。SYSDBAとして接続するユーザー、またはDBAロールを使用して接続するユーザーにはこの権限があります。

すでに実行中のインスタンスを制限モードにするには:

- ENABLE RESTRICTED SESSION句を指定して、SQL文ALTER SYSTEMを実行します。

実行中インスタンスを制限モードにすると、ユーザー・セッションは終了されないか、終了された場合は影響を受けます。したがって、インスタンスを制限モードにした場合は、管理タスクを実行する前に現行のユーザー・セッションをすべて停止(終了)することを考慮します。

インスタンスの制限モードを解除するには、`ALTER SYSTEM`で`DISABLE RESTRICTED SESSION`句を指定します。

関連項目:

- ユーザー・セッションを停止する方法は、「セッションの停止」を参照してください
- インスタンスを制限モードにする理由を学習するには、「起動時にインスタンスへのアクセスを制限する方法」を参照してください

親トピック: データベースの可用性の変更

3.3 データベースの停止

SQL*PlusまたはOracle Restartでデータベースを停止できます。

- データベースの停止について
  Oracle Restartを使用していない場合、データベース・インスタンスを停止するには、SQL*PlusでSYSOPER、SYSDBA、SYSBACKUPまたはSYSDGとして接続し、SHUTDOWNコマンドを発行します。Oracle Restartでデータベースを管理している場合は、`srvctl stop database`コマンドを使用してデータベースを停止する方法をお薦めします。
- NORMALモードによる停止
  NORMALモードでデータベースを停止する場合、現在、接続しているすべてのユーザーが切断するまで待機してから、データベースが停止します。通常モードはデフォルトの停止モードです。
- IMMEDIATEモードによる停止
  IMMEDIATEモードでデータベースを停止する場合、Oracle Databaseにより実行中のSQL文が終了され、ユーザーが切断されます。アクティブなトランザクションは終了され、コミットされていない変更はロールバックされます。
- TRANSACTIONALモードによる停止
  TRANSACTIONALモードでデータベースを停止する場合、ユーザーが新規トランザクションを開始できなくなりますが、現在のトランザクションがすべて完了するまで待機してから、データベースが停止します。現行トランザクションの性質によっては、停止までにかなりの時間がかかる場合があります。
- ABORTモードによる停止
  データベース・インスタンスを強制終了することによって、データベースをただちに停止できます。
- 停止のタイムアウト
  ユーザーによる切断またはトランザクションの完了を待機する停止モードには、待機時間に制限があります。

親トピック: 起動と停止

3.3.1 データベースの停止について

Oracle Restartを使用していない場合、データベース・インスタンスを停止するには、SQL*PlusでSYSOPER、SYSDBA、SYSBACKUPまたはSYSDGとして接続し、SHUTDOWNコマンドを発行します。Oracle Restartでデータベースを管理している場合は、`srvctl stop database`コマンドを使用してデータベースを停止する方法をお薦めします。

停止処理が完了するまで、データベース停止を開始したセッションに制御が戻されません。停止処理の進行中に接続しようとす

るユーザーは、次のようなメッセージを受け取ります。

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

ノート:

共有サーバー・プロセスを介してデータベースに接続している場合は、データベースを停止できません。

データベースには、複数の停止モード(NORMAL、IMMEDIATE、TRANSACTIONALおよびABORT)があります。一部の停止モードでは、データベースを実際に停止する前に、特定のイベント(トランザクションの完了またはユーザーによる切断など)の発生を待機します。これらのイベントに対するタイムアウト間隔は1時間です。

関連項目:

Oracle Restartの詳細は、「Oracle Databaseの自動再起動の構成」を参照してください。

親トピック: データベースの停止

3.3.2 NORMALモードによる停止

NORMALモードでデータベースを停止する場合、現在、接続しているすべてのユーザーが切断するまで待機してから、データベースが停止します。NORMALモードはデフォルトの停止モードです。

データベースを通常の状況で停止するには、次のいずれかのコマンドを使用します。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`SHUTDOWN [NORMAL]`	`srvctl stop database -db db_unique_name -stopoption normal`

SQL*Plus `SHUTDOWN`コマンドの`NORMAL`句はオプションです(これがデフォルトの停止方法であるため)。SRVCTLの場合、`-stopoption`オプションを省略すると、停止操作はデータベースのOracle Restart構成に格納されている停止オプションに従って進行します。デフォルトの停止オプションは`IMMEDIATE`です。

通常のデータベース停止では、次のように処理が進みます。

- 文が発行された後は、新しい接続は許可されません。
- データベースは、停止する前に、現在データベースに接続しているすべてのユーザーによるデータベースからの切断を待機します。

次にデータベースを起動するときに、インスタンス・リカバリ手順は必要ありません。

親トピック: データベースの停止

3.3.3 IMMEDIATEモードによる停止

IMMEDIATEモードでデータベースを停止する場合、Oracle Databaseにより実行中のSQL文が終了され、ユーザーが切断されます。アクティブなトランザクションは終了され、コミットされていない変更はロールバックされます。

データベースの即時停止は、次のような状況のときにのみ使用します。

- 自動バックアップおよび無人バックアップを開始する場合

- 電源が間もなく停止する場合
- データベースやデータベース・アプリケーションの一部が異常に動作している場合で、ユーザーにログオフを依頼できない場合、またはユーザーがログオフできない場合

データベースを即時に停止するには、次のいずれかのコマンドを使用します。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`SHUTDOWN IMMEDIATE`	`srvctl stop database -db db_unique_name -stopoption immediate`

このデータベース停止では、次のように処理が進みます。

- 文が発行されると、新しい接続は許可されず、新しいトランザクションは開始できません。
- すべての未コミットのトランザクションは、ロールバックされます。(長くかかる未コミットのトランザクションが存在する場合、この停止方法ではその名前に反して即座に終了しない場合があります。)
- Oracle Databaseは、現在データベースに接続しているユーザーが切断されるのを待機しません。アクティブなトランザクションは暗黙的にロールバックされ、接続中のユーザーはすべて切断されます。

次にデータベースを起動するときに、インスタンス・リカバリ手順は必要ありません。

親トピック: データベースの停止

3.3.4 TRANSACTIONALモードによる停止

TRANSACTIONALモードでデータベースを停止する場合、ユーザーが新規トランザクションを開始できなくなりますが、現在のトランザクションがすべて完了するまで待機してから、データベースが停止します。現行トランザクションの性質によっては、停止までにかなりの時間がかかる場合があります。

アクティブ・トランザクションを完了してから、計画どおりにインスタンスを停止する場合は、次のいずれかのコマンドを使用します。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`SHUTDOWN TRANSACTIONAL`	`srvctl stop database -db db_unique_name -stopoption transactional`

このデータベース停止では、次のように処理が進みます。

- 文が発行されると、新しい接続は許可されず、新しいトランザクションは開始できません。
- すべてのトランザクションが完了すると、まだインスタンスに接続されているすべてのクライアントが切断されます。
- この時点で、インスタンスは`SHUTDOWN IMMEDIATE`文を発行した場合と同じように停止します。

次にデータベースを起動するときに、インスタンス・リカバリ手順は必要ありません。

TRANSACTIONALオプションによる停止では、クライアントの作業内容が失われずに済み、すべてのユーザーがログオフする必要がなくなります。

親トピック: データベースの停止

3.3.5 ABORTモードによる停止

データベース・インスタンスを強制終了することによって、データベースをただちに停止できます。

この停止方法は、次の場合を除いてできるだけ使用を避けてください。

- データベース、もしくはそのアプリケーションの1つが不適切に機能していて、かつ他のいずれの停止モードも作動しない場合。
- データベースをただちに停止する必要がある(たとえば、電源遮断が1分以内に発生することがわかっている)場合。
- データベース・インスタンスを起動するときに問題が発生した場合。

トランザクションとユーザーの接続を強制終了してデータベースを停止する必要がある場合は、次のいずれかのコマンドを使用します。

SQL*Plus	SRVCTL(Oracle Restartを使用している場合)
`SHUTDOWN ABORT`	`srvctl stop database -db db_unique_name -stopoption abort`

このデータベース停止では、次のように処理が進みます。

- 文が発行されると、新しい接続は許可されず、新しいトランザクションは開始できません。
- Oracle Databaseによって処理されている現行のクライアントSQL文はただちに終了します。
- コミットされていないトランザクションはロールバックされません。
- Oracle Databaseは、現在データベースに接続しているユーザーが切断されるのを待機しません。接続中のユーザーはすべて暗黙的に切断されます。

次にデータベースを起動するときに、自動インスタンス・リカバリ手順が必要になります。

親トピック: データベースの停止

3.3.6 停止のタイムアウト

ユーザーによる切断またはトランザクションの完了を待機する停止モードには、待機時間に制限があります。

停止をブロックするすべてのイベントが1時間以内に発生しない場合、停止操作は「`ORA-01013: ユーザーによって現行の操作の取消しがリクエストされました`」というメッセージを表示して強制終了されます。また、このメッセージは停止プロセスを中断した場合(たとえばCTRL-Cを押した場合など)にも表示されます。インスタンスの停止を中断しようとしないことをお薦めします。停止プロセスが完了してから、インスタンスを再起動してください。

ORA-01013が発生した後は、インスタンスが予測できない状態になると考える必要があります。このため、SHUTDOWNコマンドを再発行して、停止プロセスを継続する必要があります。後続のSHUTDOWNコマンドが引き続き失敗する場合は、SHUTDOWN ABORTコマンドを発行してインスタンスを停止する必要があります。その後でインスタンスを再起動できます。

親トピック: データベースの停止

3.4 データベースの静止

データベースの静止中は、DBAによるトランザクション、問合せ、フェッチまたはPL/SQL文のみ実行できます。

- データベースの静止について

データベースを、DBAによるトランザクション、問合せ、フェッチまたはPL/SQL文の実行のみ許可された状態にすることが必要な場合があります。このような状態は、システム上でDBA以外によるトランザクション、問合せ、フェッチまたはPL/SQL文が実行されていないという意味で、静止状態と呼びます。

- データベースの静止状態への変更
  データベースを静止状態にした場合、DBA以外によるアクティブ・セッションは非アクティブになるまで続行されます。アクティブなセッションとは、トランザクション、問合せ、フェッチまたはPL/SQL文を現在実行しているセッション、または現在なんらかの共有リソース(エンキューなど)を保持しているセッションです。非アクティブなセッションがアクティブになることはできません。
- 通常操作へのシステムのリストア
  システムを通常操作にリストアすると、DBA以外によるすべてのアクティビティを続行できます。
- インスタンスの静止状態の表示
  V$INSTANCEビューを問い合せて、インスタンスの静止状態を表示できます。

親トピック: 起動と停止

3.4.1 データベースの静止について

データベースを、DBAによるトランザクション、問合せ、フェッチまたはPL/SQL文の実行のみ許可された状態にすることが必要な場合があります。このような状態は、システム上でDBA以外によるトランザクション、問合せ、フェッチまたはPL/SQL文が実行されていないという意味で、静止状態と呼びます。

ノート:

静止中のデータベースに関するこの項の説明では、DBA がユーザーSYS または SYSTEM として定義されています。DBA ロールを持つユーザーなどの他のユーザーには、ALTER SYSTEM QUIESCE DATABASE 文の発行や、データベース静止後の処理の継続は許可されていません。

データベースを静止状態にすることで、管理者は、それ以外の状態では安全に実行できない処理を実行できます。次の処理を実行できます。

- 同時ユーザー・トランザクションが同じオブジェクトにアクセスした場合に失敗する処理。たとえば、データベース表のスキーマの変更や、NOWAITロックが必要な既存表への列の追加などがこれに該当します。
- 同時ユーザー・トランザクションによって好ましくない中間影響を受けるアクション、たとえば表を最初にエクスポートしてから削除し、最後にインポートする場合のような、複数ステップ手順からなる表の再編成処理など。表が削除された後であっても、インポートされる前に同時ユーザーが表にアクセスしようとすると、状況を正確に認識できません。

データベースの静止機能がない場合は、データベースを停止し、制限モードで再度オープンする必要があります。これは、特に24時間、365日の可用性が必要なシステムにとっては重大な制限です。データベースの静止によって、ユーザーに対する遮断とデータベースの停止と再起動に伴う停止時間を最小限にできるため、制限が大幅に緩和されます。

データベースが静止中の場合、DBA以外のセッションがアクティブにならないようにするために、データベース・リソース・マネージャの機能が使用されています。したがって、この文が有効な間に現行のリソース・プランを変更しようとすると、その処理はシステムが静止解除されるまでキューに待機します。データベース・リソース・マネージャの詳細は、「Oracle Database Resource Managerを使用したリソースの管理」を参照してください。

親トピック: データベースの静止

3.4.2 データベースの静止状態への変更

データベースを静止状態にした場合、DBA以外によるアクティブ・セッションは非アクティブになるまで続行されます。アクティブなセッションとは、トランザクション、問合せ、フェッチまたはPL/SQL文を現在実行しているセッション、または現在なんらかの共有リソース(エンキューなど)を保持しているセッションです。非アクティブなセッションがアクティブになることはできません。

たとえば、ユーザーが非アクティブなセッションを強制的にアクティブにしようとしてSQL問合せを発行すると、その問合せは停止したようになります。後でデータベースが静止解除されると、セッションが再開され、ブロックされていた処理が実行されます。

- データベースを静止状態にするには、次のSQL文を発行します。

```
ALTER SYSTEM QUIESCE RESTRICTED;
```

DBA以外のセッションがすべて非アクティブになると、ALTER SYSTEM QUIESCE RESTRICTED文が完了し、データベースは静止状態となります。Oracle Real Application Clusters環境でこの文を発行すると、文を発行したインスタンスだけでなく、すべてのインスタンスが影響を受けます。

ALTER SYSTEM QUIESCE RESTRICTED文は、アクティブなセッションが非アクティブになるまで、長時間待機する場合があります。V$BLOCKING_QUIESCEビューを問い合せると、静止操作をブロックしているセッションを判別できます。このビューでは、1つの列SID(セッションID)のみが返されます。これをV$SESSIONと結合すると、次の例のように、セッションに関する詳細を取得できます。

```
select bl.sid, user, osuser, type, program
from v$blocking_quiesce bl, v$session se
where bl.sid = se.sid;
```

データベース静止の要求を中断した場合、またはアクティブなセッションがすべて静止する前にセッションが異常終了した場合は、Oracle Databaseによって、この文による部分的な影響がすべて自動的に取り消されます。

複数のOracle Call Interface (OCI)の連続したフェッチによって問合せが実行されている場合、ALTER SYSTEM QUIESCE RESTRICTED文はすべてのフェッチが完了するまで待機しません。現行のフェッチの完了のみを待機します。

専用サーバー接続の場合も、共有サーバー接続の場合も、この文の発行後、DBA以外のユーザーがログインしようとすると、その処理はデータベース・リソース・マネージャによってすべてキューに送られ、進行しません。ユーザーにはログインが停止したように見えます。データベースが静止解除されると、ログインは再開されます。

文を発行したセッションが終了しても、データベースは静止状態のままです。DBAは、データベースを静止解除される文を明示的に発行するために、データベースにログインする必要があります。

> ノート:
>
> データベースが静止状態の場合でも、Oracle Database のバックグラウンド・プロセスが内部の目的のために更新を実行している可能性があるため、データベースが静止状態の場合はコールド・バックアップを実行できません。また、オンライン・データファイルのヘッダーは引き続きアクセス可能に見えます。このファイル・ヘッダーの状態は、データベースが正しく停止された場合とは異なります。ただし、データベースが静止状態の間でも、オンライン・バックアップ操作は実行できます。

関連項目:

- V$BLOCKING_QUIESCEビューの詳細は、『Oracle Databaseリファレンス』を参照してください

- V$SESSIONビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: データベースの静止

3.4.3 通常操作へのシステムのリストア

システムを通常操作にリストアすると、DBA以外によるすべてのアクティビティを続行できます。

- データベースを通常操作にリストアするには、次のSQL文を発行します。

```
ALTER SYSTEM UNQUIESCE;
```

Oracle Real Application Clusters環境では、データベースを静止した場合と同じセッションまたは同じインスタンスでこの文を発行する必要はありません。ALTER SYSTEM UNQUIESCE文を発行したセッションが異常終了した場合、Oracle Databaseサーバーは静止解除を確実に完了します。

親トピック: データベースの静止

3.4.4 インスタンスの静止状態の表示

V$INSTANCEビューを問い合せて、インスタンスの静止状態を表示できます。

インスタンスの静止状態を表示するには:

- V$INSTANCEビューのACTIVE_STATE列を問い合せます。

列の値は、次のいずれかになります。

- NORMAL: 通常の静止していない状態。
- QUIESCING: 静止途中の状態。DBA以外の一部のセッションがアクティブな状態です。
- QUIESCED: 静止状態。DBA以外のセッションは非アクティブで、許可されていない状態です。

親トピック: データベースの静止

3.5 データベースの一時停止と再開

ALTER SYSTEM SUSPEND文は、データファイル(ファイル・ヘッダーとファイル・データ)および制御ファイルへの入出力(I/O)をすべて停止します。一時停止状態によって、I/Oに干渉されずにデータベースのバックアップを作成できます。データベースを一時停止すると、実行中のすべてのI/O操作の完了が許可され、新しいデータベース・アクセスはキューに待機した状態になります。通常のデータベース操作を再開するには、ALTER SYSTEM RESUME文を使用します。

データベース操作を一時停止するには:

- ALTER SYSTEM SUSPEND文を実行します。

データベース操作を再開するには:

- ALTER SYSTEM RESUME文を実行します。

SUSPENDコマンドは、インスタンスに固有ではありません。Oracle Real Application Clusters環境では、あるシステムでSUSPENDコマンドを発行すると、内部ロッキング・メカニズムを通じてインスタンス間で停止要求が伝播し、特定のクラスタのアクティブ・インスタンスがすべて停止します。ただし、あるインスタンスの一時停止中に新しいインスタンスを起動すると、新しいインスタンスは一時停止されません。

SUSPENDコマンドとRESUMEコマンドは、異なるインスタンスから発行できます。たとえば、インスタンス1、2および3の実行中に、

インスタンス1からALTER SYSTEM SUSPEND文を発行した場合は、インスタンス1、2または3から同様にRESUME文を発行できます。

一時停止/再開機能は、ディスクやファイルをミラー化してそのミラーを分割できるシステムで役立ち、バックアップとリストアの代替ソリューションを提供します。書込み中に既存のデータベースからミラー化されたディスクを分割できないシステムを使用している場合は、この一時停止/再開機能を使用すると容易に分割できます。

一時停止したデータベースのコピーにはコミット前の更新が含まれるため、一時停止/再開機能は通常の停止操作の簡易版ではありません。

ノート:

表領域をホット・バックアップ・モードに設定する代替手段として ALTER SYSTEM SUSPEND 文を使用しないでください。データベースの一時停止操作ではなく、ALTER TABLESPACE BEGIN BACKUP 文を使用してください。

次の文は、ALTER SYSTEM SUSPEND/RESUMEの使用方法を示しています。データベースの状態を確認するために、V$INSTANCEビューを問い合せています。

```
SQL> ALTER SYSTEM SUSPEND;
System altered
SQL> SELECT DATABASE_STATUS FROM V$INSTANCE;
DATABASE_STATUS
---------
SUSPENDED
SQL> ALTER SYSTEM RESUME;
System altered
SQL> SELECT DATABASE_STATUS FROM V$INSTANCE;
DATABASE_STATUS
---------
ACTIVE
```

関連項目:

データベースの一時停止/再開機能を使用してデータベースをバックアップする方法の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: 起動と停止

3.6 インスタンス中断の遅延

INSTANCE_ABORT_DELAY_TIME初期化パラメータを使用して、エラーによりインスタンスが中断したときに、データベースの停止を遅延する時間を秒単位で指定します。

エラーによってOracleデータベース・インスタンスが中断することがあります。INSTANCE_ABORT_DELAY_TIME初期化パラメータを使用して、インスタンスの停止を遅延する時間を指定します。データベース管理者が遅延時間を使用して、エラーについての情報を取得し、インスタンスが中断したときに発生する可能性のある問題を最小限に抑えることができます。たとえば、データベース管理者が遅延時間を使用して診断を取得し、透過アプリケーション・フェイルオーバー(TAF)を使用して接続をリダイレクトし、バッファ・キャッシュをフラッシュできます。遅延した中断が開始されると、アラート・ログにメッセージが書き込まれます。

注意:

INSTANCE_ABORT_DELAY_TIME を大きすぎる値に設定しないでください。インスタンスはエラーが原因でクローズするため、一部のプロセスやリソースが破損したり使用できなくなっている可能性があり、複雑なアクションは実行できない可能性があります。

インスタンス中断を遅延するには:

- INSTANCE_ABORT_DELAY_TIME初期化パラメータで秒数を設定して、エラーが原因でインスタンスが中断するときに停止を遅延する時間を指定します。

  このパラメータはデフォルトでは0に設定されます。

例3-1 INSTANCE_ABORT_DELAY_TIME初期化パラメータの設定

```
ALTER SYSTEM SET INSTANCE_ABORT_DELAY_TIME=60;
```

親トピック: 起動と停止

4 Oracle Databaseの自動再起動の構成

Oracle Restart機能を使用してOracle Databaseを構成すると、ハードウェアまたはソフトウェアに障害が発生した後やデータベース・ホスト・コンピュータが再起動した場合は常に、データベース、リスナーおよびその他のOracleコンポーネントを自動的に再起動できます。

- Oracle Restartの理解
  Oracle Restartは、シングル・インスタンス環境でのOracleデータベースの可用性を高めます。
- Oracle Restartの構成
  Oracle Restartを構成するには、コンポーネントの追加、コンポーネントの削除、またはコンポーネントのオプションの変更ができます。
- Oracle Restartで管理されているコンポーネントの起動と停止
  Oracle Restartの使用時は、コンポーネントの起動と停止にSRVCTLユーティリティを使用することをお薦めします。
- メンテナンス操作のためのOracle Restartの停止および再起動
  OracleホームのいくつかのコンポーネントがOracle Restartで管理されている場合、Oracle RestartおよびOracle Restartで管理されているコンポーネントをOracleホーム内で停止できます。
- Oracle RestartのSRVCTLコマンド・リファレンス
  Oracle Restartに固有のSRVCTLコマンドの構文とオプションの詳細を参照できます。
- CRSCTLコマンド・リファレンス
  Oracle Restartに関連のあるCRSCTLコマンドの構文の詳細を参照できます。

親トピック: 基本データベース管理

4.1 Oracle Restartについて

Oracle Restartは、シングル・インスタンス環境におけるOracleデータベースの可用性を向上します。

- Oracle Restartの概要
  Oracle Restartをインストールすると、ハードウェアまたはソフトウェアに障害が発生した後やデータベース・ホスト・コンピュータが再起動した場合は常に、様々なOracleコンポーネントを自動的に再起動できます。
- 起動の依存性について
  Oracle Restartでは、Oracleコンポーネントがコンポーネントの依存性に従って正しい順序で起動されます。
- Oracle Restartを使用した起動と停止について
  Oracle Restartでは、必要に応じて様々なOracleコンポーネントを自動的に再起動し、ユーザーがシステムを手動で停止したとき、Oracleコンポーネントを適切な順序で自動的に停止します。
- Oracle Restartの起動と停止について
  CRSCTLユーティリティにより、Oracle Restartを起動および停止します。
- Oracle Restart構成
  Oracle Restartでは、管理するすべてのOracleコンポーネントのリストと、各コンポーネントの構成情報を保持します。
- Oracle RestartとOracle Data Guardとの統合
  Oracle RestartはOracle Data Guard(Data Guard)およびOracle Data Guard Broker(ブローカ)と統合されています。
- Oracle Restartとの高速アプリケーション通知
  高速アプリケーション通知(FAN)の高可用性イベントを公開するために、Oracle RestartでOracle Notification Services (ONS)およびOracleアドバンスト・キューが使用されます。統合されているOracle Clientは、FANを使

用して、サービスまたはインスタンスが停止した場合に高速通知をクライアントに提供します。クライアントは、プライマリ・データベースとスタンバイ・データベース間のデータベース接続のフェイルオーバーを自動化できます。

親トピック: Oracle Databaseの自動再起動の構成

4.1.1 Oracle Restartの概要

Oracle Restartをインストールすると、ハードウェアまたはソフトウェアに障害が発生した後やデータベース・ホスト・コンピュータが再起動した場合は常に、様々なOracleコンポーネントを自動的に再起動できます。

表4-1にこれらのコンポーネントを示します。

表4-1 Oracle Restartで自動的に再起動されるOracleコンポーネント

コンポーネント	ノート
データベース・インスタンス	Oracle Restart は 1 台のホスト・コンピュータで複数のデータベースに対応できます。
Oracle Net リスナー	-
データベース・サービス	インストール時に作成されたデフォルトのサービスは Oracle Database によって自動的に管理されるため、含まれません。また、データベースの作成中またはグローバル・サービス中に作成されたデフォルトのサービスも含まれません。グローバル・サービスの詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してください。
Oracle Automatic Storage Management(Oracle ASM)インスタンス	-
Oracle ASM ディスク・グループ	ディスク・グループの再起動はディスク・グループのマウントを指します。
Oracle Notification Services(ONS)	スタンドアロン・サーバー(Oracle Restart)環境の Oracle Grid Infrastructure では、高速アプリケーション通知(FAN)を介したプライマリ・データベースとスタンバイ・データベース間の接続のフェイルオーバーを自動化するために、Oracle Data Guard のインストールで ONS を使用できます。ONS は、フェイルオーバーの際に FAN イベントを統合されているクライアントに送信するサービスです。

Oracle Restartでは定期的にチェック操作を実行し、前述のコンポーネントの状態を監視します。コンポーネントのチェックが失敗すると、そのコンポーネントは停止され、再起動されます。

Oracle Restartはスタンドアロン・サーバー(非クラスタ化)環境でのみ使用します。Oracle Real Application Clusters(Oracle RAC)環境では、Oracle Clusterwareによってコンポーネントを自動再起動する機能が提供されます。

Oracle Restartは、Oracle Databaseホームとは別にインストールするOracle Grid Infrastructureホームから実行され

ます。プラットフォーム用のOracle Grid Infrastructureホームのインストールの詳細は、『Oracle Grid Infrastructureインストレーション・ガイド』を参照してください。

関連項目:

- 「Oracle Restartの構成」
- Oracle Automatic Storage Managementの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

親トピック: Oracle Restartについて

4.1.2 起動の依存性について

Oracle Restartでは、Oracleコンポーネントがコンポーネントの依存性に従って正しい順序で起動されます。

たとえば、データベース・ファイルがOracle ASMディスク・グループに格納されている場合、データベース・インスタンスの起動前に、Oracle ASMインスタンスが起動されて必要なディスク・グループがマウントされていることが確認されます。同様に、コンポーネントの停止が必要な場合、依存コンポーネントが最初に正常に停止されることが確認されます。

Oracle Restartでは、データベース・インスタンスとOracle Netリスナー(リスナー)の間の弱い依存性も管理されます。データベース・インスタンスが起動される際に、Oracle Restartがリスナーの起動を試みます。リスナーの起動に失敗しても、データベースは引き続き起動されます。後でリスナーに障害が発生した場合、Oracle Restartはデータベース・インスタンスを停止および再起動しません。

親トピック: Oracle Restartについて

4.1.3 Oracle Restartを使用した起動と停止について

Oracle Restartでは、必要に応じて様々なOracleコンポーネントを自動的に再起動したり、ユーザーがシステムを手動で停止したとき、Oracleコンポーネントを適切な順序で自動的に停止します。

ただし、個別のOracleコンポーネントを手動で起動または停止する場合があります。Oracle Restartには、Oracle Restartで管理されているコンポーネントを手動で開始および停止する場合に使用する、Server Control(SRVCTL)ユーティリティが含まれています。Oracle Restartの使用中は、コンポーネントの開始と停止をSRVCTLを使用して手動で実行することをお薦めします。

SRVCTLを使用してコンポーネントを停止した後は、障害が発生してもそのコンポーネントはOracle Restartで自動再起動されません。SRVCTLを使用してそのコンポーネントを起動すると、そのコンポーネントでは再度自動再起動が利用できます。

SQL*Plus、リスナー制御ユーティリティ(LSNRCTL)、ASMCMDなどのOracleユーティリティはOracle Restartに統合されています。SQL*Plusを使用してデータベースを停止した場合、Oracle Restartではこの操作をデータベース障害とは解釈せず、データベースは再起動されません。同様に、SQL*PlusまたはASMCMDを使用してOracle ASMインスタンスを停止した場合、Oracle Restartで再起動されません。

SRVCTLを使用したコンポーネントの起動とSQL*Plus(または別のユーティリティ)を使用した起動の重大な相違を次に示します。

- SRVCTLを使用してコンポーネントを起動すると、このコンポーネントが依存するコンポーネントが最初に正しい順序で自動起動されます。
- SQL*Plus(または別のユーティリティ)を使用してコンポーネントを起動すると、依存性チェーンの他のコンポーネントは

自動起動されません。このコンポーネントが依存するコンポーネントが起動されていることを確認する必要があります。

さらに、Oracle Restartで管理されているすべてのコンポーネントを、指定したOracleホームで1つのコマンドを使用して起動および停止できます。OracleホームはOracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。この機能はパッチのインストール時に役立ちます。

関連項目:

「Oracle Restartで管理されているコンポーネントの起動と停止」

親トピック: Oracle Restartについて

4.1.4 Oracle Restartの起動と停止について

CRSCTLユーティリティでは、Oracle Restartを起動および停止します。

また、CRSCTLユーティリティを使用して、Oracle高可用性サービスを有効化または無効化できます。Oracle RestartではOracle高可用性サービスを使用して、Oracle Restartで管理されているすべてのコンポーネントを自動的に起動および停止します。たとえば、Oracle高可用性サービス・デーモンでは、データベース、リスナーおよびOracle ASMインスタンスを自動的に起動します。Oracle高可用性サービスを無効にした場合、ノードの再起動時に、Oracle Restartで管理されているコンポーネントは起動されません。

通常、Oracleのインストールで実行されているすべてのOracleソフトウェアを停止する必要がある場合にCRSCTLユーティリティを使用します。たとえば、パッチをインストールするか、オペレーティング・システムのメンテナンスを実行する際にOracle Restartの停止が必要になる場合があります。メンテナンスの完了後、CRSCTLユーティリティを使用してOracle Restartを起動します。

関連項目:

CRSCTLユーティリティの使用方法は、「メンテナンス操作のためのOracle Restartの停止および再起動」

親トピック: Oracle Restartについて

4.1.5 Oracle Restart構成

Oracle Restartでは、管理するすべてのOracleコンポーネントのリストと、各コンポーネントの構成情報を保持します。

この情報全体をOracle Restart構成と呼びます。コンポーネントがOracle Restartで起動される場合、そのコンポーネントの構成情報に従って起動されます。たとえば、Oracle Restart構成にはデータベースのサーバー・パラメータ・ファイル(SPFILE)の場所や、リスナーがリスニングするTCPポートなどが含まれています。

Oracle Restartをインストールし、Database Configuration Assistant(DBCA)でデータベースを作成した場合、データベースはDBCAによってOracle Restart構成に自動的に追加されます。DBCAによってデータベースが起動されると、データベースと他のコンポーネント間で必要な依存性(データベースがデータを格納するディスク・グループなど)が確立され、Oracle Restartでのデータベースの管理が開始されます。

SRVCTLコマンドを使用すると、Oracle Restart構成にコンポーネントを手動で追加および削除できます。たとえば、データベースがすでに実行されているホストにOracle Restartをインストールした場合、SRVCTLを使用してそのデータベースをOracle Restart構成に追加できます。コンポーネントを手動でOracle Restart構成に追加してSRVCTLを使用して起動すると、そのコンポーネントはOracle Restartで管理されるようになり、必要に応じて再起動されます。

ノート:

Oracle Restart 構成へのコンポーネントの追加は、Oracle Restart を使用したコンポーネントの登録とも呼ばれます。

他のSRVCTLコマンドを使用すると、Oracle Restartで管理されているコンポーネントのステータスと構成を表示したり、コンポーネントの管理を一時的に無効にした後再度有効にするなどの操作を実行できます。

ノート:

Oracle Database 19c 以降、SERVICE_NAMES パラメータをお客様が使用することは非推奨になりました。今後のリリースでサポートが終了する可能性があります。サービスを管理するには、SRVCTL または GDSCTL コマンドライン・ユーティリティ、または DBMS_SERVICE パッケージを使用することをお薦めします。

Oracle Restartのインストール時、Oracleコンポーネントを作成する多くの操作によりOracle Restart構成にコンポーネントが自動的に追加されます。表4-2に、作成操作、および作成されたコンポーネントが自動的に追加されるかどうかを示します。

表4-2 作成操作とOracle Restart構成

作成操作	**作成されたコンポーネントが自動的にOracle Restart構成に追加されるかどうか。**
OUI または DBCA を使用してデータベースを作成します。	はい
CREATE DATABASE SQL 文を使用してデータベースを作成します。	いいえ
OUI、DBCA または ASMCA を使用して Oracle ASM インスタンスを作成します。	はい
ディスク・グループ(メソッド)を作成します。	はい
NETCA を使用してリスナーを追加します。	はい
SRVCTL を使用してデータベース・サービスを作成します。	はい
SERVICE_NAMES 初期化パラメータを変更してデータベース・サービスを作成します。脚注 1	いいえ
DBMS_SERVICE.CREATE_SERVICE を使用してデータベース・サービスを作成します。	いいえ

作成操作	作成されたコンポーネントが自動的にOracle Restart構成に追加されるかどうか。
スタンバイ・データベースを作成します。	いいえ

脚注1

Oracle Restartを使用している場合はお薦めできません

表4-3に、削除操作と、削除されたコンポーネントがOracle Restart構成からも自動的に削除されるかどうかを示します。

表4-3 削除操作とOracle Restart構成

操作	削除されたコンポーネントがOracle Restart構成から自動的に削除されるかどうか。
DBCA を使用してデータベースを削除します。	はい
オペレーティング・システムのコマンドでデータベース・ファイルを削除し、データベースを削除します。脚注 2	いいえ
NETCA を使用してリスナーを削除します。	はい
Oracle ASM ディスク・グループを削除します(任意の方法)。	はい
SRVCTL を使用してデータベース・サービスを削除します。	はい
他の方法でデータベース・サービスを削除します。	いいえ

脚注2

お薦めしません

親トピック: Oracle Restartについて

4.1.6 Oracle RestartとOracle Data Guardとの統合

Oracle RestartはOracle Data Guard(Data Guard)およびOracle Data Guard Broker(ブローカ)と統合されています。

ロール変更リクエストに対応してデータベースの停止と再起動が必要になった場合、正しい順序で(依存性を考慮して)Oracle Restart構成の設定に従ってデータベースが停止され、再起動されます。Data Guardロールが遷移した後、新しいデータベース・ロールで実行するように構成されているデータベース・サービスがすべてアクティブになっており、新しいロールで実行するように構成されていないサービスがすべて停止していることもOracle Restartによって確認されます。

また、Oracle Restart構成では次のコンポーネントのData Guard関係の構成オプションをサポートします。

- データベース: Oracle Restart構成にデータベースを追加する場合、データベースの現在のData Guardロールを指定できます(PRIMARY、PHYSICAL_STANDBY、LOGICAL_STANDBYまたはSNAPSHOT_STANDBY)。ブローカを使用して後でロールを変更すると、Oracle Restartではデータベースの構成を新しいロールで自動的に更新します。ブローカを使用せずにデータベース・ロールを変更した場合、Oracle Restart構成でデータベースのロールを手動で変更して新しいロールを反映させる必要があります。
- データベース・サービス—データベース・サービスをOracle Restart構成に追加する場合、サービスに1つ以上のData Guardロールを指定できます。この構成オプションが設定された場合、サービス・ロールの1つが現在のデータベース・ロールに一致する場合にのみ、データベースのオープン時にサービスは起動されます。

関連項目:

- Oracle Data Guardの詳細は、『Oracle Data Guard概要および管理』を参照してください。
- 「Oracle Restartとの高速アプリケーション通知」
- 「プライマリ・データベースとスタンバイ・データベースの接続のフェイルオーバーの自動化」

親トピック: Oracle Restartについて

4.1.7 Oracle Restartとの高速アプリケーション通知

高速アプリケーション通知(FAN)の高可用性イベントを公開するために、Oracle RestartでOracle Notification Services (ONS)およびOracleアドバンスト・キューが使用されます。統合されているOracle Clientは、FANを使用して、サービスまたはインスタンスが停止した場合に高速通知をクライアントに提供します。クライアントは、プライマリ・データベースとスタンバイ・データベース間のデータベース接続のフェイルオーバーを自動化できます。

- 高速アプリケーション通知の概要
  FANは、UPイベントやDOWNイベントなどのサービス・ステータス変更を含めた構成の変更について他のプロセスに通知するためにOracle Restartで使用できる高可用性通知メカニズムです。
- サービスとFANにおけるアプリケーションの高可用性
  Oracle Databaseはサービスの可用性の維持に重点を置いています。Oracle Restartでは、Oracleサービスが常に使用できるように設定されています。Oracle Restartではデータベースおよびそのサービスが監視され、構成時にFANを使用してイベント通知が送信されます。

関連項目:

Oracle Databaseアドバンスト・キューイング・ユーザーズ・ガイド

親トピック: Oracle Restartについて

4.1.7.1 高速アプリケーション通知の概要

FANは、UPイベントやDOWNイベントなどのステータス変更を含めた構成の変更について他のプロセスに通知するためにOracle Restartで使用できる高可用性通知メカニズムです。

FANには、インスタンスまたはサーバーで障害が発生した場合に、実行中のトランザクションを即座に終了する機能が備えられています。統合されているOracle Clientは、イベントおよび応答を受信します。アプリケーションによる応答は、エラーをユーザー

に伝播するか、またはトランザクションを再実行しアプリケーション・ユーザーからエラーをマスクすることによって実行されます。DOWNイベントが発生すると、統合されているクライアントでは、終了されたデータベースへの接続が即座にクリーン・アップされます。UPイベントが発生すると、クライアントでは新しいプライマリ・データベース・インスタンスへの新しい接続が作成されます。

Oracle Restartでは、管理対象のインスタンスまたはサービスが起動または停止するたびに、FANイベントが公開されます。フェイルオーバーの後、Oracle Data Guard Broker(ブローカ)によってFANイベントが公開されます。FANイベントは次の方法で使用できます。

- Oracle Database JDBC、Universal Connection Pool for Java、Oracle Call InterfaceおよびOracle Database ODP.NETのいずれかのOracle統合データベース・クライアントを使用している場合は、プログラムを変更することなくOracle RestartでFANを使用できます。これらのクライアントには、フェイルオーバー後に自動的に新規プライマリ・データベースに接続する高速接続フェイルオーバー(FCF)を構成できます。
- FANのサーバー側コールアウトをデータベース層で構成できます。

プライマリ・データベース障害などのDOWNイベントでは、新しいプライマリ・データベースにできるだけ早くフェイルオーバーできるようにするため、FANからクライアントへの通知が即座に実行されます。クライアントではタイムアウトが発生するまで待機しません。クライアントに対する通知は即座に実行され、クライアントは通知された時点でフェイルオーバーするように構成しておく必要があります。

UPイベントでは、サービスおよびインスタンスが起動されている場合、アプリケーションが追加のリソースを即時に利用できるように、新しい接続を作成できます。

また、サーバー側のコールアウトを介して、FANを使用して次の操作を実行することもできます。

- ステータス情報のログへの記録
- リソースの起動に失敗した場合のDBAの呼出しまたはチケットのオープン
- サービスと同じ場所に配置する必要がある外部依存アプリケーションの自動的な起動

FANイベントは、ONSおよびOracle Databaseアドバンスト・キューイングのキューを使用して発行されます。キューは、サーバーを構成するときに自動的に構成されます。ONSはSRVCTLコマンドを使用して手動で構成する必要があります。

Connection Manager(CMAN)およびOracle Net ServicesのリスナーはFANイベントと統合されているため、CMANおよびリスナーでは、障害が発生したデータベースに対する接続要求が誤って送信されないように、障害が発生したインスタンスによって提供されるサービスを即座に登録解除できます。

関連項目:

Oracle Data Guard環境におけるFANイベントの詳細は、『Oracle Data Guard Broker』を参照してください。

親トピック: Oracle Restartとの高速アプリケーション通知

4.1.7.2 サービスおよびFANを使用したアプリケーションの高可用性

Oracle Databaseはサービスの可用性の維持に重点を置いています。Oracle Restartでは、Oracleサービスが常に使用できるように設定されています。Oracle Restartではデータベースおよびそのサービスが監視され、構成時にFANを使用してイベント通知が送信されます。

- 計画外停止の管理
  Oracle Restartでは、停止を検出すると、障害が発生したコンポーネントを孤立させ、依存コンポーネントをリカバリします。障害が発生したコンポーネントがデータベース・インスタンスの場合は、Oracle Data Guardがスタンバイ・データ

ベースにフェイルオーバーした後、新しいプライマリ・データベース上のOracle Restartによって、現在のロールで定義されたサービスが開始されます。

- 計画停止の管理
  Oracle Restartには、プライマリ・データベースのシャットダウンが必要となる修復、アップグレードおよび変更向けに、アプリケーション・ユーザーに対するサービス停止を最小限に抑えるための、サービスを無効化および有効化するインタフェースが備えられています。
- 高速アプリケーション通知の高可用性イベント
  FANイベント・レコード・パラメータとイベント・タイプについて理解します。
- 高速アプリケーション通知のコールアウトの使用
  FANコールアウトは、高可用性イベントが発生したときにOracle Restartで即座に実行されるサーバー側の実行可能ファイルです。
- 高速アプリケーション通知と統合されているOracle Client
  Oracle Restartデータベースへの接続に使用される多くの一般的なOracle Clientドライバは、FANと統合されています。そのため、FANを使用する最も簡単な方法は、Oracle統合クライアントを使用することです。

親トピック: Oracle Restartとの高速アプリケーション通知

4.1.7.2.1 計画外停止の管理

Oracle Restartでは、停止を検出すると、障害が発生したコンポーネントを孤立させ、依存コンポーネントをリカバリします。障害が発生したコンポーネントがデータベース・インスタンスの場合は、Oracle Data Guardがスタンバイ・データベースにフェイルオーバーした後、新しいプライマリ・データベース上のOracle Restartによって、現在のロールで定義されたサービスが開始されます。

FANイベントは、Oracle RestartおよびOracle Data Guard Brokerによって、ONSとアドバンスト・キューイングを介して公開されます。また、FANコールアウトを使用して通知を実行することもできます。

ノート:

Oracle Restart では、コールアウトの実行順序は保証されません。コールアウトは非同期で実行され、スケジュールは変動します。

Oracle Restartでは、データベースだけでなく、リスナーやOracle Automatic Storage Management (Oracle ASM)プロセスなどのサブシステムの再起動も含めた、再起動およびリカバリが自動的に行われます。FANコールアウトを使用して、障害管理システムに障害を報告して、修復ジョブを起動できます。

親トピック: サービスおよびFANを使用したアプリケーションの高可用性

4.1.7.2.2 計画停止の管理

Oracle Restartには、プライマリ・データベースのシャットダウンが必要になる修復、アップグレードおよび変更向けに、アプリケーション・ユーザーに対するサービス停止を最小限に抑えるための、サービスを無効化および有効化するインタフェースが備えられています。

Oracle RestartとともにOracle Data Guard Brokerを使用すると、計画停止の期間中、データベース・サービスがプライマリからスタンバイにフェイルオーバーするように調整できます。処理が完了した後、サービスを通常の動作に戻すことができます。

サービスの管理ポリシーによって、データベースの再起動時にサービスを自動的に開始するかどうかが制御されます。サービスの管理ポリシーを`AUTOMATIC`に設定すると、自動的に再起動されます。サービスの管理ポリシーを`MANUAL`に設定すると、手動

で開始する必要があります。

関連項目:

「コンポーネントのOracle Restart構成の変更」

親トピック: サービスおよびFANを使用したアプリケーションの高可用性

4.1.7.2.3 高速アプリケーション通知の高可用性イベント

FANイベント・レコード・パラメータとイベント・タイプについて理解します。

表4-4に、FANイベント・レコード・パラメータとイベント・タイプに続き、イベント・プロパティの名前と値のペアを示します。イベント・タイプは常に最初のエントリであり、タイムスタンプは常に最後のエントリとなります。次の例では、名前と値のペアの名前はFan event type(service_member)に示され、名前と値のペアの値はPropertiesに示されています。

```
FAN event type: service_member
Properties: version=1.0 service=ERP database=FINPROD instance=FINPROD host=node1
status=up
```

表4-4 イベント・レコード・パラメータと説明

パラメータ	説明
VERSION	イベント・レコードのバージョン。リリースの変更を識別するために使用されます。
EVENT TYPE	SERVICE、SERVICE_MEMBER、DATABASE、INSTANCE、NODE、ASM、SRV_PRECONNECT。データベースおよびインスタンス・タイプには、DB_UNIQUE_NAME.DB_DOMAIN などのデータベース・サービスが示されます。
DATABASE UNIQUE NAME	サービスをサポートしている一意のデータベースを示し、DB_UNIQUE_NAME 初期化パラメータの値と一致します。このパラメータのデフォルト値は、初期化パラメータ DB_NAME の値です。
INSTANCE	サービスをサポートしているインスタンスの名前であり、ORACLE_SID の値と一致します。
NODE NAME	サービスをサポートしているノードまたは停止したノードの名前であり、Cluster Synchronization Services(CSS)で認識されるノード名と一致します。
SERVICE	サービス名であり、DBA_SERVICES のサービスと一致します。
STATUS	値は、UP、DOWN、NOT_RESTARTING、PRECONN_UP、PRECONN_DOWN および UNKNOWN です。
REASON	Data_Guard_Failover、Failure、Dependency、User、Autostart、

パラメータ	説明
	Restart。
CARDINALITY	すべての UP イベントに含まれている、現在アクティブなサービス・メンバー数を示します。
TIMESTAMP	通知イベントをオーダーする場合に使用するローカル・タイム・ゾーン。

表4-5に示すように、FANレコードは各セッションのデータベースの署名と一致します。

表4-5 FANパラメータおよび一致するデータベースの署名

FANパラメータ	一致するOracle Databaseの署名
SERVICE	sys_context('userenv', 'service_name')
DATABASE UNIQUE NAME	sys_context('userenv', 'db_unique_name')
INSTANCE	sys_context('userenv', 'instance_name')
NODE NAME	sys_context('userenv', 'server_host')

親トピック: サービスおよびFANを使用したアプリケーションの高可用性

4.1.7.2.4 高速アプリケーション通知のコールアウトの使用方法

FANコールアウトは、高可用性イベントが発生したときにOracle Restartで即座に実行されるサーバー側の実行可能ファイルです。

FANコールアウトを使用すると、イベントが発生したとき、次のようなアクティビティを自動化できます。

- 障害追跡チケットのオープン
- ページャへのメッセージの送信
- 電子メールの送信
- サーバー側のアプリケーションの起動および停止
- 各イベントの発生とともにロギングすることによるアップタイム・ログのメンテナンス

FANコールアウトを使用するには:

- プライマリおよびスタンバイの両方のデータベース・サーバーのディレクトリgrid_home/racg/usrcoに実行可能ファイルを配置します。スクリプトを使用している場合は、実行可能ファイルの最初の行にシェルを設定します。

次に、grid_home/racg/usrco/callout.shコールアウトのサンプル・ファイルを示します。

```
#! /bin/ksh
FAN_LOGFILE= [your path name]/admin/log/`hostname`_uptime.log
echo $* "reported="`date` >> $FAN_LOGFILE &
```

次に、前述の例の出力を示します。

```
NODE VERSION=1.0 host=sun880-2 status=nodedown reason=
timestamp=08-Oct-2004 04:02:14 reported=Fri Oct 8 04:02:14 PDT 2004
```

表4-5に示すように、FANレコードは各セッションのデータベースの署名と一致します。この情報を使用すると、FANイベントのデータに該当するセッションでアクションを実行できます。

関連項目:

コールアウトおよびイベントの詳細は、表4-4を参照してください。

親トピック: サービスおよびFANを使用したアプリケーションの高可用性

4.1.7.2.5 高速アプリケーション通知と統合されているOracle Client

Oracle Restartデータベースへの接続に使用される多くの一般的なOracle Clientドライバは、FANと統合されています。そのため、FANを使用する最も簡単な方法は、Oracle統合クライアントを使用することです。

CMANセッション・プール、Oracle Call Interface、Universal Connection Pool for Java、JDBC simplefan APIおよびODP.NET接続プールを使用できます。この機能は、使用可能なプライマリ・データベースにアプリケーションからいつでも一貫して接続できるようにすることを目的としています。

関連項目:

「プライマリ・データベースとスタンバイ・データベースの接続のフェイルオーバーの自動化」

親トピック: サービスおよびFANを使用したアプリケーションの高可用性

4.2 Oracle Restartの構成

Oracle Restartを構成するには、コンポーネントの追加、コンポーネントの削除、またはコンポーネントのオプションの変更ができます。

- Oracle Restartの構成について
  スタンドアロン・サーバー用にOracle GridインフラストラクチャをインストールすることによってOracle Restartをインストールしてからデータベースを作成すると、データベースがOracle Restart構成に自動的に追加され、必要に応じて自動的に再起動されます。しかし、データベースがすでに存在するホスト・コンピュータにOracle Restartをインストールした場合、データベース、リスナー、Oracle Automatic Storage Management(Oracle ASM)インスタンスおよびその他のコンポーネントをOracle Restart構成に手動で追加する必要があります。
- SRVCTLの実行準備
  多くのOracle Restartのタスクは、SRVCTLユーティリティの実行を必要とします。正しいOracleホームからSRVCTLを実行していること、および正しいユーザー・アカウントでホスト・コンピュータにログインしていることを確認する必要があります。
- SRVCTLのヘルプの表示
  SRVCTLユーティリティのオンライン・ヘルプが使用できます。
- Oracle Restart構成へのコンポーネントの追加
  ほとんどの場合、Oracle Restartを実行しているホストでOracleコンポーネントを作成すると、Oracle Restart構成にコンポーネントが自動的に追加されます。ただし、場合によっては、コンポーネントを手動で追加する必要があります。
- Oracle Restart構成からのコンポーネントの削除
  Oracleでお薦めする方法を使用してOracleコンポーネントを削除した場合、そのコンポーネントはOracle Restart構成からも自動的に削除されます。

- Oracle Restartでのコンポーネント管理の無効化と有効化
  Oracle Restartでのコンポーネント管理は一時的に無効にできます。この機能を実行するケースの1つとして、コンポーネントのメンテナンスを実行する場合があります。たとえば、コンポーネントの修復が必要な場合、障害の発生時やホスト・コンピュータの再起動時にコンポーネントを自動的に再起動しない方がよいことがあります。メンテナンスの完了後、コンポーネントの管理を再度有効にできます。
- コンポーネントのステータスの表示
  SRVCTLを使用すると、Oracle Restartで管理されているコンポーネントの実行ステータス(実行中または実行していない)を表示できます。一部のコンポーネントでは、他の情報も表示されます。
- コンポーネントのOracle Restart構成の表示
  SRVCTLを使用すると、任意のコンポーネントのOracle Restart構成を表示できます。Oracle Restartではコンポーネント・タイプごとに異なる構成情報を保持します。SRVCTLコマンドの1形式を使用して、Oracle Restartで管理されているコンポーネントのリストを取得できます。
- コンポーネントのOracle Restart構成の変更
  SRVCTLを使用して、コンポーネントのOracle Restart構成を変更できます。たとえば、Oracle Restartがリスナーを起動したときにリスナーがリスニングするポート番号や、Oracle Restartがデータベースを起動する際に指し示すサーバー・パラメータ・ファイル(SPFILE)を変更できます。
- Oracle Restart構成の環境変数の管理
  Oracle Restart構成には環境変数の名前と値のペアを格納できます。
- SRVCTLを使用したデータベース・サービスの作成と削除
  Oracle Restartを使用してデータベースを管理する場合、データベース・サービスの作成と削除にはSRVCTLを使用することをお薦めします。SRVCTLを使用してデータベース・サービスを追加すると、サービスはOracle Restart構成に自動的に追加され、サービスとデータベース間の依存性が確立されます。したがって、サービスの起動時にデータベースが起動されていない場合、Oracle Restartで最初にデータベースが起動されます。
- Oracle Restart環境でのFANイベントの有効化
  Oracle Restartを有効にして高速アプリケーション通知(FAN)イベントを公開するには、Oracle Restartサーバーおよび統合されているクライアントを含むOracle Notification Services (ONS)ネットワークを作成する必要があります。
- プライマリ・データベースとスタンバイ・データベースの接続のフェイルオーバーの自動化
  Oracle RestartおよびOracle Data Guardのプライマリ・データベースとスタンバイ・データベースを使用する構成では、スイッチオーバーまたはフェイルオーバーの際にデータベース・サービスがプライマリからスタンバイに自動的にフェイルオーバーします。
- クライアントでの高速接続フェイルオーバーの有効化
  JDBC、OCIまたはODP.NETを使用するクライアントのようなFAN統合クライアントに、高可用性を提供します。高速接続フェイルオーバーを使用するようにクライアントを構成した場合、クライアントが自動的にFANイベントをサブスクライブし、データベースのUPイベントおよびDOWNイベントに対処できます。それに対応して、Oracle Databaseは、要求されたデータベース・サービスを提供するアクティブ・インスタンスにクライアントを接続します。

親トピック: Oracle Databaseの自動再起動の構成

4.2.1 Oracle Restartの構成について

スタンドアロン・サーバー用にOracle GridインフラストラクチャをインストールすることによってOracle Restartをインストールしてからデータベースを作成すると、データベースがOracle Restart構成に自動的に追加され、必要に応じて自動的に再起動されます。しかし、データベースがすでに存在するホスト・コンピュータにOracle Restartをインストールした場合、データベース、リスナー、Oracle Automatic Storage Management(Oracle ASM)インスタンスおよびその他のコンポーネントをOracle

Restart構成に手動で追加する必要があります。

Oracle Restartでデータベースを管理するよう構成した後、次の作業が必要なことがあります。

- Oracle Restart構成に他のコンポーネントを追加します。
- Oracle Restart構成からコンポーネントを削除します。
- 1つ以上のコンポーネントのOracle Restartでの管理を一時停止します。
- 個々のコンポーネントのOracle Restart構成オプションを変更します。

関連項目:

「Oracle Restartについて」

親トピック: Oracle Restartの構成

4.2.2 SRVCTLの実行準備

多くのOracle Restartのタスクは、SRVCTLユーティリティの実行を必要とします。正しいOracleホームからSRVCTLを実行していること、および正しいユーザー・アカウントでホスト・コンピュータにログインしていることを確認する必要があります。

表4-6は、SRVCTLで構成できるコンポーネントを示しており、SRVCTLを実行する必要があるOracleホームをコンポーネントごとに示しています。

表4-6 SRVCTLを起動するOracleホームの決定

構成対象コンポーネント	SRVCTLを起動するOracleホーム
データベース、データベース・サービス	データベース・ホーム
Oracle ASM インスタンス、ディスク・グループ、リスナー脚注3、ONS	Oracle Grid インフラストラクチャ・ホーム

脚注3

Oracle Grid Infrastructureホームからリスナーが起動されたことが前提となっています。既存のデータベースにOracle Restartをインストールした場合は、リスナーがデータベース・ホームから起動されている場合があるため、その場合はデータベース・ホームからSRVCTLを起動します。

SRVCTLの実行を準備するには::

1. 表4-6を使用して、SRVCTLを実行する必要があるOracleホームを決定します。
2. Oracle Restart構成を変更するSRVCTLコマンド(`add`、`remove`、`enable`、`disable`など)を実行する場合、次のいずれかを実施します。
   - UNIXおよびLinuxの場合、ステップ1で決定したOracleホームをインストールしたユーザーとしてデータベース・ホスト・コンピュータにログインします。
   - Windowsの場合、データベース・ホスト・コンピュータに管理者としてログインします。

   上記以外の場合、任意のユーザーとしてホスト・コンピュータにログインします。

3. SRVCTLコマンドの入力に使用するコマンド・ウィンドウを開きます。

   コマンドを入力するには、SRVCTLプログラムがPATH環境変数で指定されていることを確認する必要があります。指定されていない場合は、プログラムに絶対パスを入力できます。

親トピック: Oracle Restartの構成

4.2.3 SRVCTLのヘルプの表示

SRVCTLユーティリティのオンライン・ヘルプが利用できます。

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

```
srvctl
```

詳細なヘルプを表示するには、次のコマンドを入力します。

```
srvctl -help
```

特定コマンドの詳細なヘルプを表示するには、次のように入力します。

```
srvctl command -help
```

たとえば、addコマンドのヘルプおよび各コンポーネント・タイプの様々なオプションに関するヘルプを表示するには、次のように入力します。

```
srvctl add -help
```

特定のコンポーネント・タイプに対する特定のコマンドに関する詳細なヘルプを表示するには、次のように入力します。

```
srvctl command object -help
```

たとえば、データベース・サービスの追加に関するヘルプを表示するには、次のコマンドを入力します。

```
srvctl add service -help
```

SRVCTLコマンドのリストは「Oracle RestartのSRVCTLコマンド・リファレンス」を、コンポーネントのリストは表4-7をそれぞれ参照してください。

Oracle Database 12cからは、キーワード・パラメータが優先されるために、単一文字パラメータは非推奨になりました。下位互換性をサポートするために、単一文字パラメータと新しいキーワード・パラメータを組み合せて使用できます。ヘルプではデフォルトでキーワード・パラメータが表示されますが、該当する場合は、`-help`パラメータの後に`-compatible`パラメータを追加することによって、等価の単一文字を取得できます。

たとえば、等価の単一文字を含む、データベース・サービスの追加に関するヘルプを表示するには、次のコマンドを入力します。

```
srvctl add service -help -compatible
```

等価の単一文字が、キーワード・パラメータの横のカッコ内に表示されます。Oracle Database 12c以降の新規パラメータには、等価の単一文字はありません。

親トピック: Oracle Restartの構成

4.2.4 Oracle Restart構成へのコンポーネントの追加

ほとんどの場合、Oracle Restartを実行しているホストでOracleコンポーネントを作成すると、Oracle Restart構成にコン

ポーネントが自動的に追加されます。ただし、場合によっては、コンポーネントを手動で追加する必要があります。

(表4-2を参照してください。)その後、コンポーネントは必要に応じて自動的に再起動されます。

次の場合、SRVCTLを使用してOracle Restart構成にコンポーネントを手動で追加する必要があります。

- データベースの作成後にOracle Restartをインストールします。
- CREATE DATABASE SQL文を使用して、同じホスト・コンピュータにOracle Databaseを追加で作成します。
- DBMS_SERVICE.CREATE_SERVICEパッケージ・プロシージャを使用してデータベース・サービスを作成します。(SRVCTLを使用することをお薦めします。)

ノート:

Oracle Restart 構成へのコンポーネントの追加は、Oracle Restart を使用したコンポーネントの登録とも呼ばれます。

Oracle Restart構成にコンポーネントを追加してもそのコンポーネントは起動されません。起動するには、srvctl startコマンドを使用する必要があります。

SRVCTLを使用してOracle Restart構成にコンポーネントを追加する場合、コンポーネントのオプション構成設定を指定できます。

SRVCTLを使用してOracle Restart構成にコンポーネントを追加するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

   ```
   srvctl add object options
   ```

   objectは表4-7に示したコンポーネントの1つです。各コンポーネントに利用できるオプションについては、SRVCTLのaddコマンドを参照してください。

例4-1 データベースの追加

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースを追加します。必須の-oraclehomeオプションではOracleホームの場所を指定します。

```
srvctl add database -db dbcrm -oraclehome
/u01/app/oracle/product/database_release_number/dbhome_1
```

例4-2 データベース・サービスの追加

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースに対し、crmbatchという新しいデータベース・サービスを作成し、Oracle Restart構成に追加します。

```
srvctl add service -db dbcrm -service crmbatch
```

他の例については、「SRVCTLを使用したデータベース・サービスの作成と削除」を参照してください。

例4-3 デフォルト・リスナーの追加

この例ではOracle Restart構成にデフォルト・リスナーを追加します。

```
srvctl add listener
```

ノート:

Oracle Restart 構成にデータベースをインストールするか、手動でデータベースを追加するときに、Oracle Grid Infrastructure のインストール環境の所有者ユーザーが別個にある場合は、Oracle Grid Infrastructure コンポーネントがデータベースに接続できるようにするために、そのデータベースの OSRACDBA グループのメンバーとしてグリッド・ユーザーを追加する必要もあります。これは、Oracle Grid Infrastructure コンポーネントが、データベースを起動および停止するために、データベースに SYSRAC として接続できる必要があるためです。

たとえば、Oracle Grid Infrastructure ホームをインストールしたホスト・ユーザーの名前が `grid` で、Oracle ホームの OSRACDBA の名前が `racdba` である場合、ユーザー`grid` は `racdba` グループのメンバーであることが必要です。

関連項目:

- 「Oracle Restartで管理されているコンポーネントの起動と停止」
- 「オペレーティング・システム・グループ」
- 「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.5 Oracle Restart構成からのコンポーネントの削除

Oracleでお薦めする方法を使用してOracleコンポーネントを削除した場合、そのコンポーネントはOracle Restart構成からも自動的に削除されます。

たとえば、Database Configuration Assistant (DBCA)を使用してデータベースを削除すると、DBCAによって、Oracle Restart構成からデータベースが削除されます。同様に、Oracle Net Configuration Assistant(NETCA)を使用してリスナーを削除すると、NETCAによって、Oracle Restart構成からリスナーが削除されます。他の例については、表4-3を参照してください。推奨以外の方法や手動でOracleコンポーネントを削除する場合、最初にSRVCTLを使用してOracle Restart構成からコンポーネントを削除する必要があります。そうしないと、エラーが発生することがあります。

Oracle Restart構成からコンポーネントを削除するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

   ```
   srvctl remove object [options]
   ```

   `object`は表4-7に示したコンポーネントの1つです。各コンポーネントに利用できるオプションについては、SRVCTLのremoveコマンドを参照してください。

例4-4 データベースの削除

この例ではDB_UNIQUE_NAMEが`dbcrm`であるデータベースを削除します。

```
srvctl remove database -db dbcrm
```

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.6 Oracle Restartでのコンポーネント管理の無効化と有効化

Oracle Restartでのコンポーネント管理は一時的に無効にできます。この機能を実行するケースの1つとして、コンポーネントのメンテナンスを実行する場合があります。たとえば、コンポーネントの修復が必要な場合、障害の発生時やホスト・コンピュータの再起動時にコンポーネントを自動的に再起動しない方がよいことがあります。メンテナンスの完了後、コンポーネントの管理を再度有効にできます。

コンポーネントを無効にすると、次のようになります。

- 自動的に再起動しなくなります。
- 依存性によって自動的に起動しなくなります。
- SRVCTLで起動できません。
- このリソースに依存するコンポーネントは、自動的に開始または再起動しなくなります。

コンポーネントの自動再起動を無効または有効にするには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のいずれかを行います:
   - コンポーネントを無効にするには、次のコマンドを入力します。

     ```
     srvctl disable object [options]
     ```

   - コンポーネントを有効にするには、次のコマンドを入力します。

     ```
     srvctl enable object [options]
     ```

   `object`を表4-7に示したコンポーネントの1つと置き換えます。各コンポーネントに利用できるオプションについては、SRVCTLのdisableコマンドおよびenableコマンドを参照してください。

例4-5 データベースの自動再起動の無効化

この例では`DB_UNIQUE_NAME`が`dbcrm`であるデータベースの自動再起動を無効にします。

```
srvctl disable database -db dbcrm
```

例4-6 Oracle ASMディスク・グループの自動再起動の無効化

この例では、`recovery`というOracle ASMディスク・グループの自動再起動を無効にします。

```
srvctl disable diskgroup -diskgroup recovery
```

例4-7 Oracle ASMディスク・グループの自動再起動の有効化

この例では、`recovery`というディスク・グループの自動再起動を再度有効にします。

```
srvctl enable diskgroup -diskgroup recovery
```

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.7 コンポーネント・ステータスの表示

SRVCTLを使用すると、Oracle Restartで管理されているコンポーネントの実行ステータス(実行中または実行していない)を表示できます。一部のコンポーネントでは、他の情報も表示されます。

コンポーネント・ステータスを表示するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

```
srvctl status object [options]
```

`object`は表4-7に示したコンポーネントの1つです。各コンポーネントに利用できるオプションについては、SRVCTLのstatusコマンドを参照してください。

例4-8 データベースのステータスの表示

この例では、`DB_UNIQUE_NAME`が`dbcrm`であるデータベースのステータスを表示します。

```
srvctl status database -db dbcrm
Database is running.
```

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.8 コンポーネントのOracle Restart構成の表示

SRVCTLを使用すると、任意のコンポーネントのOracle Restart構成を表示できます。Oracle Restartではコンポーネント・タイプごとに異なる構成情報を保持します。SRVCTLコマンドの1形式を使用して、Oracle Restartで管理されているコンポーネントのリストを取得できます。

コンポーネントの構成を表示するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

```
srvctl config object options
```

`object`は表4-7に示したコンポーネントの1つです。各コンポーネントに利用できるオプションについては、SRVCTLのconfigコマンドを参照してください。

例4-9 Oracle Restartで管理されているすべてのデータベースのリストの表示

```
srvctl config database
dbcrm
orcl
```

例4-10 特定のデータベースの構成の表示

この例では、DB_UNIQUE_NAMEがorclであるデータベースの構成を表示します。

```
srvctl config database -db orcl
Database unique name: orcl
Database name: orcl
Oracle home: /u01/app/oracle/product/database_release_number/dbhome_1
Oracle user: oracle
Spfile: +DATA/orcl/spfileorcl.ora
Domain: us.example.com
Start options: open
Stop options: immediate
Database role:
Management policy: automatic
Disk Groups: DATA
Services: mfg,sales
```

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.9 コンポーネントのOracle Restart構成の変更

SRVCTLを使用して、コンポーネントのOracle Restart構成を変更できます。たとえば、Oracle Restartがリスナーを起動したときにリスナーがリスニングするポート番号や、Oracle Restartがデータベースを起動する際に指し示すサーバー・パラメータ・ファイル(SPFILE)を変更できます。

コンポーネントのOracle Restart構成を変更するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

   ```
   srvctl modify object options
   ```

   objectは表4-7に示したコンポーネントの1つです。各コンポーネントに利用できるオプションについては、SRVCTLのmodifyコマンドを参照してください。

例4-11 データベースのOracle Restart構成の変更

DB_UNIQUE_NAMEがdbcrmであるデータベースの場合、次のコマンドを使用すると、管理ポリシーはMANUALに、起動オプションはNOMOUNTに変更されます。

```
srvctl modify database -db dbcrm -policy MANUAL -startoption NOMOUNT
```

MANUAL管理ポリシーを設定すると、データベース・ホスト・コンピュータの再起動時にデータベースが自動的に起動されなくなります。ただし、Oracle Restartでデータベースの監視が継続され、データベースに障害が発生すると再起動されます。

関連項目:

- 「コンポーネントのOracle Restart構成の表示」
- 「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.10 Oracle Restart構成の環境変数の管理

Oracle Restart構成には環境変数の名前と値のペアを格納できます。

- Oracle Restart構成の環境変数について
  Oracle Restart構成に環境変数値を設定できます。
- 環境変数の設定と設定解除
  SRVCTLを使用して、コンポーネントに対するOracle Restart構成の環境変数値を設定したり、設定を解除します。
- 環境変数の表示
  SRVCTLを使用して、コンポーネントに対するOracle Restart構成の環境変数の値を表示します。

親トピック: Oracle Restartの構成

4.2.10.1 Oracle Restart構成の環境変数について

Oracle Restart構成に環境変数値を設定できます。

通常、Oracle Databaseを起動する前に環境変数(ORACLE_HOMEとORACLE_SIDを除く)を設定する場合、これらの環境変数値をOracle Restart構成に設定できます。次のコンポーネントの個々の構成には任意の数の環境変数を格納できます。

- データベース・インスタンス
- リスナー
- Oracle ASMインスタンス

前述のコンポーネントの1つがOracle Restartで起動されると、そのコンポーネントの環境変数はコンポーネント構成に格納された値に最初に設定されます。Oracleコンポーネントで使用される環境変数はこの方法で設定できますが、この機能の主な対象はオペレーティング・システムの環境変数です。

次の項では、環境変数の設定、設定解除および表示の手順を説明しています。

- 環境変数の設定と設定解除
- 環境変数の表示

> ノート:
>
> この機能は ORACLE_HOME、ORACLE_SID などの標準環境変数の設定には使用しないでください。これらの環境変数は Oracle Restart で自動的に設定されます。

親トピック: Oracle Restart構成の環境変数の管理

4.2.10.2 環境変数の設定と設定解除

SRVCTLを使用して、コンポーネントに対するOracle Restart構成の環境変数値を設定したり、設定を解除します。

構成の環境変数の設定または設定解除を実行するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のいずれかを行います:
   - 構成の環境変数を設定するには、次のコマンドを入力します。

```
srvctl setenv {asm|database|listener} options
```

- 構成から環境変数を削除するには、次のコマンドを入力します。

```
srvctl unsetenv {asm|database|listener} options
```

各コンポーネントに利用できるオプションについては、SRVCTLのsetenvコマンドおよびunsetenvコマンドを参照してください。

例4-12 データベース環境変数の設定

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのOracle Restart構成にNLS_LANGおよびAIX AIXTHREAD_SCOPE環境変数を設定します。

```
srvctl setenv database -db dbcrm -envs "NLS_LANG=AMERICAN_AMERICA.AL32UTF8,
   AIXTHREAD_SCOPE=S"
```

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restart構成の環境変数の管理

4.2.10.3 環境変数の表示

SRVCTLを使用して、コンポーネントに対するOracle Restart構成の環境変数の値を表示します。

構成の環境変数値を表示するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

```
srvctl getenv {database|listener|asm} options
```

各コンポーネントに利用できるオプションについては、SRVCTLのgetenvコマンドを参照してください。

例4-13 データベースのすべての環境変数の表示

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのOracle Restart構成にある環境変数を取得および表示します。

```
srvctl getenv database -db dbcrm
dbcrm:
NLS_LANG=AMERICAN_AMERICA
AIXTHREAD_SCOPE=S
GCONF_LOCAL_LOCKS=1
```

例4-14 データベースの特定の環境変数の表示

この例では、同じデータベースのOracle Restart構成からNLS_LANGおよびAIXTHREAD_SCOPE環境変数を取得および表示します。

```
srvctl getenv database -db dbcrm -envs "NLS_LANG,AIXTHREAD_SCOPE"
dbcrm:
NLS_LANG=AMERICAN_AMERICA
AIXTHREAD_SCOPE=S
```

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restart構成の環境変数の管理

4.2.11 SRVCTLを使用したデータベース・サービスの作成と削除

Oracle Restartを使用してデータベースを管理する場合、データベース・サービスの作成と削除にはSRVCTLを使用することをお薦めします。SRVCTLを使用してデータベース・サービスを追加すると、サービスはOracle Restart構成に自動的に追加され、サービスとデータベース間の依存性が確立されます。したがって、サービスの起動時にデータベースが起動されていない場合、Oracle Restartで最初にデータベースが起動されます。

SRVCTLを使用してデータベース・サービスを削除すると、サービスはOracle Restart構成からも削除されます。

SRVCTLを使用してデータベース・サービスを作成するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

   ```
   srvctl add service -db db_unique_name -service service_name [options]
   ```

   データベース・サービスが作成され、Oracle Restart構成に追加されます。利用できるオプションについては、srvctl add serviceコマンドを参照してください。

SRVCTLを使用してデータベース・サービスを削除するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のコマンドを入力します。

   ```
   srvctl remove service -db db_unique_name -service service_name [-force]
   ```

   データベース・サービスがOracle Restart構成から削除されます。`-force`フラグがあると、サービスが実行中の場合でも削除されます。このフラグがないと、サービスが実行の場合にはエラーが発生します。

例4-15 データベース・サービスの作成

この例では、`DB_UNIQUE_NAME`が`dbcrm`であるデータベースに対して`crmbatch`という新しいデータベース・サービスを作成します。

```
srvctl add service -db dbcrm -service crmbatch
```

例4-16 ロールベースのデータベース・サービスの作成

この例では、`crmbatch`データベース・サービスを作成し、`PHYSICAL_STANDBY`のData Guardロールを割り当てます。`dbcrm`データベースの現在のロールがフィジカル・スタンバイである場合にのみ、このサービスは自動的に起動されます。

```
srvctl add service -db dbcrm -service crmbatch -role PHYSICAL_STANDBY
```

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.12 Oracle Restart環境でのFANイベントの有効化

Oracle Restartを有効にして高速アプリケーション通知(FAN)イベントを公開するには、Oracle Restartサーバーおよび統合されているクライアントを含むOracle Notification Services(ONS)ネットワークを作成する必要があります。

このクライアントには、Oracle Connection Manager(CMAN)、Java Database Connectivity(JDBC)およびUniversal Connection Pool(UCP)クライアントを含めることができます。Oracle Call InterfaceまたはODP.NETクライアントを使用している場合は、サービスに対してOracle Advanced Queuing(AQ)HA通知を有効にする必要があります。また、ONSがサーバー上で稼働している必要があります。

Oracle Restart環境でFANイベントを有効にするには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. データベースがすでにOracle Restartで管理されていない場合は、データベースをOracle Restart構成に追加します。「Oracle Restart構成へのコンポーネントの追加」を参照してください。
3. ONSを構成に追加します。

   ```
   srvctl add ons
   ```

   ONSは、追加時には無効になっています。
4. ONSの有効化:

   ```
   srvctl enable ons
   ```
5. ONSの起動:

   ```
   srvctl start ons
   ```
6. Oracle Restart構成にサービスを追加します。

   Oracle Call InterfaceおよびODP.NETクライアントについては、データベース・キューを有効にするため-notificationオプションがTRUEに設定されていることを確認します。

   「SRVCTLを使用したデータベース・サービスの作成と削除」を参照してください。
7. 各クライアントで高速接続フェイルオーバーを有効化します。「クライアントでの高速接続フェイルオーバーの有効化」を参照してください。

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.13 プライマリ・データベースとスタンバイ・データベースの接続のフェイルオーバーの自動化

Oracle RestartおよびOracle Data Guardのプライマリ・データベースとスタンバイ・データベースを使用する構成では、スイッチオーバーまたはフェイルオーバーの際にデータベース・サービスがプライマリからスタンバイに自動的にフェイルオーバーします。

Oracle Notification Services(ONS)を使用すると、プライマリ・データベースとスタンバイ・データベース間におけるサービスのフェイルオーバーをクライアントに即座に通知できます。Oracle Data Guard Brokerでは、フェイルオーバーが発生すると、高

速アプリケーション通知(FAN)を使用してクライアントに通知が送信されます。統合されているOracle Clientでは、接続が自動的にフェイルオーバーされ、アプリケーションによって障害をエンド・ユーザーからマスクできます。

接続のフェイルオーバーを自動化するには、Oracle Restartサーバーおよび統合されているクライアント(CMAN、リスナー、JDBCおよびUCP)を含むONSネットワークを作成する必要があります。Oracle Call InterfaceまたはODP.NETクライアントを使用している場合は、Oracleアドバンスト・キューイングのキューを有効にする必要があります。サービスのフェイルオーバーを自動化するには、データベースおよびサービスをOracle RestartとOracle Data Guard Brokerで管理する必要があります。

プライマリ・データベースとスタンバイ・データベース間におけるサービスのフェイルオーバーを自動化するには:

1. Oracle Data Guard Brokerでプライマリ・データベースとスタンバイ・データベースを構成します。『Oracle Data Guard Broker』を参照してください。
2. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
3. プライマリ・サーバー上のOracle Restart構成にプライマリ・データベースが追加されていない場合は、追加します。データベース・ロールには、PRIMARYを指定してください。「Oracle Restart構成へのコンポーネントの追加」を参照してください。
4. スタンバイ・サーバー上のOracle Restart構成にスタンバイ・データベースが追加されていない場合は、追加します。適切なスタンバイ・データベース・ロールを指定してください。
5. プライマリ・データベース・サーバーおよびスタンバイ・データベース・サーバーの両方で、FANイベントを有効にします。「Oracle Restart環境でのFANイベントの有効化」を参照してください。
6. プライマリ・データベースおよびスタンバイ・データベース上のOracle Restart構成のデータベースに接続するためにクライアントで使用するサービスを追加します。サービスを追加する場合は、次のことを確認してください。
   - サービスごとに、`-role`オプションが適切なロールに設定されていること
   - ODP.NETまたはOracle Call Interfaceを使用している場合は、`-notification`オプションがTRUEに設定されていること

   「SRVCTLを使用したデータベース・サービスの作成と削除」を参照してください。
7. 各クライアントで高速接続フェイルオーバーを有効化します。「クライアントでの高速接続フェイルオーバーの有効化」を参照してください。

関連項目:

「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: Oracle Restartの構成

4.2.14 クライアントでの高速接続フェイルオーバーの有効化

高速接続フェイルオーバーは、JDBC、OCIまたはODP.NETを使用するクライアントなど、高速アプリケーション通知(FAN)に統合されたクライアントに高可用性を提供します。高速接続フェイルオーバーを使用するようにクライアントを構成した場合、クライアントが自動的にFANイベントをサブスクライブし、データベースのUPイベントおよびDOWNイベントに対処できます。それに対応して、Oracle Databaseは、要求されたデータベース・サービスを提供するアクティブ・インスタンスにクライアントを接続します。

- クライアントでの高速接続フェイルオーバーの有効化について
  スタンバイ・データベースが存在する構成では、Oracle Notification Services (ONS)をOracle Restart構成に追加した後でサービスに対してOracle Advanced Queuing (AQ) HA通知を有効にすると、クライアントで高速接続フェイルオーバーを有効化できます。

- JDBCクライアントでの高速接続フェイルオーバーの有効化
  Oracle Universal Connection PoolでFANを有効にすると、クライアントで高速接続フェイルオーバー(FCF)が有効になります。アプリケーションでは、シックまたはシンのいずれかのJDBCクライアントを使用してFCFを使用できます。
- Oracle Call Interfaceクライアントでの高速接続フェイルオーバーの有効化
  Oracle Call Interfaceクライアントは、Oracle Restart高可用性FANイベントの発生時に通知を受信して応答するように登録することで、高速接続フェイルオーバー(FCF)を有効にできます。
- ODP.NETクライアントでの高速接続フェイルオーバーの有効化
  Oracle Data Provider for .NET(ODP.NET)接続プールでは、サービスが停止したことを示す通知をサブスクライブできます。DOWNイベントが発生すると、インスタンスが停止する接続プール内のセッションがOracle Databaseによってクリーン・アップされ、有効でなくなる接続がODP.NETによって事前に処理されます。

親トピック: Oracle Restartの構成

4.2.14.1 クライアントでの高速接続フェイルオーバーの有効化について

スタンバイ・データベースが存在する構成では、Oracle Notification Services(ONS)をOracle Restart構成に追加した後でサービスに対してOracle Advanced Queuing(AQ)HA通知を有効にすると、クライアントで高速接続フェイルオーバーを有効化できます。

クライアントでは、高速アプリケーション通知(FAN)イベントを受信し、Oracle Data Guardのフェイルオーバーの後で現在のプライマリ・データベースへの接続を再配置できます。ONSの追加については、「プライマリ・データベースとスタンバイ・データベースの接続のフェイルオーバーの自動化」を参照してください。

スタンバイ・データベースが構成されていないデータベースでも、クライアントのFANイベントを構成できます。フェイルオーバーが発生した場合、データベースへの接続を再試行するようにクライアントを構成できます。障害が発生したデータベースはOracle Restartによって再起動されるため、データベースが再起動した時点でクライアントは再接続できます。この項の例で示すように、接続文字列に適切な遅延と再試行を確実にプログラムしてください。

親トピック: クライアントでの高速接続フェイルオーバーの有効化

4.2.14.2 JDBCクライアントでの高速接続フェイルオーバーの有効化

Oracle Universal Connection PoolでFANを有効にすると、クライアントで高速接続フェイルオーバー(FCF)が有効になります。アプリケーションでは、シックまたはシンのいずれかのJDBCクライアントを使用してFCFを使用できます。

JDBCクライアントを構成するには、データソースに対して最初の`getConnection()`要求を行う前に、`FastConnectionFailoverEnabled`プロパティを設定します。高速接続フェイルオーバーを有効にすると、接続キャッシュ内のすべての接続にフェイルオーバーが適用されます。アプリケーションでConnection Cache Managerを使用して接続キャッシュを明示的に作成する場合は、まず`FastConnectionFailoverEnabled`を設定する必要があります。

この項では、Universal Connection Poolを使用してJDBCのFCFを有効にする方法について説明します。シックJDBCクライアントで高速接続フェイルオーバーを有効にする場合は、クライアントおよびサービスの両方で透過アプリケーション・フェイルオーバー(TAF)を有効にしないでください。シンまたはシックJDBCクライアントでFCFを有効にすると、受信用の接続プールが有効になり、すべてのFANイベントが処理されます。

JDBCクライアントで高速接続フェイルオーバーを有効にするには:

1. 次の例に示すように、キャッシュ対応のデータソースで、データソース・プロパティ`FastConnectionFailoverEnabled`を`true`に設定して、Oracle JDBCの暗黙的な接続キャッシュでFANを有効にします。

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
```

```
pds.setONSConfiguration("nodes=primaryhost:6200,standbyhost:6200");
pds.setFastConnectionFailoverEnabled(true);
pds.setURL("jdbc:oracle:thin:@(DESCRIPTION=
  (LOAD_BALANCE=on)
  (ADDRESS=(PROTOCOL=TCP)(HOST=primaryhost)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=standbyhost)(PORT=1521))
  (CONNECT_DATA=(service_name=service_name)))");
......
```

この例では、primaryhostはプライマリ・データベースのサーバーであり、standbyhostはスタンバイ・データベースのサーバーです。

アプリケーションのCLASSPATHには、ucp.jarとons.jarの両方が必要です。

ノート:

データソースに変更を加えずに FAN を有効にするには、システム・プロパティ-D oracle.jdbc.FastConnectionFailover=true を使用します。

2. アプリケーションを開始する際に、ons.jarファイルがアプリケーションのCLASSPATHに存在することを確認してください。ons.jarファイルは、Oracle Clientのインストールの一部です。

関連項目:

- Oracle Database JDBC開発者ガイド
- Oracle Universal Connection Pool開発者ガイド

親トピック: クライアントでの高速接続フェイルオーバーの有効化

4.2.14.3 Oracle Call Interfaceクライアントでの高速接続フェイルオーバーの有効化

Oracle Call Interfaceクライアントは、Oracle Restart高可用性FANイベントの発生時に通知を受信して応答するように登録することで、高速接続フェイルオーバー(FCF)を有効にできます。

これにより、Oracle Call Interfaceのセッション・フェイルオーバーのレスポンス時間が向上し、中断した接続を接続プールおよびセッション・プールから削除できます。この機能は、透過アプリケーション・フェイルオーバー(TAF)、接続プールまたはセッション・プールを使用するものも含めたOracle Call Interfaceアプリケーションで動作します。

まず、高可用性イベントのサービスを有効にして、アドバンスト・キューイングのALERT_QUEUEを自動的に移入する必要があります。アプリケーションでTAFを使用している場合は、サービスのTAF設定を有効にします。Oracle Restartデータベースに接続するようにクライアント・アプリケーションを構成します。クライアントでは、イベントが発生するたびに使用されるコールバックを登録できます。これによって、接続障害が検出されるまでの時間が短縮されます。

Oracle Call Interfaceでは、DOWNイベントの処理時に次の操作が行われます。

- クライアントで影響を受ける接続が終了し、エラーが戻されます。
- Oracle Call Interface接続プールおよびOracle Call Interfaceセッション・プールから接続が削除されます。

  セッション・プールでは各セッションが接続プールの物理接続にマップされており、接続ごとに複数のセッションが存在できます。
- TAFが構成されている場合は、接続がフェイルオーバーされます。

TAFが構成されていない場合は、クライアントのみがエラーを受信します。

ノート:

Oracle Call Interface では、UP イベントは管理されません。

Oracle Call Interfaceクライアントで高速接続フェイルオーバーを有効化するには:

1. 使用しているサービスで、SRVCTLのmodifyコマンドを使用してサービスの値を設定することによって、アドバンスト・キューイング通知が有効になっていることを確認します。たとえば:

```
srvctl modify service -db proddb -service gl.us.example.com -notification
true -role primary -failovertype select -failovermethod basic -failoverretry 5
-failoverdelay 180 -clbgoal long
```

2. 次に示すように、環境作成時にOCI_EVENTSを有効にします。

```
( OCIEnvCreate(...) )
```

3. クライアント・アプリケーションをクライアント・スレッドまたはオペレーティング・システムのライブラリにリンクします。
4. (オプション)クライアントのEVENTコールバックを登録します。
5. クライアントですべてのプライマリ・ホストとスタンバイ・ホストがADDRESS_LISTに含まれているOracle Net接続記述子が使用されていることを確認します。たとえば:

```
gl =
(DESCRIPTION =
 (CONNECT_TIMEOUT=10)(RETRY_COUNT=3)
   (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON1)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO1)(PORT = 1521))
     (LOAD_BALANCE = yes)
)
(CONNECT_DATA=
  (SERVICE_NAME=gl.us.example.com)))
```

アラート情報を表示するには、DBA_OUTSTANDING_ALERTSおよびDBA_ALERT_HISTORYビューを問い合せます。

関連項目:

- Oracle Call Interfaceプログラマーズ・ガイド
- TAFの構成の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

親トピック: クライアントでの高速接続フェイルオーバーの有効化

4.2.14.4 ODP.NETクライアントでの高速接続フェイルオーバーの有効化

Oracle Data Provider for .NET(ODP.NET)接続プールでは、サービスが停止したことを示す通知をサブスクライブできます。DOWNイベントが発生すると、インスタンスが停止する接続プール内のセッションがOracle Databaseによってクリーン・アップされ、有効でなくなる接続がODP.NETによって事前に処理されます。

3つすべてのODP.NETプロバイダ(コア、管理対象および管理対象外)でFCFがサポートされています。

ODP.NETクライアントで高速接続フェイルオーバーを有効にするには:

1. 次の例に示すように、SRVCTLのmodify serviceコマンドを使用することで高速アプリケーション通知(FAN)を有

効にします。

```
srvctl modify service –db dbname –service gl -notification true
```

2. FAN高可用性イベントをサブスクライブすることによって、ODP.NET接続プールの高速接続フェイルオーバーを有効にします。接続時にHAベント接続文字列属性をtrueに設定します。ODP.NETの新しいバージョンでは、HAイベントはデフォルトでtrueに設定されています。プーリング属性はtrue (デフォルト)に設定する必要があります。次にこれらの設定の例を示します。user_nameはユーザーの名前、passwordはユーザーのパスワードを表します。

```
// C#
using System;
using Oracle.ManagedDataAccess.Client;
//using Oracle.DataAccess.Client;

class HAEventEnablingSample
{
  static void Main()
  {
    OracleConnection con = new OracleConnection();

    // Open a connection using ConnectionString attributes
    // Also, enable "load balancing"
    con.ConnectionString =
      "User Id=user_name;Password=password;Data Source=oracle;" +
      "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
      "HA Events=true;Incr Pool Size=5;Decr Pool Size=2";

    con.Open();

    // Create more connections and perform work against the database here.

    // Dispose OracleConnection object
    con.Dispose();
  }
}
```

3. クライアントですべてのプライマリ・ホストとスタンバイ・ホストがADDRESS_LISTに含まれているOracle Net接続記述子が使用されていることを確認します。たとえば:

```
gl =
(DESCRIPTION =
 (CONNECT_TIMEOUT=10)(RETRY_COUNT=3)
   (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON1)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO1)(PORT = 1521))
     (LOAD_BALANCE = yes)
 )
 (CONNECT_DATA=
  (SERVICE_NAME=gl.us.example.com))
 )
)
```

関連項目:

- ODP.NETの詳細は、『Oracle Data Provider for .NET開発者ガイドfor Microsoft Windows』を参照してください。
- 「Oracle RestartのSRVCTLコマンド・リファレンス」

親トピック: クライアントでの高速接続フェイルオーバーの有効化

4.3 Oracle Restartで管理されているコンポーネントの起動と停止

Oracle Restartの使用中は、コンポーネントの開始と停止はSRVCTLユーティリティを使用して実行することを強くお薦めします。

コンポーネントの起動と停止にSRVCTLユーティリティを使用する理由は、次のとおりです。

- SRVCTLを使用してコンポーネントを起動すると、このコンポーネントが依存しているコンポーネントがOracle Restartで最初に起動されます。SRVCTLを使用してコンポーネントを停止すると、Oracle Restartで最初に依存コンポーネントを停止できます。
- SRVCTLでは常にOracle Restart構成に従ってコンポーネントを起動します。他の方法でコンポーネントを起動すると、Oracle Restart構成に従わない可能性があります。

  たとえば、Oracle Restart構成にデータベースを追加したときにサーバー・パラメータ・ファイル(SPFILE)の場所を指定し、この場所がSPFILEのデフォルトの場所ではなかった場合、SQL*Plusを使用してデータベースを起動すると、構成に指定されたSPFILEが使用されないことがあります。

  データベース・インスタンスの構成オプションの表は、srvctl add databaseコマンドを参照してください。
- SRVCTLを使用してコンポーネントを起動すると、コンポーネントのOracle Restart構成に格納されている環境変数が設定されます。

  詳細は、「Oracle Restart構成の環境変数の管理」を参照してください。

SRVCTLを使用して、Oracle Restartで管理されているコンポーネントを起動および停止できます。

SRVCTLを使用してOracle Restartで管理されているコンポーネントを起動または停止するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. 次のいずれかを行います:
   - コンポーネントを起動にするには、次のコマンドを入力します。

     ```
     srvctl start object [options]
     ```

   - コンポーネントを停止するには、次のコマンドを入力します。

     ```
     srvctl stop object [options]
     ```

   `object`は表4-7に示したコンポーネントの1つです。各コンポーネントに利用できるオプションについては、SRVCTLのstartコマンドおよびstopコマンドを参照してください。

例4-17 データベースの起動

この例では、`DB_UNIQUE_NAME`がdbcrmであるデータベースを起動します。

```
srvctl start database -db dbcrm
```

例4-18 データベースNOMOUNTの起動

この例では、データベースをマウントせずにデータベース・インスタンスを起動します。

```
srvctl start database -db dbcrm -startoption nomount
```

例4-19 デフォルト・リスナーの起動

この例では、デフォルト・リスナーを起動します。

```
srvctl start listener
```

例4-20 指定されたリスナーの起動

この例では、crmlistenerというリスナーを起動します。

```
srvctl start listener -listener crmlistener
```

例4-21 データベース・サービスの起動

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのデータベース・サービスbizdevとsupportを起動します。データベースが起動されていない場合、Oracle Restartで最初にデータベースが起動されます。

```
srvctl start service -db dbcrm -service "bizdev,support"
```

例4-22 Oracle ASMディスク・グループの起動(マウント)

この例では、Oracle ASMディスク・グループdataとrecoveryを起動(マウント)します。このコマンドを実行するユーザーはOSASMグループのメンバーであることが必要です。

```
srvctl start diskgroup -diskgroup "data,recovery"
```

例4-23 データベースの停止

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースを停止します。停止オプション(-stopoption)が指定されていないため、データベースはOracle Restart構成の停止オプションに従って停止します。デフォルトの停止オプションはIMMEDIATEです。

```
srvctl stop database -db dbcrm
```

例4-24 ABORTオプションを使用したデータベースの停止

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのSHUTDOWN ABORTを実行します。

```
srvctl stop database -db dbcrm -stopoption abort
```

ノート:

Oracle 実行可能ファイルを再リンクした後、Oracle Restart の使用中は SRVCTL ユーティリティを使用してコンポーネントを開始および停止します。通常、Linux または UNIX ベースのオペレーティング・システムでは、オペレーティング・システムのパッチを適用した後やオペレーティング・システムのアップグレード後に、Oracle 実行可能ファイルの再リンクが必要になります。再リンクの詳細は、『Oracle Database 管理者リファレンス for Linux and UNIX-Based Operating Systems』を参照してください。

SQL*Plus を使用してコンポーネントを開始および停止する場合は、再リンク後に最初に setasmgidwrap スクリプトを実行する必要があります。このスクリプトの実行の詳細は、『Oracle Database アップグレード・ガイド』を参照してください。

関連項目:

SRVCTLのstartコマンド

4.4 メンテナンス操作のためのOracle Restartの停止および再起動

Oracleホームのいくつかのコンポーネントが Oracle Restartで管理されている場合、Oracle RestartおよびOracle Restartで管理されているコンポーネントをOracleホーム内で停止できます。

また、Oracle Restartを無効にして、ノードが再起動した場合にOracle Restartが再起動しないようにすることもできます。この設定は、Oracleホームを含むメンテナンスを実行する場合(パッチのインストールなど)に必要になることがあります。メンテナンス操作の完了後、Oracle Restartを有効にして再起動し、Oracle Restartで管理されているコンポーネントをOracleホーム内で再起動できます。

停止と起動の操作にはSRVCTLユーティリティとCRSCTLユーティリティを使用します。

- `stop home` SRVCTLコマンドでは、Oracle Restartで管理されているすべてのコンポーネントを、指定したOracleホーム内で停止します。`start home` SRVCTLコマンドではこれらのコンポーネントを起動します。Oracleホームは Oracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。

  homeオブジェクトを使用すると、`-statefile`オプションで指定した状態ファイルによって各コンポーネントの状態が追跡されます。`stop`および`status`コマンドでは状態ファイルを作成します。`start`コマンドでは状態ファイルを使用して再起動するコンポーネントを特定します。

  さらに、`status home`コマンドを使用して、Oracle Restartで管理されているコンポーネントのステータスを確認できます。

- `stop` CRSCTLコマンドはOracle Restartを停止し、`disable` CRSCTLコマンドはOracle Restartで管理されるコンポーネントが自動的に再起動しないようにします。`enable` CRSCTLコマンドは自動再起動を有効にし、`start` CRSCTLコマンドはOracle Restartを再起動します。

パッチのインストール中にOracleホームのコンポーネントを停止および起動するには:

1. 「SRVCTLの実行準備」で説明したように、SRVCTLの実行を準備します。
2. Oracle Restartで管理されているコンポーネントをOracleホーム内で停止するには、次のSRVCTLユーティリティを使用します。

   ```
   srvctl stop home -oraclehome oracle_home -statefile state_file [-stopoption
   stop_options] [-force]
   ```

   ここで、`oracle_home`はOracleホームの完全なパスを表し、`state_file`は状態ファイルの完全なパスを表します。Oracleホームの状態情報は、指定した状態ファイルに記録されます。状態ファイルの場所は、ステップ7で指定する必要があるのでノートにとっておいてください。

   Oracle Gridインフラストラクチャ・ホームのコンポーネントを停止する前に、依存するOracle Databaseホームのコンポーネントを先に停止する必要があります。

3. Oracle Gridインフラストラクチャ・ホームにパッチを適用している場合は、Oracle Restartを無効にして停止します。それ以外の場合は、ステップ4に進みます。

   Oracle Restartを無効にして停止するには、CRSCTLユーティリティを使用して次のコマンドを実行します。

   ```
   crsctl disable has
   crsctl stop has
   ```

4. メンテナンス操作を実行します。

5. Oracle Restartで管理されているコンポーネントの自動再起動を有効にするには、次のCRSCTLユーティリティを使用します。

```
crsctl enable has
```

6. Oracle Restartを起動するには、次のCRSCTLユーティリティを使用します。

```
crsctl start has
```

7. ステップ2で停止されたコンポーネントを起動するには、次のSRVCTLユーティリティを使用します。

```
srvctl start home -oraclehome oracle_home -statefile state_file
```

状態ファイルは、ステップ2で指定した状態ファイルと一致している必要があります。

8. (オプション)Oracle Restartで管理されているコンポーネントのステータスをOracleホーム内で確認するには、次のSRVCTLユーティリティを使用します。

```
srvctl status home -oraclehome oracle_home -statefile state_file
```

例4-25 Oracle Restartで管理されているコンポーネントのOracleホーム内での停止

```
srvctl stop home -oraclehome /u01/app/oracle/product/database_release_number/dbhome_1
-statefile /usr1/or_state
```

例4-26 Oracle Restartで管理されているコンポーネントのOracleホーム内での起動

```
srvctl start home -oraclehome
/u01/app/oracle/product/database_release_number/dbhome_1 -statefile /usr1/or_state
```

例4-27 Oracle Restartで管理されているコンポーネント・ステータスのOracleホーム内での表示

```
srvctl status home -oraclehome
/u01/app/oracle/product/database_release_number/dbhome_1 -statefile /usr1/or_state
```

関連項目:

- srvctl stop homeコマンド
- srvctl status homeコマンド
- srvctl start homeコマンド
- 「CRSCTLコマンド・リファレンス」

親トピック: Oracle Databaseの自動再起動の構成

4.5 Oracle RestartのSRVCTLコマンド・リファレンス

Oracle Restartに固有のSRVCTLコマンドの構文とオプションの詳細を参照できます。

SRVCTLコマンドの詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

SRVCTLコマンドの構文とオプションの概要

SRVCTLでは、次のコマンド構文が想定されます。

```
srvctl command object options
```

ここで:

- commandは`start`、`stop`、`remove`などの動詞です。
- objectはデータベース、リスナーなど、SRVCTLコマンドの実行対象コンポーネントです。コンポーネントの略称も使用できます。コンポーネントとその略称の詳細なリストは、表4-7を参照してください。
- optionsは、コマンドの追加パラメータを使用できるように、前述のコマンドの組合せの使用範囲を拡大します。たとえば、`-db`オプションはデータベースの一意名が後に続くことを示し、`-service`オプションはデータベース・サービス名のカンマ区切りリストが後に続くことを示しています。

ノート:

Windows プラットフォームでカンマ区切りのリストを指定する場合、リストを二重引用符で囲む必要があります("...,...")。また、リスト・メンバーにシェルのメタ文字が含まれている場合、UNIX と Linux プラットフォームでも二重引用符を使用する必要があります。

大/小文字の区別

SRVCTLコマンドおよびコンポーネントでは大/小文字が区別されません。オプションでは大/小文字が区別されます。データベースとデータベース・サービス名では大/小文字が区別されず、大/小文字が保持されます。

コマンド・パラメータ入力ファイル

コマンド・パラメータは、コマンドラインで直接指定するかわりに、ファイルに指定することもできます。次のような状況では、コマンド・パラメータ入力ファイルを使用すると役立ちます。

- 非常に長いパラメータ値または非常に多数のパラメータを指定してコマンドを実行する場合。
- 特定の特殊文字のシェル処理を省略する場合。

コマンド・パラメータ入力ファイルを指定するには、コマンド・パラメータ・ファイルの場所である値とともに`-file`パラメータを使用します。SRVCTLでは、コマンドラインではなく、コマンド・パラメータ・ファイルのコマンド・パラメータが処理されます。

SRVCTLコンポーネントの概要

表4-7に、SRVCTLコマンドの`object`部分に使用できるキーワードを示します。各コンポーネント・キーワードの完全名と略称のいずれかを使用できます。

表4-7 コンポーネント・キーワードと略称

コンポーネント	略称	説明
asm	asm	Oracle ASM インスタンス
database	db	データベース・インスタンス
diskgroup	dg	Oracle ASM ディスク・グループ
home	home	Oracle ホームまたは Oracle Clusterware ホーム

コンポーネント	略称	説明
listener	lsnr	Oracle Netリスナー
service	serv	データベース・サービス
ons	ons	Oracle Notification Services(ONS)

- add
  srvctl addコマンドでは、指定したコンポーネントをOracle Restart構成に追加します。また、必要に応じてコンポーネントのOracle Restart構成パラメータを設定します。コンポーネントを追加した後、Oracle Restartで管理が開始され、コンポーネントは必要に応じて再起動されます。
- config
  srvctl configコマンドでは、指定したコンポーネントまたはコンポーネント・セットのOracle Restart構成を表示します。
- disable
  コンポーネントを無効にし、そのコンポーネントのOracle Restartでの管理を一時停止します。
- downgrade
  手動でデータベースをダウングレードした後に、srvctl downgradeコマンドでデータベース構成をダウングレードします。
- enable
  srvctl enableコマンドは、指定した無効なコンポーネントを再有効化します。
- getenv
  データベース、リスナーまたはOracle ASMインスタンスのOracle Restart構成から環境変数とその値を取得して表示します。
- modify
  コンポーネントのOracle Restart構成を変更します。変更内容は、コンポーネントが次に再起動されたときに反映されます。
- remove
  Oracle Restart構成から指定したコンポーネントを削除します。コンポーネントはOracle Restartで管理されなくなります。コンポーネントの環境変数設定も削除されます。
- setenv
  setenvコマンドでは、データベース、リスナーまたはOracle ASMインスタンスのOracle Restart構成に環境変数の値を設定します。
- start
  指定したコンポーネントを起動します。
- status
  指定したコンポーネントまたはコンポーネント・セットの実行ステータスを表示します。
- stop
  指定したコンポーネントを停止します。
- unsetenv
  unsetenvコマンドでは、データベース、リスナーまたはOracle ASMインスタンスのOracle Restart構成から1つ以上の環境変数を削除します。

- update
  srvctl updateコマンドは、実行中のデータベースを更新して指定の起動オプションに切り替えます。
- upgrade
  srvctl upgradeコマンドは、リソース・タイプとリソースを古いバージョンから新しいバージョンにアップグレードします。

関連項目:

表4-1

親トピック: Oracle Databaseの自動再起動の構成

4.5.1 add

srvctl addコマンドでは、指定したコンポーネントをOracle Restart構成に追加します。また、必要に応じてコンポーネントのOracle Restart構成パラメータを設定します。コンポーネントを追加した後、Oracle Restartで管理が開始され、コンポーネントは必要に応じて再起動されます。

srvctl add操作を実行するには、適切なユーザー・アカウントを使用してデータベース・ホスト・コンピュータにログインする必要があります。詳細は、「SRVCTLの実行準備」を参照してください。

ノート:

Oracle ASM ディスク・グループには、srvctl add コマンドはありません。ディスク・グループは、初回マウント時に自動的に Oracle Restart 構成に追加されます。Oracle Restart 構成からディスク・グループを削除した後に再度追加する場合は、SQL*Plus を使用して Oracle ASM インスタンスに接続し、ALTER DISKGROUP...MOUNT コマンドを使用します。

- srvctl add asm
  Oracle Restart構成にOracle ASMインスタンスを追加します。
- srvctl add database
  Oracle Restart構成にデータベースを追加します。
- srvctl add listener
  Oracle Restart構成にリスナーを追加します。
- srvctl add ons
  Oracle Restart構成にOracle Notification Services (ONS)を追加します。
- srvctl add service
  Oracle Restart構成にデータベース・サービスを追加します。

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.1.1 srvctl add asm

Oracle Restart構成にOracle ASMインスタンスを追加します。

- 構文およびオプション
- 例

親トピック: add

4.5.1.1.1 構文およびオプション

srvctl add asmコマンドは次の構文で使用します。

```
srvctl add asm [-listener listener_name] [-spfile spfile]
  [-pwfile password_file_path] [-diskstring asm_diskstring]
```

表4-8 srvctl add asmのオプション

オプション	説明
-listener listener_name	Oracle ASM が登録する必要があるリスナーの名前。このリスナーとの弱い依存性が確立されます。(Oracle ASM インスタンスを起動する前に、リスナーの起動が試行されます。リスナーが起動しない場合でも、Oracle ASM インスタンスは起動されます。後でリスナーに障害が発生した場合、Oracle Restart で Oracle ASM は再起動されません。) 省略すると、デフォルトで listener というリスナーに設定されます。
-spfile spfile	データベースのサーバー・パラメータ・ファイルのフルパス。省略すると、デフォルトの SPFILE が使用されます。
-pwfile password_file_path	Oracle ASM パスワード・ファイルのフルパス。
-diskstring asm_diskstring	Oracle ASM ディスク・グループ検出文字列。Oracle ASM 検出文字列は、Oracle ASM インスタンスによって検出されるディスクのセットを制限する文字列のカンマ区切りリストです。検出文字列には、ワイルドカード文字を含めることができます。文字列のいずれかに一致するディスクのみが検出されます。

親トピック: srvctl add asm

4.5.1.1.2 例

次に、このコマンドの例を示します。

```
srvctl add asm -listener crmlistener
```

関連項目:

Oracle ASMのディスク・グループ検出文字列の詳細は、Oracle Automatic Storage Management管理者ガイドを参照してください。

親トピック: srvctl add asm

4.5.1.2 srvctl add database

Oracle Restart構成にデータベースを追加します。

Oracle Restart構成にデータベースを追加した後、データベースからOracle ASMディスク・グループのデータにアクセスした場合、データベースとディスク・グループ間の依存性が作成されます。その後、データベースの起動前にディスク・グループがマウントさ

れていることがOracle Restartで確認されます。

ただし、Oracle Restart構成にデータベースを追加するときにデータベースとOracle ASMインスタンスが実行されていない場合、SRVCTLコマンドに-diskgroupオプションを指定して、データベースとディスク・グループ間の依存性を手動で確立する必要があります。この後の例を参照してください。

ノート:

Oracle Restart の構成に手動でデータベースを追加する場合は、そのデータベースの OSDBA グループのメンバーとして Oracle Grid インフラストラクチャ・ソフトウェアの所有者を追加することも必要となります。これは、Grid インフラストラクチャ・コンポーネントが、データベースを起動および停止するために、データベースに SYSDBA として接続できる必要があるためです。

たとえば、Grid インフラストラクチャをインストールしたホスト・ユーザーの名前が grid で、新しいデータベースの OSDBA グループの名前が dba である場合、ユーザーgrid は dba グループのメンバーであることが必要です。

- 構文およびオプション
- 例

親トピック: add

4.5.1.2.1 構文およびオプション

srvctl add databaseコマンドは、次の構文で使用します。

```
srvctl add database -db db_unique_name -oraclehome oracle_home
  [-domain domain_name] [-dbname db_name] [-instance instance_name]
  [-spfile spfile][-pwfile password_file_path] [-startoption start_options]
  [-stopoption stop_options]
  [-role {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY |
          SNAPSHOT_STANDBY | FAR_SYNC}]
  [-policy {AUTOMATIC | MANUAL | NORESTART}] [-diskgroup disk_group_list]
  [-verbose]
```

表4-9 srvctl add databaseのオプション

構文	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-oraclehome oracle_home	データベースの Oracle ホームのフルパス。
-domain domain_name	データベースのドメイン。DB_DOMAIN 初期化パラメータに一致する必要があります。
-dbname db_name	指定する場合、DB_NAME 初期化パラメータの設定に一致する必要があります。DB_NAME が-db オプションで指定された一意の名前と異なる場合、このオプションを指定

構文	説明
	する必要があります。
-instance instance_name	インスタンス名。 インスタンス名が-db オプションで指定された一意の名前と異なる場合、このオプションを指定する必要があります。たとえば、一意の名前にアンダースコアが含まれていて、インスタンス名がアンダースコアを省略している場合、このパラメータを使用してインスタンス名を指定します。
-spfile spfile	データベースのサーバー・パラメータ・ファイルのフルパス。省略すると、デフォルトの SPFILE が使用されます。
-pwfile password_file_path	データベース・パスワード・ファイルのフルパス。
-startoption start_options	データベースの起動オプション(OPEN、MOUNT または NOMOUNT)。省略すると、デフォルトで OPEN に設定されます。 関連項目: 起動オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。
-stopoption stop_options	データベースの停止オプション(NORMAL、IMMEDIATE、TRANSACTIONAL または ABORT)。省略すると、デフォルトで IMMEDIATE に設定されます。 関連項目: 停止オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』も参照してください。
-role {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY | FAR_SYNC}	データベースの現在のロール(PRIMARY、PHYSICAL_STANDBY、LOGICAL_STANDBY、SNAPSHOT_STANDBY または FAR_SYNC)。デフォルトは PRIMARY です。Oracle Data Guard 環境でのみ適用できます。 関連項目: データベース・ロールの詳細は、『Oracle Data Guard 概要および管理』を参照してください。
-policy {AUTOMATIC | MANUAL | NORESTART}	データベースの管理ポリシー。 ● AUTOMATIC(デフォルト): データベースは、データベース・ホスト・コンピュータの再起動時に前回の実行状態(起動または停止)へ自動的に戻ります。 ● MANUAL: データベース・ホスト・コンピュータの再起動時にデータベースは自動的には再起動されません。MANUAL を設定しても、データベースは実行中に Oracle

構文	説明
	Restart で監視され、障害が発生すると再起動されます。 ● NORESTART: MANUAL 設定と同様に、データベース・ホスト・コンピュータの再起動時にデータベースは自動的には再起動されません。ただし、NORESTART 設定では、障害が発生しても、データベースを再起動することはありません。
-diskgroup disk_group_list	データベースが依存するディスク・グループのカンマ区切りのリスト。Oracle Restart では、データベースを起動する前にこれらのディスク・グループがマウントされていることを確認します。このオプションは、データベースの追加時にデータベース・インスタンスと Oracle ASM インスタンスが起動されていなかった場合にのみ必要です。それ以外の場合、データベースとディスク・グループ間の依存性は自動的に記録されています。
-verbose	冗長出力

親トピック: srvctl add database

4.5.1.2.2 例

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースを追加します。

```
srvctl add database -db dbcrm -oraclehome
/u01/app/oracle/product/database_release_number/dbhome_1
```

この例では、同じデータベースを追加し、データベースおよびディスク・グループDATAとRECOVERY間の依存性も確立します。

```
srvctl add database -db dbcrm -oraclehome
/u01/app/oracle/product/database_release_number/dbhome_1
  -diskgroup "DATA,RECOVERY"
```

関連項目:

- 「Oracle RestartとOracle Data Guardとの統合」
- 『Oracle Data Guard概要および管理』

親トピック: srvctl add database

4.5.1.3 srvctl add listener

Oracle Restart構成にリスナーを追加します。

- 構文およびオプション
- 例

親トピック: add

4.5.1.3.1 構文およびオプション

srvctl add listenerコマンドは、次の構文で使用します。

```
srvctl add listener [-listener listener_name] [-endpoints endpoints] [-skip]
  [-oraclehome oracle_home]
```

表4-10 srvctl add listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、デフォルトで LISTENER に設定されます。
`-endpoints endpoints`	カンマ区切りの TCP ポートまたはリスナー・エンドポイント。省略すると、デフォルトで TCP:1521 になります。endpoints の構文は次のとおりです。 `"[TCP:]port[, ...] [/IPC:key] [/NMP:pipe_name] [/TCPS:s_port] [/SDP:port]"`
`-skip`	指定されたエンドポイントとのポートの不一致のチェックをスキップします。
`-oraclehome oracle_home`	リスナーの Oracle ホーム。省略すると、Oracle Grid インフラストラクチャ・ホームが想定されます。

親トピック: srvctl add listener

4.5.1.3.2 例

次のコマンドでは、データベースOracleホームから実行され、TCPポート1522をリスニングしているリスナー(LISTENER)を追加します。

```
srvctl add listener -endpoints TCP:1522
  -oraclehome /u01/app/oracle/product/database_release_number/dbhome_1
```

親トピック: srvctl add listener

4.5.1.4 srvctl add ons

Oracle Restart構成にOracle Notification Services (ONS)を追加します。

Oracle Data Guardのフェイルオーバーの後で高速アプリケーション通知(FAN)イベントの送信を有効にするには、ONSをOracle Restart構成に追加する必要があります。

ONSは、Oracle Restart構成に追加した時点では無効になっています。有効にするには、`srvctl enable ons`コマンドを使用します。

- 構文およびオプション

関連項目:

「srvctl enable ons」

親トピック: add

4.5.1.4.1 構文およびオプション

`srvctl add ons`コマンドは次の構文で使用します。

```
srvctl add ons [-emport em_port] [-onslocalport ons_local_port]
  [-onsremoteport ons_remote_port] [-remoteservers host[:port],[host[:port]...]]
  [-verbose]
```

表4-11 srvctl add onsのオプション

オプション	説明
`-emport em_port`	Oracle Enterprise Manager Cloud Control (Cloud Control)の ONS リスニング・ポート。デフォルトは 2016 です。
`-onslocalport ons_local_port`	ローカル・クライアント接続の ONS リスニング・ポート。デフォルトは 6100 です。
`-onsremoteport ons_remote_port`	リモート・ホストからの接続用の ONS リスニング・ポート。デフォルトは 6200 です。
`-remoteservers host[:port],[host[:port],...`	ONS ネットワークの一部であるリモート・ホストの `host:port` のペアのリスト。 ノート: リモート・ホストの `port` が指定されていない場合、`ons_remote_port` が使用されます。
`-verbose`	冗長出力

親トピック: srvctl add ons

4.5.1.5 srvctl add service

Oracle Restart構成にデータベース・サービスを追加します。

データベース・サービスがない場合、データベース・サービスを作成します。`DBMS_SERVICE` PL/SQLパッケージを使用するよりも、この方法でサービスを作成することをお薦めします。

- 構文およびオプション
- 例

親トピック: add

4.5.1.5.1 構文およびオプション

`srvctl add service`コマンドは次の構文で使用します。

```
srvctl add service -db db_unique_name -service service_name
  [-role [PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]]
  [-policy {AUTOMATIC | MANUAL}]
  [-failovertype {NONE | SESSION | SELECT | TRANSACTION}]
  [-failovermethod {NONE | BASIC}] [-failoverdelay integer]
  [-failoverretry integer] [-clbgoal {SHORT | LONG}]
  [-rlbgoal {SERVICE_TIME | THROUGHPUT | NONE}] [-notification {TRUE | FALSE}]
  [-edition edition_name] [-pdb pluggable_database]
  [-sql_translation_profile sql_translation_profile]
  [-commit_outcome {TRUE | FALSE}] [-retention retention]
  [-replay_init_time replay_init_time] [-drain_timeout timeout]
  [-stopoption stop_option] [-session_state {STATIC | DYNAMIC}]
  [-global {TRUE | FALSE}] [-maxlag max_lag_time] [-force] [-verbose]
```

表4-12 srvctl add serviceのオプション

オプション	説明

オプション	説明
-db db_unique_name	データベースの一意の名前 名前は DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-service service_name	データベース・サービス名
-role [PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]	サービス・ロールのリスト。 このオプションは、Oracle Data Guard 環境でのみ適用できます。このオプションを指定すると、データベースのオープン時に、サービス・ロールの 1 つが現在のデータベース・ロールに一致する場合にのみサービスが起動されます。 関連項目: データベース・ロールの詳細は、『Oracle Data Guard 概要および管理』を参照してください。
-policy {AUTOMATIC | MANUAL}	サービスの管理ポリシー。 AUTOMATIC(デフォルト)の場合、サービスは計画された再起動(SRVCTL を使用)または障害の後、データベースの再起動時に自動的に起動されます。ただし、自動再起動はサービス・ロールの影響も受けます(-role オプション)。 MANUAL の場合、データベースの計画された再起動(SRVCTL を使用)時にサービスは自動的には再起動されません。MANUAL を設定しても、サービスは実行中に Oracle Restart で監視され、障害が発生すると再起動されます。
-failovertype {NONE |SESSION | SELECT | TRANSACTION}	OCI および Java のアプリケーション・コンティニュイティを有効にするには、TRANSACTION を使用します。 フェイルオーバーのタイプが TRANSACTION の場合、リカバリ可能なエラーを受信したとき、OCI および Java は処理中のトランザクションのリカバリを試みます。フェイルオーバーのタイプが TRANSACTION の場合は、-commit_outcome オプションを TRUE に設定する必要があります。 OCI で透過的アプリケーション・フェイルオーバー(TAF)を有効にするには、SELECT または SESSION を使用します。
-failovermethod {NONE | BASIC}	下位互換性のみのための TAF フェイルオーバー方法。 フェイルオーバー・タイプ(-failovertype)に NONE 以外の値を設定する場合は、このオプ

オプション	説明
	ションに BASIC を使用します。
-failoverdelay integer	アプリケーション・コンティニュイティおよび TAF での、フェイルオーバーにおける各インシデントの再接続試行間の時間遅延(秒)。
-failoverretry integer	アプリケーション・コンティニュイティおよび TAF での、インシデント後の接続試行回数。
-clbgoal {SHORT | LONG}	接続ロード・バランシングの目標。 ランタイム・ロード・バランシングの場合は、SHORT を使用します。 バッチ・ジョブなどの長時間実行される接続の場合は、LONG を使用します。
-rlbgoal {SERVICE_TIME | THROUGHPUT | NONE}	ランタイム・ロード・バランシングの目標。 レスポンス時間によって接続を均衡化するには、SERVICE_TIME を使用します。 スループットによって接続を均衡化するには、THROUGHPUT を使用します。
-notification {TRUE | FALSE}	OCI 接続の場合は、高速アプリケーション通知(FAN)を有効にします。
-edition edition_name	サービスの初期セッション・エディション サービスにエディションを指定すると、そのサービスを指定するそれ以降のすべて接続で、初期セッション・エディションとしてこのエディションが使用されます。ただし、セッション接続で異なるエディションを指定した場合は、そのセッション接続で指定したエディションが初期セッション・エディションとして使用されます。 SRVCTL は、指定されたエディション名を検証しません。接続中、接続ユーザーは指定されたエディションの USE 権限を持っている必要があります。そのエディションが存在しないか、接続ユーザーが指定されたエディションの USE 権限を持たない場合は、エラーが発生します。
-pdb pluggable_database	マルチテナント・コンテナ・データベース(CDB)でサービスに関連付けるためのプラガブル・データベース(PDB)の名前 このオプションを空の文字列に設定した場合、サービスはルートに関連付けられます。
-sql_translation_profile sql_translation_profile	Oracle 以外のデータベースから Oracle Database にアプリケーションを移行した後に追加するサービスの SQL トランザクション・プロファイル。 このパラメータは、DBMS_SERVICE サービス属性の SQL トランザクション・プロファイル・パラ

オプション	説明
	メータに対応します。 ノート: ● SQL トランザクション・フレームワークを使用する前に、すべてのサーバー側のアプリケーション・オブジェクトとデータを Oracle Database に移行する必要があります。 ● SQL トランザクション・プロファイルを表示するには、`srvctl config service` コマンドを使用します。 関連項目: SQL 翻訳プロファイルの使用の詳細は、Oracle Database SQL 翻訳および移行ガイドを参照してください。
`-commit_outcome {TRUE | FALSE}`	トランザクション・ガードでは、TRUE にすると、リカバリ可能な停止によってトランザクションのセッションが失敗した後に、トランザクションのコミット結果にアクセスできます。 FALSE(デフォルト)の場合、トランザクションのコミット結果は保持されません。 このオプションを TRUE に設定すると、トランザクションのコミット結果は永続的になり、アプリケーションでは停止後のトランザクションのコミット・ステータスを確認できます。ユーザー定義サービスの場合、`commit_outcome` を TRUE に設定できます。 `commit_outcome` 設定は、Oracle Active Data Guard および読取り専用データベースには影響を与えません。 関連項目: 詳細は、『Oracle Database 開発ガイド』を参照してください。
`-retention retention`	`commit_outcome` が TRUE に設定されている場合、コミット結果が保持される時間(秒)をこのオプションで決定します。デフォルト値は 24 時間(86400)です。 `commit_outcome` が FALSE に設定されている場合、このオプションは設定できません。
`-replay_init_time replay_init_time`	アプリケーション・コンティニュイティについて、このオプションで、要求の最初の操作を実行する時間と、再接続の成功後にリプレイを開始する準備が完了する時間の差異(秒)を指定します。アプリケーション・コンティニュイティは指定した時間が経過するまでリプレイされません。このオプションは、長い時間が経過した後にシステムがリカバリされたときに、トランザクションが意図せず実行されることを回避するためのものです。デフォルトは 5 分(300)です。最大値は 24 時間(86400)です。 `failovertype` が `TRANSACTION` に設定されていない場合、このオプションは使用されません。
`-drain_timeout`	このオプションは、リソース・ドレインの完了までの許容時間を秒数で指定します。指定できる値

オプション	説明
timeout	は、NULL、0 または任意の正の整数です。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。ドレインの動作は、-stopoption オプションの設定によって異なります。 デフォルト値は NULL で、このオプションが設定されていないことを意味します。このオプションを設定せず、-drain_timeout がサービスに設定されている場合は、その値が使用されます。 0 を設定した場合、ドレインは行われません。
-stopoption stop_option	このオプションは、サービスを停止するモードを指定します。次の値を使用できます。 ● IMMEDIATE は、サービスが停止する前にセッションを排出できるように指定します。 ● TRANSACTIONAL は、–drain_timeout オプションに指定されている時間だけセッションがドレインできるように指定します。この時間制限に達するとサービスが停止され、残っているセッションは終了します。 ● デフォルトは NONE です。
-session_state {STATIC | DYNAMIC}	アプリケーション・コンティニュイティ向けに、このパラメータはトランザクション型ではないセッションの状態がアプリケーションによって変更されるかどうかを指定します。ほとんどのアプリケーションには DYNAMIC の設定をお薦めします。 ノート: このパラメータが考慮されるのは、アプリケーション・コンティニュイティ用に -failovertype が TRANSACTION に設定されている場合のみです。リクエスト時に非トランザクションが変更される方法を示します。セッション状態の例としては、NLS 設定、オプティマイザのプリファレンス、イベントの設定、PL/SQL グローバル変数、一時表、アドバンスト・キュー、LOB および結果キャッシュがあります。リクエストの開始後に非トランザクション値が変更された場合、デフォルトの DYNAMIC を使用します。ほとんどのアプリケーションで DYNAMIC モードを使用する必要があります。不明な場合は、DYNAMIC モードを使用してください。
-global {TRUE | FALSE}	TRUE の場合、サービスはグローバル・データ・サービス(GDS)のサービスとなり、Global Services Manager (GSM)によって管理されます。 FALSE(デフォルト)の場合、サービスは GDS サービスになりません。 サービスを追加した後にサービスのグローバル属性を変更することはできません。 詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してくだ

オプション	説明
	さい。
-maxlag maximum_lag_time	最大レプリケーション遅延時間(秒数)。負でない整数である必要があります。デフォルト値はANY です。
-force	ネットワークにリスナーが構成されていない場合にも、追加操作を強制します。
-verbose	冗長出力

親トピック: srvctl add service

4.5.1.5.2 例

この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのsalesサービスを追加します。dbcrmがPRIMARYモードである場合にのみ、サービスが起動されます。

```
srvctl add service -db dbcrm -service sales -role PRIMARY
```

関連項目:

- このコマンドのオプションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_SERVICEパッケージに関する項を参照してください。
- 「Oracle RestartとOracle Data Guardとの統合」
- 『Oracle Data Guard概要および管理』
- プラガブル・データベース(PDB)のサービスの作成方法、変更方法または削除方法の詳細は、『Oracle Multitenant管理者ガイド』を参照してください

親トピック: srvctl add service

4.5.2 config

srvctl configコマンドでは、指定したコンポーネントまたはコンポーネント・セットのOracle Restart構成を表示します。

- srvctl config asm
  Oracle ASMインスタンスのOracle Restart構成情報を表示します。
- srvctl config database
  指定したデータベースのOracle Restart構成情報を表示します。または、Oracle Restartで管理されているすべてのデータベースを示します。
- srvctl config listener
  Oracle Restartで管理されているすべてのリスナーまたは指定したリスナーのOracle Restart構成情報を表示します。
- srvctl config ons
  Oracle Notification Services (ONS)の現在の構成情報を表示します。
- srvctl config service

指定したデータベースについて、指定したデータベース・サービスまたはOracle Restartで管理されているすべてのデータベース・サービスのOracle Restart構成情報を表示します。

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.2.1 srvctl config asm

Oracle ASMインスタンスのOracle Restart構成情報を表示します。

- 構文およびオプション
- 例

親トピック: config

4.5.2.1.1 構文およびオプション

`srvctl config asm`コマンドは次の構文で使用します。

```
srvctl config asm [-all]
```

表4-13 srvctl config asmのオプション

オプション	説明
`-all`	有効/無効ステータスも表示します。

親トピック: srvctl config asm

4.5.2.1.2 例

次に、このコマンドの例を示します。

```
srvctl config asm -all
asm home: /u01/app/oracle/product/database_release_number/grid
ASM is enabled.
```

親トピック: srvctl config asm

4.5.2.2 srvctl config database

指定したデータベースのOracle Restart構成情報を表示します。または、Oracle Restartで管理されているすべてのデータベースを示します。

- 構文およびオプション
- 例

親トピック: config

4.5.2.2.1 構文およびオプション

`srvctl config database`コマンドは次の構文で使用します。

```
srvctl config database [-db db_unique_name [-all]] [-verbose]
```

表4-14 srvctl config databaseのオプション

オプション	説明

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-all	詳細な構成情報が表示されます。
-verbose	冗長出力

親トピック: srvctl config database

4.5.2.2.2 例

このコマンドの次の例では、Oracle Restartで管理されているすべてのデータベースを表示します。

```
srvctl config database
dbcrm
orcl
```

このコマンドの次の例では、DB_UNIQUE_IDがorclであるデータベースの構成および有効/無効ステータスを表示します。

```
srvctl config database -db orcl -all
Database unique name: orcl
Database name: orcl
Oracle home: /u01/app/oracle/product/database_release_number/dbhome_1
Oracle user: oracle
Spfile: +DATA/orcl/spfileorcl.ora
Domain: us.example.com
Start options: open
Stop options: immediate
Database role:
Management policy: automatic
Disk Groups: DATA
Services: mfg,sales
Database is enabled
```

親トピック: srvctl config database

4.5.2.3 srvctl config listener

Oracle Restartで管理されているすべてのリスナーまたは指定したリスナーのOracle Restart構成情報を表示します。

- 構文およびオプション
- 例

親トピック: config

4.5.2.3.1 構文およびオプション

srvctl config listenerコマンドは次の構文で使用します。

```
srvctl config listener [-listener listener_name]
```

表4-15 srvctl config listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、Oracle Restart で管理されているすべてのリスナーの構成情報が表示されます。

親トピック: srvctl config listener

4.5.2.3.2 例

この例では、デフォルト・リスナーの構成情報と有効/無効ステータスを表示します。

```
srvctl config listener
Name: LISTENER
Home: /u01/app/oracle/product/database_release_number/dbhome_1
End points: TCP:1521
Listener is enabled.
```

親トピック: srvctl config listener

4.5.2.4 srvctl config ons

Oracle Notification Services (ONS)の現在の構成情報を表示します。

- 構文およびオプション

親トピック: config

4.5.2.4.1 構文およびオプション

`srvctl config ons`コマンドは次の構文で使用します。

```
srvctl config ons
```

親トピック: srvctl config ons

4.5.2.5 srvctl config service

指定したデータベースについて、指定したデータベース・サービスまたはOracle Restartで管理されているすべてのデータベース・サービスのOracle Restart構成情報を表示します。

- 構文およびオプション
- 例

親トピック: config

4.5.2.5.1 構文およびオプション

`srvctl config service`コマンドは次の構文で使用します。

```
srvctl config service -db db_unique_name [-service service_name] [-verbose]
```

表4-16 srvctl config serviceのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期

オプション	説明
	化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-service service_name	データベース・サービス名省略すると、Oracle Restart で管理されているすべてのデータベース・サービスの構成情報が表示されます。
-verbose	冗長出力

親トピック: srvctl config service

4.5.2.5.2 例

次に、このコマンドの例を示します。

```
srvctl config service -db dbcrm -service sales
Service name: sales
Service is enabled
Cardinality: SINGLETON
Disconnect: true
Service role: PRIMARY
Management policy: automatic
DTP transaction: false
AQ HA notifications: false
Failover type: NONE
Failover method: NONE
TAF failover retries: 0
TAF failover delay: 0
Connection Load Balancing Goal: NONE
Runtime Load Balancing Goal: NONE
TAF policy specification: NONE
Edition: e2
```

親トピック: srvctl config service

4.5.3 disable

コンポーネントを無効にし、そのコンポーネントのOracle Restartでの管理を一時停止します。

srvctl disableコマンドは、コンポーネントを修復するかメンテナンス用に停止する必要があり、自動再起動しない場合に使用します。コンポーネントを無効にすると、次のようになります。

- 自動的に再起動しなくなります。
- 依存性によって自動的に起動しなくなります。
- SRVCTLで起動できません。

srvctl disable操作を実行するには、適切なユーザー・アカウントを使用してデータベース・ホスト・コンピュータにログインする必要があります。詳細は、「SRVCTLの実行準備」を参照してください。

- srvctl disable asm
  Oracle ASMインスタンスを無効化します。
- srvctl disable database
  指定したデータベースを無効にします。

- srvctl disable diskgroup
  Oracle ASMディスク・グループを無効化します。
- srvctl disable listener
  指定したリスナーまたはすべてのリスナーを無効にします。
- srvctl disable ons
  Oracle Notification Services (ONS)を無効にします。
- srvctl disable service
  1つ以上のデータベース・サービスを無効にします。

関連項目:

enableコマンド

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.3.1 srvctl disable asm

Oracle ASMインスタンスを無効にします。

- 構文およびオプション

親トピック: disable

4.5.3.1.1 構文およびオプション

`srvctl disable asm`コマンドは次の構文で使用します。

```
srvctl disable asm
```

親トピック: srvctl disable asm

4.5.3.2 srvctl disable database

指定したデータベースを無効にします。

- 構文およびオプション
- 例

親トピック: disable

4.5.3.2.1 構文およびオプション

`srvctl disable database`コマンドは次の構文で使用します。

```
srvctl disable database -db db_unique_name
```

表4-17 srvctl disable databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では

オプション	説明
	DB_NAME の設定を使用します。

親トピック: srvctl disable database

4.5.3.2.2 例

次に、このコマンドの例を示します。

```
srvctl disable database -db dbcrm
```

親トピック: srvctl disable database

4.5.3.3 srvctl disable diskgroup

Oracle ASMディスク・グループを無効にします。

- 構文およびオプション
- 例

親トピック: disable

4.5.3.3.1 構文およびオプション

`srvctl disable diskgroup`コマンドは次の構文で使用します。

```
srvctl disable diskgroup -diskgroup diskgroup_name
```

表4-18 srvctl disable diskgroupのオプション

オプション	説明
`-diskgroup diskgroup_name`	ディスク・グループ名

親トピック: srvctl disable diskgroup

4.5.3.3.2 例

次に、このコマンドの例を示します。

```
srvctl disable diskgroup -diskgroup DATA
```

親トピック: srvctl disable diskgroup

4.5.3.4 srvctl disable listener

指定したリスナーまたはすべてのリスナーを無効にします。

- 構文およびオプション
- 例

親トピック: disable

4.5.3.4.1 構文およびオプション

`srvctl disable listener`コマンドは次の構文で使用します。

```
srvctl disable listener [-listener listener_name]
```

表4-19 srvctl disable listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、すべてのリスナーが無効になります。

親トピック: srvctl disable listener

4.5.3.4.2 例

次に、このコマンドの例を示します。

```
srvctl disable listener -listener crmlistener
```

親トピック: srvctl disable listener

4.5.3.5 srvctl disable ons

Oracle Notification Services (ONS)を無効にします。

- 構文およびオプション

親トピック: disable

4.5.3.5.1 構文およびオプション

`srvctl disable ons`コマンドは次の構文で使用します。

```
srvctl disable ons [-verbose]
```

表4-20 srvctl disable onsのオプション

オプション	説明
`-verbose`	冗長出力

親トピック: srvctl disable ons

4.5.3.6 srvctl disable service

1つ以上のデータベース・サービスを無効にします。

- 構文およびオプション
- 例

親トピック: disable

4.5.3.6.1 構文およびオプション

`srvctl disable service`コマンドは次の構文で使用します。

```
srvctl disable service -db db_unique_name -service service_name_list
  [-global_override
```

表4-21 srvctl disable serviceのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-service service_name_list	データベース・サービス名のカンマ区切りのリスト。
-global_override	サービスがグローバル・データ・サービス(GDS)のサービスである場合、サービスを無効にするときには、このオプションを指定する必要があります。 GDS サービスを無効にしようとしたときに、-global_override が含まれていない場合は、エラーが返されます。 サービスが GDS サービスでない場合、このオプションは無視されます。 詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してください。

親トピック: srvctl disable service

4.5.3.6.2 例

次の例では、データベース・サービスsalesとmfgを無効にします。

```
srvctl disable service -db dbcrm -service sales,mfg
```

親トピック: srvctl disable service

4.5.4 downgrade

手動でデータベースをダウングレードした後に、srvctl downgradeコマンドでデータベース構成をダウングレードします。

- srvctl downgrade database
  srvctl downgrade databaseコマンドは、データベースとそのサービスの構成を現行バージョンから指定した下位バージョンにダウングレードします。

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.4.1 srvctl downgrade database

srvctl downgrade databaseコマンドは、データベースとそのサービスの構成を現行バージョンから指定した下位バージョンにダウングレードします。

- 構文およびオプション

親トピック: downgrade

4.5.4.1.1 構文およびオプション

srvctl downgrade databaseコマンドは、次の構文で使用します。

```
srvctl downgrade database -db db_unique_name -oraclehome oracle_home
       -targetversion to_version
```

表4-22 srvctl downgrade databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
`-oraclehome oracle_home`	データベースの Oracle ホームのフルパス。
`-targetversion to_version`	ダウングレード先のバージョン

親トピック: srvctl downgrade database

4.5.5 enable

`srvctl enable`コマンドは、指定した無効なコンポーネントを再有効化します。

コンポーネントを有効にした場合:

- Oracle Restartで自動再起動します。
- 依存性を介して自動起動します。
- SRVCTLを使用して手動で起動します。

コンポーネントがすでに有効になっている場合、コマンドは無視されます。

Oracle Restart構成にコンポーネントを追加すると、コンポーネントはデフォルトで有効になります。

`srvctl enable`操作を実行するには、適切なユーザー・アカウントを使用してデータベース・ホスト・コンピュータにログインする必要があります。詳細は、「SRVCTLの実行準備」を参照してください。

- srvctl enable asm
  Oracle ASMインスタンスを有効化します。
- srvctl enable database
  指定したデータベースを有効化します。
- srvctl enable diskgroup
  Oracle ASMディスク・グループを有効化します。
- srvctl enable listener
  指定したリスナーまたはすべてのリスナーを有効化します。
- srvctl enable ons
  Oracle Notification Services (ONS)を有効化します。
- srvctl enable service
  指定したデータベースの1つ以上のデータベース・サービスを有効化します。

関連項目:

disableコマンド

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.5.1 srvctl enable asm

Oracle ASMインスタンスを有効化します。

- 構文およびオプション

親トピック: enable

4.5.5.1.1 構文およびオプション

`srvctl enable asm`コマンドは次の構文で使用します。

```
srvctl enable asm
```

親トピック: srvctl enable asm

4.5.5.2 srvctl enable database

指定したデータベースを有効にします。

- 構文およびオプション
- 例

親トピック: enable

4.5.5.2.1 構文およびオプション

`srvctl enable database`コマンドは次の構文で使用します。

```
srvctl enable database -db db_unique_name
```

表4-23 srvctl enable databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。

親トピック: srvctl enable database

4.5.5.2.2 例

次に、このコマンドの例を示します。

```
srvctl enable database -db dbcrm
```

親トピック: srvctl enable database

4.5.5.3 srvctl enable diskgroup

Oracle ASMディスク・グループを有効にします。

- 構文およびオプション
- 例

親トピック: enable

4.5.5.3.1 構文およびオプション

`srvctl enable diskgroup`コマンドは次の構文で使用します。

```
srvctl enable diskgroup -diskgroup diskgroup_name
```

表4-24 srvctl enable diskgroupのオプション

オプション	説明
`-diskgroup diskgroup_name`	ディスク・グループ名

親トピック: srvctl enable diskgroup

4.5.5.3.2 例

次に、このコマンドの例を示します。

```
srvctl enable diskgroup -diskgroup DATA
```

親トピック: srvctl enable diskgroup

4.5.5.4 srvctl enable listener

指定したリスナーまたはすべてのリスナーを有効にします。

- 構文およびオプション
- 例

親トピック: enable

4.5.5.4.1 構文およびオプション

`srvctl enable listener`コマンドは次の構文で使用します。

```
srvctl enable listener [-listener listener_name]
```

表4-25 srvctl enable listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、すべてのリスナーが有効になります。

親トピック: srvctl enable listener

4.5.5.4.2 例

次に、このコマンドの例を示します。

```
srvctl enable listener -listener crmlistener
```

親トピック: srvctl enable listener

4.5.5.5 srvctl enable ons

Oracle Notification Services (ONS)を有効にします。

- 構文およびオプション

親トピック: enable

4.5.5.5.1 構文およびオプション

`srvctl enable ons`コマンドは次の構文で使用します。

```
srvctl enable ons [-verbose]
```

表4-26 srvctl enable onsのオプション

オプション	説明
`-verbose`	冗長出力

親トピック: srvctl enable ons

4.5.5.6 srvctl enable service

指定したデータベースの1つ以上のデータベース・サービスを有効にします。

- 構文およびオプション
- 例

親トピック: enable

4.5.5.6.1 構文およびオプション

`srvctl enable service`コマンドは次の構文で使用します。

```
srvctl enable service -db db_unique_name -service service_name_list
  [-global_override]
```

表4-27 srvctl enable serviceのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
`-service service_name_list`	データベース・サービス名のカンマ区切りのリスト。
`-global_override`	サービスがグローバル・データ・サービス(GDS)のサービスである場合、サービスを有効にするときには、このオプションを指定する必要があります。

オプション	説明
	GDS サービスを有効にしようとしたときに、`-global_override` が含まれていない場合は、エラーが返されます。 サービスが GDS サービスでない場合、このオプションは無視されます。 詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してください。

親トピック: srvctl enable service

4.5.5.6.2 例

次の例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのデータベース・サービスsalesとmfgを有効にします。

```
srvctl enable service -db dbcrm -service "sales,mfg"
```

親トピック: srvctl enable service

4.5.6 getenv

データベース、リスナーまたはOracle ASMインスタンスのOracle Restart構成から環境変数とその値を取得して表示します。

- srvctl getenv asm
  Oracle ASMインスタンスに対して構成された環境変数を表示します。
- srvctl getenv database
  指定したデータベースに対して構成された環境変数を表示します。
- srvctl getenv listener
  指定したリスナーに対して構成された環境変数を表示します。

関連項目:

- setenvコマンド
- unsetenvコマンド
- 「Oracle Restart構成の環境変数の管理」

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.6.1 srvctl getenv asm

Oracle ASMインスタンスに対して構成された環境変数を表示します。

- 構文およびオプション
- 例

親トピック: getenv

4.5.6.1.1 構文およびオプション

`srvctl getenv asm`コマンドは次の構文で使用します。

```
srvctl getenv asm [-envs name_list]
```

表4-28 srvctl getenv asmのオプション

オプション	説明
-envs name_list	表示する環境変数名のカンマ区切りのリスト。省略すると、SRVCTL によって、Oracle ASM に対して構成されたすべての環境変数が表示されます。

親トピック: srvctl getenv asm

4.5.6.1.2 例

次の例では、Oracle ASMインスタンスに対して構成されたすべての環境変数を表示します。

```
srvctl getenv asm
```

親トピック: srvctl getenv asm

4.5.6.2 srvctl getenv database

指定したデータベースに対して構成された環境変数を表示します。

- 構文およびオプション
- 例

親トピック: getenv

4.5.6.2.1 構文およびオプション

srvctl getenv databaseコマンドは次の構文で使用します。

```
srvctl getenv database -db db_unique_name [-envs name_list]
```

表4-29 srvctl getenv databaseのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-envs name_list	表示する環境変数名のカンマ区切りのリスト。省略すると、SRVCTL によって、構成されたすべての環境変数が表示されます。

親トピック: srvctl getenv database

4.5.6.2.2 例

次の例では、DB_UNIQUE_NAMEがdbcrmであるデータベースに対して構成されたすべての環境変数を表示します。

```
srvctl getenv database -db dbcrm
```

親トピック: srvctl getenv database

4.5.6.3 srvctl getenv listener

指定したリスナーに対して構成された環境変数を表示します。

- 構文およびオプション
- 例

親トピック: getenv

4.5.6.3.1 構文およびオプション

`srvctl getenv listener`コマンドは次の構文で使用します。

```
srvctl getenv listener [-listener listener_name] [-envs name_list]
```

表4-30 srvctl getenv listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、SRVCTL によって、すべてのリスナーの環境変数が表示されます。
`-envs name_list`	表示する環境変数名のカンマ区切りのリスト。省略すると、SRVCTL によって、構成されたすべての環境変数が表示されます。

親トピック: srvctl getenv listener

4.5.6.3.2 例

次の例では、`crmlistener`というリスナーに対して構成されたすべての環境変数を表示します。

```
srvctl getenv listener -listener crmlistener
```

親トピック: srvctl getenv listener

4.5.7 modify

コンポーネントのOracle Restart構成を変更します。変更内容は、コンポーネントが次に再起動されたときに反映されます。

`srvctl modify`操作を実行するには、適切なユーザー・アカウントを使用してデータベース・ホスト・コンピュータにログインする必要があります。詳細は、「SRVCTLの実行準備」を参照してください。

- srvctl modify asm
  Oracle ASMインスタンスのOracle Restart構成を変更します。
- srvctl modify database
  データベースのOracle Restart構成を変更します。
- srvctl modify listener
  指定したリスナーまたはすべてのリスナーのOracle Restart構成を変更します。
- srvctl modify ons
  Oracle Notification Services (ONS)を変更します。
- srvctl modify service
  データベース・サービスのOracle Restart構成を変更します。

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.7.1 srvctl modify asm

Oracle ASMインスタンスのOracle Restart構成を変更します。

- 構文およびオプション
- 例

親トピック: modify

4.5.7.1.1 構文およびオプション

`srvctl modify asm`コマンドは次の構文で使用します。

```
srvctl modify asm [-listener listener_name] [-spfile spfile]
  [-pwfile password_file_path] [-diskstring asm_diskstring]
```

表4-31 srvctl modify asmのオプション

オプション	説明
`-listener listener_name`	Oracle ASM が登録する必要があるリスナーの名前。このリスナーとの弱い依存性が確立されます。(Oracle Restart では、Oracle ASM を起動する前にこのリスナーの起動を確認します。)
`-spfile spfile`	データベースのサーバー・パラメータ・ファイルのフルパス。省略すると、デフォルトの SPFILE が使用されます。
`-pwfile password_file_path`	Oracle ASM パスワード・ファイルのフルパス。
`-diskstring asm_diskstring`	Oracle ASM ディスク・グループ検出文字列。Oracle ASM 検出文字列は、Oracle ASM インスタンスによって検出されるディスクのセットを制限する文字列のカンマ区切りリストです。検出文字列には、ワイルドカード文字を含めることができます。文字列のいずれかに一致するディスクのみが検出されます。

親トピック: srvctl modify asm

4.5.7.1.2 例

次に、このコマンドの例を示します。

```
srvctl modify asm -listener crmlistener
```

関連項目:

Oracle ASMのディスク・グループ検出文字列の詳細は、Oracle Automatic Storage Management管理者ガイドを参照してください。

親トピック: srvctl modify asm

4.5.7.2 srvctl modify database

データベースのOracle Restart構成を変更します。

- 構文およびオプション
- 例

親トピック: modify

4.5.7.2.1 構文およびオプション

srvctl modify databaseコマンドは次の構文で使用します。

```
srvctl modify database -db db_unique_name [-oraclehome oracle_home]
  [-user oracle_user] [-domain domain_name] [-dbname db_name]
  [-instance instance_name] [-instance instance_name] [-spfile spfile]
  [-pwfile password_file_path] [-startoption start_options]
  [-stopoption stop_options]
  [-role {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY}]
  [-policy {AUTOMATIC | MANUAL | NORESTART}]
  [{-diskgroup "diskgroup_list" | -nodiskgroup}] [-force]
```

表4-32 srvctl modify databaseのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-user oracle_user	Oracle ホーム・ディレクトリを所有する Oracle ユーザーの名前。
-diskgroup disk_group_list	データベースが依存するディスク・グループのカンマ区切りのリスト。Oracle Restart では、データベースを起動する前にこれらのディスク・グループがマウントされていることを確認します。このオプションは、データベースの追加時にデータベース・インスタンスと Oracle ASM インスタンスが起動されていなかった場合にのみ必要です。それ以外の場合、データベースとディスク・グループ間の依存性は自動的に記録されています。
-nodiskgroup	Oracle ASM ディスク・グループに対するデータベースの依存性を削除します。
-force	一部のリソースが停止されている場合にも、操作を強制します。
(その他のオプション)	表 4-9 を参照してください

親トピック: srvctl modify database

4.5.7.2.2 例

次の例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのロールをLOGICAL_STANDBYに変更します。

```
srvctl modify database -db dbcrm -role logical_standby
```

関連項目:

- 「Oracle RestartとOracle Data Guardとの統合」
- 『Oracle Data Guard概要および管理』

親トピック: srvctl modify database

4.5.7.3 srvctl modify listener

指定したリスナーまたはすべてのリスナーのOracle Restart構成を変更します。

- 構文およびオプション
- 例

親トピック: modify

4.5.7.3.1 構文およびオプション

`srvctl modify listener`コマンドは次の構文で使用します。

```
srvctl modify listener [-listener listener_name] [-endpoints endpoints]
  [-oraclehome oracle_home]
```

表4-33 srvctl modify listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、すべてのリスナー構成が変更されます。
`-endpoints endpoints`	カンマ区切りの TCP ポートまたはリスナー・エンドポイント。endpoints の構文は次のとおりです。 `"[TCP:]port[, ...] [/IPC:key] [/NMP:pipe_name] [/TCPS:s_port] [/SDP:port]"`
`-oraclehome oracle_home`	リスナーの新しい Oracle ホーム。

親トピック: srvctl modify listener

4.5.7.3.2 例

この例では、`crmlistener`というリスナーがリスニングしているTCPポートを変更します。

```
srvctl modify listener -listener crmlistener -endpoints TCP:1522
```

親トピック: srvctl modify listener

4.5.7.4 srvctl modify ons

Oracle Notification Services (ONS)を変更します。

- 構文およびオプション

親トピック: modify

4.5.7.4.1 構文およびオプション

`srvctl modify ons`コマンドは次の構文で使用します。

```
srvctl modify ons [-emport em_port] [-onslocalport ons_local_port]
  [-onsremoteport ons_remote_port] [-remoteservers host[:port],[host[:port]...]]
  [-verbose]
```

表4-34 srvctl modify onsのオプション

オプション	説明
`-emport em_port`	Cloud Control の ONS リスニング・ポート。デフォルトは 2016 です。
`-onslocalport ons_local_port`	ローカル・クライアント接続の ONS リスニング・ポート。
`-onsremoteport ons_remote_port`	リモート・ホストからの接続用の ONS リスニング・ポート。
`-remoteservers host[:port],[host[:port],...`	ONS ネットワークの一部であるリモート・ホストの `host:port` のペアのリスト。 ノート: リモート・ホストの `port` が指定されていない場合、`ons_remote_port` が使用されます。
`-verbose`	冗長出力

親トピック: srvctl modify ons

4.5.7.5 srvctl modify service

データベース・サービスのOracle Restart構成を変更します。

ノート:

構成変更は必要最小限にすること、およびオンライン・サービス変更の進行中は他のサービス操作を実行しないことをお薦めします。

- 構文およびオプション
- 例

親トピック: modify

4.5.7.5.1 構文およびオプション

`srvctl modify service`コマンドは次の構文で使用します。

```
srvctl modify service -db db_unique_name -service service_name
  [-role [PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]]
  [-policy {AUTOMATIC | MANUAL}]
  [-failovertype {NONE | SESSION | SELECT | TRANSACTION}]
  [-failovermethod {NONE | BASIC}] [-failoverdelay integer]
  [-failoverretry integer] [-clbgoal {SHORT | LONG}]
  [-rlbgoal {SERVICE_TIME | THROUGHPUT | NONE}] [-notification {TRUE | FALSE}]
  [-edition edition_name] [-pdb pluggable_database]
  [-sql_translation_profile sql_translation_profile]
  [-commit_outcome {TRUE | FALSE}] [-retention retention]
  [-replay_init_time replay_init_time] [-drain_timeout timeout]
```

```
  [-stopoption stop_option] [-session_state {STATIC | DYNAMIC}]
  [-global_override] [-verbose]
```

表4-35 srvctl modify serviceのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前 名前は DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-service service_name	サービス名
-role [PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]	サービス・ロールのリスト。 このオプションは、Oracle Data Guard 環境でのみ適用できます。このオプションを指定すると、データベースの起動時に、サービス・ロールの 1 つが現在のデータベース・ロールに一致する場合にのみサービスは起動されます。 関連項目: データベース・ロールの詳細は、『Oracle Data Guard 概要および管理』を参照してください。
-policy {AUTOMATIC | MANUAL}	サービスの管理ポリシー。 AUTOMATIC(デフォルト)の場合、サービスは計画された再起動(SRVCTL を使用)または障害の後、データベースの再起動時に自動的に起動されます。ただし、自動再起動はサービス・ロールの影響も受けます(-role オプション)。 MANUAL の場合、データベースの計画された再起動(SRVCTL を使用)時にサービスは自動的には再起動されません。MANUAL を設定しても、サービスは実行中に Oracle Restart で監視され、障害が発生すると再起動されます。
-failovertype {NONE |SESSION | SELECT | TRANSACTION}	OCI および Java のアプリケーション・コンティニュイティを有効にするには、TRANSACTION を使用します。 フェイルオーバーのタイプが TRANSACTION の場合、リカバリ可能なエラーを受信したとき、OCI および Java は処理中のトランザクションのリカバリを試みます。フェイルオーバーのタイプが TRANSACTION の場合は、commit_outcome オプションを TRUE に設定する必要があります。 OCI で透過的アプリケーション・フェイルオーバー(TAF)を有効にするには、SELECT または SESSION を使用します。

オプション	説明
-failovermethod {NONE | BASIC}	下位互換性のみのための TAF フェイルオーバー方法。 フェイルオーバー・タイプ(-failovertype)に NONE 以外の値を設定する場合は、このオプションに BASIC を使用します。
-failoverdelay integer	アプリケーション・コンティニュイティおよび TAF での、フェイルオーバーにおける各インシデントの再接続試行間の時間遅延(秒)。
-failoverretry integer	アプリケーション・コンティニュイティおよび TAF での、インシデント後の接続試行回数。
-clbgoal {SHORT | LONG}	接続ロード・バランシングの目標。 ランタイム・ロード・バランシングの場合は、SHORT を使用します。 バッチ・ジョブなどの長時間実行される接続の場合は、LONG を使用します。
-rlbgoal {SERVICE_TIME | THROUGHPUT | NONE}	ランタイム・ロード・バランシングの目標。 レスポンス時間によって接続を均衡化するには、SERVICE_TIME を使用します。 スループットによって接続を均衡化するには、THROUGHPUT を使用します。
-notification {TRUE | FALSE}	OCI 接続の場合は、高速アプリケーション通知(FAN)を有効にします。
-edition edition_name	サービスの初期セッション・エディション このオプションを指定しない場合、サービスのエディションは変更されません。 このオプションが指定されているものの、edition_name が空である場合、エディションは NULL に設定されます。NULL のエディションには効果がありません。 サービスにエディションを指定すると、そのサービスを指定するそれ以降のすべて接続で、初期セッション・エディションとしてこのエディションが使用されます。ただし、セッション接続で異なるエディションを指定した場合は、そのセッション接続で指定したエディションが初期セッション・エディションとして使用されます。 SRVCTL は、指定されたエディション名を検証しません。接続中、接続ユーザーは指定されたエディションの USE 権限を持っている必要があります。そのエディションが存在しないか、接続ユーザーが指定されたエディションの USE 権限を持たない場合は、エラーが発生します。
-pdb pluggable_database	CDB で、サービスと関連付ける PDB の名前。

オプション	説明
	このオプションを空の文字列に設定した場合、サービスはルートに関連付けられます。
- sql_translation_profile sql_translation_profile	Oracle 以外のデータベースから Oracle Database にアプリケーションを移行した後に追加するサービスの SQL トランザクション・プロファイル。 ノート: SQL トランザクション・フレームワークを使用する前に、すべてのサーバー側のアプリケーション・オブジェクトとデータを Oracle Database に移行する必要があります。 関連項目: SQL 翻訳プロファイルの使用の詳細は、Oracle Database SQL 翻訳および移行ガイドを参照してください。
-commit_outcome {TRUE | FALSE}	トランザクション・ガードでは、TRUE にすると、リカバリ可能な停止によってトランザクションのセッションが失敗した後に、トランザクションのコミット結果にアクセスできます。 FALSE(デフォルト)の場合、トランザクションのコミット結果は保持されません。 このオプションを TRUE に設定すると、トランザクションのコミット結果は永続的になり、アプリケーションでは停止後のトランザクションのコミット・ステータスを確認できます。ユーザー定義サービスの場合、commit_outcome を TRUE に設定できます。 commit_outcome 設定は、Oracle Active Data Guard および読取り専用データベースには影響を与えません。 関連項目: 詳細は、『Oracle Database 開発ガイド』を参照してください。
-retention retention	commit_outcome が TRUE に設定されている場合、コミット結果が保持される時間(秒)をこのオプションで決定します。デフォルト値は 24 時間(86400)です。 commit_outcome が FALSE に設定されている場合、このオプションは設定できません。
-replay_init_time replay_init_time	アプリケーション・コンティニュイティについて、このオプションで、要求の最初の操作を実行する時間と、再接続の成功後にリプレイを開始する準備が完了する時間の差異(秒)を指定します。アプリケーション・コンティニュイティは指定した時間が経過するまでリプレイされません。このオプションは、長い時間が経過した後にシステムがリカバリされたときに、トランザクションが意図せず実行されることを回避するためのものです。デフォルトは 5 分(300)です。最大値は 24 時間(86400)です。 failovertype が TRANSACTION に設定されていない場合、このオプションは使用されません。
-drain_timeout timeout	このオプションは、リソース・ドレインの完了までの許容時間を秒数で指定します。指定できる値

オプション	説明
	は、NULL、0 または任意の正の整数です。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。ドレインの動作は、-stopoption オプションの設定によって異なります。 デフォルト値は NULL で、このオプションが設定されていないことを意味します。このオプションを設定せず、-drain_timeout がサービスに設定されている場合は、その値が使用されます。 0 を設定した場合、ドレインは行われません。
-stopoption stop_option	このオプションは、サービスを停止するモードを指定します。次の値を使用できます。 ● IMMEDIATE は、サービスが停止する前にセッションを排出できるように指定します。 ● TRANSACTIONAL は、–drain_timeout オプションに指定されている時間だけセッションがドレインできるように指定します。この時間制限に達するとサービスが停止され、残っているセッションは終了します。 ● デフォルトは NONE です。
-session_state {STATIC | DYNAMIC}	アプリケーション・コンティニュイティ向けに、このパラメータはトランザクション型ではないセッションの状態がアプリケーションによって変更されるかどうかを指定します。ほとんどのアプリケーションには DYNAMIC の設定をお薦めします。 ノート: このパラメータが考慮されるのは、アプリケーション・コンティニュイティ用に-failovertype が TRANSACTION に設定されている場合のみです。リクエスト時に非トランザクションが変更される方法を示します。セッション状態の例としては、NLS 設定、オプティマイザのプリファレンス、イベントの設定、PL/SQL グローバル変数、一時表、アドバンスト・キュー、LOB および結果キャッシュがあります。リクエストの開始後に非トランザクション値が変更された場合、デフォルトの DYNAMIC を使用します。ほとんどのアプリケーションで DYNAMIC モードを使用する必要があります。不明な場合は、DYNAMIC モードを使用してください。
-global_override	サービスがグローバル・データ・サービス(GDS)のサービスである場合、次のいずれかのサービス属性を変更するときには、このオプションを指定する必要があります。 ● -role ● -policy ● -failovertype

オプション	説明
	• -failovermethod • -failoverdelay • -failoverretry • -edition • -clbgoal • -rlbgoal • -notification GDS サービスの、これらのいずれかのオプションを変更しようとしたときに、-global_override が含まれていない場合は、エラーが返されます。 サービスが GDS サービスでない場合、このオプションは無視されます。 詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してください。
-verbose	冗長出力

親トピック: srvctl modify service

4.5.7.5.2 例

次のコマンドでは、DB_UNIQUE_NAMEがdbcrmであるデータベースで、supportというデータベース・サービスのOracle Data Guardロールをstandbyに変更します。

```
srvctl modify service -db dbcrm -service support -role standby
```

関連項目:

PDBに関連付けられたサービスの管理の詳細は、『Oracle Multitenant管理者ガイド』を参照してください

親トピック: srvctl modify service

4.5.8 remove

Oracle Restart構成から指定したコンポーネントを削除します。コンポーネントはOracle Restartで管理されなくなります。コンポーネントの環境変数設定も削除されます。

Oracle Restart構成からコンポーネントを削除する前に、SRVCTLを使用してコンポーネントを停止する必要があります。コンポーネントは削除前に無効にすることをお薦めしますが、必須ではありません。

srvctl remove操作を実行するには、適切なユーザー・アカウントを使用してデータベース・ホスト・コンピュータにログインする必要があります。詳細は、「SRVCTLの実行準備」を参照してください。

- srvctl remove asm
  Oracle ASMインスタンスを削除します。
- srvctl remove database
  データベースを削除します。最初に確認のプロンプトを表示します。
- srvctl remove diskgroup
  Oracle ASMディスク・グループを削除します。
- srvctl remove listener
  指定したリスナーまたはすべてのリスナーを削除します。
- srvctl remove ons
  Oracle Notification Services (ONS)を削除します。
- srvctl remove service
  指定したデータベース・サービスを削除します。

関連項目:

- stopコマンド
- disableコマンド

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.8.1 srvctl remove asm

Oracle ASMインスタンスを削除します。

- 構文およびオプション
- 例

親トピック: remove

4.5.8.1.1 構文およびオプション

`srvctl remove asm`コマンドは次の構文で使用します。

```
srvctl remove asm [-force]
```

表4-36 srvctl remove asmのオプション

オプション	説明
`-force`	Oracle ASM を使用するディスク・グループおよびデータベースが存在する場合、または Oracle ASM インスタンスが実行中の場合でも、強制的に削除します。

親トピック: srvctl remove asm

4.5.8.1.2 例

次に、このコマンドの例を示します。

```
srvctl remove asm
```

親トピック: srvctl remove asm

4.5.8.2 srvctl remove database

データベースを削除します。最初に確認のプロンプトを表示します。

- 構文およびオプション
- 例

親トピック: remove

4.5.8.2.1 構文およびオプション

`srvctl remove database`コマンドは次の構文で使用します。

ノート:

SYS ユーザーのパスワードを使用して SYS ユーザーとしてデータベースに接続する場合は、このコマンドを実行した後に、パスワード・ファイルがデフォルトの場所にあることを確認します。

```
srvctl remove database -db db_unique_name [-force] [-noprompt] [-verbose]
```

表4-37 srvctl remove databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
`-force`	強制。実行中でもデータベースを削除します。
`-noprompt`	確認のプロンプトを表示せず、即時削除します。
`-verbose`	冗長出力。成功または失敗のメッセージが表示されます。

親トピック: srvctl remove database

4.5.8.2.2 例

次に、このコマンドの例を示します。

```
srvctl remove database -db dbcrm
```

親トピック: srvctl remove database

4.5.8.3 srvctl remove diskgroup

Oracle ASMディスク・グループを削除します。

- 構文およびオプション
- 例

親トピック: remove

4.5.8.3.1 構文およびオプション

srvctl remove diskgroupコマンドは次の構文で使用します。

```
srvctl remove diskgroup -diskgroup diskgroup_name [-force]
```

表4-38 srvctl remove diskgroupのオプション

オプション	説明
-diskgroup diskgroup_name	ディスク・グループ名
-force	強制。ファイルが開いていてもディスク・グループを削除します。

親トピック: srvctl remove diskgroup

4.5.8.3.2 例

この例では、DATAというディスク・グループを削除します。このディスク・グループでファイルが開いている場合、エラーが返されます。

```
srvctl remove diskgroup -diskgroup DATA
```

親トピック: srvctl remove diskgroup

4.5.8.4 srvctl remove listener

指定したリスナーまたはすべてのリスナーを削除します。

- 構文およびオプション
- 例

親トピック: remove

4.5.8.4.1 構文およびオプション

srvctl remove listenerコマンドは次の構文で使用します。

```
srvctl remove listener [-listener listener_name | -all] [-force]
```

表4-39 srvctl remove listenerのオプション

オプション	説明
-listener listener_name	削除するリスナーの名前。省略すると、デフォルトの LISTENER が使用されます。
-all	すべてのリスナーを削除します。
-force	強制。データベースで使用されていてもリスナーを削除します。

親トピック: srvctl remove listener

4.5.8.4.2 例

次のコマンドではリスナーlsnr01を削除します。

```
srvctl remove listener -listener lsnr01
```

親トピック: srvctl remove listener

4.5.8.5 srvctl remove ons

Oracle Notification Services (ONS)を削除します。

- 構文およびオプション

親トピック: remove

4.5.8.5.1 構文およびオプション

srvctl remove onsコマンドは次のように使用します。

```
srvctl remove ons [-force] [-verbose]
```

表4-40 srvctl remove onsのオプション

オプション	説明
-force	強制。有効な ONS も削除します。
-verbose	冗長出力

親トピック: srvctl remove ons

4.5.8.6 srvctl remove service

指定したデータベース・サービスを削除します。

- 構文およびオプション
- 例

親トピック: remove

4.5.8.6.1 構文およびオプション

srvctl remove serviceコマンドは次のように使用します。

```
srvctl remove service -db db_unique_name -service service_name [-global_override]
```

表4-41 srvctl remove serviceのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。

オプション	説明
`-service service_name`	サービス名
`-global_override`	サービスがグローバル・データ・サービス(GDS)のサービスである場合、サービスを削除するときには、このオプションを指定する必要があります。 GDS サービスを削除しようとしたときに、`-global_override` が含まれていない場合は、エラーが返されます。 サービスが GDS サービスでない場合、このオプションは無視されます。 詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してください。

親トピック: srvctl remove service

4.5.8.6.2 例

次に、このコマンドの例を示します。

```
srvctl remove service -db dbcrm -service sales
```

親トピック: srvctl remove service

4.5.9 setenv

`setenv`コマンドでは、データベース、リスナーまたはOracle ASMインスタンスのOracle Restart構成に環境変数の値を設定します。

`srvctl setenv`操作を実行するには、適切なユーザー・アカウントを使用してデータベース・ホスト・コンピュータにログインする必要があります。詳細は、「SRVCTLの実行準備」を参照してください。

- srvctl setenv asm
  Oracle ASMインスタンスのOracle Restart構成に環境変数の値を設定します。Oracle Restartでは、インスタンスを起動する前に、環境変数を構成に格納された値に設定します。
- srvctl setenv database
  データベース・インスタンスのOracle Restart構成に環境変数の値を設定します。Oracle Restartでは、インスタンスを起動する前に、環境変数を構成に格納された値に設定します。
- srvctl setenv listener
  リスナーのOracle Restart構成に環境変数の値を設定します。Oracle Restartでは、リスナーを起動する前に、環境変数を構成に格納された値に設定します。

関連項目:

- getenvコマンド
- unsetenvコマンド
- 「Oracle Restart構成の環境変数の管理」

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.9.1 srvctl setenv asm

Oracle ASMインスタンスのOracle Restart構成に環境変数の値を設定します。Oracle Restartでは、インスタンスを起動する前に、環境変数を構成に格納された値に設定します。

- 構文およびオプション
- 例

親トピック: setenv

4.5.9.1.1 構文およびオプション

`srvctl setenv asm`コマンドは次の構文で使用します。

```
srvctl setenv asm {-envs name=val[,name=val,...] | -env name=val}
```

表4-42 srvctl setenv databaseのオプション

オプション	説明
`-envs name=val[,name=val,...]`	環境変数の名前と値のペアのカンマ区切りのリスト。
`-env name=val`	単一環境変数をカンマや他の特殊文字を含む値に設定可能にする

親トピック: srvctl setenv asm

4.5.9.1.2 例

次の例では、AIXオペレーティング・システムの環境変数`AIXTHREAD_SCOPE`をOracle ASMインスタンスの構成に設定します。

```
srvctl setenv asm -envs AIXTHREAD_SCOPE=S
```

親トピック: srvctl setenv asm

4.5.9.2 srvctl setenv database

データベース・インスタンスのOracle Restart構成に環境変数の値を設定します。Oracle Restartでは、インスタンスを起動する前に、環境変数を構成に格納された値に設定します。

- 構文およびオプション
- 例

親トピック: setenv

4.5.9.2.1 構文およびオプション

`srvctl setenv database`コマンドは次の構文で使用します。

```
srvctl setenv database -db db_unique_name
  {-envs name=val[,name=val,...] | -env name=val}
```

表4-43 srvctl setenv databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
`-envs name=val[,name=val,...]`	環境変数の名前と値のペアのカンマ区切りのリスト。
`-env name=val`	単一環境変数をカンマや他の特殊文字を含む値に設定可能にする

親トピック: srvctl setenv database

4.5.9.2.2 例

次の例では、DB_UNIQUE_NAMEがdbcrmであるデータベースの構成にLANG環境変数を設定します。

```
srvctl setenv database -db dbcrm -envs LANG=en
```

親トピック: srvctl setenv database

4.5.9.3 srvctl setenv listener

リスナーのOracle Restart構成に環境変数の値を設定します。Oracle Restartでは、リスナーを起動する前に、環境変数を構成に格納された値に設定します。

- 構文およびオプション
- 例

親トピック: setenv

4.5.9.3.1 構文およびオプション

`srvctl setenv listener`コマンドは次の構文で使用します。

```
srvctl setenv listener [-listener listener_name]
  {-envs name=val[,name=val,...] | -env name=val}
```

表4-44 srvctl setenv listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、すべてのリスナー構成に指定した環境変数が設定されます。
`-envs name=val[,name=val,...]`	環境変数の名前と値のペアのカンマ区切りのリスト。
`-env name=val`	単一環境変数をカンマや他の特殊文字を含む値に設定可能にする

親トピック: srvctl setenv listener

4.5.9.3.2 例

次の例では、AIXオペレーティング・システムの環境変数AIXTHREAD_SCOPEをcrmlistenerというリスナーの構成に設定します。

```
srvctl setenv listener -listener crmlistener -envs AIXTHREAD_SCOPE=S
```

親トピック: srvctl setenv listener

4.5.10 start

指定したコンポーネントを起動します。

- srvctl start asm
  Oracle ASMインスタンスを起動します。
- srvctl start database
  指定したデータベース・インスタンスを起動します。
- srvctl start diskgroup
  Oracle ASMディスク・グループを起動(マウント)します。
- srvctl start home
  Oracle Restartで管理されているすべてのコンポーネントを、指定したOracleホーム内で起動します。OracleホームはOracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。
- srvctl start listener
  指定したリスナーまたはすべてのリスナーを起動します。
- srvctl start ons
  Oracle Notification Services (ONS)を起動します。
- srvctl start service
  指定したデータベース・サービスを起動します。

関連項目:

「Oracle Restartで管理されているコンポーネントの起動と停止」

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.10.1 srvctl start asm

Oracle ASMインスタンスを起動します。

このコマンドでは、SRVCTLは"/ as sysasm"で接続して操作を実行します。このような操作を実行するには、Oracle Gridインフラストラクチャ・ホームの実行可能ファイルの所有者がOSASMグループのメンバーであることが必要です。また、コマンドを実行するユーザーもOSASMグループに属する必要があります。

- 構文およびオプション
- 例

親トピック: start

4.5.10.1.1 構文およびオプション

srvctl start asmコマンドは次の構文で使用します。

```
srvctl start asm [-startoption start_options]
```

表4-45 srvctl start asmのオプション

オプション	説明
`-startoption start_options`	起動コマンドのオプションのカンマ区切りのリスト(OPEN、MOUNT、NOMOUNT または FORCE)。省略すると、デフォルトで通常の起動(OPEN)に設定されます。 関連項目: 起動オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。

親トピック: srvctl start asm

4.5.10.1.2 例

この例では、Oracle ASMインスタンスを起動します。その後、ASM_DISKGROUPS初期化パラメータで指定されたディスク・グループをマウントします。

```
srvctl start asm
```

この例では、ディスク・グループをマウントせずにOracle ASMインスタンスを起動します。

```
srvctl start asm -startoption nomount
```

親トピック: srvctl start asm

4.5.10.2 srvctl start database

指定したデータベース・インスタンスを起動します。

このコマンドでは、SRVCTLは"/ as sysdba"で接続して操作を実行します。このような操作を実行するには、データベースOracleホームのOracle実行可能ファイルの所有者がOSDBAグループ(たとえば、UNIXおよびLinuxのdbaグループ)のメンバーであることが必要です。また、コマンドを実行するユーザーもOSDBAグループに属する必要があります。

- 構文およびオプション
- 例

親トピック: start

4.5.10.2.1 構文およびオプション

`srvctl start database`コマンドは次の構文で使用します。

```
srvctl start database -db db_unique_name [-startoption start_options] [-verbose]
```

表4-46 srvctl start databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。

オプション	説明
`-startoption start_options`	起動コマンドのオプションのカンマ区切りのリスト(たとえば、OPEN、MOUNT、NOMOUNT、RESTRICT など)。 ノート: ● このコマンド・パラメータでは PFILE オプションまたは QUIET オプションをサポートしていませんが、他のすべてのデータベース起動オプションをサポートしています。 ● 起動オプションに複数の語を指定する場合(`read only`、`read write` など)、語をスペースで区切り、一重引用符(`''`)で囲みます。たとえば`'read only'`とします。 関連項目: 起動オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。
`-verbose`	冗長出力

親トピック: srvctl start database

4.5.10.2.2 例

次に、このコマンドの例を示します。

```
srvctl start database -db dbcrm -startoption nomount
```

親トピック: srvctl start database

4.5.10.3 srvctl start diskgroup

Oracle ASMディスク・グループを起動(マウント)します。

- 構文およびオプション
- 例

親トピック: start

4.5.10.3.1 構文およびオプション

`srvctl start diskgroup`コマンドは次の構文で使用します。

```
srvctl start diskgroup -diskgroup diskgroup_name
```

表4-47 srvctl start diskgroupのオプション

オプション	説明
`-diskgroup diskgroup_name`	ディスク・グループ名

親トピック: srvctl start diskgroup

4.5.10.3.2 例

次に、このコマンドの例を示します。

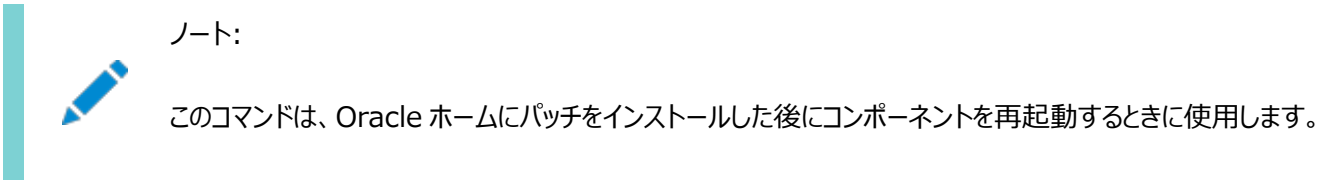
```
srvctl start diskgroup -diskgroup DATA
```

親トピック: srvctl start diskgroup

4.5.10.4 srvctl start home

Oracle Restartで管理されているすべてのコンポーネントを、指定したOracleホーム内で起動します。Oracleホームは Oracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。

このコマンドでは`srvctl stop home`によって停止されたコンポーネントを起動します。このコマンドでは指定した状態ファイルの情報を使用して、起動するコンポーネントを特定します。



ノート:

このコマンドは、Oracle ホームにパッチをインストールした後にコンポーネントを再起動するときに使用します。

- 構文およびオプション

親トピック: start

4.5.10.4.1 構文およびオプション

`srvctl start home`コマンドは次の構文で使用します。

```
srvctl start home -oraclehome oracle_home -statefile state_file
```

表4-48 srvctl start homeのオプション

オプション	説明
`-oraclehome oracle_home`	Oracle ホームの完全なパス。
`-statefile state_file`	状態ファイルの完全なパス。状態ファイルには Oracle ホームにあるコンポーネントの現在の状態情報が入ります。`srvctl stop home` コマンドまたは `srvctl status home` コマンドが実行されると、このファイルが作成されます。

親トピック: srvctl start home

4.5.10.5 srvctl start listener

指定したリスナーまたはすべてのリスナーを起動します。

- 構文およびオプション
- 例

親トピック: start

4.5.10.5.1 構文およびオプション

`srvctl start listener`コマンドは次の構文で使用します。

```
srvctl start listener [-listener listener_name]
```

表4-49 srvctl start listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、Oracle Restart で管理されているすべてのリスナーが起動されます。

親トピック: srvctl start listener

4.5.10.5.2 例

次に、このコマンドの例を示します。

```
srvctl start listener -listener listener
```

親トピック: srvctl start listener

4.5.10.6 srvctl start ons

Oracle Notification Services (ONS)を起動します。

- 構文およびオプション

親トピック: start

4.5.10.6.1 構文およびオプション

`srvctl start ons`コマンドは次の構文で使用します。

```
srvctl start ons [-verbose]
```

表4-50 srvctl start onsのオプション

オプション	説明
`-verbose`	冗長出力

親トピック: srvctl start ons

4.5.10.7 srvctl start service

指定したデータベース・サービスを起動します。

- 構文およびオプション
- 例

親トピック: start

4.5.10.7.1 構文およびオプション

`srvctl start service`コマンドは次の構文で使用します。

```
srvctl start service -db db_unique_name [-service service_name_list |
  -pdb pluggable_database] [-startoption start_options] [-global_override] [-verbose]
```

表4-51 srvctl start serviceのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
`-service service_name_list`	サービス名のカンマ区切りリスト。このサービス名リストはオプションであり、指定しないと、SRVCTL でデータベースのサービスがすべて起動されます。
`-pdb pluggable_database`	CDB で、サービスに関連付けられた PDB の名前 このオプションを空の文字列に設定した場合、サービスはルートに関連付けられます。
`-startoption start_options`	データベースを最初に起動する必要がある場合、データベースの起動オプション(OPEN、MOUNT、NOMOUNT など)。 関連項目: 起動オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。
`-global_override`	サービスがグローバル・データ・サービス(GDS)のサービスである場合、サービスを起動するときには、このオプションを指定する必要があります。 GDS サービスを起動しようとしたときに、`-global_override` が含まれていない場合は、エラーが返されます。 サービスが GDS サービスでない場合、このオプションは無視されます。 詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してください。
`-verbose`	冗長出力

親トピック: srvctl start service

4.5.10.7.2 例

次の例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのsalesデータベース・サービスを起動します。

```
srvctl start service -db dbcrm -service sales
```

親トピック: srvctl start service

4.5.11 status

指定したコンポーネントまたはコンポーネント・セットの実行ステータスを表示します。

- srvctl status asm
  Oracle ASMインスタンスの実行ステータスを表示します。
- srvctl status database
  指定したデータベースの実行ステータスを表示します。
- srvctl status diskgroup
  Oracle ASMディスク・グループの実行ステータスを表示します。
- srvctl status home
  Oracle Restartで管理されているすべてのコンポーネントの実行ステータスを、指定したOracleホーム内で表示します。OracleホームはOracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。
- srvctl status listener
  指定したリスナーまたはOracle Restartで管理されているすべてのリスナーの実行ステータスを表示します。
- srvctl status ons
  Oracle Notification Services (ONS)の実行ステータスを表示します。
- srvctl status service
  1つ以上のデータベース・サービスの実行ステータスを表示します。

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.11.1 srvctl status asm

Oracle ASMインスタンスの実行ステータスを表示します。

- 構文およびオプション
- 例

親トピック: status

4.5.11.1.1 構文およびオプション

`srvctl status asm`コマンドは次の構文で使用します。

```
srvctl status asm [-all] [-verbose]
```

表4-52 srvctl status asmのオプション

オプション	説明
`-all`	有効/無効ステータスも表示します。
`-verbose`	冗長出力

親トピック: srvctl status asm

4.5.11.1.2 例

次に、このコマンドの例を示します。

```
srvctl status asm
ASM is running on dbhost
```

親トピック: srvctl status asm

4.5.11.2 srvctl status database

指定したデータベースの実行ステータスを表示します。

- 構文およびオプション
- 例

親トピック: status

4.5.11.2.1 構文およびオプション

srvctl status databaseコマンドは次の構文で使用します。

```
srvctl status database -db db_unique_name [-force] [-verbose]
```

表4-53 srvctl status databaseのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-force	データベースが無効な場合、メッセージを表示します。
-verbose	冗長出力。実行中のデータベース・サービスを一覧表示します。

親トピック: srvctl status database

4.5.11.2.2 例

次に、このコマンドの例を示します。

```
srvctl status database -db dbcrm -verbose
Database dbcrm is running with online services mfg,sales
```

親トピック: srvctl status database

4.5.11.3 srvctl status diskgroup

Oracle ASMディスク・グループの実行ステータスを表示します。

- 構文およびオプション
- 例

親トピック: status

4.5.11.3.1 構文およびオプション

srvctl status diskgroupコマンドは次の構文で使用します。

```
srvctl status diskgroup -diskgroup diskgroup_name [-all] [-verbose]
```

表4-54 srvctl status diskgroupのオプション

オプション	説明
`-diskgroup diskgroup_name`	ディスク・グループ名
`-all`	有効/無効ステータスも表示します。
`-verbose`	冗長出力。実行中のデータベース・サービスを一覧表示します。

親トピック: srvctl status diskgroup

4.5.11.3.2 例

次に、このコマンドの例を示します。

```
srvctl status diskgroup -diskgroup DATA
Disk Group DATA is running on dbhost
```

親トピック: srvctl status diskgroup

4.5.11.4 srvctl status home

Oracle Restartで管理されているすべてのコンポーネントの実行ステータスを、指定したOracleホーム内で表示します。OracleホームはOracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。

このコマンドではコンポーネントの現在のステータスを指定した状態ファイルに書き込みます。

- 構文およびオプション

親トピック: status

4.5.11.4.1 構文およびオプション

`srvctl status home`コマンドは次の構文で使用します。

```
srvctl status home -oraclehome oracle_home -statefile state_file
```

表4-55 srvctl status homeのオプション

オプション	説明
`-oraclehome oracle_home`	Oracle ホームの完全なパス。
`-statefile state_file`	状態ファイルの完全なパス。

親トピック: srvctl status home

4.5.11.5 srvctl status listener

指定したリスナーまたはOracle Restartで管理されているすべてのリスナーの実行ステータスを表示します。

- 構文およびオプション
- 例

親トピック: status

4.5.11.5.1 構文およびオプション

`srvctl status listener`コマンドは次の構文で使用します。

```
srvctl status listener [-listener listener_name] [-verbose]
```

表4-56 srvctl status listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、すべてのリスナーのステータスが表示されます。
`-verbose`	冗長出力。実行中のデータベース・サービスを一覧表示します。

親トピック: srvctl status listener

4.5.11.5.2 例

次に、このコマンドの例を示します。

```
srvctl status listener -listener crmlistener
Listener CRMLISTENER is running on dbhost
```

親トピック: srvctl status listener

4.5.11.6 srvctl status ons

Oracle Notification Services (ONS)の実行ステータスを表示します。

- 構文およびオプション

親トピック: status

4.5.11.6.1 構文およびオプション

`srvctl status ons`コマンドは次の構文で使用します。

```
srvctl status ons [-verbose]
```

表4-57 srvctl status onsのオプション

オプション	説明
`-verbose`	冗長出力。実行中のデータベース・サービスを一覧表示します。

親トピック: srvctl status ons

4.5.11.7 srvctl status service

1つ以上のデータベース・サービスの実行ステータスを表示します。

- 構文およびオプション
- 例

親トピック: status

4.5.11.7.1 構文およびオプション

srvctl status serviceコマンドは次の構文で使用します。

```
srvctl status service -db db_unique_name
  [-service service_name_list | -pdb pluggable_database]
  [-force] [-verbose]
```

表4-58 srvctl status serviceのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-service service_name_list	サービス名のカンマ区切りリスト。省略すると、指定されたデータベースの全データベース・サービスのステータスが一覧表示されます。
-pdb pluggable_database	マルチテナント・コンテナ・データベース(CDB)でサービスに関連付けられているプラガブル・データベース(PDB)の名前 このオプションを空の文字列に設定した場合、サービスはルートに関連付けられます。
-force	サービスが無効な場合、メッセージを表示します。
-verbose	冗長出力

親トピック: srvctl status service

4.5.11.7.2 例

次の例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのサービスsalesの実行のステータスを表示します。

```
srvctl status service -db dbcrm -service sales
Service sales is running on dbhost
```

親トピック: srvctl status service

4.5.12 stop

指定したコンポーネントを停止します。

srvctl stopコマンドを発行した後にコンポーネントを停止状態のままにする場合、コンポーネントを無効にします。disable コマンドを参照してください。

ノート:

コンポーネントを停止しても無効にしていない場合は、計画された別の操作の結果として再起動されることがあります。つまり、停止されたコンポーネントは障害の結果として再起動されることはありませんが、`srvctl start` コマンドで依存コンポーネントが起動されると、起動されることがあります。

- srvctl stop asm
  Oracle ASMインスタンスを停止します。
- srvctl stop database
  データベースとそのサービスを停止します。
- srvctl stop diskgroup
  Oracle ASMディスク・グループを停止(ディスマウント)します。
- srvctl stop home
  Oracle Restartで管理されているすべてのコンポーネントを、指定したOracleホーム内で停止します。OracleホームはOracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。
- srvctl stop listener
  指定したリスナーまたはOracle Restartで管理されているすべてのリスナーを停止します。リスナーを停止しても、リスナーに登録されたデータベースは停止されません。
- srvctl stop ons
  Oracle Notification Services (ONS)を停止します。
- srvctl stop service
  1つ以上のデータベース・サービスを停止します。

関連項目:

「Oracle Restartで管理されているコンポーネントの起動と停止」

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.12.1 srvctl stop asm

Oracle ASMインスタンスを停止します。

- 構文およびオプション
- 例

親トピック: stop

4.5.12.1.1 構文およびオプション

`srvctl stop asm`コマンドは次の構文で使用します。

```
srvctl stop asm [-stopoption stop_options] [-force]
```

表4-59 srvctl stop asmのオプション

オプション	説明
`-stopoption stop_options`	停止操作のオプション(`NORMAL`、`TRANSACTIONAL`、`IMMEDIATE`、`ABORT` など)。 関連項目: 停止オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』も参

オプション	説明
	照してください。
-force	強制。ディスク・グループが現在起動済(マウント済)の場合は、指定しておく必要があります。このオプションを指定すると、Oracle ASM を停止する前に、SRVCTL がディスク・グループを停止できるようになります。依存関係にある各データベース・インスタンスも停止オプションに従って停止されますが、構成されている停止オプションが失敗した場合は、ABORT オプションで停止されます。

親トピック: srvctl stop asm

4.5.12.1.2 例

次に、このコマンドの例を示します。

```
srvctl stop asm -stopoption abort -force
```

親トピック: srvctl stop asm

4.5.12.2 srvctl stop database

データベースおよびそのサービスを停止します。

- 構文およびオプション
- 例

親トピック: stop

4.5.12.2.1 構文およびオプション

srvctl stop databaseコマンドは次の構文で使用します。

```
srvctl stop database -db db_unique_name [-stopoption stop_options]
  [-drain_timeout timeout] [-force] [-verbose]
```

表4-60 srvctl stop databaseのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-stopoption stop_options	SHUTDOWN コマンド・オプション(NORMAL、TRANSACTIONAL、IMMEDIATE、ABORT など)。デフォルトは IMMEDIATE です。
-drain_timeout timeout	このオプションは、リソース・ドレインの完了までの許容時間を秒数で指定します。指定できる値は、NULL、0 または任意の正の整数です。

オプション	説明
	ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。ドレインの動作は、`-stopoption` オプションの設定によって異なります。 デフォルト値は `NULL` で、このオプションが設定されていないことを意味します。このオプションを設定せず、`-drain_timeout` がサービスに設定されている場合は、その値が使用されます。 `0` を設定した場合、ドレインは行われません。
`-force`	データベース、そのサービス、およびサービスに依存するリソースを停止します。
`-verbose`	冗長出力

親トピック: srvctl stop database

4.5.12.2.2 例

次に、このコマンドの例を示します。

```
srvctl stop database -db dbcrm
```

親トピック: srvctl stop database

4.5.12.3 srvctl stop diskgroup

Oracle ASMディスク・グループを停止(ディスマウント)します。

- 構文およびオプション
- 例

親トピック: stop

4.5.12.3.1 構文とオプション

`srvctl stop diskgroup`コマンドは次の構文で使用します。

```
srvctl stop diskgroup -diskgroup diskgroup_name [-force]
```

表4-61 srvctl stop diskgroupのオプション

オプション	説明
`-diskgroup diskgroup_name`	ディスク・グループ名
`-force`	強制。ディスク・グループの一部のファイルが開いている場合でも、ディスク・グループをディスマウントします。

親トピック: srvctl stop diskgroup

4.5.12.3.2 例

この例では、DATAというディスク・グループを停止します。このディスク・グループでファイルが開いている場合、エラーが返されます。

```
srvctl stop diskgroup -diskgroup DATA
```

親トピック: srvctl stop diskgroup

4.5.12.4 srvctl stop home

Oracle Restartで管理されているすべてのコンポーネントを、指定したOracleホーム内で停止します。Oracleホームは Oracle DatabaseホームまたはOracle Gridインフラストラクチャ・ホームです。

このコマンドでは指定した状態ファイルで停止したコンポーネントを特定します。

ノート:

- Oracle Grid インフラストラクチャ・ホームのコンポーネントを停止する前に、依存する Oracle Database ホームのコンポーネントを停止します。
- このコマンドは、Oracle ホームにパッチをインストールする前にコンポーネントを停止するときに使用します。

- 構文およびオプション

親トピック: stop

4.5.12.4.1 構文およびオプション

`srvctl stop home`コマンドは次の構文で使用します。

```
srvctl stop home -oraclehome oracle_home -statefile state_file
  [-stopoption stop_options] [-force]
```

表4-62 srvctl stop homeのオプション

オプション	説明
`-oraclehome oracle_home`	Oracle ホームの完全なパス。
`-statefile state_file`	状態ファイルに書き込む完全なパス。
`-stopoption stop_options`	データベースの SHUTDOWN コマンド・オプション(NORMAL、TRANSACTIONAL、IMMEDIATE、ABORT など)。デフォルトは IMMEDIATE です。 関連項目: 停止オプションの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』も参照してください。
`-force`	強制的に各コンポーネントを停止します。

親トピック: srvctl stop home

4.5.12.5 srvctl stop listener

指定したリスナーまたはOracle Restartで管理されているすべてのリスナーを停止します。リスナーを停止しても、リスナーに登録されたデータベースは停止されません。

- 構文およびオプション
- 例

親トピック: stop

4.5.12.5.1 構文およびオプション

`srvctl stop listener`コマンドは次の構文で使用します。

```
srvctl stop listener [-listener listener_name] [-force]
```

表4-63 srvctl stop listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、Oracle Restart で管理されているすべてのリスナーが停止されます。
`-force`	強制。`-f` オプションを指定した stop コマンドを Oracle Clusterware に渡します。Oracle Clusterware の-f オプションの詳細は、『Oracle Clusterware 管理およびデプロイメント・ガイド』を参照してください。

親トピック: srvctl stop listener

4.5.12.5.2 例

次に、このコマンドの例を示します。

```
srvctl stop listener -listener crmlistener
```

親トピック: srvctl stop listener

4.5.12.6 srvctl stop ons

Oracle Notification Services (ONS)を停止します。

- 構文およびオプション

親トピック: stop

4.5.12.6.1 構文およびオプション

`srvctl stop ons`コマンドは次の構文で使用します。

```
srvctl stop ons [-verbose]
```

表4-64 srvctl stop onsのオプション

オプション	説明

オプション	説明
-verbose	冗長出力

親トピック: srvctl stop ons

4.5.12.7 srvctl stop service

1つ以上のデータベース・サービスを停止します。

- 構文およびオプション
- 例

親トピック: stop

4.5.12.7.1 構文およびオプション

srvctl stop serviceコマンドは次の構文で使用します。

```
srvctl stop service -db db_unique_name [-service service_name_list |
  -pdb pluggable_database] [-drain_timeout timeout] [-stopoption stop_option]
  [-global_override] [-wait wait_option] [-force] [-verbose]
```

表4-65 srvctl stop serviceのオプション

オプション	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-service service_name_list	データベース・サービス名のカンマ区切りのリスト。サービス名リストを指定しない場合、SRVCTL によって、データベースのすべてのサービスが停止されます。
-pdb pluggable_database	CDB で、サービスに関連付けられた PDB の名前 このオプションを空の文字列に設定した場合、サービスはルートに関連付けられます。
-drain_timeout timeout	このオプションは、リソース・ドレインの完了までの許容時間を秒数で指定します。指定できる値は、NULL、0 または任意の正の整数です。 ドレイン期間は、計画的なメンテナンス操作のために意図されています。ドレイン期間中は、現在のすべてのクライアント要求は処理されますが、新しい要求は受け入れません。ドレインの動作は、-stopoption オプションの設定によって異なります。 デフォルト値は NULL で、このオプションが設定されていないことを意味します。このオプションを設定せず、-drain_timeout がサービスに設定されている場合は、その値が使用されま

オプション	説明
	す。 0 を設定した場合、ドレインは行われません。
-stopoption stop_option	このオプションは、サービスを停止するモードを指定します。次の値を使用できます。 ● IMMEDIATE は、サービスが停止する前にセッションを排出できるように指定します。 ● TRANSACTIONAL は、–drain_timeout オプションに指定されている時間だけセッションがドレインできるように指定します。この時間制限に達するとサービスが停止され、残っているセッションは終了します。 ● デフォルトは NONE です。
-global_override	サービスがグローバル・データ・サービス(GDS)のサービスである場合、サービスを停止するときには、このオプションを指定する必要があります。 GDS サービスを停止しようとしたときに、-global_override が含まれていない場合は、エラーが返されます。 サービスが GDS サービスでない場合、このオプションは無視されます。 詳細は、Oracle Database Global Data Services 概要および管理ガイドを参照してください。
-wait wait_option	このオプションは、サービスを停止する前に、サービス・ドレインが完了するまで待つかどうかを指定します。待つ場合は YES、待たずにサービスを停止する場合は NO を指定します。
-force	強制。このオプションでは、停止したすべてのサービスのセッションが即座に切断されます。コミットされていないトランザクションはロールバックされます。このオプションを指定しないと、アクティブなセッションはサービスに接続されたままになりますが、それ以上サービスには接続できなくなります。
-verbose	冗長出力

親トピック: srvctl stop service

4.5.12.7.2 例

次の例では、DB_UNIQUE_NAMEがdbcrmであるデータベースのsalesデータベース・サービスを停止します。

```
srvctl stop service -db dbcrm -service sales
```

親トピック: srvctl stop service

4.5.13 unsetenv

unsetenvコマンドでは、データベース、リスナーまたはOracle ASMインスタンスのOracle Restart構成から1つ以上の環境変数を削除します。

srvctl unsetenv操作を実行するには、適切なユーザー・アカウントを使用してデータベース・ホスト・コンピュータにログインする必要があります。詳細は、「SRVCTLの実行準備」を参照してください。

- srvctl unsetenv asm
  Oracle ASMインスタンスのOracle Restart構成から指定した環境変数を削除します。
- srvctl unsetenv database
  指定したデータベースのOracle Restart構成から指定した環境変数を削除します。
- srvctl unsetenv listener
  指定したリスナーまたはすべてのリスナーのOracle Restart構成から指定した環境変数を削除します。

関連項目:

- setenvコマンド
- getenvコマンド
- 「Oracle Restart構成の環境変数の管理」

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.13.1 srvctl unsetenv asm

Oracle ASMインスタンスのOracle Restart構成から指定した環境変数を削除します。

- 構文およびオプション
- 例

親トピック: unsetenv

4.5.13.1.1 構文およびオプション

srvctl unsetenv asmコマンドは次の構文で使用します。

```
srvctl unsetenv asm -envs name_list
```

表4-66 srvctl unsetenv asmのオプション

オプション	説明
-envs name_list	削除する環境変数のカンマ区切りのリスト

親トピック: srvctl unsetenv asm

4.5.13.1.2 例

次の例では、AIXオペレーティング・システムの環境変数AIXTHREAD_SCOPEをOracle ASMインスタンスの構成から削除します。

```
srvctl unsetenv asm -envs AIXTHREAD_SCOPE
```

親トピック: srvctl unsetenv asm

4.5.13.2 srvctl unsetenv database

指定したデータベースのOracle Restart構成から指定した環境変数を削除します。

- 構文およびオプション
- 例

親トピック: unsetenv

4.5.13.2.1 構文およびオプション

`srvctl unsetenv database`コマンドは次のように使用します。

```
srvctl unsetenv database -db db_unique_name -envs name_list
```

表4-67 srvctl unsetenv databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
`-envs name_list`	削除する環境変数のカンマ区切りのリスト

親トピック: srvctl unsetenv database

4.5.13.2.2 例

次の例では、DB_UNIQUE_NAMEが`dbcrm`であるデータベースのOracle Restart構成から`AIXTHREAD_SCOPE`環境変数を削除します。

```
srvctl unsetenv database -db dbcrm -envs AIXTHREAD_SCOPE
```

親トピック: srvctl unsetenv database

4.5.13.3 srvctl unsetenv listener

指定したリスナーまたはすべてのリスナーのOracle Restart構成から指定した環境変数を削除します。

- 構文およびオプション
- 例

親トピック: unsetenv

4.5.13.3.1 構文およびオプション

`srvctl unsetenv listener`コマンドは次の構文で使用します。

```
srvctl unsetenv listener [-listener listener_name] -envs name_list
```

表4-68 srvctl unsetenv listenerのオプション

オプション	説明
`-listener listener_name`	リスナー名。省略すると、すべてのリスナーの構成から指定した環境変数が削除されます。
`-envs name_list`	削除する環境変数のカンマ区切りのリスト

親トピック: srvctl unsetenv listener

4.5.13.3.2 例

次の例では、`crmlistener`というリスナーのリスナー構成からAIXオペレーティング・システムの環境変数`AIXTHREAD_SCOPE`を削除します。

```
srvctl unsetenv listener -listener crmlistener -envs AIXTHREAD_SCOPE
```

親トピック: srvctl unsetenv listener

4.5.14 update

`srvctl update`コマンドは、実行中のデータベースを更新して指定の起動オプションに切り替えます。

- srvctl update database
  `srvctl update database`コマンドは、データベースのオープン・モードを変更します。

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.14.1 srvctl update database

`srvctl update database`コマンドは、データベースのオープン・モードを変更します。

- 構文およびオプション

親トピック: update

4.5.14.1.1 構文およびオプション

`srvctl update database`コマンドは、次のように使用します。

```
srvctl update database -db db_unique_name --startoption start_options
```

表4-69 srvctl upgrade databaseのオプション

オプション	説明
`-db db_unique_name`	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
`-startoption start_options`	データベースの起動オプション。起動オプションには、OPEN、MOUNT、READ ONLY などがあります。

親トピック: srvctl update database

4.5.15 upgrade

srvctl upgradeコマンドは、リソース・タイプとリソースを古いバージョンから新しいバージョンにアップグレードします。

- srvctl upgrade database
  srvctl upgrade databaseコマンドは、このコマンドの実行元であるデータベース・ホームのバージョンに、データベースの構成とそのすべてのサービスをアップグレードします。

親トピック: Oracle RestartのSRVCTLコマンド・リファレンス

4.5.15.1 srvctl upgrade database

srvctl upgrade databaseコマンドは、このコマンドの実行元であるデータベース・ホームのバージョンに、データベースの構成とそのすべてのサービスをアップグレードします。

- 構文およびオプション

親トピック: upgrade

4.5.15.1.1 構文およびオプション

srvctl upgrade databaseコマンドは、次の構文で使用します。

```
srvctl upgrade database -db db_unique_name -oraclehome oracle_home
```

表4-70 srvctl upgrade databaseのオプション

パラメータ	説明
-db db_unique_name	データベースの一意の名前。DB_UNIQUE_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME が指定されていない場合、このオプションは DB_NAME 初期化パラメータの設定に一致する必要があります。DB_UNIQUE_NAME のデフォルト設定では DB_NAME の設定を使用します。
-oraclehome oracle_home	データベースの Oracle ホームのフルパス。

親トピック: srvctl upgrade database

4.6 CRSCTLコマンド・リファレンス

Oracle Restartに関連のあるCRSCTLコマンドの構文の詳細を参照できます。

ノート:

これらの CRSCTL コマンドを実行するには、root ユーザーであるか、または Oracle Grid インフラストラクチャの所有者である必要があります。

CRSCTLコマンド構文の概要

CRSCTLでは、次のコマンド構文が想定されます。

```
crsctl command has
```

commandは`start`、`stop`、`enable`などの動詞です。hasオブジェクトはOracle高可用性サービスを示します。

大/小文字の区別

CRSCTLコマンドとコンポーネントでは大/小文字が区別されません。

- check
  Oracle Restartの状態を表示します。
- config
  Oracle Restartの構成を表示します。
- disable
  Oracle Restartの自動再起動を無効化します。
- enable
  Oracle Restartの自動再起動を有効化します。
- start
  Oracle Restartを起動します。
- stop
  Oracle Restartを停止します。

親トピック: Oracle Databaseの自動再起動の構成

4.6.1 check

Oracle Restartの状態を表示します。

構文およびオプション

```
crsctl check has
```

親トピック: CRSCTLコマンド・リファレンス

4.6.2 config

Oracle Restartの構成を表示します。

構文およびオプション

```
crsctl config has
```

親トピック: CRSCTLコマンド・リファレンス

4.6.3 disable

Oracle Restartの自動再起動を無効化します。

構文およびオプション

```
crsctl disable has
```

親トピック: CRSCTLコマンド・リファレンス

4.6.4 enable

Oracle Restartの自動再起動を有効化します。

構文およびオプション

```
crsctl enable has
```

親トピック: CRSCTLコマンド・リファレンス

4.6.5 start

Oracle Restartを起動します。

構文およびオプション

```
crsctl start has
```

親トピック: CRSCTLコマンド・リファレンス

4.6.6 stop

Oracle Restartを停止します。

構文およびオプション

```
crsctl stop has [-f]
```

表4-71 crsctl stop hasのオプション

オプション	説明
`-f`	強制。Oracle Restart で管理されるリソースがまだ実行中の場合は、それらのリソースを正常に停止しようとします。リソースを正常に停止できない場合は、リソースを強制的に停止しようとします。 たとえば、Oracle ASM インスタンスが実行中の場合、`SHUTDOWN IMMEDIATE` では Oracle ASM インスタンスを正常に停止しようとしますが、`SHUTDOWN ABORT` では Oracle ASM インスタンスを強制的に停止しようとします。 `-f` オプションが指定されていない場合、このコマンドは Oracle Restart で管理されるリソースを正常に停止しようとしますが、強制的に停止しようとはしません。 ノート: データベース・リソースの場合、`-f` オプションが指定されているかどうかにかかわらず、このコマンドは常に `SHUTDOWN ABORT` を使用します。

親トピック: CRSCTLコマンド・リファレンス

5 プロセスの管理

Oracle Databaseでは、複数のユーザーおよびアプリケーションが同時に1つのデータベース・インスタンスに接続できるように、複数のプロセスを使用しています。

- 専用サーバー・プロセスと共有サーバー・プロセスについて
  Oracle Databaseでは、インスタンスに接続されているユーザー・プロセスの要求を処理するために、サーバー・プロセスが作成されます。
- データベース常駐接続プーリングの理解
  データベース常駐接続プーリング(DRCP)は、データベース接続を取得し、比較的短時間の処理を実行した後、データベース接続を解放するような一般的なWebアプリケーション使用に対して、データベース・サーバーの接続プールを提供します。DRCPは専用サーバーをプールします。プール・サーバーは、サーバー・フォアグラウンド・プロセスとデータベース・セッションの組合せに相当します。
- プロキシ常駐接続プーリングについて
  プロキシ常駐接続プーリングには、Traffic DirectorモードのOracle Connection Managerを使用して構成できるプロキシ常駐接続プールを使用します。プロキシ常駐接続プーリングにより、データベース・クライアントの高可用性、高度なセキュリティおよびパフォーマンスが提供されます。
- Oracle Databaseの共有サーバー構成
  共有サーバーを有効にし、共有サーバーの初期化パラメータを設定または変更できます。
- データベース常駐接続プーリングの構成
  データベース・サーバーは、データベース常駐接続プーリングを使用できるように事前構成されています。ただし、この機能は、接続プールを起動して明示的に使用可能にする必要があります。
- Oracle Databaseバックグランド・プロセスについて
  マルチプロセスのOracle Databaseシステムでは、最高のパフォーマンスを提供し、多くのユーザーが同時に使用できるように、バックグラウンド・プロセスが使用されています。バックグラウンド・プロセスでは、ユーザー・プロセスごとに実行される複数のデータベース・プログラムによって処理される機能が統合されます。バックグラウンド・プロセスは、非同期的にI/Oを実行して他のOracle Databaseプロセスを監視することによって、並列性を高め、パフォーマンスと信頼性を向上させます。
- 事前作成されたプロセスの管理
  Oracle Databaseでは、プロセスを事前に作成してクライアント接続パフォーマンスを向上できます。
- SQLのパラレル実行用プロセスの管理
  SQL文のパラレル処理を管理できます。この構成では、Oracle Databaseによって、SQL文の処理作業を複数のパラレル・プロセスに分割できます。
- 外部プロシージャのプロセスの管理
  外部プロシージャは、プログラミング言語で作成され、共有ライブラリに格納されるプロシージャまたはファンクションです。Oracleサーバーは、PL/SQLルーチンを使用して、外部プロシージャまたはファンクションをコールできます。
- セッションの停止
  時々、現行のユーザー・セッションを停止する必要があります。たとえば、管理操作を実行する場合に、すべての管理セッション以外のセッションを停止する必要があります。
- プロセスおよびセッションのデータ・ディクショナリ・ビュー
  プロセスおよびセッションの情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

親トピック: 基本データベース管理

5.1 専用サーバー・プロセスと共有サーバー・プロセスについて

Oracle Databaseでは、インスタンスに接続されているユーザー・プロセスの要求を処理するために、サーバー・プロセスが作成されます。

サーバー・プロセスには、次の2つのプロセスがあります。

- 専用サーバー・プロセス。単一のユーザー・プロセスのみを処理します。
- 共有サーバー・プロセス。複数のユーザー・プロセスを処理できます。

データベースでは、専用サーバー・プロセスは常に使用可能な状態ですが、共有サーバーは、1つ以上の初期化パラメータを特別に設定して、構成および使用可能にする必要があります。

- 専用サーバー・プロセス
  専用サーバー・プロセスは、1つのユーザー・プロセスのみを処理します。
- 共有サーバー・プロセス
  共有サーバー・プロセスは、複数のユーザー・プロセスを処理できます。

親トピック: プロセスの管理

5.1.1 専用サーバー・プロセス

専用サーバー・プロセスは、1つのユーザー・プロセスのみを処理します。

図5-1は、専用サーバー・プロセスの仕組みを示しています。この図では、専用サーバー・プロセスを介して、2つのユーザー・プロセスがデータベースに接続されています。

一般的には、共有サーバーを使用し、ディスパッチャを介して接続することをお薦めします。これを図5-2に示します。共有サーバー・プロセスは、実行中のインスタンスに必要なプロセスの数を少なくできるため、効率が向上します。

ただし、次の状況では、ユーザーと管理者は、専用サーバー・プロセスを使用して明示的にインスタンスに接続する必要があります。

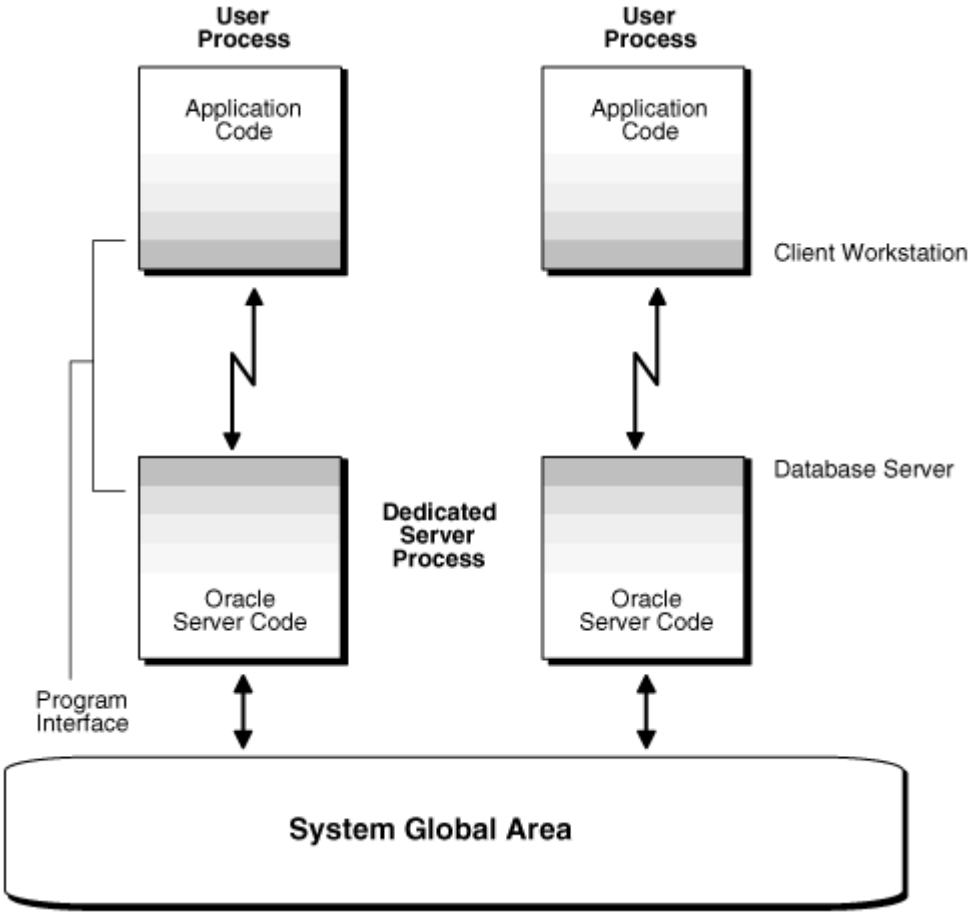
- バッチ・ジョブを実行する場合(たとえば、ジョブがサーバー・プロセスに対して持つアイドル時間がほとんどないか、まったくない場合)
- Recovery Manager(RMAN)を使用して、データベースをバックアップ、リストアまたはリカバリする場合

Oracle Databaseが共有サーバー用に構成されている場合に専用サーバー接続を要求するには、専用サーバーを使用するように構成されているネット・サービス名を使用して接続する必要があります。具体的に言うと、ネット・サービス名の接続記述子にSERVER=DEDICATED句を含めます。

関連項目:

専用サーバー接続を要求する方法の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

図5-1 Oracle Databaseの専用サーバー・プロセス



親トピック: 専用サーバー・プロセスと共有サーバー・プロセスについて

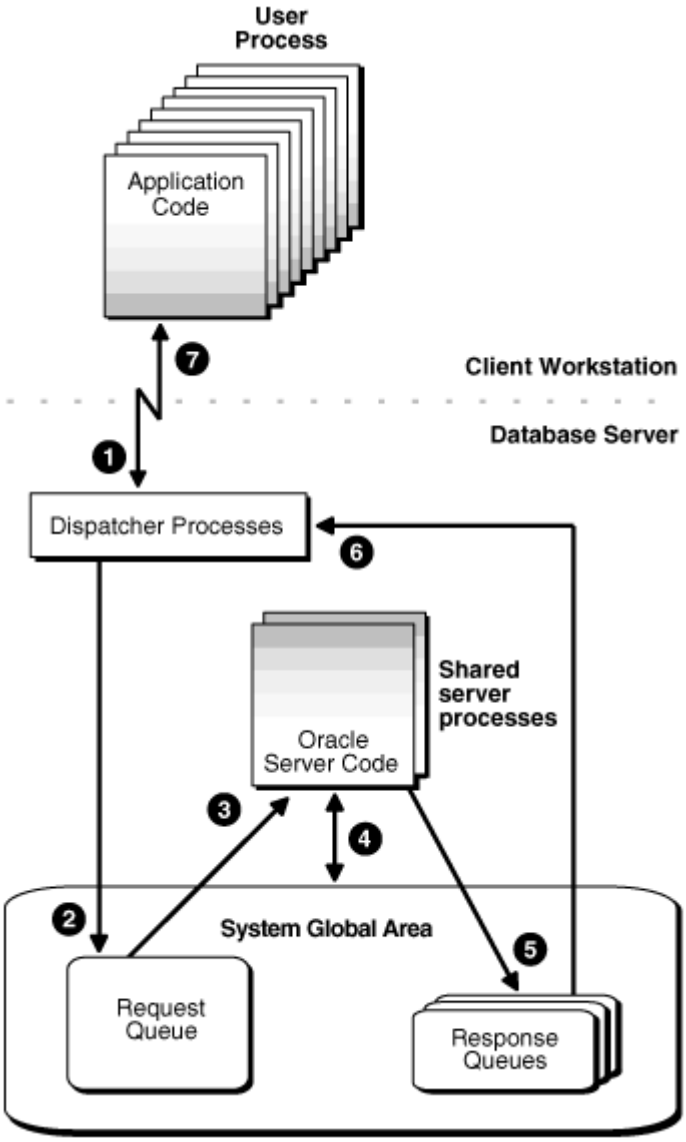
5.1.2 共有サーバー・プロセス

共有サーバー・プロセスは、複数のユーザー・プロセスを処理できます。

専用サーバー・プロセスを使用した注文入力システムを例にとって考えてみます。顧客が注文受付に電話で商品を注文すると、担当者はその注文をデータベースに入力します。取引のほとんどの間、担当者は顧客と電話で話しています。この間、サーバー・プロセスは必要ないため、担当者のユーザー・プロセス専用のサーバー・プロセスはアイドル状態のままです。アイドル状態のサーバー・プロセスはシステム・リソースを保持しているため、他の担当者が注文を入力する際にシステムのパフォーマンスが低下します。

共有サーバー・アーキテクチャでは、接続ごとに専用サーバー・プロセスは必要ありません(図5-2を参照)。

図5-2 Oracle Databaseの共有サーバー・プロセス



共有サーバー構成を使用する場合、クライアントのユーザー・プロセスはディスパッチャに接続します。ディスパッチャは複数のクライアント接続を同時にサポートできます。各クライアント接続はバーチャル・サーキットにバインドされており、これはディスパッチャが使用する共有メモリーの1つで、クライアント・データベースの接続要求および応答を目的としています。ディスパッチャは、リクエストが到着するとバーチャル・サーキットを共通キューに配置します。

アイドル状態の共有サーバー・プロセスは、共通キューからバーチャル・サーキットを選択して要求を処理し、そのバーチャル・サーキットを解放して、共通キューから別のバーチャル・サーキットを取り出します。このアプローチでは、小さいサーバー・プロセス・プールで大量のクライアントを処理することが可能です。専用サーバー・モデルと比較した共有サーバー・アーキテクチャの大きな利点は、システム・リソースが少なくて済むため、ユーザー数の増加に対応できることです。

リソース管理をさらに向上させるために、共有サーバーにセッション多重化を構成でき、これにより、オペレーティング・システムのリソースを節約するために、単一のネットワーク接続で通信用の複数のセッションが結合されます。

共有サーバー・アーキテクチャには、Oracle Net Servicesが必要です。共有サーバーを使用するユーザー・プロセスは、Oracle Databaseインスタンスと同じシステム上にある場合でも、必ずOracle Net Servicesを介して接続してください。

関連項目:

セッションの多重化などの機能を含む共有サーバーの詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

5.2 データベース常駐接続プーリングの理解

データベース常駐接続プーリング(DRCP)は、データベース接続を取得し、比較的短時間の処理を実行した後、データベース接続を解放するような一般的なWebアプリケーション使用に対して、データベース・サーバーの接続プールを提供します。DRCPは専用サーバーをプールします。プール・サーバーは、サーバー・フォアグラウンド・プロセスとデータベース・セッションの組合せに相当します。

DRCPは、中間層プロセス内のスレッド間で接続を共有する中間層接続プールを補完します。また、DRCPを使用すると、同じ中間層ホスト上の中間層プロセス間、および異なる中間層ホスト上の中間層プロセス間でもデータベース接続を共有できます。この結果、大量のクライアント接続をサポートするために必要になる基本データベース・リソースが大幅に減少するため、データベース層のメモリー・フットプリントが縮小し、中間層とデータベース層の両方のスケーラビリティが向上します。すぐに使用できるサーバーのプールを保持することで、クライアント接続の作成と切断のコストが削減されるという利点もあります。

DRCPは、中間層接続プーリングを実行できない(PHP/Apacheなどの)マルチプロセスのシングル・スレッド・アプリケーション・サーバーを含むアーキテクチャに特に適しています。データベースは、DRCPを使用すると同時接続の数を数万まで増やすことができます。

ノート:

- Oracle Database 12c リリース 2 (12.2)以降では、同じユーザーに属するプロキシ・セッションを共有できます。
- Windows プラットフォームでは、`SQLNET.AUTHENTICATION_SERVICES` パラメータ値の `nts` 設定が DRCP でサポートされていません。

関連項目:

- DRCPの詳細は、『Oracle Database概要』を参照してください。
- DRCPの使用に関する制限事項など、DRCPの詳細は、『Oracle Database開発ガイド』を参照してください。
- DRCPセッションを取得する場合に使用できるオプションの詳細は、『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。
- プロキシ・セッションの共有については、『Oracle Database開発者ガイド』

データベース常駐接続プーリングを使用する場合

データベース常駐接続プーリングは、複数のクライアントがデータベースにアクセスする場合で、次のいずれかに該当する場合に有効です。

- 大量のクライアント接続を最小限のメモリー使用でサポートする必要がある場合。
- 複数の類似クライアント・アプリケーションがあり、セッションの共有または再利用が可能な場合。

  類似アプリケーションとは、同じデータベース資格証明で接続し、同じスキーマを使用するアプリケーションです。
- クライアント・アプリケーションでデータベース接続を取得し、比較的短時間の処理を実行した後、データベース接続を

解放する場合。

- クライアント要求にまたがるセッション・アフィニティが必要ない場合。
- クライアント側に複数のプロセスおよび複数のホストが存在する場合。

データベース常駐接続プーリングの利点

データベース常駐接続プーリングの使用には次の利点があります。

- 複数の中間層クライアント・アプリケーション間でリソースを共有できます。
- リソース使用量が減少するため、データベースおよびアプリケーションのスケーラビリティが向上します。

データベース常駐接続プーリングおよびLOGON/LOGOFFトリガー

LOGONトリガーは、DRCPで認証が実施されるたびに、または新規セッションが作成されるたびに起動します。

LOGOFFトリガーは、DRCPでログオフされるたび、またセッションが破棄されると起動します。このため、LOGOFFトリガーは、アイドル時間制限のために、セッションが終了すると起動します。

- DRCP、専用サーバーおよび共有サーバーの比較
  専用サーバー、共有サーバーおよびデータベース常駐接続プーリングの相違点を理解します。

関連項目:

- Oracle Database PL/SQL言語リファレンス
- Oracle Databaseセキュリティ・ガイド

親トピック: プロセスの管理

5.2.1 DRCP、専用サーバーおよび共有サーバーの比較

専用サーバー、共有サーバーおよびデータベース常駐接続プーリングの相違点を理解します。

表5-1に、専用サーバー、共有サーバーおよびデータベース常駐接続プーリングの相違点を示します。

表5-1 専用サーバー、共有サーバーおよびデータベース常駐接続プーリング

専用サーバー	共有サーバー	データベース常駐接続プーリング
クライアント要求を受け取ると、クライアント用に新規サーバー・プロセスとセッションが作成されます。	クライアントから最初の要求を受け取ると、ディスパッチャ・プロセスによって要求が共通キューに入れられます。要求は使用可能な共有サーバー・プロセスによって取り出されます。その後のクライアントと共有サーバー・プロセス間の通信はディスパッチャ・プロセスによって管理されます。	クライアントから最初の要求を受け取ると、接続ブローカによって使用可能なプール・サーバーが選択され、そのプール・サーバーにクライアント接続が渡されます。 使用可能なプール・サーバーがない場合は、接続ブローカによって作成されます。プールが最大サイズに達した場合、クライアント要求は、プール・サーバーが使用可能になるまで待機キューに入れられます。
データベース・リソースを解放すると、セッ	データベース・リソースを解放すると、セッ	データベース・リソースを解放すると、プー

専用サーバー	共有サーバー	データベース常駐接続プーリング
ションおよびサーバー・プロセスが終了します。	ションが終了します。	ル・サーバーがプールに解放されます。
メモリー要件は、サーバー・プロセスとセッションの数に比例します。クライアントごとに 1 つのサーバーと 1 つのセッションが存在します。	メモリー要件は、共有サーバーとセッションの合計に比例します。クライアントごとに 1 つのセッションが存在します。	メモリー要件は、プール・サーバーとそのセッションの数に比例します。プール・サーバーごとに 1 つのセッションが存在します。
セッション・メモリーは PGA から割り当てられます。	セッション・メモリーは SGA から割り当てられます。	セッション・メモリーは PGA から割り当てられます。

専用サーバー、共有サーバーおよびデータベース常駐接続プーリングのメモリー使用量の例

各セッションに必要なメモリーが400KB、各サーバー・プロセスに必要なメモリーが4MBであるアプリケーションについて考えます。プール・サイズは100で、使用される共有サーバーの数は100です。

5000のクライアント接続がある場合、各構成で使用されるメモリーは次のようになります。

- 専用サーバー

  使用メモリー = 5000 X (400KB + 4MB) = 22GB

- 共有サーバー

  使用メモリー = 5000 X 400KB + 100 X 4MB = 2.5GB

  2.5GBのうち2GBはSGAから割り当てられます。

- データベース常駐接続プーリング

  使用メモリー = 100 X (400KB + 4MB) + (5000 X 35KB)= 615MB

  ブローカへの各接続のコストは、約35KBです。

親トピック: データベース常駐接続プーリングの理解

5.3 プロキシ常駐接続プーリングについて

プロキシ常駐接続プーリングには、Traffic DirectorモードのOracle Connection Managerを使用して構成できるプロキシ常駐接続プールを使用します。プロキシ常駐接続プーリングにより、データベース・クライアントの高可用性、高度なセキュリティおよびパフォーマンスが提供されます。

プロキシ常駐接続プーリングを使用してデータベース・インスタンスに接続できるデータベース・クライアントは、Oracle Call Interface (OCI)、Java Database Connectivity (JDBC)、Oracle Data Provider for .NET (ODP.Net)、Open Database Connectivity (ODBC)、Pro*C、Pro*COBOL、PHP OCI8拡張モジュール、Node.js node-oracledbドライバ、Python cx_Oracle、ROracle、Ruby-oci8、Perl DBD::OracleまたはOracle C++ Call Interface (OCCI)のいずれかのテクノロジに基づいている必要があります。

ノート:

プロキシ常駐接続プーリングは、Oracle Database 18c 以降で使用できます。

プロキシ常駐接続プーリングを使用する場合

プロキシ常駐接続プーリングは、複数のクライアントがデータベースにアクセスする場合で、次のいずれかに該当する場合に有効です。

- 大量のクライアント接続を、それより少ないデータベースへの接続数でサポートする必要がある場合。
- 1つのデータベース接続を中間層接続プール間で共有する必要がある場合。
- 64Kを超えるセッションをサポートする必要がある場合(64kセッションの制限があるため共有サーバーを使用できない場合)。
- 透過アプリケーション・フェイルオーバー(TAF)とOracle RACをサポートしていない古いクライアントや、Oracleデータベース常駐接続プーリング(DRCP)、高速アプリケーション通知(FAN)またはアプリケーション・コンティニュイティ(AC)を使用しないクライアントで高可用性をサポートする必要がある場合。

プロキシ常駐接続プーリングの利点

プロキシ常駐接続プーリングの使用には、次の主な利点があります。

- 高可用性の向上(計画と計画外)
- データベース・セキュリティの向上
- データベース接続の多重化

関連項目:

Traffic DirectorモードのOracle Connection Managerを使用したプロキシ常駐接続プーリングの有効化の詳細は、『*Oracle Database Net Services*管理者ガイド』を参照してください。

- 「Traffic DirectorモードのOracle Connection Managerの使用について」
- 「Traffic DirectorモードのOracle Connection Managerの構成」

親トピック: プロセスの管理

5.4 Oracle Databaseの共有サーバー構成

共有サーバーを有効にし、共有サーバーの初期化パラメータを設定または変更できます。

- 共有サーバー用初期化パラメータ
  共有サーバーの処理は、初期化パラメータのセットにより制御されます。
- 共有サーバーのメモリー管理
  共有サーバーが動作するには、共有プールまたはラージ・プールにユーザー・グローバル領域(UGA)が必要です。同時セッション数が少ないインストールでは、通常、これらのシステム・グローバル領域(SGA)コンポーネントのデフォルト・サイズで十分です。ただし、セッション数が多い場合は、共有サーバーに対応できるようにメモリーをチューニングする必要があります。
- 共有サーバーの使用可能化

共有サーバーを使用可能にするには、SHARED_SERVERS初期化パラメータに0(ゼロ)より大きい値を設定します。他の共有サーバー初期化パラメータの設定は不要です。

- ディスパッチャの構成
  DISPATCHERS初期化パラメータは、共有サーバー・アーキテクチャ内のディスパッチャ・プロセスを構成します。共有サーバーが動作するためには、1つのディスパッチャ・プロセスが最小限必要です。ディスパッチャを指定せずに、SHARED_SERVERを0(ゼロ)以外の値に設定して共有サーバーを使用可能にすると、デフォルトで、Oracle Databaseによって1つのディスパッチャがTCPプロトコル用に作成されます。
- 共有サーバーの無効化
  共有サーバーを無効にするには、SHARED_SERVERSを0に設定します。この操作は、ALTER SYSTEM文を使用して動的に行うことができます。
- 共有サーバーのデータ・ディクショナリ・ビュー
  共有サーバーの構成情報およびパフォーマンスの監視のためにデータ・ディクショナリ・ビューを問い合せることができます。

関連項目:

- 「専用サーバー・プロセスと共有サーバー・プロセスについて」
- ALTER SYSTEM文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: プロセスの管理

5.4.1 共有サーバー用初期化パラメータ

初期化パラメータのセットにより、共有サーバー操作を制御します。

共有サーバー操作を制御する初期化パラメータは、次のとおりです。

- SHARED_SERVERS: 起動する初期共有サーバー数および最低限保持する共有サーバー数を指定します。共有サーバーを使用するための必須パラメータはこのパラメータのみです。
- MAX_SHARED_SERVERS: 同時に実行可能な共有サーバーの最大数を指定します。
- SHARED_SERVER_SESSIONS: 同時に実行可能な共有サーバー・ユーザー・セッションの合計数を指定します。このパラメータを設定すると、専用サーバーのユーザー・セッションを確保することが可能になります。
- DISPATCHERS: 共有サーバー・アーキテクチャのディスパッチャ・プロセスを構成します。
- MAX_DISPATCHERS: 同時に実行できるディスパッチャ・プロセスの最大数を指定します。このパラメータは、現在は無視できます。将来のリリースで、同時接続数によってディスパッチャ数が自動的にチューニングされるようになるまで使用しません。
- CIRCUITS: 受信および発信用のネットワーク・セッションに使用可能なバーチャル・サーキットの合計数を指定します。

関連項目:

これらの初期化パラメータの詳細は、『Oracle Databaseリファレンス』

親トピック: Oracle Databaseの共有サーバー構成

5.4.2 共有サーバーのメモリー管理

共有サーバーが動作するには、共有プールまたはラージ・プールにユーザー・グローバル領域(UGA)が必要です。同時セッション数が少ないインストールでは、通常、これらのシステム・グローバル領域(SGA)コンポーネントのデフォルト・サイズで十分です。ただし、セッション数が多い場合は、共有サーバーに対応できるようにメモリーをチューニングする必要があります。

ガイドラインは、『Oracle Databaseパフォーマンス・チューニング・ガイド』のメモリーの構成と使用に関する項を参照してください。

親トピック: Oracle Databaseの共有サーバー構成

5.4.3 共有サーバーの使用可能化

共有サーバーを使用可能にするには、SHARED_SERVERS初期化パラメータに0(ゼロ)より大きい値を設定します。他の共有サーバー初期化パラメータの設定は不要です。

- 動的に共有サーバーを設定するには、ALTER SYSTEM文でSHARED_SERVERS初期化パラメータを0以外の値に設定します。
- データベースの起動時にSHARED_SERVERS初期化パラメータを0以外の値に設定するには、それを初期化パラメータ・ファイルに含めます。

共有サーバーが動作するには最低1つのディスパッチャが必要であるため、構成されていない場合でも、ディスパッチャは起動されます。ディスパッチャについては、「ディスパッチャの構成」を参照してください。

ノート:

SHARED_SERVERS がデータベースの起動時に初期化パラメータ・ファイルに組み込まれていない場合でも、DISPATCHERS が組み込まれ、少なくとも 1 つのディスパッチャが指定されていると、共有サーバーは使用可能になります。この場合、SHARED_SERVERS のデフォルトは 1 です。

SHARED_SERVERS と DISPATCHERS のいずれも初期化ファイルに組み込まれていない場合は、インスタンスが起動された後に DISPATCHERS パラメータを単に変更しても共有サーバーを起動することはできません。共有サーバーを起動するには、SHARED_SERVERS を 0(ゼロ)以外の値に変更する必要があります。

ノート:

Database Configuration Assistant(DBCA)で Oracle Database を作成した場合、DBCA によって Oracle XML DB(XDB)のディスパッチャが構成されます。これは、HTTP や FTP などの XDB プロトコルには共有サーバーが必要になるためです。これにより、SHARED_SERVER の値は 1 になります。共有サーバーは使用可能ですが、この構成では、XDB サービスに接続して共有サーバーを使用できるのは 1 つのセッションのみです。共有サーバーを標準的なデータベース・セッション(SQL 文を発行)に対応できるようにするには、ディスパッチャ構成を追加するか、既存の構成を XDB 固有ではない構成に置き換える必要があります。手順については、「ディスパッチャの構成」を参照してください。

- SHARED_SERVERSの値の決定について
  SHARED_SERVERS初期化パラメータは、インスタンスの起動時に作成する共有サーバーの最小数を指定します。インスタンスの起動後は、既存の共有サーバーがビジーになる頻度と要求キューの長さに基づいて、Oracle Database

が共有サーバーの数を動的に調整します。

- 共有サーバー・プロセス数の削減
  最低限アクティブに保つ必要がある共有サーバーの数を削減するには、SHARED_SERVERSパラメータを小さい値に動的に設定します。設定後は、共有サーバー数がSHARED_SERVERSパラメータの値に減少するまで、非アクティブになる共有サーバーにはPMONによって終了のマークが付けられます。
- 共有サーバー・プロセス数の制限
  MAX_SHARED_SERVERS初期化パラメータは、PMONが自動的に作成可能な共有サーバーの最大数を指定します。デフォルト値はありません。
- 共有サーバー・セッション数の制限
  SHARED_SERVER_SESSIONS初期化パラメータは、同時共有サーバー・ユーザー・セッションの最大数を指定します。
- 共有メモリーの保護
  CIRCUITS初期化パラメータは、共有メモリーに作成可能なバーチャル・サーキットの最大数を設定します。このパラメータにデフォルトはありません。値が指定されていない場合は、DISPATCHERS初期化パラメータおよびシステム・リソースの制限範囲内でサーキットが必要に応じて作成されます。

親トピック: Oracle Databaseの共有サーバー構成

5.4.3.1 SHARED_SERVERSの値の決定について

SHARED_SERVERS初期化パラメータは、インスタンスの起動時に作成する共有サーバーの最小数を指定します。インスタンスの起動後は、既存の共有サーバーがビジーになる頻度と要求キューの長さに基づいて、Oracle Databaseが共有サーバーの数を動的に調整します。

標準的なシステムでは、共有サーバーの数は、10接続ごとに共有サーバー1つという割合で安定します。OLTPアプリケーションでは、要求率が低い場合、または要求に対してサーバー使用の比率が低い場合は、接続数/サーバー数の割合が高くなります。対照的に、要求率が高いか、要求に対してサーバー使用の比率が大きいアプリケーションの場合は、接続数/サーバー数の割合が低くなります。

PMON(プロセス・モニター)バックグラウンド・プロセスは、SHARED_SERVERSで指定された値を下回る状態まで共有サーバーを終了させることはできません。したがって、このパラメータを使用すると、偶発的な負荷の変動のためにPMONが共有サーバーを終了および再起動することがなくなるため、負荷が安定し、システムへの過負荷を最小にできます。

システムの平均的な負荷がわかっている場合は、SHARED_SERVERSを最適な値に設定できます。次に、このパラメータの使用方法の例を示します。

1000人の従業員が勤務するテレマーケティング・センターでデータベースが使用されているとします。従業員は、平均して勤務時間の90%を顧客との電話応対に費やし、レコードの検索と更新には10%のみ費やしています。従業員が顧客と電話をしている間に共有サーバーが終了し、データベースにアクセスしたときに再び起動することがないように、DBAは最適な共有サーバー数として100を指定しています。

ただし、すべての勤務時間帯で同じ数の従業員が勤務しているわけではありません。夜間の時間帯は200人の従業員で十分です。SHARED_SERVERSは動的なパラメータであるため、DBAは、夜間の共有サーバーの数を20に減らし、バッチ・ジョブなどの他のタスクにリソースが解放されるようにします。

親トピック: 共有サーバーの使用可能化

5.4.3.2 共有サーバー・プロセス数の削減

最低限アクティブに保つ必要がある共有サーバーの数を削減するには、SHARED_SERVERSパラメータを小さい値に動的に設

定します。設定後は、共有サーバー数がSHARED_SERVERSパラメータの値に減少するまで、非アクティブになる共有サーバーにはPMONによって終了のマークが付けられます。

- 動的に共有サーバーを設定するには、ALTER SYSTEM文でSHARED_SERVERS初期化パラメータを0以外の値に設定します。

たとえば、次の文は、共有サーバーの数を削減します。

```
ALTER SYSTEM SET SHARED_SERVERS = 5;
```

SHARED_SERVERSを0(ゼロ)に設定すると、共有サーバーは使用禁止になります。詳細は、「共有サーバーの無効化」を参照してください。

親トピック: 共有サーバーの使用可能化

5.4.3.3 共有サーバー・プロセス数の制限

MAX_SHARED_SERVERS初期化パラメータは、PMONが自動的に作成可能な共有サーバーの最大数を指定します。デフォルト値はありません。

値が指定されていない場合、PMONは次の制限内で、負荷によって必要とされる共有サーバーを可能なかぎり起動します。

- プロセスの制限(PROCESSES初期化パラメータによって設定)
- 最小空きプロセス・スロット数(少なくとも総プロセス・スロット数の1/8、またはPROCESSESが24未満に設定されている場合は2スロット)
- システム・リソース

共有サーバーのプロセス数を制限するには:

- MAX_SHARED_SERVERS初期化パラメータを設定します。

SHARED_SERVERSの値が、MAX_SHARED_SERVERSの値より優先されます。したがって、SHARED_SERVERSをMAX_SHARED_SERVERSより大きい値に設定すると、MAX_SHARED_SERVERSの値を超える数の共有サーバーをPMONに起動させることができます。その後は、MAX_SHARED_SERVERSの値をSHARED_SERVERSより大きい値に動的に変更することで、共有サーバー数に新しい上限を設定できます。

共有サーバー数を制限する主な理由は、メモリーやCPUタイムなどのリソースを他のプロセス用に確保しておくためです。たとえば、前述のテレマーケティング・センターの例について考えてみます。

DBAは、夜間のバッチ・ジョブ用にリソースの2/3を確保しようとします。DBAは、MAX_SHARED_SERVERSの値を最大プロセス数(PROCESSES)の1/3より小さい値に設定します。この設定によって、DBAは、すべての従業員がデータベースに同時にアクセスした場合でも、バッチ・ジョブは専用サーバーに接続でき、従業員の要求処理後に共有サーバーが停止するまで待機する必要がないことを確認します。

共有サーバー数を制限するもう1つの理由は、多数のサーバー・プロセスが同時に実行されることによって、大量のスワッピングが発生してシステムの処理速度が低下しないようにするためです。ただし、この場合はMAX_SHARED_SERVERSよりPROCESSESが上限として機能します。

この他にも、共有サーバー数を制限する理由として、テスト、デバッグ、パフォーマンス分析およびチューニングがあります。たとえば、特定のユーザー・コミュニティを効率的にサポートするために必要な共有サーバーの数を調べるために、MAX_SHARED_SERVERSの値を非常に小さい数に設定しておき、ユーザーが応答時間の遅延を認識しない数まで増やすことができます。

親トピック: 共有サーバーの使用可能化

5.4.3.4 共有サーバー・セッション数の制限

SHARED_SERVER_SESSIONS初期化パラメータは、同時共有サーバー・ユーザー・セッションの最大数を指定します。

動的パラメータであるこのパラメータを設定すると、データベース・セッションを専用サーバー用に確保できます。この結果、データベースのバックアップやリカバリなど、専用サーバーを必要とする管理タスクが共有サーバー・セッションによって専有されることはありません。

共有サーバーのセッション数を制限するには:

- SHARED_SERVER_SESSIONS初期化パラメータを設定します。

このパラメータにデフォルト値はありません。値が指定されていない場合は、SESSIONS初期化パラメータの制限範囲内で共有サーバー・セッションが必要に応じて作成されます。

親トピック: 共有サーバーの使用可能化

5.4.3.5 共有メモリーの保護

CIRCUITS初期化パラメータは、共有メモリーに作成可能なバーチャル・サーキットの最大数を設定します。このパラメータにデフォルトはありません。値が指定されていない場合は、DISPATCHERS初期化パラメータおよびシステム・リソースの制限範囲内でサーキットが必要に応じて作成されます。

共有メモリーに作成可能なバーチャル・サーキット数を制限することにより共有メモリーを保護するには:

- CIRCUITS初期化パラメータを設定します。

親トピック: 共有サーバーの使用可能化

5.4.4 ディスパッチャの構成

DISPATCHERS初期化パラメータは、共有サーバー・アーキテクチャ内のディスパッチャ・プロセスを構成します。共有サーバーが動作するためには、1つのディスパッチャ・プロセスが最小限必要です。ディスパッチャを指定せずに、SHARED_SERVERを0(ゼロ)以外の値に設定して共有サーバーを使用可能にすると、デフォルトで、Oracle Databaseによって1つのディスパッチャがTCPプロトコル用に作成されます。

この構成に相当する初期化パラメータDISPATCHERSの明示的設定は次のとおりです。

```
dispatchers="(PROTOCOL=tcp)"
```

次のいずれかの条件に適合する場合は、DISPATCHERS初期化パラメータを使用して、追加のディスパッチャを構成できます。

- TCP/IP以外のプロトコルを構成する必要がある場合。DISPATCHERSパラメータの次のいずれかの属性でプロトコル・アドレスを構成します。
    - ADDRESS
    - DESCRIPTION
    - PROTOCOL
- 次のオプションのディスパッチャ属性を1つ以上構成する場合。
    - DISPATCHERS
    - CONNECTIONS

- SESSIONS
- LISTENER
- MULTIPLEX
- SERVICE

ノート:

このパラメータの構成には、Database Configuration Assistant が役立ちます。

TCP/IP以外のプロトコルを構成するか、追加のディスパッチャを構成するには:

- DISPATCHERS初期化パラメータを設定し、適切な属性を指定します。

- DISPATCHERS初期化パラメータの属性
  DISPATCHERS初期化パラメータには、複数の属性を設定できます。
- ディスパッチャ数の決定
  オペレーティング・システムに対して各プロセスごとに可能な接続数を把握した後、インスタンス起動時に各ネットワーク・プロトコルに対して作成する初期ディスパッチャの数を計算します。
- 初期ディスパッチャ数の設定
  複数のディスパッチャ構成を指定するには、DISPATCHERSにカンマ区切りの文字列リストを設定するか、初期化パラメータ・ファイルに複数のDISPATCHERS初期化パラメータを指定します。
- ディスパッチャ数の変更
  インスタンスのディスパッチャ・プロセス数を制御することができます。共有サーバー数と異なり、ディスパッチャ数は自動的に変更されません。ディスパッチャの数は、明示的にALTER SYSTEM文で変更します。ディスパッチャ数は、MAX_DISPATCHERSパラメータで指定された制限を超えて増加することができます。
- 特定のディスパッチャ・プロセスの停止
  ALTER SYSTEM SET DISPATCHERS文では、ディスパッチャ数の削減のために停止されるディスパッチャは、データベースによって決定されます。別の方法として、特定のディスパッチャ・プロセスを停止できます。

親トピック: Oracle Databaseの共有サーバー構成

5.4.4.1 DISPATCHERS初期化パラメータの属性

DISPATCHERS初期化パラメータには、複数の属性を設定できます。

プロトコル・アドレスは必須です。プロトコル・アドレスは次の属性を1つ以上使用して指定します。

属性	説明
ADDRESS	ディスパッチャがリスニングするエンドポイントのネットワーク・プロトコル・アドレスを指定します。
DESCRIPTION	ネットワーク・プロトコル・アドレスなど、ディスパッチャがリスニングするエンドポイントのネットワークの説明を指定します。構文は次のとおりです。 (DESCRIPTION=(ADDRESS=...))

属性	説明
PROTOCOL	ディスパッチャによってリスニング・エンドポイントが生成されるネットワーク・プロトコルを指定します。たとえば: (PROTOCOL=tcp) プロトコル・アドレスの構文の詳細は、『Oracle Database Net Services リファレンス』を参照してください。

次の属性は、この構成が使用するディスパッチャ数を指定します。これはオプションで、デフォルトは1です。

属性	説明
DISPATCHERS	起動する初期ディスパッチャの数を指定します。

次の属性は、構成の各ディスパッチャのネットワーク属性に関する情報をインスタンスに通知します。これらの属性はすべてオプションです。

属性	説明
CONNECTIONS	各ディスパッチャに対して許容されるネットワーク接続の最大数を指定します。
SESSIONS	各ディスパッチャに対して許容されるネットワーク・セッションの最大数を指定します。
LISTENER	LREG プロセスがディスパッチャ情報を登録するリスナーの別名を指定します。ネーミング・メソッドによって解決される別名を設定してください。
MULTIPLEX	Oracle Connection Manager のセッションの多重化機能を使用可能にする場合に使用します。
SERVICE	ディスパッチャがリスナーに登録するサービス名を指定します。

完全な属性名または先頭の3文字以上で構成されたサブストリングを指定できます。たとえば、SESSIONS=3、SES=3、SESS=3またはSESSI=3のように指定できます。

関連項目:

DISPATCHERS初期化パラメータの属性の詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: ディスパッチャの構成

5.4.4.2 ディスパッチャ数の決定

オペレーティング・システムに対して各プロセスごとに可能な接続数を把握した後、インスタンス起動時に各ネットワーク・プロトコルに対して作成する初期ディスパッチャの数を計算します。

インスタンスの起動時に作成される初期のディスパッチャ数を計算するには、次の式を使用します。

```
Number of dispatchers =
   CEIL ( max. concurrent sessions / connections for each dispatcher )
```

CEILは、最も近い整数に切り上げた結果を返します。

たとえば、各プロセスごとに970の接続数をサポートできるシステムがある場合に、次の最大セッション数を仮定します。

- TCP/IPを介して最大4,000のセッションが同時に接続します。
- SSL付きTCP/IPを介して最大2,500のセッションが同時に接続します。

この場合は、次のように、TCP/IP用のDISPATCHERS属性には最低で5ディスパッチャ(4000÷970)を、SSL付きTCP/IP用には最低で3ディスパッチャ(2500÷970)を設定する必要があります。

```
DISPATCHERS='(PROT=tcp)(DISP=5)', '(PROT=tcps)(DISP=3)'
```

パフォーマンスによっては、ディスパッチャ数の調整が必要になる場合があります。

親トピック: ディスパッチャの構成

5.4.4.3 初期ディスパッチャ数の設定

複数のディスパッチャ構成を指定するには、DISPATCHERSにカンマ区切りの文字列リストを設定するか、初期化パラメータ・ファイルに複数のDISPATCHERS初期化パラメータを指定します。

- DISPATCHERS初期化パラメータを設定します。

DISPATCHERSを複数回指定する場合、各行は初期化パラメータ・ファイル内で互いに隣接している必要があります。内部では、Oracle Databaseによって、INDEX値(0(ゼロ)から開始)が各DISPATCHERSパラメータに割り当てられます。このDISPATCHERSパラメータは、後で、ALTER SYSTEM文の中で索引番号を使用して参照できます。

次に、DISPATCHERS初期化パラメータの設定例をいくつか示します。

例: 標準

これは、DISPATCHERS初期化パラメータの設定の標準的な例です。

```
DISPATCHERS="(PROTOCOL=TCP)(DISPATCHERS=2)"
```

例: ディスパッチャに使用するIPアドレスの強制

次の仮定的な例では、指定されたIPアドレスでリッスンする2つのディスパッチャを作成します。アドレスは、インスタンスが存在するホストの有効なIPアドレスである必要があります。(ホストに複数のIPアドレスを構成できます。)

```
DISPATCHERS="(ADDRESS=(PROTOCOL=TCP)(HOST=144.25.16.201))(DISPATCHERS=2)"
```

例: ディスパッチャが使用するポートの強制

ディスパッチャが特定のポートをリスニング・エンドポイントとして使用するように強制するには、次のようにPORT属性を追加します。

```
DISPATCHERS="(ADDRESS=(PROTOCOL=TCP)(PORT=5000))"
DISPATCHERS="(ADDRESS=(PROTOCOL=TCP)(PORT=5001))"
```

親トピック: ディスパッチャの構成

5.4.4.4 ディスパッチャ数の変更

インスタンス内のディスパッチャ・プロセスの数を制御できます。共有サーバー数と異なり、ディスパッチャ数は自動的に変更されま

せん。ディスパッチャの数は、明示的にALTER SYSTEM文で変更します。ディスパッチャ数は、MAX_DISPATCHERSパラメータで指定された制限を超えて増加することができます。

1. 次のビューを監視し、ディスパッチャ・プロセスの負荷を判別します。
   - V$QUEUE
   - V$DISPATCHER
   - V$DISPATCHER_RATE

   これらのビューによって、ディスパッチャ・プロセスに対する負荷が一貫して高いことが判明した場合は、追加のディスパッチャ・プロセスを起動してユーザー要求をルーティングすることで、パフォーマンスを改善できます。逆に、ディスパッチャの負荷が一貫して低い場合は、ディスパッチャの数を少なくすることにより、パフォーマンスを改善できます。

   関連項目:

   これらのビューを監視してディスパッチャの負荷とパフォーマンスを判別する方法については、『Oracle Databaseパフォーマンス・チューニング・ガイド』
2. インスタンスの実行中にディスパッチャの数を動的に変更するには、ALTER SYSTEM文を使用して、既存のディスパッチャ構成に対するDISPATCHERS属性の設定を変更します。新規のディスパッチャ構成を追加して、異なるネットワーク属性でディスパッチャを起動することもできます。

特定のディスパッチャ構成に対するディスパッチャ数を少なくしても、ディスパッチャが即時に削除されるわけではありません。Oracle Databaseは、ユーザーの切断にあわせてDISPATCHERSで指定した制限数に達するまでディスパッチャを停止します。

たとえば、初期化パラメータ・ファイルの次のDISPATCHERS設定でインスタンスが起動されたとします。

```
DISPATCHERS='(PROT=tcp)(DISP=2)', '(PROT=tcps)(DISP=2)'
```

TCP/IPプロトコル用のディスパッチャ数を2から3に増やし、SSL付きTCP/IPプロトコル用のディスパッチャ数を2から1に減らすには、次の文を発行します。

```
ALTER SYSTEM SET DISPATCHERS = '(INDEX=0)(DISP=3)', '(INDEX=1)(DISP=1)';
```

または

```
ALTER SYSTEM SET DISPATCHERS = '(PROT=tcp)(DISP=3)', '(PROT=tcps)(DISP=1)';
```

ノート:

(DISP=1)を指定する必要はありません。1 は DISPATCHERS パラメータのデフォルト値のため、これはオプションです。

現在起動しているTCP/IP用のディスパッチャ・プロセスが2以下の場合は、新しいプロセスが作成されます。SSL付きTCP/IP用のディスパッチャ・プロセスが現在複数起動している場合は、接続ユーザーの切断にあわせて余分なプロセスが停止されます。

- ディスパッチャの変更のノート
  ディスパッチャの変更の詳細を理解します。

親トピック: ディスパッチャの構成

5.4.4.4.1 ディスパッチャ数変更時のノート

ディスパッチャの変更の詳細を理解します。

- INDEXキーワードを使用すると、変更するディスパッチャ構成を識別できます。このINDEXを指定しないと、指定したDESCRIPTION、ADDRESSまたはPROTOCOLと一致する最初のディスパッチャ構成が変更されます。既存のディスパッチャ構成の中で一致するものが見つからなかった場合は、新しいディスパッチャが追加されます。
- INDEX値の範囲は0からn-1で、nは現行のディスパッチャ構成の数です。ALTER SYSTEM文でINDEX値にnを指定すると(nは現行のディスパッチャ構成の数)、新しいディスパッチャ構成が追加されます。
- 現行のディスパッチャ構成の値、つまり、ディスパッチャの数などを確認するには、V$DISPATCHER_CONFIG動的パフォーマンス・ビューを問い合せます。ディスパッチャが関連付けられているディスパッチャ構成を確認するには、V$DISPATCHERビューのCONF_INDX列を問い合せます。
- ディスパッチャ構成のDESCRIPTION、ADDRESS、PROTOCOL、CONNECTIONSおよびMULTIPLEX属性を変更しても、その変更は既存のディスパッチャには反映されず、新しいディスパッチャにのみ反映されます。したがって、構成に関連付けられているすべてのディスパッチャに対して変更を反映するには、DISPATCHERSパラメータを変更した後で、既存のディスパッチャを強制的に停止し、新しく指定したプロパティに配置された新しいディスパッチャをデータベースで起動する必要があります。

  LISTENER属性とSERVICES属性は、同じ制約の対象となりません。これらの属性は、変更した構成に関連付けられている既存のディスパッチャに適用されます。SESSIONS属性は、その値が減らされた場合のみ、既存のディスパッチャに適用されます。反対に値が増加された場合は、新しく起動されるディスパッチャにのみ適用されます。

親トピック: ディスパッチャ数の変更

5.4.4.5 特定のディスパッチャ・プロセスの停止

ALTER SYSTEM SET DISPATCHERS文では、ディスパッチャ数の削減のために停止されるディスパッチャは、データベースによって決定されます。別の方法として、特定のディスパッチャ・プロセスを停止できます。

1. 停止するディスパッチャ・プロセスの名前を識別するには、V$DISPATCHER動的パフォーマンス・ビューを使用します。

   ```
   SELECT NAME, NETWORK FROM V$DISPATCHER;
   ```

   各ディスパッチャは、Dnnn形式の名前で一意に識別されます。
2. ALTER SYSTEM SHUTDOWN IMMEDIATE文を実行し、ディスパッチャの名前を指定します。

   たとえば、ディスパッチャD002を停止するには、次の文を発行します。

   ```
   ALTER SYSTEM SHUTDOWN IMMEDIATE 'D002';
   ```

   IMMEDIATEキーワードを指定すると、ディスパッチャによる新規接続の受入れが停止し、データベースはそのディスパッチャを介した既存のすべての接続を即時に終了します。すべてのセッションがクリーン・アップされてから、ディスパッチャ・プロセスが停止します。IMMEDIATEを指定しなかった場合、ディスパッチャは、接続ユーザーがすべて切断され、接続がすべて終了するまで待ってから停止します。

親トピック: ディスパッチャの構成

5.4.5 共有サーバーの使用禁止

共有サーバーを無効にするには、SHARED_SERVERSを0に設定します。この操作は、ALTER SYSTEM文を使用して動的に行うことができます。

- SHARED_SERVERS初期化パラメータを0に設定します。

共有サーバーを無効にすると、新しいクライアントは共有モードで接続できなくなります。ただし、Oracle Databaseではすべての共有サーバーの接続がクローズされるまで一部の共有サーバーが保持されます。保持される共有サーバーの数は、SHARED_SERVERSの変更前の設定で指定されていた数、またはMAX_SHARED_SERVERSパラメータの値のいずれか小さい値です。SHARED_SERVERSとMAX_SHARED_SERVERSの両方が0(ゼロ)に設定されている場合は、すべての共有サーバーが停止し、残りの共有サーバー・クライアントからの要求は、SHARED_SERVERSまたはMAX_SHARED_SERVERSの値が再度引き上げられるまでキューで待機します。

ディスパッチャを一度停止して、すべての共有サーバー・クライアントを切断するには、次の文を入力します。

```
ALTER SYSTEM SET DISPATCHERS = '';
```

親トピック: Oracle Databaseの共有サーバー構成

5.4.6 共有サーバーのデータ・ディクショナリ・ビュー

共有サーバーの構成情報およびパフォーマンスの監視のためにデータ・ディクショナリ・ビューを問い合せることができます。

ビュー	説明
V$DISPATCHER	名前、ネットワーク・アドレス、状態、各種使用統計、索引番号などのディスパッチャ・プロセス情報を提供します。
V$DISPATCHER_CONFIG	ディスパッチャに関する構成情報を提供します。
V$DISPATCHER_RATE	ディスパッチャ・プロセスのレート統計が含まれています。
V$QUEUE	共有サーバー・メッセージ・キューについての情報が含まれています。
V$SHARED_SERVER	共有サーバーについての情報が含まれています。
V$CIRCUIT	バーチャル・サーキットについての情報が含まれています。バーチャル・サーキットとは、ディスパッチャおよびサーバーを介したデータベースへのユーザー接続です。
V$SHARED_SERVER_MONITOR	共有サーバーのチューニング情報が含まれています。
V$SGA	各種のシステム・グローバル領域(SGA)グループに関するサイズ情報が含まれています。この情報は、共有サーバーのチューニング時に役立ちます。
V$SGASTAT	チューニングに役立つ SGA 関連の詳細な統計情報が含まれています。
V$SHARED_POOL_RESERVED	共有プール内で予約済のプールと領域をチューニングする際に役立つ統計リストが含まれています。

関連項目:

共有サーバーの監視とチューニングの詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』

親トピック: Oracle Databaseの共有サーバー構成

5.5 データベース常駐接続プーリングの構成

データベース・サーバーは、データベース常駐接続プーリングを使用できるように事前に構成されています。ただし、この機能は、接続プールを起動して明示的に使用可能にする必要があります。

- データベース常駐接続プーリングの初期化パラメータ
  データベース常駐接続プーリングを構成するには、初期化パラメータを設定します。
- データベース常駐接続プーリングを使用可能にする方法
  Oracle Databaseには、SYS_DEFAULT_CONNECTION_POOLと呼ばれるデフォルトの接続プールがあります。デフォルトでは、このプールは作成されますが起動はされません。データベース常駐接続プーリングを使用可能にするには、接続プールを明示的に起動する必要があります。
- データベース常駐接続プーリングの接続プールの構成
  接続プールは、デフォルトのパラメータ値を使用して構成します。DBMS_CONNECTION_POOLパッケージ内のプロシージャを使用すると、使用方法に応じて接続プールを構成できます。Oracle Real Application Clusters(Oracle RAC)環境では、構成パラメータは各Oracle RACインスタンスに適用されます。
- データベース常駐接続プーリングのデータ・ディクショナリ・ビュー
  接続プールの情報の取得およびデータベース常駐接続プーリングのパフォーマンスの監視のために、データ・ディクショナリ・ビューを問い合せることができます。
- 接続プールの接続の状態の調査
  接続プールの各接続の現在の状態を調べるために、V$CPOOL_CONN_INFOビューを問い合せることができます。

関連項目:

「データベース常駐接続プーリングの理解」

親トピック: プロセスの管理

5.5.1 データベース常駐接続プーリングの初期化パラメータ

データベース常駐接続プーリングを構成するには、初期化パラメータを設定します。

データベース常駐接続プーリング(DRCP)に専用最適化の使用を構成するには、DRCP_DEDICATED_OPT初期化パラメータを使用します。専用最適化を有効にするには、DRCP_DEDICATED_OPTにYesを設定します。専用最適化では、DRCPブローカへの接続数がDRCP最大サイズよりも少ない場合、DRCPは専用サーバーと同様に動作します。

認証プールを構成するには、次の初期化パラメータを使用します。

- MAX_AUTH_SERVERS

  認証プール内の認証サーバーの最大数を指定します。認証プールは、接続プールとは分離されており、クライアント・アプリケーションがDRCPに接続するときにユーザー接続を認証します。このパラメータには、MIN_AUTH_SERVERS初期化パラメータに指定した値より大きい正の整数を設定します。

- MIN_AUTH_SERVERS

  認証プール内の認証サーバーの最小数を指定します。このパラメータには、MAX_AUTH_SERVERS初期化パラメータに指定した値より小さい正の整数を設定します。

親トピック: データベース常駐接続プーリングの構成

5.5.2 データベース常駐接続プーリングを使用可能にする方法

Oracle Databaseには、SYS_DEFAULT_CONNECTION_POOLと呼ばれるデフォルトの接続プールがあります。デフォルトでは、このプールは作成されますが起動はされません。データベース常駐接続プーリングを使用可能にするには、接続プールを明示的に起動する必要があります。

データベース常駐接続プーリングを使用可能にするには:

1. 「データベース常駐接続プールの起動」の説明に従って、データベース常駐接続プールを起動します。
2. 「接続プールへのクライアント接続要求のルーティング」の説明に従って、クライアント接続要求を接続プールにルーティングします。

データベース常駐接続プールの起動

接続プールを起動するには:

1. SQL*Plusを起動して、SYSユーザーとしてデータベースに接続します。
2. 次のコマンドを発行します。

```
SQL> EXECUTE DBMS_CONNECTION_POOL.START_POOL();
```

起動されると、接続プールは明示的に停止されるまでこの状態のままです。接続プールは、データベース・インスタンスが再起動されると、インスタンスの停止時点でアクティブだった場合は自動的に再起動されます。

Oracle Real Application Clusters(Oracle RAC)環境では、インスタンスを使用して接続プールを管理できます。プール構成に対する変更は、すべてのOracle RACインスタンスで適用されます。

接続プールへのクライアント接続要求のルーティング

クライアント・アプリケーションでは、接続文字列で接続タイプをPOOLEDに指定する必要があります。

次の例は、クライアントをデータベース常駐接続プールに接続する簡単な接続文字列を示しています。

```
examplehost.company.com:1521/books.company.com:POOLED
```

次の例は、クライアントをデータベース常駐接続プールに接続するTNS接続記述子を示しています。

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=myhost)
       (PORT=1521))(CONNECT_DATA=(SERVICE_NAME=sales)
       (SERVER=POOLED)))
```

ノート:

データベース常駐接続プールへのクライアント接続では、TCP プロトコルのみがサポートされます。

データベース常駐接続プーリングを使用禁止にする方法

データベース常駐接続プーリングを使用禁止にするには、接続プールを明示的に停止する必要があります。次のステップを実行します。

1. SQL*Plusを起動して、SYSユーザーとしてデータベースに接続します。
2. 次のコマンドを発行します。

```
SQL> EXECUTE DBMS_CONNECTION_POOL.STOP_POOL();
```

関連項目:

DBMS_CONNECTION_POOLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ノート:

データベース常駐接続プールを使用禁止にする操作は、サーバーに渡されたすべてのクライアント要求が完了しないと完了できません。

親トピック: データベース常駐接続プーリングの構成

5.5.3 データベース常駐接続プーリングの接続プールの構成

接続プールはデフォルトのパラメータ値を使用して構成されます。DBMS_CONNECTION_POOLパッケージ内のプロシージャを使用すると、使用方法に応じて接続プールを構成できます。Oracle Real Application Clusters(Oracle RAC)環境では、構成パラメータは各Oracle RACインスタンスに適用されます。

CONFIGURE_POOLプロシージャの使用

DBMS_CONNECTION_POOLパッケージのCONFIGURE_POOLプロシージャを使用すると、拡張オプションを指定して接続プールを構成できます。このプロシージャは通常、接続プールのすべてのパラメータを変更する必要がある場合に使用します。

ALTER_PARAMプロシージャ

DBMS_CONNECTION_POOLパッケージのALTER_PARAMプロシージャを使用すると、特定の構成パラメータを他のパラメータに影響を与えることなく変更できます。たとえば、次のコマンドでは、使用されるプール・サーバーの最小数が変更されます。

```
SQL> EXECUTE DBMS_CONNECTION_POOL.ALTER_PARAM ('','MINSIZE','10');
```

次の例では、各接続ブローカで処理できる最大接続数が50000に変更されます。

```
SQL> EXECUTE DBMS_CONNECTION_POOL.ALTER_PARAM ('','MAXCONN_CBROK','50000');
```

このコマンドを実行する前に、データベースがインストールされているプラットフォームで許可される最大接続数がMAXCONN_CBROKに設定した値未満でないことを確認します。

たとえば、Linuxでは、/etc/security/limits.confファイルに次のエントリがあると、ユーザーtest_userに許可される最大接続数は30000となります。

```
test_user HARD NOFILE 30000
```

各接続ブローカで処理できる最大接続数を50000に設定するには、最初にlimits.confファイルの値を50000以上の値に変更します。

接続プールのデフォルト設定のリストア

接続プールのパラメータを変更した後でデフォルトのプール設定に戻す場合は、DBMS_CONNECTION_POOLパッケージの

RESTORE_DEFAULTプロシージャを使用します。接続プールの設定をデフォルトに戻すコマンドは、次のとおりです。

```
SQL> EXECUTE DBMS_CONNECTION_POOL.RESTORE_DEFAULTS();
```

- データベース常駐接続プーリングの構成パラメータ
  DBMS_CONNECTION_POOLパッケージ内のサブプログラムのパラメータを指定して、データベース常駐接続プーリングを構成できます。

関連項目:

DBMS_CONNECTION_POOLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: データベース常駐接続プーリングの構成

5.5.3.1 データベース常駐接続プーリングの構成パラメータ

DBMS_CONNECTION_POOLパッケージ内のサブプログラムのパラメータを指定して、データベース常駐接続プーリングを構成できます。

次の表は、接続プールのために設定できるパラメータを示しています。

表5-2 データベース常駐接続プーリングの構成パラメータ

パラメータ名	説明
MINSIZE	プール内のプール・サーバーの最小数。デフォルト値は 4 です。
MAXSIZE	プール内のプール・サーバーの最大数。デフォルト値は 40 です。 接続プールでは、プールされたサーバーの 5%が認証のために保持され、少なくとも 1 つのプールされたサーバーが認証のために常時保持されます。このパラメータを設定する場合は、認証と接続の両方に十分なプール・サーバーがあることを確認します。
INCRSIZE	クライアント・アプリケーション要求の受取り時にサーバーが使用可能でない場合にプールが増分されるプール・サーバーの数。デフォルト値は 2 です。
SESSION_CACHED_CURSORS	各プール・サーバー・セッションでキャッシュするセッション・カーソルの数。デフォルト値は 20 です。
INACTIVITY_TIMEOUT	プール・サーバーがプールでアイドル状態のままで待機する最大時間(秒数)。この時間を過ぎると、サーバーは終了される。デフォルト値は 300 です。 このパラメータはプールが MINSIZE の場合は適用されません。
MAX_THINK_TIME	クライアントがプールからプール・サーバーを取得した後で非アクティブ状態でいる最大時間(秒数)。クライアント・アプリケーションがプールからプール・サーバーを取

パラメータ名	説明
	得した後、MAX_THINK_TIME で指定した時間内にデータベース・コールを発行しない場合、プール・サーバーは解放されてクライアント接続が終了します。そのため、このような接続でラウンドトリップ・コールが試行されると、アプリケーションで ORA-3113 または ORA-3115 エラーが発生する可能性があります。
MAX_TXN_THINK_TIME	プールからオープン・トランザクションを含むプール・サーバーを取得した後のクライアントの最大非アクティブ時間(秒数)。クライアント・アプリケーションがプールからプール・サーバーを取得した後、MAX_TXN_THINK_TIME で指定した時間内にデータベース・コールを発行しない場合、プール・サーバーは解放されてクライアント接続が終了します。このパラメータのデフォルト値は、MAX_THINK_TIME パラメータの値です。オープン・トランザクションを含む接続のための時間を確保するために、MAX_TXN_THINK_TIME パラメータの値は、MAX_THINK_TIME 値よりも高い値に設定できます。
MAX_USE_SESSION	プール・サーバーがプールから取得されプールに解放される回数。デフォルト値は 500000 です。
MAX_LIFETIME_SESSION	プール・サーバーがプールに存在している時間(秒数)。デフォルト値は 86400 です。
NUM_CBROK	クライアント要求を処理するために作成される接続ブローカの数。デフォルト値は 1 です。 複数の接続ブローカ・プロセスを作成すると、大量のクライアント・アプリケーションがある場合に、クライアント接続要求の負荷を分散できます。
MAXCONN_CBROK	各接続ブローカで処理できる最大接続数。 デフォルト値は 40000 です。ただし、データベースがインストールされているプラットフォームで許可される最大接続がデフォルト値より少ない場合は、MAXCONN_CBROK を使用して設定した値はその値で上書きされます。 MAXCONN_CBROK で指定した接続数がサポートされるように、オペレーティング・システムのプロセスごとのファイル記述子に関する制限は十分大きい値に設定してください。

関連項目:

DBMS_CONNECTION_POOLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: データベース常駐接続プーリングの接続プールの構成

5.5.4 データベース常駐接続プーリングのデータ・ディクショナリ・ビュー

接続プールの情報の取得およびデータベース常駐接続プーリングのパフォーマンスの監視のために、データ・ディクショナリ・ビューを問い合せることができます。

表5-3 データベース常駐接続プーリングのデータ・ディクショナリ・ビュー

ビュー	説明
DBA_CPOOL_INFO	プール・ステータス、接続の最大数と最小数、アイドル・セッションのタイムアウトなど、接続プールに関する情報が含まれます。
V$CPOOL_CONN_INFO	接続ブローカへの各接続に関する情報が含まれます。
V$CPOOL_STATS	セッション要求の数、要求と一致するセッションがプールで検出された回数、セッション要求の合計待機時間など、プール統計が含まれます。
V$CPOOL_CC_INFO	プールと接続のクラス・マッピングに関する情報が含まれています。
V$CPOOL_CC_STATS	プールの接続クラス・レベルの統計が含まれます。

親トピック: データベース常駐接続プーリングの構成

5.5.5 接続プールの接続状態の判別

接続プールの各接続の現在の状態を調べるために、V$CPOOL_CONN_INFOビューを問い合せることができます。

このビューの問合せにより、各接続の状態についての詳細を得ることができます。たとえば、ビジーまたはアイドル状態の接続を判別できます。この情報を判別するには:

- V$CPOOL_CONN_INFOビューを問い合せます。

例5-1 接続の待ち時間の判別

次の問合せは、WAITING状態の接続の待ち時間を示します。

```
SELECT USERNAME, SERVICE, LAST_WAIT_TIME
   FROM V$CPOOL_CONN_INFO
   WHERE CONNECTION_STATUS = 'WAITING';
```

例5-2 接続がアクティブになってからの経過時間の判別

次の問合せは、ACTIVE状態の接続について、各接続がアクティブになってからの経過時間を示します。

```
SELECT USERNAME, SERVICE, LAST_ACTIVE_TIME
   FROM V$CPOOL_CONN_INFO
   WHERE CONNECTION_STATUS = 'ACTIVE';
```

例5-3 最長のアクティブ状態を持つ実行中の接続のリスト

次の問合せは、ACTIVE状態が最長時間となっている接続を示します。

```
SELECT USERNAME, SERVICE, ACTIVE_TIME
   FROM V$CPOOL_CONN_INFO
   WHERE CONNECTION_STATUS = 'ACTIVE'
   ORDER BY ACTIVE_TIME DESC;
```

例5-4 待機キュー内の最も古い接続の待ち時間の判別

次の問合せは、WAITING状態のセッションの待ち時間を示します。

```
SELECT USERNAME, SERVICE, LAST_WAIT_TIME
   FROM V$CPOOL_CONN_INFO
   WHERE LAST_WAIT_TIME = (
      SELECT max(LAST_WAIT_TIME)
         FROM V$CPOOL_CONN_INFO
         WHERE CONNECTION_STATUS = 'WAITING');
```

親トピック: データベース常駐接続プーリングの構成

5.6 Oracle Databaseバックグラウンド・プロセスについて

マルチプロセスのOracle Databaseシステムでは、最高のパフォーマンスを提供し、多くのユーザーが同時に使用できるように、バックグラウンド・プロセスが使用されています。バックグラウンド・プロセスでは、ユーザー・プロセスごとに実行される複数のデータベース・プログラムによって処理される機能が統合されます。バックグラウンド・プロセスは、非同期的にI/Oを実行して他のOracle Databaseプロセスを監視することによって、並列性を高め、パフォーマンスと信頼性を向上させます。

表5-4に基礎的なバックグラウンド・プロセスを示しますが、これらの多くはこのドキュメントの他の項で詳細に説明されています。追加のデータベース機能やオプションを使用している場合、これ以外のバックグラウンド・プロセスも実行される場合があります。たとえば:

- Oracle Databaseアドバンスト・キューイングを使用している場合は、キュー・モニター(QMNn)・バックグラウンド・プロセスが存在します。
- データファイルをストレージ・サブシステムの物理デバイスにマッピングするためにFILE_MAPPING初期化パラメータをtrueに設定している場合は、FMONプロセスが存在します。
- Oracle Automatic Storage Management(Oracle ASM)を使用している場合は、Oracle ASM固有の追加のバックグラウンド・プロセスが存在します。

表5-4 Oracle Databaseバックグラウンド・プロセス

プロセス名	説明
データベース・ライター(DBWnまたは BWnn)	データベース・ライターは、変更があったブロックをデータベース・バッファ・キャッシュからデータファイルに書き込みます。Oracle Database は最高で 100 のデータベース・ライター・プロセスを行います。最初の 36 個のデータベース・ライター・プロセスの名前は、DBW0-DBW9 および DBWa-DBWz です。37 番目から 100 番目のデータベース・ライター・プロセスの名前は、BW36-BW99 です。 データベース・ライター・プロセスの数は、DB_WRITER_PROCESSES 初期化パラメータで指定します。データベースは、CPU 数とプロセッサ・グループ数に基づいて、この初期化パラメータに適切なデフォルト設定を選択またはユーザー指定の設定を調整します。 DB_WRITER_PROCESSES 初期化パラメータの設定の詳細は、『Oracle Database パ

プロセス名	説明
	フォーマンス・チューニング・ガイド』を参照してください。
ログ・ライター(LGWR)	ログ・ライター・プロセスは、ディスクに REDO ログ・エントリを書き込みます。REDO ログ・エントリは、システム・グローバル領域(SGA)の REDO ログ・バッファに生成される。LGWR は、REDO ログ・ファイルに REDO ログ・エントリを順番に書き込む。データベースに多重化された REDO ログがある場合、LGWR は REDO ログ・ファイルのグループに REDO ログ・エントリを書き込む。ログ・ライター・プロセスについては、「REDO ログの管理」を参照してください。
チェックポイント(CKPT)	システム・グローバル領域内で変更があったすべてのデータベース・バッファは、特定の時点で DBWn によってデータファイルに書き込まれます。このイベントはチェックポイントと呼ばれます。チェックポイント・プロセスは、最新のチェックポイントを示すために、チェックポイントで DBWn にシグナルを出し、データベースのすべてのデータファイルと制御ファイルを更新する役割を持ちます。
システム・モニター(SMON)	システム・モニターは、障害の発生したインスタンスの再起動時にリカバリを実行します。Oracle Real Application Clusters データベースでは、1 つのインスタンスの SMON プロセスが、障害を起こした他のインスタンスのインスタンス・リカバリを実行できます。また、SMON により、不要になった一時セグメントがクリーン・アップされ、ファイル読込みエラーやオフライン・エラーのためにシステム障害時やインスタンス・リカバリ時にスキップされた停止後のトランザクションがリカバリされます。これらのトランザクションは、表領域またはファイルがオンラインに戻るときに、SMON によって最後にリカバリされます。
プロセス・モニター(PMON)	プロセス・モニターは、ユーザー・プロセスが失敗したときにプロセス・リカバリを実行します。PMON は、失敗したプロセスを検出する役割を担います。このため、PMON は、CLMN プロセスおよび CLnn スレーブによって実行されるクリーンアップを調整する役割を担います。クリーンアップにより、プロセスが使用していたリソースが開放されます。
アーカイバ(ARCn)	REDO ログ・ファイルは、ログ・ファイルが一杯になるか、ログ・スイッチが発生すると、1 つ以上のアーカイバ・プロセスによってアーカイブ記憶域にコピーされます。アーカイバ・プロセスは、「アーカイブ REDO ログ・ファイルの管理」の主題です。
リカバラ(RECO)	リカバラ・プロセスは、ネットワーク障害やシステム障害が原因で分散データベース内で保留されている分散トランザクションを解決するために使用されます。ローカル RECO は一定周期でリモート・データベースへの接続を試み、保留中の分散トランザクションのローカル部分のコミットまたはロールバックを自動的に完了させる。このプロセスの詳細と開始方法は、「分散トランザクションの管理」を参照してください。
ディスパッチャ(Dnnn)	ディスパッチャはオプションのバックグラウンド・プロセスで、共有サーバー構成の使用時にのみ存在します。共有サーバーの詳細は、前述の「Oracle Database の共有サーバー構成」を参

プロセス名	説明
	照してください。

関連項目:

Oracle Databaseバックグラウンド・プロセスの完全なリストは、『Oracle Databaseリファレンス』を参照してください。

親トピック: プロセスの管理

5.7 事前作成されたプロセスの管理

Oracle Databaseでは、プロセスを事前に作成してクライアント接続パフォーマンスを向上できます。

- 事前作成されたプロセスの管理について
  Oracle Databaseでは、プロセス・プールのフォアグラウンドおよびバックグラウンド・プロセスを事前作成できます。
- 事前作成されたプロセスのプールの管理
  DBMS_PROCESSパッケージを使用して、フォアグラウンド・プロセス・プールの事前作成プロセスの数を構成および変更できます。

親トピック: プロセスの管理

5.7.1 事前作成されたプロセスの管理について

Oracle Databaseは、プロセス・プールのフォアグラウンドおよびバックグラウンド・プロセスを事前作成できます。

専用ブローカが有効にされるか、スレッド実行モードが有効にされると、Oracle Databaseはフォアグラウンド・プロセスを事前作成します。フォアグラウンド・プロセスが必要となったときに、事前作成されたプロセスを内部的に使用して作成時間を短縮します。THREADED_EXECUTION初期化パラメータをTRUEに設定すると、データベースはスレッド実行モードで実行されます。このパラメータがFALSE(デフォルト)に設定されている場合、データベースはプロセス・モードで実行され、Oracle Databaseはプロセス・プールにフォアグラウンドおよびバックグラウンド・プロセスを事前作成しません。

プロセスが事前作成されている場合、クライアントの接続時間を効率化できます。スレッド実行モードが有効な場合、Oracle Databaseはデフォルトで様々なリクエスト・プールにプロセスを事前作成します。リクエスト・プールごとにプロセスの種類が異なります。V$PROCESS_POOLビューにこれらのプールの情報が表示され、DBMS_PROCESSパッケージを使用してこれらのプールを管理できます。

親トピック: 事前作成されたプロセスの管理

5.7.2 事前作成されたプロセスのプールの管理

DBMS_PROCESSパッケージを使用して、フォアグラウンド・プロセス・プールの事前作成プロセスの数を構成および変更できます。

Oracle Databaseは、クライアント接続の効率を改善するプロセス・プールを作成できます。これらのプールを管理するために、DBMS_PROCESSパッケージを使用できます。V$PROCESS_POOLビューの問合せにより、現在のプロセス・プールを表示できます。

プロセス・プールは、データベースがマルチスレッドOracle Databaseモデルで実行されている場合にのみ作成されます。

1. 必要な権限を持つユーザーとしてデータベースに接続します。

ユーザーにはSYSDBA管理権限が必要であり、接続時にAS SYSDBAを使用してこの権限を行使する必要があります。

2. プロセス・プールを管理するために、DBMS_PROCESSパッケージのサブプログラムを実行します。

例5-5 プロセス・プールの停止

この例では、SYS_DEFAULT_FOREGROUND_POOLプロセス・プールを停止します。

```
exec DBMS_PROCESS.STOP_POOL('SYS_DEFAULT_FOREGROUND_POOL');
```

プロセス・プールが停止しているとき、それに対するV$PROCESS_POOLビューのENABLED列はFALSEです。

例5-6 プロセス・プールの開始

この例では、SYS_DEFAULT_FOREGROUND_POOLプロセス・プールを開始します。

```
exec DBMS_PROCESS.START_POOL('SYS_DEFAULT_FOREGROUND_POOL');
```

プロセス・プールが有効であるとき、それに対するV$PROCESS_POOLビューのENABLED列はTRUEです。

例5-7 プロセス・プールの構成

V$PROCESS_POOLビューの問合せにより、プロセス・プールの現在の構成を確認できます。たとえば、次の問合せは、プロセス・プールの現在の構成を表示します。

```
COLUMN POOL_NAME FORMAT A30
COLUMN ENABLED FORMAT A7
COLUMN MIN_COUNT FORMAT 9999999
COLUMN BATCH_COUNT FORMAT 9999999
COLUMN INIT_COUNT FORMAT 9999999
SELECT POOL_NAME, ENABLED, MIN_COUNT, BATCH_COUNT, INIT_COUNT
   FROM V$PROCESS_POOL;
```

次のような結果を想定します。

```
POOL_NAME                      ENABLED MIN_COUNT BATCH_COUNT INIT_COUNT
------------------------------ ------- --------- ----------- ----------
SYS_DEFAULT_FOREGROUND_POOL    TRUE           10          20         29
```

このプロセス・プールで、事前作成される最小プロセス数を20、バッチで事前作成されるプロセスの数を30、および事前作成される初期のプロセス数を40に変更するには、次のプロシージャを実行します。

```
BEGIN
   DBMS_PROCESS.CONFIGURE_POOL(
      POOL_NAME   => 'SYS_DEFAULT_FOREGROUND_POOL',
      MIN_COUNT   => 20,
      BATCH_COUNT => 30,
      INIT_COUNT  => 40);
END;
/
```

問合せを再度実行することにより、変更を確認できます。

関連項目:

- THREADED_EXECUTION初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください
- DBMS_PROCESSパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

5.8 SQLのパラレル実行用プロセスの管理

SQL文のパラレル実行を管理できます。この構成では、Oracle Databaseによって、SQL文の処理作業を複数のパラレル・プロセスに分割できます。

ノート:

この項に記載されているパラレル実行機能は、Oracle Database Enterprise Edition で利用できます。

- パラレル実行サーバーについて
  多数のSQL文の実行をパラレル化できます。並列度は、単一の処理に対応付け可能なパラレル実行サーバーの数で表されます。
- セッションのパラレル実行の変更
  セッションのSQLのパラレル実行は、`ALTER SESSION`文を使用して制御できます。

5.8.1 パラレル実行サーバーについて

多数のSQL文の実行をパラレル化できます。並列度は、単一の処理に対応付け可能なパラレル実行サーバーの数で表されます。

並列度は、次の要素によって決まります。

- 文のPARALLEL句
- 問合せ内で参照されているオブジェクトの場合は、オブジェクトが作成または変更されたときに使用されたPARALLEL句
- 文に挿入されたパラレル・ヒント
- データベースによって決定されたデフォルト

SQLのパラレル実行の使用例は、「表作成のパラレル化」を参照してください。

インスタンスが起動すると、Oracle Databaseによって、パラレル操作に使用可能なパラレル実行サーバーのプールが作成されます。パラレル実行コーディネータと呼ばれるプロセスがパラレル実行サーバーのプールの実行をディスパッチし、これらすべてのパラレル実行サーバーからユーザーへの結果の送信を調整します。

PARALLEL_MAX_SERVERS初期化パラメータ値はデフォルトで0より大きい値に設定されているため、パラレル実行サーバーはデフォルトで使用可能です。これらのプロセスは、並列化を利用できる様々なOracle Databaseの機能によって使用されます。関連する初期化パラメータは、データベースによって大半のユーザー用にチューニングされますが、必要に応じて環境にあわせて変更できます。簡単にチューニングできるように、一部のパラメータは動的に変更できます。

並列化は、トランザクション・リカバリ、レプリケーション、SQL実行など、様々な機能で使用できます。ここで説明しているSQLのパラレル実行の場合、1つの文の実行フェーズ全体を通して、複数のパラレル実行サーバー・プロセスが対応付けられます。文の処理が完了すると、その文に対応付けられていたプロセスが他の文を処理できるようになります。

ノート:

データベースでパラレル SQL 実行を無効にするには、PARALLEL_MAX_SERVERS 初期化パラメータ値を 0 に設定します。

関連項目:

- パラレル・ヒントの使用の詳細は、『Oracle Database SQLチューニング・ガイド』
- パラレル実行の使用に関する詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: SQLのパラレル実行用プロセスの管理

5.8.2 セッションのパラレル実行の変更

セッションのSQLのパラレル実行は、ALTER SESSION文を使用して制御できます。

- SQLのパラレル実行を使用禁止にする方法
  SQLのパラレル実行は、ALTER SESSION DISABLE PARALLEL DML|DDL|QUERY文で使用禁止にできます。この文を発行した後は、後続のすべてのDML(INSERT、UPDATE、DELETE)、DDL(CREATE、ALTER)または問合せ(SELECT)操作がシリアルに実行されます。関係する表または索引にパラレル属性が指定されている場合でも、それとは無関係に文はシリアルに実行されます。ただし、PARALLELヒントが指定された文は、セッションの設定を無効にします。
- SQLのパラレル実行を使用可能にする方法
  SQLのパラレル実行は、ALTER SESSION ENABLE PARALLEL DML|DDL|QUERY文で使用可能にできます。その後、PARALLEL句またはパラレル・ヒントを文に対応付けると、そのDML文、DDL文または問合せ文はパラレルで実行されます。DDL文と問合せ文のパラレル実行はデフォルトで使用可能です。
- SQLのパラレル実行の強制
  ALTER SESSION FORCE PARALLEL DML|DDL|QUERY文により、パラレル化が可能な後続のすべてのDML、DDLまたは問合せ文をパラレルで実行するように強制できます。

親トピック: SQLのパラレル実行用プロセスの管理

5.8.2.1 SQLのパラレル実行を使用禁止にする方法

SQLのパラレル実行は、ALTER SESSION DISABLE PARALLEL DML|DDL|QUERY文で使用禁止にできます。この文を発行した後は、後続のすべてのDML(INSERT、UPDATE、DELETE)、DDL(CREATE、ALTER)または問合せ(SELECT)操作がシリアルに実行されます。関係する表または索引にパラレル属性が指定されている場合でも、それとは無関係に文はシリアルに実行されます。ただし、PARALLELヒントが指定された文は、セッションの設定を無効にします。

- DML、DDLまたは問合せ操作を無効にするために、適切なALTER SESSION DISABLE PARALLEL文を実行します。

たとえば、パラレルDDL操作を無効にするには、次の文を実行します。

```
ALTER SESSION DISABLE PARALLEL DDL;
```

親トピック: セッションのパラレル実行の変更

5.8.2.2 SQLのパラレル実行を使用可能にする方法

SQLのパラレル実行は、ALTER SESSION ENABLE PARALLEL DML|DDL|QUERY文で使用可能にできます。その後、PARALLEL句またはパラレル・ヒントを文に対応付けると、そのDML文、DDL文または問合せ文はパラレルで実行されます。DDL文と問合せ文のパラレル実行はデフォルトで使用可能です。

- DML、DDLまたは問合せ操作を有効にするために、適切なALTER SESSION DISABLE PARALLEL文を実行します。

たとえば、DML文は、ALTER SESSION文を明示的に発行して、パラレルDMLを使用可能にした場合のみパラレル化できます。

```
ALTER SESSION ENABLE PARALLEL DML;
```

親トピック: セッションのパラレル実行の変更

5.8.2.3 SQLのパラレル実行の強制

ALTER SESSION FORCE PARALLEL DML|DDL|QUERY文を発行すると、後続のすべてのDML、DDLまたは問合せ文をパラレルで実行するように強制できます。

特定の並列度を強制的に適用して、後続の文に対応付けられたPARALLEL句を無効にできます。ALTER SESSION文で並列度を指定しなかった場合は、デフォルトの並列度が使用されます。文レベルのパラレル・ヒントを指定すると、強制的な並列度がオーバーライドされます。表レベルのパラレル・ヒントの場合、動作は、すべての表にヒントが提供されているかどうかによって異なります。すべての表に表レベルのパラレル・ヒントが含まれる場合、それらのヒントの中で最大の値が使用されます。少なくとも1つの表に表レベルのパラレル・ヒントが含まれていない場合、使用される並列度は、パラレル・ヒント間で最も高い値およびALTER SESSIONコマンドで指定された並列度です。

パラレル実行を強制する手順は、次のとおりです。

- ALTER SESSION FORCE PARALLEL文を実行します。

たとえば、次の文は、後続の文のパラレル実行を強制し、優先する並列度を5に設定します。

```
ALTER SESSION FORCE PARALLEL DDL PARALLEL 5;
```

親トピック: セッションのパラレル実行の変更

5.9 外部プロシージャのプロセスの管理

外部プロシージャは、プログラミング言語で作成され、共有ライブラリに格納されるプロシージャまたはファンクションです。Oracleサーバーは、PL/SQLルーチンを使用して、外部プロシージャまたはファンクションをコールできます。

- 外部プロシージャについて
  外部プロシージャはC、C++、Javaなどのプログラミング言語で記述されているプロシージャであり、データベースの外部でコンパイルおよび格納された後、ユーザー・セッションによってコールされます。たとえば、PL/SQLプログラム・ユニットから、特別な用途の処理を実行するのに必要な1つ以上のCルーチンをコールできます。
- 外部プロシージャ・コールを有効化するためのDBAのタスク
  外部プロシージャ・コールを有効にするには、リスナーの変更およびライブラリの管理が必要です。

親トピック: プロセスの管理

5.9.1 外部プロシージャについて

外部プロシージャはC、C++、Javaなどのプログラミング言語で記述されているプロシージャであり、データベースの外部でコンパイルおよび格納された後、ユーザー・セッションによってコールされます。たとえば、PL/SQLプログラム・ユニットから、特別な用途の処理を実行するのに必要な1つ以上のCルーチンをコールできます。

これらのコール可能ルーチンは、Dynamic Link Library(DLL)またはJavaクラス・メソッドの場合はライブラリ・ユニットに格納されており、ベース言語で登録されています。Oracle Databaseには、コール仕様という特別な用途のインタフェースが用意されており、ユーザーはこのインタフェースを使用して外部プロシージャをコールできます。

ユーザー・セッションによって外部プロシージャが呼び出されると、データベースによって外部プロシージャ・エージェントがデータベースのホスト・コンピュータ上で開始されます。エージェントのデフォルトの名前は`extproc`です。各セッションには専用エージェントがあります。必要に応じて、エージェントが特定のオペレーティング・システム・ユーザーとして実行されるよう、資格証明を作成できます。セッションが終了すると、データベースではそのエージェントを終了します。

ユーザー・アプリケーションは、外部プロシージャ・エージェントに対してDLLまたはライブラリ・ユニットの名前、外部プロシージャの名前およびすべての関連するパラメータを渡します。次に、外部プロシージャ・エージェントはDLLまたはライブラリ・ユニットをロードして外部プロシージャを実行し、外部プロシージャから返された値をアプリケーションに返送します。

関連項目:

外部プロシージャの詳細は、『Oracle Database開発ガイド』

親トピック: 外部プロシージャのプロセスの管理

5.9.2 外部プロシージャ・コールを有効化するためのDBAのタスク

外部プロシージャ・コールを有効にするには、リスナーの変更およびライブラリの管理が必要です。

外部プロシージャ・コールを有効にするには、次のDBAタスクが必要になる場合があります。

- `extproc`エージェントを起動するためのリスナーの構成

  デフォルトでは、`extproc`はデータベースによって起動されます。次の状況では、リスナーによって`extproc`が起動されるように、このデフォルト構成を変更する必要があります。

  - マルチスレッドの`extproc`エージェントを使用する場合
  - データベースがWindowsにおいて共有サーバー・モードで実行されている場合
  - `LIBRARY`指定の`AGENT`句または`PROCEDURE`または`FUNCTION`指定の`AGENT IN`句によって、外部プロシージャが異なる`extproc`エージェントにリダイレクトされる場合

  デフォルト構成を変更する手順は、『Oracle Database開発ガイド』を参照してください。

- ライブラリの管理、またはライブラリの管理に関連のある権限の付与

  DLL文には、ライブラリと呼ばれるスキーマ・オブジェクトを経由してアクセスする必要があります。セキュリティを確保するために、デフォルトではDBAロールを持つユーザーのみがライブラリを作成および管理できます。このため、次のことを依頼される場合があります。

  - ライブラリの場所に対して`CREATE DIRECTORY`文を使用して、ディレクトリ・オブジェクトを作成します。ディレクトリ・オブジェクトの作成後、`CREATE LIBRARY`文を使用して、このディレクトリ・オブジェクトをライブラリの

場所に指定できます。

- DBMS_CREDENTIAL.CREATE_CREDENTIAL PL/SQLプロシージャを使用して、資格証明を作成します。資格証明の作成後、CREATE LIBRARY文を使用して、extprocエージェントを特定のオペレーティング・システム・ユーザーとして実行するライブラリに、この資格証明を関連付けることができます。
- CREATE LIBRARY文を使用して、開発者が必要とするライブラリ・オブジェクトを作成します。
- 権限CREATE LIBRARY、CREATE ANY LIBRARY、ALTER ANY LIBRARY、EXECUTE ANY LIBRARY、EXECUTE ON library_nameおよびEXECUTE ON directory_objectを開発者に付与します。

  これらの権限の明示的付与は、信頼できるユーザーに対してのみ行ってください。決してPUBLICロールに対して付与しないでください。ライブラリにPL/SQLインタフェースを作成する場合、EXECUTE権限のみをPL/SQLインタフェースに付与してください。基礎になるライブラリにEXECUTEを付与しないでください。PL/SQLインタフェースを作成するには、ライブラリに対するEXECUTEオブジェクト権限が必要です。ただし、ユーザーは自らのスキーマに対しては自動的にこの権限を持っています。ライブラリに対してEXECUTEオブジェクト権限を明示的に付与する必要はほとんどりません。

関連項目:

- CREATE LIBRARY文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。
- DBMS_CREDENTIAL.CREATE_CREDENTIALプロシージャを使用した資格証明の作成の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- DBMS_CREDENTIALパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- Oracle Schedulerジョブでの資格証明の使用の詳細は、「スケジューラ・ジョブの資格証明の指定」

親トピック: 外部プロシージャのプロセスの管理

5.10 セッションの終了

時々、現行のユーザー・セッションを停止する必要があります。たとえば、管理操作を実行する場合に、すべての管理セッション以外のセッションを停止する必要があります。

- セッションの終了について
  セッションが終了すると、そのセッションのアクティブ・トランザクションがロールバックされ、そのセッションが保持していたリソース(ロックやメモリー領域など)がただちに解放されて、他のセッションで使用可能になります。
- 終了するセッションの識別
  終了するセッションを識別するには、セッションの索引番号とシリアル番号を指定します。
- アクティブなセッションの終了
  アクティブなセッションを終了すると、セッションが終了します。
- 非アクティブなセッションの終了
  終了時にセッションがOracle Databaseに対してSQLコールを実行していない場合(INACTIVE)、ORA-00028のメッセージはただちには返されません。このメッセージは、その後ユーザーが停止したセッションを使用しようとしたときに返されます。
- セッション内のSQL文の取消し

セッション内のSQL文は、ALTER SYSTEM CANCEL SQL文を使用して取消しできます。

親トピック: プロセスの管理

5.10.1 セッションの終了について

セッションが停止すると、そのセッションのアクティブ・トランザクションがロールバックされ、そのセッションが保持していたリソース(ロックやメモリー領域など)がただちに解放されて、他のセッションで使用可能になります。

SQL文ALTER SYSTEM KILL SESSIONを使用して、現行のセッションを停止します。次の文は、システム識別子が7でシリアル番号が15のセッションを停止します。

```
ALTER SYSTEM KILL SESSION '7,15';
```

DBMS_SERVICE.DISCONNECT_SESSIONプロシージャを使用して、現行インスタンスにおいて指定したサービスからセッションを停止することもできます。

関連項目:

DISCONNECT_SESSIONプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: セッションの終了

5.10.2 停止するセッションの識別

停止するセッションを識別するには、セッションの索引番号とシリアル番号を指定します。

セッションのシステム識別子(SID)とシリアル番号を識別するには:

- V$SESSION動的パスワード・ビューを問い合せます。

たとえば、次の問合せはユーザーjwardのすべてのセッションを識別します。

```
SELECT SID, SERIAL#, STATUS
  FROM V$SESSION
  WHERE USERNAME = 'JWARD';
SID    SERIAL#    STATUS
-----  ---------  --------
    7         15  ACTIVE
   12         63  INACTIVE
```

Oracle Databaseに対してSQLコールを実行しているとき、セッションはACTIVEです。データベースに対してSQLコールを実行していないとき、セッションはINACTIVEです。

関連項目:

セッションの状態値については、『Oracle Databaseリファレンス』

親トピック: セッションの終了

5.10.3 アクティブ・セッションの停止

アクティブなセッションを終了すると、セッションが終了します。

セッションの終了時にユーザー・セッションがトランザクションを処理している場合(STATUSがACTIVE)、トランザクションはロールバックされ、ユーザーはただちに次のメッセージを受け取ります。

```
ORA-00028: your session has been killed
```

ORA-00028メッセージを受け取った後、データベースに再接続する前に追加の文を実行すると、Oracle Databaseは次のメッセージを返します。

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

ネットワークI/Oやトランザクションのロールバックを実行している場合は、アクティブ・セッションを中断できません。そのセッションは、操作が完了するまで停止できません。この場合、セッションは、停止するまですべてのリソースを保持します。また、セッションを停止させるためにALTER SYSTEM文を発行したセッションは、対象のセッションが停止するまで最大60秒待機します。中断できなかった操作が1分経過しても終了しない場合、ALTER SYSTEM文の発行者は、セッションが停止されることを示すマークが設定されたというメッセージを受け取ります。停止マークが付けられたセッションは、V$SESSIONでの状態(STATUS)がKILLEDになり、サーバー(SERVER)がPSEUDO以外の値になります。

アプリケーション・コンティニュイティを使用している場合、アクティブ・セッションのアクティビティはセッションが終了したときにリカバリされます。セッションの停止後にセッションをリカバリしない場合は、NOREPLAYキーワードをALTER SYSTEM文に組み込むことができます。たとえば、次の文では、セッションをリカバリしないことを指定します。

```
ALTER SYSTEM KILL SESSION '7,15' NOREPLAY;
```

DBMS_SERVICE.DISCONNECT_SESSIONプロシージャを使用して1つ以上のセッションを停止する場合、アプリケーション・コンティニュイティによってセッションがリカバリされないように、disconnect_optionパラメータにDBMS_SERVICE.NOREPLAYを指定できます。たとえば、すべてのセッションをサービスsales.example.comから切断し、セッションをリカバリしないことを指定するには、次のプロシージャを実行します。

```
BEGIN
  DBMS_SERVICE.DISCONNECT_SESSION(
    service_name      => 'sales.example.com',
    disconnect_option => DBMS_SERVICE.NOREPLAY);
END;
/
```

関連項目:

- 「トランザクション・ガードおよびアプリケーション・コンティニュイティ」
- DISCONNECT_SESSIONプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: セッションの終了

5.10.4 非アクティブ・セッションの停止

停止時にセッションがOracle Databaseに対してSQLコールを実行していない場合(INACTIVE)、ORA-00028のメッセージはただちには返されません。このメッセージは、その後ユーザーが停止したセッションを使用しようとしたときに返されます。

非アクティブ・セッションを停止すると、V$SESSIONビューのセッションのSTATUSがKILLEDになります。停止したセッションの行は、ユーザーが再びそのセッションを使用しようとしてORA-00028メッセージを受け取った後、V$SESSIONから削除されます。

次の例では、非アクティブ・セッションを停止しています。最初にV$SESSIONを問い合せてセッションのSIDとSERIAL#を識別

してから、セッションを停止しています。

```
SELECT SID,SERIAL#,STATUS,SERVER
   FROM V$SESSION
   WHERE USERNAME = 'JWARD';
SID    SERIAL#   STATUS     SERVER
-----  --------  ---------  ---------
    7        15  INACTIVE   DEDICATED
   12        63  INACTIVE   DEDICATED
2 rows selected.
ALTER SYSTEM KILL SESSION '7,15';
Statement processed.
SELECT SID, SERIAL#, STATUS, SERVER
   FROM V$SESSION
   WHERE USERNAME = 'JWARD';
SID    SERIAL#   STATUS     SERVER
-----  --------  ---------  ---------
    7        15  KILLED     PSEUDO
   12        63  INACTIVE   DEDICATED
2 rows selected.
```

親トピック: セッションの終了

5.10.5 セッション内のSQL文の取消し

セッション内のSQL文は、ALTER SYSTEM CANCEL SQL文を使用して取消しできます。

セッションを終了するのではなく、セッション内の高負荷SQL文を取り消すことができます。DML文を取り消すと、その文はロールバックされます。

ALTER SYSTEM CANCEL SQL文に必須の句は次のとおりです。

- SID – セッションID
- SERIAL – セッション・シリアル番号

ALTER SYSTEM CANCEL SQL文のオプションの句は次のとおりです。

- INST_ID – インスタンスID
- SQL_ID – SQL文のSQL ID

セッションのこの情報は、GV$SESSIONビューを問い合せることで確認できます。

SQL文を取り消す場合の構文は、次のとおりです。

```
ALTER SYSTEM CANCEL SQL 'SID, SERIAL, @INST_ID, SQL_ID';
```

次の例では、セッション識別子が20、セッション・シリアル番号が51142、SQL IDが8vu7s907prbgrのSQL文を取り消します。

```
ALTER SYSTEM CANCEL SQL '20, 51142, 8vu7s907prbgr';
```

ノート:

- @INST_ID の指定を省略すると、現在のセッションのインスタンス ID が使用されます。
- SQL_ID の指定を省略すると、指定されたセッションで現在実行中の SQL 文が終了されます。

関連項目:

- 高負荷SQL文の識別の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください
- GV$SESSIONビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: セッションの終了

5.11 プロセスおよびセッションのデータ・ディクショナリ・ビュー

プロセスおよびセッションの情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

ビュー	説明
V$PROCESS	現在アクティブになっているプロセスの情報が含まれています。
V$SESSION	現行のセッションごとにセッション情報がリストされます。
V$SESS_IO	各ユーザー・セッションの I/O 統計情報が含まれています。
V$SESSION_LONGOPS	6 秒(絶対時間)以上実行されていた各種操作の状態が表示されます。現在、状態が表示される操作は、多数のバックアップおよびリカバリ機能、統計収集および問合せの実行などです。Oracle Database のリリースごとにさらに操作が追加されます。
V$SESSION_WAIT	各セッションの現在または最後の待機が表示されます。
V$SESSION_WAIT_HISTORY	各アクティブ・セッションの最後の 10 個の待機イベントがリストされます。
V$WAIT_CHAINS	ブロックされたセッションに関する情報が表示されます。
V$SESSTAT	セッション統計情報が含まれています。
V$RESOURCE_LIMIT	一部のシステム・リソースについて、現行および最大のグローバル・リソース使用率が表示されます。
V$SQLAREA	共有 SQL 領域に関する統計情報が含まれています。SQL 文字列ごとに 1 行ずつ含まれます。メモリー内にあり、解析済で、実行準備のできている SQL 文に関する統計情報も提供します。

親トピック: プロセスの管理

6 メモリーの管理

メモリー管理には、データベースの変更に応じたOracle Databaseインスタンス・メモリー構造の最適なサイズのメンテナンスが含まれます。

- メモリー管理について
  管理する必要があるメモリー構造は、システム・グローバル領域(SGA)とインスタンス・プログラム・グローバル領域(インスタンスPGA)です。Oracle Databaseでは様々なメモリー管理方法がサポートされており、これらは初期化パラメータの設定で選択されます。
- メモリー・アーキテクチャの概要
  Oracle Databaseに関連する基本メモリー構造を理解します。
- 自動メモリー管理の使用
  Oracle Databaseインスタンスが自動的にメモリーを管理およびチューニングできるようにすることができます。
- メモリーの手動構成
  個々のメモリー・コンポーネントのサイズをより直接的に制御する場合は、自動メモリー管理を無効にし、手動メモリー管理用にデータベースを構成できます。
- 強制フル・データベース・キャッシュ・モードの使用方法
  Oracle Databaseインスタンスは、全データベースをバッファ・キャッシュにキャッシュできます。
- Database Smart Flash Cacheの構成
  データベース・スマート・フラッシュ・キャッシュ機能は、ソリッド・ステート・デバイス(SSD)テクノロジを使用してデータベース・バッファ・キャッシュを透過的に拡張します。Database Smart Flash Cacheにより、ディスクI/O量が削除され、同等サイズのRAMを追加するより低いコストでOracleデータベースのパフォーマンスを大幅に向上させることができます。
- Oracle Database In-Memoryによる問合せパフォーマンスの向上
  Oracle Database In-Memory (Database In-Memory)は、Oracle Database 12cリリース1 (12.1.0.2)に最初に導入された一連の機能であり、リアルタイム分析および混合ワークロードのパフォーマンスを大幅に改善します。
- Memoptimized Rowstoreを使用する高パフォーマンス・データ・ストリーミングの有効化
  Memoptimized Rowstoreでは、IoT (モノのインターネット)アプリケーションなど、アプリケーションでの高パフォーマンス・データ・ストリーミングが可能になります。この機能は、通常は、同時に多数のクライアントから単一行挿入で少量のデータをストリーミングし、また、非常に高い頻度でクライアントに対してデータを問い合せます。
- メモリー管理の参考情報
  自動メモリー管理は、一部のプラットフォームでのみサポートされます。また、メモリー管理に関する情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

親トピック: 基本データベース管理

6.1 メモリー管理について

管理する必要があるメモリー構造は、システム・グローバル領域(SGA)とインスタンス・プログラム・グローバル領域(インスタンスPGA)です。Oracle Databaseでは様々なメモリー管理方法がサポートされており、これらは初期化パラメータの設定で選択されます。

自動メモリー管理

Oracle DatabaseはSGAメモリーとインスタンスPGAメモリーを完全に自動的に管理できます。インスタンスで使用される合計メモリー・サイズを指定するだけで、Oracle Databaseが必要に応じてSGAとインスタンスPGAの間でメモリーを動的に交換し、

処理ニーズに対応します。この機能を自動メモリー管理と呼びます。このメモリー管理の方法では、データベースは個別SGAコンポーネントのサイズと個別PGAのサイズも動的にチューニングします。SGAおよびPGAメモリーの合計サイズが4GB以下であるデータベースの自動メモリー管理をお薦めします。

手動メモリー管理

個々のメモリー・コンポーネントのサイズをより直接的に制御する場合は、自動メモリー管理を使用禁止にして、手動メモリー管理用にデータベースを構成できます。手動メモリー管理として使用できる方法はいくつかあります。これらの方法には、自動の部分が、ある度合で残っているものがあります。それらの方法では、DBAに要求される作業量および知識量が異なります。次の方法があります。

- 自動共有メモリー管理 - SGA用
- 手動共有メモリー管理 - SGA用
- 自動PGAメモリー管理 - インスタンスPGA用
- 手動PGAメモリー管理 - インスタンスPGA用

これらのメモリー管理方法については、この章で後述します。

Database Configuration Assistant (DBCA)を使用してデータベースを作成する際に基本インストール・オプションを選択すると、システム・メモリーが4GB以下である場合、自動メモリー管理が使用可能になります。システム・メモリーが4GBを超える場合は、自動メモリー管理は使用禁止になり、自動共有メモリー管理が使用可能になります。拡張インストールを選択した場合は、DBCAを使用して、自動メモリー管理または自動共有メモリー管理を選択できます。

SGAおよびPGAメモリーの合計サイズが4GB以上である場合の自動共有メモリー管理をお薦めします。

ノート:

メモリーを管理する最も簡単な方法は、Oracle Enterprise Manager Database Express (EM Express)または Oracle Enterprise Manager Cloud Control (Cloud Control)のグラフィカル・ユーザー・インタフェースを使用する方法です。

EM Express を使用したメモリー管理の詳細は、『Oracle Database 2 日でデータベース管理者』を参照してください。

Cloud Control を使用したメモリー管理の詳細は、Cloud Control のオンライン・ヘルプを参照してください。

関連項目:

メモリーの自動管理および手動管理の様々な方法については、『Oracle Database概要』を参照してください。

親トピック: メモリーの管理

6.2 メモリー・アーキテクチャの概要

Oracle Databaseに関連する基本メモリー構造を理解します。

Oracle Databaseに関連する基本的なメモリー構造は、次のような領域で構成されています。

- システム・グローバル領域(SGA)

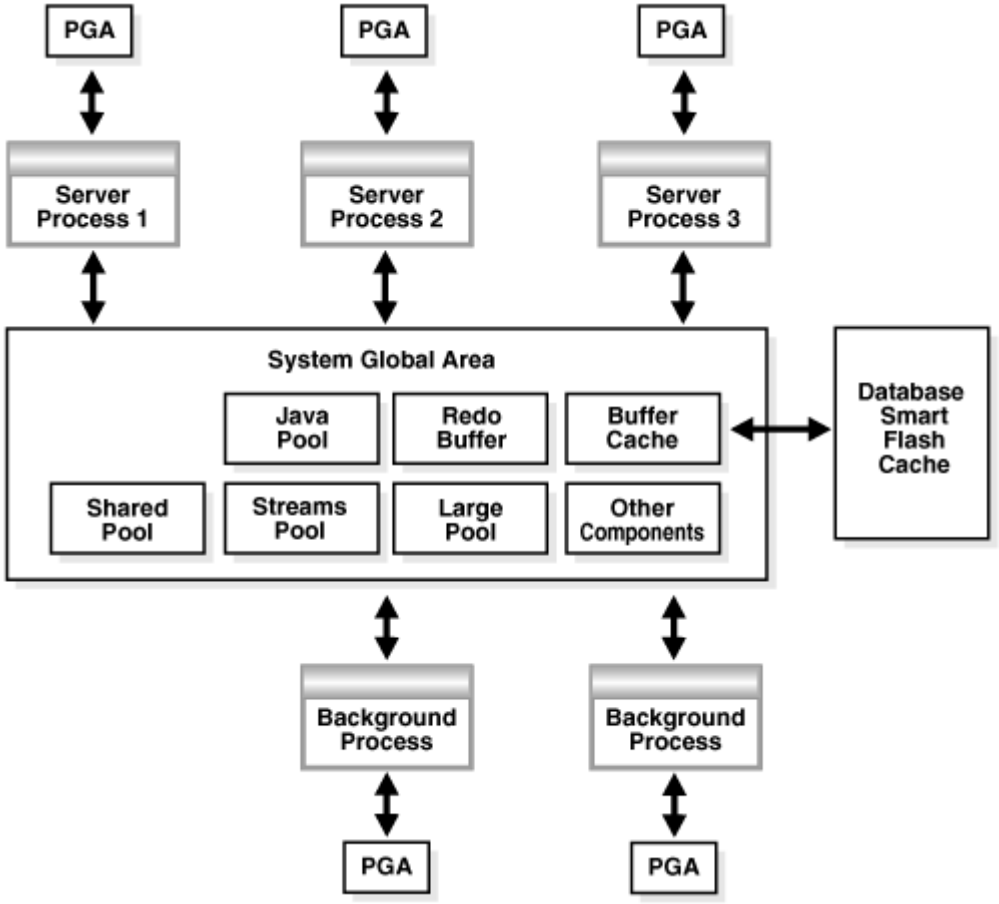
  SGAは、SGAコンポーネントと呼ばれる共有メモリー構造のグループで、1つのOracle Databaseインスタンスに関するデータと制御情報が保存されています。SGAは、すべてのサーバー・プロセスとバックグラウンド・プロセスで共有されます。SGAに格納されるデータには、キャッシュ・データ・ブロックや共有SQL領域などがあります。

- プログラム・グローバル領域(PGA)

  PGAは、サーバー・プロセスのデータおよび制御情報が含まれるメモリー領域です。これはサーバー・プロセスの開始時にOracle Databaseによって作成される非共有メモリーです。PGAへのアクセスは、サーバー・プロセスごとに排他的です。各サーバー・プロセスごとに1つのPGAが存在します。バックグラウンド・プロセスにも独自のPGAが割り当てられます。Oracle Databaseインスタンスに連結されるすべてのバックグラウンドおよびサーバー・プロセスに割り当てられるPGAメモリーの合計はインスタンスPGAメモリー合計と呼ばれ、個々のPGAの集合はインスタンスPGA合計、または単にインスタンスPGAと呼ばれます。

図6-1に、これらのメモリー構造の関係を示します。

図6-1 Oracle Databaseのメモリー構造



データベースがSolarisまたはOracle Linux上で稼働している場合、オプションで別のメモリー・コンポーネント、Database Smart Flash Cacheもオプションで追加できます。Database Smart Flash Cacheは、SGAに常駐するバッファ・キャッシュの拡張機能で、データベース・ブロックにレベル2のキャッシュを提供します。これにより、読取り集中型のオンライン・トランザクション処理(OLTP)ワークロードと非定型問合せの両方、およびデータ・ウェアハウス環境での大量データ変換について、応答時間およびスループット全体を改善できます。Database Smart Flash Cacheは、フラッシュ・メモリーを使用するソリッド状態の記憶装置である1つ以上のフラッシュ・ディスク装置に常駐します。Database Smart Flash Cacheは、一般的に、メイン・メモリーの増設よりも経済的であり、かつ、ディスク・ドライブよりもはるかに高速です。

Oracle Database 12cリリース1 (12.1.0.2)以上では、大規模表キャッシュにより、シリアル問合せおよびパラレル問合せで

バッファ・キャッシュを使用できるようになります。大規模表キャッシュは、データ・ウェアハウス環境の大規模表について、このような表がバッファ・キャッシュに完全には収まらない場合でも、効率的なキャッシングを簡単にできるようにします。表スキャンでは、次の場合に大規模表キャッシュを使用できます。

- パラレル問合せ

  シングル・インスタンス・データベースおよびOracle Real Application Clusters (Oracle RAC)データベースで、DB_BIG_TABLE_CACHE_PERCENT_TARGET初期化パラメータがゼロ以外の値に設定され、PARALLEL_DEGREE_POLICYがAUTOまたはADAPTIVEに設定されている場合、パラレル問合せで大規模表キャッシュを使用できます。

- シリアル問合せ

  シングル・インスタンス構成のみで、DB_BIG_TABLE_CACHE_PERCENT_TARGET初期化パラメータがゼロ以外の値に設定されている場合、シリアル問合せで大規模表キャッシュを使用できます。

関連項目:

- 「Database Smart Flash Cacheの構成」
- Oracle Databaseインスタンスのメモリー・アーキテクチャの詳細は、『Oracle Database概要』を参照してください。
- DB_BIG_TABLE_CACHE_PERCENT_TARGET初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- PARALLEL_DEGREE_POLICY初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 大規模表キャッシュの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: メモリーの管理

6.3 自動メモリー管理の使用

Oracle Databaseインスタンスが自動的にメモリーを管理およびチューニングできるようにすることができます。

- 自動メモリー管理について
  インスタンス・メモリーの管理は、Oracle Databaseインスタンスが自動的に管理およびチューニングできるようにすることが最も簡単な方法です。これを実行するには、ほとんどのプラットフォームの場合、ターゲット・メモリー・サイズ初期化パラメータ(MEMORY_TARGET)とオプションで最大メモリー・サイズ初期化パラメータ(MEMORY_MAX_TARGET)のみを設定します。
- 自動メモリー管理の有効化
  データベースの作成時に(DBCAで適切なオプションを選択するか、またはCREATE DATABASE SQL文に適切な初期化パラメータを設定して)自動メモリー管理を使用可能にしなかった場合は、後で自動メモリー管理を使用可能にできます。自動メモリー管理を有効化する場合は、データベースを停止して再起動する必要があります。
- 自動メモリー管理の監視およびチューニング
  動的なパフォーマンス・ビューV$MEMORY_DYNAMIC_COMPONENTSに、SGAとインスタンスPGAの合計サイズなど、動的にチューニングされたすべてのメモリー・コンポーネントの現行サイズが示されます。

親トピック: メモリーの管理

6.3.1 自動メモリー管理について

インスタンス・メモリーを管理する最も単純な方法は、Oracle Databaseインスタンスで自動的に管理およびチューニングされるようにすることです。これを実行するには、ほとんどのプラットフォームの場合、ターゲット・メモリー・サイズ初期化パラメータ(MEMORY_TARGET)とオプションで最大メモリー・サイズ初期化パラメータ(MEMORY_MAX_TARGET)のみを設定します。

インスタンスによって使用される合計メモリーはMEMORY_TARGETの値に基づき、相対的に一定なままとなり、メモリーはインスタンスによってシステム・グローバル領域(SGA)とインスタンス・プログラム・グローバル領域(インスタンスPGA)の間で自動的に配分されます。メモリー要件の変化に応じて、メモリーはインスタンスによってSGAとインスタンスPGAの間で動的に再配分されます。

自動メモリー管理が有効化されていない場合は、SGAとインスタンスPGAの両方を手動でサイズ設定する必要があります。

MEMORY_TARGET初期化パラメータは動的であるため、データベースの再起動なしにいつでもMEMORY_TARGETを変更できます。MEMORY_MAX_TARGETは動的ではなく、誤ってMEMORY_TARGETを高く設定しすぎないための上限として機能し、今後インスタンス・メモリーの合計を増加する際に、データベース・インスタンス用に十分なメモリーを確保するために使用されます。また、一部のSGAコンポーネントは簡単に縮小できないか、または最小サイズに抑える必要があるため、インスタンスではユーザーがMEMORY_TARGETの設定を低くしすぎないようにします。

ノート:

- データベース・インスタンスの合計物理メモリーが 4GB を超える場合は、データベースのインストール時および作成時に自動メモリー管理オプションを指定できません。このような環境では、自動共有メモリー管理の使用をお薦めします。
- LOCK_SGA 初期化パラメータが TRUE に設定されている場合は、自動メモリー管理を使用可能にできません。このパラメータの詳細は、『Oracle Database リファレンス』を参照してください。

関連項目:

「自動メモリー管理をサポートするプラットフォーム」

親トピック: 自動メモリー管理の使用

6.3.2 自動メモリー管理の有効化

データベースの作成時に(DBCAで適切なオプションを選択するか、またはCREATE DATABASE SQL文に適切な初期化パラメータを設定して)自動メモリー管理を使用可能にしなかった場合は、後で自動メモリー管理を使用可能にできます。自動メモリー管理を有効化する場合は、データベースを停止して再起動する必要があります。

自動メモリー管理を有効にするには:

1. SQL*Plusを起動して、SYSDBA管理権限でOracle Databaseインスタンスに接続します。

   手順については、「SQL*Plusを使用したデータベースへの接続」および「データベース管理者の認証」を参照してください。

2. MEMORY_TARGETの最小値を次の方法で計算します。

   a. 次のSQL*Plusコマンドを入力して、SGA_TARGETおよびPGA_AGGREGATE_TARGETの現行サイズ(MB)を確認します。

```
SHOW PARAMETER SGA_TARGET
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------
--
sga_target                           big integer 272M
SHOW PARAMETER PGA_AGGREGATE_TARGET
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------
--
pga_aggregate_target                 big integer 90M
```

SGA_TARGETパラメータが未設定の場合、その設定の詳細は、「自動共有メモリー管理を使用可能にする方法」を参照してください。

b. 次の問合せを実行して、データベースの起動以降に割り当てられた最大インスタンスPGA (MB)を確認します。

```
SELECT VALUE/1048576 FROM V$PGASTAT WHERE NAME='maximum PGA allocated';
```

c. ステップ2bおよびPGA_AGGREGATE_TARGETの問合せの結果を比較して最大値を算定します。この値にSGA_TARGETを加算します。

```
MEMORY_TARGET = SGA_TARGET + MAX(PGA_AGGREGATE_TARGET, MAXIMUM PGA
ALLOCATED)
```

たとえば、前述のようにSGA_TARGETが272M、PGA_AGGREGATE_TARGETが90Mで、割当て済の最大PGAが120Mと確認された場合、MEMORY_TARGETは392M(272M + 120M)以上にする必要があります。

3. 使用するMEMORY_TARGETの値を選択します。

この値は、ステップ2で計算した最小値にするか、または使用可能な物理メモリーが十分ある場合はこれより大きい値を使用できます。

4. MEMORY_MAX_TARGET初期化パラメータについては、予測可能な範囲で、データベースに割り当てる予定の最大メモリー量に決定します。つまり、SGAとインスタンスPGAサイズの合計に対する最大値を決定します。この値は、前述のステップで選択したMEMORY_TARGETの値以上に設定できます。

5. 次のいずれかの操作を行います。

a. Oracle Databaseインスタンスをサーバー・パラメータ・ファイルを使用して起動(Database Configuration Assistant(DBCA)を使用してデータベースを作成した場合のデフォルト)した場合は、次のコマンドを入力します。

```
ALTER SYSTEM SET MEMORY_MAX_TARGET = nM SCOPE = SPFILE;
```

nはステップ4で計算した値です。

SCOPE = SPFILE句を指定すると、サーバー・パラメータ・ファイル内の値のみが設定され、実行中のインスタンスに対する値は設定されません。MEMORY_MAX_TARGETは動的な初期化パラメータではないため、このSCOPE句を組み込む必要があります。

b. インスタンスをテキスト形式の初期化パラメータ・ファイルを使用して起動した場合は、ファイルを手動で編集して次の文を組み込みます。

```
memory_max_target = nM
memory_target = mM
```

nはステップ4で決定した値、mはステップ3で決定した値です。

ノート:

テキスト形式の初期化パラメータ・ファイルでは、MEMORY_MAX_TARGET の行を省略して MEMORY_TARGET の値を指定した場合、データベースによって、MEMORY_MAX_TARGET は MEMORY_TARGET の値に自動的に設定されます。MEMORY_TARGET の行を省略して MEMORY_MAX_TARGET を指定した場合、MEMORY_TARGET パラメータはデフォルトで 0 (ゼロ)に設定されます。起動後、MEMORY_MAX_TARGET の値を超えないかぎり、MEMORY_TARGET を 0 以外の値に動的に変更できます。

6. データベースを停止して再起動します。

   手順については、「起動と停止」を参照してください。

7. Oracle Databaseインスタンスをサーバー・パラメータ・ファイルを使用して起動した場合は、次のコマンドを入力します。

```
ALTER SYSTEM SET MEMORY_TARGET = nM;
ALTER SYSTEM SET SGA_TARGET = 0;
ALTER SYSTEM SET PGA_AGGREGATE_TARGET = 0;
```

   nはステップ3で決定した値です。

ノート:

MEMORY_TARGET を設定すると、SGA_TARGET 設定は SGA の最小サイズになり、PGA_AGGREGATE_TARGET 設定はインスタンス PGA の最小サイズになります。説明したようにこの両方をゼロに設定すると、最小値の設定がなくなり、SGA およびインスタンス PGA は、その合計が MEMORY_TARGET 設定以下であれば、必要に応じて大きくなります。SQL 作業領域のサイズ変更は、自動のままです。

SGA_TARGET および PGA_AGGREGATE_TARGET のパラメータ値を 0(ゼロ)に設定する文を省略し、いずれかまたは両方の値を正数にしておくことができます。この場合、値は SGA またはインスタンス PGA のサイズの最小値として機能します。

また、PGA_AGGREGATE_LIMIT 初期化パラメータを使用して、PGA メモリーにインスタンス全体の厳格な制限を設定することもできます。PGA_AGGREGATE_LIMIT は、自動メモリー管理を使用しているかどうかに関係なく設定できます。「自動 PGA メモリー管理の使用」を参照してください。

関連項目:

- 「自動メモリー管理について」
- 「メモリー・アーキテクチャの概要」
- ALTER SYSTEM SQL文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 自動メモリー管理の使用

6.3.3 自動メモリー管理の監視およびチューニング

動的なパフォーマンス・ビューV$MEMORY_DYNAMIC_COMPONENTSに、SGAとインスタンスPGAの合計サイズなど、動的にチューニングされたすべてのメモリー・コンポーネントの現行サイズが示されます。

- MEMORY_TARGET初期化パラメータのチューニング・アドバイスについては、V$MEMORY_TARGET_ADVICEビューを問い合せます。

たとえば、次の問合せを実行します。

```
SQL>  select * from v$memory_target_advice order by memory_size;

MEMORY_SIZE MEMORY_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR    VERSION
----------- ------------------ ------------ -------------------- ----------
        180                 .5          458                1.344          0
        270                .75          367               1.0761          0
        360                  1          341                    1          0
        450               1.25          335                .9817          0
        540                1.5          335                .9817          0
        630               1.75          335                .9817          0
        720                  2          335                .9817          0
```

MEMORY_SIZE_FACTORが1の行には、MEMORY_TARGET初期化パラメータで設定されたメモリーの現行サイズ、および現行のワークロードを完了するために必要なDB時間の量が示されています。上下の行の結果には、いくつかのMEMORY_TARGETの代替サイズが示されています。各代替サイズについて、サイズ・ファクタ(現行サイズの乗数)、MEMORY_TARGETパラメータが代替サイズに変更された場合に現行のワークロードを完了するために必要な見積DB時間が示されます。合計メモリー・サイズがMEMORY_TARGETの現行サイズより小さい場合は、見積DB時間が増加していることに注意してください。また、この例では、合計メモリー・サイズを450MBより大きくしても効果がないことに注意してください。ただし、ワークロード全体がまだ実行されていない場合、この状況は変わる可能性があります。

EM Expressには、MEMORY_TARGETの最適なサイズを選択できる使いやすいグラフィカル・メモリー・アドバイザが用意されています。詳細は、『Oracle Database 2日でデータベース管理者』を参照してください。

関連項目:

- V$MEMORY_DYNAMIC_COMPONENTS動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- V$MEMORY_TARGET_ADVICE動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- DB時間の定義については、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: 自動メモリー管理の使用

6.4 メモリーの手動構成

個々のメモリー・コンポーネントのサイズをより直接的に制御する場合は、自動メモリー管理を使用禁止にして、手動メモリー管理用にデータベースを構成できます。

- 手動メモリー管理について
  手動でメモリーを管理するには、SGA用に2つ、およびインスタンスPGA用に2つの異なる方法があります。
- 自動共有メモリー管理の使用

自動共有メモリー管理により、SGAメモリー管理が簡単になります。

- 手動共有メモリー管理の使用
  共有メモリーを手動で管理するには、最初に、自動メモリー管理と自動共有メモリー管理がどちらも無効になっていることを確認します。その次に、メモリー・コンポーネントを手動で構成、監視およびチューニングします。
- 自動PGAメモリー管理の使用
  デフォルトでは、インスタンスPGA専用のメモリーの合計量を、Oracle Databaseが自動的に、グローバルに管理します。この量は、初期化パラメータPGA_AGGREGATE_TARGETを設定することによって制御できます。
- 手動PGAメモリー管理の使用
  Oracle Databaseは、手動でSQL作業領域をチューニングする、手動によるPGAメモリー管理をサポートします。

親トピック: メモリーの管理

6.4.1 手動メモリー管理について

SGAおよびインスタンスPGAについて、それぞれ2つの異なる手動メモリー管理方法があります。

SGAの2つの手動メモリー管理方法では、DBAに要求される作業量および知識量が異なります。自動共有メモリー管理では、SGAのターゲット・サイズと最大サイズを設定します。設定後、データベースによって、SGAの合計サイズが指定したターゲットに適合するように設定され、多数のSGAコンポーネントのサイズが動的にチューニングされます。手動共有メモリー管理では、複数の個別SGAコンポーネントのサイズを設定することで、SGA全体のサイズを決定します。その後、個別SGAコンポーネントを継続的に手動でチューニングします。

インスタンスPGAに関しては、自動PGAメモリー管理があり、インスタンスPGAのターゲット・サイズを設定します。設定後、データベースによって、インスタンスPGAのサイズが指定したターゲットに適合するように設定され、個々のPGAのサイズが動的にチューニングされます。また、手動PGAメモリー管理では、各タイプのSQL演算子(ソートやハッシュ結合など)のタイプごとに作業領域の最大サイズを設定します。このメモリー管理方法は、サポートされていますがお薦めしません。

関連項目:

Oracle Databaseのメモリー管理方法の概要は、『Oracle Database概要』を参照してください。

親トピック: メモリーの手動構成

6.4.2 自動共有メモリー管理の使用

自動共有メモリー管理によってSGAメモリー管理が簡素化されます。

- 自動共有メモリー管理について
  自動共有メモリー管理では、SGA_TARGET初期化パラメータを使用してインスタンスに使用可能なSGAメモリーの合計量を指定し、メモリーを最も効率的に使用できるようにするために、Oracle Databaseが様々なSGAコンポーネントにこのメモリーを自動的に配分します。
- SGAのコンポーネントおよびグラニュル
  SGAは、メモリー割当て要求の特定のクラスを満たすために使用されるメモリーのプールであるいくつかのメモリー・コンポーネントで構成されます。
- 最大SGAサイズの設定
  SGA_MAX_SIZE初期化パラメータは、インスタンスの存続期間のためのシステム・グローバル領域の最大サイズを指定します。
- SGAターゲット・サイズの設定

自動共有メモリー管理機能を有効にするには、SGA_TARGET初期化パラメータを0 (ゼロ)以外の値に設定します。このパラメータにより、SGAの合計サイズが設定されます。これは、個別のコンポーネントのセットの割当てメモリーを制御するパラメータにかわるもので、これらは必要に応じて自動的かつ動的にサイズ調整(チューニング)されるようになりました。

- 自動共有メモリー管理を使用可能にする方法
  自動共有メモリー管理(ASMM)を使用可能にする手順は、ASMMへの変更を、手動共有メモリー管理または自動メモリー管理から行うかどうかによって異なります。
- 自動的にサイズ調整されるSGAコンポーネントの最小値の設定
  自動的にサイズ調整されるSGAコンポーネントのサイズは、これらのコンポーネントに対応するパラメータの最小値を指定することにより、いくらかの制御を行使できます。この設定は、特定のコンポーネントで最低限のメモリーが確保されていないとアプリケーションが正しく動作しないことがわかっている場合に便利です。
- SGA_TARGETの動的変更
  SGA_TARGETパラメータは、SGA_MAX_SIZEパラメータに指定された値まで動的に増加でき、少なくすることもできます。
- 自動的にサイズ調整されるコンポーネントのパラメータの変更
  自動共有メモリー管理が有効な場合、自動的にサイズ調整されるコンポーネントに手動で指定したサイズは、コンポーネントのサイズの下限としての役割を果たします。この制限は、対応するパラメータの値を変更することで動的に変更できます。
- 手動でサイズ調整するコンポーネントのパラメータの変更
  手動でサイズ調整するコンポーネントのパラメータも、動的に変更できます。ただし、最小サイズは設定しないで、パラメータの値によって、対応するコンポーネントの正確なサイズを指定します。

関連項目:

- SGAのコンポーネントのチューニング方法の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: メモリーの手動構成

6.4.2.1 自動共有メモリー管理について

自動共有メモリー管理では、SGA_TARGET初期化パラメータを使用してインスタンスに使用可能なSGAメモリーの合計量を指定し、メモリーを最も効率的に使用できるようにするために、Oracle Databaseが様々なSGAコンポーネントにこのメモリーを自動的に配分します。

自動共有メモリー管理が使用可能な場合、様々なSGAコンポーネントのサイズは自由に変更され、ワークロードのニーズにあわせてチューニングされるため、追加構成の必要はありません。データベースによって、必要に応じて使用可能なメモリーが様々なコンポーネントに自動的に配分されるため、システムでは使用可能なすべてのSGAメモリーを最大限に使用できます。

サーバー・パラメータ・ファイル(SPFILE)を使用している場合、データベースでは、自動チューニングされたSGAコンポーネントのサイズがインスタンス停止後も保持されます。このため、データベース・インスタンスでは、インスタンスが起動するたびにワークロードの特性を再度認識する必要がありません。インスタンスでは、以前のインスタンスの情報に基づいて、前回インスタンスが停止した時点でのワークロードを継続して評価できます。

親トピック: 自動共有メモリー管理の使用

6.4.2.2 SGAのコンポーネントおよびグラニュル

SGAは、特定のクラスのメモリー割当て要求を満たすために使用されるメモリーのプールであるいくつかのメモリー・コンポーネントで構成されます。

メモリー・コンポーネントの例として、共有プール(SQLおよびPL/SQL実行のメモリー割当てに使用)、Javaプール(Javaオブジェクトおよびその他のJava実行メモリーに使用)、およびバッファ・キャッシュ(キャッシュ・ディスク・ブロックに使用)などがあります。すべてのSGAコンポーネントがグラニュルという単位で領域を割当ておよび割当て解除します。Oracle Databaseは、各SGAコンポーネントに対するグラニュルの内部数でSGAメモリーの使用状況を追跡します。

SGAの動的コンポーネント用のメモリーは、グラニュル単位で割り当てられます。グラニュル・サイズは、インスタンスが起動するときに要求されるSGAメモリーの量によって決まります。グラニュル・サイズは、SGA_MAX_SIZE初期化パラメータの値に基づいています。表6-1は、SGAメモリーの様々な量でのグラニュル・サイズを示しています。

表6-1 グラニュル・サイズ

SGAメモリーの量	グラニュル・サイズ
1GB 以下	4MB
1GB より大きく 8GB 以下	16MB
8GB より大きく 16GB 以下	32MB
16GB より大きく 32GB 以下	64MB
32GB より大きく 64GB 以下	128MB
64GB より大きく 128GB 以下	256MB
128GB よりも大きい	512MB

一部にプラットフォーム依存性が存在する場合があります。詳細は、使用しているオペレーティング・システム固有のマニュアルを参照してください。

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

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

親トピック: 自動共有メモリー管理の使用

6.4.2.3 最大SGAサイズの設定

インスタンスの存続期間中のシステム・グローバル領域の最大サイズは、SGA_MAX_SIZE初期化パラメータによって決まります。

システム・グローバル領域の最大サイズを設定するには:

- SGA_MAX_SIZE初期化パラメータを設定します。

バッファ・キャッシュ、共有プール、ラージ・プール、JavaプールおよびStreamsプールのサイズに影響を与える初期化パラメータは動的に変更できますが、これらのサイズとSGAの他のコンポーネント(固定SGA、可変SGAおよびREDOログ・バッファ)のサイズとの合計が、SGA_MAX_SIZEで指定された値を超えるような変更はできません。

SGA_MAX_SIZEが指定されていない場合は、初期化時に指定またはデフォルト設定されたコンポーネントすべての合計がデフォルト値として選択されます。SGA_MAX_SIZEに指定した値が、データベースの初期化時に、すべてのコンポーネントに対してパラメータ・ファイルで明示的にまたはデフォルトで割り当てたメモリーの合計より少ない場合、SGA_MAX_SIZEの設定は無視され、このパラメータに対する適切な値が選択されます。

親トピック: 自動共有メモリー管理の使用

6.4.2.4 SGAターゲット・サイズの設定

自動共有メモリー管理機能を有効にするには、SGA_TARGET初期化パラメータを0(ゼロ)以外の値に設定します。このパラメータにより、SGAの合計サイズが設定されます。これは、個別のコンポーネントのセットの割当てメモリーを制御するパラメータにかわるもので、これらは必要に応じて自動的かつ動的にサイズ調整(チューニング)されるようになりました。

自動共有メモリー管理機能を有効にするには:

- SGA_TARGET初期化パラメータをゼロ以外の値に設定します。

ノート:

- 自動共有メモリー管理が機能するには、STATISTICS_LEVEL 初期化パラメータが TYPICAL(デフォルト)または ALL に設定されている必要があります。

- より簡単に自動共有メモリー管理を使用可能にするには、EM Express を使用します。自動共有メモリー管理を使用可能にし、合計 SGA サイズを設定すると、EM Express によって ALTER SYSTEM 文が自動的に生成され、SGA_TARGET が指定したサイズに設定され、自動サイズ設定 SGA コンポーネントがすべてゼロに設定されます。詳細は、『Oracle Database 2 日でデータベース管理者』を参照してください。

SQL*Plus を使用して SGA_TARGET を設定する場合は、自動的にサイズ調整される SGA コンポーネントを 0(ゼロ)または最小値に設定する必要があります。

- SGAターゲットと自動的にサイズ調整されるSGAコンポーネント
  SGA_TARGETを設定すると、いくつかのSGAコンポーネントが自動的にサイズ調整されます。
- SGAと仮想メモリー
  ほとんどのシステムで最適なパフォーマンスを実現するには、SGA全体が実メモリーに収まる必要があります。実メモリーに収まらず、その一部を格納するために仮想メモリーが使用される場合は、データベース・システム全体のパフォーマンスが大幅に低下する可能性があります。これは、オペレーティング・システムによってSGAの一部でページング(ディスクの読取りおよび書込み)が実行されるためです。
- SGAターゲット・サイズの監視とチューニング
  V$SGAINFOビューは、様々なSGAコンポーネントの現在のチューニング・サイズに関する情報を提供します。
  V$SGA_TARGET_ADVICEビューでは、SGA_TARGETの値の決定に役立つ情報が提供されます。

親トピック: 自動共有メモリー管理の使用

6.4.2.4.1 SGAターゲットおよび自動サイズ設定SGAコンポーネント

SGA_TARGETを設定すると、いくつかのSGAコンポーネントが自動的にサイズ調整されます。

次の表に、SGA_TARGETを設定すると自動的にサイズ調整されるSGAコンポーネントを示します。各SGAコンポーネントについては、関連する初期化パラメータを示します。

表6-2 自動サイズ設定SGAコンポーネントと対応するパラメータ

SGAコンポーネント	初期化パラメータ
固定 SGA および Oracle Database インスタンスに必要なその他の内部割当て	該当なし
共有プール	SHARED_POOL_SIZE
ラージ・プール	LARGE_POOL_SIZE
Java プール	JAVA_POOL_SIZE
バッファ・キャッシュ	DB_CACHE_SIZE
Streams プール	STREAMS_POOL_SIZE

表6-3にリストされているパラメータは、設定された場合、SGA_TARGETからメモリーを取得しますが、表6-2にリストされたコンポーネントに使用されるメモリーは残されます。

表6-3 SGA_TARGETの領域を使用する手動サイズ設定SGAコンポーネント

SGAコンポーネント	初期化パラメータ
ログ・バッファ	LOG_BUFFER
KEEP バッファ・キャッシュおよび RECYCLE バッファ・キャッシュ	DB_KEEP_CACHE_SIZE DB_RECYCLE_CACHE_SIZE
非標準ブロック・サイズ・バッファ・キャッシュ	DB_nK_CACHE_SIZE

SGA_TARGETを0(ゼロ)以外の値に設定する以外に、自動サイズ設定SGAコンポーネントの完全な自動チューニングを使用可能にするために、表6-2に示されているすべての初期化パラメータを0(ゼロ)に設定する必要があります。

または、自動サイズ設定SGAコンポーネントを0(ゼロ)以外の値に設定すると、その値がSGAチューニング中のコンポーネントの最小設定として使用されます。詳細は後に説明します。

親トピック: SGAターゲット・サイズの設定

6.4.2.4.2 SGAおよび仮想メモリー

ほとんどのシステムで最適なパフォーマンスを実現するには、SGA全体が実メモリーに収まる必要があります。実メモリーに収まら

ず、その一部を格納するために仮想メモリーが使用される場合は、データベース・システム全体のパフォーマンスが大幅に低下する可能性があります。これは、オペレーティング・システムによってSGAの一部でページング(ディスクの読取りおよび書込み)が実行されるためです。

ページング・アクティビティを監視する方法は、使用しているオペレーティング・システムのマニュアルを参照してください。Cloud Controlを使用して、ページング・アクティビティを表示することもできます。詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: SGAターゲット・サイズの設定

6.4.2.4.3 SGAターゲット・サイズの監視およびチューニング

V$SGAINFOビューでは、様々なSGAコンポーネントの現行のチューニング・サイズに関する情報が提供されます。V$SGA_TARGET_ADVICEビューでは、SGA_TARGETの値の決定に役立つ情報が提供されます。

SGAターゲット・サイズを監視およびチューニングするには:

- V$SGAINFOおよびV$SGA_TARGET_ADVICEビューを問い合せます。

たとえば、次の問合せを実行します。

```
SQL> select * from v$sga_target_advice order by sga_size;

  SGA_SIZE SGA_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR ESTD_PHYSICAL_READS
---------- --------------- ------------ ------------------ ------------------
       290              .5       448176             1.6578            1636103
       435             .75       339336             1.2552            1636103
       580               1       270344                  1            1201780
       725            1.25       239038              .8842             907584
       870             1.5       211517              .7824             513881
      1015            1.75       201866              .7467             513881
      1160               2       200703              .7424             513881
```

このビューの情報は、自動メモリー管理用のV$MEMORY_TARGET_ADVICEビューで提供される情報と同じです。このビューについては、「自動メモリー管理の監視およびチューニング」を参照してください。

EM Expressには、SGA_TARGETの最適なサイズを選択できる使いやすいグラフィカル・メモリー・アドバイザが用意されています。詳細は、『Oracle Database 2日でデータベース管理者』を参照してください。

関連項目:

- V$SGAINFOビューの詳細は、『Oracle Databaseリファレンス』を参照してください
- V$SGA_TARGET_ADVICEビューの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: SGAターゲット・サイズの設定

6.4.2.5 自動共有メモリー管理を使用可能にする方法

自動共有メモリー管理(ASMM)を使用可能にする手順は、手動共有メモリー管理からASMMに変更するか、自動メモリー管理からASMMに変更するかによって異なります。

手動共有メモリー管理からASMMに変更するには:

1. 次の問合せを実行してSGA_TARGETの値を取得します。

```
SELECT (
   (SELECT SUM(value) FROM V$SGA) -
```

```
   (SELECT CURRENT_SIZE FROM V$SGA_DYNAMIC_FREE_MEMORY)
   ) "SGA_TARGET"
FROM DUAL;
```

2. テキスト形式の初期化パラメータ・ファイルを編集してデータベースを再起動するか、または次の文を発行してSGA_TARGETの値を設定します。

```
ALTER SYSTEM SET SGA_TARGET=value [SCOPE={SPFILE|MEMORY|BOTH}]
```

   valueはステップ1で計算した値、またはすべてのSGAコンポーネント・サイズの合計とSGA_MAX_SIZEの間の値です。ALTER SYSTEM文とそのSCOPE句の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

3. 次のいずれかの操作を行います。
   - より完全な自動チューニングを行うには、表6-2に記載されている自動サイズ設定SGAコンポーネントの値を0(ゼロ)に設定します。このためには、テキスト形式の初期化パラメータ・ファイルを編集するか、またはALTER SYSTEM文を発行します。
   - 1つ以上の自動サイズ設定SGAコンポーネントの最小サイズを制御するには、これらのコンポーネントのサイズを目的の値に設定します。(詳細は、次の項を参照してください。)その他の自動サイズ設定SGAコンポーネントの値を0(ゼロ)に設定します。このためには、テキスト形式の初期化パラメータ・ファイルを編集するか、またはALTER SYSTEM文を発行します。

自動メモリー管理からASMMに変更するには:

1. MEMORY_TARGET初期化パラメータを0(ゼロ)に設定します。

```
ALTER SYSTEM SET MEMORY_TARGET = 0;
```

   データベースによって、現在のSGAメモリー割当てに基づいてSGA_TARGETが設定されます。

2. 次のいずれかを行います:
   - より完全な自動チューニングを行うには、表6-2に記載されている自動サイズ設定SGAコンポーネントのサイズを0(ゼロ)に設定します。このためには、テキスト形式の初期化パラメータ・ファイルを編集するか、またはALTER SYSTEM文を発行します。
   - 1つ以上の自動サイズ設定SGAコンポーネントの最小サイズを制御するには、これらのコンポーネントのサイズを目的の値に設定します。(詳細は、次の項を参照してください。)その他の自動サイズ設定SGAコンポーネントのサイズを0(ゼロ)に設定します。このためには、テキスト形式の初期化パラメータ・ファイルを編集するか、またはALTER SYSTEM文を発行します。

例6-1 ASMMの使用

たとえば、手動共有メモリー管理用にインスタンスのパラメータが現在次のように構成されていて、SGA_MAX_SIZEが1200Mに設定されているとします。

- SHARED_POOL_SIZE = 200M
- DB_CACHE_SIZE = 500M
- LARGE_POOL_SIZE = 200M

また、問合せが次のような結果であるとします。

問合せ	結果

問合せ	結果
SELECT SUM(value) FROM V$SGA	1200M
SELECT CURRENT_SIZE FROM V$SGA_DYNAMIC_FREE_MEMORY	208M

次の文を発行することによって、自動共有メモリー管理を活用できます。

```
ALTER SYSTEM SET SGA_TARGET = 992M;
ALTER SYSTEM SET SHARED_POOL_SIZE = 0;
ALTER SYSTEM SET LARGE_POOL_SIZE = 0;
ALTER SYSTEM SET JAVA_POOL_SIZE = 0;
ALTER SYSTEM SET DB_CACHE_SIZE = 0;
ALTER SYSTEM SET STREAMS_POOL_SIZE = 0;
```

992Mは、1200M - 208Mから算出された値です。

親トピック: 自動共有メモリー管理の使用

6.4.2.6 自動サイズ設定SGAコンポーネントの最小値の設定

自動サイズ設定SGAコンポーネントに対応するパラメータに最小値を指定して、これらのコンポーネントのサイズをある程度制御できます。この設定は、特定のコンポーネントで最低限のメモリーが確保されていないとアプリケーションが正しく動作しないことがわかっている場合に便利です。

コンポーネントのためのSGA領域の最小値を指定するには:

- 対応する初期化パラメータの値を設定します。

1つ以上の自動サイズ設定コンポーネントの最小サイズを手動で制限すると、動的な調整に使用されるメモリーの合計量が減少します。この量が減少すると、システムがワークロードの変動に適応するシステムの機能が次々に制限されます。したがって、この方法は特別な場合を除いてお薦めしません。デフォルトの自動管理は、システム・パフォーマンスおよび使用可能なリソースの使用が最大になるように動作します。

関連トピック

- SGAターゲットおよび自動サイズ設定SGAコンポーネント

親トピック: 自動共有メモリー管理の使用

6.4.2.7 SGA_TARGETの動的変更

SGA_TARGETパラメータは、SGA_MAX_SIZEパラメータに指定された値まで動的に増やすことができ、減らすこともできます。

SGA_TARGETの値を減らした場合は、メモリーを解放する1つ以上の自動調整コンポーネントがシステムによって識別されます。SGA_TARGETは、1つ以上の自動調整コンポーネントがその最小サイズに達するまで減らすことができます。Oracle Databaseでは、SGA_TARGETの最小許容値を決定するときに、自動サイズ設定コンポーネントに設定された値、SGA_TARGET領域を使用する手動サイズ設定コンポーネント、CPUの数など、いくつかの要因が考慮されます。

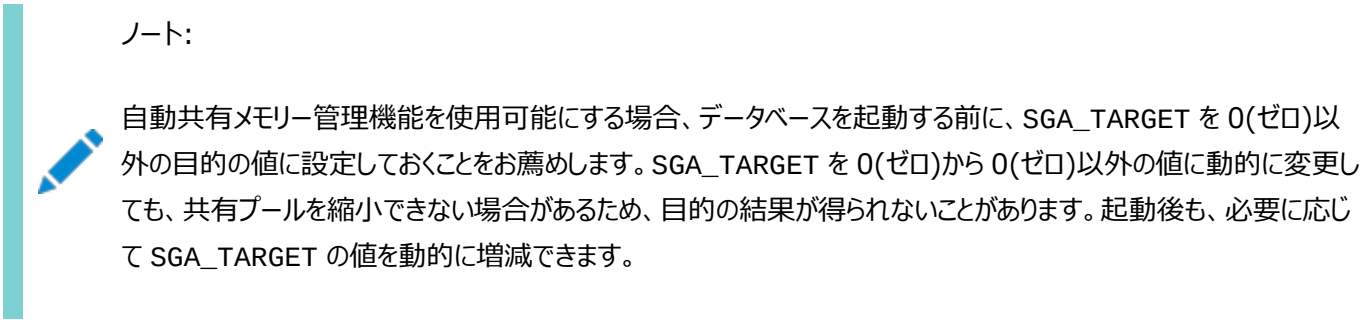
SGA_TARGETが変更されたときに、消費される物理メモリーの量の変更は、オペレーティング・システムによって異なります。動的共有メモリーをサポートしない一部のUNIXプラットフォームでは、SGAで使用される物理メモリーはSGA_MAX_SIZEパラメータの値と同じです。SGA_TARGETをSGA_MAX_SIZEより小さい値に設定した場合、このようなプラットフォームでは実際の利点はありません。したがって、これらのプラットフォームでは、SGA_MAX_SIZEを設定しないことをお薦めします。

SolarisやWindowsなどの他のプラットフォームでは、SGAで消費される物理メモリーはSGA_TARGETと同じです。

たとえば、次のような構成環境とします。

- SGA_MAX_SIZE = 1024M
- SGA_TARGET = 512M
- DB_8K_CACHE_SIZE = 128M

この例では、SGA_TARGETは1024Mまでサイズを増やすことができ、自動サイズ設定コンポーネントの1つ以上がその最小サイズに達するまでサイズを減らすこともできます。正確な値は、システムのCPU数などの環境要因によって決定します。ただし、DB_8K_CACHE_SIZEの値は常に128Mで固定です。



ノート:

自動共有メモリー管理機能を使用可能にする場合、データベースを起動する前に、SGA_TARGET を 0(ゼロ)以外の目的の値に設定しておくことをお薦めします。SGA_TARGET を 0(ゼロ)から 0(ゼロ)以外の値に動的に変更しても、共有プールを縮小できない場合があるため、目的の結果が得られないことがあります。起動後も、必要に応じて SGA_TARGET の値を動的に増減できます。

親トピック: 自動共有メモリー管理の使用

6.4.2.8 自動サイズ設定コンポーネントのパラメータの変更

自動共有メモリー管理が有効な場合、自動的にサイズ調整されるコンポーネントに手動で指定したサイズは、コンポーネントのサイズの下限としての役割を果たします。この制限は、対応するパラメータの値を変更することで動的に変更できます。

特定のSGAコンポーネントに指定したサイズの下限が現在のサイズより小さい場合、そのコンポーネントのサイズはすぐには変更されません。新しい設定は自動チューニング・アルゴリズムを将来において減少した最小サイズに制限するのみです。

コンポーネント・サイズの下限を設定するには:

- コンポーネントの初期化パラメータを最小値に設定します。

たとえば、次のような構成を考えてみます。

- SGA_TARGET = 512M
- LARGE_POOL_SIZE = 256M
- 現在の実際のラージ・プール・サイズ = 284M

この例では、LARGE_POOL_SIZEの値をコンポーネントの実際の現在のサイズよりも大きい値に増加すると、増加した最小サイズに対応するように、システムによってコンポーネントが拡大されます。たとえば、LARGE_POOL_SIZEの値を300Mに増加すると、ラージ・プールが300Mに達するまで増分的に増加されます。このサイズ変更は、1つ以上の自動調整されたコンポーネントを犠牲にして行われます。LARGE_POOL_SIZEの値を200に減らした場合、そのコンポーネントのサイズはすぐに変更されません。新しい設定は、後のラージ・プール・サイズの削減のみを200Mに制限します。

ノート:

SGA_TARGET が設定されていない場合、自動共有メモリー管理機能は使用できません。したがって、すべてのコ

ンポーネント・パラメータのサイズ変更に関するルールは、以前のリリースの場合と同じです。

親トピック: 自動共有メモリー管理の使用

6.4.2.9 手動サイズ設定コンポーネントのパラメータの変更

手動サイズ設定コンポーネントのパラメータも動的に変更できます。ただし、最小サイズは設定しないで、パラメータの値によって、対応するコンポーネントの正確なサイズを指定します。

手動サイズ設定コンポーネントのサイズを増やすと、余分なメモリーが1つ以上の自動サイズ設定コンポーネントから削除されます。手動サイズ設定コンポーネントのサイズを減らすと、解放されたメモリーが複数の自動サイズ設定コンポーネントに提供されます。

コンポーネントの正確なサイズを変更するには:

- コンポーネントの初期化パラメータを設定します。

たとえば、次のような構成を考えてみます。

- SGA_TARGET = 512M
- DB_8K_CACHE_SIZE = 128M

この例のDB_8K_CACHE_SIZEを16M増やして144Mにすると、その16Mが、自動サイズ設定コンポーネントから削除されます。同様に、DB_8K_CACHE_SIZEを16M減らして112Mにすると、その16Mが、自動サイズ設定コンポーネントに提供されます。

親トピック: 自動共有メモリー管理の使用

6.4.3 手動共有メモリー管理の使用

共有メモリーを手動で管理するには、最初に、自動メモリー管理と自動共有メモリー管理がどちらも無効になっていることを確認します。その次に、メモリー・コンポーネントを手動で構成、監視およびチューニングします。

- 手動共有メモリー管理について
  自動メモリー管理または自動共有メモリー管理を使用しない場合は、いくつかのSGAコンポーネントのサイズを手動で構成し、データベース・ワークロードの変更に基づいてこれらのサイズを監視およびチューニングする必要があります。これらのSGAコンポーネントのサイズを制御するパラメータの設定のガイドラインに従うことができます。
- 手動共有メモリー管理を使用可能にする方法
  それ自体で手動共有メモリー管理を使用可能にする初期化パラメータはありません。自動メモリー管理と自動共有メモリー管理の両方を使用禁止にすることによって、手動共有メモリー管理が事実上使用可能になります。
- バッファ・キャッシュ初期化パラメータの設定
  SGAのバッファ・キャッシュ・コンポーネントのサイズは、バッファ・キャッシュ初期化パラメータによって決まります。
- 共有プール・サイズの指定
  SHARED_POOL_SIZE初期化パラメータは、SGAの共有プール・コンポーネントのサイズを指定または調整できる動的パラメータです。Oracle Databaseによって適切なデフォルト値が選択されます。
- ラージ・プール・サイズの指定
  LARGE_POOL_SIZE初期化パラメータは、SGAのラージ・プール・コンポーネントのサイズを指定または調整できる動的パラメータです。
- Javaプール・サイズの指定
  JAVA_POOL_SIZE初期化パラメータは、SGAのJavaプール・コンポーネントのサイズを指定または調整できる動的パラメータです。

- Streamsプール・サイズの指定
  STREAMS_POOL_SIZE初期化パラメータは、SGAのStreamsプール・コンポーネントのサイズを指定または調整できる動的パラメータです。
- 結果キャッシュの最大サイズの指定
  RESULT_CACHE_MAX_SIZE初期化パラメータは、SGAの結果キャッシュ・コンポーネントの最大サイズを指定できる動的なパラメータです。
- その他のSGA初期化パラメータの指定
  いくつかの追加の初期化パラメータを設定して、SGAのメモリーの使用方法を制御できます。

親トピック: メモリーの手動構成

6.4.3.1 手動共有メモリー管理について

自動メモリー管理または自動共有メモリー管理を使用しない場合は、いくつかのSGAコンポーネントのサイズを手動で構成し、データベース・ワークロードの変更に基づいてこれらのサイズを監視およびチューニングする必要があります。これらのSGAコンポーネントのサイズを制御するパラメータの設定のガイドラインに従うことができます。

DBCAを使用してデータベースを作成し、手動共有メモリー管理を選択した場合、バッファ・キャッシュ、共有プール、ラージ・プールおよびJavaプールのサイズを入力する必須フィールドがDBCAで表示されます。次に、対応する初期化パラメータが、作成されるサーバー・パラメータ・ファイル(SPFILE)に設定されます。かわりにデータベースをCREATE DATABASE SQL文とテキスト初期化パラメータ・ファイルで作成した場合、次のいずれかを実行します。

- SGAコンポーネント・サイズを設定する初期化パラメータの値を指定します。
- テキスト形式の初期化ファイルからSGAコンポーネント・サイズのパラメータを削除します。サイズを設定していないコンポーネントに対しては、Oracle Databaseによって適切なデフォルトが選択されます。

親トピック: 手動共有メモリー管理の使用

6.4.3.2 手動共有メモリー管理を使用可能にする方法

手動共有メモリー管理自体を使用可能にする初期化パラメータはありません。自動メモリー管理と自動共有メモリー管理の両方を使用禁止にすることによって、手動共有メモリー管理が事実上使用可能になります。

手動共有メモリー管理を使用可能にするには:

1. MEMORY_TARGET初期化パラメータを0(ゼロ)に設定します。
2. SGA_TARGET初期化パラメータを0(ゼロ)に設定します。

次に、以降の項の説明に従って、様々なSGAコンポーネントの値を設定する必要があります。

親トピック: 手動共有メモリー管理の使用

6.4.3.3 バッファ・キャッシュ初期化パラメータの設定

SGAコンポーネントであるバッファ・キャッシュのサイズは、バッファ・キャッシュ初期化パラメータによって決まります。

これらのパラメータを使用して、データベースで使用される各ブロック・サイズのキャッシュ・サイズを指定します。これらの初期化パラメータはすべて動的です。

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

Oracle Databaseでは、データベース内で複数のブロック・サイズがサポートされます。非標準のブロック・サイズを設定した表領域を作成する場合は、これらの表領域を格納するための非標準のブロック・サイズ・バッファを構成する必要があります。SYSTEM表領域には標準のブロック・サイズが使用されます。標準のブロック・サイズを指定するには、初期化パラメータDB_BLOCK_SIZEを設定します。指定できる値は2Kから32Kです。

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

非標準のブロック・サイズ・バッファのサイズおよび数は、次のパラメータによって指定します。

```
DB_2K_CACHE_SIZE
DB_4K_CACHE_SIZE
DB_8K_CACHE_SIZE
DB_16K_CACHE_SIZE
DB_32K_CACHE_SIZE
```

各パラメータは、対応するブロック・サイズのキャッシュ・サイズを指定します。

ノート:

- 最大ブロック・サイズに関するプラットフォーム固有の制限が適用されるため、プラットフォームによっては、これらのサイズの一部は指定できない場合があります。
- 32KB のブロック・サイズは、64 ビットのプラットフォームでのみ有効です。

- ブロック・サイズおよびキャッシュ・サイズの設定例
  ブロック・サイズおよびキャッシュ・サイズの設定の例を示します。
- 複数のバッファ・プール
  異なるバッファ・プールを持つデータベース・バッファ・キャッシュを構成して、バッファ・キャッシュ内にデータを保持するか、またはデータ・ブロックの使用直後に新しいデータがバッファを使用できるようにするかを指定できます。

関連項目:

表領域の非標準のブロック・サイズの指定

親トピック: 手動共有メモリー管理の使用

6.4.3.3.1 ブロック・サイズおよびキャッシュ・サイズの設定例

ブロック・サイズおよびキャッシュ・サイズの設定の例を示します。

```
DB_BLOCK_SIZE=4096
DB_CACHE_SIZE=1024M
DB_2K_CACHE_SIZE=256M
DB_8K_CACHE_SIZE=512M
```

この例では、パラメータDB_BLOCK_SIZEは、データベースの標準ブロック・サイズを4Kに設定しています。標準ブロック・サイズ・バッファのキャッシュ・サイズは1024MBです。また、2KBおよび8KBのキャッシュ・サイズがそれぞれ256MBと512MBで構成されます。

ノート:

DB_nK_CACHE_SIZE パラメータを使用して、標準ブロック・サイズのキャッシュ・サイズを指定することはできません。DB_BLOCK_SIZE の値が nKB の場合に DB_nK_CACHE_SIZE を設定しても無効です。標準ブロック・サイズのキャッシュ・サイズは、常に DB_CACHE_SIZE の値によって決まります。

キャッシュのサイズは制限されているため、ディスク上のすべてのデータをキャッシュに格納することはできません。キャッシュが一杯の場合、その後キャッシュ・ミスが発生すると、新しいデータ用の領域を空けるために、Oracle Databaseはキャッシュにすでに存在する使用済データをディスクに書き込みます。(バッファが使用済でない場合、新しいブロックをバッファに読み込む前にディスクに書き込む必要はありません。)ディスクに書き込まれ、上書きされた結果がその後アクセスされると、さらにキャッシュ・ミスが発生します。

キャッシュのサイズは、データを要求したときにキャッシュ・ヒットになる確率に影響します。キャッシュが大きい場合は、要求されたデータがキャッシュに入っている可能性が高くなります。キャッシュのサイズを大きくすると、データ要求がキャッシュ・ヒットになる確率が高くなります。

インスタンスの実行中も、データベースを停止せずにバッファ・キャッシュのサイズを変更できます。この操作にはALTER SYSTEM文を使用します。

個々のキャッシュ・コンポーネントのサイズおよび保留中のサイズ変更操作を追跡するには、固定ビューV$BUFFER_POOLを使用します。

親トピック: バッファ・キャッシュ初期化パラメータの設定

6.4.3.3.2 複数バッファ・プール

異なるバッファ・プールを持つデータベース・バッファ・キャッシュを構成して、バッファ・キャッシュ内にデータを保持するか、またはデータ・ブロックの使用直後に新しいデータがバッファを使用できるようにするかを指定できます。

その後、特定のスキーマ・オブジェクト(表、クラスタ、索引およびパーティション)を適切なバッファ・プールに割り当て、キャッシュからデータ・ブロックをエージ・アウトする方法を制御できます。

- KEEPバッファ・プールでは、スキーマ・オブジェクトのデータ・ブロックがメモリーに保持されます。
- RECYCLEバッファ・プールでは、データ・ブロックが不要になるとすぐにメモリーから除去されます。
- DEFAULTバッファ・プールには、いずれのバッファ・プールにも割り当てられていないスキーマ・オブジェクトのデータ・ブロックと、明示的にDEFAULTプールに割り当てられたスキーマ・オブジェクトのデータ・ブロックが含まれます。

KEEPバッファ・プールとRECYCLEバッファ・プールを構成する初期化パラメータは、DB_KEEP_CACHE_SIZEとDB_RECYCLE_CACHE_SIZEです。

ノート:

複数バッファ・プールは、標準ブロック・サイズに対してのみ使用可能です。非標準ブロック・サイズのキャッシュには、1 つの DEFAULT プールのみ使用できます。

関連項目:

バッファ・キャッシュのチューニング方法、および複数バッファ・プールの詳細は、『Oracle Databaseパフォーマンス・チューニング・

ガイド』を参照してください。

親トピック: バッファ・キャッシュ初期化パラメータの設定

6.4.3.4 共有プール・サイズの指定

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

Oracle Database 10gより前のリリースでは、割り当てられる共有プール・メモリーの量は、SHARED_POOL_SIZE初期化パラメータの値に、インスタンスの起動時に計算された内部SGAオーバーヘッドの量を加算した値と等しい値でした。内部SGAオーバーヘッドとは、他の複数の初期化パラメータの値に基づいて起動時にOracle Databaseによって割り当てられるメモリーです。このメモリーは、SGAの様々なサーバー・コンポーネントの状態を維持するために使用されます。たとえば、SHARED_POOL_SIZEパラメータが64MBに設定されていて、計算された内部SGAオーバーヘッドの値が12MBである場合、共有プールの実際のサイズは64 + 12 = 76MBですが、SHARED_POOL_SIZEパラメータの値は64MBと表示されます。

Oracle Database 10g以降では、内部SGAオーバーヘッドのサイズは、ユーザー指定のSHARED_POOL_SIZEの値に含まれます。つまり、自動メモリー管理または自動共有メモリー管理を使用していない場合、起動時に割り当てられる共有プール・メモリーの量はSHARED_POOL_SIZE初期化パラメータの値と等しくなり、グラニュル・サイズの倍数に丸められます。したがって、このパラメータには、必要な共有プール・サイズに内部SGAオーバーヘッドを加えた値を設定する必要があります。前述の例の場合、SHARED_POOL_SIZEパラメータが起動時に64MBに設定されているとすると、内部SGAオーバーヘッドの値が変わっていなければ、起動後に使用可能な共有プールの値は、64 - 12 = 52MBになります。起動後に共有プール・メモリーとして有効な値を64MB維持するには、SHARED_POOL_SIZEパラメータを64 + 12 = 76MBに設定する必要があります。

Oracle Database 10gよりも前のリリースから移行する場合、移行ユーティリティにより、このパラメータの新しい値が、アップグレード前の環境での内部SGAオーバーヘッド値とこのパラメータの以前の値に基づいて推奨されます。Oracle Database 10g以降では、SGAの内部オーバーヘッド(共有プールの起動オーバーヘッドとも呼ばれます)の正確な値をV$SGAINFOビューから問い合せることができます。また、手動共有メモリー管理モードでは、ユーザーが指定したSHARED_POOL_SIZE値が内部SGAのオーバーヘッドの要件も満たせないほど小さい場合は、Oracle Databaseの起動時にORA-00371エラーが生成され、SHARED_POOL_SIZEパラメータの推奨値が示されます。自動共有メモリー管理を使用している場合は、共有プールが自動的に調整され、ORA-00371エラーは生成されません。

- 結果キャッシュと共有プール・サイズ

親トピック: 手動共有メモリー管理の使用

6.4.3.4.1 結果キャッシュと共有プール・サイズ

結果キャッシュのメモリーは共有プールから取得されます。したがって、結果キャッシュの最大サイズを大きくする場合は、共有プールのサイズ設定時にこのサイズを考慮する必要があります。

関連項目:

「結果キャッシュの最大サイズの指定」

親トピック: 共有プール・サイズの指定

6.4.3.5 ラージ・プール・サイズの指定

LARGE_POOL_SIZE初期化パラメータは、SGAのコンポーネントであるラージ・プールのサイズを指定または調整する動的なパラメータです。

ラージ・プールは、SGAのオプションのコンポーネントです。ラージ・プールを作成する場合は、LARGE_POOL_SIZEパラメータを設定する必要があります。ラージ・プールの構成方法は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: 手動共有メモリー管理の使用

6.4.3.6 Javaプール・サイズの指定

JAVA_POOL_SIZE初期化パラメータは、SGAのJavaプール・コンポーネントのサイズを指定または調整できる動的パラメータです。

Oracle Databaseによって適切なデフォルト値が選択されます。Javaプールの構成方法は、『Oracle Database Java開発者ガイド』を参照してください。

親トピック: 手動共有メモリー管理の使用

6.4.3.7 Streamsプール・サイズの指定

STREAMS_POOL_SIZE初期化パラメータは、SGAのコンポーネントであるStreamsプールのサイズを指定または調整する動的なパラメータです。

STREAMS_POOL_SIZEが0(ゼロ)に設定されている場合、Oracle Streams製品では、メモリーが必要時にバッファ・キャッシュからStreamsプールに転送されます。

親トピック: 手動共有メモリー管理の使用

6.4.3.8 結果キャッシュの最大サイズの指定

RESULT_CACHE_MAX_SIZE初期化パラメータは、SGAの結果キャッシュ・コンポーネントの最大サイズを指定できるようにする動的なパラメータです。

通常、デフォルトの最大サイズは、SGAが使用可能な合計メモリーと使用中のメモリー管理方式に基づいてデータベースによって選択されるため、このパラメータを指定する必要はありません。現在のデフォルト最大サイズを調べるには、RESULT_CACHE_MAX_SIZEパラメータの値を表示します。この最大サイズを変更するには、ALTER SYSTEM文でRESULT_CACHE_MAX_SIZEを設定するか、テキスト初期化パラメータ・ファイルでこのパラメータを指定します。いずれの場合も、値は近似の32Kの倍数に切り上げられます。

インスタンスの起動時にRESULT_CACHE_MAX_SIZEが0(ゼロ)の場合、結果キャッシュは使用禁止です。使用可能にするには、RESULT_CACHE_MAX_SIZEを0(ゼロ)以外の値に設定して(またはテキスト形式の初期化パラメータ・ファイルからこのパラメータを削除してデフォルトの最大サイズを取得して)、データベースを再起動する必要があります。

結果キャッシュが使用禁止の状態でデータベースを起動した後、ALTER SYSTEM文を使用してRESULT_CACHE_MAX_SIZEを0(ゼロ)以外の値に設定し、その後データベースを再起動していない場合、RESULT_CACHE_MAX_SIZEパラメータの値を問い合せると、結果キャッシュが使用禁止状態のままでも0(ゼロ)以外の値が返されます。したがって、RESULT_CACHE_MAX_SIZEの値は、結果キャッシュが使用可能かどうかを判断する最も信頼できる方法ではありません。かわりに、次の問合せを使用できます。

```
SELECT dbms_result_cache.status() FROM dual;
DBMS_RESULT_CACHE.STATUS()
----------------------------------------
ENABLED
```

結果キャッシュのメモリーは共有プールから取得されるため、結果キャッシュの最大サイズを大きくする場合は、共有プールのサイズを大きくすることも考慮する必要があります。

ビューV$RESULT_CACHE_STATISTICSおよびPL/SQLパッケージ・プロシージャDBMS_RESULT_CACHE.MEMORY_REPORTでは、結果キャッシュに現在割り当てられているメモリー量の判断に役立つ情報が表示されます。

PL/SQLパッケージのファンクションDBMS_RESULT_CACHE.FLUSHでは、結果キャッシュがクリアされ、すべてのメモリーが共有プールに解放されます。

関連項目:

- 結果キャッシュの詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。
- DBMS_RESULT_CACHEパッケージのプロシージャとファンクションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- V$RESULT_CACHE_STATISTICSビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- クラスタ・データベースのRESULT_CACHE_MAX_SIZEの設定方法は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

親トピック: 手動共有メモリー管理の使用

6.4.3.9 その他のSGA初期化パラメータの指定

いくつかの追加の初期化パラメータを設定して、SGAのメモリー使用方法を制御できます。

- 物理メモリー
  LOCK_SGAパラメータをTRUEに設定すると、SGA全体が物理メモリーにロックされます。
- SGA開始アドレス
  SHARED_MEMORY_ADDRESSパラメータとHI_SHARED_MEMORY_ADDRESSは、実行時にSGAの開始アドレスを指定します。

親トピック: 手動共有メモリー管理の使用

6.4.3.9.1 物理メモリー

LOCK_SGAパラメータをTRUEに設定すると、SGA全体が物理メモリーにロックされます。

このパラメータは自動メモリー管理では使用できません。

関連項目:

- これらの初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 「自動メモリー管理の使用」
- 自動共有メモリー管理の使用

親トピック: その他のSGA初期化パラメータの指定

6.4.3.9.2 SGA開始アドレス

SHARED_MEMORY_ADDRESSパラメータとHI_SHARED_MEMORY_ADDRESSパラメータでは、実行時のSGAの開始アドレスが指定されます。

これらのパラメータはほとんど使用されません。64ビットのプラットフォームの場合、HI_SHARED_MEMORY_ADDRESSでは、64

ビット・アドレスの上位32ビットが指定されます。

関連項目:

- SHARED_MEMORY_ADDRESS初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- HI_SHARED_MEMORY_ADDRESS初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 「自動メモリー管理の使用」
- 自動共有メモリー管理の使用

親トピック: その他のSGA初期化パラメータの指定

6.4.4 自動PGAメモリー管理の使用

デフォルトで、Oracle Databaseでは、インスタンスPGA専用のメモリーの合計量が自動的かつグローバルに管理されます。この量は、初期化パラメータPGA_AGGREGATE_TARGETを設定することによって制御できます。

Oracle Databaseでは、すべてのデータベース・サーバー・プロセスとバックグラウンド・プロセスに割り当てられるPGAメモリーの合計量がこのターゲット値を超えないようにします。

DBCAを使用してデータベースを作成する場合は、合計インスタンスPGAの値を指定できます。その後、DBCAによって、作成されるサーバー・パラメータ・ファイル(SPFILE)に、PGA_AGGREGATE_TARGET初期化パラメータが設定されます。合計インスタンスPGAを指定しない場合は、DBCAによって適切なデフォルト値が選択されます。

CREATE DATABASE SQL文とテキスト形式のパラメータ・ファイルを使用してデータベースを作成する場合は、PGA_AGGREGATE_TARGETの値を指定できます。このパラメータを省略すると、データベースによってデフォルト値が選択されます。

自動PGAメモリー管理を使用すると、SQL作業領域のサイズが自動で設定され、すべての*_AREA_SIZE初期化パラメータが無視されます。特定の時期に、インスタンス上のアクティブな作業領域で使用可能なPGAメモリーの合計量がパラメータPGA_AGGREGATE_TARGETから自動的に導出されます。この量は、PGA_AGGREGATE_TARGETの値から、他の目的で割り当てられたPGAメモリー(たとえば、セッション・メモリー)を減算した値に設定されます。結果のPGAメモリーは、その特定のメモリー要件に基づいて個々のアクティブな作業領域に割り当てられます。

PGAメモリーの使用統計を提供する動的なパフォーマンス・ビューが用意されています。これらの統計のほとんどは、PGA_AGGREGATE_TARGETが設定されると使用可能になります。

- 作業領域メモリーの割当ておよび使用に関する統計は、次の動的パフォーマンス・ビューで表示できます。

  - V$SYSSTAT
  - V$SESSTAT
  - V$PGASTAT
  - V$SQL_WORKAREA
  - V$SQL_WORKAREA_ACTIVE

- V$PROCESSビューの次の3つの列では、Oracle Databaseプロセスによって割り当てられ使用されているPGAメモリーがレポートされます。

  - PGA_USED_MEM
  - PGA_ALLOC_MEM
  - PGA_MAX_MEM

PGA_AGGREGATE_TARGET設定がターゲットとなります。したがって、Oracle Databaseでは、PGAのメモリー使用量をこのターゲットに合わせて制限することを試みますが、使用量がこの設定を超える場合があります。PGAメモリー使用量について厳格な制限を指定するには、PGA_AGGREGATE_LIMIT初期化パラメータを使用します。PGAサイズがこの制限を超えないことが、Oracle Databaseによって保証されます。この制限を超えた場合、データベースは最も調整が困難なPGAメモリーを割り当てられているセッションのコールを終了します。PGA_AGGREGATE_LIMITは、自動メモリー管理を使用しているかどうかに関係なく設定できます。PGA_AGGREGATE_LIMITが設定されていない場合、Oracle Databaseによって、適切なデフォルトの制限が決定されます。このパラメータ詳細は、『Oracle Databaseリファレンス』を参照してください。

ノート:

自動 PGA メモリー管理方法は、専用および共有のサーバー・プロセスによって割り当てられた作業領域に適用されます。専用および共有サーバー・モードでの PGA メモリー割当ての詳細は、『Oracle Database 概要』を参照してください。

関連項目:

- この項で説明されている初期化パラメータおよびビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- この項で説明されているビューの使用の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: メモリーの手動構成

6.4.5 手動PGAメモリー管理の使用

Oracle Databaseでは、SQL作業領域を手動でチューニングする手動PGAメモリー管理がサポートされています。

Oracle Database 10gより前のリリースでは、データベース管理者がパラメータSORT_AREA_SIZE、HASH_AREA_SIZE、BITMAP_MERGE_AREA_SIZEおよびCREATE_BITMAP_AREA_SIZEを設定することで、SQL作業領域の最大サイズを制御していました。しかし、最大作業領域サイズは、データ入力サイズとシステム内でアクティブな作業領域の合計数から選択するのが理想であるため、これらのパラメータの設定は困難です。これら2つの要因は、作業領域および時間により大幅に変動します。このため、各種*_AREA_SIZEパラメータを最適な状況でチューニングすることは困難です。

このような理由から、自動PGAメモリー管理を使用可能にしておくことをお薦めします。

SQL作業領域を手動でチューニングする場合は、WORKAREA_SIZE_POLICY初期化パラメータをMANUALに設定する必要があります。

ノート:

初期化パラメータ WORKAREA_SIZE_POLICY は、セッション・レベルおよびシステム・レベルのパラメータで、設定できる値は MANUAL または AUTO の 2 つのみです。デフォルトは AUTO です。PGA_AGGREGATE_TARGET を設定した後、メモリー管理モードを自動と手動の間で切り替えることができます。WORKAREA_SIZE_POLICY が AUTO に設定されている場合、*_AREA_SIZE パラメータの設定は無視されます。

親トピック: メモリーの手動構成

6.5 強制フル・データベース・キャッシュ・モードの使用方法

Oracle Databaseインスタンスは、データベース全体をバッファ・キャッシュにキャッシュできます。

ノート:

この機能は、Oracle Database 12c リリース 1 (12.1.0.2)以降で使用可能です。

- 強制フル・データベース・キャッシュ・モードについて
  デフォルト・キャッシュ・モードでは、ユーザーが大きな表の問合せを行うとき、バッファ・キャッシュからより多くの有用なデータが削除される可能性があるため、Oracle Databaseは基礎となるデータを必ずしもキャッシュしません。Oracle Database 12cリリース1 (12.1.0.2)以上では、データベース全体をバッファ・キャッシュにキャッシュするのに十分な領域があり、そうすることが有益であるとOracle Databaseインスタンスが判断した場合、インスタンスはデータベース全体をバッファ・キャッシュに自動的にキャッシュします。
- 強制フル・データベース・キャッシュ・モードを有効にする前に
  データベース・インスタンスに対して強制フル・データベース・キャッシュ・モードを有効にするには、データベースは12.0.0以上の互換性レベルである必要があります。また、バッファ・キャッシュは必ずデータベース全体をキャッシュするのに十分な大きさにします。
- 強制フル・データベース・キャッシュ・モードの有効化
  データベースの強制フル・データベース・キャッシュ・モードを有効にすることができます。
- 強制フル・データベース・キャッシュ・モードの無効化
  データベースの強制フル・データベース・キャッシュ・モードを無効にすることができます。

親トピック: メモリーの管理

6.5.1 強制フル・データベース・キャッシュ・モードについて

デフォルト・キャッシュ・モードでは、ユーザーが大規模表を問い合せた場合、キャッシュすることでバッファ・キャッシュからより有用なデータが削除される可能性があるため、Oracle Databaseは基礎になるデータを必ずしもキャッシュしません。Oracle Database 12cリリース1 (12.1.0.2)以上では、データベース全体をバッファ・キャッシュにキャッシュするのに十分な領域があり、そうすることが有益であるとOracle Databaseインスタンスが判断した場合、インスタンスはデータベース全体をバッファ・キャッシュに自動的にキャッシュします。

データベース全体をバッファ・キャッシュにキャッシュすると、パフォーマンスが改善されることがあります。`ALTER DATABASE FORCE FULL DATABASE CACHING`文を使用すると、強制的にインスタンスでデータベースをバッファ・キャッシュにキャッシュさせることができます。この文は、インスタンスを強制フル・データベース・キャッシュ・モードにします。このモードでは、Oracle Databaseにより、バッファ・キャッシュはデータベース全体のキャッシュに十分な大きさであるとみなされ、それ以降にアクセスされるすべてのブロックのキャッシュが試行されます。

Oracle Databaseインスタンスが強制フル・データベース・キャッシュ・モードである場合、次の問合せではYESが返されます。

```
SELECT FORCE_FULL_DB_CACHING FROM V$DATABASE;
```

インスタンスがデフォルト・キャッシュ・モードである場合、`NOCACHE` LOBはバッファ・キャッシュにキャッシュされません。しかし、インスタンスが強制フル・データベース・キャッシュ・モードである場合、`NOCACHE` LOBはバッファ・キャッシュにキャッシュできます。また、SecureFiles LOB記憶域を使用するLOBとBasicFiles LOB記憶域を使用するLOBはいずれも、強制フル・データベース・キャッシュ・モードでのみバッファ・キャッシュにキャッシュできます。

ノート:

- インスタンスが強制フル・データベース・キャッシュ・モードになると、データベース・オブジェクトはバッファ・キャッシュにすぐにロードされません。かわりに、アクセス時に、バッファ・キャッシュにキャッシュされます。

- マルチテナント環境では、強制フル・データベース・キャッシュ・モードは、プラガブル・データベース(PDB)全部を含め、マルチテナント・コンテナ・データベース(CDB)全体に適用されます。

- 強制フル・データベース・キャッシュ・モードに関する情報は、制御ファイルに格納されます。制御ファイルを置換または再作成すると、強制フル・データベース・キャッシュ・モードに関する情報は失われます。復元された制御ファイルにこの情報が含まれるかどうかは、制御ファイルをバックアップした時期によって決まります。

関連項目:

- Oracle Multitenant管理者ガイド
- 「制御ファイルの管理」
- 強制フル・データベース・キャッシュ・モードを使用する場合の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: 強制フル・データベース・キャッシュ・モードの使用方法

6.5.2 強制フル・データベース・キャッシュ・モードを有効にする前に

データベース・インスタンスに対して強制フル・データベース・キャッシュ・モードを有効にするには、データベースは12.0.0以上の互換性レベルである必要があります。また、バッファ・キャッシュは必ずデータベース全体をキャッシュするのに十分な大きさにします。

自動メモリー管理にSGA_TARGETまたはMEMORY_TARGET初期化パラメータを使用するようにデータベースを構成する場合、バッファ・キャッシュのサイズはワークロードによって変わります。インスタンスが通常のワークロード下にあるときに、次の問合せを実行してバッファ・キャッシュ・サイズを見積ります。

```
SELECT NAME, BYTES FROM V$SGAINFO WHERE NAME='Buffer Cache Size';
```

この問合せは、あらゆる使用可能なブロック・サイズについてバッファ・キャッシュ・サイズを返します。データベースで複数のブロック・サイズを使用している場合、使用可能な各ブロック・サイズのバッファ・キャッシュ・サイズはそのブロック・サイズの合計データベース・サイズより大きくなるようにするのが最適です。

DB_nK_CACHE_SIZE初期化パラメータを指定すると、デフォルト以外のブロック・サイズのバッファ・キャッシュ・サイズを確認できます。SGA_TARGETまたはMEMORY_TARGETでは、ワークロードに応じてデフォルト・プールのデフォルト・ブロック・サイズのバッファ・キャッシュ・サイズが変更される場合があります。次の問合せは、デフォルト・プールのデフォルト・ブロック・サイズについて現在のバッファ・キャッシュ・サイズを返します。

```
SELECT COMPONENT, CURRENT_SIZE FROM V$SGA_DYNAMIC_COMPONENTS
   WHERE COMPONENT LIKE 'DEFAULT buffer cache';
```

データベース全体をバッファ・キャッシュで実行するためのメモリー要件を見積っている場合、バッファ・キャッシュのサイズを次のいずれかとして見積ることができます。

- SGA_TARGETを使用する予定の場合、バッファ・キャッシュ・サイズはSGA_TARGETの60%と見積ることができます。

- MEMORY_TARGETを使用する予定の場合、SGAサイズはMEMORY_TARGETの60%、バッファ・キャッシュ・サイズはSGAサイズの60%と見積ることができます。つまり、バッファ・キャッシュ・サイズはMEMORY_TARGETの36%と見積ることができます。

関連項目:

「自動メモリー管理の使用」

親トピック: 強制フル・データベース・キャッシュ・モードの使用方法

6.5.3 強制フル・データベース・キャッシュ・モードの有効化

データベースの強制フル・データベース・キャッシュ・モードを有効にすることができます。

1. ALTER DATABASEシステム権限を持つユーザーとしてインスタンスに接続します。
2. データベースはマウントされ、オープンされていないことを確認します。

   「インスタンスを起動し、データベースをマウントする方法」を参照してください。
3. 次のSQL文を発行します。

   ```
   ALTER DATABASE FORCE FULL DATABASE CACHING;
   ```
4. (オプション)データベースをオープンします。

   ```
   ALTER DATABASE OPEN;
   ```

親トピック: 強制フル・データベース・キャッシュ・モードの使用方法

6.5.4 強制フル・データベース・キャッシュ・モードの無効化

データベースの強制フル・データベース・キャッシュ・モードを無効にすることができます。

1. ALTER DATABASEシステム権限を持つユーザーとしてインスタンスに接続します。
2. データベースはマウントされ、オープンされていないことを確認します。

   「インスタンスを起動し、データベースをマウントする方法」を参照してください。
3. 次のSQL文を発行します。

   ```
   ALTER DATABASE NO FORCE FULL DATABASE CACHING;
   ```
4. (オプション)データベースをオープンします。

   ```
   ALTER DATABASE OPEN;
   ```

親トピック: 強制フル・データベース・キャッシュ・モードの使用方法

6.6 Database Smart Flash Cacheの構成

データベース・スマート・フラッシュ・キャッシュ機能は、ソリッド・ステート・デバイス(SSD)テクノロジを使用してデータベース・バッファ・キャッシュを透過的に拡張します。Database Smart Flash Cacheにより、ディスクI/O量が削除され、同等サイズのRAMを追加するより低いコストでOracleデータベースのパフォーマンスを大幅に向上させることができます。

- Database Smart Flash Cacheを構成する場合

いくつかの条件が満たされた場合は、データベース・スマート・フラッシュ・キャッシュの構成を検討する必要があります。

- Database Smart Flash Cacheのサイズの設定
  一般的に、データベース・スマート・フラッシュ・キャッシュのサイズは、バッファ・キャッシュのサイズの2から10倍に設定します。
- Database Smart Flash Cacheのメモリーのチューニング
  バッファ・キャッシュからデータベース・スマート・フラッシュ・キャッシュに移動される各データベース・ブロックにつき、そのブロックに関する少量のメタデータがバッファ・キャッシュに保持されます。
- Database Smart Flash Cacheの初期化パラメータ
  データベース・スマート・フラッシュ・キャッシュを構成するために、初期化パラメータのセットを使用できます。
- Oracle Real Application Clusters環境のDatabase Smart Flash Cache
  Oracle Real Application Clusters環境では、すべてのインスタンスにデータベース・スマート・フラッシュ・キャッシュを構成するか、どのインスタンスにも構成しないことをお薦めします。また、各インスタンスに構成されたフラッシュ・キャッシュの合計サイズはだいたい同じである必要があります。

関連項目:

データベース・スマート・フラッシュ・キャッシュについては「メモリー・アーキテクチャの概要」

親トピック: メモリーの管理

6.6.1 Database Smart Flash Cacheを構成する場合

いくつかの条件が満たされた場合は、データベース・スマート・フラッシュ・キャッシュの構成を検討する必要があります。

次のすべての条件が該当する場合は、データベース・スマート・フラッシュ・キャッシュの追加を検討してください。

- データベースがSolarisまたはOracle Linuxオペレーティング・システムで実行されています。Database Smart Flash Cacheは、これらのオペレーティング・システムでのみサポートされます。
- 自動ワークロード・リポジトリ(AWR)レポートまたはSTATSPACKレポートのバッファ・プール・アドバイザセクションには、バッファ・キャッシュのサイズを2倍にすると効果的であることが示されています。
- `db file sequential read`は上位待機イベントです。
- 手元に予備のCPUがあります。

ノート:

複数のインスタンス間で 1 つのフラッシュ・ファイルを共有することはできません。ただし、論理ボリューム・マネージャまたは同様のツールを使用してフラッシュ・デバイスを静的にパーティション化している場合、複数のインスタンス間で 1 つのフラッシュ・デバイスを共有できます。

親トピック: Database Smart Flash Cacheの構成

6.6.2 Database Smart Flash Cacheのサイズの設定

一般的に、Database Smart Flash Cacheのサイズはバッファ・キャッシュのサイズの2から10倍に設定します。

乗数を2未満にすると、効果がありません。自動共有メモリー管理を使用している場合は、Database Smart Flash

CacheをSGA_TARGETのサイズの2倍から10倍にします。この計算に使用するSGA_TARGETのサイズは、フル・サイズではなく、その80%でも十分です。

親トピック: Database Smart Flash Cacheの構成

6.6.3 Database Smart Flash Cacheのメモリーのチューニング

バッファ・キャッシュからDatabase Smart Flash Cacheにデータベース・ブロックを移動するたびに、そのブロックに関する少量のメタデータがバッファ・キャッシュに保持されます。

単一インスタンス・データベースの場合、このメタデータによって、約100バイトが消費されます。Oracle Real Application Clusters(Oracle RAC)データベースの場合は、約200バイト消費されます。このため、Database Smart Flash Cacheを追加するときには、この余分なメモリー要件を考慮する必要があります。

データベース・スマート・フラッシュ・キャッシュのためにメモリーをチューニングするには、次のいずれかのアクションを完了します。

- 手動でメモリーを管理している場合は、構成されているデータベース・スマート・フラッシュ・キャッシュに収まるデータベース・ブロックの数に100 (Oracle RACの場合は200)を乗じた値とほぼ等しくなるまでバッファ・キャッシュのサイズを増やしてください。
- 自動メモリー管理を使用している場合は、前述のアルゴリズムを使用して、MEMORY_TARGET初期化パラメータのサイズを増やしてください。先にMEMORY_MAX_TARGET初期化パラメータのサイズを増やすことが必要になる場合があります。
- 自動共有メモリー管理を使用している場合は、SGA_TARGET初期化パラメータのサイズを増やしてください。

また、フラッシュ・キャッシュを使用するOracle RACデータベースでは、グローバル・キャッシュ・サービス(GCS)リソース用に共有プールに追加メモリーを割り当てる必要があります。各GCSリソースは、共有プールに約208バイト必要です。

ノート:

- バッファ・キャッシュ・サイズを増やさなくても、Database Smart Flash Cache を追加できます。この場合、バッファ・キャッシュの有効サイズが減少します。場合によっては、この損失分はサイズの大きい Database Smart Flash Cache を使用することによって相殺できます。
- データベース・スマート・フラッシュ・キャッシュをフラッシュするには、ALTER SYSTEM FLUSH FLASH_CACHE 文を発行します。データベース・スマート・フラッシュ・キャッシュのフラッシュは、リライトされた問合せ、または同一の開始点からの一連の問合せのパフォーマンスを測定する必要がある場合、またはキャッシュが破損している可能性がある場合に有効です。

関連項目:

「メモリー管理について」

親トピック: Database Smart Flash Cacheの構成

6.6.4 Database Smart Flash Cacheの初期化パラメータ

データベース・スマート・フラッシュ・キャッシュを構成するために、初期化パラメータのセットを使用できます。

表6-4 Database Smart Flash Cacheの初期化パラメータ

パラメータ	説明
DB_FLASH_CACHE_FILE	Database Smart Flash Cache が含まれるファイルのパスおよびファイル名のリストを、オペレーティング・システムのファイル・システムまたは Oracle Automatic Storage Management ディスク・グループ内で指定します。指定されたファイルが存在しない場合は、起動時にデータベースによって作成されます。各ファイルはフラッシュ・デバイスに常駐する必要があります。Database Smart Flash Cache をディスク・ドライブ(スピンドル)に構成すると、パフォーマンスが低下する可能性があります。最大で 16 個のファイルがサポートされます。
DB_FLASH_CACHE_SIZE	Database Smart Flash Cache 内の各ファイルのサイズを指定します。各サイズは、DB_FLASH_CACHE_FILE で指定されたファイルに対応しています。ファイルおよびサイズは、指定された順序で対応しています。指定されたサイズの数が指定されたファイルの数に一致しない場合は、エラーが発生します。 サイズ指定は、そのフラッシュ・デバイスの物理メモリー・サイズ以下である必要があります。サイズは、nG というように、ギガバイト(GB)数で示します。たとえば、16GB の Database Smart Flash Cache を指定するには、DB_FLASH_CACHE_SIZE 値を 16G に設定します。

たとえば、Database Smart Flash Cacheで次のフラッシュ・デバイスが使用されていると想定します。

ファイル	サイズ
/dev/sda	32G
/dev/sdb	32G
/dev/sdc	64G

初期化パラメータを次の値に設定できます。

```
DB_FLASH_CACHE_FILE = /dev/sda, /dev/sdb, /dev/sdc
DB_FLASH_CACHE_SIZE = 32G, 32G, 64G
```

V$FLASHFILESTATビューを問い合せて、各ファイルの累積待機時間および読取り数を確認し、平均待機時間を計算できます。

ALTER SYSTEMを使用して、使用禁止にするフラッシュ・デバイスごとにDB_FLASH_CACHE_SIZEをゼロに設定できます。また、ALTER SYSTEMを使用して、使用禁止にした任意のフラッシュ・デバイスのサイズを元のサイズに戻し、再び使用可能にすることもできます。ただし、Database Smart Flash Cacheのサイズを動的に変更することはできません。

関連項目:

この項で説明している初期化パラメータの詳細およびV$FLASHFILESTATビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: Database Smart Flash Cacheの構成

6.6.5 Oracle Real Application Clusters環境のDatabase Smart Flash Cache

Oracle Real Application Clusters環境では、すべてのインスタンスについてDatabase Smart Flash Cacheを構成するか、または一切構成しないことをお薦めします。また、各インスタンスに構成されたフラッシュ・キャッシュの合計サイズはだいたい同じである必要があります。

親トピック: Database Smart Flash Cacheの構成

6.7 Oracle Database In-Memoryによる問合せパフォーマンスの向上

Oracle Database In-Memory (Database In-Memory)は、Oracle Database 12cリリース1 (12.1.0.2)に最初に導入された一連の機能であり、リアルタイム分析および混合ワークロードのパフォーマンスを大幅に改善します。

Database In-Memoryの機能により、次の操作を実行する問合せのパフォーマンスが大幅に向上します。

- 大量の行のスキャンと<、>、=、INなどの演算子を使用するフィルタの適用
- 100列のうち5列にアクセスする問合せなど、大量に列がある表またはマテリアライズド・ビューからの少数の列の選択
- SQL演算子を使用したLOB列の選択
- 大きなファクト表と小さなディメンション表の結合
- データの集計

Database In-Memory機能セットには、インメモリー列ストア(IM列ストア)、高度な問合せ最適化、および可用性ソリューションが含まれています。

- IM列ストア

  IM列ストアは、Database In-Memoryの主要機能です。IM列ストアでは、表、パーティションおよび個別の列のコピーが、高速スキャン向けに最適化された、特別な圧縮列形式で保持されます。IM列ストアは、システム・グローバル領域(SGA)のオプション部分である、インメモリー領域に存在します。

  IM列ストアは行ベース・ストレージまたはデータベース・バッファ・キャッシュに代わるものではありませんが、それを補完します。データベースでは、行ベースおよび列形式の両方でメモリー内にデータを格納できるようになり、両方の長所が提供されます。IM列ストアにより、ディスク形式とは無関係な、トランザクションの一貫性がある、表データのコピーがさらに提供されます。

- 高度な問合せ最適化

  Database In-Memoryには、分析問合せのためのいくつかのパフォーマンス最適化が含まれています。

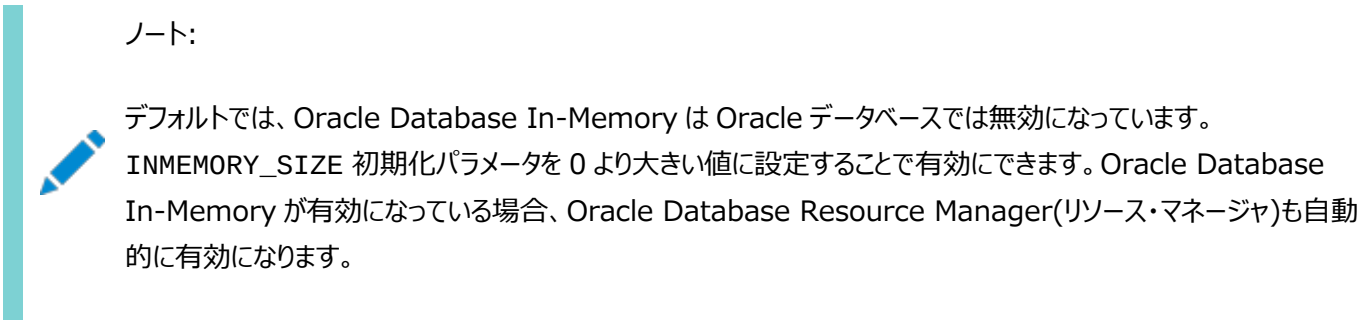
  - インメモリー式(IM式): IM列ストアでホットな式の特定と移入が可能です。
  - 結合グループ: 列値の解凍およびハッシュ化にかかわるパフォーマンスのオーバーヘッドを排除できます。
  - インメモリー集計(IM集計): 大きなファクト表と小さなディメンション表を結合する集計問合せのパフォーマンスが向上します。

- 再移入：変更されたオブジェクトをIM列ストアに自動的に再移入することで、問合せのパフォーマンスが向上します。
- インメモリー動的スキャン(IM動的スキャン)：CPUリソースがアイドル状態のときに軽量スレッドを使用して自動的に表スキャンをパラレル化することで、問合せのパフォーマンスが向上します。

- 高可用性のサポート

  Database In-Memoryには、次の可用性機能が組み込まれています。

  - データベース・インスタンス再起動時のIM列ストアへのデータ移入時間を短縮します。この機能は、インメモリー・ファスト・スタート(IMファスト・スタート)機能を使用して実現されます。
  - Oracle Real Application Clusters (Oracle RAC)環境の各ノードでIM列ストアを提供します。
  - Active Data Guard環境のスタンバイ・データベースでIM列ストアを提供します。



ノート:

デフォルトでは、Oracle Database In-Memory は Oracle データベースでは無効になっています。`INMEMORY_SIZE` 初期化パラメータを 0 より大きい値に設定することで有効にできます。Oracle Database In-Memory が有効になっている場合、Oracle Database Resource Manager(リソース・マネージャ)も自動的に有効になります。

関連項目:

- 『Oracle Database In-Memoryガイド』
- Oracleビデオ: Oracle Database In-Memoryの管理

親トピック: メモリーの管理

6.8 Memoptimized Rowstoreを使用する高パフォーマンス・データ・ストリーミングの有効化

Memoptimized Rowstoreでは、IoT (モノのインターネット)アプリケーションなど、アプリケーションでの高パフォーマンス・データ・ストリーミングが可能になります。この機能は、通常は、同時に多数のクライアントから単一行挿入で少量のデータをストリーミングし、また、非常に高い頻度でクライアントに対してデータを問い合せます。

Memoptimized Rowstoreには、次の機能があります。

- 高速インジェスト

  高速インジェストでは、データベースへの頻度の高い単一行データ挿入の処理が最適化されます。高速インジェストでは、データ挿入のパフォーマンスが向上するように、ディスク書込み前の挿入のバッファリングにラージ・プールが使用されます。

- 高速参照

  高速参照では、頻度の高い問合せでデータベースからデータを高速に取得できます。高速参照では、問合せのパフォーマンスが向上するように、表から問い合せたデータのバッファリングに、memoptimizeプールというSGA内の別個

のメモリー領域が使用されます。

ノート:

高速参照を使用する場合は、MEMOPTIMIZE_POOL_SIZE 初期化パラメータを使用して、適切なメモリー・サイズを memoptimize プールに割り当てる必要があります。

関連項目:

- Memoptimized Rowstoreの構成と使用の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください
- memoptimizeプールのメモリー・アーキテクチャの詳細は、『Oracle Database概要』を参照してください
- MEMOPTIMIZE_POOL_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: メモリーの管理

6.9 メモリー管理の参考情報

自動メモリー管理は、一部のプラットフォームでのみサポートされています。また、メモリー管理に関する情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

- 自動メモリー管理をサポートするプラットフォーム
  自動メモリー管理は、いくつかのプラットフォームでサポートされています。
- メモリー管理のデータ・ディクショナリ・ビュー
  動的パフォーマンス・ビューのセットがメモリー管理の情報を提供します。

親トピック: メモリーの管理

6.9.1 自動メモリー管理をサポートするプラットフォーム

いくつかのプラットフォームで、自動メモリー管理がサポートされます。

次のプラットフォームでは自動メモリー管理がサポートされます。自動メモリー管理は、SGAおよびPGAのサイズを自動的にチューニングするOracle Databaseの機能で、パフォーマンスを最適化するために、要求に応じてメモリーが領域間で再配分されます。

- Linux
- Solaris
- Windows
- HP-UX
- AIX

親トピック: メモリー管理の参考情報

6.9.2 メモリー管理のデータ・ディクショナリ・ビュー

動的パフォーマンス・ビューのセットがメモリー管理の情報を提供します。

ビュー	説明
V$SGA	システム・グローバル領域(SGA)に関する要約情報が表示されます。
V$SGAINFO	様々な SGA コンポーネントのサイズ、グラニュル・サイズおよび空きメモリーなど、SGA に関するサイズ情報が表示されます。
V$SGASTAT	共有プール、ラージ・プール、Java プールおよび Streams プール内に割り当てられているメモリーの量に関する詳細情報が表示されます。
V$PGASTAT	PGA メモリー使用統計と、自動 PGA メモリー・マネージャが使用可能な場合(PGA_AGGREGATE_TARGET が設定されている場合)はその統計が表示されます。V$PGASTAT での累積値は、インスタンスの起動以降に蓄積されます。
V$MEMORY_DYNAMIC_COMPONENTS	自動調整された静的なすべてのメモリー・コンポーネントの現在のサイズに関する情報と、各コンポーネントで発生した最後の操作(たとえば、拡大または縮小)が表示されます。
V$SGA_DYNAMIC_COMPONENTS	すべての SGA コンポーネントの現在のサイズと各コンポーネントの最後の操作が表示されます。
V$SGA_DYNAMIC_FREE_MEMORY	将来の動的な SGA サイズ変更操作に使用可能な SGA メモリーの容量に関する情報が表示されます。
V$MEMORY_CURRENT_RESIZE_OPS	現在進行中のサイズ変更操作に関する情報が表示されます。サイズ変更の操作は、SGA、インスタンス PGA または動的な SGA コンポーネントの拡大または縮小です。
V$SGA_CURRENT_RESIZE_OPS	現在進行中の動的 SGA コンポーネントのサイズ変更操作に関する情報が表示されます。
V$MEMORY_RESIZE_OPS	SGA_TARGET および PGA_AGGREGATE_TARGET に対する自動拡張および縮小操作も含めて、最後に完了した 800 件のメモリー・コンポーネント・サイズ変更操作に関する情報が表示されます。
V$SGA_RESIZE_OPS	最後に完了した 800 件の SGA コンポーネント・サイズ変更操作に関する情報が表示されます。

ビュー	説明
V$MEMORY_TARGET_ADVICE	自動メモリー管理を使用可能にした場合、MEMORY_TARGET のチューニングに役立つ情報が表示されます。
V$SGA_TARGET_ADVICE	SGA_TARGET のチューニングに役立つ情報が表示されます。
V$PGA_TARGET_ADVICE	PGA_AGGREGATE_TARGET のチューニングに役立つ情報が表示されます。
V$IM_SEGMENTS	IM 列ストア内のすべてのセグメントに割り当てられている記憶域に関する情報が表示されます。 ノート: このビューは、Oracle Database 12c リリース 1 (12.1.0.2)以上で使用できます。

親トピック: メモリー管理の参考情報

7 ユーザーの管理とデータベースのセキュリティ保護

各データベースのセキュリティ・ポリシーを設定します。

- データベースに対するセキュリティ・ポリシー設定の重要性
  すべてのデータベースに対してセキュリティ・ポリシーを設定することが重要です。セキュリティ・ポリシーによって、不慮または不正によるデータの破壊またはデータベース・インフラストラクチャの損傷からデータベースを保護する方法が設定されます。
- ユーザーとリソースの管理
  データベースに接続するには、各ユーザーが、データベースに事前に定義された有効なユーザー名を指定する必要があります。ユーザーにはアカウントが設定され、ユーザーに関する情報がデータベース・ディクショナリに格納されている必要があります。
- ユーザー権限とロール
  権限とロールは、ユーザーのデータへのアクセスおよび実行可能なSQL文のタイプを制御するために使用します。
- データベース・アクティビティの監査
  選択したユーザーのデータベース操作は、管理者が実行した操作も含めて、監視および記録できます。システム全体の処理および個々のデータベース・オブジェクトで実行された処理を監視できます。このタイプの監視は、データベース監査と呼ばれます。
- 事前定義のユーザー・アカウント
  Oracle Databaseには、いくつかの事前定義のユーザー・アカウントが用意されています。

親トピック: 基本データベース管理

7.1 データベースに対するセキュリティ・ポリシー設定の重要性

すべてのデータベースに対してセキュリティ・ポリシーを設定することが重要です。セキュリティ・ポリシーによって、不慮または不正によるデータの破壊またはデータベース・インフラストラクチャの損傷からデータベースを保護する方法が設定されます。

各データベースに、データベースのセキュリティ・ポリシーを実装および維持するセキュリティ管理者と呼ばれる管理者を設定します。データベース・システムが小さい場合は、データベース管理者がセキュリティ管理者を担当することもできます。ただし、データベース・システムが大きい場合は、指定したユーザーまたはユーザーのグループが単独でセキュリティ管理者として担当することもあります。

データベースに対するセキュリティ・ポリシー設定の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: ユーザーの管理とデータベースのセキュリティ保護

7.2 ユーザーとリソースの管理

データベースに接続するには、各ユーザーが、データベースに事前に定義された有効なユーザー名を指定する必要があります。ユーザーにはアカウントが設定され、ユーザーに関する情報がデータベース・ディクショナリに格納されている必要があります。

データベース・ユーザー(アカウント)を作成するときは、ユーザーに関する次の属性を指定します。

- ユーザー名
- 認証方式
- デフォルト表領域

- 一時表領域
- その他の表領域および割当て制限
- ユーザー・プロファイル

ユーザーの作成および管理方法を学習するには、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: ユーザーの管理とデータベースのセキュリティ保護

7.3 ユーザー権限とロール

権限とロールは、ユーザーのデータへのアクセスおよび実行可能なSQL文のタイプを制御するために使用します。

次の表は、権限とロールの、3つのタイプを示しています。

タイプ	説明
システム権限	通常は管理者によってのみ付与される、システムが定義する権限。これらの権限は、ユーザーによる特定のデータベース操作の実行を許可します。
オブジェクト権限	システムによって定義された権限。特定のオブジェクトへのアクセスを制御します。
ロール	権限や他のロールの集合。システムによって定義されたロールも存在しますが、大部分は管理者によって作成されます。権限や他のロールをグループ化するロールによって、複数の権限またはロールをユーザーに容易に付与できます。

権限とロールを付与する権限を所有しているユーザーは、権限とロールを他のユーザーに付与できます。権限とロールの付与は、管理者レベルで開始します。データベースの作成時に、管理ユーザーSYSが作成され、システム権限およびOracle Databaseで事前定義されたロールがすべて付与されます。次に、ユーザーSYSは、権限とロールを他のユーザーに付与し、そのユーザーが別のユーザーに対して特定の権限を付与できる権限を与えることもできます。

ユーザーの権限とロールの管理方法を学習するには、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: ユーザーの管理とデータベースのセキュリティ保護

7.4 データベース・アクティビティの監査

選択したユーザーのデータベース操作は、管理者が実行した操作も含めて、監視および記録できます。システム全体の処理および個々のデータベース・オブジェクトで実行された処理を監視できます。このタイプの監視は、データベース監査と呼ばれます。

統合監査ポリシーを作成し、SQL文を使用してそれらの監査ポリシーを管理できます。Oracle Databaseには標準監査設定を含むデフォルトの統合監査ポリシーが用意されており、カスタム統合監査ポリシーを作成できます。DBMS_FGA PL/SQLパッケージを使用して、ファイングレイン監査ポリシーを作成することもできます。

関連項目:

データベース監査の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: ユーザーの管理とデータベースのセキュリティ保護

7.5 事前定義されたユーザー・アカウント

Oracle Databaseには、いくつかの事前定義のユーザー・アカウントが用意されています。

次の3種類の事前定義のアカウントがあります。

- 管理アカウント(SYS、SYSTEM、SYSBACKUP、SYSDG、SYSKM、SYSRAC、SYSMAN、DBSNMP)

  SYS、SYSTEM、SYSBACKUP、SYSDG、SYSKM、SYSRACの詳細は、「データベース管理者のセキュリティと権限について」を参照してください。SYSMANは、Oracle Enterprise Manager Cloud Control (Cloud Control)の管理タスクの実行に使用されます。DBSNMPアカウントは、データベースを監視および管理するためにCloud Controlの管理エージェントで使用されます。これらのアカウントは削除することができません。

- サンプル・スキーマのアカウント

  これらのオプションのアカウントは、Oracle Databaseのマニュアルや説明書の例で使用されます。サンプル・スキーマのアカウントは、HR、SHおよびOEです。

- 内部アカウント

  個々のOracle Database機能またはコンポーネントが独自のスキーマを持てるよう、これらのアカウントが作成されます。内部のアカウントを削除しないでください。また内部のアカウントでログインしないでください。

ノート:

Oracle Database 19c 以上では、SYS およびサンプル・スキーマを除き、Oracle Database で提供されるユーザー・アカウントのほとんどはスキーマ専用アカウント(つまり、これらのアカウントはパスワードなしで作成される)です。これにより、悪質なユーザーがこれらのアカウントにログインできなくなります。これらのアカウントには、認証する必要が生じたときにいつでもパスワードを割り当てることができますが、セキュリティを強化するために、認証が必要なくなった場合にはこれらのアカウントをスキーマ専用アカウントに戻すことをお薦めします。

関連項目:

- Oracle Databaseに用意されているすべての事前定義済アカウントの詳細は、Oracle Databaseセキュリティ・ガイドを参照してください。
- スキーマ専用アカウントの詳細は、Oracle Databaseセキュリティ・ガイドを参照してください。
- Oracle Databaseに用意されているすべてのサンプル・スキーマの詳細は、Oracle Databaseサンプル・スキーマを参照してください。

親トピック: ユーザーの管理とデータベースのセキュリティ保護

8 データベースの監視

データベースの動作を定期的に監視することは重要です。監視することによって、気付いていないエラーの情報を取得できるのみでなく、データベースの正常な動作について理解を深めることもできます。正常な動作を理解しておくことで、なんらかの誤りがある場合に状況を容易に認識できるようになります。

- エラーおよびアラートの監視
  データベースのエラーやアラートを監視することにより、問題を予防、検出および解決できます。
- パフォーマンスの監視
  パフォーマンスの監視には、ロックおよび待機イベントの監視、一連のデータ・ディクショナリ・ビューの問合せが含まれます。
- 隔離されたオブジェクトの監視
  オブジェクトの隔離により、破損したリカバリ不能なオブジェクトが存在する場合でも、Oracleデータベースが機能できるようにすることができます。V$QUARANTINEビューに、隔離されたオブジェクトの情報が含まれています。

親トピック: 基本データベース管理

8.1 エラーおよびアラートの監視

データベースのエラーやアラートを監視することにより、問題を予防、検出および解決できます。

ノート:

データベースでのエラーおよびアラートを監視する場合、Oracle Enterprise Manager Cloud Control (Cloud Control)のデータベース・ホームページを使用する方法が最も簡単であり、かつ最適です。詳細は、Cloud Control のオンライン・ヘルプを参照してください。この項で説明するのは、データ・ディクショナリ・ビュー、PL/SQL パッケージおよびその他のコマンドライン機能を使用する別の監視方法です。

- トレース・ファイルおよびアラート・ログを使用したエラーの監視
  トレース・ファイルは、問題を調査するために使用される診断データを含むファイルです。アラート・ログは、データベース・メッセージおよびエラーの時系列ログを提供するファイルです。
- サーバー生成アラートを使用したデータベースの監視
  サーバー生成アラートは、切迫した問題のOracle Databaseサーバーからの通知です。

親トピック: データベースの監視

8.1.1 トレース・ファイルおよびアラート・ログを使用したエラーの監視

トレース・ファイルは、問題の調査に使用される診断データが含まれるファイルです。アラート・ログは、データベース・メッセージおよびエラーの時系列ログを提供するファイルです。

- トレース・ファイルおよびアラート・ログを使用したエラーの監視について
  トレース・ファイルおよびアラート・ログには、エラーに関する情報が含まれています。
- アラート・ログのサイズの制御
  アラート・ログのサイズを制御するには、必要でないファイルを手動で削除する必要があります。削除しないと、ファイルが引き続き追加されます。
- トレース・ファイル・サイズの制御

初期化パラメータMAX_DUMP_FILE_SIZEを使用して、すべてのトレース・ファイル(アラート・ログを除く)の最大サイズを制御できます。

- Oracle Databaseがトレース・ファイルに書き込む時期の制御
  適切な場合、バックグラウンド・プロセスは常にトレース・ファイルに書き込みます。
- 共有サーバー・セッション用トレース・ファイルの読込み
  共有サーバーが使用可能な場合、ディスパッチャを使用している各セッションは共有サーバー・プロセスにルーティングされ、セッションのトレースが有効になっている場合(またはエラーが発生した場合)のみ、トレース情報がサーバー・トレース・ファイルに書き込まれます。そのため、ディスパッチャを使用して接続していた特定のセッションのトレースを追跡する場合、複数の共有サーバー・トレース・ファイルを調べる必要がある場合があります。

親トピック: エラーおよびアラートの監視

8.1.1.1 トレース・ファイルおよびアラート・ログを使用したエラーの監視について

トレース・ファイルとアラート・ログには、エラーについての情報が含まれています。

各サーバー・プロセスとバックグラウンド・プロセスは、対応するトレース・ファイルに情報を書き込むことができます。プロセスによって内部エラーが検出されると、エラー情報が関連トレース・ファイルにダンプされます。トレース・ファイルに書き込まれる情報の一部はデータベース管理者用であり、その他の情報はOracleサポート・サービス用です。また、トレース・ファイルの情報は、アプリケーションとインスタンスのチューニングにも使用されます。

ノート:

クリティカル・エラーの場合は、自動診断リポジトリにインシデントおよびインシデント・ダンプも作成されます。詳細は、問題の診断と解決を参照してください。

アラート・ログはメッセージとエラーの履歴ログであり、次の項目が含まれます。

- 発生したすべての内部エラー(ORA-00600)、ブロック破損エラー(ORA-01578)およびデッドロック・エラー(ORA-00060)
- いくつかのCREATE、ALTERおよびDROP文などの管理操作、およびSTARTUP、SHUTDOWNおよびARCHIVELOG文
- 共有サーバーとディスパッチャ・プロセスの機能に関係するメッセージとエラー
- マテリアライズド・ビューの自動リフレッシュ中に発生したエラー
- データベースとインスタンスの起動時点でデフォルト以外の値があったすべての初期化パラメータの値

Oracle Databaseは、オペレータのコンソール上に情報を表示するかわりに(一部のシステムではコンソール上に情報を表示する)、アラート・ログを使用して、これらの操作を記録します。操作が成功すると、タイムスタンプとともに「completed」というメッセージがアラート・ログに書き込まれます。

アラート・ログは、XML形式ファイルとテキスト形式ファイルの両方として維持されます。いずれの形式のアラート・ログもテキスト・エディタで表示でき、XML形式のファイルはADRCIユーティリティを使用してXMLタグが削除された状態で表示できます。

バックグラウンド・プロセスでエラーが発生していないかどうかを確認するために、インスタンスのアラート・ログとトレース・ファイルを定期的にチェックします。たとえば、ログ・ライター・プロセス(LGWR)でログ・グループのメンバーに書き込むことができないと、LGWRトレース・ファイルとアラート・ログに、問題の内容を示すエラー・メッセージが書き込まれます。このようなエラー・メッセージが見つかった場合は、メディアまたはI/Oの問題が発生しているため、ただちに解決する必要があります。

Oracle Databaseは、他の重要な統計に加えて、初期化パラメータの値もアラート・ログに書き込みます。

バックグラウンドおよびサーバー・プロセスのアラート・ログおよびすべてのトレース・ファイルは自動診断リポジトリに書き込まれ、これらの場所はDIAGNOSTIC_DEST初期化パラメータで指定されます。トレース・ファイル名はオペレーティング・システム固有ですが、通常、各ファイルにはそれを書き込むLGWRやRECOなどのプロセスの名前が含まれます。

関連項目:

- 自動診断リポジトリ(ADR)の詳細は、問題の診断と解決を参照してください。
- アラート・ログの詳細は、「アラート・ログ」を参照してください。
- 「アラート・ログの表示」
- ADRCIユーティリティの詳細は、『Oracle Databaseユーティリティ』を参照してください。
- トレース・ファイルの名前の詳細は、オペレーティング・システム固有のOracleマニュアルを参照してください。

親トピック: トレース・ファイルおよびアラート・ログを使用したエラーの監視

8.1.1.2 アラート・ログのサイズの制御

アラート・ログのサイズを制御するには、不要になったファイルを手動で削除する必要があります。削除しないと、ファイルが引き続き追加されます。

インスタンスの実行中にアラート・ログを削除しても問題はありませんが、削除する前にアラート・ログのアーカイブ・コピーを作成してください。このアーカイブ・コピーは、インスタンス履歴の調査を必要とする問題が発生したときに役立ちます。

アラート・ログのサイズを制御するには:

- アラート・ログ・ファイルを削除します。

親トピック: トレース・ファイルおよびアラート・ログを使用したエラーの監視

8.1.1.3 トレース・ファイル・サイズの制御

初期化パラメータMAX_DUMP_FILE_SIZEを使用して、すべてのトレース・ファイル(アラート・ログを除く)の最大サイズを制御できます。

このパラメータは、次の方法で設定できます。

- 数値により、オペレーティング・システム・ブロック内の最大サイズを指定します。制限を取得するために、指定された値がブロック・サイズで乗算されます。
- 数値の後に接尾辞K、MまたはGを付けることによって、ファイル・サイズをKB、MBまたはGB単位で指定します。
- デフォルトであるUNLIMITEDでは、制限は指定されません。
- トレース・ファイル・セグメンテーションとMAX_DUMP_FILE_SIZE
  Oracle Databaseでは、MAX_DUMP_FILE_SIZE初期化パラメータで指定された制限に基づいて、自動的にトレース・ファイルをセグメントに分割できます。制限に達すると、データベースで順序番号が使用され、現在のトレース・ファイルの名前が変更され、元の名前を使用して空のファイルが作成されます。

関連項目:

- MAX_DUMP_FILE_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- IPSの詳細は、「Oracle Databaseの障害診断インフラストラクチャについて」

親トピック: トレース・ファイルおよびアラート・ログを使用したエラーの監視

8.1.1.3.1 トレース・ファイルのセグメント化およびMAX_DUMP_FILE_SIZE

Oracle Databaseでは、MAX_DUMP_FILE_SIZE初期化パラメータを使用して指定した制限に基づいて、トレース・ファイルに対し自動的にセグメントを作成できます。制限に達すると、データベースで順序番号が使用され、現在のトレース・ファイルの名前が変更され、元の名前を使用して空のファイルが作成されます。

次の表は、MAX_DUMP_FILE_SIZEの設定に基づいたトレース・ファイルのセグメントを示しています。

表8-1 MAX_DUMP_FILE_SIZEパラメータおよびトレース・ファイルのセグメント化

MAX_DUMP_FILE_SIZEの設定	トレース・ファイルのセグメント化
UNLIMITED	トレース・ファイルにセグメントは作成されません。
15M より大きい	トレース・ファイルは、MAX_DUMP_FILE_SIZE の設定の 1/5 の境界でセグメント化されます。この境界未満のサイズのトレース・ファイルはセグメント化されません。たとえば、MAX_DUMP_FILE_SIZE の設定が 100M の場合、境界は 20MB (100MB の 1/5) になります。
15M 以下	トレース・ファイルにセグメントは作成されません。

最大5までセグメントを作成することが可能ですが、セグメントの合計サイズはMAX_DUMP_FILE_SIZEの制限以下にする必要があります。トレース・ファイルのすべてのセグメントの合計サイズが指定した制限を超える場合、1番目のセグメントの後の最も古いセグメントが削除され、新しい空のセグメントが作成されます。したがって、トレース・ファイルには常に最新のトレース情報を含んでいることになります。最初のセグメントは、プロセスの初期状態に該当する情報が含まれる可能性があるため削除されません。

セグメントは、トレース・ファイルのスペース管理を向上させます。具体的には、セグメントによって次の方法でトレース・ファイルを管理することが可能になります。

- 古いトレース・ファイルは、不要になった場合消去できます。
- 小さなトレース・ファイルで問題を診断し、インシデント・パッケージング・サービス(IPS)用にパッケージするトレース・ファイルを分離できます。

ノート:

インシデントを含む時間範囲をカバーするセグメントは削除されません。それは 5 個のデフォルト・セグメントに加えて維持されます。

親トピック: トレース・ファイル・サイズの制御

8.1.1.4 Oracle Databaseがトレース・ファイルに書き込む時期の制御

バックグラウンド・プロセスは適宜、トレース・ファイルに情報を書き込みます。

ARCnバックグラウンド・プロセスの場合、LOG_ARCHIVE_TRACE初期化パラメータを介して、生成されるトレース情報の量およびタイプを制御することが可能です。これを行うには:

- 「ARCHIVELOGプロセスによって生成されるトレース出力の制御」の項で説明されている手順を実行してください。

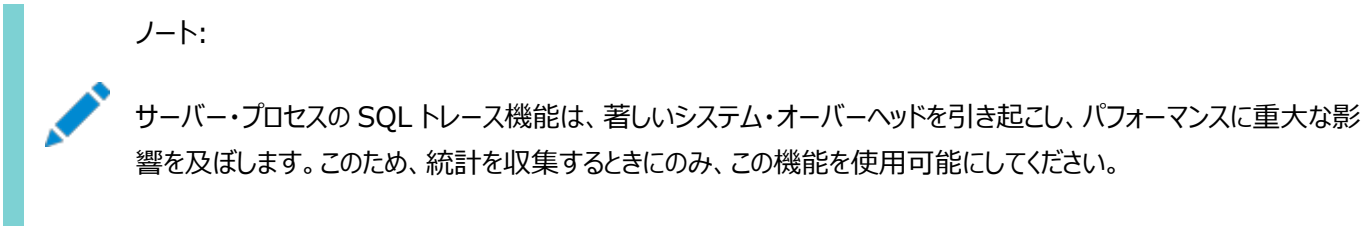
他のバックグラウンド・プロセスには、このような柔軟性はありません。

クリティカル・エラーが発生したときは必ず、サーバー・プロセスのためにトレース・ファイルに情報が書き込まれます。また、初期化パラメータSQL_TRACE = TRUEを設定すると、SQLトレース機能が有効になり、インスタンスに対するすべてのSQL文の処理についてパフォーマンス統計が生成され、自動診断リポジトリに書き込まれます。

必要に応じて、サーバー・プロセスに対するトレース・ファイルの生成を要求できます。SQL文ALTER SESSION SET SQL_TRACEを使用すると、SQL_TRACE初期化パラメータの現行の値にかかわらず、対応するサーバー・プロセスのために各セッションのトレース・ロギングを使用可能または使用禁止にできます。次の例は、特定のセッションに対してSQLトレース機能を使用可能にします。

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

セッションのSQLトレースを制御するには、DBMS_SESSIONまたはDBMS_MONITORパッケージを使用します。



ノート:

サーバー・プロセスの SQL トレース機能は、著しいシステム・オーバーヘッドを引き起こし、パフォーマンスに重大な影響を及ぼします。このため、統計を収集するときにのみ、この機能を使用可能にしてください。

関連項目:

- データベースでクリティカル・エラー(インシデントとも呼ばれる)がどのように処理されるかの詳細は、問題の診断と解決を参照してください。

親トピック: トレース・ファイルおよびアラート・ログを使用したエラーの監視

8.1.1.5 共有サーバー・セッション用トレース・ファイルの読込み

共有サーバーが使用可能な場合、ディスパッチャを使用している各セッションは共有サーバー・プロセスにルーティングされ、セッションのトレースが有効になっている場合(またはエラーが発生した場合)のみ、トレース情報がサーバー・トレース・ファイルに書き込まれます。そのため、ディスパッチャを使用して接続していた特定のセッションのトレースを追跡する場合、複数の共有サーバー・トレース・ファイルを調べる必要がある場合があります。

これを容易にするため、Oracleではコマンドライン・ユーティリティ・プログラムtrcsessを用意しており、これを使用すると特定のユーザー・セッションに関連のあるすべてのトレース情報が1箇所にまとめられ、情報が時間の順に並べられます。

関連項目:

SQLトレース機能の使用、および生成されたトレース・ファイルを解析するTKPROFとtrcsessの使用の詳細は、『Oracle Database SQLチューニング・ガイド』

親トピック: トレース・ファイルおよびアラート・ログを使用したエラーの監視

8.1.2 サーバー生成アラートを使用したデータベースの監視

サーバー生成アラートとは、切迫した問題に関するOracle Databaseサーバーからの通知です。

- サーバー生成アラートによるデータベースの監視について
  サーバー生成アラートに、問題を修正するための提案が含まれている場合があります。問題の状況が解決した場合も通知が発行されます。
- サーバー生成アラートのしきい値の設定と取得
  サーバー・アラート・メトリックのしきい値設定は、DBMS_SERVER_ALERT PL/SQLパッケージのSET_THRESHOLDおよびGET_THRESHOLDプロシージャを使用して表示および変更できます。
- サーバー生成アラートの表示
  サーバー生成アラートを表示するには、Cloud Controlのデータベース・ホーム・ページにアクセスするのが最も簡単な方法ですが、これ以外にもこれらのアラートを表示する方法があります。
- サーバー生成アラートのデータ・ディクショナリ・ビュー
  サーバー生成アラートの情報についてデータ・ディクショナリ・ビューを問い合せることができます。

親トピック: エラーおよびアラートの監視

8.1.2.1 サーバー生成アラートによるデータベースの監視について

サーバー生成アラートには、問題を修正するための提案が含まれていることがあります。問題の状況が解決した場合も通知が発行されます。

アラートは、問題が発生した場合や次のようなメトリックに予期した値に対してデータが一致しない場合に自動的に生成されます。

- 秒当たりの物理読込み数
- 秒当たりのユーザー・コミット数
- SQLサービス応答時間

サーバー生成アラートは、しきい値レベルに基づいて、または単純にイベントの発生に基づいて生成されるように設定できます。しきい値ベースのアラートは、しきい値の警告およびクリティカル・レベルの両方でトリガーできます。これらのレベルの値には、顧客定義または内部値のいずれをも使用でき、一部のアラートには、必要に応じて変更可能な、デフォルトのしきい値レベルが設定されています。たとえばデフォルトで、サーバー生成アラートは領域使用状況について、領域の使用率が85%の警告レベルまたは97%のクリティカルしきい値レベルを超過すると生成されます。しきい値レベルに基づかないアラートには、次のようなものがあります。

- スナップショットが古すぎます
- 再開可能セッションが一時停止されました
- リカバリ領域の領域使用

アラート・メッセージは、ユーザーSYSが所有する事前定義された永続キューALERT_QUEに送信されます。Cloud Controlはこのキューを読み込み、未処理のサーバー・アラートの通知を送信したり、場合によっては問題を修正するための処理を推奨します。アラートはCloud Controlのデータベース・ホームページに表示され、選択した管理者に電子メールまたはページャ通知を送信するように構成できます。アラートをアラート・キューに書き込めない場合、アラートに関するメッセージがOracle Databaseのアラート・ログに書き込まれます。

バックグラウンド・プロセスは、自動ワークロード・リポジトリにデータを定期的にフラッシュして、メトリック値の履歴を取得します。ア

ラート履歴表とALERT_QUEは、システムによって一定の間隔で自動的にパージされます。

親トピック: サーバー生成アラートを使用したデータベースの監視

8.1.2.2 サーバー生成アラートのしきい値の設定と取得

サーバー・アラート・メトリックのしきい値設定は、DBMS_SERVER_ALERT PL/SQLパッケージのSET_THRESHOLDおよびGET_THRESHOLDプロシージャを使用して表示および変更できます。

ノート:

しきい値を設定および取得する最も便利な方法は、Cloud Control のグラフィカル・インタフェースを使用する方法です。手順は、アラート管理に関する Cloud Control のオンライン・ヘルプを参照してください。

- しきい値レベルの設定
  DBMS_SERVER_ALERTパッケージのSET_THRESHOLDプロシージャは、しきい値のレベルを設定できます。
- しきい値情報の取得
  DBMS_SERVER_ALERTパッケージのGET_THRESHOLDプロシージャは、しきい値の情報を取得できます。

関連項目:

DBMS_SERVER_ALERTパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: サーバー生成アラートを使用したデータベースの監視

8.1.2.2.1 しきい値レベルの設定

DBMS_SERVER_ALERTパッケージのSET_THRESHOLDプロシージャは、しきい値のレベルを設定できます。

しきい値のレベルを設定するには:

- DBMS_SERVER_ALERTパッケージのSET_THRESHOLDプロシージャを実行し、適切な引数を指定します。

次の例は、SET_THRESHOLDプロシージャを使用して、インスタンスに対する各ユーザー・コールのCPU時間にしきい値を設定する方法を示しています。

```
DBMS_SERVER_ALERT.SET_THRESHOLD(
 DBMS_SERVER_ALERT.CPU_TIME_PER_CALL, DBMS_SERVER_ALERT.OPERATOR_GE, '8000',
 DBMS_SERVER_ALERT.OPERATOR_GE, '10000', 1, 2, 'inst1',
 DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE, 'main.regress.rdbms.dev.us.example.com');
```

この例では、各ユーザー・コールのCPU時間が8,000マイクロ秒を超えた場合は警告アラートが、各ユーザー・コールのCPU時間が10,000マイクロ秒を超えた場合はクリティカル・アラートが発行されます。次の引数が指定されています。

- CPU_TIME_PER_CALLは、メトリック識別子を指定しています。サポートされるメトリックのリストは、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- 観察期間は1分に設定しています。条件がしきい値を逸脱し、ここに指定した期間(分単位)にわたって逸脱状態が継続すると、アラートが発行されます。
- 連続発生回数は2回に設定しています。ここに指定した回数分メトリック値がしきい値に違反すると、アラートが生成されます。

- インスタンスの名前はinst1に設定しています。
- 定数DBMS_ALERT.OBJECT_TYPE_SERVICEは、しきい値が設定されるオブジェクト型を指定します。この例では、サービス名はmain.regress.rdbms.dev.us.example.comです。

親トピック: サーバー生成アラートのしきい値の設定と取得

8.1.2.2.2 しきい値情報の取得

DBMS_SERVER_ALERTパッケージのGET_THRESHOLDプロシージャは、しきい値の情報を取得できます。

しきい値を取得するには:

- DBMS_SERVER_ALERTパッケージのGET_THRESHOLDプロシージャを実行し、適切な引数を指定します。

次の例は、しきい値を取得します。

```
DECLARE
 warning_operator         BINARY_INTEGER;
 warning_value            VARCHAR2(60);
 critical_operator        BINARY_INTEGER;
 critical_value           VARCHAR2(60);
 observation_period       BINARY_INTEGER;
 consecutive_occurrences  BINARY_INTEGER;
BEGIN
 DBMS_SERVER_ALERT.GET_THRESHOLD(
 DBMS_SERVER_ALERT.CPU_TIME_PER_CALL, warning_operator, warning_value,
    critical_operator, critical_value, observation_period,
    consecutive_occurrences, 'inst1',
 DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE, 'main.regress.rdbms.dev.us.example.com');
 DBMS_OUTPUT.PUT_LINE('Warning operator:       ' || warning_operator);
 DBMS_OUTPUT.PUT_LINE('Warning value:          ' || warning_value);
 DBMS_OUTPUT.PUT_LINE('Critical operator:      ' || critical_operator);
 DBMS_OUTPUT.PUT_LINE('Critical value:         ' || critical_value);
 DBMS_OUTPUT.PUT_LINE('Observation_period:     ' || observation_period);
 DBMS_OUTPUT.PUT_LINE('Consecutive occurrences:' || consecutive_occurrences);
END;
/
```

DBA_THRESHOLDSビューを使用して、特定のしきい値設定をチェックすることもできます。たとえば:

```
SELECT metrics_name, warning_value, critical_value, consecutive_occurrences
   FROM DBA_THRESHOLDS
   WHERE metrics_name LIKE '%CPU Time%';
```

親トピック: サーバー生成アラートのしきい値の設定と取得

8.1.2.3 サーバー生成アラートの表示

サーバー生成アラートを表示するには、Cloud Controlのデータベース・ホーム・ページにアクセスするのが最も簡単な方法ですが、これ以外にもこれらのアラートを表示する方法があります。

Cloud Controlではなく、独自のツールを使用してアラートを表示する場合は、サーバー生成アラートを表示するための次のステップを実行してください。

1. ALERT_QUEにサブスクライブします。
2. ALERT_QUEを読み取ります。
3. アラートのしきい値レベルを設定した後にアラート通知を表示します

エージェントを作成し、ALERT_QUEにエージェントをサブスクライブするには、次のステップを実行します。

1. DBMS_AQADMパッケージのCREATE_AQ_AGENTプロシージャを実行します。
2. DBMS_AQADMパッケージのADD_SUBSCRIBERプロシージャを実行します。
3. 保護されたALERT_QUEのキューで待機しているメッセージにアクセスできるのは、サブスクライブしているエージェントに関連付けられたユーザーのみであるため、サブスクライブしているエージェントとデータベース・ユーザーを関連付けます。
4. DBMS_AQADMパッケージのENABLE_DB_ACCESSおよびGRANT_QUEUE_PRIVILEGEプロシージャを実行することにより、ユーザーにエンキュー権限を割り当てます。
5. アラートがALERT_QUEにエンキューされたときに非同期通知を受信するために、DBMS_AQ.REGISTERプロシージャで登録します。通知は、電子メール、HTTPポストまたはPL/SQLプロシージャの形式にできます。

アラート・メッセージを読むには、次のステップを実行します。

1. DBMS_AQ.DEQUEUEプロシージャまたはOCIAQDeqコールを使用します。
2. メッセージをデキューした後、DBMS_SERVER_ALERT.EXPAND_MESSAGEプロシージャを使用して、メッセージのテキストをオープンします。

関連項目:

- DBMS_AQパッケージの詳細は、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照
- DBMS_AQADMパッケージについては、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

親トピック: サーバー生成アラートを使用したデータベースの監視

8.1.2.4 サーバー生成アラートのデータ・ディクショナリ・ビュー

サーバー生成アラートの情報について、データ・ディクショナリ・ビューに問い合せることができます。

ビュー	説明
DBA_THRESHOLDS	インスタンスに定義されているしきい値設定をリストします。
DBA_OUTSTANDING_ALERTS	データベースで未処理のアラートを示します。
DBA_ALERT_HISTORY	クリアされたアラートの履歴をリストします。
V$ALERT_TYPES	各アラートのグループやタイプなどの情報を提供します。
V$METRICNAME	メトリックの名前、識別子およびシステム・メトリックに関する他の情報が含まれています。
V$METRIC	システム・レベルのメトリック値が含まれています。
V$METRIC_HISTORY	システム・レベルのメトリック値の履歴が含まれています。

親トピック: サーバー生成アラートを使用したデータベースの監視

8.2 パフォーマンスの監視

パフォーマンスの監視には、ロックおよび待機イベントの監視、一連のデータ・ディクショナリ・ビューの問合せが含まれます。

データベース・パフォーマンスの監視の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』および『Oracle Database SQLチューニング・ガイド』を参照してください。

- ロックの監視
  ロックは、同じリソースにアクセスする複数のトランザクション間で破壊的な相互作用を防止するメカニズムです。リソースは、表や行などのユーザー・オブジェクト、またはメモリー内の共有データ構造やデータ・ディクショナリ行など、ユーザーに対して表示されないシステム・オブジェクトの場合があります。
- 待機イベントの監視について
  待機イベントは、処理を継続する前にイベントが完了するまで待機する必要があることを示すために、サーバー・プロセスによって増分される統計です。セッションは、入力の待機、オペレーティング・システムによるディスクへの書込みなどのサービス完了の待機、ロックまたはラッチの待機など、様々な理由で待機することがあります。
- パフォーマンス監視のデータ・ディクショナリ・ビュー
  Oracle Databaseインスタンスを監視するために、データ・ディクショナリ・ビューのセットを問い合せることができます。

親トピック: データベースの監視

8.2.1 ロックの監視

ロック は、同じリソースにアクセスしている複数のトランザクション間で破壊的な相互作用が起きないようにするメカニズムです。リソースは、表や行などのユーザー・オブジェクト、またはメモリー内の共有データ構造やデータ・ディクショナリ行など、ユーザーに対して表示されないシステム・オブジェクトの場合があります。

SQL文の実行中にOracle Databaseが必要なロックを自動的に取得および管理するため、ユーザーが特に意識する必要はありません。ただし、手動でデータをロックすることも可能です。

互いにロックしたデータを2人以上のユーザーが待機している場合は、デッドロックが発生する可能性があります。デッドロックが発生すると、トランザクションの処理を継続できなくなる場合があります。Oracle Databaseは、デッドロック状況を自動的に検出し、そのデッドロックに関係している文の1つをロールバックしてデッドロック状況を解決し、競合している行ロックの一方を解放します。

Oracle Databaseは、デッドロックを回避するように設計されているため、デッドロックは頻繁には発生しません。最も多いデッドロックは、トランザクションがデータベースのデフォルトのロックを明示的に無視する場合です。デッドロックはデータベースのパフォーマンスに影響を与える可能性があるため、ロックを監視できるスクリプトとビューが用意されています。

ロックを監視するには:

1. ロック・ビューを作成する`catblock.sql`を実行します。
2. `catblock.sql`によって作成されたビューを使用して、システムでロックを待機中のセッションと、それらが待機しているロックをツリー形式で表示するために、`utllockt.sql`スクリプトを実行します。

スクリプト・ファイルの場所は、オペレーティング・システムごとに異なります。

関連項目:

- 「パフォーマンス監視データ・ディクショナリ・ビュー」
- ロックの詳細は、『Oracle Database概要』を参照してください。

親トピック: パフォーマンスの監視

8.2.2 待機イベントの監視について

待機イベントとは、イベントの完了を待機してから処理を続行する必要があることを示す統計で、サーバー・プロセスによって増分されます。セッションは、入力の待機、オペレーティング・システムによるディスクへの書込みなどのサービス完了の待機、ロックまたはラッチの待機など、様々な理由で待機することがあります。

セッションがリソースを待機している場合、そのセッションでは効率的な作業が実行されていません。待機の多くはソースに関係しています。待機イベントのデータによって、ラッチの競合、バッファの競合、I/Oの競合など、パフォーマンスに影響を与える可能性がある様々な問題の兆候が明らかになります。

このために、待機イベントの統計を表示するビューが用意されています。これらのビューの説明およびインスタンスをチューニングする際のビューの役割については、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

親トピック: パフォーマンスの監視

8.2.3 パフォーマンス監視のデータ・ディクショナリ・ビュー

Oracle Databaseインスタンスを監視するために、データ・ディクショナリ・ビューのセットを問い合せることができます。

これらのビューは一般的なものです。プロセス固有の他のビューについては、そのプロセスに関する項を参照してください。

ビュー	説明
V$LOCK	現在 Oracle Database によって保持されているロックと、未処理のロック要求またはラッチ要求が表示されます。
DBA_BLOCKERS	別のセッションが待機しているオブジェクトのロックを保持しているセッションが表示されます。
DBA_WAITERS	ロックされているオブジェクトを待機しているセッションが表示されます。
DBA_DDL_LOCKS	データベース内のすべての DDL ロックおよび DDL ロックに対する未処理の要求すべてが表示されます。
DBA_DML_LOCKS	データベース内のすべての DML ロックおよび DML ロックに対する未処理の要求すべてが表示されます。
DBA_LOCK	データベース内のすべてのロックまたはラッチ、およびロックまたはラッチに対する未処理の要求すべてが表示されます。
DBA_LOCK_INTERNAL	保持しているロックまたはラッチごとに 1 行、ロックまたはラッチの未処理要求ごとに 1 行の情報が表示されます。

ビュー	説明
V$LOCKED_OBJECT	システム上のすべてのトランザクションが獲得したすべてのロックがリストされます。
V$SESSION_WAIT	アクティブ・セッションが待機しているリソースまたはイベントが表示されます。
V$SYSSTAT	セッション統計情報が含まれています。
V$RESOURCE_LIMIT	一部のシステム・リソースについて、現行および最大のグローバル・リソース使用率が表示されます。
V$SQLAREA	共有 SQL 領域に関する統計情報が、SQL 文字列ごとに 1 行ずつ含まれています。また、メモリー内にあり、解析済で、実行準備のできている SQL 文に関する統計情報も提供します。
V$LATCH	非親ラッチの統計情報と、親ラッチのサマリー統計情報が含まれています。

親トピック: パフォーマンスの監視

8.3 隔離されたオブジェクトの監視

オブジェクトの隔離により、破損したリカバリ不能なオブジェクトが存在する場合でも、Oracleデータベースが機能できるようにすることができます。V$QUARANTINEビューに、隔離されたオブジェクトの情報が含まれています。

- オブジェクトの隔離について
  オブジェクトの隔離は、エラーが発生したオブジェクトを隔離し、システムへの影響についてオブジェクトを監視します。
- 隔離されたオブジェクトの表示
  V$QUARANTINEビューに、現在隔離されているオブジェクトの情報が格納されています。

親トピック: データベースの監視

8.3.1 オブジェクトの隔離について

オブジェクトの隔離は、エラーが発生したオブジェクトを隔離し、システムへの影響についてオブジェクトを監視します。

ORA-00600およびORA-07445などのいくつかのOracle Databaseエラーにより、通常ではプロセスが終了し、これによりデータベースが終了する可能性があります。このようなエラーが発生した場合、データベースの実行を継続できるように、オブジェクトの隔離はエラーの発生したリソースの隔離を試みます。データベースの残りの部分に影響を与えないように、リソースはメモリーに隔離されます。V$QUARANTINEビューに、現在隔離されているオブジェクトの情報が格納されています。

ほとんどのデータベース・リソースは、データベースを終了する可能性があるエラーを発生することがあります。たとえば、ライブラリ・キャッシュ・メモリー・オブジェクトは、このようなエラーを発生することがあります。

マルチテナント環境では、場合によっては、マルチテナント・コンテナ・データベース(CDB)がオブジェクトの隔離を使用して、CDBを終了せずに、重大なエラーの発生したプラガブル・データベース(PDB)を隔離し、終了することもできます。

通常、隔離されたリソースは、データベースが再起動するまで隔離されたままです。リソースがCDBにPDBのために隔離されている場合、PDBがクローズし、再オープンするまで、リソースは隔離されています。

親トピック: 隔離されたオブジェクトの監視

8.3.2 隔離されたオブジェクトの表示

V$QUARANTINEビューに、現在隔離されているオブジェクトの情報が格納されています。

1. データベースに管理ユーザーとして接続します。
2. V$QUARANTINEビューを問い合せます。

例8-1 V$QUARANTINEビューの問合せ

この問合せは、現在隔離されているリソースを表示します。

```
COLUMN OBJECT FORMAT A10
COLUMN ADDRESS FORMAT A10
COLUMN BYTES FORMAT 999999999
COLUMN ERROR FORMAT A20
COLUMN TIMESTAMP FORMAT A20
SELECT OBJECT, ADDRESS, BYTES, ERROR, TIMESTAMP
   FROM V$QUARANTINE;
```

出力は次のようになります。

```
OBJECT     ADDRESS          BYTES ERROR                TIMESTAMP
---------- ---------- ---------- -------------------- --------------------
session    0000000078       9528 ORA-00600: internal  16-SEP-15 01.17.42.2
           B54BC8                error code, argument 85878 PM -07:00
                                 s: [12345], [], [],
                                 [], [], [], [], [],
                                 [], [], [], []
```

この出力は、隔離されたリソースについて次のことを示しています。

- リソースの名前は「session」です。
- 隔離されているメモリー領域の開始アドレスは0000000078B54BC8です。通常、これは、この例のセッションなどのリソースのアドレスです。
- リソースは、隔離中に9528バイトのメモリーを使用しています。
- リソースが隔離される原因となったエラー・メッセージは、ORA-00600内部エラー・コードです。
- タイムスタンプがエラーの日付と時間を示しています。

親トピック: 隔離されたオブジェクトの監視

9 問題の診断と解決

Oracle Databaseには、データベースの問題を診断して解決するために、診断データの収集と管理ための高度な障害診断インフラストラクチャが含まれています。診断データには、以前のリリースにも含まれていたトレース・ファイル、ダンプおよびコア・ファイルに加えて、顧客やOracleサポート・サービスが問題を迅速かつ効率的に識別、調査、追跡、解決できる新しいタイプの診断データが含まれています。

- Oracle Databaseの障害診断インフラストラクチャについて
  Oracle Databaseには、データベースの問題を予防、検出、診断および解決するための障害診断インフラストラクチャが含まれています。
- 問題の調査、報告および解決について
  Enterprise Managerサポート・ワークベンチ(サポート・ワークベンチ)により、問題(クリティカル・エラー)を調査して報告し、場合によっては、解決できます。実行する必要がある一般的なタスクのセットをまとめた「ロードマップ」を使用できます。
- 問題の診断
  この項では、Oracleデータベースでの問題を診断するための様々な方法について説明します。
- 問題の報告
  Enterprise Managerサポート・ワークベンチ(サポート・ワークベンチ)を使用して、カスタム・インシデント・パッケージを作成、編集およびアップロードできます。カスタム・インシデント・パッケージにより、Oracleサポート・サービスに送信する診断データを詳細に制御できます。
- 問題の解決
  この項では、SQL修復アドバイザやデータ・リカバリ・アドバイザなどのアドバイザ・ツール、およびリソース・マネージャおよび関連APIなどのリソース管理ツールを使用してデータベースの問題を解決する方法について説明します。

親トピック: 基本データベース管理

9.1 Oracle Databaseの障害診断インフラストラクチャについて

Oracle Databaseには、データベースの問題を防止、検出、診断および解決するための障害診断性インフラストラクチャが含まれています。

- 障害診断インフラストラクチャの概要
  障害診断インフラストラクチャは、問題の防止、検出、診断および解決に役立ちます。特に対象とする問題はクリティカル・エラーで、これには、コードの不具合、メタデータの破損、顧客データの破損によって生じるエラーなどがあります。
- インシデントと問題
  問題とは、データベース・インスタンス、Oracle Automatic Storage Management (Oracle ASM)インスタンス、またはその他のOracle製品やコンポーネントにおけるクリティカル・エラーです。インシデントとは、問題の1回の発生です。
- 障害診断インフラストラクチャのコンポーネント
  障害診断インフラストラクチャは、自動診断リポジトリ(ADR)、様々なログ、トレース・ファイル、Enterprise Managerサポート・ワークベンチ、およびADRCIコマンドライン・ユーティリティを含む複数のコンポーネントで構成されています。
- 自動診断リポジトリの構造、内容および場所
  自動診断リポジトリ(ADR)は、データベースの外部に格納されているディレクトリ構造です。そのため、データベースが停止しているときに問題の診断に使用できます。

親トピック: 問題の診断と解決

9.1.1 障害診断インフラストラクチャの概要

障害診断インフラストラクチャは、問題の防止、検出、診断および解決に役立ちます。特に対象とする問題はクリティカル・エラーで、これには、コードの不具合、メタデータの破損、顧客データの破損によって生じるエラーなどがあります。

クリティカル・エラーが発生すると、そのエラーにはインシデント番号が割り当てられ、エラーの診断データ(トレース・ファイルなど)が即座に取得され、その番号でタグ付けされます。このデータは、自動診断リポジトリ(ADR) (データベースの外部のファイルベースのリポジトリ)に格納され、後からインシデント番号によって取得して分析できます。

障害診断インフラストラクチャの目的は次のとおりです。

- 初期障害の診断
- 問題の防止
- 問題が検出された後の損害および中断の抑制
- 問題の診断時間の短縮
- 問題の解決時間の短縮
- 顧客とOracleサポートのやり取りの簡略化

これらの目的を達成するための主要なテクノロジは、次のとおりです。

- 初期障害発生時の診断データの自動取得—クリティカル・エラーの場合は、初期障害時にエラー情報を取得することによって、問題の早期解決と停止時間の短縮の可能性が飛躍的に増大します。常時機能しているメモリーベースのトレース・システムは、多くのデータベース・コンポーネントから診断データを事前に収集するため、問題の根本原因を特定するのに役立ちます。このような事前の診断データは、飛行機の「ブラック・ボックス」フライト・レコーダで収集されるデータに類似しています。問題が検出されると、アラートが生成されて、障害診断インフラストラクチャがアクティブになり、診断データが取得されて格納されます。データはデータベース外部のリポジトリに格納され(このため、データベースが停止しているときに使用できます)、コマンドライン・ユーティリティおよびOracle Enterprise Manager Cloud Control (Cloud Control)を使用して容易にアクセスできます。
- 標準化されたトレース形式—すべてのデータベース・コンポーネント間でトレース形式を標準化することによって、DBAおよびOracleサポート・サービス担当者は単一のツール・セットを使用して問題を分析できます。問題の診断はさらに容易となり、停止時間は短縮されます。
- ヘルス・チェック: クリティカル・エラーが検出されると、障害診断インフラストラクチャでは1つ以上のヘルス・チェックが実行され、クリティカル・エラーの詳細な分析が実行されます。ヘルス・チェックの結果は、エラーについて収集された他の診断データに追加されます。各ヘルス・チェックでは、データ・ブロックの破損、UNDOおよびREDOの破損、データ・ディクショナリの破損などが検出されます。DBAは、定期的にまたは必要に応じてこれらのヘルス・チェックを手動で起動できます。
- インシデント・パッケージ化サービス(IPS)およびインシデント・パッケージ: IPSを使用すると、クリティカル・エラーに関する診断データ(トレース、ダンプ、ヘルス・チェック・レポートなど)を自動的かつ容易に収集し、それらのデータをzipファイルにパッケージ化してOracleサポートに転送できます。クリティカル・エラーに関するすべての診断データは、そのエラーのインシデント番号でタグ付けされるため、分析に必要なファイルを判別するためにトレース・ファイルやその他のファイルを検索する必要はなく、インシデント・パッケージング・サービスにより必要なファイルが自動的に特定され、zipファイルに追加されます。zipファイルを作成する前に、IPSはまず、インシデント・パッケージという中間論理構造(パッケージ)に診断データを収集します。パッケージは自動診断リポジトリに格納されます。必要に応じて、この中間論理構造にアクセスして、その内容を表示および修正したり、詳細な診断データをいつでも追加または削除でき、さらに準備ができたらパッケージ

からzipファイルを作成できます。これらのステップが完了すれば、Oracleサポートにアップロードするzipファイルの準備は完了です。

- データ・リカバリ・アドバイザ: データ・リカバリ・アドバイザはデータベースのヘルス・チェックおよびRMANに統合され、データ破損の問題の表示、各問題の程度の評価(クリティカル、高優先度、低優先度)、問題の影響の説明、修復オプションの推奨、顧客が選択したオプションの実行可能性チェック、および修復プロセスの自動化を実行します。
- SQLテスト・ケース・ビルダー: 多くのSQL関連の問題では、再現可能なテスト・ケースの取得が、問題の迅速な解決にとって重要な要因となります。問題とそれが発生した環境に関する可能なかぎりの情報を収集することは困難で時間がかかる場合がありますが、SQLテスト・ケース・ビルダーは、この収集作業を自動化します。迅速に収集された情報をOracleサポート・サービスにアップロードすると、サポート担当者は問題を簡単かつ正確に再現できます。

親トピック: Oracle Databaseの障害診断インフラストラクチャについて

9.1.2 インシデントと問題

問題とは、データベース・インスタンス、Oracle Automatic Storage Management(Oracle ASM)インスタンス、またはその他のOracle製品やコンポーネントにおけるクリティカル・エラーです。インシデントとは、問題の1回の発生です。

- インシデントおよび問題について
  クリティカル・エラーの診断と解決を容易にするために、障害診断インフラストラクチャには、問題とインシデントというOracle Databaseの2つの概念が導入されています。
- インシデントのフラッド制御
  短時間に数十、場合によっては数百ものインシデントが生成されることも考えられます。その場合、生成される診断データが多すぎて、ADRの領域が大量に消費され、問題の診断および解決に手間がかかることにもなります。このような理由から、障害診断インフラストラクチャでは、特定のしきい値に達すると、インシデントの生成にフラッド制御が適用されます。
- トポロジ全体に関連する問題
  データベース・インスタンスで識別されるすべての問題については、診断能力フレームワークによって、Oracle Databaseインストールのトポロジ全体に関連する問題を識別できます。

親トピック: Oracle Databaseの障害診断インフラストラクチャについて

9.1.2.1 インシデントおよび問題について

クリティカル・エラーの診断と解決を容易にするために、障害診断インフラストラクチャには、問題とインシデントというOracle Databaseの概念が導入されています。

問題とは、データベース・インスタンス、Oracle Automatic Storage Management(Oracle ASM)インスタンス、またはその他のOracle製品やコンポーネントにおけるクリティカル・エラーです。クリティカル・エラーには、内部エラー(ORA-00600など)やその他の深刻なエラー(ORA-07445(オペレーティング・システム例外)またはORA-04031(共有プールのメモリー不足))が含まれます。問題はADRで追跡されます。各問題には、問題を説明する文字列である問題キーがあります。これには、エラー・コード(ORA 600など)、場合によってはさらに1つ以上のエラー・パラメータが含まれます。

インシデントとは、問題の1回の発生です。問題(クリティカル・エラー)が複数回発生すると、インシデントはそれぞれの発生分に対して作成されます。インシデントには日時が記録され、自動診断リポジトリ(ADR)で追跡されます。各インシデントは、ADR内で一意の数値であるインシデントIDによって識別されます。インシデントが発生すると、データベースでは次の処理が実行されます。

- アラート・ログにエントリを作成します。

- インシデント・アラートをCloud Controlに送信します。
- インシデントに関する初期障害診断データをダンプ・ファイルの形式(インシデント・ダンプ)で収集します。
- インシデント・ダンプをインシデントIDにタグ付けします。
- そのインシデント用に作成されたADRサブディレクトリにインシデント・ダンプを格納します。

通常、クリティカル・エラーの診断と解決は、インシデント・アラートから開始されます。インシデント・アラートは、Cloud Controlのデータベース・ホームページまたはOracle Automatic Storage Managementのホームページに表示されます。また、データベース・ホームページは、Oracle ASMインスタンスやその他のOracle製品またはコンポーネントのクリティカル・アラートを「関連アラート」セクションに表示します。アラートを表示した後、問題および関連するインシデントは、Cloud ControlまたはADRCIコマンドライン・ユーティリティを使用して表示できます。

関連項目:

- 「サポート・ワークベンチを使用した問題の表示」
- 「問題の調査、報告および解決について」
- 「ADRCIコマンドライン・ユーティリティ」

親トピック: インシデントと問題

9.1.2.2 インシデントのフラッド制御

問題によって、短期間の間に数十、場合によっては数百ものインシデントが生成されることも考えられます。その場合、生成される診断データが多すぎて、ADRの領域が大量に消費され、問題の診断および解決に手間がかかることにもなります。このような理由から、障害診断インフラストラクチャでは、特定のしきい値に達すると、インシデントの生成にフラッド制御が適用されます。

フラッド制御されたインシデントとは、アラート・ログ・エントリは作成されてADRに記録されるが、インシデント・ダンプは生成されないインシデントです。フラッド制御インシデントは、診断データによりシステムに過大な負荷をかけることなく、クリティカル・エラーが継続的に発生していることを知らせる手段となります。Cloud ControlまたはADRCIのコマンドライン・ユーティリティを使用してインシデントを表示するときは、フラッド制御されたインシデントを表示するか非表示にするかを選択できます。

インシデントのフラッド制御のしきい値レベルは事前定義されており、変更できません。しきい値レベルは次のように定義されています。

- 1時間に同じ問題キーに対して5つのインシデントが発生した場合、その問題キーに対する後続のインシデントはフラッド制御されます。その問題キーに対するインシデントの通常(非フラッド制御)の記録は、次の1時間から再開されます。
- 1日に同じ問題キーに対して25のインシデントが発生した場合、その問題キーに対する後続のインシデントはフラッド制御されます。その問題キーに対するインシデントの通常の記録は、翌日から再開されます。

さらに、1時間に同じ問題キーに対して50のインシデントが発生した場合、または1日に同じ問題キーに対して250のインシデントが発生した場合、その問題キーに対する後続のインシデントはADRに記録されません。この場合は、後続のインシデントが記録されないことを示すメッセージがアラート・ログに書き込まれます。問題キーに対してインシデントが継続して生成されている間は、1時間または1日が経過するまで、10分ごとにこのメッセージがアラート・ログに追加されます。1時間または1日が経過すると、その問題キーに対するインシデントの通常の記録が再開されます。

親トピック: インシデントと問題

9.1.2.3 トポロジ全体に関連する問題

データベース・インスタンスで識別されるすべての問題については、診断能力フレームワークによって、Oracle Databaseインストールのトポロジ全体に関連する問題が識別できます。

単一インスタンス環境では、関連する問題はローカルのOracle ASMインスタンスで識別できます。Oracle RAC環境では、関連する問題はデータベース・インスタンスまたは他のノード上のOracle ASMインスタンスで識別できます。問題を調査する場合、関連する問題の情報を表示して収集できます。

問題が指定期間内に発生したり、同じ実行コンテキスト識別子を共有している場合、その問題は元の問題に関連しています。実行コンテキスト識別子(ECID)は、後からデータを取得するためにOracle Databaseにコールするオラクル Fusion Middlewareへのコールなど、Oracleソフトウェア・スタックを介した単一のコールに対するタグ付けおよび追跡のために使用されるグローバルに一意の識別子です。通常、ECIDは中間層で生成され、Oracle Call Interface (OCI)の属性としてデータベースに渡されます。Oracleソフトウェア・スタックの複数の層に単一のコールによる障害が存在する場合、生成された問題には同じECIDがタグ付けされるため、相関関係を確認できます。さらに、源になっている問題が発生した層を判別できます。

親トピック: インシデントと問題

9.1.3 障害診断インフラストラクチャのコンポーネント

障害診断インフラストラクチャは、自動診断リポジトリ(ADR)、様々なログ、トレース・ファイル、Enterprise Managerサポート・ワークベンチ、およびADRCIコマンドライン・ユーティリティを含む複数のコンポーネントで構成されています。

- 自動診断リポジトリ(ADR)
  ADRは、データベース診断データ(トレース、ダンプ、アラート・ログ、状態モニターのレポートなど)のファイルベース・リポジトリです。複数のインスタンスや製品にまたがる一元化されたディレクトリ構造を持っています。
- アラート・ログ
  アラート・ログは、メッセージとエラーの発生順のログのXMLファイルです。
- トレース・ファイル、ダンプおよびコア・ファイル
  トレース・ファイル、ダンプおよびコア・ファイルには、問題の調査に使用する診断データが含まれています。これらはADRに格納されます。
- DDLログ
  データ定義言語(DDL)ログは、形式および基本動作がアラート・ログと同じファイルですが、データベースによって発行されたDDL文のみが含められます。
- デバッグ・ログ
  Oracle Databaseコンポーネントは、検出しているコンポーネントの正しい動作を妨げないが、通常とは異なる条件、状態またはイベントを検出できます。コンポーネントでは、これらの条件、状態またはイベントについての警告を発行できます。デバッグ・ログは、これらの警告を記録するファイルとなります。
- ADRのその他の内容
  ADRには、前の各項で説明したファイルに加えて、状態モニター・レポート、データ修復レコード、SQLテスト・ケース、インシデント・パッケージなどが格納されます。これらのコンポーネントについては、この章で後述します。
- Enterprise Managerサポート・ワークベンチ
  Enterprise Managerサポート・ワークベンチ(サポート・ワークベンチ)は、問題(クリティカル・エラー)の調査、レポート、および場合によっては修復を、すべて使いやすいグラフィカル・インタフェースによって実現するファシリティです。
- ADRCIコマンドライン・ユーティリティ
  ADRコマンド・インタプリタ(ADRCI)は、問題の調査、ヘルス・チェック・レポートの表示、および初期障害診断データのパッケージ化を、すべてコマンドライン環境内で実行できるようにするユーティリティです。

親トピック: Oracle Databaseの障害診断インフラストラクチャについて

9.1.3.1 自動診断リポジトリ(ADR)

ADRは、データベース診断データ(トレース、ダンプ、アラート・ログ、状態モニターのレポートなど)のファイルベース・リポジトリです。複数のインスタンスや製品にまたがる一元化されたディレクトリ構造を持っています。

データベース、Oracle Automatic Storage Management (Oracle ASM)、リスナー、Oracle ClusterwareなどのOracle製品またはコンポーネントでは、すべての診断データをADRに格納します。各製品のインスタンスはそれぞれ、診断データをADR内の独自のホーム・ディレクトリの下に配置します。たとえば、共有記憶域とOracle ASMを使用するOracle Real Application Clusters環境では、各データベース・インスタンスと各Oracle ASMインスタンスにADRホーム・ディレクトリがあります。ADRの統一されたディレクトリ構造、製品およびインスタンス間で一貫性のある診断データ形式、および統一されたツール・セットにより、ユーザーおよびOracleサポートは、複数のインスタンス間の診断データの相関関係を確認して分析できます。Oracle Clusterwareの場合、クラスタ内の各ホスト・ノードにADRホーム・ディレクトリが1つあります。

ノート:

アラート・ログを含むすべての診断データが ADR に格納されるため、初期化パラメータの BACKGROUND_DUMP_DEST および USER_DUMP_DEST が非推奨になりました。これらのパラメータは、ADR の場所を指定する初期化パラメータ DIAGNOSTIC_DEST に置き換えられています。

関連トピック

- 自動診断リポジトリの構造、内容および場所

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.3.2 アラート・ログ

アラート・ログは、メッセージとエラーの発生順のログのXMLファイルです。

各ADRホームにアラート・ログが1つあります。各アラート・ログは、データベース、Oracle ASM、リスナー、Oracle Clusterwareなどのコンポーネント・タイプに固有です。

データベースの場合、アラート・ログには次のものに関するメッセージが含まれます。

- クリティカル・エラー(インシデント)
- 管理操作(データベースの起動と停止、データベースのリカバリ、表領域の作成や削除など)。
- マテリアライズド・ビューの自動リフレッシュ中に発生したエラー
- その他のデータベース・イベント

アラート・ログは、Cloud ControlおよびADRCIユーティリティを使用して、テキスト形式(XMLタグは削除されます)で表示できます。また、下位互換性のためにADRに格納されたテキスト形式のアラート・ログもあります。ただし、テキスト形式は構造化されておらず、リリースごとに変更される場合があるため、アラート・ログの内容の解析はXML形式で行うことをお薦めします。

関連項目:

- 「ADRCIコマンドライン・ユーティリティ」

- 「アラート・ログの表示」

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.3.3 トレース・ファイル、ダンプおよびコア・ファイル

トレース・ファイル、ダンプおよびコア・ファイルには、問題の調査に使用する診断データが含まれています。これらはADRに格納されます。

- トレース・ファイル
  各サーバー・プロセスとバックグラウンド・プロセスは、対応するトレース・ファイルに情報を書き込むことができます。トレース・ファイルはプロセスの存続期間にわたって定期的に更新され、プロセスの環境、ステータス、アクティビティおよびエラーに関する情報を格納できます。さらに、プロセスでクリティカル・エラーが検出されると、そのエラーに関する情報が関連のトレース・ファイルに書き込まれます。
- ダンプ
  ダンプは、特殊なタイプのトレース・ファイルです。ダンプでは、通常、1つのイベント(インシデントなど)に対応して1回だけ診断データが出力されますが、トレースでは、多くの場合、診断データが連続して出力されます。
- コア・ファイル
  コア・ファイルには、メモリー・ダンプがポート固有の形式ですべてバイナリで格納されます。

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.3.3.1 トレース・ファイル

各サーバー・プロセスとバックグラウンド・プロセスは、対応するトレース・ファイルに情報を書き込むことができます。トレース・ファイルはプロセスの存続期間にわたって定期的に更新され、プロセスの環境、ステータス、アクティビティおよびエラーに関する情報を格納できます。さらに、プロセスでクリティカル・エラーが検出されると、そのエラーに関する情報が関連のトレース・ファイルに書き込まれます。

SQLトレース機能でも、個別のSQL文に関するパフォーマンス情報を提供するトレース・ファイルが作成されます。SQLトレース機能は、セッションまたはインスタンスに対して使用可能にできます。

トレース・ファイルの名前は、プラットフォームに依存します。通常、データベース・バックグラウンド・プロセスのトレース・ファイル名には、Oracle SID、バックグラウンド・プロセス名およびオペレーティング・システムのプロセス番号が含まれ、サーバー・プロセスのトレース・ファイル名には、Oracle SID、「ora」の文字列、およびオペレーティング・システムのプロセス番号が含まれます。ファイル拡張子は`.trc`です。たとえば、サーバー・プロセスのトレース・ファイル名は、orcl_ora_344.trcのようになります。トレース・ファイルは、対応するトレース・メタデータ(`.trm`)ファイルを伴う場合があり、このファイルにはトレース・ファイルの構造情報が含まれていて、検索やナビゲーションに使用されます。

Oracle Databaseには、トレース・ファイルの分析に役立つツールが用意されています。アプリケーション・トレース機能、SQLトレース機能およびトレース・ツールの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

関連トピック

- トレース・ファイルの検索
- アプリケーション・トレースの実行

親トピック: トレース・ファイル、ダンプおよびコア・ファイル

9.1.3.3.2 ダンプ

ダンプは、特殊なタイプのトレース・ファイルです。ダンプでは、通常、1つのイベント(インシデントなど)に対応して1回だけ診断データが出力されますが、トレースでは、多くの場合、診断データが連続して出力されます。

インシデントが発生すると、データベースは、そのインシデント用に作成されたインシデント・ディレクトリに1つ以上のダンプを書き込みます。インシデントのダンプには、ファイル名にインシデント番号も付きます。

親トピック: トレース・ファイル、ダンプおよびコア・ファイル

9.1.3.3.3 コア・ファイル

コア・ファイルには、メモリー・ダンプがポート固有の形式ですべてバイナリで格納されます。

コア・ファイル名は、文字列「core」とオペレーティング・システム・プロセスIDで構成されます。コア・ファイルを使用するのはOracleサポート・サービスのエンジニアのみです。プラットフォームによっては、コア・ファイルがない場合があります。

親トピック: トレース・ファイル、ダンプおよびコア・ファイル

9.1.3.4 DDLログ

データ定義言語(DDL)ログは、形式および基本動作がアラート・ログと同じファイルですが、データベースによって発行されたDDL文のみが含められます。

DDLログは、RDBMSコンポーネントのみを対象として、およびENABLE_DDL_LOGGING初期化パラメータがTRUEに設定されている場合にのみ、作成されます。このパラメータがFALSEに設定されている場合、DDL文はいずれのログにも含められません。

DDLログには、データベースによって発行されたDDL文ごとに1つのログ・レコードが含められます。DDLログは、IPSインシデント・パッケージに含まれています。

同じ情報が含まれている2つのDDLログが存在します。1つはXMLファイルで、もう1つはテキスト・ファイルです。DDLログはADRホームのlog/ddlサブディレクトリに格納されます。

関連項目:

ENABLE_DDL_LOGGING初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.3.5 デバッグ・ログ

Oracle Databaseコンポーネントでは、通常と異なるものの、検出しているコンポーネントの正しい操作を妨げない、条件、状態またはイベントを検出できます。コンポーネントでは、これらの条件、状態またはイベントについての警告を発行できます。デバッグ・ログは、これらの警告を記録するファイルとなります。

デバッグ・ログに記録されるこれらの警告は重大ではなく、インシデントの生成やアラート・ログへの書込みは必要ではありません。これらの警告は問題が発生した場合の診断に必要になる可能性があるため、ログ・ファイルにレコードが生成されます。

デバッグ・ログの形式および基本動作はアラート・ログと同じですが、修正が必要な可能性のある、考えられる問題に関する情報のみが含められます。

デバッグ・ログでは、アラート・ログとトレース・ファイルの情報量が削減されています。また、デバッグ情報の可視性が高められています。

デバッグ・ログは、IPSインシデント・パッケージに含まれています。デバッグ・ログの内容は、Oracleサポート用です。データベース管理者は、デバッグ・ログを直接使用しないでください。

ノート:

Oracle Database 12c 以降には個別のデバッグ・ログがあるため、アラート・ログおよびトレース・ファイルは簡素化されています。これらに格納される、デバッグ・ログに記録されている警告のタイプは削減されました。

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.3.6 ADRのその他の内容

ADRには、前の各項で説明したファイルに加えて、状態モニター・レポート、データ修復レコード、SQLテスト・ケース、インシデント・パッケージなどが格納されます。これらのコンポーネントについては、この章で後述します。

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.3.7 Enterprise Managerサポート・ワークベンチ

Enterprise Managerサポート・ワークベンチ(サポート・ワークベンチ)は、使いやすいグラフィカル・インタフェースで問題(クリティカル・エラー)を調査およびレポートできる機能で、場合によってはその問題を修復することもできます。

サポート・ワークベンチでは、セルフサービス方式で、初期障害の診断データの収集、サポート・リクエスト番号の取得、および診断データのOracleサポートへのアップロードを最小限の労力と非常に短い時間で行えるので、問題の解決にかかる時間を短縮できます。また、サポート・ワークベンチでは、SQL関連の問題やデータ破損の問題などを解決できるOracleアドバイザへの容易なアクセスを推奨および提供します。

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.3.8 ADRCIコマンドライン・ユーティリティ

ADRコマンド・インタプリタ(ADRCI)は、問題の調査、ヘルス・チェック・レポートの表示、および初期障害診断データのパッケージ化を、すべてコマンドライン環境内で実行できるユーティリティです。

その後、このパッケージをOracleサポートにアップロードできます。また、ADRCIを使用すると、ADRに格納されているトレース・ファイルの名前の表示、およびアラート・ログの表示(XMLタグは削除されます)を、フィルタ処理の有無を選択して実行できます。

ADRCIの詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: 障害診断インフラストラクチャのコンポーネント

9.1.4 自動診断リポジトリの構造、内容および場所

自動診断リポジトリ(ADR)は、データベースの外部に格納されているディレクトリ構造です。そのため、データベースが停止しているときに問題の診断に使用できます。

ADRのルート・ディレクトリはADRベースと呼ばれます。その場所は`DIAGNOSTIC_DEST`初期化パラメータによって設定されます。このパラメータを省略するかNULLのままにすると、データベースでは起動時に`DIAGNOSTIC_DEST`を次のように設定します。

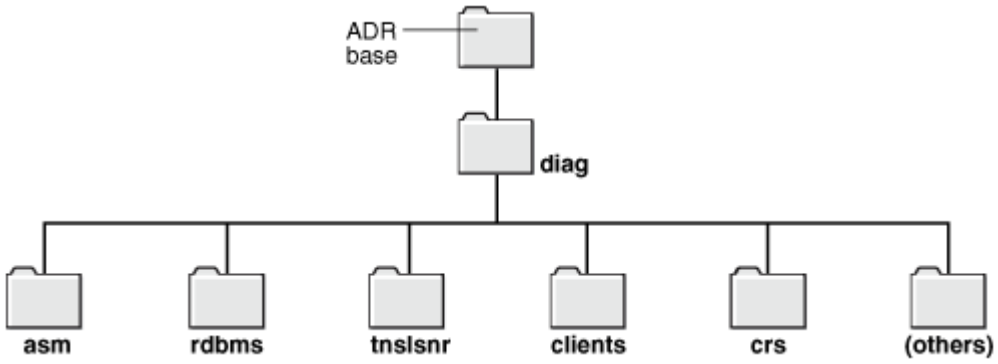
- 環境変数`ORACLE_BASE`が設定されている場合、`DIAGNOSTIC_DEST`は`ORACLE_BASE`で指定されたディレクトリに設定されます。
- 環境変数`ORACLE_BASE`が設定されていない場合、`DIAGNOSTIC_DEST`はORACLE_HOME/logに設定されます。

ADRベース内では、複数のADRホームが存在することがあり、各ADRホームは、特定のOracle製品やコンポーネントの特定

のインスタンスに対するすべての診断データ(トレース、ダンプ、アラート・ログなど)のルート・ディレクトリです。たとえば、Oracle ASMを使用するOracle Real Application Clusters環境では、各データベース・インスタンス、Oracle ASMインスタンスおよびリスナーにADRホームがあります。

ADRホームは、製品またはコンポーネントのタイプに基づいて名前が付けられたADRベース・サブディレクトリにあります。図9-1は、これらのトップレベルのサブディレクトリを示しています。

図9-1 ADR内の製品/コンポーネント・タイプのサブディレクトリ



ノート:

構成に応じて、ADR に追加のサブディレクトリが作成される場合もあります。一部の製品では、期限切れの診断データが ADR から自動的にパージされます。その他の製品の場合は、ADRCI ユーティリティ PURGE コマンドを一定の間隔で使用して、期限切れの診断データをパージできます。

各ADRホームの場所は、ADRの基本ディレクトリから始まる次のパスで指定されます。

```
diag/product_type/product_id/instance_id
```

例として、表9-1に、Oracle Databaseインスタンスの様々なパス・コンポーネントの値を示します。

表9-1 Oracle DatabaseのADRホームのパス・コンポーネント

パス・コンポーネント	Oracle Databaseの値
product_type	rdbms
product_id	DB_UNIQUE_NAME
instance_id	SID

たとえば、SIDとデータベースの一意の名前が両方ともorclbiのデータベースと同じ場合、ADRホームの場所は次のようになります。

```
ADR_base/diag/rdbms/orclbi/orclbi/
```

同様に、単一インスタンス環境におけるOracle ASMインスタンスへのADRホームのパスは、次のとおりです。

```
ADR_base/diag/asm/+asm/+asm/
```

ADRホームのサブディレクトリ

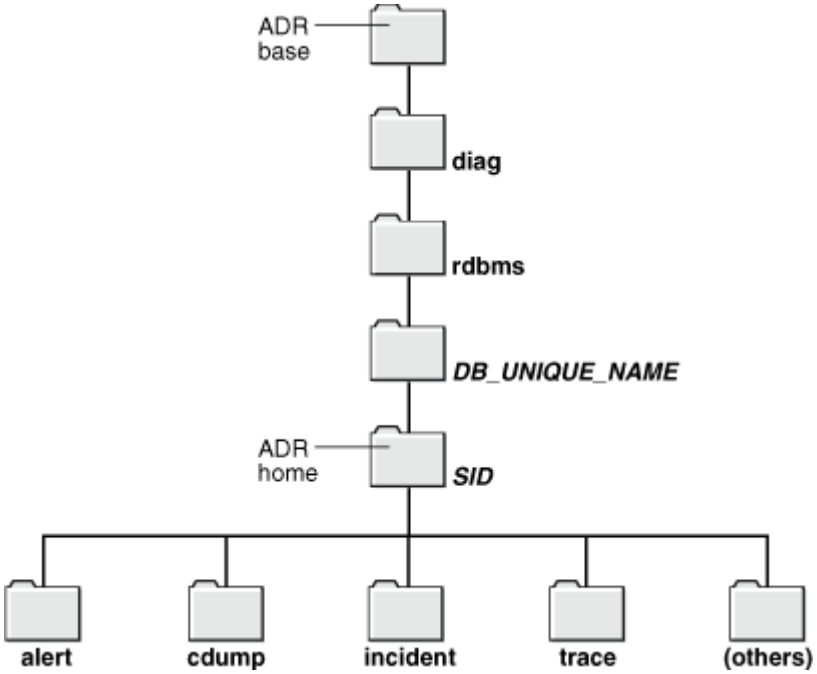
各ADRホーム・ディレクトリ内には、診断データが含まれるサブディレクトリがあります。表9-2に、これらのサブディレクトリの一部とその内容を示します。

表9-2 ADRホーム・サブディレクトリ

サブディレクトリ名	目次
alert	XML 形式のアラート・ログ。
cdump	コア・ファイル
incident	複数のサブディレクトリ。各サブディレクトリは特定のインシデントにちなんで名付けられ、それぞれにそのインシデントにのみ関係するダンプが含められる
trace	バックグラウンドおよびサーバー・プロセスのトレース・ファイル、SQL トレース・ファイル、およびテキスト形式のアラート・ログ。
(その他)	ADR ホームのその他のサブディレクトリには、インシデント・パッケージ、状態モニター・レポート、アラート以外のログ(DDL ログやデバッグ・ログなど)およびその他の情報が格納される。

図9-2に、データベース・インスタンスのADRのディレクトリ階層全体を示します。

図9-2 データベース・インスタンスのADRディレクトリ構造



Oracle Clusterware環境でのADR

Oracle ClusterwareではADRを使用し、独自のOracleホームおよびOracleベースがあります。Oracle ClusterwareのADRディレクトリ構造は、データベース・インスタンスのものとは異なります。1つのシステムではOracle Clusterwareのインスタンスは1つしかないため、Clusterware ADRホームでは、区別子としてシステムのホスト名のみを使用します。

Oracle Clusterwareが構成されると、ADRホームでは、製品タイプとインスタンスIDの両方にcrsを使用し、製品IDにはシステム・ホスト名を使用します。したがって、dbprod01という名前のホストでは、CRS ADRホームは次のようになります。

```
ADR_base/diag/crs/dbprod01/crs/
```

関連項目:

Oracle Clusterware管理およびデプロイメント・ガイド

Oracle Real Application Clusters環境でのADR

Oracle Real Application Clusters(Oracle RAC)環境では、ADRベースはノードごとに独自のローカル記憶域に設定するか、ADRベースは共有記憶域に設定できます。ADRCIを使用して、すべてのインスタンスから集計された診断データを1つのレポートに表示できます。

Oracle ClientにおけるADR

インストールされたOracle Clientには、Oracle Clientコンポーネントのクリティカルな障害に関連する診断データのADRが含まれます。Oracle ClientとともにADRCIユーティリティがインストールされており、これにより、診断データを確認して、Oracleサポートにアップロードするためにパッケージ化できます。

V$DIAG_INFOビューを使用したADRの場所の表示

V$DIAG_INFOビューには、現在のOracle Databaseインスタンスにとって重要なADRの場所がすべてリストされます。

```
SELECT * FROM V$DIAG_INFO;
INST_ID NAME                 VALUE
------- -------------------- ----------------------------------------------------------
------
      1 Diag Enabled         TRUE
      1 ADR Base             /u01/oracle
      1 ADR Home             /u01/oracle/diag/rdbms/orclbi/orclbi
      1 Diag Trace           /u01/oracle/diag/rdbms/orclbi/orclbi/trace
      1 Diag Alert           /u01/oracle/diag/rdbms/orclbi/orclbi/alert
      1 Diag Incident        /u01/oracle/diag/rdbms/orclbi/orclbi/incident
      1 Diag Cdump           /u01/oracle/diag/rdbms/orclbi/orclbi/cdump
      1 Health Monitor       /u01/oracle/diag/rdbms/orclbi/orclbi/hm
      1 Default Trace File
/u01/oracle/diag/rdbms/orclbi/orclbi/trace/orcl_ora_22769.trc
      1 Active Problem Count  8
      1 Active Incident Count 20
```

次の表で、このビューに表示されるいくつかの情報を説明します。

表9-3 V$DIAG_INFOビューのデータ

名前	説明
ADR Base	ADR ベースのパス
ADR Home	現行データベース・インスタンスの ADR ホームのパス
Diag Trace	バックグラウンド・プロセスのトレース・ファイル、サーバー・プロセスのトレース・ファイル、SQL トレース・ファイル、およびテキスト形式のアラート・ログの場所
Diag Alert	XML 形式のアラート・ログの場所

名前	説明
Default Trace File	現行セッションのトレース・ファイルへのパス

V$DIAG_CRITICAL_ERRORビューを使用したクリティカル・エラーの表示

V$DIAG_CRITICAL_ERRORには、現在のOracle Databaseリリースでクリティカル・エラーとして指定されている、内部エラー以外のすべてのエラーがリストされます。内部エラーは常にクリティカル・エラーとして指定されているため、このビューに内部エラーはリストされません。

次の例は、V$DIAG_CRITICAL_ERRORビューの出力を示しています。

```
SELECT * FROM V$DIAG_CRITICAL_ERROR;
FACILITY   ERROR
---------- --------------------------------------------------------------
ORA        7445
ORA        4030
ORA        4031
ORA        29740
ORA        255
ORA        355
ORA        356
ORA        239
ORA        240
ORA        494
ORA        3137
ORA        227
ORA        353
ORA        1578
ORA        32701
ORA        32703
ORA        29770
ORA        29771
ORA        445
ORA        25319
OCI        3106
OCI        3113
OCI        3135
```

次の表で、このビューに表示される情報を説明します。

表9-4 V$DIAG_CRITICAL_ERRORビューのデータ

列	説明
FACILITY	そのエラーをレポート可能な機能(Oracle Database (ORA)、Oracle Call Interface (OCI)など)
ERROR	エラー番号

関連項目:

内部エラーの詳細は、「インシデントおよび問題について」

親トピック: Oracle Databaseの障害診断インフラストラクチャについて

9.2 問題の調査、報告および解決について

Enterprise Managerサポート・ワークベンチ(サポート・ワークベンチ)により、問題(クリティカル・エラー)を調査して報告し、場合によっては、問題を解決できます。実行する必要がある一般的なタスクのセットをまとめた「ロードマップ」を使用できます。

ノート:

この項で説明するタスクは、すべて Cloud Control ベースです。すべてのタスク(または同等のタスク)は、ADRCI コマンドライン・ユーティリティ、PL/SQL パッケージ(DBMS_HM、DBMS_SQLDIAG など)、およびその他のソフトウェア・ツールを使用して実行できます。ADRCI ユーティリティの詳細は『Oracle Database ユーティリティ』を、PL/SQL パッケージの詳細は『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

- 問題の調査、報告および解決のロードマップ
  問題の調査は、Cloud Controlの「サポート・ワークベンチ」ホームページから開始できます。ただし、標準的なワークフローはデータベースのホームページのクリティカル・エラー・アラートから始まります。
- タスク1: Cloud Controlでのクリティカル・エラー・アラートの表示
  問題(クリティカル・エラー)を調査するプロセスでは、最初にデータベース・ホームページまたはOracle Automatic Storage Managementホームページでクリティカル・エラー・アラートを検討します。
- タスク2: 問題の詳細の表示
  インシデント・マネージャ問題の詳細ページから調査を続けます。
- タスク3: (オプション)追加の診断情報の収集
  問題に関する追加の診断情報を収集するために、次のアクティビティを実行できます。この追加情報は、診断データに自動的に追加され、Oracleサポート・サービスにアップロードされます。これらのアクティビティの実行に関して不明な点がある場合は、Oracleサポートの担当者に確認してください。
- タスク4: (オプション)サービス・リクエストの作成
  この段階で、Oracleサポート・サービス・リクエストを作成し、問題の情報とともにサービス・リクエスト番号を記録できます。
- タスク5: 診断データのパッケージ化とOracleサポートへのアップロード
  このタスクでは、サポート・ワークベンチのクイック・パッケージング・プロセスを使用して、問題に関する診断情報をパッケージ化し、Oracleサポートにアップロードします。
- タスク6: サービス・リクエストの追跡と修復の実施
  診断情報をOracleサポートにアップロードした後、サービス・リクエストの追跡、追加の診断情報の収集および修復の実施のために様々なアクティビティを実行できます。

関連項目:

問題とその診断データの詳細は、「Oracle Databaseの障害診断インフラストラクチャについて」

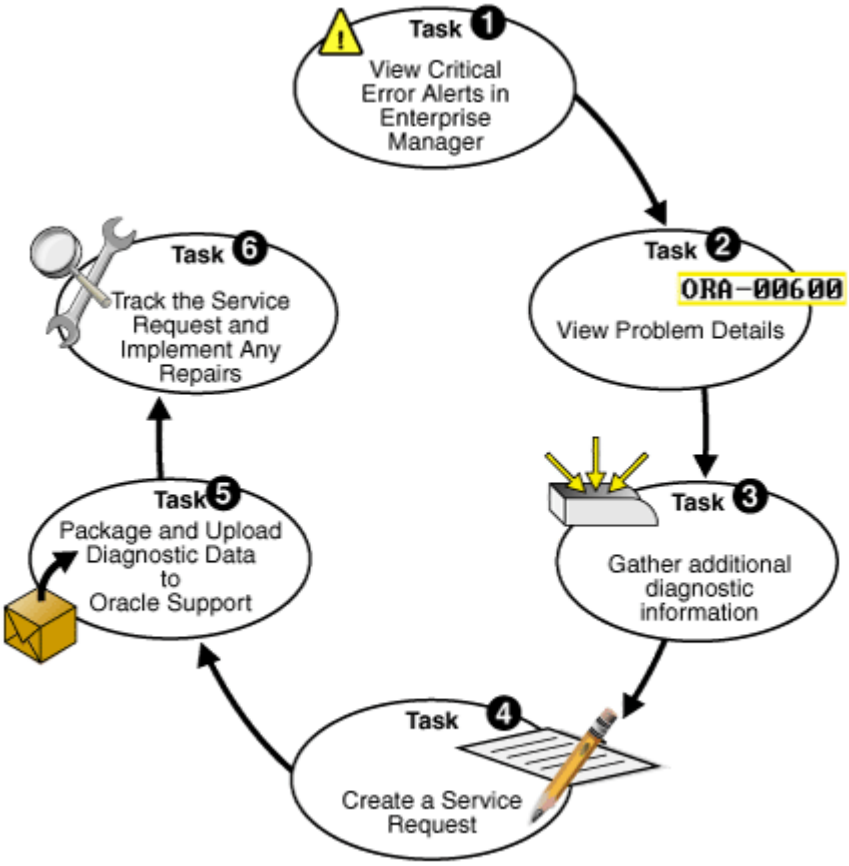
親トピック: 問題の診断と解決

9.2.1 問題の調査、報告および解決のロードマップ

問題の調査は、Cloud Controlの「サポート・ワークベンチ」ホームページから開始できます。ただし、標準的なワークフローはデータベースのホームページのクリティカル・エラー・アラートから始まります。

図9-3に、問題を調査およびレポートし、場合によってはその問題を修復するためのタスクを示します。

図9-3 問題の調査、レポートおよび解決のためのワークフロー



タスクの説明を次に示します。この後の項では、各タスクについて詳しく説明します。

- タスク1: Cloud Controlでのクリティカル・エラー・アラートの表示

  最初に、Cloud Controlのデータベース・ホームページにアクセスし、クリティカル・エラー・アラートを検討します。詳細を表示するアラートを選択し、「問題の詳細」ページに進みます。

- タスク2: 問題の詳細の表示

  問題の詳細を検証し、その問題に対して記録されたすべてのインシデントのリストを表示します。自動的に実行された状態チェックの結果を表示します。

- タスク3: (オプション)追加の診断情報の収集

  必要に応じて、追加のヘルス・チェックまたは他の診断を実行します。SQL関連エラーの場合は、必要に応じてSQLテスト・ケース・ビルダーを起動して、SQL問題に関連する必要なすべてのデータを収集し、Oracleサポート・サービスで問題を再現できる方法で情報をパッケージ化します。

- タスク4: (オプション)サービス・リクエストの作成

  必要に応じて、My Oracle Supportでサービス・リクエストを作成し、問題情報にサービス・リクエスト番号を追加して記録します。このステップをスキップした場合、サービス・リクエストは後で作成するか、サポート・ワークベンチで作成することができます。

- タスク5: 診断データのパッケージ化とOracleサポート・サービスへのアップロード

  問題のために収集された診断データを自動的にパッケージ化してOracleサポートにアップロードするガイド付きワークフロー(ウィザード)を起動します。

- タスク6: サービス・リクエストの追跡および修復の実施

  オプションとして、サービス・リクエストのアクティビティ・ログをサポート・ワークベンチでメンテナンスします。Oracleアドバイザを実行して、SQLエラーまたは破損データを修復します。

関連項目:

「サポート・ワークベンチを使用した問題の表示」

親トピック: 問題の調査、報告および解決について

9.2.2 タスク1: Cloud Controlでのクリティカル・エラー・アラートの表示

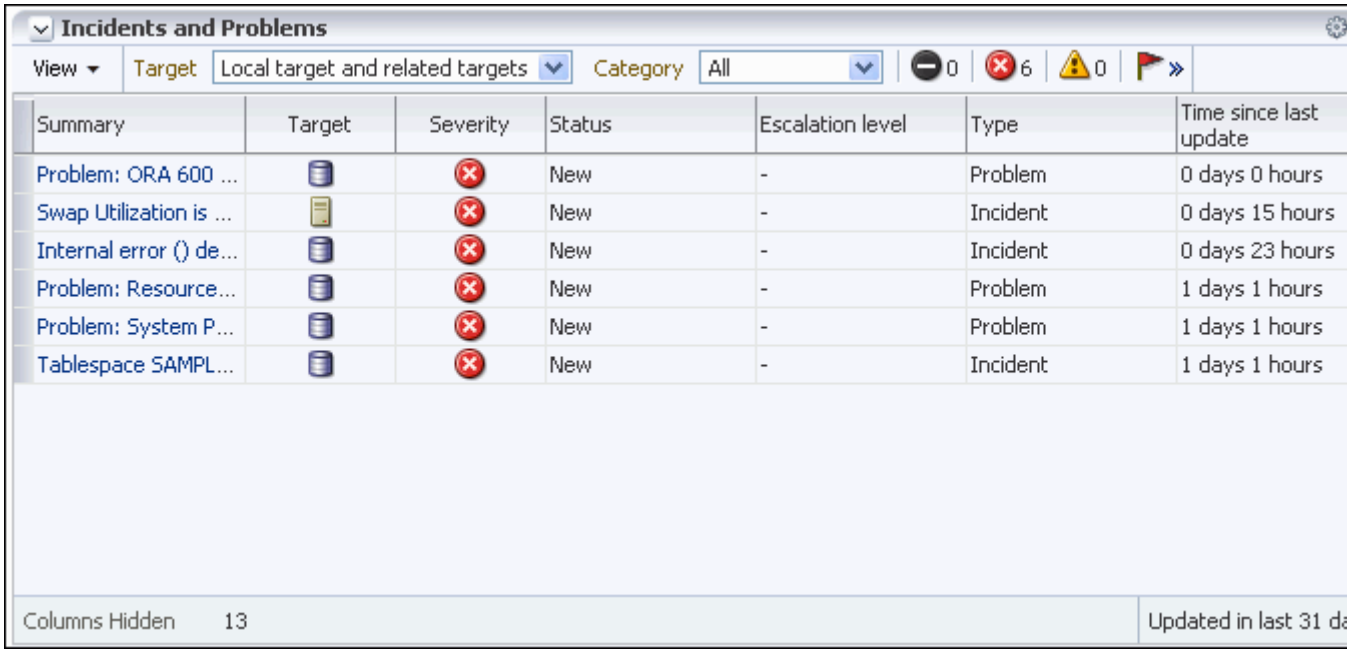
問題(クリティカル・エラー)を調査するプロセスでは、最初にデータベース・ホームページまたはOracle Automatic Storage Managementホームページでクリティカル・エラー・アラートを検討します。

クリティカル・エラー・アラートを表示するには、次のようにします。

1. Cloud Controlのデータベース・ホームページにアクセスします。
2. 「インシデントと問題」セクションでアラートを表示します。

   必要に応じて、「アラート」ヘッダーの横にある表示/非表示アイコンをクリックしてアラートを表示します。

   また、「カテゴリ」リストで特定のカテゴリを選択し、このカテゴリについてのみアラートを表示することもできます。



3. 「サマリー」列で、調査するクリティカル・エラー・アラートのメッセージをクリックします。

   「インシデント・マネージャ」の「問題の詳細」ページの「一般」サブページが表示されます。このページには次の内容が表示されます。

   - 問題の詳細
   - 「トラッキング」セクションのアラートに関するコメントを承認、クリアまたは記録できるようにする各種のコントロール
   - サポート・ワークベンチを使用して問題を診断し、「ガイドされた解決」セクションに診断情報を格納できる各種

のリンク。

調査中の問題のタイプに応じて、その他のセクションが表示される場合があります。

問題に関する詳細を表示するには、「インシデント・マネージャ」の「問題の詳細」ページで次のサブページをクリックします。

- 「インシデント」サブページには、問題の個々のインシデントに関する情報が含まれています。
- 「My Oracle Supportのナレッジ」サブページでは、My Oracle Supportにアクセスして問題に関する詳細を入手できます。
- 「更新」サブページには、その問題について入力されたすべての更新が表示されます。
- 「関連する問題」サブページには、現在の問題と同じ問題キーを持つその他の未解決問題が表示されます。

4. 次のアクションのいずれかを実行します。
   - 調査中のクリティカル・エラー・アラートに関連する問題の詳細を表示するには、「タスク2: 問題の詳細の表示」に進みます。
   - 関連する問題がいくつかあり、それについての詳細情報を表示する場合は、次のステップを一通り行います。
     - 「サポート・ワークベンチを使用した問題の表示」で説明しているように、問題とインシデントを表示します。
     - 「サポート・ワークベンチを使用した問題の表示」で説明しているように、1つの問題を選択して問題の詳細を表示します。
     - 「タスク3: (オプション)追加の診断情報の収集」を続行します。

親トピック: 問題の調査、報告および解決について

9.2.3 タスク2: 問題の詳細の表示

インシデント・マネージャ問題の詳細ページから調査を続けます。

問題の詳細を表示するには、次のようにします。

1. インシデント・マネージャ問題の詳細ページの一般サブページで、「診断」サブセクションの「サポート・ワークベンチ: 問題の詳細」をクリックします。

   「サポート・ワークベンチ」の「問題の詳細」ページが表示されます。

2. (オプション)次の処理を1つ以上完了します。
   - 「調査と解決」セクションの「診断」で、「トポロジ全体に関連する問題」をクリックします。

     ローカルのOracle Automatic Storage Management(Oracle ASM)インスタンス、またはOracle Real Application Clusters環境における他のノード上のデータベースまたはOracle ASMインスタンスにおける関連する問題を示すページが表示されます。Cloud Controlのデータベース・ホームページ上の「関連アラート」セクションにクリティカル・アラートが表示されている場合は、このステップをお薦めします。

     詳細は、「トポロジ全体に関連する問題」を参照してください。
   - 「インシデント」サブページにインシデントの詳細を表示するには、インシデントを選択して「表示」をクリックします。

     インシデントの詳細ページにダンプ・ファイル・サブページが表示されます。

- 「インシデントの詳細」ページで、「チェッカ結果」を選択して「チェッカ結果」サブページを表示します。

  このページには、クリティカル・エラー検出時に自動的に実行された状態チェックの結果が表示されます。

親トピック: 問題の調査、報告および解決について

9.2.4 タスク3: (オプション)追加の診断情報の収集

次のアクティビティを実行して、問題に関する追加の診断情報を収集します。この追加情報は、診断データに自動的に追加され、Oracleサポート・サービスにアップロードされます。これらのアクティビティの実行に関して不明な点がある場合は、Oracleサポートの担当者に確認してください。

- 追加のヘルス・チェックを手動で起動します。

  状態モニターによる事前対策的な問題識別を参照してください。

- SQLテスト・ケース・ビルダーを起動します。

  SQLテスト・ケース・ビルダーを使用したテスト・ケースの作成を参照してください。

親トピック: 問題の調査、報告および解決について

9.2.5 タスク4: (オプション)サービス・リクエストの作成

ここでは、Oracleサポート・サービスへのサービス・リクエストを作成し、サービス・リクエスト番号を問題情報とともに記録できます。

このタスクをスキップした場合は、「タスク5: 診断データのパッケージ化とOracleサポートへのアップロード」で、サポート・ワークベンチによりドラフトのサービス・リクエストが自動的に作成されます。

サービス・リクエストを作成するには、次のようにします。

1. 「エンタープライズ」メニューから、「My Oracle Support」を選択してから、「サービス・リクエスト」を選択します。

   My Oracle Supportのログインおよび登録のページが表示されます。

2. My Oracle Supportにログインし、通常の方法でサービス・リクエストを作成します。

   (オプション)次のステップのためにサービス・リクエスト番号(SR#)を覚えておきます。

3. (オプション)問題の詳細ページに戻り、次の手順を実行します。

   a. 「サマリー」セクションで、「SR#」ラベルの横にある「編集」ボタンをクリックします。

   b. SR#を入力してから、「OK」をクリックします。

   SR#が問題の詳細ページに記録されます。この情報は参照専用です。「問題の詳細」ページに戻る場合の詳細は、「サポート・ワークベンチを使用した問題の表示」を参照してください。

親トピック: 問題の調査、報告および解決について

9.2.6 タスク5: 診断データのパッケージ化とOracleサポート・サービスへのアップロード

このタスクでは、サポート・ワークベンチのクイック・パッケージング・プロセスを使用して、問題に関する診断情報をパッケージ化し、Oracleサポート・サービスにアップロードします。

クイック・パッケージングは、最小限のステップをガイド付きワークフロー(ウィザード)にまとめたものです。ウィザードを使用して、1つの問題に対してインシデント・パッケージ(パッケージ)を作成し、そのパッケージからZIPファイルを作成してアップロードします。ただし、クイック・パッケージングでは、アップロードする診断情報の編集やその他のカスタマイズは行えません。ただし、クイック・パッ

ケージングを使用すると、直接的かつ簡単な方法で診断データをパッケージ化してアップロードできます。

診断情報に含まれる機密データの編集や削除、追加のユーザー・ファイル(アプリケーション構成ファイルやスクリプトなど)の同封、その他のカスタマイズをアップロード前に実行するには、(手動のプロセスとステップが多い)カスタム・パッケージング・プロセスを使用する必要があります。手順は、問題の報告を参照してください。このタスク5の手順のかわりにその手順を行う場合は、その手順を行い、完了してから「タスク6: サービス・リクエストの追跡と修復の実装」に進んでください。

ノート:

サポート・ワークベンチは Oracle Configuration Manager を使用して診断データをアップロードします。Oracle Configuration Manager がインストールされていなかったり、適切に構成されていないと、アップロードに失敗する場合があります。この場合、メッセージが表示され、Oracle サポートへファイルを手動でアップロードするように要求されます。アップロードは、My Oracle Support を使用して手動で実行できます。

Oracle Configuration Manager の詳細は、『Oracle Configuration Manager インストレーションおよび管理ガイド』を参照してください。

診断データをパッケージ化してOracleサポートにアップロードするには、次のようにします。

1. サポート・ワークベンチ問題の詳細ページの調査と解決」セクションで、「クイック・パッケージ」をクリックします。

   「クイック・パッケージ」ウィザードの「新規パッケージの作成」ページが表示されます。

   ノート:

   まだ「問題の詳細」ページに移動していない場合は、このページに戻るための手順について、「サポート・ワークベンチを使用した問題の表示」を参照してください。

2. (オプション)パッケージ名と説明を入力します。
3. ページの残りのフィールドをすべて入力します。すでにこの問題に関するサービス・リクエストを作成している場合は、「新規サービス・リクエスト(SR)の作成」に対して「いいえ」オプション・ボタンを選択します。

   「新規サービス・リクエスト(SR)の作成」に対して「はい」オプション・ボタンを選択すると、「クイック・パッケージング」ウィザードにより、自動的にドラフトのサービス・リクエストが作成されます。後でMy Oracle Supportにログインして、このサービス・リクエストの詳細を入力する必要があります。

   「次へ」をクリックします。

   クイック・パッケージング・ウィザードによって表示されるページには、新しいパッケージを作成するコマンドが処理中であることが示されます。終了すると、クイック・パッケージング: コンテンツの表示ページが表示されます。

4. 「コンテンツの表示」ページの内容を確認し、作成されたパッケージのサイズをノートにとってから「次へ」をクリックします。

   クイック・パッケージング: マニフェストの作成ページが表示されます。

5. このページの情報を確認し、(「パス」ヘッダーの横にリストされる)マニフェストの場所をノートにとります。情報を確認したら「次へ」をクリックします。

   クイック・パッケージング: スケジュール・ページが表示されます。

6. 「即時」または「後で」のいずれかを選択します。「後で」を選択する場合は、パッケージをMy Oracle Supportに発行する時間に関する追加情報を指定します。選択を行い、必要な情報を指定した後で、「発行」をクリックします。

   「処理中: パッケージのパッケージング中およびOracleへの送信中」という進捗ページが表示されます。

「クイック・パッケージング」ウィザードが完了したとき、新しいドラフトのサービス・リクエストが作成されている場合、Cloud ControlでMy Oracle Supportに表示される確認メッセージには、ドラフトのサービス・リクエストへのリンクが含まれています。リンクをクリックして、サービス・リクエストを確認および編集できます。

「クイック・パッケージング」ウィザードによって作成されたパッケージは、サポート・ワークベンチで引き続き使用できます。カスタム・パッケージング操作を使用してパッケージを変更し(新規インシデントの追加など)、後でパッケージを再度アップロードできます。「インシデント・パッケージの表示と変更」を参照してください。

親トピック: 問題の調査、報告および解決について

9.2.7 タスク6: サービス・リクエストの追跡および修復の実施

診断情報をOracleサポート・サービスにアップロードした後は、様々なアクティビティを実行してサービス・リクエストを追跡し、追加の診断情報を収集して、修復を実装できます。

このアクティビティには次のものがあります。

- 問題情報にOracleバグ番号を追加します。

  そのためには、問題の詳細ページで「バグ#」ラベルの横にある「編集」ボタンをクリックします。この情報は参照専用です。

- 問題のアクティビティ・ログにコメントを追加します。

  このアクティビティは、組織内の他のDBAと問題ステータスや履歴情報を共有する場合に実行します。たとえば、Oracleサポート・サービスとの通信結果を記録できます。コメントを追加するステップは、次のとおりです。

  1. 「サポート・ワークベンチを使用した問題の表示」で説明しているように、この問題の「問題の詳細」ページにアクセスします。
  2. 「アクティビティ・ログ」をクリックして、アクティビティ・ログ・サブページを表示します。
  3. 「コメント」フィールドにコメントを入力し、「コメントの追加」をクリックします。

     コメントがアクティビティ・ログに記録されます。

- 新規に発生したインシデントをパッケージ化して再アップロードします。

  このアクティビティについては、問題の報告で説明されているカスタム・パッケージング方法を使用する必要があります。

- ヘルス・チェックを実行します。

  状態モニターによる事前対策的な問題識別を参照してください。

- 推奨されるOracleアドバイザを実行して修復を実施します。

  推奨されたアドバイザには、次のいずれかの方法でアクセスします。

  1. 問題の詳細ページ—「調査と解決」セクションの「セルフ・サービス」タブ
  2. 「サポート・ワークベンチ」ホームページ—「チェッカ結果」サブページ
  3. 「インシデントの詳細」ページ—「チェッカ結果」サブページ

  表9-5に、クリティカル・エラーの修復に役立つアドバイザを示します。

表9-5 クリティカル・エラーの修復に役立つOracleアドバイザ

アドバイザ	対象となるクリティカル・エラー	参照
データ・リカバリ・アドバイザ	破損ブロック、破損または欠落しているファイル、その他のデータ障害	「データ・リカバリ・アドバイザを使用したデータ破損の修復」
SQL 修復アドバイザ	SQL 文のエラー	「SQL 修復アドバイザを使用したSQL エラーの修復」

関連項目:

「インシデントの詳細」ページの「チェッカ結果」サブページを表示する手順は、「サポート・ワークベンチを使用した問題の表示」を参照してください

親トピック: 問題の調査、報告および解決について

9.3 問題の診断

この項では、Oracleデータベースでの問題を診断するための様々な方法について説明します。

- 反応的な問題識別
  この項では、Oracleデータベースの問題を反応的に識別する方法について説明します。
- 状態モニターによる事前対策的な問題識別
  状態モニターでデータベースに対して診断チェックを実行できます。
- その他の診断データの収集
  この項では、アラート・ログおよびトレース・ファイルを使用して追加で診断データを収集する方法について説明します。
- SQLテスト・ケース・ビルダーを使用したテスト・ケースの作成
  SQLテスト・ケース・ビルダーは、異なるデータベース・インスタンスでの問題の再現に必要な情報を自動的に収集するツールです。

親トピック: 問題の診断と解決

9.3.1 反応的な問題識別

この項では、Oracleデータベースの問題を反応的に識別する方法について説明します。

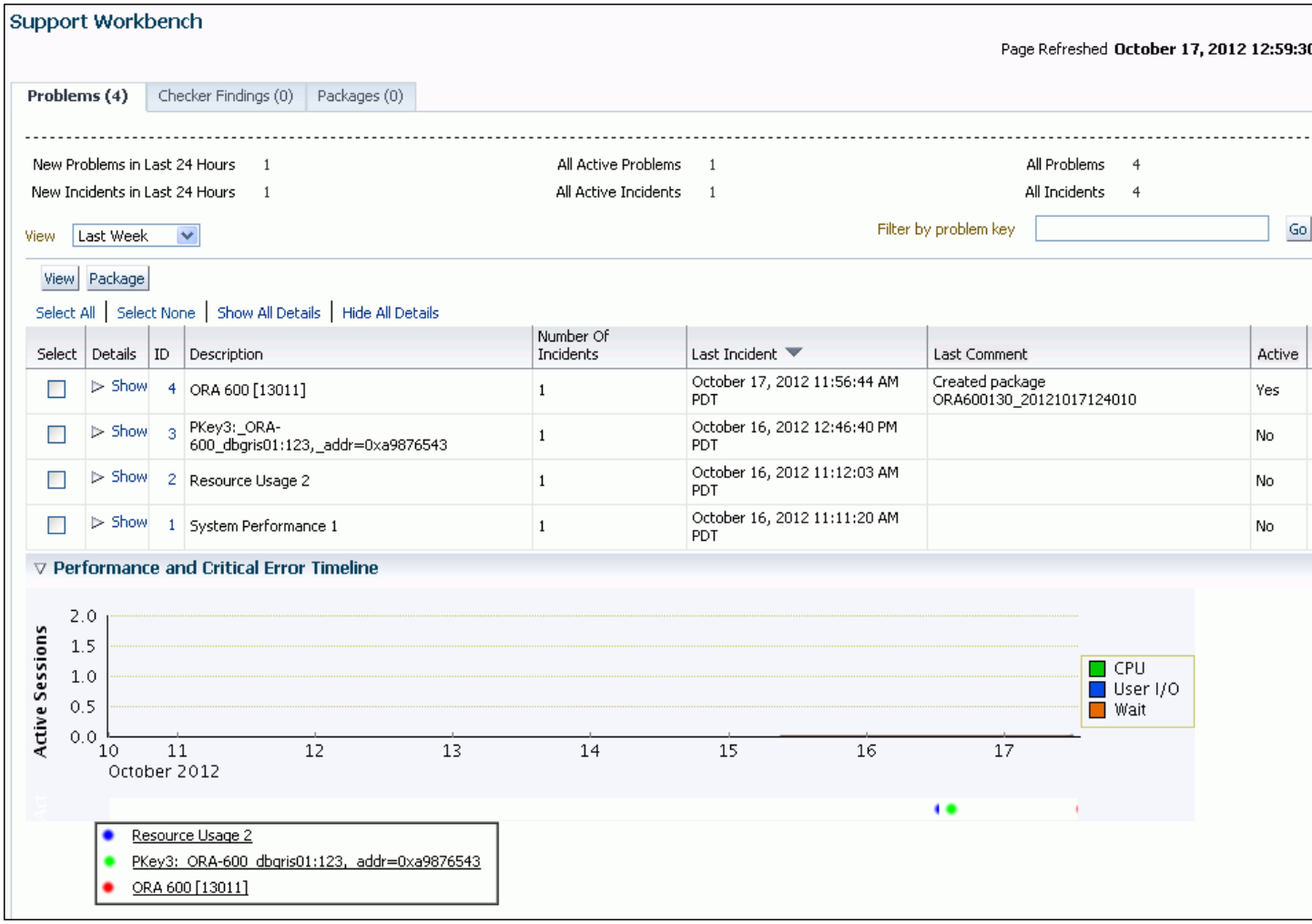
- サポート・ワークベンチを使用した問題の表示
  Cloud Controlでサポート・ワークベンチのホームページを使用して、すべての問題、または特定の期間内の問題を表示できます。
- 自動診断リポジトリへの問題の手動追加
  Cloud Controlのサポート・ワークベンチを使用して、手動でADRに問題を追加できます。
- インシデントの手動作成
  自動診断リポジトリ・コマンド・インタプリタ(ADRCI)ユーティリティを使用して、手動でインシデントを作成できます。

親トピック: 問題の診断

9.3.1.1 サポート・ワークベンチを使用した問題の表示

Cloud Controlでサポート・ワークベンチのホームページを使用して、すべての問題、または特定の期間内の問題を表示できます。

図9-4 Cloud Controlのサポート・ワークベンチのホームページ



「サポート・ワークベンチ」ホームページ(データベースまたはOracle ASM)にアクセスするには:

1. Cloud Controlのデータベース・ホームページにアクセスします。
2. 「Oracle Database」メニューから、「診断」を選択し、次に、「サポート・ワークベンチ」を選択します。

   データベース・インスタンスの「サポート・ワークベンチ」ホームページに「問題」サブページが表示されます。デフォルトでは、過去24時間の問題が表示されます。
3. Oracle ASMインスタンスの「サポート・ワークベンチ」ホームページを表示するには、「関連リンク」セクションの「サポート・ワークベンチ(+ASM_hostname)」リンクをクリックします。

問題とインシデントを表示するには:

1. 「サポート・ワークベンチ」ホームページの「表示」リストから希望する期間を選択します。すべての問題を表示するには、「すべて」を選択します。
2. (オプション)「パフォーマンスとクリティカル・エラーのタイムライン」セクションが非表示の場合、セクション・ヘッダーの横にある「表示/非表示」アイコンをクリックして、セクションを表示します。

   このセクションでは、パフォーマンスの変化とインシデントの発生の関連を表示できます。

3. (オプション)「詳細」列の下の「表示」をクリックして、特定の問題に関するすべてのインシデントのリストを表示します。次に、インシデントIDをクリックして、インシデントの詳細ページを表示します。

特定の問題の詳細を表示するには:

1. サポート・ワークベンチのホームページで、問題を選択して「表示」をクリックします。

   問題の詳細ページにインシデント・サブページが表示されます。「インシデント」サブページには、オープンしてダンプが生成された(つまり、フラッド制御されていない)すべてのインシデントが表示されます。

2. (オプション)通常のインシデントとフラッド制御されたインシデントの両方を表示するには、「ダンプされたデータ」リストで「すべて」を選択します。
3. (オプション)インシデントの詳細を表示するには、インシデントを選択して「表示」をクリックします。

   インシデントの詳細ページが表示されます。

4. (オプション)「インシデントの詳細」ページで、インシデントのチェッカ結果を表示するには、「チェッカ結果」をクリックします。
5. (オプション)インシデントの詳細ページに、そのインシデントに対して実行可能なユーザー・アクションを表示するには、「追加の診断」をクリックします。各ユーザー・アクションにより、インシデントまたはその問題に関する追加の診断情報を収集できます。

関連項目:

「インシデントのフラッド制御」

親トピック: 反応的な問題識別

9.3.1.2 自動診断リポジトリへの問題の手動追加

Cloud Controlのサポート・ワークベンチを使用して、手動でADRに問題を追加できます。

システムで生成された問題(内部でデータベースに対して生成されたクリティカル・エラーなど)は、自動的に自動診断リポジトリ(ADR)に追加され、サポート・ワークベンチ内で追跡されます。

サポート・ワークベンチから、これらの問題に関する診断データの追加収集、Oracleサポートへの診断データのアップロード、および場合によっては、問題の解決もでき、このすべてを、「問題の調査、報告および解決について」で説明されている使いやすいワークフローを使用して実行できます。

気付いた問題を、同じワークフローで処理できるように、手動でADRに追加する必要がある場合もあります。このような問題の例には、自動データベース診断モニター(ADDM)で診断されなかったグローバル・データベースのパフォーマンスに関する問題があります。サポート・ワークベンチは、このようなユーザー報告の問題を作成して処理するメカニズムを備えています。

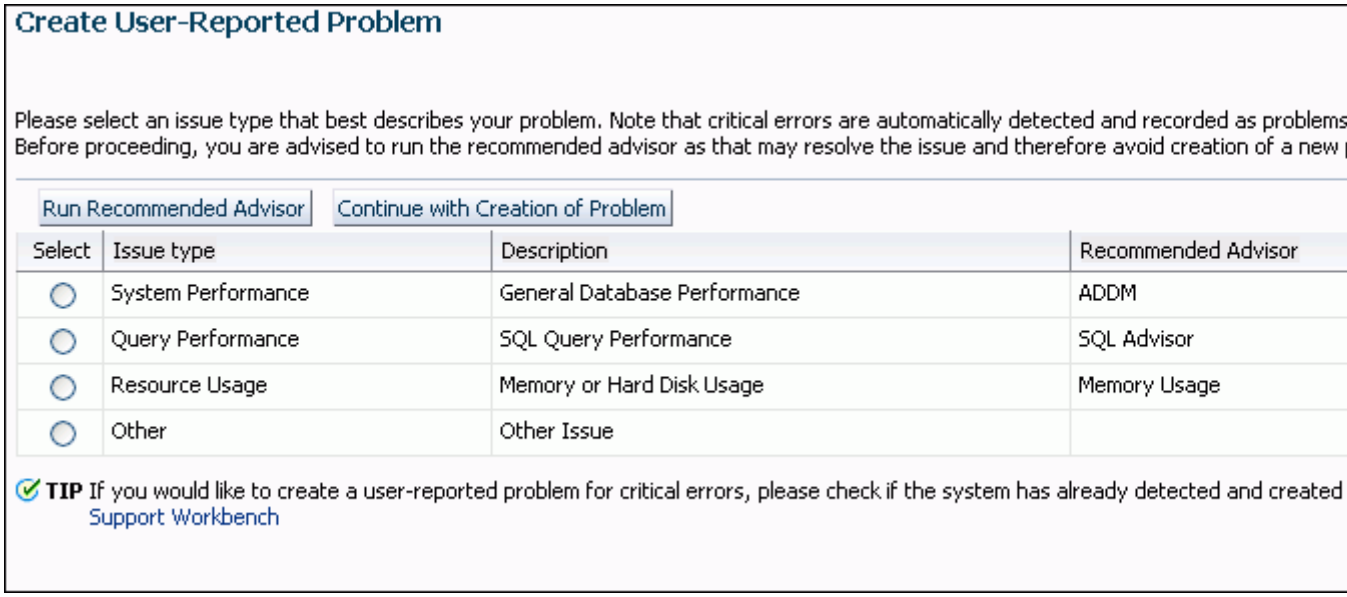
ユーザー報告の問題を作成するには:

1. 「サポート・ワークベンチ」ホームページにアクセスします。

   手順については、「サポート・ワークベンチを使用した問題の表示」を参照してください。

2. 「関連リンク」で「ユーザー報告の問題の作成」をクリックします。

   「ユーザー報告の問題の作成」ページが表示されます。



3. リストされている問題のタイプのいずれかと問題が一致する場合は、その問題のタイプを選択して「推奨アドバイザの実行」をクリックし、Oracleアドバイザを実行して問題の解決を試みます。
4. 推奨アドバイザで問題が解決しない場合、またはアドバイザを実行しなかった場合は、次のいずれかを実行します。
   - リストされている問題のタイプのいずれかと問題が一致する場合は、その問題のタイプを選択して「問題の作成の続行」をクリックします。
   - リストされている問題のいずれかのタイプと問題が一致しない場合は、問題のタイプ「その他」を選択して、「問題の作成の続行」をクリックします。

   問題の詳細ページが表示されます。
5. 「問題の詳細」ページの手順に従います。

   詳細は、問題の調査、報告および解決についてを参照してください。

関連項目:

問題とADRの詳細は、「Oracle Databaseの障害診断インフラストラクチャについて」

親トピック: 反応的な問題識別

9.3.1.3 インシデントの手動作成

自動診断リポジトリ・コマンド・インタプリタ(ADRCI)ユーティリティを使用して、手動でインシデントを作成できます。

ADRCIユーティリティを使用してインシデントを手動で作成するには:

1. ORACLE_HOMEおよびPATH環境変数が適切に設定されていることを確認します。PATH環境変数には、ORACLE_HOME/binディレクトリが含まれている必要があります。
2. オペレーティング・システムのコマンド・プロンプトで次のコマンドを実行して、ADRCIユーティリティを起動します。

   ```
   ADRCI
   ```

   ADRCIユーティリティが起動し、次のプロンプトが表示されます。

   ```
   adrci>
   ```

3. 次の構文を使用してADRCIコマンドを実行し、インシデントを手動で作成します。

```
adrci> dde create incident type incident_type
```

incident_type値で、作成するインシデントのタイプを指定します。

関連項目:

- ADRCIユーティリティの詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: 反応的な問題識別

9.3.2 状態モニターによる事前対策的な問題識別

状態モニターにより、データベースの診断チェックを実行できます。

- 状態モニターについて
  Oracle Databaseは、データベースに対して診断チェックを実行する、状態モニターと呼ばれるフレームワークを備えています。
- ヘルス・チェックの手動実行
  状態モニターでは、DBMS_HM PL/SQLパッケージを使用するか、「アドバイザ・セントラル」ページの「チェッカ」サブページにあるCloud Controlインタフェースを使用して、手動でヘルス・チェックを実行できます。
- チェッカ・レポートの表示
  チェッカを実行した後、その実行のレポートを表示できます。レポートには、結果、推奨事項およびその他の情報が含まれます。このレポートは、Cloud Control、ADRCIユーティリティまたはDBMS_HM PL/SQLパッケージを使用して表示できます。次の表に、各表示方法で使用可能なレポート形式を示します。
- 状態モニターのビュー
  チェッカ・レポートを要求するかわりに、レポートが作成されたADRデータを直接問い合せると、特定のチェッカの実行結果を表示できます。
- ヘルス・チェック・パラメータの参考情報
  いくつかのヘルス・チェックでは、パラメータが必要です。デフォルト値が(なし)のパラメータは必須です。

親トピック: 問題の診断

9.3.2.1 状態モニターについて

Oracle Databaseは、データベースに対して診断チェックを実行する、状態モニターと呼ばれるフレームワークを備えています。

- 状態モニターについて
  状態モニター・チェック(チェッカ、ヘルス・チェックまたはチェックとも呼ばれる)では、データベースの様々なレイヤーとコンポーネントが検証されます。
- ヘルス・チェックのタイプ
  状態モニターでは、いくつかの異なるタイプのチェックが実行されます。

親トピック: 状態モニターによる事前対策的な問題識別

9.3.2.1.1 状態モニター・チェックについて

状態モニター・チェック(チェッカ、ヘルス・チェックまたはチェックとも呼ばれます)では、データベースの様々なレイヤーとコンポーネントが検証されます。

ヘルス・チェックでは、ファイルの破損、物理ブロックおよび論理ブロック破損、UNDOおよびREDOの破損、データ・ディクショナリの破損などが検出されます。また、ヘルス・チェックの結果のレポートが生成され、多くの場合、問題を解決するための推奨事項が示されます。ヘルス・チェックは次の2つの方法で実行できます。

- リアクティブ—障害診断インフラストラクチャでは、クリティカル・エラーの内容に応じてヘルス・チェックを自動的に実行できます。
- 手動—DBAは、DBMS_HM PL/SQLパッケージまたはCloud Controlインタフェースのいずれかを使用して、ヘルス・チェックを手動で実行できます。必要に応じて定期的にチェッカを実行できますが、Oracleサポートがサービス・リクエストについて共同で作業しているときにチェッカの実行をお願いさせていただくこともあります。

状態モニター・チェックでは、結果、推奨事項およびその他の情報が自動診断リポジトリ(ADR)に格納されます。

ヘルス・チェックは次の2つのモードで実行できます。

- DBオンライン・モードは、データベースがオープン状態(つまり、OPENモードまたはMOUNTモード)のときは、チェックを実行できることを意味します。
- DBオフライン・モードは、インスタンスは使用可能だがデータベース自体がクローズ(つまり、NOMOUNTモード)している場合にチェックを実行できることを意味します。

DBオンライン・モードでは、すべてのヘルス・チェックを実行できます。DBオフライン・モードで使用できるのは、REDOの整合性チェックとDB構造の整合性チェックのみです。

ノート:

「自動診断リポジトリ(ADR)」

親トピック: 状態モニターについて

9.3.2.1.2 ヘルス・チェックのタイプ

状態モニターでは、いくつかの異なるタイプのチェックが実行されます。

状態モニターでは次のチェックが実行されます。

- DB構造の整合性チェック—このチェックではデータベース・ファイルの整合性が検証され、これらのファイルがアクセス不可、破損状態または不整合の場合はエラーが報告されます。データベースがマウント・モードまたはオープン・モードの場合は、制御ファイルに列記されたログ・ファイルとデータファイルがチェックされます。データベースがNOMOUNTモードの場合は、制御ファイルのみがチェックされます。
- データ・ブロックの整合性チェック—このチェックではチェックサム・エラー、先頭/末尾の不一致、ブロック内の論理的不整合などのディスク・イメージ・ブロック破損が検出されます。ほとんどの破損はブロック・メディア・リカバリを使用して修復できます。破損ブロック情報はV$DATABASE_BLOCK_CORRUPTIONビューでも取得されます。このチェックではブロック間またはセグメント間の破損は検出されません。
- REDOの整合性チェック—このチェックではアクセス可能性と破損についてREDOログの内容がスキャンされ、可能な場合はアーカイブ・ログもスキャンされます。REDOの整合性チェックではアーカイブ・ログやREDOの破損などのエラーが報告されます。
- UNDOセグメントの整合性チェック—このチェックでは、論理的なUNDO破損が検出されます。UNDO破損の場所を特定した後、このチェックではPMONおよびSMONを使用して、破損したトランザクションのリカバリを試みます。このリカ

バリに失敗した場合、状態モニターは破損の情報をV$CORRUPT_XID_LISTに格納します。ほとんどのUNDO破損は、コミットを強制実行することで解決できます。

- トランザクションの整合性チェック—このチェックは、1つの特定トランザクションのみがチェックされることを除いて、UNDOセグメントの整合性チェックと同じです。
- ディクショナリの整合性チェック—このチェックでは、tab$、col$などのコア・ディクショナリ・オブジェクトの整合性が検証されます。次の操作を実行します。
  - 各ディクショナリ・オブジェクトに対するディクショナリ・エントリの内容が確認されます。
  - 行間レベルのチェックが実行され、ディクショナリの行で論理的制約が適用されていることが確認されます。
  - オブジェクト関係のチェックが実行され、ディクショナリ・オブジェクト間で親子関係が規定されていることが確認されます。

  ディクショナリの整合性チェックは、次のディクショナリ・オブジェクトに対して実行されます。

  tab$、clu$、fet$、uet$、seg$、undo$、ts$、file$、obj$、ind$、icol$、col$、user$、con$、cdef$、ccol$、bootstrap$、objauth$、ugroup$、tsq$、syn$、view$、typed_view$、superobj$、seq$、lob$、coltype$、subcoltype$、ntab$、refcon$、opqtype$、dependency$、access$、viewcon$、icoldep$、dual$、sysauth$、objpriv$、defrole$およびecol$

親トピック: 状態モニターについて

9.3.2.2 ヘルス・チェックの手動実行

状態モニターでは、DBMS_HM PL/SQLパッケージを使用するか、「アドバイザ・セントラル」ページの「チェッカ」サブページにあるCloud Controlインタフェースを使用して、手動でヘルス・チェックを実行できます。

- DBMS_HM PL/SQLパッケージを使用したヘルス・チェックの実行
  ヘルス・チェックを実行するためのDBMS_HMプロシージャは、RUN_CHECKと呼ばれています。
- Cloud Controlを使用したヘルス・チェックの実行
  Cloud Controlは、状態モニター・チェッカを実行するためのインタフェースを提供します。

親トピック: 状態モニターによる事前対策的な問題識別

9.3.2.2.1 DBMS_HM PL/SQLパッケージを使用したヘルス・チェックの実行

ヘルス・チェックを実行するためのDBMS_HMプロシージャはRUN_CHECKです。

1. RUN_CHECKをコールするには、次のようにチェック名と実行名を指定します。

```
BEGIN
DBMS_HM.RUN_CHECK('Dictionary Integrity Check', 'my_run');
END;
/
```

2. ヘルス・チェック名のリストを取得するには、次の問合せを実行します。

```
SELECT name FROM v$hm_check WHERE internal_check='N';
```

出力は次のようになります。

```
NAME
----------------------------------------------------------------
DB Structure Integrity Check
Data Block Integrity Check
```

```
Redo Integrity Check
Transaction Integrity Check
Undo Segment Integrity Check
Dictionary Integrity Check
```

ほとんどのヘルス・チェックに入力パラメータを指定できます。パラメータ名と説明はV$HM_CHECK_PARAMビューを使用して表示できます。一部のパラメータは必須ですが、オプションのパラメータもあります。オプションのパラメータを省略すると、デフォルトが使用されます。次の問合せでは、すべてのヘルス・チェックのパラメータ情報が表示されます。

```
SELECT c.name check_name, p.name parameter_name, p.type,
p.default_value, p.description
FROM v$hm_check_param p, v$hm_check c
WHERE p.check_id = c.id and c.internal_check = 'N'
ORDER BY c.name;
```

入力パラメータは、セミコロン(;)で区切られた名前/値のペアとしてinput_params引数で渡されます。次の例では、トランザクションIDをパラメータとして「トランザクションの整合性チェック」に渡す方法を示します。

```
BEGIN
  DBMS_HM.RUN_CHECK (
   check_name   => 'Transaction Integrity Check',
   run_name     => 'my_run',
   input_params => 'TXN_ID=7.33.2');
END;
/
```

関連項目:

- 「ヘルス・チェック・パラメータの参考情報」
- DBMS_HMの使用例の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ヘルス・チェックの手動実行

9.3.2.2.2 Cloud Controlを使用したヘルス・チェックの実行

Cloud Controlは、状態モニター・チェッカを実行するためのインタフェースを備えています。

Cloud Controlを使用して状態モニター・チェッカを実行するには:

1. 「データベース・ホーム」ページにアクセスします。
2. 「パフォーマンス」メニューから、「アドバイザ・ホーム」を選択します。
3. 「チェッカ」をクリックして「チェッカ」サブページを表示します。
4. 「チェッカ」セクションで、実行するチェッカをクリックします。
5. 入力パラメータの値を入力するか、オプション・パラメータの場合は空白のままにしてデフォルトを使用します。
6. 「OK」をクリックし、パラメータを確認して「OK」を再度クリックします。

親トピック: ヘルス・チェックの手動実行

9.3.2.3 チェッカ・レポートの表示

チェッカを実行した後は、その実行内容のレポートを表示できます。レポートには、結果、推奨事項およびその他の情報が含まれます。このレポートは、Cloud Control、ADRCIユーティリティまたはDBMS_HM PL/SQLパッケージを使用して表示できます。次の表に、各表示方法で使用可能なレポート形式を示します。

- チェッカ・レポートの表示について
  チェッカの実行結果(結果、推奨事項およびその他の情報)はADRに格納されますが、レポートはすぐに生成されません。
- Cloud Controlを使用したレポートの表示
  Cloud Controlを使用して、特定のチェッカの実行に関する状態モニター・レポートと結果を表示することもできます。
- DBMS_HMを使用したレポートの表示
  状態モニター・チェッカ・レポートは、DBMS_HMパッケージ・ファンクションGET_RUN_REPORTを使用して表示できます。
- ADRCIユーティリティを使用したレポートの表示
  ADRCIユーティリティを使用して、状態モニター・チェッカ・レポートを作成および表示できます。

親トピック: 状態モニターによる事前対策的な問題識別

9.3.2.3.1 チェッカ・レポートの表示について

チェッカの実行結果(結果、推奨事項およびその他の情報)はADRに格納されますが、レポートはすぐに生成されません。

レポートの表示方法	使用可能なレポート形式
Cloud Control	HTML
DBMS_HM PL/SQL パッケージ	HTML、XML およびテキスト
ADRCI ユーティリティ	XML

DBMS_HM PL/SQLパッケージまたはCloud Controlを使用してレポートを要求したときに、レポートがまだ生成されていない場合は、最初にADR内のチェッカ実行データからレポートが生成され、現行インスタンス用のADRホームのHMサブディレクトリにXML形式でレポート・ファイルとして格納された後に、レポートが表示されます。レポートがすでに生成されている場合は、そのレポートが表示されます。ADRCIユーティリティを使用する場合、レポートがまだ生成されていないときは、最初にレポート・ファイルを生成するためのコマンドを実行し、次にその内容を表示するためのコマンドを実行する必要があります。

チェッカ・レポートは、Cloud Controlを使用して表示することをお薦めします。

ノート:

「自動診断リポジトリ(ADR)」

親トピック: チェッカ・レポートの表示

9.3.2.3.2 Cloud Controlを使用したレポートの表示

Cloud Controlを使用して、特定のチェッカ実行の状態モニター・レポートと結果を表示できます。

Cloud Controlを使用して実行結果を表示するには:

1. 「データベース・ホーム」ページにアクセスします。
2. 「パフォーマンス」メニューから、「アドバイザ・ホーム」を選択します。
3. 「チェッカ」をクリックして「チェッカ」サブページを表示します。
4. 表示するチェッカ実行の実行名をクリックします。

「実行の詳細」ページにチェッカ実行の「結果」サブページが表示されます。

5. 「実行」をクリックして「実行」サブページを表示します。

   チェッカ実行に関する詳細がCloud Controlに表示されます。

6. 「レポートの表示」をクリックして、チェッカ実行のレポートを表示します。

   レポートが新規のブラウザ・ウィンドウに表示されます。

親トピック: チェッカ・レポートの表示

9.3.2.3.3 DBMS_HMを使用したレポートの表示

状態モニター・チェッカ・レポートは、DBMS_HMパッケージ・ファンクションGET_RUN_REPORTを使用して表示できます。

このファンクションを使用すると、HTML、XMLまたはテキスト形式のレポートを要求できます。デフォルトはテキスト形式で、次にSQL*Plusの例を示します。

```
SET LONG 100000
SET LONGCHUNKSIZE 1000
SET PAGESIZE 1000
SET LINESIZE 512
SELECT DBMS_HM.GET_RUN_REPORT('HM_RUN_1061') FROM DUAL;
DBMS_HM.GET_RUN_REPORT('HM_RUN_1061')
------------------------------------------------------------------------------

 Run Name                      : HM_RUN_1061
 Run Id                        : 1061
 Check Name                    : Data Block Integrity Check
 Mode                          : REACTIVE
 Status                        : COMPLETED
 Start Time                    : 2007-05-12 22:11:02.032292 -07:00
 End Time                      : 2007-05-12 22:11:20.835135 -07:00
 Error Encountered             : 0
 Source Incident Id            : 7418
 Number of Incidents Created   : 0

Input Parameters for the Run
 BLC_DF_NUM=1
 BLC_BL_NUM=64349

Run Findings And Recommendations
 Finding
 Finding Name  : Media Block Corruption
 Finding ID    : 1065
 Type          : FAILURE
 Status        : OPEN
 Priority      : HIGH
 Message       : Block 64349 in datafile 1:
                 '/u01/app/oracle/dbs/t_db1.f' is media corrupt
 Message       : Object BMRTEST1 owned by SYS might be unavailable
 Finding
 Finding Name  : Media Block Corruption
 Finding ID    : 1071
 Type          : FAILURE
 Status        : OPEN
 Priority      : HIGH
 Message       : Block 64351 in datafile 1:
                 '/u01/app/oracle/dbs/t_db1.f' is media corrupt
 Message       : Object BMRTEST2 owned by SYS might be unavailable
```

関連項目:

DBMS_HMパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: チェッカ・レポートの表示

9.3.2.3.4 ADRCIユーティリティを使用したレポートの表示

ADRCIユーティリティを使用して、状態モニター・チェッカ・レポートを作成および表示できます。

ADRCIを使用してチェッカ・レポートを作成および表示するには:

1. ORACLE_HOMEおよびPATH環境変数が正しく設定されていることを確認し、オペレーティング・システムのコマンド・プロンプトで次のコマンドを実行してADRCIユーティリティを起動します。

   ```
   ADRCI
   ```

   ADRCIユーティリティが起動し、次のプロンプトが表示されます。

   ```
   adrci>
   ```

   必要に応じて、現行のADRホームを変更できます。すべてのADRホームをリストするにはSHOW HOMESコマンドを、現行のADRホームを変更するにはSET HOMEPATHコマンドを使用します。詳細は、『Oracle Databaseユーティリティ』を参照してください。

2. 次のコマンドを入力します。

   ```
   show hm_run
   ```

   このコマンドによって、ADRに登録されている(V$HM_RUNに格納された)すべてのチェッカ実行がリストされます。

3. レポートを作成するチェッカ実行を検索し、チェッカ実行名をノートにとります。このチェッカ実行のレポートがすでに生成されている場合は、REPORT_FILEフィールドにファイル名が表示されます。レポートが生成されていない場合は、次のコマンドを実行してレポートを生成します。

   ```
   create report hm_run run_name
   ```

4. レポートを表示するには、次のコマンドを入力します。

   ```
   show report hm_run run_name
   ```

ノート:

「自動診断リポジトリ(ADR)」

親トピック: チェッカ・レポートの表示

9.3.2.4 状態モニターのビュー

チェッカ・レポートを要求するかわりに、レポートが作成されたADRデータを直接問い合せると、特定のチェッカ実行の結果を表示できます。

このデータは、V$HM_RUN、V$HM_FINDINGおよびV$HM_RECOMMENDATIONの各ビューを使用して表示できます。

次の例では、V$HM_RUNビューを問い合せて、チェッカ実行の履歴を表示します。

```
SELECT run_id, name, check_name, run_mode, src_incident FROM v$hm_run;
    RUN_ID NAME         CHECK_NAME                       RUN_MODE SRC_INCIDENT
---------- ------------ -------------------------------- -------- ------------
```

```
         1 HM_RUN_1     DB Structure Integrity Check       REACTIVE            0
       101 HM_RUN_101   Transaction Integrity Check        REACTIVE         6073
       121 TXNCHK       Transaction Integrity Check        MANUAL              0
       181 HMR_tab$     Dictionary Integrity Check         MANUAL              0
           .
           .
           .
       981 Proct_ts$    Dictionary Integrity Check         MANUAL              0
      1041 HM_RUN_1041  DB Structure Integrity Check       REACTIVE            0
      1061 HM_RUN_1061  Data Block Integrity Check         REACTIVE         7418
```

次の例では、RUN_ID 1061を指定してV$HM_FINDINGビューを問い合せ、リアクティブなデータ・ブロック・チェックの結果詳細を取得します。

```
SELECT type, description FROM v$hm_finding WHERE run_id = 1061;
TYPE          DESCRIPTION
------------- ----------------------------------------
FAILURE       Block 64349 in datafile 1: '/u01/app/orac
              le/dbs/t_db1.f' is media corrupt

FAILURE       Block 64351 in datafile 1: '/u01/app/orac
              le/dbs/t_db1.f' is media corrupt
```

関連項目:

- 「ヘルス・チェックのタイプ」
- V$HM_*ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 状態モニターによる事前対策的な問題識別

9.3.2.5 ヘルス・チェック・パラメータの参考情報

いくつかのヘルス・チェックでは、パラメータが必要です。デフォルト値が(なし)のパラメータは必須です。

表9-6 「データ・ブロックの整合性チェック」のパラメータ

パラメータ名	タイプ	デフォルト値	説明
BLC_DF_NUM	数値	(なし)	ブロック・データ・ファイル番号
BLC_BL_NUM	数値	(なし)	データ・ブロック番号

表9-7 「REDOの整合性チェック」のパラメータ

パラメータ名	タイプ	デフォルト値	説明
SCN_TEXT	テキスト	0	最新の良好な REDO の SCN(既知の場合)

表9-8 「UNDOセグメントの整合性チェック」のパラメータ

パラメータ名	タイプ	デフォルト値	説明

パラメータ名	タイプ	デフォルト値	説明
USN_NUMBER	テキスト	(なし)	UNDO セグメント番号

表9-9 「トランザクションの整合性チェック」のパラメータ

パラメータ名	タイプ	デフォルト値	説明
TXN_ID	テキスト	(なし)	トランザクション ID

表9-10 「ディクショナリの整合性チェック」のパラメータ

パラメータ名	タイプ	デフォルト値	説明
CHECK_MASK	テキスト	ALL	使用可能な値は、次のとおりです。 ● COLUMN_CHECKS—列のチェックのみを実行します。コア・テーブル内の列レベルの制約を検証します。 ● ROW_CHECKS—行のチェックのみを実行します。コア・テーブル内の行レベルの制約を検証します。 ● REFERENTIAL_CHECKS—参照チェックのみを実行します。コア・テーブル内の参照の制約を検証します。 ● ALL—すべてのチェックを実行します。
TABLE_NAME	テキスト	ALL_CORE_TABLES	チェックする単一のコア・テーブルの名前。省略すると、すべてのコア・テーブルがチェックされます。

親トピック: 状態モニターによる事前対策的な問題識別

9.3.3 その他の診断データの収集

この項では、アラート・ログおよびトレース・ファイルを使用して追加で診断データを収集する方法について説明します。

- アラート・ログの表示
  アラート・ログは、テキスト・エディタ、Cloud ControlまたはADRCIユーティリティで表示できます。
- トレース・ファイルの検索

トレース・ファイルは、自動診断リポジトリ(ADR)の各ADRホーム下にあるtraceディレクトリに格納されます。このディレクトリ内の個々のトレース・ファイルを検索するには、データ・ディクショナリ・ビューを使用します。たとえば、現行セッションのトレース・ファイルへのパス、または各Oracle Databaseプロセスのトレース・ファイルへのパスを検索できます。

親トピック: 問題の診断

9.3.3.1 アラート・ログの表示

アラート・ログは、テキスト・エディタ、Cloud ControlまたはADRCIユーティリティを使用して表示できます。

Cloud Controlでアラート・ログを表示するには:

1. Cloud Controlのデータベース・ホームページにアクセスします。
2. 「Oracle Database」メニューから、「診断」を選択し、次に、「サポート・ワークベンチ」を選択します。
3. 「関連リンク」で「アラート・ログの内容」をクリックします。

   「アラート・ログの内容の表示」ページが表示されます。
4. 表示するエントリの数を選択して「実行」をクリックします。

テキスト・エディタを使用してアラート・ログを表示するには:

1. SQL*PlusまたはSQL Developerなどの問合せツールを使用して、データベースに接続します。
2. 「V$DIAG_INFOビューを使用したADRの場所の表示」に示すように、V$DIAG_INFOビューを問い合せます。
3. XMLタグがないテキストのみのアラート・ログを表示するステップは、次のとおりです。
   a. V$DIAG_INFO問合せ結果からDiag Traceエントリに対応するパスをノートにとり、ディレクトリをそのパスに変更します。
   b. テキスト・エディタを使用して、alert_SID.logファイルをオープンします。
4. XML形式のアラート・ログを表示するステップは、次のとおりです。
   a. V$DIAG_INFO問合せ結果からDiag Alertエントリに対応するパスをノートにとり、ディレクトリをそのパスに変更します。
   b. テキスト・エディタを使用して、log.xmlファイルをオープンします。

関連項目:

ADRCIユーティリティを使用してテキスト形式(XMLタグは削除されます)のアラート・ログを表示し、そのアラート・ログに対して問合せを実行する方法については、『Oracle Databaseユーティリティ』を参照してください。

親トピック: その他の診断データの収集

9.3.3.2 トレース・ファイルの検索

トレース・ファイルは、自動診断リポジトリ(ADR)内の各ADRホーム下にあるtraceディレクトリに格納されます。このディレクトリ内の個々のトレース・ファイルを検索するには、データ・ディクショナリ・ビューを使用します。たとえば、現行セッションのトレース・ファイルへのパス、または各Oracle Databaseプロセスのトレース・ファイルへのパスを検索できます。

現行セッションのトレース・ファイルを検索するには:

- 次の問合せを発行します。

```
SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Default Trace File';
```

トレース・ファイルへのフルパスが返されます。

現行インスタンスのすべてのトレース・ファイルを検索するには:

- 次の問合せを発行します。

```
SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Diag Trace';
```

現在のインスタンスのADRトレース・ディレクトリへのパスが返されます。

各Oracle Databaseプロセスのトレース・ファイルを特定するには:

- 次の問合せを発行します。

```
SELECT PID, PROGRAM, TRACEFILE FROM V$PROCESS;
```

関連項目:

- 「自動診断リポジトリの構造、内容および場所」
- 『Oracle Databaseユーティリティ』のADRCI SHOW TRACEFILEコマンドに関する項

親トピック: その他の診断データの収集

9.3.4 SQLテスト・ケース・ビルダーを使用したテスト・ケースの作成

SQLテスト・ケース・ビルダーは、異なるデータベース・インスタンス内の問題の再現に必要な情報を自動的に収集するツールです。

SQLテスト・ケースは、パフォーマンスの問題が発生した特定のSQL文の実行計画を開発者が再現できるようにするための一連の情報です。

この項では、次の項目について説明します。

- SQLテスト・ケース・ビルダーの目的
  SQLテスト・ケース・ビルダーによって、問題に関する情報とその問題が発生した環境に関する情報を収集し再現するプロセスが自動化されます。
- SQLテスト・ケース・ビルダーの概念
  SQLテスト・ケース・ビルダーの主な概念には、SQLインシデント、記録される情報のタイプおよび出力の形式が含まれます。
- SQLテスト・ケース・ビルダーのユーザー・インタフェース
  SQLテスト・ケース・ビルダーにアクセスするには、Cloud Controlを使用するか、コマンドラインでPL/SQLを使用します。
- SQLテスト・ケース・ビルダーの実行
  Cloud Controlを使用してSQLテスト・ケース・ビルダーを実行できます。

親トピック: 問題の診断

9.3.4.1 SQLテスト・ケース・ビルダーの目的

SQLテスト・ケース・ビルダーによって、問題に関する情報とその問題が発生した環境に関する情報を収集し再現するプロセスが自動化されます。

ほとんどのSQLコンポーネントでは、再現可能なテスト・ケースを取得することが、迅速に不具合を解決するための最も重要な要因となります。ユーザーにとって最も長く手間のかかるステップでもあります。SQLテスト・ケース・ビルダーの目的は、SQLインシデントに関連する情報をできるだけ多く収集し、Oracleスタッフが別のシステムで問題を再現できるようにパッケージ化することです。

SQLテスト・ケース・ビルダーの出力は、事前に定義されたディレクトリに集められたスクリプトです。これらのスクリプトには、別のデータベース・インスタンスでの必要なすべてのオブジェクトと環境の再作成に必要なコマンドが含まれています。テスト・ケースの準備が整ったら、そのディレクトリのZIPファイルを作成して別のデータベースに移動するか、またはそのファイルをOracleサポートにアップロードできます。

親トピック: SQLテスト・ケース・ビルダーを使用したテスト・ケースの作成

9.3.4.2 SQLテスト・ケース・ビルダーの概念

SQLテスト・ケース・ビルダーの主な概念には、SQLインシデント、記録される情報のタイプおよび出力の形式が含まれます。

この項では、次の項目について説明します。

- SQLインシデント
  Oracle Databaseの障害診断インフラストラクチャでは、**インシデント**は1つの問題の1回の発生を表します。
- SQLテスト・ケース・ビルダーで取得される情報
  SQLテスト・ケース・ビルダーは、SQL問合せおよびその環境に関する永続的な情報を取得します。
- SQLテスト・ケース・ビルダーの出力
  SQLテスト・ケース・ビルダーの出力は、環境と必要なすべてのオブジェクトの再作成に必要なコマンドが格納された一連のファイルです。

親トピック: SQLテスト・ケース・ビルダーを使用したテスト・ケースの作成

9.3.4.2.1 SQLインシデント

Oracle Databaseの障害診断インフラストラクチャでは、**インシデント**は1つの問題の1回の発生を表します。

SQLインシデントはSQL関連の問題です。問題(クリティカル・エラー)が複数回発生したとき、それぞれの発生に対してインシデントが作成されます。インシデントには日時が記録され、自動診断リポジトリ(ADR)で追跡されます。各インシデントには、ADR内で一意である数値のインシデントIDがあります。

SQLテスト・ケース・ビルダーには、コマンドラインで常時アクセスできます。Oracle Enterprise Manager Cloud Control(Cloud Control)では、SQLテスト・ケース・ページは、SQLインシデントが見つかった場合にのみ利用できます。

親トピック: SQLテスト・ケース・ビルダーの概念

9.3.4.2.2 SQLテスト・ケース・ビルダーで取得される情報

SQLテスト・ケース・ビルダーは、SQL問合せおよびその環境に関する永続的な情報を取得します。

この情報には、実行中の問合せ、表と索引の定義(実際のデータではありません)、PL/SQLパッケージおよびプログラム・ユニット、オプティマイザ統計、SQL計画ベースライン、初期化パラメータの設定などがあります。Oracle Database 12c以降では、文の実行の一部としてのみ使用できる情報など、一時情報の取得や再現もSQLテスト・ケース・ビルダーで行われるようになりました。

SQLテスト・ケース・ビルダーでは、次のものがサポートされています。

- 適応計画

  SQLテスト・ケース・ビルダーは、適応計画に関して行われた決定への入力を取得し、各決定の時点でそれらを再現し

ます。適応計画の場合、最終プランの決定には、各バッファ統計コレクタの最終統計値で十分です。

- 自動メモリー管理

  データベースは、各SQL操作で要求されたメモリーを自動的に処理します。ソートなどのアクションは、パフォーマンスに重大な影響を及ぼす可能性があります。SQLテスト・ケース・ビルダーでは、データベースで割り当てられたメモリーの場所とその割当て量などのメモリー・アクティビティが記録されています。

- 動的統計

  動的統計とは、述語の選択性を見積もるために、データベースが再帰的SQL文を実行して表のブロックの小さいランダム・サンプルをスキャンする最適化手法です。異なるデータベース上の動的統計の再収集では、必ずしも同じ結果(データの消失時間など)が生成されるわけではありません。問題を再現するには、SQLテスト・ケース・ビルダーでソース・データベースから動的統計の結果をエクスポートします。テスト・データベースでは、SQLテスト・ケース・ビルダーによって、動的統計を再収集するのではなくソース・データベースから取得した同じ値を再利用します。

- 複数の実行のサポート

  SQLテスト・ケース・ビルダーでは、問合せの複数の実行の間に累積された動的情報を取得できます。この機能は自動再最適化において重要です。

- コンパイル環境およびバインド値の再現

  コンパイル環境設定は、問合せ最適化コンテキストの重要な一部です。ソース・データベースで問題の問合せが実行された際には、ユーザーが変更したデフォルトではない設定がSQLテスト・ケース・ビルダーで取得されます。デフォルトではないパラメータ値が使用されている場合、SQLテスト・ケース・ビルダーは、問合せを実行する前に、その同じ値を再構築します。

- オブジェクトの統計履歴

  オブジェクトの統計履歴は、統計値の変更によって計画に変更が生じたかどうかを確認する場合に役に立ちます。DBMS_STATSは、履歴をデータ・ディクショナリ内に格納します。SQLテスト・ケース・ビルダーは、エクスポートの間にこの統計データをステージング表に格納します。インポートの間には、SQLテスト・ケース・ビルダーによって、統計履歴のデータがステージング表からターゲット・データベースに自動的にリロードされます。

- 文の履歴

  文の履歴は、適応カーソル共有、静的フィードバックおよびカーソル共有の不具合に関する問題の診断の際に重要です。この履歴には、実行計画とコンパイル統計および実行統計が含まれます。

関連項目:

- 適応問合せ計画、補助的な動的統計、自動再最適化およびSQL計画ベースラインの詳細は、Oracle Database SQLチューニング・ガイドを参照してください。
- DBMS_STATSパッケージについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: SQLテスト・ケース・ビルダーの概念

9.3.4.2.3 SQLテスト・ケース・ビルダーの出力

SQLテスト・ケース・ビルダーの出力は、環境と必要なすべてのオブジェクトの再作成に必要なコマンドが格納された一連のファイルです。

SQLテスト・ケース・ビルダーでは、デフォルトで次のディレクトリにファイルが格納されます。ここでのincnumはインシデント番号を指し、runnumは実行番号を指します。

```
$ADR_HOME/incident/incdir_incnum/SQLTCB_runnum
```

たとえば、有効な出力ファイル名は次のようになります。

```
$ORACLE_HOME/log/diag/rdbms/dbsa/dbsa/incident/incdir_2657/SQLTCB_1
```

次の例に示すように、SQL_TCB_DIRという名前のディレクトリ・オブジェクトを作成し、プロシージャDBMS_SQLDIAG.EXPORT_SQL_TESTCASEを実行することで、SQLテスト・ケース・ビルダー・ファイルの格納先となる特定のディレクトリを指定することもできます。

```
CREATE OR REPLACE DIRECTORY SQL_TCB_DIR '/tmp';
DECLARE
tc CLOB;
BEGIN
  DBMS_SQLDIAG.EXPORT_SQL_TESTCASE (
    directory => 'SQL_TCB_DIR',
    sql_text  => 'select * from hr_table',
    testcase  => tc);
END;
```

ノート:

データベース管理者は、ディレクトリ・オブジェクト SQL_TCB_DIR で指定されたオペレーティング・システム・ディレクトリに対する読取りおよび書込みアクセス権限を持っている必要があります。

DBMS_SQLDIAG.EXPORT_SQL_TESTCASEプロシージャのtestcase_nameパラメータを使用してテスト・ケースの名前を指定することもできます。テスト・ケース名は、SQLテスト・ケース・ビルダーによって生成されるすべてのファイルの接頭辞として使用されます。

テスト・ケース名を指定しない場合、SQLテスト・ケース・ビルダーによって、次の形式のデフォルト・テスト・ケース名が使用されます。

```
oratcb_connectionId_sqlId_sequenceNumber_sessionId
```

ここで、connectionIdはデータベース接続ID、sqlIdはSQL文ID、sequenceNumberは内部順序番号、sessionIdはデータベース・セッションIDです。

DBMS_SQLDIAG.EXPORT_SQL_TESTCASEプロシージャのctrlOptionsパラメータを使用してSQLテスト・ケース・ビルダーの出力に含める追加情報を指定することもできます。ctrlOptionsパラメータに指定できるオプションの一部を次に示します。

- compress: このオプションは、SQLテスト・ケース・ビルダー出力ファイルをzipファイルに圧縮するために使用します。
- diag_event: このオプションは、SQLテスト・ケース・ビルダー出力に含めるトレース情報のレベルを指定するために使用します。
- problem_type: このオプションは、SQLテスト・ケース・ビルダーのテスト・ケースの問題タイプを割り当てるために使用します。たとえば、テスト・ケースがパフォーマンス回帰の問題に関連している場合は、problem_typeオプションにPERFORMANCEの値を割り当てることができます。

次の例に示すように、V$SQL_TESTCASESビューを問い合せることで、SQLテスト・ケース・ビルダーによって生成されたすべて

のテスト・ケースに関する情報を表示できます。

```
select testcase_name, sql_text from v$sql_testcases;
TESTCASE_NAME                              SQL_TEXT
-----------------------------------------  ----------------------
oratcb_0_am8q8kudm02v9_1_00244CC50001   select * from hr_table
```

> ノート:
>
> V$SQL_TESTCASES ビューには、SQL_TCB_DIR という名前の SQL テスト・ケース・ビルダーのルート・ディレクトリ・オブジェクトが存在する必要があります。Oracle Autonomous Database 環境では、プロビジョニング中にこのディレクトリ・オブジェクトが各 POD に自動的に作成されます。オンプレミス・データベースの場合は、SQL テスト・ケース・ビルダーのルート・ディレクトリ・オブジェクト SQL_TCB_DIR を明示的に作成する必要があり、作成しないと V$SQL_TESTCASES ビューに情報が表示されません。データベース管理者は、ディレクトリ・オブジェクト SQL_TCB_DIR で指定されたオペレーティング・システム・ディレクトリに対する読取りおよび書込みアクセス権限を持っている必要があります。

関連項目:

- DBMS_SQLDIAG.EXPORT_SQL_TESTCASEプロシージャの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- V$SQL_TESTCASESビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: SQLテスト・ケース・ビルダーの概念

9.3.4.3 SQLテスト・ケース・ビルダーのユーザー・インタフェース

SQLテスト・ケース・ビルダーにアクセスするには、Cloud Controlを使用するか、コマンドラインでPL/SQLを使用します。

この項では、次の項目について説明します。

- SQLテスト・ケース・ビルダーのグラフィカル・インタフェース
  Cloud Control内では、「インシデント・マネージャ」ページまたは「サポート・ワークベンチ」ページからSQLテスト・ケース・ビルダーにアクセスできます。
- SQLテスト・ケース・ビルダーのコマンドライン・インタフェース
  DBMS_SQLDIAGパッケージは、SQLテスト・ケース・ビルダーに関連するタスクを実行します。

親トピック: SQLテスト・ケース・ビルダーを使用したテスト・ケースの作成

9.3.4.3.1 SQLテスト・ケース・ビルダーのグラフィカル・インタフェース

Cloud Control内では、インシデント・マネージャ・ページまたはサポート・ワークベンチ・ページからSQLテスト・ケース・ビルダーにアクセスできます。

この項では、次の項目について説明します。

- インシデント・マネージャへのアクセス
  データベース・ホーム・ページの「インシデントと問題」セクションから、インシデント・マネージャに移動できます。
- サポート・ワークベンチへのアクセス

「Oracleデータベース」メニューから、サポート・ワークベンチに移動できます。

親トピック: SQLテスト・ケース・ビルダーのユーザー・インタフェース

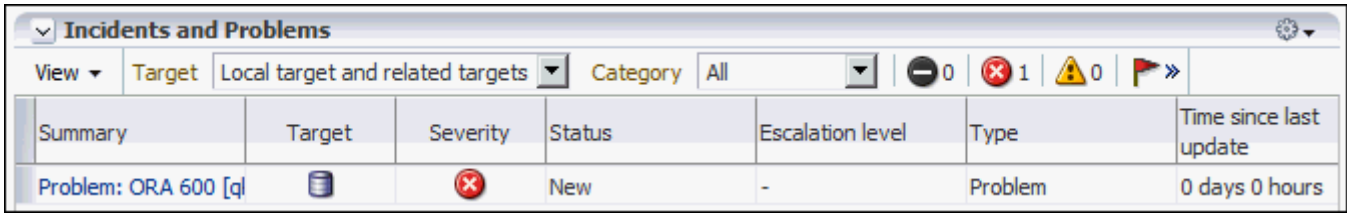
9.3.4.3.1.1 インシデント・マネージャへのアクセス

データベース・ホーム・ページの「インシデントと問題」セクションから、インシデント・マネージャに移動できます。

インシデント・マネージャにアクセスするには:

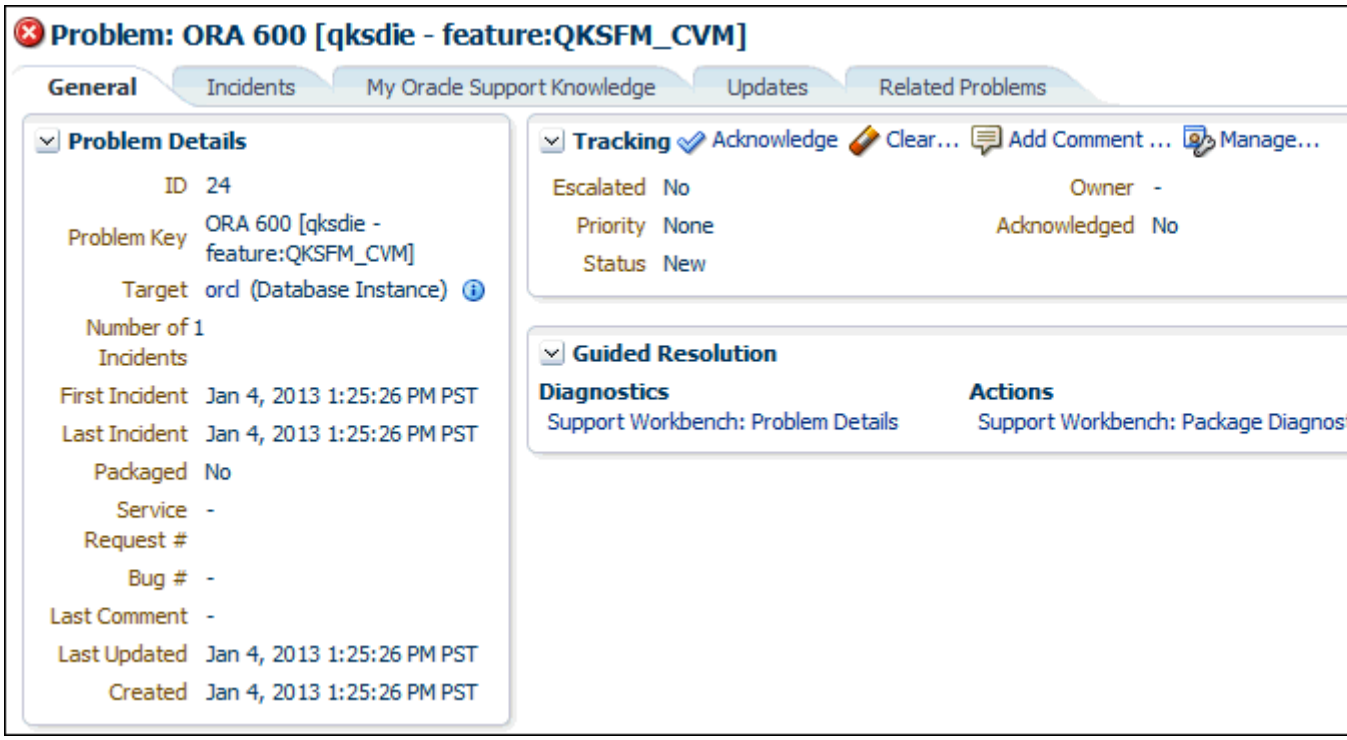
1. 適切な資格証明を使用してCloud Controlにログインします。
2. 「ターゲット」メニューの下で、「データベース」を選択します。
3. データベース・ターゲットのリストで、管理対象のOracle Databaseインスタンスのターゲットを選択します。
4. データベースの資格証明の入力を求められた場合は、実行するタスクに必要な最小限の資格証明を入力します。
5. インシデントと問題セクションで、調査するSQLインシデントを特定します。

   次の例のように、ORA 600エラーはSQLインシデントです。



6. インシデントの概要をクリックします。

   「インシデント・マネージャ」の「問題の詳細」ページが表示されます。



   表にリストされたインシデントを含む「サポート・ワークベンチ」ページが表示されます。

親トピック: SQLテスト・ケース・ビルダーのグラフィカル・インタフェース

9.3.4.3.1.2 サポート・ワークベンチへのアクセス

「Oracleデータベース」メニューから、サポート・ワークベンチに移動できます。

サポート・ワークベンチにアクセスするには:

1. 適切な資格証明を使用してCloud Controlにログインします。
2. 「ターゲット」メニューの下で、「データベース」を選択します。
3. データベース・ターゲットのリストで、管理対象のOracle Databaseインスタンスのターゲットを選択します。
4. データベースの資格証明の入力を求められた場合は、実行するタスクに必要な最小限の資格証明を入力します。
5. 「Oracle Database」メニューから、「診断」→「サポート・ワークベンチ」を選択します。

   表にリストされたインシデントを含む「サポート・ワークベンチ」ページが表示されます。

親トピック: SQLテスト・ケース・ビルダーのグラフィカル・インタフェース

9.3.4.3.2 SQLテスト・ケース・ビルダーのコマンドライン・インタフェース

DBMS_SQLDIAGパッケージは、SQLテスト・ケース・ビルダーに関連するタスクを実行します。

このパッケージは、SQLテスト・ケース・ビルダー用の様々なサブプログラムで構成されます。次の表にその一部を示します。

表9-11 DBMS_SQLDIAGパッケージ内のSQLテスト・ケース・ファンクション

プロシージャ	説明
EXPORT_SQL_TESTCASE	SQL テスト・ケースをユーザー指定のディレクトリにエクスポートします。
EXPORT_SQL_TESTCASE_DIR_BY_INC	引数として渡されたインシデント ID に対応する SQL テスト・ケースをエクスポートします。
EXPORT_SQL_TESTCASE_DIR_BY_TXT	引数として渡された SQL テキストに対応する SQL テスト・ケースをエクスポートします。
IMPORT_SQL_TESTCASE	SQL テスト・ケースをスキーマにインポートします。
REPLAY_SQL_TESTCASE	SQL テスト・ケースの再現を自動化します。
EXPLAIN_SQL_TESTCASE	SQL テスト・ケースについて記述します。

関連項目:

DBMS_SQLDIAGパッケージについてさらに学習するには、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください

親トピック: SQLテスト・ケース・ビルダーのユーザー・インタフェース

9.3.4.4 SQLテスト・ケース・ビルダーの実行

Cloud Controlを使用してSQLテスト・ケース・ビルダーを実行できます。

前提条件

このチュートリアルでは、次のことが前提となっています。

- 内部エラーが発生する次のEXPLAIN PLAN文をユーザーshとして実行します。

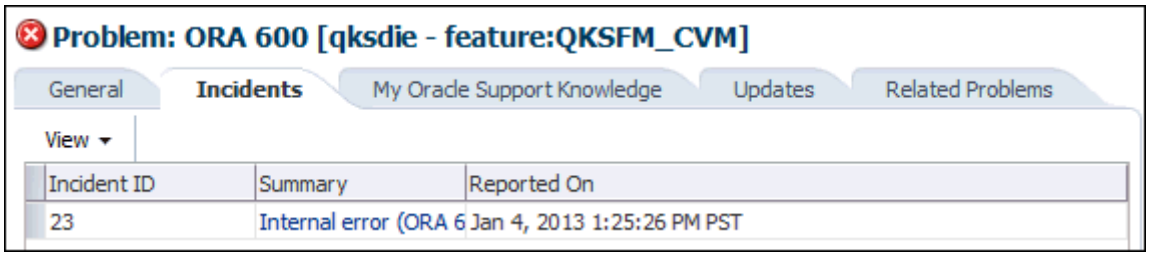
```
EXPLAIN PLAN FOR
  SELECT unit_cost, sold
  FROM   costs c,
         ( SELECT /*+ merge */ p.prod_id, SUM(quantity_sold) AS sold
           FROM   products p, sales s
           WHERE  p.prod_id = s.prod_id
           GROUP BY p.prod_id ) v
  WHERE  c.prod_id = v.prod_id;
```

- データベース・ホーム・ページのインシデントと問題セクションに、内部エラーによって生成されるSQLインシデントが表示されます。
- 「インシデント・マネージャへのアクセス」の説明に従って、「インシデントの詳細」ページにアクセスします。

SQLテスト・ケース・ビルダーを実行するには:

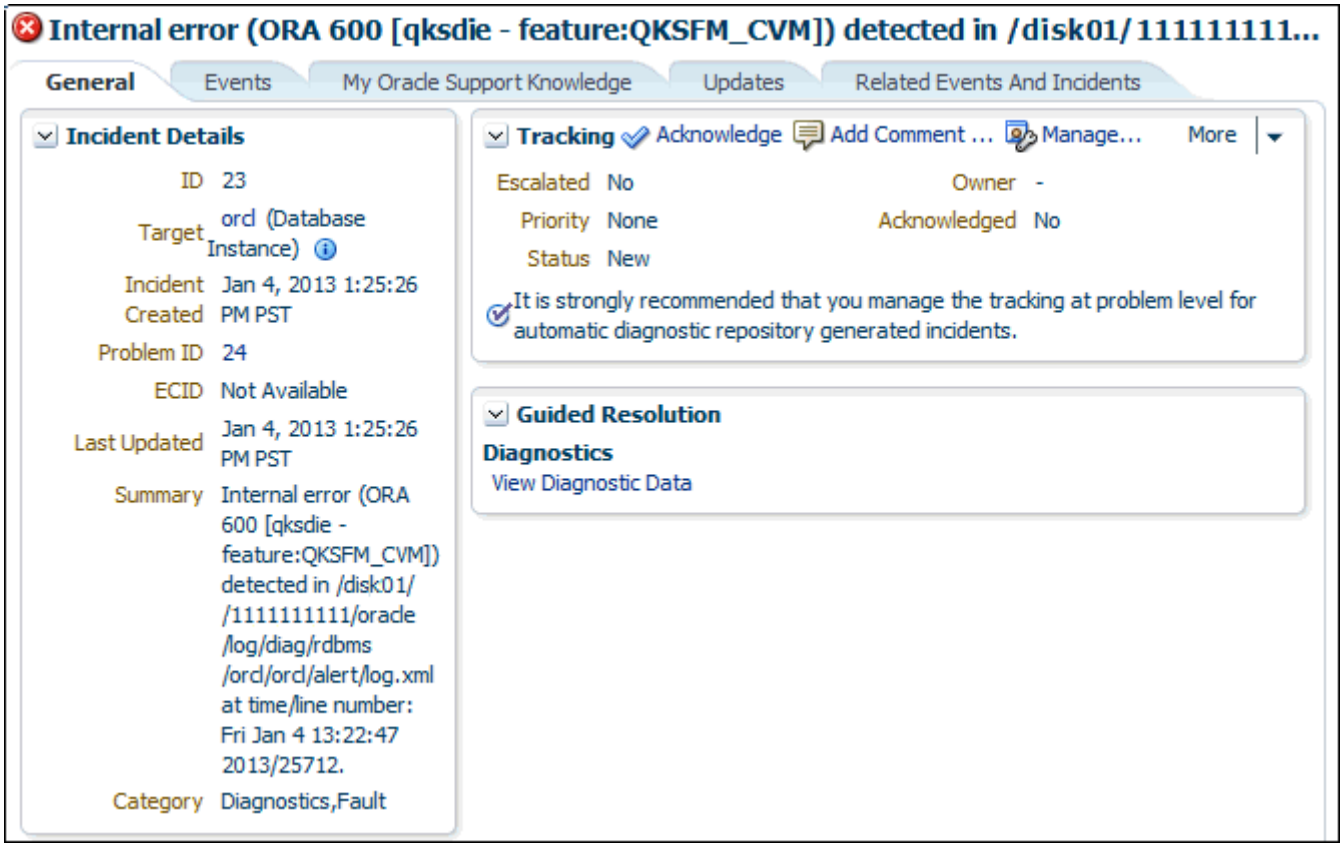
1. 「インシデント」タブをクリックします。

   問題の詳細ページが表示されます。



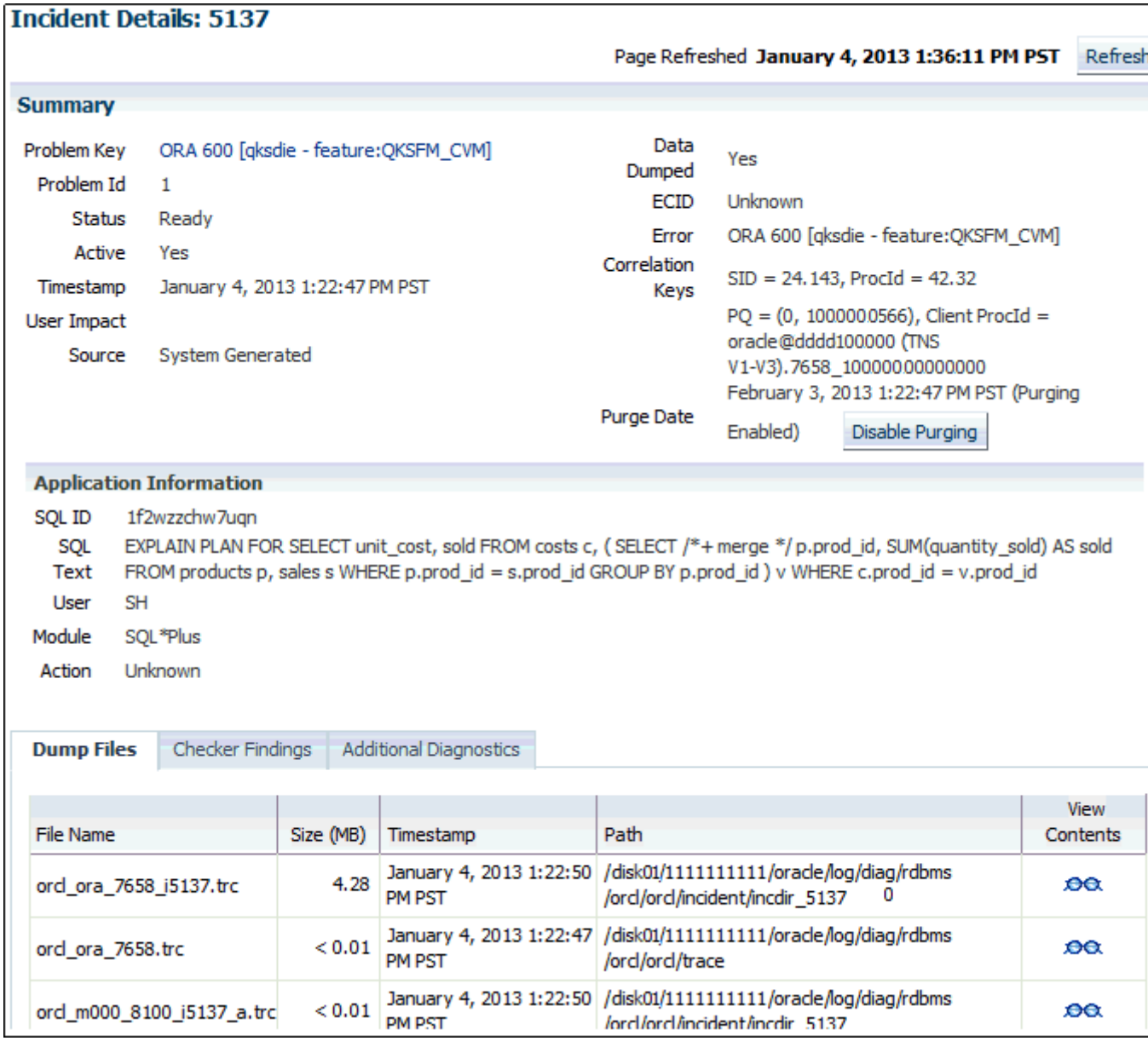
2. インシデントの概要をクリックします。

   インシデントの詳細ページが表示されます。



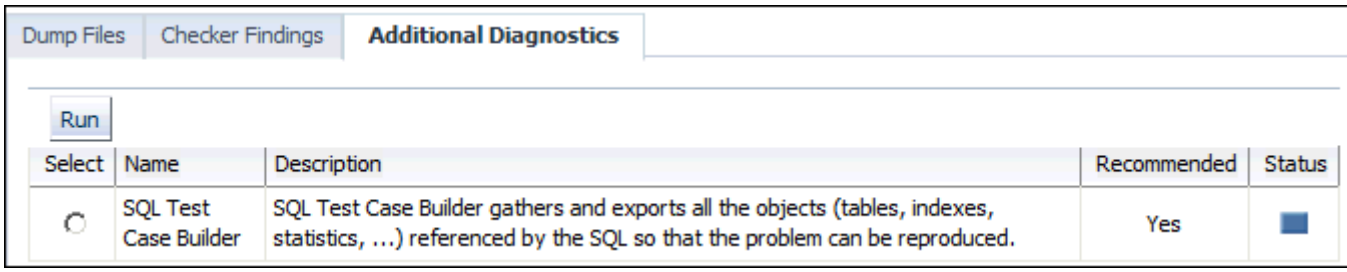
3. 「ガイドされた解決」で、「診断データの表示」をクリックします。

   「インシデントの詳細: incident_number」ページが表示されます。



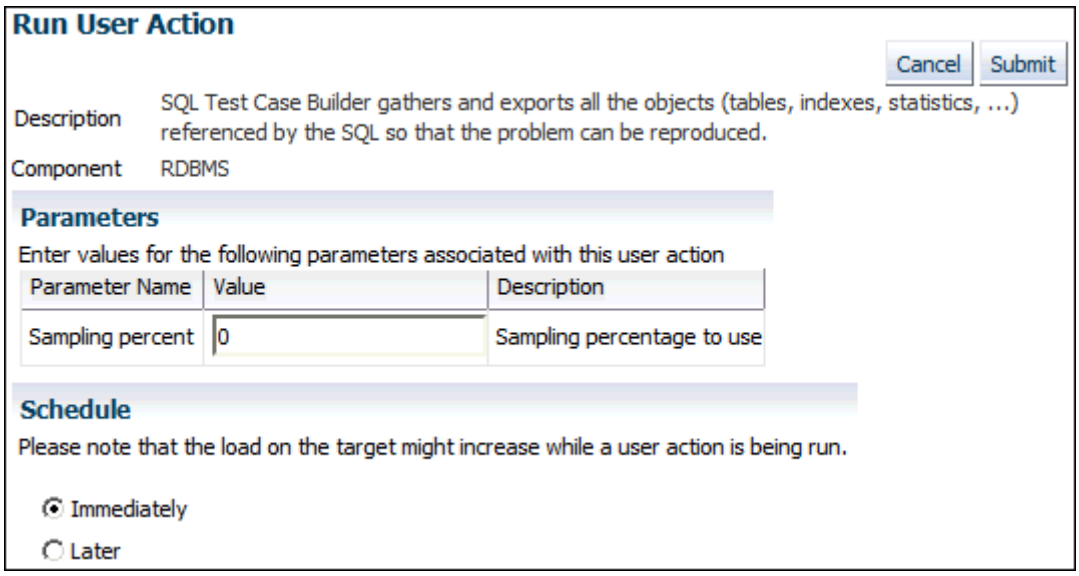
4. 「アプリケーション情報」セクションで、「追加の診断」をクリックします。

   「追加の診断」サブページが表示されます。



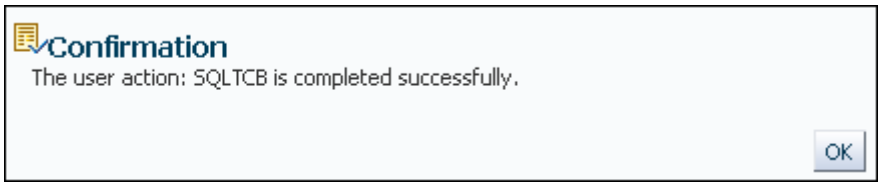
5. 「SQLテスト・ケース・ビルダー」を選択して、「実行」をクリックします。

   「ユーザー処理を実行」ページが表示されます。



6. サンプリング割合を選択し(オプション)、「送信」をクリックします。

   処理が完了すると、「確認」ページが表示されます。



7. 「SQLテスト・ケース・ビルダーの出力」に説明されている場所にあるSQLテスト・ケース・ファイルにアクセスします。

親トピック: SQLテスト・ケース・ビルダーを使用したテスト・ケースの作成

9.4 問題の報告

Enterprise Managerサポート・ワークベンチ(サポート・ワークベンチ)を使用して、カスタム・インシデント・パッケージを作成、編集およびアップロードできます。カスタム・インシデント・パッケージにより、Oracleサポート・サービスに送信する診断データを詳細に制御できます。

- インシデント・パッケージ
  インシデント・パッケージ(パッケージ)と呼ばれる中間論理構造に診断データを収集できます。
- カスタム・パッケージングを使用した問題のパッケージ化とアップロード
  カスタム・インシデント・パッケージ(パッケージ)を作成し、アップロードするには、サポート・ワークベンチ(サポート・ワークベンチ)を使用します。アップロードする前に、診断データ・ファイルをパッケージから手動で追加、編集、および削除できます。
- インシデント・パッケージの表示と変更
  カスタム・パッケージング方法を使用してインシデント・パッケージを作成した後、Oracleサポートにアップロードする前に、そのパッケージのコンテンツを表示または変更できます。
- 相関パッケージの作成、編集およびアップロード
  Oracleサポートにパッケージをアップロードした後、1つ以上の相関パッケージを作成およびアップロードできます。
- 相関パッケージの削除
  相関パッケージは、そのパッケージを作成したターゲットのサポート・ワークベンチで削除します。
- インシデント・パッケージのプリファレンスの設定
  インシデント・パッケージのプリファレンスを設定できます。インシデント・パッケージのプリファレンスの例には、インシデント情報を保持する日数や、各問題のパッケージに含める最初と最後のインシデント数などがあります。

関連項目:

「Oracle Databaseの障害診断インフラストラクチャについて」

親トピック: 問題の診断と解決

9.4.1 インシデント・パッケージ

インシデント・パッケージ(パッケージ)と呼ばれる中間論理構造に診断データを収集できます。

- インシデント・パッケージについて
  Oracleサポートに診断データをアップロードするためのカスタマイズ・アプローチでは、最初に、インシデント・パッケージ(パッケージ)と呼ばれる中間論理構造内にデータを収集します。
- インシデント・パッケージ内の関係付けられた診断データについて
  問題を診断するには、問題に直接関連する診断データのみでなく、直接関連するデータと相関する診断データも調べることが必要になる場合があります。
- クイック・パッケージングとカスタム・パッケージングについて
  サポート・ワークベンチでは、インシデント・パッケージを作成してアップロードするために、クイック・パッケージング方法とカスタム・パッケージング方法の2つの方法を提供しています。
- 相関パッケージについて
  相関パッケージは、関連する問題の診断データをパッケージおよびアップロードするための手段を提供します。

親トピック: 問題の報告

9.4.1.1 インシデント・パッケージについて

診断データをカスタマイズしてOracleサポート・サービスにアップロードするには、最初に、インシデント・パッケージ(パッケージ)と呼ばれる中間論理構造にデータを収集します。

パッケージは自動診断リポジトリ(ADR)に格納されているメタデータの集合で、ADR内外の診断データファイルやその他のファイルを指し示します。パッケージを作成するときは、そのパッケージに追加する問題を1つ以上選択します。次に、サポート・ワークベンチによって、選択した問題に関連する問題情報、インシデント情報および診断データ(トレース・ファイル、ダンプなど)がパッケージに自動的に追加されます。1つの問題に対して多数のインシデント(同じ問題の多数の発生)が存在する場合があるため、デフォルトでは、各問題の最初の3つと最後の3つのインシデントのみがパッケージに追加され、発生から90日を超えるインシデントは除外されます。このデフォルトの数は、サポート・ワークベンチの「インシデント・パッケージング構成」ページで変更できます。

パッケージが作成されたら、あらゆるタイプの外部ファイルをパッケージに追加し、パッケージから選択したファイルを削除し、重要なデータを削除するパッケージ内の選択したファイルを編集できます。パッケージの内容を追加および削除する場合は、パッケージのメタデータのみが変更されます。

診断データをOracleサポート・サービスにアップロードする準備ができたら、最初に、パッケージのメタデータで参照されるすべてのファイルを含めてZIPファイルを作成します。次に、Oracle Configuration Managerを使用してそのZIPファイルをアップロードします。

ノート:

Oracle Configuration Manager をインストールしていないか、適切に構成していない場合は、My Oracle

Support を使用して ZIP ファイルを手動でアップロードする必要があります。

関連トピック

- カスタム・パッケージングを使用した問題のパッケージ化とアップロード
- インシデント・パッケージの表示と変更
- 構成管理の概要

親トピック: インシデント・パッケージ

9.4.1.2 インシデント・パッケージ内の関係付けられた診断データについて

問題を診断するには、問題に直接関連する診断データのみでなく、直接関連するデータと相関する診断データも調べる必要がある場合があります。

診断データは、時間、プロセスIDまたはその他の基準によって関係付けることができます。たとえば、インシデントの検証では、そのインシデントの5分後に発生したインシデントの検証も役立つ場合があります。同様に、インシデントの診断データには、インシデント発生時に実行されていたOracle Databaseプロセスのトレース・ファイルが含まれていることは明確ですが、元のプロセスに関連する他のプロセスのトレース・ファイルを含めることが役に立つ場合もあります。

このため、問題とそれに関連するインシデントがパッケージに追加されるときは、同時に、関連するトレース・ファイルとともに関係付けられたインシデントが追加されます。

パッケージの物理ファイルを作成する過程で、サポート・ワークベンチはインシデント・パッケージング・サービスを要求してパッケージをファイナライズします。ファイナライズとは、パッケージ内のインシデントに時間で関係付けられた追加のトレース・ファイル、および他の診断情報(アラート・ログ、ヘルス・チェッカ・レポート、SQLテスト・ケース、構成情報など)をパッケージに追加することを意味します。そのため、ZIPファイル内のファイル数が、サポート・ワークベンチでパッケージ・コンテンツとして表示されたファイルの数より多くなる場合があります。

インシデント・パッケージング・サービスは一連のルールに従って、既存のパッケージ・データに関係付けるADR内のトレース・ファイルを決定します。一部のルールは、Cloud Controlの「インシデント・パッケージング構成」ページで変更できます。

当初のパッケージ・データと関係付けられた追加データには機密情報が含まれている可能性があるため、Oracleサポート・サービスにアップロードする前に、このような機密情報を含むファイルを削除または編集することが重要です。このため、サポート・ワークベンチでは、パッケージをファイナライズするコマンドを独立した操作として実行できます。パッケージを手動でファイナライズした後は、パッケージ・コンテンツを検証し、ファイルを削除または編集してから、ZIPファイルを生成してアップロードできます。

ノート:

パッケージのファイナライズは、パッケージがクローズして変更不可になることではありません。ファイナライズしたパッケージには、引続き診断データを追加できます。また、同じパッケージを複数回ファイナライズすることもできます。ファイナライズするたびに、関係付けられた新しいデータが追加されます。

関連トピック

- インシデント・パッケージのプリファレンスの設定

親トピック: インシデント・パッケージ

9.4.1.3 クイック・パッケージングとカスタム・パッケージングについて

サポート・ワークベンチには、インシデント・パッケージを作成してアップロードする方法として、クイック・パッケージング方法とカスタム・パッケージング方法があります。

クイック・パッケージング—最小限のステップがある自動化された方法で、ガイド付きワークフロー(ウィザード)に編成されています。1つの問題を選択してパッケージ名と説明を入力し、パッケージ・コンテンツのアップロードを即時に実行するか、指定の日時に実行するかをスケジュールします。サポート・ワークベンチでは、問題に関連する診断データを自動的にパッケージに追加してファイナライズし、ZIPファイルを作成してアップロードします。この方法では、パッケージ・ファイルを追加、編集または削除したり、SQLテスト・ケースなどの他の診断データを追加することはできません。ただし、この方法は、初期障害診断データをOracleサポート・サービスにアップロードする最も簡単かつ迅速な方法です。クイック・パッケージングは、「問題の調査、報告および解決について」で説明しているワークフローで使用される方法です。

クイック・パッケージングが完了した後、ウィザードで作成されたパッケージはそのまま残ります。このパッケージは、カスタム・パッケージング操作を使用して後で変更し、手動で再アップロードできます。

カスタム・パッケージング—ステップが多く、より詳細な手動の方法です。これは、パッケージング・プロセスの詳細な制御を行うエキスパートのサポート・ワークベンチ・ユーザーを対象としています。カスタム・パッケージングでは、1つ以上の問題のある新規パッケージを作成、または1つ以上の問題を既存のパッケージに追加できます。これらの各種の操作を新規または更新されたパッケージで実行でき、これには次の操作が含まれます。

- 問題やインシデントを追加または削除できます。
- パッケージに対してトレース・ファイルを追加、編集または削除できます。
- 任意のタイプの外部ファイルを追加または削除できます。
- その他の診断データ(SQLテスト・ケースなど)を追加できます。
- パッケージを手動でファイナライズしてからパッケージ・コンテンツを表示して、機密データを編集または削除する必要があるか、ファイルを削除してパッケージ・サイズを縮小する必要があるかを決定できます。

これらの操作には、Oracleサポートに送信する診断情報が十分であることを確認するまで、数日間かかる場合があります。

カスタム・パッケージングでは、ZIPファイルの作成とOracleサポートへのアップロードの要求を、2つの独立したステップとして実行できます。各ステップは、即時に実行するか、日時をスケジュールして実行できます。

関連トピック

- 問題の調査、報告および解決について
- タスク5: 診断データのパッケージ化とOracleサポート・サービスへのアップロード

親トピック: インシデント・パッケージ

9.4.1.4 相関パッケージについて

相関パッケージは、関連する問題に対して診断データのパッケージングおよびアップロードのための手段を提供します。

データベース・インスタンスの問題は、他のデータベース・インスタンスまたはOracle Automatic Storage Managementインスタンスに関連する問題がある場合があります。データベース・インスタンスの問題に対するパッケージ(メイン・パッケージ)を作成およびアップロードした後、関連する問題についての相関パッケージを作成およびアップロードできます。これは、サポート・ワークベンチのカスタム・パッケージング・ワークフローでのみ実行できます。

関連トピック

- トポロジ全体に関連する問題

- 相関パッケージの作成、編集およびアップロード

親トピック: インシデント・パッケージ

9.4.2 カスタム・パッケージングを使用した問題のパッケージ化とアップロード

サポート・ワークベンチ(サポート・ワークベンチ)を使用して、カスタム・インシデント・パッケージ(パッケージ)を作成およびアップロードできます。アップロードする前に、診断データ・ファイルをパッケージから手動で追加、編集、および削除できます。

カスタム・パッケージングを使用して問題をパッケージ化およびアップロードするには:

1. 「サポート・ワークベンチ」ホームページにアクセスします。

   手順については、「サポート・ワークベンチを使用した問題の表示」を参照してください。

2. (オプション)パッケージに含める各問題について、問題に関連付けられているサービス・リクエスト番号(SR#)を指定します(ある場合)。指定するには、各問題について次のステップを実行します。

   a. 「サポート・ワークベンチ」ホームページの下部にある「問題」サブページで、問題を選択して「表示」をクリックします。

   > ノート:
   >
   > 問題のリストに対象の問題がみつからない場合、または問題数が多くてスクロールできない場合は、「表示」リストから期間を選択して「実行」をクリックします。対象の問題を選択して「表示」をクリックします。

   問題の詳細ページが表示されます。

   b. SR#ラベルの横にある「編集」をクリックし、サービス・リクエスト番号を入力して、「OK」をクリックします。

   「問題の詳細」ページにサービス・リクエスト番号が表示されます。

   c. ページの上部にあるロケータ・リンクで「サポート・ワークベンチ」をクリックし、「サポート・ワークベンチ」ホームページに戻ります。

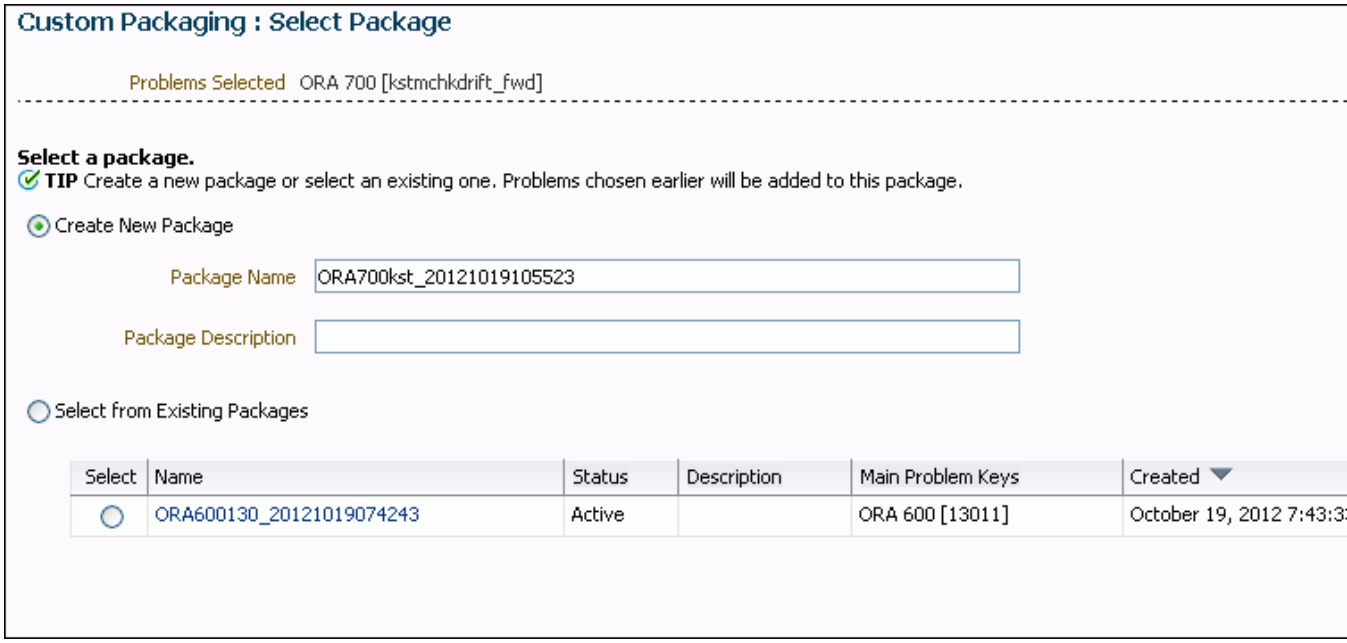
3. 「サポート・ワークベンチ」ホームページで、パッケージ化する問題を選択して「パッケージ」をクリックします。

   「パッケージング・モードの選択」ページが表示されます。

   > ノート:
   >
   > パッケージング・プロセスでは、関係付けられた追加の問題を自動的に選択してパッケージに追加できます。関係付けられた問題の例には、選択した問題の数分後に発生した問題があります。詳細は、「インシデント・パッケージ内の関係付けられた診断データについて」を参照してください。

4. 「カスタム・パッケージング」オプションを選択して「続行」をクリックします。

   「カスタム・パッケージング: パッケージの選択」ページが表示されます。

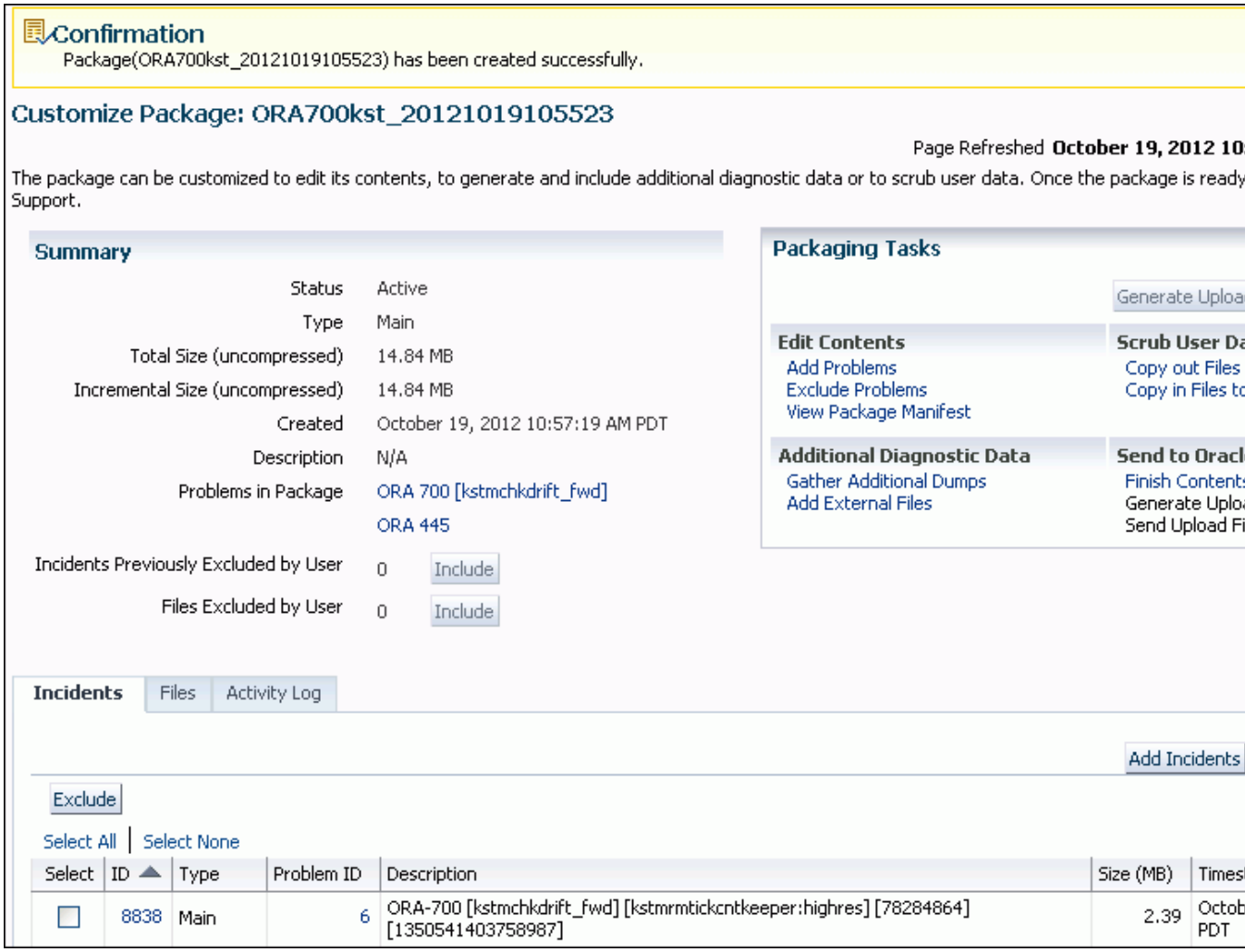


5. 次のいずれかの操作を行います。

   a. 新規パッケージを作成するには、「新規パッケージの作成」オプションを選択し、パッケージ名と説明を入力して「OK」をクリックします。

   b. 選択した問題を既存のパッケージに追加するには、「既存パッケージからの選択」オプションを選択し、更新するパッケージを選択して「OK」をクリックします。

「パッケージのカスタマイズ」ページが表示されます。選択するパッケージ・タスクのセレクションに加え、パッケージに含まれている問題およびインシデントが表示されます。新しいパッケージまたは更新した既存パッケージに対して実行できます。



6. (オプション)「パッケージング・タスク」セクションで、リンクをクリックして1つ以上のパッケージング・タスクを実行します。または、「パッケージのカスタマイズ」ページとそのサブページにある他のコントロールを使用して、パッケージを操作します。完了後に「パッケージのカスタマイズ」ページに戻ります。

   いくつかの一般的なパッケージング・タスクの手順については、「インシデント・パッケージの表示と変更」を参照してください。

7. 「パッケージのカスタマイズ」ページの「パッケージング・タスク」セクションで、「Oracleサポートに送信」ヘッダー下にある「コンテンツ準備の終了」をクリックしてパッケージをファイナライズします。

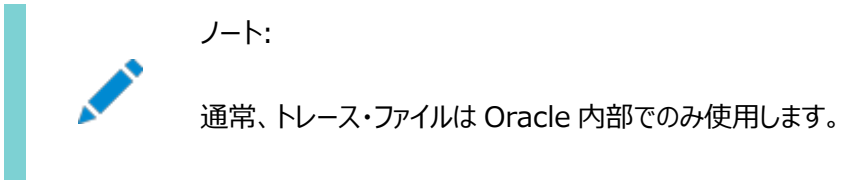
   パッケージに含まれるファイルのリスト(または部分的なリスト)が表示されます。(これには少し時間がかかる場合があります。)リストには、相関診断情報を含むかどうかを識別し、終了プロセスによって追加されたファイルが含まれます。

   パッケージのファイナライズの定義の概要については、「インシデント・パッケージ内の関係付けられた診断データについて」を参照してください。

8. 「ファイル」をクリックして、パッケージ内のファイルをすべて表示します。リストを検証して、公開できない機密データのファイルがあるかどうかを確認します。該当するファイルが見つかった場合は、それらのファイルを除外(削除)または編集します。

   ファイルを編集および削除する手順については、「インシデント・パッケージ・ファイルの編集(コピー・アウトとコピー・イン)」および「インシデント・パッケージ・ファイルの削除」を参照してください。

   ファイルの内容を表示するには、ファイルの表の右側の列にある眼鏡アイコンをクリックします。ホスト資格証明を入力します(要求された場合)。



ノート:

通常、トレース・ファイルは Oracle 内部でのみ使用します。

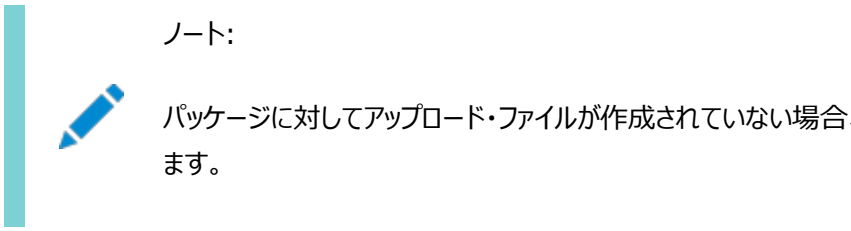
9. 「アップロード・ファイルの生成」をクリックします。

「アップロード・ファイルの生成」ページが表示されます。

10. 「全体」または「増分」オプションを選択して、全パッケージのZIPファイルまたは増加分パッケージのZIPファイルを生成します。

全パッケージのZIPファイルの場合は、常にパッケージのすべてのコンテンツ(元のコンテンツおよび関係付けられたすべてのデータ)がZIPファイルに追加されます。

増加分パッケージのZIPファイルの場合は、同じパッケージのZIPファイルが最後に作成された時点以降に新規追加または変更された診断情報のみがZIPファイルに追加されます。たとえば、パッケージに対して生成された物理ファイルにトレース・ファイルが最後に追加された時点以降にトレース情報がそのトレース・ファイルに追加された場合は、そのトレース・ファイルが増加分パッケージのZIPファイルに追加されます。逆に、パッケージに対して最後にアップロードした時点以降にトレース・ファイルを変更していない場合、そのトレース・ファイルは増加分パッケージのZIPファイルに追加されません。



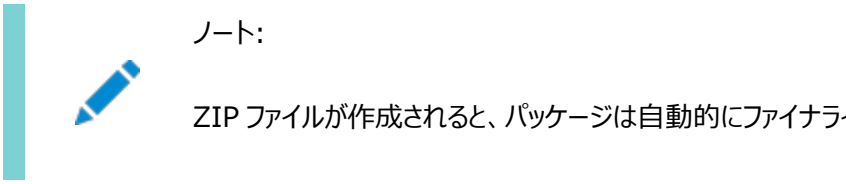
ノート:

パッケージに対してアップロード・ファイルが作成されていない場合、「増分」オプションは使用不可になります。

11. ファイル作成を即時実行するか、後の日時に実行するかをスケジュールして(「即時」または「後で」を選択)、「発行」をクリックします。

ファイル作成ではシステム・リソースを大量に使用する場合があるため、ファイルの作成は、システム使用量が少ない時期にスケジュールすることをお薦めします。

「処理中」ページが表示され、ZIPファイルが作成されます。処理が完了すると、確認ページが表示されます。



ノート:

ZIP ファイルが作成されると、パッケージは自動的にファイナライズされます。

12. 「OK」をクリックします。

「パッケージのカスタマイズ」ページに戻ります。

13. 「Oracleに送信」をクリックします。

「アップロード・ファイルの表示/送信」ページが表示されます。

14. (オプション)相関パッケージを作成してオラクル社に送信するには、「相関パッケージの送信」リンクをクリックします。

「相関パッケージの作成、編集およびアップロード」を参照してください。相関パッケージに対する作業が完了した後、ページの上部にある「パッケージの詳細」リンクをクリックし、「パッケージのカスタマイズ」をクリックし、次に、再度「Oracle

に送信」をクリックして、「アップロード・ファイルの表示/送信」ページに戻ります。

15. アップロードするZIPファイルを選択して「Oracleに送信」をクリックします。

「Oracleに送信」ページが表示されます。選択したZIPファイルが表にリストされます。

16. 要求されたMy Oracle Support情報を入力します。「新規サービス・リクエスト(SR)の作成」の横にある「はい」または「いいえ」を選択します。「はい」を選択すると、ドラフトのサービス・リクエストが作成されます。この場合は、後でMy Oracle Supportにログインして、サービス・リクエストの詳細を入力する必要があります。「いいえ」を選択した場合は、既存のサービス・リクエスト番号を入力します。

17. アップロードを即時または後の日時にスケジュールして、「発行」をクリックします。

「処理中」ページが表示されます。アップロードが正常に完了した場合は、確認ページが表示されます。アップロードが完了できなかった場合は、エラー・ページが表示されます。エラー・ページには、ZIPファイルをOracleに手動でアップロードするように要求するメッセージが表示される場合があります。その場合の手順については、Oracleサポート・サービス担当者に問い合せてください。

18. 「OK」をクリックします。

「アップロード・ファイルの表示/送信」ページに戻ります。「送信時刻」列で、アップロードしたファイルのステータスをチェックします。

ノート:

サポート・ワークベンチでは、Oracle Configuration Manager を使用して物理ファイルをアップロードします。Oracle Configuration Manager がインストールされていなかったり、適切に構成されていないと、アップロードに失敗する場合があります。失敗した場合は、ファイルを Oracle サポート・サービスに手動でアップロードするように要求するメッセージが、パッケージの ZIP ファイルへのパスとともに表示されます。アップロードは、My Oracle Support を使用して手動で実行できます。

Oracle Configuration Manager の詳細は、『Oracle Configuration Manager インストレーションおよび管理ガイド』を参照してください。

19. (オプション)相関パッケージを作成およびアップロードします。

手順については、「相関パッケージの作成、編集およびアップロード」を参照してください。

関連項目:

- 「インシデントおよび問題について」
- 「インシデント・パッケージについて」
- 「クイック・パッケージングとカスタム・パッケージングについて」

親トピック: 問題の報告

9.4.3 インシデント・パッケージの表示と変更

カスタム・パッケージング方法を使用してインシデント・パッケージを作成した後は、Oracleサポート・サービスへのアップロード前に、

そのパッケージのコンテンツを表示または変更できます。

さらに、クイック・パッケージング方法を使用して診断データをパッケージ化してアップロードした後は、サポート・ワークベンチで作成したパッケージのコンテンツを表示または変更して、パッケージを再アップロードできます。パッケージを変更するには、パッケージ・タスクの選択対象から選択しますが、それらのほとんどは「パッケージのカスタマイズ」ページで使用できます。

- パッケージの詳細の表示
  「パッケージの詳細」ページには、パッケージ内のインシデント、トレース・ファイルおよびその他のファイルに関する情報が含まれており、アクティビティ・ログを表示してパッケージに追加できます。
- 「パッケージのカスタマイズ」ページへのアクセス
  「パッケージのカスタマイズ」ページでは、問題の追加および削除、パッケージ・ファイルの追加、削除およびスクラブ(編集)、パッケージのZIPファイルの生成およびアップロードなどの様々なパッケージング・タスクを実行します。
- インシデント・パッケージ・ファイルの編集(コピー・アウトとコピー・イン)
  サポート・ワークベンチでは、インシデント・パッケージ内の1つ以上のファイルを編集できます。
- インシデント・パッケージへの外部ファイルの追加
  インシデント・パッケージには、任意のタイプの外部ファイルを追加できます。
- インシデント・パッケージ・ファイルの削除
  インシデント・パッケージから任意のタイプのファイルを1つ以上削除できます。
- インシデント・パッケージのアクティビティ・ログの表示と更新
  サポート・ワークベンチでは、各インシデント・パッケージのアクティビティ・ログが保持されます。

関連項目:

- 「インシデント・パッケージについて」
- 「カスタム・パッケージングを使用した問題のパッケージ化とアップロード」

親トピック: 問題の報告

9.4.3.1 パッケージの詳細の表示

「パッケージの詳細」ページには、パッケージ内のインシデント、トレース・ファイルおよびその他のファイルに関する情報が含まれており、アクティビティ・ログを表示してパッケージに追加できます。

パッケージの詳細を表示するには:

1. 「サポート・ワークベンチ」ホームページにアクセスします。

   手順については、「サポート・ワークベンチを使用した問題の表示」を参照してください。

2. 「パッケージ」をクリックして「パッケージ」のサブページを表示します。

   自動診断リポジトリ(ADR)に現時点で格納されているパッケージのリストが表示されます。

3. (オプション)表示するパッケージの数を減らすには、リストの上部にある「検索」フィールドにテキストを入力して「実行」をクリックします。

   パッケージ名に検索テキストが含まれているパッケージがすべて表示されます。パッケージの完全なリストを表示するには、「検索」フィールドからテキストを削除し、「実行」を再度クリックします。

4. 「パッケージ名」列で、対象のパッケージのリンクをクリックします。

   「パッケージの詳細」ページが表示されます。

親トピック: インシデント・パッケージの表示と変更

9.4.3.2 「パッケージのカスタマイズ」ページへのアクセス

「パッケージのカスタマイズ」ページでは、各種のパッケージング・タスク(問題の追加および削除、パッケージ・ファイルの追加、削除およびスクラブ(編集)、パッケージのZIPファイルの生成およびアップロードなど)を実行します。

「パッケージのカスタマイズ」ページにアクセスするには:

1. 「パッケージの詳細の表示」で説明しているように、特定のパッケージの「パッケージの詳細」ページにアクセスします。
2. 「パッケージのカスタマイズ」をクリックします。

   「パッケージのカスタマイズ」ページが表示されます。

親トピック: インシデント・パッケージの表示と変更

9.4.3.3 インシデント・パッケージ・ファイルの編集(コピー・アウトとコピー・イン)

サポート・ワークベンチを使用すると、インシデント・パッケージ内の1つ以上のファイルを編集できます。

この作業は、ファイル内の機密データを削除または上書きする場合に必要です。パッケージ・ファイルを編集するには、最初にパッケージから宛先ディレクトリにファイルをコピー・アウトし、テキスト・エディタまたは他のユーティリティを使用して編集してから、そのファイルをパッケージにコピーし直して、元のパッケージ・ファイルを上書きします。

次の手順では、パッケージがすでに作成されて診断データが格納されていることを前提としています。

インシデント・パッケージ・ファイルを編集するには:

1. 対象のインシデント・パッケージの「パッケージのカスタマイズ」ページにアクセスします。

   手順については、「「パッケージのカスタマイズ」ページへのアクセス」を参照してください。

2. 「パッケージング・タスク」セクションで、「ユーザー・データのスクラブ」ヘッダー下にある「コンテンツ編集のためのファイルのコピー・アウト」をクリックします。

   ホスト資格証明を要求された場合は、資格証明を入力し、「OK」をクリックします。

   ファイルのコピー・アウト・ページが表示されます。ファイルをコピーできるホスト名が表示されます。

3. 次のいずれかを実行して、ファイルの宛先ディレクトリを指定します。

   - 「宛先フォルダ」フィールドにディレクトリ・パスを入力します。
   - 「宛先フォルダ」フィールドの横にある拡大鏡アイコンをクリックして、次のステップを実行します。

     a. ホスト資格証明がプロンプトで要求された場合は、ファイルのコピー・アウト先のホストの資格証明を入力して、「OK」をクリックします。(「優先資格証明として保存」を選択すると次回から資格証明を要求するプロンプトが表示されなくなります)。

        「参照および選択: ファイルまたはディレクトリ」ウィンドウが表示されます。

     b. 対象の宛先ディレクトリを選択して「選択」をクリックします。

        「参照および選択: ファイルまたはディレクトリ」ウィンドウがクローズし、選択したディレクトリへのパスが「ファイルのコピー・アウト」ページの「宛先フォルダ」フィールドに表示されます。

4. 「コピー・アウトするファイル」で、対象のファイルを選択して「OK」をクリックします。

ノート:

対象のファイルが見つからない場合は、別のページにある場合があります。「次」リンクをクリックして、次のページを表示します。対象のファイルがみつかるまで、「次」をクリックし続けるか、ファイル番号のリスト(「次」リンクの左側)から番号を選択します。ファイルを選択して「OK」をクリックします。

「パッケージのカスタマイズ」ページに戻ると、コピー・アウトされたファイルをリストした確認メッセージが表示されます。

5. テキスト・エディタまたは他のユーティリティを使用して、ファイルを編集します。
6. 「パッケージのカスタマイズ」ページの「パッケージング・タスク」セクションで、「ユーザー・データのスクラブ」ヘッダー下にある「コンテンツ置換えのためのファイルのコピー・イン」をクリックします。

   ファイルのコピー・イン・ページが表示されます。コピーしたファイルが表示されます。
7. コピー・インするファイルを選択して「OK」をクリックします。

   ファイルがパッケージにコピー・インされて、既存のファイルが上書きされます。「パッケージのカスタマイズ」ページに戻ると、コピー・インされたファイルをリストした確認メッセージが表示されます。

親トピック: インシデント・パッケージの表示と変更

9.4.3.4 インシデント・パッケージへの外部ファイルの追加

インシデント・パッケージには、任意のタイプの外部ファイルを追加できます。

外部ファイルをインシデント・パッケージに追加するには:

1. 対象のインシデント・パッケージの「パッケージのカスタマイズ」ページにアクセスします。

   手順については、「「パッケージのカスタマイズ」ページへのアクセス」を参照してください。
2. 「ファイル」リンクをクリックして「ファイル」サブページを表示します。

   このページでは、パッケージに対してファイルを追加または削除できます。
3. 「外部ファイルの追加」をクリックします。

   外部ファイルの追加ページが表示されます。これには選択したファイルからホスト名が表示されます。
4. 次のいずれかを実行して、追加するファイルを指定します。
   - 「ファイル名」フィールドに、ファイルへのフルパスを入力します。
   - 「ファイル名」フィールドの横にある拡大鏡アイコンをクリックして、次のステップを実行します。
     a. ホスト資格証明がプロンプトで要求された場合は、外部ファイルが存在するホストの資格証明を入力して、「OK」をクリックします。(「優先資格証明として保存」を選択すると次回から資格証明を要求するプロンプトが表示されなくなります)。
     b. 「参照および選択: ファイルまたはディレクトリ」ウィンドウで必要なファイルを選択し、「選択」をクリックします。

        「参照および選択」ウィンドウがクローズし、選択したファイルへのパスが「外部ファイルの追加」ページの「ファイル名」フィールドに表示されます。
5. 「OK」をクリックします。

「パッケージのカスタマイズ」ページに戻ると、「ファイル」サブページが表示されます。これで、選択したファイルがファイル・リストに表示されます。

親トピック: インシデント・パッケージの表示と変更

9.4.3.5 インシデント・パッケージ・ファイルの削除

インシデント・パッケージから任意のタイプのファイルを1つ以上削除できます。

インシデント・パッケージ・ファイルを削除するには:

1. 対象のインシデント・パッケージの「パッケージのカスタマイズ」ページにアクセスします。

   手順については、「「パッケージのカスタマイズ」ページへのアクセス」を参照してください。

2. 「ファイル」リンクをクリックして「ファイル」サブページを表示します。

   パッケージに含まれているファイルのリストが表示されます。

   パッケージの物理ファイルをまだ生成していない場合、すべてのパッケージ・ファイルがリストに表示されます。すでに物理ファイルを生成している場合、ファイル・リストの上部に「表示」リストが表示されます。ここで、増分パッケージのコンテンツのみを表示するか、パッケージの内容すべてを表示するかを選択できます。デフォルトではパッケージの内容の増分を表示します。このデフォルト設定では、パッケージの物理ファイルが最後に生成されたとき以降に作成または変更されたパッケージ・ファイルのみが表示されます。すべてのパッケージ・ファイルを表示するには、「表示」リストから「全パッケージ・コンテンツ」を選択します。

3. 削除するファイルを選択して「除外」をクリックします。

> ノート:
>
> 対象のファイルが見つからない場合は、別のページにある場合があります。「次」リンクをクリックして、次のページを表示します。対象のファイルがみつかるまで、「次」をクリックし続けるか、ファイル番号のリスト(「次」リンクの左側)から番号を選択します。ファイルを選択して「削除」をクリックします。

親トピック: インシデント・パッケージの表示と変更

9.4.3.6 インシデント・パッケージのアクティビティ・ログの表示と更新

サポート・ワークベンチでは、インシデント・パッケージごとにアクティビティ・ログが保持されます。

パッケージに対して実行したほとんどのアクティビティ(ファイルの追加や削除、パッケージのZIPファイルの作成など)はログに記録されます。独自のノートをログに追加することもできます。特に、これは、複数のデータベース管理者がパッケージを使用している場合に役立ちます。

インシデント・パッケージのアクティビティ・ログを表示および更新するには:

1. 対象のインシデント・パッケージの「パッケージの詳細」ページにアクセスします。

   手順については、「パッケージの詳細の表示」を参照してください。

2. 「アクティビティ・ログ」リンクをクリックして、「アクティビティ・ログ」サブページを表示します。

   アクティビティ・ログが表示されます。

3. 自分のコメントをアクティビティ・ログに追加するには、「コメント」フィールドにテキストを入力して、「コメントの追加」をク

リックします。

コメントがリストに追加されます。

親トピック: インシデント・パッケージの表示と変更

9.4.4 相関パッケージの作成、編集およびアップロード

Oracleサポート・サービスにパッケージをアップロードした後、1つ以上の相関パッケージを作成およびアップロードできます。

データベース・ホームページの「関連アラート」セクションにクリティカル・アラートが表示されている場合は上記をお薦めします。相関パッケージは、メイン・パッケージと呼ばれる元のパッケージと関連付けられています。メイン・パッケージには、データベース・インスタンス内で発生した問題が含まれます。相関パッケージには、他のインスタンス(Oracle ASMインスタンスやその他のデータベース・インスタンス)で発生した問題や、メイン・パッケージにある問題に関連する問題が含まれます。相関パッケージは、関連するインスタンスにそれぞれ1つのみ存在できます。

相関パッケージを作成、編集およびアップロードするには:

1. メイン・パッケージの「パッケージの詳細」ページを表示します。

   手順については、「パッケージの詳細の表示」を参照してください。

2. 「パッケージの詳細」ページ上で「パッケージのカスタマイズ」をクリックします。
3. 「パッケージのカスタマイズ」ページの「パッケージング・タスク」セクションの「追加の診断データ」で、「相関パッケージの作成/更新」をクリックします。
4. 「相関パッケージ」ページの「相関パッケージ」で、インシデントのあるインスタンスを1つ以上選択して「作成」をクリックします。

   確認メッセージが表示され、新規に作成された相関パッケージのパッケージIDが「ID」列に表示されます。

5. 相関パッケージを作成したインスタンスを選択し、「コンテンツ準備の終了」をクリックします。

   確認メッセージが表示されます。

6. (オプション)次のステップで相関パッケージを表示および編集します。

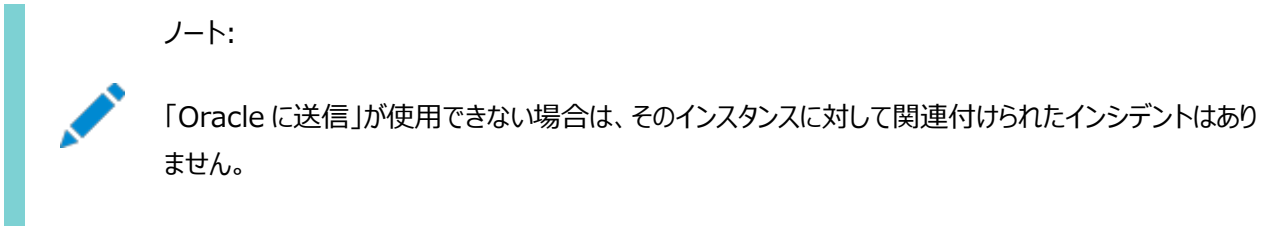
   a. パッケージIDをクリックしてパッケージを表示します。

      資格証明を要求された場合は、「ログイン」をクリックします。

   b. 「パッケージの詳細」ページで「ファイル」をクリックしてパッケージ内のファイルを表示します。

   c. 「パッケージのカスタマイズ」をクリックして、「インシデント・パッケージの表示と変更」に記載されている目的のカスタマイズ・タスクを実行します。

7. アップロードするそれぞれの相関パッケージについて、「アップロード・ファイルの生成」をクリックします。
8. オラクル社に送信するそれぞれの相関パッケージについて、「Oracleに送信」をクリックします。



ノート:

「Oracle に送信」が使用できない場合は、そのインスタンスに対して関連付けられたインシデントはありません。

関連項目:

- 「相関パッケージについて」
- 「トポロジ全体に関連する問題」

親トピック: 問題の報告

9.4.5 相関パッケージの削除

相関パッケージは、そのパッケージを作成したターゲットのサポート・ワークベンチで削除します。

たとえば、Oracle ASMインスタンス・ターゲットに対して相関パッケージを作成した場合は、そのOracle ASMインスタンスのサポート・ワークベンチにアクセスします。

相関パッケージを削除するには:

1. 相関パッケージを作成したターゲットのサポート・ワークベンチにアクセスします。

   ヒント:

   サポート・ワークベンチのページ下部の「関連リンク」セクションを参照してください。または、「サポート・ワークベンチを使用した問題の表示」を参照してください

2. 「パッケージ」をクリックして「パッケージ」のサブページを表示します。
3. リストから、相関パッケージを検索します。パッケージの説明を表示し、相関パッケージであることを確認します。
4. パッケージを選択して「削除」をクリックします。
5. 確認ページで「はい」をクリックします。

関連項目:

- 「相関パッケージについて」
- 「トポロジ全体に関連する問題」

親トピック: 問題の報告

9.4.6 インシデント・パッケージのプリファレンスの設定

インシデント・パッケージのプリファレンスを設定できます。インシデント・パッケージのプリファレンスの例には、インシデント情報を保持する日数や、各問題のパッケージに含める最初と最後のインシデント数などがあります。

デフォルトでは、問題のインシデントが多数存在する場合、最初の3つと最後の3つのインシデントのみがパッケージ化されます。これらまたはその他のインシデント・パッケージのプリファレンスは、Cloud ControlまたはADRCIユーティリティを使用して変更できます。

Cloud Controlを使用してインシデント・パッケージのプリファレンスを設定するには:

1. 「サポート・ワークベンチ」ホームページにアクセスします。

   手順については、「サポート・ワークベンチを使用した問題の表示」を参照してください。

2. ページの下部にある「関連リンク」セクションで、「インシデント・パッケージング構成」をクリックします。

   「インシデント・パッケージング構成の表示」ページが表示されます。「ヘルプ」をクリックして、このページの設定の説明を表示します。

3. 「編集」をクリックします。

   「インシデント・パッケージング構成の編集」ページが表示されます。

4. 設定を編集し、「OK」をクリックして変更を適用します。

関連項目:

- 「インシデント・パッケージについて」
- 「インシデントおよび問題について」
- 「タスク5: 診断データのパッケージ化とOracleサポートへのアップロード」
- ADRCIの詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: 問題の報告

9.5 問題の解決

この項では、SQL修復アドバイザやデータ・リカバリ・アドバイザなどのアドバイザ・ツール、およびリソース・マネージャおよび関連APIなどのリソース管理ツールを使用してデータベースの問題を解決する方法について説明します。

- SQL修復アドバイザを使用したSQLエラーの修復
  まれなケースで、SQL文がクリティカル・エラーで失敗した場合は、SQL修復アドバイザを実行して失敗した文の修復を試みることができます。
- データ・リカバリ・アドバイザを使用したデータ破損の修復
  データ・ブロックの破損、UNDO破損、データ・ディクショナリの破損などを修復するには、データ・リカバリ・アドバイザを使用します。
- 過剰なシステム・リソースを使用するSQL文の実行計画の隔離
  Oracle Database 19c以上では、SQL隔離インフラストラクチャ(SQL隔離)を使用して、リソース・マネージャによって終了されOracleデータベース内の過剰なシステム・リソースを使用するSQL文の実行計画を隔離できます。個々のSQL文には複数の実行計画がある場合があり、隔離された実行計画を使用しようとするとそのSQL文は実行できなくなるため、データベースのパフォーマンスの低下を防ぐことができます。

親トピック: 問題の診断と解決

9.5.1 SQL修復アドバイザを使用したSQLエラーの修復

ほとんど発生しませんが、SQL文がクリティカル・エラーで失敗した場合は、SQL修復アドバイザを実行して失敗した文を修復できます。

- SQL修復アドバイザについて
  SQL文が重大なエラーで失敗した後に、SQL修復アドバイザを実行します。
- Cloud Controlを使用したSQL修復アドバイザの実行
  Cloud Controlのサポート・ワークベンチの「問題の詳細」ページからSQL修復アドバイザを実行できます。
- DBMS_SQLDIAGパッケージのサブプログラムを使用したSQL修復アドバイザの実行

DBMS_SQLDIAGパッケージのサブプログラムを使用してSQL修復アドバイザを実行できます。

- Cloud Controlを使用したSQLパッチの表示、無効化または削除
  SQL修復アドバイザでSQLパッチを適用した後、Cloud Controlを使用して、それを表示してその存在を確認することや、無効化または削除することもできます。パッチを無効化または削除する理由の1つに、後続リリースのOracle Databaseをインストールすることで、パッチ適用済SQL文のエラーの原因となっていた不具合が修正されている場合があります。
- DBMS_SQLDIAGパッケージのサブプログラムを使用したSQLパッチの無効化または削除
  SQL修復アドバイザでSQLパッチを適用した後、DBMS_SQLDIAGパッケージのサブプログラムを使用してそれを無効化または削除できます。パッチを無効化または削除する理由の1つに、後続リリースのOracle Databaseをインストールすることで、パッチ適用済SQL文のエラーの原因となっていた不具合が修正されている場合があります。
- DBMS_SQLDIAGパッケージのサブプログラムを使用したパッチのエクスポートとインポート
  SQL修復アドバイザを使用して作成されたパッチは、DBMS_SQLDIAGパッケージのサブプログラムを使用して、あるシステムからエクスポートし別のシステムにインポートできます。

親トピック: 問題の解決

9.5.1.1 SQL修復アドバイザについて

SQL修復アドバイザは、重大なエラーが発生してSQL文が失敗した場合に実行します。

このアドバイザによって、文が分析され、多くの場合、文を修復するためにパッチの適用が推奨されます。リコメンデーションを実装すると、適用されたSQLパッチによって、問合せオプティマイザで将来実行する場合の代替実行計画が選択され、失敗が回避されます。

Cloud Control、またはDBMS_SQLDIAGパッケージのサブプログラムを使用して、SQL修復アドバイザを実行できます。

親トピック: SQL修復アドバイザを使用したSQLエラーの修復

9.5.1.2 Cloud Controlを使用したSQL修復アドバイザの実行

Cloud Controlのサポート・ワークベンチの「問題の詳細」ページからSQL修復アドバイザを実行できます。

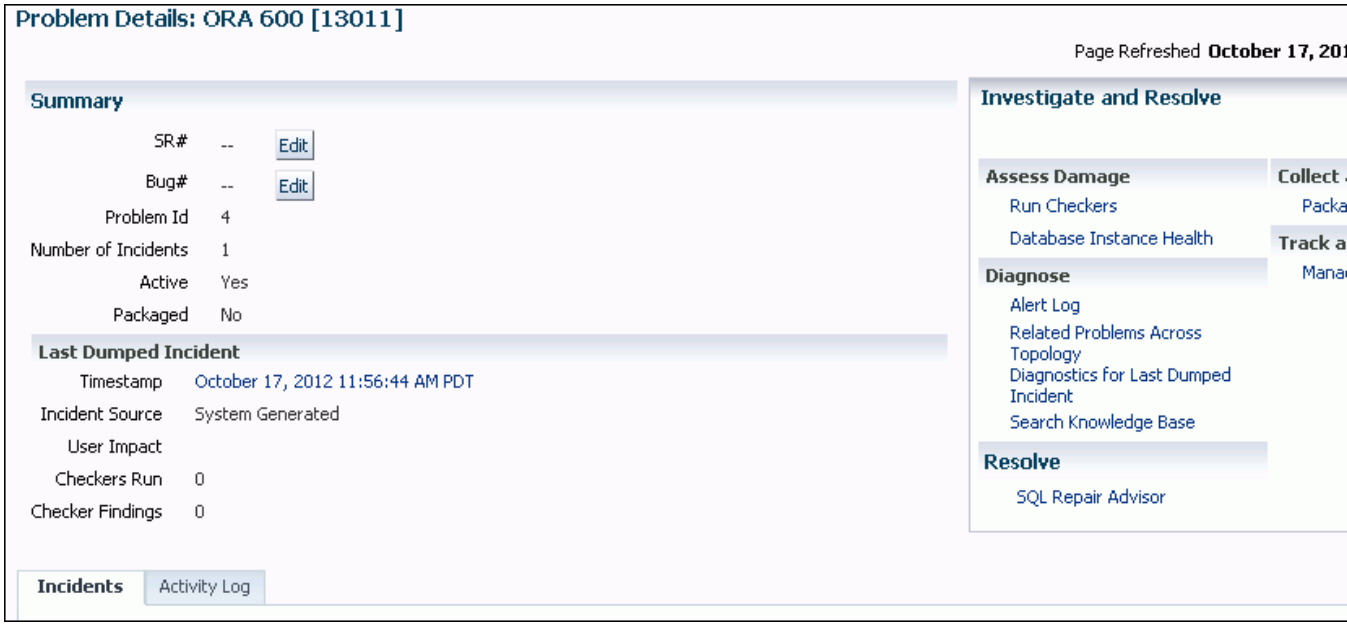
通常、これを実行するのは、SQL文に起因するクリティカル・エラーの通知をすでに受け取り、「問題の調査、報告および解決について」で説明したワークフローに従っている場合です。

Cloud Controlを使用してSQL修復アドバイザを実行するには:

1. 失敗したSQL文に関連する問題の「問題の詳細」ページにアクセスします。

   手順については、「サポート・ワークベンチを使用した問題の表示」を参照してください。

2. 「調査と解決」セクションの「解決」ヘッダーで、「SQL修復アドバイザ」をクリックします。

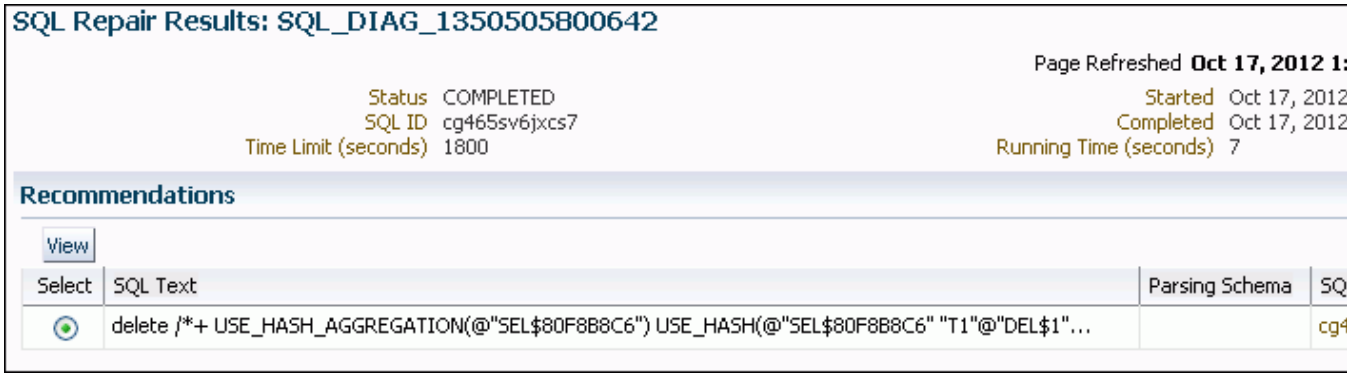


3. 「SQL修復アドバイザ」ページで次のステップを実行します。

   a. 必要な場合は現在のタスク名を変更し、必要に応じてタスクの説明を入力し、アドバイザ・タスクのオプションの時間制限を変更またはクリアし、設定を調整してアドバイザを即時に実行するか、後の日時に実行するかをスケジュールします。

   b. 「送信」をクリックします

「処理中」ページが表示されます。その少し後に「SQL修復結果」ページが表示されます。



「SQLパッチ」列のチェック・マークは、推奨事項があることを示します。この列にチェック・マークがない場合は、SQL修復アドバイザがSQL文に対してパッチを提示できなかったことを意味します。

ノート:

「SQL 修復結果」ページが表示されない場合は、表示するために次のステップを実行します。

   a. データベース・ホームページに移動します。

   b. 「パフォーマンス」メニューから、「アドバイザ・ホーム」を選択します。

   c. 「アドバイザ・セントラル」ページの「結果」リストで、SQL 修復アドバイザの最新エントリを探しま

す。

d. そのエントリを選択して「結果の表示」をクリックします。

4. 推奨事項がある場合は、「SQLパッチ」列にチェック・マークが表示されます。推奨事項を参照するには、「表示」をクリックします。

   「修復の推奨」ページに、文に対する推奨パッチが表示されます。

5. 「実装」をクリックします。

   「SQL修復結果」ページに戻り、確認メッセージが表示されます。

6. (オプション)「SQLワークシートを使用して検証」をクリックして、SQLワークシートの文を実行し、パッチによって文が正常に修復したことを検証します。

親トピック: SQL修復アドバイザを使用したSQLエラーの修復

9.5.1.3 DBMS_SQLDIAGパッケージのサブプログラムを使用したSQL修復アドバイザの実行

DBMS_SQLDIAGパッケージのサブプログラムを使用してSQL修復アドバイザを実行できます。

通常、これを実行するのは、SQL文に起因するクリティカル・エラーの通知を受け取り、「問題の調査、報告および解決について」で説明したワークフローに従っている場合です。

SQL修復アドバイザを実行するには、DBMS_SQLDIAGパッケージのサブプログラムであるCREATE_DIAGNOSIS_TASKを使用して診断タスクを作成し、EXECUTE_DIAGNOSIS_TASKを使用してその診断タスクを実行します。SQL修復アドバイザは、最初に重大なエラーを再現し、次にACCEPT_SQL_PATCHサブプログラムを使用して適用できるSQLパッチの形式での対処方法の生成を試みます。

ノート:

Oracle Database 19c 以上では、単一のサブプログラム SQL_DIAGNOSE_AND_REPAIR を使用することで、診断タスクの作成、診断タスクの実行、および指定された SQL 文の SQL パッチ推奨の受入もできます。したがって、SQL_DIAGNOSE_AND_REPAIR サブプログラムは、CREATE_DIAGNOSIS_TASK、EXECUTE_DIAGNOSIS_TASK および ACCEPT_SQL_PATCH サブプログラムの機能をすべて実行できます。

DBMS_SQLDIAGパッケージのサブプログラムを使用してSQL修復アドバイザを実行するには:

1. 問題が発生したSQL文の特定

   重大なエラーが発生した次のSQL文について考えてみます。

```
DELETE FROM t t1
WHERE t1.a = 'a' AND
      ROWID <> (SELECT MAX(ROWID)
                FROM t t2
                WHERE t1.a = t2.a AND
                      t1.b = t2.b AND
                      t1.d = t2.d)
```

   SQL修復アドバイザを使用して、この重大なエラーを修復します。

2. 診断タスクの作成

DBMS_SQLDIAG.CREATE_DIAGNOSIS_TASKを実行します。オプションのタスク名、アドバイザ・タスクのオプションの時間制限、および問題のタイプを指定できます。次の例では、SQLテキストを指定し、タスク名をerror_task、問題のタイプをDBMS_SQLDIAG.PROBLEM_TYPE_COMPILATION_ERRORと指定しています。

```
DECLARE
  rep_out   CLOB;
  t_id      VARCHAR2(50);
BEGIN
  t_id := DBMS_SQLDIAG.CREATE_DIAGNOSIS_TASK (
          sql_text => 'DELETE FROM t t1
                       WHERE t1.a = ''a'' AND
                             ROWID <> (SELECT MAX(ROWID)
                                       FROM t t2
                                       WHERE t1.a = t2.a AND
                                             t1.b = t2.b AND
                                             t1.d = t2.d)',
          task_name => 'error_task',
          problem_type => DBMS_SQLDIAG.PROBLEM_TYPE_COMPILATION_ERROR);
```

3. 診断タスクの実行

SQL修復アドバイザの対処方法生成および分析フェーズを実行するには、CREATE_DIAGNOSIS_TASKで戻されたタスクIDを使用して、DBMS_SQLDIAG.EXECUTE_DIAGNOSIS_TASKを実行します。多少の遅延後に、SQL修復アドバイザに戻ります。SQL修復アドバイザは、その実行の一環として検索結果の記録を残し、この記録には、SQL修復アドバイザのレポート機能を使用してアクセス可能です。

```
DBMS_SQLDIAG.EXECUTE_DIAGNOSIS_TASK (t_id);
```

4. 診断タスクのレポートの生成

診断タスクの分析には、DBMS_SQLDIAG.REPORT_DIAGNOSIS_TASKを使用してアクセスします。SQL修復アドバイザは、対処方法を検出できた場合、SQLパッチを推奨します。SQLパッチは、SQLプロファイルと類似していますが、SQLプロファイルとは異なり、コンパイル・エラーまたは実行エラーに対処するために使用します。

```
rep_out := DBMS_SQLDIAG.REPORT_DIAGNOSIS_TASK (t_id, DBMS_SQLDIAG.TYPE_TEXT);
DBMS_OUTPUT.PUT_LINE ('Report : ' ||  rep_out);
END;
/
```

5. パッチの適用

レポートでパッチが推奨されている場合は、DBMS_SQLDIAG.ACCEPT_SQL_PATCHを実行してパッチを受け入れることができます。このプロシージャは、引数としてタスク名を使用します。

```
EXECUTE DBMS_SQLDIAG.ACCEPT_SQL_PATCH(task_name => 'error_task', task_owner =>
'SYS', replace => TRUE);
```

6. パッチのテスト

パッチを受け入れたため、SQL文を再実行できます。今回は重大なエラーが発生しなくなります。この文でEXPLAIN PLANを実行すると、計画を生成するためにSQLパッチが使用されたことが示されます。

```
DELETE FROM t t1
WHERE t1.a = 'a' AND
      ROWID <> (SELECT max(rowid)
                FROM t t2
                WHERE t1.a = t2.a AND
                      t1.b = t2.b AND
                      t1.d = t2.d);
```

関連項目:

DBMS_SQLDIAGパッケージのサブプログラムの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: SQL修復アドバイザを使用したSQLエラーの修復

9.5.1.4 Cloud Controlを使用したSQLパッチの表示、無効化または削除

SQL修復アドバイザでSQLパッチを適用した後、Cloud Controlを使用して、それを表示してその存在を確認することや、無効化または削除することもできます。パッチを無効化または削除する理由の1つに、後続リリースのOracle Databaseをインストールすることで、パッチ適用済SQL文のエラーの原因となっていた不具合が修正されている場合があります。

Cloud Controlを使用してSQLパッチを表示、無効化または削除するには:

1. Cloud Controlのデータベース・ホームページにアクセスします。
2. 「パフォーマンス」メニューから「SQL」を選択し、次に、「SQL計画管理」を選択します。

   「SQL計画管理」ページが表示されます。
3. 「SQLパッチ」をクリックして、「SQLパッチ」サブページを表示します。

   「SQLパッチ」サブページに、データベース内のすべてのSQLパッチが表示されます。
4. 関連するSQLテキストを検証して、特定のパッチを検索します。

   SQLテキストをクリックして、その文の完全なテキストを表示します。SQLテキストを表示した後、「戻る」をクリックします。
5. 「SQLパッチ」サブページでパッチを無効化するには、パッチを選択して「無効化」をクリックします。

   確認メッセージが表示され、パッチのステータスがDISABLEDに変更されます。後でパッチを再び有効にするには、パッチを選択して「有効化」をクリックします。
6. パッチを削除するには、パッチを選択して「削除」をクリックします。

   確認メッセージが表示されます。

関連項目:

「SQL修復アドバイザについて」

親トピック: SQL修復アドバイザを使用したSQLエラーの修復

9.5.1.5 DBMS_SQLDIAGパッケージのサブプログラムを使用したSQLパッチの無効化または削除

SQL修復アドバイザでSQLパッチを適用した後、DBMS_SQLDIAGパッケージのサブプログラムを使用してそれを無効化または削除できます。パッチを無効化または削除する理由の1つに、後続リリースのOracle Databaseをインストールすることで、パッチ適用済SQL文のエラーの原因となっていた不具合が修正されている場合があります。

DBMS_SQLDIAGパッケージのサブプログラムを使用してSQLパッチを無効化するには:

ステータス値DISABLEDによって無効にするパッチの名前を指定して、プロシージャDBMS_SQLDIAG.ALTER_SQL_PATCHを実行します。

次の例では、SQLパッチsql_patch_12345を無効にします。

```
EXEC DBMS_SQLDIAG.ALTER_SQL_PATCH('sql_patch_12345', 'STATUS', 'DISABLED');
```

DBMS_SQLDIAGパッケージのサブプログラムを使用してSQLパッチを削除するには:

削除するパッチの名前を指定して、プロシージャDBMS_SQLDIAG.DROP_SQL_PATCHを実行します。パッチ名は、EXPLAIN PLANセクションから取得することも、ビューDBA_SQL_PATCHESを問い合せて取得することもできます。

次の例では、SQLパッチsql_patch_12345を削除します。

```
EXEC DBMS_SQLDIAG.DROP_SQL_PATCH('sql_patch_12345');
```

関連項目:

- 「SQL修復アドバイザについて」
- DBMS_SQLDIAGパッケージのサブプログラムの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: SQL修復アドバイザを使用したSQLエラーの修復

9.5.1.6 DBMS_SQLDIAGパッケージのサブプログラムを使用したパッチのエクスポートとインポート

SQL修復アドバイザを使用して作成されたパッチは、DBMS_SQLDIAGパッケージのサブプログラムを使用して、あるシステムからエクスポートし別のシステムにインポートできます。

パッチは、ステージング表を使用して、あるシステムからエクスポートして別のシステムにインポートできます。SQL診断セットの場合と同様に、ステージング表に挿入する操作はパックと呼ばれ、ステージング表のデータからパッケージを作成する操作はアンパックと呼ばれます。

DBMS_SQLDIAGパッケージのサブプログラムを使用してパッチをエクスポートおよびインポートするには:

1. DBMS_SQLDIAG.CREATE_STGTAB_SQLPATCHをコールして、ユーザーSHが所有するステージング表を作成します。

   ```
   EXEC DBMS_SQLDIAG.CREATE_STGTAB_SQLPATCH(
       table_name          =>  'STAGING_TABLE',
       schema_name         =>  'SH');
   ```

2. DBMS_SQLDIAG.PACK_STGTAB_SQLPATCHを1回以上コールして、ステージング表にSQLパッチのデータを書き込みます。この場合、現行のスキーマ所有者が所有するステージング表に、DEFAULTカテゴリ内のすべてのSQLパッチのデータをコピーします。

   ```
   EXEC DBMS_SQLDIAG.PACK_STGTAB_SQLPATCH(
       staging_table_name  =>  'STAGING_TABLE');
   ```

3. この場合、現行のスキーマ所有者が所有するステージング表には、1つのSQLパッチSP_FIND_EMPLOYEEのみがコピーされます。

   ```
   EXEC DBMS_SQLDIAG.PACK_STGTAB_SQLPATCH(
       patch_name          =>  'SP_FIND_EMPLOYEE',
       staging_table_name  =>  'STAGING_TABLE');
   ```

   その後、データポンプ、インポート・コマンドとエクスポート・コマンド、またはデータベース・リンクのいずれかを使用して、ステージング表を別のシステムに移動できます。

4. DBMS_SQLDIAG.UNPACK_STGTAB_SQLPATCHをコールして、ステージング表内のパッチ・データから、新しいシス

テムにSQLパッチを作成します。この場合、ステージング表に格納されているSP_FIND_EMPLOYEEパッチのデータ内の名前を'SP_FIND_EMP_PROD'に変更します。

```
exec dbms_sqldiag.remap_stgtab_sqlpatch(
   old_patch_name      =>  'SP_FIND_EMPLOYEE',
   new_patch_name      =>  'SP_FIND_EMP_PROD',
```

関連項目:

DBMS_SQLDIAGパッケージのサブプログラムの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: SQL修復アドバイザを使用したSQLエラーの修復

9.5.2 データ・リカバリ・アドバイザを使用したデータ破損の修復

データ・ブロックの破損、UNDO破損、データ・ディクショナリの破損などを修復するには、データ・リカバリ・アドバイザを使用します。

データ・リカバリ・アドバイザはEnterprise Managerサポート・ワークベンチ(サポート・ワークベンチ)、状態モニターおよびRMANユーティリティに統合され、データ破損の問題の表示、各問題の程度の評価(クリティカル、高優先度、低優先度)、問題の影響の説明、修復オプションの推奨、顧客が選択したオプションの実行可能性チェック、および修復プロセスの自動化を実行します。

データ・リカバリ・アドバイザの使用方法の詳細は、Cloud Controlのオンライン・ヘルプを参照してください。この項では、サポート・ワークベンチからアドバイザにアクセスする方法について説明します。

データ・リカバリ・アドバイザは、データ破損またはその他のデータ障害に関連するヘルス・チェックの結果を表示する場合にサポート・ワークベンチによって自動的に推奨され、サポート・ワークベンチからアクセスできます。データ・リカバリ・アドバイザは「アドバイザ・セントラル」ページからも使用できます。

Cloud Controlでデータ・リカバリ・アドバイザにアクセスするには:

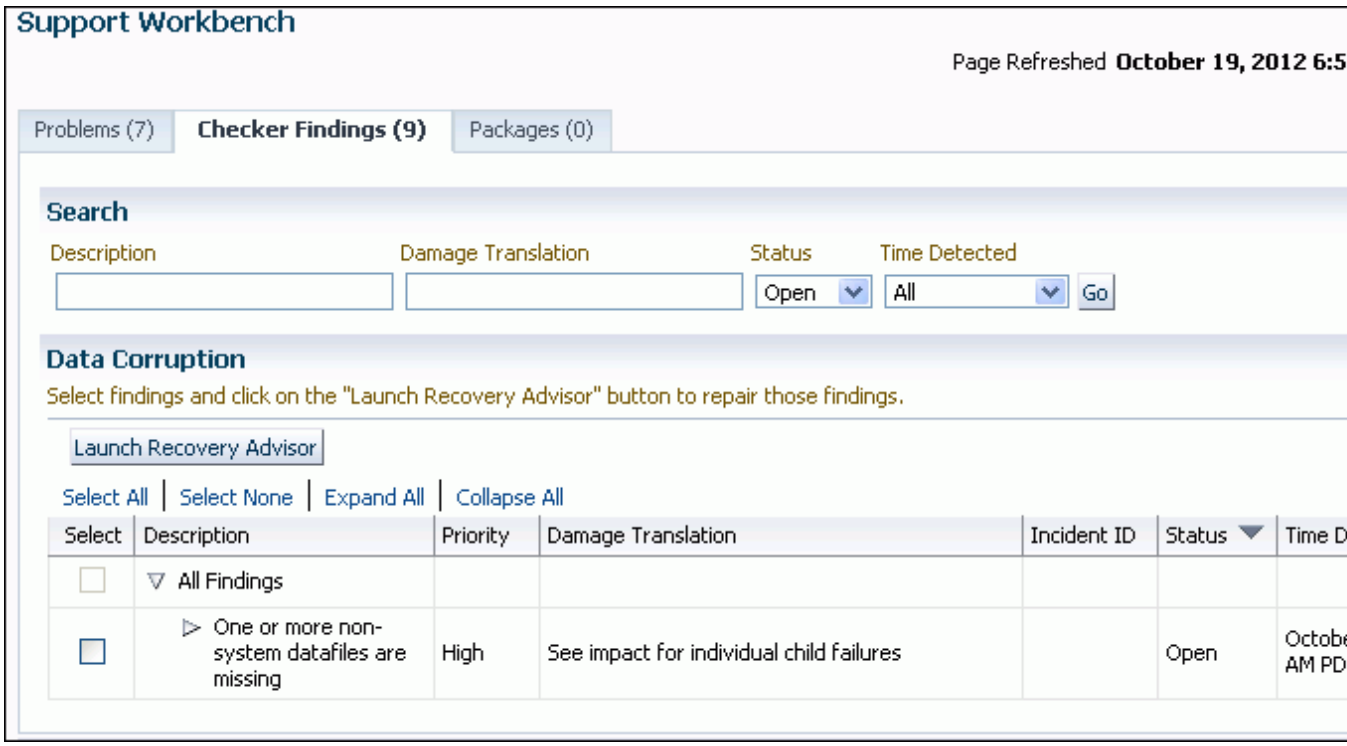
1. Cloud Controlのデータベース・ホームページにアクセスします。

   データ・リカバリ・アドバイザを使用できるのは、SYSDBAとして接続している場合のみです。

2. 「Oracle Database」メニューから、「診断」を選択し、次に、「サポート・ワークベンチ」を選択します。

3. 「チェッカ結果」をクリックします。

   チェッカ結果サブページが表示されます。



4. 1つ以上のデータ破損結果を選択して、「リカバリ・アドバイザの起動」をクリックします。

関連項目:

データ・リカバリ・アドバイザの詳細は、『Oracle Database 2日でデータベース管理者』および『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: 問題の解決

9.5.3 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

Oracle Database 19c以上では、SQL隔離インフラストラクチャ(SQL隔離)を使用して、リソース・マネージャによって終了されOracleデータベース内の過剰なシステム・リソースを使用するSQL文の実行計画を隔離できます。個々のSQL文には複数の実行計画がある場合があり、隔離された実行計画を使用しようとするとそのSQL文は実行できなくなるため、データベースのパフォーマンスの低下を防ぐことができます。

- SQL文の実行計画の隔離について
  SQL隔離インフラストラクチャ(SQL隔離)を使用して、リソース・マネージャによって終了されOracleデータベース内の過剰なシステム・リソースを使用するSQL文の実行計画を隔離できます。SQL文の隔離された実行計画は再実行できなくなるため、データベースのパフォーマンスの低下を防ぐことができます。
- SQL文の実行計画に対する隔離構成の作成
  これらのDBMS_SQLQパッケージ・ファンクション(CREATE_QUARANTINE_BY_SQL_IDまたはCREATE_QUARANTINE_BY_SQL_TEXT)のいずれかを使用して、SQL文の実行計画について隔離構成を作成できます。
- 隔離構成での隔離しきい値の指定
  SQL文の実行計画について隔離構成を作成した後、DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用して、隔離しきい値を指定できます。いずれかのリソース・マネージャしきい値が、SQL文の隔離構成で指定された隔離しきい値以下である場合、その隔離構成で指定された実行計画を使用すると、SQL文は実行できなくなります。

- 隔離構成の有効化と無効化
  DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用して隔離構成を有効化または無効化できます。隔離構成は、作成時にデフォルトで有効になります。
- 隔離構成の詳細の表示
  DBA_SQL_QUARANTINEビューを問い合せることで、すべての隔離構成の詳細を取得できます。
- 隔離構成の削除
  使用されていない隔離構成は、53週間後に自動的にパージまたは削除されます。
  DBMS_SQLQ.DROP_QUARANTINEプロシージャを使用して隔離構成を削除することもできます。
  DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用して、隔離構成の自動削除を無効にできます。
- SQL文の隔離された実行計画の詳細の表示
  V$SQLビューとGV$SQLビューを問い合せることで、SQL文の隔離された実行計画について詳細を取得できます。
- あるデータベースから別のデータベースへの隔離構成の転送
  DBMS_SQLQパッケージのサブプログラム(CREATE_STGTAB_QUARANTINE、PACK_STGTAB_QUARANTINEおよびUNPACK_STGTAB_QUARANTINE)を使用して、あるデータベースから別のデータベースに隔離構成を転送できます。
- 例: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離
  この例では、リソース・マネージャを使用して構成されたリソース使用制限を超えた場合に、SQL文の実行計画がどのように隔離されるかを示します。

親トピック: 問題の解決

9.5.3.1 SQL文の実行計画の隔離について

SQL隔離インフラストラクチャ(SQL隔離)を使用して、リソース・マネージャによって終了されOracleデータベース内の過剰なシステム・リソースを使用するSQL文の実行計画を隔離できます。SQL文の隔離された実行計画は再実行できなくなるため、データベースのパフォーマンスの低下を防ぐことができます。

リソース・マネージャを使用して、SQL文に対してシステム・リソース消費の上限(リソース・マネージャしきい値)を構成できます。リソース・マネージャは、リソース・マネージャしきい値を超えるSQL文を終了します。以前のOracle Databaseリリースでは、リソース・マネージャによって終了されたSQL文は、再実行された場合、リソース・マネージャによってその再実行が許可され、リソース・マネージャしきい値を超えると再度終了されます。したがって、このようなSQL文を再実行できると、システム・リソースの浪費になります。

Oracle Database 19c以上では、SQL隔離を使用して、リソース・マネージャによって終了されたSQL文の実行計画を、再実行できないように自動的に隔離できます。SQLの隔離情報は、データ・ディクショナリに定期的に永続化されます。リソース・マネージャがSQL文を終了すると、文が隔離されるまでに数分かかる場合があります。

ノート:

各種エディションおよびサービスでサポートされる機能の詳細は、『Oracle Database ライセンス情報ユーザー・マニュアル』を参照

さらに、SQL隔離を使用すると、DBMS_SQLQパッケージのサブプログラムを使用して様々なシステム・リソースの消費に対するしきい値(リソース・マネージャしきい値と同様)を指定することにより、SQL文の実行計画の隔離構成も作成できます。これらのしきい値は、隔離しきい値と呼ばれます。いずれかのリソース・マネージャしきい値が、SQL文の隔離構成で指定された隔離しきい値以下である場合、その隔離構成で指定された実行計画を使用すると、SQL文は実行できなくなります。

次のステップに従って、DBMS_SQLQパッケージのサブプログラムを使用してSQL文の実行計画について隔離しきい値を手動で

設定します。

1. SQL文の実行計画について隔離構成を作成します
2. 隔離構成に隔離しきい値を指定します

DBMS_SQLQパッケージのサブプログラムを使用して、隔離構成に関連する次の操作を実行することもできます。

- 隔離構成の有効化または無効化
- 隔離構成の削除
- あるデータベースから別のデータベースへの隔離構成の転送

ノート:

- 隔離構成は、SQL 文の実行計画に固有です。2 つの異なる SQL 文が同じ実行計画を使用する場合は、同じ隔離構成を共有しません。
- 実行計画は、リソース・マネージャによって終了される固有の SQL 文に対して隔離されます。したがって、SQL 文に対して隔離された実行計画は、リソース・マネージャによってまだ終了されていない別の SQL 文に対して隔離されません。
- SQL 文の実行計画について隔離構成が作成されていない場合、または隔離構成に隔離しきい値が指定されていない場合、SQL 文の実行計画は、リソース・マネージャしきい値のいずれかを超えたことでリソース・マネージャによって終了されると自動的に隔離されます。

たとえば、SQL文の実行時間を10秒(リソース・マネージャしきい値)に制限するリソース・マネージャのリソース・プランを考えてみます。SQL文Q1は、このリソース・プランがアプリケーションであるとします。Q1の実行時間が10秒を超えると、リソース・マネージャによって終了されます。次に、SQL隔離によって、実行計画に固有のQ1の隔離構成が作成され、実行時間10秒が隔離しきい値として隔離構成に保存されます。

同じ実行計画を使用してQ1が再度実行され、リソース・マネージャしきい値が10秒である場合、Q1の実行には10秒以上かかるため、SQL隔離では、隔離しきい値10秒を参照し、Q1が最終的にリソース・マネージャによって終了されると判断して、Q1の実行を許可しません。

リソース・マネージャしきい値が5秒に変更され、Q1が同じ実行計画を使用して再度実行される場合、Q1の実行には10秒以上かかるため、SQL隔離では、隔離しきい値10秒を参照し、Q1が最終的にリソース・マネージャによって終了されると判断して、Q1の実行を許可しません。

リソース・マネージャしきい値が15秒に変更され、Q1が同じ実行計画を使用して再度実行される場合、SQL隔離では、隔離しきい値10秒を参照し、Q1の実行には10秒以上かかると判断しますが、Q1の実行が15秒以内に完了する可能性があるため、Q1の実行を許可します。

ノート:

隔離しきい値は SQL 文の実行計画に固有で、SQL 文とその実行計画が超えるリソース・マネージャしきい値に基づいて SQL 隔離によって自動的に設定されます。DBMS_SQLQ パッケージのサブプログラムを使用して、SQL 文の特定の実行計画について隔離しきい値を手動で設定することもできます。

関連項目:

- SQL文の実行計画に対する隔離構成の作成
- 隔離構成での隔離しきい値の指定
- 隔離構成の有効化と無効化
- 隔離構成の詳細の表示
- 隔離構成の削除
- リソース・マネージャを使用してSQL文のリソース使用制限を構成する方法の詳細は、リソース制限の設定による自動切替えの指定を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.2 SQL文の実行計画に対する隔離構成の作成

DBMS_SQLQパッケージ・ファンクション(CREATE_QUARANTINE_BY_SQL_IDまたはCREATE_QUARANTINE_BY_SQL_TEXT)のいずれかを使用して、SQL文の実行計画について隔離構成を作成できます。

次の例では、SQL文(SQL IDが8vu7s907prbgr)の実行計画(ハッシュ値が3488063716)の隔離構成を作成します。

```
DECLARE
    quarantine_config VARCHAR2(30);
BEGIN
    quarantine_config := DBMS_SQLQ.CREATE_QUARANTINE_BY_SQL_ID(
                            SQL_ID => '8vu7s907prbgr',
                            PLAN_HASH_VALUE => '3488063716');
END;
/
```

実行計画を指定しないか、それをNULLとして指定すると、固有の隔離構成がすでに作成されている実行計画を除き、SQL文のすべての実行計画に隔離構成が適用されます。

次の例では、SQL文(SQL IDが152sukb473gsk)のすべての実行計画について隔離構成を作成します。

```
DECLARE
    quarantine_config VARCHAR2(30);
BEGIN
    quarantine_config := DBMS_SQLQ.CREATE_QUARANTINE_BY_SQL_ID(
                            SQL_ID => '152sukb473gsk');
END;
/
```

次の例では、SQL文'select count(*) from emp'のすべての実行計画について隔離構成を作成します。

```
DECLARE
    quarantine_config VARCHAR2(30);
BEGIN
    quarantine_config := DBMS_SQLQ.CREATE_QUARANTINE_BY_SQL_TEXT(
                            SQL_TEXT => to_clob('select count(*) from emp'));
END;
/
```

CREATE_QUARANTINE_BY_SQL_IDファンクションとCREATE_QUARANTINE_BY_SQL_TEXTファンクションでは、隔離構成の名前が返されます。これは、DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用してSQL文の実行計画について隔離しきい値を指定するために使用できます。

関連項目:

- 隔離構成での隔離しきい値の指定
- DBMS_SQLQパッケージのサブプログラムの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.3 隔離構成での隔離しきい値の指定

SQL文の実行計画について隔離構成を作成した後、DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用して、隔離しきい値を指定できます。いずれかのリソース・マネージャしきい値が、SQL文の隔離構成で指定された隔離しきい値以下である場合、その隔離構成で指定された実行計画を使用すると、SQL文は実行できなくなります。

DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用して、次のリソースの隔離しきい値を隔離構成に指定できます。

- CPU時間
- 経過時間
- I/O (MB単位)
- 物理I/O要求の数
- 論理I/O要求の数

次の例では、隔離構成SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4に対して、CPU時間に指定された隔離しきい値が5秒で、経過時間が10秒です。

```
BEGIN
    DBMS_SQLQ.ALTER_QUARANTINE(
       QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
       PARAMETER_NAME  => 'CPU_TIME',
       PARAMETER_VALUE => '5');
    DBMS_SQLQ.ALTER_QUARANTINE(
       QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
       PARAMETER_NAME  => 'ELAPSED_TIME',
       PARAMETER_VALUE => '10');
END;
/
```

この隔離構成で指定された実行計画を使用してSQL文が実行され、CPU時間のリソース・マネージャしきい値が5秒以下または経過時間が10秒以下である場合、SQL文は実行できなくなります。

ノート:

いずれかのリソース・マネージャしきい値が、SQL 文の隔離構成で指定された隔離しきい値以下である場合、その隔離構成で指定された実行計画を使用すると、SQL 文は実行できなくなります。

隔離構成の隔離しきい値の問合せ

DBMS_SQLQ.GET_PARAM_VALUE_QUARANTINEファンクションを使用して隔離構成の隔離しきい値を問い合せることができます。次の例では、隔離構成SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4のCPU時間消費について隔離しきい値が返されます。

```
DECLARE
    quarantine_config_setting_value VARCHAR2(30);
BEGIN
    quarantine_config_setting_value :=
        DBMS_SQLQ.GET_PARAM_VALUE_QUARANTINE(
              QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
              PARAMETER_NAME  => 'CPU_TIME');
END;
/
```

隔離構成からの隔離しきい値の削除

PARAMETER_VALUEの値としてDBMS_SQLQ.DROP_THRESHOLDを指定することで、隔離構成から隔離しきい値を削除できます。次の例では、隔離構成SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4からCPU時間消費の隔離しきい値を削除します。

```
BEGIN
    DBMS_SQLQ.ALTER_QUARANTINE(
       QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
       PARAMETER_NAME  => 'CPU_TIME',
       PARAMETER_VALUE => DBMS_SQLQ.DROP_THRESHOLD);
END;
/
```

関連項目:

DBMS_SQLQ.ALTER_QUARANTINEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.4 隔離構成の有効化と無効化

DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用して、隔離構成を有効または無効にできます。隔離構成は、作成時にデフォルトで有効になります。

次の例では、SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4という名前の隔離構成を無効にします。

```
BEGIN
    DBMS_SQLQ.ALTER_QUARANTINE(
       QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
       PARAMETER_NAME  => 'ENABLED',
       PARAMETER_VALUE => 'NO');
END;
/
```

次の例では、SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4という名前の隔離構成を有効にします。

```
BEGIN
    DBMS_SQLQ.ALTER_QUARANTINE(
       QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
       PARAMETER_NAME  => 'ENABLED',
       PARAMETER_VALUE => 'YES');
END;
/
```

関連項目:

DBMS_SQLQ.ALTER_QUARANTINEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.5 隔離構成の詳細の表示

DBA_SQL_QUARANTINEビューを問い合せることで、すべての隔離構成の詳細を取得できます。

DBA_SQL_QUARANTINEビューには、各隔離構成に関する次の情報が表示されます。

- 隔離構成名
- 隔離構成が適用されるSQL文
- 隔離構成が適用される実行計画のハッシュ値
- 隔離構成のステータス(有効または無効)
- 隔離構成の自動パージのステータス(はい、またはいいえ)
- 隔離構成で指定されている隔離しきい値:
    - CPU時間
    - 経過時間
    - I/O (MB単位)
    - 物理I/O要求の数
    - 論理I/O要求の数
- 隔離構成が作成された日時
- 隔離構成が最後に実行された日時

関連項目:

DBA_SQL_QUARANTINEビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.6 隔離構成の削除

使用されていない隔離構成は、53週間後に自動的にパージまたは削除されます。DBMS_SQLQ.DROP_QUARANTINEプロシージャを使用して隔離構成を削除することもできます。DBMS_SQLQ.ALTER_QUARANTINEプロシージャを使用して、隔離構成の自動削除を無効にできます。

次の例では、隔離構成SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4の自動削除を無効にします。

```
BEGIN
    DBMS_SQLQ.ALTER_QUARANTINE(
       QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
       PARAMETER_NAME  => 'AUTOPURGE',
       PARAMETER_VALUE => 'NO');
END;
/
```

次の例では、隔離構成SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4の自動削除を有効にします。

```
BEGIN
    DBMS_SQLQ.ALTER_QUARANTINE(
       QUARANTINE_NAME => 'SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4',
       PARAMETER_NAME  => 'AUTOPURGE',
       PARAMETER_VALUE => 'YES');
END;
```

```
/
```

次の例では、隔離構成SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4を削除します。

```
BEGIN
    DBMS_SQLQ.DROP_QUARANTINE('SQL_QUARANTINE_3z0mwuq3aqsm8cfe7a0e4');
END;
/
```

関連項目:

DBMS_SQLQパッケージのサブプログラムの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.7 SQL文の隔離された実行計画の詳細の表示

V$SQLビューとGV$SQLビューを問い合せることで、SQL文の隔離された実行計画について詳細を取得できます。

V$SQLビューとGV$SQLビューの次の列には、SQL文の実行計画の隔離情報が示されます。

- SQL_QUARANTINE: この列には、SQL文の実行計画に対する隔離構成の名前が示されます。
- AVOIDED_EXECUTIONS: この列には、実行計画の隔離後にそのSQL文の実行計画の実行を防いだ回数が示されます。

関連項目:

V$SQLビューとGV$SQLビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.8 あるデータベースから別のデータベースへの隔離構成の転送

DBMS_SQLQパッケージのサブプログラム(CREATE_STGTAB_QUARANTINE、PACK_STGTAB_QUARANTINEおよびUNPACK_STGTAB_QUARANTINE)を使用して、あるデータベースから別のデータベースに隔離構成を転送できます。

たとえば、テスト・データベースで隔離構成をテストし、それらが適切に機能したことを確認したとします。その後、これらの隔離構成を本番データベースにロードできます。

次の例では、DBMS_SQLQパッケージのサブプログラムを使用してあるデータベース(ソース・データベース)から別のデータベース(宛先データベース)に隔離構成を転送するステップを説明します。

1. SQL*Plusを使用して、管理権限があるユーザーとしてソース・データベースに接続し、DBMS_SQLQ.CREATE_STGTAB_QUARANTINEプロシージャでステージング表を作成します。

   次の例では、TBL_STG_QUARANTINEというステージング表を作成します。

   ```
   BEGIN
     DBMS_SQLQ.CREATE_STGTAB_QUARANTINE (
       staging_table_name => 'TBL_STG_QUARANTINE');
   END;
   /
   ```

2. 宛先データベースに転送する隔離構成をステージング表に追加します。

次の例では、QUARANTINE_CONFIG_という名前で始まるすべての隔離構成をステージング表TBL_STG_QUARANTINEに追加します。

```
DECLARE
  quarantine_configs NUMBER;
BEGIN
  quarantine_configs := DBMS_SQLQ.PACK_STGTAB_QUARANTINE(
                            staging_table_name => 'TBL_STG_QUARANTINE',
                            name => 'QUARANTINE_CONFIG_%');
END;
/
```

DBMS_SQLQ.PACK_STGTAB_QUARANTINEファンクションでは、ステージング表に追加された隔離構成の数が返されます。

3. Oracle Data Pump Exportユーティリティを使用して、ステージング表TBL_STG_QUARANTINEをダンプ・ファイルにエクスポートします。
4. ソース・データベース・システムから宛先データベース・システムにダンプ・ファイルを転送します。
5. 宛先データベース・システムで、Oracle Data Pump Importユーティリティを使用して、ステージング表TBL_STG_QUARANTINEをダンプ・ファイルから宛先データベースにインポートします。
6. SQL*Plusを使用して、管理権限があるユーザーとして宛先データベースに接続し、インポートしたステージング表から隔離構成を作成します。

   次の例では、インポートしたステージング表TBL_STG_QUARANTINEに格納されているすべての隔離構成に基づいて、宛先データベースに隔離構成を作成します。

```
DECLARE
  quarantine_configs NUMBER;
BEGIN
  quarantine_configs := DBMS_SQLQ.UNPACK_STGTAB_QUARANTINE(
                            staging_table_name => 'TBL_STG_QUARANTINE');
END;
/
```

DBMS_SQLQ.UNPACK_STGTAB_QUARANTINEファンクションでは、宛先データベースで作成された隔離構成の数が返されます。

関連項目:

DBMS_SQLQパッケージのサブプログラムの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

9.5.3.9 例: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

この例では、リソース・マネージャを使用して構成されたリソース使用制限を超えた場合に、SQL文の実行計画がどのように隔離されるかを示します。

1. リソース・マネージャを使用して、ユーザーHRによって実行されるSQL文の実行時間制限(3秒)を指定します。

   次のコードは、DBMS_RESOURCE_MANAGERパッケージのサブプログラムを使用して複雑なリソース・プランを作成することによって、次の操作を実行します。

- コンシューマ・グループTEST_RUNAWAY_GROUPを作成します。
- ユーザーHRをTEST_RUNAWAY_GROUPコンシューマ・グループに割り当てます。
- 実行時間が3秒を超えるとSQL文を終了するリソース・プランLIMIT_RESOURCEを作成します。
- LIMIT_RESOURCEリソース・プランをTEST_RUNAWAY_GROUPコンシューマ・グループに割り当てます。

```
connect / as sysdba
begin
  -- Create a pending area
  dbms_resource_manager.create_pending_area();
  -- Create a consumer group 'TEST_RUNAWAY_GROUP'
  dbms_resource_manager.create_consumer_group (
    consumer_group => 'TEST_RUNAWAY_GROUP',
    comment        => 'This consumer group limits execution time for SQL
statements'
  );
  -- Map the sessions of the user 'HR' to the consumer group
'TEST_RUNAWAY_GROUP'
  dbms_resource_manager.set_consumer_group_mapping(
    attribute      => DBMS_RESOURCE_MANAGER.ORACLE_USER,
    value          => 'HR',
    consumer_group => 'TEST_RUNAWAY_GROUP'
  );
  -- Create a resource plan 'LIMIT_RESOURCE'
  dbms_resource_manager.create_plan(
    plan    => 'LIMIT_RESOURCE',
    comment => 'Terminate SQL statements after exceeding total execution time'
  );
  -- Create a resource plan directive by assigning the 'LIMIT_RESOURCE' plan to
  -- the 'TEST_RUNAWAY_GROUP' consumer group
  -- Specify the execution time limit of 3 seconds for SQL statements belonging
to
  -- the 'TEST_RUNAWAY_GROUP' group
  dbms_resource_manager.create_plan_directive(
    plan             => 'LIMIT_RESOURCE',
    group_or_subplan => 'TEST_RUNAWAY_GROUP',
    comment          => 'Terminate SQL statements when they exceed the' ||
                        'execution time of 3 seconds',
    switch_group     => 'CANCEL_SQL',
    switch_time      => 3,
    switch_estimate  => false
  );
  -- Allocate resources to the sessions not covered by the currently active
plan
  -- according to the OTHER_GROUPS directive
  dbms_resource_Manager.create_plan_directive(
    plan              => 'LIMIT_RESOURCE',
    group_or_subplan  => 'OTHER_GROUPS',
    comment           => 'Ignore'
  );
  -- Validate and submit the pending area
  dbms_resource_manager.validate_pending_area();
  dbms_resource_manager.submit_pending_area();
  -- Grant switch privilege to the 'HR' user to switch to the
'TEST_RUNAWAY_GROUP'
  -- consumer group
  dbms_resource_manager_privs.grant_switch_consumer_group('HR',
                                                          'TEST_RUNAWAY_GROUP',
                                                          false);

  -- Set the initial consumer group of the 'HR' user to 'TEST_RUNAWAY_GROUP'
  dbms_resource_manager.set_initial_consumer_group('HR',
                                                   'TEST_RUNAWAY_GROUP');
end;
/
-- Set the 'LIMIT_RESOURCE' plan as the top plan for the Resource Manager
```

```
alter system set RESOURCE_MANAGER_PLAN = 'LIMIT_RESOURCE' scope = memory;
-- Unlock the HR user and assign it the DBA role
alter user hr identified by hr_user_password account unlock;
grant dba to hr;
-- Flush the shared pool
alter system flush shared_pool;
```

2. HRユーザーとしてOracleデータベースに接続し、実行時間制限の3秒を超えるSQL文を実行します。

```
select count(*)
from employees emp1, employees emp2,
     employees emp3, employees emp4,
     employees emp5, employees emp6,
     employees emp7, employees emp8,
     employees emp9, employees emp10
where rownum <= 100000000;
```

SQL文は実行時間制限の3秒を超えるとリソース・マネージャによって終了され、次のエラー・メッセージが表示されます。

```
ORA-00040: active time limit exceeded - call aborted
```

SQL文の実行計画が隔離リストに追加されるため、再度実行できなくなります。

3. SQL文を再度実行します。

SQL文の実行計画が隔離されているため、次のエラー・メッセージが表示されてSQL文は即時に終了します。

```
ORA-56955: quarantined plan used
```

4. v$sqlビューとdba_sql_quarantineビューを問い合せて、SQL文の隔離された実行計画の詳細を表示します。

- v$sqlビューを問い合せます。v$sqlビューには、SQL文の各種統計(隔離統計を含む)に関する情報が表示されます。

```
select sql_text, plan_hash_value, avoided_executions, sql_quarantine
from v$sql
where sql_quarantine is not null;
```

この問合せの出力は次のようになります。

```
SQL_TEXT                                 PLAN_HASH_VALUE
AVOIDED_EXECUTIONS   SQL_QUARANTINE
-----------------------------------   --------------   ---------------
--   ---------------------------------
select count(*)                          3719017987        1
SQL_QUARANTINE_3uuhv1u5day0yf6ed7f0c
from employees emp1, employees emp2,
     employees emp3, employees emp4,
     employees emp5, employees emp6,
     employees emp7, employees emp8,
     employees emp9, employees emp10
where rownum <= 100000000;
```

sql_quarantine列には、SQL文の実行計画に対する隔離構成の自動生成名が示されます。

- dba_sql_quarantineビューを問い合せます。dba_sql_quarantineビューには、SQL文の実行計画の隔離構成に関する情報が表示されます。

```
select sql_text, name, plan_hash_value, last_executed, enabled
from dba_sql_quarantine;
```

この問合せの出力は次のようになります。

```
SQL_TEXT                                 NAME
PLAN_HASH_VALUE    LAST_EXECUTED                  ENABLED
-----------------------------------    ---------------------------------
--    ---------------    ----------------------------    -------
select count(*)
SQL_QUARANTINE_3uuhv1u5day0yf6ed7f0c    3719017987         14-JAN-19
02.19.01.000000 AM    YES
from employees emp1, employees emp2,
     employees emp3, employees emp4,
     employees emp5, employees emp6,
     employees emp7, employees emp8,
     employees emp9, employees emp10
where rownum <= 100000000;
```

name列には、SQL文の実行計画に対する隔離構成の自動生成名が示されます。

5. 例の環境をクリーン・アップします。

次のコードは、この例で作成されたすべてのデータベース・オブジェクトを削除します。

```
connect / as sysdba
begin
  for quarantineObj in (select name from dba_sql_quarantine) loop
    sys.dbms_sqlq.drop_quarantine(quarantineObj.name);
  end loop;
end;
/
alter system set RESOURCE_MANAGER_PLAN = '' scope = memory;
execute dbms_resource_manager.clear_pending_area();
execute dbms_resource_manager.create_pending_area();
execute dbms_resource_manager.delete_plan('LIMIT_RESOURCE');
execute dbms_resource_manager.delete_consumer_group('TEST_RUNAWAY_GROUP');
execute dbms_resource_manager.validate_pending_area();
execute dbms_resource_manager.submit_pending_area();
```

関連項目:

- DBMS_RESOURCE_MANAGERパッケージのサブプログラムを使用した複雑なリソース・プランの作成の詳細は、複雑なリソース・プランの作成を参照してください。
- DBMS_RESOURCE_MANAGERパッケージのサブプログラムの詳細は、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください。
- V$SQLビューの詳細は、Oracle Databaseリファレンスを参照してください。

親トピック: 過剰なシステム・リソースを使用するSQL文の実行計画の隔離

第II部 Oracle Databaseの構造と記憶域

データベース構造とストレージ・コンポーネントを作成および管理できます。

- 制御ファイルの管理
  制御ファイルを作成、バックアップおよび削除できます。
- REDOログの管理
  REDOログの管理では、REDOログ・グループとメンバーの作成、REDOログ・メンバーの再配置と名前変更、REDOログ・グループとメンバーの削除、およびログ切替えの強制などのタスクを行います。
- アーカイブREDOログ・ファイルの管理
  アーカイブREDOログ・ファイルの管理では、NOARCHIVELOGまたはARCHIVELOGモードの選択、およびアーカイブ先の指定などのタスクを行います。
- 表領域の管理
  表領域はデータベース記憶域の単位で、関連する論理構造をグループ化します。データベースのデータファイルは表領域に格納されます。
- データファイルおよび一時ファイルの管理
  データファイルおよび一時ファイルの管理では、ファイルの作成、変更および削除などのタスクを行います。
- データのトランスポート
  データのトランスポートにより、1つのデータベースから他へデータを移動します。
- UNDOの管理
  デフォルトのインストールでは、Oracle DatabaseによってUNDOが自動的に管理されます。通常、DBAによる操作は不要です。ただし、インストールでOracle Flashback操作を使用する場合は、それらの操作が正常に終了するように、UNDO管理タスクをいくつか実行することが必要な場合があります。
- Oracle Managed Filesの使用
  Oracle Databaseは、データベースを構成するファイルを管理できます。

10 制御ファイルの管理

制御ファイルを作成、バックアップおよび削除できます。

- 制御ファイルの概要
  制御ファイルはデータベースの物理構造を記録した小さなバイナリ・ファイルであり、すべてのOracle Databaseに含まれています。
- 制御ファイルのガイドライン
  ガイドラインに従って、データベースの制御ファイルを管理できます。
- 制御ファイルの作成
  制御ファイルを作成、コピー、名前変更および再配置できます。
- 制御ファイル作成後のトラブルシューティング
  CREATE CONTROLFILE文を実行した後で、エラーが発生する場合があります。
- 制御ファイルのバックアップ
  制御ファイルをバックアップするには、ALTER DATABASE BACKUP CONTROLFILE文を使用します。
- 現行のコピーを使用した制御ファイルのリカバリ
  現行のバックアップまたは多重化コピーから制御ファイルをリカバリできます。
- 制御ファイルの削除
  制御ファイルは削除できますが、データベースには常に少なくとも2つの制御ファイルが存在する必要があります。
- 制御ファイルのデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューのセットに対して制御ファイルに関する情報を問い合せることができます。

関連項目:

- 制御ファイルの概要は、『Oracle Database概要』を参照してください。
- Oracle Databaseサーバーによって作成および管理される制御ファイルの作成方法は、「Oracle Managed Filesの使用」を参照してください

親トピック: Oracle Databaseの構造と記憶域

10.1 制御ファイルの概要

制御ファイルはデータベースの物理構造を記録した小さなバイナリ・ファイルであり、すべてのOracle Databaseに含まれています。

制御ファイルには、次の情報が格納されています。

- データベース名
- 対応するデータファイルとREDOログ・ファイルの名前と位置
- データベース作成のタイムスタンプ
- 現行のログ順序番号
- チェックポイント情報

制御ファイルは、データベースがオープンしているときに必ずOracle Databaseサーバーが書き込めるように、使用可能にしてお

く必要があります。制御ファイルがないと、データベースがマウントできず、リカバリが困難になります。

Oracle Databaseの制御ファイルはデータベースとともに作成されます。デフォルトでは、データベースの作成時に、制御ファイルのコピーが少なくとも1つ作成されます。デフォルトで複数のコピーが作成されるオペレーティング・システムもあります。データベース作成時に、制御ファイルのコピーを2つ以上作成することをお薦めします。その後も、制御ファイルを失ったり、制御ファイル内の設定を変更する場合には、制御ファイルを作成できます。

親トピック: 制御ファイルの管理

10.2 制御ファイルのガイドライン

ガイドラインに従って、データベースの制御ファイルを管理できます。

- 制御ファイルのファイル名の指定
  データベース初期化パラメータ・ファイルのCONTROL_FILES初期化パラメータを使用して、制御ファイル名を指定します。インスタンスは起動時にすべてのリストされたファイルを認識して開き、データベースの動作中はインスタンスがすべてのリストされた制御ファイルに書き込み、ファイルを維持します。
- 異なるディスク上での制御ファイルの多重化
  Oracle Databaseには少なくとも2つの制御ファイルを作成し、各ファイルを異なる物理ディスクに配置するようにします。
- 制御ファイルのバックアップ
  制御ファイルのバックアップは非常に重要です。初期設定時およびデータベースの物理構造を変更したときは、必ずバックアップを作成してください。
- 制御ファイルのサイズ管理
  制御ファイルのサイズを決定する主な要因は、対応するデータベースを作成したCREATE DATABASE文のMAXDATAFILES、MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORYおよびMAXINSTANCESパラメータに設定された値です。

親トピック: 制御ファイルの管理

10.2.1 制御ファイルのファイル名の指定

データベース初期化パラメータ・ファイルのCONTROL_FILES初期化パラメータを使用して、制御ファイル名を指定します。インスタンスは起動時にすべてのリストされたファイルを認識して開き、データベースの動作中はインスタンスがすべてのリストされた制御ファイルに書き込み、ファイルを維持します。

データベースの作成前にCONTROL_FILESに対してファイルを指定しない場合は、次のように処理されます。

- Oracle Managed Filesを使用していない場合、データベースではデフォルトのファイル名で制御ファイルが作成されます。デフォルトのファイル名はオペレーティング・システムによって異なります。
- Oracle Managed Filesを使用している場合は、この機能を使用可能にするために設定した初期化パラメータによって制御ファイルの名前と位置が決定されます。
- Oracle Automatic Storage Management (Oracle ASM)を使用している場合は、不完全なOracle ASMファイル名をDB_CREATE_FILE_DESTおよびDB_RECOVERY_FILE_DEST初期化パラメータに設定できます。Oracle ASMによって、制御ファイルが適切な場所に自動的に作成されます。

関連トピック

- 初期制御ファイルの作成

- Oracle Managed Filesの使用
- Oracle Automatic Storage Management管理者ガイド

親トピック: 制御ファイルのガイドライン

10.2.2 異なるディスク上での制御ファイルの多重化

Oracle Databaseには少なくとも2つの制御ファイルを作成し、各ファイルを異なる物理ディスクに配置するようにします。

ディスク障害によって制御ファイルが破損した場合は、対応するインスタンスを必ず停止します。ディスク・ドライブを修復後、他のディスク上にある制御ファイルの正常なコピーを使用して破損した制御ファイルをリストアすると、インスタンスを再起動できます。この場合は、メディア・リカバリは不要です。

多重制御ファイルは、次のように動作します。

- データベースは、データベース初期化パラメータ・ファイルの初期化パラメータCONTROL_FILESにリストされているすべてのファイル名に対して、情報を書き込みます。
- データベースの稼働中に、CONTROL_FILESパラメータにリストされている最初のファイルのみデータベースによって読み込まれます。
- データベースの稼働中に制御ファイルのいずれかが使用できなくなった場合、インスタンスは動作不能になり、終了します。

ノート:

データベースには最低 2 つ以上の制御ファイルを作成し、それらを異なる物理ディスク上に配置することをお薦めします。

制御ファイルを多重化する方法の1つは、REDOログが多重化されている場合、REDOログ・グループのメンバーが格納されているすべてのディスク・ドライブに制御ファイルのコピーを格納することです。このようにファイルを配置することによって、単一のディスク障害のために制御ファイルとREDOログ・グループがすべて失われる危険が少なくなります。

親トピック: 制御ファイルのガイドライン

10.2.3 制御ファイルのバックアップ

制御ファイルのバックアップは非常に重要です。初期設定時およびデータベースの物理構造を変更したときは、必ずバックアップを作成してください。

たとえば、次のような構造上の変更を行った場合は、バックアップを作成する必要があります。

- データファイルの追加、削除または名前変更
- 表領域の追加または削除、表領域の読取り/書込み状態の変更
- REDOログ・ファイルまたはREDOログ・グループの追加と削除

制御ファイルのバックアップ方法については、「制御ファイルのバックアップ」を参照してください。

親トピック: 制御ファイルのガイドライン

10.2.4 制御ファイルのサイズ管理

制御ファイルのサイズを決定する主な要因は、対応するデータベースを作成したCREATE DATABASE文のMAXDATAFILES、MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORYおよびMAXINSTANCESパラメータに設定された値です。

これらのパラメータの値を大きくすると、対応するデータベースの制御ファイルのサイズも大きくなります。

関連項目:

- 制御ファイルの最大サイズの詳細は、使用しているオペレーティング・システム固有のOracleマニュアルを参照してください。
- CREATE DATABASE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 制御ファイルのガイドライン

10.3 制御ファイルの作成

制御ファイルを作成、コピー、名前変更および再配置できます。

- 初期制御ファイルの作成
  Oracle Databaseの初期制御ファイルは、CREATE DATABASE文を発行したときに作成されます。
- 制御ファイルの追加コピーの作成、名前変更および再配置
  多重化のために制御ファイルのコピーを追加作成するには、既存の制御ファイルを新しい位置にコピーし、そのファイル名を制御ファイルのリストに追加します。
- 新しい制御ファイルの作成
  データベースのすべての制御ファイルが恒久的な損傷を受け、制御ファイルのバックアップがない場合、またはデータベース名を変更する場合は、新規制御ファイルを作成できます。

親トピック: 制御ファイルの管理

10.3.1 初期制御ファイルの作成

Oracle Databaseの初期制御ファイルは、CREATE DATABASE文を発行したときに作成されます。

制御ファイルの名前は、データベースの作成時に使用される初期化パラメータ・ファイルのCONTROL_FILESパラメータで指定します。CONTROL_FILESで指定するファイル名はパスを含めて完全に指定する必要があり、オペレーティング・システムによって異なります。CONTROL_FILES初期化パラメータの例を次に示します。

```
CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
                 /u02/oracle/prod/control02.ctl,
                 /u03/oracle/prod/control03.ctl)
```

指定した名前のファイルがデータベースの作成時に存在している場合、CREATE DATABASE文でCONTROLFILE REUSE句を指定しないとエラーが発生します。また、古い制御ファイルのサイズが新しい制御ファイルのSIZEパラメータと異なる場合、REUSE句は使用できません。

制御ファイルのサイズはOracle Databaseの一部のリリースでは異なり、制御ファイルで指定されるファイルの数が変更されるタイミングもリリースによって異なります。MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILESおよびMAXINSTANCESなどの構成パラメータは制御ファイルのサイズに影響します。

CONTROL_FILES初期化パラメータの値を後で変更して、制御ファイルを追加したり、既存の制御ファイルの名前や位置を変

更できます。

関連項目:

制御ファイルの指定方法の詳細は、使用しているオペレーティング・システム固有のOracleマニュアルを参照してください。

親トピック: 制御ファイルの作成

10.3.2 制御ファイルの追加コピーの作成、名前変更および再配置

多重化のために制御ファイルのコピーを追加作成するには、既存の制御ファイルを新しい位置にコピーし、そのファイル名を制御ファイルのリストに追加します。

同様に、制御ファイルの名前を変更するには、既存の制御ファイルを新しい名前や位置にコピーし、制御ファイル・リストのファイル名を変更します。どちらの場合も、作業中に制御ファイルが変更されないように、制御ファイルをコピーする前にデータベースを停止してください。

現行の制御ファイルの多重化コピーを追加、または制御ファイル名を変更するには:

1. データベースを停止します。
2. オペレーティング・システムのコマンドを使用して、既存の制御ファイルを新しい位置にコピーします。
3. データベース初期化パラメータ・ファイルのCONTROL_FILESパラメータを編集して、新しい制御ファイル名を追加するか、または既存の制御ファイル名を変更します。
4. データベースを再起動します。

親トピック: 制御ファイルの作成

10.3.3 新しい制御ファイルの作成

データベースのすべての制御ファイルが恒久的な損傷を受け、制御ファイルのバックアップがない場合、またはデータベース名を変更する場合は、新規制御ファイルを作成できます。

- 新しい制御ファイルを作成する場合
  特定の状況では新規制御ファイルを作成する必要があります。
- CREATE CONTROLFILE文
  CREATE CONTROLFILE文を使用して、データベースの新しい制御ファイルを作成できます。
- 新しい制御ファイルの作成
  データベースの新しい制御ファイルを作成できます。

親トピック: 制御ファイルの作成

10.3.3.1 新しい制御ファイルを作成する場合

特定の状況では新規制御ファイルを作成する必要があります。

次の状況の場合に、新しい制御ファイルを作成する必要があります。

- データベースの制御ファイルがすべて破損し、制御ファイルのバックアップがない場合。
- データベース名を変更する場合。

  たとえば、分散環境でデータベース名が別のデータベース名と競合する場合は、データベース名を変更します。

ノート:

データベース名と DBID(内部データベース識別子)は、DBNEWID ユーティリティを使用して変更できます。このユーティリティの使用の詳細は、『Oracle Database ユーティリティ』を参照してください。

親トピック: 新しい制御ファイルの作成

10.3.3.2 CREATE CONTROLFILE文

CREATE CONTROLFILE文を使用して、データベースの新しい制御ファイルを作成できます。

次の文は、prodデータベース(これまで別のデータベース名を使用していたデータベース)に対する新しい制御ファイルを作成します。

```
CREATE CONTROLFILE
   SET DATABASE prod
   LOGFILE GROUP 1 ('/u01/oracle/prod/redo01_01.log',
                    '/u01/oracle/prod/redo01_02.log'),
           GROUP 2 ('/u01/oracle/prod/redo02_01.log',
                    '/u01/oracle/prod/redo02_02.log'),
           GROUP 3 ('/u01/oracle/prod/redo03_01.log',
                    '/u01/oracle/prod/redo03_02.log')
   RESETLOGS
   DATAFILE '/u01/oracle/prod/system01.dbf' SIZE 3M,
            '/u01/oracle/prod/rbs01.dbs' SIZE 5M,
            '/u01/oracle/prod/users01.dbs' SIZE 5M,
            '/u01/oracle/prod/temp01.dbs' SIZE 5M
   MAXLOGFILES 50
   MAXLOGMEMBERS 3
   MAXLOGHISTORY 400
   MAXDATAFILES 200
   MAXINSTANCES 6
   ARCHIVELOG;
```

ノート:

- CREATE CONTROLFILE 文は、指定したデータファイルと REDO ログ・ファイルを破損する可能性があります。ファイル名を省略すると、そのファイルのデータが失われたり、データベース全体にアクセスできなくなる場合があります。この文を発行するときには十分に注意し、必ず「新しい制御ファイルの作成」の手順に従ってください。
- 新しい制御ファイルを作成する前にデータベースの FORCE LOGGING を使用可能にしていて、この設定を引き続き有効にする場合は、CREATE CONTROLFILE 文で FORCE LOGGING 句を指定する必要があります。「FORCE LOGGING モードの指定」を参照してください。

関連項目:

CREATE CONTROLFILE文の詳細な構文は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: 新しい制御ファイルの作成

10.3.3.3 新しい制御ファイルの作成

データベースの新しい制御ファイルを作成できます。

新しい制御ファイルを作成するには、次のステップを実行します。

1. データベースのデータファイルとREDOログ・ファイルすべてのリストを作成します。

   「制御ファイルのバックアップ」に記載されている制御ファイルのバックアップの推奨事項に従っている場合は、現在のデータベース構造を反映するデータファイルとREDOログ・ファイルのリストがすでに作成されています。しかし、そのようなリストがない場合は、次の文を実行することでリストが生成されます。

   ```
   SELECT MEMBER FROM V$LOGFILE;
   SELECT NAME FROM V$DATAFILE;
   SELECT VALUE FROM V$PARAMETER WHERE NAME = 'control_files';
   ```

   このようなリストが手元になく、制御ファイルが破損してデータベースをオープンできない場合は、データベースを構成するデータファイルとREDOログ・ファイルをすべて特定してください。新しい制御ファイルを作成すると、ステップ5で指定したファイル以外はリカバリできなくなります。さらに、SYSTEM表領域を構成するファイルが1つでも欠落していると、データベースをリカバリできないことがあります。

2. データベースを停止します。

   データベースがオープンしている場合は、できるかぎり通常モードでデータベースを停止します。IMMEDIATE句またはABORT句は、他に方法がない場合にのみ使用してください。

3. データベースのすべてのデータファイルとREDOログ・ファイルをバックアップします。
4. 新しいインスタンスを起動します。ただし、データベースのマウントとオープンは行いません。

   ```
   STARTUP NOMOUNT
   ```

5. CREATE CONTROLFILE文を使用して、データベースの新しい制御ファイルを作成します。

   制御ファイルに加えて、REDOログ・グループも失ってしまった場合は、新しい制御ファイルの作成時にRESETLOGS句を指定します。この場合は、失ったREDOログをリカバリする必要があります(ステップ8)。データベースの名前を変更した場合は、必ずRESETLOGS句を指定してください。それ以外の場合は、NORESETLOGS句を選択してください。

6. 新しい制御ファイルのバックアップをオフラインの記憶デバイスに格納します。バックアップ作成の手順については、「制御ファイルのバックアップ」を参照してください。
7. データベースのCONTROL_FILES初期化パラメータを編集し、ステップ5で作成してデータベースの一部となったすべての制御ファイルを指定します(バックアップ制御ファイルは含めません)。データベースの名前を変更する場合、インスタンスのパラメータ・ファイルのDB_NAMEパラメータを編集して新しい名前を指定します。
8. 必要に応じて、データベースをリカバリします。データベースをリカバリしない場合は、ステップ9にスキップしてください。

   リカバリの一部として制御ファイルを作成している場合は、データベースをリカバリしてください。NORESETLOGS句を使用して新しい制御ファイルを作成した場合は、クローズ状態の完全なデータベース・リカバリ操作によってデータベースをリカバリできます。

   RESETLOGS句を使用して新しい制御ファイルを作成した場合は、USING BACKUP CONTROL FILEを指定する必要があります。オンラインREDOログ、アーカイブREDOログ・ファイルまたはデータファイルが失われた場合は、これらのファイルのリカバリ手順に従ってください。

関連項目:

データベースのリカバリ、および失われた制御ファイルのリカバリ方法については、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

9. 次のいずれかの方法で、データベースをオープンします。
   - リカバリを実行しなかった場合、またはステップ8でクローズ状態の完全なデータベース・リカバリを実行した場合は、通常のステップでデータベースをオープンします。

     ```
     ALTER DATABASE OPEN;
     ```

   - 制御ファイルの作成時にRESETLOGSを指定した場合は、ALTER DATABASE文でRESETLOGSを指定します。

     ```
     ALTER DATABASE OPEN RESETLOGS;
     ```

これでデータベースはオープンされ、使用可能になります。

親トピック: 新しい制御ファイルの作成

10.4 制御ファイル作成後のトラブルシューティング

CREATE CONTROLFILE文を実行した後で、エラーが発生する場合があります。

- 欠落したファイルや余分なファイルのチェック
  新しい制御ファイルを作成し、それを使用してデータベースをオープンした後、アラート・ログをチェックして、データ・ディクショナリと制御ファイルの間で不整合(データ・ディクショナリにはデータファイルがあるが、制御ファイルには含まれていないなど)が検出されていないかを確認してください。
- CREATE CONTROLFILEでのエラー処理
  新しい制御ファイルの作成後にデータベースをマウントおよびオープンしようとしたときにOracle Databaseによってエラーが送信された場合、最も可能性の高い原因は、CREATE CONTROLFILE文からファイルを省略したか、リストされていない必要のあるファイルが含まれていることです。

親トピック: 制御ファイルの管理

10.4.1 欠落したファイルや余分なファイルのチェック

新しい制御ファイルを作成し、それを使用してデータベースをオープンした後、アラート・ログをチェックして、データ・ディクショナリと制御ファイルの間で不整合(データ・ディクショナリにはデータファイルがあるが、制御ファイルには含まれていないなど)が検出されていないかを確認してください。

データ・ディクショナリ内にはデータファイルが存在していて、新しい制御ファイルには含まれていない場合、データベースは制御ファイル内にMISSINGnnnn (nnnnは10進数のファイル番号)という名前のプレースホルダ・エントリを作成します。制御ファイル内のMISSINGnnnnには、オフラインであること、およびメディア・リカバリを必要とすることを示すフラグが設定されます。

MISSINGnnnnに対応する実際のデータファイルが読取り専用または通常のオフラインである場合、MISSINGnnnnの名前を実際のデータファイルの名前に変更することによってデータファイルにアクセス可能になります。MISSINGnnnnに対応するデータファイルが読取り専用または通常のオフラインではない場合、データファイルはメディア・リカバリを必要としますが、RESETLOGSの結果これができないため、名前の変更操作によってデータファイルをアクセス可能にできません。この場合、データファイルを含む表領域を削除する必要があります。

反対に、制御ファイルに指定されているデータファイルがデータ・ディクショナリに存在しない場合、データベースは新しい制御ファイルからそのファイルへの参照を削除します。どちらの場合も、検出された状態を通知するメッセージがアラート・ログに書き込まれま

す。

親トピック: 制御ファイル作成後のトラブルシューティング

10.4.2 CREATE CONTROLFILEでのエラー処理

新しい制御ファイルの作成後にデータベースをマウントおよびオープンしようとしたときにOracle Databaseによってエラーが送信された場合、最も可能性の高い原因は、CREATE CONTROLFILE文からファイルを省略したか、リストされていない必要のあるファイルが含まれていることです。

通常、エラーはORA-01173、ORA-01176、ORA-01177、ORA-01215またはORA-01216です。この場合は、「新しい制御ファイルの作成」で作成したバックアップのファイルをリストアし、正しいファイル名を使用してそのタスクの手順を再実行してください。

親トピック: 制御ファイル作成後のトラブルシューティング

10.5 制御ファイルのバックアップ

制御ファイルをバックアップするには、ALTER DATABASE BACKUP CONTROLFILE文を使用します。

次の2つのオプションがあります。

- 次の文を使用して、制御ファイルをバイナリ・ファイル(既存の制御ファイルの複製)にバックアップを作成します。

  ```
  ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/control.bkp';
  ```

- 後で制御ファイルの再作成に使用できるSQL文を生成します。

  ```
  ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
  ```

  このコマンドによってSQLスクリプトがトレース・ファイルに書き込まれ、これを取得および編集して制御ファイルを再作成できます。トレース・ファイルの名前と場所は、アラート・ログで調べられます。

  関連項目:

  - 制御ファイルのバックアップの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください
  - 「アラート・ログの表示」

親トピック: 制御ファイルの管理

10.6 現行のコピーを使用した制御ファイルのリカバリ

現行のバックアップまたは多重化コピーから制御ファイルをリカバリできます。

- 制御ファイルのコピーを使用した制御ファイル破損からのリカバリ
  制御ファイルが破損した場合は、制御ファイルのコピーを使用してリカバリできます。
- 制御ファイルのコピーを使用した永続的なメディア障害からのリカバリ
  恒久的なメディア障害がある場合は、制御ファイルのコピーを使用してリカバリできます。

親トピック: 制御ファイルの管理

10.6.1 制御ファイルのコピーを使用した制御ファイル破損からのリカバリ

制御ファイルが破損した場合は、制御ファイルのコピーを使用してリカバリできます。

この方法では、CONTROL_FILESパラメータで指定されている制御ファイルの1つが破損し、制御ファイルのディレクトリにはまだアクセス可能で、制御ファイルの多重化コピーがある場合を想定しています。

1. インスタンスを停止してから、オペレーティング・システム・コマンドを使用して、破損した制御ファイルに正常なコピーを上書きします。

```
% cp /u03/oracle/prod/control03.ctl  /u02/oracle/prod/control02.ctl
```

2. SQL*Plusを起動してデータベースをオープンします。

```
SQL> STARTUP
```

親トピック: 現行のコピーを使用した制御ファイルのリカバリ

10.6.2 制御ファイルのコピーを使用した永続的なメディア障害からのリカバリ

恒久的なメディア障害がある場合は、制御ファイルのコピーを使用してリカバリできます。

この方法では、永続的なメディア障害のために、CONTROL_FILESパラメータで指定されている制御ファイルの1つにアクセスできず、制御ファイルの多重化コピーがある場合を想定しています。

1. インスタンスを停止してから、オペレーティング・システム・コマンドを使用して、制御ファイルの現行のコピーをアクセス可能な新しい位置にコピーします。

```
% cp /u01/oracle/prod/control01.ctl  /u04/oracle/prod/control03.ctl
```

2. 初期化パラメータ・ファイルのCONTROL_FILESパラメータを編集し、破損したファイルの位置を新しい位置に置き換えます。

```
CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
                 /u02/oracle/prod/control02.ctl,
                 /u04/oracle/prod/control03.ctl)
```

3. SQL*Plusを起動してデータベースをオープンします。

```
SQL> STARTUP
```

多重制御ファイルがある場合は、CONTROL_FILES初期化パラメータを編集することで、データベースを迅速に起動できます。破損した制御ファイルをCONTROL_FILES設定から削除すると、即時にデータベースを再起動できます。次に、破損した制御ファイルを再作成し、CONTROL_FILES初期化パラメータにリカバリした制御ファイルを設定してから、データベースを停止し、再起動します。

親トピック: 現行のコピーを使用した制御ファイルのリカバリ

10.7 制御ファイルの削除

制御ファイルは削除できますが、データベースには常に少なくとも2つの制御ファイルが存在する必要があります。

たとえば、制御ファイルの位置が不適切な場合は、データベースからその制御ファイルを削除できます。

1. データベースを停止します。
2. データベース初期化パラメータ・ファイルのCONTROL_FILESパラメータを編集して、古い制御ファイル名を削除します。

3. データベースを再起動します。

ノート:

この操作では、不要な制御ファイルをディスクから物理的に削除することはできません。データベースから制御ファイルを削除した後、オペレーティング・システムのコマンドを使用して不要なファイルを削除してください。

親トピック: 制御ファイルの管理

10.8 制御ファイルのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューのセットに対して制御ファイルに関する情報を問い合せることができます。

次のビューには、制御ファイルに関する情報が表示されます。

ビュー	説明
V$DATABASE	制御ファイル内のデータベース情報が表示されます。
V$CONTROLFILE	制御ファイル名が一覧表示されます。
V$CONTROLFILE_RECORD_SECTION	制御ファイルのレコード・セクションに関する情報が表示されます。
V$PARAMETER	CONTROL_FILES 初期化パラメータで指定されている制御ファイルの名前が表示されます。

この例では、制御ファイル名が一覧表示されます。

```
SQL> SELECT NAME FROM V$CONTROLFILE;
NAME
------------------------------------
/u01/oracle/prod/control01.ctl
/u02/oracle/prod/control02.ctl
/u03/oracle/prod/control03.ctl
```

親トピック: 制御ファイルの管理

11 REDOログの管理

REDOログの管理では、REDOログ・グループおよびメンバーの作成、REDOログ・メンバーの再配置および名前変更、REDOログ・グループおよびメンバーの削除、ログ・スイッチの強制などのタスクを完了します。

- REDOログの概要
  リカバリ操作にとって最も重要な構造であるREDOログは、2つ以上の事前に割り当てられたファイルで構成されており、データベースへの変更があるたびにその変更内容が格納されます。Oracle Databaseの各インスタンスには、インスタンス障害の場合にデータベースを保護するためにREDOログが関連付けられています。
- REDOログの計画
  データベース・インスタンスREDOログを構成する場合は、ガイドラインに従うことができます。
- REDOログ・グループおよびメンバーの作成
  データベースのREDOログを事前に計画し、データベースの作成時に必要なREDOログ・ファイルのグループとメンバーをすべて作成してください。しかし、グループやメンバーの追加作成が必要な状況もあります。たとえば、REDOログにグループを追加することによって、REDOログ・グループの可用性の問題を解決できます。
- REDOログ・メンバーの再配置および名前変更
  オペレーティング・システムのコマンドを使用してREDOログを再配置し、ALTER DATABASE文を使用してデータベースにそのREDOログの新しい名前(位置)を通知できます。
- REDOログ・グループおよびメンバーの削除
  場合によっては、REDOログ・メンバーを含むグループ全体を削除できます。
- ログ・スイッチの強制
  ログ・スイッチは、LGWRがあるREDOログ・グループへの書込みを中止して、別のログ・グループへの書込みを開始するときに発生します。デフォルトでは、現行のREDOログ・ファイル・グループが一杯になると、ログ・スイッチが自動的に発生します。
- REDOログ・ファイル内のブロックの検証
  データベースは、チェックサムを使用してREDOログ・ファイル内のブロックを検証するように構成できます。
- REDOログ・ファイルのクリア
  データベースがオープンしている間にREDOログ・ファイルが破損し、その結果アーカイブが継続できなくなり、データベース・アクティビティが停止することがあります。
- FORCE LOGGING設定の優先順位
  FORCE LOGGINGおよびNOLOGGINGは、データベース、プラガブル・データベース(PDB)、表領域またはデータベース・オブジェクトなどの様々なレベルで設定できます。1つ以上のレベルでFORCE LOGGINGが設定されている場合、FORCE LOGGING設定の優先順位によってREDOログに記録される内容が決まります。
- REDOログ・データ・ディクショナリ・ビュー
  REDOログの情報について、一連のデータ・ディクショナリ・ビューを問い合せることができます。

関連項目:

Oracle Databaseサーバーによって作成および管理されるREDOログ・ファイルの詳細は、「Oracle Managed Filesの使用」

親トピック: Oracle Databaseの構造と記憶域

11.1 REDOログの概要

リカバリ操作にとって最も重要な構造であるREDOログは、2つ以上の事前に割り当てられたファイルで構成されており、データベースへの変更があるたびにその変更内容が格納されます。Oracle Databaseの各インスタンスには、インスタンス障害の場合にデータベースを保護するためにREDOログが関連付けられています。

- REDOスレッド
  複数のデータベース・インスタンスに関する内容では、各データベース・インスタンスのREDOログはREDOスレッドとも呼ばれます。
- REDOログの内容
  REDOログ・ファイルには、REDOレコードが書き込まれます。
- Oracle DatabaseによるREDOログへの書込み
  データベースのREDOログは、2つ以上のREDOログ・ファイルから構成されます。データベースでは、一方のファイルのアーカイブ中にも(データベースがARCHIVELOGモードのとき)、他方のファイルが常に書込み可能であることを保証するために、最低2つのファイルを必要とします。

親トピック: REDOログの管理

11.1.1 REDOスレッド

複数のデータベース・インスタンスに関する内容では、各データベース・インスタンスのREDOログはREDOスレッドとも呼ばれます。

標準的な構成では、Oracle Databaseにアクセスするデータベース・インスタンスは1つのみであるため、スレッドは1つしか存在しません。ただし、Oracle Real Application Clusters環境では、複数のインスタンスが単一のデータベースに同時にアクセスし、各インスタンスが専用のREDOスレッドを持ちます。各インスタンスが別々のREDOスレッドを使用するため、1つのセットのREDOログ・ファイルで競合が回避され、その結果、潜在的なパフォーマンスのボトルネックを解消できます。

この章では、標準的な単一インスタンスのOracle Databaseで、REDOログを構成して管理する方法について説明します。文のすべての説明と例では、スレッド番号を1と想定します。Oracle Real Application Clusters環境におけるREDOログ・グループについては、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

親トピック: REDOログの概要

11.1.2 REDOログの内容

REDOログ・ファイルには、REDOレコードが書き込まれます。

REDOレコードはREDOエントリとも呼ばれ、変更ベクトル(データベース内の単一ブロックに加えられた変更の記述)のグループからなっています。たとえば、従業員表の給与値を変更する場合は、その表のデータ・セグメント・ブロック、UNDOセグメント・データ・ブロックおよびUNDOセグメントのトランザクション表の変更内容を記述する変更ベクトルを含むREDOレコードが生成されます。

REDOエントリには、UNDOセグメントなど、データベースに対するすべての変更の再構築に使用できるデータが記録されます。したがって、REDOログによってロールバック・データも保護されます。REDOデータベースを使用してデータベースをリカバリすると、データベースはREDOレコード内の変更ベクトルを読み込んで変更内容を関連ブロックに適用します。

REDOレコードは、循環方式でシステム・グローバル領域(SGA)のREDOログ・バッファに入れられ(「Oracle DatabaseによるREDOログへの書込み」を参照)、データベースのバックグラウンド・プロセスであるログ・ライター(LGWR)によってREDOログ・ファイルの1つに書き込まれます。トランザクションがコミットされると、LGWRによってそのトランザクションのREDOレコードがSGAのREDOログ・バッファからREDOログ・ファイルに書き込まれ、コミットされた各トランザクションのREDOレコードを識別するためにシ

ステム変更番号 (SCN)が割り当てられます。特定のトランザクションに対応付けられたすべてのREDOログ・レコードがディスク上のオンライン・ログに安全に書き込まれた場合にのみ、ユーザー・プロセスはトランザクションがコミットされたことを示す通知を受け取ります。

また、REDOレコードは、対応するトランザクションがコミットされる前にREDOログ・ファイルに書き込むこともできます。REDOログ・バッファが一杯になるか、別のトランザクションがコミットされると、一部のREDOレコードがコミットされていない可能性があっても、LGWRはREDOログ・バッファ内のすべてのREDOログ・エントリをREDOログ・ファイルにフラッシュします。データベースでは、必要に応じて、これらの変更をロールバックできます。

親トピック: REDOログの概要

11.1.3 Oracle DatabaseによるREDOログへの書込み

データベースのREDOログは、2つ以上のREDOログ・ファイルから構成されます。データベースでは、一方のファイルのアーカイブ中にも(データベースがARCHIVELOGモードのとき)、他方のファイルが常に書込み可能であることを保証するために、最低2つのファイルを必要とします。

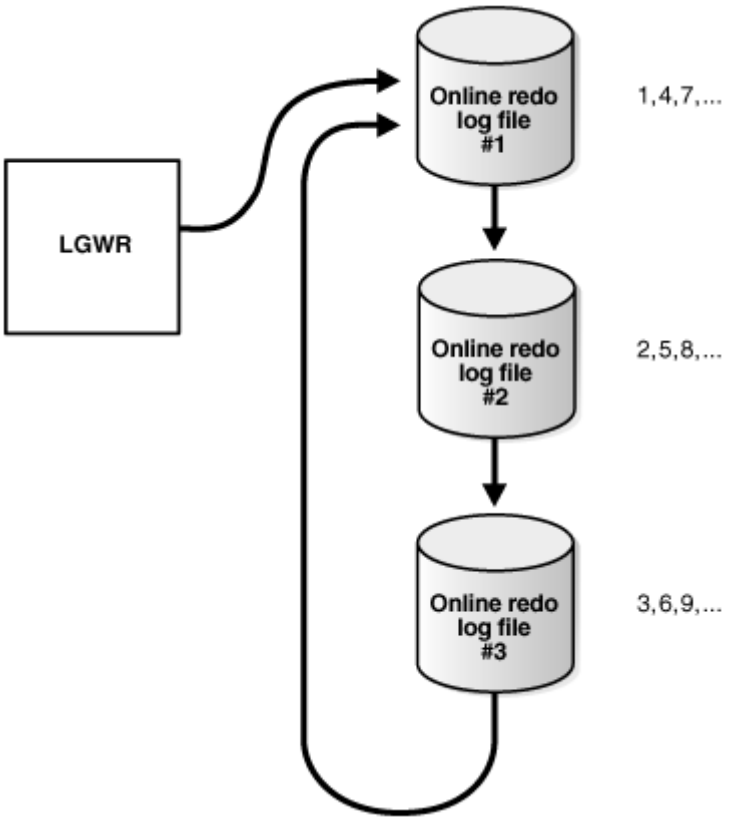
詳細は、「アーカイブREDOログ・ファイルの管理」を参照してください。

LGWRは、REDOログ・ファイルに循環方式で書き込みます。つまり、現行のREDOログ・ファイルが一杯になると、LGWRは次に使用可能なREDOログ・ファイルへの書込みを開始します。使用可能な最後のREDOログ・ファイルが一杯になると、LGWRは最初のREDOログ・ファイルに戻って書込みを行い、再び循環を開始します。図11-1に、REDOログ・ファイルへの循環方式の書込みを示します。各ラインの隣の番号は、LGWRが各REDOログ・ファイルに書き込む順序を示しています。

一杯になったREDOログ・ファイルは、アーカイブが使用可能になっているかどうかに応じて、LGWRで再利用できます。

- アーカイブが使用禁止になっている場合(データベースがNOARCHIVELOGモードのとき)、一杯になったREDOログ・ファイルは、そこに記録された変更がデータファイルに書き込まれた後に使用可能になります。
- アーカイブが有効(データベースがARCHIVELOGモード)の場合は、記録された変更がデータファイルに書き込まれ、さらにそのファイルがアーカイブされた後、一杯になったREDOログ・ファイルをLGWRで使用できるようになります。

図11-1 LGWRによるREDOログ・ファイルの再利用



- アクティブ(カレント)および非アクティブなREDOログ・ファイル
  Oracle Databaseは、REDOログ・ファイルを一度に1つのみ使用して、REDOログ・バッファから書き込まれたREDOレコードを格納します。LGWRが書込み中のREDOログ・ファイルを現行のREDOログ・ファイルと呼びます。
- ログ・スイッチとログ順序番号
  ログ・スイッチは、データベースが、あるREDOログ・ファイルへの書込みを終了して他のファイルへの書込みを開始するポイントです。現行のREDOログ・ファイルが完全に一杯になり、引き続き次のREDOログ・ファイルへの書込みが必要になると、通常はログ・スイッチが発生します。

親トピック: REDOログの概要

11.1.3.1 アクティブ(カレント)および非アクティブなREDOログ・ファイル

Oracle Databaseは、REDOログ・ファイルを一度に1つのみ使用して、REDOログ・バッファから書き込まれたREDOレコードを格納します。LGWRが書込み中のREDOログ・ファイルを現行のREDOログ・ファイルと呼びます。

インスタンス・リカバリに必要なREDOログ・ファイルをアクティブなREDOログ・ファイルと呼びます。また、インスタンス・リカバリには不要なREDOログ・ファイルを 非アクティブなREDOログ・ファイルと呼びます。

アーカイブを使用可能にしている場合は(データベースがARCHIVELOGモードのとき)、いずれかのアーカイバ・バックグラウンド・プロセス(ARCn)によって内容がアーカイブされるまで、データベースはアクティブなオンライン・ログ・ファイルの再利用または上書きができません。アーカイブが使用禁止になっている場合(データベースがNOARCHIVELOGモードのとき)は、最後のREDOログ・ファイルが一杯になって非アクティブになると、LGWRは順序内の次のログ・ファイルを上書きして書込みを継続します。

親トピック: Oracle DatabaseによるREDOログへの書込み

11.1.3.2 ログ・スイッチとログ順序番号

ログ・スイッチは、データベースが、あるREDOログ・ファイルへの書込みを終了して他のファイルへの書込みを開始するポイントです。現行のREDOログ・ファイルが完全に一杯になり、引き続き次のREDOログ・ファイルへの書込みが必要になると、通常はログ・スイッチが発生します。

ただし、ログ・スイッチは、現行のREDOログ・ファイルが完全に一杯になっているかどうかに関係なく、定期的に発生するように構成できます。ログ・スイッチは、手動で強制的に発生させることもできます。

Oracle Databaseは、ログ・スイッチが発生してLGWRが書込みを開始するたびに、各REDOログ・ファイルに新しいログ順序番号を割り当てます。データベースがREDOログ・ファイルをアーカイブしても、そのファイルのログ順序番号は変わりません。一巡して再び使用可能になったREDOログ・ファイルには、次に使用可能なログ順序番号が割り当てられます。

各オンラインREDOログ・ファイルまたはアーカイブREDOログ・ファイルは、そのログ順序番号で一意に識別されます。クラッシュ、インスタンスまたはメディア・リカバリ時に、データベースでは、必要なアーカイブおよびREDOログ・ファイルのログ順序番号を使用して、REDOログ・ファイルが昇順で正しく適用されます。

親トピック: Oracle DatabaseによるREDOログへの書込み

11.2 REDOログの計画

データベース・インスタンスREDOログを構成する場合は、ガイドラインに従うことができます。

- REDOログ・ファイルの多重化
  REDOログそのものに関わる障害から保護するために、Oracle Databaseでは多重REDOログ、つまりREDOログの2つ以上の同一のコピーを異なる場所に自動的に維持する機能を使用できます。
- 異なるディスクへのREDOログ・メンバーの配置
  多重REDOログ・ファイルを設定する場合は、グループのメンバーを異なる物理ディスク上に配置します。このようにすると、1つのディスクで障害が発生しても、LGWRが使用できなくなるのはグループの1つのメンバーのみで、それ以外のメンバーは引き続きLGWRにアクセスできるので、インスタンスは継続して機能します。
- REDOログ・ファイルのサイズの計画
  REDOログ・ファイルのサイズを設定するときは、REDOログをアーカイブするかどうかを考慮してください。REDOログ・ファイルのサイズは、一杯になったグループを1つのオフライン記憶メディア(テープやディスクなど)にアーカイブできるとともに、そのメディア上の未使用領域が最小になるように設定します。
- REDOログ・ファイルのブロック・サイズの計画
  2KBから32KBの間で設定可能なデータベース・ブロック・サイズと異なり、REDOログ・ファイルではディスクの物理セクター・サイズと同じブロック・サイズが常にデフォルトとなります。これは通例として512バイトです。
- 適切なREDOログ・ファイル数の選択
  データベース・インスタンスに対するREDOログ・ファイルの適切な数を決定する最良の方法は、様々な構成をテストすることです。最適な構成では、LGWRがREDOログ情報を書き込むのを妨げない最小の数がグループ数になります。
- アーカイブ・タイムラグの制御
  すべての有効なREDOログ・スレッドが定期的に現在のログを切り替えるように強制できます。

親トピック: REDOログの管理

11.2.1 REDOログ・ファイルの多重化

REDOログそのものに関わる障害から保護するために、Oracle Databaseでは多重REDOログ、つまりREDOログの2つ以上の同一のコピーを異なる場所に自動的に維持する機能を使用できます。

これを最大限に活用するには、これらの場所は異なるディスク上に置きます。ただし、REDOログのすべてのコピーが同じディスク上にあっても、冗長性によってI/Oエラー、ファイル破損などに対して保護されます。REDOログ・ファイルを多重化すると、LGWRによって同じREDOログ情報が複数の同一のREDOログ・ファイルに書き込まれるため、REDOログの単一の障害箇所は問題になりません。

多重化を実装するには、REDOログ・ファイルのグループを作成します。グループは、REDOログ・ファイルとその多重化されたコピーで構成されます。それぞれの同一コピーはグループのメンバーと呼ばれます。各REDOログ・グループは、グループ1、グループ2のように数値で定義されます。

図11-2 多重REDOログ・ファイル

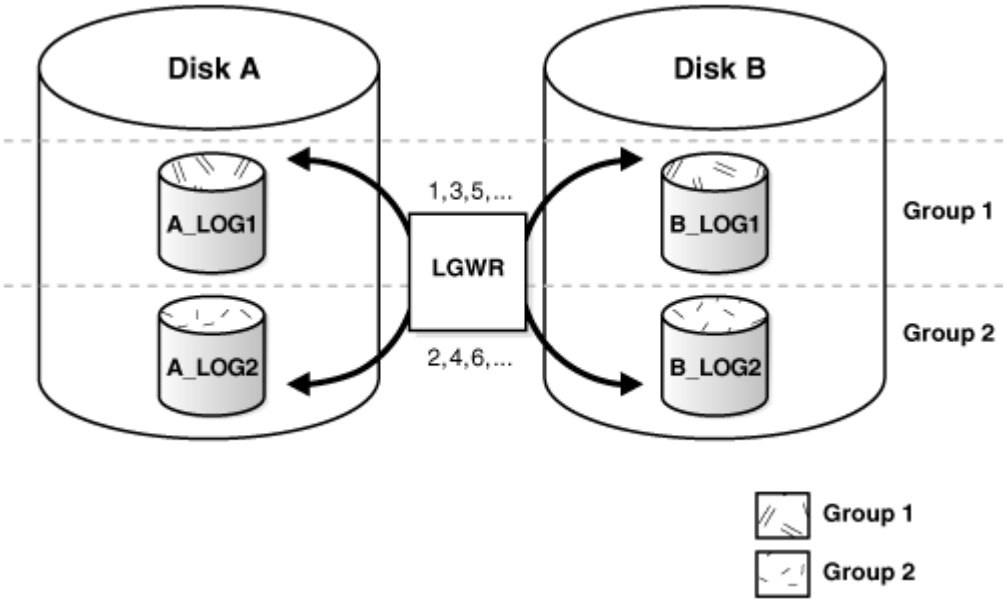


図11-2では、A_LOG1とB_LOG1はどちらもグループ1のメンバーで、A_LOG2とB_LOG2はどちらもグループ2のメンバーです。同じグループの各メンバーは同じサイズである必要があります。

LGWRによって割り当てられる同一のログ順序番号で示されるように、ログ・ファイル・グループの各メンバーは同時にアクティブ(LGWRによって同時に書き込まれる)になります。図11-2では、最初にLGWRはA_LOG1およびB_LOG1の両方に同時に書き込みます。次にA_LOG2およびB_LOG2の両方に同時に書き込み、以下同様に続きます。LGWRでは、異なるグループのメンバー(たとえばA_LOG1とB_LOG2)に同時に書き込まれることはありません。

ノート:

REDO ログ・ファイルは多重化することをお薦めします。これは、リカバリが必要になったときにログ・ファイルのデータが失われていると、致命的な事態を招くおそれがあるためです。REDO ログを多重化すると、データベースの実行時の I/O の量が増加するため注意してください。構成によっては、これによってデータベース・パフォーマンス全体が影響を受けます。

- REDOログの障害への対処
  LGWRがグループのメンバーに書き込めない場合、データベースはそのメンバーにINVALIDを示すマークを付け、LGWRトレース・ファイルとデータベース・アラート・ログに、アクセス不可能ファイルの問題を示すエラー・メッセージを書き込みます。
- 有効な構成と無効な構成
  ほとんどの場合、多重REDOログを対称にします(すべてのREDOログのグループに同じ数のメンバーを使用します)。ただし、データベースにとっては、多重REDOログが対称である必要はありません。

親トピック: REDOログの計画

11.2.1.1 REDOログの障害への対処

LGWRがグループのメンバーに書き込めない場合、データベースはそのメンバーにINVALIDを示すマークを付け、LGWRトレー

ス・ファイルとデータベース・アラート・ログに、アクセス不可能ファイルの問題を示すエラー・メッセージを書き込みます。

REDOログ・メンバーが使用不可能な場合のLGWR固有の処理は、次の表に示すように、使用不可能になった理由によって決定します。

条件	LGWRの処理
LGWR がグループ内の最低 1 つのメンバーに正常に書き込める場合	通常どおり書込みが行われます。LGWR はグループのうち使用可能なメンバーにのみ書き込み、使用不可のメンバーは無視します。
グループをアーカイブする必要があるため、LGWR がログ・スイッチの発生時に次のグループにアクセスできない場合	データベース操作は、グループが使用可能になるか、グループがアーカイブされるまで一時的に停止します。
メディア障害のため、ログ・スイッチの発生時に LGWR が次のグループのどのメンバーにもアクセスできない場合	Oracle Database はエラーを返し、データベース・インスタンスは停止します。この場合は、データベース上で REDO ログ・ファイルの損失からのメディア・リカバリを実行する必要があります。 データベースのチェックポイントが損失した REDO ログを超えて移動した場合、その REDO ログに記録されたデータは、データベースによってデータファイルに保存されているため、メディア・リカバリは必要ありません。アクセスできない REDO ログ・グループのみ削除してください。データベースによって不良ログがアーカイブされていない場合は、`ALTER DATABASE CLEAR LOGFILE UNARCHIVED` を使用してアーカイブを使用禁止にしてから、ログを削除してください。
LGWR が書込み中に、グループのすべてのメンバーに突然アクセスできなくなった場合	Oracle Database はエラーを返し、データベース・インスタンスは即座に停止します。この場合は、メディア・リカバリを実行する必要があります。ログのドライブを不注意からオフにした場合など、ログを含むメディアが実際には失われていなければ、メディア・リカバリは不要です。この場合は、ドライブをオンに戻して、データベースで自動インスタンス・リカバリを実行します。

親トピック: REDOログ・ファイルの多重化

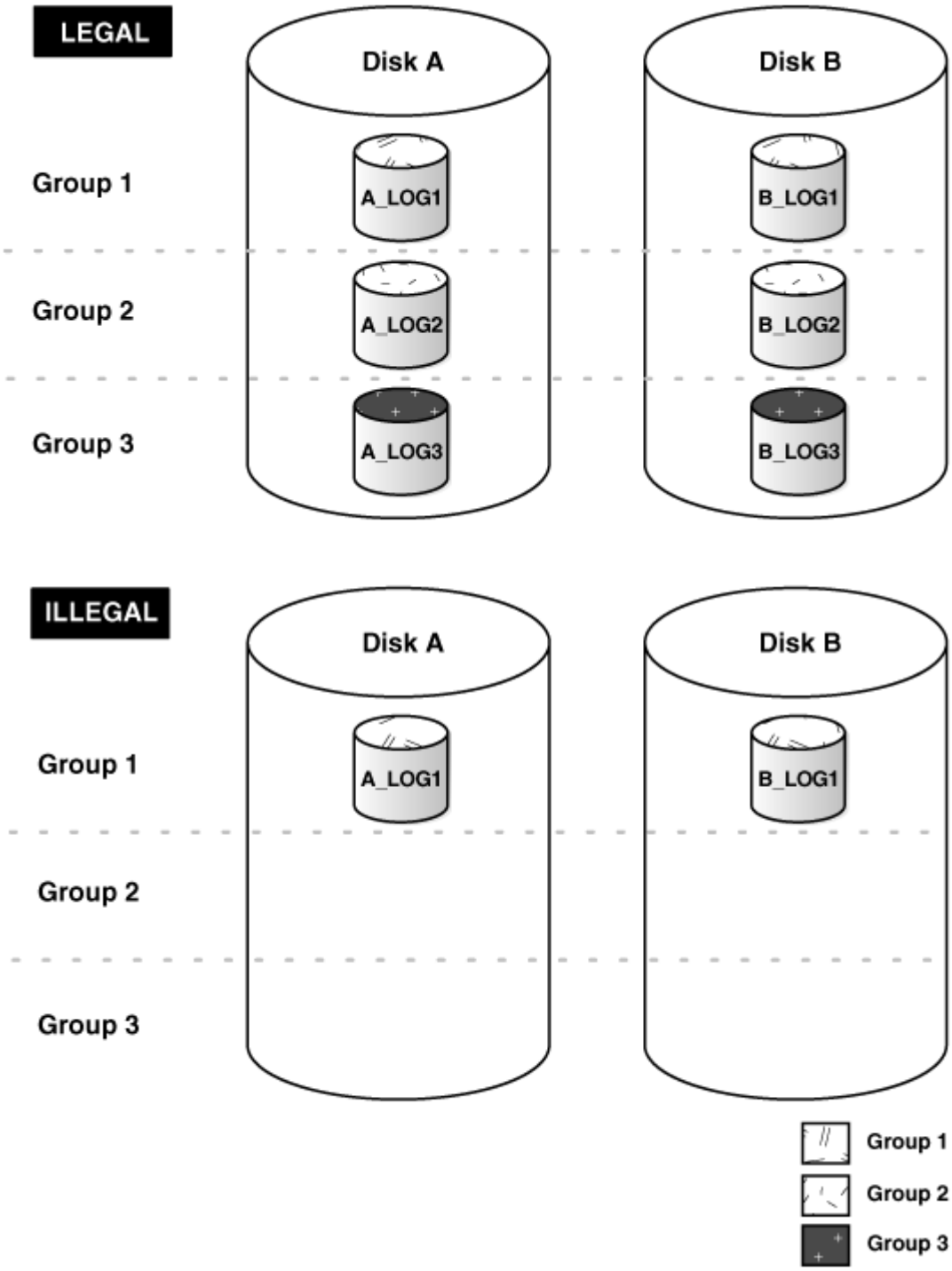
11.2.1.2 有効な構成と無効な構成

ほとんどの場合、多重REDOログを対称にします(すべてのREDOログのグループに同じ数のメンバーを使用します)。ただし、データベースにとっては、多重REDOログが対称である必要はありません。

たとえば、1つのグループには1つのメンバーのみを含め、他のグループには2つのメンバーを含められます。この構成によって、一時的に一部のREDOログ・メンバーに影響を与えても他のメンバーに影響が及ばないようなディスク障害に対して保護できます。

インスタンスのREDOログに関する唯一の要件は、最低2つのグループを持つことです。図11-3に、多重REDOログに有効な構成と無効な構成を示します。2番目の構成は、グループが1つしかないため無効です。

図11-3 多重REDOログ・ファイルの有効な構成と無効な構成



親トピック: REDOログ・ファイルの多重化

11.2.2 異なるディスクへのREDOログ・メンバーの配置

多重REDOログ・ファイルを設定する場合は、グループのメンバーを異なる物理ディスク上に配置します。このようにすると、1つのディスクで障害が発生しても、LGWRが使用できなくなるのはグループの1つのメンバーのみで、それ以外のメンバーは引き続きLGWRにアクセスできるので、インスタンスは継続して機能します。

REDOログをアーカイブする場合は、バックグラウンド・プロセスLGWRとARCn間の競合をなくすために、REDOログ・メンバーを複数のディスクに分散します。たとえば、多重化REDOログ・メンバーのグループが2つある場合(二重化REDOログ)、各メンバーを異なるディスクに配置し、アーカイブ先を5つ目のディスクに設定します。このようにすると、LGWR(メンバーへの書込み)とARCn(メンバーの読込み)間の競合は発生しません。

データ・ブロックやREDOレコードの書込み時の競合を減らすために、データファイルもREDOログ・ファイルとは異なるディスク上に配置することをお薦めします。

親トピック: REDOログの計画

11.2.3 REDOログ・ファイルのサイズの計画

REDOログ・ファイルのサイズを設定するときは、REDOログをアーカイブするかどうかを考慮してください。REDOログ・ファイルのサイズは、一杯になったグループを1つのオフライン記憶メディア(テープやディスクなど)にアーカイブできるとともに、そのメディア上の未使用領域が最小になるように設定します。

たとえば、一杯になった1つのREDOログ・グループを1本のテープにアーカイブし、そのテープの記憶容量の49%が未使用のまま残っているとします。この場合は、REDOログ・ファイルのサイズを小さくして、テープごとに2つずつREDOログ・グループがアーカイブされるように構成することをお薦めします。

同一の多重REDOログ・グループのメンバーは、すべて同じサイズにする必要があります。異なるグループのメンバーは異なるサイズにすることができます。ただし、グループ間でファイル・サイズを変えても特に利点はありません。ログ・スイッチ間にチェックポイントが発生するように設定していない場合は、グループのサイズを同一に設定して、チェックポイントが一定の間隔で発生することを保証してください。

REDOログ・ファイルの最小サイズは4MBです。

関連項目:

使用しているオペレーティング・システム固有のOracleマニュアルを参照してください。REDOログ・ファイルのデフォルトのサイズは、オペレーティング・システムによって異なります。

親トピック: REDOログの計画

11.2.4 REDOログ・ファイルのブロック・サイズの計画

2KBから32KBの間で設定可能なデータベース・ブロック・サイズと異なり、REDOログ・ファイルではディスクの物理セクター・サイズと同じブロック・サイズが常にデフォルトとなります。これは通例として512バイトです。

新しい大容量ディスク・ドライブでは、ECC機能およびフォーマット効率向上のために4KBのセクター・サイズを提供しているものもあります。Oracle Databaseプラットフォームの大部分では、この大きい方のセクター・サイズを検出できます。その後、データベースはブロック・サイズが4KBのREDOログ・ファイルをそれらのディスク上に自動的に作成します。

ただし、4KBのブロック・サイズでは、REDO時のディスク領域の消費が増加します。実際に、4KBのブロックと512バイトのブロックではREDO時のディスク領域の消費量は大幅に異なります。V$SESSTATおよびV$SYSSTATビューに格納されている統計を表示して、REDO時のディスク領域の消費量を判別できます。

```
SQL> SELECT name, value FROM v$sysstat WHERE name = 'redo wastage';

NAME                                  VALUE
-------------------------------- ----------
redo wastage                        17941684
```

REDO時のディスク領域の消費を増やさないようにするために、エミュレーションモードのディスク(ディスクのインタフェースにおいて512バイトのセクター・サイズをエミュレートする、セクター・サイズが4KBのディスク・ドライブ)を使用している場合は、REDOログのブロック・サイズに512バイトまたは一部のプラットフォームで1KBを指定し、デフォルトの4KBをオーバーライドできます。ただし、REDOログの書込みが4KBの物理セクターの始まりの位置と揃えられていない場合は、パフォーマンスが大幅に低下します。4KBの物理セクターで512バイトのスロット8つのうち7つの位置が揃っていないために、通常はパフォーマンスの低下が発生します。このため、セクター・サイズが4Kのエミュレーションモード・ディスク上でREDOログのブロック・サイズを計画する場合には、パ

フォーマンスおよびディスク領域の消費のバランスを検討する必要があります。

CREATE DATABASE、ALTER DATABASE、およびCREATE CONTROLFILE文内のキーワードBLOCKSIZEによって、オンラインREDOログ・ファイルのブロック・サイズを指定できます。一部のプラットフォームで使用可能なブロック・サイズは、512および4096です。その他のプラットフォームで使用可能なブロック・サイズは、1024および4096です。

次の文は、ブロックサイズが512バイトのREDOログ・ファイルのグループを追加します。BLOCKSIZE 512の句は、セクター・サイズが512バイトのディスクでは有効ですが、必須ではありません。セクター・サイズが4KBのエミュレーションモードのディスクでは、BLOCKSIZE 512の句がデフォルトの4KBをオーバーライドします。

```
ALTER DATABASE orcl ADD LOGFILE
  GROUP 4 ('/u01/logs/orcl/redo04a.log','/u01/logs/orcl/redo04b.log')
  SIZE 100M BLOCKSIZE 512 REUSE;
```

REDOログ・ファイルのブロック・サイズを確認するには、次の問合せを実行します。

```
SQL> SELECT BLOCKSIZE FROM V$LOG;
BLOCKSIZE
---------
      512
```

関連項目:

- ALTER DATABASE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- V$SESSTATビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- V$SYSSTATビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: REDOログの計画

11.2.5 適切なREDOログ・ファイル数の選択

データベース・インスタンスに対するREDOログ・ファイルの適切な数を決定する最良の方法は、様々な構成をテストすることです。最適な構成では、LGWRがREDOログ情報を書き込むのを妨げない最小の数がグループ数になります。

データベース・インスタンスに必要なグループが2つのみの場合もあります。また、LGWRが使用可能な再利用グループが常に存在するようにするため、データベース・インスタンスにグループを追加する必要がある場合もあります。テスト中に、現行のREDOログ構成に問題がないかどうかを確認するには、LGWRトレース・ファイルおよびデータベース・アラート・ログの内容を調べるのが最も容易です。チェックポイントが未完了か、またはグループがアーカイブされていないため、LGWRが頻繁にグループを待機することをメッセージが示す場合は、グループを追加してください。

インスタンスのREDOログ構成を設定または変更する前に、REDOログ・ファイル数を制限するパラメータを検討してください。次のパラメータは、データベースに追加できるREDOログ・ファイルの数を制限します。

- CREATE DATABASE文のMAXLOGFILESパラメータは、データベース当たりのREDOログ・ファイルの最大グループ数を決定します。グループの値は1からMAXLOGFILESです。MAXLOGFILESの制限を超えることができ、制御ファイルは必要に応じて拡張します。CREATE DATABASE文にMAXLOGFILESパラメータが指定されていない場合は、オペレーティング・システム固有のデフォルト値が使用されます。
- CREATE DATABASE文のMAXLOGMEMBERSパラメータは、グループに含まれるメンバーの最大値を決定します。この上限値を変更する唯一の方法は、MAXLOGFILESの場合と同様、データベースまたは制御ファイルを再作成することです。したがって、データベースを作成する前にこの上限値を十分検討してください。CREATE DATABASE文に

MAXLOGMEMBERSパラメータが指定されていない場合は、オペレーティング・システムのデフォルト値が使用されます。

関連項目:

MAXLOGFILESパラメータおよびMAXLOGMEMBERSパラメータのデフォルト値と有効な値については、オペレーティング・システム固有のOracleマニュアルを参照してください。

親トピック: REDOログの計画

11.2.6 アーカイブ・タイムラグの制御

すべての有効なREDOログ・スレッドが定期的に現在のログを切り替えるように強制できます。

プライマリ/スタンバイ・データベース構成では、REDOログをプライマリ・サイトにアーカイブし、その後スタンバイ・データベースに送信することによって変更がスタンバイ・データベースに反映されます。スタンバイ・データベースは、プライマリ・データベースのREDOログの変更がアーカイブされたREDOログにアーカイブされ、送信されるまで待機する必要があるため、スタンバイ・データベースに適用されている変更は、プライマリ・データベースで発生している変更から遅れることになります。この遅延を制限するために、ARCHIVE_LAG_TARGET初期化パラメータを設定できます。このパラメータを設定して、遅延時間を秒単位で指定できます。

- ARCHIVE_LAG_TARGET初期化パラメータの設定
  ARCHIVE_LAG_TARGET初期化パラメータを設定した場合、データベースによりインスタンスの現在のREDOログが定期的に確認され、ログを切り替えるタイミングが決定されます。
- ARCHIVE_LAG_TARGETの設定に影響する要因
  ARCHIVE_LAG_TARGET初期化パラメータを設定する場合は、いくつかの検討する要素があります。

親トピック: REDOログの計画

11.2.6.1 ARCHIVE_LAG_TARGET初期化パラメータの設定

ARCHIVE_LAG_TARGET初期化パラメータを設定した場合、データベースによりインスタンスの現在のREDOログが定期的に確認され、ログを切り替えるタイミングが決定されます。

次の条件が満たされると、インスタンスはログを切り替えます。

- カレント・ログがn秒前に作成され、カレント・ログのアーカイブ見積り時間がm秒(この時間はカレント・ログで使用されているREDOブロック数に比例する)の場合に、n +mがARCHIVE_LAG_TARGET初期化パラメータの値を超えている。
- カレント・ログにREDOレコードが含まれている。

Oracle Real Application Clusters環境では、他のスレッドが遅れている場合、インスタンスによって他のスレッドも切り替えられ、ログがアーカイブされます。これは、Oracle Real Application Clustersで2つのノードを持つプライマリ/セカンダリ構成を実行しているときのように、クラスタ内に他のインスタンスよりも稼働率の低いインスタンスがある場合に特に有効です。

ARCHIVE_LAG_TARGET初期化パラメータは、データベースの現在のログの持続時間の上限を秒単位で指定します。予測されるアーカイブ時間も考慮されるため、これは厳密なログ切替え時間ではありません。

- ARCHIVE_LAG_TARGET初期化パラメータを設定します。

次の初期化パラメータ設定では、ログ・スイッチ間隔を30分(標準的な値)に設定します。

```
ARCHIVE_LAG_TARGET = 1800
```

0(ゼロ)を指定すると、時間ベースのログ・スイッチ機能は使用禁止になります。これがデフォルトの設定です。

スタンバイ・データベースが存在していなくても、ARCHIVE_LAG_TARGET初期化パラメータを設定できます。たとえば、強制的にログ・スイッチを発生させてアーカイブを行うために、ARCHIVE_LAG_TARGET初期化パラメータを設定できます。

ARCHIVE_LAG_TARGETは動的パラメータで、ALTER SYSTEM SET文を使用して設定できます。

ノート:

Oracle Real Application Clusters 環境では、すべてのインスタンスの ARCHIVE_LAG_TARGET パラメータに同じ値を指定する必要があります。異なる値を設定すると、予測できない動作が発生します。

親トピック: アーカイブ・タイムラグの制御

11.2.6.2 ARCHIVE_LAG_TARGETの設定に影響する要因

ARCHIVE_LAG_TARGET初期化パラメータを設定する場合は、いくつかの検討する要素があります。

ARCHIVE_LAG_TARGET初期化パラメータを設定するかどうか、またはその設定値を決定するには、次の要因を考慮してください。

- 切替え(およびアーカイブ)に伴うオーバーヘッド
- ログが一杯になったときに発生する通常のログ・スイッチの頻度
- スタンバイ・データベースで許容できるREDO損失の量

指定した間隔より短い頻度ですでにログ・スイッチが自然に発生している状況では、ARCHIVE_LAG_TARGETを設定しても特に利点はありません。ただし、REDOの生成速度が一定でない場合は、時間間隔を指定することで、各カレント・ログが使用される時間範囲の上限を設定できます。

ARCHIVE_LAG_TARGET初期化パラメータに極端に小さい値を指定すると、パフォーマンスに悪影響を及ぼすことがあります。これは、小さい値によって頻繁にログ・スイッチが発生するためです。このパラメータには、プライマリ・データベースのパフォーマンスを低下させないように適切な値を設定してください。

親トピック: アーカイブ・タイムラグの制御

11.3 REDOログ・グループおよびメンバーの作成

データベースのREDOログを事前に計画し、データベースの作成時に必要なREDOログ・ファイルのグループとメンバーをすべて作成してください。しかし、グループやメンバーの追加作成が必要な状況もあります。たとえば、REDOログにグループを追加することによって、REDOログ・グループの可用性の問題を解決できます。

新たにREDOログ・グループおよびメンバーを作成するには、ALTER DATABASEシステム権限が必要です。データベースには、最大MAXLOGFILES個までグループを作成できます。

- REDOログ・グループの作成
  REDOログ・ファイルの新しいグループを作成するには、SQL文ALTER DATABASEでADD LOGFILE句を指定します。
- REDOログ・メンバーの作成
  完全なREDOログ・ファイルのグループを作成する必要がない場合もあります。つまり、グループはすでに存在しているが、そのグループの1つ以上のメンバーが(ディスク障害などにより)削除されたために完全ではない場合です。このような場合は、既存のグループに新しいメンバーを追加できます。

関連項目:

ALTER DATABASE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: REDOログの管理

11.3.1 REDOログ・グループの作成

REDOログ・ファイルの新しいグループを作成するには、SQL文ALTER DATABASEでADD LOGFILE句を指定します。

- ADD LOGFILE句を含むALTER DATABASE SQL文を実行します。

たとえば、次の文は、データベースに新しいREDOログ・グループを追加します。

```
ALTER DATABASE
  ADD LOGFILE ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 100M;
```

ノート:

新しいログ・メンバーのフルパス名を入力し、その場所を指定してください。指定しない場合、オペレーティング・システムに応じて、ファイルがデータベース・サーバーのデフォルト・ディレクトリまたはカレント・ディレクトリのいずれかに作成されます。

また、次のようにGROUP句を使用して、グループの識別番号を指定することもできます。

```
ALTER DATABASE
  ADD LOGFILE GROUP 10 ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo')
      SIZE 100M BLOCKSIZE 512;
```

グループ番号を使用することによって、REDOログ・グループの管理がより簡単になります。ただし、グループ番号には1からMAXLOGFILESの範囲の値を使用する必要があります。REDOログ・ファイルのグループ番号をスキップする(つまり、グループに10、20、30などの番号を付ける)と、データベースの制御ファイル内で不要な領域が消費されてしまうため、グループ番号はスキップしないでください。

前の文でのBLOCKSIZE句はオプションです。詳細は、「REDOログ・ファイルのブロック・サイズの計画」を参照してください。

親トピック: REDOログ・グループおよびメンバーの作成

11.3.2 REDOログ・メンバーの作成

完全なREDOログ・ファイルのグループを作成する必要がない場合もあります。つまり、グループはすでに存在しているが、そのグループの1つ以上のメンバーが(ディスク障害などにより)削除されたために完全ではない場合です。このような場合は、既存のグループに新しいメンバーを追加できます。

既存のグループのための新しいREDOログ・メンバーを作成するには:

- ADD LOGFILE MEMBER句を含むALTER DATABASE SQL文を実行します。

たとえば、次の文は、REDOログ・グループ2に新しいREDOログ・メンバーを追加します。

```
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2b.rdo' TO GROUP 2;
```

ファイル名は指定する必要がありますが、サイズを指定する必要はありません。新しいメンバーのサイズは、既存グループのメン

バーのサイズによって決まります。

ALTER DATABASE文を使用するときは、次の例のように、TO句にグループの他のメンバーをすべて指定することによって、目的のグループを選択的に識別できます。

```
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2c.rdo'
    TO ('/oracle/dbs/log2a.rdo', '/oracle/dbs/log2b.rdo');
```

ノート:

オペレーティング・システム上のファイルを作成する位置を指定するには、新しいログ・メンバーのファイル名を絶対パスで指定してください。そうしないと、ファイルはオペレーティング・システムごとに異なるデータベースのデフォルトのディレクトリまたはカレント・ディレクトリに作成されます。また、新しいログ・メンバーの状態は INVALID として表示されるので注意してください。これは正常であり、最初に使用するときにアクティブ(空白)に変更されます。

親トピック: REDOログ・グループおよびメンバーの作成

11.4 REDOログ・メンバーの再配置および名前変更

オペレーティング・システムのコマンドを使用してREDOログを再配置し、ALTER DATABASE文を使用してデータベースにそのREDOログの新しい名前(位置)を通知できます。

たとえば、REDOログ・ファイルが現在保存されているディスクを削除する場合や、データファイルや複数のREDOログ・ファイルが同一のディスク上に格納されていて、競合を少なくするためにそれらを分離する場合に、この手順が必要となります。

REDOログ・メンバーの名前を変更するには、ALTER DATABASEシステム権限が必要です。さらに、ファイルを目的の位置にコピーするためのオペレーティング・システム権限と、データベースをオープンしてバックアップするための権限が必要になることもあります。

REDOログを再配置したり、データベースにその他の構造上の変更を加えたりする前には、各操作の実行中に発生する問題に備えて、データベース全体のバックアップを作成してください。また、今後発生する可能性のある問題に備えて、一連のREDOログ・ファイルを名前変更または再配置した後、ただちにデータベースの制御ファイルのバックアップを作成してください。

REDOログを再配置するステップは、次のとおりです。これらのステップでは、次の状況を想定しています。

- ログ・ファイルは、diskaおよびdiskbという2つのディスク上にあります。
- REDOログは多重化され、最初のグループはメンバー/diska/logs/log1a.rdo と /diskb/logs/log1b.rdoからなり、2番目のグループはメンバー/diska/logs/log2a.rdoと /diskb/logs/log2b.rdoからなっています。
- diskaにあるREDOログ・ファイルをdiskcに再配置する必要があります。新しいファイル名には新しい位置が反映され、/diskc/logs/log1c.rdoおよび/diskc/logs/log2c.rdoとなります。

REDOログ・メンバーの名前を変更するには:

1. データベースを停止します。

   ```
   SHUTDOWN
   ```

2. REDOログ・ファイルを新しい位置にコピーします。

   REDOログ・メンバーなどのオペレーティング・システム・ファイルは、適切なオペレーティング・システム・コマンドを使用して

コピーする必要があります。ファイルのコピーに関する説明は、オペレーティング・システム固有のマニュアルを参照してください。

ノート:

SQL*Plus の HOST コマンドを使用すると、既存の SQL*Plus を終了せずにオペレーティング・システム・コマンドを使用してファイルをコピーできます(その他のオペレーティング・システムのコマンドも実行できます)。HOST という語のかわりに 1 文字を使用するオペレーティング・システムもあります。たとえば、UNIX では感嘆符(!)が使用できます。

次の例では、オペレーティング・システム・コマンド(UNIX)を使用して、REDOログ・メンバーを新しい位置に移動しています。

```
mv /diska/logs/log1a.rdo /diskc/logs/log1c.rdo
mv /diska/logs/log2a.rdo /diskc/logs/log2c.rdo
```

3. データベースを起動して、マウントします。ただし、オープンはしません。

```
CONNECT / as SYSDBA
STARTUP MOUNT
```

4. REDOログ・メンバーの名前を変更します。

ALTER DATABASE文でRENAME FILE句を使用して、データベースのREDOログ・ファイルの名前を変更します。

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

5. 通常の操作を実行するためにデータベースをオープンします。

REDOログの変更は、データベースがオープンされたときに有効となります。

```
ALTER DATABASE OPEN;
```

親トピック: REDOログの管理

11.5 REDOログ・グループおよびメンバーの削除

場合によっては、REDOログ・メンバーを含むグループ全体を削除できます。

たとえば、インスタンスのREDOログのグループ数を少なくする場合などです。また、1つ以上の特定のREDOログ・メンバーの削除が必要になる場合もあります。たとえば、ディスク障害が発生した場合は、アクセスできないファイルに書き込まれないように、障害のあったディスク上のREDOログ・ファイルをすべて削除します。これ以外にも、特定のREDOログ・ファイルが不要になることがあります。たとえば、適切ではない位置にファイルを配置した場合です。

- ログ・グループの削除
  REDOログ・グループを削除できます。
- REDOログ・メンバーの削除
  REDOログ・メンバーを削除できます。

親トピック: REDOログの管理

11.5.1 ログ・グループの削除

REDOログ・グループを削除できます。

REDOログ・グループを削除するには、ALTER DATABASEシステム権限が必要です。REDOログ・グループを削除する前に、次の制限と注意点について検討してください。

- グループ内のメンバー数にかかわらず、インスタンスには少なくとも2つのREDOログ・ファイルのグループが必要です。(1つのグループは1つ以上のメンバーから構成されます。)
- REDOログ・グループは、非アクティブである場合にのみ削除できます。カレントのグループを削除する必要がある場合は、最初にログ・スイッチを発生させる必要があります。
- 削除する前に、REDOログ・グループがアーカイブされていることを確認します(アーカイブが使用可能になっている場合)。アーカイブされているかどうかを確認するには、V$LOGビューを使用します。

```
SELECT GROUP#, ARCHIVED, STATUS FROM V$LOG;
   GROUP# ARC STATUS
--------- --- ----------------
        1 YES ACTIVE
        2 NO  CURRENT
        3 YES INACTIVE
        4 YES INACTIVE
```

REDOログ・グループを削除するには:

- DROP LOGFILE句を含むALTER DATABASE SQL文を実行します。

たとえば、次の文は、REDOログ・グループ3を削除します。

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

データベースからREDOログ・グループを削除するときは、Oracle Managed Files機能を使用していないかぎり、オペレーティング・システム・ファイルはディスクから削除されません。より正確に言えば、対応するデータベースの制御ファイルが更新されて、そのグループのメンバーがデータベース構造から削除されます。REDOログ・グループを削除した後は、この処理が正常に終了したことを確認してから、適切なオペレーティング・システム・コマンドを使用して、削除したREDOログ・ファイルを実際に削除します。

Oracle Managed Files機能を使用している場合は、オペレーティング・システム・ファイルのクリーン・アップが自動的に実行されます。

親トピック: REDOログ・グループおよびメンバーの削除

11.5.2 REDOログ・メンバーの削除

REDOログ・メンバーを削除できます。

REDOログ・メンバーを削除するには、ALTER DATABASEシステム権限が必要です。各REDOログ・メンバーを削除する前に、次の制限と注意点について検討してください。

- REDOログ・ファイルを削除することにより、多重REDOログ・ファイルを一時的に非対称にしても問題はありません。たとえば、多重化したREDOログ・ファイルのグループを使用している場合、他のすべてのグループにメンバーが2つずつ残っていても、あるグループのメンバーを1つ削除できます。ただし、すべてのグループに少なくともメンバーが2つ存在するように、この状態をただちに訂正し、REDOログの単一の障害箇所が発生する可能性を取り除いてください。
- グループ内のメンバー数にかかわらず、インスタンスには常に、少なくとも2つの有効なREDOログ・ファイル・グループが必要です。(1つのグループは1つ以上のメンバーから構成されます。)削除するメンバーがグループの最後の有効なメン

バーである場合は、他のメンバーが有効にならないかぎり、そのメンバーを削除できません。REDOログ・ファイルの状態を確認するには、V$LOGFILEビューを使用します。REDOログ・ファイルは、データベースがアクセスできないとINVALIDになります。データベースがそのログ・ファイルを完全でない、または正しくないと判断すると、そのログ・ファイルはSTALEになります。この失効したログ・ファイルは、次にそのグループがアクティブ・グループになったときに、再び有効になります。

- REDOログ・メンバーは、アクティブまたはカレント・グループの一部ではない場合のみ削除できます。アクティブ・グループのメンバーを削除する場合は、最初にログ・スイッチを発生させます。
- メンバーを削除する前に、そのREDOログ・メンバーが属するグループがアーカイブされていることを確認します(アーカイブが使用可能になっている場合)。アーカイブされているかどうかを確認するには、V$LOGビューを使用します。

特定の非アクティブなREDOログ・メンバーを削除するには:

- DROP LOGFILE MEMBER句を含むALTER DATABASE文を実行します。

次の文は、REDOログ/oracle/dbs/log3c.rdoを削除します。

```
ALTER DATABASE DROP LOGFILE MEMBER '/oracle/dbs/log3c.rdo';
```

データベースからREDOログ・メンバーを削除しても、オペレーティング・システム・ファイルはディスクから削除されません。より正確に言えば、対応するデータベースの制御ファイルが更新されて、データベース構造からメンバーが削除されます。REDOログ・ファイルを削除した後は、この処理が正常に終了したことを確認してから、適切なオペレーティング・システム・コマンドを使用して、削除したREDOログ・ファイルを実際に削除します。

アクティブ・グループのメンバーを削除するには、最初にログ・スイッチを発生させる必要があります。

親トピック: REDOログ・グループおよびメンバーの削除

11.6 ログ・スイッチの強制

ログ・スイッチは、LGWRがあるREDOログ・グループへの書込みを中止して、別のログ・グループへの書込みを開始するときに発生します。デフォルトでは、現行のREDOログ・ファイル・グループが一杯になると、ログ・スイッチが自動的に発生します。

REDOログのメンテナンス操作を実行するために、ログ・スイッチを強制的に発生させて、現在アクティブなグループを非アクティブの状態に変更できます。たとえば、現在アクティブなグループを削除する場合は、アクティブでない状態になるまでそのグループを削除できません。また、現在アクティブなグループのメンバーが完全に一杯になる前に、特定の時点でそのグループをアーカイブする必要がある場合にも、ログ・スイッチの強制的な実行が必要です。このオプションは、一杯になるまで長い時間を必要とする大きなREDOログ・ファイルが含まれた構成で有効です。

ログ・スイッチを強制するには、ALTER SYSTEM権限が必要です。

ログ・スイッチを強制するには、

- SWITCH LOGFILE句を含むALTER SYSTEM文を実行します。

たとえば、次の文は、ログ・スイッチを強制します。

```
ALTER SYSTEM SWITCH LOGFILE;
```

親トピック: REDOログの管理

11.7 REDOログ・ファイル内のブロックの検証

データベースは、チェックサムを使用してREDOログ・ファイル内のブロックを検証するように構成できます。

初期化パラメータDB_BLOCK_CHECKSUMをTYPICAL(デフォルト)に設定すると、ディスクに書き込まれる各データベース・ブロック(カレント・ログに書き込まれている各REDOログ・ブロックを含む)のチェックサムが計算されます。チェックサムは、ブロックのヘッダーに格納されます。

Oracle Databaseは、チェックサムを使用してREDOログ・ブロック内の破損を検出します。リカバリ処理中にREDOログ・ブロックがアーカイブ・ログから読み込まれるとき、およびブロックがアーカイブ・ログ・ファイルに書き込まれるとき、そのブロックの検証が行われます。破損が検出されると、エラーが発生しアラート・ログに書き込まれます。

REDOログ・ブロックのアーカイブ時に破損が検出されると、システムは、グループ内の別のメンバーからそのブロックを読み込もうとします。REDOログ・グループ内のすべてのメンバーでブロックが破損していると、アーカイブ処理は継続できません。

DB_BLOCK_CHECKSUMパラメータの値は、ALTER SYSTEM文で動的に変更できます。

> ノート:
>
> DB_BLOCK_CHECKSUM を使用可能にすると、わずかなオーバーヘッドとデータベースのパフォーマンスの低下が発生します。データベースのパフォーマンスを監視して、パフォーマンスを犠牲にしてでもデータ・ブロックのチェックサムを使用して破損を検出する利点があるかを判断してください。

関連項目:

DB_BLOCK_CHECKSUM初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: REDOログの管理

11.8 REDOログ・ファイルのクリア

データベースがオープンしている間にREDOログ・ファイルが破損し、その結果アーカイブが継続できなくなり、データベース・アクティビティが停止することがあります。

この状況で、データベースを停止せずにファイルを再初期化する手順:

- ALTER DATABASE CLEAR LOGFILE SQL文を実行します。

次の文は、REDOログ・グループ3のログ・ファイルをクリアします。

```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

この文は、REDOログの削除が不可能な次の2つの状況に対応できます。

- ログ・グループが2つのみの場合
- 破損したREDOログ・ファイルがカレント・グループに属する場合

破損したREDOログ・ファイルがアーカイブされていない場合は、この文にUNARCHIVEDキーワードを使用します。

```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 3;
```

この文によって、破損したREDOログはクリアされ、アーカイブを回避できます。クリアされたREDOログは、アーカイブされていなく

ても使用できます。

バックアップのリカバリに必要なログ・ファイルをクリアすると、そのバックアップからのリカバリ処理ができなくなります。データベースは、そのバックアップからのリカバリ処理ができないことを示すメッセージをアラート・ログに書き込みます。

ノート:

アーカイブされていない REDO ログ・ファイルをクリアする場合は、データベースのバックアップをもう 1 つ作成する必要があります。

オフライン表領域をオンラインにするために必要な、アーカイブされていないREDOログを初期化するには、ALTER DATABASE CLEAR LOGFILE文でUNRECOVERABLE DATAFILE句を指定します。

オフライン表領域をオンラインにするために必要なREDOログをクリアすると、その表領域は二度とオンラインにはできません。表領域を削除するか、不完全リカバリを実行する必要があります。正常にオフライン化された表領域には、リカバリは必要ありません。

親トピック: REDOログの管理

11.9 FORCE LOGGING設定の優先順位

FORCE LOGGINGおよびNOLOGGINGは、データベース、プラガブル・データベース(PDB)、表領域またはデータベース・オブジェクトなどの様々なレベルで設定できます。1つ以上のレベルでFORCE LOGGINGが設定されている場合、FORCE LOGGING設定の優先順位によってREDOログに記録される内容が決まります。

マルチテナント・コンテナ・データベース(CDB)および非CDBをFORCE LOGGINGモードに置くことができます。このモードでは、一時表領域および一時セグメントでの変更を除くデータベースのすべての変更が記録されます。この設定は、各表領域で指定するNOLOGGINGまたはFORCE LOGGING設定、および各データベース・オブジェクトで指定するNOLOGGING設定より優先され、これらの設定には影響されません。

また、表領域をFORCE LOGGINGモードに置くこともできます。個々のオブジェクトのNOLOGGING設定をオーバーライドして、一時セグメントへの変更を除き、表領域内のすべてのオブジェクトに対するすべての変更が記録されます。

さらに、logging_clauseを使用して様々なタイプのデータベース・オブジェクトのロギング属性を指定することで、REDOログ・ファイルに特定のDML操作をログするかどうか(LOGGINGまたはNOLOGGING)を決定することができます。次のデータベース・オブジェクト・タイプのロギング属性を指定できます。

- 表
- 索引
- マテリアライズド・ビュー

次の表は、各レベルのログ設定および非CDBの結果を示しています。

表11-1 非CDBのFORCE LOGGING設定の優先順位

データベース	表領域	データベース・オブジェクトのLOGGING属性	結果
FORCE LOGGING	無視	無視	ログ
NO FORCE LOGGING	FORCE LOGGING	無視	ログ

データベース	表領域	データベース・オブジェクトのLOGGING属性	結果
NO FORCE LOGGING	NO FORCE LOGGING	LOGGING	ログ
NO FORCE LOGGING	NO FORCE LOGGING	NOLOGGING	ログされない

次の表は、各レベルのログ設定およびCDBの結果を示しています。

表11-2 CDBのFORCE LOGGING設定の優先順位

CDB	PDB	表領域	データベース・オブジェクトのLOGGING属性	結果
FORCE LOGGING	無視	無視	無視	ログ
NO FORCE LOGGING	ENABLE FORCE LOGGING	無視	無視	ログ
NO FORCE LOGGING	ENABLE FORCE NOLOGGING	無視	無視	ログされない
NO FORCE LOGGING	DISABLE FORCE [NO]LOGGING (設定なし)	FORCE LOGGING	無視	ログ
NO FORCE LOGGING	DISABLE FORCE [NO]LOGGING (設定なし)	NO FORCE LOGGING	LOGGING	ログ
NO FORCE LOGGING	DISABLE FORCE [NO]LOGGING (設定なし)	NO FORCE LOGGING	NOLOGGING	ログされない

親トピック: REDOログの管理

11.10 REDOログのデータ・ディクショナリ・ビュー

REDOログに関する情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

次のビューには、REDOログに関する情報が表示されます。

ビュー	説明
V$LOG	制御ファイルの REDO ログ・ファイル情報が表示されます。
V$LOGFILE	REDO ログ・グループとメンバーおよびメンバーの状態を識別します。
V$LOG_HISTORY	ログの履歴情報が含まれます。

次の問合せは、データベースのREDOログに関する制御ファイル情報を返します。

```
SELECT GROUP#, THREAD#, SEQUENCE#, BYTES, MEMBERS, ARCHIVED,
       STATUS, FIRST_CHANGE#, FIRST_TIME
  FROM V$LOG;
GROUP# THREAD#   SEQ   BYTES  MEMBERS  ARC STATUS     FIRST_CHANGE# FIRST_TIM
------ ------- ----- -------  -------  --- ---------  ------------- ---------
     1       1 10605 1048576        1  YES ACTIVE          11515628 16-APR-00
     2       1 10606 1048576        1  NO  CURRENT         11517595 16-APR-00
     3       1 10603 1048576        1  YES INACTIVE        11511666 16-APR-00
```

```
     4       1 10604 1048576          1  YES INACTIVE          11513647 16-APR-00
```

グループのすべてのメンバーの名前を表示するには、次の問合せを使用します。

```
SELECT GROUP#, STATUS, MEMBER FROM V$LOGFILE;
GROUP#   STATUS  MEMBER
------  -------  ----------------------------------
     1           D:¥ORANT¥ORADATA¥IDDB2¥REDO04.LOG
     2           D:¥ORANT¥ORADATA¥IDDB2¥REDO03.LOG
     3           D:¥ORANT¥ORADATA¥IDDB2¥REDO02.LOG
     4           D:¥ORANT¥ORADATA¥IDDB2¥REDO01.LOG
```

メンバーのSTATUSが空白の場合、そのファイルは使用中です。

親トピック: REDOログの管理

12 アーカイブREDOログ・ファイルの管理

アーカイブREDOログ・ファイルは、NOARCHIVELOGまたはARCHIVELOGモードの選択やアーカイブ先の指定などのタスクを実行することで管理します。

- アーカイブREDOログの概要
  Oracle Databaseでは、書込み済のREDOログ・ファイル・グループを、アーカイブREDOログと総称される1つ以上のオフラインの保存先に保存できます。
- NOARCHIVELOGモードとARCHIVELOGモードの選択
  データベースをNOARCHIVELOGモードとARCHIVELOGモードのどちらで実行するかを選択する必要があります。
- アーカイブの制御
  データベースのアーカイブ・モードを設定し、アーカイブ・プロセス数を調整できます。
- アーカイブ先の指定
  REDOログをアーカイブする前に、アーカイブ先を指定し、アーカイブ先の様々な状態を理解する必要があります。
- ログ転送モードについて
  アーカイブ・ログをアーカイブ先に転送する場合のモードには、ノーマル・アーカイブ転送およびスタンバイ転送という2つのモードがあります。ノーマル転送では、ファイルはローカル・ディスクに転送されます。スタンバイ転送では、ファイルはネットワークを介してローカルまたはリモートのスタンバイ・データベースに転送されます。
- アーカイブ先の障害管理
  アーカイブ先で発生した障害が、自動アーカイブ・モードで操作している場合のエラー原因となることがあります。Oracle Databaseには、アーカイブ先の障害に関連する問題を最小限に抑えるためのプロシージャが用意されています。
- ARCHIVELOGプロセスによって生成されるトレース出力の制御
  バックグラウンド・プロセスは適宜、トレース・ファイルに情報を書き込みます。ARCHIVELOGプロセスの場合は、トレース・ファイルに書き込む出力を制御できます。
- アーカイブREDOログに関する情報の表示
  アーカイブREDOログに関する情報を表示するには、動的パフォーマンス・ビューまたはARCHIVE LOG LISTコマンドを使用します。

関連項目:

- Oracle Databaseサーバーによって作成および管理されるアーカイブREDOログの作成方法は、「Oracle Managed Filesの使用」を参照してください
- Oracle Real Application Clusters環境でのアーカイブに固有の情報については、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

親トピック: Oracle Databaseの構造と記憶域

12.1 アーカイブREDOログの概要

Oracle Databaseでは、書込み済のREDOログ・ファイル・グループを、アーカイブREDOログと総称される1つ以上のオフラインの保存先に保存できます。

REDOログ・ファイルをアーカイブREDOログ・ファイルに変更するプロセスは、アーカイブと呼ばれます。このプロセスを実行できるのは、データベースがARCHIVELOGモードで動作している場合のみです。自動または手動アーカイブを選択できます。

アーカイブREDOログ・ファイルは、REDOログ・グループの書込み済メンバーのいずれかのコピーです。REDOログ・グループの同一メンバーのREDOエントリ、および一意のログ順序番号が含まれています。たとえば、REDOログを多重化しており、グループ1に同一のメンバー・ファイルa_log1とb_log1が含まれている場合、アーカイバ・プロセス(ARCn)によってこれらのメンバー・ファイルのうちの1つがアーカイブされます。a_log1が破損した場合でも、ARCnは同一のb_log1をアーカイブできます。アーカイブREDOログには、アーカイブを有効にした後に作成されたすべてのグループのコピーが含まれます。

データベースがARCHIVELOGモードで稼働しているときは、REDOログ・グループがアーカイブされないかぎり、ログ・ライター・プロセス(LGWR)はREDOログ・グループを再利用(上書き)できません。自動アーカイブが使用可能な場合は、バックグラウンド・プロセスARCnによってアーカイブ操作が自動的に実行されます。データベースは必要に応じて複数のアーカイバ・プロセスを起動して、一杯になったREDOログのアーカイブが遅れないようにします。

アーカイブREDOログ・ファイルは、次の目的に使用できます。

- データベースのリカバリ
- スタンバイ・データベースの更新
- LogMinerユーティリティを使用したデータベースの履歴情報の取得

関連項目:

アーカイブREDOログ・ファイルの使用方法は、次のマニュアルを参照してください。

- Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド
- スタンバイ・データベースの設定とメンテナンスは、『Oracle Data Guard概要および管理』を参照してください。
- LogMinerのPL/SQLパッケージの使用方法は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: アーカイブREDOログ・ファイルの管理

12.2 NOARCHIVELOGモードとARCHIVELOGモードの選択

データベースをNOARCHIVELOGモードとARCHIVELOGモードのどちらで実行するかを選択する必要があります。

一杯になったREDOログ・ファイル・グループをアーカイブ可能にするかどうかは、データベース上で実行されているアプリケーションの可用性と信頼性の要件によって決まります。ディスク障害の発生時にもデータベース内のデータが失われないようにする場合は、ARCHIVELOGモードを使用します。一杯になったREDOログ・ファイルをアーカイブすると、管理作業が増えます。

- NOARCHIVELOGモードによるデータベースの実行
  データベースをNOARCHIVELOGモードで実行すると、REDOログはアーカイブされません。
- ARCHIVELOGモードによるデータベースの実行
  データベースをARCHIVELOGモードで実行する場合は、REDOログのアーカイブを使用可能にします。

親トピック: アーカイブREDOログ・ファイルの管理

12.2.1 NOARCHIVELOGモードによるデータベースの実行

データベースをNOARCHIVELOGモードで実行すると、REDOログはアーカイブされません。

データベースの制御ファイルは、グループが一杯になってもアーカイブする必要がないことを示します。したがって、ログ・スイッチが発生して、一杯になったグループがアクティブでなくなると、そのグループはLGWRで再利用できるようになります。

NOARCHIVELOGモードでは、データベースはインスタンス障害からは保護されますが、メディア障害からは保護されません。オンラインREDOログ・グループに格納されているデータベースへの最新の変更のみをインスタンスのリカバリに使用できます。データベースがNOARCHIVELOGモードのときにメディア障害が発生した場合、最後にデータベース全体のバックアップを行った時点までのデータベースをリストアできます。そのバックアップ以降のトランザクションはリカバリできません。

NOARCHIVELOGモードでは、オンラインでの表領域のバックアップを実行できず、また、データベースがARCHIVELOGモードだった際に作成したオンラインの表領域バックアップも使用できません。NOARCHIVELOGモードで運用しているデータベースをリストアする場合、使用できるバックアップは、データベースのクローズ中に作成したデータベース全体のバックアップのみです。そのため、データベースをNOARCHIVELOGモードで稼働する場合は、定期的かつ頻繁にデータベース全体のバックアップを実行する必要があります。

親トピック: NOARCHIVELOGモードとARCHIVELOGモードの選択

12.2.2 ARCHIVELOGモードによるデータベースの実行

データベースをARCHIVELOGモードで実行する場合は、REDOログのアーカイブを使用可能にします。

データベースの制御ファイルは、一杯になったREDOログ・ファイルのグループがアーカイブされるまでは、LGWRでこのグループを再使用できないことを示します。一杯になったグループは、ログ・スイッチの発生直後からアーカイブに使用できます。

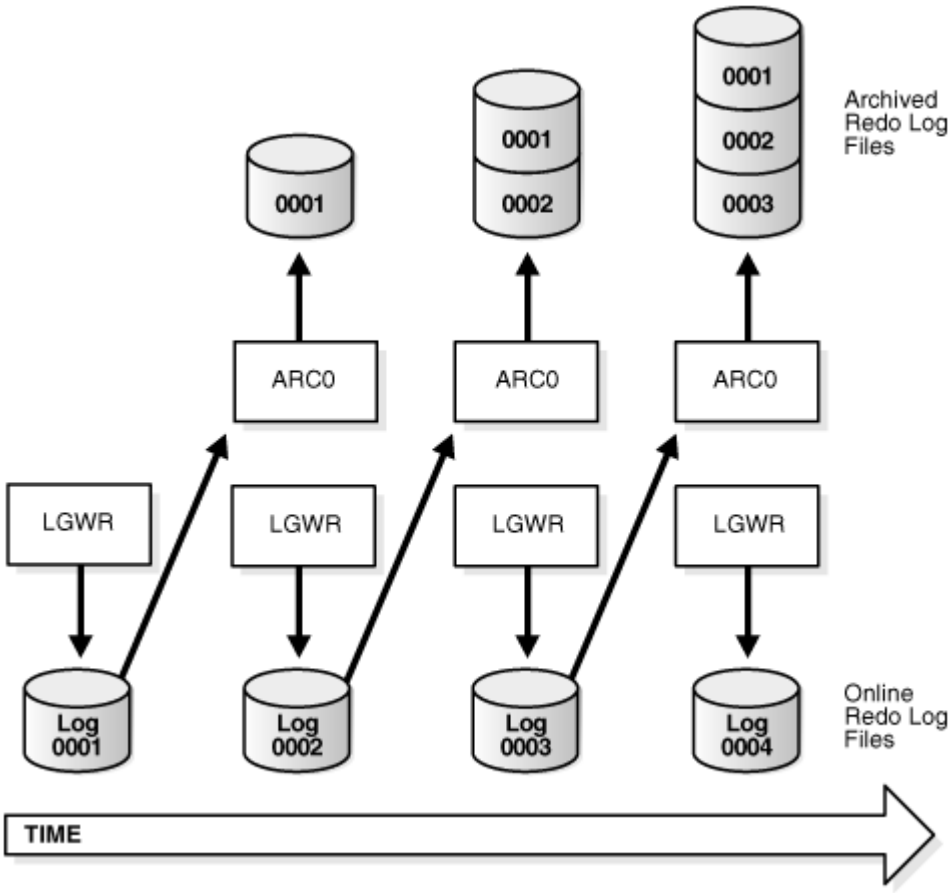
一杯になったグループのアーカイブには、次のような利点があります。

- データベースのバックアップ、オンラインREDOログおよびアーカイブREDOログ・ファイルが揃っていると、オペレーティング・システムやディスクに障害が発生しても、コミットされたすべてのトランザクションをリカバリできることが保証されます。
- アーカイブ・ログを使用可能にしておく場合、データベースがオープンされていて、システムが正常に使用できるときに実行したバックアップを使用できます。
- スタンバイに、元のデータベースのアーカイブREDOログ・ファイルを継続的に適用することにより、スタンバイ・データベースを元のデータベースに対して最新の状態に保つことができます。

一杯になったREDOログ・ファイルを自動的にアーカイブするようにインスタンスを構成する方法と、手動でアーカイブする方法があります。通常は、自動アーカイブの方が便利で効率的です。図12-1は、アーカイバ・プロセス(この図ではARC0)によって、一杯になったREDOログ・ファイルがデータベースのアーカイブREDOログに書き込まれる過程を示しています。

分散データベース内のデータベースをすべてARCHIVELOGモードで操作している場合は、調整式分散データベース・リカバリを実行できます。ただし、分散データベース内のデータベースのいずれかがNOARCHIVELOGモードで操作されている場合、(すべてのデータベースの整合性を維持するために)グローバルな分散データベースのリカバリは、NOARCHIVELOGモードで操作しているデータベース全体の最新のバックアップによって制限されます。

図12-1 ARCHIVELOGモードでのREDOログ・ファイルの使用



ヒント:

アーカイブ REDO ログ・ファイルとそれに対応するデータベース・バックアップは、ローカル・ディスクからテープなどの永続的なオフライン記憶メディアに移動しておくことをお薦めします。アーカイブ・ログは主としてデータベース・リカバリに使用されるため、プライマリ・データベースに障害が発生した場合でも、これらのログが安全であることを保証する必要があります。

親トピック: NOARCHIVELOGモードとARCHIVELOGモードの選択

12.3 アーカイブの制御

データベースのアーカイブ・モードを設定し、アーカイブ・プロセス数を調整できます。

- 初期データベース・アーカイブ・モードの設定
  初期のアーカイブ・モードは、CREATE DATABASE文でデータベース作成の一部として設定します。
- データベース・アーカイブ・モードの変更
  データベース・アーカイブ・モードを変更するには、ALTER DATABASE文でARCHIVELOG句またはNOARCHIVELOG句を使用します。
- 手動アーカイブの実行
  通常は、自動アーカイブの方が便利で効率的です。ただし、データベースの構成は手動アーカイブでのみ実行できます。
- アーカイバ・プロセス数の調整
  LOG_ARCHIVE_MAX_PROCESSES初期化パラメータは、データベースが最初に起動するARCnプロセスの数を指定します。デフォルトのプロセス数は4です。

関連項目:

アーカイブ・モードの制御に関する追加情報は、オペレーティング・システム固有のOracleマニュアルを参照してください

親トピック: アーカイブREDOログ・ファイルの管理

12.3.1 初期データベース・アーカイブ・モードの設定

初期のアーカイブ・モードは、CREATE DATABASE文でデータベース作成の一部として設定します。

多くの場合、このプロセスで生成されるREDO情報はアーカイブする必要がないため、データベース作成時にはNOARCHIVELOGモード(デフォルト)を使用できます。初期のアーカイブ・モードを変更するかどうかは、データベースの作成後に決定します。

ARCHIVELOGモードを指定した場合は、アーカイブREDOログ・ファイルのアーカイブ先を初期化パラメータで指定する必要があります(「アーカイブ先の初期化パラメータの設定」を参照)。

親トピック: アーカイブの制御

12.3.2 データベース・アーカイブ・モードの変更

データベース・アーカイブ・モードを変更するには、ALTER DATABASE文でARCHIVELOG句またはNOARCHIVELOG句を使用します。

また、データベース・アーカイブ・モードを変更するには、管理者権限(AS SYSDBA)でデータベースに接続する必要があります。

次のステップは、データベース・アーカイブ・モードをNOARCHIVELOGからARCHIVELOGに切り替えます。

1. データベース・インスタンスを停止します。

   ```
   SHUTDOWN IMMEDIATE
   ```

   データベースがオープンされている場合は、アーカイブ・モードを切り替える前にクローズし、対応するインスタンスを停止する必要があります。メディア・リカバリを必要とするデータファイルがある場合は、モードをARCHIVELOGからNOARCHIVELOGに変更できません。

2. データベースをバックアップします。

   データベースに重要な変更をする前に、データベースのデータを保護するため必ずバックアップを作成してください。このバックアップは、NOARCHIVELOGモードでのデータベースの最終バックアップとなり、ARCHIVELOGモードへの切替え中に問題が生じた場合に使用できます。データベース・バックアップの作成については、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

3. 初期化パラメータ・ファイルを編集して、アーカイブREDOログ・ファイルのアーカイブ先を初期化パラメータで指定します(「アーカイブ先の初期化パラメータの設定」を参照)。
4. 新しいインスタンスを起動し、データベースをマウントします。オープンはしません。

   ```
   STARTUP MOUNT
   ```

   アーカイブを使用可能または使用禁止にするには、データベースをマウントして、オープンしないようにする必要があります。

5. データベース・アーカイブ・モードを変更します。通常の操作を実行するためにデータベースをオープンします。

   ```
   ALTER DATABASE ARCHIVELOG;
   ALTER DATABASE OPEN;
   ```

6. データベースを停止します。

```
SHUTDOWN IMMEDIATE
```

7. データベースをバックアップします。

   データベースのアーカイブ・モードを変更すると、制御ファイルが更新されます。変更後は、すべてのデータベース・ファイルと制御ファイルのバックアップを作成する必要があります。以前のバックアップはNOARCHIVELOGモードで作成されているため、使用できなくなります。

   関連項目:

   Real Application Clusters使用時のアーカイブ・モード切替えの詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください

親トピック: アーカイブの制御

12.3.3 手動アーカイブの実行

通常は、自動アーカイブの方が便利で効率的です。ただし、データベースの構成は手動アーカイブでのみ実行できます。

データベースを手動ARCHIVELOGモードで操作している場合は、一杯になったREDOログ・ファイルの非アクティブ・グループをアーカイブしないと、データベース操作が一時的に停止する可能性があります。

データベースを手動アーカイブ・モードで操作するには:

1. 「データベース・アーカイブ・モードの変更」で説明している手順に従いますが、ALTER DATABASE文は次の文で置換します。

```
ALTER DATABASE ARCHIVELOG MANUAL;
```

2. 管理者権限を持つユーザーとして、データベースに接続します。
3. データベースがマウントされているか、オープンしていることを確認します。
4. 手動アーカイブを実行するには、ALTER SYSTEM文でARCHIVE LOG句を指定します。たとえば次の文は、アーカイブされていないREDOログ・ファイルをすべてアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG ALL;
```

手動アーカイブ・モードを使用している場合、アーカイブ先にスタンバイ・データベースは指定できません。

また、自動アーカイブが使用可能な場合でも、一杯になったREDOログ・メンバーの非アクティブ・グループを別の位置に再度アーカイブする場合などに手動アーカイブを使用できます。この場合は、手動アーカイブが完了していなくてもインスタンスではREDOログ・グループを再利用できるため、ファイルが上書きされる場合があります。このような場合は、アラート・ログにエラー・メッセージが書き込まれます。

関連トピック

- ARCHIVELOGモードによるデータベースの実行

親トピック: アーカイブの制御

12.3.4 アーカイバ・プロセス数の調整

LOG_ARCHIVE_MAX_PROCESSES初期化パラメータは、データベースが最初に起動するARCnプロセスの数を指定します。デフォルトのプロセス数は4です。

追加のARCnプロセスの開始の実行時オーバーヘッドを回避するには:

- LOG_ARCHIVE_MAX_PROCESSES初期化パラメータを設定して、インスタンス起動時に最大30のARCnプロセスを開始することを指定します。

LOG_ARCHIVE_MAX_PROCESSESパラメータは動的で、ALTER SYSTEM 文を使用して変更できます。

次の文は、データベースが起動時に6つのARCnプロセスを起動するようにデータベースを構成します。

```
ALTER SYSTEM SET LOG_ARCHIVE_MAX_PROCESSES=6;
```

また、この文は現在実行中のインスタンスにただちに影響します。ここでは現在実行されているARCnプロセスの数を増加または減少させて6にします。

親トピック: アーカイブの制御

12.4 アーカイブ先の指定

REDOログをアーカイブする前に、アーカイブ先を指定し、アーカイブ先の様々な状態を理解する必要があります。

「アーカイブREDOログに関する情報の表示」に示す動的パフォーマンス・ビュー(V$)を使用して、必要なすべてのアーカイブ情報を参照できます。

- アーカイブ先の初期化パラメータの設定
  REDOログのアーカイブ先を単一の場所にするか、または複数の場所にするかを選択できます。
- ログ・アーカイブ先グループでの代替アーカイブ先の拡張
  ログ・アーカイブ先グループを使用して、代替アーカイブ先の数を拡張できます。
- アーカイブ先の状態の理解
  複数の変数によって、アーカイブ先のステータスが決定されます。
- 代替アーカイブ先の指定
  別のアーカイブ先に障害が発生した場合にのみ、ある場所がアーカイブ先になるように指定する場合は、その場所を代替アーカイブ先にできます。ローカルとリモート両方のアーカイブ先を代替アーカイブ先にできます。

親トピック: アーカイブREDOログ・ファイルの管理

12.4.1 アーカイブ先の初期化パラメータの設定

REDOログのアーカイブ先を単一の場所にするか、または複数の場所にするかを選択できます。

アーカイブ先は、ローカル(ローカル・ファイル・システムまたはOracle Automatic Storage Management(Oracle ASM)ディスク・グループ内)またはリモート(スタンバイ・データベース上)を選択できます。複数の場所にアーカイブする場合は、一杯になった各REDOログ・ファイルのコピーが各アーカイブ先に書き込まれます。これらの冗長コピーは、アーカイブ先のいずれかで障害が発生した場合でもアーカイブ・ログを常に使用可能にするのに役立ちます。

単一のアーカイブ先にのみアーカイブするには:

- LOG_ARCHIVE_DEST初期化パラメータを使用してアーカイブ先を指定します。

複数のアーカイブ先にアーカイブするには:

- LOG_ARCHIVE_DEST_n初期化パラメータを使用して2箇所以上の場所にアーカイブするか、またはLOG_ARCHIVE_DESTおよびLOG_ARCHIVE_DUPLEX_DEST初期化パラメータを使用してプライマリおよびセカンダリのアーカイブ先にのみアーカイブすることを選択します。

ローカル・アーカイブ先の場合、ローカル・ファイル・システムまたはOracle ASMディスク・グループ以外に、高速リカバリ領域にアーカイブできます。データベースは高速リカバリ領域を使用して、バックアップおよびリカバリに関連する様々なファイルを格納し、それらのディスク領域を自動的に管理します。高速リカバリ領域の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

通常、アーカイブ・ログのアーカイブ先はデータベース計画で決定し、データベースのインストール時にアーカイブ先の初期化パラメータを設定します。ただし、ALTER SYSTEMコマンドを使用して、データベースの実行後にアーカイブ先を動的に追加または変更できます。アーカイブ先の変更は、次のログ・スイッチ(自動または手動)で有効になります。

次の表にアーカイブ先の選択肢の要約を示します。この後の各項で、詳細を説明します。

方法	初期化パラメータ	ホスト	例
1	LOG_ARCHIVE_DEST_n ここで: n は、1 から 31 の整数です。アーカイブ先の 1 から 10 はローカルまたはリモートの場所として使用可能です。11 から 31 はリモートのアーカイブ先としてのみ使用可能です。	ローカル/リモート	LOG_ARCHIVE_DEST_1 = 'LOCATION = /disk1/arc' LOG_ARCHIVE_DEST_2 = 'LOCATION=/disk2/arc' LOG_ARCHIVE_DEST_3 = 'SERVICE=standby1'
2	LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST	ローカルのみ	LOG_ARCHIVE_DEST = '/disk1/arc' LOG_ARCHIVE_DUPLEX_DEST ='/disk2/arc'

- 方法1: LOG_ARCHIVE_DEST_nパラメータの使用
  LOG_ARCHIVE_DEST_n初期化パラメータを使用して、アーカイブ・ログの別のアーカイブ先を指定できます。
- 方法2: LOG_ARCHIVE_DESTおよびLOG_ARCHIVE_DUPLEX_DESTの使用
  最大2つのアーカイブ先ディレクトリを指定するには、LOG_ARCHIVE_DESTパラメータを使用してプライマリ・アーカイブ先を指定し、必要に応じてLOG_ARCHIVE_DUPLEX_DESTでセカンダリ・アーカイブ先を指定します。

親トピック: アーカイブ先の指定

12.4.1.1 方法1: LOG_ARCHIVE_DEST_nパラメータの使用

LOG_ARCHIVE_DEST_n初期化パラメータを使用して、アーカイブ・ログの別のアーカイブ先を指定できます。

LOG_ARCHIVE_DEST_n初期化パラメータ(nは1から31の整数)を設定して、1から31を指定します。末尾に番号が付いた各パラメータによって、特定のアーカイブ先を一意に識別します。

LOG_ARCHIVE_DEST_nの位置は、次の表に示すキーワードを使用して指定します。

キーワード	意味	例

キーワード	意味	例
LOCATION	ローカル・ファイル・システムの位置または Oracle ASM ディスク・グループ	LOG_ARCHIVE_DEST_n = 'LOCATION=/disk1/arc' LOG_ARCHIVE_DEST_n = 'LOCATION=+DGROUP1/orcl/arc_1'
LOCATION	高速リカバリ領域	LOG_ARCHIVE_DEST_n = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
SERVICE	Oracle Net のサービス名を介したリモート・アーカイブ。	LOG_ARCHIVE_DEST_n = 'SERVICE=standby1'

LOCATIONキーワードを使用する場合は、次のいずれかを指定します。

- オペレーティング・システムのローカル・ファイル・システムの有効なパス名
- Oracle ASMディスク・グループ
- 高速リカバリ領域を示すキーワードUSE_DB_RECOVERY_FILE_DEST

SERVICEを指定した場合、Oracle Netがスタンバイ・データベースの接続記述子に解決できるネット・サービス名を提供します。この接続記述子には、リモート・データベースへの接続に必要な情報が含まれています。

LOG_ARCHIVE_DEST_n初期化パラメータを使用してアーカイブREDOログ・ファイルのアーカイブ先を設定するステップは、次のとおりです。

1. LOG_ARCHIVE_DEST_n初期化パラメータを設定し、1から31のアーカイブ先を指定します。たとえば、次のように入力します。

   ```
   LOG_ARCHIVE_DEST_1 = 'LOCATION = /disk1/archive'
   LOG_ARCHIVE_DEST_2 = 'LOCATION = /disk2/archive'
   LOG_ARCHIVE_DEST_3 = 'LOCATION = +RECOVERY/orcl/arc_3'
   ```

   スタンバイ・データベースにアーカイブする場合は、SERVICEキーワードを使用して有効なネット・サービス名を指定します。たとえば、次のように入力します。

   ```
   LOG_ARCHIVE_DEST_4 = 'SERVICE = standby1'
   ```

2. (オプション)スレッド番号をファイル名の一部に含めるために%tを、ログ順序番号を含めるために%sを、RESETLOGS ID (ub4で表されるタイムスタンプ値)を含めるために%rを使用して、LOG_ARCHIVE_FORMAT初期化パラメータを設定します。ファイル名の左に0(ゼロ)を埋め込むには、大文字(%T、%Sおよび%R)を使用します。

   ノート:

   LOG_ARCHIVE_FORMAT パラメータを指定する場合は、データベースでリセットログ ID (%r)を指定する必要があります。このパラメータのデフォルトは、オペレーティング・システムによって異なります。

   データベースのインカネーションは、RESETLOGS オプションを指定してデータベースをオープンすると変更されます。%r を指定すると、アーカイブ REDO ログ・ファイル名からリセットログ ID が取得されます。このリカバリ方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

次に、LOG_ARCHIVE_FORMATの設定例を示します。

```
LOG_ARCHIVE_FORMAT = arch_%t_%s_%r.arc
```

この設定では、スレッド1、ログ順序番号100、101および102、リセットログID 509210197について次のようなアーカイブ・ログが生成されます。リセットログIDが同一の場合は、すべてのファイルが同じデータベース・インカネーションに含まれることを示します。

```
/disk1/archive/arch_1_100_509210197.arc,
/disk1/archive/arch_1_101_509210197.arc,
/disk1/archive/arch_1_102_509210197.arc
/disk2/archive/arch_1_100_509210197.arc,
/disk2/archive/arch_1_101_509210197.arc,
/disk2/archive/arch_1_102_509210197.arc
/disk3/archive/arch_1_100_509210197.arc,
/disk3/archive/arch_1_101_509210197.arc,
/disk3/archive/arch_1_102_509210197.arc
```

LOG_ARCHIVE_FORMAT初期化パラメータは無視される場合もあります。このパラメータ詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: アーカイブ先の初期化パラメータの設定

12.4.1.2 方法2: LOG_ARCHIVE_DESTおよびLOG_ARCHIVE_DUPLEX_DESTの使用

最大2つのアーカイブ先ディレクトリを指定するには、LOG_ARCHIVE_DESTパラメータを使用して1次アーカイブ先を指定し、必要に応じてLOG_ARCHIVE_DUPLEX_DESTで2次アーカイブ先を指定します。

アーカイブ先は、ローカルである必要があります。データベースでは、REDOログはどちらかのパラメータで指定したすべてのアーカイブ先ディレクトリにアーカイブされます。

方法2を使用するステップは、次のとおりです。

1. LOG_ARCHIVE_DESTおよびLOG_ARCHIVE_DUPLEX_DESTパラメータにアーカイブ先を指定します(ALTER SYSTEM文を使用して、LOG_ARCHIVE_DUPLEX_DESTを動的に指定することもできます)。たとえば、次のように入力します。

```
LOG_ARCHIVE_DEST = '/disk1/archive'
LOG_ARCHIVE_DUPLEX_DEST = '/disk2/archive'
```

2. 方法1のステップ2で説明したように、LOG_ARCHIVE_FORMAT初期化パラメータを設定します。

ノート:

高速リカバリ領域を構成し(DB_RECOVERY_FILE_DEST および DB_RECOVERY_FILE_DEST_SIZE パラメータを設定)、ローカル・アーカイブ先を指定しない場合、データベースによって自動的に高速リカバリ領域はローカル・アーカイブ先として選択され、LOG_ARCHIVE_DEST_1 は USE_DB_RECOVERY_FILE_DEST に設定されます。

警告:

アーカイブ・ログのアーカイブ先に常に十分なディスク領域があることを確認する必要があります。データベースがログ・ファイルをアーカイブする際にディスクが一杯であるというエラーが発生すると、致命的なエラーとなり、データベースが

応答しなくなります。アラート・ログでディスクが一杯であるというメッセージを確認できます。

関連項目:

- REDOログのアーカイブ制御に使用される初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください
- スタンバイ・アーカイブ先の指定に使用するLOG_ARCHIVE_DEST_n初期化パラメータの使用方法は、『Oracle Data Guard概要および管理』を参照してください。この初期化パラメータには他にも指定できるキーワードがありますが、このマニュアルでは説明されていません。
- ネット・サービス名および接続記述子についての説明は、『Oracle Database Net Services管理者ガイド』を参照してください。
- 高速リカバリ領域の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: アーカイブ先の初期化パラメータの設定

12.4.2 ログ・アーカイブ先グループを使用した代替アーカイブ先の拡張

ログ・アーカイブ先グループを使用して、代替アーカイブ先の数を拡張できます。

- ログ・アーカイブ先グループについて
  ログ・アーカイブ先グループで複数のアーカイブ先を指定し、グループ内のアーカイブ先に優先順位を付けることができます。複数のグループを指定して、データベースに対して可能なアーカイブ先の数を拡張できます。
- ログ・アーカイブ先グループの指定
  LOG_ARCHIVE_DEST_n初期化パラメータのGROUP属性を使用して、ログ・アーカイブ先グループを指定します。

親トピック: アーカイブ先の指定

12.4.2.1 ログ・アーカイブ先グループについて

ログ・アーカイブ先グループで複数のアーカイブ先を指定し、グループ内のアーカイブ先に優先順位を付けることができます。複数のグループを指定して、データベースに対して可能なアーカイブ先の数を拡張できます。

ログ・アーカイブ先グループを指定するには、LOG_ARCHIVE_DEST_n初期化パラメータのGROUP属性を使用します。1つのグループに最大30のログ・アーカイブ先を含めることができます。各グループの1つのメンバーがアクティブであり、他のメンバーはアーカイブ先に障害が発生した場合に使用できます。アーカイブ先が非アクティブになった場合、Oracle Databaseはグループ内に使用可能なアーカイブ先が1つ以上あれば、そのアーカイブ先に切り替えます。PRIORITY属性でアーカイブ先に優先順位を付けることで、最初に使用するアーカイブ先を指定できます。

ログ・アーカイブ先グループは、グループの作成時に指定されるグループ番号で参照されます。最大8つのグループを使用できます。グループ内のREDOデータをアーカイブする場所を指定するには、すべてのログ・アーカイブの保存先にSERVICE属性を指定する必要があります。

グループ内のアーカイブ先に優先順位を付けるには、アーカイブ先のPRIORITY属性を1から8の範囲の整数に設定します。数値が小さいほど優先度が高くなります。優先順位によって、データベースがマウントされたとき、またはアクティブなアーカイブ先に障害が発生したときにグループ内のどのアーカイブ先がアクティブになるかが決まります。たとえば、2のPRIORITY値は7のPRIORITY値よりも優先順位が高くなります。したがって、グループ内の現在アクティブでPRIORITY値が1のアーカイブ先が非アクティブになった場合、PRIORITY値が2のアーカイブ先は、PRIORITY値が7のアーカイブ先よりも前に使用されます。

PRIORITY属性がアーカイブ先に設定されていない場合、デフォルト値は1です。

優先順位は、以前に失敗したアーカイブ先が使用可能になったときにも考慮されます。アクティブなアーカイブ先が失敗し、Oracle Databaseがより低い優先順位のアーカイブ先に切り替えた場合、Oracle Databaseはより高い優先順位のアーカイブ先が再度使用可能になったときに、そのアーカイブ先に切り替えます。たとえば、優先順位が1のアクティブなアーカイブ先が非アクティブになり、Oracle Databaseが優先順位2のアーカイブ先に切り替えた場合、Oracle Databaseは、優先順位2のアーカイブ先に障害が発生していなくても、優先順位1のアーカイブ先が再度使用可能になったときに、そのアーカイブ先に再度切り替えます。

ただし、同じグループに割り当てられた複数のアーカイブ先が同じ優先順位を持つことがあります。たとえば、優先順位が1のアーカイブ先が3つある場合があります。このようなグループでは、アーカイブ先の障害により、同じ優先順位の別のメンバーに切り替えられます。この場合、両方のアーカイブ先が同じ優先順位を持つため、元のアーカイブ先が再度使用可能になったときに、そのアーカイブ先への切替えは行われません。最初のアーカイブ先が再度使用可能になった後で2番目のアーカイブ先に障害が発生した場合、データベースは最初のアーカイブ先またはグループ内の同じ優先順位を持つ別のアーカイブ先に切り替えます。

関連項目:

『Oracle Data Guard概要および管理』

親トピック: ログ・アーカイブ先グループを使用した代替アーカイブ先の拡張

12.4.2.2 ログ・アーカイブ先グループの指定

LOG_ARCHIVE_DEST_n初期化パラメータのGROUP属性を使用して、ログ・アーカイブ先グループを指定します。

最大8つのログ・アーカイブ先グループを作成でき、各グループには最大30のアーカイブ先を指定できます。

ログ・アーカイブ先グループを指定するには、データベースをARCHIVELOGモードで実行する必要があります。

- LOG_ARCHIVE_DEST_n初期化パラメータを設定し、GROUP属性を含めてログ・アーカイブ先グループを指定します。

  オプションで、PRIORITY属性を含めて、システムの起動時またはアーカイブ先の障害発生時にアクティブにするグループ内のログ・アーカイブ先を指定します。

例12-1 2つのログ・アーカイブ先グループの指定

この例では、2つのログ・アーカイブ先グループ(1および2)を指定します。各グループには3つのログ・アーカイブ先が指定されています。

```
LOG_ARCHIVE_DEST_1 = 'SERVICE=SITEa VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) GROUP=1'
LOG_ARCHIVE_DEST_2 = 'SERVICE=SITEb VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) GROUP=1'
LOG_ARCHIVE_DEST_3 = 'SERVICE=SITEc VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) GROUP=1'
LOG_ARCHIVE_DEST_4 = 'SERVICE=SITE1 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) GROUP=2'
LOG_ARCHIVE_DEST_5 = 'SERVICE=SITE2 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) GROUP=2'
LOG_ARCHIVE_DEST_6 = 'SERVICE=SITE3 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) GROUP=2'
```

例12-2 ログ・アーカイブ先グループ内での優先順位の指定

この例では、単一のログ・アーカイブ先グループ内のアーカイブ先に異なる優先順位レベルを指定します。具体的には、アーカイブ先1および2はどちらも優先順位レベル1で、アーカイブ先3は優先順位レベル2、アーカイブ先4は優先順位レベル3です。

```
LOG_ARCHIVE_DEST_1 = 'SERVICE=SITE1 SYNC VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
GROUP=1 PRIORITY=1'
LOG_ARCHIVE_DEST_2 = 'SERVICE=SITE2 SYNC VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
GROUP=1 PRIORITY=1'
LOG_ARCHIVE_DEST_3 = 'SERVICE=SITE3 ASYNC VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
GROUP=1 PRIORITY=2'
```

```
LOG_ARCHIVE_DEST_4 = 'SERVICE=SITE4 ASYNC VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
GROUP=1 PRIORITY=3'
```

この例で、サイト1、2および3はREDOを転送するだけのOracle Data Guard遠隔同期インスタンス、およびサイト4は実際のリモート・スタンバイ・データベースにすることができます。または、サイト1、2、3および4のすべてをスタンバイ・データベースにして、それらがアクティブな宛先であるときにREDOを他のサイトにカスケードするように構成できます。

次の優先順位のルールに従って処理が行われます。

- アーカイブ先1とアーカイブ先2はどちらも優先順位レベル1であるため、このどちらかをデフォルトのアクティブなアーカイブにすることができます。
- アーカイブ先1がアクティブで、その後使用不能になった場合、Oracle Databaseはアーカイブ先2に切り替えます。同様に、アーカイブ先2がアクティブで、その後使用不能になった場合、Oracle Databaseはアーカイブ先1に切り替えます。アーカイブ先1または2がどちらも使用可能な場合、その一方が使用されます。
- アーカイブ先1とアーカイブ先2の両方が使用不能になった場合は、アーカイブ先3が使用されます。
- アーカイブ先3がアクティブなときに、アーカイブ先1またはアーカイブ先2が使用可能になった場合、Oracle Databaseは使用可能な優先順位1のアーカイブ先に切り替えます。
- アーカイブ先1、2および3がすべて使用不能になった場合は、アーカイブ先4が使用されます。
- アーカイブ先4がアクティブなときにアーカイブ先1、2または3が使用可能になった場合、Oracle Databaseはまず使用可能な優先順位1のアーカイブ先に切り替えて、次に、使用可能な優先順位2のアーカイブ先に切り替えます。

関連項目:

『Oracle Data Guard概要および管理』

親トピック: ログ・アーカイブ先グループを使用した代替アーカイブ先の拡張

12.4.3 アーカイブ先の状態の理解

複数の変数によって、アーカイブ先のステータスが決定されます。

各アーカイブ先は次のような可変特性を持っており、これらの特性によってその状態が決まります。

- Valid/Invalid: ディスクの位置またはサービス名情報が指定されているかどうか、およびそれらが有効かどうかを示します。
- Enabled/Disabled: 位置の使用可能状態と、データベースがアーカイブ先を使用できるかどうかを示します。
- Active/Inactive: アーカイブ先へのアクセスに問題があったかどうかを示します。

これらの特性は、何通りかの組合せが可能です。インスタンスの各アーカイブ先について現在の状態などの情報を取得するには、V$ARCHIVE_DESTビューを問い合せます。

LOG_ARCHIVE_DEST_STATE_n(nは1から31の整数)初期化パラメータを使用すると、指定したアーカイブ先(n)の使用可能状態を制御できます。

- ENABLEは、アーカイブ先としてデータベースが使用できることを示します。
- DEFERは、その位置が一時的に使用禁止になっていることを示します。
- ALTERNATEは、代替アーカイブ先を示します。代替アーカイブ先の使用可能状態はDEFERです。その親アーカイブ

先に障害が発生すると、代替アーカイブ先の使用可能状態はENABLEになります。ALTERNATEは、LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31のアーカイブ先には指定できません。

親トピック: アーカイブ先の指定

12.4.4 代替アーカイブ先の指定

別のアーカイブ先に障害が発生した場合にのみ、ある場所がアーカイブ先になるように指定する場合は、その場所を代替アーカイブ先にできます。ローカルとリモート両方のアーカイブ先を代替アーカイブ先にできます。

次の例では、LOG_ARCHIVE_DEST_4がLOG_ARCHIVE_DEST_3の代替アーカイブ先になります。

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_4 = 'LOCATION=/disk4/arch';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_3 = 'LOCATION=/disk3/arch MAX_FAILURE=1
   ALTERNATE=LOG_ARCHIVE_DEST_4';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4=ALTERNATE;
SQL> SELECT dest_name, status, destination FROM v$archive_dest;

DEST_NAME                STATUS    DESTINATION
------------------------ --------- -------------------------------------------------
LOG_ARCHIVE_DEST_1       VALID     /disk1/arch
LOG_ARCHIVE_DEST_2       VALID     /disk2/arch
LOG_ARCHIVE_DEST_3       VALID     /disk3/arch
LOG_ARCHIVE_DEST_4       ALTERNATE /disk4/arch
```

親トピック: アーカイブ先の指定

12.5 ログ転送モードについて

アーカイブ・ログをアーカイブ先に転送する場合のモードには、ノーマル・アーカイブ転送およびスタンバイ転送という2つのモードがあります。ノーマル転送では、ファイルはローカル・ディスクに転送されます。スタンバイ転送では、ファイルはネットワークを介してローカルまたはリモートのスタンバイ・データベースに転送されます。

- 通常転送モード
  通常転送モードでは、アーカイブ先はデータベースの別のディスク・ドライブです。
- スタンバイ転送モード
  スタンバイ転送モードでは、アーカイブ先はローカルまたはリモートのスタンバイ・データベースです。

親トピック: アーカイブREDOログ・ファイルの管理

12.5.1 通常転送モード

通常転送モードでは、アーカイブ先はデータベースの別のディスク・ドライブです。

この構成では、アーカイブがインスタンスに必要な他のファイルと競合せず、短時間で完了します。アーカイブ先は、LOG_ARCHIVE_DEST_nまたはLOG_ARCHIVE_DESTパラメータで指定します。

親トピック: ログ転送モードについて

12.5.2 スタンバイ転送モード

スタンバイ転送モードでは、アーカイブ先はローカルまたはリモートのスタンバイ・データベースです。

ノート:

ローカル・ディスク上でスタンバイ・データベースをメンテナンスすることも可能ですが、スタンバイ・データベースはリモート・サイトでメンテナンスし、最大限の障害対策を講じることをお薦めします。

関連項目:

- 『Oracle Data Guard概要および管理』
- サービス名を使用してリモート・データベースに接続する方法の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください

親トピック: ログ転送モードについて

12.6 アーカイブ先の障害管理

アーカイブ先で発生した障害が、自動アーカイブ・モードで操作している場合のエラー原因になることがあります。Oracle Databaseには、アーカイブ先の障害に関連する問題を最小限に抑えるためのプロシージャが用意されています。

- 正常なアーカイブ先の最小数の指定
  オプションの初期化パラメータLOG_ARCHIVE_MIN_SUCCEED_DEST=nによって、データベースがオンライン・ログ・ファイルを再利用できるようになるまでにREDOログ・グループを正常にアーカイブすることが必要なアーカイブ先の最小数が決定されます。デフォルト値は1です。nの有効値は、二重化を使用する場合は1から2、多重化を使用する場合は1から31です。
- 障害アーカイブ先への再アーカイブ
  LOG_ARCHIVE_DEST_nパラメータのREOPEN属性を使用して、エラーの発生後にARCnが障害アーカイブ先への再アーカイブを試行するかどうかと、その時期を指定します。REOPENは、OPENエラーのみでなく、すべてのエラーに適用されます。

親トピック: アーカイブREDOログ・ファイルの管理

12.6.1 正常なアーカイブ先の最小数の指定

オプションの初期化パラメータ LOG_ARCHIVE_MIN_SUCCEED_DEST=nによって、データベースがオンライン・ログ・ファイルを再利用できるようになるまでにREDOログ・グループを正常にアーカイブすることが必要なアーカイブ先の最小数が決定されます。デフォルト値は1です。nの有効値は、二重化を使用する場合は1から2、多重化を使用する場合は1から31です。

- 必須およびオプションのアーカイブ先の指定
  LOG_ARCHIVE_DEST_nパラメータを使用すると、アーカイブ先としてOPTIONAL(デフォルト)またはMANDATORYを指定できます。
- 正常なアーカイブ先の数の指定: 使用例
  LOG_ARCHIVE_DEST_nおよびLOG_ARCHIVE_MIN_SUCCEED_DEST初期化パラメータの関係は、使用例を見ると理解しやすくなります。

親トピック: アーカイブ先の障害管理

12.6.1.1 必須およびオプションのアーカイブ先の指定

LOG_ARCHIVE_DEST_n初期化パラメータを使用すると、アーカイブ先としてOPTIONAL(デフォルト)またはMANDATORYを

指定できます。

- LOG_ARCHIVE_DEST_n初期化パラメータでアーカイブ先をOPTIONAL (デフォルト)またはMANDATORYに設定します。

LOG_ARCHIVE_MIN_SUCCEED_DEST=nパラメータでは、すべてのMANDATORYアーカイブ先と、非スタンバイのOPTIONALアーカイブ先をいくつか使用して、LGWRがオンライン・ログを上書きできるかどうかが判断されます。以下のルールが適用されます。

- アーカイブ先としてMANDATORY属性を指定しないと、OPTIONALが指定されます。
- ローカル・アーカイブ先を少なくとも1つは指定する必要があります。この場合は、OPTIONALまたはMANDATORYを宣言できます。
- MANDATORY属性は、LOG_ARCHIVE_DEST_1からLOG_ARCHIVE_DEST_10のアーカイブ先にのみ指定できます。
- LOG_ARCHIVE_MIN_SUCCEED_DESTの最小値は1なので、LOG_ARCHIVE_MIN_SUCCEED_DEST=nの値を指定すると、Oracle Databaseでは、少なくとも1つのローカル・アーカイブ先がMANDATORYとして扱われます。
- LOG_ARCHIVE_MIN_SUCCEED_DESTには、アーカイブ先の数を超える値や、MANDATORYのアーカイブ先の数とOPTIONALのローカル・アーカイブ先の数との合計を超える値は指定できません。
- MANDATORYのアーカイブ先にDEFERを指定した場合で、アーカイブ・ログがスタンバイ・サイトに転送されないままオンライン・ログが上書きされるときは、ログを手動でスタンバイ・サイトに転送する必要があります。

アーカイブ・ログを二重化する場合は、LOG_ARCHIVE_DESTおよびLOG_ARCHIVE_DUPLEX_DESTパラメータを使用して、アーカイブ先が必須かオプションかを指定できます。以下のルールが適用されます。

- LOG_ARCHIVE_DESTによって宣言されたアーカイブ先は必須です。
- LOG_ARCHIVE_DUPLEX_DESTによって宣言されたアーカイブ先は、LOG_ARCHIVE_MIN_SUCCEED_DEST = 1であればオプション、LOG_ARCHIVE_MIN_SUCCEED_DEST = 2であれば必須です。

親トピック: 正常なアーカイブ先の最小数の指定

12.6.1.2 正常なアーカイブ先の数の指定: 使用例

LOG_ARCHIVE_DEST_nおよびLOG_ARCHIVE_MIN_SUCCEED_DEST初期化パラメータの関係は、使用例を見ると理解しやすくなります。

- オプションのローカル・アーカイブ先へのアーカイブ例
  この例では、それぞれOPTIONALとして宣言している3つのローカル・アーカイブ先にアーカイブします。
- 必須およびオプションのアーカイブ先へのアーカイブ例
  この使用例では、MANDATORYおよびOPTIONALのローカル・アーカイブ先にアーカイブします。

親トピック: 正常なアーカイブ先の最小数の指定

12.6.1.2.1 オプションのローカル・アーカイブ先へのアーカイブ例

この例では、それぞれOPTIONALとして宣言している3つのローカル・アーカイブ先にアーカイブします。

表12-1に、この場合のLOG_ARCHIVE_MIN_SUCCEED_DEST=nに考えられる値を示します。

表12-1 使用例1のLOG_ARCHIVE_MIN_SUCCEED_DESTの値

値	意味
1	データベースは、最低 1 つの OPTIONAL のアーカイブ先へのアーカイブに成功した場合のみログ・ファイルを再利用できます。
2	データベースは、最低 2 つの OPTIONAL のアーカイブ先へのアーカイブに成功した場合のみログ・ファイルを再利用できます。
3	データベースは、OPTIONAL のすべてのアーカイブ先へのアーカイブに成功した場合のみログ・ファイルを再利用できます。
4 以上	エラーです。値がアーカイブ先数を超えています。

この例は、LOG_ARCHIVE_DEST_nパラメータを使用してアーカイブ先を明示的にMANDATORYに設定していない場合でも、LOG_ARCHIVE_MIN_SUCCEED_DESTが1、2または3に設定されていれば、データベースは必ずこれらの位置の1つ以上に正常にアーカイブすることを示しています。

親トピック: 正常なアーカイブ先の数の指定: 使用例

12.6.1.2.2 必須およびオプションのアーカイブ先へのアーカイブ例

この例では、MANDATORYおよびOPTIONALのローカル・アーカイブ先にアーカイブします。

次のような状況を考えてみます。

- MANDATORYのアーカイブ先は2つ指定されている。
- OPTIONALのアーカイブ先は2つ指定されている。
- アーカイブ先は、いずれもスタンバイ・データベースではない。

表12-2に、LOG_ARCHIVE_MIN_SUCCEED_DEST=nに考えられる値を示します。

表12-2 使用例2のLOG_ARCHIVE_MIN_SUCCEED_DESTの値

値	意味
1	データベースは、この値を無視して MANDATORY のアーカイブ先の数(この例では 2)を使用します。
2	データベースは、OPTIONAL のアーカイブ先へのアーカイブに失敗しても、ログ・ファイルを再利用できます。
3	データベースは、最低 1 つの OPTIONAL のアーカイブ先へのアーカイブに成功した場合のみログを再利用できます。
4	データベースは、OPTIONAL のアーカイブ先へのアーカイブに両方とも成功した場合のみログを再利用できます。

値	意味
5 以上	エラーです。値がアーカイブ先数を超えています。

この例は、アーカイブ先が少なくなるようにLOG_ARCHIVE_MIN_SUCCEED_DESTを設定した場合でも、データベースはその設定とは無関係に、MANDATORYとして指定されているアーカイブ先に必ずアーカイブすることを示しています。

親トピック: 正常なアーカイブ先の数の指定: 使用例

12.6.2 障害アーカイブ先への再アーカイブ

LOG_ARCHIVE_DEST_nパラメータのREOPEN属性を使用して、エラーの発生後にARCnが障害アーカイブ先への再アーカイブを試行するかどうかと、その時期を指定します。REOPENは、OPENエラーのみでなく、すべてのエラーに適用されます。

REOPEN=nでは、ARCnが障害アーカイブ先の再オープンを試行するまでの最小秒数を設定します。nのデフォルト値は300秒です。値に0(ゼロ)を指定すると、REOPEN属性はオフになり、ARCnは障害発生後にアーカイブを試行しません。REOPENキーワードを指定しない場合、ARCnはエラー発生後にアーカイブ先を再オープンしません。

REOPENを使用して、ARCnが再接続とアーカイブ・ログ転送を試行する回数を指定することはできません。REOPENは成功または失敗で終了します。

OPTIONALアーカイブ先にREOPENを指定すると、データベースはエラーがある場合にオンライン・ログを上書きできます。MANDATORYのアーカイブ先にREOPENを指定すると、正常にアーカイブできない場合に本番データベースの機能が停止します。この状況では、次の方法を検討してください。

- 障害アーカイブ先に手動でアーカイブする。
- アーカイブ先を遅延させる、アーカイブ先をオプションとして指定する、サービスを変更するのいずれかの方法によってアーカイブ先を変更する。
- アーカイブ先を削除する。

REOPENキーワードを使用する場合は、次の点に注意してください。

- ARCnは、ログ・ファイルの先頭からアーカイブ操作を開始する場合にのみ、アーカイブ先を再オープンし、実行中の操作の途中で再オープンすることはありません。ARCnは、常に先頭からログ・コピーを再試行します。
- 特定の時間またはデフォルト設定でREOPENを指定した場合、ARCnは記録されたエラー発生時刻からREOPEN間隔が経過した時刻が現在時刻より前かどうかをチェックします。現在時刻より前であれば、ARCnはログ・コピーを再試行します。
- REOPEN句は、ACTIVE=TRUEのアーカイブ先状態に影響を及ぼします。VALIDおよびENABLED状態は変化しません。

  ここは間違っています。アーカイブ先は非アクティブ、有効または無効になります。ACTIVEステータスはありません。そのため、「REOPEN句はアーカイブ先ステータスをVALIDに設定する」とする必要があると思われます。DL

親トピック: アーカイブ先の障害管理

12.7 ARCHIVELOGプロセスによって生成されるトレース出力の制御

バックグラウンド・プロセスは適宜、トレース・ファイルに情報を書き込みます。ARCHIVELOGプロセスの場合は、トレース・ファイルに書き込む出力を制御できます。

archivelogプロセス用のトレース・ファイルに生成される出力を制御するには:

- LOG_ARCHIVE_TRACE初期化パラメータを設定してトレース・レベルを0、1、2、4、8などに指定します。

パラメータ値として、必要な各トレース・レベルの合計を設定することにより、トレース・レベルを組み合せることができます。たとえば、LOG_ARCHIVE_TRACE=12に設定すると、トレース・レベル8および4の出力が生成されます。また、プライマリ・データベースとスタンバイ・データベースには、異なる値を設定できます。

LOG_ARCHIVE_TRACEパラメータのデフォルト値は0(ゼロ)です。このレベルでは、エラー条件が発生すると、ARCHIVELOGプロセスによって適切なアラートおよびトレース・エントリが生成されます。

このパラメータの値は、ALTER SYSTEM文で動的に変更できます。たとえば:

```
ALTER SYSTEM SET LOG_ARCHIVE_TRACE=12;
```

この方法で行った変更は、次回のアーカイブ操作の開始時に有効になります。

関連項目:

- 「トレース・ファイルおよびアラート・ログを使用したエラーの監視」
- LOG_ARCHIVE_TRACE初期化パラメータの有効な値の説明などの詳細は、『Oracle Databaseリファレンス』を参照してください。
- このパラメータをスタンバイ・データベースで使用する方法については、『Oracle Data Guard概要および管理』を参照してください

親トピック: アーカイブREDOログ・ファイルの管理

12.8 アーカイブREDOログに関する情報の表示

アーカイブREDOログに関する情報を表示するには、動的パフォーマンス・ビューまたはARCHIVE LOG LISTコマンドを使用します。

- アーカイブREDOログ・ファイルのビュー
  アーカイブREDOログ・ファイルに関する情報を動的パフォーマンス・ビューのセットに問い合せることができます。
- ARCHIVE LOG LISTコマンドの使用
  SQL*PlusコマンドのARCHIVE LOG LISTを使用して、接続されているインスタンスのアーカイブ情報を表示します。

親トピック: アーカイブREDOログ・ファイルの管理

12.8.1 アーカイブREDOログ・ファイルのビュー

アーカイブREDOログ・ファイルに関する情報を動的パフォーマンス・ビューのセットに問い合せることができます。

アーカイブREDOログ・ファイルに関して役立つ情報を含む動的パフォーマンス・ビューがいくつかあります。次の表に要約を示します。

動的パフォーマンス・ビュー	説明
V$DATABASE	データベースがARCHIVELOGモードとNOARCHIVELOGモードのいずれであるか、およびMANUAL(アーカイブ・モード)が指定されているかどうかが表示されま

動的パフォーマンス・ビュー	説明
	す。
V$ARCHIVED_LOG	制御ファイルに格納されたアーカイブ・ログ履歴情報が表示されます。リカバリ・カタログを使用している場合は、RC_ARCHIVED_LOG ビューにも同様の情報が含まれます。
V$ARCHIVE_DEST	現行インスタンス、すべてのアーカイブ先、各アーカイブ先の現行の値、モードおよび状態が表示されます。
V$ARCHIVE_PROCESSES	インスタンスの各アーカイブ・プロセスの状態情報が表示されます。
V$BACKUP_REDOLOG	アーカイブ・ログのバックアップ情報が含まれます。リカバリ・カタログを使用している場合は、RC_BACKUP_REDOLOG ビューにも同様の情報が含まれます。
V$LOG	データベースの REDO ログ・グループをすべて表示し、その中でアーカイブする必要があるグループを示します。
V$LOG_HISTORY	アーカイブ済ログや各アーカイブ・ログの SCN 範囲などのログ履歴情報が含まれます。

たとえば、次の問合せでは、アーカイブする必要があるREDOログ・グループが表示されます。

```
SELECT GROUP#, ARCHIVED
   FROM SYS.V$LOG;
GROUP#     ARC
--------   ---
       1   YES
       2   NO
```

現行のアーカイブ・モードを確認するには、V$DATABASEビューを問い合せます。

```
SELECT LOG_MODE FROM SYS.V$DATABASE;
LOG_MODE
------------
NOARCHIVELOG
```

親トピック: アーカイブREDOログに関する情報の表示

12.8.2 ARCHIVE LOG LISTコマンドの使用

SQL*PlusコマンドのARCHIVE LOG LISTを使用して、接続されているインスタンスのアーカイブ情報を表示します。

たとえば:

```
SQL> ARCHIVE LOG LIST
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            D:¥oracle¥oradata¥IDDB2¥archive
Oldest online log sequence     11160
Next log sequence to archive   11163
```

```
Current log sequence           11163
```

この表示は、現行インスタンスのアーカイブREDOログの設定に関して必要なすべての情報を示します。

- このデータベースは現在ARCHIVELOGモードで操作されています。
- 自動アーカイブは使用可能です。
- アーカイブREDOログのアーカイブ先は、D:¥oracle¥oradata¥IDDB2¥archiveです。
- 一杯になったREDOログ・グループのうち、最も古いものの順序番号は11160です。
- 一杯になったREDOログ・グループのうち、次にアーカイブされるものの順序番号は11163です。
- 現行のREDOログ・ファイルの順序番号は11163です。

関連項目:

ARCHIVE LOG LISTコマンドの詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください

親トピック: アーカイブREDOログに関する情報の表示

13 表領域の管理

表領域は、関連する論理構造をグループ化するデータベース記憶域です。データベースのデータファイルは表領域に格納されます。

- 表領域を管理するためのガイドライン
  ガイドラインに従って表領域に対して作業できます。
- 表領域の作成
  表領域を作成して、表と索引などの関連する論理構造をグループ化します。データベースのデータファイルは表領域に格納されます。
- インメモリー列ストアへの表領域の格納の検討
  表領域の作成または変更時に、インメモリー列ストアに対して表領域を有効にできます。インメモリー列ストアに対して表領域が有効になると、表領域内のすべての表はデフォルトでインメモリー列ストアに対して有効になります。
- 表領域の非標準のブロック・サイズの指定
  DB_BLOCK_SIZE初期化パラメータで指定された標準のデータベース・ブロック・サイズとは異なるブロック・サイズの表領域を作成できます。この機能によって、ブロック・サイズの異なる表領域をデータベース間でトランスポートできます。
- REDOレコードの書込みの制御
  一部のデータベース操作については、データベースでREDOレコードを生成するかどうかを制御できます。
- 表領域の可能性の変更
  オンラインの表領域をオフライン化すると、一般的な使用を一時的に禁止にできます。データベースの残りの部分はオープンしていて使用可能であり、ユーザーはデータにアクセスできます。逆に、オフライン状態の表領域をオンライン化して、データベース・ユーザーがその表領域内のスキーマ・オブジェクトを使用できるようにすることもできます。表領域の可用性を変更するには、データベースをオープンする必要があります。
- 読取り専用表領域の使用
  表領域は読取り専用モードに設定できます。これにより、それに格納されているデータの更新が禁止されます。
- 表領域の変更とメンテナンス
  データファイルと一時ファイルの追加などの作業を行って、表領域を変更およびメンテナンスできます。
- 表領域の名前変更
  永続表領域または一時表領域の名前は、ALTER TABLESPACE文のRENAME TO句を使用して変更できます。
- 表領域の削除
  表領域とその内容が不要になった場合は、その表領域と内容(表領域に含まれるセグメント)をデータベースから削除できます。
- シャドウ表領域を使用した消失書込み保護の管理
  データ・ブロックの消失書込みは、永続ストレージでは書込みが行われなかったが、I/Oサブシステムではブロック書込みの完了が確認された場合に発生します。消失書込みに対する保護として、シャドウ消失書込み保護を使用できます。
- SYSAUX表領域の管理
  SYSAUX表領域は、データベースの作成時に、SYSTEM表領域の補助表領域としてインストールされます。これまで個別に表領域を作成して使用していた一部のデータベースのコンポーネントは、SYSAUX表領域に含まれるようになりました。
- ローカル管理表領域の問題の修正
  Oracle Databaseにはローカル管理表領域の問題を修正する手段が用意されています。
- ローカル管理表領域へのSYSTEM表領域の移行
  DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCALプロシージャを使用して、SYSTEM表領域をディ

クショナリ管理からローカル管理に移行します。

- 表領域の情報の表示
  Oracle Databaseのデータ・ディクショナリ・ビューを使用して、表領域に関する情報を問い合せることができます。

関連項目:

- 『Oracle Database概要』
- Oracle Databaseサーバーによって作成および管理されるデータファイルと一時ファイルの作成方法は、「Oracle Managed Filesの使用」を参照してください
- 「データベース間での表領域のトランスポート」

親トピック: Oracle Databaseの構造と記憶域

13.1 表領域を管理するためのガイドライン

ガイドラインに従って表領域に対して作業できます。

- 複数の表領域の使用
  複数の表領域を使用すると、より柔軟にデータベースを操作できます。
- ユーザーに対する表領域割当て制限の割当て
  表、クラスタ、マテリアライズド・ビュー、索引およびその他のオブジェクトを作成しようとするユーザーには、そのオブジェクトを作成するための権限と、そのオブジェクトのセグメントを格納する表領域の割当て制限(領域の許容または制限)を付与します。

親トピック: 表領域の管理

13.1.1 複数の表領域の使用

データベース操作を実行する際に複数の表領域を使用すると、システムの柔軟性が向上します。

データベースに複数の表領域があるときには、次のことが可能です。

- ユーザー・データをデータ・ディクショナリ・データから分離し、I/Oの競合を減らす。
- あるアプリケーションのデータを別のアプリケーションのデータから分離し、表領域をオフライン化する必要が生じた場合に、複数のアプリケーションが影響を受けないようにする。
- I/Oの競合を低減するために、異なるディスク・ドライブ上に異なる表領域のデータファイルを配置する。
- 別の表領域をオンライン状態に維持しながら、個々の表領域をオフライン化して、全体の可用性を高める。
- 高い更新アクティビティ、読取り専用アクティビティ、一時セグメント記憶域など、異なるタイプのデータベース利用のために異なる表領域を確保することによって、表領域利用を最適化する。
- 表領域のバックアップを個別に作成する。

一部のオペレーティング・システムでは、同時にオープン可能なファイルの数に制限が設けられています。このような制限によって、同時にオンライン化可能な表領域の数に影響が出ることがあります。そのため、使用しているオペレーティング・システムの制限を超えないように、表領域を効率よく計画する必要があります。表領域はデータベースの要件を満たすために必要な数だけ作成し、構成するファイル数もできるかぎり少なくなるようにしてください。表領域のサイズを大きくする必要がある場合は、小さいデータファイルを多数作成するのではなく、1つまたは2つの大きなデータファイルを追加するか、または自動拡張を使用可能にして

データファイルを作成します。

これらの要素を考慮に入れてデータを再検討し、データベース設計に必要になる表領域の数を決定してください。

親トピック: 表領域を管理するためのガイドライン

13.1.2 ユーザーに対する表領域割当て制限の割当て

表、クラスタ、マテリアライズド・ビュー、索引およびその他のオブジェクトを作成しようとするユーザーには、そのオブジェクトを作成するための権限と、そのオブジェクトのセグメントを格納する表領域の割当て制限(領域の許容または制限)を付与します。

ノート:

パッケージ、プロシージャ、ファンクションなどの PL/SQL オブジェクトの場合、ユーザーにはオブジェクトを作成する権限のみが必要です。これらの PL/SQL オブジェクトを作成するための、明示的な表領域割当ては必要ありません。

関連項目:

ユーザーの作成方法および表領域割当て制限の割当て方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: 表領域を管理するためのガイドライン

13.2 表領域の作成

表領域を作成して、表と索引などの関連する論理構造をグループ化します。データベースのデータファイルは表領域に格納されます。

- 表領域の作成について
  新しい表領域を作成するには、SQL文CREATE TABLESPACEまたはCREATE TEMPORARY TABLESPACEを使用します。表領域を作成するには、CREATE TABLESPACEシステム権限が必要です。
- ローカル管理表領域
  ローカル管理表領域では、各データファイルに格納されるビットマップを使用してエクステントを管理します。
- bigfile表領域
  bigfile表領域を使用してデータベースの記憶域容量を増やし、多数のデータファイルと一時ファイルを管理する負荷を減らすことができます。
- デフォルト圧縮属性を持つ表領域
  表領域の作成時に、表領域に作成されるすべての表と索引またはそれらのパーティションをデフォルトで圧縮するように指定できます。
- 暗号化された表領域
  永続表領域を暗号化して機密データを保護できます。
- 一時表領域
  一時表領域によって、メモリー内に収まらない複数のソート操作の同時実行性が向上します。また、一時表領域によって、ソート中の領域管理操作の効率も向上します。
- 一時表領域グループ
  一時表領域グループは、データベースに対してデフォルトの一時表領域として割り当てられる表領域グループです。

親トピック: 表領域の管理

13.2.1 表領域の作成について

新しい表領域を作成するには、SQL文CREATE TABLESPACEまたはCREATE TEMPORARY TABLESPACEを使用します。表領域を作成するには、CREATE TABLESPACEシステム権限が必要です。

表領域を作成する場合、表領域を格納するデータベースを作成する必要があります。どのデータベースでも、重要な表領域はSYSTEM表領域です。ここにはデータ・ディクショナリやシステム・ロールバック・セグメントなど、データベース・サーバー機能の基本になる情報が格納されます。SYSTEM表領域は、データベース作成時に最初に作成される表領域です。これは他のすべての表領域と同様に管理されますが、より高いレベルの権限が必要であり、また一部制限事項があります。たとえば、SYSTEM表領域の名前変更、削除、オフライン化は実行できません。

SYSAUX表領域は、データベースの作成時に必ず作成され、SYSTEM表領域の補助表領域として機能します。これには、様々なOracle製品および機能で使用されるスキーマが格納されるため、各製品で独自の表領域を持つ必要がなくなります。SYSTEM表領域と同様に、SYSAUX表領域を管理するためにはより高いレベルのセキュリティが必要で、この表領域の名前変更や削除はできません。SYSAUX表領域の管理については、「SYSAUX表領域の管理」を参照してください。

表領域を作成するステップはオペレーティング・システムによって異なりますが、必ず最初に、オペレーティング・システムを使用して、データファイルが割り当てられるディレクトリ構造を作成する必要があります。ほとんどのオペレーティング・システムでは、新しい表領域を作成するとき、またはデータファイルを加えて既存の表領域を変更するときに、データファイルのサイズと完全なファイル名を指定します。新しい表領域を作成するとき、または既存の表領域を変更するときは、いずれの場合も、データベースは、指定されたとおりにデータファイルを自動的に割り当てて、フォーマットします。

また、CREATE UNDO TABLESPACE文を使用して、UNDOレコードを格納するために特別に設計されたUNDO表領域と呼ばれる特殊なタイプの表領域を作成できます。これらは、データベースが生成するレコードで、リカバリや読込み一貫性のために、またはROLLBACK文の要求を受けて、データベースの変更をロールバックまたは取り消す際に使用されます。UNDO表領域の作成および管理の詳細は、「UNDOの管理」を参照してください。

ALTER TABLESPACEまたはALTER DATABASE文を使用して、表領域を変更できます。そのためには、ALTER TABLESPACEまたはALTER DATABASEシステム権限が必要です。

関連項目:

- データベース作成時に作成される表領域の詳細は、「Oracle Databaseの作成および構成」および使用しているオペレーティング・システムのOracle Databaseインストレーション・ガイドを参照してください。
- CREATE TABLESPACE、CREATE TEMPORARY TABLESPACE、ALTER TABLESPACEおよびALTER DATABASEの各文の構文とセマンティクスの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 非標準のブロック・サイズを持つ表領域の作成に必要な初期化パラメータの詳細は、「データベース・ブロック・サイズの指定」を参照してください

親トピック: 表領域の作成

13.2.2 ローカル管理表領域

ローカル管理表領域では、各データファイルに格納されるビットマップを使用してエクステントを管理します。

- ローカル管理表領域の使用について
  ローカル管理表領域では、その表領域内のすべてのエクステント情報がビットマップを使用して追跡されます。

- ローカル管理表領域の作成
  ローカル管理表領域を作成するには、CREATE TABLESPACE文のEXTENT MANAGEMENT句にLOCALを指定します。
- ローカル管理表領域のセグメント領域管理の指定
  ローカル管理表領域の場合、Oracle Databaseでセグメント領域の管理に使用できる方法には、自動と手動の2つがあります。

親トピック: 表領域の作成

13.2.2.1 ローカル管理表領域の使用について

ローカル管理表領域では、その表領域内のすべてのエクステント情報がビットマップを使用して追跡されます。

ローカル管理表領域には、次の利点があります。

- 領域操作が高速かつ同時に実行されます。領域の割当てと割当て解除によって、ローカル管理のリソース(ヘッダー・ファイルに格納されているビットマップ)が変更されます。
- パフォーマンスが向上します。
- ローカル管理の一時表領域ではUNDOやREDOが生成されないため、読取り可能なスタンバイ・データベースを使用できます。
- AUTOALLOCATE句を指定すると、データベースで適切なエクステント・サイズが自動的に選択されるため、領域割当てが簡素化されます。
- 必要な情報はファイル・ヘッダーとビットマップ・ブロックに格納されるため、データ・ディクショナリに対するユーザーの依存性が低下します。
- ローカル管理表領域では、使用可能エクステントを結合する必要はありません。

SYSTEM表領域も含めて、すべての表領域をローカルに管理できます。

DBMS_SPACE_ADMINパッケージによって、ローカル管理表領域のメンテナンス手順が提供されます。

関連項目:

- 「ローカル管理のSYSTEM表領域の作成」、「ローカル管理表領域へのSYSTEM表領域の移行」および「ローカル管理表領域の問題の診断と修復」
- 単一データファイルまたは一時ファイルのみ格納される、別のタイプのローカル管理表領域を作成する方法の詳細は、「bigfile表領域」を参照してください。
- DBMS_SPACE_ADMINパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ローカル管理表領域

13.2.2.2 ローカル管理表領域の作成

ローカル管理表領域を作成するには、CREATE TABLESPACE文のEXTENT MANAGEMENT句にLOCALを指定します。

これは新しい永続表領域に対するデフォルトですが、AUTOALLOCATE句またはUNIFORM句を指定するには、EXTENT MANAGEMENT LOCAL句を指定する必要があります。AUTOALLOCATE句(デフォルト)を指定してデータベースでエクステントを自動的に管理するか、または表領域を特定サイズ(UNIFORM)の均一エクステントで管理するかを指定できます。

様々なサイズのオブジェクトが表領域に含まれ、異なるエクステント・サイズの多数のエクステントが必要と予測される場合は、AUTOALLOCATEを選択してください。領域の割当てと割当て解除を厳密に制御しなくてもよい場合は、AUTOALLOCATEを選択すると表領域の管理作業が簡素化されます。この設定では、ある程度の領域が無駄になるという短所もありますが、ほとんどの場合、Oracle Databaseによって領域が管理されるという利点の方が重要です。

未使用領域の厳密な制御が必要で、オブジェクトに割り当てられる領域、エクステントの数とサイズを正確に予測できる場合は、UNIFORMを選択してください。この設定によって、表領域から使用できない領域がなくなります。

エクステント管理のタイプを明示的に指定しない場合は、Oracle DatabaseによってエクステントDatabase管理が次のように判断されます。

- CREATE TABLESPACE文にDEFAULT記憶域句を指定しない場合、データベースでは自動割当てのローカル管理表領域が作成されます。
- CREATE TABLESPACE文にDEFAULT記憶域句を指定した場合、データベースでは次のことが考慮されます。
  - MINIMUM EXTENT句を指定した場合、MINIMUM EXTENT、INITIALおよびNEXTの値が等しく、PCTINCREASEの値が0 (ゼロ)かどうかが評価されます。その場合、データベースにより、エクステント・サイズ = INITIALで均一なローカル管理表領域が作成されます。MINIMUM EXTENT、INITIALおよびNEXTパラメータの値が等しくない場合、またはPCTINCREASEが0(ゼロ)でない場合は、指定したエクステント記憶域パラメータが無視され、自動割当てのローカル管理表領域が作成されます。
  - MINIMUM EXTENT句を指定しなかった場合、データベースは、INITIALおよびNEXTの記憶域値が等しく、PCTINCREASEが0(ゼロ)であるかどうかのみを評価します。その場合、表領域はローカル管理で均一です。それ以外の場合は、ローカル管理の表領域が作成され、自動的に割り当てられます。

たとえば、次の文は、ローカル管理表領域lmtbsbを作成し、AUTOALLOCATEを指定しています。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

AUTOALLOCATEを指定すると、表領域はシステム管理になり、最小エクステント・サイズは64KBとなります。

AUTOALLOCATEのかわりにUNIFORMを指定すると、表領域は均一サイズのエクステントで管理されます。このサイズは、UNIFORMのSIZE句で指定できます。SIZEを指定しないと、デフォルト・サイズは1MBになります。

次の例は、均一の128Kエクステントを持つ表領域を作成します。(2Kブロックを使用するデータベースでは、各エクステントは64のデータベース・ブロックに相当します。)それぞれの128Kエクステントは、このファイルのエクステント・ビットマップ内のビットとして表されます。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

EXTENT MANAGEMENT LOCALを明示的に指定する場合は、DEFAULTの記憶域句、MINIMUM EXTENTまたはTEMPORARYを指定できません。ローカル管理の一時表領域を作成するには、CREATE TEMPORARY TABLESPACE文を使用します。

ノート:

ローカル管理表領域にデータファイルを割り当てる場合は、ユーザー領域の一部である、領域管理に使用されるメタデータ(エクステント・ビットマップまたは領域ヘッダー・セグメント)用の領域を考慮する必要があります。たとえば、エクステント管理句で UNIFORM 句を指定し、その際に SIZE パラメータを省略した場合、デフォルトのエクステント・サイ

ズは 1MB です。この場合、データファイルに指定されたサイズは 1MB よりも大きい(最低 1 ブロックとビットマップの領域を足したもの)必要があります。

親トピック: ローカル管理表領域

13.2.2.3 ローカル管理表領域のセグメント領域管理の指定

ローカル管理表領域では、Oracle Databaseでセグメント領域の管理に使用できる方法には、自動と手動の2つがあります。

手動セグメント領域管理では、「空きリスト」と呼ばれるリンクされたリストを使用してセグメント内の空き領域を管理しますが、自動セグメント領域管理では、ビットマップを使用します。自動セグメント領域管理の方がより効率的な方法で、すべての新しい永続ローカル管理表領域のデフォルトです。

自動セグメント領域管理では、手動セグメント領域管理に比べて領域の使用効率が向上します。また、ユーザー数やインスタンス数の増加につれて拡張されるという点で自己チューニング型です。Oracle Real Application Clusters環境の場合、自動セグメント領域管理ではインスタンスに対する領域の動的アフィニティに対処できます。さらに、多くの標準的な処理負荷の場合、自動セグメント領域管理を使用したアプリケーションのパフォーマンスは、手動セグメント領域管理を使用して適切にチューニングされたアプリケーションよりも優れています。

自動セグメント領域管理は新しい永続ローカル管理表領域のデフォルトですが、SEGMENT SPACE MANAGEMENT AUTO句を使用して明示的に使用可能にできます。

たとえば、次の文は、自動セグメント領域管理を行うlmtbsb表領域を作成します。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL
    SEGMENT SPACE MANAGEMENT AUTO;
```

SEGMENT SPACE MANAGEMENT MANUAL句を指定すると、自動セグメント領域管理を使用禁止にできます。

表領域の作成時に指定するセグメント領域管理方法は、その後表領域に作成されたすべてのセグメントに対して適用されます。表領域のセグメント領域管理モードは変更できません。

ノート:

- エクステント管理を LOCAL UNIFORM に設定する場合は、各エクステントに 5 以上のデータベース・ブロックがあることを確認する必要があります。
- エクステント管理を LOCAL AUTOALLOCATE に設定する場合、およびデータベースのブロック・サイズが 16KB 以上の場合は、最小で 5 ブロック(64KB に切り上げられる)のエクステントが作成され、セグメント領域管理が管理されます。
- SYSTEM 表領域に自動セグメント領域管理を指定することはできません。

自動セグメント領域管理を使用するローカル管理表領域は、単一ファイル表領域、つまり、bigfile表領域(「bigfile表領域」を参照)としても作成できます。

親トピック: ローカル管理表領域

13.2.3 bigfile表領域

bigfile表領域を使用してデータベースの記憶域容量を増やし、多数のデータファイルと一時ファイルを管理する負荷を減らすことができます。

- bigfile表領域について
  bigfile表領域は、単一で非常に大きい(最大40億ブロック)データファイルを持つ可能性がある表領域です。これに対して、従来のsmallfile表領域には複数のデータファイルを格納できますが、各データファイルは大きくありません。
- bigfile表領域の作成
  bigfile表領域を作成するには、CREATE TABLESPACE文のBIGFILEキーワード(CREATE BIGFILE TABLESPACE ...)を指定します。
- bigfile表領域の識別
  データ・ディクショナリ・ビューのセットを問い合せて、bigfile表領域についての情報を取得できます。

親トピック: 表領域の作成

13.2.3.1 bigfile表領域について

bigfile表領域は、単一で非常に大きい(最大40億ブロック)データファイルを持つ可能性がある表領域です。これに対して、従来のsmallfile表領域には複数のデータファイルを格納できますが、各データファイルは大きくありません。

bigfile表領域の利点は、次のとおりです。

- 8000ブロックを持つbigfile表領域には、32TBのデータファイルを格納できます。32000ブロックを持つbigfile表領域には、128TBのデータファイルを格納できます。1つのOracle Databaseのデータファイルの最大数は制限されています(通常は64000ファイル)。したがって、bigfile表領域によって、Oracle Databaseの記憶域容量が大幅に増加します。
- bigfile表領域を使用すると、データベースに必要なデータファイルの数を減らすことができます。また、別の利点として、CREATE DATABASE文とCREATE CONTROLFILE文のDB_FILES初期化パラメータとMAXDATAFILESパラメータを調整すると、データファイル情報に必要なSGA領域の量と制御ファイルのサイズを削減できます。
- bigfile表領域によるデータファイルの透過性によって、データベース管理が簡素化されます。ALTER TABLESPACE文のSQL構文を使用すると、基礎になる各データファイルではなく表領域で操作を実行できます。

bigfile表領域は、自動セグメント領域管理を指定したローカル管理表領域でのみサポートされます。ただし、ローカル管理のUNDO表領域、一時表領域およびSYSTEM表領域の3つは例外です。

ノート:

- bigfile 表領域は、ストライプ化や RAID、および動的に拡張可能な論理ボリュームをサポートする、自動ストレージ管理(Oracle ASM)などの論理ボリューム・マネージャとともに使用することを目的にしています。
- パラレル問合せ実行および RMAN のバックアップ・パラレル化で問題が生じる可能性があるため、ストライプ化をサポートしていないシステムには大型ファイル表領域を作成しないでください。
- 表領域の容量が制限される場合があるため、大規模なファイル・サイズをサポートしていないプラットフォームで bigfile 表領域を使用することはお薦めしません。サポートされているファイルの最大サイズの詳細は、ご使用のオペレーティング・システム固有のドキュメントを参照してください。

親トピック: bigfile表領域

13.2.3.2 bigfile表領域の作成

bigfile表領域を作成するには、CREATE TABLESPACE文のBIGFILEキーワード(CREATE BIGFILE TABLESPACE...)を指定します。

Oracle Databaseは、自動セグメント領域管理を指定したローカル管理表領域を自動的に作成します。この文には、EXTENT MANAGEMENT LOCALおよびSEGMENT SPACE MANAGEMENT AUTOを必要に応じて指定できます。ただし、EXTENT MANAGEMENT DICTIONARYまたはSEGMENT SPACE MANAGEMENT MANUALを指定すると、データベースはエラーを返します。この文の残りの構文はCREATE TABLESPACE文と同じですが、指定できるのは1つのデータファイルのみです。たとえば:

```
CREATE BIGFILE TABLESPACE bigtbs
    DATAFILE '/u02/oracle/data/bigtbs01.dbf' SIZE 50G
...
```

SIZEは、キロバイト(KB)、メガバイト(MB)、ギガバイト(GB)またはテラバイト(TB)で指定できます。

データベース作成時にデフォルトの表領域タイプをBIGFILEに設定している場合は、CREATE TABLESPACE文にBIGFILEキーワードを指定する必要はありません。この場合、bigfile表領域はデフォルトで作成されます。

データベース作成時にデフォルトの表領域タイプをBIGFILEに設定しておきながら、従来の表領域(smallfile)を作成する場合は、CREATE SMALLFILE TABLESPACE文を指定すると、デフォルトの表領域タイプよりも作成する表領域が優先されます。

関連項目:

「データベース作成時のbigfile表領域のサポート」

親トピック: bigfile表領域

13.2.3.3 bigfile表領域の識別

データ・ディクショナリ・ビューのセットを問い合せて、bigfile表領域についてのデータを取得できます。

次のビューには、表領域をbigfile表領域として識別するBIGFILE列が含まれています。

- DBA_TABLESPACES
- USER_TABLESPACES
- V$TABLESPACE

これらのビューを問い合せて、bigfile表領域についてのデータを取得します。

bigfile表領域は、その単一データファイルの相対ファイル番号によっても識別できます。その番号は、ほとんどのプラットフォームで1024ですが、OS/390では4096です。

親トピック: bigfile表領域

13.2.4 デフォルト圧縮属性を持つ表領域

表領域の作成時に、表領域に作成されるすべての表と索引またはそれらのパーティションをデフォルトで圧縮するように指定できます。

- デフォルト圧縮属性を持つ表領域について

表領域を作成する際に、表領域に作成されるすべての表および索引のデータのデフォルト圧縮を指定できます。デフォルト圧縮レベルは、表領域を構成するパーティションにも適用されます。このデータを圧縮するとディスク使用量を削減できます。

- デフォルト圧縮属性を使用した表領域の作成
  表領域を作成するときに表圧縮のタイプを指定するには、DEFAULTキーワードを使用し、その後に圧縮タイプを含む表圧縮句を指定します。DEFAULTキーワードを使用して、その後に索引圧縮句と索引圧縮タイプを指定すると、索引圧縮のタイプを指定することもできます。

親トピック: 表領域の作成

13.2.4.1 デフォルト圧縮属性を持つ表領域について

表領域を作成する際に、表領域に作成されるすべての表および索引のデータのデフォルト圧縮を指定できます。デフォルト圧縮レベルは、表領域を構成するパーティションにも適用されます。このデータを圧縮するとディスク使用量を削減できます。

親トピック: デフォルト圧縮属性を持つ表領域

13.2.4.2 デフォルト圧縮属性を持つ表領域の作成

表領域を作成するときに、DEFAULTキーワードを使用し、その後に圧縮タイプを含む表圧縮句を指定すると、表圧縮のタイプを指定できます。DEFAULTキーワードを使用して、その後に索引圧縮句と索引圧縮タイプを指定すると、索引圧縮のタイプを指定することもできます。

次の文は、表領域内に作成されるすべての表およびパーティションで、特に指定がないかぎり高度な行圧縮が使用されることを示しています。

```
CREATE TABLESPACE ... DEFAULT ROW STORE COMPRESS ADVANCED ... ;
```

デフォルトの表領域圧縮指定は、その表領域に表またはパーティションを作成するときに上書きできます。

次の文は、表領域内に作成されるすべての索引で、特に指定がないかぎり、高レベルの拡張索引圧縮が使用されることを示しています。

```
CREATE TABLESPACE ... DEFAULT INDEX COMPRESS ADVANCED HIGH ... ;
```

デフォルトの表領域圧縮指定は、その表領域に索引を作成するときに上書きできます。

親トピック: デフォルト圧縮属性を持つ表領域

13.2.5 暗号化された表領域

永続表領域を暗号化して機密データを保護できます。

- 暗号化された表領域について
  暗号化表領域では、主にデータベース以外の手段による未承認のアクセスからデータを保護します。たとえば、あるOracle Databaseから別のOracle Databaseに移動するため、または格納用のオフサイト施設に移動するために暗号化された表領域をバックアップ・メディアに書き込むと、表領域は暗号化されたままになります。
- 暗号化された表領域の作成
  暗号化された表領域を作成して、未承認のアクセスに対してデータを保護できます。
- 暗号化された表領域に関する情報の表示
  DBA_TABLESPACESおよびUSER_TABLESPACESデータ・ディクショナリ・ビューを問い合せて、暗号化された表領域に関する情報を取得できます。

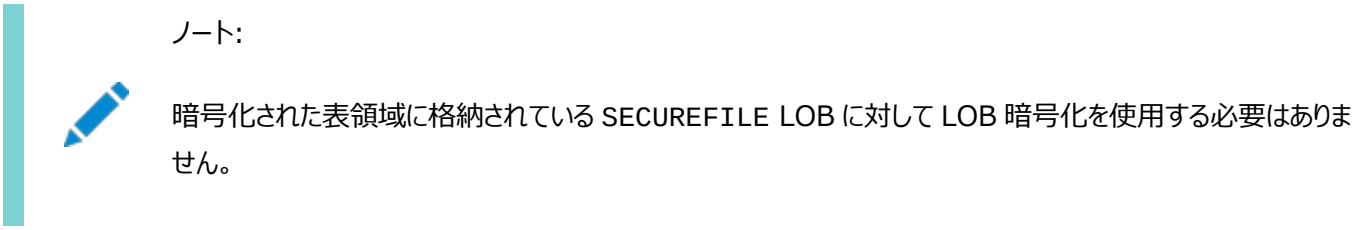
13.2.5.1 暗号化された表領域について

暗号化表領域では、主にデータベース以外の手段による未承認のアクセスからデータを保護します。たとえば、あるOracle Databaseから別のOracle Databaseに移動するため、または格納用のオフサイト施設に移動するために暗号化された表領域をバックアップ・メディアに書き込むと、表領域は暗号化されたままになります。

また、暗号化表領域では、データベースのセキュリティ機能を回避して、オペレーティング・システムのファイル・システムから直接データベース・ファイルにアクセスしようとするユーザーからデータを保護します。表領域の暗号化はアプリケーションに対して完全に透過的であるため、アプリケーションの変更は不要です。

表領域の暗号化によって、すべてのセキュリティ問題に対処できるわけではありません。たとえば、データベース内からのアクセスは制御できません。暗号化された表領域に格納されているオブジェクトに対する権限が付与されたユーザーは、追加のパスワードやキーを指定せずにそれらのオブジェクトにアクセスできます。

表領域を暗号化すると、すべての表領域ブロックが暗号化されます。暗号化は、表、クラスタ、索引、LOB(`BASICFILE`と`SECUREFILE`)、表パーティション、索引パーティションなどを含むすべてのセグメント・タイプに対してサポートされています。



ノート:

暗号化された表領域に格納されている `SECUREFILE` LOB に対して LOB 暗号化を使用する必要はありません。

最大限のセキュリティを確保するために、暗号化された表領域のデータは、UNDO表領域、REDOログおよび一時表領域に書き込まれる場合は自動的に暗号化されます。ただし、Oracle Database 12cリリース2 (12.2)以降では、UNDO表領域および一時表領域をオプションで暗号化できます。

異なる表領域に別のパーティションがあるパーティション表およびパーティション索引表の場合は、同じ表または索引で、暗号化された表領域と暗号化されていない表領域の両方を使用できます。

表領域暗号化はOracle Databaseの透過的データ暗号化機能を使用しますが、この機能を使用するには、データベースのマスター暗号化キーを保存するためにキーストアを作成する必要があります。暗号化された表領域を作成する場合、および暗号化データを格納または取得する場合は、キーストアがオープンしている必要があります。キーストアは、オープンするとすべてのセッションで使用可能になり、明示的にクローズするか、データベースが停止されるまではオープンしたままになります。

透過的データ暗号化では、次のタイプの暗号化アルゴリズム(Advanced Encryption Standard (AES)アルゴリズム、Triple Data Encryption Standard (3DES)アルゴリズムなど)を含む、業界標準の暗号化アルゴリズムがサポートされています。

- Advanced Encryption Standard(AES)
- ARIA
- GHOST
- SEED
- Triple Data Encryption Standard

サポートされる暗号化アルゴリズムの詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

暗号化キーの長さはアルゴリズム名で示されています。たとえば、AES128アルゴリズムでは128ビットのキーが使用されます。表

領域の作成時に使用するアルゴリズムを指定し、異なる表領域で別々のアルゴリズムを使用できます。理論上は、キーの長さが長くなるほどセキュリティが強化されますが、その分、CPUオーバーヘッドがかかります。アルゴリズムをCREATE TABLESPACE文で指定しなかった場合、AES128がデフォルトです。表領域の暗号化にはディスク領域オーバーヘッドは発生しません。

暗号化された表領域を作成した後で、ALTER TABLESPACE文を使用して複合化したり、キーを変更できます。また、ALTER TABLESPACE文を使用して、暗号化されていない表を暗号化することもできます。

制限事項

暗号化された表領域の制限事項は、次のとおりです。

- 暗号化された表領域は、別のデータベースにトランスポートする際に制限を受けます。「データのトランスポートに関する一般的な制限事項」を参照してください。
- 暗号化された表領域を使用してデータベースをリカバリする際(SHUTDOWN ABORTまたはデータベース・インスタンスを停止させる致命的なエラーの後など)には、リカバリ・プロセスがデータ・ブロックおよびREDOを復号化できるように、データベースをマウントしてからデータベースをオープンするまでにキーストアをオープンする必要があります。

また、透過的データ暗号化に関する一般的な制限事項は、『Oracle Database Advanced Securityガイド』を参照してください。

関連項目:

- 透過的データ暗号化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。
- 表領域全体を暗号化する代替方法は、「機密データを格納する列の暗号化」を参照してください
- Oracle Real Application Clusters環境でのキーストアの使用方法は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。
- CREATE TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 暗号化された表領域

13.2.5.2 暗号化された表領域の作成

暗号化された表領域を作成して、未承認のアクセスに対してデータを保護できます。

表領域を暗号化するには、COMPATIBLE初期化パラメータを11.2.0以上に設定した状態でデータベースをオープンする必要があります。表領域を作成できるユーザーは、暗号化された表領域も作成できます。

例

暗号化された表領域を作成するには:

- ENCRYPTION句を指定してCREATE TABLESPACE文を実行します。

次の文では、デフォルトの暗号化アルゴリズムを使用して暗号化された表領域を作成しています。

```
CREATE TABLESPACE securespace
DATAFILE '/u01/app/oracle/oradata/orcl/secure01.dbf' SIZE 100M
ENCRYPTION ENCRYPT;
```

次の文では、AES256アルゴリズムを使用して同じ表領域を作成しています。

```
CREATE TABLESPACE securespace
DATAFILE '/u01/app/oracle/oradata/orcl/secure01.dbf' SIZE 100M
ENCRYPTION USING 'AES256' ENCRYPT;
```

関連項目:

Oracle Database Advanced Securityガイド

親トピック: 暗号化された表領域

13.2.5.3 暗号化された表領域に関する情報の表示

DBA_TABLESPACESおよびUSER_TABLESPACESデータ・ディクショナリ・ビューを問い合せて、暗号化された表領域に関する情報を取得できます。

DBA_TABLESPACESとUSER_TABLESPACESのデータ・ディクショナリ・ビューには、ENCRYPTEDという列が含まれています。表領域が暗号化されている場合は、この列にYESと表示されます。

V$ENCRYPTED_TABLESPACESビューには、現在暗号化されているすべての表領域がリストされます。次の問合せでは、暗号化された表領域の名前と暗号化アルゴリズムが表示されます。

```
SELECT t.name, e.encryptionalg algorithm
FROM  v$tablespace t, v$encrypted_tablespaces e
WHERE t.ts# = e.ts#;

NAME                           ALGORITHM
------------------------------ ---------
SECURESPACE                    AES256
```

ノート:

既存の表領域を暗号化された表領域に変換できます。

関連項目:

暗号化表領域への既存の表領域の変換の詳細は、『Oracle Database Advanced Securityガイド』を参照してください

親トピック: 暗号化された表領域

13.2.6 一時表領域

一時表領域によって、メモリー内に収まらない複数のソート操作の同時実行性が向上します。また、一時表領域によって、ソート中の領域管理操作の効率も向上します。

- 一時表領域について
  一時表領域には、セッションの間のみ存続する一時データが格納されます。一時表領域を使用すると、メモリーに格納できない複数のソート操作の同時実行性を改善し、ソート時の領域管理操作の効率を改善できます。
- ローカル管理の一時表領域の作成
  ローカル管理表領域では、領域の管理がより簡単で効率的であるため、一時表領域には理想的です。
- bigfile一時表領域の作成
  通常の表領域と同様に、単一ファイル(bigfile)の一時表領域を作成できます。
- 一時表領域の領域使用情報の表示
  DBA_TEMP_FREE_SPACEディクショナリ・ビューには、各一時表領域の領域使用に関する情報が表示されます。

親トピック: 表領域の作成

13.2.6.1 一時表領域について

一時表領域には、セッションの間のみ存続する一時データが格納されます。一時表領域を使用すると、メモリーに格納できない複数のソート操作の同時実行性を改善し、ソート時の領域管理操作の効率を改善できます。

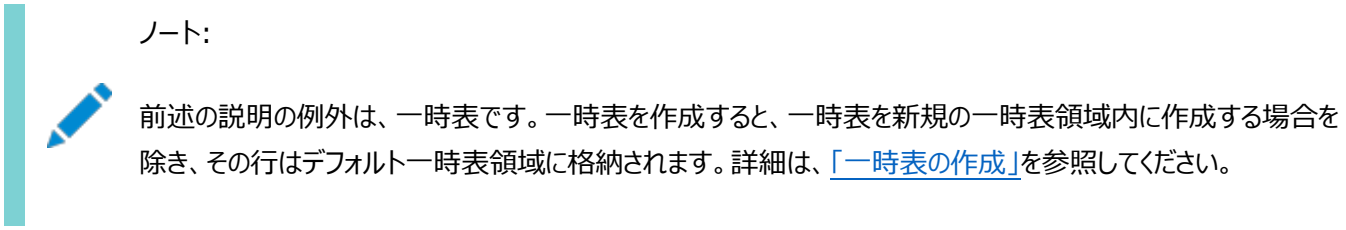
一時表領域は、次の情報を格納するために使用します。

- 中間ソート結果
- 一時表と一時索引
- 一時LOB
- 一時Bツリー

特定のインスタンスのソート操作はすべて、一時表領域内の1つのソート・セグメントを共有し、ソート・セグメントは、一時領域を必要とするソート操作を実行する各インスタンスに存在します。ソート・セグメントは、一時表領域を使用してソート処理を実行する最初の文によってインスタンスの起動後に作成され、停止時にのみ解放されます。

デフォルトでは、新規にOracle Databaseをインストールするたびに、TEMPという単一の一時表領域が作成されます。追加の一時表領域は、CREATE TABLESPACE文で作成できます。一時表領域を各データベース・ユーザーに割り当てるには、CREATE USER文またはALTER USER文を使用します。複数のユーザーが単一の一時表領域を共有できます。

一時表領域には、オブジェクトを明示的に作成できません。



ノート:

前述の説明の例外は、一時表です。一時表を作成すると、一時表を新規の一時表領域内に作成する場合を除き、その行はデフォルト一時表領域に格納されます。詳細は、「一時表の作成」を参照してください。

Oracle Database 12cリリース2 (12.2)以降では、ローカル一時表領域を使用できます。ローカル一時表領域には、すべてのデータベース・インスタンスの共有されない個別の一時ファイルが格納されます。ローカル一時表領域は、ソート、ハッシュ集約および結合に関連する問合せなど、SQL文の一時的な結果を自動書出しするためのみに使用されます。これらの結果にはインスタンス内でのみアクセスできます。これに対して、共有一時表領域は共有ディスクに存在し、すべてのインスタンスで使用可能です。ローカル一時表領域を作成するには、CREATE LOCAL TEMPORARY TABLESPACE文を使用します。共有一時表領域はOracle Databaseの以前のリリースでも使用可能で、「一時表領域」と呼ばれていました。この『*Oracle Database*管理者ガイド』では、特に記載がないかぎり、「一時表領域」という用語は共有一時表領域を意味します。

デフォルト一時表領域

一時表領域が明示的に割り当てられていないユーザーは、データベースのデフォルト一時表領域(新規インストールではTEMP)を使用します。データベースのデフォルト一時表領域は次のコマンドで変更できます。

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE tablespace_name;
```

データベースの現行のデフォルト一時表領域を判断するには、次の問合せを実行します。

```
SELECT PROPERTY_NAME, PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE
   PROPERTY_NAME='DEFAULT_TEMP_TABLESPACE';
PROPERTY_NAME                  PROPERTY_VALUE
------------------------------ ------------------------------
DEFAULT_TEMP_TABLESPACE        TEMP
```

一時表領域の領域割当て

一時表領域のソート・セグメントの領域割当てと割当て解除は、V$SORT_SEGMENTビューを使用して表示できます。V$TEMPSEG_USAGEビューでは、そのセグメント内の現行のソート・ユーザーが識別されます。

一時表領域を使用するソート操作が完了しても、ソート・セグメントに割り当てられたエクステントの割当ては解除されず、エクステントには、使用可能マークおよび再利用可能マークが付けられます。DBA_TEMP_FREE_SPACEビューには、各一時表領域の割当済領域の合計と空き領域が表示されます。詳細は、「一時表領域の領域使用情報の表示」を参照してください。大量の未使用領域があるローカル管理の一時表領域は手動で縮小できます。詳細は、「ローカル管理の一時表領域の縮小」を参照してください。

関連項目:

- ユーザーの作成方法および一時表領域の割当て方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- ローカル一時表領域、共有一時表領域およびデフォルト一時表領域の詳細は、『Oracle Database概要』を参照してください
- V$SORT_SEGMENT、V$TEMPSEG_USAGEおよびDBA_TEMP_FREE_SPACEの各ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- ソートのチューニングの説明は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。
- ローカル一時表領域については、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

親トピック: 一時表領域

13.2.6.2 ローカル管理の一時表領域の作成

ローカル管理表領域では、領域の管理がより簡単で効率的であるため、一時表領域には理想的です。

ローカル管理の一時表領域では、一時表領域外のデータを変更せず、一時表領域データのREDOを生成しない一時ファイルが使用されます。このため、読取り専用データベースまたはスタンバイ・データベースでディスク上ソート操作を実行できます。

一時ファイルの情報を表示するには、データファイルの場合とは異なるビューも使用します。V$TEMPFILEおよびDBA_TEMP_FILESビューは、V$DATAFILEおよびDBA_DATA_FILESビューに相当します。

ローカル管理の一時表領域を作成するには、CREATE TEMPORARY TABLESPACE文を使用します。この文を発行するには、CREATE TABLESPACEシステム権限が必要です。

次の文では、各エクステントが16MBの一時表領域が作成されます。16MBの各エクステント(標準ブロック・サイズが2KBのときは8000個のブロックに相当)は、このファイルのビットマップに1ビットで表されます。

```
CREATE TEMPORARY TABLESPACE lmtemp TEMPFILE '/u02/oracle/data/lmtemp01.dbf'
     SIZE 20M REUSE
     EXTENT MANAGEMENT LOCAL UNIFORM SIZE 16M;
```

すべての一時表領域は均一サイズのローカル管理エクステントを使用して作成されるため、一時表領域の場合、エクステント管理句はオプションです。EXTENT SIZE句でエクステント・サイズを指定した場合は、それが使用されます。指定しない場合は、Oracle Databaseで表領域サイズおよびファイル・サイズを使用してデフォルトのエクステント・サイズを決定します。

ノート:

オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。使用しているシステムでデータベースがこのように一時ファイルを割り当てているかどうかは、オペレーティング・システムのマニュアルを参照して判断してください。

親トピック: 一時表領域

13.2.6.3 bigfile一時表領域の作成

通常の表領域と同様に、単一ファイル(大型ファイル)の一時表領域を作成できます。

bigfile一時表領域を作成するには、次のようにします。

- CREATE BIGFILE TEMPORARY TABLESPACE文を実行して、単一の一時ファイル表領域を作成します。

bigfile表領域の詳細は、「bigfile表領域の作成」および「bigfile表領域の変更」の項を参照してください。ただし、データファイルのかわりに一時ファイルを使用する一時表領域を作成する点を考慮に入れてください。

親トピック: 一時表領域

13.2.6.4 一時表領域の領域使用情報の表示

DBA_TEMP_FREE_SPACEディクショナリ・ビューには、各一時表領域の領域使用に関する情報が表示されます。

この情報には、割当て済領域と空き領域が含まれます。これらの統計を表示するには、次の文を使用してこのビューを問い合せます。

```
SELECT * from DBA_TEMP_FREE_SPACE;

TABLESPACE_NAMETABLESPACE_SIZE ALLOCATED_SPACE FREE_SPACE
---------------------------------- --------------- --------------- ----------
TEMP25060966425060966424956108 8
```

親トピック: 一時表領域

13.2.7 一時表領域グループ

一時表領域グループは、データベースに対してデフォルトの一時表領域として割り当てられる表領域グループです。

- 複数の一時表領域: 表領域グループの使用
  表領域グループを使用して複数の表領域から一時領域を消費できます。単一の一時表領域ではなく表領域グループを使用することによって、ソート(特に多数のパーティションがある表でのソート)の結果を保持するのに1つの表領域では不十分な場合に発生する問題を回避できます。パラレル実行のサーバーで表領域グループを使用すると、1回のパラレル操作で複数の一時表領域を使用できます。
- 表領域グループの作成
  表領域グループは、CREATE TEMPORARY TABLESPACE文またはALTER TABLESPACE文にTABLESPACE GROUP句を指定したときに、指定した表領域グループが存在していない場合に暗黙的に作成されます。
- 表領域グループのメンバーの変更
  表領域を既存の表領域グループに追加するには、CREATE TEMPORARY TABLESPACE文またはALTER TABLESPACE文のTABLESPACE GROUP句で既存の表領域グループ名を指定します。

- デフォルト一時表領域としての表領域グループの割当て
  ALTER DATABASE...DEFAULT TEMPORARY TABLESPACE文を使用して、表領域グループをデータベースのデフォルト一時表領域として割り当てます。

親トピック: 表領域の作成

13.2.7.1 複数の一時表領域: 表領域グループの使用

ユーザーは、一時表領域グループを使用して複数の表領域から一時領域を消費できます。単一の一時表領域ではなく表領域グループを使用することによって、ソート(特に多数のパーティションがある表でのソート)の結果を保持するのに1つの表領域では不十分な場合に発生する問題を回避できます。パラレル実行のサーバーで表領域グループを使用すると、1回のパラレル操作で複数の一時表領域を使用できます。

表領域グループには、次の特性があります。

- これには1つ以上の表領域が含まれます。1つのグループに含まれる表領域の数に明示的な制限はありません。
- 表領域のネームスペースを共有するため、表領域グループにはグループ内の表領域と同じ名前は付けられません。
- 表領域グループ名は、データベースにデフォルト一時表領域を割り当てるとき、またはユーザーに一時表領域を割り当てるときに表領域名が表示される場所に指定できます。

表領域グループは明示的に作成しません。表領域グループは、最初の一時表領域がグループに割り当てられると暗黙的に作成されます。また、表領域グループに含まれる最後の一時表領域がグループから削除されると、その表領域グループが削除されます。

DBA_TABLESPACE_GROUPSビューには、表領域グループとそのメンバーの表領域がリスト表示されます。

関連項目:

一時表領域または表領域グループのユーザーへの割当ての詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: 一時表領域グループ

13.2.7.2 表領域グループの作成

表領域グループは、CREATE TEMPORARY TABLESPACE文またはALTER TABLESPACE文にTABLESPACE GROUP句を指定したときに、指定した表領域グループが存在していない場合に暗黙的に作成されます。

たとえば、group1およびgroup2が両方とも存在しない場合は、次の文によってこの2つのグループが作成され、それぞれのグループには指定の表領域のみがメンバーとして含まれます。

```
CREATE TEMPORARY TABLESPACE lmtemp2 TEMPFILE '/u02/oracle/data/lmtemp201.dbf'
     SIZE 50M
     TABLESPACE GROUP group1;
ALTER TABLESPACE lmtemp TABLESPACE GROUP group2;
```

親トピック: 一時表領域グループ

13.2.7.3 表領域グループのメンバーの変更

表領域を既存の表領域グループに追加するには、CREATE TEMPORARY TABLESPACE文またはALTER TABLESPACE文のTABLESPACE GROUP句で既存の表領域グループ名を指定します。

たとえば、次の文は表領域を既存のグループに追加します。これにより、表領域lmtemp3を作成してgroup1に追加します。その結果、group1には表領域lmtemp2とlmtemp3が含まれます。

```
CREATE TEMPORARY TABLESPACE lmtemp3 TEMPFILE '/u02/oracle/data/lmtemp301.dbf'
     SIZE 25M
     TABLESPACE GROUP group1;
```

次の文も表領域を既存のグループに追加しますが、この場合、表領域lmtemp2はすでにgroup1に属しているため、実質的にはgroup1からgroup2への移動となります。

```
ALTER TABLESPACE lmtemp2 TABLESPACE GROUP group2;
```

この結果、group2にはlmtempとlmtemp2が含まれ、group1にはtmtemp3のみが含まれます。

次の文を使用すると、表領域をグループから削除できます。

```
ALTER TABLESPACE lmtemp3 TABLESPACE GROUP '';
```

これによって、表領域lmtemp3はどのグループにも属さなくなります。さらに、group1に属するメンバーがなくなるため、group1は暗黙的に削除されます。

親トピック: 一時表領域グループ

13.2.7.4 デフォルト一時表領域としての表領域グループの割当て

ALTER DATABASE...DEFAULT TEMPORARY TABLESPACE文を使用して、表領域グループをデータベースのデフォルト一時表領域として割り当てます。

たとえば:

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE group2;
```

これで、明示的に一時表領域が割り当てられていないユーザーは表領域lmtempとlmtemp2を使用することになります。

表領域グループがデフォルト一時表領域に指定されている場合、そのグループのメンバーの表領域は削除できません。メンバーの表領域を削除するには、最初にその表領域を表領域グループから削除する必要があります。同様に、単一の一時表領域がデフォルト一時表領域に指定されている場合は、その一時表領域は削除できません。

親トピック: 一時表領域グループ

13.3 インメモリー列ストアへの表領域の格納の検討

表領域の作成または変更時に、インメモリー列ストアに対して表領域を有効にできます。インメモリー列ストアに対して表領域が有効になると、表領域内のすべての表はデフォルトでインメモリー列ストアに対して有効になります。

ノート:

この機能は、Oracle Database 12c リリース 1 (12.1.0.2)以降で使用可能です。

インメモリー列ストアは、システム・グローバル領域(SGA)のオプション部分で、高速スキャン用に最適化された表、表パーティション、その他のデータベース・オブジェクトのコピーが格納されます。インメモリー列ストアでは、表データがSGAに行ではなく列ごとに格納されます。

関連項目:

「Oracle Database In-Memoryによる問合せパフォーマンスの向上」

親トピック: 表領域の管理

13.4 表領域の非標準のブロック・サイズの指定

DB_BLOCK_SIZE初期化パラメータで指定された標準のデータベース・ブロック・サイズとは異なるブロック・サイズの表領域を作成できます。この機能によって、ブロック・サイズの異なる表領域をデータベース間でトランスポートできます。

データベース標準のブロック・サイズとは異なるブロック・サイズを持つ表領域を作成するには、次のようにします。

- CREATE TABLESPACE文のBLOCKSIZE句を使用します。

BLOCKSIZE句を正しく実行するためには、DB_CACHE_SIZEと、少なくとも1つのDB_nK_CACHE_SIZE初期化パラメータをすでに設定している必要があります。さらに、BLOCKSIZE句で指定する整数を1つのDB_nK_CACHE_SIZEパラメータの設定に対応付ける必要があります。冗長な指定になりますが、BLOCKSIZEをDB_BLOCK_SIZE初期化パラメータで指定されている標準のブロック・サイズと同じに指定することも可能です。

次の文は表領域lmtbsbを作成しますが、ブロック・サイズをDB_BLOCK_SIZE初期化パラメータで指定されている標準のデータベース・ブロック・サイズとは異なるサイズにします。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K
    BLOCKSIZE 8K;
```

関連項目:

- 「データベース・ブロック・サイズの指定」
- DB_CACHE_SIZEおよびDB_nK_CACHE_SIZEのパラメータ設定の詳細は、「バッファ・キャッシュ初期化パラメータの設定」を参照してください
- 「データベース間での表領域のトランスポート」

親トピック: 表領域の管理

13.5 REDOレコードの書込みの制御

一部のデータベース操作については、データベースでREDOレコードを生成するかどうかを制御できます。

REDOを使用しないと、メディア・リカバリはできません。ただし、REDOの生成を抑制するとパフォーマンスが改善されるため、簡単にリカバリできる操作に適している場合があります。たとえば、CREATE TABLE...AS SELECT文は、データベース障害やインスタンス障害が発生した場合に操作を繰り返すことができます。

表領域内のオブジェクトに対してこのような操作を実行するときにREDOを抑制するには、次のようにします。

- CREATE TABLESPACE文でNOLOGGING句を指定します。

この句を指定しないか、かわりにLOGGINGを指定した場合は、表領域内のオブジェクトに変更が行われるとREDOが生成されます。一時セグメントや一時表領域の場合は、ロギング属性に関係なくREDOは生成されません。

表領域レベルで指定するロギング属性は、その表領域内で作成されるオブジェクトのデフォルト属性になります。このデフォルトのロギング属性は、CREATE TABLE文を使用するなど、スキーマ・オブジェクト・レベルでLOGGINGやNOLOGGINGを指定するこ

とで上書きできます。

スタンバイ・データベースがある場合は、NOLOGGING句を指定すると、そのスタンバイ・データベースの可用性と精度に問題が生じます。この問題を克服するために、FORCE LOGGINGモードを指定できます。CREATE TABLESPACE文にFORCE LOGGING句を指定すると、表領域内のオブジェクトを変更するすべての操作について、REDOレコードを強制的に生成させることができます。これにより、オブジェクト・レベルでの指定が上書きされます。

FORCE LOGGINGモードの表領域を別のデータベースにトランスポートすると、新しい表領域ではFORCE LOGGINGモードは維持されません。

関連項目:

- NOLOGGINGモードで実行できる操作の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- FORCE LOGGINGモードの詳細と、CREATE DATABASE文でFORCE LOGGING句を使用する効果の詳細は、「FORCE LOGGINGモードの指定」を参照してください。

親トピック: 表領域の管理

13.6 表領域の可用性の変更

オンラインの表領域をオフライン化すると、一般的な使用を一時的に禁止にできます。データベースの残りの部分はオープンしていて使用可能であり、ユーザーはデータにアクセスできます。逆に、オフライン状態の表領域をオンライン化して、データベース・ユーザーがその表領域内のスキーマ・オブジェクトを使用できるようにすることもできます。表領域の可用性を変更するには、データベースをオープンする必要があります。

表領域の可用性を変更するには、ALTER TABLESPACE文を使用します。そのためには、ALTER TABLESPACEまたはMANAGE TABLESPACEシステム権限が必要です。

- 表領域のオフライン化
  表領域をオフラインにすると、通常アクセスができなくなります。
- 表領域のオンライン化
  データベースがオープンされている場合は、いつでもOracle Database内の任意の表領域をオンライン化できます。通常、表領域は、データベース・ユーザーがその中のデータを使用できるようにオンラインになっています。

関連項目:

表領域内の各データファイルの可用性を変更する方法の詳細は、「データファイルの可用性の変更」を参照してください

親トピック: 表領域の管理

13.6.1 表領域のオフライン化

表領域をオフラインにすると、通常アクセスができなくなります。

次のような場合、表領域をオフライン化する場合があります。

- データベースの一部のみを使用できないようにし、残りの部分には正常にアクセスできるようにする場合
- 表領域のバックアップをオフラインで実行する場合(ただし、表領域はオンラインでも、使用中でもバックアップ可能です)
- アプリケーションの更新またはメンテナンスを行う間、アプリケーションとその表グループを一時的にアクセス不可にする場

合

- 表領域のデータファイルの名前の変更または再配置をする場合

  詳細は、「データファイルの名前変更と再配置」を参照してください。

表領域をオフライン化するには、次のようにします。

- OFFLINE句を指定してALTER TABLESPACE文を実行します。

表領域をオフライン化すると、その関連ファイルがすべてオフライン化されます。

次の表領域はオフライン化できません。

- SYSTEM
- UNDO表領域
- 一時表領域

表領域をオフライン化する前に、その表領域がデフォルト表領域としてすでに割り当てられているユーザーの表領域割当ての変更を考慮してください。このようなユーザーは表領域がオフラインの間その中のオブジェクトにアクセスできないため、表領域割当ての変更をお薦めします。

ALTER TABLESPACE...OFFLINE文では、次のパラメータを指定できます。

句	説明
NORMAL	表領域のどのデータファイルにもエラー条件が存在していない場合は、この表領域を通常の方法でオフライン化できます。書込みエラーが発生していると、現時点では表領域のデータファイルをオフライン化することはできません。OFFLINE NORMAL を指定すると、データベースは表領域のデータファイルすべてのチェックポイントを取ってから、それらのファイルをオフライン化します。NORMAL はデフォルトです。
TEMPORARY	表領域の 1 つまたは複数のデータファイルについてエラー条件が存在している場合でも、表領域を一時的にオフライン化できます。OFFLINE TEMPORARY を指定すると、データベースはまだオフライン化されていないデータファイルのチェックポイントを取ってから、これらのファイルをオフライン化します。 オフラインになっているファイルがないときに表領域を一時的にオフライン化する場合は、表領域をオンラインに戻す前にメディア・リカバリを実行する必要はありません。しかし、表領域の 1 つまたは複数のファイルが書込みエラーのためにオフラインになっており、この表領域を一時的にオフライン化する場合は、表領域をオンラインに戻す前にリカバリする必要があります。
IMMEDIATE	データベースによりデータファイル上のチェックポイントがチェックされることなく、表領域をただちにオフラインにできます。OFFLINE IMMEDIATE を指定すると、表領域をオンライン化する前に表領域のメディア・リカバリが必要です。データベースが NOARCHIVELOG で稼働している場合は、表領域を即時にオフライン化することはできません。

ノート:

表領域をオフライン化する必要がある場合は、可能なかぎり NORMAL 句(デフォルト)を使用してください。この設定によって、不完全リカバリの後に ALTER DATABASE OPEN RESETLOGS 文を使用して REDO ログ順序をリセットした場合でも、表領域をオンラインに戻すためのリカバリが不要になることが保証されます。

TEMPORARYは、表領域を通常の方法でオフライン化できないときのみ指定してください。この場合、エラーのためにオフライン化されたファイルのみをリカバリする必要があり、その後に表領域をオンライン化できます。IMMEDIATEは、NORMAL設定とTEMPORARY設定を試した後にのみ指定してください。

次の例では、users表領域を通常の方法でオフライン化しています。

```
ALTER TABLESPACE users OFFLINE NORMAL;
```

親トピック: 表領域の可用性の変更

13.6.2 表領域のオンライン化

データベースがオープンされている場合は、いつでもOracle Database内の任意の表領域をオンライン化できます。通常、表領域は、データベース・ユーザーがその中のデータを使用できるようにオンラインになっています。

表領域をオンライン化するには、次のようにします。

- ONLINE句を指定してALTER TABLESPACE文を実行します。

オンライン化しようとする表領域が、正常に(ALTER TABLESPACE OFFLINE文のNORMAL句を使用して)オフライン化されていない場合は、最初にメディア・リカバリをしないかぎりオンライン化できません。メディア・リカバリを実行しないと、エラーが返されて表領域はオフラインのままになります。

たとえば、次の文はusers表領域をオンラインにします。

```
ALTER TABLESPACE users ONLINE;
```

関連項目:

メディア・リカバリの実行方法の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: 表領域の可用性の変更

13.7 読取り専用表領域の使用

表領域は読取り専用モードに設定できます。これにより、それに格納されているデータの更新が禁止されます。

- 読取り専用表領域について
  表領域を読取り専用にすると、表領域のデータファイルに対して書込み操作ができなくなります。
- 表領域を読取り専用にする方法
  表領域を読取り専用にするには、READ ONLY句を指定してALTER TABLESPACE文を使用します。
- 読取り専用表領域を書込み可能にする方法
  読取り専用表領域を書込み可能にすると、表領域のデータファイルに対して書込み操作が可能なります。
- WORMデバイスでの読取り専用表領域の作成
  読取り専用表領域は、CD-ROMまたはWORM (ライト・ワンス・リード・メニー)デバイスに作成できます。

- 読取り専用表領域内にあるデータファイルのオープンの遅延
  読取り専用表領域のデータファイルのオープンを、アクセスが試行されるまで遅延できます。

親トピック: 表領域の管理

13.7.1 読取り専用表領域について

表領域を読取り専用にすると、表領域のデータファイルに対して書込み操作ができなくなります。

読取り専用表領域の主な目的は、データベース内の大規模かつ静的部分のバックアップおよびリカバリを実行しなくて済むようにすることです。また、読取り専用表領域は、ユーザーが履歴データを変更できないように履歴データを完全に保護する手段でもあります。表領域を読取り専用にすると、その表領域内のすべての表はユーザーの更新権限レベルに関係なく更新できません。

ノート:

表領域は、それが作成されたデータベース内でしかオンライン化できないため、読取り専用にすること自体でアーカイブ要件やデータ公開要件を満たすことはできません。ただし、「データベース間での表領域のトランスポート」で説明するように、トランスポータブル表領域機能を使用すると、これらの要件を満たすことができます。

表や索引などの項目は読取り専用表領域から削除できますが、読取り専用表領域内のオブジェクトは作成または変更できません。ALTER TABLE...ADDまたはALTER TABLE...MODIFYなど、データ・ディクショナリ内のファイル記述を更新する文は実行できますが、新しい記述は表領域を読取り/書込み用にするまでは使用できません。表定義の変更時に、データ型がBLOBの列を追加することはできないということに注意してください。

読取り専用表領域は、他のデータベースにトランスポートすることもできます。読取り専用表領域は更新できないため、CD-ROMまたはWrite Once-Read Many(WORM)デバイスに格納できます。

関連項目:

「データベース間での表領域のトランスポート」

親トピック: 読取り専用表領域の使用

13.7.2 表領域を読取り専用にする方法

表領域を読取り専用にするには、READ ONLY句を指定してALTER TABLESPACE文を使用します。

すべての表領域は、最初は読取り/書込み用として作成されます。そのためには、ALTER TABLESPACEまたはMANAGE TABLESPACEシステム権限が必要です。

表領域を読取り専用にするには、あらかじめ次の条件を満たす必要があります。

- 表領域は必ずオンラインにする。これにより、表領域に適用する必要のあるUNDO情報がないことが保証されます。
- 表領域をアクティブなUNDO表領域またはSYSTEM表領域にしない。
- 表領域を現行のオンライン・バックアップに含めない(オンライン・バックアップは、終了時に表領域内にあるすべてのデータファイルのヘッダー・ファイルを更新するためです)。
- 表領域を一時表領域にできない。

表領域を読取り専用に変更するには、次のようにします。

- ALTER TABLESPACE文でREAD ONLY句を使用します。

たとえば、次の文はflights表領域を読取り専用にします。

```
ALTER TABLESPACE flights READ ONLY;
```

読取り専用表領域のデータにアクセスする際のパフォーマンスを向上させるため、表領域を読取り専用にする直前に、表領域内の表のブロックすべてにアクセスする問合せを発行することをお薦めします。各表に対してSELECT COUNT (*)などの単純な問合せを実行しておくと、それ以降、表領域のデータ・ブロックに最も効率的にアクセスできるようになります。これによって、最後にブロックを変更したトランザクションの状態をデータベースが確認する必要がなくなるからです。

データベースのトランザクション処理中に、ALTER TABLESPACE...READ ONLY文を発行できます。この文が発行されると、表領域は一時的に読取り専用モードになり、ALTERコマンドは、既存のトランザクションがコミットまたはロールバックにより完了するまで待機します。表領域に対するそれ以後のDML操作は許可されず、DML文でそれ以降の変更をしようとすると、エラーが返されます。

ALTER TABLESPACE...READ ONLY文は、戻る前に、表領域に対する変更が保留中またはコミット解除されたトランザクション、およびこの文の発行前に開始されたトランザクションがコミットまたはロールバックされるのを待機します。文の発行前に開始されたトランザクションがアクティブなままであっても、表領域に対する変更をロールバックしてセーブポイントまでロールバックすると、文はこのアクティブ・トランザクションを待機しなくなります。

ALTER TABLESPACE文の完了までに長時間かかる場合は、読取り専用状態になるのを妨げているトランザクションを識別できます。次に、それらのトランザクションの所有者に通知し、必要に応じてトランザクションを終了させるかどうかを決定できます。

次の例では、ALTER TABLESPACE...READ ONLY文に対応するトランザクション・エントリが識別され、そのセッション・アドレス(saddr)が表示されます。

```
SELECT SQL_TEXT, SADDR
    FROM V$SQLAREA,V$SESSION
    WHERE V$SQLAREA.ADDRESS = V$SESSION.SQL_ADDRESS
        AND SQL_TEXT LIKE 'alter tablespace%';
SQL_TEXT                                 SADDR
---------------------------------------- --------
alter tablespace tbs1 read only          80034AF0
```

各アクティブ・トランザクションの開始システム変更番号(SCN)は、V$TRANSACTIONビューに格納されています。このビューを開始SCNの昇順でソートして表示すると、トランザクションが実行順にリストされます。前述の例の場合は、読取り専用文のトランザクション・エントリのセッション・アドレスがわかっているので、V$TRANSACTIONビューで特定できます。開始SCNよりも小さい番号を持つトランザクション(以前に実行されたトランザクションを示します)はすべて、表領域の停止とその後の読取り専用状態になるのを妨げている可能性があります。

```
SELECT SES_ADDR, START_SCNB
    FROM V$TRANSACTION
    ORDER BY START_SCNB;
SES_ADDR START_SCNB
-------- ----------
800352A0       3621   --> waiting on this txn
80035A50       3623   --> waiting on this txn
80034AF0       3628   --> this is the ALTER TABLESPACE statement
80037910       3629   --> don't care about this txn
```

この時点で、ブロックしているトランザクションの所有者を見つけることができます。

```
SELECT T.SES_ADDR, S.USERNAME, S.MACHINE
   FROM V$SESSION S, V$TRANSACTION T
```

```
   WHERE T.SES_ADDR = S.SADDR
   ORDER BY T.SES_ADDR
SES_ADDR USERNAME             MACHINE
-------- -------------------- --------------------
800352A0 DAVIDB               DAVIDBLAP             --> Contact this user
80035A50 MIKEL                LAB61                 --> Contact this user
80034AF0 DBA01                STEVEFLAP
80037910 NICKD                NICKDLAP
```

表領域を読取り専用にした後は、その表領域のバックアップをただちに作成することをお薦めします。表領域は読取り専用になっているかぎり変更できないため、それ以後のバックアップは不要です。

関連項目:

『Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド』

親トピック: 読取り専用表領域の使用

13.7.3 読取り専用表領域を書込み可能にする方法

読取り専用表領域を書込み可能にすると、表領域のデータファイルに対して書込み操作が可能になります。

そのためには、ALTER TABLESPACEまたはMANAGE TABLESPACEシステム権限が必要です。

表領域を書込み操作可能に変更するには、次のようにします。

- ALTER TABLESPACE文でREAD WRITEキーワードを使用します。

表領域を読取り/書込み用にするには、前提条件として、表領域のみでなく、そのすべてのデータファイルをオンライン化する必要があります。データファイルをオンライン化するには、ALTER DATABASE 文のDATAFILE...ONLINE句を使用します。データファイルの現行の状態を確認するには、V$DATAFILEビューを使用します。

たとえば、次の文はflights表領域を書込み可能にします。

```
ALTER TABLESPACE flights READ WRITE;
```

読取り専用表領域を書込み可能にすると、データファイルの制御ファイル・エントリが更新されるため、読取り専用バージョンのデータファイルをリカバリの開始点として使用できます。

親トピック: 読取り専用表領域の使用

13.7.4 WORMデバイスでの読取り専用表領域の作成

読取り専用表領域は、CD-ROMまたはWORM (ライト・ワンス・リード・メニー)デバイスに作成できます。

CD-ROMまたはWORMデバイスに読取り専用表領域を作成するには、これらのステップに従います。

1. 別のデバイスに書込み可能表領域を作成します。その表領域に属するオブジェクトを作成して、データを挿入します。
2. 表領域を読取り専用に変更します。
3. 表領域のデータファイルをWORMデバイスにコピーします。ファイルをコピーするには、オペレーティング・システムのコマンドを使用します。
4. 表領域をオフライン化します。
5. データファイルの名前を、WORMデバイスにコピーしたファイルと一致するように変更します。これには、RENAME DATAFILE句を指定したALTER TABLESPACE文を使用します。データファイルの名前を変更すると、制御ファイルに記述されているこれらのファイルの名前も変更されます。

6. 表領域をオンライン化します。

親トピック: 読取り専用表領域の使用

13.7.5 読取り専用表領域内にあるデータファイルのオープンの遅延

読取り専用表領域のデータファイルのオープンを、アクセスが試行されるまで遅延できます。

大規模データベースのほとんどが、アクセス速度の遅いデバイスや階層形式の記憶デバイス上にある読取り専用表領域に格納されている場合は、READ_ONLY_OPEN_DELAYED初期化パラメータをTRUEに設定することを検討する必要があります。これにより、読取り専用表領域内のデータファイルは、そこに格納されたデータの読取り試行時に初めてアクセスされるため、主にデータベースのオープンなど、特定の操作が高速になります。

READ_ONLY_OPEN_DELAYED=TRUEに設定すると、次のような副次的な影響があります。

- オープン時には、読取り専用の欠落ファイルや不良ファイルが検出されません。アクセスしようとしたときに初めて検出されます。
- ALTER SYSTEM CHECK DATAFILESでは、読取り専用ファイルはチェックされません。
- ALTER TABLESPACE...ONLINEおよびALTER DATABASE DATAFILE...ONLINEでは、読取り専用ファイルはチェックされません。最初のアクセス時にのみチェックされます。
- V$RECOVER_FILE、V$BACKUPおよびV$DATAFILE_HEADERは、読取り専用ファイルにアクセスしません。読取り専用ファイルは結果リスト上に「DELAYED OPEN」というエラーで示され、他の列の値は0(ゼロ)になります。
- V$DATAFILEは読取り専用ファイルにアクセスしません。読取り専用ファイルにはサイズ「0」がリストされます。
- V$RECOVERY_LOGは読取り専用ファイルにアクセスしません。リカバリに必要な可能性があるログは、リストに追加されません。
- ALTER DATABASE NOARCHIVELOGは読取り専用ファイルにアクセスしません。リカバリが必要な読取り専用ファイルがあっても続行されます。

ノート:

- RECOVER DATABASE および ALTER DATABASE OPEN RESETLOGS は、パラメータ値に関係なく、すべての読取り専用データファイルに引き続きアクセスします。これらの操作で読取り専用ファイルへのアクセスを回避する場合は、該当ファイルをオフライン化します。
- バックアップ制御ファイルを使用すると、一部のファイルの読取り専用状態が不正確になる場合があります。これにより、これらの操作の一部で予期しない結果が返されることがあります。この状況には注意が必要です。

親トピック: 読取り専用表領域の使用

13.8 表領域の変更とメンテナンス

データファイルと一時ファイルの追加などの作業を行って、表領域を変更およびメンテナンスできます。

- 表領域のサイズの拡大
  表領域のサイズを増やすには、表領域のデータファイルのサイズを増やすか、またはデータファイルを追加します。

- ローカル管理表領域の変更
  ローカル管理表領域に対して、データファイルの追加、可用性の変更、読取り専用または読取り/書込みへの変更、名前の変更、または自動拡張の有効化/無効化を行うことができます。
- bigfile表領域の変更
  bigfile表領域のサイズ変更または自動拡張が可能です。
- ローカル管理の一時表領域の変更
  ローカル管理の一時表領域を変更して、一時ファイルの追加、一時ファイルのオフライン化または一時ファイルのオンライン化を行うことができます。
- ローカル管理の一時表領域の縮小
  ローカル管理の一時表領域を縮小して未使用領域を解放できます。

親トピック: 表領域の管理

13.8.1 表領域のサイズの拡大

表領域のサイズを増やすには、表領域のデータファイルのサイズを増やすか、またはデータファイルを追加します。

詳細は、「データファイルのサイズ変更」および「データファイルの作成および表領域への追加」を参照してください。

また、データファイルおよびbigfile表領域に対して自動ファイル拡張(AUTOEXTEND)を有効にできます。詳細は、「データファイルの自動拡張機能の使用可能および使用禁止」を参照してください。

親トピック: 表領域の変更とメンテナンス

13.8.2 ローカル管理表領域の変更

ローカル管理表領域に対して、データファイルの追加、可用性の変更、読取り専用または読取り/書込みへの変更、名前の変更、または自動拡張の有効化/無効化を行うことができます。

ローカル管理表領域をローカル管理の一時表領域に変更したり、セグメント領域の管理方法を変更することはできません。ローカル管理表領域では、使用可能エクステントを結合する必要はありません。ただし、次のような操作の場合は、ALTER TABLESPACE文をローカル管理表領域に対して使用できます。

- データファイルを追加する場合。たとえば:

```
ALTER TABLESPACE lmtbsb
   ADD DATAFILE '/u02/oracle/data/lmtbsb02.dbf' SIZE 1M;
```

- 表領域の可用性(ONLINE/OFFLINE)を変更する場合。
- 表領域を読取り専用または読取り/書込み用にする場合。
- データファイルの名前を変更したり、表領域内のデータファイルのサイズの自動拡張を使用可能または使用禁止にする場合。

関連項目:

- 「表領域の可用性の変更」
- 「読取り専用表領域について」
- 「データファイルおよび一時ファイルの管理」

親トピック: 表領域の変更とメンテナンス

13.8.3 bigfile表領域の変更

bigfile表領域のサイズを変更または自動拡張できます。

ALTER TABLESPACE文の次の2つの句は、bigfile表領域使用時におけるデータファイルの透過性をサポートします。

- RESIZE: RESIZE句を使用すると、bigfile表領域内の単一データファイルを参照せずに、そのデータファイルのサイズを絶対サイズに変更できます。たとえば:

  ```
  ALTER TABLESPACE bigtbs RESIZE 80G;
  ```

- AUTOEXTEND(ADD DATAFILE句の範囲外で使用):

  bigfile表領域では、ADD DATAFILE句の範囲外でAUTOEXTEND句を使用できます。たとえば:

  ```
  ALTER TABLESPACE bigtbs AUTOEXTEND ON NEXT 20G;
  ```

bigfile表領域に対してADD DATAFILE句を指定すると、エラーが発生します。

親トピック: 表領域の変更とメンテナンス

13.8.4 ローカル管理の一時表領域の変更

ローカル管理の一時表領域を変更して、一時ファイルの追加、一時ファイルのオフライン化または一時ファイルのオンライン化を行うことができます。

ノート:

ALTER TABLESPACE 文に TEMPORARY キーワードを指定して、ローカル管理の永続表領域をローカル管理の一時表領域に変更することはできません。ローカル管理の一時表領域を作成するには、CREATE TEMPORARY TABLESPACE 文を使用する必要があります。

次の例に示すとおり、ALTER TABLESPACEを使用すると、一時ファイルを追加したり、オフライン化またはオンライン化できます。

```
ALTER TABLESPACE lmtemp
   ADD TEMPFILE '/u02/oracle/data/lmtemp02.dbf' SIZE 18M REUSE;
ALTER TABLESPACE lmtemp TEMPFILE OFFLINE;
ALTER TABLESPACE lmtemp TEMPFILE ONLINE;
```

ノート:

一時表領域はオフライン化できません。かわりに、一時ファイルをオフライン化します。V$TEMPFILE ビューには、一時ファイルのオンライン化の状態が表示されます。

ALTER DATABASE文を使用すると、一時ファイルを変更できます。

次の文では、一時ファイルをオフラインにしてから、オンラインに戻しています。前述の例の最後の2つのALTER TABLESPACE文と同様に動作します。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' OFFLINE;
```

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' ONLINE;
```

次の文では、一時ファイルのサイズが変更されます。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' RESIZE 18M;
```

次の文では、一時ファイルが削除され、オペレーティング・システム・ファイルが削除されます。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' DROP
    INCLUDING DATAFILES;
```

この一時ファイルが属していた表領域は残ります。アラート・ログには、一時ファイルが削除されたことを示すメッセージが書き込まれます。オペレーティング・システム・エラーによってファイルが削除されなかった場合でも文は正常終了しますが、エラーを示すメッセージがアラート・ログに書き込まれます。

ALTER DATABASE文を使用して、既存の一時ファイルの自動拡張を使用可能または使用禁止にしたり、一時ファイル名を変更することもできます。必要な構文は、『Oracle Database SQL言語リファレンス』を参照してください。

ノート:

一時ファイルの名前を変更するには、一時ファイルをオフライン化し、オペレーティング・システムのコマンドを使用して、その一時ファイルを名前変更または再配置します。次に、ALTER DATABASE RENAME FILE コマンドを使用してデータベースの制御ファイルを更新します。

親トピック: 表領域の変更とメンテナンス

13.8.5 ローカル管理の一時表領域の縮小

ローカル管理の一時表領域を縮小して、未使用領域を解放できます。

データベースで大規模なソート操作を実行すると、一時表領域が増大し、ディスク領域の容量が大幅に占有される場合があります。ソート操作が完了しても余分になった領域は解放されず、使用可能マークと再利用可能マークが付けられるのみです。したがって、単一の大規模なソート操作を実行すると、ソート操作の完了後に大量の割当て済一時領域が未使用のままになります。このため、データベースでは、ローカル管理の一時表領域を縮小して未使用領域を解放できます。

一時表領域を縮小するには、次にようにします。

- ALTER TABLESPACE文のSHRINK SPACE句を使用します。

一時表領域の特定の一時ファイルを縮小するには、次のようにします。

- ALTER TABLESPACE文のSHRINK TEMPFILE句を使用します。

縮小すると、表領域または一時ファイルの他の属性を維持しながら可能なかぎり空き領域を解放します。オプションのKEEP句は、表領域または一時ファイルの最小サイズを定義します。

縮小はオンライン操作です。これは、ユーザー・セッションは必要に応じてソート・エクステントの割当てを継続でき、すでに実行中の問合せは影響を受けないことを意味します。

次の例では、20MBのサイズを確保しながら、ローカル管理の一時表領域のlmtmp1を縮小しています。

```
ALTER TABLESPACE lmtemp1 SHRINK SPACE KEEP 20M;
```

次の例では、ローカル管理の一時表領域lmtmp2の一時ファイルlmtemp02.dbfを縮小しています。KEEP句を省略してい

るため、データベースでは、一時ファイルを最小可能サイズまで縮小することを試みます。

```
ALTER TABLESPACE lmtemp2 SHRINK TEMPFILE '/u02/oracle/data/lmtemp02.dbf';
```

親トピック: 表領域の変更とメンテナンス

13.9 表領域の名前変更

永続表領域または一時表領域の名前は、ALTER TABLESPACE文のRENAME TO句を使用して変更できます。

たとえば、次の文はusers表領域の名前を変更します。

```
ALTER TABLESPACE users RENAME TO usersts;
```

表領域の名前を変更すると、データ・ディクショナリ、制御ファイルおよび(オンライン)データファイル・ヘッダー内でその表領域名への参照がすべて更新されます。表領域IDは変更されないため、たとえば、その表領域がユーザーのデフォルト表領域の場合、DBA_USERSビューには名前が変更された表領域がユーザーのデフォルト表領域として表示されます。

この文の操作では、次の点に注意してください。

- 名前を変更する表領域がSYSTEM表領域またはSYSAUX表領域の場合、名前は変更されず、エラーが発生します。
- 表領域内のデータファイルがオフラインの場合、または表領域がオフラインの場合、その表領域の名前は変更されず、エラーが発生します。
- 表領域が読取り専用の場合、データファイル・ヘッダーは更新されません。これは破損とはみなされませんが、データファイル・ヘッダーの名前が変更されなかったことを示すメッセージがアラート・ログに書き込まれます。データ・ディクショナリと制御ファイルは更新されます。
- 表領域がデフォルト一時表領域の場合は、データベース・プロパティ表内の対応するエントリが更新され、DATABASE_PROPERTIESビューに新しい名前が表示されます。
- 表領域がUNDO表領域で、次の条件が満たされると、サーバー・パラメータ・ファイル(SPFILE)の表領域名は新しい表領域名に変更されます。
    - サーバー・パラメータ・ファイルを使用して、インスタンスを起動した場合。
    - 表領域名がインスタンスに対してUNDO_TABLESPACEで指定されている場合。

  従来の初期化パラメータ・ファイル(PFILE)を使用している場合は、初期化パラメータ・ファイルを手動で変更する必要があることを示すメッセージがアラート・ログに書き込まれます。

親トピック: 表領域の管理

13.10 表領域の削除

表領域とその内容が不要になった場合は、その表領域と内容(表領域に含まれるセグメント)をデータベースから削除できます。

表領域を削除するには、DROP TABLESPACEシステム権限が必要です。

ノート:

削除された表領域のデータはリカバリできません。そのため、削除しようとしている表領域に含まれているデータはすべて、将来的に必要のないことを確認してください。また、表領域をデータベースから削除する直前および直後に、デー

タベースの完全バックアップを作成する必要があります。誤って表領域を削除した場合や、今後表領域を削除した後にデータベースに問題が発生した場合にデータベースをリカバリできるように、データベースをバックアップすることをお薦めします。

表領域を削除すると、対応付けられたデータベースの制御ファイル中のファイル・ポインタのみが削除されます。必要に応じて、削除された表領域を構成していたオペレーティング・システム・ファイル(データファイル)を削除するようにOracle Databaseに指示することもできます。表領域の削除と同時にデータファイルを削除するようにデータベースに指示しない場合は、後でオペレーティング・システムの適切なコマンドを使用して削除する必要があります。

アクティブなセグメントを含む表領域は削除できません。たとえば、表領域内の表が現在使用されている場合、またはコミットされていないトランザクションをロールバックする必要があるUNDOデータが表領域に含まれている場合、その表領域は削除できません。表領域はオンラインでもオフラインでもかまいませんが、削除する前にオフラインにすることをお薦めします。

表領域を削除するには、次のようにします。

- DROP TABLESPACE文を使用します。

、次の文はusers表領域を、その中のセグメントも含めて削除します。

```
DROP TABLESPACE users INCLUDING CONTENTS;
```

表領域が空の場合(表、ビューまたは他の構造が格納されていない場合)は、INCLUDING CONTENTS 句を指定する必要はありません。CASCADE CONSTRAINTS句を使用すると、表領域内の表の主キーと一意キーを参照する別の表領域の表から、すべての参照整合性制約を削除できます。

表領域の削除と同時に表領域に対応付けられたデータファイルを削除するには、INCLUDING CONTENTS AND DATAFILES句を使用します。次の文は、users表領域とそれに対応付けられているデータファイルを削除します。

```
DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;
```

アラート・ログには、削除された各データファイルのメッセージが書き込まれます。オペレーティング・システム・エラーによってファイルが削除されなかった場合でもDROP TABLESPACE文は正常終了しますが、エラーを示すメッセージがアラート・ログに書き込まれます。

関連項目:

「データファイルの削除」

親トピック: 表領域の管理

13.11 シャドウ表領域を使用した消失書込み保護の管理

データ・ブロックの消失書込みは、永続ストレージで書込みが行われなかったにもかかわらず、I/Oサブシステムでブロック書込みの完了が確認された場合に発生します。消失書込みに対する保護として、シャドウ消失書込み保護を使用できます。

- シャドウ消失書込み保護について
  データ・ブロックの消失書込みは、書込みは発生しなかったがI/Oサブシステムでブロック書込みの完了を確認した場合、またはブロックの前のイメージで現在のイメージが上書きされた場合に発生します。シャドウ消失書込み保護により、表領域または個々のデータファイルの消失書込みに対して保護できます。
- シャドウ消失書込み保護のためのシャドウ表領域の作成
  シャドウ消失書込み保護のためのシャドウ表領域を作成するには、LOST WRITE PROTECTION句を指定して

CREATE BIGFILE TABLESPACE文を発行します。

- データベースのシャドウ消失書込み保護の有効化
  マルチテナント・コンテナ・データベース(CDB)または非CDBのシャドウ消失書込み保護を有効にするには、ENABLE LOST WRITE PROTECTION句を指定したALTER DATABASE文を使用します。プラガブル・データベース(PDB)のシャドウ消失書込み保護を有効にするには、ENABLE LOST WRITE PROTECTION句を指定したALTER PLUGGABLE DATABASE文を使用します。
- 表領域とデータ・ファイルに対するシャドウ消失書込み保護の有効化
  表領域とデータ・ファイルに対してシャドウ消失書込み保護を有効化できます。
- データベースに対するシャドウ消失書込み保護の無効化
  マルチテナント・コンテナ・データベース(CDB)または非CDBのシャドウ消失書込み保護を無効化するには、DISABLE LOST WRITE PROTECTION句を指定してALTER DATABASE文を発行します。プラガブル・データベース(PDB)のシャドウ消失書込み保護を無効化するには、DISABLE LOST WRITE PROTECTION句を指定してALTER PLUGGABLE DATABASE文を発行します。
- シャドウ消失書込み保護の削除または一時停止
  表領域またはデータ・ファイルに対するシャドウ消失書込み保護を削除または一時停止できます。
- シャドウ表領域の削除
  シャドウ表領域はDROP TABLESPACE文を使用して削除できます。INCLUDING CONTENTS句を指定したDROP TABLESPACE文を使用すると、シャドウ表領域は内容とともに削除されます。INCLUDING CONTENTS句の指定なしでDROP TABLESPACE文を使用すると、シャドウ表領域の削除前に、その内容が別のシャドウ表領域に移動されます(この表領域が存在していて、十分な空き領域がある場合)。

親トピック: 表領域の管理

13.11.1 シャドウ消失書込み保護について

データ・ブロックの書込み欠落は、書込みが実際には行われていないのにブロック書込みの完了がI/Oサブシステムで認識される場合や、ブロックの前回イメージが現在のイメージを上書きする場合に発生します。シャドウ消失書込み保護により、表領域または個々のデータファイルの消失書込みに対して保護できます。

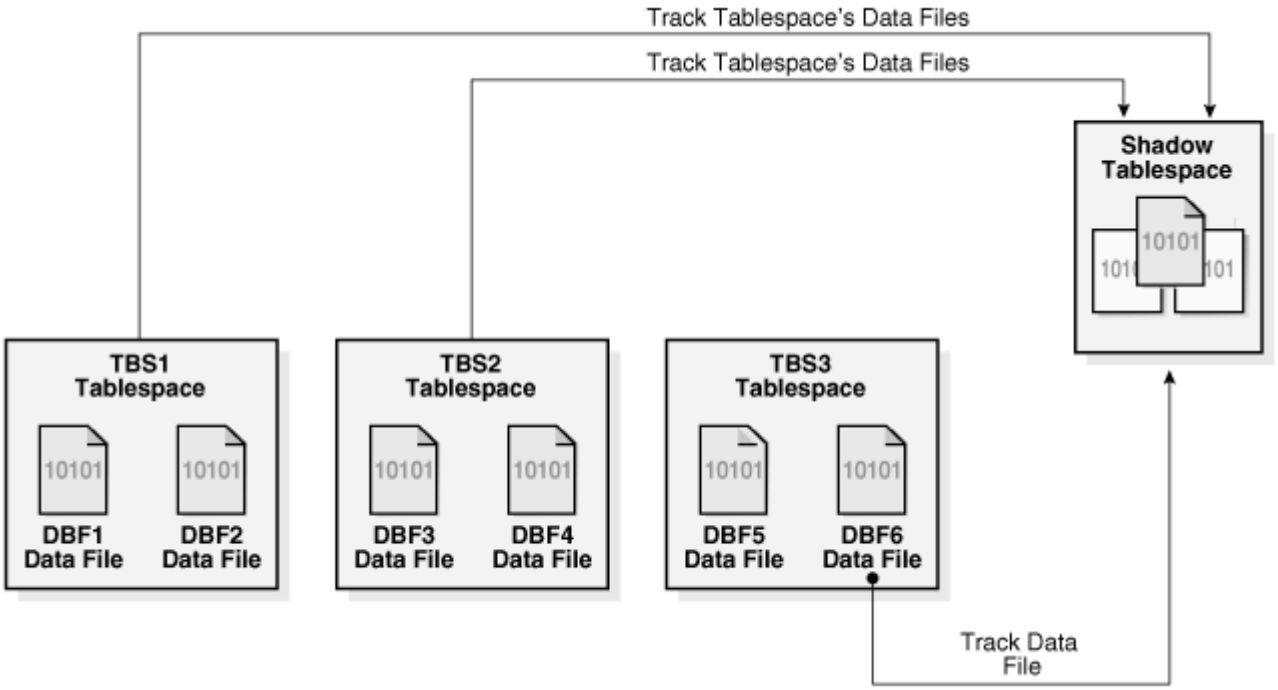
シャドウ消失書込み保護は、消失書込みを高速に検出して即座に応答します。シャドウ消失書込み保護を使用すれば、データ消失を最小限に抑え、データベース修復に必要な時間を短縮できます。

シャドウ消失書込み保護を使用するには、データベースに対してこの保護を有効にし、1つ以上のシャドウ表領域を作成する必要があります。シャドウ表領域は特別な目的のためのbigfile表領域で、追跡されるデータファイルのシステム変更番号(SCN)のみが格納されます。シャドウ表領域を作成するには、CREATE TABLESPACE文でLOST WRITE PROTECTION句を指定します。

追跡されるデータ・ブロックをディスクから読み取るときに、シャドウ消失書込み保護は、シャドウ表領域のブロックのSCNを、読取り中のブロックの最新の書込みのSCNと比較して、消失書込みを検出します。シャドウ・エントリのSCNが読取り中のデータ・ブロックのSCNより大きい場合は、消失書込みが発生しました。消失書込みが検出されると、エラーが返されます。

書込み欠落を検知しないと、不適切なデータが他のDMLトランザクションに使用されるためにデータ破損が生じる可能性があります。シャドウ消失書込み保護は、消費される前に消失書込みを検出してデータ破損を回避します。特定の表領域およびデータ・ファイルについてシャドウ書込み欠落保護を有効化できます。このため、最も重要なデータに対してのみ有効にできます。すべてのデータを追跡するために使用する必要はありません。さらに、シャドウ表領域は柔軟性があります。あるシャドウ表領域を別のシャドウ表領域に置き換えて、その構成や場所を変更できます。

図13-1 シャドウ消失書込み保護



シャドウ消失書込み保護が有効な場合、通常のDML操作、およびSQL*Loaderの従来型パス・ロードおよびダイレクト・パス・ロード操作に対して有効です。さらに、Recovery Manager (RMAN)バックアップに対しても有効です。RMANバックアップは、読取り中のブロックの消失書込みをチェックし、そのようなブロックが見つかった場合にエラーを発行します。

表領域またはデータファイルに対してシャドウ消失書込み保護を有効にした後で、新しい消失書込み情報の収集および新しい消失書込みのチェックを停止する必要がある場合は、保護を一時停止できます。シャドウ消失書込み保護を一時停止すると、トラッキング・データがシャドウ表領域に保持され、シャドウ消失書込み保護を再度有効化できます。データ・ファイルまたは表領域のシャドウ消失書込み保護を削除すると、そのトラッキング・データが削除されて再使用できなくなります。

表領域のシャドウ消失書込み保護を有効にする場合は、`ALTER TABLESPACE`文で`LOST WRITE PROTECTION`句を指定します。また、データ・ファイルのシャドウ消失書込み保護を有効にする場合は、`ALTER DATABASE data_file_name`文で`LOST WRITE PROTECTION`句を指定します。表領域のシャドウ消失書込み保護が有効な場合、表領域の現在および将来のすべてのデータファイルに対してシャドウ消失書込み保護が有効になります。

Oracle Databaseでは、追跡されたデータ・ファイルが特定のシャドウ表領域に自動的に割り当てられます。特定のデータ・ファイルに使用されるシャドウ表領域を指定することはできません。シャドウ表領域の量は、シャドウ消失書込み保護が有効なデータファイルが使用する領域の2%以上の必要があります。

ノート:

- 追跡されるデータファイルのサイズを大きくすると、シャドウ消失書込み保護は、対応するシャドウ表領域のトラッキング・データのサイズ変更を試みます。すべてのデータを追跡するための領域が不十分な場合、シャドウ消失書込み保護はログに警告メッセージを挿入し、使用可能なシャドウ領域を使用して、可能なかぎりデータの追跡を続行します。
- データベースのフラッシュバックを実行すると、シャドウ消失書込み保護のデータが削除されます。フラッシュバック後、シャドウ消失書込み保護は再移入されたデータを追跡し、ブロックが更新されるとシャドウ・トラッキング・データも更新されます。
- シャドウ消失書込み保護は、`DB_LOST_WRITE_PROTECT` 初期化パラメータで構成された消失書込

み保護およびスタンバイ・データベースとは関連しません。

親トピック: シャドウ表領域を使用した消失書込み保護の管理

13.11.2 シャドウ消失書込み保護のためのシャドウ表領域の作成

シャドウ消失書込み保護のためのシャドウ表領域を作成するには、LOST WRITE PROTECTION句を指定してCREATE BIGFILE TABLESPACE文を発行します。

シャドウ消失書込み保護が有効な任意の表領域またはデータファイルは、シャドウ表領域を使用できます。シャドウ表領域の量は、シャドウ消失書込み保護が有効なデータファイルが使用する領域の2%以上の必要があります。シャドウ表領域はbigfile表領域の必要があります。

ノート:

シャドウ表領域を作成するには、18.0.0 以上のデータベース互換性レベルが必要になります。

データベースでシャドウ表領域を作成するには:

1. SQL*Plusで、CREATE TABLESPACEシステム権限を持つユーザーとしてデータベースに接続します。
2. LOST WRITE PROTECTION句を指定してCREATE BIGFILE TABLESPACE文を発行します。

例13-1 シャドウ消失書込み保護のためのシャドウ表領域の作成

この例では、シャドウ消失書込み保護のための表領域として、shadow_lwp1表領域を作成します。

```
CREATE BIGFILE TABLESPACE shadow_lwp1 DATAFILE 'shadow_lwp1.df'
   SIZE 10M LOST WRITE PROTECTION;
```

親トピック: シャドウ表領域を使用した消失書込み保護の管理

13.11.3 データベースのシャドウ消失書込み保護の有効化

マルチテナント・コンテナ・データベース(CDB)または非CDBのシャドウ消失書込み保護を有効にするには、ENABLE LOST WRITE PROTECTION句を指定したALTER DATABASE文を使用します。プラガブル・データベース(PDB)のシャドウ消失書込み保護を有効にするには、ENABLE LOST WRITE PROTECTION句を指定したALTER PLUGGABLE DATABASE文を使用します。

個々の表領域およびデータ・ファイルのシャドウ消失書込み保護を有効化する前に、少なくとも1つのシャドウ表領域を作成し、そのシャドウ表領域を格納しているデータベースのシャドウ消失書込み保護を有効化する必要があります。その後で、ALTER TABLESPACE文を使用して表領域のシャドウ消失書込み保護を有効にすることと、ALTER DATABASE文を使用してデータ・ファイルのシャドウ消失書込み保護を有効にすることが可能になります。

ノート:

- データベースのシャドウ消失書込み保護を有効化する場合は、データベースの互換性レベルが 18.0.0 以上で、少なくとも 1 つのシャドウ表領域が存在している必要があります。
- CDB ルートのシャドウ消失書込み保護を有効化または無効化しても、PDB のシャドウ消失書込み保護

には影響しません。そのため、シャドウ消失書込み保護は、CDB ルートで無効化されていても PDB で有効化できます。

- データベースのシャドウ消失書込み保護を有効にすると、シャドウ表領域が自動的に割り当てられます。

データベースのシャドウ消失書込み保護を有効にするには:

1. SQL*Plusで、必要な権限を持つユーザーに接続します。
   - 非CDBまたはCDBルートで、ALTER DATABASEシステム権限を持つユーザーとして接続します。
   - アプリケーション・ルート、PDBまたはアプリケーションPDBで、ALTER PLUGGABLE DATABASEシステム権限を持つユーザーとして接続します。
2. 次のいずれかを行います:
   - 非CDBまたはCDBルートの場合は、ENABLE LOST WRITE PROTECTION句を指定して、ALTER DATABASE文を発行します。
   - アプリケーション・ルート、PDBまたはアプリケーションPDBの場合は、ENABLE LOST WRITE PROTECTION句を指定して、ALTER PLUGGABLE DATABASE文を発行します。

例13-2 非CDBまたはCDBルートのシャドウ消失書込み保護の有効化

```
ALTER DATABASE ENABLE LOST WRITE PROTECTION;
```

例13-3 PDBのシャドウ消失書込み保護の有効化

```
ALTER PLUGGABLE DATABASE ENABLE LOST WRITE PROTECTION;
```

関連トピック

- シャドウ消失書込み保護のためのシャドウ表領域の作成

親トピック: シャドウ表領域を使用した消失書込み保護の管理

13.11.4 表領域とデータファイルに対するシャドウ消失書込み保護の有効化

表領域およびデータファイルに対してシャドウ消失書込み保護を有効にできます。

表領域に対してシャドウ消失書込み保護を有効化するには、ENABLE LOST WRITE PROTECTION句を指定してALTER TABLESPACE文を発行します。データファイルに対してシャドウ消失書込み保護を有効化するには、ENABLE LOST WRITE PROTECTION句を指定してALTER DATABASE data_file_name文を発行します。表領域に対してシャドウ消失書込み保護を有効にすると、その表領域のすべてのデータファイル、およびその表領域に追加されるデータファイルに対してシャドウ消失書込み保護が有効になります。

ノート:

- 表領域またはデータファイルのシャドウ消失書込み保護を有効化するには、データベースに対するシャドウ消失書込み保護が有効で、少なくとも 1 つのシャドウ表領域が存在する必要があります。
- 表領域またはデータ・ファイルのシャドウ消失書込み保護を有効にすると、シャドウ表領域が自動的に割り当てられます。

表領域またはデータ・ファイルのシャドウ消失書込み保護を有効にするには:

1. SQL*Plusで、必要な権限を持つユーザーとしてデータベースに接続します。
    - 表領域のシャドウ消失書込み保護を有効にする場合は、ALTER TABLESPACE権限を持つユーザーとして接続します。
    - 非CDBまたはCDBルートで使用されるデータ・ファイルのシャドウ消失書込み保護を有効化する場合は、ALTER DATABASE権限を持つユーザーとして接続します。
    - アプリケーション・ルート、PDBまたはアプリケーションPDBで使用されるデータ・ファイルのシャドウ消失書込み保護を有効化する場合は、ALTER PLUGGABLE DATABASE権限を持つユーザーとして接続します。
2. 次のアクションのいずれかを実行します。
    - 表領域に対してシャドウ消失書込み保護を有効化するには、ENABLE LOST WRITE PROTECTION句を指定してALTER TABLESPACE文を発行します。
    - 非CDBまたはCDBルートで使用されているデータ・ファイルのシャドウ消失書込み保護を有効化するには、ENABLE LOST WRITE PROTECTION句を指定して、ALTER DATABASE DATAFILE data_file_name文を発行します(data_file_nameはデータ・ファイルの名前に置き換えます)。
    - アプリケーション・ルート、PDBまたはアプリケーションPDBで使用されているデータ・ファイルのシャドウ消失書込み保護を有効化するには、ENABLE LOST WRITE PROTECTION句を指定して、ALTER PLUGGABLE DATABASE DATAFILE data_file_name文を発行します(data_file_nameはデータ・ファイルの名前に置き換えます)。

例13-4 表領域のシャドウ消失書込み保護の有効化

この例では、tbsu1表領域のシャドウ消失書込み保護を有効にしています。

```
ALTER TABLESPACE tbsu1 ENABLE LOST WRITE PROTECTION;
```

例13-5 非CDBまたはCDBルートで使用されるデータ・ファイルのシャドウ消失書込み保護の有効化

この例では、dfile1.dfデータ・ファイルのシャドウ消失書込み保護を有効にしています。

```
ALTER DATABASE DATAFILE 'dfile1.df' ENABLE LOST WRITE PROTECTION;
```

例13-6 アプリケーション・ルート、PDBまたはアプリケーションPDBで使用されるデータ・ファイルのシャドウ消失書込み保護の有効化

この例では、dfile2.dfデータ・ファイルのシャドウ消失書込み保護を有効にしています。

```
ALTER PLUGGABLE DATABASE DATAFILE 'dfile2.df' ENABLE LOST WRITE PROTECTION;
```

関連トピック

- データベースのシャドウ消失書込み保護の有効化
- シャドウ消失書込み保護のためのシャドウ表領域の作成

親トピック: シャドウ表領域を使用した消失書込み保護の管理

13.11.5 データベースのシャドウ消失書込み保護の無効化

マルチテナント・コンテナ・データベース(CDB)または非CDBのシャドウ消失書込み保護を無効化するには、DISABLE LOST WRITE PROTECTION句を指定してALTER DATABASE文を発行します。プラガブル・データベース(PDB)のシャドウ消失書込み保護を無効化するには、DISABLE LOST WRITE PROTECTION句を指定してALTER PLUGGABLE DATABASE文を発行します。

データベースのシャドウ消失書込み保護を無効にすると、そのデータベース内の表領域またはデータ・ファイルは、シャドウ消失書

込み保護で保護できなくなります。

ノート:

- シャドウ消失書込み保護を無効化しても、既存のシャドウ表領域のデータは削除されませんが、このデータは更新されることもチェックされることもなくなります。シャドウ表領域内のデータの削除が必要になった場合は、INCLUDING CONTENTS 句を指定した DROP TABLESPACE 文を使用すると、シャドウ表領域を削除できます。
- CDB ルートのシャドウ消失書込み保護を有効化または無効化しても、PDB のシャドウ消失書込み保護には影響しません。

データベースのシャドウ消失書込み保護を無効にするには:

1. SQL*Plusで、必要な権限を持つユーザーに接続します。
   - 非CDBまたはCDBルートで、ALTER DATABASEシステム権限を持つユーザーとして接続します。
   - アプリケーション・ルート、PDBまたはアプリケーションPDBで、ALTER PLUGGABLE DATABASEシステム権限を持つユーザーとして接続します。
2. 次のいずれかを行います:
   - 非CDBまたはCDBルートの場合は、DISABLE LOST WRITE PROTECTION句を指定して、ALTER DATABASE文を発行します。
   - アプリケーション・ルート、PDBまたはアプリケーションPDBの場合は、DISABLE LOST WRITE PROTECTION句を指定して、ALTER PLUGGABLE DATABASE文を発行します。

例13-7 非CDBまたはCDBルートのシャドウ消失書込み保護の無効化

```
ALTER DATABASE DISABLE LOST WRITE PROTECTION;
```

例13-8 PDBのシャドウ消失書込み保護の無効化

```
ALTER PLUGGABLE DATABASE DISABLE LOST WRITE PROTECTION;
```

関連トピック

- シャドウ消失書込み保護の削除または一時停止

親トピック: シャドウ表領域を使用した消失書込み保護の管理

13.11.6 シャドウ消失書込み保護の削除または一時停止

表領域またはデータファイルに対するシャドウ消失書込み保護を削除または一時停止できます。

表領域またはデータファイルのシャドウ消失書込み保護が不要になった場合は、次のいずれかのオプションを選択できます。

- シャドウ消失書込み保護を削除できます。このオプションでは、表領域またはデータファイルのトラッキング情報がシャドウ表領域から削除されます。さらに、表領域またはデータ・ファイルの新しい消失書込み情報の収集も停止され、新しい消失書込みのチェックも停止されます。
- シャドウ消失書込み保護を一時停止できます。このオプションでは、表領域またはデータ・ファイルの新しい消失書込み情報の収集と、新しい消失書込みのチェックが停止されます。ただし、古い消失書込み情報はシャドウ表領域に保持されます。表領域またはデータファイルに対してシャドウ消失書込み保護を再度有効にすると、古い消失書込み情報を使用できます。

表領域に対するシャドウ消失書込み保護を削除または一時停止すると、その表領域のすべてのデータファイルに対するシャドウ消失書込み保護が削除または一時停止されます。

表領域またはデータ・ファイルのシャドウ消失書込み保護を削除または一時停止するには:

1. SQL*Plusで、必要な権限を持つユーザーとしてデータベースに接続します。
    - 表領域のシャドウ消失書込み保護を削除または一時停止する場合は、ALTER TABLESPACE権限を持つユーザーとして接続します。
    - 非CDBまたはCDBルートで使用されるデータ・ファイルのシャドウ消失書込み保護を削除または一時停止する場合は、ALTER DATABASE権限を持つユーザーとして接続します。
    - アプリケーション・ルート、PDBまたはアプリケーションPDBで使用されるデータ・ファイルのシャドウ消失書込み保護を削除または一時停止する場合は、ALTER PLUGGABLE DATABASE権限を持つユーザーとして接続します。
2. 次のアクションのいずれかを実行します。
    - 表領域のシャドウ消失書込み保護を削除または一時停止するには、REMOVE LOST WRITE PROTECTION句(削除の場合)またはSUSPEND LOST WRITE PROTECTION句(一時停止の場合)を指定して、ALTER TABLESPACE文を発行します。
    - 非CDBまたはCDBルートで使用されているデータ・ファイルのシャドウ消失書込み保護を削除または一時停止するには、REMOVE LOST WRITE PROTECTION句(削除の場合)またはSUSPEND LOST WRITE PROTECTION句(一時停止の場合)を指定して、ALTER DATABASE DATAFILE data_file_name文を発行します(data_file_nameはデータ・ファイルの名前に置き換えます)。
    - アプリケーション・ルート、PDBまたはアプリケーションPDBで使用されているデータ・ファイルのシャドウ消失書込み保護を削除または一時停止するには、REMOVE LOST WRITE PROTECTION句(削除の場合)またはSUSPEND LOST WRITE PROTECTION句(一時停止の場合)を指定して、ALTER PLUGGABLE DATABASE DATAFILE data_file_name文を発行します(data_file_nameはデータ・ファイルの名前に置き換えます)。

例13-9 表領域のシャドウ消失書込み保護の削除

この例では、tbsu1表領域のシャドウ消失書込み保護を削除しています。

```
ALTER TABLESPACE tbsu1 REMOVE LOST WRITE PROTECTION;
```

例13-10 非CDBで使用されるデータ・ファイルのシャドウ消失書込み保護の一時停止

この例では、非CDBで使用されているdfile1.dfデータ・ファイルのシャドウ消失書込み保護を一時停止しています。

```
ALTER DATABASE DATAFILE 'dfile1.df' SUSPEND LOST WRITE PROTECTION;
```

例13-11 PDBで使用されるデータ・ファイルのシャドウ消失書込み保護の削除

この例では、PDBで使用されているdfile2.dfデータ・ファイルのシャドウ消失書込み保護を削除しています。

```
ALTER PLUGGABLE DATABASE DATAFILE 'dfile2.df' SUSPEND LOST WRITE PROTECTION;
```

親トピック: シャドウ表領域を使用した消失書込み保護の管理

13.11.7 シャドウ表領域の削除

シャドウ表領域は、DROP TABLESPACE文を使用して削除できます。INCLUDING CONTENTS句を指定したDROP TABLESPACE文を使用すると、シャドウ表領域は内容とともに削除されます。INCLUDING CONTENTS句の指定なしでDROP TABLESPACE文を使用すると、シャドウ表領域の削除前に、その内容が別のシャドウ表領域に移動されます(この表

領域が存在していて、十分な空き領域がある場合)。

親トピック: シャドウ表領域を使用した消失書込み保護の管理

13.12 SYSAUX表領域の管理

SYSAUX表領域は、データベースの作成時に、SYSTEM表領域の補助表領域としてインストールされます。これまで個別に表領域を作成して使用していた一部のデータベースのコンポーネントは、SYSAUX表領域に含まれるようになりました。

SYSAUX表領域が使用不可能になった場合でも、データベースのコア機能は実行可能です。SYSAUX表領域を使用するデータベース機能は、エラーが発生したり、機能が制限されることがあります。

- SYSAUX表領域に含まれる占有データの監視
  SYSAUX表領域の占有データを監視できます。
- SYSAUX表領域内外への占有データの移動
  V$SYSAUX_OCCUPANTSビューで、SYSAUX表領域の各占有データの移動プロシージャを使用できます。
- SYSAUX表領域のサイズの制御
  SYSAUX表領域は、いくつかのデータベース・コンポーネントによって占有され、その合計サイズはそれらのコンポーネントが消費する領域によって決定します。同様に、コンポーネントが消費する領域は、使用される機能およびデータベース・ワークロードの性質によって決定します。

親トピック: 表領域の管理

13.12.1 SYSAUX表領域に含まれる占有データの監視

SYSAUX表領域の占有データを監視できます。

SYSAUX表領域の登録済占有データのリストは、「SYSAUX表領域について」を参照してください。これらのコンポーネントはSYSAUX表領域を使用し、SYSAUX表領域を占有する方法がインストール時に提供されます。

SYSAUX表領域の占有データを監視するには、次のようにします。

- V$SYSAUX_OCCUPANTSビューを問い合せます。

このビューには、SYSAUX表領域の占有データに関する次の情報がリスト表示されます。

- 占有データの名前
- 占有データの説明
- スキーマ名
- 移動プロシージャ
- 現行の領域使用

ビュー情報は、占有データ別にメンテナンスされます。

関連項目:

V$SYSAUX_OCCUPANTSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: SYSAUX表領域の管理

13.12.2 SYSAUX表領域内外への占有データの移動

V$SYSAUX_OCCUPANTSビューで、SYSAUX表領域の各占有データの移動プロシージャを使用できます。

コンポーネントのインストール時には、コンポーネントをSYSAUXに常駐させないように指定することもできます。また、後でコンポーネントを指定の表領域に再配置する必要が生じた場合、そのコンポーネントについては、V$SYSAUX_OCCUPANTSビューで指定した移動プロシージャを使用して移動を実行できます。

移動プロシージャを使用すると、コンポーネントを他の表領域からSYSAUX表領域に移動することもできます。

親トピック: SYSAUX表領域の管理

13.12.3 SYSAUX表領域のサイズの制御

SYSAUX表領域は、いくつかのデータベース・コンポーネントによって占有され、その合計サイズはそれらのコンポーネントが消費する領域によって決定します。同様に、コンポーネントが消費する領域は、使用される機能およびデータベース・ワークロードの性質によって決定します。

SYSAUX表領域を最も大きく占有するのは自動ワークロード・リポジトリ(AWR)です。AWRが消費する領域は、特定の時間におけるシステム内でのアクティブなセッションの数、スナップショット間隔、履歴データ保存期間など、いくつかの要因によって決定します。アクティブな同時セッション数の平均が10の標準的なシステムでは、AWRデータ用として約200から300MBの領域が必要になる場合があります。AWRのサイズを制御するには、スナップショット間隔と履歴データ保存期間を変更します。

SYSAUX表領域のもう1つの主要な占有データは、埋込みのOracle Enterprise Manager Cloud Controlリポジトリです。このリポジトリはCloud Controlで使用され、そのメタデータが格納されます。このリポジトリのサイズは、データベース・アクティビティ、およびリポジトリに格納された構成関連情報によって異なります。

他のデータベース・コンポーネントが消費するSYSAUX表領域の領域サイズは、関連する機能(Oracle TextやOracle Streamsなど)を使用中の場合のみ大きくなります。このような機能を使用していない場合、これらのコンポーネントはSYSAUX表領域のサイズに大きく影響しません。

次の表に、システム構成と予測される負荷に基づいてSYSAUX表領域のサイズを設定するためのガイドラインを示します。

パラメータ/推奨設定	小	中	大
CPU 数	2	8	32
同時実行セッションの数	10	20	100
ユーザー・オブジェクト数: 表および索引	500	5,000	50,000
デフォルト構成で安定した状態での予測 SYSAUX のサイズ	500MB	2GB	5GB

親トピック: SYSAUX表領域の管理

13.13 ローカル管理表領域の問題の修正

Oracle Databaseにはローカル管理表領域の問題を修正する手段が用意されています。

- ローカル管理表領域の問題の診断と修復

Oracle Databaseには、DBMS_SPACE_ADMINパッケージが組み込まれています。このパッケージは、ローカル管理表領域の問題の診断と修復に使用するサポートの集まりです。

- 使用例1: 割当て済ブロックが空き(オーバーラップなし)とマークされているときのビットマップの修復
  TABLESPACE_VERIFYプロシージャの使用時に、ビットマップ内で「空き」マークが付いているブロックがセグメントに割り当てられても、セグメント間のオーバーラップがレポートされていないことが検出された場合。
- 使用例2: 破損したセグメントの削除
  ビットマップに「空き」マークが付いたセグメント・ブロックがあるため、セグメントを削除できない場合。このセグメントには、自動的に「破損」マークが付けられます。
- 使用例3: オーバーラップがレポートされたビットマップの修復
  TABLESPACE_VERIFYプロシージャの使用時にオーバラップがレポートされた場合。前の内部エラーに基づいて、一部の実データを削除する必要があります。
- 使用例4: ビットマップ・ブロックのメディア破損の訂正
  ビットマップ・ブロックのセットにメディア破損がある場合。
- 使用例5: ディクショナリ管理表領域からローカル管理表領域への移行
  TABLESPACE_MIGRATE_TO_LOCALプロシージャを使用して、ディクショナリ管理表領域をローカル管理表領域に移行する場合。

親トピック: 表領域の管理

13.13.1 ローカル管理表領域の問題の診断と修復

Oracle Databaseには、DBMS_SPACE_ADMINパッケージが組み込まれています。このパッケージは、ローカル管理表領域の問題の診断と修復に使用するサポートの集まりです。

DBMS_SPACE_ADMINパッケージのプロシージャ

次の表に、DBMS_SPACE_ADMINパッケージに含まれるプロシージャを示します。各プロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

プロシージャ	説明
ASSM_SEGMENT_VERIFY	自動セグメント領域管理が使用可能な表領域内に作成されたセグメントの整合性を検証します。sid_ora_process_id.trc という名前のダンプ・ファイルを、V$DIAG_INFO ビューの Diag Trace エントリに対応する位置に出力します。 手動セグメント領域管理が指定されている表領域には、SEGMENT_VERIFY を使用します。
ASSM_TABLESPACE_VERIFY	自動セグメント領域管理が使用可能な表領域の整合性を検証します。sid_ora_process_id.trc という名前のダンプ・ファイルを、V$DIAG_INFO ビューの Diag Trace エントリに対応する位置に出力します。 手動セグメント領域管理が指定されている表領域には、TABLESPACE_VERIFY を使用します。

プロシージャ	説明
DROP_EMPTY_SEGMENTS	空の表か、または表のパーティションと依存オブジェクトからセグメントを削除します。
MATERIALIZE_DEFERRED_SEGMENTS	セグメント作成の遅延およびその依存オブジェクトがある表および表のパーティションのセグメントをマテリアライズします。
SEGMENT_CORRUPT	セグメントを破損または有効としてマークし、適切なエラーのリカバリを可能にします。
SEGMENT_DROP_CORRUPT	現在破損としてマークされているセグメントを削除します(領域の再生なし)。
SEGMENT_DUMP	特定のセグメントのセグメント・ヘッダーおよびビットマップ・ブロックを、V$DIAG_INFO ビューの Diag Trace エントリに対応する位置にある sid_ora_process_id.trc という名前のダンプ・ファイルにダンプします。オプションで、セグメント・ヘッダーおよびビットマップ・ブロック・サマリーを含む簡略化したダンプ(各ブロックの空き領域率が含まれていない)を選択できます。
SEGMENT_VERIFY	セグメントのエクステント・マップの一貫性を検証します。
TABLESPACE_FIX_BITMAPS	適切な DBA 範囲(エクステント)にビットマップ内で使用可能マークまたは使用済マークを付けます。
TABLESPACE_FIX_SEGMENT_STATES	移行が停止した表領域内のセグメントの状態を修正します。
TABLESPACE_MIGRATE_FROM_LOCAL	ローカル管理表領域を、ディクショナリ管理表領域に移行します。
TABLESPACE_MIGRATE_TO_LOCAL	ディクショナリ管理表領域をローカル管理表領域に移行します。
TABLESPACE_REBUILD_BITMAPS	適切なビットマップを再作成します。
TABLESPACE_REBUILD_QUOTAS	特定の表領域の割当て制限を再作成します。
TABLESPACE_RELOCATE_BITMAPS	ビットマップを指定の保存先に再配置します。
TABLESPACE_VERIFY	表領域内のセグメントについて、ビットマップとエクステント・マップが同期していることを検証します。

次の使用例では、DBMS_SPACE_ADMINパッケージを使用して問題を診断し、解決できる代表的な状況について説明しま

す。

ノート:

前述の一部のプロシージャは、正しく使用しないとデータが消失してリカバリ不能になる場合があります。これらのプロシージャに不明な点がある場合は、Oracle サポート・サービスと共同で作業を行ってください。

関連項目:

- DBMS_SPACE_ADMINパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください
- 「V$DIAG_INFOビューを使用したADRの場所の表示」

親トピック: ローカル管理表領域の問題の修正

13.13.2 使用例1: 割当て済ブロックが空き(オーバーラップなし)とマークされているときのビットマップの修復

TABLESPACE_VERIFYプロシージャの使用時に、ビットマップ内で「空き」マークが付いているブロックがセグメントに割り当てられても、セグメント間のオーバーラップがレポートされていないことが検出された場合。

この使用例では、次のタスクを実行してください。

1. SEGMENT_DUMPプロシージャをコールして、管理者がそのセグメントに割り当てた範囲をダンプします。
2. 範囲ごとに、TABLESPACE_EXTENT_MAKE_USEDオプションを指定してTABLESPACE_FIX_BITMAPSプロシージャをコールし、領域に使用済のマークを付けます。
3. TABLESPACE_REBUILD_QUOTASをコールして割当て制限を再作成します。

親トピック: ローカル管理表領域の問題の修正

13.13.3 使用例2: 破損したセグメントの削除

ビットマップに「空き」マークが付いたセグメント・ブロックがあるため、セグメントを削除できない場合。このセグメントには、自動的に「破損」マークが付けられます。

この使用例では、次のタスクを実行してください。

1. SEGMENT_VERIFY_EXTENTS_GLOBALオプションを指定してSEGMENT_VERIFYプロシージャをコールします。オーバーラップがレポートされない場合は、ステップ2から5までを実行します。
2. SEGMENT_DUMPプロシージャをコールして、そのセグメントに割り当てられたデータ・ブロック・アドレス範囲をダンプします。
3. 範囲ごとに、TABLESPACE_EXTENT_MAKE_FREEオプションを指定してTABLESPACE_FIX_BITMAPSプロシージャをコールし、領域に「空き」のマークを付けます。
4. SEGMENT_DROP_CORRUPTをコールしてSEG$エントリを削除します。
5. TABLESPACE_REBUILD_QUOTASをコールして割当て制限を再作成します。

親トピック: ローカル管理表領域の問題の修正

13.13.4 使用例3: オーバーラップがレポートされたビットマップの修復

TABLESPACE_VERIFYプロシージャで、いくつかオーバーラップがレポートされる場合。前の内部エラーに基づいて、一部の実データを削除する必要があります。

この場合、表t1などの削除するオブジェクトを選択してから、次のタスクを実行します。

1. t1がオーバーラップしているすべてのオブジェクトのリストを作成します。
2. 表t1を削除します。必要に応じて、SEGMENT_DROP_CORRUPTプロシージャをコールしてフォローアップします。
3. t1がオーバーラップしていたすべてのオブジェクトに対して、SEGMENT_VERIFYプロシージャをコールします。必要に応じて、TABLESPACE_FIX_BITMAPSプロシージャをコールして該当するビットマップに使用済を示すマークを付けます。
4. TABLESPACE_VERIFYプロシージャを再度実行し、問題が解決したかどうかを検証します。

親トピック: ローカル管理表領域の問題の修正

13.13.5 使用例4: ビットマップ・ブロックのメディア破損の訂正

ビットマップ・ブロックの集合にメディア破損がある場合。

この使用例では、次のタスクを実行してください。

1. すべてのビットマップ・ブロック、または1つしか破損していない場合はそのブロックに対して、TABLESPACE_REBUILD_BITMAPSプロシージャをコールします。
2. TABLESPACE_REBUILD_QUOTASをコールして割当て制限を再作成します。
3. TABLESPACE_VERIFYプロシージャをコールして、ビットマップの整合性を検証します。

親トピック: ローカル管理表領域の問題の修正

13.13.6 使用例5: ディクショナリ管理表領域からローカル管理表領域への移行

TABLESPACE_MIGRATE_TO_LOCALプロシージャを使用して、ディクショナリ管理表領域をローカル管理表領域に移行する場合。

この操作はオンラインで実行されますが、領域管理操作は移行が完了するまでブロックされます。このため、移行処理中にデータの読取りや変更はできますが、大量のデータをロードする場合は追加のエクステントの割当てが必要になるため、操作がブロックされる場合があります。

データベースのブロック・サイズは2KB、表領域tbs_1の既存のエクステント・サイズは10、50および10,000ブロック(それぞれ使用済、使用済および使用可能)とします。MINIMUM EXTENT値は20KB(10ブロック)です。システムにビットマップの割当て単位を選択させることができます。MINIMUM EXTENTを超えない範囲の最大公分母であることから、10ブロックの値が選択されます。

tbs_1をローカル管理表領域に変換する文は、次のとおりです。

```
EXEC DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL ('tbs_1');
```

割当て単位のサイズを指定する場合は、必ずシステムによって計算される単位サイズの因数にします。

親トピック: ローカル管理表領域の問題の修正

13.14 ローカル管理表領域へのSYSTEM表領域の移行

DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCALプロシージャを使用して、SYSTEM表領域をディクショナリ

管理からローカル管理に移行します。

移行を実行する前に、次の条件を満たす必要があります。

- データベースのデフォルト一時表領域がSYSTEMではないこと。
- ディクショナリ管理表領域にロールバック・セグメントがないこと。
- ローカル管理表領域に1つ以上のオンライン・ロールバック・セグメントがあるか、自動UNDO管理を使用している場合は、UNDO表領域がオンラインになっていること。
- UNDO領域を含む表領域(つまり、ロールバック・セグメントを含む表領域またはUNDO表領域)を除き、すべての表領域が読取り専用モードになっていること。
- SYSAUX表領域がオフラインであること。
- システムが制限モードになっていること。
- データベースのコールド・バックアップがあること。

コールド・バックアップを除き、前述のすべての条件はTABLESPACE_MIGRATE_TO_LOCALプロシージャにより施行されます。

次の文は、この移行を実行します。

```
SQL> EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL('SYSTEM');
```

ノート:

SYSTEM 表領域をローカル管理に移行すると、データベース内のディクショナリ管理表領域を読取り/書込み用にできなくなります。ディクショナリ管理表領域を読取り/書込みモードで使用する必要がある場合は、これらの表領域をローカル管理に移行してから、SYSTEM 表領域を移行することをお薦めします。

親トピック: 表領域の管理

13.15 表領域の情報の表示

Oracle Databaseのデータ・ディクショナリ・ビューを使用して、表領域に関する情報を問い合せることができます。

- 表領域のデータ・ディクショナリ・ビュー
  次のデータ・ディクショナリ・ビューおよび動的パフォーマンス・ビューは、データベースの表領域に関して役立つ情報を提供します。
- 例1: 表領域とデフォルト記憶域パラメータの表示
  DBA_TABLESPACESビューを問い合せて、名前およびデフォルト記憶域パラメータをリスト表示できます。
- 例2: データファイルとデータベースの対応する表領域の表示
  DBA_DATA_FILESビューを問い合せて、名前、サイズおよびデータベースの対応する表領域をリスト表示できます。
- 例3: 各表領域の空き領域(エクステント)の統計の表示
  DBA_FREE_SPACEビューを問い合せて、データベースの各表領域の空きエクステントおよび結合アクティビティに関する統計を表示できます。

親トピック: 表領域の管理

13.15.1 表領域のデータ・ディクショナリ・ビュー

次のデータ・ディクショナリ・ビューおよび動的パフォーマンス・ビューは、データベースの表領域に関して役立つ情報を提供します。

ビュー	説明
V$TABLESPACE	制御ファイルに記述されているすべての表領域の名前と番号。
V$ENCRYPTED_TABLESPACES	暗号化されたすべての表領域の名前と暗号化アルゴリズム。
DBA_TABLESPACES、USER_TABLESPACES	すべての(またはユーザーがアクセス可能な)表領域の説明。
DBA_TABLESPACE_GROUPS	表領域グループとそのグループに属する表領域。
DBA_SEGMENTS、USER_SEGMENTS	すべての(またはユーザーがアクセス可能な)表領域内のセグメントに関する情報。
DBA_EXTENTS、USER_EXTENTS	すべての(またはユーザーがアクセス可能な)表領域内のデータ・エクステントに関する情報。
DBA_FREE_SPACE、USER_FREE_SPACE	すべての(またはユーザーがアクセス可能な)表領域内の使用可能エクステントに関する情報。
DBA_TEMP_FREE_SPACE	各一時表領域の割当て済領域の合計と空き領域。
V$DATAFILE	所有する表領域の表領域番号など、すべてのデータファイルに関する情報。
V$TEMPFILE	所有する表領域の表領域番号など、すべての一時ファイルに関する情報。
DBA_DATA_FILES	表領域に属するファイル(データファイル)。
DBA_TEMP_FILES	一時表領域に属するファイル(一時ファイル)。
V$TEMP_EXTENT_MAP	ローカル管理の一時表領域すべての全エクステントに関する情報。
V$TEMP_EXTENT_POOL	ローカル管理の一時表領域の場合、キャッシュされ、各インスタンスで使用されている一時領域の状態。
V$TEMP_SPACE_HEADER	各一時ファイルの使用済領域/空き領域。

ビュー	説明
DBA_USERS	すべてのユーザーのデフォルト表領域と一時表領域。
DBA_TS_QUOTAS	すべてのユーザーの表領域割当て制限。
V$SORT_SEGMENT	特定インスタンス内のすべてのソート・セグメントに関する情報。このビューは、表領域が TEMPORARY タイプの場合にかぎり更新されます。
V$TEMPSEG_USAGE	一時または永続表領域のユーザーに使用される一時(ソート)セグメントの説明。

親トピック: 表領域の情報の表示

13.15.2 例1: 表領域とデフォルト記憶域パラメータの表示

DBA_TABLESPACESビューを問い合せて、名前およびデフォルト記憶域パラメータをリスト表示できます。

データベースに含まれるすべての表領域の名前とデフォルト記憶域パラメータをすべて表示するには、DBA_TABLESPACESビューに対して次の問合せを使用します。

```
SELECT TABLESPACE_NAME "TABLESPACE",
   INITIAL_EXTENT "INITIAL_EXT",
   NEXT_EXTENT "NEXT_EXT",
   MIN_EXTENTS "MIN_EXT",
   MAX_EXTENTS "MAX_EXT",
   PCT_INCREASE
   FROM DBA_TABLESPACES;
TABLESPACE  INITIAL_EXT  NEXT_EXT  MIN_EXT   MAX_EXT    PCT_INCREASE
----------  -----------  --------  -------   -------    ------------
RBS             1048576   1048576        2        40               0
SYSTEM           106496    106496        1        99               1
TEMP             106496    106496        1        99               0
TESTTBS           57344     16384        2        10               1
USERS             57344     57344        1        99               1
```

親トピック: 表領域の情報の表示

13.15.3 例2: データファイルとデータベースの対応する表領域の表示

DBA_DATA_FILESビューを問い合せて、名前、サイズおよびデータベースの対応する表領域をリスト表示できます。

データファイルの名前、サイズおよびデータベースの対応する表領域を表示するには、DBA_DATA_FILESビューに対して次の問合せを入力します。

```
SELECT  FILE_NAME, BLOCKS, TABLESPACE_NAME
   FROM DBA_DATA_FILES;
FILE_NAME                                    BLOCKS  TABLESPACE_NAME
-------------                              ----------  ------------------
/U02/ORACLE/IDDB3/DBF/RBS01.DBF                  1536  RBS
/U02/ORACLE/IDDB3/DBF/SYSTEM01.DBF               6586  SYSTEM
/U02/ORACLE/IDDB3/DBF/TEMP01.DBF                 6400  TEMP
/U02/ORACLE/IDDB3/DBF/TESTTBS01.DBF              6400  TESTTBS
/U02/ORACLE/IDDB3/DBF/USERS01.DBF                 384  USERS
```

親トピック: 表領域の情報の表示

13.15.4 例3: 各表領域の空き領域(エクステント)の統計の表示

DBA_FREE_SPACEビューを問い合せて、データベースの各表領域の空きエクステントおよび結合アクティビティに関する統計を表示できます。

データベース内の各表領域について、使用可能エクステントと結合アクティビティの統計を生成するには、次の問合せを入力します。

```
SELECT TABLESPACE_NAME "TABLESPACE", FILE_ID,
   COUNT(*)    "PIECES",
   MAX(blocks) "MAXIMUM",
   MIN(blocks) "MINIMUM",
   AVG(blocks) "AVERAGE",
   SUM(blocks) "TOTAL"
   FROM DBA_FREE_SPACE
GROUP BY TABLESPACE_NAME, FILE_ID;
TABLESPACE    FILE_ID  PIECES   MAXIMUM    MINIMUM  AVERAGE    TOTAL
----------    -------  ------   -------    -------  -------   ------
RBS                 2       1       955        955      955      955
SYSTEM              1       1       119        119      119      119
TEMP                4       1      6399       6399     6399     6399
TESTTBS             5       5      6364          3     1278     6390
USERS               3       1       363        363      363      363
```

PIECESは表領域ファイル内の空き領域エクステント数、MAXIMUMおよびMINIMUMはデータベース・ブロック内の領域で連続する最大領域と最小領域、AVERAGEは空き領域があるエクステントの平均ブロック・サイズ、そしてTOTALは各表領域ファイル内の空き領域のブロック数を示します。新しいオブジェクトを作成しようとしているとき、またはセグメントを拡張予定であり、表領域に十分な領域があることを確かめるときに、この問合せを使用します。

親トピック: 表領域の情報の表示

14 データファイルおよび一時ファイルの管理

データファイルおよび一時ファイルの管理では、ファイルの作成、変更および削除などのタスクを行います。

ノート:

一時ファイルは、一時表領域にのみ関連付けられたデータファイルの特殊なクラスです。この章で説明する内容は、相違点が示されている場合を除き、データファイルと一時ファイルの両方に適用されます。一時ファイルの詳細は、「ローカル管理の一時表領域の作成」を参照してください

- データファイルを管理するためのガイドライン
  ガイドラインに従ってデータファイルを管理できます。
- データファイルの作成および表領域への追加
  様々なSQL文を使用して、データファイルを作成し、それらを表領域に関連付けることができます。
- データファイルのサイズ変更
  データファイルのサイズを変更できます。たとえば、データベースにさらに領域が必要になった場合、1つ以上のデータファイルのサイズを大きくできます。
- データファイルの可用性の変更
  データファイルのオフライン・バックアップやオフライン・データファイルの再配置などの特定のタスクを実行するには、データファイルの可用性を変更する必要があります。
- データファイルの名前変更と再配置
  オンラインまたはオフライン・データファイルの名前を変更して、それらの名前や位置を変更できます。
- データファイルの削除
  単一のデータファイルまたは一時ファイルを削除するには、ALTER TABLESPACE文のDROP DATAFILE句およびDROP TEMPFILE句を使用できます。
- データファイル内のデータ・ブロックの検証
  データ・ブロックの検証のためにチェックサムを使用するようにデータベースを構成するには、初期化パラメータDB_BLOCK_CHECKSUMをTYPICAL(デフォルト)に設定します。
- データベース・サーバーを使用したファイルのコピー
  データベース内でのファイルのコピー、またはデータベース間でのファイルの転送には、DBMS_FILE_TRANSFERパッケージを使用できます。
- 物理デバイスへのファイルのマッピング
  データファイルがファイル・システム・ファイルである環境では、表領域と基礎となるデバイスとの関連付けを比較的簡単に確認できます。Oracle Databaseには、ファイルとデバイスとのマッピングを提供するDBA_TABLESPACES、DBA_DATA_FILES、V$DATAFILEなどのビューが用意されています。これらのマッピングをデバイス統計と併用して、I/Oパフォーマンスを評価できます。
- データファイルのデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューのセットが、データベースのデータファイルに関する有用な情報を提供します。

関連項目:

- Oracle Databaseサーバーによって作成および管理されるデータファイルと一時ファイルの作成方法は、「Oracle

Managed Filesの使用」を参照してください

- 『Oracle Database概要』

親トピック: Oracle Databaseの構造と記憶域

14.1 データファイルを管理するためのガイドライン

ガイドラインに従ってデータファイルを管理できます。

- データファイルについて
  データファイルは、データベース内のすべての論理構造のデータを格納する、オペレーティング・システムの物理ファイルです。データファイルは、表領域ごとに明示的に作成する必要があります。
- データファイル数の決定
  データベースのためのデータファイルの数を決定する必要があります。
- データファイルのサイズ設定
  表領域を作成するときは、予測されるデータベース・オブジェクトのサイズを推定し、十分なデータファイルを作成する必要があります。
- 適切なデータファイルの配置
  表領域の位置は、その表領域を構成するデータファイルの物理位置によって決定されます。コンピュータのハードウェア資源を適切に使用してください。
- REDOログ・ファイルから分離したデータファイルの格納
  データファイルは、REDOログ・ファイルを格納する同じディスク・ドライブに格納しないでください。データファイルとREDOログ・ファイルが同じディスク・ドライブに格納されていて、このディスク・ドライブで障害が発生すると、これらのファイルをデータベースのリカバリ手順で使用できなくなります。

親トピック: データファイルおよび一時ファイルの管理

14.1.1 データファイルについて

データファイルは、データベース内に存在するすべての論理構造のデータを格納する、オペレーティング・システムの物理ファイルです。データファイルは、表領域ごとに明示的に作成する必要があります。

Oracle Databaseでは、データファイルを一意に識別するために使用される絶対ファイル番号と相対ファイル番号という2つの関連するファイル番号を各データファイルに割り当てます。この番号について、次の表で説明します。

ファイル番号のタイプ	説明
絶対	データベース内のデータファイルを一意に識別します。このファイル番号は、ファイル名を使用するかわりにデータファイルを参照する多くの SQL 文で使用できます。絶対ファイル番号は、V$DATAFILE ビューまたは V$TEMPFILE ビューの FILE#列、または DBA_DATA_FILES ビューまたは DBA_TEMP_FILES ビューの FILE_ID 列で確認できます。
相対	表領域内のデータファイルを一意に識別します。小規模および中規模サイズのデータベースでは、多くの場合、相対ファイル番号と絶対ファイル番号は同じです。ただし、データベース内のデータファイル数が一定のしきい値(通常は 1023)を超えている場合は、相対ファイル番号と絶対ファイル番号が異なります。bigfile 表領域では、相対ファイル番号は常に 1024(OS/390 プ

ファイル番号のタイプ	説明
	ラットフォームでは 4096)です。

親トピック: データファイルを管理するためのガイドライン

14.1.2 データファイル数の決定

データベースのためのデータファイルの数を決定する必要があります。

- データファイル数の決定について
  少なくとも1つのデータファイルが、データベースのSYSTEMおよびSYSAUX表領域のために必要です。データベースにはこれ以外に、複数の表領域とそれに関連するデータファイルまたは一時ファイルが含まれている必要があります。データベースで作成するデータファイルの数に応じて、初期化パラメータの設定値およびCREATE DATABASE文の句の指定が決定します。
- DB_FILES初期化パラメータの値の決定
  Oracle Databaseインスタンスを起動すると、DB_FILES初期化パラメータは、データファイル情報のために予約されるSGA領域の量、つまり、インスタンスのために作成可能な最大データファイル数を示します。
- データファイルを表領域に追加するときの制限事項の考慮
  表領域にデータファイルを追加するときに考慮する必要があるいくつかの制限事項があります。
- データファイル数のパフォーマンスへの影響の考慮
  表領域内、つまりデータベース内に含まれるデータファイルの数は、パフォーマンスに影響を与える可能性があります。

親トピック: データファイルを管理するためのガイドライン

14.1.2.1 データファイル数の決定について

データベースのSYSTEM表領域およびSYSAUX表領域には、少なくとも1つのデータファイルが必要です。データベースにはこれ以外に、複数の表領域とそれに関連するデータファイルまたは一時ファイルが含まれている必要があります。データベースで作成するデータファイルの数に応じて、初期化パラメータの設定値およびCREATE DATABASE文の句の指定が決定します。

オペレーティング・システムによっては、Oracle Databaseに格納できるデータファイルの数が制限される場合があります。また、データファイル数およびその割当ての方法と場所によって、データベースのパフォーマンスが影響を受ける可能性があることを考慮してください。

ノート:

データベース内のデータファイルの数を制御してその管理を簡素化する方法の 1 つが、bigfile 表領域の使用です。bigfile 表領域は 1 つの大型データファイルのみで構成され、大規模データベースを使用する場合、および論理ボリューム・マネージャを使用してオペレーティング・システム・ファイルを管理する場合に特に役立ちます。bigfile 表領域については、「bigfile 表領域」を参照してください。

データベースのデータファイルの数を決める際は、次のガイドラインを考慮してください。

親トピック: データファイル数の決定

14.1.2.2 DB_FILES初期化パラメータの値の決定

Oracle Databaseインスタンスが起動されると、DB_FILES初期化パラメータによって、データファイル情報のために確保する

SGA領域の大きさと、それに伴ってそのインスタンスに作成可能な最大データファイル数が指定されます。

この制限は、そのインスタンスが存続期間にわたって適用されます。DB_FILESの値は(初期化パラメータの設定を変更することによって)変更可能ですが、新しい値はインスタンスが停止されて再起動されるまで有効になりません。

DB_FILESの値を決めるときは、次の点を考慮してください。

- DB_FILESの値が小さすぎる場合は、最初にデータベースを停止しないと、DB_FILES制限を超えてデータファイルを追加できません。
- DB_FILESの値が大きすぎると、メモリーが不必要に消費されます。

親トピック: データファイル数の決定

14.1.2.3 データファイルを表領域に追加するときの制限事項の考慮

表領域にデータファイルを追加するときに考慮すべきいくつかの制限事項があります。

データファイルを従来のsmallfile表領域に追加するときには、次の制限事項があります。

- ほとんどのオペレーティング・システムでは、1つのプロセスで同時にオープンできるファイルの数に制限があります。オープン・ファイル数がオペレーティング・システム制限に達すると、それ以上データファイルを作成できなくなります。
- オペレーティング・システムでは、データファイルの数とサイズに制限があります。
- データベースでは、インスタンスによってオープンされるOracle Databaseのデータファイルの最大数が制限されます。この制限はオペレーティング・システムによって異なります。
- DB_FILES初期化パラメータで指定したデータファイルの数を超えることはできません。
- CREATE DATABASE文またはCREATE CONTROLFILE文を発行するときに、MAXDATAFILESパラメータによって制御ファイルのデータファイル部分の初期サイズを指定します。ただし、新しく追加するファイルの番号がMAXDATAFILESより大きくDB_FILES以下であれば、データファイルのセクションにより多数のファイルを格納できるように、制御ファイルが自動的に拡張されます。

親トピック: データファイル数の決定

14.1.2.4 データファイル数のパフォーマンスへの影響の考慮

表領域、そして最終的にはデータベースに格納されるデータファイルの数は、パフォーマンスに影響を与える可能性があります。

Oracle Databaseでは、オペレーティング・システムで定義されている制限よりも多くの数のデータファイルをデータベース内に作成できます。データベースのDBWnプロセスは、すべてのオンライン・データファイルをオープンできます。Oracle Databaseには、オープン・ファイル記述子をキャッシュとして処理し、オープン・ファイル記述子の数がオペレーティング・システムで定義されている制限に達したときに自動的にファイルをクローズする機能があります。この機能は、パフォーマンスに悪影響を与えるおそれがあります。できれば、オープン・ファイル記述子に関するオペレーティング・システムの制限を調整して、それがデータベース内のオンライン・データファイルの数より大きくなるようにしてください。

関連項目:

- オペレーティング・システムの制限の詳細は、オペレーティング・システム固有のOracleマニュアルを参照してください。
- CREATE DATABASE文またはCREATE CONTROLFILE文のMAXDATAFILESパラメータの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: データファイル数の決定

14.1.3 データファイルのサイズ設定

表領域を作成するときは、データベース・オブジェクトの将来的なサイズを見積り、十分な数のデータファイルを作成する必要があります。

必要に応じて、後で表領域に割り当てられたディスク領域のすべての容量(その結果としてデータベースの容量)を大きくするために、データファイルを作成して表領域に追加できます。可能であれば、データがすべてのデバイスに均等に配分されるように、データファイルを複数のデバイスに作成してください。

親トピック: データファイルを管理するためのガイドライン

14.1.4 適切なデータファイルの配置

表領域の位置は、その表領域を構成するデータファイルの物理的な位置によって決まります。コンピュータのハードウェア資源を適切に使用してください。

たとえば、データベースの格納に複数のディスク・ドライブを使用できる場合は、競合する可能性があるデータファイルを個別のディスクに配置することを検討してください。これにより、ユーザーが情報を問い合せたときに、両方のディスク・ドライブが同時に動作し、同時にデータを取得できます。

関連項目:

I/Oおよびデータファイルの配置の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』

親トピック: データファイルを管理するためのガイドライン

14.1.5 REDOログ・ファイルから分離したデータファイルの格納

データファイルは、データベースのREDOログ・ファイルが格納されているディスク・ドライブに格納しないでください。データファイルとREDOログ・ファイルが同じディスク・ドライブに格納されていて、このディスク・ドライブで障害が発生すると、これらのファイルをデータベースのリカバリ手順で使用できなくなります。

REDOログ・ファイルを多重化すると、全REDOログ・ファイルが失われる可能性が低くなるので、データファイルを一部のREDOログ・ファイルと同じドライブに格納できます。

親トピック: データファイルを管理するためのガイドライン

14.2 データファイルの作成および表領域への追加

様々なSQL文を使用して、データファイルを作成して表領域と関連付けることができます。

どの文でも、作成するデータファイルのファイル仕様を指定するか、またはOracle Managed Files機能を使用してデータベース・サーバーが作成し管理するファイルを作成できます。表には、データファイルの作成に使用する文の簡単な説明と、このマニュアル内に記載されている文の詳細説明への参照が示されています。

SQL文	説明	関連情報
CREATE TABLESPACE	表領域とそれを構成するデータファイルを	「表領域の作成」

SQL文	説明	関連情報
	作成します。	
CREATE TEMPORARY TABLESPACE	ローカル管理の一時表領域とそれを構成する一時ファイル(一時ファイルは特殊なデータファイルです)を作成します。	「ローカル管理の一時表領域の作成」
ALTER TABLESPACE ... ADD DATAFILE	データファイルを作成して表領域に追加します。	「ローカル管理表領域の変更」
ALTER TABLESPACE ... ADD TEMPFILE	一時ファイルを作成して一時表領域に追加します。	「ローカル管理の一時表領域の変更」
CREATE DATABASE	データベースとそれに関連付けられたデータファイルを作成します。	「CREATE DATABASE文を使用したデータベースの作成の概要」
ALTER DATABASE ... CREATE DATAFILE	古いデータファイルにかわる新しい空のデータファイルを作成します(バックアップがない状態で失われたデータファイルを再作成する場合に役立ちます)。	『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

表領域に新しいデータファイルを追加するときにファイル名を完全に指定しないと、データファイルは、オペレーティング・システムに応じてデフォルトのデータベース・ディレクトリまたはカレント・ディレクトリに作成されます。データファイルには、常に完全修飾名を使用することをお薦めします。既存のファイルを再利用する場合以外は、新しいファイル名が他のファイルと競合しないことを確認してください。すでに削除済の旧ファイルは上書きされます。

データファイルを作成する文が失敗した場合は、作成されたオペレーティング・システム・ファイルがすべて削除されます。ただし、ファイル・システムやストレージ・サブシステムで発生する多数の潜在的なエラーが原因で、オペレーティング・システムのコマンドを使用した手動でのファイル削除が必要になる場合があります。

親トピック: データファイルおよび一時ファイルの管理

14.3 データファイルのサイズ変更

データファイルのサイズを変更できます。たとえば、データベースにさらに領域が必要になった場合、1つ以上のデータファイルのサイズを大きくできます。

- データファイルの自動拡張の有効化と無効化
  データベースにより多くの領域が必要になるとデータファイルのサイズが自動的に増加するように、データファイルを作成または既存データファイルを変更できます。ファイル・サイズは、指定された単位ずつ、指定の最大サイズまで増加します。
- 手動によるデータファイルのサイズ変更
  ALTER DATABASE文を使用して、データファイルのサイズを手動で増減できます。

親トピック: データファイルおよび一時ファイルの管理

14.3.1 データファイルの自動拡張機能の使用可能および使用禁止

データファイルを作成するか、既存のデータファイルを変更して、データベースでより多くの領域が必要になった場合に自動的にサイズが増えるようにすることができます。ファイル・サイズは、指定された単位ずつ、指定の最大サイズまで増加します。

データファイルを自動的に拡張するように設定しておくと、次のような利点があります。

- 表領域で領域が足りなくなった場合に、管理者が即時に介入する必要性が減ります。
- エクステントの割当て失敗が原因でアプリケーションが停止または一時停止することがなくなります。

自動ファイル拡張を指定するには、次のSQL文を使用してデータファイルを作成するときにAUTOEXTEND ON句を指定します。

- CREATE DATABASE
- ALTER DATABASE
- CREATE TABLESPACE
- ALTER TABLESPACE

データファイルの自動拡張を有効または無効にするには:

1. データファイルが自動拡張可能であるかどうかを判断するには、DBA_DATA_FILESビューの問合せを実行し、AUTOEXTENSIBLE列を調べます。
2. AUTOEXTEND句を含むALTER DATABASE文を使用して、既存データファイルの自動ファイル拡張を有効/無効にするか、手動でデータファイルのサイズを変更します。bigfile表領域の場合は、AUTOEXTEND句を含むALTER TABLESPACE文を使用します。

次の例は、users表領域に追加するデータファイルの自動拡張機能を使用可能にします。

```
ALTER TABLESPACE users
    ADD DATAFILE '/u02/oracle/rbdb1/users03.dbf' SIZE 10M
      AUTOEXTEND ON
      NEXT 512K
      MAXSIZE 250M;
```

NEXTの値は、データファイルの拡張時にこのファイルに追加される増分値の最小サイズです。MAXSIZEの値は、自動拡張可能なファイルの最大サイズです。

次の例は、データファイルの自動拡張機能を使用禁止にします。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/users03.dbf'
    AUTOEXTEND OFF;
```

関連項目:

データファイルを作成または変更するためのSQL文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: データファイルのサイズ変更

14.3.2 手動によるデータファイルのサイズ変更

手動でデータファイルのサイズを増減させるには、ALTER DATABASE文を使用します。

そのため、さらにデータファイルを追加しなくてもデータベースに領域を追加できます。この機能は、データベースで許容されているデータファイルの最大数に達することが懸念される場合に有効です。

bigfile表領域では、ALTER TABLESPACE文を使用してデータファイルのサイズを変更できます。bigfile表領域にはデータファイルを追加できません。

また、データファイルのサイズを手動で縮小することで、データベース内の未使用領域を再生できます。これは、領域要件の見積りの誤りを訂正する際に有効です。

次の例では、データファイル/u02/oracle/rbdb1/stuff01.dbfが250MBまで拡張していることを前提としています。ただし、その表領域には現在小さなオブジェクトが格納されているので、データファイルのサイズを縮小できます。

次の文は、データファイル/u02/oracle/rbdb1/stuff01.dbfのサイズを縮小します。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf'
   RESIZE 100M;
```

ノート:

必ずしもファイルのサイズを指定した値まで縮小できるわけではありません。ファイルに格納されているデータ量が指定の縮小サイズよりも大きい場合は、エラーが戻されます。

親トピック: データファイルのサイズ変更

14.4 データファイルの可用性の変更

データファイルのオフライン・バックアップやオフライン・データファイルの再配置などの特定のタスクを実行するには、データファイルの可用性を変更する必要があります。

- データファイルの可用性の変更について
  個々のデータファイルまたは一時ファイルの可用性は、それらをオフラインまたはオンラインにすることにより、変更できます。オフラインのデータファイルはデータベースに使用できず、オンライン化されるまでアクセスできません。
- ARCHIVELOGモードでデータファイルをオンライン化またはオフライン化する方法
  個々のデータファイルをオンラインまたはオフラインにするには、ALTER DATABASE文を発行し、DATAFILE句を含めます。
- NOARCHIVELOGモードでデータファイルをオフライン化する方法
  データベースがNOARCHIVELOGモードのときにデータファイルをオフラインにするには、DATAFILE句およびOFFLINE FOR DROP句の両方を含むALTER DATABASE文を使用します。
- 表領域内のすべてのデータファイルおよび一時ファイルの可用性の変更
  ALTER TABLESPACE文の句により、表領域内のすべてのデータファイルまたは一時ファイルのオンラインまたはオフライン状態を変更できます。

親トピック: データファイルおよび一時ファイルの管理

14.4.1 データファイルの可用性の変更について

個々のデータファイルまたは一時ファイルの可用性は、ファイルをオフラインにするか、オンラインにして変更できます。オフラインのデータファイルはデータベースに使用できず、オンライン化されるまでアクセスできません。

データファイルの可用性の変更には、次のような理由があります。

- データファイルのオフライン・バックアップを実行するため。

- オフラインのデータファイルの名前を変更するか、再配置する。最初に、データファイルをオフライン化または表領域をオフライン化できます。
- データベースのデータファイルへの書込みに問題があり、データファイルが自動的にオフライン化された場合。この場合は、後で問題を解決してから、データファイルを手動でオンライン化できます。
- データファイルが破損または欠落している場合。データベースをオープンするには、その前にデータファイルをオフライン化する必要があります。

読取り専用表領域のデータファイルはオフライン化またはオンライン化ができますが、ファイルをオンライン化しても表領域の読取り専用状態に影響を与えることはありません。表領域が読取り/書込み可能な状態に戻るまで、このデータファイルには書き込めません。

ノート:

表領域自体をオフライン化することによって、表領域のすべてのデータファイルを一時的に使用禁止にできます。表領域をオンラインに戻すには、表領域のこれらのファイルをそのままにしておく必要がありますが、「データファイルの名前変更と再配置」に示されているのと同様の手順に従ってファイルを再配置または名前変更できます。

詳細は、「表領域のオフライン化」を参照してください。

データファイルをオンライン化またはオフライン化するには、ALTER DATABASEシステム権限が必要です。ALTER TABLESPACE文を使用してすべてのデータファイルまたは一時ファイルをオフライン化するには、ALTER TABLESPACEまたはMANAGE TABLESPACEシステム権限が必要です。Oracle Real Application Clusters環境では、データベースを排他モードでオープンする必要があります。

親トピック: データファイルの可用性の変更

14.4.2 ARCHIVELOGモードでデータファイルをオンライン化またはオフライン化する方法

個々のデータファイルをオンラインまたはオフラインにするには、ALTER DATABASE文を発行し、DATAFILE句を含めます。

次の文は、指定したデータファイルをオンライン化します。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf' ONLINE;
```

これと同じファイルをオフライン化するには、次の文を発行します。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf' OFFLINE;
```

ノート:

この形式の ALTER DATABASE 文を使用するには、データベースを ARCHIVELOG モードにしてください。NOARCHIVELOG モードのときにデータファイルをオフライン化すると、ファイルが失われるおそれがあるので、この要件により、データファイルが誤って失われないようにします。

親トピック: データファイルの可用性の変更

14.4.3 NOARCHIVELOGモードでデータファイルをオフライン化する方法

データベースがNOARCHIVELOGモードのときにデータファイルをオフライン化するには、DATAFILE句およびOFFLINE FOR DROP句を指定してALTER DATABASE文を使用します。

- OFFLINEキーワードを指定すると、破損しているかどうかに関係なくデータファイルにOFFLINEのマークが付くため、データベースをオープンできます。
- FOR DROPキーワードによって、データファイルが後で削除されるようにマークが付けられます。このようなデータファイルは、オンラインに戻すことはできません。

> ノート:
>
> この操作では、実際にはデータファイルは削除されません。データ・ディクショナリには残っているので、次のいずれかの方法で削除する必要があります。
>
> - ALTER TABLESPACE ... DROP DATAFILE 文
>
>   OFFLINE FOR DROP の後、この方法はディクショナリ管理表領域のみに機能します。
>
> - DROP TABLESPACE ... INCLUDING CONTENTS AND DATAFILES 文
>
> - 前述の方法で失敗した場合は、オペレーティング・システム・コマンドを使用してデータファイルを削除します。この方法は、データ・ディクショナリのデータファイルおよび制御ファイルへの参照が残るため、望ましい方法ではありません。

次の文は、指定したデータファイルをオフライン化し、削除対象としてマークを付けます。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/users03.dbf' OFFLINE FOR DROP;
```

親トピック: データファイルの可用性の変更

14.4.4 表領域内のすべてのデータファイルおよび一時ファイルの可用性の変更

ALTER TABLESPACE文で句を指定することにより、表領域内にあるすべてのデータファイルまたは一時ファイルのオンラインまたはオフラインの状態を変更できます。

具体的には、オンライン/オフラインの状態に影響を与える文として次のものがあります。

- ALTER TABLESPACE ... DATAFILE {ONLINE|OFFLINE}
- ALTER TABLESPACE ... TEMPFILE {ONLINE|OFFLINE}

入力が必要なのは表領域名のみであり、個々のデータファイルや一時ファイルを入力する必要はありません。すべてのデータファイルまたは一時ファイルが影響を受けますが、表領域そのもののオンライン/オフラインの状態は変わりません。

ほとんどの場合、データベースがマウントされていれば、オープンしていなくても、前述のALTER TABLESPACE文を発行できます。ただし、表領域がSYSTEM表領域、UNDO表領域、またはデフォルト一時表領域である場合は、データベースをオープンしないでください。ALTER DATABASE DATAFILE文およびALTER DATABASE TEMPFILE文にもONLINE/OFFLINE句がありますが、これらの文では表領域のファイル名をすべて入力する必要があります。

この操作は表領域の可用性を変更するALTER TABLESPACE...ONLINE|OFFLINE文とは操作が異なるため、構文も異なります。ALTER TABLESPACE文は表領域だけでなくデータファイルもオフラインにしますが、一時表領域または一時ファイルの状態を変更するためには使用できません。

親トピック: データファイルの可用性の変更

14.5 データファイルの名前変更と再配置

オンラインまたはオフライン・データファイルの名前を変更して、それらの名前や位置を変更できます。

- オンライン・データファイルの名前変更と再配置
  オンラインのデータファイルを名前変更または再配置するために、ALTER DATABASE MOVE DATAFILE SQL文を使用できます。この文を使用すると、データベースのオープン時にユーザーがデータファイルにアクセスしている際、データファイルを名前変更したり再配置できます。
- オフライン・データファイルの名前変更と再配置
  オフラインのデータファイルを名前変更および再配置できます。

親トピック: データファイルおよび一時ファイルの管理

14.5.1 オンライン・データファイルの名前変更と再配置

ALTER DATABASE MOVE DATAFILE SQL文を使用して、オンライン・データファイルを名前変更または再配置できます。この文を使用すると、データベースのオープン時にユーザーがデータファイルにアクセスしている際、データファイルを名前変更したり再配置できます。

オンライン・データファイルを名前変更または再配置すると、データベースの制御ファイルに記録されている、データファイルへのポインタが変更されます。また、ファイルは、オペレーティング・システム・レベルでも物理的に名前変更または再配置されます。

次のいずれかのタスクを実行する場合、ユーザーがデータファイルにアクセスできるようにする必要があるため、オンライン・データファイルを名前変更または再配置することができます。

- あるタイプの記憶域から別のタイプの記憶域に、データファイルを移動します。
- 頻繁にアクセスしないデータファイルを、より低コストの記憶域に移動します。
- 表領域を読取り専用にし、そのデータファイルを1回のみ書込み可能な記憶域に移動します。
- データベースをOracle Automatic Storage Management(Oracle ASM)に移動します。

ALTER DATABASE MOVE DATAFILE文を実行するときに、同じ名前のファイルが宛先の場所に存在する場合は、REUSEオプションを指定して既存のファイルを上書きできます。REUSEが指定されておらず、同じ名前のファイルが宛先の場所に存在しない場合、既存のファイルは上書きされず、この文はエラーを返します。

デフォルトでは、ALTER DATABASE MOVE DATAFILE文を実行してデータファイルに新しい場所を指定すると、この文によってデータファイルが移動されます。ただし、KEEPオプションを指定すると、データファイルを古い場所に保持し、新しい場所にコピーできます。この場合、文が正常に完了すると、データベースでは新しい場所のデータファイルのみが使用されます。

ALTER DATABASE MOVE DATAFILE文を使用してデータファイルを名前変更または再配置すると、Oracle Databaseでは、操作の実行時にデータファイルのコピーが作成されます。操作中に元のデータファイルとコピー用に適切なディスク領域があることを確認してください。

DBA_DATA_FILESビューを問い合せて、データファイルごとの名前、場所およびオンライン化の状態を表示できます。

ノート:

- 指定されたデータファイルがオフラインの場合は、ALTER DATABASE MOVE DATAFILE 文によってエラーが出力されます。
- スタンバイ・データベースを使用している場合は、プライマリとスタンバイ(フィジカルまたはロジカル)でオンライン・データファイル移動操作を個別に実行できます。スタンバイはデータファイルがプライマリで移動されても影響を受けず、その逆の場合も同様です。詳細は、『Oracle Data Guard 概要および管理』を参照してください。

- フラッシュバック操作を行っても、移動されたデータファイルは前の場所に再配置されません。オンラインでデータファイルをある場所から別の場所に移動し、後で移動前の時点にデータベースをフラッシュバックする場合、データファイルの場所は新しいままですが、データファイルの内容はフラッシュバックで指定された時点の内容に変更されます。フラッシュバック・データベース操作の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- Windows プラットフォームでデータファイルを再配置する場合、KEEP オプションを省略しても、元のデータファイルは古い場所に保持されます。この場合、文が正常に完了すると、データベースでは新しい場所のデータファイルのみが使用されます。操作が完了した後、必要に応じて古いデータファイルを手動で削除できます。

オンライン・データファイルを名前変更または再配置するには:

1. SQL*Plusで、ALTER DATABASEシステム権限を持つユーザーとしてデータベースに接続します。

   「SQL*Plusを使用したデータベースの起動」を参照してください。

2. ALTER DATABASE MOVE DATAFILE文を実行して、データファイルを指定します。

例14-1 オンライン・データファイルの名前変更

この例では、データファイルを同じ場所に保持したまま、データファイルuser1.dbfの名前をuser01.dbfに変更します。

```
ALTER DATABASE MOVE DATAFILE '/u01/oracle/rbdb1/user1.dbf'
  TO '/u01/oracle/rbdb1/user01.dbf';
```

例14-2 オンライン・データファイルの再配置

この例では、データファイルuser1.dbfを/u01/oracle/rbdb1/ディレクトリから/u02/oracle/rbdb1/ディレクトリに移動します。操作後、このファイルは/u01/oracle/rbdb1/ディレクトリにはありません。

```
ALTER DATABASE MOVE DATAFILE '/u01/oracle/rbdb1/user1.dbf'
  TO '/u02/oracle/rbdb1/user1.dbf';
```

例14-3 オンライン・データファイルのコピー

この例では、データファイルuser1.dbfを/u01/oracle/rbdb1/ディレクトリから/u02/oracle/rbdb1/ディレクトリにコピーします。操作後、古いファイルは/u01/oracle/rbdb1/ディレクトリに保持されます。

```
ALTER DATABASE MOVE DATAFILE '/u01/oracle/rbdb1/user1.dbf'
  TO '/u02/oracle/rbdb1/user1.dbf' KEEP;
```

例14-4 オンライン・データファイルの再配置および既存のファイルの上書き

この例では、データファイルuser1.dbfを/u01/oracle/rbdb1/ディレクトリから/u02/oracle/rbdb1/ディレクトリに移動します。同じ名前のファイルが/u02/oracle/rbdb1/ディレクトリに存在する場合、この文によりファイルが上書きされます。

```
ALTER DATABASE MOVE DATAFILE '/u01/oracle/rbdb1/user1.dbf'
  TO '/u02/oracle/rbdb1/user1.dbf' REUSE;
```

例14-5 オンライン・データファイルのOracle ASMへの再配置

この例では、データファイルuser1.dbfを/u01/oracle/rbdb1/ディレクトリからOracle ASMの場所に移動します。

```
ALTER DATABASE MOVE DATAFILE '/u01/oracle/rbdb1/user1.dbf'
  TO '+dgroup_01/data/orcl/datafile/user1.dbf';
```

例14-6 あるASMの場所から別のASMの場所へのファイルの移動

この例では、データファイルをあるOracle ASMの場所から別のOracle ASMの場所に移動します。

```
ALTER DATABASE MOVE DATAFILE '+dgroup_01/data/orcl/datafile/user1.dbf'
  TO '+dgroup_02/data/orcl/datafile/user1.dbf';
```

データファイルをミラー化してから元のファイルの場所をミラーから削除することにより、Oracle ASMを使用してオンライン・データファイルを移動することもできます。オンライン・データファイルの移動操作は、ALTER DATABASE MOVE DATAFILE文のかわりにOracle ASMを使用した方が高速です。

関連項目:

- ALTER DATABASE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- Oracle Automatic Storage Management管理者ガイド

親トピック: データファイルの名前変更と再配置

14.5.2 オフライン・データファイルの名前変更と再配置

オフライン・データファイルを名前変更および再配置できます。

オフライン・データファイルを名前変更および再配置すると、データベース制御ファイルに記録されている、データファイルへのポインタのみが変更されます。ファイルは物理的に名前が変更されず、オペレーティング・システム・レベルでコピーされません。

- 単一の表領域のオフライン・データファイルを名前変更および再配置する手順
  単一の表領域のために使用できるオフラインのデータファイルを名前変更および再配置できます。これらの手順を実行するには、ALTER TABLESPACEシステム権限が必要です。
- 複数の表領域のオフライン・データファイルの名前変更と再配置
  ALTER DATABASE RENAME FILE文を使用して、1つ以上の表領域内のデータファイルを名前変更および再配置できます。

親トピック: データファイルの名前変更と再配置

14.5.2.1 単一の表領域のオフライン・データファイルを名前変更および再配置する手順

単一の表領域のために使用できるオフラインのデータファイルを名前変更および再配置できます。これらの手順を実行するには、ALTER TABLESPACEシステム権限が必要です。

- 単一の表領域のオフライン・データファイルの名前変更
  単一の表領域内のオフライン・データファイルの名前を変更できます。

- 単一の表領域のオフライン・データファイルの再配置
  単一の表領域内のオフライン・データファイルを再配置できます。

関連項目:

データファイルの名前変更または再配置の準備のための表領域のオフライン化の詳細は、「表領域のオフライン化」を参照してください

親トピック: オフライン・データファイルの名前変更と再配置

14.5.2.1.1 単一の表領域のオフライン・データファイルの名前変更

単一の表領域内のオフライン・データファイルの名前を変更できます。

単一の表領域のオフライン・データファイルの名前を変更するステップは、次のとおりです。

1. データファイルを含む表領域をオフライン化します。データベースはオープンしている必要があります。

   たとえば:

   ```
   ALTER TABLESPACE users OFFLINE NORMAL;
   ```

2. オペレーティング・システムを使用してデータファイルの名前を変更します。
3. ALTER TABLESPACE文にRENAME DATAFILE句を指定して、データベース内のファイル名を変更します。

   たとえば、次の文はデータファイル/u02/oracle/rbdb1/user1.dbfおよび/u02/oracle/rbdb1/user2.dbfをそれぞれ/u02/oracle/rbdb1/users01.dbfおよび/u02/oracle/rbdb1/users02.dbfに名前変更します。

   ```
   ALTER TABLESPACE users
       RENAME DATAFILE '/u02/oracle/rbdb1/user1.dbf',
                       '/u02/oracle/rbdb1/user2.dbf'
                    TO '/u02/oracle/rbdb1/users01.dbf',
                       '/u02/oracle/rbdb1/users02.dbf';
   ```

   古いデータファイルと新しいデータファイルを正しく識別するために、必ず完全なファイル名(パスを含む)を指定してください。特に、古いデータファイル名は、データ・ディクショナリのDBA_DATA_FILESビューに表示されるとおり、正確に指定してください。

4. データベースをバックアップします。データベースの構造を変更した後は、即時にデータベースの完全バックアップを実行してください。
5. ONLINE句を指定したALTER TABLESPACE文を使用して、この表領域をオンラインに戻します。

   ```
   ALTER TABLESPACE users ONLINE
   ```

親トピック: 単一の表領域のオフライン・データファイルを名前変更および再配置する手順

14.5.2.1.2 単一の表領域のオフライン・データファイルの再配置

単一の表領域内のオフライン・データファイルを再配置できます。

ここでは、オフライン・データファイルを再配置する手順の例を示します。

想定する条件は、次のとおりです。

- オープンしているデータベースにusersという表領域が存在し、すべて同じディスク上に配置されたデータファイルによって構成されています。

- users表領域のデータファイルを、別の分離されたディスク・ドライブに再配置します。
- 現在、オープンしているデータベースに管理者権限で接続しています。
- データベースの現行のバックアップは取得済です。

ステップは次のとおりです。

1. 特定のファイルの名前やサイズが不明な場合は、データ・ディクショナリ・ビューDBA_DATA_FILESを問い合せて情報を取得できます。

```
SQL> SELECT FILE_NAME, BYTES FROM DBA_DATA_FILES
  2> WHERE TABLESPACE_NAME = 'USERS';
FILE_NAME                                  BYTES
------------------------------------------ ----------------
/u02/oracle/rbdb1/users01.dbf              102400000
/u02/oracle/rbdb1/users02.dbf              102400000
```

2. データファイルを含む表領域をオフライン化します。

```
ALTER TABLESPACE users OFFLINE NORMAL;
```

3. オペレーティング・システムを使用し、データファイルを新しい位置にコピーして名前変更します。「データベース・サーバーを使用したファイルのコピー」で説明するDBMS_FILE_TRANSFERパッケージを使用して、ファイルをコピーできます。

ノート:

SQL*Plus の HOST コマンドを使用すると、SQL*Plus を一時的に終了し、オペレーティング・システムのコマンドを実行してファイルをコピーできます。

4. データベース内のデータファイルの名前を変更します。

   users表領域を構成するファイルのデータファイル・ポインタは、対応付けられているデータベースの制御ファイルに記録されていますが、これらのポインタをこの時点で旧ファイル名から新ファイル名に変更する必要があります。

   ALTER TABLESPACE...RENAME DATAFILE文を使用します。

```
ALTER TABLESPACE users
    RENAME DATAFILE '/u02/oracle/rbdb1/users01.dbf',
                    '/u02/oracle/rbdb1/users02.dbf'
                 TO '/u03/oracle/rbdb1/users01.dbf',
                    '/u04/oracle/rbdb1/users02.dbf';
```

5. データベースをバックアップします。データベースの構造を変更した後は、即時にデータベースの完全バックアップを実行してください。
6. ONLINE句を指定したALTER TABLESPACE文を使用して、この表領域をオンラインに戻します。

```
ALTER TABLESPACE users ONLINE
```

親トピック: 単一の表領域のオフライン・データファイルを名前変更および再配置する手順

14.5.2.2 複数の表領域のオフライン・データファイルの名前変更および再配置

1つ以上の表領域のデータファイルは、ALTER DATABASE RENAME FILE文を使用して、名前変更および再配置できます。

1回の操作で複数の表領域のデータファイルを名前変更または再配置する方法は、これ以外にありません。この手順を実行す

るには、ALTER DATABASEシステム権限が必要です。

> ノート:
>
> SYSTEM 表領域、デフォルト一時表領域、またはアクティブな UNDO 表領域のデータファイルを名前変更または再配置する場合、これらの表領域はオフライン化できないため、この ALTER DATABASE 文を使用する必要があります。

複数の表領域のデータファイルの名前を変更するステップは、次のとおりです。

1. データベースがマウントされ、クローズされていることを確認します。

   > ノート:
   >
   > データベースは必要な場合クローズする必要はありませんが、データファイル(または一時ファイル)はオフラインにする必要があります。

2. オペレーティング・システムで新しい位置と名前を指定して、名前変更するデータファイルをコピーします。「データベース・サーバーを使用したファイルのコピー」で説明するDBMS_FILE_TRANSFERパッケージを使用して、ファイルをコピーできます。
3. ALTER DATABASEを使用して、データベースの制御ファイル内のファイル・ポインタの名前を変更します。

   たとえば、次の文はデータファイル/u02/oracle/rbdb1/sort01.dbfおよび/u02/oracle/rbdb1/user3.dbfをそれぞれ/u02/oracle/rbdb1/temp01.dbfおよび/u02/oracle/rbdb1/users03.dbfに名前変更します。

   ```
   ALTER DATABASE
       RENAME FILE '/u02/oracle/rbdb1/sort01.dbf',
                   '/u02/oracle/rbdb1/user3.dbf'
                TO '/u02/oracle/rbdb1/temp01.dbf',
                   '/u02/oracle/rbdb1/users03.dbf';
   ```

   古いデータファイルと新しいデータファイルを正しく識別するために、必ず完全なファイル名(パスを含む)を指定してください。特に、古いデータファイル名は、DBA_DATA_FILESビューに表示されるとおり、正確に指定してください。

4. データベースをバックアップします。データベースの構造を変更した後は、即時にデータベースの完全バックアップを実行してください。

親トピック: オフライン・データファイルの名前変更と再配置

14.6 データファイルの削除

ALTER TABLESPACE文のDROP DATAFILE句およびDROP TEMPFILE句を使用して、1つのデータファイルまたは一時ファイルを削除できます。

データファイルは空である必要があります。(データファイルから割り当てられているエクステントが存在しない場合、そのデータファイルは空であるとみなされます。)データファイルまたは一時ファイルを削除すると、データ・ディクショナリおよび制御ファイルからそのデータファイルまたは一時ファイルへの参照が削除され、ファイル・システムまたはOracle Automatic Storage Management(Oracle ASM)ディスク・グループから物理ファイルが削除されます。

次の例では、Oracle ASMディスク・グループDGROUP1の別名example_df3.fで識別されたデータファイルを削除します。データファイルは表領域exampleに属します。

```
ALTER TABLESPACE example DROP DATAFILE '+DGROUP1/example_df3.f';
```

次の例では、表領域lmtempに属する一時ファイルlmtemp02.dbfを削除します。

```
ALTER TABLESPACE lmtemp DROP TEMPFILE '/u02/oracle/data/lmtemp02.dbf';
```

これは、次の文と同じです。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' DROP
     INCLUDING DATAFILES;
```

ノート:

一時ファイルを使用しているセッションがある場合にその一時ファイルを削除しようとすると、エラーが返され、一時ファイルは削除されません。この場合、一時ファイルはオフライン化され、一時ファイルがオフラインの間は一時ファイルを使用しようとする問合せは失敗します。

ALTER TABLESPACE構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

データファイルの削除に関する制限事項

データファイルおよび一時ファイルの削除に関する制限事項は、次のとおりです。

- データベースは、オープンされている必要があります。
- データファイルが空でない場合、そのファイルは削除できません。

  空でなく、スキーマ・オブジェクトを削除しても空にできないデータファイルを削除する場合は、そのデータファイルを含む表領域を削除する必要があります。
- 表領域の最初のデータファイルまたは1つのみのデータファイルは削除できません。

  したがって、bigfile表領域にはDROP DATAFILEは使用できません。
- ディクショナリ管理からローカル管理に移行した読取り専用表領域のデータファイルは削除できません。他のすべての読取り専用表領域からのデータファイルの削除はサポートされています。
- SYSTEM表領域のデータファイルは削除できません。
- ローカル管理表領域のデータファイルがオフラインの場合、そのファイルは削除できません。

関連項目:

表領域の削除

親トピック: データファイルおよび一時ファイルの管理

14.7 データファイル内のデータ・ブロックの検証

チェックサムを使用してデータ・ブロックを検証するようにデータベースを構成するには、初期化パラメータDB_BLOCK_CHECKSUMをTYPICAL(デフォルト)に設定します。

この設定は、DBWnプロセスおよびダイレクト・ローダーが各ブロックのチェックサムを計算し、ブロックをディスクに書き込むときにこのチェックサムをブロック・ヘッダーに格納します。

ブロックが読み込まれるときにチェックサムが検証されるのは、DB_BLOCK_CHECKSUMがTRUEに設定され、前回ブロックが書き込まれたときにチェックサムが格納されている場合のみです。破損が検出されると、メッセージORA-01578が返され、破損に関する情報がアラート・ログに書き込まれます。

DB_BLOCK_CHECKSUMパラメータの値は、ALTER SYSTEM文で動的に変更できます。このパラメータの設定に関係なく、SYSTEM表領域のデータ・ブロックは常にチェックサムを使用して検証されます。

関連項目:

DB_BLOCK_CHECKSUM初期化パラメータの詳細は、『Oracle Databaseリファレンス』

親トピック: データファイルおよび一時ファイルの管理

14.8 データベース・サーバーを使用したファイルのコピー

データベース内でのファイルのコピー、またはデータベース間でのファイルの転送には、DBMS_FILE_TRANSFERパッケージを使用できます。

- データベース・サーバーを使用したファイルのコピーについて
  データベース内でのファイルのコピー、またはデータベース間でのファイルの転送では、トランスポータブル表領域機能を使用する場合のように、必ずしもオペレーティング・システムを使用する必要はありません。この目的には、DBMS_FILE_TRANSFERパッケージを使用できます。
- ローカル・ファイル・システム上のファイルのコピー
  ローカル・ファイル・システム上のファイルをコピーするには、DBMS_FILE_TRANSFERパッケージのCOPY_FILEプロシージャを使用できます。
- サード・パーティのファイル転送
  DBMS_FILE_TRANSFERパッケージのプロシージャは、通常ではローカル・プロシージャ・コールとして起動されますが、リモート・プロシージャ・コールとして呼び出すこともできます。リモート・プロシージャ・コールとして起動すると、別のデータベースに接続している場合でも、データベース内のファイルをコピーできます。
- 拡張ファイル転送メカニズム
  DBMS_FILE_TRANSFERパッケージとDBMS_SCHEDULERパッケージの両方を使用して、より高度なファイル転送メカニズムを作成できます。
- ファイル転送とDBMS_SCHEDULERパッケージ
  単一データベース内およびデータベース間でファイルを自動的に転送するには、DBMS_SCHEDULERパッケージを使用できます。

親トピック: データファイルおよび一時ファイルの管理

14.8.1 データベース・サーバーを使用したファイルのコピーについて

データベース内のファイルをコピーする場合、またはデータベース間でファイルを転送する場合(トランスポータブル表領域機能を使用して転送する場合など)は、必ずしもオペレーティング・システムを使用する必要はありません。この目的には、DBMS_FILE_TRANSFERパッケージを使用できます。

DBMS_FILE_TRANSFERパッケージでは、ローカル・ファイル・システムまたはOracle Automatic Storage

Management(Oracle ASM)ディスク・グループをファイル転送の移動元または移動先として使用できます。Oracle ASMへの転送またはOracle ASMからの転送に含めることができるのは、Oracle Databaseファイル(データファイル、一時ファイル、制御ファイルなど)のみです。

UNIXシステムの場合、DBMS_FILE_TRANSFERパッケージで作成されたファイルの所有者は、インスタンスを実行するシャドウ・プロセスの所有者になります。通常、この所有者はORACLEです。DBMS_FILE_TRANSFERを使用して作成されたファイルは、常に、データベース内のすべてのプロセスによる書込みと読取りが可能ですが、そのようなファイルを直接読取りまたは書込みを行うのに必要な権限を持っていないユーザーは、システム管理者からのアクセスが必要な場合があります。

注意:

データベースによって変更されているファイルのコピーまたは転送に DBMS_FILE_TRANSFER パッケージを使用しないでください。コピーや転送に使用すると、ファイルの整合性がなくなる可能性があります。

関連項目:

- DBMS_FILE_TRANSFERパッケージの使用例は、ファイルのローカル・ファイル・システムへのコピーを参照してください。
- データベース間での表領域のトランスポート方法の詳細は、データベース間での表領域のトランスポートを参照してください。
- DBMS_FILE_TRANSFERパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: データベース・サーバーを使用したファイルのコピー

14.8.2 ファイルのローカル・ファイル・システムへのコピー

ローカル・ファイル・システム上のファイルをコピーするには、DBMS_FILE_TRANSFERパッケージのCOPY_FILEプロシージャを使用できます。

次の例では、DBMS_FILE_TRANSFERパッケージのCOPY_FILEプロシージャを使用してローカル・ファイル・システム上のファイルをコピーしています。この例では、/usr/admin/sourceディレクトリ内のdb1.datという名前のバイナリ・ファイルを、ローカル・ファイル・システムの/usr/admin/destinationディレクトリにdb1_copy.datという名前でコピーします。

1. 権限を付与でき、SQLを使用してディレクトリ・オブジェクトを作成できる管理ユーザーとしてSQL*Plusで接続します。
2. SQLコマンドCREATE DIRECTORYを使用して、コピー元ファイルが格納されるディレクトリのディレクトリ・オブジェクトを作成します。ディレクトリ・オブジェクトは、ディレクトリの別名に類似しています。たとえば、SOURCE_DIRというディレクトリ・オブジェクトを、使用しているコンピュータ・システムの/usr/admin/sourceディレクトリに対して作成するには、次の文を実行します。

   ```
   CREATE DIRECTORY SOURCE_DIR AS '/usr/admin/source';
   ```

3. SQLコマンドCREATE DIRECTORYを使用して、バイナリ・ファイルがコピーされるディレクトリのディレクトリ・オブジェクトを作成します。たとえば、DEST_DIRというディレクトリ・オブジェクトを、使用しているコンピュータ・システムの/usr/admin/destinationディレクトリに対して作成するには、次の文を実行します。

   ```
   CREATE DIRECTORY DEST_DIR AS '/usr/admin/destination';
   ```

4. COPY_FILEプロシージャを実行するユーザーに対して、必要な権限を付与します。この例では、strmadminユーザーがプロシージャを実行します。

```
GRANT EXECUTE ON DBMS_FILE_TRANSFER TO strmadmin;
GRANT READ ON DIRECTORY source_dir TO strmadmin;
GRANT WRITE ON DIRECTORY dest_dir TO strmadmin;
```

5. strmadminユーザーとして接続し、プロンプトが表示されたらユーザー・パスワードを入力します。

```
CONNECT strmadmin
```

6. COPY_FILEプロシージャを実行して、ファイルをコピーします。

```
BEGIN
  DBMS_FILE_TRANSFER.COPY_FILE(
        source_directory_object       =>  'SOURCE_DIR',
        source_file_name              =>  'db1.dat',
        destination_directory_object  =>  'DEST_DIR',
        destination_file_name         =>  'db1_copy.dat');
END;
/
```

このプロシージャを実行する前に、source_directory_objectパラメータによって指定されたディレクトリ内のファイルをsource_file_nameパラメータに指定し、destination_directory_objectパラメータに指定された新しい場所のファイルの新しい名前をdestination_file_nameパラメータに指定する必要があります。source_directory_objectおよびdestination_directory_objectパラメータのディレクトリ・オブジェクトには、相対パスおよびシンボリック・リンクは許可されません。

注意:

データベースによって変更されているファイルのコピーまたは転送に DBMS_FILE_TRANSFER パッケージを使用しないでください。コピーや転送に使用すると、ファイルの整合性がなくなる可能性があります。

親トピック: データベース・サーバーを使用したファイルのコピー

14.8.3 サード・パーティ・ファイル転送

DBMS_FILE_TRANSFERパッケージのプロシージャは、通常、ローカル・プロシージャ・コールとして起動しますが、リモート・プロシージャ・コールとして起動することもできます。リモート・プロシージャ・コールとして起動すると、別のデータベースに接続している場合でも、データベース内のファイルをコピーできます。

たとえば、次のリモート・プロシージャ・コールを実行すると、別のデータベースに接続していても、データベースDBでファイルをコピーできます。

```
DBMS_FILE_TRANSFER.COPY_FILE@DB(...)
```

また、リモート・プロシージャ・コールを使用すると、いずれのデータベースにも接続していなくても、2つのデータベース間でファイルをコピーできます。たとえば、データベースAに接続し、ファイルをデータベースBからデータベースCに転送できます。この場合、データベースAは、転送するファイルの転送元でも転送先でもないため、サード・パーティになります。

サード・パーティ・ファイル転送では、ファイルのプッシュとプルの両方が可能です。前述の例では、AからBまたはCへのデータベース・リンクがあり、そのデータベースから別のデータベースへのデータベース・リンクがある場合は、サード・パーティ・ファイル転送を実行できます。データベースAからBおよびC両方へのデータベース・リンクは必要ありません。

たとえば、AからBへのデータベース・リンクがあり、BからCへの別のデータベース・リンクがある場合は、データベースAで次のプロシージャを実行してファイルをBからCに転送できます。

```
DBMS_FILE_TRANSFER.PUT_FILE@B(...)
```

この構成では、ファイルをプッシュします。

または、AからCへのデータベース・リンクがあり、CからBへの別のデータベース・リンクがある場合は、データベースAで次のプロシージャを実行してファイルをBからCに転送できます。

```
DBMS_FILE_TRANSFER.GET_FILE@C(...)
```

この構成では、ファイルをプルします。

親トピック: データベース・サーバーを使用したファイルのコピー

14.8.4 拡張ファイル転送メカニズム

DBMS_FILE_TRANSFERパッケージとDBMS_SCHEDULERパッケージの両方を使用すると、複雑なファイル転送メカニズムを作成できます。

たとえば、転送するファイルのコピーが複数のデータベースに存在する場合は、ソースの可用性、ソースのロード、宛先データベースへの通信帯域幅などの要因を検討して、最初にアクセスするソース・データベース、および障害発生時にアクセスを試みるソース・データベースを決定します。この場合、それらの要因に関する情報を入手する必要があるため、要因を検討するメカニズムを構築する必要があります。

別の例として、ロードよりも完了時間が短いことが重要である場合は、複数のスケジューラ・ジョブを発行してファイル転送をパラレルで実行できます。また、ソース・データベースと転送先データベースのファイル・レイアウトに関する知識があれば、使用するI/Oデバイスが異なる場合のみ同時転送を実行またはスケジュールすることで、ディスクの競合を最小限にできます。

親トピック: データベース・サーバーを使用したファイルのコピー

14.8.5 ファイル転送とDBMS_SCHEDULERパッケージ

DBMS_SCHEDULERパッケージを使用して、単一のデータベース内およびデータベース間でファイルを自動的に転送できます。

DBMS_SCHEDULERパッケージではサード・パーティ・ファイル転送もサポートされています。スケジューラによって実行されるファイル転送が長時間実行される場合は、ファイルの読取りまたは書込みを行うデータベースでV$SESSION_LONGOPS動的パフォーマンス・ビューを使用してファイル転送を監視できます。スケジューラ・ジョブで使用するデータベース・リンクは、必ず固定ユーザー・データベース・リンクです。

再開可能なスケジューラ・ジョブを使用すると、特に断続的に障害が発生する場合に、ファイル転送の信頼性を自動的に改善できます。転送先ファイルがクローズする前にファイル転送が失敗した場合は、部分的に書き込まれた転送先ファイルがデータベースによって削除された後、ファイル転送を最初から再開できます。したがって、ジョブの残りの部分が再開可能な場合は、再開可能なスケジューラ・ジョブを使用してファイルを転送することを検討してください。スケジューラ・ジョブの詳細は、「Oracle Schedulerによるジョブのスケジュール」を参照してください。

ノート:

再開可能な 1 つのジョブで複数のファイルを転送する場合は、すでに転送済のファイルと転送されていないファイル

がある状態でジョブを再開する方法を検討する必要があります。

親トピック: データベース・サーバーを使用したファイルのコピー

14.9 ファイルと物理デバイスのマッピング

データファイルがファイル・システム・ファイルである環境では、表領域と基礎になるデバイスとの関連付けを調べるのは比較的容易です。Oracle Databaseには、ファイルとデバイスとのマッピングを提供するDBA_TABLESPACES、DBA_DATA_FILES、V$DATAFILEなどのビューが用意されています。これらのマッピングをデバイス統計と併用して、I/Oパフォーマンスを評価できます。

ただし、ホスト・ベースの論理ボリューム・マネージャ(LVM)と、Redundant Array of Inexpensive Disks(RAID)機能を提供する洗練されたストレージ・サブシステムの導入によって、ファイルからデバイスへのマッピングを判別するのが難しくなっています。最新ファイルがブラック・ボックスに隠れていると、そのファイルを判断するのが難しくなるため、問題が発生します。この項では、この問題を解決するためのOracle Databaseのアプローチについて説明します。

ノート:

ここでは、Oracle Database のファイル・マッピング・インタフェースの概要と、DBMS_STORAGE_MAP パッケージおよび動的パフォーマンス・ビューを使用してファイルと物理デバイスのマッピングを公開する方法について説明します。Oracle Enterprise Manager Cloud Control を使用すると、この機能にさらに容易にアクセスできます。それによって提供される使いやすいグラフィカル・インタフェースを使用して、ファイルと物理デバイスをマップできます。詳細は、Cloud Control のオンライン・ヘルプを参照してください。

- Oracle Databaseのファイル・マッピング・インタフェース
  I/Oパフォーマンスを把握するには、ファイルが存在する記憶域の階層の詳細な知識を持っている必要があります。
- Oracle Databaseのファイル・マッピング・インタフェースの動作
  Oracle Databaseファイル・マッピングには、次のコンポーネントが含まれます: FMONバックグラウンド・プロセス、FMPUTLプロセスおよびマッピング・ライブラリ。
- Oracle Databaseのファイル・マッピング・インタフェースの使用方法
  Oracle Databaseのファイル・マッピング・インタフェースを使用してファイル・マッピングを有効にし、ビューのセットでファイル・マッピングに関する情報を取得できます。
- ファイル・マッピングの例
  Oracle Databaseファイル・マッピング機能のいくつかの強力な機能の例を示します。

親トピック: データファイルおよび一時ファイルの管理

14.9.1 Oracle Databaseのファイル・マッピング・インタフェースの概要

I/Oパフォーマンスを把握するには、ファイルが格納されている記憶域の階層の詳細を知る必要があります。

Oracle Databaseには、ファイル、論理ビューの中間レイヤーおよび実際の物理デバイスのマッピング全体を表示するメカニズムが用意されています。この表示には、動的パフォーマンス・ビュー(V$ビュー)のセットが使用されます。これらのビューを使用すると、ファイル・ブロックがあるディスクを正確に特定できます。

これらのビューを作成するために、ストレージ・ベンダーは特定のI/Oスタック要素のマッピングを受け持つマッピング・ライブラリを提供する必要があります。データベースは、バックグラウンド・プロセスFMONによって起動される外部の非Oracle Databaseプロ

セスを介して、これらのライブラリと通信します。FMONは、マッピング情報の管理を受け持ちます。OracleにはPL/SQLパッケージDBMS_STORAGE_MAPが用意されており、このパッケージを使用して、マッピング・ビューを移入するマッピング操作を起動します。

ノート:

Oracle Automatic Storage Management を使用していない場合、ファイル・マッピング・インタフェースは、Windows プラットフォームでは使用できません。Oracle Automatic Storage Management を使用している場合、ファイル・マッピング・インタフェースは、すべてのプラットフォームで使用できます。

関連項目:

Oracle ASMでのファイル・マッピングの使用の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

親トピック: ファイルと物理デバイスのマッピング

14.9.2 Oracle Databaseのファイル・マッピング・インタフェースの動作

Oracle Databaseファイル・マッピングには、次のコンポーネントが含まれます: FMONバックグラウンド・プロセス、FMPUTLプロセスおよびマッピング・ライブラリ。

- ファイル・マッピングの構成要素
  ファイル・マッピング・メカニズムには、いくつかの構成要素が含まれます。
- マッピング構造
  マッピング・ビューの情報を解釈するには、マッピング構造およびこれらの構造のOracle Database表現を理解する必要があります。
- マッピング構造の例
  マッピング構造の例を示します。
- 構成ID
  構成IDは、要素またはファイルに関連付けられたバージョン情報を取得します。

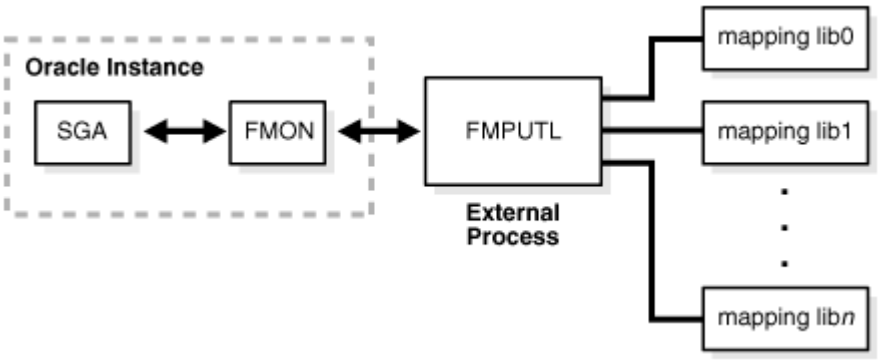
親トピック: ファイルと物理デバイスのマッピング

14.9.2.1 ファイル・マッピングの構成要素

ファイル・マッピング・メカニズムには、いくつかの構成要素が含まれます。

次の図は、ファイル・マッピング・メカニズムの構成要素を示しています。

図14-1 ファイル・マッピングの構成要素



ノート:

Oracle Database 12c 以降では、FILE_MAPPING 初期化パラメータ、FMPUTL プロセスおよびマッピング・ライブラリは非推奨となっています。

- FMON
  FMONは、FILE_MAPPING初期化パラメータがtrueに設定されると、データベースによって開始されるバックグラウンド・プロセスです。FMONはマップ情報を構築し、変更が生じたときにマッピング情報を更新します。
- 外部プロセス(FMPUTL)
  FMONは、ベンダーが提供するマッピング・ライブラリと直接通信するFMPUTLと呼ばれる外部の非Oracle Databaseプロセスを生成します。
- マッピング・ライブラリ
  Oracle Databaseは、特定のマッピング・ライブラリが所有する要素のマッピング情報を検出するためにマッピング・ライブラリを使用します。

親トピック: Oracle Databaseのファイル・マッピング・インタフェースの動作

14.9.2.1.1 FMON

FMONは、FILE_MAPPING初期化パラメータがtrueに設定されている場合に、データベースにより起動されるバックグラウンド・プロセスです。FMONはマップ情報を構築し、変更が生じたときにマッピング情報を更新します。

FMONの役割は、次のとおりです。

- SGAに格納されるマッピング情報を作成します。この情報は、次の構造で構成されます。
  - ファイル
  - ファイル・システムのエクステント
  - 要素
  - サブ要素

  これらの構造については、「マッピング構造」を参照してください。
- 次の原因で変更が発生した場合にマッピング情報をリフレッシュします。
  - データファイル(サイズ)の変更
  - データファイルの追加または削除
  - 記憶域の構成変更(低頻度)
- マッピング情報をデータ・ディクショナリに保存して、起動操作と停止操作の間も持続する情報のビューを保持します。

- インスタンスの起動時にマッピング情報をSGAにリストアします。これにより、インスタンスを起動するたびにマッピング情報全体を再作成するという、高コストの操作が不要になります。

DBMS_STORAGE_MAPパッケージで起動されるプロシージャを使用すると、このマッピングを制御しやすくなります。

親トピック: ファイル・マッピングの構成要素

14.9.2.1.2 外部プロセス(FMPUTL)

FMONは外部の非Oracle DatabaseプロセスFMPUTLを起動し、このプロセスはベンダーが提供するマッピング・ライブラリと直接通信します。

このプロセスは、I/Oスタックのすべてのレベルにマッピング・ライブラリが存在していれば、すべてのレベルを通じてマッピング情報を取得します。一部のプラットフォームでは、I/Oマッピング・スタックの全レベルを通じてマッピングするにはルート権限が必要であるため、外部プロセスのSETUIDビットをONに設定する必要があります。

この外部プロセスの役割は、マッピング・ライブラリを検出してアドレス空間に動的にロードすることです。

親トピック: ファイル・マッピングの構成要素

14.9.2.1.3 マッピング・ライブラリ

Oracle Databaseはマッピング・ライブラリを使用して、特定のマッピング・ライブラリが所有する要素のマッピング情報を検出します。

これらのマッピング・ライブラリを通じて、個々のI/Oスタック要素に関する情報が伝達されます。この情報を使用して、ユーザーが問合せできる動的パフォーマンス・ビューが移入されます。

マッピングを完成するには、すべてのスタック・レベルにマッピング・ライブラリが存在する必要があり、各ライブラリがI/Oマッピング・スタックの独自部分を所有できます。たとえば、VERITAS VxVMライブラリはVERITASボリューム・マネージャに関連するスタック要素を所有し、EMCライブラリはI/Oマッピング・スタックのうちすべてのEMCストレージ固有レイヤーを所有します。

マッピング・ライブラリはベンダーから提供されます。ただし、現在、OracleにはEMCストレージ用のマッピング・ライブラリが用意されています。データベース・サーバーに使用可能なマッピング・ライブラリは、特殊ファイルfilemap.ora内で識別されます。

親トピック: ファイル・マッピングの構成要素

14.9.2.2 マッピング構造

マッピング・ビューの情報を解釈するには、マッピング構造およびこれらの構造のOracle Database表現を理解する必要があります。

マッピング情報を構成する基本構造は、次のとおりです。

- ファイル

  すべてのマッピング構造は、ファイル・サイズ、ファイルを構成するファイル・システムのエクステント数およびファイル・タイプなど、ファイルの属性セットを提供します。

- ファイル・システムのエクステント

  ファイル・システムのエクステントのマッピング構造では、1つの要素にあるブロックの連続するチャンクが記述されます。これには、デバイス・オフセット、エクステント・サイズ、ファイル・オフセット、タイプ(データまたはパリティ)およびエクステントが常駐する要素の名前が含まれます。

ノート:

ファイル・システムのエクステントは、Oracle Database のエクステントとは異なります。ファイル・システムのエクステントは、そのファイル・システムで管理されるデバイスに書き込まれる連続する物理データ・ブロックです。Oracle Database のエクステントは、表領域エクステントなど、データベースで管理される論理構造です。

- 要素

  要素のマッピング構造は、I/Oスタック内の記憶域コンポーネントを記述する抽象マッピング構造です。要素には、ミラー、ストライプ、パーティション、RAID5、連結要素およびディスクがあります。これらの構造は、マッピングのビルディング・ブロックです。

- サブ要素

  副要素のマッピング構造では、I/Oマッピング・スタック内のある要素と次の要素のリンクが記述されます。この構造には、副要素番号、サイズ、副要素が存在する要素の名前および要素のオフセットが含まれます。

次の例は、これらのマッピング構造すべてを示しています。

親トピック: Oracle Databaseのファイル・マッピング・インタフェースの動作

14.9.2.3 マッピング構造の例

マッピング構造の例を示します。

XとYの2つのデータファイルで構成されるOracle Databaseを考えてみます。ファイルXとYはいずれも、ボリュームAにマウントされているファイル・システムに存在します。ファイルXは2つのエクステントによって構成され、ファイルYは1つのエクステントのみによって構成されています。

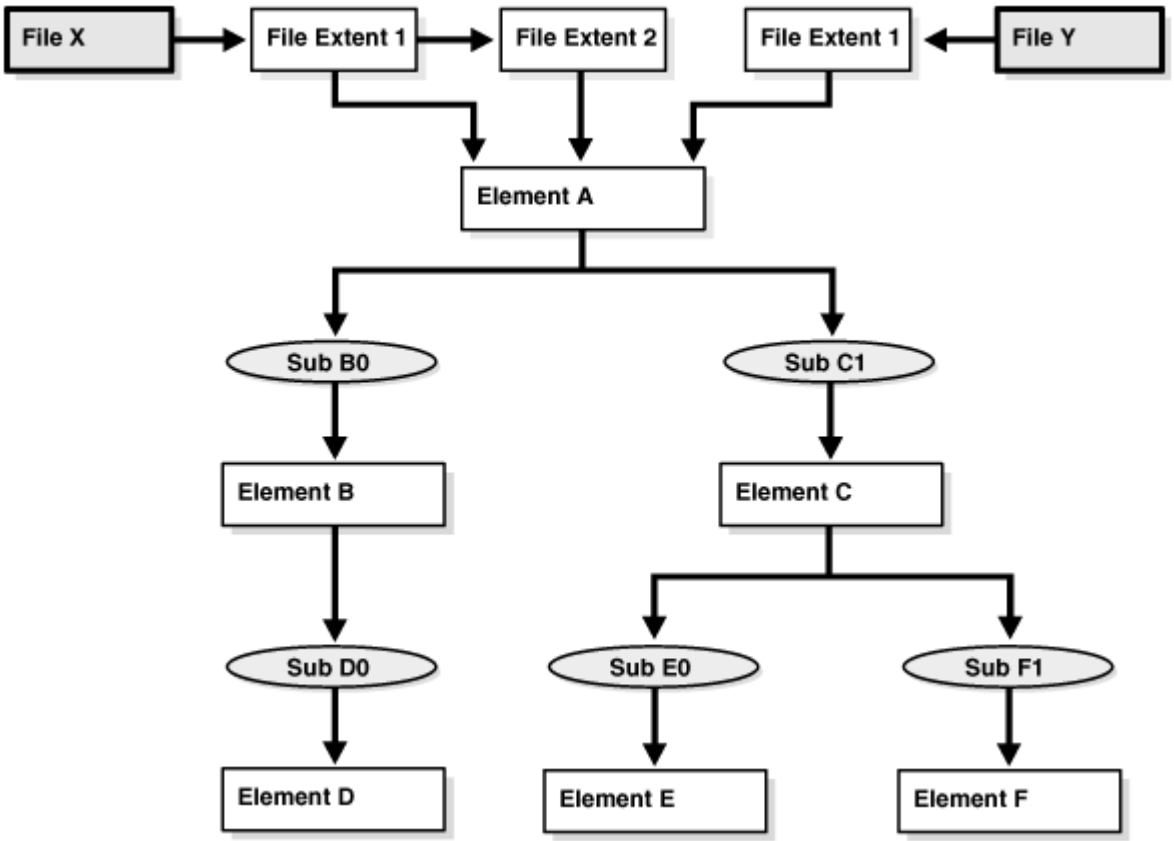
ファイルXの2つのエクステントとファイルYの1つのエクステントは、いずれも要素Aにマップされます。要素Aは、要素BおよびCにストライプ化されています。要素Aは、副要素B0とC1を介してそれぞれ要素BとCにマップされます。

要素Bは、要素D(物理ディスク)のパーティションで、副要素D0を介して要素Dにマップされます。

要素Cは、副要素E0とF1を介してそれぞれ要素EとF(両方とも物理ディスク)にまたがってミラー化されています。

図14-2は、すべてのマッピング構造を示しています。

図14-2 マッピング構造の図



この図が示すマッピング構造は、Oracle Databaseインスタンスのマッピング情報全体を記述するには十分であり、ファイル内の各論理ブロックをI/Oスタック内の各レベルで1つ(またはミラー化の場合は1つ以上)の(要素名、要素オフセットの)タプルにマップしていることに注意してください。

親トピック: Oracle Databaseのファイル・マッピング・インタフェースの動作

14.9.2.4 構成ID

構成IDでは、要素またはファイルに関連付けられたバージョン情報を取得します。

ベンダーのライブラリには構成IDが用意されており、変更があるたびに更新されます。構成IDがなければ、データベースではマッピングに変更があったかどうかを指示できません。

構成IDには、次の2種類があります。

- 永続

  この種の構成IDは、インスタンスが停止されても持続します。

- 非永続

  この種の構成IDは、インスタンスが停止すると持続しません。データベースでは、インスタンスが稼働している間にのみマッピング情報をリフレッシュできます。

親トピック: Oracle Databaseのファイル・マッピング・インタフェースの動作

14.9.3 Oracle Databaseのファイル・マッピング・インタフェースの使用方法

Oracle Databaseのファイル・マッピング・インタフェースを使用してファイル・マッピングを有効にし、ビューのセットにファイル・マッピングに関する情報を取得できます。

- ファイル・マッピングの有効化
  ファイル・マッピングを有効にできます。

- DBMS_STORAGE_MAPパッケージの使用
  DBMS_STORAGE_MAPパッケージにより、マッピング操作を制御できます。
- ファイル・マッピング・ビューからの情報の取得
  DBMS_STORAGE_MAPパッケージによって生成されるマッピング情報は、動的パフォーマンス・ビューで取得されます。

親トピック: ファイルと物理デバイスのマッピング

14.9.3.1 ファイル・マッピングの有効化

ファイル・マッピングを有効にできます。

ファイル・マッピングを有効にするには:

1. 32ビット・プラットフォームの場合は/opt/ORCLfmap/prot1_32/etcディレクトリ、64ビット・プラットフォームの場合は/opt/ORCLfmap/prot1_64/etcディレクトリに、有効なfilemap.oraファイルが存在することを確認します。

   > ノート:
   >
   > filemap.ora ファイルの形式と内容についてはこの項で説明しますが、これはあくまでも参考情報です。filemap.ora ファイルは、システムのインストール時にデータベースによって作成されます。ベンダーから独自ライブラリが提供されるまで、filemap.ora ファイルのエントリは 1 つのみで、オラクル社が提供する EMC ライブラリに関するものです。EMC Symmetrix 配列が使用可能な場合のみ、このファイルを手動で変更する必要があり、このエントリのコメントを解除します。

   filemap.oraファイルは、使用可能なすべてのマッピング・ライブラリが記述されている構成ファイルです。FMONを使用するには、filemap.oraファイルが存在し、マッピング・ライブラリへの有効なパスを指している必要があります。それ以外の場合は、正常に起動しません。

   ライブラリごとに、次の行をfilemap.oraに含める必要があります。

   ```
   lib=vendor_name:mapping_library_path
   ```

   ここで:

   - vendor_nameには、EMC Symmetricライブラリの場合は`Oracle`を指定します。
   - mapping_library_pathには、マッピング・ライブラリのフルパスを指定します。

   このファイル内のライブラリの順序がきわめて重要であることに注意してください。各ライブラリは、構成ファイル内での順序に基づいて問合せされます。

   ファイル・マッピング・サービスは、使用可能なマッピング・ライブラリがなくても起動できます。filemap.oraファイルは、空であっても存在する必要があります。この場合、マッピング・サービスは、新しいマッピング情報を検出できないという制約を伴います。この種の構成で許可されるのは、リストア操作と削除操作のみです。

2. FILE_MAPPING初期化パラメータをTRUEに設定します。

   このパラメータを設定するためにインスタンスを停止する必要はありません。次のALTER SYSTEM文を使用して設定できます。

   ```
   ALTER SYSTEM SET FILE_MAPPING=TRUE;
   ```

3. 適切なDBMS_STORAGE_MAPマッピング・プロシージャを起動します。次の2つのオプションがあります。
   - コールド・スタートの使用例では、Oracle Databaseが起動するのみで、まだマッピング操作は起動されてい

ません。DBMS_STORAGE_MAP.MAP_ALLプロシージャを実行して、データベースに関連するI/Oサブシステム全体のマッピング情報を作成します。

- ウォーム・スタートの使用例では、マッピング情報はすでに作成されており、DBMS_STORAGE_MAP.MAP_SAVEプロシージャを起動してマッピング情報をデータ・ディクショナリに保存するかどうかをオプションで選択できます。(このプロシージャは、デフォルトでDBMS_STORAGE_MAP.MAP_ALL( )内で起動します。)これにより、SGA内のすべてのマッピング情報がディスクに強制的にフラッシュされます。

  データベースの再起動後に、DBMS_STORAGE_MAP.RESTORE( )を使用してマッピング情報をSGAにリストアします。必要な場合は、DBMS_STORAGE_MAP.MAP_ALL( )をコールしてマッピング情報をリフレッシュできます。

親トピック: Oracle Databaseのファイル・マッピング・インタフェースの使用方法

14.9.3.2 DBMS_STORAGE_MAPパッケージの使用

DBMS_STORAGE_MAPパッケージによってマッピング操作を制御できる。

次の表に、使用可能な各種プロシージャを示します。

プロシージャ	用途:
MAP_OBJECT	オブジェクト名、所有者およびタイプで識別されるデータベース・オブジェクトのマッピング情報を作成します。
MAP_ELEMENT	指定した要素のマッピング情報を作成します。
MAP_FILE	指定したファイル名のマッピング情報を作成します。
MAP_ALL	すべてのタイプのデータベース・ファイル(アーカイブ・ログ以外)のマッピング情報全体を作成します。
DROP_ELEMENT	指定した要素のマッピング情報を削除します。
DROP_FILE	指定したファイル名のファイル・マッピング情報を削除します。
DROP_ALL	このインスタンスの SGA からすべてのマッピング情報を削除します。
SAVE	マッピング全体の再生成に必要な情報をデータ・ディクショナリに保存します。
RESTORE	マッピング情報全体をデータ・ディクショナリからインスタンスの共有メモリーにロードします。
LOCK_MAP	このインスタンスの SGA 内でマッピング情報をロックします。

プロシージャ	用途:
UNLOCK_MAP	このインスタンスの SGA 内でマッピング情報のロックを解除します。

関連項目:

- DBMS_STORAGE_MAPパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』
- DBMS_STORAGE_MAPパッケージの使用例については、「ファイル・マッピングの例」

親トピック: Oracle Databaseのファイル・マッピング・インタフェースの使用方法

14.9.3.3 ファイル・マッピング・ビューからの情報の取得

DBMS_STORAGE_MAPパッケージにより生成されたマッピング情報は、動的パフォーマンス・ビューで取得されます。

次の表に、これらのビューの概要を示します。

ビュー	説明
V$MAP_LIBRARY	外部プロセスにより動的にロードされたすべてのマッピング・ライブラリのリストが含まれています。
V$MAP_FILE	インスタンスの共有メモリーにあるすべてのファイル・マッピング構造のリストが含まれています。
V$MAP_FILE_EXTENT	インスタンスの共有メモリーにあるすべてのファイル・システム・エクステントのマッピング構造のリストが含まれています。
V$MAP_ELEMENT	インスタンスの SGA にあるすべての要素マッピング構造のリストが含まれています。
V$MAP_EXT_ELEMENT	すべての要素マッピングの補足情報が含まれています。
V$MAP_SUBELEMENT	インスタンスの共有メモリーにあるすべての副要素マッピング構造のリストが含まれています。
V$MAP_COMP_LIST	すべての要素マッピング構造の補足情報が含まれています。
V$MAP_FILE_IO_STACK	ファイルの記憶域コンテナの階層配列が一連の行として表示されます。各行は 1 つの階層レベルを表します。

ただし、DBMS_STORAGE_MAP.MAP_OBJECTプロシージャにより生成された情報は、グローバルな一時表MAP_OBJECTに取得されます。この表には、オブジェクトの記憶域コンテナの階層配置が表示されます。この表の各行は、階層レベルを表します。MAP_OBJECT表の内容は、次のとおりです。

列	データ型	説明
OBJECT_NAME	VARCHAR2(2000)	オブジェクト名
OBJECT_OWNER	VARCHAR2(2000)	オブジェクトの所有者
OBJECT_TYPE	VARCHAR2(2000)	オブジェクト・タイプ
FILE_MAP_IDX	NUMBER	ファイルの索引(V$MAP_FILE 内の FILE_MAP_IDX に対応)
DEPTH	NUMBER	I/O スタック内の要素の深さ
ELEM_IDX	NUMBER	要素に対応する索引。
CU_SIZE	NUMBER	要素上に連続して存在する、ファイルの論理ブロックの連続するセット(HKB 単位)。
STRIDE	NUMBER	この要素上で連続しているファイル内の連続単位(CU)間の HKB 数。RAID5 ファイルとストライプ・ファイルに使用される。
NUM_CU	NUMBER	この要素上で相互に隣接していて、ファイル内で STRIDE HKB で区切られている連続単位の数。RAID5 の場合、連続単位数にはパリティ・ストライプも含まれる。
ELEM_OFFSET	NUMBER	要素オフセット(HKB 単位)
FILE_OFFSET	NUMBER	ファイルの先頭から連続単位の先頭バイトまでのオフセット(HKB 単位)
DATA_TYPE	VARCHAR2(2000)	データ型(DATA、PARITY または DATA AND PARITY)
PARITY_POS	NUMBER	パリティの位置。RAID5 の場合のみ。このフィールドはパリティとデータ部分を区別するために必要である。
PARITY_PERIOD	NUMBER	パリティ間隔。RAID5 の場合のみ。

親トピック: Oracle Databaseのファイル・マッピング・インタフェースの使用方法

14.9.4 ファイル・マッピングの例

Oracle Databaseファイル・マッピング機能のいくつかの強力な機能の例を示します。

これらの機能には次のものが含まれます。

- 特定のデバイスにまたがるすべてのデータベース・ファイルをマップする機能
- 特定のファイルを対応するデバイスにマップする機能
- I/Oスタックのすべてのレベルでブロックを配分するなど、特定のデータベース・オブジェクトをマップする機能

次の2つのデータファイルで構成されるOracle Databaseインスタンスを考えてみます。

- `t_db1.f`
- `t_db2.f`

この2つのファイルは、VERITAS VxVMホスト・ベースのストライプ化ボリューム`/dev/vx/dsk/ipfdg/ipf-vol1`にマウントされたSolaris UFSファイル・システム上で作成されており、このボリュームはEMC Symmetrix配列から外部化された次のホスト・デバイスで構成されているとします。

- `/dev/vx/rdmp/c2t1d0s2`
- `/dev/vx/rdmp/c2t1d1s2`

次の例では、`MAP_ALL()`操作を実行する必要があることに注意してください。

- 例1: デバイス中にわたるすべてのデータベース・ファイルのマッピング
  この例は、ホスト・デバイスに関連付けられたすべてのOracle Databaseファイルを返します。
- 例2: 対応するデバイスへのファイルのマッピング
  この例は、データ・ファイルのトポロジ・グラフを表示します。
- 例3: データベース・オブジェクトのマッピング
  この例は、表のI/Oスタック内の全レベルでのブロックの分散を表示します。

親トピック: ファイルと物理デバイスのマッピング

14.9.4.1 例1: 1つのデバイスにまたがるすべてのデータベース・ファイルのマッピング

この例は、ホスト・デバイスに関連付けられたすべてのOracle Databaseファイルを返します。

次の問合せでは、`/dev/vx/rdmp/c2t1d1s2`ホスト・デバイスに関連付けられたすべてのOracle Databaseファイルが戻されます。

```
SELECT UNIQUE me.ELEM_NAME, mf.FILE_NAME
   FROM V$MAP_FILE_IO_STACK fs, V$MAP_FILE mf, V$MAP_ELEMENT me
   WHERE mf.FILE_MAP_IDX = fs.FILE_MAP_IDX
   AND me.ELEM_IDX = fs.ELEM_IDX
   AND me.ELEM_NAME = '/dev/vx/rdmp/c2t1d1s2';
```

問合せ結果は次のとおりです。

```
ELEM_NAME                 FILE_NAME
------------------------  --------------------------------
/dev/vx/rdmp/c2t1d1s2     /oracle/dbs/t_db1.f
/dev/vx/rdmp/c2t1d1s2     /oracle/dbs/t_db2.f
```

親トピック: ファイル・マッピングの例

14.9.4.2 例2: ファイルから対応するデバイスへのマッピング

この例は、データ・ファイルのトポロジ・グラフを表示します。

次の問合せでは、`/oracle/dbs/t_db1.f`データファイルのトポロジ・グラフが表示されます。

```
WITH fv AS
```

```
  (SELECT FILE_MAP_IDX, FILE_NAME FROM V$MAP_FILE
   WHERE FILE_NAME = '/oracle/dbs/t_db1.f')
SELECT fv.FILE_NAME, LPAD(' ', 4 * (LEVEL - 1)) || el.ELEM_NAME ELEM_NAME
   FROM V$MAP_SUBELEMENT sb, V$MAP_ELEMENT el, fv,
     (SELECT UNIQUE ELEM_IDX FROM V$MAP_FILE_IO_STACK io, fv
      WHERE io.FILE_MAP_IDX = fv.FILE_MAP_IDX) fs
   WHERE el.ELEM_IDX = sb.CHILD_IDX
   AND fs.ELEM_IDX = el.ELEM_IDX
   START WITH sb.PARENT_IDX IN
     (SELECT DISTINCT ELEM_IDX
      FROM V$MAP_FILE_EXTENT fe, fv
      WHERE fv.FILE_MAP_IDX = fe.FILE_MAP_IDX)
   CONNECT BY PRIOR sb.CHILD_IDX = sb.PARENT_IDX;
```

表示されるトポロジ・グラフは次のとおりです。

```
FILE_NAME                     ELEM_NAME
-----------------------       --------------------------------------------------
/oracle/dbs/t_db1.f           _sym_plex_/dev/vx/rdsk/ipfdg/ipf-vol1_-1_-1
/oracle/dbs/t_db1.f               _sym_subdisk_/dev/vx/rdsk/ipfdg/ipf-vol1_0_0_0
/oracle/dbs/t_db1.f                   /dev/vx/rdmp/c2t1d0s2
/oracle/dbs/t_db1.f                       _sym_symdev_000183600407_00C
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00C_0
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00C_1
/oracle/dbs/t_db1.f               _sym_subdisk_/dev/vx/rdsk/ipfdg/ipf-vol1_0_1_0
/oracle/dbs/t_db1.f                   /dev/vx/rdmp/c2t1d1s2
/oracle/dbs/t_db1.f                       _sym_symdev_000183600407_00D
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00D_0
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00D_1
```

親トピック: ファイル・マッピングの例

14.9.4.3 例3: データベース・オブジェクトのマッピング

この例は、表のI/Oスタック内の全レベルでのブロックの分散を表示します。

この例では、scott.bonus表についてI/Oスタックの全レベルにおけるブロックの分散を表示します。

次のように、最初にMAP_OBJECT()操作を実行する必要があります。

```
EXECUTE DBMS_STORAGE_MAP.MAP_OBJECT('BONUS','SCOTT','TABLE');
```

問合せは次のとおりです。

```
SELECT io.OBJECT_NAME o_name, io.OBJECT_OWNER o_owner, io.OBJECT_TYPE o_type,
       mf.FILE_NAME, me.ELEM_NAME, io.DEPTH,
      (SUM(io.CU_SIZE * (io.NUM_CU - DECODE(io.PARITY_PERIOD, 0, 0,
                         TRUNC(io.NUM_CU / io.PARITY_PERIOD)))) / 2) o_size
   FROM MAP_OBJECT io, V$MAP_ELEMENT me, V$MAP_FILE mf
   WHERE io.OBJECT_NAME =  'BONUS'
   AND   io.OBJECT_OWNER = 'SCOTT'
   AND   io.OBJECT_TYPE =  'TABLE'
   AND   me.ELEM_IDX = io.ELEM_IDX
   AND   mf.FILE_MAP_IDX = io.FILE_MAP_IDX
   GROUP BY io.ELEM_IDX, io.FILE_MAP_IDX, me.ELEM_NAME, mf.FILE_NAME, io.DEPTH,
         io.OBJECT_NAME, io.OBJECT_OWNER, io.OBJECT_TYPE
   ORDER BY io.DEPTH;
```

問合せの結果は次のとおりです。o_size列がKB単位で表されていることに注意してください。

```
O_NAME O_OWNER O_TYPE  FILE_NAME            ELEM_NAME                      DEPTH
O_SIZE
------ ------- ------  -------------------  -----------------------------  ------  --
----
```

```
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  /dev/vx/dsk/ipfdg/ipf-vol1          0
20
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_plex_/dev/vx/rdsk/ipf          1
20
                                           pdg/if-vol1_-1_-1
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_subdisk_/dev/vx/rdsk/          2
12
                                           ipfdg/ipf-vol1_0_1_0
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_subdisk_/dev/vx/rdsk/ipf       2
8
                                           dg/ipf-vol1_0_2_0
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  /dev/vx/rdmp/c2t1d1s2               3
12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  /dev/vx/rdmp/c2t1d2s2               3
8
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_symdev_000183600407_00D        4
12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_symdev_000183600407_00E        4
8
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00D_0      5
12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00D_1      5
12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00E_0      6
8
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00E_1      6
8
```

親トピック: ファイル・マッピングの例

14.10 データファイルのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューのセットは、データベースのデータファイルに関して役立つ情報を提供します。

ビュー	説明
DBA_DATA_FILES	属している表領域やファイル ID など、各データファイルに関する記述情報が表示されます。ファイル ID を使用すると、他のビューと結合して詳細情報を得ることができます。
DBA_EXTENTS USER_EXTENTS	DBA ビューには、データベース内のすべてのセグメントを構成するエクステントが表示されます。エクステントを含むデータファイルのファイル ID が含まれます。USER ビューには、現行ユーザーの所有するオブジェクトに属するセグメントのエクステントが表示されます。
DBA_FREE_SPACE USER_FREE_SPACE	DBA ビューには、すべての表領域の使用可能エクステントが表示されます。エクステントを含むデータファイルのファイル ID が含まれます。USER ビューには、現行ユーザーからアクセス可能な表領域の使用可能エクステントが表示されます。
V$DATAFILE	制御ファイル内のデータファイル情報が含まれます。
V$DATAFILE_HEADER	データファイル・ヘッダーからの情報が含まれます。

ここでは、これらのビューの1つであるV$DATAFILEの使用例を示します。

```
SELECT NAME,
    FILE#,
    STATUS,
    CHECKPOINT_CHANGE# "CHECKPOINT"
  FROM   V$DATAFILE;
NAME                                 FILE#     STATUS       CHECKPOINT
-------------------------------      -----     -------       ----------
/u01/oracle/rbdb1/system01.dbf           1     SYSTEM              3839
/u02/oracle/rbdb1/temp01.dbf             2     ONLINE              3782
/u02/oracle/rbdb1/users03.dbf            3     OFFLINE             3782
```

FILE#には、各データファイルのファイル番号がリストされ、データベースとともに作成されたSYSTEM表領域の最初のデータファイルが常にファイル1になります。STATUSには、データファイルに関するその他の情報がリストされます。データファイルがSYSTEM表領域の一部である場合、そのステータスはSYSTEMです(リカバリが必要な場合を除く)。SYSTEM表領域以外のデータファイルがオンラインである場合、そのステータスはONLINEです。SYSTEM表領域以外のデータファイルがオフラインである場合、そのステータスはOFFLINEまたはRECOVERです。CHECKPOINTには、データファイルの最新チェックポイントで書き込まれた最後のSCN (システム変更番号)がリストされます。

親トピック: データファイルおよび一時ファイルの管理

15 データのトランスポート

データのトランスポートにより、1つのデータベースから他へデータを移動します。

- データのトランスポートについて
  データベース、表領域、表、パーティションおよびサブパーティションのレベルでデータをトランスポートできます。
- データベースのトランスポート
  新しいOracle Databaseインスタンスへデータベースをトランスポートできます。
- データベース間での表領域のトランスポート
  表領域をデータベース間でトランスポートできます。
- データベース間での表、パーティションまたはサブパーティションのトランスポート
  表、パーティションおよびサブパーティションをデータベース間でトランスポートできます。
- プラットフォーム間でのデータの変換
  トランスポータブル操作を実行するときに、ソース・プラットフォームとターゲット・プラットフォームでendiannessが異なる場合は、トランスポートするデータをターゲット・プラットフォームの形式に変換する必要があります。ソース・プラットフォームとターゲット・プラットフォームでendiannessが同じ場合は、データ変換は必要ありません。データを変換するには、DBMS_FILE_TRANSFERパッケージまたはRMAN CONVERTコマンドを使用できます。
- データファイルを転送するためのガイドライン
  データファイルを転送するときにはガイドラインに従う必要があります。

親トピック: Oracle Databaseの構造と記憶域

15.1 データのトランスポートについて

データベース、表領域、表、パーティションおよびサブパーティションのレベルでデータをトランスポートできます。

- データのトランスポートの目的
  データのトランスポートは、同じデータのエクスポート/インポートまたはアンロード/ロードを実行するよりはるかに高速です。高速になるのは、ユーザー定義の表領域では実際のデータをすべて含むデータファイルがターゲットの場所にコピーされ、データ・ポンプを使用してデータベース・オブジェクトのメタデータのみを新しいデータベースに転送するためです。
- データのトランスポート: 使用例
  データのトランスポートは、次のような場合に役立ちます。
- プラットフォーム間でのデータ・トランスポート
  プラットフォーム間でデータをトランスポートできます。
- データのトランスポートに関する一般的な制限事項
  データのトランスポートに関する一般的な制限事項があります。また、フル・トランスポータブル・エクスポート/インポート、トランスポータブル表領域またはトランスポータブル表に固有の制限事項もあります。
- データのトランスポートの互換性に関する注意事項
  データをトランスポートする場合、Oracle Databaseはターゲット・データベースが稼働する最低限の互換性レベルを計算します。

親トピック: データのトランスポート

15.1.1 データのトランスポートの目的

データのトランスポートは、同じデータのエクスポート/インポートまたはアンロード/ロードを実行するよりはるかに高速です。高速に

なるのは、ユーザー定義の表領域では実際のデータをすべて含むデータファイルがターゲットの場所にコピーされ、データ・ポンプを使用してデータベース・オブジェクトのメタデータのみを新しいデータベースに転送するためです。

データは、次のいずれかのレベルでトランスポートできます。

- データベース

  フル・トランスポータブル・エクスポート/インポート機能を使用すると、データベース全体を異なるデータベース・インスタンスに移動できます。

- 表領域

  トランスポータブル表領域機能を使用すると、表領域セットをデータベース間で移動できます。

- 表、パーティションおよびサブパーティション

  トランスポータブル表機能を使用すると、表、パーティションおよびサブパーティションのセットをデータベース間で移動できます。

トランスポータブル表領域およびトランスポータブル表では、ユーザー定義表領域に存在するデータのみがトランスポートされます。ただし、フル・トランスポータブル・エクスポート/インポートでは、ユーザー定義表領域と管理表領域(SYSTEMやSYSAUXなど)の両方に存在するデータがトランスポートされます。フル・トランスポータブル・エクスポート/インポートでは、ユーザー定義表領域内に含まれるオブジェクトのメタデータ、および管理表領域内に含まれるユーザー定義オブジェクトのメタデータとデータの両方がトランスポートされます。特に、フル・トランスポータブル・エクスポート/インポートにおいては、エクスポート・ダンプ・ファイルにはユーザー定義表領域内に含まれるオブジェクトのメタデータのみが含められますが、これには管理表領域内に含まれるユーザー定義オブジェクトのメタデータとデータの両方が含まれます。

親トピック: データのトランスポートについて

15.1.2 データのトランスポート: 使用例

データのトランスポートは、次のような場合に役立ちます。

- フル・トランスポータブル・エクスポート/インポートの使用例
  フル・トランスポータブル・エクスポート/インポート機能は、いくつかの使用例で役立ちます。
- トランスポータブル表領域またはトランスポータブル表の使用例
  トランスポータブル表領域またはトランスポータブル表機能は、いくつかの使用例で役立ちます。

親トピック: データのトランスポートについて

15.1.2.1 フル・トランスポータブル・エクスポート/インポートの使用例

フル・トランスポータブル・エクスポート/インポート機能は、いくつかの使用例で役立ちます。

- 非CDBのCDBへの移動
  マルチテナント・アーキテクチャを使用すると、Oracle Databaseを、ユーザーが作成した1つ以上のプラガブル・データベース(PDB)を含むマルチテナント・コンテナ・データベース(CDB)として機能させることができます。データベースをトランスポートすることによって、非CDBをデータベース(CDB)に移動できます。
- 新しいコンピュータ・システムへのデータベースの移動
  フル・トランスポータブル・エクスポート/インポートを使用すると、データベースを別のコンピュータ・システムに移動できます。ハードウェアをアップグレードしたり、データベースを異なるプラットフォームに移動するために、データベースを新しいコンピュータ・システムに移動できます。
- Oracle Databaseの新しいリリースへのアップグレード

フル・トランスポータブル・エクスポート/インポートを使用すると、データベースをOracle Database 11g リリース2 (11.2.0.3)以上からOracle Database 19cにアップグレードできます。

親トピック: データのトランスポート: 使用例

15.1.2.1.1 非CDBのCDBへの移動

マルチテナント・アーキテクチャを使用すると、Oracle Databaseを、ユーザーが作成した1つ以上のプラガブル・データベース(PDB)を含むマルチテナント・コンテナ・データベース(CDB)として機能させることができます。データベースをトランスポートすることによって、非CDBをデータベース(CDB)に移動できます。

トランスポートされたデータベースは、CDBに関連付けられたプラガブル・データベース(PDB)になります。フル・トランスポータブル・エクスポート/インポートでは、Oracle Database 19c CDBに、Oracle Database 11g リリース2 (11.2.0.3)以上を効率的に移動できます。

関連項目:

- エクスポート・ダンプ・ファイルを使用して非CDBをCDBにトランスポートする手順の説明は、エクスポート・ダンプ・ファイルを使用したデータベースのトランスポートを参照してください
- ネットワーク経由で非CDBをCDBにトランスポートする手順の説明は、ネットワーク経由でのデータベースのトランスポートを参照してください。
- Oracle Multitenant管理者ガイド

親トピック: フル・トランスポータブル・エクスポート/インポートの使用例

15.1.2.1.2 新しいコンピュータ・システムへのデータベースの移動

フル・トランスポータブル・エクスポート/インポートを使用すると、データベースを別のコンピュータ・システムに移動できます。ハードウェアをアップグレードしたり、データベースを異なるプラットフォームに移動するために、データベースを新しいコンピュータ・システムに移動できます。

関連項目:

- 「データベースのトランスポート」
- 「プラットフォーム間でのデータ・トランスポート」

親トピック: フル・トランスポータブル・エクスポート/インポートの使用例

15.1.2.1.3 Oracle Databaseの新しいリリースへのアップグレード

フル・トランスポータブル・エクスポート/インポートを使用すると、データベースをOracle Database 11g リリース2 (11.2.0.3)以上からOracle Database 19cにアップグレードできます。

そのためには、Oracle Database 19cをインストールし、空のデータベースを作成します。次に、フル・トランスポータブル・エクスポート/インポートを使用して、Oracle Database 11gリリース2 (11.2.0.3)データベースをOracle Database 19cデータベースにトランスポートします。

関連項目:

- 「データベースのトランスポート」
- Oracle Databaseインストレーション・ガイド

親トピック: フル・トランスポータブル・エクスポート/インポートの使用例

15.1.2.2 トランスポータブル表領域またはトランスポータブル表の使用例

トランスポータブル表領域またはトランスポータブル表機能は、いくつかの使用例で役立ちます。

- トランスポータブル表領域またはトランスポータブル表に適用される使用例
  一部の使用例では、トランスポータブル表領域またはトランスポータブル表のいずれかが役立ちます。また、トランスポータブル表領域のみが役立つ場合、またはトランスポータブル表のみが役立つ場合もあります。
- データ・ウェアハウスのためのパーティションのトランスポートと連結
  トランスポータブル表およびトランスポータブル表領域を使用して、データ・ウェアハウスのためにパーティションを連結できます。
- 構造化データのCDでの公開
  トランスポータブル表領域とトランスポータブル表の両方で、構造化データをCDで公開できます。
- 複数データベースで同じ表領域を読取り専用でマウントする方法
  トランスポータブル表領域を使用して、表領域を読取り専用で複数のデータベースにマウントできます。
- 履歴データのアーカイブ
  トランスポータブル表領域またはトランスポータブル表を使用する場合、トランスポートされるデータは任意のOracle Databaseにインポートできる自己完結型のファイル・セットです。そのため、トランスポータブル表領域およびトランスポータブル表の手順を使用して、企業データ・ウェアハウスに旧データまたは履歴データをアーカイブできます。
- トランスポータブル表領域を使用したTSPITRの実行
  トランスポータブル表領域を使用して、表領域のPoint-in-Timeリカバリ(TSPITR)を実行できます。
- 個々の表のコピーまたは移動
  トランスポータブル表を使用して、表を含む表領域全体をトランスポートしないで、表または表セットを別のデータベースに移動できます。トランスポータブル表を使用して、個々のパーティションとサブパーティションを別のデータベースにコピーまたは移動することもできます。

親トピック: データのトランスポート: 使用例

15.1.2.2.1 トランスポータブル表領域またはトランスポータブル表に適用される使用例

場合によっては、トランスポータブル表領域かトランスポータブル表のどちらかが役立つことがあります。また、トランスポータブル表領域のみが役立つ場合、またはトランスポータブル表のみが役立つ場合もあります。

表15-1に、各使用例で使用できる機能を示します。

表15-1 トランスポータブル表領域およびトランスポータブル表の使用例

使用例	トランスポータブル表領域	トランスポータブル表
データ・ウェアハウスのためのパーティションのトランスポートと連結	はい	はい
構造化データの CD での公開	はい	はい

使用例	トランスポータブル表領域	トランスポータブル表
履歴データのアーカイブ	はい	はい
トランスポータブル表領域を使用した TSPITR の実行	はい	不可
個々の表のコピーまたは移動	不可	はい

次の項では、これらの使用例について詳しく説明します。

親トピック: トランスポータブル表領域またはトランスポータブル表の使用例

15.1.2.2.2 データ・ウェアハウスのためのパーティションのトランスポートと連結

トランスポータブル表およびトランスポータブル表領域を使用して、データ・ウェアハウスのためにパーティションを連結できます。

標準的な企業のデータ・ウェアハウスには、1つ以上の大きいファクト表が含まれています。これらのファクト表は、企業データ・ウェアハウスを履歴データベースにするために、日付別にパーティション化されていることがあります。この場合、索引を作成すると、スター・クエリーを高速化できます。履歴データベースから最も古いパーティションを削除するたびにグローバル索引を再作成しなくても済むように、この種の履歴パーティション表についてローカル索引を作成することをお薦めします。

たとえば、1か月分のデータをデータ・ウェアハウスに毎月ロードする場合を考えます。データ・ウェアハウスには、salesという大型のファクト表があり、次の列が含まれています。

```
CREATE TABLE sales (invoice_no NUMBER,
   sale_year  INT NOT NULL,
   sale_month INT NOT NULL,
   sale_day   INT NOT NULL)
   PARTITION BY RANGE (sale_year, sale_month, sale_day)
     (partition jan2011 VALUES LESS THAN (2011, 2, 1),
      partition feb2011 VALUES LESS THAN (2011, 3, 1),
      partition mar2011 VALUES LESS THAN (2011, 4, 1),
      partition apr2011 VALUES LESS THAN (2011, 5, 1),
      partition may2011 VALUES LESS THAN (2011, 6, 1),
      partition jun2011 VALUES LESS THAN (2011, 7, 1));
```

次のようにして、ローカルの非同一キー索引を作成します。

```
CREATE INDEX sales_index ON sales(invoice_no) LOCAL;
```

最初は、すべてのパーティションは空で、同じデフォルト表領域にあります。パーティションを毎月1つ作成し、パーティション表salesに連結する必要があります。

現在が2011年7月で、7月分の売上データをパーティション表にロードするとします。ステージング・データベース内で、sales表と同じ列の型を持つ表jul_salesを作成します。必要に応じて、この表を作成する前に新しい表領域ts_julを作成し、この表領域に表を作成することもできます。表jul_salesは、CREATE TABLE ... AS SELECT文を使用して作成できます。jul_salesを作成して移入した後に、sales表内のローカル索引と同じ列に索引を付けて、この表の索引jul_sale_indexを作成することもできます。データ・ウェアハウス環境でステージング表を作成し、移入する方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

表を作成し、索引を構築した後、次のいずれかの方法で表のデータをデータ・ウェアハウスにトランスポートします。

- トランスポータブル表を使用してjul_sales表をデータ・ウェアハウスにトランスポートできます。

- ts_jul表領域を作成した場合は、トランスポータブル表領域を使用して表領域ts_julをデータ・ウェアハウスにトランスポートできます。

データ・ウェアハウスでは、7月分の売上データ用のsales表にパーティションを追加します。これにより、ローカルの非同一キー索引用にも1つのパーティションが作成されます。

```
ALTER TABLE sales ADD PARTITION jul2011 VALUES LESS THAN (2011, 8, 1);
```

トランスポートされた表jul_salesを新しいパーティションに変換して、表salesに連結します。

```
ALTER TABLE sales EXCHANGE PARTITION jul2011 WITH TABLE jul_sales
   INCLUDING INDEXES
   WITHOUT VALIDATION;
```

この文により、新しいデータがパーティション表に連結され、7月分の売上データが新しいパーティションjul2011に格納されます。また、索引jul_sale_indexがsales表のローカル索引のパーティションに変換されます。この文では、構造情報を操作するだけであり、データベースのポインタを切り替えれば済むので、結果は即時に返されます。新しいパーティション内のデータが旧パーティション内のデータとオーバーラップしないことがわかっている場合は、WITHOUT VALIDATION句を指定してください。この句を指定しないと、新しいパーティションの範囲を検証するために、そこに含まれる新しいデータがすべて検査されます。

sales表のすべてのパーティションが同じステージング・データベースから取り込まれる場合(ステージング・データベースが破壊されることはありません)、変換文は常に成功します。ただし、一般に、パーティション表のデータが異なるデータベースから取り込まれる場合は、変換操作が失敗する可能性があります。たとえば、salesのjan2011パーティションが同じステージング・データベースから取り込まれていない場合は、前述の変換操作が失敗して次のエラーが返されることがあります。

```
ORA-19728: data object number conflict between table JUL_SALES and partition JAN2011
in table SALES
```

この競合を解決するには、次の文を発行して、競合しているパーティションを移動します。

```
ALTER TABLE sales MOVE PARTITION jan2011;
```

次に、変換操作を再試行してください。

交換が成功した後は、jul_salesとjul_sale_indexを削除しても安全です(どちらも空になっています)。これで、7月分の売上データはデータ・ウェアハウスに正常にロードされたことになります。

親トピック: トランスポータブル表領域またはトランスポータブル表の使用例

15.1.2.2.3 構造化データのCDでの公開

トランスポータブル表領域とトランスポータブル表の両方で、構造化データをCDで公開できます。

データファイルやエクスポート・ダンプ・ファイルなどの公開するデータをCDにコピーできます。これにより、このCDを配布できます。トランスポータブル表領域を使用している場合は、データをCDにコピーする前にトランスポータブル・セットを生成する必要があります。

顧客は、このCDを受け取ったら、CDからディスク記憶域にデータファイルをコピーすることなく、既存のデータベースにCDの内容を追加できます。たとえば、Microsoft WindowsシステムでD:ドライブがCDドライブであるとします。データファイルcatalog.fおよびエクスポート・ダンプ・ファイルexpdat.dmpのデータを次のようにインポートできます。

```
impdp user_name/password DUMPFILE=expdat.dmp DIRECTORY=dpump_dir
   TRANSPORT_DATAFILES='D:¥catalog.f'
```

CDは、データベースの稼働中に取り出すことができます。この場合、そのデータへの後続の問合せでは、CD上のデータファイルをオープンできないことを示すエラーが返されます。ただし、このデータベースの他の部分への操作は影響を受けません。CDをドライ

ブに戻すと、データは再び読取り可能になります。

CDを取り出すのは、読取り専用表領域のデータファイルを削除するのと同じことです。データベースを停止して再起動すると、削除されたデータファイルが見つからず、データベースをオープンできないことを示すメッセージが表示されます(初期化パラメータREAD_ONLY_OPEN_DELAYEDをTRUEに設定していない場合)。READ_ONLY_OPEN_DELAYEDがTRUEに設定されている場合は、データを問い合せるときにのみ、このファイルが読み込まれます。したがって、CDからデータをトランスポートする場合は、そのCDがデータベースに永続的に連結されないかぎり、READ_ONLY_OPEN_DELAYED初期化パラメータをTRUEに設定します。

親トピック: トランスポータブル表領域またはトランスポータブル表の使用例

15.1.2.2.4 複数データベースで同じ表領域を読取り専用でマウントする方法

トランスポータブル表領域を使用すると、複数のデータベースで1つの表領域を読取り専用でマウントできます。

これにより、データを別々のディスクに複製しなくても、異なるデータベースで同じデータを共有できます。表領域のデータファイルは、どのデータベースからもアクセス可能にする必要があります。データベースの破損を回避するために、表領域はマウント先のすべてのデータベース内で読取り専用のままにしてください。表領域のデータファイルはオペレーティング・システム・レベルで読取り専用にする必要があります。

次に、複数のデータベースで同じ表領域を読取り専用でマウントする方法を2つ示します。

- 表領域の元のデータベースが表領域を共有するデータベースと異なる場合。

  ソース・データベースでトランスポータブル・セットを生成し、すべてのデータベースからアクセス可能なディスクにそのトランスポータブル・セットを格納し、その後、表領域をマウントする各データベースにメタデータをインポートします。

- 表領域が、その表領域を共有するデータベースの1つに属する場合。

  データファイルがすでに共有ディスクに格納されているとします。表領域がすでに格納されているデータベース上で、この表領域を読取り専用にしてトランスポータブル・セットを生成し、データファイルは共有ディスク上の同じ位置に残したまま、表領域を他のデータベースにインポートします。

ディスクを複数のコンピュータからアクセス可能にするには、いくつかの方法があります。クラスタ・ファイル・システムまたはRAWディスクのいずれかを使用できます。ネットワーク・ファイル・システム(NFS)を使用することも可能ですが、NFSの停止中にユーザーが共有表領域を問い合せると、NFS操作がタイムアウトになるまでデータベースが停止することがあります。

後で、一部のデータベースから読取り専用表領域を削除できます。読取り専用表領域を削除しても、表領域のデータファイルは変更されません。したがって、削除操作によって表領域が破損することはありません。表領域をマウントしているデータベースが1つしかない場合を除き、表領域は読取り/書込み可能にしないでください。

親トピック: トランスポータブル表領域またはトランスポータブル表の使用例

15.1.2.2.5 履歴データのアーカイブ

トランスポータブル表領域またはトランスポータブル表を使用する場合、トランスポートされるデータは任意のOracle Databaseにインポートできる自己完結型のファイル・セットです。そのため、トランスポータブル表領域およびトランスポータブル表の手順を使用して、企業データ・ウェアハウスに旧データまたは履歴データをアーカイブできます。

関連項目:

詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

親トピック: トランスポータブル表領域またはトランスポータブル表の使用例

15.1.2.2.6 トランスポータブル表領域を使用したTSPITRの実行

トランスポータブル表領域を使用して、表領域のpoint-in-timeリカバリ(TSPITR)を実行できます。

関連項目:

トランスポータブル表領域を使用してTSPITRを実行する方法は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください

親トピック: トランスポータブル表領域またはトランスポータブル表の使用例

15.1.2.2.7 個々の表のコピーまたは移動

トランスポータブル表を使用して、表を含む表領域全体をトランスポートしないで、表または表セットを別のデータベースに移動できます。トランスポータブル表を使用して、個々のパーティションとサブパーティションを別のデータベースにコピーまたは移動することもできます。

関連項目:

「データベース間での表、パーティションまたはサブパーティションのトランスポート」

親トピック: トランスポータブル表領域またはトランスポータブル表の使用例

15.1.3 プラットフォーム間でのデータ・トランスポート

プラットフォーム間でデータをトランスポートできます。

プラットフォーム間でデータをトランスポートする機能を使用すると、次のことが可能です。

- データベースをプラットフォーム間で移行できます。
- コンテンツ・プロバイダは、簡単にかつ効率的に構造化データを公開し、別のプラットフォームでOracle Databaseを実行している顧客に配布できます。
- データ・ウェアハウス環境からデータ・マート(多くの場合、小規模プラットフォームで実行されている)へのデータの配布を簡素化できます。
- 異なるオペレーティング・システムまたはプラットフォーム上のOracle Databaseインストール間で読取り専用表領域を共有できます。この場合、次の各項で説明するように、それらのプラットフォームおよびendiannessが同じプラットフォームからストレージ・システムにアクセスできることが前提となります。

多くのプラットフォーム(すべてではありません)では、クロス・プラットフォームでのデータのトランスポートがサポートされます。V$TRANSPORTABLE_PLATFORMビューを問い合せると、サポートされているプラットフォームを参照して、各プラットフォームのendian形式(バイトの並び順)を確認できます。次の問合せを実行すると、プラットフォーム間でのデータのトランスポートがサポートされているプラットフォームが表示されます。

```
COLUMN PLATFORM_NAME FORMAT A40
COLUMN ENDIAN_FORMAT A14

SELECT PLATFORM_ID, PLATFORM_NAME, ENDIAN_FORMAT
  FROM V$TRANSPORTABLE_PLATFORM
  ORDER BY PLATFORM_ID;
PLATFORM_ID PLATFORM_NAME                              ENDIAN_FORMAT
----------- ---------------------------------------- --------------
          1 Solaris[tm] OE (32-bit)                  Big
```

```
         2 Solaris[tm] OE (64-bit)                 Big
         3 HP-UX (64-bit)                          Big
         4 HP-UX IA (64-bit)                       Big
         5 HP Tru64 UNIX                           Little
         6 AIX-Based Systems (64-bit)              Big
         7 Microsoft Windows IA (32-bit)           Little
         8 Microsoft Windows IA (64-bit)           Little
         9 IBM zSeries Based Linux                 Big
        10 Linux IA (32-bit)                       Little
        11 Linux IA (64-bit)                       Little
        12 Microsoft Windows x86 64-bit            Little
        13 Linux x86 64-bit                        Little
        15 HP Open VMS                             Little
        16 Apple Mac OS                            Big
        17 Solaris Operating System (x86)          Little
        18 IBM Power Based Linux                   Big
        19 HP IA Open VMS                          Little
        20 Solaris Operating System (x86-64)       Little
        21 Apple Mac OS (x86-64)                   Little
```

ソース・プラットフォームとターゲット・プラットフォームが同じendiannessの場合、データはデータ変換なしでソース・プラットフォームからターゲット・プラットフォームにトランスポートされます。

ソース・プラットフォームとターゲット・プラットフォームでendiannessが異なる場合は、トランスポートするデータをターゲット・プラットフォームの形式に変換する必要があります。次のいずれかの方法使用してデータを変換できます。

- DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャ

  これらのプロシージャのいずれかを使用して、ソース・プラットフォームとターゲット・プラットフォーム間でデータファイルを移動する場合、各データファイル内の各ブロックがターゲット・プラットフォームのendiannessに変換されます。変換はターゲット・プラットフォームで発生します。

- RMAN CONVERTコマンド

  ソース・プラットフォームまたはターゲット・プラットフォームでRMAN CONVERTコマンドを実行します。このコマンドにより、トランスポートするデータがターゲット・プラットフォームの形式に変換されます。

ノート:

UNDO セグメントを含むデータファイルでは、異なる endian 形式間でのデータファイルの変換はサポートされていません。

データファイルのデータを別のプラットフォームにトランスポートする前に、そのデータが属しているプラットフォームをデータファイル・ヘッダーで識別する必要があります異なるプラットフォームのOracle Databaseインストール間で読取り専用表領域をトランスポートするには、少なくとも1回データファイルを読取り/書込みにします。

関連項目:

「プラットフォーム間でのデータの変換」

親トピック: データのトランスポートについて

15.1.4 データのトランスポートに関する一般的な制限事項

データのトランスポートに関する一般的な制限事項があります。また、フル・トランスポータブル・エクスポート/インポート、トランス

ポータブル表領域またはトランスポータブル表に固有の制限事項もあります。

データをトランスポートする場合は、次の一般的な制限事項に注意してください。

- ソース・データベースとターゲット・データベースで、互換性のあるデータベース文字セットを使用している必要があります。具体的には、次の内容のいずれかを満たしている必要があります。
  - ソース・データベースとターゲット・データベースのデータベース文字セットが同じです。
  - ソース・データベース文字セットがターゲット・データベース文字セットの厳格な(バイナリ)サブセットであり、かつ、次の3つの条件が満たされています。
    - ソース・データベースは、Oracle Database 10g リリース1 (10.1.0.3)以上です。
    - トランスポート対象の表領域に、文字長セマンティクスが使用される表の列が含まれていないか、またはソースとターゲットの両方のデータベースにおいて、データベース文字セットの最大文字幅が同じです。
    - トランスポート対象のデータにはCLOBデータ型の列が含まれていないか、またはソース・データベースとターゲット・データベースにおいて、データベース文字セットが両方ともシングルバイトであるか、両方ともマルチバイトです。
  - ソース・データベース文字セットがターゲット・データベース文字セットの厳格な(バイナリ)サブセットであり、かつ、次の2つの条件が満たされています。
    - ソース・データベースは、Oracle Database 10g リリース1 (10.1.0.3)以前です。
    - ソース・データベースとターゲット・データベース文字セットにおいて、最大文字幅が同じです。

  ノート:

  Oracle Database によって認識される文字セット間のサブセットとスーパーセットの関係は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- ソースとターゲットのデータベースで、互換性のある各国語文字セットを使用している必要があります。具体的には、次の内容のいずれかを満たしている必要があります。
  - ソースとターゲットのデータベースの各国語文字セットが同じです。
  - ソース・データベースはOracle Database 10g リリース1 (10.1.0.3)以上であり、トランスポート対象の表領域にはNCHAR、NVARCHAR2、NCLOBのデータ型の列が含まれていません。
- トランスポータブル・エクスポート操作を実行する場合は、次の制限が適用されます。
  - エクスポートを実行するユーザーのデフォルトの表領域を、転送対象となっている表領域のいずれかにすることはできません。
  - エクスポートを実行するユーザーのデフォルトの表領域を、書込み可能にする必要があります。
- 非CDBで、ターゲット・データベースに同じ名前の表領域が含まれている場合は、表領域をトランスポートできません。

  CDBで、ターゲットのコンテナに同じ名前の表領域が含まれている場合は、表領域をトランスポートできません。ただし、異なるコンテナには同じ名前の表領域を格納できます。

  REMAP_TABLESPACEインポート・パラメータを使用して、データベース・オブジェクトを異なる表領域にインポートでき

ます。または、トランスポート操作を実行する前に、トランスポート対象の表領域またはターゲットの表領域のいずれかの名前を変更できます。

Oracle Database 12cリリース2 (12.2)以降では、Recovery Manager (RMAN)のRECOVERコマンドで、表領域を再マッピングしながら表を異なるスキーマに移動できます。詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

- CDBでは、デフォルトのデータ・ポンプ・ディレクトリ・オブジェクトDATA_PUMP_DIRはPDBでは機能しません。データ・ポンプ・エクスポートおよびインポートで使用する明示的なディレクトリ・オブジェクトをPDB内に定義する必要があります。
- XMLTypeを含むデータのトランスポートには、次の制限事項があります。
  - ターゲット・データベースにXML DBがインストールされている必要があります。
  - XMLType表が参照するスキーマをXML DB標準スキーマにすることはできません。
  - トランスポートされたXMLType表のスキーマがターゲット・データベース内に存在しない場合は、スキーマがインポートおよび登録されます。ターゲット・データベース内にすでにスキーマが存在する場合は、インポート中にメッセージが表示されます。
  - XMLTypeを含むデータのメタデータは、データ・ポンプのみを使用してエクスポートおよびインポートする必要があります。

  次の問合せでは、XMLTypeを含む表領域のリストが返されます。

```
select distinct p.tablespace_name from dba_tablespaces p,
  dba_xml_tables x, dba_users u, all_all_tables t where
  t.table_name=x.table_name and t.tablespace_name=p.tablespace_name
  and x.owner=u.username;
```

  XMLTypeの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

- 解釈がアプリケーション固有で、データベースに対して不透明なタイプ(RAW、BFILEなど)は、トランスポートできますが、クロス・プラットフォームのトランスポート操作では変換されません。このタイプの実際の構造はアプリケーションのみが認識するため、このタイプが新規プラットフォームに移動した後、アプリケーションではendiannessの問題に対処する必要があります。OPAQUE型を使用するタイプとオブジェクトも、直接的または間接的にこの制限の影響を受けます。
- TIMESTAMP WITH LOCAL TIME ZONE (TSLTZ)データを含む表がある表領域をタイム・ゾーンが異なるデータベース間でトランスポートする場合、TSLTZデータを含む表はトランスポートされません。エラー・メッセージで、トランスポートされなかった表の説明が示されます。ただし、表領域内のTSLTZデータを含まない表はトランスポートされます。

  データベースのタイム・ゾーンを確認するには、次の問合せを使用します。

```
SELECT DBTIMEZONE FROM DUAL;
```

  ALTER DATABASE SQL文を使用すると、データベースのタイム・ゾーンを変更できます。

  データ・ポンプを使用すると、トランスポート操作の完了後に、TSLTZデータを含む表の従来のエクスポート/インポートを実行できます。

- プラットフォーム間でのトランスポート操作の一部として、アナリティック・ワークスペースは使用できません。ソース・プラットフォームおよびターゲット・プラットフォームが別な場合、Data Pumpエクスポート/インポートを使用し、アナリティック・ワークスペースをエクスポートおよびインポートします。アナリティック・ワークスペースの詳細は、『Oracle OLAP DMLリファレンス』を参照してください。

ノート:

データ・ポンプ・エクスポート・ユーティリティ expdp またはインポート・ユーティリティ impdp は、Oracle サポート・サービスから要求された場合以外、SYSDBA として起動しないでください。SYSDBA は内部的に使用され、一般ユーザーとは異なる特別な機能を持ちます。

関連トピック

- フル・トランスポータブル・エクスポート/インポートに関する制限事項
- トランスポータブル表領域に関する制限事項
- トランスポータブル表に関する制限事項

親トピック: データのトランスポートについて

15.1.5 データのトランスポートの互換性に関する注意事項

データをトランスポートする場合、Oracle Databaseはターゲット・データベースが稼働する最低限の互換性レベルを計算します。

ターゲット・データベースが同じプラットフォームにある場合も、別のプラットフォームにある場合も、トランスポータブル表領域を使用して、同じ互換性または高い互換性が設定されているターゲット・データベースにソース・データベースから表領域または表をトランスポートできます。ソース・データベースの互換性レベルがターゲット・データベースの互換性レベルよりも高い場合、データ・トランスポート操作が失敗します。

次の表に、様々な使用例でのソース・データベースとターゲット・データベースの互換性の最低要件を示します。ソース・データベースとターゲット・データベースの互換性設定は同一である必要はありません。

表15-2 トランスポート・シナリオの互換性の最低要件

トランスポートの使用例	ソースの最低のデータベース互換性	ターゲットの最低のデータベース互換性
フル・トランスポータブル・エクスポート/インポートを使用したデータベースのトランスポート	12.0 (Oracle Database 12c 以降のデータベースの COMPATIBLE 初期化パラメータの設定) 12 (11.2.0.3 以降のデータベースの VERSION Oracle Data Pump エクスポート・パラメータの設定)	12.0 (COMPATIBLE 初期化パラメータの設定)
トランスポータブル表領域を使用した、同じプラットフォーム上のデータベース間での表領域のトランスポート	8.0 (COMPATIBLE 初期化パラメータの設定)	8.0 (COMPATIBLE 初期化パラメータの設定)
トランスポータブル表領域を使用した、ターゲット・データベースとデータベース・ブロック・サイズが異なる表領域のトランスポート	9.0 (COMPATIBLE 初期化パラメータの設定)	9.0 (COMPATIBLE 初期化パラメータの設定)

トランスポートの使用例	ソースの最低のデータベース互換性	ターゲットの最低のデータベース互換性
トランスポータブル表領域を使用した、異なるプラットフォーム上のデータベース間での表領域のトランスポート	10.0 (COMPATIBLE 初期化パラメータの設定)	10.0 (COMPATIBLE 初期化パラメータの設定)
データベース間での表のトランスポート	11.2.0 (Oracle Database 12c 以降のデータベースの COMPATIBLE 初期化パラメータの設定)	11.2.0 (COMPATIBLE 初期化パラメータの設定)

ノート:

- フル・トランスポータブル・エクスポートおよびインポートを使用する場合、ソース・データベースは Oracle Database 11g リリース 2 (11.2.0.3)以降のデータベースであり、ターゲット・データベースは Oracle Database 12c 以降のデータベースである必要があります。
- Oracle Database 11g リリース 2 (11.2.0.3)以降のデータベースから Oracle Database 12c 以降のデータベースにトランスポートする場合は、VERSION Oracle Data Pump エクスポート・パラメータを 12 以上に設定する必要があります。
- Oracle Database 19c データベースから Oracle Database 19c データベース以降のリリースにトランスポートする場合は、COMPATIBLE 初期化パラメータを 19.0.0 以上に設定する必要があります。

親トピック: データのトランスポートについて

15.2 データベースのトランスポート

新しいOracle Databaseインスタンスへデータベースをトランスポートできます。

- フル・トランスポータブル・エクスポート/インポートの概要
  フル・トランスポータブル・エクスポート/インポート機能を使用すると、データベース全体を別のOracle Databaseインスタンスにコピーできます。
- フル・トランスポータブル・エクスポート/インポートに関する制限事項
  フル・トランスポータブル・エクスポート/インポートには制限事項があります。
- エクスポート・ダンプ・ファイルを使用したデータベースのトランスポート
  エクスポート・ダンプ・ファイルを使用してデータベースをトランスポートできます。
- ネットワーク経由でのデータベースのトランスポート
  ネットワーク経由でデータベースをトランスポートできます。

親トピック: データのトランスポート

15.2.1 フル・トランスポータブル・エクスポート/インポートの概要

フル・トランスポータブル・エクスポート/インポート機能を使用すると、データベース全体を別のOracle Databaseインスタンスに

コピーできます。

データ・ポンプを使用すると、エクスポート・ダンプ・ファイルを生成し、必要に応じてダンプ・ファイルをターゲット・データベースにトランスポートした後、エクスポート・ダンプ・ファイルをインポートできます。また、データ・ポンプを使用すると、ネットワークを介してデータベースをコピーすることもできます。

トランスポートするデータベース内の表領域は、ディクショナリ管理表領域またはローカル管理表領域のいずれかになります。データベース内の表領域は、ターゲット・データベースの標準ブロック・サイズと同じブロック・サイズにする必要はありません。

ノート:

この方法でデータベースをトランスポートする場合は、エクスポートが完了するまで、データベース内のユーザー定義表領域を読取り専用モードにする必要があります。これが望ましくない場合は、トランスポータブル表領域をバックアップ機能から使用できます。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

関連項目:

「データのトランスポートについて」

親トピック: データベースのトランスポート

15.2.2 フル・トランスポータブル・エクスポート/インポートに関する制限事項

フル・トランスポータブル・エクスポート/インポートには制限事項があります。

フル・トランスポータブル・エクスポート/インポートに関する次の制限事項に注意してください。

- フル・トランスポータブル・エクスポート/インポートには、「データのトランスポートに関する一般的な制限事項」で説明されている一般的な制限事項が適用されます。
- フル・トランスポータブル・エクスポート/インポートでは、ダイレクト・パスや外部表などの従来のデータ・ポンプ・エクスポート/インポートを使用して、管理表領域内のユーザー定義のデータベース・オブジェクトをエクスポートおよびインポートできます。管理表領域は、Oracle Databaseに用意されたSYSTEM表領域やSYSAUX表領域などの非ユーザー表領域です。
- フル・トランスポータブル・エクスポート/インポートでは、管理表領域(SYSTEMやSYSAUXなど)とユーザー定義表領域の両方で定義されているデータベース・オブジェクトはトランスポートできません。たとえば、パーティション表は、ユーザー定義表領域と管理表領域の両方に格納されることがあります。データベースにこのようなデータベース・オブジェクトがある場合は、それらのすべてのデータベース・オブジェクトが、管理表領域またはユーザー定義表領域のいずれかに格納されるように再定義した後、トランスポートできます。データベース・オブジェクトを再定義できない場合は、従来のデータ・ポンプ・エクスポート/インポートを使用できます。
- フル・トランスポータブル・エクスポート/インポートを使用して、ネットワーク経由でデータベースをトランスポートする場合に、監査証跡情報自体がユーザー定義表領域に格納されていると、管理表領域(SYSTEMやSYSAUXなど)に格納されている表に対して監査を有効にできません。詳細は、『*Oracle Database*セキュリティ・ガイド』を参照してください。
- トランザクション・ガードが有効になっている場合、フル・トランスポータブル・データベース・インポートは実行できません。全データベースのインポートの間はトランザクション・ガードを無効にしてください。

関連トピック

- 監査の概要

親トピック: データベースのトランスポート

15.2.3 エクスポート・ダンプ・ファイルを使用したデータベースのトランスポート

エクスポート・ダンプ・ファイルを使用してデータベースをトランスポートできます。

次のタスクのリストでは、エクスポート・ダンプ・ファイルを使用したデータベースのトランスポート処理の概要を示します。各タスクの詳細は、後続の例で示します。

1. ソース・データベースで、各ユーザー定義表領域を読取り専用モードに構成し、データベースをエクスポートします。

   次のパラメータが指定された値に設定されていることを確認します。

   - TRANSPORTABLE=ALWAYS
   - FULL=Y

   ソース・データベースがOracle Database 11gデータベース(11.2.0.3以降)の場合、VERSIONパラメータを12以上に設定する必要があります。

   ソース・データベースに暗号化された表領域、または暗号化された列を含む表が格納された表領域が含まれている場合は、ENCRYPTION_PWD_PROMPT=YESを指定するか、ENCRYPTION_PASSWORDパラメータを指定する必要があります。

   エクスポート・ダンプ・ファイルには、ユーザー定義表領域に格納されたオブジェクトのメタデータ、および管理表領域(SYSTEMやSYSAUXなど)に格納されたユーザー定義オブジェクトのメタデータとデータの両方が含まれています。

2. エクスポート・ダンプ・ファイルをトランスポートします。

   エクスポート・ダンプ・ファイルをターゲット・データベースへアクセス可能な場所にコピーします。

3. データベース内のユーザー定義表領域すべてのデータファイルをトランスポートします。

   データファイルをターゲット・データベースへアクセス可能な場所にコピーします。

   ソース・プラットフォームとターゲット・プラットフォームが異なる場合は、V$TRANSPORTABLE_PLATFORMビューに対して問合せを実行して、各プラットフォームのendian形式をチェックできます(「プラットフォーム間でのデータのトランスポート」を参照)。

   ソース・プラットフォームのendian形式がターゲット・プラットフォームのendian形式と異なる場合は、次のいずれかの方法を使用してデータファイルを変換します。

   - DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用して、データファイルを転送します。これらのプロシージャを使用すると、データファイルがターゲット・プラットフォームのendian形式に自動的に変換されます。
   - RMAN CONVERTコマンドを使用して、データファイルをターゲット・プラットフォームのendian形式に変換します。

   ノート:

   UNDO セグメントを含むデータファイルでは、異なる endian 形式間でのデータファイルの変換

はサポートされていません。

詳細は、「プラットフォーム間でのデータの変換」を参照してください。

4. (オプション)ソース・データベースでユーザー定義表領域を読取り/書込みモードに戻します。
5. ターゲット・データベースで、データベースをインポートします。

   インポートが完了したとき、ユーザー定義表領域は読取り/書込みモードになります。

例

データベースをトランスポートするためのこれらのタスクは、この例で詳しく説明します。この例では、ソース・プラットフォームがSolarisで、ターゲット・プラットフォームがMicrosoft Windowsであることを前提としています。

また、ソース・プラットフォームには次のデータファイルと表領域があるとします。

表領域	タイプ	データファイル
sales	ユーザー定義	/u01/app/oracle/oradata/mydb/sales01.dbf
customers	ユーザー定義	/u01/app/oracle/oradata/mydb/cust01.dbf
employees	ユーザー定義	/u01/app/oracle/oradata/mydb/emp01.dbf
SYSTEM	管理	/u01/app/oracle/oradata/mydb/system01.dbf
SYSAUX	管理	/u01/app/oracle/oradata/mydb/sysaux01.dbf

この例では、さらに次のことを想定しています。

- ターゲット・データベースは、ソース・データベースからデータを移入する新しいデータベースです。ソース・データベースの名前はmydbです。
- ソース・データベースとターゲット・データベースの両方がOracle Database 19cデータベースです。

エクスポート・ダンプ・ファイルを使用してデータベースをトランスポートするには、次のタスクを実行します。

タスク 1　エクスポート・ダンプ・ファイルの作成

次のステップを実行して、エクスポート・ダンプ・ファイルを生成します。

1. SQL*Plus を起動し、管理者として、あるいは ALTER TABLESPACE または MANAGE TABLESPACE システム権限を持つユーザーとしてデータベースに接続します。
2. データベース内のすべてのユーザー定義表領域を読取り専用にします。

```
ALTER TABLESPACE sales READ ONLY;
ALTER TABLESPACE customers READ ONLY;
ALTER TABLESPACE employees READ ONLY;
```

3. DATAPUMP_EXP_FULL_DATABASE ロールを持つユーザーとしてデータ・ポンプ・エクスポート・ユーティリティを起動し、フル・トランスポータブル・エクスポート/インポート・オプションを指定します。

```
SQL> HOST
$ expdp user_name full=y dumpfile=expdat.dmp directory=data_pump_dir
        transportable=always logfile=export.log
Password: password
```

トランスポータブル・オプションを使用するかどうかを設定する TRANSPORTABLE=ALWAYS を常に指定する必要があります。

この例では、次のデータ・ポンプ・パラメータを指定します。

- FULL パラメータでは、データベース全体をエクスポートすることを指定します。
- DUMPFILE パラメータでは、作成する構造情報エクスポート・ダンプ・ファイルの名前を expdat.dmp と指定します。
- DIRECTORY パラメータでは、オペレーティング・システムまたは Oracle Automatic Storage Management のダンプ・ファイルの場所を示すディレクトリ・オブジェクトを指定します。DIRECTORY オブジェクトはデータ・ポンプを起動する前に作成し、ディレクトリに対する READ および WRITE オブジェクト権限をエクスポート・ユーティリティを実行するユーザーに付与する必要があります。CREATE DIRECTORY コマンドの詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

  非 CDB で、ディレクトリ・オブジェクト DATA_PUMP_DIR が自動的に作成されます。このディレクトリへの読取りおよび書込みアクセス権が DBA ロールに(したがって、ユーザー SYS および SYSTEM に)自動的に付与されます。

  ただし、ディレクトリ・オブジェクト DATA_PUMP_DIR は、PDB では自動的には作成されません。このため、PDB にインポートする場合は、PDB にディレクトリ・オブジェクトを作成し、データ・ポンプを実行するときにそのディレクトリ・オブジェクトを指定します。

  関連項目:

  - DIRECTORY パラメータを省略する場合のデフォルト・ディレクトリの詳細は、『Oracle Database ユーティリティ』を参照してください。
  - PDB の詳細は、『Oracle Multitenant 管理者ガイド』を参照してください

- LOGFILE パラメータでは、エクスポート・ユーティリティによって書き込まれるログ・ファイルのファイ

ル名を指定します。この例では、ログ・ファイルの書込み先はダンプ・ファイルと同じディレクトリですが、ログ・ファイルは別の場所に書き込むことができます。

Oracle Database 11g リリース 2 (11.2.0.3)または Oracle Database 11g 以上のデータベースでフル・トランスポータブル・エクスポートを実行するには、次の例のように、VERSION パラメータを使用します。

```
expdp user_name full=y dumpfile=expdat.dmp directory=data_pump_dir
      transportable=always version=12 logfile=export.log
```

フル・トランスポータブル・インポートは、Oracle Database 12c 以降のデータベースでのみサポートされます。

ノート:

この例では、データ・ポンプ・ユーティリティを使用してエクスポートするのは、ユーザー定義表領域のデータ・ディクショナリの構造情報(メタデータ)のみです。実際のデータは管理表領域(SYSTEM および SYSAUX)についてのみアンロードされるため、この操作は大規模なユーザー定義表領域の場合にも比較的短時間で完了します。

4. ログ・ファイルでエラーを確認し、ターゲット・データベースにトランスポートする必要があるダンプ・ファイルとデータファイルをノートにとります。expdp により、これらのファイルの名前とパスが次のようなメッセージに出力されます。

```
******************************************************************************
****
Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
  /u01/app/oracle/admin/mydb/dpdump/expdat.dmp
******************************************************************************
****
Datafiles required for transportable tablespace SALES:
  /u01/app/oracle/oradata/mydb/sales01.dbf
Datafiles required for transportable tablespace CUSTOMERS:
  /u01/app/oracle/oradata/mydb/cust01.dbf
Datafiles required for transportable tablespace EMPLOYEES:
  /u01/app/oracle/oradata/mydb/emp01.dbf
```

5. 完了した後、終了して SQL*Plus に戻ります。

```
$ exit
```

関連項目:

データ・ポンプ・ユーティリティの使用方法は、『Oracle Database ユーティリティ』を参照してください

ダンプ・ファイルを、DATA_PUMP_DIR ディレクトリ・オブジェクトで指し示されているディレクトリ、または他の任意のディレクトリにトランスポートします。新しい場所はターゲット・データベースへアクセス可能であることが必要です。

ターゲット・データベースで、次の問合せを実行して DATA_PUMP_DIR の場所を確認します。

```
SELECT * FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME = 'DATA_PUMP_DIR';
OWNER      DIRECTORY_NAME   DIRECTORY_PATH
---------- ---------------- ----------------------------------
SYS        DATA_PUMP_DIR    C:¥app¥orauser¥admin¥orawin¥dpdump¥
```

タスク 3　ユーザー定義表領域のデータファイルのトランスポート

データベースのユーザー定義表領域のデータファイルをターゲット・データベースへアクセス可能な場所にトランスポートします。

この例では、次のデータファイルをソース・データベースからターゲット・データベースに転送します。

- sales01.dbf
- cust01.dbf
- emp01.dbf

ソース・プラットフォームとは異なるプラットフォームにデータベースをトランスポートする場合は、ソースおよびターゲット・プラットフォームの両方でプラットフォーム間のデータベース・トランスポートがサポートされているかどうかを確認し、それぞれのプラットフォームの endianness を判別します。両方のプラットフォームの endianness が同じ場合、変換は必要ありません。同じでない場合は、ソース・データベースまたはターゲット・データベースのどちらかでデータベース内の各表領域を変換する必要があります。

データベースを異なるプラットフォームにトランスポートする場合は、各プラットフォームで次の問合せを実行できます。問合せで行が返される場合、そのプラットフォームではプラットフォーム間の表領域トランスポートがサポートされています。

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

ソース・プラットフォームでの問合せの結果は次のとおりです。

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Solaris[tm] OE (32-bit)          Big
```

ターゲット・プラットフォームからの問合せの結果は、次のとおりです。

```
PLATFORM_NAME                    ENDIAN_FORMAT
```

```
---------------------------------- --------------
Microsoft Windows IA (32-bit)      Little
```

この例では、endian 形式が異なることがわかります。したがって、この場合、データベースをトランスポートするには変換が必要です。DBMS_FILE_TRANSFER パッケージの GET_FILE または PUT_FILE プロシージャを使用して、データファイルを転送します。これらのプロシージャを使用すると、データファイルがターゲット・プラットフォームの endian 形式に自動的に変換されます。データファイルを、ターゲット・データベースの既存データファイルの場所にトランスポートします。UNIX および Linux プラットフォームでは、この場所は通常、/u01/app/oracle/oradata/dbname/または +DISKGROUP/dbname/datafile/です。また、RMAN CONVERT コマンドを使用してデータファイルを変換できます。詳細は、「プラットフォーム間でのデータの変換」を参照してください。

ノート:

表領域の endianness を変換する必要がない場合は、任意のファイル転送方法を使用してファイルを転送できます。

タスク 4　(オプション)表領域を読取り/書込みモードに戻す

次のように、トランスポートした表領域をソース・データベースで再び読取り/書込みモードにします。

```
ALTER TABLESPACE sales READ WRITE;
ALTER TABLESPACE customers READ WRITE;
ALTER TABLESPACE employees READ WRITE;
```

インポート・プロセスが成功したことを先に確認するために、このタスクを延期することができます。

タスク 5　ターゲット・データベースでのデータベースのインポート

DATAPUMP_IMP_FULL_DATABASE ロールを持つユーザーとしてデータ・ポンプ・インポート・ユーティリティを起動し、フル・トランスポータブル・エクスポート/インポート・オプションを指定します。

```
impdp user_name full=Y dumpfile=expdat.dmp directory=data_pump_dir
   transport_datafiles=
      '/u01/app/oracle/oradata/mydb/sales01.dbf',
      '/u01/app/oracle/oradata/mydb/cust01.dbf',
      '/u01/app/oracle/oradata/mydb/emp01.dbf'
   logfile=import.log
Password: password
```

この例では、次のデータ・ポンプ・パラメータを指定します。

- FULL パラメータでは、データベース全体を FULL モードでインポートすることを指定します。
- DUMPFILE パラメータでは、インポートされるユーザー定義表領域のメタデータおよび管理表領域のメタデータとデータの両方が含まれるエクスポート・ファイルを指定します。

- DIRECTORY パラメータでは、エクスポート・ダンプ・ファイルの場所を識別するディレクトリ・オブジェクトを指定します。DIRECTORY オブジェクトはデータ・ポンプを起動する前に作成し、ディレクトリに対する READ および WRITE オブジェクト権限をインポート・ユーティリティを実行するユーザーに付与する必要があります。CREATE DIRECTORY コマンドの詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

  非 CDB で、ディレクトリ・オブジェクト DATA_PUMP_DIR が自動的に作成されます。このディレクトリへの読取りおよび書込みアクセス権が DBA ロールに(したがって、ユーザーSYS および SYSTEM に)自動的に付与されます。

  ただし、ディレクトリ・オブジェクト DATA_PUMP_DIR は、PDB では自動的には作成されません。このため、PDB にインポートする場合は、PDB にディレクトリ・オブジェクトを作成し、データ・ポンプを実行するときにそのディレクトリ・オブジェクトを指定します。

  関連項目:

- DIRECTORY パラメータを省略する場合のデフォルト・ディレクトリの詳細は、『Oracle Database ユーティリティ』を参照してください。

- PDB の詳細は、『Oracle Multitenant 管理者ガイド』を参照してください

- TRANSPORT_DATAFILES パラメータによって、インポートするすべてのデータファイルを識別します。

  多くのデータファイルがある場合は、PARFILE パラメータで指定されたパラメータ・ファイルで TRANSPORT_DATAFILES パラメータを複数回指定できます。

- LOGFILE パラメータでは、インポート・ユーティリティによって書き込まれるログ・ファイルのファイル名を指定します。この例では、ログ・ファイルの書込み先はダンプ・ファイルの読取り元と同じディレクトリですが、ログ・ファイルは別の場所に書き込むことができます。

この文が正常に実行された後、インポート・ログ・ファイルをチェックして、予期しないエラーが発生していないことを確認します。

多数のデータファイルを扱う場合、データファイル名のリストを文の行で指定することは煩雑です。また、文の行制限を超える場合もあります。このような場合には、インポート・パラメータ・ファイルを使用できます。たとえば、次のようにしてデータ・ポンプ・インポート・ユーティリティを起動できます。

```
impdp user_name parfile='par.f'
```

たとえば、par.f には次の行が含まれる場合があります。

```
FULL=Y
DUMPFILE=expdat.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES=
'/u01/app/oracle/oradata/mydb/sales01.dbf',
'/u01/app/oracle/oradata/mydb/cust01.dbf',
'/u01/app/oracle/oradata/mydb/emp01.dbf'
LOGFILE=import.log
```

ノート:

- インポート中に、メタデータのロードのためにユーザー定義表領域を一時的に読取り/書込みにすることがあります。インポート中にデータに対するユーザー変更が行われないことを確認してください。インポートが正常に完了したとき、すべてのユーザー定義表領域は読取り/書込みになります。
- ネットワーク・データベース・インポートを実行する場合は、TRANSPORTABLE パラメータを always に設定する必要があります。
- CDB 内の PDB にインポートする場合は、ユーザー名の後に PDB の接続識別子を指定します。たとえば、PDB の接続識別子が hrpdb である場合は、Oracle Data Pump インポート・ユーティリティを実行するとき、次のように入力します。

```
impdp user_name@hrpdb ...
```

関連項目:

- インポート・ユーティリティの使用方法は、『Oracle Database ユーティリティ』を参照してください。
- Oracle Multitenant 管理者ガイド

親トピック: データベースのトランスポート

15.2.4 ネットワーク経由でのデータベースのトランスポート

ネットワーク経由でデータベースをトランスポートできます。

ネットワーク経由でデータベースをトランスポートするには、NETWORK_LINKパラメータを使用してインポートを実行しますが、インポートはデータベース・リンクを使用して実行され、ダンプ・ファイルは関連しません。

次のタスクのリストでは、ネットワーク経由での、データベースのトランスポート処理の概要を示します。各タスクの詳細は、後続の例で示します。

1. ターゲット・データベースからソース・データベースへデータベース・リンクを作成します。

   インポート操作は、ターゲット・データベースでDATAPUMP_IMP_FULL_DATABASEロールを持つユーザーが実行する必要があり、データベース・リンクは、ソース・データベースでDATAPUMP_EXP_FULL_DATABASEロールを持つユー

ザーに接続する必要があります。ソース・データベースのユーザーは、SYSDBA管理権限を持つユーザーにすることはできません。データベース・リンクが接続ユーザー・データベース・リンクである場合は、ターゲット・データベース上のユーザーをSYSDBA管理権限を持つユーザーにはできません。接続ユーザー・データベース・リンクの詳細は、「データベース・リンクのユーザー」を参照してください。

2. ソース・データベースで、データベース内のユーザー定義表領域を読取り専用にします。
3. データベース内のユーザー定義表領域すべてのデータファイルをトランスポートします。

   データファイルをターゲット・データベースへアクセス可能な場所にコピーします。

   ソース・プラットフォームとターゲット・プラットフォームが異なる場合は、V$TRANSPORTABLE_PLATFORMビューに対して問合せを実行して、各プラットフォームのendian形式をチェックできます(「プラットフォーム間でのデータのトランスポート」を参照)。

   ソース・プラットフォームのendian形式がターゲット・プラットフォームのendian形式と異なる場合は、次のいずれかの方法を使用してデータファイルを変換します。

   - DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用して、データファイルを転送します。これらのプロシージャを使用すると、データファイルがターゲット・プラットフォームのendian形式に自動的に変換されます。
   - RMAN CONVERTコマンドを使用して、データファイルをターゲット・プラットフォームのendian形式に変換します。

   > ノート:
   >
   > UNDO セグメントを含むデータファイルでは、異なる endian 形式間でのデータファイルの変換はサポートされていません。

   詳細は、「プラットフォーム間でのデータの変換」を参照してください。
4. ターゲット・データベースで、データベースをインポートします。

   データ・ポンプ・ユーティリティを起動して、ユーザー定義表領域のメタデータおよび管理表領域のメタデータとデータの両方をインポートします。

   次のパラメータが指定された値に設定されていることを確認します。

   - TRANSPORTABLE=ALWAYS
   - TRANSPORT_DATAFILES=list_of_datafiles
   - FULL=Y
   - NETWORK_LINK=source_database_link

     source_database_linkをソース・データベースへのデータベース・リンクの名前に置き換えます。
   - VERSION=12

     ソース・データベースがOracle Database 11g リリース2 (11.2.0.3)またはOracle Database 11g以上のデータベースである場合、VERSIONパラメータが必要で、これを12に設定する必要があります。ソース・データベースがOracle Database 12c以降のデータベースの場合は、VERSIONパラメータは必要ありません。

ソース・データベースに暗号化された表領域、または暗号化された列を含む表が格納された表領域が含まれている場合は、ENCRYPTION_PWD_PROMPT=YESを指定するか、ENCRYPTION_PASSWORDパラメータを指定する必要があります。

データ・ポンプ・ネットワーク・インポートでは、ユーザー定義表領域に格納されたオブジェクトのメタデータ、および管理表領域(SYSTEMやSYSAUXなど)に格納されたユーザー定義オブジェクトのメタデータとデータの両方がコピーされます。

インポートが完了したとき、ユーザー定義表領域は読取り/書込みモードになります。

5. (オプション)ソース・データベースでユーザー定義表領域を読取り/書込みモードに戻します。

例

データベースをトランスポートするタスクについては、次のデータファイルと表領域を想定した例で詳細に説明します。

表領域	タイプ	データファイル
sales	ユーザー定義	/u01/app/oracle/oradata/mydb/sales01.dbf
customers	ユーザー定義	/u01/app/oracle/oradata/mydb/cust01.dbf
employees	ユーザー定義	/u01/app/oracle/oradata/mydb/emp01.dbf
SYSTEM	管理	/u01/app/oracle/oradata/mydb/system01.dbf
SYSAUX	管理	/u01/app/oracle/oradata/mydb/sysaux01.dbf

この例では、さらに次のことを想定しています。

- ターゲット・データベースは、ソース・データベースからデータを移入する新しいデータベースです。ソース・データベースの名前はsourcedbです。
- ソース・データベースとターゲット・データベースは、endiannessが同一の同じプラットフォームで実行されている必要があります。

  プラットフォームのendiannessをチェックするには、次の問合せを実行します。

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

- sales表領域は暗号化されています。他の表領域は暗号化されていません。
- ソース・データはOracle Database 11g リリース2 (11.2.0.3)データベースで、ターゲット・データベースはOracle Database 19cデータベースです。

ノート:

この例では、Oracle Database 11g リリース 2 (11.2.0.3)を CDB 内の新しい Oracle Database 19c PDB にトランスポートするために必要なタスクを説明しています。これらのタスクでは、非 CDB を別の非 CDB にト

ランスポートする方法も示されています。

関連項目:

Oracle Multitenant管理者ガイド

ネットワーク経由でデータベースをトランスポートするには、次のタスクを実行します。

タスク 1　ターゲット・データベースからソース・データベースへのデータベース・リンクの作成

次のステップを実行して、ターゲット・データベースからソース・データベースへデータベース・リンクを作成します。

1. ソース・データベースとターゲット・データベースの間にネットワーク接続が構成されていることを確認します。

   詳細は、『Oracle Database Net Services 管理者ガイド』を参照してください。

2. SQL*Plus を起動し、データ・ポンプ・インポートによってデータベースをトランスポートする管理者としてターゲット・データベースに接続します。データベースをトランスポートするには、このユーザーに DATAPUMP_IMP_FULL_DATABASE ロールが必要です。

   手順は、「SQL*Plus を使用したデータベースへの接続」を参照してください。

3. データベース・リンクを作成します。

   ```
   CREATE PUBLIC DATABASE LINK sourcedb USING 'sourcedb';
   ```

   USING 句でソース・データベースのサービス名を指定します。

   インポート操作中、データベース・リンクをソース・データベースで DATAPUMP_EXP_FULL_DATABASE ロールを持つユーザーに接続する必要があります。ソース・データベースのユーザーは、SYSDBA 管理権限を持つユーザーにすることはできません。

関連項目:

- 「データベース・リンクの作成」
- Oracle Database SQL 言語リファレンス

タスク 2　ユーザー定義表領域を読取り専用にする

ステップは次のとおりです。

1. SQL*Plus を起動し、管理者として、あるいは ALTER TABLESPACE または MANAGE TABLESPACE システム権限を持つユーザーとしてソース・データベースに接続します。

   手順は、「SQL*Plus を使用したデータベースへの接続」を参照してください。

2. データベース内のすべてのユーザー定義表領域を読取り専用にします。

```
ALTER TABLESPACE sales READ ONLY;
ALTER TABLESPACE customers READ ONLY;
ALTER TABLESPACE employees READ ONLY;
```

タスク 3　ユーザー定義表領域のデータファイルのトランスポート

データファイルを、ターゲット・データベースの既存データファイルの場所にトランスポートします。

UNIX および Linux プラットフォームでは、この場所は通常、/u01/app/oracle/oradata/dbname/または+DISKGROUP/dbname/datafile/です。

この例では、次のデータファイルをソース・データベースからターゲット・データベースに転送します。

- `sales01.dbf`
- `cust01.dbf`
- `emp01.dbf`

関連項目:

「データファイルを転送するためのガイドライン」

タスク 4　ターゲット・データベースでのデータベースのインポート

DATAPUMP_IMP_FULL_DATABASE ロールを持つユーザーとしてデータ・ポンプ・インポート・ユーティリティを起動し、フル・トランスポータブル・エクスポート/インポート・オプションを指定します。

```
impdp user_name full=Y network_link=sourcedb transportable=always
   transport_datafiles=
      '/u01/app/oracle/oradata/mydb/sales01.dbf',
      '/u01/app/oracle/oradata/mydb/cust01.dbf',
      '/u01/app/oracle/oradata/mydb/emp01.dbf'
   encryption_pwd_prompt=YES version=12 logfile=import.log
Password: password
```

この例では、次のデータ・ポンプ・パラメータを指定します。

- FULL パラメータでは、データベース全体を FULL モードでインポートすることを指定します。
- NETWORK_LINK パラメータでは、ネットワーク・インポートに使用されるデータベース・リンクを指定します。
- TRANSPORTABLE パラメータでは、インポートでトランスポータブル・オプションを使用することを指定します。
- TRANSPORT_DATAFILES パラメータによって、インポートするすべてのデータファイルを識別します。

  多くのデータファイルがある場合は、PARFILE パラメータで指定されたパラメータ・ファイルで TRANSPORT_DATAFILES パラメータを複数回指定できます。
- ENCRYPTION_PWD_PROMPT パラメータは、暗号化パスワードの入力を要求するようデータ・ポンプに指示し、データ・ポンプはネットワーク接続経由で送信されるデータとメタデータを暗号化します。暗号化された表領域または暗号化された列を含む表がインポート操作に含まれている場合は、ENCRYPTION_PWD_PROMPT パラメータまたは ENCRYPTION_PASSWORD パラメータが必須になります。
- ソース・データベースは、Oracle Database 11g リリース 2 (11.2.0.3)または Oracle Database 11g 以上のデータベースであるため、VERSION パラメータを 12 に設定します。
- LOGFILE パラメータでは、インポート・ユーティリティによって書き込まれるログ・ファイルのファイル名を指定します。

この文が正常に実行された後、インポート・ログ・ファイルをチェックして、予期しないエラーが発生していないことを確認します。

多数のデータファイルを扱う場合、データファイル名のリストを文の行で指定することは煩雑です。また、文の行制限を超える場合もあります。このような場合には、インポート・パラメータ・ファイルを使用できます。

暗号化された表領域または暗号化された列を含む表がインポート操作に含まれている場合も、インポート・パラメータ・ファイルを使用することをお薦めします。この場合、インポート・パラメータ・ファイルで ENCRYPTION_PWD_PROMPT=YES を指定します。

たとえば、次のようにしてデータ・ポンプ・インポート・ユーティリティを起動できます。

```
impdp user_name parfile='par.f'
```

たとえば、par.f には次の行が含まれる場合があります。

```
FULL=Y
NETWORK_LINK=sourcedb
TRANSPORTABLE=always
TRANSPORT_DATAFILES=
'/u01/app/oracle/oradata/mydb/sales01.dbf',
'/u01/app/oracle/oradata/mydb/cust01.dbf',
'/u01/app/oracle/oradata/mydb/emp01.dbf'
ENCRYPTION_PWD_PROMPT=YES
VERSION=12
LOGFILE=import.log
```

ノート:

- インポート中に、メタデータのロードのためにユーザー定義表領域を一時的に読取り/書込みにすることがあります。インポート中にデータに対するユーザー変更が行われないことを確認してください。インポートが正常に完了したとき、すべてのユーザー定義表領域は読取り/書込みになります。
- CDB 内の PDB にインポートする場合は、ユーザー名の後に PDB の接続識別子を指定します。たとえば、PDB の接続識別子が `hrpdb` である場合は、Oracle Data Pump インポート・ユーティリティを実行するとき、次のように入力します。

```
impdp user_name@hrpdb ...
```

関連項目:

インポート・ユーティリティの使用方法は、『Oracle Database ユーティリティ』を参照してください。

タスク 5　(オプション)ユーザー定義表領域を読取り/書込みモードに戻す

次のように、ユーザー定義表領域をソース・データベースで再び読取り/書込みモードにします。

```
ALTER TABLESPACE sales READ WRITE;
ALTER TABLESPACE customers READ WRITE;
ALTER TABLESPACE employees READ WRITE;
```

インポート・プロセスが成功したことを先に確認するために、このタスクを延期することができます。

親トピック: データベースのトランスポート

15.3 データベース間での表領域のトランスポート

データベース間で表領域をトランスポートできます。

ノート:

トランスポータブル表領域セットを別のプラットフォーム上の Oracle Database にインポートするには、両方のデータ

ベースの互換性が 10.0.0 以上に設定されている必要があります。リリース・レベルをまたいだ表領域のトランスポートにおけるデータベース互換性の詳細は、「データのトランスポートの互換性に関する注意事項」を参照してください。

- トランスポータブル表領域の概要
  トランスポータブル表領域機能を使用すると、表領域セットを別のOracle Databaseにコピーできます。
- トランスポータブル表領域に関する制限事項
  この項では、トランスポータブル表領域の制限事項を示します。
- データベース間での表領域のトランスポート
  表領域または表領域セットをデータベース間でトランスポートできます。

親トピック: データのトランスポート

15.3.1 トランスポータブル表領域の概要

トランスポータブル表領域機能を使用すると、表領域セットを別のOracle Databaseにコピーできます。

トランスポートする表領域は、ディクショナリ管理表領域またはローカル管理表領域のいずれかになります。トランスポートする表領域は、ターゲット・データベースの標準ブロック・サイズと同じブロック・サイズにする必要はありません。これらの使用例の説明は、「データのトランスポート: 使用例」を参照してください。

表領域をトランスポートする方法は、次の2通りあります。

- 手動で、この項で説明されているステップを実行します。これには、SQL*Plusおよびデータ・ポンプへのコマンドの発行が含まれます。
- Oracle Enterprise Manager Cloud Controlの「表領域のトランスポート」ウィザードの使用。

  「表領域のトランスポート」ウィザードを実行するには:

  1. DATAPUMP_EXP_FULL_DATABASEロールを持つユーザーでCloud Controlにログインします。
  2. データベース・ホームページにアクセスします。
  3. 「スキーマ」メニューから、「データベースのエクスポート/インポート」を選択し、次に「表領域のトランスポート」を選択します。

ノート:

- この方法で表領域をトランスポートする場合は、トランスポート処理が完了するまで、トランスポート対象の表領域を読取り専用モードにする必要があります。これが望ましくない場合は、トランスポータブル表領域をバックアップ機能から使用できます。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- トランスポータブル表領域にはデータ・ポンプを使用する必要があります。XMLType データを Oracle Database 10g リリース 2 (10.2)以前のデータベースに下位移行できるのは、元のインポート・ユーティリティ(IMP)とエクスポート・ユーティリティ(EXP)を使用できる環境がある場合のみです。これらのユーティリティの詳細は『Oracle Database ユーティリティ』を、XMLTypes の詳細は『Oracle XML DB 開発者ガイド』を参照してください。

関連項目:

- 「データのトランスポートについて」
- データ・ウェアハウス環境でトランスポータブル表領域を使用する方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

親トピック: データベース間での表領域のトランスポート

15.3.2 トランスポータブル表領域に関する制限事項

この項では、トランスポータブル表領域の制限事項を示します。

トランスポータブル表領域に関する次の制限事項に注意してください。

- トランスポータブル表領域には、「データのトランスポートに関する一般的な制限事項」で説明されている一般的な制限事項が適用されます。
- 表領域セットをトランスポートする場合、基礎になるオブジェクトを持つオブジェクト(マテリアライズド・ビューなど)またはオブジェクトを含むオブジェクト(パーティション表など)は、それらのオブジェクトがすべて表領域セットに含まれている場合のみトランスポートできます。
- トランスポータブル表領域では、タイム・ゾーン・ファイルのバージョンが異なるプラットフォーム間で、TIMESTAMP WITH TIMEZONE (TSTZ)データを含む表をトランスポートできません。トランスポータブル表領域操作では、これらの表をスキップします。これらの表は、従来と同じようにエクスポートおよびインポートできます。

  詳細は、『Oracle Databaseユーティリティ』を参照してください。
- トランスポータブル表領域セットに、SYSTEMやSYSAUXなどの管理表領域を含めることはできません。
- トランスポータブル表領域には、TDE列暗号化を使用して暗号化された列を含む表を含めることはできません
- 表領域がTDEを使用して暗号化されている場合、この表領域は同じエンディアン形式を使用するプラットフォームにのみトランスポートできます。エンディアン間で移動させる必要がある場合は、表領域を復号化してトランスポートし、再暗号化する必要があります。Oracle Databaseリリース12.2以降では、これらの操作はオンラインで実行できます。

親トピック: データベース間での表領域のトランスポート

15.3.3 データベース間での表領域のトランスポート

データベース間で表領域または表領域のセットをトランスポートできます。

次のタスクのリストでは、表領域のトランスポート処理の概要を示します。各タスクの詳細は、後続の例で示します。

1. 自己完結型の表領域セットの選択
2. ソース・データベースで、表領域セットを読取り専用モードに構成してから、トランスポータブル表領域セットを生成します。

   トランスポータブル表領域セット(トランスポータブル・セット)は、トランスポートされる表領域セットのデータファイルと、そのセットの構造情報(メタデータ)を含むエクスポート・ダンプ・ファイルから構成されます。データ・ポンプを使用してエクスポートを実行します。
3. エクスポート・ダンプ・ファイルをトランスポートします。

エクスポート・ダンプ・ファイルをターゲット・データベースへアクセス可能な場所にコピーします。

4. 表領域セットのトランスポート

   データファイルをターゲット・データベースへアクセス可能なディレクトリにコピーします。

   ソース・プラットフォームとターゲット・プラットフォームが異なる場合は、V$TRANSPORTABLE_PLATFORMビューに対して問合せを実行して、各プラットフォームのendian形式をチェックできます。

   ソース・プラットフォームのendian形式がターゲット・プラットフォームのendian形式と異なる場合は、次のいずれかの方法を使用してデータファイルを変換します。

   - DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用して、データファイルを転送します。これらのプロシージャを使用すると、データファイルがターゲット・プラットフォームのendian形式に自動的に変換されます。
   - RMAN CONVERTコマンドを使用して、データファイルをターゲット・プラットフォームのendian形式に変換します。

   ノート:

   UNDO セグメントを含むデータファイルでは、異なる endian 形式間でのデータファイルの変換はサポートされていません。

5. (オプション)ソース・データベースで表領域を読取り/書込みモードに戻します。
6. ターゲット・データベースで、データベース・セットをインポートします。

   データ・ポンプ・ユーティリティを実行して、表領域セットのメタデータをインポートします。

例15-1 例

表領域をトランスポートするタスクについては、次のデータファイルと表領域を想定した例で詳細に説明します。

表領域	データファイル
sales_1	/u01/app/oracle/oradata/salesdb/sales_101.dbf
sales_2	/u01/app/oracle/oradata/salesdb/sales_201.dbf

- タスク1: 自己完結型の表領域セットの選択
  トランスポータブル・セットのデータベース・オブジェクトとトランスポータブル・セット外のデータベース・オブジェクトの間に、論理的または物理的な依存関係がある場合があります。トランスポートできるのは、自己完結型である表領域セットのみです。つまり、表領域セット内のデータベース・オブジェクトは、その表領域セット外のデータベース・オブジェクトのいずれにも依存しません。
- タスク2: トランスポータブル表領域セットの生成
  トランスポートする表領域セットが自己完結型であることを確認した後で、トランスポータブル表領域セットを生成します。
- タスク3: エクスポート・ダンプ・ファイルのトランスポート
  ダンプ・ファイルを、DATA_PUMP_DIRディレクトリ・オブジェクトで指し示されているディレクトリ、または他の任意のディレクトリにトランスポートします。新しい場所はターゲット・データベースへアクセス可能であることが必要です。
- タスク4: 表領域セットのトランスポート

表領域のデータファイルをターゲット・データベースへアクセス可能なディレクトリにトランスポートします。

- タスク5: (オプション)表領域を読取り/書込みモードに戻す
  トランスポートした表領域をソース・データベースで再び読取り/書込みモードにします。
- タスク6: 表領域セットのインポート
  トランスポータブル表領域に対する操作を完了するために、表領域セットをインポートします。

関連トピック

- プラットフォーム間でのデータ・トランスポート
- プラットフォーム間でのデータの変換

親トピック: データベース間での表領域のトランスポート

15.3.3.1 タスク1: 自己完結型の表領域セットの選択

トランスポータブル・セットのデータベース・オブジェクトとトランスポータブル・セット外のデータベース・オブジェクトの間に、論理的または物理的な依存関係がある場合があります。トランスポートできるのは、自己完結型である表領域セットのみです。つまり、表領域セット内のデータベース・オブジェクトは、その表領域セット外のデータベース・オブジェクトのいずれにも依存しません。

次に、自己完結した表領域に違反する例を示します。

- 表領域セット内に、そのセットに含まれない表に関する索引が含まれている場合。

  ノート:

  表に対応する索引が表領域セットの外部にある場合は、違反になりません。

- パーティション表の一部が表領域セットに含まれている場合。

  コピーする表領域セットは、パーティション化した表のすべてのパーティションが含まれている状態、またはまったく含まれていない状態にしてください。パーティション表のサブセットをトランスポートするには、パーティションを表に変換する必要があります。

  パーティションの変換の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

- 参照整合性制約がセット境界を越えて別の表を指している場合。

  表領域セットをトランスポートするときには、参照整合性制約を含めるかどうかを選択できます。ただし、そうすることによって、表領域セットの自己完結性に影響を与える場合があります。制約をトランスポートしなければ、その制約はポインタとはみなされません。

- 表領域セット内の表に、そのセットに含まれないLOBを指すLOB列が含まれている場合

- ユーザーAが登録されたXML DBスキーマ(*.xsd)にユーザーBが登録されたグローバル・スキーマをインポートする際、ユーザーAのデフォルト表領域が表領域A、ユーザーBのデフォルト表領域が表領域Bで、表領域Aのみが表領域セットに含まれている場合。

表領域セットが自己完結型かどうかを判別するには、オラクル社が提供するDBMS_TTSパッケージのTRANSPORT_SET_CHECKプロシージャを実行します。このプロシージャを実行するには、EXECUTE_CATALOG_ROLEロール(最初はSYSに付与されている)を付与されている必要があります。

DBMS_TTS.TRANSPORT_SET_CHECKプロシージャを実行するときは、自己完結かどうかを調べるトランスポータブル・セットの表領域のリストを指定します。制約を含むかどうかを指定することもできます。厳密または完全な完結であるかを調べる場合

は、TTS_FULL_CHECKパラメータをTRUEに設定する必要があります。

厳密または完全な完結のチェックは、トランスポータブル・セットから外部への参照のみではなく、外部からトランスポータブル・セットへの参照も捕捉する必要がある場合に実行します。依存オブジェクトがトランスポータブル・セットに完全に含まれているか、またはトランスポータブル・セットの外部にのみ存在することが必要な場合は、表領域のPoint-in-Timeリカバリ(TSPITR)を実行します。

たとえば、表tを含んでいるが、その索引iを含んでいない表領域に対してTSPITRを実行すると、トランスポート後に索引とデータの整合性がなくなるため、これは違反になります。完全完結チェックを実行することにより、トランスポータブル・セットからの依存関係またはトランスポータブル・セットへの依存関係がないことが保証されます。詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』のTSPITRの例を参照してください。

ノート:

デフォルトでは、トランスポータブル表領域は、完全完結しているかどうかではなく自己完結しているかどうかがチェックされます。

次の文を使用して、表領域sales_1およびsales_2が自己完結しているかどうかを、参照整合性制約を考慮して(TRUEを指定して)調べます。

```
EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK('sales_1,sales_2', TRUE);
```

DBMS_TTS.TRANSPORT_SET_CHECKプロシージャを実行した後に、TRANSPORT_SET_VIOLATIONSビューからすべての違反を選択して表示できます。表領域セットが自己完結している場合、このビューは空になります。次の例は、表領域セットの境界を超えている外部キー定数dept_fkと、表領域セットに部分的に含まれているパーティション表jim.salesという、2つの違反がある場合を示しています。

```
SELECT * FROM TRANSPORT_SET_VIOLATIONS;
VIOLATIONS
--------------------------------------------------------------------------------
Constraint DEPT_FK between table JIM.EMP in tablespace SALES_1 and table
JIM.DEPT in tablespace OTHER
Partitioned table JIM.SALES is partially contained in the transportable set
```

sales_1およびsales_2をトランスポータブルにする前に、これらの違反を解決する必要があります。次のタスクで説明するように、整合性制約違反を回避するための選択肢の1つとして、整合性制約をエクスポートしない方法があります。

関連項目:

- DBMS_TTSパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- TSPITRにおけるDBMS_TTSパッケージの使用の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: データベース間での表領域のトランスポート

15.3.3.2 タスク2: トランスポータブル表領域セットの生成

トランスポートする表領域セットが自己完結型であることを確認した後で、トランスポータブル表領域セットを生成します。

トランスポータブル表領域セットを生成するには:

1. SQL*Plusを起動し、管理者として、あるいはALTER TABLESPACEまたはMANAGE TABLESPACEシステム権限を持つユーザーとしてデータベースに接続します。
2. セット内のすべての表領域を読取り専用にします。

```
ALTER TABLESPACE sales_1 READ ONLY;
ALTER TABLESPACE sales_2 READ ONLY;
```

3. DATAPUMP_EXP_FULL_DATABASEロールを持つユーザーとしてデータ・ポンプ・エクスポート・ユーティリティを実行し、トランスポータブル・セット内の表領域を指定します。

```
SQL> HOST
$ expdp user_name dumpfile=expdat.dmp directory=data_pump_dir
        transport_tablespaces=sales_1,sales_2 logfile=tts_export.log
Password: password
```

トランスポータブル・オプションを使用することを指定するTRANSPORT_TABLESPACESを常に指定する必要があります。この例では、次の追加のデータ・ポンプ・パラメータを指定します。

- DUMPFILEパラメータでは、作成する構造情報エクスポート・ダンプ・ファイルの名前をexpdat.dmpと指定します。
- DIRECTORYパラメータでは、オペレーティング・システムまたはOracle Automatic Storage Managementのダンプ・ファイルの場所を示すディレクトリ・オブジェクトを指定します。DIRECTORYオブジェクトはデータ・ポンプを起動する前に作成し、ディレクトリに対するREADおよびWRITEオブジェクト権限をエクスポート・ユーティリティを実行するユーザーに付与する必要があります。

  非CDBで、ディレクトリ・オブジェクトDATA_PUMP_DIRが自動的に作成されます。このディレクトリへの読取りおよび書込みアクセス権がDBAロールに(したがって、ユーザーSYSおよびSYSTEMに)自動的に付与されます。

  ただし、ディレクトリ・オブジェクトDATA_PUMP_DIRは、PDBでは自動的には作成されません。このため、PDBにインポートする場合は、PDBにディレクトリ・オブジェクトを作成し、データ・ポンプを実行するときにそのディレクトリ・オブジェクトを指定します。
- LOGFILEパラメータでは、エクスポート・ユーティリティ用に作成するログ・ファイルを指定します。この例では、ログ・ファイルはダンプ・ファイルと同じディレクトリに作成されますが、ログ・ファイルの格納には他のディレクトリを指定できます。
- トリガーと索引は、デフォルトでエクスポート操作に含まれています。

表領域のトランスポート操作を厳密完結チェック付きで実行するには、次の例に示すように、TRANSPORT_FULL_CHECKパラメータを使用します。

```
expdp use_name dumpfile=expdat.dmp directory=data_pump_dir
      transport_tablespaces=sales_1,sales_2 transport_full_check=y
      logfile=tts_export.log
```

この場合は、データ・ポンプ・エクスポート・ユーティリティによって、トランスポータブル・セット内のオブジェクトとトランスポータブル・セット外のオブジェクトとの間に依存性がないことを検証します。トランスポートする表領域セットが自己完結していない場合、エクスポートは失敗し、トランスポータブル・セットが自己完結していないことがわかります。これらの違反を解決してから、このタスクを再度実行する必要があります。

ノート:

この例では、データ・ポンプ・ユーティリティを使用してエクスポートするのは、表領域のデータ・ディクショナリの構造情報(メタデータ)のみです。実際のデータはアンロードされないため、この操作は大規模な表領域セットの場合でも比較的早く完了します。

4. expdpユーティリティは、次の例に示すように、ダンプ・ファイルおよびデータファイルの名前およびパスをコマンドラインに表示します。これらは、ターゲット・データベースへのトランスポートに必要なファイルです。また、エラーがないかログ・ファイルを確認します。

```
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
  /u01/app/oracle/admin/salesdb/dpdump/expdat.dmp
******************************************************************************
Datafiles required for transportable tablespace SALES_1:
  /u01/app/oracle/oradata/salesdb/sales_101.dbf
Datafiles required for transportable tablespace SALES_2:
  /u01/app/oracle/oradata/salesdb/sales_201.dbf
```

5. データ・ポンプ・エクスポート操作が完了したら、expdpユーティリティを終了してSQL*Plusに戻ります。

```
$ EXIT
```

関連項目:

- CREATE DIRECTORYコマンドの詳細は、『Oracle Database SQL言語リファレンス』を参照してください
- DIRECTORYパラメータを省略する場合のデフォルト・ディレクトリの詳細は、『Oracle Databaseユーティリティ』を参照してください。
- データ・ポンプ・ユーティリティの使用方法は、『Oracle Databaseユーティリティ』を参照してください
- PDBの詳細は、『Oracle Multitenant管理者ガイド』を参照してください

親トピック: データベース間での表領域のトランスポート

15.3.3.3 タスク3: エクスポート・ダンプ・ファイルのトランスポート

ダンプ・ファイルを、DATA_PUMP_DIRディレクトリ・オブジェクトで指し示されているディレクトリ、または他の任意のディレクトリにトランスポートします。新しい場所はターゲット・データベースへアクセス可能であることが必要です。

ターゲット・データベースで、次の問合せを実行してDATA_PUMP_DIRの場所を確認します。

```
SELECT * FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME = 'DATA_PUMP_DIR';
OWNER      DIRECTORY_NAME    DIRECTORY_PATH
---------- ---------------- ----------------------------------
SYS        DATA_PUMP_DIR    C:¥app¥orauser¥admin¥orawin¥dpdump¥
```

親トピック: データベース間での表領域のトランスポート

15.3.3.4 タスク4: 表領域セットのトランスポート

表領域のデータファイルをターゲット・データベースへアクセス可能なディレクトリにトランスポートします。

この例では、次のファイルをソース・データベースからターゲット・データベースに転送します。

- sales_101.dbf
- sales_201.dbf

ソース・プラットフォームとは異なるプラットフォームに表領域セットをトランスポートする場合は、ソースおよびターゲット・プラットフォームの両方でプラットフォーム間の表領域トランスポートがサポートされているかどうかを確認し、それぞれのプラットフォームのendiannessを判別します。両方のプラットフォームのendiannessが同じ場合、変換は必要ありません。同じでない場合は、ソース・データベースまたはターゲット・データベースでデータ変換を実行する必要があります。

sales_1およびsales_2を異なるプラットフォームにトランスポートする場合は、各プラットフォームで次の問合せを実行できます。問合せで行が返される場合、そのプラットフォームではプラットフォーム間の表領域トランスポートがサポートされています。

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

ソース・プラットフォームでの問合せの結果は次のとおりです。

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Solaris[tm] OE (32-bit)          Big
```

ターゲット・プラットフォームからの結果は、次のとおりです。

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Microsoft Windows IA (32-bit)    Little
```

この例では、endian形式が異なることがわかります。したがって、この場合、データベースをトランスポートするには変換が必要です。DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用して、データファイルを転送します。これらのプロシージャを使用すると、データファイルがターゲット・プラットフォームのendian形式に自動的に変換されます。データファイルを、ターゲット・データベースの既存データファイルの場所にトランスポートします。UNIXおよびLinuxプラットフォームでは、この場所は通常、/u01/app/oracle/oradata/dbname/または+DISKGROUP/dbname/datafile/です。または、データファイルを変換するために使用することもできます。

ノート:

- RMAN CONVERT コマンドを使用する場合、UNDO セグメントを含むデータファイルでは、異なる endian 形式間でのデータファイルの変換はサポートされていません。
- 表領域の endianness を変換する必要がない場合は、任意のファイル転送方法を使用してファイルを転送できます。

関連トピック

- プラットフォーム間でのデータの変換
- データファイルを転送するためのガイドライン

親トピック: データベース間での表領域のトランスポート

15.3.3.5 タスク5: (オプション)表領域を読取り/書込みモードに戻す

トランスポートした表領域をソース・データベースで再び読取り/書込みモードにします。

次の文で、sales_1およびsales_2表領域を読取り/書込みモードにします。

```
ALTER TABLESPACE sales_1 READ WRITE;
ALTER TABLESPACE sales_2 READ WRITE;
```

インポート・プロセスが成功したことを先に確認するために、このタスクを延期することができます。

親トピック: データベース間での表領域のトランスポート

15.3.3.6 タスク6: 表領域セットのインポート

トランスポータブル表領域に対する操作を完了するために、表領域セットをインポートします。

表領域セットをインポートするには:

1. DATAPUMP_IMP_FULL_DATABASEロールを持つユーザーとしてデータ・ポンプ・インポート・ユーティリティを実行し、表領域メタデータをインポートします。

```
impdp user_name dumpfile=expdat.dmp directory=data_pump_dir
   transport_datafiles=
   'c:¥app¥orauser¥oradata¥orawin¥sales_101.dbf',
   'c:¥app¥orauser¥oradata¥orawin¥sales_201.dbf'
   remap_schema=sales1:crm1  remap_schema=sales2:crm2
   logfile=tts_import.log
Password: password
```

この例では、次のデータ・ポンプ・パラメータを指定します。

- DUMPFILEパラメータでは、インポートされる表領域のメタデータが含まれるエクスポート・ファイルを指定します。
- DIRECTORYパラメータでは、エクスポート・ダンプ・ファイルの場所を識別するディレクトリ・オブジェクトを指定します。DIRECTORYオブジェクトはデータ・ポンプを実行する前に作成し、ディレクトリに対するREADおよびWRITEオブジェクト権限をインポート・ユーティリティを実行するユーザーに付与する必要があります。CREATE DIRECTORYコマンドの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

  非CDBで、ディレクトリ・オブジェクトDATA_PUMP_DIRがデータベースにより自動的に作成されます。このディレクトリへの読取りおよび書込みアクセス権がDBAロールに(したがって、ユーザーSYSおよびSYSTEMに)自動的に付与されます。

  ただし、ディレクトリ・オブジェクトDATA_PUMP_DIRは、PDBではデータベースにより自動的には作成されません。このため、PDBにインポートする場合は、PDBにディレクトリ・オブジェクトを作成し、データ・ポンプを実行するときにそのディレクトリ・オブジェクトを指定します。

  関連項目:

  - DIRECTORYパラメータを省略する場合のデフォルト・ディレクトリの詳細は、『Oracle Database ユーティリティ』を参照してください。
  - PDBの詳細は、『Oracle Multitenant管理者ガイド』を参照してください

- TRANSPORT_DATAFILESパラメータによって、インポートする表領域が含まれるすべてのデータファイルを識別します。

  多くのデータファイルがある場合は、PARFILEパラメータで指定されたパラメータ・ファイルでTRANSPORT_DATAFILESパラメータを複数回指定できます。

- REMAP_SCHEMAパラメータによって、データベース・オブジェクトの所有権を変更します。REMAP_SCHEMAを指定しない場合、すべてのデータベース・オブジェクト(表や索引など)はソース・データベースと同じユーザー・スキーマ内で作成され、そのユーザーはターゲット・データベース内にすでに存在している必要があります。ユーザーが存在しない場合は、インポート・ユーティリティによってエラーが返されます。この例では、ソース・データ

ベース内でsales1が所有している表領域セット内のオブジェクトは、表領域セットをインポートした後のターゲット・データベース内ではcrm1の所有となります。同様に、ソース・データベース内でsales2が所有しているオブジェクトは、ターゲット・データベース内ではcrm2の所有となります。この例では、ターゲット・データベース内にユーザーsales1およびsales2は存在する必要はありませんが、ユーザーcrm1およびcrm2が存在する必要があります。

Oracle Database 12cリリース2 (12.2)以降では、Recovery Manager (RMAN)のRECOVERコマンドで、表を再マッピングしながら表を異なるスキーマに移動できます。詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

- LOGFILEパラメータでは、インポート・ユーティリティによって書き込まれるログ・ファイルのファイル名を指定します。この例では、ログ・ファイルの書込み先はダンプ・ファイルの読取り元と同じディレクトリですが、ログ・ファイルは別の場所に書き込むことができます。

この文が正常に実行されると、コピーするセット内のすべての表領域は読取り専用モードのままになります。インポート・ログ・ファイルを確認して、エラーが発生しなかったことを確認します。

多数のデータファイルを処理する場合は、データファイル・リストが文の行の制限を超えることがあるため、データファイル名のリストを文の行で指定することは煩雑です。このような場合には、インポート・パラメータ・ファイルを使用できます。たとえば、次のようにしてデータ・ポンプ・インポート・ユーティリティを実行できます。

```
impdp user_name parfile='par.f'
```

par.fパラメータ・ファイルには、次の情報が含まれています。

```
DUMPFILE=expdat.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES=
'C:¥app¥orauser¥oradata¥orawin¥sales_101.dbf',
'C:¥app¥orauser¥oradata¥orawin¥sales_201.dbf'
REMAP_SCHEMA=sales1:crm1  REMAP_SCHEMA=sales2:crm2
LOGFILE=tts_import.log
```

関連項目:

インポート・ユーティリティの使用方法は、『Oracle Databaseユーティリティ』を参照してください。

2. 必要に応じて、ターゲット・データベースで表領域を読取り/書込みモードにします。

親トピック: データベース間での表領域のトランスポート

15.4 データベース間での表、パーティションまたはサブパーティションのトランスポート

データベース間で表、パーティションおよびサブパーティションをトランスポートできます。

- トランスポータブル表の概要
  トランスポータブル表機能を使用すると、表、パーティションまたはサブパーティションのセットを別のOracle Databaseにコピーできます。トランスポータブル表操作では、指定した表、パーティションまたはサブパーティションのメタデータがターゲット・データベースに移動されます。
- トランスポータブル表に関する制限事項
  トランスポータブル表には制限事項があります。

- エクスポート・ダンプ・ファイルを使用した表、パーティションまたはサブパーティションのトランスポート
  エクスポート・ファイルを使用して、表、パーティションまたはサブパーティションをデータベース間でトランスポートできます。
- ネットワーク経由での表、パーティションまたはサブパーティションのトランスポート
  ネットワーク経由で表をトランスポートするには、NETWORK_LINKパラメータを使用してインポートを実行しますが、インポートはデータベース・リンクを使用して実行され、ダンプ・ファイルは関連しません。

親トピック: データのトランスポート

15.4.1 トランスポータブル表の概要

トランスポータブル表機能を使用すると、表、パーティションまたはサブパーティションのセットを別のOracle Databaseにコピーできます。トランスポータブル表操作では、指定した表、パーティションまたはサブパーティションのメタデータがターゲット・データベースに移動されます。

トランスポータブル表操作では、指定した表が使用する表領域が自動的に識別されます。データを移動するには、これらの表領域のデータファイルをターゲット・データベースにコピーします。データ・ポンプ・インポートでは、トランスポータブル表操作に含まれない表、パーティションまたはサブパーティションによって占有されているデータファイルのブロックが自動的に解放されます。また、トランスポータブル表操作に含まれない表の依存オブジェクトによって占有されているブロックも自動的に解放されます。

表、パーティションおよびサブパーティションのトランスポートは、次の方法で実行できます。

- エクスポート・ダンプ・ファイルの使用

  エクスポート時に、TABLESパラメータを指定し、TRANSPORTABLEパラメータをALWAYSに設定します。インポート時には、TRANSPORTABLEパラメータを指定しないでください。データ・ポンプ・インポートによってトランスポータブル表操作が自動的に認識されます。

- ネットワーク経由

  インポート時に、TABLESパラメータを指定し、TRANSPORTABLEパラメータをALWAYSに設定し、NETWORK_LINKパラメータをソース・データベースを識別するように設定します。

親トピック: データベース間での表、パーティションまたはサブパーティションのトランスポート

15.4.2 トランスポータブル表に関する制限事項

トランスポータブル表には制限事項があります。

トランスポータブル表に関する次の制限事項に注意してください。

- トランスポータブル表には、「データのトランスポートに関する一般的な制限事項」で説明されている一般的な制限事項が適用されます。
- 同じスキーマに同じ名前の表を含むターゲット・データベースに表をトランスポートすることはできません。ただし、REMAP_TABLEインポート・パラメータを使用して、データを異なる表にインポートできます。または、トランスポート操作を実行する前に、トランスポート対象の表またはターゲット表のいずれかの名前を変更できます。

  Oracle Database 12cリリース2 (12.2)以降では、Recovery Manager (RMAN)のRECOVERコマンドで、表を再マッピングしながら表を異なるスキーマに移動できます。詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- タイム・ゾーン・ファイルのバージョンが異なるプラットフォーム間では、TIMESTAMP WITH TIMEZONE (TSTZ)データを含む表をトランスポートできません。

詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: データベース間での表、パーティションまたはサブパーティションのトランスポート

15.4.3 エクスポート・ダンプ・ファイルを使用した表、パーティションまたはサブパーティションのトランスポート

エクスポート・ファイルを使用して、データベース間で表、パーティションまたはサブパーティションをトランスポートできます。

次のタスクのリストでは、エクスポート・ダンプ・ファイルを使用したデータベース間での表のトランスポート処理の概要を示します。各タスクの詳細は、後続の例で示します。

1. 表、パーティションまたはサブパーティションのセットを選択します。

   パーティションをトランスポートする場合、トランスポータブル表操作で指定できるのは1つの表のパーティションのみであり、同じ操作で他の表をトランスポートすることはできません。また、トランスポータブル表操作で表のパーティションのサブセットのみをエクスポートすると、インポート時にそれぞれのパーティションが非パーティション表になります。

2. ソース・データベースで、表、パーティションまたはサブパーティションのデータファイルに関連付けられた表領域を読取り専用モードにします。

   表の表領域を表示するには、DBA_TABLESビューを問い合せます。表領域のデータファイルを表示するには、DBA_DATA_FILESビューを問い合せます。

3. データ・ポンプ・エクスポートを実行します。

4. エクスポート・ダンプ・ファイルをトランスポートします。

   エクスポート・ダンプ・ファイルをターゲット・データベースへアクセス可能な場所にコピーします。

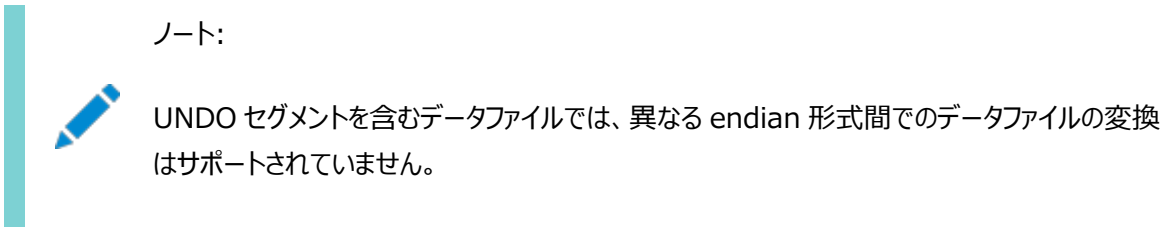
5. 表、パーティションまたはサブパーティションのデータファイルをトランスポートします。

   データファイルをターゲット・データベースへアクセス可能な場所にコピーします。

   ソース・プラットフォームとターゲット・プラットフォームが異なる場合は、V$TRANSPORTABLE_PLATFORMビューに対して問合せを実行して、各プラットフォームのendian形式をチェックできます(「プラットフォーム間でのデータのトランスポート」を参照)。

   ソース・プラットフォームのendian形式がターゲット・プラットフォームのendian形式と異なる場合は、次のいずれかの方法を使用してデータファイルを変換します。

   - DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用して、データファイルを転送します。これらのプロシージャを使用すると、データファイルがターゲット・プラットフォームのendian形式に自動的に変換されます。
   - RMAN CONVERTコマンドを使用して、データファイルをターゲット・プラットフォームのendian形式に変換します。



ノート:

UNDO セグメントを含むデータファイルでは、異なる endian 形式間でのデータファイルの変換はサポートされていません。

詳細は、「プラットフォーム間でのデータの変換」を参照してください。

6. (オプション)ソース・データベースで表領域を読取り/書込みモードに戻します。
7. ターゲット・データベースで、インポートを実行します。

   データ・ポンプ・ユーティリティを起動して、表のメタデータをインポートします。

例

データ・ポンプ・ダンプ・ファイルを使用して、表、パーティションおよびサブパーティションをトランスポートするタスクについては、次のパーティションがsh.sales_prt表に存在することを想定した例で、詳細に説明します。

- sales_q1_2000
- sales_q2_2000
- sales_q3_2000
- sales_q4_2000

この例では、これらのパーティションの2つをターゲット・データベースにトランスポートします。

次のSQL文では、shスキーマのsales_prt表とそのパーティション、および表の表領域とデータファイルが作成されます。また、この文では、shサンプル・スキーマのデータを使用してパーティションにデータが挿入されます。

```
CREATE TABLESPACE sales_prt_tbs
   DATAFILE 'sales_prt.dbf' SIZE 20M
   ONLINE;

CREATE TABLE sh.sales_prt
    (prod_id        NUMBER(6),
     cust_id        NUMBER,
     time_id        DATE,
     channel_id     CHAR(1),
     promo_id       NUMBER(6),
     quantity_sold  NUMBER(3),
     amount_sold    NUMBER(10,2))
        PARTITION BY RANGE (time_id)
         (PARTITION SALES_Q1_2000 VALUES LESS THAN
                        (TO_DATE('01-APR-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE =
American')),
          PARTITION SALES_Q2_2000 VALUES LESS THAN
                       (TO_DATE('01-JUL-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE =
American')),
          PARTITION SALES_Q3_2000 VALUES LESS THAN
                        (TO_DATE('01-OCT-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE =
American')),
          PARTITION SALES_Q4_2000 VALUES LESS THAN
                        (TO_DATE('01-JAN-2001','DD-MON-YYYY','NLS_DATE_LANGUAGE =
American')))
 TABLESPACE sales_prt_tbs;

INSERT INTO sh.sales_prt PARTITION(sales_q1_2000)
  SELECT * FROM sh.sales PARTITION(sales_q1_2000);
INSERT INTO sh.sales_prt PARTITION(sales_q2_2000)
  SELECT * FROM sh.sales PARTITION(sales_q2_2000);
INSERT INTO sh.sales_prt PARTITION(sales_q3_2000)
  SELECT * FROM sh.sales PARTITION(sales_q3_2000);

INSERT INTO sh.sales_prt PARTITION(sales_q4_2000)
  SELECT * FROM sh.sales PARTITION(sales_q4_2000);
COMMIT;
```

この例では、さらに次のことを想定しています。

- ソース・データベースの名前はsourcedbです。
- ソース・データベースとターゲット・データベースは、endiannessが同一の同じプラットフォームで実行されている必要があります。プラットフォームのendiannessをチェックするには、次の問合せを実行します。

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

- sales_q1_2000パーティションとsales_q2_2000パーティションのみがターゲット・データベースにトランスポートされます。他の2つのパーティションはトランスポートされません。

エクスポート・ダンプ・ファイルを使用してパーティションをトランスポートするには、次のタスクを実行します。

タスク 1　エクスポート・ダンプ・ファイルの作成

次のステップを実行して、エクスポート・ダンプ・ファイルを生成します。

1. SQL*Plus を起動し、管理者として、あるいは ALTER TABLESPACE または MANAGE TABLESPACE システム権限を持つユーザーとしてソース・データベースに接続します。

   手順は、「SQL*Plus を使用したデータベースへの接続」を参照してください。

2. トランスポートする表を含むすべての表領域を読取り専用にします。

```
ALTER TABLESPACE sales_prt_tbs READ ONLY;
```

3. DATAPUMP_EXP_FULL_DATABASE ロールを持つユーザーとしてデータ・ポンプ・エクスポート・ユーティリティを起動し、トランスポータブル表オプションを指定します。

```
SQL> HOST
expdp user_name dumpfile=sales_prt.dmp  directory=data_pump_dir
      tables=sh.sales_prt:sales_q1_2000,sh.sales_prt:sales_q2_2000
      transportable=always logfile=exp.log
Password: password
```

   トランスポータブル・オプションを使用することを指定する TRANSPORTABLE=ALWAYS を常に指定する必要があります。

   この例では、次の追加のデータ・ポンプ・パラメータを指定します。

   - DUMPFILE パラメータでは、作成する構造情報エクスポート・ダンプ・ファイルの名前を sales_prt.dmp と指定します。
   - DIRECTORY パラメータでは、オペレーティング・システムまたは Oracle Automatic Storage Management のダンプ・ファイルの場所を示すディレクトリ・オブジェクトを指定します。DIRECTORY オブジェクトはデータ・ポンプを起動する前に作成し、ディレクトリに対する READ および WRITE オブジェクト権限をエクスポート・ユーティリティを実行するユーザーに付与する必要があります。CREATE DIRECTORY コマンドの詳細は、『Oracle Database SQL 言語リファレン

ス』を参照してください。

非 CDB で、ディレクトリ・オブジェクト DATA_PUMP_DIR が自動的に作成されます。このディレクトリへの読取りおよび書込みアクセス権が DBA ロールに(したがって、ユーザー SYS および SYSTEM に)自動的に付与されます。

ただし、ディレクトリ・オブジェクト DATA_PUMP_DIR は、PDB では自動的には作成されません。このため、PDB にインポートする場合は、PDB にディレクトリ・オブジェクトを作成し、データ・ポンプを実行するときにそのディレクトリ・オブジェクトを指定します。

関連項目:

- DIRECTORY パラメータを省略する場合のデフォルト・ディレクトリの詳細は、『Oracle Database ユーティリティ』を参照してください。
- PDB の詳細は、『Oracle Multitenant 管理者ガイド』を参照してください
- TABLES パラメータでは、エクスポートする表、パーティションまたはサブパーティションを指定します。
- LOGFILE パラメータでは、エクスポート・ユーティリティによって書き込まれるログ・ファイルのファイル名を指定します。この例では、ログ・ファイルの書込み先はダンプ・ファイルと同じディレクトリですが、ログ・ファイルは別の場所に書き込むことができます。

4. ログ・ファイルで予期しないエラーを確認し、ターゲット・データベースにトランスポートする必要があるダンプ・ファイルとデータファイルをノートにとります。expdp により、これらのファイルの名前とパスが次のようなメッセージに出力されます。

```
Processing object type TABLE_EXPORT/TABLE/PLUGTS_BLK
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/END_PLUGTS_BLK
Master table "SYSTEM"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
****
Dump file set for SYSTEM.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/rdbms/log/sales_prt.dmp
******************************************************************************
****
Datafiles required for transportable tablespace SALES_PRT_TBS:
  /u01/app/oracle/oradata/sourcedb/sales_prt.dbf
Job "SYSTEM"."SYS_EXPORT_TABLE_01" successfully completed at 11:32:13
```

5. 完了した後、終了して SQL*Plus に戻ります。

```
$ exit
```

関連項目:

データ・ポンプ・ユーティリティの使用方法は、『Oracle Database ユーティリティ』を参照してください

タスク 2　エクスポート・ダンプ・ファイルのトランスポート

ダンプ・ファイルを、ターゲット・データベース上の DATA_PUMP_DIR ディレクトリ・オブジェクトで指し示されているディレクトリ、または他の任意のディレクトリにトランスポートします。新しい場所はターゲット・データベースへアクセス可能であることが必要です。

この例では、sales_prt.dmp ダンプ・ファイルをソース・データベースからターゲット・データベースに転送します。

ターゲット・データベースで、次の問合せを実行して DATA_PUMP_DIR の場所を確認します。

```
SELECT * FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME = 'DATA_PUMP_DIR';
OWNER      DIRECTORY_NAME   DIRECTORY_PATH
---------- ---------------- ----------------------------------
SYS        DATA_PUMP_DIR    /u01/app/oracle/rdbms/log/
```

タスク 3　表のデータファイルのトランスポート

トランスポートする表を含む表領域のデータファイルを、ターゲット・データベースへアクセス可能な場所にトランスポートします。

通常は、データファイルを、ターゲット・データベースの既存データファイルの場所にトランスポートします。UNIX および Linux プラットフォームでは、この場所は通常、/u01/app/oracle/oradata/dbname/または+DISKGROUP/dbname/datafile/です。

この例では、sales_prt.dbf データファイルをソース・データベースからターゲット・データベースに転送します。

関連項目:

「データファイルを転送するためのガイドライン」

タスク 4　(オプション)表領域を読取り/書込みモードに戻す

次のように、トランスポートする表を含む表領域をソース・データベースで再び読取り/書込みモードにします。

```
ALTER TABLESPACE sales_prt_tbs READ WRITE;
```

インポート・プロセスが成功したことを先に確認するために、このタスクを延期することができます。

タスク 5　ターゲット・データベースでのパーティションのインポート

ターゲット・データベースで、DATAPUMP_IMP_FULL_DATABASE ロールを持つユーザーとしてデータ・ポンプ・インポート・ユーティリティを起動し、トランスポータブル表オプションを指定します。

```
impdp user_name dumpfile=sales_prt.dmp directory=data_pump_dir
   transport_datafiles='/u01/app/oracle/oradata/targetdb/sales_prt.dbf'
   tables=sh.sales_prt:sales_q1_2000,sh.sales_prt:sales_q2_2000
   logfile=imp.log
Password: password
```

この例では、次のデータ・ポンプ・パラメータを指定します。

- DUMPFILE パラメータでは、インポートされるデータのメタデータが含まれるエクスポート・ファイルを指定します。
- DIRECTORY パラメータでは、エクスポート・ダンプ・ファイルの場所を識別するディレクトリ・オブジェクトを指定します。DIRECTORY オブジェクトはデータ・ポンプを起動する前に作成し、ディレクトリに対する READ および WRITE オブジェクト権限をインポート・ユーティリティを実行するユーザーに付与する必要があります。CREATE DIRECTORY コマンドの詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

  非 CDB で、ディレクトリ・オブジェクト DATA_PUMP_DIR が自動的に作成されます。このディレクトリへの読取りおよび書込みアクセス権が DBA ロールに(したがって、ユーザーSYS および SYSTEM に)自動的に付与されます。

  ただし、ディレクトリ・オブジェクト DATA_PUMP_DIR は、PDB では自動的には作成されません。このため、PDB にインポートする場合は、PDB にディレクトリ・オブジェクトを作成し、データ・ポンプを実行するときにそのディレクトリ・オブジェクトを指定します。

  関連項目:

- DIRECTORY パラメータを省略する場合のデフォルト・ディレクトリの詳細は、『Oracle Database ユーティリティ』を参照してください。
- PDB の詳細は、『Oracle Multitenant 管理者ガイド』を参照してください
- TRANSPORT_DATAFILES パラメータによって、インポートするすべてのデータファイルを識別しま

す。

多くのデータファイルがある場合は、PARFILE パラメータで指定されたパラメータ・ファイルで TRANSPORT_DATAFILES パラメータを複数回指定できます。

- TABLES パラメータでは、インポートする表、パーティションまたはサブパーティションを指定します。
- LOGFILE パラメータでは、インポート・ユーティリティによって書き込まれるログ・ファイルのファイル名を指定します。この例では、ログ・ファイルの書込み先はダンプ・ファイルの読取り元と同じディレクトリですが、ログ・ファイルは別の場所に書き込むことができます。

この文が正常に実行された後、インポート・ログ・ファイルをチェックして、予期しないエラーが発生していないことを確認します。

多数のデータファイルを扱う場合、データファイル名のリストを文の行で指定することは煩雑です。また、文の行制限を超える場合もあります。このような場合には、インポート・パラメータ・ファイルを使用できます。たとえば、次のようにしてデータ・ポンプ・インポート・ユーティリティを起動できます。

```
impdp user_name parfile='par.f'
```

たとえば、par.f には次の行が含まれる場合があります。

```
DUMPFILE=sales_prt.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES='/u01/app/oracle/oradata/targetdb/sales_prt.dbf'
TABLES=sh.sales_prt:sales_q1_2000,sh.sales_prt:sales_q2_2000
LOGFILE=imp.log
```

ノート:

- この例ではパーティションのサブセットをトランスポートするため、パーティションはターゲット・データベースで個別の表としてインポートされます。
- インポート中、メタデータのロードのために表領域を一時的に読取り/書込みにすることがあります。インポート中にデータに対するユーザー変更が行われないことを確認してください。インポートが正常に完了したとき、すべてのユーザー定義表領域は読取り/書込みになります。
- ネットワーク・データベース・インポートを実行する場合は、TRANSPORTABLE パラメータを always に設定する必要があります。

関連項目:

インポート・ユーティリティの使用方法は、『Oracle Database ユーティリティ』を参照してください。

15.4.4 ネットワーク経由での表、パーティションまたはサブパーティションのトランスポート

ネットワーク経由で表をトランスポートするには、NETWORK_LINKパラメータを使用してインポートを実行し、インポートはデータベース・リンクを使用して実行され、ダンプ・ファイルは関連しません。

次のタスクのリストでは、ネットワーク経由でのデータベース間の表、パーティションおよびサブパーティションのトランスポート処理の概要を示します。各タスクの詳細は、後続の例で示します。

1. 表、パーティションまたはサブパーティションのセットを選択します。

   パーティションをトランスポートする場合、トランスポータブル表操作で指定できるのは1つの表のパーティションのみであり、同じ操作で他の表をトランスポートすることはできません。また、トランスポータブル表操作で表のパーティションのサブセットのみをエクスポートすると、インポート時にそれぞれのパーティションが非パーティション表になります。

2. ソース・データベースで、表、パーティションまたはサブパーティションのデータファイルに関連付けられた表領域を読取り専用モードにします。

   表の表領域を表示するには、DBA_TABLESビューを問い合せます。表領域のデータファイルを表示するには、DBA_DATA_FILESビューを問い合せます。

3. 表、パーティションまたはサブパーティションのデータファイルをトランスポートします。

   データファイルをターゲット・データベースへアクセス可能な場所にコピーします。

   ソース・プラットフォームとターゲット・プラットフォームが異なる場合は、V$TRANSPORTABLE_PLATFORMビューに対して問合せを実行して、各プラットフォームのendian形式をチェックできます(「プラットフォーム間でのデータのトランスポート」を参照)。

   ソース・プラットフォームのendian形式がターゲット・プラットフォームのendian形式と異なる場合は、次のいずれかの方法を使用してデータファイルを変換します。

   - DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用して、データファイルを転送します。これらのプロシージャを使用すると、データファイルがターゲット・プラットフォームのendian形式に自動的に変換されます。
   - RMAN CONVERTコマンドを使用して、データファイルをターゲット・プラットフォームのendian形式に変換します。

   > ノート:
   >
   > UNDO セグメントを含むデータファイルでは、異なる endian 形式間でのデータファイルの変換はサポートされていません。

   詳細は、「プラットフォーム間でのデータの変換」を参照してください。

4. ターゲット・データベースで、インポートを実行します。

   データ・ポンプ・ユーティリティを起動して、表のメタデータをインポートします。

5. (オプション)ソース・データベースで表領域を読取り/書込みモードに戻します。

例

ネットワーク経由で表をトランスポートするタスクについては、表がソース・データベースに存在することを想定した次の例で詳細に説明します。

表	表領域	データファイル
hr.emp_ttbs	emp_tsp	/u01/app/oracle/oradata/sourcedb/emp.dbf
oe.orders_ttbs	orders_tsp	/u01/app/oracle/oradata/sourcedb/orders.dbf

この例では、これらの表をターゲット・データベースにトランスポートします。この例を実行するには、これらの表はソース・データベースに存在する必要があります。

次のSQL文では、hrスキーマの表およびその表の表領域とデータファイルが作成されます。また、この文では、hrおよびoeサンプル・スキーマのデータを使用して表にデータが挿入されます。

```
CREATE TABLESPACE emp_tsp
   DATAFILE 'emp.dbf' SIZE 1M
   ONLINE;
CREATE TABLE hr.emp_ttbs(
   employee_id    NUMBER(6),
   first_name     VARCHAR2(20),
   last_name      VARCHAR2(25),
   email          VARCHAR2(25),
   phone_number   VARCHAR2(20),
   hire_date      DATE,
   job_id         VARCHAR2(10),
   salary         NUMBER(8,2),
   commission_pct NUMBER(2,2),
   manager_id     NUMBER(6),
   department_id  NUMBER(4))
 TABLESPACE emp_tsp;
INSERT INTO hr.emp_ttbs SELECT * FROM hr.employees;
CREATE TABLESPACE orders_tsp
   DATAFILE 'orders.dbf' SIZE 1M
   ONLINE;
CREATE TABLE oe.orders_ttbs(
   order_id      NUMBER(12),
   order_date    TIMESTAMP WITH LOCAL TIME ZONE,
   order_mode    VARCHAR2(8),
   customer_id   NUMBER(6),
   order_status  NUMBER(2),
   order_total   NUMBER(8,2),
   sales_rep_id  NUMBER(6),
   promotion_id  NUMBER(6))
 TABLESPACE orders_tsp;
INSERT INTO oe.orders_ttbs SELECT * FROM oe.orders;
COMMIT;
```

この例では、さらに次のことを想定しています。

- ソース・データベースの名前はsourcedbです。
- ソース・データベースとターゲット・データベースは、endiannessが同一の同じプラットフォームで実行されている必要があります。プラットフォームのendiannessをチェックするには、次の問合せを実行します。

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

ネットワーク経由で表をトランスポートするには、次のタスクを実行します。

タスク 1　ターゲット・データベースからソース・データベースへのデータベース・リンクの作成

次のステップを実行して、ターゲット・データベースからソース・データベースへデータベース・リンクを作成します。

1. ソース・データベースとターゲット・データベースの間にネットワーク接続が構成されていることを確認します。

   詳細は、『Oracle Database Net Services 管理者ガイド』を参照してください。

2. SQL*Plus を起動し、データ・ポンプ・インポートによってデータをトランスポートする管理者としてターゲット・データベースに接続します。データをトランスポートするには、このユーザーに DATAPUMP_IMP_FULL_DATABASE ロールが必要です。

   手順は、「SQL*Plus を使用したデータベースへの接続」を参照してください。

3. データベース・リンクを作成します。

   ```
   CREATE PUBLIC DATABASE LINK sourcedb USING 'sourcedb';
   ```

   USING 句でソース・データベースのサービス名を指定します。

   インポート操作中、データベース・リンクをソース・データベースで DATAPUMP_EXP_FULL_DATABASE ロールを持つユーザーに接続する必要があります。ソース・データベースのユーザーは、SYSDBA 管理権限を持つユーザーにすることはできません。

関連項目:

- 「データベース・リンクの作成」
- Oracle Database SQL 言語リファレンス

タスク 2　表を含む表領域を読取り専用にする

ソース・データベースで、次のステップを実行します。

1. SQL*Plus を起動し、管理者として、あるいは ALTER TABLESPACE または MANAGE TABLESPACE システム権限を持つユーザーとしてソース・データベースに接続します。

   手順は、「SQL*Plus を使用したデータベースへの接続」を参照してください。

2. トランスポートするデータを含むすべての表領域を読取り専用にします。

```
ALTER TABLESPACE emp_tsp READ ONLY;
ALTER TABLESPACE orders_tsp READ ONLY;
```

タスク３　表のデータファイルのトランスポート

トランスポートする表を含む表領域のデータファイルを、ターゲット・データベースへアクセス可能な場所にトランスポートします。

通常は、データファイルを、ターゲット・データベースの既存データファイルの場所にトランスポートします。UNIX および Linux プラットフォームでは、この場所は通常、/u01/app/oracle/oradata/dbname/または+DISKGROUP/dbname/datafile/です。

この例では、`emp.dbf` および `orders.dbf` データファイルをソース・データベースからターゲット・データベースに転送します。

関連項目:

「データファイルを転送するためのガイドライン」

タスク４　ターゲット・データベースでのデータベースのインポート

DATAPUMP_IMP_FULL_DATABASE ロールを持つユーザーとしてデータ・ポンプ・インポート・ユーティリティを起動し、フル・トランスポータブル・エクスポート/インポート・オプションを指定します。

```
impdp user_name network_link=sourcedb transportable=always
   transport_datafiles=
      '/u01/app/oracle/oradata/targetdb/emp.dbf'
      '/u01/app/oracle/oradata/targetdb/orders.dbf'
   tables=hr.emp_ttbs,oe.orders_ttbs
   logfile=import.log
Password: password
```

この例では、次のデータ・ポンプ・パラメータを指定します。

- `NETWORK_LINK` パラメータでは、ネットワーク・インポートに使用されるソース・データベースへのデータベース・リンクを指定します。
- `TRANSPORTABLE` パラメータでは、インポートでトランスポータブル・オプションを使用することを指定します。
- `TRANSPORT_DATAFILES` パラメータによって、インポートするすべてのデータファイルを識別します。

多くのデータファイルがある場合は、PARFILE パラメータで指定されたパラメータ・ファイルで TRANSPORT_DATAFILES パラメータを複数回指定できます。

- TABLES パラメータでは、インポートする表を指定します。
- LOGFILE パラメータでは、インポート・ユーティリティによって書き込まれるログ・ファイルのファイル名を指定します。

この文が正常に実行された後、インポート・ログ・ファイルをチェックして、予期しないエラーが発生していないことを確認します。

多数のデータファイルを扱う場合、データファイル名のリストを文の行で指定することは煩雑です。また、文の行制限を超える場合もあります。このような場合には、インポート・パラメータ・ファイルを使用できます。たとえば、次のようにしてデータ・ポンプ・インポート・ユーティリティを起動できます。

```
impdp user_name parfile='par.f'
```

たとえば、par.f には次の行が含まれる場合があります。

```
NETWORK_LINK=sourcedb
TRANSPORTABLE=always
TRANSPORT_DATAFILES=
      '/u01/app/oracle/oradata/targetdb/emp.dbf'
      '/u01/app/oracle/oradata/targetdb/orders.dbf'
TABLES=hr.emp_ttbs,oe.orders_ttbs
LOGFILE=import.log
```

ノート:

インポート中に、メタデータのロードのためにユーザー定義表領域を一時的に読取り/書込みにすることがあります。インポート中にデータに対するユーザー変更が行われないことを確認してください。インポートが正常に完了したとき、すべてのユーザー定義表領域は読取り/書込みになります。

関連項目:

インポート・ユーティリティの使用方法は、『Oracle Database ユーティリティ』を参照してください。

タスク 5　(オプション)表領域を読取り/書込みモードに戻す

次のように、トランスポートする表を含む表をソース・データベースで再び読取り/書込みにします。

```
ALTER TABLESPACE emp_tsp READ WRITE;
ALTER TABLESPACE orders_tsp READ WRITE;
```

親トピック: データベース間での表、パーティションまたはサブパーティションのトランスポート

15.5 プラットフォーム間でのデータの変換

トランスポータブル操作を実行するときに、ソース・プラットフォームとターゲット・プラットフォームでendiannessが異なる場合は、トランスポートするデータをターゲット・プラットフォームの形式に変換する必要があります。ソース・プラットフォームとターゲット・プラットフォームでendiannessが同じ場合は、データ変換は必要ありません。データを変換するには、DBMS_FILE_TRANSFERパッケージまたはRMAN CONVERTコマンドを使用できます。

ノート:

これらの項には記載されていない制限事項が適用されることがあります。詳細は、次のドキュメントを参照してください。

- プラットフォームの endianness をチェックする方法の詳細は、「プラットフォーム間でのデータのトランスポート」を参照してください
- DBMS_FILE_TRANSFER パッケージに関する制限事項の詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。
- RMAN CONVERT コマンドに関する制限事項の詳細は、『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

- DBMS_FILE_TRANSFERパッケージを使用したプラットフォーム間でのデータの変換
  DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用すると、データファイルの転送中にプラットフォーム間でデータを変換できます。
- RMANを使用したプラットフォーム間でのデータの変換
  RMAN CONVERTコマンドを使用してデータを変換する場合は、データ・ポンプ・エクスポートを実行した後にソース・プラットフォームでデータを変換するか、またはデータ・ポンプ・インポートを実行する前にターゲット・プラットフォームでデータを変換できます。いずれの場合も、データファイルをソース・システムからターゲット・システムに転送する必要があります。

親トピック: データのトランスポート

15.5.1 DBMS_FILE_TRANSFERパッケージを使用したプラットフォーム間でのデータの変換

DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用すると、データファイルの転送中にプラットフォーム間でデータを変換できます。

これらのプロシージャのいずれかを使用して、ソース・プラットフォームとターゲット・プラットフォーム間でデータファイルを移動する場合、各データファイル内の各ブロックがターゲット・プラットフォームのendiannessに変換されます。

この項では、例を使用して、データファイルを異なるプラットフォームに変換するためにDBMS_FILE_TRANSFERパッケージを使用する方法を示します。この例では、次のことを想定しています。

- GET_FILEプロシージャでデータファイルを転送します。
- mytable.342.123456789データファイルを異なるプラットフォームに転送します。

- ソース・プラットフォームのendiannessはターゲット・プラットフォームのendiannessと異なります。
- ソース・データベースのグローバル名は`dbsa.example.com`です。
- ソース・データベースとターゲット・データベースの両方でOracle Automatic Storage Management(Oracle ASM)が使用されています。

ノート:

DBMS_FILE_TRANSFER パッケージを使用して、endianness が同じプラットフォーム間でデータファイルを転送することもできます。

`GET_FILE`プロシージャを使用して転送することによってデータファイルを変換するには、次のステップを実行します。

1. SQL*Plusを使用して、ディレクトリ・オブジェクトを作成できる管理ユーザーとしてソース・データベースに接続します。
2. ターゲット・データベースに転送するデータファイルを格納するディレクトリ・オブジェクトを作成します。

   たとえば、`+data/dbsa/datafile`ディレクトリに対して`sales_dir_source`というディレクトリ・オブジェクトを作成するには、次のSQL文を実行します。

   ```
   CREATE OR REPLACE DIRECTORY sales_dir_source
      AS '+data/dbsa/datafile';
   ```

   ディレクトリ・オブジェクトの作成時に、指定したファイル・システム・ディレクトリが存在している必要があります。

3. SQL*Plusを使用して、データベース・リンクの作成、ディレクトリ・オブジェクトの作成および`DBMS_FILE_TRANSFER`パッケージのプロシージャの実行ができる管理ユーザーとしてターゲット・データベースに接続します。
4. ターゲット・データベースからソース・データベースへデータベース・リンクを作成します。

   ソース・データベースの接続されるユーザーには、ステップ2で作成したディレクトリ・オブジェクトについての読取り権限が必要です。

5. ソース・データベースから転送するデータファイルを格納するディレクトリ・オブジェクトを作成します。

   `DBMS_FILE_TRANSFER`パッケージのプロシージャを実行するローカル・データベースのユーザーには、ディレクトリ・オブジェクトについての書込み権限が必要です。

   たとえば、`+data/dbsb/datafile`ディレクトリに対して`sales_dir_target`というディレクトリ・オブジェクトを作成するには、次のSQL文を実行します。

   ```
   CREATE OR REPLACE DIRECTORY sales_dir_target
     AS '+data/dbsb/datafile';
   ```

6. `DBMS_FILE_TRANSFER`パッケージの`GET_FILE`プロシージャを実行して、データファイルを転送します。

   たとえば、次のプロシージャを実行し、ステップ4で作成したデータベース・リンクを使用して、`mytable.342.123456789`データファイルをソース・データベースからターゲット・データベースに転送します。

   ```
   BEGIN
     DBMS_FILE_TRANSFER.GET_FILE(
       source_directory_object      => 'sales_dir_source',
       source_file_name             => 'mytable.342.123456789',
       source_database              => 'dbsa.example.com',
       destination_directory_object => 'sales_dir_target',
       destination_file_name        => 'mytable');
   END;
   /
   ```

ノート:

この例では、宛先データファイル名は mytable です。Oracle ASM では、GET_FILE プロシージャの destination_file_name パラメータで完全修飾ファイル名形式を使用できません。

関連トピック

- SQL*Plusを使用したデータベースへの接続について
- データベース・リンクの作成

関連項目:

- DBMS_FILE_TRANSFERパッケージの使用方法の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- ASMの完全修飾ファイル名形式の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。
- データベース・リンクの作成方法は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: プラットフォーム間でのデータの変換

15.5.2 RMANを使用したプラットフォーム間でのデータの変換

RMAN CONVERTコマンドを使用してデータを変換する場合は、データ・ポンプ・エクスポートを実行した後にソース・プラットフォームでデータを変換するか、またはデータ・ポンプ・インポートを実行する前にターゲット・プラットフォームでデータを変換できます。いずれの場合も、データファイルをソース・システムからターゲット・システムに転送する必要があります。

次のRMAN CONVERTコマンドを使用してデータを変換できます。

- CONVERT DATAFILE
- CONVERT TABLESPACE
- CONVERT DATABASE

ノート:

- RMAN CONVERT コマンドにはデータ型の制約が適用されます。
- RMAN CONVERT コマンドでは、UNDO セグメントを含むデータファイルについて、異なる endian 形式間でのデータファイルの変換はサポートされていません。

- エクスポート後のソース・システムでの表領域の変換
  例を使用して、表領域を異なるプラットフォームに変換するために、RMAN CONVERT TABLESPACEコマンドを使用する方法を示します。
- インポート前のターゲット・システムでのデータファイルの変換
  例を使用して、データファイルを異なるプラットフォームに変換するために、RMAN CONVERT DATAFILEコマンドを使用する方法を示します。

関連項目:

- Oracle Databaseバックアップおよびリカバリ・リファレンス
- Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド

親トピック: プラットフォーム間でのデータの変換

15.5.2.1 エクスポート後のソース・システムでの表領域の変換

例を使用して、表領域を異なるプラットフォームに変換するために、RMAN CONVERT TABLESPACEコマンドを使用する方法を示します。

この例では、次のことを想定しています。

- sales_1およびsales_2表領域を異なるプラットフォームにトランスポートします。
- ソース・プラットフォームのendiannessはターゲット・プラットフォームのendiannessと異なります。
- 表領域セットをターゲット・システムにトランスポートする前に、ソース・システムでデータを変換します。
- ソース・データベースでデータ・ポンプ・エクスポートを完了しています。

ソース・システムで表領域を変換するには、次のステップを実行します。

1. コマンド・プロンプトで、RMANを起動してソース・データベースに接続します。

```
$ RMAN TARGET /
Recovery Manager: Release 12.1.0.1.0 - Production
Copyright (c) 1982, 2012, Oracle and/or its affiliates.  All rights reserved.
connected to target database: salesdb (DBID=3295731590)
```

2. RMAN CONVERT TABLESPACEコマンドを使用してデータファイルを変換し、ソース・プラットフォーム上の一時的な場所に格納します。

   この例では、一時的な場所はディレクトリ/tmpと想定し、すでに作成されているとします。変換されたデータファイルの名前は、システムによって割り当てられます。

```
RMAN> CONVERT TABLESPACE sales_1,sales_2
2> TO PLATFORM 'Microsoft Windows IA (32-bit)'
3> FORMAT '/tmp/%U';
Starting conversion at source at 30-SEP-08
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile conversion
input datafile file number=00007
name=/u01/app/oracle/oradata/salesdb/sales_101.dbf
converted datafile=/tmp/data_D-SALESDB_I-1192614013_TS-SALES_1_FNO-7_03jru08s
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:45
channel ORA_DISK_1: starting datafile conversion
input datafile file number=00008
name=/u01/app/oracle/oradata/salesdb/sales_201.dbf
converted datafile=/tmp/data_D-SALESDB_I-1192614013_TS-SALES_2_FNO-8_04jru0aa
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:25
Finished conversion at source at 30-SEP-08
```

関連項目:

RMANのCONVERTコマンドの詳細は、『Oracle Databaseバックアップおよびリカバリ・リファレンス』を参照してください。

3. Recovery Managerを終了します。

```
RMAN> exit
Recovery Manager complete.
```

4. データファイルをターゲット・システムに転送します。

関連トピック

- データファイルを転送するためのガイドライン

親トピック: RMANを使用したプラットフォーム間でのデータの変換

15.5.2.2 インポート前のターゲット・システムでのデータファイルの変換

例を使用して、データファイルを異なるプラットフォームに変換するために、RMAN CONVERT DATAFILEコマンドを使用する方法を示します。

変換時、データファイルは、表領域名ではなく、ファイル名で指定します。表領域のメタデータがインポートされるまで、ターゲット・インスタンスでは対象の表領域名を認識できません。

この例では、次のことを想定しています。

- トランスポートする表領域のデータファイルはまだ変換していません。

  DBMS_FILE_TRANSFERパッケージを使用してデータファイルをターゲット・システムに転送した場合、データファイルはファイル転送中に自動的に変換されています。「DBMS_FILE_TRANSFERパッケージを使用したプラットフォーム間でのデータの変換」を参照してください。

- 次のデータファイルを異なるプラットフォームにトランスポートします。
  - C:¥Temp¥sales_101.dbf
  - C:¥Temp¥sales_201.dbf
- ソース・プラットフォームのendiannessはターゲット・プラットフォームのendiannessと異なります。
- データ・ポンプ・インポートを実行する前に、ターゲット・システムでデータを変換します。
- 変換されたデータファイルは、C:¥app¥orauser¥oradata¥orawin¥(ターゲット・システムの既存データファイルの場所)に配置されます。

ターゲット・システムで表領域を変換するには、次のステップを実行します。

1. SQL*Plusを実行している場合は、ホスト・システムに戻ります。

```
SQL> HOST
```

2. RMAN CONVERT DATAFILEコマンドを使用して、データファイルをターゲット・プラットフォームに変換します。

```
C:¥>RMAN TARGET /
Recovery Manager: Release 12.1.0.1.0 - Production
Copyright (c) 1982, 2012, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORAWIN (DBID=3462152886)
RMAN> CONVERT DATAFILE
2>'C:¥Temp¥sales_101.dbf',
3>'C:¥Temp¥sales_201.dbf'
4>TO PLATFORM="Microsoft Windows IA (32-bit)"
5>FROM PLATFORM="Solaris[tm] OE (32-bit)"
6>DB_FILE_NAME_CONVERT=
7>'C:¥Temp¥', 'C:¥app¥orauser¥oradata¥orawin¥'
8> PARALLELISM=4;
```

ソースの場所、ターゲットの場所、あるいはその両方でOracle Automatic Storage Management(Oracle ASM)を使用しない場合、ソース・プラットフォームとターゲット・プラットフォームはオプションです。RMANでは、データファイルを調べてソース・プラットフォームを判別し、デフォルトのターゲット・プラットフォームは、変換を実行するホストのプラットフォームになります。

ソースおよびターゲットの場所の両方でOracle ASMを使用する場合は、DB_FILE_NAME_CONVERT句でソース・プラットフォームとターゲット・プラットフォームを指定する必要があります。

関連項目:

RMANのCONVERTコマンドの詳細は、『Oracle Databaseバックアップおよびリカバリ・リファレンス』を参照してください。

3. Recovery Managerを終了します。

```
RMAN> exit
Recovery Manager complete.
```

親トピック: RMANを使用したプラットフォーム間でのデータの変換

15.6 データファイルを転送するためのガイドライン

データファイルを転送する際には、ガイドラインに従う必要があります。

ソースとターゲットの両方がファイル・システムの場合は、次の機能を使用してトランスポートできます。

- フラット・ファイルをコピーする機能(オペレーティング・システムのコピー・ユーティリティやFTPなど)
- DBMS_FILE_TRANSFERパッケージ
- RMAN
- CDで配布する機能

ソースまたはターゲットのいずれかがOracle Automatic Storage Management(Oracle ASM)ディスク・グループの場合は、次の機能を使用して転送できます。

- XML DBリポジトリの/sys/asm仮想フォルダ間でのFTP

  詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。
- DBMS_FILE_TRANSFERパッケージ
- RMAN

管理表領域(SYSTEMやSYSAUXなど)またはUNDO表領域や一時表領域のデータファイルはトランスポートしないでください。

データを受け取るデータベースの標準ブロック・サイズと異なるブロック・サイズのデータをトランスポートする場合は、最初にDB_nK_CACHE_SIZE初期化パラメータ・エントリを受取り側データベースのパラメータ・ファイル内に設定する必要があります。

たとえば、ブロック・サイズが8KBのデータを標準ブロック・サイズが4KBのデータベースにトランスポートする場合は、DB_8K_CACHE_SIZE初期化パラメータ・エントリをパラメータ・ファイルに含める必要があります。このエントリがまだパラメータ・ファイルに含まれていない場合は、ALTER SYSTEM SET文を使用してこのパラメータを設定できます。

DB_nK_CACHE_SIZE初期化パラメータの値の指定方法は、『Oracle Databaseリファレンス』を参照してください。

Oracle Database 12cからは、DBMS_FILE_TRANSFERパッケージのGET_FILEまたはPUT_FILEプロシージャを使用

して、データファイルの転送中にプラットフォーム間でデータを変換できます。「プラットフォーム間でのデータの変換」を参照してください。

Oracle Database 12cからは、RMANでネットワーク対応のリストアを使用してファイルを転送できます。RMANは、RESTOREコマンドのFROM SERVICE句を使用して、リモート・データベース・インスタンスからネットワーク経由でデータファイルをリストアします。ネットワーク対応リストアの主な利点は、バックアップをディスク上のステージング領域にリストアする必要性およびコピーを転送する必要性がなくなる点です。したがって、ネットワーク対応リストアを使用すると、ディスク領域および時間が節約されます。この手法には、ファイル転送時に、使用されたデータ・ブロックのみを圧縮、暗号化および転送できるという利点もあります。詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

ノート:

UNIX dd ユーティリティを使用してデータベース間で RAW デバイス・ファイルをコピーする場合は注意が必要です。Oracle Database 12c 以降では、データベース・ファイルに対して RAW デバイスがサポートされていません。dd ユーティリティは、ソース RAW デバイス・ファイル全体をコピーするために使用するか、またはソース RAW デバイス・ファイルの特定の範囲のみをコピーするようにオプションを指定して起動できます。

データファイルには非表示の制御情報も格納されているため、RAW デバイス・ファイルの実際のデータファイル・サイズを確認するのは困難です。dd ユーティリティを使用して RAW デバイスを操作する必要がある場合は、ソース RAW デバイス・ファイルの内容全体を指定します。データベース・ファイルの内容を RAW デバイスから ASM またはファイル・システムに移動する場合は、Oracle Database 12c 以降で RAW デバイスがサポートされていないため、オラクル社が提供する RMAN などのツールを使用してください。

関連項目:

DBMS_FILE_TRANSFERパッケージを使用して、トランスポートされるファイルとそのメタデータをコピーする方法は、「データベース・サーバーを使用したファイルのコピー」を参照してください

親トピック: データのトランスポート

16 UNDOの管理

デフォルトのインストールで、Oracle DatabaseによってUNDOが自動的に管理されます。通常、DBAによる操作は不要です。ただし、インストールでOracle Flashback操作を使用する場合は、それらの操作が正常に終了するように、UNDO管理タスクをいくつか実行することが必要な場合があります。

- UNDOの概要
  Oracle Databaseでは、データベースの変更をロールバックまたは取り消すために使用する情報を作成して管理します。これらの情報は、主にコミットされる前のトランザクションの処理レコードから構成されます。これらのレコードを総称してUNDOと呼びます。
- 自動UNDO管理の概要
  Oracle DatabaseでUNDO情報と領域を自動的に管理できます。
- 最小UNDO保存期間の設定
  最小UNDO保存期間(秒単位)を指定するには、UNDO_RETENTION初期化パラメータを設定します。
- 固定サイズのUNDO表領域のサイズ変更
  UNDO保存期間の自動チューニングは、通常、固定サイズのUNDO表領域に有効です。固定サイズの表領域を使用する場合は、UNDOアドバイザを使用すると、必要な容量を見積るのに役立ちます。
- UNDO表領域の管理
  UNDO表領域の管理では、表領域の作成、変更および削除などのタスクを行います。UNDO表領域を切り替えたり、UNDO領域のユーザー割当てを決定することもできます。
- 自動UNDO管理への移行
  現在、ロールバック・セグメントを使用してUNDO領域を管理している場合は、データベースを自動UNDO管理に移行することをお薦めします。
- 一時UNDOの管理
  デフォルトでは、一時表のUNDOレコードはUNDO表領域に格納され、REDOにログが記録されますが、これは永続表のUNDOが管理される方法と同じです。ただし、TEMP_UNDO_ENABLED初期化パラメータを使用すると、一時表のUNDOと永続表のUNDOを区別できます。このパラメータをTRUEに設定した場合、一時表のUNDOはtemporary undoと呼ばれます。
- UNDO領域のデータ・ディクショナリ・ビュー
  ビューのセットを問い合せて、自動UNDO管理モードのUNDO領域についての情報を取得できます。

関連項目:

Oracle Databaseによってデータファイルが作成および管理されるUNDO表領域の作成方法は、「Oracle Managed Filesの使用」を参照してください。

親トピック: Oracle Databaseの構造と記憶域

16.1 UNDOの概要

Oracle Databaseでは、データベースの変更をロールバックまたは取り消すために使用する情報を作成して管理します。これらの情報は、主にコミットされる前のトランザクションの処理レコードから構成されます。これらのレコードを総称してUNDOと呼びます。

UNDOレコードは次の処理に使用されます。

- ROLLBACK文を発行したときのトランザクションのロールバック
- データベースのリカバリ
- 読取り一貫性の実現
- Oracle Flashback Queryを使用した過去のある時点のデータの分析
- Oracle Flashback機能を使用した論理的な破損のリカバリ

ROLLBACK文を発行すると、コミットされていないトランザクションによってデータベースに加えられた変更が、UNDOレコードを使用して取り消されます。データベース・リカバリ時は、REDOログからデータファイルに適用されたコミットされていない変更が、UNDOレコードを使用してすべて取り消されます。UNDOレコードは、あるユーザーがデータを変更しているときに同じデータに同時にアクセスしようとしている別のユーザーのために、そのデータの変更前のイメージを維持することによって読込み一貫性を提供します。

関連項目:

『Oracle Database概要』

親トピック: UNDOの管理

16.2 自動UNDO管理の概要

Oracle Databaseは、UNDO情報と領域を自動的に管理できます。

- 自動UNDO管理の概要
  Oracleには、UNDO情報と領域を管理するために、自動UNDO管理と呼ばれる完全に自動化されたメカニズムが用意されています。自動UNDO管理では、データベースによってUNDOセグメントがUNDO表領域で管理されます。
- UNDO保存期間
  UNDO保存期間とは、Oracle Databaseで古いUNDO情報を上書きしないで保存する最低期間のことです。

親トピック: UNDOの管理

16.2.1 自動UNDO管理の概要

Oracleには、ロールバック情報と領域を管理するための、自動UNDO管理と呼ばれる完全に自動化されたメカニズムが用意されています。自動UNDO管理では、データベースによってUNDOセグメントがUNDO表領域で管理されます。

自動UNDO管理は、新しくインストールされたデータベースのデフォルト・モードです。Database Configuration Assistant(DBCA)を使用してデータベースを作成すると、UNDOTBS1という自動拡張可能なUNDO表領域が自動的に作成されます。

UNDO表領域は明示的に作成することもできます。UNDO表領域の作成方法については、「UNDO表領域の作成」を参照してください。

データベース・インスタンスが起動すると、データベースは最初に使用可能になったUNDO表領域を自動的に選択します。使用可能なUNDO表領域がない場合、インスタンスはUNDO表領域なしで起動され、UNDOレコードはSYSTEM表領域に格納されます。これは推奨される状態ではなく、アラート・ログ・ファイルには、システムがUNDO表領域のない状態で稼働していることを伝える警告メッセージが書き込まれます。

データベースに複数のUNDO表領域があるときは、必要に応じて、起動時に特定のUNDO表領域を使用するように指定する

こともできます。これには、次の例のように、UNDO_TABLESPACE初期化パラメータを設定します。

```
UNDO_TABLESPACE = undotbs_01
```

初期化パラメータで指定された表領域が存在しない場合、STARTUPコマンドは失敗します。Oracle Real Application Clusters環境でUNDO_TABLESPACEパラメータを使用すると、インスタンスに特定のUNDO表領域を割り当てることができます。

データベースは手動UNDO管理モードでも稼働できます。このモードでは、UNDO領域がロールバック・セグメントを介して管理され、UNDO表領域は使用されません。

ノート:

ロールバック・セグメントの領域管理は複雑です。データベースを自動 UNDO 管理モードのままにすることをお薦めします。

次に、UNDO管理用の初期化パラメータの概要を示します。

初期化パラメータ	説明
UNDO_MANAGEMENT	AUTO または NULL の場合は、自動 UNDO 管理を使用可能にします。MANUAL の場合は、手動 UNDO 管理モードを設定します。デフォルトは AUTO です。
UNDO_TABLESPACE	自動 UNDO 管理モードの場合のみ有効です(オプション)。UNDO 表領域の名前を指定します。データベースに複数の UNDO 表領域があり、データベース・インスタンスで特定の UNDO 表領域を使用するように指定する場合にのみ使用します。

自動UNDO管理が使用可能な場合は、初期化パラメータ・ファイルに手動UNDO管理に関するパラメータが含まれていても、それらは無視されます。

ノート:

Oracle Database の以前のリリースでは、手動 UNDO 管理モードがデフォルトです。自動 UNDO 管理モードに変更するには、UNDO 表領域を作成してから UNDO_MANAGEMENT 初期化パラメータを AUTO に変更する必要があります。Oracle Database が Oracle9i 以上で、自動 UNDO 管理に変更する場合、手順については『Oracle Database アップグレード・ガイド』を参照してください。

UNDO_MANAGEMENT 初期化パラメータが NULL の場合、Oracle Database 11g 以上では自動 UNDO 管理モードにデフォルト設定されますが、以前のリリースでは手動 UNDO 管理モードにデフォルト設定されます。したがって、以前のリリースを現在のリリースにアップグレードする場合は注意が必要です。UNDO 表領域のサイズの設定方法に関する情報など、自動 UNDO 管理モードに移行するための適切な方法については、『Oracle Database アップグレード・ガイド』を参照してください。

親トピック: 自動UNDO管理の概要

16.2.2 UNDO保存期間

UNDO保存期間とは、Oracle Databaseで古いUNDO情報を上書きしないで保存する最低期間のことです。

- UNDOの保存期間
  自動UNDO管理が有効な場合は、常に現在のUNDO保存期間が存在します。これは、Oracle Databaseで古いUNDO情報を上書きしないで保存する最低期間のことです。
- UNDOの保存期間の自動チューニング
  Oracle Databaseでは、UNDO表領域の構成方法に基づいて、UNDO保存期間を自動的にチューニングします。
- 保存期間の保証
  長時間実行される問合せやOracle Flashback操作を正常に行うために、保存期間の保証を有効化できます。
- UNDOの保存期間のチューニングとアラートしきい値
  固定サイズのUNDO表領域を使用する場合、データベースでは、データベース統計とUNDO表領域のサイズに基づいて最適な保存期間を計算します。
- チューニング済UNDO保存期間の追跡
  現行の保存期間を判別するには、V$UNDOSTATビューのTUNED_UNDORETENTION列を問い合せます。

親トピック: 自動UNDO管理の概要

16.2.2.1 UNDOの保存期間

自動UNDO管理が使用可能な場合は、常に現在のUNDO保存期間が存在します。これは、Oracle Databaseが古いロールバック情報を上書きするまでの最小保存期間です。

トランザクションがコミットされると、ロールバックまたはトランザクション・リカバリの実行にUNDOデータは不要になります。しかし、長時間実行の問合せ中にデータ・ブロックの変更前のイメージを生成する場合は、読込み一貫性を保証するために古いロールバック情報が必要になることがあります。また、いくつかのOracle Flashback機能を正常に終了させるには、古いロールバック情報が必要になる場合があります。このため、古いロールバック情報をできるかぎり長い期間保存することをお薦めします。

現在のUNDO保存期間よりも古い(コミット済の)UNDO情報は期限切れと呼ばれ、この領域は新しいトランザクションによって上書きできるようになります。現在のUNDO保存期間内の古いUNDO情報は期限切れでないUNDO情報と呼ばれ、読取り一貫性およびOracle Flashback操作のために保存されます。

Oracle Databaseでは、UNDO表領域サイズとシステム・アクティビティに基づいて、UNDO保存期間を自動的にチューニングします。必要に応じてUNDO_RETENTION初期化パラメータを設定することで、最小UNDO保存期間(秒単位)を指定できます。次に、UNDO保存期間にこのパラメータが具体的にどのような影響を与えるかを説明します。

- 固定サイズのUNDO表領域の場合、UNDO_RETENTIONパラメータは無視されます。データベースでは、システム・アクティビティとUNDO表領域サイズに基づいて、常に最適な保存期間を確保するようにUNDO保存期間をチューニングします。詳細は、「UNDOの保存期間の自動チューニング」を参照してください。
- AUTOEXTENDオプションが有効なUNDO表領域の場合、データベースでは、UNDO_RETENTIONで指定された最小保存期間を維持しようとします。空き領域が少なくなると、期限切れでないロールバック情報を上書きするかわりに、表領域が自動的に拡張されます。自動拡張可能なUNDO表領域に対してMAXSIZE句が指定されている場合は、最大サイズに到達すると、データベースは期限切れでないロールバック情報の上書きを開始する場合があります。DBCAで自動的に作成されたUNDOTBS1表領域は、自動的に拡張します。

親トピック: UNDO保存期間

16.2.2.2 UNDOの保存期間の自動チューニング

Oracle Databaseでは、UNDO表領域の構成方法に基づいて、UNDO保存期間を自動的にチューニングします。

- UNDO表領域がAUTOEXTENDオプションで構成されている場合、データベースでは、UNDOの保存期間を、システムでアクティブな最長実行問合せより若干長くなるように動的にチューニングします。ただし、この保存期間は、Oracle Flashback操作に対応するには不十分な場合があります。Oracle Flashback操作で「スナップショットが古すぎます」エラーが発生する場合は、これらの操作をサポートするのに十分なUNDOデータが保存されるように、ユーザーが介入する必要があることを示しています。Oracle Flashback機能にさらに対応するには、UNDO_RETENTIONパラメータを予想される最長のOracle Flashback操作時間と同じ値に設定したり、UNDO表領域を固定サイズに変更することもできます。
- UNDO表領域が固定サイズの場合、データベースでは、表領域のサイズと現行のシステムの負荷に対して最適な保存期間を確保するように、UNDO保存期間を動的にチューニングします。通常、この最適な保存期間は、アクティブな最長実行問合せの期間よりもかなり長くなります。

  UNDO表領域を固定サイズに変更する場合は、十分な大きさの表領域サイズを選択する必要があります。小さすぎるUNDO表領域サイズを選択すると、次の2つのエラーが発生する可能性があります。

  - DMLが失敗する可能性。これは、新しいトランザクションに対するUNDOを格納するための十分な領域がないためです。
  - 「スナップショットが古すぎます」というエラーが発生し、長時間実行問合せが失敗します。これは、UNDOデータが不足しているために読取り一貫性を維持できないことを意味します。

  詳細は、「固定サイズのUNDO表領域のサイズ変更」を参照してください。

> ノート:
>
> UNDO 保存の自動チューニングは、LOB に対してサポートされていません。これは、LOB のロールバック情報が UNDO 表領域ではなく、セグメント自体に格納されるためです。LOB の場合、データベースでは、UNDO_RETENTION で指定された最小保存期間を維持しようとします。ただし、空き領域が少なくなると、期限切れでない LOB のロールバック情報が上書きされる場合があります。

関連項目:

「最小UNDO保存期間の設定」

親トピック: UNDO保存期間

16.2.2.3 保存期間の保証

長時間実行される問合せやOracle Flashback操作を正常に行うために、保存期間の保証を有効化できます。

保存期間の保証を有効にすると、指定したUNDOの最小保存期間が保証され、UNDO表領域の領域不足によってトランザクションが失敗した場合でも、期限切れでないUNDOデータは上書きされません。保存期間の保証を有効にしないと、領域が十分でない場合、期限切れでないUNDOが上書きされる場合があるため、システムのUNDO保存期間が短くなります。このオプションはデフォルトでは無効です。

警告:

保存期間の保証を有効にすると、複数の DML 操作が失敗する可能性があります。この機能は注意して使用してください。

保存期間の保証を有効にするには、CREATE DATABASE文またはCREATE UNDO TABLESPACE文を使用してUNDO表領域を作成するときに、そのUNDO表領域に対してRETENTION GUARANTEE句を指定します。または、後でこの句をALTER TABLESPACE文で指定することもできます。保存期間の保証を無効にするには、RETENTION NOGUARANTEE句を使用します。

DBA_TABLESPACESビューを使用して、UNDO表領域の保存期間の保証の設定を確認できます。RETENTION列には、GUARANTEE、NOGUARANTEEまたはNOT APPLY(NOT APPLYはUNDO表領域以外の表領域で使用)の値が表示されます。

親トピック: UNDO保存期間

16.2.2.4 UNDOの保存期間のチューニングとアラートしきい値

固定サイズのUNDO表領域を使用する場合、データベースでは、データベース統計とUNDO表領域のサイズに基づいて最適な保存期間を計算します。

最適なUNDO管理を実現するために、データベースでは、UNDOの保存期間を、表領域サイズの100%ではなく、70%を基にチューニングするか、または使用済領域に対する警告アラートしきい値の率でチューニングするか、いずれか低い方に基づいてチューニングします。(警告アラートしきい値のデフォルトは70%ですが、変更が可能です。)したがって、UNDO表領域の警告アラートしきい値を70%未満に設定すると、チューニングされるUNDO保存期間の長さが短くなることがあります。表領域のアラートしきい値の詳細は、「表領域のアラートの管理」を参照してください。

親トピック: UNDO保存期間

16.2.2.5 チューニング済UNDO保存期間の追跡

現行の保存期間を判別するには、V$UNDOSTATビューのTUNED_UNDORETENTION列を問い合せます。

このビューには、過去4日間における10分単位の統計収集間隔ごとに1行が表示されます。(4日以前のデータは、DBA_HIST_UNDOSTATビューに表示されます。)TUNED_UNDORETENTIONは秒数で表示されます。

```
select to_char(begin_time, 'DD-MON-RR HH24:MI') begin_time,
to_char(end_time, 'DD-MON-RR HH24:MI') end_time, tuned_undoretention
from v$undostat order by end_time;
BEGIN_TIME      END_TIME        TUNED_UNDORETENTION
--------------- --------------- -------------------
04-FEB-05 00:01 04-FEB-05 00:11               12100
      ...
07-FEB-05 23:21 07-FEB-05 23:31               86700
07-FEB-05 23:31 07-FEB-05 23:41               86700
07-FEB-05 23:41 07-FEB-05 23:51               86700
07-FEB-05 23:51 07-FEB-05 23:52               86700
576 rows selected.
```

V$UNDOSTATの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: UNDO保存期間

16.3 最小UNDO保存期間の設定

最小UNDO保存期間(秒単位)を指定するには、UNDO_RETENTION初期化パラメータを設定します。

「UNDOの保存期間」で説明されているように、現行のUNDO保存期間はUNDO_RETENTIONよりも大きくなるように、または保存期間の保証が有効な場合を除き、空き領域が少ない場合はUNDO_RETENTIONよりも少なくなるように自動的にチューニングされます。

最小UNDO保存期間を設定するには:

- 次のいずれかの操作を行います。
  - 初期化パラメータ・ファイルでUNDO_RETENTIONを設定します。

    ```
    UNDO_RETENTION = 1800
    ```

  - UNDO_RETENTIONは、ALTER SYSTEM文を使用していつでも変更できます。

    ```
    ALTER SYSTEM SET UNDO_RETENTION = 2400;
    ```

UNDO_RETENTIONパラメータの変更は即時に反映されますが、その効果は、現行のUNDO表領域に十分な領域がある場合のみ表れます。

親トピック: UNDOの管理

16.4 固定サイズのUNDO表領域のサイズ変更

UNDO保存期間の動的チューニングは、通常、固定サイズのUNDO表領域に有効です。固定サイズの表領域を使用する場合は、UNDOアドバイザを使用すると、必要な容量を見積るのに役立ちます。

UNDOアドバイザには、Oracle Enterprise Manager Database Express (EM Express)またはDBMS_ADVISOR PL/SQLパッケージを介してアクセスできます。このアドバイザには、EM Expressを介してアクセスすることをお薦めします。EM Expressを介してUNDOアドバイザを使用する方法の詳細は、『Oracle Database 2日でデータベース管理者』を参照してください。

UNDOアドバイザは、自動ワークロード・リポジトリ(AWR)に収集されたデータに基づいて分析します。したがって、AWRで適切なワークロード統計を使用可能にし、UNDOアドバイザが正確な推奨事項を作成できるようにすることが重要です。新規作成したデータベースでは、適切な統計がすぐに使用できない場合があります。このような場合は、少なくとも1つのワークロード・サイクルが完了するまで、デフォルトの自動拡張可能なUNDO表領域を引き続き使用してください。

AWR統計の収集間隔と保存期間の調整は、アドバイザが作成する推奨事項の精度とタイプに影響します。詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

UNDOアドバイザを使用する場合は、最初に次の2つの値を見積ります。

- 最長実行問合せの予想される長さ

  データベースによるワークロード・サイクルが完了すると、「自動UNDO管理」ページの「システム・アクティビティ」サブページにある「最長実行問合せ」フィールドを表示できます。

- Oracle Flashback操作に必要な最長間隔

  たとえば、過去に最大48時間のOracle Flashback問合せを実行したと予想される場合、Oracle Flashback要件は48時間です。

次に、これら2つの値のうち最大値を選択し、その値をUNDOアドバイザへの入力として使用します。

UNDOアドバイザの実行によって、UNDO表領域のサイズは変更されません。アドバイザは推奨事項を単に返します。表領域のデータファイルを固定サイズに変更するには、ALTER DATABASE文を使用する必要があります。

次の例では、UNDO表領域にundotbs.dbfという自動拡張可能なデータファイルあると仮定します。この例では、表領域を300MBの固定サイズに変更します。

```
ALTER DATABASE DATAFILE '/oracle/dbs/undotbs.dbf' RESIZE 300M;
ALTER DATABASE DATAFILE '/oracle/dbs/undotbs.dbf' AUTOEXTEND OFF;
```

ノート:

UNDO 表領域を固定サイズにする場合は、まずデータベース作成後に十分な時間をかけて完全なワークロードを実行することをお薦めします。これにより、UNDO 表領域がワークロードを処理するための必要最小限のサイズになります。その後、必要に応じて UNDO アドバイザを使用して、長時間実行される問合せや Oracle Flashback 操作に対応するために、UNDO 表領域をどの程度拡大するかを決定します。

- UNDOアドバイザのPL/SQLインタフェースのアクティブ化
  UNDOアドバイザをアクティブにするには、アドバイザ・フレームワークを使用してUNDOアドバイザ・タスクを作成します。

関連項目:

UNDOアドバイザを使用してUNDO表領域の最小サイズを計算する手順は、『Oracle Database 2日でデータベース管理者』を参照してください。

親トピック: UNDOの管理

16.4.1 UNDOアドバイザのPL/SQLインタフェースのアクティブ化

UNDOアドバイザをアクティブにするには、アドバイザ・フレームワークを使用してUNDOアドバイザ・タスクを作成します。

次の例では、UNDO表領域を評価するためのUNDOアドバイザ・タスクを作成します。アドバイザの名前は'Undo Advisor'です。分析は自動ワークロード・リポジトリのスナップショットに基づいて実行され、このスナップショットは、START_SNAPSHOTパラメータとEND_SNAPSHOTパラメータを設定して指定する必要があります。次の例では、START_SNAPSHOTが1で、END_SNAPSHOTは2です。

```
DECLARE
   tid    NUMBER;
   tname  VARCHAR2(30);
   oid    NUMBER;
BEGIN
   DBMS_ADVISOR.CREATE_TASK('Undo Advisor', tid, tname, 'Undo Advisor Task');
   DBMS_ADVISOR.CREATE_OBJECT(tname, 'UNDO_TBS', null, null, null, 'null', oid);
   DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'TARGET_OBJECTS', oid);
   DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'START_SNAPSHOT', 1);
   DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'END_SNAPSHOT', 2);
   DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'INSTANCE', 1);
   DBMS_ADVISOR.execute_task(tname);
END;
/
```

アドバイザ・タスクを作成した後は、EM Expressの自動データベース診断モニターに出力および推奨事項を表示できます。こ

の情報は、DBA_ADVISOR_*データ・ディクショナリ・ビュー(DBA_ADVISOR_TASKS、DBA_ADVISOR_OBJECTS、DBA_ADVISOR_FINDINGS、DBA_ADVISOR_RECOMMENDATIONSなど)にも表示されます。

関連項目:

- 様々なアドバイザにアドバイザ・タスクを作成する例は、「セグメント・アドバイザの使用」を参照してください
- EM Expressの自動データベース診断モニターの詳細は、『Oracle Database 2日でデータベース管理者』を参照してください。
- DBA_ADVISOR_*データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 固定サイズのUNDO表領域のサイズ変更

16.5 UNDO表領域の管理

UNDO表領域の管理では、表領域の作成、変更および削除などのタスクを行います。UNDO表領域を切り替えたり、UNDO領域のユーザー割当てを決定することもできます。

- UNDO表領域の作成
  新規インストールでは、Database Configuration Assistant (DBCA)によってUNDO表領域が自動的に作成されますが、UNDO表領域を手動で作成することが必要な場合があります。
- UNDO表領域の変更
  UNDO表領域を変更するには、ALTER TABLESPACE文を使用します。
- UNDO表領域の削除
  UNDO表領域を削除するには、DROP TABLESPACE文を使用します。
- UNDO表領域の切替え
  あるUNDO表領域から別のUNDO表領域に切り替えることができます。UNDO_TABLESPACE初期化パラメータは動的パラメータであるため、ALTER SYSTEM SET文を使用して新しいUNDO表領域を割り当てることができます。
- UNDO領域に対するユーザー割当ての確立
  Oracle Database Resource Managerを使用すると、UNDO領域に対するユーザー割当てを確立できます。DBAは、データベース・リソース・マネージャのディレクティブUNDO_POOLを使用して、ユーザーのグループ(リソース・コンシューマ・グループ)が消費するUNDO表領域の量を制限できます。
- UNDO表領域に対する領域のアラートしきい値の管理
  Oracle Databaseでは、表領域に使用可能な領域が少なくなると事前にアラートが生成されるため、表領域のディスク領域の使用状況を管理するために役立ちます。

親トピック: UNDOの管理

16.5.1 UNDO表領域の作成

新規インストールでは、Database Configuration Assistant (DBCA)によってUNDO表領域が自動的に作成されますが、UNDO表領域を手動で作成することが必要な場合があります。

- UNDO表領域の作成について
  データベースを作成する際に、CREATE DATABASE文を使用してUNDO表領域を作成できます。既存のデータベースの場合は、CREATE UNDO TABLESPACE文を使用してUNDO表領域を作成できます。
- CREATE DATABASEを使用したUNDO表領域の作成

CREATE DATABASE文でUNDO TABLESPACE句を使用すると、特定のUNDO表領域を作成できます。

- CREATE UNDO TABLESPACE文の使用
  CREATE UNDO TABLESPACE 文はCREATE TABLESPACE文とほぼ同じですが、UNDOキーワードを指定します。UNDO表領域の属性のほとんどはデータベースが決定しますが、データベース管理者はDATAFILE句を指定できます。

親トピック: UNDO表領域の管理

16.5.1.1 UNDO表領域の作成について

データベースを作成する際に、CREATE DATABASE文を使用してUNDO表領域を作成できます。既存のデータベースの場合は、CREATE UNDO TABLESPACE文を使用してUNDO表領域を作成できます。

UNDO表領域の作成には2つの方法があります。1つは、CREATE DATABASE文の発行時にUNDO表領域を作成する方法です。これは、データベースを新規作成中にインスタンスが自動UNDO管理モードで起動したとき(UNDO_MANAGEMENT = AUTO)に実行されます。もう1つは、既存のデータベースで使用する方法です。この場合はCREATE UNDO TABLESPACE文を使用します。

UNDO表領域にはデータベース・オブジェクトは作成できません。システム管理のUNDOデータ用に確保されています。

Oracle Databaseでは、単一ファイルUNDO表領域を作成できます。単一ファイル(bigfile)表領域については、「bigfile表領域」を参照してください。

親トピック: UNDO表領域の作成

16.5.1.2 CREATE DATABASEを使用したUNDO表領域の作成

CREATE DATABASE文でUNDO TABLESPACE句を使用すると、特定のUNDO表領域を作成できます。

次の文は、CREATE DATABASE文でのUNDO TABLESPACE句の使用例を示しています。ここでは、UNDO表領域にundotbs_01という名前を付け、/u01/oracle/rbdb1/undo0101.dbfという1つのデータファイルを割り当てています。

```
CREATE DATABASE rbdb1
     CONTROLFILE REUSE
     .
     .
     .
     UNDO TABLESPACE undotbs_01 DATAFILE '/u01/oracle/rbdb1/undo0101.dbf';
```

CREATE DATABASEの実行中にUNDO表領域を正常に作成できない場合は、CREATE DATABASE操作全体が失敗します。データベース・ファイルをクリーン・アップし、エラーを訂正して、再度CREATE DATABASE操作を実行する必要があります。

また、データベース作成時に、CREATE DATABASE文を使用して単一ファイルUNDO表領域を作成できます。これについては、「データベース作成時のbigfile表領域のサポート」を参照してください。

関連項目:

CREATE DATABASE文を使用してUNDO表領域を作成する構文については、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: UNDO表領域の作成

16.5.1.3 CREATE UNDO TABLESPACE文の使用

CREATE UNDO TABLESPACE 文はCREATE TABLESPACE文とほぼ同じですが、UNDOキーワードを指定します。UNDO表領域の属性のほとんどはデータベースが決定しますが、データベース管理者はDATAFILE句を指定できます。

この例では、AUTOEXTENDオプションを使用してundotbs_02 UNDO表領域を作成しています。

```
CREATE UNDO TABLESPACE undotbs_02
     DATAFILE '/u01/oracle/rbdb1/undo0201.dbf' SIZE 2M REUSE AUTOEXTEND ON;
```

複数のUNDO表領域を作成できますが、UNDO表領域は1つのみアクティブにできます。

関連項目:

CREATE UNDO TABLESPACE文を使用してUNDO表領域を作成する構文については、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: UNDO表領域の作成

16.5.2 UNDO表領域の変更

UNDO表領域を変更するには、ALTER TABLESPACE文を使用します。

ただし、UNDO表領域のほとんどはシステムが管理しているため、考慮が必要になるのは次の操作のみです。

- データファイルの追加
- データファイル名の変更
- データファイルのオンライン化またはオフライン化
- データファイルのオープン状態のバックアップの開始または終了
- UNDOの保存期間の保証の有効化または無効化

DBAが変更可能な属性もこれらの属性のみです。

UNDO表領域が領域不足の場合、または領域不足の発生を防止する場合は、さらにファイルを追加したり、既存のデータファイルのサイズを変更できます。

次の例では、UNDO表領域undotbs_01にデータファイルを1つ追加しています。

```
ALTER TABLESPACE undotbs_01
     ADD DATAFILE '/u01/oracle/rbdb1/undo0102.dbf' AUTOEXTEND ON NEXT 1M
         MAXSIZE UNLIMITED;
```

ALTER DATABASE...DATAFILE文を使用すると、データファイルのサイズを変更または拡張できます。

関連項目:

- 「データファイルのサイズ変更」
- ALTER TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: UNDO表領域の管理

16.5.3 UNDO表領域の削除

UNDO表領域を削除するには、DROP TABLESPACE文を使用します。

次の例では、UNDO表領域undotbs_01を削除しています。

```
DROP TABLESPACE undotbs_01;
```

UNDO表領域は、現在どのインスタンスでも使用されていない場合にのみ削除できます。UNDO表領域に処理中のトランザクションが含まれている場合(トランザクションが失敗してまだリカバリされていない場合など)、DROP TABLESPACE文は失敗します。しかし、DROP TABLESPACEは、UNDO表領域に期限切れでない(保存期間内である)ロールバック情報が含まれている場合でもUNDO表領域を削除するため、既存の問合せでロールバック情報を必要とする場合は、UNDO表領域を削除しないように注意する必要があります。

UNDO表領域に対するDROP TABLESPACEは、DROP TABLESPACE...INCLUDING CONTENTSと同じように動作します。つまり、UNDO表領域の内容はすべて削除されます。

関連項目:

DROP TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: UNDO表領域の管理

16.5.4 UNDO表領域の切替え

あるUNDO表領域から別のUNDO表領域に切り替えることができます。UNDO_TABLESPACE初期化パラメータは動的パラメータであるため、ALTER SYSTEM SET文を使用して新しいUNDO表領域を割り当てることができます。

次の文は、新しいUNDO表領域に切り替えます。

```
ALTER SYSTEM SET UNDO_TABLESPACE = undotbs_02;
```

undotbs_01が現行のUNDO表領域であるとすると、このコマンドが正常に実行された後、インスタンスはundotbs_01のかわりにundotbs_02をUNDO表領域として使用します。

切替え先の表領域が次のいずれかの条件を満たす場合はエラーがレポートされ、切替えは行われません。

- 表領域が存在しない場合
- 表領域がUNDO表領域ではない場合
- 表領域が別のインスタンスによってすでに使用されている場合(Oracle RAC環境のみ)

切替え操作が実行されている間、データベースはオンラインであり、このコマンドの実行中でもユーザー・トランザクションを実行できます。切替え操作が正常に完了すると、切替え操作開始後に開始されたすべてのトランザクションが新しいUNDO表領域内のトランザクション表に割り当てられます。

切替え操作は、古いUNDO表領域内のトランザクションがコミットされるまで待機しません。古いUNDO表領域内に未処理のトランザクションがある場合、古いUNDO表領域はPENDING OFFLINEモード(状態)になります。このモードでは、既存のトランザクションは引き続き実行できますが、新しいユーザー・トランザクションのUNDOレコードをこのUNDO表領域に格納することはできません。

UNDO表領域は、切替え操作が正常に完了した後も、このPENDING OFFLINEモードのまま存在できます。PENDING OFFLINEのUNDO表領域は、別のインスタンスが使用することも、削除することもできません。最終的に、すべてのアクティブな

トランザクションがコミットされた後、UNDO表領域は自動的にPENDING OFFLINEモードからOFFLINEモードに移行します。それ以降は、他のインスタンスが(Oracle Real Application Cluster環境で)そのUNDO表領域を使用できます。

UNDO TABLESPACEのパラメータ値を''(2つの一重引用符)に設定した場合は、現行のUNDO表領域が次の使用可能なUNDO表領域に切り替えられます。使用可能なUNDO表領域がない場合もあるため、この文の使用には注意が必要です。

次の例では、現行のUNDO表領域の割当てを解除しています。

```
ALTER SYSTEM SET UNDO_TABLESPACE = '';
```

親トピック: UNDO表領域の管理

16.5.5 UNDO領域に対するユーザー割当ての確立

Oracle Database Resource Managerを使用すると、UNDO領域に対するユーザー割当てを確立できます。DBAは、データベース・リソース・マネージャのディレクティブUNDO_POOLを使用して、ユーザーのグループ(リソース・コンシューマ・グループ)が消費するUNDO表領域の量を制限できます。

UNDOプールは、コンシューマ・グループごとに指定できます。UNDOプールによって、コンシューマ・グループが生成できるUNDOの合計量が制御されます。コンシューマ・グループが生成するUNDOの合計量がそのUNDO制限を超えると、UNDOを生成している現行のUPDATEトランザクションが終了します。コンシューマ・グループの他のメンバーは、UNDO領域がプールから解放されるまで、新たに更新を実行できなくなります。

UNDO_POOLディレクティブが明示的に定義されていないときは、ユーザーは無制限にUNDO領域を使用できます。

関連項目:

Oracle Database Resource Managerを使用したリソースの管理

親トピック: UNDO表領域の管理

16.5.6 UNDO表領域に対する領域のアラートしきい値の管理

Oracle Databaseでは、表領域に使用可能な領域が少なくなると事前にアラートが生成されるため、表領域のディスク領域の使用状況を管理するために役立ちます。

UNDO表領域のアラートしきい値の設定方法は、「表領域のアラートの管理」を参照してください。

Oracle Databaseでは、UNDO領域の事前アラート以外に、「スナップショットが古すぎます」エラーが発生するような長時間実行される問合せがシステムにある場合にもアラートが生成されます。過剰にアラートが生成されるのを防ぐため、長時間実行される問合せのアラートは24時間以上の間隔をおいて発行されます。アラートが生成された場合は、EM Expressの「UNDOアドバイザ」ページをチェックしてUNDO表領域に関する詳細情報を参照できます。EM Expressを介してUNDOアドバイザを使用する方法の詳細は、『Oracle Database 2日でデータベース管理者』を参照してください。

親トピック: UNDO表領域の管理

16.6 自動UNDO管理への移行

現在、ロールバック・セグメントを使用してUNDO領域を管理している場合は、データベースを自動UNDO管理に移行することをお薦めします。

手順については、『Oracle Databaseアップグレード・ガイド』を参照してください。

16.7 一時UNDOの管理

デフォルトでは、一時表のUNDOレコードはUNDO表領域に格納され、REDOにログが記録されますが、これは永続表のUNDOが管理される方法と同じです。ただし、TEMP_UNDO_ENABLED初期化パラメータを使用すると、一時表のUNDOと永続表のUNDOを区別できます。このパラメータをTRUEに設定した場合、一時表のUNDOはtemporary undoと呼ばれます。

- 一時UNDOの管理について
  一時UNDOレコードはデータベースの一時表領域に格納され、REDOログには記録されません。一時UNDOを有効にすると、一時表領域によって使用されるセグメントの一部に一時UNDOが格納され、これらのセグメントは一時UNDOセグメントと呼ばれます。
- 一時UNDOの有効化と無効化
  セッションまたはシステムの一時UNDOを有効化または無効化できます。そのためには、TEMP_UNDO_ENABLED初期化パラメータを設定します。

親トピック: UNDOの管理

16.7.1 一時UNDOの管理について

一時UNDOレコードはデータベースの一時表領域に格納され、REDOログには記録されません。一時UNDOを有効にすると、一時表領域によって使用されるセグメントの一部に一時UNDOが格納され、これらのセグメントは一時UNDOセグメントと呼ばれます。

一時UNDOを有効にすると、UNDOレコードが占める一時表領域のサイズを大きくすることが必要になることがあります。

一時UNDOを有効にすると、次の利点があります。

- 一時UNDOによって、UNDO表領域に格納されるUNDOの量が削減されます。

  UNDO表領域内のUNDOが減少すると、UNDOレコードのUNDO保存期間要件がより現実的なものになることがあります。

- 一時UNDOにより、REDOログのサイズが縮小されます。

  REDOログに書き込まれるデータが減少することによってパフォーマンスが向上し、解析するREDOデータが減少することによって、LogMinerなどのREDOログ・レコードを解析するコンポーネントのパフォーマンスが向上します。

- 一時UNDOによって、フィジカル・スタンバイ・データベース内の一時表に対する、Oracle Active Data Guardオプションを使用したデータ操作言語(DML)操作が可能になります。ただし、一時表を作成するデータ定義言語(DDL)操作はプライマリ・データベースで発行する必要があります。

特定のセッションまたはシステム全体に対して一時UNDOを有効にできます。セッションに対してALTER SESSION文を使用して一時UNDOを有効にすると、そのセッションによって、他のセッションに影響を及ぼすことなく、一時UNDOが作成されます。システムに対してALTER SYSTEM文を使用して一時UNDOを有効にすると、既存のすべてのセッションと新規セッションによって一時UNDOが作成されます。

セッションで一時オブジェクトが最初に使用されるとき、TEMP_UNDO_ENABLED初期化パラメータの現在の値がセッションの残りの部分に設定されます。したがって、セッションに対して一時UNDOが有効であり、かつ、そのセッションで一時オブジェクトが使用されている場合、セッションに対して一時UNDOを無効にすることはできません。同様に、セッションに対して一時UNDOが無効であり、かつ、そのセッションで一時オブジェクトが使用されている場合、セッションに対して一時UNDOを有効にすることはでき

ません。

Oracle Active Data Guardオプションが指定されたフィジカル・スタンバイ・データベースに対しては、一時UNDOがデフォルトで有効になっています。このデフォルト設定のため、TEMP_UNDO_ENABLED初期化パラメータは、Active Data Guardオプションが指定されたフィジカル・スタンバイ・データベースには影響を及ぼしません。

ノート:

一時 UNDO は、データベースの互換性レベルが 12.0.0 以上である場合にのみ有効にできます。

関連項目:

- 「一時表の作成」
- 「UNDOの保存期間」
- TEMP_UNDO_ENABLED初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 『Oracle Data Guard概要および管理』
- 一時UNDOセグメントの詳細は、『Oracle Database概要』を参照してください。

親トピック: 一時UNDOの管理

16.7.2 一時UNDOの有効化と無効化

セッションまたはシステムに対して一時UNDOを有効または無効にできます。そのためには、TEMP_UNDO_ENABLED初期化パラメータを設定します。

一時UNDOを有効または無効にするには:

1. SQL*Plusで、データベースに接続します。

   セッションに対して一時UNDOを有効または無効にする場合は、SQL*Plusでセッションを開始します。

   システムに対して一時UNDOを有効または無効にする場合は、SQL*PlusでALTER SYSTEMシステム権限がある管理ユーザーとして接続します。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. TEMP_UNDO_ENABLED初期化パラメータを設定します。

   - セッションに対して一時UNDOを有効にするには、次のSQL文を実行します。

     ```
     ALTER SESSION SET TEMP_UNDO_ENABLED = TRUE;
     ```

   - セッションに対して一時UNDOを無効にするには、次のSQL文を実行します。

     ```
     ALTER SESSION SET TEMP_UNDO_ENABLED = FALSE;
     ```

   - システムに対して一時UNDOを有効にするには、次のSQL文を実行します。

     ```
     ALTER SYSTEM SET TEMP_UNDO_ENABLED = TRUE;
     ```

     システムに対して一時UNDOを有効にした後、ALTER SESSION文を使用してセッションの一時UNDOを

無効にできます。

- システムに対して一時UNDOを無効にするには、次のSQL文を実行します。

```
ALTER SYSTEM SET TEMP_UNDO_ENABLED = FALSE;
```

システムに対して一時UNDOを無効にした後、ALTER SESSION文を使用してセッションの一時UNDOを有効にできます。

サーバー・パラメータ・ファイルまたはテキスト初期化パラメータ・ファイルでTEMP_UNDO_ENABLEDをTRUEに設定して、システムに対して一時UNDOを有効にすることもできます。この場合、システムに対してはALTER SYSTEM文、またはセッションに対してはALTER SESSION文によって一時UNDOを無効にしないかぎり、すべての新規セッションで一時UNDOが作成されます。

関連項目:

- TEMP_UNDO_ENABLED初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- Oracle Data Guard環境での一時UNDOの有効化と無効化の詳細は、『Oracle Data Guard概要および管理』を参照してください。

親トピック: 一時UNDOの管理

16.8 UNDO領域のデータ・ディクショナリ・ビュー

ビューのセットを問い合せて、自動UNDO管理モードのUNDO領域についての情報を取得できます。

ここで紹介したビュー以外にも、表領域やデータファイルの情報を表示するビューを使用して、情報を取得できます。これらのビューの詳細は、「データファイルのデータ・ディクショナリ・ビュー」を参照してください。

UNDO表領域に関する領域情報を取得するために、次の動的パフォーマンス・ビューが役立ちます。

ビュー	説明
V$UNDOSTAT	UNDO 領域の監視とチューニングのための統計情報が含まれます。このビューは、現行の作業負荷に必要な UNDO 領域の量を見積る際に利用できます。また、データベースはこの情報を使用して、システム内の UNDO の使用方法をチューニングします。このビューの情報は、自動 UNDO 管理モードでのみ意味があります。
V$TEMPUNDOSTAT	一時 UNDO 領域の監視とチューニングのための統計情報が含まれます。このビューは、一時表領域で現行の作業負荷に必要な一時 UNDO 領域の量を見積る際に利用できます。また、データベースはこの情報を使用して、システム内の一時 UNDO の使用方法をチューニングします。このビューは、一時 UNDO が有効な場合にのみ意味があります。
V$ROLLSTAT	自動 UNDO 管理モードの場合、このビューの情報は、UNDO 表領域内の UNDO セグメントの動作を反映します。
V$TRANSACTION	UNDO セグメント情報が含まれます。

ビュー	説明
DBA_UNDO_EXTENTS	UNDO 表領域内の各エクステントの状態およびサイズを示します。
DBA_HIST_UNDOSTAT	V$UNDOSTAT 情報の統計スナップショットが含まれます。

V$UNDOSTATビューは、現行インスタンス内のUNDO領域におけるトランザクションの実行の効果を監視する際に役立ちます。UNDO領域の消費、トランザクションの同時実行性、UNDO保存期間のチューニング、インスタンス内の長時間実行される問合せの長さおよびSQL IDに関する統計が使用できます。

ビュー内の各行には、インスタンス内で10分ごとに収集された統計が表示されます。各行は、BEGIN_TIME列の値の降順で表示されます。各行は(BEGIN_TIME、END_TIME)でマークされた時間間隔に属します。各列は、その時間間隔中に特定の統計に関して収集されたデータを表します。ビューの最初の行は、現行の(部分的な)時間間隔に関する統計を示します。このビューには、4日サイクルによる合計576行が含まれます。

次の例は、V$UNDOSTATビューに対する問合せの結果を示したものです。

```
SELECT TO_CHAR(BEGIN_TIME, 'MM/DD/YYYY HH24:MI:SS') BEGIN_TIME,
       TO_CHAR(END_TIME, 'MM/DD/YYYY HH24:MI:SS') END_TIME,
       UNDOTSN, UNDOBLKS, TXNCOUNT, MAXCONCURRENCY AS "MAXCON"
       FROM v$UNDOSTAT WHERE rownum <= 144;

BEGIN_TIME          END_TIME               UNDOTSN   UNDOBLKS   TXNCOUNT     MAXCON
------------------- ------------------- ---------- ---------- ---------- ----------
10/28/2004 14:25:12 10/28/2004 14:32:17          8         74   12071108          3
10/28/2004 14:15:12 10/28/2004 14:25:12          8         49   12070698          2
10/28/2004 14:05:12 10/28/2004 14:15:12          8        125   12070220          1
10/28/2004 13:55:12 10/28/2004 14:05:12          8         99   12066511          3
...
10/27/2004 14:45:12 10/27/2004 14:55:12          8         15   11831676          1
10/27/2004 14:35:12 10/27/2004 14:45:12          8        154   11831165          2

144 rows selected.
```

この例は、10/27/2004の14:35:12から24時間前までの間に、システムでUNDO領域がどのように消費されたのかを示しています。

親トピック: UNDOの管理

17 Oracle Managed Filesの使用

Oracle Databaseは、データベースを構成するファイルを管理できます。

- Oracle Managed Filesについて
  Oracle Managed Filesは、データベースの管理を簡素化し、エラーを削減し、ディスク領域の無駄を減らします。
- Oracle Managed Filesの作成および使用の有効化
  Oracle Managed Filesを有効化し、使用するには、いくつかの初期化パラメータを設定します。
- Oracle Managed Filesの作成
  Oracle Managed Filesを使用して、データ・ファイル、一時ファイル、制御ファイル、REDOログ・ファイルおよびアーカイブ・ログを作成できます。
- Oracle Managed Filesの操作
  Oracle Managed Filesのファイル名は、既存のファイルを識別するためにファイル名が使用される場合ならいつでもSQL文で使用可能です。
- Oracle Managed Filesの使用例
  使用例では、Oracle Managed Filesの使用方法を示します。

親トピック: Oracle Databaseの構造と記憶域

17.1 Oracle Managed Filesについて

Oracle Managed Filesは、データベースの管理を簡素化し、エラーを削減し、ディスク領域の浪費を減らします。

- Oracle Managed Filesとは
  Oracle Managed Filesの使用は、Oracle Databaseの管理を簡素化します。Oracle Managed Filesにより、DBAは、Oracle Databaseを構成するオペレーティング・システム・ファイルを直接に管理する必要がなくなります。
- Oracle Managed Filesを使用できるユーザー
  Oracle Managed Filesは、特定のデータベース・タイプに最も有効です。
- 論理ボリューム・マネージャとは
  論理ボリューム・マネージャ(LVM)は、ほとんどのオペレーティング・システムで使用可能なソフトウェア・パッケージです。論理ディスク・マネージャ(LDM)と呼ばれることもあります。これを使用すると、複数の物理ディスクの断片を単一の連続するアドレス空間に結合し、ソフトウェアの上位レイヤーに対して1つのディスクとして表示できます。
- ファイル・システムとは
  ファイル・システムは、連続したディスク・アドレス・スペース内に構築されたデータ構造です。ファイル・マネージャ(FM)は、ファイル・システムを操作するソフトウェア・パッケージですが、これがファイル・システムと呼ばれる場合もあります。
- Oracle Managed Filesを使用する利点
  Oracle Managed Filesは、いくつかの利点を提供します。
- Oracle Managed Filesと既存の機能
  Oracle Managed Filesの使用によって、既存の機能がなくなることはありません。

親トピック: Oracle Managed Filesの使用

17.1.1 Oracle Managed Filesとは

Oracle Managed Filesを使用すると、Oracle Databaseの管理が簡素化されます。Oracle Managed Filesにより、DBAは、Oracle Databaseを構成するオペレーティング・システム・ファイルを直接に管理する必要がなくなります。

Oracle Managed Filesでは、データベース・オブジェクト・レベルでデータベースがファイルを自動的に作成、命名および管理するファイル・システム・ディレクトリを指定します。たとえば、指定する必要があるのは作成する表領域のみで、DATAFILE句を使用して表領域のデータファイルの名前およびパスを指定する必要はありません。この機能は、論理ボリューム・マネージャ(LVM)とともに使用すると効果的です。

データベースでは内部的に標準ファイル・システム・インタフェースを使用し、必要に応じて、次のデータベース構造のファイルを作成および削除します。

- 表領域
- REDOログ・ファイル
- 制御ファイル
- アーカイブ・ログ
- ブロック・チェンジ・トラッキング・ファイル
- フラッシュバック・ログ
- RMANバックアップ

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

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

関連項目:

Oracle Managed Filesの機能を拡充するOracle Databaseの統合ファイル・システムおよびボリューム・マネージャであるOracle Automatic Storage Management(Oracle ASM)の詳細は、Oracle Automatic Storage Management管理者ガイドを参照してください。Oracle Managed Filesによって、ファイルは自動的に作成および管理されますが、Oracle ASMでは、ストライプ化、ソフトウェアのミラー化、動的な記憶域の構成などの追加機能を利用でき、サード・パーティ製の論理ボリューム・マネージャを購入する必要がありません。

親トピック: Oracle Managed Filesについて

17.1.2 Oracle Managed Filesの使用対象

Oracle Managed Filesは、特定のデータベース・タイプに最も有効です。

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

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

Oracle Managed Filesではオペレーティング・システムのファイル・システムを使用する必要があるため、ディスク上のファイルの配置方法を制御できず、一部のI/Oをチューニングできなくなります。

親トピック: Oracle Managed Filesについて

17.1.3 論理ボリューム・マネージャの概要

論理ボリューム・マネージャ(LVM)は、大部分のオペレーティング・システムに付属するソフトウェア・パッケージです。論理ディスク・マネージャ(LDM)と呼ばれることもあります。これを使用すると、複数の物理ディスクの断片を単一の連続するアドレス空間に結合し、ソフトウェアの上位レイヤーに対して1つのディスクとして表示できます。

LVMを使用することで、基礎になるどの物理ディスクよりも、論理ボリュームに優れた容量、パフォーマンス、信頼性および可用性の特性を持たせることができます。これらの特性を実装するために、ミラーリング、ストライピング、連結およびRAID 5などの技術が使用されています。

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

親トピック: Oracle Managed Filesについて

17.1.4 ファイル・システムの概要

ファイル・システムとは、連続するディスク・アドレス空間内に構築されたデータ構造です。ファイル・マネージャ(FM)は、ファイル・システムを操作するソフトウェア・パッケージですが、これがファイル・システムと呼ばれる場合もあります。

オペレーティング・システムには必ずファイル・マネージャが組み込まれています。ファイル・マネージャの主要なタスクは、ファイル・システム内のファイルにディスク領域への割当てまたは割当て解除です。

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

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

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

親トピック: Oracle Managed Filesについて

17.1.5 Oracle Managed Filesの使用上の利点

Oracle Managed Filesは、いくつかの利点を提供します。

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

- データベースの管理が容易になります。

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

- 管理者による誤ったファイルの指定が原因で破損することが少なくなります。

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

- 不要なファイルの存在によるディスク領域の浪費が減少します。

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

- テスト・データベースおよび開発データベースを容易に作成できます。

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

- 移植可能なサード・パーティ製ツールの開発が容易になります。

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

親トピック: Oracle Managed Filesについて

17.1.6 Oracle Managed Filesと既存の機能

Oracle Managed Filesを使用しても、既存の機能が不要になるわけではありません。

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

親トピック: Oracle Managed Filesについて

17.2 Oracle Managed Filesの作成および使用の有効化

Oracle Managed Filesを有効化し、使用するには、いくつかの初期化パラメータを設定します。

- Oracle Managed Filesを有効化する初期化パラメータ
  次の表に、Oracle Managed Filesの使用を可能にする初期化パラメータを示します。
- DB_CREATE_FILE_DEST初期化パラメータの設定
  DB_CREATE_FILE_DEST初期化パラメータは、重要なデータベース・ファイルの場所を指定します。
- DB_RECOVERY_FILE_DESTパラメータの設定
  初期化パラメータ・ファイルにDB_RECOVERY_FILE_DESTおよびDB_RECOVERY_FILE_DEST_SIZEパラメータを設定して、高速リカバリ領域のデフォルトの場所を識別できるようにします。
- DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータの設定
  DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータは、REDOログ・ファイルおよび制御ファイルの場所を指定します。

親トピック: Oracle Managed Filesの使用

17.2.1 Oracle Managed Filesを有効化する初期化パラメータ

次の表に、Oracle Managed Filesの使用を有効化する初期化パラメータを示します。

初期化パラメータ	説明
DB_CREATE_FILE_DEST	作成操作でファイル仕様を指定しなかった場合に、データベースによってデータファイルまたは一時ファイルが作成されるデフォルトのファイル・システム・ディレクトリまたはOracle ASM ディスク・グループの場所を定義します。DB_CREATE_ONLINE_LOG_DEST_n を指定していない場合は、REDO ログ・ファイルおよび制御ファイルのデフォルトの場所としても使用されます。
DB_CREATE_ONLINE_LOG_DEST_n	作成操作でファイル仕様を指定しなかった場合に、REDO ログ・ファイルおよび制御ファイルが作成されるデフォルトのファイル・システム・ディレクトリまたは Oracle ASM ディスク・グループの場所を定義します。n を変更することによってこの初期化パラメータを複数回使用でき、n は REDO ログ・ファイルまたは制御ファイルの多重化コピーを指定します。多重化コピーは最大 5 つまで指定できます。
DB_RECOVERY_FILE_DEST	データベースによって RMAN によるバックアップ(フォーマット・オプションが使用されていない場合)、アーカイブ・ログ(他のローカル・アーカイブ先が構成されていない場合)およびフラッシュバック・ログが作成されるデフォルトのファイル・システム・ディレクトリまたは Oracle ASM ディスク・グループである高速リカバリ領域の場所を定義します。DB_CREATE_ONLINE_LOG_DEST_n を指定していない場合は、REDO ログ・ファイルおよび制御ファイルまたはその多重コピーのデフォルトの場所としても使用されます。このパラメータが指定されている場合、DB_RECOVERY_FILE_DEST_SIZE 初期化パラメータも指定する必要があります。

これらのパラメータによって指定するファイル・システム・ディレクトリはすでに存在している必要があり、データベースによって作成されません。また、ディレクトリには、データベースによるファイルの作成権限が必要です。

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

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

関連項目:

- 初期化パラメータの詳細は、『Oracle Databaseリファレンス』
- 「Oracle Managed Filesの命名方法」

親トピック: Oracle Managed Filesの作成および使用の有効化

17.2.2 DB_CREATE_FILE_DEST初期化パラメータの設定

DB_CREATE_FILE_DEST初期化パラメータは、重要なデータベース・ファイルの場所を指定します。

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

- データファイル
- 一時ファイル
- REDOログ・ファイル
- 制御ファイル
- ブロック・チェンジ・トラッキング・ファイル

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

```
DB_CREATE_FILE_DEST = '/u01/app/oracle/oradata'
```

親トピック: Oracle Managed Filesの作成および使用の有効化

17.2.3 DB_RECOVERY_FILE_DESTパラメータの設定

初期化パラメータ・ファイルにDB_RECOVERY_FILE_DESTおよびDB_RECOVERY_FILE_DEST_SIZEパラメータを設定して、高速リカバリ領域のデフォルトの場所を識別できるようにします。

高速リカバリ領域には、次が格納されます。

- REDOログ・ファイルまたはその多重コピー
- 制御ファイルまたはその多重コピー
- RMANバックアップ(データファイルのコピー、制御ファイルのコピー、バックアップ・ピース、制御ファイルの自動バックアップ)
- アーカイブ・ログ
- フラッシュバック・ログ

ファイル・システム・ディレクトリの名前を指定して、これらに対するオペレーティング・システム・ファイルを作成するためのデフォルトの場所にします。たとえば:

```
DB_RECOVERY_FILE_DEST = '/u01/app/oracle/fast_recovery_area'
DB_RECOVERY_FILE_DEST_SIZE = 20G
```

親トピック: Oracle Managed Filesの作成および使用の有効化

17.2.4 DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータの設定

DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータは、REDOログ・ファイルおよび制御ファイルの場所を指定します。

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

- REDOログ・ファイル
- 制御ファイル

ファイル・システム・ディレクトリまたはOracle ASMディスク・グループの名前を指定して、これらに対するファイルを作成するための

デフォルトの場所にします。多重コピーを配置する場所は最大5つまで指定できます。

REDOログ・ファイルおよび制御ファイルの作成時のみ、このパラメータはDB_CREATE_FILE_DESTおよびDB_RECOVERY_FILE_DEST初期化パラメータで指定されたデフォルトの場所よりも優先されます。DB_CREATE_FILE_DESTパラメータを指定せず、DB_CREATE_ONLINE_LOG_DEST_nパラメータのみを指定している場合は、REDOログ・ファイルと制御ファイルのみがOracle Managed Filesとして作成されます。

少なくとも2つのパラメータを指定することをお薦めします。たとえば:

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

これによって、ファイルを多重化できるため、REDOログ・ファイルまたは制御ファイルの保存先のどちらかで障害が発生した場合のフォルト・トレランスが向上します。

親トピック: Oracle Managed Filesの作成および使用の有効化

17.3 Oracle Managed Filesの作成

Oracle Managed Filesを使用して、データ・ファイル、一時ファイル、制御ファイル、REDOログ・ファイルおよびアーカイブ・ログを作成できます。

- Oracle DatabaseによるOracle Managed Filesの作成
  いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesを作成します。
- Oracle Managed Filesの命名方法
  Oracle Managed Filesのファイル名は、ファイル命名に関するOptimal Flexible Architecture (OFA)標準に準拠しています。
- データベース作成時のOracle Managed Filesの作成
  CREATE DATABASE文は、Oracle Managed Filesに関連するアクションを実行できます。
- Oracle Managed Filesを使用した表領域用データファイルの作成
  いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesを使用して表領域のデータファイルを作成できます。
- Oracle Managed Filesを使用した一時表領域用一時ファイルの作成
  いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesを使用して一時表領域の一時ファイルを作成できます。
- Oracle Managed Filesを使用した制御ファイルの作成
  いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesを使用して制御ファイルを作成できます。
- Oracle Managed Filesを使用したREDOログ・ファイルの作成
  REDOログ・ファイルはデータベース作成時に作成されます。次のいずれかの文を発行した場合にも作成できます: ALTER DATABASE ADD LOGFILEおよびALTER DATABASE OPEN RESETLOGS。
- Oracle Managed Filesを使用したアーカイブ・ログの作成
  アーカイブ・ログは、バックグラウンド・プロセスまたはSQL文によって作成されます。

親トピック: Oracle Managed Filesの使用

17.3.1 Oracle DatabaseによるOracle Managed Filesの作成

いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesが作成されます。

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

- 初期化パラメータ・ファイルに、DB_CREATE_FILE_DEST、DB_RECOVERY_FILE_DESTまたはDB_CREATE_ONLINE_LOG_DEST_n初期化パラメータのいずれかを指定している場合
- DB_RECOVERY_FILE_DEST、DB_CREATE_FILE_DESTまたはDB_CREATE_ONLINE_LOG_DEST_n初期化パラメータのいずれかを動的に設定するために、ALTER SYSTEM文を発行した場合
- DB_CREATE_FILE_DEST、DB_RECOVERY_FILE_DESTまたはDB_CREATE_ONLINE_LOG_DEST_n初期化パラメータのいずれかを動的に設定するために、ALTER SESSION文を発行した場合

Oracle Managed Filesを作成する文がエラーを検出した場合、またはなんらかの障害のために完了しなかった場合は、その文によって作成されたOracle Managed Filesはすべて、エラーまたは障害のリカバリの一部として自動的に削除されます。ただし、ファイル・システムやストレージ・サブシステムで発生する多数の潜在的なエラーが原因で、オペレーティング・システムのコマンドを使用した手動でのファイル削除が必要になる場合があります。

親トピック: Oracle Managed Filesの作成

17.3.2 Oracle Managed Filesの命名方法

Oracle Managed Filesのファイル名は、ファイル命名に関するOptimal Flexible Architecture(OFA)標準に準拠しています。

ノート:

この項で説明する命名方法は、オペレーティング・システムのファイル・システムに作成されるファイルにのみ適用されます。Oracle Automatic Storage Management(Oracle ASM)ディスク・グループで作成されたファイルの命名方式の詳細は、Oracle Automatic Storage Management 管理者ガイドを参照してください。

割り当てられた名前は、次の要件を満たしています。

- データベース・ファイルが他のすべてのファイルと容易に区別できます。
- 1つのデータベース・タイプのファイルが、他のデータベース・タイプのファイルと容易に区別できます。
- ファイルはファイル・タイプ固有の重要な属性に明確に関連付けられます。たとえば、データファイルの表領域への関連付けを容易にするためにデータファイル名に表領域名を含めたり、アーカイブ・ログ名にスレッド、順序および作成日を含めることが可能です。

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

- デフォルトの作成場所。
- ファイルのタイプに基づいて選択されたファイル名テンプレート。テンプレートは、オペレーティング・システムのプラットフォームや、Oracle Automatic Storage Managementを使用しているかどうかによって異なります。
- Oracle Databaseまたはオペレーティング・システムによって作成された一意の文字列。これにより、ファイル作成によって既存のファイルが破損しないこと、および誤って他のファイルが選択されないことが保証されます。

具体的な例として、Solarisファイル・システムにおけるOracle Managed Filesのファイル名の書式を次に示します。

```
destination_prefix/o1_mf_%t_%u_.dbf
```

ここで:

- destination_prefixはdestination_location/db_unique_name/datafileです。

  ここで:

  - destination_locationは、DB_CREATE_FILE_DESTに指定した場所です。
  - db_unique_nameは、ターゲット・データベースのグローバルな一意の名前(DB_UNIQUE_NAME初期化パラメータ)です。DB_UNIQUE_NAMEパラメータがない場合は、DB_NAME初期化パラメータ値が使用されます。
- %tは表領域の名前を表します。
- %uは一意性を保証する8文字の文字列を表します。

たとえば、次のようなパラメータ設定を考えてみます。

```
DB_CREATE_FILE_DEST    = /u01/app/oracle/oradata
DB_UNIQUE_NAME = PAYROLL
```

この例のデータファイル名は次のようになります。

```
/u01/app/oracle/oradata/PAYROLL/datafile/o1_mf_tbs1_2ixh90q_.dbf
```

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

後続の例では、SolarisファイルシステムでOMFアーカイブ先として表示されるOracle Managed Filesの名前を使用します。

ノート:

Oracle Managed Files は名前に基づいて識別されます。ファイル名を変更すると、データベースでは Oracle Managed Files として認識できなくなり、ファイルは適切に管理されません。

親トピック: Oracle Managed Filesの作成

17.3.3 データベース作成時のOracle Managed Filesの作成

CREATE DATABASE文は、Oracle Managed Filesに関連するアクションを実行できます。

ノート:

この項で説明するルールとデフォルトは、Database Configuration Assistant(DBCA)によるデータベースの作成にも適用されます。DBCA では、グラフィカル・インタフェースを使用して Oracle Managed Files を有効化し、この項で説明する初期化パラメータに対応するファイルの場所を指定できます。

- データベース作成時の制御ファイルの指定
  データベース作成時に、CONTROL_FILES初期化パラメータによって指定されたファイルに制御ファイルが作成されます。

- データベース作成時のREDOログ・ファイルの指定
  CREATE DATABASE文でLOGFILE句は必須でなく、これを省略すると、Oracle Managed FilesのREDOログ・ファイルを作成する簡単な手段が提供されます。
- データベース作成時のSYSTEM表領域およびSYSAUX表領域用データファイルの指定
  CREATE DATABASE文でDATAFILE句またはSYSAUX DATAFILE句は必須でなく、これを省略すると、SYSTEM表領域およびSYSAUX表領域用のOracle Managed Filesのデータファイルが簡単に作成されます。
- データベース作成時のUNDO表領域データファイルの指定
  UNDO TABLESPACE句のDATAFILE副次句は必須でなく、ファイル指定でファイル名は必要ありません。
- データベース作成時のデフォルト一時表領域用一時ファイルの指定
  DEFAULT TEMPORARY TABLESPACE句のTEMPFILE副次句は必須でなく、ファイル指定でファイル名は必要ありません。
- Oracle Managed Filesを使用したCREATE DATABASE文の例
  例では、Oracle Managed Files機能を使用する場合にCREATE DATABASE文を使用したデータベースの作成を示しています。

関連項目:

CREATE DATABASE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: Oracle Managed Filesの作成

17.3.3.1 データベース作成時の制御ファイルの指定

データベース作成時には、CONTROL_FILES初期化パラメータによって指定されたファイルで、制御ファイルが作成されます。

CONTROL_FILESパラメータが設定されておらず、Oracle Managed Filesの作成に必要な初期化パラメータが少なくとも1つ設定されている場合は、制御ファイルのデフォルトの保存先にOracle Managed Filesの制御ファイルが作成されます。デフォルトの保存先は、次の優先度に従って定義されます。

- DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータで指定されている1つ以上の制御ファイル。最初のディレクトリに作成されたファイルが主制御ファイルになります。DB_CREATE_ONLINE_LOG_DEST_nが指定されている場合は、DB_CREATE_FILE_DESTまたはDB_RECOVERY_FILE_DEST(高速リカバリ領域)に制御ファイルは作成されません。
- DB_CREATE_ONLINE_LOG_DEST_nに値が指定されていない場合で、DB_CREATE_FILE_DESTおよびDB_RECOVERY_FILE_DESTの両方に値が設定されている場合は、それぞれの場所に1つの制御ファイルが作成されます。DB_CREATE_FILE_DESTに指定された場所が主制御ファイルの場所になります。
- DB_CREATE_FILE_DESTに対してのみ値が指定されている場合は、その場所に1つの制御ファイルが作成されます。
- DB_RECOVERY_FILE_DESTに対してのみ値が指定されている場合は、その場所に1つの制御ファイルが作成されます。

CONTROL_FILESパラメータが設定されておらず、これらの初期化パラメータがいずれも設定されていない場合は、Oracle Databaseのデフォルトのアクションはオペレーティング・システムによって異なります。最低1つの制御ファイルのコピーが、デフォルトの場所(オペレーティング・システムによって異なる)に作成されます。このようにして作成された制御ファイルのコピーはOracle Managed Filesではなく、初期化パラメータ・ファイルにCONTROL_FILES初期化パラメータを追加する必要があります。

Oracle Managed Filesの制御ファイルが作成された場合で、サーバー・パラメータ・ファイルが存在するときは、サーバー・パラメータ・ファイルにCONTROL_FILES初期化パラメータのエントリが追加されます。サーバー・パラメータ・ファイルが存在しない場

合は、CONTROL_FILES初期化パラメータのエントリをテキスト形式の初期化パラメータ・ファイルに手動で追加する必要があります。

関連項目:

制御ファイルの管理

親トピック: データベース作成時のOracle Managed Filesの作成

17.3.3.2 データベース作成時のREDOログ・ファイルの指定

CREATE DATABASE文でLOGFILE句は必須でなく、単純にこれを省略するとOracle Managed FilesのREDOログ・ファイルが作成されます。

LOGFILE句を省略すると、デフォルトのREDOログ・ファイルの保存先にREDOログ・ファイルが作成されます。デフォルトの保存先は、次の優先度に従って定義されます。

- DB_CREATE_ONLINE_LOG_DEST_nが設定されている場合は、指定された各ディレクトリにログ・ファイルのメンバーが作成されます。最大数はMAXLOGMEMBERS初期化パラメータの値です。
- DB_CREATE_ONLINE_LOG_DEST_nパラメータが設定されていない場合で、DB_CREATE_FILE_DESTおよびDB_RECOVERY_FILE_DEST初期化パラメータの両方が設定されているときは、それぞれの場所に1つのOracle Managed Filesのログ・ファイルのメンバーが作成されます。DB_CREATE_FILE_DESTアーカイブ先のログ・ファイルが最初のメンバーです。
- DB_CREATE_FILE_DEST初期化パラメータのみが指定されている場合は、その場所にログ・ファイルのメンバーが作成されます。
- DB_RECOVERY_FILE_DEST初期化パラメータのみが指定されている場合は、その場所にログ・ファイルのメンバーが作成されます。

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

オプションで、Oracle Managed FilesのREDOログ・ファイルを作成し、LOGFILE句を含めながらファイル名を省略することにより、デフォルト属性を上書きできます。REDOログ・ファイルも同様に作成されますが、異なる点は、CREATE DATABASEのLOGFILE句にファイル名が入力されておらず、Oracle Managed Filesの作成に必要な初期化パラメータが1つも提供されていない場合、CREATE DATABASE文が失敗することです。

関連項目:

「REDOログの管理」

親トピック: データベース作成時のOracle Managed Filesの作成

17.3.3.3 データベース作成時のSYSTEM表領域およびSYSAUX表領域用データファイルの指定

CREATE DATABASE文でDATAFILE句またはSYSAUX DATAFILE句は必須でなく、単純にこれを省略するとSYSTEM表領域およびSYSAUX表領域用のOracle Managed Filesのデータファイルが作成されます。

DATAFILE句を省略すると、次の処理のいずれかが実行されます。

- DB_CREATE_FILE_DESTが設定されている場合は、SYSTEM表領域用およびSYSAUX表領域用のOracle Managed Filesのデータファイルが DB_CREATE_FILE_DESTディレクトリに1つずつ作成されます。

- DB_CREATE_FILE_DESTが設定されていない場合は、オペレーティング・システム固有の名前とサイズで、SYSTEM表領域のデータファイルおよびSYSAUX表領域のデータファイルが1つずつ作成されます。この方法で作成されたSYSTEM表領域のデータファイルまたはSYSAUX表領域のデータファイルはOracle Managed Filesではありません。

デフォルトでは、SYSTEMおよびSYSAUX表領域用を含めてOracle Managed Filesのデータファイルは100MBで、自動拡張可能です。自動拡張が必要な場合、データファイルは既存サイズまたは100MB単位(いずれか小さい方)で拡張されます。データファイルの指定時(CREATE操作またはALTER TABLESPACE操作時)に、STORAGE句のNEXTパラメータを使用して、自動拡張可能単位を明示的に指定することもできます。

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

Oracle Managed Filesのデフォルト属性を上書きする際、SIZE値が指定されており、AUTOEXTEND句が指定されていない場合、データファイルは自動拡張可能ではありません。

親トピック: データベース作成時のOracle Managed Filesの作成

17.3.3.4 データベース作成時のUNDO表領域データファイルの指定

UNDO TABLESPACE句のDATAFILE副次句は必須でなく、ファイル仕様に必ずしもファイル名を指定する必要はありません。

ファイル名を指定せず、DB_CREATE_FILE_DESTパラメータが設定されている場合は、Oracle Managed Filesのデータファイルが DB_CREATE_FILE_DESTディレクトリに作成されます。DB_CREATE_FILE_DESTが設定されていない場合は、構文エラーで文が失敗します。

UNDO TABLESPACE句自体は、CREATE DATABASE文のオプションです。この句を指定せず、自動UNDO管理モードが使用可能な場合(デフォルト)は、次のルールに従って、SYS_UNDOTSという名前のデフォルトのUNDO表領域が作成され、自動拡張可能な20MBのデータファイルが割り当てられます。

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

関連項目:

「UNDOの管理」

親トピック: データベース作成時のOracle Managed Filesの作成

17.3.3.5 データベース作成時のデフォルト一時表領域用一時ファイルの指定

DEFAULT TEMPORARY TABLESPACE句のTEMPFILE副次句は必須でなく、ファイル仕様に必ずしもファイル名を指定する必要はありません。

ファイル名を指定せず、DB_CREATE_FILE_DESTパラメータが設定されている場合は、Oracle Managed Filesの一時ファイルが DB_CREATE_FILE_DESTディレクトリに作成されます。DB_CREATE_FILE_DESTが設定されていない場合は、

構文エラーでCREATE DATABASE文が失敗します。

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

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

親トピック: データベース作成時のOracle Managed Filesの作成

17.3.3.6 Oracle Managed Filesを使用したCREATE DATABASE文の例

例では、Oracle Managed Files機能を使用する場合にCREATE DATABASE文を使用したデータベースの作成を示しています。

CREATE DATABASE: 例1

この例では、次のOracle Managed Filesを使用してデータベースを作成します。

- ディレクトリ/u01/app/oracle/oradataのSYSTEM表領域用データファイル。サイズは無制限に自動拡張可能です。
- ディレクトリ/u01/app/oracle/oradataのSYSAUX表領域用データファイル。サイズは無制限に自動拡張可能です。表領域は、自動セグメント領域管理によってローカルに管理されます。
- それぞれ100MBのメンバーを2つ含む2つのオンライン・ログ・グループ。/u02/oradataと/u03/oradataに1つずつ作成されます。
- 自動UNDO管理モードが使用可能の場合(デフォルト)は、ディレクトリ/u01/app/oracle/oradataのUNDO表領域用データファイル。サイズは20MBで、無制限に自動拡張可能です。SYS_UNDOTSという名前のUNDO表領域が作成されます。
- CONTROL_FILES初期化パラメータが指定されていない場合は、2つの制御ファイルが/u02/oradataと/u03/oradataに1つずつ作成されます。/u02/oradataの制御ファイルが主制御ファイルになります。

Oracle Managed Filesに関連する初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/app/oracle/oradata'
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata'
```

SQLプロンプトから次の文を発行します。

```
CREATE DATABASE sample;
```

ローカル管理されるSYSTEM表領域を持つデータベースを作成するには、EXTENT MANAGEMENT LOCAL句を追加します。

```
CREATE DATABASE sample EXTENT MANAGEMENT LOCAL;
```

この句を使用しない場合は、SYSTEM表領域はディクショナリ管理されます。ローカル管理のSYSTEM表領域を作成することをお薦めします。

CREATE DATABASE: 例2

この例では、次のOracle Managed Filesを使用してデータベースを作成します。

- ディレクトリ/u01/app/oracle/oradataのSYSTEM表領域用データファイル。サイズは無制限に自動拡張可能です。

- ディレクトリ/u01/app/oracle/oradataのSYSAUX表領域用データファイル。サイズは無制限に自動拡張可能です。表領域は、自動セグメント領域管理によってローカルに管理されます。
- ディレクトリ/u01/app/oracle/oradata の2つのREDOログ・ファイル(各100MB)。これらのファイルは多重化されていません。
- ディレクトリ/u01/app/oracle/oradataのUNDO表領域用データファイル。サイズは20MBで、無制限に自動拡張可能です。SYS_UNDOTSという名前のUNDO表領域が作成されます。
- /u01/app/oracle/oradataの1つの制御ファイル。

この例では、次のように想定されています。

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

SQLプロンプトから次の文を発行します。

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/app/oracle/oradata';
CREATE DATABASE sample2 EXTENT MANAGEMENT LOCAL;
```

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

CREATE DATABASE: 例3

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

- ディレクトリ/u01/app/oracle/oradataのSYSTEM表領域用データファイル(400MB)。SIZEが指定されているため、このファイルは自動拡張可能ではありません。
- ディレクトリ/u01/app/oracle/oradata.のSYSAUX表領域用データファイル(200MB)SIZEが指定されているため、このファイルは自動拡張可能ではありません。表領域は、自動セグメント領域管理によってローカルに管理されます。
- それぞれ100MBのメンバーを2つ含む2つのREDOログ・グループ。ディレクトリ/u02/oradataと/u03/oradataに1つずつ作成されます。
- デフォルト一時表領域dflt_tsに対して、ディレクトリ/u01/app/oracle/oradataの一時ファイル(10MB)。SIZEが指定されているため、このファイルは自動拡張可能ではありません。
- UNDO表領域undo_tsに対して、ディレクトリ/u01/app/oracle/oradataのデータファイル(100MB)。SIZEが指定されているため、このファイルは自動拡張可能ではありません。
- CONTROL_FILES初期化パラメータが指定されていない場合は、2つの制御ファイルがディレクトリ/u02/oradataと/u03/oradataに1つずつ作成されます。/u02/oradataの制御ファイルが主制御ファイルになります。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/app/oracle/oradata'
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata'
```

SQLプロンプトから次の文を発行します。

```
CREATE DATABASE sample3
EXTENT MANAGEMENT LOCAL
DATAFILE SIZE 400M
SYSAUX DATAFILE SIZE 200M
DEFAULT TEMPORARY TABLESPACE dflt_ts TEMPFILE SIZE 10M
UNDO TABLESPACE undo_ts DATAFILE SIZE 100M;
```

関連項目:

「ローカル管理のSYSTEM表領域の作成」

親トピック: データベース作成時のOracle Managed Filesの作成

17.3.4 Oracle Managed Filesを使用した表領域用データファイルの作成

いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesを使用して表領域のデータファイルを作成できます。

- Oracle Managed Filesを使用した表領域用データファイルの作成について
  いくつかの条件が満たされた場合、次のSQL文は、Oracle Managed Filesを使用して表領域のデータファイルを作成できます: `CREATE TABLESPACE`、`CREATE UNDO TABLESPACE`および`ALTER TABLESPACE ... ADD DATAFILE`。
- CREATE TABLESPACE: 例
  Oracle Managed Filesを使用して表領域を作成する例を示します。
- CREATE UNDO TABLESPACE: 例
  UNDO表領域を作成する例を示します。
- ALTER TABLESPACE: 例
  表領域にOracle Managed Filesの自動拡張可能なデータファイルを追加する例を示します。

親トピック: Oracle Managed Filesの作成

17.3.4.1 Oracle Managed Filesを使用した表領域用データファイルの作成について

いくつかの条件が満たされた場合、次のSQL文は、Oracle Managed Filesを使用して表領域のデータファイルを作成できます: `CREATE TABLESPACE`、`CREATE UNDO TABLESPACE`および`ALTER TABLESPACE ... ADD DATAFILE`。

次の文は、データファイルを作成できます。

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

表領域を作成するときは、永続表領域とUNDO表領域のいずれの場合も、`DATAFILE`句はオプションです。`DATAFILE`句を指定する場合、ファイル名はオプションです。`DATAFILE`句またはファイル名を省略すると、次のルールが適用されます。

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

`ALTER TABLESPACE...ADD DATAFILE`文で表領域にデータファイルを追加する場合、ファイル名はオプションです。ファ

イル名を省略すると、前の段落で説明したのと同じルールが適用されます。

デフォルトでは、永続表領域用のOracle Managed Filesのデータファイルのサイズは100MBです。このファイルは自動的に拡張可能で、最大サイズに制限はありません。ただし、DATAFILE句でSIZE値を指定する(AUTOEXTEND句は指定しない)ことによって、これらのデフォルトを上書きした場合、データファイルは自動拡張可能ではありません。

関連項目:

- 「データベース作成時のSYSTEM表領域およびSYSAUX表領域用データファイルの指定」
- 「データベース作成時のUNDO表領域データファイルの指定」
- 「表領域の管理」

親トピック: Oracle Managed Filesを使用した表領域用データファイルの作成

17.3.4.2 CREATE TABLESPACE: 例

Oracle Managed Filesを使用して表領域を作成する例を示します。

関連項目:

CREATE TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』

CREATE TABLESPACE: 例1

次の例では、データファイルを作成するデフォルトの場所を/u01/oradataに設定し、その場所のデータファイルを含む表領域tbs_1を作成します。データファイルは100MBで、無制限に自動拡張可能です。

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

CREATE TABLESPACE: 例2

この例では、/u01/oradataディレクトリにデータファイルを含むtbs_2という表領域を作成します。データファイルの初期サイズは400 MBで、SIZE句が指定されているため自動拡張可能ではありません。

初期化パラメータ・ファイルで、次のパラメータを設定します。

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

SQLプロンプトから次の文を発行します。

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

CREATE TABLESPACE: 例3

この例では、/u01/oradataディレクトリに自動拡張可能なデータファイルを含むtbs_3という表領域を作成します(初期サイズ100MB、最大サイズ800MB)。

初期化パラメータ・ファイルで、次のパラメータを設定します。

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

SQLプロンプトから次の文を発行します。

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

CREATE TABLESPACE: 例4

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

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

親トピック: Oracle Managed Filesを使用した表領域用データファイルの作成

17.3.4.3 CREATE UNDO TABLESPACE: 例

UNDO表領域を作成する例を示します。

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

1. 次の初期化パラメータを設定します。

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

2. 次のSQL文を発行します。

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

関連項目:

CREATE UNDO TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: Oracle Managed Filesを使用した表領域用データファイルの作成

17.3.4.4 ALTER TABLESPACE: 例

表領域にOracle Managed Filesの自動拡張可能なデータファイルを追加する例を示します。

この例では、自動拡張可能なOracle Managed Filesのデータファイルをtbs_1表領域に追加しています。データファイルは初期サイズが100MBで、最大サイズが800MBです。

1. 次の初期化パラメータを設定します。

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

2. 次のSQL文を発行します。

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

関連項目:

ALTER TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: Oracle Managed Filesを使用した表領域用データファイルの作成

17.3.5 Oracle Managed Filesを使用した一時表領域用一時ファイルの作成

いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesを使用して一時表領域の一時ファイルを作成できます。

- Oracle Managed Filesを使用した一時表領域用一時ファイルの作成について
  いくつかの条件が満たされた場合、次のSQL文は、Oracle Managed Filesを使用して表領域の一時ファイルを作成できます: `CREATE TEMPORARY TABLESPACE`および`ALTER TABLESPACE ... ADD TEMPFILE`。
- CREATE TEMPORARY TABLESPACE: 例
  一時表領域を作成する例を示します。
- ALTER TABLESPACE... ADD TEMPFILE: 例
  一時表領域に一時ファイルを追加する例を示します。

親トピック: Oracle Managed Filesの作成

17.3.5.1 Oracle Managed Filesを使用した一時表領域用一時ファイルの作成について

いくつかの条件が満たされた場合、次のSQL文は、Oracle Managed Filesを使用して表領域の一時ファイルを作成できます: `CREATE TEMPORARY TABLESPACE`および`ALTER TABLESPACE ... ADD TEMPFILE`。

ここでは、一時ファイルを作成する次の文について説明します。

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

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

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

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

Oracle Managed Filesのデフォルト属性を上書きする際、`SIZE`値が指定されており、`AUTOEXTEND`句が指定されていない場合、データファイルは自動拡張可能ではありません。

関連項目:

「データベース作成時のデフォルト一時表領域用一時ファイルの指定」

親トピック: Oracle Managed Filesを使用した一時表領域用一時ファイルの作成

17.3.5.2 CREATE TEMPORARY TABLESPACE: 例

一時表領域を作成する例を示します。

次の例では、データファイルを作成するデフォルトの場所を`/u01/oradata`に設定してから、その場所の一時ファイルを含む表領域`temptbs_1`を作成します。一時ファイルは100MBで、無制限に自動拡張可能です。

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

関連項目:

CREATE TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: Oracle Managed Filesを使用した一時表領域用一時ファイルの作成

17.3.5.3 ALTER TABLESPACE... ADD TEMPFILE: 例

一時表領域に一時ファイルを追加する例を示します。

次の例では、データファイルを作成するデフォルトの場所を/u03/oradataに設定してから、デフォルトの場所のtemptbs_1という表領域に一時ファイルを追加しています。一時ファイルの初期サイズは100MBです。これは無制限に自動拡張可能です。

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

関連項目:

ALTER TABLESPACE文の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: Oracle Managed Filesを使用した一時表領域用一時ファイルの作成

17.3.6 Oracle Managed Filesを使用した制御ファイルの作成

いくつかの条件が満たされた場合、Oracle DatabaseではOracle Managed Filesを使用して制御ファイルを作成できます。

- Oracle Managed Filesを使用した制御ファイルの作成について
  いくつかの条件が満たされた場合、CREATE CONTROLFILE SQL文により、Oracle Managed Filesを使用して制御ファイルを作成できます。
- NORESETLOGSキーワードを使用したCREATE CONTROLFILE: 例
  NORESETLOGSキーワードを含むCREATE CONTROLFILE文を使用して制御ファイルを作成する例を示します。
- RESETLOGSキーワードを使用したCREATE CONTROLFILE: 例
  RESETLOGSキーワードを含むCREATE CONTROLFILE文を使用して制御ファイルを作成する例を示します。

親トピック: Oracle Managed Filesの作成

17.3.6.1 Oracle Managed Filesを使用した制御ファイルの作成について

いくつかの条件が満たされた場合、CREATE CONTROLFILE SQL文により、Oracle Managed Filesを使用して制御ファイルを作成できます。

CREATE CONTROLFILE文を発行すると、CONTROL_FILES初期化パラメータによって指定されたファイルで、制御ファイルが作成(REUSEを指定した場合は再利用)されます。CONTROL_FILESパラメータが設定されていない場合は、制御ファイルのデフォルトの保存先に制御ファイルが作成されます。デフォルトの保存先は、「データベース作成時の制御ファイルの指定」に記載されている優先度によって決まります。

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

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

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

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

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

関連項目:

- CREATE CONTROLFILE文の詳細は、『Oracle Database SQL言語リファレンス』
- 「データベース作成時の制御ファイルの指定」

親トピック: Oracle Managed Filesを使用した制御ファイルの作成

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

NORESETLOGSキーワードを含むCREATE CONTROLFILE文を使用して制御ファイルを作成する例を示します。

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

```
CREATE CONTROLFILE
     DATABASE sample
     LOGFILE
       GROUP 1 ('/u01/oradata/SAMPLE/onlinelog/o1_mf_1_o220rtt9_.log',
                '/u02/oradata/SAMPLE/onlinelog/o1_mf_1_v2o0b2i3_.log')
                 SIZE 100M,
       GROUP 2 ('/u01/oradata/SAMPLE/onlinelog/o1_mf_2_p22056iw_.log',
                '/u02/oradata/SAMPLE/onlinelog/o1_mf_2_p02rcyg3_.log')
                 SIZE 100M
     NORESETLOGS
     DATAFILE '/u01/oradata/SAMPLE/datafile/o1_mf_system_xu34ybm2_.dbf'
              SIZE 100M,
              '/u01/oradata/SAMPLE/datafile/o1_mf_sysaux_aawbmz51_.dbf'
              SIZE 100M,
              '/u01/oradata/SAMPLE/datafile/o1_mf_sys_undo_apqbmz51_.dbf'
              SIZE 100M
     MAXLOGFILES 5
     MAXLOGHISTORY 100
     MAXDATAFILES 10
     MAXINSTANCES 2
     ARCHIVELOG;
```

親トピック: Oracle Managed Filesを使用した制御ファイルの作成

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

RESETLOGSキーワードを含むCREATE CONTROLFILE文を使用して制御ファイルを作成する例を示します。

次の文は、RESETLOGSオプションを指定したCREATE CONTROLFILE文の例です。DB_CREATE_FILE_DEST、DB_RECOVERY_FILE_DESTおよびDB_CREATE_ONLINE_LOG_DEST_nを組み合せて設定する必要があります。

```
CREATE CONTROLFILE
     DATABASE sample
     RESETLOGS
     DATAFILE '/u01/oradata/SAMPLE/datafile/o1_mf_system_aawbmz51_.dbf',
              '/u01/oradata/SAMPLE/datafile/o1_mf_sysaux_axybmz51_.dbf',
              '/u01/oradata/SAMPLE/datafile/o1_mf_sys_undo_azzbmz51_.dbf'
     SIZE 100M
     MAXLOGFILES 5
     MAXLOGHISTORY 100
     MAXDATAFILES 10
     MAXINSTANCES 2
     ARCHIVELOG;
```

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

親トピック: Oracle Managed Filesを使用した制御ファイルの作成

17.3.7 Oracle Managed Filesを使用したREDOログ・ファイルの作成

REDOログ・ファイルはデータベース作成時に作成されます。次のいずれかの文を発行した場合にも作成できます: ALTER DATABASE ADD LOGFILEおよびALTER DATABASE OPEN RESETLOGS。

- ALTER DATABASE ADD LOGFILE文の使用
  ALTER DATABASE ADD LOGFILE文により、現在のREDOログに後で新しいグループを追加できます。
- ALTER DATABASE OPEN RESETLOGS文の使用
  前にRESETLOGSを指定して制御ファイルを作成しており、その際、ファイル名を指定しなかった場合、または存在しないファイル名を指定した場合は、ALTER DATABASE OPEN RESETLOGS文を発行したときに、REDOログ・ファイルが作成されます。

関連項目:

ALTER DATABASE文の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: Oracle Managed Filesの作成

17.3.7.1 ALTER DATABASE ADD LOGFILE文の使用

ALTER DATABASE ADD LOGFILE文を使用すると、現行のREDOログ・ファイルに後から新しいグループを追加できます。

Oracle Managed Filesを使用している場合、ADD LOGFILE句のファイル名はオプションです。ファイル名を省略した場合は、ログ・ファイルのデフォルトの保存先にREDOログ・ファイルが作成されます。デフォルトの保存先は、「データベース作成時のREDOログ・ファイルの指定」に記載されている優先度によって決まります。

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

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

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

関連項目:

- 「データベース作成時のREDOログ・ファイルの指定」
- 「Oracle Managed Filesを使用した制御ファイルの作成について」

新しいREDOログ・ファイルの追加: 例

次の例では、一方のメンバーが/u01/oradataに、もう一方のメンバーが/u02/oradataに存在するログ・グループを作成します。各ログ・ファイルのサイズは100MBです。

初期化パラメータ・ファイルで、次のパラメータを設定します。

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

SQLプロンプトから次の文を発行します。

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

親トピック: Oracle Managed Filesを使用したREDOログ・ファイルの作成

17.3.7.2 ALTER DATABASE OPEN RESETLOGS文の使用

前にRESETLOGSを指定して制御ファイルを作成しており、その際、ファイル名を指定しなかった場合、または存在しないファイル名を指定した場合は、ALTER DATABASE OPEN RESETLOGS文を発行したときに、REDOログ・ファイルが作成されます。

制御ファイル内に何も指定されていない場合に、REDOログ・ファイルの格納ディレクトリを決めるルールは、「データベース作成時のREDOログ・ファイルの指定」に記載されているルールと同じです。

親トピック: Oracle Managed Filesを使用したREDOログ・ファイルの作成

17.3.8 Oracle Managed Filesを使用したアーカイブ・ログの作成

アーカイブ・ログは、バックグラウンド・プロセスまたはSQL文によって作成されます。

アーカイブ・ログは、次の場合にDB_RECOVERY_FILE_DESTの場所に作成されます。

- ARCバックグラウンド・プロセスまたはLGWRバックグラウンド・プロセスがオンラインREDOログをアーカイブする場合。または、
- ALTER SYSTEM ARCHIVE LOG CURRENT文が発行された場合。

たとえば、次のパラメータ設定が初期化パラメータ・ファイルに含まれているとします。

```
DB_RECOVERY_FILE_DEST_SIZE = 20G
DB_RECOVERY_FILE_DEST      = '/u01/oradata'
LOG_ARCHIVE_DEST_1         = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
```

親トピック: Oracle Managed Filesの作成

17.4 Oracle Managed Filesの操作

Oracle Managed Filesのファイル名は、既存のファイルを識別するためにファイル名が使用される場合ならいつでもSQL文で使用可能です。

これらのファイル名は、他のファイル名と同様に制御ファイルと、またバックアップおよびリカバリにRecovery Manager (RMAN)を使用している場合はRMANカタログに格納されます。これらは、データファイルおよび一時ファイルの監視に使用される通常の固定および動的パフォーマンス・ビューで参照可能です(V$DATAFILEまたはDBA_DATA_FILESなど)。

次に、データベース生成ファイル名を使用した文の例を示します。

```
SQL> ALTER DATABASE
  2> RENAME FILE '/u01/oradata/mydb/datafile/o1_mf_tbs01_ziw3bopb_.dbf'
  3> TO '/u01/oradata/mydb/tbs0101.dbf';
SQL> ALTER DATABASE
  2> DROP LOGFILE '/u01/oradata/mydb/onlinelog/o1_mf_1_wo94n2xi_.log';
SQL> ALTER TABLE emp
  2> ALLOCATE EXTENT
  3> (DATAFILE '/u01/oradata/mydb/datafile/o1_mf_tbs1_2ixfh90q_.dbf');
```

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

ファイルの削除またはファイルの名前変更の操作、およびスタンバイ・データベースに関連した操作を含むいくつかのケースでは、Oracle Managed Filesの動作が異なる場合があります。

- データファイルおよび一時ファイルの削除
  データベースによって管理されていないファイルとは異なり、Oracle Managed Filesのデータファイルまたは一時ファイルを削除すると、制御ファイルからファイル名が削除されて、ファイル・システムからファイルが自動的に削除されます。
- REDOログ・ファイルの削除
  Oracle Managed FilesのREDOログ・ファイルを削除すると、そのOracle Managed Filesが削除されます。削除するグループまたはメンバーを指定します。
- ファイルの名前変更
  Oracle Managed Filesでは、ファイル名を変更するSQL文によって、実際のオペレーティング・システム上のファイル名が変更されることはなく、制御ファイル内の名前が変更されます。
- スタンバイ・データベースの管理
  スタンバイ・データベースのデータファイル、制御ファイルおよびREDOログ・ファイルは、データベースで管理できます。プライマリ・データベースでOracle Managed Filesが使用されているかどうかは関係ありません。

親トピック: Oracle Managed Filesの使用

17.4.1 データファイルおよび一時ファイルの削除

データベースによって管理されていないファイルとは異なり、Oracle Managed Filesのデータファイルまたは一時ファイルを削除すると、制御ファイルからファイル名が削除されて、ファイル・システムからファイルが自動的に削除されます。

Oracle Managed Filesを削除する文は、次のとおりです。

- `DROP TABLESPACE`
- `ALTER DATABASE TEMPFILE ... DROP`

これらの文を使用して、いつでもOracle Managed Filesあるいはそれ以外のファイルを削除することもできます。

- `ALTER TABLESPACE ... DROP DATAFILE`
- `ALTER TABLESPACE ... DROP TEMPFILE`

親トピック: Oracle Managed Filesの操作

17.4.2 REDOログ・ファイルの削除

Oracle Managed FilesのREDOログ・ファイルを削除すると、そのOracle Managed Filesが削除されます。削除するグループまたはメンバーを指定します。

次の文は、REDOログ・ファイルを削除します。

- `ALTER DATABASE DROP LOGFILE`
- `ALTER DATABASE DROP LOGFILE MEMBER`

親トピック: Oracle Managed Filesの操作

17.4.3 ファイルの名前変更

Oracle Managed Filesでは、ファイル名を変更するSQL文によって、実際のオペレーティング・システム上のファイル名が変更されることはなく、制御ファイル内の名前が変更されます。

ファイルの名前を変更するには、次の文が使用されます。

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

この文を発行するときは、オペレーティング・システムのファイル名の規則を使用して各ファイル名を指定する必要があります。

ノート:

変更前のファイルが Oracle Managed Files で、そのファイルが存在している場合は削除されます。

親トピック: Oracle Managed Filesの操作

17.4.4 スタンバイ・データベースの管理

スタンバイ・データベースのデータファイル、制御ファイルおよびREDOログ・ファイルは、データベースで管理できます。プライマリ・データベースでOracle Managed Filesが使用されているかどうかは関係ありません。

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

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

親トピック: Oracle Managed Filesの操作

17.5 Oracle Managed Filesの使用例

使用例では、Oracle Managed Filesの使用方法を示します。

- 使用例1: 多重REDOログを含むデータベースの作成および管理
  多重化されたREDOログを含むデータベースを作成および管理する例を示します。
- 使用例2: データベース領域と高速リカバリ領域を含むデータベースの作成と管理
  データベース領域と高速リカバリ領域の両方を含むデータベースを作成および管理する例を示します。

- 使用例3: 既存のデータベースへのOracle Managed Filesの追加
  既存データベースにOracle Managed Filesを追加する例を示します。

親トピック: Oracle Managed Filesの使用

17.5.1 使用例1: 多重REDOログを含むデータベースの作成および管理

多重化されたREDOログを含むデータベースを作成および管理する例を示します。

この使用例では、DBAが、データファイルとREDOログ・ファイルが異なるディレクトリに存在するデータベースを作成します。REDOログ・ファイルと制御ファイルは多重化されています。データベースはUNDO表領域を使用し、デフォルト一時表領域を持っています。このデータベースの作成とメンテナンスに関するタスクは、次のとおりです。

1. 初期化パラメータの設定

   DBAは、データベースを作成する前に、初期化パラメータ・ファイルに3つの汎用的なファイル作成デフォルトを設定します。自動UNDO管理モード(デフォルト)も使用可能にします。

   ```
   DB_CREATE_FILE_DEST = '/u01/oradata'
   DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata'
   DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata'
   UNDO_MANAGEMENT = AUTO
   ```

   DB_CREATE_FILE_DESTパラメータは、データファイルおよび一時ファイルのデフォルトのファイル・システム・ディレクトリを設定します。

   DB_CREATE_ONLINE_LOG_DEST_1およびDB_CREATE_ONLINE_LOG_DEST_2パラメータは、REDOログ・ファイルと制御ファイルを作成するためのデフォルトのファイル・システム・ディレクトリを設定します。REDOログ・ファイルと制御ファイルは、2つのディレクトリの間で多重化されます。

2. データベースの作成

   初期化パラメータの設定が完了すると、次の文でデータベースを作成できます。

   ```
   SQL> CREATE DATABASE sample
   2>   DEFAULT TEMPORARY TABLESPACE dflttmp;
   ```

   DATAFILE句が指定されておらず、DB_CREATE_FILE_DEST初期化パラメータが設定されているため、SYSTEM表領域のデータファイルはデフォルトのファイル・システム(この使用例では/u01/oradata)に作成されます。ファイル名は、データベースによって一意に生成されます。データファイルは初期サイズが100MBで、無制限に自動拡張可能です。このファイルはOracle Managed Filesです。SYSAUX表領域についても同様のデータファイルが作成されます。

   LOGFILE句が指定されていないため、2つのREDOログ・グループが作成されます。各グループにはそれぞれ2つのメンバーがあり、一方のメンバーはDB_CREATE_ONLINE_LOG_DEST_1、もう一方のメンバーはDB_CREATE_ONLINE_LOG_DEST_2に作成されます。ファイル名は、データベースによって一意に生成されます。ログ・ファイルのサイズは100MBです。ログ・ファイルのメンバーはOracle Managed Filesです。

   同様に、CONTROL_FILES初期化パラメータが存在せず、2つのDB_CREATE_ONLINE_LOG_DEST_n初期化パラメータが指定されているため、2つの制御ファイルが作成されます。DB_CREATE_ONLINE_LOG_DEST_1の場所に存在する制御ファイルが主制御ファイルで、DB_CREATE_ONLINE_LOG_DEST_2の場所に存在する制御ファイルは多重化コピーです。ファイル名は、データベースによって一意に生成されます。これらはOracle Managed Filesです。サーバー・パラメータ・ファイルがある場合、CONTROL_FILES初期化パラメータが生成されます。

   自動UNDO管理モードが指定されていますが、UNDO表領域が指定されておらず、DB_CREATE_FILE_DEST初

期化パラメータが設定されているため、UNDOTBSという名前のデフォルトUNDO表領域がDB_CREATE_FILE_DESTによって指定されたディレクトリに作成されます。データファイルは20MBで、自動拡張可能です。これはOracle Managed Filesです。

最後に、dflttmpという名前のデフォルト一時表領域が指定されています。DB_CREATE_FILE_DESTがパラメータ・ファイルに含まれているため、dflttmpの一時ファイルがそのパラメータによって指定されたディレクトリに作成されます。一時ファイルは100MBで、無制限に自動拡張可能です。これはOracle Managed Filesです。

作成されたファイルを、生成ファイル名によるファイル・ツリーで表現すると次のようになります。

```
/u01
     /oradata
          /SAMPLE
               /datafile
                    /o1_mf_system_cmr7t30p_.dbf
                    /o1_mf_sysaux_cmr7t88p_.dbf
                    /o1_mf_sys_undo_2ixfh90q_.dbf
                    /o1_mf_dflttmp_157se6ff_.tmp
/u02
     /oradata
          /SAMPLE
               /onlinelog
                    /o1_mf_1_0orrm31z_.log
                    /o1_mf_2_2xyz16am_.log
               /controlfile
                    /o1_mf_cmr7t30p_.ctl
/u03
     /oradata
          /SAMPLE
               /onlinelog
                    /o1_mf_1_ixfvm8w9_.log
                    /o1_mf_2_q89tmp28_.log
               /controlfile
                    /o1_mf_x1sr8t36_.ctl
```

内部的に生成されたファイル名は、通常のビューを選択して表示できます。たとえば:

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
----------------------------------------------------
/u01/oradata/SAMPLE/datafile/o1_mf_system_cmr7t30p_.dbf
/u01/oradata/SAMPLE/datafile/o1_mf_sysaux_cmr7t88p_.dbf
/u01/oradata/SAMPLE/datafile/o1_mf_sys_undo_2ixfh90q_.dbf
3 rows selected
```

3. 制御ファイルの管理

データベースの作成時に制御ファイルが作成され、パラメータ・ファイルにCONTROL_FILES初期化パラメータが追加されました。必要に応じて、DBAはCREATE CONTROLFILE文を使用して、データベース用の制御ファイルを再作成したり、新しい制御ファイルを作成できます。

DATAFILE句およびLOGFILE句には、正しいOracle Managed Filesのファイル名を指定する必要があります。ALTER DATABASE BACKUP CONTROLFILE TO TRACE文は、正しいファイル名を含むスクリプトを生成します。また、ファイル名は、V$DATAFILE、V$TEMPFILEおよびV$LOGFILEビューを選択して確認することもできます。次の例では、サンプル・データベースの制御ファイルを再作成しています。

```
CREATE CONTROLFILE REUSE
  DATABASE sample
  LOGFILE
    GROUP 1('/u02/oradata/SAMPLE/onlinelog/o1_mf_1_0orrm31z_.log',
            '/u03/oradata/SAMPLE/onlinelog/o1_mf_1_ixfvm8w9_.log'),
```

```
  GROUP 2('/u02/oradata/SAMPLE/onlinelog/o1_mf_2_2xyz16am_.log',
          '/u03/oradata/SAMPLE/onlinelog/o1_mf_2_q89tmp28_.log')
NORESETLOGS
DATAFILE '/u01/oradata/SAMPLE/datafile/o1_mf_system_cmr7t30p_.dbf',
         '/u01/oradata/SAMPLE/datafile/o1_mf_sysaux_cmr7t88p_.dbf',
         '/u01/oradata/SAMPLE/datafile/o1_mf_sys_undo_2ixfh90q_.dbf',
         '/u01/oradata/SAMPLE/datafile/o1_mf_dflttmp_157se6ff_.tmp'
MAXLOGFILES 5
MAXLOGHISTORY 100
MAXDATAFILES 10
MAXINSTANCES 2
ARCHIVELOG;
```

この文で作成される制御ファイルは、データベースを作成したときに生成されたCONTROL_FILES初期化パラメータの指定どおりに配置されます。REUSE句が指定されているので、既存のファイルがすべて上書きされます。

4. REDOログの管理

REDOログ・ファイルの新しいグループを作成するには、DBAがALTER DATABASE ADD LOGFILE文を使用します。次の文は、DB_CREATE_ONLINE_LOG_DEST_1とDB_CREATE_ONLINE_LOG_DEST_2にメンバーを持つログ・ファイルを追加します。これらのファイルは、Oracle Managed Filesです。

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

ログ・ファイルのメンバーは、完全なファイル名を指定することにより、引き続き追加および削除できます。

GROUP句を使用して、ログ・グループを削除できます。次の例では、Oracle Managed Filesのログ・ファイルの各メンバーに対応するオペレーティング・システム・ファイルが自動的に削除されます。

```
SQL> ALTER DATABASE DROP LOGFILE GROUP 3;
```

5. 表領域の管理

sampleデータベースで今後表領域を作成する際、すべてのデータファイルがデフォルトで配置される記憶域は、DB_CREATE_FILE_DEST初期化パラメータで指定された場所(この使用例では/u01/oradata)です。ファイル名を指定せずにデータファイルを作成すると、そのファイルは初期化パラメータDB_CREATE_FILE_DESTで指定されたファイル・システムに配置されます。たとえば:

```
SQL> CREATE TABLESPACE tbs_1;
```

この文は、/u01/oradataを記憶域とする表領域を作成します。作成されるデータファイルは初期サイズが100MBで、無制限に自動拡張可能です。このデータファイルはOracle Managed Filesです。

表領域を削除すると、その表領域に対応するOracle Managed Filesも自動的に削除されます。次の文は、表領域とその格納に使用されているすべてのOracle Managed Filesを削除します。

```
SQL> DROP TABLESPACE tbs_1;
```

最初のデータファイルが一杯になっても、新しいデータファイルは自動的に作成されません。別のOracle Managed Filesのデータファイルを追加することによって、表領域を拡張できます。次の文は、DB_CREATE_FILE_DESTで指定された場所に別のデータファイルを追加します。

```
SQL> ALTER TABLESPACE tbs_1 ADD DATAFILE;
```

デフォルトのファイル・システムを変更するには、初期化パラメータを変更します。これによって既存のデータファイルが変更されることはありません。その後の作成にのみ影響します。これは次の文を使用して動的に実行できます。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='/u04/oradata';
```

6. REDO情報のアーカイブ

   REDOログ・ファイルのアーカイブは、Oracle Managed FilesとOracle Managed Files以外のファイルの間で違いはありません。アーカイブREDOログ・ファイルのファイル・システム上のアーカイブ先は、LOG_ARCHIVE_DEST_n初期化パラメータで指定できます。ファイル名は、LOG_ARCHIVE_FORMATパラメータまたはそのデフォルトに基づいて生成されます。アーカイブ・ログはOracle Managed Filesではありません。

7. バックアップ、リストアおよびリカバリ

   Oracle Managed Filesは標準オペレーティング・システム・ファイルと互換性があるため、オペレーティング・システム・ユーティリティを使用してバックアップまたはリストアを実行できます。データベースのバックアップ、リストアおよびリカバリを実行する既存の方法はすべて、Oracle Managed Filesに対しても機能します。

親トピック: Oracle Managed Filesの使用例

17.5.2 使用例2: データベース領域と高速リカバリ領域を含むデータベースの作成と管理

データベースと高速リカバリ領域の両方を含むデータベースを作成および管理する例を示します。

この使用例では、DBAが、制御ファイルおよびREDOログ・ファイルを多重化するデータベースを作成します。アーカイブ・ログとRMANによるバックアップは、高速リカバリ領域に作成されます。このデータベースの作成とメンテナンスに関するタスクは、次のとおりです。

1. 初期化パラメータの設定

   DBAは、次の汎用的なファイル作成デフォルトを設定します。

```
DB_CREATE_FILE_DEST = '/u01/oradata'
DB_RECOVERY_FILE_DEST_SIZE = 10G
DB_RECOVERY_FILE_DEST = '/u02/oradata'
LOG_ARCHIVE_DEST_1 = 'LOCATION = USE_DB_RECOVERY_FILE_DEST'
```

   DB_CREATE_FILE_DESTパラメータは、データファイル、一時ファイル、制御ファイルおよびREDOログのデフォルトのファイル・システム・ディレクトリを設定します。

   DB_RECOVERY_FILE_DESTパラメータは、制御ファイル、REDOログおよびRMANによるバックアップのデフォルトのファイル・システム・ディレクトリを設定します。

   LOG_ARCHIVE_DEST_1構成の'LOCATION=USE_DB_RECOVERY_FILE_DEST'は、アーカイブ・ログをDB_RECOVERY_FILE_DESTの場所にリダイレクトします。

   DB_CREATE_FILE_DESTパラメータとDB_RECOVERY_FILE_DESTパラメータは、ログ・ファイルおよび制御ファイル作成用のデフォルトのディレクトリを設定します。REDOログ・ファイルと制御ファイルは、2つのディレクトリの間で多重化されます。

2. データベースの作成
3. 制御ファイルの管理
4. REDOログの管理
5. 表領域の管理

   タスク2、3、4と5は使用例1と同じです。ただし、制御ファイルとREDOログは、DB_CREATE_FILE_DESTとDB_RECOVERY_FILE_DESTの場所の間で多重化されます。

6. REDOログ情報のアーカイブ

   オンライン・ログのアーカイブは、Oracle Managed FilesとOracle Managed Files以外のファイルの間には違いはありません。DB_RECOVERY_FILE_DESTで作成されるアーカイブ・ログは、Oracle Managed Filesです。

7. バックアップ、リストアおよびリカバリ

   Oracle Managed Filesは標準オペレーティング・システム・ファイルと互換性があるため、オペレーティング・システム・ユーティリティを使用してバックアップまたはリストアを実行できます。データベースのバックアップ、リストアおよびリカバリを実行する既存の方法はすべて、Oracle Managed Filesに対しても機能します。フォーマット・オプションが指定されていない場合、RMANによるすべてのディスクのバックアップは、DB_RECOVERY_FILE_DESTの場所に作成されます。バックアップはOracle Managed Filesです。

親トピック: Oracle Managed Filesの使用例

17.5.3 使用例3: 既存のデータベースへのOracle Managed Filesの追加

既存データベースにOracle Managed Filesを追加する例を示します。

この例では、Oracle Managed Filesが含まれていない既存のデータベースに対して、DBAがOracle Managed Filesを含む新しい表領域を作成し、/u03/oradataディレクトリにその表領域を配置しようとしていると想定しています。

1. 初期化パラメータの設定

   データファイルの自動作成を可能にするために、DB_CREATE_FILE_DEST初期化パラメータを、データファイルを作成するファイル・システム・ディレクトリに設定します。これは、次のように動的に実行できます。

   ```
   SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u03/oradata';
   ```

2. 表領域の作成

   DB_CREATE_FILE_DESTの設定が完了すると、CREATE TABLESPACE文からDATAFILE句を省略できます。データファイルは、DB_CREATE_FILE_DESTで指定された場所にデフォルトで作成されます。たとえば:

   ```
   SQL> CREATE TABLESPACE tbs_2;
   ```

   tbs_2表領域を削除すると、データファイルが自動的に削除されます。

親トピック: Oracle Managed Filesの使用例

第III部 スキーマ・オブジェクト

Oracle Databaseでスキーマ・オブジェクトを作成および管理できます。

- スキーマ・オブジェクトの管理
  Oracle Databaseで複数のタイプのスキーマ・オブジェクトを作成および管理できます。
- スキーマ・オブジェクトの領域の管理
  スキーマ・オブジェクトの領域の管理には、表領域アラートと領域割当ての管理、未使用の領域の再利用、未使用のオブジェクト記憶領域の削除、領域使用量の監視および容量計画などのタスクが含まれます。
- 表の管理
  表の管理には、表の作成、表のロード、表の変更および表の削除などのタスクが含まれます。
- 索引の管理
- クラスタの管理
  クラスタを使用してパフォーマンスを改善し、必要なディスク領域を削減できます。
- ハッシュ・クラスタの管理
  ハッシュ・クラスタによってデータ取得のパフォーマンスを改善できます。
- ビュー、順序およびシノニムの管理
  Oracle Databaseでビュー、順序およびシノニムを作成および管理できます。
- 破損データの修復
  データ・ブロックの破損を検出して修正できます。

18 スキーマ・オブジェクトの管理

Oracle Databaseで複数のタイプのスキーマ・オブジェクトを作成および管理できます。

- 一度の操作で複数の表やビューを作成する方法
  CREATE SCHEMA文を使用すると、一度の操作で複数の表やビューを作成し、権限を付与できます。個々の表やビューの作成が失敗したり、権限の付与が失敗したりすると、文全体がロールバックされます。オブジェクトは作成されず、権限も付与されません。
- 表、索引およびクラスタの分析
  スキーマ・オブジェクトに関する統計を収集し、統計を分析し、スキーマ・オブジェクトを検証できます。
- 表とクラスタの切捨て
  表(またはクラスタ)は残したままで、内容が完全に空になるように、表のすべての行またはクラスタ化表のグループ内のすべての行を削除できます。たとえば、月ごとのデータが含まれている表では、各月の終わりにそのデータをアーカイブした後で、表を空にする(すべての行を削除する)必要があります。
- トリガーの使用可能および使用禁止
  データベース・トリガーはデータベースに格納されているプロシージャで、表への行の追加など、特定の状況が発生した場合に実行されます。
- 整合性制約の管理
  整合性制約とは、表の1つ以上の列に格納される値を制限するルールです。CREATE TABLE文またはALTER TABLE文に制約句を指定することにより、その制約の影響を受ける列と、制約の条件を識別できます。
- スキーマ・オブジェクトの名前変更
  複数の方法でオブジェクトの名前を変更できます。
- オブジェクト依存性の管理
  Oracle Databaseには、依存オブジェクトが参照オブジェクトに関して常に最新であることを確認する自動メカニズムが用意されています。無効なオブジェクトを手動で再コンパイルすることもできます。
- オブジェクトの名前解決の管理
  SQL文で参照されるオブジェクト名は、ピリオドで区切られた複数の断片から構成できます。Oracle Databaseはオブジェクト名を解決するために特定の処理を実行します。
- 異なるスキーマへの切替え
  ALTER SESSIONのSQL文を使用して別のスキーマに切り替えることができます。
- エディションの管理
  エディションに基づく再定義によりアプリケーションをアップグレードするアプリケーション開発者が、DBA権限を必要とするエディション関連のタスクを実行するように要求する場合があります。
- スキーマ・オブジェクト情報の表示
  Oracle DatabaseにはPL/SQLパッケージが用意されており、スキーマ・オブジェクト情報の表示に使用できるオブジェクトおよびデータ・ディクショナリ・ビューを作成したDDLを判断できます。

親トピック: スキーマ・オブジェクト

18.1 一度の操作で複数の表やビューを作成する方法

CREATE SCHEMA文を使用すると、一度の操作で複数の表やビューを作成し、権限を付与できます。個々の表やビューの作成が失敗したり、権限の付与が失敗したりすると、文全体がロールバックされます。オブジェクトは作成されず、権限も付与されません。

具体的には、CREATE SCHEMA文には、CREATE TABLE、CREATE VIEWおよびGRANT文のみを含めることができます。指定した文を発行するための権限を持っている必要があります。実際にスキーマが作成されるのではなく、スキーマはCREATE USER文を使用してユーザーを作成したときに作成されます。そのかわりに、この文はスキーマを移入します。

次の文は、2つの表とそれらのデータを結合するビューを作成します。

```
CREATE SCHEMA AUTHORIZATION scott
    CREATE TABLE dept (
        deptno NUMBER(3,0) PRIMARY KEY,
        dname VARCHAR2(15),
        loc VARCHAR2(25))
    CREATE TABLE emp (
        empno NUMBER(5,0) PRIMARY KEY,
        ename VARCHAR2(15) NOT NULL,
        job VARCHAR2(10),
        mgr NUMBER(5,0),
        hiredate DATE DEFAULT (sysdate),
        sal NUMBER(7,2),
        comm NUMBER(7,2),
        deptno NUMBER(3,0) NOT NULL
        CONSTRAINT dept_fkey REFERENCES dept)
   CREATE VIEW sales_staff AS
        SELECT empno, ename, sal, comm
        FROM emp
        WHERE deptno = 30
        WITH CHECK OPTION CONSTRAINT sales_staff_cnst
        GRANT SELECT ON sales_staff TO human_resources;
```

CREATE SCHEMA文は、STORAGE句など、ANSIのCREATE TABLE文とCREATE VIEW文を拡張したOracle Database独自の機能をサポートしていません。

関連項目:

CREATE SCHEMA文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: スキーマ・オブジェクトの管理

18.2 表、索引およびクラスタの分析

スキーマ・オブジェクトに関する統計を収集し、統計を分析し、スキーマ・オブジェクトを検証できます。

- 表、索引およびクラスタの分析について
  スキーマ・オブジェクトに関する統計を収集し、その情報を分析できます。
- DBMS_STATSを使用した表および索引統計の収集
  DBMS_STATSパッケージまたはANALYZE文を使用して、表、索引またはクラスタの物理記憶特性の統計を収集できます。これらの統計はデータ・ディクショナリに格納され、オプティマイザで使用して、分析対象オブジェクトにアクセスするSQL文に最も効率的な実行計画を選択できます。
- 表、索引、クラスタおよびマテリアライズド・ビューの妥当性チェック
  表、索引、クラスタまたはマテリアライズド・ビューの構造の整合性を検証するには、VALIDATE STRUCTUREオプションを指定したANALYZE文を使用します。
- 問合せによる表および索引の相互検証
  ANALYZE文の完了までかなりの時間がかかる場合があります。そのような場合は、SQL問合せを使用して索引を検証できます。
- 表とクラスタの連鎖行のリスト

表またはクラスタの連鎖行と移行行は、LIST CHAINED ROWS句を指定したANALYZE文を使用して検出できます。この文の結果は、LIST CHAINED ROWS句によって返される情報を受け入れるために明示的に作成した指定の表に格納されます。この結果は、行を更新するための領域が十分であるかどうかを判断するうえで役立ちます。

親トピック: スキーマ・オブジェクトの管理

18.2.1 表、索引およびクラスタの分析について

スキーマ・オブジェクトに関する情報を収集して、その情報を分析できます。

次の目的でスキーマ・オブジェクト(表、索引またはクラスタ)を分析します。

- 統計の収集と管理
- 記憶形式の妥当性の検証
- 表またはクラスタの移行行と連鎖行の識別

ノート:

オプティマイザ統計の収集に、ANALYZE で COMPUTE 句および ESTIMATE 句を使用しないでください。これらの句は非推奨になりました。かわりに、DBMS_STATS パッケージを使用します。このパッケージでは、パラレルでの統計の収集、パーティション化されたオブジェクトのグローバル統計の収集、その他の方法での統計収集の微調整を行うことができます。統計に依存するコストベース・オプティマイザでは、最終的には DBMS_STATS を使用して収集された統計のみが使用されます。DBMS_STATS パッケージの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

コストベース・オプティマイザに関連しない統計収集には、ANALYZE 文(DBMS_STATS ではなく)を使用する必要があります。たとえば、次のような場合です。

- VALIDATE 句、LIST CHAINED ROWS 句の使用
- 空きリスト・ブロックの情報を収集する場合

親トピック: 表、索引およびクラスタの分析

18.2.2 DBMS_STATSを使用した表および索引統計の収集

DBMS_STATSパッケージまたはANALYZE文を使用して、表、索引またはクラスタの物理記憶特性の統計を収集できます。これらの統計はデータ・ディクショナリに格納され、オプティマイザで使用して、分析対象オブジェクトにアクセスするSQL文に最も効率的な実行計画を選択できます。

オプティマイザ統計の収集には、より多様性のあるDBMS_STATSパッケージを使用することをお薦めしますが、空きブロックや平均容量など、オプティマイザに関連付けられていない統計の収集にはANALYZE文を使用する必要があります。

DBMS_STATSパッケージを使用すると、パラレル実行を利用した統計収集と統計の外部操作ができます。統計をデータ・ディクショナリ以外の表に格納し、オプティマイザに影響を与えずにその統計を操作できます。統計をデータベース間でコピーしたり、バックアップ・コピーを作成できます。

次のDBMS_STATSプロシージャにより、オプティマイザ統計を収集できます。

- GATHER_INDEX_STATS
- GATHER_TABLE_STATS
- GATHER_SCHEMA_STATS
- GATHER_DATABASE_STATS

関連項目:

- DBMS_STATSを使用してオプティマイザ統計を収集する方法は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- DBMS_STATSパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

親トピック: 表、索引およびクラスタの分析

18.2.3 表、索引、クラスタおよびマテリアライズド・ビューの妥当性チェック

表、索引、クラスタまたはマテリアライズド・ビューの構造の整合性を検証するには、VALIDATE STRUCTUREオプションを指定したANALYZE文を使用します。

構造が有効な場合、エラーは返されません。しかし、構造が破損していると、エラー・メッセージが出力されます。

たとえば、ハードウェアやその他のシステムに障害が発生した場合、索引が破損し、正しく機能しなくなる可能性があります。索引の妥当性をチェックすると、索引内のすべてのエントリが、対応付けられた表の正しい行を示しているかを確認できます。索引が破損した場合は、その索引を削除して再作成できます。

表、索引またはクラスタが破損している場合は、削除して再作成します。マテリアライズド・ビューが破損している場合は、完全リフレッシュを実行し、問題が修正されたことを確認します。問題が修正されない場合は、マテリアライズド・ビューを削除して再作成します。

次の文は、emp表を分析します。

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

CASCADEオプションを含めると、オブジェクトとすべての依存オブジェクト(索引など)の妥当性をチェックできます。次の文は、emp表と、それに対応付けられているすべての索引の妥当性をチェックします。

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

デフォルトでは、CASCADEオプションによって、完全な妥当性チェックが実行されます。この操作はリソースを消費する可能性があるため、FAST句を使用してより高速なバージョンの妥当性チェックを実行できます。このバージョンでは、最適化されたチェック・アルゴリズムを使用して破損の有無をチェックしますが、破損の詳細はレポートしません。FASTチェックで破損が検出された場合は、FAST句を指定せずにCASCADEオプションを使用すると、破損箇所を特定できます。次の文では、emp表と、それに対応付けられているすべての索引の妥当性チェックを高速に実行します。

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

高速な検証を実行しても非常に長い時間がかかる場合は、SQL問合せを使用して索引を個別に検証できます。「問合せによる表および索引の相互検証」を参照してください。

妥当性をチェックするオブジェクトに対してDMLを実行している間でも、オンラインで構造の妥当性をチェックするように指定できます。オブジェクトに影響を及ぼす実行中のDMLによって妥当性が包括的でなくなりますが、これはオンラインでANALYZEを実

行できる柔軟性によって相殺されます。次の文は、emp表と、それに対応付けられているすべての索引の妥当性をオンラインでチェックします。

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

関連項目:

ANALYZE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表、索引およびクラスタの分析

18.2.4 問合せによる表および索引の相互検証

ANALYZE文の完了には、非常に長い時間がかかる場合があります。そのような場合は、SQL問合せを使用して索引を検証できます。

問合せによって、表と索引との間に矛盾があることが確認された場合は、ANALYZE文を使用して、索引を詳細に分析できます。通常、データベース内のほとんどのオブジェクトは破損していないため、この簡単な問合せを使用して、多数の表を破損の候補として削除し、破損している可能性のある表のみでANALYZE文を使用できます。

索引を検証するには、次の問合せを実行します。

```
SELECT /*+ FULL(ALIAS) PARALLEL(ALIAS, DOP) */ SUM(ORA_HASH(ROWID))
   FROM table_name ALIAS
   WHERE ALIAS.index_column IS NOT NULL
      MINUS SELECT /*+ INDEX_FFS(ALIAS index_name)
      PARALLEL_INDEX(ALIAS, index_name, DOP) */ SUM(ORA_HASH(ROWID))
   FROM table_name ALIAS WHERE ALIAS.index_column IS NOT NULL;
```

問合せを実行するときに、次の内容を置換します。

- table_nameプレースホルダに表の名前を入力します。
- index_columnプレースホルダに索引列を入力します。
- index_nameプレースホルダに索引名を入力します。

問合せによって行が返される場合、矛盾が発生している可能性があるため、ANALYZE文を使用して診断できます。

関連項目:

ANALYZE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表、索引およびクラスタの分析

18.2.5 表とクラスタの連鎖行のリスト

表またはクラスタの連鎖行と移行行は、LIST CHAINED ROWS句を指定したANALYZE文を使用して検出できます。この文の結果は、LIST CHAINED ROWS句によって返される情報を受け入れるために明示的に作成した指定の表に格納されます。この結果は、行を更新するための領域が十分であるかどうかを判断するうえで役立ちます。

- CHAINED_ROWS表の作成
  ANALYZE...LIST CHAINED ROWS文によって返されるデータを格納する表を作成するには、UTLCHAIN.SQLまたはUTLCHN1.SQLスクリプトを実行します。

- 表内の移行行または連鎖行の解消
  CHAINED_ROWS表の情報を使用すると、既存表内にある移行行と連鎖行を低減または解消できます。

親トピック: 表、索引およびクラスタの分析

18.2.5.1 CHAINED_ROWS表の作成

ANALYZE...LIST CHAINED ROWS文によって返されるデータを格納する表を作成するには、UTLCHAIN.SQLまたはUTLCHN1.SQLスクリプトを実行します。

これらのスクリプトは、データベースに付属しています。これらのスクリプトは、スクリプトを実行するユーザーのスキーマ内にCHAINED_ROWSという名前の表を作成します。

ノート:

CHAINED_ROWS 表を作成するためにどちらのスクリプトを実行するかは、データベースの互換性レベルと分析する表のタイプによって決まります。詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

CHAINED_ROWS表を作成した後、ANALYZE文のINTO句にその表を指定します。たとえば、次の文は、CHAINED_ROWS表に、emp_deptクラスタ内の連鎖行に関する情報を含む行を挿入します。

```
ANALYZE CLUSTER emp_dept LIST CHAINED ROWS INTO CHAINED_ROWS;
```

関連項目:

- CHAINED_ROWS表の詳細は、『Oracle Databaseリファレンス』を参照してください
- 過剰な行連鎖のある表についてのセグメント・アドバイザのレポート方法の詳細は、「セグメント・アドバイザの使用」を参照してください。

親トピック: 表とクラスタの連鎖行のリスト

18.2.5.2 表内の移行行または連鎖行の解消

CHAINED_ROWS表の情報を使用すると、既存表内にある移行行と連鎖行を低減または解消できます。

次の手順を実行します。

1. ANALYZE文を使用して、移行行と連鎖行に関する情報を収集します。

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

2. 出力表を問い合せます。

   ```
   SELECT *
   FROM CHAINED_ROWS
   WHERE TABLE_NAME = 'ORDER_HIST';
   OWNER_NAME  TABLE_NAME  CLUST... HEAD_ROWID          TIMESTAMP
   ----------  ----------  -----... ------------------  ---------
   SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAA  04-MAR-96
   SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAB  04-MAR-96
   SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAC  04-MAR-96
   ```

   移行行または連鎖行がすべてリストされます。

3. 出力表の問合せによって、移行行または連鎖行が多数存在することがわかれば、以降のステップを実行して移行行を解消します。
4. 既存表と同じ列を持つ中間表を作成して、移行行と連鎖行を格納します。

```
CREATE TABLE int_order_hist
   AS SELECT *
      FROM order_hist
      WHERE ROWID IN
         (SELECT HEAD_ROWID
            FROM CHAINED_ROWS
            WHERE TABLE_NAME = 'ORDER_HIST');
```

5. 既存表から移行行と連鎖行を削除します。

```
DELETE FROM order_hist
   WHERE ROWID IN
      (SELECT HEAD_ROWID
         FROM CHAINED_ROWS
         WHERE TABLE_NAME = 'ORDER_HIST');
```

6. 中間表の行を既存表に挿入します。

```
INSERT INTO order_hist
   SELECT *
   FROM int_order_hist;
```

7. 中間表を削除します。

```
DROP TABLE int_order_history;
```

8. ステップ1で収集した情報を出力表から削除します。

```
DELETE FROM CHAINED_ROWS
   WHERE TABLE_NAME = 'ORDER_HIST';
```

9. 再度ANALYZE文を使用してから、出力表を問い合せます。

出力表に表示された行は連鎖しています。連鎖行を解消するには、データ・ブロックのサイズを大きくする以外にありません。すべての状況において連鎖を回避することはほぼ不可能です。多くの場合、LONG列や大きいCHAR列またはVARCHAR2列を持つ表では、連鎖の発生は避けられません。

親トピック: 表とクラスタの連鎖行のリスト

18.3 表とクラスタの切捨て

表(またはクラスタ)は残したままで、内容が完全に空になるように、表のすべての行またはクラスタ化表のグループ内のすべての行を削除できます。たとえば、月ごとのデータが含まれている表では、各月の終わりにそのデータをアーカイブした後で、表を空にする(すべての行を削除する)必要があります。

- DELETEを使用した表の切捨て
  DELETEのSQL文を使用して表の行を削除できます。
- DROPおよびCREATEを使用した表の切捨て
  表を削除してから、再作成して表を切り捨てることができます。
- TRUNCATEの使用
  TRUNCATE文を使用して表のすべての行を削除できます。

親トピック: スキーマ・オブジェクトの管理

18.3.1 DELETEを使用した表の切捨て

DELETEのSQL文を使用して表の行を削除できます。

たとえば、次の文はemp表からすべての行を削除します。

```
DELETE FROM emp;
```

DELETE文を使用するときに、表またはクラスタに多数の行が存在していると、それらの行を削除する際に相当のシステム・リソースが使用されます。たとえば、CPU時間、その表と対応付けられた索引のREDOログ領域、UNDOセグメント領域などのリソースが必要です。また、各行が削除されるときに、トリガーが起動される場合があります。結果的に空になる表またはクラスタに事前に割り当てられた領域は、行を削除してもそのオブジェクトに対応付けられたままです。DELETEを使用すると削除する行を選択できますが、TRUNCATEとDROPの場合はオブジェクト全体が削除されます。

関連項目:

DELETE文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表とクラスタの切捨て

18.3.2 DROPおよびCREATEを使用した表の切捨て

表を削除してから、再作成して表を切り捨てることができます。

たとえば、次の例では、emp表を削除してから再作成しています。

```
DROP TABLE emp;
CREATE TABLE emp ( ... );
```

表やクラスタを削除してから再作成すると、対応付けられた索引、整合性制約およびトリガーもすべて削除され、削除された表またはクラスタ化表に依存するオブジェクトはすべて無効になります。また、削除された表またはクラスタ化表に対する権限付与もすべて削除されます。

親トピック: 表とクラスタの切捨て

18.3.3 TRUNCATEの使用

TRUNCATE文を使用して表のすべての行を削除できます。

たとえば、次の文はemp表を切り捨てます。

```
TRUNCATE TABLE emp;
```

TRUNCATE文を使用すると、表またはクラスタからすべての行を高速かつ効率的に削除できます。TRUNCATE文ではロールバック情報は生成されず、即時にコミットが実行されます。これはDDL文であるため、ロールバックはできません。TRUNCATE文は、切り捨てられる表に関連付けられた構造(制約およびトリガー)または認可には影響を与えません。また、TRUNCATE文では、表を切り捨てた後で、表に現在割り当てられている領域を、その表を含む表領域に戻すかどうかも指定できます。

自分のスキーマにある表またはクラスタは切り捨てることができます。DROP ANY TABLEシステム権限を持っているユーザーは、どのスキーマ内の表またはクラスタでも切り捨てることができます。

親キーを含む表またはクラスタ化表を切り捨てる際は、別の表で定義されているすべての参照外部キーを事前に使用禁止にする必要があります。自己参照制約を使用禁止にする必要はありません。

TRUNCATE文によって表から行を削除する場合、表に対応付けられているトリガーは起動されません。また、TRUNCATE文は、監査が使用可能の場合でも、DELETE文に対応するどのような監査情報も生成しません。そのかわりに、発行されたTRUNCATE文に対して、単一の監査レコードが生成されます。

ハッシュ・クラスタや、ハッシュ・クラスタまたは索引クラスタ内の表を個別に切り捨てることはできません。索引クラスタを切り捨てると、そのクラスタ内のすべての表からすべての行が削除されます。個々のクラスタ化表からすべての行を削除する必要がある場合は、DELETE文を使用するか、または表を削除してから再作成してください。

TRUNCATE文には、表またはクラスタに現在割り当てられている領域が、切捨て後、その表またはクラスタを含む表領域に返されるかどうかを制御するオプションがいくつか用意されています。

これらのオプションは対応する索引にも適用されます。表またはクラスタを切り捨てると、対応付けられた索引もすべて切り捨てられます。切り捨てられた表、クラスタまたは対応付けられた索引の記憶域パラメータは、切捨て後も変わりません。

TRUNCATEのオプションは、次のとおりです。

- デフォルトのオプションDROP STORAGEは、文実行後の表に割り当てられたエクステントの数をMINEXTENTSの元の設定まで減らします。解放されたエクステントはシステムに戻され、他のオブジェクトによって使用できます。
- DROP ALL STORAGEは、セグメントを削除します。TRUNCATE TABLE文に加えて、DROP ALL STORAGEはALTER TABLE TRUNCATE (SUB)PARTITION文にも適用されます。また、このオプションは切捨て対象のパーティションに対応付けられた依存オブジェクト・セグメントも削除します。

  DROP ALL STORAGEは、クラスタには使用できません。

  ```
  TRUNCATE TABLE emp DROP ALL STORAGE;
  ```

- REUSE STORAGEを指定すると、表またはクラスタに対して現在割り当てられているすべての領域は割り当てられたままになります。たとえば、次の文はemp_deptクラスタを切り捨てて、クラスタに対してそれまでに割り当てられているすべてのエクステントを、今後の挿入と削除のためにそのまま残します。

  ```
  TRUNCATE CLUSTER emp_dept REUSE STORAGE;
  ```

関連項目:

- TRUNCATE TABLE文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- TRUNCATE CLUSTER文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 監査の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: 表とクラスタの切捨て

18.4 トリガーの使用可能および使用禁止

データベース・トリガーとは、データベースに格納されており、表に行を追加するなどの特定の条件が発生したときにアクティブ化(起動)されるプロシージャです。

トリガーを使用してデータベースの標準機能を補完することにより、データベース管理システムを高度にカスタマイズできます。たとえば、表に対するDML操作を制限するトリガーを作成して、通常の営業時間中に発行された文のみ許可できます。

- トリガーの使用可能および使用禁止について
  トリガーが起動される文を発行したときに、トリガー制限(存在する場合)がTRUEと評価された場合は、使用可能トリ

ガーによってトリガー本体が実行されます。デフォルトでは、トリガーを最初に作成したときに使用可能に設定されます。トリガーが起動される文を発行したときに、トリガー制限(存在する場合)がTRUEと評価された場合でも、使用禁止トリガーはトリガー本体を実行しません。

- トリガーを使用可能にする方法
  使用禁止のトリガーを使用可能にするには、ENABLEオプションを指定したALTER TRIGGER文を使用します。
- トリガーを使用禁止にする方法
  トリガーを使用禁止にするには、DISABLEオプションを指定したALTER TRIGGER文を使用します。

親トピック: スキーマ・オブジェクトの管理

18.4.1 トリガーの使用可能および使用禁止について

トリガーが起動される文を発行したときに、トリガー制限(存在する場合)がTRUEと評価された場合は、使用可能トリガーによってトリガー本体が実行されます。デフォルトでは、トリガーを最初に作成したときに使用可能に設定されます。トリガーが起動される文を発行したときに、トリガー制限(存在する場合)がTRUEと評価された場合でも、使用禁止トリガーはトリガー本体を実行しません。

データベース・トリガーは、表、スキーマまたはデータベースに対応付けることができます。データベース・トリガーは、次の場合に暗黙的に起動されます。

- 対応付けられている表に対してDML文(INSERT、UPDATE、DELETE)が実行されたとき
- データベースまたはスキーマ内のオブジェクトに対して、特定のDDL文(ALTER、CREATE、DROPなど)が実行されたとき
- 指定したデータベース・イベントが発生したとき(STARTUP、SHUTDOWN、SERVERERRORなど)

このリストがすべてではありません。トリガーを起動する文とデータベース・イベントの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

トリガーを作成するには、CREATE TRIGGER文を使用します。トリガーは、トリガー・イベントの前(BEFORE)、後(AFTER)またはトリガー・イベントのかわりに(INSTEAD OF)起動するように定義できます。次の文は、表scott.empに対してトリガーscott.emp_permit_changesを作成します。このトリガーは、指定されたいずれかの文が実行される前に起動します。

```
CREATE TRIGGER scott.emp_permit_changes
     BEFORE
     DELETE OR INSERT OR UPDATE
     ON scott.emp
     .
     .
     .
pl/sql block
     .
     .
     .
```

後でDROP TRIGGER文を発行し、トリガーをデータベースから削除できます。

ALTER TABLE文を使用してトリガーを使用可能または使用禁止にするには、表を所有しているか、表に対するALTERオブジェクト権限があるか、またはALTER ANY TABLEシステム権限があることが必要です。また、ALTER TRIGGER文を使用してトリガーを個別に使用可能または使用禁止にするには、トリガーを所有しているか、またはALTER ANY TRIGGERシステム権限を持っている必要があります。

関連項目:

- トリガーの詳細は、『Oracle Database概要』を参照してください。
- CREATE TRIGGER文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。
- トリガーの作成と使用の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

親トピック: トリガーの使用可能および使用禁止

18.4.2 トリガーを使用可能にする方法

使用禁止のトリガーを使用可能にするには、ENABLEオプションを指定したALTER TRIGGER文を使用します。

たとえば、inventory表に定義されているreorderという使用禁止のトリガーを使用可能にするには、次の文を入力します。

```
ALTER TRIGGER reorder ENABLE;
```

ENABLE ALL TRIGGERSオプションを指定したALTER TABLE文を使用すれば、特定の表に定義されているトリガーをすべて使用可能にできます。たとえば、inventory表に定義されているトリガーをすべて使用可能にするには、次の文を入力します。

```
ALTER TABLE inventory
    ENABLE ALL TRIGGERS;
```

関連項目:

ALTER TRIGGER文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: トリガーの使用可能および使用禁止

18.4.3 トリガーを使用禁止にする方法

トリガーを使用禁止にするには、DISABLEオプションを指定したALTER TRIGGER文を使用します。

次の条件のいずれか1つが成り立つ場合は、一時的にトリガーを使用禁止にすることを検討してください。

- トリガーの参照するオブジェクトが使用可能でない場合。
- 大規模なデータ・ロードを実行する際に、トリガーを起動せずに迅速にデータをロードする場合。
- トリガーが適用される表にデータをロードする場合。

たとえば、inventory表に定義されているトリガーreorderを使用禁止にするには、次の文を入力します。

```
ALTER TRIGGER reorder DISABLE;
```

DISABLE ALL TRIGGERSオプションを指定したALTER TABLE文を使用すれば、表に関連するトリガーをすべて同時に使用禁止にできます。たとえば、inventory表に定義されているトリガーをすべて使用禁止にするには、次の文を入力します。

```
ALTER TABLE inventory
    DISABLE ALL TRIGGERS;
```

親トピック: トリガーの使用可能および使用禁止

18.5 整合性制約の管理

整合性制約とは、表の1つ以上の列に格納される値を制限するルールです。CREATE TABLE文またはALTER TABLE文に制約句を指定することにより、その制約の影響を受ける列と、制約の条件を識別できます。

- 整合性制約の状態
  整合性制約はビジネス・ルールを適用し、無効な情報が表に入力されるのを防止します。
- 定義時の整合性制約の設定
  CREATE TABLE文またはALTER TABLE文で整合性制約を定義するときにENABLE/DISABLE句を指定して、その制約を使用可能/使用禁止、妥当性チェックあり/妥当性チェックなしの状態にできます。制約の定義時にENABLE/DISABLE句を指定しなければ、自動的にその制約は妥当性チェックありで使用可能な状態になります。
- 既存の整合性制約の変更、名前変更または削除
  ALTER TABLE文を使用して、制約を使用可能または使用禁止にする他、制約を変更または削除することもできます。制約を規定するためにUNIQUEまたはPRIMARY KEY索引が使用されている場合、その索引に対応する制約を削除または使用禁止にすると、明示的に指定しないかぎり、索引は削除されます。
- 制約チェックの遅延
  データベースが制約をチェックしたときに制約が満たされていない場合は、エラーが通知されます。制約の妥当性チェックは、トランザクションが終わるまで遅延できます。SET CONSTRAINTS文を発行すると、トランザクションの実行中、または別のSET CONSTRAINTS文によってモードが再設定されるまで、SET CONSTRAINTSモードが継続します。
- 制約例外のレポート
  制約の妥当性チェック時に例外が存在すると、エラーが返され、整合性制約は妥当性チェックなしの状態のままになります。整合性制約の例外が存在しているために文が正常に実行されない場合、文はロールバックされます。例外が存在している場合は、制約の例外をすべて更新または削除するまで、制約の妥当性はチェックできません。
- 制約情報の表示
  表の制約定義を表示し、制約で指定されている列を識別できるように、ビューのセットが用意されています。

関連項目:

- 整合性制約の詳細は、『Oracle Database概要』を参照してください。
- アプリケーションで整合性制約を使用する際の詳細と使用例は、『Oracle Database開発ガイド』を参照してください。

親トピック: スキーマ・オブジェクトの管理

18.5.1 整合性制約の状態

整合性制約はビジネス・ルールを適用し、無効な情報が表に入力されるのを防止します。

- 整合性制約の状態について
  制約は、使用可能(ENABLE)と使用禁止(DISABLE)のいずれの状態にするかを指定できます。制約が使用可能になっている場合は、データベース内でデータが入力または更新されるときにチェックが行われ、制約に従っていないデータは入力されません。制約が使用禁止になっている場合は、ルールに従っていないデータでもデータベースに入力できます。
- 制約の使用禁止について
  整合性制約で定義されたルールを強制するためには、制約が常に使用可能な必要がありますが、状況によっては制約を使用禁止にすることを検討できます。
- 制約の使用可能について

制約が使用可能になっている場合、制約に違反する行は表に挿入されません。

- 妥当性チェックなしで使用可能な制約状態について
  制約が妥当性チェックなしで使用可能な状態にある場合、それ以後の文はすべて、制約に従っているかどうかがチェックされます。ただし、表の既存データはチェックされません。
- 整合性制約の効率的な使用: 手順
  整合性制約の状態を特定の順序で使用することが重要です。

親トピック: 整合性制約の管理

18.5.1.1 整合性制約の状態について

制約は、使用可能(ENABLE)と使用禁止(DISABLE)のいずれの状態にするかを指定できます。制約が使用可能になっている場合は、データベース内でデータが入力または更新されるときにチェックが行われ、制約に従っていないデータは入力されません。制約が使用禁止になっている場合は、ルールに従っていないデータでもデータベースに入力できます。

また、表の既存データが必ず制約に従うように指定できます(VALIDATE)。逆にNOVALIDATEを指定すると、既存データが制約に従っていることは保証されません。

表に定義されている整合性制約は、次のいずれかの状態にあります。

- ENABLE、VALIDATE
- ENABLE、NOVALIDATE
- DISABLE、VALIDATE
- DISABLE、NOVALIDATE

これらの状態の意味と組合せの結果の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。ここでは、これらの結果のいくつかについて説明します。

親トピック: 整合性制約の状態

18.5.1.2 制約の使用禁止について

整合性制約で定義されたルールを強制するためには、制約が常に使用可能な必要がありますが、状況によっては制約を使用禁止にすることを検討できます。

しかし、次のような場合は、パフォーマンス上の理由から、表の整合性制約を一時的に使用禁止にすることを検討してください。

- 表に大量のデータをロードする場合
- 表に大規模な変更を加えるバッチ操作を実行する場合(たとえば、既存の番号に1000を加えてすべての従業員番号を変更する場合)
- 表を1つずつインポートまたはエクスポートする場合

これら3つの場合には、整合性制約を一時的に使用禁止にすることにより、操作のパフォーマンスを改善できます。これは、特にデータ・ウェアハウス構成に当てはまります。

制約が使用禁止である間は、その制約に違反するデータを入力できます。したがって、前述の操作を終了した後に、制約を必ず使用可能にする必要があります。

親トピック: 整合性制約の状態

18.5.1.3 制約の使用可能について

制約が使用可能になっている場合、制約に違反する行は表に挿入されません。

しかし、制約が使用禁止の場合は、制約に違反する行でも表に挿入できます。このような行を制約の例外と呼びます。制約が妥当性チェックなしで使用可能な状態にある場合、制約が使用禁止になっていた間に入力された違反データはそのまま残っています。制約を妥当性チェック済の状態にするためには、制約に違反する行を更新または削除する必要があります。

制約を使用可能にするときに、特定の整合性制約に対する例外を指定できます。「制約例外のレポート」を参照してください。制約に違反している行はすべてEXCEPTIONS表に格納され、検証できます。

親トピック: 整合性制約の状態

18.5.1.4 妥当性チェックなしで使用可能な制約状態について

制約が妥当性チェックなしで使用可能な状態にある場合、それ以後の文はすべて、制約に従っているかどうかがチェックされます。ただし、表の既存データはチェックされません。

妥当性チェックなしで使用可能な状態の制約を持つ表には、無効なデータが含まれる可能性がありますが、無効なデータを新たに追加することはできません。妥当性チェックなしで使用可能な制約は、有効なオンライン・トランザクション処理(OLTP)データをアップロードしているデータ・ウェアハウス構成で役立ちます。

制約を使用可能にする場合に、妥当性チェックは必ずしも必要ではありません。妥当性チェックなしで制約を使用可能にする方が、妥当性チェックありで制約を使用可能にするよりはるかに高速です。また、すでに使用可能になっている制約の妥当性をチェックする場合、妥当性チェック中のDMLロックは必要ありません(すでに使用禁止にした制約の妥当性をチェックする場合とは異なります)。これは、制約の規定により、妥当性チェック中に違反データが挿入されないことが保証されているためです。したがって、妥当性チェックなしで使用可能にすれば、制約を使用可能にすることによって一般に生じる停止時間を短縮できます。

親トピック: 整合性制約の状態

18.5.1.5 整合性制約の効率的な使用: 手順

整合性制約の状態を特定の順序で使用することが重要です。

整合性制約の状態を次の順序で使用したときに、最も大きな利点が得られます。

1. 使用禁止状態。
2. 操作(ロード、エクスポート、インポート)の実行。
3. 妥当性チェックなしで使用可能な状態。
4. 使用可能状態。

制約をこの順序で使用する際の利点は、次のとおりです。

- ロックが保持されません。
- すべての制約を同時に使用可能状態にすることができます。
- 制約を使用可能にする処理がパラレルで行われます。
- 表での同時アクティビティを実行できます。

親トピック: 整合性制約の状態

18.5.2 定義時の整合性制約の設定

CREATE TABLE文またはALTER TABLE文で整合性制約を定義するときにENABLE/DISABLE句を指定して、その制約

を使用可能/使用禁止、妥当性チェックあり/妥当性チェックなしの状態にできます。制約の定義時にENABLE/DISABLE句を指定しなければ、自動的にその制約は妥当性チェックありで使用可能な状態になります。

- 定義時に制約を使用禁止にする方法
  定義時に整合性制約を使用禁止にできます。
- 定義時に制約を使用可能にする方法
  定義時に整合性制約を使用可能にできます。

親トピック: 整合性制約の管理

18.5.2.1 定義時に制約を使用禁止にする方法

定義時に整合性制約を使用禁止にできます。

次のCREATE TABLE文とALTER TABLE文は、整合性制約を定義して、使用禁止にします。

```
CREATE TABLE emp (
    empno NUMBER(5) PRIMARY KEY DISABLE,   . . . ;
ALTER TABLE emp
   ADD PRIMARY KEY (empno) DISABLE;
```

整合性制約を定義して使用禁止にするALTER TABLE文は、表の行がその整合性制約に違反しているために失敗することはありません。制約のルールが施行されていないので、制約の定義が許可されます。

親トピック: 定義時の整合性制約の設定

18.5.2.2 定義時に制約を使用可能にする方法

定義時に整合性制約を使用可能にできます。

次のCREATE TABLE文とALTER TABLE文は、整合性制約を定義して、使用可能にします。

```
CREATE TABLE emp (
    empno NUMBER(5) CONSTRAINT emp.pk PRIMARY KEY,   . . . ;
ALTER TABLE emp
    ADD CONSTRAINT emp.pk PRIMARY KEY (empno);
```

整合性制約を定義して使用可能にするALTER TABLE文は、表の行が整合性制約に違反しているために失敗する場合があります。この場合、その文はロールバックされ、制約定義は格納されず、使用可能にもなりません。

UNIQUEまたはPRIMARY KEY制約を使用可能にすると、対応する索引が作成されます。

ノート:

並列性を利用できるように制約を使用可能にする効率的な手順は、「整合性制約の効率的な使用: 手順」を参照してください。

関連項目:

制約に対応付けられた索引の作成

親トピック: 定義時の整合性制約の設定

18.5.3 既存の整合性制約の変更、名前変更または削除

ALTER TABLE文では、制約を使用可能または使用禁止にする他、制約を変更または削除することもできます。制約を規定するためにUNIQUEまたはPRIMARY KEY索引が使用されている場合、その索引に対応する制約を削除または使用禁止にすると、明示的に指定しないかぎり、索引は削除されます。

使用可能な外部キーが主キーまたは一意キーを参照している間は、主キー制約または一意キー制約、または索引を使用禁止にしたり削除することはできません。

- 制約の使用禁止および使用可能
  使用可能な整合性制約を使用禁止にし、使用禁止の整合性制約を使用可能にできます。
- 制約名の変更
  ALTER TABLE...RENAME CONSTRAINT文を使用すると、表に対する既存の制約の名前を変更できます。新しい制約名には、ユーザーの既存の制約名と競合しない名前を指定する必要があります。
- 制約の削除
  整合性制約は、規定するルールが成立しなくなった場合、またはその制約が不要になった場合に削除できます。

親トピック: 整合性制約の管理

18.5.3.1 制約の使用禁止および使用可能

使用可能な整合性制約を使用禁止にし、使用禁止の整合性制約を使用可能にできます。

次の文は、使用可能状態の整合性制約を使用禁止にします。2番目の文では、対応する索引を保持するように指定しています。

```
ALTER TABLE dept
    DISABLE CONSTRAINT dname_ukey;
ALTER TABLE dept
    DISABLE PRIMARY KEY KEEP INDEX,
    DISABLE UNIQUE (dname, loc) KEEP INDEX;
```

次の文は、使用禁止状態の整合性制約を妥当性チェックなしで使用可能な状態にします。

```
ALTER TABLE dept
    ENABLE NOVALIDATE CONSTRAINT dname_ukey;
ALTER TABLE dept
    ENABLE NOVALIDATE PRIMARY KEY,
    ENABLE NOVALIDATE UNIQUE (dname, loc);
```

次の文は、使用禁止状態の整合性制約を使用可能にするか、または妥当性チェックありの状態にします。

```
ALTER TABLE dept
    MODIFY CONSTRAINT dname_key VALIDATE;
ALTER TABLE dept
    MODIFY PRIMARY KEY ENABLE NOVALIDATE;
```

次の文は、使用禁止状態の整合性制約を使用可能にします。

```
ALTER TABLE dept
    ENABLE CONSTRAINT dname_ukey;
ALTER TABLE dept
    ENABLE PRIMARY KEY,
    ENABLE UNIQUE (dname, loc);
```

UNIQUEキーまたはPRIMARY KEY制約、およびすべての依存するFOREIGN KEY制約を一度に使用禁止または削除するには、DISABLE句またはDROP句のCASCADEオプションを使用します。たとえば、次の文はPRIMARY KEY制約とこれに依

存するFOREIGN KEY制約を使用禁止にします。

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

親トピック: 既存の整合性制約の変更、名前変更または削除

18.5.3.2 制約名の変更

ALTER TABLE...RENAME CONSTRAINT文を使用すると、表に対する既存の制約の名前を変更できます。新しい制約名には、ユーザーの既存の制約名と競合しない名前を指定する必要があります。

次の文は、表deptに対するdname_ukey制約の名前を変更します。

```
ALTER TABLE dept
    RENAME CONSTRAINT dname_ukey TO dname_unikey;
```

制約名を変更しても、実表に対するすべての依存性は引き続き有効です。

RENAME CONSTRAINT句を使用すると、制約のシステム生成名を変更できます。

親トピック: 既存の整合性制約の変更、名前変更または削除

18.5.3.3 制約の削除

整合性制約は、規定するルールが成立しなくなった場合、またはその制約が不要になった場合に削除できます。

制約を削除するには、ALTER TABLE文で次のいずれかの句を指定します。

- DROP PRIMARY KEY
- DROP UNIQUE
- DROP CONSTRAINT

次の2つの文は、整合性制約を削除します。2番目の文は、PRIMARY KEY制約に対応する索引を保持します。

```
ALTER TABLE dept
    DROP UNIQUE (dname, loc);
ALTER TABLE emp
    DROP PRIMARY KEY KEEP INDEX
    DROP CONSTRAINT dept_fkey;
```

FOREIGN KEYがUNIQUEまたはPRIMARY KEYを参照している場合は、DROP文にCASCADE CONSTRAINTS句を指定しないかぎり、制約を削除できません。

親トピック: 既存の整合性制約の変更、名前変更または削除

18.5.4 制約チェックの遅延

データベースが制約をチェックしたときに制約が満たされていない場合は、エラーが通知されます。制約の妥当性チェックは、トランザクションが終わるまで遅延できます。SET CONSTRAINTS文を発行すると、トランザクションの実行中、または別のSET CONSTRAINTS文によってモードが再設定されるまで、SET CONSTRAINTSモードが継続します。

ノート:

- SET CONSTRAINT 文は、トリガーの内部では発行できません。

- 遅延可能な一意キーと主キーは、必ず非一意索引を使用する必要があります。

- すべての制約を遅延に設定する方法
  トランザクションの制約を遅延する必要がある場合、データ操作に使用するアプリケーションでは、実際にデータの処理を始める前にすべての制約を遅延に設定する必要があります。
- コミットのチェック(オプション)
  COMMITの発行直前にSET CONSTRAINTS ALL IMMEDIATE文を発行することにより、制約違反をチェックできます。

親トピック: 整合性制約の管理

18.5.4.1 すべての制約を遅延に設定する方法

トランザクションの制約を遅延する必要がある場合、データ操作に使用するアプリケーションでは、実際にデータの処理を始める前にすべての制約を遅延に設定する必要があります。

遅延可能制約をすべて遅延に設定するには、次のDML文を使用します。

```
SET CONSTRAINTS ALL DEFERRED;
```

ノート:

SET CONSTRAINTS 文は、現行のトランザクションにのみ適用されます。制約を作成したときに指定したデフォルトは、その制約が存在するかぎり保持されています。ALTER SESSION SET CONSTRAINTS 文は、現行のセッションにしか適用されません。

親トピック: 制約チェックの遅延

18.5.4.2 コミットのチェック(オプション)

COMMITの発行直前にSET CONSTRAINTS ALL IMMEDIATE文を発行することにより、制約違反をチェックできます。

制約になんらかの問題があると、この文は失敗し、エラーの原因となっている制約が識別されます。制約違反のままコミットすると、トランザクションはロールバックされ、エラー・メッセージが返されます。

親トピック: 制約チェックの遅延

18.5.5 制約例外のレポート

制約の妥当性チェック時に例外が存在すると、エラーが返され、整合性制約は妥当性チェックなしの状態のままになります。整合性制約の例外が存在しているために文が正常に実行されない場合、文はロールバックされます。例外が存在している場合は、制約の例外をすべて更新または削除するまで、制約の妥当性はチェックできません。

整合性制約に違反している行を判断するには、ENABLE句にEXCEPTIONSオプションを指定してALTER TABLE文を発行します。EXCEPTIONSオプションにより、例外を含むすべての行の行ID、表所有者、表名および制約名が指定した表に格納されます。

制約を使用可能にする前に、ENABLE句のEXCEPTIONSオプションからの情報を格納する適切な例外レポート表を作成する必要があります。例外表を作成するには、UTLEXCPT.SQLスクリプトまたはUTLEXPT1.SQLスクリプトを実行します。

ノート:

EXCEPTIONS 表を作成するためにどちらのスクリプトを実行するかは、分析する表のタイプによって決まります。詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

これらのスクリプトのどちらを使用しても、EXCEPTIONSという名前の表が作成されます。また、スクリプトを変更して再実行すると、新たに別の名前の例外表を作成できます。

次の文は、dept表のPRIMARY KEYを検証します。例外が存在すると、EXCEPTIONS表に情報が挿入されます。

```
ALTER TABLE dept ENABLE PRIMARY KEY EXCEPTIONS INTO EXCEPTIONS;
```

dept表に重複する主キー値が存在し、deptのPRIMARY KEY制約の名前がsys_c00610である場合は、次の問合せによって例外が表示されます。

```
SELECT * FROM EXCEPTIONS;
```

次の例外が表示されます。

```
fROWID                 OWNER      TABLE_NAME      CONSTRAINT
------------------  ---------  --------------  -----------
AAAAZ9AABAAABvqAAB  SCOTT      DEPT            SYS_C00610
AAAAZ9AABAAABvqAAG  SCOTT      DEPT            SYS_C00610
```

次の文および結果のように、例外レポート表およびマスター表の行を結合した詳細な問合せの実行により、特定の制約に違反している実際の行を表示できます。

```
SELECT deptno, dname, loc FROM dept, EXCEPTIONS
    WHERE EXCEPTIONS.constraint = 'SYS_C00610'
    AND dept.rowid = EXCEPTIONS.row_id;
DEPTNO     DNAME              LOC
---------- --------------     -----------
10         ACCOUNTING         NEW YORK
10         RESEARCH           DALLAS
```

制約に違反している行はすべて更新するか、または制約を含む表から削除する必要があります。例外を更新する場合は、制約に違反する値を、制約を満たす値またはNULLに変更します。マスター表の行を更新または削除した後、以後取得する例外レポートとの混同を避けるために、例外レポート表の例外に対応する行は削除します。マスター表と例外レポート表を更新する文は、トランザクションの一貫性を保証するために、同じトランザクション内で実行してください。

前述の例の例外を訂正するために、次のトランザクションを発行できます。

```
UPDATE dept SET deptno = 20 WHERE dname = 'RESEARCH';
DELETE FROM EXCEPTIONS WHERE constraint = 'SYS_C00610';
COMMIT;
```

例外管理の最終的な目的は、例外レポート表の例外をすべて取り除くことにあります。

ノート:

制約が使用禁止になっている表の現在の例外を訂正している間に、他のユーザーが新しい例外を作成する文を発行する可能性があります。これを避けるには、例外を取り除く前に、制約に ENABLE NOVALIDATE のマークを付けます。

関連項目:

EXCEPTIONS表の詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 整合性制約の管理

18.5.6 制約情報の表示

表の制約定義を表示し、制約で指定されている列を識別できるように、ビューのセットが用意されています。

ビュー	説明
DBA_CONSTRAINTS ALL_CONSTRAINTS USER_CONSTRAINTS	DBA ビューには、データベース内のすべての制約定義が表示されます。ALL ビューには、現行ユーザーがアクセス可能な制約定義が表示されます。USER ビューには、現行ユーザーが所有している制約定義が表示されます。
DBA_CONS_COLUMNS ALL_CONS_COLUMNS USER_CONS_COLUMNS	DBA ビューには、制約で指定されているデータベース内のすべての列が表示されます。ALL ビューには、制約で指定されていて、現行ユーザーがアクセス可能な列のみが表示されます。USER ビューには、制約で指定されていて、現行ユーザーが所有している列のみが表示されます。

関連項目:

- *_CONSTRAINTSビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- *_CONS_COLUMNSビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 整合性制約の管理

18.6 スキーマ・オブジェクトの名前変更

オブジェクト名は複数の方法で変更できます。

オブジェクト名を変更するには、そのオブジェクトが自分のスキーマ内に存在する必要があります。スキーマ・オブジェクトは、次のいずれかの方法で名前を変更できます。

- オブジェクトを削除して再作成する。
- RENAME文を使用してオブジェクトの名前を変更する。
- ALTER ... RENAME文を使用してオブジェクトの名前を変更する(索引およびトリガーの場合)。

オブジェクトを削除して再作成する場合、そのオブジェクトに付与された権限はすべて失われます。オブジェクトを再作成するときに、再度権限を付与してください。

RENAME文を使用して、表、ビュー、順序またはそれらのプライベート・シノニムの名前を変更することもできます。RENAME文を使用すると、そのオブジェクトの整合性制約、索引および権限付与は新しい名前に引き継がれます。たとえば、次の文はsales_staffビューの名前を変更します。

```
RENAME sales_staff TO dept_30;
```

ノート:

ストアド PL/SQL プログラム・ユニット、パブリック・シノニムまたはクラスタに対しては、RENAME を使用できません。これらのオブジェクトの名前を変更するには、削除してから再作成してください。

スキーマ・オブジェクト名を変更する前に、次のような影響について検討する必要があります。

- 名前を変更されたオブジェクトに依存しているビューとPL/SQLプログラム・ユニットはすべて無効になるため、次に使用する前に再コンパイルする必要があります。
- 名前を変更されたオブジェクトのシノニムを使用すると、必ずエラーが返されます。

関連項目:

RENAME文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: スキーマ・オブジェクトの管理

18.7 オブジェクト依存性の管理

Oracle Databaseには、依存オブジェクトが参照オブジェクトに関して常に最新であることを確認する自動メカニズムが用意されています。無効なオブジェクトを手動で再コンパイルすることもできます。

- オブジェクト依存性とオブジェクトの無効化について
  スキーマ・オブジェクトには、他のオブジェクトを参照するタイプのものがあります。他のオブジェクトを参照するオブジェクトは依存オブジェクトと呼ばれ、参照されるオブジェクトは参照オブジェクトと呼ばれます。これらの参照はコンパイル時に確立され、コンパイラで解決できない場合は、コンパイル中の依存オブジェクトが「無効」とマークされます。
- DDLを使用した手動による無効なオブジェクトの再コンパイル
  単一のスキーマ・オブジェクトを手動で再コンパイルするには、ALTER文を使用します。
- PL/SQLパッケージのプロシージャを使用した手動による無効なオブジェクトの再コンパイル
  RECOMP_SERIALプロシージャは、スキーマ名引数が指定されない場合、指定されたスキーマのすべての無効なオブジェクト、またはデータベース内のすべての無効なオブジェクトを再コンパイルします。RECOMP_PARALLELプロシージャも同様ですが、複数のCPUをパラレルに使用します。

親トピック: スキーマ・オブジェクトの管理

18.7.1 オブジェクト依存性とオブジェクトの無効化について

スキーマ・オブジェクトには、他のオブジェクトを参照するタイプのものがあります。他のオブジェクトを参照するオブジェクトは依存オブジェクトと呼ばれ、参照されるオブジェクトは参照オブジェクトと呼ばれます。これらの参照はコンパイル時に確立され、コンパイラで解決できない場合は、コンパイル中の依存オブジェクトが「無効」とマークされます。

たとえば、ビューには表または他のビューを参照する問合せが含まれ、PL/SQLサブプログラムは他のサブプログラムを起動し、静的SQLを使用して表やビューを参照します。

Oracle Databaseには、依存オブジェクトが参照オブジェクトに関して常に最新であることを確認する自動メカニズムが用意されています。依存オブジェクトが作成されると、データベースによって、依存オブジェクトとその参照オブジェクト間の依存性が追跡されます。参照オブジェクトが依存オブジェクトに影響を与えるような方法で変更されると、依存オブジェクトには無効のマークが

付けられます。無効になった依存オブジェクトは、参照オブジェクトの新しい定義で再コンパイルして使用できるようにする必要があります。再コンパイルは、無効な依存オブジェクトが参照されると自動的に実行されます。

無効化はデータベース上で稼働するアプリケーションに影響を及ぼすため、スキーマ・オブジェクトを無効化する可能性がある変更を理解しておくことが重要です。ここでは、オブジェクトが無効になる仕組み、無効なオブジェクトを識別する方法、および無効なオブジェクトの妥当性をチェックする方法について説明します。

オブジェクトの無効化

通常、正常に実行中のアプリケーションでは表構造の変更やビュー定義またはストアド・プロシージャ定義の変更は行われないため、標準的な稼働中アプリケーションでビューやストアド・プロシージャが無効になることは予期されていません。表、ビューまたはPL/SQLユニットへの変更は、パッチ・スクリプトまたは非定型DDL文を使用してアプリケーションにパッチを適用するとき、またはアップグレードするときに発生するのが一般的です。パッチが適用されて参照オブジェクトのセットが変更された後、依存オブジェクトが無効のまま残る場合があります。

データベース内の無効な一連のオブジェクトを表示するには、次の問合せを使用します。

```
SELECT object_name, object_type FROM dba_objects
WHERE status = 'INVALID';
```

スキーマ・オブジェクトが無効になると、Oracle Enterprise Manager Cloud Controlのデータベース・ホームページにアラートが表示されます。

オブジェクトの無効化によって、アプリケーションは次の2つの影響を受けます。第1に、無効なオブジェクトは、再検証されるまでアプリケーションで使用できません。再検証によって、アプリケーション実行の待機時間が長くなります。無効なオブジェクトが多数ある場合は、初回実行時の待機時間が長時間になる可能性があります。第2に、プロシージャ、ファンクションまたはパッケージの無効化によって、そのプロシージャ、ファンクションまたはパッケージを同時に実行している他のセッションで例外が発生する可能性があります。アプリケーションを別のセッションで使用しているときにパッチを適用すると、アプリケーションを実行しているセッションによって、使用中のオブジェクトが無効化されたことが通知され、ORA-04061、ORA-04064、ORA-04065またはORA-04068の4つの例外のうちのいずれか1つが発生します。これらの例外は、パッチ適用後にアプリケーション・セッションを再起動して修正する必要があります。

適切なSQL文にCOMPILE句を指定して、スキーマ・オブジェクトを強制的に再コンパイルできます。詳細は、「DDLを使用した手動による無効なオブジェクトの再コンパイル」を参照してください。

無効なオブジェクトが多数存在することが判明している場合は、UTL_RECOMP PL/SQLパッケージを使用して一括再コンパイルを実行します。詳細は、「PL/SQLパッケージのプロシージャを使用した手動による無効なオブジェクトの再コンパイル」を参照してください。

次に、スキーマ・オブジェクトの無効化に関する一般的な規則をいくつか示します。

- 参照オブジェクトとその依存オブジェクトの間で、データベースは、依存関係に含まれる参照オブジェクトの要素を追跡します。たとえば、単一表のビューで表内の列のサブセットのみが選択された場合、それらの列のみが依存関係に含まれます。オブジェクトの各依存関係について、依存関係に含まれる要素の定義が変更されると(要素の削除も含む)、依存オブジェクトは無効になります。逆に、依存関係に含まれない要素の定義のみが変更された場合、依存オブジェクトは有効なままです。

  したがって、開発者がスキーマ・オブジェクトの変更時に注意することにより、多くの場合、依存オブジェクトの無効化とそれによるデータベースへの不要な追加作業の発生を回避できます。

- 依存オブジェクトは連鎖的に無効になります。オブジェクトがなんらかの理由で無効になると、そのオブジェクトのすべての依存オブジェクトがただちに無効になります。

- スキーマ・オブジェクトに対するオブジェクト権限を取り消すと、依存オブジェクトは連鎖的に無効になります。

関連項目:

スキーマ・オブジェクトの依存性の詳細は、『Oracle Database概要』を参照してください。

親トピック: オブジェクト依存性の管理

18.7.2 DDLを使用した手動による無効なオブジェクトの再コンパイル

単一のスキーマ・オブジェクトを手動で再コンパイルするには、ALTER文を使用します。

たとえば、パッケージ本体のPkg1を再コンパイルするには、次のDDL文を実行します。

```
ALTER PACKAGE pkg1 COMPILE REUSE SETTINGS;
```

関連項目:

様々なALTER文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: オブジェクト依存性の管理

18.7.3 PL/SQLパッケージのプロシージャを使用した手動による無効なオブジェクトの再コンパイル

RECOMP_SERIALプロシージャは、スキーマ名引数が指定されない場合、指定されたスキーマのすべての無効なオブジェクト、またはデータベース内のすべての無効なオブジェクトを再コンパイルします。RECOMP_PARALLELプロシージャも同様ですが、複数のCPUをパラレルに使用します。

アプリケーションのアップグレードまたはパッチの適用に続いて、無効なオブジェクトを再検証して、オンデマンドのオブジェクト再検証で生じるアプリケーションの待ち時間を回避することをお薦めします。Oracleには、オブジェクトの再検証で役立つUTL_RECOMPパッケージが用意されています。

例

次のPL/SQLブロックを実行して、データベース内の無効なオブジェクトすべてをパラレルに、依存順序に従って再検証します。

```
begin
   utl_recomp.recomp_parallel();
end;
/
```

DBMS_UTILITYパッケージを使用して、無効なオブジェクトを個別に再検証することもできます。次のスクリプトは、HRスキーマのUPDATE_SALARYプロシージャを再検証するPL/SQLブロックです。

```
begin
   dbms_utility.validate('HR', 'UPDATE_SALARY', namespace=>1);
end;
/
```

次のスクリプトは、パッケージ本体のHR.ACCT_MGMTを再検証するPL/SQLブロックです。

```
begin
   dbms_utility.validate('HR', 'ACCT_MGMT', namespace=>2);
end;
```

```
/
```

関連項目:

- UTL_RECOMPパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- DBMS_UTILITYパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: オブジェクト依存性の管理

18.8 オブジェクトの名前解決の管理

SQL文で参照されるオブジェクト名は、ピリオドで区切られた複数の断片から構成できます。Oracle Databaseはオブジェクト名を解決するために特定の処理を実行します。

ここでは、データベースでオブジェクト名を解決する方法を説明します。

1. Oracle Databaseは、SQL文で参照される名前の最初の断片を識別しようとします。たとえば、scott.empの最初の断片はscottです。断片が1つしか存在しない場合、その断片は最初の断片とみなされます。
    a. 現行スキーマ内で、オブジェクト名の最初の断片に一致するオブジェクトが検索されます。このようなオブジェクトが見つからない場合は、ステップ1.bに進みます。
    b. オブジェクト名の最初の断片に一致するパブリック・シノニムが検索されます。このようなパブリック・シノニムが見つからない場合は、ステップ1.cに進みます。
    c. データベースでは、オブジェクト名の最初の断片に一致するスキーマが検索されます。一致が見つかると、そのスキーマは完全修飾スキーマとなり、ステップ1.dに進みます。

       ステップ1.cでスキーマが見つからない場合、オブジェクトは識別されず、データベースからエラーが返されます。
    d. 完全修飾スキーマ内で、オブジェクト名の2番目の断片に一致するオブジェクトが検索されます。

       2番目の断片が、前回識別されたスキーマ内のオブジェクトに対応しない場合、または2番目の断片がない場合は、データベースはエラーを返します。
2. スキーマ・オブジェクトは修飾されています。名前の残りの断片は、見つかったオブジェクトの有効な部分に一致する必要があります。たとえば、名前がscott.emp.deptnoで、scottがスキーマとして識別され、empが表として識別された場合は、(empが表であるため)deptnoは列に対応する必要があります。また、empがパッケージとして識別された場合、deptnoはそのパッケージのパブリック定数、変数、プロシージャまたはファンクションに対応する必要があります。

分散データベースにおいて、グローバル・オブジェクト名が明示的またはシノニム内で間接的に使用されている場合、ローカル・データベースはローカルで参照を解決します。たとえば、シノニムをリモート表のグローバル・オブジェクト名として解決します。部分的に解決された文はリモート・データベースに転送され、前述の手順に従って、リモート・データベースでオブジェクトの解決が行われます。

データベースによる参照の解決方法の関係で、あるオブジェクトが、他のオブジェクトが存在しないことに依存している可能性があります。この状況が発生するのは、依存するオブジェクトが使用している参照の解析方法が、他のオブジェクトが存在しているときには異なる場合です。たとえば、次のような場合を考えてみます。

- 現時点では、companyスキーマに表empが含まれています。

- company.empに対してPUBLICシノニムempが作成され、company.empに対するSELECT権限がPUBLICロールに付与されます。
- jwardスキーマには、表またはプライベート・シノニムempは含まれていません。
- ユーザーjwardが、次の文を使用して自分のスキーマにビューを作成します。

```
CREATE VIEW dept_salaries AS
     SELECT deptno, MIN(sal), AVG(sal), MAX(sal) FROM emp
     GROUP BY deptno
     ORDER BY deptno;
```

jwardがdept_salariesビューを作成すると、empへの参照は、jward.empを表、ビューまたはプライベート・シノニムとして検索し、いずれも見つからない場合はパブリック・シノニムempとして検索して見つけることで解決されます。その結果、jward.dept_salariesは、jward.empが存在しないことと、public.empが存在することに依存していることがわかります。

ここで、jwardが次の文を使用して自分のスキーマに新しいビューempを作成するとします。

```
CREATE VIEW emp AS
     SELECT empno, ename, mgr, deptno
     FROM company.emp;
```

jward.empの構造がcompany.empとは異なることに注意してください。

データベースは、オブジェクト定義内の参照を解決しようとする際に、新規の依存オブジェクトの、「存在しない」オブジェクト(存在していたとすると、オブジェクトの定義の解析を変えてしまうスキーマ・オブジェクト)への依存性に内部的に注目します。存在しないオブジェクトを後で作成する場合は、このような依存性に注意する必要があります。存在しないオブジェクトを作成する場合、依存オブジェクトを再コンパイルおよび検証できるようにすべての依存オブジェクトを無効化するとともに、依存するすべてのファンクション索引に使用禁止とマークする必要があります。

したがって、前述の例では、jward.empが作成されると、jward.dept_salariesはjward.empに依存するため無効になります。その後、jward.dept_salariesが使用されると、データベースはビューの再コンパイルを試みます。empへの参照を解決するときに、jward.empが見つかります(public.empは参照先のオブジェクトではなくなっています)。jward.empにはsal列がないため、ビューを置換するときにエラーが見つかり、ビューは無効のままになります。

要約すると、存在しないオブジェクトを後で作成する場合は、オブジェクトの解決中にチェックされる存在しないオブジェクトへの依存性を管理する必要があります。

関連項目:

分散データベースにおける名前解決の詳細は、「スキーマ・オブジェクトとデータベース・リンク」を参照してください

親トピック: スキーマ・オブジェクトの管理

18.9 異なるスキーマへの切替え

ALTER SESSIONのSQL文を使用して別のスキーマに切り替えることができます。

次の文は、現行セッションのスキーマを、この文で指定するスキーマ名に設定します。

```
ALTER SESSION SET CURRENT_SCHEMA = <schema name>
```

その後のSQL文では、修飾子が省略されている場合に、Oracle Databaseによって、このスキーマ名がスキーマ修飾子として

使用されます。また、データベースでは、指定したスキーマの一時表領域が、一時データベース・オブジェクトのソート、結合および格納に使用されます。セッションには元の権限が保持され、前述のALTER SESSION文によって余分な権限は取得されません。

次の例では、プロンプトが表示されたらパスワードを入力します。

```
CONNECT scott
ALTER SESSION SET CURRENT_SCHEMA = joe;
SELECT * FROM emp;
```

empは識別されたスキーマではないため、表名はjoeスキーマのもとで解決されます。ただし、scottがjoe.emp表の選択権限を持たない場合、scottはSELECT文を実行できません。

親トピック: スキーマ・オブジェクトの管理

18.10 エディションの管理

エディションに基づく再定義によりアプリケーションをアップグレードするアプリケーション開発者が、DBA権限を必要とするエディション関連のタスクを実行するように要求する場合があります。

- エディションおよびエディションに基づく再定義について
  エディションベースの再定義を使用すると、アプリケーションの使用中にアプリケーションのデータベース・オブジェクトをアップグレードできるため、停止時間を最小限に抑えることや解消することが可能です。これは、エディションと呼ばれるプライベート環境でデータベース・オブジェクトを変更(再定義)することによって実行します。
- エディションに基づく再定義のためのDBAのタスク
  エディションに基づく再定義に関連するタスクを行うには、ユーザーが必要な権限を持っている必要があります。
- データベースのデフォルトのエディションの設定
  データベースには必ずデフォルト・エディションがあります。これは、接続時にエディションが明示的に示されない場合にデータベース・セッションが最初に使用するエディションです。
- データベースのデフォルト・エディションの問合せ
  データベースのデフォルト・エディションはデータベース・プロパティとして格納されています。
- データベース・サービスのエディション属性の設定
  データベース・サービスの作成時にそのエディション属性を設定したり、既存のデータベース・サービスのエディション属性を変更できます。
- エディションの使用
  特定のエディションのオブジェクトを表示または変更するには、最初にそのエディションを使用する必要があります。データベースに接続したときに、使用するエディションを指定できます。エディションを指定しない場合は、データベースのデフォルトのエディションでセッションが開始されます。
- エディションのデータ・ディクショナリ・ビュー
  エディションの管理に役立つデータ・ディクショナリ・ビューがいくつかあります。

親トピック: スキーマ・オブジェクトの管理

18.10.1 エディションおよびエディションに基づく再定義について

エディションベースの再定義を使用すると、アプリケーションの使用中にアプリケーションのデータベース・オブジェクトをアップグレードできるため、停止時間を最小限に抑えることや解消することが可能です。これは、エディションと呼ばれるプライベート環境でデータベース・オブジェクトを変更(再定義)することによって実行します。

すべての変更が実行およびテストされている場合のみ、アプリケーションの新バージョンをユーザーが使用できるようにしてください。

関連項目:

エディションに基づく再定義の詳細な説明は、『Oracle Database開発ガイド』を参照してください。

親トピック: エディションの管理

18.10.2 エディションに基づく再定義のためのDBAのタスク

エディションに基づく再定義に関連するタスクを行うには、ユーザーが必要な権限を持っている必要があります。

表18-1に、通常はDBAにのみ付与される権限を必要とするエディション関連のタスクの要約を示します。DBAのロールを付与されているユーザーはこれらのタスクを実行可能です。

表18-1 エディションに基づく再定義のためのDBAのタスク

タスク	参照
エディションを作成、変更、および削除する権限の付与または取消し	CREATE EDITION および DROP EDITION SQL 文
スキーマのエディションの有効化	Oracle Database 開発ガイド
データベースのデフォルト・エディションの設定	「データベースのデフォルトのエディションの設定」
データベース・サービスのエディション属性を設定します。	「データベース・サービスのエディション属性の設定」

親トピック: エディションの管理

18.10.3 データベースのデフォルトのエディションの設定

データベースには必ずデフォルトのエディションがあります。これは、接続時にエディションが明示的に示されない場合にデータベース・セッションが最初に使用するエディションです。

データベースのデフォルト・エディションを設定するには:

1. データベースにALTER DATABASE権限とエディションに対するUSE権限WITH GRANT OPTIONを持つユーザーとして接続します。
2. 次の文を入力します。

```
ALTER DATABASE DEFAULT EDITION = edition_name;
```

関連項目:

「SQL*Plusを使用したデータベースへの接続」

親トピック: エディションの管理

18.10.4 データベースのデフォルト・エディションの問合せ

データベースのデフォルト・エディションは、データベースのプロパティとして格納されています。

データベースのデフォルト・エディションを問い合せるには:

1. 任意のユーザーとしてデータベースに接続します。
2. 次の文を入力します。

```
SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE
   PROPERTY_NAME = 'DEFAULT_EDITION';
PROPERTY_VALUE
------------------------------
ORA$BASE
```

ノート:

プロパティ名の DEFAULT_EDITION は大文字/小文字が区別されます。必ず大文字で入力してください。

親トピック: エディションの管理

18.10.5 データベース・サービスのエディション属性の設定

データベース・サービスの作成時にそのエディション属性を設定したり、既存のデータベース・サービスのエディション属性を変更できます。

ノート:

インスタンスのデータベース・サービスの数には、上限があります。この制限の詳細は、『Oracle Database リファレンス』を参照してください。

- データベース・サービスのエディション属性の設定について
  サービスのエディション属性を設定すると、クライアント接続やDBMS_SCHEDULERジョブなどサービスを指定する後続のすべての接続はこのエディションを初期セッション・エディションとして使用します。ただし、セッション接続に別のエディションが指定されている場合は、セッション接続に指定されているエディションがセッション・エディションに使用されます。
- データベース・サービス作成時のエディション属性の設定
  SRVCTLユーティリティまたはDBMS_SERVICEパッケージを使用して、サービスの作成時にデータベース・サービスのエディション属性を設定できます。
- 既存のデータベース・サービスのエディション属性の設定
  既存のデータベース・サービスのエディション属性を設定するには、SRVCTLユーティリティまたはDBMS_SERVICEパッケージを使用します。

親トピック: エディションの管理

18.10.5.1 データベース・サービスのエディション属性の設定について

サービスのエディション属性を設定すると、クライアント接続やDBMS_SCHEDULERジョブなどサービスを指定する後続のすべての接続はこのエディションを初期セッション・エディションとして使用します。ただし、セッション接続に別のエディションが指定されている場合は、セッション接続に指定されているエディションがセッション・エディションに使用されます。

データベース・サービスのエディション属性をチェックするには、ALL_SERVICESビューまたはDBA_SERVICESビューのEDITION列に問い合せます。

親トピック: データベース・サービスのエディション属性の設定

18.10.5.2 データベース・サービス作成時のエディション属性の設定

RVCTLユーティリティまたはDBMS_SERVICEパッケージを使用して、サービスの作成時にデータベース・サービスのエディション属性を設定できます。

データベース・サービスのエディション属性を設定するには、「データベース・サービスの作成」の手順に従い、適切なオプションを使用します。

- 単一インスタンス・データベースをOracle Restartで管理している場合は、SRVCTLユーティリティを使用してデータベース・サービスを作成し、-editionオプションを指定してそのエディション属性を設定します。

  この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースを対象に、crmbatchという名前で新規データベース・サービスを作成し、データベース・サービスのエディション属性をe2に設定します。

  ```
  srvctl add service -db dbcrm -service crmbatch -edition e2
  ```

- 単一インスタンス・データベースをOracle Restartで管理していない場合は、DBMS_SERVICE.CREATE_SERVICEプロシージャを使用し、editionパラメータを指定してデータベース・サービスのエディション属性を設定します。

親トピック: データベース・サービスのエディション属性の設定

18.10.5.3 既存のデータベース・サービスのエディション属性の設定

既存のデータベース・サービスのエディション属性を設定するには、SRVCTLユーティリティまたはDBMS_SERVICEパッケージを使用します。

既存のデータベース・サービスのエディション属性を設定するには:

1. データベース・サービスを停止します。
2. 適切なオプションを使用して、データベース・サービスのエディション属性を設定します。
   - 単一インスタンス・データベースをOracle Restartで管理している場合は、SRVCTLユーティリティを使用してデータベース・サービスを変更し、-editionオプションを指定してそのエディション属性を設定します。

     この例では、DB_UNIQUE_NAMEがdbcrmであるデータベースを対象に、crmbatchという名前のデータベース・サービスを変更し、データベース・サービスのエディション属性をe3に設定します。

     ```
     srvctl modify service -db dbcrm -service crmbatch -edition e3
     ```

   - 単一インスタンス・データベースをOracle Restartで管理していない場合は、DBMS_SERVICE.MODIFY_SERVICEプロシージャを使用し、editionパラメータを指定してデータベース・サービスのエディション属性を設定します。MODIFY_SERVICEプロシージャを実行するときには、modify_editionパラメータがTRUEに設定されていることを確認してください。
3. データベース・サービスを起動します。

関連項目:

- Oracle Restartを使用したデータベース・サービスの管理については、「Oracle Databaseの自動再起動の構成」を参照してください
- DBMS_SERVICEパッケージを使用したデータベース・サービスの管理の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: データベース・サービスのエディション属性の設定

18.10.6 エディションの使用

特定のエディションのオブジェクトを表示または変更するには、最初にそのエディションを使用する必要があります。データベースに接続したときに、使用するエディションを指定できます。エディションを指定しない場合は、データベースのデフォルトのエディションでセッションが開始されます。

異なるエディションを使用するには、次の文を発行します。

```
ALTER SESSION SET EDITION=edition_name;
```

次の文は、現在のエディションをe2に設定してから、ora$baseに設定します。

```
ALTER SESSION SET EDITION=e2;
...
ALTER SESSION SET EDITION=ora$base;
```

関連項目:

- エディションの使用および現在のエディションの判別方法の詳細は、『Oracle Database開発ガイド』を参照してください。
- 「SQL*Plusを使用したデータベースへの接続」

親トピック: エディションの管理

18.10.7 エディションのデータ・ディクショナリ・ビュー

エディションの管理に役立つデータ・ディクショナリ・ビューがいくつかあります。

次の表に、そのうちの3つを示します。完全なリストは、『Oracle Database開発ガイド』を参照してください。

ビュー	説明
*_EDITIONS	データベース内のすべてのエディションがリストされます。(ノート: USER_EDITIONS は存在しません。)
*_OBJECTS	現在のエディションで表示可能な(実在するか、継承されている)データベースのすべてのオブジェクトを示します。
*_OBJECTS_AE	エディション全体のデータベース内にあるすべての実在のオブジェクトを示します。

親トピック: エディションの管理

18.11 スキーマ・オブジェクト情報の表示

Oracle DatabaseにはPL/SQLパッケージが用意されており、スキーマ・オブジェクト情報の表示に使用できるオブジェクトおよびデータ・ディクショナリ・ビューを作成したDDLを判断できます。

- PL/SQLパッケージを使用したスキーマ・オブジェクト情報の表示
  オラクル社が提供するPL/SQLパッケージDBMS_METADATA.GET_DDLを使用すると、スキーマ・オブジェクトに関するメタデータを(オブジェクトの作成に使用するDDLの形式で)取得できます。
- スキーマ・オブジェクトのデータ・ディクショナリ・ビュー
  これらのビューにはスキーマ・オブジェクトに関する一般的な情報が表示されます。

親トピック: スキーマ・オブジェクトの管理

18.11.1 PL/SQLパッケージを使用したスキーマ・オブジェクト情報の表示

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

関連項目:

DBMS_METADATAパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

例: DBMS_METADATAパッケージの使用

DBMS_METADATAパッケージは、スキーマ・オブジェクトの完全な定義を取得するための強力なツールです。これにより、あるオブジェクトのすべての属性を1回のパスで取得できます。オブジェクトは、その作成(再作成)に使用できるDDLで表されます。

次の文では、GET_DDLファンクションを使用して、現行スキーマ内にあるすべての表のDDLをフェッチし、ネストした表とオーバーフロー・セグメントを除外しています。また、SQL DDLで記憶域句が返されないようにするため、SET_TRANSFORM_PARAM(ハンドル値として「現行セッション用」を意味するDBMS_METADATA.SESSION_TRANSFORMをとる)を使用してそれを指定しています。セッション・レベルの変換パラメータは、最後にデフォルトにリセットされています。変換パラメータ値は、いったん設定すると、明示的にデフォルトにリセットされるまで有効です。

```
EXECUTE DBMS_METADATA.SET_TRANSFORM_PARAM(
     DBMS_METADATA.SESSION_TRANSFORM,'STORAGE',false);
SELECT DBMS_METADATA.GET_DDL('TABLE',u.table_name)
     FROM USER_ALL_TABLES u
     WHERE u.nested='NO'
     AND (u.iot_type is null or u.iot_type='IOT');
EXECUTE DBMS_METADATA.SET_TRANSFORM_PARAM(
     DBMS_METADATA.SESSION_TRANSFORM,'DEFAULT');
```

DBMS_METADATA.GET_DDLからの出力はLONGデータ型です。SQL*Plusを使用している場合は、出力がデフォルトで切り捨てられる場合があります。出力が切り捨てられないようにするには、DBMS_METADATA.GET_DDL文を発行する前に、次のSQL*Plusコマンドを発行してください。

```
SQL> SET LONG 9999
```

親トピック: スキーマ・オブジェクト情報の表示

18.11.2 スキーマ・オブジェクトのデータ・ディクショナリ・ビュー

次のビューには、スキーマ・オブジェクトに関する一般的な情報が表示されます。

ビュー	説明
DBA_OBJECTS ALL_OBJECTS USER_OBJECTS	DBA ビューには、データベース内のすべてのスキーマ・オブジェクトが表示されます。ALL ビューには、現行ユーザーがアクセス可能なオブジェクトが表示されます。USER ビューには、現行ユーザーが所有しているオブジェクトが表示されます。
DBA_CATALOG ALL_CATALOG USER_CATALOG	データベース内にあるすべての表、ビュー、シノニムおよび順序の名前、タイプおよび所有者(USER ビューでは所有者は表示されない)がリストされます。
DBA_DEPENDENCIES ALL_DEPENDENCIES USER_DEPENDENCIES	プロシージャ、パッケージ、ファンクション、パッケージ本体およびトリガーの間の依存性(データベース・リンクを持たないビューへの依存性など)がすべてリストされます。

- 例1: スキーマ・オブジェクトのタイプ別表示
  USER_OBJECTSビューに対する問合せを実行して、問合せを発行したユーザーが所有するすべてのオブジェクトのリストを表示できます。
- 例2: ビューとシノニムの依存性の表示
  ビューまたはシノニムを作成するとき、ビューやシノニムはその基礎になるベース・オブジェクトに基づきます。ビューの依存性を明確にするには、ALL_DEPENDENCIES、USER_DEPENDENCIESおよびDBA_DEPENDENCIESデータ・ディクショナリ・ビューを使用します。

親トピック: スキーマ・オブジェクト情報の表示

18.11.2.1 例1: スキーマ・オブジェクトのタイプ別表示

USER_OBJECTSビューに対する問合せを実行して、問合せを発行したユーザーが所有するすべてのオブジェクトのリストを表示できます。

次の問合せは、問合せを発行しているユーザーが所有しているオブジェクトをすべてリストします。

```
SELECT OBJECT_NAME, OBJECT_TYPE
    FROM USER_OBJECTS;
```

問合せの出力は次のとおりです。

```
OBJECT_NAME                    OBJECT_TYPE
-----------------------------  -------------------
EMP_DEPT                       CLUSTER
EMP                            TABLE
DEPT                           TABLE
EMP_DEPT_INDEX                 INDEX
```

```
PUBLIC_EMP                     SYNONYM
EMP_MGR                        VIEW
```

親トピック: スキーマ・オブジェクトのデータ・ディクショナリ・ビュー

18.11.2.2 例2: ビューとシノニムの依存性の表示

ビューまたはシノニムを作成するとき、ビューやシノニムはその基礎になるベース・オブジェクトに基づきます。ビューの依存性を明確にするには、ALL_DEPENDENCIES、USER_DEPENDENCIESおよびDBA_DEPENDENCIESデータ・ディクショナリ・ビューを使用します。

シノニムのベース・オブジェクトのリストを表示するには、ALL_SYNONYMS、USER_SYNONYMSおよびDBA_SYNONYMSデータ・ディクショナリ・ビューを使用します。たとえば、次の問合せは、ユーザーjwardによって作成されたシノニムのベース・オブジェクトをリストします。

```
SELECT TABLE_OWNER, TABLE_NAME, SYNONYM_NAME
    FROM DBA_SYNONYMS
    WHERE OWNER = 'JWARD';
```

問合せの出力は次のとおりです。

```
TABLE_OWNER              TABLE_NAME   SYNONYM_NAME
-----------------------  -----------  -----------------
SCOTT                    DEPT         DEPT
SCOTT                    EMP          EMP
```

親トピック: スキーマ・オブジェクトのデータ・ディクショナリ・ビュー

19 スキーマ・オブジェクトの領域の管理

スキーマ・オブジェクトの領域の管理には、表領域のアラートと領域の割当て、未使用の領域の再利用、未使用のオブジェクト記憶領域の削除、領域使用量の監視および容量計画などのタスクが含まれます。

- 表領域のアラートの管理
  Oracle Databaseでは、使用可能な領域が少なくなると事前にアラートで通知されるため、表領域のディスク領域を管理するために役立ちます。
- 再開可能領域割当ての管理
  大規模なデータベース処理を一時停止して後で再開できます。
- 未使用領域の再生
  未使用領域を再利用できます。セグメント・アドバイザはOracle Databaseコンポーネントで、再利用可能な領域があるセグメントを識別します。
- 未使用オブジェクト記憶域の削除
  DBMS_SPACE_ADMINパッケージにはDROP_EMPTY_SEGMENTSプロシージャが含まれており、このプロシージャを使用すると、以前のリリースから移行された空の表およびパーティションのセグメントを削除できます。可能な場合には、索引セグメントなどの表の依存オブジェクトのセグメントもこれに含まれます。
- データ型の領域使用の理解
  表またはその他のデータ構造を作成する際には、必要となる領域の大きさを把握しておく必要があります。領域要件は、データ型ごとに異なります。
- スキーマ・オブジェクトの領域使用情報の表示
  Oracle Databaseには、スキーマ・オブジェクトの領域使用に関する情報を表示するためのデータ・ディクショナリ・ビューとPL/SQLパッケージが用意されています。
- データベース・オブジェクトの容量計画
  Oracle Databaseでは、Cloud ControlまたはDBMS_SPACE PL/SQLパッケージの2つの方法でデータベース・オブジェクトの容量を計画できます。DBMS_SPACEパッケージには、新しいオブジェクトのサイズを予測したり、既存のデータベース・オブジェクトのサイズを監視できる3つのプロシージャがあります。

親トピック: スキーマ・オブジェクト

19.1 表領域のアラートの管理

Oracle Databaseでは、使用可能な領域が少なくなると事前にアラートで通知されるため、表領域のディスク領域を管理するのに役立ちます。

- 表領域のアラートの管理について
  デフォルトで、警告およびクリティカルの2つのアラートしきい値が定義されています。警告のしきい値は、領域が残り少なくなり始める境界値です。クリティカルのしきい値は、即時に注意を喚起する必要のある深刻な境界値です。データベースは、両方のしきい値でアラートを発行します。
- アラートしきい値の設定
  各表領域には、パーセント・フルのしきい値のみ、空き領域のしきい値のみ、または同時に両方のしきい値タイプを設定できます。どちらのタイプのしきい値も、0(ゼロ)に設定すると無効になります。
- アラートの表示
  アラートを表示するには、Cloud Controlのデータベース・ホームページにアクセスし、「インシデントと問題」セクションを表示します。

- 制限事項
  しきい値ベースのアラートには、次の制限があります。

親トピック: スキーマ・オブジェクトの領域の管理

19.1.1 表領域のアラートの管理について

デフォルトで、警告およびクリティカルの2つのアラートしきい値が定義されています。警告のしきい値は、領域が残り少なくなり始める境界値です。クリティカルのしきい値は、即時に注意を喚起する必要のある深刻な境界値です。データベースは、両方のしきい値でアラートを発行します。

ローカル管理表領域とディクショナリ管理表領域の両方に対してアラートしきい値を指定するには、次の2つの方法があります。

- パーセント・フルによる方法

  警告のしきい値とクリティカルのしきい値の両方について、使用済領域が合計領域の一定割合以上になるとアラートが発行されます。

- 空き領域(KB単位)による方法

  警告のしきい値とクリティカルのしきい値の両方について、空き領域が一定容量(KB)未満になるとアラートが発行されます。空き領域のしきい値は、表領域が大規模な場合に便利です。

ローカル管理表領域のアラートはサーバーで生成されます。ディクショナリ管理表領域の場合は、Oracle Enterprise Manager Cloud Control (Cloud Control)がこの機能を提供します。詳細は、「サーバー生成アラートを使用したデータベースの監視」を参照してください。

新しい表領域には、次のようにアラートしきい値が割り当てられます。

- ローカル管理表領域—ローカル管理表領域を新規作成すると、データベースに定義されているデフォルトのしきい値がその表領域に割り当てられます。新しく作成されたデータベースには、警告のしきい値に85%使用済、クリティカルのしきい値に97%使用済のデフォルトが割り当てられます。新しいデータベースに対する空き領域のしきい値のデフォルトは、両方ともゼロ(無効)です。これらのデータベースのデフォルトは変更可能で、その手順については後で説明します。
- ディクショナリ管理表領域—ディクショナリ管理表領域を新規作成すると、Cloud Controlのメトリック・カテゴリ「表領域の空き領域(MB)(ディクショナリ管理)」および「表領域使用率(%)(ディクショナリ管理)」の「その他すべて」にリストされているしきい値が割り当てられます。これらの値は、「メトリックとポリシー設定」ページで変更できます。

> ノート:
>
> Oracle 9i 以前から Oracle Database 10g 以降にアップグレードするデータベースでは、すべてのローカル管理表領域のアラートしきい値のデータベースのデフォルトは、0(ゼロ)に設定されます。この設定は、アラート・メカニズムを事実上使用禁止にして、新しく移行されたデータベースへの過剰なアラートを回避しています。

親トピック: 表領域のアラートの管理

19.1.2 アラートしきい値の設定

各表領域には、パーセント・フルのしきい値のみ、空き領域のしきい値のみ、または同時に両方のしきい値タイプを設定できます。どちらのタイプのしきい値も、0(ゼロ)に設定すると無効になります。

理想的な警告のしきい値は、クリティカルのしきい値が発行される前に問題を解決できる時間を考慮して、早めにアラートを発

行する設定です。クリティカルのしきい値は、ただちに処理してサービスの損失を回避できるよう十分早めにアラートを発行するように設定します。

ローカル管理表領域のアラートしきい値を設定するには:

- 次のいずれかの操作を行います。
  - Cloud Controlの「表領域」ページを使用します。

    表領域の領域使用量アラートしきい値の変更の詳細は、Cloud Controlのオンライン・ヘルプを参照してください。
  - DBMS_SERVER_ALERT.SET_THRESHOLDパッケージ・プロシージャをコールします。

    詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ディクショナリ管理表領域のアラートしきい値を設定するには:

- Cloud Controlの「表領域」ページを使用します。

  表領域の領域使用量アラートしきい値の変更の詳細は、Cloud Controlのオンライン・ヘルプを参照してください。

例 - Cloud Controlを使用したアラートしきい値の設定

表領域に対する領域使用率がしきい値に達した場合、Cloud Controlはアラートを受け取ります。領域使用率のアラートは2種類あり、「警告」は、表領域の使用量が低い場合に、「クリティカル」は、表領域がほぼ一杯でアクションがすぐに必要な場合に発行されます。

警告アラートとクリティカル・アラートの両方に、次の方法でアラートしきい値を指定できます。

- 使用領域ごと(%)

  使用されている領域が全領域の一定の割合以上になった場合、アラートが発行されます。
- 空き領域ごと(MB)

  空き領域がMB単位を下回る場合、アラートが発行されます。

  空き領域のしきい値は、表領域が大きい場合ほど有用です。たとえば、10TBの表領域で限界のアラートの割合を全体の99%に設定すると、100GBの空き領域が切った場合データベースによってアラートが発行されることになります。通常、空き領域が100GBになったということは重大な状態ではないため、アラートは有効とはいえません。この表領域の場合、空き領域が5GBを切った場合に限界のアラートを発行するような設定の方が、空き領域のしきい値が有効に使用されています。

表領域に対する警告および限界のアラートの両方に関して、使用されている領域のしきい値または空き領域のしきい値のいずれか、または両方を有効にできます。

表領域の領域使用量アラートしきい値を変更するには:

1. データベース・ホームページに移動します。
2. 「管理」メニューから、「記憶域」を選択し、「表領域」を選択します。

   「表領域」ページが表示されます。
3. 変更するしきい値を持つ表領域を選択し、「編集」をクリックします。

   表領域の編集ページ、および一般サブページが表示されます。
4. ページ上部の「しきい値」タブをクリックし、しきい値サブページを表示します。

5. 「使用済領域(%)」セクションで、次のいずれかを実行します。

- デフォルトしきい値を許可します。
- 「しきい値の指定」を選択し、「警告(%)」のしきい値および「クリティカル(%)」のしきい値を入力します。
- 「しきい値の無効化」を選択し、すべてのしきい値を無効にします。

6. 「空き領域(MB)」セクションで、次のいずれかを行います。

- デフォルトしきい値を許可します。
- 「しきい値の指定」を選択し、「警告(MB)」のしきい値および「クリティカル(MB)」のしきい値を入力します。
- 「しきい値の無効化」を選択し、残りの空き領域のしきい値を無効にします。

7. 「適用」をクリックします。

確認メッセージが表示されます。

例 - パッケージ・プロシージャを使用したアラートしきい値の設定

次の例は、USERS表領域について空き領域のしきい値を10MB(警告)および2MB(クリティカル)に設定し、パーセント・フルのしきい値を無効にします。USERS表領域はローカル管理の表領域です。

```
BEGIN
DBMS_SERVER_ALERT.SET_THRESHOLD(
   metrics_id              => DBMS_SERVER_ALERT.TABLESPACE_BYT_FREE,
   warning_operator        => DBMS_SERVER_ALERT.OPERATOR_LE,
   warning_value           => '10240',
   critical_operator       => DBMS_SERVER_ALERT.OPERATOR_LE,
   critical_value          => '2048',
   observation_period      => 1,
   consecutive_occurrences => 1,
   instance_name           => NULL,
   object_type             => DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE,
   object_name             => 'USERS');
DBMS_SERVER_ALERT.SET_THRESHOLD(
   metrics_id              => DBMS_SERVER_ALERT.TABLESPACE_PCT_FULL,
   warning_operator        => DBMS_SERVER_ALERT.OPERATOR_GT,
   warning_value           => '0',
   critical_operator       => DBMS_SERVER_ALERT.OPERATOR_GT,
   critical_value          => '0',
   observation_period      => 1,
   consecutive_occurrences => 1,
   instance_name           => NULL,
   object_type             => DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE,
   object_name             => 'USERS');
END;
/
```

ノート:

パーセント・フルのしきい値に 0(ゼロ)以外の値を設定する場合は、「以上」演算子 OPERATOR_GE を使用します。

データベースのデフォルトしきい値への表領域のリストア

ローカル管理表領域のアラートしきい値に明示的に値を設定した後は、その値をDBMS_SERVER_ALERT.SET_THRESHOLDを使用してNULLに設定することで、データベースのデフォルト値に戻すことがで

きます。

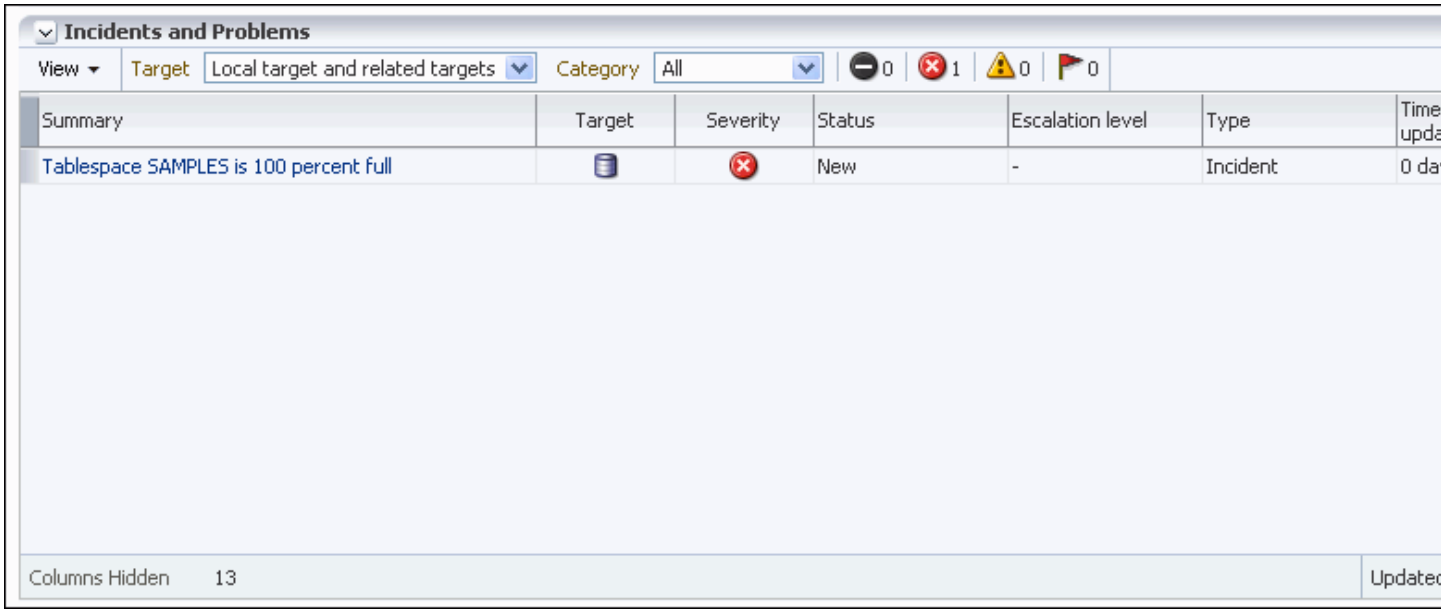
データベースのデフォルトしきい値の変更

ローカルに管理されている表領域のデータベースのデフォルトしきい値を変更するには、前の例に示されているように`DBMS_SERVER_ALERT.SET_THRESHOLD`を起動しますが、`object_name`を`NULL`に設定します。データベースのデフォルトを使用するすべての表領域で、新しいデフォルトに切り替わります。

親トピック: 表領域のアラートの管理

19.1.3 アラートの表示

アラートを表示するには、Cloud Controlのデータベース・ホームページにアクセスし、「インシデントと問題」セクションを表示します。



ローカル管理表領域のアラートは、`DBA_OUTSTANDING_ALERTS`ビューを使用して表示することもできます。詳細は、「サーバー生成アラートのデータ・ディクショナリ・ビュー」を参照してください。

親トピック: 表領域のアラートの管理

19.1.4 制限事項

しきい値ベースのアラートには、いくつかの制限があります。

これらの制限には、次のものが含まれます。

- アラートは、オフラインまたは読取り専用モードのローカル管理表領域については発行されません。ただし、それらの表領域が読取り/書込みモードまたは使用可能になると、そのアラート・システムは再アクティブ化されます。
- 表領域をオフライン化または読取り専用モードにする場合は、しきい値を0(ゼロ)に設定して、表領域に対するアラートを使用禁止にする必要があります。後で表領域を再度オンライン化または読取り/書込みモードにする場合は、しきい値を再設定してアラートを再び使用可能にできます。

関連項目:

- サーバー生成アラートに関する一般的な詳細は、「サーバー生成アラートを使用したデータベースの監視」を参照してく

ださい

- DBMS_SERVER_ALERTパッケージのプロシージャとその使用方法の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- 表領域内で使用されていない領域を再生する方法は、「未使用領域の再生」を参照してください
- リサイクル・ビン領域を再生する方法は、「リサイクル・ビン内のオブジェクトのパージ」を参照してください

親トピック: 表領域のアラートの管理

19.2 再開可能領域割当ての管理

大規模なデータベース処理を一時停止して後で再開できます。

- 再開可能領域割当ての概要
  Oracle Databaseでは、領域割当てが失敗した場合に大規模なデータベース処理を一時停止して、後で再開するための方法が提供されています。そのため、Oracle Databaseサーバーがユーザーにエラーを返すかわりに対処措置を講じることができます。エラー条件が訂正されると、一時停止していた処理が自動的に再開します。この機能のことを、再開可能領域割当てと呼びます。また、影響を受ける文のことを、再開可能文と呼びます。
- 再開可能領域割当ての有効化および無効化について
  SQL文を実行して特定の初期化パラメータを設定し、再開可能領域割当てを有効または無効にできます。
- LOGONトリガーを使用したデフォルト再開可能モードの設定
  RESUMABLE_TIMEOUT初期化パラメータを設定する以外に、デフォルトの再開可能モードを設定するもう1つの方法は、データベース・レベルのLOGONトリガーを登録して、再開可能を有効化してタイムアウト間隔を設定するように、ユーザーのセッションを変更する方法です。
- 一時停止文の検出
  再開可能文が一時停止するとき、クライアントにはエラーは通知されません。訂正処理を実行するため、Oracle Databaseではユーザーにエラーを通知して状況に関する情報を提供するかわりの手段が提供されています。
- 操作一時停止アラート
  再開可能セッションが一時停止されると、操作を完了するためにリソースを割り当てる必要があるオブジェクトに対して、操作一時停止アラートが発行されます。
- 再開可能領域割当ての例: AFTER SUSPENDトリガーの登録
  この例では、システム全体で有効なAFTER SUSPENDトリガーを作成し、ユーザーSYSとしてデータベース・レベルで登録する方法を示します。

親トピック: スキーマ・オブジェクトの領域の管理

19.2.1 再開可能領域割当ての概要

Oracle Databaseでは、領域割当てが失敗した場合に大規模なデータベース処理を一時停止して、後で再開するための方法が提供されています。そのため、Oracle Databaseサーバーがユーザーにエラーを返すかわりに対処措置を講じることができます。エラー条件が訂正されると、一時停止していた処理が自動的に再開します。この機能のことを、再開可能領域割当てと呼びます。また、影響を受ける文のことを、再開可能文と呼びます。

- 再開可能領域割当ての動作
  再開可能領域割当ての動作の概要を示します。
- 再開可能な操作
  一部の操作は再開可能です。

- 訂正可能なエラー
  一部のエラーは訂正可能です。
- 再開可能領域割当てと分散処理
  分散環境で、ユーザーが再開可能領域割当てを有効または無効にした場合、またはDBAがRESUMABLE_TIMEOUT初期化パラメータを変更した場合、その影響を受けるのはローカルのインスタンスです。RESUMABLEはリモートで有効にできません。
- パラレル実行と再開可能領域割当て
  パラレル実行では、パラレル実行サーバー・プロセスの1つで訂正可能なエラーが発生した場合、そのサーバー・プロセスの実行が一時停止します。

親トピック: 再開可能領域割当ての管理

19.2.1.1 再開可能領域割当ての動作

次に、再開可能領域割当ての動作の概要を示します。

1. 文が再開可能モードで実行されるのは、その文のセッションが、次のいずれかの処理によって再開可能領域割当てに対応している場合のみです。
   - RESUMABLE_TIMEOUT初期化パラメータが0 (ゼロ)以外の値に設定されている場合に、ALTER SESSION ENABLE RESUMABLE文が実行前にセッション内で発行された。
   - ALTER SESSION ENABLE RESUMABLE TIMEOUT timeout_value文が実行前にセッション内で発行され、timeout_valueが0 (ゼロ)以外の値である。
2. 次のいずれかの条件が成立すると、再開可能文が一時停止します(非再開可能文では、これらの条件に対応するエラーが通知されます)。
   - 領域不足条件
   - 最大エクステント数到達条件
   - スペース割当制限超過条件。
3. 再開可能文の実行が一時停止すると、ユーザー指定の操作の実行、エラーの記録および文の実行状態の問合せを行うメカニズムがただちに動作します。再開可能文が一時停止すると、次の処理が実行されます。
   - エラーがアラート・ログに記録されます。
   - 一時停止された再開可能セッションのアラートが発行されます。
   - ユーザーがAFTER SUSPENDシステム・イベントに対してトリガーを登録していた場合は、そのユーザー・トリガーが実行されます。ユーザー指定のPL/SQLプロシージャは、DBMS_RESUMABLEパッケージとDBA_RESUMABLEまたはUSER_RESUMABLEビューを使用して、エラー・メッセージ・データにアクセスできます。
4. 文が中断されると、自動的にトランザクションも中断されます。その結果、すべてのトランザクション・リソースが文の一時停止から再開までの間保持されます。
5. ユーザーの介入や他の問合せによってソート領域が解放されるなどの結果としてエラー条件が解決されると、一時停止していた文が自動的に実行を再開し、一時停止された再開可能セッションのアラートがクリアされます。
6. 一時停止した文は、DBMS_RESUMABLE.ABORT( )プロシージャを使用して、強制的に例外を発生できます。このプロシージャは、DBAまたは文を発行したユーザーがコールできます。

7. RESUMABLE_TIMEOUT初期化パラメータまたはALTER SESSION ENABLE RESUMABLE TIMEOUT文で指定された一時停止のタイムアウト間隔は、再開可能文に対応付けられています。タイムアウト間隔内にエラー条件が解決されない場合は、タイムアウト間隔の間一時停止していた再開可能文がリストアされ、ユーザーに例外が返されます。
8. 再開可能文は、実行中に一時停止と再開を複数回繰り返すことができます。

親トピック: 再開可能領域割当ての概要

19.2.1.2 再開可能な操作

一部の操作は再開可能です。

再開可能な操作は、次のとおりです。

- 問合せ

  一時領域(ソート領域)を使い果たしたSELECT文は、再開可能な実行の候補になります。OCIの使用時は、OCIStmtExecute()およびOCIStmtFetch()の各コールが候補になります。

- DML

  INSERT文、UPDATE文およびDELETE文が候補になります。それらを実行するために使用するインタフェースは何でもかまいません。OCI、PL/SQLまたは別のインタフェースを使用できます。また、外部表からのINSERT INTO...SELECTも再開可能にできます。

- インポート/エクスポート

  SQL*Loaderについては、リカバリ可能なエラーの後に文が再開可能かどうかをコマンドライン・パラメータで制御します。

- DDL

  次の文が、再開可能な実行の候補になります。

  - CREATE TABLE ... AS SELECT
  - CREATE INDEX
  - ALTER INDEX ... REBUILD
  - ALTER TABLE ... MOVE PARTITION
  - ALTER TABLE ... SPLIT PARTITION
  - ALTER INDEX ... REBUILD PARTITION
  - ALTER INDEX ... SPLIT PARTITION
  - CREATE MATERIALIZED VIEW
  - CREATE MATERIALIZED VIEW LOG

親トピック: 再開可能領域割当ての概要

19.2.1.3 訂正可能なエラー

一部のエラーは訂正可能です。

訂正可能なエラーには、次の3つのクラスがあります。

- 領域不足条件

  データベースの操作によって、表領域内の表、索引、一時セグメント、UNDOセグメント、クラスタ、LOB、表パーティ

ションまたは索引パーティション用のエクステントをこれ以上取得できません。たとえば、次のエラーはこのカテゴリに属します。

```
ORA-01653 unable to extend table ... in tablespace ...
ORA-01654 unable to extend index ... in tablespace ...
```

- 最大エクステント数到達条件

  表、索引、一時セグメント、UNDOセグメント、クラスタ、LOB、表パーティションまたは索引パーティション内のエクステント数が、オブジェクトで定義されている最大エクステント数に等しくなりました。たとえば、次のエラーはこのカテゴリに属します。

```
ORA-01631 max # extents ... reached in table ...
ORA-01632 max # extents ... reached in index ...
```

- スペース割当制限超過条件

  ユーザーが自分に割り当てられている表領域内のスペース割当制限を超過しました。具体的には、次のエラーによって示されます。

```
ORA-01536 space quote exceeded for tablespace string
```

親トピック: 再開可能領域割当ての概要

19.2.1.4 再開可能領域割当てと分散処理

分散環境で、ユーザーが再開可能領域割当てを有効または無効にした場合、またはDBAがRESUMABLE_TIMEOUT初期化パラメータを変更した場合、その影響を受けるのはローカルのインスタンスです。RESUMABLEはリモートで有効にできません。

分散トランザクションで、リモート・インスタンスのセッションが一時停止されるのは、リモート・インスタンスがそのサイトでインスタンスまたはセッションに対してRESUMABLEをすでに有効にしている場合のみです。

親トピック: 再開可能領域割当ての概要

19.2.1.5 パラレル実行と再開可能領域割当て

パラレル実行では、パラレル実行サーバー・プロセスの1つで訂正可能なエラーが発生した場合、そのサーバー・プロセスの実行が一時停止します。

他のパラレル実行サーバー・プロセスでは、エラーが発生するまで、または一時停止したサーバー・プロセスに(直接または間接に)ブロックされるまで、個々のタスクの実行が継続されます。訂正可能なエラーが解決すると、一時停止したプロセスが実行を再開し、パラレル操作の実行は継続されます。一時停止したプロセスが終了した場合、パラレル操作は終了し、ユーザーに対してエラーが返されます。

異なるパラレル実行プロセスで、1つ以上の訂正可能なエラーが発生することがあります。この結果、AFTER SUSPENDトリガーが複数回、パラレルに発行される場合があります。また、あるパラレル実行サーバー・プロセスが一時停止中に他のパラレル実行サーバー・プロセスで訂正不可能なエラーが発生すると、一時停止していた文はただちに終了します。

パラレル実行については、すべてのパラレル実行コーディネータとサーバー・プロセスがDBA_RESUMABLEまたはUSER_RESUMABLEビューに独自のエントリを持っています。

親トピック: 再開可能領域割当ての概要

19.2.2 再開可能領域割当ての有効化および無効化

SQL文を実行して特定の初期化パラメータを設定し、再開可能領域割当てを有効または無効にできます。

- 再開可能領域割当ての有効化および無効化について
  再開可能領域割当ては、再開可能モードを有効にしたセッションの中で文を実行するときのみ可能です。
- RESUMABLE_TIMEOUT初期化パラメータの設定
  RESUMABLE_TIMEOUT初期化パラメータを設定して、デフォルトのシステム全体のタイムアウト間隔を指定できます。
- ALTER SESSIONを使用した再開可能領域割当ての有効化と無効化
  セッション内で、ユーザーはALTER SESSION SET文を発行してRESUMABLE_TIMEOUT初期化パラメータを設定し、再開可能領域割当てを有効にしてタイムアウト値を変更するか、または再開可能モードを無効にできます。

親トピック: 再開可能領域割当ての管理

19.2.2.1 再開可能領域割当ての有効化および無効化について

再開可能領域割当ては、再開可能モードを有効にしたセッションの中で文を実行するときのみ可能です。

再開可能領域割当てがセッションで有効になるのは、ALTER SESSION ENABLE RESUMABLE文が実行され、そのセッションのRESUMABLE_TIMEOUT初期化パラメータが0 (ゼロ)以外の値に設定されている場合です。RESUMABLE_TIMEOUT初期化パラメータをシステム・レベルで設定すると、その設定は、タイムアウト値が指定されていないALTER SESSION ENABLE RESUMABLE文のデフォルトになります。ALTER SESSION ENABLE RESUMABLE文でタイムアウト値を指定すると、システム・デフォルトがオーバーライドされます。

ALTER SESSION ENABLE RESUMABLE文を実行しても、次の場合はセッションで再開可能領域割当てが無効になります。:

- セッションでALTER SESSION ENABLE RESUMABLE文が実行されない場合。
- セッションでALTER SESSION DISABLE RESUMABLE文が実行された場合。
- セッションでALTER SESSION ENABLE RESUMABLE文が実行され、タイムアウト値が0 (ゼロ)の場合。

ノート:

一時停止した文はなんらかのシステム・リソースを保持している可能性があるため、ユーザーが再開可能領域割当てを有効にして、再開可能文を実行するには、RESUMABLE システム権限が付与されている必要があります。

親トピック: 再開可能領域割当ての有効化および無効化

19.2.2.2 RESUMABLE_TIMEOUT初期化パラメータの設定

RESUMABLE_TIMEOUT初期化パラメータを設定して、デフォルトのシステム全体のタイムアウト間隔を指定できます。

たとえば、初期化パラメータ・ファイルでRESUMABLE_TIMEOUTパラメータを次のように設定すると、タイムアウト間隔は1時間に設定されます。

```
RESUMABLE_TIMEOUT   = 3600
```

このパラメータが0 (ゼロ)に設定された場合、タイムアウト値を指定しないでALTER SESSION ENABLE RESUMABLE文を実行するセッションでも、再開可能領域割当ては無効になります。

ALTER SYSTEM SET文を使用して、このパラメータの値をシステム・レベルで変更することもできます。たとえば、次の文では、

タイムアウト値を指定しないでALTER SESSION ENABLE RESUMABLE文を実行するすべてのセッションに対して再開可能領域割当てを無効にします。

```
ALTER SYSTEM SET RESUMABLE_TIMEOUT=0;
```

親トピック: 再開可能領域割当ての有効化および無効化

19.2.2.3 ALTER SESSIONを使用した再開可能領域割当ての有効化と無効化

セッション内で、ユーザーはALTER SESSION SET文を発行してRESUMABLE_TIMEOUT初期化パラメータを設定し、再開可能領域割当てを有効にしてタイムアウト値を変更するか、または再開可能モードを無効にできます。

次のSQL文を使用して、デフォルトのシステムRESUMABLE_TIMEOUT値でセッションの再開可能モードを有効化できます。

```
ALTER SESSION ENABLE RESUMABLE;
```

再開可能モードを無効化するには、次の文を使用します。

```
ALTER SESSION DISABLE RESUMABLE;
```

新しいセッションのデフォルトの再開可能モードは無効です。

タイムアウト間隔や、再開可能文の識別に使用する名前も指定できます。次の項では、各操作について説明します。

- タイムアウト間隔の指定
  介入操作が発生しなかった場合に一時停止した文がエラーとなるタイムアウト間隔は、再開可能モードを有効化するときに指定できます。
- 再開可能文の命名
  再開可能文は、名前で識別するように設定できます。

関連項目:

「LOGONトリガーを使用したデフォルト再開可能モードの設定」

親トピック: 再開可能領域割当ての有効化および無効化

19.2.2.3.1 タイムアウト間隔の指定

介入操作が発生しなかった場合に一時停止した文がエラーになるタイムアウト間隔は、再開可能モードを有効化するときに指定できます。

次の文は、再開可能トランザクションが3600秒後にタイムアウトし、エラーになることを指定します。

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600;
```

TIMEOUTの値は、別のALTER SESSION ENABLE RESUMABLE文や他の手段によって変更されるまで、またはセッションが終了するまで有効です。RESUMABLE_TIMEOUT初期化パラメータが設定されていない場合、ENABLE RESUMABLE TIMEOUT句を使用して再開可能モードを有効にするときのデフォルトのタイムアウト間隔は7200秒になります。

関連項目:

再開可能領域割当てのタイムアウト間隔を変更するその他の方法の詳細は、「RESUMABLE_TIMEOUT初期化パラメータの設定」を参照してください

親トピック: ALTER SESSIONを使用した再開可能領域割当ての有効化と無効化

19.2.2.3.2 再開可能文の命名

再開可能文は、名前で識別するように設定できます。

次の文は、再開可能文に名前を割り当てます。

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600 NAME 'insert into table';
```

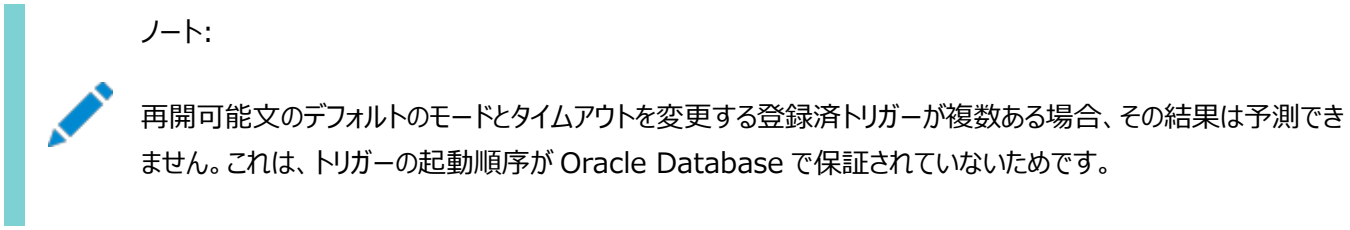
NAMEの値は、別のALTER SESSION ENABLE RESUMABLE文によって変更されるまで、またはセッションが終了するまで有効です。NAMEのデフォルト値は、'User username(userid), Session sessionid, Instance instanceid'です。

文の名前は、DBA_RESUMABLEビューおよびUSER_RESUMABLEビューで再開可能文を識別する際に使用します。

親トピック: ALTER SESSIONを使用した再開可能領域割当ての有効化と無効化

19.2.3 LOGONトリガーを使用したデフォルト再開可能モードの設定

RESUMABLE_TIMEOUT初期化パラメータを設定する以外に、デフォルトの再開可能モードを設定するもう1つの方法は、データベース・レベルのLOGONトリガーを登録して、再開可能を有効化してタイムアウト間隔を設定するように、ユーザーのセッションを変更する方法です。



ノート:

再開可能文のデフォルトのモードとタイムアウトを変更する登録済トリガーが複数ある場合、その結果は予測できません。これは、トリガーの起動順序が Oracle Database で保証されていないためです。

親トピック: 再開可能領域割当ての管理

19.2.4 一時停止文の検出

再開可能文が一時停止するとき、クライアントにはエラーは通知されません。訂正処理を実行するため、Oracle Databaseではユーザーにエラーを通知して状況に関する情報を提供するかわりの手段が提供されています。

- ユーザーへの通知: AFTER SUSPENDシステム・イベントおよびトリガー
  再開可能文で訂正可能なエラーが発生すると、システムは内部的にAFTER SUSPENDシステム・イベントを生成します。ユーザーは、このイベントに対するトリガーをデータベース・レベルとスキーマ・レベルの両方で登録できます。ユーザーがこのシステム・イベントを処理するトリガーを登録した場合、トリガーはSQL文が一時停止した後に実行されます。
- ビューを使用した一時停止文情報の取得
  ビューのセットに対する問合せにより、再開可能文の状態に関する情報を取得できます。
- DBMS_RESUMABLEパッケージの使用方法
  DBMS_RESUMABLEパッケージは、再開可能領域割当ての管理に役立ちます。

親トピック: 再開可能領域割当ての管理

19.2.4.1 ユーザーへの通知: AFTER SUSPENDシステム・イベントおよびトリガー

再開可能文で訂正可能なエラーが発生すると、システムは内部的にAFTER SUSPENDシステム・イベントを生成します。ユーザーは、このイベントに対するトリガーをデータベース・レベルとスキーマ・レベルの両方で登録できます。ユーザーがこのシステム・イ

ベントを処理するトリガーを登録した場合、トリガーはSQL文が一時停止した後に実行されます。

AFTER SUSPENDトリガー内部で実行されたSQL文は、再開不可能であり、かつ自律型です。トリガー内部で開始されたトランザクションは、SYSTEMロールバック・セグメントを使用します。これらの条件が課されるのは、デッドロックを回避し、文と同じエラー条件に直面する可能性を少なくするためです。

ユーザーはUSER_RESUMABLEビュー、DBA_RESUMABLEビューまたはDBMS_RESUMABLE.SPACE_ERROR_INFOファンクションをトリガー内部で使用して、再開可能文に関する情報を取得できます。

また、トリガーではDBMS_RESUMABLEパッケージをコールして、一時停止した文の終了や再開可能タイムアウト値の変更を実行できます。次の例では、DBMS_RESUMABLEをコールしてタイムアウトを3時間に設定する、システム全体で有効なAFTER SUSPENDトリガーを作成することによって、デフォルトのシステム・タイムアウトを変更します。

```
CREATE OR REPLACE TRIGGER resumable_default_timeout
AFTER SUSPEND
ON DATABASE
BEGIN
   DBMS_RESUMABLE.SET_TIMEOUT(10800);
END;
/
```

関連項目:

トリガーとシステム・イベントの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

親トピック: 一時停止文の検出

19.2.4.2 ビューを使用した一時停止文情報の取得

ビューのセットに対する問合せにより、再開可能文の状態に関する情報を取得できます。

ビュー	説明
DBA_RESUMABLE USER_RESUMABLE	これらのビューには、現在実行中または一時停止しているすべての再開可能文に関する行が含まれます。これらは、再開可能文の実行状態を監視したり、再開可能文に関する特定の情報を取得するために、DBA、AFTER SUSPEND トリガーまたは別のセッションが使用できます。
V$SESSION_WAIT	ある文が一時停止すると、その文を起動したセッションは待機状態になります。このビューには、「文が一時停止され、エラーのクリアを待機しています」という内容の EVENT 列を持つセッションに対して 1 行が挿入されます。

親トピック: 一時停止文の検出

19.2.4.3 DBMS_RESUMABLEパッケージの使用方法

DBMS_RESUMABLEパッケージは、再開可能領域割当ての管理に役立ちます。

次のプロシージャを起動できます。

プロシージャ	説明

プロシージャ	説明
ABORT(sessionID)	このプロシージャは、一時停止された再開可能文を終了します。パラメータ sessionID は、文が実行されているセッション ID です。パラレル DML/DDL の場合、sessionID はパラレル DML/DDL に参加している任意のセッション ID です。 Oracle Database では、ABORT 操作が常に正常終了することが保証されています。これは、AFTER SUSPEND トリガーの内部からでも外部からでもコールできます。 ABORT のコール元は、sessionID を持つセッションの所有者、ALTER SYSTEM 権限を持っているユーザー、DBA 権限を持っているユーザーのいずれかである必要があります。
GET_SESSION_TIMEOUT(sessionID)	このファンクションは、sessionID を持つセッションで設定されている再開可能領域割当ての現行のタイムアウト値を返します。タイムアウトは秒数で返されます。セッションが存在しない場合、このファンクションは-1 を返します。
SET_SESSION_TIMEOUT(sessionID, timeout)	このプロシージャは、sessionID を持つセッションに対して再開可能領域割当てのタイムアウト間隔を設定します。パラメータ timeout の単位は秒数です。新しい timeout 設定は、即時にセッションに適用されます。セッションが存在しない場合は何も起こりません。
GET_TIMEOUT( )	このファンクションは、現行セッションで設定されている再開可能領域割当ての現行の timeout 値を返します。戻される値は秒で示されます。
SET_TIMEOUT(timeout)	このプロシージャは、現行セッションに対して再開可能領域割当ての timeout 値を設定します。パラメータ timeout の単位は秒数です。新しいタイムアウト設定はセッションに即時適用されます。

関連項目:

DBMS_RESUMABLEパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 一時停止文の検出

19.2.5 操作一時停止アラート

再開可能セッションが一時停止されると、操作を完了するためにリソースを割り当てる必要があるオブジェクトに対して、操作一時停止アラートが発行されます。

リソースが割り当てられて操作が完了すると、操作一時停止アラートはクリアされます。システム生成アラートの詳細は、「表領域のアラートの管理」を参照してください。

親トピック: 再開可能領域割当ての管理

19.2.6 再開可能領域割当ての例: AFTER SUSPENDトリガーの登録

この例では、システム全体で有効なAFTER SUSPENDトリガーを作成し、ユーザーSYSとしてデータベース・レベルで登録する方法を示します。

任意のセッションで再開可能文が一時停止すると、このトリガーは次の2つのうちどちらかの処理を実行します。

- UNDOセグメントが領域上限に達した場合は、メッセージがDBAに送られ、文が終了します。
- 他のリカバリ可能なエラーが発生した場合には、タイムアウト間隔が8時間にリセットされます。

この例で使用する文を次に示します。

```
CREATE OR REPLACE TRIGGER resumable_default
AFTER SUSPEND
ON DATABASE
DECLARE
   /* declare transaction in this trigger is autonomous */
   /* this is not required because transactions within a trigger
      are always autonomous */
   PRAGMA AUTONOMOUS_TRANSACTION;
   cur_sid           NUMBER;
   cur_inst          NUMBER;
   errno             NUMBER;
   err_type          VARCHAR2;
   object_owner      VARCHAR2;
   object_type       VARCHAR2;
   table_space_name  VARCHAR2;
   object_name       VARCHAR2;
   sub_object_name   VARCHAR2;
   error_txt         VARCHAR2;
   msg_body          VARCHAR2;
   ret_value         BOOLEAN;
   mail_conn         UTL_SMTP.CONNECTION;
BEGIN
   -- Get session ID
   SELECT DISTINCT(SID) INTO cur_SID FROM V$MYSTAT;
   -- Get instance number
   cur_inst := userenv('instance');
   -- Get space error information
   ret_value :=
   DBMS_RESUMABLE.SPACE_ERROR_INFO(err_type,object_type,object_owner,
        table_space_name,object_name, sub_object_name);
   /*
   -- If the error is related to undo segments, log error, send email
   -- to DBA, and terminate the statement. Otherwise, set timeout to 8 hours.
   --
   -- sys.rbs_error is a table which is to be
   -- created by a DBA manually and defined as
   -- (sql_text VARCHAR2(1000), error_msg VARCHAR2(4000),
   -- suspend_time DATE)
   */
   IF OBJECT_TYPE = 'UNDO SEGMENT' THEN
       /* LOG ERROR */
       INSERT INTO sys.rbs_error (
           SELECT SQL_TEXT, ERROR_MSG, SUSPEND_TIME
           FROM DBMS_RESUMABLE
           WHERE SESSION_ID = cur_sid AND INSTANCE_ID = cur_inst
        );
```

```
       SELECT ERROR_MSG INTO error_txt FROM DBMS_RESUMABLE
           WHERE SESSION_ID = cur_sid and INSTANCE_ID = cur_inst;
        -- Send email to receipient through UTL_SMTP package
        msg_body:='Subject: Space Error Occurred
                   Space limit reached for undo segment ' || object_name ||
                   on ' || TO_CHAR(SYSDATE, 'Month dd, YYYY, HH:MIam') ||
                   '. Error message was ' || error_txt;
        mail_conn := UTL_SMTP.OPEN_CONNECTION('localhost', 25);
        UTL_SMTP.HELO(mail_conn, 'localhost');
        UTL_SMTP.MAIL(mail_conn, 'sender@localhost');
        UTL_SMTP.RCPT(mail_conn, 'recipient@localhost');
        UTL_SMTP.DATA(mail_conn, msg_body);
        UTL_SMTP.QUIT(mail_conn);
        -- Terminate the statement
        DBMS_RESUMABLE.ABORT(cur_sid);
    ELSE
        -- Set timeout to 8 hours
        DBMS_RESUMABLE.SET_TIMEOUT(28800);
    END IF;
    /* commit autonomous transaction */
    COMMIT;
END;
/
```

親トピック: 再開可能領域割当ての管理

19.3 未使用領域の再利用

未使用領域を再利用できます。セグメント・アドバイザはOracle Databaseコンポーネントで、再利用可能な領域があるセグメントを識別します。

- 未使用領域の再利用について
  時間の経過とともに、表領域内のオブジェクトを更新および削除すると、新しいデータに対して個別に再利用するには不十分な小さい空き領域が作成されます。このような空き領域は、断片化された空き領域と呼ばれます。
- セグメント・アドバイザ
  セグメント・アドバイザは、再生に使用できる領域があるセグメントを特定します。
- オンラインによるデータベース・セグメントの縮小
  Oracle Databaseセグメントの最高水位標の下の断片化された空き領域を再生するには、オンラインによるセグメントの縮小を使用します。
- 未使用領域の割当て解除
  未使用領域の割当てを解除する場合は、未使用の(最高水位標)データベース・セグメントの最後で未使用領域を解放して、表領域の他のセグメントで領域を使用できるようにします。

親トピック: スキーマ・オブジェクトの領域の管理

19.3.1 未使用領域の再利用について

時間の経過とともに、表領域内のオブジェクトを更新および削除すると、新しいデータに対して個別に再利用するには不十分な小さい空き領域が作成されます。このような空き領域は、断片化された空き領域と呼ばれます。

オブジェクトに断片化された空き領域があると、結果的に多くの無駄な領域が生じ、データベースのパフォーマンスに影響を与える場合があります。断片化を解消して領域を再生するには、オンラインによるセグメントの縮小を実行することをお薦めします。このプロセスは、最高水位標の下の断片化された空き領域を統合し、セグメントを圧縮します。圧縮すると最高水位標が移動し、その結果、最高水位標の上に新しい空き領域ができます。その後、この最高水位標よりも上の領域は、割当て解除されます。セグメントは、この操作の開始から終了までの大半で、問合せおよびDMLを使用でき、特別なディスク領域の割当ては不要で

す。

オンラインによるセグメントの縮小で利点を得られるセグメントを特定するには、セグメント・アドバイザを使用します。セグメント・アドバイザは、自動セグメント領域管理(ASSM)を備えたローカル管理表領域のセグメントに対してのみ使用できます。調査可能なセグメントの種類については、他にも制限があります。詳細は、「オンラインによるデータベース・セグメントの縮小」を参照してください。

再生可能な領域を含む表がオンラインによるセグメントの縮小に適していない場合、または領域の再生中に表の論理属性または物理属性を変更する場合は、セグメントの縮小にかわる手段として、表のオンライン再定義を使用できます。オンライン再定義は再編成とも呼ばれます。オンライン再定義は、オンラインによるセグメントの縮小とは異なり、別のディスク領域を割り当てる必要があります。詳細は、「表のオンライン再定義」を参照してください。

親トピック: 未使用領域の再利用

19.3.2 セグメント・アドバイザ

セグメント・アドバイザは、再生に使用できる領域があるセグメントを特定します。

- セグメント・アドバイザについて
  セグメント・アドバイザは、自動ワークロード・リポジトリ(AWR)の使用状況と増加に関する統計情報を調べ、セグメントのデータをサンプリングして、分析を実行します。
- セグメント・アドバイザの使用
  セグメント・アドバイザを使用するには、自動セグメント・アドバイザの結果を確認し、オプションで、手動でセグメント・アドバイザを実行します。
- 自動セグメント・アドバイザ
  自動セグメント・アドバイザは、すべてのメンテナンス・ウィンドウ内で実行するように構成されている自動化メンテナンス・タスクです。
- 手動によるセグメント・アドバイザの実行
  セグメント・アドバイザは、Cloud ControlまたはPL/SQLパッケージのプロシージャ・コールを使用して、いつでも手動で実行できます。
- セグメント・アドバイザの結果の表示
  セグメント・アドバイザでは、様々な種類の結果(推奨事項、結果、処置、オブジェクト)が作成されます。
- 自動セグメント・アドバイザの構成
  自動セグメント・アドバイザは自動化メンテナンス・タスクです。そのため、このタスクの実行時に変更を加える場合は、Cloud ControlまたはPL/SQLパッケージ・プロシージャ・コールを使用できます。適切なリソース・プランを変更することで、タスクに割り当てられているリソースを制御することもできます。
- 自動セグメント・アドバイザ情報の表示
  ビューを問い合せて、自動セグメント・アドバイザに固有の情報を表示できます。

親トピック: 未使用領域の再利用

19.3.2.1 セグメント・アドバイザについて

セグメント・アドバイザは、自動ワークロード・リポジトリ(AWR)の使用状況と増加に関する統計情報を調べ、セグメントのデータをサンプリングして、分析を実行します。

メンテナンス・ウィンドウの間に自動化メンテナンス・タスクとして実行されるように構成されていますが、オンデマンド(手動)で実行することもできます。セグメント・アドバイザの自動化メンテナンス・タスクのことを、自動セグメント・アドバイザと呼びます。この情報は、容量を計画したり、縮小するセグメントを決定するときに使用できます。

セグメント・アドバイザは、次の種類のアドバイスを生成します。

- セグメント・アドバイザにより、オブジェクトにかなりの量の空き領域があることが判断されると、オンラインによるセグメントの縮小が提案されます。自動セグメント領域管理のない表領域内の表の場合など、オブジェクトがセグメントの縮小に適さない表である場合は、オンラインによる表の再定義が提案されます。
- 高度な行圧縮方法を使用して表を圧縮することで効果が期待できるとセグメント・アドバイザが判断した場合、そのことを推奨事項として表示します。(自動セグメント・アドバイザの場合のみ。「自動セグメント・アドバイザ」を参照。)
- セグメント・アドバイザにより、特定のしきい値を超える行連鎖のある表が検出されると、表に過剰な連鎖行があることが記録されます。

ノート:

セグメント・アドバイザによるフラグ設定の対象は、行を長くする更新によって生じた行連鎖のタイプのみです。

領域管理のアラートを受け取った場合、または領域の再生を決定した場合は、セグメント・アドバイザを開始してください。

親トピック: セグメント・アドバイザ

19.3.2.2 セグメント・アドバイザの使用

セグメント・アドバイザを使用するには、自動セグメント・アドバイザの結果を確認し、オプションで、手動でセグメント・アドバイザを実行します。

セグメント・アドバイザを使用するには:

1. 自動セグメント・アドバイザの結果を確認します。

   自動セグメント・アドバイザを理解するには、この後の「自動セグメント・アドバイザ」を参照してください。結果の表示方法の詳細は、「セグメント・アドバイザの結果の表示」を参照してください。

2. (オプション)セグメント・アドバイザを手動で再実行し、個々のセグメントの更新結果を取得します。

   この後の「手動によるセグメント・アドバイザの実行」を参照してください。

親トピック: セグメント・アドバイザ

19.3.2.3 自動セグメント・アドバイザ

自動セグメント・アドバイザは、すべてのメンテナンス・ウィンドウ内で実行するように構成されている自動化メンテナンス・タスクです。

自動セグメント・アドバイザは、すべてのデータベース・オブジェクトを分析するわけではありません。かわりに、データベース統計を調査し、セグメント・データをサンプリングした後、次のような分析対象オブジェクトを選択します。

- 領域のクリティカルまたは警告のしきい値を超えた表領域
- アクティビティが最も多いセグメント
- 増加率が最も高いセグメント

また、自動セグメント・アドバイザでは、高度な行圧縮方法で圧縮した場合に節約できる領域の大きさを判断するために、10MB以上で、索引が3つ以上存在する表が評価されます。

分析対象オブジェクトが選択されても、セグメント・アドバイザがオブジェクトを処理する前にメンテナンス・ウィンドウが終了する場合、そのオブジェクトは、自動セグメント・アドバイザの次回の実行に組み込まれます。

自動セグメント・アドバイザによって分析対象として選択された表領域とセグメントのセットは変更できません。ただし、自動セグメント・アドバイザのタスクの有効化または無効化、自動セグメント・アドバイザの実行予定回数、または自動化メンテナンス・タスクのシステム・リソース使用率は変更できます。詳細は、「自動セグメント・アドバイザの構成」を参照してください。

関連項目:

- 「セグメント・アドバイザの結果の表示」
- 自動データベース・メンテナンス・タスクの管理
- 高度な行圧縮の詳細は、「表圧縮の使用」を参照してください

親トピック: セグメント・アドバイザ

19.3.2.4 手動によるセグメント・アドバイザの実行

セグメント・アドバイザは、Cloud ControlまたはPL/SQLパッケージのプロシージャ・コールを使用して、いつでも手動で実行できます。

セグメント・アドバイザを手動で実行する理由には、次のものがあります。

- 自動セグメント・アドバイザが選択しなかった表領域またはセグメントを分析するため。
- 個々の表領域またはセグメントを再度分析して、最新の推奨事項を入手するため。

セグメント・アドバイザには、3種類のレベルでアドバイスを要求できます。

- セグメント・レベル—非パーティション表、パーティション表のパーティションまたはサブパーティション、索引、またはLOB列など、単一のセグメントに対してアドバイスが生成されます。
- オブジェクト・レベル—表や索引など、オブジェクト全体についてアドバイスが生成されます。オブジェクトがパーティション化されている場合は、オブジェクトのすべてのパーティションに対してアドバイスが生成されます。また、Cloud Controlからセグメント・アドバイザを手動で実行する場合は、表の索引やLOBセグメントなど、オブジェクトの依存オブジェクトについてアドバイスを要求できます。
- 表領域レベル—表領域のすべてのセグメントに対してアドバイスが生成されます。

表19-2にあるOBJECT_TYPE列は、アドバイスを要求できるオブジェクトのタイプを示しています。

- Cloud Controlを使用したセグメント・アドバイザの手動実行
  セグメント・アドバイザは、Cloud Controlを使用して手動で実行できます
- PL/SQLを使用したセグメント・アドバイザの手動実行
  セグメント・アドバイザは、DBMS_ADVISORパッケージを使用して実行できます。

親トピック: セグメント・アドバイザ

19.3.2.4.1 Cloud Controlを使用したセグメント・アドバイザの手動実行

Cloud Controlを使用して、セグメント・アドバイザを手動で実行できます

Cloud Controlを使用してセグメント・アドバイザを手動で実行するには、OEM_ADVISORロールが付与されている必要があります。セグメント・アドバイザを実行するには、次の2つの方法があります。

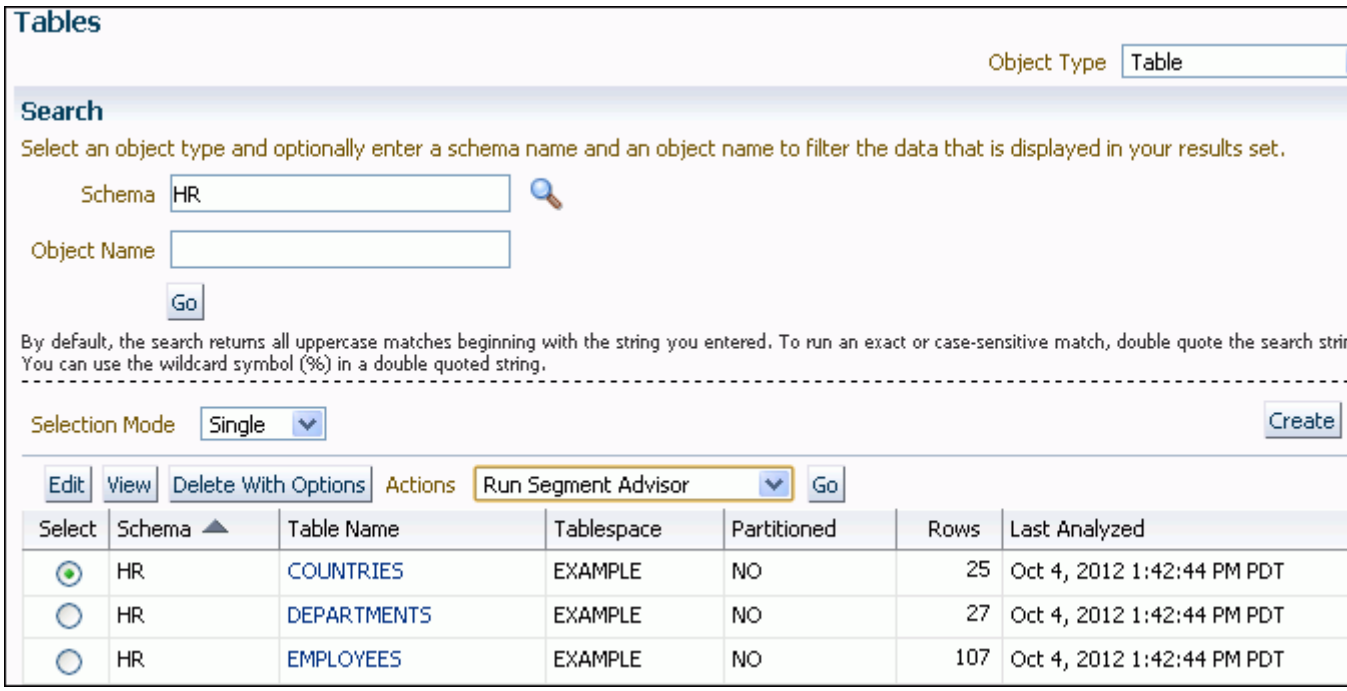
- セグメント・アドバイザ・ウィザードの使用

  この方法を使用すると、表領域レベルまたはオブジェクト・レベルでアドバイスを要求できます。オブジェクト・レベルでは、表、索引、表パーティションおよび索引パーティションについてアドバイスを要求できます。

- スキーマ・オブジェクトを表示するページでの「セグメント・アドバイザの実行」コマンドの使用。

  たとえば、(「スキーマ」メニューからアクセス可能な)「表」ページに表を表示する場合は、表を選択して「アクション」メニューから「セグメント・アドバイザの実行」を選択します。

  図19-1 「表」ページ



  この方法では、スキーマ・オブジェクトの依存オブジェクトをセグメント・アドバイザの実行対象に含めることができます。たとえば、表を選択して「セグメント・アドバイザの実行」を選択すると、パーティション、索引セグメント、LOBセグメントなど、表の依存オブジェクトが表示されます。依存オブジェクトを選択することで、そのオブジェクトを実行対象に指定できます。

両方の場合に、Cloud Controlは、Oracle Database Schedulerジョブとしてセグメント・アドバイザ・タスクを作成します。即座に実行するようにジョブをスケジュールすることも、スケジューラが提供する高度なスケジューリング機能を利用することもできます。

セグメント・アドバイザ・ウィザードを使用してセグメント・アドバイザを手動で実行するには:

1. データベース・ホームページにアクセスします。
2. 「パフォーマンス」メニューから、「アドバイザ・ホーム」を選択します。

   「アドバイザ・セントラル」ページが表示されます。(図19-2を参照してください。)

3. 「アドバイザ」の下の「セグメント・アドバイザ」をクリックします。

   セグメント・アドバイザ・ウィザードの最初のページが表示されます。

4. セグメント・アドバイザのジョブをスケジュールするウィザードの各ステップに従い、ウィザードの最終ページで「発行」をクリックします。

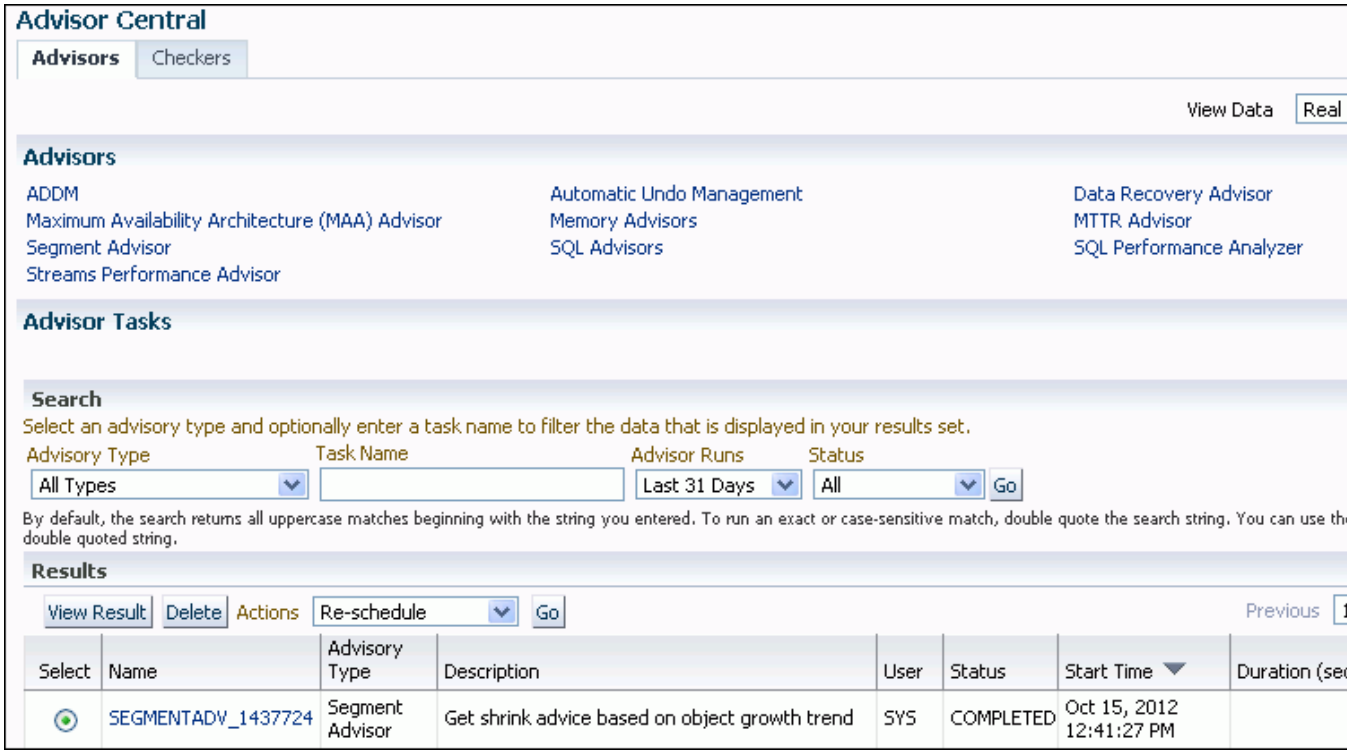
   「アドバイザ・セントラル」ページが表示され、「結果」ヘッダーの下のリストの最初に新しいセグメント・アドバイザ・ジョブが表示されます。ジョブ・ステータスは、SCHEDULEDまたはRUNNINGになります。(自分のジョブが表示されない場合は、

リストの上の検索フィールドを使用して表示してください。)

5. ジョブのステータスをチェックします。COMPLETEDでない場合は、ページの上部にある「リフレッシュ」コントロールを使用してページをリフレッシュします。(ブラウザのリフレッシュ・アイコンは使用しないでください。)

   ジョブ・ステータスがCOMPLETEDに変わった場合は、「選択」列をクリックしてジョブを選択し、「結果の表示」をクリックします。

   図19-2 「アドバイザ・セントラル」ページ



関連項目:

スケジューラの高度なスケジューリング機能の詳細は、「Oracle Schedulerを使用したジョブのスケジューリング」を参照してください。

親トピック: 手動によるセグメント・アドバイザの実行

19.3.2.4.2 PL/SQLを使用したセグメント・アドバイザの手動実行

セグメント・アドバイザは、DBMS_ADVISORパッケージを使用して実行できます。

パッケージ・プロシージャを使用してセグメント・アドバイザのタスクを作成し、タスクの引数を設定してからタスクを実行します。そのためには、ADVISOR権限が必要です。表19-1に、セグメント・アドバイザに関連するプロシージャを示します。これらのプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

表19-1 セグメント・アドバイザに関連するDBMS_ADVISORパッケージ・プロシージャ

パッケージ・プロシージャ名	説明
CREATE_TASK	このプロシージャを使用して、セグメント・アドバイザのタスクを作成します。ADVISOR_NAME パラメータの値に、'Segment Advisor'を指定します。

パッケージ・プロシージャ名	説明
CREATE_OBJECT	このプロシージャを使用して、セグメント領域のアドバイス対象オブジェクトを識別します。このプロシージャのパラメータ値は、オブジェクト・タイプによって異なります。表 19-2 に、各オブジェクト・タイプのパラメータ値を示します。 ノート: IOT オーバーフロー・セグメントについてアドバイスを要求するには、TABLE、TABLE PARTITION または TABLE SUBPARTITION のオブジェクト型を使用します。次の問合せを使用して、IOT のオーバーフロー・セグメントを検索し、CREATE_OBJECT で使用するオーバーフロー・セグメントの表名を判断します。 `select table_name, iot_name, iot_type from dba_tables;`
SET_TASK_PARAMETER	このプロシージャを使用して、必要なセグメント・アドバイスを指定します。表 19-3 に、このプロシージャに関連する入力パラメータを示します。リストされていないパラメータは、セグメント・アドバイザでは使用されません。
EXECUTE_TASK	このプロシージャを使用して、セグメント・アドバイザのタスクを実行します。

表19-2 DBMS_ADVISOR.CREATE_OBJECTの入力パラメータ

OBJECT_TYPE	ATTR1	ATTR2	ATTR3	ATTR4
TABLESPACE	表領域名	NULL	NULL	未使用。NULL を指定します。
TABLE	スキーマ名	表名	NULL	未使用。NULL を指定します。
INDEX	スキーマ名	索引名	NULL	未使用。NULL を指定します。
TABLE PARTITION	スキーマ名	表名	表パーティション名	未使用。NULL を指定します。
INDEX PARTITION	スキーマ名	索引名	索引パーティション名	未使用。NULL を指定します。
TABLE SUBPARTITION	スキーマ名	表名	表サブパーティション名	未使用。NULL を指定します。
INDEX SUBPARTITION	スキーマ名	索引名	索引サブパーティション名	未使用。NULL を指定

OBJECT_TYPE	ATTR1	ATTR2	ATTR3	ATTR4
				します。
LOB	スキーマ名	セグメント名	NULL	未使用。NULL を指定します。
LOB PARTITION	スキーマ名	セグメント名	LOB パーティション名	未使用。NULL を指定します。
LOB SUBPARTITION	スキーマ名	セグメント名	LOB サブパーティション名	未使用。NULL を指定します。

表19-3 DBMS_ADVISOR.SET_TASK_PARAMETERの入力

入力パラメータ	説明	使用可能な値	デフォルト値
time_limit	セグメント・アドバイザを実行する時間制限を秒単位で指定します。	任意の秒数。	UNLIMITED
recommend_all	セグメント・アドバイザですべてのセグメントについて結果を生成するかどうかを指定します。	TRUE: 領域再生の推奨事項に関係なく、指定されたすべてのセグメントに対して結果が生成されます。 FALSE: 領域再生の推奨事項を生成するオブジェクトに対してのみ結果が生成されます。	TRUE

例

次の例は、DBMS_ADVISORプロシージャを使用して、サンプル表hr.employeesに対してセグメント・アドバイザを実行する方法を示しています。これらのパッケージ・プロシージャを実行するユーザーには、そのパッケージに対するオブジェクトのEXECUTE権限、またはADVISORシステム権限が必要です。

オブジェクト型TABLEをDBMS_ADVISOR.CREATE_OBJECTに渡すと、オブジェクト・レベルを要求したことになります。表がパーティション化されていない場合は、表セグメントが分析されます(索引またはLOBセグメントなどの依存セグメントは対象外です)。表がパーティション化されている場合は、すべての表パーティションが分析され、個別に結果と推奨事項が生成されます。

```
variable id number;
begin
  declare
  name varchar2(100);
  descr varchar2(500);
  obj_id number;
  begin
```

```
  name:='Manual_Employees';
  descr:='Segment Advisor Example';
  dbms_advisor.create_task (
    advisor_name     => 'Segment Advisor',
    task_id          => :id,
    task_name        => name,
    task_desc        => descr);
  dbms_advisor.create_object (
    task_name        => name,
    object_type      => 'TABLE',
    attr1            => 'HR',
    attr2            => 'EMPLOYEES',
    attr3            => NULL,
    attr4            => NULL,
    attr5            => NULL,
    object_id        => obj_id);
  dbms_advisor.set_task_parameter(
    task_name        => name,
    parameter        => 'recommend_all',
    value            => 'TRUE');
  dbms_advisor.execute_task(name);
  end;
end;
/
```

親トピック: 手動によるセグメント・アドバイザの実行

19.3.2.5 セグメント・アドバイザの結果の表示

セグメント・アドバイザでは、様々な種類の結果(推奨事項、結果、処置、オブジェクト)が作成されます。

次の方法で結果を表示できます。

- Cloud Controlの使用
- DBA_ADVISOR_*ビューの問合せ
- DBMS_SPACE.ASA_RECOMMENDATIONSファンクションのコール

表19-4に、様々な種類の結果および関連するDBA_ADVISOR_*ビューを示します。

表19-4 セグメント・アドバイザの結果の種類

結果の種類	関連するビュー	説明
推奨事項	DBA_ADVISOR_RECOMMENDATIONS	セグメントの縮小、再編成または圧縮が効果的な場合は、そのセグメントに対して推奨事項が生成されます。表 19-5 に、生成される結果および推奨事項の例を示します。
結果	DBA_ADVISOR_FINDINGS	結果とは、分析対象のセグメントでセグメント・アドバイザが観察した内容のレポートです。結果には、分析対象の各セグメントの使用領域と空き領域の統計情報が含まれます。すべての結果が推奨事項になるわけではありません。(推奨事項は少数でも、多くの結果がある場合もあります。)セグメント・アドバイザをPL/SQL を使用して手動で実行する際に、SET_TASK_PARAMETER プロシージャの recommend_all を'TRUE'に指定すると、セグメントに対

結果の種類	関連するビュー	説明
		する推奨事項があってもなくても、分析の対象になる各セグメントに対する結果が生成されます。行連鎖アドバイスの場合、自動セグメント・アドバイザは結果のみを生成し、推奨事項は生成しません。自動セグメント・アドバイザに領域の再生に関する推奨事項がない場合、結果は生成されません。ただし、高度な行圧縮の効果が期待できる表に対しては、自動セグメント・アドバイザの結果が生成される場合があります。
処置	DBA_ADVISOR_ACTIONS	すべての推奨事項は、セグメントの縮小、オンライン再定義(再編成)または圧縮の実施を提案する処置に関連付けられます。DBA_ADVISOR_ACTIONS ビューには、セグメントの縮小または表の圧縮を実行するために使用できる SQL、またはオブジェクトを再編成するための提案が表示されます。
オブジェクト	DBA_ADVISOR_OBJECTS	すべての結果、推奨事項および処置はオブジェクトに関連付けられています。表領域やパーティション表の場合のように、複数のセグメントをセグメント・アドバイザが分析する場合、分析対象のセグメントごとに 1 つのエントリが DBA_ADVISOR_OBJECTS ビューに作成されます。表 19-2 には、分析対象のセグメントに関する情報を問い合せるための、このビューの列が定義されています。結果、推奨事項および処置のビューのオブジェクトを、このビューのオブジェクトと関係づけることができます。

- Cloud Controlを使用したセグメント・アドバイザの結果の表示
  Cloud Controlでは、自動セグメント・アドバイザの実行とセグメント・アドバイザの手動実行の両方についてセグメント・アドバイザの結果を表示できます。
- DBA_ADVISOR_*ビューの問合せによるセグメント・アドバイザの結果の表示
  DBA_ADVISOR_*ビューを問い合せて、セグメント・アドバイザの結果を表示できます。
- DBMS_SPACE.ASA_RECOMMENDATIONSを使用したセグメント・アドバイザの結果の表示
  DBMS_SPACEパッケージのASA_RECOMMENDATIONSプロシージャは、自動セグメント・アドバイザの実行およびセグメント・アドバイザの手動実行(オプション)に対する結果や推奨事項を含むネストされた表オブジェクトを返します。

関連項目:

DBMS_SPACE.ASA_RECOMMENDATIONSファンクションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

親トピック: セグメント・アドバイザ

19.3.2.5.1 Cloud Controlを使用したセグメント・アドバイザの結果の表示

Cloud Controlでは、自動セグメント・アドバイザの実行とセグメント・アドバイザの手動実行の両方についてセグメント・アドバ

イザの結果を表示できます。

次の種類の結果を表示できます。

- すべての推奨事項(自動および手動によるセグメント・アドバイザの複数の実行)
- 自動セグメント・アドバイザの最新の実行での推奨事項
- 特定の実行での推奨事項
- 行連鎖の結果

自動セグメント・アドバイザの最新の実行で分析されたセグメントを一覧に表示することもできます。

Cloud Controlを使用してセグメント・アドバイザの結果を表示するには-すべての実行:

1. データベース・ホームページにアクセスします。
2. 「管理」メニューから、「記憶域」を選択し、「セグメント・アドバイザ」を選択します。

   「セグメント・アドバイザ推奨」ページが表示されます。推奨事項は、表領域別に編成されます。
3. 推奨事項が提示されている場合は、表領域を選択し、次に「推奨事項の詳細」をクリックします。

   「推奨事項の詳細」ページが表示されます。このページから推奨事項アクティビティ(縮小または再編成)を開始できます。

ノート:

エラーが発生するため、LOB を縮小または再編成しないでください。

ヒント:

リストのエントリは、再生可能領域の大きい順にソートされています。列ヘッダーをクリックすると、ソート順を変更したり、昇順から降順に変更できます。

Cloud Controlを使用してセグメント・アドバイザの結果を表示するには-自動セグメント・アドバイザの最新の実行:

1. データベース・ホームページにアクセスします。
2. 「管理」メニューから、「記憶域」を選択し、「セグメント・アドバイザ」を選択します。

   「セグメント・アドバイザ推奨」ページが表示されます。推奨事項は、表領域別に編成されます。

   「セグメント・アドバイザ推奨」ページが表示されます。
3. 「表示」リストで、「最後の自動実行からの推奨事項」を選択します。
4. 推奨事項が提示されている場合は、表領域を選択し、「推奨事項の詳細」をクリックします。

   「推奨事項の詳細」ページが表示されます。このページから推奨事項アクティビティ(縮小または再編成)を開始できます。

ノート:

エラーが発生するため、LOB を縮小または再編成しないでください。

Cloud Controlを使用してセグメント・アドバイザの結果を表示するには-特定の実行:

1. データベース・ホームページにアクセスします。
2. 「パフォーマンス」メニューから、「アドバイザ・ホーム」を選択します。

   「アドバイザ・セントラル」ページが表示されます。(図19-2を参照してください。)
3. タスクが「結果」ヘッダーの下のリストに表示されていることを確認します。タスクがない場合は、次のステップを実行します。
   a. ページの「検索」セクションのアドバイザ・タイプで、「セグメント・アドバイザ」を選択します。
   b. 「アドバイザ実行」リストで、「すべて」または該当する期間を選択します。
   c. (オプション)タスク名を入力します。
   d. 「実行」をクリックします。

      「結果」セクションにセグメント・アドバイザのタスクが表示されます。
4. ジョブのステータスをチェックします。COMPLETEDでない場合は、ページの上部にある「リフレッシュ」コントロールを使用してページをリフレッシュします。(ブラウザのリフレッシュ・アイコンは使用しないでください。)
5. タスク名をクリックします。

   セグメント・アドバイザのタスクのページに、表領域別に編成された推奨事項が表示されます。
6. リストで表領域を選択して、「推奨事項の詳細」をクリックします。

   「推奨事項の詳細」ページが表示されます。このページから推奨事項アクティビティ(縮小または再編成)を開始できます。

行連鎖の結果を表示するには:

1. データベース・ホームページにアクセスします。
2. 「管理」メニューから、「記憶域」を選択し、「セグメント・アドバイザ」を選択します。

   「セグメント・アドバイザ推奨」ページが表示されます。推奨事項は、表領域別に編成されます。

   「セグメント・アドバイザ推奨」ページが表示されます。
3. 「関連リンク」ヘッダーの下にある「行チェーン分析」をクリックします。

   「行チェーン分析」ページには、連鎖行があるすべてのセグメントと各連鎖行の割合が表示されます。

親トピック: セグメント・アドバイザの結果の表示

19.3.2.5.2 DBA_ADVISOR_*ビューの問合せによるセグメント・アドバイザの結果の表示

DBA_ADVISOR_*ビューを問い合せて、セグメント・アドバイザの結果を表示できます。

表19-5のヘッダーには、セグメント・アドバイザからの出力が表示されるDBA_ADVISOR_*ビューの列が示されます。これらのビューの詳細は、『Oracle Databaseリファレンス』を参照してください。この表に、考えられる出力の要約を示します。また、表19-2に、分析されたセグメントの情報が表示されるDBA_ADVISOR_OBJECTSビューの列を示します。

DBA_ADVISOR_*ビューを問い合せる前に、DBA_ADVISOR_TASKSのSTATUS列を問い合せることで、セグメント・アドバイザのタスクが完了していることを確認できます。

```
select task_name, status from dba_advisor_tasks
   where owner = 'STEVE' and advisor_name = 'Segment Advisor';

TASK_NAME                      STATUS
------------------------------ -----------
Manual Employees               COMPLETED
```

次の例は、ユーザーSTEVEが発行したセグメント・アドバイザのすべての実行から結果を取得するために、DBA_ADVISOR_* ビューを問い合せる方法を示しています。

```
select af.task_name, ao.attr2 segname, ao.attr3 partition, ao.type, af.message
  from dba_advisor_findings af, dba_advisor_objects ao
  where ao.task_id = af.task_id
  and ao.object_id = af.object_id
  and ao.owner = 'STEVE';

TASK_NAME          SEGNAME      PARTITION       TYPE              MESSAGE
------------------ ------------ --------------- ----------------- --------------------
------
Manual_Employees   EMPLOYEES                    TABLE             The free space in
the obje
                                                                  ct is less than 10MB.

Manual_Salestable4 SALESTABLE4  SALESTABLE4_P1  TABLE PARTITION  Perform shrink,
estimated
                                                                  savings is 74444154
bytes.

Manual_Salestable4 SALESTABLE4  SALESTABLE4_P2  TABLE PARTITION  The free space in
the obje
                                                                  ct is less than 10MB.
```

表19-5 セグメント・アドバイザの結果: 要約

DBA_ADVISOR_FINDINGSのMESSAGE列	DBA_ADVISOR_FINDINGSのMORE_INFO列	DBA_ADVISOR_RECOMMENDATIONSのBENEFIT_TYPE列	DBA_ADVISOR_ACTIONSのATTR1列
情報が不十分なため、推奨事項を作成できません。	-	-	-
オブジェクト内の空き領域が10MB 以下です。	割当領域: xxx: 使用領域: xxx: 再利用可能領域 : xxx	-	-
オブジェクト内には空き領域がありますが、...のため縮小できません。	割当領域: xxx: 使用領域: xxx: 再利用可能領域 : xxx	-	-
オブジェクト内の空き領域は前回のエクステントのサイズ未満です。	割当領域: xxx: 使用領域: xxx: 再利用可能領域 : xxx	-	-
xxx バイトの節約が予測さ	割当領域: xxx: 使用領	xxx バイトの節約が予測される	実行するコマンド。たとえば:

DBA_ADVISOR_FINDINGSのMESSAGE列	DBA_ADVISOR_FINDINGSのMORE_INFO列	DBA_ADVISOR_RECOMMENDATIONSのBENEFIT_TYPE列	DBA_ADVISOR_ACTIONSのATTR1列
れるため、縮小を実行してください。	域: xxx: 再利用可能領域 : xxx	ため、縮小を実行してください。	ALTER object SHRINK SPACE;)
表 schema.table の行移動を有効にして縮小を実行してください。xxx バイトの節約が予測されるためです。	割当領域: xxx: 使用領域: xxx: 再利用可能領域 : xxx	表 schema.table の行移動を有効にして縮小を実行してください。xxx バイトの節約が予測されるためです。	実行するコマンド。たとえば: ALTER object SHRINK SPACE;)
オブジェクト object の再編成を実行してください。xxx バイトの節約が予測されるためです。 (ノート: これは、オンラインによるセグメントの縮小に適していない再生可能な領域を持つオブジェクトに対する結果です。)	割当領域: xxx: 使用領域: xxx: 再利用可能領域 : xxx	オブジェクト object の再編成を実行してください。xxx バイトの節約が予測されるためです。	再編成の実行
オブジェクトには、再編成によって削除できる連鎖行があります。	xx パーセントの連鎖行を再編成で削除できます。	-	-
オブジェクト object_name を圧縮してください。xxx バイトの節約が予測されるためです。 (この結果は、自動セグメント・アドバイザによってのみ生成されます)	オブジェクト object_name を圧縮してください。xxx バイトの節約が予測されるためです。	-	実行するコマンド。たとえば: ALTER TABLE T1 ROW STORE COMPRESS ADVANCED この結果については、DBA_ADVISOR_ACTIONS の ATTR2 列も参照してください。

親トピック: セグメント・アドバイザの結果の表示

19.3.2.5.3 DBMS_SPACE.ASA_RECOMMENDATIONSを使用したセグメント・アドバイザの結果の表示

DBMS_SPACEパッケージのASA_RECOMMENDATIONSプロシージャは、自動セグメント・アドバイザの実行およびセグメント・アドバイザの手動実行に対する結果や推奨事項を含むネストされた表オブジェクトを返します。

このプロシージャでは、必要な結合がすべて実行され、使用しやすい形式で情報が返されるため、このプロシージャを呼び出す方が、DBA_ADVISOR_*ビューで作業するより簡単な場合があります。

次の問合せは、自動セグメント・アドバイザの最新の実行による推奨事項と、推奨事項に従うための実行コマンドの例を戻します。

```
select tablespace_name, segment_name, segment_type, partition_name,
recommendations, c1 from
table(dbms_space.asa_recommendations('FALSE', 'FALSE', 'FALSE'));
TABLESPACE_NAME                SEGMENT_NAME                    SEGMENT_TYPE
------------------------------ ------------------------------ --------------
PARTITION_NAME
------------------------------
RECOMMENDATIONS
--------------------------------------------------------------------------------
C1
--------------------------------------------------------------------------------
TVMDS_ASSM                     ORDERS1                         TABLE PARTITION
ORDERS1_P2
Perform shrink, estimated savings is 57666422 bytes.
alter table "STEVE"."ORDERS1" modify partition "ORDERS1_P2" shrink space

TVMDS_ASSM                     ORDERS1                         TABLE PARTITION
ORDERS1_P1
Perform shrink, estimated savings is 45083514 bytes.
alter table "STEVE"."ORDERS1" modify partition "ORDERS1_P1" shrink space

TVMDS_ASSM_NEW                 ORDERS_NEW                      TABLE

Perform shrink, estimated savings is 155398992 bytes.
alter table "STEVE"."ORDERS_NEW" shrink space

TVMDS_ASSM_NEW                 ORDERS_NEW_INDEX                INDEX

Perform shrink, estimated savings is 102759445 bytes.
alter index "STEVE"."ORDERS_NEW_INDEX" shrink space
```

DBMS_SPACE.ASA_RECOMMENDATIONSの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: セグメント・アドバイザの結果の表示

19.3.2.6 自動セグメント・アドバイザの構成

自動セグメント・アドバイザは自動化メンテナンス・タスクの1つです。そのため、このタスクの実行時に変更を加える場合は、Cloud ControlまたはPL/SQLパッケージ・プロシージャ・コールを使用できます。適切なリソース・プランを変更することで、タスクに割り当てられているリソースを制御することもできます。

これらはPL/SQLパッケージ・プロシージャをコールして変更できますが、Cloud Controlを使用する方がさらに簡単です。

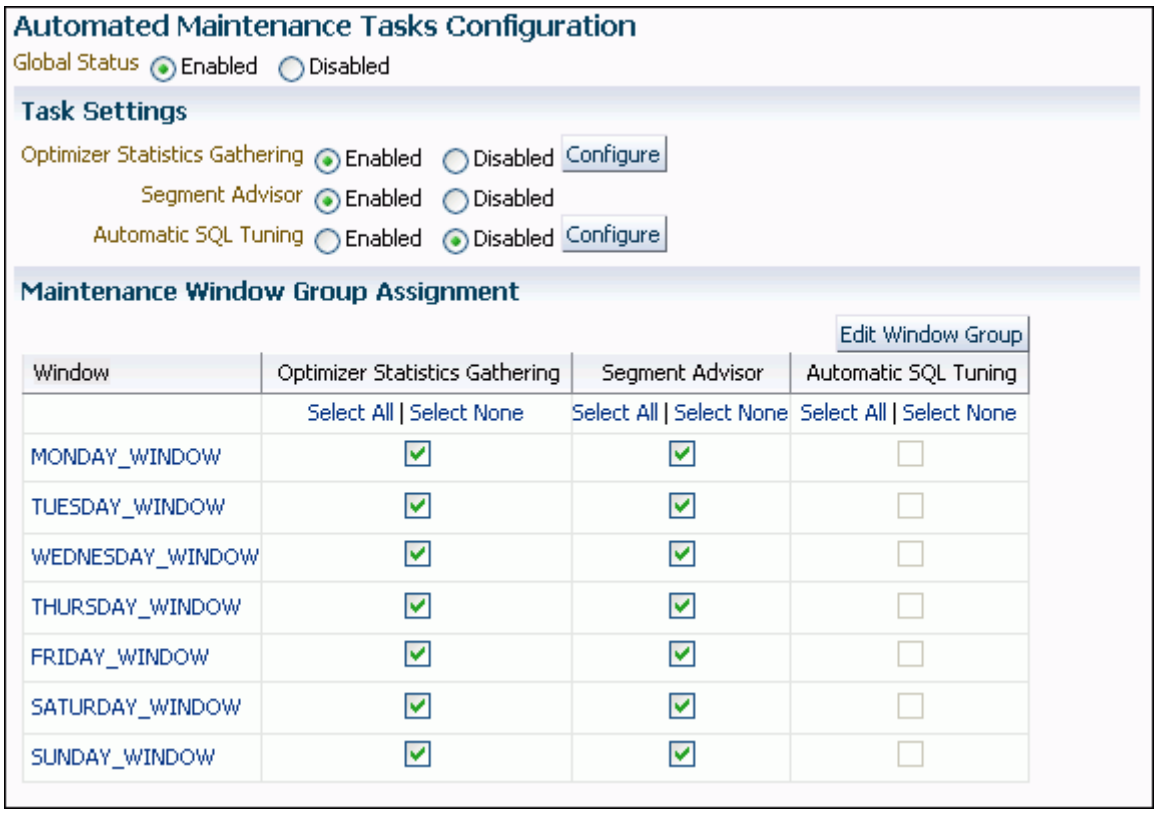
Cloud Controlを使用して自動セグメント・アドバイザのタスクを構成するには:

1. ユーザーSYSTEMとしてCloud Controlにログインします。
2. 「データベース・ホーム」ページにアクセスします。
3. 「管理」メニューから、「記憶域」を選択し、「セグメント・アドバイザ」を選択します。

   「セグメント・アドバイザ推奨」ページが表示されます。
4. 「関連リンク」ヘッダーの下にある「自動化メンテナンス・タスク」リンクをクリックします。

   自動化メンテナンス・タスク・ページが表示されます。
5. 「構成」をクリックします。

   「自動化メンテナンス・タスク構成」ページが表示されます。



6. 自動セグメント・アドバイザを完全に無効化するには、「タスク設定」の下にある「セグメント・アドバイザ」ラベルの横の「無効」を選択して、「適用」をクリックします。
7. 特定のメンテナンス・ウィンドウの自動セグメント・アドバイザを無効化するには、「セグメント・アドバイザ」列の下の該当するチェック・ボックスをクリアし、「適用」をクリックします。
8. メンテナンス・ウィンドウの開始時間、終了時間および継続時間を変更するには、「ウィンドウ・グループの編集」をクリックします。

   ウィンドウ・グループの編集ページが表示されます。メンテナンス・ウィンドウの名前をクリックして「編集」をクリックし、ウィンドウのスケジュールを変更します。

関連項目:

- 自動データベース・メンテナンス・タスクの管理

親トピック: セグメント・アドバイザ

19.3.2.7 自動セグメント・アドバイザ情報の表示

ビューを問い合せて、自動セグメント・アドバイザに固有の情報を表示できます。

ビュー	説明
DBA_AUTO_SEGADV_SUMMARY	このビューには、各行に 1 件ずつ自動セグメント・アドバイザの実行が要約されます。フィールドには、処理された表領域やセグメントの数、および作成された推奨事項の件数が表示されます。
DBA_AUTO_SEGADV_CTL	自動セグメント・アドバイザがセグメントの選択および処理に使用する制御情報が表示されます。各行には、単一のオブジェクト(表領域またはセグメ

ビュー	説明
	ント)に関する情報(オブジェクトが処理されたかどうか、処理された場合はオブジェクトを処理したタスク ID やその選択理由など)が格納されます。

親トピック: セグメント・アドバイザ

19.3.3 オンラインによるデータベース・セグメントの縮小

Oracle Databaseセグメントの最高水位標の下の断片化された空き領域を再生するには、オンラインによるセグメントの縮小を使用します。

セグメントの縮小の利点は、次のとおりです。

- データの縮小はキャッシュ利用の向上につながります。つまり、オンライン・トランザクション処理(OLTP)のパフォーマンスが改善されます。
- データが縮小されることで、全表スキャンを実行する際に必要なスキャン・ブロック数が少なくなります。つまり、意思決定支援システム(DSS)のパフォーマンスが改善されます。

セグメントの縮小は、オンラインのインプレース操作です。セグメント縮小のデータ移動フェーズでは、DML操作および問合せを発行できます。縮小操作の最後に領域が割当て解除される際は、短い時間ですが同時DML操作がブロックされます。索引は、縮小操作中も保持され、操作が完了した後も使用できます。セグメントの縮小では、余分なディスク領域の割当ては不要です。

セグメントの縮小では、最高水位標の上下両方の未使用領域を再生します。これに対して、領域の割当て解除では、最高水位標よりも上の未使用領域のみを再生します。縮小操作では、デフォルトの場合、セグメントを縮小して最高水位標を調整し、回復した領域が解放されます。

セグメントの縮小には、新しい位置への行の移動が必要です。したがって、最初に、縮小するオブジェクトの行を移動可能にし、オブジェクトに定義したROWIDベースのトリガーを無効にする必要があります。表内の行を移動可能にするには、`ALTER TABLE ... ENABLE ROW MOVEMENT`コマンドを使用します。

縮小操作は、自動セグメント領域管理(ASSM)を使用しているローカル管理表領域内のセグメントに対してのみ実行できます。ASSM表領域内では、次の表を除くすべてのセグメント・タイプがオンラインによるセグメントの縮小に適しています。

- IOTマッピング表
- ROWIDベースのマテリアライズド・ビューを備えた表
- ファンクション索引がある表
- `SECUREFILE` LOB
- 次の圧縮メソッドで圧縮された表:
  - `ROW STORE COMPRESS BASIC`を使用した基本表圧縮
  - `COLUMN STORE COMPRESS FOR QUERY`を使用したウェアハウス圧縮
  - `COLUMN STORE COMPRESS FOR ARCHIVE`を使用したアーカイブ圧縮

  ただし、`ROW STORE COMPRESS ADVANCED`を使用した高度な行圧縮で圧縮された表は、オンラインのセグメント縮小の対象になります。表圧縮メソッドの詳細は、「表圧縮の使用」を参照してください。

ノート:

オンラインでデータベース・セグメントを縮小すると、依存データベース・オブジェクトが無効になることがあります。「オブジェクト依存性とオブジェクトの無効化について」を参照してください。

関連項目:

ALTER TABLEコマンドの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

オンラインによるセグメントの縮小の起動

オンラインによるセグメントの縮小を起動する前に、セグメント・アドバイザの結果および推奨事項を表示します。詳細は、「セグメント・アドバイザの使用」を参照してください。

Cloud ControlまたはSQL*PlusのSQLコマンドを使用して、オンラインによるセグメントの縮小を起動します。ここからは、コマンドラインによる方法について説明します。

ノート:

Cloud Control の「推奨事項の詳細」ページから、セグメントの縮小を直接起動できます。また、Cloud Control の個々の表に対してセグメントの縮小を起動するには、「表」ページに表を表示し、表を選択した後、「アクション」リストで「セグメントの縮小」をクリックします。(図 19-1 を参照してください。)Cloud Control で同様の操作を実行すると、索引やマテリアライズド・ビューなどを縮小できます。

表、索引構成表、索引、パーティション、サブパーティション、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログの領域を縮小できます。この縮小には、SHRINK SPACE句を指定したALTER TABLE、ALTER INDEX、ALTER MATERIALIZED VIEW文またはALTER MATERIALIZED VIEW LOG文を使用します。

次の2つの句を必要に応じて使用することで、縮小操作の処理方法を制御できます。

- COMPACT句を指定すると、セグメントの縮小操作を2つのフェーズに分割できます。COMPACTを指定すると、Oracle Databaseはセグメント領域の断片化を解消して表の行を縮小しますが、将来のある時点まで最高水位標のリセットおよび領域の割当て解除を延期します。このオプションは、操作に影響される可能性がある長時間実行される問合せがあり、再生されたブロックから読取りが試行される場合に役立ちます。断片化解消および縮小の結果は、第2のフェーズでデータの移動をやり直す必要がないように、ディスクに保存されます。第2のフェーズを完了するには、オフピーク時にCOMPACT句を指定せずにSHRINK SPACE句を再度発行します。
- CASCADE句を指定すると、オブジェクトのすべての依存セグメントにセグメントの縮小操作が拡張されます。たとえば、表セグメントを縮小する際にCASCADEを使用すると、表のすべての索引も縮小されます。(パーティション表のパーティションを縮小するためにCASCADEを指定する必要はありません。)指定したオブジェクトの依存セグメントのリストを表示するには、DBMS_SPACEパッケージのOBJECT_DEPENDENT_SEGMENTSプロシージャを実行できます。

DDL操作と同様に、セグメントの縮小によって、後続のSQL文が再度解析されることになります。これは、COMPACT句を指定しないかぎり、カーソルが無効になるためです。

例

表およびその依存セグメント(BASICFILE LOBセグメントを含む)のすべてを縮小します。

```
ALTER TABLE employees SHRINK SPACE CASCADE;
```

BASICFILE LOBセグメントのみを縮小します。

```
ALTER TABLE employees MODIFY LOB (perf_review) (SHRINK SPACE);
```

パーティション表の単一パーティションを縮小します。

```
ALTER TABLE customers MODIFY PARTITION cust_P1 SHRINK SPACE;
```

IOT索引セグメントとオーバーフロー・セグメントを縮小します。

```
ALTER TABLE cities SHRINK SPACE CASCADE;
```

IOTオーバーフロー・セグメントのみを縮小します。

```
ALTER TABLE cities OVERFLOW SHRINK SPACE;
```

関連項目:

- SHRINK SPACE句のあるALTER TABLE、ALTER INDEX、ALTER MATERIALIZED VIEWおよびALTER MATERIALIZED VIEW LOG文の構文と制限事項は、『Oracle Database SQL言語リファレンス』を参照してください。
- LOBセグメントの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

親トピック: 未使用領域の再利用

19.3.4 未使用領域の割当て解除

未使用領域の割当てを解除する場合は、未使用の(最高水位標)データベース・セグメントの最後で未使用領域を解放して、表領域の他のセグメントで領域を使用できるようにします。

割当て解除の前に、セグメントの最高水位標の位置および未使用領域の量に関する情報を返す、DBMS_SPACEパッケージのUNUSED_SPACEプロシージャを実行できます。自動セグメント領域管理を備えたローカル管理表領域のセグメントの場合は、SPACE_USAGEプロシージャを使用すると、未使用領域に関するより正確な情報が得られます。

関連項目:

DBMS_SPACEパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

次の文は、セグメント(表、索引またはクラスタ)内の未使用領域の割当てを解除します。

```
ALTER TABLE table DEALLOCATE UNUSED KEEP integer;
ALTER INDEX index DEALLOCATE UNUSED KEEP integer;
ALTER CLUSTER cluster DEALLOCATE UNUSED KEEP integer;
```

KEEP句はオプションで指定でき、セグメントに保持される領域の量を指定できます。DBA_FREE_SPACEビューを調べることで、割当て解除された領域が開放されたことを確認できます。

関連項目:

- 未使用領域の割当て解除の構文とセマンティクスの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- DBA_FREE_SPACEビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 未使用領域の再利用

19.4 未使用オブジェクト記憶域の削除

DBMS_SPACE_ADMINパッケージにはDROP_EMPTY_SEGMENTSプロシージャが含まれており、このプロシージャを使用すると、以前のリリースから移行された空の表およびパーティションのセグメントを削除できます。可能な場合には、索引セグメントなどの表の依存オブジェクトのセグメントもこれに含まれます。

次の例では、データベース内のすべての表から空のセグメントを削除しています。

```
BEGIN
  DBMS_SPACE_ADMIN.DROP_EMPTY_SEGMENTS();
END;
```

次は、HR.EMPLOYEES表から依存オブジェクトを含め空のセグメントを削除しています。

```
BEGIN
  DBMS_SPACE_ADMIN.DROP_EMPTY_SEGMENTS(
    schema_name  => 'HR',
    table_name   => 'EMPLOYEES');
END;
```

このプロシージャには、11.2.0以降との互換性が必要です。

関連項目:

このプロシージャの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

親トピック: スキーマ・オブジェクトの領域の管理

19.5 データ型の領域使用の理解

表またはその他のデータ構造を作成する際には、必要になる領域の大きさを把握しておく必要があります。領域要件は、データ型ごとに異なります。

『Oracle Database PL/SQL言語リファレンス』および『Oracle Database SQL言語リファレンス』には、データ型および領域要件の詳細な説明があります。

親トピック: スキーマ・オブジェクトの領域の管理

19.6 スキーマ・オブジェクトの領域使用情報の表示

Oracle Databaseには、スキーマ・オブジェクトの領域使用に関する情報を表示するためのデータ・ディクショナリ・ビューとPL/SQLパッケージが用意されています。

- PL/SQLパッケージを使用したスキーマ・オブジェクトの領域使用情報の表示
  DBMS_SPACEサブプログラムのセットを使用して、スキーマ・オブジェクトに関する情報を表示できます。
- スキーマ・オブジェクトの領域使用のデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューのセットに、スキーマ・オブジェクトの領域使用に関する情報が表示されます。

親トピック: スキーマ・オブジェクトの領域の管理

19.6.1 PL/SQLパッケージを使用したスキーマ・オブジェクトの領域使用情報の表示

DBMS_SPACEサブプログラムのセットを使用して、スキーマ・オブジェクトに関する情報を表示できます。

パッケージとプロシージャ/ファンクション	説明
DBMS_SPACE.UNUSED_SPACE	オブジェクト(表、索引またはクラスタ)の未使用領域に関する情報を返します。
DBMS_SPACE.FREE_BLOCKS	セグメントの空き領域が空きリストで管理されている(つまり、セグメント領域管理が MANUAL である)オブジェクト(表、索引またはクラスタ)の空きデータ・ブロックに関する情報を返します。
DBMS_SPACE.SPACE_USAGE	セグメント領域管理が AUTO であるオブジェクト(表、索引またはクラスタ)の空きデータ・ブロックに関する情報を返します。

関連項目:

DBMS_SPACEパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

例: DBMS_SPACE.UNUSED_SPACEの使用

次のSQL*Plusの例では、DBMS_SPACEパッケージを使用して、未使用領域情報を取得しています。

```
SQL> VARIABLE total_blocks NUMBER
SQL> VARIABLE total_bytes NUMBER
SQL> VARIABLE unused_blocks NUMBER
SQL> VARIABLE unused_bytes NUMBER
SQL> VARIABLE lastextf NUMBER
SQL> VARIABLE last_extb NUMBER
SQL> VARIABLE lastusedblock NUMBER
SQL> exec DBMS_SPACE.UNUSED_SPACE('SCOTT', 'EMP', 'TABLE', :total_blocks, -
>    :total_bytes,:unused_blocks, :unused_bytes, :lastextf, -
>    :last_extb, :lastusedblock);
PL/SQL procedure successfully completed.
SQL> PRINT
TOTAL_BLOCKS
------------
           5
TOTAL_BYTES
-----------
      10240
...
LASTUSEDBLOCK
-------------
            3
```

親トピック: スキーマ・オブジェクトの領域使用情報の表示

19.6.2 スキーマ・オブジェクトの領域使用のデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューのセットに、スキーマ・オブジェクトの領域使用に関する情報が表示されます。

次のビューには、スキーマ・オブジェクトの領域使用に関する情報が表示されます。

ビュー	説明
DBA_SEGMENTS USER_SEGMENTS	DBA ビューには、すべてのデータベース・セグメントに割り当てられている記憶域が表示されます。USER ビューには、現行のユーザーのセグメントに割り当てられている記憶域が表示されます。
DBA_EXTENTS USER_EXTENTS	DBA ビューには、データベース内のすべてのセグメントを構成するエクステントが表示されます。USER ビューには、現行のユーザーのセグメントを構成するエクステントが表示されます。
DBA_FREE_SPACE USER_FREE_SPACE	DBA ビューには、すべての表領域の使用可能エクステントが表示されます。USER ビューには、ユーザーが割当て制限を持つ表領域の空き領域情報が表示されます。

- 例1: セグメント情報の表示
  DBA_SEGMENTSビューを問い合せてセグメント情報を表示できます。
- 例2: エクステント情報の表示
  DBA_EXTENTSデータ・ディクショナリ・ビューを問い合せて、データベース内の現在割り当てられているエクステントに関する情報を表示できます。
- 例3: 表領域内の空き領域(エクステント)の表示
  DBA_FREE_SPACEデータ・ディクショナリ・ビューを問い合せて、データベース内の空きエクステント(セグメントに割り当てられていないエクステント)に関する情報を表示できます。

親トピック: スキーマ・オブジェクトの領域使用情報の表示

19.6.2.1 例1: セグメント情報の表示

DBA_SEGMENTSビューを問い合せてセグメント情報を表示できます。

次の問合せは、スキーマhrの各索引セグメントの名前とサイズを返します。

```
SELECT SEGMENT_NAME, TABLESPACE_NAME, BYTES, BLOCKS, EXTENTS
    FROM DBA_SEGMENTS
    WHERE SEGMENT_TYPE = 'INDEX'
    AND OWNER='HR'
    ORDER BY SEGMENT_NAME;
```

問合せ出力は、次のとおりです。

```
SEGMENT_NAME              TABLESPACE_NAME    BYTES BLOCKS EXTENTS
------------------------- --------------- -------- ------ -------
COUNTRY_C_ID_PK           EXAMPLE            65536     32       1
DEPT_ID_PK                EXAMPLE            65536     32       1
DEPT_LOCATION_IX          EXAMPLE            65536     32       1
EMP_DEPARTMENT_IX         EXAMPLE            65536     32       1
EMP_EMAIL_UK              EXAMPLE            65536     32       1
EMP_EMP_ID_PK             EXAMPLE            65536     32       1
EMP_JOB_IX                EXAMPLE            65536     32       1
EMP_MANAGER_IX            EXAMPLE            65536     32       1
EMP_NAME_IX               EXAMPLE            65536     32       1
JHIST_DEPARTMENT_IX       EXAMPLE            65536     32       1
JHIST_EMPLOYEE_IX         EXAMPLE            65536     32       1
```

```
JHIST_EMP_ID_ST_DATE_PK    EXAMPLE            65536     32       1
JHIST_JOB_IX               EXAMPLE            65536     32       1
JOB_ID_PK                  EXAMPLE            65536     32       1
LOC_CITY_IX                EXAMPLE            65536     32       1
LOC_COUNTRY_IX             EXAMPLE            65536     32       1
LOC_ID_PK                  EXAMPLE            65536     32       1
LOC_STATE_PROVINCE_IX      EXAMPLE            65536     32       1
REG_ID_PK                  EXAMPLE            65536     32       1
19 rows selected.
```

親トピック: スキーマ・オブジェクトの領域使用のデータ・ディクショナリ・ビュー

19.6.2.2 例2: エクステント情報の表示

DBA_EXTENTSデータ・ディクショナリ・ビューを問い合せて、データベース内の現在割り当てられているエクステントに関する情報を表示できます。

たとえば、次の問合せによって、hrスキーマの各索引セグメントに割り当てられているエクステントとそれらのエクステントのサイズが識別されます。

```
SELECT SEGMENT_NAME, SEGMENT_TYPE, TABLESPACE_NAME, EXTENT_ID, BYTES, BLOCKS
    FROM DBA_EXTENTS
    WHERE SEGMENT_TYPE = 'INDEX'
    AND OWNER='HR'
    ORDER BY SEGMENT_NAME;
```

問合せ出力は、次のとおりです。

```
SEGMENT_NAME              SEGMENT_TYPE TABLESPACE_NAME EXTENT_ID    BYTES BLOCKS
------------------------- ------------ --------------- --------- -------- ------
COUNTRY_C_ID_PK           INDEX        EXAMPLE                 0    65536     32
DEPT_ID_PK                INDEX        EXAMPLE                 0    65536     32
DEPT_LOCATION_IX          INDEX        EXAMPLE                 0    65536     32
EMP_DEPARTMENT_IX         INDEX        EXAMPLE                 0    65536     32
EMP_EMAIL_UK              INDEX        EXAMPLE                 0    65536     32
EMP_EMP_ID_PK             INDEX        EXAMPLE                 0    65536     32
EMP_JOB_IX                INDEX        EXAMPLE                 0    65536     32
EMP_MANAGER_IX            INDEX        EXAMPLE                 0    65536     32
EMP_NAME_IX               INDEX        EXAMPLE                 0    65536     32
JHIST_DEPARTMENT_IX       INDEX        EXAMPLE                 0    65536     32
JHIST_EMPLOYEE_IX         INDEX        EXAMPLE                 0    65536     32
JHIST_EMP_ID_ST_DATE_PK   INDEX        EXAMPLE                 0    65536     32
JHIST_JOB_IX              INDEX        EXAMPLE                 0    65536     32
JOB_ID_PK                 INDEX        EXAMPLE                 0    65536     32
LOC_CITY_IX               INDEX        EXAMPLE                 0    65536     32
LOC_COUNTRY_IX            INDEX        EXAMPLE                 0    65536     32
LOC_ID_PK                 INDEX        EXAMPLE                 0    65536     32
LOC_STATE_PROVINCE_IX     INDEX        EXAMPLE                 0    65536     32
REG_ID_PK                 INDEX        EXAMPLE                 0    65536     32
19 rows selected.
```

hrスキーマの場合は、複数のエクステントが割り当てられているセグメントはありません。

親トピック: スキーマ・オブジェクトの領域使用のデータ・ディクショナリ・ビュー

19.6.2.3 例3: 表領域内の空き領域(エクステント)の表示

DBA_FREE_SPACEデータ・ディクショナリ・ビューを問い合せて、データベース内の空きエクステント(セグメントに割り当てられていないエクステント)に関する情報を表示できます。

たとえば、次の問合せは、SMUNDO表領域内の使用可能エクステントとして使用可能な空き領域を示します。

```
SELECT TABLESPACE_NAME, FILE_ID, BYTES, BLOCKS
```

```
    FROM DBA_FREE_SPACE
    WHERE TABLESPACE_NAME='SMUNDO';
```

問合せ出力は、次のとおりです。

```
TABLESPACE_NAME  FILE_ID    BYTES BLOCKS
--------------- -------- -------- ------
SMUNDO                 3    65536     32
SMUNDO                 3    65536     32
SMUNDO                 3    65536     32
SMUNDO                 3    65536     32
SMUNDO                 3    65536     32
SMUNDO                 3    65536     32
SMUNDO                 3   131072     64
SMUNDO                 3   131072     64
SMUNDO                 3    65536     32
SMUNDO                 3  3407872   1664
10 rows selected.
```

親トピック: スキーマ・オブジェクトの領域使用のデータ・ディクショナリ・ビュー

19.7 データベース・オブジェクトの容量計画

Oracle Databaseでは、Cloud ControlまたはDBMS_SPACE PL/SQLパッケージの2つの方法でデータベース・オブジェクトの容量を計画できます。DBMS_SPACEパッケージには、新しいオブジェクトのサイズを予測したり、既存のデータベース・オブジェクトのサイズを監視できる3つのプロシージャがあります。

ここでは、PL/SQLによる方法について説明します。Cloud Controlを使用した容量計画の詳細は、Cloud Controlのオンライン・ヘルプおよび「セグメント・アドバイザの使用」を参照してください。

- 表の領域使用の見積り
  データベース表のサイズは、表領域の記憶域属性、表領域のブロック・サイズ、他の様々な要因によって大きく変化する可能性があります。DBMS_SPACEパッケージのCREATE_TABLE_COSTプロシージャを使用すると、表を作成する際の領域使用コストを予測できます。
- 索引の領域使用の見積り
  DBMS_SPACEパッケージのCREATE_INDEX_COSTプロシージャを使用すると、既存の表に索引を作成する際の領域使用コストを予測できます。
- オブジェクト増加傾向の取得
  DBMS_SPACEパッケージ・プロシージャのOBJECT_GROWTH_TREND関数では、特定の時点におけるオブジェクトの領域使用が各行に記述された1行以上の表が作成されます。

親トピック: スキーマ・オブジェクトの領域の管理

19.7.1 表の領域使用の見積り

データベース表のサイズは、表領域の記憶域属性、表領域のブロック・サイズ、他の様々な要因によって大きく変化する可能性があります。DBMS_SPACEパッケージのCREATE_TABLE_COSTプロシージャを使用すると、表を作成する際の領域使用コストを予測できます。

このプロシージャのパラメータの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

プロシージャには、2つの異形があります。第1の異形は、行の平均サイズを使用してサイズを見積ります。第2の異形は、列情報を使用して表のサイズを見積ります。両方の異形ともに入力として次の値が必要です。

- TABLESPACE_NAME: オブジェクトが作成される表領域。デフォルトはSYSTEM表領域です。

- ROW_COUNT: 表の予測行数。
- PCT_FREE: 更新の結果、既存の行を将来拡張するように各ブロックに確保する空き領域の割合(パーセンテージ)。

さらに、第1の異形には、AVG_ROW_SIZEの値も入力する必要があります。値は、予測された行サイズの平均をバイト単位で指定します。

また、第2の異形には、予想された各列値をCOLINFOSに指定する必要があります。この値は、属性COL_TYPE(列のデータ型)とCOL_SIZE(列の文字数またはバイト数)からなるオブジェクト型です。

このプロシージャでは、2つの値が戻ります。

- USED_BYTES: ブロック・メタデータ、PCT_FREE領域などのオーバーヘッドを含め、データとして使用される実際のバイト数。
- ALLOC_BYTES: 表領域のエクステントの特性を考慮してオブジェクトに割り当てられる予測した領域の大きさ。

ノート:

パーティション表のすべての新規セグメントについて、第 1 エクステントのデフォルトのサイズは 64KB ではなく 8MB です。このことは、パーティション表に対する挿入と問合せのパフォーマンス向上に役立ちます。パーティション表の初期サイズが大きくても、十分なデータが挿入されると、領域消費は以前のリリースと同じになります。このデフォルトは、表の記憶域句に INITIAL サイズを設定して上書きできます。この新しいデフォルトは、表パーティションおよび LOB パーティションにのみ適用されます。

親トピック: データベース・オブジェクトの容量計画

19.7.2 索引の領域使用の見積り

DBMS_SPACEパッケージのCREATE_INDEX_COSTプロシージャを使用すると、既存の表に索引を作成する際の領域使用コストを予測できます。

プロシージャには次の値を入力する必要があります。

- DDL: 索引を作成するCREATE INDEX文。このDDL文に指定する表は、既存の表であることが必要です。
- [オプション] PLAN_TABLE: 使用するPLAN TABLEの名前。デフォルトはNULLです。

このプロシージャから戻る結果は、セグメントに対して収集された統計によって異なります。したがって、このプロシージャを実行する直前に必ず統計を取得してください。最近の統計がない状態でもエラーになりませんが、不適切な結果が戻る可能性があります。このプロシージャでは、次の値が戻ります。

- USED_BYTES: 実際の索引データを表すバイト数。
- ALLOC_BYTES: 表領域の索引に割り当てられる領域の大きさ。

親トピック: データベース・オブジェクトの容量計画

19.7.3 オブジェクト増加傾向の取得

DBMS_SPACEパッケージ・プロシージャのOBJECT_GROWTH_TREND関数では、特定の時点におけるオブジェクトの領域使用が各行に記述された1行以上の表が作成されます。

この関数は、自動ワークロード・リポジトリから領域使用の合計を取得するか、現在の領域使用を計算して、自動ワークロード・

リポジトリから取得した領域使用の変更履歴と結合します。この関数のパラメータの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

関数には次の値を入力する必要があります。

- OBJECT_OWNER: オブジェクトの所有者。
- OBJECT_NAME: オブジェクトの名前。
- PARTITION_NAME: 適切な場合は、表または索引パーティションの名前を指定します。それ以外の場合は、NULLを指定します。
- OBJECT_TYPE: オブジェクトの種類。
- START_TIME: 増加傾向分析の開始を示すTIMESTAMP値。
- END_TIME: 増加傾向分析の終了を示すTIMESTAMP値。デフォルトはNOWです。
- INTERVAL: 関数が領域使用情報を取得するレポート間隔の長さ(分単位)。
- SKIP_INTERPOLATED: INTERVALの前後に記録された統計情報に基づいて、この関数が値を除外する('YES')かしない('NO')かを決定します。要求したレポート作成間隔と実際の記録間隔がどのように関連しているかをより明確に表示できるため、チャートではなく表として結果表が表示される場合には、この設定が便利です。

この関数は、1つの間隔のオブジェクトの領域使用情報が各行に示された表を返します。返される表が非常に大きい場合は、情報が作成されると同時に別のアプリケーションが使用できるように結果がパイプライン化されます。出力表には次の列があります。

- TIMEPOINT: レポート間隔の時間を示すTIMESTAMP値。

  最も古いオブジェクトの記録統計よりも前のTIME値の記録は作成されません。
- SPACE_USAGE: オブジェクト・データとして実際に使用されているバイト数。
- SPACE_ALLOC: その時点で表領域のオブジェクトに割り当てられていたバイト数。
- QUALITY: 要求されたレポート間隔と実際の統計記録が一致している程度を示す値。オブジェクト・サイズ使用統計には保証されているレポート間隔はなく、実際のレポート間隔は時間の経過およびオブジェクトによって変わるため、この情報が役立ちます。

  QUALITY列には、次の値が設定されます。

  - GOOD: 記録されたタイムスタンプが入力パラメータに指定したINTERVALの10%以内で、TIMEの値が記録統計に常に基づいている場合。
  - INTERPOLATED: 値はGOODの基準を満たしていないが、TIME値の前後の記録統計に基づいている場合。現在のインメモリー統計は、クラスタ内のすべてのインスタンスで収集でき、現時点の記録値として取り扱うことができます。
  - PROJECTION: 表が作成された時点ではTIME値が未来であった場合。Oracle Real Application Clusters環境では、統計の記録ルールによって、どのオブジェクトが選択されるかを各インスタンスが独自に選択できるようになっています。

  この関数から戻る出力は、Oracle RAC環境のすべてのインスタンスについて記録した値の集合です。各値は、GOODおよびINTERPOLATED値の組合せから計算できます。その値の少なくとも80%がGOODのインスタンス値から導出された場合、戻される集計値はGOODにマークされます。

親トピック: データベース・オブジェクトの容量計画

20 表の管理

表の管理には、表の作成、表のロード、表の変更および表の削除などのタスクが含まれます。

Live SQL:

この章の例に関連する Oracle Live SQL での例を参照して実行するには、Oracle Live SQL: 表の作成および変更にアクセスしてください。

- 表について
  表は、Oracle Databaseのデータ記憶域の基本単位です。データは、行および列に格納されます。
- 表を管理するためのガイドライン
  これらのガイドラインに従うことで、表の作成や表データのロード、更新および問合せを行うときに、表の管理が容易になり、パフォーマンスの向上にもつながります。
- 表の作成
  表はSQL文CREATE TABLEを使用して作成します。
- 表のロード
  ここでは、データを表にロードするための手法について説明します。
- バルク更新のパフォーマンス最適化
  DBMS_REDEFINITIONパッケージのEXECUTE_UPDATEプロシージャを使用して、表のバルク更新のパフォーマンスを最適化できます。REDOログに更新が記録されないため、パフォーマンスが最適化されます。
- 表に関する統計の自動収集
  PL/SQLパッケージDBMS_STATSを使用すると、コストベースの最適化に関する統計を生成および管理できます。このパッケージを使用して、統計の収集、変更、表示、エクスポート、インポートおよび削除ができます。また、すでに収集した統計を識別または命名する際も、このパッケージを使用できます。
- 表の変更
  表を変更するにはALTER TABLE文を使用します。表を変更するには、その表が自分のスキーマに含まれているか、その表のALTERオブジェクト権限またはALTER ANY TABLEシステム権限のいずれかを持っている必要があります。
- 表のオンライン再定義
  表の論理構造または物理構造を変更できます。
- エラーが発生した表の変更の調査と取消し
  表に対してエラーが発生する変更を調査して取り消せるようにするために、Oracle Databaseには、データベース・オブジェクトの過去の状態を表示したり、Point-in-Timeメディア・リカバリを使用せずにデータベース・オブジェクトを以前の状態に戻すために使用できる一連の機能が用意されています。これらの機能はOracle Flashback機能と呼ばれます。
- Oracle Flashback Tableを使用した表のリカバリ
  Oracle Flashback Tableを使用すると、表を以前の時点の状態にリストアできます。
- 表の削除
  不要になった表を削除するには、DROP TABLE文を使用します。
- フラッシュバック・ドロップの使用とリサイクル・ビンの管理
  表を削除した場合、その表に関連付けられている領域はデータベースによってすぐには削除されません。データベースによってこの表の名前が変更され、すべての関連オブジェクトとともにリサイクル・ビンへ入れられますが、後に表が誤って削

除されたことがわかった場合、リサイクル・ビンからリカバリすることができます。この機能はフラッシュバック・ドロップと呼ばれ、表のリストアにはFLASHBACK TABLE文が使用されます。

- 索引構成表の管理
  索引構成表の記憶域の編成はプライマリBツリー索引の変形です。ヒープ構成表とは異なり、データは主キーの順に格納されます。
- パーティション表の管理
  パーティション表では、データをパーティションと呼ばれるさらに小さく管理が容易な単位に分割でき、さらにサブパーティションに分割することもできます。各パーティションには、圧縮の有効化または無効化、圧縮のタイプ、物理記憶域設定、表領域など別個の物理属性を指定できるため、可用性およびパフォーマンスをより適切にチューニングできる構造になります。さらに、各パーティションを個別に管理できるため、バックアップや管理が簡素化され、これらの処理に必要な時間を削減できます。
- 外部表の管理
  外部表は、データベースに存在しない表です。これらは、データベース外部の、オブジェクト記憶域または外部ファイル(オペレーティング・システム・ファイルやHadoop Distributed File System (HDFS)ファイルなど)に存在します。
- ハイブリッド・パーティション表の管理
  ハイブリッド・パーティション表は、一部のパーティションがデータベース内にあり一部のパーティションがデータベース外部の外部ファイル(オペレーティング・システム・ファイルやHadoop Distributed File System (HDFS)ファイルなど)内にあるパーティション表です。
- 不変表の管理
  不変表を使用すると、不正なデータ変更から保護できます。
- ブロックチェーン表の管理
  ブロックチェーン表は、重要なアクション、資産、エンティティおよびドキュメントを記録するデータを、犯罪者、ハッカーおよび不正行為による不正な変更または削除から保護します。ブロックチェーン表は、データベースを使用した不正な変更を防ぎ、データベースをバイパスする不正な変更を検出します。
- 表のデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューのセットを問い合せて、表についてのデータを取得できます。

親トピック: スキーマ・オブジェクト

20.1 表について

表は、Oracle Databaseのデータ記憶域の基本単位です。データは、行および列に格納されます。

表は、`employees`などの表名と一連の列で定義します。各列には列名(`employee_id`、`last_name`、`job_id`など)、データ型(VARCHAR2、DATE、NUMBERなど)および幅を指定します。幅は、DATEデータ型の場合のようにデータ型によって事前に決定されている場合があります。NUMBERデータ型の列の場合は、幅ではなく、精度および位取りを定義します。行は、単一のレコードに対応する列情報の集合です。

表の各列にはルールを指定できます。これらのルールは整合性制約と呼ばれています。一例として`NOT NULL`の整合性制約があります。この制約では、列のすべての行に値が含まれている必要があります。

透過的データ暗号化を起動して、データを暗号化してから格納できます。ユーザーが、オペレーティング・システムのツールを使用してOracleデータファイルの内容を直接参照することによって、データベース・アクセス制御メカニズムを迂回しようとした場合でも、暗号化によって、このようなユーザーが機密データを参照できないようにします。

表には仮想列を含めることもできます。仮想列は表の他の列とほぼ同じですが、値が式を評価して導出される点が異なります。式には、同じ表からの列、制約、SQL関数およびユーザー定義PL/SQL関数を含めることができます。仮想列に明示的に書き

込むことはできません。

列の型には、LOB、VARRAYおよびネストした表のように専用セグメントに格納されるものがあります。LOBとVARRAYはLOBセグメントに格納されますが、ネストした表は記憶表に格納されます。これらのセグメントに対してSTORAGE句を指定し、表レベルで指定した記憶域パラメータを上書きできます。

表を作成した後は、SQL文またはOracleのバルク・ロード・ユーティリティを使用してデータ行を挿入します。表データは、SQLを使用して問合せ、削除または更新できます。

関連項目:

- 表の概要は、『Oracle Database概要』を参照してください。
- Oracle Databaseのデータ型の説明は、『Oracle Database SQL言語リファレンス』を参照してください。
- 表の領域を管理するためのガイドラインは、「スキーマ・オブジェクトの領域の管理」を参照してください
- 整合性制約の指定や表の分析など、表の管理に関するその他の詳細は、「スキーマ・オブジェクトの管理」を参照してください
- 透過的データ暗号化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

親トピック: 表の管理

20.2 表を管理するためのガイドライン

ガイドラインに従うことで、表の作成や表データのロード、更新および問合せを行うときに、表の管理が容易になり、パフォーマンスの向上にもつながります。

- 作成前の表の設計
  通常、アプリケーション開発者は、表などのアプリケーションの要素を設計する必要があります。データベース管理者は、アプリケーション表を保持する、基礎になる表領域に対する属性の設定を担当します。
- 作成する表のタイプの指定
  Oracle Databaseでは様々なタイプの表を作成できます。
- 各表の位置の指定
  新しい表を格納する表領域を識別するには、CREATE TABLE文にTABLESPACE句を指定します。パーティション表の場合は、各パーティションを格納する表領域をオプションとして指定できます。
- 表作成のパラレル化
  CREATE TABLE文で副問合せ(AS SELECT)を使用して表を作成する際は、パラレル実行を使用できます。複数のプロセスが同時に動作して表を作成するため、表を作成するときのパフォーマンスが向上します。
- 表作成時のNOLOGGINGの使用
  表を最も効率よく作成するには、CREATE TABLE...AS SELECT文でNOLOGGING句を使用します。NOLOGGING句を指定すると、表の作成中に最小限のREDO情報しか生成されません。
- 表圧縮の使用
  データベースのサイズが大きくなった場合、表圧縮を使用して領域を節約し、パフォーマンスを改善することを検討してください。
- Enterprise Manager Cloud Controlを使用した表圧縮の管理
  Oracle Enterprise Manager Cloud Controlで表圧縮を管理できます。
- セグメント・レベルおよび行レベルの圧縮層の使用

セグメント・レベルの圧縮層を使用すると、表内のセグメント・レベルで圧縮を指定できます。行レベルの圧縮層を使用すると、表内の行レベルで圧縮を指定できます。同じ表でこれらの組合せを使用して、表内のデータが格納および管理される方法を細かく制御できます。

- 属性クラスタ表の使用
  属性クラスタ表とは、ユーザー指定のクラスタリング・ディレクティブに基づいてディスク上に近接近でデータを格納するヒープ構成表です。
- ゾーン・マップの使用
  ゾーンとは、ディスク上で連続している一群のデータ・ブロックです。ゾーン・マップでは、個々のゾーンすべてについて、指定された列の最小値および最大値を追跡管理します。
- インメモリー列ストアへの表の格納
  インメモリー列ストアは、システム・グローバル領域(SGA)のオプション部分で、高速スキャン用に最適化された表、表パーティション、その他のデータベース・オブジェクトのコピーが格納されます。インメモリー列ストアでは、表データがSGAに行ではなく列ごとに格納されます。
- 不可視の列の使用
  表を使用するアプリケーションを中断しないで表を変更する場合に、不可視の列を使用できます。
- 機密データを格納する列の暗号化
  機密データが格納された表の列を個々に暗号化できます。機密データには、社会保障番号、クレジット・カード番号、医療記録などがあります。列の暗号化は、アプリケーションに対して完全に透過的ですが、いくつか制限事項があります。
- セグメント作成の遅延の理解
  ローカルで管理される表領域内にヒープ構成表を作成すると、最初の行が挿入されるまで表セグメントの作成が遅延されます。
- セグメントのマテリアライズ
  `DBMS_SPACE_ADMIN`パッケージには`MATERIALIZE_DEFERRED_SEGMENTS( )`プロシージャが含まれており、このプロシージャを使用すると、セグメント作成の遅延が有効であるときに作成された表、表パーティションおよび依存オブジェクトのセグメントをマテリアライズできます。
- 表サイズの見積りと見積りに応じた計画
  表を作成する前に表のサイズを見積ります。見積りは、なるべくデータベース計画の一部として実行します。データベース表のサイズと用途を確認することは、データベース計画の重要な部分です。
- 表作成時の制限事項
  表を作成するときには制限事項を考慮する必要があります。

親トピック: 表の管理

20.2.1 作成前の表の設計

通常、アプリケーション開発者は、表などのアプリケーションの要素を設計する必要があります。データベース管理者は、アプリケーション表を保持する、基礎になる表領域に対する属性の設定を担当します。

DBAまたはアプリケーション開発者は(あるいは双方が協力して)、サイトの業務に基づいて実際の表の作成を担当します。表を設計する場合は、アプリケーション開発者と協力し、次のガイドラインを考慮してください。

- 表、列、索引およびクラスタには、内容を表現する名前を使用します。
- 表名および列に略語を使用する場合や、単数形と複数形の使用には、一貫性を持たせます。
- `COMMENT`コマンドを使用して、各表とその列の意味を記載します。
- 各表は正規化します。

- 各列に適切なデータ型を選択します。
- 一部の表に仮想列を1つ以上追加した場合、アプリケーションに利点があるかどうかを検討します。
- 記憶域を節約するために、NULLを許可する列を最後に定義します。
- 記憶域を節約し、SQL文のパフォーマンスを最適化するために、適切な場合は必ず表をクラスタ化します。

表を作成する前に、整合性制約の使用についても判断します。表の列に整合性制約を定義することによって、データベースのビジネス・ルールを自動的に徹底できます。

親トピック: 表を管理するためのガイドライン

20.2.2 作成する表のタイプの指定

Oracle Databaseでは様々なタイプの表を作成できます。

作成できる表のタイプは次のとおりです。

表のタイプ	説明
通常の(ヒープ構成)表	この章の主な説明の対象でもある基本的で多目的な表です。この表のデータは、順序付けされていないコレクション(ヒープ)として格納されます。
クラスタ化表	クラスタ化表は、クラスタの一部である表です。クラスタとは、各表が共通の列を共有していて一緒に使用されるケースが多いために、同じデータ・ブロックを共有する表のグループです。 クラスタおよびクラスタ化表の詳細は、「クラスタの管理」を参照してください。
索引構成表	通常の(ヒープ構成)表とは異なり、索引構成表のデータは B ツリーの索引構造に主キー・ソート方式で格納されます。B ツリーの各索引エントリには、索引構成表の行の主キーの列値のみでなく、非キーの列値も格納されます。 索引構成表の詳細は、「索引構成表の管理」を参照してください。
パーティション表	パーティション表では、データをパーティションと呼ばれるさらに小さく管理が容易な単位に分割でき、さらにサブパーティションに分割することもできます。各パーティションには、圧縮の有効化または無効化、圧縮のタイプ、物理記憶域設定、表領域など別個の物理属性を指定できるため、可用性およびパフォーマンスをより適切にチューニングできる構造になります。さらに、各パーティションを個別に管理できるため、バックアップや管理が簡素化され、これらの処理に必要な時間を削減できます。 パーティション表の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。
外部表	外部表は、データベース内にない、データベース外部の外部ファイル(オペレーティング・システム・ファイルや Hadoop Distributed File System (HDFS)ファイルなど)内にあ

表のタイプ	説明
	る表です。 外部表の詳細は、外部表の管理を参照してください。
ハイブリッド・パーティション表	ハイブリッド・パーティション表は、一部のパーティションがデータベース内にあり一部のパーティションがデータベース外部の外部ファイル(オペレーティング・システム・ファイルやHadoop Distributed File System (HDFS)ファイルなど)内にあるパーティション表です。 ハイブリッド・パーティション表の詳細は、Oracle Database VLDB およびパーティショニング・ガイドを参照してください。

親トピック: 表を管理するためのガイドライン

20.2.3 各表の位置の指定

新しい表を格納する表領域を識別するには、CREATE TABLE文にTABLESPACE句を指定します。パーティション表の場合は、各パーティションを格納する表領域をオプションとして指定できます。

使用する表領域に対する適切なシステム権限と割当て権限があることを確認してください。CREATE TABLE文で表領域を指定しない場合は、作成したユーザーのデフォルト表領域内に表が作成されます。

新しい表を含む表領域を指定するときは、その選択が意味することを確実に理解しておいてください。各表の作成時に表領域を適切に指定することによって、データベース・システムのパフォーマンスが向上し、データベース管理に必要な時間を短縮できます。

次のように、表領域を指定しない場合や不適切な表領域を指定した場合は、パフォーマンスに影響を与えます。

- ユーザーのオブジェクトをSYSTEM表領域に作成すると、データ・ディクショナリ・オブジェクトとユーザー・オブジェクトの両方が同じデータファイルを求めて競合し、データベースのパフォーマンスが低下するおそれがあります。ユーザーのオブジェクトはSYSTEM表領域に格納しないでください。これを回避するには、データベースに表領域が作成される際に、すべてのユーザーにデフォルトの表領域が割り当てられていることを確認します。
- アプリケーションに関係する表をいろいろな表領域に無計画に格納すると、アプリケーションのデータ管理操作(バックアップやリカバリなど)に要する時間が増大する可能性があります。

親トピック: 表を管理するためのガイドライン

20.2.4 表作成のパラレル化

CREATE TABLE文で副問合せ(AS SELECT)を使用して表を作成する際は、パラレル実行を使用できます。複数のプロセスが同時に動作して表を作成するため、表を作成するときのパフォーマンスが向上します。

表作成のパラレル化の説明は、「表作成のパラレル化」を参照してください。

親トピック: 表を管理するためのガイドライン

20.2.5 表作成時のNOLOGGINGの使用

表を最も効率よく作成するには、CREATE TABLE...AS SELECT文でNOLOGGING句を使用します。NOLOGGING句を指定すると、表の作成中に最小限のREDO情報しか生成されません。

NOLOGGING句を使用すると、次のような利点があります。

- REDOログ・ファイルの領域を節約できます。
- 表の作成に要する時間が削減できます。
- 大規模な表のパラレル作成のパフォーマンスが向上します。

また、NOLOGGING句を指定することで、SQL*Loaderを使用した後続のダイレクト・ロードおよびダイレクト・ロードINSERT操作がロギングされなくなります。後続のデータ操作文(DML)文(UPDATE、DELETEおよび従来型パスの挿入)は、表のNOLOGGING属性の影響を受けず、REDOを生成します。

表の作成後にその表の損失(たとえば、表の作成に使用したデータにアクセスできなくなるなど)を避ける必要がある場合は、作成直後に表のバックアップを取得してください。一時的に使用するために作成する表など、そのような予防策が不要な場合もあります。

一般に、NOLOGGINGを指定して表を作成するときは、小規模な表より大規模な表の方が相対的にパフォーマンスの向上が大きくなります。小規模な表の場合は、NOLOGGINGを指定しても、表作成に要する時間にほとんど影響はありません。一方、大規模な表では、特に表作成をパラレル化したときにもパフォーマンスが著しく向上します。

親トピック: 表を管理するためのガイドライン

20.2.6 表圧縮の使用

データベースのサイズが大きくなった場合、表圧縮を使用して領域を節約し、パフォーマンスを改善することを検討してください。

- 表圧縮について
  圧縮を使用すると、ディスク領域が節約され、データベース・バッファ・キャッシュのメモリー使用が削減されて、読込み中の問合せ実行速度が大幅に向上します。
- 表圧縮に関連のある例
  例を使用して表圧縮の使用方法を説明します。
- 圧縮とパーティション表
  表には圧縮パーティションと非圧縮パーティションの両方を含めることができ、異なるパーティションでは異なる圧縮方法を使用できます。表に対する圧縮の設定とそのパーティションに対する設定が一致しない場合、パーティションについてはパーティションの設定が優先されます。
- 表が圧縮されているかどうかの確認
  *_TABLESデータ・ディクショナリ・ビューで、圧縮表にはCOMPRESSION列にENABLEDと表示されます。
- 圧縮されている行の確認
  行の圧縮レベルを確認するには、DBMS_COMPRESSIONパッケージのGET_COMPRESSION_TYPEファンクションを使用します。
- 圧縮レベルの変更
  パーティション、表または表領域の圧縮レベルは変更できます。
- 圧縮表の列の追加と削除
  圧縮表に列を追加したり、圧縮表から列を削除する際には制限が適用されます。
- ハイブリッド列圧縮表のエクスポートおよびインポート

ハイブリッド列圧縮表は、データ・ポンプ・インポート・ユーティリティのimpdpコマンドを使用してインポートできます。

- ハイブリッド列圧縮表のリストア
  ハイブリッド列圧縮表をバックアップからリストアする必要がある場合があります。表はハイブリッド列圧縮をサポートしているシステム、またはハイブリッド列圧縮をサポートしていないシステムにリストアできます。
- 圧縮表に関するノートおよび制限事項
  圧縮表に関するノートと制限事項を考慮してください。
- 圧縮表のパック
  基本表圧縮またはハイブリッド列圧縮で圧縮した表で従来のDMLを使用すると、挿入および更新されるすべての行は非圧縮、または低レベルの圧縮形式で保存されます。このような行が圧縮されるように圧縮表をパックするには、ALTER TABLE MOVE文を使用できます。

親トピック: 表を管理するためのガイドライン

20.2.6.1 表圧縮について

圧縮を使用すると、ディスク領域が節約され、データベース・バッファ・キャッシュのメモリー使用が削減されて、読込み中の問合せ実行速度が大幅に向上します。

圧縮には、データのロードやDMLのためのCPUオーバーヘッドがかかります。ただし、この負荷はI/O要件の削減によって相殺されます。圧縮された表データはメモリー内で圧縮されたままになるため、より多くの行をデータベース・バッファ・キャッシュ(および有効になっている場合はフラッシュ・キャッシュ)に収めることができ、圧縮によってDML操作のパフォーマンスを向上させることもできます。

表の圧縮は、アプリケーションに対して完全に透過的です。意思決定支援システム(DSS)、オンライン・トランザクション処理(OLTP)システムおよびアーカイブ・システムで役立ちます。

圧縮は、表領域、表またはパーティションに対して指定できます。表領域レベルで指定した場合、その表領域内に作成されるすべての表がデフォルトで圧縮されます。

Oracle Databaseでは、表の圧縮でいくつかの方法がサポートされます。これらの方法を表20-1にまとめます。

表20-1 表圧縮方法

表の圧縮方法	圧縮レベル	CPUオーバーヘッド	アプリケーション	ノート
基本表圧縮	高	最小	DSS	なし。
高度な行圧縮	高	最小	OLTP、DSS	なし。
ウェアハウス圧縮(ハイブリッド列圧縮)	より高い	より高い	DSS	圧縮レベルと CPU オーバーヘッドは、指定された圧縮レベル(LOW または HIGH)に応じて変化します。
アーカイブ圧縮(ハイブリッド列圧縮)	最高	最高	アーカイブ	圧縮レベルと CPU オーバーヘッドは、指定された圧縮レベル(LOW または HIGH)に応じて変化します。

基本表圧縮、ウェアハウス圧縮またはアーカイブ圧縮を使用する場合、圧縮はデータが表にバルク・ロードまたは配列挿入され

るときにのみ実行されます。

基本表圧縮では、サポートされるデータ型およびSQL操作が制限されます。

高度な行圧縮はOLTPアプリケーション向けで、すべてのSQL操作によって操作されたデータを圧縮します。高度な行圧縮を使用する場合、圧縮は、表に対するデータの挿入、更新またはバルク・ロード中に実行されます。高度な行圧縮が許可される操作は次のとおりです。

- 単一行の挿入と更新

  挿入および更新は即時に圧縮されません。すでに圧縮されているブロックを更新する場合、更新されない列は通常、圧縮されたままです。更新された列は、圧縮されていないブロックと同様に非圧縮形式で格納されます。更新された値は、ブロックがデータベース制御されたしきい値に達すると、再圧縮されます。挿入されたデータも、ブロック内のデータがデータベース制御されたしきい値に達すると、圧縮されます。

- 配列の挿入

  配列挿入には、APPENDヒントを使用しないINSERT INTO SELECTのSQL文、およびPL/SQLやOracle Call Interface (OCI)などのプログラム・インタフェースからの配列の挿入があります。

- 次のダイレクト・パスINSERT方法

  - ダイレクト・パスSQL*Loader
  - CREATE TABLE AS SELECT文
  - パラレルINSERT文
  - APPENDまたはAPPEND_VALUESヒントを指定したINSERT文

  これらのダイレクト・パスINSERTメソッドを使用した挿入は、すぐに圧縮されます。

ウェアハウス圧縮とアーカイブ圧縮では、ハイブリッド列圧縮テクノロジが使用されるため、最高の圧縮レベルが実現します。ハイブリッド列圧縮テクノロジでは、行優先ストレージではなく、修正された形式の列指向ストレージが使用されます。これにより、データベースでは、同様のデータをまとめて格納できるため、圧縮アルゴリズムの効率性が向上します。データを更新する場合、ハイブリッド列圧縮ではより多くのCPUが使用され、将来の更新を迅速に行うために、更新された行は行形式に移動されます。このような最適化が行われるため、更新頻度の低いデータにのみこの圧縮機能を使用してください。

より高い圧縮レベルのハイブリッド列圧縮は、ダイレクト・パス・インサートまたは配列挿入を使用したデータの場合のみ可能です。従来の挿入および更新もサポートされますが、その場合、行が列形式から行形式に移動され、圧縮レベルも低下します。自動データ最適化(ADO)ポリシーを使用して、これらの行をハイブリッド列圧縮の目的のレベルに自動的に戻すことができます。

ハイブリッド列圧縮(ウェアハウスとアーカイブ)で配列挿入をすぐに圧縮するには、次の条件を満たしている必要があります。

- 自動セグメント領域管理(ASSM)が有効なローカル管理表領域に、表が格納されている。
- データベースの互換性レベルが12.2.0以上。

圧縮方法に関係なく、圧縮されたブロックに対するDELETE操作は、圧縮されていないブロックに対するDELETE操作と同じです。SQL DELETE操作によってデータ・ブロックで取得される領域はすべて、後続のSQL INSERT操作で再利用されます。ハイブリッド列圧縮テクノロジを使用すると、圧縮単位内の行がすべて削除された場合、圧縮単位内の領域は再利用に使用できます。

表20-2は、表の各圧縮方法の特徴を示しています。

表20-2 表の圧縮の特徴

表の圧縮方法	CREATE/ALTER TABLEの構文	ダイレクト・パス挿入または配列挿入	ノート
基本表圧縮	`ROW STORE COMPRESS [BASIC]`	行は基本表圧縮方式で圧縮されます。	`ROW STORE COMPRESS` と `ROW STORE COMPRESS BASIC` は同等です。 ダイレクト・パス・インサートまたは配列挿入を使用せずに挿入された行および更新された行は圧縮されません。
高度な行圧縮	`ROW STORE COMPRESS ADVANCED`	行は高度な行圧縮方式で圧縮されます。	ダイレクト・パス・インサートまたは配列挿入の使用に関係なく、挿入された行と更新された行は高度な行圧縮を使用して圧縮されます。
ウェアハウス圧縮(ハイブリッド列圧縮)	`COLUMN STORE COMPRESS FOR QUERY [LOW|HIGH]`	行はウェアハウス圧縮方式で圧縮されます。	この圧縮方式は高い CPU オーバーヘッドが発生する可能性があります。 更新された行およびダイレクト・パス・インサートまたは配列挿入を使用せずに挿入された行は、列形式ではなく行形式で格納されるため、圧縮レベルが低下します。
アーカイブ圧縮(ハイブリッド列圧縮)	`COLUMN STORE COMPRESS FOR ARCHIVE [LOW|HIGH]`	行はアーカイブ圧縮方式で圧縮されます。	この圧縮方式は高い CPU オーバーヘッドが発生する可能性があります。 更新された行およびダイレクト・パス・インサートまたは配列挿入を使用せずに挿入された行は、列形式ではなく行形式で格納されるため、圧縮レベルが低下します。

表圧縮の指定には、CREATE TABLE文のCOMPRESS句を使用します。既存の表で圧縮を有効にするには、ALTER TABLE文でこれらの句を使用します。この場合、圧縮を使用可能にした後で挿入または更新されたデータのみが圧縮されます。ALTER TABLE MOVE文を使用した場合にも、挿入および更新されたデータの圧縮が有効になりますが、この文では既存のデータも圧縮されます。同様に、ALTER TABLE...NOCOMPRESS文を使用すると、既存の圧縮表に対する表圧縮を使用禁止にできます。この場合、すでに圧縮されているすべてのデータは圧縮されたままですが、新規データは圧縮されずに挿入されます。

COLUMN STORE COMPRESS FOR QUERY HIGHオプションは、デフォルトのデータ・ウェアハウス圧縮モードです。Exadataストレージでハイブリッド列圧縮を使用する場合に、高い圧縮レベルと優れたパフォーマンスが実現します。COLUMN STORE COMPRESS FOR QUERY LOWオプションは、ロード・パフォーマンスが非常に重要な環境で使用する必要があります。このオプションでは、COLUMN STORE COMPRESS FOR QUERY HIGHオプションで圧縮されたデータより高速にロードが行われます。

COLUMN STORE COMPRESS FOR ARCHIVE LOWオプションは、デフォルトのアーカイブ圧縮モードです。これにより、高い圧縮レベルが実現し、頻繁にアクセスしないデータに最適です。めったにアクセスされないデータに対しては、COLUMN STORE COMPRESS FOR ARCHIVE HIGHオプションを使用する必要があります。

DBMS_COMPRESSIONパッケージで提供される圧縮アドバイザを使用すると、特定の表に特定の圧縮方法を適用したときに予想される圧縮レベルを確認できます。

ノート:

ハイブリッド列圧縮は、基礎となるストレージ・システムに依存します。詳細は、『Oracle Database ライセンス情報』を参照してください。

関連項目:

- 表圧縮の概要は、『Oracle Database概要』を参照してください。
- デフォルト圧縮属性を持つ表領域について

親トピック: 表圧縮の使用

20.2.6.2 表圧縮に関連のある例

表圧縮の使用例を示します。

例20-1 高度な行圧縮を使用した表の作成

次の例では、表ordersに対して高度な行圧縮を使用可能にします。

```
CREATE TABLE orders  ...  ROW STORE COMPRESS ADVANCED;
```

orders表のデータは、ダイレクト・パスINSERT、配列挿入および従来のDMLで圧縮されます。

例20-2 基本表圧縮を使用した表の作成

次に示す同等の文では、データ・ウェアハウス内のファクト表であるsales_history表に対して基本表圧縮を使用可能にします。

```
CREATE TABLE sales_history  ...  ROW STORE COMPRESS BASIC;
CREATE TABLE sales_history  ...  ROW STORE COMPRESS;
```

この表には問合せが頻繁に実行されますが、DMLの実行は想定されていません。

例20-3 ダイレクト・パス・インサートを使用した表への行の挿入

この例では、ダイレクト・パスINSERTを使用してsales_history表に行を挿入する場合にAPPENDヒントを使用する方法を示しています。

```
INSERT /*+ APPEND */ INTO sales_history SELECT * FROM sales WHERE cust_id=8890;
COMMIT;
```

例20-4 配列挿入を使用した表への行の挿入

この例では、SQLで配列挿入を使用して、sales_history表に行を挿入しています。

```
INSERT INTO sales_history SELECT * FROM sales WHERE cust_id=8890;
COMMIT;
```

この例では、PL/SQLで配列挿入を使用して、hr.jobs_test表に行を挿入しています。

```
DECLARE
   TYPE table_def IS TABLE OF hr.jobs%ROWTYPE;
   array table_def := table_def();
BEGIN
   SELECT * BULK COLLECT INTO array FROM hr.jobs;
   FORALL i in array.first .. array.last
   INSERT INTO hr.jobs_test VALUES array(i);
COMMIT;
END;
/
```

ノート:

ハイブリッド列圧縮(ウェアハウスとアーカイブ)を使用する場合は、SQL、PL/SQL または OCI で配列挿入を実行して即座に圧縮するには、自動セグメント領域管理(ASSM)が有効なローカル管理表領域に表が格納されており、データベースの互換性レベルが 12.2.0 以上である必要があります。

例20-5 ウェアハウス圧縮を使用した表の作成

この例では、表sales_historyでハイブリッド列圧縮を使用可能にします。

```
CREATE TABLE sales_history  ...  COLUMN STORE COMPRESS FOR QUERY;
```

表は、デフォルトのCOLUMN STORE COMPRESS FOR QUERY HIGHオプションを使用して作成されます。このオプションは、基本表圧縮または高度な行圧縮よりも高レベルの圧縮を実現します。問合せが頻繁に表に対して実行され、DMLが予期されない場合に適しています。

例20-6 アーカイブ圧縮を使用した表の作成

次の例では、表sales_historyでハイブリッド列圧縮を使用可能にします。

```
CREATE TABLE sales_history  ...  COLUMN STORE COMPRESS FOR ARCHIVE;
```

表は、デフォルトのCOLUMN STORE COMPRESS FOR ARCHIVE LOWオプションを使用して作成されます。このオプションは、基本圧縮、高度な行圧縮またはウェアハウス圧縮よりも高レベルの圧縮を実現します。ロード・パフォーマンスが非常に重要であり、データへのアクセスが頻繁でない場合に適しています。デフォルトのCOLUMN STORE COMPRESS FOR ARCHIVE LOWオプションでは、COLUMN STORE COMPRESS FOR ARCHIVE HIGHオプションより低レベルの圧縮を実現します。

親トピック: 表圧縮の使用

20.2.6.3 圧縮とパーティション表

表には圧縮パーティションと非圧縮パーティションの両方を含めることができ、異なるパーティションでは異なる圧縮方法を使用できます。表に対する圧縮の設定とそのパーティションに対する設定が一致しない場合、パーティションについてはパーティションの設定が優先されます。

パーティションの圧縮方法を変更するには、次のどちらかを実行します。

- 新しいデータのみの圧縮方法を変更するには、ALTER TABLE ... MODIFY PARTITION ... COMPRESS ...を使用します。
- 新しいデータと既存のデータの両方の圧縮方法を変更するには、ALTER TABLE ... MOVE PARTITION ... COMPRESS ...または表のオンライン再定義のどちらかを使用します。

これらの文を実行する場合は、圧縮方法を指定します。たとえば、次の文を実行して、新規データと既存データの両方の圧縮方式を高度な行圧縮に変更します。

```
ALTER TABLE ... MOVE PARTITION ... ROW STORE COMPRESS ADVANCED...
```

親トピック: 表圧縮の使用

20.2.6.4 表が圧縮されているかどうかの確認

*_TABLESデータ・ディクショナリ・ビューで、圧縮表にはCOMPRESSION列にENABLEDと表示されます。

パーティション表では、この列はNULLですが、*_TAB_PARTITIONSビューのCOMPRESSION列に、圧縮されているパーティションが表示されます。また、COMPRESS_FOR列には、表またはパーティションで使用中の圧縮方法が表示されます。

```
SQL> SELECT table_name, compression, compress_for FROM user_tables;

TABLE_NAME       COMPRESSION   COMPRESS_FOR
---------------- ------------  -----------------
T1               DISABLED
T2               ENABLED       BASIC
T3               ENABLED       ADVANCED
T4               ENABLED       QUERY HIGH
T5               ENABLED       ARCHIVE LOW

SQL> SELECT table_name, partition_name, compression, compress_for
  FROM user_tab_partitions;
TABLE_NAME  PARTITION_NAME   COMPRESSION   COMPRESS_FOR
----------- ---------------- -----------   ------------------------------
SALES       Q4_2004          ENABLED       ARCHIVE HIGH
  ...
SALES       Q3_2008          ENABLED       QUERY HIGH
SALES       Q4_2008          ENABLED       QUERY HIGH
SALES       Q1_2009          ENABLED       ADVANCED
SALES       Q2_2009          ENABLED       ADVANCED
```

親トピック: 表圧縮の使用

20.2.6.5 圧縮されている行の確認

行の圧縮レベルを確認するには、DBMS_COMPRESSIONパッケージのGET_COMPRESSION_TYPEファンクションを使用します。

たとえば、次の問合せは、hr.employees表の行の圧縮タイプを返します。

```
SELECT DECODE(DBMS_COMPRESSION.GET_COMPRESSION_TYPE(
                 ownname    => 'HR',
                 tabname    => 'EMPLOYEES',
                 subobjname => '',
                 row_id     => 'AAAVEIAAGAAAABTAAD'),
   1,  'No Compression',
   2,  'Advanced Row Compression',
   4,  'Hybrid Columnar Compression for Query High',
   8,  'Hybrid Columnar Compression for Query Low',
   16, 'Hybrid Columnar Compression for Archive High',
   32, 'Hybrid Columnar Compression for Archive Low',
   4096, 'Basic Table Compression',
   'Unknown Compression Type') compression_type
FROM DUAL;
```

関連項目:

GET_COMPRESSION_TYPEの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 表圧縮の使用

20.2.6.6 圧縮レベルの変更

パーティション、表または表領域の圧縮レベルは変更できます。

たとえば、ある企業でその売上データにウェアハウス圧縮を使用している一方で、6か月より古い売上データにはめったにアクセスしないとします。売上データがその経過時間に基づいてパーティション化された表に格納されている場合、古いデータの圧縮レベルをアーカイブ圧縮に変更して、ディスク領域を解放できます。

パーティションまたはサブパーティションの圧縮レベルを変更するには、次の文を使用できます。

- ALTER TABLE ... MOVE PARTITION ... ONLINE
- ALTER TABLE ... MOVE SUBPARTITION ... ONLINE

この2つの文では、ONLINEキーワードがサポートされており、移動中のパーティションまたはサブパーティションに対してDML操作を中断なく実行できます。これらの文では、パーティションまたはサブパーティションの移動中に更新されたすべての索引も自動的に保持します。ALTER TABLE...MODIFY PARTITION文またはオンライン再定義を使用して、パーティションの圧縮レベルを変更することもできます。

表がパーティション化されていない場合、ALTER TABLE...MOVE...COMPRESS FOR...文を使用して圧縮レベルを変更できます。ALTER TABLE...MOVE文では、コマンドの実行中に表に対するDML文は許可されません。ただし、オンライン再定義を使用して表を圧縮することもでき、これにより、再定義中でも問合せおよびDML文で表を使用できるようになります。

表領域の圧縮レベルを変更するには、ALTER TABLESPACE文を使用します。

関連項目:

- ALTER TABLEコマンドの詳細は、「新規セグメントまたは表領域への表の移動」を参照してください
- 「表のオンライン再定義」
- DBMS_REDEFINITIONパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 表圧縮の使用

20.2.6.7 圧縮表の列の追加と削除

圧縮表に列を追加したり、圧縮表から列を削除する際には制限が適用されます。

圧縮表に列を追加する場合、次の制限が適用されます。

- 高度な行圧縮、ウェアハウス圧縮およびアーカイブ圧縮: 追加された列にデフォルト値が指定され、表がすでに移入されている場合、最適化された追加列動作の条件を満たしている必要があります。これらの条件については、『Oracle Database SQL言語リファレンス』を参照してください。

圧縮表の列を削除する場合、次の制限が適用されます。

- 基本表圧縮: 列の削除はサポートされていません。
- 高度な行圧縮、ウェアハウス圧縮およびアーカイブ圧縮: DROP COLUMNはサポートされていますが、長時間実行され

る解凍と再圧縮の操作を避けるために、データベースでは列が内部的にUNUSEDに設定されます。

親トピック: 表圧縮の使用

20.2.6.8 ハイブリッド列圧縮表のエクスポートおよびインポート

ハイブリッド列圧縮表は、データ・ポンプのインポート・ユーティリティのimpdpコマンドを使用してインポートできます。

デフォルトでは、impdpコマンドは表プロパティを保存し、インポートされた表はハイブリッド列圧縮表となります。ハイブリッド列圧縮をサポートしていない表領域では、impdpコマンドは失敗し、エラーが表示されます。表はexpdpコマンドでエクスポートすることもできます。

ハイブリッド列圧縮表は、impdpコマンドのTRANSFORM:SEGMENT_ATTRIBUTES=nオプション句を使用して、非圧縮表としてインポートできます。

非圧縮表または高度な行圧縮表は、インポート中にハイブリッド列圧縮形式に変換できます。ハイブリッド列圧縮表でない表をハイブリッド列圧縮表に変換するには、次のようにします。

1. ALTER TABLESPACE ... SET DEFAULT COMPRESSコマンドを使用して、表領域のデフォルト圧縮を指定します。
2. インポート中にインポートされた表のSEGMENT_ATTRIBUTESオプションを上書きします。

関連項目:

- データ・ポンプのインポート・ユーティリティの詳細は、『Oracle Databaseユーティリティ』を参照してください。
- ALTER TABLESPACEコマンドの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表圧縮の使用

20.2.6.9 ハイブリッド列圧縮表のリストア

ハイブリッド列圧縮表をバックアップからリストアする必要がある場合があります。表はハイブリッド列圧縮をサポートしているシステム、またはハイブリッド列圧縮をサポートしていないシステムにリストアできます。

ハイブリッド列圧縮が含まれる表をハイブリッド列圧縮をサポートしているシステムにリストアする場合は、通常どおり、Oracle Recovery Manager (RMAN)を使用してファイルをリストアします。

ハイブリッド列圧縮表がハイブリッド列圧縮をサポートしていないシステムにリストアされている場合は、表をハイブリッド列圧縮から高度な行圧縮形式または非圧縮形式に変換する必要があります。表をリストアするには、次のようにします。

1. 環境に非圧縮形式または高度な行圧縮形式のデータを保存するのに十分な記憶域があることを確認します。
2. RMANを使用して、ハイブリッド列圧縮表領域をリストアします。
3. 次のいずれかのアクションを実行して、表をハイブリッド列圧縮から高度な行圧縮形式または非圧縮形式に変換します。
   - 次の文を使用して、データ圧縮をハイブリッド列圧縮からROW STORE COMPRESS ADVANCEDに変更します。

     ```
     ALTER TABLE table_name MOVE ROW STORE COMPRESS ADVANCED;
     ```

   - 次の文を使用して、データ圧縮をハイブリッド列圧縮からNOCOMPRESSに変更します。

     ```
     ALTER TABLE table_name MOVE NOCOMPRESS;
     ```

   - 各パーティションをNOCOMPRESSに変更するには、次の文を使用します。

```
ALTER TABLE table_name MOVE PARTITION partition_name NOCOMPRESS;
```

各パーティションは個別に変更します。

移動中のパーティションに対するDMLが必要な場合は、ONLINEキーワードを含めます。

```
ALTER TABLE table_name MOVE PARTITION partition_name NOCOMPRESS ONLINE;
```

パーティションをオンラインで移動するのには、オフラインで移動するよりも時間がかかる場合があります。

- 次の文を使用して、データをNOCOMPRESSに並列で移動します。

```
ALTER TABLE table_name MOVE NOCOMPRESS PARALLEL;
```

関連項目:

- RMANの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- ALTER TABLEコマンドの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表圧縮の使用

20.2.6.10 圧縮表に関するノートおよび制限事項

圧縮表に関連するノートと制限事項を考慮してください。

圧縮表に関して、次のノートや制限があります。

- 次のタイプの表では、高度な行圧縮、ウェアハウス圧縮およびアーカイブ圧縮はサポートされません。
  - 索引構成表
  - 外部表
  - LONG列またはLONG RAW列を含む表
  - 一時表
  - ROWDEPENDENCIESが有効な表
  - クラスタ表
- 次の圧縮メソッドで圧縮された表では、オンラインのセグメント縮小はサポートされません。
  - ROW STORE COMPRESS BASICを使用した基本表圧縮
  - COLUMN STORE COMPRESS FOR QUERYを使用したウェアハウス圧縮
  - COLUMN STORE COMPRESS FOR ARCHIVEを使用したアーカイブ圧縮
- この項で説明する表圧縮方法は、SecureFilesラージ・オブジェクト(LOB)には適用されません。SecureFiles LOBには独自の圧縮方法があります。詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- 圧縮テクノロジではCPUを使用します。追加の負荷を処理するために使用可能なCPUが十分にあることを確認する必要があります。
- 基本表圧縮で作成された表では、特に指定しないかぎり、PCT_FREEパラメータが自動的に0 (ゼロ)に設定されます。

親トピック: 表圧縮の使用

20.2.6.11 圧縮表のパック

基本表圧縮またはハイブリッド列圧縮で圧縮した表で従来のDMLを使用すると、挿入および更新されるすべての行は非圧縮、または低レベルの圧縮形式で保存されます。このような行が圧縮されるように圧縮表をパックするには、ALTER TABLE MOVE文を使用できます。

この操作には表の排他ロックが必要なため、この操作が完了するまで更新とロードを実行しないでください。このような状況が望ましくない場合は、表のオンライン再定義を使用できます。

パーティションまたはサブパーティションを移動する場合、移動中のパーティションまたはサブパーティションに対してDML操作の実行を中断して、ALTER TABLE MOVE文を使用してパーティションまたはサブパーティションを圧縮できます。

関連項目:

- 制限事項を含むALTER TABLE...COMPRESS文およびALTER TABLE...MOVE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 表のパーティション化の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。
- 「表のオンライン再定義」
- 表、パーティションまたはサブパーティションの移動の詳細は、「新規セグメントおよび表領域への表の移動」を参照してください

親トピック: 表圧縮の使用

20.2.7 Enterprise Manager Cloud Controlを使用した表圧縮の管理

Oracle Enterprise Manager Cloud Controlで、表の圧縮を管理できます。

- 表圧縮とEnterprise Manager Cloud Control
  Enterprise Managerには、データベースおよび表領域レベルの圧縮の機能を要約する複数の集中圧縮ページが表示され、様々な圧縮ページへのリンクが含まれます。「圧縮」ページには、データベースおよび表領域レベルで圧縮された記憶域のサマリーが表示されます。
- データベース・レベルの圧縮サマリーの表示
  データベース・レベルで圧縮サマリー情報を表示できます。
- 表領域レベルの圧縮サマリーの表示
  表領域レベルで圧縮サマリー情報を表示できます。
- 圧縮率の見積り
  特定のオブジェクトの圧縮率を計算するために、圧縮アドバイザを実行できます。
- オブジェクトの圧縮
  表などのオブジェクトを圧縮できます。
- 圧縮アドバイスの表示
  セグメント・アドバイザから圧縮アドバイスを表示し、それらに基づいてアクションを実行できます。
- オブジェクトでの自動データ最適化の開始
  オブジェクトに対する自動データ最適化を開始できます。

親トピック: 表を管理するためのガイドライン

20.2.7.1 表圧縮とEnterprise Manager Cloud Control

Enterprise Managerには、データベースおよび表領域レベルの圧縮の機能を要約する複数の集中圧縮ページが表示され、様々な圧縮ページへのリンクが含まれます。「圧縮」ページには、データベースおよび表領域レベルで圧縮された記憶域のサマリーが表示されます。

データベースの圧縮サマリー・ページには、データベース・レベルに応じて、データベースの合計サイズ(圧縮と未圧縮の両方のオブジェクトすべての合計サイズ)、データベースの圧縮オブジェクトの合計サイズ、データベースの未圧縮オブジェクトの合計サイズ、および圧縮オブジェクトの合計サイズとデータベースの合計サイズの比率が表示されます。これには、データベース内で圧縮されている記憶域の量に関する概略が提供されます。表示される情報に基づいてアクションを実行できます。

同様に、表領域の圧縮サマリー・ページには、表領域レベルに応じて、表領域の合計サイズ(圧縮と未圧縮の両方のオブジェクトすべての合計サイズ)、表領域の圧縮オブジェクトの合計サイズ、表領域の未圧縮オブジェクトの合計サイズ、および圧縮オブジェクトの合計サイズと表領域の合計サイズの比率が表示されます。

圧縮機能を使用すると、次のタスクを実行できます。

- データベースレベルの上位100の表領域および表領域レベルの上位100のオブジェクトで、圧縮された記憶域のサマリーを表示します。表領域の合計サイズ、表領域の圧縮サイズ、表領域の未圧縮サイズ、および表領域内で圧縮された記憶域の割合を含む、最も大きいデータベース記憶域を使用する上位100の各表領域内で圧縮された記憶域の量のサマリーを表示できます。表示された情報に基づいて、圧縮タスクを実行できます。
- 表、索引、LOB (ラージ・オブジェクト)およびDBFS (Oracle Database File System)の4つのオブジェクト・タイプの各圧縮タイプ別に、圧縮された記憶域のサイズを表示します。
- 特定のオブジェクトの圧縮率を計算します。
- オブジェクト(表領域、表、パーティションまたはLOB)を圧縮します。これにより、記憶域を節約できます。圧縮アドバイザを実行して、保存可能な記憶域の量を確認し、オブジェクトで圧縮アクションを実行できます。
- セグメント・アドバイザから圧縮アドバイスを表示します。セグメント・アドバイザへのリンクにアクセスして、セグメントを圧縮できます。

親トピック: Enterprise Manager Cloud Controlを使用した表圧縮の管理

20.2.7.2 データベース・レベルの圧縮サマリーの表示

データベース・レベルで圧縮サマリー情報を表示できます。

1. 「管理」メニューから、「記憶域」を選択し、「圧縮」を選択します。

   Enterprise Managerに、「上位100の表領域の圧縮サマリー」ページが表示されます。

2. 「領域使用量」セクション内のデータベースの合計サイズ、データベースの圧縮オブジェクトの合計サイズ、圧縮オブジェクトの合計サイズとデータベースの合計サイズの比率、および未圧縮オブジェクトのサイズを含む、データベース・レベルの記憶域圧縮のサマリー情報を表示できます。セグメント数の同様の情報は、「セグメント数」」セクションにも表示されます。
3. 表、索引、LOB (ラージ・オブジェクト)およびDBFS (Oracle Database File System)の4つのオブジェクト・タイプの各圧縮タイプ別に、使用される記憶域のサイズを表示できます。チャートの各色をクリックすると、セグメントの圧縮サマリー・ページが表示され、特定のオブジェクト・タイプおよび圧縮タイプの上位100のセグメントの圧縮情報がデータベースのサイズ別に表示されます。

親トピック: Enterprise Manager Cloud Controlを使用した表圧縮の管理

20.2.7.3 表領域レベルの圧縮サマリーの表示

表領域レベルで圧縮サマリー情報を表示できます。

1. 「管理」メニューから、「記憶域」を選択し、「圧縮」を選択します。

   Enterprise Managerに、「上位100の表領域の圧縮サマリー」ページが表示されます。

2. 「サイズ別の上位100の永続表領域」表で、圧縮サマリーを表示する表領域の行をクリックします。
3. 「圧縮詳細の表示」をクリックします。

   Enterprise Managerに、「表領域内の上位100のオブジェクトの圧縮サマリー」ページが表示されます。このページから、表領域の合計サイズ、表領域の圧縮オブジェクトの合計サイズ、圧縮オブジェクトの合計サイズと表領域の合計サイズの比率、および表領域の未圧縮オブジェクトのサイズを表示できます。

   表、索引、LOBおよびDBFSの4つのオブジェクト・タイプの各圧縮タイプ別に、圧縮された表領域の記憶域のサイズを表示することもできます。チャートの各色をクリックすると、セグメントの圧縮サマリー・ダイアログ・ボックスが表示され、特定のオブジェクト・タイプおよび圧縮タイプの上位100のセグメントの圧縮情報が表領域のサイズ別に表示されます。

   最後に、表領域の記憶域を最も多く使用する上位100の各セグメントの圧縮サマリーを表示できます。

親トピック: Enterprise Manager Cloud Controlを使用した表圧縮の管理

20.2.7.4 圧縮率の見積り

特定のオブジェクトの圧縮率を計算するために、圧縮アドバイザを実行できます。

1. 「管理」メニューから、「記憶域」を選択し、「圧縮」を選択します。

   Enterprise Managerに、「上位100の表領域の圧縮サマリー」ページが表示されます。

2. 「サイズ別の上位100の永続表領域」表から、表領域を選択して、「圧縮詳細の表示」をクリックし、選択した表領域の圧縮の詳細を表示します。

   Enterprise Managerに「サイズ別の上位100のオブジェクト」表が表示されます。

3. オブジェクトを選択し、オブジェクトの「予測される圧縮率」をクリックします。

   Enterprise Managerに「予測される圧縮率」ダイアログ・ボックスが表示されます。次の情報を入力します。

   - 「入力パラメータ」セクションで、一時スクラッチ表領域を入力または選択します。名前を直接入力するか、アイコンをクリックしたときに表示されるリストから選択できます。
   - 圧縮タイプを入力します。「基本」、「拡張」、「問合せ低」、「問合せ高」、「アーカイブ低」、「アーカイブ高」から選択できます。HCC圧縮タイプ(「問合せ低」、「問合せ高」、「アーカイブ低」または「アーカイブ高」)の場合、表に100万行以上あることを確認してください。
   - 「ジョブのスケジュール」セクションで、ジョブの名前と説明を入力します。
   - 「スケジュール」セクションで、いつ開始するか、ジョブを繰り返すかどうか、猶予期間を設けるかどうか、期間の情報など、ジョブ情報を入力します。
   - データベース資格証明とホスト資格証明をそれぞれのセクションに入力します。
   - 「OK」をクリックします。

ジョブが即時実行またはスケジュールされ、「表領域内の上位100のオブジェクトの圧縮サマリー」ページに戻ります。

親トピック: Enterprise Manager Cloud Controlを使用した表圧縮の管理

20.2.7.5 オブジェクトの圧縮

表などのオブジェクトを圧縮できます。

1. 「管理」メニューから、「記憶域」を選択し、「圧縮」を選択します。

   Enterprise Managerに、「上位100の表領域の圧縮サマリー」ページが表示されます。

2. 「サイズ別の上位100の永続表領域」表から、表領域を選択して、「圧縮詳細の表示」をクリックし、選択した表領域の圧縮の詳細を表示します。

   Enterprise Managerに、「表領域内の上位100のオブジェクトの圧縮サマリー」ページが表示されます。

3. 表などのオブジェクトを選択し、「圧縮」をクリックしてオブジェクトを圧縮します。

親トピック: Enterprise Manager Cloud Controlを使用した表圧縮の管理

20.2.7.6 圧縮アドバイスの表示

セグメント・アドバイザから圧縮アドバイスを表示し、それらに基づいてアクションを実行できます。

1. 「管理」メニューから、「記憶域」を選択し、「圧縮」を選択します。

   Enterprise Managerに、「上位100の表領域の圧縮サマリー」ページが表示されます。

2. 「圧縮アドバイス」セクションで、「圧縮アドバイスのあるセグメント」フィールドに表示される番号をクリックします。

   Enterprise Managerに「セグメント・アドバイザ推奨」ページが表示されます。メンテナンス・ウィンドウ内で、自動セグメント・アドバイザ・ジョブを使用してセグメントの問題を検出できます。推奨事項は、直近に実行された自動セグメント・アドバイザ・ジョブおよびユーザーがスケジュールしたセグメント・アドバイザ・ジョブから導出されます。

親トピック: Enterprise Manager Cloud Controlを使用した表圧縮の管理

20.2.7.7 オブジェクトでの自動データ最適化の開始

オブジェクトに対する自動データ最適化を開始できます。

1. 「管理」メニューから、「記憶域」を選択し、「圧縮」を選択します。

   Enterprise Managerに、「上位100の表領域の圧縮サマリー」ページが表示されます。

2. 「サイズ別の上位100の永続表領域」表から、表領域を選択して、「圧縮詳細の表示」をクリックし、選択した表領域の圧縮の詳細を表示します。

   Enterprise Managerに、「表領域内の上位100のオブジェクトの圧縮サマリー」ページが表示されます。

3. 「サイズ別の上位100のオブジェクト」から、オブジェクトを選択し、自動データ圧縮をクリックします。

   Enterprise Managerに、オブジェクトの「編集」ページが表示され、そのオブジェクトで自動データの最適化を開始できます。

親トピック: Enterprise Manager Cloud Controlを使用した表圧縮の管理

20.2.8 セグメント・レベルおよび行レベルの圧縮層の使用

セグメント・レベルの圧縮層を使用すると、表内のセグメント・レベルで圧縮を指定できます。行レベルの圧縮層を使用すると、表内の行レベルで圧縮を指定できます。同じ表でこれらの組合せを使用して、表内のデータが格納および管理される方法を細かく制御できます。

セグメントおよび行に対するユーザー変更は時間の経過に伴って変化するため、それらのユーザー変更の圧縮レベルを変更することは、多くの場合、効果的です。たとえば、一部のセグメントおよび行がデータベースに追加された後、短期間は頻繁に変更されるが、時間の経過に伴って変更の頻度が低下することがあります。

圧縮層を使用すると、ルールに基づいて、どのセグメントおよび行を圧縮するかを指定できます。たとえば、2週間変更されていない行を高度な行圧縮方式で圧縮することを指定できます。また、6か月間変更されていないセグメントをウェアハウス圧縮方式で圧縮することを指定できます。

セグメント・レベルおよび行レベルの圧縮層を使用する前に、次の前提条件を満たしている必要があります。

- HEAT_MAP初期化パラメータをONに設定する必要があります。
- COMPATIBLE初期化パラメータは12.0.0以上に設定されている必要があります。

セグメント・レベルの圧縮層または行レベルの圧縮層を使用するには、次のいずれかのSQL文を実行し、ルールを指定する自動データ最適化(ADO)ポリシーを含めます。

- CREATE TABLE
- ALTER TABLE

例20-7 行レベルの圧縮層

この例では、oe.orders表に対して行レベルの圧縮層を指定します。変更なしで14日が経過した後、Oracle Databaseによって、ウェアハウス(QUERY)圧縮を使用して行が圧縮されます。

```
ALTER TABLE oe.orders ILM ADD POLICY
  COLUMN STORE COMPRESS FOR QUERY
  ROW
  AFTER 14 DAYS OF NO MODIFICATION;
```

例20-8 セグメント・レベルの圧縮層

この例では、oe.order_items表に対してセグメント・レベルの圧縮層を指定します。Oracle Databaseでは、セグメント内の行の変更またはセグメント内の行にアクセスする問合せが行われずに6か月間経過すると、アーカイブ(ARCHIVE HIGH)圧縮を使用してセグメントが圧縮されます。

```
ALTER TABLE oe.order_items ILM ADD POLICY
  COLUMN STORE COMPRESS FOR ARCHIVE HIGH
  SEGMENT
  AFTER 6 MONTHS OF NO ACCESS;
```

ノート:

これらの例では、基礎となるストレージ・システムに依存するハイブリッド列圧縮が指定されています。詳細は、『Oracle Database ライセンス情報』を参照してください。

関連項目:

- 様々な圧縮レベルの詳細は、「表圧縮の使用」を参照してください
- 「Oracle Database In-Memoryによる問合せパフォーマンスの向上」
- セグメント・レベルの圧縮層と行レベルの圧縮層の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

20.2.9 属性クラスタ表の使用

属性クラスタ化表は、ユーザー指定のクラスタリング・ディレクティブに基づいて近接度の近いデータをディスク上に格納する、ヒープ構成表です。

ノート:

この機能は、Oracle Database 12c リリース 1 (12.1.0.2)以降で使用可能です。

ディレクティブは次のようになります。

- `CLUSTERING ... BY LINEAR ORDER`ディレクティブは、指定の列に従って表のデータを並べます。

  問合せでクラスタリング句に指定された列の接頭辞を修飾とする場合、`BY LINEAR ORDER`クラスタリング(デフォルト)が最適です。たとえば、`sh.sales`の問合せで、顧客IDまたは顧客IDと製品IDの両方を指定する場合、線形の列順序`cust_id`、`prod_id`を使用して、表のデータをクラスタ化できます。指定の列は、複数の表に存在してもかまいません。

- `CLUSTERING ... BY INTERLEAVED ORDER`ディレクティブは、複数列I/Oの削減を可能にするz-order関数のような特殊なアルゴリズムを使用して1つ以上の表のデータを並べます。

  問合せで様々な列の組合せを指定する場合、`BY INTERLEAVED ORDER`クラスタリングが最適です。列は、1つ以上の表に存在してもかまいません。たとえば、`sh.sales`の問合せで様々な順序で異なるディメンションを指定する場合、そのディメンションの列に従って`sales`表のデータをクラスタ化できます。

属性クラスタリングは次のタイプの操作で使用できます。

- ダイレクト・パス`INSERT`

  「ダイレクト・パス・インサートを使用したINSERTパフォーマンスの向上」を参照してください。

- オンライン再定義

  「表のオンライン再定義」を参照してください。

- `ALTER TABLE ... MOVE`操作などのデータ移動操作

  「新規セグメントまたは表領域への表の移動」を参照してください。

- `ALTER TABLE ... MERGE PARTITION`操作などの、新しいセグメントを作成するパーティション・メンテナンス操作

  詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

属性クラスタリングは、従来のDMLでは無視されます。

属性クラスタ表には次の利点があります。

- 属性クラスタリングが一般的な索引アクセスと連携している場合、表参照では、より最適化された単一ブロックI/Oが可能です。たとえば、属性クラスタリング用に選択した先頭列での索引範囲スキャンでは、最適化されたI/Oが可能です。

- データ順序付けにより、Exadata記憶域索引に対するより最適なプルーニングおよびインメモリー最小/最大プルーニン

グが可能になります。

- 他の表から結合された属性に基づいて、ファクト表をクラスタ化できます。
- 属性クラスタリングによってデータ圧縮が改善でき、この方法で表スキャンのコストが間接的に改善されます。ディスク上で同じ値が互いに接近している場合、データベースでは、これらの値を容易に圧縮できます。

データ・ウェアハウス環境では属性クラスタ表がよく使用されますが、このような利点を享受できるどの環境でも、この表は有用です。CREATE TABLE SQL文でCLUSTERING句を使用して属性クラスタ表を作成します。

関連項目:

- 属性クラスタ表の概念は、『Oracle Database概要』を参照してください。
- 属性クラスタ化表の使用の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください
- Oracle Database SQL言語リファレンス

親トピック: 表を管理するためのガイドライン

20.2.10 ゾーン・マップの使用

ゾーンとは、ディスク上の連続したデータ・ブロックのセットです。ゾーン・マップでは、個々のゾーンすべてについて、指定された列の最小値および最大値を追跡管理します。

ノート:

この機能は、Oracle Database 12c リリース 1 (12.1.0.2)以降で使用可能です。

ゾーン・マップに格納されている列に関する述語がSQL文に含まれている場合、データベースでは、述語の値をゾーンに格納されている最小値および最大値と比較して、SQL実行時にどのゾーンを読み取るかを判断します。ゾーン・マップの主要な利点は、表スキャンのI/O削減です。I/Oは、問合せ結果で不要な表ブロックをスキップすると減ります。CREATE MATERIALIZED ZONEMAP SQL文を使用してゾーン・マップを作成します。

表で属性クラスタリングが指定されている場合は常に、クラスタ列に関するゾーン・マップを自動的に作成できます。クラスタリングのために、列の最小値および最大値は属性クラスタ表の連続するデータ・ブロックと相関するため、関連付けられたゾーン・マップを使用してより効果的なI/Oプルーニングが可能になります。

ノート:

ゾーン・マップおよび属性クラスタ表は、一緒に使用することも、別々に使用することもできます。

関連項目:

- 「属性クラスタ表の使用」
- ゾーン・マップの概念は、『Oracle Database概要』を参照してください。
- ゾーン・マップの使用の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

- CREATE MATERIALIZED ZONEMAP文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表を管理するためのガイドライン

20.2.11 インメモリー列ストアへの表の格納

インメモリー列ストアは、システム・グローバル領域(SGA)のオプション部分で、高速スキャン用に最適化された表、表パーティション、その他のデータベース・オブジェクトのコピーが格納されます。インメモリー列ストアでは、表データがSGAに行ではなく列ごとに格納されます。

ノート:

この機能は、Oracle Database 12c リリース 1 (12.1.0.2)以降で使用可能です。

関連項目:

- 「Oracle Database In-Memoryによる問合せパフォーマンスの向上」
- 『Oracle Database概要』

親トピック: 表を管理するためのガイドライン

20.2.12 不可視の列の使用

表を使用するアプリケーションを中断することなく、表に変更を加える場合、不可視の列を使用できます。

- 不可視の列の理解
  表の列を個々に不可視にできます。表の一般的なアクセスでは、表の不可視の列は表示されません。
- 不可視の列と列の順序
  不可視の列と列の順序については、特別な考慮事項があります。

親トピック: 表を管理するためのガイドライン

20.2.12.1 不可視の列の理解

表の個々の列を不可視にできます。表の一般的なアクセスでは、表の不可視の列は表示されません。

たとえば、次の操作では、不可視の列は出力に表示されません。

- SQLのSELECT * FROM文
- SQL*PlusのDESCRIBEコマンド
- PL/SQLの%ROWTYPE属性宣言
- Oracle Call Interface (OCI)の説明

列リストで不可視の列を明示的に指定した場合にのみ、SELECT文を使用して不可視の列の出力を表示できます。同様に、INSERT文の列リストで不可視の列を明示的に指定した場合にのみ、不可視の列に値を挿入できます。INSERT文の列リストを省略した場合、その文では可視の列にのみ値を挿入できます。

表の作成中または表に列を追加するときに列を不可視にでき、後で、表を変更して、不可視にした列を可視にできます。また、

表を変更して、可視の列を不可視にすることもできます。

表を使用するアプリケーションを中断しないで表を変更する場合に、不可視の列を使用することがあります。表に不可視の列を追加した後、不可視の列にアクセスする必要がある問合せおよびその他の操作では、名前で明示的にその列を参照する必要があります。不可視の列が考慮されるアプリケーションを移行するとき、不可視の列を可視にできます。

仮想列は不可視にできます。また、表を作成するときに、不可視の列をパーティション化キーとして使用することもできます。

不可視の列には、次の制限が適用されます。

- 次のタイプの表に不可視の列を含めることはできません。
  - 外部表
  - クラスタ表
  - 一時表
- ユーザー定義型の属性は不可視にできません。

ノート:

不可視の列は、システム生成の非表示列とは同じではありません。不可視の列を可視にすることはできますが、非表示列を可視にすることはできません。

関連項目:

- 「表の作成」
- 「表の列の追加」
- 「既存の列定義の変更」

親トピック: 不可視の列の使用

20.2.12.2 不可視の列と列の順序

不可視の列と列の順序については、特別な考慮事項があります。

データベースでは、通常、CREATE TABLE文でリストされた順序で列が格納されます。表に新しい列を追加すると、その新しい列は表の列順の最後の列になります。

表に1つ以上の不可視の列が含まれている場合、不可視の列は表の列順には含まれません。表のすべての列にアクセスする場合は、列の順序が重要となります。たとえば、SELECT * FROM文では、表の列順で列が表示されます。不可視の列は表のこのタイプの汎用アクセスには含まれないため、列順に含まれません。

不可視の列を可視にすると、その列は表の列順に最後の列として含められます。可視の列を不可視にした場合、不可視の列は列順に含められないため、表の可視の列の順序が並べ替えられることがあります。

たとえば、不可視の列のある次の表を考えてみます。

```
CREATE TABLE mytable (a INT, b INT INVISIBLE, c INT);
```

列bは不可視であるため、この表の列順は次のようになります。

列	列の順序
a	1
c	2

次に、列bを可視にします。

```
ALTER TABLE mytable MODIFY (b VISIBLE);
```

列bを可視にすると、列bは表の列順の最後の列になります。したがって、表の列順は次のようになります。

列	列の順序
a	1
c	2
b	3

不可視の列を含む表の列順を示す別の例を考えてみます。次の表には、不可視の列は含まれていません。

```
CREATE TABLE mytable2 (x INT, y INT, z INT);
```

この表の列順は次のようになります。

列	列の順序
x	1
y	2
z	3

次に、列yを不可視にします。

```
ALTER TABLE mytable2 MODIFY (y INVISIBLE);
```

列yを不可視にすると、列yは表の列順に含まれなくなるため、列zの列順が変更されます。したがって、表の列順は次のようになります。

列	列の順序
x	1
z	2

列yを再び可視にします。

```
ALTER TABLE mytable2 MODIFY (y VISIBLE);
```

列yは表の列順の最後になります。

列	列の順序
x	1
z	2
y	3

親トピック: 不可視の列の使用

20.2.13 機密データを格納する列の暗号化

機密データを格納する個々の表の列を暗号化できます。機密データには、社会保障番号、クレジット・カード番号、医療記録などがあります。列の暗号化は、アプリケーションに対して完全に透過的ですが、いくつか制限事項があります。

暗号化は、セキュリティの問題をすべて解決するわけではありませんが、ユーザーがデータベースのセキュリティ機能を迂回して、オペレーティング・システムのファイル・システムから直接データベース・ファイルにアクセスしようとした場合に、そのユーザーからデータを保護します。

列暗号化はOracle Databaseの透過的データ暗号化機能を使用しますが、この機能を使用するには、データベースのマスター暗号化キーを保存するためにキーストアを作成する必要があります。暗号化列を含む表を作成する場合、および暗号化データを格納または取得する場合は、キーストアがオープンしている必要があります。キーストアは、オープンするとすべてのセッションで使用可能になり、明示的にクローズするか、データベースが停止されるまではオープンしたままになります。

透過的データ暗号化では、次のタイプの暗号化アルゴリズム(Advanced Encryption Standard (AES)アルゴリズム、Triple Data Encryption Standard (3DES)アルゴリズムなど)を含む、業界標準の暗号化アルゴリズムがサポートされています。

- Advanced Encryption Standard(AES)
- ARIA
- GHOST
- SEED
- Triple Data Encryption Standard

サポートされる暗号化アルゴリズムの詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

使用するアルゴリズムは表の作成時に選択します。表のすべての暗号化列で同じアルゴリズムが使用されます。デフォルトはAES192です。暗号化キーの長さはアルゴリズム名で示されています。たとえば、AES128アルゴリズムでは128ビットのキーが使用されます。

1つ以上の表にある多数の列を暗号化する場合は、かわりに表領域全体を暗号化してその表領域にこれらの表を格納することも考慮できます。表領域の暗号化でも同様に透過的データ暗号化機能が使用されますが、物理的なブロック・レベルで暗号化されるため、多数の列を暗号化するよりパフォーマンスが向上します。表領域レベルで暗号化する別の理由は、列暗号化の次の制限事項に対処するためです。

- オブジェクト・データ型などの特定のデータ型は、列暗号化ではサポートされていません。
- 暗号化列がある表が含まれた表領域に対しては、トランスポータブル表領域機能を使用できません。
- その他の制限の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

関連項目:

- 透過的データ暗号化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。
- キーストアの作成およびオープンの手順は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。
- CREATE TABLE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- Oracle Real Application Clusters環境でのキーストアの使用方法は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

親トピック: 表を管理するためのガイドライン

20.2.14 セグメント作成の遅延の理解

ローカルで管理される表領域内にヒープ構成表を作成すると、最初の行が挿入されるまで表セグメントの作成が遅延されます。

さらに、セグメントの作成は、表のすべてのLOB列、表作成の一環として暗黙的に作成されたすべての索引、および後から明示的にその表に作成されたすべての索引に対して遅延されます。

この領域割当て方法の利点は次のとおりです。

- インストール時に何百もの表が作成され、その表の多くに一度もデータが移入されないようなアプリケーションで、ディスク領域が大幅に削減されます。
- アプリケーションのインストール時間が短縮されます。

最初の行が挿入される際に新しいセグメントを作成する必要があるため、パフォーマンスが多少低下します。

セグメント作成の遅延を有効にするには、互換性を`11.2.0`以上に設定する必要があります。

CREATE TABLE文に新たに導入された句は、次のとおりです。

- `SEGMENT CREATION DEFERRED`
- `SEGMENT CREATION IMMEDIATE`

これらの句は、セグメントの作成を遅延するDEFERRED_SEGMENT_CREATION初期化パラメータのデフォルト設定であるTRUEを上書きします。セグメントの作成の遅延を無効にするには、このパラメータをFALSEに設定します。

セグメント作成を遅延する設定で表を作成すると、新しい表が*_TABLESビューに表示されますが、その表のエントリは最初の行を入力するまで*_SEGMENTSビューに表示されません。

セグメント作成の遅延を確認するには、非パーティション表の場合は*_TABLES、*_INDEXES、*_LOBSの各ビュー、パーティション表の場合は*_TAB_PARTITIONS、*_IND_PARTITIONS、*_LOB_PARTITIONSの各ビューでSEGMENT_CREATED列を参照します。

ノート:

この新しい割当て方法では、適切な容量計画を行い、表へのデータ移入時にセグメントの作成を処理できるだけの十分なディスク領域がデータベースにあるようにすることが重要です。「データベース・オブジェクトの容量計画」を参照してください。

次の例では2つの表を作成し、遅延セグメント作成を実現します。最初の表はSEGMENT CREATION DEFERRED句を使用します。最初はセグメントが作成されません。2番目の表はSEGMENT CREATION IMMEDIATE句を使用するため、セグメントが即時作成されます。

```
CREATE TABLE part_time_employees (
    empno NUMBER(8),
    name VARCHAR2(30),
    hourly_rate NUMBER (7,2)
    )
    SEGMENT CREATION DEFERRED;

CREATE TABLE hourly_employees (
    empno NUMBER(8),
    name VARCHAR2(30),
    hourly_rate NUMBER (7,2)
    )
   SEGMENT CREATION IMMEDIATE
   PARTITION BY RANGE(empno)
    (PARTITION empno_to_100 VALUES LESS THAN (100),
    PARTITION empno_to_200 VALUES LESS THAN (200));
```

USER_SEGMENTSに対する次の問合せにより、HOURLY_EMPLOYEESの行はパーティションごとに1つずつ計2つ返されますが、PART_TIME_EMPLOYEESの行はこの表のセグメント作成が遅延されたため返されません。

```
SELECT segment_name, partition_name FROM user_segments;

SEGMENT_NAME         PARTITION_NAME
-------------------- ------------------------------
HOURLY_EMPLOYEES     EMPNO_TO_100
HOURLY_EMPLOYEES     EMPNO_TO_200
```

USER_TABLESビューには、PART_TIME_EMPLOYEESにセグメントがないことが示されます。

```
SELECT table_name, segment_created FROM user_tables;


TABLE_NAME                     SEGMENT_CREATED
------------------------------ ----------------------------------------
PART_TIME_EMPLOYEES            NO
HOURLY_EMPLOYEES               N/A
```

HOURLY_EMPLOYEES表がパーティション表である場合、SEGMENT_CREATED列はN/Aになっています。これは、USER_TABLESビューには、パーティション表のこの列に関する情報がないためです。次のように、USER_TAB_PARTITIONSビューから参照できます。

```
SELECT table_name, segment_created, partition_name
 FROM user_tab_partitions;
TABLE_NAME           SEGMENT_CREATED      PARTITION_NAME
-------------------- -------------------- ------------------------------
HOURLY_EMPLOYEES     YES                  EMPNO_TO_100
HOURLY_EMPLOYEES     YES                  EMPNO_TO_200
```

次の文は、従業員をこれらの表に追加しています。

```
INSERT INTO hourly_employees VALUES (99, 'FRose', 20.00);
INSERT INTO hourly_employees VALUES (150, 'LRose', 25.00);
```

```
INSERT INTO part_time_employees VALUES (50, 'KReilly', 10.00);
```

前述のように同じSELECT文を繰り返すと、行データが挿入されるため、PART_TIME_EMPLOYEESにセグメントが作成されました。HOURLY_EMPLOYEESは前述のままです。

```
SELECT segment_name, partition_name FROM user_segments;

SEGMENT_NAME          PARTITION_NAME
--------------------- ------------------------------
PART_TIME_EMPLOYEES
HOURLY_EMPLOYEES      EMPNO_TO_100
HOURLY_EMPLOYEES      EMPNO_TO_200

SELECT table_name, segment_created FROM user_tables;

TABLE_NAME            SEGMENT_CREATED
--------------------- --------------------
PART_TIME_EMPLOYEES   YES
HOURLY_EMPLOYEES      N/A
```

USER_TAB_PARTITIONSビューに変更はありません。

関連項目:

セグメント作成の遅延に関するノートおよび制限は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表を管理するためのガイドライン

20.2.15 セグメントのマテリアライズ

DBMS_SPACE_ADMINパッケージにはMATERIALIZE_DEFERRED_SEGMENTS( )プロシージャが含まれており、このプロシージャを使用すると、セグメント作成の遅延が有効であるときに作成された表、表パーティションおよび依存オブジェクトのセグメントをマテリアライズできます。

最初から必要以上のセグメントを設定して不必要にデータベース・リソースを使用するのではなく、必要に応じてセグメントを追加できます。

次の例では、HRスキーマのEMPLOYEES表のセグメントをマテリアライズしています。

```
BEGIN
  DBMS_SPACE_ADMIN.MATERIALIZE_DEFERRED_SEGMENTS(
    schema_name  => 'HR',
    table_name   => 'EMPLOYEES');
END;
```

関連項目:

このプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

親トピック: 表を管理するためのガイドライン

20.2.16 表サイズの見積りと見積りに応じた計画

表を作成する前に表のサイズを見積ります。見積りは、なるべくデータベース計画の一部として実行します。データベース表のサ

イズと用途を確認することは、データベース計画の重要な部分です。

表の見積りサイズの合計と、索引、UNDO領域およびREDOログ・ファイルの見積りを使用して、作成するデータベースを格納するために必要なディスク容量を決定できます。この見積りによって、適切なハードウェアを購入できます。

見積ったサイズと個々の表サイズの増加率を使用すると、作成する表に最適な表領域の属性とその基礎になるデータファイルを的確に判断できます。これによって、表のディスク領域の管理が容易になり、表を使用するアプリケーションのI/Oパフォーマンスが向上します。

関連項目:

「データベース・オブジェクトの容量計画」

親トピック: 表を管理するためのガイドライン

20.2.17 表作成時の制限事項

表を作成するときには制限事項を考慮する必要があります。

表の計画と使用に影響を与える可能性のある制限事項がいくつかあります。

- オブジェクト型を含む表は、Oracle8より古いバージョンのデータベースにインポートできません。
- エクスポートされた表は、異なるスキーマで同じ名前の付いた既存の表にマージできません。
- オリジナルのデータがデータベースにまだ存在するときは、型とエクステント表を異なるスキーマには移動できません。
- Oracle Databaseには、表が持つ列(またはオブジェクト型の属性)の合計数に制限があります。この制限の詳細は、『Oracle Databaseリファレンス』を参照してください。

  ユーザー定義型のデータを含む表を作成すると、ユーザー定義型の列はその型データを格納するリレーショナル列にマップされます。これにより、追加のリレーショナル列が作成されます。これらのリレーショナル列は「非表示」で、DESCRIBE表の文では表示されず、SELECT *文でも返されません。したがって、オブジェクト表、REFの列を持つリレーショナル表、VARRAY、ネストした表またはオブジェクト型を作成するときは、データベースが表に対して実際に作成した列の合計数が、指定した数よりも多くなることがあるので注意してください。

  関連項目:

  ユーザー定義型の詳細は、Oracle Databaseオブジェクト・リレーショナル開発者ガイドを参照

親トピック: 表を管理するためのガイドライン

20.3 表の作成

表はSQL文CREATE TABLEを使用して作成します。

自分のスキーマに新しい表を作成するには、CREATE TABLEシステム権限が必要です。別のユーザーのスキーマに表を作成するには、CREATE ANY TABLEシステム権限が必要です。また、表の所有者には、その表を含む表領域に対する割当て制限またはUNLIMITED TABLESPACEシステム権限が必要です。

- 例: 表の作成
  例を使用して表の作成を説明します。

- 一時表の作成
  一時表は、複数のDML操作の実行によって作成されるため、結果セットがバッファリング(一時的に保存)されるアプリケーションに有用です。グローバル一時表またはプライベート一時表のどちらかを作成できます。
- 表作成のパラレル化
  表の作成にAS SELECT句を指定して、別の表からデータを移入すると、パラレル実行を使用できます。

関連項目:

この章で説明しているCREATE TABLEなどのSQL文の正確な構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表の管理

20.3.1 例: 表の作成

例を使用して表の作成を説明します。

次の文を発行すると、表admin_empがhrスキーマに作成され、admin_tbs表領域に格納されます。

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 表の作成および変更で参照して実行してください。

```
CREATE TABLE hr.admin_emp (
empno NUMBER(5) PRIMARY KEY, ename VARCHAR2(15) NOT NULL, ssn NUMBER(9) ENCRYPT
USING 'AES256', job VARCHAR2(10), mgr NUMBER(5), hiredate DATE DEFAULT (sysdate), photo
BLOB, sal NUMBER(7,2), hrly_rate NUMBER(7,2) GENERATED ALWAYS AS (sal/2080), comm
NUMBER(7,2), deptno NUMBER(3) NOT NULL CONSTRAINT admin_dept_fkey REFERENCES
hr.departments (department_id), comments VARCHAR2(32767), status VARCHAR2(10)
INVISIBLE) TABLESPACE admin_tbs STORAGE ( INITIAL 50K); COMMENT ON TABLE
hr.admin_emp IS 'Enhanced employee table';
```

この例では、次に注意してください。

- 表の複数の列で整合性制約が定義されています。
- STORAGE句では、第1エクステントのサイズが指定されています。この句の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 1つの列(ssn)で、Oracle Databaseの透過的データ暗号化機能を使用した暗号化が定義されています。したがって、このCREATE TABLE文を正常に実行するためには、キーストアがオープンしている必要があります。
- photo列はデータ型がBLOBで、これはラージ・オブジェクト(LOB)と呼ばれるデータ型のセットのメンバーです。LOBは、準構造化データ(XMLツリーなど)および非構造化データ(カラー・イメージのビット・ストリームなど)の保存に使用されます。
- 1つの列(hrly_rate)が仮想列として定義されています。この列は、年収を2,080で除算して従業員の時給を計算しています。仮想列に関するルールの説明は、『Oracle Database SQL言語リファレンス』を参照してください。

- comments列は、4000バイトより大きいVARCHAR2列です。Oracle Database 12cからは、VARCHAR2、NVARCHAR2およびRAWデータ型の最大サイズが32767バイトに増加されました。

  拡張データ型を使用するには、MAX_STRING_SIZE初期化パラメータをEXTENDEDに設定します。このパラメータの設定の詳細は、『Oracle Databaseリファレンス』を参照してください。

- status列は不可視です。

- COMMENT文を使用して、表に関するコメントが格納されています。*_TAB_COMMENTSデータ・ディクショナリ・ビューを問い合せると、このようなコメントを取得できます。詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

関連項目:

- 表の列に指定できるデータ型の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 「整合性制約の管理」
- 「不可視の列の理解」
- 透過的データ暗号化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。
- LOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

親トピック: 表の作成

20.3.2 一時表の作成

一時表は、複数のDML操作の実行によって作成されるため、結果セットがバッファリング(一時的に保存)されるアプリケーションに有用です。グローバル一時表またはプライベート一時表のどちらかを作成できます。

- 一時表の概要
  一時表には、トランザクションまたはセッションの期間中にのみ存在するデータを保持します。
- 一時表作成時の考慮事項
  一時表の作成時には、いくつかの考慮事項がある点に注意してください。
- グローバル一時表の作成
  グローバル一時表は、ディスクに格納される永続的なデータベース・オブジェクトであり、データベースに接続しているすべてのセッションで参照できます。
- プライベート一時表の作成
  プライベート一時表は、トランザクションまたはセッションの終了時に削除される一時データベース・オブジェクトです。プライベート一時表はメモリーに格納され、その表を作成したセッションでのみ参照できます。

親トピック: 表の作成

20.3.2.1 一時表の概要

一時表には、トランザクションまたはセッションの期間中にのみ存在するデータが保持されます。

一時表内のデータはセッション専用です。各セッションで参照および変更できるのは、そのセッション自体のデータのみです。

グローバル一時表または**プライベート一時表**のどちらかを作成できます。次の表に、これらの本質的な相違点を示します。

表20-3 一時表の特徴

特徴	グローバル	プライベート
命名規則	永続表の場合と同じです	先頭に ORA$PTT_を付ける必要があります
表定義の可視性	すべてのセッション	表を作成したセッションのみ
表定義の記憶域	ディスク	メモリーのみ
タイプ	トランザクション固有(ON COMMIT DELETE ROWS)、またはセッション固有(ON COMMIT PRESERVE ROWS)	トランザクション固有(ON COMMIT DROP DEFINITION)、またはセッション固有(ON COMMIT PRESERVE DEFINITION)

3番目のタイプの一時表は、**cursor-duration一時表**と呼ばれるもので、特定のタイプの問合せに対してデータベースによって自動的に作成されます。

関連項目:

cursor-duration一時表についてさらに学習するには、『Oracle Database SQLチューニング・ガイド』を参照してください。

親トピック: 一時表の作成

20.3.2.2 一時表作成時の考慮事項

一時表の作成時には、いくつかの考慮事項がある点に注意してください。

永続表とは異なり、一時表の作成時にはセグメントは自動的に割り当てられません。かわりに、最初にINSERT(またはCREATE TABLE AS SELECT)が実行されると、セグメントが割り当てられます。したがって、最初のINSERTの前に、SELECT、UPDATEまたはDELETEが実行されると、表が空に見えます。

既存の一時表でDDL操作(TRUNCATEを除く)が許可されるのは、その一時表にバインドされているセッションがない場合のみです。

トランザクションをロールバックすると、入力したデータは消失しますが、表定義はそのまま残ります。

トランザクション固有の一時表では、1回に1トランザクションのみが許可されます。単一のトランザクションに複数の自律型トランザクションがある場合、各自律型トランザクションは、直前のトランザクションのコミット直後にのみ表を使用できます。

一時表のデータは、その定義どおり一時的なため、一時表データのバックアップとリカバリはシステム障害のイベントでは使用できません。このような障害に備えて、一時表データを保存する代替方法を用意してください。

親トピック: 一時表の作成

20.3.2.3 グローバル一時表の作成

グローバル一時表は、ディスクに格納される永続的なデータベース・オブジェクトであり、データベースに接続しているすべてのセッションで参照できます。

- グローバル一時表の作成について
  グローバル一時表のメタデータは、複数のユーザーとそのユーザーのセッションで参照できますが、その内容はセッションに対してローカルになります。
- 例: グローバル一時表の作成

グローバル一時表の作成方法を例で示します。

親トピック: 一時表の作成

20.3.2.3.1 グローバル一時表の作成について

グローバル一時表のメタデータは、複数のユーザーとそのユーザーのセッションで参照できますが、その内容はセッションに対してローカルになります。

たとえば、Webベースの航空予約アプリケーションでは、顧客がオプションの旅程を複数作成できます。各旅程はグローバル一時表の行で表されます。アプリケーションは、旅程への変更を反映するように行を更新します。使用する旅程を顧客が決定すると、アプリケーションは、該当する旅程の行を永続表に移動します。

セッションの開始時から終了時まで旅程データはプライベートです。セッションの終了時に、オプションの旅程は削除されます。

グローバル一時表の定義はすべてのセッションで参照できますが、グローバル一時表内のデータを参照できるのは、そのデータを表に挿入するセッションのみです。

グローバル一時表は、CREATE GLOBAL TEMPORARY TABLE文を使用して作成します。ON COMMIT句は、表内のデータがトランザクション固有(デフォルト)またはセッション固有のいずれであるかを示し、それぞれの意味は次のとおりです。

ON COMMIT設定	意味
DELETE ROWS	トランザクション固有のグローバル一時表を作成します。セッションは、最初に表に挿入するトランザクションでグローバル一時表にバインドされます。バインドは、トランザクション終了時に消失します。表は、各コミット後に切捨て(すべての行を削除)が行われます。
PRESERVE ROWS	セッション固有のグローバル一時表を作成します。セッションは、そのセッションで最初の表への挿入でグローバル一時表にバインドされます。このバインドは、セッションの最後で、またはセッション内で表に対する TRUNCATE が発行されることによって消去されます。表は、セッション終了時に切り捨てられます。

親トピック: グローバル一時表の作成

20.3.2.3.2 例: グローバル一時表の作成

グローバル一時表の作成方法を例で示します。

次の文では、トランザクション固有のグローバル一時表を作成します。

```
CREATE GLOBAL TEMPORARY TABLE admin_work_area_trans
(startdate DATE,
enddate DATE,
class CHAR(20))
ON COMMIT DELETE ROWS;
```

次の文では、セッション固有のグローバル一時表を作成します。

```
CREATE GLOBAL TEMPORARY TABLE admin_work_area_session
(startdate DATE,
enddate DATE,
class CHAR(20))
ON COMMIT PRESERVE ROWS;
```

グローバル一時表には索引を作成できます。この索引も一時索引であり、索引内のデータのセッションまたはトランザクションの

有効範囲は、基礎になる表のデータと同じです。

デフォルトでは、グローバル一時表の行は、それを作成したユーザーのデフォルト一時表領域に保存されます。ただし、グローバル一時表の作成時にCREATE GLOBAL TEMPORARY TABLEのTABLESPACE句を使用することで、別の表領域にグローバル一時表を割り当てることができます。この機能を使用すると、グローバル一時表に使用する領域を節約できます。たとえば、多数の小さなグローバル一時表操作を実行する必要があり、デフォルトの一時表領域がソート操作用に構成されていて、かつ大きなエクステント・サイズを使用する場合、これらの小さな操作は大量の不必要なディスク領域を消費します。このような場合、小さいエクステント・サイズを持つ別の一時表領域を割り当てることをお薦めします。

次の2つの文では、64KBのエクステント・サイズでグローバル一時表領域を作成して、その表領域内に新しい一時表を作成します。

```
CREATE TEMPORARY TABLESPACE tbs_t1
TEMPFILE 'tbs_t1.f' SIZE 50m REUSE AUTOEXTEND ON
MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;
CREATE GLOBAL TEMPORARY TABLE admin_work_area
(startdate DATE,
enddate DATE,
class CHAR(20))
ON COMMIT DELETE ROWS
TABLESPACE tbs_t1;
```

関連項目:

- 「一時表領域について」
- グローバル一時表を作成するためのCREATE TABLE文の使用方法と適用される制限の詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: グローバル一時表の作成

20.3.2.4 プライベート一時表の作成

プライベート一時表は、トランザクションまたはセッションの終了時に削除される一時データベース・オブジェクトです。プライベート一時表はメモリーに格納され、その表を作成したセッションでのみ参照できます。

- プライベート一時表の作成について
  プライベート一時表のメタデータと内容は、それを作成したセッション内でのみ参照できます。
- 例: プライベート一時表の作成
  プライベート一時表の作成例を示します。

親トピック: 一時表の作成

20.3.2.4.1 プライベート一時表の作成について

プライベート一時表のメタデータと内容は、それを作成したセッション内でのみ参照できます。

プライベート一時表は、次のような状況で役立ちます。

- アプリケーションで、データを1回移入して、数回の読込み後、トランザクションまたはセッションの終了時に破棄する一時的な表に一時データを保存する場合
- セッションが無期限に維持され、別のトランザクション用に別の一時表を作成する必要がある場合
- 一時表の作成時に、新しいトランザクションの開始や既存のトランザクションのコミットができない場合

- 同じユーザーの異なるセッションで一時表に同じ名前を使用する必要がある場合
- 読取り専用データベースに一時表が必要な場合

たとえば、1つのスキーマのみを使用するレポート・アプリケーションがあるとします。このアプリケーションは異なるレポートを実行するために、そのスキーマで複数の接続を使用するとします。セッションはそれぞれのトランザクションで計算用にプライベート一時表を使用し、各セッションは同じ名前のプライベート一時表を作成します。各トランザクションがコミットされると、その一時データは不要になります。プライベート一時表の定義とプライベート一時表内のデータは、どちらも表を作成したセッションでのみ参照できます。

プライベート一時表は、CREATE PRIVATE TEMPORARY TABLE文を使用して作成します。ON COMMIT句は、表内のデータがトランザクション固有(デフォルト)またはセッション固有のいずれであるかを示し、それぞれの意味は次のとおりです。

ON COMMIT設定	意味
DROP DEFINITION	トランザクション固有のプライベート一時表を作成します。表内のすべてのデータが失われ、トランザクションの終了時に表が削除されます。
PRESERVE DEFINITION	セッション固有のプライベート一時表を作成します。表内のすべてのデータが失われ、表を作成したセッションの終了時に表が削除されます。

ノート:

プライベート一時表の名前は、初期化パラメータ private_temp_table_prefix に従って接頭辞を付ける必要があります。

親トピック: プライベート一時表の作成

20.3.2.4.2 例: プライベート一時表の作成

プライベート一時表の作成例を示します。

次の文では、トランザクション固有のプライベート一時表を作成します。

```
CREATE PRIVATE TEMPORARY TABLE ORA$PTT_sales_ptt_transaction
    (time_id      DATE,
     amount_sold  NUMBER(10,2))
   ON COMMIT DROP DEFINITION;
```

次の文では、セッション固有のプライベート一時表を作成します。

```
CREATE PRIVATE TEMPORARY TABLE ORA$PTT_sales_ptt_session
    (time_id      DATE,
     amount_sold  NUMBER(10,2))
   ON COMMIT PRESERVE DEFINITION;
```

デフォルトでは、プライベート一時表の行は、それを作成したユーザーのデフォルト一時表領域に保存されます。ただし、一時表の作成時にCREATE PRIVATE TEMPORARY TABLEのTABLESPACE句を使用することで、別の表領域に一時表を割り当てることができます。

関連項目:

- 「一時表領域について」
- プライベート一時表を作成するためのCREATE TABLE文の使用方法と適用される制限の詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: プライベート一時表の作成

20.3.3 表作成のパラレル化

表の作成にAS SELECT句を指定して、別の表からデータを移入すると、パラレル実行を使用できます。

CREATE TABLE...AS SELECT文には、CREATE部分(DDL)とSELECT部分(問合せ)の2つの部分があります。Oracle Databaseでは、この文の両方の部分をパラレル化できます。CREATEの部分がパラレル化されるのは、次の中の1つに該当する場合です。

- PARALLEL句がCREATE TABLE...AS SELECT文に含まれている。
- ALTER SESSION FORCE PARALLEL DDL文が指定されている。

問合せ部分がパラレル化されるのは、次のすべてに該当する場合です。

- 問合せにパラレル・ヒントの指定(PARALLELまたはPARALLEL_INDEX)が含まれるか、またはCREATEの部分にPARALLEL句が含まれているか、または問合せで参照されているスキーマ・オブジェクトにPARALLEL宣言が関連付けられている。
- 問合せに指定された少なくとも1つの表で、全表スキャンまたは複数のパーティションに及ぶ索引レンジ・スキャンが必要である。

表の作成をパラレル化した場合、その表には対応付けられたパラレル宣言(PARALLEL句)が付きます。表に対するその後のすべてのDMLまたは問合せでは、パラレル化が可能な場合、パラレル実行の使用が試みられます。

表の作成をパラレル化し、表圧縮を使用して圧縮形式で結果を格納する簡単な文を次に示します。

```
CREATE TABLE hr.admin_emp_dept
     PARALLEL COMPRESS
     AS SELECT * FROM hr.employees
     WHERE department_id = 10;
```

この場合のPARALLEL句は、表の作成時に最適な数のパラレル実行サーバーを選択することをデータベースに指示しています。

関連項目:

- パラレル実行の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。
- 「SQLのパラレル実行用プロセスの管理」

親トピック: 表の作成

20.4 表のロード

データを表にロードするための手法について説明します。

ノート:

パーティション表のすべての新規セグメントについて、第 1 エクステントのデフォルトのサイズは 64KB ではなく 8MB です。このことは、パーティション表に対する挿入と問合せのパフォーマンス向上に役立ちます。パーティション表の初期サイズが大きくても、十分なデータが挿入されると、領域消費は以前のリリースと同じになります。このデフォルトは、表の記憶域句に INITIAL サイズを設定して上書きできます。この新しいデフォルトは、表パーティションおよび LOB パーティションにのみ適用されます。

- 表のロード方法
  表にデータを挿入または初期ロードするには、いくつかの方法があります。
- ダイレクト・パスINSERTを使用したINSERTパフォーマンスの向上
  大量のデータをロードするときは、ダイレクト・パスINSERTを使用してロードのパフォーマンスを高めることができます。
- 従来型のインサートを使用した表のロード
  従来型のINSERT処理では、表内の空き領域が再利用され、新規に挿入するデータと既存のデータが相互に配置されます。このような操作では、参照整合性制約も維持されます。ダイレクト・パスINSERT操作と異なり、従来型のINSERT操作は表に対する排他的ロックを必要としません。
- DMLエラー・ロギングを使用したバルクINSERT失敗の回避
  DMLエラー・ロギング機能を使用して、バルクINSERTの失敗を回避できます。

親トピック: 表の管理

20.4.1 表のロード方法

表にデータを挿入または初期ロードするには、いくつかの方法があります。

最も一般的に使用される方法は、次のとおりです。

方法	説明
SQL*Loader	これは、外部ファイルから Oracle Database の表にデータをロードする Oracle のユーティリティ・プログラムです。 Oracle Database 12c からは、SQL*Loader でエクスプレス・モードがサポートされます。SQL*Loader のエクスプレス・モードを使用すると、ファイルを制御する必要がなくなります。エクスプレス・モードにより、外部ファイルからのデータのロードが簡素化されます。エクスプレス・モードでは、SQL*Loader は外部表ロード・メソッドの使用を試行します。外部表ロード・メソッドが使用できない場合、SQL*Loader はダイレクト・パスの使用を試行します。ダイレクト・パスが使用できない場合、SQL*Loader は従来型パスを使用します。 SQL*Loader のエクスプレス・モードでは、表の列の型に基づいて入力データ型が自動的に識別され、並列度が制御されます。SQL*Loader ではデフォルトを使用して使用方法が簡素化されますが、多くのデフォルトはコマンド・ライン・パラメータで上書きできます。必要に応じて、エクスプレス・モードを使用するかわりに、ダイレクト・パスまたは従来型パスのロード・メソッドを指定できます。 SQL*Loader の詳細は、『Oracle Database ユーティリティ』を参照してください。

方法	説明
CREATE TABLE ... AS SELECT 文(CTAS)	この SQL 文を使用すると、表を作成し、外部表を含む別の既存の表から選択したデータを移入できます。
INSERT 文	INSERT 文を使用すると、列値を指定するか、または外部表を含む別の既存の表からデータを選択する副問合せを指定することによって、行を表に追加できます。 INSERT 文の 1 つの形式では、ダイレクト・パス INSERT が使用可能になり、これよってパフォーマンスを改善でき、バルク・ロードの際に役立ちます。「ダイレクト・パス・インサートを使用した INSERT パフォーマンスの向上」を参照してください。 大量のデータを挿入するときにエラーが発生した場合の文の終了およびロールバックを回避するには、DML エラー・ロギングを指定して挿入できます。「DML エラー・ロギングを使用したバルク INSERT 失敗の回避」を参照してください。
MERGE 文	MERGE 文を使用すると、別の既存の表から行を選択することによって、行を表に挿入するか、または表の行を更新できます。新しいデータの行が、表にすでに存在している項目に対応している場合は UPDATE が実行され、対応する項目がない場合は INSERT が実行されます。

ノート:

このマニュアルに記載されている、表にデータを挿入する詳細と例は少数です。データ・ウェアハウスおよびアプリケーション開発に関するオラクル社のマニュアルには、表へのデータの挿入および操作に関する広範囲にわたる情報が記載されています。参照:

- Oracle Database データ・ウェアハウス・ガイド
- Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド

関連項目:

「外部表の管理」

親トピック: 表のロード

20.4.2 ダイレクト・パスINSERTを使用したINSERTパフォーマンスの向上

大量のデータをロードするときは、ダイレクト・パスINSERTを使用してロードのパフォーマンスを高めることができます。

- ダイレクト・パスINSERTについて
  ダイレクト・パス・インサート操作は、一般に従来の挿入操作より高速です。
- ダイレクト・パスINSERTの動作

ダイレクト・パスINSERTは、パーティション表と非パーティション表の両方で使用できます。

- ダイレクト・パスINSERTを使用したデータのロード
  ダイレクト・パス INSERT SQL文を使用してパラレル・モードでデータを挿入するか、またはOracleのSQL*Loaderユーティリティをダイレクト・パス・モードで使用することによって、ダイレクト・パス INSERTを使用してデータをロードできます。ダイレクト・パスINSERTは、シリアル・モードまたはパラレル・モードで実行できます。
- ダイレクト・パスINSERTのロギング・モード
  ダイレクト・パスINSERTでは、インサート処理中のREDOおよびUNDO情報を記録するかどうかを選択できます。
- ダイレクト・パスINSERTのその他の考慮事項

親トピック: 表のロード

20.4.2.1 ダイレクト・パスINSERTについて

ダイレクト・パス・インサート操作は、一般に従来の挿入操作より高速です。

Oracle Databaseでは、次の2つのいずれかの方法でデータが挿入されます。

- 従来型のINSERT処理では、表内の空き領域が再利用され、新規に挿入するデータと既存のデータが相互に配置されます。このような操作では、参照整合性制約も維持されます。
- ダイレクト・パスINSERT処理では、表内の既存データの後ろに挿入データが追加されます。データは、バッファ・キャッシュを回避してデータファイルに直接書き込まれます。表の空き領域は再利用されず、参照整合性制約は無視されます。ダイレクト・パスINSERTは従来型のインサートよりもパフォーマンスが大幅に優れています。

データベースは、1つのプロセスが文を実行するシリアル・モードか、同時に多数のプロセスが連携して1つのSQL文を実行するパラレル・モードでデータを挿入できます。後者はパラレル実行と呼ばれます。

ダイレクト・パスINSERTの利点は、次のとおりです。

- ダイレクト・パスINSERTでは、REDOおよびUNDOエントリのロギングを使用禁止にしてロード時間を削減できます。これに対して、従来型のインサート処理では空き領域を再利用し、参照整合性を維持するため、これらのエントリを常にロギングする必要があります。
- ダイレクト・パスINSERT処理は、パラレル・モードで実行する場合でも、トランザクションの原子性が保証されます。原子性は、パラレル・ダイレクト・パス・ロード(SQL*Loaderを使用)では保証されません。

パラレル・ダイレクト・パス・ロード実行時のSQL*LoaderとINSERT文の間の大きな違いの1つは、SQL*Loaderによるパラレル・ダイレクト・パス・ロード中にエラーが発生した場合、ロードは完了しますが、一部の索引にはロードの終了時にUNUSABLEのマークが付くことです。対照的に、パラレル・ダイレクト・パスINSERTの場合は、索引更新時にエラーが発生すると、文がロールバックされます。

ノート:

従来の INSERT 操作では、挿入中の NOT NULL 制約の違反をチェックします。そのため、従来の INSERT 操作の NOT NULL 制約に違反すると、挿入時にエラーが返されます。ダイレクト・パス INSERT 操作では、挿入前に NOT NULL 制約の違反をチェックします。そのため、ダイレクト・パス INSERT 操作の NOT NULL 制約に違反した場合は、挿入前にエラーが返されます。

親トピック: ダイレクト・パスINSERTを使用したINSERTパフォーマンスの向上

20.4.2.2 ダイレクト・パスINSERTの動作

ダイレクト・パスINSERTは、パーティション表と非パーティション表の両方で使用できます。

- パーティション表または非パーティション表へのシリアル・ダイレクト・パスINSERT
  単一のプロセスは、表セグメントまたは各パーティション・セグメントの現在の最高水位標を超えてもデータを挿入します。(**最高水位標**とは、ブロックがデータを受け取るためにフォーマットされたことのないレベルを指します。)COMMITが実行されると、最高水位標が新しい値に更新され、データがユーザーに表示可能になります。
- パーティション表へのパラレル・ダイレクト・パスINSERT
  この状況はシリアル・ダイレクト・パスINSERTに似ています。各パラレル実行サーバーが1つ以上のパーティションに割り当てられ、1つのパーティションでは1つのプロセスしか実行されません。
- 非パーティション表へのパラレル・ダイレクト・パスINSERT
  各パラレル実行サーバーは、新しい一時セグメントを割り当て、その一時セグメントにデータを挿入します。COMMITを実行すると、パラレル実行コーディネータが新しい一時セグメントをプライマリ表セグメントにマージし、ユーザーにデータが表示されます。

親トピック: ダイレクト・パスINSERTを使用したINSERTパフォーマンスの向上

20.4.2.2.1 パーティション表または非パーティション表へのシリアル・ダイレクト・パスINSERT

単一のプロセスは、表セグメントまたは各パーティション・セグメントの現在の最高水位標を超えてもデータを挿入します。(**最高水位標**とは、ブロックがデータを受け取るためにフォーマットされたことのないレベルを指します。)COMMITが実行されると、最高水位標が新しい値に更新され、データがユーザーに表示可能になります。

親トピック: ダイレクト・パスINSERTの動作

20.4.2.2.2 パーティション表へのパラレル・ダイレクト・パスINSERT

この状況は、シリアル・ダイレクト・パスINSERTと類似しています。各パラレル実行サーバーが1つ以上のパーティションに割り当てられ、1つのパーティションでは1つのプロセスしか実行されません。

各パラレル実行サーバー割り当てられたパーティション・セグメントの現在の最高水位標を超えてもデータを挿入します。COMMITが実行されると、各パーティション・セグメントの最高水位標が新しい値に更新され、データがユーザーに表示可能になります。

親トピック: ダイレクト・パスINSERTの動作

20.4.2.2.3 非パーティション表へのパラレル・ダイレクト・パスINSERT

各パラレル実行サーバーは、新しい一時セグメントを割り当て、その一時セグメントにデータを挿入します。COMMITを実行すると、パラレル実行コーディネータが新しい一時セグメントをプライマリ表セグメントにマージし、ユーザーにデータが表示されます。

親トピック: ダイレクト・パスINSERTの動作

20.4.2.3 ダイレクト・パスINSERTを使用したデータのロード

ダイレクト・パスINSERT SQL文を使用してパラレル・モードでデータを挿入するか、またはOracleのSQL*Loaderユーティリティをダイレクト・パス・モードで使用することによって、ダイレクト・パスINSERTを使用してデータをロードできます。ダイレクト・パスINSERTは、シリアル・モードまたはパラレル・モードで実行できます。

- SQL文を使用したシリアル・モード・インサート
  SQL文を使用したシリアル・モードのダイレクト・パスINSERTは様々な方法でアクティブにできます。
- SQL文を使用したパラレル・モード・インサート

パラレル・モードで挿入する場合は、ダイレクト・パスINSERTがデフォルトです。ただし、NOAPPEND PARALLELヒントを使用して、従来型のINSERTを使用したパラレル・モードでの挿入も実行できます。

親トピック: ダイレクト・パスINSERTを使用したINSERTパフォーマンスの向上

20.4.2.3.1 SQL文を使用したシリアル・モード・インサート

SQL文を使用したシリアル・モードのダイレクト・パスINSERTは様々な方法でアクティブにできます。

SQLを使用したシリアル・モードでのダイレクト・パスINSERTは、次の方法でアクティブ化します。

- 副問合せを使用してINSERTを実行している場合は、INSERTキーワードの直後またはINSERT文の副問合せのSELECTキーワードの直後にある各INSERT文にAPPENDヒントを指定します。
- VALUES句を使用してINSERTを実行している場合は、INSERTキーワードの直後にある各INSERT文にAPPEND_VALUESヒントを指定します。VALUES句を使用するダイレクト・パスINSERTは、ロードする行が数百、数千またはさらに膨大な数になる場合の使用が最適です。一般的な使用例として、OCIを使用した配列の挿入があります。また、PL/SQLのFORALL文での挿入に使用する例もあります。

VALUES句を使用するINSERT文にAPPENDヒント(APPEND_VALUESヒントではなく)を指定すると、このAPPENDヒントは無視され、従来の挿入が実行されます。

ダイレクト・パスINSERTを実行するためにAPPENDヒントを使用する例を次に示します。

```
INSERT /*+ APPEND */ INTO sales_hist SELECT * FROM sales WHERE cust_id=8890;
```

次のPL/SQLコードの一部は、APPEND_VALUESヒントの使用例です。

```
FORALL i IN 1..numrecords
  INSERT /*+ APPEND_VALUES */ INTO orderdata
  VALUES(ordernum(i), custid(i), orderdate(i),shipmode(i), paymentid(i));
COMMIT;
```

親トピック: ダイレクト・パスINSERTを使用したデータのロード

20.4.2.3.2 SQL文を使用したパラレル・モード・インサート

パラレル・モードで挿入する場合は、ダイレクト・パスINSERTがデフォルトです。ただし、NOAPPEND PARALLELヒントを使用して、従来型のINSERTを使用したパラレル・モードでの挿入も実行できます。

パラレルDMLモードで実行するには、次の要件を満たす必要があります。

- Oracle Enterprise Editionがインストールされていること。
- セッションでパラレルDMLが使用可能であること。そのためには、次の文を発行します。

  ```
  ALTER SESSION { ENABLE | FORCE } PARALLEL DML;
  ```

- 次の要件を最低1つ満たす必要があります。
  - ターゲット標のパラレル属性を、作成時またはその後に指定すること。
  - 各挿入操作に対してPARALLELヒントを指定すること。
  - データベースの初期化パラメータPARALLEL_DEGREE_POLICYをAUTOに指定すること。

ダイレクト・パスINSERTを使用禁止にするには、各INSERT文にNOAPPENDヒントを指定します。この指定によって、パラレルDMLモードが無視されます。

ノート:

ダイレクト・パス INSERT によって挿入したデータを、挿入した直後に問合せまたは変更することはできません。問合せまたは変更を試みると、ORA-12838 エラーが発生します。新しく挿入したデータの読取りまたは変更を試みる前に、COMMIT 文を発行する必要があります。

関連項目:

- 「従来型のインサートを使用した表のロード」
- ヒントの使用方法の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- INSERT文の副問合せ構文の詳細およびダイレクト・パスINSERTの使用に関する追加の制限事項は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: ダイレクト・パスINSERTを使用したデータのロード

20.4.2.4 ダイレクト・パスINSERTのロギング・モード

ダイレクト・パスINSERTでは、インサート処理中のREDOおよびUNDO情報を記録するかどうかを選択できます。

ダイレクト・パスINSERTのロギング・モードは次のように指定します。

- 作成時(CREATE文で)または作成後(ALTER文で)に、表、パーティション、索引またはLOB記憶域について、ロギング・モードを指定できます。
- 作成時または作成後にLOGGINGまたはNOLOGGINGを指定しないと、次のようにデフォルト設定されます。
  - パーティションのロギング属性は、その表のロギング属性にデフォルト設定されます。
  - 表または索引のロギング属性は、常駐している表領域のロギング属性にデフォルト設定されます。
  - LOB記憶域のロギング属性は、LOB記憶域にCACHEを指定した場合はLOGGINGにデフォルト設定されます。CACHEを指定しなかった場合、ロギング属性はLOB値が常駐している表領域の属性にデフォルト設定されます。
- CREATE TABLESPACEまたはALTER TABLESPACE文で、表領域のロギング属性を設定します。

ノート:

データベースまたは表領域が FORCE LOGGING モードの場合、ダイレクト・パス INSERT は、ロギング設定に関係なく常にログに記録されます。

- ロギング付きダイレクト・パスINSERT
  このモードでは、Oracle Databaseによってインスタンスおよびメディア・リカバリの完全なREDOロギングが実行されます。
- ロギングなしダイレクト・パスINSERT
  このモードでは、REDOロギングまたはUNDOロギングを実行せずにOracle Databaseによってデータが挿入されます。かわりにデータベースは少数のブロック範囲無効化REDOレコードをロギングし、定期的に最新の直接書込みに関する情報で制御ファイルを更新します。

親トピック: ダイレクト・パスINSERTを使用したINSERTパフォーマンスの向上

20.4.2.4.1 ロギング付きダイレクト・パスINSERT

このモードでは、Oracle Databaseによってインスタンスの完全なREDOロギングおよびメディア・リカバリが実行されます。

データベースがARCHIVELOGモードの場合は、REDOログをテープにアーカイブできます。データベースがNOARCHIVELOGモードの場合、インスタンスのクラッシュはリカバリできますが、ディスク障害はリカバリできません。

親トピック: ダイレクト・パスINSERTのロギング・モード

20.4.2.4.2 ロギングなしダイレクト・パスINSERT

このモードでは、Oracle DatabaseはデータをREDOまたはUNDOロギングなしでデータを挿入します。かわりにデータベースは少数のブロック範囲無効化REDOレコードをロギングし、定期的に最新の直接書込みに関する情報で制御ファイルを更新します。

ロギングなしダイレクト・パスINSERTを使用すると、パフォーマンスが向上します。ただし、後でメディア・リカバリを実行する必要がある場合は、REDOデータがロギングされていないため、無効化REDOレコードによって一連のブロックに論理的破損のマークが付きます。したがって、このようなインサート処理の後にはデータをバックアップすることが重要です。

制御ファイルの定期更新を無効にして、リカバリ不能なダイレクト・パス・インサートのパフォーマンスを大幅に向上させることができます。このことを行うには、DB_UNRECOVERABLE_SCN_TRACKING初期化パラメータをFALSEに設定します。ただし、これらの制御ファイルの更新を無効化してリカバリ不能なダイレクト・パス・インサートを実行すると、データファイルが現在リカバリ不能かどうかをデータベースに問い合せて正確に判断できなくなります。

関連項目:

- リカバリ不能なデータファイルの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- 『Oracle Data Guard概要および管理』のリカバリ不能操作後にバックアップが必要かどうかの判断に関する項を参照してください。

親トピック: ダイレクト・パスINSERTのロギング・モード

20.4.2.5 ダイレクト・パスINSERTのその他の考慮事項

ダイレクト・パスINSERTを使用する際は、圧縮表、索引メンテナンス、ディスク領域およびロック関連の問題を考慮してください。

- 圧縮表とダイレクト・パスINSERT
  表が基本表圧縮で作成されている場合は、ダイレクト・パスINSERTを使用して、表のデータをロード時に圧縮する必要があります。表が拡張行、ウェアハウスまたはアーカイブ圧縮で作成されている場合は、ダイレクト・パスINSERTによって最適な圧縮率が得られます。
- ダイレクト・パスINSERTでの索引メンテナンス
  索引がある(パーティションまたは非パーティション)表では、ダイレクト・パスINSERT処理の終了時に、Oracle Databaseが索引メンテナンスを実行します。
- ダイレクト・パスINSERTでの領域に関する考慮事項
  ダイレクト・パスINSERTは、従来型パスINSERTよりも多くの領域を必要とします。
- ダイレクト・パスINSERTでのロックに関する考慮事項
  ダイレクト・パスINSERT中は、データベースは表(またはパーティション表のすべてのパーティション)に対して排他的ロック

を取得します。

親トピック: ダイレクト・パスINSERTを使用したINSERTパフォーマンスの向上

20.4.2.5.1 圧縮表とダイレクト・パスINSERT

表が基本表圧縮で作成されている場合は、ダイレクト・パスINSERTを使用して、表のデータをロード時に圧縮する必要があります。表が拡張行、ウェアハウスまたはアーカイブ圧縮で作成されている場合は、ダイレクト・パスINSERTによって最適な圧縮率が得られます。

詳細は、「表圧縮の使用」を参照してください。

親トピック: ダイレクト・パスINSERTのその他の考慮事項

20.4.2.5.2 ダイレクト・パスINSERTでの索引メンテナンス

索引がある(パーティションまたは非パーティション)表では、ダイレクト・パスINSERT処理の終了時に、Oracle Databaseが索引メンテナンスを実行します。

この索引メンテナンスは、パラレル・ダイレクト・パスINSERTに対してはパラレル実行サーバーで、シリアル・ダイレクト・パスINSERTに対してはシングル・プロセスで実行されます。INSERT処理の前に索引を使用禁止にし、後で再作成することによって、索引メンテナンスでのパフォーマンスへの影響を回避できます。

関連項目:

「索引の使用禁止化」

親トピック: ダイレクト・パスINSERTのその他の考慮事項

20.4.2.5.3 ダイレクト・パスINSERTでの領域に関する考慮事項

ダイレクト・パスINSERTは、従来型パスINSERTよりも多くの領域を必要とします。

すべてのシリアル・ダイレクト・パスINSERT処理では、パーティション表へのパラレル・ダイレクト・パスINSERTと同様に、影響を受けるセグメントの最高水位標の上にデータが挿入されます。このため、追加の領域が必要となります。

非パーティション表へのパラレル・ダイレクト・パスINSERTは、各並列度ごとに一時セグメントを作成するため、より多くの領域を必要とします。非パーティション表が自動セグメント領域管理モードのローカル管理表領域にない場合は、NEXTおよびPCTINCREASE記憶域パラメータ、およびMINIMUM EXTENT表領域パラメータの値を変更して、一時セグメントに十分な(かつ過剰ではない)記憶域を用意してください。次の事項を考慮に入れ、これらのパラメータに値を選択します。

- 各エクステントのサイズは、さほど小さくありません(1MB以上)。この設定は、オブジェクト内のエクステント総数に影響を与えます。
- 各エクステントのサイズが小さいと、パラレルINSERTでは、必要以上に大きいセグメントで領域を無駄にすることになります。

これらのパラメータは、ダイレクト・パスINSERT処理の完了後に、シリアル処理に適した設定に再設定できます。

親トピック: ダイレクト・パスINSERTのその他の考慮事項

20.4.2.5.4 ダイレクト・パスINSERTでのロックに関する考慮事項

ダイレクト・パスINSERT中は、データベースは表(またはパーティション表のすべてのパーティション)に対して排他的ロックを取得します。

その結果、ユーザーは同時挿入、更新、または削除操作を表に対して実行できず、同時索引作成および構築操作は許可されません。ただし、同時問合せはサポートされますが、問合せで返されるのは挿入操作前の情報のみです。

親トピック: ダイレクト・パスINSERTのその他の考慮事項

20.4.3 従来型のインサートを使用した表のロード

従来型のINSERT処理では、表内の空き領域が再利用され、新規に挿入するデータと既存のデータが相互に配置されます。このような操作では、参照整合性制約も維持されます。ダイレクト・パスINSERT操作と異なり、従来型のINSERT操作は表に対する排他的ロックを必要としません。

ダイレクト・パスINSERT操作に適用される他のいくつかの制約も、従来型のINSERT操作には該当しません。これらの制限事項については、『Oracle Database SQL言語リファレンス』を参照してください。

従来型のINSERT操作は、NOAPPENDヒントを使用して、シリアル・モードまたはパラレル・モードで実行できます。

従来型のINSERTをシリアル・モードで実行するためにNOAPPENDヒントを使用する例を次に示します。

```
INSERT /*+ NOAPPEND */ INTO sales_hist SELECT * FROM sales WHERE cust_id=8890;
```

従来型のINSERTをパラレル・モードで実行するためにNOAPPENDヒントを使用する例を次に示します。

```
INSERT /*+ NOAPPEND PARALLEL */ INTO sales_hist
   SELECT * FROM sales;
```

パラレルDMLモードで実行するには、次の要件を満たす必要があります。

- Oracle Enterprise Editionがインストールされていること。
- セッションでパラレルDMLが使用可能であること。そのためには、次の文を発行します。

  ```
  ALTER SESSION { ENABLE | FORCE } PARALLEL DML;
  ```

- 次の要件を最低1つ満たす必要があります。
  - ターゲット標のパラレル属性を、作成時またはその後に指定すること。
  - 各挿入操作に対してPARALLELヒントを指定すること。
  - データベースの初期化パラメータPARALLEL_DEGREE_POLICYをAUTOに指定すること。

親トピック: 表のロード

20.4.4 DMLエラー・ロギングを使用したバルクINSERT失敗の回避

DMLエラー・ロギング機能を使用して、バルクINSERTの失敗を回避できます。

- DMLエラー・ロギングを使用したデータ挿入
  副問合せでINSERT文を使用して表をロードすると、エラーが発生した場合は文が終了して文全体がロールバックされます。これは、時間とシステム・リソースを無駄に消費することになります。このようなINSERT文の場合は、DMLエラー・ロギング機能を使用することで、この状況を回避できます。
- エラー・ロギング表の書式
  エラー・ロギング表には固有の書式があります。
- エラー・ロギング表の作成
  エラー・ロギング表は手動で作成できます。または、PL/SQLパッケージを使用して自動的に作成できます。
- エラー・ロギングの制限事項と注意

一部のエラーはエラー・ロギング表に記録されません。

親トピック: 表のロード

20.4.4.1 DMLエラー・ロギングを使用したデータ挿入

副問合せでINSERT文を使用して表をロードすると、エラーが発生した場合は文が終了して文全体がロールバックされます。これは、時間とシステム・リソースを無駄に消費することになります。このようなINSERT文の場合は、DMLエラー・ロギング機能を使用することで、この状況を回避できます。

DMLエラー・ロギングを使用するには、データベースがDML操作中に検出したエラーを記録するエラー・ロギング表の名前を指定する文の句を追加します。このエラー・ロギング句をINSERT文に追加すると、特定のタイプのエラーで文が終了およびロールバックしなくなります。そのかわり、それぞれのエラーがロギングされ、文が続行します。その後、エラーの発生した行に対して修正アクションを取ることができます。

DMLエラー・ロギングは、INSERT、UPDATE、MERGEおよびDELETE文で機能します。ここでは、特にINSERT文について説明します。

DMLエラー・ロギングを使用してデータを挿入するには:

1. エラー・ロギング表を作成します。(オプション)

   表は、手動で作成するか、またはDBMS_ERRLOGパッケージを使用して自動的に作成できます。詳細は、「エラー・ロギング表の作成」を参照してください。

2. エラー・ロギング句を指定してINSERT文を実行します。この句は、次のように動作します。
   - 必要に応じて、作成したエラー・ロギング表を参照します。エラー・ロギング表名を指定しない場合、データベースは、デフォルトの名前のエラー・ロギング表に記録します。デフォルトのエラー・ロギング表名は、ERR$_の後に、挿入対象になる表名の最初の25文字を付加した名前です。
   - オプションで、エラーの原因となった文を識別するためにエラー・ログに追加されるタグ(丸括弧でくくられた数値または文字列リテラル)が含まれます。タグを省略すると、NULL値が使用されます。
   - 必要に応じて、REJECT LIMIT副次句を指定します。

     この副次句は、INSERT文が終了およびロールバックされるまでに検出が許可されるエラーの最大数を示します。UNLIMITEDも指定できます。デフォルトの拒否の上限は0(ゼロ)で、これは最初のエラーが検出されると、エラーがロギングされて文がロールバックされることを表します。パラレルDML操作では、拒否の上限は各パラレル実行サーバーに対して適用されます。

   ノート:

   拒否の上限を超えて文がロールバックされた場合、エラー・ロギング表には、その時点までに記録されたログ・エントリが保持されます。

   エラー・ロギング句の構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

3. エラー・ロギング表を問い合せ、エラーの原因となった行に対する訂正処理を実行します。

   エラー・ロギング表の構造は、後述の「エラー・ロギング表の書式」を参照してください。

例20-9 DMLエラー・ロギングを使用したデータ挿入

次の文は、DW_EMPL表に行を挿入し、ERR_EMPL表にエラーを記録します。タグ'daily_load'は、各ログ・エントリにコピーされます。エラー数が25を超えると、文が終了してロールバックされます。

```
INSERT INTO dw_empl
  SELECT employee_id, first_name, last_name, hire_date, salary, department_id
  FROM employees
  WHERE hire_date > sysdate - 7
  LOG ERRORS INTO err_empl ('daily_load') REJECT LIMIT 25
```

他の例は、『Oracle Database SQL言語リファレンス』および『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

親トピック: DMLエラー・ロギングを使用したバルクINSERT失敗の回避

20.4.4.2 エラー・ロギング表の書式

エラー・ロギング表には固有の書式があります。

エラー・ロギング表は、次の2つの部分で構成されます。

- エラーを説明する一連の必須列。たとえば、1列は、Oracleエラー番号を含みます。

  表20-4に、これらのエラーを説明する列を示します。

- エラーの原因となった行のデータが格納される一連のオプション列。列名は、挿入対象の表(DML表)の列名に対応しています。

  エラー・ロギング表のこの部分の列数は、0(ゼロ)、1または複数(最大でDML表の列数)の場合があります。DML表の列と同じ名前の列がエラー・ロギング表に存在する場合は、対応するデータが、障害のある挿入予定の行から、このエラー・ロギング表の列に書き込まれます。DML表の列に対応する列がエラー・ロギング表にない場合、その列は記録されません。エラー・ロギング表に、DML表の列と一致しない名前の列がある場合、その列は無視されます。

  型変換エラーが発生する場合があるため、エラー・ロギング表のオプション列のデータ型は、データの消失または変換エラーなしで値を取得できる型である必要があります。(オプションのログの列がDML表の列と同じ型だった場合、問題の発生したデータをログに取得すると、エラーの原因となった同じデータ変換の問題が発生する可能性があります。)データベースは、変換エラーの原因となったデータについて、可能なかぎり意味のある値をロギングしようとします。値を導出できなかった場合、NULLが列にロギングされます。エラー・ロギング表への挿入時にエラーが発生すると、文が終了します。

  表20-5に、DML表の各データ型について、使用を推奨するエラー・ロギング表の列のデータ型を示します。DBMS_ERRLOGパッケージを使用してエラー・ロギングを自動的に作成すると、これらの推奨データ型が使用されます。

表20-4 エラーを説明する必須列

列名	データ型	説明
ORA_ERR_NUMBER$	NUMBER	Oracle エラー番号
ORA_ERR_MESG$	VARCHAR2(2000)	Oracle エラー・メッセージのテキスト
ORA_ERR_ROWID$	ROWID	エラーとなった行の ROWID(更新および削除の場合)

列名	データ型	説明
ORA_ERR_OPTYP$	VARCHAR2(2)	操作の種類: 挿入(I)、更新(U)、削除(D) ノート: MERGE 操作の UPDATE 句および INSERT 句のエラーは、U および I の値で区別されます。
ORA_ERR_TAG$	VARCHAR2(2000)	ユーザーがエラー・ロギング句に指定したタグの値

表20-5 エラー・ロギング表の列のデータ型

DML表の列の型	エラー・ロギング表の列の型	ノート
NUMBER	VARCHAR2(4000)	変換エラーを記録できます。
CHAR/VARCHAR2(n)	VARCHAR2(4000)	情報の消失なしで値を記録します。
NCHAR/NVARCHAR2(n)	NVARCHAR2(4000)	情報の消失なしで値を記録します。
DATE/TIMESTAMP	VARCHAR2(4000)	情報の消失なしで値を記録します。デフォルトの日時書式マスクを使用して文字書式に変換します。
RAW	RAW(2000)	情報の消失なしで値を記録します。
ROWID	UROWID	ROWID 型を記録します。
LONG/LOB		サポートされていません
ユーザー定義型		サポートされていません

親トピック: DMLエラー・ロギングを使用したバルクINSERT失敗の回避

20.4.4.3 エラー・ロギング表の作成

エラー・ロギング表は手動で作成できます。または、PL/SQLパッケージを使用して自動的に作成できます。

- エラー・ロギング表の自動作成
  エラー・ロギング表を自動作成するには、DBMS_ERRLOGパッケージを使用します。
- 手動によるエラー・ロギング表の作成
  エラー・ロギング表を手動で作成するには標準DDLを使用します。

親トピック: DMLエラー・ロギングを使用したバルクINSERT失敗の回避

20.4.4.3.1 エラー・ロギング表の自動作成

エラー・ロギング表を自動作成するには、DBMS_ERRLOGパッケージを使用します。

CREATE_ERROR_LOGプロシージャは、エラーを説明するための必須列および指定されたDML表の列をすべて備えたエラー・ロギング表を作成し、表20-5に示したデータ型マッピングを実行します。

次の文は、前述の例で使用したエラー・ロギング表を作成します。

```
EXECUTE DBMS_ERRLOG.CREATE_ERROR_LOG('DW_EMPL', 'ERR_EMPL');
```

DBMS_ERRLOGの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: エラー・ロギング表の作成

20.4.4.3.2 手動によるエラー・ロギング表の作成

エラー・ロギング表を手動で作成するには、標準DDLを使用します。

表の構造の要件は、「エラー・ロギング表の書式」を参照してください。エラーを説明するための必須列はすべて挿入する必要があります。列は順不同にできますが、必須列は表の最初の方の列に指定する必要があります。

親トピック: エラー・ロギング表の作成

20.4.4.4 エラー・ロギングの制限事項と注意

一部のエラーはエラー・ロギング表に記録されません。

Oracle Databaseは、DML操作中に次のエラーを記録します。

- 列の値が大きすぎる場合
- 制約(NOT NULL制約、一意制約、参照制約、CHECK制約)違反の場合
- トリガー実行時にエラーが発生した場合
- 副問合せの列と表内の対応する列との間の型変換でエラーが発生した場合
- パーティション・マッピング・エラー
- 特定のMERGE操作エラー(ORA-30926: MERGE操作の安定したセット行を取得できません)

一部のエラーは記録されずに、DML操作の終了およびロールバックが実施されます。これらのエラーの一覧とDMLロギングの他の制約は、『Oracle Database SQL言語リファレンス』のINSERTに関する項でerror_logging_clauseの説明を参照してください。

- 領域の要件
  DMLエラー・ロギングを使用するには、その前に領域の要件について考慮する必要があります。挿入する表の領域のみでなく、エラー・ロギング表の領域も必要です。
- セキュリティ
  DMLエラー・ロギングを指定したINSERT文を発行するユーザーには、エラー・ロギング表に対するINSERT権限が必要です。

親トピック: DMLエラー・ロギングを使用したバルクINSERT失敗の回避

20.4.4.4.1 領域に関する考慮事項

DMLエラー・ロギングを使用するには、その前に領域の要件について考慮する必要があります。挿入する表の領域のみでなく、エラー・ロギング表の領域も必要です。

親トピック: エラー・ロギングの制限事項と注意

20.4.4.4.2 セキュリティ

DMLエラー・ロギングを指定したINSERT文を発行するユーザーには、エラー・ロギング表に対するINSERT権限が必要です。

関連項目:

DMLエラー・ロギングの例は、『Oracle Database SQL言語リファレンス』および『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

親トピック: エラー・ロギングの制限事項と注意

20.5 バルク更新のパフォーマンス最適化

DBMS_REDEFINITIONパッケージのEXECUTE_UPDATEプロシージャを使用して、表のバルク更新のパフォーマンスを最適化できます。REDOログに更新が記録されないため、パフォーマンスが最適化されます。

EXECUTE_UPDATEプロシージャは、仮表、マテリアライズド・ビュー、マテリアライズド・ビュー・ログなどのオンライン表再定義のコンポーネントを自動的に使用して、表のバルク更新を最適化します。また、EXECUTE_UPDATEプロシージャは影響を受ける行のフラグメンテーションも除去して、アトミックな更新を保証します。バルク更新でエラーが発生した場合は、ABORT_UPDATEプロシージャを使用して、EXECUTE_UPDATEプロシージャで行った変更を元に戻すことができます。

EXECUTE_UPDATEプロシージャには、次の制限が適用されます。

- オンライン表再定義に適用されるすべての制限が、EXECUTE_UPDATEプロシージャとABORT_UPDATEプロシージャにも適用されます。
- 1つの表に複数のEXECUTE_UPDATEプロシージャを同時には実行できません。
- EXECUTE_UPDATEプロシージャが表に対して実行されているときに、別のセッションからその表に対してDMLによる変更を行わないでください。それらのDMLによる変更は、EXECUTE_UPDATEプロシージャが完了したときに失われます。
- UPDATE文の実行時に起動するトリガーを表に設定することはできません。
- EXECUTE_UPDATEプロシージャに渡されるUPDATE文には、拡張パーティション名を持つ表を指定できません。
- ユーザー定義型(VARRAY、REFおよびネストした表)を表に使用できません。
- Oracle提供のタイプ(ANYTYPE、ANYDATASET、URIタイプ、SDO_TOPO_GEOMETRY、SDO_GEORASTERおよびExpression)を表に使用できません。
- 次のタイプの列を表に使用できません: 非表示列、仮想列、未使用列、疑似列またはアイデンティティ列。
- オブジェクト表は表として指定できません。
- 表に仮想プライベート・データベース(VPD)ポリシーを指定することはできません。
- 表にチェック制約を指定できません。
- 表で行アーカイブを有効にできません。

バルク更新のパフォーマンスを最適化するには:

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。
   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があ

ります。

2. EXECUTE_UPDATEプロシージャを実行して、バルク更新を実行するSQL文を指定します。
   エラーが発生した場合は、ABORT_UPDATEプロシージャを使用して、EXECUTE_UPDATEプロシージャで行った変更を元に戻すことができます。
3. 更新したデータをバックアップします。
   EXECUTE_UPDATEプロシージャでは変更がREDOログに記録されないため、更新された表を含むデータベースまたは表領域をバックアップしないかぎり、リカバリは実行できません。

例20-10 製品データに対する最適化されたバルク更新の実行

この例では、oe.order_items表のバルク更新を実行しています。具体的には、product_idが3106である各注文項目のunit_priceに45を設定しています。バルク更新が失敗した場合は、ABORT_UPDATEプロシージャを使用して、EXECUTE_UPDATEプロシージャで実行したすべての変更をキャンセルし、データをプロシージャ実行前の状態に戻すことができます。

```
DECLARE
   update_stmt VARCHAR2(300) := 'UPDATE oe.order_items SET unit_price = 45
                                    WHERE product_id = 3106';
BEGIN
   DBMS_REDEFINITION.EXECUTE_UPDATE(update_stmt);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('No Data found for SELECT');
    DBMS_REDEFINITION.ABORT_UPDATE(update_stmt);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ('Reason for failure is'|| SQLERRM);
       IF (SQLCODE = 100)
       THEN
          DBMS_REDEFINITION.ABORT_UPDATE(update_stmt);
       END IF;
END;
/
```

親トピック: 表の管理

20.6 表に関する統計の自動収集

PL/SQLパッケージDBMS_STATSを使用すると、コストベースの最適化に関する統計を生成および管理できます。このパッケージを使用して、統計の収集、変更、表示、エクスポート、インポートおよび削除ができます。また、すでに収集した統計を識別または命名する際も、このパッケージを使用できます。

以前は、DBMS_STATSを使用可能にし、CREATE(またはALTER) TABLE文でMONITORINGキーワードを指定して、表の統計を自動的に収集していました。MONITORINGおよびNOMONITORINGキーワードは非推奨になり、統計は自動的に収集されます。これらのキーワードを指定しても無視されます。

監視では、統計が最後に収集された時点以降表に対して実行されたINSERT、UPDATEおよびDELETEの概数が追跡されます。影響を受ける行数に関する情報は、SMONが周期的に(およそ3時間ごとに)データをデータ・ディクショナリに取り込むまで、システム・グローバル領域(SGA)に保持されます。このデータ・ディクショナリの情報は、DBA_TAB_MODIFICATIONS,ALL_TAB_MODIFICATIONSまたはUSER_TAB_MODIFICATIONSビューで表示可能です。データベースはこれらのビューを使用して、失効した統計を持つ表を識別します。

STATISTICS_LEVEL初期化パラメータのデフォルトはTYPICALで、これは自動統計収集機能を有効にします。自動統計収集とDBMS_STATSパッケージによって、オプティマイザは正確な実行計画を生成できます。STATISTICS_LEVEL初期化パラメータをBASICに設定すると、Oracle Database機能で必要とされる多くの重要な統計が収集されません。すべての表

の監視を使用禁止にするには、STATISTICS_LEVEL初期化パラメータをBASICに設定します。自動統計収集とDBMS_STATSパッケージによって、オプティマイザは正確な実行計画を生成できます。

関連項目:

- STATISTICS_LEVEL初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- オプティマイザ統計の管理の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- DBMS_STATSパッケージの使用方法の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- スケジューラを使用して統計を自動的に収集する方法は、「自動化メンテナンス・タスクについて」を参照してください

親トピック: 表の管理

20.7 表の変更

表を変更するにはALTER TABLE文を使用します。表を変更するには、その表が自分のスキーマに含まれているか、その表のALTERオブジェクト権限またはALTER ANY TABLEシステム権限のいずれかを持っている必要があります。

ノート:

表を変更する前に、表を変更した結果についてよく理解しておいてください。これらの結果については、『Oracle Database SQL 言語リファレンス』の ALTER TABLE 句の説明を参照してください。

パッケージのビュー、マテリアライズド・ビュー、トリガー、ドメイン索引、ファンクション索引、CHECK 制約、ファンクション、プロシージャが実表に依存する場合は、その実表または列を変更すると依存するオブジェクトに影響する可能性があります。データベースによる依存性管理の詳細は、「オブジェクト依存性の管理」を参照してください。

- ALTER TABLE文を使用する理由
  ALTER TABLE文を使用する理由がいくつかあります。
- 表の物理属性の変更
  表の物理属性を変更する際には、いくつかの考慮事項があります。
- 新規セグメントまたは表領域への表の移動
  新規セグメントまたは表領域に表を移動して、圧縮を有効にしたり、データ・メンテナンスを実行できます。
- 表の記憶域の手動割当て
  Oracle Databaseは、必要に応じて表のデータ・セグメントに追加のエクステントを動的に割り当てます。ただし、表に追加のエクステントを明示的に割り当てることもできます。たとえば、Oracle Real Application Clusters環境で、表のエクステントを特定のインスタンスに対して明示的に割り当てることが可能です。
- 既存の列定義の変更
  既存の列定義を変更するには、ALTER TABLE...MODIFY文を使用します。列のデータ型、デフォルト値、列制約、列の式(仮想列の場合)、列の暗号化および可視/不可視プロパティは変更できます。
- 表の列の追加
  既存の表に列を追加するには、ALTER TABLE...ADD文を使用します。
- 表の列名の変更

Oracle Databaseでは、表の既存の列の名前を変更できます。列名を変更するには、ALTER TABLE文のRENAME COLUMN句を使用します。

- 表の列の削除
  索引構成表などの表から、不要になった列を削除できます。これにより、データベースの領域を解放でき、データをエクスポート/インポートしてから索引と制約を再作成する必要がなくなります。
- 表を読取り専用モードにする方法
  表を読取り専用モードにするには、ALTER TABLE...READ ONLY文を使用し、表を読取り/書込みモードに戻すには、ALTER TABLE...READ WRITE文を使用します。

親トピック: 表の管理

20.7.1 ALTER TABLE文を使用する理由

ALTER TABLE文を使用する理由がいくつかあります。

ALTER TABLE文は、表に影響を与える次の処理を実行するために使用できます。

- 物理的な特性(INITRANSまたは記憶域パラメータ)を変更する場合
- 表を新しいセグメントまたは表領域に移動する場合
- 明示的にエクステントを割り当てるか、未使用領域の割当てを解除する場合
- 列を追加、削除または名前変更する場合、あるいは既存の列定義(データ型、長さ、デフォルト値、NOT NULL整合性制約、列の式(仮想列の場合)および暗号化プロパティ)を変更する場合
- 表のロギング属性を変更する場合
- CACHE/NOCACHE属性を変更する場合
- 表に関連付けられている整合性制約を追加、変更または削除する場合
- 表に関連付けられている整合性制約またはトリガーを使用可能にするか、使用禁止にする場合
- 表の並列度を変更する場合
- 表の名前を変更する場合
- 表を読取り専用モードにする場合および読取り/書込みモードに戻す場合
- 索引構成表の特性を追加または変更する場合
- 外部表の特性を変更する場合
- LOB列を追加または変更する場合
- オブジェクト型、ネストした表またはVARRAYの列を追加または変更する場合
- 表パーティションを変更する場合

  Oracle Database 12cからは、パーティションの分割操作やパーティションのマージ操作などの操作を、一度に3つ以上のパーティションまたはサブパーティションに対して実行できます。詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

これらの多くのタスクについて、次の各項で説明します。

親トピック: 表の変更

20.7.2 表の物理属性の変更

表の物理属性を変更する際には、いくつかの考慮事項があります。

表のトランザクション・エントリ設定INITRANSを変更する場合、INITRANSの新しい設定は、その後表に割り当てられるデータ・ブロックにのみ適用されます。

記憶域パラメータINITIALとMINEXTENTSは変更できません。他の記憶域パラメータ(NEXT、PCTINCREASEなど)の新しい設定はすべて、その後に表に割り当てられるエクステントにのみ影響します。次に割り当てられるエクステントのサイズはNEXTおよびPCTINCREASEの現在の値によって決定され、これらのパラメータの以前の値には基づきません。

関連項目:

物理属性句と記憶域句の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表の変更

20.7.3 新規セグメントまたは表領域への表の移動

新規セグメントまたは表領域に表を移動して、圧縮を有効にしたり、データ・メンテナンスを実行できます。

- 新規セグメントまたは表領域への表の移動について
  ターゲット表領域に適切な割当てを持っている場合は、ALTER TABLE...MOVE [PARTITION|SUBPARTITION]文を使用して、表、パーティションまたはサブパーティションを移動して、圧縮や表領域などの物理的な記憶域属性を変更できます。
- 表の移動
  表を新しいセグメントまたは表領域に移動するには、ALTER TABLE...MOVE文を使用します。
- 表パーティションまたはサブパーティションのオンラインでの移動
  表パーティションまたはサブパーティションを移動するには、それぞれALTER TABLE...MOVE PARTITION文またはALTER TABLE...MOVE SUBPARTITION文を使用します。

親トピック: 表の変更

20.7.3.1 新規セグメントまたは表領域への表の移動について

ターゲット表領域に適切な割当てを持っている場合は、ALTER TABLE...MOVE [PARTITION|SUBPARTITION]文を使用して、表、パーティションまたはサブパーティションを移動して、圧縮や表領域などの物理的な記憶域属性を変更できます。

ALTER TABLE...MOVE文では、ONLINEキーワードがサポートされており、移動中の表、パーティションまたはサブパーティションに対してデータ操作言語(DML)による操作を中断なく実行できます。以下の文を使用して、表、パーティションまたはサブパーティションをオンラインで移動できます。

- ALTER TABLE ... MOVE ... ONLINE
- ALTER TABLE ... MOVE PARTITION ... ONLINE
- ALTER TABLE ... MOVE SUBPARTITION ... ONLINE

表を移動すると、表の行のROWIDが変わります。ONLINEキーワードおよびUPDATE INDEXES句を指定して表を移動する場合、移動操作中も索引が使用可能になります。UPDATE INDEXES句を指定し、ONLINEキーワードを指定しない場合は、移動操作完了後すぐに索引が使用可能になります。UPDATE INDEXES句で変更できるのは、表のグローバル索引の記憶域プロパティ、または表のグローバル・パーティション索引の索引パーティションの記憶域プロパティのみです。UPDATE INDEXES句を指定しない場合、rowidを変更すると、表の索引にUNUSABLEのマークが付き、これらの索引を使用して表に

アクセスするDMLに対しては、ORA-01502エラーが返されます。この場合、表の索引を削除または再作成する必要があります。

移動操作により表の統計は無効になるため、表を移動した後に新しい統計を収集する必要があります。

表にLOB列が含まれている場合は、この文を使用して、明示的に指定したLOBデータおよび(表に関連付けられた)LOB索引セグメントを表とともに移動できます。特に指定しない場合、デフォルトではLOBデータとLOB索引セグメントは移動されません。

親トピック: 新規セグメントまたは表領域への表の移動

20.7.3.2 表の移動

表を新しいセグメントまたは表領域に移動するには、ALTER TABLE...MOVE文を使用します。

この文にONLINEキーワードを指定すると、移動中の表に対してデータ操作言語(DML)による操作を中断なく実行できます。ONLINEキーワードを指定しない場合は、移動中は表のデータに対して同時DML操作を実行できません。

表を移動するには:

1. SQL*Plusで、表の変更に必要な権限を持つユーザーとして接続します。

   表の変更に必要な権限の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

2. ALTER TABLE ... MOVE文を実行します。

例20-11 オンライン・モードによる新規表領域への表の移動

次の文は、新しい記憶域パラメータを指定して、hr.jobs表をオンラインで新しいセグメントおよび表領域に移動します。ONLINEキーワードは、移動中も表に対してDML操作を中断なく実行できることを意味します。hr_tbs表領域が存在している必要があります。

```
ALTER TABLE hr.jobs MOVE ONLINE
      STORAGE ( INITIAL 20K
                NEXT 40K
                MINEXTENTS 2
                MAXEXTENTS 20
                PCTINCREASE 0 )
  TABLESPACE hr_tbs;
```

Example 20-12 表の移動および表の索引の更新

次の文で、表とその索引を作成したとします。

```
CREATE TABLE dept_exp (
     DEPTNO NUMBER (2) NOT NULL,
     DNAME VARCHAR2 (14),
     LOC VARCHAR2 (13))
   TABLESPACE tbs_1;
CREATE INDEX i1_deptno ON dept_exp(deptno) TABLESPACE tbs_1;
CREATE INDEX i2_dname ON dept_exp(dname) TABLESPACE tbs_1;
```

次の文で、表を新しい表領域(tbs_2)に移動し、表を圧縮します。これは索引i2_dbnameも表領域tbs_2に移動し、移動操作後にi1_deptno索引とi2_dname索引の両方が使用可能になるように指定します。

```
ALTER TABLE dept_exp MOVE
   COMPRESS TABLESPACE tbs_2
   UPDATE INDEXES
      (i1_deptno TABLESPACE tbs_1,
       i2_dname TABLESPACE tbs_2);
```

この文にはONLINEキーワードは指定されていません。ただし、移動操作中も表に対してDML操作を中断なく実行できる必要

がある場合、または移動操作中も索引を使用できる必要がある場合、ONLINEキーワードはサポートされます。

これらの文を実行する前に、tbs_1およびtbs_2表領域が存在している必要があります。

親トピック: 新規セグメントまたは表領域への表の移動

20.7.3.3 表パーティションまたはサブパーティションのオンラインでの移動

表パーティションまたはサブパーティションを移動するには、それぞれALTER TABLE...MOVE PARTITION文またはALTER TABLE...MOVE SUBPARTITION文を使用します。

これらの文のいずれかとともにONLINEキーワードを使用すると、移動中のパーティションまたはサブパーティションに対してDML操作を中断なく実行し続けることができます。ONLINEキーワードを含めない場合、移動操作が完了するまで、パーティションまたはサブパーティション内のデータに対するDML操作は許可されません。

UPDATE INDEXES句を含めると、これらの文によって、移動中にローカル索引とグローバル索引の両方がメンテナンスされます。このため、これらの文とともにONLINEキーワードを使用すると、移動後にパーティションのパフォーマンスを回復するために、グローバル索引をメンテナンスして手動で索引を再構築する時間を省くことができます。

表パーティションおよびサブパーティションの移動には、いくつかの制限事項が適用されます。これらの制限事項については、『Oracle Database SQL言語リファレンス』を参照してください。

表パーティションまたはサブパーティションをオンラインで移動するには:

1. SQL*Plusで、表の変更、およびパーティションまたはサブパーティションの移動に必要な権限のあるユーザーとして接続します。

   必要な権限の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. ALTER TABLE ... MOVE PARTITIONまたはALTER TABLE ... MOVE SUBPARTITION文を実行します。

例20-13 新規セグメントへの表パーティションの移動

次の文では、sh.sales表のsales_q4_2003パーティションを高度な行圧縮および索引メンテナンスが組み込まれた新しいセグメントに移動します。

```
ALTER TABLE sales MOVE PARTITION sales_q4_2003
  ROW STORE COMPRESS ADVANCED UPDATE INDEXES ONLINE;
```

関連項目:

- Oracle Database VLDBおよびパーティショニング・ガイド
- Oracle Database SQL言語リファレンス

親トピック: 新規セグメントまたは表領域への表の移動

20.7.4 表の記憶域の手動割当て

Oracle Databaseは、必要に応じて表のデータ・セグメントに追加のエクステントを動的に割り当てます。ただし、表に追加のエクステントを明示的に割り当てることもできます。たとえば、Oracle Real Application Clusters環境で、表のエクステントを特定のインスタンスに対して明示的に割り当てることが可能です。

表に新しいエクステントを割り当てるには、ALTER TABLE...ALLOCATE EXTENT文を使用します。

また、ALTER TABLE文のDEALLOCATE UNUSED句を使用して、未使用領域の割当てを明示的に解除することもできます。詳細は、「未使用領域の再利用」を参照してください。

親トピック: 表の変更

20.7.5 既存の列定義の変更

既存の列定義を変更するには、ALTER TABLE...MODIFY文を使用します。列のデータ型、デフォルト値、列制約、列の式(仮想列の場合)、列の暗号化および可視/不可視プロパティは変更できます。

既存のデータがすべて新しい長さを満たしている場合は、既存の列の長さを拡張または縮小できます。Oracle Database 12cからは、VARCHAR2、NVARCHAR2およびRAWデータ型の最大サイズの32767バイトを指定できます。このリリースより前は、VARCHAR2およびNVARCHAR2データ型の最大サイズは4000バイト、RAWデータ型の最大サイズは2000バイトでした。拡張データ型を使用するには、MAX_STRING_SIZE初期化パラメータをEXTENDEDに設定します。

列は、バイト・セマンティクスからCHARセマンティクスに、あるいはその逆に変更できます。空でないCHAR列の長さを縮小するには、初期化パラメータBLANK_TRIMMING=TRUEを設定する必要があります。

データ型CHARの列長を拡張するために表を変更している場合、特に表の行数が多い場合は、この操作は時間がかかり、さらに相当な追加記憶域を必要とする可能性があります。これは、各行のCHAR値に空白を埋めて、新しい列長に合わせる必要があるためです。

列の可視/不可視プロパティを変更する場合、同じSQL文に他の列変更オプションを含めることはできません。

例20-14 4000バイトより大きいサイズへの列の長さの変更

この例では、oe.product_information表のproduct_description列の長さを32767バイトに変更します。

```
ALTER TABLE oe.product_information MODIFY(product_description VARCHAR2(32767));
```

関連項目:

- 表の列の変更の詳細およびその他の制限事項は、『Oracle Database SQL言語リファレンス』を参照してください。
- MAX_STRING_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 表の変更

20.7.6 表の列の追加

既存の表に列を追加するには、ALTER TABLE...ADD文を使用します。

次の文は、hr.admin_emp表を変更して新しい列bonusを追加します。

```
ALTER TABLE hr.admin_emp
      ADD (bonus NUMBER (7,2));
```

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 表の作成および変更で参照して実行してください。

表に新しい列を追加すると、DEFAULT句を指定しないかぎり、その列は最初はNULLです。一部の表タイプのNULL値可能列にDEFAULT句を指定すると、デフォルト値はメタデータとして格納されますが、列自体にはデータは移入されません。ただし、デフォルト値が結果セットに戻されるように、新しい列を指定する後続の問合せは再書込みされます。この動作によって、操作のリソース使用率と記憶域要件が最適化されます。

NOT NULL制約付きの列を追加できるのは、表に行がまったく含まれていない場合、またはデフォルト値を指定する場合のみです。

ノート:

- 表での基本表圧縮を有効化する場合は、デフォルト値を指定しない場合にのみ、列を追加できます。
- 表での高度な行圧縮を有効化する場合は、デフォルト値を指定してもしなくても、その表に列を追加できます。
- 新しい列が仮想列の場合、値は列の式によって決定されます。(仮想列の値は問い合せられたときにのみ計算される点に注意してください。)

関連項目:

- 表の列の追加に関するルールおよび制限事項は、『Oracle Database SQL言語リファレンス』を参照してください。
- 「表圧縮の使用」
- 『Oracle Database概要』
- 仮想列の例については、「例: 表の作成」を参照してください

親トピック: 表の変更

20.7.7 表の列名の変更

Oracle Databaseでは、表の既存の列の名前を変更できます。列名を変更するには、ALTER TABLE文のRENAME COLUMN句を使用します。

新しい名前には、表の既存の列名と競合しない名前を指定する必要があります。RENAME COLUMN句とともに他の句は使用できません。

次の文は、hr.admin_emp表のcomm列の名前を変更します。

```
ALTER TABLE hr.admin_emp
      RENAME COLUMN comm TO commission;
```

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 表の作成および変更で参照して実行してください。

前述のように、表の列を変更すると、依存するオブジェクトが無効になる可能性があります。ただし、列名を変更すると、ファンク

ション索引とCHECK制約が引き続き有効になるように、関連するデータ・ディクショナリ表が更新されます。

また、Oracle Databaseでは列制約の名前も変更できます。この操作の説明は、「制約名の変更」を参照してください。

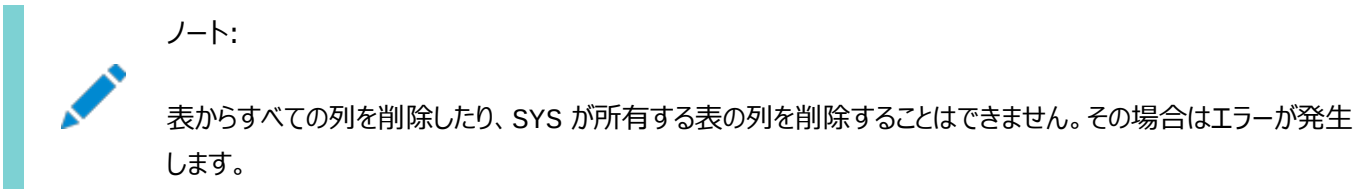
ノート:

ALTER TABLE の RENAME TO 句の構文は RENAME COLUMN 句に似ていますが、表自体の名前変更に使用します。

親トピック: 表の変更

20.7.8 表の列の削除

索引構成表などの表から、不要になった列を削除できます。これにより、データベースの領域を解放でき、データをエクスポート/インポートしてから索引と制約を再作成する必要がなくなります。



ノート:

表からすべての列を削除したり、SYS が所有する表の列を削除することはできません。その場合はエラーが発生します。

- 表から列を削除する方法
  ALTER TABLE...DROP COLUMN文を発行すると、列記述子およびターゲット列に関連付けられているデータが表の各行から削除されます。1つの文で複数の列を削除できます。
- 列に未使用のマークを付ける方法
  大きい表のすべての行から列データを削除する際に所要時間が重要な場合は、ALTER TABLE...SET UNUSED文を使用できます。
- 未使用列の削除
  未使用の列に対して実行できるアクションはALTER TABLE...DROP UNUSED COLUMNS文のみです。表から未使用の列を物理的に削除し、ディスク領域を再生します。
- 圧縮表の列の削除
  表での高度な行圧縮を有効化する場合は、表の列を削除できます。基本表圧縮のみを有効化する場合は、表の列を削除できません。

関連項目:

表からの列の削除に関するその他の制限事項およびオプションの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 表の変更

20.7.8.1 表から列を削除する方法

ALTER TABLE...DROP COLUMN文を発行すると、列記述子およびターゲット列に関連付けられているデータが表の各行から削除されます。1つの文で複数の列を削除できます。

次の文は、hr.admin_emp表から列を削除する操作の例を示しています。最初の文は、sal列のみを削除します。

```
ALTER TABLE hr.admin_emp DROP COLUMN sal;
```

次の文は、bonus列とcomm列を両方とも削除します。

```
ALTER TABLE hr.admin_emp DROP (bonus, commission);
```

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 表の作成および変更で参照して実行してください。

親トピック: 表の列の削除

20.7.8.2 列に未使用マークを付ける方法

大きい表のすべての行から列データを削除する際に所要時間が重要な場合は、ALTER TABLE...SET UNUSED文を使用できます。

この文は1つ以上の列に未使用マークを付けますが、実際にターゲット列を削除したり該当列が占めるディスク領域をリストアすることはありません。ただし、未使用マークが付けられた列は、問合せやデータ・ディクショナリ・ビューに表示されなくなり、その名前が削除されて新しい列に再利用できるようになります。ほとんどの場合、その列に定義されている制約、索引および統計も削除されます。例外として、未使用とマークされたLOB列の内部索引は削除されません。

hiredate列とmgr列に未使用マークを付けるには、次の文を実行します。

```
ALTER TABLE hr.admin_emp SET UNUSED (hiredate, mgr);
```

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 表の作成および変更で参照して実行してください。

後でALTER TABLE...DROP UNUSED COLUMNS文を発行し、未使用マークが付いている列を削除できます。表の特定列の明示的な削除文を発行すると、未使用列もターゲット表から削除されます。

データ・ディクショナリ・ビューUSER_UNUSED_COL_TABS、ALL_UNUSED_COL_TABSまたはDBA_UNUSED_COL_TABSを使用すると、未使用の列を含むすべての表を表示できます。COUNTフィールドには、表の未使用の列数が表示されます。

```
SELECT * FROM DBA_UNUSED_COL_TABS;
OWNER                          TABLE_NAME                     COUNT
------------------------------ ------------------------------ -----
HR                             ADMIN_EMP                          2
```

外部表の場合は、SET UNUSED文がALTER TABLE DROP COLUMN文に透過的に変換されます。外部表はデータベース内でメタデータのみで構成されているため、DROP COLUMN文はSET UNUSED文の実行と同じことになります。

親トピック: 表の列の削除

20.7.8.3 未使用列の削除

未使用の列に対して実行できるアクションはALTER TABLE...DROP UNUSED COLUMNS文のみです。表から未使用の列を物理的に削除し、ディスク領域を再生します。

次のALTER TABLE文では、オプションの句CHECKPOINTが指定されています。この句を指定すると、指定した行数(この場合は250行)が処理された後に、チェックポイントが適用されます。チェックポイントによって、列削除操作中に累積されるUNDOログの量が減少し、UNDO領域が使い果たされるおそれがなくなります。

```
ALTER TABLE hr.admin_emp DROP UNUSED COLUMNS CHECKPOINT 250;
```

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 表の作成および変更で参照して実行してください。

親トピック: 表の列の削除

20.7.8.4 圧縮表の列の削除

表での高度な行圧縮を有効化する場合は、表の列を削除できます。基本表圧縮のみを有効化する場合は、表の列を削除できません。

関連項目:

「表圧縮の使用」

親トピック: 表の列の削除

20.7.9 表を読取り専用モードにする方法

表を読取り専用モードにするには、ALTER TABLE...READ ONLY文を使用し、表を読取り/書込みモードに戻すには、ALTER TABLE...READ WRITE文を使用します。

読取り専用モードが有効な表の例に、構成表があります。アプリケーションに含まれている構成表が、インストール後変更されず、ユーザーによる変更を禁止する必要がある場合は、アプリケーションのインストール・スクリプトによって、これらの表を読取り専用モードにできます。

表を読取り専用モードにするには、その表に対するALTER TABLE権限、またはALTER ANY TABLE権限が必要です。また、COMPATIBLE初期化パラメータが11.2.0以上に設定されている必要があります。

次の例は、SALES表を読取り専用モードにします。

```
ALTER TABLE SALES READ ONLY;
```

次の例は、表を読取り/書込みモードに戻します。

```
ALTER TABLE SALES READ WRITE;
```

表が読取り専用モードの場合、表データの変更操作は許可されません。表またはパーティションが読取り専用モードにされたら、その後の表に対するSELECT column_list ON table_name文では、常に同じデータ・セットが返される必要があります。

読取り専用表で許可されない操作は、次のとおりです。

- 読取り専用表または読取り専用パーティションに対するすべてのDML操作
- TRUNCATE TABLE

- SELECT FOR UPDATE
- ALTER TABLE RENAME/DROP COLUMN
- 読取り専用パーティションまたは読取り専用表のパーティションのDROP
- ALTER TABLE SET COLUMN UNUSED
- ALTER TABLE DROP/TRUNCATE/EXCHANGE (SUB)PARTITION
- 読取り専用表が関係している型に対するALTER TABLE UPGRADE INCLUDING DATAまたはALTER TYPE CASCADE INCLUDING TABLE DATA
- オンライン再定義
- FLASHBACK TABLE

読取り専用表で許可される操作は、次のとおりです。

- SELECT
- CREATE/ALTER/DROP INDEX
- ALTER TABLE ADD/MODIFY COLUMN
- ALTER TABLE ADD/MODIFY/DROP/ENABLE/DISABLE CONSTRAINT
- 物理的なプロパティ変更のためのALTER TABLE
- ALTER TABLE DROP UNUSED COLUMNS
- ALTER TABLE ADD/COALESCE/MERGE/MODIFY/MOVE/RENAME/SPLIT (SUB)PARTITION
- ALTER TABLE MOVE
- ALTER TABLE ENABLE ROW MOVEMENTおよびALTER TABLE SHRINK
- RENAME TABLEおよびALTER TABLE RENAME TO
- DROP TABLE
- ALTER TABLE DEALLOCATE UNUSED
- ALTER TABLE ADD/DROP SUPPLEMENTAL LOG

関連項目:

- ALTER TABLE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- 読取り専用パーティションの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: 表の変更

20.8 表のオンライン再定義

表の論理構造または物理構造を変更できます。

- 表のオンライン再定義について
  データベース・システムでは、問合せまたはDMLのパフォーマンスを改善したり、アプリケーションの変更に対応したり、記憶域を管理するために、表の構造を論理的または物理的に変更する必要が生じることがあります。
  DBMS_REDEFINITIONパッケージを使用して、オンラインで表を再定義できます。
- 表のオンライン再定義の機能

表のオンライン再定義により、表をオンラインにしたまま、いくつかの方法で表を変更できます。

- DBMS_REDEFINITIONパッケージに必要な権限
  パッケージのサブプログラムを実行するには、DBMS_REDEFINITIONパッケージの実行権限が必要です。DBMS_REDEFINITIONパッケージの実行権限は、EXECUTE_CATALOG_ROLEに付与されます。
- 表のオンライン再定義に関する制限事項
  表のオンライン再定義にはいくつかの制限事項が適用されます。
- REDEF_TABLEプロシージャを使用したオンライン再定義の実行
  DBMS_REDEFINITIONパッケージのREDEF_TABLEプロシージャを使用して、表の記憶域プロパティのオンライン再定義を実行できます。
- DBMS_REDEFINITIONの複数のプロシージャを使用した表のオンライン再定義
  DBMS_REDEFINITIONパッケージの複数のプロシージャを使用して、表のオンライン再定義を実行できます。
- 再定義プロセスの結果
  再定義プロセスにはいくつかの結果があります。
- 中間での同期化の実行
  元の表に対して多数のDML文を実行する場合は、再定義プロセス中に仮表を元の表と同期できます。
- 表のオンライン再定義中に依存マテリアライズド・ビューをリフレッシュする方法
  表のオンライン再定義中に、高速リフレッシュ可能な依存マテリアライズド・ビューをリフレッシュするには、REDEF_TABLEプロシージャまたはSTART_REDEF_TABLEプロシージャで、refresh_dep_mviewsパラメータをYに設定します。
- 表のオンライン再定義の進行状況の監視
  V$ONLINE_REDEFビューを問い合せて、表のオンライン再定義操作の進行状況を監視できます。
- 失敗後の表のオンライン再定義の再開
  表のオンライン再定義が失敗した場合は、DBA_REDEFINITION_STATUSビューでエラー情報および再開可能性に関する情報を確認できます。
- 表のオンライン再定義のロールバック
  表のオンライン再定義後のロールバックを有効にして、表を元の定義に戻し、表に対して行ったDML変更を保持できます。
- エラー後の表のオンライン再定義の終了およびクリーン・アップ
  オンライン再定義プロセスを終了できます。これにより、再定義プロセスに対応付けられた一時ログおよび一時表が削除されます。このプロシージャをコールした後は、仮表とその依存オブジェクトを削除できます。
- 1つ以上のパーティションのオンライン再定義
  表の1つ以上のパーティションのオンライン再定義を実行できます。これは、異なる表領域にパーティションを移動する際に、移動中でもパーティションに対してDMLを使用できるようにする場合などに便利です。
- 表のオンライン再定義の例
  例を使用して表のオンライン再定義を説明します。

親トピック: 表の管理

20.8.1 表のオンライン再定義について

データベース・システムでは、問合せまたはDMLのパフォーマンスを改善したり、アプリケーションの変更に対応したり、記憶域を管理するために、表の構造を論理的または物理的に変更する必要が生じることがあります。DBMS_REDEFINITIONパッケージを使用して、オンラインで表を再定義できます。

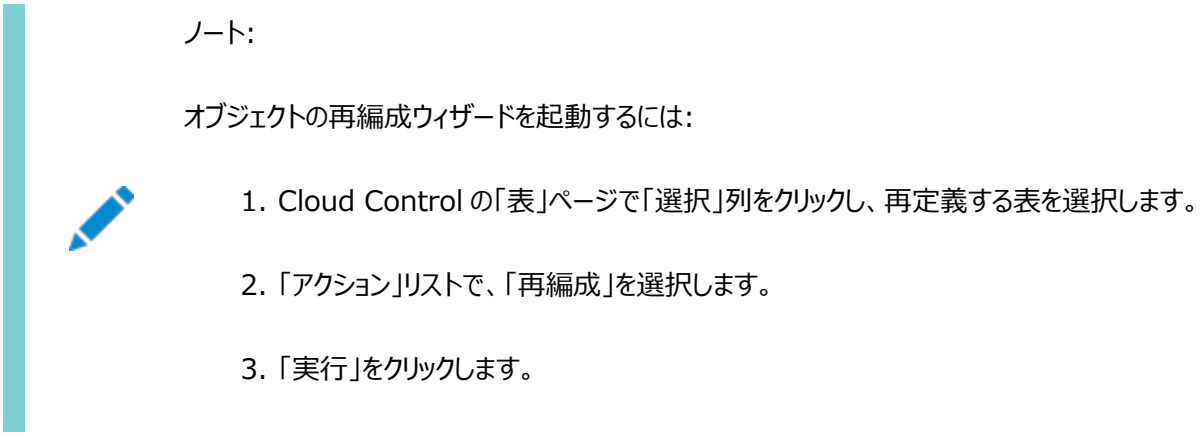
Oracle Databaseには、表の可用性に大きな影響を与えずに表の構造を変更できるメカニズムが用意されています。このメカ

ニズムは、表のオンライン再定義と呼ばれます。表のオンライン再定義では、表を再定義する従来の方法に比べて、可用性が大幅に向上します。

オンラインで表を再定義している間も、その再定義プロセスの大部分で、問合せおよびDMLを使用してその表にアクセスできます。通常、表が排他モードでロックされるのは、そのサイズや再定義の複雑さに関係なくわずかな間のみで、ユーザーに対しては完全に透過的です。ただし、再定義中に数多くの同時DML操作がある場合、表をロックできるようになる前に、より長く待機することが必要になることがあります。

表のオンライン再定義には、再定義の対象になる表が使用している領域とほぼ同等の空き領域が必要です。新しい列を追加する場合は、より多くの領域が必要になります。

表のオンライン再定義を実行するには、Oracle Enterprise Manager Cloud Control (Cloud Control)のオブジェクトの再編成ウィザードまたはDBMS_REDEFINITIONパッケージを使用します。



ノート:

オブジェクトの再編成ウィザードを起動するには:

1. Cloud Control の「表」ページで「選択」列をクリックし、再定義する表を選択します。
2. 「アクション」リストで、「再編成」を選択します。
3. 「実行」をクリックします。

関連項目:

DBMS_REDEFINITIONパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 表のオンライン再定義

20.8.2 表のオンライン再定義の機能

表のオンライン再定義により、表をオンラインにしたまま、いくつかの方法で表を変更できます。

表のオンライン再定義では、次のことが可能です。

- 表またはクラスタの記憶域パラメータの変更
- 異なる表領域への表またはクラスタの移動

ノート:

表を別の表領域に移動する際に、DML でその表を使用する必要性がない場合は、より簡単な ALTER TABLE MOVE コマンドを使用できます。「新規セグメントまたは表領域への表の移動」を参照してください。

- 表またはクラスタの1つ以上の列の追加、変更、または削除

- パーティション化サポートの追加または削除(非クラスタ化表のみ)
- パーティション構造の変更
- 同じスキーマ内の別の表領域へのパーティションの移動を含む、単一の表パーティションまたはサブパーティションの物理的なプロパティの変更

  Oracle Database 12cからは、表のオンライン再定義を使用しないで、パーティションまたはサブパーティションをオンラインで移動できます。移動中のパーティションまたはサブパーティションに対してDML操作を中断なく実行し続けることができます。「新規セグメントまたは表領域への表の移動」を参照してください。
- マテリアライズド・ビュー・ログまたはOracle Databaseアドバンスト・キューイングのキュー表の物理的なプロパティの変更

  ノート:

  DBMS_REDEFINITION パッケージの REDEF_TABLE プロシージャは、Oracle Database Advanced Queuing キュー表の物理プロパティの変更はサポートしません。
- パラレル問合せのサポートの追加
- 表またはクラスタの再作成による断片化の低減

  ノート:

  多くの場合、オンラインによるセグメントの縮小が断片化を削減する簡単な方法です。詳細は、「未使用領域の再利用について」を参照してください。
- 通常の表(ヒープ構成表)から索引構成表へ、または索引構成表から通常の表への編成の変更。
- リレーショナル表からオブジェクト列を持つ表へ、またはオブジェクト列を持つ表からリレーショナル表への変換。
- オブジェクト表からリレーショナル表またはオブジェクト列を持つ表へ、あるいはリレーショナル表またはオブジェクト列を持つ表からオブジェクト表への変換。
- 表、パーティション、索引キーまたはLOB列の圧縮、あるいはその圧縮タイプの変更
- BasicFiles LOB記憶域からSecureFiles LOB記憶域へ、またはSecureFiles LOB記憶域からBasicFiles LOB記憶域へのLOB列の変換
- 表のオンライン再定義後のロールバックを有効にして、表を元の定義に戻し、表に対して行ったDML変更を保持できます。
- 表のオンライン再定義中に、高速リフレッシュ可能な依存マテリアライズド・ビューをリフレッシュするには、REDEF_TABLEプロシージャまたはSTART_REDEF_TABLEプロシージャで、refresh_dep_mviewsパラメータをYに設定します。
- V$ONLINE_REDEFビューを問い合せて、表のオンライン再定義操作の進行状況を監視できます。
- 表のオンライン再定義が失敗したときは、多くの場合、失敗の原因となった問題を修正し、最後に停止したところからオンライン再定義プロセスを再開できます。

これらの使用例の2つ以上を1つの操作に組み合せることができます。例は、「表のオンライン再定義の例」の例8を参照してください。

親トピック: 表のオンライン再定義

20.8.3 DBMS_REDEFINITIONパッケージに必要な権限

DBMS_REDEFINITIONパッケージの実行権限は、パッケージ内のサブプログラムの実行に必要です。
DBMS_REDEFINITIONパッケージの実行権限は、EXECUTE_CATALOG_ROLEに付与されます。

さらに、パッケージを使用してユーザーのスキーマの表を再定義するには、ユーザーは次の権限を付与されている必要があります。

- CREATE TABLE
- CREATE MATERIALIZED VIEW

COPY_TABLE_DEPENDENTSプロシージャを実行するには、CREATE TRIGGER権限も必要です。

パッケージを使用して他のスキーマの表を再定義するには、ユーザーは次の権限を付与されている必要があります。

- CREATE ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE
- LOCK ANY TABLE
- SELECT ANY TABLE

他のスキーマの表でCOPY_TABLE_DEPENDENTSを実行するには、次の追加権限が必要です。

- CREATE ANY TRIGGER
- CREATE ANY INDEX

親トピック: 表のオンライン再定義

20.8.4 表のオンライン再定義に関する制限事項

表のオンライン再定義には、いくつかの制限が適用されます。

表のオンライン再定義には、次の制限が適用されます。

- 表を主キーまたは擬似主キー(すべてのコンポーネント列がNULLでない制約を持つ一意キーまたは制約)を使用して再定義する場合、再定義後の表でも同じ主キーまたは擬似主キー列を使用する必要があります。ROWIDを使用して再定義する表に、索引構成表を含めないでください。
- マテリアライズド・ビュー・ログが含まれている表を再定義した後に依存マテリアライズド・ビューをリフレッシュする場合は、完全リフレッシュを実行する必要があります。

  この制限には例外があります。表のオンライン再定義にREDEF_TABLEまたはSTART_REDEF_TABLEプロシージャを使用し、プロシージャでrefresh_dep_mviewsパラメータをYに設定した場合、増分リフレッシュのために構成された依存マテリアライズド・ビューが、表のオンライン再定義操作中にリフレッシュされます。
- n-wayマスター構成でレプリケートされた表の再定義は可能ですが、水平サブセット化(表内の行のサブセット)、垂直サブセット化(表内の列のサブセット)または列変換は使用できません。
- 索引構成表のオーバーフロー表は、個別にオンライン再定義できません。
- Flashbackデータ・アーカイブが有効になっている表は、オンラインで再定義できません。Flashbackデータ・アーカイブを仮表に対して有効にはできません。

- 複数のLONG列を保持している表は、オンラインで再定義できますが、これらの列は、CLOBに変換する必要があります。また、LONG RAW列は、BLOBに変換する必要があります。LOB列を持つ表は、オンライン再定義可能です。
- パラレル実行のためのリソースが十分なシステムで、仮表がパーティション化されていない環境では、次の場合にのみ、LONG列からLOB列への再定義をパラレルで実行できます。
    - 仮表へのLOB列の格納に使用するセグメントが、自動セグメント領域管理(ASSM)を使用できるローカル管理表領域に属している場合。
    - 単一のLONG列から単一のLOB列への簡単なマッピングで、仮表に存在するLOB列が1つのみの場合。

  仮表がパーティション化されている場合は、パラレル実行でパーティション化する通常の方法が適用されます。
- SYSおよびSYSTEMスキーマ内の表は、オンライン再定義できません。
- 一時表は再定義できません。
- 表内の行のサブセットは再定義できません。
- 仮表の列を元の表の列にマッピングするときに使用できるのは、値がすぐに決定される単純な式、順序およびSYSDATEのみです。たとえば、副問合せは使用できません。
- 新しい列を再定義の一部として追加しようとして、それらの列に列マッピングがない場合は、その再定義が完了するまでNOT NULLを宣言しないでください。
- 再定義しようとする表と仮表の間では参照制約を作成できません。
- 表の再定義は、NOLOGGINGモードでは実行できません。
- マテリアライズド・ビュー・ログおよびキュー表の場合、オンライン再定義は物理的なプロパティの変更に制限されます。水平サブセット化または垂直サブセット化が使用できず、列の変換もできません。列マッピング文字列に唯一有効な値はNULLです。
- 1つ以上のネストした表が含まれているパーティションに対するオンライン再定義は実行できません。
- VARRAYは、列マッピングにCAST演算子を使用してネストした表に変換できます。ただし、ネストした表をVARRAYに変換することはできません。
- DBMS_REDEFINITION.START_REDEF_TABLEプロシージャのcol_mappingパラメータ内の列に順序が含まれている場合、orderby_colsパラメータはNULLでなくてはなりません。
- 仮想プライベート・データベース(VPD)セキュリティ・ポリシーが適用された表では、copy_vpd_optパラメータをDBMS_REDEFINITION.CONS_VPD_AUTOとして指定した場合、次の制限が適用されます。
    - 元の表と仮表の間の列マッピング文字列は、NULLまたは'*'にする必要があります。
    - VPDポリシーを仮表に指定することはできません。

  「オンライン再定義時の仮想プライベート・データベース(VPD)ポリシーの処理」を参照してください。また、VPDポリシーの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- 参照パーティション化によって複数の表が関連付けられている場合、異なるDBMS_REDEFINITIONセッションにおいてそれらの表でオンライン再定義を同時に実行することはできません。

  参照パーティション化の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。
- オブジェクト表またはXMLType表のオンライン再定義は、他の表に再定義済の表を参照するREF列がある場合に、他の表のREFの参照先がない状態を引き起こす可能性があります。

REFの参照先がないことの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

- Oracle Label Security (OLS)を使用する表は、オンライン再定義できません。

  Oracle Label Security管理者ガイドを参照してください。

- ファイングレイン・アクセス制御が設定された表は、オンライン再定義できません。

- Oracle Real Application Securityを使用する表は、オンライン再定義できません。

  Oracle Databaseセキュリティ・ガイドを参照してください。

親トピック: 表のオンライン再定義

20.8.5 REDEF_TABLEプロシージャを使用したオンライン再定義の実行

DBMS_REDEFINITIONパッケージのREDEF_TABLEプロシージャを使用して、表の記憶域プロパティのオンライン再定義を実行できます。

REDEF_TABLEプロシージャを使用すると、次のプロパティを変更するときに、表の記憶域プロパティのオンライン再定義を1つのステップで実行できます。

- 表、パーティション、索引、LOB列などの表領域の変更
- 表、パーティション、索引キー、LOB列などの圧縮タイプの変更
- LOB列の場合、SECUREFILEまたはBASICFILE記憶域の変更

オンライン再定義操作がこれらの変更に限定されない場合は、複数のステップで表のオンライン再定義を実行する必要があります。これらのステップでは、DBMS_REDEFINITIONパッケージ内の複数のプロシージャ(CAN_REDEF_TABLE、START_REDEF_TABLE、COPY_TABLE_DEPENDENTSおよびFINISH_REDEF_TABLE)を呼び出します。

ノート:

REDEF_TABLE プロシージャを使用して表を再定義した場合は、表のオンライン再定義のロールバックはサポートされません。

関連項目:

- プロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください
- 「表のオンライン再定義の例」の例1
- 詳細は、「DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行」を参照してください

親トピック: 表のオンライン再定義

20.8.6 DBMS_REDEFINITIONの複数のプロシージャを使用した表のオンライン再定義

DBMS_REDEFINITIONパッケージの複数のプロシージャを使用して、表のオンライン再定義を実行できます。

- DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行
  DBMS_REDEFINITIONパッケージの複数のプロシージャを使用して、表のオンライン再定義を実行できます。

- 列マッピング文字列の作成
  引数としてSTART_REDEF_TABLEに渡す列マッピング文字列には、カンマで区切られた列マッピングのペアのリストが含まれています。
- オンライン再定義中の仮想プライベート・データベース(VPD)ポリシーの処理
  再定義する元の表にVPDポリシーが指定されている場合、START_REDEF_TABLEプロシージャでcopy_vpd_optパラメータを使用して、オンライン再定義中にこれらのポリシーを処理できます。
- 依存オブジェクトの自動作成
  仮表に対する依存オブジェクトを自動的に作成するには、COPY_TABLE_DEPENDENTSプロシージャを使用します。
- 依存オブジェクトの手動による作成
  SQL*PlusまたはCloud Controlで、仮表に対する依存オブジェクトを手動で作成する場合は、REGISTER_DEPENDENT_OBJECTプロシージャを使用して依存オブジェクトを登録する必要があります。依存オブジェクトを登録すると、再定義の完了プロセスで、依存オブジェクト名を再定義前の名前にリストアできます。

親トピック: 表のオンライン再定義

20.8.6.1 DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行

DBMS_REDEFINITIONパッケージの複数のプロシージャを使用して、表のオンライン再定義を実行できます。

複数のステップで表のオンライン再定義を実行するには:

1. 再定義方法(キー別またはROWID別)を選択します。

   キー別ー再定義に使用する主キーまたは疑似主キーを選択します。疑似主キーは、NOT NULL制約が指定されているすべての構成要素の列を備えた一意のキーです。この方法の場合、表の再定義前のバージョンと再定義後のバージョンの主キー列は同じになります。これはデフォルトの再定義方法であり、この方法を使用することをお薦めします。

   ROWID別ーキーが存在しない場合、この方法を使用します。この方法では、M_ROW$$という名前の非表示列が、表の再定義後のバージョンに追加されます。再定義の完了後、この列を削除するか、未使用としてマークすることをお薦めします。再定義の最終フェーズで、この列は自動的に未使用に設定されます。その後、ALTER TABLE ... DROP UNUSED COLUMNS文を使用して削除できます。

   この方法は、索引構成表に対しては使用できません。

2. CAN_REDEF_TABLEプロシージャを起動して、表をオンラインで再定義できることを確認します。表がオンライン再定義の候補でない場合、このプロシージャは表をオンライン再定義できない理由を示すエラーを出力します。
3. 必要な論理属性と物理属性のすべてを備えた空の仮表を(再定義する表と同じスキーマ内に)作成します。削除される列の場合は、仮表の定義に含めないでください。列を追加する場合は、その列の定義を仮表に追加します。列を変更する場合は、必要なプロパティを備えた仮表にその列を作成します。

   再定義する表の索引、制約、権限付与およびトリガーすべてを備えた仮表を作成する必要はありません。これらは、依存オブジェクトをコピーするときにステップ7で定義します。

4. ROWIDによる方法でパーティション表を再定義する場合は、仮表での行移動を有効にします。

   ```
   ALTER TABLE ... ENABLE ROW MOVEMENT;
   ```

5. (オプション)次のステップのパフォーマンスを改善するために、大きい表の再定義をパラレルで実行する場合は、次の文を発行します。

   ```
   ALTER SESSION FORCE PARALLEL DML PARALLEL degree-of-parallelism;
   ALTER SESSION FORCE PARALLEL QUERY PARALLEL degree-of-parallelism;
   ```

6. 次の情報を指定してSTART_REDEF_TABLEをコールし、再定義プロセスを開始します。
   - unameおよびorig_tableパラメータでそれぞれ再定義する表のスキーマと表名
   - int_tableパラメータの仮表名
   - col_mappingパラメータの再定義される表の列を仮表の列にマップする列マッピング文字列

     詳細は、「列マッピング文字列の作成」を参照してください。
   - options_flagパラメータの再定義方法

     再定義方法を指定するために、パッケージ定数が用意されています。
     DBMS_REDEFINITION.CONS_USE_PKは、主キーまたは擬似主キーを使用して再定義が実行されるように指定するために使用します。DBMS_REDEFINITION.CONS_USE_ROWIDは、ROWIDを使用して再定義が実行されるように指定するために使用します。この引数を指定しない場合は、デフォルトの再定義方法(CONS_USE_PK)が使用されます。
   - orderby_colsパラメータの行の順序に使用する列(オプション)
   - part_nameパラメータのパーティション名(パーティション表の1つのパーティションまたは複数のパーティションを再定義する場合)

     詳細は、「1つ以上のパーティションのオンライン再定義」を参照してください。
   - copy_vpd_optパラメータで表に対して定義される仮想プライベート・データベース(VPD)ポリシーの処理方法

     詳細は、「オンライン再定義中の仮想プライベート・データベース(VPD)ポリシーの処理」を参照してください。

   このプロセスにはデータのコピー操作が含まれるため、多少の時間を要する可能性があります。再定義する表は、プロセスの開始から終了まで問合せおよびDMLで使用できます。

   ノート:

   - DBA_REDEFINITION_OBJECTS ビューを問い合せると、オンライン再定義に現在関連するオブジェクトをリストできます。
   - なんらかの理由で START_REDEF_TABLE に失敗した場合は、ABORT_REDEF_TABLE をコールする必要があります。コールしないと、表を再定義する後続の試行でエラーが発生します。

7. 依存オブジェクト(トリガー、索引、マテリアライズド・ビュー・ログ、権限付与および制約など)および統計を、再定義される表から仮表へ、次の2つの方法のいずれかを使用してコピーします。方法1の方がより自動化されているため優先されますが、方法2の使用を選択する場合もあります。また、方法1では、表統計を仮表にコピーできます。
   - 方法1: 依存オブジェクトの自動作成

     COPY_TABLE_DEPENDENTSプロシージャを使用して、仮表に対する依存オブジェクトを自動的に作成します。このプロシージャは、依存オブジェクトの登録も実施します。依存オブジェクトを登録することで、これらのオブジェクトの個別情報とコピーされた複製を、再定義完了プロセスの一部として後で自動的にスワップできます。その結果、再定義が完了すると、依存オブジェクトの名前がオリジナルの依存オブジェクトと同じ名前になります。

     詳細は、「依存オブジェクトの自動作成」を参照してください。

- 方法2: 依存オブジェクトの手動による作成

  仮表に対する依存オブジェクトは、手動で作成して登録できます。詳細は、「依存オブジェクトの手動による作成」を参照してください。

> ノート:
>
> Oracle9i では、トリガー、索引、権限付与および制限を仮表に手動で作成する必要がありましたが、まだその必要がある場合もあります。いずれの場合でも、仮表に関係のある参照制限(仮表が参照制約の親表または子表の場合)は、使用不可で作成される必要があります。オンライン再定義が完了した後、参照制限は自動的に使用可能になります。さらに、再定義プロセスが完了または終了されるまで、仮表に定義されたトリガーはまったく実行されません。

8. FINISH_REDEF_TABLEプロシージャを実行して、表の再定義を完了します。このプロシージャの実行中、元の表はそのデータ量とは無関係に、わずかな時間ですが排他モードでロックされます。ただし、FINISH_REDEF_TABLE部分は、保留中のDMLすべてがコミットされるのを待機してから、再定義を完了します。

   FINISH_REDEF_TABLEプロシージャのdml_lock_timeoutパラメータを使用して、保留中のDMLのコミットをプロシージャが待機する時間を指定できます。このパラメータでは、プロシージャが正常に終了する前に待機する秒数を指定します。このパラメータにNULL以外の値を指定した場合、FINISH_REDEF_TABLEプロシージャを再起動し、タイムアウトしたポイントから続行できます。パラメータをNULLに設定した場合、プロシージャはタイムアウトしません。この場合、プロシージャを手動で停止すると、ABORT_REDEF_TABLEプロシージャを使用して表のオンライン再定義を終了し、ステップ6からやり直す必要があります。

9. 仮表に対する長時間実行の問合せがある場合は、完了するのを待ってから、仮表を削除します。

   仮表に対するアクティブな問合せの実行中に仮表を削除すると、ORA-08103エラー(「現在、指定したオブジェクトは存在しません。」)が発生する場合があります。

関連項目:

- 「表のオンライン再定義の例」
- パッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

親トピック: DBMS_REDEFINITIONの複数のプロシージャを使用した表のオンライン再定義

20.8.6.2 列マッピング文字列の作成

引数としてSTART_REDEF_TABLEに渡す列マッピング文字列には、カンマで区切られた列マッピングのペアのリストが含まれています。

各ペアの構文は次のとおりです。

```
[expression]  column_name
```

column_nameは、仮表の列を意味します。オプションのexpressionには、SQL(SELECT)文の式のルールに従って、再定義する表の列、定数、演算子、関数またはメソッド・コールなどを指定できます。ただし、使用できるのは、値がすぐに決定される単純な副次式、つまり、ある評価と次の評価で結果が変化しない副次式と、順序およびSYSDATEのみです。副問合せは使用できません。最も簡単な場合、式は再定義する表の列名のみで構成されます。

式を指定すると、その値は再定義の過程で仮表内の指定の列に配置されます。式を省略した場合は、再定義する表と仮表の両方にcolumn_nameという列が存在し、再定義する表にあるその列の値が仮表の同じ列に配置されていると想定されます。

たとえば、再定義する表のoverride列をoverride_commissionという名前に変更し、すべてのオーバーライド・コミッションを2%増加する場合、正しい列マッピングのペアは次のとおりです。

```
override*1.02  override_commission
```

列マッピング文字列に'*'またはNULLを指定すると、すべての列(名前は変更されない)が仮表に配置されることになります。それ以外の場合は、文字列で明示的に指定した列のみが仮表に配置されます。列マッピングのペアの順序は重要ではありません。

列マッピング文字列の例は、「表のオンライン再定義の例」を参照してください。

データ変換

いくつか制限がありますが、列のマッピングの際にデータ型を変換できます。

'*'またはNULLを列マッピング文字列として指定した場合は、SQLで許可される暗黙的な変換のみがサポートされます。たとえば、CHARからVARCHAR2に、INTEGERからNUMBERに変換できます。

あるオブジェクト型から別のオブジェクト型への変換や、あるコレクション型から別のコレクション型への変換など、その他のデータ型変換を実行するには、変換を実行する式とともに列マッピングのペアを指定する必要があります。式には、CAST関数、TO_NUMBERなどの組込み関数、作成した変換関数などを指定できます。

親トピック: DBMS_REDEFINITIONの複数のプロシージャを使用した表のオンライン再定義

20.8.6.3 オンライン再定義中の仮想プライベート・データベース(VPD)ポリシーの処理

再定義する元の表にVPDポリシーが指定されている場合、START_REDEF_TABLEプロシージャでcopy_vpd_optパラメータを使用して、オンライン再定義中にこれらのポリシーを処理できます。

このパラメータには、次の値を指定できます。

パラメータ値	説明
DBMS_REDEFINITION.CONS_VPD_NONE	元の表に VPD ポリシーがない場合は、この値を指定します。この値がデフォルトです。 この表を指定したときに、元の表に VPD ポリシーが存在する場合は、エラーが発生します。
DBMS_REDEFINITION.CONS_VPD_AUTO	オンライン再定義中に元の表から新しい表に VPD ポリシーを自動的にコピーするには、この値を指定します。
DBMS_REDEFINITION.CONS_VPD_MANUAL	オンライン再定義中に元の表から新しい表に VPD ポリシーを手動でコピーするには、この値を指定します。

元の表にVPDポリシーが指定されていない場合は、copy_vpd_optパラメータにデフォルト値のDBMS_REDEFINITION.CONS_VPD_NONEを指定します。

元の表と仮表で列名と列の型が同じ場合は、copy_vpd_optパラメータにDBMS_REDEFINITION.CONS_VPD_AUTOを指定します。この値を使用するには、元の表と仮表の間の列マッピング文字列をNULLまたは'*'にする必要があります。copy_vpd_optパラメータにDBMS_REDEFINITION.CONS_VPD_AUTOを使用した場合、表の所有者およびオンライン再定義を開始したユーザーのみがオンライン再定義中に仮表にアクセスできます。

次のいずれかの条件が満たされる場合は、copy_vpd_optパラメータにDBMS_REDEFINITION.CONS_VPD_MANUALを指定します。

- 元の表にVPDポリシーが指定されており、元の表と仮表の間に列マッピングがあります。
- 表のオンライン再定義中にVPDポリシーを追加または変更します。

VPDポリシーを手動でコピーするには、START_REDEF_TABLEプロシージャを実行する前に、仮表にVPDポリシーを指定します。表のオンライン再定義が完了すると、再定義された表には変更されたポリシーが指定されています。

関連項目:

- VPDポリシーが指定された表に関する制限事項の詳細は、「表のオンライン再定義に関する制限事項」を参照してください
- VPDポリシーが指定された表を再定義する例は、「表のオンライン再定義の例」を参照してください
- Oracle Databaseセキュリティ・ガイド

親トピック: DBMS_REDEFINITIONの複数のプロシージャを使用した表のオンライン再定義

20.8.6.4 依存オブジェクトの自動作成

仮表に対する依存オブジェクトを自動的に作成するには、COPY_TABLE_DEPENDENTSプロシージャを使用します。

num_errors出力引数をチェックすることで、依存オブジェクトのコピー中にエラーが発生したかどうかを検出できます。ignore_errors引数をTRUEに設定すると、COPY_TABLE_DEPENDENTSプロシージャは、オブジェクト作成時にエラーを検出しても、依存オブジェクトのコピーを続行します。DBA_REDEFINITION_ERRORSビューを問い合せることで、これらのエラーを確認できます。

エラーには、次のような理由があります。

- システム・リソースの不足
- 依存オブジェクトの再コーディングが必要になるような表の論理構造の変更。

  この種のエラーの説明は、「表のオンライン再定義の例」の例3を参照してください。

ignore_errorsをFALSEに設定すると、COPY_TABLE_DEPENDENTSプロシージャは、エラーを検出すると、オブジェクトのコピーをただちに停止します。

エラーを修正してからCOPY_TABLE_DEPENDENTSプロシージャを再実行することで、依存オブジェクトのコピーを再試行できます。「依存オブジェクトの手動による作成」に説明されているように、オブジェクトを手動で作成し、それらを登録することもできます。COPY_TABLE_DEPENDENTSプロシージャは、必要に応じて何回でも使用できます。オブジェクトがすでに正常にコピーされている場合は、再度コピーされません。

親トピック: DBMS_REDEFINITIONの複数のプロシージャを使用した表のオンライン再定義

20.8.6.5 依存オブジェクトの手動による作成

SQL*PlusまたはCloud Controlで、仮表に対する依存オブジェクトを手動で作成する場合は、REGISTER_DEPENDENT_OBJECTプロシージャを使用して依存オブジェクトを登録する必要があります。依存オブジェクトを登録すると、再定義の完了プロセスで、依存オブジェクト名を再定義前の名前にリストアできます。

次に、依存オブジェクトを手動で作成するために必要な変更の例を示します。

- 別の表領域への索引の移動
- 索引の列の変更
- 制約の変更
- トリガーの変更
- マテリアライズド・ビュー・ログの変更

REGISTER_DEPENDENT_OBJECTプロシージャを実行するときに、dep_typeパラメータで依存オブジェクトのタイプを指定する必要があります。このパラメータで次の定数を指定できます。

- 依存オブジェクトが索引の場合はDEMS_REDEFINITION.CONS_INDEX
- 依存オブジェクト・タイプが制約の場合はDEMS_REDEFINITION.CONS_CONSTRAINT
- 依存オブジェクトがトリガーの場合はDEMS_REDEFINITION.CONS_TRIGGER
- 依存オブジェクトがマテリアライズド・ビュー・ログの場合はDEMS_REDEFINITION.CONS_MVLOG

COPY_TABLE_DEPENDENTSプロシージャによる依存オブジェクトのコピーがエラーとなり、手動による介入が必要な場合は、REGISTER_DEPENDENT_OBJECTプロシージャを使用します。

DBA_REDEFINITION_OBJECTSビューを問い合せることによって、登録されている依存オブジェクトを判断できます。このビューには、REGISTER_DEPENDENT_OBJECTプロシージャで明示的に登録、またはCOPY_TABLE_DEPENDENTSプロシージャで暗黙的に登録された依存オブジェクトが表示されます。このビューには、現在の情報のみが表示されます。

UNREGISTER_DEPENDENT_OBJECTプロシージャを使用すると、再定義している表および仮表に対する依存オブジェクトの登録を解除できます。

ノート:

- 手動で作成する依存オブジェクトは、対応する元の依存オブジェクトと同一である必要はありません。たとえば、マテリアライズド・ビュー・ログを仮表に手動で作成する場合は、別の列を記録できます。また、仮表の依存オブジェクトが増減してもかまいません。
- 指定されたLOBセグメントが再定義する表に含まれている場合、LOBセグメント名は、オンライン再定義中にシステム生成の名前に置き換えられます。これを回避するには、新しいLOBセグメント名で仮表を作成できます。

関連項目:

依存オブジェクトを登録する例は、「表のオンライン再定義の例」の例4を参照してください

親トピック: DBMS_REDEFINITIONの複数のプロシージャを使用した表のオンライン再定義

20.8.7 再定義プロセスの結果

再定義プロセスにはいくつかの結果があります。

再定義プロセスの最終的な結果は、次のようになります。

- REDEF_TABLEまたはCOPY_TABLE_DEPENDENTSが使用された場合、元の表は、仮表の列、索引、制約、権限付与、トリガーおよび統計を使用して再定義されます。
- REGISTER_DEPENDENT_OBJECTを明示的に使用するか、またはCOPY_TABLE_DEPENDENTSを暗黙的に使用して登録された依存オブジェクトは、自動的に名前が変更されるため、再定義した表の依存オブジェクト名は再定義の前と同じになります。

ノート:

登録または自動コピーが行われていない依存オブジェクトの名前は、手動で変更する必要があります。

- 仮表に関与する参照制約は、再定義した表に関与して使用可能になります。
- (再定義前に)元の表に定義されていた索引、トリガー、マテリアライズド・ビュー・ログ、権限付与および制約がすべて仮表に移動し、ユーザーが仮表を削除したときに同時に削除されます。再定義する前に元の表に関与していた参照制約がすべて仮表に関与し、使用禁止になります。
- 一部のPL/SQLオブジェクト、ビュー、シノニムおよびその他の表依存オブジェクトが、無効になる場合があります。変更された表の要素に依存するオブジェクトのみが無効になります。たとえば、再定義で変更されなかった再定義表の列のみを問い合せるPL/SQLプロシージャは有効のままです。スキーマ・オブジェクトの依存性の詳細は、「オブジェクト依存性の管理」を参照してください。

親トピック: 表のオンライン再定義

20.8.8 中間での同期化の実行

元の表に対して多数のDML文を実行する場合は、再定義プロセス中に仮表を元の表と同期できます。

START_REDEF_TABLEをコールして再定義プロセスを開始してからFINISH_REDEF_TABLEコールが完了するまでの間に、元の表に対して多数のDML文が実行される可能性があります。これが問題になることがわかっている場合は、定期的に仮表を元の表と同期化することをお薦めします。

START_REDEF_TABLEプロシージャを使用して表のオンライン再定義操作を開始した場合は、同期を容易にするために内部マテリアライズド・ビューが作成されます。この内部マテリアライズド・ビューがリフレッシュされて、仮表が元の表と同期されます。

仮表と元の表を同期するには、次にようにします。

- DBMS_REDEFINITIONパッケージのSYNC_INTERIM_TABLEプロシージャを実行します。

このプロシージャをコールすると、FINISH_REDEF_TABLEで再定義プロセスを完了するための時間が短縮されます。SYNC_INTERIM_TABLEをコールできる回数に制限はありません。

FINISH_REDEF_TABLEの実行中に元の表がロックされるわずかな時間は、SYNC_INTERIM_TABLEのコールの有無とは関係ありません。

20.8.9 表のオンライン再定義中に依存マテリアライズド・ビューをリフレッシュする方法

表のオンライン再定義中に、高速リフレッシュ可能な依存マテリアライズド・ビューをリフレッシュするには、REDEF_TABLEプロシージャまたはSTART_REDEF_TABLEプロシージャで、refresh_dep_mviewsパラメータをYに設定します。

依存マテリアライズド・ビューは、再定義される表で定義されているマテリアライズド・ビューです。表のオンライン再定義後に依存マテリアライズド・ビューの完全リフレッシュを実行すると、時間がかかることがあります。表のオンライン再定義中に、高速リフレッシュ可能なマテリアライズド・ビューを増分リフレッシュすると、処理を効率化できます。

依存マテリアライズド・ビューのリフレッシュには、次の制限が適用されます。

- マテリアライズド・ビューが高速リフレッシュ可能な必要があります。
- ROWIDマテリアライズド・ビューはサポートされません。
- マテリアライズド結合ビューはサポートされません。

表のオンライン再定義後に、依存ROWIDマテリアライズド・ビューおよびマテリアライズド結合ビューの完全リフレッシュが必要です。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。
   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。
   「SQL*Plusを使用したデータベースへの接続」を参照してください。
2. 次のいずれかの方法で、表のオンライン再定義を実行します。
   - REDEF_TABLEプロシージャを実行し、refresh_dep_mviewsパラメータがYに設定されていることを確認します。

     この方法では、再定義操作の最後に一度、依存マテリアライズド・ビューの高速リフレッシュが実行されます。
   - START_REDEF_TABLEプロシージャを使用して表のオンライン再定義を開始し、refresh_dep_mviewsパラメータがYに設定されていることを確認します。この方法はFINISH_REDEF_TABLEプロシージャで終了します。

     この方法では、START_REDEF_TABLEプロシージャの実行時、SYNC_INTERIM_TABLEプロシージャの実行のたび、およびFINISH_REDEF_TABLEプロシージャの実行時に、依存マテリアライズド・ビューの高速リフレッシュが実行されます。

ノート:

- 表のオンライン再定義操作の refresh_dep_mviews パラメータの値を確認するには、DBA_REDEFINITION_STATUS ビューを問い合せます。
- 表のオンライン再定義中に自動的に実行されるリフレッシュの進行状況を確認するには、V$ONLINE_REDEF ビューの REFRESH_STATEMENT_SQL_ID 列および REFRESH_STATEMENT 列を問い合せます。REFRESH_STATEMENT_SQL_ID 列で返される SQL_ID 値を使用して、V$SQL ビューや V$SQL_MONITOR ビューなどでリフレッシュの進行状況を監視できます。
- 表のオンライン再定義操作中に refresh_dep_mviews パラメータの値を変更する

場合は、DBMS_REDEFINITION.SET_PARAM プロシージャを使用してパラメータをリセットします。

例20-15 REDEF_TABLEプロシージャ実行中の依存マテリアライズド・ビューのリフレッシュ

この例では、hr.employees表を再定義して、高度な行圧縮で表を圧縮し、依存マテリアライズド・ビューをリフレッシュしています。

```
BEGIN
  DBMS_REDEFINITION.REDEF_TABLE(
    uname                 => 'HR',
    tname                 => 'EMPLOYEES',
    table_compression_type => 'ROW STORE COMPRESS ADVANCED',
    refresh_dep_mviews    => 'Y');
END;
/
```

例20-16 START_REDEF_TABLEプロシージャによる開始時の依存マテリアライズド・ビューのリフレッシュ

oe.orders表を再定義する必要があるとします。次に表の定義を示します。

```
CREATE TABLE oe.orders(
     order_id      NUMBER(12),
     order_date    TIMESTAMP WITH LOCAL TIME ZONE,
     order_mode    VARCHAR2(8),
     customer_id   NUMBER(6),
     order_status  NUMBER(2),
     order_total   NUMBER(8,2),
     sales_rep_id  NUMBER(6),
     promotion_id  NUMBER(6));
```

この例では、表を再定義してorder_mode列のサイズを16に増やしています。次に仮表の定義を示します。

```
CREATE TABLE oe.int_orders(
     order_id      NUMBER(12),
     order_date    TIMESTAMP WITH LOCAL TIME ZONE,
     order_mode    VARCHAR2(16),
     customer_id   NUMBER(6),
     order_status  NUMBER(2),
     order_total   NUMBER(8,2),
     sales_rep_id  NUMBER(6),
     promotion_id  NUMBER(6));
```

また、この表には依存するマテリアライズド・ビューがあるとします。表には、次の文で作成されるマテリアライズド・ビュー・ログが含まれます。

```
CREATE MATERIALIZED VIEW LOG ON oe.orders WITH PRIMARY KEY, ROWID;
```

oe.orders表には、次の依存マテリアライズド・ビューが含まれます。

```
CREATE MATERIALIZED VIEW oe.orders_pk REFRESH FAST AS
 SELECT * FROM oe.orders;
CREATE MATERIALIZED VIEW oe.orders_rowid REFRESH FAST WITH ROWID AS
 SELECT * FROM oe.orders;
```

oe.orders_pkマテリアライズド・ビューは高速リフレッシュ可能な主キー・マテリアライズド・ビューです。したがって、表のオンライン再定義中にリフレッシュできます。

oe.orders_rowidマテリアライズド・ビューは高速リフレッシュ可能ですが、これはROWIDマテリアライズド・ビューです。したがって、表のオンライン再定義中にリフレッシュできません。

次のステップで、oe.orders表のオンライン再定義を実行し、同時にoe.orders_pkマテリアライズド・ビューをリフレッシュします。

1. 再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE(
    uname              => 'oe',
    orig_table         => 'orders',
    int_table          => 'int_orders',
    options_flag       => DBMS_REDEFINITION.CONS_USE_PK,
    refresh_dep_mviews => 'Y');
END;
/
```

2. 依存オブジェクトをコピーします。(oe.int_ordersに対するトリガー、索引、マテリアライズド・ビュー・ログ、権限付与および制約がある場合、それらは自動的に作成されます。)

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'oe',
    orig_table       => 'orders',
    int_table        => 'int_orders',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => TRUE,
    num_errors       => num_errors);
END;
/
```

3. 再定義ステータスを確認します。

```
SELECT REDEFINITION_ID, REFRESH_DEP_MVIEWS
   FROM DBA_REDEFINITION_STATUS
   WHERE BASE_TABLE_OWNER = 'OE' AND BASE_TABLE_NAME = 'ORDERS';
```

4. 元の表に対してDMLを実行します。たとえば:

```
INSERT INTO oe.orders VALUES(3000,sysdate,'direct',102,1,42283.2,154,NULL);
COMMIT;
```

5. 仮表oe.int_ordersを同期します。このステップで、依存マテリアライズド・ビューoe.orders_pkがリフレッシュされます。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'OE',
    orig_table => 'ORDERS',
    int_table  => 'INT_ORDERS');
END;
/
```

6. oe.orders表の依存マテリアライズド・ビューのリフレッシュ・ステータスを確認します。

```
SELECT m.OWNER, m.MVIEW_NAME, m.STALENESS, m.LAST_REFRESH_DATE
   FROM ALL_MVIEWS m, ALL_MVIEW_DETAIL_RELATIONS d
   WHERE m.OWNER=d.OWNER AND
         m. MVIEW_NAME=d.MVIEW_NAME AND
         d.DETAILOBJ_OWNER = 'OE' AND
```

```
          d.DETAILOBJ_NAME = 'ORDERS';
```

oe.orders_pkマテリアライズド・ビューは前のステップでリフレッシュされたため、STALENESSステータスはFRESHです。oe.orders_rowidマテリアライズド・ビューは前のステップでリフレッシュされなかったため、STALENESSステータスはNEEDS_COMPLILEです。

7. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'OE',
    orig_table => 'ORDERS',
    int_table  => 'INT_ORDERS');
END;
/
```

oe.orders_pkマテリアライズド・ビューを問い合せると、表のオンライン再定義中にリフレッシュされたため、oe.orders表に挿入された新しい行がマテリアライズド・ビューに存在することを確認できます。

関連トピック

- 表のオンライン再定義の進行状況の監視

親トピック: 表のオンライン再定義

20.8.10 表のオンライン再定義の進行状況の監視

V$ONLINE_REDEFビューを問い合せて、表のオンライン再定義操作の進行状況を監視できます。

表のオンライン再定義プロセス中は、一部の操作の実行に時間がかかる場合があります。これらの操作の実行中にV$ONLINE_REDEFビューを問い合せて、操作の進行状況について詳細な情報を確認できます。たとえば、DBMS_REDEFINITION.START_REDEF_TABLEプロシージャでデータを仮表にロードするために時間がかかることがあります。

V$ONLINE_REDEFビューのPROGRESS列に、操作の完了がパーセンテージで示されます。このビューには、OPERATION列の操作を完了するために必要なステップの総数のうち、現在のステップが示されます。たとえば、操作に10のステップがある場合は、この列にStep 6 out of 10のように示されます。ビューにはSUBOPERATION列およびDETAILED_MESSAGE列も含まれており、現在の操作についてより詳細な情報を確認できます。

表のオンライン再定義プロセス中に内部マテリアライズド・ビューが作成され、このマテリアライズド・ビューが一部の操作の実行中にリフレッシュされて、元の表と仮表の同期が維持されます。表のオンライン再定義中に自動的に実行されるリフレッシュの進行状況を確認するには、V$ONLINE_REDEFビューのREFRESH_STATEMENT_SQL_ID列およびREFRESH_STATEMENT列を問い合せます。REFRESH_STATEMENT_SQL_ID列で返されるSQL_ID値を使用して、V$SQLビューやV$SQL_MONITORビューなどでリフレッシュの進行状況を監視できます。

1. 表のオンライン再定義を実行中のセッションとは別個のセッションのデータベースに接続します。
2. V$ONLINE_REDEFビューを問い合せます。

例20-17 表のオンライン再定義の進行状況の監視

この例では、cust_alt_phone_number列を追加して、Oracleが提供するsh.customers表を再定義します。

```
CREATE TABLE customers (
    cust_id                 NUMBER          NOT NULL,
    cust_first_name         VARCHAR2(20)    NOT NULL,
    cust_last_name          VARCHAR2(40)    NOT NULL,
    cust_gender             CHAR(1)         NOT NULL,
```

```
    cust_year_of_birth     NUMBER(4)       NOT NULL,
    cust_marital_status    VARCHAR2(20),
    cust_street_address    VARCHAR2(40)    NOT NULL,
    cust_postal_code       VARCHAR2(10)    NOT NULL,
    cust_city              VARCHAR2(30)    NOT NULL,
    cust_city_id           NUMBER          NOT NULL,
    cust_state_province    VARCHAR2(40)    NOT NULL,
    cust_state_province_id NUMBER          NOT NULL,
    country_id             NUMBER          NOT NULL,
    cust_main_phone_number VARCHAR2(25)    NOT NULL,
    cust_income_level      VARCHAR2(30),
    cust_credit_limit      NUMBER,
    cust_email             VARCHAR2(50),
    cust_total             VARCHAR2(14)    NOT NULL,
    cust_total_id          NUMBER          NOT NULL,
    cust_src_id            NUMBER,
    cust_eff_from          DATE,
    cust_eff_to            DATE,
    cust_valid             VARCHAR2(1));
```

この表には大量のデータが含まれるため、表のオンライン再定義プロセスの一部の操作には時間がかかります。この例では、V$ONLINE_REDEFビューを問い合せて様々な操作を監視します。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 仮表sh.int_customersを作成します。

```
CREATE TABLE sh.int_customers (
    cust_id                NUMBER          NOT NULL,
    cust_first_name        VARCHAR2(20)    NOT NULL,
    cust_last_name         VARCHAR2(40)    NOT NULL,
    cust_gender            CHAR(1)         NOT NULL,
    cust_year_of_birth     NUMBER(4)       NOT NULL,
    cust_marital_status    VARCHAR2(20),
    cust_street_address    VARCHAR2(40)    NOT NULL,
    cust_postal_code       VARCHAR2(10)    NOT NULL,
    cust_city              VARCHAR2(30)    NOT NULL,
    cust_city_id           NUMBER          NOT NULL,
    cust_state_province    VARCHAR2(40)    NOT NULL,
    cust_state_province_id NUMBER          NOT NULL,
    country_id             NUMBER          NOT NULL,
    cust_main_phone_number VARCHAR2(25)    NOT NULL,
    cust_income_level      VARCHAR2(30),
    cust_credit_limit      NUMBER,
    cust_email             VARCHAR2(50),
    cust_total             VARCHAR2(14)    NOT NULL,
    cust_total_id          NUMBER          NOT NULL,
    cust_src_id            NUMBER,
    cust_eff_from          DATE,
    cust_eff_to            DATE,
    cust_valid             VARCHAR2(1),
    cust_alt_phone_number  VARCHAR2(25));
```

3. 再定義プロセスを開始し、操作の進行状況を監視します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE(
    uname        => 'sh',
    orig_table   => 'customers',
    int_table    => 'int_customers',
```

```
    options_flag => DBMS_REDEFINITION.CONS_USE_PK);
END;
/
```

この操作の実行中に、表のオンライン再定義を実行しているセッションとは別のセッションでV$ONLINE_REDEFビューを問い合せて進行状況を監視します。

```
SELECT * FROM V$ONLINE_REDEF;
```

この問合せの出力例を次に示します。

- OPERATIONにSTART_REDEF_TABLE
- SUBOPERATIONにcomplete refresh the materialized view
- PROGRESSにstep 6 out of 7

4. 依存オブジェクトをコピーします。(sh.int_customersに対するトリガー、索引、マテリアライズド・ビュー・ログ、権限付与および制約がある場合、それらは自動的に作成されます。)

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'sh',
    orig_table       => 'customers',
    int_table        => 'int_customers',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => TRUE,
    num_errors       => num_errors);
END;
/
```

この操作の実行中に、表のオンライン再定義を実行しているセッションとは別のセッションでV$ONLINE_REDEFビューを問い合せて進行状況を監視します。

```
SELECT * FROM V$ONLINE_REDEF;
```

この問合せの出力例を次に示します。

- OPERATIONにCOPY_TABLE_DEPENDENTS
- SUBOPERATIONにcopy the indexes
- PROGRESSにstep 3 out of 7

このコールでは、ignore_errors引数がTRUEに設定されていることに注意してください。これは、仮表が主キー制約付きで作成されており、COPY_TABLE_DEPENDENTSによって、主キー制約と索引が元の表からコピーされる際にエラーが発生するためです。これらのエラーは無視できます。

5. 仮表hr.int_emp_redefを同期します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'sh',
    orig_table => 'customers',
    int_table  => 'int_customers');
END;
/
```

6. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'sh',
    orig_table => 'customers',
    int_table  => 'int_customers');
END;
/
```

関連トピック

- 表のオンライン再定義中に依存マテリアライズド・ビューをリフレッシュする方法

親トピック: 表のオンライン再定義

20.8.11 失敗後の表のオンライン再定義の再開

表のオンライン再定義が失敗した場合は、DBA_REDEFINITION_STATUSビューでエラー情報および再開可能性に関する情報を確認できます。

RESTARTABLEがYの場合は、エラーを修正して、最後に停止したところからオンライン再定義プロセスを再開できます。RESTARTABLEがNの場合は、再定義操作を停止する必要があります。

場合によっては、失敗後に表のオンライン再定義を再開できます。操作の再開とは、失敗のために停止したところからオンライン再定義プロセスが開始し、処理内容が失われないことを意味します。たとえば、「表領域で表を拡張できません」エラーによってSYNC_INTERIM_TABLEプロシージャ・コールが失敗した場合、不足した表領域のサイズを増やすことで問題を修正して、SYNC_INTERIM_TABLEプロシージャ・コールを再実行できます。

表のオンライン再定義が失敗した場合は、次のステップを実行してそれを再開できます。

1. DBA_REDEFINITION_STATUSビューを問い合せて、失敗の原因および修正に必要なアクションを確認します。たとえば、次の問合せを実行します。

```
SELECT BASE_TABLE_NAME,
       INTERIM_OBJECT_NAME,
       OPERATION,
       STATUS,
       RESTARTABLE,
       ACTION
   FROM DBA_REDEFINITION_STATUS;
```

   RESTARTABLEの値がYの場合は、操作を再開できます。RESTARTABLE値がNの場合は操作を再開できないため、再定義を最初から再実行する必要があります。
2. 前のステップの問合せ結果で指定されたアクションを実行します。
3. 問合せ結果で指定された操作でオンライン再定義を再開し、後続のすべての操作を実行して、表のオンライン再定義を終了します。

例20-18 SYNC_INTERIM_TABLEプロシージャ・コールの失敗

この例は、次に示すエラーによって、SYNC_INTERIM_TABLEプロシージャ・コール時に失敗したオンライン再定義操作の再開を示しています。

```
BEGIN
DBMS_REDEFINITION.SYNC_INTERIM_TABLE('U1', 'ORIG', 'INT');
END;
/
ORA-42009: error occurred while synchronizing the redefinition
ORA-01653: unable to extend table U1.INT by 8 in tablespace my_tbs
```

```
ORA-06512: at "SYS.DBMS_REDEFINITION", line 148
ORA-06512: at "SYS.DBMS_REDEFINITION", line 2807
ORA-06512: at line 2
```

1. DBA_REDEFINITION_STATUSビューを問い合せます。

```
SELECT BASE_TABLE_NAME,  INT_TABLE_NAME, OPERATION, STATUS, RESTARTABLE, ACTION
   FROM DBA_REDEFINITION_STATUS;
BASE_TABLE_NAME INT_OBJ_NAME OPERATION          STATUS  RESTARTABLE ACTION
--------------- ------------ ------------------ ------- ----------- ---------
ORIG            INT          SYNC_INTERIM_TABLE FAILED  Y           Fix error
```

問合せ結果ではRESTARTABLEがYになっているため、オンライン再定義操作を再開できます。操作を再開するには、操作の失敗時に返されたエラーを修正してから再開します。この例ではエラーは、ORA-01653:「表U1.INTを8(表領域my_tbs)で拡張できません」です。

2. my_tbs表領域にデータファイルを追加してサイズを大きくします。

```
ALTER TABLESPACE my_tbs
   ADD DATAFILE '/u02/oracle/data/my_tbs2.dbf' SIZE 100M;
```

3. SYNC_INTERIM_TABLEプロシージャ・コールを再実行します。

```
BEGIN
   DBMS_REDEFINITION.SYNC_INTERIM_TABLE('U1', 'ORIG', 'INT');
END;
/
```

例20-19 マテリアライズド・ビュー・ログの問題

元の表に対する再定義が開始した後で、マテリアライズド・ビュー・ログに問題が発生する場合があります。たとえば、マテリアライズド・ビュー・ログがなんらかの原因で誤って削除されたり、破損する可能性があります。この場合、次のようなエラーが返されます。

```
ERROR at line 1:
ORA-42010: error occurred while synchronizing the redefinition
ORA-12034: materialized view log on "HR"."T1" younger than last refresh
```

次のSQL文で作成された表を再定義するとします。

```
CREATE TABLE hr.t1(
      c1 NUMBER PRIMARY KEY,
      c2 NUMBER)
   TABLESPACE example_tbs;
```

次のSQL文で仮表を作成し、表の表領域を変更するとします。

```
CREATE TABLE hr.int_t1(
      c1 NUMBER PRIMARY KEY,
      c2 NUMBER)
   TABLESPACE hr_tbs;
```

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE(
```

```
    uname           => 'hr',
    orig_table      => 't1',
    int_table       => 'int_t1');
END;
/
```

3. 元の表のマテリアライズド・ビュー・ログを削除します。

```
DROP MATERIALIZED VIEW LOG ON hr.t1;
```

4. 元の表に新しいマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON hr.t1
   WITH COMMIT SCN PURGE
   IMMEDIATE ASYNCHRONOUS;
```

5. 仮表hr.int_t1を同期します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'hr',
    orig_table => 't1',
    int_table  => 'int_t1');
END;
/
BEGIN
*
ERROR at line 1:
ORA-42010: error occurred while synchronizing the redefinition
ORA-12034: materialized view log on "HR"."T1" younger than last refresh
```

6. エラーが返されたため、DBA_REDEFINITION_STATUSビューを確認します。

```
COLUMN BASE_OBJECT_NAME FORMAT A11
COLUMN OPERATION FORMAT A10
COLUMN STATUS FORMAT A10
COLUMN RESTARTABLE FORMAT A11
COLUMN ERR_TXT FORMAT A15
COLUMN ACTION FORMAT A18
SELECT BASE_OBJECT_NAME, OPERATION, STATUS, RESTARTABLE, ERR_TXT, ACTION
   FROM DBA_REDEFINITION_STATUS
   ORDER BY BASE_TABLE_NAME, BASE_OBJECT_NAME;
BASE_OBJECT OPERATION  STATUS     RESTARTABLE ERR_TXT         ACTION
----------- ---------- ---------- ----------- --------------- ----------------
-
T1          SYNC_REDEF Failure    N           ORA-12034: mate Abort
redefinition
            _TABLE                            rialized view l
                                              og on "HR"."T1"
                                               younger than l
                                              ast refresh
```

問合せ結果で、RESTARTABLEがNになっており、ACTION列に表のオンライン再定義操作を終了する必要があると示されているため、オンライン再定義操作を再開できません。

7. 表のオンライン再定義操作を終了します。

```
BEGIN
  DBMS_REDEFINITION.ABORT_REDEF_TABLE(
    uname      => 'hr',
    orig_table => 't1',
    int_table  => 'int_t1');
END;
/
```

親トピック: 表のオンライン再定義

20.8.12 表のオンライン再定義のロールバック

表のオンライン再定義後のロールバックを有効にして、表を元の定義に戻し、表に対して行ったDML変更を保持できます。

- 表のオンライン再定義のロールバックについて
  表のオンライン再定義後に、表をオンライン再定義前の定義にロールバックし、同時に、表に対して行われたデータ操作言語(DML)変更をすべて保持できます。
- 表のオンライン再定義のロールバックの実行
  DBMS_REDEFINITIONパッケージのROLLBACKプロシージャは、DML変更を保持しながら、オンラインで再定義された表を元の定義に戻します。

親トピック: 表のオンライン再定義

20.8.12.1 表のオンライン再定義のロールバックについて

表のオンライン再定義後に、表をオンライン再定義前の定義にロールバックし、同時に、表に対して行われたデータ操作言語(DML)変更をすべて保持できます。

場合によっては、表のオンライン再定義を元に戻す必要が生じることがあります。たとえば、再定義後に表の操作のパフォーマンスが再定義前より低下する場合があります。このような場合、再定義後に表に対して行われたDML変更をすべて保持しながら、表を元の定義にロールバックできます。表のオンライン再定義のロールバックは、主に、再定義によって表の記憶域の特性が変化した場合、および変更により予期しないパフォーマンス低下が発生した場合に使用します。

表のオンライン再定義のロールバックを有効にするには、DBMS_REDEFINITION.START_TABLE_REDEFプロシージャでENABLE_ROLLBACKパラメータをTRUEに設定する必要があります。このパラメータをTRUEに設定すると、再定義中に作成された仮表が再定義完了後もOracle Databaseに保持されます。SYNC_INTERIM_TABLEプロシージャを実行して仮表を定期的に同期し、再定義された表に対して行われたDML変更を仮表に適用できます。内部マテリアライズド・ビューおよびマテリアライズド・ビュー・ログを使用して、仮表をメンテナンスできます。表のオンライン再定義をロールバックする場合は、仮表が同期され、表が元の定義になるようにOracle Databaseは仮表にスイッチ・バックします。

表のオンライン再定義のロールバックには、次の制限が適用されます。

- 元の表の列と暫定表の列に1対1のマッピングがない場合は、再定義中に列マッピングに演算子またはファンクションを使用できません。

  元の表の列と暫定表の列に1対1のマッピングがある場合は、列マッピングに演算子またはファンクションを使用できます。
- 再定義のロールバックが有効な場合は、表のオンライン再定義がロールバックされるか終了するまで、表を再度再定義はできません。

親トピック: 表のオンライン再定義のロールバック

20.8.12.2 表のオンライン再定義のロールバックの実行

DBMS_REDEFINITIONパッケージのROLLBACKプロシージャは、DML変更を保持しながら、オンラインで再定義された表を元の定義に戻します。

ROLLBACKプロシージャを使用するには、表のオンライン再定義時に、表のオンライン再定義のロールバックを有効にする必要があります。表のオンライン再定義で行われた変更を保持する場合は、ABORT_ROLLBACKプロシージャを実行します。

1. START_REDEF_TABLEプロシージャで開始し、FINISH_REDEF_TABLEプロシージャで終了する表のオンライン

再定義を実行します。

START_REDEF_TABLEプロシージャで、ENABLE_ROLLBACKパラメータをTRUEに設定する必要があります。このパラメータのデフォルトはFALSEです。

2. **オプション:** SYNC_INTERIM_TABLEプロシージャを定期的に実行して、再定義された表に対して行われたDML変更を仮表に適用できます。

   DML変更を定期的に仮表に適用することで、表のオンライン再定義のロールバックのパフォーマンスを改善できます。

3. 次のいずれかのオプションを選択します。
   - 表のオンライン再定義で行われた変更を元に戻して、元の表の定義に戻す場合は、DBMS_REDEFINITIONパッケージのROLLBACKプロシージャを実行します。
   - 表のオンライン再定義で行われた変更を保持する場合は、DBMS_REDEFINITIONパッケージのABORT_ROLLBACKプロシージャを実行します。

     ロールバックを終了すると、仮表のメンテナンスが停止し、ロールバックを有効にしたマテリアライズド・ビューおよびマテリアライズド・ビュー・ログが削除されます。

例20-20 表のオンライン再定義のロールバック

この例では、表の記憶域特性の変更による表のオンライン再定義について説明します。特に、この例ではオンライン再定義中に表の表領域を圧縮します。オンライン再定義の完了後に、表のパフォーマンスを評価する必要があるとします。表のパフォーマンスが予想より低い場合は、オンライン再定義による変更をロールバックできます。

次の文で元の表領域と表を作成したとします。

```
CREATE TABLESPACE tst_rollback_tbs
   DATAFILE 'tst_rollback_tbs.dbf' SIZE 10M
   ONLINE;
CREATE TABLE hr.tst_rollback
    (rllbck_id    NUMBER(6) PRIMARY KEY,
     rllbck_name  VARCHAR2(20))
   TABLESPACE tst_rollback_tbs
   STORAGE (INITIAL 2M);
```

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 仮表用に圧縮された表領域を作成します。

   ```
   CREATE TABLESPACE tst_cmp_rollback_tbs
      DEFAULT ROW STORE COMPRESS ADVANCED
      DATAFILE 'tst_cmp_rollback_tbs.dbf' SIZE 10M
      ONLINE;
   ```

3. 仮表hr.int_tst_rollbackを作成します。

   ```
   CREATE TABLE hr.int_tst_rollback
       (rllbck_id    NUMBER(6) PRIMARY KEY,
        rllbck_name  VARCHAR2(20))
      TABLESPACE tst_cmp_rollback_tbs
      STORAGE (INITIAL 2M);
   ```

   仮表が、前のステップで作成した圧縮された表領域を使用していることを確認します。

4. 再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE(
    uname           => 'hr',
    orig_table      => 'tst_rollback',
    int_table       => 'int_tst_rollback',
    options_flag    => DBMS_REDEFINITION.CONS_USE_PK,
    enable_rollback => TRUE);
END;
/
```

enable_rollbackがTRUEに設定されており、オンライン再定義による変更がロールバックできることを確認します。

5. 依存オブジェクトをコピーします。

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'hr',
    orig_table       => 'tst_rollback',
    int_table        => 'int_tst_rollback',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => TRUE,
    num_errors       => num_errors);
END;
/
```

6. DBA_REDEFINITION_ERRORSビューを問い合せて、エラーをチェックします。

```
SET LONG  8000
SET PAGES 8000
COLUMN OBJECT_NAME HEADING 'Object Name' FORMAT A20
COLUMN BASE_TABLE_NAME HEADING 'Base Table Name' FORMAT A10
COLUMN DDL_TXT HEADING 'DDL That Caused Error' FORMAT A40

SELECT OBJECT_NAME, BASE_TABLE_NAME, DDL_TXT FROM
         DBA_REDEFINITION_ERRORS;
```

主キーと索引に関連するエラーは無視できます。

7. 仮表hr.int_tst_rollbackを同期します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'hr',
    orig_table => 'tst_rollback',
    int_table  => 'int_tst_rollback');
END;
/
```

8. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'hr',
    orig_table => 'tst_rollback',
    int_table  => 'int_tst_rollback');
END;
/
```

このステップが終了するまでに、わずかな間のみ、表hr.tst_rollbckが排他モードでロックされます。このコールの

後、表hr.tst_rollbackはhr.int_tst_rollback表のすべての属性を持つように再定義されます。この例では、hr.tst_rollbck表の表領域が圧縮されました。

9. 評価期間中に、仮表hr.int_tst_rollbackを定期的に同期できます。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'hr',
    orig_table => 'tst_rollback',
    int_table  => 'int_tst_rollback');
END;
/
```

表を同期すると、再定義された表に対して行われたDML変更によって元の表が更新されます。定期的に表を同期すると、元の表に対するDML変更を削減できるため、ロールバック操作が効率的になります。

DBA_REDEFINITION_STATUSビューのSTATUS列を問い合せると、ロールバック操作のステータスを判別できます。

10. 次のいずれかのアクションを実行します。

- 表のパフォーマンスが予想より低かったため、オンライン再定義による変更をロールバックするとします。

```
BEGIN
  DBMS_REDEFINITION.ROLLBACK(
    uname      => 'hr',
    orig_table => 'tst_rollback',
    int_table  => 'int_tst_rollback');
END;
/
```

- 再定義された表のパフォーマンスが予想どおりだったため、ロールバックを終了して表のオンライン再定義による変更内容を保持し、ロールバックを有効にするデータベース・オブジェクトをクリーン・アップするとします。

```
BEGIN
  DBMS_REDEFINITION.ABORT_ROLLBACK(
    uname      => 'hr',
    orig_table => 'tst_rollback',
    int_table  => 'int_tst_rollback');
END;
/
```

親トピック: 表のオンライン再定義のロールバック

20.8.13 エラー後の表のオンライン再定義の終了およびクリーン・アップ

オンライン再定義プロセスを終了できます。これにより、再定義プロセスに対応付けられた一時ログおよび一時表が削除されます。このプロシージャをコールした後は、仮表とその依存オブジェクトを削除できます。

再定義プロセス中にエラーが発生したときにオンライン再定義プロセスを終了する場合、または手動で再定義プロセスを終了するには:

- ABORT_REDEF_TABLEプロシージャを実行します。

オンライン再定義プロセスの再起動が必要な場合は、最初にABORT_REDEF_TABLEをコールしないと、表を再定義する後続の試みでエラーが発生します。


ノート:

FINISH_REDEF_TABLE プロシージャがタイムアウトしたために再定義プロセスが停止した場合は、ABORT_REDEF_TABLE プロシージャをコールする必要はありません。タイムアウト間隔は、FINISH_REDEF_TABLE プロシージャの dml_lock_timeout パラメータで制御します。詳細は、「DBMS_REDEFINITION の複数のプロシージャを使用したオンライン再定義の実行」のステップ p 8 を参照してください

親トピック: 表のオンライン再定義

20.8.14 1つ以上のパーティションのオンライン再定義

表の1つ以上のパーティションをオンラインで再定義できます。これは、異なる表領域にパーティションを移動する際に、移動中でもパーティションに対してDMLを使用できるようにする場合などに便利です。

一度に表の複数のパーティションを再定義できます。その場合は、表の再定義プロセス中に複数の仮表が必要になります。表の再定義を完了するには、十分な空き領域とUNDO領域があることを確認してください。

複数のパーティションを再定義する場合、特定のパーティションでエラーが発生した場合にも再定義が続行されるように指定できます。そのためには、DBMS_REDEFINITIONパッケージの再定義プロシージャでcontinue_after_errorsパラメータをTRUEに設定します。DBA_REDEFINITION_STATUSビューをチェックして、再定義プロセス中にエラーが発生したかどうかを確認できます。このビューのSTATUS列には、各パーティションの再定義プロセスが成功したか失敗したかが表示されます。

リソース要件を低減するために、表全体を一度に1つのパーティションずつ再定義することもできます。たとえば、異なる表領域に非常に大きな表を移動するには、表を1度に1つのパーティションずつ移動することで、移動を完了するために必要な空き領域とUNDO領域を最小化できます。

パーティションの再定義は、次の点で表の再定義とは異なります。

- 依存オブジェクトをコピーする必要はありません。1つのパーティションを再定義する場合、COPY_TABLE_DEPENDENTSプロシージャの使用は有効ではありません。
- 仮表に対してローカル索引を手動で作成し、登録する必要があります。

  「依存オブジェクトの手動による作成」を参照してください。
- START_REDEF_TABLEには、NULLの列マッピング文字列が必要です。

ノート:

Oracle Database 12c からは、より簡単な ALTER TABLE...MOVE PARTITION ... ONLINE 文を使用して、表のオンライン再定義を使用せずにパーティションまたはサブパーティションをオンラインで移動できます。移動中のパーティションまたはサブパーティションに対して DML 操作を中断なく実行し続けることができます。「新規セグメントまたは表領域への表の移動」を参照してください。

- 単一パーティションのオンライン再定義のルール
  単一パーティションを再定義するための基本的な仕組みは、データベースのパーティション交換機能(ALTER TABLE...EXCHANGE PARTITION)です。

関連項目:

親トピック: 表のオンライン再定義

20.8.14.1 単一パーティションのオンライン再定義のルール

単一パーティションを再定義するための基本的な仕組みは、データベースのパーティション交換機能(ALTER TABLE...EXCHANGE PARTITION)です。

したがって、単一パーティションのオンライン定義のルールと制限事項は、この仕組みに基づいて決まります。一般的には、次の制限事項があります。

- 論理的な変更(列の追加や削除など)は許可されません。
- パーティション化する方法の変更(レンジ・パーティション化からハッシュ・パーティション化への変更など)は許可されません。

仮表を定義する際のルールは、次のとおりです。

- 再定義するパーティションが、レンジ、ハッシュまたはリスト・パーティションである場合は、非パーティションの仮表が必要です。
- 再定義するパーティションがレンジ-ハッシュ・コンポジット・パーティション表のレンジ・パーティションである場合は、ハッシュ・パーティション表の仮表が必要です。また、仮表のパーティション化キーは、レンジ-ハッシュ・パーティション表のサブパーティション化キーと同一であり、仮表のパーティション数は、再定義するレンジ・パーティションのサブパーティション数と同一であることが必要です。
- 再定義するパーティションがレンジ-リスト・コンポジット・パーティション表のレンジ・パーティションである場合は、リスト・パーティション表の仮表が必要です。また、仮表のパーティション化キーは、レンジ-リスト・パーティション表のサブパーティション化キーと同一であり、仮表のリスト・パーティションの値リストは、再定義するレンジ・パーティションのリスト・サブパーティションの値リストと正確に一致している必要があります。
- 仮表を圧縮するよう定義する場合、ROWIDによる再定義ではなく、キーによる再定義を使用する必要があります。

次の補足ルールは、再定義する表がパーティション化された索引構成表である場合に適用されます。

- 仮表も索引構成されている必要があります。
- 元の表と仮表では、同じ列に対する主キーが同じ順序で保持されている必要があります。
- 接頭辞圧縮が使用可能な場合は、元の表と仮表の両方で、同じ接頭辞の長さで接頭辞圧縮が使用可能になっている必要があります。
- オーバーフロー・セグメントがある場合は、元の表と仮表の両方にあるか、または両方にないことが必要です。マッピング表についても同様です。

関連項目:

- 『Oracle Database VLDBおよびパーティショニング・ガイド』のパーティションの交換に関する項
- パーティションを使用して表を再定義する例は、「表のオンライン再定義の例」を参照してください

親トピック: 1つ以上のパーティションのオンライン再定義

20.8.15 表のオンライン再定義の例

例を使用して表のオンライン再定義を説明します。

次の各例について、すべてのDBMS_REDEFINITIONサブプログラムの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

例	説明
例1	REDEF_TABLE プロシージャを使用して、表の記憶域プロパティを 1 つのステップで再定義します。
例2	新しい列を追加しパーティションを追加することで、表を再定義します。
例3	オブジェクト・データ型を使用して表を再定義します。
例4	手動で登録した依存オブジェクトを使用して表を再定義します。
例5	複数のパーティションを異なる表領域に移動して再定義します。
例6	表のいずれの列のプロパティも変更せずに、仮想プライベート・データベース(VPD)ポリシーを含む表を再定義します。
例7	VPD ポリシーが指定された表を再定義し、表のいずれかの列のプロパティを変更します。
例8	オンライン再定義で複数の変更を実行して、表を再定義します。

例1

この例では、REDEF_TABLEプロシージャを使用した表の記憶域プロパティのオンライン再定義を示しています。

元の表(print_ads)は、次のようにpmスキーマで定義されています。

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
AD_ID                                               NUMBER(6)
AD_TEXT                                             CLOB
```

この表では、LOB列ad_textでBasicFiles LOB記憶域が使用されます。

表の索引は、次のSQL文を使用して作成されました。

```
CREATE INDEX pm.print_ads_ix
```

```
    ON print_ads (ad_id)
  TABLESPACE example;
```

表は次のようにして再定義します。

- 表は、高度な行圧縮を使用して圧縮します。
- 表の表領域をEXAMPLEからNEWTBSに変更します。この例では、NEWTBS表領域が存在しているものとします。
- 索引は、COMPRESS 1圧縮を使用して圧縮します。
- 索引の表領域をEXAMPLEからNEWIDXTBSに変更します。この例では、NEWIDXTBS表領域が存在しているものとします。
- 表のLOB列は、COMPRESS HIGH圧縮を使用して圧縮します。
- LOB列の表領域をEXAMPLEからNEWLOBTBSに変更します。この例では、NEWLOBTBS表領域が存在しているものとします。
- LOB列をSecureFiles LOB記憶域に変更します。

この再定義のステップは、次のとおりです。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. REDEF_TABLEプロシージャを実行します。

```
BEGIN
  DBMS_REDEFINITION.REDEF_TABLE(
    uname                      => 'PM',
    tname                      => 'PRINT_ADS',
    table_compression_type     => 'ROW STORE COMPRESS ADVANCED',
    table_part_tablespace      => 'NEWTBS',
    index_key_compression_type => 'COMPRESS 1',
    index_tablespace           => 'NEWIDXTBS',
    lob_compression_type       => 'COMPRESS HIGH',
    lob_tablespace             => 'NEWLOBTBS',
    lob_store_as               => 'SECUREFILE');
END;
/
```

ノート:

エラーが発生した場合は、仮表が削除され、REDEF_TABLE プロシージャを再実行する必要があります。

例2

この例は、新しい列を追加しパーティションを追加することによる表のオンライン再定義を示しています。

元の表(emp_redef)のhrスキーマでの定義は、次のとおりです。

```
Name      Type
--------- ----------------------------
EMPNO     NUMBER(5)    <- Primary key
ENAME     VARCHAR2(15)
```

```
JOB         VARCHAR2(10)
DEPTNO      NUMBER(3)
```

表は次のようにして再定義します。

- 新しい列mgr、hiredate、salおよびbonusを追加します。
- 新しい列bonusを0 (ゼロ)に初期化します。
- 列deptnoの値を10増やしています。
- 再定義された表をempnoの範囲でパーティション化します。

この再定義のステップは、次のとおりです。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 表がオンライン再定義の候補であることを確認します。この場合は、主キーまたは疑似主キーを使用して再定義が実行されるように指定します。

   ```
   BEGIN
     DBMS_REDEFINITION.CAN_REDEF_TABLE(
       uname        => 'hr',
       tname        =>'emp_redef',
       options_flag => DBMS_REDEFINITION.CONS_USE_PK);
   END;
   /
   ```

3. 仮表hr.int_emp_redefを作成します。

   ```
   CREATE TABLE hr.int_emp_redef
           (empno      NUMBER(5) PRIMARY KEY,
            ename      VARCHAR2(15) NOT NULL,
            job        VARCHAR2(10),
            mgr        NUMBER(5),
            hiredate   DATE DEFAULT (sysdate),
            sal        NUMBER(7,2),
            deptno     NUMBER(3) NOT NULL,
            bonus      NUMBER (7,2) DEFAULT(0))
        PARTITION BY RANGE(empno)
          (PARTITION emp1000 VALUES LESS THAN (1000) TABLESPACE admin_tbs,
           PARTITION emp2000 VALUES LESS THAN (2000) TABLESPACE admin_tbs2);
   ```

   指定した表領域が存在することを確認します。

4. 再定義プロセスを開始します。

   ```
   BEGIN
     DBMS_REDEFINITION.START_REDEF_TABLE(
       uname        => 'hr',
       orig_table   => 'emp_redef',
       int_table    => 'int_emp_redef',
       col_mapping  => 'empno empno, ename ename, job job, deptno+10 deptno,
                        0 bonus',
       options_flag => DBMS_REDEFINITION.CONS_USE_PK);
   END;
   /
   ```

5. 依存オブジェクトをコピーします。(hr.int_emp_redefに対するトリガー、索引、マテリアライズド・ビュー・ログ、権限付与および制約がある場合、それらは自動的に作成されます。)

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'hr',
    orig_table       => 'emp_redef',
    int_table        => 'int_emp_redef',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => TRUE,
    num_errors       => num_errors);
END;
/
```

このコールでは、ignore_errors引数がTRUEに設定されていることに注意してください。これは、仮表が主キー制約付きで作成されており、COPY_TABLE_DEPENDENTSによって、主キー制約と索引が元の表からコピーされる際にエラーが発生するためです。これらのエラーは無視できますが、後続のステップに記載されている問合せを実行して、他のエラーの存在を確認する必要があります。

6. DBA_REDEFINITION_ERRORSビューを問い合せて、エラーをチェックします。

```
SET LONG  8000
SET PAGES 8000
COLUMN OBJECT_NAME HEADING 'Object Name' FORMAT A20
COLUMN BASE_TABLE_NAME HEADING 'Base Table Name' FORMAT A10
COLUMN DDL_TXT HEADING 'DDL That Caused Error' FORMAT A40

SELECT OBJECT_NAME, BASE_TABLE_NAME, DDL_TXT FROM
         DBA_REDEFINITION_ERRORS;
Object Name          Base Table DDL That Caused Error
-------------------- ---------- ----------------------------------------
SYS_C006796          EMP_REDEF  CREATE UNIQUE INDEX "HR"."TMP$$_SYS_C006
                                7960" ON "HR"."INT_EMP_REDEF" ("EMPNO")
                                  PCTFREE 10 INITRANS 2 MAXTRANS 255
                                  STORAGE(INITIAL 65536 NEXT 1048576 MIN
                                EXTENTS 1 MAXEXTENTS 2147483645
                                  PCTINCREASE 0 FREELISTS 1 FREELIST GRO
                                UPS 1
                                  BUFFER_POOL DEFAULT)
                                  TABLESPACE "ADMIN_TBS"
SYS_C006794          EMP_REDEF  ALTER TABLE "HR"."INT_EMP_REDEF" MODIFY
                                ("ENAME" CONSTRAINT "TMP$$_SYS_C0067940"
                                 NOT NULL ENABLE NOVALIDATE)
SYS_C006795          EMP_REDEF  ALTER TABLE "HR"."INT_EMP_REDEF" MODIFY
                                ("DEPTNO" CONSTRAINT "TMP$$_SYS_C0067950
                                " NOT NULL ENABLE NOVALIDATE)
SYS_C006796          EMP_REDEF  ALTER TABLE "HR"."INT_EMP_REDEF" ADD CON
                                STRAINT "TMP$$_SYS_C0067960" PRIMARY KEY
                                 ("EMPNO")
                                  USING INDEX PCTFREE 10 INITRANS 2 MAXT
                                RANS 255
                                  STORAGE(INITIAL 65536 NEXT 1048576 MIN
                                EXTENTS 1 MAXEXTENTS 2147483645
                                  PCTINCREASE 0 FREELISTS 1 FREELIST GRO
                                UPS 1
                                  BUFFER_POOL DEFAULT)
                                  TABLESPACE "ADMIN_TBS"  ENABLE NOVALID
                                ATE
```

これらのエラーは、仮表にある既存の主キー制約に起因しているため、無視できます。このアプローチでは、再定義後の表の主キー制約名と索引名が変更されていることに注意してください。別のアプローチを使用すると、エラーの発生と名前の変更を回避できますが、仮表は主キー制約なしで定義されることになります。この例の場合、主キー制約と索引は元の表からコピーされます。

ノート:

最良のアプローチは、主キー制約付きで仮表を定義し、REGISTER_DEPENDENT_OBJECT を使用して主キー制約と索引を登録してから、COPY_TABLE_DEPENDENTS で残りの依存オブジェクトをコピーすることです。このアプローチでは、エラーが回避され、再定義した表には常に主キーがあり、依存オブジェクト名も変わりません。

7. (オプション)仮表hr.int_emp_redefを同期化します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'hr',
    orig_table => 'emp_redef',
    int_table  => 'int_emp_redef');
END;
/
```

8. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'hr',
    orig_table => 'emp_redef',
    int_table  => 'int_emp_redef');
END;
/
```

このステップが終了するまでに、わずかな間のみ、表hr.emp_redefが排他モードでロックされます。このコールの後、表hr.emp_redefはhr.int_emp_redef表のすべての属性を持つように再定義されます。

このプロシージャのdml_lock_timeoutパラメータにNULL以外の値を指定することを検討してください。詳細は、「DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行」のステップp 8を参照してください。

9. 仮表に対する長時間実行の問合せがある場合は、完了するのを待ってから、仮表を削除します。

例3

この例では、列をオブジェクト属性に変更するために表を再定義します。再定義した表にオブジェクト型の新しい列を確保します。

元の表(customer)の定義は、次のとおりです。

```
Name         Type
------------ -------------
CID          NUMBER            <- Primary key
NAME         VARCHAR2(30)
STREET       VARCHAR2(100)
CITY         VARCHAR2(30)
STATE        VARCHAR2(2)
ZIP          NUMBER(5)
```

新しいオブジェクトの型定義は、次のとおりです。

```
CREATE TYPE addr_t AS OBJECT (
   street VARCHAR2(100),
   city VARCHAR2(30),
   state VARCHAR2(2),
   zip NUMBER(5, 0) );
/
```

再定義のステップは、次のとおりです。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 表がオンライン再定義の候補であることを確認します。主キーまたは疑似主キーを使用して再定義が実行されるように指定します。

   ```
   BEGIN
     DBMS_REDEFINITION.CAN_REDEF_TABLE(
       uname        => 'steve',
       tname        =>'customer',
       options_flag => DBMS_REDEFINITION.CONS_USE_PK);
   END;
   /
   ```

3. 仮表int_customerを作成します。

   ```
   CREATE TABLE int_customer(
     CID   NUMBER,
     NAME  VARCHAR2(30),
     ADDR  addr_t);
   ```

   仮表には主キーが定義されていないことに注意してください。ステップ6で依存オブジェクトがコピーされると、主キー制約と索引がコピーされます。

4. customerは大きい表であるため、後続のステップのためにパラレル操作を指定します。

   ```
   ALTER SESSION FORCE PARALLEL DML PARALLEL 4;
   ALTER SESSION FORCE PARALLEL QUERY PARALLEL 4;
   ```

5. 主キーを使用して再定義プロセスを開始します。

   ```
   BEGIN
     DBMS_REDEFINITION.START_REDEF_TABLE(
      uname       => 'steve',
      orig_table  => 'customer',
      int_table   => 'int_customer',
      col_mapping => 'cid cid, name name,
         addr_t(street, city, state, zip) addr');
   END;
   /
   ```

   addr_t(street, city, state, zip)は、オブジェクト・コンストラクタへのコールです。

6. 依存オブジェクトをコピーします。

   ```
   DECLARE
   num_errors PLS_INTEGER;
   BEGIN
     DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
   ```

```
    uname            => 'steve',
    orig_table       => 'customer',
    int_table        => 'int_customer',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => FALSE,
    num_errors       => num_errors,
    copy_statistics  => TRUE);
END;
/
```

このコールの最後の引数は、表の統計が仮表にコピーされることを意味します。

7. 必要に応じて、仮表を同期化します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'steve',
    orig_table => 'customer',
    int_table  => 'int_customer');
END;
/
```

8. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'steve',
    orig_table => 'customer',
    int_table  => 'int_customer');
END;
/
```

このプロシージャのdml_lock_timeoutパラメータにNULL以外の値を指定することを検討してください。詳細は、「DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行」のステップp 8を参照してください。

9. 仮表に対する長時間実行の問合せがある場合は、完了するのを待ってから、仮表を削除します。

例4

この例では、依存オブジェクトを手動で作成および登録する必要がある場合を考えてみます。

再定義する表の定義は、次のとおりです。

```
CREATE TABLE steve.t1
  (c1 NUMBER);
```

表には列c1の索引があります。

```
CREATE INDEX steve.index1 ON steve.t1(c1);
```

再定義後に、列c1が列c2になる例について考えてみます。この場合、COPY_TABLE_DEPENDENTSは、index1に対応して、仮表に対する索引の作成を試行し、仮表には存在しない列c1に対して索引の作成を試行します。これは結果的にエラーとなります。したがって、列c2に対しては、索引を手動で作成して登録する必要があります。

再定義のステップは、次のとおりです。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. CAN_REDEF_TABLEを使用してt1がオンライン定義の候補であることを確認し、次にSTART_REDEF_TABLEを使用して再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.CAN_REDEF_TABLE(
   uname        => 'steve',
   tname        => 't1',
   options_flag => DBMS_REDEFINITION.CONS_USE_ROWID);
END;
/
```

3. 仮表int_t1を作成し、列c2に対して索引int_index1を作成します。

```
CREATE TABLE steve.int_t1
  (c2 NUMBER);
CREATE INDEX steve.int_index1 ON steve.int_t1(c2);
```

4. 再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE(
    uname        => 'steve',
    orig_table   => 't1',
    int_table    => 'int_t1',
    col_mapping  => 'c1 c2',
    options_flag => DBMS_REDEFINITION.CONS_USE_ROWID);
END;
/
```

5. 元(index1)と仮(int_index1)の依存オブジェクトを登録します。

```
BEGIN
 DBMS_REDEFINITION.REGISTER_DEPENDENT_OBJECT(
   uname         => 'steve',
   orig_table    => 't1',
   int_table     => 'int_t1',
   dep_type      => DBMS_REDEFINITION.CONS_INDEX,
   dep_owner     => 'steve',
   dep_orig_name => 'index1',
   dep_int_name  => 'int_index1');
END;
/
```

6. 依存オブジェクトをコピーします。

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'steve',
    orig_table       => 't1',
    int_table        => 'int_t1',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => TRUE,
    num_errors       => num_errors);
END;
```

```
/
```

7. 必要に応じて、仮表を同期化します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'steve',
    orig_table => 't1',
    int_table  => 'int_t1');
END;
/
```

8. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'steve',
    orig_table => 't1',
    int_table  => 'int_t1');
END;
/
```

9. 仮表に対する長時間実行の問合せがある場合は、完了するのを待ってから、仮表を削除します。

例5

この例では、複数のパーティションの再定義を示します。レンジ・パーティション化されたsales tableの2つのパーティションを新しい表領域に移動します。再定義するパーティションが含まれている表は、次のようにして定義します。

```
CREATE TABLE steve.salestable
  (s_productid NUMBER,
  s_saledate DATE,
  s_custid NUMBER,
  s_totalprice NUMBER)
  TABLESPACE users
  PARTITION BY RANGE(s_saledate)
  (PARTITION sal10q1 VALUES LESS THAN (TO_DATE('01-APR-2010', 'DD-MON-YYYY')),
  PARTITION sal10q2 VALUES LESS THAN (TO_DATE('01-JUL-2010', 'DD-MON-YYYY')),
  PARTITION sal10q3 VALUES LESS THAN (TO_DATE('01-OCT-2010', 'DD-MON-YYYY')),
  PARTITION sal10q4 VALUES LESS THAN (TO_DATE('01-JAN-2011', 'DD-MON-YYYY')));
```

この例では、sal10q1パーティションをsales1表領域に移動し、sal10q2パーティションをsales2表領域に移動します。sal10q3およびsal10q4パーティションは移動しません。

パーティションを移動するには、表領域sales1およびsales2が存在している必要があります。次の例では、これらの表領域を作成します。

```
CREATE TABLESPACE sales1 DATAFILE '/u02/oracle/data/sales01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

CREATE TABLESPACE sales2 DATAFILE '/u02/oracle/data/sales02.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

ノート:

この操作は、2 つの ALTER TABLE ... MOVE PARTITION ... ONLINE 文を実行することによって行うこともできます。「新規セグメントまたは表領域への表の移動」を参照してください。

表には、次のように定義されたローカル・パーティション索引があります。

```
CREATE INDEX steve.sales_index ON steve.salestable
   (s_saledate, s_productid, s_custid) LOCAL;
```

ステップは、次のとおりです。次のプロシージャ・コールでは、パーティション名(part_name)という特別な引数に注目してください。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. salestableが再定義の候補であることを確認します。

   ```
   BEGIN
     DBMS_REDEFINITION.CAN_REDEF_TABLE(
      uname        => 'steve',
      tname        => 'salestable',
      options_flag => DBMS_REDEFINITION.CONS_USE_ROWID,
      part_name    => 'sal10q1, sal10q2');
   END;
   /
   ```

3. 新しい表領域に仮表を作成します。これはレンジ・パーティションの再定義であるため、仮表は非パーティション表です。

   ```
   CREATE TABLE steve.int_salestb1
     (s_productid NUMBER,
     s_saledate DATE,
     s_custid NUMBER,
     s_totalprice NUMBER)
     TABLESPACE sales1;
   CREATE TABLE steve.int_salestb2
     (s_productid NUMBER,
     s_saledate DATE,
     s_custid NUMBER,
     s_totalprice NUMBER)
     TABLESPACE sales2;
   ```

4. ROWIDを使用して再定義プロセスを開始します。

   ```
   BEGIN
     DBMS_REDEFINITION.START_REDEF_TABLE(
      uname        => 'steve',
      orig_table   => 'salestable',
      int_table    => 'int_salestb1, int_salestb2',
      col_mapping  => NULL,
      options_flag => DBMS_REDEFINITION.CONS_USE_ROWID,
      part_name    => 'sal10q1, sal10q2',
      continue_after_errors => TRUE);
   END;
   /
   ```

   part_nameパラメータでは両方のパーティションを指定し、int_tableパラメータでは各パーティションの仮表を指定していることに注意してください。また、特定のパーティションでエラーが発生した場合にも再定義プロセスが続行されるように、continue_after_errorsパラメータがTRUEに設定されています。

5. 仮表に対してローカル索引を手動で作成します。

   ```
   CREATE INDEX steve.int_sales1_index ON steve.int_salestb1
   (s_saledate, s_productid, s_custid)
   TABLESPACE sales1;
   CREATE INDEX steve.int_sales2_index ON steve.int_salestb2
   (s_saledate, s_productid, s_custid)
   TABLESPACE sales2;
   ```

6. 必要に応じて、仮表を同期化します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
   uname      => 'steve',
   orig_table => 'salestable',
   int_table  => 'int_salestb1, int_salestb2',
   part_name    => 'sal10q1, sal10q2',
   continue_after_errors => TRUE);
END;
/
```

7. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
   uname      => 'steve',
   orig_table => 'salestable',
   int_table  => 'int_salestb1, int_salestb2',
   part_name    => 'sal10q1, sal10q2',
   continue_after_errors => TRUE);
END;
/
```

   このプロシージャのdml_lock_timeoutパラメータにNULL以外の値を指定することを検討してください。詳細は、「DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行」のステップp 8を参照してください。

8. 仮表に対する長時間実行の問合せがある場合は、完了するのを待ってから、仮表を削除します。

9. (オプション) DBA_REDEFINITION_STATUSビューを問い合せて、各パーティションの再定義が成功したことを確認します。

```
SELECT BASE_TABLE_OWNER, BASE_TABLE_NAME, OPERATION, STATUS
  FROM DBA_REDEFINITION_STATUS;
```

   いずれかのパーティションの再定義が失敗した場合は、DBA_REDEFINITION_ERRORSビューを問い合せて失敗の原因を確認します。失敗の原因となった状況を修正し、オンライン再定義を再度実行します。

次の問合せは、表内の2つのパーティションが新しい表領域に移動されたことを示しています。

```
SELECT PARTITION_NAME, TABLESPACE_NAME FROM DBA_TAB_PARTITIONS
 WHERE TABLE_NAME = 'SALESTABLE';

PARTITION_NAME                 TABLESPACE_NAME
------------------------------ ------------------------------
SAL10Q1                        SALES1
SAL10Q2                        SALES2
SAL10Q3                        USERS
SAL10Q4                        USERS

4 rows selected.
```

例6

この例は、仮想プライベート・データベース(VPD)ポリシーが指定された表のオンライン再定義を示しています。この例では、表の列名および列の型を変更しないで、表のすべてのトリガーを無効にします。

再定義する表の定義は、次のとおりです。

```
CREATE TABLE hr.employees(
   employee_id     NUMBER(6)  PRIMARY KEY,
```

```
   first_name     VARCHAR2(20),
   last_name      VARCHAR2(25)
                  CONSTRAINT     emp_last_name_nn  NOT NULL,
   email          VARCHAR2(25)
                  CONSTRAINT     emp_email_nn  NOT NULL,
   phone_number   VARCHAR2(20),
   hire_date      DATE
                  CONSTRAINT     emp_hire_date_nn  NOT NULL,
   job_id         VARCHAR2(10)
                  CONSTRAINT     emp_job_nn  NOT NULL,
   salary         NUMBER(8,2),
   commission_pct NUMBER(2,2),
   manager_id     NUMBER(6),
   department_id  NUMBER(4),
                  CONSTRAINT     emp_salary_min
                     CHECK (salary > 0),
                  CONSTRAINT     emp_email_uk
                     UNIQUE (email));
```

HRサンプル・スキーマをインストールした場合は、データベースにこの表が存在します。

次のauth_emp_dep_100ファンクションがVPDポリシーに対して作成されるとします。

```
CREATE OR REPLACE FUNCTION hr.auth_emp_dep_100(
  schema_var IN VARCHAR2,
  table_var IN VARCHAR2
 )
 RETURN VARCHAR2
 AS
  return_val VARCHAR2 (400);
  unm        VARCHAR2(30);
 BEGIN
  SELECT USER INTO unm FROM DUAL;
  IF (unm = 'HR') THEN
  return_val := NULL;
 ELSE
  return_val := 'DEPARTMENT_ID = 100';
  END IF;
 RETURN return_val;
END auth_emp_dep_100;
/
```

次のADD_POLICYプロシージャでは、auth_emp_dep_100ファンクションを使用して、元の表hr.employeesのVPDポリシーを指定しています。

```
BEGIN
  DBMS_RLS.ADD_POLICY (
    object_schema    => 'hr',
    object_name      => 'employees',
    policy_name      => 'employees_policy',
    function_schema  => 'hr',
    policy_function  => 'auth_emp_dep_100',
    statement_types  => 'select, insert, update, delete'
   );
 END;
/
```

この例では、hr.employees表を再定義して、そのすべてのトリガーを無効にします。再定義中に列名や列の型は変更されません。したがって、START_REFEF_TABLEプロシージャのcopy_vpd_optにDBMS_REDEFINITION.CONS_VPD_AUTOを指定します。

この再定義のステップは、次のとおりです。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限およびVPDポリシーの管理に必要な権限のあるユーザーと

して接続します。

特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限、およびDBMS_RLSパッケージに対するEXECUTE権限を持っている必要があります。

「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 表がオンライン再定義の候補であることを確認します。この場合は、主キーまたは疑似主キーを使用して再定義が実行されるように指定します。

```
BEGIN
  DBMS_REDEFINITION.CAN_REDEF_TABLE('hr','employees',
      DBMS_REDEFINITION.CONS_USE_PK);
END;
/
```

3. 仮表hr.int_employeesを作成します。

```
CREATE TABLE hr.int_employees(
   employee_id    NUMBER(6),
   first_name     VARCHAR2(20),
   last_name      VARCHAR2(25),
   email          VARCHAR2(25),
   phone_number   VARCHAR2(20),
   hire_date      DATE,
   job_id         VARCHAR2(10),
   salary         NUMBER(8,2),
   commission_pct NUMBER(2,2),
   manager_id     NUMBER(6),
   department_id  NUMBER(4));
```

4. 再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE (
    uname          => 'hr',
    orig_table     => 'employees',
    int_table      => 'int_employees',
    col_mapping    => NULL,
    options_flag   => DBMS_REDEFINITION.CONS_USE_PK,
    orderby_cols   => NULL,
    part_name      => NULL,
    copy_vpd_opt   => DBMS_REDEFINITION.CONS_VPD_AUTO);
END;
/
```

copy_vpd_optパラメータがDBMS_REDEFINITION.CONS_VPD_AUTOに設定されている場合、表の所有者およびオンライン再定義を開始したユーザーのみがオンライン再定義中に仮表にアクセスできます。

また、col_mappingパラメータがNULLに設定されていることにも注意してください。copy_vpd_optパラメータがDBMS_REDEFINITION.CONS_VPD_AUTOに設定されている場合、col_mappingパラメータはNULLまたは'*'であることが必要です。「オンライン再定義時の仮想プライベート・データベース(VPD)ポリシーの処理」を参照してください。

5. 依存オブジェクトをコピーします。(hr.int_employeesに対するトリガー、索引、マテリアライズド・ビュー・ログ、権限付与および制約がある場合、それらは自動的に作成されます。)

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'hr',
```

```
    orig_table       => 'employees',
    int_table        => 'int_employees',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => FALSE,
    num_errors       => num_errors);
END;
/
```

6. 仮表に対するすべてのトリガーを無効にします。

```
ALTER TABLE hr.int_employees
  DISABLE ALL TRIGGERS;
```

7. (オプション)仮表hr.int_employeesを同期化します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'hr',
    orig_table => 'employees',
    int_table  => 'int_employees');
END;
/
```

8. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'hr',
    orig_table => 'employees',
    int_table  => 'int_employees');
END;
/
```

このステップが終了するまでに、わずかな間のみ、表hr.employeesが排他モードでロックされます。このコールの後、表hr.employeesはhr.int_employees表のすべての属性を持つように再定義されます。

このプロシージャのdml_lock_timeoutパラメータにNULL以外の値を指定することを検討してください。詳細は、「DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行」のステップp 8を参照してください。

9. 仮表に対する長時間実行の問合せがある場合は、完了するのを待ってから、仮表を削除します。

例7

この例は、仮想プライベート・データベース(VPD)ポリシーが指定された表のオンライン再定義を示しています。この例では、表の列名を変更します。

再定義する表の定義は、次のとおりです。

```
CREATE TABLE oe.orders(
   order_id      NUMBER(12)  PRIMARY KEY,
   order_date    TIMESTAMP WITH LOCAL TIME ZONE CONSTRAINT order_date_nn NOT NULL,
   order_mode    VARCHAR2(8),
   customer_id   NUMBER(6) CONSTRAINT order_customer_id_nn NOT NULL,
   order_status  NUMBER(2),
   order_total   NUMBER(8,2),
   sales_rep_id  NUMBER(6),
   promotion_id  NUMBER(6),
   CONSTRAINT    order_mode_lov
                 CHECK (order_mode in ('direct','online')),
```

```
   CONSTRAINT    order_total_min
                 check (order_total >= 0));
```

OEサンプル・スキーマをインストールした場合は、データベースにこの表が存在します。

次のauth_ordersファンクションがVPDポリシーに対して作成されるとします。

```
CREATE OR REPLACE FUNCTION oe.auth_orders(
  schema_var IN VARCHAR2,
  table_var IN VARCHAR2
 )
 RETURN VARCHAR2
 AS
  return_val VARCHAR2 (400);
  unm        VARCHAR2(30);
 BEGIN
  SELECT USER INTO unm FROM DUAL;
  IF (unm = 'OE') THEN
  return_val := NULL;
 ELSE
  return_val := 'SALES_REP_ID = 159';
  END IF;
 RETURN return_val;
END auth_orders;
/
```

次のADD_POLICYプロシージャでは、auth_ordersファンクションを使用して、元の表oe.ordersのVPDポリシーを指定しています。

```
BEGIN
  DBMS_RLS.ADD_POLICY (
    object_schema    => 'oe',
    object_name      => 'orders',
    policy_name      => 'orders_policy',
    function_schema  => 'oe',
    policy_function  => 'auth_orders',
    statement_types  => 'select, insert, update, delete');
 END;
/
```

この例では、表を再定義してsales_rep_id列をsale_pidに変更します。再定義中に1つ以上の列名または列の型を変更する場合、START_REFEF_TABLEプロシージャのcopy_vpd_optに DBMS_REDEFINITION.CONS_VPD_MANUALを指定する必要があります。

この再定義のステップは、次のとおりです。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限およびVPDポリシーの管理に必要な権限のあるユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限、およびDBMS_RLSパッケージに対するEXECUTE権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 表がオンライン再定義の候補であることを確認します。この場合は、主キーまたは疑似主キーを使用して再定義が実行されるように指定します。

   ```
   BEGIN
     DBMS_REDEFINITION.CAN_REDEF_TABLE(
       uname        => 'oe',
       tname        => 'orders',
       options_flag => DBMS_REDEFINITION.CONS_USE_PK);
   ```

```
END;
/
```

3. 仮表oe.int_ordersを作成します。

```
CREATE TABLE oe.int_orders(
   order_id      NUMBER(12),
   order_date    TIMESTAMP WITH LOCAL TIME ZONE,
   order_mode    VARCHAR2(8),
   customer_id   NUMBER(6),
   order_status  NUMBER(2),
   order_total   NUMBER(8,2),
   sales_pid     NUMBER(6),
   promotion_id  NUMBER(6));
```

仮表ではsales_rep_id列がsales_pid列に変更されていることに注意してください。

4. 再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE (
    uname          => 'oe',
    orig_table     => 'orders',
    int_table      => 'int_orders',
    col_mapping    => 'order_id order_id, order_date order_date, order_mode
                       order_mode, customer_id customer_id, order_status
                       order_status, order_total order_total, sales_rep_id
                       sales_pid, promotion_id promotion_id',
    options_flag   => DBMS_REDEFINITION.CONS_USE_PK,
    orderby_cols   => NULL,
    part_name      => NULL,
    copy_vpd_opt   => DBMS_REDEFINITION.CONS_VPD_MANUAL);
END;
/
```

元の表と仮表で列名が異なるため、copy_vpd_optパラメータに
DBMS_REDEFINITION.CONS_VPD_MANUALを指定する必要があります。「オンライン再定義時の仮想プライベート・データベース(VPD)ポリシーの処理」を参照してください。

5. 仮表に対するVPDポリシーを作成します。

この例では、次のステップを実行します。

a. sales_rep_id列のかわりにsales_pid列を指定するVPDポリシーのために、
auth_orders_sales_pidという新しいファンクションを作成します。

```
CREATE OR REPLACE FUNCTION oe.auth_orders_sales_pid(
  schema_var IN VARCHAR2,
  table_var IN VARCHAR2
 )
 RETURN VARCHAR2
 AS
  return_val VARCHAR2 (400);
  unm        VARCHAR2(30);
 BEGIN
  SELECT USER INTO unm FROM DUAL;
  IF (unm = 'OE') THEN
  return_val := NULL;
 ELSE
  return_val := 'SALES_PID = 159';
  END IF;
 RETURN return_val;
END auth_orders_sales_pid;
/
```

b. ADD_POLICYプロシージャを実行し、新しいファンクションauth_orders_sales_pidおよび仮表int_ordersを指定します。

```
BEGIN
  DBMS_RLS.ADD_POLICY (
    object_schema    => 'oe',
    object_name      => 'int_orders',
    policy_name      => 'orders_policy',
    function_schema  => 'oe',
    policy_function  => 'auth_orders_sales_pid',
    statement_types  => 'select, insert, update, delete');
 END;
/
```

6. 依存オブジェクトをコピーします。(oe.int_ordersに対するトリガー、索引、マテリアライズド・ビュー・ログ、権限付与および制約がある場合、それらは自動的に作成されます。)

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'oe',
    orig_table       => 'orders',
    int_table        => 'int_orders',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
    copy_privileges  => TRUE,
    ignore_errors    => TRUE,
    num_errors       => num_errors);
END;
/
```

このコールでは、ignore_errors引数がTRUEに設定されていることに注意してください。これは、元の表にはsales_rep_id列に関連する索引および制約があり、仮表ではこの列がsales_pidに変更されるためです。次のステップではエラーを示し、仮表に対して索引と制約を作成する方法を説明します。

7. DBA_REDEFINITION_ERRORSビューを問い合せて、エラーをチェックします。

```
SET LONG  8000
SET PAGES 8000
COLUMN OBJECT_NAME HEADING 'Object Name' FORMAT A20
COLUMN BASE_TABLE_NAME HEADING 'Base Table Name' FORMAT A10
COLUMN DDL_TXT HEADING 'DDL That Caused Error' FORMAT A40

SELECT OBJECT_NAME, BASE_TABLE_NAME, DDL_TXT FROM
         DBA_REDEFINITION_ERRORS;
Object Name          Base Table DDL That Caused Error
-------------------- ---------- ----------------------------------------
ORDERS_SALES_REP_FK  ORDERS     ALTER TABLE "OE"."INT_ORDERS" ADD CONSTR
                                AINT "TMP$$_ORDERS_SALES_REP_FK1" FOREIG
                                N KEY ("SALES_REP_ID")
                                          REFERENCES "HR"."EMPLOYEES"
                                ("EMPLOYE
                                E_ID") ON DELETE SET NULL DISABLE
ORD_SALES_REP_IX     ORDERS     CREATE INDEX "OE"."TMP$$_ORD_SALES_REP_I
                                X0" ON "OE"."INT_ORDERS" ("SALES_REP_ID"
                                )
                                  PCTFREE 10 INITRANS 2 MAXTRANS 255 COM
                                PUTE STATISTICS
                                  STORAGE(INITIAL 65536 NEXT 1048576 MIN
                                EXTENTS 1 MAXEXTENTS 2147483645
                                  PCTINCREASE 0 FREELISTS 1 FREELIST GRO
                                UPS 1
```

```
                                    BUFFER_POOL DEFAULT)
                                    TABLESPACE "EXAMPLE"
TMP$$_ORDERS_SALES_R ORDERS       ALTER TABLE "OE"."INT_ORDERS" ADD CONSTR
EP_FK0                            AINT "TMP$$_TMP$$_ORDERS_SALES_RE0" FORE
                                  IGN KEY ("SALES_REP_ID")
                                            REFERENCES "HR"."INT_EMPLOYEES"
                                  ("EMP
                                  LOYEE_ID") ON DELETE SET NULL DISABLE
```

必要に応じて、出力でレポートされたエラーを修正します。

この例では、元の表にsales_rep_id列に対する索引および外部キー制約があります。列名がsales_rep_idからsales_pidに変更されたため、索引と制約を仮表にコピーできませんでした。

問題を修正するには、次のステップを実行して、仮表に対する索引と制約を追加します。

a. 索引を追加します。

```
ALTER TABLE oe.int_orders
  ADD (CONSTRAINT orders_sales_pid_fk
       FOREIGN KEY (sales_pid)
       REFERENCES hr.employees(employee_id)
       ON DELETE SET NULL);
```

b. 外部キー制約を追加します。

```
CREATE INDEX ord_sales_pid_ix ON oe.int_orders (sales_pid);
```

8. (オプション)仮表oe.int_ordersを同期化します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname      => 'oe',
    orig_table => 'orders',
    int_table  => 'int_orders');
END;
/
```

9. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname      => 'oe',
    orig_table => 'orders',
    int_table  => 'int_orders');
END;
/
```

このステップが終了するまでに、わずかな間のみ、表oe.ordersが排他モードでロックされます。このコールの後、表oe.ordersはoe.int_orders表のすべての属性を持つように再定義されます。

このプロシージャのdml_lock_timeoutパラメータにNULL以外の値を指定することを検討してください。詳細は、「DBMS_REDEFINITIONの複数のプロシージャを使用したオンライン再定義の実行」のステップp 8を参照してください。

10. 仮表に対する長時間実行の問合せがある場合は、完了するのを待ってから、仮表を削除します。

例8

この例では、オンライン再定義を使用した表の複数の変更について説明します。

再定義する表の定義は、次のとおりです。

```
CREATE TABLE testredef.original(
   col1 NUMBER PRIMARY KEY,
   col2 VARCHAR2(10),
   col3 CLOB,
   col4 DATE)
ORGANIZATION INDEX;
```

表は次のようにして再定義します。

- 表は、高度な行圧縮を使用して圧縮します。
- LOB列をSecureFiles LOB記憶域に変更します。
- 表の表領域をexampleからtestredeftbsに変更し、表のブロック・サイズを8KBから16KBに変更します。

  この例では、データベースのブロック・サイズを8KBとしています。また、この例では、DB_16K_CACHE_SIZE初期化パラメータが設定され、testredef表領域が16KBのブロック・サイズで作成されていると仮定しています。たとえば:

  ```
  CREATE TABLESPACE testredeftbs
    DATAFILE '/u01/app/oracle/oradata/testredef01.dbf' SIZE 500M   EXTENT
  MANAGEMENT LOCAL AUTOALLOCATE
    SEGMENT SPACE MANAGEMENT AUTO
    BLOCKSIZE 16384;
  ```

- 表はcol1列でパーティション化します。
- col5列を追加します。
- col2列を削除します。
- 列col3およびcol4の名前を変更し、表におけるその位置を変更します。
- col3列のタイプをDATEからTIMESTAMPに変更します。
- 表を索引構成表(IOT)からヒープ構成表に変更します。
- 表の断片化を解消します。

  断片化の解消を示すには、表にデータが移入されている必要があります。この例の目的のために、次のPL/SQLブロックを使用して表にデータを移入できます。

  ```
  DECLARE
    V_CLOB CLOB;
  BEGIN
     FOR I IN 0..999 LOOP
        V_CLOB := NULL;
        FOR J IN 1..1000 LOOP
           V_CLOB := V_CLOB||TO_CHAR(I,'0000');
        END LOOP;
        INSERT INTO testredef.original VALUES(I,TO_CHAR(I),V_CLOB,SYSDATE+I);
        COMMIT;
     END LOOP;
     COMMIT;
  END;
  /
  ```

  次のSQL文を実行し、3番目の行を削除して表を断片化します。

  ```
  DELETE FROM testredef.original WHERE (COL1/3) <> TRUNC(COL1/3);
  ```

  DBMS_SPACE.SPACE_USAGEプロシージャを使用して断片化を確認できます。

関連項目:

DBMS_SPACE.SPACE_USAGEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

この再定義のステップは、次のとおりです。

1. SQL*Plusで、表のオンライン再定義の実行に必要な権限を持つユーザーとして接続します。

   特に、ユーザーは「DBMS_REDEFINITIONパッケージに必要な権限」に記載されている権限を持っている必要があります。

   「SQL*Plusを使用したデータベースへの接続」を参照してください。

2. 表がオンライン再定義の候補であることを確認します。この場合は、主キーまたは疑似主キーを使用して再定義が実行されるように指定します。

```
BEGIN
  DBMS_REDEFINITION.CAN_REDEF_TABLE(
    uname        => 'testredef',
    tname        => 'original',
    options_flag => DBMS_REDEFINITION.CONS_USE_PK);
END;
/
```

3. 仮表testredef.interimを作成します。

```
CREATE TABLE testredef.interim(
    col1 NUMBER,
    col3 TIMESTAMP,
    col4 CLOB,
    col5 VARCHAR2(3))
    LOB(col4) STORE AS SECUREFILE (NOCACHE FILESYSTEM_LIKE_LOGGING)
    PARTITION BY RANGE (COL1) (
       PARTITION par1 VALUES LESS THAN (333),
       PARTITION par2 VALUES LESS THAN (666),
       PARTITION par3 VALUES LESS THAN (MAXVALUE))
   TABLESPACE testredeftbs
   ROW STORE COMPRESS ADVANCED;
```

4. 再定義プロセスを開始します。

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE(
     uname        => 'testredef',
     orig_table   => 'original',
     int_table    => 'interim',
     col_mapping  => 'col1 col1, TO_TIMESTAMP(col4) col3, col3 col4',
     options_flag => DBMS_REDEFINITION.CONS_USE_PK);
END;
/
```

5. 依存オブジェクトをコピーします。

```
DECLARE
num_errors PLS_INTEGER;
BEGIN
  DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
    uname            => 'testredef',
    orig_table       => 'original',
    int_table        => 'interim',
    copy_indexes     => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
    copy_triggers    => TRUE,
    copy_constraints => TRUE,
```

```
    copy_privileges  => TRUE,
    ignore_errors    => TRUE,
    num_errors       => num_errors);
END;
/
```

6. 必要に応じて、仮表を同期化します。

```
BEGIN
  DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
    uname       => 'testredef',
    orig_table  => 'original',
    int_table   => 'interim');
END;
/
```

7. 再定義を完了します。

```
BEGIN
  DBMS_REDEFINITION.FINISH_REDEF_TABLE(
    uname       => 'testredef',
    orig_table  => 'original',
    int_table   => 'interim');
END;
/
```

関連項目:

Oracle Databaseサンプル・スキーマ

親トピック: 表のオンライン再定義

20.9 エラーが発生した表の変更の調査と取消し

表に対してエラーが発生する変更を調査して取り消せるようにするために、Oracle Databaseには、データベース・オブジェクトの過去の状態を表示したり、Point-in-Timeメディア・リカバリを使用せずにデータベース・オブジェクトを以前の状態に戻すために使用できる一連の機能が用意されています。これらの機能はOracle Flashback機能と呼ばれます。

エラーが発生する変更を調査するために、複数のOracle Flashback問合せを使用して、特定の時点における行データを表示できます。さらに効率的な方法として、Oracle Flashback Version Queryを使用して、ある期間にわたる行への変更すべてを表示できます。この機能では、SELECT文にVERSIONS句を追加できるため、行の値への変更を表示するシステム変更番号(SCN)またはタイムスタンプの範囲を指定できます。この問合せでは、変更の原因となったトランザクションなど、関連するメタデータを返すこともできます。

エラーが発生するトランザクションを特定した後、Oracle Flashback Transaction Queryを使用して、そのトランザクションで実行された他の変更を特定できます。次に、Oracle Flashback Transactionを使用して、エラーが発生するトランザクションを取り消すことができます。(Oracle Flashback Transactionでは、依存するすべてのトランザクション、つまりエラーが発生するトランザクションと同じ行が関係する後続のトランザクションも取り消す必要があることに注意してください。)「Oracle Flashback Tableを使用した表のリカバリ」で説明されているOracle Flashback Tableも使用できます。

ノート:

Oracle Flashback 機能を使用するには、自動 UNDO 管理を使用している必要があります。「自動 UNDO

管理の概要」を参照してください。

関連項目:

Oracleフラッシュバック機能の詳細は、『Oracle Database開発ガイド』を参照してください。

親トピック: 表の管理

20.10 Oracle Flashback Tableを使用した表のリカバリ

Oracle Flashback Tableを使用すると、表を以前の時点の状態にリストアできます。

この機能では、ユーザーまたはアプリケーションにより、誤って変更または削除された表のリカバリを行うための、迅速なオンラインによる解決方法が提供されています。多くの場合、Oracle Flashback Tableを使用することで、管理者がより複雑なポイント・イン・タイム・リカバリ操作を実行する必要はなくなります。

Oracle Flashback Table:

- 指定された表のすべてのデータが、タイムスタンプまたはSCNで表された以前の時点にリストアされます。
- リストア操作はオンラインで実行されます。
- アプリケーションがフラッシュバックされた表を使用して機能するために必要な索引、トリガー、制約など、表の属性すべてが自動的に維持されます。
- 分散環境におけるすべてのリモート状態が維持されます。たとえば、レプリケート表がフラッシュバックされる場合は、レプリケーションに必要な表の変更すべてが維持されます。
- 制約によって指定されているデータの整合性が維持されます。表は、表の制約すべてに違反していないことを条件にしてフラッシュバックされます。この制約には、FLASHBACK TABLE文の対象になっている表とFLASHBACK TABLE文の対象になっていない表との間に指定されている参照整合性制約も含まれます。
- フラッシュバック操作後も、元の表のデータは消失しません。後で、元の状態に戻すことができます。

ノート:

Oracle Flashback Table を使用するには、自動 UNDO 管理を使用している必要があります。「自動 UNDO 管理の概要」を参照してください。

関連項目:

FLASHBACK TABLE文の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: 表の管理

20.11 表の削除

不要になった表を削除するには、DROP TABLE文を使用します。

削除する表は、自分のスキーマに含まれているか、またはDROP ANY TABLEシステム権限を持っている必要があります。

ノート:

表を削除する前に、表を削除した結果についてよく理解しておいてください。

- 表を削除すると、その表定義はデータ・ディクショナリから削除されます。その結果、表のすべての行はアクセスできなくなります。
- 表に対応付けられている索引とトリガーは、すべて削除されます。
- 削除した表に依存しているビューと PL/SQL プログラム・ユニットはすべてそのまま残りますが、無効になります(使用できません)。データベースによる依存性管理の詳細は、「オブジェクト依存性の管理」を参照してください。
- 削除する表のシノニムはすべてそのまま残りますが、使用するとエラーが返されます。
- 削除した表に割り当てられていたエクステントは表領域の空き領域にすべて戻され、新しいエクステントまたは新しいオブジェクトを必要とするその他のオブジェクトによって再利用されます。クラスタ化表に対応する行はすべて、そのクラスタのブロックから削除されます。クラスタ化表の説明は、「クラスタの管理」を参照してください。

次の文は、hr.int_admin_emp表を削除します。

```
DROP TABLE hr.int_admin_emp;
```

削除する表に、他の表の外部キーが参照している主キーまたは一意キーが含まれていて、その子表のFOREIGN KEY制約を削除する場合は、次のようにDROP TABLE文にCASCADE句を指定します。

```
DROP TABLE hr.admin_emp CASCADE CONSTRAINTS;
```

表を削除した場合、通常、その表に関連付けられている領域はデータベースによってすぐには解放されません。そのかわりに、データベースは表の名前を変更してリサイクル・ビンに入れるため、後に表が誤って削除されたことがわかった場合、FLASHBACK TABLE文を使用してリカバリできます。DROP TABLE文の発行時に、表に関連付けられている空間をすぐに解放する場合は、次の文に示すようにPURGE句を含めます。

```
DROP TABLE hr.admin_emp PURGE;
```

表を削除するかわりに、切捨てを使用する場合もあります。TRUNCATE文は、表からすべての行を削除するための高速で効率的な方法ですが、切り捨てる表に関連付けられた構造(列定義、制限、トリガーなど)または権限付与には影響しません。TRUNCATE文の説明は、「表とクラスタの切捨て」を参照してください。

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 表の作成および変更で参照して実行してください。

親トピック: 表の管理

20.12 フラッシュバック・ドロップの使用とリサイクル・ビンの管理

表を削除した場合、その表に関連付けられている領域はデータベースによってすぐには削除されません。データベースによってこの表の名前が変更され、すべての関連オブジェクトとともにリサイクル・ビンへ入れられますが、後に表が誤って削除されたことがわかった場合、リサイクル・ビンからリカバリすることができます。この機能はフラッシュバック・ドロップと呼ばれ、表のリストアにはFLASHBACK TABLE文が使用されます。

この目的のためのFLASHBACK TABLE文の使用方法を説明する前に、リサイクル・ビンの機能と、その内容の管理方法を理解することが重要です。

- リサイクル・ビンの概要
  リサイクル・ビンとは、実際には、削除されたオブジェクトに関する情報を含んでいるデータ・ディクショナリ表です。削除された表および関連するオブジェクト(索引、制約、ネストした表など)は、実際には削除されず、領域を占有しています。
- リサイクル・ビンの有効化と無効化
  リサイクル・ビンが有効化されていると、削除した表とその依存オブジェクトはリサイクル・ビンに配置されます。リサイクル・ビンが無効になっている場合、削除された表およびその依存オブジェクトはリサイクル・ビンに配置されず削除されるため、リカバリするには他の手段(バックアップからのリカバリなど)を使用する必要があります。
- リサイクル・ビン内のオブジェクトの表示と問合せ
  リサイクル・ビンのオブジェクトに関する情報を取得するために、Oracle Databaseには2つのビューが用意されています。
- リサイクル・ビン内のオブジェクトのパージ
  リサイクル・ビンから項目をリストアすることはないと判断した場合は、PURGE文を使用して、項目および関連するオブジェクトをリサイクル・ビンから削除し、記憶域を解放できます。実行するには、項目を削除する場合と同じ権限が必要です。
- リサイクル・ビンからの表のリストア
  FLASHBACK TABLE ... TO BEFORE DROP文を使用すると、リサイクル・ビンからオブジェクトをリカバリできます。

親トピック: 表の管理

20.12.1 リサイクル・ビンの概要

リサイクル・ビンとは、実際には、削除されたオブジェクトに関する情報を含んでいるデータ・ディクショナリ表です。削除された表および関連するオブジェクト(索引、制約、ネストした表など)は、実際には削除されず、領域を占有しています。

この領域は、リサイクル・ビンから明確にパージされるまで、または、あまり可能性はありませんが、表領域の制約のためにデータベースによるパージが必要になるまでは、ユーザー領域の割当てにとって不利です。

ユーザーにSYSDBA権限がない場合、リサイクル・ビンの中でユーザーが所有するオブジェクトは、アクセス権があるオブジェクトのみであるため、各ユーザーには各自のリサイクル・ビンがあるとみなすことができます。リサイクル・ビンにある各自のオブジェクトは、次の文を使用して表示できます。

```
SELECT * FROM RECYCLEBIN;
```

DROP TABLE SQL文のみがオブジェクトをリサイクル・ビンに配置します。これにより、表とその表に関連するオブジェクトが追加されるため、それらをグループとしてリカバリできるようになります。表自体の他に、リサイクル・ビンに追加された関連するオブジェクトにも、次のタイプのオブジェクトを含めることができます。

- ネストした表
- LOBセグメント
- 索引

- 制約(外部キー制約以外)
- トリガー
- クラスタ

表領域をその内容も含めて削除すると、表領域内のオブジェクトはリサイクル・ビンに配置されず、その表領域に配置されていたオブジェクトに対するリサイクル・ビン内のエントリはすべてパージされます。内容を含まない表領域を削除した場合、つまり空の表領域を削除した場合も、表領域内のオブジェクトに対するリサイクル・ビン内のエントリがすべてパージされます。同様に、それぞれの削除操作は次のように処理されます。

- ユーザーを削除すると、そのユーザーが所有しているオブジェクトはリサイクル・ビンには配置されず、リサイクル・ビン内のオブジェクトがすべてパージされます。
- クラスタを削除すると、そのメンバー表はリサイクル・ビンには配置されず、リサイクル・ビン内の古いメンバー表がすべてパージされます。
- タイプを削除すると、サブタイプなどの依存オブジェクトはリサイクル・ビンには配置されず、リサイクル・ビン内の古い依存オブジェクトがすべてパージされます。

リサイクル・ビン内のオブジェクト名の変更

削除された表をリサイクル・ビンに移動すると、その表とその表に関連するオブジェクトには、システムで生成された名前が割り当てられます。名前の変更は、複数の表が同じ名前の場合に発生する可能性がある、名前の競合を回避するために必要です。名前の変更は、次の状況で発生します。

- ユーザーが表を削除し、同じ名前で表を作成し、その後、作成した表を再度削除した場合。
- 2人のユーザーが同じ名前の表を持ち、両方のユーザーが各自の表を削除した場合。

名前変更の表記規則は、次のとおりです。

```
BIN$unique_id$version
```

ここで:

- `unique_id`は、このオブジェクトに対する、26文字からなるグローバルに一意の識別子です。これによって、リサイクル・ビンの名前がすべてのデータベース全体で一意に識別されます。
- `version`は、データベースによって割り当てられるバージョン番号です。

親トピック: フラッシュバック・ドロップの使用とリサイクル・ビンの管理

20.12.2 リサイクル・ビンの有効化と無効化

リサイクル・ビンが有効化されていると、削除した表とその依存オブジェクトはリサイクル・ビンに配置されます。リサイクル・ビンが無効になっている場合、削除された表およびその依存オブジェクトはリサイクル・ビンに配置されず削除されるため、リカバリするには他の手段(バックアップからのリカバリなど)を使用する必要があります。

リサイクルビンを無効にしても、リサイクルビンにすでにあるオブジェクトはパージされず、影響も受けません。デフォルトで、リサイクル・ビンは有効になっています。

リサイクル・ビンは、`recyclebin`初期化パラメータを変更して有効化および無効化できます。このパラメータは動的ではないため、ALTER SYSTEM文で変更したときにはデータベースの再起動が必要です。

リサイクル・ビンを有効化するには:

1. 次のいずれかの文を発行します。

```
ALTER SESSION SET recyclebin = ON;
ALTER SYSTEM SET recyclebin = ON SCOPE = SPFILE;
```

2. ALTER SYSTEMを使用した場合は、データベースを再起動します。

リサイクル・ビンを無効化するには:

1. 次のいずれかの文を発行します。

```
ALTER SESSION SET recyclebin = OFF;
ALTER SYSTEM SET recyclebin = OFF SCOPE = SPFILE;
```

2. ALTER SYSTEMを使用した場合は、データベースを再起動します。

関連項目:

- 初期化パラメータの詳細は、「初期化パラメータと初期化パラメータ・ファイルについて」を参照してください
- 動的な初期化パラメータおよび静的な初期化パラメータについては、「初期化パラメータ値の変更」を参照してください

親トピック: フラッシュバック・ドロップの使用とリサイクル・ビンの管理

20.12.3 リサイクル・ビン内のオブジェクトの表示と問合せ

リサイクル・ビンのオブジェクトに関する情報を取得するために、Oracle Databaseには2つのビューが用意されています。

ビュー	説明
USER_RECYCLEBIN	ユーザーはこのビューを使用して、リサイクル・ビンにある削除された自分のオブジェクトを表示できます。使用しやすいように、シノニム RECYCLEBIN があります。
DBA_RECYCLEBIN	管理者はこのビューを使用して、リサイクル・ビンにある削除されたすべてのオブジェクトを表示できます。

これらのビューの使用目的の1つは、次の例のように、削除したオブジェクトに対してデータベースが割り当てた名前を識別することにあります。

```
SELECT object_name, original_name FROM dba_recyclebin
   WHERE owner = 'HR';
OBJECT_NAME                    ORIGINAL_NAME
------------------------------ --------------------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 EMPLOYEES
```

リサイクル・ビンの内容は、SQL*PlusのSHOW RECYCLEBINコマンドを使用して表示することもできます。

```
SQL> show recyclebin
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
EMPLOYEES        BIN$yrMKlZaVMhfgNAgAIMenRA==$0 TABLE        2003-10-27:14:00:19
```

リサイクル・ビンにあるオブジェクトは、他のオブジェクトと同じ要領で問い合せることができます。ただし、オブジェクトの名前は、リサイクル・ビンの中で識別されているとおりに指定する必要があります。たとえば:

```
SELECT * FROM "BIN$yrMKlZaVMhfgNAgAIMenRA==$0";
```

親トピック: フラッシュバック・ドロップの使用とリサイクル・ビンの管理

20.12.4 リサイクル・ビン内のオブジェクトのパージ

リサイクル・ビンから項目をリストアすることはないと判断した場合は、PURGE文を使用して、項目および関連するオブジェクトをリサイクル・ビンから削除し、記憶域を解放できます。実行するには、項目を削除する場合と同じ権限が必要です。

PURGE文を使用して表をパージする場合、リサイクル・ビンでの表の名前、または表の元の名前を使用できます。「リサイクル・ビン内のオブジェクトの表示と問合せ」で説明されているように、リサイクル・ビンでの名前はDBA_またはUSER_RECYCLEBINビューから取得できます。次の仮定的な例では、リサイクル・ビンに配置されたときにBIN$jsleilx392mk2=293$0に名前が変更された表hr.int_admin_empをパージします。

```
PURGE TABLE "BIN$jsleilx392mk2=293$0";
```

次の文を使用しても同様の結果となります。

```
PURGE TABLE int_admin_emp;
```

PURGE文を使用すると、指定の表領域からリサイクル・ビンのすべてのオブジェクトをパージ、または指定のユーザーに属する表領域オブジェクトのみをパージできます。次に例を示します。

```
PURGE TABLESPACE example;
PURGE TABLESPACE example USER oe;
```

次の文を使用することで、ユーザーは独自のオブジェクトのリサイクル・ビンをパージして、オブジェクトの領域を解放できます。

```
PURGE RECYCLEBIN;
```

SYSDBA権限またはPURGE DBA_RECYCLEBINシステム権限がある場合は、前述の文のRECYCLEBINのかわりに、DBA_RECYCLEBINを指定することによって、リサイクル・ビン全体をパージできます。

また、PURGE文を使用して、リサイクル・ビンから索引をパージ、またはリサイクル・ビンから指定の表領域にあるすべてのオブジェクトをパージすることもできます。

関連項目:

PURGE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: フラッシュバック・ドロップの使用とリサイクル・ビンの管理

20.12.5 リサイクル・ビンからの表のリストア

FLASHBACK TABLE ... TO BEFORE DROP文を使用すると、リサイクル・ビンからオブジェクトをリカバリできます。

ごみ箱内の表の名前または元の表の名前のいずれかを指定できます。オプションのRENAME TO句を使用すると、表をリカバリするときに名前を変更できます。「リサイクル・ビン内のオブジェクトの表示と問合せ」で説明されているように、リサイクル・ビンでの名前はDBA_またはUSER_RECYCLEBINビューから取得できます。FLASHBACK TABLE ... TO BEFORE DROP文を使用するためには、表の削除に必要な権限と同じ権限が必要です。

次の例は、int_admin_emp表をリストアし、その表に新しい名前を割り当てます。

```
FLASHBACK TABLE int_admin_emp TO BEFORE DROP
   RENAME TO int2_admin_emp;
```

表を複数回削除した場合、システムが生成するリサイクル・ビンでの名前が非常に有用です。たとえば、リサイクル・ビンに

int2_admin_emp表の3つのバージョンがあり、2番目のバージョンをリカバリするとします。これを実行するには、2つのFLASHBACK TABLE文を実行するか、または次の例に示すように、リサイクル・ビンを問い合せ、適切なシステム生成名にフラッシュバックできます。問合せに作成時間を含めると、正しい表をリストアしていることを確認できます。

```
SELECT object_name, original_name, createtime FROM recyclebin;
OBJECT_NAME                    ORIGINAL_NAME   CREATETIME
------------------------------ --------------- -------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 INT2_ADMIN_EMP  2006-02-05:21:05:52
BIN$yrMKlZaVMhfgNAgAIMenRA==$0 INT2_ADMIN_EMP  2006-02-05:21:25:13
BIN$yrMKlZaQMhfgNAgAIMenRA==$0 INT2_ADMIN_EMP  2006-02-05:22:05:53
FLASHBACK TABLE "BIN$yrMKlZaVMhfgNAgAIMenRA==$0" TO BEFORE DROP;
```

依存オブジェクトのリストア

リサイクル・ビンから表をリストアすると、索引などの依存オブジェクトは元の名前が復元されず、システム生成のリサイクル・ビンの名前のままになります。元の名前をリストアするには、依存オブジェクトの名前を手動で変更する必要があります。依存オブジェクトの元の名前を手動でリストアする場合は、表をリストアする前に、各依存オブジェクトのリサイクル・ビン内のシステム生成の名前をノートにとっておいてください。

次の例では、HRサンプル・スキーマから、削除した表JOB_HISTORYの索引の一部の元の名前をリストアします。この例では、HRユーザーとしてログインしていることを想定しています。

1. JOB_HISTORYの削除後、リサイクル・ビンからリストアする前に、次の問合せを実行します。

```
SELECT OBJECT_NAME, ORIGINAL_NAME, TYPE FROM RECYCLEBIN;
OBJECT_NAME                    ORIGINAL_NAME              TYPE
------------------------------ -------------------------- --------
BIN$DBo9UChtZSbgQFeMiAdCcQ==$0 JHIST_JOB_IX               INDEX
BIN$DBo9UChuZSbgQFeMiAdCcQ==$0 JHIST_EMPLOYEE_IX          INDEX
BIN$DBo9UChvZSbgQFeMiAdCcQ==$0 JHIST_DEPARTMENT_IX        INDEX
BIN$DBo9UChwZSbgQFeMiAdCcQ==$0 JHIST_EMP_ID_ST_DATE_PK    INDEX
BIN$DBo9UChxZSbgQFeMiAdCcQ==$0 JOB_HISTORY                TABLE
```

2. 次のコマンドを実行して表をリストアします。

```
FLASHBACK TABLE JOB_HISTORY TO BEFORE DROP;
```

3. 次の問合せを実行して、すべてのJOB_HISTORY索引がシステム生成のリサイクル・ビン名を保持していることを確認します。

```
SELECT INDEX_NAME FROM USER_INDEXES WHERE TABLE_NAME = 'JOB_HISTORY';

INDEX_NAME
------------------------------
BIN$DBo9UChwZSbgQFeMiAdCcQ==$0
BIN$DBo9UChtZSbgQFeMiAdCcQ==$0
BIN$DBo9UChuZSbgQFeMiAdCcQ==$0
BIN$DBo9UChvZSbgQFeMiAdCcQ==$0
```

4. 次のようにして、最初の2つの索引の元の名前をリストアします。

```
ALTER INDEX "BIN$DBo9UChtZSbgQFeMiAdCcQ==$0" RENAME TO JHIST_JOB_IX;
ALTER INDEX "BIN$DBo9UChuZSbgQFeMiAdCcQ==$0" RENAME TO JHIST_EMPLOYEE_IX;
```

システム生成の名前は、二重引用符で囲む必要があります。

親トピック: フラッシュバック・ドロップの使用とリサイクル・ビンの管理

20.13 索引構成表の管理

索引構成表の記憶域の編成は、プライマリBツリー索引の変形です。ヒープ構成表とは異なり、データは主キーの順に格納されます。

- 索引構成表の概要
  索引構成表は、プライマリBツリーの異形である記憶域編成を持っています。順序付けされていないコレクション(ヒープ)としてデータを格納する通常の(ヒープ構成)表とは異なり、索引構成表のデータはBツリーの索引構造に主キー・ソート方式で格納されます。索引構造の各リーフ・ブロックには、キー列と非キー列の両方が格納されます。
- 索引構成表の作成
  索引構成表により、高速な主キー・アクセスと高可用性が実現します。
- 索引構成表のメンテナンス
  索引構成表と通常の表の相違点は、物理的な構成のみです。論理的には、通常の表と同じように操作されます。INSERT、SELECT、DELETEおよびUPDATEの各文では、通常の表を指定する場合と同じように、索引構成表を指定できます。
- 索引構成表に対する2次索引の作成
  2次索引は索引構成表の索引です。2次索引は独立したスキーマ・オブジェクトであり、索引構成表とは別に格納されます。
- 索引構成表の分析
  通常の表と同様に、索引構成表の分析にはDBMS_STATSパッケージ、またはANALYZE文を使用します。
- 索引構成表でのORDER BY句の使用
  ORDER BY句が主キー列またはその接頭辞のみを参照する場合、行は主キー列でソートされた状態で返されるため、オプティマイザはソートのオーバーヘッドを回避します。
- 索引構成表の標準的な表への変換
  索引構成表を標準的な(ヒープ構成)表に変換するには、Oracleのインポート/エクスポート・ユーティリティ、あるいはCREATE TABLE...AS SELECT文を使用できます。

親トピック: 表の管理

20.13.1 索引構成表の概要

索引構成表は、プライマリBツリーの異形である記憶域編成を持っています。順序付けされていないコレクション(ヒープ)としてデータを格納する通常の(ヒープ構成)表とは異なり、索引構成表のデータはBツリーの索引構造に主キー・ソート方式で格納されます。索引構造の各リーフ・ブロックには、キー列と非キー列の両方が格納されます。

索引構成表の構造には、次の利点があります。

- 索引のみのスキャンで十分なため、主キーに対して高速にランダム・アクセスできます。また、索引構造以外に表記憶域がないため、新しい行の追加、行の更新、行の削除などにより表データを変更すると、索引構造の更新のみが実行されます。
- 行が主キー順にクラスタ化されているため、主キーに対して高速にレンジ・アクセスできます。
- 主キーの複製が回避されるため、記憶域の所要量を低く抑えられます。ヒープ構成表の場合、主キーは索引と基礎になる表の両方には格納されません。

索引構成表は、すべての表機能を備えています。制約、トリガー、LOB列とオブジェクト列、パーティション化、パラレル操作、オンライン再編成、およびレプリケーションなどの機能をサポートします。さらに、次の機能も提供します。

- 接頭辞圧縮
- オーバーフロー記憶域と固有の列配置
- ビットマップ索引を含めた2次索引。

高速な主キー・アクセスと高可用性を必要とするOLTPアプリケーションには、索引構成表が理想的です。たとえば、電子注文処理に使用される注文表の問合せおよびDMLは大部分が主キー・アクセスに基づいており、同時DMLの大量ボリュームが行の変更や索引での非効率な領域の使用の原因となり、再編成が頻繁に必要となります。索引構成表は、2次索引を無効化せずにオンラインで再編成できるため、ウィンドウの使用を制限される時間が大幅に短縮または排除されます。

索引構成表は、アプリケーション固有の索引構造をモデル化するのに適しています。たとえば、テキスト、イメージおよびオーディオ・データを含むコンテンツ・ベースの情報検索アプリケーションには、索引構成表を使用して有効にモデル化できる逆索引が必要です。インターネット検索エンジンの基本の構成要素は、索引構成表を使用してモデル化できる逆向きの索引です。

これらは、索引構成表のアプリケーションのほんの数例です。

関連項目:

- 索引構成表の詳細は、『Oracle Database概要』を参照してください
- 索引構成表のパーティション化の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: 索引構成表の管理

20.13.2 索引構成表の作成

索引構成表により、高速な主キー・アクセスと高可用性が実現します。

- 索引構成表の作成について
  CREATE TABLE文を使用して索引構成表を作成します。
- 例: 索引構成表の作成
  例を使用して索引構成表の作成を説明します。
- 索引構成表に対する制限
  索引構成表を作成する際には、いくつかの制限が適用されます。
- オブジェクト型を含む索引構成表の作成
  索引構成表にオブジェクト型を格納できます。
- しきい値の選択と監視
  キー列、および最初のいくつかの非キー列(頻繁にアクセスされる場合)を取り込めるしきい値を選択します。
- INCLUDING句の使用
  PCTTHRESHOLDの指定に加え、INCLUDING句を使用して、索引構成表にキー列とともに保存する非キー列を制御できます。
- 索引構成表作成のパラレル化
  CREATE TABLE...AS SELECT文を使用すると、索引構成表を作成して、既存の表からその索引構成表にデータをロードできます。PARALLEL句を指定することによって、ロードをパラレルで実行できます。
- 接頭辞圧縮の使用
  接頭辞圧縮(キー圧縮とも呼ばれます)を使用して索引構成表を作成すると、キー列の接頭辞が同じ値で繰り返し格納されるのを回避できます。

親トピック: 索引構成表の管理

20.13.2.1 索引構成表の作成について

CREATE TABLE文を使用して索引構成表を作成します。

索引構成表を作成する際には、次のような追加情報を指定する必要があります。

- ORGANIZATION INDEX修飾子。これによって、索引構成表であることを示します。
- 主キー。主キーは、単一列主キーの場合は列制約句、複数列主キーの場合は表制約句によって指定します。

必要に応じて、次の情報を指定できます。

- OVERFLOW句。この句は、非キー列の一部を別のオーバーフロー・データ・セグメントに格納できるようにすることにより、Bツリー索引の稠密なクラスタを保ちます。
- PCTTHRESHOLD値。これは、オーバーフロー・セグメントが使用されている際、索引ブロックに保存される行の部分の最大サイズを、ブロック・サイズのパーセンテージとして定義します。この行サイズが最大値を超える行の列は、オーバーフロー・セグメントに格納されます。行は、列境界で先頭部分と後尾部分の2つの部分に分割されます。先頭部分は指定されたしきい値に収まり、索引リーフ・ブロックのキーとともに格納されます。後尾部分は1つ以上の行の部分としてオーバーフロー領域に格納されます。このようにして、索引エントリにはキー値、指定したしきい値に収まる非キー列値、および行の残りの部分へのポインタが含まれます。
- INCLUDING句。この句は、主キーとともに索引ブロックに格納される非キー列を指定するために使用できます。

親トピック: 索引構成表の作成

20.13.2.2 例: 索引構成表の作成

例を使用して索引構成表の作成を説明します。

次の文によって、索引構成表が作成されます。

```
CREATE TABLE admin_docindex(
        token char(20),
        doc_id NUMBER,
        token_frequency NUMBER,
        token_offsets VARCHAR2(2000),
        CONSTRAINT pk_admin_docindex PRIMARY KEY (token, doc_id))
    ORGANIZATION INDEX
    TABLESPACE admin_tbs
    PCTTHRESHOLD 20
    OVERFLOW TABLESPACE admin_tbs2;
```

この例では、token列とdoc_id列で構成される主キーを使用して、admin_docindexという索引構成表を作成します。OVERFLOW句とPCTTHRESHOLD句では、行の長さが索引ブロック・サイズの20%を超えた場合に、そのしきい値を超えた列とその後のすべての列がオーバーフロー・セグメントに移動されるように指定しています。オーバーフロー・セグメントは、admin_tbs2表領域に格納されます。

関連項目:

索引構成表を作成する構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 索引構成表の作成

20.13.2.3 索引構成表に対する制限

索引構成表を作成する際には、いくつかの制限が適用されます。

索引構成表の作成には、次の制限があります。

- 列の最大数は1000です。
- キー列と非キー列を含めて、行の索引部分の最大列数は255です。255個を超える列が必要な場合は、オーバーフロー・セグメントを使用する必要があります。
- 主キーに含めることができる最大列数は32です。
- PCTTHRESHOLDは1から50の範囲内である必要があります。デフォルトは50です。
- すべてのキー列は、指定したしきい値内に収まる必要があります。
- 行の最大サイズが索引ブロック・サイズの50%を超える場合に、オーバーフロー・セグメントを指定しないと、CREATE TABLE文が失敗します。
- 索引構成表に仮想列は設定できません。
- 表に外部キーが含まれ、外部キーの親が索引構成表の場合は、別のセッションが親表のキー以外の列を更新中のとき、外部キーを含む行を更新するセッションがハングすることがあります。

  たとえば、departments表が索引構成表で、department_idがその主キーであるとします。employees表は、departments表の外部キーであるdepartment_id列を持ちます。あるセッションが、departments表のdepartment_idが20である行のdepartment_nameを更新中のとき、別のセッションが、employees表のdepartment_idが20である行を更新中であるとします。この場合、departments表を更新中のセッションがコミットされるかロールバックされるまで、employees表を更新中のセッションがハングすることがあります。
- 1つ以上のLOB列を含む索引構成表は、パラレルに移動できません。

親トピック: 索引構成表の作成

20.13.2.4 オブジェクト型を含む索引構成表の作成

索引構成表は、オブジェクト型を格納できます。

次の例は、オブジェクト型admin_typを作成し、オブジェクト型admin_typの列を含む索引構成表を作成しています。

```
CREATE OR REPLACE TYPE admin_typ AS OBJECT
    (col1 NUMBER, col2 VARCHAR2(6));
CREATE TABLE admin_iot (c1 NUMBER primary key, c2 admin_typ)
    ORGANIZATION INDEX;
```

オブジェクト型の索引構成表を作成することもできます。たとえば:

```
CREATE TABLE admin_iot2 OF admin_typ (col1 PRIMARY KEY)
    ORGANIZATION INDEX;
```

次に、索引構成表がネストした表を効率的に格納する例を示します。ネストした表の列ごとに、ネストした表のすべての行を保持する記憶表が内部的に作成されます。

```
CREATE TYPE project_t AS OBJECT(pno NUMBER, pname VARCHAR2(80));
/
CREATE TYPE project_set AS TABLE OF project_t;
/
CREATE TABLE proj_tab (eno NUMBER, projects PROJECT_SET)
    NESTED TABLE projects STORE AS emp_project_tab
```

```
                ((PRIMARY KEY(nested_table_id, pno))
    ORGANIZATION INDEX)
    RETURN AS LOCATOR;
```

ネストした表のシングル・インスタンスに属する行は、nested_table_id列で識別されます。ネストした表の列を格納するために通常の表が使用される場合、ネストした表の行は、一般的にクラスタ化が解除されます。ただし、索引構成表を使用する場合、ネストした表はnested_table_id列に基づいてクラスタ化できます。

関連項目:

- 索引構成表の作成に使用する構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- パーティション化された索引構成表の作成方法の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。
- オブジェクト型については、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

親トピック: 索引構成表の作成

20.13.2.5 しきい値の選択と監視

キー列とともに最初のいくつかの非キー列が頻繁にアクセスされる場合は、その非キー列を取り込めるしきい値を選択してください。

しきい値を選択した後、指定した値が適切な値であることを確認するために、表を監視できます。ANALYZE TABLE ... LIST CHAINED ROWS文を使用して、しきい値を超える行の数と、どの行がしきい値を超えているかを判断できます。

関連項目:

- 連鎖行の詳細は、「表とクラスタの連鎖行のリスト」を参照してください
- ANALYZE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 索引構成表の作成

20.13.2.6 INCLUDING句の使用

PCTTHRESHOLDの指定に加え、INCLUDING句を使用して、索引構成表にキー列とともに保存する非キー列を制御できます。

データベースでは、索引リーフ・ブロックのINCLUDING句に指定された列とその列までのすべての非キー列が、指定したしきい値を超えないかぎり格納されます。INCLUDING句で指定された列を超えた非キー列は、すべてオーバーフロー領域に格納されます。INCLUDINGとPCTTHRESHOLD句が競合する場合、PCTTHRESHOLDが優先されます。

ノート:

Oracle Database では、主キー・ベースのアクセス効率を高めるために、索引構成表のすべての主キー列が、表の先頭に(キー順に)移動されます。次に例を示します。

```
CREATE TABLE admin_iot4(a INT, b INT, c INT, d INT,
                  primary key(c,b))
```

```
    ORGANIZATION INDEX;
```

格納後の列順は、a b c d ではなく、c b a d となります。格納された列順に基づき、最後の主キー列は b になります。INCLUDING 列には、最後の主キー列(この例では b)と非キー列(つまり、格納後の列順で b の後の任意の列)のどちらでも指定できます。

次のCREATE TABLE文は、「例: 索引構成表の作成」で示した文と類似していますが、token_offsets列の値が常にオーバーフロー領域に格納される索引構成表を作成するように変更されています。

```
CREATE TABLE admin_docindex2(
        token CHAR(20),
        doc_id NUMBER,
        token_frequency NUMBER,
        token_offsets VARCHAR2(2000),
        CONSTRAINT pk_admin_docindex2 PRIMARY KEY (token, doc_id))
    ORGANIZATION INDEX
    TABLESPACE admin_tbs
    PCTTHRESHOLD 20
    INCLUDING token_frequency
    OVERFLOW TABLESPACE admin_tbs2;
```

この例では、索引リーフ・ブロック内のキー列値とともに、token_offsetsまでの非キー列のみ(この場合は1つの列のみ)が格納されます。

親トピック: 索引構成表の作成

20.13.2.7 索引構成表作成のパラレル化

CREATE TABLE...AS SELECT文を使用すると、索引構成表を作成して、既存の表からその索引構成表にデータをロードできます。PARALLEL句を指定することによって、ロードをパラレルで実行できます。

次の文は、従来型の表hr.jobsから行を選択し、索引構成表をパラレルに作成します。

```
CREATE TABLE admin_iot3(i PRIMARY KEY, j, k, l)
     ORGANIZATION INDEX
     PARALLEL
     AS SELECT * FROM hr.jobs;
```

この文によって、SQL*Loaderを使用するパラレル・バルク・ロードの代替手段が提供されます。

親トピック: 索引構成表の作成

20.13.2.8 接頭辞圧縮の使用

接頭辞圧縮(キー圧縮とも呼ばれます)を使用して索引構成表を作成すると、キー列の接頭辞が同じ値で繰り返し格納されるのを回避できます。

接頭辞圧縮によって、索引キーは接頭辞および接尾辞エントリに分割されます。圧縮するために、接頭辞エントリは索引ブロック内のすべての接尾辞エントリ間で共有されます。このような共有によって、領域が大幅に節約され、各索引ブロックに格納できるキー数が増え、パフォーマンスが向上します。

接頭辞圧縮を使用可能にするには、次の操作を行う際にCOMPRESS句を使用します。

- 索引構成表の作成
- 索引構成表の移動

また、接頭辞の長さをキー列の数で指定できます。これにより、キー列が接頭辞および接尾辞エントリにどのように分割されるか

が決まります。

```
CREATE TABLE admin_iot5(i INT, j INT, k INT, l INT, PRIMARY KEY (i, j, k))
    ORGANIZATION INDEX COMPRESS;
```

この文は、次の文と等価です。

```
CREATE TABLE admin_iot6(i INT, j INT, k INT, l INT, PRIMARY KEY(i, j, k))
    ORGANIZATION INDEX COMPRESS 2;
```

値リスト(1,2,3)、(1,2,4)、(1,2,7)、(1,3,5)、(1,3,4)、(1,4,4)では、(1,2)、(1,3)の反復的な発生が圧縮されます。

また、次のように、圧縮に使用されるデフォルトの接頭辞の長さを変更することもできます。

```
CREATE TABLE admin_iot7(i INT, j INT, k INT, l INT, PRIMARY KEY (i, j, k))
    ORGANIZATION INDEX COMPRESS 1;
```

値リスト(1,2,3)、(1,2,4)、(1,2,7)、(1,3,5)、(1,3,4)、(1,4,4)では、1の反復的な発生が圧縮されます。

圧縮は、次のように使用禁止にすることができます。

```
ALTER TABLE admin_iot5 MOVE NOCOMPRESS;
```

接頭辞圧縮のアプリケーションは、株価など、単一の項目に属して一連のタイムスタンプを表す行を使用する時系列のアプリケーションで使用されます。索引構成表には、主キーに従って行をクラスタ化する機能があるため、このようなアプリケーションには効果的です。索引構成表を主キー(株式銘柄、タイムスタンプ)で定義することによって、時系列データを効率的に格納および操作できます。接頭辞圧縮を採用した索引構成表を使用することによって、項目識別子(株式銘柄など)の反復的な発生を圧縮して、記憶域を大幅に節約できます。

関連項目:

接頭辞圧縮の詳細は、『Oracle Database概要』を参照してください

親トピック: 索引構成表の作成

20.13.3 索引構成表のメンテナンス

索引構成表と通常の表の相違点は、物理的な構成のみです。論理的には、通常の表と同じように操作されます。INSERT、SELECT、DELETEおよびUPDATEの各文では、通常の表を指定する場合と同じように、索引構成表を指定できます。

- 索引構成表の変更
  通常の表に使用可能な変更オプションはすべて索引構成表にも使用できます。使用可能なオプションには、ADD、MODIFY、DROP COLUMNSおよびCONSTRAINTSがあります。ただし、索引構成表の主キー制約は、削除、遅延または使用禁止にできません。
- 索引構成表の移動(再作成)
  索引構成表は主としてBツリー索引に格納されるため、増分更新の結果として断片化が生じることがあります。ただし、このような断片化は、ALTER TABLE...MOVE文を使用して索引を再作成することで低減できます。

親トピック: 索引構成表の管理

20.13.3.1 索引構成表の変更

通常の表に使用可能な変更オプションはすべて索引構成表にも使用できます。使用可能なオプションには、ADD、MODIFY、DROP COLUMNSおよびCONSTRAINTSがあります。ただし、索引構成表の主キー制約は、削除、遅延または使用禁止にで

きません。

ALTER TABLE文を使用すると、主キー索引セグメントとオーバーフロー・データ・セグメントの物理属性と記憶域属性を変更できます。OVERFLOWキーワードより前に指定したすべての属性は、主キー索引セグメントに適用できます。OVERFLOWキーワードより後に指定したすべての属性は、オーバーフロー・データ・セグメントに適用できます。たとえば、次のようにして、主キー索引セグメントのINITRANSを4に、オーバーフロー・データ・セグメントのINITRANSを6に設定できます。

```
ALTER TABLE admin_docindex INITRANS 4 OVERFLOW INITRANS 6;
```

また、PCTTHRESHOLDおよびINCLUDING列の値も変更できます。後続の操作では、新しい設定を使用して、先頭部分とオーバーフローの後尾の部分に行が分割されます。たとえば、admin_docindex表のPCTHRESHOLDおよびINCLUDING列の値を次のように変更できます。

```
ALTER TABLE admin_docindex PCTTHRESHOLD 15 INCLUDING doc_id;
```

INCLUDING列をdoc_idに設定すると、その後のすべての列、つまりtoken_frequencyおよびtoken_offsetsはオーバーフロー・データ・セグメントに格納されます。

オーバーフロー・データ・セグメントなしで作成された索引構成表の場合は、ADD OVERFLOW句を使用してオーバーフロー・データ・セグメントを追加できます。たとえば、次のように表admin_iot3にオーバーフロー・セグメントを追加できます。

```
ALTER TABLE admin_iot3 ADD OVERFLOW TABLESPACE admin_tbs2;
```

親トピック: 索引構成表のメンテナンス

20.13.3.2 索引構成表の移動(再作成)

索引構成表は主としてBツリー索引に格納されるため、増分更新の結果として断片化が生じることがあります。ただし、このような断片化は、ALTER TABLE...MOVE文を使用して索引を再作成することで低減できます。

次の文は、索引構成表admin_docindexを再作成します。

```
ALTER TABLE admin_docindex MOVE;
```

ONLINEキーワードを使用して、索引構成表をオンラインで再作成できます。OVERFLOWキーワードを指定すると、オーバーフロー・データ・セグメントが存在する場合はそれが再作成されます。たとえば、admin_docindex表を再作成し、オーバーフロー・データ・セグメントを再作成しない場合は、次のようにオンラインで移動します。

```
ALTER TABLE admin_docindex MOVE ONLINE;
```

admin_docindex表とオーバーフロー・データ・セグメントを再作成するには、次の文のように移動操作を実行します。この文は、表とオーバーフロー・データ・セグメントを新しい表領域に移動する方法も示しています。

```
ALTER TABLE admin_docindex MOVE TABLESPACE admin_tbs2
    OVERFLOW TABLESPACE admin_tbs3;
```

次の最後の文で、LOB列(CLOB)を持つ索引構成表が作成されます。その後、この表はLOB索引とともに移動し、データ・セグメントが再作成され新しい表領域に移動します。

```
CREATE TABLE admin_iot_lob
   (c1 number (6) primary key,
    admin_lob CLOB)
   ORGANIZATION INDEX
   LOB (admin_lob) STORE AS (TABLESPACE admin_tbs2);
.
.
.
ALTER TABLE admin_iot_lob MOVE LOB (admin_lob) STORE AS (TABLESPACE admin_tbs3);
```

関連項目:

索引構成表のLOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

親トピック: 索引構成表のメンテナンス

20.13.4 索引構成表に対する2次索引の作成

2次索引は、索引構成表の索引です。2次索引は独立したスキーマ・オブジェクトであり、索引構成表とは別に格納されます。

- 索引構成表に対する2次索引について
  索引構成表に2次索引を作成することで、複数のアクセス・パスを提供できます。
- 索引構成表に対する2次索引の作成
  索引構成表に2次索引を作成できます。
- 論理ROWIDの物理的不確定要素のメンテナンス
  論理ROWIDには、不確定要素が作成される際に行のブロック位置を識別する不確定要素を含めることができます。完全なキー検索を実行するかわりに、不確定要素を使用してブロックが直接検索されます。
- 索引構成表に対するビットマップ索引の指定
  索引構成表とともにマッピング表が作成される場合は、索引構成表でのビットマップ索引がサポートされます。

親トピック: 索引構成表の管理

20.13.4.1 索引構成表に対する2次索引について

索引構成表に2次索引を作成することで、複数のアクセス・パスを提供できます。

索引構成表の2次索引は、2つの点で通常の表の索引とは異なります。

- 索引構成表の2次索引には、物理的なROWIDではなく、論理的なROWIDが格納されます。これが必要な理由は、Bツリー索引の行にある本来の可動性のために、その行に永続的な物理アドレスがないためです。列の物理的な位置が変化しても、その論理的なROWIDは有効です。この効果の1つは、ALTER TABLE ... MOVEなどの表のメンテナンス操作によって2次索引が使用禁止状態にならないことです。
- 論理ROWIDにも、列が見つかる可能性があるデータベースのブロック・アドレスを識別する物理的な不確定要素が含まれています。物理的な不確定要素が正しい場合、2次キーが見つかると、2次索引スキャンによって単一の追加I/Oが生じます。パフォーマンスは、通常の表での2次索引スキャンの場合と同様です。

一意の2次索引、一意でない2次索引、機能ベースの2次索引およびビットマップ索引が、索引構成表の2次索引としてサポートされます。

親トピック: 索引構成表に対する2次索引の作成

20.13.4.2 索引構成表に対する2次索引の作成

索引構成表に対して2次索引を作成できます。

次の文は、索引構成表docindexに2次索引を作成します、doc_idとtokenはキー列です。

```
CREATE INDEX Doc_id_index on Docindex(Doc_id, Token);
```

この2次索引によって、問合せ(次の文にあるdoc_idの述語に関係する問合せ)が効率的に処理されます。

```
SELECT Token FROM Docindex WHERE Doc_id = 1;
```

親トピック: 索引構成表に対する2次索引の作成

20.13.4.3 論理ROWIDの物理的不確定要素のメンテナンス

論理ROWIDには、不確定要素が作成される際に行のブロック位置を識別する不確定要素を含めることができます。完全なキー検索を実行するかわりに、不確定要素を使用してブロックが直接検索されます。

ただし、新しい行が挿入されると、不確定要素は失効する可能性があります。索引は論理ROWIDの主キー構成要素を介してそのまま使用できますが、行へのアクセスは遅くなります。

1. 不確定要素の失効を監視するには、DBMS_STATSパッケージを使用して索引統計を収集します。

   既存の不確定要素が有効かどうかがチェックされ、有効な不確定要素を保持している行の割合がデータ・ディクショナリに記録されます。

2. DBA_INDEXESビュー(および関連するビュー)のPCT_DIRECT_ACCESS列を問い合せて、既存の不確定要素に関する統計を表示します。

3. 新しい不確定要素を取得するために、2次索引を再作成できます。

索引構成表に対する2次索引の再作成には、通常の表に対する索引の再作成とは異なり、実表の読込みが必要です。

不確定要素を修正する迅速で手軽な方法は、ALTER INDEX ... UPDATE BLOCK REFERENCES文を使用する方法です。この文はオンラインで実行されますが、DMLは基礎になる索引構成表でそのまま実行できます。

2次索引を再作成した後、あるいは不確定要素のブロック参照を更新した後は、索引統計を再度収集してください。

親トピック: 索引構成表に対する2次索引の作成

20.13.4.4 索引構成表に対するビットマップ索引の指定

索引構成表とともにマッピング表が作成される場合は、索引構成表でのビットマップ索引がサポートされます。

ビットマップ索引を作成するには、索引構成表の作成に使用するCREATE TABLE文、または後でマッピング表を追加するALTER TABLE文に、MAPPING TABLE句を指定します。

関連項目:

マッピング表の説明は、『Oracle Database概要』を参照してください

親トピック: 索引構成表に対する2次索引の作成

20.13.5 索引構成表の分析

通常の表と同様に、索引構成表の分析にはDBMS_STATSパッケージ、またはANALYZE文を使用します。

- 索引構成表のオプティマイザ統計の収集
  オプティマイザ統計を収集するには、DBMS_STATSパッケージを使用します。
- 索引構成表の構造の検証
  索引構成表の構造を検証、または連鎖行をリストするには、ANALYZE文を使用します。

親トピック: 索引構成表の管理

20.13.5.1 索引構成表のオプティマイザ統計の収集

オプティマイザ統計を収集するには、DBMS_STATSパッケージを使用します。

たとえば、次の文はhrスキーマの索引構成表countriesについて統計を収集します。

```
EXECUTE DBMS_STATS.GATHER_TABLE_STATS ('HR','COUNTRIES');
```

DBMS_STATSパッケージでは、主キー索引セグメントとオーバーフロー・データ・セグメントの両方が分析され、表の論理統計と物理統計が算出されます。

- 論理統計は、USER_TABLES、ALL_TABLESまたはDBA_TABLESを使用して問合せできます。
- 主キー索引セグメントの物理統計を問い合せるには、USER_INDEXES、ALL_INDEXESまたはDBA_INDEXES(および主キー索引名)を使用します。たとえば、表admin_docindexの主キー索引セグメントの物理統計は、次のようにして取得できます。

  ```
  SELECT LAST_ANALYZED, BLEVEL,LEAF_BLOCKS, DISTINCT_KEYS
     FROM DBA_INDEXES WHERE INDEX_NAME= 'PK_ADMIN_DOCINDEX';
  ```

- オーバーフロー・データ・セグメントの物理統計を問い合せるには、USER_TABLES、ALL_TABLESまたはDBA_TABLESを使用します。IOT_TYPE = 'IOT_OVERFLOW'で検索すると、オーバーフロー・エントリを識別できます。たとえば、admin_docindex表に対応付けられたオーバーフロー・データ・セグメントの物理属性は、次のようにして取得できます。

  ```
  SELECT LAST_ANALYZED, NUM_ROWS, BLOCKS, EMPTY_BLOCKS
     FROM DBA_TABLES WHERE IOT_TYPE='IOT_OVERFLOW'
            and IOT_NAME= 'ADMIN_DOCINDEX';
  ```

関連項目:

- オプティマイザ統計の収集の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- DBMS_STATSパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 索引構成表の分析

20.13.5.2 索引構成表の構造の検証

索引構成表の構造を検証、または連鎖行をリストするには、ANALYZE文を使用します。

これらの操作は、このマニュアルの該当する項で説明されています。

- 「表、索引、クラスタおよびマテリアライズド・ビューの妥当性チェック」
- 「表とクラスタの連鎖行のリスト」

ノート:

索引構成表の連鎖行をリストする場合は、特別な注意が必要です。詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

親トピック: 索引構成表の分析

20.13.6 索引構成表でのORDER BY句の使用

ORDER BY句が主キー列またはその接頭辞のみを参照する場合、行は主キー列でソートされた状態で返されるため、オプティマイザはソートのオーバーヘッドを回避します。

データはすでに主キーでソートされているので、次の2つの問合せはソートのオーバーヘッドを回避します。

```
SELECT * FROM admin_docindex2 ORDER BY token, doc_id;
SELECT * FROM admin_docindex2 ORDER BY token;
```

ただし、主キー列の接尾辞または非主キー列にORDER BY句がある場合は、別のソートが必要になります(他の2次索引が定義されていない場合)。

```
SELECT * FROM admin_docindex2 ORDER BY doc_id;
SELECT * FROM admin_docindex2 ORDER BY token_frequency;
```

親トピック: 索引構成表の管理

20.13.7 索引構成表の標準的な表への変換

索引構成表を標準的な(ヒープ構成)表に変換するには、Oracleのインポート/エクスポート・ユーティリティ、あるいはCREATE TABLE...AS SELECT文を使用できます。

索引構成表を標準的な表に変換するには:

- 従来型パスを使用して、索引構成表のデータをエクスポートします。
- 同じ定義で、標準的な表の定義を作成します。
- IGNORE=y(オブジェクト存在エラーを無視する)を指定して、索引構成表のデータをインポートします。

ノート:

索引構成表を標準的な表に変換する前に、Oracle8 より古いバージョンのエクスポート・ユーティリティでは索引構成表をエクスポートできないことに注意してください。

関連項目:

従来のIMPおよびEXPユーティリティとデータ・ポンプ・インポート/エクスポート・ユーティリティの使用方法の詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: 索引構成表の管理

20.14 パーティション表の管理

パーティション表では、データをパーティションと呼ばれるさらに小さく管理が容易な単位に分割でき、さらにサブパーティションに分割することもできます。各パーティションには、圧縮の有効化または無効化、圧縮のタイプ、物理記憶域設定、表領域など別個の物理属性を指定できるため、可用性およびパフォーマンスをより適切にチューニングできる構造になります。さらに、各パーティションを個別に管理できるため、バックアップや管理が簡素化され、これらの処理に必要な時間を削減できます。

パーティション表の管理の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: 表の管理

20.15 外部表の管理

外部表は、データベースに存在しない表です。これらは、データベース外部の、オブジェクト記憶域または外部ファイル(オペレーティング・システム・ファイルやHadoop Distributed File System (HDFS)ファイルなど)に存在します。

- 外部表について
  Oracle Databaseでは、外部表内のデータへの読取り専用アクセスが可能です。外部表はデータベース内に存在しない表として定義されており、アクセス・ドライバが提供されていればどのようなフォーマットにすることもできます。
- 外部表の作成
  外部表は、CREATE TABLE文でORGANIZATION EXTERNAL句を使用して作成します。この文は、データ・ディクショナリにメタデータのみを作成します。
- 外部表の変更
  ALTER TABLE文を使用して外部表を変更できます。
- 外部表の前処理
  ユーザー指定のプリプロセッサ・プログラムで外部表を前処理できます。前処理プログラムを使用すると、アクセス・ドライバでサポートされていない形式のファイルにあるデータを利用できます。
- 問合せによる外部表のパラメータ上書き
  SELECT文のEXTERNAL MODIFY句を使用して、外部表のパラメータを変更できます。
- インライン外部表の作成
  インライン外部表により、SQL文の一部として外部表の実行時定義が可能になり、データ・ディクショナリに永続オブジェクトとして外部表を作成する必要がなくなります。
- 外部表のパーティション化
  大量のデータがある場合は、外部表をパーティション化して、問合せのパフォーマンスを高速化し、データのメンテナンスを強化できます。
- 外部表の削除
  外部表の場合、DROP TABLE文によって、データベース内の表メタデータのみが削除されます。データベースの外に常駐する実際のデータには影響を与えません。
- 外部表のシステム権限およびオブジェクト権限
  外部表のシステム権限およびオブジェクト権限は、標準的な表のサブセットになります。

親トピック: 表の管理

20.15.1 外部表について

Oracle Databaseでは、外部表内のデータへの読取り専用アクセスが可能です。外部表はデータベース内に存在しない表として定義されており、アクセス・ドライバが提供されていればどのようなフォーマットにすることもできます。

外部表を記述するメタデータを提供することで、外部表内のデータをあたかも標準的なデータベース表内に存在しているデータのように公開できます。外部データは、SQLを使用して直接およびパラレルに問合せできます。

外部表のデータは、選択、結合、ソートなどが行えます。外部表のビューやシノニムも作成できます。ただし、外部表に対してDML操作(UPDATE、INSERTまたはDELETE)は実行できず、索引も作成できません。

外部表は、プラットフォームに依存しないフォーマット(オラクル社で開発され、Oracle Data Pumpで使用可能)に、任意のSELECT文の結果をアンロードするためのフレームワークを提供します。外部表は、データ・ウェアハウスで一般的な、抽出、変換およびロード(ETL)の基本タスクを実行する際に役立つ手段を提供します。

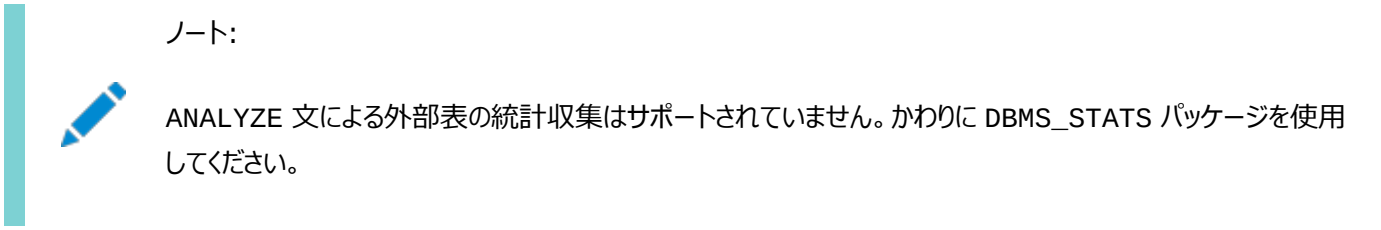
外部表のメタデータは、CREATE TABLE...ORGANIZATION EXTERNAL文を使用して定義します。外部表の定義は、外部データを最初にデータベースにロードしなくても外部データに対して任意のSQL問合せを実行できるビューとみなすことができます。表内の外部データを読み込むために使用されているメカニズムが、アクセス・ドライバです。外部表を使用してデータをアンロードすると、SELECT文のデータ型に基づいてメタデータが自動的に作成されます。

Oracle Databaseでは、外部表のためのアクセス・ドライバを提供しています。デフォルトのアクセス・ドライバはORACLE_LOADERで、Oracleのローダー・テクノロジを使用して外部ファイルからデータを読み込むことができます。ORACLE_LOADERアクセス・ドライバは、SQL*Loaderユーティリティの制御ファイル構文のサブセットであるデータ・マッピング機能を提供します。もう1つのアクセス・ドライバORACLE_DATAPUMPは、データをアンロード(つまり、データベースからデータを読み取り、1つ以上の外部ファイルで表された外部表にそのデータを挿入)してから、データをOracle Databaseに再ロードします。

Oracle Database 12cリリース2 (12.2)以降では、新しいアクセス・ドライバのORACLE_HIVEとORACLE_HDFSを使用できます。ORACLE_HIVEアクセス・ドライバはApache Hiveに格納されたデータを抽出できます。ORACLE_HDFSアクセス・ドライバは、Hadoop Distributed File System (HDFS)に格納されたデータを抽出できます。

Oracle Databaseリリース18c以降では、インライン外部表がサポートされています。インライン外部表により、SQL文の一部として外部表の実行時定義が可能になり、データ・ディクショナリに永続オブジェクトとして外部表を作成する必要がなくなります。

Oracle Databaseリリース19.10以降、オブジェクト記憶域が外部表データのソースとしてサポートされています。これにより、Oracleデータベースはクラウド・アプリケーションに保存されたデータを使用できます。ORC、Parquet、Avroなどの追加のデータ形式が、ORACLE_BIGDATAアクセス・ドライバでサポートされています。



ノート:

ANALYZE 文による外部表の統計収集はサポートされていません。かわりに DBMS_STATS パッケージを使用してください。

関連項目:

- 外部表に対する制限の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- アクセス・ドライバの詳細は、『Oracle Databaseユーティリティ』を参照してください。
- データ・ウェアハウス環境でETLに外部表を使用する方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- DBMS_STATSパッケージの使用方法の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- ORACLE_HIVEドライバとORACLE_HDFSドライバの詳細、および外部表の詳細は、『Oracle Databaseユーティリティ』を参照してください

親トピック: 外部表の管理

20.15.2 外部表の作成

外部表は、CREATE TABLE文でORGANIZATION EXTERNAL句を使用して作成します。この文は、データ・ディクショナリにメタデータのみを作成します。

ノート:

- Oracle Database 12c リリース 2 (12.2)以降では、大量データの高速問合せパフォーマンスおよび機能拡張されたデータ保守のために、外部表をパーティション化できます。
- 外部表に仮想列を設定できます。ただし、外部表の仮想列は evaluation_edition_clause または unusable_edition_clause を使用して定義はできません。

例20-21 外部表の作成とデータのロード

この例では、外部表を作成してから、データをデータベース表にアップロードしています。あるいは、CREATE TABLE文のAS subquery句を指定し、外部表フレームワークを介してデータをアンロードできます。外部表のデータ・ポンプ・アンロードは、ORACLE_DATAPUMPアクセス・ドライバのみを使用できます。

外部表のデータはempxt1.datおよびempxt2.datという2つのテキスト・ファイルに存在します。

ファイルempxt1.datには、次のサンプル・データが収められています。

```
360,Jane,Janus,ST_CLERK,121,17-MAY-2001,3000,0,50,jjanus
361,Mark,Jasper,SA_REP,145,17-MAY-2001,8000,.1,80,mjasper
362,Brenda,Starr,AD_ASST,200,17-MAY-2001,5500,0,10,bstarr
363,Alex,Alda,AC_MGR,145,17-MAY-2001,9000,.15,80,aalda
```

ファイルempxt2.datには、次のサンプル・データが収められています。

```
401,Jesse,Cromwell,HR_REP,203,17-MAY-2001,7000,0,40,jcromwel
402,Abby,Applegate,IT_PROG,103,17-MAY-2001,9000,.2,60,aapplega
403,Carol,Cousins,AD_VP,100,17-MAY-2001,27000,.3,90,ccousins
404,John,Richardson,AC_ACCOUNT,205,17-MAY-2001,5000,0,110,jrichard
```

次のSQL文は、スキーマhrに外部表admin_ext_employeesを作成し、外部表からhr.employees表にデータをロードします。

```
CONNECT  /  AS SYSDBA;
-- Set up directories and grant access to hr
CREATE OR REPLACE DIRECTORY admin_dat_dir
    AS '/flatfiles/data';
CREATE OR REPLACE DIRECTORY admin_log_dir
    AS '/flatfiles/log';
CREATE OR REPLACE DIRECTORY admin_bad_dir
    AS '/flatfiles/bad';
GRANT READ ON DIRECTORY admin_dat_dir TO hr;
GRANT WRITE ON DIRECTORY admin_log_dir TO hr;
GRANT WRITE ON DIRECTORY admin_bad_dir TO hr;
-- hr connects. Provide the user password (hr) when prompted.
CONNECT hr
-- create the external table
CREATE TABLE admin_ext_employees
                   (employee_id       NUMBER(4),
                    first_name        VARCHAR2(20),
                    last_name         VARCHAR2(25),
                    job_id            VARCHAR2(10),
                    manager_id        NUMBER(4),
                    hire_date         DATE,
                    salary            NUMBER(8,2),
                    commission_pct    NUMBER(2,2),
                    department_id     NUMBER(4),
                    email             VARCHAR2(25)
                   )
     ORGANIZATION EXTERNAL
     (
```

```
       TYPE ORACLE_LOADER
       DEFAULT DIRECTORY admin_dat_dir
       ACCESS PARAMETERS
       (
         records delimited by newline
         badfile admin_bad_dir:'empxt%a_%p.bad'
         logfile admin_log_dir:'empxt%a_%p.log'
         fields terminated by ','
         missing field values are null
         ( employee_id, first_name, last_name, job_id, manager_id,
           hire_date char date_format date mask "dd-mon-yyyy",
           salary, commission_pct, department_id, email
         )
       )
       LOCATION ('empxt1.dat', 'empxt2.dat')
     )
     PARALLEL
     REJECT LIMIT UNLIMITED;
-- enable parallel for loading (good if lots of data to load)
ALTER SESSION ENABLE PARALLEL DML;
-- load the data in hr employees table
INSERT INTO employees (employee_id, first_name, last_name, job_id, manager_id,
                       hire_date, salary, commission_pct, department_id, email)
            SELECT * FROM admin_ext_employees;
```

この例について、次の各段落で説明します。

この例で、最初の数行の文は、データソースを保存するオペレーティング・システム・ディレクトリ用のディレクトリ・オブジェクトと、アクセス・パラメータで指定される不良レコードやログ・ファイル用のディレクトリ・オブジェクトを作成します。また、必要に応じてREADまたはWRITEのディレクトリ・オブジェクト権限を付与する必要があります。

ノート:

ディレクトリ・オブジェクトまたは BFILE を作成する場合は、次の条件が満たされているかどうかを確認してください。

- オペレーティング・システム・ファイルが、シンボリック・リンクまたはハード・リンクではないこと。
- Oracle Database のディレクトリ・オブジェクトに指定されているオペレーティング・システムのディレクトリ・パスが、既存のオペレーティング・システムのディレクトリ・パスであること。
- ディレクトリ・オブジェクトに指定されているオペレーティング・システムのディレクトリ・パスの構成要素に、シンボリック・リンクが含まれていないこと。

TYPE指定は、外部表のアクセス・ドライバを示します。アクセス・ドライバは、データベースに対する外部データを解析するAPIです。TYPE指定を省略した場合は、ORACLE_LOADERがデフォルトのアクセス・ドライバになります。AS subquery句を指定して、1つのOracle Databaseからデータをアンロードし、同一または異なるOracle Databaseに再ロードする場合は、ORACLE_DATAPUMPアクセス・ドライバを指定する必要があります。

ACCESS PARAMETERS句で指定するアクセス・パラメータは、データベースには不透明です。これらのアクセス・パラメータはアクセス・ドライバによって定義されるもので、データベースが外部表にアクセスするときにアクセス・ドライバに提供されます。ORACLE_LOADERアクセス・パラメータの詳細は、『Oracle Databaseユーティリティ』を参照してください。

PARALLEL句は、データソースに対するパラレル問合せを可能にします。パラレル化の最小単位はデフォルトではデータソースですが、データソース内部でのパラレル・アクセスは可能なかぎり実装されます。たとえば、PARALLEL=3と指定すると、データソー

スに対して複数のパラレル実行サーバーを稼働しておくことができます。しかし、データソース内部でのパラレル・アクセスは、次の条件がすべて成り立つ場合にのみ、アクセス・ドライバによって提供されます。

- メディアが、データソース内部でのランダムな位置指定をサポートしている。
- レコード境界をランダムな位置から検索できる。
- データファイルが、複数のチャンクに分割することが適切なほど十分大きい。

ノート:

PARALLEL 句の指定は、大量のデータを扱う場合にのみ有効です。データが大量でない場合には、PARALLEL 句を指定すると悪影響を及ぼす可能性が高いので、お薦めできません。

REJECT LIMIT句は、外部データの問合せ中に発生する可能性のあるエラーの数に上限を設けないことを指定します。パラレル・アクセスの場合、REJECT LIMITが各パラレル実行サーバーに個別に適用されます。たとえば、REJECT LIMITに10を指定すると、各パラレル問合せプロセスで10個まで拒否が許可されます。このため、並行度が2、REJECT LIMITが10の場合、拒否が10個から20個の間で文が失敗する場合があります。1つのパラレル・サーバーで10個の拒否をすべて処理する場合、制限に達するため文が終了します。ただし、1つのパラレル実行サーバーが9個の拒否を処理し、もう1つのパラレル実行サーバーが9個の拒否を処理する場合、18個の拒否で文が成功します。したがって、パラレル問合せに関して正確に規定されるREJECT LIMITの値は、0 (ゼロ)およびUNLIMITEDのみです。

この例では、INSERT INTO TABLE文によって外部データソースからOracle Database SQLエンジンへのデータフローが生成され、そこでデータが処理されます。外部表ソースからのデータがアクセス・ドライバで解析されて外部表インタフェースに提供されると、外部データがその外部表現からOracle Databaseの内部データ型に変換されます。

例20-22 Oracle Cloudに格納されたデータの外部表の作成

この例では、Oracle Cloud Infrastructure Object Storage (オブジェクト記憶域)に格納されているデータにアクセスできる外部表を作成します。

外部表を作成する前に、オブジェクト記憶域の資格証明を格納するための資格証明オブジェクトを作成する必要があります。資格証明オブジェクトは、オブジェクト記憶域の資格証明へのアクセスに必要なユーザー名とパスワード(またはAPI署名キー)を暗号化形式で格納します。資格証明パスワードは、クラウド・サービスでユーザー名に対して作成された認証トークンと一致する必要があります。資格証明で指定されたアイデンティティが、オブジェクト・ストアの基礎となるデータにアクセスできることを確認してください。オブジェクト・ストアの資格証明を変更しないかぎり、このステップが必要なのは1回のみです。

次の例では、MY_OCI_CREDという名前の資格証明オブジェクトを作成します。

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'MY_OCI_CRED',
    username => 'oss_user@example.com',
    password => 'password');
END;
/
```

DBMS_CLOUD.CREATE_EXTERNAL_TABLEプロシージャを使用して、ソース・ファイルの上部に外部表を作成します。このプロシージャは、サポートされているクラウド・オブジェクト・ストレージ・サービスの外部ファイルをサポートします。資格証明は表レベルのプロパティであるため、外部ファイルは同じオブジェクト・ストアに存在する必要があります。

次の文は、オブジェクト記憶域に格納されているデータにアクセスするための外部表を作成します。外部表はソース・ファイル

channels.txtに基づいています。

```
BEGIN
   DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    table_name =>'CHANNELS_EXT',
    credential_name =>'MY_OCI_CRED',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/channels.txt',
    format => json_object('delimiter' value ','),
    column_list => 'CHANNEL_ID NUMBER, CHANNEL_DESC VARCHAR2(20), CHANNEL_CLASS
VARCHAR2(20)' );
END;
/
```

ここで:

- credential_nameは、作成された資格証明オブジェクトの名前(MY_OCI_CRED)です。
- file_uri_listは、問い合せるソース・ファイルのカンマ区切りリストです。
- formatは、ソース・ファイルの形式を記述するために指定できるオプションを定義します。
- column_listは、ソース・ファイル内の列定義のカンマ区切りリストです。

関連項目:

- 外部表を作成するためのCREATE TABLE文の構文の詳細、および句の使用の制限は、『Oracle Database SQL言語リファレンス』を参照してください。
- DBMS_CLOUDパッケージとそのプロシージャの詳細は、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください。

親トピック: 外部表の管理

20.15.3 外部表の変更

ALTER TABLE文を使用して外部表を変更できます。

外部表の特性を変更するには、表20-6のいずれかのALTER TABLE句を使用します。これ以外の句は使用できません。

表20-6 外部表のALTER TABLE句

ALTER TABLE句	説明	例
REJECT LIMIT	拒否の上限を変更します。デフォルト値は 0 です。	ALTER TABLE admin_ext_employees REJECT LIMIT 100;
PROJECT COLUMN	アクセス・ドライバが後続の問合せで行の妥当性をチェックする方法を決定します。 ● PROJECT COLUMN REFERENCED: アクセス・ドライバは、問合せの選択リストのみを処理します。この設定では、同一の外部表とは異なる列リストを問い	ALTER TABLE admin_ext_employees PROJECT COLUMN REFERENCED; ALTER TABLE admin_ext_employees PROJECT COLUMN ALL;

ALTER TABLE句	説明	例
	合せたときに、一貫した行のセットが提供されない可能性があります。 ● PROJECT COLUMN ALL: アクセス・ドライバは、外部表に定義されているすべての列を処理します。この設定では、外部表を問い合せたときに、常に一貫した行のセットが提供されます。これはデフォルトです。	
DEFAULT DIRECTORY	デフォルトのディレクトリ指定を変更します。	ALTER TABLE admin_ext_employees DEFAULT DIRECTORY admin_dat2_dir;
ACCESS PARAMETERS	外部表のメタデータの削除と再作成を行わずにアクセス・パラメータを変更できます。	ALTER TABLE admin_ext_employees ACCESS PARAMETERS (FIELDS TERMINATED BY ';');
LOCATION	外部表のメタデータの削除と再作成を行わずにデータソースを変更できます。	ALTER TABLE admin_ext_employees LOCATION ('empxt3.txt', 'empxt4.txt');
PARALLEL	標準的な表の場合と同じです。並列度を変更できます。	新しい構文はありません
ADD COLUMN	標準的な表の場合と同じです。外部表に列を追加できます。仮想列は使用できません。	新しい構文はありません。
MODIFY COLUMN	標準的な表の場合と同じです。外部表の列を変更できます。仮想列は使用できません。	新しい構文はありません。
SET UNUSED	ALTER TABLE DROP COLUMN コマンドに透過的に変換されます。外部表はデータベース内でメタデータのみで構成されているため、DROP COLUMN コマンドは SET UNUSED コマンドの実行と同じことになります。	新しい構文はありません。
DROP COLUMN	標準的な表の場合と同じです。外部表の列を削除できます。	新しい構文はありません。

ALTER TABLE句	説明	例
RENAME TO	標準的な表の場合と同じです。外部表の名前を変更できます。	新しい構文はありません。

親トピック: 外部表の管理

20.15.4 外部表の前処理

ユーザー指定のプリプロセッサ・プログラムで外部表を前処理できます。前処理プログラムを使用すると、アクセス・ドライバでサポートされていない形式のファイルにあるデータを利用できます。

注意:

PREPROCESSOR 句を使用するときに考慮する必要があるセキュリティ上の注意事項があります。詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

たとえば、圧縮形式で格納されているデータにアクセスできます。ORACLE_LOADERアクセス・ドライバに対して解凍プログラムを指定すると、アクセス・ドライバでデータが処理される際にデータを解凍できます。

前処理機能を使用するには、ORACLE_LOADERアクセス・ドライバのアクセス・パラメータでPREPROCESSOR句を指定する必要があります。プリプロセッサはディレクトリ・オブジェクトである必要があり、外部表にアクセスするユーザーには、そのディレクトリ・オブジェクトに対するEXECUTE権限が必要です。次の例にはPREPROCESSOR句が含まれており、ディレクトリとプリプロセッサ・プログラムを指定しています。

```
CREATE TABLE sales_transactions_ext
(PROD_ID NUMBER,
 CUST_ID NUMBER,
 TIME_ID DATE,
 CHANNEL_ID CHAR,
 PROMO_ID NUMBER,
 QUANTITY_SOLD NUMBER,
 AMOUNT_SOLD NUMBER(10,2),
 UNIT_COST NUMBER(10,2),
 UNIT_PRICE NUMBER(10,2))
ORGANIZATION external
(TYPE oracle_loader
 DEFAULT DIRECTORY data_file_dir
 ACCESS PARAMETERS
  (RECORDS DELIMITED BY NEWLINE
   CHARACTERSET AL32UTF8
   PREPROCESSOR exec_file_dir:'zcat'
   BADFILE log_file_dir:'sh_sales.bad_xt'
   LOGFILE log_file_dir:'sh_sales.log_xt'
   FIELDS TERMINATED BY "|" LDRTRIM
  ( PROD_ID,
    CUST_ID,
    TIME_ID,
    CHANNEL_ID,
    PROMO_ID,
    QUANTITY_SOLD,
    AMOUNT_SOLD,
    UNIT_COST,
    UNIT_PRICE))
 location ('sh_sales.dat.gz')
)REJECT LIMIT UNLIMITED;
```

PREPROCESSOR句は、Oracle Database Vaultを使用しているデータベースには使用できません。

ノート:

Windows プラットフォームでは、プリプロセッサ・プログラムの拡張子は .bat または .cmd である必要があります。

関連項目:

- PREPROCESSOR句の詳細は、『Oracle Databaseユーティリティ』を参照してください。
- PREPROCESSOR句のセキュリティ上の注意事項の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: 外部表の管理

20.15.5 問合せによる外部表のパラメータ上書き

SELECT文のEXTERNAL MODIFY句を使用して、外部表のパラメータを変更できます。

EXTERNAL MODIFY句を使用して、次に示す外部表の句を上書きできます。

- DEFAULT DIRECTORY
- LOCATION
- ACCESS PARAMETERS
- REJECT LIMIT

単一の問合せの複数の句を変更できます。LOCATIONおよびREJECT LIMITにはバインド変数を指定できますが、DEFAULT DIRECTORYまたはACCESS PARAMETERSには指定できません。

変更は問合せのみに適用されます。表に永続的には影響しません。

パーティション化された外部表の場合は、表レベルの句のみを上書きできます。

1. 外部表の問合せに必要な権限を持つユーザーとして、データベースに接続します。
2. 外部表に対して、EXTERNAL MODIFY句を指定してSELECT文を発行します。

例 20-23 問合せによる外部表のパラメータ上書き

sales_externalという名前の外部表で、REJECT LIMITが25に設定されているとします。次の問合せで、この設定をREJECT LIMIT UNLIMITEDに変更します。

```
SELECT * FROM sales_external EXTERNAL MODIFY (LOCATION ('sales_9.csv')
   REJECT LIMIT UNLIMITED);
```

親トピック: 外部表の管理

20.15.6 インライン外部表の使用

インライン外部表により、SQL文の一部として外部表の実行時定義が可能になり、データ・ディクショナリに永続オブジェクトとして外部表を作成する必要がなくなります。

インライン外部表では、外部表の作成にCREATE TABLE文で使用したものと同じ構文を、実行時にSELECT文で使用でき

ます。インライン外部表は、問合せブロックのFROM句で指定します。インライン外部表を含む問合せには、結合や集計などのために標準の表を含めることもできます。

次のSQL文では、外部データに対する実行時の問合せを実行します。

```
SELECT * FROM EXTERNAL (
    (time_id         DATE NOT NULL,
     prod_id         INTEGER NOT NULL,
     quantity_sold   NUMBER(10,2),
     amount_sold     NUMBER(10,2))
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY data_dir1
   ACCESS PARAMETERS (
      RECORDS DELIMITED BY NEWLINE
      FIELDS TERMINATED BY '|')
   LOCATION ('sales_9.csv') REJECT LIMIT UNLIMITED) sales_external;
```

これまでにsales_externalという名前の表が作成されていませんが、この問合せは外部データを読み取って、その結果を返します。

ノート:

インライン外部表では、パーティション化がサポートされません。問合せでは、スキャンするディレクトリとファイルを制御できるため、問合せに不要なファイルを省略することでプルーニングを実施できます。

親トピック: 外部表の管理

20.15.7 外部表のパーティション化

大量のデータがある場合は、外部表をパーティション化して、問合せのパフォーマンスを高速化し、データのメンテナンスを強化できます。

- 外部表のパーティション化について
  外部表のデータのパーティション化は、データベースに格納されている表のパーティション化とほぼ同様ですが、いくつか違いがあります。パーティション化された外部表のファイルは、ファイル・システム、Apache Hive記憶域またはHadoop Distributed File System (HDFS)に格納できます。
- パーティション化された外部表の制限
  パーティション化された外部表にはいくつかの制限が適用されます。
- パーティション化された外部表の作成
  パーティション化された非コンポジット外部表を作成するには、ORGANIZATION EXTERNAL句およびPARTITION BY句を指定して、CREATE TABLE文を発行します。パーティション化されたコンポジット外部表を作成するには、SUBPARTITION BY句も指定する必要があります。
- パーティション化された外部表の変更
  ALTER TABLE文を使用して、パーティション化された外部表の表レベルの外部パラメータを変更できますが、パーティション・レベルおよびサブパーティション・レベルのパラメータは変更できません。

親トピック: 外部表の管理

20.15.7.1 外部表のパーティション化について

外部表のデータのパーティション化は、データベースに格納されている表のパーティション化とほぼ同様ですが、いくつか違いがあり

ます。パーティション化された外部表のファイルは、ファイル・システム、Apache Hive記憶域またはHadoop Distributed File System (HDFS)に格納できます。

外部表のパーティション化を開始する前に、パーティション化に関する概念について、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

外部表をパーティション化する主な理由は、データベースに格納される表のパーティション化と同様のパフォーマンス改善の利点を活用するためです。特に、パーティション・プルーニングおよびパーティション単位の結合によって、問合せのパフォーマンスを改善できます。パーティション・プルーニングでは、問合せが1つのパーティションのみに適用されるため、すべてのデータではなく、外部表のデータのサブセットを対象に問合せを実行できます。パーティション・ワイズ結合は、2つの表を結合する際に両方の表が結合キーでパーティション化される場合や、参照パーティション表を親表と結合する場合に適用できます。パーティション・ワイズ結合では、大きな結合が小さな結合に分割され、その分割が各パーティションで行われるため結合全体がこれまでよりも短い時間で完了します。

データベースの表でサポートされるほとんどのパーティション方法は、外部表の場合もサポートされます。外部表は範囲またはリストに基づいてパーティション化可能で、コンポジット・パーティション化もサポートされます。ただし、外部表の場合はハッシュ・パーティション化はサポートされません。

データベースに格納されるパーティション化表の場合、表領域を使用して各パーティションの記憶域を指定できます。パーティション化された外部表の場合は、各パーティションのディレクトリとファイルを指定して、各パーティションの記憶域を指定します。

パーティション化された外部表を作成するための句

次に、パーティション化されていない外部表を作成する句を示します。

- `TYPE` - 外部表のタイプに応じてアクセス・ドライバを指定します(`ORACLE_LOADER`、`ORACLE_DATAPUMP`、`ORACLE_HIVE`、`ORACLE_HDFS`)。
- `DEFAULT DIRECTORY` - 明示的にディレクトリ・オブジェクトに名前を付けないすべての入出力ファイルについて、使用するデフォルトのディレクトリをディレクトリ・オブジェクトで指定します。
- `ACCESS PARAMETERS` - 外部データ・ソースを記述します。
- `LOCATION` - 外部表のファイルを指定します。
- `REJECT LIMIT` - 外部表の問合せ中に発生する可能性のあるエラーの数の制限を指定します。

パーティション化された外部表を作成する際には、各パーティションを定義するPARTITION句を指定する必要があります。次の表は、外部表の作成時に各レベルで指定できる句の説明です。

表20-7 外部表の句とパーティション化

句	表レベル	パーティション・レベル	サブパーティション・レベル
TYPE	使用可能	使用不可	使用不可
DEFAULT DIRECTORY	使用可能	使用可能	使用可能
ACCESS PARAMETERS	使用可能	使用不可	使用不可
LOCATION	使用不可	使用可能	使用可能

句	表レベル	パーティション・レベル	サブパーティション・レベル
REJECT LIMIT	使用可能	使用不可	使用不可

非コンポジット・パーティション化表の場合は、パーティションに対するLOCATION句で、パーティションのファイルを指定する必要があります。コンポジット・パーティション化表の場合は、サブパーティションに対するLOCATION句で、サブパーティションのファイルを指定する必要があります。パーティションにサブパーティションがある場合は、サブパーティションに対してLOCATION句を指定できますが、パーティションに対しては指定できません。パーティションまたはサブパーティションに対するLOCATION句を省略した場合は、空のパーティションまたは空のサブパーティションが作成されます。

LOCATION句では、ファイルがdirectory:fileの形式で指定され、1つの句で複数のファイルを指定できます。directoryの部分はオプションです。次のルールは、パーティションまたはサブパーティションによって使用されるディレクトリに適用されます。

- パーティションまたはサブパーティションに対するLOCATION句でディレクトリを指定する場合、指定はその場所にのみに適用されます。
- 特定のパーティションに対するLOCATION句では、ディレクトリが指定されていない各ファイルについて、パーティション・レベルまたは表レベルのDEFAULT DIRECTORY句によって指定されたディレクトリが(パーティション・レベル優先で)使用されます。

  たとえば、CREATE TABLE文のORGANIZATION EXTERNAL句にDEFAULT DIRECTORY句が含まれており、この文のPARTITION句ではLOCATION句のファイルのディレクトリが指定されていない場合、ファイルは、表に対するDEFAULT DIRECTORY句で指定されたディレクトリを使用します。
- 特定のサブパーティションに対するLOCATION句では、ディレクトリが指定されていない各ファイルについて、サブパーティション、パーティションまたは表レベルのDEFAULT DIRECTORY句によって指定されたディレクトリが(サブパーティション・レベル優先で)使用されます。

  たとえば、PARTITION句にDEFAULT DIRECTORY句が含まれており、パーティションのSUBPARITION句ではLOCATION句のファイルのディレクトリが指定されていない場合、ファイルは、パーティションに対するDEFAULT DIRECTORY句で指定されたディレクトリを使用します。
- パーティションまたはサブパーティションのデフォルト・ディレクトリは、LOCATION句では指定できません。DEFAULT DIRECTORY句でのみ指定できます。

関連項目:

例20-25のディレクトリ・ルールの説明

ORACLE_HIVEアクセス・ドライバの使用方法

Apache Hiveでは独自のパーティション化を使用します。パーティション化された外部表を作成するには、DBMS_HADOOPパッケージのCREATE_EXTDDL_FOR_HIVEプロシージャを使用します。このプロシージャによってデータ定義言語(DDL)文が生成され、これを使用して、Apache Hive記憶域のパーティション化に対応するようにパーティション化された外部表を作成できます。

DBMS_HADOOPパッケージにはSYNC_PARTITIONS_FOR_HIVEプロシージャも含まれています。このプロシージャは、Apache Hive記憶域のパーティション化された外部表のパーティションを、Oracle Databaseに格納される同じ表のパーティション・メタデータと自動的に同期します。

関連トピック

- 外部表の変更
- 『Oracle Databaseユーティリティ』
- Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス

親トピック: 外部表のパーティション化

20.15.7.2 パーティション化された外部表の制限

パーティション化された外部表にはいくつかの制限が適用されます。

次に、パーティション化された外部表の制限を示します。

- パーティション化されていない外部表に適用されるすべての制限は、パーティション化されている外部表にも適用されます。
- パーティションの最大数など、データベースに格納される表に適用されるパーティションの制限が、パーティション化された外部表にも適用されます。
- Oracle Databaseは、パーティションの外部ファイルにパーティション定義を満たすデータが含まれることを保証できません。
- DEFAULT DIRECTORYおよびLOCATION句のみを、PARTITIONまたはSUBPARTITION句で指定できます。
- パーティション化された外部表をALTER TABLE文で変更する場合、MODIFY PARTITION、EXCHANGE PARTITION、MOVE PARTITION、MERGE PARTITIONS、SPLIT PARTITION、COALESCE PARTITIONおよびTRUNCATE PARTITION句はサポートされません。
- 参照パーティション化、自動リスト・パーティション化および間隔パーティション化はサポートされません。
- サブパーティション・テンプレートはサポートされません。
- ORACLE_DATAPUMPアクセス・ドライバでは、CREATE TABLE AS SELECT文を使用して、パーティションの外部ファイルを移入できません。
- パーティション化された外部表の場合、増分統計は収集されません。
- 他のドライバの場合に使用できるパーティション化方法に対する制限に加えて、ORACLE_HIVEアクセス・ドライバの場合、レンジ・パーティション化とコンポジット・パーティション化もサポートされません。
- SELECT文でEXTERNAL MODIFY句を指定して、パーティション・レベルまたはサブパーティション・レベルの句を上書きはできません。EXTERNAL MODIFY句で上書きできるのは、表レベルでサポートされる外部の句のみです。パーティション化された外部表の場合、LOCATION句は表レベルでは使用できないため、EXTERNAL MODIFY句では上書きできません。

関連項目:

- 外部表について
- 外部表を作成するためのCREATE TABLE文の構文の詳細、および句の使用の制限は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 外部表のパーティション化

20.15.7.3 パーティション化された外部表の作成

パーティション化された非コンポジット外部表を作成するには、ORGANIZATION EXTERNAL句およびPARTITION BY句を指定して、CREATE TABLE文を発行します。パーティション化されたコンポジット外部表を作成するには、SUBPARTITION BY句も指定する必要があります。

PARTITION BY句とSUBPARTITION BY句で、各パーティションとサブパーティションの外部ファイルの場所を指定します。

パーティション化された外部表を作成するには、データベースの互換性レベルが12.2.0以上の必要があります。

1. 外部表の作成に必要な権限を持つユーザーとして、データベースに接続します。
   必要な権限の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
2. ORGANIZATION EXTERNAL句およびPARTITION BY句を指定して、CREATE TABLE文を発行します。パーティション化されたコンポジット表の場合は、SUBPARTITION BY句も指定します。

例20-24 すべてのパーティションに共通のアクセス・パラメータを使用するパーティション化された外部表の作成

この例では、order_date列の日付データでパーティション化されるorders_external_rangeという名前の外部表を作成します。ACCESS PARAMETERS句は、ORACLE_LOADERアクセス・ドライバについて表レベルで指定します。data_dir1ディレクトリ・オブジェクトは、パーティションmonth1、month2およびmonth3に使用されるデフォルトのディレクトリ・オブジェクトです。pmaxパーティションの場合は、DEFAULT DIRECTORY句でdata_dir2ディレクトリ・オブジェクトが指定されるため、data_dir2ディレクトリ・オブジェクトがpmaxパーティションに使用されます。

```
-- Set up directories and grant access to oe
CREATE OR REPLACE DIRECTORY data_dir1
    AS '/flatfiles/data1';
CREATE OR REPLACE DIRECTORY data_dir2
    AS '/flatfiles/data2';
CREATE OR REPLACE DIRECTORY bad_dir
    AS '/flatfiles/bad';
CREATE OR REPLACE DIRECTORY log_dir
    AS '/flatfiles/log';
GRANT READ ON DIRECTORY data_dir1 TO oe;
GRANT READ ON DIRECTORY data_dir2 TO oe;
GRANT WRITE ON DIRECTORY bad_dir TO oe;
GRANT WRITE ON DIRECTORY log_dir TO oe;
-- oe connects. Provide the user password (oe) when prompted.
CONNECT oe
-- create the partitioned external table
CREATE TABLE orders_external_range(
    order_id           NUMBER(12),
    order_date         DATE NOT NULL,
    customer_id        NUMBER(6) NOT NULL,
    order_status       NUMBER(2),
    order_total        NUMBER(8,2),
    sales_rep_id       NUMBER(6))
ORGANIZATION EXTERNAL(
   TYPE ORACLE_LOADER
   DEFAULT DIRECTORY data_dir1
   ACCESS PARAMETERS(
      RECORDS DELIMITED BY NEWLINE
      BADFILE bad_dir: 'sh%a_%p.bad'
      LOGFILE log_dir: 'sh%a_%p.log'
      FIELDS TERMINATED BY '|'
      MISSING FIELD VALUES ARE NULL))
PARALLEL
REJECT LIMIT UNLIMITED
PARTITION BY RANGE (order_date)
  (PARTITION month1 VALUES LESS THAN (TO_DATE('31-12-2014', 'DD-MM-YYYY'))
        LOCATION ('sales_1.csv'),
    PARTITION month2 VALUES LESS THAN (TO_DATE('31-01-2015', 'DD-MM-YYYY'))
        LOCATION ('sales_2.csv'),
```

```
    PARTITION month3 VALUES LESS THAN (TO_DATE('28-02-2015', 'DD-MM-YYYY'))
        LOCATION ('sales_3.csv'),
    PARTITION pmax VALUES LESS THAN (MAXVALUE)
        DEFAULT DIRECTORY data_dir2 LOCATION('sales_4.csv'));
```

前述の例では、デフォルト・ディレクトリdata_dir2がpmaxパーティションに指定されています。次のようにLOCATION句を使用して、このパーティションの特定の場所に対してディレクトリを指定することもできます。

```
PARTITION pmax VALUES LESS THAN (MAXVALUE)
   LOCATION ('data_dir2:sales_4.csv')
```

この場合、data_dir2ディレクトリがsales_4.csvの場所に対して指定されていますが、data_dir2ディレクトリはこのパーティションのデフォルト・ディレクトリではありません。したがって、pmaxパーティションのデフォルト・ディレクトリは、表のデフォルト・ディレクトリと同じdata_dir1です。

例20-25 パーティション化されたコンポジット・リスト・レンジ外部表の作成

この例では、region列のデータでパーティション化されるaccountsという名前の外部表を作成します。このパーティションは、balance列のデータの範囲を使用してサブパーティション化されます。ACCESS PARAMETERS句は、ORACLE_LOADERアクセス・ドライバについて表レベルで指定します。各サブパーティションにLOCATION句を指定します。

表レベルのDEFAULT DIRECTORY句がdata_dir1ディレクトリ・オブジェクトに設定されており、次のような場合を除き、このディレクトリ・オブジェクトがすべてのサブパーティションに使用されます。

- パーティションp_southcentralの場合は、パーティション・レベルのDEFAULT DIRECTORY句がdata_dir2ディレクトリ・オブジェクトに設定されています。このパーティションの場合、サブディレクトリp_sc_low、p_sc_highおよびp_sc_extraordinaryがこのデフォルト・ディレクトリを使用します。
- パーティションp_southcentralでは、サブパーティションp_sc_averageのサブパーティション・レベルのDEFAULT DIRECTORY句がdata_dir3ディレクトリ・オブジェクトに設定されており、このサブパーティションはdata_dir3ディレクトリ・オブジェクトを使用します。
- 前述のように、p_sc_highサブパーティションのデフォルト・ディレクトリはdata_dir2です。p_sc_highサブパーティションにはDEFAULT DIRECTORY句がないため、p_southcentralパーティションに対してPARTITION BY句で指定されたDEFAULT DIRECTORYからデフォルト・ディレクトリdata_dir2が継承されます。p_sc_highサブパーティションのファイルは次のディレクトリを使用します。
    - psch1.csvファイルはdata_dir2 (サブパーティションのデフォルト・ディレクトリ)を使用します。
    - psch2.csvファイルは、場所としてdata_dir4ディレクトリが指定されているため、data_dir4ディレクトリを使用します。

```
-- Set up the directories and grant access to oe
CREATE OR REPLACE DIRECTORY data_dir1
    AS '/stage/data1_dir';
CREATE OR REPLACE DIRECTORY data_dir2
    AS '/stage/data2_dir';
CREATE OR REPLACE DIRECTORY data_dir3
    AS '/stage/data3_dir';
CREATE OR REPLACE DIRECTORY data_dir4
    AS '/stage/data4_dir';
CREATE OR REPLACE DIRECTORY bad_dir
    AS '/stage/bad_dir';
CREATE OR REPLACE DIRECTORY log_dir
    AS '/stage/log_dir';
GRANT READ ON DIRECTORY data_dir1 TO oe;
GRANT READ ON DIRECTORY data_dir2 TO oe;
GRANT READ ON DIRECTORY data_dir3 TO oe;
```

```
GRANT READ ON DIRECTORY data_dir4 TO oe;
GRANT WRITE ON DIRECTORY bad_dir TO oe;
GRANT WRITE ON DIRECTORY log_dir TO oe;
-- oe connects. Provide the user password (oe) when prompted.
CONNECT oe
-- create the partitioned external table
CREATE TABLE accounts
( id             NUMBER,
  account_number NUMBER,
  customer_id    NUMBER,
  balance        NUMBER,
  branch_id      NUMBER,
  region         VARCHAR(2),
  status         VARCHAR2(1)
)
ORGANIZATION EXTERNAL(
   TYPE ORACLE_LOADER
   DEFAULT DIRECTORY data_dir1
   ACCESS PARAMETERS(
      RECORDS DELIMITED BY NEWLINE
      BADFILE bad_dir: 'sh%a_%p.bad'
      LOGFILE log_dir: 'sh%a_%p.log'
      FIELDS TERMINATED BY '|'
      MISSING FIELD VALUES ARE NULL))
PARALLEL
REJECT LIMIT UNLIMITED
PARTITION BY LIST (region)
SUBPARTITION BY RANGE (balance)
( PARTITION p_northwest VALUES ('OR', 'WA')
  ( SUBPARTITION p_nw_low VALUES LESS THAN (1000) LOCATION ('pnwl.csv'),
    SUBPARTITION p_nw_average VALUES LESS THAN (10000) LOCATION ('pnwa.csv'),
    SUBPARTITION p_nw_high VALUES LESS THAN (100000) LOCATION ('pnwh.csv'),
    SUBPARTITION p_nw_extraordinary VALUES LESS THAN (MAXVALUE) LOCATION ('pnwe.csv')
  ),
  PARTITION p_southwest VALUES ('AZ', 'UT', 'NM')
  ( SUBPARTITION p_sw_low VALUES LESS THAN (1000) LOCATION ('pswl.csv'),
    SUBPARTITION p_sw_average VALUES LESS THAN (10000) LOCATION ('pswa.csv'),
    SUBPARTITION p_sw_high VALUES LESS THAN (100000) LOCATION ('pswh.csv'),
    SUBPARTITION p_sw_extraordinary VALUES LESS THAN (MAXVALUE) LOCATION ('pswe.csv')
  ),
  PARTITION p_northeast VALUES ('NY', 'VM', 'NJ')
  ( SUBPARTITION p_ne_low VALUES LESS THAN (1000) LOCATION ('pnel.csv'),
    SUBPARTITION p_ne_average VALUES LESS THAN (10000) LOCATION ('pnea.csv'),
    SUBPARTITION p_ne_high VALUES LESS THAN (100000) LOCATION ('pneh.csv'),
    SUBPARTITION p_ne_extraordinary VALUES LESS THAN (MAXVALUE) LOCATION ('pnee.csv')
  ),
  PARTITION p_southeast VALUES ('FL', 'GA')
  ( SUBPARTITION p_se_low VALUES LESS THAN (1000) LOCATION ('psel.csv'),
    SUBPARTITION p_se_average VALUES LESS THAN (10000) LOCATION ('psea.csv'),
    SUBPARTITION p_se_high VALUES LESS THAN (100000) LOCATION ('pseh.csv'),
    SUBPARTITION p_se_extraordinary VALUES LESS THAN (MAXVALUE) LOCATION ('psee.csv')
  ),
  PARTITION p_northcentral VALUES ('SD', 'WI')
  ( SUBPARTITION p_nc_low VALUES LESS THAN (1000) LOCATION ('pncl.csv'),
    SUBPARTITION p_nc_average VALUES LESS THAN (10000) LOCATION ('pnca.csv'),
    SUBPARTITION p_nc_high VALUES LESS THAN (100000) LOCATION ('pnch.csv'),
    SUBPARTITION p_nc_extraordinary VALUES LESS THAN (MAXVALUE) LOCATION ('pnce.csv')
  ),
  PARTITION p_southcentral VALUES ('OK', 'TX') DEFAULT DIRECTORY data_dir2
  ( SUBPARTITION p_sc_low VALUES LESS THAN (1000) LOCATION ('pscl.csv'),
    SUBPARTITION p_sc_average VALUES LESS THAN (10000)
       DEFAULT DIRECTORY data_dir3 LOCATION ('psca.csv'),
    SUBPARTITION p_sc_high VALUES LESS THAN (100000)
       LOCATION ('psch1.csv','data_dir4:psch2.csv'),
    SUBPARTITION p_sc_extraordinary VALUES LESS THAN (MAXVALUE)
       LOCATION ('psce.csv')
  )
```

```
);
```

関連項目:

『Oracle Database VLDBおよびパーティショニング・ガイド』

親トピック: 外部表のパーティション化

20.15.7.4 パーティション化された外部表の変更

ALTER TABLE文を使用して、パーティション化された外部表の表レベルの外部パラメータを変更できますが、パーティション・レベルおよびサブパーティション・レベルのパラメータは変更できません。

外部ファイルの場所はPARTITION BYおよびSUBPARTITION BY句で指定します。パーティションの外部ファイルは、パーティションのPARTITION BY句で指定します。サブパーティションの外部ファイルは、サブパーティションのSUBPARTITION BY句で指定します。

唯一の例外として、パーティション化された外部表の作成の際には、表レベルのLOCATION句は指定できません。したがって、パーティション化された外部表を変更するALTER TABLE文に、表レベルのLOCATION句は追加できません。

パーティション・レベルでは、ADD、DROPおよびRENAME操作のみサポートされます。ALTER TABLE文で、既存のパーティションおよびサブパーティションの属性は変更できません。ただし、新しいパーティションやサブパーティションを追加するときに、PARTITION句またはSUBPARTITION句にDEFAULT DIRECTORYおよびLOCATION句を含めることができます。

1. 外部表の変更に必要な権限を持つユーザーとして、データベースに接続します。
2. ALTER TABLE文を発行します。

例20-26 パーティション化された外部表のパーティションの名前変更

この例は、orders_external_rangeというパーティション化されている外部表のパーティションを名前変更しています。

```
ALTER TABLE orders_external_range RENAME PARTITION pmax TO other_months;
```

親トピック: 外部表のパーティション化

20.15.8 外部表の削除

外部表の場合、DROP TABLE文によって、データベース内の表メタデータのみが削除されます。データベースの外に常駐する実際のデータには影響を与えません。

親トピック: 外部表の管理

20.15.9 外部表のシステム権限およびオブジェクト権限

外部表のシステム権限およびオブジェクト権限は、標準的な表のサブセットになります。

外部表に適用できるシステム権限は、次のものにかぎられます。

- ALTER ANY TABLE
- CREATE ANY TABLE
- DROP ANY TABLE
- READ ANY TABLE
- SELECT ANY TABLE

外部表に適用できるオブジェクト権限は、次のものにかぎられます。

- ALTER
- READ
- SELECT

ただし、ディレクトリには次のオブジェクト権限が対応付けられています。

- READ
- WRITE

外部表では、データソースのあるディレクトリ・オブジェクトに対してREAD権限が必要であり、同時に、不良ファイル、ログ・ファイルまたは廃棄ファイルのあるディレクトリ・オブジェクトに対してWRITE権限が必要です。

親トピック: 外部表の管理

20.16 ハイブリッド・パーティション表の管理

ハイブリッド・パーティション表は、一部のパーティションがデータベース内にあり一部のパーティションがデータベース外部の外部ファイル(オペレーティング・システム・ファイルやHadoop Distributed File System (HDFS)ファイルなど)内にあるパーティション表です。

ノート:

外部表に適用される制限事項は、ハイブリッド・パーティション表にも適用されます。

関連項目:

- 「外部表のパーティション化」
- ハイブリッド・パーティション表の管理の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: 表の管理

20.17 不変表の管理

不変表を使用すると、不正なデータ変更から保護できます。

- 不変表について
  不変表は、インサイダーによる不正なデータ変更および人為的エラーによる偶発的なデータ変更を防止する読取り専用表です。
- 不変表を管理するためのガイドライン
  ガイドラインに従って不変表に対して作業できます。
- 不変表の作成
  CREATE IMMUTABLE TABLE文を使用して不変表を作成します。不変表は指定されたスキーマに作成され、データ・ディクショナリに表メタデータが追加されます。
- 不変表の変更
  不変表の保存期間および不変表内の行の保存期間を変更できます。

- 不変表からの行の削除
  指定した保存期間外の行のみを不変表から削除できます。
- 不変表の削除
  不変表は、空の場合、またはその表の保存期間で定義されている期間が経過するまで変更されていなかった場合に削除できます。
- 不変表のデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューは、データベース内の不変表に関する情報を提供します。

親トピック: 表の管理

20.17.1 不変表について

不変表は、インサイダーによる不正なデータ変更および人為的エラーによる偶発的なデータ変更を防止する読取り専用表です。

不正な変更は、インサイダー資格証明へのアクセス権を持つ、セキュリティ侵害を受けた従業員または悪意のある従業員によって試行される可能性があります。

不変表には新しい行を追加できますが、既存の行は変更できません。保存期間は、不変表と不変表内の行の両方に対して指定する必要があります。指定した行の保存期間が経過すると、行は不要になります。不変表から削除できるのは、不要になった行のみです。

不変表には、システム生成の非表示列が含まれます。この列は、ブロックチェーン表の列と同じです。行が挿入されると、ORABCTAB_CREATION_TIME$列とORABCTAB_USER_NUMBER$列にNULL以外の値が設定されます。残りのシステム生成の非表示列の値は、NULLに設定されます。

不変表を使用する場合、既存のアプリケーションを変更する必要はありません。

表20-8 不変表とブロックチェーン表の違い

不変表	ブロックチェーン表
不変表は、ユーザー資格証明へのアクセス権を持つ、悪意のあるインサイダーまたはセキュリティ侵害を受けたインサイダーによる不正な変更を防ぎます。	ブロックチェーン表には、悪意のあるインサイダーまたはセキュリティ侵害を受けたインサイダーによる不正な変更の防止に加えて、次の機能があります。 ● Oracle Database ソフトウェアのバイパスによる不正な変更を検出します ● エンド・ユーザーの偽装およびユーザー名へのデータの挿入(認可なし)を検出します ● データの改ざんを防止し、データが表に実際に挿入されたことを確認します
行は連鎖されません。	最初の行を除き、各行は暗号化ハッシュを使用することで前の行に連鎖されています。行のハッシュ値は、行データとチェーン内の前の行のハッシュ値に基づいて計算されます。 行を変更するとチェーンが中断され、行が改ざんされたことが示され

不変表	ブロックチェーン表
	ます。
行の挿入では、コミット時に追加の処理は必要ありません。	行を連鎖させるには、コミット時に追加の処理時間が必要です。

関連トピック

- ブロックチェーン表の非表示列

親トピック: 不変表の管理

20.17.2 不変表を管理するためのガイドライン

ガイドラインに従って不変表に対して作業できます。

- 不変表の保存期間の指定
  不変表の保存期間を設定するには、CREATE IMMUTABLE TABLE文でNO DROP句を使用します。
- 不変表の行の保存期間の指定
  不変表の行の保存期間を指定するには、CREATE IMMUTABLE TABLE文でNO DELETE句を使用します。
- 不変表の制限事項
  不変表の使用には、特定の制限事項があります。

親トピック: 不変表の管理

20.17.2.1 不変表の保存期間の指定

不変表の保存期間を指定するには、CREATE IMMUTABLE TABLE文でNO DROP句を使用します。

不変表は、空でないかぎり、指定された保存期間内には削除できません。

次のいずれかの句を指定します。

- NO DROP

  不変表は削除できません。

- NO DROP UNTIL n DAYS IDLE

  最新の行の経過日数がn日未満である場合、不変表は削除できません。nの最小許容値は0です。ただし、不変表のセキュリティを確保するには、最小値を16以上に設定することをお薦めします。

  表の保存期間を0日に設定するには、初期化パラメータBLOCKCHAIN_TABLE_MAX_NO_DROPに0を設定する必要があります。この設定は、不変表をテストする場合に便利です。このパラメータに0を設定すると、許可される保存期間は0日のみになります。後でゼロ以外の保存期間を設定するには、BLOCKCHAIN_TABLE_MAX_NO_DROP初期化パラメータの値をリセットする必要があります。

ALTER TABLE文を使用して、不変表の保存期間を長くしてください。保存期間を短くすることはできません。

親トピック: 不変表を管理するためのガイドライン

20.17.2.2 不変表の行の保存期間の指定

不変表の行の保存期間を指定するには、CREATE IMMUTABLE TABLE文でNO DELETE句を使用します。

保存期間により、不変表から行を削除できるタイミングが制御されます。

次のいずれかのオプションを使用して、行の保存期間を指定します。

- `NO DELETE [LOCKED]`

  `NO DELETE`を使用すると、行は不変表から削除できなくなります。

  行が不変表から削除されないようにするには、`CREATE IMMUTABLE TABLE`文で`NO DELETE LOCKED`句を使用します。`LOCKED`キーワードは、行の保存設定を変更できないことを指定します。

- `NO DELETE UNTIL n DAYS AFTER INSERT [LOCKED]`

  行は、追加後n日経過するまで削除できません。この設定を変更して保存期間を長くするには、`NO DELETE UNTIL`句を指定した`ALTER TABLE`文を使用します。保存期間を短くすることはできません。

  nの最小値は16日間です。`LOCKED`を含めると、それ以降は行の保存を変更できなくなります。

親トピック: 不変表を管理するためのガイドライン

20.17.2.3 不変表の制限事項

不変表の使用には、特定の制限事項があります。

- 不変表では、データ型`ROWID`、`UROWID`、`LONG`、オブジェクト型、REF、VARRAY、ネストした表、`TIMESTAMP WITH TIME ZONE`、`TIMESTAMP WITH LOCAL TIME ZONE`、`BFILE`および`XMLType`はサポートされません。

  `XMLType`表もサポートされていません。

- ユーザー作成列の最大数は980です。
- 次の操作は、不変表ではサポートされていません。
  - CDBルートまたはアプリケーション・ルートでの不変表の作成
  - 行の更新、行のマージ、列の追加、列名の変更、列の削除およびパーティションの削除
  - 不変表の切捨て
  - シャード表
  - パラレルDMLを使用したデータのダイレクト・パス・ロードおよび挿入
  - フラッシュバック表
  - 更新操作を開始する`BEFORE ROW`トリガーを定義すること(その他のトリガーは許容されています)
  - 自動データ最適化(ADO)ポリシーの作成
  - Oracle Label Security (OLS)ポリシーの作成
  - `DBMS_REDEFINITION`パッケージを使用したオンライン再定義
  - 一時ロジカル・スタンバイおよびローリング・アップグレード

    不変表でのDDLおよびDMLはサポートされておらず、レプリケートされません。

  - ロジカル・スタンバイおよびOracle GoldenGate

    不変表でのDDLおよびDMLは、プライマリ・データベースでは成功しますが、スタンバイ・データベースにはレプリケートされません。

- 通常の表から不変表への変換、またはその逆の変換

- データベースのフラッシュバック・データベースおよびPoint-in-Timeリカバリでは、不変表を含むすべての表に対する変更が元に戻されます。たとえば、データベースがSCN 1000までフラッシュバックされた場合、またはバックアップがリストアされてSCN 1000までリカバリされた場合、SCN 1000以降のすべての変更がデータベースから削除されます。

  Oracle Databaseは、物理的および論理的な破損を元に戻すために必要になる可能性があるため、フラッシュバックおよびPoint-in-Timeリカバリ操作を妨げません。

- 不変表での保存ポリシーの正しい施行は、システム時間に依存しています。システム時間を変更する権限は広く付与せず、システム時間に対する変更は監査および確認する必要があります。

親トピック: 不変表を管理するためのガイドライン

20.17.3 不変表の作成

CREATE IMMUTABLE TABLE文を使用して、不変表を作成します。不変表は指定されたスキーマに作成され、データ・ディクショナリに表メタデータが追加されます。

COMPATIBLE初期化パラメータは19.11.0.0以上に設定されている必要があります。

自分のスキーマ内に不変表を作成するには、CREATE TABLEシステム権限が必要です。他のユーザーのスキーマ内に不変表を作成するには、CREATE ANY TABLEシステム権限が必要です。CREATE IMMUTABLE TABLE文では、NO DROP句およびNO DELETE句が必須です。

例20-27 不変表の作成

次の例では、ユーザー・スキーマにtrade_ledgerという名前の不変表を作成します。行は、挿入後100日経過するまで削除できません。この不変表は、非アクティブな状態で40日間経過した後にのみ削除できます。

```
CREATE IMMUTABLE TABLE trade_ledger (id NUMBER, luser VARCHAR2(40), value NUMBER)
       NO DROP UNTIL 40 DAYS IDLE
       NO DELETE UNTIL 100 DAYS AFTER INSERT;
```

親トピック: 不変表の管理

20.17.4 不変表の変更

不変表と不変表内の行の保存期間は変更できます。

不変表は、自分のスキーマに含まれているか、その不変表のALTERオブジェクト権限またはALTER ANY TABLEシステム権限のいずれかを持っている必要があります。

不変表の定義を変更するには、ALTER TABLE文をNO DROP句またはNO DELETE句とともに使用します。

不変表の保存期間を短くすることはできません。

例20-28 不変表の保存期間の変更

次の文では、不変表trade_ledgerの定義を変更して、最新の行が50日経過するまでは削除できないように指定します。NO DROP句の以前の値は30日でした。

```
ALTER TABLE trade_ledger NO DROP UNTIL 50 DAYS IDLE;
```

例20-29 不変表の行の保存期間の変更

次の文では、不変表trade_ledgerの定義を変更して、行の作成後に120日経過するまでは行を削除できないように指

定します。

```
ALTER TABLE trade_ledger NO DELETE UNTIL 120 DAYS AFTER INSERT;
```

親トピック: 不変表の管理

20.17.5 不変表からの行の削除

指定した保存期間外の行のみを不変表から削除できます。

SYSユーザーまたはスキーマの所有者は、不変表の行を削除できます。

指定した保存期間を超えるすべての行、または指定した時間より前に作成された不要な行を削除するには、DBMS_IMMUTABLE_TABLE.DELETE_EXPIRED_ROWSプロシージャを使用します。

例20-30 不変表からのすべての期限切れ行の削除

次の例では、SYSとして接続している場合、保存期間外にある不変表trade_ledger内のすべての行を削除します。削除された行数は、出力パラメータnum_rowsに格納されます。

```
DECLARE
   num_rows NUMBER;
BEGIN
   DBMS_IMMUTABLE_TABLE.DELETE_EXPIRED_ROWS('EXAMPLES','TRADE_LEDGER', NULL,
num_rows);
   DBMS_OUTPUT.PUT_LINE('Number_of_rows_deleted = ' || num_rows);
END;
/
```

例20-31 作成時間に基づいた適格な行の削除

次の例では、SYSとして接続している場合、現在のシステム日付の30日前に作成された不要な行を削除します。削除された行数は、出力パラメータnum_rowsに格納されます。

```
DECLARE
       num_rows NUMBER;
BEGIN
       DBMS_IMMUTABLE_TABLE.DELETE_EXPIRED_ROWS('EXAMPLES','TRADE_LEDGER', SYSDATE-30,
num_rows);
       DBMS_OUTPUT.PUT_LINE('Number_of_rows_deleted=' || num_rows);
END;
/
```

親トピック: 不変表の管理

20.17.6 不変表の削除

不変表は、空の場合、またはその表の保存期間で定義されている期間が経過するまで変更されていなかった場合に削除できます。

削除する不変表は、自分のスキーマに含まれているか、またはDROP ANY TABLEシステム権限を持っている必要があります。

DROP TABLE文を使用して、不変表を削除します。不変表を削除すると、データ・ディクショナリから定義が削除され、そのすべての行が削除され、表に定義されている索引およびトリガーも削除されます。

次の文は、examplesスキーマ内のtrade_ledgerという名前の不変表を削除します。

```
DROP TABLE examples.trade_ledger;
```

親トピック: 不変表の管理

20.17.7 不変表のデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューは、データベース内の不変表に関する情報を提供します。

不変表の詳細は、DBA_IMMUTABLE_TABLES、USER_IMMUTABLE_TABLESまたはALL_IMMUTABLE_TABLESのいずれかのビューを問い合せます。情報には、行の保存期間と表の保存期間が含まれます。DBAビューはデータベース内のすべての不変表を表し、ALLビューはユーザーがアクセスできるすべての不変表を表し、USERビューはユーザーが所有する不変表に制限されます。

例20-32 不変表の情報の表示

次の問合せでは、examplesスキーマの不変表trade_ledgerの詳細が表示されます。

```
SELECT row_retention "Row Retention Period", row_retention_locked "Row Retention
Lock", table_inactivity_retention "Table Retention Period"
FROM dba_immutable_tables
WHERE table_name = 'TRADE_LEDGER';
Row Retention Period Row Retention Locked Table Retention Period
-------------------- -------------------- ----------------------
                 110                   NO                     16
```

親トピック: 不変表の管理

20.18 ブロックチェーン表の管理

ブロックチェーン表は、重要なアクション、資産、エンティティおよびドキュメントを記録するデータを、犯罪者、ハッカーおよび不正行為による不正な変更または削除から保護します。ブロックチェーン表は、データベースを使用した不正な変更を防ぎ、データベースをバイパスする不正な変更を検出します。

- ブロックチェーン表について
  ブロックチェーン表は、行を複数のチェーンにまとめて編成する挿入専用の表です。最初の行を除いて、チェーン内の各行は、暗号化ハッシュを使用することでチェーン内の前の行に連鎖されています。
- ブロックチェーン表の管理のガイドライン
  ガイドラインに従って、ブロックチェーン表を作成および使用できます。
- ブロックチェーン表の作成
  ブロックチェーン表は、CREATE BLOCKCHAIN TABLE文を使用して作成します。この文により、指定したスキーマにブロックチェーン表が作成され、データ・ディクショナリに表メタデータが作成されます。
- ブロックチェーン表の変更
  ブロックチェーン表とブロックチェーン表内の行の保存期間は変更できます。
- ブロックチェーン表の行の署名に使用する証明書の追加
  証明書は、ブロックチェーン表の行の署名を検証するために使用できます。
- 証明書の削除
  ブロックチェーン表の行の署名を検証するための証明書は、不要になったときには削除してください。
- ブロックチェーン表の行への署名の追加
  行に署名することで、すでに作成されている行にユーザー署名を設定します。署名はオプションですが、署名により改ざんに対する追加のセキュリティが得られます。
- ブロックチェーン表のデータの検証
  PL/SQLプロシージャDBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSでは、ブロックチェーン表の行が挿入後に変

更されていないことを確認します。改ざん耐性があることは、ブロックチェーン表の主要な要件です。

- ブロックチェーン表の整合性の検証
  ブロックチェーン表のデータが損なわれていないことを継続的に検証することで、ブロックチェーン表の整合性を維持します。
- ブロックチェーン表からの行の削除
  保存期間外の行のみをブロックチェーン表から削除できます。
- ブロックチェーン表の削除
  ブロックチェーン表は、行が含まれていない場合、またはその保存期間により定義された期間に変更されていない場合に削除できます。
- 行のハッシュを計算するための行内容のデータ形式の判別
  行のハッシュ値を計算するために、DBMS_BLOCKCHAIN_TABLE.GET_BYTES_FOR_ROW_HASHプロシージャを使用して行内容のデータ形式を判別します。
- 行の署名を計算するためのデータ形式の判別
  行の署名の計算に使用される行内容のデータ形式を判別できます。行署名は、その行のハッシュ値に基づいて計算されます。
- ブロックチェーン表のデータのバイト値の表示
  ブロックチェーン表のデータのバイト値(行と列の両方)を取得できます。
- ブロックチェーン表のデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューは、ブロックチェーン表に関する情報を提供します。

親トピック: 表の管理

20.18.1 ブロックチェーン表について

ブロックチェーン表は、行を複数のチェーンにまとめて編成する挿入専用の表です。最初の行を除いて、チェーン内の各行は、暗号化ハッシュを使用することでチェーン内の前の行に連鎖されています。

ブロックチェーン表内の行には、改ざん耐性があります。各行には、その行のデータとチェーン内の前の行のハッシュ値に基づいた暗号化ハッシュ値が含まれています。ある行が改ざんされると、その行のハッシュ値が変わるため、チェーン内の次の行のハッシュ値も変わります。不正防止の強化のために、行にオプションのユーザー署名を追加できます。ブロックチェーン表の行に署名する場合は、デジタル証明書を使用する必要があります。ブロックチェーン表のチェーンの検証時、データベースでは行署名を検証するための証明書が必要になります。

ブロックチェーン表は、索引を作成することも、パーティション化することもできます。ブロックチェーン表から行を削除するかどうか、またいつ削除するかを制御できます。また、ブロックチェーン表の削除を許可するかどうかを制御することもできます。ブロックチェーン表は、トランザクションおよび問合せ内の(通常の)表とともに使用できます。

ブロックチェーン表は、参加者がOracle Databaseを信頼するが、データが改ざんされていないことを検証する手段が必要なブロックチェーン・アプリケーションを実装するために使用できます。この参加者は、Oracle Databaseを信頼して、検証可能で改ざん耐性のあるトランザクションのブロックチェーンを維持する様々なデータベース・ユーザーです。すべての参加者には、ブロックチェーン表にデータを挿入する権限が必要です。ブロックチェーンの内容は、アプリケーションによって定義および管理します。検証可能な暗号保護されたデータ管理手法を備えた信頼できるプロバイダを活用することにより、そのようなアプリケーションは分散型コンセンサスの要件を回避できます。これにより、分散型ピア・ツー・ピア・ブロックチェーンの保護のほとんどが提供されますが、分散型コンセンサスを使用するピア・ツー・ピア・ブロックチェーンと比較して、スループットが大幅に向上し、トランザクション待機時間が短縮されます。

ブロックチェーン表は、集中管理アプリケーションにとってデータの不変性が重要で、現行トランザクションおよび履歴トランザクショ

ンの改ざん不可能な状態の元帳を維持する必要がある場合に使用します。ブロックチェーン表はビルディング・ブロックです。集中管理ブロックチェーンを実装するタスクを実行するのに必要なトリガーまたはストアド・プロシージャを定義する必要があります。情報ライフサイクル管理(ILM)は、ブロックチェーン表のデータのライフサイクルを管理するために使用されます。ブロックチェーン表の1つ以上のパーティションのデータが古い場合は、ILM手法を使用してコストの低い記憶域に移動できます。

- ブロックチェーン表を使用する利点
  ブロックチェーン表は、犯罪者、ハッカーおよび不正行為からのデータ保護に焦点を当てて、企業や政府が直面するデータ保護の課題に対処します。
- ブロックチェーン表での行の連鎖
  ブロックチェーン表の行は、チェーン内の前の行と連鎖しています。また、行のチェーンは、すべての参加者による検証が可能です。
- ブロックチェーン表の非表示列
  ブロックチェーン表の各行には、データベースによって値が管理される非表示列が含まれています。

親トピック: ブロックチェーン表の管理

20.18.1.1 ブロックチェーン表を使用する利点

ブロックチェーン表は、犯罪者、ハッカーおよび不正行為からのデータ保護に焦点を当てて、企業や政府が直面するデータ保護の課題に対処します。

従来のデータ・セキュリティ技術では、許可されていないユーザーによる重要なデータへのアクセスを防止することに重点を置いていました。この手法には、パスワード、権限、暗号化およびファイアウォールの使用が含まれます。ブロックチェーン表は、重要なアクション、資産、エンティティおよびドキュメントを記録するデータの不正な変更または削除を防止することにより、データ・セキュリティを強化します。重要なレコードを不正に変更すると、資産の紛失、ビジネスの喪失、および法的な問題が発生する可能性があります。

ブロックチェーン表には、次の利点があります。

- 盗まれたインサイダー内部資格証明を使用するインサイダーまたは犯罪者によるデータの不正な変更を防止します

  これは、表を挿入専用にすることで実現されます。データベースでは、ユーザーが次のアクションを実行する機能を削除することで既存のデータを変更または削除することはできません。

  - 行の更新または削除
  - ブロックチェーン表定義の変更
  - ブロックチェーン表から更新可能な表への変換、またはその逆の変換
  - データベース・ディクショナリ内の表メタデータの変更

- ハッカーによる検出されないデータ変更を防止します

  これは、次の方法を使用することにより実現されます。

  - データベースで計算された行ハッシュを使用した、挿入時のブロックチェーン表の行の暗号化チェーン。これには、現在の行のデータと前の行のハッシュ、およびチェーンを検証するための対応するPL/SQLファンクションが含まれます。

    行を変更するとチェーンが中断され、行が改ざんされたことが示されます。暗号化チェーンは、ハッカーがデータベースまたはオペレーティング・システムを制御している場合でも効果的です。SQLをバイパスするデータ変更は、Oracle提供のPL/SQLファンクションを使用して検出できます。

- リクエストに応じて生成され、否認防止のためにデータベース・スキーマ所有者の秘密キーで署名されたブロックチェーン表の暗号ダイジェスト

  暗号ダイジェストは、ブロックチェーン表の内容(表内のすべてのチェーンの最後の行のメタデータ列)に基づいて計算されます。そのため、データを変更するとダイジェスト値が変更されます。暗号ダイジェストを定期的に計算して、安全なリポジトリまたは関係者に配布できます。データベース・オペレータまたはプロ級のハッカーによって行われた不正な変更の隠蔽を検出するには、Oracle提供のPL/SQLファンクションを使用して、2つのタイムスタンプ間の行の範囲のダイジェストを検証します。

- 盗まれたエンド・ユーザーの資格証明を使用した、検出されない不正なデータ変更を防止します

  これは、次の方法を使用することにより実現されます。

  - Oracle提供のPL/SQLファンクションを使用して行データに署名を挿入することによる、エンド・ユーザーによる新しいデータの暗号署名

    エンド・ユーザーは、デジタル証明書と秘密キーを使用して、ブロックチェーン表に挿入する行に暗号的に署名できます。これにより、別のエンド・ユーザー、盗まれたエンド・ユーザー資格証明を持つハッカー、またはアプリケーション資格証明チェックをバイパスする権限のないユーザーによる偽装を防ぐことができます。また、記録されているデータが転送中またはアプリケーションで変更されていないこと、およびデータが実際にエンド・ユーザーによって挿入されたことを確認するためにも役立ちます。データが自分の秘密キーで署名され、ユーザーが提供するPKI証明書を使用して公開キーで検証された場合、エンド・ユーザーはデータが他のユーザーによって挿入されたと主張できないため、ユーザーの署名は否認防止を提供します。

  - 表スキーマ所有者によるブロックチェーン表ダイジェストの暗号署名

    ブロックチェーン表の署名付きダイジェストは、エンド・ユーザー・データが受信および記録されたこと、エンド・ユーザーが提供した署名が記録されたデータと一致すること、および暗号ダイジェストに新しいデータが含まれていることを保証します。これにより、エンド・ユーザーによるデータの否認を防ぎます。

- データベースを使用した不正な変更を防ぎ、データベースをバイパスする不正な変更を検出します。
- ブロックチェーン・テクノロジをOracleデータベースに統合することで、既存のアプリケーションへの変更を最小限に抑え、新しいインフラストラクチャ要件なしでデータ保護を強化します。
- 問合せとトランザクションでデータベース表と通常の表を混在させることができます。
- 暗号的に保護されたデータに対して、分析機能などの高度なOracleデータベース機能を使用できます。
- ユーザーは、監査証跡に対してペアのブロックチェーン表を作成し、元の表に対してトリガーされたストアド・プロシージャを使用してブロックチェーン表に履歴を記録することにより、通常の表のすべてのトランザクションの履歴を保持できます。

親トピック: ブロックチェーン表について

20.18.1.2 ブロックチェーン表での行の連鎖

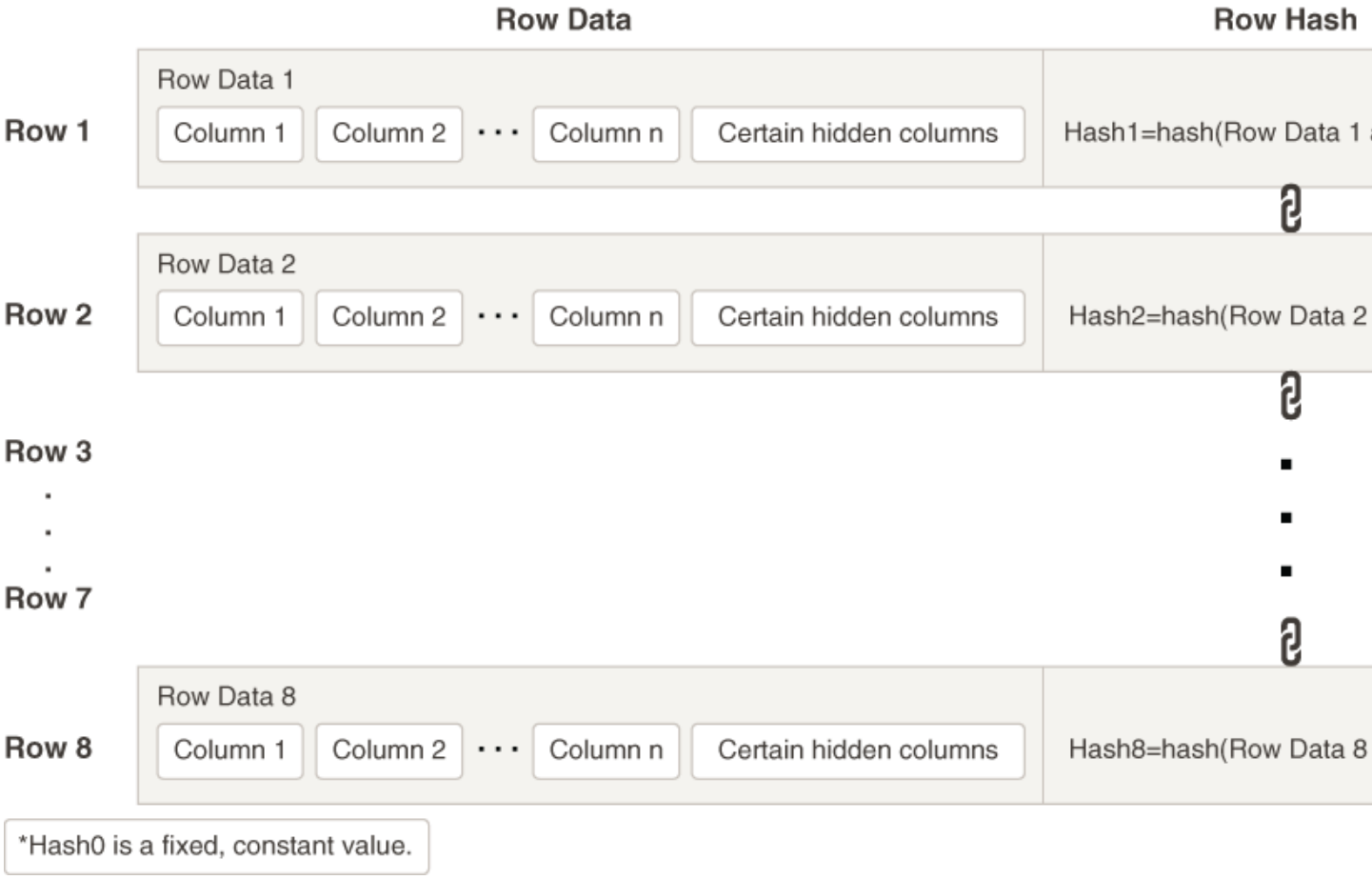
ブロックチェーン表の行は、チェーン内の前の行と連鎖しています。また、行のチェーンは、すべての参加者による検証が可能です。

Oracle Real Application Clusters (Oracle RAC)インスタンスごとに、ブロックチェーン表には、32のチェーン(0から31の範囲)が格納されています。1つのチェーンには複数の行が含まれています。また、チェーンはインスタンスIDとチェーンIDの一意の組合せによって識別されます。1つの行は、ユーザーの列と、データベースによって作成される非表示列で構成されています。行が挿入されると、チェーン内で一意の順序番号が割り当てられ、チェーン内の前の行にリンクされます。行の順序番号は、チェーン内の前の行の順序番号より1つ上です。チェーン内の最初の行を除く各行には、一意の前の行があります。行は、インスタンスID、チェーンIDおよび順序番号の組合せを使用して、一意に識別できます。インスタンスID、チェーンID、順序番号の組

合せに対する索引を作成することをお薦めします。

図20-1に、行が連鎖される方法を示します。行の行データは、ユーザー列と特定の非表示列で構成されます。行のハッシュ値は、行データとチェーン内の前の行のハッシュ値に基づいて計算されます。ハッシュ値の計算には、SHA2-512ハッシュ・アルゴリズムが使用されます。各行は、ハッシュ値を使用して連鎖されています。

図20-1 ブロックチェーン表の単一チェーンの行



単一のトランザクションで、複数のブロックチェーン表に行を挿入できます。1つのトランザクションによって挿入されたブロックチェーン表の行は同じチェーンに追加され、チェーン上の位置はブロックチェーン表に挿入された順序に対応しています。トランザクションのコミット時に、データベースで行のチェーンが自動的に選択されます。

複数のユーザーがブロックチェーン表内の同じチェーンに同時に行を挿入した場合、行の追加順序は、該当する行を挿入したトランザクションのコミット順序によって決まります。

行は、トランザクションのコミット時にブロックチェーンにリンクされます。単一のトランザクションに多数の行を挿入すると、コミット待機時間が長くなります。したがって、単一のトランザクションに非常に多くの行を挿入することは避けるようにお薦めします。

関連トピック

- ブロックチェーン表参照

親トピック: ブロックチェーン表について

20.18.1.3 ブロックチェーン表の非表示列

ブロックチェーン表の各行には、データベースによって値が管理される非表示列が含まれています。

非表示列は、挿入した行のコミット時に移入されます。行の順序付けを実装するため、また、データに改ざん耐性があることを確認するために使用されます。非表示列に対して索引を作成できます。非表示列は、問合せに列名を明示的に含めることで

のみ表示できます。

表20-9 ブロックチェーン表の非表示列

列名	データ型	説明
ORABCTAB_INST_ID$	NUMBER (22)	行の挿入先のデータベース・インスタンスのインスタンス ID。
ORABCTAB_CHAIN_ID$	NUMBER (22)	行が挿入されるデータベース・インスタンス内のチェーンのチェーン ID。チェーン ID の有効な値は 0 から 31 です。
ORABCTAB_SEQ_NUM$	NUMBER(22)	チェーンの行の順序番号。ブロックチェーン表のチェーンに挿入された各行には、1 から始まる一意の順序番号が割り当てられます。行の順序番号は、チェーン内の前の行の順序番号より 1 つ上です。この列を使用して、欠落している行を検出できます。 インスタンス ID、チェーン ID および順序番号の組合せによって、ブロックチェーン表内の行が一意に識別されます。
ORABCTAB_CREATION_TIME$	TIMESTAMPWITHTIMEZONE	行が作成された時間(UTC 形式)。
ORABCTAB_USER_NUMBER$	NUMBER (22)	行を挿入したデータベース・ユーザーのユーザーID。
ORABCTAB_HASH$	RAW(2000)	行のハッシュ値。ハッシュ値は、行の行内容およびチェーン内の前の行のハッシュ値に基づいて計算されます。
ORABCTAB_SIGNATURE$	RAW(2000)	行のユーザー署名。署名は、行のハッシュ値を使用して計算されます。
ORABCTAB_SIGNATURE_ALG$	NUMBER(22)	署名される行のユーザー署名を生成するために使用される署名アルゴリズム。
ORABCTAB_SIGNATURE_CERT$	RAW(16)	署名される行に対する署名に関連付けられた証明書の GUID。
ORABCTAB_SPARE$	RAW(2000)	この列は、今後使用するために予約されています。

親トピック: ブロックチェーン表について

20.18.2 ブロックチェーン表を管理するためのガイドライン

ガイドラインに従って、ブロックチェーン表を作成および使用できます。

ノート:

表作成のガイドラインは、ブロックチェーン表にも適用されます。この項では追加のガイドラインについて説明します。

- データベース・インスタンス内のチェーンごとに、現行のハッシュとそれに対応する順序番号を、データベースの外部に定期的に保存します。これにより、ブロックチェーン表内のチェーンが短縮または上書きされていないことを確認できます。
- Oracle Data Guard環境では、データの損失を回避するために、最大保護モードまたは最大化要請モードの使用を検討してください。

- ブロックチェーン表の保存期間の指定
  ブロックチェーン表の保存期間を指定するには、CREATE BLOCKCHAIN TABLE文でNO DROP句を使用します。
- ブロックチェーン表の行の保存期間の指定
  ブロックチェーン表の行の保存期間を指定するには、CREATE BLOCKCHAIN TABLE文でNO DELETE句を使用します。
- ブロックチェーン表の制限事項
  ブロックチェーン表の使用には、特定の制限事項があります。

親トピック: ブロックチェーン表の管理

20.18.2.1 ブロックチェーン表の保存期間の指定

ブロックチェーン表の保存期間を指定するには、CREATE BLOCKCHAIN TABLE文でNO DROP句を使用します。

ブロックチェーン表に行が含まれている場合、指定された保存期間内には削除できません。

保存期間を指定するには、次のいずれかの句を含めます。

- NO DROP

  ブロックチェーン表は削除できません。

- NO DROP UNTIL n DAYS IDLE

  最新の行の経過日数がn日未満である場合、ブロックチェーン表は削除できません。nの最小許容値は0です。ただし、ブロックチェーン表のセキュリティを確保するには、最小値を16以上に設定することをお薦めします。

  表の保存期間を0日に設定するには、BLOCKCHAIN_TABLE_MAX_NO_DROP動的初期化パラメータに0を設定する必要があります。この設定は、ブロックチェーン表をテストする場合に便利です。このパラメータに0を設定すると、許可される保存期間は0日のみになります。後でゼロ以外の保存期間を設定するには、BLOCKCHAIN_TABLE_MAX_NO_DROPパラメータの値をリセットする必要があります。

  ALTER TABLE文を使用して、ブロックチェーン表の保存期間を長くしてください。保存期間を短くすることはできません。

親トピック: ブロックチェーン表の管理のガイドライン

20.18.2.2 ブロックチェーン表の行の保存期間の指定

ブロックチェーン表の行の保存期間を指定するには、CREATE BLOCKCHAIN TABLE文でNO DELETE句を使用します。

保存期間により、ブロックチェーン表から行を削除できるタイミングが制御されます。次のいずれかのオプションを使用して、保存期間を指定します。

- `NO DELETE [LOCKED]`

  `NO DELETE`を使用すると、行はブロックチェーン表から削除できなくなります。

  行がブロックチェーン表から削除されないようにするには、`CREATE BLOCKCHAIN TABLE`文で`NO DELETE LOCKED`句を使用します。`LOCKED`キーワードは、行の保存設定を変更できないことを指定します。

- `NO DELETE UNTIL n DAYS AFTER INSERT [LOCKED]`

  行は、追加後n日経過するまで削除できません。`ALTER TABLE`に`NO DELETE UNTIL`句を指定することで、この設定を変更して保存期間を長くすることもできます。保存期間を短くすることはできません。

  nの最小値は16日間です。`LOCKED`を含めると、それ以降は行の保存を変更できなくなります。

親トピック: ブロックチェーン表の管理のガイドライン

20.18.2.3 ブロックチェーン表の制限事項

ブロックチェーン表の使用には、特定の制限事項があります。

- 次のデータ型は、ブロックチェーン表ではサポートされていません: `ROWID`、`UROWID`、`LONG`、オブジェクト型、REF、VARRAY、ネストした表、`TIMESTAMP WITH TIME ZONE`、`TIMESTAMP WITH LOCAL TIME ZONE`、`BFILE`および`XMLType`。

  `XMLType`表はサポートされていません。

- ユーザー作成列の最大数は980です。
- 次の操作は、ブロックチェーン表ではサポートされていません。
  - CDBルートまたはアプリケーション・ルートでのブロックチェーン表の作成
  - 行の更新およびマージ
  - 列の追加、削除および名前変更
  - ブロックチェーン表の切捨て
  - パーティションの削除
  - パラレルDMLを使用したデータの挿入
  - シャード表
  - 分散トランザクション

    Active Data Guard DMLリダイレクトを使用したブロックチェーン表へのデータの挿入は、インメモリー・データベース・リンクを使用した分散トランザクション・フレームワークに内部的に依存しているため、サポートされていません。

  - ダイレクト・パス・ロード
  - フラッシュバック表
  - 更新操作を開始する`BEFORE ROW`トリガーを定義すること(その他のトリガーは許容されています)
  - XAトランザクション
  - 自動データ最適化(ADO)ポリシーの作成
  - Oracle Virtual Private Database (VPD)ポリシーの作成

- Oracle Label Security (OLS)ポリシーの作成
- DBMS_REDEFINITIONパッケージを使用したオンライン再定義
- Oracle Data Pumpエクスポートおよびインポート

  ブロックチェーン表は、システム生成の非表示列なしで、通常の表としてエクスポートおよびインポートされます。
- 一時ロジカル・スタンバイおよびローリング・アップグレード

  ブロックチェーン表でのDDLおよびDMLはサポートされておらず、レプリケートされません。
- ロジカル・スタンバイおよびOracle GoldenGate

  ブロックチェーン表でのDDLおよびDMLは、プライマリ・データベースでは成功しますが、スタンバイ・データベースにはレプリケートされません。
- 通常の表からブロックチェーン表への変換、またはその逆の変換

- データベースのフラッシュバック・データベースおよびPoint-in-Timeリカバリでは、ブロックチェーン表を含むすべての表に対する変更が元に戻されます。たとえば、データベースがSCN 1000までフラッシュバックされた場合、またはバックアップがリストアされてSCN 1000までリカバリされた場合、SCN 1000以降のすべての変更がデータベースから削除されます。Oracle Databaseは、物理的および論理的な破損を元に戻すために必要になる可能性があるため、フラッシュバックおよびPoint-in-Timeリカバリ操作を妨げません。ブロックチェーン表のデータ損失を検出するには、署名済ブロックチェーン・ダイジェストを定期的に公開する必要があります。
- ブロックチェーン表での保存ポリシーの正しい施行は、システム時間に依存しています。システム時間を変更する権限は広く付与せず、システム時間に対する変更は監査および確認する必要があります。

親トピック: ブロックチェーン表の管理のガイドライン

20.18.3 ブロックチェーン表の作成

ブロックチェーン表は、CREATE BLOCKCHAIN TABLE文を使用して作成します。この文により、指定したスキーマにブロックチェーン表が作成され、データ・ディクショナリに表メタデータが作成されます。

ノート:

ブロックチェーン表は、ルート・コンテナ内およびアプリケーション・ルート・コンテナ内には作成できません。

自分のスキーマ内にブロックチェーン表を作成するには、CREATE TABLEシステム権限が必要です。他のユーザーのスキーマ内にブロックチェーン表を作成するには、CREATE ANY TABLEシステム権限が必要です。

CREATE BLOCKCHAIN TABLE文では、NO DROP、NO DELETE、HASHING USINGおよびVERSION句が必須です。

例20-33 単純なブロックチェーン表の作成

この例では、スキーマに指定の列を持つbank_ledgerという名前のブロックチェーン表を作成します。行は削除できません。このブロックチェーン表は、非アクティブな状態で31日経過した後にのみ削除できます。

```
CREATE BLOCKCHAIN TABLE bank_ledger (bank VARCHAR2(128), deposit_date DATE,
deposit_amount NUMBER)
         NO DROP UNTIL 31 DAYS IDLE
         NO DELETE LOCKED
         HASHING USING "SHA2_512" VERSION "v1";
```

例20-34 パーティション化されたブロックチェーン表の作成

この例では、指定された列およびパーティションを使用してブロックチェーン表bctab_partを作成します。この表は、非アクティブな状態で16日間経過した後にのみ削除できます。行は、挿入後25日経過するまで削除できません。ブロックチェーン表は、trans_date列でパーティション化されます。

```
CREATE BLOCKCHAIN TABLE bctab_part (trans_id number primary key, sender varchar2(50),
recipient varchar2(50), trans_date DATE, amount number)
     NO DROP UNTIL 16 DAYS IDLE
     NO DELETE UNTIL 25 DAYS AFTER INSERT
     HASHING USING "SHA2_512" VERSION "v1"
     PARTITION BY RANGE(trans_date)
      (PARTITION p1 VALUES LESS THAN (TO_DATE('30-09-2019','dd-mm-yyyy')),
       PARTITION p2 VALUES LESS THAN (TO_DATE('31-12-2019','dd-mm-yyyy')),
       PARTITION p3 VALUES LESS THAN (TO_DATE('31-03-2020','dd-mm-yyyy')),
       PARTITION p4 VALUES LESS THAN (TO_DATE('30-06-2020','dd-mm-yyyy'))
      );
```

例20-35 ブロックチェーン表の列の表示(非表示列を含む)

この例では、非表示列も含めて、ブロックチェーン表の列の詳細を表示します。

```
    Col ID Column Name                    Data Type                      Data Length
---------- ------------------------------ ------------------------------ -----------
         1 BANK                           VARCHAR2                              128
         2 DEPOSIT_DATE                   DATE                                    7
         3 DEPOSIT_AMOUNT                 NUMBER                                 22
         4 ORABCTAB_INST_ID$              NUMBER                                 22
         5 ORABCTAB_CHAIN_ID$             NUMBER                                 22
         6 ORABCTAB_SEQ_NUM$              NUMBER                                 22
         7 ORABCTAB_CREATION_TIME$        TIMESTAMP(6) WITH TIME ZONE            13
         8 ORABCTAB_USER_NUMBER$          NUMBER                                 22
         9 ORABCTAB_HASH$                 RAW                                  2000
        10 ORABCTAB_SIGNATURE$            RAW                                  2000
        11 ORABCTAB_SIGNATURE_ALG$        NUMBER                                 22
        12 ORABCTAB_SIGNATURE_CERT$       RAW                                    16
        13 ORABCTAB_SPARE$                RAW                                  2000
13 rows selected.
```

親トピック: ブロックチェーン表の管理

20.18.4 ブロックチェーン表の変更

ブロックチェーン表とブロックチェーン表内の行の保存期間は変更できます。

ブロックチェーン表定義の変更中は、保存期間を短縮できません。たとえば、ブロックチェーン表を作成して、保存期間を30日に設定したとします。この設定を後で変更して保存期間を20日に設定することはできません。

- ALTER TABLE文は、NO DROP句またはNO DELETE句とともに使用します。NO DELETE LOCKED句を使用することで、ブロックチェーン表から行を削除できないように指定します。

  次の文では、ブロックチェーン表bank_ledgerの定義を変更して、最新の行が16日経過するまでは削除できないように指定します。

  ```
  ALTER TABLE bank_ledger NO DROP UNTIL 16 DAYS IDLE;
  ```

  次の文では、ブロックチェーン表bctabの定義を変更して、行の作成後に20日経過するまでは行を削除できないように指定します。LOCKED句は、この設定の変更を禁止することを示しています。

  ```
  ALTER TABLE bctab NO DELETE UNTIL 20 DAYS AFTER INSERT LOCKED;
  ```

親トピック: ブロックチェーン表の管理

20.18.5 ブロックチェーン表の行の署名に使用する証明書の追加

証明書は、ブロックチェーン表の行の署名を検証するために使用できます。

X.509デジタル証明書を認証局(CA)から取得する必要がありますこの証明書は、データベースにBLOBとして追加され、ブロックチェーン表の1つ以上の行の署名を追加および検証するために使用されます。複数の証明書を使用して、1つのブロックチェーン表内の行に署名することもできます。デジタル証明書の操作には、OpenSSL APIを使用します。

追加するデジタル証明書は、BLOBとしてデータベースに格納する必要があります。このBLOBは、ディレクトリ・オブジェクトに収容できます。

- 証明書を追加するには、DBMS_USER_CERTS.ADD_CERTIFICATEプロシージャを使用します。

  データベースに証明書を追加すると、一意の証明書IDが割り当てられます。このIDは、DBMS_USER_CERTS.ADD_CERTIFICATEプロシージャの出力です。証明書IDは、ブロックチェーン表の行の署名を追加および検証するときに使用します。この証明書IDを忘れてしまうと、それに関連付けられたデジタル証明書を使用できなくなります。

例20-36 デジタル証明書のデータベースへの追加

この例では、デジタル証明書を追加します。この証明書はバイナリ形式でファイルu1_cert.derに保存されます。このファイルは、MY_DIRディレクトリ・オブジェクトに保存されます。DBMS_LOBパッケージ内のプロシージャを使用して、証明書を開き、その内容を変数bufferに読み込みます。変数cert_idに、プロシージャの出力(証明書ID)を保存します。

```
DECLARE
        file       BFILE;
        buffer     BLOB;
        amount     NUMBER := 32767;
        cert_id  RAW(16);
BEGIN
        file := BFILENAME('MY_DIR', 'u1_cert.der');
        DBMS_LOB.FILEOPEN(file);
        DBMS_LOB.READ(file, amount, 1, buffer);
        DBMS_LOB.FILECLOSE(file);
        DBMS_USER_CERTS.ADD_CERTIFICATE(buffer, cert_id);
        DBMS_OUTPUT.PUT_LINE('Certificate ID = ' || cert_id);
END;
/
Certificate ID = 9D267F1C280B60D8E053E5885A0A25FA
PL/SQL procedure successfully completed.
```

例20-37 証明書に関する情報の表示

この例では、DBA_CERTIFICATESデータ・ディクショナリ・ビューを問い合せることで、既存の証明書に関する情報を表示します。それ以外にも、証明書に関する情報はCDB_CERTIFICATESビューとUSER_CERTIFICATESビューに含まれています。

```
SELECT user_name, distinguished_name, UTL_RAW.LENGTH(certificate_id) CERT_ID_LEN,
DBMS_LOB.GETLENGTH(certificate) CERT_LEN
FROM DBA_CERTIFICATES ORDER BY user_name;
USER_NAME  DISTINGUISHED_NAME
---------- --------------------------------------------------------------------------
CERT_ID_LEN   CERT_LEN
------------ ----------
U1         CN=USER1,OU=Americas,O=oracle,L=redwoodshores,ST=CA,C=US
           16         835
U2         CN=USER2,OU=IT-Department,O=Global-Security,L=London,ST=London,C=GB
           16        1465
```

親トピック: ブロックチェーン表の管理

20.18.6 証明書の削除

ブロックチェーン表の行の署名を検証するための証明書は、不要になったときには削除してください。

データベースから証明書を削除するには、SYSBA権限が付与されているか、証明書の所有者になっている必要があります。また、データベースに証明書を追加したときに生成されたGUIDがわかっている必要もあります。

- 証明書を削除するには、DBMS_USER_CERTS.DROP_CERTIFICATEプロシージャを使用します。

例20-38 証明書の削除

この例では、GUIDが9CCC45ABA31D5DC2E0532A26C40A860Fの証明書を削除します。

```
declare
    certificate_guid RAW(16):='9CCC45ABA31D5DC2E0532A26C40A860F';
begin
    DBMS_USER_CERTS.DROP_CERTIFICATE(certificate_guid);
end;
```

親トピック: ブロックチェーン表の管理

20.18.7 ブロックチェーン表の行への署名の追加

行に署名することで、すでに作成されている行にユーザー署名を設定します。署名はオプションですが、署名により改ざんに対する追加のセキュリティが得られます。

Oracle Databaseは、現在のユーザーが更新する行とハッシュを所有していることを検証して、指定があれば保存されている行のハッシュ値と照合します。ブロックチェーン表の行に署名を追加するときには、デジタル証明書を使用する必要があります。署名は、指定したデジタル証明書と署名アルゴリズムを使用して検証されます。

サポートされている署名アルゴリズムは、SIGN_ALGO_RSA_SHA2_256、SIGN_ALGO_RSA_SHA2_384およびSIGN_ALGO_RSA_SHA2_512です。

ブロックチェーン表の行に署名を追加するには、その行の既存の署名がNULLである必要があります。ブロックチェーン表に対するINSERT権限が必要です。

- DBMS_BLOCKCHAIN_TABLE.SIGN_ROWプロシージャを使用して、既存の行に署名を追加します。

ノート:

DBMS_BLOCKCHAIN_TABLE.SIGN_ROW プロシージャはプラガブル・データベース(PDB)に固有の情報に依存し、Oracle Data Pump 以外のユーザー、アプリケーションまたはユーティリティによって現在の PDB に挿入された行にのみ適用されます。たとえば、PDB my_pdb1 のブロックチェーン表に行を挿入し、トランザクションをコミットし、Oracle Data Pump を使用してブロックチェーン表をエクスポートし、Oracle Data Pump を使用してブロックチェーン表を PDB my_pdb2 にインポートするとします。DBMS_BLOCKCHAIN_TABLE.SIGN_ROW プロシージャを使用して PDB my_pdb2 でこの行に署名しようとすると、例外が発生します。

Oracle Data Pump を使用してブロックチェーン表のコピーを作成する前に、署名する必要があるブロックチェーン表のすべての行に署名する必要があります。

例20-39 ブロックチェーン表の行の署名

この例では、bank_ledger表の銀行名が「my_bank」の行に署名を追加します。この表はexamplesスキーマ内にありま

す。署名は、標準のOpenSSLコマンドを使用してデータベースの外部で計算され、ファイルulr1_sign.datにバイナリ形式で保存されます。署名アルゴリズムには、DBMS_BLOCKCHAIN_TABLE.SIGN_ALGO_RSA_SHA2_512が使用されます。変数cert_guidは、証明書のGUIDを表します。この証明書は、データベースに追加して、署名の生成に使用したものです。

```
DECLARE
        inst_id binary_integer;
        chain_id binary_integer;
        sequence_no binary_integer;
        file BFILE;
        amount NUMBER;
        signature RAW(2000);
        cert_guid RAW (16) := HEXTORAW('9CCC45ABA31D5DC2E0532A26C40A860F');
BEGIN
        SELECT ORABCTAB_INST_ID$, ORABCTAB_CHAIN_ID$, ORABCTAB_SEQ_NUM$ INTO inst_id,
chain_id, sequence_no
                FROM bank_ledger WHERE bank='my_bank';
        file := bfilename('MY_DIR1', 'u1r1_sign.dat');
        DBMS_LOB.FILEOPEN(file);
        dbms_lob.READ(file, amount, 1, signature);
        dbms_lob.FILECLOSE(file);
        DBMS_BLOCKCHAIN_TABLE.SIGN_ROW('EXAMPLES','BANK_LEDGER', inst_id, chain_id,
sequence_no, NULL, signature, cert_guid,
DBMS_BLOCKCHAIN_TABLE.SIGN_ALGO_RSA_SHA2_512);
END;
/
PL/SQL procedure successfully completed.
SQL> SELECT bank, UTL_RAW.LENGTH(ORABCTAB_SIGNATURE$) sign_len,
ORABCTAB_SIGNATURE_ALG$,
  2    UTL_RAW.LENGTH(ORABCTAB_SIGNATURE_CERT$) sign_cert_guid_len
  3    FROM examples.bank_ledger ORDER BY bank;
BANK               SIGN_LEN ORABCTAB_SIGNATURE_ALG$ SIGN_CERT_GUID_LEN
---------------- ---------- ---------------------- ------------------
my_bank                 512                      1                 16
bank2                   256                      3                 16
```

関連項目:

DBMS_BLOCKCHAIN_TABLE.SIGN_ROWプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

親トピック: ブロックチェーン表の管理

20.18.8 ブロックチェーン表のデータの検証

PL/SQLプロシージャDBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSでは、ブロックチェーン表の行が挿入後に変更されていないことを確認します。改ざん耐性があることは、ブロックチェーン表の主要な要件です。

このプロシージャを実行するには、ブロックチェーン表に対するSELECT権限が必要です。

- DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSプロシージャを使用して、ブロックチェーン表内のハッシュ列の整合性を検証します。行に署名が含まれている場合は、その署名を検証できます。

  ブロックチェーン表内のすべての行を検証することも、検証の必要がある行をフィルタする基準を指定することもできます。行のフィルタリングには、インスタンスID、チェーンIDまたは行作成時刻を使用できます。

例20-40 特定のインスタンス内のブロックチェーン表の行の検証

次のPL/SQLブロックは、1から4までのインスタンスIDを持つブロックチェーン表bank_ledger内の行が、作成後に改ざんされ

ていないことを検証します。

DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSプロシージャのverify_signatureパラメータが省略されているため、デフォルト値のTRUEが使用されます。行の内容と行の署名(ある場合)が検証されます。verify_signatureパラメータをFALSEに設定すると、行の内容は検証されますが、行署名は検証されません。署名の検証は、このプロセスに費やす追加の時間とリソースを節約するために省略することもできます。

```
DECLARE
        verify_rows NUMBER;
        instance_id NUMBER;
BEGIN
        FOR instance_id IN 1 .. 4 LOOP
             DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS('EXAMPLES','BANK_LEDGER', NULL, NULL,
instance_id, NULL, verify_rows);
             DBMS_OUTPUT.PUT_LINE('Number of rows verified in instance Id '||
instance_id || ' = '|| verify_rows);
        END LOOP;
END;
/
Number of rows verified in instance Id 1 = 3
Number of rows verified in instance Id 2 = 12
Number of rows verified in instance Id 3 = 8
Number of rows verified in instance Id 4 = 10
PL/SQL procedure successfully completed.
```

関連項目:

DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

親トピック: ブロックチェーン表の管理

20.18.9 ブロックチェーン表の整合性の検証

ブロックチェーン表のデータが損なわれていないことを継続的に検証することで、ブロックチェーン表の整合性を維持します。

ブロックチェーン表データの整合性を検証するには:

1. DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSプロシージャを使用して、ブロックチェーン表のすべてのチェーン間のリンクを検証します。

   ユーザー署名を含む行の場合は、行署名も検証されます。

2. DBMS_BLOCKCHAIN_TABLE.GET_SIGNED_BLOCKCHAIN_DIGESTファンクションを使用して、ブロックチェーン表の署名および署名ダイジェストを生成します。

   この署名と署名ダイジェストは、T1の時点で生成されたものとします。生成日時など、生成されたこれらの詳細をリポジトリに格納します。リポジトリは、ブロックチェーン表を格納するデータベースの外部にある必要があります。これは、別のリレーショナル・データベースにすることもできます。

3. 別の時点で、DBMS_BLOCKCHAIN_TABLE.GET_SIGNED_BLOCKCHAIN_DIGESTファンクションを使用して、ブロックチェーン表の署名および署名ダイジェストを生成します。

   この署名と署名ダイジェストは、T2の時点で生成されたものとします。生成された詳細を生成日時とともにリポジトリに格納します。

4. DBMS_BLOCKCHAIN_TABLE.VERIFY_TABLE_BLOCKCHAINプロシージャを実行して、T1とT2の間に作成さ

れた行の整合性を検証します。

このプロシージャへの入力は、T1およびT2の時点で生成された署名ダイジェストです。行の整合性は、署名ダイジェストの一部である時間情報を使用して検証されます。

5. ステップ2から4のプロセスを異なる期間で繰り返し、異なる期間に挿入された行の整合性を確認します。
たとえば、T3とT4の時点の署名ダイジェストを計算し、T3とT4の時点の間の期間に作成された行の整合性を検証します。

ブロックチェーン表データの整合性を定期的に検証することをお薦めします。異なる期間で継続的に比較および検証するこの手法により、ブロックチェーン表の行が損なわれていないことが保証されます。

- ブロックチェーン表の署名ダイジェストの生成
署名ダイジェストは、メタデータと、ブロックチェーン表の各チェーンの最後の行に関するデータで構成されます。ブロックチェーン表データの整合性を検証するときに使用できます。
- 指定した期間内に作成されたブロックチェーン表の行の検証
指定した期間内に作成された行を検証すると、その期間中のブロックチェーン表の整合性を検証できます。

親トピック: ブロックチェーン表の管理

20.18.9.1 ブロックチェーン表の署名ダイジェストの生成

署名ダイジェストは、ブロックチェーン表の各チェーンの最後の行に関するメタデータとデータで構成されます。ブロックチェーン表データの整合性を検証するときに使用できます。

データベースは、署名ダイジェストの内容に基づいて署名を計算します。署名では、ブロックチェーン表の所有者の秘密キーおよび証明書が使用されます。サードパーティ・ツールを使用して、データベースによって生成された署名を検証できます。様々なタイミングで生成された署名および署名ダイジェストをリポジトリに格納していることを確認します。

ブロックチェーン表データの整合性を維持することの重要な側面は、すべての行が変更されないようにすることです。署名ダイジェストを計算すると、特定の時点におけるすべてのチェーンの最後の行に関するメタデータとデータのスナップショットが提供されます。この情報はリポジトリに格納する必要があります。様々なタイミングで生成された署名ダイジェストは、`DBMS_BLOCKCHAIN_TABLE.VERIFY_TABLE_BLOCKCHAIN`プロシージャへの入力を構成します。この手順を使用して、指定した2つの時点の間に作成された行の整合性を検証します。

前提条件

ブロックチェーン表の所有者の証明書は、`DBMS_USER_CERTS.ADD_CERTIFICATE`プロシージャを使用してデータベースに追加する必要があります。ブロックチェーン表の所有者のPKI秘密キーおよび証明書はウォレットに格納する必要があります。非CDBの場合、ウォレットは`WALLET_ROOT/bctable/`ディレクトリに配置する必要があります。PDBの場合、ウォレットは`WALLET_ROOT/pdb_guid/bctable/`ディレクトリに格納する必要があります。`pdb_guid`は、ブロックチェーン表を含むPDBのGUIDです。`WALLET_ROOT`には、各PDBのサブディレクトリを含むディレクトリ・ツリーのルートへのパスを指定します。ルートの下では、ディレクトリ構造がPDBに関連付けられている様々なウォレットの格納に使用されます。

ブロックチェーン表の署名ダイジェストおよび署名を生成するには:

- `DBMS_BLOCKCHAIN_TABLE.GET_SIGNED_BLOCKCHAIN_DIGEST`ファンクションを使用します。

  このファンクションは、最初にブロックチェーン表の所有者の証明書が有効かどうかを確認します。次に、データ型`BLOB`の署名ダイジェストと、署名ダイジェストのPL/SQL配列バージョンを計算します。署名ダイジェストには、ブロックチェーン表の各チェーンの最後の行のメタデータとデータが含まれます。PL/SQL配列は、署名ダイジェストの各チェーンの最後の行を識別します。このファンクションは、署名ダイジェストに基づく署名を返します。

ノート:

署名ダイジェストには、プラガブル・データベース(PDB)に固有の表情報が含まれます。したがって、この署名ダイジェストは、それが作成された PDB、およびダイジェストの作成に使用された表に対してのみ使用できます。

例20-41 ブロックチェーン表の署名ダイジェストと署名の生成

この例では、署名ダイジェストを計算し、ブロックチェーンEXAMPLES.BANK_LEDGERの署名を生成します。署名ダイジェストはバイナリ形式で、各チェーンの最後の行のメタデータとデータで構成されます。これは、signed_bytesに格納されています。署名ダイジェストのPL/SQL配列バージョンは、出力パラメータsigned_row_arrayに格納されます。署名の生成に使用される証明書のGUIDは、certificate_guidに格納されます。使用されるアルゴリズムは、DBMS_BLOCKCHAIN_TABLE.SIGN_ALGO_RSA_SHA2_512です。

```
DECLARE
   signed_bytes             BLOB:=EMPTY_BLOB();
   signed_row_array       SYS.ORABCTAB_ROW_ARRAY_T;
   certificate_guid       RAW(2000);
   signature              RAW(2000);
BEGIN
   signature := DBMS_BLOCKCHAIN_TABLE.GET_SIGNED_BLOCKCHAIN_DIGEST('EXAMPLES',
                  'BANK_LEDGER', signed_bytes, signed_row_array,
                  certificate_guid, dbms_blockchain_table.SIGN_ALGO_RSA_SHA2_512);
   DBMS_OUTPUT.PUT_LINE('Certificate GUID = ' || certificate_guid);
   DBMS_OUTPUT.PUT_LINE('Signature length = ' || UTL_RAW.LENGTH(signature));
   DBMS_OUTPUT.PUT_LINE('Number of chains = ' || signed_row_array.count);
   DBMS_OUTPUT.PUT_LINE('Signature content buffer length = ' ||
DBMS_LOB.GETLENGTH(signed_bytes));
END;
/
Certificate GUID = AF27H7FE3EEA473GE0783FE56A0AFCEB
Signature length = 256
Number of chains = 10
Signature content buffer length = 1248
PL/SQL procedure successfully completed.
```

関連トピック

- ブロックチェーン表の署名ダイジェストの形式

親トピック: ブロックチェーン表の整合性の検証

20.18.9.2 指定した期間内に作成されたブロックチェーン表の行の検証

指定した期間内に作成された行を検証すると、その期間中のブロックチェーン表の整合性を検証できます。

DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSプロシージャは、ブロックチェーン表のすべての行の整合性を検証します。表全体を毎回検証するかわりに、最後の検証以降に作成された行のみを検証できます。たとえば、2日前にDBMS_BLOCKCHAIN_TABLE.VERIFY_ROWSを実行した場合、その検証後に追加された行のみを検証できます。

指定した期間内に作成されたブロックチェーン表の行を検証するには:

- DBMS_BLOCKCHAIN_TABLE.VERIFY_TABLE_BLOCKCHAINプロシージャを使用します。

  このプロシージャへの入力は、DBMS_BLOCKCHAIN_TABLE.GET_SIGNED_BLOCKCHAIN_DIGESTファンクションを使用して異なる時点に生成された2つの署名ダイジェストです。検証の期間は、署名ダイジェストの情報を使用して決定されます。最初の署名ダイジェストからの最小行作成時間および2番目の署名ダイジェストからの最大行作成時間が考慮されます。出力は、検証された行数です。

ノート:

両方の署名ダイジェストを、現在のプラガブル・データベース(PDB)内および同じブロックチェーン表に対して生成する必要があります。たとえば、PDB my_pdb1 にブロックチェーン表の署名ダイジェストを作成し、Oracle Data Pump を使用してブロックチェーン表をエクスポートし、Oracle Data Pump を使用してブロックチェーン表を PDB my_pdb2 にインポートするとします。PDB my_pdb1 で作成された署名ダイジェストは、PDB my_pdb2 では使用できません。PDB my_pdb2 に新しい署名ダイジェストを作成する必要があります。

例20-42 指定した期間に作成されたブロックチェーン表の行の検証

この例では、指定した期間にEXAMPLES.BANK_LEDGERブロックチェーン表に作成された行を検証します。2回の異なる時点のブロックチェーン表の署名ダイジェストは、signed_bytes1およびsigned_bytes2に格納されます。signed_bytes1の値は、署名ダイジェストおよび署名のリポジトリであるsigned_digest_repo表から読み取られます。signed_bytes2の値は、DBMS_BLOCKCHAIN_TABLE.GET_SIGNED_BLOCKCHAIN_DIGESTファンクションを使用して計算されます。signed_bytes1の最小行作成時間とsigned_bytes2の最大行作成時間の間に作成された行が検証の対象となります。

```
DECLARE
    signature              RAW(2000);
    sign_row_array         SYS.ORABCTAB_ROW_ARRAY_T;
    signed_bytes1          BLOB;
    certificate_guid       RAW(2000);
    signed_bytes2          BLOB;
    rows_verified          NUMBER;
BEGIN
    SELECT signed_digest INTO signed_bytes1 FROM signed_digest_repo WHERE
time>=SYSDATE-1;
    signature := DBMS_BLOCKCHAIN_TABLE.GET_SIGNED_BLOCKCHAIN_DIGEST('EXAMPLES',
                     'BANK_LEDGER', signed_bytes2, sign_row_array,
certificate_guid);
    DBMS_BLOCKCHAIN_TABLE.VERIFY_TABLE_BLOCKCHAIN(signed_bytes2,signed_bytes1,
rows_verified);
    dbms_output.put_line('Rows verified = ' || rows_verified);
END;
/
Rows verified = 10
PL/SQL procedure successfully completed.
```

親トピック: ブロックチェーン表の整合性の検証

20.18.10 ブロックチェーン表からの行の削除

保存期間外の行のみをブロックチェーン表から削除できます。

SYSユーザーまたはスキーマの所有者は、ブロックチェーン表から行を削除できます。

PL/SQLプロシージャDBMS_BLOCKCHAIN_TABLE.DELETE_EXPIRED_ROWSは、保存期間を超えている行をブロックチェーン表から削除します。保存期間内にないすべての行、または指定した日付より前に作成された行を削除できます。

例20-43 ブロックチェーン表からの適格な行の削除

次の例では、SYSとして接続した場合、保存ウィンドウ外にあるブロックチェーン表bank_ledger内のすべての行を削除します。削除された行数は、出力パラメータnum_rowsに格納されます。

```
DECLARE
        num_rows NUMBER;
BEGIN
```

```
        DBMS_BLOCKCHAIN_TABLE.DELETE_EXPIRED_ROWS('EXAMPLES','BANK_LEDGER', NULL,
num_rows);
        DBMS_OUTPUT.PUT_LINE('Number_of_rows_deleted=' || num_rows);
END;
/
Number_of_rows_deleted=2

PL/SQL procedure successfully completed.
```

例20-44 作成時間に基づいた適格な行の削除

次の例では、SYSとして接続した場合、10-OCT-2019.より前に作成された保存期間外の行を削除します。削除された行数は、出力パラメータnum_rowsに格納されます。

```
DECLARE
        num_rows NUMBER;
BEGIN
        DBMS_BLOCKCHAIN_TABLE.DELETE_EXPIRED_ROWS('EXAMPLES','BANK_LEDGER',
TO_DATE('10-OCT-19','DD-MON-YY'), num_rows);
        DBMS_OUTPUT.PUT_LINE('Number_of_rows_deleted=' || num_rows);
END;
        Number_of_rows_deleted=5

        PL/SQL procedure successfully completed.
```

親トピック: ブロックチェーン表の管理

20.18.11 ブロックチェーン表の削除

ブロックチェーン表は、行が含まれていない場合、またはその保存期間により定義された期間に変更されていない場合に削除できます。

ブロックチェーン表が自分のスキーマに含まれているか、DROP ANY TABLEシステム権限を持っている必要があります。ブロックチェーン表を削除する場合は、PURGEオプションを含めることをお薦めします。

- DROP TABLE文を使用して、ブロックチェーン表を削除します。ブロックチェーン表を削除すると、データ・ディクショナリから定義が削除され、そのすべての行が削除され、ブロックチェーン表に定義されている索引およびトリガーも削除されます。

  次のコマンドは、examplesスキーマ内のmy_blockchain_tableという名前のブロックチェーン表を削除します。

  ```
  DROP TABLE examples.my_blockchain_table PURGE;
  ```

親トピック: ブロックチェーン表の管理

20.18.12 行のハッシュを計算するための行内容のデータ形式の判別

行のハッシュ値を計算するには、行内容のデータ形式を判別するために
DBMS_BLOCKCHAIN_TABLE.GET_BYTES_FOR_ROW_HASHプロシージャを使用します。

データベースによって計算された行ハッシュ値を個別に検証する場合は、まず、その行内容のデータ形式(バイト)を判別します。次に、チェーンの前の行の行内容とハッシュ値の組合せに対して、SHA2-512ハッシュ・アルゴリズムを使用します。

ブロックチェーン表の行ハッシュを無効にせずにデータベース文字セットおよび各国語文字セットを変更できるようにするために、ブロックチェーン表の各行ハッシュは、文字データ型または文字LOBデータ型を持つ各列の正規化された値に対して計算されます。具体的には、ハッシュされる前に、VARCHAR2列またはCHAR列の値がAL32UTF8表現に変換されます。ハッシュされる前に、NVARCHAR2列、NCHAR列、CLOB列またはNCLOB列の値がAL16UTF16表現に変換されます。すでにAL32UTF8または

AL16UTF16になっている列値は変換されませんが、不正な文字コードがないかチェックされる可能性があります。

CHARおよびNCHARの値は、末尾の空白を削除することでさらに正規化されます。すべての空白で構成されるCHARまたはNCHAR値は、nullにならないように単一の空白に正規化されます。

DBMS_BLOCKCHAIN_TABLE.GET_BYTES_FOR_ROW_HASHプロシージャを使用して、行のハッシュを計算するときに、行内容のデータ形式を判別します。このプロシージャは、指定された行とチェーン内の前の行のハッシュ値(データ形式で)のバイトを列位置の順序で返します。

行を指定するには、その行のインスタンスID、チェーンIDおよび順序番号を指定する必要があります。

例20-45 ストアド行のハッシュ値の検証

この例では、特定のデータベース・インスタンスに最後に追加された行の行内容とBANK_LEDGER表のチェーンのデータ形式を取得します。DBMS_CRYPTO.HASH関数は、ハッシュ値の計算のために使用しています。ハッシュ値を個別に検証するために、計算されたハッシュ値とデータベースから取得した値を比較しています。

DBMS_CRYPTOパッケージの実行に必要な権限を持っている必要があります。データ形式の値は1である必要があります。

```
set serveroutput on;
DECLARE
        row_data BLOB;
        row_id ROWID;
        row_hash RAW(64);
        computed_hash RAW(64);
        buffer RAW(4000);
        inst_id BINARY_INTEGER;
        chain_id BINARY_INTEGER;
        sequence_no BINARY_INTEGER;
BEGIN
        -- Get the row details and hash value of the most recently inserted row with
the specified instance ID and chain ID
        SELECT MAX(ORABCTAB_SEQ_NUM$) INTO sequence_no
               FROM EXAMPLES.BANK_LEDGER
               WHERE ORABCTAB_INST_ID$=1 AND ORABCTAB_CHAIN_ID$=4;
        SELECT ORABCTAB_INST_ID$, ORABCTAB_CHAIN_ID$, ORABCTAB_SEQ_NUM$,
ORABCTAB_HASH$ INTO inst_id, chain_id, sequence_no, row_hash
               FROM EXAMPLES.BANK_LEDGER
               WHERE ORABCTAB_INST_ID$=1 AND ORABCTAB_CHAIN_ID$=4 AND
ORABCTAB_SEQ_NUM$ = sequence_no;
        -- Compute the row hash externally from row column bytes
        DBMS_BLOCKCHAIN_TABLE.GET_BYTES_FOR_ROW_HASH('EXAMPLES', 'BANK_LEDGER',
inst_id, chain_id, sequence_no, 1, row_data);
        computed_hash := DBMS_CRYPTO.HASH(row_data, DBMS_CRYPTO.HASH_SH512);
        -- Verify that the row's hash and externally computed hash are same
        if UTL_RAW.COMPARE(row_hash, computed_hash) = 0 THEN
                DBMS_OUTPUT.PUT_LINE('Hash verification successful');
        else
                DBMS_OUTPUT.PUT_LINE('Hash verification failed');
END IF;
END;
Hash verification successful
PL/SQL procedure successfully completed.
```

関連トピック

- ブロックチェーン表参照

親トピック: ブロックチェーン表の管理

20.18.13 行の署名を計算するためのデータ形式の判別

行の署名の計算に使用される行内容のデータ形式を判別できます。行署名は、その行のハッシュ値に基づいて計算されます。

DBMS_BLOCKCHAIN_TABLE.GET_BYTES_FOR_ROW_SIGNATUREプロシージャを使用して、行の署名を計算するための行内容のデータ形式を判別します。このプロシージャは、行内容の形式で指定された行のバイトを返します。

例20-46 ブロックチェーン表内の行の署名の計算

この例では、examples.bank_ledger表のbank値が'my_bank'の行のバイトを計算します。これらのバイトは、変数row_dataに格納されます。

```
DECLARE
        row_data BLOB;
        buffer RAW(4000);
        inst_id BINARY_INTEGER;
        chain_id BINARY_INTEGER;
        sequence_no BINARY_INTEGER;
        row_len BINARY_INTEGER;
BEGIN
        SELECT ORABCTAB_INST_ID$, ORABCTAB_CHAIN_ID$, ORABCTAB_SEQ_NUM$ INTO inst_id,
chain_id, sequence_no
             FROM EXAMPLES.BANK_LEDGER where bank='my_bank';

DBMS_BLOCKCHAIN_TABLE.GET_BYTES_FOR_ROW_SIGNATURE('EXAMPLES','BANK_LEDGER',inst_id,
chain_id, sequence_no, 1, row_data);
        row_len := DBMS_LOB.GETLENGTH(row_data);
        DBMS_LOB.READ(row_data, row_len, 1, buffer);
END;
/
PL/SQL procedure successfully completed.
```

親トピック: ブロックチェーン表の管理

20.18.14 ブロックチェーン表のデータのバイト値の表示

ブロックチェーン表のデータのバイト値(行と列の両方)を取得できます。

次のいずれかのプロシージャを使用して、ブロックチェーン表または通常の表にあるデータのバイト値を表示します。

- DBMS_TABLE_DATA.GET_BYTES_FOR_COLUMNプロシージャは、単一列の列データをバイトで判別するために使用します。

  次の例では、examplesスキーマのbank_ledger表にある1つのbank列のバイト値を判別します。

  ```
  DECLARE
          row_id ROWID;
          col_data BLOB;
          buffer RAW(4000);
          data_len BINARY_INTEGER;
  begin
          SELECT rowid INTO row_id FROM bank_ledger WHERE bank='my_bank';
          DBMS_TABLE_DATA.GET_BYTES_FOR_COLUMN('EXAMPLES', 'BANK_LEDGER',
  row_id,'BANK', col_data);
          data_len := dbms_lob.getlength(col_data);
          DBMS_LOB.READ(col_data, data_len, 1, buffer);
          DBMS_OUTPUT.PUT_LINE('len=' || data_len || ', data=' ||
  RAWTOHEX(buffer));
  end;
  ```

- 列のセットの列データをバイト単位で判別するには、DBMS_TABLE_DATA.GET_BYTES_FOR_COLUMNSプロシー

ジャを使用します。VARRAYを使用して、プロシージャに列のセットを指定します。

- DBMS_TABLE_DATA.GET_BYTES_FOR_ROWプロシージャを使用して、単一行の行データをバイト単位で判別します。

  次の例では、表bank_ledgerの特定の行の行データをバイト単位で表示します。

```
DECLARE
        row_data blob;
        data_len binary_integer;
        row_id rowid;
        inst_id binary_integer;
        chain_id binary_integer;
        sequence_no binary_integer;
BEGIN
        SELECT rowid INTO row_id FROM bank_ledger WHERE bank='my_bank';
        DBMS_TABLE_DATA.GET_BYTES_FOR_ROW('EXAMPLES','BANK_LEDGER',row_id,
row_data);
        data_len := DBMS_LOB.GETLENGTH(row_data);
        DBMS_OUTPUT.PUT_LINE('Row data-length=' || data_len);
END;
/
Row data-length=908.
PL/SQL procedure successfully completed.
```

関連トピック

- ブロックチェーン表参照

親トピック: ブロックチェーン表の管理

20.18.15 ブロックチェーン表のデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューは、ブロックチェーン表に関する情報を提供します。

ブロックチェーン表の詳細は、DBA_BLOCKCHAIN_TABLES、ALL_BLOCKCHAIN_TABLESまたはUSER_BLOCKCHAIN_TABLESのいずれかのビューを問い合せてください。情報には、行の保存期間、表の保存期間、行の連鎖に使用したハッシュ処理アルゴリズムが含まれます。DBAビューはデータベース内のすべてのブロックチェーン表を表し、ALLビューはユーザーがアクセスできるすべてのブロックチェーン表を表し、USERビューはユーザーが所有するブロックチェーン表に制限されます。

例20-47 ブロックチェーン表の情報の表示

次のコマンドでは、examplesスキーマ内のブロックチェーン表bank_ledgerの詳細が表示されます。

```
SELECT row_retention "Row Retention Period", row_retention_locked "Row Retention
Lock", table_inactivity_retention "Table Retention Period", hash_algorithm "Hash
Algorithm"
FROM dba_blockchain_tables WHERE table_name='BANK_LEDGER';
Row Retention Period Row Retention Lock   Table  Retention Period Hash Algorithm
-------------------- ------------------  ------------------------ --------------
                  16 YES                                       31 SHA2_512
```

親トピック: ブロックチェーン表の管理

20.19 表のデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューのセットを問い合せて、表についてのデータを取得できます。

ビュー	説明
DBA_TABLES ALL_TABLES USER_TABLES	DBA ビューには、データベース内のすべてのリレーショナル表が表示されます。ALL ビューには、ユーザーがアクセス可能なすべての表が表示されます。USER ビューは、ユーザーが所有する表のみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_TAB_COLUMNS ALL_TAB_COLUMNS USER_TAB_COLUMNS	これらのビューには、データベース内の表の列、ビューおよびクラスタが表示されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_ALL_TABLES ALL_ALL_TABLES USER_ALL_TABLES	これらのビューには、データベース内のすべてのリレーショナル表およびオブジェクト表が表示されます。オブジェクト表については、このマニュアルでは詳しく説明していません。
DBA_TAB_COMMENTS ALL_TAB_COMMENTS USER_TAB_COMMENTS	これらのビューには、表およびビューのコメントが表示されます。コメントは、COMMENT 文を使用して入力します。
DBA_COL_COMMENTS ALL_COL_COMMENTS USER_COL_COMMENTS	これらのビューには、表およびビューの列のコメントが表示されます。コメントは、COMMENT 文を使用して入力します。
DBA_EXTERNAL_TABLES ALL_EXTERNAL_TABLES USER_EXTERNAL_TABLES	これらのビューには、データベースで定義されている外部表の特定の属性がリストされます。
DBA_EXTERNAL_LOCATIONS ALL_EXTERNAL_LOCATIONS USER_EXTERNAL_LOCATIONS	これらのビューには、外部表のデータソースがリストされます。

ビュー	説明
DBA_XTERNAL_PART_TABLES ALL_XTERNAL_PART_TABLES USER_XTERNAL_PART_TABLES	これらのビューには、データベースで定義されているパーティション化された外部表の特定の属性がリストされます。
DBA_XTERNAL_TAB_PARTITIONS ALL_XTERNAL_TAB_PARTITIONS USER_XTERNAL_TAB_PARTITIONS	これらのビューには、データベースで定義されているパーティション化された外部表のパーティション・レベルの情報がリストされます。
DBA_XTERNAL_TAB_SUBPARTITIONS ALL_XTERNAL_TAB_SUBPARTITIONS USER_XTERNAL_TAB_SUBPARTITIONS	これらのビューには、データベースで定義されているパーティション化された外部表のサブパーティション・レベルの情報がリストされます。
DBA_XTERNAL_LOC_PARTITIONS ALL_XTERNAL_LOC_PARTITIONS USER_XTERNAL_LOC_PARTITIONS	これらのビューには、外部表のパーティションのデータソースがリストされます。
DBA_XTERNAL_LOC_SUBPARTITIONS ALL_XTERNAL_LOC_SUBPARTITIONS USER_XTERNAL_LOC_SUBPARTITIONS	これらのビューには、外部表のサブパーティションのデータソースがリストされます。

ビュー	説明
DBA_TAB_HISTOGRAMS ALL_TAB_HISTOGRAMS USER_TAB_HISTOGRAMS	これらのビューには、表およびビューに関するヒストグラムが表示されます。
DBA_TAB_STATISTICS ALL_TAB_STATISTICS USER_TAB_STATISTICS	これらのビューには、表のオプティマイザ統計が格納されます。
DBA_TAB_COL_STATISTICS ALL_TAB_COL_STATISTICS USER_TAB_COL_STATISTICS	これらのビューは、関連する TAB_COLUMNS ビューから抽出された列の統計およびヒストグラム情報を提供します。
DBA_TAB_MODIFICATIONS ALL_TAB_MODIFICATIONS USER_TAB_MODIFICATIONS	これらのビューには、表統計が最後に収集された時点以降変更された表が表示されます。これらのビューは即時には移入されず、ある程度の時間(通常は 3 時間)が経過した後に移入されます。
DBA_ENCRYPTED_COLUMNS ALL_ENCRYPTED_COLUMNS USER_ENCRYPTED_COLUMNS	これらのビューには、暗号化された表の列がリストされ、各列に使用している暗号化アルゴリズムがリストされます。
DBA_UNUSED_COL_TABS ALL_UNUSED_COL_TABS USER_UNUSED_COL_TABS	これらのビューには、ALTER TABLE ... SET UNUSED 文によって未使用のマークが付けられた列を持つ表がリストされます。
DBA_PARTIAL_DROP_TABS ALL_PARTIAL_DROP_TABS USER_PARTIAL_DROP_TABS	これらのビューには、DROP COLUMN 操作が一部完了している表がリストされます。これらの操作は、ユーザーによる中断やシステム障害が原因で不完全になることがあります。

例: 列情報の表示

_COLUMNS接尾辞で終わるビューのいずれかを使用すると、名前、データ型、長さ、精度、位取り、デフォルト・データ値などの列情報を表示できます。たとえば、次の問合せは、emp表とdept表のデフォルトの列値をすべてリストします。

```
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, LAST_ANALYZED
    FROM DBA_TAB_COLUMNS
    WHERE OWNER = 'HR'
    ORDER BY TABLE_NAME;
```

問合せの出力は次のとおりです。

```
TABLE_NAME           COLUMN_NAME          DATA_TYPE   DATA_LENGTH LAST_ANALYZED
-------------------- -------------------- ---------- ------------ -------------
COUNTRIES            COUNTRY_ID           CHAR                2 05-FEB-03
COUNTRIES            COUNTRY_NAME         VARCHAR2           40 05-FEB-03
COUNTRIES            REGION_ID            NUMBER             22 05-FEB-03
DEPARTMENTS          DEPARTMENT_ID        NUMBER             22 05-FEB-03
DEPARTMENTS          DEPARTMENT_NAME      VARCHAR2           30 05-FEB-03
DEPARTMENTS          MANAGER_ID           NUMBER             22 05-FEB-03
DEPARTMENTS          LOCATION_ID          NUMBER             22 05-FEB-03
EMPLOYEES            EMPLOYEE_ID          NUMBER             22 05-FEB-03
EMPLOYEES            FIRST_NAME           VARCHAR2           20 05-FEB-03
EMPLOYEES            LAST_NAME            VARCHAR2           25 05-FEB-03
EMPLOYEES            EMAIL                VARCHAR2           25 05-FEB-03
.
.
.
LOCATIONS            COUNTRY_ID           CHAR                2 05-FEB-03
REGIONS              REGION_ID            NUMBER             22 05-FEB-03
REGIONS              REGION_NAME          VARCHAR2           25 05-FEB-03
51 rows selected.
```

関連項目:

- オブジェクト表の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください
- ヒストグラムおよび表の統計生成の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- 「表、索引およびクラスタの分析について」

親トピック: 表の管理

21 索引の管理

索引は、データ・アクセスを高速化できます。索引を作成、変更、監視および削除できます。

- 索引について
  索引は表およびクラスタと関連付けられているオプション構造で、これにより表に対するSQL問合せをより迅速に実行できます。
- 索引を管理するためのガイドライン
  ガイドラインに従って索引を管理できます。
- 索引の作成
  いくつかの異なるタイプの索引を作成できます。明示的な索引の作成、および制約に関連付けられた索引の作成ができます。
- 索引の変更
  索引の記憶域の特性を変更する、再作成する、使用不能にする、または表示/非表示にするなどのタスクを完了することによって、索引を変更することができます。
- 索引の使用領域の監視
  索引のキー値が頻繁に挿入、更新および削除されると、時間が経つにつれて領域の効率性が失われることがあります。
- 索引の削除
  索引は、DROP INDEX文により削除できます。
- 自動索引の管理
  自動索引作成機能を使用することで、Oracleデータベースで自動索引を構成して使用し、データベースのパフォーマンスを向上させることができます。
- 索引のデータ・ディクショナリ・ビュー
  索引に関する情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

親トピック: スキーマ・オブジェクト

21.1 索引について

索引は表およびクラスタと関連付けられているオプション構造で、これにより表に対するSQL問合せをより迅速に実行できます。

このマニュアルで、索引の使用により、索引がない場合よりも高速に情報を検索できるように、Oracle Databaseの索引は、表データへのより高速なアクセス・パスを提供します。索引は問合せをリライトすることなく使用できます。結果は同じですが、より高速に得られます。

Oracle Databaseは、補完的なパフォーマンス機能を持つ複数の索引付け方法を提供します。それらは次のとおりです。

- Bツリー索引: デフォルトの設定で、最も一般的です。
- Bツリー・クラスタ索引: 特にクラスタ用に定義します。
- ハッシュ・クラスタ索引: 特にハッシュ・クラスタ用に定義します。
- グローバル索引とローカル索引: パーティション表とパーティション索引に関連します。
- 逆キー索引: Oracle Real Application Clustersアプリケーションに最も役立ちます。
- ビットマップ索引: サイズが小さいので、小さい値の集合を持つ列に効果的です。
- ファンクション索引: 事前計算された関数や式の値を含みます。

- ドメイン索引: アプリケーションまたはカートリッジに固有の索引です。

索引は、対応付けられた表内のデータから論理的にも物理的にも独立しています。索引は独立した構造体であり、記憶域を必要とします。索引は、実表、データベース・アプリケーションまたはその他の索引に影響を与えることなく、作成または削除できます。索引に対応する表に対して行の挿入、更新および削除が発生すると、データベースは索引を自動的にメンテナンスします。索引を削除しても、すべてのアプリケーションは引き続き動作可能です。ただし、それまで索引が付けられていたデータへのアクセスが遅くなります。

関連項目:

- 索引の概要については、『Oracle Database概要』を参照してください。
- スキーマ・オブジェクトの領域の管理

親トピック: 索引の管理

21.2 索引を管理するためのガイドライン

ガイドラインに従って索引を管理できます。

- 表データ挿入後の索引の作成
  データは、通常、SQL*Loaderまたはインポート・ユーティリティを使用して表に挿入またはロードされます。データの挿入またはロードの後に表の索引を作成すると効率がよくなります。データのロード前に1つ以上の索引を作成すると、データベースに行が挿入されるたびに索引を更新する必要があります。
- 正しい表および列への索引付け
  索引付けに適した表および列についてのガイドラインに従います。
- パフォーマンスのための索引列の順序
  CREATE INDEX文の列の順序は、問合せのパフォーマンスに影響を与える可能性があります。一般的には、最も頻繁に使用する列を最初に指定します。
- 表当たりの索引数の制限
  表は、任意の数の索引を持つことができます。ただし、索引の数が多いほど、表を変更するときに発生するオーバーヘッドが増加します。
- 不必要な索引の削除
  必要とされない索引は、削除することをお薦めします。
- 索引およびセグメント作成の遅延
  索引セグメントの作成は、対応付けられた表でセグメント作成の遅延が発生すると、遅延されます。これは、索引セグメントの作成には対応付けられた表の動作が反映されるためです。
- 索引サイズの見積りと記憶域パラメータの設定
  索引を作成する前にそのサイズを見積っておくと、ディスク領域の計画と管理がいっそう容易になります。
- 各索引の表領域の指定
  任意の表領域に索引を作成できます。索引は、その索引を付けた表と同じ表領域にも、異なる表領域にも作成できます。
- 索引作成のパラレル化
  表作成をパラレル化できるのと同様に、索引の作成もパラレル化できます。複数のプロセスが同時に動作して索引を作成するため、1つのサーバー・プロセスが順に索引を作成する場合よりも高速に索引を作成できます。
- 索引作成時のNOLOGGINGの使用

CREATE INDEX文でNOLOGGINGを指定すると、索引の作成時に最小限のREDOログ・レコードしか生成されません。

- 使用禁止または不可視索引の使用について
  使用禁止または不可視索引は、バルク・ロードのパフォーマンスを向上させる場合、索引を削除する前に削除によって発生する影響をテストする場合、またはオプティマイザによるその索引の使用を停止する場合に使用します。
- 同じ列セットに対する複数の索引の作成について
  索引がいくらか異なる場合、同じ列セットに複数の索引を作成できます。たとえば、同じ列セットに対してBツリー索引とビットマップ索引を作成できます。
- 索引の結合と再作成に関するコストと利点の検討
  不適切な索引サイズの設定やサイズの増加によって、索引の断片化が生じることがあります。断片化を解消または低減するには、索引を再作成するか、索引を結合します。ただし、どちらの作業を行う場合も、事前に各選択肢のコストと利点を分析し、状況に最も有効な方法を選択してください。
- 制約を使用禁止または削除する前のコストの検討
  一意キーと主キーには対応する索引があるため、UNIQUE制約やPRIMARY KEY制約を使用禁止または削除するかどうかを検討するときには、索引の削除と作成にかかわるコストを考慮に入れてください。
- 索引数を減らすためのインメモリー列ストアの使用の検討
  インメモリー列ストアは、システム・グローバル領域(SGA)のオプション部分で、高速スキャン用に最適化された表、表パーティション、その他のデータベース・オブジェクトのコピーが格納されます。インメモリー列ストアでは、表データがSGAに行ではなく列ごとに格納されます。

関連項目:

- Oracleが提供する各種索引付け方法の説明など、索引と索引付けの概念に関する情報は、『Oracle Database概要』を参照してください。
- ビットマップ索引の詳細は、『Oracle Database SQLチューニング・ガイド』および『Oracle Databaseデータ・ウェアハウス・ガイド』
- ドメイン固有のオペレータと索引付け方法の定義方法およびOracle Databaseサーバーへの統合方法については、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

親トピック: 索引の管理

21.2.1 表データ挿入後の索引の作成

データは、通常、SQL*Loaderまたはインポート・ユーティリティを使用して表に挿入またはロードされます。データの挿入またはロードの後に表の索引を作成すると効率がよくなります。データのロード前に1つ以上の索引を作成すると、データベースに行が挿入されるたびに索引を更新する必要があります。

すでにデータが格納されている表に索引を作成するには、ソート領域が必要です。索引の作成ユーザーに割り当てられているメモリーから確保されるソート領域もあります。各ユーザーのソート領域の大きさは、初期化パラメータSORT_AREA_SIZEによって決まります。また、データベースでは、索引作成のときにのみユーザーの一時表領域に割り当てられる一時セグメントとの間でソート情報のスワップが行われます。

特定の条件下では、SQL*Loaderのダイレクト・パス・ロードを使用してデータを表にロードし、データがロードされたときに索引が作成されるようにすることも可能です。

関連項目:

SQL*Loaderを使用したダイレクト・パス・ロードの詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: 索引を管理するためのガイドライン

21.2.2 正しい表および列への索引付け

索引付けに適した表および列についてのガイドラインに従います。

索引を作成するかどうか判断する際は、次のガイドラインを参考にしてください。

- 大きな表で頻繁に検索される行の割合が15%未満の場合は索引を作成してください。割合は、表スキャンの相対速度、および索引キーに対する行データの分散度によって大きく異なります。表スキャンが高速であるほど割合は低くなり、クラスタ化されている行データが多いほど割合は高くなります。
- 複数の表を結合するパフォーマンスを改善するには、結合に使用する列に索引を付けます。

ノート:

主キーおよび一意キーには自動的に索引が作成されますが、外部キーには必要に応じて索引を作成できます。

- 問合せに時間がかかりすぎる場合は、表のサイズを確認してください。大幅に変更されている場合、既存の索引(存在する場合)を確認する必要がある場合があります。

索引付けに適した列

列のタイプによっては、索引を付ける方が望ましいものがあります。次のような特性を1つ以上持つ列には、索引の作成を検討してください。

- 一意の値が比較的多い。
- 値の範囲が広い(通常の索引が適している)。
- 値の範囲が狭い(ビットマップ索引が適している)。
- 列に多くのNULLが含まれているが、通常の問合せでは必ず値を持つ列を選択する。この場合は次の句を使用します。

  ```
  WHERE COL_X > -9.99 * power(10,125)
  ```

  この句は、次の句よりも適しています。

  ```
  WHERE COL_X IS NOT NULL
  ```

  これは、最初の句ではCOL_Xの索引を使用しているためです(COL_Xは数値列とします)。

索引付けに適さない列

次のような特性を持つ列は、索引付けには適していません。

- 列にNULLが多く含まれていて、NULL以外の値を検索することがない。

LONG列およびLONG RAW列には索引を作成できません。

仮想列

仮想列には、一意索引または非一意索引を作成できます。仮想列に定義された表の索引は、表のファンクション索引と同じです。

関連項目:

「ファンクション索引の作成」

親トピック: 索引を管理するためのガイドライン

21.2.3 パフォーマンスのための索引列の順序付け

CREATE INDEX文の列の順序は、問合せのパフォーマンスに影響を与えます。一般的には、最も頻繁に使用する列を最初に指定します。

たとえば、col1、col2およびcol3の各列にアクセスする問合せを高速にするために、列にまたがる単一の索引を作成すると、col1のみにアクセスする問合せ、またはcol1とcol2にアクセスする問合せが速くなります。しかし、col2のみにアクセスする問合せ、col3のみにアクセスする問合せ、およびcol2とcol3にアクセスする問合せは速くなりません。

ノート:

先頭列のカーディナリティが非常に低い場合などには、データベースでは、このタイプの索引が使用されることがあります。索引スキップ・スキャンの詳細は、『Oracle Database 概要』を参照してください。

親トピック: 索引を管理するためのガイドライン

21.2.4 表当たりの索引数の制限

表は、多数の索引を持つことができます。ただし、索引の数が多いほど、表を変更するときに発生するオーバーヘッドが増加します。

特に、行を挿入したり削除したりするときは、その表の索引もすべて更新する必要があります。また、列を更新するときには、その列を含む索引もすべて更新する必要があります。

このように、表からデータを検索する速度とその表を更新する速度は二律背反的です。たとえば、表が主に読取り専用である場合、索引を増やすと有効ですが、表が頻繁に更新される場合は、索引を少なくすることをお薦めします。

親トピック: 索引を管理するためのガイドライン

21.2.5 不必要な索引の削除

必要とされない索引は、削除することをお薦めします。

次のような状況では、索引の削除を検討してください。

- 問合せを高速化しない。これには、たとえば表が非常に小さい場合や、表の行数は多いものの、索引エントリが非常に少ない場合などがあります。
- アプリケーションの問合せが索引を使用しない場合。
- 索引を再作成する前にいったん削除する必要がある場合。

関連項目:

「索引の使用状況の監視」

親トピック: 索引を管理するためのガイドライン

21.2.6 索引およびセグメント作成の遅延

索引セグメントの作成は、対応付けられた表でセグメント作成の遅延が発生すると、遅延されます。これは、索引セグメントの作成には対応付けられた表の動作が反映されるためです。

関連項目:

詳細は、「セグメント作成の遅延の理解」を参照してください

親トピック: 索引を管理するためのガイドライン

21.2.7 索引サイズの見積りと記憶域パラメータの設定

索引を作成する前にそのサイズを見積っておくと、ディスク領域の計画と管理がいっそう容易になります。

索引の見積りサイズの合計と、表、UNDO表領域およびREDOログ・ファイルの見積りを使用して、作成するデータベースを格納するために必要なディスク容量を決定できます。この見積りを利用して適切なハードウェアを購入できます。

個々の索引の見積りサイズを使用することで、索引が使用するディスク領域をより適切に管理できます。索引を作成するときに、適切な記憶域パラメータを設定し、その索引を使用するアプリケーションのI/Oパフォーマンスを改善できます。たとえば、索引を作成する前に最大サイズを予測したとします。次に索引の作成時に記憶域パラメータを設定すると、表データ・セグメントに対して割り当てられるエクステントが少なくなり、すべての索引データはディスク領域の比較的連続したセクションに格納されます。これにより、この索引に関するディスクのI/O操作にかかる時間が削減されます。

単一の索引エントリの最大サイズは、データベースのブロック・サイズによって異なります。

索引を主キー制約または一意キー制約を規定するために使用する場合、その索引のために作成する索引セグメントの記憶域パラメータは、次のどちらかの方法で設定できます。

- `ENABLE ... CREATE TABLE`文または`ALTER TABLE`文の`USING INDEX`句
- `ALTER INDEX`文の`STORAGE`句

関連項目:

- 索引のサイズに関する制限の詳細は、Oracle Databaseリファレンスを参照してください
- 拡張データ型列への索引の作成の詳細は、Oracle Database SQL言語リファレンスを参照してください

親トピック: 索引を管理するためのガイドライン

21.2.8 各索引の表領域の指定

索引はどの表領域にも作成できます。索引は、その索引を付けた表と同じ表領域にも、異なる表領域にも作成できます。

表とその索引に対して同じ表領域を使用すると、(表領域やファイルのバックアップなどの)データベースのメンテナンスやアプリケーションの可用性確保の面で便利です。これは、すべての関連するデータが常にまとまってオンラインになっているためです。

表とその索引に対して(異なるディスク上にある)異なる表領域を使用すると、表と索引を同じ表領域に格納するよりも、パフォーマンスが向上します。これは、ディスクの競合が解消されるためです。表とその索引に対して異なる表領域を使用し、一方の(データまたは索引のいずれかを含む)表領域がオフラインになっている場合には、その表を参照している文が動作する保証はありません。

親トピック: 索引を管理するためのガイドライン

21.2.9 索引作成のパラレル化

表作成をパラレル化できるのと同様に、索引の作成もパラレル化できます。複数のプロセスが同時に動作して索引を作成するため、1つのサーバー・プロセスが順に索引を作成する場合よりも高速に索引を作成できます。

索引を並行して作成する場合、問合せサーバー・プロセスごとに別々の記憶域パラメータが使用されます。したがって、INITIAL値を5MB、並行度を12で索引を作成する場合は、作成時に60MB以上の記憶域を使用します。

関連項目:

パラレル実行の使用に関する詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: 索引を管理するためのガイドライン

21.2.10 索引作成時のNOLOGGINGの使用

CREATE INDEX文でNOLOGGINGを指定すると、索引の作成時に最小限のREDOログ・レコードしか生成されません。

> ノート:
>
> NOLOGGING を使用して作成された索引はアーカイブされないため、索引作成後にバックアップを実行してください。

NOLOGGINGを使用して索引を作成すると、次のような利点があります。

- REDOログ・ファイルの領域を節約できます。
- 索引の作成に要する時間が削減できます。
- 大規模な索引のパラレル作成のパフォーマンスが向上します。

一般に、LOGGINGを指定しないで索引を作成した場合、小規模な索引より大規模な索引の方が相対的にパフォーマンスの向上が大きくなります。小規模な索引をLOGGINGを指定しないで作成しても、索引作成に要する時間にはほとんど影響しません。一方、大規模な索引では、特に索引作成のパラレル化もあわせて指定したときに、パフォーマンスが著しく向上します。

親トピック: 索引を管理するためのガイドライン

21.2.11 使用禁止または不可視索引の使用について

使用禁止または不可視索引は、バルク・ロードのパフォーマンスを向上させる場合、索引を削除する前に削除によって発生する影響をテストする場合、またはオプティマイザによるその索引の使用を停止する場合に使用します。

使用禁止索引

使用禁止索引は、オプティマイザで無視され、DMLではメンテナンスされません。索引を使用禁止にする理由の1つに、バルク・

ロードのパフォーマンス向上があります。(行の挿入時にデータベースで索引のメンテナンスを実行する必要がないため、バルク・ロードが速くなります。)索引を削除した後に再度作成する場合はCREATE INDEX文のパラメータを正確に覚えておく必要がありますが、この方法では索引を使用禁止にした後に再構築できます。

使用禁止状態の索引を作成することも、既存の索引または索引パーティションに使用禁止のマークを付けることもできます。場合によっては、データベースで索引の構築中に障害が発生したときなどに、索引に使用禁止のマークが付けられることがあります。パーティション化された索引のパーティションのうちの1つを使用禁止にした場合、その索引の他のパーティションは有効なままです。

使用禁止の索引または索引パーティションを使用するには、再構築するか、または削除して再作成する必要があります。表を切り捨てると、使用禁止の索引が有効になります。

既存の索引を使用禁止にすると、その索引セグメントが削除されます。

使用禁止の索引の機能は、SKIP_UNUSABLE_INDEXESの初期化パラメータの設定により異なります。SKIP_UNUSABLE_INDEXESがTRUE(デフォルト)の場合は、次のようになります。

- 表に対するDML文は続行されますが、使用禁止索引のメンテナンスは実行されません。
- 一意制約を規定する使用禁止索引がある場合は、DML文はエラーで終了します。
- パーティション化されていない索引では、オプティマイザはSELECT文のアクセス計画の作成時に使用禁止索引を考慮しません。唯一の例外は、索引がINDEX( )のヒントで明示的に指定された場合です。
- 1つ以上のパーティションが使用禁止となっているパーティション索引に対しては、オプティマイザでは表の拡張を使用できます。オプティマイザでは、表の拡張を使用して、問合せをUNION ALL文に変換し、ここに、索引付けされたパーティションにアクセスしたり、使用禁止索引を使用してパーティションにアクセスする副問合せを含めます。オプティマイザでは、パーティションで使用できる、最も効率的なアクセス方法を選択できます。表の拡張の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

SKIP_UNUSABLE_INDEXESがFALSEの場合は、次のようになります。

- 使用禁止の索引または索引パーティションがある場合は、これらの索引または索引パーティションを更新するDML文はエラーで終了します。
- SELECT文は、使用禁止索引または使用禁止索引パーティションが存在しても、オプティマイザでアクセス計画に使用されない場合は、通常どおり実行されます。ただし、オプティマイザが使用禁止索引または使用禁止索引パーティションを使用する場合は、この文はエラーで終了します。

不可視索引

不可視索引を作成したり、既存の索引を不可視にできます。不可視索引は、セッションまたはシステム・レベルでOPTIMIZER_USE_INVISIBLE_INDEXES初期化パラメータを明示的にTRUEに設定しないかぎり、オプティマイザで無視されます。使用禁止索引と異なり、不可視索引はDML文中でも維持されます。パーティション化された索引は不可視にできますが、個別の索引パーティションを不可視にし、残りのパーティションを可視のままにすることはできません。

不可視索引を使用して、次のことを実行できます。

- 索引を削除する前に、削除した状態をテストできます。
- アプリケーション全体に影響を与えずに、アプリケーションの特定の操作またはモジュールに対して一時的な索引構造を使用できます。
- 索引がすでに存在する列セットに索引を追加します。

関連項目:

- 「使用禁止索引の作成」
- 「不可視索引の作成」
- 「索引の使用禁止化」
- 「索引の不可視化または可視化」

親トピック: 索引を管理するためのガイドライン

21.2.12 同じ列セットに対する複数の索引の作成について

同じ列セットに対して、一部が異なる複数の索引を作成できます。たとえば、同じ列セットに対してBツリー索引とビットマップ索引を作成できます。

同じ列のセットに複数の索引がある場合、同時に表示できる索引は1つのみで、他の索引は不可視である必要があります。

同じ列セットに対して異なる索引を作成することは、要求を満たすための柔軟性を実現するのに役立ちます。また、既存の索引を削除して異なる属性を使用して再作成することなく、アプリケーションの移行を実行するために、同じ列セットに対して複数の索引を作成することもできます。

使用例が異なると、役立つ索引のタイプが異なります。たとえば、Bツリー索引は、数多くの同時トランザクションが発生するオンライン・トランザクション処理(OLTP)システムで一般的に使用され、ビットマップ索引は、ほとんど問合せに使用されるデータ・ウェアハウス・システムで一般的に使用されます。同様に、ローカル・パーティション化索引とグローバル・パーティション化索引は、異なる使用例において役立ちます。ローカル・パーティション索引は、パーティションのメンテナンス操作が自動的に適用されるため、管理が容易です。グローバル・パーティション索引は、索引のパーティション化スキームを表のパーティション化スキームと異なるものにする場合に役立ちます。

次のうち、少なくとも1つの索引特性が異なる場合に、同じ列セットに対して複数の索引を作成できます。

- 索引のタイプが異なります。

  異なるタイプの索引の詳細は、「索引について」および『Oracle Database概要』を参照してください。

  ただし、次の例外があります。

  - 同じ列セットに対して、Bツリー索引とBツリー・クラスタ索引を作成することはできません。
  - 同じ列セットに対して、Bツリー索引と索引構成表を作成することはできません。

- 索引によって、異なるパーティション化が使用されます。

  パーティション化は、次のように異なることがあります。

  - パーティション化されていない索引と、パーティション化された索引
  - ローカル・パーティション索引と、グローバル・パーティション索引
  - パーティション化タイプ(レンジまたはハッシュ)が異なる索引

- 索引の一意性プロパティが異なります。

  同じ列セットに対して、一意索引と、非一意索引を作成できます。

関連項目:

- 「同じ列セットに対する複数の索引の作成」
- 「使用禁止または不可視索引の使用について」

親トピック: 索引を管理するためのガイドライン

21.2.13 索引の結合と再作成に関するコストと利点の検討

不適切な索引サイズの設定やサイズの拡大によって、索引の断片化が生じることがあります。断片化を解消または低減するには、索引を再作成するか、索引を結合します。ただし、どちらの作業を行う場合も、事前に各選択肢のコストと利点を分析し、状況に最も有効な方法を選択してください。

表21-1は、索引を再作成する場合と結合する場合のコストと利点を示しています。

表21-1 索引の結合と再作成に関するコストと利点

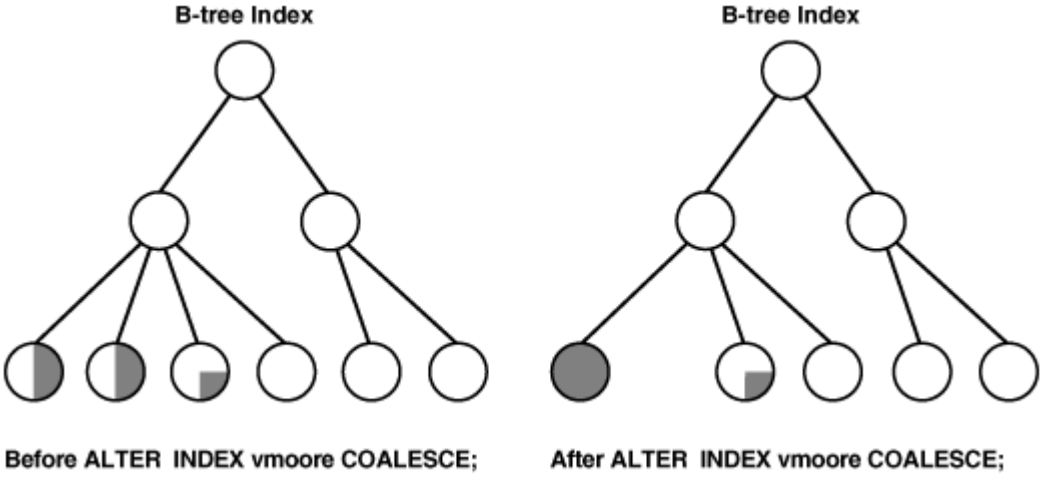
索引の再作成	索引の結合
索引を別の表領域に迅速に移動できる。	索引を別の表領域に移動することはできない。
多くのディスク領域を必要とし、コストが高い。	必要なディスク領域が少ないため、コストが低い。
新しいツリーを作成して、可能であればその高さを縮小する。	ツリーの同じブランチ内のリーフ・ブロックを結合する。
オリジナルの索引を削除せずに、記憶域パラメータと表領域パラメータを迅速に変更できる。	索引のリーフ・ブロックを迅速に解放できる。

再利用のために解放できるBツリー索引のリーフ・ブロックがある場合は、次の文を使用してそのようなリーフ・ブロックをマージできます。

```
ALTER INDEX vmoore COALESCE;
```

図21-1は、ALTER INDEX COALESCEが索引vmooreに与える影響を示しています。操作を実行する前は、最初の2つのリーフ・ブロックは50%使用されています。したがって、フラグメンテーションを削減して最初のブロック全体を使用しながら、2番目のブロックを解放することが可能になります。

図21-1 索引の結合



親トピック: 索引を管理するためのガイドライン

21.2.14 制約を使用禁止または削除する前のコストの検討

一意キーと主キーには対応する索引があるため、UNIQUE制約やPRIMARY KEY制約を使用禁止または削除するかどうかを検討するときには、索引の削除と作成にかかわるコストを考慮に入れてください。

また、UNIQUEキーやPRIMARY KEY制約に対する索引が大きい場合には、その索引を削除して再作成するよりも、その制約を使用可能な状態のままにする方が時間を節約できます。UNIQUE制約やPRIMARY KEY制約を削除または使用禁止にするときには、索引を保持するか削除するかを明示的に指定することもできます。

関連項目:

「整合性制約の管理」

親トピック: 索引を管理するためのガイドライン

21.2.15 索引数を減らすためのインメモリー列ストアの使用の検討

インメモリー列ストアは、システム・グローバル領域(SGA)のオプション部分で、高速スキャン用に最適化された表、表パーティション、その他のデータベース・オブジェクトのコピーが格納されます。インメモリー列ストアでは、表データがSGAに行ではなく列ごとに格納されます。

ノート:

この機能は、Oracle Database 12c リリース 1 (12.1.0.2)以降で使用可能です。

OLTPまたはデータ・ウェアハウス環境で使用される表には通常、分析およびレポート作成の問合せのパフォーマンスを向上させるために、複数の索引が作成されます。このような索引は、データ操作言語(DML)文のパフォーマンスを低下させる可能性があります。インメモリー列ストアに表を格納すると、問合せのパフォーマンスに影響を及ぼすことなく、分析およびレポート作成の問合せに使用される索引を大幅に削減または消去できます。このような索引を消去すると、トランザクションおよびデータ・ロード操作のパフォーマンスを向上させることができます。

関連項目:

「Oracle Database In-Memoryによる問合せパフォーマンスの向上」

親トピック: 索引を管理するためのガイドライン

21.3 索引の作成

いくつかの異なるタイプの索引を作成できます。明示的な索引の作成、および制約に関連付けられた索引の作成ができます。

Live SQL:

Oracle Live SQL での索引の作成に関連する例を参照して実行するには、Oracle Live SQL: 索引の作

成にアクセスしてください。

- 索引の作成の前提条件
  索引を作成する前に、前提条件を満たす必要があります。
- 索引の明示的な作成
  SQL文CREATE INDEXを使用して、明示的に(整合性制約外に)索引を作成できます。
- 一意索引の明示的な作成
  索引は、一意にも非一意にもできます。一意索引を使用すると、表の複数行のキー列に重複した値が入らないことが保証されます。非一意索引では、列の値にこのような制限はありません。
- 制約に対応付けられた索引の作成
  SQL文CREATE TABLEまたはALTER TABLEを発行すると、制約に関連付けられた索引を作成できます。
- 大きな索引の作成
  非常に大きな索引を作成する場合は、索引の作成のために大きな一時表領域の割当てを検討してください。
- 索引のオンラインでの作成
  オンラインで索引を作成および再作成できます。そのため、実表の索引を作成または再作成する際に、同時に実表も更新できます。
- ファンクション索引の作成
  ファンクション索引により、関数や式から返される値を修飾する問合せが容易になります。関数や式の値は、事前に計算されて索引に格納されます。
- 圧縮された索引の作成
  データベースが大きくなったら、ディスク領域を節約するために、索引圧縮の使用を検討してください。
- 使用禁止索引の作成
  UNUSABLE状態の索引を作成した場合、オプティマイザで無視され、DMLではメンテナンスされません。使用禁止の索引を使用可能にする場合、再構築するか、または削除して再作成する必要があります。
- 不可視索引の作成
  不可視索引とは、セッションまたはシステム・レベルでOPTIMIZER_USE_INVISIBLE_INDEXES初期化パラメータを明示的にTRUEに設定しないかぎり、オプティマイザで無視される索引です。
- 同じ列セットに対する複数の索引の作成
  索引がいくらか異なる場合、同じ列セットに複数の索引を作成できます。

親トピック: 索引の管理

21.3.1 索引作成の前提条件

索引を作成する前に、前提条件を満たす必要があります。

自分のスキーマ内に索引を作成する場合は、次のうち少なくとも1つの前提条件が満たされている必要があります。

- 索引を付ける表またはクラスタが、自分のスキーマに格納されている。
- 索引を付ける表に対するINDEX権限を持っている。
- CREATE ANY INDEXシステム権限を持っている。

別のスキーマ内に索引を作成する場合は、次のすべての前提条件が満たされている必要があります。

- CREATE ANY INDEXシステム権限を持っている。
- 目的のスキーマの所有者が、索引または索引パーティションを作成する表領域への割当て制限を持っているか、

UNLIMITED TABLESPACEシステム権限を持っている。

親トピック: 索引の作成

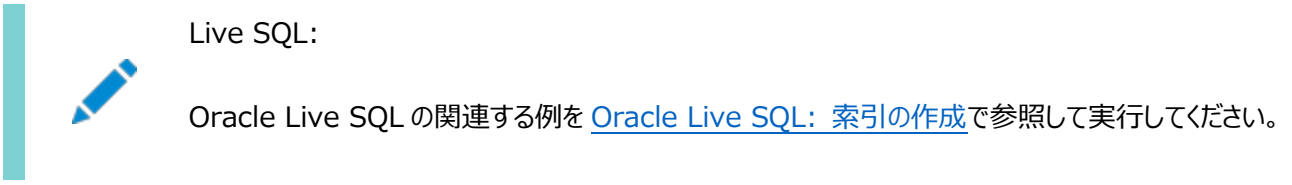
21.3.2 索引の明示的な作成

SQL文CREATE INDEXを使用して、索引を明示的に(整合性制約の他に)作成できます。

次の文は、emp表のename列に対してemp_enameという名前の索引を作成します。

```
CREATE INDEX emp_ename ON emp(ename)
      TABLESPACE users
      STORAGE (INITIAL 20K
      NEXT 20k);
```

索引に対して、複数の記憶域設定および1つの表領域が明示的に指定されています。索引に記憶域オプション(INITIALやNEXTなど)を指定しない場合、デフォルトの表領域または指定された表領域のデフォルトの記憶域オプションが自動的に使用されます。



Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 索引の作成で参照して実行してください。

関連項目:

CREATE INDEX文の構文と制限事項については、『Oracle Database SQL言語リファレンス』

親トピック: 索引の作成

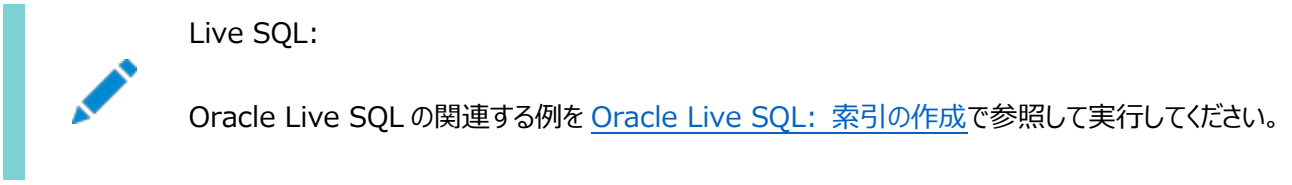
21.3.3 一意索引の明示的な作成

索引は、一意にすることも、重複を許可することもできます。一意索引を使用すると、表の複数行のキー列に重複した値が入らないことが保証されます。非一意索引では、列の値にこのような制限はありません。

一意索引を作成するには、CREATE UNIQUE INDEX文を使用します。次の例では、一意索引を作成しています。

```
CREATE UNIQUE INDEX dept_unique_index ON dept (dname)
      TABLESPACE indx;
```

別の方法として、目的の列に一意整合性制約を定義することもできます。データベースでは、一意のキーに対して自動的に一意索引を定義することによって、一意整合性制約を規定します。この詳細は次の項に記載されています。ただし、問合せのパフォーマンス向上のために必要な索引は、一意索引も含め、明示的に作成することをお薦めします。



Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 索引の作成で参照して実行してください。

関連項目:

パフォーマンス向上のための索引作成の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

親トピック: 索引の作成

21.3.4 制約に対応付けられた索引の作成

SQL文CREATE TABLEまたはALTER TABLEを発行すると、制約に関連付けられた索引を作成できます。

- 制約に対応付けられた索引の作成について
  Oracle Databaseは、表にUNIQUEキー整合性制約またはPRIMARY KEY整合性制約を規定するために、一意キーまたは主キーの一意索引を作成します。
- 制約に対応付けられた索引に対する記憶域オプションの指定
  USING INDEX句を使用すると、UNIQUE制約またはPRIMARY KEY制約に対応する索引の記憶域オプションを設定できます。
- 制約に対応付けられた索引の指定
  制約に対応付けられた索引の詳細を指定できます。

親トピック: 索引の作成

21.3.4.1 制約に対応付けられた索引の作成について

Oracle Databaseは、表にUNIQUEキー整合性制約またはPRIMARY KEY整合性制約を規定するために、一意キーまたは主キーの一意索引を作成します。

この索引は、制約を使用可能にしたときに、データベースによって自動的に作成されます。CREATE TABLE文またはALTER TABLE文を発行して索引を作成する場合は、それ以外に必要なアクションはありませんが、必要に応じて、USING INDEX句を指定して索引作成を制御できます。これは、制約を定義して使用可能にする場合、および定義したが使用禁止にしていた制約を使用可能にする場合のどちらでも可能です。

UNIQUE制約またはPRIMARY KEY制約を使用可能にし、対応する索引を作成するには、表の所有者が索引を格納する表領域の割当て制限またはUNLIMITED TABLESPACEシステム権限を持っている必要があります。特に指定しないかぎり、制約に対応付けられた索引には、常に制約と同じ名前が付けられます。

ノート:

並列性を利用できるように制約を使用可能にする効率的な手順は、「整合性制約の効率的な使用: 手順」を参照してください。

親トピック: 制約に対応付けられた索引の作成

21.3.4.2 制約に対応付けられた索引に対する記憶域オプションの指定

USING INDEX句を使用すると、 UNIQUE制約またはPRIMARY KEY制約に対応する索引の記憶域オプションを設定できます。

次のCREATE TABLE文は、主キー制約を使用可能にして、対応する索引の記憶域オプションを指定します。

```
CREATE TABLE emp (
     empno NUMBER(5) PRIMARY KEY, age INTEGER)
     ENABLE PRIMARY KEY USING INDEX
     TABLESPACE users;
```

親トピック: 制約に対応付けられた索引の作成

21.3.4.3 制約に対応付けられた索引の指定

制約に対応付けられた索引の詳細を指定できます。

UNIQUE制約およびPRIMARY KEY制約に対応付けられた索引をより明示的に制御する場合、データベースでは次のことが可能です。

- 制約を規定するために使用する既存の索引の指定
- 索引の作成と制約の規定に使用するCREATE INDEX文の指定

これらのオプションは、USING INDEX句を使用して指定します。次にいくつかの例を示します。

例1:

```
CREATE TABLE a (
     a1 INT PRIMARY KEY USING INDEX (create index ai on a (a1)));
```

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 索引の作成で参照して実行してください。

例2:

```
CREATE TABLE b(
     b1 INT,
     b2 INT,
     CONSTRAINT bu1 UNIQUE (b1, b2)
                    USING INDEX (create unique index bi on b(b1, b2)),
     CONSTRAINT bu2 UNIQUE (b2, b1) USING INDEX bi);
```

例3:

```
CREATE TABLE c(c1 INT, c2 INT);
CREATE INDEX ci ON c (c1, c2);
ALTER TABLE c ADD CONSTRAINT cpk PRIMARY KEY (c1) USING INDEX ci;
```

単一の文で制約とともに索引を作成し、同じ文でその索引を別の制約にも使用する場合、Oracleでは、索引を再使用する前に索引を作成するための句の再編成を試みます。

関連項目:

「整合性制約の管理」

親トピック: 制約に対応付けられた索引の作成

21.3.5 大きな索引の作成

非常に大きな索引を作成する場合は、索引の作成のために大きな一時表領域の割当てを検討してください。

これを行うには、次のステップを完了します。

1. CREATE TABLESPACE文またはCREATE TEMPORARY TABLESPACE文を使用して、新しい一時表領域を作成します。

2. これを自分の新しい一時表領域にするには、ALTER USER文のTEMPORARY TABLESPACEオプションを使用します。
3. CREATE INDEX文を使用して、索引を作成します。
4. DROP TABLESPACE文を使用して、この表領域を削除します。次にALTER USER文を使用して、自分の一時表領域を元の一時表領域に再設定します。

この手順により、通常使用している一時表領域(ほとんどの場合、共有されている)が極度に肥大化して今後のパフォーマンスに影響を及ぼす問題を回避できます。

親トピック: 索引の作成

21.3.6 オンラインでの索引の作成

索引をオンラインで作成および再作成できます。そのため、実表の索引を作成または再作成する際に、同時に実表も更新できます。

索引の作成中にDML操作を実行できますが、DDL操作はできません。索引をオンラインで作成または再作成するとき、パラレルDMLはサポートされません。

次の文は、オンラインでの索引作成操作を示しています。

```
CREATE INDEX emp_name ON emp (mgr, emp1, emp2, emp3) ONLINE;
```

ノート:

オンライン索引ビルドが完了するまでの時間は、表のサイズおよび同時に実行している DML 文の数に比例することに注意してください。したがって、オンライン索引ビルドは DML アクティビティが低いときに開始することをお薦めします。

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 索引の作成で参照して実行してください。

関連項目:

「既存の索引の再作成」

親トピック: 索引の作成

21.3.7 ファンクション索引の作成

ファンクション索引を使用すると、関数や式から返される値を修飾する問合せが可能です。関数や式の値は、事前に計算されて索引に格納されます。

ユーザー定義のファンクションに基づく索引の場合は、従来型の索引を作成するための前提条件に加えて、これらのファンクションにDETERMINISTICのマークが設定されている必要があります。また、ファンクションに基づく索引はファンクションの所有者の資格証明を使用して実行されるため、ファンクションに対するEXECUTEオブジェクト権限がある必要があります。

ノート:

CREATE INDEX では、ファンクション索引で最後に使用された関数のタイムスタンプが格納されます。このタイムスタンプは、索引の妥当性チェック時に更新されます。ファンクション索引について表領域の Point-in-Time リカバリを実行する場合、索引で最後に使用された関数のタイムスタンプが索引に格納されたタイムスタンプより新しい場合は、その索引には無効を示すマークが設定されます。ANALYZE INDEX...VALIDATE STRUCTURE 文を使用して、この索引の妥当性をチェックする必要があります。

ファンクション索引の具体例として、ファンクションarea(geo)に定義されているファンクション索引(area_index)を定義する次の文を示します。

```
CREATE INDEX area_index ON rivers (area(geo));
```

次のSQL文では、area(geo)がWHERE句で参照されているため、オプティマイザは索引area_indexの使用を考慮します。

```
SELECT id, geo, area(geo), desc
     FROM rivers
     WHERE Area(geo) >5000;
```

ファンクション索引は使用するファンクションに依存するため、ファンクションの変更時に無効にできます。ファンクションが有効な場合は、ALTER INDEX...ENABLE文を使用して、使用禁止になっているファンクション索引を使用可能にすることができます。ALTER INDEX...DISABLE文では、ファンクション索引を使用禁止にすることができます。ファンクションの本体で作業をする場合は、ファンクション索引を使用禁止にするか検討してください。

ノート:

ファンクション索引を作成するかわりに、仮想列をターゲット表に追加して仮想列に索引を付けることができます。詳細は、「表について」を参照してください。

関連項目:

- ファンクション索引の詳細は、『Oracle Database概要』を参照してください。
- アプリケーションでファンクション索引を使用する際の詳細と使用例については、『Oracle Database開発ガイド』を参照してください。

親トピック: 索引の作成

21.3.8 圧縮索引の作成

データベースが大きくなったら、ディスク領域を節約するために、索引圧縮の使用を検討してください。

- 接頭辞圧縮を使用した索引の作成
  接頭辞圧縮(キー圧縮とも呼ばれます)を使用して索引を作成すると、キー列の接頭辞が同じ値で繰り返し格納されることが回避されます。接頭辞圧縮は、先頭列で数多くの重複が発生する非一意索引に対して最も有効です。
- 高度な索引圧縮を使用した索引の作成
  高度な索引圧縮は、接頭辞圧縮の候補として適切でないものを含め、サポートされているすべての索引で適切に機

能します。拡張索引圧縮を使用して索引を作成すると、索引への効率的なアクセスを提供しつつ、すべての一意の索引および一意ではない索引のサイズが減少し、圧縮率が著しく向上します。

親トピック: 索引の作成

21.3.8.1 接頭辞圧縮を使用した索引の作成

接頭辞圧縮(キー圧縮とも呼ばれます)を使用して索引を作成すると、キー列の接頭辞が同じ値で繰り返し格納されることが回避されます。接頭辞圧縮は、先頭列で数多くの重複が発生する非一意索引に対して最も有効です。

接頭辞圧縮によって、索引キーは接頭辞および接尾辞エントリに分割されます。圧縮するために、接頭辞エントリは索引ブロック内のすべての接尾辞エントリ間で共有されます。このような共有によって、領域が大幅に節約され、各索引ブロックに格納できるキー数が増え、パフォーマンスが向上します。

接頭辞圧縮は、次のような状況で役立ちます。

- ROWIDを追加してキーを一意にしている非一意索引がある場合。このような状況で接頭辞圧縮を使用すると、重複キーは接頭辞エントリとして索引ブロックにROWIDなしで格納されます。残りの行は、ROWIDのみからなる接尾辞エントリとなります。
- 一意の複数列索引がある場合。

接頭辞圧縮を使用可能にするには、COMPRESS句を使用します。また、接頭辞の長さをキー列の数で指定して、キー列が接頭辞および接尾辞エントリにどのように分割されるかを識別できます。たとえば、次の文は、索引リーフ・ブロックの重複するキーを圧縮します。

```
CREATE INDEX  hr.emp_ename ON emp(ename)
   TABLESPACE users
   COMPRESS 1;
```

また、再作成中に、COMPRESS句を指定することもできます。たとえば、次のようにして、再作成中に圧縮を使用禁止にすることができます。

```
ALTER INDEX hr.emp_ename REBUILD NOCOMPRESS;
```

ALL_INDEXESビューおよびALL_PART_INDEXESビューのCOMPRESSION列には、索引が圧縮されるかどうかが表示され、圧縮される場合は索引で使用可能な圧縮のタイプが表示されます。

Live SQL:

Oracle Live SQL の関連する例を Oracle Live SQL: 索引の作成で参照して実行してください。

関連項目:

- Oracle Database SQL言語リファレンス
- 接頭辞圧縮の詳細は、『Oracle Database概要』を参照してください

親トピック: 圧縮索引の作成

21.3.8.2 拡張索引圧縮を使用した索引の作成

拡張索引圧縮は、接頭辞圧縮の適切な候補ではない索引を含むすべてのサポートされている索引で正常に動作します。拡

張索引圧縮を使用して索引を作成すると、索引への効率的なアクセスを提供しつつ、すべての一意の索引および一意ではない索引のサイズが減少し、圧縮率が著しく向上します。

パーティション索引の場合は、パーティションごとにパーティションの圧縮タイプを指定できます。親索引が圧縮されていない場合でも、索引パーティションに対して拡張索引圧縮を指定できます。

拡張索引圧縮は、ブロック・レベルに作用し、各ブロックを最適に圧縮できます。

拡張索引圧縮を有効にするにはCOMPRESS ADVANCED句を使用し、次の圧縮レベルを指定できます。

- LOW: このレベルは、最小のCPUオーバーヘッドで低い圧縮率を提供します。COMPRESS ADVANCED LOWを有効化する前に、データベースの互換性レベルが12.1.0以上である必要があります。
- HIGH: このレベル(デフォルト)では、わずかなCPUオーバーヘッドでより高い圧縮率が提供されます。COMPRESS ADVANCED HIGHを有効化する前に、データベースの互換性レベルが12.2.0以上である必要があります。

  CREATE INDEX DDL文を実行すると、ブロックは行で満たされます。高圧縮レベルでブロックが一杯になると、次の行を挿入するための十分な領域が確保される場合、ブロックは拡張索引圧縮を使用して圧縮されます。ブロックが一杯なると、受信キーの挿入のための十分な領域が確保される場合、ブロックは拡張索引圧縮を使用して再圧縮され分割を回避します。

ALL_INDEXESビューのCOMPRESSION列には、索引が圧縮されるかどうかが表示され、圧縮される場合は索引で使用可能な圧縮のタイプが表示されます。COMPRESSION列に指定できる値は、ADVANCED HIGH、ADVANCED LOW、DISABLEDまたはENABLEDです。ALL_IND_PARTITIONSビューおよびALL_IND_SUBPARTITIONSビューのCOMPRESSION列は、パーティションまたはサブパーティションで索引圧縮がENABLEDまたはDISABLEDのいずれであるかを示します。

ノート:

- 拡張索引圧縮は、ビットマップ索引または索引構成表ではサポートされていません。
- 低レベルの拡張索引圧縮を有効にした場合は、単一列の一意の索引に拡張索引圧縮を指定できません。高レベルの拡張索引圧縮が有効にされている場合、この制限は適用されません。

例21-1 索引作成時に低レベルの拡張索引圧縮を有効にする

たとえば、次の文を実行すると、hr.emp_mndp_ix索引の作成時に低レベルの拡張索引圧縮が使用可能になります。

```
CREATE INDEX hr.emp_mndp_ix ON hr.employees(manager_id, department_id)
   COMPRESS ADVANCED LOW;
```

例21-2 索引の再作成時に高レベルの拡張索引圧縮を有効にする

また、索引の再作成中に、COMPRESS ADVANCED句を指定することもできます。たとえば、次のようにして、再作成中にhr.emp_manager_ix索引に対して高レベルの拡張索引圧縮を使用可能にできます。

```
ALTER INDEX hr.emp_manager_ix REBUILD COMPRESS ADVANCED HIGH;
```

親トピック: 圧縮索引の作成

21.3.9 使用禁止索引の作成

UNUSABLE状態の索引を作成した場合、オプティマイザで無視され、DMLではメンテナンスされません。使用禁止の索引を使

用可能にする場合、再構築するか、または削除して再作成する必要があります。

索引がパーティション化されている場合、索引パーティションはすべてUNUSABLEとしてマークされます。

データベースは使用禁止索引の作成時に索引セグメントを作成しません。

次の手順では、使用禁止の索引を作成する方法と、データベースに索引に関する詳細を問い合せる方法を示します。

使用禁止索引を作成するには:

1. 必要に応じて、索引を付ける表を作成します。

   たとえば、次のように、hr.employees_partという名前でハッシュ・パーティション表を作成します。

```
sh@PROD> CONNECT hr
Enter password: **
Connected.
hr@PROD> CREATE TABLE employees_part
  2    PARTITION BY HASH (employee_id) PARTITIONS 2
  3    AS SELECT * FROM employees;

Table created.
hr@PROD> SELECT COUNT(*) FROM employees_part;

  COUNT(*)
----------
       107
```

2. キーワードUNUSABLEで索引を作成します。

   次の例では、employees_partにローカル・パーティション索引を作成し、索引パーティションに
   p1_i_emp_enameおよびp2_i_emp_enameという名前を付け、p1_i_emp_enameを使用禁止にしています。

```
hr@PROD> CREATE INDEX i_emp_ename ON employees_part (employee_id)
  2    LOCAL (PARTITION p1_i_emp_ename UNUSABLE, PARTITION p2_i_emp_ename);

Index created.
```

3. (オプション)データ・ディクショナリに問い合せて索引が使用禁止になっていることを確認できます。

   次の例では、索引i_emp_enameとその2つのパーティションのステータスを問い合せて、パーティション
   p2_i_emp_enameのみが使用禁止になっていることを示しています。

```
hr@PROD> SELECT INDEX_NAME AS "INDEX OR PARTITION NAME", STATUS
  2  FROM   USER_INDEXES
  3  WHERE  INDEX_NAME = 'I_EMP_ENAME'
  4  UNION ALL
  5  SELECT PARTITION_NAME AS "INDEX OR PARTITION NAME", STATUS
  6  FROM   USER_IND_PARTITIONS
  7  WHERE  PARTITION_NAME LIKE '%I_EMP_ENAME%';

INDEX OR PARTITION NAME        STATUS
------------------------------ --------
I_EMP_ENAME                    N/A
P1_I_EMP_ENAME                 UNUSABLE
P2_I_EMP_ENAME                 USABLE
```

4. (オプション)データ・ディクショナリに問い合せて、パーティションの記憶域が存在するかどうかを確認できます。

   たとえば、次の問合せは索引パーティションp2_i_emp_enameのみがセグメントを占有していることを示しています。
   p1_i_emp_enameは使用禁止として作成したため、セグメントが割り当てられませんでした。

```
hr@PROD> COL PARTITION_NAME FORMAT a14
hr@PROD> COL SEG_CREATED FORMAT a11
```

```
hr@PROD> SELECT p.PARTITION_NAME, p.STATUS AS "PART_STATUS",
  2         p.SEGMENT_CREATED AS "SEG_CREATED",
  3  FROM   USER_IND_PARTITIONS p, USER_SEGMENTS s
  4  WHERE  s.SEGMENT_NAME = 'I_EMP_ENAME';

PARTITION_NAME PART_STA SEG_CREATED
-------------- -------- -----------
P2_I_EMP_ENAME USABLE   YES
P1_I_EMP_ENAME UNUSABLE NO
```

関連項目:

- 「使用禁止または不可視索引の使用について」
- 「索引の使用禁止化」
- 使用禁止索引の作成方法および制限事項の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 索引の作成

21.3.10 不可視索引の作成

不可視索引とは、セッションまたはシステム・レベルでOPTIMIZER_USE_INVISIBLE_INDEXES初期化パラメータを明示的にTRUEに設定しないかぎり、オプティマイザで無視される索引です。

不可視索引を作成するには:

- CREATE INDEX文をINVISIBLEキーワードとともに指定します。

  次の文は、emp表のename列に対してemp_enameという名前の不可視索引を作成します。

```
CREATE INDEX emp_ename ON emp(ename)
      TABLESPACE users
      STORAGE (INITIAL 20K
      NEXT 20k)
      INVISIBLE;
```

関連項目:

- 「使用禁止または不可視索引の使用について」
- 「索引の不可視化または可視化」
- 不可視索引の作成の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 索引の作成

21.3.11 同じ列セットに対する複数の索引の作成

同じ列セットに対して、一部が異なる複数の索引を作成できます。

同じ列セットに対して複数の索引を作成する場合は、次の前提条件が満たされている必要があります。

- 「索引の作成」に示されている、必要な権限の前提。
- 同じ列セットに対する索引のうち、任意の時点で可視にできるのは1つのみとなります。

可視の索引を作成する場合、その列セットに対するすべての既存の索引が不可視である必要があります。

かわりに、列セットに対して不可視索引を作成できます。

たとえば、次のステップを実行すると、oe.orders表の同じ列セットに対して、Bツリー索引とビットマップ索引が作成されます。

1. oe.orders表のcustomer_id列とsales_rep_id列に対して、Bツリー索引を作成します。

   ```
   CREATE INDEX oe.ord_customer_ix1 ON oe.orders (customer_id, sales_rep_id);
   ```

   oe.ord_customer_ix1索引はデフォルトで可視となります。

2. ステップ1で作成した索引を変更して、不可視にします。

   ```
   ALTER INDEX oe.ord_customer_ix1 INVISIBLE;
   ```

   かわりに、ステップ1でINVISIBLE句を追加すると、このステップを省略できます。

3. oe.orders表のcustomer_id列とsales_rep_id列に対して、ビットマップ索引を作成します。

   ```
   CREATE BITMAP INDEX oe.ord_customer_ix2 ON oe.orders (customer_id,
   sales_rep_id);
   ```

   oe.ord_customer_ix2索引はデフォルトで可視となります。

   ステップ1で作成したoe.ord_customer_ix1索引が可視である場合、このステップのCREATE BITMAP INDEX文ではエラーが返されます。

関連項目:

- 「同じ列セットに対する複数の索引の作成について」
- 「使用禁止または不可視索引の使用について」
- 「不可視索引の作成」

親トピック: 索引の作成

21.4 索引の変更

索引は、記憶域の特性の変更、再作成、使用不能化、表示/非表示にするなどのタスクを完了することで変更できます。

- 索引の変更について
  索引を変更するには、その索引が自分のスキーマに含まれているか、またはALTER ANY INDEXシステム権限を持っている必要があります。
- 索引の記憶域特性の変更
  主キーと一意キーの整合性制約を規定するためにデータベースによって作成される索引も含めて、どの索引の記憶域パラメータも、ALTER INDEX文を使用して変更できます。
- 既存の索引の再作成
  索引を再作成するときは、既存の索引をデータソースとして使用できます。このようにして索引を作成すると、記憶特性の変更や新しい表領域への移動ができます。既存のデータソースに基づいて索引を再作成すると、ブロック内の断片化も解消されます。
- 索引の使用禁止化
  索引を使用禁止にした場合、オプティマイザで無視され、DMLではメンテナンスされません。パーティション化された索

引のパーティションのうちの1つを使用禁止にした場合、その索引の他のパーティションは有効なままです。

- 索引の不可視化または可視化
  索引を非表示にするのは、使用禁止または削除の代替手段です。
- 索引の名前変更
  RENAME句を含むALTER INDEX文を使用して、索引の名前を変更できます。
- 索引の使用状況の監視
  Oracle Databaseには、索引を監視し、それらが使用されているかどうかを判別する手段が用意されています。未使用の索引がある場合は、それを削除して不要な文によるオーバーヘッドを解消できます。

親トピック: 索引の管理

21.4.1 索引の変更について

索引を変更するには、その索引が自分のスキーマに含まれているか、またはALTER ANY INDEXシステム権限を持っている必要があります。

ALTER INDEX文では、次の操作を実行できます。

- 既存の索引の再作成または結合
- 未使用領域の割当て解除または新規エクステントの割当て
- パラレル実行の指定(または指定解除)およびその並列度の変更
- 記憶域パラメータまたは物理属性の変更
- LOGGINGまたはNOLOGGINGの指定
- 接頭辞圧縮の使用可能または使用禁止の設定
- 拡張圧縮を有効化または無効化します
- 索引へのUNUSABLEマークの設定
- 索引の不可視化
- 索引の名前変更
- 索引使用状況の監視の開始または停止

索引の列構造は変更できません。

関連項目:

- ALTER INDEX文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 索引の変更

21.4.2 索引の記憶域特性の変更

主キーと一意キーの整合性制約を規定するためにデータベースによって作成される索引も含めて、どの索引の記憶域パラメータも、ALTER INDEX文を使用して変更できます。

たとえば、次の文はemp_ename索引を変更します。

```
ALTER INDEX emp_ename
```

```
     STORAGE (NEXT 40);
```

パラメータINITIALとMINEXTENTSは変更できません。他の記憶域パラメータの新しい設定はすべて、その後に索引に割り当てられるエクステントにのみ影響します。

整合性制約を実装する索引では、ENABLE句のUSING INDEX副次句を指定したALTER TABLE文を発行することによって、記憶域パラメータを調整できます。たとえば、次の文は、表empに作成された索引の記憶域オプションを変更して、主キー制約を規定します。

```
ALTER TABLE emp
     ENABLE PRIMARY KEY USING INDEX;
```

関連項目:

ALTER INDEX文の構文と制限事項については、『Oracle Database SQL言語リファレンス』

親トピック: 索引の変更

21.4.3 既存の索引の再作成

索引を再作成するときは、既存の索引をデータソースとして使用できます。このようにして索引を作成すると、記憶特性の変更や新しい表領域への移動ができます。既存のデータソースに基づいて索引を再作成すると、ブロック内の断片化も解消されます。

索引を削除してからCREATE INDEX文を使用するよりも、既存の索引を再構築する方がパフォーマンスに優れています。既存の索引を再作成する前に、「索引の結合と再作成に関するコストと利点の検討」の説明に従って、索引を再作成する方法と結合する方法のコストと利点を比較してください。

次の文は、既存の索引emp_nameを再作成します。

```
ALTER INDEX emp_name REBUILD;
```

REBUILD句は索引名の直後に使用し、他のオプションの前に置く必要があります。DEALLOCATE UNUSED句とともに使用することはできません。

索引はオンラインで再作成できます。オンラインで再作成することにより、再作成と同時に実表を更新できます。次の文は、emp_name索引をオンラインで再作成します。

```
ALTER INDEX emp_name REBUILD ONLINE;
```

別のユーザーのスキーマにオンラインで索引を再作成するには、ALTER ANY INDEXシステム権限が必要です。

ノート:

オンラインでの索引の再作成には、別の方式による索引の再作成に比べ、処理できるキーの最大長に関して、より厳しい制限があります。オンラインでの再作成時に ORA-01450(キーの最大長超過)エラーが発生した場合は、オフラインで再作成し結合を試みるか、索引を削除し再作成します。

索引の再作成に必要な大きさの領域がない場合、かわりに索引を結合する方法が使用できます。索引の結合はオンライン操作です。

関連項目:

- 「索引のオンラインでの作成」
- 「索引の領域使用の監視」

親トピック: 索引の変更

21.4.4 索引の使用禁止化

索引を使用禁止にした場合、オプティマイザで無視され、DMLではメンテナンスされません。パーティション化された索引のパーティションのうちの1つを使用禁止にした場合、その索引の他のパーティションは有効なままです。

使用禁止の索引または索引パーティションを使用する場合は、事前に索引または索引パーティションを再作成するか、または削除した後再作成する必要があります。

次の手順では、索引および索引パーティションを使用禁止にする方法と、オブジェクトのステータスを問い合せる方法を示します。

索引を使用禁止にするには:

1. データ・ディクショナリに問い合せて、既存の索引または索引パーティションが使用可能であるか、使用禁止であるかを確認します。

   たとえば、次の問合せを発行します(スペースの節約のため出力を切り捨てています)。

```
hr@PROD> SELECT INDEX_NAME AS "INDEX OR PART NAME", STATUS, SEGMENT_CREATED
  2  FROM   USER_INDEXES
  3  UNION ALL
  4  SELECT PARTITION_NAME AS "INDEX OR PART NAME", STATUS, SEGMENT_CREATED
  5  FROM   USER_IND_PARTITIONS;

INDEX OR PART NAME              STATUS   SEG
------------------------------ -------- ---
I_EMP_ENAME                    N/A      N/A
JHIST_EMP_ID_ST_DATE_PK        VALID    YES
JHIST_JOB_IX                   VALID    YES
JHIST_EMPLOYEE_IX              VALID    YES
JHIST_DEPARTMENT_IX            VALID    YES
EMP_EMAIL_UK                   VALID    NO
.
.
.
COUNTRY_C_ID_PK                VALID    YES
REG_ID_PK                      VALID    YES
P2_I_EMP_ENAME                 USABLE   YES
P1_I_EMP_ENAME                 UNUSABLE NO

22 rows selected.
```

   前述の出力は、索引パーティションp1_i_emp_enameのみが使用禁止であることを示しています。

2. 索引または索引パーティションを使用禁止にするには、UNUSABLEキーワードを指定します。

   次の例では、索引emp_email_ukを使用禁止にしています。

```
hr@PROD> ALTER INDEX emp_email_uk UNUSABLE;

Index altered.
```

   次の例では、索引パーティションp2_i_emp_enameを使用禁止にしています。

```
hr@PROD> ALTER INDEX i_emp_ename MODIFY PARTITION p2_i_emp_ename UNUSABLE;
```

```
Index altered.
```

3. (オプション)データ・ディクショナリに問い合せて、ステータス変更を確認できます。

   たとえば、次の問合せを発行します(スペースの節約のため出力を切り捨てています)。

```
hr@PROD> SELECT INDEX_NAME AS "INDEX OR PARTITION NAME", STATUS,
  2  SEGMENT_CREATED
  3  FROM   USER_INDEXES
  4  UNION ALL
  5  SELECT PARTITION_NAME AS "INDEX OR PARTITION NAME", STATUS,
  6  SEGMENT_CREATED
  7  FROM   USER_IND_PARTITIONS;

INDEX OR PARTITION NAME         STATUS   SEG
------------------------------- -------- ---
I_EMP_ENAME                     N/A      N/A
JHIST_EMP_ID_ST_DATE_PK         VALID    YES
JHIST_JOB_IX                    VALID    YES
JHIST_EMPLOYEE_IX               VALID    YES
JHIST_DEPARTMENT_IX             VALID    YES
EMP_EMAIL_UK                    UNUSABLE NO
.
.
.
COUNTRY_C_ID_PK                 VALID    YES
REG_ID_PK                       VALID    YES
P2_I_EMP_ENAME                  UNUSABLE NO
P1_I_EMP_ENAME                  UNUSABLE NO

22 rows selected.
```

   i_emp_enameセグメントとemp_email_ukセグメントが消費している領域を問い合せると、どちらのセグメントも存在しなくなったことが示されます。

```
hr@PROD> SELECT SEGMENT_NAME, BYTES
  2  FROM   USER_SEGMENTS
  3  WHERE  SEGMENT_NAME IN ('I_EMP_ENAME', 'EMP_EMAIL_UK');

no rows selected
```

関連項目:

- 「使用禁止または不可視索引の使用について」
- 「使用禁止索引の作成」
- UNUSABLEキーワードの詳細および制限については、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 索引の変更

21.4.5 索引の不可視化または可視化

索引を非表示にするのは、使用禁止または削除の代替手段です。

不可視索引は、セッションまたはシステム・レベルでOPTIMIZER_USE_INVISIBLE_INDEXES初期化パラメータを明示的にTRUEに設定しないかぎり、オプティマイザで無視されます。個別の索引パーティションは不可視にできません。試みた場合はエラーとなります。

索引を不可視にするには:

- 次のSQL文を発行します。

```
ALTER INDEX index INVISIBLE;
```

不可視状態の索引を可視にするには:

- 次のSQL文を発行します。

```
ALTER INDEX index VISIBLE;
```

ノート:

同じ列セットに対して複数の索引がある場合、これらの索引のうち、任意の時点で可視にできるのは 1 つのみとなります。列セットに対する索引を可視にしようとしたとき、同じ列セットに対する別の索引が可視であると、エラーが返されます。

索引が可視か不可視かを判別するには:

- USER_INDEXES、ALL_INDEXESまたはDBA_INDEXESディクショナリ・ビューの問合せを行います。

  たとえば、索引ind1が不可視かどうかを判別するには、次の問合せを発行します。

```
SELECT INDEX_NAME, VISIBILITY FROM USER_INDEXES
   WHERE INDEX_NAME = 'IND1';
INDEX_NAME   VISIBILITY
----------   ----------
IND1         VISIBLE
```

関連項目:

- 「使用禁止または不可視索引の使用について」
- 「不可視索引の作成」
- 「同じ列セットに対する複数の索引の作成」

親トピック: 索引の変更

21.4.6 索引の名前変更

RENAME句を含むALTER INDEX文を使用して、索引の名前を変更できます。

索引の名前を変更するには、次の文を発行します。

```
ALTER INDEX index_name RENAME TO new_name;
```

親トピック: 索引の変更

21.4.7 索引の使用状況の監視

Oracle Databaseには、索引を監視し、それらが使用されているかどうかを判別する手段が用意されています。未使用の索引がある場合は、それを削除して不要な文によるオーバーヘッドを解消できます。

索引の使用状況の監視を開始するには、次の文を発行します。

```
ALTER INDEX index MONITORING USAGE;
```

その後、監視を停止するには、次の文を発行します。

```
ALTER INDEX index NOMONITORING USAGE;
```

監視中の索引について、それが使用されているかどうかを確認するには、ビューUSER_OBJECT_USAGEを問い合せます。このビューにはUSED列があり、監視期間中に索引が使用されたかどうかによってYESまたはNOの値を持ちます。また、このビューには監視の開始時間と停止時間も記録され、MONITORING列(YES/NO)には使用状況の監視が現在アクティブであるかどうかが示されます。

MONITORING USAGEを指定するたびに、指定した索引のUSER_OBJECT_USAGEビューはリセットされます。前回の使用方法の情報はクリアまたはリセットされ、新しい開始時間が記録されます。NOMONITORING USAGEを指定すると、これ以上の監視は実行されず、監視期間の終了時間が記録されます。次にALTER INDEX...MONITORING USAGE文が発行されるまで、ビューの情報は変更されずに保持されます。

親トピック: 索引の変更

21.5 索引の領域使用の監視

索引のキー値が頻繁に挿入、更新および削除されると、時間が経つにつれて領域の効率性が失われることがあります。

索引の領域の使用状況の効率性を定期的に監視するには、まずANALYZE INDEX...VALIDATE STRUCTURE文を使用し、次にINDEX_STATSビューを問い合せて索引構造を分析します。

```
SELECT PCT_USED FROM INDEX_STATS WHERE NAME = 'index';
```

索引の領域使用の割合は、どれくらい頻繁に索引キーが挿入、更新または削除されるかによって変化します。次の一連の操作を何度か実行し、索引の平均的な領域使用効率の履歴を作成してください。

- 統計分析
- 索引の検証
- PCT_USEDのチェック
- 索引の削除および再作成(または結合)

索引の領域使用がその平均を下回っているときは、索引を削除してから再作成または結合することによって、索引の領域を圧縮できます。

関連項目:

「表、索引およびクラスタの分析について」

親トピック: 索引の管理

21.6 索引の削除

索引は、DROP INDEX文により削除できます。

索引を削除するには、その索引が自分のスキーマに含まれているか、またはDROP ANY INDEXシステム権限を持っている必要があります。

索引を削除するのは、次のような場合です。

- 索引が不要になった場合。
- 対応する表に対して発行した問合せで、索引が予想されたパフォーマンスの改善を達成していない場合。たとえば、表が非常に小さい、または表には多くの行があるものの、索引エントリが非常に少ないなどがこれに該当します。
- アプリケーションに索引を使用するデータ問合せが含まれない場合。
- 索引が無効になり、再作成する前に削除する必要がある場合。
- 索引がかなり断片化し、再作成する前に削除する必要がある場合。

索引が削除されると、その索引のセグメントのエクステントはすべて、索引を含んでいる表領域に戻され、表領域内の他のオブジェクトで利用できます。

索引を削除する方法は、索引の作成方法、つまりCREATE INDEX文によって索引を明示的に作成したか、または表にキー制約を定義することによって索引を暗黙的に作成したかによって異なります。CREATE INDEX文を使用して明示的に作成した索引は、DROP INDEX文で削除できます。次の文は、emp_ename索引を削除します。

```
DROP INDEX emp_ename;
```

使用可能になっているUNIQUEキー制約やPRIMARY KEY制約に対応付けられた索引のみを削除することはできません。制約に対応付けられた索引を削除するには、制約自体を使用禁止にするかまたは削除します。

ノート:

表を削除すると、対応する索引はすべて自動的に削除されます。

関連項目:

- DROP INDEX文の構文と制限事項については、『Oracle Database SQL言語リファレンス』
- 「整合性制約の管理」
- 索引の削除にかわる方法については、「索引の不可視化または可視化」

親トピック: 索引の管理

21.7 自動索引の管理

自動索引作成機能を使用することで、Oracleデータベースで自動索引を構成して使用し、データベースのパフォーマンスを向上させることができます。

- 自動索引作成について
  自動索引作成機能では、Oracleデータベース内の索引管理タスクが自動化されます。自動索引作成では、アプリケーション・ワークロードの変化に基づいてデータベース内の索引が自動的に作成および削除されるため、データベース・パフォーマンスが向上します。自動的に管理される索引は、自動索引と呼ばれます。
- 自動索引作成の動作
  この項では、自動索引作成がどのように機能するかを説明します。
- Oracle Databaseでの自動索引作成の構成
  DBMS_AUTO_INDEX.CONFIGUREプロシージャを使用して、Oracleデータベースでの自動索引作成を構成できます。

- 自動索引作成レポートの生成
  Oracleデータベースでの自動索引作成操作に関連するレポートを生成するには、DBMS_AUTO_INDEXパッケージのREPORT_ACTIVITYファンクションとREPORT_LAST_ACTIVITYファンクションを使用します。
- 自動索引作成情報を含むビュー
  一連のデータ・ディクショナリ・ビューを問い合せて、Oracleデータベースでの自動索引に関する情報を取得できます。

親トピック: 索引の管理

21.7.1 自動索引作成について

自動索引作成機能では、Oracleデータベース内の索引管理タスクが自動化されます。自動索引作成では、アプリケーション・ワークロードの変化に基づいてデータベース内の索引が自動的に作成および削除されるため、データベース・パフォーマンスが向上します。自動的に管理される索引は、自動索引と呼ばれます。

索引構造は、データベース・パフォーマンスにとって重要な機能です。OLTPアプリケーションでは、索引は非常に重要です。これらのアプリケーションでは、大規模なデータ・セットが使用され、1日に何百万ものSQL文が実行されます。データ・ウェアハウス・アプリケーションでも、索引は非常に重要です。これらのアプリケーションでは、通常は、非常に大きな表から比較的少量のデータを問い合せます。アプリケーション・ワークロードに変化があったときに索引を更新しないと、既存の索引が原因で、データベースのパフォーマンスが大幅に低下する可能性があります。

自動索引作成では、アプリケーション・ワークロードの変化に基づいてOracleデータベース内で索引を自動的かつ動的に管理することで、データベースのパフォーマンスが向上します。

自動索引作成では、次の機能が提供されます。

- バックグラウンドで、事前定義された時間間隔で定期的に自動索引作成プロセスを実行します。
- アプリケーション・ワークロードを分析し、それに応じて新しい索引を作成し、低パフォーマンスな既存の索引を削除して、データベース・パフォーマンスを向上させます。
- ALTER TABLE MOVEなどの表パーティション化メンテナンス操作のために使用禁止とマークされている索引を再作成します。
- データベースでの自動索引作成の構成、および自動索引作成操作に関連するレポートの生成のためのPL/SQL APIを提供します。

ノート:

- 自動索引は、ローカル B ツリー索引です。
- 自動索引は、パーティション表および非パーティション表に対して作成できます。
- 自動索引は、一時表に対しては作成できません。

親トピック: 自動索引の管理

21.7.2 自動索引作成の動作

この項では、自動索引作成がどのように機能するかを説明します。

自動索引作成プロセスは、バックグラウンドで15分ごとに実行され、次の操作を実行します。

1. 自動索引候補は、SQL文での表の列の使用方法に基づいて識別されます。表統計が最新であることを確認します。統計がない表は、自動索引付けの対象になりません。失効している統計を持つ表は、自動索引付けの対象になりません。
2. 索引候補は、最初の作成時には非表示かつ使用不可になります。アプリケーション・ワークロードには表示されません。非表示の自動索引は、アプリケーション・ワークロードのSQL文では使用できません。

   自動索引は単一列にも複数列にもなります。これらは、次のもの対して考慮されます。

   - 表の列(仮想列を含む)
   - パーティション化された表およびパーティション化されていない表
   - 選択した式(JSON式など)
3. ワークロードSQL文のサンプルが候補索引に対してテストされます。この検証フェーズでは、SQL文に対するパフォーマンス効果を測定できるように、一部またはすべての候補索引が作成されて有効になります。この検証ステップにおいて、すべての候補索引は非表示のままです。

   候補索引を使用してSQL文のパフォーマンスが向上しない場合、索引は非表示のままになります。
4. SQLパフォーマンスの向上のために見つかった有効な候補索引は、アプリケーション・ワークロードに表示されて使用可能になります。SQLパフォーマンスの改善につながらない候補索引は非表示に戻り、しばらくすると使用できなくなります。

   検証段階で、索引が有益であることはわかったものの個々のSQL文でパフォーマンスの低下が発生した場合、索引が表示されたときに、低下を防ぐためにSQL計画ベースラインが作成されます。
5. 使用禁止および未使用の有効な索引は、自動索引付けプロセスによって削除されます。

> ノート:
>
> デフォルトでは、未使用の自動索引は 373 日後に削除されます。データベース内の未使用の自動索引の保存期間は、DBMS_AUTO_INDEX.CONFIGURE プロシージャを使用して構成できます。

関連項目:

「Oracle Databaseでの自動索引作成の構成」

親トピック: 自動索引の管理

21.7.3 Oracle Databaseでの自動索引作成の構成

DBMS_AUTO_INDEX.CONFIGUREプロシージャを使用して、Oracleデータベースでの自動索引作成を構成できます。

次の例では、DBMS_AUTO_INDEX.CONFIGUREプロシージャを使用して指定できる構成設定のいくつかを説明します。

データベースでの自動索引作成の有効化および無効化

AUTO_INDEX_MODE構成設定を使用すると、データベースでの自動索引作成を有効または無効にできます。

次の文は、データベースでの自動索引作成を有効にし、新しい自動索引を可視索引として作成することで、SQL文で使用できるようにします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','IMPLEMENT');
```

次の文は、データベースでの自動索引作成を有効にしますが、新しい自動索引を不可視索引として作成することで、SQL文で使用できないようにします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','REPORT ONLY');
```

次の文は、データベースでの自動索引作成を無効にすることで、新しい自動索引が作成されないようにします(既存の自動索引は有効のまま)。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','OFF');
```

自動索引を使用できるスキーマの指定

AUTO_INDEX_SCHEMA構成設定を使用すると、自動索引を使用できるスキーマを指定できます。

ノート:

データベースで自動索引作成が有効になっている場合、そのデータベース内のすべてのスキーマはデフォルトで自動索引を使用できます。

次の文は、SHスキーマとHRスキーマを除外リストに追加し、SHスキーマとHRスキーマで自動索引を使用できないようにします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SCHEMA', 'SH', FALSE);
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SCHEMA', 'HR', FALSE);
```

次の文は、除外リストからHRスキーマを削除することで、HRスキーマが自動索引を使用できるようにします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SCHEMA', 'HR', NULL);
```

次の文は、除外リストからすべてのスキーマを削除することで、データベース内のすべてのスキーマが自動索引を使用できるようにします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SCHEMA', NULL, TRUE);
```

未使用の自動索引の保存期間の指定

AUTO_INDEX_RETENTION_FOR_AUTO構成設定を使用すると、データベース内の未使用の自動索引を保存する期間を指定できます。未使用の自動索引は、指定した保存期間の後に削除されます。

ノート:

デフォルトでは、未使用の自動索引は 373 日後に削除されます。

次の文は、未使用の自動索引の保存期間を90日間に設定します。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_RETENTION_FOR_AUTO', '90');
```

次の文は、自動索引の保存期間をデフォルト値である373日間にリセットします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_RETENTION_FOR_AUTO', NULL);
```

未使用の非自動索引の保存期間の指定

AUTO_INDEX_RETENTION_FOR_MANUAL構成設定を使用すると、データベース内の未使用の非自動索引(手動作成された索引)を保存する期間を指定できます。未使用の非自動索引は、指定した保存期間の後に削除されます。

ノート:

デフォルトでは、未使用の非自動索引が自動索引作成プロセスで削除されることはありません。

次の文は、未使用の非自動索引の保存期間を60日間に設定します。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_RETENTION_FOR_MANUAL', '60');
```

次の文は、未使用の非自動索引の保存期間をNULLに設定して、それらが自動索引作成プロセスによって削除されないようにします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_RETENTION_FOR_MANUAL', NULL);
```

自動索引作成ログの保存期間の指定

AUTO_INDEX_REPORT_RETENTION構成設定を使用すると、データベース内の自動索引作成ログを保存する期間を指定できます。自動索引作成ログは、指定した保存期間の後に削除されます。

ノート:

デフォルトでは、自動索引作成ログは 373 日後に削除されます。

次の文は、自動索引作成ログの保存期間を60日間に設定します。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_REPORT_RETENTION', '60');
```

次の文は、自動索引作成ログの保存期間をデフォルト値である373日間にリセットします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_REPORT_RETENTION', NULL);
```

ノート:

自動索引作成レポートは、自動索引作成ログに基づいて生成されます。したがって、AUTO_INDEX_REPORT_RETENTION 構成設定を使用して指定された自動索引作成ログの保存期間を超える期間については、自動索引作成レポートを生成できません。

自動索引を格納する表領域の指定

AUTO_INDEX_DEFAULT_TABLESPACE構成設定を使用すると、自動索引を格納する表領域を指定できます。Oracle所有の表領域(SYSAUXなど)をデフォルトの表領域として指定することはできません。

ノート:

デフォルトでは、データベース作成時に指定された永続表領域が、自動索引の格納に使用されます。

次の文は、自動索引を格納するためにTBS_AUTO表領域を指定します。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_DEFAULT_TABLESPACE', 'TBS_AUTO');
```

自動索引用に割り当てる表領域の割合の指定

AUTO_INDEX_SPACE_BUDGET構成設定を使用すると、自動索引用に割り当てる表領域の割合を指定できます。この構成設定は、自動索引の格納に使用される表領域が、データベース作成時に指定されたデフォルトの永続表領域である場合(つまり、AUTO_INDEX_DEFAULT_TABLESPACE構成設定に値が指定されていない場合)にのみ指定できます。

次の文は、自動索引用に表領域の5パーセントを割り当てます。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_SPACE_BUDGET', '5');
```

自動索引の拡張索引圧縮の構成

AUTO_INDEX_COMPRESSION構成設定を使用して、自動索引で拡張索引圧縮を使用する必要があるかどうかを指定できます。拡張索引圧縮は、Oracle Advanced Compressionオプションの一部です。

次の例では、自動索引の作成時に拡張索引圧縮を有効にします。

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_COMPRESSION','ON');
```

関連項目:

Oracle Advanced Compressionオプションの詳細は、『Oracle Databaseライセンス情報ユーザー・マニュアル』を参照してください。

関連項目:

DBMS_AUTO_INDEX.CONFIGUREプロシージャを使用して指定できる、自動索引作成に関連する構成設定の完全なリストは、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください。

親トピック: 自動索引の管理

21.7.4 自動索引作成レポートの生成

Oracleデータベースでの自動索引作成操作に関連するレポートを生成するには、DBMS_AUTO_INDEXパッケージのREPORT_ACTIVITYファンクションとREPORT_LAST_ACTIVITYファンクションを使用します。

関連項目:

DBMS_AUTO_INDEXパッケージのREPORT_ACTIVITYファンクションとREPORT_LAST_ACTIVITYファンクションの構文については、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照してください。

特定の期間の自動索引作成操作のレポートの生成

次の例では、過去24時間の自動索引作成操作について一般情報を含むレポートを生成します。デフォルトでは、このレポートはプレーン・テキスト形式で生成されます。

```
declare
  report clob := null;
begin
  report := DBMS_AUTO_INDEX.REPORT_ACTIVITY();
end;
```

次の例では、2018年11月の自動索引作成操作について基本情報を含むレポートを生成します。このレポートはHTML形式で生成され、自動索引作成操作のサマリーのみを含んでいます。

```
declare
  report clob := null;
begin
  report := DBMS_AUTO_INDEX.REPORT_ACTIVITY(
              activity_start => TO_TIMESTAMP('2018-11-01', 'YYYY-MM-DD'),
              activity_end   => TO_TIMESTAMP('2018-12-01', 'YYYY-MM-DD'),
```

```
              type           => 'HTML',
              section        => 'SUMMARY',
              level          => 'BASIC');
end;
```

最後の自動索引作成操作のレポートの生成

次の例では、最後の自動索引作成操作について一般情報を含むレポートを生成します。デフォルトでは、このレポートはプレーン・テキスト形式で生成されます。

```
declare
  report clob := null;
begin
  report := DBMS_AUTO_INDEX.REPORT_LAST_ACTIVITY();
end;
```

次の例では、最後の自動索引作成操作について基本情報を含むレポートを生成します。このレポートには、最後の自動索引作成操作のサマリー、索引詳細およびエラー情報が表示されます。このレポートはHTML形式で生成されます。

```
declare
  report clob := null;
begin
  report := DBMS_AUTO_INDEX.REPORT_LAST_ACTIVITY(
              type    => 'HTML',
              section => 'SUMMARY +INDEX_DETAILS +ERRORS',
              level   => 'BASIC');
end;
```

親トピック: 自動索引の管理

21.7.5 自動索引作成情報を含むビュー

一連のデータ・ディクショナリ・ビューを問い合せて、Oracleデータベースでの自動索引に関する情報を取得できます。

次のビューには、自動索引作成の構成設定およびOracleデータベースで作成された自動索引に関する情報が示されます。

ビュー	説明
DBA_AUTO_INDEX_CONFIG	自動索引作成の現在の構成設定が示されます。
DBA_INDEXES ALL_INDEXES USER_INDEXES	これらのビューの AUTO 列は、索引が自動索引であるかどうか(YES|NO)を示します。

関連項目:

これらのビューの完全な説明は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 自動索引の管理

21.8 索引のデータ・ディクショナリ・ビュー

索引に関する情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

次のビューには、索引に関する情報が示されます。

ビュー	説明
DBA_INDEXES ALL_INDEXES USER_INDEXES	DBA ビューには、データベース内のすべての表の索引に関する情報が示されます。ALL ビューには、ユーザーがアクセスできるすべての表の索引に関する情報が示されます。USER ビューは、ユーザーが所有する索引のみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_IND_COLUMNS ALL_IND_COLUMNS USER_IND_COLUMNS	これらのビューには、表の索引の列に関する情報が示されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_IND_PARTITIONS ALL_IND_PARTITIONSALL_IND_PARTITIONS USER_IND_PARTITIONS	これらのビューには、各索引パーティションについて、パーティションの詳細、パーティションの記憶域パラメータ、および DBMS_STATS パッケージによって生成された様々なパーティション統計が示されます。
DBA_IND_EXPRESSIONS ALL_IND_EXPRESSIONS USER_IND_EXPRESSIONS	これらのビューには、表のファンクション索引の式に関する情報が示されます。
DBA_IND_STATISTICS ALL_IND_STATISTICS USER_IND_STATISTICS	これらのビューには、索引のオプティマイザ統計に関する情報が示されます。
INDEX_STATS INDEX_HISTOGRAM	これらのビューには、最後の ANALYZE INDEX...VALIDATE STRUCTURE 文に関する情報が示されます。
USER_OBJECT_USAGE	このビューには、ALTER INDEX...MONITORING USAGE 文によって生成された索引使用情報が示されます。

関連項目:

これらのビューの完全な説明は、Oracle Databaseリファレンスを参照してください

親トピック: 索引の管理

22 クラスタの管理

クラスタを使用してパフォーマンスを改善し、必要なディスク領域を削減できます。

- クラスタについて
  クラスタは、表データを格納するために選択可能なオプションの方法を提供します。クラスタは、同じデータ・ブロックを共有する表のグループで構成されています。表をグループ化する理由は、各表が共通の列を共有しており、一緒に使用されるケースが多いためです。
- クラスタを管理するためのガイドライン
  クラスタを管理するためにガイドラインに従うことができます。
- クラスタおよびそれらを使用するオブジェクトの作成
  クラスタを作成するには、CREATE CLUSTER文を使用します。クラスタ表を作成するには、CLUSTER句を指定したCREATE TABLE文を使用します。クラスタ索引を作成するには、CLUSTER句を指定したCREATE INDEX文を使用します。
- クラスタおよびそれらを使用するオブジェクトの変更
  クラスタを変更して、その物理属性、サイズおよびデフォルトの並列度を変更できます。
- クラスタおよびそれらを使用するオブジェクトの削除
  クラスタはDROP CLUSTER文を使用して削除します。クラスタ表はDROP TABLE文を使用して削除します。クラスタ索引はDROP INDEX文を使用して削除します。
- クラスタのデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューのセットに対してクラスタに関する情報を問い合せることができます。

親トピック: スキーマ・オブジェクト

22.1 クラスタについて

クラスタは、表データを格納するために選択可能なオプションの方法を提供します。クラスタは、同じデータ・ブロックを共有する表のグループで構成されています。表をグループ化する理由は、各表が共通の列を共有しており、一緒に使用されるケースが多いためです。

たとえば、emp表とdept表がdeptno列を共有しているとします。emp表およびdept表をクラスタ化する場合(図22-1を参照)、Oracle Databaseでは、emp表およびdept表の各部門の行はすべて、物理的に同じデータ・ブロックに格納されます。

クラスタは、異なる表の関連する行を同じデータ・ブロックに格納します。そのため、クラスタを正しく使用することには、主に次のような利点があります。

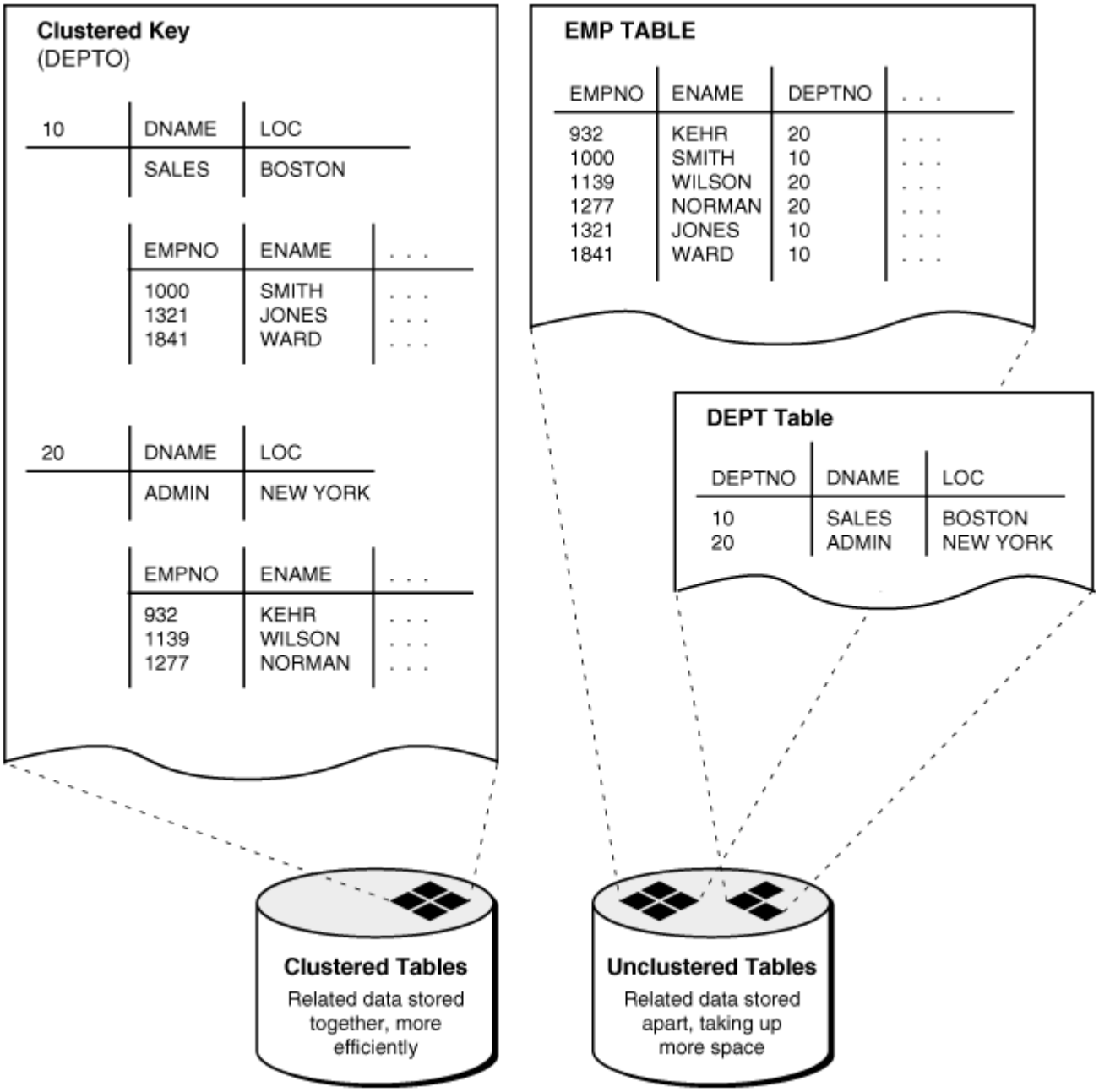
- クラスタ化表の結合によって、ディスクI/Oが減少し、アクセス時間が改善されます。
- クラスタ・キーは、クラスタ化表が共有する列または列のグループです。最初にクラスタを作成するときに、クラスタ・キー列を指定します。その後、そのクラスタに追加する表を作成するたびに同じ列を指定します。各クラスタ・キー値は、その値が含まれている異なる表行数に関係なく、クラスタとクラスタ索引のそれぞれに1度しか格納されません。

  したがって、関連する表と索引データをクラスタに格納する方が、クラスタ化されていない表形式の場合よりも、必要になる記憶域が削減される場合があります。たとえば、図22-1では、emp表とdept表の両方にある同じ値を含んでいる多数の行に対して、各クラスタ・キー(各deptno)が1回しか格納されていないことに注意してください。

クラスタを作成した後、クラスタ内に表を作成できます。ただし、クラスタ化表に行を挿入する前に、クラスタ索引を作成しておく必要があります。クラスタを使用しても、クラスタ化表での索引の追加作成に影響はなく、通常どおり作成および削除できます。

別々にアクセスされることの多い表には、クラスタを使用しないでください。

図22-1 クラスタ化表データ



関連項目:

- もう1つのタイプのクラスタ(ハッシュ・クラスタ)の詳細は、「ハッシュ・クラスタの管理」を参照してください
- この章のタスクを実行する前に、「スキーマ・オブジェクトの領域の管理」を一読されることをお薦めします

親トピック: クラスタの管理

22.2 クラスタを管理するためのガイドライン

クラスタを管理するためにガイドラインに従うことができます。

- クラスタに適した表の選択
  主に問合せであり、まとめて問い合されることの多い表にクラスタを使用します。
- クラスタ・キーに適した列の選択

クラスタ・キー列の選択には注意が必要です。表を結合する問合せで複数列を使用する場合、クラスタ・キーはコンポジット・キーにします。一般に、適切なクラスタ索引を表す特性は、適切な索引を表す特性と同じです。

- 平均クラスタ・キーとその対応行が必要とする領域の指定
  CREATE CLUSTER文には、平均クラスタ・キーとその対応行が必要とするバイト数の見積りであるオプション句SIZEがあります。
- 各クラスタとクラスタ索引の行の位置の指定
  新しいクラスタまたは索引を格納する表領域を識別するには、CREATE CLUSTER/INDEX文に必ずTABLESPACE句を指定します。
- クラスタ・サイズの見積りと記憶域パラメータの設定
  クラスタを作成する前に、クラスタ・サイズを見積り、クラスタのデータ・セグメントの記憶域パラメータを設定します。

関連項目:

- クラスタの詳細は、『Oracle Database概要』を参照してください。
- どのようなときにクラスタを使用するかについてのガイドラインは、『Oracle Database SQLチューニング・ガイド』を参照してください。

親トピック: クラスタの管理

22.2.1 クラスタに適した表の選択

主に問合せであり、まとめて問い合されることの多い表にクラスタを使用します。

次のような条件が成り立つ場合は、表にクラスタを使用します。

- 表が主に問合せに使用される場合、つまり、主に挿入または更新に使用されない場合。
- 複数の表のレコードを頻繁に問い合せたり、結合する場合。

親トピック: クラスタを管理するためのガイドライン

22.2.2 クラスタ・キーに適した列の選択

クラスタ・キー列の選択には注意が必要です。表を結合する問合せで複数列を使用する場合、クラスタ・キーはコンポジット・キーにします。一般に、適切なクラスタ索引を表す特性は、適切な索引を表す特性と同じです。

適切な索引を表す特性の詳細は、「索引を管理するためのガイドライン」を参照してください。

適切なクラスタ・キーの条件は、1つのキー値に対応している行のグループで1つのデータ・ブロックがほぼ一杯になるような、一意の値を持つことです。クラスタ・キー値ごとの行が少なすぎると、領域を浪費し、結果的にパフォーマンスが低下します。共通の値を共有する行がわずかしかないクラスタ・キーは、クラスタを作成するときにSIZEに小さい値を指定しないかぎり、ブロック内の領域を浪費する可能性があります(「平均クラスタ・キーとその対応行が必要とする領域の指定」を参照してください)。

クラスタ・キー値ごとの行が多すぎると、そのキーを持つ行を見つけるために余分な検索が起こる可能性があります。クラスタ・キーの値があまりにも一般的な場合(たとえば、maleとfemaleといった性別など)は、過度の検索が発生し、クラスタ化していない場合よりパフォーマンスが低下するおそれがあります。

クラスタ索引は一意にできません。また、LONG型として定義されている列を含むことはできません。

親トピック: クラスタを管理するためのガイドライン

22.2.3 平均クラスタ・キーとその対応行が必要とする領域の指定

CREATE CLUSTER文には、平均クラスタ・キーとその対応行が必要とするバイト数の見積りであるオプション句SIZEがあります。

SIZEパラメータは、次のタスクを実行する際にデータベースによって使用されます。

- クラスタ化したデータ・ブロック内に収めることのできるクラスタ・キー(および対応する行)の数を見積る場合。
- クラスタ化したデータ・ブロックに配置するクラスタ・キーの数を制限する場合。これにより、クラスタ内のキーの格納効率が最大になります。

SIZEは、クラスタ・キーが使用できる領域を制限しません。たとえば、2つのクラスタ・キーが1つのデータ・ブロックに収まるようにSIZEが設定された場合、どちらのクラスタ・キーでも、その利用可能なデータ・ブロック領域をいくらでも使用できます。

デフォルトでは、データベースは1つのクラスタ・キーとそれに対応する行をそのクラスタのデータ・セグメントの1データ・ブロックに格納します。ブロック・サイズはオペレーティング・システムによって異なることがありますが、クラスタ化表が異なるシステム上にある他のデータベースにインポートされるときにも、ブロック当たり1つのキーというルールは守られます。

クラスタ・キー値に対応するすべての行を1つのブロック内に収めることができない場合は、ブロックをまとめて連鎖することにより、特定のキーを持つすべての値へのアクセスの高速化が図られます。クラスタ索引は、クラスタ・キー値と対応する行をそれぞれに含むブロックの連鎖の始点を示します。複数のキーが1つのブロックに収まるようなクラスタのSIZEである場合、ブロックは複数の連鎖に所属することがあります。

親トピック: クラスタを管理するためのガイドライン

22.2.4 各クラスタとクラスタ索引の行の位置の指定

新しいクラスタまたは索引を格納する表領域を識別するには、CREATE CLUSTER/INDEX文に必ずTABLESPACE句を指定します。

適切な権限と表領域割当て制限を持つユーザーであれば、現在オンライン状態の表領域内に新しいクラスタおよび関連するクラスタ索引を作成できます。

クラスタとそのクラスタ索引は、異なる表領域内に作成できます。実際、クラスタとその索引を、それぞれ異なる記憶デバイス上に格納された異なる表領域内に作成すると、ディスク競合を最小限に抑えて、表データと索引データを同時に検索できます。

親トピック: クラスタを管理するためのガイドライン

22.2.5 クラスタ・サイズの見積りと記憶域パラメータの設定

クラスタを作成する前に、クラスタ・サイズを見積り、クラスタのデータ・セグメントの記憶域パラメータを設定します。

クラスタを作成する前にクラスタ・サイズを見積る利点は、次のとおりです。

- クラスタの見積りサイズの合計と、索引およびREDOログ・ファイルの見積りを使用して、作成するデータベースを格納するために必要なディスク容量を決定できます。この見積りを利用して適切なハードウェアを購入できます。
- 個々のクラスタの見積りサイズを使用することで、クラスタが使用するディスク領域をより適切に管理できます。クラスタを作成するときに、適切な記憶域パラメータを設定し、そのクラスタを使用するアプリケーションのI/Oパフォーマンスを改善できます。

表をクラスタに配置する個別のCREATE文またはALTER文ではなく、CREATE CLUSTER句またはALTER CLUSTER文のSTORAGE句を使用してクラスタのデータ・セグメントの記憶域パラメータを設定します。クラスタ化された表の作成時または変更

時に指定した記憶域パラメータは無視されます。クラスタに設定された記憶域パラメータは、表の記憶域パラメータを上書きします。

親トピック: クラスタを管理するためのガイドライン

22.3 クラスタの作成およびクラスタを使用するオブジェクト

クラスタを作成するには、CREATE CLUSTER文を使用します。クラスタ表を作成するには、CLUSTER句を指定したCREATE TABLE文を使用します。クラスタ索引を作成するには、CLUSTER句を指定したCREATE INDEX文を使用します。

- クラスタの作成
  クラスタを作成するには、CREATE CLUSTER文を使用します。
- クラスタ化表の作成
  クラスタに表を作成するには、CLUSTER句を指定したCREATE TABLE文を使用します。
- クラスタ索引の作成
  クラスタ索引は、クラスタ化表に行を挿入する前に作成する必要があります。

親トピック: クラスタの管理

22.3.1 クラスタの作成

クラスタを作成するには、CREATE CLUSTER文を使用します。

自分のスキーマにクラスタを作成するには、CREATE CLUSTERシステム権限とそのクラスタを格納する表領域に対する割当て制限を持っているか、またはUNLIMITED TABLESPACEシステム権限を持っている必要があります。

別のユーザーのスキーマにクラスタを作成するには、CREATE ANY CLUSTERシステム権限が必要です。さらに、所有者はそのクラスタを格納する表領域に対する割当て制限を持っているか、またはUNLIMITED TABLESPACEシステム権限を持っている必要があります。

次の文は、deptno列によってクラスタ化された、emp表とdept表を格納するクラスタemp_deptを作成します。

```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
   SIZE 600
   TABLESPACE users
   STORAGE (INITIAL 200K
      NEXT 300K
      MINEXTENTS 2
      PCTINCREASE 33);
```

この例のようにINDEXキーワードを指定しない場合は、デフォルトで索引クラスタが作成されます。また、ハッシュ・パラメータ(HASHKEYS、HASH ISまたはSINGLE TABLE HASHKEYS)を指定すると、ハッシュ・クラスタを作成できます。ハッシュ・クラスタについては、「ハッシュ・クラスタの管理」を参照してください。

親トピック: クラスタの作成およびクラスタを使用するオブジェクト

22.3.2 クラスタ化表の作成

クラスタに表を作成するには、CLUSTER句を指定したCREATE TABLE文を使用します。

クラスタに表を作成するには、CREATE TABLEシステム権限またはCREATE ANY TABLEシステム権限のどちらかが必要です。なお、クラスタ内に表を作成するために、表領域割当て制限またはUNLIMITED TABLESPACEシステム権限は必要ありません。

たとえば、次の文を使用すると、emp_deptクラスタ内にemp表とdept表を作成できます。

```
CREATE TABLE emp (
   empno NUMBER(5) PRIMARY KEY,
   ename VARCHAR2(15) NOT NULL,
   . . .
   deptno NUMBER(3) REFERENCES dept)
   CLUSTER emp_dept (deptno);
CREATE TABLE dept (
   deptno NUMBER(3) PRIMARY KEY, . . . )
   CLUSTER emp_dept (deptno);
```

ノート:

クラスタ化表のスキーマは、CREATE TABLE 文で指定できます。クラスタ化表は、そのクラスタを含むスキーマとは異なるスキーマに配置できます。また、列名は一致しなくてもかまいませんが、その構造は同じである必要があります。

関連項目:

クラスタ表を作成するCREATE TABLE文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: クラスタの作成およびクラスタを使用するオブジェクト

22.3.3 クラスタ索引の作成

クラスタ索引は、クラスタ化表に行を挿入する前に作成する必要があります。

クラスタ索引を作成するには、次の条件のいずれかが成り立つ必要があります。

- スキーマにクラスタが含まれている。
- CREATE ANY INDEXシステム権限を持っている。

どちらの場合も、クラスタ索引を格納する表領域に対する割当て制限またはUNLIMITED TABLESPACEシステム権限が必要です。

次の文は、emp_deptクラスタに対するクラスタ索引を作成します。

```
CREATE INDEX emp_dept_index
   ON CLUSTER emp_dept
   TABLESPACE users
   STORAGE (INITIAL 50K
      NEXT 50K
      MINEXTENTS 2
      MAXEXTENTS 10
      PCTINCREASE 33);
```

クラスタ索引句(ON CLUSTER)では、クラスタ索引を作成するクラスタemp_deptを識別します。この文では、クラスタとクラスタ索引の記憶域設定も明示的に指定しています。

関連項目:

クラスタ索引を作成するCREATE INDEX文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: クラスタの作成およびクラスタを使用するオブジェクト

22.4 クラスタの変更およびクラスタを使用するオブジェクト

クラスタを変更して、その物理属性、サイズおよびデフォルトの並列度を変更できます。

- クラスタの変更
  ALTER CLUSTER文を使用してクラスタを変更します。
- クラスタ化表の変更
  ALTER TABLE文を使用してクラスタ化表を変更できますが、クラスタ化表の一部のパラメータはALTER TABLE文で設定できません。
- クラスタ索引の変更
  クラスタ索引は、他の索引と同じように変更できます。

親トピック: クラスタの管理

22.4.1 クラスタの変更

ALTER CLUSTER文を使用してクラスタを変更します。

クラスタを変更するには、そのクラスタが自分のスキーマに含まれているか、またはALTER ANY CLUSTERシステム権限を持っている必要があります。既存のクラスタを変更することにより、次の設定を変更できます。

- 物理属性(INITRANSおよび記憶域特性)
- クラスタ・キー値のすべての行を格納するために必要な平均使用可能空き領域(SIZE)
- デフォルトの並列度

また、クラスタに新しいエクステントを明示的に割り当てたり、クラスタの最後にある未使用のエクステントの割当てを解除できます。データベースは、必要に応じてクラスタのデータ・セグメントに追加のエクステントを動的に割り当てます。ただし、状況によっては、クラスタに追加のエクステントを明示的に割り当てることもできます。たとえば、Real Application Clustersを使用している場合、クラスタのエクステントを特定のインスタンスに明示的に割り当てることができます。クラスタに新しいエクステントを割り当てるには、ALLOCATE EXTENT句を指定したALTER CLUSTER文を使用します。

クラスタ・サイズ・パラメータ(SIZE)を変更すると、そのクラスタにすでに割り当てられているブロックと今後割り当てられるブロックを含め、そのクラスタが使用するすべてのデータ・ブロックに対して新しい設定が適用されます。すでに表に割り当てられているブロックは、即時ではなく、必要なときに再編成されます。

クラスタのトランザクション・エントリ設定INITRANSを変更すると、INITRANSの新しい設定は、その後クラスタに割り当てられるブロックにのみ適用されます。

記憶域パラメータINITIALとMINEXTENTSは変更できません。他の記憶域パラメータの新しい設定はすべて、その後にクラスタに割り当てられるエクステントにのみ影響します。

クラスタを変更するには、ALTER CLUSTER文を使用します。

関連項目:

ALTER CLUSTER文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: クラスタの変更およびクラスタを使用するオブジェクト

22.4.2 クラスタ化表の変更

ALTER TABLE文を使用してクラスタ化表を変更できますが、クラスタ化表の一部のパラメータはALTER TABLE文で設定できません。

ただし、ALTER TABLE文でクラスタ化表に対してデータ・ブロック領域パラメータ、トランザクション・エントリ・パラメータまたは記憶域パラメータを設定すると、エラー・メッセージ(ORA-01771、クラスタ表に対するオプションが無効です。)が生成されます。データベースは、クラスタのパラメータをすべてのクラスタ化表に対して使用します。そのため、ALTER TABLE文は、クラスタ化表の列の追加または変更、クラスタ・キー以外の列の削除、整合性制約またはトリガーの追加、削除、有効化、無効化のみに使用できます。表の変更の詳細は、「表の変更」を参照してください。

関連項目:

ALTER TABLE文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: クラスタの変更およびクラスタを使用するオブジェクト

22.4.3 クラスタ索引の変更

クラスタ索引は、他の索引と同じように変更できます。

「索引の変更」を参照してください。

ノート:

クラスタ索引のサイズを見積るときには、索引が実際の行ではなく各クラスタ・キーに付いていることに注意してください。したがって、各キーは索引内に 1 度しか現れません。

親トピック: クラスタの変更およびクラスタを使用するオブジェクト

22.5 クラスタの削除およびクラスタを使用するオブジェクト

クラスタはDROP CLUSTER文を使用して削除します。クラスタ表はDROP TABLE文を使用して削除します。クラスタ索引はDROP INDEX文を使用して削除します。

- クラスタの削除
  クラスタはDROP CLUSTER文を使用して削除できます。
- クラスタ化表の削除
  クラスタ化表は、そのクラスタ、他のクラスタ化表またはクラスタ索引に影響を及ぼすことなく、個別に削除できます。クラスタ化表は、非クラスタ化表を削除する場合と同じように、DROP TABLE文を使用して削除します。
- クラスタ索引の削除
  クラスタ索引は、そのクラスタまたはクラスタ化表に影響を及ぼすことなく削除できます。ただし、クラスタ索引がなければクラスタ化表は使用できないため、クラスタへのアクセスを許可するにはクラスタ索引を再作成する必要があります。

親トピック: クラスタの管理

22.5.1 クラスタの削除

クラスタはDROP CLUSTER文を使用して削除できます。

クラスタ内の表が不要になった場合は、そのクラスタを削除できます。クラスタを削除すると、そのクラスタ内の表および対応するクラスタ索引も削除されます。クラスタのデータ・セグメントとクラスタ索引の索引セグメントの両方に属するすべてのエクステントは、それらを含んでいる表領域に戻され、その表領域内の他のセグメントで使用可能になります。

表を含まないクラスタとそのクラスタ索引を削除するには、DROP CLUSTER文を使用します。たとえば、次の文は空のクラスタemp_deptを削除します。

```
DROP CLUSTER emp_dept;
```

クラスタに1つ以上のクラスタ化表が含まれており、その表も同様に削除する場合は、次のように、DROP CLUSTER文にINCLUDING TABLES句を追加します。

```
DROP CLUSTER emp_dept INCLUDING TABLES;
```

INCLUDING TABLES句を指定していない場合に、クラスタに表が含まれていると、エラーが返されます。

クラスタ内の1つ以上の表が、クラスタ外の表のFOREIGN KEY制約によって参照される主キーまたは一意のキーを含んでいる場合、依存するFOREIGN KEY制約も削除しないと、そのクラスタを削除できません。これは、次の例に示すように、DROP CLUSTER文のCASCADE CONSTRAINTS句を使用すると容易に実行できます。

```
DROP CLUSTER emp_dept INCLUDING TABLES CASCADE CONSTRAINTS;
```

制約が存在している場合に、CASCADE CONSTRAINTS句を使用しないと、データベースはエラーを返します。

関連項目:

DROP CLUSTER文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: クラスタの削除およびクラスタを使用するオブジェクト

22.5.2 クラスタ化表の削除

クラスタ化表は、そのクラスタ、他のクラスタ化表またはクラスタ索引に影響を及ぼすことなく、個別に削除できます。クラスタ化表は、非クラスタ化表を削除する場合と同じように、DROP TABLE文を使用して削除します。

クラスタを削除するには、そのクラスタが自分のスキーマに含まれているか、またはDROP ANY CLUSTERシステム権限を持っている必要があります。クラスタ化表をそのクラスタの所有者が所有していなくても、その表を含むクラスタを削除するために特別な権限は必要ありません。

「表の列の削除」を参照してください。

> ノート:
>
> 単一の表をクラスタから削除するときは、表の各行が個別に削除されます。クラスタ全体を最も効率よく削除するには、INCLUDING TABLES 句を指定した DROP CLUSTER 文を使用して、そのクラスタを表も含めて削除します。クラスタの残りの部分をそのままにしておく場合のみ、(DROP TABLE 文を使用して)クラスタから表を個別に削除してください。

親トピック: クラスタの削除およびクラスタを使用するオブジェクト

22.5.3 クラスタ索引の削除

クラスタ索引は、そのクラスタまたはクラスタ化表に影響を及ぼすことなく削除できます。ただし、クラスタ索引がなければクラスタ化表は使用できないため、クラスタへのアクセスを許可するにはクラスタ索引を再作成する必要があります。

クラスタ索引は、断片化したクラスタ索引を再作成する過程で削除される場合があります。

ノート:

ハッシュ・クラスタ索引は削除できません。

関連項目:

「索引の削除」

親トピック: クラスタの削除およびクラスタを使用するオブジェクト

22.6 クラスタのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューのセットに対してクラスタに関する情報を問い合せることができます。

次のビューには、クラスタに関する情報が表示されます。

ビュー	説明
DBA_CLUSTERS ALL_CLUSTERS USER_CLUSTERS	DBA ビューには、データベース内のすべてのクラスタが表示されます。ALL ビューには、ユーザーがアクセス可能なすべてのクラスタが表示されます。USER ビューは、ユーザーが所有するクラスタのみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_CLU_COLUMNS USER_CLU_COLUMNS	これらのビューでは、表の列とクラスタの列がマップされています。

関連項目:

これらのビューの完全な説明は、『Oracle Databaseリファレンス』を参照してください。

親トピック: クラスタの管理

23 ハッシュ・クラスタの管理

ハッシュ・クラスタは、データ検索のパフォーマンスを向上できます。

- ハッシュ・クラスタについて
  ハッシュ・クラスタへの表の格納は、データ検索のパフォーマンスを改善するためのオプションの方法です。ハッシュ・クラスタは、索引付きの非クラスタ化表または索引クラスタの代替手段を提供します。
- ハッシュ・クラスタを使用する場合
  ハッシングが最も有用である状況とまったく利点がない状況を対比することにより、どのような場合にハッシュ・クラスタを使用するかを決定できます。ハッシングではなく索引を使用する場合は、表を個別に格納するのか、またはクラスタの一部として格納するのかを考慮してください。
- 異なるタイプのハッシュ・クラスタの作成
  異なるタイプのハッシュ・クラスタを作成するために、HASHKEYS句を含むCREATE CLUSTER文を使用できます。
- ハッシュ・クラスタの変更
  ハッシュ・クラスタは、ALTER CLUSTER文を使用して変更できます。
- ハッシュ・クラスタの削除
  ハッシュ・クラスタは、DROP CLUSTER文を使用して削除できます。
- ハッシュ・クラスタ・データ・ディクショナリ・ビュー
  ハッシュ・クラスタに関する情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

親トピック: スキーマ・オブジェクト

23.1 ハッシュ・クラスタについて

ハッシュ・クラスタに表を格納することは、データ検索のパフォーマンスを改善するための1つの選択肢です。ハッシュ・クラスタは、索引付きの非クラスタ化表または索引クラスタの代替手段を提供します。

索引付きの表または索引クラスタでは、Oracle Databaseは、表内の行の位置を特定するために、別個の索引に格納しているキー値を使用します。ハッシングを使用するには、ハッシュ・クラスタを作成し、そこに表をロードします。表の行は物理的にはハッシュ・クラスタに格納され、ハッシュ関数の結果に従って検索されます。

Oracle Databaseはハッシュ関数を使用して、特定のクラスタ・キー値に基づく、ハッシュ値と呼ばれる数値の分布を生成します。ハッシュ・クラスタのキーは、索引クラスタのキーと同じように単一列キーでもコンポジット・キー(複数列キー)でもかまいません。ハッシュ・クラスタ内で行を検索または格納する場合、データベースは行のクラスタ・キー値にハッシュ関数を適用します。結果として生成されるハッシュ値はクラスタ内のデータ・ブロックに対応しており、データベースは、発行された文のかわりにそのデータ・ブロックに対して読取りまたは書込みを行います。

索引付きの表または索引クラスタ内の行を検索または格納するためには、少なくとも2回(通常はそれ以上)のI/Oを実行する必要があります。

- 1回以上のI/Oによる、索引内でのキー値の検索または格納
- 別のI/Oによる、表またはクラスタ内での行の読取りまたは書込み

一方、ハッシュ・クラスタでは、ハッシュ関数を使用して行の位置が特定されるので、I/Oは必要ありません。結果として、ハッシュ・クラスタ内の行の読込みや書込みに必要とされるのは最小限のI/O操作のみになります。

関連項目:

この章のタスクを実行する前に、「スキーマ・オブジェクトの領域の管理」を一読されることをお薦めします。

親トピック: ハッシュ・クラスタの管理

23.2 ハッシュ・クラスタを使用する場合

ハッシングが最も有用である状況とまったく利点がない状況を対比することにより、どのような場合にハッシュ・クラスタを使用するかを決定できます。ハッシングではなく索引を使用する場合は、表を個別に格納するのか、またはクラスタの一部として格納するのかを考慮してください。

ノート:

ハッシングを使用する場合でも、クラスタ・キーを含む表のどの列にも異なる索引を設定できます。

- ハッシングが有効な状況
  ハッシングは、ほとんどの問合せがクラスタ・キーの等価問合せであり、ハッシュ・クラスタ内の表のサイズが主に静的である場合に有効です。
- ハッシングが有利ではない状況
  ハッシングは、いくつかの状況では有利ではありません。

親トピック: ハッシュ・クラスタの管理

23.2.1 ハッシングが有効な状況

ハッシングは、ほとんどの問合せがクラスタ・キーの等価問合せであり、ハッシュ・クラスタ内の表のサイズが主に静的である場合に有効です。

ハッシングは、次のような状況で有効です。

- ほとんどの問合せが次のようなクラスタ・キーとの等式を含んでいる場合。

  ```
  SELECT ... WHERE cluster_key = ...;
  ```

  このような場合、等価条件内のクラスタ・キーがハッシュされ、対応するハッシュ・キーが通常1回の読込みで検索されます。それに対して、索引付きの表では、最初にキー値を索引内で検索する必要があり(通常複数回の読込み)、その後で行が表から読み込まれます(別の読込み)。

- ハッシュ・クラスタ内の表のサイズが最初から固定されていて、行数とそのクラスタ内の表が必要とする領域を決定できる場合。ハッシュ・クラスタ内の表でそのクラスタの初期割当てより多くの領域が必要な場合、オーバーフロー・ブロックが必要になるためにパフォーマンスがかなり低下するおそれがあります。

親トピック: ハッシュ・クラスタを使用する場合

23.2.2 ハッシングが不利な状況

ハッシングは、いくつかの状況では有利ではありません。

ハッシングは次のような状況では有効ではありません。

- ほとんどの問合せがクラスタ・キー値全体にわたって行を検索する場合。たとえば、全表スキャンや次のような問合せで

は、ハッシュ関数は特定のハッシュ・キーの位置を決定するために使用できません。そのかわりに、全表スキャンと同等の機能を実行して問合せの行をフェッチする必要があります。

```
SELECT . . . WHERE cluster_key < . . . ;
```

索引では、キー値はその索引内で順序付けられており、問合せのWHERE句を満たすクラスタ・キー値を、比較的少ないI/Oで見つけることができます。

- 表が固定でなく、継続的に拡大する場合。表が無制限に拡大する場合、表(そのクラスタ)の存続期間にわたって必要な領域を事前に定義することはできません。
- アプリケーションが頻繁に表の全表スキャンを実行し、その表内でデータが散在している場合。この状況でハッシングを使用すると、全表スキャンの処理時間が長くなります。
- 最終的にハッシュ・クラスタが必要とする領域を事前に割り当てることができない場合。

親トピック: ハッシュ・クラスタを使用する場合

23.3 様々なタイプのハッシュ・クラスタの作成

異なるタイプのハッシュ・クラスタを作成するために、HASHKEYS句を含むCREATE CLUSTER文を使用できます。

- ハッシュ・クラスタの作成
  ハッシュ・クラスタを作成するには、HASHKEYS句を指定したCREATE CLUSTER文を使用します。
- ソートされたハッシュ・クラスタの作成
  ソートされたハッシュ・クラスタは、ソートされた順序で効率的に値を返すことができるように、ハッシュ関数の各値に対応する行を格納します。常にソート順にデータを使用するアプリケーションの場合、ソートされたハッシュ・クラスタを使用することによって論理I/Oを最小化でき、これにより、より迅速にデータを取得できます。
- 単一表ハッシュ・クラスタの作成
  表の行への高速アクセスを提供する単一表ハッシュ・クラスタを作成できます。ただし、この表はハッシュ・クラスタ内の唯一の表にする必要があります。
- ハッシュ・クラスタ内の領域使用の制御
  ハッシュ・クラスタを作成するときは、パフォーマンスと使用領域が最適になるように、クラスタ・キーを正しく選択し、HASH IS、SIZEおよびHASHKEYSの各パラメータを設定することが重要です。次に示すガイドラインでは、これらのパラメータを設定する方法について説明します。
- ハッシュ・クラスタに必要なサイズの見積り
  索引クラスタと同様に、ハッシュ・クラスタ内のデータに必要な記憶領域を推定することが重要です。

親トピック: ハッシュ・クラスタの管理

23.3.1 ハッシュ・クラスタの作成

ハッシュ・クラスタを作成するには、HASHKEYS句を指定したCREATE CLUSTER文を使用します。

次の文は、trialno列(クラスタ・キー)でクラスタ化されたクラスタをtrial_clusterという名前で作成します。

```
CREATE CLUSTER trial_cluster ( trialno NUMBER(5,0) )
    TABLESPACE users
    STORAGE ( INITIAL 250K
              NEXT 50K
              MINEXTENTS 1
              MAXEXTENTS 3
              PCTINCREASE 0 )
    HASH IS trialno
```

```
    HASHKEYS 150;
```

次の文は、trial_clusterハッシュ・クラスタにtrial表を作成します。

```
CREATE TABLE trial (
    trialno NUMBER(5,0) PRIMARY KEY,
    ... )
    CLUSTER trial_cluster (trialno);
```

索引クラスタの場合と同様に、ハッシュ・クラスタのキーは、単一列でもコンポジット・キー(複数列のキー)でもかまいません。前述の例では、trialno列がキーとなっています。

HASHKEYS値(この例では150)は、ハッシュ関数が生成できる一意のハッシュ値の数を指定し、制限します。指定した値は最も近い素数に丸められます。

HASH IS句を指定しない場合は、内部ハッシュ関数が使用されます。すでにクラスタ・キーがその範囲に均一に分布する一意識別子の場合は、内部ハッシュ関数を無視し、前述の例のようにクラスタ・キーをハッシュ値として指定できます。また、HASH IS句を使用して、ユーザー定義のハッシュ関数を指定することもできます。

ハッシュ・クラスタのクラスタ索引は作成できませんが、ハッシュ・クラスタ・キーに索引を作成する必要はありません。

関連項目:

クラスタ内に表を作成する方法、索引とハッシュ・クラスタに共通するCREATE CLUSTER文のパラメータ設定のガイドライン、およびクラスタを作成するために必要な権限の追加情報は、「クラスタの管理」

親トピック: 様々なタイプのハッシュ・クラスタの作成

23.3.2 ソートされたハッシュ・クラスタの作成

ソートされたハッシュ・クラスタには、ハッシュ関数の各値に対応する行が、ソートされた順序で効率よくデータベースから返すことができるような方法で格納されます。常にソート順にデータを使用するアプリケーションの場合、ソートされたハッシュ・クラスタを使用することによって論理I/Oを最小化でき、これにより、より迅速にデータを取得できます。

ある電話会社では、1つの交換機を介した固定数の発信電話番号について、詳細な通話記録を保存するとします。各発信電話番号から、無制限の通話件数が発生する可能性があります。

アプリケーションは、通話が発信されると通話記録を保存します。各通話には、タイムスタンプによって識別される詳細な通話記録があります。たとえば、通話記録がタイムスタンプ0とともに保存され、続いて通話記録がタイムスタンプ1とともに格納されるというように保存されます。

各発信電話番号の請求書が生成されるときには、先入れ先出し(FIFO)で処理されます。次の表に、3つの発信電話番号について詳細な例を示します。

telephone_number	call_timestamp
6505551212	0, 1, 2, 3, 4, ...
6505551213	0, 1, 2, 3, 4, ...
6505551214	0, 1, 2, 3, 4, ...

次のSQL文では、telephone_number列がハッシュ・キーです。ハッシュ・クラスタは、call_timestampおよびcall_duration列でソートされます。この例では表定義のクラスタ化とソート用の列にクラスタ定義と同じ名前を使用していますが、これは必須ではありません。ハッシュ・キーの数は、10桁の電話番号に基づいています。

```
CREATE CLUSTER call_detail_cluster (
   telephone_number NUMBER,
   call_timestamp   NUMBER SORT,
   call_duration    NUMBER SORT )
  HASHKEYS 10000
  HASH IS telephone_number
  SIZE 256;
CREATE TABLE call_detail (
   telephone_number     NUMBER,
   call_timestamp       NUMBER   SORT,
   call_duration        NUMBER   SORT,
   other_info           VARCHAR2(30) )
  CLUSTER call_detail_cluster (
   telephone_number, call_timestamp, call_duration );
```

例23-1 連続した順序で挿入されるデータ

この例に示すように、call_detail表に行をFIFOで供給するとします。

```
INSERT INTO call_detail VALUES (6505551212, 0, 9, 'misc info');
INSERT INTO call_detail VALUES (6505551212, 1, 17, 'misc info');
INSERT INTO call_detail VALUES (6505551212, 2, 5, 'misc info');
INSERT INTO call_detail VALUES (6505551212, 3, 90, 'misc info');
INSERT INTO call_detail VALUES (6505551213, 0, 35, 'misc info');
INSERT INTO call_detail VALUES (6505551213, 1, 6, 'misc info');
INSERT INTO call_detail VALUES (6505551213, 2, 4, 'misc info');
INSERT INTO call_detail VALUES (6505551213, 3, 4, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 0, 15, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 1, 20, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 2, 1, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 3, 25, 'misc info');
COMMIT;
```

例23-2 call_detailの問合せ

この例では、SET AUTOTRACE ONの後、電話番号6505551212のコール詳細についてcall_detail表を問い合せます。

```
SQL> SET AUTOTRACE ON;
SQL> SELECT * FROM call_detail WHERE telephone_number = 6505551212;

TELEPHONE_NUMBER CALL_TIMESTAMP CALL_DURATION OTHER_INFO
---------------- -------------- ------------- ------------------------------
      6505551212              0             9 misc info
      6505551212              1            17 misc info
      6505551212              2             5 misc info
      6505551212              3            90 misc info

Execution Plan
----------------------------------------------------------
Plan hash value: 2118876266

-------------------------------------------------------------------------------
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |     1 |    56 |     0   (0)|
|*  1 |  TABLE ACCESS HASH| CALL_DETAIL |     1 |    56 |            |
-------------------------------------------------------------------------------
```

問合せは、問合せプランにソートがないにもかかわらず、タイムスタンプ順に行を取得しています。

次に、既存の行を削除し、その同じ行を不規則な順序で挿入するとします。

```
DELETE FROM call_detail;
INSERT INTO call_detail VALUES (6505551213, 3, 4, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 0, 15, 'misc info');
INSERT INTO call_detail VALUES (6505551212, 0, 9, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 1, 20, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 2, 1, 'misc info');
INSERT INTO call_detail VALUES (6505551213, 1, 6, 'misc info');
INSERT INTO call_detail VALUES (6505551213, 2, 4, 'misc info');
INSERT INTO call_detail VALUES (6505551214, 3, 25, 'misc info');
INSERT INTO call_detail VALUES (6505551212, 1, 17, 'misc info');
INSERT INTO call_detail VALUES (6505551212, 2, 5, 'misc info');
INSERT INTO call_detail VALUES (6505551212, 3, 90, 'misc info');
INSERT INTO call_detail VALUES (6505551213, 0, 35, 'misc info');
COMMIT;
```

call_detailに同じ問合せを再実行すると、ORDER BY句を指定していない場合でも、データベースから再度行がソート順に取得されます。データベース内部でソートが実行されるため、SORT ORDER BY操作は問合せプランに表示されません。

ここで、非クラスタ化表call_detail_nonclusteredを作成して、例23-1に示したものと同じ値をロードするとします。データをソート順に取得するには、次のようにORDER BY句を使用する必要があります。

```
SQL> SELECT * FROM call_detail_nonclustered WHERE telephone_number = 6505551212
  2  ORDER BY call_timestamp, call_duration;

TELEPHONE_NUMBER CALL_TIMESTAMP CALL_DURATION OTHER_INFO
---------------- -------------- ------------- ------------------------------
      6505551212              0             9 misc info
      6505551212              1            17 misc info
      6505551212              2             5 misc info
      6505551212              3            90 misc info

Execution Plan
----------------------------------------------------------
Plan hash value: 2555750302

-------------------------------------------------------------------------------------
|Id| Operation          | Name                      |Rows|Bytes|Cost (%CPU)|Time  |
-------------------------------------------------------------------------------------
| 0| SELECT STATEMENT   |                           | 4  | 224 | 4 (25)| 00:00:01 |
| 1|  SORT ORDER BY     |                           | 4  | 224 | 4 (25)| 00:00:01 |
|*2|   TABLE ACCESS FULL| CALL_DETAIL_NONCLUSTERED | 4  | 224 | 3  (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

前述のプランで示した非クラスタ化の場合、ソートを行うにはクラスタ化の場合よりもコストがかかります。ソートされたハッシュ・クラスタに格納されていない表では、行、バイト、コスト、時間のすべてが大きくなります。

親トピック: 様々なタイプのハッシュ・クラスタの作成

23.3.3 単一表ハッシュ・クラスタの作成

表の行への高速アクセスを提供する単一表ハッシュ・クラスタを作成できます。ただし、この表はハッシュ・クラスタ内の唯一の表にする必要があります。

ハッシュ・キーとデータ行の間に1対1のマッピングが必要になるためです。次の文は、クラスタ・キーvarietyを持つ単一表ハッシュ・クラスタpeanutを作成します。

```
CREATE CLUSTER peanut (variety NUMBER)
   SIZE 512 SINGLE TABLE HASHKEYS 500;
```

HASHKEYS値は最も近い素数に丸められるため、このクラスタはそれぞれサイズ512バイトのハッシュ・キー値を最大503個持ちます。SINGLE TABLE句は、ハッシュ・クラスタにのみ有効です。また、必ずHASHKEYSも指定する必要があります。

関連項目:

CREATE CLUSTER文の構文は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 様々なタイプのハッシュ・クラスタの作成

23.3.4 ハッシュ・クラスタ内の領域使用の制御

ハッシュ・クラスタを作成するときは、パフォーマンスと使用領域が最適になるように、クラスタ・キーを正しく選択し、HASH IS、SIZEおよびHASHKEYSの各パラメータを設定することが重要です。次に示すガイドラインでは、これらのパラメータを設定する方法について説明します。

- キーの選択
  正しいクラスタ・キーの選択は、クラスタ表に対して最もよく発行される問合せのタイプによって決まります。
- HASH ISの設定
  HASH ISパラメータは、クラスタ・キーがNUMBERデータ型の単一の列であり、均一に分布した整数を含む場合にのみ指定します。
- SIZEの設定
  SIZEは、ハッシュ・キーに対応するすべての行を保持するために必要な領域の平均サイズに設定します。
- HASHKEYSの設定
  ハッシュ・クラスタを作成し、ハッシュ・クラスタのハッシュ値の数を指定するために、HASHKEYS句を指定します。
- ハッシュ・クラスタ内の使用領域の制御
  例では、クラスタ・キーを正しく選択し、HASH IS、SIZEおよびHASHKEYSの各パラメータを設定する方法を示します。すべての例において、データ・ブロック・サイズは2KBで、利用可能なデータ領域(ブロック・サイズからオーバーヘッドを差し引いた領域)の平均は各ブロックの1,950バイトとします。

親トピック: 様々なタイプのハッシュ・クラスタの作成

23.3.4.1 キーの選択

正しいクラスタ・キーの選択は、クラスタ化表に対して最もよく発行される問合せのタイプによって決まります。

たとえば、ハッシュ・クラスタ内のemp表について検討します。問合せが従業員番号によって行を頻繁に選択する場合、最適なクラスタ・キーはempno列です。問合せが部門によって行を頻繁に選択する場合、最適なクラスタ・キーはdeptno列です。単一の表を含むハッシュ・クラスタの場合は、通常、含まれる表の主キー全体をクラスタ・キーにします。

ハッシュ・クラスタのキーは、索引クラスタのキーと同じように単一列でもコンポジット・キー(複数列キー)でもかまいません。コンポジット・キーによるハッシュ・クラスタは、データベースの内部ハッシュ関数を使用する必要があります。

親トピック: ハッシュ・クラスタ内の領域使用の制御

23.3.4.2 HASH ISパラメータの設定

HASH ISパラメータを指定するのは、クラスタ・キーがNUMBERデータ型の単一の列であり、均一に分布した整数を含む場合のみです。

この条件を満たしていれば、それぞれの一意のクラスタ・キー値が衝突(同じハッシュ値を持つクラスタ・キーが2つ発生すること)せずに一意のハッシュ値にハッシュするように、クラスタ内に行を分散させることができます。この条件が当てはまらない場合は、この

句を指定せずに内部ハッシュ関数を使用してください。

親トピック: ハッシュ・クラスタ内の領域使用の制御

23.3.4.3 SIZEパラメータの設定

SIZEは、ハッシュ・キーに対応するすべての行を保持するために必要な領域の平均サイズに設定します。

そのため、SIZEを適切に決定するには、格納するデータの特性をよく理解する必要があります。

- ハッシュ・クラスタに含まれている表が1つで、その表の行のハッシュ・キー値が一意(1つの値につき1行)である場合、SIZEはクラスタ内の平均の行サイズに設定できます。
- ハッシュ・クラスタが複数の表を含む場合、SIZEは代表的なハッシュ値に対応するすべての行を保持するために必要な領域の平均サイズに設定できます。

また、SIZEの見積り値を決定した後で、次の点を考慮してください。SIZEの値が小さい場合(データ・ブロックごとに5つ以上のハッシュ・キーを割り当てることができる場合)は、その値をCREATE CLUSTER文のSIZEに使用できます。しかし、SIZEの値が大きい場合(データ・ブロックごとに割当て可能なハッシュ・キーが4つ以下の場合)は、衝突の発生頻度を予測し、データ検索パフォーマンスと領域使用効率のどちらを重視するか検討する必要があります。

- ハッシュ・クラスタで内部ハッシュ関数を使用せず(HASH ISを指定した場合)、衝突がわずかであるか、または衝突のないことが予想される場合は、見積り値をそのままSIZEに設定できます。この場合、衝突は発生せず、領域は可能なかぎり効率的に使用されます。

. 挿入時に頻繁な衝突が予想される場合、行を格納するためにオーバーフロー・ブロックが割り当てられる可能性は高くなります。衝突が頻繁に起こる場合にブロックのオーバーフローの可能性を軽減し、最大のパフォーマンスを引き出すには、次のようにSIZEを調整する必要があります。

ブロック当たりの使用可能領域/算出されたSIZE	SIZEの設定
1	SIZE
2	SIZE + 15%
3	SIZE + 12%
4	SIZE + 8%
>4	SIZE

ただし、SIZEの値を過大に見積ると、クラスタ内の未使用領域を増やすことになります。領域効率がデータ検索のパフォーマンスよりも重要な場合は、前述の表で示した調整を無視して、SIZEに元の値を使用してください。

親トピック: ハッシュ・クラスタ内の領域使用の制御

23.3.4.4 HASHKEYSパラメータの設定

HASHKEYSを指定すると、ハッシュ・クラスタを作成し、ハッシュ・クラスタのハッシュ値の数を指定できます。

ハッシュ・クラスタ内の行を最大限まで分散させるために、データベースはHASHKEYS値を最も近い素数に丸めます。

親トピック: ハッシュ・クラスタ内の領域使用の制御

23.3.4.5 ハッシュ・クラスタ内の使用領域の制御

例では、クラスタ・キーを正しく選択し、HASH IS、SIZEおよびHASHKEYSの各パラメータを設定する方法を示します。すべての例において、データ・ブロック・サイズは2KBで、利用可能なデータ領域(ブロック・サイズからオーバーヘッドを差し引いた領域)の平均は各ブロックの1,950バイトとします。

- ハッシュ・クラスタ内の使用領域の制御: 例1
  ハッシュ・クラスタの領域を制御する例を示します。
- ハッシュ・クラスタ内の使用領域の制御: 例2
  ハッシュ・クラスタの領域を制御する例を示します。

親トピック: ハッシュ・クラスタ内の領域使用の制御

23.3.4.5.1 ハッシュ・クラスタ内の使用領域の制御: 例1

ハッシュ・クラスタの領域を制御する例を示します。

ハッシュ・クラスタにemp表をロードすることになりました。ほとんどの問合せは、従業員番号によって従業員レコードを検索します。emp表の最大行数は常に10,000、平均の行サイズは55バイトと見積られています。

この場合は、empnoをクラスタ・キーにします。この列には一意の整数が格納されているので、内部ハッシュ関数は無視できます。SIZEは平均の行サイズ(55バイト)に設定できます。これにより、各データ・ブロックに34のハッシュ・キーが割り当てられます。HASHKEYSは、表の行数である10,000に設定できます。この値は、10,000より大きい最初の素数である10,007に切り上げられます。

```
CREATE CLUSTER emp_cluster (empno
NUMBER)
. . .
SIZE 55
HASH IS empno HASHKEYS 10000;
```

親トピック: ハッシュ・クラスタ内の使用領域の制御

23.3.4.5.2 ハッシュ・クラスタ内の使用領域の制御: 例2

ハッシュ・クラスタの領域を制御する例を示します。

この例の条件は、「ハッシュ・クラスタ内の使用領域の制御: 例1」の例と同様です。ただし、この例では、ほとんどの場合、行が部門番号によって検索されるとします。平均10名の従業員を持つ部門が最大で1,000部門存在します。部門番号は10ずつ増加します(0、10、20、30、...)。

この場合は、deptnoをクラスタ・キーにします。この列には一様に分布する整数が格納されているので、内部ハッシュ関数は無視できます。事前に見積ったSIZE (各部門のすべての行を保持するために必要な領域の平均サイズ)は55 * 10バイト、つまり550バイトです。SIZEにこの値を使用して、各データ・ブロックに3つのハッシュ・キーのみを割り当てることができます。いくらかの衝突が予想される状況で、最大限のデータ検索パフォーマンスが必要な場合、オーバーフロー・ブロックを必要とする衝突を防ぐために、見積りのSIZEを少し変更してください。SIZEを12%調整して620バイトにすることにより(「SIZEの設定」を参照)、予想される衝突を考慮した上で行の領域をより多く確保できます。

HASHKEYSは、一意の部門番号の数である1,000に設定できます。この値は、1,000より大きい最初の素数である1,009に切り上げられます。

```
CREATE CLUSTER emp_cluster (deptno NUMBER)
. . .
SIZE 620
```

```
HASH IS deptno HASHKEYS 1000;
```

親トピック: ハッシュ・クラスタ内の使用領域の制御

23.3.5 ハッシュ・クラスタに必要なサイズの見積り

索引クラスタの場合と同じように、ハッシュ・クラスタ内のデータに必要な記憶域を見積ることは重要です。

Oracle Databaseは、SIZEおよびHASHKEYSの設定に従って、ハッシュ表の格納に十分な領域の初期割当てを保証します。ハッシュ表サイズを考慮せずに記憶域パラメータINITIAL、NEXTおよびMINEXTENTSを設定した場合は、少なくともSIZE*HASHKEYSに達するまで増分(追加の)エクステントが割り当てられます。たとえば、データ・ブロック・サイズが2KB、各ブロックの使用可能なデータ領域(ブロック・サイズからオーバーヘッドを差し引いた領域)が約1,900バイトの場合に、CREATE CLUSTER文でSTORAGEパラメータとHASHパラメータを次のように指定したとします。

```
STORAGE (INITIAL 100K
    NEXT 150K
    MINEXTENTS 1
    PCTINCREASE 0)
SIZE 1500
HASHKEYS 100
```

この例では、各データ・ブロックにハッシュ・キーを1つのみ割り当てることができます。そのため、ハッシュ・クラスタが必要とする初期領域は少なくとも200KB(100×2KB)です。しかし、記憶域パラメータの設定にはこの要件が考慮されていません。そのため、100KBの初期のエクステントと150KBの増分のエクステントがハッシュ・クラスタに割り当てられます。

一方、HASHパラメータが次のように指定されたとします。

```
SIZE 500 HASHKEYS 100
```

この場合、各データ・ブロックに3つのハッシュ・キーが割り当てられます。そのため、ハッシュ・クラスタが必要とする初期領域は少なくとも68KB(34×2KB)です。記憶域パラメータの初期設定はこの要件を満たしているため、100KBの初期エクステントがハッシュ・クラスタに割り当てられます。

親トピック: 様々なタイプのハッシュ・クラスタの作成

23.4 ハッシュ・クラスタの変更

ハッシュ・クラスタは、ALTER CLUSTER文を使用して変更できます。

たとえば、次のALTER CLUSTER文は、emp_deptクラスタを変更します。

```
ALTER CLUSTER emp_dept . . . ;
```

ハッシュ・クラスタの変更に関係する問題は、「クラスタの変更」で説明されている索引クラスタを変更する場合と同じです。ただし、SIZE、HASHKEYSおよびHASH ISの各パラメータはALTER CLUSTER文に指定できません。これらのパラメータを変更するには、クラスタを再作成して、元のクラスタからデータをコピーする必要があります。

親トピック: ハッシュ・クラスタの管理

23.5 ハッシュ・クラスタの削除

ハッシュ・クラスタは、DROP CLUSTER文を使用して削除できます。

たとえば、次のDROP CLUSTER文は、emp_deptクラスタを削除します。

```
DROP CLUSTER emp_dept;
```

ハッシュ・クラスタ内の表は、DROP　TABLE文を使用して削除されます。ハッシュ・クラスタとハッシュ・クラスタ内の表の削除についての問題は、索引クラスタの場合と同じです。

関連項目:

「クラスタの削除」

親トピック: ハッシュ・クラスタの管理

23.6 ハッシュ・クラスタのデータ・ディクショナリ・ビュー

ハッシュ・クラスタに関する情報について一連のデータ・ディクショナリ・ビューを問い合せることができます。

次のビューには、ハッシュ・クラスタに関する情報が表示されます。

ビュー	説明
DBA_CLUSTERS ALL_CLUSTERS USER_CLUSTERS	DBA ビューには、データベース内のすべてのクラスタ(ハッシュ・クラスタを含む)が表示されます。ALL ビューには、ユーザーがアクセス可能なすべてのクラスタが表示されます。USER ビューは、ユーザーが所有するクラスタのみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE　文によって生成される統計が含まれます。
DBA_CLU_COLUMNS USER_CLU_COLUMNS	これらのビューでは、表の列とクラスタの列がマップされています。
DBA_CLUSTER_HASH_EXPRESSIONS ALL_CLUSTER_HASH_EXPRESSIONS USER_CLUSTER_HASH_EXPRESSIONS	これらのビューには、ハッシュ・クラスタのハッシュ関数がリストされます。

親トピック: ハッシュ・クラスタの管理

24 ビュー、順序およびシノニムの管理

Oracle Databaseでビュー、順序およびシノニムを作成および管理できます。

- ビューの管理
  ビューの作成、ビューの置換、ビューの変更およびビューの削除などのタスクを実行できます。
- 順序の管理
  順序の作成、順序の変更、順序の使用および順序の削除などのタスクを実行できます。
- シノニムの管理
  シノニムの作成、シノニムの使用およびシノニムの削除などのタスクを実行できます。
- ビュー、順序およびシノニムのデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューを問い合せて、ビュー、シノニムおよび順序に関する情報を取得できます。

親トピック: スキーマ・オブジェクト

24.1 ビューの管理

ビューの作成、ビューの置換、ビューの変更およびビューの削除などのタスクを実行できます。

Live SQL:

Oracle Live SQL でビューの管理に関連する例を参照および実行するには、「Oracle Live SQL: ビューの作成、置換および削除」に移動してください。

- ビューについて
  ビューとは、表または表の組合せの論理表現です。本質的には、ビューはストアド・クエリーです。
- ビューおよび結合ビューの作成
  ビューを作成するには、CREATE VIEW文を使用します。ビューはそれぞれ、表、マテリアライズド・ビューまたは他のビューを参照する問合せによって定義されます。FROM句で、複数の実表またはビューを指定する結合ビューを作成することもできます。
- ビューの置換
  ビューを置換するには、ビューを削除して再作成するか、OR REPLACE句を指定してCREATE VIEW文を発行します。
- 問合せでのビューの使用
  ビューの問合せを実行できます。ビューに対してデータ操作言語(DML)の操作も実行できますが、一部の制限があります。
- DML文と結合ビュー
  結合ビューに対してDML文を発行するときに制限が適用されます。
- ビューの変更
  ALTER VIEW文は、無効なビューを明示的に再コンパイルする場合にのみ使用します。
- ビューの削除
  ビューを削除するには、DROP VIEW文を使用します。

親トピック: ビュー、順序およびシノニムの管理

24.1.1 ビューについて

ビューとは、表または表の組合せの論理表現です。本質的には、ビューはストアド・クエリーです。

ビューのデータは、そのビューの基礎になる表から生成されます。この表を実表と呼びます。実表は、実際の表の場合もビュー自体の場合もあります。ビューに対して実行するすべての操作は、実際にはビューの実表に影響します。ビューの使用方法は、表の場合とほぼ同じです。通常の表と同様に、ビューに対する問合せ、更新、挿入および削除ができます。

ビューによって、他の表およびビューに常駐するデータの様々な表現(サブセットまたはスーパーセットなど)が可能です。ビューを使用すると、ユーザーの必要にあわせて調整したデータ表現ができます。

ノート:

ビューの 1 つの特殊なタイプがエディショニング・ビューで、エディションに基づく再定義を使用したオンラインでのアプリケーションのアップグレードをサポートするためにのみ使用されるビューです。ビューの管理に関する項のここからは、エディショニング・ビューを除くすべてのビューについて説明します。エディショニング・ビューおよびエディションに基づく再定義の詳細は、『Oracle Database 開発ガイド』を参照してください。

関連項目:

ビューの概要については、『Oracle Database概要』を参照してください。

親トピック: ビューの管理

24.1.2 ビューおよび結合ビューの作成

ビューを作成するには、CREATE VIEW文を使用します。ビューはそれぞれ、表、マテリアライズド・ビューまたは他のビューを参照する問合せによって定義されます。FROM句で、複数の実表またはビューを指定する結合ビューを作成することもできます。

- ビューの作成
  ビューを作成するには、CREATE VIEW文を使用します。
- 結合ビューの作成
  CREATE VIEW文のFROM句を使用して、複数の実表またはビューを指定するビューを作成することもできます。この種のビューを結合ビューと呼びます。
- ビュー作成時の問合せ定義の展開
  ビューが作成されるときに、Oracle Databaseは最上位のビュー問合せのワイルドカード(*)を列リストに展開します。結果の問合せデータ・ディクショナリに格納され、副問合せはそのまま残されます。
- エラー付きビューの作成
  CREATE VIEW文に構文エラーがない場合は、そのビューを定義している問合せを実行できなくても、データベースはビューを作成できます。この場合、ビューは、エラー付きで作成されたとみなされます。

親トピック: ビューの管理

24.1.2.1 ビューの作成

ビューを作成するには、CREATE VIEW文を使用します。

ビューを作成するには、次に示す要件を満たす必要があります。

- 自分のスキーマに新しいビューを作成するには、CREATE VIEWシステム権限が必要です。別のユーザーのスキーマ内にビューを作成するには、CREATE ANY VIEWシステム権限が必要です。これらの権限は明示的に取得するか、またはロールを介して取得できます。
- どのスキーマのビューであっても、その所有者は、ビュー定義で参照されるすべてのオブジェクトにアクセスする権限を明示的に付与されている必要があります。所有者がロールを介してこれらの権限を取得することはできません。また、ビューの機能は、ビューの所有者の権限によって決まります。たとえば、ビューの所有者にScottのemp表のINSERT権限しかない場合、emp表に新しい行を挿入するためにはビューを使用できますが、このビューの行を選択(SELECT)、更新(UPDATE)または削除(DELETE)するためには使用できません。
- ビューの所有者がビューにアクセスする権限を他のユーザーに付与しようとする場合は、ベース・オブジェクトに対するGRANT OPTION付きのオブジェクト権限、またはADMIN OPTION付きのシステム権限が必要です。

ビューを作成するには、CREATE VIEW文を使用します。ビューはそれぞれ、表、マテリアライズド・ビューまたは他のビューを参照する問合せによって定義されます。すべての副問合せと同様に、ビューを定義する問合せにFOR UPDATE句を含めることはできません。

次の文は、hr.departments表のデータのサブセットに対してビューを作成します。

```
CREATE VIEW departments_hq AS
      SELECT department_id, department_name, location_id
      FROM hr.departments
      WHERE location_id = 1700
    WITH CHECK OPTION CONSTRAINT departments_hq_cnst;
```

departments_hqビューを定義する問合せは、場所1700の行のみを参照します。また、CHECK OPTIONは、そのビューが選択できない行に対してINSERT文およびUPDATE文を発行できないように制約(departments_hq_cnst)付きでビューを作成します。たとえば、次のINSERT文では、departments_hqビュー(場所が1700の行のみを含む)を使用してdepartments表に行が正常に挿入されます。

```
INSERT INTO departments_hq VALUES (300, 'NETWORKING', 1700);
```

しかし、次のINSERT文は、departments_hqビューを使用しても選択できない場所2700の行を挿入しようとしているため、エラーが返されます。

```
INSERT INTO departments_hq VALUES (301, 'TRANSPORTATION', 2700);
```

WITH READ ONLY句を指定してビューを作成できますが、これにより、このビューからは、実表の更新、挿入または削除ができなくなります。WITH句を指定しない場合、一部の制限を伴いますが、ビューは従来どおり更新可能です。

不可視の列を含むビューを作成することもできます。たとえば、次の文ではdepartments_hq_manビューが作成され、manager_id列が不可視になります。

```
CREATE VIEW departments_hq_man
         (department_id, department_name, manager_id INVISIBLE, location_id)
      AS SELECT department_id, department_name, manager_id, location_id
      FROM hr.departments
      WHERE location_id = 1700
    WITH CHECK OPTION CONSTRAINT departments_hq_man_cnst;
```

関連項目:

- CREATE VIEW文の構文とセマンティクスの詳細は、『Oracle Database SQL言語リファレンス』を参照してください

- 「不可視の列の理解」

親トピック: ビューおよび結合ビューの作成

24.1.2.2 結合ビューの作成

CREATE VIEW文のFROM句を使用して、複数の実表またはビューを指定するビューを作成することもできます。この種のビューを結合ビューと呼びます。

次の文は、emp表とdept表のデータを結合するdivision1_staffビューを作成します。

```
CREATE VIEW division1_staff AS
      SELECT ename, empno, job, dname
      FROM emp, dept
      WHERE emp.deptno IN (10, 30)
         AND emp.deptno = dept.deptno;
```

更新可能な結合ビューは、UPDATE、INSERTおよびDELETE操作が可能な結合ビューです。詳細は、「結合ビューの更新」を参照してください。

親トピック: ビューおよび結合ビューの作成

24.1.2.3 ビュー作成時の問合せ定義の展開

ビューが作成されるときに、Oracle Databaseは最上位のビュー問合せのワイルドカード(*)を列リストに展開します。結果の問合せデータ・ディクショナリに格納され、副問合せはそのまま残されます。

展開された列リスト中の列名は引用符で囲まれており、これは、ベース・オブジェクトの列がもともと引用符付きで入力された可能性があり、問合せの構文を正しいものにするためには引用符が必要であることを示しています。

たとえば、deptビューが次のように作成される場合を想定します。

```
CREATE VIEW dept AS SELECT * FROM scott.dept;
```

データベースは、deptビューを定義している問合せを次のように格納します。

```
SELECT "DEPTNO", "DNAME", "LOC" FROM scott.dept;
```

エラー付きで作成されたビューでは、ワイルドカードは展開されません。エラーなしでビューがコンパイルされると、定義された問合せのワイルドカードが展開されます。

親トピック: ビューおよび結合ビューの作成

24.1.2.4 エラー付きビューの作成

CREATE VIEW文に構文エラーがない場合は、そのビューを定義している問合せを実行できなくても、データベースはビューを作成できます。この場合、ビューは、エラー付きで作成されたとみなされます。

たとえば、存在しない表や既存の表の無効な列を参照するビューを作成するとき、またはビューの所有者が必要な権限を持っていないときでも、ビューを作成し、データ・ディクショナリに登録できます。ただし、そのビューは使用できません。

エラー付きのビューを作成するには、CREATE VIEW文のFORCE句を指定する必要があります。

```
CREATE FORCE VIEW AS ...;
```

デフォルトでは、エラー付きのビューはINVALIDとして作成されます。このようなビューを作成しようとすると、ビューがエラー付きで作成されたことを示すメッセージが返されます。状況が変化して無効なビューの問合せが実行可能になると、ビューは再コンパイルされ、有効(使用可能)になります。条件の変更とビューに及ぼす影響の詳細は、「オブジェクト依存性の管理」を参照してく

ださい。

親トピック: ビューおよび結合ビューの作成

24.1.3 ビューの置換

ビューを置換するには、ビューを削除して再作成するか、OR REPLACE句を指定してCREATE VIEW文を発行します。

ビューを置換するには、ビューの削除および作成に必要なすべての権限が必要です。ビューの定義を変更する場合は、そのビューを置換する必要があり、ビューの定義の変更にALTER VIEW文は使用できません。次の方法でビューを置換できます。

- ビューを削除してから再作成します。

ノート:

ビューを削除するときに、ロールおよびユーザーに付与された対応するオブジェクト権限はすべて取り消されます。ビューを再作成してから、再度権限を付与してください。

- OR REPLACE句を含むCREATE VIEW文によって、ビューを再定義します。OR REPLACE句は、ビューの現行の定義を置換し、現行のセキュリティ認可を保存します。たとえば、前述のように、sales_staffビューを作成し、いくつかのオブジェクト権限をロールと他のユーザーに付与した場合を想定します。ただし、ここではsales_staffビューを再定義して、WHERE句に指定されている部門番号を変更するものとします。この場合は、次の文によって、sales_staffビューの現行バージョンを置換できます。

```
CREATE OR REPLACE VIEW sales_staff AS
     SELECT empno, ename, deptno
     FROM emp
     WHERE deptno = 30
     WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

ビューを置換する前に、次の影響を検討してください。

- ビューを置換することによって、データ・ディクショナリ内のビュー定義が置換されます。ビューによって参照される、基礎になるオブジェクトは影響を受けません。
- 前のビューにはCHECK OPTIONで制約を定義していたが、新しいビュー定義には指定しない場合、その制約は削除されます。
- 置換されたビューに依存しているビューはすべて無効(使用不可)になります。また、ビューの新しいバージョンでの変更内容によっては、依存しているPL/SQLプログラム・ユニットも無効になる場合があります。たとえば、ビューのWHERE句のみが変更された場合、依存しているPL/SQLプログラム・ユニットは有効のままです。ただし、ビューの列数、列名またはデータ型が変更された場合は、依存しているPL/SQLプログラム・ユニットも無効になります。データベースによる依存性管理の詳細は、「オブジェクト依存性の管理」を参照してください。

親トピック: ビューの管理

24.1.4 問合せでのビューの使用

ビューの問合せを実行できます。ビューに対してデータ操作言語(DML)の操作も実行できますが、一部の制限があります。

ビューに対して問合せを発行したり、INSERT、UPDATEまたはDELETE文を発行するためには、そのビューに対するSELECT、READ、INSERT、UPDATEまたはDELETEの各オブジェクト権限を、明示的にまたはロールを介して持っている必要があります。

ビューは、表と同じ方法で問い合せることができます。たとえば、Division1_staffビューを問い合せるには、そのビューを参照する有効なSELECT文を入力します。

```
SELECT * FROM Division1_staff;
ENAME        EMPNO       JOB              DNAME
-------------------------------------------------
CLARK         7782       MANAGER          ACCOUNTING
KING          7839       PRESIDENT        ACCOUNTING
MILLER        7934       CLERK            ACCOUNTING
ALLEN         7499       SALESMAN         SALES
WARD          7521       SALESMAN         SALES
JAMES         7900       CLERK            SALES
TURNER        7844       SALESMAN         SALES
MARTIN        7654       SALESMAN         SALES
BLAKE         7698       MANAGER          SALES
```

一部の制限を伴いますが、ビューを使用して、実表に対する行の挿入、更新または削除ができます。次の文は、sales_staffビューを使用してemp表に新しい行を挿入します。

```
INSERT INTO sales_staff
    VALUES (7954, 'OSTER', 30);
```

ビューに対するDML操作の制限では、次の基準がリストされている順に適用されます。

1. ビューの定義にSETまたはDISTINCT演算子、GROUP BY句またはグループ関数を含む問合せが使用されている場合、そのビューによる実表への行の挿入、更新または削除はできません。
2. ビューの定義にWITH CHECK OPTIONが使用されていて、行を実表から選択できない場合、そのビューによる実表への行の挿入または更新はできません。
3. DEFAULT句を持たないNOT NULL列がビューから省略されている場合、そのビューによる実表への行の挿入はできません。
4. ビューの作成にDECODE(deptno, 10, "SALES", ...)のような式が使用されている場合、そのビューによる実表への行の挿入または更新はできません。

sales_staffビューのWITH CHECK OPTIONで作成された制約によって許可されるのは、部門番号30を持つ行のemp表への挿入または更新のみです。一方、次の文(つまり、deptno列を除外する)でsales_staffビューが定義されるとします。

```
CREATE VIEW sales_staff AS
    SELECT empno, ename
    FROM emp
    WHERE deptno = 10
    WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

このビュー定義を検討すると、既存のレコードのempnoまたはenameフィールドは更新できますが、ビューではdeptnoフィールドを変更できないため、sales_staffビューを介してemp表に行を挿入することはできません。deptnoフィールドにDEFAULT値10を定義していた場合は、挿入を実行できます。

ユーザーが無効なビューを参照しようとすると、次のエラー・メッセージが返されます。

```
ORA-04063: view 'view_name' has errors
```

このエラー・メッセージが返されるのは、ビューは存在しているが、問合せのエラーが原因で使用禁止状態の場合です(このビューが最初に作成されたときにあったエラーか、またはビューは正常に作成されたが、基礎になるオブジェクトが変更または削除されたために、後で使用禁止状態になったかは関係ありません)。

親トピック: ビューの管理

24.1.5 DML文と結合ビュー

結合ビューに対してDML文を発行するときに制限が適用されます。

- 結合ビューの更新
  更新可能な結合ビュー(変更可能な結合ビューと呼ぶこともあります)とは、SELECT文の最上位のFROM句に複数の表を含むビューで、WITH READ ONLY句の制限を受けないものです。
- キー保存表
  キー保存表の概念は、結合ビューを更新するうえでの制限を理解するために重要です。表のすべてのキーが結合の結果のキーでもある場合、その表はキー保存になります。つまり、キー保存表とは、結合後もそのキーを保存している表のことです。
- DML文と結合ビューのルール
  一般規則では、結合ビューにおいて、UPDATE、INSERTまたはDELETE文では、基礎となる実表を1つしか変更できません。
- 外部結合が含まれるビューの更新
  外部結合が含まれるビューは、変更可能な場合もあります。
- UPDATABLE_COLUMNSビューの使用
  従来どおり更新可能な結合ビューを識別する際に、このビューのセットが役立ちます。

親トピック: ビューの管理

24.1.5.1 結合ビューの更新

更新可能な結合ビュー(変更可能な結合ビューと呼ぶこともあります)とは、SELECT文の最上位のFROM句に複数の表を含むビューで、WITH READ ONLY句の制限を受けないものです。

更新可能な結合ビューに関する規則を次の表に示します。これらの基準を満たすビューは、従来どおり更新可能とみなされます。

規則	説明
一般規則	結合ビューに対する INSERT、UPDATE または DELETE 操作は、基礎になる実表を一度に 1 つしか変更できません。
UPDATE 規則	結合ビューの更新可能な列はすべて、キー保存表の列にマップする必要があります。キー保存表については、「キー保存表」を参照してください。ビューの定義に WITH CHECK OPTION 句が使用されている場合は、すべての結合列および繰返し表の列はいずれも更新できません。
DELETE 規則	結合の中にキー保存表が 1 つしかない場合には、結合ビューから行を削除できます。このキー保存表は、FROM 句で繰り返すことができます。ビューの定義に WITH CHECK OPTION 句が使用されていて、キー保存表が繰り返される場合は、そのビューから行を削除できません。
INSERT 規則	INSERT 文では、明示的にも暗黙的にも非キー保存表の列を参照しないでください。結合ビューの定義に WITH CHECK OPTION 句が使用されている場合は、INSERT 文を使用できません。

結合ビュー内の列が従来どおり更新可能かどうかを示すデータ・ディクショナリ・ビューがあります。これらのビューの詳細は、「UPDATABLE_COLUMNSビューの使用」を参照してください。

ノート:

結合ビューが従来どおり更新可能かどうかについては、いくつかのその他の制限と条件が影響します。これらの制限と条件については、『Oracle Database SQL 言語リファレンス』の CREATE VIEW 文の説明を参照してください。

ビューが従来どおり更新可能でない場合は、そのビューに INSTEAD OF トリガーを作成して更新可能にできます。トリガーの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

また、ビューが別のネストされたビュー上の結合である場合は、そのネストされたビューをトップレベル・ビューにマージ可能である必要があります。マージ可能なビューとマージ不可能なビューの詳細、およびオプティマイザによってビューを参照する文が最適化される方法の概要は、『Oracle Database SQL チューニング・ガイド』を参照してください。

ここでは、従来どおり更新可能な結合ビューの規則の例を示し、キー保存表について説明します。それぞれの例は、表に主キーと外部キーを明示的に定義した場合、または一意索引を定義した場合にのみ動作します。次の文は、empおよびdeptの制約が適切に設定された表定義を作成します。

```
CREATE TABLE dept (
      deptno        NUMBER(4) PRIMARY KEY,
      dname         VARCHAR2(14),
      loc           VARCHAR2(13));

CREATE TABLE emp (
      empno        NUMBER(4) PRIMARY KEY,
      ename        VARCHAR2(10),
      job          VARCHAR2(9),
      mgr          NUMBER(4),
      sal          NUMBER(7,2),
      comm         NUMBER(7,2),
      deptno       NUMBER(2),
      FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO));
```

また、前述の例の主キーと外部キーの制約を省略し、dept(deptno)にUNIQUE INDEXを作成した場合でも、後続の例は動作可能です。

次の文は、例で参照されるemp_dept結合ビューを作成します。

```
CREATE VIEW emp_dept AS
      SELECT emp.empno, emp.ename, emp.deptno, emp.sal, dept.dname, dept.loc
      FROM emp, dept
      WHERE emp.deptno = dept.deptno
         AND dept.loc IN ('DALLAS', 'NEW YORK', 'BOSTON');
```

親トピック: DML文と結合ビュー

24.1.5.2 キー保存表

キー保存表の概念は、結合ビューを更新するうえでの制限を理解するために重要です。表のすべてのキーが結合の結果のキーでもある場合、その表はキー保存になります。つまり、キー保存表とは、結合後もそのキーを保存している表のことです。

ノート:

表をキー保存にするために、表の 1 つ以上のキーを選択する必要はありません。1 つ以上のキーを選択した場合

に、そのキーが結合の結果のキーであれば十分です。

表のキー保存特性は、表内の実際のデータには依存しません。これは、そのスキーマの特性です。たとえば、emp表で各部門に多くても1人の従業員しか含まれていない場合、empとdeptの結合の結果ではdeptnoは一意ですが、deptはキー保存表ではありません。

emp_deptからすべての行を選択すると、結果は次のようになります。

```
EMPNO      ENAME      DEPTNO  DNAME          LOC
---------- ---------- ------- -------------- -----------
      7782 CLARK           10 ACCOUNTING     NEW YORK
      7839 KING            10 ACCOUNTING     NEW YORK
      7934 MILLER          10 ACCOUNTING     NEW YORK
      7369 SMITH           20 RESEARCH       DALLAS
      7876 ADAMS           20 RESEARCH       DALLAS
      7902 FORD            20 RESEARCH       DALLAS
      7788 SCOTT           20 RESEARCH       DALLAS
      7566 JONES           20 RESEARCH       DALLAS
8 rows selected.
```

このビューでは、empnoがemp表のキー、および結合の結果のキーでもあるため、empがキー保存表です。deptnoはdept表のキーですが、結合のキーではないため、deptはキー保存表ではありません。

親トピック: DML文と結合ビュー

24.1.5.3 DML文と結合ビューのルール

一般規則では、結合ビューにおいて、UPDATE、INSERTまたはDELETE文では、基礎になる実表を1つしか変更できません。

- UPDATE文と結合ビュー
  例を使用して、結合ビューを変更できるUPDATE文について説明します。
- DELETE文と結合ビュー
  ほとんどの結合ビューでは、結合の中にキー保存表が1つしかない場合のみ、削除が成功します。このキー保存表は、FROM句で繰り返すことができます。
- INSERT文と結合ビュー
  例を使用して、結合ビューを変更できるINSERT文について説明します。

親トピック: DML文と結合ビュー

24.1.5.3.1 UPDATE文と結合ビュー

例を使用して、結合ビューを変更できるUPDATE文について説明します。

次の例は、emp_deptビューを正常に変更するUPDATE文を示したものです。

```
UPDATE emp_dept
     SET sal = sal * 1.10
     WHERE deptno = 10;
```

次に示すUPDATE文は、emp_deptビューには使用できません。

```
UPDATE emp_dept
     SET loc = 'BOSTON'
     WHERE ename = 'SMITH';
```

この文はdept実表を変更しようとしますが、dept表はemp_deptビューのキー保存表でないため、エラー番号ORA-01779「キー保存されていない表にマップする列は変更できません」が出力され、文を実行できません。

一般に、結合ビューの更新可能な列はすべて、キー保存表の列にマップする必要があります。ビューの定義にWITH CHECK OPTION句が使用されている場合は、すべての結合列およびビューで2回以上参照されている表から取得したすべての列はいずれも更新できません。

したがって、たとえばemp_deptビューの定義にWITH CHECK OPTIONが使用されている場合、次に示すUPDATE文は失敗します。

```
UPDATE emp_dept
     SET deptno = 10
     WHERE ename = 'SMITH';
```

結合列を更新しようとするため、この文は失敗となります。

関連項目:

UPDATE文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: DML文と結合ビューのルール

24.1.5.3.2 DELETE文と結合ビュー

ほとんどの結合ビューでは、結合の中にキー保存表が1つしかない場合のみ、削除が成功します。このキー保存表は、FROM句で繰り返すことができます。

次のDELETE文は、emp_deptビューに対して動作します。

```
DELETE FROM emp_dept
     WHERE ename = 'SMITH';
```

emp_deptビューに対するこのDELETE文は、実表empに対するDELETE操作に変換でき、表empは結合ビュー内の唯一のキー保存表であるため、この文は有効です。

次のビューでは、2つのキー保存表がありますが、同じ表であるため、DELETE操作を実行できます。つまり、キー保存表が繰り返されています。この場合、この削除文は、FROM句の最初の表(この例では、e1)で操作されます。

```
CREATE VIEW emp_emp AS
     SELECT e1.ename, e2.empno, e2.deptno
     FROM emp e1, emp e2
     WHERE e1.empno = e2.empno;
```

ビューの定義にWITH CHECK OPTION句が使用されていて、キー保存表が繰り返される場合は、そのビューから行を削除できません。

```
CREATE VIEW emp_mgr AS
     SELECT e1.ename, e2.ename mname
     FROM emp e1, emp e2
     WHERE e1.mgr = e2.empno
     WITH CHECK OPTION;
```

ノート:

- DELETE 文は、その文の WHERE 句内で結合条件としてビューの作成に使用したものと同じ列を使用する場合、結合内に異なるキー保存表が存在していても削除操作に成功します。この場合、DELETE 文は FROM 句の最初の表で機能するので、FROM 句の表は WHERE 句の表と異なってもかまいません。

- WHERE 句を使用しない場合でも、DELETE 文は正常に実行されます。
- 結合条件としてビューの作成に使用されたものではなく、その WHERE 句で別の列を使用する場合でも、DELETE 文は正常に実行されます。
- 主キーは 2 番目の表で定義されていないため、DELETE 文はすべての場合に FROM 句の 2 番目の表で機能します。
- 主キーが 2 番目の表で定義されると、DELETE 文は FROM 句内の最初の表で機能します。

関連項目:

DELETE文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: DML文と結合ビューのルール

24.1.5.3.3 INSERT文と結合ビュー

例を使用して、結合ビューを変更できるINSERT文について説明します。

emp_deptビューに対する次のINSERT文は正常に動作します。

```
INSERT INTO emp_dept (ename, empno, deptno)
   VALUES ('KURODA', 9010, 40);
```

変更されるキー保存実表は1つのみであり(emp)、40はdept表の有効なdeptnoであるため(つまり、emp表に対するFOREIGN KEY整合性制約を満たすため)、この文は動作します。

次のINSERT文は、実表empに対するUPDATEが失敗するのと同じ理由で失敗します。つまり、77というdeptnoが存在しないため、emp表に対するFOREIGN KEY整合性制約に違反します。

```
INSERT INTO emp_dept (ename, empno, deptno)
   VALUES ('KURODA', 9010, 77);
```

次のINSERT文はエラー番号ORA-01776「結合ビューを介して複数の実表を変更できません。」が出力されて失敗します。

```
INSERT INTO emp_dept (empno, ename, loc)
   VALUES (9010, 'KURODA', 'BOSTON');
```

INSERTは、暗黙的にも明示的にも、非キー保存表の列を参照できません。結合ビューの定義にWITH CHECK OPTION句が使用されている場合は、その結合ビューに対してINSERTを実行できません。

関連項目:

INSERT文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: DML文と結合ビューのルール

24.1.5.4 外部結合が含まれるビューの更新

外部結合が含まれるビューは、変更可能な場合もあります。

たとえば:

```
CREATE VIEW emp_dept_oj1 AS
    SELECT empno, ename, e.deptno, dname, loc
    FROM emp e, dept d
    WHERE e.deptno = d.deptno (+);
```

次の文を発行した場合:

```
SELECT * FROM emp_dept_oj1;
```

結果は次のとおりです。

```
EMPNO   ENAME      DEPTNO  DNAME          LOC
------- ---------- ------- -------------- -------------
7369    SMITH      40      OPERATIONS     BOSTON
7499    ALLEN      30      SALES          CHICAGO
7566    JONES      20      RESEARCH       DALLAS
7654    MARTIN     30      SALES          CHICAGO
7698    BLAKE      30      SALES          CHICAGO
7782    CLARK      10      ACCOUNTING     NEW YORK
7788    SCOTT      20      RESEARCH       DALLAS
7839    KING       10      ACCOUNTING     NEW YORK
7844    TURNER     30      SALES          CHICAGO
7876    ADAMS      20      RESEARCH       DALLAS
7900    JAMES      30      SALES          CHICAGO
7902    FORD       20      RESEARCH       DALLAS
7934    MILLER     10      ACCOUNTING     NEW YORK
7521    WARD       30      SALES          CHICAGO
14 rows selected.
```

emp_dept_oj1のemp実表の列は、empが結合の中のキー保存表であるため、ビューを介して変更可能です。

次のビューにも外部結合が含まれています。

```
CREATE VIEW emp_dept_oj2 AS
SELECT e.empno, e.ename, e.deptno, d.dname, d.loc
FROM emp e, dept d
WHERE e.deptno (+) = d.deptno;
```

次の文を発行した場合:

```
SELECT * FROM emp_dept_oj2;
```

結果は次のとおりです。

```
EMPNO      ENAME      DEPTNO    DNAME          LOC
---------- ---------- --------- -------------- ----
7782       CLARK      10        ACCOUNTING     NEW YORK
7839       KING       10        ACCOUNTING     NEW YORK
7934       MILLER     10        ACCOUNTING     NEW YORK
7369       SMITH      20        RESEARCH       DALLAS
7876       ADAMS      20        RESEARCH       DALLAS
7902       FORD       20        RESEARCH       DALLAS
7788       SCOTT      20        RESEARCH       DALLAS
7566       JONES      20        RESEARCH       DALLAS
7499       ALLEN      30        SALES          CHICAGO
7698       BLAKE      30        SALES          CHICAGO
7654       MARTIN     30        SALES          CHICAGO
7900       JAMES      30        SALES          CHICAGO
7844       TURNER     30        SALES          CHICAGO
7521       WARD       30        SALES          CHICAGO
                                 OPERATIONS     BOSTON
15 rows selected.
```

このビューでは、結合の結果のempno列がNULL(前述のSELECT文の最終行)になる可能性があるため、empはキー保存表ではなくなります。したがって、このビューでは、UPDATE、DELETEおよびINSERT操作を実行できません。

別のネストされたビュー上の外部結合を含むビューの場合は、表を含むビューがその外部ビューの最上位まですべてマージされる場合に、その表はキー保存になります。外部結合されているビューは現在、単純な場合にのみマージされます。たとえば:

```
SELECT col1, col2, ... FROM T;
```

ビューのSELECTリストには式がありません。

ビューが更新可能かどうか不確かな場合は、USER_UPDATABLE_COLUMNSビューから選択することで確認できます。たとえば:

```
SELECT owner, table_name, column_name, updatable FROM USER_UPDATABLE_COLUMNS
     WHERE TABLE_NAME = 'EMP_DEPT_VIEW';
```

ここでは、次のような出力が返されます。

```
OWNER       TABLE_NAME      COLUMN_NAM      UPD
----------  ----------      ----------      ---
SCOTT       EMP_DEPT_V      EMPNO           NO
SCOTT       EMP_DEPT_V      ENAME           NO
SCOTT       EMP_DEPT_V      DEPTNO          NO
SCOTT       EMP_DEPT_V      DNAME           NO
SCOTT       EMP_DEPT_V      LOC             NO
5 rows selected.
```

親トピック: DML文と結合ビュー

24.1.5.5 UPDATABLE_COLUMNSビューの使用

従来どおり更新可能な結合ビューを識別する際に、このビューのセットが役立ちます。

ビュー	説明
DBA_UPDATABLE_COLUMNS	変更可能なすべての表とビューのすべての列が表示されます。
ALL_UPDATABLE_COLUMNS	ユーザーがアクセス可能で変更可能なすべての表とビューのすべての列が表示されます。
USER_UPDATABLE_COLUMNS	ユーザーのスキーマ内で変更可能なすべての表とビューのすべての列が表示されます。

次に、emp_deptビュー内の更新可能な列を示します。

```
SELECT COLUMN_NAME, UPDATABLE
      FROM USER_UPDATABLE_COLUMNS
      WHERE TABLE_NAME = 'EMP_DEPT';
COLUMN_NAME                    UPD
------------------------------ ---
EMPNO                          YES
ENAME                          YES
DEPTNO                         YES
SAL                            YES
DNAME                          NO
LOC                            NO
6 rows selected.
```

関連項目:

更新可能な列のビューの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: DML文と結合ビュー

24.1.6 ビューの変更

ALTER VIEW文は、無効なビューを明示的に再コンパイルする場合にのみ使用します。

ビューの定義を変更する場合は、「ビューの置換」を参照してください。

ALTER VIEW文を使用すると、実際に使用する前に再コンパイル・エラーを調べることができます。ビューの実表の1つを変更したとき、変更がビューまたはそれに依存する他のオブジェクトに影響しないようにするには、そのビューを明示的に再コンパイルします。

ALTER VIEW文を使用するには、そのビューが自分のスキーマに含まれているか、またはALTER ANY TABLEシステム権限を持っている必要があります。

関連項目:

ALTER VIEW文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: ビューの管理

24.1.7 ビューの削除

ビューを削除するには、DROP VIEW文を使用します。

自分のスキーマにあるビューはすべて削除できます。別のユーザーのスキーマ内にあるビューを削除するには、DROP ANY VIEWシステム権限が必要です。ビューを削除するには、DROP VIEW文を使用します。たとえば、次の文はemp_deptビューを削除します。

```
DROP VIEW emp_dept;
```

関連項目:

DROP VIEW文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: ビューの管理

24.2 順序の管理

順序の作成、順序の変更、順序の使用および順序の削除などのタスクを実行できます。

- 順序について
  順序とは、複数のユーザーが一意の整数を生成するために使用できるデータベース・オブジェクトです。順序ジェネレータは順序番号を生成し、一意主キーの自動的な生成、および複数の行または表の間のキーの調整に使用できます。
- 順序の作成
  順序を作成するには、CREATE SEQUENCE文を使用します。

- 順序の変更
  順序を変更するには、ALTER SEQUENCE文を使用します。
- 順序の使用
  複数のユーザーが順序にアクセスして増分できます。
- 順序の削除
  不要になった順序は、DROP SEQUENCE文を使用して削除できます。

親トピック: ビュー、順序およびシノニムの管理

24.2.1 順序について

順序とは、複数のユーザーが一意の整数を生成するために使用できるデータベース・オブジェクトです。順序ジェネレータは順序番号を生成し、一意主キーの自動的な生成、および複数の行または表の間のキーの調整に使用できます。

順序がない場合、シーケンシャル値はプログラムによって生成されるのみです。新しい主キー値は、最後に生成された値を選択し、その値を増分することによって取得できます。この方法では、トランザクション中のロックが必要になるため、複数のユーザーが主キーの次の値を待機することになります。この待機をシリアライズと呼びます。開発者がこのような構成メンバーをアプリケーションに設定している場合は、順序へのアクセスに置き換えるように薦めてください。順序によってシリアライズが解消され、アプリケーションの同時実行性が改善されます。

関連項目:

順序の概要については、『Oracle Database概要』を参照してください。

親トピック: 順序の管理

24.2.2 順序の作成

順序を作成するには、CREATE SEQUENCE文を使用します。

自分のスキーマに順序を作成するには、CREATE SEQUENCEシステム権限が必要です。別のユーザーのスキーマ内に順序を作成するには、CREATE ANY SEQUENCE権限が必要です。

たとえば、次の文は、emp表のempno列に対して従業員番号を生成するために使用する順序を作成します。

```
CREATE SEQUENCE emp_sequence
       INCREMENT BY 1
       START WITH 1
       NOMAXVALUE
       NOCYCLE
       CACHE 10;
```

複数のパラメータを指定して、順序の機能を制御できます。これらのパラメータを使用して、順序の昇順または降順、順序の開始点、最大値と最小値、および順序値の間隔を指定できます。NOCYCLEオプションは、順序が最大値または最小値に達すると、それ以上の値を生成できなくなることを示します。

CACHE句は順序番号により高速にアクセスできるように、順序番号の集合をメモリーに事前割当てし、維持します。キャッシュ内の最後の順序番号が使用されると、別の順序の集合がキャッシュ内に読み込まれます。

順序番号の集合をキャッシュする場合に、順序番号がスキップされることがあります。たとえば、インスタンスが異常停止すると(たとえばインスタンス障害が発生したり、SHUTDOWN ABORT文が発行されたりすると)、キャッシュされているが使用されていない順序番号は失われます。また、使用されても保存されなかった順序番号も失われます。さらに、エクスポートとインポートの後、

データベースがキャッシュされた順序番号をスキップすることもあります。詳細は、『Oracle Databaseユーティリティ』を参照してください。

関連項目:

- CREATE SEQUENCE文の構文については、『Oracle Database SQL言語リファレンス』を参照してください
- Oracle Real Application Clusters環境で順序を使用する方法については、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

親トピック: 順序の管理

24.2.3 順序の変更

順序を変更するには、ALTER SEQUENCE文を使用します。

順序を変更するには、スキーマに順序が含まれている必要があり、順序に対するALTERオブジェクト権限またはALTER ANY SEQUENCEシステム権限が必要です。順序を変更して、順序番号の生成方法を定義するパラメータを変更できます。順序の開始点を変更するには、順序を削除してから再作成するか、RESTART句を使用して順序を再開します。昇順の場合、RESTART句はNEXTVALをMINVALUEにリセットします。降順の場合、NEXTVALはMAXVALUEにリセットされます。

次の例では、emp_sequenceの順序を変更します。

```
ALTER SEQUENCE emp_sequence
    INCREMENT BY 10
    MAXVALUE 10000
    CYCLE
    CACHE 20;
```

関連項目:

ALTER SEQUENCE文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: 順序の管理

24.2.4 順序の使用

複数のユーザーが順序にアクセスして増分できます。

順序を使用するには、順序が自分のスキーマに含まれているか、または別のユーザーの順序に対するSELECTオブジェクト権限が付与されている必要があります。定義済の順序は、複数のユーザー(該当する順序を含む順序に対するSELECTオブジェクト権限を持つユーザー)が待機せずにアクセスおよび増分できます。順序を増分したトランザクションの完了を待機せずに、その順序は再び増分されます。

次の各項の例では、マスター/ディテール表の関係における順序の使用方法について説明します。顧客からの注文の情報を格納する2つの表orders_tab(マスター表)およびline_items_tab(ディテール表)で部分的に構成される注文入力システムを想定します。order_seqという名前の順序が、次の文で定義されます。

```
CREATE SEQUENCE Order_seq
    START WITH 1
    INCREMENT BY 1
    NOMAXVALUE
    NOCYCLE
    CACHE 20;
```

- 順序の参照
  順序は、NEXTVALおよびCURRVAL疑似列を使用して、SQL文内で参照できます。それぞれの新しい順序番号は、順序の疑似列NEXTVALへの参照によって生成され、現行の順序番号は、疑似列CURRVALを使用して繰り返し参照できます。
- 順序番号のキャッシュ
  順序番号のキャッシュによりアクセス時間を改善できます。
- 順序をスケーラブルにする方法
  順序は、CREATE SEQUENCE文またはALTER SEQUENCE文にSCALE句を指定することでスケーラブルにすることができます。

親トピック: 順序の管理

24.2.4.1 順序の参照

順序は、NEXTVALおよびCURRVAL疑似列を使用して、SQL文内で参照できます。それぞれの新しい順序番号は、順序の疑似列NEXTVALへの参照によって生成され、現行の順序番号は、疑似列CURRVALを使用して繰り返し参照できます。

NEXTVALおよびCURRVALは、予約語またはキーワードではなく、SELECT、INSERTまたはUPDATEのようなSQL文の中で疑似列名として使用できます。

- NEXTVALを使用した順序番号の生成
  順序番号を生成して使用するには、SQL文でseq_name.NEXTVALを参照します。
- CURRVALを使用した順序番号の使用
  自分のセッションの現在の順序値を使用または参照するには、SQL文でseq_name.CURRVALを参照します。
- NEXTVALおよびCURRVALの使用と制限事項
  CURRVALおよびNEXTVALは特定の場所で使用可能で、その使用には制限が適用されます。

親トピック: 順序の使用

24.2.4.1.1 NEXTVALを使用した順序番号の生成

順序番号を生成して使用するには、SQL文でseq_name.NEXTVALを参照します。

たとえば、顧客からの発注を受けると想定します。順序番号は、値のリストの中で参照できます。たとえば:

```
INSERT INTO Orders_tab (Orderno, Custno)
    VALUES (Order_seq.NEXTVAL, 1032);
```

または、順序番号は、UPDATE文のSET句の中で参照できます。たとえば:

```
UPDATE Orders_tab
    SET Orderno = Order_seq.NEXTVAL
    WHERE Orderno = 10112;
```

順序番号は、問合せまたは副問合せのSELECTの最も外側でも参照できます。たとえば:

```
SELECT Order_seq.NEXTVAL FROM dual;
```

定義されているように、order_seq.NEXTVALへの最初の参照が値1を返します。order_seq.NEXTVALを参照する後続の各文は、次の順序番号(2、3、4、. . .)を生成します。疑似列NEXTVALは、必要に応じた数の新しい順序番号を生成するために使用されます。ただし、各行に生成される順序番号は1つのみです。つまり、1つの文でNEXTVALが複数回参照される場合、最初の参照によって次の番号が生成され、文の中のすべての後続の参照によって同じ番号が返されます。

生成された順序番号は、その番号を生成したセッションに対してのみ使用可能です。トランザクションのコミットまたはロールバック

には関係なく、order_seq.NEXTVALを参照する他のユーザーは一意の値を取得します。2人のユーザーが同時に同じ順序にアクセスしている場合、他方のユーザーも順序番号を生成しているため、各ユーザーが受け取る順序番号に食い違いが生じる可能性があります。

親トピック: 順序の参照

24.2.4.1.2 CURRVALを使用した順序番号の使用

自分のセッションの現在の順序値を使用または参照するには、SQL文でseq_name.CURRVALを参照します。

現行のユーザー・セッション(現行または既存のトランザクション内)でseq_name.NEXTVALが参照された場合のみ、CURRVALが使用されます。CURRVALは、同一文内で複数回の場合を含め、必要な回数だけ参照できます。次の順序番号は、NEXTVALが参照されると生成されます。前述の例を続けると、注文の明細項目を挿入することで、顧客からの発注を完了します。

```
INSERT INTO Line_items_tab (Orderno, Partno, Quantity)
    VALUES (Order_seq.CURRVAL, 20321, 3);
INSERT INTO Line_items_tab (Orderno, Partno, Quantity)
    VALUES (Order_seq.CURRVAL, 29374, 1);
```

前項で指定したINSERT文が、新しい順序番号347を生成した場合、この項の文で挿入された両方の行は注文番号347の行を挿入します。

親トピック: 順序の参照

24.2.4.1.3 NEXTVALおよびCURRVALの使用と制限事項

CURRVALおよびNEXTVALは特定の場所で使用可能で、その使用には制限が適用されます。

CURRVALおよびNEXTVALは、次の場所で使用できます。

- INSERT文のVALUES句
- SELECT文のSELECTリスト
- ビューの問合せまたはマテリアライズド・ビューの問合せ

  ただし、マテリアライズド・ビューにCURRVALおよびNEXTVALを使用すると、マテリアライズド・ビューが複雑になります。このため、高速リフレッシュできません。
- UPDATE文のSET句

CURRVALおよびNEXTVALは、次の場所では使用できません。

- 副問合せ
- DISTINCT演算子を指定したSELECT文
- GROUP BYまたはORDER BY句を指定したSELECT文
- UNION、INTERSECTまたはMINUS集合演算子を指定した別のSELECT文と組み合せたSELECT文
- SELECT文のWHERE句
- CHECK制約の条件

親トピック: 順序の参照

24.2.4.2 順序番号のキャッシュ

順序番号のキャッシュによりアクセス時間を改善できます。

- 順序番号のキャッシュについて
  順序番号は、システム・グローバル領域(SGA)内の順序キャッシュに保持できます。順序キャッシュ内の順序番号へのアクセスは、順序番号をディスクから読み込むより高速です。
- 順序キャッシュの自動サイズ変更について
  順序キャッシュの自動サイズ変更により、順序を使用する高速挿入ワークロードのパフォーマンスが大幅に向上します。
- 順序キャッシュ内のエントリの数
  アプリケーションが順序キャッシュ内の順序にアクセスする場合、順序番号は高速に読み込まれます。ただし、アプリケーションがキャッシュ内にない順序にアクセスする場合は、順序番号が使用される前に、順序がディスクからキャッシュに読み込まれる必要があります。
- 各順序キャッシュ・エントリ内の値の数
  順序が順序キャッシュに読み込まれるとき、キャッシュ・エントリに順序値が生成および格納されます。その後、これらの値に高速にアクセスできます。

親トピック: 順序の使用

24.2.4.2.1 順序番号のキャッシュについて

順序番号は、システム・グローバル領域(SGA)内の順序キャッシュに保持できます。順序キャッシュ内の順序番号へのアクセスは、順序番号をディスクから読み込むより高速です。

順序キャッシュは、複数のエントリで構成されています。各エントリには、単一の順序の順序番号を多数保持できます。

すべての順序番号に高速にアクセスするには、次のガイドラインに従ってください。

- 順序キャッシュが、アプリケーションによって同時に使用されるすべての順序を保持できるようにしてください。
- 順序キャッシュ内に保持された各順序の値の数を増やしてください。

親トピック: 順序番号のキャッシュ

24.2.4.2.2 順序キャッシュの自動サイズ変更について

順序キャッシュの自動サイズ変更により、順序を使用する高速挿入ワークロードのパフォーマンスが大幅に向上します。

各インスタンスの自動順序キャッシュ・サイズは、順序番号の使用率に基づいて動的に計算されます。各インスタンスは、手動で構成された順序キャッシュ・サイズの最大値と、次の10秒間の予測キャッシュ・サイズ要件をキャッシュします。順序キャッシュ・サイズは、順序の使用方法に基づいて縮小または拡大できます。順序キャッシュ・サイズが無制限に増加しないように、キャッシュ・サイズおよびキャッシュ・サイズの各増分は制限されます。

順序キャッシュの自動サイズ変更には、次の制限事項があります。

- Oracle Real Application Clusters (Oracle RAC)の順序付きシーケンスではサポートされていません。
- サイクル順序の場合、自動順序キャッシュ・サイズの上限は1サイクルのサイズです。

親トピック: 順序番号のキャッシュ

24.2.4.2.3 順序キャッシュ内のエントリの数

アプリケーションが順序キャッシュ内の順序にアクセスする場合、順序番号は高速に読み込まれます。ただし、アプリケーションがキャッシュ内にない順序にアクセスする場合は、順序番号が使用される前に、順序がディスクからキャッシュに読み込まれる必要があります。

アプリケーションが多数の順序を同時に使用する場合は、順序キャッシュの大きさが不足して、すべての順序を保持できないことがあります。このような場合、順序番号へのアクセスにはディスク読取りが頻繁に必要になります。すべての順序に高速にアクセスするには、キャッシュに十分なエントリを用意し、アプリケーションが同時に使用するすべての順序を保持してください。

親トピック: 順序番号のキャッシュ

24.2.4.2.4 各順序キャッシュ・エントリ内の値の数

順序が順序キャッシュに読み込まれるとき、キャッシュ・エントリに順序値が生成および格納されます。その後、これらの値に高速にアクセスできます。

キャッシュに格納される順序値の数は、CREATE SEQUENCE文のCACHEパラメータによって決まります。このパラメータのデフォルト値は20です。

次のCREATE SEQUENCE文はseq2順序を作成し、SEQUENCEキャッシュ内に50個の順序値を格納します。

```
CREATE SEQUENCE seq2
    CACHE 50;
```

次に、seq2の最初の50個の値がキャッシュから読み込まれます。51番目の値がアクセスされると、次の50個の値がディスクから読み込まれます。

CACHEに上限を選択することにより、ディスクから順序キャッシュへの読取りを減らし、より長く連続する順序番号にアクセスできます。ただし、インスタンス障害が発生した場合は、キャッシュ内のすべての順序値が失われます。エクスポートの実行中にトランザクションが順序番号へのアクセスを続ける場合、エクスポートとインポートの後で、キャッシュされた順序番号がスキップされることもあります。

CREATE SEQUENCE文の中でNOCACHEオプションを使用する場合、順序値は順序キャッシュに格納されません。この場合は、順序にアクセスするたびにディスク読取りが必要となります。このようなディスク読取りにより、順序へのアクセスが遅くなります。次のCREATE SEQUENCE文はSEQ3順序を作成しますが、その値をキャッシュ内に格納しません。

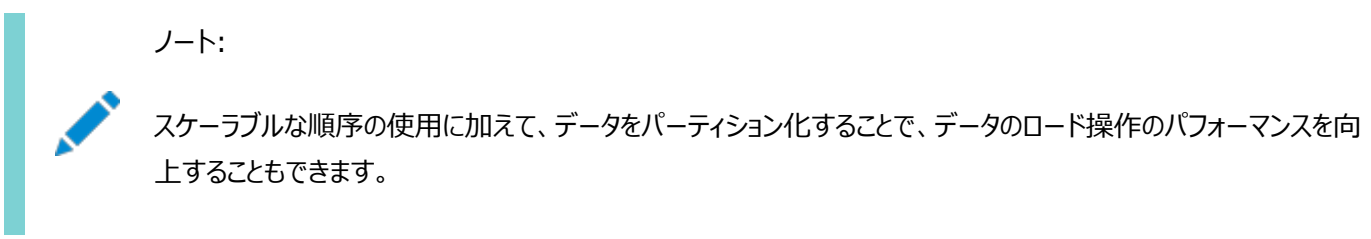
```
CREATE SEQUENCE seq3
    NOCACHE;
```

親トピック: 順序番号のキャッシュ

24.2.4.3 順序をスケーラブルにする方法

順序は、CREATE SEQUENCE文またはALTER SEQUENCE文にSCALE句を指定することでスケーラブルにすることができます。

スケーラブルな順序は、並行性が高レベルになるデータ収集ワークロードのために、順序付けされていない主キーまたは一意キーの生成に使用すると特に効果的です。この機能は、単一Oracleデータベース・インスタンスやOracle RACデータベースにとってメリットがあります。スケーラブルな順序により、順序と索引ブロックの競合が大幅に減少します。また、CREATE SEQUENCE文またはALTER SEQUENCE文のCACHE句を使用して非常に巨大な順序キャッシュを構成するというソリューションと比べて、優れたデータのロードのスケーラビリティが提供されます。



ノート:

スケーラブルな順序の使用に加えて、データをパーティション化することで、データのロード操作のパフォーマンスを向上することもできます。

スケーラブルな順序を定義する構文は次のとおりです。

```
CREATE | ALTER SEQUENCE sequence_name
   ...
   SCALE [EXTEND | NOEXTEND] | NOSCALE
   ...
```

SCALE句が指定されていると、6桁の数値のスケーラブルな順序のオフセット番号が順序の桁の前に追加されます。

```
   scalable sequence number = 6 digit scalable sequence offset number || normal
sequence number
```

各要素の意味は次のとおりです。

- ||は連結演算子です。
- 6桁のスケーラブルな順序のオフセット番号= 3桁のインスタンス・オフセット番号|| 3桁のセッション・オフセット番号。

  3桁のインスタンス・オフセット番号は、[(instance id % 100) + 100]として生成されます。3桁のセッション・オフセット番号は、[session id % 1000]として生成されます。

さらに、SCALE句には、EXTENDオプションまたはNOEXTENDオプションを指定することもできます。

- EXTENDオプション

  SCALE句にEXTENDオプションを指定すると、スケーラブルな順序の値は、[X桁+ Y桁]になります。Xは、スケーラブルな順序のオフセット番号の桁数です(デフォルトは6桁)。Yは、MAXVALUE句で指定された桁数です。

  たとえば、MINVALUEが1、MAXVALUEが100 (3桁)の昇順のスケーラブルな順序にEXTENDオプションを指定すると、スケーラブルな順序値は、9桁(6桁のスケーラブルな順序のオフセット番号+ 3桁のMAXVALUE)になり、その形式は次のようになります。

  ```
     6 digit scalable sequence offset number || 001
     6 digit scalable sequence offset number || 002
     6 digit scalable sequence offset number || 003
     ...
     6 digit scalable sequence offset number || 100
  ```

- NOEXTENDオプション

  SCALE句にNOEXTENDオプション(デフォルトのオプション)を指定すると、スケーラブルな順序の桁数はMAXVALUE句で指定した桁数を超えることができなくなります。

  たとえば、MINVALUEが1、MAXVALUEが1000000 (7桁)の昇順のスケーラブルな順序にNOEXTENDオプションを指定すると、1000000のMAXVALUEは7桁になるため、スケーラブルな順序の値は7桁になり、その形式は次のようになります。

  ```
     6 digit scalable sequence offset number || 1
     6 digit scalable sequence offset number || 2
     6 digit scalable sequence offset number || 3
     ...
     6 digit scalable sequence offset number || 9
  ```

  このスケーラブルな順序に対して、[6 digit scalable sequence offset number || 9]の順序値の後にNEXTVAL操作を実行すると、次のエラー・メッセージが報告されます。これは、その次の順序値が[6 digit scalable sequence offset number || 10]になり、その順序値は8桁で、1000000 (7桁)のMAXVALUEよりも桁数が大きくなるためです。

  ```
  ORA-64603: NEXTVAL cannot be instantiated for SQ. Widen the sequence by 1
  digits or alter sequence with SCALE EXTEND.
  ```

ノート:

NOEXTEND オプションは、固定幅の列の移入に順序が使用される既存のアプリケーションと統合する際に役立ちます。

既存のスケーラブルな順序をスケーラブルではない順序に変換するには、ALTER SEQUENCE文でNOSCALE句を使用します。

ノート:

スケーラブルな順序の番号はグローバルに順序付けされないため、スケーラブルな順序には順序付けを指定しないことをお薦めします。

順序がスケーラブルかどうか、またはスケーラブルな順序が拡張可能かどうかを確認するには、DBA_SEQUENCESビュー、USER_SEQUENCESビューおよびALL_SEQUENCESビューで次の列の値を確認します。

表24-1 DBA_SEQUENCESビュー、SEQUENCESビューおよびALL_SEQUENCESビューのスケーラブルな順序に関連する列

列名	説明
SCALE_FLAG	順序がスケーラブルな順序であるかどうかを示します。 ● Y ● N
EXTEND_FLAG	スケーラブルな順序が拡張可能かどうかを示します。つまり、MAXVALUE に指定された値を超えて順序値が拡張できるように、スケーラブルな順序に EXTEND オプションが適用されているかどうかを示します。 ● Y ● N

親トピック: 順序の使用

24.2.5 順序の削除

不要になった順序は、DROP SEQUENCE文を使用して削除できます。

自分のスキーマ内の順序はどれでも削除できます。別のスキーマ内の順序を削除するには、DROP ANY SEQUENCEシステム権限が必要です。たとえば、次の文はorder_seq順序を削除します。

```
DROP SEQUENCE order_seq;
```

順序を削除すると、その定義がデータ・ディクショナリから削除されます。順序のシノニムはそのまま残りますが、参照時にエラーが返されます。

関連項目:

DROP SEQUENCE文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: 順序の管理

24.3 シノニムの管理

シノニムの作成、シノニムの使用およびシノニムの削除などのタスクを実行できます。

- シノニムについて
  シノニムとはスキーマ・オブジェクトの別名です。
- シノニムの作成
  シノニムを作成するには、CREATE SYNONYM文を使用します。
- DML文でのシノニムの使用
  シノニムは、そのシノニムの基礎となるオブジェクトを参照するのと同じ方法で、DML文で参照できます。
- シノニムの削除
  不要になったシノニムを削除するには、DROP SYNONYM文を使用します。プライベート・シノニムを削除する場合は、PUBLICキーワードを省略します。パブリック・シノニムを削除する場合は、PUBLICキーワードを指定します。

親トピック: ビュー、順序およびシノニムの管理

24.3.1 シノニムについて

シノニムは、スキーマ・オブジェクトの別名です。

シノニムは、オブジェクトの名前および所有者をマスキングし、分散データベースのリモート・オブジェクトに対する位置の透過性を提供することで、一定レベルのセキュリティを提供できます。また、データベース・ユーザーにとって、シノニムは使いやすく、SQL文の複雑さが軽減されます。

シノニムを使用することで、基礎になるオブジェクトの名前変更または移動が可能になります。その場合、シノニムの再定義のみで、そのシノニムに基づくアプリケーションは変更しなくてもそのまま機能します。

パブリック・シノニムとプライベート・シノニムの両方を作成できます。パブリック・シノニムはPUBLICという名前の特別なユーザー・グループによって所有され、データベース内のすべてのユーザーがアクセスできます。プライベート・シノニムは、特定のユーザーのスキーマ内に含まれており、そのユーザーおよび基礎になるオブジェクトの権限受領者のみが使用できます。

シノニム自体は安全ではありません。シノニムのオブジェクト権限を付与した場合、実際には基礎になるオブジェクトの権限を付与することになり、そのシノニムはGRANT文でオブジェクトの別名としてのみ機能します。

関連項目:

シノニムの詳細は、『Oracle Database概要』を参照してください。

親トピック: シノニムの管理

24.3.2 シノニムの作成

シノニムを作成するには、CREATE SYNONYM文を使用します。

自分のスキーマに新しいプライベート・シノニムを作成するには、CREATE SYNONYMシステム権限が必要です。別のユーザーのスキーマ内にプライベート・シノニムを作成するには、CREATE ANY SYNONYM権限が必要です。パブリック・シノニムを作成するには、CREATE PUBLIC SYNONYMシステム権限が必要です。

シノニムを作成するときには、CREATE SYNONYM文を正常に実行するために、基礎になるスキーマ・オブジェクトは必要なく、また、そのオブジェクトにアクセスする権限も不要です。次の文は、jwardのスキーマに含まれるemp表のパブリック・シノニムpublic_empを作成します。

```
CREATE PUBLIC SYNONYM public_emp FOR jward.emp
```

リモート・プロシージャまたはファンクションのシノニムを作成する場合は、そのリモート・プロシージャまたはファンクションのスキーマ名でリモート・オブジェクトを修飾する必要があります。また、リモート・オブジェクトが存在するデータベースに、ローカル・パブリック・シノニムを作成できますが、この場合は、プロシージャまたはファンクションへの後続のすべてのコールに、そのデータベース・リンクを含める必要があります。

関連項目:

CREATE SYNONYM文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: シノニムの管理

24.3.3 DML文でのシノニムの使用

シノニムは、そのシノニムの基礎になるオブジェクトを参照するのと同じ方法で、DML文で参照できます。

基礎になるオブジェクトへのアクセスに必要な権限が、明示的に、使用可能なロールから、またはPUBLICから付与されている場合は、自分のスキーマまたはパブリック・シノニムに含まれるプライベート・シノニムを正常に使用できます。また、基礎になるオブジェクトについて必要なオブジェクト権限が付与されている場合は、別のスキーマ内のプライベート・シノニムを参照することもできます。

付与されているオブジェクト権限のみを使用して、別のユーザーのシノニムを参照できます。たとえば、jward.emp表に対するSELECT権限のみがあるときに、シノニムjward.employeeがjward.empに対して作成された場合は、jward.employeeシノニムの問合せはできますが、jward.employeeシノニムを使用した行の挿入はできません。

たとえば、employeeというシノニムが表またはビューを参照する場合は、次の文が有効になります。

```
INSERT INTO employee (empno, ename, job)
    VALUES (emp_sequence.NEXTVAL, 'SMITH', 'CLERK');
```

fire_empというシノニムがスタンドアロン・プロシージャまたはパッケージ・プロシージャを参照する場合は、それをコマンドで実行できます。

```
EXECUTE Fire_emp(7344);
```

親トピック: シノニムの管理

24.3.4 シノニムの削除

不要になったシノニムを削除するには、DROP SYNONYM文を使用します。プライベート・シノニムを削除する場合は、PUBLICキーワードを省略します。パブリック・シノニムを削除する場合は、PUBLICキーワードを指定します。

自分のスキーマ内のプライベート・シノニムはどれでも削除できます。別のユーザーのスキーマ内にあるプライベート・シノニムを削

除するには、DROP ANY SYNONYMシステム権限が必要です。パブリック・シノニムを削除するには、DROP PUBLIC SYNONYMシステム権限が必要です。

たとえば、次の文はプライベート・シノニムempを削除します。

```
DROP SYNONYM emp;
```

次の文は、パブリック・シノニムpublic_empを削除します。

```
DROP PUBLIC SYNONYM public_emp;
```

シノニムを削除すると、その定義がデータ・ディクショナリから削除されます。削除したシノニムを参照するオブジェクトはすべて残ります。ただし、それらのオブジェクトは無効(使用不可)になります。シノニムの削除が他のスキーマ・オブジェクトに与える影響の詳細は、「オブジェクト依存性の管理」を参照してください。

関連項目:

DROP SYNONYM文の構文と詳細は、『Oracle Database SQL言語リファレンス』を参照してください

親トピック: シノニムの管理

24.4 ビュー、順序およびシノニムのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューを問い合せて、ビュー、シノニムおよび順序に関する情報を取得できます。

次のビューには、ビュー、シノニムおよび順序に関する情報が表示されます。

ビュー	説明
DBA_VIEWS ALL_VIEWS USER_VIEWS	DBA ビューには、データベース内のすべてのビューが表示されます。ALL ビューは、現行ユーザーがアクセスできるビューのみに制限されます。USER ビューは、現行ユーザーが所有するビューのみに制限されます。
DBA_SYNONYMS ALL_SYNONYMS USER_SYNONYMS	これらのビューには、シノニムが表示されます。
DBA_SEQUENCES ALL_SEQUENCES USER_SEQUENCES	これらのビューには、順序が表示されます。
DBA_UPDATABLE_COLUMNS	これらのビューには、更新可能な結合ビューの列がすべて表示されます。

ビュー	説明
ALL_UPDATABLE_COLUMNS	
USER_UPDATABLE_COLUMNS	

親トピック: ビュー、順序およびシノニムの管理

25 破損データの修復

データ・ブロックの破損を検出し、修正できます。

ノート:

DBMS_REPAIR パッケージについて詳しくない場合は、このパッケージに含まれる修復プロシージャを実行する際に、Oracle サポート・サービスのアナリストと共同で作業することをお薦めします。

- データ・ブロック破損を修復するオプション
  Oracle Databaseでは、データ・ブロックの破損を検出および修正するための様々な方法を提供しています。
- DBMS_REPAIRパッケージの内容
  DBMS_REPAIRパッケージには、データ破損修復プロシージャが含まれており、表および索引にある破損ブロックを検出して修復できます。
- DBMS_REPAIRパッケージの使用方法
  データ・ブロックの破損に対処するために、DBMS_REPAIRパッケージを使用できます。
- DBMS_REPAIRの例
  DBMS_REPAIRパッケージの使用方法の例を示します。

親トピック: スキーマ・オブジェクト

25.1 データ・ブロック破損を修復するオプション

Oracle Databaseには、データ・ブロックの破損を検出して修正するために、複数の方法が用意されています。

その1つは、破損の検出後にオブジェクトを削除して再作成することです。しかし、この方法が必ずしも可能とはかぎらず、またそれが望ましくない場合もあります。データ・ブロックの破損が行のサブセットにかぎられている場合は、破損した行を除くすべてのデータを選択して表を再作成する方法があります。

また、DBMS_REPAIRパッケージを使用してデータ・ブロック破損を管理する方法もあります。DBMS_REPAIRを使用すると、表と索引の破損ブロックを検出して修復できます。オブジェクトは、再作成または修復の試行中でも続けて使用できます。

Recovery Manager (RMAN)のコマンドRECOVER BLOCKを使用して、破損したデータ・ブロックまたはデータ・ブロックのセットを修復することもできます。

ノート:

データの損失を伴う破損の場合は、そのデータがデータベース・システム全体にどのように格納されているかを分析して理解する必要があります。修復の内容によっては、データを失ったり、論理的一貫性が損なわれる場合があります。このパッケージで提供される修復アプローチが特定の破損に対して適切かどうかを個々に判断する必要があります。

関連項目:

RMANのRECOVER BLOCKコマンドの詳細は、『Oracle Databaseバックアップおよびリカバリ・リファレンス』を参照してくださ

い。

親トピック: 破損データの修復

25.2 DBMS_REPAIRパッケージの内容

DBMS_REPAIRパッケージには、データ破損修復プロシージャが含まれており、表および索引にある破損ブロックを検出して修復できます。

- DBMS_REPAIRプロシージャ
  DBMS_REPAIRパッケージのプロシージャにより、破損したブロックを検出および修復できます。
- DBMS_REPAIRプロシージャに関する制限事項
  いくつかの制限事項がDBMS_REPAIRプロシージャに適用されます。

関連項目:

DBMS_REPAIRプロシージャの構文、制限事項および例外の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 破損データの修復

25.2.1 DBMS_REPAIRプロシージャ

DBMS_REPAIRパッケージのプロシージャにより、破損したブロックを検出および修復できます。

次の表は、DBMS_REPAIRパッケージに含まれているプロシージャの一覧を示します。

プロシージャ名	説明
ADMIN_TABLES	修復表および孤立キー表の管理機能(作成、削除、パージ)を提供します。 ノート: これらの表は常に SYS スキーマに作成されます。
CHECK_OBJECT	表または索引の破損を検出し、レポートします。
DUMP_ORPHAN_KEYS	破損データ・ブロック内の行を指す索引エントリをレポートします。
FIX_CORRUPT_BLOCKS	すでに CHECK_OBJECT プロシージャで破損ブロックとして識別されているブロックにソフトウェア破損を示すマークを付けます。
REBUILD_FREELISTS	オブジェクトの空きリストを再作成します。
SEGMENT_FIX_STATUS	セグメント領域管理が AUTO の場合に、ビットマップ・エントリの破損状態を修正する機能を提供します。
SKIP_CORRUPT_BLOCKS	このプロシージャを使用すると、表と索引のスキャン時に、破損マークが付いたブロックが無視されます。使用しない場合は、破損マークが付いたブロックが検出されたときに

プロシージャ名	説明
	エラーORA-01578 が返されます。

これらのプロシージャの詳細と使用例は、「DBMS_REPAIRの例」を参照してください。

親トピック: DBMS_REPAIRパッケージの内容

25.2.2 DBMS_REPAIRプロシージャに関する制限事項

いくつかの制限および制約事項がDBMS_REPAIRプロシージャに適用されます。

DBMS_REPAIRプロシージャには、次の制約があります。

- LOBデータ型、ネストした表およびVARRAYを含む表はサポートされますが、表外格納の列は無視されます。
- クラスタは、SKIP_CORRUPT_BLOCKSおよびREBUILD_FREELISTSプロシージャではサポートされますが、CHECK_OBJECTプロシージャではサポートされません。
- 索引構成表およびLOB索引はサポートされません。
- グローバル一時表はサポートされていません。
- DUMP_ORPHAN_KEYSプロシージャは、ビットマップ索引またはファンクション索引には機能しません。
- DUMP_ORPHAN_KEYSプロシージャで処理される最大キー長は、3,950バイトです。

親トピック: DBMS_REPAIRパッケージの内容

25.3 DBMS_REPAIRパッケージの使用方法

データ・ブロックの破損に対処するために、DBMS_REPAIRパッケージを使用できます。

- タスク1: 破損の検出とレポート
  最初のタスクでは、破損を検出しレポートします。レポートでは、ブロックに関する問題が明らかになるだけでなく、それに対応する修復ディレクティブも識別されます。
- タスク2: DBMS_REPAIRの使用に伴うコストと利点の評価
  DBMS_REPAIRを使用する前に、それを使用する利点と損失を比較検討する必要があります。また、破損オブジェクトの対応手段として使用可能な他のオプションも検討してください。
- タスク3: オブジェクトを使用可能にする
  DBMS_REPAIRでは、表および索引のスキャン時に破損が無視され、オブジェクトが使用可能になります。
- タスク4: 破損の修復および失われたデータの再作成
  オブジェクトが使用可能になったら、次の修復処置を実行します。

親トピック: 破損データの修復

25.3.1 タスク1: 破損の検出とレポート

最初のタスクでは、破損を検出しレポートします。レポートでは、ブロックに関する問題が明らかになるだけでなく、それに対応する修復ディレクティブも識別されます。

- 破損の検出とレポートについて
  破損を検出するには、いくつかの方法があります。

- DBMS_REPAIR: CHECK_OBJECTおよびADMIN_TABLESプロシージャの使用
  CHECK_OBJECTプロシージャは、指定されたオブジェクトのブロック破損をチェックし、レポートします。
  ADMIN_TABLESプロシージャは、破損の修正を円滑に行うために修復表を作成します。
- DB_VERIFY: オフライン・データベース・チェックの実行
  データ破損が発生した場合は、オフライン診断ユーティリティとしてDB_VERIFYを使用します。
- ANALYZE: 破損のレポート
  ANALYZE TABLE...VALIDATE STRUCTURE文は、分析するオブジェクトの構造の妥当性をチェックします。オブジェクトの構造内で破損が検出されると、エラー・メッセージが表示されます。この場合、オブジェクトを削除して作成しなおす必要があります。
- DB_BLOCK_CHECKING初期化パラメータ
  DB_BLOCK_CHECKING初期化パラメータをTRUEに設定すると、データベースのブロック・チェックを使用可能にすることができます。

親トピック: DBMS_REPAIRパッケージの使用方法

25.3.1.1 破損の検出とレポートについて

破損を検出するには、いくつかの方法があります。

表25-1に、異なる検出方法を示します。

表25-1 破損検出方法の比較

検出方法	説明
DBMS_REPAIR PL/SQL パッケージ	指定した表、パーティションまたは索引のブロック・チェックを実行します。修復表に結果を移入します。
DB_VERIFY ユーティリティ	オフライン・データベースでブロック・チェックを実行します。
ANALYZE TABLE SQL 文	VALIDATE STRUCTURE オプションを指定すると、索引、表またはクラスタの構造の整合性が ANALYZE TABLE 文によって検証され、表と索引が同期しているかどうかがチェックまたは検証されます。
DB_BLOCK_CHECKING 初期化パラメータ	DB_BLOCK_CHECKING=TRUE の場合は、実際に破損マークを付ける前に、破損ブロックが識別されます。チェックは、ブロックの変更時に実行されます。

親トピック: タスク1: 破損の検出とレポート

25.3.1.2 DBMS_REPAIR: CHECK_OBJECTおよびADMIN_TABLESプロシージャの使用

CHECK_OBJECTプロシージャは、指定されたオブジェクトのブロック破損をチェックしてレポートします。ADMIN_TABLESプロシージャは、破損の修正を円滑に行うために修復表を作成します。

CHECK_OBJECTプロシージャでは、索引と表に対するANALYZE...VALIDATE STRUCTURE文と同様に、索引とデータ・ブロックに対してブロック・チェックが実行されます。

CHECK_OBJECTでは、破損がレポートされるだけでなく、そのオブジェクトに対して後でFIX_CORRUPT_BLOCKSを実行した場合に行われる修正も識別されます。この情報は修復表への移入によって使用可能になるため、最初にADMIN_TABLESプ

ロシージャで修復表を作成しておく必要があります。

CHECK_OBJECTプロシージャを実行した後は、修復表の簡単な問合せによってそのオブジェクトの破損および修復ディレクティブが表示されます。この情報に基づいて、レポートされた問題に最も適切な対処方法を評価できます。

親トピック: タスク1: 破損の検出とレポート

25.3.1.3 DB_VERIFY: オフライン・データベース・チェックの実行

データ破損が発生した場合は、オフライン診断ユーティリティとしてDB_VERIFYを使用します。

関連項目:

DB_VERIFYの詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: タスク1: 破損の検出とレポート

25.3.1.4 ANALYZE: 破損のレポート

ANALYZE TABLE...VALIDATE STRUCTURE文は、分析するオブジェクトの構造の妥当性をチェックします。オブジェクトの構造内で破損が検出されると、エラー・メッセージが表示されます。この場合、オブジェクトを削除して作成しなおす必要があります。

ANALYZE TABLE文のCASCADE句を使用すると、1回の操作で、表とすべての索引の構造をチェックできます。この操作ではリソースを大量に消費する可能性があるため、軽量なチェックを実行するFASTオプションを使用できます。詳細は、「表、索引、クラスタおよびマテリアライズド・ビューの妥当性チェック」を参照してください。

関連項目:

- ANALYZE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: タスク1: 破損の検出とレポート

25.3.1.5 DB_BLOCK_CHECKING初期化パラメータ

DB_BLOCK_CHECKING初期化パラメータをTRUEに設定すると、データベースのブロック・チェックを使用可能にすることができます。

これにより、データ・ブロックおよび索引ブロックが変更された際には、必ずそのブロックの内部一貫性がチェックされます。DB_BLOCK_CHECKINGは、ALTER SYSTEM SET文で変更可能な動的パラメータです。システム表領域では、ブロック・チェックは常に使用可能になっています。

注意:

このパラメータでブロック・チェックを有効にする前に、データベース内の論理的な破損を検出して修復することをお薦めします。そうしないと、論理的な破損を含むブロックは、ブロック・チェックが有効になった後に「ソフト破損」としてマークされ、ブロックは DML 文によって変更されます。これにより、ORA-1578 エラーが発生し、ブロックは読取り不可になります。

関連項目:

DB_BLOCK_CHECKING初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

論理破損の検出および修復の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

親トピック: タスク1: 破損の検出とレポート

25.3.2 タスク2: DBMS_REPAIRの使用に伴うコストと利点の評価

DBMS_REPAIRを使用する前に、その利害得失を検討する必要があります。また、破損オブジェクトの対応手段として使用可能な他のオプションも検討してください。

次の質問に答えることから開始してください。

- 破損の範囲はどの程度ですか。

  破損の有無と修復アクションの要不要を判断するには、CHECK_OBJECTプロシージャを実行して修復表を問い合せます。

- ブロック破損の対応手段として使用可能な他のオプションがありますか。次のことを考慮してください。
  - 他のソースからのデータが使用可能な場合は、そのオブジェクトを削除し、再作成して再移入する。
  - CREATE TABLE...AS SELECT文を発行して、破損表から新しい表を作成する。
  - SELECT文から破損行を除外して、破損を無視する。
  - メディア・リカバリを実行します。
- DBMS_REPAIRを使用してオブジェクトを使用可能にした場合に、どのような論理的な破損や副作用が生じますか。それらの問題に対処できますか。そのためにはどんな作業が必要ですか。

  破損マークが付いたブロックの行にはアクセスできない場合があります。また、正常にアクセスできる行が含まれているブロックでも、破損マークが付いている場合があります。

  ブロックに破損マークが付いている場合は、参照整合性制約が壊れていることがあります。この場合は、制約を使用禁止にし、再び使用可能にすると、不整合がレポートされます。すべての問題を解決すれば、再び制約を使用できるようになります。

  表にトリガーが定義されている場合は、論理的な破損が生じることがあります。たとえば、行を再度挿入したときに、挿入トリガーが起動されるかどうかを確認します。これらの問題に対処するには、インストレーションでどのトリガーがどのように使用されているかを理解する必要があります。

  索引と表が同期化されていない場合は、DUMP_ORPHAN_KEYSプロシージャを実行して、破損データの再作成に役立つ情報をキーから取得します。次に、ALTER INDEX...REBUILD ONLINE文を発行し、表と索引を同期化します。

- 修復によってデータが失われる場合に、このデータを取り出すことができますか。

  データ・ブロックに破損マークが付いている場合は、索引からデータを取り出すことができます。この情報を取り出すには、DUMP_ORPHAN_KEYSプロシージャを利用します。

親トピック: DBMS_REPAIRパッケージの使用方法

25.3.3 タスク3: オブジェクトの使用可能化

DBMS_REPAIRを使用して表と索引のスキャン時に破損を無視することにより、オブジェクトを使用可能にします。

- 破損の修復: FIX_CORRUPT_BLOCKSおよびSKIP_CORRUPT_BLOCKSプロシージャの使用
  DBMS_REPAIRの機能の適用範囲外にある破損をスキップする環境を設定し、それによって破損オブジェクトを使用可能にできます。
- 破損ブロックをスキップする操作の意味
  破損ブロックをスキップすると、いくつかの状況では、問合せから異なる結果が返されることがあります。

親トピック: DBMS_REPAIRパッケージの使用方法

25.3.3.1 破損の修復: FIX_CORRUPT_BLOCKSおよびSKIP_CORRUPT_BLOCKSプロシージャの使用

DBMS_REPAIRの機能の適用範囲外にある破損をスキップする環境を設定し、それによって破損オブジェクトを使用可能にできます。

破損が、データ・ブロック内の不良行などのデータの損失を伴う場合は、FIX_CORRUPT_BLOCKSプロシージャによって、そのようなブロックすべてに破損マークが付けられます。次に、破損マークが付いたブロックをスキップするSKIP_CORRUPT_BLOCKSプロシージャを実行できます。SKIP_FLAGパラメータがプロシージャに設定されている場合は、破損マークが付いているすべてのブロックが、表と索引のスキャンでスキップされます。これはメディアとソフトウェアの両方の破損ブロックに適用されます。

親トピック: タスク3: オブジェクトの使用可能化

25.3.3.2 破損ブロックをスキップする操作の意味

破損したブロックをスキップすると、いくつかの状況では、問合せから異なる結果が返されることがあります。

索引と表が同期化されていない場合、ある問合せで索引のみをプローブし、後続の問合せで索引と表の両方をプローブするような状況下では、SET TRANSACTION READ ONLYトランザクションの一貫性が保たれないことがあります。表ブロックに破損マークが付いている場合、この2つの問合せは異なる結果を返すので、読取り専用トランザクションのルールに違反します。この場合の対処方法の1つとして、SET TRANSACTION READ ONLYトランザクション内で破損をスキップしないようにします。

これと同様の問題は、連鎖している行の選択時にも発生します。同じ行を問い合せても、破損にアクセスできる場合とできない場合があるため、異なった結果が生じます。

親トピック: タスク3: オブジェクトの使用可能化

25.3.4 タスク4: 破損の修復および失われたデータの再作成

オブジェクトを使用可能にした後で、次の修復アクティビティを実行します。

- DUMP_ORPHAN_KEYSプロシージャを使用したデータのリカバリ
  DUMP_ORPHAN_KEYSプロシージャは、破損データ・ブロック内の行を指す索引エントリをレポートします。この種の索引エントリがすべて、破損のキーとROWIDを格納する孤立キー表に挿入されます。
- SEGMENT_FIX_STATUSプロシージャを使用したセグメント・ビットマップの修正
  セグメントの空き領域の管理にビットマップが使用されている場合(SEGMENT SPACE MANAGEMENT AUTO)、SEGMENT_FIX_STATUSプロシージャを使用します。

親トピック: DBMS_REPAIRパッケージの使用方法

25.3.4.1 DUMP_ORPHAN_KEYSプロシージャを使用したデータのリカバリ

DUMP_ORPHAN_KEYSプロシージャは、破損データ・ブロック内の行を指す索引エントリをレポートします。この種の索引エントリがすべて、破損のキーとROWIDを格納する孤立キー表に挿入されます。

索引エントリ情報を取り出した後、ALTER INDEX...REBUILD ONLINE文を使用して索引を再作成できます。

親トピック: タスク4: 破損の修復および失われたデータの再作成

25.3.4.2 SEGMENT_FIX_STATUSプロシージャを使用したセグメント・ビットマップの修正

セグメントの空き領域の管理にビットマップが使用されている場合(SEGMENT SPACE MANAGEMENT AUTO)、SEGMENT_FIX_STATUSプロシージャを使用します。

このプロシージャは、対応するブロックの現在の内容に基づいてビットマップ・エントリの状態を再計算します。また、ビットマップ・エントリを特定の値に設定するように指定することもできます。通常は、状態が適切に再計算されるので、値を強制的に設定する必要はありません。

親トピック: タスク4: 破損の修復および失われたデータの再作成

25.4 DBMS_REPAIRの例

DBMS_REPAIRパッケージの使用方法の例を示します。

- 例: 修復表または孤立キー表の作成
  修復表は、破損に関する情報を提供します。孤立キー表は、破損した行を指す索引エントリに関する情報を提供します。
- 例: 破損の検出
  CHECK_OBJECTプロシージャによって破損を検出する例を示します。
- 例: 破損ブロックの修正
  FIX_CORRUPT_BLOCKSプロシージャによって破損ブロックを修正する例を示します。
- 例: 破損データ・ブロックを指す索引エントリの検索
  DUMP_ORPHAN_KEYSプロシージャを使用して破損データ・ブロックを指す索引エントリを検索する例を示します。
- 例: 破損ブロックのスキップ
  SKIP_CORRUPT_BLOCKSプロシージャを使用して破損ブロックをスキップする例を示します。

親トピック: 破損データの修復

25.4.1 例: 修復表または孤立キー表の作成

修復表は、破損に関する情報を提供します。孤立キー表は、破損した行を指す索引エントリに関する情報を提供します。

- 修復表または孤立キー表について
  ADMIN_TABLEプロシージャは、修復表または孤立キー表の作成、パージまたは削除に使用します。
- 例: 修復表の作成
  ADMIN_TABLESプロシージャを使用して修復表を作成する例を示します。
- 例: 孤立キー表の作成
  ADMIN_TABLESプロシージャを使用して孤立キー表を作成する例を示します。

親トピック: DBMS_REPAIRの例

25.4.1.1 修復表または孤立キー表について

ADMIN_TABLEプロシージャは、修復表または孤立キー表の作成、パージまたは削除に使用します。

修復表は、CHECK_OBJECTプロシージャによって検出された破損の内容と、FIX_CORRUPT_BLOCKSプロシージャを実行した場合にこれらの破損がどのように処理されるかを示す情報を提供します。また、FIX_CORRUPT_BLOCKSプロシージャの実行が必要かを判断する際にも使用されます。

孤立キー表は、DUMP_ORPHAN_KEYSプロシージャの実行時に使用され、破損行を指す索引エントリが格納されます。DUMP_ORPHAN_KEYSプロシージャは、そのアクティビティをロギングし、索引情報を使用可能な形にして、孤立キー表に移入します。

親トピック: 例: 修復表または孤立キー表の作成

25.4.1.2 例: 修復表の作成

ADMIN_TABLESプロシージャを使用して修復表を作成する例を示します。

次の例では、users表領域の修復表を作成しています。

```
BEGIN
  DBMS_REPAIR.ADMIN_TABLES (
     TABLE_NAME => 'REPAIR_TABLE',
     TABLE_TYPE => dbms_repair.repair_table,
     ACTION     => dbms_repair.create_action,
     TABLESPACE => 'USERS');
END;
/
```

修復表または孤立キー表それぞれについて、存在しなくなったオブジェクトに関連する行を除外するビューも作成されます。ビュー名は、修復表または孤立キー表の名前に対応しており、接頭辞DBA_が付いています(たとえば、DBA_REPAIR_TABLE、DBA_ORPHAN_KEY_TABLE)。

次の問合せでは、users表領域に作成された修復表が表示されます。

```
DESC REPAIR_TABLE
 Name                          Null?    Type
 ----------------------------- -------- --------------
 OBJECT_ID                     NOT NULL NUMBER
 TABLESPACE_ID                 NOT NULL NUMBER
 RELATIVE_FILE_ID              NOT NULL NUMBER
 BLOCK_ID                      NOT NULL NUMBER
 CORRUPT_TYPE                  NOT NULL NUMBER
 SCHEMA_NAME                   NOT NULL VARCHAR2(128)
 OBJECT_NAME                   NOT NULL VARCHAR2(128)
 BASEOBJECT_NAME                        VARCHAR2(128)
 PARTITION_NAME                         VARCHAR2(128)
 CORRUPT_DESCRIPTION                    VARCHAR2(2000)
 REPAIR_DESCRIPTION                     VARCHAR2(200)
 MARKED_CORRUPT                NOT NULL VARCHAR2(10)
 CHECK_TIMESTAMP               NOT NULL DATE
 FIX_TIMESTAMP                          DATE
 REFORMAT_TIMESTAMP                     DATE
```

親トピック: 例: 修復表または孤立キー表の作成

25.4.1.3 例: 孤立キー表の作成

ADMIN_TABLESプロシージャを使用して孤立キー表を作成する例を示します。

次の例は、users表領域の孤立キー表の作成方法を示しています。

```
BEGIN
  DBMS_REPAIR.ADMIN_TABLES (
     TABLE_NAME => 'ORPHAN_KEY_TABLE',
     TABLE_TYPE => dbms_repair.orphan_table,
     ACTION     => dbms_repair.create_action,
     TABLESPACE => 'USERS');
END;
/
```

次の問合せでは、孤立キー表の定義を表示しています。

```
DESC ORPHAN_KEY_TABLE
 Name                           Null?    Type
 ------------------------------ -------- -----------------
 SCHEMA_NAME                    NOT NULL VARCHAR2(128)
 INDEX_NAME                     NOT NULL VARCHAR2(128)
 IPART_NAME                              VARCHAR2(128)
 INDEX_ID                       NOT NULL NUMBER
 TABLE_NAME                     NOT NULL VARCHAR2(128)
 PART_NAME                               VARCHAR2(128)
 TABLE_ID                       NOT NULL NUMBER
 KEYROWID                       NOT NULL ROWID
 KEY                            NOT NULL ROWID
 DUMP_TIMESTAMP                 NOT NULL DATE
```

親トピック: 例: 修復表または孤立キー表の作成

25.4.2 例: 破損の検出

CHECK_OBJECTプロシージャによって破損を検出する例を示します。

CHECK_OBJECTプロシージャは、指定されたオブジェクトをチェックし、破損および修復ディレクティブに関する情報を修復表に移入します。オブジェクトの一部をチェックする場合は、必要に応じて、範囲、パーティション名またはサブパーティション名を指定できます。

妥当性チェックでは、オブジェクト内部でそれまでに破損マークが付けられていないブロックがすべてチェックされます。ブロックごとに、トランザクションおよびデータ・レイヤー部分の自己整合性がチェックされます。CHECK_OBJECTの実行中に、破損バッファ・キャッシュ・ヘッダーを持つブロックが検出されると、そのブロックはスキップされます。

scott.dept表に対するCHECK_OBJECTプロシージャの実行例を次に示します。

```
SET SERVEROUTPUT ON
DECLARE num_corrupt INT;
BEGIN
 num_corrupt := 0;
 DBMS_REPAIR.CHECK_OBJECT (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'DEPT',
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     CORRUPT_COUNT =>  num_corrupt);
 DBMS_OUTPUT.PUT_LINE('number corrupt: ' || TO_CHAR (num_corrupt));
END;
/
```

SQL*Plusには、1つの破損を示す次の行が出力されます。

```
number corrupt: 1
```

修復表を問い合せると、破損の説明および修復アクションに関する提案を含む情報が表示されます。

```
SELECT OBJECT_NAME, BLOCK_ID, CORRUPT_TYPE, MARKED_CORRUPT,
       CORRUPT_DESCRIPTION, REPAIR_DESCRIPTION
```

```
     FROM REPAIR_TABLE;
OBJECT_NAME                    BLOCK_ID CORRUPT_TYPE MARKED_COR
------------------------------ ---------- ------------ ----------
CORRUPT_DESCRIPTION
--------------------------------------------------------------------------------
REPAIR_DESCRIPTION
--------------------------------------------------------------------------------
DEPT                                    3            1 FALSE
kdbchk: row locked by non-existent transaction
        table=0   slot=0
        lockid=32   ktbbhitc=1
mark block software corrupt
```

まだ破損ブロックに破損マークが付いていないため、ここで重要なデータを抽出します。ブロックに破損マークが付けられた後は、そのブロック全体がスキップされます。

親トピック: DBMS_REPAIRの例

25.4.3 例: 破損ブロックの修正

FIX_CORRUPT_BLOCKSプロシージャによって破損ブロックを修正する例を示します。

CHECK_OBJECTプロシージャによって生成した修復表内の情報に基づいて、FIX_CORRUPT_BLOCKSプロシージャを使用し、指定したオブジェクトの破損ブロックを修正します。ブロックの変更前には、そのブロックがまだ破損状態にあるかどうかが確認されます。破損ブロックは、そのブロックにソフトウェア破損のマークを付けることによって修復されます。修復が実行されると、修復表内の対応する行がタイムスタンプ付きで更新されます。

次の例では、CHECK_OBJECTプロシージャによってレポートされた表scott.deptの破損ブロックを修正しています。

```
SET SERVEROUTPUT ON
DECLARE num_fix INT;
BEGIN
 num_fix := 0;
 DBMS_REPAIR.FIX_CORRUPT_BLOCKS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME=> 'DEPT',
     OBJECT_TYPE => dbms_repair.table_object,
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     FIX_COUNT=> num_fix);
 DBMS_OUTPUT.PUT_LINE('num fix: ' || TO_CHAR(num_fix));
END;
/
```

SQL*Plusでは次の行が出力されます。

```
num fix: 1
```

次の問合せによって、修復が完了していることが確認されます。

```
SELECT OBJECT_NAME, BLOCK_ID, MARKED_CORRUPT
     FROM REPAIR_TABLE;
OBJECT_NAME                    BLOCK_ID MARKED_COR
------------------------------ ---------- ----------
DEPT                                    3 TRUE
```

親トピック: DBMS_REPAIRの例

25.4.4 例: 破損データ・ブロックを指す索引エントリの検索

DUMP_ORPHAN_KEYSプロシージャを使用して破損データ・ブロックを指す索引エントリを検索する例を示します。

DUMP_ORPHAN_KEYSプロシージャは、破損データ・ブロック内の行を指す索引エントリをレポートします。索引エントリごとに、指定した孤立キー表に1行ずつ挿入されます。この孤立キー表は、事前に作成しておく必要があります。

この情報は、表内の失われた行を再作成する場合や診断に使用します。

ノート:

このプロシージャは、修復表で識別された表に対応付けられている索引ごとに実行する必要があります。

この例では、pk_deptはscott.dept表の索引です。これがスキャンされて、破損したデータ・ブロックの行を指す索引エントリがあるかどうかが見極められます。

```
SET SERVEROUTPUT ON
DECLARE num_orphans INT;
BEGIN
 num_orphans := 0;
 DBMS_REPAIR.DUMP_ORPHAN_KEYS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'PK_DEPT',
     OBJECT_TYPE => dbms_repair.index_object,
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     ORPHAN_TABLE_NAME=> 'ORPHAN_KEY_TABLE',
     KEY_COUNT => num_orphans);
 DBMS_OUTPUT.PUT_LINE('orphan key count: ' || TO_CHAR(num_orphans));
END;
/
```

次の出力は、孤立キーが3つあることを示しています。

```
orphan key count: 3
```

孤立キー表の索引エントリは、索引の再作成が必要であることを示しています。索引の再作成により、表プローブと索引プローブが同じ結果セットを返すことが保証されます。

親トピック: DBMS_REPAIRの例

25.4.5 例: 破損ブロックのスキップ

SKIP_CORRUPT_BLOCKSプロシージャを使用して破損ブロックをスキップする例を示します。

SKIP_CORRUPT_BLOCKSプロシージャは、指定されたオブジェクトの索引および表のスキャン時に破損ブロックをスキップするかしないかを指定します。オブジェクトが表であれば、スキップは表とその索引に適用されます。オブジェクトがクラスタのときは、クラスタ内のすべての表およびその各索引に適用されます。

次の例では、scott.dept表のソフトウェア破損ブロックのスキップを可能に設定しています。

```
BEGIN
  DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'DEPT',
     OBJECT_TYPE => dbms_repair.table_object,
     FLAGS => dbms_repair.skip_flag);
END;
/
```

DBA_TABLESビューを使用してscottの表を問い合せると、表scott.deptのSKIP_CORRUPTが使用可能になっていることが示されます。

```
SELECT OWNER, TABLE_NAME, SKIP_CORRUPT FROM DBA_TABLES
    WHERE OWNER = 'SCOTT';
OWNER                          TABLE_NAME                     SKIP_COR
------------------------------ ------------------------------ --------
SCOTT                          ACCOUNT                        DISABLED
SCOTT                          BONUS                          DISABLED
SCOTT                          DEPT                           ENABLED
SCOTT                          DOCINDEX                       DISABLED
SCOTT                          EMP                            DISABLED
SCOTT                          RECEIPT                        DISABLED
SCOTT                          SALGRADE                       DISABLED
SCOTT                          SCOTT_EMP                      DISABLED
SCOTT                          SYS_IOT_OVER_12255             DISABLED
SCOTT                          WORK_AREA                      DISABLED
10 rows selected.
```

親トピック: DBMS_REPAIRの例

第IV部 データベース・リソースの管理とタスクのスケジューリング

自動データベース・メンテナンス・タスク、データベース・リソースおよびタスク・スケジュールを管理できます。

- 自動データベース・メンテナンス・タスクの管理
  Oracle Databaseでは、データベース管理者が通常実行する一般的なメンテナンス・タスクのうちのいくつかが自動化されました。これらの自動化メンテナンス・タスクは、システムの負荷が軽いと予測される時間帯に実行されます。管理者は、個々のメンテナンス・タスクを有効または無効にしたり、これらのタスクの実行時期やタスクに対するリソース割当てを構成できます。
- Oracle Database Resource Managerを使用したリソースの管理
  Oracle Database Resource Manager (リソース・マネージャ)により、データベース・リソースの割当てを管理できます。
- Oracle Schedulerの概要
  Oracle Schedulerによりタスクをスケジュールできます。
- Oracle Schedulerを使用したジョブのスケジューリング
  Oracle Schedulerでジョブを作成、実行および管理できます。
- Oracle Schedulerの管理
  Oracle Schedulerを構成、管理、監視およびトラブルシューティングできます。

26 自動データベース・メンテナンス・タスクの管理

Oracle Databaseでは、データベース管理者が通常実行する一般的なメンテナンス・タスクのうちのいくつかが自動化されました。これらの自動化メンテナンス・タスクは、システムの負荷が軽いと予測される時間帯に実行されます。管理者は、個々のメンテナンス・タスクを有効または無効にしたり、これらのタスクの実行時期やタスクに対するリソース割当てを構成できます。

ノート:

この章では、PL/SQL パッケージを使用して、自動化メンテナンス・タスクを管理する方法について説明します。簡単な方法として、Oracle Enterprise Manager Cloud Control (Cloud Control)のグラフィカル・インタフェースを使用する方法があります。

Cloud Control を使用して自動メンテナンス・タスクを管理するには:

1. データベース・ホームページにアクセスします。
2. 「管理」メニューから、「Oracle Scheduler」を選択し、「自動化メンテナンス・タスク」を選択します。
3. 「自動化メンテナンス・タスク」ページで、「構成」をクリックします。

- 自動化メンテナンス・タスクについて
  自動化メンテナンス・タスクとは、データベースのメンテナンス操作を実行するために、一定の間隔をおいて自動的に開始されるタスクです。問合せオプティマイザのスキーマ・オブジェクトに関する統計を収集するタスクなどはその一例です。
- メンテナンス・ウィンドウについて
  メンテナンス・ウィンドウは、自動化メンテナンス・タスクが実行される連続した時間間隔です。メンテナンス・ウィンドウは、MAINTENANCE_WINDOW_GROUPという名前のウィンドウ・グループに属するOracle Schedulerウィンドウです。
- 自動化メンテナンス・タスクの構成
  メンテナンス・ウィンドウのサブセットで特定のメンテナンス・タスクを有効または無効にするには、DBMS_AUTO_TASK_ADMIN PL/SQLパッケージを使用します。
- メンテナンス・ウィンドウの構成
  事前定義のメンテナンス・ウィンドウをデータベース環境に適した時間に調整したり、新しいメンテナンス・ウィンドウを作成できます。メンテナンス・ウィンドウは、DBMS_SCHEDULER PL/SQLパッケージを使用してカスタマイズできます。
- 自動化メンテナンス・タスクに対するリソース割当ての構成
  自動化メンテナンス・タスクに対するリソース割当てを増減できます。
- 自動化メンテナンス・タスクの参照情報
  Oracle Databaseには事前定義のメンテナンス・ウィンドウが用意されています。また、データ・ディクショナリ・ビューもあるため、自動化メンテナンスに関する情報を問い合せることができます。

親トピック: データベース・リソースの管理とタスクのスケジューリング

26.1 自動化メンテナンス・タスクについて

自動化メンテナンス・タスクとは、データベースのメンテナンス操作を実行するために、一定の間隔をおいて自動的に開始されるタスクです。問合せオプティマイザのスキーマ・オブジェクトに関する統計を収集するタスクなどはその一例です。

自動化メンテナンス・タスクはメンテナンス・ウィンドウで実行されますが、これはシステム負荷が低い時間に発生するように設定された事前定義済の時間間隔です。管理者は、データベースのリソース使用パターンに基づいてメンテナンス・ウィンドウをカスタマイズしたり、特定のデフォルト・ウィンドウでの実行を禁止できます。独自のメンテナンス・ウィンドウを作成することもできます。

Oracle Databaseでは、次の自動化メンテナンス・タスクが事前定義されています。

- 自動オプティマイザ統計収集—データベース内に統計がないか、古い統計のみがあるすべてのスキーマ・オブジェクトに関するオプティマイザ統計を収集します。このタスクで収集された統計は、SQLの実行パフォーマンスを改善するためにSQL問合せオプティマイザによって使用されます。

  関連項目:

  統計の自動収集の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

- オプティマイザ統計アドバイザー—統計収集の方法を分析し、統計収集を微調整するための変更内容を提案します。

  関連項目:

  Oracle Database SQLチューニング・ガイド

- 自動セグメント・アドバイザー—再生可能な領域が存在しているセグメントを識別し、それらのセグメントの断片化を解消する方法について推奨事項を生成します。

  セグメント・アドバイザを手動で実行すると、最新の推奨事項を取得したり、自動セグメント・アドバイザで再生可能な領域を調査しなかったセグメントについて推奨事項を取得することもできます。

  関連項目:

  詳細は、「セグメント・アドバイザの使用」を参照してください。

- 自動SQLチューニング・アドバイザー—高負荷のSQL文のパフォーマンスを調査し、それらの文のチューニング方法について推奨事項を生成します。このアドバイザは、SQLプロファイルの推奨事項を自動的に実装するように構成できます。

  関連項目:

  SQLチューニング・アドバイザの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

- SQL計画管理(SPM)展開アドバイザー—SQL計画ベースラインに最近追加された計画を展開します。アドバイザを使用することにより手動で実行する必要がなくなるため、計画の展開が容易になります。

  関連項目:

  SPM展開アドバイザの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

デフォルトでは、これらの自動化メンテナンス・タスクは、すべてのメンテナンス・ウィンドウで実行するように構成されています。

親トピック: 自動データベース・メンテナンス・タスクの管理

26.2 メンテナンス・ウィンドウについて

メンテナンス・ウィンドウは、自動化メンテナンス・タスクが実行される連続した時間間隔です。メンテナンス・ウィンドウは、MAINTENANCE_WINDOW_GROUPという名前のウィンドウ・グループに属するOracle Schedulerウィンドウです。

スケジューラのウィンドウは、単純な繰返し間隔(毎週土曜の午前0時から午前6時の間など)の場合や複雑な間隔(会社の休業日を除いて、毎月最後の稼働日の午前0時から午前6時の間など)の場合があります。

メンテナンス・ウィンドウがオープンすると、Oracle Databaseでは、そのウィンドウでの実行がスケジュールされている各メンテナンス・タスクに対してOracle Schedulerのジョブが作成されます。各ジョブには、実行時に生成されるジョブ名が割り当てられます。自動化メンテナンス・タスクのジョブ名はすべてORA$ATで始まります。たとえば、自動セグメント・アドバイザのジョブには、ORA$AT_SA_SPC_SY_26という名前が指定されます。自動化メンテナンス・タスクのジョブが終了すると、そのジョブはOracle Schedulerのジョブ・システムから削除されます。ただし、スケジューラのジョブ履歴には記録が残ります。

ノート:

ジョブ履歴を表示するには、SYS ユーザーでログインする必要があります。

非常に長いメンテナンス・ウィンドウの場合、自動SQLチューニング・アドバイザを除くすべての自動化メンテナンス・タスクは4時間ごとに再起動されます。この機能によって、ウィンドウ・サイズに関係なく、メンテナンス・タスクが定期的に実行されます。

自動化メンテナンス・タスクのフレームワークは、データベースに定義されているメンテナンス・ウィンドウに依存しています。表26-1に、新規Oracle Databaseの各インストールで自動的に定義されるメンテナンス・ウィンドウのリストを示します。

関連項目:

- ウィンドウとグループの詳細は、「ジョブおよびスケジューラ・オブジェクトのサポートについて」を参照してください。

親トピック: 自動データベース・メンテナンス・タスクの管理

26.3 自動化メンテナンス・タスクの構成

メンテナンス・ウィンドウのサブセットで特定のメンテナンス・タスクを有効または無効にするには、DBMS_AUTO_TASK_ADMIN PL/SQLパッケージを使用します。

- すべてのメンテナンス・ウィンドウに対するメンテナンス・タスクの有効化と無効化
  1回の操作ですべてのメンテナンス・ウィンドウに対して特定の自動化メンテナンス・タスクを無効または有効にできます。
- 特定のメンテナンス・ウィンドウに対するメンテナンス・タスクの有効化と無効化
  デフォルトでは、すべての事前定義のメンテナンス・ウィンドウで、すべてのメンテナンス・タスクが実行されます。管理者は、特定のウィンドウに対してメンテナンス・タスクを無効にできます。

親トピック: 自動データベース・メンテナンス・タスクの管理

26.3.1 すべてのメンテナンス・ウィンドウに対するメンテナンス・タスクの有効化と無効化

1回の操作ですべてのメンテナンス・ウィンドウに対して特定の自動化メンテナンス・タスクを無効または有効にできます。

管理者は、1回の操作ですべてのメンテナンス・ウィンドウに対して特定の自動化メンテナンス・タスクを無効にできます。無効にするには、DBMS_AUTO_TASK_ADMIN PL/SQLパッケージのDISABLEプロシージャをwindow_name引数なしでコールし

ます。たとえば、次のプロシージャを使用して自動SQLチューニング・アドバイザ・タスクを完全に無効にできます。

```
BEGIN
  dbms_auto_task_admin.disable(
    client_name => 'sql tuning advisor',
    operation   => NULL,
    window_name => NULL);
END;
/
```

このメンテナンス・タスクを再度有効にするには、次のようにENABLEプロシージャを使用します。

```
BEGIN
  dbms_auto_task_admin.enable(
    client_name => 'sql tuning advisor',
    operation   => NULL,
    window_name => NULL);
END;
/
```

client_name引数に使用するタスク名は、DBA_AUTOTASK_CLIENTデータベース・ディクショナリ・ビューにリストされています。

すべてのウィンドウに対して自動化メンテナンス・タスクすべてを有効または無効にするには、引数を指定せずにENABLEまたはDISABLEプロシージャをコールします。

```
EXECUTE DBMS_AUTO_TASK_ADMIN.DISABLE;
```

関連項目:

- 「自動化メンテナンス・タスクのデータベース・ディクショナリ・ビュー」
- DBMS_AUTO_TASK_ADMIN PL/SQLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 自動化メンテナンス・タスクの構成

26.3.2 特定のメンテナンス・ウィンドウに対するメンテナンス・タスクの有効化と無効化

デフォルトでは、すべてのメンテナンス・タスクが事前定義のすべてのメンテナンス・ウィンドウで実行されます。管理者は、特定のウィンドウに対してメンテナンス・タスクを無効にできます。

次の例は、ウィンドウMONDAY_WINDOWでの自動SQLチューニング・アドバイザの実行を無効にしています。

```
BEGIN
  dbms_auto_task_admin.disable(
client_name => 'sql tuning advisor',
operation   => NULL,
window_name => 'MONDAY_WINDOW');
END;
/
```

親トピック: 自動化メンテナンス・タスクの構成

26.4 メンテナンス・ウィンドウの構成

事前定義のメンテナンス・ウィンドウをデータベース環境に適した時間に調整したり、新しいメンテナンス・ウィンドウを作成できま

す。メンテナンス・ウィンドウは、DBMS_SCHEDULER PL/SQLパッケージを使用してカスタマイズできます。

- メンテナンス・ウィンドウの変更
  DBMS_SCHEDULER PL/SQLパッケージには、ウィンドウの属性を変更するためのSET_ATTRIBUTEプロシージャが含まれています。
- 新規メンテナンス・ウィンドウの作成
  新規メンテナンス・ウィンドウを作成するには、Oracle Schedulerウィンドウ・オブジェクトを作成して、ウィンドウ・グループMAINTENANCE_WINDOW_GROUPに追加する必要があります。
- メンテナンス・ウィンドウの削除
  既存のメンテナンス・ウィンドウを削除するには、MAINTENANCE_WINDOW_GROUPウィンドウ・グループから該当するウィンドウを削除します。

親トピック: 自動データベース・メンテナンス・タスクの管理

26.4.1 メンテナンス・ウィンドウの変更

DBMS_SCHEDULER PL/SQLパッケージには、ウィンドウの属性を変更するためのSET_ATTRIBUTEプロシージャが含まれています。

たとえば、次のスクリプトは、メンテナンス・ウィンドウSATURDAY_WINDOWの期間を4時間に変更します。

```
BEGIN
  dbms_scheduler.disable(
    name  => 'SATURDAY_WINDOW');
  dbms_scheduler.set_attribute(
    name      => 'SATURDAY_WINDOW',
    attribute => 'DURATION',
    value     => numtodsinterval(4, 'hour'));
  dbms_scheduler.enable(
    name => 'SATURDAY_WINDOW');
END;
/
```

ウィンドウを変更する際は、事前にDBMS_SCHEDULER.DISABLEサブプログラムを使用してそのウィンドウを無効にし、終了した後にDBMS_SCHEDULER.ENABLEを使用して再度有効にする必要があります。現在オープン中のウィンドウを変更した場合は、そのウィンドウの次回オープンまで変更内容が反映されません。

関連項目:

ウィンドウの変更の詳細は、「ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理」を参照してください。

親トピック: メンテナンス・ウィンドウの構成

26.4.2 新規メンテナンス・ウィンドウの作成

新規メンテナンス・ウィンドウを作成するには、Oracle Schedulerのウィンドウを作成して、ウィンドウ・グループMAINTENANCE_WINDOW_GROUPに追加する必要があります。

DBMS_SCHEDULER.CREATE_WINDOWパッケージ・プロシージャを使用してウィンドウを作成し、DBMS_SCHEDULER.ADD_GROUP_MEMBERプロシージャを使用して、この新規ウィンドウをウィンドウ・グループに追加します。

次の例は、EARLY_MORNING_WINDOWという名前のメンテナンス・ウィンドウを作成しています。このウィンドウは、午前5時から午前6時まで毎日1時間実行されます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_WINDOW(
    window_name     => 'EARLY_MORNING_WINDOW',
    duration        =>  NUMTODSINTERVAL(1, 'hour'),
    resource_plan   => 'DEFAULT_MAINTENANCE_PLAN',
    repeat_interval => 'FREQ=DAILY;BYHOUR=5;BYMINUTE=0;BYSECOND=0');
  DBMS_SCHEDULER.ADD_GROUP_MEMBER(
    group_name  => 'MAINTENANCE_WINDOW_GROUP',
    member      => 'EARLY_MORNING_WINDOW');
END;
/
```

関連項目:

- 「ウィンドウの作成」
- DBMS_SCHEDULERパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: メンテナンス・ウィンドウの構成

26.4.3 メンテナンス・ウィンドウの削除

既存のメンテナンス・ウィンドウを削除するには、MAINTENANCE_WINDOW_GROUPウィンドウ・グループから該当するウィンドウを削除します。

ウィンドウは存在し続けますが、自動化メンテナンス・タスクは実行されません。このウィンドウに割り当てられているOracle Schedulerの他のジョブは、引き続き通常どおりに実行されます。

次の例は、ウィンドウ・グループからEARLY_MORNING_WINDOWを削除しています。

```
BEGIN
  DBMS_SCHEDULER.REMOVE_GROUP_MEMBER(
    group_name  => 'MAINTENANCE_WINDOW_GROUP',
    member      => 'EARLY_MORNING_WINDOW');
END;
/
```

関連項目:

- 「ウィンドウ・グループからのメンバーの削除」
- 「ウィンドウの削除」
- DBMS_SCHEDULERパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: メンテナンス・ウィンドウの構成

26.5 自動化メンテナンス・タスクに対するリソース割当ての構成

自動化メンテナンス・タスクに対するリソース割当てを増減できます。

- 自動化メンテナンス・タスクに対するリソース割当てについて
  デフォルトでは、事前定義のすべてのメンテナンス・ウィンドウで、リソース・プランDEFAULT_MAINTENANCE_PLANが

使用されます。自動化メンテナンス・タスクは、そのサブプランORA$AUTOTASKのもとで実行されます。このサブプランによって、リソース割当て合計に対する配分が複数のメンテナンス・タスク間に均一に分割されます。

- 自動化メンテナンス・タスクに対するリソース割当ての変更
  メンテナンス・ウィンドウ内の自動化メンテナンス・タスクに対するリソース割当てを変更するには、そのウィンドウのリソース・プランのサブプランORA$AUTOTASKに割り当てられているリソースの割合を変更する必要があります。

関連項目:

Oracle Database Resource Managerを使用したリソースの管理

親トピック: 自動データベース・メンテナンス・タスクの管理

26.5.1 自動化メンテナンス・タスクに対するリソース割当てについて

デフォルトでは、事前定義のすべてのメンテナンス・ウィンドウで、リソース・プランDEFAULT_MAINTENANCE_PLANが使用されます。自動化メンテナンス・タスクは、そのサブプランORA$AUTOTASKのもとで実行されます。このサブプランによって、リソース割当て合計に対する配分が複数のメンテナンス・タスク間に均一に分割されます。

DEFAULT_MAINTENANCE_PLANには、次のリソース割当てが定義されています。

コンシューマ・グループ/サブプラン	レベル1	最大使用率の制限
ORA$AUTOTASK	5%	90
OTHER_GROUPS	20%	-
SYS_GROUP	75%	-

このプランでは、SYS_GROUPコンシューマ・グループのセッションが優先されます。(このグループのセッションは、SYSおよびSYSTEMのユーザー・アカウントで作成されたセッションです。)SYS_GROUPのセッションで使用されないリソース割当ては、プランの他のコンシューマ・グループやサブプランに属するセッションによって共有されます。この割当てでは、5%がメンテナンス・タスクに、20%がユーザー・セッションに割り当てられます。ORA$AUTOTASKの最大使用率の制限は90です。つまり、CPUがアイドル状態であっても、このグループ/プランにはCPUリソースの90%を超える値を割り当てることができません。

自動化メンテナンス・タスクに対するリソース割当てを増減するには、DEFAULT_MAINTENANCE_PLANを調整します。詳細は、「自動化メンテナンス・タスクに対するリソース割当ての変更」を参照してください。

リソース・プランと同様に、コンシューマ・グループまたはサブプランによって使用されていない割当て部分は、他のコンシューマ・グループまたはサブプランで使用できます。データベース・リソース・マネージャは、CPUの100%が使用されるまで、リソース・プランによるリソース割当ての制限を開始しません。

ノート:

DEFAULT_MAINTENANCE_PLAN はデフォルトですが、任意のリソース・プランを任意のメンテナンス・ウィンドウに割り当てることができます。メンテナンス・ウィンドウのリソース・プランを変更する場合は、サブプラン ORA$AUTOTASK が新しいプランに含まれていることを確認してください。

関連項目:

リソース・プランの詳細は、「Oracle Database Resource Managerを使用したリソースの管理」を参照してください。

親トピック: 自動化メンテナンス・タスクに対するリソース割当ての構成

26.5.2 自動化メンテナンス・タスクに対するリソース割当ての変更

メンテナンス・ウィンドウ内の自動化メンテナンス・タスクに対するリソース割当てを変更するには、そのウィンドウのリソース・プランのサブプランORA$AUTOTASKに割り当てられているリソースの割合を変更する必要があります。

(デフォルトでは、事前定義の各メンテナンス・ウィンドウのリソース・プランはDEFAULT_MAINTENANCE_PLANです。)また、プランのトップレベルでのリソース割当てが100%になるように、ウィンドウのリソース・プランの1つ以上のサブプランまたはコンシューマ・グループに対するリソース割当てを調整する必要があります。リソース割当ての変更の詳細は、「Oracle Database Resource Managerを使用したリソースの管理」を参照してください。

親トピック: 自動化メンテナンス・タスクに対するリソース割当ての構成

26.6 自動化メンテナンス・タスクの参照情報

Oracle Databaseには事前定義されたメンテナンス・ウィンドウがあります。また、データ・ディクショナリ・ビューもあるため、自動化メンテナンスに関する情報を問い合せることができます。

- 事前定義のメンテナンス・ウィンドウ
  デフォルトでは、7種類のメンテナンス・ウィンドウが事前定義されており、それぞれが曜日を表します。
- 自動化メンテナンス・タスクのデータベース・ディクショナリ・ビュー
  データ・ディクショナリ・ビューのセットを問い合せて、自動化メンテナンス・タスクについての情報を取得できます。

親トピック: 自動データベース・メンテナンス・タスクの管理

26.6.1 事前定義のメンテナンス・ウィンドウ

デフォルトでは、7種類のメンテナンス・ウィンドウが事前定義されており、それぞれが曜日を表します。

週末のメンテナンス・ウィンドウであるSATURDAY_WINDOWとSUNDAY_WINDOWは、平日のメンテナンス・ウィンドウよりも期間が長くなっています。ウィンドウ・グループMAINTENANCE_WINDOW_GROUPは、この7種類のウィンドウで構成されています。表26-1に、事前定義のメンテナンス・ウィンドウのリストを示します。

表26-1 事前定義のメンテナンス・ウィンドウ

ウィンドウ名	説明
MONDAY_WINDOW	月曜の午後 10 時に開始して午前 2 時に終了します。
TUESDAY_WINDOW	火曜の午後 10 時に開始して午前 2 時に終了します。
WEDNESDAY_WINDOW	水曜の午後 10 時に開始して午前 2 時に終了します。

ウィンドウ名	説明
THURSDAY_WINDOW	木曜の午後 10 時に開始して午前 2 時に終了します。
FRIDAY_WINDOW	金曜の午後 10 時に開始して午前 2 時に終了します。
SATURDAY_WINDOW	土曜の午前 6 時に開始して 20 時間後に終了します。
SUNDAY_WINDOW	日曜の午前 6 時に開始して 20 時間後に終了します。

親トピック: 自動化メンテナンス・タスクの参照情報

26.6.2 自動化メンテナンス・タスクのデータベース・ディクショナリ・ビュー

データ・ディクショナリ・ビューのセットを問い合せて、自動化メンテナンス・タスクについての情報を取得できます。

表26-2に、自動化メンテナンス・タスクのデータベース・ディクショナリ・ビューに関する情報を示します。

表26-2 自動化メンテナンス・タスクのデータベース・ディクショナリ・ビュー

ビュー名	説明
DBA_AUTOTASK_CLIENT_JOB	自動化メンテナンス・タスク用に作成され、現在実行中のスケジューラのジョブに関する情報が含まれます。また、これらのジョブのターゲット・オブジェクトだけでなく、同じタスクを以前にインスタンス化したときの付加的な統計に関する情報も示します。この付加的なデータの一部は、一般的なスケジューラ・ビューから取得されます。
DBA_AUTOTASK_CLIENT	7 日間および 30 日間の各自動化メンテナンス・タスクに関する統計データが表示されます。
DBA_AUTOTASK_JOB_HISTORY	自動化メンテナンス・タスクのジョブ実行履歴がリストされます。ジョブの実行が終了すると、このビューにジョブが追加されます。
DBA_AUTOTASK_WINDOW_CLIENTS	MAINTENANCE_WINDOW_GROUP に属するウィンドウが、各メンテナンス・タスクのウィンドウの「有効」または「無効」ステータスとともにリストされます。主に Cloud Control によって使用されます。
DBA_AUTOTASK_CLIENT_HISTORY	各自動化メンテナンス・タスクのジョブ実行回数について、ウィンドウ別に履歴が表示されます。この情報は、Cloud Control の「ジョブ履歴」ページで確認できます。

関連項目:

ビューの列の詳細は、「リソース・マネージャのデータ・ディクショナリ・ビュー」を参照してください。

親トピック: 自動化メンテナンス・タスクの参照情報

27 Oracle Database Resource Managerを使用したリソースの管理

Oracle Database Resource Manager(リソース・マネージャ)を使用して、データベースのリソース割当てを管理できます。

ノート:

この章では、PL/SQL パッケージ・プロシージャを使用してリソース・マネージャを管理する方法について説明します。リソース・マネージャをより簡単に管理するには、Oracle Enterprise Manager Cloud Control (Cloud Control)のグラフィカル・ユーザー・インタフェースを使用します。Cloud Control を使用したリソース・マネージャの管理の詳細は、Cloud Control のオンライン・ヘルプを参照してください。

Cloud Control でリソース・マネージャを使用するには:

1. データベース・ホームページにアクセスします。
2. 「管理」メニューから、「リソース・マネージャ」を選択します。

- Oracle Database Resource Managerについて
  Oracle Database Resource Manager (リソース・マネージャ)を使用すると、システム・リソースやデータベース・リソースを求めて競合するデータベース内の複数のワークロードを管理できます。
- リソース・コンシューマ・グループへのセッションの割当て
  データベース管理者、ユーザーおよびアプリケーションがセッションをリソース・コンシューマ・グループに割り当てる際に使用できる自動および手動の方法があります。セッションがリソース・コンシューマ・グループに割り当てられると、Oracle Database Resource Manager(リソース・マネージャ)では、そのセッションに対するリソース割当てが管理の対象となります。
- リソース・マネージャによって管理されるリソースのタイプ
  リソース・プラン・ディレクティブは、リソース・コンシューマ・グループまたはサブプランへのリソースの割当て方法を指定します。各ディレクティブでは、リソースをそのコンシューマ・グループまたはサブプランに割り当てるための複数の異なる方法を指定できます。
- 単純なリソース・プランの作成
  CREATE_SIMPLE_PLANプロシージャを使用すると、多くの状況に対応できる単純なリソース・プランを迅速に作成できます。
- 複雑なリソース・プランの作成
  複雑なリソース・プランが必要な場合は、ペンディング・エリアと呼ばれるステージング領域で、プランとそのディレクティブやコンシューマ・グループを作成し、そのプランの妥当性をチェックしてから、データ・ディクショナリに保存する必要があります。
- Oracle Database Resource Managerの有効化とプランの切替え
  Oracle Database Resource Manager(リソース・マネージャ)を有効にするには、RESOURCE_MANAGER_PLAN初期化パラメータを設定します。このパラメータは、トップレベルのプランを指定し、現行のインスタンスで使用するプランを識別します。このパラメータでプランを指定しない場合、リソース・マネージャは有効になりません。
- 各種の方法を組み合せたOracle Database Resource Managerの例

リソース・マネージャでリソースを割り当てる方法を例で示します。

- 単一サーバーにおける複数のデータベース・インスタンスの管理
  Oracle Databaseには、複数のデータベース・インスタンスを実行する複数CPUサーバーでCPU割当てを管理する方法が用意されています。この方法はインスタンス・ケージングと呼ばれます。インスタンス・ケージングとOracle Database Resource Manager(リソース・マネージャ)が連携して、複数インスタンス間で必要なサービス・レベルをサポートします。
- コンシューマ・グループ、プランおよびディレクティブのメンテナンス
  Oracle Database Resource Manager (リソース・マネージャ)のコンシューマ・グループ、リソース・プランおよびリソース・プラン・ディレクティブをメンテナンスできます。メンテナンス・タスクは、DBMS_RESOURCE_MANAGER PL/SQLパッケージを使用して実行します。
- データベース・リソース・マネージャの構成とステータスの表示
  Oracle Database Resource Manager (リソース・マネージャ)の現在の構成とステータスを表示するには、いくつかの静的データ・ディクショナリ・ビューと動的パフォーマンス・ビューを使用できます。
- Oracle Database Resource Managerの監視
  動的パフォーマンス・ビューのセットにより、Oracle Database Resource Manager設定の結果を監視できます。
- オペレーティング・システムのリソース制御との相互作用
  多くのオペレーティング・システムではリソース管理ツールを提供しています。これらのツールの名前には、「workload manager」や「resource manager」などの語句が含まれており、管理者が定義したポリシーを使用して、単一のサーバー・リソースを複数のアプリケーションで共有できることを意図しています。たとえば、HP社のProcess Resource Managerや、Solaris Containers、ZonesおよびResource Poolsなどがあります。
- Oracle Database Resource Managerの参照情報
  リソース・マネージャには、事前定義済のリソース・プラン、コンシューマ・グループおよびコンシューマ・グループ・マッピング・ルールが含まれます。リソース・マネージャ構成に関する情報をデータ・ディクショナリ・ビューに問い合せることができます。

親トピック: データベース・リソースの管理とタスクのスケジューリング

27.1 Oracle Database Resource Managerについて

Oracle Database Resource Manager (リソース・マネージャ)を使用すると、システム・リソースやデータベース・リソースを求めて競合するデータベース内の複数のワークロードを管理できます。

- リソース・マネージャが提供するワークロード管理のソリューション
  リソース・マネージャでは、データベースがハードウェア・リソースの割当て方法をより詳細に制御できます。
- リソース・マネージャの要素
  リソース・マネージャには、管理できる複数の要素が含まれます。
- リソース・マネージャの管理権限について
  リソース・マネージャを管理するために必要な権限を持っている必要があります。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.1.1 リソース・マネージャが提供するワークロード管理のソリューション

リソース・マネージャでは、ハードウェア・リソースの割当て方法をデータベースで厳密に制御できます。

データベース・リソースの割当てがオペレーティング・システムによって決定される場合は、次のようなワークロードの管理に関する問題が生じることがあります。

- 過度のオーバーヘッド

過剰なオーバーヘッドは、サーバー・プロセスの数が多いときに、Oracle Databaseサーバー・プロセス間でオペレーティング・システムのコンテキストが切り替わることによって発生します。

- 非効率的なスケジューリング

  オペレーティング・システムは、データベース・サーバーがラッチを保持している間にデータベース・サーバーのスケジュールを解除しますが、これは非効率的です。

- 不適当なリソースの割当て

  オペレーティング・システムはタスク間の優先度付けができないため、すべてのアクティブなプロセスの間で均等にリソースを分配します。

- パラレル実行サーバーやアクティブ・セッションなど、データベース固有のリソースを管理できない

リソース・マネージャは、ハードウェア・リソースの割当て方法をデータベースで厳密に制御することによって、これらの問題を克服します。複数のコンカレント・ユーザー・セッションがあり、各セッションでは異なる優先度のジョブが実行される環境では、すべてのセッションが同等に処理されるわけではありません。リソース・マネージャを使用すると、セッション属性に基づいてセッションを複数のグループに分類し、それらのグループに対して、アプリケーション環境のハードウェア利用が最適化されるようにリソースを配分できます。

リソース・マネージャを使用すると、次のことが可能になります。

- システムにかかる負荷やユーザー数に関係なく、特定のセッションのCPUが最小量になることを保証します。
- 様々なユーザーおよびアプリケーションに、比率を指定してCPU時間を割り当てて、使用可能なCPUを分配します。データ・ウェアハウスの場合は、バッチ・ジョブよりもリレーショナル・オンライン分析処理(ROLAP)アプリケーションへの割当て率を高くできます。
- ユーザー・グループのメンバーが実行する処理の並列度を制限します。
- パラレル・ステートメント・キュー内のパラレル・ステートメントの順序を管理します。優先度の低いユーザー・グループからのパラレル・ステートメントよりも前に、重要なアプリケーションからのパラレル・ステートメントをエンキューできます。
- ユーザー・グループが使用できるパラレル実行サーバーの数を制限します。これにより、使用可能なすべてのパラレル実行サーバーが1つのユーザー・グループにのみ割り当てられることを回避できます。
- アクティブ・セッション・プールを作成します。アクティブ・セッション・プールは、ユーザー・グループ内で同時にアクティブにできる指定された最大数のユーザー・セッションで構成されています。最大数を超える追加セッションは実行待ちのキューに送られますが、キューで待機しているジョブが終了するまでのタイムアウト周期を指定できます。アクティブ・セッション・プールによって、リソースについて実際に競合するセッションの総数が制限されるため、アクティブなセッションの進行を早めることができます。
- リソースの監視

  リソース使用量に関する統計を自動的に記録します。これらの統計は、リアルタイムSQL監視およびリソース・マネージャ動的パフォーマンス・ビュー(V$RSRC_*ビュー)を使用して調べることができます。リアルタイムSQL監視およびリソース・マネージャの動的パフォーマンス・ビューの詳細は、「Oracle Database Resource Managerの監視」を参照してください。

- ユーザーのグループに属する各セッションによって使用されるPGAメモリーの量を制限します。
- 次の方法でリソース集中型のセッションまたはコールを管理します。
  - セッションまたはコールによって使用されるCPU、物理I/O、論理I/Oまたは経過時間が指定量を超過してい

る状況を検出し、自動的に、セッションまたはコールを終了するか、リソースの割当てが少ない、あるいはグループが使用できるCPUの割合に制限があるコンシューマ・グループに切り替えます。システム・リソースの過剰な消費により終了したSQL文は隔離されます。つまり、後続の実行中にコンパイル・エラーが発生することにより、それらの再実行は許可されません。

論理I/Oは、バッファI/Oとも呼ばれ、バッファ・キャッシュ内でのバッファの読取りおよび書込みを表しています。要求されたバッファがメモリーにない場合、データベースでは物理I/Oを実行してディスクまたはフラッシュ・キャッシュからメモリーにバッファをコピーし、次に論理I/Oを実行して、キャッシュされたバッファを読み取ります。

- リアルタイムSQL監視を使用して、指定量を超えるCPU、物理I/O、論理I/Oまたは経過時間を使用するSQL文に関する詳細情報を記録します。
- 自動ワークロード・リポジトリ(AWR)を使用して、指定量を超えるCPU、物理I/O、論理I/Oまたは経過時間を使用するSQL文の永続レコードを分析します。
- セッションに関連するその他の処理を実行しないで、リソース集中型のセッションに関する情報をログに記録します。

- ある操作に対するオプティマイザの見積り実行時間が指定された制限を超える場合は、その操作が実行されないようにします。
- セッションのアイドル時間を制限します。この制限は、他のセッションをブロックしているセッションのみに限定することもできます。
- ワークロード要件の変更に基づいて、データベースで様々なリソース・プランを使用できます。インスタンスを停止して再起動しなくても、たとえば、昼間のリソース・プランから夜間のリソース・プランに変更するなど、リソース・プランを動的に変更できます。リソース・プランの変更は、Oracle Schedulerを使用してスケジュールすることもできます。詳細は、「Oracle Schedulerの概要」を参照してください。

親トピック: Oracle Database Resource Managerについて

27.1.2 リソース・マネージャの要素

リソース・マネージャには、管理できる複数の要素が含まれます。

- リソース・マネージャの要素について
  リソース・マネージャの要素には、リソース・コンシューマ・グループ、リソース・プランおよびリソース・プラン・ディレクティブが含まれます。
- リソース・コンシューマ・グループについて
  リソース・コンシューマ・グループ(コンシューマ・グループ)とは、処理要件に基づいてグループ化されたユーザー・セッションの集合です。
- リソース・プラン・ディレクティブについて
  リソース・マネージャは、現在アクティブなリソース・プランに属するリソース・プラン・ディレクティブ(ディレクティブ)のセットに従って、リソースをコンシューマ・グループに割り当てます。
- リソース・プランについて
  リソース・プランは、リソース・コンシューマ・グループへのリソースの割当て方法を指定するディレクティブのコンテナです。
- 例: 単純なリソース・プラン
  単純なリソース・プランの例を示します。
- サブプランについて
  リソース・プラン・ディレクティブ(ディレクティブ)は、コンシューマ・グループを参照するかわりに、別のリソース・プランを参照

できます。この場合、そのプランはサブプランと呼ばれます。

- 例: サブプランを含むリソース・プラン
  サブプランを含むリソース・プランの例を示します。

親トピック: Oracle Database Resource Managerについて

27.1.2.1 リソース・マネージャの要素について

リソース・マネージャの要素には、リソース・コンシューマ・グループ、リソース・プランおよびリソース・プラン・ディレクティブが含まれます。

要素	説明
リソース・コンシューマ・グループ	リソースの要件に基づいてグループ化されたセッションのグループ。リソース・マネージャは、個々のセッションではなくリソース・コンシューマ・グループにリソースを割り当てます。
リソース・プラン	リソース・コンシューマ・グループへのリソースの割当て方法を指定するディレクティブのコンテナ。特定のリソース・プランをアクティブ化することで、データベースによるリソースの割当て方法を指定します。
リソース・プラン・ディレクティブ	リソース・コンシューマ・グループを特定のプランに関連付け、そのリソース・コンシューマ・グループに対するリソースの割当て方法を指定します。

これらの要素の作成とメンテナンスには、DBMS_RESOURCE_MANAGER PL/SQLパッケージを使用します。要素は、データ・ディクショナリ内の表に格納されます。要素に関する情報は、データ・ディクショナリ・ビューを使用して表示できます。

関連項目:

「リソース・マネージャのデータ・ディクショナリ・ビュー」

親トピック: リソース・マネージャの要素

27.1.2.2 リソース・コンシューマ・グループについて

リソース・コンシューマ・グループ(コンシューマ・グループ)とは、処理要件に基づいてグループ化されたユーザー・セッションの集合です。

セッションが作成されると、そのセッションは、管理者が設定したマッピング・ルールに基づいてコンシューマ・グループに自動的にマップされます。データベース管理者(DBA)は、セッションを、異なるコンシューマ・グループに手動で切り替えることができます。同様に、アプリケーションでは、そのセッションを特定のコンシューマ・グループに切り替えるPL/SQLパッケージ・プロシージャを実行できます。

リソース・マネージャによるリソース(CPUなど)の割当て先は、コンシューマ・グループのみであるため、セッションがコンシューマ・グループのメンバーになると、そのリソース割当ては、コンシューマ・グループの割当てによって決定されます。

常にデータ・ディクショナリに存在する特別なコンシューマ・グループがあります。これらのグループは、変更したり削除することはできません。これらを次に示します。

- SYS_GROUP

これは、ユーザー・アカウントSYSまたはSYSTEMによって作成されたすべてのセッションの初期コンシューマ・グループです。この初期コンシューマ・グループは、コンシューマ・グループへのセッションのマッピング・ルールで上書きできます。

- OTHER_GROUPS

  このコンシューマ・グループには、コンシューマ・グループに割り当てられていないすべてのセッションが含まれます。すべてのリソース・プランにOTHER_GROUPSへのディレクティブが含まれている必要があります。

アクティブなプランに含まれるリソース・コンシューマ・グループ数は、28以内にする必要があります。

関連項目:

- 表27-6
- 「コンシューマ・グループへのセッションのマッピング・ルールの指定」

親トピック: リソース・マネージャの要素

27.1.2.3 リソース・プラン・ディレクティブについて

リソース・マネージャは、現在アクティブなリソース・プランに属するリソース・プラン・ディレクティブ(ディレクティブ)のセットに従って、リソースをコンシューマ・グループに割り当てます。

リソース・プランとそのリソース・プラン・ディレクティブは親子関係にあります。各ディレクティブは1つのコンシューマ・グループを参照し、現在アクティブなプランの2つのディレクティブで同じコンシューマ・グループを参照することはできません。

ディレクティブには、コンシューマ・グループへのリソース割当てを制限できる複数の方法があります。たとえば、CPUにおいてコンシューマ・グループが確保できるCPU全体の割合を制御したり、コンシューマ・グループ内でアクティブにできるセッションの総数を制限できます。詳細は、「リソース・マネージャによって管理されるリソースのタイプ」を参照してください。

親トピック: リソース・マネージャの要素

27.1.2.4 リソース・プランについて

リソース・プランは、リソース・コンシューマ・グループへのリソースの割当て方法を指定するディレクティブのコンテナです。

各Oracle Databaseに事前定義されているリソース・プランの他に、リソース・プランは必要な数だけ作成できます。ただし、1度にアクティブにできるリソース・プランは1つのみです。リソース・プランがアクティブな場合は、その子の各リソース・プラン・ディレクティブが異なるコンシューマ・グループへのリソース割当てを制御します。各プランには、OTHER_GROUPSというコンシューマ・グループにリソースを割り当てるディレクティブが必要です。OTHER_GROUPSは、現在アクティブなプランの一部でないコンシューマ・グループに属しているすべてのセッションに適用されます。

ノート:

「リソース・プラン」(または単に「プラン」)という用語はリソース・マネージャの 1 つの要素を示しますが、この章では、完全なリソース・プラン・スキーマ(リソース・プランの要素そのもの、そのリソース・プラン・ディレクティブ、ディレクティブによって参照されるコンシューマ・グループなど)を指す場合にも使用します。たとえば、この章で DAYTIME リソース・プランに触れたときは、DAYTIME というリソース・プラン要素を意味する場合も、DAYTIME リソース・プランとそのディレクティブが定義する特定のリソース割当てスキーマを意味する場合もあります。したがって、簡潔に言えば、「DAYTIME プランは、バッチ・アプリケーションよりも対話型アプリケーションに近い」といえます。

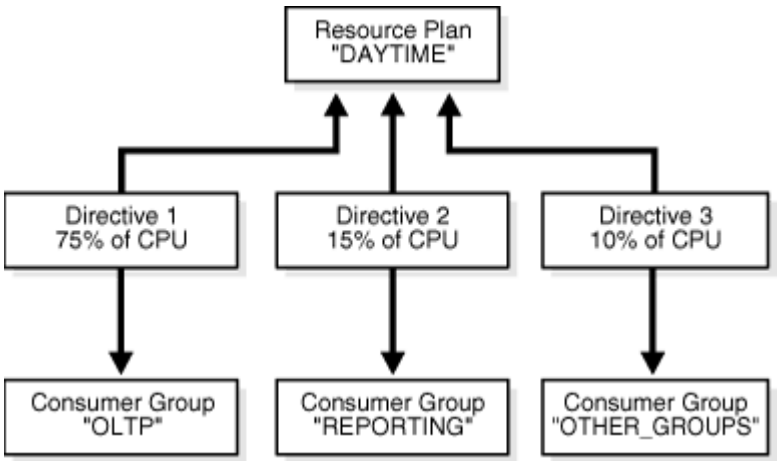
親トピック: リソース・マネージャの要素

27.1.2.5 例: 単純なリソース・プラン

単純なリソース・プランの例を示します。

図27-1は、日中にオンライン・トランザクション処理(OLTP)アプリケーションとレポート・アプリケーションを同時に実行する組織の単純なリソース・プランを示しています。現在、DAYTIMEのプランがアクティブになっており、3つのリソース・コンシューマ・グループ間でCPUリソースを割り当てます。具体的には、CPUタイムの75%がOLTPに、15%がREPORTSに、残りの10%がOTHER_GROUPSに割り当てられています。リソースの競合がない場合、どのグループも保証されているより多くのリソースを使用できます。たとえば、OLTPはCPUの75%を保証されていますが、リソースの競合がない場合は、最大でCPUの100%を使用できます。

図27-1 単純なリソース・プラン



Oracle Databaseには、単純なリソース・プランを迅速に作成できるプロシージャ(CREATE_SIMPLE_PLAN)が用意されています。このプロシージャについては、「単純なリソース・プランの作成」を参照してください。

ノート:

現在アクティブなリソース・プランは、CPU 使用率が 100%になるまで割当てを規定しません。CPU 使用率が 100%未満の場合、データベースは CPU による制限を受けないため、すべてのセッションが必要なリソース割当てを獲得できるように割当てを規定する必要はありません。

さらに、割当てが規定されている場合は、あるコンシューマ・グループが使用していない割当てを、他のコンシューマ・グループが使用できます。前述の例では、OLTP グループが割当てをすべて使用していない場合、リソース・マネージャは、REPORTS グループまたは OTHER_GROUPS グループが、未使用の割当てを使用することを許可します。

親トピック: リソース・マネージャの要素

27.1.2.6 サブプランについて

リソース・プラン・ディレクティブ(ディレクティブ)は、コンシューマ・グループを参照するかわりに、別のリソース・プランを参照できます。この場合、そのプランはサブプランと呼ばれます。

サブプラン自体に、リソースをコンシューマ・グループと他のサブプランに割り当てるディレクティブがあります。その場合、リソースの割当て方式は次のように機能します。トップレベルのリソース・プラン(現在アクティブなプラン)によって、リソースがコンシューマ・グループおよびサブプランに分割されます。次に、各サブプランによって、リソース割当て合計の一部がそのコンシューマ・グループとサ

ブプランに割り当てられます。階層的なプランは、必要な数のサブプランを使用して作成できます。

リソース・サブプランは、リソース・プランを作成する方法と同じ方法で作成します。サブプランとしてのみ使用されるプランを作成するには、パッケージ・プロシージャDBMS_RESOURCE_MANAGER.CREATE_PLANでSUB_PLAN引数を使用します。

トップレベルのプランでは、サブプランを1回のみ参照できます。サブプランは、OTHER_GROUPSへのディレクティブを必要とせず、リソース・プランとして設定できません。

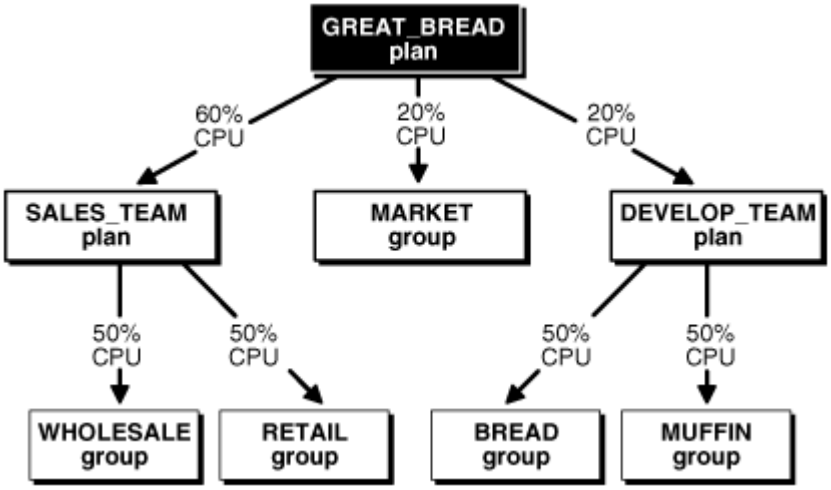
親トピック: リソース・マネージャの要素

27.1.2.7 例: サブプランを含むリソース・プラン

サブプランを含むリソース・プランの例を示します。

この例では、Great Bread Companyによって、CPUリソースが図27-2のように割り当てられています。この図は、トップレベルのプラン(GREAT_BREAD)とその子すべてを示しています。わかりやすくするために、OTHER_GROUPSコンシューマ・グループを指定する要件は無視し、リソース・プラン・ディレクティブは(プランの一部であっても)省略されています。かわりに、ディレクティブが割り当てるCPUの比率を、プラン、サブプランおよびコンシューマ・グループ間を結ぶ線に沿って表示しています。

図27-2 サブプランを含むリソース・プラン



このGREAT_BREADプランでは、次のようにリソースを割り当てます。

- CPUリソースの20%をコンシューマ・グループMARKETに割り当てます。
- CPUリソースの60%をサブプランSALES_TEAMに割り当てます。このサブプランでは、リソースがWHOLESALEコンシューマ・グループとRETAILコンシューマ・グループ間で均等に分割されます。
- CPUリソースの20%をサブプランDEVELOP_TEAMに割り当てます。このサブプランでは、リソースがBREADコンシューマ・グループとMUFFINコンシューマ・グループ間で均等に分割されます。

サブプランまたはコンシューマ・グループは、複数の親を持つ場合があります。たとえば、MARKETグループがSALES_TEAMサブプランに含まれていた場合です。ただし、プラン内でループすることはできません。たとえば、SALES_TEAMサブプランには、GREAT_BREADプランを参照するディレクティブを設定できません。

関連項目:

複雑なリソース・プランの例は、「各種の方法を組み合せたOracle Database Resource Managerの例」を参照してください。

親トピック: リソース・マネージャの要素

27.1.3 リソース・マネージャの管理権限について

リソース・マネージャを管理するために必要な権限を持っている必要があります。

リソース・マネージャを管理するには、システム権限ADMINISTER_RESOURCE_MANAGERが必要です。この権限(ADMINオプション付き)は、DBAロールを介してデータベース管理者に付与されます。

リソース・マネージャの管理者は、DBMS_RESOURCE_MANAGER PL/SQLパッケージ内のすべてのプロシージャを実行できます。

ADMINオプションを持つ管理者は、必要に応じて管理権限を他のユーザーまたはロールに付与できます。そうするには、DBMS_RESOURCE_MANAGER_PRIVS PL/SQLパッケージを使用します。次の表に、関連するパッケージ・プロシージャの一覧を示します。

プロシージャ	説明
GRANT_SYSTEM_PRIVILEGE	ユーザーまたはロールに、ADMINISTER_RESOURCE_MANAGER システム権限を付与します。
REVOKE_SYSTEM_PRIVILEGE	ユーザーまたはロールから、ADMINISTER_RESOURCE_MANAGER システム権限を取り消します。

次のPL/SQLブロックは、ユーザーHRに管理権限を付与しますが、HRにADMINオプションは付与しません。そのため、HRはDBMS_RESOURCE_MANAGERパッケージのすべてのプロシージャを実行できますが、HRはGRANT_SYSTEM_PRIVILEGEプロシージャを使用して他のユーザーに管理権限を付与することはできません。

```
BEGIN
  DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE(
   GRANTEE_NAME   => 'HR',
   PRIVILEGE_NAME => 'ADMINISTER_RESOURCE_MANAGER',
   ADMIN_OPTION   => FALSE);
END;
/
```

この権限は、REVOKE_SYSTEM_PRIVILEGEプロシージャを使用して取り消すことができます。

ノート:

ADMINISTER_RESOURCE_MANAGER システム権限の付与または取消しは、DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用する場合のみ行うことができます。SQL の GRANT 文または REVOKE 文によって、付与したり取り消すことはできません。

関連項目:

- DBMS_RESOURCE_MANAGERパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- DBMS_RESOURCE_MANAGER_PRIVSパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

- ADMINオプションの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください

親トピック: Oracle Database Resource Managerについて

27.2 リソース・コンシューマ・グループへのセッションの割当て

データベース管理者、ユーザーおよびアプリケーションがセッションをリソース・コンシューマ・グループに割り当てる際に使用できる自動および手動の方法があります。セッションがリソース・コンシューマ・グループに割り当てられると、Oracle Database Resource Manager(リソース・マネージャ)では、そのセッションに対するリソース割当てが管理の対象となります。

ノート:

コンシューマ・グループに割り当てられていないセッションは、コンシューマ・グループ OTHER_GROUPS に配置されます。

- リソース・コンシューマ・グループへのセッション割当ての概要
  リソース・マネージャを有効にする前に、ユーザー・セッションをリソース・コンシューマ・グループに割り当てる方法を指定する必要があります。
- 初期リソース・コンシューマ・グループの割当て
  セッションの初期コンシューマ・グループは、管理者が構成するマッピング・ルールによって決まります。
- コンシューマ・グループへのセッションのマッピング・ルールの指定
  コンシューマ・グループへのセッションのマッピング・ルールを作成し、優先順位を付けることができます。
- リソース・コンシューマ・グループの切替え
  セッションのリソース・コンシューマ・グループを切り替えることができます。
- コンシューマ・グループの自動切替えの指定
  特定の条件を満たしたときに、セッションを別のコンシューマ・グループに自動的に切り替えるように、リソース・マネージャを構成できます。
- スイッチ特権の付与と取消し
  ユーザーまたはアプリケーションがセッションを指定されたリソース・コンシューマ・グループに切り替えるには、スイッチ特権が必要です。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.2.1 リソース・コンシューマ・グループへのセッション割当ての概要

リソース・マネージャを有効にする前に、ユーザー・セッションをリソース・コンシューマ・グループに割り当てる方法を指定する必要があります。

これを行うために、マッピング・ルールを作成すると、セッションの起動時に、セッション属性に基づいてリソース・マネージャによって自動的に各セッションがコンシューマ・グループに割り当てられるようにすることができます。セッションがその初期コンシューマ・グループに割り当てられ実行された後は、プロシージャをコールして、セッションを別のコンシューマ・グループに手動で切り替えることができます。この操作は、リソースを過剰に使用しているセッションを、よりリソース割当ての制限されたコンシューマ・グループに移動する必要がある場合などに実行します。また、ユーザーおよびアプリケーションにスイッチ特権を付与して、ユーザーおよびアプリケーションが、あるコンシューマ・グループから別のコンシューマ・グループに自分のセッションを切り替えることができるようにすることができます。

セッション属性に変更がある場合やセッションが指定のリソース使用制限を超える場合は、あるコンシューマ・グループから別の

(通常は優先度の低い)コンシューマ・グループにセッションを自動的に切り替えることもできます。

親トピック: リソース・コンシューマ・グループへのセッションの割当て

27.2.2 初期リソース・コンシューマ・グループの割当て

セッションの初期コンシューマ・グループは、管理者が構成するマッピング・ルールによって決まります。

マッピング・ルールの構成方法は、「コンシューマ・グループへのセッションのマッピング・ルールの指定」を参照してください。

親トピック: リソース・コンシューマ・グループへのセッションの割当て

27.2.3 コンシューマ・グループへのセッションのマッピング・ルールの指定

コンシューマ・グループへのセッションのマッピング・ルールを作成し、優先順位を付けることができます。

- コンシューマ・グループへのセッションのマッピング・ルールについて
  セッションの初期コンシューマ・グループを指定し、セッション属性が変更された場合に別のコンシューマ・グループにセッションを動的に切り替えることができます。
- コンシューマ・グループのマッピング・ルールの作成
  セッション属性と値のペアをコンシューマ・グループにマップするには、SET_CONSUMER_GROUP_MAPPINGプロシージャを使用します。
- コンシューマ・グループのマッピング・ルールの変更と削除
  コンシューマ・グループ・マッピング・ルールを変更するには、必要な属性と値のペアに対してSET_CONSUMER_GROUP_MAPPINGプロシージャを実行し、新しいコンシューマ・グループを指定します。
- マッピング・ルールの優先度の作成
  競合するマッピング・ルールを解決するために、最も高い重要度のセッション属性から最も低いセッション属性まで優先度を設定できます。

親トピック: リソース・コンシューマ・グループへのセッションの割当て

27.2.3.1 コンシューマ・グループへのセッションのマッピング・ルールについて

セッションの初期コンシューマ・グループを指定し、セッション属性が変更された場合に別のコンシューマ・グループにセッションを動的に切り替えることができます。

コンシューマ・グループへのセッションのマッピング・ルールを作成すると、次のことができます。

- セッションの初期コンシューマ・グループをセッション属性に基づいて指定できます。
- リソース・マネージャを使用して、実行中のセッションをセッション属性の変更に基づいて別のコンシューマ・グループに動的に切り替えることができます。

このマッピング・ルールは、セッション属性(ユーザー名、セッションがデータベースへの接続に使用したサービス、クライアント・プログラムの名前など)に基づいています。

マッピング・ルール間の競合を解決するために、リソース・マネージャでは、各ルールが優先度別に順序付けされます。たとえば、ユーザーSCOTTがSALESサービスを使用してデータベースに接続すると仮定します。あるマッピング・ルールには、ユーザーSCOTTがMED_PRIORITYコンシューマ・グループで起動することが記載され、別のルールには、SALESサービスに接続するセッションはHIGH_PRIORITYコンシューマ・グループで起動することが記載されている場合、この競合は、マッピング・ルールの優先度によって解決されます。

マッピング・ルールの基本になるセッション属性には、ログイン属性とランタイム属性の2つのタイプがあります。ログイン属性が有効

なのは、セッションのログイン時、つまり、リソース・マネージャによってセッションの初期コンシューマ・グループが決定されるときのみです。ランタイム属性は、セッションのログイン中とログイン後に適用されます。ランタイム属性のいずれかを変更することによって、ログイン済セッションを別のコンシューマ・グループに再割当てできます。

コンシューマ・グループへのセッションの自動割当てを構成するには、SET_CONSUMER_GROUP_MAPPINGおよびSET_CONSUMER_GROUP_MAPPING_PRIプロシージャを使用します。これらのプロシージャに対してはペンディング・エリアを使用する必要があります。(ペンディング・エリアを作成してプロシージャを実行し、必要に応じてペンディング・エリアの妥当性をチェックしてから、そのペンディング・エリアを発行する必要があります。ペンディング・エリアの使用例は、「複雑なリソース・プランの作成」を参照してください。)

セッションは、マッピング・ルールを介して様々な時点で特定のコンシューマ・グループに自動的に切り替えられます。

- 最初のセッションのログイン時には、マッピング・ルールが評価されてそのセッションの初期グループが判別されます。
- セッション属性が新しい値に動的に変更されると(可能性があるのはランタイム属性のみ)、マッピング・ルールが再評価され、セッションが別のコンシューマ・グループに切り替わる場合があります。

事前定義のコンシューマ・グループ・マッピング・ルール

各Oracle Databaseには、事前定義のコンシューマ・グループ・マッピング・ルールのセットがあります。

- 「リソース・コンシューマ・グループについて」で説明したように、ユーザー・アカウントSYSまたはSYSTEMによって作成されたすべてのセッションは、最初にSYS_GROUPコンシューマ・グループにマップされます。
- データ・ポンプを使用してデータ・ロードを実行するセッションまたはRMANを使用してバックアップまたはコピー操作を実行するセッションは、表27-7で指定されている事前定義のコンシューマ・グループに自動的にマップされます。

DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPINGプロシージャを使用して、これらの事前定義のマッピング・ルールを変更または削除できます。

関連項目:

- 「初期リソース・コンシューマ・グループの割当て」
- 「マッピング・ルールを使用した自動切替えの指定」

親トピック: コンシューマ・グループへのセッションのマッピング・ルールの指定

27.2.3.2 コンシューマ・グループのマッピング・ルールの作成

セッション属性と値のペアをコンシューマ・グループにマップするには、SET_CONSUMER_GROUP_MAPPINGプロシージャを使用します。

このプロシージャのパラメータは、次のとおりです。

パラメータ	説明
ATTRIBUTE	セッション属性タイプ。パッケージ定数として指定されています。
VALUE	属性の値。

パラメータ	説明
CONSUMER_GROUP	この属性と値のペアのマッピング先のコンシューマ・グループ。

ATTRIBUTEには、次のいずれかを指定できます。

属性	タイプ	説明
ORACLE_USER	ログイン	Oracle Database ユーザー名。
SERVICE_NAME	ログイン	クライアントが接続の確立に使用したデータベース・サービス名。
CLIENT_OS_USER	ログイン	ログインしているクライアントのオペレーティング・システム・ユーザー名。
CLIENT_PROGRAM	ログイン	サーバーへのログインに使用されたクライアント・プログラム名。
CLIENT_MACHINE	ログイン	クライアントが接続に使用しているコンピュータ名。
CLIENT_ID	ログイン	セッションのクライアント識別子 クライアント識別子のセッション属性は、DBMS_SESSION.SET_IDENTIFIER プロシージャで設定します。
MODULE_NAME	ランタイム	DBMS_APPLICATION_INFO.SET_MODULE プロシージャによる設定、またはそれに相当する OCI 属性の設定に従って現在実行されているアプリケーション内のモジュール名。
MODULE_NAME_ACTION	ランタイム	次のプロシージャのいずれかによる設定、またはそれに相当する OCI 属性の設定に従って実行されている現行モジュールと処理の組合せ。 ● DBMS_APPLICATION_INFO.SET_MODULE ● DBMS_APPLICATION_INFO.SET_ACTION この属性にはモジュール名を指定し、その後ろにピリオド(.)および処理名を順に続けます(module_name.action_name)。
SERVICE_MODULE	ランタイム	service_name.module_name の書式によるサービス名およびモジュール名の組合せ。
SERVICE_MODULE_ACTION	ランタイム	service_name.module_name.action_name の書式によるサービス名、モジュール名および処理名の組合せ。

属性	タイプ	説明
ORACLE_FUNCTION	ランタイム	RMAN またはデータ・ポンプ操作。有効な値は DATALOAD、BACKUP および COPY です。これらの値それぞれに事前定義のマッピングがあります。セッションでこれらの機能のいずれかを実行している場合、事前定義のコンシューマ・グループに自動的にマップされます。詳細は、表 27-7 を参照してください。

たとえば、次のPL/SQLブロックでは、ログインするたびに、ユーザーSCOTTがDEV_GROUPコンシューマ・グループにマップされます。

```
BEGIN
  DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING
     (DBMS_RESOURCE_MANAGER.ORACLE_USER, 'SCOTT', 'DEV_GROUP');
END;
/
```

この場合も、SET_CONSUMER_GROUP_MAPPINGプロシージャを実行する前にペンディング・エリアを作成する必要があります。

SET_CONSUMER_GROUP_MAPPINGプロシージャではvalueパラメータでほとんどの属性の値に対してワイルドカードを使用できます。ワイルドカードを使用して値を指定するには、SQLのLIKE演算子と同じセマンティクスを使用します。具体的には、ワイルドカードは次のセマンティクスを使用します。

- %は、複数文字のワイルドカード
- _は、1文字のワイルドカード
- ¥は、ワイルドカードをエスケープする

ワイルドカードが使用できるのは、属性が次のいずれかの場合のみです。

- CLIENT_OS_USER
- CLIENT_PROGRAM
- CLIENT_MACHINE
- MODULE_NAME
- MODULE_NAME_ACTION
- SERVICE_MODULE
- SERVICE_MODULE_ACTION

親トピック: コンシューマ・グループへのセッションのマッピング・ルールの指定

27.2.3.3 コンシューマ・グループのマッピング・ルールの変更と削除

コンシューマ・グループ・マッピング・ルールを変更するには、必要な属性と値のペアに対してSET_CONSUMER_GROUP_MAPPINGプロシージャを実行し、新しいコンシューマ・グループを指定します。

ルールを削除するには、必要な属性と値のペアに対してSET_CONSUMER_GROUP_MAPPINGプロシージャを実行し、NULLのコンシューマ・グループを指定します。

親トピック: コンシューマ・グループへのセッションのマッピング・ルールの指定

27.2.3.4 マッピング・ルールの優先度の作成

競合するマッピング・ルールを解決するために、最も高い重要度のセッション属性から最も低いセッション属性まで優先度を設定できます。

各属性に、1(最も高い重要度)から12(最も低い重要度)の一意の整数を設定するには、SET_CONSUMER_GROUP_MAPPING_PRIプロシージャを使用します。次の例は、この優先度の設定方法を示しています。

```
BEGIN
  DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING_PRI(
    EXPLICIT => 1,
    SERVICE_MODULE_ACTION => 2,
    SERVICE_MODULE => 3,
    MODULE_NAME_ACTION => 4,
    MODULE_NAME => 5,
    SERVICE_NAME => 6,
    ORACLE_USER => 7,
    CLIENT_PROGRAM => 8,
    CLIENT_OS_USER => 9,
    CLIENT_MACHINE => 10,
    CLIENT_ID => 11);
END;
/
```

この例では、データベース・ユーザー名の優先度は7(重要度が低い)に設定され、モジュール名の優先度は5(重要度が高い)に設定されています。

ノート:

SET_CONSUMER_GROUP_MAPPING_PRI では、疑似属性 EXPLICIT を引数として含める必要があります。これは、1 に設定する必要があります。これは、明示的なコンシューマ・グループの切替えに最も高い優先度が指定されていることを示します。『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』で詳細に説明されている次のパッケージ・プロシージャを使用して、コンシューマ・グループを明示的に切り替えます。

- DBMS_SESSION.SWITCH_CURRENT_CONSUMER_GROUP
- DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS
- DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER

マッピング・ルールの優先度がどのように機能するかを示すために、引き続き前の例を使用して、DEV_GROUPコンシューマ・グループへのユーザーSCOTTのマッピングの他に、次のようなモジュール名のマッピング・ルールがあると仮定します。

```
BEGIN
  DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING
     (DBMS_RESOURCE_MANAGER.MODULE_NAME, 'EOD_REPORTS', 'LOW_PRIORITY');
END;
/
```

ユーザーSCOTTのセッションのアプリケーションによって、モジュール名がEOD_REPORTSに設定された場合、そのセッションはLOW_PRIORITYコンシューマ・グループに再割当てされます。これは、モジュール名マッピングの方が、データベース・ユーザー・マッピングより優先度が高いためです。

セッション属性の現在の優先度を確認するには、ビューDBA_RSRC_MAPPING_PRIORITYを問い合せます。

関連項目:

- DBMS_APPLICATION_INFO.SET_MODULEプロシージャによるモジュール名の設定の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

- 「スイッチ特権の付与と取消し」

親トピック: コンシューマ・グループへのセッションのマッピング・ルールの指定

27.2.4 リソース・コンシューマ・グループの切替え

セッションのリソース・コンシューマ・グループを切り替えることができます。

- 手動によるリソース・コンシューマ・グループの切替え
  実行中のセッションのリソース・コンシューマ・グループを変更できます。
- ユーザーまたはアプリケーションに対する手動によるコンシューマ・グループの切替えの有効化
  管理者は、ユーザーがDBMS_SESSIONパッケージのSWITCH_CURRENT_CONSUMER_GROUPプロシージャを使用して、現在のコンシューマ・グループを切り替えられるように、スイッチ特権をユーザーに付与できます。

親トピック: リソース・コンシューマ・グループへのセッションの割当て

27.2.4.1 手動によるリソース・コンシューマ・グループの切替え

実行中のセッションのリソース・コンシューマ・グループを変更できます。

- 手動によるリソース・コンシューマ・グループの切替えについて
  DBMS_RESOURCE_MANAGER PL/SQLパッケージには、管理者が実行中のセッションのリソース・コンシューマ・グループを変更できるように、2つのプロシージャが用意されています。
- 単一のセッションの切替え
  SWITCH_CONSUMER_GROUP_FOR_SESSプロシージャは、指定したセッションを、指定したリソース・コンシューマ・グループに即時に移動します。このプロシージャを使用すると、セッションの優先度を実質的に変更できます。
- ユーザーの全セッションの切替え
  SWITCH_CONSUMER_GROUP_FOR_USERプロシージャは、指定したユーザー名に関連するすべてのセッションのリソース・コンシューマ・グループを変更します。

親トピック: リソース・コンシューマ・グループの切替え

27.2.4.1.1 手動によるリソース・コンシューマ・グループの切替えについて

DBMS_RESOURCE_MANAGER PL/SQLパッケージには、管理者が実行中のセッションのリソース・コンシューマ・グループを変更できるように、2つのプロシージャが用意されています。

この2つのプロシージャでは、コーディネータのセッションに対応付けられたパラレル実行サーバー・セッションのコンシューマ・グループも変更できます。これらのプロシージャで行った変更は永続的ではなく、現行セッションのみに影響します。ユーザーの初期コンシューマ・グループも変更されません。

あるユーザーがCPUを過剰に使用している場合は、そのユーザーのセッションを終了するかわりに、ユーザーのコンシューマ・グループをリソースの割当てが低いグループに変更できます。

親トピック: 手動によるリソース・コンシューマ・グループの切替え

27.2.4.1.2 単一のセッションの切替え

SWITCH_CONSUMER_GROUP_FOR_SESSプロシージャは、指定したセッションを、指定したリソース・コンシューマ・グループに即時に移動します。このプロシージャを使用すると、セッションの優先度を実質的に変更できます。

SWITCH_CONSUMER_GROUP_FOR_SESSプロシージャは、Oracle Real Application Clusters (Oracle RAC)インスタンス固有です。切り替えるセッションが実行されているのと同じOracle RACインスタンスのプラガブル・データベースに接続して

から、このプロシージャを実行する必要があります。

次のPL/SQLブロックは、特定のセッションを新しいコンシューマ・グループに切り替えます。セッション識別子(SID)は17、セッション・シリアル番号(SERIAL#)は12345、新しいコンシューマ・グループはHIGH_PRIORITYコンシューマ・グループです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS ('17', '12345',
   'HIGH_PRIORITY');
END;
/
```

SID、セッション・シリアル番号、およびセッションの現行リソース・コンシューマ・グループは、V$SESSIONビューを使用して確認できます。

関連項目:

V$SESSIONビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 手動によるリソース・コンシューマ・グループの切替え

27.2.4.1.3 ユーザーの全セッションの切替え

SWITCH_CONSUMER_GROUP_FOR_USERプロシージャは、指定したユーザー名に関連するすべてのセッションのリソース・コンシューマ・グループを変更します。

次のPL/SQLブロックは、ユーザーHRに属しているすべてのセッションをLOW_GROUPコンシューマ・グループに切り替えます。

```
BEGIN
  DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER ('HR',
    'LOW_GROUP');
END;
/
```

親トピック: 手動によるリソース・コンシューマ・グループの切替え

27.2.4.2 ユーザーまたはアプリケーションに対する手動によるコンシューマ・グループの切替えの有効化

管理者は、ユーザーがDBMS_SESSIONパッケージのSWITCH_CURRENT_CONSUMER_GROUPプロシージャを使用して、現在のコンシューマ・グループを切り替えられるように、スイッチ特権をユーザーに付与できます。

ユーザーが対話型セッション(SQL*Plusなど)でこのプロシージャを実行するか、アプリケーションでこのプロシージャをコールすることで、セッションを切り替え、実質的にその優先度を動的に変更できます。

ユーザーは、SWITCH_CURRENT_CONSUMER_GROUPプロシージャを使用して、スイッチ特権があるコンシューマ・グループにのみ切り替えることができます。他のプロシージャからこのプロシージャがコールされた場合、ユーザーはコール側のプロシージャの所有者がスイッチ特権を持っているコンシューマ・グループに切り替えることができます。

このプロシージャのパラメータは、次のとおりです。

パラメータ	説明
NEW_CONSUMER_GROUP	切替え先のコンシューマ・グループ。
OLD_CONSUMER_GROUP	切替え元のコンシューマ・グループの名前が戻ります。このパラメータは、後で元のコンシュー

パラメータ	説明
	マ・グループに再び切り替えるときに使用できます。
INITIAL_GROUP_ON_ERROR	切替えエラー発生時の動作を制御します。 TRUE に設定すると、エラーが発生した場合にユーザーは初期コンシューマ・グループに切り替わります。 FALSE の場合は、そのままエラーが出力されます。

次のSQL*Plusセッションは、新しいコンシューマ・グループへの切替え方法を示しています。出力パラメータold_groupの値が出力されているため、変更前のコンシューマ・グループ名がどのように保存されているかがわかります。

```
SET serveroutput on
DECLARE
    old_group varchar2(30);
BEGIN
  DBMS_SESSION.SWITCH_CURRENT_CONSUMER_GROUP('BATCH_GROUP', old_group, FALSE);
  DBMS_OUTPUT.PUT_LINE('OLD GROUP = ' || old_group);
END;
/
```

次の行が出力されます。

```
OLD GROUP = OLTP_GROUP
```

リソース・マネージャでは、セッションをすでに配置されているコンシューマ・グループに切り替えるために SWITCH_CURRENT_CONSUMER_GROUPプロシージャがコールされている場合でも、切替えが発生したとみなされます。

ノート:

リソース・マネージャは、一般的なデータベース・ユーザー名を使用してアプリケーションにログインしている環境でも機能します。DBMS_SESSION パッケージは、セッションの開始時、または特定のモジュールがコールされたときにセッションのコンシューマ・グループ割当てを切り替えるために使用できます。

関連項目:

- 「スイッチ特権の付与と取消し」
- DBMS_SESSIONパッケージのその他の例と詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: リソース・コンシューマ・グループの切替え

27.2.5 コンシューマ・グループの自動切替えの指定

特定の条件を満たしたときに、セッションを別のコンシューマ・グループに自動的に切り替えるように、リソース・マネージャを構成できます。

自動切替えは、セッション属性が変更された結果、新規マッピング・ルールが有効になった場合、または、セッションが、そのコンシューマ・グループに設定されているCPU、物理I/Oまたは論理I/Oリソースの使用制限を超えた場合、またはそのコンシューマ・グループに設定されている経過時間制限を超えた場合に行われることがあります。

- マッピング・ルールを使用した自動切替えの指定
  セッションの実行中にセッション属性が変更されると、コンシューマ・グループへのセッションのマッピング・ルールが再評価されます。新しいルールが有効になると、セッションが別のコンシューマ・グループに移動されることがあります。
- リソース制限の設定による自動切替えの指定
  指定された制限を超えるCPUリソース、物理I/Oリソースまたは論理I/Oリソースを使用する、リソース集中型のセッションまたはコールを管理できます。リソース集中型のセッションはSQL問合せ、リソース集中型のコールはPL/SQLコールです。

親トピック: リソース・コンシューマ・グループへのセッションの割当て

27.2.5.1 マッピング・ルールを使用した自動切替えの指定

セッションの実行中にセッション属性が変更されると、コンシューマ・グループへのセッションのマッピング・ルールが再評価されます。新しいルールが有効になると、セッションが別のコンシューマ・グループに移動されることがあります。

詳細は、「コンシューマ・グループへのセッションのマッピング・ルールの指定」を参照してください。

親トピック: コンシューマ・グループの自動切替えの指定

27.2.5.2 リソース制限の設定による自動切替えの指定

指定された制限を超えるCPUリソース、物理I/Oリソースまたは論理I/Oリソースを使用する、リソース集中型のセッションまたはコールを管理できます。リソース集中型のセッションはSQL問合せ、リソース集中型のコールはPL/SQLコールです。

コンシューマ・グループに対してリソース・プラン・ディレクティブを作成する場合は、そのグループのセッションに対してCPU、物理I/Oまたは論理I/Oリソースの使用制限を指定できます。物理I/Oと論理I/Oの制限は別々に指定できます。経過時間の制限を指定することもできます。SWITCH_FOR_CALLリソース・プラン・ディレクティブがFALSEに設定されている場合、リソース・マネージャはセッションの開始からこれらの制限を強制します。SWITCH_FOR_CALLリソース・プラン・ディレクティブがTRUEに設定されている場合、リソース・マネージャはSQL操作またはPL/SQLブロックの開始からこれらの制限を強制します。

その後、単一のセッションまたはコールが制限のいずれかを超過した場合に実行する処理を指定できます。次のような処理が考えられます。

- セッションを指定のコンシューマ・グループに動的に切り替えます。

  通常、ターゲット・コンシューマ・グループは、より低いリソース割当てのコンシューマ・グループです。
- セッションは終了します。
- セッションの現行のSQL文を終了します。
- セッションに関する情報を記録しますが、セッションに関するその他の処理は実行しません。

このタイプのセッション自動切替えに関連するリソース・プラン・ディレクティブの属性は、次のとおりです。

- SWITCH_GROUP
- SWITCH_TIME
- SWITCH_ESTIMATE
- SWITCH_IO_MEGABYTES
- SWITCH_IO_REQS

- SWITCH_FOR_CALL
- SWITCH_IO_LOGICAL
- SWITCH_ELAPSED_TIME

これらの属性の詳細は、「リソース・プラン・ディレクティブの作成」を参照してください。

切替えは、実行中でリソースを使用しているセッション(ユーザー入力またはCPUサイクルを待機中でないセッション)に対して行われます。セッションが切り替えられた後は、アイドル状態になるまでターゲット・コンシューマ・グループで継続され、アイドル状態になると、元のコンシューマ・グループに切替えが戻されます。ただし、SWITCH_FOR_CALLがTRUEに設定されている場合、リソース・マネージャは、セッションがアイドル状態になるまで待機して元のリソース・コンシューマ・グループにセッションを戻す処理を行いません。かわりに、現在のトップレベルのコールが完了すると、セッションは戻されます。PL/SQLの場合、トップレベルのコールは、1つのコールとして処理されるPL/SQLブロック全体です。SQLの場合、トップレベルのコールは、個々のSQL文です。

SWITCH_FOR_CALLは、中間層のサーバーがセッション・プーリングを使用している3層アプリケーションの場合に有効です。

新しいグループのアクティブ・セッション・プールが一杯の場合も、切り替えたセッションは引き続き実行できます。このような状況では、コンシューマ・グループは、アクティブ・セッション・プールで指定された数より多いセッションを実行できます。

SWITCH_FOR_CALLがFALSEの場合、リソース・マネージャは、コールの間隔が一定の時間を超えたセッションをアイドル状態とみなします。この時間間隔は数秒で、設定できません。

次に、リソース制限に基づいた自動切替えの例を示します。これらの例を実行する前に、ペンディング・エリアを作成する必要があります。

例1

次のPL/SQLブロックはOLTPグループのリソース・プラン・ディレクティブを作成し、そのディレクティブはセッション内のコールのCPU時間が5秒を超えるとそのグループのすべてのセッションをLOW_GROUPコンシューマ・グループに切り替えます。この例は、予想外に時間がかかる問合せによって過剰なリソースが使用されないようにします。通常、切替え先のコンシューマ・グループは、より低いリソース割当てのコンシューマ・グループです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN             => 'DAYTIME',
   GROUP_OR_SUBPLAN => 'OLTP',
   COMMENT          => 'OLTP group',
   MGMT_P1          => 75,
   SWITCH_GROUP     => 'LOW_GROUP',
   SWITCH_TIME      => 5);
END;
/
```

例2

次のPL/SQLブロックはOLTPグループのリソース・プラン・ディレクティブを作成し、そのディレクティブはセッションで物理I/O要求が10,000を超えるか転送されたデータが2,500 MBを超えると、そのグループのすべてのセッションを一時的にLOW_GROUPコンシューマ・グループに切り替えます。問題を引き起こしているトップレベルのコールが完了すると、セッションは元のグループに戻されます。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN                => 'DAYTIME',
   GROUP_OR_SUBPLAN    => 'OLTP',
   COMMENT             => 'OLTP group',
   MGMT_P1             => 75,
   SWITCH_GROUP        => 'LOW_GROUP',
   SWITCH_IO_REQS      => 10000,
```

```
   SWITCH_IO_MEGABYTES => 2500,
   SWITCH_FOR_CALL     => TRUE);
END;
/
```

例3

次のPL/SQLブロックはREPORTINGグループのリソース・プラン・ディレクティブを作成し、そのディレクティブはCPUタイムが60秒を超えるすべてのセッションを終了します。この例は、リソース集中型の問合せによって過剰なリソースが使用されないようにします。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN             => 'DAYTIME',
   GROUP_OR_SUBPLAN => 'REPORTING',
   COMMENT          => 'Reporting group',
   MGMT_P1          => 75,
   SWITCH_GROUP     => 'KILL_SESSION',
   SWITCH_TIME      => 60);
END;
/
```

この例では、予約済のコンシューマ・グループ名KILL_SESSIONがSWITCH_GROUPで指定されています。そのため、切替え基準が満たされると、セッションは終了します。その他の予約済のコンシューマ・グループ名は、CANCEL_SQLとLOG_ONLYです。CANCEL_SQLを指定した場合、切替え基準が満たされると現在のコールは取り消されますが、セッションは終了しません。LOG_ONLYを指定した場合、セッションに関する情報がリアルタイムSQL監視で記録されますが、セッションに関する特定の処理は実行されません。

例4

次のPL/SQLブロックはOLTPグループのリソース・プラン・ディレクティブを作成し、そのディレクティブはセッションで論理I/O要求が100を超えると、そのグループのすべてのセッションを一時的にLOW_GROUPコンシューマ・グループに切り替えます。問題を引き起こしているトップレベルのコールが完了すると、セッションは元のグループに戻されます。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN                => 'DAYTIME',
   GROUP_OR_SUBPLAN    => 'OLTP',
   COMMENT             => 'OLTP group',
   MGMT_P1             => 75,
   SWITCH_GROUP        => 'LOW_GROUP',
   SWITCH_IO_LOGICAL   => 100,
   SWITCH_FOR_CALL     => TRUE);
END;
/
```

例5

次のPL/SQLブロックはOLTPグループのリソース・プラン・ディレクティブを作成し、そのディレクティブはセッション内のコールが5分(300秒)を超えると、そのグループのすべてのセッションを一時的にLOW_GROUPコンシューマ・グループに切り替えます。問題を引き起こしているトップレベルのコールが完了すると、セッションは元のグループに戻されます。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN                => 'DAYTIME',
   GROUP_OR_SUBPLAN    => 'OLTP',
   COMMENT             => 'OLTP group',
   MGMT_P1             => 75,
   SWITCH_GROUP        => 'LOW_GROUP',
   SWITCH_FOR_CALL     => TRUE,
```

```
   SWITCH_ELAPSED_TIME => 300);
END;
/
```

関連項目:

- 「リソース・プラン・ディレクティブの作成」
- 論理I/Oの詳細は、「リソース・マネージャが提供するワークロード管理のソリューション」を参照してください

親トピック: コンシューマ・グループの自動切替えの指定

27.2.6 スイッチ特権の付与と取消し

ユーザーまたはアプリケーションがセッションを指定されたリソース・コンシューマ・グループに切り替えるには、スイッチ特権が必要です。

- スイッチ特権の付与と取消しについて
  DBMS_RESOURCE_MANAGER_PRIVS PL/SQLパッケージを使用すると、ユーザー、ロールまたはPUBLICにスイッチ特権を付与したり取り消すことができます。スイッチ特権を使用すると、ユーザーまたはアプリケーションが、指定されているリソース・コンシューマ・グループにセッションを切り替えることができるようになります。
- スイッチ特権の付与
  GRANT_SWITCH_CONSUMER_GROUPプロシージャを使用して、特定のコンシューマ・グループに切り替えるための権限をユーザーに付与できます。
- スイッチ特権の取消し
  REVOKE_SWITCH_CONSUMER_GROUPプロシージャを使用して、特定のコンシューマ・グループに切り替えるためのユーザーの権限を取り消すことができます。

親トピック: リソース・コンシューマ・グループへのセッションの割当て

27.2.6.1 スイッチ特権の付与と取消しについて

DBMS_RESOURCE_MANAGER_PRIVS PL/SQLパッケージを使用すると、ユーザー、ロールまたはPUBLICにスイッチ特権を付与したり取り消すことができます。スイッチ特権を使用すると、ユーザーまたはアプリケーションが、指定されているリソース・コンシューマ・グループにセッションを切り替えることができるようになります。

このパッケージでは、スイッチ特権を取り消すこともできます。次の表に、関連するパッケージ・プロシージャの一覧を示します。

プロシージャ	説明
GRANT_SWITCH_CONSUMER_GROUP	ユーザー、ロールまたは PUBLIC に、指定したリソース・コンシューマ・グループに切り替える許可を付与します。
REVOKE_SWITCH_CONSUMER_GROUP	ユーザー、ロールまたは PUBLIC から、指定したリソース・コンシューマ・グループに切り替える許可を取り消します。

OTHER_GROUPSでは、スイッチ特権がPUBLICに付与されています。したがって、すべてのユーザーに、このコンシューマ・グループに対するスイッチ特権が自動的に付与されます。

次の切替えには明示的なスイッチ特権は必要ありません。

- DBMS_RESOURCE_MANAGERパッケージのSET_CONSUMER_GROUP_MAPPINGプロシージャによって指定されたコンシューマ・グループ・マッピングがあり、このマッピングのために、セッションは、異なるコンシューマ・グループに切り替わります。「コンシューマ・グループのマッピング・ルールの作成」を参照してください。
- リソース・プラン・ディレクティブのswitch_groupパラメータの設定に基づいて切替え条件が満たされたとき、コンシューマ・グループの自動切替えが行われます。

その他のすべての場合、セッションをコンシューマ・グループに切り替えるには、明示的なスイッチ特権が必要となります。

関連項目:

- 「ユーザーまたはアプリケーションに対する手動によるコンシューマ・グループの切替えの有効化」
- 「コンシューマ・グループの自動切替えの指定」

親トピック: スイッチ特権の付与と取消し

27.2.6.2 スイッチ特権の付与

GRANT_SWITCH_CONSUMER_GROUPプロシージャを使用して、特定のコンシューマ・グループに切り替えるための権限をユーザーに付与できます。

次の例では、コンシューマ・グループOLTPに切り替える権限をユーザーSCOTTに付与しています。

```
BEGIN
  DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP (
   GRANTEE_NAME   => 'SCOTT',
   CONSUMER_GROUP => 'OLTP',
   GRANT_OPTION   =>  TRUE);
END;
/
```

ユーザーSCOTTには、OLTPのスイッチ特権を他のユーザーに付与する許可も付与されます。

特定のリソース・コンシューマ・グループに切り替えるロールに権限を付与すると、そのロールを付与されたユーザーおよびそのロールを有効にしているユーザーは、各自のセッションをそのコンシューマ・グループに切り替えることができます。

特定のコンシューマ・グループに切り替える許可をPUBLICに付与すると、だれでもそのグループに切り替えることができます。

GRANT_OPTION引数がTRUEの場合、コンシューマ・グループのスイッチ特権が付与されたユーザーは、そのコンシューマ・グループのスイッチ特権を他のユーザーに付与することもできます。

親トピック: スイッチ特権の付与と取消し

27.2.6.3 スイッチ特権の取消し

REVOKE_SWITCH_CONSUMER_GROUPプロシージャを使用して、特定のコンシューマ・グループに切り替えるためのユーザーの権限を取り消すことができます。

次の例では、コンシューマ・グループOLTPに切り替えるユーザーSCOTTの権限を取り消しています。

```
BEGIN
  DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP (
   REVOKEE_NAME   => 'SCOTT',
   CONSUMER_GROUP => 'OLTP');
END;
/
```

特定のコンシューマ・グループに対するユーザーのスイッチ特権を取り消した場合は、そのユーザーがそのコンシューマ・グループに手動で切り替えようとすると失敗します。このユーザーのセッションは、自動的にOTHER_GROUPSに割り当てられます。

コンシューマ・グループに対するスイッチ特権をロールから取り消すと、そのロールを介してのみコンシューマ・グループのスイッチ特権を与えられているユーザーは、そのコンシューマ・グループに切り替えることができなくなります。

コンシューマ・グループに対するスイッチ特権をPUBLICから取り消すと、直接またはロールを介してスイッチ特権を明示的に割り当てられているユーザー以外は、そのコンシューマ・グループに切り替えることができなくなります。

親トピック: スイッチ特権の付与と取消し

27.3 リソース・マネージャによって管理されるリソースのタイプ

リソース・プラン・ディレクティブは、リソース・コンシューマ・グループまたはサブプランへのリソースの割当て方法を指定します。各ディレクティブでは、リソースをそのコンシューマ・グループまたはサブプランに割り当てるための複数の異なる方法を指定できます。

- CPU
  CPUリソースを管理するために、リソース・マネージャはリソースをコンシューマ・グループに割り当て、割り当てられたものの使用されなかったCPUリソースを再配分します。特定のコンシューマ・グループに割り当てることができるCPUリソースの量に制限を設定することもできます。
- Exadata I/O
  管理属性を使用すると、Exadata I/OのCPUリソース割当てを指定できます。
- パラレル実行サーバー
  リソース・マネージャでは、データベースに対して使用可能なパラレル実行サーバーの使用を管理できます。
- プログラム・グローバル領域(PGA)
  PGAリソースを管理するために、リソース・マネージャは特定のコンシューマ・グループ内の各セッションに割り当てることのできるPGAメモリーの量を制限できます。
- リソース集中型の問合せ
  リソース集中型のセッションおよびコールは、適切に管理されない場合、パフォーマンス全体に悪影響を及ぼす可能性があります。リソース・マネージャでは、セッションまたはコールが指定量を超えるCPU、物理I/O、論理I/Oまたは経過時間を使用したときに処理を行うことができます。リソース・マネージャでは、セッションまたはコールをCPUの割当てが少ないコンシューマ・グループに切り替えたり、セッションまたはコールを終了できます。
- キューイングを備えたアクティブ・セッション・プール
  コンシューマ・グループ内で同時にアクティブにできるセッションの最大数を制御できます。この最大数によって、アクティブ・セッション・プールが定義されます。
- UNDOプール
  UNDOプールは、コンシューマ・グループごとに指定できます。UNDOプールは、コンシューマ・グループが生成できる、コミットされていないトランザクションに対するUNDO合計量を制御します。
- アイドル時間制限
  セッションがアイドル状態のままでいられる時間(セッションが終了されるまでの時間)を指定できます。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.3.1 CPU

CPUリソースを管理するために、リソース・マネージャはリソースをコンシューマ・グループに割り当て、割り当てられたものの使用されなかったCPUリソースを再配分します。特定のコンシューマ・グループに割り当てることができるCPUリソースの量に制限を設定することもできます。

- 管理属性
  管理属性では、コンシューマ・グループおよびサブプランの間でのCPUリソースの割当て方法を指定します。
- 使用率制限
  UTILIZATION_LIMIT属性は、リソース・コンシューマ・グループのCPU使用率に絶対上限を設定する場合に使用します。この絶対的な制限により、プラン内でのCPUの再配布は無視されます。

親トピック: リソース・マネージャによって管理されるリソースのタイプ

27.3.1.1 管理属性

管理属性では、コンシューマ・グループおよびサブプランの間でのCPUリソースの割当て方法を指定します。

複数レベル(最大8レベル)のCPUリソース割当てによって、プラン内でCPU使用の優先度を設定できます。レベル2のコンシューマ・グループとサブプランは、レベル1に割り当てられなかったリソース、またはレベル1に割り当てられたがレベル1のコンシューマ・グループまたはサブプランで完全には使用されなかったリソースを取得します。同様に、レベル3のリソース・コンシューマには、レベル1と2の割当ての一部が残っている場合のみ、リソースが割り当てられます。リソース・プランには、各コンシューマ・グループのリソース割当てのディレクティブが含まれ、割合およびレベルで構成されます。複数のレベルを使用すると、優先度を設定できるのみでなく、すべての主リソースと残りのリソースの使用方法を明示的に指定できます。

管理属性MGMT_Pn(nは1から8の整数)は、複数レベルのCPUリソース割当てを指定する場合に使用します。たとえば、MGMT_P1ディレクティブ属性を使用してレベル1でCPUリソース割当てを指定し、MGMT_P2ディレクティブ属性を使用してレベル2でリソース割当てを指定します。

パラレル・ステートメント・キューイングを制御するには、管理属性とともに、並列度制限やパラレル・サーバー制限などのパラレル・ステートメント・ディレクティブ属性を使用します。パラレル・ステートメント・キューイングが使用されている場合、管理属性はどのコンシューマ・グループが次のパラレル・ステートメントを発行できるかを決定するために使用されます。たとえば、コンシューマ・グループのMGMT_P1ディレクティブ属性を80に設定した場合、そのグループが次のパラレル・ステートメントを発行する確率は80%になります。

関連項目:

パラレル・ステートメント・キューイングの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

表27-1に、3つのレベルがある単純なリソース・プランを示します。

表27-1 3つのレベルがある単純なリソース・プラン

コンシューマ・グループ	レベル1のCPU割当て	レベル2のCPU割当て	レベル3のCPU割当て
HIGH_GROUP	80%		
LOW_GROUP		50%	
MAINT_SUBPLAN		50%	
OTHER_GROUPS			100%

優先度の高いアプリケーションは、CPUの80%が割り当てられるHIGH_GROUPで実行されます。HIGH_GROUPはレベル1で

あるため、CPU使用率の優先度がありますが、最大でCPUの80%のみです。CPUの残りの20%は、レベル2のLOW_GROUPとMAINT_SUPLANによって50%ずつ共有されます。レベル1と2で未使用の割当ては、レベル3のOTHER_GROUPSで使用できます。OTHER_GROUPSには同じレベルに兄弟関係のあるコンシューマ・グループやサブプランがないため、100%が指定されます。

特定のレベル内で、CPU割当ては固定ではありません。特定のコンシューマ・グループまたはサブプランでの負荷に余裕がある場合、残りのCPUを残りのコンシューマ・グループまたはサブプランに割り当てることができます。したがって、レベルが1つのみの場合、あるコンシューマ・グループまたはサブプランが使用していない割当ては、兄弟関係のある他のコンシューマ・グループまたはサブプランに再配分できます。複数のレベルがある場合、未使用の割当ては次のレベルのコンシューマ・グループまたはサブプランに再配分されます。最後のレベルに未使用の割当てがある場合、それらの割当ては他のすべてのレベルに、それぞれに指定された割当てに比例して再配分できます。

未使用の割当ての1つのレベルから他のレベルへの再配分の例として、特定の期間で、HIGH_GROUPがCPUの25%のみを使用した場合、LOW_GROUPとMAINT_SUBPLANで75%を共有できます。レベル2での75%の未使用部分は、レベル3のOTHER_GROUPSが使用できるようになります。ただし、レベル3でOTHER_GROUPSにセッション・アクティビティがない場合は、レベル2での75%はプランの他のすべてのコンシューマ・グループおよびサブプランに比例で再配分できます。

親トピック: CPU

27.3.1.2 使用率制限

UTILIZATION_LIMIT属性は、リソース・コンシューマ・グループのCPU使用率に絶対上限を設定する場合に使用します。この絶対的な制限により、プラン内でのCPUの再配布は無視されます。

前の使用例で、他の部分が非アクティブであるために、LOW_GROUPがCPUの90%を取得したとします。重要ではないセッションにCPUを大量に使用させないために、LOW_GROUPがサーバーの90%を使用できないようにするとします。リソース・プラン・ディレクティブのUTILIZATION_LIMIT属性を使用して、この状況を防ぐことができます。

UTILIZATION_LIMIT属性の設定はオプションです。コンシューマ・グループに対するこの属性を省略した場合は、コンシューマ・グループが使用できるCPUの量に制限はありません。このため、他のすべてのアプリケーションがアイドル状態である場合、UTILIZATION_LIMITが設定されていないコンシューマ・グループに100%のCPUリソースを割り当てることができます。

レベルの制限を使用しないで、コンシューマ・グループのCPU使用率を制限する唯一の手段としてUTILIZATION_LIMIT属性を使用することもできます。

表27-2に、前のプランを変更したプランを示します。このプランでは、UTILIZATION_LIMITを使用して、CPU使用率の上限をLOW_GROUPの場合は75%、MAINT_SUBPLANの場合は50%、OTHER_GROUPSの場合は75%に設定します。(使用率の制限の合計が100%を超える場合があります。各制限は個別に適用されます。)

表27-2 使用率制限が適用された3つのレベルがあるリソース・プラン

コンシューマ・グループ	レベル1のCPU割当て	レベル2のCPU割当て	レベル3のCPU割当て	使用率制限
HIGH_GROUP	80%			
LOW_GROUP		50%		75%
MAINT_SUBPLAN		50%		50%

コンシューマ・グループ	レベル1のCPU割当て	レベル2のCPU割当て	レベル3のCPU割当て	使用率制限
OTHER_GROUPS			100%	75%

表27-2で説明する例では、HIGH_GROUPが特定の時間にCPUの10%のみを使用している場合、残りの90%をLOW_GROUPおよびMAINT_SUBPLANのコンシューマ・グループがレベル2で使用できます。LOW_GROUPがCPUの20%のみを使用している場合、70%をMAINT_SUBPLANに割り当てることができます。ただし、MAINT_SUBPLANではUTILIZATION_LIMITを50%に設定します。このため、より多くのCPUリソースが使用可能である場合も、サブプランMAINT_SUBPLANに属するコンシューマ・グループには50%を超えたCPUを割り当てることができません。

サブプランとそのサブプランに含まれているコンシューマ・グループの両方にUTILIZATION_LIMITを設定できます。このような場合、コンシューマ・グループの制限はサブプランとそのコンシューマ・グループに指定されている制限を使用して算出されます。たとえば、MAINT_SUBPLANには、コンシューマ・グループMAINT_GROUP1とMAINT_GROUP2が含まれています。MAINT_GROUP1のUTILIZATION_LIMITは40%に設定されています。ただし、MAINT_SUBPLANの制限は50%に設定されています。このため、コンシューマ・グループMAINT_GROUP1の制限は50%の40%、つまり20%になります。コンシューマ・グループとそのグループが属するサブプランの両方に制限が指定されている場合に、コンシューマ・グループのUTILIZATION_LIMITを計算する方法の例については、「例4 - コンシューマ・グループおよびサブプランに使用率制限を指定する」を参照してください。

関連項目:

- 「リソース・プラン・ディレクティブの作成」
- 「各種の方法を組み合せたOracle Database Resource Managerの例」

親トピック: CPU

27.3.2 Exadata I/O

管理属性を使用すると、Exadata I/OのCPUリソース割当てを指定できます。

関連項目:

Exadataの入出力に関する管理属性の使用の詳細は、Exadataのドキュメントを参照してください。

親トピック: リソース・マネージャによって管理されるリソースのタイプ

27.3.3 パラレル実行サーバー

リソース・マネージャでは、データベースに対して使用可能なパラレル実行サーバーの使用を管理できます。

- 並列度制限
  管理者は、1つのコンシューマ・グループ内での操作に対して最大並列度を制限できます。
  PARALLEL_DEGREE_LIMIT_P1ディレクティブ属性は、コンシューマ・グループの並列度を指定する場合に使用します。
- パラレル・サーバー制限
  PARALLEL_SERVER_LIMITディレクティブ属性は、特定のコンシューマ・グループが使用できるパラレル実行サー

バー・プールの最大割合を指定する場合に使用します。特定のコンシューマ・グループで使用されているパラレル実行サーバーの数は、そのコンシューマ・グループのすべてのセッションで使用されているパラレル実行サーバーの合計となります。

- パラレル・キューのタイムアウト
  PARALLEL_QUEUE_TIMEOUTディレクティブ属性を使用すると、パラレル・ステートメントがパラレル・ステートメント・キュー内でタイムアウトになるまで待機する最大時間を秒単位で指定できます。

親トピック: リソース・マネージャによって管理されるリソースのタイプ

27.3.3.1 並列度制限

管理者は、1つのコンシューマ・グループ内での操作に対して最大並列度を制限できます。
PARALLEL_DEGREE_LIMIT_P1ディレクティブ属性は、コンシューマ・グループの並列度を指定する場合に使用します。

並列度制限は、コンシューマ・グループ内の1つの操作に対して適用されます(コンシューマ・グループ内のすべての操作の合計並列度は制限されません)。ただし、PARALLEL_DEGREE_LIMIT_P1とPARALLEL_SERVER_LIMITの両方のディレクティブ属性を組み合せて、任意の制御を行うことは可能です。PARALLEL_SERVER_LIMIT属性の詳細は、「パラレル・サーバー制限」を参照してください。

関連項目:

プロデューサ/コンシューマ操作での並列度の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: パラレル実行サーバー

27.3.3.2 パラレル・サーバー制限

PARALLEL_SERVER_LIMITディレクティブ属性は、特定のコンシューマ・グループが使用できるパラレル実行サーバー・プールの最大割合を指定する場合に使用します。特定のコンシューマ・グループで使用されているパラレル実行サーバーの数は、そのコンシューマ・グループのすべてのセッションで使用されているパラレル実行サーバーの合計となります。

単一のコンシューマ・グループが数多くのパラレル・ステートメントを起動することによって、使用可能なすべてのパラレル実行サーバーが使用されることがあります。この場合、別のコンシューマ・グループから優先度の高いパラレル・ステートメントが実行されても、このグループにパラレル実行サーバーを割り当てることができません。このような状況を回避するには、特定のコンシューマ・グループが使用できるパラレル実行サーバーの数を制限します。優先度の高いコンシューマ・グループに対してディレクティブPARALLEL_STMT_CRITICALをBYPASS_QUEUEに設定し、そのコンシューマ・グループのパラレル・ステートメントがパラレル・ステートメント・キューを無視するように設定することもできます。

たとえば、PARALLEL_SERVERS_TARGET初期化パラメータによって設定されたパラレル実行サーバーの合計数が32であり、コンシューマ・グループMY_GROUPのPARALLEL_SERVER_LIMITディレクティブ属性が50%に設定されているとします。このコンシューマ・グループは最大で32の50%、つまり16台のパラレル実行サーバーを使用できます。

リソース・プランで管理属性(MGMT_P1、MGMT_P2など)が設定された場合は、管理属性ごとに別々のパラレル・ステートメント・キューが先入れ先出し(FIFO)キューとして管理されます。

リソース・プランで管理属性が設定されない場合は、1つのパラレル・ステートメント・キューがFIFOキューとして管理されます。

Oracle Real Application Clusters (Oracle RAC)環境の場合、パラレル実行サーバーのターゲット数はすべてのOracle RACインスタンスでの(PARALLEL_SERVER_LIMIT * PARALLEL_SERVERS_TARGET /100)の合計となります。あるコンシューマ・グループによって、前述のように算出した数以上のパラレル実行サーバーが使用されている場合、そのコン

シューマ・グループは制限値を超えているため、そのコンシューマ・グループのパラレル・ステートメントはキューされます。

コンシューマ・グループにOracle RACデータベース内で実行されているパラレル・ステートメントがない場合、最初のパラレル・ステートメントはPARALLEL_SERVER_LIMITで指定された制限を超えることができます。

ノート:

Oracle Real Application Clusters (Oracle RAC)環境では、PARALLEL_SERVER_LIMIT 属性が単一のインスタンスではなくクラスタ全体に適用されます。

- パラレル・サーバー制限を使用したパラレル・ステートメント・キューイングの管理
  PARALLEL_SERVER_LIMIT属性を使用すると、いつコンシューマ・グループからパラレル・ステートメントをキューできるかを指定できます。Oracle Databaseには、コンシューマ・グループごとにパラレル・ステートメント・キューが保持されます。

関連項目:

- 「リソース・プラン・ディレクティブの作成」
- 「パラレル・サーバー制限を使用したパラレル・ステートメント・キューイングの管理」
- パラレル・ステートメント・キューイングの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

親トピック: パラレル実行サーバー

27.3.3.2.1 パラレル・サーバー制限を使用したパラレル・ステートメント・キューイングの管理

PARALLEL_SERVER_LIMIT属性を使用すると、いつコンシューマ・グループからパラレル・ステートメントをキューできるかを指定できます。Oracle Databaseには、コンシューマ・グループごとにパラレル・ステートメント・キューが保持されます。

コンシューマ・グループからのパラレル・ステートメントは実行されず、そのかわり、次の条件が満たされた場合に、そのコンシューマ・グループのパラレル・ステートメント・キューに追加されます。

- PARALLEL_DEGREE_POLICYがAUTOに設定されています。

  この初期化パラメータをAUTOに設定すると、自動並列度(Auto DOP)、パラレル・ステートメント・キューイングおよびインメモリー・パラレル実行が有効になります。

  PARALLEL_DEGREE_POLICYがMANUALまたはLIMITEDに設定されているパラレル・ステートメントはただちに実行され、パラレル・ステートメント・キューに追加されません。

- すべてのコンシューマ・グループのアクティブなパラレル実行サーバー数が、PARALLEL_SERVERS_TARGET初期化パラメータの設定を超えています。この条件は、PARALLEL_SERVER_LIMITを指定しているかどうかに関係なく適用されます。PARALLEL_SERVER_LIMITが指定されていない場合は、100%にデフォルト設定されます。

- コンシューマ・グループのアクティブなパラレル実行サーバーの合計数およびパラレル・ステートメントの並列度が、アクティブなパラレル実行サーバーのターゲット数を超えています。

  アクティブなパラレル実行サーバーのターゲット数は、次のように計算されます。

  PARALLEL_SERVER_LIMIT/100 * PARALLEL_SERVERS_TARGET

ノート:

すべてのセッションでパラレル実行サーバーの使用状況が監視されますが、設定したパラレル実行サーバー・ディレクティブ属性は、パラレル・ステートメント・キューイングが有効になっている(PARALLEL_DEGREE_POLICY が AUTO に設定されている)セッションにのみ影響を与えます。セッションの PARALLEL_DEGREE_POLICY が MANUAL に設定されている場合、このセッションからのパラレル・ステートメントはキューされません。ただし、このようなセッションで使用されているパラレル実行サーバーは、PARALLEL_SERVER_LIMIT の制限の決定に使用されるカウントに数えられます。この制限を超えても、このセッションからのパラレル・ステートメントはキューされません。

関連項目:

「パラレル・サーバー制限」

親トピック: パラレル・サーバー制限

27.3.3.3 パラレル・キューのタイムアウト

PARALLEL_QUEUE_TIMEOUTディレクティブ属性を使用すると、パラレル・ステートメントがパラレル・ステートメント・キュー内でタイムアウトになるまで待機する最大時間を秒単位で指定できます。

パラレル・ステートメント・キューイングを使用すると、データベースにパラレル・ステートメントを実行するほどのリソースがない場合、そのステートメントは必要なリソースが使用可能になるまでキューで待機します。ただし、パラレル・ステートメントが意図した時間よりも長くパラレル・ステートメント・キューで待機する場合があります。このような状況を回避するには、パラレル・ステートメントがパラレル・ステートメント・キューで待機できる最大時間を指定します。

PARALLEL_QUEUE_TIMEOUT属性は、コンシューマ・グループごとに設定できます。この属性は、リソース・プランの他の管理属性(MGMT_P1、MGMT_P2など)を指定していなくても適用できます。

関連項目:

パラレル・ステートメント・キューイングの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

ノート:

パラレル・ステートメント・キューはクラスタ全体で機能するため、パラレル・ステートメント・キューに関連するすべてのディレクティブもクラスタ全体で機能します。

タイムアウトになったパラレル文の処理方法は、コンシューマ・グループごとにPQ_TIMEOUT_ACTION属性を設定することで制御できます。この属性は、次の値に設定できます。

- CANCEL - 文の実行はエラーORA-07454で終了します。これは、タイムアウトになったパラレル文のデフォルトの動作です。
- RUN - 文は即時実行されます。文を即時実行するためのパラレル・サーバーが不足している場合は、並列度を下げて実行するように文がダウングレードされます。

関連項目:

すべてのパラレル実行サーバー・ディレクティブ属性を組み合せて使用する方法の詳細は、「ディレクティブ属性を使用したパラレル・ステートメントの管理の例」を参照してください

親トピック: パラレル実行サーバー

27.3.4 プログラム・グローバル領域(PGA)

PGAリソースを管理するために、リソース・マネージャは、特定のコンシューマ・グループの各セッションに割当て可能なPGAメモリー量を制限できます。

コンシューマ・グループ内の各セッションのPGAリソースを制限するには、パッケージ・プロシージャDBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE内でsession_pga_limitパラメータを設定します。このパラメータの値は、コンシューマ・グループの各セッションに許可されるPGAメモリー容量の最大値(MB単位)です。セッションがそのコンシューマ・グループに設定されている制限を超過すると、ORA-10260エラーが発生します。この制限には、パラレル問合せスレーブおよびジョブ・キュー・プロセスが含まれます。

たとえば、不適切に記述されたPL/SQLコードは制限なしのPGA量を消費することがあります。session_pga_limitパラメータを使用してPL/SQLコードを実行するセッションを制限して、これらのセッションが過剰な量のPGAリソースを使用しないようにすることができます。

次の表は、PGA制限のある単純なリソース・プランを示しています。

表27-3 PGA制限のある単純なリソース・プラン

コンシューマ・グループ	session_pga_limit値
HIGH_GROUP	20
LOW_GROUP	10
MAINT_SUBPLAN	Null (制限なし)
OTHER_GROUPS	Null (制限なし)

このリソース・プランでは、優先度の高いアプリケーションがHIGH_GROUP内で実行され、そのグループ内の各セッションは20 MBのPGAリソースに制限されます。LOW_GROUP内の低優先度アプリケーションによって使用されるセッションは、10 MBのPGAリソースに制限されます。MAINT_SUPLAN内のメンテナンス・ジョブに使用されるセッションおよびOTHER_GROUPS内のその他のセッションは、無制限のPGAリソースを使用できます。

ノート:

PGA_AGGREGATE_LIMIT 初期化パラメータでインスタンス全体の PGA 使用率を制限できます。

親トピック: リソース・マネージャによって管理されるリソースのタイプ

27.3.5 リソース集中型の問合せ

リソース集中型のセッションおよびコールは、適切に管理されない場合、パフォーマンス全体に悪影響を及ぼす可能性があります。リソース・マネージャでは、セッションまたはコールが指定量を超えるCPU、物理I/O、論理I/Oまたは経過時間を使用したときに処理を行うことができます。リソース・マネージャでは、セッションまたはコールをCPUの割当てが少ないコンシューマ・グループに切り替えたり、セッションまたはコールを終了できます。

ノート:

Oracle Database 12c リリース 2 (12.2)以降では、リソース・マネージャは特定のコンシューマ・グループ内の各セッションに割り当てることのできる PGA メモリーの量を制限できます。

- コンシューマ・グループの自動切替え
  指定のコンシューマ・グループにセッションを自動的に切り替えるための基準を指定して、リソース割当てを制御できます。
- SQLの取消しとセッションの終了
  リソース・マネージャを使用すると、CPUとI/Oなどのシステム・リソースの使用量に基づいて、長時間実行SQL問合せを取り消したり、長時間実行セッションを終了することができます。
- 実行時間制限
  ある操作に許可される最大の実行時間を指定できます。

関連項目:

プログラム・グローバル領域(PGA)

親トピック: リソース・マネージャによって管理されるリソースのタイプ

27.3.5.1 コンシューマ・グループの自動切替え

指定のコンシューマ・グループにセッションを自動的に切り替えるための基準を指定して、リソース割当てを制御できます。

通常、この方法は、グループ内の典型的なセッションに対するリソース使用予測を上回ったセッションを、優先度の高いコンシューマ・グループ(システム・リソースを高い比率で使用するグループ)から優先度の低いコンシューマ・グループに切り替えるために使用します。

詳細は、「リソース制限の設定による自動切替えの指定」を参照してください。

親トピック: リソース集中型の問合せ

27.3.5.2 SQLの取消しとセッションの終了

リソース・マネージャを使用すると、CPUとI/Oなどのシステム・リソースの使用量に基づいて、長時間実行SQL問合せを取り消したり、長時間実行セッションを終了することができます。

リソース・マネージャによって取り消されたSQL問合せは、DBMS_SQLQパッケージのサブプログラムを使用することで、それらの問合せを再実行できないように隔離を構成できます。

関連項目:

- システム・リソースの消費に基づいてSQL問合せの取消しまたはセッションの終了を行うためのリソース・マネージャの構

成方法の詳細は、リソース制限の設定による自動切替えの指定を参照してください。

- DBMS_SQLQパッケージのサブプログラムを使用してSQL問合せの隔離設定を構成する方法の詳細は、過剰なシステム・リソースを使用するSQL文の実行計画の隔離を参照してください。

親トピック: リソース集中型の問合せ

27.3.5.3 実行時間制限

ある操作に許可される最大の実行時間を指定できます。

ある操作についてデータベースが見積った実行時間が、指定された最大実行時間より長い場合、その操作はエラーを出力して終了します。このエラーはトラップ可能であり、操作は再スケジュールできます。

親トピック: リソース集中型の問合せ

27.3.6 キューイングを備えたアクティブ・セッション・プール

コンシューマ・グループ内で同時にアクティブにできるセッションの最大数を制御できます。この最大数によって、アクティブ・セッション・プールが定義されます。

アクティブ・セッションとは、現在、トランザクションまたはSQL文を処理しているセッションです。具体的には、アクティブ・セッションは、ユーザー・エンキューを保持した状態でトランザクション内にあるか、またはオープン・カーソルを持ちつつ5秒を超える長さにわたりアイドル状態になっていないかのいずれかです。アクティブ・セッションは、セッションがブロックされている場合であっても(たとえば、I/O要求の完了の待機中など)、アクティブであるとみなされます。アクティブ・セッション・プールが一杯になると、コールの処理を試行しているセッションはキューに送られます。アクティブ・セッションが完了すると、キュー内の最初のセッションがキューから削除されて実行がスケジュールされます。また、実行キュー内のセッションがタイムアウトして、そのコールがエラーで終了されるまでの時間も指定できます。

OLTPワークロードにはアクティブ・セッションの制限を使用しないでください。また、接続プーリングまたはパラレル・ステートメント・キューイングを実装する目的でアクティブ・セッションの制限を使用しないでください。

パラレル・ステートメントを管理するには、PARALLEL_SERVER_LIMIT属性および管理属性(MGMT_P1、MGMT_P2など)とともに、パラレル・ステートメント・キューイングを使用する必要があります。

親トピック: リソース・マネージャによって管理されるリソースのタイプ

27.3.7 UNDOプール

UNDOプールは、コンシューマ・グループごとに指定できます。UNDOプールは、コンシューマ・グループが生成できる、コミットされていないトランザクションに対するUNDO合計量を制御します。

コンシューマ・グループが生成するUNDOの合計量がそのUNDO制限を超えると、そのUNDOを生成している現行のデータ操作言語(DML)文が終了します。コンシューマ・グループの他のメンバーは、UNDO領域がプールから解放されるまで、新たにデータ操作を実行できません。

親トピック: リソース・マネージャによって管理されるリソースのタイプ

27.3.8 アイドル時間制限

セッションがアイドル状態のままでいられる時間(セッションが終了されるまでの時間)を指定できます。

また、アイドル状態で他のセッションをブロックしているセッションに適用される、さらに厳密なアイドル時間制限も指定できます。

27.4 単純なリソース・プランの作成

CREATE_SIMPLE_PLANプロシージャを使用すると、多くの状況に対応できる単純なリソース・プランを迅速に作成できます。

このプロシージャでは、1回のプロシージャ・コールの実行で、コンシューマ・グループを作成し、そのグループにリソースを割り当てることができます。このプロシージャを使用する際は、ペンディング・エリアの作成、各コンシューマ・グループの個別作成、リソース・プラン・ディレクティブの指定のために、後述のプロシージャをコールする必要はありません。

CREATE_SIMPLE_PLANプロシージャには、次の引数を指定します。

パラメータ	説明
SIMPLE_PLAN	計画の名前。
CONSUMER_GROUP1	1 番目のグループのコンシューマ・グループ名
GROUP1_PERCENT	このグループに割り当てる CPU リソース
CONSUMER_GROUP2	2 番目のグループのコンシューマ・グループ名
GROUP2_PERCENT	このグループに割り当てる CPU リソース
CONSUMER_GROUP3	3 番目のグループのコンシューマ・グループ名
GROUP3_PERCENT	このグループに割り当てる CPU リソース
CONSUMER_GROUP4	4 番目のグループのコンシューマ・グループ名
GROUP4_PERCENT	このグループに割り当てる CPU リソース
CONSUMER_GROUP5	5 番目のグループのコンシューマ・グループ名
GROUP5_PERCENT	このグループに割り当てる CPU リソース
CONSUMER_GROUP6	6 番目のグループのコンシューマ・グループ名
GROUP6_PERCENT	このグループに割り当てる CPU リソース
CONSUMER_GROUP7	7 番目のグループのコンシューマ・グループ名

パラメータ	説明
GROUP7_PERCENT	このグループに割り当てる CPU リソース
CONSUMER_GROUP8	8 番目のグループのコンシューマ・グループ名
GROUP8_PERCENT	このグループに割り当てる CPU リソース

このプロシージャでは、最大8個のコンシューマ・グループを指定できます。サポート対象のリソース割当て方法は、CPUによる方法のみです。プランでは、EMPHASIS CPU割当てポリシー(デフォルト)が使用され、各コンシューマ・グループでは、ROUND_ROBINスケジューリング・ポリシー(これもデフォルト)が使用されます。プラン内の指定された各コンシューマ・グループには、レベル2のCPUパーセンテージが割り当てられます。このプランには、SYS_GROUP(ユーザーSYSおよびSYSTEM用にシステムによって定義された初期コンシューマ・グループ)とOTHER_GROUPSも暗黙的に含まれています。SYS_GROUPコンシューマ・グループには、レベル1でCPUの100%が割り当てられ、OTHER_GROUPSには、レベル3でCPUの100%が割り当てられます。

例: CREATE_SIMPLE_PLANプロシージャを使用した単純なプランの作成

次のスクリプトは、ユーザー指定の2つのコンシューマ・グループを備えた単純なリソース・プランを作成するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN(SIMPLE_PLAN => 'SIMPLE_PLAN1',
   CONSUMER_GROUP1 => 'MYGROUP1', GROUP1_PERCENT => 80,
   CONSUMER_GROUP2 => 'MYGROUP2', GROUP2_PERCENT => 20);
END;
/
```

この文を実行すると、次のプランが作成されます。

コンシューマ・グループ	レベル1	レベル2	レベル3
SYS_GROUP	100%	-	-
MYGROUP1	-	80%	-
MYGROUP2	-	20%	-
OTHER_GROUPS	-	-	100%

関連項目:

- EMPHASIS CPU割当てポリシーの詳細は、「リソース・プランの作成」を参照してください
- ROUND_ROBINスケジューリング・ポリシーの詳細は、「リソース・コンシューマ・グループの作成」を参照してください
- 「リソース・マネージャの要素」

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.5 複雑なリソース・プランの作成

複雑なリソース・プランが必要な場合は、ペンディング・エリアと呼ばれるステージング領域で、プランとそのディレクティブやコンシューマ・グループを作成し、そのプランの妥当性をチェックしてから、データ・ディクショナリに保存する必要があります。

複雑なリソース・プランの作成に必要なステップの概要は、次のとおりです。

ノート:

複雑なリソース・プランとは、DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN プロシージャでは作成できないリソース・プランを指します。

ステップ1: ペンディング・エリアを作成します。

ステップ2: コンシューマ・グループを作成、変更または削除します。

ステップ3: コンシューマ・グループにセッションをマップします。

ステップ4: リソース・プランを作成します。

ステップ5: リソース・プラン・ディレクティブを作成します。

ステップ6: ペンディング・エリアの妥当性をチェックします。

ステップ7: ペンディング・エリアを発行します。

これらのステップは、DBMS_RESOURCE_MANAGER PL/SQLパッケージのプロシージャを使用して完了します。

- ペンディング・エリアについて
  ペンディング・エリアとは、現在実行中のアプリケーションに影響を与えることなく、新規リソース・プランの作成、既存のプランの更新、またはプランの削除を実行できるステージング領域です。
- ペンディング・エリアの作成
  ペンディング・エリアはCREATE_PENDING_AREAプロシージャを使用して作成します。
- リソース・コンシューマ・グループの作成
  リソース・コンシューマ・グループを作成するには、CREATE_CONSUMER_GROUPプロシージャを使用します。
- コンシューマ・グループへのセッションのマッピング
  SET_CONSUMER_GROUP_MAPPINGプロシージャを使用して、コンシューマ・グループにセッションをマップできます。
- リソース・プランの作成
  リソース・プランを作成するには、CREATE_PLANプロシージャを使用します。
- リソース・プラン・ディレクティブの作成
  リソース・プラン・ディレクティブを作成するには、CREATE_PLAN_DIRECTIVEプロシージャを使用します。各ディレクティブはプランまたはサブプランに属しており、リソースをコンシューマ・グループまたはサブプランに割り当てます。
- ペンディング・エリアの妥当性チェック
  ペンディング・エリアで変更を追加しているときは、いつでもVALIDATE_PENDING_AREAをコールして、そのペンディング・エリアの妥当性を確認できます。
- ペンディング・エリアの発行
  変更の妥当性チェックを完了した後は、SUBMIT_PENDING_AREAプロシージャをコールして変更内容をアクティブにします。
- ペンディング・エリアのクリア

CLEAR_PENDING_AREAプロシージャを使用して、ペンディング・エリアをいつでもクリアできます。

関連項目:

- 事前定義のコンシューマ・グループ・マッピング・ルール
- DBMS_RESOURCE_MANAGER PL/SQLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- 「リソース・マネージャの要素」

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.5.1 ペンディング・エリアについて

ペンディング・エリアとは、現在実行中のアプリケーションに影響を与えることなく、新規リソース・プランの作成、既存のプランの更新、またはプランの削除を実行できるステージング領域です。

ペンディング・エリアを作成すると、データベースによってペンディング・エリアが初期化され、既存のプランがペンディング・エリアにコピーされて、既存のプランを更新できるようになります。

ヒント:

ペンディング・エリアを作成した後、DBA_RSRC_PLANS データ・ディクショナリ・ビューを問い合せてすべてのプランをリストすると、各プランについて 2 つのコピー(PENDING ステータスが付いているコピーと付いていないコピー)が表示されます。PENDING ステータスのプランには、ペンディング・エリアの作成後に実行した変更内容が反映されています。保留中の変更は、コンシューマ・グループの場合は DBA_RSRC_CONSUMER_GROUPS を使用して、リソース・プラン・ディレクティブの場合は DBA_RSRC_PLAN_DIRECTIVES を使用して表示できます。詳細は、「リソース・マネージャのデータ・ディクショナリ・ビュー」を参照してください。

変更を追加した後のペンディング・エリアは、発行する前に妥当性をチェックします。発行すると、保留中の変更内容がすべてデータ・ディクショナリに適用され、ペンディング・エリアはクリアされ、解除されます。

最初にペンディング・エリアを作成せずにプランを作成、更新または削除しようとすると(つまり、コンシューマ・グループまたはリソース・プラン・ディレクティブを作成、更新または削除しようとすると)、エラー・メッセージが表示されます。

ペンディング・エリアを発行しても、新しく作成したプランはアクティブ化されません(新しいプランまたは更新後のプランの情報がデータ・ディクショナリに格納されるだけです)。ただし、現在アクティブなプランを変更した場合、そのプランは新しいプラン定義で再アクティブ化されます。リソース・プランをアクティブ化する方法の詳細は、「Oracle Database Resource Managerの有効化とプランの切替え」を参照してください。

ペンディング・エリアを作成した後は、そのペンディング・エリアを発行またはクリアするか、ログアウトするまで、他のユーザーはペンディング・エリアを作成できません。

親トピック: 複雑なリソース・プランの作成

27.5.2 ペンディング・エリアの作成

ペンディング・エリアはCREATE_PENDING_AREAプロシージャを使用して作成します。

例: ペンディング・エリアの作成

次のスクリプトは、ペンディング・エリアを作成して初期化するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
END;
/
```

親トピック: 複雑なリソース・プランの作成

27.5.3 リソース・コンシューマ・グループの作成

リソース・コンシューマ・グループを作成するには、CREATE_CONSUMER_GROUPプロシージャを使用します。

次のパラメータを指定できます。

パラメータ	説明
CONSUMER_GROUP	コンシューマ・グループに割り当てる名前。
COMMENT	任意のコメント。
CPU_MTH	非推奨。MGMT_MTH を使用してください。
MGMT_MTH	コンシューマ・グループのセッション間で CPU を分配するためのリソース割当て方法。デフォルトは'ROUND-ROBIN'で、ラウンドロビン・スケジューラを使用して、セッションの適切な実行を保証します。'RUN-TO-COMPLETION'では、長時間実行セッションを他のセッションより先にスケジュールすることを指定します。この設定によって、長時間実行セッション(バッチ処理など)をより早く完了できるようになります。

例: リソース・コンシューマ・グループの作成

次のスクリプトは、グループ内のセッションにリソースを割り当てるデフォルトの方法(ROUND-ROBIN)を使用して、OLTPというコンシューマ・グループを作成するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
   CONSUMER_GROUP => 'OLTP',
   COMMENT        => 'OLTP applications');
END;
/
```

関連項目:

- 「コンシューマ・グループの更新」
- 「コンシューマ・グループの削除」

親トピック: 複雑なリソース・プランの作成

27.5.4 コンシューマ・グループへのセッションのマッピング

SET_CONSUMER_GROUP_MAPPINGプロシージャを使用して、コンシューマ・グループにセッションをマップできます。

次のパラメータを指定できます。

パラメータ	説明
ATTRIBUTE	パッケージ定数として指定されているセッション属性タイプ。
VALUE	属性の値
CONSUMER_GROUP	コンシューマ・グループの名前。

例: コンシューマ・グループへのセッションのマップ

次のスクリプトは、oeユーザーをOLTPコンシューマ・グループにマップするPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING(
    ATTRIBUTE      => DBMS_RESOURCE_MANAGER.ORACLE_USER,
    VALUE          => 'OE',
    CONSUMER_GROUP => 'OLTP');
END;
/
```

関連項目:

「コンシューマ・グループのマッピング・ルールの作成」

親トピック: 複雑なリソース・プランの作成

27.5.5 リソース・プランの作成

リソース・プランを作成するには、CREATE_PLANプロシージャを使用します。

次の表のパラメータを指定できます。最初の2つのパラメータは必須です。他のパラメータはオプションです。

パラメータ	説明
PLAN	グループに割り当てる名前。
COMMENT	任意の内容を表現するコメント。
CPU_MTH	非推奨。MGMT_MTH を使用してください。
ACTIVE_SESS_POOL_MTH	アクティブ・セッション・プールのリソース割当て方法。デフォルトの ACTIVE_SESS_POOL_ABSOLUTE は、選択できる唯一の方法です。

パラメータ	説明
PARALLEL_DEGREE_LIMIT_MTH	デフォルトは PARALLEL_DEGREE_LIMIT_ABSOLUTE で制限を指定するリソース割当て方法で、これは使用可能な唯一の方法です。
QUEUEING_MTH	キューイングのリソース割当て方法。キュー内の非アクティブ・セッションをキューから削除し、アクティブ・セッション・プールに追加する順序を制御します。デフォルトは FIFO_TIMEOUT で、これは使用可能な唯一の方法です。
MGMT_MTH	各コンシューマ・グループまたは各サブプランが取得する CPU の量を指定するためのリソース割当て方法。デフォルトの方法 'EMPHASIS' は単一レベルまたは複数レベルのプラン用の方法で、コンシューマ・グループ間での CPU の分配方法の指定にパーセンテージを使用します。'RATIO' は単一レベルのプラン用の方法で、CPU の分配方法の指定に割合を使用します。
SUB_PLAN	TRUE の場合は、プランをトップレベルのプランとして使用できません。サブプランとしてのみ使用できます。デフォルトは FALSE です。

例: リソース・プランの作成

次のスクリプトは、DAYTIMEというリソース・プランを作成するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN(
   PLAN    => 'DAYTIME',
   COMMENT => 'More resources for OLTP applications');
END;
/
```

- RATIO CPU割当て方法について
  RATIO方法は、単一レベルのCPU割当てのみを使用する単純なプランを対象にした、代替のCPU割当て方法です。

親トピック: 複雑なリソース・プランの作成

27.5.5.1 RATIO CPU割当て方法について

RATIO方法は、単一レベルのCPU割当てのみを使用する単純なプランを対象にした、代替のCPU割当て方法です。

パーセンテージのかわりに、各コンシューマ・グループに与えるCPUの割合に対応する数を指定します。RATIO方法を使用するには、CREATE_PLANプロシージャのMGMT_MTH引数に'RATIO'を設定します。この方法を使用するプランの例は、「リソース・プラン・ディレクティブの作成」を参照してください。

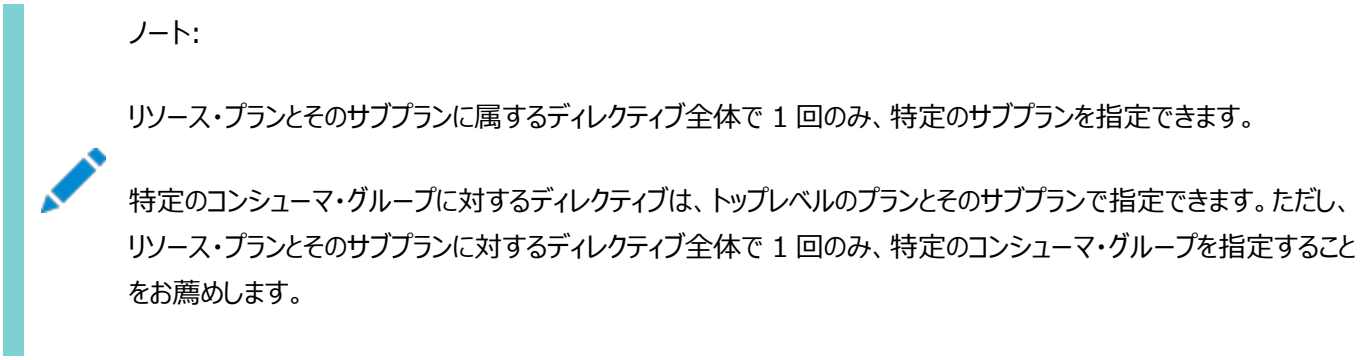
関連項目:

- 「プランの更新」
- 「プランの削除」

親トピック: リソース・プランの作成

27.5.6 リソース・プラン・ディレクティブの作成

リソース・プラン・ディレクティブを作成するには、CREATE_PLAN_DIRECTIVEプロシージャを使用します。各ディレクティブはプランまたはサブプランに属しており、リソースをコンシューマ・グループまたはサブプランに割り当てます。



ノート:

リソース・プランとそのサブプランに属するディレクティブ全体で 1 回のみ、特定のサブプランを指定できます。

特定のコンシューマ・グループに対するディレクティブは、トップレベルのプランとそのサブプランで指定できます。ただし、リソース・プランとそのサブプランに対するディレクティブ全体で 1 回のみ、特定のコンシューマ・グループを指定することをお薦めします。

次のパラメータを指定できます。

パラメータ	説明
PLAN	ディレクティブが属しているリソース・プランの名前。
GROUP_OR_SUBPLAN	リソースを割り当てるコンシューマ・グループまたはサブプランの名前。
COMMENT	任意のコメント。
CPU_P1	非推奨。MGMT_P1 を使用してください。
CPU_P2	非推奨。MGMT_P2 を使用してください。
CPU_P3	非推奨。MGMT_P3 を使用してください。
CPU_P4	非推奨。MGMT_P4 を使用してください。
CPU_P5	非推奨。MGMT_P5 を使用してください。
CPU_P6	非推奨。MGMT_P6 を使用してください。
CPU_P7	非推奨。MGMT_P7 を使用してください。
CPU_P8	非推奨。MGMT_P8 を使用してください。
ACTIVE_SESS_POOL_P1	コンシューマ・グループ内で同時にアクティブにできるセッションの最大数を指定します。他のセッションは、非アクティブ・セッション・キューで実行を待機します。デフォルトは UNLIMITED です。

パラメータ	説明
QUEUEING_P1	非アクティブ・セッション・キュー内で実行を待機しているセッションがタイムアウトしてコールが終了されるまでの時間を秒数で指定します。デフォルトは UNLIMITED です。
PARALLEL_DEGREE_LIMIT_P1	任意の操作の並列度を制限します。デフォルトは UNLIMITED です。
SWITCH_GROUP	切替え基準が満たされた場合に、セッションの切替え先になるコンシューマ・グループを指定します。 グループ名が CANCEL_SQL の場合は、切替え基準が満たされると、現在のコールは取り消されます。グループ名が CANCEL_SQL の場合は、SWITCH_FOR_CALL パラメータは常に TRUE に設定され、ユーザー指定の設定は無効になります。 グループ名が KILL_SESSION の場合は、切替え基準が満たされると、そのセッションは終了します。 グループ名が LOG_ONLY の場合は、セッションに関する情報がリアルタイム SQL 監視で記録されますが、セッションに関する特定の処理は実行されません。 NULL の場合、セッションは切り替えられず、追加のロギングは実行されません。デフォルトは NULL です。このパラメータが NULL に設定され、他の切替えパラメータが NULL 以外に設定された場合は、エラーが返されます。 ノート: CANCEL_SQL、KILL_SESSION および LOG_ONLY は、予約済のコンシューマ・グループ名です。これらのいずれかの名前でコンシューマ・グループを作成しようとすると、エラーが発生します。
SWITCH_TIME	処理が実行されるまでにコールが実行可能な時間を CPU 秒数で指定します。デフォルトは UNLIMITED です。処理は SWITCH_GROUP で指定されます。
SWITCH_ESTIMATE	TRUE の場合は、各コールの実行時間が見積られ、実行時間の見積りが SWITCH_TIME を超える場合は、コールを開始する前にセッションを SWITCH_GROUP に切り替えます。デフォルトは FALSE です。 実行時間の見積りはオプティマイザから取得されます。見積りの正確さは、様々な要因(特にオプティマイザ統計の品質)によって異なります。一般的に、統計には±10 分未満の誤差があると考えてください。
MAX_EST_EXEC_TIME	コールに許可される最大の実行時間を CPU 秒数で指定します。あるコールに対するオプティマイザの見積り実行時間が MAX_EST_EXEC_TIME を超える場合、そのコールは実行されず、ORA-07455 が発行されます。オプティマイザが判断しない場合は、このディレクティブの効果はありません。デフォルトは UNLIMITED です。

パラメータ	説明
	見積りの正確さは、様々な要因(特にオプティマイザ統計の品質)によって異なります。
UNDO_POOL	コンシューマ・グループで生成される、コミットされていないトランザクションの UNDO 合計量の最大値を KB で設定します。デフォルトは UNLIMITED です。
MAX_IDLE_TIME	セッションの最大アイドル時間を秒数で示します。デフォルトは NULL で、無制限を意味します。
MAX_IDLE_BLOCKER_TIME	ブロックしているセッションの最大アイドル時間を秒数で示します。デフォルトは NULL で、無制限を意味します。
SWITCH_TIME_IN_CALL	非推奨。SWITCH_FOR_CALL を使用してください。
MGMT_P1	MGMT_MTH パラメータが EMPHASIS に設定されているプランの場合は、レベル 1 の CPU パーセンテージを指定します。MGMT_MTH パラメータが RATIO に設定されている場合は、CPU 使用の比重を指定します。MGMT_Pn パラメータのデフォルトはすべて NULL です。
MGMT_P2	EMPHASIS の場合は、レベル 2 の CPU パーセンテージを指定します。RATIO には適用しません。
MGMT_P3	EMPHASIS の場合は、レベル 3 の CPU パーセンテージを指定します。RATIO には適用しません。
MGMT_P4	EMPHASIS の場合は、レベル 4 の CPU パーセンテージを指定します。RATIO には適用しません。
MGMT_P5	EMPHASIS の場合は、レベル 5 の CPU パーセンテージを指定します。RATIO には適用しません。
MGMT_P6	EMPHASIS の場合は、レベル 6 の CPU パーセンテージを指定します。RATIO には適用しません。
MGMT_P7	EMPHASIS の場合は、レベル 7 の CPU パーセンテージを指定します。RATIO には適用しません。
MGMT_P8	EMPHASIS の場合は、レベル 8 の CPU パーセンテージを指定します。RATIO には適用しません。

パラメータ	説明
SWITCH_IO_MEGABYTES	処理が実行される前に、セッションで移動(読取りおよび書込み)できる物理 I/O のサイズ(MB)を指定します。デフォルトは UNLIMITED です。処理は SWITCH_GROUP で指定されます。
SWITCH_IO_REQS	処理が実行される前にセッションが実行できる物理 I/O 要求の数を指定します。デフォルトは UNLIMITED です。処理は SWITCH_GROUP で指定されます。
SWITCH_FOR_CALL	TRUE の場合は、SWITCH_TIME、SWITCH_IO_MEGABYTES または SWITCH_IO_REQS に基づいて別のコンシューマ・グループに自動的に切り替えたセッションが、トップレベルのコールが完了した時に当初のコンシューマ・グループに戻されます。デフォルトは NULL です。
PARALLEL_QUEUE_TIMEOUT	パラレル・ステートメントがパラレル・ステートメント・キューで待機できる最大時間を秒単位で指定します。この時間を超えると、タイムアウトします。
PARALLEL_SERVER_LIMIT	特定のコンシューマ・グループが使用できるパラレル実行サーバー・プールの最大割合を指定します。特定のコンシューマ・グループで使用されているパラレル実行サーバーの数は、そのコンシューマ・グループのすべてのセッションで使用されているパラレル実行サーバーの合計となります。
UTILIZATION_LIMIT	コンシューマ・グループに許可する最大 CPU 使用率を指定します。この値は、すべてのレベルの CPU 割当て(MGMT_P1 から MGMT_P8)よりも優先され、未使用の割当てが再配分されるときの合計 CPU 使用率の制限にも適用されます。この属性を指定して、MGMT_P1 から MGMT_P8 を NULL にしておくことができます。
SWITCH_IO_LOGICAL	SWITCH_GROUP によって指定された処理をトリガーする論理 I/O 要求の数。他の切替えディレクティブと同様に、SWITCH_FOR_CALL が TRUE の場合、論理 I/O 要求の数はコールの開始から累積されます。それ以外の場合、論理 I/O 要求の数は、セッションの間中累積されます。
SWITCH_ELAPSED_TIME	処理をトリガーする経過時間(秒単位)は、SWITCH_GROUP で指定されます。他の切替えディレクティブと同様に、SWITCH_FOR_CALL が TRUE の場合、経過時間はコールの開始から累積されます。そうでない場合、経過時間はセッションの間累積される。
SHARES	マルチテナント・コンテナ・データベース(CDB)内のプラガブル・データベース(PDB)間にリソースを割り当てます。また、PDB 内のコンシューマ・グループ間でリソースを割り当てます。

パラメータ	説明
PARALLEL_STMT_CRITICAL	コンシューマ・グループのパラレル・ステートメントが重要かどうかを指定します。 BYPASS_QUEUE が指定されている場合、コンシューマ・グループのパラレル・ステートメントは重要となります。これらの文はパラレル・キューを無視し、即座に処理されます。 FALSE または NULL(デフォルト)が指定されている場合、コンシューマ・グループのパラレル・ステートメントは重要となりません。これらの文は必要なときにパラレル・キューに追加されます。
SESSION_PGA_LIMIT	特定のコンシューマ・グループ内の各セッションに割り当てることのできる最大 PGA メモリーの量を MB 単位で指定します。セッションが制限を超えた場合、プロセスは ORA-10260 エラーで終了します。

例1

次のスクリプトは、プランDAYTIMEのリソース・プラン・ディレクティブを作成するPL/SQLブロックです。(DAYTIMEプランおよびOLTPコンシューマ・グループがペンディング・エリアに作成済であることを前提としています。)

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN             => 'DAYTIME',
   GROUP_OR_SUBPLAN => 'OLTP',
   COMMENT          => 'OLTP group',
   MGMT_P1          => 75);
END;
/
```

このディレクティブは、レベル1でのCPUリソースの75%をOLTPコンシューマ・グループに割り当てます。

REPORTINGコンシューマ・グループを作成し、次のPL/SQLブロックを実行することもできます。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN                     => 'DAYTIME',
   GROUP_OR_SUBPLAN         => 'REPORTING',
   COMMENT                  => 'Reporting group',
   MGMT_P1                  => 15,
   PARALLEL_DEGREE_LIMIT_P1 => 8,
   ACTIVE_SESS_POOL_P1      => 4,
   SESSION_PGA_LIMIT        => 20);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   PLAN                     => 'DAYTIME',
   GROUP_OR_SUBPLAN         => 'OTHER_GROUPS',
   COMMENT                  => 'This one is required',
   MGMT_P1                  => 10);
END;
/
```

このプランでは、コンシューマ・グループREPORTINGに対して操作の最大並列度が8に設定されていますが、他のコンシューマ・グループの並列度に制限はありません。また、このREPORTINGグループには、同時にアクティブにできるセッションの最大数が4に設定されています。各セッションでは、最大20 MBのPGAメモリーを使用できます。

例2

この例では、CPUの割当てにRATIO方式を使用し、使用率ではなく比率を使用します。アプリケーション・スイートで、「Gold」、「Silver」および「Bronze」の3つのサービス・レベルをクライアントに提供しているものとします。3つのコンシューマ・グループをGOLD_CG、SILVER_CGおよびBRONZE_CGという名前で作成し、次のリソース・プランを作成します。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PLAN
   (PLAN              => 'SERVICE_LEVEL_PLAN',
    MGMT_MTH          => 'RATIO',
    COMMENT           => 'Plan that supports three service levels');
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
   (PLAN              => 'SERVICE_LEVEL_PLAN',
    GROUP_OR_SUBPLAN => 'GOLD_CG',
    COMMENT           => 'Gold service level customers',
    MGMT_P1           => 10);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
   (PLAN              => 'SERVICE_LEVEL_PLAN',
    GROUP_OR_SUBPLAN => 'SILVER_CG',
    COMMENT           => 'Silver service level customers',
    MGMT_P1           => 5);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
   (PLAN              => 'SERVICE_LEVEL_PLAN',
    GROUP_OR_SUBPLAN => 'BRONZE_CG',
    COMMENT           => 'Bronze service level customers',
    MGMT_P1           => 2);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
    (PLAN              => 'SERVICE_LEVEL_PLAN',
    GROUP_OR_SUBPLAN => 'OTHER_GROUPS',
    COMMENT           => 'Lowest priority sessions',
    MGMT_P1           => 1);
END;
/
```

CPU割当ての割合は、GOLD_CG、SILVER_CG、BRONZE_CGおよびOTHER_GROUPSコンシューマ・グループに対して、それぞれ10:5:2:1の割合です。

セッションがGOLD_CGおよびSILVER_CGコンシューマ・グループにのみ存在する場合、CPU割当ての割合は、この2つのグループ間で10:5になります。

- リソース・プラン・ディレクティブの競合
  このことは許可されますが、トップレベルのプランおよびそのいずれのサブプランからも同じコンシューマ・グループが参照されないようにすることをお薦めします。

親トピック: 複雑なリソース・プランの作成

27.5.6.1 競合するリソース・プラン・ディレクティブ

このことは許可されますが、トップレベルのプランおよびそのいずれのサブプランからも同じコンシューマ・グループが参照されないようにすることをお薦めします。

トップレベルのプランと複数のサブプランから同じコンシューマ・グループを参照する場合があります。このような場合は、複数のリソース・プラン・ディレクティブによって同じコンシューマ・グループが参照されます。

同様に、複数のリソース・プラン・ディレクティブによって同じコンシューマ・グループが参照されている場合、ディレクティブが競合します。このことは許可されますが、複数のリソース・プラン・ディレクティブによって同じコンシューマ・グループが参照されないようにすることをお薦めします。

関連項目:

- 「リソース・プラン・ディレクティブの更新」
- 「リソース・プラン・ディレクティブの削除」

親トピック: リソース・プラン・ディレクティブの作成

27.5.7 ペンディング・エリアの妥当性チェック

ペンディング・エリアで変更を追加しているときは、いつでもVALIDATE_PENDING_AREAをコールして、そのペンディング・エリアの妥当性を確認できます。

従う必要がある規則は、次のとおりです。これらの規則が妥当性チェック・プロシージャによってチェックされます。

- プランにループを含めることはできません。ループが発生するのは、サブプランに、プラン階層ではそのサブプランの上位になるプランを参照するディレクティブが含まれている場合です。たとえば、サブプランはトップレベルのプランを参照できません。
- プラン・ディレクティブで参照されるプランやリソース・コンシューマ・グループは、すべて存在している必要があります。
- すべてのプランに、プランまたはリソース・コンシューマ・グループを指すプラン・ディレクティブが必要です。
- 特定のレベルの全パーセンテージの合計は、100以下であることが必要です。
- アクティブなインスタンスで現在トップレベルのプランとして使用されているプランは、削除できません。
- 次のパラメータは、リソース・コンシューマ・グループを参照するプラン・ディレクティブでのみ使用できます。他のリソース・プランを参照するプラン・ディレクティブでは使用できません。
    - ACTIVE_SESS_POOL_P1
    - MAX_EST_EXEC_TIME
    - MAX_IDLE_BLOCKER_TIME
    - MAX_IDLE_TIME
    - PARALLEL_DEGREE_LIMIT_P1
    - QUEUEING_P1
    - SESSION_PGA_LIMIT
    - SWITCH_ESTIMATE
    - SWITCH_FOR_CALL
    - SWITCH_GROUP
    - SWITCH_IO_MEGABYTES
    - SWITCH_IO_REQS
    - SWITCH_TIME
    - UNDO_POOL
    - UTILIZATION_LIMIT
- アクティブなプランに含まれるリソース・コンシューマ・グループ数は、28以内にする必要があります。また、プランは最大28の子を持つことができます。
- プランとリソース・コンシューマ・グループには、異なる名前を使用する必要があります。
- アクティブなプラン内のどこかに、OTHER_GROUPSのプラン・ディレクティブが存在する必要があります。これによって、現在のアクティブ・プラン内に含まれるコンシューマ・グループのいずれにも属していないセッションに、OTHER_GROUPSのディレクティブで指定したリソースが割り当てられます。

これらの規則のいずれかに違反すると、VALIDATE_PENDING_AREAからエラー・メッセージが返されます。その場合は、変更を加えて問題を修正し、プロシージャを再度コールできます。

プラン・ディレクティブによって参照されない孤立したコンシューマ・グループを作成できます。これによって、当面は使用しないものの、将来的に実装するプランの一部になる、コンシューマ・グループを作成できます。

例: ペンディング・エリアの妥当性チェック

次のスクリプトは、ペンディング・エリアの妥当性をチェックするPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
END;
/
```

関連項目:

「ペンディング・エリアについて」

親トピック: 複雑なリソース・プランの作成

27.5.8 ペンディング・エリアの発行

変更の妥当性チェックを完了した後は、SUBMIT_PENDING_AREAプロシージャをコールして変更内容をアクティブにします。

SUBMITプロシージャでは妥当性チェックも実行されるため、妥当性チェック・プロシージャを別個にコールする必要はありません。ただし、プランを大幅に変更している場合は、通常、変更の妥当性チェックを段階的に実施する方が問題のデバッグ作業が容易になります。ペンディング・エリア内のすべての変更について妥当性チェックが成功するまで、変更は発行されません(つまり、アクティブになりません)。

SUBMIT_PENDING_AREAプロシージャは、変更の妥当性チェックとコミットに成功すると、ペンディング・エリアをクリア(解除)します。

ノート:

VALIDATE_PENDING_AREA が正常に終了しても、SUBMIT_PENDING_AREA のコールが失敗する場合があります。これが起こるのは、VALIDATE_PENDING_AREA をコールしてから SUBMIT_PENDING_AREA をコールするまでの間に、削除しようとしているプランがインスタンスによってロードされた場合などです。

例: ペンディング・エリアの発行

次のスクリプトは、ペンディング・エリアを発行するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

関連項目:

「ペンディング・エリアについて」

親トピック: 複雑なリソース・プランの作成

27.5.9 ペンディング・エリアのクリア

CLEAR_PENDING_AREAプロシージャを使用して、ペンディング・エリアをいつでもクリアできます。

次のスクリプトは、変更のすべてをペンディング・エリアからクリアし、ペンディング・エリアを解除するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
END;
/
```

CLEAR_PENDING_AREAをコールした後、再び変更を実施するには、その前にCREATE_PENDING_AREAプロシージャをコールする必要があります。

関連項目:

「ペンディング・エリアについて」

親トピック: 複雑なリソース・プランの作成

27.6 Oracle Database Resource Managerの有効化とプランの切替え

Oracle Database Resource Manager(リソース・マネージャ)を有効にするには、RESOURCE_MANAGER_PLAN初期化パラメータを設定します。このパラメータは、トップレベルのプランを指定し、現行のインスタンスで使用するプランを識別します。このパラメータでプランを指定しない場合、リソース・マネージャは有効になりません。

デフォルトでは、次の状況を除き、リソース・マネージャは有効になっていません。

- この項の後半で説明されている、事前定義済メンテナンス・ウィンドウの間。
- INMEMORY_SIZE初期化パラメータを0より大きい値に設定することでOracle Database In-Memoryが有効になっているとき。

テキスト形式の初期化パラメータ・ファイルに次のように記述すると、データベースの起動時にリソース・マネージャがアクティブ化され、トップレベルのプランがmydb_planとして設定されます。

```
RESOURCE_MANAGER_PLAN = mydb_plan
```

DBMS_RESOURCE_MANAGER.SWITCH_PLANパッケージ・プロシージャまたはALTER SYSTEM文を使用して、リソース・マネージャをアクティブ化(または解除)したり、現行のトップレベル・プランを変更することもできます。

次のSQL文は、トップレベルのプランをmydb_planに設定し、リソース・マネージャをアクティブ化します(アクティブ化されていない場合)。

```
ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'mydb_plan';
```

指定したプランがデータ・ディクショナリに存在しない場合は、エラー・メッセージが返されます。

Oracle Schedulerのウィンドウでのリソース・マネージャの自動有効化

リソース・プランを指定するOracle Schedulerのウィンドウがオープン状態の場合、リソース・マネージャは自動的にアクティブ化

されます。スケジューラのウィンドウがクローズ状態の場合、そのウィンドウに関連付けられているリソース・プランは無効で、スケジューラのウィンドウがオープンする前に実行されていたリソース・プランが再度有効になります。(ウィンドウのオープン前に有効なリソース・プランがなかった場合、リソース・マネージャは無効になります。)したがって、ウィンドウのリソース・プランはあらゆるインスタンスで有効になります。

デフォルトでは、MAINTENANCE_WINDOW_GROUPウィンドウ・グループのメンバーである事前定義済のスケジューラ・ウィンドウであり、DEFAULT_MAINTENANCE_PLANリソース・プランを指定するメンテナンス・ウィンドウの間に一連の自動化されたメンテナンス・タスクが実行されることに注意してください。そのため、リソース・マネージャは、メンテナンス・ウィンドウ中にデフォルトでアクティブ化されます。必要な場合は、これらのメンテナンス・ウィンドウを変更して、別のリソース・プランを使用できます。

ノート:

メンテナンス・ウィンドウに関連付けられたプランを変更する場合は、サブプラン ORA$AUTOTASK が新しいプランに含まれていることを確認してください。

関連項目:

- 「ウィンドウ」
- 自動データベース・メンテナンス・タスクの管理

Oracle Schedulerのウィンドウでのプラン切替えの無効化

場合によっては、スケジューラのウィンドウ境界でのリソース・マネージャ・プランの自動変更は、望ましくない可能性があります。たとえば、終了する必要のある重要なタスクがあり、リソース・マネージャ・プランにタスクの優先度を設定してある場合、設定を変更するまでは同じプランのままであることを想定しています。しかし、スケジューラのウィンドウはプランの設定後にアクティブになるため、タスクの実行中にリソース・マネージャ・プランが変更される可能性があります。

この状況を回避するには、次のSQL文に示すように、RESOURCE_MANAGER_PLAN初期化パラメータをシステムに必要なプランの名前に設定し、その名前の前に「FORCE:」を付加します。

```
ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'FORCE:mydb_plan';
```

接頭辞FORCEを使用することで、現行のリソース・プランの変更は、データベース管理者がRESOURCE_MANAGER_PLAN初期化パラメータの値を変更した場合のみ可能であることを示します。この制限を解除するには、プラン名の前に「FORCE:」を付加せずに、そのコマンドを再実行します。

DBMS_RESOURCE_MANAGER.SWITCH_PLANパッケージ・プロシージャには同様の機能があります。

関連項目:

DBMS_RESOURCE_MANAGER.SWITCH_PLANの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

リソース・マネージャの無効化

リソース・マネージャを無効にするには、次のステップを完了します。

1. 次のSQL文を発行します。

```
ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = '';
```

2. すべてのOracle Schedulerのウィンドウからリソース・マネージャの関連を削除します。

   そのためには、resource_plan属性のリソース・プランを参照するすべてのスケジューラのウィンドウについて、DBMS_SCHEDULER.SET_ATTRIBUTEプロシージャを使用して、resource_planを空の文字列('')に設定します。SYSユーザーでログインしていない場合は、SYSスキーマ名を使用してウィンドウ名を修飾します。スケジューラのウィンドウは、DBA_SCHEDULER_WINDOWSデータ・ディクショナリ・ビューで表示できます。詳細は、「ウィンドウの変更」および『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

> ノート:
>
> デフォルトでは、すべてのメンテナンス・ウィンドウが DEFAULT_MAINTENANCE_PLAN リソース・プランを参照します。リソース・マネージャを完全に無効にするには、すべてのメンテナンス・ウィンドウを変更してこのプランを削除する必要があります。ただし、自動化メンテナンス・タスクによるリソース使用が規制されなくなるため、他のセッションのパフォーマンスに悪影響を与える可能性があることに注意してください。メンテナンス・ウィンドウの詳細は、「自動データベース・メンテナンス・タスクの管理」を参照してください。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.7 各種の方法を組み合せたOracle Database Resource Managerの例

リソース・マネージャでリソースを割り当てる方法を例で示します。

- 複数レベルのプランの例
  複数レベルのプランの例を示します。
- 使用率制限の属性を使用した例
  UTILIZATION_LIMITディレクティブ属性を使用すると、アプリケーションのCPU使用率を制限できます。この属性の一般的な使用例として、データベース統合をあげることができます。
- 各種のリソース割当て方法を使用した例
  各種のリソース割当て方法を使用した例を示します。
- ディレクティブ属性を使用したパラレル・ステートメントの管理の例
  ディレクティブ属性を使用したパラレル・ステートメントの管理の例を示します。
- オラクル社が提供する複合ワークロード・プラン
  Oracle Databaseには、事前定義済のリソース・プランMIXED_WORKLOAD_PLAN (バッチ操作よりも対話型の操作を優先するプラン)と、Oracleが推奨する必須のサブプランとコンシューマ・グループが用意されています。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.7.1 複数レベルのプランの例

複数レベルのプランの例を示します。

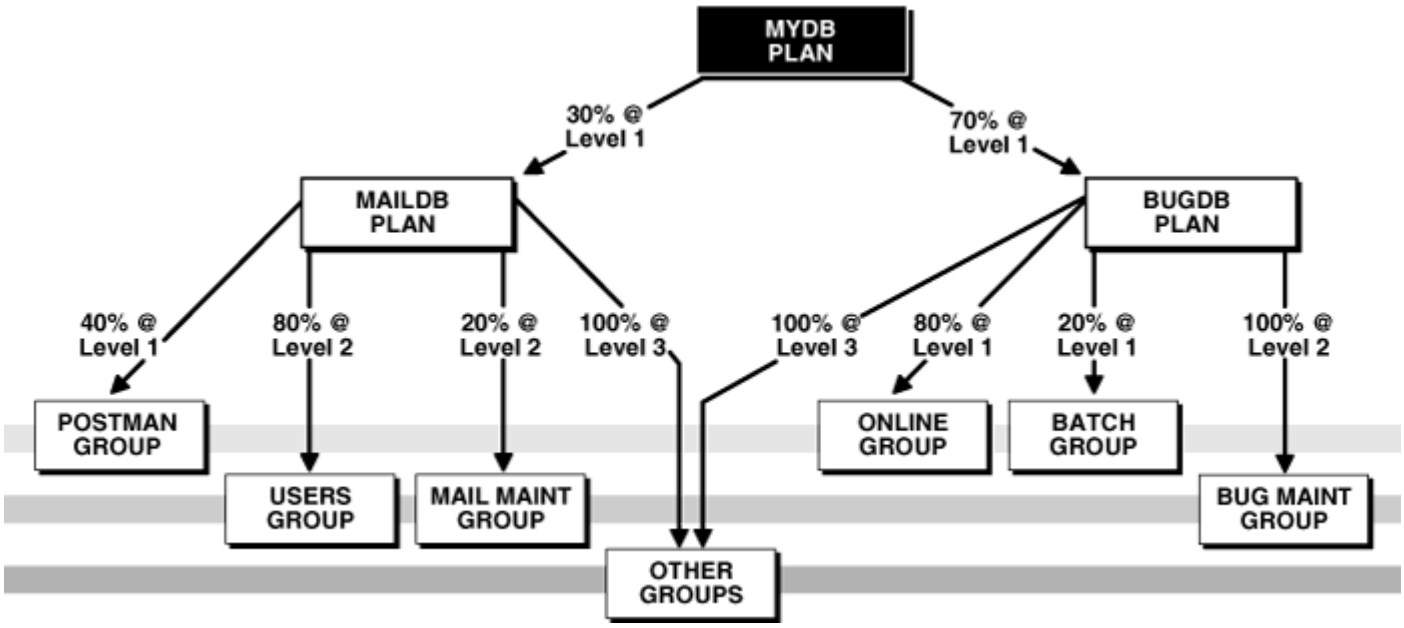
次のスクリプトは、図27-3に示されている複数レベルのプランを作成するPL/SQLブロックです。デフォルトのリソース割当て方法の設定は、すべてのプランとリソース・コンシューマ・グループに使用されます。

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'bugdb_plan',
```

```
   COMMENT => 'Resource plan/method for bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'maildb_plan',
   COMMENT => 'Resource plan/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'mydb_plan',
   COMMENT => 'Resource plan/method for bug and mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Online_group',
   COMMENT => 'Resource consumer group/method for online bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Batch_group',
   COMMENT => 'Resource consumer group/method for batch job bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Maint_group',
   COMMENT => 'Resource consumer group/method for users sessions for bug db maint');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Users_group',
   COMMENT => 'Resource consumer group/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Postman_group',
   COMMENT => 'Resource consumer group/method for mail postman');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_Maint_group',
   COMMENT => 'Resource consumer group/method for users sessions for mail db maint');
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'Online_group',
   COMMENT => 'online bug users sessions at level 1', MGMT_P1 => 80, MGMT_P2=> 0);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'Batch_group',
   COMMENT => 'batch bug users sessions at level 1', MGMT_P1 => 20, MGMT_P2 => 0,
   PARALLEL_DEGREE_LIMIT_P1 => 8);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'Bug_Maint_group',
   COMMENT => 'bug maintenance users sessions at level 2', MGMT_P1 => 0, MGMT_P2 =>
100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'OTHER_GROUPS',
   COMMENT => 'all other users sessions at level 3', MGMT_P1 => 0, MGMT_P2 => 0,
   MGMT_P3 => 100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'Postman_group',
   COMMENT => 'mail postman at level 1', MGMT_P1 => 40, MGMT_P2 => 0);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'Users_group',
   COMMENT => 'mail users sessions at level 2', MGMT_P1 => 0, MGMT_P2 => 80);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'Mail_Maint_group',
   COMMENT => 'mail maintenance users sessions at level 2', MGMT_P1 => 0, MGMT_P2 =>
20);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'OTHER_GROUPS',
   COMMENT => 'all other users sessions at level 3', MGMT_P1 => 0, MGMT_P2 => 0,
   MGMT_P3 => 100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan',
   GROUP_OR_SUBPLAN => 'maildb_plan',
   COMMENT=> 'all mail users sessions at level 1', MGMT_P1 => 30);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan',
   GROUP_OR_SUBPLAN => 'bugdb_plan',
   COMMENT => 'all bug users sessions at level 1', MGMT_P1 => 70);
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

妥当性チェックはSUBMIT_PENDING_AREAによって暗黙的に実行されるので、直前のVALIDATE_PENDING_AREAのコールはオプションです。

図27-3 複数レベルのプラン・スキーマ



このプラン・スキーマでは、次のようにCPUリソースが割り当てられています。

- mydb_planでは、CPUの30%がmaildb_planサブプランに割り当てられ、70%がbugdb_planサブプランに割り当てられます。サブプランは両方ともレベル1です。mydb_plan自体にはレベル1の下にレベルがないため、レベル1で一方のサブプランが使用していないリソース割当ては、その兄弟サブプランが使用できます。したがって、maildb_planがCPUの20%のみを使用する場合、CPUの80%はbugdb_planが使用できます。
- maildb_planおよびbugdb_planでは、レベル1、2および3で割当てを定義します。これらのサブプランの各レベルは、その親プランmydb_planのレベルから独立しています。つまり、プラン・スキーマのすべてのプランとサブプランには、独自のレベル1、レベル2、レベル3などがあります。
- maildb_planに割り当てられたCPUの30%について、その40%(実質的にはCPU合計の12%)はレベル1のPostman_groupに割り当てられます。Postman_groupにはレベル1で兄弟がないため、レベル1では暗黙的に60%が残っています。この60%は、レベル2のUsers_groupとMail_Maint_groupがそれぞれ80%と20%の割合で共有します。この60%の他に、Users_groupとMail_Maint_groupは、レベル1のPostman_groupが使用していないリソース(40%の一部)も使用できます。
- 複数レベルのプランの場合、未使用のリソースは同じレベルの兄弟ではなく、次の下位レベルのコンシューマ・グループまたはサブプランに再割当てされるため、レベル2のUsers_groupとMail_Maint_groupのどちらでも使用されないCPUリソースはOTHER_GROUPSに割り当てられます。したがって、Users_groupが80%ではなく70%しか使用していない場合に、残りの10%をMail_Maint_groupが使用することはできません。この10%は、レベル3のOTHER_GROUPSのみが使用できます。
- bugdb_planサブプランに割り当てられたCPUの70%は、そのコンシューマ・グループに同様に割り当てられます。Online_groupまたはBatch_groupがその割当てのすべてを使用していない場合、残りはBug_Maint_groupが使用できます。Bug_Maint_groupがその割当てのすべてを使用していない場合、残りはOTHER_GROUPSに割り当てられます。

親トピック: 各種の方法を組み合せたOracle Database Resource Managerの例

27.7.2 使用率制限の属性を使用した例

UTILIZATION_LIMITディレクティブ属性を使用すると、アプリケーションのCPU使用率を制限できます。この属性の一般的な使用例として、データベース統合をあげることができます。

データベースの統合では、次の処理が必要になる場合があります。

- あるアプリケーションが別のアプリケーションのパフォーマンスに及ぼす影響を管理します。

  このパフォーマンスの影響を管理する方法として、アプリケーションごとにコンシューマ・グループを作成し、各コンシューマ・グループにリソースを割り当てる方法をあげることができます。

- 各アプリケーションの使用率を制限します。

  一般的に、CPUリソースの特定の使用率を各コンシューマ・グループに割り当てること以外に、グループごとに最大CPU使用率を制限することが必要になる場合があります。この制限によって、他のすべてのコンシューマ・グループがアイドル状態のときに、あるコンシューマ・グループがすべてのCPUリソースを使用することを回避できます。

  場合によっては、他のアプリケーションからのワークロードに関係なく、すべてのアプリケーション・ユーザーに一貫性のあるパフォーマンスを提供することが必要になる場合があります。そのためには、リソース・プランのコンシューマ・グループごとに使用率制限を指定します。

次の例は、UTILIZATION_LIMITリソース・プラン・ディレクティブ属性を使用して次のことを行う方法を示しています。

- データベースCPU使用率の合計の制限
- リソース集中型の問合せの隔離
- アプリケーションのCPU使用率の制限
- メンテナンス・ウィンドウの期間中のCPU使用率の制限

例1 - 全体的なデータベースCPU使用率の制限

この例では、データベース負荷にかかわらず、Oracle DatabaseからのシステムワークロードはCPUの90%を超えません。サーバーを共有する他のアプリケーション用にCPUの10%を残します。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.CREATE_PLAN(
    PLAN    => 'MAXCAP_PLAN',
    COMMENT => 'Limit overall database CPU');

  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
    PLAN              => 'MAXCAP_PLAN',
    GROUP_OR_SUBPLAN  => 'OTHER_GROUPS',
    COMMENT           => 'This group is mandatory',
    UTILIZATION_LIMIT => 90);
  DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

OTHER_GROUPSのプラン・ディレクティブ以外のプラン・ディレクティブはないため、すべてのセッションはOTHER_GROUPSにマップされます。

例2 - リソース集中型の問合せの検査

この例では、リソース集中型の問合せは使用率制限が20%のコンシューマ・グループに切り替えられ、ユーザーが介入するまでに使用されるリソース量は制限されます。リソース集中型の問合せとは、ここでは10分を超えるCPU時間を必要とする問合せとします。セッション・マッピング・ルールによってすべてのセッションはSTART_GROUPで開始されると想定します。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
     CONSUMER_GROUP => 'START_GROUP',
```

```
    COMMENT        => 'Sessions start here');

  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
    CONSUMER_GROUP => 'QUARANTINE_GROUP',
    COMMENT        => 'Sessions switched here to quarantine them');

  DBMS_RESOURCE_MANAGER.CREATE_PLAN(
    PLAN    => 'Quarantine_plan',
    COMMENT => 'Quarantine runaway queries');

  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
    PLAN                  => 'Quarantine_plan',
    GROUP_OR_SUBPLAN      => 'START_GROUP',
    COMMENT               => 'Max CPU 10 minutes before switch',
    MGMT_P1               => 75,
    switch_group          => 'QUARANTINE_GROUP',
    switch_time           => 600);

  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
    PLAN                  => 'Quarantine_plan',
    GROUP_OR_SUBPLAN      => 'OTHER_GROUPS',
    COMMENT               => 'Mandatory',
    MGMT_P1               => 25);

  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
    PLAN                  => 'Quarantine_plan',
    GROUP_OR_SUBPLAN      => 'QUARANTINE_GROUP',
    COMMENT               => 'Limited CPU',
    MGMT_P2               => 100,
    UTILIZATION_LIMIT     => 20);
  DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

ノート:

QUARANTINE_GROUP の使用率制限を 0(ゼロ)に設定して、リソース集中型の問合せを完全に検査できますが、このようにしないことをお薦めします。他のセッションが必要とするリソース(PGA メモリー、ロックなど)をリソース集中型の問合せが保持している場合、0(ゼロ)を割り当てる設定によりデッドロックが発生する場合があります。

例3 - アプリケーションのCPUの制限

この例では、アプリケーション・セッションはマッピング・ルールによって4つのアプリケーション・グループのいずれかにマップされると想定します。各アプリケーション・グループには30%の使用率制限が割り当てられています。これにより、1つのアプリケーションのCPU使用率は30%に制限されます。UTILIZATION_LIMIT値の合計が100%を超えていますが、このことはすべてのアプリケーションが同時にアクティブにはならない状況において許容されます。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();

  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(
    CONSUMER_GROUP => 'APP1_GROUP',
    COMMENT        => 'Apps group 1');
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
    CONSUMER_GROUP => 'APP2_GROUP',
    COMMENT        => 'Apps group 2');
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
    CONSUMER_GROUP => 'APP3_GROUP',
    COMMENT        => 'Apps group 3');
```

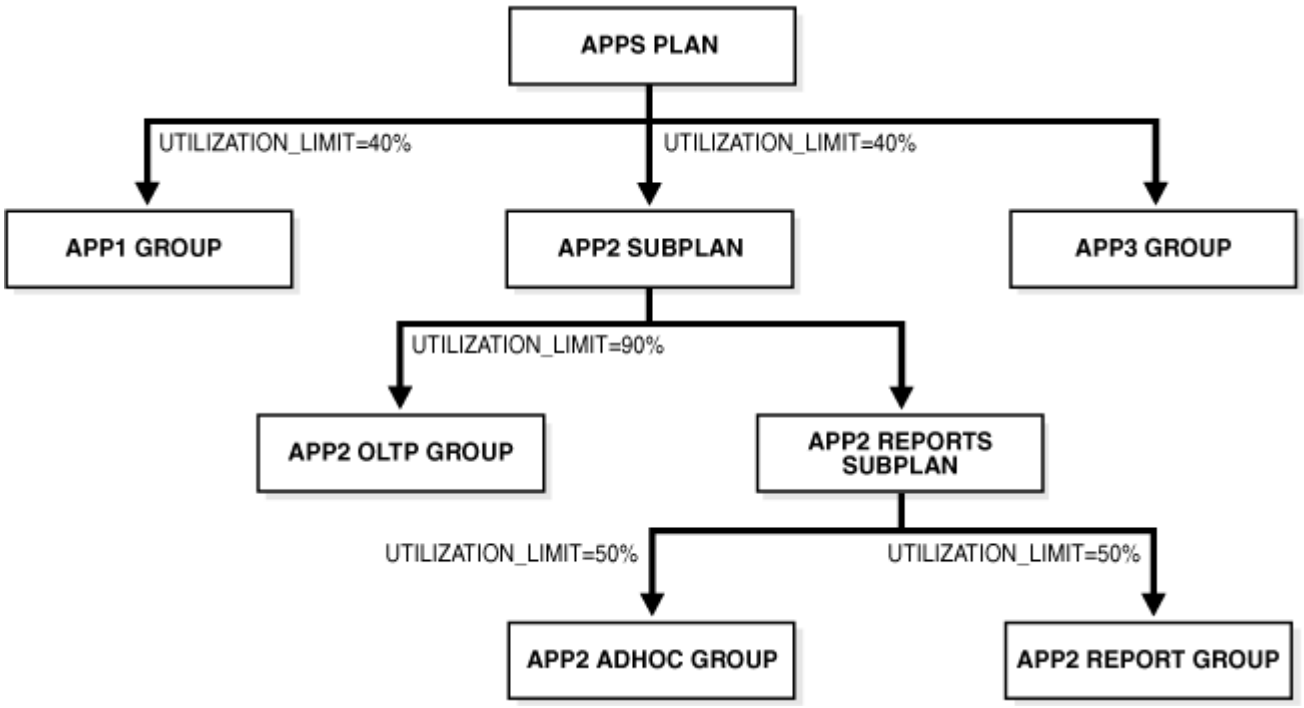
```
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
     CONSUMER_GROUP => 'APP4_GROUP',
     COMMENT        => 'Apps group 4');

  DBMS_RESOURCE_MANAGER.CREATE_PLAN(
    PLAN    => 'apps_plan',
    COMMENT => 'Application consolidation');

  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'apps_plan',
    GROUP_OR_SUBPLAN      => 'APP1_GROUP',
    COMMENT               => 'Apps group 1',
    UTILIZATION_LIMIT     => 30);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'apps_plan',
    GROUP_OR_SUBPLAN      => 'APP2_GROUP',
    COMMENT               => 'Apps group 2',
    UTILIZATION_LIMIT     => 30);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'apps_plan',
    GROUP_OR_SUBPLAN      => 'APP3_GROUP',
    COMMENT               => 'Apps group 3',
    UTILIZATION_LIMIT     => 30);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'apps_plan',
    GROUP_OR_SUBPLAN      => 'APP4_GROUP',
    COMMENT               => 'Apps group 4',
    UTILIZATION_LIMIT     => 30);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'apps_plan',
    GROUP_OR_SUBPLAN      => 'OTHER_GROUPS',
    COMMENT               => 'Mandatory',
    UTILIZATION_LIMIT     => 20);

  DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

4つすべてのアプリケーション・グループが割り当てられたCPU(この例では30%)を完全に使用できる場合は、各アプリケーション・グループに割り当てられる最小CPUは、すべてのアプリケーション・グループの制限全体に占めるアプリケーション・グループの制限の比率として計算されます。この例では、4つすべてのアプリケーション・グループに使用率制限として30%が割り当てられています。このため、4つすべてのグループがそれぞれの制限を完全に使用する場合、各グループに対するCPU割当ては30/(30+30+30+30)で25%になります。

例4 - コンシューマ・グループおよびサブプランに使用率制限を指定する

次の例では、図27-4に示すように、サブプランおよびそのサブプラン内のコンシューマ・グループに対して`UTILIZATION_LIMIT`を設定した場合に、使用率制限がどのように計算されるかについて説明します。わかりやすくするために、`OTHER_GROUPS`コンシューマ・グループを指定する要件は無視し、リソース・プラン・ディレクティブは(プランの一部であっても)省略されています。

図27-4 サブプランおよびコンシューマ・グループに最大使用率が設定されたリソース・プラン



次のPL/SQLブロックは、図27-4で説明するプランを作成します。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
    DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
     CONSUMER_GROUP => 'APP1_GROUP',
     COMMENT        => 'Group for application #1');
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
     CONSUMER_GROUP => 'APP2_OLTP_GROUP',
     COMMENT        => 'Group for OLTP activity in application #2');
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
     CONSUMER_GROUP => 'APP2_ADHOC_GROUP',
     COMMENT        => 'Group for ad-hoc queries in application #2');
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
     CONSUMER_GROUP => 'APP2_REPORT_GROUP',
     COMMENT        => 'Group for reports in application #2');
   DBMS_RESOURCE_MANAGER.CREATE_PLAN(
    PLAN    => 'APPS_PLAN',
    COMMENT => 'Plan for managing 3 applications');
  DBMS_RESOURCE_MANAGER.CREATE_PLAN(
    PLAN    => 'APP2_SUBPLAN',
    COMMENT => 'Subplan for managing application #2',
    SUB_PLAN => TRUE);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN(
    PLAN    => 'APP2_REPORTS_SUBPLAN',
    COMMENT => 'Subplan for managing reports in application #2',
    SUB_PLAN => TRUE);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'APPS_PLAN',
    GROUP_OR_SUBPLAN      => 'APP1_GROUP',
    COMMENT               => 'Limit CPU for application #1 to 40%',
    UTILIZATION_LIMIT     => 40);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'APPS_PLAN',
    GROUP_OR_SUBPLAN      => 'APP2_SUBPLAN',
    COMMENT               => 'Limit CPU for application #2 to 40%',
    UTILIZATION_LIMIT     => 40);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                  => 'APP2_SUBPLAN',
    GROUP_OR_SUBPLAN      => 'APP2_OLTP_GROUP',
    COMMENT               => 'Limit CPU for OLTP to 90% of application #2',
    UTILIZATION_LIMIT     => 90);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
```

```
    PLAN                   => 'APP2_SUBPLAN',
    GROUP_OR_SUBPLAN       => 'APP2_REPORTS_SUBPLAN',
    COMMENT                => 'Subplan for ad-hoc and normal reports for application
#2');
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                   => 'APP2_REPORTS_SUBPLAN',
    GROUP_OR_SUBPLAN       => 'APP2_ADHOC_GROUP',
    COMMENT                => 'Limit CPU for ad-hoc queries to 50% of application #2
reports',
    UTILIZATION_LIMIT      => 50);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                   => 'APP2_REPORTS_SUBPLAN',
    GROUP_OR_SUBPLAN       => 'APP2_REPORT_GROUP',
    COMMENT                => 'Limit CPU for reports to 50% of application #2 reports',
    UTILIZATION_LIMIT      => 50);
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    PLAN                   => 'APPS_PLAN',
    GROUP_OR_SUBPLAN       => 'OTHER_GROUPS',
    COMMENT                => 'No directives for default users');
   DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

この例では、コンシューマ・グループAPP1_GROUPおよびサブプランAPP2_SUBPLANの最大CPU使用率は、40%に設定されます。コンシューマ・グループAPP2_ADHOC_GROUPおよびAPP2_REPORT_GROUPの制限は、50%に設定されます。

サブプランAPP2_REPORTS_SUBPLANに指定されている制限がないため、その親サブプランAPP2_SUBPLANの制限を継承して40%になります。コンシューマ・グループAPP2_REPORT_GROUPの絶対上限は、その親サブプランの50%として計算され、その結果、40%の50%、つまり20%になります。

同じく、コンシューマ・グループAPP2_ADHOC_GROUPはサブプランAPP2_REPORTS_SUBPLANに含まれているため、その制限は親サブプランの割合として計算されます。コンシューマ・グループAPP2_ADHOC_GROUPの使用率制限は40%の50%、つまり20%になります。

コンシューマ・グループAPP2_OLTP_GROUPの最大CPU使用率は、90%に設定されます。APP2_OLTP_GROUPの親サブプランAPP2_SUBPLANの制限は40%です。このため、グループAPP2_OLTP_GROUPの絶対上限は40%の90%、つまり36%になります。

親トピック: 各種の方法を組み合せたOracle Database Resource Managerの例

27.7.3 各種のリソース割当て方法を使用した例

各種のリソース割当て方法を使用した例を示します。

ここで示す例は、パッケージ化されたEnterprise Resource Planning(ERP)またはCustomer Relationship Management(CRM)アプリケーションをサポートするデータベース用のプランを表します。このような環境で必要な処理は多種多様です。大規模なパラレル問合せを含む長時間実行のバッチ・ジョブとともに、短いトランザクションや短い問合せが混在している場合があります。その目的は、バッチ・ジョブをパラレルに実行しながら、オンライン・トランザクション処理(OLTP)の適切な応答時間を得ることにあります。

次の表にプランがまとめられています。

グループ	CPUリソース割当て%	パラレル文のキューイング	コンシューマ・グループの自動切替え	最大見積り実行時間	UNDOプール	各セッションのPGA制限

グループ	CPUリソース割当て%	パラレル文のキューイング	コンシューマ・グループの自動切替え	最大見積り実行時間	UNDOプール	各セッションのPGA制限
oltp	60%	--	切替え先グループ: batch 切替え時間: 3秒	--	200K	20M
batch	30%	パラレル・サーバー制限: 8 パラレル・キューのタイムアウト: 600秒	--	3600 秒	--	--
OTHER_GROUPS	10%	--	--	--	--	--

次の文は、この表のプランをerp_planという名前で作成します。

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'erp_plan',
  COMMENT => 'Resource plan/method for ERP Database');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'oltp',
  COMMENT => 'Resource consumer group/method for OLTP jobs');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'batch',
  COMMENT => 'Resource consumer group/method for BATCH jobs');
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'erp_plan',
  GROUP_OR_SUBPLAN => 'oltp', COMMENT => 'OLTP sessions', MGMT_P1 => 60,
  SWITCH_GROUP => 'batch', SWITCH_TIME => 3, UNDO_POOL => 200,
  SWITCH_FOR_CALL => TRUE, SESSION_PGA_LIMIT => 20);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'erp_plan',
  GROUP_OR_SUBPLAN => 'batch', COMMENT => 'BATCH sessions', MGMT_P1 => 30,
  PARALLEL_SERVER_LIMIT => 8, PARALLEL_QUEUE_TIMEOUT => 600,
  MAX_EST_EXEC_TIME => 3600);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'erp_plan',
  GROUP_OR_SUBPLAN => 'OTHER_GROUPS', COMMENT => 'mandatory', MGMT_P1 => 10);
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

親トピック: 各種の方法を組み合せたOracle Database Resource Managerの例

27.7.4 ディレクティブ属性を使用したパラレル・ステートメントの管理の例

ディレクティブ属性を使用したパラレル・ステートメントの管理の例を示します。

一般的に、データ・ウェアハウス環境は、リソース要件が様々に異なる多様なユーザーで構成されています。処理要件が共通するユーザーは、コンシューマ・グループにグループ化されます。コンシューマ・グループURGENT_GROUPは、経営陣に重要な情報を提供するためのレポートを実行するユーザーで構成されています。このグループは、多数のパラレル問合せを生成します。コン

シューマ・グループETL_GROUPに属するユーザーは、ソース・システムからデータをインポートし、抽出、変換、ロード(ETL)の各操作を実行します。グループOTHER_GROUPSには、非定型問合せを実行するユーザーが含まれています。パフォーマンスを最適に保つと同時に、このような多様なユーザー・グループの要件を管理する必要があります。

次のディレクティブ属性を使用すると、パラレル・ステートメントの実行を管理および最適化できます。

- MGMT_Pn
- PARALLEL_SERVER_LIMIT
- PARALLEL_STMT_CRITICAL
- PARALLEL_QUEUE_TIMEOUT
- PQ_TIMEOUT_ACTION
- PARALLEL_DEGREE_LIMIT_P1
- SHARES

ノート:

- MGMT_Pn 管理属性と SHARES 属性では、パラレル文キューから実行するパラレル文を選択する方法を制御します。あるコンシューマ・グループのパラレル・ステートメントを別のグループよりも優先させるには、そのグループの管理属性の値を大きくします。
- マルチテナント環境では、詳細なワークロードごとの管理が必要な場合、SHARES 属性を使用して、パラレル文のキューイング・リソースを含むプラガブル・データベース(PDB)のリソース割当ての共有を指定できます。また、前述した別のディレクティブ属性を使用することもできます。

表27-4に、データ・ウェアハウス・ユーザーのニーズを管理するために使用可能なプランDW_PLANのリソース割当てを示します。このプランには、コンシューマ・グループURGENT_GROUP、ETL_GROUPおよびOTHER_GROUPSが含まれています。この例は、1つのアプリケーションまたはコンシューマ・グループですべての使用可能なパラレル実行サーバーを使用しないことが確実である場合のディレクティブ属性の使用例を示しています。

表27-4 パラレル・ステートメント・ディレクティブが設定されたリソース・プラン

コンシューマ・グループ	レベル1のCPU割当て	レベル2のCPU割当て	レベル3のCPU割当て	PARALLEL_DEGREE_LIMIT_P1	PARALLEL_SERVER_LIMIT	PARALLEL_QUEUE_TIMEOUT
URGENT_GROUP	100%			12		
ETL_GROUP		100%		8	50%	
OTHER_GROUPS			100%	2	50%	360

この例では、パラメータPARALLEL_SERVERS_TARGET初期化パラメータが64に設定されており、使用可能なパラレル実行サーバーの数が64台ということになります。パラレル・ステートメントの実行に使用できるパラレル実行サーバーの総数は64で、それを超えると、PARALLEL_DEGREE_POLICYがAUTOに設定されたURGENT_GROUPセッションはパラレル・ステートメント・キューに追加されます。ETL_GROUPおよびOTHER_GROUPSのPARALLEL_SERVER_LIMIT属性は50%で、これらのグ

ループが使用できるパラレル実行サーバーの最大数は64の50%、つまり各グループとも32台のパラレル実行サーバーを使用できます。

コンシューマ・グループからのパラレル・ステートメントがキューされるのは、PARALLEL_DEGREE_POLICYパラメータがAUTOに設定され、コンシューマ・グループのアクティブなサーバーの総数がPARALLEL_SERVERS_TARGETよりも高くなっている場合のみです。PARALLEL_DEGREE_POLICYがMANUALまたはLIMITEDに設定されている場合、パラレル・ステートメントは、十分な数のパラレル実行サーバーが使用できるときにのみ、実行されます。このようなパラレル・ステートメントで使用されるパラレル実行サーバーは、コンシューマ・グループで使用されるパラレル実行サーバーの総数にカウントされます。ただし、パラレル・ステートメントはパラレル・ステートメント・キューに追加されません。

ヒント:

優先度が低いアプリケーションの場合、PARALLEL_DEGREE_LIMIT_P1 および PARALLEL_SERVER_LIMIT に小さい値を設定するのが一般的です。

URGENT_GROUPにはレベル1で100%が割り当てられているため、そのパラレル・ステートメントはパラレル・ステートメント・キューの他のコンシューマ・グループよりも前に常にデキューされます。URGENT_GROUPにはPARALLEL_SERVER_LIMITディレクティブ属性がありませんが、このグループのセッションによって発行された文は、その文を実行できるだけのパラレル実行サーバーが使用可能でない場合にはキューされます。

URGENT_GROUPのリソース・プラン・ディレクティブを作成するときは、PARALLEL_STMT_CRITICALパラメータをBYPASS_QUEUEに設定できます。この設定では、コンシューマ・グループからのパラレル・ステートメントは、パラレル・ステートメント・キューを無視して即座に処理されます。ただし、パラレル実行サーバーの数がPARALLEL_SERVERS_TARGET初期化パラメータの設定を超えることがあり、PARALLEL_MAX_SERVERS初期化パラメータで設定された制限に達した場合、並列度が低くなることがあります。

PARALLEL_DEGREE_LIMIT_P1で表される並列度は、URGENT_GROUPの場合12に設定されます。したがって、URGENT_GROUPの各パラレル・ステートメントは最大で12台のパラレル実行サーバーを使用できます。同様に、ETL_GROUPの各パラレル・ステートメントは最大で8台のパラレル実行サーバーを使用でき、OTHER_GROUPSの各パラレル・ステートメントは2台のパラレル実行サーバーを使用できます。

たとえば、ある時間にパラレル・ステートメントがETL_GROUPからのみ実行され、このパラレル・ステートメントによって、このグループで使用可能な32台のパラレル実行サーバーのうち26台が使用されているとします。このコンシューマ・グループからのセッションのPARALLEL_DEGREE_POLICYは、AUTOに設定されています。別のパラレル・ステートメントがPARALLEL_DEGREE_LIMIT_P1属性が8の設定でETL_GROUPから起動されても、ETL_GROUPで使用可能なパラレル実行サーバーは、32-26=6パラレル実行サーバーであるので、この問合せはすぐには実行されません。新しいパラレル・ステートメントは、必要な数のパラレル実行サーバーがETL_GROUPで使用可能になるまでキューで待機します。

ETL_GROUPのパラレル・ステートメントの実行中、OTHER_GROUPSからパラレル・ステートメントが開始されたとします。このグループでは32台のパラレル実行サーバーを使用できるので、パラレル・ステートメントは実行されます。

OTHER_GROUPSのPARALLEL_QUEUE_TIMEOUT属性は、360に設定されています。このため、このグループからのパラレル・ステートメントは、360秒間のみパラレル実行サーバーのキューに残ることができます。この時間が経過すると、パラレル・ステートメントはキューから削除され、エラーORA-07454が返されます。

関連項目:

- 「パラレル実行サーバー」
- 「リソース・プラン・ディレクティブの作成」

親トピック: 各種の方法を組み合せたOracle Database Resource Managerの例

27.7.5 オラクル社が提供する複合ワークロード・プラン

Oracle Databaseには、事前定義済のリソース・プランMIXED_WORKLOAD_PLAN (バッチ操作よりも対話型の操作を優先するプラン)と、Oracleが推奨する必須のサブプランとコンシューマ・グループが用意されています。

MIXED_WORKLOAD_PLANは、次のように定義されています。

	CPUリソース割当て				
グループまたはサブプラン	**レベル1**	**レベル2**	**レベル3**	**コンシューマ・グループの自動切替え**	**最大並列度**
BATCH_GROUP			100%		
INTERACTIVE_GROUP		85%		切替え先グループ: BATCH_GROUP 切替え時間: 60 秒 コールに対する切替え: TRUE	1
ORA$AUTOTASK		5%			
OTHER_GROUPS		5%			
SYS_GROUP	100%				

このプランのINTERACTIVE_GROUPは短いトランザクションを対象にしているため、60秒を超えるCPU時間を使用するコールは、長いバッチ操作を対象にしたBATCH_GROUPに自動的に切り替わります。

使用環境に適する場合は、この事前定義済のプランを使用できます。(このプランは変更したり、使用しない場合に削除できます。)BATCH_GROUPおよびINTERACTIVE_GROUPの名前に特別な意味はありません。この名前はグループの用途を反映しているだけで、ユーザーは任意に、これらのグループにアプリケーション・セッションをマップし、それに応じてCPUリソースの割当て使用率を調整して、使用する対話型アプリケーションおよびバッチ・アプリケーションにあわせた適切なリソース管理を実現できます。たとえば、対話型アプリケーションをINTERACTIVE_GROUPコンシューマ・グループの下で実行するには、ユーザー名、サービス名、プログラム名、モジュール名または処理に基づいて、このコンシューマ・グループに対話型アプリケーションのユーザー・セッションをマップする必要があります(「コンシューマ・グループへのセッションのマッピング・ルールの指定」を参照)。同じように、BATCH_GROUPにバッチ・アプリケーションをマップする必要があります。最後に、このプランを有効にする必要があります(「Oracle Database Resource Managerの有効化とプランの切替え」を参照)。

このプランの他のリソース・コンシューマ・グループとサブプランについては、表27-5および表27-6を参照してください。

親トピック: 各種の方法を組み合せたOracle Database Resource Managerの例

27.8 単一サーバーにおける複数のデータベース・インスタンスの管理

Oracle Databaseには、複数のデータベース・インスタンスを実行する複数CPUサーバーでCPU割当てを管理する方法が用意されています。この方法はインスタンス・ケージングと呼ばれます。インスタンス・ケージングとOracle Database Resource Manager(リソース・マネージャ)が連携して、複数インスタンス間で必要なサービス・レベルをサポートします。

- インスタンス・ケージングについて
  各データベース・インスタンスのCPU使用率を制限する簡単な方法は、インスタンス・ケージングを使用することです。インスタンス・ケージングは、初期化パラメータを使用して、インスタンスが同時に使用できるCPU数を制限する方法です。
- インスタンス・ケージングの有効化
  CPUディレクティブでリソース・プランを作成し、CPU_COUNT初期化パラメータを設定することで、インスタンス・ケージングの使用を有効にできます。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.8.1 インスタンス・ケージングについて

各データベース・インスタンスのCPU使用率を制限する簡単な方法は、インスタンス・ケージングを使用することです。インスタンス・ケージングは、初期化パラメータを使用して、インスタンスが同時に使用できるCPU数を制限する方法です。

1台のマルチCPU搭載サーバーで複数のOracle Databaseインスタンスを実行するように決定することがあります。その代表的な理由は、サーバーの統合、つまり使用できるハードウェア・リソースをより効率的に使用するためです。1台のサーバーで複数のインスタンスが実行されている場合、インスタンスはCPUリソースを競い合います。1つのリソース集中型のデータベース・インスタンスが、他のインスタンスのパフォーマンスを大きく低下させる場合があります。たとえば、16個のCPUのシステムで4つのデータベース・インスタンスが存在する場合に、ある1つのデータベース・インスタンスに大きな負荷がかかっている間、オペレーティング・システムによって、このインスタンスの実行にCPUの大半が使用される可能性があります。これにより、他の3つのインスタンスのパフォーマンスが低下することがあります。このようなCPU割当ては、オペレーティング・システムのみによって決定され、通常はユーザーが制御することはできません。

前の例で、インスタンス・ケージングを使用して4つのインスタンスそれぞれのCPU数を4に制限すると、1つのインスタンスによって他のインスタンスが妨害される可能性が低くなります。4つのCPUに制約されると、インスタンスはCPUにバインドされます。このとき、リソース・マネージャによって、インスタンスに対して設定したリソース・プランに基づき、様々なデータベース・セッション間でのCPUの割当てが開始されます。このように、インスタンス・ケージングとリソース・マネージャによって、単一サーバーで複数のインスタンスを管理する簡単で効果的な方法が提供されます。

サーバーのインスタンス・ケージングには、次の2つの一般的なアプローチがあります。

- オーバーサブスクライブ—このアプローチは、開発システムやテスト・システムなどの重要度の低いデータベースや、負荷が少なく、かつ重要度の低い本番システムに対して使用します。このアプローチでは、各インスタンスのCPU制限の合計がシステム上の実際のCPU数を超過します。たとえば、4つのデータベース・インスタンスがある4 CPUシステムで、各インスタンスを3つのCPUに制限します。この方法でサーバーがオーバーサブスクライブされると、インスタンスが相互にパフォーマンスに影響を及ぼします。ただし、インスタンス・ケージングによってその影響は制限され、パフォーマンスはある程度予測可能になります。ただし、インスタンスの1つに高負荷の期間がある場合、CPUでそれを処理できます。1つ以上のインスタンスが頻繁にアイドルまたは低負荷になる可能性があるため、このアプローチは重要度の低いシステムに適切です。
- パーティション化—このアプローチは、インスタンスが相互に妨害しないようにする必要がある、重要な本番システムに適

しています。すべての割当ての合計がサーバー上のCPU数と等しくなるようにCPUを割り当てます。たとえば、16 CPUサーバー・システムで、最初のインスタンスに8つのCPU、2番目のインスタンスに4つのCPU、残りの2つのインスタンスに2つずつのCPUを割り当てます。CPUリソースを各データベース・インスタンス専用にすることによって、1つのインスタンスへの負荷が別のインスタンスに影響を及ぼすことがなくなり、各インスタンスは想定したとおりに実行されます。

使用率制限が適用されたインスタンス・ケージングの使用

リソース・プランでインスタンス・ケージングを有効にし、使用率制限を設定した場合、絶対上限は割り当てられたCPUリソースの割合として計算されます。

たとえば、インスタンス・ケージングを有効にし、CPU_COUNTを4に設定し、コンシューマ・グループの使用率制限が50%である場合、コンシューマ・グループは4つのCPUの最大50%、つまり2つのCPUを使用できます。

親トピック: 単一サーバーにおける複数のデータベース・インスタンスの管理

27.8.2 インスタンス・ケージングの有効化

CPUディレクティブでリソース・プランを作成し、CPU_COUNT初期化パラメータを設定することで、インスタンス・ケージングの使用を有効にできます。

インスタンス・ケージングを有効化するには、サーバー上の各インスタンスに対して次の処理を実行します。

1. リソース・プランを割り当てることでリソース・マネージャを有効化し、MGMT_P1からMGMT_P8のパラメータを使用してリソース・プランにCPUディレクティブがあるようにします。

   手順については、「Oracle Database Resource Managerの有効化とプランの切替え」を参照してください。

2. cpu_count初期化パラメータを設定します。

   これは動的パラメータであり、次の文を使用して設定できます。

   ```
   ALTER SYSTEM SET CPU_COUNT = 4;
   ```

親トピック: 単一サーバーにおける複数のデータベース・インスタンスの管理

27.9 コンシューマ・グループ、プランおよびディレクティブのメンテナンス

Oracle Database Resource Manager (リソース・マネージャ)のコンシューマ・グループ、リソース・プランおよびリソース・プラン・ディレクティブをメンテナンスできます。メンテナンス・タスクは、DBMS_RESOURCE_MANAGER PL/SQLパッケージを使用して実行します。

- コンシューマ・グループの更新
  コンシューマ・グループ情報を更新するには、UPDATE_CONSUMER_GROUPプロシージャを使用します。
- コンシューマ・グループの削除
  DELETE_CONSUMER_GROUPプロシージャは、指定されたコンシューマ・グループを削除します。
- プランの更新
  プラン情報を更新するには、UPDATE_PLANプロシージャを使用します。
- プランの削除
  DELETE_PLANプロシージャは、指定したプランと、それに対応付けられているすべてのプラン・ディレクティブを削除します。
- リソース・プラン・ディレクティブの更新
  プラン・ディレクティブを更新するには、UPDATE_PLAN_DIRECTIVEプロシージャを使用します。

- リソース・プラン・ディレクティブの削除
  リソース・プラン・ディレクティブを削除するには、DELETE_PLAN_DIRECTIVEプロシージャを使用します。

関連項目:

- 事前定義のコンシューマ・グループ・マッピング・ルール
- DBMS_RESOURCE_MANAGER PL/SQLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.9.1 コンシューマ・グループの更新

コンシューマ・グループ情報を更新するには、UPDATE_CONSUMER_GROUPプロシージャを使用します。

コンシューマ・グループを更新するには:

1. ペンディング・エリアを作成します。
2. UPDATE_CONSUMER_GROUPプロシージャを実行します。

   引数を指定せずにUPDATE_CONSUMER_GROUPプロシージャを実行すると、データ・ディクショナリ内のコンシューマ・グループ情報は変更されません。
3. ペンディング・エリアを発行します。

関連トピック

- ペンディング・エリアの作成
- ペンディング・エリアの発行

親トピック: コンシューマ・グループ、プランおよびディレクティブのメンテナンス

27.9.2 コンシューマ・グループの削除

DELETE_CONSUMER_GROUPプロシージャは、指定されたコンシューマ・グループを削除します。

コンシューマ・グループを削除するには:

1. ペンディング・エリアを作成します。
2. DELETE_CONSUMER_GROUPプロシージャを実行します。
3. ペンディング・エリアを発行します。

コンシューマ・グループを削除すると、そのグループを初期コンシューマ・グループとして割り当てられているすべてのユーザーには、初期コンシューマ・グループとしてOTHER_GROUPSが割り当てられます。削除したコンシューマ・グループに属している現在実行中のセッションはすべて、コンシューマ・グループ・マッピング・ルールに基づいて新しいコンシューマ・グループに割り当てられます。マッピングでセッションのコンシューマ・グループが見つからなかった場合、そのセッションはOTHER_GROUPSに切り替わります。

リソース・プラン・ディレクティブが参照しているコンシューマ・グループは削除できません。

関連トピック

- ペンディング・エリアの作成

- ペンディング・エリアの発行

親トピック: コンシューマ・グループ、プランおよびディレクティブのメンテナンス

27.9.3 プランの更新

プラン情報を更新するには、UPDATE_PLANプロシージャを使用します。

プランを更新するには:

1. ペンディング・エリアを作成します。
2. UPDATE_PLANプロシージャを実行します。たとえば、次のスクリプトは、COMMENTパラメータを更新するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.UPDATE_PLAN(
   PLAN => 'DAYTIME',
   NEW_COMMENT => '50% more resources for OLTP applications');
END;
/
```

   引数を指定せずにUPDATE_PLANプロシージャを実行すると、データ・ディクショナリ内のプラン情報は変更されません。

3. ペンディング・エリアを発行します。

関連トピック

- ペンディング・エリアの作成
- ペンディング・エリアの発行

親トピック: コンシューマ・グループ、プランおよびディレクティブのメンテナンス

27.9.4 プランの削除

DELETE_PLANプロシージャは、指定したプランと、それに対応付けられているすべてのプラン・ディレクティブを削除します。

計画を削除するには:

1. ペンディング・エリアを作成します。
2. DELETE_PLAN_CASCADEプロシージャを実行します。たとえば、次のスクリプトは、great_breadプランとそのディレクティブを削除するPL/SQLブロックです。

```
BEGIN
  DBMS_RESOURCE_MANAGER.DELETE_PLAN(PLAN => 'great_bread');
END;
/
```

   引数を指定せずにUPDATE_PLANプロシージャを実行すると、データ・ディクショナリ内のプラン情報は変更されません。

   削除したディレクティブが参照していたリソース・コンシューマ・グループは削除されませんが、great_breadプランとの関連は解除されます。

   DELETE_PLAN_CASCADEプロシージャは、指定のプランとその子孫すべて(プラン・ディレクティブ、および必須のマークが付けられていないサブプランとリソース・コンシューマ・グループ)を削除します。エラーが発生したDELETE_PLAN_CASCADEはロールバックされ、プランは変更されません。

   現在アクティブなプランは削除できません。

3. ペンディング・エリアを発行します。

関連トピック

- ペンディング・エリアの作成
- ペンディング・エリアの発行

親トピック: コンシューマ・グループ、プランおよびディレクティブのメンテナンス

27.9.5 リソース・プラン・ディレクティブの更新

プラン・ディレクティブを更新するには、UPDATE_PLAN_DIRECTIVEプロシージャを使用します。

リソース・プラン・ディレクティブを更新するには:

1. ペンディング・エリアを作成します。
2. UPDATE_PLAN_DIRECTIVEプロシージャを実行します。

   次の例ではディレクティブにコメントを追加します。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
         PLAN              => 'SIMPLE_PLAN1',
         GROUP_OR_SUBPLAN => 'MYGROUP1',
         NEW_COMMENT       => 'Higher priority'
        );
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

   コメントをクリアする(NULLにする)には、NULL文字列('')を渡します。数値のディレクティブ・パラメータをクリアする(0またはNULLにする)には、新しい値を-1に設定します。

```
BEGIN
  DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
         PLAN                   => 'SIMPLE_PLAN1',
         GROUP_OR_SUBPLAN       => 'MYGROUP1',
         NEW_MAX_EST_EXEC_TIME => -1
        );
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

   UPDATE_PLAN_DIRECTIVEプロシージャの引数を指定しない場合、ディレクティブ内の対応パラメータは変更されません。

3. ペンディング・エリアを発行します。

関連トピック

- ペンディング・エリアの作成
- ペンディング・エリアの発行

親トピック: コンシューマ・グループ、プランおよびディレクティブのメンテナンス

27.9.6 リソース・プラン・ディレクティブの削除

リソース・プラン・ディレクティブを削除するには、DELETE_PLAN_DIRECTIVEプロシージャを使用します。

リソース・プラン・ディレクティブを削除するには:

1. ペンディング・エリアを作成します。
2. DELETE_PLAN_DIRECTIVEプロシージャを実行します。
3. ペンディング・エリアを発行します。

関連トピック

- ペンディング・エリアの作成
- ペンディング・エリアの発行

親トピック: コンシューマ・グループ、プランおよびディレクティブのメンテナンス

27.10 データベース・リソース・マネージャの構成とステータスの表示

Oracle Database Resource Manager (リソース・マネージャ)の現在の構成とステータスを表示するには、いくつかの静的データ・ディクショナリ・ビューと動的パフォーマンス・ビューを使用できます。

- ユーザーまたはロールに権限付与されたコンシューマ・グループの表示
  DBA_RSRC_CONSUMER_GROUP_PRIVSビューには、ユーザーまたはロールに付与されているコンシューマ・グループが表示されます。
- プラン情報の表示
  DBA_RSRC_PLANSビューを使用して、データベースに定義されているすべてのリソース・プランを表示する例を示します。
- セッションの現行コンシューマ・グループの表示
  V$SESSIONビューを使用すれば、セッションに現在割り当てられているコンシューマ・グループを表示できます。
- 現在アクティブなプランの表示
  V$RSRC_PLANビューには現在アクティブなプランが表示されます。

関連項目:

すべての静的データ・ディクショナリ・ビューと動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.10.1 ユーザーまたはロールに権限付与されたコンシューマ・グループの表示

DBA_RSRC_CONSUMER_GROUP_PRIVSビューには、ユーザーまたはロールに付与されているコンシューマ・グループが表示されます。

具体的には、ユーザーまたはロールが属することが許可されているグループ、または切替え可能なグループが表示されます。たとえば、次に示すビューの場合、ユーザーSCOTTは常にSALESコンシューマ・グループで開始され、特定の付与によってMARKETINGグループに切り替えることが可能であり、さらにDEFAULT_CONSUMER_GROUP (OTHER_GROUPS)グループとLOW_GROUPグループはPUBLICに付与されているので、これらのグループに切り替えることができます。また、SCOTTは他の

ユーザーにSALESグループを付与できますが、MARKETINGグループは付与できません。

```
SELECT * FROM dba_rsrc_consumer_group_privs;
GRANTEE             GRANTED_GROUP                  GRANT_OPTION INITIAL_GROUP
------------------- ------------------------------ ------------ -------------
PUBLIC              DEFAULT_CONSUMER_GROUP         YES          YES
PUBLIC              LOW_GROUP                      NO           NO
SCOTT               MARKETING                      NO           NO
SCOTT               SALES                          YES          YES
SYSTEM              SYS_GROUP                      NO           YES
```

SCOTTには、DBMS_RESOURCE_MANAGER_PRIVSパッケージを使用してこれらのグループに切り替える機能がすでに付与されています。

親トピック: データベース・リソース・マネージャの構成とステータスの表示

27.10.2 プラン情報の表示

DBA_RSRC_PLANSビューを使用して、データベースに定義されているすべてのリソース・プランを表示する例を示します。

すべてのプランのステータスは、ペンディング・エリアに存在しないことを意味するNULLです。

ノート:

ペンディング・エリア内のプランのステータスは PENDING です。ペンディング・エリア内のプランは編集中です。

```
SELECT plan,status,comments FROM dba_rsrc_plans;
```

PLAN STATUS COMMENTS ---------- -------- ---------- DSS_PLAN Example plan for DSS workloads that prio... ETL_CRITICAL_PLAN Example plan for DSS workloads that prio... MIXED_WORKLOAD_PLAN Example plan for a mixed workload that p... DEFAULT_MAINTENANCE_PLAN Default plan for maintenance windows tha... DEFAULT_PLAN Default, basic, pre-defined plan that pr... INTERNAL_QUIESCE Plan for quiescing the database. This p... INTERNAL_PLAN Internally-used plan for disabling the r... . . .

親トピック: データベース・リソース・マネージャの構成とステータスの表示

27.10.3 セッションの現行コンシューマ・グループの表示

V$SESSIONビューを使用すれば、セッションに現在割り当てられているコンシューマ・グループを表示できます。

次の例では、V$SESSIONビューを問い合せます。

```
SELECT sid,serial#,username,resource_consumer_group FROM v$session;
SID    SERIAL#  USERNAME                  RESOURCE_CONSUMER_GROUP
-----  -------  ------------------------  --------------------------------
   11       136 SYS                       SYS_GROUP
   13     16570 SCOTT                     SALES
   ...
```

親トピック: データベース・リソース・マネージャの構成とステータスの表示

27.10.4 現在アクティブなプランの表示

V$RSRC_PLANビューには現在アクティブなプランが表示されます。

この例では、mydb_plan (「複数レベルのプランの例」の例で作成したもの)をトップレベルのプランに設定します。次に、V$RSRC_PLANビューを問い合せて、現在アクティブなプランを表示します。このビューには、現在のトップレベルのプランとそのすべての子サブプランが表示されます。

```
ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = mydb_plan;
System altered.
SELECT name, is_top_plan FROM v$rsrc_plan;
NAME             IS_TOP_PLAN
---------------------------
MYDB_PLAN        TRUE
MAILDB_PLAN      FALSE
BUGDB_PLAN       FALSE
```

親トピック: データベース・リソース・マネージャの構成とステータスの表示

27.11 Oracle Database Resource Managerの監視

動的パフォーマンス・ビューのセットにより、Oracle Database Resource Manager設定の結果を監視できます。

次の動的パフォーマンス・ビューは、Oracle Database Resource Managerの設定結果の監視に役立ちます。

- V$RSRC_PLAN
- V$RSRC_CONSUMER_GROUP
- V$RSRC_SESSION_INFO
- V$RSRC_PLAN_HISTORY
- V$RSRC_CONS_GROUP_HISTORY
- V$RSRCMGRMETRIC
- V$RSRCMGRMETRIC_HISTORY

これらのビューには、次の情報が表示されます。

- 現在のステータス情報
- リソース・プランのアクティブ化の履歴
- リソース・コンシューマ・グループとセッションによるリソース使用とCPU待ちに関する現在の統計と履歴の統計

さらに、履歴統計は、DBA_HIST_RSRC_PLANおよびDBA_HIST_RSRC_CONSUMER_GROUPの各ビューを介して表示できます。これらのビューには、それぞれV$RSRC_PLAN_HISTORYとV$RSRC_CONS_GROUP_HISTORYの自動ワークロード・リポジトリ(AWR)スナップショットが含まれています。

チューニングに役立つように、V$RSRCMGRMETRICおよびV$RSRCMGRMETRIC_HISTORYの各ビューには、過去1時間に、CPU待ちに費やした時間とCPUの使用量が、コンシューマ・グループごとに分単位で表示されます。Cloud Controlでは、これらのメトリックも「リソース・マネージャ統計」ページでグラフィカルに表示できます。

リソース・マネージャを使用可能にすると、リソース・マネージャによってリソース使用率に関する統計が自動的に記録され、リアルタイムSQL監視およびリソース・マネージャの動的パフォーマンス・ビューを使用して、それらの統計を検証できます。

リアルタイムSQL監視を使用するには、Cloud Controlで「SQLモニター」ページにアクセスするか、またはV$SQL_MONITORビューおよび他の関連するビューを問い合せます。V$SQL_MONITORビューには、リソース・マネージャによってコンシューマ・グループについて実行された最後の処理に関する情報も、RM_CONSUMER_GROUP、RM_LAST_ACTION、RM_LAST_ACTION_REASONおよびRM_LAST_ACTION_TIMEの列に含まれます。

また、次の動的パフォーマンス・ビューにはリソース使用率に関する統計が含まれています。

- V$RSRCMGRMETRIC
- V$RSRCMGRMETRIC_HISTORY
- V$RSRC_CONSUMER_GROUP
- V$RSRC_CONS_GROUP_HISTORY

関連項目:

リアルタイムSQL監視の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

V$RSRC_PLAN

このビューには、現在アクティブなリソース・プランとそのサブプランが表示されます。

```
SELECT name, is_top_plan FROM v$rsrc_plan;

NAME                             IS_TOP_PLAN
-------------------------------- -----------
DEFAULT_PLAN                     TRUE
ORA$AUTOTASK                     FALSE
ORA$AUTOTASK_HIGH_SUB_PLAN       FALSE
```

IS_TOP_PLANがTRUEのプランは、現在アクティブな(トップレベルの)プランです。他のプランは、トップレベルのプランのサブプランか、リスト内の他のサブプランのサブプランです。

このビューには、その他に次の情報なども含まれます。

- INSTANCE_CAGING列に、インスタンス・ケージングが有効であるかどうかが表示されます。
- CPU_MANAGED列に、CPUが管理されているかどうかが表示されます。
- PARALLEL_EXECUTION_MANAGED列に、パラレル・ステートメント・キューイングが有効であるかどうかが表示されます。

関連項目:

Oracle Databaseリファレンス

V$RSRC_CONSUMER_GROUP

V$RSRC_CONSUMER_GROUPビューは、CPU、I/O、パラレル実行サーバーなど、使用されているリソースを監視する場合に使用します。また、CPUリソースの管理、リソース集中型の問合せの管理、パラレル・ステートメント・キューイングなどに関連する統計を監視する場合にも使用できます。すべての統計は、プランがアクティブ化された時点からの累積です。

```
SELECT name, active_sessions, queue_length,
  consumed_cpu_time, cpu_waits, cpu_wait_time
  FROM v$rsrc_consumer_group;
NAME               ACTIVE_SESSIONS QUEUE_LENGTH CONSUMED_CPU_TIME  CPU_WAITS
CPU_WAIT_TIME
------------------ --------------- ------------ ----------------- ---------- --------
-----
OLTP_ORDER_ENTRY                 1            0             29690        467
6709
OTHER_GROUPS                     0            0           5982366       4089
60425
SYS_GROUP                        1            0           2420704        914
19540
```

```
DSS_QUERIES                    4            2          4594660       3004
55700
```

前述の問合せ結果では、DSS_QUERIESコンシューマ・グループには、アクティブ・セッション・プールに4つのセッションがあり、2つのセッションがキューでアクティブ化を待機しています。

このビューの主な測定値はCPU_WAIT_TIMEです。この測定値は、リソース管理のために、コンシューマ・グループのセッションがCPUの割当てを待機していた合計時間を示します。この測定値には、ラッチまたはエンキューの競合、I/O待機などに起因する待機は含まれません。

ノート:

STATISTICS_LEVEL 初期化パラメータが ALL または TYPICAL に設定されている場合、V$RSRC_CONSUMER_GROUP ビューでは、リソース・マネージャによって現在管理されていないリソースの統計が記録されます。

関連項目:

Oracle Databaseリファレンス

V$RSRC_SESSION_INFO

このビューを使用して、1つ以上のセッションのステータスを監視します。このビューには、リソース・マネージャによるセッションへの影響が表示されます。次の情報が提供されます。

- セッションが現在属しているコンシューマ・グループ。
- セッションが当初属していたコンシューマ・グループ。
- コンシューマ・グループへのセッションのマップに使用されたセッション属性。
- セッションの状態(RUNNING、WAIT_FOR_CPU、QUEUEDなど)。
- メトリックに関する現在の統計と累積の統計(CPU使用、待機時間、待ち行列時間、使用されるアクティブなパラレル・サーバーの数など)。現在の統計には、セッションが現在のコンシューマ・グループに所属してからの統計が反映されています。累積の統計には、セッションが作成された以降に所属したすべてのコンシューマ・グループでの統計が反映されています。

```
SELECT se.sid sess_id, co.name consumer_group,
 se.state, se.consumed_cpu_time cpu_time, se.cpu_wait_time, se.queued_time
 FROM v$rsrc_session_info se, v$rsrc_consumer_group co
 WHERE se.current_consumer_group_id = co.id;
SESS_ID CONSUMER_GROUP     STATE     CPU_TIME CPU_WAIT_TIME QUEUED_TIME
------- ------------------ -------- --------- ------------- -----------
    113 OLTP_ORDER_ENTRY   WAITING     137947         28846           0
    135 OTHER_GROUPS       IDLE        785669         11126           0
    124 OTHER_GROUPS       WAITING      50401         14326           0
    114 SYS_GROUP          RUNNING        495             0           0
    102 SYS_GROUP          IDLE         88054            80           0
    147 DSS_QUERIES        WAITING     460910        512154           0
```

このビューのCPU_WAIT_TIMEには、V$RSRC_CONSUMER_GROUPビューと同様の意味がありますが、個々のセッションに適用されます。

このビューとV$SESSIONビューを結合して、さらに詳細なセッションに関する情報を表示できます。

関連項目:

- Oracle Databaseリファレンス
- Oracle Database VLDBおよびパーティショニング・ガイド

V$RSRC_PLAN_HISTORY

このビューには、インスタンスでリソース・プランがいつ有効または無効になったかが表示されます。各リソース・プランのアクティブ化または解除には連番が割り当てられます。このビューの各エントリについては、プランの各コンシューマ・グループに対応するエントリがV$RSRC_CONS_GROUP_HISTORYビューにあり、コンシューマ・グループの累積統計が表示されます。この2つのビューは、それぞれのSEQUENCE#列で結合しています。

```
SELECT sequence# seq, name plan_name,
to_char(start_time, 'DD-MON-YY HH24:MM') start_time,
to_char(end_time, 'DD-MON-YY HH24:MM') end_time, window_name
FROM v$rsrc_plan_history;
 SEQ PLAN_NAME                    START_TIME      END_TIME        WINDOW_NAME
---- ---------------------------- --------------- --------------- ---------------
   1                              29-MAY-07 23:05 29-MAY-07 23:05
   2 DEFAULT_MAINTENANCE_PLAN     29-MAY-07 23:05 30-MAY-07 02:05 TUESDAY_WINDOW
   3                              30-MAY-07 02:05 30-MAY-07 22:05
   4 DEFAULT_MAINTENANCE_PLAN     30-MAY-07 22:05 31-MAY-07 02:05 WEDNESDAY_WINDOW
   5                              31-MAY-07 02:05 31-MAY-07 22:05
   6 DEFAULT_MAINTENANCE_PLAN     31-MAY-07 22:05                 THURSDAY_WINDOW
```

PLAN_NAMEの下のNULL値は、プランがアクティブにならなかったことを示します。

このビューのAWRスナップショットはDBA_HIST_RSRC_PLANビューに格納されます。

関連項目:

Oracle Databaseリファレンス

V$RSRC_CONS_GROUP_HISTORY

このビューは、時間の経過とともに、リソースがコンシューマ・グループ間でどのように共有されていたかを理解するのに役立ちます。sequence#列は、V$RSRC_PLAN_HISTORYビューの同名の列に対応しています。そのため、コンシューマ・グループ統計の各行について、アクティブであったプランを判別できます。

```
SELECT sequence# seq, name, cpu_wait_time, cpu_waits,
consumed_cpu_time FROM v$rsrc_cons_group_history;

 SEQ NAME                      CPU_WAIT_TIME  CPU_WAITS CONSUMED_CPU_TIME
---- ------------------------ ------------- ---------- ----------------
   2 SYS_GROUP                        18133        691         33364431
   2 OTHER_GROUPS                     51252        825        181058333
   2 ORA$AUTOTASK_MEDIUM_GROUP           21          5          4019709
   2 ORA$AUTOTASK_URGENT_GROUP           35          1           198760
   2 ORA$AUTOTASK_STATS_GROUP             0          0                0
   2 ORA$AUTOTASK_SPACE_GROUP             0          0                0
   2 ORA$AUTOTASK_SQL_GROUP               0          0                0
   2 ORA$AUTOTASK_HEALTH_GROUP            0          0                0
   4 SYS_GROUP                        40344         85         42519265
   4 OTHER_GROUPS                    123295       1040        371481422
   4 ORA$AUTOTASK_MEDIUM_GROUP            1          4          7433002
   4 ORA$AUTOTASK_URGENT_GROUP        22959        158         19964703
```

```
  4 ORA$AUTOTASK_STATS_GROUP              0          0                  0
    .
    .
```

このビューのAWRスナップショットはDBA_HIST_RSRC_CONSUMER_GROUPビューに格納されます。コンシューマ・グループ統計の各履歴セットについて、アクティブであったプランを判別するには、DBA_HIST_RSRC_CONSUMER_GROUPとDBA_HIST_RSRC_PLANを併用します。

> ノート:
>
> STATISTICS_LEVEL 初期化パラメータが ALL または TYPICAL に設定されている場合、V$RSRC_CONS_GROUP_HISTORY ビューでは、リソース・マネージャによって現在管理されていないリソースの統計が記録されます。

関連項目:

- Oracle Databaseリファレンス
- AWRの詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

V$RSRCMGRMETRIC

このビューを使用すると、セッション数の観点または過去1分間の使用状況の観点から、CPUメトリックをミリ秒単位で追跡できます。各コンシューマ・グループのリアルタイムなメトリックが提供されるため、ワークロードの実行中に、CPUリソースの使用状況を継続的に監視する場合に非常に便利です。

このビューを使用すると、コンシューマ・グループの最大CPU使用率と平均CPU使用率を、使用したCPU時間、CPU待ち時間、CPUを消費しているセッションの平均数、CPU割当てを待機しているセッションの数など他のコンシューマ・グループ設定と比較できます。たとえば、コンシューマ・グループが使用したCPUリソースの量およびコンシューマ・グループがリソース割当てを待機した期間を表示できます。または、各コンシューマ・グループから実行されているセッションの数とアクティブなセッションの総数とを対比できます。

CPU使用率でCPUの消費量を追跡するには、CPU_UTILIZATION_LIMIT列およびAVG_CPU_UTILIZATION列を使用します。AVG_CPU_UTILIZATION列には、コンシューマ・グループによるサーバーのCPUの平均使用率が表示されます。CPU_UTILIZATION_LIMIT列は、コンシューマ・グループが使用できるサーバーのCPUの最大割合を表します。この制限は、UTILIZATION_LIMITディレクティブ属性を使用して設定されます。

```
SELECT consumer_group_name, cpu_utilization_limit,
avg_cpu_utilization FROM v$rsrcmgrmetric;
```

CPUの消費量および制限量をミリ秒単位で追跡するには、CPU_CONSUMED_TIME列およびCPU_TIME_WAIT列を使用します。NUM_CPUS列は、リソース・マネージャが管理しているCPUの数を表します。

```
SELECT consumer_group_name, cpu_consumed_time,
cpu_wait_time, num_cpus FROM v$rsrcmgrmetric;
```

CPUの消費量および制限量をセッションの数で追跡するには、RUNNING_SESSIONS_LIMIT、AVG_RUNNING_SESSIONS、AVG_WAITING_SESSIONSの各列を使用します。RUNNING_SESSIONS_LIMIT列には、特定のコンシューマ・グループからいつでも実行できるセッションの最大数が表示されます。この制限は、コンシューマ・グループまたはコンシューマ・グループが含まれているサブプランに設定するUTILIZATION_LIMITディレクティブ属性で定義されます。

コンシューマ・グループごとに、AVG_RUNNING_SESSIONS列にはCPUを消費しているセッションの平均数が表示され、AVG_WAITING_SESSIONS列にはCPUを待機しているセッションの平均数が表示されます。

```
SELECT sequence#, consumer_group_name, running_sessions_limit,
avg_running_sessions, avg_waiting_sessions FROM v$rsrcmgrmetric;
```

パラレル・ステートメントおよびコンシューマ・グループのパラレル・サーバーの使用を追跡するには、AVG_ACTIVE_PARALLEL_STMTS、AVG_QUEUED_PARALLEL_STMTS、AVG_ACTIVE_PARALLEL_SERVERS、AVG_QUEUED_PARALLEL_SERVERSおよびPARALLEL_SERVERS_LIMIT列を使用します。AVG_ACTIVE_PARALLEL_STMTSとAVG_ACTIVE_PARALLEL_SERVERSには、実行されているパラレル・ステートメントの平均数、およびパラレル・ステートメントによって使用されるパラレル・サーバーの平均数が表示されます。AVG_QUEUED_PARALLEL_STMTSとAVG_QUEUED_PARALLEL_SERVERSには、キュー内のパラレル・ステートメントの平均数、およびキュー内のパラレル・ステートメントによって要求されたパラレル・サーバーの平均数が表示されます。PARALLEL_SERVERS_LIMITには、コンシューマ・グループで使用できるパラレル・サーバーの数が表示されます。

```
SELECT avg_active_parallel_stmts, avg_queued_parallel_stmts,
avg_active_parallel_servers, avg_queued_parallel_servers, parallel_servers_limit
FROM v$rsrcmgrmetric;
```

ノート:

STATISTICS_LEVEL 初期化パラメータが ALL または TYPICAL に設定されている場合、V$RSRCMGRMETRIC ビューでは、リソース・マネージャによって現在管理されていないリソースの統計が記録されます。

関連項目:

Oracle Databaseリファレンス

V$RSRCMGRMETRIC_HISTORY

V$RSRCMGRMETRIC_HISTORYの各列は、V$RSRCMGRMETRICと同じビューです。これらのビューの唯一の違いは、V$RSRCMGRMETRICには過去1分間のみのメトリックが含まれているのに対して、V$RSRCMGRMETRIC_HISTORYには過去60分間のメトリックが含まれていることです。

ノート:

STATISTICS_LEVEL 初期化パラメータが ALL または TYPICAL に設定されている場合、V$RSRCMGRMETRIC_HISTORY ビューでは、リソース・マネージャによって現在管理されていないリソースの統計が記録されます。

関連項目:

Oracle Databaseリファレンス

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.12 オペレーティング・システムのリソース制御との相互作用

多くのオペレーティング・システムではリソース管理ツールを提供しています。これらのツールの名前には、「workload manager」や「resource manager」などの語句が含まれており、管理者が定義したポリシーを使用して、単一のサーバー・リソースを複数のアプリケーションで共有できることを意図しています。たとえば、HP社のProcess Resource Managerや、Solaris Containers、ZonesおよびResource Poolsなどがあります。

- オペレーティング・システムのリソース制御を使用するためのガイドライン
  オペレーティング・システムのリソース制御を使用する場合は、ガイドラインに従います。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.12.1 オペレーティング・システムのリソース制御を使用するためのガイドライン

オペレーティング・システムのリソース制御を使用する場合は、ガイドラインに従います。

オペレーティング・システムによるリソース制御をOracle Databaseと併用する場合は、次のガイドラインに従って慎重に使用する必要があります。

- 1つのノードに複数のインスタンスがあり、それらの間でリソースを配分する場合は、各インスタンスを専用のオペレーティング・システム・リソース・マネージャ・グループまたは管理対象エンティティに割り当てる必要があります。管理対象エンティティの複数のインスタンスを実行するには、インスタンス・ケージングを使用して管理対象エンティティ内のCPUリソースをインスタンス間で配分する方法を管理します。Oracle Database Resource Managerは、CPUリソースを管理するときに、各インスタンスに対するCPUリソースが一定量であると予測します。インスタンス・ケージングを使用しない場合は、使用可能なCPUリソースは管理対象エンティティ内のCPUの数と同じであると予測します。インスタンス・ケージングを使用する場合、使用可能なCPUリソースはCPU_COUNT初期化パラメータの値と同じであると予測します。CPUリソースが想定よりも少ない場合、Oracle Database Resource Managerはリソース・プランでのリソース割当ての強制が有効になりません。インスタンス・ケージングの詳細は、「単一サーバーにおける複数のデータベース・インスタンスの管理」を参照してください。
- インスタンスのすべてのプロセスを実行している専用エンティティが、1つの優先度(リソース使用)レベルで実行されている必要があります。
- 専用エンティティに割り当てたCPUリソースは、数分に1回の割合を超える頻度では変更できません。

  オペレーティング・システムのリソース・マネージャによって、Oracleインスタンスに割り当てられているCPUリソースが急激に変更された場合、Oracle Database Resource ManagerがCPUリソースを効果的に管理できないことがあります。特に、Oracleインスタンスに割り当てられているCPUリソースが2、3分未満の間隔で頻繁に変更されると、Oracleでは2、3分ごとにしかこのような変更をチェックしないため、Oracleで観測されない可能性があります。このような場合、Oracle Database Resource Managerで、実際の使用可能量より多くのCPUリソースが使用できると判断されて多すぎるプロセスがスケジュールされたり、実際の使用可能量より少ないCPUリソースしか使用できないと判断されて少なすぎるプロセスがスケジュールされる可能性があります。多すぎるプロセスがスケジュールされた場合、UTILIZATION_LIMITディレクティブを超えてしまい、CPUディレクティブが正しく規定されない可能性があります。少なすぎるプロセスがスケジュールされた場合、Oracleインスタンスによってサーバーのリソースが十分に使用されない可能性があります。
- プロセスの優先度管理が使用禁止になっている必要があります。
- 個々のデータベース・プロセスを異なる優先度レベルで管理する機能(たとえば、UNIXプラットフォームでのniceコマンドの使用)は、サポートされていません。インスタンスがクラッシュするなど、重大な結果になる可能性があります。オペ

レーティング・システムのリソース制御を使用して、Oracle Databaseインスタンスが固定されているメモリーを管理できる場合も、同様に望ましくない結果になることが予想されます。

親トピック: オペレーティング・システムのリソース制御との相互作用

27.13 Oracle Database Resource Managerの参照情報

リソース・マネージャには、事前定義済のリソース・プラン、コンシューマ・グループおよびコンシューマ・グループ・マッピング・ルールが含まれます。リソース・マネージャ構成に関する情報をデータ・ディクショナリ・ビューに問い合せることができます。

- 事前定義のリソース・プランおよびコンシューマ・グループ
  Oracle Databaseには、事前定義のリソース・プランが含まれます。
- 事前定義のコンシューマ・グループ・マッピング・ルール
  Oracle Databaseには、事前定義済のコンシューマ・グループ・マッピング・ルールが含まれます。
- リソース・マネージャのデータ・ディクショナリ・ビュー
  データ・ディクショナリ・ビューのセットに対してデータベース・リソース管理に関する情報を問い合せることができます。

親トピック: Oracle Database Resource Managerを使用したリソースの管理

27.13.1 事前定義のリソース・プランおよびコンシューマ・グループ

Oracle Databaseには、事前定義のリソース・プランが含まれます。

各Oracle Databaseに事前定義されているリソース・プランを表27-5にリストし、リソース・コンシューマ・グループを表27-6にリストします。ビューDBA_RSRC_PLANSおよびDBA_RSRC_CONSUMER_GROUPSを問い合せることによって、これらの内容を確認できます。

次の問合せにより、プラン例DSS_PLANでのCPU割当てが表示されます。

```
SELECT group_or_subplan, mgmt_p1, mgmt_p2, mgmt_p3, mgmt_p4
   FROM dba_rsrc_plan_directives WHERE plan = 'DSS_PLAN';

GROUP_OR_SUBPLAN                  MGMT_P1    MGMT_P2    MGMT_P3    MGMT_P4
------------------------------ ---------- ---------- ---------- ----------
SYS_GROUP                              75          0          0          0
DSS_CRITICAL_GROUP                     18          0          0          0
DSS_GROUP                               3          0          0          0
ETL_GROUP                               1          0          0          0
BATCH_GROUP                             1          0          0          0
ORA$AUTOTASK                            1          0          0          0
OTHER_GROUPS                            1          0          0          0
```

表27-5 事前定義のリソース・プラン

リソース・プラン	説明
DEFAULT_MAINTENANCE_PLAN	メンテナンス・ウィンドウのデフォルト・プラン。このプランの詳細は、「自動化メンテナンス・タスクに対するリソース割当てについて」を参照してください。メンテナンス・ウィンドウはOracle Scheduler の通常のウィンドウであるため、必要に応じて、それらに関連付けられているリソース・プランを変更できます。メンテナンス・ウィンドウのリソース・プランを変更する場合は、サブプラン ORA$AUTOTASK が新しいプランに含まれていることを確認してください。

リソース・プラン	説明
DEFAULT_PLAN	SYS_GROUP 操作の優先度を設定し、自動化メンテナンス操作と診断操作に最小限のリソースを割り当てる基本のデフォルト・プラン。
DSS_PLAN	重要ではない DSS 問合せや ETL 操作よりも重要な DSS 問合せを優先させるデータ・ウェアハウス・プランの例。
ETL_CRITICAL_PLAN	DSS 問合せよりも ETL 操作を優先させるデータ・ウェアハウス・プランの例。
INTERNAL_PLAN	リソース・マネージャの無効化用。内部使用のみに対応しています。
INTERNAL_QUIESCE	データベースの静止用。このプランは直接アクティブ化できません。アクティブ化するには QUIESCE コマンドを使用します。
MIXED_WORKLOAD_PLAN	バッチ操作よりも対話型操作を優先させる複合ワークロード・プランの例。詳細は、「オラクル社が提供する複合ワークロード・プラン」を参照してください。

表27-6 事前定義のリソース・コンシューマ・グループ

リソース・コンシューマ・グループ	説明
BATCH_GROUP	バッチ操作用のコンシューマ・グループ。プラン例 MIXED_WORKLOAD_PLAN によって参照されます。
DSS_CRITICAL_GROUP	重要な DSS 問合せ用のコンシューマ・グループ。プラン例 DSS_PLAN および ETL_CRITICAL_PLAN によって参照されます。
DSS_GROUP	重要ではない DSS 問合せ用のコンシューマ・グループ。プラン例 DSS_PLAN および ETL_CRITICAL_PLAN によって参照されます。
ETL_GROUP	ETL ジョブ用のコンシューマ・グループ。プラン例 DSS_PLAN および ETL_CRITICAL_PLAN によって参照されます。
INTERACTIVE_GROUP	対話型 OLTP 操作用のコンシューマ・グループ。プラン例 MIXED_WORKLOAD_PLAN によって参照されます。
LOW_GROUP	優先度が低いセッション用のコンシューマ・グループ。
ORA$AUTOTASK	メンテナンス・タスク用のコンシューマ・グループ。
OTHER_GROUPS	明示的な初期コンシューマ・グループを持たないすべてのセッションのデフォルトのコン

リソース・コンシューマ・グループ	説明
	シューマ・グループは、コンシューマ・グループへのセッションのマッピング・ルールによってコンシューマ・グループにマップされないか、現在アクティブなリソース・プラン内にないコンシューマ・グループにマップされます。 OTHER_GROUPS は、すべてのプランで指定されているリソース・プラン・ディレクティブを保持する必要があります。これをマッピング・ルールによって明示的にセッションに割り当てることはできません。
SYS_GROUP	システム管理者用のコンシューマ・グループ。これは、ユーザー・アカウント SYS または SYSTEM によって作成されたすべてのセッションの初期コンシューマ・グループです。この初期コンシューマ・グループは、コンシューマ・グループへのセッションのマッピング・ルールで上書きできます。

親トピック: Oracle Database Resource Managerの参照情報

27.13.2 事前定義のコンシューマ・グループ・マッピング・ルール

Oracle Databaseには、事前定義済のコンシューマ・グループ・マッピング・ルールが含まれます。

表27-7に、Oracle Databaseで事前定義されているコンシューマ・グループ・マッピング・ルールの要約を示します。ビューDBA_RSRC_GROUP_MAPPINGSを問い合せることで、これらのルールを確認できます。DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPINGプロシージャを使用して、これらのマッピング・ルールを変更または削除できます。

表27-7 事前定義のコンシューマ・グループ・マッピング・ルール

属性	値	マップされるコンシューマ・グループ	ノート
ORACLE_USER	SYS	SYS_GROUP	
ORACLE_USER	SYSTEM	SYS_GROUP	
ORACLE_FUNCTION	BACKUP	BATCH_GROUP	セッションでは、RMAN によるバックアップ操作が実行されます。操作が開始すると、セッションは自動的に BATCH_GROUP に切り替えられます。
ORACLE_FUNCTION	COPY	BATCH_GROUP	セッションでは、RMAN によるコピー操作が実行されます。操作が開始すると、セッションは自動的に BATCH_GROUP に切り替えられます。
ORACLE_FUNCTION	DATALOAD	ETL_GROUP	セッションでは、データ・ポンプによるデータ・ロード操作が実行されます。操作が開始すると、セッションは自動的に ETL_GROUP に切り替えられま

属性	値	マップされるコンシューマ・グループ	ノート
			す。

関連項目:

「コンシューマ・グループへのセッションのマッピング・ルールの指定」

親トピック: Oracle Database Resource Managerの参照情報

27.13.3 リソース・マネージャのデータ・ディクショナリ・ビュー

データ・ディクショナリ・ビューのセットに対してデータベース・リソース管理に関する情報を問い合せることができます。

表27-8に、リソース・マネージャに関連付けられているビューをリストします。

表27-8 リソース・マネージャのデータ・ディクショナリ・ビュー

ビュー	説明
DBA_RSRC_CONSUMER_GROUP_PRIVS USER_RSRC_CONSUMER_GROUP_PRIVS	DBA ビューには、すべてのリソース・コンシューマ・グループと、それが付与されているユーザーおよびロールがリストされます。USER ビューには、ユーザーに付与されたすべてのリソース・コンシューマ・グループがリストされます。
DBA_RSRC_CONSUMER_GROUPS	データベースに存在するすべてのリソース・コンシューマ・グループがリストされます。
DBA_RSRC_MANAGER_SYSTEM_PRIVS USER_RSRC_MANAGER_SYSTEM_PRIVS	DBA ビューには、リソース・マネージャのシステム権限が付与されているユーザーとロールがすべてリストされます。USER ビューには、DBMS_RESOURCE_MANAGER パッケージのシステム権限が付与されているユーザーがすべてリストされます。
DBA_RSRC_PLAN_DIRECTIVES	データベースに存在するすべてのリソース・プラン・ディレクティブがリストされます。
DBA_RSRC_PLANS	データベースに存在するすべてのリソース・プランがリストされます。
DBA_RSRC_GROUP_MAPPINGS	すべてのセッション属性に対する様々なマッピングのペアがすべてリストされます。
DBA_RSRC_MAPPING_PRIORITY	各属性の現在のマッピング優先度がリストされます。
DBA_HIST_RSRC_PLAN	リソース・プランのアクティブ化に関する履歴情報が表示されます。このビューに

ビュー	説明
	は、V$RSRC_PLAN_HISTORY の AWR スナップショットが含まれています。
DBA_HIST_RSRC_CONSUMER_GROUP	コンシューマ・グループに関する履歴統計情報が表示されます。このビューには、V$RSRC_CONS_GROUP_HISTORY の AWR スナップショットが含まれています。
DBA_USERS USER_USERS	DBA ビューには、データベースのすべてのユーザーに関する情報が含まれます。これには、各ユーザーの初期リソース・コンシューマ・グループが含まれます。USER ビューには、現行ユーザーに関する情報が含まれます。これには、現行ユーザーの初期リソース・コンシューマ・グループが含まれます。
V$RSRC_CONS_GROUP_HISTORY	V$RSRC_PLAN_HISTORY ビューの各エントリについて、プランの各コンシューマ・グループに対応するエントリが含まれており、コンシューマ・グループの累積統計が表示されます。
V$RSRC_CONSUMER_GROUP	アクティブなリソース・コンシューマ・グループに関する情報が表示されます。このビューは、チューニングに使用できます。
V$RSRCMGRMETRIC	過去 1 分間のリソース使用履歴と累積 CPU 待ち時間(リソース管理に起因するもの)がコンシューマ・グループごとに表示されます。
V$RSRCMGRMETRIC_HISTORY	過去 1 時間のリソース使用履歴と累積 CPU 待ち時間(リソース管理に起因するもの)がコンシューマ・グループごとに分単位で表示されます。新しいリソース・プランが有効な場合、履歴はクリアされます。
V$RSRC_PLAN	現在のアクティブ・リソース・プランの名前がすべて表示されます。
V$RSRC_PLAN_HISTORY	そのインスタンスでリソース・マネージャのプランがいつ有効または無効になったかが表示されます。時間の経過とともに、どのようにリソースがコンシューマ・グループ間で共有されていたかを理解するのに役立ちます。
V$RSRC_SESSION_INFO	各セッションについてリソース・マネージャの統計が表示されます。リソース・マネージャによるセッションへの影響が表示されます。チューニングに使用できます。
V$SESSION	現行セッションごとにセッション情報が表示されます。その中には、各現行セッションのリソース・コンシューマ・グループの名前が含まれます。

親トピック: Oracle Database Resource Managerの参照情報

28 Oracle Schedulerの概要

Oracle Schedulerを使用してタスクをスケジュールできます。

- Oracle Schedulerの概要
  数百または数千ものタスクのスケジューリングを簡素化するために、Oracle Databaseにはエンタープライズ・ジョブ・スケジューラ、Oracle Schedulerが組み込まれています。Oracle Scheduler(スケジューラ)は、DBMS_SCHEDULER PL/SQLパッケージのプロシージャおよびファンクションにより実装されます。
- ジョブおよびスケジューラ・オブジェクトのサポート
  タスクをスケジュールするためにジョブおよびその他のスケジューラ・オブジェクトを使用します。
- ジョブに関する追加説明
  様々なタイプのジョブがあります。ジョブ・インスタンスは、ジョブの特定の実行を表します。ジョブの引数を指定して、デフォルトのプログラムの引数値をオーバーライドできます。
- スケジューラのアーキテクチャ
  スケジューラ・コンポーネントはジョブを処理します。
- スケジューラによるOracle Data Guardのサポート
  Oracle Database 11g リリース1 (11.1)からは、データベースがプライマリ・データベースかロジカル・スタンバイ・データベースかに基づいて、スケジューラがOracle Data Guard環境でジョブを実行できるようになりました。

親トピック: データベース・リソースの管理とタスクのスケジューリング

28.1 Oracle Schedulerの概要

数百または数千ものタスクのスケジューリングを簡素化するために、Oracle Databaseにはエンタープライズ・ジョブ・スケジューラであるOracle Schedulerが組み込まれています。Oracle Scheduler(スケジューラ)は、DBMS_SCHEDULER PL/SQLパッケージのプロシージャおよびファンクションにより実装されます。

スケジューラを使用すると、企業環境で様々なコンピューティング・タスクをいつどこで実施するかを制御できます。これらのタスクの効率的な管理および計画に役立ちます。多数の日常的なコンピューティング・タスクが手動で操作することなく確実に実行されるため、操作コストの削減、信頼性の高いルーチンの実現、人為的なエラーの最小化および必要期間の短縮が可能です。

スケジューラでは、洗練された柔軟なエンタープライズ・スケジューリング機能が提供されます。次のことを実行できます。

- データベース・プログラム・ユニットの実行

  ローカル・データベースまたは1つ以上のリモートOracle Databaseで、プログラム・ユニット(PL/SQL無名ブロック、PL/SQLストアド・プロシージャおよびJavaストアド・プロシージャ)を実行できます。

- 外部実行可能ファイル(データベースの外部の実行可能ファイル)の実行

  アプリケーション、シェル・スクリプト、バッチ・ファイルなどの外部実行可能ファイルをローカル・システムまたは1つ以上のリモート・システムで実行できます。リモート・システムにOracle Databaseをインストールする必要はありません。スケジューラ・エージェントのみが必要です。スケジューラ・エージェントは、Oracle Databaseがサポートするすべてのプラットフォームおよび他の一部のプラットフォームで使用できます。

- 次の方法によるジョブ実行のスケジュール。

  - 時間ベースのスケジューリング

    特定の日時に1回または繰り返し実行するようにジョブをスケジュールできます。「特定の休日を除き毎週月

曜日と木曜日の午前3時」または「業務上の各四半期の最終水曜日」のように、複雑な繰返し間隔を定義できます。詳細は、「ジョブの作成、実行および管理」を参照してください。

- イベント・ベースのスケジューリング

  システム・イベントまたはビジネス・イベントに対応してジョブを開始できます。アプリケーションは、イベントを検出するとスケジューラに通知します。送信された通知のタイプに応じて、スケジューラは特定のジョブを開始します。イベントベースのスケジューリングの例として、ファイルがシステムに着信した時点、在庫が事前に指定したレベルを下回った時点、またはトランザクションに失敗した時点でジョブを開始させる場合があります。Oracle Database 11g リリース2 (11.2)以降では、File Watcherと呼ばれるスケジューラ・オブジェクトによって、ファイルがリモート・システムに到着したときに開始するジョブの構成タスクが簡略化されます。詳細は、「イベントを使用したジョブの開始」を参照してください。

- 依存性スケジューリング

  前の1つ以上のタスクの結果に基づいてタスクを実行するようにスケジューラを設定できます。ブランチやネストしたチェーンを含んだ複雑な依存性チェーンを定義できます。詳細は、「ジョブ・チェーンの作成と管理」を参照してください。

- ビジネス要件に基づくジョブの優先度付け。

  スケジューラを使用すると、競合するジョブ間のリソース割当てを制御でき、ビジネス・ニーズに基づいてジョブ処理を調整できます。これは次の方法で実現されます。

  - ジョブ・クラスによるリソースの制御

    共通の特性や動作を共有するジョブは、ジョブ・クラスと呼ばれるさらに大きなエンティティにグループ化できます。クラス間に優先度を付けるには、各クラスに割り当てられるリソースを制御します。そのため、重要なジョブが優先され、ジョブの完了に必要なリソースを確保できます。たとえば、データ・ウェアハウスをロードする重要なプロジェクトがある場合は、複数のデータ・ウェアハウス・ジョブをすべて1つのクラスにまとめ、使用可能なリソースの割当て比率を高くすることによって、その他のジョブより優先的に実行できます。また、ジョブ・クラス内のジョブに相対的な優先度を割り当てることもできます。

  - スケジュールに基づいたジョブの優先度の制御

    スケジュールに基づいてジョブの優先度を変更できます。重要なジョブの定義は時間の経過とともに変化する場合があるため、スケジューラでは、期間ごとにジョブ間での優先度を変更することもできます。たとえば、データ・ウェアハウスのロードに使用される抽出、転送およびロード(ETL)のジョブが、オフピーク時に重要になり、ピーク時には重要にならない場合があります。また、業務上の四半期の終了時期に実行する必要があるジョブは、ETLジョブよりも優先させることが必要になる場合があります。このような場合は、各ジョブ・クラスに割り当てるリソースを変更して、クラス間での優先度を変更できます。詳細は、「ジョブ・クラスの作成」および「ウィンドウの作成」を参照してください。

- ジョブの管理と監視

  作成から完了までの間にジョブが通過する複数の状態を管理および監視できます。スケジューラはアクティビティをログに記録して、Oracle Enterprise Manager Cloud ControlまたはSQLを使用してビューを問い合せることで、ジョブのステータスやジョブの最終実行時刻などの情報を簡単に追跡できるようにします。このビューからジョブとその実行内容に関する有益な情報が得られるため、これを使用してジョブをより適切にスケジュールし管理できます。たとえば、DBAは特定のユーザーの失敗ジョブをすべて簡単に追跡できます。「スケジューラのデータ・ディクショナリ・ビュー」を参照してください。

複数の宛先のジョブ(1つのデータベースで定義され、複数のリモート・ホストで実行されるジョブ)を作成する場合、宛先ごとにジョブの状態を個別に監視するか、または親ジョブの状態を全体的に監視できます。

ジョブを詳細に監視するために、スケジューラがイベント・キューに提供するジョブの状態変更通知をアプリケーションによってサブスクライブできます。また、スケジューラでは、ジョブの状態が変化したとき、電子メール通知を送信することもできます。

「スケジューラの監視と管理」を参照してください。

- クラスタ化された環境におけるジョブの実行と管理

  クラスタは、同じタスクを実行するために連携して動作するデータベース・インスタンスのセットです。Oracle Real Application Clusters(Oracle RAC)は、アプリケーションを変更せずにスケーラビリティと信頼性を提供します。スケジューラは、このようなクラスタ化された環境でのジョブの実行を完全にサポートします。システムの負荷を均等にし、パフォーマンスを向上させるために、ジョブを実行するデータベース・サービスを指定することもできます。詳細は、「スケジューラとReal Application Clusters」を参照してください。

親トピック: Oracle Schedulerの概要

28.2 ジョブおよびスケジューラ・オブジェクトのサポート

タスク・スケジューリングのために、ジョブおよび他のスケジューラ・オブジェクトを使用します。

- ジョブおよびスケジューラ・オブジェクトのサポートについて
  スケジューラを使用するには、スケジューラ・オブジェクトを作成します。スキーマ・オブジェクトには、ジョブ・スケジューリングの内容、時期および場所を定義します。スケジューラ・オブジェクトにより、モジュール化された方法でタスクを管理できます。この方法のメリットの1つは、既存のタスクに類似する新規タスクを作成する際にオブジェクトを再利用できることです。
- プログラム
  プログラム・オブジェクト(プログラム)では、スケジューラによって実行される内容が記述されます。
- スケジュール
  スケジュール・オブジェクト(スケジュール)は、ジョブの実行時期と実行回数を指定します。
- ジョブ
  ジョブは、ユーザーが定義したタスクを表します。
- 宛先
  ジョブを実行するための外部およびデータベース宛先を指定できます。
- File Watcher
  あるファイルがシステムに到着するとスケジューラによってジョブが開始されるようにする場合、File Watcherオブジェクト(File Watcher)にはそのファイルの場所、名前および他のプロパティを定義します。
- 資格証明
  資格証明は、専用のデータベース・オブジェクトに格納されるユーザー名とパスワードのペアです。
- チェーン
  チェーンは、前の1つ以上のジョブの結果に応じて異なるジョブが開始される依存性スケジューリングを実装できる手段です。
- ジョブ・クラス
  ジョブ・クラスにより、メンバー・ジョブへの同じ属性の割当て、メンバー・ジョブ用のリソース割当ての設定、優先順位のためのジョブのグループ化が可能になります。
- ウィンドウ
  ウィンドウは、ジョブを実行する時間の間隔です。

- グループ
  グループは、スケジューラ・オブジェクトのリストを指定します。
- 非互換性
  非互換性定義(または非互換性)は、互換性のないジョブまたはプログラムを指定し、該当する場合は一度に1つのグループのみを実行できるようにします。

親トピック: Oracle Schedulerの概要

28.2.1 ジョブおよびスケジューラ・オブジェクトのサポートについて

スケジューラを使用するには、スケジューラ・オブジェクトを作成します。スキーマ・オブジェクトには、ジョブ・スケジューリングの内容、時期および場所を定義します。スケジューラ・オブジェクトにより、モジュール化された方法でタスクを管理できます。この方法のメリットの1つは、既存のタスクに類似する新規タスクを作成する際にオブジェクトを再利用できることです。

重要なスケジューラ・オブジェクトはジョブです。ジョブは、実行する処理、処理のスケジュール、処理が行われる場所を定義します。他のほとんどのスケジューラ・オブジェクトは、ジョブをサポートするために作成されます。

> ノート:
>
> DBMS_JOB パッケージは Oracle Scheduler ジョブで置き換えられますが、下位互換性を保つために引き続きサポートされます。この章では、スケジューラ・ジョブのみを使用しているものと仮定します。一度に両方を使用しているか、DBMS_JOB からスケジューラ・ジョブに移行している場合は、「DBMS_JOB のサポート」を参照してください。

これらの各オブジェクトの詳細は、後で説明します。

スケジューラ・オブジェクトはスキーマに属しているため、オブジェクト権限を付与できます。ジョブ・クラス、ウィンドウおよびウィンドウ・グループのような一部のスケジューラ・オブジェクトは、ユーザーがSYSでない場合も常にSYSスキーマに作成されます。他のオブジェクトはすべて、ユーザー自身のスキーマまたは指定のスキーマに作成されます。

関連項目:

「スケジューラ権限」

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.2 プログラム

プログラム・オブジェクト(プログラム)では、スケジューラによって実行される内容が記述されます。

プログラムの要素は次のとおりです。

- 処理: ストアド・プロシージャ名、オペレーティング・システムのファイル・システムにある実行可能ファイル(外部実行可能ファイル)の名前、PL/SQL無名ブロックのテキストなど。
- タイプ: STORED_PROCEDURE、PLSQL_BLOCK、SQL_SCRIPT、EXTERNAL_SCRIPT、BACKUP_SCRIPTまたはEXECUTABLEがあり、EXECUTABLEは外部実行可能ファイルを示します。
- 引数の数: ストアド・プロシージャまたは外部実行可能ファイルが受け入れる引数の数。

プログラムはジョブとは別のエンティティです。ジョブは特定の時刻または特定のイベントの発生により実行されて、特定のプログラ

ムを起動します。既存のプログラム・オブジェクトを指し示すジョブを作成できるため、様々なジョブが同じプログラムを使用して、別の時刻に別の設定でプログラムを実行できます。正しい権限があると、様々なユーザーが同じプログラムを再定義せずに使用できます。そのため、既存のプログラムのリストからユーザーが選択できるプログラム・ライブラリを作成できます。

プログラムで参照されるストアド・プロシージャまたは外部実行可能ファイルが引数を受け入れる場合は、これらの引数をプログラムの作成後に個別ステップで定義します。各引数のデフォルト値を定義することもできます。

関連項目:

- 「プログラムの作成」
- ジョブの概要については、「ジョブ」

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.3 スケジュール

スケジュール・オブジェクト(スケジュール)は、ジョブの実行時期と実行回数を指定します。

スケジュールは複数のジョブで共有できます。たとえば、業務上の四半期の終了時期は、多くのジョブにとって共通の期間である可能性があります。ジョブの作成者は、新規のジョブを定義するたびに四半期の終了時期のスケジュールを定義するかわりに、名前付きのスケジュールを指し示すことができます。

スケジュールには、次の2種類があります。

- 時間スケジュール

  時間スケジュールでは、ジョブを即時に実行するか、後で実行するようにスケジュールできます。時間スケジュールには、開始日時、オプションの終了日時およびオプションの繰返し間隔が含まれています。

- イベント・スケジュール

  イベント・スケジュールでは、在庫がしきい値を下回った時点、またはファイルがシステムに到着した時点など、特定のイベントが発生した時点でジョブを実行するように指定できます。イベントの詳細は、「イベントを使用したジョブの開始」を参照してください。

関連項目:

「スケジュールの作成」

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.4 ジョブ

ジョブは、ユーザーが定義したタスクを表します。

- ジョブについて
  ジョブ・オブジェクト(ジョブ)は、ユーザー定義のタスクを記述するメタデータのコレクションです。実行される内容(処理)、タイミング(1回のみ、定期的なスケジュールまたはトリガー・イベント)、場所(宛先)および使用される資格証明を定義します。ジョブにはスキーマである所有者が存在し、ジョブはそのスキーマで作成されます。
- ジョブの処理の指定
  ジョブの処理を指定するには、実行するデータベース・プログラム・ユニットまたは外部の実行可能ファイル、または既存

のプログラム・オブジェクト(プログラム)の名前を指定します。

- ジョブ・スケジュールの指定
  ジョブ・スケジュールは、ジョブ・オブジェクトの属性または既存スケジュール・オブジェクト(スケジュール)の名前を設定することによって指定できます。
- ジョブの宛先の指定
  様々な方法でジョブの宛先を指定できます。
- ジョブの資格証明の指定
  名前付き資格証明オブジェクトを指定することによってジョブの資格証明を指定するか、ジョブの資格証明属性をNULLのままにすることを許可します。

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.4.1 ジョブについて

ジョブ・オブジェクト(ジョブ)は、ユーザー定義のタスクを記述するメタデータのコレクションです。実行される内容(処理)、タイミング(1回のみ、定期的なスケジュールまたはトリガー・イベント)、場所(宛先)および使用される資格証明を定義します。ジョブにはスキーマである所有者が存在し、ジョブはそのスキーマで作成されます。

データベース・プログラム・ユニットを実行するジョブをデータベース・ジョブと呼びます。外部実行可能ファイルを実行するジョブを外部ジョブと呼びます。

1つ以上のリモートの場所でデータベース・プログラム・ユニットを実行するジョブをリモート・データベース・ジョブと呼びます。1つ以上のリモートの場所で外部実行可能ファイルを実行するジョブをリモート外部ジョブと呼びます。

1つ以上の宛先を指定して、ジョブが実行される場所を定義します。宛先もスケジューラ・オブジェクトであり、この項で後述します。宛先を指定しないと、ローカル・データベースでジョブが実行されるとみなされます。

親トピック: ジョブ

28.2.4.2 ジョブの処理の指定

ジョブの処理を指定するには、実行するデータベース・プログラム・ユニットまたは外部の実行可能ファイル、または既存のプログラム・オブジェクト(プログラム)の名前を指定します。

ジョブの処理は、次のいずれかの方法で指定します。

- 実行するデータベース・プログラム・ユニットまたは外部実行可能ファイルをジョブ属性として指定する方法。この方法をジョブの処理のインライン指定と呼びます。
- 既存のプログラム名をジョブ属性として指定し、実行するデータベース・プログラム・ユニットまたは外部実行可能ファイルをプログラムで指定する方法。ジョブ所有者には、プログラムに対するEXECUTE権限またはEXECUTE ANY PROGRAMシステム権限が必要です。

親トピック: ジョブ

28.2.4.3 ジョブ・スケジュールの指定

ジョブ・スケジュールは、ジョブ・オブジェクトの属性または既存スケジュール・オブジェクト(スケジュール)の名前を設定することによって指定できます。

ジョブ・スケジュールは、次のいずれかの方法で指定します。

- ジョブ・オブジェクトの属性を設定して開始日時、終了日時および繰返し間隔を定義するか、またはジョブを開始するイベントを定義する方法。この方法をスケジュールのインライン指定と呼びます。

- 開始日時、終了日時および繰返し間隔を定義する、またはイベントを定義する、既存のスケジュールの名前をジョブ属性として指定する方法。

親トピック: ジョブ

28.2.4.4 ジョブの宛先の指定

様々な方法でジョブの宛先を指定できます。

ジョブの宛先は、次のいずれかの方法で指定します。

- 指定された単一の宛先オブジェクトをジョブ属性として指定する方法。この場合、ジョブは1つのリモートの場所で実行されます。
- 指定された宛先グループ(リモートの場所のリストと同じ)をジョブ属性として指定する方法。この場合、ジョブはすべてのリモートの場所で実行されます。
- 宛先属性を定義せずにジョブをローカルに実行する方法。ジョブは次のいずれかを実行します。
    - ローカル・データベースのデータベース・プログラム・ユニット(ジョブが作成されたデータベース)
    - ジョブ処理タイプに応じた、ローカル・ホストの外部実行可能ファイル

親トピック: ジョブ

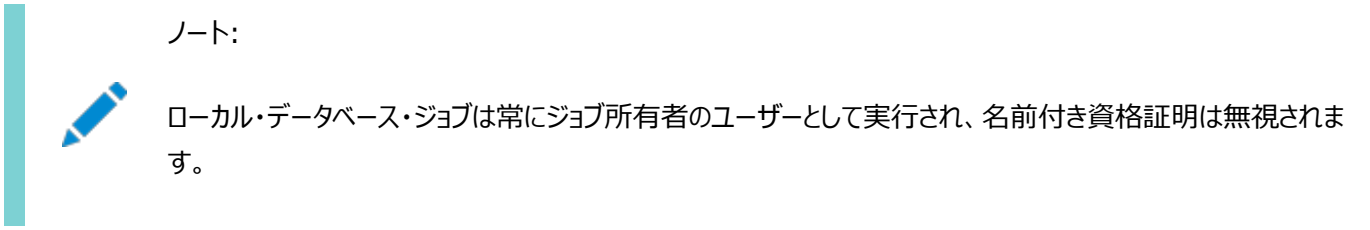
28.2.4.5 ジョブの資格証明の指定

名前付き資格証明オブジェクトを指定することによってジョブの資格証明を指定するか、ジョブの資格証明属性をNULLのままにすることを許可します。

ジョブの資格証明は、次のいずれかの方法で指定します。

- ジョブ属性を名前付き資格証明オブジェクトとして指定する方法で、データベースのユーザー名とパスワードが含まれます(データベース・ジョブの場合)。

  ジョブは、資格証明に指定されているユーザーとして実行されます。
- ジョブの資格証明属性をNULLのままにする方法で、この場合、ローカル・データベース・ジョブはジョブの所有者として実行されます。(表28-1を参照してください。)ジョブの所有者は、ジョブが作成されたスキーマです。



ノート:

ローカル・データベース・ジョブは常にジョブ所有者のユーザーとして実行され、名前付き資格証明は無視されます。

ジョブを作成して有効化した後は、スケジューラによりスケジュールに従って自動的にジョブが実行されるか、または指定のイベントが検出されたときに実行されます。ジョブの実行ステータスとジョブ・ログは、データ・ディクショナリ・ビューを問い合せることで確認できます。ジョブが複数の宛先で実行される場合は、各宛先でジョブの状態を問い合せることができます。

関連項目:

- 「宛先」
- 「ジョブに関する追加説明」

- 「ジョブの作成」
- 「スケジューラのデータ・ディクショナリ・ビュー」

親トピック: ジョブ

28.2.5 宛先

ジョブを実行するための外部およびデータベース宛先を指定できます。

- 宛先について
  宛先オブジェクト(宛先)は、ジョブを実行する場所を定義します。
- 宛先およびスケジューラ・エージェントについて
  宛先オブジェクトで指定されたリモートの場所でスケジューラ・エージェントが実行されており、かつ、ジョブを作成するデータベースにそのエージェントが登録されている必要があります。

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.5.1 宛先について

宛先オブジェクト(宛先)は、ジョブを実行する場所を定義します。

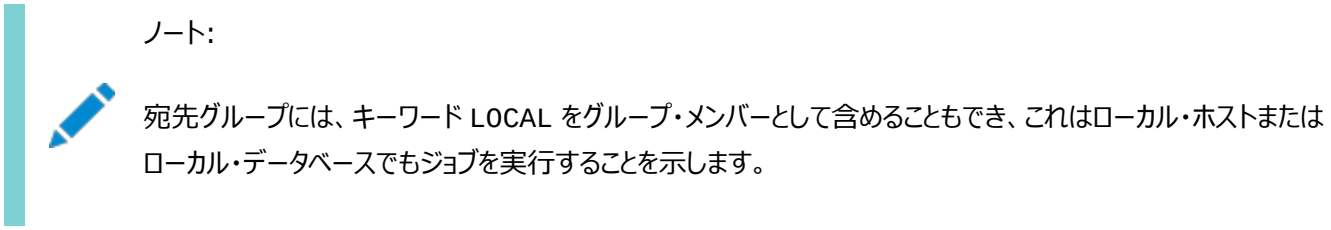
宛先には、次の2種類があります。

- 外部宛先: リモート外部ジョブを実行するリモート・ホストの名前とIPアドレスを指定します。
- データベース宛先: リモート・データベース・ジョブを実行するリモート・データベース・インスタンスを指定します。

外部実行可能ファイルを実行するジョブ(外部ジョブ)では外部宛先を指定する必要があり、データベース・プログラム・ユニットを実行するジョブ(データベース・ジョブ)ではデータベース宛先を指定する必要があります。

ジョブの作成時に宛先を指定すると、その宛先でジョブが実行されます。宛先を指定しないと、ジョブが作成されたシステムでローカルに実行されます。

宛先のリストで構成される宛先グループを作成して、この宛先グループをジョブの作成時に参照することもできます。この場合、ジョブはグループのすべての宛先で実行されます。



ノート:

宛先グループには、キーワード LOCAL をグループ・メンバーとして含めることもでき、これはローカル・ホストまたはローカル・データベースでもジョブを実行することを示します。

関連項目:

「グループ」

別のユーザーによって作成された宛先を使用する場合に、オブジェクト権限は必要ありません。

親トピック: 宛先

28.2.5.2 宛先およびスケジューラ・エージェントについて

宛先オブジェクトで指定されたリモートの場所でスケジューラ・エージェントが実行されており、かつ、ジョブを作成するデータベース

にそのエージェントが登録されている必要があります。

スケジューラ・エージェントを使用すると、ローカル・スケジューラとリモート・ホストとの通信や、リモート・ホストでのジョブの開始と停止が可能になり、リモート・ジョブの状態をローカル・データベースに戻すことができます。詳細は、「宛先の指定」を参照してください。

- 外部宛先
  外部宛先を明示的に作成することはできません。外部宛先は、データベースにスケジューラ・エージェントを登録するとき、ローカル・データベースで作成されます。
- データベース宛先
  データベース宛先は、DBMS_SCHEDULER.CREATE_DATABASE_DESTINATIONプロシージャを使用して作成します。

親トピック: 宛先

28.2.5.2.1 外部宛先

外部宛先を明示的に作成することはできません。外部宛先は、データベースにスケジューラ・エージェントを登録するとき、ローカル・データベースで作成されます。

外部宛先に割り当てられる名前は、エージェント名と同じです。エージェント名はエージェントのインストール後に構成できますが、ホスト名の最初の部分(最初のドット区切りの前)であるデフォルトのエージェント名も使用できます。たとえば、dbhost1.us.example.comというホストにエージェントをインストールした場合、デフォルトのエージェント名はDBHOST1になります。

親トピック: 宛先およびスケジューラ・エージェントについて

28.2.5.2.2 データベース宛先

データベース宛先は、DBMS_SCHEDULER.CREATE_DATABASE_DESTINATIONプロシージャを使用して作成します。

> ノート:
>
> ローカル・ホストで複数のデータベース・インスタンスが実行されている場合は、他のインスタンスを指定してデータベース宛先を作成することによって、そのインスタンスでジョブを実行できます。したがって、リモート・データベース・インスタンスは、必ずしもリモート・ホストに存在している必要はありません。このような追加のインスタンスでリモート・データベース・ジョブを実行できるようにするには、ローカル・ホストでスケジューラ・エージェントが実行されている必要があります。

関連項目:

- 「宛先の指定」
- 「リモート・ホストでのスケジューラ・エージェントのインストールと構成」

親トピック: 宛先およびスケジューラ・エージェントについて

28.2.6 File Watcher

あるファイルがシステムに到着するとスケジューラによってジョブが開始されるようにする場合、File Watcherオブジェクト(File Watcher)にはそのファイルの場所、名前および他のプロパティを定義します。

ファイル・ウォッチャを作成し、次に、そのファイル・ウォッチャを参照する任意の数のイベントベースのジョブまたはイベント・スケジュールを作成します。ファイル・ウォッチャでは、指定されたファイルの到着を検出すると、到着イベントを呼び出します。ファイルの到着イベントによって開始されたジョブでは、新しく到着したファイルを示すイベント・メッセージを取得できます。

File Watcherでは、ローカル・システム(Oracle Databaseを実行しているホスト・コンピュータと同じ)またはリモート・システム(リモート・システムでスケジューラ・エージェントが稼働している場合)でファイルが監視されます。

File Watcherを使用するには、データベースのJava仮想マシン(JVM)コンポーネントをインストールする必要があります。

詳細は、「File Watcherについて」を参照してください。

関連項目:

「File WatcherおよびFile Watcherジョブの作成」

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.7 資格証明

資格証明は、専用のデータベース・オブジェクトに格納されるユーザー名とパスワードのペアです。

スケジューラ・ジョブは、実行されるときに資格証明を使用してデータベース・インスタンスまたはオペレーティング・システムで認証を受けます。資格証明には、次の用途があります。

- リモート・データベース・ジョブ: 資格証明には、データベースのユーザー名とパスワードが含まれています。リモート・データベース・ジョブで指定されるストアド・プロシージャまたはPL/SQLブロックは、このデータベース・ユーザーとして実行されます。
- 外部ジョブ(ローカルまたはリモート): 資格証明には、ホスト・オペレーティング・システムのユーザー名とパスワードが含まれています。ジョブの外部実行可能ファイルは、このユーザー名とパスワードを使用して実行されます。
- File Watcher: 資格証明には、ホスト・オペレーティング・システムのユーザー名とパスワードが含まれています。ファイルの到着イベントを処理するジョブは、このユーザー名とパスワードを使用して、到着したファイルにアクセスします。

データベース内の資格証明のリストを表示するには、`*_CREDENTIALS`ビューを問い合せます。資格証明パスワードは不明瞭化されて格納されるため、これらのビューには表示されません。

関連項目:

- 「スケジューラ・ジョブの資格証明の指定」
- `DBMS_CREDENTIAL.CREATE_CREDENTIAL`プロシージャを使用した資格証明の作成の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.8 チェーン

チェーンは、前の1つ以上のジョブの結果に応じて異なるジョブが開始される依存性スケジューリングを実装できる手段です。

チェーンは、依存性ルールを使用して結合された複数のステップで構成されています。依存性ルールでは、ステップまたはチェーン自体の開始または停止に使用できる条件を定義します。条件には、前のステップの成功、失敗、完了コードまたは終了コード

を使用できます。AND/ORなどの論理式を条件に使用することもできます。チェーンは、実行するタスクと、タスクの実行時期の選択に関して多くの可能なパスを持ち、ある意味でDecision Treeに似ています。

最も単純な形式のチェーンは、1つの目的のために互いにリンクされた複数のスケジューラ・プログラム・オブジェクト(プログラム)で構成されています。チェーンの例には、「プログラムAを実行してからプログラムBを実行するが、プログラムAとプログラムBの両方が正常に完了した場合にのみプログラムCを実行し、正常に完了しない場合は1時間待機してからプログラムDを実行する」などがあります。

チェーンの作成が必要な例として、融資申請を検証して承認した後に融資するなど、正常な取引のために様々なプログラムの結合を必要とする金融取引などがあります。

スケジューラ・ジョブは、単一のプログラム・オブジェクトを指し示すかわりにチェーンを指し示すことができます。その場合、ジョブはチェーンを開始する役割を持ちます。このジョブはチェーン・ジョブと呼ばれます。複数のチェーン・ジョブで同じチェーンを指し示すことができ、そのうちの複数のジョブを同時に実行することで、チェーンの様々な進捗ポイントで同じチェーンの複数インスタンスをそれぞれ作成できます。

チェーン内の各位置はステップと呼ばれます。通常は、最初の一連のチェーン・ステップを開始し、後続のステップは、1つ以上前のステップの完了に従って実行されます。各ステップは、次のいずれかを指し示します。

- プログラム・オブジェクト(プログラム)

  プログラムでは、データベース・プログラム・ユニット(ストアド・プロシージャまたはPL/SQL無名ブロックなど)または外部実行可能ファイルを実行できます。

- 別のチェーン(ネストしたチェーン)

  チェーンをネストするレベル数に制限はありません。

- イベント・スケジュール、インライン・イベントまたはFile Watcher

  イベント・スケジュールを指し示すステップ、またはインライン・イベント指定を持つステップを開始した後、ステップは特定のイベントが呼び出されるまで待機します。同様に、File Watcherインラインを参照するステップ、またはFile Watcherを参照するイベント・スケジュールを指し示すステップは、ファイルの到着イベントが呼び出されるまで待機します。ファイルの到着イベントまたは他のタイプのイベントの場合、イベントが発生すると、このステップが完了し、イベント・ステップに依存するステップが実行可能になります。チェーン内のイベントの一般例は、承認や拒否のようなユーザーの介入です。

チェーン内の複数のステップは、同じプログラムまたはネストしたチェーンを起動できます。

各ステップには、ステップが実行されるデータベース宛先または外部宛先のいずれかを指定できます。宛先を指定しないと、ステップが元の(ローカル)データベースまたはローカル・ホストで実行されます。チェーン内の各ステップは、異なる宛先で実行できます。

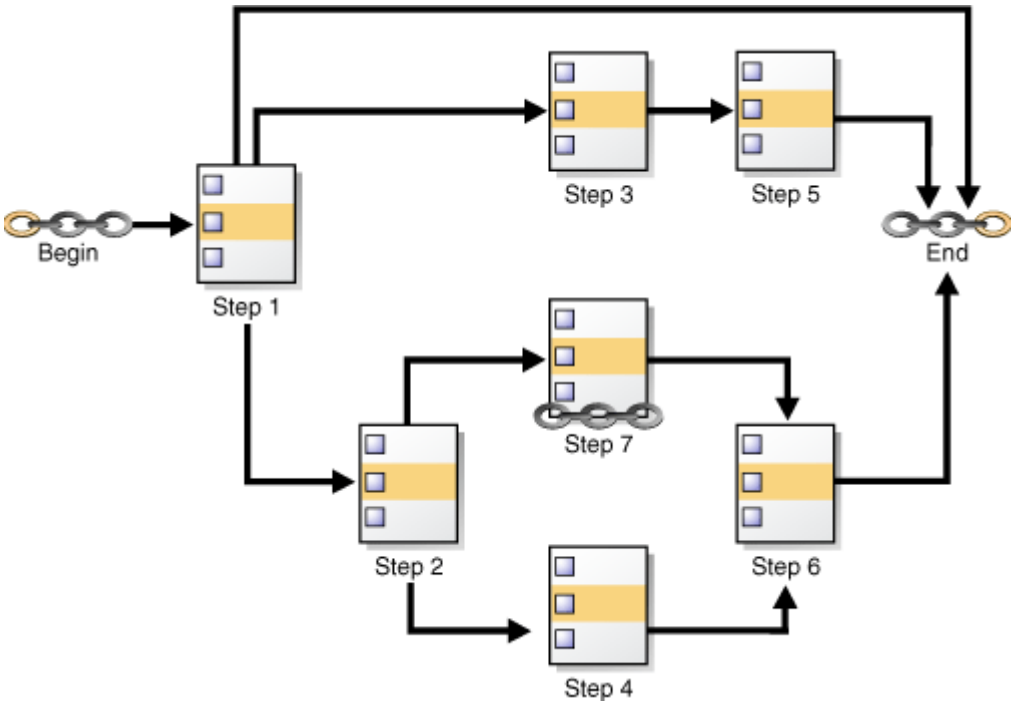
図28-1に、複数のブランチを持つチェーンを示します。この図では、アイコンを使用してBEGIN、ENDおよび下側のサブブランチ内のネストしたチェーン(ステップ7)を示しています。

この図では、ルールが次のように定義されているとします。

- ステップ1が正常に完了した場合は、ステップ2を開始します。
- ステップ1が失敗してエラー・コード20100が戻された場合は、ステップ3を開始します。
- ステップ1が失敗して他のエラー・コードが戻された場合は、チェーンを終了します。

ステップ4、5、6および7の実行は、他のルールにより制御されます。

図28-1 複数のブランチを持つチェーン



チェーンを指し示しているジョブの実行中は、実行しているチェーンのすべてのステップについて現在の状態を監視できます。各ステップには、スケジューラによって、チェーン・ジョブと同じジョブ名と所有者を設定したステップ・ジョブが作成されます。各ステップ・ジョブにはさらに、そのジョブを一意に識別するためのサブ名があります。ステップ・ジョブ・サブ名は、ビュー*_SCHEDULER_RUNNING_JOBS、*_SCHEDULER_JOB_LOGおよび*_SCHEDULER_JOB_RUN_DETAILSにJOB_SUBNAME列として、*_SCHEDULER_RUNNING_CHAINSビューにSTEP_JOB_SUBNAME列として含まれています。

関連項目:

「ジョブ・チェーンの作成と管理」

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.9 ジョブ・クラス

ジョブ・クラスにより、メンバー・ジョブへの同じ属性の割当て、メンバー・ジョブ用のリソース割当ての設定、優先順位のためのジョブのグループ化が可能になります。

通常、ジョブ・クラスを作成するのは、スケジューラ管理者のロールを持っている場合のみです。

ジョブ・クラスは、次の内容を実行します。

- メンバーのジョブへの同じ属性値セットの割当て

  各ジョブ・クラスでは、ロギング・レベルなどの属性セットを指定します。ジョブをジョブ・クラスに割り当てると、そのジョブはこれらの属性を継承します。たとえば、全給与ジョブに対するログ・エントリのパージに関して、同じポリシーを指定できます。

- メンバーのジョブに対するサービス・アフィニティの設定

  ジョブ・クラスのservice属性を、目的のデータベース・サービス名に設定できます。これにより、メンバー・ジョブを実行するReal Application Clusters環境のインスタンスを決定でき、必要に応じて、メンバー・ジョブに割り当てられているシステム・リソースも決定できます。詳細は、「スケジューラ使用時のサービス・アフィニティ」を参照してください。

- メンバーのジョブに対するリソース割当ての設定

各ジョブ・クラスでは、リソース・コンシューマ・グループを属性として指定できるため、ジョブ・クラスでは、データベース・リソース・マネージャとスケジューラ間のリンクが提供されます。このため、指定したコンシューマ・グループに属するメンバーのジョブには、現行のリソース・プランの設定に応じて、リソースが割り当てられます。

また、resource_consumer_group属性をNULLのままにし、ジョブ・クラスのservice属性を必要なデータベース・サービス名に設定できます。このサービスは、リソース・コンシューマ・グループにマップできます。resource_consumer_group属性とservice属性が設定されているときに、指定のサービスがリソース・コンシューマ・グループにマップされた場合は、resource_consumer_group属性に指定されているリソース・コンシューマ・グループが優先されます。

コンシューマ・グループに対するサービスのマッピングの詳細は、「Oracle Database Resource Managerを使用したリソースの管理」を参照してください。

- ジョブの優先度によるグループ化

  同じジョブ・クラス内では、個々のジョブに対して1から5の優先度値を割り当てて、クラス内の2つのジョブが同時に起動するようにスケジュールされた場合に、優先度の高いジョブが優先されるように設定できます。こうすることで、重要度の高いジョブが適切なタイミングで完了するのを、重要度の低いジョブが妨げないようにできます。

  2つのジョブに同じ優先値が割り当てられている場合は、開始日の早いジョブが優先されます。ジョブに優先度が割り当てられていない場合、そのジョブの優先度は3にデフォルト設定されます。

> ノート:
>
> ジョブの優先度は、同じクラス内のジョブの間で優先度を付ける場合にのみ使用されます。
>
> 同じスケジュールを共有している場合でも、クラス A 内の優先度の高いジョブが、クラス B 内の優先度の低いジョブより先に開始されることは保証されません。異なるクラスのジョブ間の優先度付けは、現行のリソース・プランと、指定されたリソース・コンシューマ・グループまたは各ジョブ・クラスのサービス名に応じて異なります。

ジョブ・クラスを定義するときは、ジョブを機能別に分類してください。マーケティング、生産、販売、財務および人事など、同様のデータにアクセスするジョブをグループに分けることを考慮してください。

次の制限事項に注意してください。

- ジョブは必ず1つのクラスに属している必要があります。ジョブを作成するときは、そのジョブが属するクラスを指定できます。クラスを指定しない場合、ジョブは自動的にDEFAULT_JOB_CLASSクラスのメンバーになります。
- クラス内にジョブが存在している状態でクラスを削除するとエラーになります。クラスのメンバーであるジョブがまだ存在している場合でもクラスを強制的に削除することはできますが、そのクラスを参照しているジョブはすべて自動的に使用禁止になり、DEFAULT_JOB_CLASSクラスに割り当てられます。削除したクラスに属しているジョブの中ですでに実行中のジョブは、ジョブの開始時に判別されたクラスの設定のまま実行されます。

関連項目:

- 「ジョブ・クラスの作成」
- ジョブ・クラスを表示するには、『Oracle Databaseリファレンス』

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.10 ウィンドウ

ウィンドウは、ジョブを実行する時間の間隔です。

- ウィンドウについて
  日または週などの様々な期間中に、ジョブを自動的に開始したり、複数のジョブ間のリソース割当てを変更するには、ウィンドウを作成します。ウィンドウは、午前12時から午前6時など、開始と終了が明確に定義された期間で表されます。
- ウィンドウの重複
  お薦めする方法ではありませんが、ウィンドウは重複して設定できます。

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.10.1 ウィンドウについて

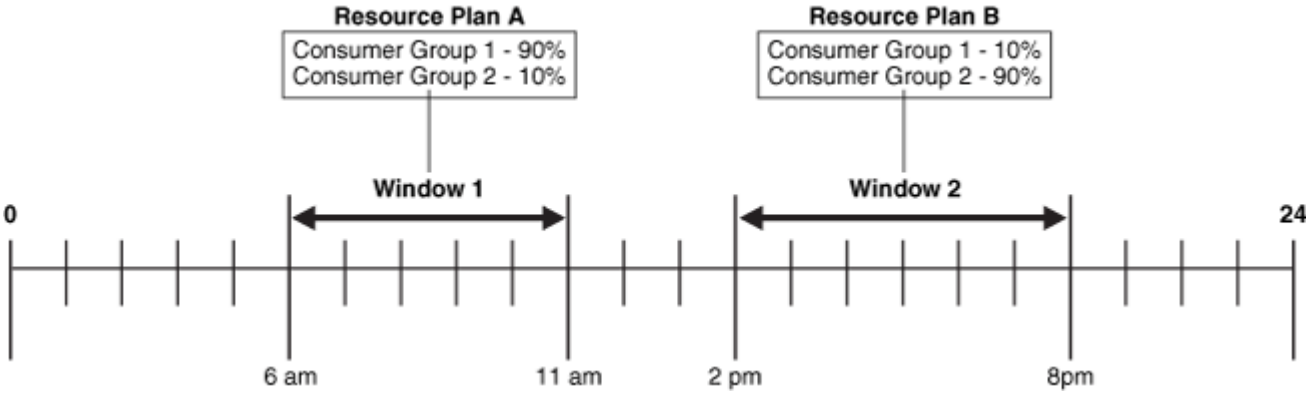
日または週などの様々な期間中に、ジョブを自動的に開始したり、複数のジョブ間のリソース割当てを変更するには、ウィンドウを作成します。ウィンドウは、午前12時から午前6時など、開始と終了が明確に定義された期間で表されます。

通常、ウィンドウを作成するのは、スケジューラ管理者のロールを持っている場合のみです。

ウィンドウは、ジョブ・クラスを使用して、リソース割当てを制御します。各ウィンドウは、ウィンドウがオープンしたとき(アクティブになったとき)に、アクティブにするリソース・プランを指定し、各ジョブ・クラスは、リソース・コンシューマ・グループを指定するか、コンシューマ・グループにマップできるデータベース・サービスを指定します。したがって、ウィンドウ内で実行するジョブには、そのジョブ・クラスのコンシューマ・グループとウィンドウのリソース・プランに応じてリソースが割り当てられます。

図28-2に、2つのウィンドウが設定された稼働日を示します。この構成では、コンシューマ・グループ1にリンクしているジョブ・クラスに属するジョブが、午後よりも午前にリソースを多く使用しています。コンシューマ・グループ2にリンクしているジョブ・クラスのジョブは、この逆です。

図28-2 ジョブに割り当てるリソースの定義に役立つウィンドウ



リソース・プランおよびコンシューマ・グループの詳細は、「Oracle Database Resource Managerを使用したリソース割当ての管理」を参照してください。

各ウィンドウには、優先度を割り当てることができます。ウィンドウが重複する場合は、優先度の高いウィンドウが優先度の低いウィンドウより優先して選択されます。スケジューラは、ウィンドウの開始時間と終了時間に従って、各ウィンドウを自動的にオープンしたり、クローズします。

ジョブは、そのschedule_name属性でウィンドウを指定できます。スケジューラは、このウィンドウがオープンするとジョブを開始します。ウィンドウがすでにオープンしている場合にそのウィンドウを指し示す新規ジョブが作成されると、そのジョブはウィンドウが次回オープンするまで開始されません。

ノート:

必要な場合は、現行のリソース・プランが切り替わらないように一時的にウィンドウをブロックできます。詳細は、「Oracle Database Resource Manager の有効化とプランの切替え」または『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』の DBMS_RESOURCE_MANAGER.SWITCH_PLAN パッケージ・プロシージャに関する項を参照してください。

関連項目:

「ウィンドウの作成」

親トピック: ウィンドウ

28.2.10.2 ウィンドウの重複

お薦めする方法ではありませんが、ウィンドウは重複して設定できます。

一度にアクティブにできるウィンドウは1つのみであるため、ウィンドウが重複したときにどのウィンドウをアクティブにするかを判別するために、次のルールが使用されます。

- 同じ優先度のウィンドウが重複している場合は、アクティブなウィンドウがオープンしたままになります。ただし、優先度の高いウィンドウと重複している場合は、優先度の低いウィンドウがクローズし、優先度の高いウィンドウがオープンします。優先度の低いウィンドウを指定しているスケジュールを持つ現在実行中のジョブは、ジョブの作成時に割り当てた動作に従って停止される場合があります。
- ウィンドウの終了時に、定義されているウィンドウが複数ある場合は、優先度の最も高いウィンドウがオープンします。すべてのウィンドウの優先度が同じ場合は、残りの時間の比率の最も高いウィンドウがオープンします。
- 削除対象のオープン中のウィンドウは、自動的にクローズします。この時点で前述のルールが適用されます。

2つのウィンドウが重複した場合は、スケジューラのログにエントリが必ず記録されます。

- ウィンドウの重複例
  重複するウィンドウの例を示します。

親トピック: ウィンドウ

28.2.10.2.1 ウィンドウの重複例

重複するウィンドウの例を示します。

図28-3に、24時間スケジュールの場合のウィンドウ、リソース・プランおよび優先度の判別方法に関する一般的な例を示します。次の2つの例では、ウィンドウ1はリソース・プラン1に、ウィンドウ2はリソース・プラン2に関連付けられ、以下も同様に関連付けられているとします。

図28-3 ウィンドウとリソース・プラン(例1)

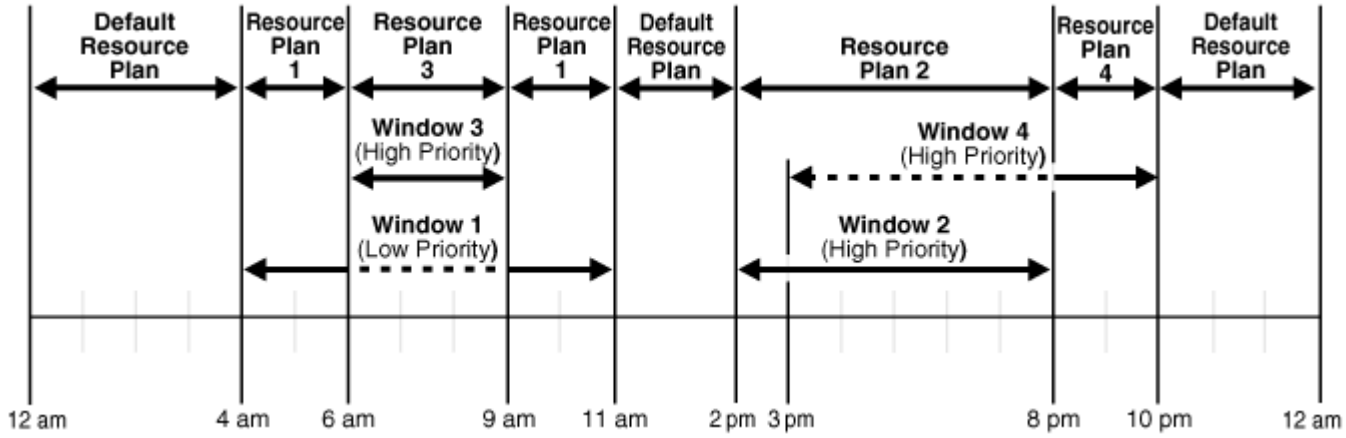


図28-3では、次の処理が発生します。

- 午前12時から午前4時

  いずれのウィンドウもオープンしていないため、デフォルトのリソース・プランが有効です。

- 午前4時から午前6時

  ウィンドウ1には低い優先度が割り当てられていますが、高い優先度のウィンドウが他に存在しないためウィンドウ1がオープンします。したがって、リソース・プラン1が有効です。

- 午前6時から午前9時

  ウィンドウ1より優先度の高いウィンドウ3がオープンします。したがって、リソース・プラン3が有効です。点線は、Window1が非アクティブであることを示しています。

- 午前9時から午前11時

  優先度が高いウィンドウがオープンしたために、ウィンドウ1は午前6時にクローズされましたが、この優先度の高いウィンドウが午前9時にクローズしたため、ウィンドウ1の元のスケジュールの2時間がウィンドウ1にまだ残っています。この残り2時間のためにそれが再オープンされ、リソース・プランが有効になります。

- 午前11時から午後2時

  いずれのウィンドウもオープンしていないため、デフォルトのリソース・プランが有効です。

- 午後2時から午後3時

  ウィンドウ2がオープンするため、リソース・プラン2が有効です。

- 午後3時から午後8時

  ウィンドウ4の優先度はウィンドウ2と同じであるため、ウィンドウ2への割込みはなく、リソース・プラン2が有効です。点線は、Window4が非アクティブであることを示しています。

- 午後8時から午後10時

  ウィンドウ4がオープンしているため、リソース・プラン4が有効です。

- 午後10時から午前12時

  いずれのウィンドウもオープンしていないため、デフォルトのリソース・プランが有効です。

図28-4に、24時間スケジュールの場合のウィンドウ、リソース・プランおよび優先度の判別方法に関する別の例を示します。

図28-4 ウィンドウとリソース・プラン(例2)

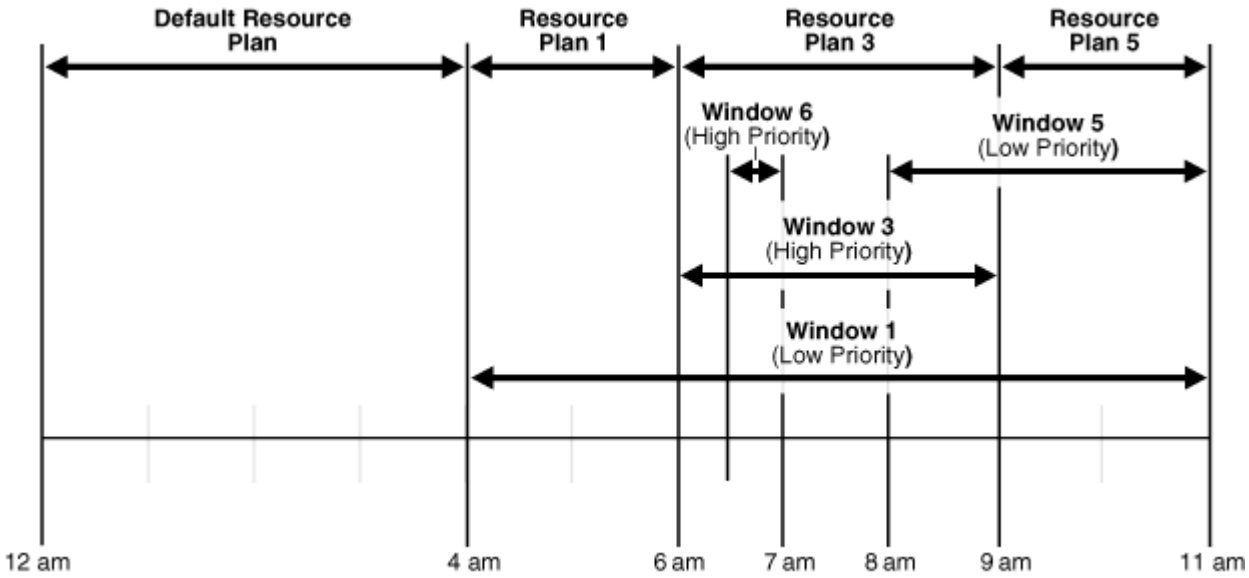


図28-4では、次の処理が発生します。

- 午前12時から午前4時

  デフォルトのリソース・プランが有効です。

- 午前4時から午前6時

  ウィンドウ1には低い優先度が割り当てられていますが、高い優先度のウィンドウが他に存在しないためウィンドウ1がオープンします。したがって、リソース・プラン1が有効です。

- 午前6時から午前9時

  ウィンドウ1より優先度の高いウィンドウ3がオープンします。ウィンドウ6は優先度の高い別のウィンドウがすでに有効であるためオープンしません。

- 午前9時から午前11時

  午前9時の時点では、ウィンドウ5またはウィンドウ1の2つが選択対象です。両方とも優先度が低いため、残り継続時間の比率の大きさに基づいて選択されます。ウィンドウ1の総継続時間と比較して、ウィンドウ5の残り時間の比率の方が大きい状態です。たとえば、ウィンドウ1が午前11時30分まで延長された場合でも、ウィンドウ5の残り継続時間に対する比率が2/3×100%であるのに対して、ウィンドウ1は2.5/7×100%であるため比率が小さくなります。したがって、リソース・プラン5が有効になります。

親トピック: ウィンドウの重複

28.2.11 グループ

グループは、スケジューラ・オブジェクトのリストを指定します。

- グループについて
  オブジェクトのリストを引数としてDBMS_SCHEDULERパッケージ・プロシージャに渡すかわりに、オブジェクトがメンバーとして含まれるグループを作成した後、グループ名をプロシージャに渡すことができます。
- 宛先グループ
  ジョブを複数の宛先で実行する場合、データベース宛先グループまたは外部宛先グループを作成し、それをジョブの`destination_name`属性に割り当てます。
- ウィンドウ・グループ
  ジョブのスケジュールで、ウィンドウを使いやすいようにグループ化できます。

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.2.11.1 グループについて

オブジェクトのリストを引数としてDBMS_SCHEDULERパッケージ・プロシージャに渡すかわりに、オブジェクトがメンバーとして含まれるグループを作成した後、グループ名をプロシージャに渡すことができます。

グループには、次の3種類があります。

- データベース宛先グループ: リモート・データベース・ジョブを実行するためのデータベース宛先がメンバーです。
- 外部宛先グループ: リモート外部ジョブを実行するための外部宛先がメンバーです。
- ウィンドウ・グループ: スケジューラのウィンドウがメンバーです。

グループ内のすべてのメンバーは同一タイプで、各メンバーが一意であることが必要です。

グループは、DBMS_SCHEDULER.CREATE_GROUPプロシージャを使用して作成します。

親トピック: グループ

28.2.11.2 宛先グループ

ジョブを複数の宛先で実行する場合、データベース宛先グループまたは外部宛先グループを作成し、それをジョブのdestination_name属性に割り当てます。

ジョブに複数の宛先を指定する場合は、宛先グループをジョブのdestination_nameとして指定する必要があります。

親トピック: グループ

28.2.11.3 ウィンドウ・グループ

ジョブのスケジュールで、ウィンドウを使いやすいようにグループ化できます。

通常、ウィンドウ・グループを作成するのは、スケジューラ管理者のロールを持っている場合のみです。

ウィンドウ・グループを使用すると、1日あるいは1週間などの間に複数期間実行するジョブを簡単にスケジュールできます。ジョブのschedule_name属性をこのウィンドウ・グループ名に設定すると、ウィンドウ・グループ内のウィンドウで指定されたすべての期間で、このジョブを実行できます。

たとえば、「Weekends」というウィンドウと「Weeknights」というウィンドウがある場合に、これらの2つのウィンドウを「Downtime」というウィンドウ・グループに追加できます。これにより、使用可能なリソースを高い割合で問合せに割り当てることができる時間帯(平日の夜と週末)のDowntimeウィンドウ・グループに問合せを実行するジョブをデータ・ウェアハウスのスタッフが作成できます。

ウィンドウ・グループのウィンドウがすでにオープンしているときに、そのウィンドウ・グループを指し示す新規ジョブが作成された場合、そのジョブはウィンドウ・グループの次のウィンドウがオープンするまで開始されません。

関連項目:

- 複数の宛先のジョブに対する宛先グループの作成
- 「ウィンドウ・グループの作成」
- 「ウィンドウ」

親トピック: グループ

28.2.12 非互換性

非互換性定義(または非互換性)は、互換性のないジョブまたはプログラムを指定し、該当する場合は一度に1つのグループのみを実行できるようにします。

たとえば、ジョブAおよびBが非互換として定義されていて、それらのジョブ・スケジュールに従うと同時に実行される場合でも、スケジューラは常にそれらのいずれかのみが実行されるようにします。

非互換性はジョブ・レベル(デフォルト)またはプログラム・レベルで定義できます。たとえば、次のような場合を考えてみます。

- ジョブJ1およびJ2はプログラムP1に基づいています。
- ジョブJ3、J4およびJ5はプログラムP2に基づいています。
- ジョブJ6およびJ7はプログラムP3に基づいています。

このシナリオでは、次のようになります。

- ジョブ・レベルの非互換性定義にJ3、J4およびJ5が指定されていて、ジョブJ3が実行されている場合、J4およびJ5はJ3が終了するまで実行できません。
- プログラム・レベルの非互換性定義にP1、P2およびP3が指定されている場合、ジョブJ1およびJ2は同時に実行できますが(ジョブ・レベルの制約によってJ1およびJ2を同時に実行することが防止されていない場合)、プログラムP2およびP3に基づくジョブはP1に基づくすべてのジョブが終了するまで実行できません。

関連項目:

非互換性定義の使用

親トピック: ジョブおよびスケジューラ・オブジェクトのサポート

28.3 ジョブに関する追加説明

様々なタイプのジョブがあります。ジョブ・インスタンスは、ジョブの特定の実行を表します。ジョブの引数を指定して、デフォルトのプログラムの引数値をオーバーライドできます。

- ジョブ・カテゴリ
  Oracle Schedulerは、いくつのジョブ・タイプをサポートします。
- ジョブ・インスタンス
  ジョブ・インスタンスは、ジョブの特定の実行を表します。1回だけ実行するようにスケジュールされているジョブには、1つのインスタンスのみがあります。繰返しスケジュールされているジョブおよびイベントが発生するたびに実行されるジョブには、複数のインスタンスがあり、ジョブの各実行がインスタンスを表します。
- ジョブ引数
  ジョブでプログラム・オブジェクト(プログラム)を参照する場合、ジョブ引数を指定してデフォルトのプログラム引数値を上書きするか、デフォルト値のないプログラム引数の値を提供できます。また、ジョブが指定するインライン処理(たとえば、ストアド・プロシージャ)に対しても引数値を提供できます。
- プログラム、ジョブおよびスケジュールの関連
  実行内容と実行時期を定義するには、プログラム、ジョブおよびスケジュール間に関連を割り当てます。

関連項目:

- 「ジョブの作成」
- 「ジョブ・ログの表示」

親トピック: Oracle Schedulerの概要

28.3.1 ジョブ・カテゴリ

Oracle Schedulerは、いくつかのジョブ・タイプをサポートします。

- データベース・ジョブ
  データベース・ジョブは、Oracle Databaseプログラム・ユニットを実行します。ローカルおよびリモート・データベース・ジョブを実行できます。
- 外部ジョブ
  外部ジョブは、データベース外で実行可能ファイルを実行します。ローカルおよびリモート外部ジョブを実行できます。
- 複数の宛先のジョブ
  複数の宛先のジョブとは、ジョブのインスタンスが複数のターゲット・データベースまたはホストで実行されても、中核となる1つのデータベースで制御および監視できるジョブのことです。
- チェーン・ジョブ
  チェーンは、依存性ベースのスケジューリングを可能にするスケジューラ・メカニズムです。
- デタッチされたジョブ
  デタッチされたジョブを使用して、スケジューラとは独立して非同期に別のプロセスで実行されるスクリプトやアプリケーションを起動できます。
- 軽量ジョブ
  軽量ジョブは、多数の短時間のジョブを頻繁に実行する場合に使用します。特定の状況では、軽量ジョブを使用することで、わずかながらパフォーマンスが向上する場合があります。
- インメモリー・ジョブ
  多数のジョブが短い期間内で作成および実行される場合は、インメモリー・ジョブを使用します。インメモリー・ジョブは、メモリー・フットプリントが少し大きくなりますが、メモリー・キャッシュが使用され、ジョブの作成と実行に必要なディスク・アクセスと時間が減少します。これによりパフォーマンスが著しく向上することがあります。
- スクリプト・ジョブ
  Oracle Database 12c以降では、SQL*Plus、RMANインタプリタまたはコマンド・シェル(Windowsの場合は`cmd.exe`、UNIXベースのシステムの場合はshシェルやその他のインタプリタなど)を使用してカスタム・ユーザー・スクリプトを実行する複数の新しいスクリプト・ジョブを使用できます。

親トピック: ジョブに関する追加説明

28.3.1.1 データベース・ジョブ

データベース・ジョブは、Oracle Databaseプログラム・ユニットを実行します。ローカルおよびリモート・データベース・ジョブを実行できます。

- データベース・ジョブについて
  データベース・ジョブでは、PL/SQL無名ブロック、PL/SQLストアド・プロシージャおよびJavaストアド・プロシージャなどのOracle Databaseプログラム・ユニットが実行されます。
- ローカル・データベース・ジョブ
  ローカル・データベース・ジョブは、ジョブの所有者であるデータベース・ユーザーとして元のデータベースで実行されます。ジョブの所有者は、ジョブが作成されたスキーマの名前です。

- リモート・データベース・ジョブ
  リモート・データベース・ジョブのターゲット・データベースは、リモート・ホスト上のOracle Databaseまたは元のデータベースと同じホスト上の別のデータベース・インスタンスにすることができます。

親トピック: ジョブ・カテゴリ

28.3.1.1.1 データベース・ジョブについて

データベース・ジョブでは、PL/SQL無名ブロック、PL/SQLストアド・プロシージャおよびJavaストアド・プロシージャなどのOracle Databaseプログラム・ユニットが実行されます。

処理がインラインで指定されているデータベース・ジョブの場合、`job_type`は`'PLSQL_BLOCK'`または`'STORED_PROCEDURE'`に設定され、`job_action`にはPL/SQL無名ブロックのテキストまたはストアド・プロシージャ名が含まれています。(プログラムがインラインで指定されているプログラム処理ではなく名前付きプログラム・オブジェクトである場合は、それに応じて対応する`program_type`および`program_action`を設定する必要があります。)

元のデータベース(ジョブが作成されたデータベース)で実行されるデータベース・ジョブをローカル・データベース・ジョブまたは単にジョブと呼びます。元のデータベース以外のターゲット・データベースで実行されるデータベース・ジョブをリモート・データベース・ジョブと呼びます。

ローカル・データベース・ジョブおよびリモート・データベース・ジョブの両方の実行結果を元のデータベースのジョブ・ログ・ビューに表示できます。

親トピック: データベース・ジョブ

28.3.1.1.2 ローカル・データベース・ジョブ

ローカル・データベース・ジョブは、ジョブの所有者であるデータベース・ユーザーとして元のデータベースで実行されます。ジョブの所有者は、ジョブが作成されたスキーマの名前です。

親トピック: データベース・ジョブ

28.3.1.1.3 リモート・データベース・ジョブ

リモート・データベース・ジョブのターゲット・データベースは、リモート・ホスト上のOracle Databaseまたは元のデータベースと同じホスト上の別のデータベース・インスタンスのどちらにもできます。

リモート・データベース・ジョブは、ジョブの`destination_name`属性に含まれる既存のデータベース宛先オブジェクトの名前を指定して識別します。

リモート・データベース・ジョブを作成するには、Oracle Database 11g リリース2 (11.2)以降が必要です。ただし、ジョブのターゲット・データベースとしては、任意のOracle Databaseリリースを使用できます。ターゲット・データベースに対するパッチは不要です。(ターゲット・データベース・ホストが元のデータベース・ホストと同じ場合でも)ターゲット・データベース・ホストにスケジューラ・エージェントをインストールして、元のデータベースにエージェントを登録する必要があるだけです。エージェントはOracle Client 11g リリース2 (11.2)以降からインストールする必要があります。

リモート・データベース・ジョブは、ターゲット・データベースで有効なユーザーとして実行する必要があります。必要なユーザー名とパスワードを、リモート・データベース・ジョブに割り当てた資格証明オブジェクトとともに指定します。

関連項目:

- 「資格証明」
- 「ジョブの作成」

- 「Oracle Schedulerエージェントを使用したリモート・ジョブの実行」
- 「ジョブ・ログの表示」

親トピック: データベース・ジョブ

28.3.1.2 外部ジョブ

外部ジョブは、データベース外で実行可能ファイルを実行します。ローカルおよびリモート外部ジョブを実行できます。

- 外部ジョブについて
  外部ジョブでは、外部実行可能ファイルが実行されます。外部実行可能ファイルは、データベースの外部で実行されるオペレーティング・システム実行可能ファイルです。
- ローカル外部ジョブについて
  ローカル外部ジョブでは、そのジョブがスケジュールされているOracle Databaseと同じコンピュータ上で外部実行可能ファイルが実行されます。このようなジョブでは、`destination_name`ジョブ属性が`NULL`になっています。
- リモート外部ジョブについて
  リモート外部ジョブは、リモート・ホスト上の外部実行可能ファイルを実行します。リモート・ホストには、Oracle Databaseがインストールされている場合も、されていない場合もあります。

親トピック: ジョブ・カテゴリ

28.3.1.2.1 外部ジョブについて

外部ジョブでは、外部実行可能ファイルが実行されます。外部実行可能ファイルは、データベースの外部で実行されるオペレーティング・システム実行可能ファイルです。

外部ジョブの場合、`job_type`は`'EXECUTABLE'`と指定されています。(名前付きプログラムを使用している場合は、対応する`program_type`が`'EXECUTABLE'`になります)。`job_action`(または対応する`program_action`)には、必要な外部実行可能ファイルのパス(コマンドライン引数を除く)が、オペレーティング・システムに依存したフルパスの形式で入ります。たとえば、`/usr/local/bin/perl`や`C:¥perl¥bin¥perl`のようになります。

Windowsのバッチ・ファイルは直接実行できないため、コマンド・プロンプト(`cmd.exe`)で実行する必要があることに注意してください。

データベース・ジョブの場合と同様に、外部ジョブを作成する際にスキーマを割り当てることができます。割り当てたスキーマはジョブの所有者になります。外部ジョブをSYSスキーマに作成することもできますが、この方法はお薦めしません。

ローカル外部ジョブまたはリモート外部ジョブを作成するには、`CREATE JOB`権限と`CREATE EXTERNAL JOB`権限の両方が必要です。

外部実行可能ファイルは、オペレーティング・システム・ユーザーとして実行する必要があります。したがって、スケジューラでは、作成する外部ジョブにオペレーティング・システムの資格証明を割り当てることができます。リモート・データベース・ジョブの場合と同様に、これらの資格証明を資格証明オブジェクト(資格証明)で指定し、資格証明を外部ジョブに割り当てます。

外部ジョブには、ローカル外部ジョブとリモート外部ジョブの2つのタイプがあります。ローカル外部ジョブでは、そのジョブがスケジュールされているデータベースと同じコンピュータ上で外部実行可能ファイルが実行されます。リモート外部ジョブは、リモート・ホスト上の実行可能ファイルを実行します。リモート・ホストにはOracle Databaseは必要なく、スケジューラ・エージェントをインストールして登録するのみで十分です。

ノート:

Windowsでは、外部実行可能ファイルを実行するホスト・ユーザーに`Log on as a batch job`ログオン権限を割り当てる必要があります。

関連項目:

- 「資格証明」
- 「Oracle Schedulerエージェントを使用したリモート・ジョブの実行」

親トピック: 外部ジョブ

28.3.1.2.2 ローカル外部ジョブについて

ローカル外部ジョブでは、そのジョブがスケジュールされているOracle Databaseと同じコンピュータ上で外部実行可能ファイルが実行されます。このようなジョブでは、`destination_name`ジョブ属性が`NULL`になっています。

ローカル外部ジョブでは、stdoutおよびstderrの出力がディレクトリORACLE_HOME/scheduler/log内のログ・ファイルに書き込まれます。これらのファイルの内容は、`DBMS_SCHEDULER.GET_FILE`で取得できます。

ローカルの外部ジョブに資格証明を割り当てる必要はありませんが、セキュリティ向上のために割り当てることをお薦めします。資格証明を割り当てない場合、デフォルトの資格証明を使用してジョブが実行されます。表28-1に、様々なプラットフォームおよび様々なジョブ所有者のデフォルト資格証明を示します。

表28-1 ローカル外部ジョブに対するデフォルトの資格証明

SYSスキーマのジョブか	プラットフォーム	デフォルトの資格証明
はい	すべて	Oracle Databaseをインストールしたユーザー。
いいえ	UNIXおよびLinux	ファイルORACLE_HOME/rdbms/admin/externaljob.oraに指定されている`run-user`属性および`run-group`属性の値
いいえ	Windows	`OracleJobSchedulerSID` Windowsサービスの実行ユーザー(ローカル・システム・アカウント、指名ローカル・ユーザーまたはドメイン・ユーザー) ノート: このサービスは手動で有効化して開始する必要があります。セキュリティ向上のため、ローカル・システム・アカウントのかわりに指名ユーザーを使用することをお薦めします。

ノート:

デフォルト資格証明は以前のリリースのOracle Databaseとの互換性のために含まれており、将来のリリースでは非推奨になる可能性があります。そのため、すべてのローカル外部ジョブに資格証明を割り当てることをお薦めし

ます。

資格証明が割り当てられていないローカル外部ジョブの実行を禁止するには、`ORACLE_HOME/rdbms/admin/externaljob.ora`ファイルから`run_user`属性を削除する(UNIXとLinuxの場合)か、`OracleJobScheduler`サービスを停止します(Windowsの場合)。SYSスキーマ内のローカル外部ジョブは、これらのステップでは実行不可になりません。

関連項目:

- ローカル外部ジョブをサポートするためのインストール後の構成ステップについては、使用しているオペレーティング・システム固有のマニュアルを参照してください。
- 例29-6

親トピック: 外部ジョブ

28.3.1.2.3 リモート外部ジョブについて

リモート外部ジョブは、リモート・ホスト上の実行可能ファイルを実行します。リモート・ホストには、Oracle Databaseがインストールされている場合も、されていない場合もあります。

特定のリモート・ホストでリモート外部ジョブを実行できるようにするには、リモート・ホストでスケジューラ・エージェントをインストールし、それをローカル・データベースで登録する必要があります。データベースは、外部実行可能ファイルを開始し、実行結果を取得するためにエージェントと通信します。

リモート外部ジョブを作成する場合、ジョブの`destination_name`属性で既存の外部宛先オブジェクトの名前を指定します。

リモート外部ジョブでは、stdoutおよびstderrの出力がディレクトリAGENT_HOME/data/log内のログ・ファイルに書き込まれます。これらのファイルの内容は、`DBMS_SCHEDULER.GET_FILE`で取得できます。例29-6は、stdout出力を取得する方法を示しています。この例はローカル外部ジョブに関するものですが、メソッドはリモート外部ジョブの場合も同じです。

関連項目:

- 「資格証明」
- 「Oracle Schedulerエージェントを使用したリモート・ジョブの実行」

親トピック: 外部ジョブ

28.3.1.3 複数の宛先のジョブ

複数の宛先のジョブとは、ジョブのインスタンスが複数のターゲット・データベースまたはホストで実行されても、中核になる1つのデータベースで制御および監視できるジョブのことです。

複数のデータベースまたは複数のホストを管理する必要があるDBAまたはシステム管理者にとって、複数の宛先のジョブは管理が大幅に簡略化されます。複数の宛先のジョブでは、次の操作を行うことができます。

- ジョブを実行する必要のある複数のデータベースまたはホストの指定。
- 複数のターゲットでスケジュールされているジョブの1回の操作による変更。
- 1つ以上のリモート・ターゲットで実行しているジョブの停止。

- 各リモート・ターゲットでのジョブ・インスタンスの状態(実行中、完了、失敗など)の判断。
- ジョブ・インスタンスの集合の全体的な状態の判断。

複数の宛先のジョブは、特定の目的のための単一エンティティとみなしたり、別の目的のために独立して実行されるジョブの集合とみなすことができます。ジョブのメタデータを作成するときには、複数の宛先のジョブは単一エンティティのように見えます。ただし、ジョブ・インスタンスが実行されているときは、各ジョブのほぼ同一コピーであるジョブの集合とみなす方が適切です。ソース・データベースで作成されたジョブを親ジョブと呼び、様々な宛先で実行されるジョブ・インスタンスを子ジョブと呼びます。

複数の宛先のジョブを作成するには、宛先グループをジョブのdestination_name属性に割り当てます。ジョブは、スケジュールされた時間または指定のイベントの検出時に、グループ内のすべての宛先で実行されます。ローカル・ホストは、ジョブが実行される宛先に含めることができます。

ジョブの処理がデータベース・プログラム・ユニットの場合は、destination_name属性でデータベース宛先グループを指定する必要があります。データベース宛先グループのメンバーには、元の(ローカル)データベースを示すデータベース宛先とキーワードLOCALが含まれます。ジョブの処理が外部実行可能ファイルの場合は、destination_name属性で外部宛先グループを指定する必要があります。外部宛先グループのメンバーには、ローカル・ホストを示す外部宛先とキーワードLOCALが含まれます。

ノート:

データベース宛先は、リモート・データベースを必ずしも参照する必要はありません。データベース宛先は、ジョブが作成されたデータベースと同じホストで実行している他のデータベースを参照できます。

複数の宛先のジョブとタイム・ゾーン

一部のジョブの宛先は、親ジョブが作成されたデータベース(元のデータベース)のタイムゾーンとは異なるタイムゾーンにある場合があります。この場合、ジョブの開始時刻は、常に元のデータベースのタイム・ゾーンを基にします。したがって、親ジョブをロンドン(英国)で作成して開始時刻を午後8時に指定し、宛先として東京、ロサンゼルス、ニューヨークを指定した場合、子ジョブはロンドン時刻の午後8時に開始します。すべての宛先での開始時刻は、システムの負荷の違いや再試行を必要とする問題などによって、多少のずれが発生する場合があります。

イベント・ベースの複数の宛先のジョブ

イベント・ベースの複数の宛先のジョブの場合、親ジョブがイベントをホストで検出すると、すべての子ジョブがすべての宛先で開始します。子ジョブは、それぞれのホストでイベントを検出しません。

関連項目:

- 「複数の宛先のジョブの作成」
- 「複数の宛先のジョブの監視」
- 「宛先グループ」
- 「イベントを使用したジョブの開始」

親トピック: ジョブ・カテゴリ

28.3.1.4 チェーン・ジョブ

チェーンは、依存性ベースのスケジューリングを可能にするスケジューラ・メカニズムです。

最も単純な形式では、プログラム・オブジェクトのグループとプログラム・オブジェクト間の依存性が定義されます。ジョブは、単一のプログラム・オブジェクトを指し示すかわりにチェーンを指し示すことができます。その場合、ジョブはチェーンを開始する役割を持ちます。チェーン・ジョブの場合、job_typeは'CHAIN'として指定されます。

関連項目:

- 「チェーン」
- 「ジョブ・チェーンの作成と管理」

親トピック: ジョブ・カテゴリ

28.3.1.5 デタッチされたジョブ

デタッチされたジョブを使用して、スケジューラとは独立して非同期に別のプロセスで実行されるスクリプトやアプリケーションを起動できます。

通常、デタッチされたジョブは別のプロセスを起動してから終了します。終了時(ジョブ処理の完了時)に、デタッチされたジョブは実行状態のままになります。この実行状態は、このジョブが起動した非同期プロセスがまだアクティブであることを示しています。非同期プロセスは、その作業を完了するときに、データベースに接続して、ジョブを終了させるDBMS_SCHEDULER.END_DETACHED_JOB_RUNを呼び出す必要があります。

use_current_sessionパラメータがTRUEに設定されている場合、run_jobを使用して手動で実行をトリガーして、デタッチされたジョブを実行することはできません。

ジョブは、detached属性がTRUEに設定されているプログラム・オブジェクト(プログラム)(デタッチ済プログラム)を指している場合、デタッチ済です。

デタッチ済ジョブは、次の2つの場合に使用します。

- 必要以上にリソースを保持するため、起動した非同期プロセスが完了するまで待機することが実際的でない場合。

  一例として、非同期Webサービスにリクエストを送信する場合があります。Webサービスが応答するのに何時間または何日もかかる場合があり、応答待ちの間、スケジューラ・ジョブ・スレーブを保留することはできません。(ジョブ・スレーブの詳細は、「スケジューラのアーキテクチャ」を参照してください。)

- プロセスによりデータベースが停止されるため、起動した非同期プロセスが完了するまで待機できない場合。

  たとえば、スケジューラ・ジョブを使用して、データベースの停止、コールド・バックアップの作成およびデータベースの再起動を実行するRMANスクリプトを起動する場合です。「デタッチ済ジョブの作成」を参照してください。

デタッチ済ジョブの動作は次のとおりです。

1. ジョブの開始時に、ジョブ・コーディネータによりジョブにジョブ・スレーブが割り当てられ、デタッチ済プログラムに定義されているプログラム処理がジョブ・スレーブにより実行されます。プログラム処理は、PL/SQLブロック、ストアド・プロシージャまたは外部実行可能ファイルのいずれかです。
2. プログラム処理は、別のスクリプトまたは実行可能ファイル(この例ではプロセスA)の即時リターン・コールを実行してから終了します。プログラム処理の動作は完了しているため、ジョブ・スレーブは終了しますが、ジョブは実行中の状態のまま残ります。
3. プロセスAにより処理が実行されます。データベースに対するDMLを実行する場合は、その処理をコミットする必要があります。処理が完了すると、プロセスAによりデータベースにログが記録され、END_DETACHED_JOB_RUNがコールされます。

4. デタッチ済ジョブが完了として記録されます。

実行中のデタッチ済ジョブを終了するには、STOP_JOBをコールする方法もあります。

関連項目:

デタッチ済ジョブを使用してデータベースのコールド・バックアップを実行する例は、「デタッチ済ジョブの作成」を参照してください

親トピック: ジョブ・カテゴリ

28.3.1.6 軽量ジョブ

頻繁に実行する短時間のジョブが多数ある場合は、軽量ジョブを使用します。特定の状況では、軽量ジョブを使用することで、わずかながらパフォーマンスが向上する場合があります。

軽量ジョブには、次の特性があります。

- 標準ジョブとは異なり、スキーマ・オブジェクトではありません。
- スキーマ・オブジェクトを作成する際のオーバーヘッドを伴わないため、作成および削除に必要な時間が標準ジョブよりも大幅に短縮されます。
- セッションの平均作成時間が標準ジョブよりも短縮されます。
- ジョブのメタデータと実行時データに関して、ディスク上のフットプリントが小さくなります。

軽量ジョブを指定するには、job_styleジョブ属性を'LIGHTWEIGHT'に設定します。(デフォルトのジョブ・スタイルは'REGULAR'です。)

プログラムやスケジュールと同様に、標準ジョブはスキーマ・オブジェクトです。通常のジョブは最も柔軟性がありますが、作成または削除されるときに多少のオーバーヘッドを伴います。ユーザーはジョブの権限を細かく制御でき、別のユーザーが所有するプログラムやストアド・プロシージャをジョブの処理として指定できます。

実行頻度が低い比較的少数のジョブを作成する必要がある場合は、軽量ジョブよりも標準ジョブが優先されます。

軽量ジョブでは、プログラム・オブジェクト(プログラム)を参照してジョブの処理を指定する必要があります。プログラムは軽量ジョブの作成時に有効化されている必要があり、プログラム・タイプは'PLSQL_BLOCK'または'STORED_PROCEDURE'であることが必要です。軽量ジョブはスキーマ・オブジェクトではないため、権限は付与できません。軽量ジョブは指定のプログラムから権限を継承します。したがって、プログラムに対して特定の権限セットを持つユーザーは、軽量ジョブに対して対応する権限を持つことになります。

関連項目:

「名前付きプログラムとジョブ・スタイルを使用したジョブの作成」

親トピック: ジョブ・カテゴリ

28.3.1.7 インメモリー・ジョブ

多数のジョブが短い期間内で作成および実行される場合は、インメモリー・ジョブを使用します。インメモリー・ジョブは、メモリー・フットプリントが少し大きくなりますが、メモリー・キャッシュが使用され、ジョブの作成と実行に必要なディスク・アクセスと時間が減少します。これによりパフォーマンスが著しく向上することがあります。

使用できるインメモリー・ジョブのタイプは、ランタイム(IN_MEMORY_RUNTIME)およびフル(IN_MEMORY_FULL)です。

- インメモリー・ランタイム・ジョブは、ライトウェイト・ジョブに基づいているため、永続的です。これらは繰返し間隔を持つことができ、複数回実行できます。

  デフォルトでは、インメモリー・ジョブはjob_class DEFAULT_IN_MEMORY_JOB_CLASSに関連付けられ、ロギング・レベルはNONEです。これは、デフォルトでは、インメモリー・ジョブではスケジューラ・ジョブに関連してロギングが生成されないため、パフォーマンスが向上することを意味します。

- インメモリー・フル・ジョブは、メモリー内にキャッシュされた状態でのみ存在するため、永続的ではありません。これらにはプログラムが関連付けられている必要があり、一度だけ実行して破棄するためのものであるため、繰返し間隔を持つことはできません。これらはディスクにバックアップがないため、作成時または実行時にREDOが生成されず、操作が大幅に速くなります。

  インメモリー・フル・ジョブは、作成されたインスタンスまたはいずれかのRACインスタンスにのみ存在します。これらは、論理または物理スタンバイ・インスタンスに伝播されないため、プライマリ・インスタンスがスタンバイに切り替えられた場合、実行できなくなります(破棄されます)。

関連項目:

「名前付きプログラムとジョブ・スタイルを使用したジョブの作成」

親トピック: ジョブ・カテゴリ

28.3.1.8 スクリプト・ジョブ

Oracle Database 12c以降では、SQL*Plus、RMANインタプリタまたはコマンド・シェル(Windowsの場合はcmd.exe、UNIXベースのシステムの場合はshシェルやその他のインタプリタなど)を使用してカスタム・ユーザー・スクリプトを実行する複数の新しいスクリプト・ジョブを使用できます。

これらのすべての実行可能ファイルには、OSの資格証明が必要となります。これらのスクリプト・ジョブは次のとおりです。

- SQLスクリプト・ジョブ: データベース宛先が必要です。

  SQLスクリプト・ジョブでは、SQL*Plusインタプリタを使用してスケジューラ・ジョブが実行されます。このため、問合せ出力の書式設定を含むすべてのSQL*Plus機能を使用できます。

  起動後にデータベースに接続するには、SQLスクリプト・ジョブに認証ステップが必要です。ユーザーは、インラインで、ジョブ処理で、またはスケジューラによって提供されるconnect_credential機能を使用して認証できます。connect_credential機能を使用するには、ユーザーはジョブのconnect_credential_name属性を設定します。その後、ジョブでは、そのconnect_credentialのユーザー名、パスワードおよびロールを使用してデータベースへの接続を試行します。

- 外部スクリプト・ジョブ: 通常の宛先が必要です。

  外部スクリプト・ジョブは新しいシェル・インタプリタを起動し、コマンドライン・スクリプトを簡単な方法で実行できるようにします。

- バックアップ・スクリプト・ジョブ: データベース宛先が必要です。

  Backupスクリプト・ジョブは、バックアップ・タスクを作成および実行するRMANセキュリティを指定する、より直接的な方法を提供します。

  起動後にデータベースに接続するには、バックアップ・スクリプト・ジョブに認証ステップが必要です。ユーザーは、インラインで、ジョブ処理で、またはスケジューラによって提供されるconnect_credential機能を使用して認証できます。

connect_credential機能を使用するには、ユーザーはジョブのconnect_credential_name属性を設定します。その後、ジョブでは、そのconnect_credentialのユーザー名、パスワードおよびロールを使用してデータベースへの接続を試行します。

ジョブまたはプログラム処理では、各インタプリタの適切なスクリプトを指すか、または適切なインライン・スクリプトが指定されている必要があります。詳細は、CREATE_JOBサブプログラムのjob_actionパラメータまたはCREATE_PROGRAMサブプログラムのprogram_actionパラメータを参照してください。

関連項目:

- CREATE_JOBパラメータの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- CREATE_PROGRAMパラメータの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・カテゴリ

28.3.2 ジョブ・インスタンス

ジョブ・インスタンスは、ジョブの特定の実行を表します。1回だけ実行するようにスケジュールされているジョブには、1つのインスタンスのみがあります。繰返しスケジュールされているジョブおよびイベントが発生するたびに実行されるジョブには、複数のインスタンスがあり、ジョブの各実行がインスタンスを表します。

たとえば、2009年10月8日火曜日のみに実行するようにスケジュールされているジョブには1つのインスタンスがあり、1週間毎日正午に実行されるジョブには7つのインスタンスがあり、ファイルがリモート・システムに到着すると実行されるジョブには、ファイル到着イベントごとに1つのインスタンスがあります。

複数の宛先のジョブには、宛先ごとに1つのインスタンスがあります。複数の宛先のジョブが繰返しスケジュールされている場合は、各宛先のジョブの実行ごとに1つのインスタンスがあります。

ジョブが作成されると、ジョブを表すエントリがスケジューラのジョブ表に1つのみ追加されます。ロギング・レベルの設定に応じて、ジョブが実行されるたびに、ジョブ・ログにエントリが追加されます。そのため、繰返しのスケジュールが設定されているジョブを作成すると、ジョブ・ビュー(*_SCHEDULER_JOBS)には1つのエントリが存在し、ジョブ・ログには複数のエントリが存在することになります。各ジョブ・インスタンスのログ・エントリは、ジョブの完了ステータス、開始時刻、終了時刻など、特定の実行に関する情報を提供します。ジョブの各実行には一意のログIDが割り当てられて、ジョブ・ログとジョブ実行詳細ビュー(*_SCHEDULER_JOB_LOGと*_SCHEDULER_JOB_RUN_DETAILS)の両方に表示されます。

関連項目:

- 「ジョブの監視」
- 「スケジューラのデータ・ディクショナリ・ビュー」

親トピック: ジョブに関する追加説明

28.3.3 ジョブ引数

ジョブでプログラム・オブジェクト(プログラム)を参照する場合、ジョブ引数を指定してデフォルトのプログラム引数値を上書きするか、デフォルト値のないプログラム引数の値を提供できます。また、ジョブが指定するインライン処理(たとえば、ストアド・プロシー

ジャ)に対しても引数値を提供できます。

必要なプログラム引数値のすべてが、参照先のプログラム・オブジェクトにデフォルトとして定義されているか、またはジョブ引数として定義されるまで、ジョブは使用できません。

ジョブの一般的な例には、一連のレポートを夜間に実行するジョブがあります。様々な部門で様々なレポートが必要な場合は、このタスクのプログラムを、様々な部門の様々なユーザー間で共有できるように作成できます。このプログラム処理では、レポート・スクリプトが実行され、プログラムには1つの引数として、部門番号を設定します。各ユーザーはこのプログラムを指し示すジョブを作成し、部門番号をジョブ引数として指定できます。

関連項目:

- 「ジョブ引数の設定」
- 「プログラム引数の定義」
- 「ジョブの作成」

親トピック: ジョブに関する追加説明

28.3.4 プログラム、ジョブおよびスケジュールの関連

実行内容と実行時期を定義するには、プログラム、ジョブおよびスケジュール間に関連を割り当てます。

図28-5に、これらの関連の例を示します。

図28-5 プログラム、ジョブおよびスケジュール間の関連

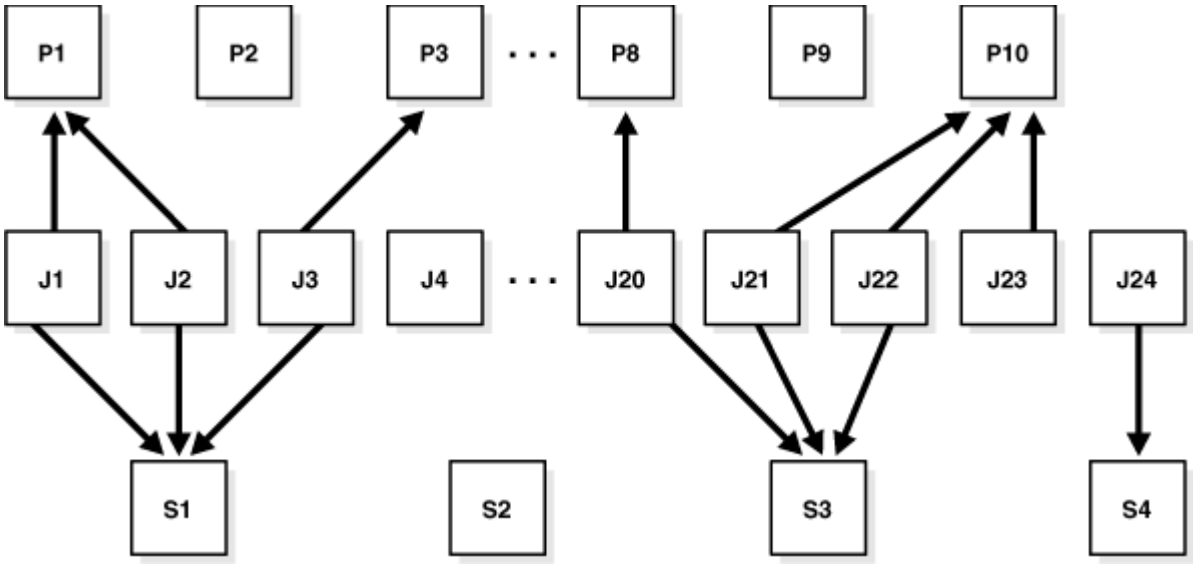


図28-5を理解するために、表を分析する場合を考えてみます。この例では、プログラムP1がDBMS_STATSパッケージを使用して表を分析します。このプログラムには、表名に対する入力パラメータがあります。2つのジョブJ1とJ2は、両方とも同じプログラムを指し示していますが、別々の表名が指定されています。さらに、スケジュールS1には、毎日午前2時の実行時間を指定します。最終的な結果として、J1とJ2で名前が指定された2つの表は、毎日午前2時に分析されます。

J4は、すべての関連情報がそのジョブ自体に定義された自己完結のジョブで、他のエンティティを指し示していないことに注意してください。P2、P9およびS2は、必要に応じて、プログラムまたはスケジュールに関連を割り当てないままにできることを示しています。たとえば、年度末の在庫を計算するプログラムを作成し、一時的に、そのプログラムをどのジョブにも割り当てないままにできます。

親トピック: ジョブに関する追加説明

28.4 スケジューラのアーキテクチャ

スケジューラ・コンポーネントはジョブを処理します。

- スケジューラ・コンポーネント
  スケジューラ・コンポーネントには、ジョブ表、ジョブ・コーディネータおよびジョブ・スレーブが含まれます。
- ジョブ表
  ジョブ表はすべてのジョブのコンテナであり、各データベースに1つの表があります。ジョブ表には、すべてのジョブに関する所有者名やロギング・レベルなどの情報が保存されています。この情報は、*_SCHEDULER_JOBSビューにあります。
- ジョブ・コーディネータ
  ジョブ・コーディネータは、ジョブ・スレーブを起動します。
- ジョブの実行方法
  送信したジョブは、ジョブ・スレーブによって実行されます。
- ジョブの完了後
  ジョブの完了後、スレーブがいくつかの処理を実行します。
- Real Application Clusters環境におけるスケジューラの使用
  Oracle Real Application Clusters環境で、スケジューラを使用できます。

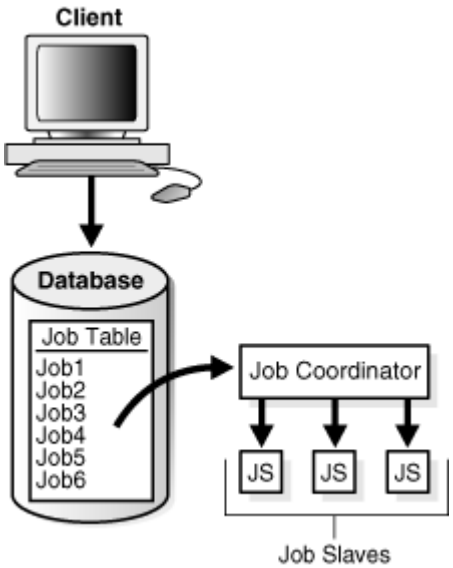
親トピック: Oracle Schedulerの概要

28.4.1 スケジューラ・コンポーネント

スケジューラ・コンポーネントには、ジョブ表、ジョブ・コーディネータおよびジョブ・スレーブが含まれます。

図28-6に、データベースによるジョブの処理方法を示します。

図28-6 スケジューラ・コンポーネント



親トピック: スケジューラのアーキテクチャ

28.4.2 ジョブ表

ジョブ表はすべてのジョブのコンテナであり、データベースごとにあります。ジョブ表には、すべてのジョブに関する所有者名やロギング・レベルなどの情報が保存されています。この情報は、*_SCHEDULER_JOBSビューにあります。

ジョブはデータベース・オブジェクトであるため、累積されて多くの領域を使用する場合があります。これを回避するために、ジョブ・オブジェクトは、ジョブの完了後、デフォルトで自動的に削除されます。この動作は、auto_dropジョブ属性によって制御されます。

使用可能なジョブのビューと管理については、「スケジューラのデータ・ディクショナリ・ビュー」を参照してください。

親トピック: スケジューラのアーキテクチャ

28.4.3 ジョブ・コーディネータ

ジョブ・コーディネータは、ジョブ・スレーブを起動します。

- ジョブ・コーディネータについて
  ジョブ・コーディネータは、データベースの制御のもと、ジョブ表の情報を使用してジョブ・スレーブを制御および起動します。
- ジョブ・コーディネータの処理
  ジョブ・コーディネータは、いくつかの処理を実行します。
- スケジューラ・ジョブ・プロセスの最大数
  コーディネータは、CPU負荷および処理中のジョブ数に基づいて、起動するジョブ・スレーブ数を自動的に決定します。

親トピック: スケジューラのアーキテクチャ

28.4.3.1 ジョブ・コーディネータについて

ジョブ・コーディネータは、データベースの制御のもと、ジョブ表の情報を使用してジョブ・スレーブを制御および起動します。

ジョブ・コーディネータ・バックグラウンド・プロセス(cjqNNN)は、必要に応じて自動的に起動および停止します。データベースの起動時に、ジョブ・コーディネータは起動されませんが、実行するジョブがあるかどうか、および近い将来開くウィンドウがあるかどうかをデータベースが監視します。ある場合は、コーディネータが起動されます。

実行中のジョブまたはウィンドウがある間は、コーディネータは継続的に実行されます。スケジューラが一定期間非アクティブな状態にあり、近い将来にスケジュールされたジョブまたはウィンドウがない場合、コーディネータは自動的に停止します。

ジョブ・コーディネータを起動するかどうかをデータベースが判断する際は、ジョブのサービス・アフィニティが考慮されます。たとえば、近い将来スケジュールされるジョブが1つのみで、4つのRACインスタンスのうち2つのみにサービス・アフィニティがあるジョブ・クラスにこのジョブが属している場合は、この2つのインスタンスのジョブ・コーディネータのみが起動されます。詳細は、「スケジューラ使用時のサービス・アフィニティ」を参照してください。

親トピック: ジョブ・コーディネータ

28.4.3.2 ジョブ・コーディネータの処理

ジョブ・コーディネータは、いくつかの処理を実行します。

ジョブ・コーディネータの機能は次のとおりです。

- ジョブ・スレーブを制御および起動します。
- ジョブ表を問い合せます。
- ジョブ表からジョブを定期的に取り出し、メモリー・キャッシュに格納します。この結果、ディスクへのトリップが減るため、パフォーマンスが向上します。
- メモリー・キャッシュからジョブを取り出し、実行するためにジョブ・スレーブに渡します。
- スレーブが不要になると、ジョブ・スレーブ・プールをクリーン・アップします。

- スケジュールされているジョブがないときは休止状態になります。
- 新規ジョブが実行されるとき、またはCREATE_JOBプロシージャを使用してジョブが作成されたときに起動します。
- データベースの異常停止後の起動時に、実行していたジョブをリカバリします。

ジョブ・コーディネータがジョブ表を確認する時間を設定する必要はありません。タイム・フレームはシステムで自動的に選択されます。

インスタンスごとに1つのジョブ・コーディネータが使用されます。これは、Oracle RAC環境の場合も同じです。

関連項目:

ジョブ・コーディネータの管理については、「スケジューラのデータ・ディクショナリ・ビュー」、Oracle RACの情報については、「Real Application Clusters環境におけるスケジューラの使用」

親トピック: ジョブ・コーディネータ

28.4.3.3 スケジューラ・ジョブ・プロセスの最大数

コーディネータは、CPU負荷および処理中のジョブ数に基づいて、起動するジョブ・スレーブ数を自動的に決定します。

JOB_QUEUE_PROCESSES初期化パラメータを使用して、スケジューラが起動できるジョブ・スレーブの数を制限できます。

関連項目:

JOB_QUEUE_PROCESSES初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: ジョブ・コーディネータ

28.4.4 ジョブの実行方法

送信したジョブは、ジョブ・スレーブによって実行されます。

ジョブの実行時期になるとジョブ・コーディネータによって起動されます。ジョブ・スレーブは、ジョブ表からジョブを実行するためのメタデータを収集します。

処理のためにジョブが取り出されると、ジョブ・スレーブは次のことを実行します。

1. プログラム引数や権限情報など、ジョブの実行に必要なすべてのメタデータを収集します。
2. ジョブの所有者としてデータベース・セッションを開始し、トランザクションを開始した後に、ジョブの実行を開始します。
3. ジョブが完了すると、スレーブはトランザクションをコミットし、終了します。
4. セッションをクローズします。

親トピック: スケジューラのアーキテクチャ

28.4.5 ジョブの完了後

ジョブの完了後、スレーブがいくつかの処理を実行します。

ジョブが終了すると、スレーブは次のことを実行します。

- 必要に応じて、ジョブを再スケジュールします。

- ジョブ表の状態を更新して、ジョブを完了するか、再スケジュールするかを反映します。
- エントリをジョブ・ログ表に挿入します。
- 実行回数を更新し、必要に応じて、失敗の回数と再試行の回数を更新します。
- クリーン・アップします。
- 新規作業を検索します(検出されない場合は休止状態になります)。

スケジューラは、必要に応じてスレーブ・プールを動的にサイズ変更します。

親トピック: スケジューラのアーキテクチャ

28.4.6 Real Application Clusters環境におけるスケジューラの使用

Oracle Real Application Clusters環境で、スケジューラを使用できます。

- スケジューラとReal Application Clusters
  Oracle Real Application Clusters (Oracle RAC)環境では、スケジューラは、データベースごとに1つのジョブ表を使用し、インスタンスごとに1つのジョブ・コーディネータを使用します。
- スケジューラ使用時のサービス・アフィニティ
  スケジューラを使用すると、ジョブを実行するデータベース・サービス(サービス・アフィニティ)を指定できます。

親トピック: スケジューラのアーキテクチャ

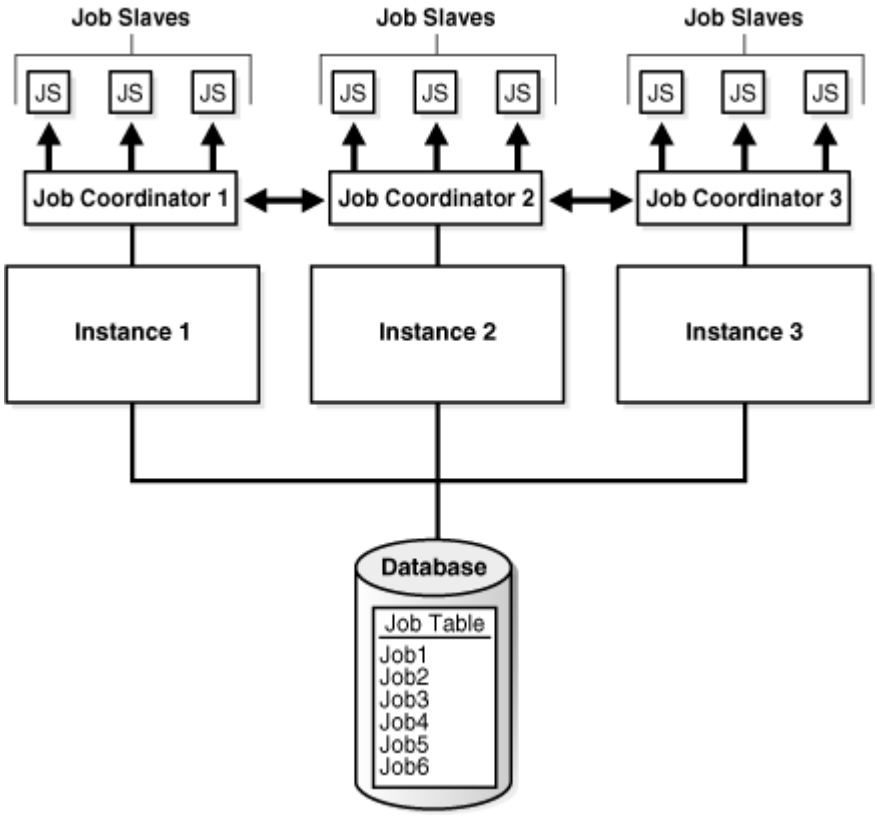
28.4.6.1 スケジューラおよびReal Application Clusters

Oracle Real Application Clusters (Oracle RAC)環境では、スケジューラは、データベースごとに1つのジョブ表を使用し、インスタンスごとに1つのジョブ・コーディネータを使用します。

ジョブ・コーディネータは相互に通信して情報を最新に保ちます。ジョブ・クラスにサービス・アフィニティがない場合、スケジューラは、使用可能なすべてのインスタンス間でジョブ・クラスのジョブの負荷を分散しようとしますが、ジョブ・クラスにサービス・アフィニティがある場合は、特定のサービスに割り当てられているインスタンス間で負荷を分散しようとします。

図28-7に、典型的なOracle RACアーキテクチャを示します。各インスタンスのジョブ・コーディネータは他のコーディネータと情報を交換します。

図28-7 Oracle RACのアーキテクチャとスケジューラ



親トピック: Real Application Clusters環境におけるスケジューラの使用

28.4.6.2 スケジューラ使用時のサービス・アフィニティ

スケジューラを使用すると、ジョブを実行するデータベース・サービス(サービス・アフィニティ)を指定できます。

これにより、インスタンスがダウンした場合に他のノードにサービスが必ず動的に割り当てられるため、インスタンス・アフィニティの場合よりも可用性がよくなります。インスタンス・アフィニティにはこの機能がないので、インスタンスがダウンすると、そのインスタンスへのアフィニティが指定されたジョブは、そのインスタンスがダウン状態から回復するまで実行されません。図28-8は、サービスおよびインスタンスの一般的な使用例を示しています。

図28-8 サービス・アフィニティとスケジューラ

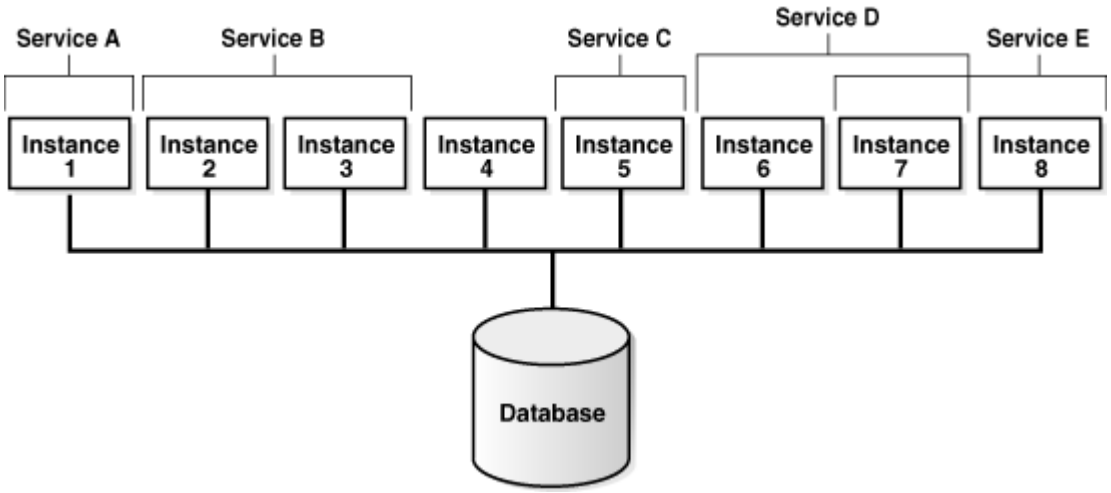


図28-8では、サービスのプロパティが変更可能で、その変更はスケジューラによって自動的に認識されます。

各ジョブ・クラスで、データベース・サービスを指定できます。サービスの指定がない場合、このジョブ・クラスは、起動しているすべてのインスタンスへのマッピングが保証されている内部サービスに属します。

親トピック: Real Application Clusters環境におけるスケジューラの使用

28.5 スケジューラによるOracle Data Guardのサポート

Oracle Database 11g リリース1 (11.1)からは、データベースがプライマリ・データベースかロジカル・スタンバイ・データベースかに基づいて、スケジューラがOracle Data Guard環境でジョブを実行できるようになりました。

フィジカル・スタンバイ・データベースの場合、スケジューラのオブジェクトに対する変更またはプライマリ・データベース上のスケジューラのジョブによるデータベースの変更は、他のデータベースの変更と同様にフィジカル・スタンバイに適用されます。

プライマリ・データベースおよびロジカル・スタンバイ・データベースの場合は、データベースがプライマリ・データベースまたはロジカル・スタンバイのロールの場合にのみ、ジョブを実行できるように指定できるという追加機能があります。これは、`DBMS_SCHEDULER.SET_ATTRIBUTE`プロシージャを使用して、`database_role`ジョブ属性を`'PRIMARY'`または`'LOGICAL STANDBY'`の2つの値のどちらかに設定して使用します。(両方のロールでジョブを実行するには、ジョブのコピーを作成して、1つのジョブの`database_role`を`'PRIMARY'`に設定し、もう1つのジョブでは`'LOGICAL STANDBY'`に設定します。)スイッチオーバーまたはフェイルオーバー時には、新しいロール固有のジョブを実行するようにスケジューラが自動的に切り替わります。プライマリ・データベースで障害が発生するまで正常に実行されたレコードを、フェイルオーバー時に使用できるように、ジョブ・イベント・ログにDMLが複製されます。

プライマリからロジカル・スタンバイへのスケジューラ・ジョブのレプリケーションは、`DBMS_ROLLING`パッケージを使用して実行されたローリング・アップグレードのアップグレード・ターゲットに制限されます。

関連項目:

- `database_role`属性の設定例は、「属性の設定例」を参照してください
- 「Oracle Data Guard環境でのジョブの作成例」
- 『Oracle Data Guard概要および管理』

親トピック: Oracle Schedulerの概要

29 Oracle Schedulerを使用したジョブのスケジューリング

Oracle Schedulerでジョブを作成、実行および管理できます。

ノート:

この章では、DBMS_SCHEDULER パッケージを使用してスケジューラ・オブジェクトを処理する方法について説明します。Oracle Enterprise Manager Cloud Control を使用して同じタスクを実行し、それらの作業の多くを Oracle SQL Developer を使用して実行できます。

DBMS_SCHEDULER の詳細は『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を、Oracle Scheduler の各ページの詳細は Cloud Control のオンライン・ヘルプを参照してください。

- スケジューラ・オブジェクトとそのネーミングについて
  Oracle Schedulerは、一連のスケジューラ・オブジェクトを作成および管理することにより操作します。各スケジューラ・オブジェクトは、[schema.]nameの形式の完全なデータベース・スキーマ・オブジェクトです。スケジューラ・オブジェクトは、データベース・オブジェクトの命名規則に正確に従い、他のデータベース・オブジェクトとSQLネームスペースを共有します。
- ジョブの作成、実行および管理
  ジョブとは、スケジュールとプログラムの組合せに、プログラムに必要な追加の引数が指定されたものです。
- ジョブを定義するためのプログラムの作成および管理
  プログラムは、特定のタスクに関するメタデータの集まりです。必要に応じて、プログラムを使用してジョブの定義を支援できます。
- ジョブを定義するためのスケジュールの作成および管理
  必要に応じて、スケジュール・オブジェクト(スケジュール)を使用してジョブの実行時期を定義できます。スケジュールは、データベースにオブジェクトとして作成および保存することによってユーザー間で共有できます。
- イベントを使用したジョブの開始
  イベントが送信されたとき、Oracle Schedulerはジョブを開始できます。イベントは、アプリケーションまたはシステム・プロセス間で送信されるメッセージです。
- ジョブ・チェーンの作成と管理
  ジョブ・チェーン(「チェーン」)は、結合した1つの目的のために互いにリンクされた一連の名前付きタスクです。
- 非互換性定義の使用
  非互換性定義(または非互換性)は、互換性のないジョブまたはプログラムを指定し、該当する場合は一度に1つのグループのみを実行できるようにします。
- ジョブ・リソースの管理
  ジョブが使用できるリソースを作成および変更したり、ジョブが使用できる指定されたリソースの量を制御できます。
- ジョブの優先度付け
  3つのスケジューラ・ジョブ(ジョブ・クラス、ウィンドウおよびウィンドウ・グループ)を使用して、Oracle Schedulerジョブに優先度を付けます。これらのオブジェクトでは、ジョブをデータベース・リソース・マネージャのコンシューマ・グループに関連付けることによって、ジョブに優先度を付けます。これにより、これらのジョブに割り当てられるリソースの量が制御されます。また、ジョブ・クラスでは、グループ内のすべてのジョブに同一のリソース・レベルが割り当てられている場合に、ジョブのグループ間に相対的な優先度を設定できます。
- ジョブの監視

いくつかの異なる方法でジョブを監視できます。

親トピック: データベース・リソースの管理とタスクのスケジューリング

29.1 スケジューラ・オブジェクトとそのネーミングについて

Oracle Schedulerは、一連のスケジューラ・オブジェクトを作成および管理することにより操作します。各スケジューラ・オブジェクトは、[schema.]nameの形式の完全なデータベース・スキーマ・オブジェクトです。スケジューラ・オブジェクトは、データベース・オブジェクトの命名規則に正確に従い、他のデータベース・オブジェクトとSQLネームスペースを共有します。

SQLの命名規則に従ってDBMS_SCHEDULERパッケージのスケジューラ・オブジェクトに名前を付けます。デフォルトでは、二重引用符で囲まれていないかぎり、スケジューラ・オブジェクトの名前は大文字になります。たとえば、ジョブを作成する際に、job_name => 'my_job'はjob_name => 'My_Job'およびjob_name => 'MY_JOB'と同じですが、job_name => '"my_job"'とは異なります。スケジューラ・オブジェクト名のカンマ区切りリストがDBMS_SCHEDULERパッケージ内で使用される場合も、これらの命名規則に従います。

関連項目:

- オブジェクトのネーミングの詳細は、『Oracle Database SQL言語リファレンス』を参照してください。
- ジョブおよびスケジューラ・オブジェクトのサポートについて

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.2 ジョブの作成、実行および管理

ジョブとは、スケジュールとプログラムの組合せに、プログラムに必要な追加の引数が指定されたものです。

- ジョブのタスクとそのプロシージャ
  DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なジョブ・タスクを管理します。
- ジョブの作成
  ジョブを作成するには、DBMS_SCHEDULERパッケージまたはCloud Controlを使用します。
- ジョブの変更
  ジョブを変更するには、その属性を変更します。そのためには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTE、SET_ATTRIBUTE_NULLまたはSET_JOB_ATTRIBUTESプロシージャまたはCloud Controlを使用します。
- ジョブの実行
  いくつかの異なる方法でジョブを実行できます。
- ジョブの停止
  1つ以上の実行中のジョブを停止するには、DBMS_SCHEDULERパッケージのSTOP_JOBプロシージャまたはCloud Controlを使用します。
- 外部ジョブの停止
  スケジューラを使用している場合、外部ジョブの実装者は、forceがFALSEに設定されたSTOP_JOBがコールされた場合に、その外部ジョブを正常にクリーン・アップするメカニズムを利用できます。
- チェーン・ジョブの停止
  実行中のチェーンを指しているジョブが停止すると、実行中のチェーンのステップがすべて停止します。
- ジョブの削除
  1つ以上のジョブを削除するには、DBMS_SCHEDULERパッケージのDROP_JOBプロシージャまたはCloud Controlを

使用します。

- 実行中のジョブの削除
  DROP_JOBプロシージャが呼び出されたときにジョブが実行中である場合、ジョブを削除しようとすると失敗します。このデフォルトの動作は、forceまたはdeferオプションを設定することによって変更できます。
- 複数のジョブの削除
  削除する複数のジョブを指定したとき、いずれかのジョブでエラーが発生した場合、その結果はDBMS_SCHEDULER.DROP_JOBプロシージャのcommit_semantics引数によって決定されます。
- ジョブの無効化
  1つ以上のジョブを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャまたはCloud Controlを使用します。
- ジョブの有効化
  1つ以上のジョブを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャまたはCloud Controlを使用します。
- ジョブのコピー
  ジョブをコピーするには、DBMS_SCHEDULERのCOPY_JOBプロシージャまたはCloud Controlを使用します。

関連項目:

ジョブの概要については、「ジョブ」。

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.2.1 ジョブのタスクとそのプロシージャ

DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なジョブ・タスクを管理します。

表29-1に、ジョブの一般的なタスクとそれに対応するプロシージャおよび権限を示します。

表29-1 ジョブのタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
ジョブの作成	CREATE_JOB または CREATE_JOBS	CREATE JOB または CREATE ANY JOB
ジョブの変更	SET_ATTRIBUTE または SET_JOB_ATTRIBUTES	ALTER または CREATE ANY JOB、あるいは所有者
ジョブの実行	RUN_JOB	ALTER または CREATE ANY JOB、あるいは所有者
ジョブのコピー	COPY_JOB	ALTER または CREATE ANY JOB、あるいは所有者
ジョブの削除	DROP_JOB	ALTER または CREATE ANY JOB、あるいは所有者
ジョブの停止	STOP_JOB	ALTER または CREATE ANY JOB、あるいは所有者

タスク	プロシージャ	必要な権限
ジョブの使用禁止	DISABLE	ALTER または CREATE ANY JOB、あるいは所有者
ジョブの使用可能化	ENABLE	ALTER または CREATE ANY JOB、あるいは所有者

権限の詳細は、「スケジューラ権限」を参照してください。

親トピック: ジョブの作成、実行および管理

29.2.2 ジョブの作成

ジョブを作成するには、DBMS_SCHEDULERパッケージまたはCloud Controlを使用します。

- ジョブ作成の概要
  1つ以上のジョブを作成するには、DBMS_SCHEDULER.CREATE_JOBまたはDBMS_SCHEDULER.CREATE_JOBSプロシージャ、あるいはCloud Controlを使用します。
- ジョブの処理、スケジュール、プログラムおよびスタイルの指定
  CREATE_JOBプロシージャはオーバーロードになるため、複数の異なる使用方法があります。
- スケジューラ・ジョブの資格証明の指定
  Oracle Schedulerには、実行前にOracle Databaseまたはオペレーティング・システムでの認証に使用するジョブの資格証明が必要です。
- 宛先の指定
  リモート外部ジョブおよびリモート・データベース・ジョブの場合は、宛先オブジェクトを作成してdestination_nameジョブ属性に割り当てることにより、ジョブの宛先を指定します。NULLのdestination_name属性が指定されたジョブは、ジョブが作成されたホストで実行されます。
- 複数の宛先のジョブの作成
  複数の宛先で実行し、1箇所で管理するジョブを作成できます。
- ジョブ引数の設定
  ジョブ引数を設定するには、SET_JOB_ARGUMENT_VALUEまたはSET_JOB_ANYDATA_VALUEプロシージャ、あるいはCloud Controlを使用します。SET_JOB_ANYDATA_VALUEは、VARCHAR2文字列として表現できない複雑なデータ型に使用されます。
- ジョブ属性の追加設定
  ジョブの作成後に、SET_ATTRIBUTEまたはSET_JOB_ATTRIBUTESプロシージャを使用して、追加のジョブ属性を設定したり、属性値を変更できます。
- デタッチ済ジョブの作成
  デタッチ済ジョブでは、detached属性がTRUEに設定されているプログラム・オブジェクト(プログラム)を指し示す必要があります。
- 単一トランザクションでの複数ジョブの作成
  多数のジョブを作成する必要がある場合は、CREATE_JOBSプロシージャを使用すると、トランザクションのオーバーヘッドを減らしてパフォーマンスを改善できる可能性があります。
- 外部ジョブの手法
  この項では、次の例を通して、外部ジョブに関するいくつかの実践的手法を示します。

親トピック: ジョブの作成、実行および管理

29.2.2.1 ジョブ作成の概要

1つ以上のジョブを作成するには、DBMS_SCHEDULER.CREATE_JOBまたはDBMS_SCHEDULER.CREATE_JOBSプロシージャ、あるいはCloud Controlを使用します。

単一のジョブを作成するには、CREATE_JOBプロシージャを使用します。このプロシージャを使用して、異なるオブジェクトに基づく様々なタイプの複数のジョブを作成するとオーバーロードになります。単一トランザクションに複数のジョブを作成する場合は、CREATE_JOBSプロシージャを使用してください。

自分のスキーマにジョブを作成するにはCREATE JOB権限が、SYS以外のすべてのスキーマにジョブを作成するにはCREATE ANY JOB権限が必要です。

作成される各ジョブに対して、ジョブ・タイプ、処理およびスケジュールを指定します。必要応じて、資格証明の名前、宛先や宛先グループ名、ジョブ・クラスおよび他の属性も指定できます。ジョブを有効にするとすぐに、スケジュールされた次の日付と時刻に、スケジューラによってジョブが自動的に実行されます。デフォルトでは、作成時にジョブは無効になっており、DBMS_SCHEDULER.ENABLEで有効にして実行する必要があります。CREATE_JOBプロシージャのenabled引数をTRUEに設定することもできます。その場合、ジョブを作成するとすぐに、スケジュールに従ってジョブを自動的に実行する準備が整います。

一部のジョブ属性はCREATE_JOBでは設定できないため、かわりにDBMS_SCHEDULER.SET_ATTRIBUTEで設定する必要があります。たとえば、ジョブにlogging_level属性を設定するには、CREATE_JOBをコールした後、SET_ATTRIBUTEをコールする必要があります。

schema.job_nameを指定すると、別のスキーマ内にジョブを作成できます。したがって、ジョブの作成者がジョブの所有者であるとはかぎりません。ジョブの所有者は、ジョブが作成されるスキーマを所有しているユーザーです。実行時のジョブのNLS環境は、そのジョブが作成された時点に存在していた環境です。

次の例では、売上集計表を更新するOPSスキーマのパッケージ・プロシージャをコールする、update_salesというデータベース・ジョブの作成について説明します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name           =>  'update_sales',
   job_type           =>  'STORED_PROCEDURE',
   job_action         =>  'OPS.SALES_PKG.UPDATE_SALES_SUMMARY',
   start_date         =>  '28-APR-08 07.00.00 PM Australia/Sydney',
   repeat_interval    =>  'FREQ=DAILY;INTERVAL=2', /* every other day */
   end_date           =>  '20-NOV-08 07.00.00 PM Australia/Sydney',
   auto_drop          =>   FALSE,
   job_class          =>  'batch_update_jobs',
   comments           =>  'My new job');
END;
/
```

destination_name属性は指定されていないため、ジョブは作成元の(ローカル)データベースで実行されます。ジョブは、ジョブを作成したユーザーとして実行されます。

repeat_interval引数では、このジョブが終了日時に達するまで1日おきに実行されるように指定しています。繰返しジョブの実行回数を制限するもう1つの方法として、max_runs属性を正の数に設定する方法があります。

ジョブの作成時、デフォルトではジョブは使用禁止になっています。ジョブは、スケジューラによって自動的に実行される前に、DBMS_SCHEDULER.ENABLEで使用可能にする必要があります。

ジョブは、デフォルトでは完了後に自動的に削除されるように設定されています。auto_drop属性をFALSEに設定すると、ジョブは保持されます。繰返しジョブは、ジョブ終了日をすぎるか、最大実行回数(max_runs)または最大失敗回数

(max_failures)に達するまでは自動削除されないことに注意してください。

作成したジョブは、*_SCHEDULER_JOBSビューを使用して問い合せることができます。

関連項目:

「スケジューラ・ジョブの資格証明の指定」

親トピック: ジョブの作成

29.2.2.2 ジョブの処理、スケジュール、プログラムおよびスタイルの指定

CREATE_JOBプロシージャはオーバーロードになるため、複数の異なる使用方法があります。

「ジョブの作成の概要」の例で示しているように、ジョブの処理とジョブの繰返し間隔をジョブ属性として指定することに加えて(インラインでのジョブ処理とジョブ・スケジュールの指定とも呼ばれる)、ジョブの処理を指定するプログラム・オブジェクト(プログラム)、繰返し間隔を指定するスケジュール・オブジェクト(スケジュール)、またはプログラムとスケジュールの両方を指すジョブを作成できます。ジョブのプログラムとジョブ・スタイルを指定してジョブを作成することもできます。

- 名前付きプログラムを使用したジョブの作成
  ジョブの処理をインラインで記述するかわりに、名前付きプログラムを指し示してジョブを作成することもできます。
- 名前付きプログラムとジョブ・スタイルを使用したジョブの作成
  名前付きプログラムとジョブ・スタイルを使用してジョブを作成できます。使用できるジョブ・スタイルは、'REGULAR'、'LIGHTWEIGHT'、'IN_MEMORY_RUNTIME'および'IN_MEMORY_FULL'です。
- 名前付きスケジュールを使用したジョブの作成
  ジョブのスケジュールをインラインにするかわりに、名前付きスケジュールを指すことにより、ジョブを作成することができます。
- 名前付きプログラムとスケジュールを使用したジョブの作成
  名前付きプログラムと名前付きスケジュールの両方を指すことにより、ジョブを作成することができます。

関連項目:

- 「プログラム」
- 「スケジュール」

親トピック: ジョブの作成

29.2.2.2.1 名前付きプログラムを使用したジョブの作成

ジョブの処理をインラインで記述するかわりに、名前付きプログラムを指し示してジョブを作成することもできます。

名前付きプログラムを使用してジョブを作成するには、ジョブの作成時にCREATE_JOBプロシージャでprogram_nameの値を指定し、job_type、job_actionおよびnumber_of_argumentsの値は指定しません。

既存のプログラムを使用してジョブを作成するには、ジョブの所有者がプログラムの所有者であるか、プログラムに対するEXECUTE権限を持っている必要があります。次のPL/SQLブロックは、my_new_job1という標準ジョブを作成する名前付きプログラムが指定されたCREATE_JOBプロシージャの例です。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name           =>  'my_new_job1',
   program_name       =>  'my_saved_program',
   repeat_interval    =>  'FREQ=DAILY;BYHOUR=12',
   comments           =>  'Daily at noon');
```

```
END;
/
```

親トピック: ジョブの処理、スケジュール、プログラムおよびスタイルの指定

29.2.2.2.2 名前付きプログラムとジョブ・スタイルを使用したジョブの作成

名前付きプログラムとジョブ・スタイルを使用してジョブを作成できます。使用できるジョブ・スタイルは、'REGULAR'、'LIGHTWEIGHT'、'IN_MEMORY_RUNTIME'および'IN_MEMORY_FULL'です。

デフォルトのジョブ・スタイルは'REGULAR'で、ジョブ・スタイルが指定されない場合使用されます。他のジョブ・タイプの例を次に示します。

LIGHTWEIGHTジョブ

次のPL/SQLブロックでは、軽量ジョブが作成されます。軽量ジョブは1つのプログラムを参照する必要があり、プログラム・タイプは'PLSQL_BLOCK'または'STORED_PROCEDURE'であることが必要です。さらに、プログラムはジョブの作成時に使用可能になっている必要があります。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name         =>  'my_lightweight_job1',
   program_name     =>  'polling_prog_n2',
   repeat_interval  =>  'FREQ=SECONDLY;INTERVAL=10',
   end_date         =>  '30-APR-09 04.00.00 AM Australia/Sydney',
   job_style        => 'LIGHTWEIGHT',
   comments         => 'Job that polls device n2 every 10 seconds');
END;
/
```

IN_MEMORY_RUNTIMEジョブ

次のPL/SQLブロックはインメモリー・ランタイム・ジョブを作成します。インメモリー・ランタイム・ジョブの要件および制約は、ライトウェイト・ジョブと同じです。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name => 'my_repeat_job',
    program_name => 'repeat_prog',
    start_date => systimestamp,
    repeat_interval => 'freq=secondly;interval=10',
    job_style => 'IN_MEMORY_RUNTIME',
    enabled => true);
END;
/
```

IN_MEMORY_FULLジョブ

次のPL/SQLはインメモリー・フル・ジョブを作成します。インメモリー・フル・ジョブにはプログラムが必要であり、スケジュールまたは繰返し間隔は指定できません。ジョブが有効にされると自動的に実行され、実行後に破棄されます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name => 'my_immediate_job',
    program_name => 'fast_op',
    job_style => 'IN_MEMORY_FULL',
    enabled => true);
END;
/
```

関連項目:

「インメモリー・ジョブ」

親トピック: ジョブの処理、スケジュール、プログラムおよびスタイルの指定

29.2.2.2.3 名前付きスケジュールを使用したジョブの作成

ジョブのスケジュールをインラインにするかわりに、名前付きスケジュールを指すことにより、ジョブを作成することができます。

名前付きスケジュールを使用してジョブを作成するには、ジョブの作成時にCREATE_JOBプロシージャでschedule_nameの値を指定し、start_date、repeat_intervalおよびend_dateの値は指定しません。

すべてのスケジュールはPUBLICに対するアクセス権限付きで作成されるため、任意の名前付きスケジュールを使用してジョブを作成できます。次のCREATE_JOBプロシージャには名前付きスケジュールがあり、my_new_job2という標準ジョブを作成します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name                =>  'my_new_job2',
   job_type                =>  'PLSQL_BLOCK',
   job_action              =>  'BEGIN SALES_PKG.UPDATE_SALES_SUMMARY; END;',
   schedule_name           =>  'my_saved_schedule');
END;
/
```

親トピック: ジョブの処理、スケジュール、プログラムおよびスタイルの指定

29.2.2.2.4 名前付きプログラムとスケジュールを使用したジョブの作成

名前付きプログラムと名前付きスケジュールの両方を指すことにより、ジョブを作成することができます。

たとえば、次のCREATE_JOBプロシージャは、既存のプログラムmy_saved_program1および既存のスケジュールmy_saved_schedule1に基づいて、標準ジョブをmy_new_job3という名前で作成します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name            =>  'my_new_job3',
   program_name        =>  'my_saved_program1',
   schedule_name       =>  'my_saved_schedule1');
END;
/
```

関連項目:

- 「ジョブを定義するためのプログラムの作成および管理」
- 「ジョブを定義するためのスケジュールの作成および管理」
- 「イベントを使用したジョブの開始」

親トピック: ジョブの処理、スケジュール、プログラムおよびスタイルの指定

29.2.2.3 スケジューラ・ジョブの資格証明の指定

Oracle Schedulerには、実行前にOracle Databaseまたはオペレーティング・システムでの認証に使用するジョブの資格証明が必要です。

ローカル外部ジョブ、リモート外部ジョブおよびリモート・データベース・ジョブの場合、ジョブを実行するための資格証明を指定する

必要があります。そのためには、資格証明オブジェクトを作成して、それをcredential_nameジョブ属性に割り当てます。

ノート:

ローカル・データベース・ジョブは常にジョブ所有者のユーザーとして実行され、名前付き資格証明は無視されます。

資格証明を作成するには、DBMS_CREDENTIAL.CREATE_CREDENTIALプロシージャをコールします。

自分のスキーマに資格証明を作成するにはCREATE CREDENTIAL権限が、SYS以外のスキーマに資格証明を作成するにはCREATE ANY CREDENTIAL権限が必要です。資格証明を使用できるのは、資格証明のEXECUTE権限がジョブの所有者に付与されているジョブ、またはジョブの所有者が資格証明の所有者を兼ねているジョブのみです。資格証明は他のスキーマ・オブジェクトのようにスキーマに属しているため、資格証明に対する権限はGRANT SQL文を使用して付与します。

例29-1 資格証明の作成

```
BEGIN
  DBMS_CREDENTIAL.CREATE_CREDENTIAL('DW_CREDENTIAL', 'dwuser', 'dW001515');
END;
/
GRANT EXECUTE ON DW_CREDENTIAL TO salesuser;
```

データベース内の資格証明のリストを表示するには、*_CREDENTIALSビューを問い合せます。資格証明パスワードは不明瞭化されて格納されるため、これらのビューには表示されません。

ノート:

*_SCHEDULER_CREDENTIALS は Oracle Database 12c では非推奨ですが、下位互換性のために引き続き使用できます。

関連項目:

DBMS_CREDENTIAL.CREATE_CREDENTIALプロシージャを使用した資格証明の作成の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

親トピック: ジョブの作成

29.2.2.4 宛先の指定

リモート外部ジョブおよびリモート・データベース・ジョブの場合は、宛先オブジェクトを作成してdestination_nameジョブ属性に割り当てることにより、ジョブの宛先を指定します。NULLのdestination_name属性が指定されたジョブは、ジョブが作成されたホストで実行されます。

- 宛先のタスクとそのプロシージャ
  宛先のタスクは、DBMS_SCHEDULERパッケージのプロシージャを使用して管理します。
- 宛先の作成
  宛先とは、ジョブを実行する場所を定義したスケジューラ・オブジェクトです。
- 複数の宛先のジョブに対する宛先グループの作成
  複数の宛先で実行するジョブを作成するには、宛先グループを作成して、そのグループをジョブの

destination_name属性に割り当てる必要があります。

- 例: リモート・データベース・ジョブの作成
  リモート・データベース・ジョブを作成する例を示します。

親トピック: ジョブの作成

29.2.2.4.1 宛先のタスクとそのプロシージャ

宛先のタスクは、DBMS_SCHEDULERパッケージのプロシージャを使用して管理します。

表29-2に、宛先のタスクとそのプロシージャおよび権限を示します。

表29-2 宛先のタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
外部宛先の作成	(なし)	「宛先の作成」を参照
外部宛先の削除	DROP_AGENT_DESTINATION	MANAGE SCHEDULER
データベース宛先の作成	CREATE_DATABASE_DESTINATION	CREATE JOB または CREATE ANY JOB
データベース宛先の削除	DROP_DATABASE_DESTINATION	CREATE ANY JOB または所有者であること
宛先グループの作成	CREATE_GROUP	CREATE JOB または CREATE ANY JOB
宛先グループの削除	DROP_GROUP	CREATE ANY JOB または所有者であること
宛先グループへのメンバーの追加	ADD_GROUP_MEMBER	ALTER または CREATE ANY JOB、あるいは所有者
宛先グループからのメンバーの削除	REMOVE_GROUP_MEMBER	ALTER または CREATE ANY JOB、あるいは所有者

親トピック: 宛先の指定

29.2.2.4.2 宛先の作成

宛先とは、ジョブを実行する場所を定義したスケジューラ・オブジェクトです。

ジョブが実行される場所を指定するには、ジョブのdestination_name属性に単一の宛先または宛先グループを指定します。destination_name属性をNULLのままにすると、ジョブはローカル・ホスト(ジョブが作成されたホスト)で実行されます。

リモート外部ジョブが実行される場所を指定するには、外部宛先を使用します。リモート・データベース・ジョブが実行される場所を指定するには、データベース宛先を使用します。

別のユーザーが作成した宛先を使用するのにオブジェクト権限は必要ありません。

外部宛先を作成するには、リモート・スケジューラ・エージェントをデータベースに登録します。

手順は、「リモート・ホストでのスケジューラ・エージェントのインストールと構成」を参照してください。

ノート:

外部宛先を作成するための DBMS_SCHEDULER パッケージ・プロシージャはありません。外部宛先は、リモート・エージェントを登録することによって暗黙的に作成します。

リモート・ジョブのターゲットになるホストで他のデータベース・インスタンスを実行している場合は、ローカル・スケジューラ・エージェントも登録できます。この場合、ローカル・ホストを参照する外部宛先が作成されます。

外部宛先の名前は自動的にエージェント名に設定されます。外部宛先が作成されたかどうかを確認するには、DBA_SCHEDULER_EXTERNAL_DESTSビューまたはALL_SCHEDULER_EXTERNAL_DESTSビューを問い合せます。

データベース宛先を作成するには、DBMS_SCHEDULER.CREATE_DATABASE_DESTINATIONプロシージャをコールします。

プロシージャの引数として外部宛先の名前を指定する必要があります。これにより、データベース宛先が指し示すリモート・ホストが指定されます。また、ネット・サービス名を指定したり、接続先のデータベース・インスタンスを識別する接続記述子を入力します。ネット・サービス名を指定した場合、その名前はローカルのtnsnames.oraファイルで解決される必要があります。データベース・インスタンスを指定しなかった場合、リモート・スケジューラ・エージェントは、エージェント構成ファイルに指定されているデフォルトのデータベースに接続されます。

データベース宛先を作成するには、CREATE JOBシステム権限が必要です。自身のスキーマ以外のスキーマにデータベース接続先を作成するには、CREATE ANY JOB権限が必要です。

例29-2 データベース宛先の作成

次の例では、DBHOST1_ORCLDWという名前のデータベース宛先を作成しています。この例では、次のことを想定しています。

- リモート・ホストdbhost1.example.comにスケジューラ・エージェントをインストールし、そのエージェントをローカル・データベースに登録していること。
- エージェント構成ファイル内のエージェント名の設定を変更していないこと。つまり、エージェント名および外部宛先の名前がデフォルトのDBHOST1に設定されていること。
- ローカル・ホスト上のNet Configuration Assistantを使用して、リモート・ホストdbhost1.example.comに存在するorcldwという名前のOracle Databaseインスタンスに対してtnsnames.ora内に接続記述子を作成していること。また、ORCLDWのネット・サービス名(別名)をこの接続記述子に割り当てていること。

```
BEGIN
 DBMS_SCHEDULER.CREATE_DATABASE_DESTINATION (
  destination_name     => 'DBHOST1_ORCLDW',
  agent                => 'DBHOST1',
  tns_name             => 'ORCLDW',
  comments             => 'Instance named orcldw on host dbhost1.example.com');
END;
/
```

データベース宛先が作成されたかどうかを確認するには、*_SCHEDULER_DB_DESTSビューを問い合せます。

関連項目:

- 宛先の詳細は、「宛先」
- リモート外部ジョブとリモート・データベース・ジョブについて学習するには、「ジョブ」

親トピック: 宛先の指定

29.2.2.4.3 複数の宛先のジョブに対する宛先グループの作成

複数の宛先で実行するジョブを作成するには、宛先グループを作成して、そのグループをジョブのdestination_name属性に割り当てる必要があります。

グループの作成時にグループ・メンバー(宛先)を指定することも、後でグループ・メンバーを追加することもできます。

宛先グループを作成するには、DBMS_SCHEDULER.CREATE_GROUPプロシージャをコールします。

リモート外部ジョブの場合、EXTERNAL_DESTタイプのグループを指定し、すべてのグループ・メンバーを外部宛先にする必要があります。リモート・データベース・ジョブの場合、DB_DESTタイプのグループを指定し、すべてのメンバーをデータベース宛先にする必要があります。

宛先グループのメンバーは、次の形式で指定する必要があります。

```
[[schema.]credential@][schema.]destination
```

ここで:

- credentialは、既存の資格証明の名前です。
- destinationは、既存のデータベース宛先または外部宛先の名前です。

接続先メンバーの資格証明部分はオプションです。省略すると、この接続先メンバーを使用するジョブはデフォルトの資格証明を使用します。

同じタイプの別のグループを宛先グループのメンバーとして含めることができます。グループの作成時に別のグループを含めた場合、その中に含まれるメンバーはスケジューラによって自動的に展開されます。

ジョブを実行する多数の宛先の1つとしてローカル・ホストを使用する場合、どちらのタイプの宛先グループのグループ・メンバーとしてもLOCALキーワードを指定できます。外部宛先グループの場合にかぎり、LOCALの前に資格証明の接頭辞を付けることができます。

グループの所有者は、そのグループを作成したユーザーです。自分のスキーマにグループを作成するにはCREATE JOBシステム権限が必要になり、別のスキーマにグループを作成するにはCREATE ANY JOBシステム権限が必要になります。SELECTをグループに付与することによって、グループのオブジェクト権限を他のユーザーに付与できます。

関連項目:

グループの概要については、「グループ」を参照してください。

例29-3 データベース宛先グループの作成

次の例では、データベース宛先グループを作成しています。一部のメンバーには資格証明がないため、この宛先グループを使用するジョブではデフォルトの資格証明を使用する必要があります。

```
BEGIN
  DBMS_SCHEDULER.CREATE_GROUP(
    GROUP_NAME    => 'all_dbs',
    GROUP_TYPE    => 'DB_DEST',
    MEMBER        => 'oltp_admin@orcl, orcldw1, LOCAL',
    COMMENTS      => 'All databases managed by me');
```

```
END;
/
```

次のコードでは、グループに別のメンバーを追加しています。

```
BEGIN
  DBMS_SCHEDULER.ADD_GROUP_MEMBER(
    GROUP_NAME    => 'all_dbs',
    MEMBER        => 'dw_admin@orcldw2');
END;
/
```

親トピック: 宛先の指定

29.2.2.4.4 例: リモート・データベース・ジョブの作成

リモート・データベース・ジョブを作成する例を示します。

次の例では、ジョブのdestination_nameオブジェクトにデータベース宛先オブジェクトを指定することで、リモート・データベース・ジョブを作成しています。リモート・データベースでジョブを認証できるように、資格証明も指定する必要があります。この例では、例29-1で作成した資格証明および例29-2で作成したデータベース宛先を使用しています。

```
BEGIN
 DBMS_SCHEDULER.CREATE_JOB (
   job_name            =>  'SALES_SUMMARY1',
   job_type            =>  'STORED_PROCEDURE',
   job_action          =>  'SALES.SALES_REPORT1',
   start_date          =>  '15-JUL-09 11.00.00 PM Europe/Warsaw',
   repeat_interval     =>  'FREQ=DAILY',
   credential_name     =>  'DW_CREDENTIAL',
   destination_name    =>  'DBHOST1_ORCLDW');
END;
/
```

親トピック: 宛先の指定

29.2.2.5 複数の宛先のジョブの作成

複数の宛先で実行し、1箇所で管理するジョブを作成できます。

このようにする代表的な理由としては、管理対象のすべてのデータベースでデータベース・メンテナンス・ジョブを実行することがあげられます。各データベースでジョブを作成するのではなく、ジョブを1つ作成し、そのジョブに対して複数の宛先を指定します。ジョブを作成したデータベース(local database)から、すべての場所のジョブのすべてのインスタンスの状態と結果を監視できます。

複数の宛先のジョブを作成するには:

- DBMS_SCHEDULER.CREATE_JOBプロシージャをコールして、ジョブのdestination_name属性にデータベース宛先グループまたは外部宛先グループの名前を設定します。

  一部の宛先グループ・メンバーに資格証明の接頭辞(スキーマ)が付いていない場合、ジョブにデフォルトの資格証明を割り当てます。

  ジョブを実行する宛先の1つとしてローカル・ホストまたはローカル・データベースを使用するには、LOCALキーワードを宛先グループのメンバーの1つにする必要があります。

宛先グループのリストを取得するには、次の問合せを発行します。

```
SELECT owner, group_name, group_type, number_of_members FROM all_scheduler_groups
  WHERE group_type = 'DB_DEST' or group_type = 'EXTERNAL_DEST';
```

```
OWNER           GROUP_NAME      GROUP_TYPE    NUMBER_OF_MEMBERS
--------------- --------------- ------------ -----------------
DBA1            ALL_DBS         DB_DEST                      4
DBA1            ALL_HOSTS       EXTERNAL_DEST                4
```

次の例では、例29-3で作成したデータベース宛先グループを使用して、複数の宛先のデータベース・ジョブを作成しています。資格証明に指定したユーザーには、ジョブ・アクションを実行する十分な権限が必要です。

```
BEGIN
 DBMS_CREDENTIAL.CREATE_CREDENTIAL('DBA_CREDENTIAL', 'dba1', 'sYs040533');
 DBMS_SCHEDULER.CREATE_JOB (
   job_name            =>  'MAINT_SET1',
   job_type            =>  'STORED_PROCEDURE',
   job_action          =>  'MAINT_PROC1',
   start_date          =>  '15-JUL-09 11.00.00 PM Europe/Warsaw',
   repeat_interval     =>  'FREQ=DAILY',
   credential_name     =>  'DBA_CREDENTIAL',
   destination_name    =>  'ALL_DBS');
END;
/
```

関連項目:

- 「複数の宛先のジョブ」
- 「複数の宛先のジョブの監視」
- 「グループ」

親トピック: ジョブの作成

29.2.2.6 ジョブ引数の設定

ジョブ引数を設定するには、SET_JOB_ARGUMENT_VALUEまたはSET_JOB_ANYDATA_VALUEプロシージャ、あるいはCloud Controlを使用します。SET_JOB_ANYDATA_VALUEは、VARCHAR2文字列として表現できない複雑なデータ型に使用されます。

ジョブの作成後、次の場合にジョブ引数の設定が必要な場合があります。

- インラインのジョブ処理が、引数を必要とするストアド・プロシージャまたはその他の実行可能ファイルの場合
- ジョブが名前付きプログラム・オブジェクトを参照していて、そのデフォルト・プログラム引数の1つ以上を上書きする場合
- ジョブが名前付きプログラム・オブジェクトを参照していて、そのプログラム引数の1つ以上にデフォルト値が割り当てられていない場合

引数を必要とする場合があるジョブの例の1つに、開始日と終了日が必要なレポート作成プログラムを開始するジョブがあります。次のコード例では、レポート作成プログラムの2番目の引数である終了日のジョブ引数が設定されます。

```
BEGIN
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE (
   job_name                => 'ops_reports',
   argument_position       => 2,
   argument_value          => '12-DEC-03');
END;
/
```

値がすでに設定されている引数についてこのプロシージャを使用すると、その引数は上書きされます。引数値を設定するには、引数名または引数の位置を使用します。引数名を使用するには、ジョブが名前付きプログラム・オブジェクトを参照し、そのプロ

グラム・オブジェクト内で引数に名前が割り当てられている必要があります。プログラムがインラインで記述されている場合にサポートされるのは、位置による設定のみです。引数は、タイプがPLSQL_BLOCKのジョブに対してはサポートされません。

設定されている値を削除するには、RESET_JOB_ARGUMENTプロシージャを使用します。このプロシージャは、通常の引数およびANYDATA引数の両方に使用できます。

SET_JOB_ARGUMENT_VALUEでは、SQLタイプの引数のみがサポートされます。したがって、SQLタイプではない引数の値(ブール値など)は、プログラムまたはジョブ引数としてサポートされません。

関連項目:

「プログラム引数の定義」

親トピック: ジョブの作成

29.2.2.7 ジョブ属性の追加設定

ジョブの作成後に、SET_ATTRIBUTEまたはSET_JOB_ATTRIBUTESプロシージャを使用して、追加のジョブ属性を設定したり、属性値を変更できます。

また、Cloud Controlを使用してジョブ属性を設定することもできます。ジョブ属性の多くはCREATE_JOBのコールを使用して設定できますが、destinationやcredential_nameなどの一部の属性は、ジョブの作成後にSET_ATTRIBUTEまたはSET_JOB_ATTRIBUTESを使用することによってのみ設定できます。

親トピック: ジョブの作成

29.2.2.8 デタッチ済ジョブの作成

デタッチ済ジョブでは、detached属性がTRUEに設定されているプログラム・オブジェクト(プログラム)を指し示す必要があります。

次のLinuxとUNIXの例では、データベースのコールド・バックアップを実行する夜間ジョブを作成しています。これには、3つのステップが含まれています。

ステップ1—RMAN起動スクリプトの作成

コールド・バックアップを実行するRMANスクリプトを呼び出すシェル・スクリプトを作成します。このシェル・スクリプトはORACLE_HOME/scripts/coldbackup.shに置かれます。これは、Oracle Databaseをインストールしたユーザー(通常はユーザーoracle)が実行できる必要があります。

```
#!/bin/sh

export ORACLE_HOME=/u01/app/oracle/product/database_release_number/db_1
export ORACLE_SID=orcl
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib

$ORACLE_HOME/bin/rman TARGET / @$ORACLE_HOME/scripts/coldbackup.rman
  trace /u01/app/oracle/backup/coldbackup.out &
exit 0
```

ステップ2—RMANスクリプトの作成

コールド・バックアップを実行してからジョブを終了する、RMANスクリプトを作成します。このスクリプトはORACLE_HOME/scripts/coldbackup.rmanに置かれます。

```
run {
# Shut down database for backups and put into MOUNT mode
shutdown immediate
```

```
startup mount

# Perform full database backup
backup full format "/u01/app/oracle/backup/%d_FULL_%U" (database) ;

# Open database after backup
alter database open;

# Call notification routine to indicate job completed successfully
sql " BEGIN  DBMS_SCHEDULER.END_DETACHED_JOB_RUN(''sys.backup_job'', 0,
  null); END; ";
}
```

ステップ3ージョブの作成とデタッチ済プログラムの使用

次のPL/SQLブロックを発行します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_PROGRAM(
    program_name   => 'sys.backup_program',
    program_type   => 'executable',
    program_action => '?/scripts/coldbackup.sh',
    enabled        =>  TRUE);
  DBMS_SCHEDULER.SET_ATTRIBUTE('sys.backup_program', 'detached', TRUE);

  DBMS_SCHEDULER.CREATE_JOB(
    job_name        => 'sys.backup_job',
    program_name    => 'sys.backup_program',
    repeat_interval => 'FREQ=DAILY;BYHOUR=1;BYMINUTE=0');
  DBMS_SCHEDULER.ENABLE('sys.backup_job');
END;
/
```

関連項目:

「デタッチされたジョブ」

親トピック: ジョブの作成

29.2.2.9 単一トランザクションでの複数ジョブの作成

多数のジョブを作成する必要がある場合は、CREATE_JOBSプロシージャを使用すると、トランザクションのオーバーヘッドを減らしてパフォーマンスを改善できる可能性があります。

例29-4に、このプロシージャを使用して単一トランザクションで複数のジョブを作成する方法を示します。

例29-4 単一トランザクションでの複数ジョブの作成

```
DECLARE
 newjob sys.job_definition;
 newjobarr sys.job_definition_array;
BEGIN
 -- Create an array of JOB_DEFINITION object types
 newjobarr := sys.job_definition_array();
 -- Allocate sufficient space in the array
 newjobarr.extend(5);
 -- Add definitions for 5 jobs
 FOR i IN 1..5 LOOP
   -- Create a JOB_DEFINITION object type
   newjob := sys.job_definition(job_name => 'TESTJOB' || to_char(i),
                     job_style => 'REGULAR',
                     program_name => 'PROG1',
                     repeat_interval => 'FREQ=HOURLY',
```

```
                    start_date => systimestamp + interval '600' second,
                    max_runs => 2,
                    auto_drop => FALSE,
                    enabled => TRUE
                   );
   -- Add it to the array
   newjobarr(i) := newjob;
 END LOOP;
 -- Call CREATE_JOBS to create jobs in one transaction
 DBMS_SCHEDULER.CREATE_JOBS(newjobarr, 'TRANSACTIONAL');
END;
/
PL/SQL procedure successfully completed.
SELECT JOB_NAME FROM USER_SCHEDULER_JOBS;

JOB_NAME
------------------------------
TESTJOB1
TESTJOB2
TESTJOB3
TESTJOB4
TESTJOB5

5 rows selected.
```

関連項目:

「軽量ジョブ」

親トピック: ジョブの作成

29.2.2.10 外部ジョブの手法

この項では、次の例を通して、外部ジョブに関するいくつかの実践的手法を示します。

例29-5 コマンド・インタプリタを実行するローカル外部ジョブの作成

次の例に、Windows上でインタプリタ・コマンド(この例ではmkdir)を実行するローカル外部ジョブの作成方法を示します。このジョブでは、/cオプションを指定してcmd.exeが実行されます。

```
BEGIN
 DBMS_SCHEDULER.CREATE_JOB(
   job_name             => 'MKDIR_JOB',
   job_type             => 'EXECUTABLE',
   number_of_arguments  => 3,
   job_action           => '¥windows¥system32¥cmd.exe',
   auto_drop            => FALSE,
   credential_name      => 'TESTCRED');
 DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('mkdir_job',1,'/c');
 DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('mkdir_job',2,'mkdir');
 DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('mkdir_job',3,'¥temp¥extjob_test_dir');
 DBMS_SCHEDULER.ENABLE('MKDIR_JOB');
END;
/
```

例29-6 ローカル外部ジョブの作成とジョブ出力の表示

LinuxおよびUNIX用のこの例では、ローカル外部ジョブを作成および実行してから、ジョブ出力を表示する方法を示します。外部ジョブが実行されると、スケジューラはジョブから出力を自動的に取得し、データベース内に格納します。

結果を確認するには、 *_SCHEDULER_JOB_RUN_DETAILSビューを問い合せます。

```
-- User scott must have CREATE JOB, CREATE CREDENTIAL, and CREATE EXTERNAL JOB
-- privileges
GRANT CREATE JOB, CREATE EXTERNAL JOB TO scott ;

CONNECT scott/password
SET SERVEROUTPUT ON

-- Create a credential for the job to use
exec DBMS_CREDENTIAL.CREATE_CREDENTIAL('my_cred','host_username','host_passwd')

-- Create a job that lists a directory. After running, the job is dropped.
BEGIN
 DBMS_SCHEDULER.CREATE_JOB(
  job_name            => 'lsdir',
  job_type            => 'EXECUTABLE',
  job_action          => '/bin/ls',
  number_of_arguments => 1,
  enabled             => false,
  auto_drop           => true,
  credential_name     => 'my_cred');
 DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('lsdir',1,'/tmp');
 DBMS_SCHEDULER.ENABLE('lsdir');
END;
/

-- Wait a bit for the job to run, and then check the job results.
SELECT job_name, status, error#, actual_start_date, additional_info
 FROM user_scheduler_job_run_details WHERE job_name='LSDIR';

-- Now use the external log id from the additional_info column to
-- formulate the log file name and retrieve the output
DECLARE
 my_clob clob;
 log_id varchar2(50);
BEGIN
 SELECT regexp_substr(additional_info,'job[_0-9]*') INTO log_id
   FROM user_scheduler_job_run_details WHERE job_name='LSDIR';
 DBMS_LOB.CREATETEMPORARY(my_clob, false);
SELECT job_name, status, error#, errors, output FROM user_scheduler_job_run_details
WHERE job_name = 'LSDIR';
END;
/
```

関連項目:

- 外部認証の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- 「外部ジョブ」
- 「外部ジョブの停止」
- 「リモート・ジョブのトラブルシューティング」

親トピック: ジョブの作成

29.2.3 ジョブの変更

ジョブを変更するには、その属性を変更します。そのためには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTE、SET_ATTRIBUTE_NULLまたはSET_JOB_ATTRIBUTESプロシージャまたはCloud Controlを使用します。

ジョブ属性の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のCREATE_JOBプロシージャに関す

る項を参照してください。

すべてのジョブは変更可能で、ジョブ名以外のすべてのジョブ属性を変更できます。変更する際に実行中のジョブ・インスタンスがある場合、そのインスタンスはこのコールによる影響を受けません。このプロシージャによる変更は、今後実行されるジョブに対してのみ反映されます。

通常、データベースによって自動的に作成されたジョブは変更しないでください。データベースによって作成されたジョブには、ジョブのビューでTRUEに設定された列SYSTEMがあります。ジョブの属性は、*_SCHEDULER_JOBSビューで使用可能です。

実行中のジョブのジョブ属性も変更できます。ただし、その変更は、スケジュールされている次回のジョブ実行時まで反映されません。

SET_ATTRIBUTE、SET_ATTRIBUTE_NULLおよびSET_JOB_ATTRIBUTESプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

次の例では、ジョブupdate_salesのrepeat_intervalを毎週水曜日の1回に変更しています。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
   name          =>  'update_sales',
   attribute     =>  'repeat_interval',
   value         =>  'freq=weekly; byday=wed');
END;
/
```

親トピック: ジョブの作成、実行および管理

29.2.4 ジョブの実行

いくつかの異なる方法でジョブを実行できます。

ジョブを実行するには、次の3つの方法があります。

- ジョブ・スケジュールに従って実行する方法—この場合、ジョブが有効であれば、スケジューラ・ジョブ・コーディネータによって自動的にジョブが取り出され、ジョブ・スレーブの制御下で実行されます。ジョブは、ジョブ所有者のユーザーとして、または資格証明のあるローカル外部ジョブの場合は、資格証明で指定したユーザーとして実行されます。ジョブが成功したかどうかを確認するには、ジョブ・ビュー(*_SCHEDULER_JOBS)またはジョブ・ログ(*_SCHEDULER_JOB_LOGおよび*_SCHEDULER_JOB_RUN_DETAILS)で問い合せる必要があります。ジョブ・スレーブおよびスケジューラ・アーキテクチャの詳細は、「ジョブの実行方法」を参照してください。
- イベントの発生時に実行する方法—イベント・キューで特定のイベントが受信されるか、File Watcherでファイル到着イベントが呼び出されると、使用可能になっているイベントベースのジョブが開始されます（「イベントを使用したジョブの開始」を参照)。(「イベントを使用したジョブの開始」を参照してください。)または、資格証明を使用するローカル外部ジョブの場合は、資格証明内に指定されているユーザーとして実行されます。ジョブが成功したかどうかを確認するには、ジョブ・ビューまたはジョブ・ログを問い合せる必要があります。
- DBMS_SCHEDULER.RUN_JOBを呼び出して実行する方法—RUN_JOBプロシージャを使用して、ジョブをテストしたり、指定したスケジュール以外で実行できます。ジョブは非同期に実行できますが、これは前の2つのジョブ実行方法、つまり同期して実行する方法に類似しており、この方法ではRUN_JOBを呼び出したセッションにユーザーがログインするとジョブが実行されます。RUN_JOBのuse_current_session引数によって、ジョブが同期または非同期のどちらで実行されるかが決まります。

  RUN_JOBでは、ジョブ名のカンマ区切りのリストを使用できます。

次の例では、2つのジョブを非同期で実行しています。

```
BEGIN
  DBMS_SCHEDULER.RUN_JOB(
    JOB_NAME            => 'DSS.ETLJOB1, DSS.ETLJOB2',
    USE_CURRENT_SESSION => FALSE);
END;
/
```

ノート:

スケジュールに従ってジョブを実行する場合は、RUN_JOB をコールする必要はありません。ジョブが使用可能であれば、スケジューラによって自動的に実行されます。

親トピック: ジョブの作成、実行および管理

29.2.5 ジョブの停止

1つ以上の実行中のジョブを停止するには、DBMS_SCHEDULERパッケージのSTOP_JOBプロシージャまたはCloud Controlを使用します。

STOP_JOBでは、ジョブ、ジョブ・クラスおよびジョブ宛先IDのカンマ区切りリストを使用できます。ジョブ宛先IDは、スケジューラによって割り当てられる番号で、ジョブ、資格証明および宛先の一意の組合せを表します。これは、複数宛先のジョブの特定の子ジョブを識別して、その子のみを停止するのに便利な方法です。子ジョブのジョブ宛先IDは*_SCHEDULER_JOB_DESTSのビューから取得します。

ジョブ・クラスが指定されている場合、そのジョブ・クラス内の実行中のジョブはすべて停止されます。たとえば、次の文では、ジョブjob1、ジョブ・クラスdw_jobs内のすべてのジョブ、および複数の宛先のジョブに含まれる2つの子ジョブが停止されます。

```
BEGIN
  DBMS_SCHEDULER.STOP_JOB('job1, sys.dw_jobs, 984, 1223');
END;
/
```

指定したジョブのインスタンスはすべて停止されます。ジョブの停止後、1回かぎりのジョブの状態はSTOPPEDに設定され、繰返しジョブの状態は、(次回のジョブ実行がスケジュールされているため)SCHEDULEDに設定されます。また、ジョブ・ログには、OPERATIONがSTOPPEDに設定され、ADDITIONAL_INFOがREASON="Stop job called by user: username"に設定されたエントリが作成されます。

デフォルトでは、スケジューラは割込みメカニズムを使用してジョブを正常に停止しようとします。この方法では、制御がスレーブ・プロセスに戻され、スレーブ・プロセスはジョブ実行の統計を収集できます。forceオプションがTRUEに設定されている場合、ジョブは即時に停止するため、そのジョブ実行について特定の実行時間の統計が使用できない場合があります。

チェーンを実行中のジョブを停止すると、実行中のすべてのステップが(各ステップでforceオプションをTRUEに設定してSTOP_JOBをコールすることによって)自動的に停止されます。

STOP_JOBのcommit_semantics引数を使用すると、複数のジョブが指定されている場合の結果および1つ以上のジョブを停止しようとしたときに発生するエラーを制御できます。この引数をABSORB_ERRORSに設定すると、エラーが発生した後もプロシージャを続行して、残りのジョブの停止試行を実行できる場合があります。エラーが発生したことがプロシージャによって示された場合は、エラーの内容を判断するために、ビューSCHEDULER_BATCH_ERRORSを問い合せることができます。コミット・セマンティクスの詳細は、「ジョブの削除」を参照してください。

STOP_JOBプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ノート:

ジョブが停止したときにロールバックされるのは、現行トランザクションのみです。これは、データの一貫性を損なう原因になる場合があります。

親トピック: ジョブの作成、実行および管理

29.2.6 外部ジョブの停止

スケジューラを使用している場合、外部ジョブの実装者は、forceがFALSEに設定されたSTOP_JOBがコールされた場合に、その外部ジョブを正常にクリーン・アップするメカニズムを利用できます。

この項で説明するメカニズムは、UNIXおよびLinuxプラットフォーム上のリモート外部ジョブにのみ適用されます。

UNIXおよびLinuxでは、スケジューラによって起動されたプロセスにSIGTERMシグナルが送信されます。外部ジョブの実装者は、割込みハンドラにSIGTERMをトラップし、ジョブの実装内容をすべてクリーン・アップして終了する必要があります。

Windowsでは、forceがFALSEに設定されたSTOP_JOBがサポートされます。スケジューラによって起動されるプロセスがコンソール・プロセスです。このプロセスを停止するために、スケジューラはCTRL+BREAKをプロセスに送信します。CTRL+BREAKは、SetConsoleCtrlHandler()ルーチンにハンドラを登録することによって処理できます。

親トピック: ジョブの作成、実行および管理

29.2.7 チェーン・ジョブの停止

実行中のチェーンを指しているジョブが停止すると、実行中のチェーンのステップがすべて停止します。

個々のチェーン・ステップの停止の詳細は、「個々のチェーン・ステップの停止」を参照してください。

親トピック: ジョブの作成、実行および管理

29.2.8 ジョブの削除

1つ以上のジョブを削除するには、DBMS_SCHEDULERパッケージのDROP_JOBプロシージャまたはCloud Controlを使用します。

DROP_JOBでは、ジョブおよびジョブ・クラスのカンマ区切りリストを使用できます。ジョブ・クラスを指定すると、そのジョブ・クラスのすべてのジョブは削除されますが、ジョブ・クラス自体は削除されません。DROP_JOBでジョブ宛先IDを使用して複数宛先ジョブの子を削除することはできません。

ジョブ・クラスを削除するには、DROP_JOB_CLASSプロシージャを使用します(「ジョブ・クラスの削除」を参照)。

次の文では、ジョブjob1とjob3、およびジョブ・クラスjobclass1とjobclass2内のすべてのジョブが削除されます。

```
BEGIN
  DBMS_SCHEDULER.DROP_JOB ('job1, job3, sys.jobclass1, sys.jobclass2');
END;
/
```

親トピック: ジョブの作成、実行および管理

29.2.9 実行中のジョブの削除

DROP_JOBプロシージャが呼び出されたときにジョブが実行中である場合、ジョブを削除しようとすると失敗します。このデフォルトの動作は、forceまたはdeferオプションを設定することによって変更できます。

forceオプションをTRUEに設定すると、最初に、実行中のジョブの停止試行がスケジューラの割込みメカニズム(forceオプションをFALSEに設定してSTOP_JOBをコール)によって行われます。ジョブが正常に停止されると、次にそのジョブが削除されます。または、STOP_JOBを最初にコールしてジョブを停止してから、DROP_JOBをコールすることもできます。STOP_JOBが失敗した場合は、forceオプションを設定してSTOP_JOBをコールできます(この場合、MANAGE SCHEDULER権限が必要です)。その後、そのジョブを削除できます。デフォルトでは、STOP_JOBおよびDROP_JOBプロシージャの両方で、forceはFALSEに設定されています。

deferオプションをTRUEに設定すると、実行中のジョブは完了してから削除されます。forceとdeferオプションは相互に排他的で、両方を設定するとエラーになります。

親トピック: ジョブの作成、実行および管理

29.2.10 複数のジョブの削除

削除する複数のジョブを指定したとき、いずれかのジョブでエラーが発生した場合、その結果はDBMS_SCHEDULER.DROP_JOBプロシージャのcommit_semantics引数によって決定されます。

この引数には、次の値を指定できます。

- STOP_ON_FIRST_ERROR(デフォルト)ー最初のエラーでコールが戻り、エラー発生前に正常終了した削除操作がディスクにコミットされます。
- TRANSACTIONALー最初のエラーでコールが戻り、エラー発生前の削除操作がロールバックされます。forceがFALSEに設定されている必要があります。
- ABSORB_ERRORSーエラーへの対応が取り組まれ、残りのジョブの削除が試行されて、正常終了したすべての削除操作がコミットされます。

commit_semanticsを設定できるのは、job_nameリストにジョブ・クラスが含まれていない場合のみです。ジョブ・クラスを指定している場合は、デフォルトのコミット・セマンティクス(STOP_ON_FIRST_ERROR)が適用されます。

次の例では、deferオプションとTRANSACTIONALコミット・セマンティクスを使用して、ジョブmyjob1およびmyjob2を削除しています。

```
BEGIN
  DBMS_SCHEDULER.DROP_JOB(
     job_name         => 'myjob1, myjob2',
     defer            => TRUE,
     commit_semantics => 'TRANSACTIONAL');
END;
/
```

次の例では、ABSORB_ERRORSコミット・セマンティクスを示しています。プロシージャのコール時にmyjob1が実行されており、myjob2は実行されていないことを想定しています。

```
BEGIN
  DBMS_SCHEDULER.DROP_JOB(
     job_name         => 'myjob1, myjob2',
     commit_semantics => 'ABSORB_ERRORS');
END;
/
Error report:
```

```
ORA-27362: batch API call completed with errors
```

SCHEDULER_BATCH_ERRORSビューを問い合せて、エラーの内容を判断できます。

```
SELECT object_name, error_code, error_message FROM scheduler_batch_errors;
OBJECT_NAME    ERROR CODE ERROR_MESSAGE
-------------- ---------- --------------------------------------------------
STEVE.MYJOB1        27478 "ORA-27478: job "STEVE.MYJOB1" is running
```

USER_SCHEDULER_JOBSを確認すると、myjob2は正常に削除され、myjob1は残っていることがわかります。

DROP_JOBプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブの作成、実行および管理

29.2.11 ジョブの無効化

1つ以上のジョブを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャまたはCloud Controlを使用します。

ジョブは他の方法で使用禁止にすることもできます。たとえば、ジョブ・クラスを削除すると、そのクラスのジョブが使用禁止になります。ジョブが指し示しているプログラムまたはスケジュールのいずれかを削除した場合も、使用禁止になります。ただし、ジョブが指し示しているプログラムまたはスケジュールを使用禁止にしてもジョブは使用禁止にならないため、スケジューラがジョブを実行しようとしたときにエラーが発生します。

ジョブを使用禁止にすると、そのジョブのメタデータはそのまま存在しますが、ジョブ自体は実行対象ではないため、ジョブ・コーディネータがこれらのジョブを取り出して処理することはありません。ジョブが使用禁止になると、ジョブ表内のそのstateは disabledに変更されます。

forceオプションをFALSEに設定して現在実行中のジョブを使用禁止にすると、エラーが返されます。forceがTRUEに設定されているときは、ジョブは使用禁止になりますが、現在実行中のインスタンスは完了できます。

commit_semanticsがSTOP_ON_FIRST_ERRORに設定されている場合は、最初のエラーでコールが戻り、エラー発生前に正常終了した使用禁止操作がディスクにコミットされます。commit_semanticsがTRANSACTIONALに、forceがFALSEに設定されている場合は、最初のエラーでコールが戻り、エラー発生前の使用禁止操作がロールバックされます。commit_semanticsがABSORB_ERRORSに設定されている場合は、エラーへの対応が取り組まれ、残りのジョブの使用禁止が試行されて、正常に終了したすべての使用禁止操作がコミットされます。エラーが発生したことがプロシージャによって示された場合は、エラーの内容を判断するために、ビューSCHEDULER_BATCH_ERRORSを問い合せることができます。

デフォルトでは、commit_semanticsはSTOP_ON_FIRST_ERRORに設定されています。

DISABLEプロシージャ・コールにジョブ名またはジョブ・クラス名のカンマ区切りのリストを指定することで、1回のコールで複数のジョブを使用禁止にすることもできます。たとえば、次の文では、ジョブとジョブ・クラスを組み合せて指定しています。

```
BEGIN
  DBMS_SCHEDULER.DISABLE('job1, job2, job3, sys.jobclass1, sys.jobclass2');
END;
/
```

DISABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブの作成、実行および管理

29.2.12 ジョブの有効化

1つ以上のジョブを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャまたはCloud Controlを使用しま

す。

このプロシージャを使用すると、ジョブは、ジョブ・コーディネータによって取り出され、処理されるようになります。ジョブは、デフォルトで使用禁止で作成されるため、実行するには使用可能にする必要があります。ジョブを使用可能にすると、妥当性チェックが実行されます。チェックに失敗すると、ジョブは使用可能になりません。

使用禁止のジョブを使用可能にすると、ジョブはスケジュールに従って実行を即時開始します。また、使用禁止のジョブを使用可能にすると、ジョブのRUN_COUNT、FAILURE_COUNTおよびRETRY_COUNT属性が再設定されます。

commit_semanticsがSTOP_ON_FIRST_ERRORに設定されている場合は、最初のエラーでコールが戻り、エラー発生前に正常終了した使用可能にする操作がディスクにコミットされます。commit_semanticsがTRANSACTIONALに設定されている場合は、最初のエラーでコールが戻り、エラー発生前の使用可能にする操作がロールバックされます。commit_semanticsがABSORB_ERRORSに設定されている場合は、エラーへの対応が取り組まれ、残りのジョブを使用可能にする操作が試行されて、正常に終了した使用可能にする操作がすべてコミットされます。エラーが発生したことがプロシージャによって示された場合は、エラーの内容を判断するために、ビューSCHEDULER_BATCH_ERRORSを問い合せることができます。

デフォルトでは、commit_semanticsはSTOP_ON_FIRST_ERRORに設定されています。

ENABLEプロシージャ・コールにジョブ名またはジョブ・クラス名のカンマ区切りのリストを指定することで、1回のコールで複数のジョブを使用可能にすることもできます。たとえば、次の文では、ジョブとジョブ・クラスを組み合せて指定しています。

```
BEGIN
 DBMS_SCHEDULER.ENABLE ('job1, job2, job3,
   sys.jobclass1, sys.jobclass2, sys.jobclass3');
END;
/
```

ENABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブの作成、実行および管理

29.2.13 ジョブのコピー

ジョブをコピーするには、DBMS_SCHEDULERのCOPY_JOBプロシージャまたはCloud Controlを使用します。

このコールによって、旧ジョブのすべての属性(ジョブ名以外)が新規ジョブにコピーされます。新規ジョブは使用禁止の状態で作成されます。

COPY_JOBプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブの作成、実行および管理

29.3 ジョブを定義するためのプログラムの作成および管理

プログラムは、特定のタスクに関するメタデータの集まりです。必要に応じて、プログラムを使用してジョブの定義を支援できます。

- プログラムのタスクとそのプロシージャ
  DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なプログラム・タスクを管理します。
- スケジューラを使用するプログラムの作成
  プログラムでは、スケジューラによって実行される内容が記述されます。
- プログラムの変更
  プログラムを変更するには、その属性を変更します。プログラムを変更するには、Cloud ControlまたはDBMS_SCHEDULER.SET_ATTRIBUTEおよびDBMS_SCHEDULER.SET_ATTRIBUTE_NULLパッケージ・プロ

シージャを使用します。

- プログラムの削除
  1つ以上のプログラムを削除するには、DBMS_SCHEDULERパッケージのDROP_PROGRAMプロシージャまたはCloud Controlを使用します。
- プログラムの無効化
  1つ以上のプログラムを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャまたはCloud Controlを使用します。
- プログラムの有効化
  1つ以上のプログラムを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャまたはCloud Controlを使用します。

関連項目:

プログラムの概要については、「プログラム」を参照してください。

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.3.1 プログラムのタスクとそのプロシージャ

DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なプログラム・タスクを管理します。

表29-3に、プログラムの一般的なタスクとそれに対応するプロシージャおよび権限を示します。

表29-3 プログラムのタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
プログラムの作成	CREATE_PROGRAM	CREATE JOB または CREATE ANY JOB
プログラムの変更	SET_ATTRIBUTE	ALTER または CREATE ANY JOB、あるいは所有者
プログラムの削除	DROP_PROGRAM	ALTER または CREATE ANY JOB、あるいは所有者
プログラムの使用禁止	DISABLE	ALTER または CREATE ANY JOB、あるいは所有者
プログラムの使用可能化	ENABLE	ALTER または CREATE ANY JOB、あるいは所有者

権限の詳細は、「スケジューラ権限」を参照してください。

親トピック: ジョブを定義するためのプログラムの作成および管理

29.3.2 スケジューラを使用するプログラムの作成

プログラムでは、スケジューラによって実行される内容が記述されます。

- プログラムの作成
  プログラムを作成するには、CREATE_PROGRAMプロシージャまたはCloud Controlを使用します。
- プログラム引数の定義

プログラムを作成した後、プログラムの引数を定義できます。

親トピック: ジョブを定義するためのプログラムの作成および管理

29.3.2.1 プログラムの作成

プログラムを作成するには、CREATE_PROGRAMプロシージャまたはCloud Controlを使用します。

デフォルトでは、プログラムは作成者のスキーマに作成されます。別のユーザーのスキーマにプログラムを作成するには、スキーマ名でプログラム名を修飾する必要があります。他のユーザーがこのプログラムを使用するには、プログラムに対するEXECUTE権限が必要であるため、プログラムを作成したら、EXECUTE権限を付与する必要があります。

次の例では、my_program1というプログラムを作成しています。

```
BEGIN
  DBMS_SCHEDULER.CREATE_PROGRAM (
   program_name           => 'my_program1',
   program_action         => '/usr/local/bin/date',
   program_type           => 'EXECUTABLE',
   comments               => 'My comments here');
END;
/
```

プログラムは、デフォルトで使用禁止状態で作成されるため、そのプログラムを指し示すジョブを使用可能にするには、プログラムを使用可能にする必要があります。

「プログラム引数の定義」に説明されているDEFINE_XXX_ARGUMENTプロシージャを使用してプログラムのすべての引数を指定する前に、引数が必要なプログラムを使用可能にしないでください。

親トピック: スケジューラを使用するプログラムの作成

29.3.2.2 プログラム引数の定義

プログラムを作成した後、プログラムの引数を定義できます。

引数は、コーリング順序内の位置によって定義し、オプションで引数名とデフォルト値を指定できます。プログラム引数のデフォルト値が定義されていない場合は、そのプログラムを参照するジョブで引数の値を指定する必要があります。(ジョブで、デフォルト値を上書きすることもできます。)すべての引数値を指定しないと、ジョブを有効にできません。

プログラム引数値を設定するには、DEFINE_PROGRAM_ARGUMENTまたはDEFINE_ANYDATA_ARGUMENTプロシージャを使用します。DEFINE_ANYDATA_ARGUMENTは、ANYDATAオブジェクト内にカプセル化する必要がある複雑な型に使用します。引数を必要とする場合があるプログラムの例の1つに、開始日と終了日が必要なレポート作成プログラムを開始するジョブがあります。次のコード例では、レポート作成プログラムの2番目の引数である終了日の引数が設定されます。この例では、SET_JOB_ANYDATA_VALUEやSET_JOB_ARGUMENT_VALUEなどの他のパッケージ・プロシージャから(位置ではなく)名前で引数を参照できるように、引数に名前が割り当てられています。

```
BEGIN
 DBMS_SCHEDULER.DEFINE_PROGRAM_ARGUMENT (
   program_name             => 'operations_reporting',
   argument_position        => 2,
   argument_name            => 'end_date',
   argument_type            => 'VARCHAR2',
   default_value            => '12-DEC-03');
END;
/
```

argument_type引数の有効な値はSQLデータ型である必要があるため、ブールはサポートされていません。外部実行可能ファイルの場合、CHARやVARCHAR2などの文字列型のみを使用できます。

プログラム引数は、名前または位置で削除できます。次に例を示します。

```
BEGIN
 DBMS_SCHEDULER.DROP_PROGRAM_ARGUMENT (
   program_name            => 'operations_reporting',
   argument_position       => 2);
 DBMS_SCHEDULER.DROP_PROGRAM_ARGUMENT (
   program_name            => 'operations_reporting',
   argument_name           => 'end_date');
END;
/
```

一部の特殊なケースでは、プログラムのロジックがスケジューラの環境によって異なる場合があります。この目的のために、スケジューラには、プログラムに引数として渡すことができる事前定義のメタデータ引数がいくつかあります。たとえば、スケジュールがウィンドウ名である一部のジョブについては、ジョブの開始時にウィンドウがオープンしている時間の長さを認識していると便利です。これは、ウィンドウ終了時間をプログラムに対するメタデータ引数として定義すると可能です。

特定ジョブのメタデータに対するアクセス権限がプログラムに必要な場合は、プログラムの実行時にスケジューラによって値が指定されるように、DEFINE_METADATA_ARGUMENTプロシージャを使用して特別なメタデータ引数を定義できます。

関連項目:

「ジョブ引数の設定」

親トピック: スケジューラを使用するプログラムの作成

29.3.3 プログラムの変更

プログラムを変更するには、その属性を変更します。プログラムを変更するには、Cloud ControlまたはDBMS_SCHEDULER.SET_ATTRIBUTEおよびDBMS_SCHEDULER.SET_ATTRIBUTE_NULLパッケージ・プロシージャを使用します。

プログラム属性の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_SCHEDULER.CREATE_PROGRAMプロシージャに関する項を参照してください。

変更したプログラムが、現在実行中のジョブで使用されている場合、そのジョブは変更操作前に定義されたプログラムで引き続き実行されます。

次の例では、プログラムmy_program1で実行される実行可能ファイルを変更しています。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
   name           => 'my_program1',
   attribute      => 'program_action',
   value          => '/usr/local/bin/salesreports1');
END;
/
```

親トピック: ジョブを定義するためのプログラムの作成および管理

29.3.4 プログラムの削除

1つ以上のプログラムを削除するには、DBMS_SCHEDULERパッケージのDROP_PROGRAMプロシージャまたはCloud Controlを使用します。

プログラムが削除されると、そのプログラムに関係する引数もすべて削除されます。プログラム名のカンマ区切りのリストを指定す

ることで、1回のコールで複数のプログラムを削除できます。たとえば、次の文では3つのプログラムが削除されます。

```
BEGIN
  DBMS_SCHEDULER.DROP_PROGRAM('program1, program2, program3');
END;
/
```

このプログラムを指定した実行中のジョブは、DROP_PROGRAMコールの影響を受けずに続行できます。

force引数をTRUEに設定すると、このプログラムを指し示しているジョブは使用禁止になり、プログラムは削除されます。force引数をデフォルトのFALSEに設定すると、プログラムを指し示すジョブがある場合はコールが失敗します。

DROP_PROGRAMプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブを定義するためのプログラムの作成および管理

29.3.5 プログラムの無効化

1つ以上のプログラムを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャまたはCloud Controlを使用します。

プログラムが使用禁止になると、ステータスがdisabledに変更されます。使用禁止プログラムとは、メタデータが存在しても、このプログラムを指定したジョブを実行できないことを意味します。

このプログラムを指し示す実行中のジョブはDISABLEコールの影響を受けず、実行を継続できます。また、プログラムが使用禁止になった場合にも、そのプログラムに関係する引数は影響を受けません。

プログラムは、プログラム引数が削除された場合やnumber_of_argumentsの変更で引数を定義できなくなった場合など、他の理由で使用禁止になる場合もあります。

DISABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブを定義するためのプログラムの作成および管理

29.3.6 プログラムの有効化

1つ以上のプログラムを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャまたはCloud Controlを使用します。

プログラムが使用可能になると、使用可能フラグがTRUEに設定されます。プログラムは、デフォルトで使用禁止で作成されるため、そのプログラムを指し示すジョブを使用可能にするには、プログラムを使用可能にする必要があります。プログラムが使用可能になる前に、処理が有効であること、およびすべての引数が定義されていることを確認するために妥当性チェックが実行されます。

ENABLEプロシージャ・コールにプログラム名のカンマ区切りのリストを指定することで、1回のコールで複数のプログラムを使用可能にできます。たとえば、次の文では3つのプログラムが使用可能になります。

```
BEGIN
  DBMS_SCHEDULER.ENABLE('program1, program2, program3');
END;
/
```

ENABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブを定義するためのプログラムの作成および管理

29.4 ジョブを定義するためのスケジュールの作成および管理

必要に応じて、スケジュール・オブジェクト(スケジュール)を使用してジョブの実行時期を定義できます。スケジュールは、データベースにオブジェクトとして作成および保存することによってユーザー間で共有できます。

- スケジュールのタスクとそのプロシージャ
  DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なスケジュール・タスクを管理します。
- スケジュールの作成
  スケジュールを作成するには、DBMS_SCHEDULERパッケージのCREATE_SCHEDULEプロシージャまたはCloud Controlを使用します。
- スケジュールの変更
  スケジュールを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEおよびSET_ATTRIBUTE_NULLプロシージャまたはCloud Controlを使用します。
- スケジュールの削除
  スケジュールを削除するには、DBMS_SCHEDULERパッケージのDROP_SCHEDULEプロシージャまたはCloud Controlを使用します。
- 繰返し間隔の設定
  ジョブの繰返し時期および頻度を制御できます。

関連項目:

- スケジュールの概要については、「スケジュール」を参照してください。
- ジョブ・リソース使用率を管理しながらジョブをスケジュールする方法は、「ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理」および「ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理」

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.4.1 スケジュールのタスクとそのプロシージャ

DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なスケジュール・タスクを管理します。

表29-4に、スケジュールの一般的なタスクとその処理に使用するプロシージャを示します。

表29-4 スケジュールのタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
スケジュールの作成	CREATE_SCHEDULE	CREATE JOB または CREATE ANY JOB
スケジュールの変更	SET_ATTRIBUTE	ALTER または CREATE ANY JOB、あるいは所有者
スケジュールの削除	DROP_SCHEDULE	ALTER または CREATE ANY JOB、あるいは所有者

詳細は、「スケジューラ権限」を参照してください。

親トピック: ジョブを定義するためのスケジュールの作成および管理

29.4.2 スケジュールの作成

スケジュールを作成するには、DBMS_SCHEDULERパッケージのCREATE_SCHEDULEプロシージャまたはCloud Controlを使用します。

スケジュールは、そのスケジュールを作成するユーザーのスキーマ内に作成され、最初に作成した時点で使用可能です。また、別のユーザーのスキーマ内に作成できます。スケジュールが作成されると、他のユーザーもそのスケジュールを使用できます。スケジュールは、PUBLICへのアクセス権を伴って作成されます。このため、スケジュールへのアクセス権限を明示的に付与する必要はありません。次の例では、スケジュールを作成しています。

```
BEGIN
 DBMS_SCHEDULER.CREATE_SCHEDULE (
  schedule_name     => 'my_stats_schedule',
  start_date        => SYSTIMESTAMP,
  end_date          => SYSTIMESTAMP + INTERVAL '30' day,
  repeat_interval   => 'FREQ=HOURLY; INTERVAL=4',
  comments          => 'Every 4 hours');
END;
/
```

関連項目:

- CREATE_SCHEDULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- 「イベント・スケジュールの作成」

親トピック: ジョブを定義するためのスケジュールの作成および管理

29.4.3 スケジュールの変更

スケジュールを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEおよびSET_ATTRIBUTE_NULLプロシージャまたはCloud Controlを使用します。

スケジュールを変更すると、スケジュールの定義が変更されます。スケジュール名以外のすべての属性を変更できます。スケジュールの属性は、*_SCHEDULER_SCHEDULESビューで使用可能です。

スケジュールを変更した場合、この変更は実行中のジョブおよびこのスケジュールを使用するオープン中のウィンドウには影響しません。この変更は、次回ジョブを実行するときまたはウィンドウをオープンするときから有効になります。

SET_ATTRIBUTEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブを定義するためのスケジュールの作成および管理

29.4.4 スケジュールの削除

スケジュールを削除するには、DBMS_SCHEDULERパッケージのDROP_SCHEDULEプロシージャまたはCloud Controlを使用します。

このプロシージャ・コールによって、スケジュール・オブジェクトがデータベースから削除されます。

DROP_SCHEDULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブを定義するためのスケジュールの作成および管理

29.4.5 繰返し間隔の設定

ジョブの繰返し時期および頻度を制御できます。

- 繰返し間隔の設定について
  ジョブの繰返し時期および間隔を制御するには、ジョブ自体またはジョブが参照する名前付きスケジュールの`repeat_interval`属性を設定します。`repeat_interval`は、`DBMS_SCHEDULER`パッケージ・プロシージャまたはCloud Controlで設定できます。
- スケジューラのカレンダ指定構文の使用方法
  ジョブの繰返し間隔を設定する主要な方法は、スケジューラのカレンダ指定式を使用して`repeat_interval`属性を設定する方法です。
- PL/SQL式の使用方法
  カレンダ指定構文より複雑な機能が必要な場合は、PL/SQL式を使用できます。ただし、PL/SQL式はウィンドウまたは名前付きスケジュールでは使用できません。PL/SQL式は、日付またはタイムスタンプに評価される必要があります。
- PL/SQL式とカレンダ指定構文の動作の相違点
  カレンダ指定式とPL/SQLの繰返し間隔では、動作に重要な相違点があります。
- 繰返し間隔と夏時間
  ジョブの繰返しでは、次回のジョブ実行時間のスケジュールは、タイム・ゾーン列のあるタイムスタンプに格納されます。

親トピック: ジョブを定義するためのスケジュールの作成および管理

29.4.5.1 繰返し間隔の設定について

ジョブの繰返し時期および間隔を制御するには、ジョブ自体またはジョブが参照する名前付きスケジュールの`repeat_interval`属性を設定します。`repeat_interval`は、`DBMS_SCHEDULER`パッケージ・プロシージャまたはCloud Controlで設定できます。

`repeat_interval`を評価すると、一連のタイムスタンプが返されます。各タイムスタンプの時点で、スケジューラがジョブを実行します。ジョブまたはスケジュールの開始日も、タイムスタンプの結果セットの決定に役立つことに注意してください。`repeat_interval`に値を指定しないと、指定した開始日に1回のみジョブが実行されます。

ジョブが開始されるとすぐに、`repeat_interval`が評価されて、次にスケジュールされるジョブの実行時刻が決定されます。ジョブがまだ実行されている間に次の実行時刻になることがありますが、ジョブの新しいインスタンスは現在のジョブが完了するまで開始されません。

関連項目:

`repeat_interval`の評価の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 繰返し間隔の設定

29.4.5.2 スケジューラのカレンダ指定構文の使用方法

ジョブの繰返し間隔を設定する主要な方法は、スケジューラのカレンダ指定式を使用して`repeat_interval`属性を設定する方法です。

関連項目:

repeat_intervalのカレンダ指定構文およびCREATE_SCHEDULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

カレンダ指定式の例

次に、簡単な繰返し間隔の例を示します。わかりやすくするために、開始日は評価結果に影響を与えないと仮定します。

毎週金曜日に実行(3つの例はすべて同等です。)

```
FREQ=DAILY; BYDAY=FRI;
FREQ=WEEKLY; BYDAY=FRI;
FREQ=YEARLY; BYDAY=FRI;
```

隔週金曜日に実行。

```
FREQ=WEEKLY; INTERVAL=2; BYDAY=FRI;
```

毎月末に実行。

```
FREQ=MONTHLY; BYMONTHDAY=-1;
```

毎月末の翌日に実行。

```
FREQ=MONTHLY; BYMONTHDAY=-2;
```

3月10日に実行(両方の例は同等です。)

```
FREQ=YEARLY; BYMONTH=MAR; BYMONTHDAY=10;
FREQ=YEARLY; BYDATE=0310;
```

10日おきに実行。

```
FREQ=DAILY; INTERVAL=10;
```

毎日午後4時、5時および6時に実行。

```
FREQ=DAILY; BYHOUR=16,17,18;
```

隔月15日に実行。

```
FREQ=MONTHLY; INTERVAL=2; BYMONTHDAY=15;
```

毎月29日に実行。

```
FREQ=MONTHLY; BYMONTHDAY=29;
```

毎月第2水曜日に実行。

```
FREQ=MONTHLY; BYDAY=2WED;
```

毎年最終金曜日に実行。

```
FREQ=YEARLY; BYDAY=-1FRI;
```

50時間おきに実行。

```
FREQ=HOURLY; INTERVAL=50;
```

隔月末に実行。

```
FREQ=MONTHLY; INTERVAL=2; BYMONTHDAY=-1;
```

毎月初めの3日間1時間おきに実行。

```
FREQ=HOURLY; BYMONTHDAY=1,2,3;
```

以降は、多少複雑な繰返し間隔の例です。

毎月最後の稼働日に実行(稼働日は月曜日から金曜日と仮定)。

```
FREQ=MONTHLY; BYDAY=MON,TUE,WED,THU,FRI; BYSETPOS=-1
```

会社の祝日を除く、毎月の最後の稼働日に実行(この例では、Company_Holidaysという既存の名前付きスケジュールを参照しています。)

```
FREQ=MONTHLY; BYDAY=MON,TUE,WED,THU,FRI; EXCLUDE=Company_Holidays; BYSETPOS=-1
```

毎週金曜日および会社の休業日の正午に実行。

```
FREQ=YEARLY;BYDAY=FRI;BYHOUR=12;INCLUDE=Company_Holidays
```

7月4日、メモリアル・デーおよびレイバー・デーの3つの祝日に実行(この例では、それぞれの祝日に対応する1日を定義したJUL4、MEMおよびLABの既存の3つの名前付きスケジュールを参照しています。)

```
JUL4,MEM,LAB
```

カレンダ指定式評価の例

繰返し間隔を"FREQ=MINUTELY;INTERVAL=2;BYHOUR=17; BYMINUTE=2,4,5,50,51,7;"に、開始日を28-FEB-2004 23:00:00に設定すると、次のスケジュールが生成されます。

```
SUN 29-FEB-2004 17:02:00
SUN 29-FEB-2004 17:04:00
SUN 29-FEB-2004 17:50:00
MON 01-MAR-2004 17:02:00
MON 01-MAR-2004 17:04:00
MON 01-MAR-2004 17:50:00
...
```

繰返し間隔を"FREQ=MONTHLY;BYMONTHDAY=15, -1"に、開始日を29-DEC-2003 9:00:00に設定すると、次のスケジュールが生成されます。

```
WED 31-DEC-2003 09:00:00
THU 15-JAN-2004 09:00:00
SAT 31-JAN-2004 09:00:00
SUN 15-FEB-2004 09:00:00
SUN 29-FEB-2004 09:00:00
MON 15-MAR-2004 09:00:00
WED 31-MAR-2004 09:00:00
...
```

繰返し間隔を"FREQ=MONTHLY;"に、開始日を29-DEC-2003 9:00:00に設定すると、次のスケジュールが生成されます。(BYMONTHDAY句がないため、開始日から日付が取得されることに注意してください。)

```
MON 29-DEC-2003 09:00:00
THU 29-JAN-2004 09:00:00
SUN 29-FEB-2004 09:00:00
MON 29-MAR-2004 09:00:00
...
```

カレンダ指定式の使用例

カレンダ指定構文を使用した次の例について考えてみます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name             => 'scott.my_job1',
   start_date           => '15-JUL-04 01.00.00 AM Europe/Warsaw',
   repeat_interval      => 'FREQ=MINUTELY; INTERVAL=30;',
   end_date             => '15-SEP-04 01.00.00 AM Europe/Warsaw',
   comments             => 'My comments here');
END;
/
```

これにより、my_job1がscottに作成されます。7月15日に始めて実行されて、9月15日まで実行されます。このジョブの開始日は7月15日、終了日は9月15日で、30分おきに実行されます。

親トピック: 繰返し間隔の設定

29.4.5.3 PL/SQL式の使用方法

カレンダ指定構文より複雑な機能が必要な場合は、PL/SQL式を使用できます。ただし、PL/SQL式はウィンドウまたは名前付きスケジュールでは使用できません。PL/SQL式は、日付またはタイムスタンプに評価される必要があります。

これ以外に制限はないため、適切なプログラミングによって、あらゆる繰返し間隔を作成できます。例として、次の文を考えてみます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name             => 'scott.my_job2',
   start_date           => '15-JUL-04 01.00.00 AM Europe/Warsaw',
   repeat_interval      => 'SYSTIMESTAMP + INTERVAL '30' MINUTE',
   end_date             => '15-SEP-04 01.00.00 AM Europe/Warsaw',
   comments             => 'My comments here');
END;
/
```

これにより、my_job1がscottに作成されます。7月15日に始めて実行されて、9月15日まで30分ごとに実行されます。repeat_intervalが、30分後の日付を返すSYSTIMESTAMP + INTERVAL '30' MINUTEに設定されているため、30分ごとにジョブが実行されます。

親トピック: 繰返し間隔の設定

29.4.5.4 PL/SQL式とカレンダ指定構文の動作の相違点

カレンダ指定式とPL/SQL繰返し間隔では、動作に重要な相違点があります。

相違点は、次のとおりです。

- 開始日
  - カレンダ指定構文を使用する場合、開始日は参照専用の日になります。そのため、スケジュールはこの日に有効になります。開始日にジョブが開始されるという意味ではありません。
  - PL/SQL式を使用した場合、開始日はジョブが最初に実行を開始した実際の時間を表します。
- 次回の実行時期
  - カレンダ指定構文を使用した場合、次回のジョブ実行時期は固定されています。
  - PL/SQL式を使用する場合、次にジョブが実行される時期は、現在のジョブ実行の実際の開始時刻によって異なります。

違いの例として、ジョブが午後2時に開始して、2時間ごとに繰り返されるようにスケジュールされているにもかかわらず、実際は2時10分に開始されたとします。

- カレンダ指定構文で繰返し間隔が指定されている場合、ジョブは4時、6時などに繰返し実行されます。
- PL/SQLを使用した場合、ジョブは4時10分に繰り返されますが、実際には次のジョブが4時11分に開始した場合は、後続の実行が6時11分になります。

これら2つの点を例で示すために、15-July-2003 1:45:00という開始日付が設定されていて、2時間ごとに繰り返す必要がある場合を考えます。"FREQ=HOURLY; INTERVAL=2; BYMINUTE=0;"というカレンダ指定構文では、次のスケジュールが生成されます。

```
TUE 15-JUL-2003  03:00:00
TUE 15-JUL-2003  05:00:00
TUE 15-JUL-2003  07:00:00
TUE 15-JUL-2003  09:00:00
TUE 15-JUL-2003  11:00:00
...
```

カレンダ式では、2時間おきの毎正時に繰返しを実行します。

一方、PL/SQL式"SYSTIMESTAMP + interval '2' hour"では、実行時間は次のようになります。

```
TUE 15-JUL-2003  01:45:00
TUE 15-JUL-2003  03:45:05
TUE 15-JUL-2003  05:45:09
TUE 15-JUL-2003  07:45:14
TUE 15-JUL-2003  09:45:20
...
```

親トピック: 繰返し間隔の設定

29.4.5.5 繰返し間隔と夏時間

ジョブの繰返しでは、次回のジョブ実行時間のスケジュールは、タイム・ゾーン列のあるタイムスタンプに格納されます。

- カレンダ指定構文を使用する場合、タイム・ゾーンはstart_dateから取り出されます。start_dateが指定されていない場合の動作の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- PL/SQL繰返し間隔を使用する場合、タイム・ゾーンはPL/SQL式が返すタイムスタンプの一部になっています。

いずれの場合も、リージョン名を使用することが重要です。たとえば、"+2:00"のような絶対タイム・ゾーン・オフセットではなく、"Europe/Istanbul"を使用します。タイム・ゾーンがリージョン名で指定されている場合のみ、スケジューラはそのリージョンに適用される夏時間調整に従います。

親トピック: 繰返し間隔の設定

29.5 イベントを使用したジョブの開始

イベントが送信されたとき、Oracle Schedulerはジョブを開始できます。イベントは、アプリケーションまたはシステム・プロセス間で送信されるメッセージです。

- イベントについて
  イベントは、なんらかの処理または発生が検出されたことを、アプリケーションまたはシステム・プロセスが別のアプリケーションまたはシステム・プロセスに示すために送信するメッセージです。イベントは、1つのアプリケーションまたはプロセスによって呼び出され(送信)、1つ以上のアプリケーションまたはプロセスによって使用されます(受信)。

- アプリケーションによって呼び出されたイベントによるジョブの開始
  Oracle Schedulerは、アプリケーションによるイベントの発生時にジョブを開始できます。
- ファイルがシステムに到着したことによるジョブの開始
  ファイルがローカル・システムまたはリモート・システムに到着したときにジョブを開始するようにスケジューラを構成できます。ジョブはイベントベースのジョブであり、ファイル到着イベントはFile Watcher(Oracle Database 11g リリース2 (11.2)で導入されたスケジューラ・オブジェクト)によって呼び出されます。

関連項目:

- 「イベントに基づくジョブとスケジュールの作成例」
- プロセス・フローを正確に制御するためにチェーンでのイベントの使用については、「ジョブ・チェーンの作成と管理」

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.5.1 イベントについて

イベントは、なんらかの処理または発生が検出されたことを、アプリケーションまたはシステム・プロセスが別のアプリケーションまたはシステム・プロセスに示すために送信するメッセージです。イベントは、1つのアプリケーションまたはプロセスによって呼び出され(送信)、1つ以上のアプリケーションまたはプロセスによって使用されます(受信)。

スケジューラでは、次の2種類のイベントが使用されます。

- アプリケーションによって呼び出されるイベント

  アプリケーションは、スケジューラが使用するイベントを呼び出すことができます。スケジューラは、ジョブを開始することでこのイベントを処理します。たとえば、在庫が一定のしきい値を下回ったことを感知した在庫追跡システムでは、在庫の補充ジョブを開始するイベントを呼び出すことができます。

  「アプリケーションによって呼び出されたイベントによるジョブの開始」を参照してください。

- File Watcherによって呼び出されるファイル到着イベント

  File Watcher (Oracle Database 11gリリース2 (11.2)で導入されたスケジューラ・オブジェクト)を作成して、システムへのファイルの到着を監視できます。その後、File Watcherでファイルの存在が検出されたときに開始するジョブを構成できます。たとえば、チェーン店用のデータ・ウェアハウスでは、店舗からアップロードされた日次売上レポートからデータがロードされます。データ・ウェアハウスのロード・ジョブは、新しい日次レポートが到着するたびに開始されます。

  「ファイルがシステムに到着したことによるジョブの開始」を参照

関連項目:

スケジューラによって呼び出されるジョブ状態変化イベントがアプリケーションで使用される仕組みの詳細は、「スケジューラによって呼び出されるイベントによるジョブ状態の監視」

親トピック: イベントを使用したジョブの開始

29.5.2 アプリケーションによって呼び出されたイベントによるジョブの開始

Oracle Schedulerは、アプリケーションによるイベントの発生時にジョブを開始できます。

- アプリケーションによって呼び出されるイベントについて

アプリケーションでは、ジョブを開始するためにスケジューラに通知するイベントを発生できます。この方法で開始されるジョブは、イベントベースのジョブと呼ばれます。

- イベントベースのジョブの作成
  イベントベースのジョブを作成するには、CREATE_JOBプロシージャまたはCloud Controlを使用します。ジョブには、ジョブ属性としてイベント情報をインラインで組み込むか、またはイベント・スケジュールを指し示してイベント情報を指定できます。時間スケジュールに基づくジョブと同様に、イベントベースのジョブは、ジョブ終了日をすぎるか、max_runsまたは最大失敗回数(max_failures)に達するまでは自動削除されません。
- イベントベースのジョブの変更
  イベントベースのジョブを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャを使用します。
- イベント・スケジュールの作成
  イベントに基づいたスケジュールを作成できます。作成したスケジュールは、複数のジョブに再利用できます。そのためには、CREATE_EVENT_SCHEDULEプロシージャまたはCloud Controlを使用します。
- イベント・スケジュールの変更
  イベント・スケジュールのイベント情報を変更する方法は、ジョブのイベント情報を変更する方法と同じです。
- イベントベースのジョブにイベント・メッセージを渡す方法
  メタデータの引数を介して、スケジューラは、イベントベースのジョブに対して、ジョブを開始したイベントのメッセージ内容を渡すことができます。

親トピック: イベントを使用したジョブの開始

29.5.2.1 アプリケーションによって呼び出されるイベントについて

アプリケーションで、ジョブを開始するようにスケジューラに通知するイベントを呼び出すことができます。この方法で開始されるジョブは、イベントベースのジョブと呼ばれます。

日付、時間、繰り返し発生する情報を定義するかわりに、イベントを参照する名前付きスケジュールを作成できます。このようなスケジュール(イベント・スケジュール)が割り当てられたジョブは、イベントが呼び出されると実行されます。

ジョブの開始をスケジューラに通知するイベントを呼び出すために、アプリケーションは、このジョブの設定時に指定したOracle Databaseアドバンスト・キューイングのキューにメッセージをエンキューします。ジョブが開始されると、ジョブはオプションでイベントのメッセージ内容を取得できます。

イベントベースのジョブを作成するには、次の2つの追加属性を設定する必要があります。

- queue_spec

  キュー仕様。アプリケーションがジョブ開始イベントを呼び出すためのメッセージをエンキューするキューの名前が含まれます。または、セキュアなキューの場合はキュー名の後にカンマとエージェント名が記述されます。

- event_condition

  メッセージでジョブを開始するためにTRUEと評価される必要のあるメッセージ・プロパティに基づく条件式。式には、Oracle Databaseアドバンスト・キューイングのルールの構文が必要です。したがって、メッセージ・ペイロードがオブジェクト・タイプで、式のオブジェクト属性の前にtab.user_dataを付加した場合は、ユーザー・データ・プロパティを式に含めることができます。

関連項目:

- キューイング・ルールの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』の

DBMS_AQADM.ADD_SUBSCRIBERプロシージャを参照してください。

- キューの作成方法の詳細は、『Oracle Databaseアドバンスト・キューイング・ユーザーズ・ガイド』を参照してください
- メッセージのエンキュー方法の詳細は、『Oracle Databaseアドバンスト・キューイング・ユーザーズ・ガイド』を参照してください

次の例では、event_conditionが、午前0時から午前9時の間に発生する在庫不足イベントのみを選択するように設定されます。メッセージ・ペイロードはevent_typeとevent_timestampの2つの属性を持つオブジェクトであるとします。

```
event_condition = 'tab.user_data.event_type = ''LOW_INVENTORY'' and
extract hour from tab.user_data.event_timestamp < 9'
```

queue_specおよびevent_conditionは、インラインのジョブ属性として指定するか、またはこれらの属性を指定したイベント・スケジュールを作成し、ジョブからこのスケジュールを指し示すことができます。

ノート:

スケジューラは、event_condition に一致するイベントが発生するたびにイベントベースのジョブを実行します。ただし、デフォルトでは、ジョブがすでに実行されている間に発生したイベントは無視され、イベントは消費されますが別のジョブの実行がトリガーされることはありません。Oracle Database 11g リリース 1 (11.1)以降では、ジョブ属性 PARALLEL_INSTANCES を TRUE に設定することで、このデフォルト動作を変更できます。この場合、ジョブのインスタンスがイベントのすべてのインスタンスに対して起動され、すべてのジョブ・インスタンスが軽量ジョブになります。詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』の SET_ATTRIBUTE プロシージャに関する項を参照してください。

表29-5に、アプリケーションによって呼び出される(スケジューラによって使用される)イベントに関する一般的な管理タスクとそれに対応するプロシージャを示します。

表29-5 アプリケーションによって呼び出されるイベントに対するイベント・タスクとそのプロシージャ

タスク	プロシージャ	必要な権限
イベントベースのジョブの作成	CREATE_JOB	CREATE JOB または CREATE ANY JOB
イベントベースのジョブの変更	SET_ATTRIBUTE	CREATE ANY JOB、あるいは変更対象のジョブの所有権またはそのジョブに対する ALTER 権限
イベント・スケジュールの作成	CREATE_EVENT_SCHEDULE	CREATE JOB または CREATE ANY JOB
イベント・スケジュールの変更	SET_ATTRIBUTE	CREATE ANY JOB、あるいは変更対象のスケジュールの所有権またはそのスケジュールに対する ALTER 権限

親トピック: アプリケーションによって呼び出されたイベントによるジョブの開始

29.5.2.2 イベントベースのジョブの作成

イベントベースのジョブを作成するには、CREATE_JOBプロシージャまたはCloud Controlを使用します。ジョブには、ジョブ属性としてイベント情報をインラインで組み込むか、またはイベント・スケジュールを指し示してイベント情報を指定できます。時間スケジュールに基づくジョブと同様に、イベントベースのジョブは、ジョブ終了日をすぎるか、max_runsまたは最大失敗回数(max_failures)に達するまでは自動削除されません。

- イベント情報をジョブ属性として指定する方法
  イベント情報をジョブ属性として指定するには、CREATE_JOBの代替構文を使用して、queue_specおよびevent_condition属性を組み込みます。
- イベント情報をイベント・スケジュールで指定する方法
  イベント・スケジュールでイベント情報を指定するには、ジョブのschedule_name属性にイベント・スケジュールの名前を設定します。

親トピック: アプリケーションによって呼び出されたイベントによるジョブの開始

29.5.2.2.1 イベント情報をジョブ属性として指定する方法

イベント情報をジョブ属性として指定するには、CREATE_JOBの代替構文を使用して、queue_specおよびevent_condition属性を組み込みます。

次の例では、項目の在庫レベルが低しきい値レベルを下回ったことがアプリケーションからスケジューラに伝達されるときに開始されるジョブを作成しています。

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name            =>  'process_lowinv_j1',
   program_name        =>  'process_lowinv_p1',
   event_condition     =>  'tab.user_data.event_type = ''LOW_INVENTORY''',
   queue_spec          =>  'inv_events_q, inv_agent1',
   enabled             =>  TRUE,
   comments            =>  'Start an inventory replenishment job');
END;
/
```

CREATE_JOBプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: イベント・ベースのジョブの作成

29.5.2.2.2 イベント情報をイベント・スケジュールで指定する方法

イベント情報をイベント・スケジュールで指定するには、ジョブのschedule_name属性にイベント・スケジュールの名前を設定します。

次の例では、イベント・スケジュールでイベント情報を指定します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name            =>  'process_lowinv_j1',
   program_name        =>  'process_lowinv_p1',
   schedule_name       =>  'inventory_events_schedule',
   enabled             =>  TRUE,
   comments            =>  'Start an inventory replenishment job');
END;
/
```

詳細は、「イベント・スケジュールの作成」を参照してください。

親トピック: イベント・ベースのジョブの作成

29.5.2.3 イベントベースのジョブの変更

イベントベースのジョブを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャを使用します。

イベントをインラインで指定するジョブの場合、SET_ATTRIBUTEを使用してqueue_spec属性とevent_condition属性を別々に設定することはできません。かわりに、event_specと呼ばれる属性を設定し、イベント条件とキュー仕様を3番目と4番目の引数としてSET_ATTRIBUTEに渡す必要があります。

次の例では、event_spec属性を使用しています。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE ('my_job', 'event_spec',
   'tab.user_data.event_type = ''LOW_INVENTORY''', 'inv_events_q, inv_agent1');
END;
/
```

SET_ATTRIBUTEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: アプリケーションによって呼び出されたイベントによるジョブの開始

29.5.2.4 イベント・スケジュールの作成

イベントに基づいたスケジュールを作成できます。作成したスケジュールは、複数のジョブに再利用できます。そのためには、CREATE_EVENT_SCHEDULEプロシージャまたはCloud Controlを使用します。

次の例では、イベント・スケジュールを作成しています。

```
BEGIN
  DBMS_SCHEDULER.CREATE_EVENT_SCHEDULE (
   schedule_name     =>  'inventory_events_schedule',
   start_date        =>  SYSTIMESTAMP,
   event_condition   =>  'tab.user_data.event_type = ''LOW_INVENTORY''',
   queue_spec        =>  'inv_events_q, inv_agent1');
END;
/
```

イベント・スケジュールを削除するには、DROP_SCHEDULEプロシージャを使用します。CREATE_EVENT_SCHEDULEの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: アプリケーションによって呼び出されたイベントによるジョブの開始

29.5.2.5 イベント・スケジュールの変更

イベント・スケジュールのイベント情報を変更する方法は、ジョブのイベント情報を変更する方法と同じです。

詳細は、「イベントベースのジョブの変更」を参照してください。

次の例は、イベント・スケジュールのイベント情報を変更するためのSET_ATTRIBUTEプロシージャおよびevent_spec属性の使用方法を示しています。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE ('inventory_events_schedule', 'event_spec',
   'tab.user_data.event_type = ''LOW_INVENTORY''', 'inv_events_q, inv_agent1');
END;
/
```

SET_ATTRIBUTEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: アプリケーションによって呼び出されたイベントによるジョブの開始

29.5.2.6 イベントベースのジョブにイベント・メッセージを渡す方法

メタデータの引数を介して、スケジューラは、イベントベースのジョブに対して、ジョブを開始したイベントのメッセージ内容を渡すことができます。

以下のルールが適用されます。

- ジョブは、タイプSTORED_PROCEDUREの名前付きプログラムを使用する必要があります。
- 名前付きプログラムの引数の1つがメタデータ引数であり、そのmetadata_attributeがEVENT_MESSAGEに設定されている必要があります。
- プログラムを実装するストアド・プロシージャには、名前付きプログラムのメタデータ引数に対応する位置に引数が必要です。引数の型は、アプリケーションがジョブ開始イベントをエンキューするキューのデータ型であることが必要です。

EVENT_MESSAGEメタデータ引数があるジョブをRUN_JOBプロシージャを使用して手動で実行した場合、その引数に渡される値はNULLです。

次の例は、イベント・メッセージの内容を受け取ることができるイベントベースのジョブを作成する方法を示しています。

```
CREATE OR REPLACE PROCEDURE my_stored_proc (event_msg IN event_queue_type)
AS
BEGIN
  -- retrieve and process message body
END;
/

BEGIN
  DBMS_SCHEDULER.CREATE_PROGRAM (
      program_name => 'my_prog',
      program_action=> 'my_stored_proc',
      program_type => 'STORED_PROCEDURE',
      number_of_arguments => 1,
      enabled => FALSE) ;

  DBMS_SCHEDULER.DEFINE_METADATA_ARGUMENT (
      program_name => 'my_prog',
      argument_position => 1 ,
      metadata_attribute => 'EVENT_MESSAGE') ;

  DBMS_SCHEDULER.ENABLE ('my_prog');
EXCEPTION
  WHEN others THEN RAISE ;
END ;
/

BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
     job_name => 'my_evt_job' ,
     program_name => 'my_prog',
     schedule_name => 'my_evt_sch',
     enabled => true,
     auto_Drop => false) ;
EXCEPTION
  WHEN others THEN RAISE ;
END ;
/
```

親トピック: アプリケーションによって呼び出されたイベントによるジョブの開始

29.5.3 ファイルがシステムに到着したことによるジョブの開始

ファイルがローカル・システムまたはリモート・システムに到着したときにジョブを開始するようにスケジューラを構成できます。ジョブはイベントベースのジョブであり、ファイル到着イベントはFile Watcher(Oracle Database 11g リリース2 (11.2)で導入されたスケジューラ・オブジェクト)によって呼び出されます。

- File Watcherについて
  File Watcherは、ファイルがシステムに到着したことによってスケジューラでジョブが開始される場合の対象ファイルの場所、名前などのプロパティを定義したスケジューラ・オブジェクトです。
- リモート・システムからのファイルの到着イベントの有効化
  リモート・システムからファイル到着イベントを受け取るには、そのシステムにスケジューラ・エージェントをインストールし、そのエージェントをデータベースに登録する必要があります。
- File WatcherおよびFile Watcherジョブの作成
  File WatcherおよびFile Watcherジョブを作成するには、いくつかのステップを実行します。
- ファイルの到着の例
  File Watcherジョブのファイル到着の例を示します。
- File Watcherの管理
  DBMS_SCHEDULER PL/SQLパッケージには、File Watcherの属性を有効化、無効化、削除および設定するためのプロシージャが用意されています。
- File Watcherの情報の表示
  *_SCHEDULER_FILE_WATCHERSビューを問い合せることによって、File Watcherに関する情報を表示できます。

親トピック: イベントを使用したジョブの開始

29.5.3.1 File Watcherについて

File Watcherは、ファイルがシステムに到着したことによってスケジューラでジョブが開始される場合の対象ファイルの場所、名前などのプロパティを定義したスケジューラ・オブジェクトです。

ファイル・ウォッチャを作成し、次に、そのファイル・ウォッチャを参照する任意の数のイベントベースのジョブまたはイベント・スケジュールを作成します。File Watcherでは、指定されたファイルの到着、新たに到着したファイルを検出すると、到着イベントを呼び出します。

新たに到着したファイルとは、変更されているため、最新の実行またはFile Watcherジョブがターゲット・ファイル・ディレクトリの監視を開始した時間よりタイムスタンプが遅いファイルです。

File Watcherがファイルが新たに到着したものか否かを判断する方法は、UNIXコマンド`ls -lrt`またはWindows DOSコマンド`dir /od`を繰り返し実行してディレクトリ内の新しいファイルを監視するのと同じです。この両方のコマンドでは必ず、最近に変更されたファイルが最後にリストされます。つまり最も古いものが最初で最も新しいものが最後になります。

ノート:

動作は次のとおりです。

UNIX `mv` コマンドはファイルの変更時間を変更しませんが、`cp` コマンドは変更します。

Windows `move/paste` および `copy/paste` コマンドは、ファイルの変更時間を変更しません。そうするに

は、move または copy コマンドの後に DOS コマンド copy /b file_name +,,を実行します。

CREATE_FILE_WATCHERプロシージャのsteady_state_durationパラメータ(『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照)は、File Watcherがファイルを検出したと判断するまでに、ファイルが未変更のままであることが必要な最小時間間隔を指定します。これは1時間を超えることはできません。パラメータがNULLの場合、内部値が使用されます。

ファイルの到着イベントによって開始されたジョブでは、新しく到着したファイルを示すイベント・メッセージを取得できます。このメッセージには、ファイルの検索、オープンおよび処理に必要な情報が含まれます。

File Watcherでは、ローカル・システム(Oracle Databaseを実行しているホスト・コンピュータ)またはリモート・システムでファイルを監視できます。リモート・システムではスケジューラ・エージェントが実行され、そのエージェントがデータベースに登録されている必要があります。

File Watcherでは、10分間隔でファイルの到着がチェックされます。この間隔は調整可能です。詳細は、「ファイルの到着を検出する間隔の変更」を参照してください。

File Watcherを使用するには、データベースのJava仮想マシン(JVM)コンポーネントをインストールする必要があります。

スキーマにFile Watcherを作成するには、CREATE JOBシステム権限が必要です。自分のスキーマと異なるスキーマ(許可されないSYSスキーマ以外)にFile Watcherを作成するには、CREATE ANY JOBシステム権限が必要です。別のスキーマ内のジョブでFile Watcherを参照できるように、File Watcherに対するEXECUTEオブジェクト権限を付与できます。また、別のユーザーがFile Watcherを変更できるように、File Watcherに対するALTERオブジェクト権限を付与できます。

親トピック: ファイルがシステムに到着したことによるジョブの開始

29.5.3.2 リモート・システムからのファイルの到着イベントの有効化

リモート・システムからファイル到着イベントを受け取るには、そのシステムにスケジューラ・エージェントをインストールし、そのエージェントをデータベースに登録する必要があります。

リモート・システムでファイル到着イベントを生成するのに、Oracle Databaseインスタンスが実行されている必要はありません。

リモート・システムでのファイル到着イベントの呼出しを有効にするには:

1. リモート外部ジョブを実行するようにローカル・データベースを設定します。

   手順は、「リモート・ジョブを実行するためのデータベースの設定の有効化と無効化」を参照してください。

2. 最初のリモート・システムでスケジューラ・エージェントをインストール、構成、登録および起動します。

   手順は、「リモート・ホストでのスケジューラ・エージェントのインストールと構成」を参照してください。

   これにより、ローカル・データベースで管理されている外部宛先のリストにリモート・ホストが追加されます。

3. 追加の各リモート・システムに対して前述のステップを繰り返します。

親トピック: ファイルがシステムに到着したことによるジョブの開始

29.5.3.3 File WatcherおよびFile Watcherジョブの作成

File WatcherおよびFile Watcherジョブを作成するには、いくつかのステップを実行します。

File Watcherを作成し、指定したファイルが到着したときに開始されるイベントベースのジョブを作成するには、次のタスクを実行します。

タスク１　- 資格証明の作成

File Watcher では、ホスト・オペレーティング・システムでファイルへのアクセスを認証するために使用される資格証明オブジェクト(資格証明)が必要になります。資格証明を作成するために必要な権限の詳細は、「資格証明」を参照してください。

次のステップを実行します。

1. 監視対象ファイルにアクセスする必要があるオペレーティング・システム・ユーザーの資格証明を作成します。

```
BEGIN
  DBMS_CREDENTIAL.CREATE_CREDENTIAL('WATCH_CREDENTIAL', 'salesapps',
      'sa324w1');
END;
/
```

2. File Watcher によって開始されるイベントベースのジョブを所有するスキーマに、資格証明に対する EXECUTE オブジェクト権限を付与します。

```
GRANT EXECUTE ON WATCH_CREDENTIAL to DSSUSER;
```

タスク２　- File Watcher の作成

次のステップを実行します。

1. 『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』の DBMS_SCHEDULER.CREATE_FILE_WATCHER プロシージャに関する項の説明に従って属性を割り当て、File Watcher を作成します。ファイル名にワイルドカード・パラメータを指定できます。DIRECTORY_PATH 属性の「?」接頭辞は、Oracle ホーム・ディレクトリへのパスを示します。NULL の destination はローカル・ホストを示します。リモート・ホストのファイルを監視するには、ビューALL_SCHEDULER_EXTERNAL_DESTS から取得できる有効な外部宛先名を指定します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_FILE_WATCHER(
    file_watcher_name => 'EOD_FILE_WATCHER',
    directory_path    => '?/eod_reports',
    file_name         => 'eod*.txt',
    credential_name   => 'watch_credential',
    destination       => NULL,
    enabled           => FALSE);
END;
/
```

2. File Watcher を参照するイベントベースのジョブを所有するスキーマに、File Watcher に対する EXECUTE を付与します。

```
GRANT EXECUTE ON EOD_FILE_WATCHER to dssuser;
```

タスク３ - プログラム・オブジェクトの作成とメタデータ引数の指定

アプリケーションでファイル到着イベント・メッセージの内容(ファイル名、ファイル・サイズなど)を取得できるように、スケジューラ・プログラム・オブジェクトを作成し、イベント・メッセージを参照するメタデータ引数を指定します。

次のステップを実行します。

1. プログラムを作成します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_PROGRAM(
    program_name        => 'dssuser.eod_program',
    program_type        => 'stored_procedure',
    program_action      => 'eod_processor',
    number_of_arguments => 1,
    enabled             => FALSE);
END;
/
```

2. event_message 属性を使用してメタデータ引数を定義します。

```
BEGIN
  DBMS_SCHEDULER.DEFINE_METADATA_ARGUMENT(
    program_name       => 'DSSUSER.EOD_PROGRAM',
    metadata_attribute => 'event_message',
    argument_position  => 1);
END;
/
```

3. プログラムで呼び出すストアド・プロシージャを作成します。

ファイル到着イベントを処理するストアド・プロシージャには、SYS.SCHEDULER_FILEWATCHER_RESULT タイプの引数(イベント・メッセージのデータ型)を指定する必要があります。この引数の位置は、定義済のメタデータ引数の位置と一致させる必要があります。これにより、プロシージャでこの抽象データ型の属性にアクセスして、到着したファイルに関する情報を把握できるようになります。

関連項目:

- DEFINE_METADATA_ARGUMENT プロシージャの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。
- SYS.SCHEDULER_FILEWATCHER_RESULT 型の詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

タスク４　- File Watcher を参照するイベントベースのジョブの作成

「イベントベースのジョブの作成」の説明に従ってイベントベースのジョブを作成します。ただし、queue_spec 属性にキューの仕様を指定するかわりに、File Watcher の名前を指定してください。通常は、event_condition ジョブ属性を null のままにしますが、必要に応じて条件を指定できます。

ジョブに対して queue_spec 属性を設定するかわりに、イベント・スケジュールを作成し、そのイベント・スケジュールの queue_spec 属性で File Watcher を参照し、ジョブの schedule_name 属性でそのイベント・スケジュールを参照できます。

次のステップを実行して、イベントベースのジョブを準備します。

1. ジョブを作成します。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB(
    job_name        => 'dssuser.eod_job',
    program_name    => 'dssuser.eod_program',
    event_condition => NULL,
    queue_spec      => 'eod_file_watcher',
    auto_drop       => FALSE,
    enabled         => FALSE);
END;
/
```

2. ジョブによって前のイベントがすでに処理されていても、ファイル到着イベントの各インスタンスに対してジョブを実行する場合は、parallel_instances 属性を TRUE に設定します。この設定を使用すると、ジョブは軽量ジョブとして実行され、ジョブの複数のインスタンスを迅速に開始できるようになります。イベントベースのジョブで別のイベントがすでに処理されている間に発生する File Watcher イベントを破棄するには、parallel_instances 属性を FALSE(デフォルト)のままにします。

```
BEGIN

DBMS_SCHEDULER.SET_ATTRIBUTE('dssuser.eod_job','parallel_instances',TRUE);
END;
/
```

この属性の詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』の SET_ATTRIBUTE に関する項を参照してください。

関連項目:

- 「イベント・スケジュールの作成」

- 名前付きプログラムとスケジュールを使用したジョブの作成

タスク５ - すべてのオブジェクトの有効化

File Watcher、プログラムおよびジョブを有効にします。

```
BEGIN

DBMS_SCHEDULER.ENABLE('DSSUSER.EOD_PROGRAM,DSSUSER.EOD_JOB,EOD_FILE_WATCHER');
END;
/
```

親トピック: ファイルがシステムに到着したことによるジョブの開始

29.5.3.4 ファイルの到着の例

File Watcherジョブのファイル到着の例を示します。

この例では、イベントベースのジョブによって、各地からの日次売上レポートがローカル・ホストに到着するのを監視しています。各レポート・ファイルが到着すると、ストアド・プロシージャによりファイルに関する情報が取得され、eod_reportsという表に格納されます。その後、定期レポート集計ジョブにより、この表を問い合せて、すべての未処理ファイルを処理し、新たに処理したファイルを処理済としてマークできます。

ここでは、次のコードを実行するデータベース・ユーザーにSYS.SCHEDULER_FILEWATCHER_RESULTデータ型に対するEXECUTEが付与されていることを想定しています。

```
BEGIN
  DBMS_CREDENTIAL.CREATE_CREDENTIAL(
     credential_name => 'watch_credential',
     username        => 'pos1',
     password        => 'jk4545st');
END;
/

CREATE TABLE eod_reports (WHEN timestamp, file_name varchar2(100),
   file_size number, processed char(1));

CREATE OR REPLACE PROCEDURE q_eod_report
  (payload IN sys.scheduler_filewatcher_result) AS
BEGIN
  INSERT INTO eod_reports VALUES
     (payload.file_timestamp,
      payload.directory_path || '/' || payload.actual_file_name,
      payload.file_size,
      'N');
END;
/

BEGIN
  DBMS_SCHEDULER.CREATE_PROGRAM(
    program_name        => 'eod_prog',
    program_type        => 'stored_procedure',
    program_action      => 'q_eod_report',
    number_of_arguments => 1,
    enabled             => FALSE);
  DBMS_SCHEDULER.DEFINE_METADATA_ARGUMENT(
    program_name        => 'eod_prog',
    metadata_attribute  => 'event_message',
    argument_position   => 1);
  DBMS_SCHEDULER.ENABLE('eod_prog');
END;
```

```
/

BEGIN
  DBMS_SCHEDULER.CREATE_FILE_WATCHER(
    file_watcher_name => 'eod_reports_watcher',
    directory_path    => '?/eod_reports',
    file_name         => 'eod*.txt',
    credential_name   => 'watch_credential',
    destination       => NULL,
    enabled           => FALSE);
END;
/

BEGIN
  DBMS_SCHEDULER.CREATE_JOB(
    job_name        => 'eod_job',
    program_name    => 'eod_prog',
    event_condition => 'tab.user_data.file_size > 10',
    queue_spec      => 'eod_reports_watcher',
    auto_drop       => FALSE,
    enabled         => FALSE);
  DBMS_SCHEDULER.SET_ATTRIBUTE('EOD_JOB','PARALLEL_INSTANCES',TRUE);
END;
/

EXEC DBMS_SCHEDULER.ENABLE('eod_reports_watcher,eod_job');
```

親トピック: ファイルがシステムに到着したことによるジョブの開始

29.5.3.5 File Watcherの管理

DBMS_SCHEDULER PL/SQLパッケージには、File Watcherの属性を有効化、無効化、削除および設定するためのプロシージャが用意されています。

- File Watcherの有効化
  File Watcherが無効になっている場合、それを有効にするにはDBMS_SCHEDULER.ENABLEを使用します。
- File Watcherの変更
  File Watcherの属性を変更するには、DBMS_SCHEDULER.SET_ATTRIBUTEおよびDBMS_SCHEDULER.SET_ATTRIBUTE_NULLパッケージ・プロシージャを使用します。
- File Watcherの無効化および削除
  File Watcherを無効にするにはDBMS_SCHEDULER.DISABLEプロシージャ、File Watcherを削除するにはDBMS_SCHEDULER.DROP_FILE_WATCHERプロシージャを使用します。
- ファイルの到着を検出する間隔の変更
  File Watcherでは、デフォルトでは10分間隔でファイルの到着がチェックされます。この間隔は変更可能です。

関連項目:

DBMS_SCHEDULER PL/SQLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ファイルがシステムに到着したことによるジョブの開始

29.5.3.5.1 File Watcherの有効化

File Watcherが無効になっている場合、それを有効にするにはDBMS_SCHEDULER.ENABLEを使用します。

これは、「タスク5: すべてのオブジェクトの有効化」で示しています。

File Watcherを有効にできるのは、そのすべての属性が有効な値に設定され、File Watcherの所有者に指定の資格証明に対するEXECUTE権限が付与されている場合のみです。

親トピック: File Watcherの管理

29.5.3.5.2 File Watcherの変更

File Watcherの属性を変更するには、DBMS_SCHEDULER.SET_ATTRIBUTEおよびDBMS_SCHEDULER.SET_ATTRIBUTE_NULLパッケージ・プロシージャを使用します。

File Watcherの属性の詳細は、CREATE_FILE_WATCHERプロシージャの説明を参照してください。

親トピック: File Watcherの管理

29.5.3.5.3 File Watcherの無効化および削除

File WatcherをDBMS_SCHEDULER.DISABLEプロシージャ、File Watcherを削除するにはDBMS_SCHEDULER.DROP_FILE_WATCHERプロシージャを使用します。

File Watcherに依存するジョブが存在する場合、File Watcherを無効にしたり、削除することはできません。このような場合に無効化または削除操作を強制するには、FORCE属性をTRUEに設定します。File Watcherの無効化または削除を強制すると、File Watcherに依存するジョブは無効になります。

親トピック: File Watcherの管理

29.5.3.5.4 ファイルの到着を検出する間隔の変更

File Watcherでは、デフォルトでは10分間隔でファイルの到着がチェックされます。この間隔は変更可能です。

ファイルの到着を検出する間隔を変更するには:

1. SYSユーザーとしてデータベースに接続します。
2. 事前定義のスケジュールSYS.FILE_WATCHER_SCHEDULEのREPEAT_INTERVAL属性を変更します。有効なカレンダ指定構文を使用してください。

   File WatcherのREPEAT_INTERVALをいずれかのSTEADY_STATE_DURATION属性値よりも小さい値に設定することはお薦めしません。

   関連項目:

   - File Watcherの属性の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
   - CREATE_FILE_WATCHERパラメータの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

   次の例では、ファイルの到着を検出する頻度を2分間隔に変更しています。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE('FILE_WATCHER_SCHEDULE', 'REPEAT_INTERVAL',
    'FREQ=MINUTELY;INTERVAL=2');
END;
/
```

親トピック: File Watcherの管理

29.5.3.6 File Watcherの情報の表示

*_SCHEDULER_FILE_WATCHERSビューを問い合せることによって、File Watcherに関する情報を表示できます。

たとえば、次の問合せを実行します。

```
SELECT file_watcher_name, destination, directory_path, file_name, credential_name
   FROM dba_scheduler_file_watchers;
FILE_WATCHER_NAME    DESTINATION          DIRECTORY_PATH       FILE_NAME
CREDENTIAL_NAME
-------------------- -------------------- -------------------- ---------- -----------
-----
MYFW                 dsshost.example.com  /tmp                 abc        MYFW_CRED
EOD_FILE_WATCHER                          ?/eod_reports        eod*.txt
WATCH_CREDENTIAL
```

関連項目:

*_SCHEDULER_FILE_WATCHERSビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: ファイルがシステムに到着したことによるジョブの開始

29.6 ジョブ・チェーンの作成と管理

ジョブ・チェーンは、結合した1つの目的のために互いにリンクされた一連の名前付きタスクです。

- ジョブ・チェーンの作成と管理について
  ジョブ・チェーンを使用して、前の1つ以上のジョブの結果に応じて異なるジョブが開始される依存性ベースのスケジューリングを実装できます。
- チェーンのタスクとそのプロシージャ
  DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なチェーン・タスクを管理します。
- チェーンの作成
  チェーンを作成するには、DBMS_SCHEDULERパッケージのCREATE_CHAINプロシージャを使用します。
- チェーン・ステップの定義
  チェーン・オブジェクトの作成後、1つ以上のチェーン・ステップを定義します。
- チェーンへのルールの追加
  チェーンにルールを追加するには、DBMS_SCHEDULERパッケージのDEFINE_CHAIN_RULEプロシージャを使用します。このプロシージャは、チェーンに追加する各ルールに対して1回ずつコールします。
- チェーン・ルールの評価間隔の設定
  スケジューラでは、チェーン・ジョブの開始時と各チェーン・ステップの終了時に、すべてのチェーン・ルールが評価されます。
- チェーンの有効化
  チェーンを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャを使用します。ジョブでチェーンを実行するには、チェーンを使用可能にする必要があります。すでに使用可能なチェーンを使用可能にしてもエラーは戻りません。
- チェーン用のジョブの作成
  チェーンを実行するには、DBMS_SCHEDULERパッケージのRUN_CHAINプロシージャを使用するか、CHAINタイプのジョブ(チェーン・ジョブ)を作成およびスケジュールする必要があります。
- チェーンの削除
  ステップおよびルールを含むチェーンを削除するには、DBMS_SCHEDULERパッケージのDROP_CHAINプロシージャを使用します。

- チェーンの実行
  チェーンを即時に実行するには、DBMS_SCHEDULERパッケージのRUN_JOBまたはRUN_CHAINプロシージャを使用します。
- チェーン・ルールの削除
  チェーンからルールを削除するには、DBMS_SCHEDULERパッケージのDROP_CHAIN_RULEプロシージャを使用します。
- チェーンの無効化
  チェーンを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャを使用します。
- チェーン・ステップの削除
  チェーンからステップを削除するには、DBMS_SCHEDULERパッケージのDROP_CHAIN_STEPプロシージャを使用します。
- チェーンの停止
  実行中のチェーンを停止するには、DBMS_SCHEDULER.STOP_JOBプロシージャをコールして、チェーン・ジョブ(チェーンを開始したジョブ)の名前を渡します。
- 個々のチェーン・ステップの停止
  ルールの条件が満たされたときに1つ以上のステップが停止するチェーン・ルールを作成するか、STOP_JOBプロシージャを呼び出すことにより、個々のチェーン・ステップを停止できます。
- チェーンの一時停止
  チェーン全体、またはチェーンの個別ブランチを一時停止できます。そのためには、DBMS_SCHEDULER.ALTER_CHAINまたはALTER_RUNNING_CHAINプロシージャで、1つ以上のステップのPAUSE属性をTRUEに設定します。
- チェーン・ステップのスキップ
  チェーンの1つ以上のステップをスキップできます。そのためには、DBMS_SCHEDULER.ALTER_CHAINまたはALTER_RUNNING_CHAINプロシージャで、1つ以上のステップのSKIP属性をTRUEに設定します。
- チェーンの一部実行
  チェーンの一部のみを実行することができます。
- 実行中のチェーンの監視
  実行中のチェーンの状態は、*_SCHEDULER_RUNNING_JOBSおよび*_SCHEDULER_RUNNING_CHAINSの2つのビューで参照できます。
- ストールしたチェーンの処理
  ステップの完了時には、常にチェーン・ルールが評価され、次に実行するステップが判別されます。いずれのルールでも別のステップが開始されず、チェーンが終了せず、チェーンのevaluation_intervalがNULLの場合は、チェーンがstalled状態になります。

関連項目:

- チェーンの概要については、「チェーン」
- 「チェーンの作成例」

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.6.1 ジョブ・チェーンの作成と管理について

ジョブ・チェーンを使用して、前の1つ以上のジョブの結果に応じて異なるジョブが開始される依存性ベースのスケジューリングを実

装できます。

チェーンを作成して使用するには、次のタスクを実行します。

タスク	参照
1. チェーン・オブジェクトを作成します。	チェーンの作成
2. チェーン内のステップを定義します。	チェーン・ステップの定義
3. ルールを追加します。	チェーンへのルールの追加
4. チェーンを使用可能にします。	チェーンの有効化
5. チェーンを指し示すジョブ(チェーン・ジョブ)を作成します。	チェーン用のジョブの作成

親トピック: ジョブ・チェーンの作成と管理

29.6.2 チェーンのタスクとそのプロシージャ

DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なチェーン・タスクを管理します。

表29-6に、チェーンに関する一般的なタスクとそれに対応するプロシージャを示します。

表29-6 チェーンのタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
チェーンの作成	CREATE_CHAIN	所有者の場合、CREATE JOB、CREATE EVALUATION CONTEXT、CREATE RULE および CREATE RULE SET。それ以外の場合、CREATE ANY JOB、CREATE ANY RULE、CREATE ANY RULE SET および CREATE ANY EVALUATION CONTEXT。
チェーンの削除	DROP_CHAIN	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB 権限。所有者以外の場合は、DROP ANY EVALUATION CONTEXT および DROP ANY RULE SET も必要です。
チェーンの変更	SET_ATTRIBUTE	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB。
実行中チェーンの変更	ALTER_RUNNING_CHAIN	ジョブの所有権、あるいはジョブに対する ALTER 権限または CREATE ANY JOB。

タスク	プロシージャ	必要な権限
チェーンの実行	RUN_CHAIN	CREATE JOB または CREATE ANY JOB また、新規ジョブの所有者には、チェーンに対する EXECUTE 権限または EXECUTE ANY PROGRAM が必要です。
チェーンへのルールの追加	DEFINE_CHAIN_RULE	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB 権限。CREATE RULE(チェーンの所有者の場合)、CREATE ANY RULE(チェーンの所有者以外の場合)。
チェーン内のルールの変更	DEFINE_CHAIN_RULE	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB 権限。チェーンの所有者以外の場合は、そのルールに対する ALTER 権限または ALTER ANY RULE が必要です。
チェーンからのルールの削除	DROP_CHAIN_RULE	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB 権限。DROP ANY RULE(チェーンの所有者以外の場合)。
チェーンの有効化	ENABLE	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB。
チェーンの無効化	DISABLE	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB。
ステップの作成	DEFINE_CHAIN_STEP	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB。
ステップの削除	DROP_CHAIN_STEP	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB。
ステップの変更(ステップへの追加の属性値の割当てを含む)	ALTER_CHAIN	チェーンの所有権、あるいはチェーンに対する ALTER 権限または CREATE ANY JOB。

親トピック: ジョブ・チェーンの作成と管理

29.6.3 チェーンの作成

チェーンを作成するには、DBMS_SCHEDULERパッケージのCREATE_CHAINプロシージャを使用します。

最初に、必要な権限があることを確認してください。詳細は、「チェーンの各権限の設定」を参照してください。

CREATE_CHAINを使用してチェーン・オブジェクトを作成した後、チェーン・ステップおよびチェーン・ルールを別々に定義します。

次の例では、チェーンを作成しています。

```
BEGIN
DBMS_SCHEDULER.CREATE_CHAIN (
   chain_name          => 'my_chain1',
   rule_set_name       => NULL,
   evaluation_interval => NULL,
   comments            => 'My first chain');
END;
/
```

rule_set_nameおよびevaluation_interval引数は通常NULLのままです。evaluation_intervalでは、チェーン・ルールが評価される繰返し間隔を定義できます。rule_set_nameは、Oracle Streams内で定義されたルール・セットです。

関連項目:

- evaluation_interval属性の詳細は、「チェーンへのルールの追加」
- CREATE_CHAINの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.4 チェーン・ステップの定義

チェーン・オブジェクトの作成後、1つ以上のチェーン・ステップを定義します。

各ステップは、次のいずれかを指し示します。

- スケジューラ・プログラム・オブジェクト(プログラム)
- 別のチェーン(ネストしたチェーン)
- イベント・スケジュール、インライン・イベントまたはFile Watcher

プログラムまたはネストしたチェーンを指し示すステップを定義するには、DEFINE_CHAIN_STEPプロシージャを使用します。次の例では、my_chain1に2つのステップを追加しています。

```
BEGIN
  DBMS_SCHEDULER.DEFINE_CHAIN_STEP (
   chain_name      =>  'my_chain1',
   step_name       =>  'my_step1',
   program_name    =>  'my_program1');
  DBMS_SCHEDULER.DEFINE_CHAIN_STEP (
   chain_name      =>  'my_chain1',
   step_name       =>  'my_step2',
   program_name    =>  'my_chain2');
END;
/
```

ステップを定義する際に、名前付きプログラムまたはチェーンが存在している必要はありません。ただし、チェーンの実行時には存在して有効になっている必要があり、そうでないとエラーが発生します。

イベントの発生を待機するステップを定義するには、DEFINE_CHAIN_EVENT_STEPプロシージャを使用します。プロシージャの引数を使用して、イベント・スケジュールを指し示すか、インラインのキュー仕様およびイベント条件を組み込むか、またはFile Watcherの名前を指定できます。この例では、名前付きイベント・スケジュール内に指定されているイベントを待機する、3番

目のチェーン・ステップが作成されます。

```
BEGIN
  DBMS_SCHEDULER.DEFINE_CHAIN_EVENT_STEP (
   chain_name           =>  'my_chain1',
   step_name            =>  'my_step3',
   event_schedule_name  =>  'my_event_schedule');
END;
/
```

イベント・ステップでは、ステップが開始されるまでそのイベントを待機することはありません。

ローカルの外部実行可能ファイルを実行するステップ

ローカルの外部実行可能ファイルを実行するステップを定義した後、次の例に示すように、ALTER_CHAINプロシージャを使用して、ステップに資格証明を割り当てる必要があります。

```
BEGIN
  DBMS_SCHEDULER.ALTER_CHAIN('chain1','step1','credential_name','MY_CREDENTIAL');
END;
/
```

リモートの宛先で実行するステップ

リモート・ホストまたはリモート・データベース上のデータベース・プログラム・ユニットで外部実行可能ファイルを実行するステップを定義した後、次の例に示すように、ALTER_CHAINプロシージャを使用して、ステップに資格証明と宛先の両方を割り当てる必要があります。

```
BEGIN
 DBMS_SCHEDULER.ALTER_CHAIN('chain1','step2','credential_name','DW_CREDENTIAL');
 DBMS_SCHEDULER.ALTER_CHAIN('chain1','step2','destination_name','DBHOST1_ORCLDW');
END;
/
```

ステップを再開可能にする方法

データベース・リカバリ後、デフォルトでは、実行中だったステップがSTOPPEDとしてマークされ、チェーンが続行されます。ALTER_CHAINを使用してチェーン・ステップのrestart_on_recovery属性をTRUEに設定すると、これらのステップをデータベース・リカバリ後に自動的に再開するように指定できます。

DEFINE_CHAIN_STEP、DEFINE_CHAIN_EVENT_STEPおよびALTER_CHAINプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

関連項目:

- 「イベントについて」
- 「File Watcherについて」
- 「資格証明」
- 「宛先」

親トピック: ジョブ・チェーンの作成と管理

29.6.5 チェーンへのルールの追加

チェーンにルールを追加するには、DBMS_SCHEDULERパッケージのDEFINE_CHAIN_RULEプロシージャを使用します。この

プロシージャは、チェーンに追加する各ルールに対して1回ずつコールします。

チェーン・ルールでは、ステップの実行時期、およびステップ間の依存関係が定義されます。各ルールには条件と処理があります。ルールが評価されるときに、ルールの条件がTRUEに評価されると、その処理が実行されます。条件では、スケジューラ・チェーン条件の構文、またはSQLのWHERE句で有効な構文を使用できます。この構文には、ステップの完了ステータスなど、チェーン・ステップの属性への参照を組み込むことができます。標準な処理は、指定したステップを実行すること、またはステップのリストを実行することです。

チェーン・ルールはすべてまとまって機能し、チェーン全体の処理を定義します。チェーン・ジョブの開始時および各ステップの終了時に、次に実行される処理を判断するためにすべてのルールが評価されます。複数のルールにTRUE条件がある場合は、複数の処理を発生させることができます。チェーンのevaluation_interval属性を設定して、ルールを一定間隔で評価させることもできます。

条件は通常、1つ以上の先行するステップの結果に基づきます。たとえば、先行する2つのステップが成功した場合はあるステップを実行し、2つのステップのいずれかが失敗した場合には別のステップを実行する場合があります。

スケジューラのチェーン条件構文には、次の2つの書式があります。

```
stepname [NOT] {SUCCEEDED|FAILED|STOPPED|COMPLETED}
stepname ERROR_CODE {comparision_operator|[NOT] IN} {integer|list_of_integers}
```

条件をブール演算子AND、ORおよびNOT( )と組み合せて条件式を作成できます。式にカッコを使用すると、評価順序を指定できます。

ステップに割り当てたプログラム内でRAISE_APPLICATION_ERROR PL/SQL文を使用して、ERROR_CODEを設定できます。この方法でプログラムにより設定されるエラー・コードは負の数値ですが、チェーン・ルール内でERROR_CODEをテストするときには正の数値をテストします。たとえば、プログラムに次の文が含まれている場合を考えます。

```
RAISE_APPLICATION_ERROR(-20100, errmsg);
```

チェーン・ルールの条件を次のように指定する必要があります。

```
stepname ERROR_CODE=20100
```

ステップ属性

次のリストに、SQL WHERE句の構文を使用する際に条件に含めることのできるステップ属性を示します。

- completed
- state
- start_date
- end_date
- error_code
- duration
-

completed属性はブール値で、state属性がSUCCEEDED、FAILEDまたはSTOPPEDのときにはTRUEです。

表29-7に、state属性に使用される値を示します。これらの値は、*_SCHEDULER_RUNNING_CHAINSビューのSTATE列で参照可能です。

表29-7 チェーン・ステップのstate属性の値

state属性の値	意味

state属性の値	意味
NOT_STARTED	ステップのチェーンは実行中ですが、ステップはまだ開始されていません。
SCHEDULED	ルールによって AFTER 句を使用してステップが開始されており、指定の待機時間はまだ満了していません。
RUNNING	ステップは実行中です。イベント・ステップの場合、ステップは開始されており、イベントを待機中です。
PAUSED	ステップの PAUSE 属性が TRUE に設定されて、ステップが一時停止します。一時停止を解除しないと、それに依存するステップは開始できません。
SUCCEEDED	ステップは正常に完了しています。ステップの ERROR_CODE は 0 です。
FAILED	ステップはエラーで完了しました。ERROR_CODE は 0(ゼロ)以外の値です。
STOPPED	ステップは STOP_JOB プロシージャで停止されました。
STALLED	ステップはネストしたチェーンであり、停止されています。

SQL WHERE句の構文のルールと例は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDEFINE_CHAIN_RULEプロシージャに関する項を参照してください。

スケジューラ・チェーン条件の構文を使用した条件の例

次の例では、スケジューラ・チェーン条件の構文を使用しています。

次の条件を含んだルールによって開始されたステップは、form_validation_stepというステップが完了(SUCCEEDED、FAILEDまたはSTOPPED)すると開始されます。

```
form_validation_step COMPLETED
```

次の条件も同様ですが、条件が満たされるにはステップが成功する必要があることを示しています。

```
form_validation_step SUCCEEDED
```

次の条件では、エラーがあるかどうかをテストしています。ステップform_validation_stepが失敗して20001以外のエラー・コードが戻された場合は、TRUEになります。

```
form_validation_step FAILED AND form_validation_step ERROR_CODE != 20001
```

他の例は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDEFINE_CHAIN_RULEプロシージャに関する項を参照してください。

SQL WHERE構文を使用した条件の例

```
':step1.state=''SUCCEEDED'''
```

チェーンの開始

チェーン・ジョブの開始時にチェーンを開始できるように、少なくとも1つのルールに、常にTRUEに評価される条件が必要です。このための最も簡単な方法は、スケジュール・チェーン条件の構文を使用している場合は条件を'TRUE'に設定すること、またはSQL構文を使用している場合は条件を'1=1'に設定することです。

チェーンの終了

少なくとも1つのチェーン・ルールに、'END'のactionが含まれている必要があります。チェーン・ジョブは、END処理が含まれているルールの1つがTRUEに評価されるまでは完了しません。通常は、異なるEND処理が設定された別々のルールがあり、エラー・コードを伴う場合と伴わない場合があります。

チェーンに実行するステップがなくなり、イベントの発生の待機中ではなく、END処理が含まれているルールでTRUEに評価されるルールがない(またはEND処理が設定されているルールがない)場合、チェーン・ジョブの状態はCHAIN_STALLEDになります。詳細は、「ストールしたチェーンの処理」を参照してください。

ルールの定義例

次の例では、step1でチェーンを開始するルールおよびstep1の完了時にstep2を開始するルールを定義しています。rule_nameおよびcommentsはオプションですが、デフォルトでNULLになります。rule_nameを使用する場合は、DEFINE_CHAIN_RULEに対する別の呼出しで、そのルールを後で再定義できます。新しい定義で、以前の定義が上書きされます。

```
BEGIN
DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   chain_name    =>   'my_chain1',
   condition     =>   'TRUE',
   action        =>   'START step1',
   rule_name     =>   'my_rule1',
   comments      =>   'start the chain');
DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   chain_name    =>   'my_chain1',
   condition     =>   'step1 completed',
   action        =>   'START step2',
   rule_name     =>   'my_rule2');
END;
/
```

関連項目:

- DEFINE_CHAIN_RULEプロシージャおよびスケジューラのチェーン条件構文の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- 「チェーンの作成例」

親トピック: ジョブ・チェーンの作成と管理

29.6.6 チェーン・ルールの評価間隔の設定

スケジューラでは、チェーン・ジョブの開始時と各チェーン・ステップの終了時に、すべてのチェーン・ルールが評価されます。

1時間に1回など、一定の時間間隔でスケジューラによってルールが評価されるようにチェーンを構成することもできます。この機能は、時間に基づいて、またはチェーン外部での発生に基づいてチェーン・ステップを開始するのに役立ちます。いくつか例を挙げます。

- チェーン・ステップがリソース集中型であるため、オフピーク時に実行する必要があるとします。ステップの条件として、別のステップの完了と午後6時から午前0時までの時間の両方を設定できます。スケジューラでは、この条件がTRUEになる

時期を判別するために、ルールをその都度評価する必要があります。

- ステップが、チェーン外部にある他のプロセスから表にデータが到着するまで待機する必要があるとします。このステップの条件として、別のステップの完了と行を含む特定の表の両方を設定できます。スケジューラでは、この条件がTRUEになる時期を判別するために、ルールをその都度評価する必要があります。この条件はSQL WHERE句構文を使用し、次のようになります。

  ```
  ':step1.state=''SUCCEEDED'' AND select count(*) from oe.sync_table > 0'
  ```

チェーンの評価間隔を設定するには、チェーンの作成時にevaluation_interval属性を設定します。この属性のデータ型はINTERVAL DAY TO SECONDです。

```
BEGIN
  DBMS_SCHEDULER.CREATE_CHAIN (
   chain_name          => 'my_chain1',
   rule_set_name       => NULL,
   evaluation_interval => INTERVAL '30' MINUTE,
   comments            => 'Chain with 30 minute evaluation interval');
END;
/
```

親トピック: ジョブ・チェーンの作成と管理

29.6.7 チェーンの有効化

チェーンを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャを使用します。ジョブでチェーンを実行するには、チェーンを使用可能にする必要があります。すでに使用可能なチェーンを使用可能にしてもエラーは戻りません。

この例では、チェーンmy_chain1が使用可能になります。

```
BEGIN
  DBMS_SCHEDULER.ENABLE ('my_chain1');
END;
/
```

ENABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ノート:

チェーンは、次のいずれかが削除された場合にスケジューラによって自動的に使用禁止にされます。

- いずれかのチェーン・ステップが指し示しているプログラム
- いずれかのチェーン・ステップが指し示しているネストしたチェーン
- いずれかのチェーン・イベント・ステップが指し示しているイベント・スケジュール

親トピック: ジョブ・チェーンの作成と管理

29.6.8 チェーン用のジョブの作成

チェーンを実行するには、DBMS_SCHEDULERパッケージのRUN_CHAINプロシージャを使用するか、CHAINタイプのジョブ(チェーン・ジョブ)を作成およびスケジュールする必要があります。

ジョブの処理では、次の例に示すように、以前に作成されたチェーン名を参照する必要があります。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name        => 'chain_job_1',
   job_type        => 'CHAIN',
   job_action      => 'my_chain1',
   repeat_interval => 'freq=daily;byhour=13;byminute=0;bysecond=0',
   enabled         => TRUE);
END;
/
```

実行中のチェーン・ジョブの各ステップには、スケジューラによって、チェーン・ジョブと同じジョブ名と所有者を設定したステップ・ジョブが作成されます。各ステップ・ジョブにはさらに、そのジョブを一意に識別するためのサブ名があります。ジョブのサブ名は、ビュー*_SCHEDULER_RUNNING_JOBS、*_SCHEDULER_JOB_LOGおよび*_SCHEDULER_JOB_RUN_DETAILSに列として表示できます。また、次の場合を除いて、ジョブのサブ名は通常はステップ名と同じです。

- ネストされたチェーンの場合、現在のステップ名がすでにジョブのサブ名として使用されている場合があります。この場合、スケジューラは、'_N'をステップ名の最後に追加しますが、ここで、Nはジョブのサブ名を一意にするための整数を表します。
- ステップ・ジョブの作成時にエラーが発生した場合、スケジューラは、ジョブ・サブ名を'step_name_0'に設定して、ジョブ・ログ・ビュー(*_SCHEDULER_JOB_LOGおよび*_SCHEDULER_JOB_RUN_DETAILS)にFAILEDエントリを記録します。

関連項目:

- CREATE_JOBプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- チェーン・ジョブを作成せずにチェーンを実行する別の方法は、「チェーンの実行」

親トピック: ジョブ・チェーンの作成と管理

29.6.9 チェーンの削除

ステップおよびルールを含むチェーンを削除するには、DBMS_SCHEDULERパッケージのDROP_CHAINプロシージャを使用します。

次の例では、my_chain1という名前のチェーンを削除しています。

```
BEGIN
  DBMS_SCHEDULER.DROP_CHAIN (
   chain_name    => 'my_chain1',
   force         => TRUE);
END;
/
```

DROP_CHAINプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.10 チェーンの実行

チェーンを即時に実行するには、DBMS_SCHEDULERパッケージのRUN_JOBまたはRUN_CHAINプロシージャを使用します。

すでにチェーン用のチェーン・ジョブを作成している場合は、RUN_JOBプロシージャを使用してそのジョブを実行(したがってチェーンを実行)できますが、RUN_JOBのuse_current_session引数をFALSEに設定する必要があります。

RUN_CHAINプロシージャを使用すると、チェーンに対するチェーン・ジョブをあらかじめ作成することなく、チェーンを実行できます。また、RUN_CHAINを使用してチェーンの一部のみを実行することもできます。

RUN_CHAINは、指定したチェーンを実行する一時ジョブを作成します。ジョブ名を指定すると、その名前のジョブが作成されますが、それ以外の場合はデフォルトのジョブ名が割り当てられます。

開始ステップ・リストを指定すると、チェーンの実行開始時にそれらのステップのみが開始されます (通常開始されるステップは、それらがリスト内にない場合は実行されません)。開始ステップのリストが提供されない場合は、チェーンが通常どおり開始されます。つまり、初期評価が行われ、実行を開始するステップが判断されます。次の例では、my_chain1を即時に実行します。

```
BEGIN
  DBMS_SCHEDULER.RUN_CHAIN (
   chain_name    =>  'my_chain1',
   job_name      =>  'partial_chain_job',
   start_steps   =>  'my_step2, my_step4');
END;
/
```

関連項目:

- 「チェーンの一部実行」
- RUN_CHAINプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.11 チェーン・ルールの削除

チェーンからルールを削除するには、DBMS_SCHEDULERパッケージのDROP_CHAIN_RULEプロシージャを使用します。

次の例では、my_rule1を削除しています。

```
BEGIN
  DBMS_SCHEDULER.DROP_CHAIN_RULE (
   chain_name    =>    'my_chain1',
   rule_name     =>    'my_rule1',
   force         =>    TRUE);
END;
/
```

DROP_CHAIN_RULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.12 チェーンの無効化

チェーンを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャを使用します。

次の例では、my_chain1を使用禁止にしています。

```
BEGIN
  DBMS_SCHEDULER.DISABLE ('my_chain1');
```

```
END;
/
```

DISABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ノート:

チェーンは、次のいずれかが削除された場合にスケジューラによって自動的に使用禁止にされます。

- いずれかのチェーン・ステップが指し示しているプログラム
- いずれかのチェーン・ステップが指し示しているネストしたチェーン
- いずれかのチェーン・イベント・ステップが指し示しているイベント・スケジュール

親トピック: ジョブ・チェーンの作成と管理

29.6.13 チェーン・ステップの削除

チェーンからステップを削除するには、DBMS_SCHEDULERパッケージのDROP_CHAIN_STEPプロシージャを使用します。

次の例では、my_chain2からmy_step2を削除しています。

```
BEGIN
  DBMS_SCHEDULER.DROP_CHAIN_STEP (
   chain_name    =>    'my_chain2',
   step_name     =>    'my_step2',
   force         =>     TRUE);
END;
/
```

DROP_CHAIN_STEPプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.14 チェーンの停止

実行中のチェーンを停止するには、DBMS_SCHEDULER.STOP_JOBプロシージャをコールして、チェーン・ジョブ(チェーンを開始したジョブ)の名前を渡します。

チェーン・ジョブを停止すると、実行中のチェーンの全ステップが停止され、チェーンが終了します。

STOP_JOBプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.15 個々のチェーン・ステップの停止

ルールの条件が満たされたときに1つ以上のステップが停止するチェーン・ルールを作成するか、STOP_JOBプロシージャを呼び出すことにより、個々のチェーン・ステップを停止できます。

停止するステップごとに、スキーマ名、チェーン・ジョブ名およびステップ・ジョブ・サブ名を指定する必要があります。

```
BEGIN
```

```
  DBMS_SCHEDULER.STOP_JOB('oe.chainrunjob.stepa');
END;
/
```

この例では、chainrunjobはチェーン・ジョブ名で、stepaはステップ・ジョブ・サブ名です。通常、ステップ・ジョブ・サブ名はステップ名と同じですが、異なる場合もあります。ステップ・ジョブ・サブ名は、*_SCHEDULER_RUNNING_CHAINSビューのSTEP_JOB_SUBNAME列から取得できます。

チェーン・ステップを停止すると、stateがSTOPPEDに設定され、チェーン・ルールが評価されて次に実行するステップが判別されます。

STOP_JOBプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.16 チェーンの一時停止

チェーン全体、またはチェーンの個別ブランチを一時停止できます。そのためには、DBMS_SCHEDULER.ALTER_CHAINまたはALTER_RUNNING_CHAINプロシージャで、1つ以上のステップのPAUSE属性をTRUEに設定します。

チェーン・ステップを一時停止すると、チェーンの実行をそのステップの実行後に一時停止できます。

ステップを一時停止すると、そのステップの実行後にstate属性がPAUSEDに変わりますが、completed属性はFALSEのままです。そのため、一時停止したステップの完了に依存しているステップは実行されません。一時停止したステップのPAUSE属性をFALSEにリセットすると、state属性が完了状態(SUCCEEDED、FAILEDまたはSTOPPED)に設定され、一時停止したステップの完了を待機しているステップを実行できるようになります。

図29-1 ステップ3で一時停止したチェーン

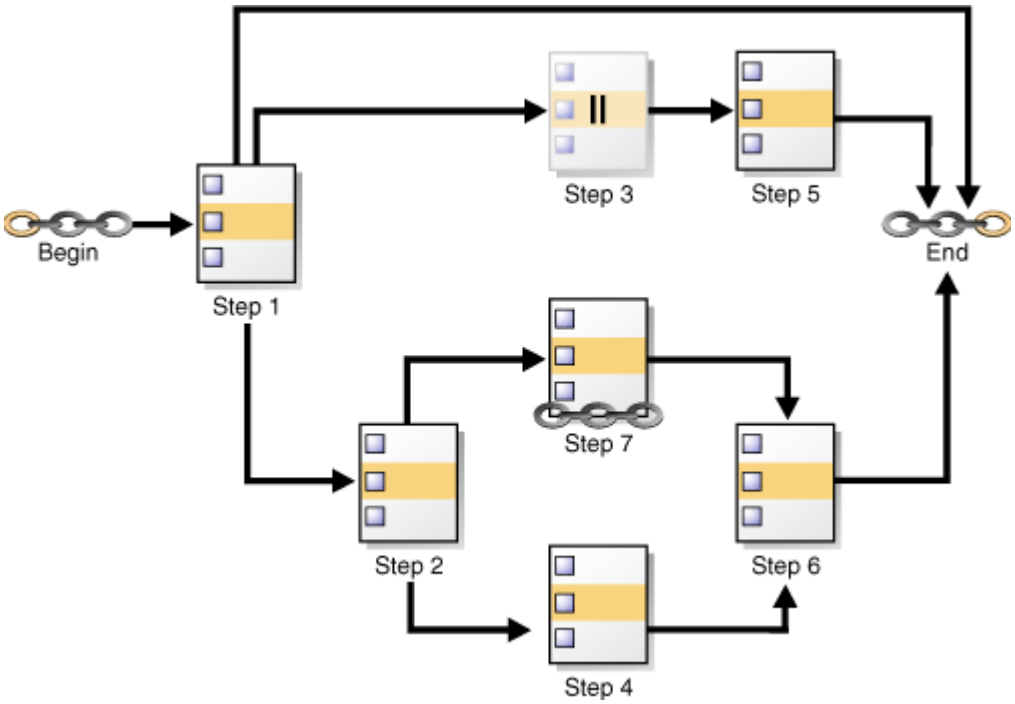


図29-1では、ステップ3が一時停止されています。ステップ3の一時停止が解除されるまで、ステップ5は実行されません。ステップ2のみを一時停止した場合、ステップ4、6および7は実行されません。ただし、ステップ1、3および5は実行できます。いずれの場合も、チェーンのブランチを1つのみ一時停止することになります。

チェーン全体を一時停止するには、チェーンのステップをすべて一時停止します。チェーンの一時停止を解除するには、チェーン・ステップの1つ、多数またはすべての一時停止を解除します。図29-1のチェーンでは、ステップ1を一時停止すると、ステップ1の

実行後にチェーン全体も一時停止します。

関連項目:

『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_SCHEDULER.ALTER_CHAINおよびDBMS_SCHEDULER.ALTER_RUNNING_CHAINプロシージャに関する項を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.17 チェーン・ステップのスキップ

チェーンの1つ以上のステップをスキップできます。そのためには、DBMS_SCHEDULER.ALTER_CHAINまたはALTER_RUNNING_CHAINプロシージャで、1つ以上のステップのSKIP属性をTRUEに設定します。

ステップのSKIP属性がTRUEの場合、そのステップを実行するためのチェーン条件を満たすと、そのステップは実行されるかわりに、即時に成功した場合と同様に処理されます。SKIPをTRUEに設定しても、実行中のステップ、遅延後に実行するようにスケジュールされているステップまたは実行済のステップには影響しません。

ステップのスキップが特に役立つのは、チェーンのテスト時です。たとえば、図29-1に示したチェーンをテストする際に、ステップ7をスキップすると、このステップはネストしたチェーンであるため、テスト時間を大幅に短縮できます。

関連項目:

「チェーン・ステップのスキップ」

親トピック: ジョブ・チェーンの作成と管理

29.6.18 チェーンの一部実行

チェーンの一部のみを実行することができます。

チェーンの一部のみを実行するには2つの方法があります。

- ALTER_CHAINプロシージャを使用して、1つ以上のステップに対してPAUSE属性をTRUEに設定してから、RUN_JOBでチェーン・ジョブを起動するか、RUN_CHAINでチェーンを起動します。一時停止しているステップに依存するステップは実行されませんが、一時停止しているステップは実行されます。

  この方法のデメリットは、チェーンを将来実行する場合に備えて、影響を受けるステップのPAUSE属性の設定をFALSEに戻す必要があることです。

- RUN_CHAINプロシージャを使用して、実行しないステップをスキップし、チェーンの特定ステップのみを開始します。

  この方法の方が簡単であり、ステップの開始前に初期状態を設定することもできます。

この2つの方法の両方を使用して、チェーンの開始時と終了時の両方でステップをスキップすることが必要な場合もあります。

詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のRUN_CHAINプロシージャに関する項を参照してください。

親トピック: ジョブ・チェーンの作成と管理

29.6.19 実行中のチェーンの監視

実行中のチェーンの状態は、*_SCHEDULER_RUNNING_JOBSおよび*_SCHEDULER_RUNNING_CHAINSの2つのビューで参照できます。

*_SCHEDULER_RUNNING_JOBSビューには、チェーン・ジョブに関する1行と実行中の各ステップに関する1行が含まれています。*_SCHEDULER_RUNNING_CHAINSビューには、各チェーン・ステップ(ネストしたチェーンを含む)に関する1行と、各ステップの実行ステータス(NOT_STARTED、RUNNING、STOPPED、SUCCEEDEDなど)が含まれています。

関連項目:

- *_SCHEDULER_RUNNING_JOBSビューの詳細は、『Oracle Databaseリファレンス』を参照してください
- *_SCHEDULER_RUNNING_CHAINSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: ジョブ・チェーンの作成と管理

29.6.20 ストールしたチェーンの処理

ステップの完了時には、常にチェーン・ルールが評価され、次に実行するステップが判別されます。いずれのルールでも別のステップが開始されず、チェーンが終了せず、チェーンのevaluation_intervalがNULLの場合は、チェーンがstalled状態になります。

チェーンが停止状態になっている場合、実行中のステップはありません。また、(指定の時間間隔だけ待機した後に)実行するようにスケジュールされているステップや、イベントを待機中のイベント・ステップもありません。この場合、チェーンを実行しているジョブの状態はCHAIN_STALLEDに設定されます。ただし、ジョブは*_SCHEDULER_RUNNING_JOBSビューにリストされたままです。

停止状態のチェーンの問題を解決するには、ビューALL_SCHEDULER_RUNNING_CHAINS(チェーン(ネストしたチェーンを含む)内のすべてのステップの状態が示されます)とビューALL_SCHEDULER_CHAIN_RULES(すべてのチェーン・ルールが含まれます)が使用されます。

ALTER_RUNNING_CHAINプロシージャを使用していずれかのステップのstateを変更すると、チェーンの実行を継続できます。たとえば、ステップ11がステップ9の正常終了まで待機してから開始するようになっており、状態の変更がこれに関係する場合、ステップ9のstateを'SUCCEEDED'に設定することもできます。

あるいは、1つ以上のルールが正しくない場合は、DEFINE_CHAIN_RULEプロシージャを使用して、(同じルール名を使用して)それらのルールを置換するか、または新規ルールを作成できます。新規および更新したルールは、実行中のチェーンと次回以降のすべてのチェーン実行に適用されます。ルールの追加または更新後は、停止状態のチェーン・ジョブでEVALUATE_RUNNING_CHAINを実行し、必要な処理をトリガーする必要があります。

親トピック: ジョブ・チェーンの作成と管理

29.7 非互換性定義の使用

非互換性定義(または非互換性)は、互換性のないジョブまたはプログラムを指定し、該当する場合は一度に1つのグループのみを実行できるようにします。

- ジョブまたはプログラムの非互換性の作成
  ジョブ・レベルまたはプログラム・レベルの非互換性を指定するには、DBMS_SCHEDULERパッケージのCREATE_INCOMPATIBILITYプロシージャを使用します。

- 非互換性へのジョブまたはプログラムの追加
  既存の非互換性定義にジョブまたはプログラムを追加するには、DBMS_SCHEDULERパッケージのADD_TO_INCOMPATIBILITYプロシージャを使用します。
- 非互換性からのジョブまたはプログラムの削除
  既存の非互換性定義からジョブまたはプログラムを削除するには、DBMS_SCHEDULERパッケージのREMOVE_FROM_INCOMPATIBILITYプロシージャを使用します。
- 非互換性の削除
  既存の非互換性定義を削除するには、DBMS_SCHEDULERパッケージのDROP_INCOMPATIBILITYプロシージャを使用します。

関連項目:

- 非互換性
- 『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のDBMS_SCHEDULERに関する項

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.7.1 ジョブまたはプログラムの非互換性の作成

ジョブ・レベルまたはプログラム・レベルの非互換性を指定するには、DBMS_SCHEDULERパッケージのCREATE_INCOMPATIBILITYプロシージャを使用します。

たとえば、次の文はjob1、job2またはjob3というジョブのいずれかのみが一度に実行できることを指定するincompat1という非互換性を作成します。

```
BEGIN
dbms_scheduler.create_incompatibility(
  incompatibility_name => 'icompat1',
  object_name => 'job1,job2,job3',
  enabled => true );
END;
/
```

object_nameには、相互に互換性がない(つまり、一度に実行できない)すべてのプログラムまたはすべてのジョブのカンマ区切りのリストを指定します。ジョブの場合、リストは複数のジョブで構成され、constraint_levelは'JOB_LEVEL' (デフォルト。例には含まれていません)である必要があります。プログラムの場合、constraint_levelには'JOB_LEVEL'または'PROGRAM_LEVEL'を指定できます。デフォルト値の'JOB_LEVEL'を設定した場合、object_nameに指定されているプログラム(複数も可)に基づく単一ジョブのみを同時に実行できます。PROGRAM_LEVELを設定した場合、プログラムに互換性はないが、同じプログラムに基づくジョブには互換性があります。

たとえば、object_nameの値が'P1,P2,P3'で、constraint_levelが'PROGRAM_LEVEL'である場合、P1に基づく多数のジョブを同時に実行できますが、P1ベースのジョブが実行されている場合、P2またはP3に基づくジョブは実行できません。同様に、P3に基づく多数のジョブを同時に実行できますが、P1またはP2に基づくジョブは実行できません。constraint_levelに'JOB_LEVEL'が設定されている場合、特定の時点で実行できるのは、プログラムP1、P2およびP3に基づくすべてのジョブのうちの単一のジョブのみです。

関連項目:

『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のCREATE INCOMPATIBILITYプロシージャに関する項

親トピック: 非互換性定義の使用

29.7.2 非互換性へのジョブまたはプログラムの追加

既存の非互換性定義にジョブまたはプログラムを追加するには、DBMS_SCHEDULERパッケージのADD_TO_INCOMPATIBILITYプロシージャを使用します。

たとえば、次の文はicomp1234という非互換性にジョブjob1を追加します。

```
BEGIN
dbms_scheduler.add_to_incompatibility(
  incompatibility_name => 'icomp1234',
  object_name => 'job1');
END;
/
```

incompatibility_nameは既存の非互換性定義の名前です。

object_nameには、ジョブまたはプログラムのカンマ区切りのリストが含まれています。

指定した追加するジョブまたはプログラムが指定した非互換性定義にすでに含まれていても、このプロシージャはエラーを生成しません。

関連項目:

『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のADD_TO_INCOMPATIBILITYプロシージャに関する項

親トピック: 非互換性定義の使用

29.7.3 非互換性からのジョブまたはプログラムの削除

既存の非互換性定義からジョブまたはプログラムを削除するには、DBMS_SCHEDULERパッケージのREMOVE_FROM_INCOMPATIBILITYプロシージャを使用します。

たとえば、次の文はicomp1234という非互換性からジョブjob1を削除します。

```
BEGIN
dbms_scheduler.remove_from_incompatibility(
  incompatibility_name => 'icomp1234',
  object_name => 'job1');
END;
/
```

incompatibility_nameは既存の非互換性定義の名前です。

object_nameには、ジョブまたはプログラムのカンマ区切りのリストが含まれています。

指定した削除するジョブまたはプログラムが指定した非互換性定義に存在しない場合、このプロシージャはエラーを生成しません。

関連項目:

『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』の
REMOVE_FROM_INCOMPATIBILITYプロシージャに関する項

親トピック: 非互換性定義の使用

29.7.4 非互換性の削除

既存の非互換性定義を削除するには、DBMS_SCHEDULERパッケージのDROP_INCOMPATIBILITYプロシージャを使用します。

たとえば、次の文は非互換性icomp1234を削除します。

```
BEGIN
dbms_scheduler.drop_incompatibility(
  incompatibility_name => 'icomp1234';
END;
/
```

incompatibility_nameは既存の非互換性定義の名前です。

関連項目:

『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のDROP_INCOMPATIBILITYプロシージャに関する項

親トピック: 非互換性定義の使用

29.8 ジョブ・リソースの管理

ジョブが使用できるリソースを作成および変更したり、ジョブが使用できる指定されたリソースの量を制御したりできます。

お客様にはリソースにアクセスする必要があるジョブがあります。使用できるそのようなリソースの数は限られているため、スケジュール・システムは各リソースおよびそれを使用しているジョブを追跡し、必要なリソースが使用可能になるまでジョブをスケジュールしないでおく必要があります。

- リソースの作成または削除
  リソースを作成するには、DBMS_SCHEDULERパッケージのCREATE_RESOURCEプロシージャを使用します。
- リソースの変更
  リソースを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEおよびSET_ATTRIBUTE_NULLプロシージャを使用します。
- ジョブのリソース制約の設定
  ジョブまたはプログラムが使用するリソースを指定するには、DBMS_SCHEDULERパッケージのSET_RESOURCE_CONSTRAINTプロシージャを使用します。

関連項目:

- 『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のDBMS_SCHEDULERに関する項

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.8.1 リソースの作成または削除

リソースを作成するには、DBMS_SCHEDULERパッケージのCREATE_RESOURCEプロシージャを使用します。

リソースは、リソースを作成しているユーザーのスキーマに作成されます。

たとえば、次の文はmy_resourceという名前のリソースを作成して、3ユニットのリソースが内部的に使用可能になるように指定し、スケジューラは3ユニットを超えるリソースがジョブによって同時に使用されないように制約を管理します。

```
BEGIN
   DBMS_SCHEDULER.CREATE_RESOURCE(
      resource_name => 'my_resource',
      units         => 3,
      state         => 'ENFORCE_CONSTRAINTS',
      comments      => 'Resource1'
   )
END;
/
```

リソースが必要なくなった場合は、DBMS_SCHEDULERパッケージのDROP_RESOURCEプロシージャを使用して削除できます。たとえば:

```
BEGIN
   DBMS_SCHEDULER.DROP_RESOURCE(
      resource_name => 'my_resource',
      force         => true
   )
END;
/
```

関連項目:

『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のCREATE_RESOURCEプロシージャに関する項

親トピック: ジョブ・リソースの管理

29.8.2 リソースの変更

リソースを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEおよびSET_ATTRIBUTE_NULLプロシージャを使用します。

リソースを変更した場合、その変更はこのリソースを使用する現在実行されているジョブには影響しません。変更はそのリソースを使用する後続のジョブで有効になります。

関連項目:

『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のSET_ATTRIBUTEプロシージャおよびSET_ATTRIBUTE_NULLプロシージャに関する項

親トピック: ジョブ・リソースの管理

29.8.3 ジョブのリソース制約の設定

ジョブまたはプログラムが使用するリソースを指定するには、DBMS_SCHEDULERパッケージの

SET_RESOURCE_CONSTRAINTプロシージャを使用します。

指定したジョブやプログラムが使用できるリソースのユニット数を指定できます。

たとえば、次の文はjob1というオブジェクトがresource1というリソースの1ユニットを使用できることを指定します。

```
BEGIN
    DBME_SCHEDULER.SET_RESOURCE_CONSTRAINT(
         OBJECT_NAME   => 'job1',
         RESOURCE_NAME => 'resource1',
         UNITS         =>1);
END;
/
```

object_nameパラメータには、プログラムまたはジョブの名前、あるいは名前のカンマ区切りのリストを指定できます。

unitsパラメータは、このプログラムまたはジョブが使用できるリソースのユニット数を指定します。unitsに0を設定した場合、プログラムやジョブはこのリソースを使用しなくなり、制約が削除されることを意味します。以前は制約がなかったリソースでunitsに0を設定すると、エラーが生成されます。

同一のリソースに複数の制約を定義する場合、オブジェクト・タイプ(ジョブまたはプログラム)が一致している必要があります。たとえば、リソースの1つ以上の制約がジョブに基づいている場合、同じリソースにプログラムに対する新しい制約を追加すると、エラーが生成されます。

関連項目:

『*Oracle Database PL/SQL*パッケージ・プロシージャおよびタイプ・リファレンス』のSET_RESOURCE_CONSTRAINTプロシージャに関する項

親トピック: ジョブ・リソースの管理

29.9 ジョブの優先度付け

3つのスケジューラ・ジョブ(ジョブ・クラス、ウィンドウおよびウィンドウ・グループ)を使用して、Oracle Schedulerジョブに優先度を付けます。これらのオブジェクトでは、ジョブをデータベース・リソース・マネージャのコンシューマ・グループに関連付けることによって、ジョブに優先度を付けます。これにより、これらのジョブに割り当てられるリソースの量が制御されます。また、ジョブ・クラスでは、グループ内のすべてのジョブに同一のリソース・レベルが割り当てられている場合に、ジョブのグループ間に相対的な優先度を設定できます。

- ジョブ・クラスのジョブ優先度の管理
  ジョブ・クラスでは、ジョブをグループ化して優先度を付けることができます。また、メンバー・ジョブに一連の属性値を簡単に割り当てることができます。ジョブ・クラスは、データベース・リソース・マネージャに関連するジョブ・クラス属性を通して各メンバー・ジョブの優先度に影響します。
- ジョブ・クラス内でのジョブの相対的な優先度の設定
  DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャを使用して、同じジョブ・クラス内のジョブの相対的な優先度を変更できます。ジョブの優先度は、1を最も高い優先度として1から5の範囲で設定する必要があります。
- ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理
  日または週などの様々な期間中に、ジョブを自動的に開始したり、複数のジョブ間のリソース割当てを変更するには、ウィンドウを作成します。ウィンドウは、時間間隔で表現されます。
- ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理
  ウィンドウ・グループを使用すると、1日あるいは1週間などの間に複数期間実行するジョブを簡単にスケジュールできま

す。ウィンドウ・グループを作成してウィンドウをグループに加え、ウィンドウ・グループ名をジョブのschedule_name属性に設定すると、ジョブがウィンドウ・グループ内のすべてのウィンドウの指定期間に実行されます。ウィンドウ・グループはSYSスキーマにあります。

- リソース・マネージャを使用したジョブ間のリソース割当て
  データベース・リソース・マネージャ(リソース・マネージャ)は、データベース・セッション間のリソースの割当て方法を制御します。スケジューラ・ジョブのような非同期セッションだけでなく、ユーザー・セッションのような同期セッションも制御します。
- ジョブに対するリソース割当ての例
  ジョブのリソースの割当て方法の例を示します。

関連項目:

Oracle Database Resource Managerを使用したリソースの管理

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.9.1 ジョブ・クラスのジョブ優先度の管理

ジョブ・クラスでは、ジョブをグループ化して優先度を付けることができます。また、メンバー・ジョブに一連の属性値を簡単に割り当てることができます。ジョブ・クラスは、データベース・リソース・マネージャに関連するジョブ・クラス属性を通して各メンバー・ジョブの優先度に影響します。

データベースとともにデフォルトのジョブ・クラスが作成されます。ジョブ・クラスを指定せずにジョブを作成すると、ジョブはこのデフォルトのジョブ・クラス(DEFAULT_JOB_CLASS)に割り当てられます。デフォルトのジョブ・クラスでは、EXECUTE権限がPUBLICに付与されているため、ジョブの作成権限を持つすべてのデータベース・ユーザーは、デフォルトのジョブ・クラスにジョブを作成できます。

- ジョブ・クラスのタスクとそのプロシージャ
  DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なジョブ・クラスのタスクを管理します。
- ジョブ・クラスの作成
  ジョブ・クラスを作成するには、DBMS_SCHEDULERパッケージのCREATE_JOB_CLASSプロシージャまたはCloud Controlを使用します。ジョブ・クラスは、常にSYSスキーマに作成されます。
- ジョブ・クラスの変更
  ジョブ・クラスを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャまたはCloud Controlを使用します。
- ジョブ・クラスの削除
  1つ以上のジョブ・クラスを削除するには、DBMS_SCHEDULERパッケージのDROP_JOB_CLASSプロシージャまたはCloud Controlを使用します。

関連項目:

- ジョブ・クラスを表示するには、『Oracle Databaseリファレンス』
- 「リソース・マネージャを使用したジョブ間のリソース割当て」
- ジョブ・クラスの概要については、「ジョブ・クラス」

親トピック: ジョブの優先度付け

29.9.1.1 ジョブ・クラスのタスクとそのプロシージャ

DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なジョブ・クラスのタスクを管理します。

表29-8に、ジョブ・クラスの一般的なタスクとそれに対応するプロシージャおよび権限を示します。

表29-8 ジョブ・クラスのタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
ジョブ・クラスの作成	CREATE_JOB_CLASS	MANAGE SCHEDULER
ジョブ・クラスの変更	SET_ATTRIBUTE	MANAGE SCHEDULER
ジョブ・クラスの削除	DROP_JOB_CLASS	MANAGE SCHEDULER

権限の詳細は、「スケジューラ権限」を参照してください。

親トピック: ジョブ・クラスのジョブ優先度の管理

29.9.1.2 ジョブ・クラスの作成

ジョブ・クラスを作成するには、DBMS_SCHEDULERパッケージのCREATE_JOB_CLASSプロシージャまたはCloud Controlを使用します。ジョブ・クラスは、常にSYSスキーマに作成されます。

次の文では、すべての財務ジョブ用のジョブ・クラスが作成されます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB_CLASS (
   job_class_name             =>  'finance_jobs',
   resource_consumer_group    =>  'finance_group');
END;
/
```

このジョブ・クラス内のすべてのジョブは、finance_groupリソース・コンシューマ・グループに割り当てられます。

ジョブ・クラスを問い合せるには、*_SCHEDULER_JOB_CLASSESビューを使用します。

関連項目:

「リソース・コンシューマ・グループについて」

親トピック: ジョブ・クラスのジョブ優先度の管理

29.9.1.3 ジョブ・クラスの変更

ジョブ・クラスを変更するには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャまたはCloud Controlを使用します。

また、ジョブ・クラス名以外のすべての属性を変更できます。ジョブ・クラスの属性は、*_SCHEDULER_JOB_CLASSESビューで使用可能です。

ジョブ・クラスを変更した場合、そのクラスに属する実行中のジョブには影響しません。この変更は、今後実行されるジョブに対してのみ有効です。

親トピック: ジョブ・クラスのジョブ優先度の管理

29.9.1.4 ジョブ・クラスの削除

1つ以上のジョブ・クラスを削除するには、DBMS_SCHEDULERパッケージのDROP_JOB_CLASSプロシージャまたはCloud Controlを使用します。

ジョブ・クラスの削除とは、そのジョブ・クラスのすべてのメタデータをデータベースから削除することです。

DROP_JOB_CLASSプロシージャ・コールにジョブ・クラス名のカンマ区切りのリストを指定することで、1回のコールで複数のジョブ・クラスを削除できます。たとえば、次の文では3つのジョブ・クラスが削除されます。

```
BEGIN
  DBMS_SCHEDULER.DROP_JOB_CLASS('jobclass1, jobclass2, jobclass3');
END;
/
```

親トピック: ジョブ・クラスのジョブ優先度の管理

29.9.2 ジョブ・クラス内でのジョブの相対的な優先度の設定

DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャを使用して、同じジョブ・クラス内のジョブの相対的な優先度を変更できます。ジョブの優先度は、1を最も高い優先度として1から5の範囲で設定する必要があります。

たとえば、次の文では、my_job1のジョブ優先度の設定が1に変更されます。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
   name           =>   'my_emp_job1',
   attribute      =>   'job_priority',
   value          =>   1);
END;
/
```

属性が変更されたことを確認するには、次の文を発行します。

```
SELECT JOB_NAME, JOB_PRIORITY FROM DBA_SCHEDULER_JOBS;
JOB_NAME                       JOB_PRIORITY
------------------------------ ------------
MY_EMP_JOB                                3
MY_EMP_JOB1                               1
MY_NEW_JOB1                               3
MY_NEW_JOB2                               3
MY_NEW_JOB3                               3
```

システムにおけるジョブの全体的な優先度は、最初に、そのジョブのジョブ・クラスが割り当てられているリソース・コンシューマ・グループと現行のリソース・プランの組合せによって決定され、次に、ジョブ・クラス内の相対的な優先度によって決定されます。

関連項目:

- 「リソース・マネージャを使用したジョブ間のリソース割当て」
- SET_ATTRIBUTEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ジョブの優先度付け

29.9.3 ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

日または週などの様々な期間中に、ジョブを自動的に開始したり、複数のジョブ間のリソース割当てを変更するには、ウィンドウを作成します。ウィンドウは、時間間隔で表現されます。

- ウィンドウでのジョブ・スケジューリングとジョブ優先度について
  ウィンドウを使用すると、異なる時間帯で別々のリソース・プランを自動的にアクティブにできます。ジョブを実行すると、そのジョブに割り当てられているリソースが、リソース・プランの変更時に変わることがわかります。
- ウィンドウのタスクとそのプロシージャ
  DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なウィンドウのタスクを管理します。
- ウィンドウの作成
  ウィンドウを作成するには、Cloud ControlまたはDBMS_SCHEDULER.CREATE_WINDOWプロシージャを使用できます。
- ウィンドウの変更
  ウィンドウを変更するには、その属性を変更します。そのためには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEおよびSET_ATTRIBUTE_NULLプロシージャまたはCloud Controlを使用します。
- ウィンドウのオープン
  ウィンドウがオープンすると、スケジューラは、ウィンドウの作成時に関連付けられたリソース・プランに切り替えます。ウィンドウのオープン時に実行中のジョブがある場合、そのジョブに割り当てられているリソースはリソース・プランの切替えによって変更される場合があります。
- ウィンドウのクローズ
  ウィンドウはスケジュールに基づいてクローズするか、手動でクローズできます。
- ウィンドウの削除
  1つ以上のウィンドウを削除するには、DBMS_SCHEDULERパッケージのDROP_WINDOWプロシージャまたはCloud Controlを使用します。
- ウィンドウの無効化
  1つ以上のウィンドウを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャまたはCloud Controlを使用します。
- ウィンドウの有効化
  1つ以上のウィンドウを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャまたはCloud Controlを使用します。

親トピック: ジョブの優先度付け

29.9.3.1 ウィンドウでのジョブ・スケジューリングとジョブ優先度について

ウィンドウを使用すると、異なる時間帯で別々のリソース・プランを自動的にアクティブにできます。ジョブを実行すると、そのジョブに割り当てられているリソースが、リソース・プランの変更時に変わることがわかります。

ジョブは、そのschedule_name属性でウィンドウを指定できます。スケジューラは、このウィンドウがオープンしているジョブを開始します。ウィンドウには、ワークロード・サイクル中の様々な時間にオープンできるように、スケジュールが関連付けられます。

ウィンドウの主な属性は次のとおりです。

- スケジュール

  ウィンドウが有効である時期を制御します。

- 期間

ウィンドウがオープンしている長さを制御します。

- リソース・プラン

  ウィンドウのオープン時にアクティブ化されるリソース・プランの名前を設定します。

特定の時間に有効にできるウィンドウは1つのみです。ウィンドウはSYSスキーマに属します。

すべてのウィンドウ・アクティビティは、*_SCHEDULER_WINDOW_LOGビューに記録されます(ウィンドウ・ログとも呼ばれる)。ウィンドウ・ログの例は、「ウィンドウ・ログ」を参照してください。

関連項目:

ウィンドウの概要については、「ウィンドウ」を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.2 ウィンドウのタスクとそのプロシージャ

DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なウィンドウ・タスクを管理します。

表29-9に、ウィンドウの一般的なタスクとその処理に使用するプロシージャを示します。

表29-9 ウィンドウのタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
ウィンドウの作成	CREATE_WINDOW	MANAGE SCHEDULER
ウィンドウのオープン	OPEN_WINDOW	MANAGE SCHEDULER
ウィンドウのクローズ	CLOSE_WINDOW	MANAGE SCHEDULER
ウィンドウの変更	SET_ATTRIBUTE	MANAGE SCHEDULER
ウィンドウの削除	DROP_WINDOW	MANAGE SCHEDULER
ウィンドウの使用禁止	DISABLE	MANAGE SCHEDULER
ウィンドウの使用可能化	ENABLE	MANAGE SCHEDULER

権限の詳細は、「スケジューラ権限」を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.3 ウィンドウの作成

ウィンドウを作成するには、Cloud ControlまたはDBMS_SCHEDULER.CREATE_WINDOWプロシージャを使用します。

パッケージ・プロシージャを使用すると、resource_planパラメータをNULLのままにできます。この場合は、ウィンドウのオープン時に現行のプランが有効のままになります。

ウィンドウを作成するには、MANAGE SCHEDULER権限が必要です。

ウィンドウにスケジュールを指定する場合、スケジューラではそのスケジュールにウィンドウがすでに定義されているかどうかはチェックされません。したがって、複数のウィンドウが重複する場合があります。また、繰返し間隔としてPL/SQL式が設定されている名前付きスケジュールの使用は、ウィンドウに対してはサポートされていません。

ウィンドウ属性の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のCREATE_WINDOWプロシージャに関する項を参照してください。

次の例では、営業時間中にmixed_workload_planリソース・プランを有効にするdaytimeというウィンドウを作成しています。

```
BEGIN
   DBMS_SCHEDULER.CREATE_WINDOW (
     window_name      => 'daytime',
     resource_plan    => 'mixed_workload_plan',
     start_date       => '28-APR-09 08.00.00 AM',
     repeat_interval  => 'freq=daily; byday=mon,tue,wed,thu,fri',
     duration         => interval '9' hour,
     window_priority  => 'low',
     comments         => 'OLTP transactions have priority');
END;
/
```

ウィンドウが正しく作成されたことを検証するには、DBA_SCHEDULER_WINDOWSビューを問い合せます。たとえば、次の文を発行します。

```
SELECT WINDOW_NAME, RESOURCE_PLAN, DURATION, REPEAT_INTERVAL FROM
DBA_SCHEDULER_WINDOWS;
WINDOW_NAME    RESOURCE_PLAN          DURATION       REPEAT_INTERVAL
-----------    -------------------    -------------  ---------------
DAYTIME        MIXED_WORKLOAD_PLAN    +000 09:00:00  freq=daily;
byday=mon,tue,wed,thu,fri
```

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.4 ウィンドウの変更

ウィンドウを変更するには、その属性を変更します。そのためには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEおよびSET_ATTRIBUTE_NULLプロシージャまたはCloud Controlを使用します。

変更するときは、WINDOW_NAME以外のすべてのウィンドウ属性を変更できます。ウィンドウ属性の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のCREATE_WINDOWプロシージャに関する項を参照してください。

ウィンドウを変更した場合、アクティブ・ウィンドウには影響しません。この変更は、次回ウィンドウをオープンするときからのみ有効になります。

すべてのウィンドウを変更できます。使用禁止のウィンドウを変更した場合は、変更した後も使用禁止のままです。使用可能なウィンドウは自動的に使用禁止になり、変更後、使用可能化プロセスで実行される妥当性チェックが成功した場合は再び使用可能になります。

SET_ATTRIBUTEおよびSET_ATTRIBUTE_NULLプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.5 ウィンドウのオープン

ウィンドウがオープンすると、スケジューラは、ウィンドウの作成時に関連付けられたリソース・プランに切り替えます。ウィンドウのオープン時に実行中のジョブがある場合、そのジョブに割り当てられているリソースはリソース・プランの切替えによって変更される場合があります。

ウィンドウをオープンする方法は、次の2通りあります。

- ウィンドウのスケジュールに基づいたオープン
- OPEN_WINDOWプロシージャを使用した手動オープン

  このプロシージャは、スケジュールとは関係なくウィンドウをオープンします。ウィンドウがオープンし、関連付けられたリソース・プランが即時に有効になります。手動でオープンできるのは、使用可能なウィンドウのみです。

  OPEN_WINDOWプロシージャで、duration属性を使用して、ウィンドウがオープンしている時間の長さを指定できます。継続時間の型はINTERVAL DAY TO SECONDです。継続時間の指定がない場合、ウィンドウはそのウィンドウに保存されている通常の継続時間の間オープン状態です。

  手動によるウィンドウのオープンは、通常のスケジュール・ウィンドウの実行には影響しません。

  手動でオープンしたウィンドウをクローズする場合で、クローズするときに他のウィンドウもある場合、どのウィンドウをオープンするかを決定するために、オーバーラップ・ウィンドウのルールが適用されます。

  すでにオープンしているウィンドウがある場合でも、OPEN_WINDOWコールまたはCloud Controlでforceオプションを TRUEに設定すると、ウィンドウを強制的にオープンできます。

  forceオプションがTRUEに設定されているときは、その時点でオープンしているウィンドウの優先度の方が高い場合でも、スケジューラはウィンドウを自動的にクローズします。この手動でオープンしたウィンドウの継続時間中は、Schedulerにより他のスケジュール・ウィンドウはオープンされません(より優先順位の高いウィンドウもオープンされません)。すでにオープンしているウィンドウをオープンできます。この場合、ウィンドウは、OPEN_WINDOWコマンドが発行された時点から、コールで指定された継続時間の間オープンしたままになります。

  これを示す例を考えます。window1は、4時間の期間という設定で作成されました。もう2時間オープンしています。この時点でOPEN_WINDOW呼出しを使用してwindow1を再びオープンし、期間を指定しないと、window1には4時間という期間が設定されるため、もう4時間オープンすることになります。30分という期間を指定すると、ウィンドウは30分でクローズします。

ウィンドウがオープンすると、ウィンドウ・ログにエントリが作成されます。

現在のリソース・プランが、ALTER SYSTEM文FORCEオプションあるいはDBMS_RESOURCE_MANAGER.SWITCH_PLAN パッケージ・プロシージャのallow_scheduler_plan_switches引数FALSEを使用して、手動で切り替えられたものである場合、ウィンドウでリソース・プランの切替えに失敗する可能性があります。この場合、リソース・プランの切替え失敗がウィンドウ・ログに書き込まれます。

関連項目:

- DBMS_SCHEDULER.OPEN_WINDOWプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- DBMS_RESOURCE_MANAGER.SWITCH_PLANプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.6 ウィンドウのクローズ

ウィンドウはスケジュールに基づいてクローズするか、手動でクローズできます。

ウィンドウをクローズする方法は、次の2通りあります。

- スケジュールに基づいたクローズ

  ウィンドウは作成時に定義されたスケジュールに基づいてクローズします。

- CLOSE_WINDOWプロシージャを使用した手動クローズ

  CLOSE_WINDOWプロシージャは、オープン中のウィンドウを期間終了前にクローズします。

ウィンドウはクローズすると無効になります。ウィンドウがクローズすると、スケジューラは、リソース・プランをそのウィンドウ期間外で有効だったリソース・プランに切り替えるか、またはウィンドウの重複がある場合は別のウィンドウに切り替えます。存在しないウィンドウ、またはオープンしていないウィンドウをクローズしようとすると、エラーが発生します。

実行中のジョブは、そのジョブが実行されているウィンドウがクローズした場合でも、stop_on_window_close属性がジョブの作成時にTRUEに設定されていないかぎり停止しません。ただし、リソース・プランが変更される場合があるため、ジョブに割り当てられているリソースは変更される可能性があります。

実行中のジョブにスケジュールとしてウィンドウ・グループが設定されていると、そのジョブは、同じウィンドウ・グループのメンバーである別のウィンドウが元のウィンドウのクローズ時にアクティブになった場合、停止しません。これは、stop_on_window_close属性がTRUEに設定された状態でジョブが作成されている場合でも同じです。

ウィンドウがクローズ状態になると、ウィンドウ・ログDBA_SCHEDULER_WINDOW_LOGにエントリが追加されます。

CLOSE_WINDOWプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.7 ウィンドウの削除

1つ以上のウィンドウを削除するには、DBMS_SCHEDULERパッケージのDROP_WINDOWプロシージャまたはCloud Controlを使用します。

ウィンドウが削除されると、そのウィンドウに関するすべてのメタデータが*_SCHEDULER_WINDOWSビューから削除されます。また、ウィンドウに対するすべての参照がウィンドウ・グループから削除されます。

DROP_WINDOWプロシージャにウィンドウ名またはウィンドウ・グループ名のカンマ区切りのリストを指定すると、1回のコールで複数のウィンドウを削除できます。たとえば、次の文ではウィンドウとウィンドウ・グループの両方が削除されます。

```
BEGIN
  DBMS_SCHEDULER.DROP_WINDOW ('window1, window2, window3,
   windowgroup1, windowgroup2');
END;
/
```

ウィンドウ・グループ名を指定した場合、ウィンドウ・グループ内のウィンドウは削除されますが、ウィンドウ・グループは削除されません。ウィンドウ・グループを削除するには、DROP_GROUPプロシージャを使用する必要があります。

DROP_GROUPプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.8 ウィンドウの無効化

1つ以上のウィンドウを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャまたはCloud Controlを使用します。

そのため、ウィンドウはオープンしません。ただし、ウィンドウのメタデータは依然存在するため、再び使用可能に設定できます。DISABLEプロシージャは複数のスケジューラ・オブジェクトに対して使用されるため、ウィンドウを使用禁止にするときは、先頭にSYSを付ける必要があります。

ウィンドウは他の理由で使用禁止になる場合もあります。たとえば、ウィンドウはそのスケジュールの終了時に使用禁止になります。また、存在しないスケジュールをウィンドウが指し示している場合も使用禁止になります。

スケジュールとしてウィンドウが設定されているジョブがある場合、そのウィンドウは、プロシージャ・コールでforceをTRUEに設定しないかぎり、使用禁止にできません。デフォルトでは、forceはFALSEに設定されます。ウィンドウが使用禁止の場合、このウィンドウがスケジュールとして設定されているジョブを使用禁止にすることはできません。

DISABLEプロシージャ・コールにウィンドウ名またはウィンドウ・グループ名のカンマ区切りのリストを指定することで、1回のコールで複数のウィンドウを使用禁止にすることもできます。たとえば、次の文ではウィンドウとウィンドウ・グループの両方が使用禁止になります。

```
BEGIN
  DBMS_SCHEDULER.DISABLE ('sys.window1, sys.window2,
   sys.window3, sys.windowgroup1, sys.windowgroup2');
END;
/
```

DISABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.3.9 ウィンドウの有効化

1つ以上のウィンドウを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャまたはCloud Controlを使用します。

使用可能なウィンドウとは、オープン可能なウィンドウのことです。ウィンドウは、デフォルトでenabledで作成されます。ENABLEプロシージャを使用してウィンドウを使用可能にすると、妥当性チェックが実行され、このチェックが成功した場合のみウィンドウは使用可能になります。ウィンドウが使用可能になると、ウィンドウ・ログ表にログが記録されます。ENABLEプロシージャは複数のスケジューラ・オブジェクトに対して使用されるため、ウィンドウを使用可能にするときは、先頭にSYSを付ける必要があります。

ウィンドウ名のカンマ区切りのリストを指定することで、1回のコールで複数のウィンドウを使用可能にできます。たとえば、次の文では3つのウィンドウが使用可能になります。

```
BEGIN
  DBMS_SCHEDULER.ENABLE ('sys.window1, sys.window2, sys.window3');
END;
/
```

ENABLEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.4 ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

ウィンドウ・グループを使用すると、1日あるいは1週間などの間に複数期間実行するジョブを簡単にスケジュールできます。ウィンドウ・グループを作成してウィンドウをグループに加え、ウィンドウ・グループ名をジョブのschedule_name属性に設定すると、

ジョブがウィンドウ・グループ内のすべてのウィンドウの指定期間に実行されます。ウィンドウ・グループはSYSスキーマにあります。

- ウィンドウ・グループのタスクとそのプロシージャ
  DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なグループ・タスクを管理します。
- ウィンドウ・グループの作成
  ウィンドウ・グループを作成するには、DBMS_SCHEDULER.CREATE_GROUPプロシージャを使用して、グループ・タイプとしてWINDOWを指定します。
- ウィンドウ・グループの削除
  1つ以上のウィンドウ・グループを削除するには、DBMS_SCHEDULERパッケージのDROP_GROUPプロシージャを使用します。
- ウィンドウ・グループへのメンバーの追加
  ウィンドウ・グループにウィンドウを追加するには、DBMS_SCHEDULERパッケージのADD_GROUP_MEMBERプロシージャを使用します。
- ウィンドウ・グループからのメンバーの削除
  ウィンドウ・グループから1つ以上のウィンドウを削除するには、DBMS_SCHEDULERパッケージのREMOVE_GROUP_MEMBERプロシージャを使用します。
- ウィンドウ・グループの有効化
  1つ以上のウィンドウ・グループを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャを使用します。
- ウィンドウ・グループの無効化
  ウィンドウ・グループを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャを使用します。

関連項目:

ウィンドウ・グループの概要については、「ウィンドウ・グループ」を参照してください。

親トピック: ジョブの優先度付け

29.9.4.1 ウィンドウ・グループのタスクとそのプロシージャ

DBMS_SCHEDULERパッケージのプロシージャを使用して、一般的なウィンドウ・グループのタスクを管理します。

表29-10に、ウィンドウ・グループの一般的なタスクとその処理に使用するプロシージャを示します。

表29-10 ウィンドウ・グループのタスクとそのプロシージャ

タスク	プロシージャ	必要な権限
ウィンドウ・グループの作成	CREATE_GROUP	MANAGE SCHEDULER
ウィンドウ・グループの削除	DROP_GROUP	MANAGE SCHEDULER
ウィンドウ・グループへのメンバーの追加	ADD_GROUP_MEMBER	MANAGE SCHEDULER
ウィンドウ・グループからのメンバーの削除	REMOVE_GROUP_MEMBER	MANAGE SCHEDULER

タスク	プロシージャ	必要な権限
ウィンドウ・グループの有効化	ENABLE	MANAGE SCHEDULER
ウィンドウ・グループの無効化	DISABLE	MANAGE SCHEDULER

権限の詳細は、「スケジューラ権限」を参照してください。

親トピック: ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.4.2 ウィンドウ・グループの作成

ウィンドウ・グループを作成するには、DBMS_SCHEDULER.CREATE_GROUPプロシージャを使用して、グループ・タイプとしてWINDOWを指定します。

グループのメンバー・ウィンドウは、グループの作成時に指定するか、または後でADD_GROUP_MEMBERプロシージャを使用して追加できます。ウィンドウ・グループは、他のウィンドウ・グループのメンバーになれません。ただし、メンバーを設定しないウィンドウ・グループは作成できます。

ウィンドウ・グループを作成し、存在しないメンバー・ウィンドウを指定すると、エラーが生成され、ウィンドウ・グループは作成されません。ウィンドウがすでにウィンドウ・グループのメンバーである場合、再度追加されることはありません。

ウィンドウ・グループはSYSスキーマで作成します。ウィンドウ・グループは、ウィンドウと同様に、PUBLICへのアクセス権を伴って作成されるため、ウィンドウ・グループへのアクセス権限は必要ありません。

次の文では、downtimeというウィンドウ・グループが作成され、2つのウィンドウ(weeknightsとweekends)が追加されます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_GROUP (
   group_name    =>  'downtime',
   group_type    =>  'WINDOW',
   member        =>  'weeknights, weekends');
END;
/
```

ウィンドウ・グループの内容を確認するには、MANAGE SCHEDULER権限を持つユーザーとして次の問合せを発行します。

```
SELECT group_name, enabled, number_of_members FROM dba_scheduler_groups
  WHERE group_type = 'WINDOW';
GROUP_NAME     ENABLED  NUMBER_OF_MEMBERS
-------------- -------- -----------------
DOWNTIME       TRUE                    2
SELECT group_name, member_name FROM dba_scheduler_group_members;
GROUP_NAME      MEMBER_NAME
--------------- --------------------
DOWNTIME        "SYS"."WEEKENDS"
DOWNTIME        "SYS"."WEEKNIGHTS"
```

親トピック: ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.4.3 ウィンドウ・グループの削除

1つ以上のウィンドウ・グループを削除するには、DBMS_SCHEDULERパッケージのDROP_GROUPプロシージャを使用します。

このプロシージャをコールすると、ウィンドウ・グループは削除されますが、ウィンドウ・グループのメンバーであるウィンドウは削除されません。ウィンドウ・グループ自体は削除せずに、ウィンドウ・グループのメンバーであるウィンドウをすべて削除するには、DROP_WINDOWプロシージャを使用し、そのコールにウィンドウ・グループ名を指定します。

DROP_GROUPプロシージャ・コールにウィンドウ・グループ名のカンマ区切りのリストを指定することで、1回のコールで複数のウィンドウ・グループを削除できます。各ウィンドウ・グループ名の前にSYSスキーマの接頭辞を付ける必要があります。たとえば、次の文では3つのウィンドウ・グループが削除されます。

```
BEGIN
DBMS_SCHEDULER.DROP_GROUP('sys.windowgroup1, sys.windowgroup2, sys.windowgroup3');
END;
/
```

親トピック: ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.4.4 ウィンドウ・グループへのメンバーの追加

ウィンドウ・グループにウィンドウを追加するには、DBMS_SCHEDULERパッケージのADD_GROUP_MEMBERプロシージャを使用します。

ウィンドウのカンマ区切りのリストを指定すると、1回のコールで複数のメンバーをウィンドウ・グループに追加できます。たとえば、次の文ではウィンドウ・グループwindow_group1に2つのウィンドウが追加されます。

```
BEGIN
  DBMS_SCHEDULER.ADD_GROUP_MEMBER ('sys.windowgroup1','window2, window3');
END;
/
```

すでにオープンしているウィンドウがウィンドウ・グループに追加された場合、スケジューラは、ウィンドウ・グループ内の次のウィンドウがオープンするまで、このウィンドウ・グループを指し示すジョブを開始しません。

親トピック: ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.4.5 ウィンドウ・グループからのメンバーの削除

ウィンドウ・グループから1つ以上のウィンドウを削除するには、DBMS_SCHEDULERパッケージのREMOVE_GROUP_MEMBERプロシージャを使用します。

stop_on_window_closeフラグが設定されているジョブが停止するのは、ウィンドウがクローズするときのみです。オープン中のウィンドウをウィンドウ・グループから削除しても、この処理への影響はありません。

ウィンドウのカンマ区切りのリストを指定すると、1回のコールで複数のメンバーをウィンドウ・グループから削除できます。たとえば、次の文では2つのウィンドウが削除されます。

```
BEGIN
  DBMS_SCHEDULER.REMOVE_GROUP_MEMBER('sys.window_group1', 'window2, window3');
END;
/
```

親トピック: ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.4.6 ウィンドウ・グループの有効化

1つ以上のウィンドウ・グループを有効にするには、DBMS_SCHEDULERパッケージのENABLEプロシージャを使用します。

デフォルトでは、ウィンドウ・グループはENABLEDで作成されます。たとえば:

```
BEGIN
  DBMS_SCHEDULER.ENABLE('sys.windowgroup1, sys.windowgroup2, sys.windowgroup3');
END;
/
```

親トピック: ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.4.7 ウィンドウ・グループの無効化

ウィンドウ・グループを無効にするには、DBMS_SCHEDULERパッケージのDISABLEプロシージャを使用します。

ウィンドウ・グループがスケジュールとして使用禁止になっているジョブは、メンバー・ウィンドウがオープンしても実行されません。ウィンドウ・グループを使用禁止にしても、そのメンバー・ウィンドウは使用禁止になりません。

また、ウィンドウ・グループ名のカンマ区切りのリストを指定することで、1回のコールで複数のウィンドウ・グループを使用禁止にすることもできます。たとえば、次の文では3つのウィンドウ・グループが使用禁止になります。

```
BEGIN
  DBMS_SCHEDULER.DISABLE('sys.windowgroup1, sys.windowgroup2, sys.windowgroup3');
END;
/
```

親トピック: ウィンドウ・グループを使用したジョブ・スケジューリングとジョブ優先度の管理

29.9.5 リソース・マネージャを使用したジョブ間のリソース割当て

データベース・リソース・マネージャ(リソース・マネージャ)は、データベース・セッション間のリソースの割当て方法を制御します。スケジューラ・ジョブのような非同期セッションだけでなく、ユーザー・セッションのような同期セッションも制御します。

データベースのすべての「作業単位」をリソース・コンシューマ・グループにグループ化し、リソース・プランを使用して、様々なコンシューマ・グループ間のリソースの割当て方法を指定します。リソース・マネージャが割り当てる主なシステム・リソースはCPUです。

スケジューラ・ジョブの場合、最初に各ジョブをジョブ・クラスに割り当てて、次にジョブ・クラスをコンシューマ・グループに関連付けることによって、リソースが割り当てられます。その後、リソースは、コンシューマ・グループ内のスケジューラ・ジョブと他のセッションとの間で分配されます。また、ジョブ・クラス内のジョブに相対的な優先度を割り当てて、それに従ってそれらのジョブにリソースが分配されるようにすることもできます。

現行のリソース・プランはいつでも手動で変更できます。かわりに、スケジューラ・ウィンドウを作成することで現行のリソース・プランを変更することもできます。ウィンドウには、リソース・プラン属性があります。ウィンドウがオープンすると、現行のプランがウィンドウのリソース・プランに切り替えられます。

スケジューラは、完了に必要なリソースを確保できないまま同時に多数のジョブを実行するかわりに、少なくとも一部のジョブを完了できるように、同時に実行するジョブの数を制限しようとします。

スケジューラとリソース・マネージャは緊密に統合されています。ジョブ・コーディネータは、リソース・マネージャからデータベース・リソースの可用性を取得します。その情報に基づいて、コーディネータは、開始するジョブの数を決定します。実行するために十分なリソースがあるジョブ・クラスのジョブのみを起動します。コンシューマ・グループに割り当てられた最大リソースに到達したとリソース・マネージャが判断するまで、そのコンシューマ・グループにマップされた特定のジョブ・クラスのジョブをコーディネータが起動し続けます。そのため、実行準備が完了したジョブがジョブ表にあっても、ジョブを実行するリソースがないため、ジョブ・コーディネータに取り出されない場合があります。そのため、スケジュールされた正確な時刻にジョブが実行される保証はありません。コーディネータは、どのコンシューマ・グループにまだ使用可能なリソースがあるかに基づいて、ジョブ表からジョブを取り出します。

リソース・マネージャでは、実行中の各ジョブに割り当てられているリソースを指定のリソース・プランに基づいて継続的に管理します。リソース・マネージャが管理できるのはデータベースのプロセスのみであることに注意してください。リソースのアクティブな管理は、外部ジョブには適用されません。

関連項目:

Oracle Database Resource Managerを使用したリソースの管理

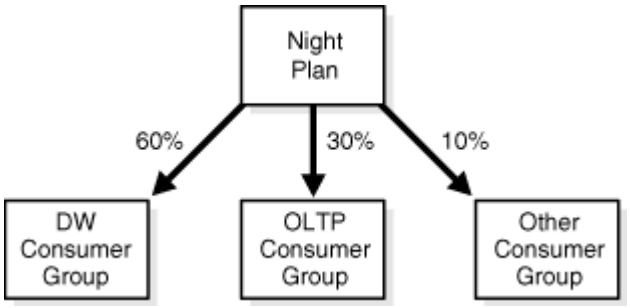
親トピック: ジョブの優先度付け

29.9.6 ジョブに対するリソース割当ての例

ジョブのリソースの割当て方法の例を示します。

アクティブなリソース・プランの名前が「Night Plan」で、コンシューマ・グループDWにマップするJC1、コンシューマ・グループOLTPにマップするJC2およびデフォルト・コンシューマ・グループにマップするJC3の3つのジョブ・クラスがあるとします。図29-2は、このシナリオを簡単な図で表したものです。

図29-2 リソース・プランの例



このリソース・プランでは、ジョブ・クラスJC1に属するジョブが明らかに優先されます。コンシューマ・グループDWはリソースを60%取得するため、ジョブ・クラスJC1に属するジョブはリソースを60%取得します。コンシューマ・グループOLTPはリソースを30%取得するため、ジョブ・クラスJC2内のジョブはリソースを30%取得します。コンシューマ・グループOtherでは、他のすべてのコンシューマ・グループがリソースを10%取得することが指定されます。そのため、ジョブ・クラスJC3に属するすべてのジョブが、10%のリソースを共有し、リソースを最大10%取得できます。

あるコンシューマ・グループで未使用のリソースは、他のコンシューマ・グループで使用できます。したがって、ジョブ・クラスJC1のジョブが割当ての60%を完全に使用していない場合、その未使用部分はクラスJC2とJC3のジョブが使用できます。リソース・マネージャは、CPU使用が100%に達するまで、リソース使用の制限を開始しません。詳細は、「Oracle Database Resource Managerを使用したリソースの管理」を参照してください。

親トピック: ジョブの優先度付け

29.10 ジョブの監視

いくつかの異なる方法でジョブを監視できます。

- ジョブの監視について
  スケジューラのジョブを監視するには、いくつかの方法があります。
- ジョブ・ログ
  ジョブ・ログでローカルおよびリモート・ジョブの結果を参照できます。
- 複数の宛先のジョブの監視
  複数の宛先のジョブの場合、親ジョブの全体的な状態は子ジョブの結果に依存します。
- スケジューラによって呼び出されるイベントによるジョブ状態の監視
  スケジューラでは、ジョブの状態が変化したとき、イベントを発生することができます。
- 電子メール通知によるジョブ状態の監視
  スケジューラは、ジョブの状態が変化したときに電子メールを送信できます。

親トピック: Oracle Schedulerを使用したジョブのスケジューリング

29.10.1 ジョブの監視について

スケジューラ・ジョブは、いくつかの方法で監視できます。

次の方法でスケジューラ・ジョブを監視できます:

- ジョブ・ログの表示

  ジョブ・ログには、*_SCHEDULER_JOB_LOGおよび*_SCHEDULER_JOB_RUN_DETAILSデータ・ディクショナリ・ビューが含まれます。

  * = {DBA|ALL|USER}

  「ジョブ・ログの表示」を参照してください。

- 追加のデータ・ディクショナリ・ビューの問合せ

  DBA_SCHEDULER_RUNNING_JOBSやDBA_SCHEDULER_RUNNING_CHAINSなどのビューを問い合せて、実行中のジョブとチェーンのステータスおよび詳細を表示します。

- スケジューラからジョブ状態イベントを受け取るアプリケーションの作成

  「スケジューラによって呼び出されるイベントによるジョブ状態の監視」を参照

- 状態変化に応じて電子メール通知を送信するジョブの構成

  「電子メール通知によるジョブ状態の監視」を参照

親トピック: ジョブの監視

29.10.2 ジョブ・ログ

ジョブ・ログでローカルおよびリモート・ジョブの結果を参照できます。

- ジョブ・ログの表示
  ジョブの実行、状態変化および失敗について、ジョブ・ログ内の情報を表示できます。ジョブ・ログには、ローカル・ジョブとリモート・ジョブの両方の結果が示されます。
- 実行詳細
  RUN、RETRY_RUNまたはRECOVERY_RUN操作に関する*_SCHEDULER_JOB_LOG内の行ごとに、*_SCHEDULER_JOB_RUN_DETAILSビューには対応する行があります。
- ジョブおよびジョブ・クラスのロギング・レベルの優先度
  ジョブおよびジョブ・クラスには、logging_level属性があります。

親トピック: ジョブの監視

29.10.2.1 ジョブ・ログの表示

ジョブの実行、状態変化および失敗について、ジョブ・ログ内の情報を表示できます。ジョブ・ログには、ローカル・ジョブとリモート・ジョブの両方の結果が示されます。

ジョブ・ログは、次の2つのデータ・ディクショナリ・ビューとして実装されます。

- *_SCHEDULER_JOB_LOG
- *_SCHEDULER_JOB_RUN_DETAILS

スケジューラは、有効なロギング・レベルに応じて、ジョブの実行時、作成時、削除時、有効化時などにジョブ・ログ・エントリを作成できます。繰返しスケジュールが設定されたジョブの場合、スケジューラではジョブ・ログ(ジョブ・インスタンスごとに1つ)に複数の

エントリが作成されます。各ログ・エントリは、ジョブの完了ステータスなど、特定の実行に関する情報を提供します。

次の例では、max_runs属性の値が4に設定されている繰返しジョブのジョブ・ログ・エントリを示しています。

```
SELECT job_name, job_class, operation, status FROM USER_SCHEDULER_JOB_LOG;
JOB_NAME          JOB_CLASS            OPERATION       STATUS
----------------- -------------------- --------------- ----------
JOB1              CLASS1               RUN             SUCCEEDED
JOB1              CLASS1               RUN             SUCCEEDED
JOB1              CLASS1               RUN             SUCCEEDED
JOB1              CLASS1               RUN             SUCCEEDED
JOB1              CLASS1               COMPLETED
```

ジョブまたはジョブ・クラスのlogging_level属性を設定することによって、ジョブ・ログに情報を書き込む頻度を制御できます。表29-11に、logging_levelに指定可能な値を示します。

表29-11 ジョブのロギング・レベル

ロギング・レベル	説明
DBMS_SCHEDULER.LOGGING_OFF	ロギングは実行されません。
DBMS_SCHEDULER.LOGGING_FAILED_RUNS	ジョブが失敗した場合にのみログ・エントリが作成されます。
DBMS_SCHEDULER.LOGGING_RUNS	ジョブが実行されるたびにログ・エントリが作成されます。
DBMS_SCHEDULER.LOGGING_FULL	ジョブが実行されるたびに、および作成、有効化、無効化、更新(SET_ATTRIBUTE を使用)、停止および削除など、ジョブに対して実行された操作ごとに、ログ・エントリが作成されます。

ジョブの実行に関するログ・エントリは、ジョブの実行が正常に完了するか、失敗するかまたは停止するまで作成されません。

次の例は、完全なジョブ・ライフサイクルのジョブ・ログ・エントリを示しています。この場合、ジョブ・クラスのロギング・レベルはLOGGING_FULLで、ジョブは非繰返しジョブです。最初に正常に実行された後、もう一度実行されるようにジョブが有効化されています。次に、停止され、削除されています。

```
SELECT to_char(log_date, 'DD-MON-YY HH24:MI:SS') TIMESTAMP, job_name,
  job_class, operation, status FROM USER_SCHEDULER_JOB_LOG
  WHERE job_name = 'JOB2' ORDER BY log_date;
TIMESTAMP            JOB_NAME  JOB_CLASS  OPERATION  STATUS
-------------------- --------- ---------- ---------- ---------
18-DEC-07 23:10:56   JOB2      CLASS1     CREATE
18-DEC-07 23:12:01   JOB2      CLASS1     UPDATE
18-DEC-07 23:12:31   JOB2      CLASS1     ENABLE
18-DEC-07 23:12:41   JOB2      CLASS1     RUN        SUCCEEDED
18-DEC-07 23:13:12   JOB2      CLASS1     ENABLE
18-DEC-07 23:13:18   JOB2                 RUN        STOPPED
18-DEC-07 23:19:36   JOB2      CLASS1     DROP
```

親トピック: ジョブ・ログ

29.10.2.2 実行詳細

RUN、RETRY_RUNまたはRECOVERY_RUN操作に関する*_SCHEDULER_JOB_LOG内の行ごとに、

*_SCHEDULER_JOB_RUN_DETAILSビューには対応する行があります。

2つの異なるビューの行は、それぞれのLOG_ID列で関係付けられています。実行詳細ビューを参照して、ジョブが失敗した理由や停止された理由を判断できます。

```
SELECT to_char(log_date, 'DD-MON-YY HH24:MI:SS') TIMESTAMP, job_name, status,
   SUBSTR(additional_info, 1, 40) ADDITIONAL_INFO
   FROM user_scheduler_job_run_details ORDER BY log_date;
TIMESTAMP            JOB_NAME   STATUS    ADDITIONAL_INFO
-------------------- ---------- --------- ----------------------------------------
18-DEC-07 23:12:41   JOB2       SUCCEEDED
18-DEC-07 23:12:18   JOB2       STOPPED   REASON="Stop job called by user:'SYSTEM'
19-DEC-07 14:12:20   REMOTE_16  FAILED    ORA-29273: HTTP request failed ORA-06512
```

実行詳細ビューには、実際のジョブ開始時刻と継続時間も含まれています。

属性STORE_OUTPUTを使用して、外部ジョブの場合はstdoutまたはデータベース・ジョブの場合はDBMS_OUTPUTに送信された出力を*_SCHEDULER_JOB_RUN_DETAILSビューで保存するように指定することもできます。STORE_OUTPUTがTRUEに設定され、LOGGING_LEVELがジョブの実行をログに記録する必要があることを示している場合は、すべての出力が収集され、このビューのBINARY_OUTPUT出力列に挿入されます。char表現はOUTPUT列から問い合せることができます。

親トピック: ジョブ・ログ

29.10.2.3 ジョブおよびジョブ・クラスのロギング・レベルの優先度

ジョブおよびジョブ・クラスには、logging_level属性があります。

表29-11に、この属性に指定できる値を示します。ジョブ・クラスのデフォルト・ロギング・レベルはLOGGING_RUNSで、個々のジョブのデフォルト・レベルはLOGGING_OFFです。ジョブ・クラスのロギング・レベルが、クラス内のジョブのロギング・レベルよりも高い場合は、ジョブ・クラスのロギング・レベルが優先されます。そのため、デフォルトでは、すべてのジョブの実行がジョブ・ログに記録されます。

非常に短くて頻度の高いジョブの場合は、個々の実行をすべて記録すると負荷が非常に高くなるため、ロギングをオフに設定して、ジョブが失敗した場合にのみロギングが行われるように設定する場合もあります。ただし、特定のクラスのジョブで発生したすべての事象の完全なロギングを取得する必要がある場合には、そのクラスの完全なロギングを有効にします。

すべてのジョブのロギングが確実に作成されるようにするには、個々のジョブ作成者がロギングをオフにできないようにする必要があります。これをサポートするために、スケジューラではクラス別のレベルがジョブの情報を記録する最低レベルとなっています。ジョブ作成者は、個々のジョブに対するロギング・レベルを上げることはできますが、下げることはできません。したがって、個々のジョブのロギング・レベルをすべてLOGGING_OFFに設定すると、クラス内のすべてのジョブがクラスでの指定に従って記録されます。

この機能は、デバッグの目的で提供されています。たとえば、クラス別のレベルでジョブ実行を記録するように設定されているときに、ジョブ・レベルでロギングがオフにされた場合でも、スケジューラはジョブの実行を記録します。ただし、ジョブ作成者が完全ロギングをオンにしているときに、クラス別のレベルでは実行のみを記録するように設定されている場合は、ジョブの上位のロギング・レベルが優先され、この個別ジョブのすべての操作がログに記録されます。このように、エンド・ユーザーは、完全ロギングをオンにして自分のジョブをテストできます。

個々のジョブのロギング・レベルを設定するには、そのジョブについてSET_ATTRIBUTEプロシージャを使用する必要があります。たとえば、mytestjobというジョブの完全ロギングをオンにするには、次の文を発行します。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
   'mytestjob', 'logging_level', DBMS_SCHEDULER.LOGGING_FULL);
END;
/
```

ジョブ・クラスのロギング・レベルを設定できるのは、MANAGE SCHEDULER権限を付与されているユーザーのみです。

関連項目:

ジョブ・クラスのロギング・レベルの設定の詳細は、「ウィンドウ・ログおよびジョブ・ログの監視と管理」

親トピック: ジョブ・ログ

29.10.3 複数の宛先のジョブの監視

複数の宛先のジョブの場合、親ジョブの全体的な状態は子ジョブの結果に依存します。

たとえば、すべての子ジョブが成功した場合、親ジョブの状態はSUCCEEDEDに設定されます。すべてが失敗した場合、親ジョブの状態はFAILEDに設定されます。一部が失敗し、一部が成功した場合、親ジョブの状態はSOME FAILEDに設定されます。

一部の宛先で子ジョブの開始が遅延するような状況では、親ジョブの状態がファイナライズされるまでに大幅な遅延が生じる可能性があります。複数の宛先のジョブが繰返しジョブの場合は、スケジュールされている次回の実行に進むジョブもあれば、前回の実行がまだ終了しないジョブが生じることもあります。この場合、親ジョブの状態はINCOMPLETEに設定されます。ただし、最終的には、遅延しているジョブも他の兄弟ジョブに追いつき、親ジョブの最終的な状態を決定できるようになります。

表29-12に、複数の宛先のジョブを対象としたジョブ監視ビューの内容を示します。

表29-12 複数の宛先のジョブを対象としたスケジューラ・データ・ディクショナリ・ビューの内容

ビュー名	目次
*_SCHEDULER_JOBS	親ジョブの 1 つのエントリ
*_SCHEDULER_RUNNING_JOBS	親ジョブの 1 つのエントリ(親ジョブが開始された場合)および実行中の子ジョブごとに 1 つのエントリ
*_SCHEDULER_JOB_LOG	親ジョブの 1 つのエントリ(親ジョブが開始された場合)(operation = 'MULTIDEST_START')、子ジョブごとに 1 つのエントリ(子ジョブが完了した場合)および親ジョブの 1 つのエントリ(最後の子ジョブが完了し、親が完了した場合)(operation = 'MULTIDEST_RUN')
*_SCHEDULER_JOB_RUN_DETAILS	子ジョブごとに 1 つのエントリ(子ジョブが完了した場合)および親ジョブの 1 つのエントリ(最後の子ジョブが完了し、親が完了した場合)
*_SCHEDULER_JOB_DESTS	親ジョブの宛先ごとに 1 つのエントリ

*_SCHEDULER_JOB_DESTSビューでは、各子ジョブに割り当てられている一意のジョブ宛先ID(job_dest_id)を調べることができます。このIDは、ジョブ、資格証明および宛先の一意の組合せを表します。このIDは、STOP_JOBプロシージャで使用できます。また、*_SCHEDULER_JOB_DESTSビューで各子ジョブのジョブ状態を監視できます。

関連項目:

- 「複数の宛先のジョブ」
- 「複数の宛先のジョブの作成」
- 「スケジューラのデータ・ディクショナリ・ビュー」

親トピック: ジョブの監視

29.10.4 スケジューラによって呼び出されるイベントによるジョブ状態の監視

スケジューラでは、ジョブの状態が変化したとき、イベントを発生することができます。

- ジョブ状態イベントについて
  ジョブの状態が変化したときに、スケジューラがイベントを呼び出すようにジョブを構成できます。
- イベントを呼び出すようにジョブを変更する方法
  ジョブのジョブ状態イベントの発生を可能にするには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャを使用してraise_eventsジョブ属性のビット・フラグをオンにします。
- ジョブの状態イベントのアプリケーションでの使用
  ジョブ状態イベントを使用するには、アプリケーションがスケジューラ・イベント・キューSYS.SCHEDULER$_EVENT_QUEUEをサブスクライブする必要があります。このキューはセキュアなキューであり、所有者はSYSです。

親トピック: ジョブの監視

29.10.4.1 ジョブ状態イベントについて

ジョブの状態が変化したときに、スケジューラがイベントを呼び出すようにジョブを構成できます。

スケジューラは、ジョブの開始時、ジョブの完了時、ジョブがその割当ての実行時間を超えたときなどにイベントを呼び出します。イベントのコンシューマは、そのイベントに対応して処理を実行するアプリケーションです。たとえば、システムの負荷が高いために、スケジュールされた開始時間から30分を経過してもジョブが開始されない場合、スケジューラは、ハンドラのアプリケーションによって低優先度のジョブを停止させてシステム・リソースを解放するイベントを呼び出すことができます。スケジューラでは、ローカル(標準)・ジョブ、リモート・データベース・ジョブ、ローカル外部ジョブおよびリモート外部ジョブに対してジョブ状態イベントを呼び出すことができます。

表29-13に、スケジューラによって呼び出されるジョブ状態イベントのタイプを示します。

表29-13 スケジューラによって呼び出されるジョブ状態イベントのタイプ

イベント・タイプ	説明
job_all_events	イベントではなく、すべてのイベントを簡単に有効にするための定数です。
job_broken	ジョブ属性 max_failures で定義されている失敗回数を超えたため、ジョブは使用禁止になり、状態が BROKEN に変更されました。
job_chain_stalled	チェーンを実行するジョブが CHAIN_STALLED 状態になりました。

イベント・タイプ	説明
	実行中または実行がスケジュールされているステップがない場合にチェーンの evaluation_interval を NULL に設定すると、実行中のチェーンは停止状態になります。手動で操作を行わないかぎり、チェーンは続行されません。
job_completed	max_runs または end_date に達したため、ジョブが終了しました。
job_disabled	Scheduler または SET_ATTRIBUTE へのコールによってジョブが使用禁止になりました。
job_failed	エラーが発生したか、異常終了したため、ジョブが失敗しました。
job_over_max_dur	ジョブは、max_run_duration 属性で指定した最大実行時間を超えました。
job_run_completed	ジョブの実行が、失敗、成功または停止しました。
job_sch_lim_reached	ジョブのスケジュール制限に達しました。ジョブ開始の遅延時間がジョブ属性 schedule_limit の値を超えたため、ジョブは開始されませんでした。
job_started	ジョブが開始されました
job_stopped	STOP_JOB へのコールによって、ジョブが停止されました。
job_succeeded	ジョブが正常に完了しました。

ジョブ状態イベントの呼出しを有効にするには、raise_eventsジョブ属性を設定します。デフォルトでは、ジョブはジョブ状態イベントを呼び出しません。

スケジューラは、イベントを呼び出すためにOracle Databaseアドバンスト・キューイングを使用します。ジョブの状態の変更に関するイベントが発生すると、スケジューラは、メッセージをデフォルトのイベント・キューにエンキューします。アプリケーションは、このキューをサブスクライブし、イベント・メッセージをデキューして、適切な処理を行います。

ジョブに対するジョブの状態変化イベントを有効(使用可能)にすると、スケジューラは、メッセージをスケジューラ・イベント・キューSYS.SCHEDULER$_EVENT_QUEUEにエンキューすることによってこれらのイベントを呼び出します。このキューはセキュアなキューであるため、アプリケーションによっては、特定のユーザーがキューの操作を実行できるようにキューを構成する必要があります。

スケジューラのイベント・キューが無制限に大きくなるのを防ぐために、デフォルトでは、スケジューラが呼び出したイベントは24時間で失効します。DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTEプロシージャを使用して、

event_expiry_timeスケジューラ属性を設定することで、この有効期間を変更できます。失効したイベントはイベント・キューから削除されます。

関連項目:

- Oracle Databaseアドバンスト・キューイングを使用したセキュアなキューの構成については、『Oracle Databaseアドバンスト・キューイング・ユーザーズ・ガイド』を参照してください。
- DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: スケジューラによって呼び出されるイベントによるジョブ状態の監視

29.10.4.2 イベントを呼び出すようにジョブを変更する方法

ジョブのジョブ状態イベントの発生を可能にするには、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャを使用してraise_eventsジョブ属性のビット・フラグをオンにします。

各ビット・フラグは、イベントの呼出しの対象になる様々なジョブ状態を表します。たとえば、最下位ビットをオンにすることで、job startedイベントを呼び出すことができます。複数の状態変化イベント・タイプを1回のコールで使用可能にするには、必要なビット・フラグの値を加算し、その結果を引数としてSET_ATTRIBUTEに提供します。

次の例では、ジョブdw_reportsに対して複数の状態変更イベントを有効にしています。次のイベント・タイプが有効になっていますが、両方ともなんらかのエラーを示しています。

- JOB_FAILED
- JOB_SCH_LIM_REACHED

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE('dw_reports', 'raise_events',
   DBMS_SCHEDULER.JOB_FAILED + DBMS_SCHEDULER.JOB_SCH_LIM_REACHED);
END;
/
```

ノート:

raise_events ジョブ属性を指定して JOB_OVER_MAX_DUR イベントを使用可能にする必要はありません。このイベントは常に使用可能になっています。

関連項目:

ジョブ状態ビット・フラグの名前と値は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_SCHEDULER.SET_ATTRIBUTEに関する項を参照してください。

親トピック: スケジューラによって呼び出されるイベントによるジョブ状態の監視

29.10.4.3 ジョブの状態イベントのアプリケーションでの使用

ジョブ状態イベントを使用するには、アプリケーションがスケジューラ・イベント・キューSYS.SCHEDULER$_EVENT_QUEUEをサブスクライブする必要があります。このキューはセキュアなキューであり、所有者はSYSです。

あるユーザーについてこのキューのサブスクリプションを作成するには、次の処理を実行します。

1. SYSユーザーまたはMANAGE ANY QUEUE権限のあるユーザーでデータベースにログインします。
2. 新規または既存のエージェントを使用してキューをサブスクライブします。
3. パッケージ・プロシージャDBMS_AQADM.ENABLE_DB_ACCESSを次のように実行します。

```
DBMS_AQADM.ENABLE_DB_ACCESS(agent_name, db_username);
```

agent_nameは、イベント・キューのサブスクライブに使用したエージェントを表し、db_usernameは、サブスクリプションを作成する対象のユーザーです。

ユーザーにデキュー権限を付与する必要はありません。スケジューラ・イベント・キューに関するデキュー権限は、PUBLICに付与されます。

または、次の例に示すように、ユーザーがADD_EVENT_QUEUE_SUBSCRIBERプロシージャを使用してスケジューラ・イベント・キューをサブスクライブすることもできます。

```
DBMS_SCHEDULER.ADD_EVENT_QUEUE_SUBSCRIBER(subscriber_name);
```

この例のsubscriber_nameは、スケジューラ・イベント・キューのサブスクライブに使用されるOracle Databaseアドバンスト・キューイング(AQ)のエージェント名です。(NULLの場合は、呼出しユーザーのユーザー名でエージェントが作成されます。)このコールによって、スケジューラ・イベント・キューのサブスクリプションが作成され、指定エージェントを使用してデキューする許可がユーザーに付与されます。サブスクリプションはルールベースです。ルールによって、ユーザーが許可されるのは、所有するジョブが呼び出したイベントの参照のみで、その他のメッセージはすべて除外されます。サブスクリプションが使用可能になると、ユーザーは一定の間隔でメッセージをポーリングするか、またはメッセージが通知されるようにAQに登録できます。

詳細は、Oracle Databaseアドバンスド・キューイング・ユーザーズ・ガイドを参照してください。

親トピック: スケジューラによって呼び出されるイベントによるジョブ状態の監視

29.10.4.3.1 スケジューラ・イベント・キュー

スケジューラ・イベント・キューSYS.SCHEDULER$_EVENT_QUEUEのタイプはscheduler$_event_infoです。このタイプの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

29.10.5 電子メール通知によるジョブ状態の監視

スケジューラでは、ジョブの状態が変化したとき、イベントを発生することができます。

- 電メール通知について
  ジョブの状態が変化したときに電子メール通知を送信するようにジョブを構成できます。
- ジョブに対する電子メール通知の追加
  ジョブに対する電子メール通知を追加するには、DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATIONパッケージ・プロシージャを使用します。
- ジョブに対する電子メール通知の削除
  ジョブに対する電子メール通知を削除するには、DBMS_SCHEDULER.REMOVE_JOB_EMAIL_NOTIFICATIONパッケージ・プロシージャを使用します。
- 電子メール通知情報の表示
  *_SCHEDULER_NOTIFICATIONSビューを問い合せることによって、現在の電子メール通知に関する情報を表示できます。

親トピック: ジョブの監視

29.10.5.1 電子メール通知について

ジョブの状態が変化したときに電子メール通知を送信するようにジョブを構成できます。

表29-13に、電子メールを送信できるジョブ状態イベントを示します。電子メール通知は、指定したジョブ状態イベントのリストに含まれる任意のイベントをトリガーとして、複数の受信者に送信できます。また、フィルタ条件を指定して、その条件に一致する通知ジョブ状態イベントのみを生成することもできます。メッセージの件名と本文の両方に、ジョブ所有者、ジョブ名、イベント・タイプ、エラー・コード、エラー・メッセージなどの変数を含めることができます。これらの変数の値は、電子メール通知が送信される前に、スケジューラによって自動的に設定されます。

単一のジョブに対して複数のジョブ状態電子メール通知を構成できます。これらの通知は、ジョブ状態イベント・リスト、受信者およびフィルタ条件によって区別できます。

たとえば、ジョブがエラー・コード600または700で失敗した場合は常に主任DBAとシニアDBAの1人に電子メールを送信するようにジョブを構成できます。また、同じジョブに対して、ジョブがスケジュールどおり開始されなかった場合は主任DBAのみに通知を送信するように構成することもできます。

電子メール通知を送信するようにジョブを構成するには、スケジューラ属性`email_server`に、電子メールの送信に使用するSMTPサーバーのアドレスを設定する必要があります。また、必要に応じて、送信者未指定のジョブに対しては、スケジューラ属性`email_sender`に、デフォルトの送信者電子メール・アドレスを設定できます。

スケジューラでは、SMTPサーバーとの通信におけるSSLプロトコルおよびTLSプロトコルがサポートされています。また、認証を必要とするSMTPサーバーもサポートされています。

関連項目:

電子メール通知関連の属性の設定の詳細は、「スケジューラのプリファレンスの設定」

親トピック: 電子メール通知によるジョブ状態の監視

29.10.5.2 ジョブに対する電子メール通知の追加

ジョブに対する電子メール通知を追加するには、`DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION`パッケージ・プロシージャを使用します。

たとえば、次のプロシージャは、`OED_JOB`ジョブの電子メール通知を追加します。

```
BEGIN
 DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION (
  job_name   =>  'EOD_JOB',
  recipients =>  'jsmith@example.com, rjones@example.com',
  sender     =>  'do_not_reply@example.com',
  subject    =>  'Scheduler Job Notification-%job_owner%.%job_name%-%event_type%',
  body       =>   '%event_type% occurred at %event_timestamp%. %error_message%',
  events     =>  'JOB_FAILED, JOB_BROKEN, JOB_DISABLED, JOB_SCH_LIM_REACHED');
END;
/
```

`subject`および`body`引数では、%文字で囲まれた変数が使用されています。複数の受信者および複数のイベントを指定した場合、指定したイベントのいずれかが呼び出されると各受信者に通知されます。これを確認するには、`USER_SCHEDULER_NOTIFICATIONS`ビューを問い合せます。

```
SELECT JOB_NAME, RECIPIENT, EVENT FROM USER_SCHEDULER_NOTIFICATIONS;
JOB_NAME    RECIPIENT            EVENT
----------- -------------------- -------------------
EOD_JOB     jsmith@example.com   JOB_FAILED
```

```
EOD_JOB     jsmith@example.com   JOB_BROKEN
EOD_JOB     jsmith@example.com   JOB_SCH_LIM_REACHED
EOD_JOB     jsmith@example.com   JOB_DISABLED
EOD_JOB     rjones@example.com   JOB_FAILED
EOD_JOB     rjones@example.com   JOB_BROKEN
EOD_JOB     rjones@example.com   JOB_SCH_LIM_REACHED
EOD_JOB     rjones@example.com   JOB_DISABLED
```

ジョブに対して異なる通知を構成するたびに、ADD_JOB_EMAIL_NOTIFICATIONをコールします。job_nameとrecipientsの指定は必須です。他のすべての引数には、デフォルト値があります。デフォルトのsenderは、前の項で説明したように、スケジューラ属性によって定義されます。subject、bodyおよびevents引数のデフォルト値は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のADD_JOB_EMAIL_NOTIFICATIONプロシージャに関する項を参照してください。

次の例では、同じジョブに対して、異なるイベントに関する追加の電子メール通知を構成しています。この例では、sender、subjectおよびbody引数のデフォルト値をそのまま使用しています。

```
BEGIN
 DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION (
  job_name          =>  'EOD_JOB',
  recipients        =>  'jsmith@example.com',
  events            =>  'JOB_OVER_MAX_DUR');
END;
/
```

この例では、events引数も省略して、イベントのデフォルト値をそのまま使用しています。

次の例は最初の例と類似していますが、この場合、フィルタ条件を使用して、エラー番号が600または700でジョブが失敗した場合にのみ電子メール通知を送信するように指定しています。

```
BEGIN
 DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION (
  job_name          => 'EOD_JOB',
  recipients        => 'jsmith@example.com, rjones@example.com',
  sender            => 'do_not_reply@example.com',
  subject           => 'Job Notification-%job_owner%.%job_name%-%event_type%',
  body              =>  '%event_type% at %event_timestamp%. %error_message%',
  events            => 'JOB_FAILED',
  filter_condition => ':event.error_code=600 or :event.error_code=700');
END;
/
```

関連項目:

『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のADD_JOB_EMAIL_NOTIFICATIONプロシージャに関する項を参照してください。

親トピック: 電子メール通知によるジョブ状態の監視

29.10.5.3 ジョブに対する電子メール通知の削除

ジョブに対する電子メール通知を削除するには、DBMS_SCHEDULER.REMOVE_JOB_EMAIL_NOTIFICATIONパッケージ・プロシージャを使用します。

たとえば、次のプロシージャは、OED_JOBジョブの電子メール通知を削除します。

```
BEGIN
 DBMS_SCHEDULER.REMOVE_JOB_EMAIL_NOTIFICATION (
  job_name    =>  'EOD_JOB',
```

```
  recipients =>  'jsmith@example.com, rjones@example.com',
  events     =>  'JOB_DISABLED, JOB_SCH_LIM_REACHED');
END;
/
```

複数の受信者および複数のイベントを指定した場合、指定した各イベントの通知が各受信者に対して削除されます。前の項で説明したものと同じ問合せを実行した場合、結果は次のようになります。

```
SELECT JOB_NAME, RECIPIENT, EVENT FROM USER_SCHEDULER_NOTIFICATIONS;
JOB_NAME    RECIPIENT            EVENT
----------- -------------------- -------------------
EOD_JOB     jsmith@example.com   JOB_FAILED
EOD_JOB     jsmith@example.com   JOB_BROKEN
EOD_JOB     rjones@example.com   JOB_FAILED
EOD_JOB     rjones@example.com   JOB_BROKEN
```

REMOVE_JOB_EMAIL_NOTIFICATION引数を指定する場合、次のルールが別途適用されます。

- events引数をNULLのままにすると、指定した受信者に対して、すべてのイベントの通知が削除されます。
- recipients引数をNULLのままにすると、すべての受信者に対して、指定したイベントの通知が削除されます。
- recipientsとeventsの両方をNULLのままにすると、該当ジョブのすべての通知が削除されます。
- 今まで通知を作成したことのない受信者とイベントを指定した場合、エラーは生成されません。

関連項目:

『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のREMOVE_JOB_EMAIL_NOTIFICATIONプロシージャに関する項を参照してください。

親トピック: 電子メール通知によるジョブ状態の監視

29.10.5.4 電子メール通知情報の表示

*_SCHEDULER_NOTIFICATIONSビューを問い合せることによって、現在の電子メール通知に関する情報を表示できます。

関連項目:

これらのビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 電子メール通知によるジョブ状態の監視

30 Oracle Schedulerの管理

Oracle Schedulerを構成、管理、監視およびトラブルシューティングできます。

ノート:

この章では、DBMS_SCHEDULER パッケージを使用して Oracle Scheduler を管理する方法について説明します。Oracle Enterprise Manager Cloud Control を使用して、数多くの同じタスクを実行できます。

DBMS_SCHEDULER の詳細は『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を、Oracle Scheduler の各ページの詳細は Cloud Control のオンライン・ヘルプを参照してください。

CDB での Oracle Scheduler の使用の詳細は、『Oracle Multitenant 管理者ガイド』を参照してください。

- Oracle Schedulerの構成
  Oracle Schedulerの構成には、権限およびプリファレンスの設定、Oracle Schedulerエージェントを使用したリモート・ジョブの実行などのタスクが含まれます。
- スケジューラの監視と管理
  現在アクティブなウィンドウとそれに関連付けられているリソース・プランの表示、現在実行中のジョブに関する情報の表示、ウィンドウおよびジョブのログの監視と管理、およびスケジューラのセキュリティ管理を実行できます。
- インポート/エクスポートおよびスケジューラ
  スケジューラ・オブジェクトをエクスポートするには、データ・ポンプ・ユーティリティ(impdpおよびexpdp)を使用します。
- スケジューラのトラブルシューティング
  スケジューラでの問題をトラブルシューティングできます。
- スケジューラの使用例
  スケジューラの使用例を示します。
- スケジューラの参照情報
  スケジューラに関連するいくつかの権限およびデータ・ディクショナリ・ビューがあります。

親トピック: データベース・リソースの管理とタスクのスケジューリング

30.1 Oracle Schedulerの構成

Oracle Schedulerの構成には、権限およびプリファレンスの設定、Oracle Schedulerエージェントを使用したリモート・ジョブの実行などのタスクが含まれます。

- Oracle Schedulerの権限の設定
  Oracle Schedulerのすべての管理タスクを実行するには、SCHEDULER_ADMINロールが必要です。通常、データベース管理者(DBA)は、DBAロールの一部として、ADMINオプション付きでこのロールをすでに持っています。
- スケジューラのプリファレンスの設定
  システム全体にわたる設定可能なスケジューラのプリファレンスがいくつかあります。これらのプリファレンスを設定するには、DBMS_SCHEDULERパッケージのSET_SCHEDULER_ATTRIBUTEプロシージャでスケジューラの属性を設定します。
- Oracle Schedulerエージェントを使用したリモート・ジョブの実行
  Oracle Schedulerエージェントは、リモート・ジョブをスケジュールおよび実行できます。

親トピック: Oracle Schedulerの管理

30.1.1 Oracle Schedulerの権限の構成

Oracle Schedulerのすべての管理タスクを実行するには、SCHEDULER_ADMINロールが必要です。通常、データベース管理者(DBA)は、DBAロールの一部として、ADMINオプション付きでこのロールをすでに持っています。

たとえば、ユーザーSYSおよびSYSTEMには、DBAロールが付与されます。このロールは、次の文を発行して別の管理者に付与できます。

```
GRANT SCHEDULER_ADMIN TO username;
```

SCHEDULER_ADMINロールは、権限を付与されたユーザーがコードを任意のユーザーとして実行できる強力なロールであるため、かわりにスケジューラの個々のシステム権限を付与することを考慮してください。オブジェクトおよびシステム権限は通常のSQL付与構文を使用して付与されます。たとえば、データベース管理者が次の文を発行したとします。

```
GRANT CREATE JOB TO scott;
```

この文が実行されると、scottは、自分のスキーマ内にジョブ、スケジュール、プログラムおよびFile Watcherを作成できます。別の例として、データベース管理者は次の文を発行できます。

```
GRANT MANAGE SCHEDULER TO adam;
```

この文が実行されると、adamはウィンドウ、ジョブ・クラスまたはウィンドウ・グループを、作成、変更または削除できます。また、adamはスケジューラ属性の設定と取得、およびスケジューラ・ログのパージもできます。

チェーンの各権限の設定

スケジューラ・チェーンでは、基礎となるOracleルール・エンジン・オブジェクトと、それに関連する権限を使用します。ユーザーが自分のスキーマにチェーンを作成するには、自分のスキーマにルール、ルール・セットおよび評価コンテキストを作成するためのルール・エンジン権限に加え、CREATE JOB権限が必要です。これらの権限は、次の文を発行することで付与できます。

```
GRANT CREATE RULE, CREATE RULE SET, CREATE EVALUATION CONTEXT TO user;
```

ユーザーが、別のスキーマにチェーンを作成するには、自分のスキーマ以外のスキーマにルール、ルール・セットおよび評価コンテキストを作成するためのルール・エンジン権限に加え、CREATE ANY JOB権限が必要です。これらの権限は、次の文を発行することで付与できます。

```
GRANT CREATE ANY RULE, CREATE ANY RULE SET,
   CREATE ANY EVALUATION CONTEXT TO user;
```

各自のスキーマ以外のスキーマのチェーンを変更または削除するには、ルール、ルール・セットおよび評価コンテキストについて、対応する各システムのルール・エンジン権限が必要です。

関連項目:

チェーンの権限の詳細は、「チェーンのタスクとそのプロシージャ」を参照してください。

親トピック: Oracle Schedulerの構成

30.1.2 スケジューラのプリファレンスの設定

システム全体にわたる設定可能なスケジューラのプリファレンスがいくつかあります。これらのプリファレンスを設定するには、DBMS_SCHEDULERパッケージのSET_SCHEDULER_ATTRIBUTEプロシージャでスケジューラの属性を設定します。

これらの属性を設定するには、MANAGE SCHEDULER権限が必要です。属性は次のとおりです。

- default_timezone

  この属性を設定することは非常に重要です。カレンダ構文式を使用する繰返しジョブおよびウィンドウには、繰返し間隔に使用するタイムゾーンに関する情報が必要です。「スケジューラのカレンダ指定構文の使用方法」を参照してください。通常は、start_dateからタイムゾーンを取得しますが、start_dateが指定されていない場合(珍しいことではありません)は、default_timezone Scheduler属性から取得します。

  スケジューラは、default_timezoneの値をオペレーティング・システム環境から導出します。オペレーティング・システムから互換性のある値を検出できない場合、スケジューラはdefault_timezoneをNULLに設定します。

  default_timezoneが正しく設定されていることを確認することは非常に重要であり、そうでない場合は正しく設定してください。これを確認するには、次の問合せを実行します。

```
SELECT DBMS_SCHEDULER.STIME FROM DUAL;

STIME
---------------------------------------------------------------------------
28-FEB-12 09.04.10.308959000 PM UTC
```

  確実に夏時間が適用されるようにするには、default_timezoneを、'-8:00'のような絶対タイム・ゾーン・オフセットではなく、リージョン名に設定することをお薦めします。たとえば、データベースが米国フロリダ州のマイアミにある場合は、次の文を発行します。

```
DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE('default_timezone','US/Eastern');
```

  同様に、データベースがパリにある場合は、この属性を'Europe/Warsaw'に設定します。有効なリージョン名のリストを表示するには、次の問合せを実行します。

```
SELECT DISTINCT TZNAME FROM V$TIMEZONE_NAMES;
```

  default_timezone属性を適切に設定しないと、繰返しジョブや繰返しウィンドウに使用されるデフォルトのタイム・ゾーンは、SYSTIMESTAMPから取り出される絶対オフセット(データベースのオペレーティング・システム環境のタイム・ゾーン)となります。これは、start_dateが設定されていない繰返しジョブや繰返しウィンドウでは、夏時間調整が適用されないことになります。

- email_server

  この属性には、ジョブの状態イベントに対する電子メール通知を送信するためにスケジューラで使用されるSMTPサーバーのアドレスを指定します。書式は次のとおりです。

```
host[:port]
```

  ここで:

  - hostは、SMTPサーバーのホスト名またはIPアドレスです。
  - portは、SMTPサーバーがリスニングするTCPポートです。指定しない場合は、デフォルト・ポート25が使用されます。

  この属性が指定されていないか、NULLまたは無効なSMTPサーバー・アドレスに設定されていると、ジョブの状態に関する電子メール通知をスケジューラで送信できません。

- email_sender

  この属性には、ジョブ状態の電子メール通知の送信者のデフォルト電子メール・アドレスを指定します。有効な電子メール・アドレスを指定する必要があります。この属性が設定されていないか、NULLに設定されている場合、送信者ア

ドレスが指定されていないジョブ状態の電子メール通知では、電子メール・ヘッダーのFROMアドレスが指定されません。

- `email_server_credential`

  この属性は、既存の資格証明オブジェクトのスキーマおよび名前を指定します。デフォルトはNULLです。

  電子メール通知が送信されると、スケジューラは`email_server_credential`が指している資格証明オブジェクトを確認して、SYSがEXECUTEオブジェクト権限を持つ有効な資格証明オブジェクトであるかどうかを判断します。`email_server`属性に指定されているSMTPサーバーが認証を必要とする場合、スケジューラは指定された資格証明オブジェクトに格納されているユーザー名とパスワードを使用して電子メール・サーバーで認証します。

  `email_server_credential`を指定している場合、`email_server`属性には認証を必要とするSMTPサーバーを指定する必要があります。

  `email_server_credential`を指定していない場合、スケジューラでは認証が構成されていないSMTPサーバーを介して通知電子メールを送信できます。

- `email_server_encryption`

  この属性は、このSMTPサーバー接続で暗号化が有効であるかどうかを示し、有効である場合には暗号化を開始する時点と使用するプロトコルも示します。

  `email_server_encryption`の値は、次のとおりです。

  `NONE`: デフォルト値で、暗号化が有効でないことを示します。

  `SSL_TLS`: 接続が確立された当初からSSLまたはTLSが使用されることを示します。両側から、どのプロトコルが最も安全であるかが判断されます。これが、このパラメータに最もよく使用される設定です。

  `STARTTLS`: 暗号化なしで接続を開始するものの、STARTTLSコマンドの指示があると、電子メール・サーバーがTLSを使用して暗号化を開始することを示します。

- `event_expiry_time`

  この属性を使用すると、スケジューラが生成したジョブの状態イベントが終了する(自動的にスケジューラ・イベント・キューからパージされる)までの時間(秒)を設定できます。NULLに設定すると、ジョブの状態イベントは24時間後に期限切れになります。

- `log_history`

  この属性は、ジョブ・ログとウィンドウ・ログの両方のログ・エントリが保持される日数を制御します。これにより、ログが無制限に大きくなるのを防ぐことができます。有効値の範囲は、0から1000000です。0に設定すると、履歴は保持されません。デフォルト値は30です。ジョブ・クラスの`log_history`属性に値を設定することにより、ジョブ・クラス・レベルでこの値をオーバーライドできます。

SET_SCHEDULER_ATTRIBUTEプロシージャの構文は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: Oracle Schedulerの構成

30.1.3 Oracle Scheduler Agentを使用したリモート・ジョブの実行

Oracle Schedulerエージェントは、リモート・ジョブをスケジュールおよび実行できます。

Oracle Scheduler Agentを使用して、スケジューラは次の2種類のリモート・ジョブをスケジュールおよび実行できます。

- リモート・データベース・ジョブ: リモート・データベース・ジョブはOracle Scheduler Agentを介して実行する必要があります。リモート・データベースと同じホストにエージェントをインストールすることをお薦めします。

リモート・データベース・ジョブを実行する場合は、Oracle Database 11g リリース2 (11.2)以降のスケジューラ・エージェントが必要です。

- リモート外部ジョブ: リモート外部ジョブは、スケジューラ・エージェントがインストールされているのと同じホストで実行されます。

  リモート外部ジョブのみを実行する場合は、Oracle Database 11g リリース1 (11.1)のスケジューラ・エージェントで十分です。

リモート外部ジョブが実行されるすべてのホストにスケジューラ・エージェントをインストールする必要があります。リモート・データベース・ジョブが実行されるリモート・データベースが動作するすべてのホストにスケジューラ・エージェントをインストールする必要があります。

リモート・ジョブが実行される各データベースでは、「リモート・ジョブを実行するためのデータベースの設定」の説明に従って、データベースとリモート・スケジューラ・エージェント間のセキュアな通信を有効にするための初期設定を行う必要があります。

リモート・ジョブの有効化には、次のステップが含まれます。

1. リモート・ジョブを実行するためのデータベースの設定の有効化と無効化
2. リモート・ホストでのスケジューラ・エージェントのインストールと構成
3. スケジューラ・エージェントによるタスクの実行

- リモート・ジョブのためのデータベースの有効化と無効化
  リモート・ジョブ用のデータベースを設定すること、およびリモート・ジョブ用のデータベースを無効にすることができます。
- リモート・ホストでのスケジューラ・エージェントのインストールと構成
  特定のホスト上のリモート・ジョブを実行できるようにするには、スケジューラ・エージェントをインストールおよび構成する必要があります。
- スケジューラ・エージェントによるタスクの実行
  スケジューラ・エージェントは、外部ジョブとデータベース・ジョブをリモート・ホストでスケジュールおよび実行できるようにするスタンドアロン・プログラムです。スケジューラ・エージェントの起動と停止には、UNIXおよびLinuxではschagentユーティリティを使用し、WindowsではOracleSchedulerExecutionAgentサービスを使用します。

関連項目:

- 「リモート外部ジョブについて」
- リモート・データベース・ジョブの詳細は、「データベース・ジョブ」

親トピック: Oracle Schedulerの構成

30.1.3.1 リモート・ジョブを実行するためのデータベースの設定の有効化と無効化

リモート・ジョブ用のデータベースの設定、およびリモート・ジョブ用のデータベースを無効にすることができます。

- リモート・ジョブを実行するためのデータベースの設定
  データベースでリモートのスケジューラ・エージェントを使用してジョブを実行するには、その前に、データベースを正しく構成し、そのエージェントをデータベースに登録しておく必要があります。
- リモート・ジョブの無効化
  データベースのリモート・ジョブは、REMOTE_SCHEDULER_AGENTユーザーを削除することで、無効にできます。

親トピック: Oracle Schedulerエージェントを使用したリモート・ジョブの実行

30.1.3.1.1 リモート・ジョブを実行するためのデータベースの設定

データベースでリモートのスケジューラ・エージェントを使用してジョブを実行するには、その前に、データベースを正しく構成し、そのエージェントをデータベースに登録しておく必要があります。

この項では、データベースに必要なエージェント登録パスワードなどの構成について説明します。「スケジューラ・エージェントのデータベースへの登録」に示されているように、後でデータベースを登録します。

登録できるスケジューラ・エージェントの数を制限して、パスワードの有効期限を設定できます。

リモート・ジョブを作成および実行するデータベースごとに1回、次のすべてのステップを実行します。

リモート・ジョブを作成および実行するデータベースを設定するには:

1. 共有サーバーが使用可能であることを確認します。

   「共有サーバーの使用可能化」を参照してください。

   同じデータベースで複数のスケジューラ・エージェントを使用する場合は、それらのエージェントが並行して動作しようとした場合のエラーを回避するために、SHARED_SERVERSデータベース初期化パラメータを十分に高い値に設定します。

   ノート:

   マルチテナント・モードで実行している場合、CDB$ROOT で匿名アカウントのロックを解除する必要があります。

   SQL*Plus を使用して、SYS ユーザーとして CDB$ROOT に接続し、次のコマンドを入力します。

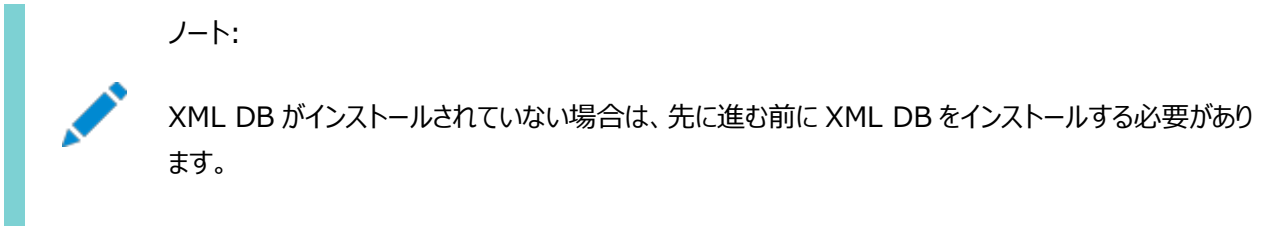
   ```
   SQL> alter session set container = CDB$ROOT;
   SQL> alter user anonymous account unlock container=current;
   ```

2. SQL*Plusを使用して、SYSユーザーとしてデータベースに接続します(マルチテナント・モード下のプラガブル・データベースを指定)。

3. 次のコマンドを入力して、XML DBオプションがインストールされていることを確認します。

   ```
   SQL> DESC RESOURCE_VIEW
   ```

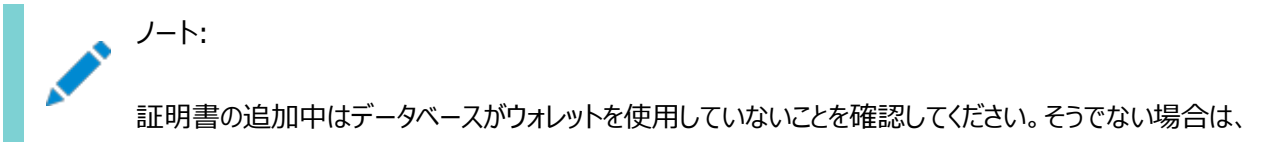
   XML DBがインストールされていない場合は、このコマンドで「オブジェクトが存在しません。」エラーが戻ります。

   ノート:

   XML DB がインストールされていない場合は、先に進む前に XML DB をインストールする必要があります。

4. HTTPS接続を使用している場合は、次に示すようにデータベース・ウォレットに証明書を追加します。

   ノート:

   証明書の追加中はデータベースがウォレットを使用していないことを確認してください。そうでない場合は、

データベースを停止している間に証明書を追加して、その後でデータベースを起動します。

a. ORACLE_HOME/admin/$ORACLE_SID/xdb_walletディレクトリに既存のデータベース・ウォレットがない場合は、orapkiコマンドライン・ユーティリティを使用してウォレットを作成します。たとえば:

```
orapki wallet create -wallet $ORACLE_HOME/admin/$ORACLE_SID/xdb_wallet -
pwd wallet_password -auto_login
```

b. orapkiコマンドライン・ユーティリティを使用してウォレットに証明書を追加します。たとえば:

```
orapki wallet add -wallet $ORACLE_HOME/admin/$ORACLE_SID/xdb_wallet -dn
CN=fully_qualified_domain_name -self_signed -pwd wallet_password -
validity number_of_days -keysize
key_size_for_the_certificate(512|1024|2048)
```

関連項目:

orapkiユーティリティを使用してデータベース・ウォレットに証明書を追加する際の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください

5. 次のように、データベースへのHTTP(S)接続を有効にします。

a. Oracle XML DBM HTTP(S)サーバーが有効であるかどうかを判断します。

HTTP接続の場合は、次のコマンドを実行します。

```
SQL> SELECT DBMS_XDB_CONFIG.GETHTTPPORT() FROM DUAL;
```

HTTPS接続の場合は、次のコマンドを実行します。

```
SQL> SELECT DBMS_XDB_CONFIG.GETHTTPSPORT() FROM DUAL;
```

文が0を返した場合、Oracle XML DBM HTTP(S)サーバーは無効になっています。

b. 0(ゼロ)以外のポートでOracle XML DB HTTP(S)サーバーを有効にするには、SYSとしてログインし、次のコマンドを実行します。

HTTP接続を使用している場合:

```
SQL> EXEC DBMS_XDB_CONFIG.SETHTTPPORT (port);
SQL> COMMIT;
```

HTTPS接続を使用している場合:

```
SQL> EXEC DBMS_XDB_CONFIG.SETHTTPSPORT (port);
SQL> COMMIT;
```

portは、データベースがHTTP(S)接続をリスニングするTCPポート番号です。

portには1から65536の整数を指定しますが、UNIXおよびLinuxの場合は、1023より大きな値にします。まだ使用されていないポート番号を選択します。

エージェント登録プロシージャ中、後で正確なプラガブル・データベースをスケジューラ・エージェントが特定できるように、各プラガブル・データベースでは固有のポート番号を使用する必要があります。

ノート:

- これにより、Oracle Real Application Clusters データベースのすべてのインスタンスで HTTP(S)接続が使用可能になります。
- Oracle Database リリース 18c 以降は、Oracle Scheduler エージェントで HTTPS 接続がサポートされています。

6. 次のコマンドを使用して、スクリプトprvtrsch.plbを実行します。

```
SQL> @?/rdbms/admin/prvtrsch.plb
```

7. SET_AGENT_REGISTRATION_PASSプロシージャを使用して、スケジューラ・エージェントの登録パスワードを設定します。

   次の例では、エージェント登録パスワードをmypasswordに設定します。

```
BEGIN
  DBMS_SCHEDULER.SET_AGENT_REGISTRATION_PASS('mypassword');
END;
/
```

ノート:

エージェント登録パスワードを設定するには、MANAGE SCHEDULER 権限が必要です。
SET_AGENT_REGISTRATION_PASS プロシージャの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

実際の登録は、「スケジューラ・エージェントのデータベースへの登録」で行います。

親トピック: リモート・ジョブを実行するためのデータベースの設定の有効化と無効化

30.1.3.1.2 リモート・ジョブの無効化

データベースのリモート・ジョブは、REMOTE_SCHEDULER_AGENTユーザーを削除することで、無効にできます。

リモート・ジョブを無効にするには:

- 次のSQL文を発行します。

```
DROP USER REMOTE_SCHEDULER_AGENT CASCADE;
```

新規スケジューラ・エージェントの登録とリモート・ジョブの実行は、prvtrsch.plbを再度実行するまで無効(使用禁止)になります。

親トピック: リモート・ジョブを実行するためのデータベースの設定の有効化と無効化

30.1.3.2 リモート・ホストでのスケジューラ・エージェントのインストールと構成

特定のホスト上のリモート・ジョブを実行できるようにするには、スケジューラ・エージェントをインストールおよび構成する必要があります。

スケジューラ・エージェントをインストールおよび構成した後、「スケジューラ・エージェントによるタスクの実行」で説明しているように、ホスト上のスケジューラ・エージェントを登録し、起動する必要があります。スケジューラ・エージェントも、独自のOracleホームに

インストールする必要があります。

スケジューラ・エージェントをリモート・ホストにインストールおよび構成するには:

1. スケジューラ・エージェント・ソフトウェアをダウンロードまたは入手します。このソフトウェアはOracle Databaseクライアントのインストール・メディア(Databaseのインストール・メディア)に収録されており、次のサイトからオンラインで入手することもできます。

   http://www.oracle.com/technology/software/products/database

2. 最初に、エージェントを登録するデータベースが正しく設定されていることを確認します。

   手順は、「リモート・ジョブを実行するためのデータベースの設定の有効化と無効化」を参照してください。

3. スケジューラ・エージェントをインストールするホストにログインします。このホストがリモート・ジョブを実行します。

   - Windowsの場合は、管理者としてログインします。
   - UNIXおよびLinuxの場合は、スケジューラ・エージェントを実行する予定のユーザーとしてログインします。このユーザーに特別な権限は必要ありません。

4. Oracle Databaseクライアント製品のインストール・メディアからOracle Universal Installer(OUI)を実行します。

   - Windowsの場合は、`setup.exe`を実行します。
   - UNIXおよびLinuxの場合は、次のコマンドを使用します。

     ```
     /directory_path/runInstaller
     ```

   `directory_path`は、Oracle Databaseクライアント製品のインストール・メディアへのパスです。

5. 「インストール・タイプの選択」ページで、「カスタム」を選択して、「次へ」をクリックします。
6. 「製品言語の選択」ページで、対象の言語を選択して、「次へ」をクリックします。
7. 「インストール場所の指定」ページで、エージェントの新しいOracleホームのパスを入力して、「次へ」をクリックします。
8. 「使用可能な製品コンポーネント」ページで、「Oracle Scheduler Agent」を選択して、「次へ」をクリックします。
9. 「Oracle Database Scheduler Agent」ページで、次の手順を実行します。

   - 「Scheduler Agentのホスト名」フィールドに、スケジューラ・エージェントをインストールするコンピュータのホスト名を入力します。
   - 「Scheduler Agentのポート番号」フィールドに、スケジューラ・エージェントが接続をリスニングするTCPポート番号を入力するか、デフォルトを使用して、「次へ」をクリックします。

     1から65535の整数を選択します。UNIXおよびLinuxの場合は、1023より大きい値にする必要があります。選択するポート番号がまだ使用されていないことを確認します。

     OUIで、一連の前提条件チェックが実行されます。前提条件チェックが失敗した場合は、問題を解決してから「次へ」をクリックします。

10. 「サマリー」ページで「終了」をクリックします。
11. (UNIXおよびLinuxのみ)OUIがスクリプト`root.sh`の実行を求めるプロンプトを表示したら、`root`ユーザーとして次のコマンドを入力します。

    ```
    script_path/root.sh
    ```

このスクリプトは、エージェントのインストール用に選択したディレクトリに格納されています。

スクリプトが完了したら、「構成スクリプトの実行」ダイアログ・ボックスで「OK」をクリックします。

12. インストールが完了したら、「閉じる」をクリックしてOUIを終了します。

13. テキスト・エディタを使用して、スケジューラ・エージェントのホーム・ディレクトリにあるエージェント構成パラメータ・ファイル`schagent.conf`の`PORT`=ディレクティブのポート番号を確認します。

14. リモート・ホスト上のファイアウォール・ソフトウェア、またはそのホストを保護する他のファイアウォールに、スケジューラ・エージェントに対応するための例外があることを確認します。

親トピック: Oracle Schedulerエージェントを使用したリモート・ジョブの実行

30.1.3.3 スケジューラ・エージェントによるタスクの実行

スケジューラ・エージェントは、外部ジョブとデータベース・ジョブをリモート・ホストでスケジュールおよび実行できるようにするスタンドアロン・プログラムです。スケジューラ・エージェントの起動と停止には、UNIXおよびLinuxでは`schagent`ユーティリティを使用し、Windowsでは`OracleSchedulerExecutionAgent`サービスを使用します。

- schagentユーティリティについて
  実行可能ユーティリティ`schagent`は、Windows、UNIXおよびLinux上のエージェントの特定のタスクを実行します。
- Windowsでのスケジューラ・エージェントの使用
  Windowsのスケジューラ・エージェント・サービスは、インストール中に自動的に作成されて起動されます。サービス名の末尾は、`OracleSchedulerExecutionAgent`になっています。
- スケジューラ・エージェントの起動
  スケジューラ・エージェントを起動すると、それが存在するホストでのリモート・ジョブの実行が可能になります。
- スケジューラ・エージェントの停止
  スケジューラ・エージェントを停止すると、スケジューラ・エージェントがあるホストでリモート・ジョブを実行できなくなります。
- スケジューラ・エージェントのデータベースへの登録
  スケジューラ・エージェントの構成が終わるとすぐに、リモート・ジョブを実行する予定の1つ以上のデータベースでエージェントを登録できます。

親トピック: Oracle Schedulerエージェントを使用したリモート・ジョブの実行

30.1.3.3.1 schagentユーティリティについて

実行可能ユーティリティ`schagent`は、Windows、UNIXおよびLinux上のエージェントの特定のタスクを実行します。

表30-1に、`schagent`のオプションが示されています。

次の適切な構文およびオプションを指定して、`schagent`を使用します。

たとえば:

UNIXおよびLinux: `AGENT_HOME/bin/schagent -status`

Windows: `AGENT_HOME/bin/schagent.exe -status`

表30-1 schagentのオプション

オプション	説明
`-start`	スケジューラ・エージェントを開始します。

オプション	説明
	UNIX と Linux のみ
`-stop`	スケジューラ・エージェントに現在実行中のすべてのジョブを停止するようにプロンプトを表示してから、実行を正常に停止します。 UNIX と Linux のみ
`-abort`	最初にジョブを停止せずに、スケジューラ・エージェントを強制的に停止します。Oracle Database 11g リリース 2 (11.2)以降。 UNIX と Linux のみ
`-status`	ローカルに実行中のスケジューラ・エージェントに関する、バージョン、稼働時間、エージェントの起動以降に実行されたジョブの合計数、現在実行中のジョブの数およびその説明を返します。
`-registerdatabase`	エージェントのホスト・コンピュータでリモート・ジョブを実行する予定の基本データベースまたは追加データベースにスケジューラ・エージェントを登録します。
`-unregisterdatabase`	データベースからエージェントを登録解除します。

親トピック: スケジューラ・エージェントによるタスクの実行

30.1.3.3.2 Windowsでのスケジューラ・エージェントの使用

Windowsのスケジューラ・エージェント・サービスは、インストール時に自動的に作成および起動されます。サービス名の末尾は、`OracleSchedulerExecutionAgent`になっています。

ノート:

Oracle Database がインストールされている Windows コンピュータ上で実行され、資格証明なしのローカル外部ジョブの実行を管理する `OracleJobScheduler` サービスと、このサービスを混同しないでください。

親トピック: スケジューラ・エージェントによるタスクの実行

30.1.3.3.3 スケジューラ・エージェントの起動

スケジューラ・エージェントを起動すると、それが存在するホストでのリモート・ジョブの実行が可能になります。

スケジューラ・エージェントを起動するには:

- 次のいずれかの操作を行います。
  - UNIXおよびLinuxの場合は、次のコマンドを実行します。

```
AGENT_HOME/bin/schagent -start
```

- Windowsの場合は、OracleSchedulerExecutionAgentで終わる名前のサービスを起動します。

親トピック: スケジューラ・エージェントによるタスクの実行

30.1.3.3.4 スケジューラ・エージェントの停止

スケジューラ・エージェントを停止すると、スケジューラ・エージェントがあるホストでリモート・ジョブを実行できなくなります。

スケジューラ・エージェントを停止するには:

- 次のいずれかの操作を行います。
  - UNIXおよびLinuxでは、表30-1に説明されている-stopまたは-abortオプションのどちらかを指定してschagentユーティリティを実行します。

    ```
    AGENT_HOME/bin/schagent -stop
    ```

  - Windowsでは、OracleSchedulerExecutionAgentで終わる名前のサービスを停止します。これは-abortオプションに相当します。

親トピック: スケジューラ・エージェントによるタスクの実行

30.1.3.3.5 スケジューラ・エージェントのデータベースへの登録

スケジューラ・エージェントの構成が終わるとすぐに、リモート・ジョブを実行する予定の1つ以上のデータベースでエージェントを登録できます。

後でログインして、追加のデータベースにエージェントを登録することもできます。

1. すでにログアウトしている場合は、次の方法で、スケジューラ・エージェントを実行しているホストにログインします。
   - Windowsの場合は、管理者としてログインします。
   - UNIXおよびLinuxの場合は、スケジューラ・エージェントをインストールしたユーザーとしてログインします。
2. スケジューラ・エージェントを登録するデータベースごとに、次のコマンドを使用します。
   - UNIXおよびLinuxの場合は、次のコマンドを実行します。

     ```
     AGENT_HOME/bin/schagent -registerdatabase db_host db_http(s)_port
     ```

   - Windowsの場合は、次のコマンドを実行します。

     ```
     AGENT_HOME/bin/schagent.exe -registerdatabase db_host db_http(s)_port
     ```

   ここで:

   - db_hostは、データベースがあるホストのホスト名またはIPアドレスです。Oracle Real Application Clusters環境では、任意のノードを指定できます。
   - db_http(s)_portは、データベースがリスニングするポート番号です(HTTP(S)接続用)。このパラメータは、「リモート・ジョブを実行するためのデータベースの設定の有効化と無効化」で事前に設定しました。次のSQL文をデータベースに発行することにより、ポート番号を確認できます。

     HTTP接続の場合:

     ```
     SELECT DBMS_XDB_CONFIG.GETHTTPPORT() FROM DUAL;
     ```

     HTTPS接続の場合:

     ```
     SELECT DBMS_XDB_CONFIG.GETHTTPSPORT() FROM DUAL;
     ```

ポート番号が0のときは、HTTP(S)接続が無効になっています。

「リモート・ジョブを実行するためのデータベースの設定の有効化と無効化」で設定したエージェント登録パスワードの入力を求めるプロンプトが表示されます。

Oracle Databaseリリース18c以降、エージェントは、ポートがHTTP接続またはHTTPS接続のどちらを使用するように構成されているかを自動的に判別します。HTTPS接続の場合、エージェントは、信頼できる証明書として追加する必要のある信頼されていない証明書の選択を求めるプロンプトも表示します。

3. エージェントのホストでリモート・ジョブを実行する追加データベースがある場合は、前のステップを繰り返します。

親トピック: スケジューラ・エージェントによるタスクの実行

30.2 スケジューラの監視と管理

現在アクティブなウィンドウとそれに関連付けられているリソース・プランの表示、現在実行中のジョブに関する情報の表示、ウィンドウおよびジョブのログの監視と管理、およびスケジューラのセキュリティ管理を実行できます。

- 現在アクティブなウィンドウとリソース・プランの表示
  DBA_SCHEDULER_WINDOWSビューの問合せにより、現在アクティブなウィンドウとそれに関連付けられたプランを表示できます。
- 現在実行中のジョブに関する情報の検索
  DBA_SCHEDULER_JOBSビューの問合せにより、ジョブの状態を確認できます。
- ウィンドウ・ログおよびジョブ・ログの監視と管理
  スケジューラは、ジョブ・ログとウィンドウ・ログという2種類のログをサポートしています。
- スケジューラのセキュリティの管理
  実行するスケジューラ操作に基づいて、適切な権限をユーザーに付与する必要があります。

親トピック: Oracle Schedulerの管理

30.2.1 現在アクティブなウィンドウとリソース・プランの表示

DBA_SCHEDULER_WINDOWSビューの問合せにより、現在アクティブなウィンドウとそれに関連付けられたプランを表示できます。

たとえば、次の文を発行します。

```
SELECT WINDOW_NAME, RESOURCE_PLAN FROM DBA_SCHEDULER_WINDOWS
WHERE ACTIVE='TRUE';
WINDOW_NAME                    RESOURCE_PLAN
------------------------------ --------------------------
MY_WINDOW10                    MY_RESOURCEPLAN1
```

アクティブなウィンドウがない場合は、次の文を発行するとアクティブなリソース・プランを表示できます。

```
SELECT * FROM V$RSRC_PLAN;
```

親トピック: スケジューラの監視と管理

30.2.2 現在実行中のジョブに関する情報の検索

DBA_SCHEDULER_JOBSビューの問合せにより、ジョブの状態を確認できます。

たとえば、次の文を発行します。

```
SELECT JOB_NAME, STATE FROM DBA_SCHEDULER_JOBS
```

```
WHERE JOB_NAME = 'MY_EMP_JOB1';
JOB_NAME                       STATE
------------------------------ ---------
MY_EMP_JOB1                    DISABLED
```

この場合は、ENABLEプロシージャを使用してジョブを使用可能にできます。表30-2に、ジョブの状態に関する有効値を示します。

表30-2 ジョブの状態

ジョブの状態	説明
disabled	ジョブは使用禁止です。
scheduled	ジョブは実行するようにスケジュールされています。
running	ジョブは現在実行中です。
completed	ジョブは完了しており、次回の実行はスケジュールされていません。
stopped	ジョブは 1 回実行するようにスケジュールされましたが、実行中に停止しました。
broken	ジョブは中断されました。
failed	ジョブは 1 回実行するようにスケジュールされ、失敗しました。
retry scheduled	ジョブは 1 回以上失敗し、再試行の実行がスケジュールされました。
succeeded	ジョブは 1 回実行するようにスケジュールされ、正常に完了しました。
chain_stalled	ジョブのタイプがチェーンで、実行されているステップ、実行がスケジュールされているステップおよびイベント待ちのイベント・ステップがなく、チェーンの evaluation_interval が NULL に設定されています。手動で操作を行わないかぎり、チェーンは続行されません。

現在実行中のジョブの進行状況を確認するには、次の文を発行します。

```
SELECT * FROM ALL_SCHEDULER_RUNNING_JOBS;
```

CPU_USED列に有効なデータを表示するには、RESOURCE_LIMIT初期化パラメータをtrueに設定する必要があります。

リモートおよびローカルのすべての宛先ですべてのジョブの状態を確認するには、次の文を発行します。

```
SELECT * FROM DBA_SCHEDULER_JOB_DESTS;
```

実行中のチェーンの一部のジョブに関する情報を得るには、次の文を発行します。

```
SELECT * FROM ALL_SCHEDULER_RUNNING_CHAINS WHERE JOB_NAME='MY_JOB1';
```

cjqNNNの形式のプロセスを検索すると、ジョブ・コーディネータが実行中かどうかをチェックできます。

関連項目:

- 「複数の宛先のジョブ」
- *_SCHEDULER_RUNNING_JOBSビューの詳細は、『Oracle Databaseリファレンス』を参照してください
- *_SCHEDULER_JOBSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: スケジューラの監視と管理

30.2.3 ウィンドウ・ログおよびジョブ・ログの監視と管理

スケジューラは、ジョブ・ログとウィンドウ・ログという2種類のログをサポートしています。

- ジョブ・ログ
  ジョブの実行、状態変化および失敗について、ジョブ・ログ内の情報を表示できます。
- ウィンドウ・ログ
  ウィンドウ・ログは、ウィンドウでの操作を記録します。
- ログのパージ
  ジョブ・ログおよびウィンドウ・ログが無計画に大きくならないように、SET_SCHEDULER_ATTRIBUTEプロシージャを使用して、保持する履歴の量(日数)を指定します。

親トピック: スケジューラの監視と管理

30.2.3.1 ジョブ・ログ

ジョブの実行、状態変化および失敗について、ジョブ・ログ内の情報を表示できます。

ジョブ・ログは、次の2つのデータ・ディクショナリ・ビューとして実装されます。

- *_SCHEDULER_JOB_LOG
- *_SCHEDULER_JOB_RUN_DETAILS

スケジューラがジョブに関して実行するロギングの量は、ジョブ・クラス・レベルと個別ジョブ・レベルの両方で制御できます。通常、クラス内のジョブに関するロギングを厳密に制御できるため、ロギングをクラス・レベルで制御します。

各種ロギング・レベルの定義と、ジョブとジョブ・クラスの間のロギング・レベルの優先度については、「ジョブ・ログの表示」を参照してください。デフォルトでは、ジョブ・クラスのロギング・レベルはLOGGING_RUNSで、すべてのジョブの実行がログに記録されます。

ジョブ・クラスを作成する際にlogging_level属性を設定するか、SET_ATTRIBUTEプロシージャを使用すると、後でロギング・レベルを変更できます。次の例では、myclass1ジョブ・クラスのジョブのロギング・レベルを、失敗した実行のみがログに記録されるLOGGING_FAILED_RUNSに設定しています。すべてのジョブ・クラスはSYSスキーマにあることに注意してください。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
   'sys.myclass1', 'logging_level', DBMS_SCHEDULER.LOGGING_FAILED_RUNS);
END;
/
```

ジョブ・クラスのロギング・レベルを設定するには、MANAGE SCHEDULER権限が付与されている必要があります。

関連項目:

- ジョブ・ログの詳細と、ジョブ・ログ・ビューに対する問合せ例については、「ジョブ・ログの表示」

- *_SCHEDULER_JOB_LOGビューの詳細は、『Oracle Databaseリファレンス』を参照してください
- *_SCHEDULER_JOB_RUN_DETAILSビューの詳細は、『Oracle Databaseリファレンス』を参照してください
- CREATE_JOB_CLASSおよびSET_ATTRIBUTEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- ログ・エントリの保存期間の設定については、「スケジューラのプリファレンスの設定」

親トピック: ウィンドウ・ログおよびジョブ・ログの監視と管理

30.2.3.2 ウィンドウ・ログ

ウィンドウ・ログは、ウィンドウでの操作を記録します。

スケジューラでは、次の操作が実行されるたびにウィンドウ・ログにエントリが1つ記録されます。

- ウィンドウの作成または削除
- ウィンドウのオープン
- ウィンドウのクローズ
- ウィンドウの重複
- ウィンドウの使用可能化または使用禁止

ウィンドウ・アクティビティのロギングには、ロギング・レベルはありません。

ウィンドウ・ログの内容を調べるには、DBA_SCHEDULER_WINDOW_LOGビューを問い合せます。次の文は、このビューの出力例を示しています。

```
SELECT log_id, to_char(log_date, 'DD-MON-YY HH24:MI:SS') timestamp,
   window_name, operation FROM DBA_SCHEDULER_WINDOW_LOG;
    LOG_ID TIMESTAMP            WINDOW_NAME       OPERATION
---------- -------------------- ----------------- --------
         4 10/01/2004 15:29:23  WEEKEND_WINDOW    CREATE
         5 10/01/2004 15:33:01  WEEKEND_WINDOW    UPDATE
        22 10/06/2004 22:02:48  WEEKNIGHT_WINDOW  OPEN
        25 10/07/2004 06:59:37  WEEKNIGHT_WINDOW  CLOSE
        26 10/07/2004 22:01:37  WEEKNIGHT_WINDOW  OPEN
        29 10/08/2004 06:59:51  WEEKNIGHT_WINDOW  CLOSE
```

DBA_SCHEDULER_WINDOWS_DETAILSビューは、以前はアクティブで現在はクローズ(完了)されているすべてのウィンドウに関する情報を提供します。次の文は、このビューの出力例を示しています。

```
SELECT LOG_ID, WINDOW_NAME, ACTUAL_START_DATE, ACTUAL_DURATION
  FROM DBA_SCHEDULER_WINDOW_DETAILS;
    LOG_ID WINDOW_NAME      ACTUAL_START_DATE                    ACTUAL_DURATION
---------- ---------------- ------------------------------------ ---------------
        25 WEEKNIGHT_WINDOW 06-OCT-04 10:02.48.832438 PM PST8PDT +000 01:02:32
        29 WEEKNIGHT_WINDOW 07-OCT-04 10.01.37.025704 PM PST8PDT +000 03:02:00
```

これらのビューの両方でログIDが対応しており、この例では、DBA_SCHEDULER_WINDOWS_DETAILSビューの行がDBA_SCHEDULER_WINDOW_LOGビューのCLOSE操作に対応しています。

関連項目:

- *_SCHEDULER_WINDOW_LOGビューの詳細は、『Oracle Databaseリファレンス』を参照してください

- DBA_SCHEDULER_WINDOWS_DETAILSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: ウィンドウ・ログおよびジョブ・ログの監視と管理

30.2.3.3 ログのパージ

ジョブ・ログおよびウィンドウ・ログが無計画に大きくならないように、SET_SCHEDULER_ATTRIBUTEプロシージャを使用して、保持する履歴の量(日数)を指定します。

1日に1回、スケジューラは、指定した履歴期間より古いすべてのログ・エントリをジョブ・ログとウィンドウ・ログから自動的にパージします。デフォルトの履歴期間は30日です。たとえば、履歴期間を90日に変更する場合は、次の文を発行します。

```
DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE('log_history','90');
```

一部のジョブ・クラスは他より重要です。そのため、クラス固有の設定を使用して、このグローバルな履歴設定をオーバーライドできます。たとえば、3つのジョブ・クラス(class1、class2およびclass3)があり、class1とclass3に対しては10日間のウィンドウ・ログの履歴を保存し、class2に対しては30日間保存するとします。これを実現するには、次の文を発行します。

```
DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE('log_history','10');
DBMS_SCHEDULER.SET_ATTRIBUTE('class2','log_history','30');
```

ジョブ・クラスの作成時には、クラス別の履歴を設定することもできます。

チェーン実行のステップに関連したログ・エントリは、メインのチェーン・ジョブのエントリがパージされるまでパージされません。

手動でのログのパージ

PURGE_LOGプロシージャを使用して、ログを手動でパージできます。例として、ジョブ・ログとウィンドウ・ログの両方からすべてのエントリをパージする文を示します。

```
DBMS_SCHEDULER.PURGE_LOG();
```

別の例として、3日前より古いすべてのエントリをジョブ・ログからパージする文を示します。ウィンドウ・ログはこの文の影響を受けません。

```
DBMS_SCHEDULER.PURGE_LOG(log_history => 3, which_log => 'JOB_LOG');
```

次の文では、10日前より古いすべてのウィンドウ・ログ・エントリと、job1およびclass2内のジョブに関連する10日前より古いすべてのジョブ・ログ・エントリがパージされます。

```
DBMS_SCHEDULER.PURGE_LOG(log_history => 10, job_name => 'job1, sys.class2');
```

親トピック: ウィンドウ・ログおよびジョブ・ログの監視と管理

30.2.4 スケジューラ・セキュリティの管理

実行するスケジューラ操作に基づいて、適切な権限をユーザーに付与する必要があります。

ジョブのスケジュールと実行にスケジューラを使用する必要がある通常のユーザーに対しては、CREATE JOBシステム権限を付与してください。システム・リソースを管理する必要があるデータベース管理者には、MANAGE SCHEDULERを付与してください。スケジューラに関するその他のシステム権限またはロールを付与するときは、十分に注意してください。特に、CREATE ANY JOBシステム権限と、この権限が含まれるSCHEDULER_ADMINロールは、コードを任意のユーザーで実行できるため、非常に強力です。これらの権限とロールの付与は、非常に強力な権限を持つロールまたはユーザーに限定してください。

外部ジョブの処理は、セキュリティに関して特に重要な問題です。データベース外でジョブを実行する必要があるユーザーにのみ、これを可能にするCREATE EXTERNAL JOBシステム権限を付与してください。スケジューラに関するセキュリティには、他に特

別な要件はありません。セキュリティの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

ユーザーがオペレーティング・システムまたはリモート・データベースにジョブを認証する資格証明を作成する必要がある場合は、CREATE CREDENTIALシステム権限を付与します。

ノート:

Oracle Database 10g リリース 1 (10.1)から Oracle Database 10g リリース 2 (10.2)以降にアップグレードすると、CREATE EXTERNAL JOB が CREATE JOB 権限を持つすべてのユーザーとロールに対して自動的に付与されます。必要のないユーザーに対してはこの権限を取り消すことをお薦めします。

親トピック: スケジューラの監視と管理

30.3 スケジューラのインポート/エクスポート

スケジューラ・オブジェクトをエクスポートするには、データ・ポンプ・ユーティリティ(impdpおよびexpdp)を使用します。

スケジューラでは、以前のインポートおよびエクスポート・ユーティリティ(IMPおよびEXP)は使用できません。また、スケジューラ・オブジェクトは、データベースが読取り専用モードの間はエクスポートできません。

エクスポートでは、スケジューラ・オブジェクトの作成に使用されたDDLが生成されます。すべての属性がエクスポートされます。インポートが実行されると、すべてのデータベース・オブジェクトが新規データベースに再作成されます。すべてのスケジュールは、それぞれのタイム・ゾーンで格納され、新規データベース内で保持されます。たとえば、「サンフランシスコにあるデータベースの月曜日午後1時(太平洋沿岸標準時)」というスケジュールは、エクスポートされドイツでデータベースにインポートされた場合でも同じです。

スケジューラの資格証明はエクスポートされますが、セキュリティ上の理由で、これらの資格証明のパスワードはエクスポートされません。スケジューラの資格証明をインポートした後は、DBMS_SCHEDULERパッケージのSET_ATTRIBUTEプロシージャを使用してパスワードを再設定する必要があります。

関連項目:

データ・ポンプの詳細は、『Oracle Databaseユーティリティ』を参照してください。

親トピック: Oracle Schedulerの管理

30.4 スケジューラのトラブルシューティング

スケジューラでの問題をトラブルシューティングできます。

- ジョブが実行されない
  ジョブはいくつかの理由で実行に失敗する場合があります。
- プログラムが無効化される
  プログラムは、プログラム引数が削除された場合やnumber_of_argumentsの変更ですべての引数が定義されなくなった場合に無効化される場合があります。
- ウィンドウの有効化に失敗する
  ウィンドウは様々な理由で有効化されない場合があります。

親トピック: Oracle Schedulerの管理

30.4.1 ジョブが実行されない

ジョブはいくつかの理由で実行に失敗する場合があります。

実行されなかった可能性のあるジョブのトラブルシューティングを開始するには、次の文を発行して、ジョブの状態を確認します。

```
SELECT JOB_NAME, STATE FROM DBA_SCHEDULER_JOBS;
```

標準的な出力は次のようになります。

```
JOB_NAME                       STATE
------------------------------ ---------
MY_EMP_JOB                     DISABLED
MY_EMP_JOB1                    FAILED
MY_NEW_JOB1                    DISABLED
MY_NEW_JOB2                    BROKEN
MY_NEW_JOB3                    COMPLETED
```

- ジョブの状態について
  ジョブが実行されない場合は、次のいずれかの状態である可能性があります: 失敗、中断、無効または完了。
- ジョブ・ログの表示
  ジョブ・ログは、重要なトラブルシューティング・ツールです。
- リモート・ジョブのトラブルシューティング
  リモート・ジョブは、リモート・ホスト上のスケジューラ・エージェントと正常に通信する必要があります。リモート・ジョブが実行されない場合は、最初にDBA_SCHEDULER_JOBSビューとジョブ・ログを確認してください。
- 障害後のジョブ・リカバリについて
  スケジューラは、中断されたジョブのリカバリを試行できます。

親トピック: スケジューラのトラブルシューティング

30.4.1.1 ジョブの状態について

ジョブが実行されない場合は、次のいずれかの状態である可能性があります: 失敗、破損、無効または完了。

- 失敗したジョブ
  ジョブ表のジョブのステータスがFAILEDの場合、そのジョブは実行のためにスケジュールされていたが実行に失敗したことを示しています。ジョブが再起動可能として指定されていた場合は、すべての再試行に失敗しています。
- 中断されたジョブ
  中断されたジョブは、特定の失敗数を超えたジョブです。この数はmax_failuresで設定され、変更できます。
- 使用禁止のジョブ
  ジョブは、いくつかの理由で無効になることがあります。
- 完了したジョブ
  ジョブは、end_dateまたはmax_runsに達すると完了します。

親トピック: ジョブが実行されない

30.4.1.1.1 失敗したジョブ

ジョブ表のジョブのステータスがFAILEDの場合、そのジョブは実行のためにスケジュールされていたが実行に失敗したことを示しています。ジョブが再起動可能として指定されていた場合は、すべての再試行に失敗しています。

実行の途中でジョブが失敗すると、最後のトランザクションのみがロールバックされます。複数のトランザクションを実行するジョブの場合は、restartableをTRUEに設定するときに注意する必要があります。失敗したジョブを問い合せるには、*_SCHEDULER_JOB_RUN_DETAILSビューを問い合せます。

親トピック: ジョブの状態について

30.4.1.1.2 中断されたジョブ

中断されたジョブとは、特定の失敗数を超えたジョブです。この数はmax_failuresで設定され、変更できます。

中断されたジョブは、ジョブ全体が中断されており、修正されるまで実行されません。デバッグとテストには、RUN_JOBプロシージャを使用できます。

中断されたジョブを問い合せるには、*_SCHEDULER_JOBSおよび*_SCHEDULER_JOB_LOGビューを問い合せます。

親トピック: ジョブの状態について

30.4.1.1.3 使用禁止のジョブ

ジョブは、いくつかの理由で無効になることがあります。

次のような理由があります:

- 手動で使用禁止にされた場合
- ジョブが属しているジョブ・クラスが削除された場合
- ジョブが指し示しているプログラム、チェーンまたはスケジュールが削除された場合
- ウィンドウまたはウィンドウ・グループがジョブのスケジュールとなっているときに、そのウィンドウまたはウィンドウ・グループが削除された場合

親トピック: ジョブの状態について

30.4.1.1.4 完了したジョブ

ジョブは、end_dateまたはmax_runsに達すると完了します。

最近正常に完了したジョブが再度実行するようにスケジュールされている場合、ジョブの状態はSCHEDULEDになります。

親トピック: ジョブの状態について

30.4.1.2 ジョブ・ログの表示

ジョブ・ログは、重要なトラブルシューティング・ツールです。

詳細および手順は、「ジョブ・ログの表示」を参照してください。

親トピック: ジョブが実行されない

30.4.1.3 リモート・ジョブのトラブルシューティング

リモート・ジョブは、リモート・ホスト上のスケジューラ・エージェントと正常にやり取りする必要があります。リモート・ジョブが実行されない場合は、最初にDBA_SCHEDULER_JOBSビューとジョブ・ログを確認してください。

次に、次の手順を実行します。

1. nslookupやpingなどのツールを使用して、リモート・システムがネットワーク上で到達可能であることを確認します。
2. GET_AGENT_VERSIONパッケージ・プロシージャをコールして、リモート・ホストでスケジューラ・エージェントの状態を確認します。

```
DECLARE
  versionnum VARCHAR2(30);
BEGIN
  versionnum := DBMS_SCHEDULER.GET_AGENT_VERSION('remote_host.example.com');
```

```
  DBMS_OUTPUT.PUT_LINE(versionnum);
END;
/
```

エラーが生成された場合、エージェントがインストールされていないか、ローカル・データベースで登録されていない可能性があります。スケジューラ・エージェントのインストール、登録および起動手順は、「Oracle Schedulerエージェントを使用したリモート・ジョブの実行」を参照してください。

親トピック: ジョブが実行されない

30.4.1.4 障害後のジョブ・リカバリについて

スケジューラは、中断されたジョブのリカバリを試行できます。

スケジューラは、次の場合に中断されたジョブのリカバリを試行します。

- データベースが異常停止した場合
- ジョブ・スレーブ・プロセスが終了したか、失敗した場合
- 外部ジョブの場合に、実行可能ファイルまたはスクリプトを起動する外部ジョブ・プロセスが終了したか、失敗した場合。(UNIXでは、外部ジョブ・プロセスは`extjob`です。Windowsでは、外部ジョブ・サービスです。)
- 外部ジョブの場合に、エンド・ユーザーの実行可能ファイルまたはスクリプトを実行するプロセスが終了したか、失敗した場合。

ジョブ・リカバリは次のように実行されます。

- スケジューラにより、障害が発生したときに実行されていたジョブのインスタンスについて、ジョブ・ログにエントリが追加されます。ログ・エントリでは、`OPERATION`は`RUN`、`STATUS`は`STOPPED`になっており、`ADDITIONAL_INFO`には次のいずれかが挿入されます。
  - `REASON=ジョブ・スレーブ・プロセスは終了しました`
  - `REASON=ORA-01014: Oracleのシャットダウン処理中です。`
- そのジョブの`restartable`が`TRUE`に設定されている場合は、ジョブが再開されます。
- `restartable`が`FALSE`に設定されている場合は、次のようになります。
  - ジョブが1回のみ実行されるジョブであり、`auto_drop`が`TRUE`に設定されている場合は、ジョブの実行が完了するとそのジョブは削除されます。
  - ジョブが1回のみ実行されるジョブであり、`auto_drop`が`FALSE`に設定されている場合は、ジョブが使用禁止になり、そのジョブの状態`state`は`STOPPED`に設定されます。
  - ジョブが繰返しジョブの場合、スケジューラは次回のジョブの実行をスケジュールし、そのジョブの状態`state`は`SCHEDULED`に設定されます。

このリカバリ・プロセスの結果、ジョブが再開されると、新規実行は`RECOVERY_RUN`操作としてジョブ・ログに記録されます。

親トピック: ジョブが実行されない

30.4.2 プログラムが無効化される

プログラムは、プログラム引数が削除された場合や`number_of_arguments`の変更ですべての引数を定義できなくなった場合に使用禁止になります。

プログラムの詳細は、「ジョブを定義するためのプログラムの作成および管理」を参照してください。

親トピック: スケジューラのトラブルシューティング

30.4.3 ウィンドウの有効化に失敗する

ウィンドウは様々な理由で有効化されない場合があります。

ウィンドウは、次の場合に有効化できません。

- スケジュールの終了時にウィンドウが使用禁止にされた場合
- すでに存在しないスケジュールを指すウィンドウが使用禁止にされた場合

ウィンドウの詳細は、「ウィンドウを使用したジョブ・スケジューリングとジョブ優先度の管理」を参照してください。

親トピック: スケジューラのトラブルシューティング

30.5 スケジューラの使用例

スケジューラの使用例を示します。

- ジョブ・クラスの作成例
  ジョブ・クラスの作成例を示します。
- 属性の設定例
  属性の設定例を示します。
- チェーンの作成例
  チェーンの作成例を示します。
- イベントに基づくジョブとスケジュールの作成例
  イベントに基づくジョブとイベント・スケジュールの作成例を示します。
- Oracle Data Guard環境でのジョブの作成例
  Oracle Data Guard環境では、スケジューラには、2つのデータベース・ロール(プライマリとロジカル・スタンバイ)に対する追加のサポートが含まれています。データベースのロールがプライマリの場合のみ、またはデータベースのロールがロジカル・スタンバイの場合のみ、ジョブが実行されるように構成できます。

親トピック: Oracle Schedulerの管理

30.5.1 ジョブ・クラスの作成例

ジョブ・クラスの作成例を示します。

ジョブ・クラスを作成するには、CREATE_JOB_CLASSプロシージャを使用します。

例30-1 ジョブ・クラスの作成

次の文ではジョブ・クラスが作成されます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB_CLASS (
   job_class_name              =>  'my_class1',
   service                     =>  'my_service1',
   comments                    =>  'This is my first job class');
END;
/
```

これにより、my_class1がSYSに作成されます。ここでは、my_service1というサービスが使用されています。ジョブ・クラスが作成されたことを確認するには、次の文を発行します。

```
SELECT JOB_CLASS_NAME FROM DBA_SCHEDULER_JOB_CLASSES
WHERE JOB_CLASS_NAME = 'MY_CLASS1';
JOB_CLASS_NAME
------------------------------
MY_CLASS1
```

例30-2 ジョブ・クラスの作成

次の文ではジョブ・クラスが作成されます。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB_CLASS (
   job_class_name             =>  'finance_jobs',
   resource_consumer_group    =>  'finance_group',
   service                    =>  'accounting',
   comments                   =>  'All finance jobs');
END;
/
```

これにより、finance_jobsがSYSに作成されます。finance_groupというリソース・コンシューマ・グループが割り当てられ、accountingサービス用のサービス・アフィニティが指定されています。accountingサービスがfinance_group以外のリソース・コンシューマ・グループにマップされている場合は、resource_consumer_group属性が優先されるので、このクラスのジョブはfinance_groupコンシューマ・グループで実行されることに注意してください。

関連項目:

CREATE_JOB_CLASSプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。詳細は、「ジョブ・クラスの作成」を参照してください

親トピック: スケジューラの使用例

30.5.2 属性の設定例

属性の設定例を示します。

属性を設定するには、SET_ATTRIBUTEおよびSET_SCHEDULER_ATTRIBUTEプロシージャを使用します。

例30-3 繰返し間隔属性の設定

次の例では、my_emp_job1が日次で実行されるように頻度が再設定されます。

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
   name           =>   'my_emp_job1',
   attribute      =>   'repeat_interval',
   value          =>   'FREQ=DAILY');
END;
/
```

変更されたことを確認するには、次の文を発行します。

```
SELECT JOB_NAME, REPEAT_INTERVAL FROM DBA_SCHEDULER_JOBS
WHERE JOB_NAME =  'MY_EMP_JOB1';
JOB_NAME             REPEAT_INTERVAL
----------------     ---------------
MY_EMP_JOB1          FREQ=DAILY
```

例30-4 一連のジョブに対する複数のジョブ属性の設定

次の例では、5つの各ジョブに4つの異なる属性が設定されます。

```
DECLARE
 newattr sys.jobattr;
 newattrarr sys.jobattr_array;
 j number;
BEGIN
 -- Create new JOBATTR array
 newattrarr := sys.jobattr_array();
 -- Allocate enough space in the array
 newattrarr.extend(20);
 j := 1;
 FOR i IN 1..5 LOOP
   -- Create and initialize a JOBATTR object type
   newattr := sys.jobattr(job_name => 'TESTJOB' || to_char(i),
                          attr_name => 'MAX_FAILURES',
                          attr_value => 5);
   -- Add it to the array.
   newattrarr(j) := newattr;
   j := j + 1;
   newattr := sys.jobattr(job_name => 'TESTJOB' || to_char(i),
                          attr_name => 'COMMENTS',
                          attr_value => 'Test job');
   newattrarr(j) := newattr;
   j := j + 1;
   newattr := sys.jobattr(job_name => 'TESTJOB' || to_char(i),
                          attr_name => 'END_DATE',
                          attr_value => systimestamp + interval '24' hour);
   newattrarr(j) := newattr;
   j := j + 1;
   newattr := sys.jobattr(job_name => 'TESTJOB' || to_char(i),
                          attr_name => 'SCHEDULE_LIMIT',
                          attr_value => interval '1' hour);
   newattrarr(j) := newattr;
   j := j + 1;
 END LOOP;
 -- Call SET_JOB_ATTRIBUTES to set all 20 set attributes in one transaction
 DBMS_SCHEDULER.SET_JOB_ATTRIBUTES(newattrarr, 'TRANSACTIONAL');
END;
/
```

関連項目:

SET_SCHEDULER_ATTRIBUTEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』および「スケジューラのプリファレンスの設定」

親トピック: スケジューラの使用例

30.5.3 チェーンの作成例

チェーンの作成例を示します。

チェーンを作成するには、CREATE_CHAINプロシージャを使用します。チェーンを作成した後、DEFINE_CHAIN_STEPまたはDEFINE_CHAIN_EVENT_STEPプロシージャを使用してチェーンにステップを追加し、DEFINE_CHAIN_RULEプロシージャを使用してルールを定義します。

例30-5 チェーンの作成

次の例では、my_program2およびmy_program3の前にmy_program1を実行するチェーンが作成されます。

my_program1の完了後、my_program2およびmy_program3は、並行して実行されます。

この例のユーザーには、CREATE EVALUATION CONTEXT、CREATE RULEおよびCREATE RULE SET権限が必要です。詳細は、「チェーンの各権限の設定」を参照してください。

```
BEGIN
  DBMS_SCHEDULER.CREATE_CHAIN (
   chain_name            =>  'my_chain1',
   rule_set_name         =>  NULL,
   evaluation_interval   =>  NULL,
   comments              =>  NULL);
END;
/

--- define three steps for this chain. Referenced programs must be enabled.
BEGIN
 DBMS_SCHEDULER.DEFINE_CHAIN_STEP('my_chain1', 'stepA', 'my_program1');
 DBMS_SCHEDULER.DEFINE_CHAIN_STEP('my_chain1', 'stepB', 'my_program2');
 DBMS_SCHEDULER.DEFINE_CHAIN_STEP('my_chain1', 'stepC', 'my_program3');
END;
/
--- define corresponding rules for the chain.
BEGIN
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE('my_chain1', 'TRUE', 'START stepA');
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   'my_chain1', 'stepA COMPLETED', 'Start stepB, stepC');
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   'my_chain1', 'stepB COMPLETED AND stepC COMPLETED', 'END');
END;
/
--- enable the chain
BEGIN
 DBMS_SCHEDULER.ENABLE('my_chain1');
END;
/
--- create a chain job to start the chain daily at 1:00 p.m.
BEGIN
 DBMS_SCHEDULER.CREATE_JOB (
   job_name        => 'chain_job_1',
   job_type        => 'CHAIN',
   job_action      => 'my_chain1',
   repeat_interval => 'freq=daily;byhour=13;byminute=0;bysecond=0',
   enabled         => TRUE);
END;
/
```

例30-6 チェーンの作成

次の例では、最初にmy_program1を実行してチェーンを作成します。正常に実行された場合はmy_program2、失敗した場合はmy_program3が実行されます。

```
BEGIN
 DBMS_SCHEDULER.CREATE_CHAIN (
   chain_name            => 'my_chain2',
   rule_set_name         => NULL,
   evaluation_interval   => NULL,
   comments              => NULL);
END;
/

--- define three steps for this chain.
BEGIN
 DBMS_SCHEDULER.DEFINE_CHAIN_STEP('my_chain2', 'step1', 'my_program1');
 DBMS_SCHEDULER.DEFINE_CHAIN_STEP('my_chain2', 'step2', 'my_program2');
 DBMS_SCHEDULER.DEFINE_CHAIN_STEP('my_chain2', 'step3', 'my_program3');
```

```
END;
/

--- define corresponding rules for the chain.
BEGIN
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE ('my_chain2', 'TRUE', 'START step1');
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   'my_chain2', 'step1 SUCCEEDED', 'Start step2');
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   'my_chain2', 'step1 COMPLETED AND step1 NOT SUCCEEDED', 'Start step3');
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   'my_chain2', 'step2 COMPLETED OR step3 COMPLETED', 'END');
END;
/
```

関連項目:

CREATE_CHAIN、DEFINE_CHAIN_STEPおよびDEFINE_CHAIN_RULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』および「スケジューラのプリファレンスの設定」

親トピック: スケジューラの使用例

30.5.4 イベントに基づくジョブとスケジュールの作成例

イベント・ベースのジョブおよびイベント・スケジュールの作成例を示します。

イベント・ベースのジョブを作成するには、CREATE_JOBプロシージャを使用します。イベント・ベースのスケジュールを作成するには、CREATE_EVENT_SCHEDULEプロシージャを使用します。

これらの例では、アプリケーションでシステムにファイルが到着したことが検出されたときに、イベントをmy_events_qキューにエンキューするアプリケーションがあることを想定しています。

例30-7 イベント・ベースのスケジュールの作成

次の例では、午前9時前にファイルがシステムに到着したことを示すイベントをスケジューラが受信するたびに、ジョブの開始に使用可能なスケジュールを作成する方法を示しています。

```
BEGIN
  DBMS_SCHEDULER.CREATE_EVENT_SCHEDULE (
   schedule_name     =>  'scott.file_arrival',
   start_date        =>  systimestamp,
   event_condition   =>  'tab.user_data.object_owner = ''SCOTT''
      and tab.user_data.event_name = ''FILE_ARRIVAL''
      and extract hour from tab.user_data.event_timestamp < 9',
   queue_spec        =>  'my_events_q');
END;
/
```

例30-8 イベント・ベースのジョブの作成

次の例では、ファイルがシステムに到着したことを示すイベントをスケジューラが受信したときに開始されるジョブを作成しています。

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
   job_name            =>  my_job,
   program_name        =>  my_program,
   start_date          =>  '15-JUL-04 1.00.00AM US/Pacific',
   event_condition     =>  'tab.user_data.event_name = ''LOW_INVENTORY''',
   queue_spec          =>  'my_events_q'
   enabled             =>  TRUE,
```

```
   comments            =>  'my event-based job');
END;
/
```

関連項目:

CREATE_JOBおよびCREATE_EVENT_SCHEDULEプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: スケジューラの使用例

30.5.5 Oracle Data Guard環境でのジョブの作成例

Oracle Data Guard環境では、スケジューラには、2つのデータベース・ロール(プライマリとロジカル・スタンバイ)に対する追加のサポートが含まれています。データベースのロールがプライマリの場合のみ、またはデータベースのロールがロジカル・スタンバイの場合のみ、ジョブが実行されるように構成できます。

そのためには、database_role属性を設定します。この例では、両方のデータベース・ロールでジョブを実行できるようにする方法を説明します。ジョブのコピーを2つ作成し、それぞれに異なるdatabase_roleを割り当てる方法を使用します。

デフォルトでは、ジョブが実行されるのは、データベースのロールがそのジョブの作成時と同じロールの場合です。両方のロールで同じジョブを実行するステップは、次のとおりです。

1. ジョブをコピーします。
2. 新しいジョブを使用可能にします。
3. 新しいジョブのdatabase_role属性を必要なロールに変更します。

この例では、まずprimary_jobというジョブをプライマリ・データベースに作成しています。次に、このジョブのコピーを作成して、そのdatabase_role属性を「LOGICAL STANDBY」に設定しています。プライマリ・データベースがロジカル・スタンバイになった場合は、そのスケジュールに従ってジョブが引き続き実行されます。

ジョブをコピーした場合、新しいジョブは使用禁止になっているため、使用可能にする必要があります。

```
BEGIN DBMS_SCHEDULER.CREATE_JOB (
     job_name        => 'primary_job',
     program_name    => 'my_prog',
     schedule_name   => 'my_sched');
 DBMS_SCHEDULER.COPY_JOB('primary_job','standby_job');
 DBMS_SCHEDULER.ENABLE(name=>'standby_job', commit_semantics=>'ABSORB_ERRORS');
 DBMS_SCHEDULER.SET_ATTRIBUTE('standby_job','database_role','LOGICAL STANDBY');
END;
/
```

この例を実行すると、DBA_SCHEDULER_JOB_ROLESビューのデータは、次のようになります。

```
SELECT JOB_NAME, DATABASE_ROLE FROM DBA_SCHEDULER_JOB_ROLES
   WHERE JOB_NAME IN ('PRIMARY_JOB','STANDBY_JOB');
JOB_NAME                DATABASE_ROLE
--------                ----------------
PRIMARY_JOB             PRIMARY
STABDBY_JOB             LOGICAL STANDBY
```

ノート:

フィジカル・スタンバイ・データベースの場合、スケジューラのオブジェクトに対する変更またはプライマリ・データベース上のスケジューラのジョブによるデータベースの変更は、他のデータベースの変更と同様にフィジカル・スタンバイに適用されます。

親トピック: スケジューラの使用例

30.6 スケジューラの参照情報

スケジューラに関連するいくつかの権限およびデータ・ディクショナリ・ビューがあります。

- スケジューラ権限
  様々なスケジューラ権限をユーザーに付与することができます。
- スケジューラのデータ・ディクショナリ・ビュー
  スケジューラに関する情報について一連のビューを問い合せることができます。

親トピック: Oracle Schedulerの管理

30.6.1 スケジューラ権限

様々なスケジューラ権限を付与できます。

表30-3および表30-4は、様々なスケジューラ権限を示しています。

表30-3 スケジューラのシステム権限

権限名	許可される操作
CREATE JOB	自分のスキーマ内にジョブ、チェーン、スケジュール、プログラム、File Watcher、宛先およびグループを作成できます。CREATE JOB 権限がなくても、自分のスキーマ内でのこれらのオブジェクトの変更および削除はいつでも可能です。この場合、CREATE ANY JOB 権限を持つ別のユーザーによって、スキーマ内にそのオブジェクトが作成されます。
CREATE ANY JOB	SYS 以外の任意のスキーマ内でジョブ、チェーン、スケジュール、プログラム、File Watcher、宛先およびグループを作成、変更および削除できます。これは非常に強力な権限で、権限受領者は任意の他のデータベース・ユーザーで PL/SQL コードを実行できるため、慎重に使用してください。
CREATE EXTERNAL JOB	この権限はデータベース外で実行するジョブを作成する場合に必要です。EXECUTABLE タイプのジョブの所有者、または EXECUTABLE タイプのプログラムを指し示すジョブの所有者にはこの権限が必要です。EXECUTABLE タイプのジョブを実行する場合は、この権限および CREATE JOB 権限が必要です。リモート・ホストからファイルを取得したり、1 つ以上のリモート・ホストにファイルを保存する場合も、この権限が必要です。
EXECUTE ANY PROGRAM	自分のジョブに任意のスキーマのプログラムまたはチェーンを使用できます。

権限名	許可される操作
EXECUTE ANY CLASS	自分のジョブを任意のジョブ・クラスの下で実行できます。
MANAGE SCHEDULER	これはスケジューラを管理するための最も重要な権限です。ジョブ・クラス、ウィンドウおよびウィンドウ・グループの作成、変更および削除、force オプションによるジョブの停止を行えます。スケジューラ属性の設定と取得、スケジューラ・ログのパージ、データベースのエージェント・パスワードの設定も行えます。

表30-4 スケジューラのオブジェクト権限

権限名	許可される操作
SELECT	SELECT をグループに付与することによって、グループのオブジェクト権限を他のユーザーに付与できます。
EXECUTE	この権限はプログラム、チェーン、File Watcher、資格証明およびジョブ・クラスのみに付与できます。EXECUTE 権限では、ジョブのオブジェクトを参照できます。オブジェクトが自分のスキーマで作成されたのではない場合に、オブジェクトを表示することもできます。
ALTER	権限付与されているオブジェクトを変更または削除できます。変更操作には、プログラム引数の使用可能化、使用禁止化、定義または削除、ジョブ引数値の設定または再設定、およびジョブの実行などの操作が含まれます。ジョブ・タイプ EXECUTABLE のジョブの制限付きの特定の属性は、ALTER オブジェクト権限を使用して変更できません。これには、job_type、job_action、number_of_arguments、event_spec、スケジュールとしての PL/SQL 日付関数の設定が含まれます。 プログラム、ジョブ、チェーン、File Watcher および資格証明の場合には、この権限により、これらのオブジェクトを所有しないスキーマを使用可能にして表示することもできます。この権限は、ジョブ、チェーン、プログラム、スケジュール、File Watcher および資格証明に対してのみ付与できます。他のタイプのスケジューラ・オブジェクトに対しては、MANAGE SCHEDULER システム権限を付与する必要があります。
ALL	この権限では、指定したオブジェクトに対して他のすべてのオブジェクト権限で許可されている操作が可能です。これは、ジョブ、プログラム、チェーン、スケジュール、File Watcher、資格証明およびジョブ・クラスに付与できます。

ノート:

別のユーザーによって作成された宛先オブジェクトを使用する場合に、オブジェクト権限は必要ありません。

SCHEDULER_ADMINロールは、表30-3に記載されているすべてのシステム権限付き(ADMINオプション付き)で作成されます。SCHEDULER_ADMINロールはDBA(ADMINオプション付き)に付与されます。

DBMS_SCHEDULERプロシージャおよびファンクションを定義者の権限PL/SQLブロックからコールする場合、オブジェクト権限はコール側のユーザーに直接付与される必要があります。すべてのPL/SQLストアド・プロシージャと同様に、DBMS_SCHEDULERは定義者の権限PL/SQLブロックからコールされると、データベース・オブジェクト上のロールを介して付与された権限を無視します。

SELECT ALL_SCHEDULER_*ビュー、SELECT USER_SCHEDULER_* ビュー、SELECT SYS.SCHEDULER$_JOBSUFFIX_S(ジョブ名生成用)、およびEXECUTE SYS.DEFAULT_JOB_CLASSの各オブジェクト権限が、PUBLICに付与されます。

親トピック: スケジューラの参照情報

30.6.2 スケジューラのデータ・ディクショナリ・ビュー

スケジューラに関する情報について一連のビューを問い合せることができます。

次の例は、my_job1の完了したインスタンスに関する情報を示しています。

```
SELECT JOB_NAME, STATUS, ERROR#
FROM DBA_SCHEDULER_JOB_RUN_DETAILS WHERE JOB_NAME = 'MY_JOB1';
JOB_NAME      STATUS           ERROR#
--------      --------------   ------
MY_JOB1       FAILURE           20000
```

表30-5に、スケジューラに関連付けられたビューを示します。*_SCHEDULER_JOBS、*_SCHEDULER_SCHEDULES、*_SCHEDULER_PROGRAMS、*_SCHEDULER_RUNNING_JOBS、*_SCHEDULER_JOB_LOGおよび*_SCHEDULER_JOB_RUN_DETAILSの各ビューは、ジョブの管理に特に役立ちます。スケジューラのビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

ノート:

次の表のビュー名の先頭にあるアスタリスクは、DBA、ALL または USER で置換されます。

表30-5 スケジューラのビュー

ビュー	説明
*_SCHEDULER_CHAIN_RULES	すべてのチェーンについて、すべてのルールが表示されます。
*_SCHEDULER_CHAIN_STEPS	すべてのチェーンについて、すべてのステップが表示されます。
*_SCHEDULER_CHAINS	すべてのチェーンが表示されます。
*_SCHEDULER_CREDENTIALS *_CREDENTIALS	すべての資格証明が表示されます。 ** *_SCHEDULER_CREDENTIALS は Oracle Database 12c では非推奨ですが、下位互換性のために引き続き使用できます。

ビュー	説明
	推奨されるビューは*_CREDENTIALS です。
*_SCHEDULER_DB_DESTS	すべてのデータベース宛先が表示されます。
*_SCHEDULER_DESTS	すべてのデータベース宛先と外部宛先が表示されます。
*_SCHEDULER_EXTERNAL_DESTS	すべての外部宛先が表示されます。
*_SCHEDULER_FILE_WATCHERS	すべての File Watcher が表示されます。
*_SCHEDULER_GLOBAL_ATTRIBUTE	スケジューラ属性の現行値が表示されます。
*_SCHEDULER_GROUP_MEMBERS	すべてのグループのすべてのグループ・メンバーが表示されます。
*_SCHEDULER_GROUPS	すべてのグループが表示されます。
*_SCHEDULER_INCOMPATIBILITY	非互換性定義のメンバーであるすべてのプログラムまたはジョブが表示されます。
*_SCHEDULER_JOB_ARGS	すべてのジョブについて、すべての引数の設定値が表示されます。
*_SCHEDULER_JOB_CLASSES	すべてのジョブ・クラスが表示されます。
*_SCHEDULER_JOB_DESTS	これらのビューには、複数の宛先のジョブの子ジョブを含む、ローカル・ジョブとリモート宛先のジョブの両方の状態が表示されます。これらのビューからジョブ宛先 ID (job_dest_id) を取得します。
*_SCHEDULER_JOB_LOG	設定されているロギング・レベルに応じて、ジョブの実行と状態変化が表示されます。
*_SCHEDULER_JOB_ROLES	Oracle Data Guard データベース・ロールによるすべてのジョブが表示されます。
*_SCHEDULER_JOB_RUN_DETAILS	完了(失敗または成功)したすべてのジョブ実行が表示されます。
*_SCHEDULER_JOBS	すべてのジョブ(使用可能なジョブと使用禁止のジョブ)が表示されます。

ビュー	説明
*_SCHEDULER_NOTIFICATIONS	すべてのジョブの状態の電子メール通知が表示されます。
*_SCHEDULER_PROGRAM_ARGS	すべてのプログラムに定義されているすべての引数が、デフォルト値(存在する場合)とともに表示されます。
*_SCHEDULER_PROGRAMS	すべてのプログラムが表示されます。
*_SCHEDULER_REMOTE_DATABASES	リモート・データベース・ジョブのソースおよび宛先として登録されている、現行のユーザーがアクセスできるリモート・データベースに関する情報が表示されます。
*_SCHEDULER_REMOTE_JOBSTATE	リモート・データベースにおいて現行のユーザーがアクセスできるジョブの状態に関する情報が表示されます。
*_SCHEDULER_RESOURCES	リソース・メタデータが表示されます。
*_SCHEDULER_RUNNING_CHAINS	実行中のすべてのチェーンが表示されます。
*_SCHEDULER_RUNNING_JOBS	現在実行中のすべてのジョブに関する状態情報が表示されます。
*_SCHEDULER_RSRC_CONSTRAINTS	ジョブまたはプログラムで使用されるリソースのタイプ、および必要とされる各リソースのユニットの数が表示されます。
*_SCHEDULER_SCHEDULES	すべてのスケジュールが表示されます。
*_SCHEDULER_WINDOW_DETAILS	完了したすべてのウィンドウの実行が表示されます。
*_SCHEDULER_WINDOW_GROUPS	すべてのウィンドウ・グループが表示されます。
*_SCHEDULER_WINDOW_LOG	ウィンドウに行われた変更の状態がすべて表示されます。
*_SCHEDULER_WINDOWS	すべてのウィンドウが表示されます。
*_SCHEDULER_WINGROUP_MEMBERS	すべてのウィンドウ・グループのメンバーが、グループ・メンバーごとに 1 行ずつ表示されます。

親トピック: スケジューラの参照情報

第V部 分散データベースの管理

分散データベース環境を管理できます。

- 分散データベースの概念
  分散データベースに関連する概念には、分散データベース・アーキテクチャ、データベース・リンク、トランザクション処理、アプリケーション開発および文字セットのサポートが含まれます。
- 分散データベースの管理
  分散データベースの管理には、グローバル名の管理、データベース・リンクの管理、位置および文の透過性の作成などのタスクが含まれます。
- 分散データベース・システムのアプリケーション開発
  分散データベース・システムのアプリケーション開発には、アプリケーション・データの分布の管理、データベース・リンクによって確立される接続の制御、参照整合性の維持、分散問合せのチューニング、およびリモート・プロシージャのエラー処理などのタスクが含まれます。
- 分散トランザクションの概念
  分散トランザクションは、分散データベースの2つ以上の異なるノード上のデータを更新します。
- 分散トランザクションの管理
  分散トランザクションの管理には、ノードのコミット・ポイント強度の指定、トランザクションの命名、およびインダウト・トランザクションの管理などのタスクが含まれます。

31 分散データベースの概念

分散データベースに関連する概念には、分散データベース・アーキテクチャ、データベース・リンク、トランザクション処理、アプリケーション開発および文字セットのサポートが含まれます。

- 分散データベース・アーキテクチャ
  分散データベース・システムを使用すると、アプリケーションはローカル・データベースおよびリモート・データベースのデータにアクセスできます。同機種間分散データベース・システムでは、個々のデータベースはOracle Databaseです。異機種間分散データベース・システムでは、少なくともデータベースの1つがOracle Database以外のデータベースです。分散データベースは、クライアント/サーバー・アーキテクチャを使用して情報の要求を処理します。
- データベース・リンク
  分散データベース・システムの中心的な概念は、データベース・リンクです。データベース・リンクは2つの物理データベース・サーバー間の接続であり、データベース・リンクを使用すると、クライアントから1つの論理データベースとしてこれらのサーバーにアクセスできます。
- 分散データベースの管理
  分散データベースの管理には、サイト自律性、セキュリティ、データベース・リンクの監査および管理ツールに関連するトピックが含まれます。
- 分散システムでのトランザクション処理
  トランザクションとは、1人のユーザーが実行する1つ以上のSQL文によって構成された論理作業単位のことです。トランザクションは、ユーザーの最初に実行可能なSQL文で開始し、そのユーザーによってコミットまたはロールバックされたときに終了します。リモート・トランザクションは、1つのリモート・ノードにアクセスする文のみで構成されます。分散トランザクションは、複数のノードにアクセスする文で構成されます。
- 分散データベース・アプリケーションの開発
  分散システムでアプリケーション開発を行うと、非分散システムには当てはまらない問題が発生します。
- 分散環境での文字セットのサポート
  分散環境では、異なるデータベースおよびクライアントが異なる文字セットを使用できます。

親トピック: 分散データベースの管理

31.1 分散データベース・アーキテクチャ

分散データベース・システムを使用すると、アプリケーションはローカル・データベースおよびリモート・データベースのデータにアクセスできます。同機種間分散データベース・システムでは、個々のデータベースはOracle Databaseです。異機種間分散データベース・システムでは、少なくともデータベースの1つがOracle Database以外のデータベースです。分散データベースは、クライアント/サーバー・アーキテクチャを使用して情報の要求を処理します。

- 同機種間分散データベース・システム
  同機種間分散データベース・システムには、Oracleデータベースのみが含まれます。
- 異機種間分散データベース・システム
  異機種間分散データベース・システムには、Oracleデータベースと非Oracleデータベースの両方が含まれます。
- クライアント/サーバー・データベース・アーキテクチャ
  データベース・サーバーとはデータベースを管理するOracleソフトウェアのことであり、クライアントとはサーバーの情報を要求するアプリケーションのことです。ネットワーク内の各コンピュータはノードと呼ばれ、1つ以上のデータベースのホストになることができます。分散データベース・システム内の各ノードは、状況に応じてクライアント、サーバー、あるいはその両方として機能します。

親トピック: 分散データベースの概念

31.1.1 同機種間分散データベース・システム

同機種間分散データベース・システムには、Oracleデータベースのみが含まれます。

- 同機種間分散データベース・システム
  同機種間分散データベース・システムとは、1つ以上のシステムに存在する2つ以上のOracle Databaseのネットワークを表します。
- 分散データベースと分散処理
  分散データベースおよび分散処理という用語は密接に関連していますが、その意味は異なります。
- 分散データベースとレプリケート・データベース
  分散データベース・システムおよびデータベース・レプリケーションという用語は関連していますが、その意味は異なります。

親トピック: 分散データベース・アーキテクチャ

31.1.1.1 同機種間分散データベース・システムについて

同機種間分散データベース・システムとは、1つ以上のシステムに存在する2つ以上のOracle Databaseのネットワークを表します。

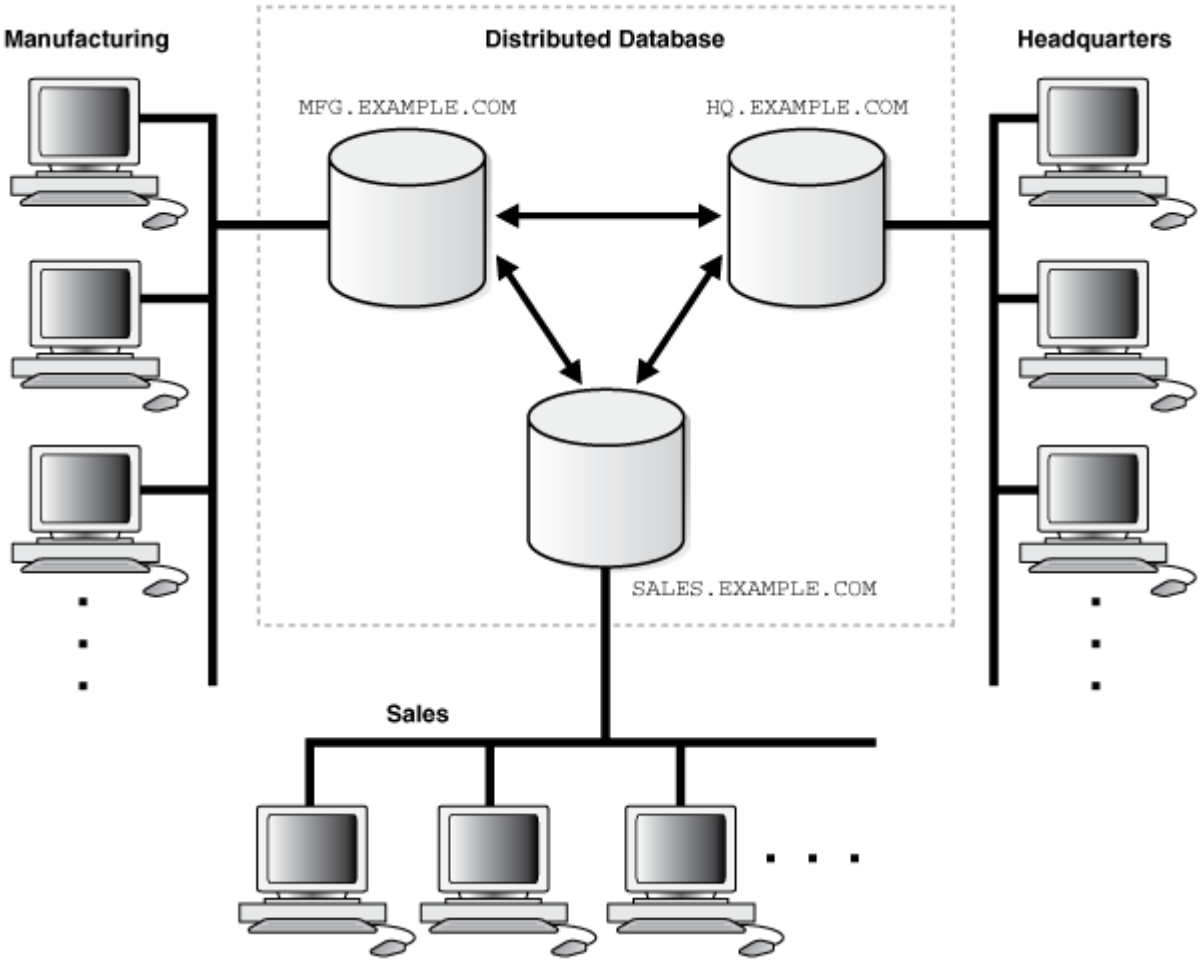
図31-1は、hq、mfgおよびsalesの3つのデータベースを接続している分散システムを示しています。アプリケーションは、1つの分散環境内にある複数のデータベースのデータに同時にアクセスしたり、データを同時に変更できます。たとえば、ローカル・データベースmfgの製造(Mfg)クライアントから発行する1回の問合せによって、ローカル・データベースのproducts表のデータとリモート・データベースhqのdept表のデータを結合したデータを取得できます。

クライアント・アプリケーションに対して、データベースの位置とプラットフォームは透過的です。また、分散システム内のリモート・オブジェクトに対してユーザーがローカル・オブジェクトと同じ構文を使用してアクセスできるように、リモート・オブジェクトのシノニムを作成することもできます。たとえば、mfgデータベースに接続しているときにデータベースhqのデータにアクセスする場合は、リモートのdept表に対するシノニムをmfg上に作成することで、次の問合せを発行できます。

```
SELECT * FROM dept;
```

このように、分散システムではローカル・データベースと同様のデータ・アクセスが可能です。mfgのユーザーは、アクセスするデータがリモート・データベースに存在していることを知る必要はありません。

図31-1 同機種間分散データベース



Oracle Databaseの分散データベース・システムは、異なるリリースのOracle Databaseで構成することが可能です。サポートされているOracle Databaseのリリースは、すべて分散データベース・システムに加わることができます。しかし、分散データベースを使用するアプリケーションでは、システムの各ノードで使用可能な機能を理解しておく必要があります。分散データベース・アプリケーションでは、Oracle Databaseでのみ使用可能なSQLの拡張をOracle7で実行できるかのような想定を行ってはなりません。

親トピック: 同機種間分散データベース・システム

31.1.1.2 分散データベースと分散処理

分散データベースおよび分散処理という用語は密接に関連していますが、その意味は異なります。

この2つの用語の定義は、次のとおりです。

- 分散データベース

  分散システムを構成する一連のデータベース。アプリケーションからは単一のデータソースのように見えます。

- 分散処理

  アプリケーションが自身のタスクをネットワーク内の異なるコンピュータ間に分散するときに発生する操作。たとえば、データベース・アプリケーションは通常、フロントエンド側のプレゼンテーション・タスクをクライアント・コンピュータに配置し、バックエンド側のデータベース・サーバーでデータベースへの共有アクセスを管理できるようにします。このことから、分散データベース・アプリケーション処理システムは、一般にクライアント/サーバー・データベース・アプリケーション・システムとも呼ばれます。

分散データベース・システムは、分散処理アーキテクチャを利用しています。たとえば、Oracle Databaseサーバーは、別のOracle Databaseサーバーが管理しているデータを要求するときにクライアントとして動作します。

親トピック: 同機種間分散データベース・システム

31.1.1.3 分散データベースとレプリケート・データベース

分散データベース・システムおよびデータベース・レプリケーションという用語は関連していますが、その意味は異なります。

純粋な(つまり、レプリケートされていない)分散データベースでは、すべてのデータとそれをサポートしているデータベース・オブジェクトの単一コピーが管理されています。通常、分散データベース・アプリケーションは、分散トランザクションを使用してローカルおよびリモートのデータにアクセスし、グローバル・データベースをリアルタイムで変更します。

ノート:

このマニュアルでは、純粋な分散データベースについてのみ説明しています。

レプリケーションという用語は、分散システムに属している複数のデータベースの間でデータベース・オブジェクトをコピーし、管理する操作を指します。レプリケーションは分散データベース・テクノロジに依存していますが、データベース・レプリケーションを使用すると、純粋な分散データベース環境では得られないような利点をアプリケーションが利用できます。

一般に、レプリケーションを使用すると代替のデータ・アクセス手段が提供されるので、ローカル・データベースのパフォーマンスが向上し、アプリケーションの可用性が保護されます。たとえば、アプリケーションは、ネットワークの通信量を最小限に抑えてパフォーマンスの最大化を図るために、リモート・サーバーではなくローカル・データベースにアクセスできます。また、ローカル・サーバーに障害が発生しても、レプリケートされたデータを持つ他のサーバーにアクセスできれば、アプリケーションは引き続き実行できます。

親トピック: 同機種間分散データベース・システム

31.1.2 異機種間分散データベース・システム

異機種間分散データベース・システムには、Oracleデータベースと非Oracleデータベースの両方が含まれます。

- 異機種間分散データベース・システムについて
  異機種間分散データベース・システムでは、少なくともデータベースの1つがOracle Database以外のシステムです。アプリケーションにとって、異機種間分散データベース・システムは1つのローカルなOracle Databaseのように見えます。ローカルのOracle Databaseサーバーでは、データの分散と異機種性は隠されています。
- 異機種間サービス
  異機種間サービス(HS)はOracle Databaseサーバー内部に統合されたコンポーネントであり、現在のOracle Transparent Gateway製品群を実現するためのテクノロジです。
- Transparent Gatewayエージェント
  Oracle Database以外の個々のシステムにアクセスする場合、異機種間サービスはTransparent Gatewayエージェントを使用して、Oracle Database以外の指定したシステムとのインタフェースの役目を果たします。エージェントはOracle Database以外のシステムに固有のものであり、システムのタイプごとに異なるエージェントが必要になります。
- Generic Connectivity
  Generic Connectivityを使用すると、異機種間サービスODBCエージェントまたは異機種間サービスOLE DBエージェントのどちらかを使用してOracle Database以外のデータ・ストアに接続できます。

親トピック: 分散データベース・アーキテクチャ

31.1.2.1 異機種間分散データベース・システムについて

異機種間分散データベース・システムでは、少なくともデータベースの1つがOracle Database以外のシステムです。アプリケー

ションにとって、異機種間分散データベース・システムは1つのローカルなOracle Databaseのように見えます。ローカルのOracle Databaseサーバーでは、データの分散と異機種性は隠されています。

Oracle Databaseサーバーは、Oracle Database以外のシステムへのアクセスに、Oracle異機種間サービスとエージェントを使用します。Oracle Database以外のデータ・ストアにOracle Transparent Gatewayを使用してアクセスする場合、エージェントはシステム固有のアプリケーションになります。たとえば、Oracle Database分散システムにSybaseデータベースを含める場合は、そのシステム内のOracle DatabaseがSybaseデータベースとやり取りできるように、Sybase固有のTransparent Gatewayを入手する必要があります。

Oracle Database以外のシステムがODBCまたはOLE DBプロトコルをサポートしている場合は、かわりにGeneric Connectivityを使用してOracle Database以外のデータ・ストアにアクセスできます。

ノート:

このマニュアルで Oracle 異機種間サービスについて説明しているのは、この章の概要的な内容のみです。異機種間サービスの詳細は、『Oracle Database Heterogeneous Connectivity ユーザーズ・ガイド』を参照してください。

親トピック: 異機種間分散データベース・システム

31.1.2.2 異機種間サービス

異機種間サービス(HS)はOracle Databaseサーバー内部に統合されたコンポーネントであり、現在のOracle Transparent Gateway製品群を実現するためのテクノロジです。

異機種間サービスは、Oracle Databaseゲートウェイ製品とその他の異機種間アクセス機能に対して共通のアーキテクチャと管理のメカニズムを提供します。また、以前にリリースされたほとんどのOracle Transparent Gatewayのユーザーに上位互換の機能を提供します。

親トピック: 異機種間分散データベース・システム

31.1.2.3 Transparent Gatewayエージェント

Oracle Database以外の個々のシステムにアクセスする場合、異機種間サービスはTransparent Gatewayエージェントを使用して、Oracle Database以外の特定のシステムとのインタフェースの役目を果たします。エージェントはOracle Database以外のシステムに固有のものであり、システムのタイプごとに異なるエージェントが必要になります。

Transparent Gatewayエージェントは、Oracle Databaseサーバー内の異機種間サービス・コンポーネントを使用して、Oracle DatabaseシステムとOracle Database以外のシステムとの間のやり取りを容易にします。エージェントはOracle Databaseサーバーのかわりに、Oracle Database以外のシステムでSQLとトランザクション要求を実行します。

関連項目:

Transparent Gatewayの詳細は、オラクル社が提供するゲートウェイ固有のマニュアルを参照してください。

親トピック: 異機種間分散データベース・システム

31.1.2.4 Generic Connectivity

Generic Connectivityを使用すると、異機種間サービスODBCエージェントまたは異機種間サービスOLE DBエージェントの

どちらかを使用してOracle Database以外のデータ・ストアに接続できます。

これらはどちらもOracle製品に標準機能として組み込まれています。ODBCまたはOLE DBの規格と互換性があれば、どのようなデータソースでもGeneric Connectivityエージェントを使用してアクセスできます。

Generic Connectivityの利点は、必ずしもシステム固有のエージェントを別途購入して構成する必要がないことです。ODBCドライバまたはOLE DBドライバを使用して、エージェントとインタフェースできます。ただし、一部のデータ・アクセス機能はTransparent Gatewayエージェントでしか利用できません。

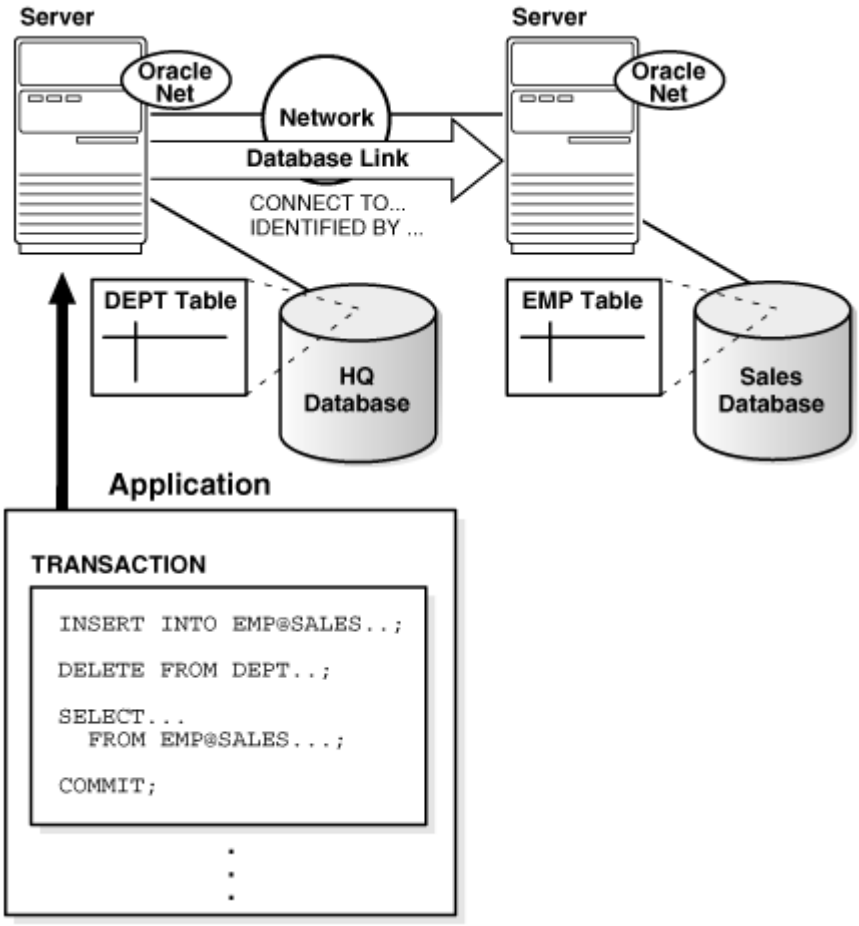
親トピック: 異機種間分散データベース・システム

31.1.3 クライアント/サーバー・データベース・アーキテクチャ

データベース・サーバーとはデータベースを管理するOracleソフトウェアのことであり、クライアントとはサーバーの情報を要求するアプリケーションのことです。ネットワーク内の各コンピュータはノードと呼ばれ、1つ以上のデータベースのホストになることができます。分散データベース・システム内の各ノードは、状況に応じてクライアント、サーバー、あるいはその両方として機能します。

図31-2のhqデータベースのホストは、そのローカル・データに対して文が発行されたときはデータベース・サーバーとして機能します。たとえば、トランザクション内の2番目の文は、ローカルのdept表に対して文を発行しています。しかし、リモート・データに対して文が発行されたときはクライアントとして機能します。たとえば、トランザクション内の最初の文は、salesデータベース内のリモートの表empに対して発行されています。

図31-2 Oracle Databaseの分散データベース・システム



クライアントは、データベース・サーバーに直接または間接に接続できます。直接接続になるのは、クライアントがサーバーに接続し、そのサーバー上に存在するデータベースの情報にアクセスするときです。たとえば、図31-2のように、hqデータベースに接続してこのデータベースのdept表にアクセスする場合は、次の文を発行できます。

```
SELECT * FROM dept;
```

この場合は、リモート・データベースのオブジェクトにアクセスしていないので、問合せは直接になります。

これに対して、間接接続になるのは、クライアントがサーバーに接続した後、別の異なるサーバー上のデータベースに格納されている情報にアクセスするときです。たとえば、図31-2のように、hqデータベースに接続した後、リモートのsalesデータベースのemp表にアクセスする場合は、次の文を発行できます。

```
SELECT * FROM emp@sales;
```

この場合は、アクセスしようとしているオブジェクトが直接接続しているデータベース上にないため、問合せは間接になります。

親トピック: 分散データベース・アーキテクチャ

31.2 データベース・リンク

分散データベース・システムの中心的な概念は、データベース・リンクです。データベース・リンクは2つの物理データベース・サーバー間の接続であり、データベース・リンクを使用すると、クライアントから1つの論理データベースとしてこれらのサーバーにアクセスできます。

- データベース・リンクの概要
  データベース・リンクは、あるOracle Databaseサーバーから別のデータベース・サーバーへの一方向の通信経路を定義するポインタです。
- 共有データベース・リンクの概要
  共有データベース・リンクとは、ローカル・サーバー・プロセスとリモート・データベースの間のリンクのことです。複数のクライアント・プロセスが同じリンクを同時に使用できるので、共有データベース・リンクと呼ばれます。
- データベース・リンクを使用する理由
  データベース・リンクの大きな利点として、ユーザーがリモート・データベースにある他のユーザーのオブジェクトに対して、オブジェクトの所有者が持つ権限の制限内でアクセスできるという点があります。つまり、ローカル・ユーザーはリモート・データベースのユーザーにならなくても、リモート・データベースへのリンクにアクセスできます。
- データベース・リンク内のグローバル・データベース名
  データベース・リンクの動作の仕組みを理解するには、まずグローバル・データベース名について理解する必要があります。分散データベース内の各データベースは、それぞれのグローバル・データベース名によって一意に識別されます。
- ループバック・データベース・リンクとしてのグローバル名
  データベースのグローバル名は、データベース・リンクを明示的に作成せずに、ループバック・データベース・リンクとして使用できます。SQL文中のデータベース・リンクが現行データベースのグローバル名と一致すると、データベース・リンクは事実上無視されます。
- データベース・リンクの名前
  通常、データベース・リンクの名前は、参照先のリモート・データベースのグローバル・データベース名と同じです。
- データベース・リンクのタイプ
  Oracle Databaseでは、プライベート、パブリックおよびグローバルのデータベース・リンクを作成できます。
- データベース・リンクのユーザー
  データベース・リンクのユーザーには、接続ユーザー、現行ユーザーおよび固定ユーザーが含まれます。
- データベース・リンクの作成: 例
  データベース・リンクを作成するには、CREATE DATABASE LINK文を使用します。
- スキーマ・オブジェクトとデータベース・リンク
  データベース・リンクを作成した後、リモート・データベースのオブジェクトにアクセスするSQL文を実行できます。特定のリモート・オブジェクトにアクセスするには、リモート・データベース内での認可も必要です。
- データベース・リンクの制限事項

データベース・リンクに適用されるいくつかの制限事項があります。

親トピック: 分散データベースの概念

31.2.1 データベース・リンクの概要

データベース・リンクは、あるOracle Databaseサーバーから別のデータベース・サーバーへの一方向の通信経路を定義するポインタです。

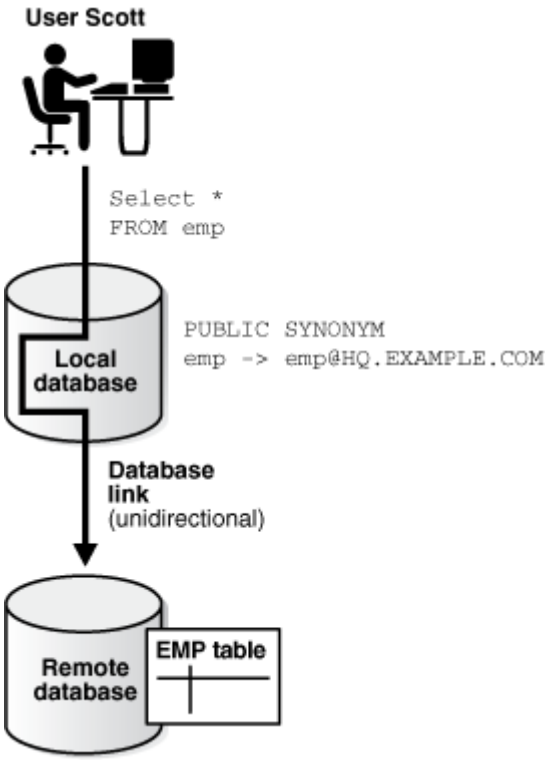
パブリック・データベース・リンクおよびプライベート・データベース・リンクの場合、実際には、リンク・ポインタはデータ・ディクショナリ表のエントリとして定義されます。リンクにアクセスするには、データ・ディクショナリ・エントリのあるローカル・データベースに接続する必要があります。グローバル・データベース・リンクの場合、リンク・ポインタはディレクトリ・サービスで定義されます。様々なタイプのデータベース・リンクの詳細は、「データベース・リンクのタイプ」を参照してください。

データベース・リンク接続が一方向であるということは、たとえばローカル・データベースAに接続しているクライアントはデータベースAに格納されているリンクを使用してリモート・データベースBの情報にアクセスできるが、データベースBに接続しているユーザーは同じリンクを使用してデータベースAのデータにアクセスできないことを意味します。データベースBのローカル・ユーザーがデータベースAのデータにアクセスする場合には、データベースBのデータ・ディクショナリに格納されるリンクを定義する必要があります。

データベース・リンク接続を使用することで、ローカル・ユーザーはリモート・データベースのデータにアクセスできるようになります。この接続を実現するためには、分散システム内の各データベースがネットワーク・ドメイン内で一意のグローバル・データベース名を持っていることが必要です。グローバル・データベース名は、分散システム内のデータベース・サーバーを一意に識別します。

図31-3は、ユーザーscottがグローバル名hq.example.comを使用して、リモート・データベースのemp表にアクセスしている例を示します。

図31-3 データベース・リンク



データベース・リンクには、プライベートとパブリックの2種類があります。プライベートの場合は、そのリンクを作成したユーザーのみがアクセスでき、パブリックの場合は、すべてのデータベース・ユーザーがアクセスできます。

データベース・リンクの間の重要な違いとして、リンク定義ごとに決められた、リンク接続の認証方法をあげることができます。ユーザーは、次のタイプのリンクによってリモート・データベースにアクセスします。

リンクのタイプ	説明
接続ユーザー・リンク	ユーザーはユーザー自身として接続します。つまり、ユーザーにはローカル・データベースのアカウントと同じユーザー名およびパスワードを持つリモート・データベースのアカウントが必要です。
固定ユーザー・リンク	ユーザーは、リンク内で参照されるユーザー名とパスワードを使用して接続します。たとえば、Jane が、ユーザー名とパスワード scott/password で hq データベースに接続する固定ユーザー・リンクを使用する場合、Jane は scott として接続し、scott に直接付与されている hq 内でのすべての権限と、hq データベースで scott に付与されているすべてのデフォルト・ロールを持ちます。
現行ユーザー・リンク	ユーザーはグローバル・ユーザーとして接続します。ローカル・ユーザーは、ストアド・プロシージャのコンテキスト内では、グローバル・ユーザーのパスワードをリンク定義に格納せずに、グローバル・ユーザーとして接続できます。たとえば、Jane は Scott が記述したプロシージャにアクセスでき、hq データベース上の Scott のアカウントと Scott のスキーマにアクセスできます。

データベース・リンクを作成するには、CREATE DATABASE LINK文を使用します。リンクを作成した後、SQL文の中でリンクを使用してスキーマ・オブジェクトを指定できます。

関連項目:

CREATE DATABASE文の構文は、『Oracle Database SQL言語リファレンス』

親トピック: データベース・リンク

31.2.2 共有データベース・リンクの概要

共有データベース・リンクとは、ローカル・サーバー・プロセスとリモート・データベースの間のリンクのことです。複数のクライアント・プロセスが同じリンクを同時に使用できるので、共有データベース・リンクと呼ばれます。

ローカル・データベースがデータベース・リンクを介してリモート・データベースに接続するとき、どちらのデータベースも専用サーバー・モードまたは共有サーバー・モードのいずれかで稼働しています。可能な組合せを次の表に示します。

ローカル・データベースのモード	リモート・データベースのモード
専用	専用
専用	共有サーバー
共有サーバー	専用
共有サーバー	共有サーバー

共有データベース・リンクは、これら4つのどの構成でも確立できます。共有リンクは、通常のデータベース・リンクと次の点で異なります。

- データベース・リンクを介して同じスキーマ・オブジェクトにアクセスする異なるユーザーの間で、ネットワーク接続を共有できます。
- ユーザーが特定のサーバー・プロセスからリモート・サーバーに接続を確立するときに、そのプロセスからリモート・サーバーに対してすでに確立されている接続を再利用できます。接続を再利用するには、その接続が同じサーバー・プロセスから同じデータベース・リンクを使用して確立されている必要があります(セッションは異なっていてもかまいません)。非共有のデータベース・リンクでは、接続が複数のセッション間で共有されることはありません。
- 共有サーバー構成で共有データベース・リンクを使用すると、ネットワーク接続はローカル・サーバーの共有サーバー・プロセスの外部で直接に確立されます。ローカル共有サーバーでの非共有のデータベース・リンクでは、この接続はローカル・ディスパッチャを介して確立されるため、ローカル・ディスパッチャのためのコンテキスト・スイッチングが発生し、データがディスパッチャを経由することになります。

関連項目:

共有サーバーの詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

親トピック: データベース・リンク

31.2.3 データベース・リンクを使用する理由

データベース・リンクの大きな利点として、ユーザーがリモート・データベースにある他のユーザーのオブジェクトに対して、オブジェクトの所有者が持つ権限の制限内でアクセスできるという点があります。つまり、ローカル・ユーザーはリモート・データベースのユーザーにならなくても、リモート・データベースへのリンクにアクセスできます。

たとえば、従業員が経費精算書を買掛管理(A/P)アプリケーションに送信し、さらにA/Pアプリケーションを使用するユーザーが従業員に関する情報をhqデータベースから取得する必要がある場合を想定します。A/Pのユーザーは、必要な情報を取得するために、hqデータベースに接続してリモートのhqデータベース内のストアド・プロシージャを実行できます。A/Pのユーザーは、自分のジョブを実行するためにhqデータベースのユーザーになる必要はなく、プロシージャに記述されている制御された方法でhqの情報にアクセスできればそれで済みます。

関連項目:

- データベース・リンク・ユーザーの詳細は、「データベース・リンクのユーザー」
- パスワードを管理者以外のユーザーから隠す方法の詳細は、「データベース・リンク情報の表示」

親トピック: データベース・リンク

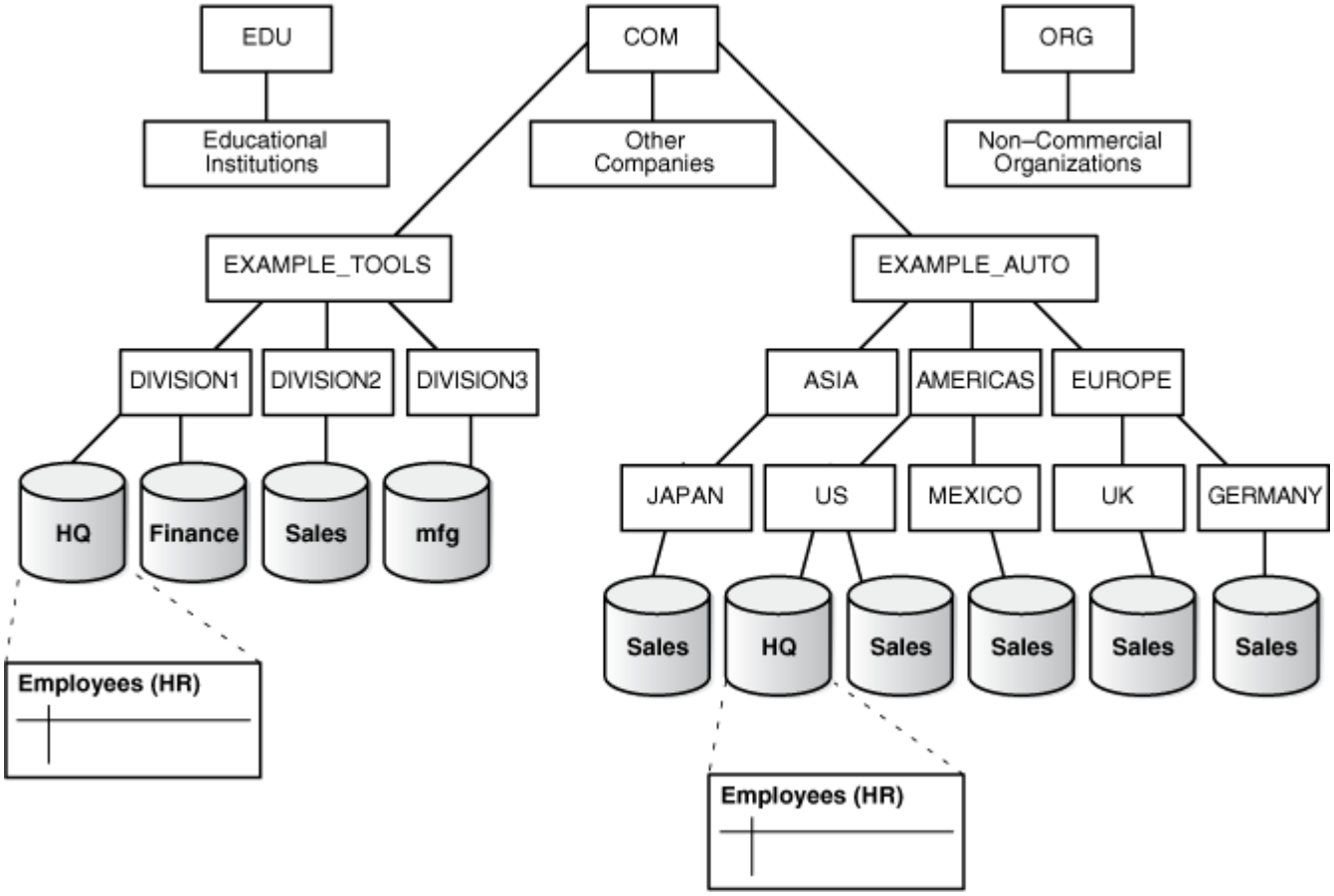
31.2.4 データベース・リンク内のグローバル・データベース名

データベース・リンクの動作の仕組みを理解するには、まずグローバル・データベース名について理解する必要があります。分散データベース内の各データベースは、それぞれのグローバル・データベース名によって一意に識別されます。

データベースは、DB_NAME初期化パラメータで指定された個々のデータベース名の前に、データベース作成時にDB_DOMAIN初期化パラメータで指定されたデータベース・ネットワーク・ドメインを付けることで、グローバル・データベース名を形成します。

たとえば、図31-4では、ネットワーク全体のデータベースを階層的な配置で表しています。

図31-4 ネットワーク化されたデータベースの階層的な配置



データベースの名前は、ツリーのリーフ(葉)から始まってルート(根)に至るまでの経路によって形成されます。たとえば、mfgデータベースは、comドメインのexample_toolsブランチのdivision3にあります。mfgのグローバル・データベース名は、ツリー内のノードを次のように連結して作成されます。

- `mfg.division3.example_tools.com`

複数のデータベースが1つの個別名を共有できますが、各データベースのグローバル・データベース名は一意にする必要があります。たとえば、ネットワーク・ドメインus.americas.example_auto.comとuk.europe.example_auto.comには、どちらにもsalesデータベースがあります。グローバル・データベース名の命名体系では、americas部門のsalesデータベースとeurope部門のsalesデータベースが次のように区別されます。

- `sales.us.americas.example_auto.com`
- `sales.uk.europe.example_auto.com`

関連項目:

グローバル・データベース名を指定および変更する方法を学習するには、「分散システムでのグローバル名の管理」

親トピック: データベース・リンク

31.2.5 ループバック・データベース・リンクとしてのグローバル名

データベースのグローバル名は、データベース・リンクを明示的に作成せずに、ループバック・データベース・リンクとして使用できます。SQL文中のデータベース・リンクが現行データベースのグローバル名と一致すると、データベース・リンクは事実上無視されます。

たとえば、データベースのグローバル名がdb1.example.comであると想定します。このデータベース上で次のSQL文を実行します。

```
SELECT * FROM hr.employees@db1.example.com;
```

この場合、SQL文の@db1.example.comの部分が事実上無視されます。

親トピック: データベース・リンク

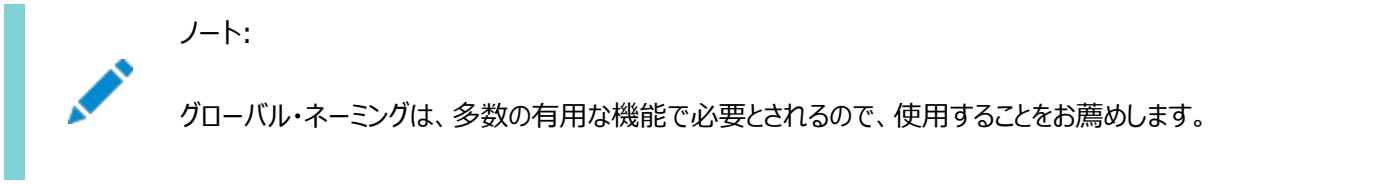
31.2.6 データベース・リンクの名前

通常、データベース・リンクの名前は、参照先のリモート・データベースのグローバル・データベース名と同じです。

たとえば、データベースのグローバル・データベース名がsales.us.example.comであれば、データベース・リンクの名前もsales.us.example.comになります。

初期化パラメータGLOBAL_NAMESをTRUEに設定すると、データベース・リンクの名前がリモート・データベースのグローバル・データベース名と同じになることが保証されます。たとえば、hqのグローバル・データベース名がhq.example.comで、GLOBAL_NAMESがTRUEの場合は、リンク名も必ずhq.example.comになります。データベースでチェックされるのはデータ・ディクショナリに保存されているグローバル・データベース名のドメイン部分で、初期化パラメータ・ファイルのDB_DOMAIN設定ではない点に注意してください(「グローバル・データベース名のドメインの変更」を参照)。

初期化パラメータGLOBAL_NAMESをFALSEに設定した場合は、グローバル・ネーミングを使用する必要はありません。データベース・リンクに自由に名前を付けられます。たとえば、hq.example.comへのデータベース・リンクにfooという名前を付けることができます。



ノート:

グローバル・ネーミングは、多数の有用な機能で必要とされるので、使用することをお薦めします。

グローバル・ネーミングを有効にすると、データベース・リンクの名前がリンクの参照先であるデータベースのグローバル名と同じになるため、データベース・リンクは本質的に分散データベースのユーザーに対して透過的になります。たとえば、次の文は、リモート・データベースsalesへのデータベース・リンクをローカル・データベース内に作成します。

```
CREATE PUBLIC DATABASE LINK sales.division3.example.com USING 'sales1';
```

関連項目:

初期化パラメータGLOBAL_NAMESの指定の詳細は、『Oracle Databaseリファレンス』

親トピック: データベース・リンク

31.2.7 データベース・リンクのタイプ

Oracle Databaseでは、プライベート、パブリックおよびグローバルのデータベース・リンクを作成できます。

これらの基本的なリンク・タイプは、どのユーザーに対してリモート・データベースへのアクセスが許可されているかによって異なります。

タイプ	所有者	説明
プライベート	リンクを作成したユーザー。次のビューによって所有権データを表示できます。 ● DBA_DB_LINKS	ローカル・データベースの特定のスキーマにリンクを作成します。プライベート・データベース・リンクの所有者またはスキーマ内の PL/SQL サブプログラムのみが、このリンクを使用して、対応するリモート・データベース内のデータベース・オブ

タイプ	所有者	説明
	● ALL_DB_LINKS ● USER_DB_LINKS	ジェクトにアクセスできます。
パブリック	PUBLIC と呼ばれるユーザー。プライベート・データベース・リンクと同じビューによって所有権データを表示できます。	データベース全体にわたるリンクを作成します。データベース内のすべてのユーザーと PL/SQL サブプログラムが、パブリック・リンクを使用して、対応するリモート・データベース内のデータベース・オブジェクトにアクセスできます。
グローバル	グローバル・データベース・リンクは、いずれのユーザーにも所有されません。グローバル・データベース・リンクはディレクトリ・サービスに存在します。	ネットワーク全体にわたるリンクを作成します。Oracle ネットワークでディレクトリ・サーバーを使用し、データベースがディレクトリ・サービスに登録されている場合、この情報をデータベース・リンクとして使用できます。どのデータベースのユーザーと PL/SQL サブプログラムでも、グローバル・データベース・リンクを使用して、対応するリモート・データベース内のオブジェクトにアクセスできます。グローバル・データベース・リンクは、ディレクトリ・サーバーのネット・サービス名の使用を指します。

分散データベースで利用するデータベース・リンクのタイプは、システムを使用しているアプリケーションの仕様要件によって決まります。リンクの選択時には、次の機能を考慮してください。

リンクのタイプ	機能
プライベート・データベース・リンク	このリンクは、パブリック・リンクやグローバル・リンクよりも安全です。その理由は、このリンクを使用してリモート・データベースにアクセスできるのが、プライベート・リンクの所有者と、同じスキーマ内のサブプログラムのみに限定されるためです。
パブリック・データベース・リンク	多数のユーザーがリモートの Oracle Database へのアクセス・パスを必要とするときは、データベース内のすべてのユーザーが使用できる 1 つのパブリック・データベース・リンクを作成できます。
グローバル・データベース・リンク	Oracle ネットワークでディレクトリ・サーバーを使用すると、管理者は、システムに存在するすべてのデータベースに対するグローバル・データベース・リンクを容易に管理できます。データベース・リンクの管理が集中化され、単純になります。 グローバル・データベース・リンク定義に関連付けられるユーザー・データはありません。グローバル・データベース・リンクは、接続されたユーザー・データベース・リンクとして操作できる必要があります。

関連項目:

- 異なるタイプのデータベース・リンクの作成方法を学習するには、「リンク・タイプの指定」

- リンクに関する情報へのアクセス方法を学習するには、「データベース・リンク情報の表示」

親トピック: データベース・リンク

31.2.8 データベース・リンクのユーザー

データベース・リンクのユーザーには、接続ユーザー、現行ユーザーおよび固定ユーザーが含まれます。

- データベース・リンク・ユーザーの概要
  リンクを作成するときは、データにアクセスするためにリモート・データベースに接続する必要のあるユーザーを決定します。
- 接続ユーザー・データベース・リンク
  接続ユーザー・リンクには、それに関連付けられた接続文字列がありません。接続ユーザー・リンクの利点は、リンクを参照するユーザーが同じユーザーとしてリモート・データベースに接続するため、資格証明がデータ・ディクショナリのリンク定義に格納されている必要がないことです。
- 固定ユーザー・データベース・リンク
  固定ユーザー・リンクの利点は、接続文字列に指定されたユーザーのセキュリティ・コンテキストを使用して、プライマリ・データベースのユーザーをリモート・データベースに接続することです。
- 現行ユーザー・データベース・リンク
  現行ユーザー・データベース・リンクでは、グローバル・ユーザーを利用します。グローバル・ユーザーは、X.509証明書またはパスワードによって認証される必要があり、リンクに関係する両方のデータベースのユーザーであることが必要です。

親トピック: データベース・リンク

31.2.8.1 データベース・リンクのユーザーの概要

リンクを作成するときは、どのユーザーがリモート・データベースに接続してデータにアクセスする必要があるかを決めます。

次の表は、データベース・リンクに関係するユーザーのカテゴリについて、それらの違いを説明したものです。

ユーザー・タイプ	説明	リンク作成構文のサンプル
接続ユーザー	固定のユーザー名およびパスワードが指定されていないデータベース・リンクにアクセスするローカル・ユーザー。SYSTEM が問合せでパブリック・リンクにアクセスする場合、接続ユーザーは SYSTEM であり、データベースはリモート・データベースの SYSTEM スキーマに接続します。 ノート: 接続ユーザーは、必ずしもリンクを作成したユーザーである必要はありません。リンクにアクセスしようとしているすべてのユーザーが接続ユーザーになります。	`CREATE PUBLIC DATABASE LINK hq USING 'hq';`
現行ユーザー	CURRENT_USER データベース・リンク内のグローバル・ユーザー。グローバル・ユーザーは、X.509 証明書(SSL 認証方式のエンタープライズ・ユーザー)またはパスワード(パスワード認証方式のエンタープライズ・ユーザー)によって認証される必要があり、リンクに関係する両方のデータベースのユーザーであることが必要です。 グローバル・セキュリティの詳細は、『Oracle Database エンタープライズ・ユー	`CREATE PUBLIC DATABASE LINK hq CONNECT TO CURRENT_USER using 'hq';`

ユーザー・タイプ	説明	リンク作成構文のサンプル
	ザー・セキュリティ管理者ガイド』を参照してください。	
固定ユーザー	ユーザー名/パスワードがリンク定義の一部になっているユーザー。リンクに固定ユーザーが含まれている場合は、その固定ユーザーのユーザー名とパスワードがリモート・データベースへの接続に使用されます。	`CREATE PUBLIC DATABASE LINK hq CONNECT TO jane IDENTIFIED BY password USING 'hq';`

ノート:

次のユーザーはデータベース・リンクのターゲット・ユーザーにはなれません: SYS および PUBLIC。

関連項目:

リンクを作成するときのユーザーの指定方法を学習するには、「リンク・ユーザーの指定」

親トピック: データベース・リンクのユーザー

31.2.8.2 接続ユーザー・データベース・リンク

接続ユーザー・リンクには、接続文字列が対応付けられていません。接続ユーザー・リンクの利点は、リンクを参照するユーザーが同じユーザーとしてリモート・データベースに接続するため、資格証明がデータ・ディクショナリのリンク定義に格納されている必要がないことです。

接続ユーザー・リンクには、いくつかのデメリットがあります。これらのリンクでは、ユーザーが接続先のリモート・データベースのアカウントと権限を持っている必要があるため、管理者の権限管理作業が増えます。また、必要以上の権限をユーザーに与えることは、セキュリティの基本概念である最低限の権限、つまり「ユーザーにはユーザー自身のジョブの実行に必要な権限のみを与える」に反することになります。

接続ユーザー・データベース・リンクの使用許可を左右する要因はいくつかありますが、最も重要な要因として、ユーザーがパスワードを使用してデータベースで認証されるのか、またはオペレーティング・システムやネットワーク認証サービスによって外部認証されるのかという要因があります。ユーザーが外部認証される場合、接続ユーザー・リンクの使用許可も、リモート・データベースでユーザーのリモート認証が許可されるかどうか(REMOTE_OS_AUTHENT初期化パラメータの設定)によって異なります。

REMOTE_OS_AUTHENTパラメータは、次のように働きます。

REMOTE_OS_AUTHENTの値	影響
リモート・データベースに対して TRUE	外部認証方式のユーザーは、接続ユーザー・データベース・リンクを使用してリモート・データベースに接続できます。
リモート・データベースに対して FALSE	外部認証方式のユーザーは、セキュリティ保護されたプロトコルまたはネットワーク認証サービス・オプションを使用しないかぎり、接続ユーザー・データベース・リンク

REMOTE_OS_AUTHENTの値	影響
	を使用してリモート・データベースに接続することはできません。

接続ユーザー・データベース・リンクが定義者権限のファンクション、プロシージャまたはパッケージ内からアクセスされる場合、定義者の認可IDを使用してリモート・ユーザーとして接続します。たとえば、ユーザーjaneがプロシージャscott.p (scottによって作成された定義者権限のプロシージャ)をコールし、プロシージャscott.pの内部からリンクが確立される場合、リンクが接続しているユーザーはscottになります。接続ユーザー・データベース・リンクを含む定義者権限のファンクション、プロシージャまたはパッケージを実行するには、ファンクション、プロシージャまたはパッケージを呼び出すユーザーにINHERIT REMOTE PRIVILEGES権限を付与する必要があります。

ノート:

REMOTE_OS_AUTHENT 初期化パラメータは非推奨です。これは、下位互換性のためにのみ残されています。

親トピック: データベース・リンクのユーザー

31.2.8.3 固定ユーザー・データベース・リンク

固定ユーザー・リンクの利点は、接続文字列で指定したユーザーのセキュリティ・コンテキストを使用して、プライマリ・データベースのユーザーをリモート・データベースに接続することです。

たとえば、ローカル・ユーザーjoeは、固定ユーザーscottとパスワードpasswordを指定したパブリック・データベース・リンクをjoeのスキーマ内に作成できます。janeがこの固定ユーザー・リンクを使用して問合せを発行すると、ローカル・データベースのユーザーであるjaneが、scott/passwordとしてリモート・データベースに接続します。

固定ユーザー・リンクでは、接続文字列にユーザー名とパスワードが対応付けられています。ユーザー名とパスワードは、データ・ディクショナリ表の別のリンク情報に格納されます。

親トピック: データベース・リンクのユーザー

31.2.8.4 現行ユーザー・データベース・リンク

現行ユーザー・データベース・リンクは、グローバル・ユーザーを利用します。グローバル・ユーザーは、X.509証明書またはパスワードによって認証される必要があり、リンクに関係する両方のデータベースのユーザーであることが必要です。

CURRENT_USERリンクを実行するユーザーは、必ずしもグローバル・ユーザーである必要はありません。たとえば、janeがパスワードを使用して買掛金データベースに(グローバル・ユーザーとしてではなく)認証されている場合、ストアド・プロシージャにアクセスしてhqデータベースからデータを取得できます。プロシージャでは、現行ユーザー・データベース・リンクを使用して彼女をグローバル・ユーザーscottとしてhqに接続しますユーザーscottはグローバル・ユーザーであり、SSL上の証明書を介して認証されますが、janeはグローバル・ユーザーではなく、SSLでは認証されません。

現行ユーザー・データベース・リンクによって、次のような結果になることに注意してください。

- 現行ユーザー・データベース・リンクがストアド・オブジェクト内からアクセスされていない場合、現行のユーザーはリンクにアクセスしている接続ユーザーと同じです。たとえば、scottが現行ユーザー・リンクを介してSELECT文を発行した場合、現行ユーザーはscottになります。
- プロシージャ、ビュー、トリガーなど、データベース・リンクにアクセスするストアド・オブジェクトを実行する場合、現行ユー

ザーとはストアド・オブジェクトを所有するユーザーで、オブジェクトをコールするユーザーではありません。たとえば、janeがプロシージャscott.p (scottが作成したプロシージャ)をコールし、コールされたプロシージャ内に現行ユーザー・リンクが表示された場合、scottがリンクの現行ユーザーになります。

- ストアド・オブジェクトが実行者権限のファンクション、プロシージャまたはパッケージである場合は、実行者の認可IDを使用してリモート・ユーザーとして接続します。たとえば、ユーザーjaneがプロシージャscott.p(scottによって作成された実行者権限のプロシージャ)をコールし、プロシージャscott.pの内部からリンクが確立される場合、リンクの現行ユーザーはjaneになります。
- データベースにエンタープライズ・ユーザーとして接続して、共有のグローバル・スキーマ内に存在するストアド・プロシージャで現行ユーザー・リンクを使用することはできません。たとえば、ユーザーjaneがデータベースhq上の共有スキーマguest内のストアド・プロシージャにアクセスする場合、リモート・データベースへのログオン用にこのスキーマ内の現行ユーザー・リンクは使用できません。

関連項目:

- データベース・リンクに関連するセキュリティ上の問題の詳細は、「分散データベースのセキュリティ」
- 実行者権限のファンクション、プロシージャまたはパッケージの詳細は、『Oracle Database PL/SQL言語リファレンス』。
- 現行ユーザー・データベース・リンクを含む定義者権限のファンクション、プロシージャまたはパッケージの実行の詳細は、Oracle Databaseセキュリティ・ガイドを参照してください

親トピック: データベース・リンクのユーザー

31.2.9 データベース・リンクの作成: 例

データベース・リンクを作成するには、CREATE DATABASE LINK文を使用します。

次の表は、ローカル・データベース内でリモートのsales.us.americas.example_auto.comデータベースへのデータベース・リンクを作成するSQL文の例です。

SQL文	接続先のデータベース	接続時のユーザー	リンク・タイプ
CREATE DATABASE LINK sales.us.americas.example_auto.com USING 'sales_us';	sales(ネット・サービス名 sales_us を使用)	接続ユーザー	プライベート接続ユーザー
CREATE DATABASE LINK foo CONNECT TO CURRENT_USER USING 'am_sls';	sales(サービス名 am_sls を使用)	現行グローバル・ユーザー	プライベート現行ユーザー
CREATE DATABASE LINK sales.us.americas.example_auto.com CONNECT TO scott IDENTIFIED BY password USING 'sales_us';	sales(ネット・サービス名 sales_us を使用)	scott(パスワード password を使用)	プライベート固定ユーザー
CREATE PUBLIC DATABASE LINK sales CONNECT TO scott IDENTIFIED BY password USING	sales(ネット・サービス	scott(パスワード	パブリック固定ユーザー

SQL文	接続先のデータベース	接続時のユーザー	リンク・タイプ
'rev';	名 rev を使用)	password を使用)	
CREATE SHARED PUBLIC DATABASE LINK sales.us.americas.example_auto.com CONNECT TO scott IDENTIFIED BY password AUTHENTICATED BY anupam IDENTIFIED BY password1 USING 'sales';	sales(ネット・サービス名 sales を使用)	scott(パスワード password を使用。認証用としてユーザー名 anupam とパスワード password1 を使用)	共有パブリック固定ユーザー

関連項目:

- リンクの作成方法を学習するには、「データベース・リンクの作成」
- CREATE DATABASE LINK文の構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: データベース・リンク

31.2.10 スキーマ・オブジェクトとデータベース・リンク

データベース・リンクを作成した後、リモート・データベースのオブジェクトにアクセスするSQL文を実行できます。特定のリモート・オブジェクトにアクセスするには、リモート・データベース内での認可も必要です。

たとえば、データベース・リンクfooを使用してリモート・オブジェクトempにアクセスするには、次の文を発行します。

```
SELECT * FROM emp@foo;
```

データベース・リンクを使用して適切な構成のオブジェクト名を作成することは、分散システムにおいて欠かせないデータ操作の1つです。

リモート・データベース・リンクを使用してプロシージャをコールし、そのプロシージャにROLLBACKコマンドが含まれている場合は、リモート・サイトで実行されたDML操作のみがロールバックされます。元のサイトで行われたDML変更はロールバックされません。

- データベース・リンクを使用したスキーマ・オブジェクトの命名
  Oracle Databaseは、グローバルにスキーマ・オブジェクトの名前を付けるために、グローバル・データベース名を使用します。
- リモート・スキーマ・オブジェクトへのアクセスに必要な認可
  リモート・スキーマ・オブジェクトにアクセスするには、リモート・データベース内のリモート・オブジェクトへのアクセス権を付与される必要があります。
- スキーマ・オブジェクトのシノニム
  Oracle Databaseでは、ユーザーからデータベース・リンク名を隠すことができるように、シノニムを作成できます。
- スキーマ・オブジェクトの名前解決
  スキーマ・オブジェクトへのアプリケーション参照を解決するために(名前解決と呼ばれるプロセス)、データベースは階層的にオブジェクト名を形成します。

親トピック: データベース・リンク

31.2.10.1 データベース・リンクを使用したスキーマ・オブジェクトの命名

Oracle Databaseは、グローバルにスキーマ・オブジェクトの名前を付けるために、グローバル・データベース名を使用します。

グローバル・データベース名は、次の形式をとります。

```
schema.schema_object@global_database_name
```

ここで:

- schemaは、データの論理構造(スキーマ・オブジェクト)の集まりです。スキーマはデータベース・ユーザーによって所有され、そのユーザーと同じ名前を持ちます。ユーザーはそれぞれ1つのスキーマを所有します。
- schema_objectは、表、索引、ビュー、シノニム、プロシージャ、パッケージ、データベース・リンクなどの論理データ構造です。
- global_database_nameは、リモート・データベースを一意に識別する名前です。この名前は、リモート・データベース初期化パラメータDB_NAMEとDB_DOMAINの連結と同じ必要があります(ただし、パラメータGLOBAL_NAMESがFALSEに設定されている場合は、任意の名前を使用できます)。

たとえば、ユーザーまたはアプリケーションは、データベースsales.division3.example.comへのデータベース・リンクを次のように使用して、リモート・データを参照できます。

```
SELECT * FROM scott.emp@sales.division3.example.com;  # emp table in scott's schema
SELECT loc FROM scott.dept@sales.division3.example.com;
```

GLOBAL_NAMESをFALSEに設定している場合は、sales.division3.example.comへのリンクに対して任意の名前を使用できます。たとえば、リンクfooをコールできます。次に、リモート・データベースに次のようにアクセスできます。

```
SELECT name FROM scott.emp@foo;  # link name different from global name
```

親トピック: スキーマ・オブジェクトとデータベース・リンク

31.2.10.2 リモート・スキーマ・オブジェクトへのアクセスに必要な認可

リモート・スキーマ・オブジェクトにアクセスするには、リモート・データベース内でそのオブジェクトへのアクセス権を付与される必要があります。

また、リモート・オブジェクトの更新、挿入または削除を実行するには、オブジェクトに対するREADまたはSELECT権限と、UPDATE、INSERTまたはDELETE権限を付与される必要があります。データベースにはリモート記述機能がないため、ローカル・オブジェクトにアクセスする場合とは異なり、リモート・オブジェクトにアクセスするにはREADまたはSELECT権限が必要です。データベースでは、構造を判断するためにリモート・オブジェクトのSELECT *を実行する必要があります。

親トピック: スキーマ・オブジェクトとデータベース・リンク

31.2.10.3 スキーマ・オブジェクトのシノニム

Oracle Databaseでは、シノニムを作成して、データベース・リンク名をユーザーから隠すことができます。

シノニムを使用すると、ローカル・データベースの表にアクセスする場合と同じ構文を使用して、リモート・データベースの表にアクセスできます。たとえば、リモート・データベース内の表に対して次の問合せを発行するとします。

```
SELECT * FROM emp@hq.example.com;
```

この場合、emp@hq.example.comに対応するシノニムempを作成すれば、同じデータにアクセスする際に次の問合せを発行できます。

```
SELECT * FROM emp;
```

関連項目:

データベース・リンクを使用して指定したオブジェクトのシノニムを作成する方法を学習するには、「シノニムを使用した位置の透過性の作成」

親トピック: スキーマ・オブジェクトとデータベース・リンク

31.2.10.4 スキーマ・オブジェクトの名前解決

スキーマ・オブジェクトへのアプリケーション参照を解決する(このプロセスを名前解決と呼びます)ために、データベースではオブジェクト名を階層的に構成しています。

たとえば、データベースでは、データベース内部の各スキーマは一意の名前を持ち、スキーマ内部では各オブジェクトが一意の名前を持つことが保証されています。そのため、スキーマ・オブジェクトの名前はデータベースの内部で常に一意になります。また、データベースはオブジェクトのローカル名へのアプリケーション参照を解決します。

分散データベースでは、表などのスキーマ・オブジェクトに対してシステム内のすべてのアプリケーションからアクセスが可能です。データベースは、グローバル・データベース名による階層ネーミング・モデルを拡張してグローバル・オブジェクト名を効果的に作成することによって、分散データベース・システム内のスキーマ・オブジェクトへの参照を解決します。たとえば、問合せでリモートの表を参照する場合は、その完全修飾名(表が存在しているデータベースを含む)を指定します。

たとえば、ローカル・データベースにユーザーSYSTEMとして接続するとします。

```
CONNECT SYSTEM@sales1
```

次に、データベース・リンクhq.example.comを使用して次の文を発行し、リモート・データベースhqのscottスキーマおよびjaneスキーマにあるオブジェクトにアクセスします。

```
SELECT * FROM scott.emp@hq.example.com;
INSERT INTO jane.accounts@hq.example.com (acc_no, acc_name, balance)
  VALUES (5001, 'BOWER', 2000);
UPDATE jane.accounts@hq.example.com
  SET balance = balance + 500;
DELETE FROM jane.accounts@hq.example.com
  WHERE acc_name = 'BOWER';
```

親トピック: スキーマ・オブジェクトとデータベース・リンク

31.2.11 データベース・リンクの制限事項

データベース・リンクに適用されるいくつかの制限事項があります。

データベース・リンクを使用して次の操作を実行することはできません。

- リモート・オブジェクトへの権限の付与。
- 一部のリモート・オブジェクトに対するDESCRIBEの実行。ただし、次のリモート・オブジェクトはDESCRIBEをサポートしています。
    - 表
    - ビュー
    - プロシージャ
    - ファンクション

- リモート・オブジェクトの分析。
- 参照整合性の定義または規定。
- リモート・データベース内のユーザーへのロールの付与。
- リモート・データベースにおけるデフォルト以外のロールの取得。たとえば、janeがローカル・データベースに接続し、scottとして接続する固定ユーザー・リンクを使用したストアド・プロシージャを実行する場合、janeはリモート・データベースにおけるscottのデフォルト・ロールを受け取ります。janeは、SET ROLEを発行してデフォルト以外のロールを取得することはできません。
- SSL、パスワードまたはMicrosoft Windowsのシステム固有の認証によって認証されていない現行ユーザー・リンクの使用。

関連項目:

- ユーザー定義タイプに対するデータベース・リンクの制限事項の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。
- LOBに対するデータベース・リンクの制限事項の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

親トピック: データベース・リンク

31.3 分散データベースの管理

分散データベースの管理には、サイト自律性、セキュリティ、データベース・リンクの監査および管理ツールに関連するトピックが含まれます。

- サイト自律性
  サイト自律性とは、分散データベース内の各サーバーが他のすべてのデータベースから独立して管理されることを意味します。
- 分散データベースのセキュリティ
  データベースは、ユーザーおよびロールのパスワード認証、ユーザーおよびロールのためのいくつかの外部認証タイプ(接続ユーザー・リンク用のKerberosバージョン5を含む)、クライアント/サーバー間およびサーバー間接続のためのログイン・パケットの暗号化を含め、分散データベース・システムのために非分散データベース環境で使用可能なすべてのセキュリティ機能をサポートします。
- データベース・リンクの監査
  監査処理は、常にローカルで実行する必要があります。つまり、適切な監査オプションがそれぞれのデータベースで設定されている場合は、ユーザーがローカル・データベース内で操作を実行し、データベース・リンクを介してリモート・データベースにアクセスすると、ローカルの操作はローカル・データベースで監査され、リモートの操作はリモート・データベースで監査されます。
- 管理ツール
  データベース管理者は、Oracle Databaseの分散データベース・システムを管理する際にいくつかのツールを選択できます。

関連項目:

- 同機種システムの管理方法を学習するには、「分散データベースの管理」
- 異機種間サービスの概念について学習するには、『Oracle Database Heterogeneous Connectivityユーザーズ・ガイド』を参照してください。

親トピック: 分散データベースの概念

31.3.1 サイト自律性

サイト自律性とは、分散データベース内の各サーバーが他のすべてのデータベースから独立して管理されることを意味します。

複数のデータベースが協調して動作する場合もありますが、各データベースはそれぞれ別々のデータ・リポジトリであり、個別に管理されます。Oracle Databaseの分散データベースのサイト自律性には、次のような利点があります。

- 独立性を維持する必要がある個々の企業またはグループの論理的な組織構造を、システムのノード群として反映できます。
- ローカル管理者がそれぞれ対応するローカル・データを管理します。したがって、個々のデータベース管理者の責任範囲が小さくなり、管理しやすくなります。
- 障害が単独で発生するため、分散データベースの他のノードに損傷を与える可能性が低くなります。1つのデータベース障害によってすべての分散操作が停止したり、パフォーマンスのボトルネックが生じることはありません。
- 孤立したシステム障害が発生した際、管理者は、システム内の他のノードとは独立してリカバリできます。
- 各ローカル・データベースにはデータ・ディクショナリが存在します。ローカル・データへのアクセスにグローバル・カタログは不要です。
- 各ノードで独立してソフトウェアをアップグレードできます。

Oracle Databaseでは分散データベース・システム内の各データベースを独立して管理できますが、システムのグローバルな要件を無視することのないようにしてください。たとえば、次のような作業が必要になることがあります。

- サーバー間接続を容易にするために作成したリンクをサポートするため、追加のユーザー・アカウントを各データベースに作成する。
- `COMMIT_POINT_STRENGTH`や`OPEN_LINKS`などの追加の初期化パラメータを設定する。

親トピック: 分散データベースの管理

31.3.2 分散データベースのセキュリティ

データベースは、ユーザーおよびロールのパスワード認証、ユーザーおよびロールのためのいくつかの外部認証タイプ(接続ユーザー・リンク用のKerberosバージョン5を含む)、クライアント/サーバー間およびサーバー間接続のためのログイン・パケットの暗号化を含め、分散データベース・システムのために非分散データベース環境で使用可能なすべてのセキュリティ機能をサポートします。

- データベース・リンクを介した認証
  データベース・リンクは、プライベートまたはパブリック、認証済または認証されていないのいずれかです。
- パスワードなしの認証
  接続ユーザーまたは現行ユーザーのデータベース・リンクを使用するときは、エンドツーエンドのセキュリティを得るために、Kerberosなどの外部認証ソースを使用できます。
- ユーザー・アカウントおよびロールのサポート
  分散データベース・システムでは、システムを使用するアプリケーションをサポートするために必要なユーザー・アカウントお

よびロールについて慎重に計画する必要があります。

- ユーザーと権限の集中管理
  ユーザーおよび権限の集中管理のために、認証方法を検討する必要があります。また、排他的にマップされたグローバル・ユーザーまたは共有スキーマ・ユーザーも考慮の対象になります。
- データの暗号化
  Oracle Advanced Securityオプションでは、データが解読または改ざんされないように、Oracle Netとその関連製品でネットワーク・データの暗号化とチェックサムも使用できます。これにより、RSA Data Security社のRC4またはデータ暗号化規格(DES)暗号化アルゴリズムを使用して、データを不当な表示から保護します。

関連項目:

外部認証の詳細は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。

親トピック: 分散データベースの管理

31.3.2.1 データベース・リンクを介した認証

データベース・リンクにはプライベートとパブリックの2種類があり、それぞれについて認証ありの場合と認証なしの場合があります。

パブリック・リンクを作成するには、リンク作成文でPUBLICキーワードを指定します。たとえば、次の文を発行できます。

```
CREATE PUBLIC DATABASE LINK foo USING 'sales';
```

認証ありのリンクを作成するには、データベース・リンク作成文でCONNECT TO句、AUTHENTICATED BY句、あるいはこれら両方の句を指定します。たとえば、次の文を発行できます。

```
CREATE DATABASE LINK sales CONNECT TO scott IDENTIFIED BY password USING 'sales';
CREATE SHARED PUBLIC DATABASE LINK sales CONNECT TO nick IDENTIFIED BY password1
     AUTHENTICATED BY david IDENTIFIED BY password2 USING 'sales';
```

次の表は、ユーザーがリンクを介してリモート・データベースにアクセスする方法を示しています。

リンク・タイプ	認証済	セキュリティ・アクセス
プライベート	いいえ	データベースでは、リモート・データベースに接続するときに、ローカル・セッションから取得したセキュリティ情報(ユーザーID/パスワード)を使用します。そのため、このリンクは接続ユーザー・データベース・リンクです。2 つのデータベース間で、パスワードが同期化されている必要があります。
プライベート	はい	ユーザーID/パスワードがローカル・セッションのコンテキストからではなく、リンク定義から取得されます。そのため、このリンクは固定ユーザー・データベース・リンクです。 この構成では、2 つのデータベース間で異なるパスワードを使用できますが、ローカル・データベース・リンクのパスワードはリモート・データベースのパスワードと一致している必要があります。
パブリック	いいえ	動作はプライベートの非認証リンクとほぼ同じですが、すべてのユーザーがこのリ

リンク・タイプ	認証済	セキュリティ・アクセス
		モート・データベースへのポインタを参照できる点が異なります。
パブリック	はい	ローカル・データベースのすべてのユーザーがリモート・データベースにアクセスでき、すべてのユーザーが同じユーザーID/パスワードを使用して接続します。

ノート:

次の初期化パラメータにより、データベース・リンク接続のセキュリティを強化できます。

- OUTBOUND_DBLINK_PROTOCOLS 初期化パラメータは、指定されたリストからのプロトコルのみを送信データベース・リンク通信で使用するように制限する Oracle Net トランスポート・プロトコルを指定できます。
- ALLOW_GLOBAL_DBLINKS 初期化パラメータは、グローバル・データベース・リンク情報のための LDAP 参照を許可または禁止できます。

関連項目:

- 『Oracle Databaseセキュリティ・ガイド』
- OUTBOUND_DBLINK_PROTOCOLS初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- ALLOW_GLOBAL_DBLINKS初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: 分散データベースのセキュリティ

31.3.2.2 パスワードなしの認証

接続ユーザーまたは現行ユーザーのデータベース・リンクを使用するときは、Kerberosなどの外部認証ソースを使用してエンドツーエンドのセキュリティを確立できます。

エンドツーエンド認証では、資格証明がサーバー間で渡され、同じドメインに属するデータベース・サーバーによって資格証明を認証できます。たとえば、janeがローカル・データベースで外部認証されているときに、接続ユーザー・リンクを使用してリモート・データベースに彼女自身として接続しようとする場合、ローカル・サーバーからリモート・データベースにセキュリティ・チケットが渡されます。

親トピック: 分散データベースのセキュリティ

31.3.2.3 ユーザー・アカウントおよびロールのサポート

分散データベース・システムでは、システムを使用するアプリケーションのサポートに必要なユーザー・アカウントおよびロールについて慎重に計画する必要があります。

次のことに注意してください。

- サーバー間接続の確立に必要なユーザー・アカウントは、分散データベース・システムのすべてのデータベースで使用可

能である必要があります。

- 分散データベース・アプリケーションのユーザーに対してアプリケーション権限を使用可能にするために必要なロールは、分散データベース・システムのすべてのデータベースに存在する必要があります。

分散データベース・システム内のノードに対応するデータベース・リンクを作成する際は、それらのリンクを使用するサーバー間接続をサポートするために、各サイトでどのユーザー・アカウントとロールが必要になるかを決めてください。

通常、分散環境では、ユーザーは多数のネットワーク・サービスへのアクセスが必要になります。個々のユーザーが個々のネットワーク・サービスにアクセスするために個別の認証を構成する必要があるときは、特に大規模なシステムの場合にセキュリティの管理が難しくなることがあります。

関連項目:

各種データベース・リンクをシステムでサポートするために使用可能にする必要があるユーザー・アカウントの詳細は、「データベース・リンクの作成」

親トピック: 分散データベースのセキュリティ

31.3.2.4 ユーザーと権限の集中管理

ユーザーおよび権限の集中管理のために、認証方法を検討する必要があります。また、排他的にマップされたグローバル・ユーザーまたは共有スキーマ・ユーザーも考慮の対象になります。

- ユーザーと権限の集中管理について
  データベースでは、様々な方法で分散システム内のユーザーとユーザーの権限を管理できます。
- 排他的にマップされたグローバル・ユーザー
  ユーザーと権限の管理を集中化するための1つのオプションは、集中ディレクトリにグローバル・ユーザーを作成し、グローバル・ユーザーが接続する必要のあるすべてのデータベースにユーザーを作成することです。
- 共有スキーマ・ユーザー
  共有スキーマ・ユーザー機能を使用すると、エンタープライズ・ディレクトリ・サービスでグローバル・ユーザーを集中管理できるようになります。このディレクトリ内で管理されるユーザーのことを、エンタープライズ・ユーザーと呼びます。

親トピック: 分散データベースのセキュリティ

31.3.2.4.1 ユーザーと権限の集中管理について

データベースでは、様々な方法で分散システム内のユーザーとユーザーの権限を管理できます。

たとえば、次のような方法があります。

- エンタープライズ・ユーザー管理

  パスワード、Kerberos、またはPKI証明書を使用して認証されるグローバル・ユーザーを作成できます。こうしたユーザーとユーザーの権限は、独立したディレクトリ・サービスを使用してディレクトリで管理できるようになります。

- ネットワーク認証サービス

  この一般的な方法によって、分散環境のセキュリティ管理が容易になります。Oracle Advanced Securityオプションを使用することで、Oracle NetとOracle Databaseの分散データベース・システムのセキュリティを強化できます。Oracle以外の認証ソリューションの例としては、Microsoft Windowsのシステム固有の認証があります。

関連項目:

- Oracle Databaseセキュリティの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください
- Oracle Databaseのエンタープライズ・ユーザー・セキュリティの詳細は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください
- Microsoft Active Directoryによるユーザーの構成の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください

親トピック: ユーザーと権限の集中管理

31.3.2.4.2 排他的にマップされたグローバル・ユーザー

ユーザーと権限の管理を集中化するための1つのオプションは、集中ディレクトリにグローバル・ユーザーを作成し、グローバル・ユーザーが接続する必要のあるすべてのデータベースにユーザーを作成することです。

たとえば、次のSQL文を使用してfredというグローバル・ユーザーを作成できます。

```
CREATE USER fred IDENTIFIED GLOBALLY AS 'CN=fred adams,O=Oracle,C=England';
```

このソリューションを使用すると、1つのグローバル・データベース・ユーザーが集中型ディレクトリで認証されるようにして、そのデータベース・ユーザーをディレクトリ・ユーザーに排他的にマップできるようになります。

排他的にマップされたグローバル・ユーザーのソリューションでは、このユーザーがアクセスする必要のあるすべてのデータベースでfredと呼ばれるユーザーを作成することが必要になります。ユーザーの多くはアプリケーション・スキーマにアクセスする権限は必要とするものの、独自のスキーマは不要であるため、すべてのグローバル・ユーザーに対してデータベースごとに別個のアカウントを作成すると、大幅なオーバーヘッドが生じます。この問題のため、データベースでは、共有スキーマ・ユーザーもサポートされています。このユーザーは、すべてのデータベースで単一の汎用スキーマにアクセスできるグローバル・ユーザーです。

親トピック: ユーザーと権限の集中管理

31.3.2.4.3 共有スキーマ・ユーザー

共有スキーマ・ユーザー機能を使用すると、エンタープライズ・ディレクトリ・サービスでグローバル・ユーザーを集中管理できるようになります。このディレクトリ内で管理されるユーザーのことを、エンタープライズ・ユーザーと呼びます。

このディレクトリには、次に関する情報を含めることができます。

- エンタープライズ・ユーザーが分散システムのどのデータベースにアクセスできるのか。
- エンタープライズ・ユーザーが各データベースのどのロールを使用できるのか。
- エンタープライズ・ユーザーが各データベースのどのスキーマに接続できるのか。

各データベースの管理者は、エンタープライズ・ユーザーが接続する必要のあるデータベースごとに各エンタープライズ・ユーザーのグローバル・ユーザー・アカウントを作成する必要はありません。そのかわりに、複数のエンタープライズ・ユーザーが共有スキーマと呼ばれる同じデータベース・スキーマに接続できます。

ノート:

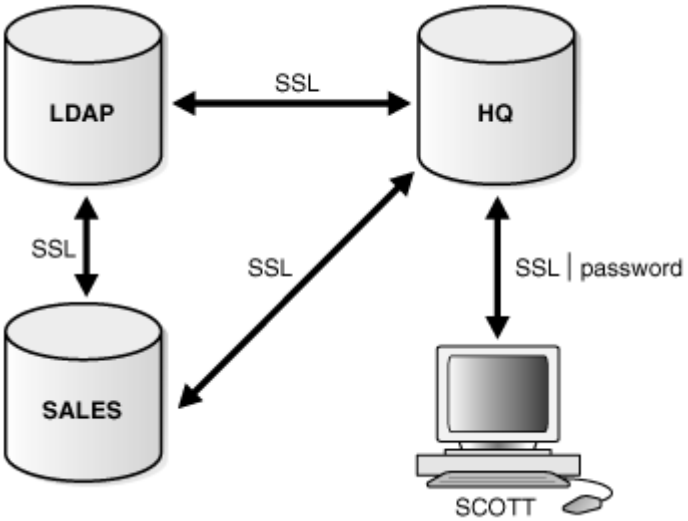
共有スキーマ内の現行ユーザー・データベース・リンクにはアクセスできません。

たとえば、jane、billおよびscottが全員、人事管理アプリケーションを使用しているとします。hqアプリケーション・オブジェクトは、hqデータベース上のguestスキーマにすべて格納されています。この場合、共有スキーマとして使用するローカル・グローバ

ル・ユーザー・アカウントを作成できます。このグローバル・ユーザー名(共有スキーマ名)はguestです。jane、billおよびscottは、いずれもディレクトリ・サービスでエンタープライズ・ユーザーとして作成されます。また、彼らをディレクトリのguestスキーマにマップし、hqアプリケーションで別の認可を割り当てることもできます。

図31-5は、エンタープライズ・ディレクトリ・サービスを使用したグローバル・ユーザーのセキュリティの例を示しています。

図31-5 グローバル・ユーザーのセキュリティ



エンタープライズ・ディレクトリ・サービスに、hqおよびsalesのエンタープライズ・ユーザーに関する次の情報が格納されているとします。

データベース	ロール	スキーマ	エンタープライズ・ユーザー
hq	clerk1	guest	bill scott
sales	clerk2	guest	jane scott

また、hqおよびsalesのローカル管理者が次の文を発行したとします。

データベース	CREATE文
hq	CREATE USER guest IDENTIFIED GLOBALLY AS ''; CREATE ROLE clerk1 GRANT select ON emp; CREATE PUBLIC DATABASE LINK sales_link CONNECT AS CURRENT_USER USING 'sales';
sales	CREATE USER guest IDENTIFIED GLOBALLY AS ''; CREATE ROLE clerk2 GRANT select ON dept;

ここで、salesに関係する分散トランザクションを実行するために、エンタープライズ・ユーザーscottがローカル・データベースhqへの接続を要求するとします。このとき、次のステップが実行されます(ただし、必ずしもこの順序どおりとはかぎりません)。

1. エンタープライズ・ユーザーscottが、SSLまたはパスワードを使用して認証されます。
2. ユーザーscottが次の文を発行します。

```
SELECT e.ename, d.loc
FROM emp e, dept@sales_link d
```

```
WHERE e.deptno=d.deptno;
```

3. データベースhqとsalesが、SSLを使用して相互に認証します。
4. データベースhqはエンタープライズ・ディレクトリ・サービスを問い合せて、エンタープライズ・ユーザーscottがhqにアクセスできるかどうかを判断し、scottがロールclerk1を使用してローカル・スキーマguestにアクセスできることを確認します。
5. データベースsalesはエンタープライズ・ディレクトリ・サービスを問い合せて、エンタープライズ・ユーザーscottがsalesにアクセスできるかどうかを判断し、scottがロールclerk2を使用してローカル・スキーマguestにアクセスできることを確認します。
6. エンタープライズ・ユーザーscottはsalesにログインし、ロールclerk2を使用してguestスキーマにアクセスします。そこでSELECTを発行し、必要な情報を取得してその情報をhqに送信します。
7. データベースhqは要求されたデータをsalesから受信し、それをクライアントscottに返します。

ノート:

Oracle Database 18c 以降の場合:

- ディレクトリ・グループをグローバル・ロールにマップするために、CREATE ROLE 文または ALTER ROLE 文の GLOBALLY AS [domain_name_of_directory_group]句を使用してグローバル・ロールを作成できます。グローバル・ユーザーは、グローバル・ロールの有効化前に、エンタープライズ・ディレクトリ・サービスでロールの使用を認可しておく必要があります。

  CREATE ROLE 文および ALTER ROLE 文の構文は、『Oracle Database SQL 言語リファレンス』を参照してください。

- Microsoft Active Directory で直接ユーザーを認証および認可できます。そのため、Oracle データベースのユーザーとロールは、Oracle Enterprise User Security (EUS)などの中間ディレクトリ・サービスを使用することなく、Active Directory のユーザーとグループに直接マップできます。

  Microsoft Active Directory によるユーザーの構成の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

関連項目:

- Oracle Databaseのエンタープライズ・ユーザー・セキュリティの詳細は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください
- Oracle Databaseセキュリティの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください

親トピック: ユーザーと権限の集中管理

31.3.2.5 データの暗号化

Oracle Advanced Securityオプションでは、データが解読または改ざんされないように、Oracle Netとその関連製品でネッ

トワーク・データの暗号化とチェックサムも使用できます。これにより、RSA Data Security社のRC4またはデータ暗号化規格(DES)暗号化アルゴリズムを使用して、データを不当な表示から保護します。

データが転送中に改ざん、削除または再現されなかったことを保証するために、Oracle Advanced Securityオプションのセキュリティ・サービスは、暗号的に安全なメッセージ・ダイジェストを生成し、ネットワーク上に送られる各パケットにそのダイジェストを含めることができます。

関連項目:

Oracle Advanced Securityオプションのこれらの機能およびその他の機能の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

親トピック: 分散データベースのセキュリティ

31.3.3 データベース・リンクの監査

操作の監査は、常にローカルで実行する必要があります。つまり、適切な監査オプションがそれぞれのデータベースで設定されている場合は、ユーザーがローカル・データベース内で操作を実行し、データベース・リンクを介してリモート・データベースにアクセスすると、ローカルの操作はローカル・データベースで監査され、リモートの操作はリモート・データベースで監査されます。

リモート・データベースでは、正常終了した接続要求とその後のSQL文が別のサーバーからのものか、またはローカルに接続しているクライアントからのものかを判断することはできません。たとえば、次のような場合を考えてみます。

- 固定ユーザー・リンク`hq.example.com`により、ローカル・ユーザー`jane`がリモート・ユーザー`scott`としてリモートの`hq`データベースに接続します。
- ユーザー`scott`は、リモート・データベースで監査されます。

リモート・データベース・セッション中に実行される操作は、あたかも`scott`が`hq`にローカルに接続し、そこで同じ操作を実行しているかのように監査されます。`jane`がリモート・データベースで何を実行しているのかを監査する場合は、リモート・データベースで監査オプションを設定し、リンクに埋め込まれているユーザー名(この場合は`hq`データベースの`scott`)の操作を捕捉する必要があります。

ノート:

グローバル・ユーザーに対応するグローバル・ユーザー名を監査できます。

リモート・オブジェクトに対してローカルの監査オプションを設定することはできません。したがって、リモート・オブジェクトへのアクセスをリモート・データベースで監査することはできますが、データベース・リンクの使用は監査できません。

親トピック: 分散データベースの管理

31.3.4 管理ツール

データベース管理者は、Oracle Databaseの分散データベース・システムを管理する際にいくつかのツールを選択できます。

- Cloud Controlと分散データベース
  Oracle Enterprise Manager Cloud Controlは、グラフィカル・ユーザー・インタフェース(GUI)を提供するOracle Database管理ツールです。Cloud Controlでは、操作性に優れたインタフェースを介して、分散データベースの管理機能を提供します。

- サード・パーティ製管理ツール
  現在、60以上の企業が、Oracleデータベースおよびネットワークの管理に役立つ、真にオープンな環境を提供する150以上の製品を生産しています。
- SNMPサポート
  Oracle Simple Network Management Protocol (SNMP)のサポートでは、ネットワーク管理機能以外に、任意のSNMPベースのネットワーク管理システムによってOracle Databaseサーバーを検索および問合せできるようにします。

親トピック: 分散データベースの管理

31.3.4.1 Cloud Controlと分散データベース

Oracle Enterprise Manager Cloud Controlは、グラフィカル・ユーザー・インタフェース(GUI)を提供するOracle Database管理ツールです。Cloud Controlでは、操作性に優れたインタフェースを介して、分散データベースの管理機能を提供します。

Cloud Controlを使用して、次のことを実行できます。

- 複数のデータベースの管理。Cloud Controlを使用して、1つのデータベースを管理したり、複数のデータベースを同時に管理できます。
- データベース管理作業の集中化。世界中のあらゆる場所のあらゆるOracle Databaseプラットフォームで稼働しているローカルおよびリモートの両方のデータベースを管理できます。また、Oracle Netがサポートしている任意のネットワーク・プロトコルでこれらのOracle Databaseプラットフォームを接続できます。
- SQL、PL/SQLおよびCloud Controlコマンドの動的な実行。Cloud Controlを使用して、文を入力、編集および実行できます。Cloud Controlにより、実行した文の履歴も保持されます。

  これにより、それらの文を再入力しなくても再実行できるので、分散データベース・システムで非常に長い文を繰り返し実行する必要がある場合に特に便利です。
- グローバル・ユーザー、グローバル・ロール、エンタープライズ・ディレクトリ・サービスなどのセキュリティ機能の管理。

親トピック: 管理ツール

31.3.4.2 サード・パーティ製管理ツール

現在、Oracle Databaseおよびネットワークの管理に役立つ製品が、60を超える企業から150以上出荷されており、真にオープンな環境を実現しています。

親トピック: 管理ツール

31.3.4.3 SNMPのサポート

Oracle Simple Network Management Protocol (SNMP)のサポートでは、ネットワーク管理機能以外に、任意のSNMPベースのネットワーク管理システムによってOracle Databaseサーバーを検索および問合せできるようにします。

SNMPは、次に示す多数の一般的なネットワーク管理システムの基盤になる公認規格です。

- HP社のOpenView
- Digital社のPOLYCENTER Manager on NetView
- IBM社のNetView/6000
- Novell社のNetWare Management System

- SunSoft社のSunNet Manager

ノート:

Oracle Net リスナーでの SNMP サポートは非推奨になりました。新しい実装では SNMP を使用しないことをお薦めします。詳細は、『Oracle Database アップグレード・ガイド』を参照してください。

親トピック: 管理ツール

31.4 分散システムでのトランザクション処理

トランザクションとは、1人のユーザーが実行する1つ以上のSQL文によって構成された論理作業単位のことです。トランザクションは、ユーザーの最初に実行可能なSQL文で開始し、そのユーザーによってコミットまたはロールバックされたときに終了します。リモート・トランザクションは、1つのリモート・ノードにアクセスする文のみで構成されます。分散トランザクションは、複数のノードにアクセスする文で構成されます。

- リモートSQL文
  リモートSQL文は、同じリモート・ノードに存在する1つ以上のリモート表の問合せまたは変更を行います。
- 分散SQL文
  分散SQL文は、2つ以上のノード上のデータの問合せまたは変更を行います。
- リモートおよび分散型の文のための共有SQL
  共有SQLを使用したリモートまたは分散型の文の仕組みは、基本的にローカル文のものと同じです。
- リモート・トランザクション
  リモート・トランザクションには1つ以上のリモート文があり、そのすべてが1つのリモート・ノードを参照しています。
- 分散トランザクション
  分散トランザクションとは、1つ以上の文からなり、それらが個別に、またはグループとして、分散データベースの複数のノードのデータを更新するトランザクションのことです。
- 2フェーズ・コミット・メカニズム
  データベースの2フェーズ・コミット・メカニズムは、分散トランザクションに関係するすべてのデータベース・サーバーが、そのトランザクション内の文のすべてをコミットするか、すべてをロールバックするかのどちらか一方のみになるように保証するものです。
- データベース・リンクの名前解決
  SQL文にグローバル・オブジェクト名への参照が含まれていると、データベースでは、グローバル・オブジェクト名に指定されているデータベース名と名前が一致するデータベース・リンクを検索します。
- スキーマ・オブジェクトの名前解決
  Oracleデータベースがリモート・データベースに接続したときのリモート・スキーマの決定方法を理解するのは重要なことです。
- ビュー、シノニムおよびプロシージャでのグローバル名前解決
  グローバル・オブジェクト名は、完全または部分的である可能性があります。

親トピック: 分散データベースの概念

31.4.1 リモートSQL文

リモートSQL文は、同じリモート・ノードに存在する1つ以上のリモート表の問合せまたは変更を行います。

リモート問合せ文とは、そのすべてが同一のリモート・ノードに存在している1つ以上のリモート表から情報を選択する問合せのことです。たとえば、次の問合せは、リモートのsalesデータベースのscottスキーマにあるdept表のデータにアクセスします。

```
SELECT * FROM scott.dept@sales.us.americas.example_auto.com;
```

リモート更新文とは、そのすべてが同一のリモート・ノードに存在している1つ以上の表のデータを変更する更新のことです。たとえば、次の問合せは、リモートのsalesデータベースのscottスキーマにあるdept表を更新します。

```
UPDATE scott.dept@mktng.us.americas.example_auto.com
  SET loc = 'NEW YORK'
  WHERE deptno = 10;
```

ノート:

リモート更新には 1 つ以上のリモート・ノードからデータを取得する副問合せを含めることができますが、更新は 1 つのリモート・ノードでのみ発生するため、その文はリモート更新として分類されます。

親トピック: 分散システムでのトランザクション処理

31.4.2 分散SQL文

分散SQL文は、2つ以上のノード上のデータの問合せまたは変更を行います。

分散問合せ文は、複数のノードから情報を取得します。たとえば、次の問合せは、ローカル・データベースのデータと同時にリモートのsalesデータベースのデータにもアクセスします。

```
SELECT ename, dname
  FROM scott.emp e, scott.dept@sales.us.americas.example_auto.com d
  WHERE e.deptno = d.deptno;
```

分散更新文は、複数のノードのデータを変更します。分散更新を行うには、プロシージャやトリガーなど、異なるノードのデータにアクセスする複数のリモート更新を含むPL/SQLサブプログラム・ユニットを使用します。たとえば、次のPL/SQLプログラム・ユニットは、ローカル・データベースとリモートのsalesデータベースにある表を更新します。

```
BEGIN
  UPDATE scott.dept@sales.us.americas.example_auto.com
    SET loc = 'NEW YORK'
    WHERE deptno = 10;
  UPDATE scott.emp
    SET deptno = 11
    WHERE deptno = 10;
END;
COMMIT;
```

データベースによってプログラム内の文がリモート・ノードに送られると、それらの文の実行はユニットとして成功または失敗します。

親トピック: 分散システムでのトランザクション処理

31.4.3 リモート文と分散型の文の共有SQL

共有SQLを使用したリモート文または分散型の文の仕組みは、本質的にはローカル文の仕組みと同じです。

SQLテキストは必ず一致している必要があり、参照先のオブジェクトも必ず一致している必要があります。可能であれば、任意の文または分解された問合せをローカルまたはリモートで処理するために共有SQL領域を使用できます。

関連項目:

共有SQLの詳細は、『Oracle Database概要』を参照してください。

親トピック: 分散システムでのトランザクション処理

31.4.4 リモート・トランザクション

リモート・トランザクションには1つ以上のリモート文があり、そのすべてが1つのリモート・ノードを参照しています。

たとえば、次のトランザクションには2つの文があり、それぞれがリモートのsalesデータベースにアクセスします。

```
UPDATE scott.dept@sales.us.americas.example_auto.com
  SET loc = 'NEW YORK'
  WHERE deptno = 10;
UPDATE scott.emp@sales.us.americas.example_auto.com
  SET deptno = 11
  WHERE deptno = 10;
COMMIT;
```

親トピック: 分散システムでのトランザクション処理

31.4.5 分散トランザクション

分散トランザクションとは、1つ以上の文からなり、それらが個別に、またはグループとして、分散データベースの複数のノードのデータを更新するようなトランザクションのことです。

たとえば、このトランザクションは、ローカル・データベースとリモートのsalesデータベースを更新します。

```
UPDATE scott.dept@sales.us.americas.example_auto.com
  SET loc = 'NEW YORK'
  WHERE deptno = 10;
UPDATE scott.emp
  SET deptno = 11
  WHERE deptno = 10;
COMMIT;
```

ノート:

トランザクションのすべての文が 1 つのリモート・ノードのみを参照している場合、そのトランザクションは分散トランザクションではなくリモート・トランザクションです。

親トピック: 分散システムでのトランザクション処理

31.4.6 2フェーズ・コミット・メカニズム

データベースの2フェーズ・コミット・メカニズムは、分散トランザクションに関係するすべてのデータベース・サーバーが、そのトランザクション内の文のすべてをコミットするか、すべてをロールバックするかのどちらか一方のみになるように保証するものです。

データベースは、トランザクションが分散か非分散かにかかわらず、トランザクション内のすべての文がユニットとしてコミットまたはロールバックすることを保証します。実行中のトランザクションの結果は、すべてのノードの全トランザクションに対して不可視であり、このような透過性は、問合せや更新、リモート・プロシージャ・コールなど、あらゆるタイプの操作を含むトランザクションにおいて成り立つことが必要です。

非分散データベースにおけるトランザクション管理の一般的なメカニズムの詳細は、『Oracle Database概要』を参照してください。分散データベースでは、データベースはネットワーク上で同じ特性を持つトランザクション管理を連携させる必要があり、ネットワークやシステムに障害が発生しても、データ整合性を維持する必要があります。

また、2フェーズ・コミット・メカニズムにより、整合性制約、リモート・プロシージャ・コールおよびトリガーによって実行される暗黙的なデータ操作言語(DML)操作が保護されます。

関連項目:

Oracle Databaseの2フェーズ・コミット・メカニズムの詳細は、「分散トランザクションの概念」

親トピック: 分散システムでのトランザクション処理

31.4.7 データベース・リンクの名前解決

SQL文にグローバル・オブジェクト名への参照が含まれていると、データベースでは、必ずグローバル・オブジェクト名の中で指定されているデータベース名と一致する名前を持つデータベース・リンクを検索します。

- データベース・リンクの名前解決について
  グローバル・オブジェクト名は、データベース・リンクを使用して指定されるオブジェクトです。
- グローバル・データベース名が完全なときの名前解決
  SQL文に完全なグローバル・データベース名が含まれる場合、データベースは、指定されたグローバル・データベース名に一致するリンクのみを検索します。
- グローバル・データベース名が部分的なときの名前解決
  ドメインのいずれかの部分が指定されている場合、データベースは、完全なグローバル・データベース名が指定されていると仮定します。
- グローバル・データベース名をまったく指定しないときの名前解決
  グローバル・オブジェクト名がローカル・データベース内のオブジェクトを参照し、データベース・リンク名が@記号を使用して指定されていない場合、データベースはオブジェクトがローカルであることを自動的に検出し、オブジェクト参照を解決するためにデータベース・リンクを検索または使用しません。
- 名前解決のための検索の終了
  データベースは、最初の一致が見つかったとき、必ずしも一致するデータベース・リンクの検索を停止しません。データベースは、リモート・データベースへの完全なパス(リモート・アカウントとサービス名の両方)がわかるまで、一致するプライベート、パブリックおよびネットワークのデータベース・リンクを検索します。

親トピック: 分散システムでのトランザクション処理

31.4.7.1 データベース・リンクの名前解決について

グローバル・オブジェクト名とは、データベース・リンクを使用して指定されたオブジェクトのことです。

グローバル・オブジェクト名の必須構成要素は、次のとおりです。

- オブジェクト名
- データベース名
- ドメイン

次の表は、明示的に指定されるグローバル・データベース・オブジェクト名の構成要素を示しています。

文	オブジェクト	データベース	ドメイン
SELECT * FROM joan.dept@sales.example.com	dept	sales	example.com
SELECT * FROM emp@mktg.us.example.com	emp	mktg	us.example.com

SQL文にグローバル・オブジェクト名への参照が含まれていると、データベースでは、必ずグローバル・オブジェクト名の中で指定されているデータベース名と一致する名前を持つデータベース・リンクを検索します。たとえば、次の文を発行した場合を考えます。

```
SELECT * FROM scott.emp@orders.us.example.com;
```

この場合は、データベースによって`orders.us.example.com`というデータベース・リンクが検索されます。データベースはこの操作を実行して、指定されたリモート・データベースへのパスを判断します。

データベースは、一致するデータベース・リンクを常に次の順序で検索します。

1. SQL文を発行したユーザーのスキーマ内のプライベート・データベース・リンク。
2. ローカル・データベース内のパブリック・データベース・リンク。
3. グローバル・データベース・リンク(ディレクトリ・サーバーが使用可能な場合のみ)。

親トピック: データベース・リンクの名前解決

31.4.7.2 グローバル・データベース名が完全なときの名前解決

SQL文に完全なグローバル・データベース名が含まれる場合、データベースは、指定されたグローバル・データベース名に一致するリンクのみを検索します。

完全なグローバル・データベース名が指定された次のSQL文を発行するとします。

```
SELECT * FROM emp@prod1.us.example.com;
```

この場合は、データベース名(`prod1`)とドメイン構成要素(`us.example.com`)の両方を指定しているため、データベースは、プライベート、パブリックおよびグローバルのデータベース・リンクを検索します。

親トピック: データベース・リンクの名前解決

31.4.7.3 グローバル・データベース名が部分的なときの名前解決

ドメインのいずれかの部分が指定されている場合、データベースは、完全なグローバル・データベース名が指定されていると仮定します。

SQL文で部分的なグローバル・データベース名を指定した場合(つまり、データベース構成要素のみを指定した場合)、データベースは`DB_DOMAIN`初期化パラメータ値を`DB_NAME`初期化パラメータ値の後に追加し、完全な名前を構成します。たとえば、次の文を発行した場合を考えます。

```
CONNECT scott@locdb
SELECT * FROM scott.emp@orders;
```

`locdb`のネットワーク・ドメインが`us.example.com`の場合、データベースはこのドメインを`orders`の後に追加し、`orders.us.example.com`という完全なグローバル・データベース名を構成します。データベースは、構築されたグローバル名のみに一致するデータベース・リンクを検索します。一致するリンクが見つからない場合、データベースはエラーを返し、SQL文は実行できません。

親トピック: データベース・リンクの名前解決

31.4.7.4 グローバル・データベース名をまったく指定しないときの名前解決

グローバル・オブジェクト名がローカル・データベース内のオブジェクトを参照していて、データベース・リンク名の指定に@記号が使用されていない場合、データベースは、オブジェクトがローカルであることを自動的に検出して検索を行わないか、またはデータベース・リンクを使用してオブジェクト参照を解決します。

たとえば、次の文を発行した場合を考えます。

```
CONNECT scott@locdb
SELECT * from scott.emp;
```

2番目の文ではデータベース・リンク接続文字列を使用してグローバル・データベース名を指定していないため、データベースは、データベース・リンクを検索しません。

親トピック: データベース・リンクの名前解決

31.4.7.5 名前解決のための検索の終了

データベースは、最初に一致したデータベース・リンクが見つかったときに、必ずしも一致するデータベース・リンクの検索を停止するとはかぎりません。データベースは、リモート・データベースへの完全なパス(リモート・アカウントとサービス名の両方)がわかるまで、一致するプライベート、パブリックおよびネットワークのデータベース・リンクを検索します。

最初に一致したデータベース・リンクが見つかると、次の表に従ってリモート・スキーマが決定されます。

ユーザー操作	データベースの応答	例
CONNECT 句は指定しないでください	接続ユーザー・データベース・リンクを使用します。	`CREATE DATABASE LINK k1 USING 'prod'`
CONNECT TO ... IDENTIFIED BY 句を指定してください	固定ユーザー・データベース・リンクを使用します。	`CREATE DATABASE LINK k2 CONNECT TO scott IDENTIFIED BY password USING 'prod'`
CONNECT TO CURRENT_USER 句が指定されている場合	現行ユーザー・データベース・リンクを使用します。	`CREATE DATABASE LINK k3 CONNECT TO CURRENT_USER USING 'prod'`
USING 句は指定しないでください	データベース文字列を指定するリンクが見つかるまで検索します。一致するデータベース・リンクが見つかっても文字列がまったく識別されない場合、データベースはエラーを返します。	`CREATE DATABASE LINK k4 CONNECT TO CURRENT_USER`

完全なパスを決定した後、データベースはリモート・セッションを作成します(同じローカル・セッションのために同一の接続がまだオープンになっていない場合)。セッションがすでに存在している場合、データベースはそのセッションを再利用します。

親トピック: データベース・リンクの名前解決

31.4.8 スキーマ・オブジェクトの名前解決

Oracleデータベースがリモート・データベースに接続したときのリモート・スキーマの決定方法を理解するのは重要なことです。

- スキーマ・オブジェクトの名前解決
  ローカルのOracle Databaseが、SQL文を発行したローカル・ユーザーのために指定のリモート・データベースに接続した後も、あたかもリモート・ユーザーが対応するSQL文を発行したかのように、オブジェクトの解決処理が続きます。
- グローバル・オブジェクトの名前解決の例: 完全なオブジェクト名
  データベースが完全なグローバル・オブジェクト名を解決し、プライベートおよびパブリックの両方のデータベース・リンクを使用してリモート・データベースへの適切なパスを決定する方法の例を示します。
- グローバル・オブジェクトの名前解決の例: 部分的なオブジェクト名
  データベースが部分的なグローバル・オブジェクト名を解決し、プライベートおよびパブリックの両方のデータベース・リンクを使用してリモート・データベースへの適切なパスを決定する方法の例を示します。

親トピック: 分散システムでのトランザクション処理

31.4.8.1 スキーマ・オブジェクトの名前解決について

SQL文を発行したローカル・ユーザーのかわりに、ローカルのOracle Databaseが指定されたリモート・データベースに接続すると、リモート・ユーザーが関連するSQL文を発行したかのようにオブジェクトの解決が継続されます。

最初に一致したデータベース・リンクが見つかると、次の規則に従ってリモート・スキーマが決定されます。

指定したリンクのタイプ	オブジェクト解決の場所
固定ユーザー・データベース・リンク	リンク作成文で指定されたスキーマ
接続ユーザー・データベース・リンク	接続ユーザーのリモート・スキーマ
現行ユーザー・データベース・リンク	現行ユーザーのスキーマ

オブジェクトが見つからなかった場合、データベースはリモート・データベースのパブリック・オブジェクトをチェックします。オブジェクトを解決できなくても確立されたリモート・セッションは残りますが、SQL文は実行できず、エラーが返されます。

次に、分散データベース・システムにおけるグローバル・オブジェクトの名前解決の例を示します。以降のすべての例で、示しているようなことを想定します。

親トピック: スキーマ・オブジェクトの名前解決

31.4.8.2 グローバル・オブジェクトの名前解決の例: 完全なオブジェクト名

データベースが完全なグローバル・オブジェクト名を解決し、プライベートおよびパブリックの両方のデータベース・リンクを使用してリモート・データベースへの適切なパスを決定する方法の例を示します。

この例では、次のことを想定しています。

- リモート・データベースの名前は、`sales.division3.example.com`です。
- ローカル・データベースの名前は、`hq.division3.example.com`です。
- ディレクトリ・サーバーは使用できません。したがって、グローバル・データベース・リンクも使用できません。

- リモート表empは、スキーマtsmith内にあります。

次の文がscottによってローカル・データベースで発行された場合を考えます。

```
CONNECT scott@hq
CREATE PUBLIC DATABASE LINK sales.division3.example.com
CONNECT TO guest IDENTIFIED BY network
  USING 'dbstring';
```

その後、JWARDが接続して次の文を発行するとします。

```
CONNECT jward@hq
CREATE DATABASE LINK sales.division3.example.com
  CONNECT TO tsmith IDENTIFIED BY radio;
UPDATE tsmith.emp@sales.division3.example.com
  SET deptno = 40
  WHERE deptno = 10;
```

データベースは、最後の文を次のように処理します。

1. データベースは、jwardのUPDATE文で完全なグローバル・オブジェクト名が参照されていると判断します。したがって、一致する名前を持つデータベース・リンクの検索がローカル・データベース内で開始されます。
2. データベースは、一致するプライベート・データベース・リンクをスキーマjward内で検索します。しかし、プライベート・データベース・リンクjward.sales.division3.example.comは、リモートのsalesデータベースへの完全なパスを示しておらず、リモート・アカウントのみを示しています。そのため、データベースは、一致するパブリック・データベース・リンクを続いて検索します。
3. データベースは、パブリック・データベース・リンクをscottのスキーマ内で検索します。データベースは、このパブリック・データベース・リンクからネット・サービス名dbstringを取得します。
4. データベースは、一致するプライベートの固定ユーザー・データベース・リンクから取得したリモート・アカウントを結合して完全なパスを決定し、次に、ユーザーtsmith/radioとしてリモートのsalesデータベースに接続します。
5. これで、リモート・データベースは、emp表へのオブジェクト参照を解決できます。データベースはtsmithスキーマ内で検索を行い、参照先のemp表を検出します。
6. リモート・データベースは文の実行を完了し、その結果をローカル・データベースに返します。

親トピック: スキーマ・オブジェクトの名前解決

31.4.8.3 グローバル・オブジェクトの名前解決の例: 部分的なオブジェクト名

データベースが部分的なグローバル・オブジェクト名を解決し、プライベートおよびパブリックの両方のデータベース・リンクを使用してリモート・データベースへの適切なパスを決定する方法の例を示します。

この例では、次のことを想定しています。

- リモート・データベースの名前は、sales.division3.example.comです。
- ローカル・データベースの名前は、hq.division3.example.comです。
- ディレクトリ・サーバーは使用できません。したがって、グローバル・データベース・リンクも使用できません。
- リモート・データベースsalesの表empはスキーマtsmith内にあり、スキーマscott内にはありません。
- リモート・データベースsalesにempというパブリック・シノニムが存在します。このシノニムは、リモート・データベースsalesのtsmith.empを指しています。

- 「グローバル・オブジェクトの名前解決の例: 完全なオブジェクト名」で示したパブリック・データベース・リンクが、ローカル・データベースhqにすでに作成されています。

```
CREATE PUBLIC DATABASE LINK sales.division3.example.com
  CONNECT TO guest IDENTIFIED BY network
  USING 'dbstring';
```

ローカル・データベースhqで次の文が発行された場合を考えます。

```
CONNECT scott@hq
CREATE DATABASE LINK sales.division3.example.com;
DELETE FROM emp@sales
  WHERE empno = 4299;
```

データベースは、最後のDELETE文を次のように処理します。

1. データベースは、scottのDELETE文で部分的なグローバル・オブジェクト名が参照されていると感知します。ローカル・データベースのドメインを使用して、次のように完全なグローバル・オブジェクト名に拡張します。

```
DELETE FROM emp@sales.division3.example.com
  WHERE empno = 4299;
```

2. データベースは、一致する名前を持つデータベース・リンクをローカル・データベース内で検索します。
3. データベースは、一致するプライベート接続ユーザー・リンクをスキーマscott内で検出しますが、そのプライベート・データベース・リンクにはパスがまったく示されていません。データベースは接続ユーザー名/パスワードをパスのリモート・アカウント部分として使用して、一致するパブリック・データベース・リンクを検索して検出します。

```
CREATE PUBLIC DATABASE LINK sales.division3.example.com
  CONNECT TO guest IDENTIFIED BY network
  USING 'dbstring';
```

4. データベースは、このパブリック・データベース・リンクからサービス名dbstringを取得します。この時点で、データベースは完全なパスを決定しました。
5. データベースは、リモート・データベースにscott/passwordとして接続して検索しますが、スキーマscott内でempというオブジェクトは検出しません。
6. リモート・データベースは、empというパブリック・シノニムを検索して見つけます。
7. リモート・データベースは文を実行し、その結果をローカル・データベースに返します。

親トピック: スキーマ・オブジェクトの名前解決

31.4.9 ビュー、シノニムおよびプロシージャでのグローバル名前解決

グローバル・オブジェクト名は、完全または部分的である可能性があります。

- ビュー、シノニムおよびプロシージャのグローバル名の解決について
  ビュー、シノニムまたはPL/SQLプログラム・ユニット(プロシージャ、ファンクション、トリガーなど)は、グローバル・オブジェクト名によってリモートのスキーマ・オブジェクトを参照できます。
- グローバル名を変更したときに起こる動作
  グローバル名の変更は、部分的なグローバル・オブジェクト名を使用してリモート・データを参照するビュー、シノニムおよびプロシージャに影響を与える可能性があります。
- グローバル名の変更例
  グローバル名の変更のシナリオを示します。

親トピック: 分散システムでのトランザクション処理

31.4.9.1 ビュー、シノニムおよびプロシージャでのグローバル名前解決について

ビュー、シノニムまたはPL/SQLプログラム・ユニット(プロシージャ、ファンクション、トリガーなど)は、グローバル・オブジェクト名によってリモートのスキーマ・オブジェクトを参照できます。

グローバル・オブジェクト名が完全な場合、データベースは、グローバル・オブジェクト名を拡張せずにオブジェクトの定義を格納します。ただし、名前が部分的な場合、データベースはローカル・データベース名のドメインを使用してその名前を拡張します。

次の表は、ビュー、シノニムおよびプログラム・ユニットの部分的なグローバル・オブジェクト名を、データベースがいつ拡張するのかを示したものです。

ユーザー操作	データベースの応答
ビューの作成	部分的なグローバル名は拡張しません。定義内の問合せのテキストがそのままデータ・ディクショナリに格納されます。そのかわりに、ビューを使用する文が解析されるたびに、データベースは、部分的なグローバル・オブジェクト名を拡張します。
シノニムの作成	部分的なグローバル名が拡張されます。データ・ディクショナリに格納されるシノニムの定義には、拡張されたグローバル・オブジェクト名が含まれます。
プログラム・ユニットのコンパイル	部分的なグローバル名が拡張されます。

親トピック: ビュー、シノニムおよびプロシージャでのグローバル名前解決

31.4.9.2 グローバル名を変更したときに起こる動作

グローバル名の変更は、部分的なグローバル・オブジェクト名を使用してリモート・データを参照するビュー、シノニムおよびプロシージャに影響を与える可能性があります。

参照先データベースのグローバル名を変更すると、ビューおよびプロシージャは存在しないデータベースまたは正しくないデータベースを参照しようとすることがあります。ただし、シノニムは実行時にデータベース・リンク名を拡張しないので、何も変わりません。

親トピック: ビュー、シノニムおよびプロシージャでのグローバル名前解決

31.4.9.3 グローバル名の変更例

グローバル名の変更のシナリオを示します。

たとえば、`sales.uk.example.com`および`hq.uk.example.com`という2つのデータベースを考えます。また、`sales`データベースに次のビューとシノニムがあるとします。

```
CREATE VIEW employee_names AS
        SELECT ename FROM scott.emp@hr;
CREATE SYNONYM employee FOR scott.emp@hr;
```

データベースは、`employee`シノニム定義を拡張して、次のように保存します。

`scott.emp@hr.uk.example.com`

- シナリオ1: 両方のデータベース名が変更された場合

シナリオは、両方のグローバル・データベース名が変更される状況を示しています。

- シナリオ2: 一方のデータベース名が変更された場合
  シナリオは、1つのグローバル・データベース名が変更される状況を示しています。

親トピック: ビュー、シノニムおよびプロシージャでのグローバル名前解決

31.4.9.3.1 使用例1: 両方のデータベース名が変更された場合

シナリオは、両方のグローバル・データベース名が変更される状況を示しています。

最初に、営業および人事の両部門が米国に配置替えされるという状況を考えます。その結果、対応するグローバル・データベース名はどちらも次のように変更されます。

- `sales.uk.example.com`は`sales.us.example.com`になります
- `hq.uk.example.com`は`hq.us.example.com`になります

次の表は、グローバル名の変更前および変更後の問合せの拡張を示しています。

salesへの問合せ	変更前の拡張	変更後の拡張
`SELECT * FROM employee_names`	`SELECT * FROM scott.emp@hr.uk.example.com`	`SELECT * FROM scott.emp@hr.us.example.com`
`SELECT * FROM employee`	`SELECT * FROM scott.emp@hr.uk.example.com`	`SELECT * FROM scott.emp@hr.uk.example.com`

親トピック: グローバル名の変更例

31.4.9.3.2 使用例2: 一方のデータベース名が変更された場合

シナリオは、1つのグローバル・データベース名が変更される状況を示しています。

この例では、営業部のみが米国に移動し、人事部は英国に留まるとします。その結果、対応するグローバル・データベース名はどちらも次のように変更されます。

- `sales.uk.example.com`は`sales.us.example.com`になります
- `hq.uk.example.com`は変わりません

次の表は、グローバル名の変更前および変更後の問合せの拡張を示しています。

salesへの問合せ	変更前の拡張	変更後の拡張
`SELECT * FROM employee_names`	`SELECT * FROM scott.emp@hr.uk.example.com`	`SELECT * FROM scott.emp@hr.us.example.com`
`SELECT * FROM employee`	`SELECT * FROM scott.emp@hr.uk.example.com`	`SELECT * FROM scott.emp@hr.uk.example.com`

この場合、`employee_names`ビューを定義している問合せが、存在しないグローバル・データベース名に拡張されます。ただし、`employee`シノニムは、引き続き正しいデータベースである`hq.uk.example.com`を参照します。

親トピック: グローバル名の変更例

31.5 分散データベース・アプリケーションの開発

分散システムでアプリケーション開発を行うと、非分散システムには当てはまらない問題が発生します。

- 分散データベース・システムにおける透過性
  最小限の労力で、システムを使用するユーザーに対してOracle Database分散データベース・システムを透過的にするアプリケーションを開発できます。透過性の目的は、分散データベース・システムを単一のOracle Databaseであるかのように処理することです。その結果、分散データベース・アプリケーション開発を困難なものにし、ユーザーの生産性を損なう恐れのある複雑さがなくなり、システムの開発者やユーザーの負担が軽減します。
- PL/SQLおよびリモート・プロシージャ・コール(RPC)
  開発者は、分散データベースで動作するアプリケーションをサポートするために、PL/SQLパッケージおよびプロシージャをコーディングできます。アプリケーションでは、ローカル・データベースでの処理を実行するためにローカル・プロシージャ・コールを実行でき、リモート・データベースでの処理を実行するためにRPCを実行できます。
- 分散問合せの最適化
  Oracle Databaseの機能の1つである分散問合せの最適化は、トランザクションが分散SQL文内で参照されているリモート表からデータを取得するときに、サイト間で必要となるデータ転送の量を減らします。

関連項目:

分散システム向けアプリケーションの開発方法を学習するには、「分散データベース・システムのアプリケーション開発」

親トピック: 分散データベースの概念

31.5.1 分散データベース・システムにおける透過性

Oracle Database分散データベース・システムを、システムを使用するユーザーに対して透過的にするアプリケーションを最小限の作業で開発できます。透過性の目的は、分散データベース・システムを単一のOracle Databaseであるかのように処理することです。その結果、分散データベース・アプリケーション開発を困難なものにし、ユーザーの生産性を損なう恐れのある複雑さがなくなり、システムの開発者やユーザーの負担が軽減します。

- 位置の透過性
  Oracle Databaseの分散データベース・システムには、アプリケーション開発者および管理者がデータベース・オブジェクトの物理的な位置をアプリケーションおよびユーザーから隠す機能があります。
- SQLおよびCOMMITの透過性
  Oracle Databaseの分散データベース・アーキテクチャは、問合せ、更新およびトランザクションの透過性を提供します。

親トピック: 分散データベース・アプリケーションの開発

31.5.1.1 位置の透過性

Oracle Databaseの分散データベース・システムには、アプリケーション開発者および管理者がデータベース・オブジェクトの物理的な位置をアプリケーションおよびユーザーから隠す機能があります。

ユーザーが、アプリケーションの接続先ノードに関係なく、表などのデータベース・オブジェクトを一様に参照できるのには、位置の透過性が関係しています。位置の透過性には、次のようないくつかの利点があります。

- データベースのユーザーはデータベース・オブジェクトの物理的な位置を知る必要がないため、リモート・データへのアクセスが簡単になります。

- 管理者は、エンド・ユーザーや既存のデータベース・アプリケーションに影響を与えることなく、データベース・オブジェクトを移動できます。

通常、管理者および開発者は、アプリケーション・スキーマ内の表およびサポート・オブジェクトに対する位置の透過性を確立するために、シノニムを使用します。たとえば、次の文は、データベース内に別のリモート・データベース内の表に対応するシノニムを作成します。

```
CREATE PUBLIC SYNONYM emp
  FOR scott.emp@sales.us.americas.example_auto.com;
CREATE PUBLIC SYNONYM dept
  FOR scott.dept@sales.us.americas.example_auto.com;
```

これにより、リモート表にアクセスする際に次のような問合せを使用する必要がなくなります。

```
SELECT ename, dname
  FROM scott.emp@sales.us.americas.example_auto.com e,
       scott.dept@sales.us.americas.example_auto.com d
  WHERE e.deptno = d.deptno;
```

アプリケーションでは、リモート表の位置を考慮せずに、単純な問合せを発行できます。

```
SELECT ename, dname
  FROM emp e, dept d
  WHERE e.deptno = d.deptno;
```

シノニムの他にも、ビューおよびストアド・プロシージャを使用して、分散データベース・システムで動作するアプリケーションに対する位置の透過性を確立できます。

親トピック: 分散データベース・システムにおける透過性

31.5.1.2 SQLおよびCOMMITの透過性

Oracle Databaseの分散データベース・アーキテクチャは、問合せ、更新およびトランザクションの透過性を提供します。

たとえば、SELECT、INSERT、UPDATE、DELETEなどの標準のSQL文は、非分散データベース環境の場合とまったく同じように動作します。また、アプリケーションでは標準SQL文COMMIT、SAVEPOINTおよびROLLBACKを使用してトランザクションを制御します。分散トランザクションを制御するために複雑なプログラミングやその他の特別な操作を行う必要はありません。

- 1つのトランザクション内の文からは任意の数のローカル表またはリモート表を参照できます。
- データベースでは、分散トランザクションに関係するノードがすべて同じ動作をすることが保証されており、それらのノードすべてがトランザクションをコミットまたはロールバックします。
- 分散トランザクションのコミット中にネットワークまたはシステムの障害が発生した場合、トランザクションは自動的かつ透過的、およびグローバルに解決されます。具体的には、ネットワークまたはシステムがリストアされると、すべてのノードがトランザクションをコミットまたはロールバックします。

データベースの内部では、コミットされた各トランザクションに対して、そのトランザクション内の文による変更を一意に識別するためのシステム変更番号(SCN)が対応付けられます。分散データベースでは、次のときに通信中のノードのSCNが調整されます。

- 1つ以上のデータベース・リンクによって表されるパスを使用して接続が確立されるとき。
- 分散SQL文が実行されるとき。
- 分散トランザクションがコミットされるとき。

特に、分散データベース・システムのノード間でSCNが調整されることにより、文とトランザクションの両方のレベルでグローバルな分散読込み一貫性が保証されることは大きな利点です。必要であれば、グローバルな時間ベースの分散リカバリを完了するこ

ともできます。

親トピック: 分散データベース・システムにおける透過性

31.5.2 PL/SQLおよびリモート・プロシージャ・コール(RPC)

開発者は、分散データベースで動作するアプリケーションをサポートするためのPL/SQLパッケージおよびプロシージャをコーディングできます。アプリケーションでは、ローカル・データベースでの処理を実行するためにローカル・プロシージャ・コールを実行でき、リモート・データベースでの処理を実行するためにRPCを実行できます。

プログラムがリモート・プロシージャをコールすると、ローカル・サーバーはすべてのプロシージャ・パラメータをコールされたリモートサーバーに渡します。たとえば、次のPL/SQLプログラム・ユニットでは、リモートの`sales`データベースにあるパッケージ・プロシージャ`del_emp`をコールし、パラメータ1257を渡します。

```
BEGIN
 emp_mgmt.del_emp@sales.us.americas.example_auto.com(1257);
END;
```

RPCを正常に実行するためには、コール側プロシージャがリモート・サイトに存在し、接続しようとするユーザーがそのプロシージャを実行するための適切な権限を持っている必要があります。

分散データベース・システム対応のパッケージおよびプロシージャを開発する際、開発者は、どのプログラム・ユニットをリモートの位置で実行し、その結果をどのようにコール側アプリケーションに返すのかについて理解した上で、コードを記述する必要があります。

親トピック: 分散データベース・アプリケーションの開発

31.5.3 分散問合せの最適化

Oracle Databaseの機能の1つである分散問合せの最適化は、トランザクションの分散SQL文内で参照されているリモート表からデータを取得するときに、サイト間で必要になるデータ転送の量を減らします。

分散問合せの最適化では、コストベース最適化の設定を使用して、リモート表から必要なデータのみを抽出するSQL式が検索または生成され、抽出されたデータはリモート・サイト(場合によってはローカル・サイト)で処理され、その結果がローカル・サイトに送信されて最終処理が行われます。表データすべてをローカル・サイトに転送して処理する際にかかる時間と比較して、この操作では必要なデータ転送量が削減されます。

`DRIVING_SITE`、`NO_MERGE`、`INDEX`など、各種のコストベース・オプティマイザ・ヒントを使用することによって、Oracle Databaseがデータを処理する場所と、データへのアクセス方法を制御できます。

関連項目:

コストベースの最適化の詳細は、「コストベース最適化の使用」

親トピック: 分散データベース・アプリケーションの開発

31.6 分散環境での文字セットのサポート

分散環境では、異なるデータベースおよびクライアントが異なる文字セットを使用できます。

- 分散環境の文字セットのサポートについて
  Oracle Databaseは、クライアント、Oracle Databaseサーバーおよび非Oracle Databaseサーバーで異なる文字セットが使用されている環境をサポートします。異機種環境のためにNCHARのサポートが提供されています。

- クライアント/サーバー環境
  クライアント/サーバー環境では、クライアントの文字セットは、Oracle Databaseサーバーの文字セットと同じにするか、またはそのサブセットに設定します。
- 同機種間分散環境
  非異機種環境では、クライアントとサーバーの文字セットは、メイン・サーバーの文字セットと同じにするか、またはそのサブセットにする必要があります。
- 異機種間分散環境
  異機種環境では、クライアント、Transparent Gatewayおよび非Oracle Databaseデータソースのグローバリゼーション・サポート・パラメータの設定は、データベース・サーバーの文字セットと同じにするか、そのサブセットにする必要があります。

親トピック: 分散データベースの概念

31.6.1 分散環境での文字セットのサポートについて

Oracle Databaseは、クライアント、Oracle DatabaseサーバーおよびOracle Database以外のサーバーが異なる文字セットを使用する環境をサポートしています。異機種環境のためにNCHARのサポートが提供されています。

各国語サポート(NLS)および異機種間サービス(HS)に関連する各種の環境変数および初期化パラメータを設定することにより、異なる文字セット間でのデータ変換を制御できます。

キャラクタの設定は、次のNLSおよびHSパラメータで定義されます。

パラメータ	環境	定義の対象
NLS_LANG（環境変数）	クライアント/サーバー	クライアント
NLS_LANGUAGE NLS_CHARACTERSET NLS_TERRITORY	クライアント/サーバー 非異機種間分散 異機種間分散	Oracle Database サーバー
HS_LANGUAGE	異機種間分散	Oracle 以外のデータベース・サーバー Transparent Gateway
NLS_NCHAR（環境変数） HS_NLS_NCHAR	異機種間分散	Oracle Database サーバー Transparent Gateway

関連項目:

- NLSパラメータの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- HSパラメータの詳細は、『Oracle Database Heterogeneous Connectivityユーザーズ・ガイド』を参照してください

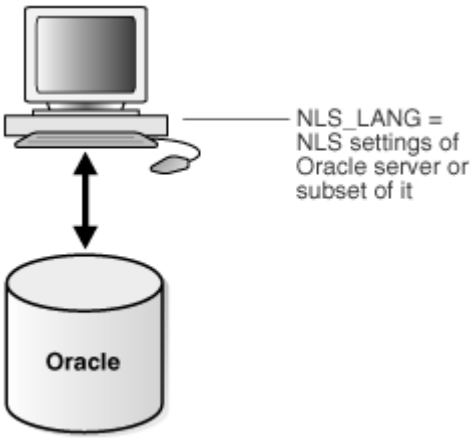
親トピック: 分散環境での文字セットのサポート

31.6.2 クライアント/サーバー環境

クライアント/サーバー環境では、クライアントの文字セットは、Oracle Databaseサーバーの文字セットと同じにするか、またはそのサブセットに設定します。

図31-6は、クライアント/サーバー環境を示しています。

図31-6 クライアント/サーバー環境におけるNLSパラメータの設定



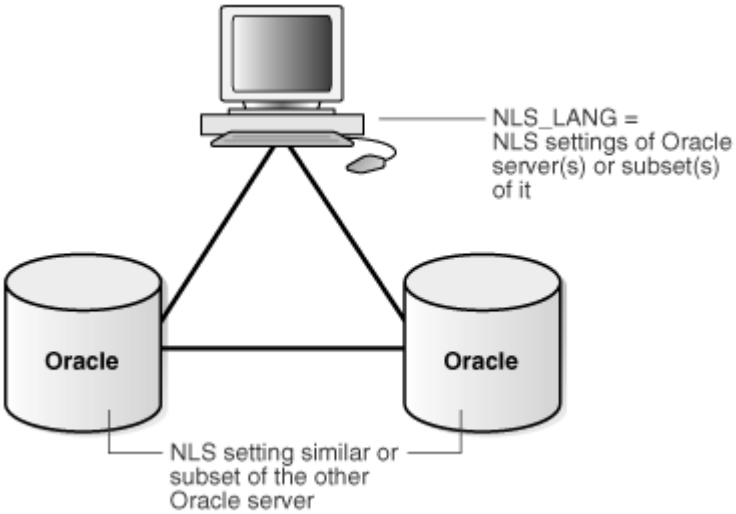
親トピック: 分散環境での文字セットのサポート

31.6.3 同機種間分散環境

非異機種環境では、クライアントとサーバーの文字セットは、メイン・サーバーの文字セットと同じにするか、またはそのサブセットにする必要があります。

図31-7は、同機種間分散環境を示しています。

図31-7 同機種環境におけるNLSパラメータの設定



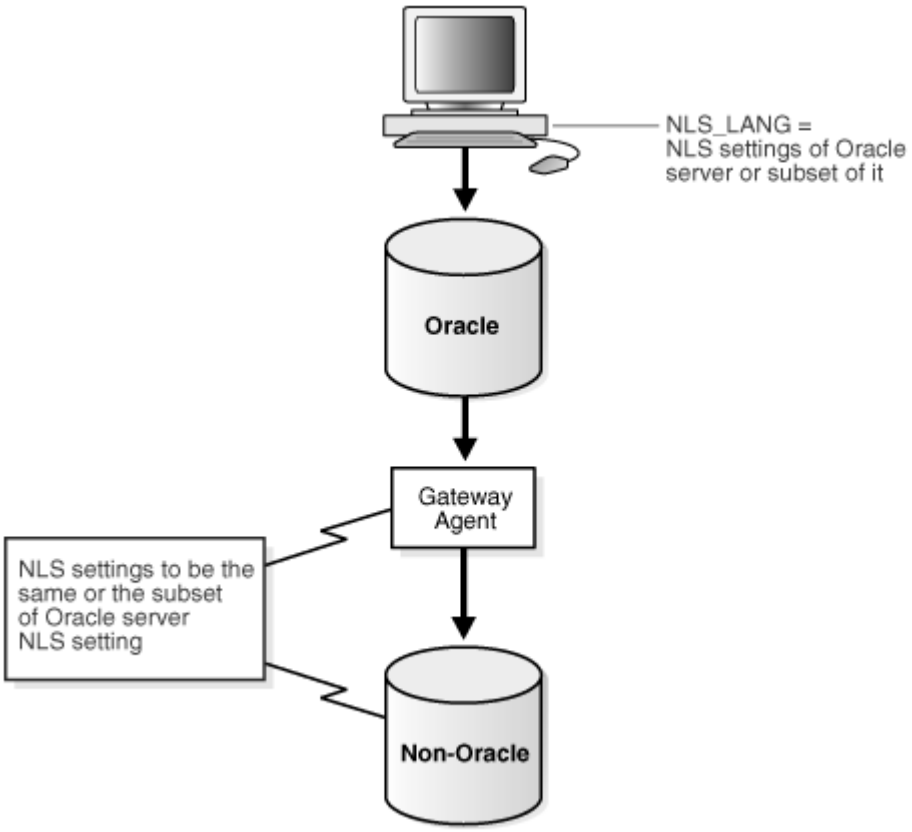
親トピック: 分散環境での文字セットのサポート

31.6.4 異機種間分散環境

異機種環境では、クライアント、Transparent Gatewayおよび非Oracle Databaseデータソースのグローバリゼーション・サポート・パラメータの設定は、データベース・サーバーの文字セットと同じにするか、そのサブセットにする必要があります。

図31-8は、異機種間分散環境を示しています。Transparent Gatewayは、グローバリゼーションを完全にサポートしています。

図31-8 異機種環境におけるNLSパラメータの設定



異機種環境では、異機種間サービス・テクノロジによって構築されたTransparent Gatewayのみが完全なNCHAR機能をサポートします。特定のTransparent GatewayがNCHARをサポートしているかどうかは、対象になるOracle Database以外のデータソースによって異なります。特定のTransparent GatewayによるNCHARサポートの処理方法の詳細は、システム固有のTransparent Gatewayのマニュアルを参照してください。

関連項目:

異機種間サービスの詳細は、『Oracle Database Heterogeneous Connectivityユーザーズ・ガイド』を参照してください。

親トピック: 分散環境での文字セットのサポート

32 分散データベースの管理

分散データベースの管理には、グローバル名の管理、データベース・リンクの管理、位置および文の透過性の作成などのタスクが含まれます。

- 分散システムでのグローバル名の管理
  分散データベース・システムでは、各データベースに一意のグローバル・データベース名が必要です。グローバル・データベース名は、システム内のデータベースを一意に識別します。分散データベースでの主な管理作業として、グローバル・データベース名の作成と変更の管理があります。
- データベース・リンクの作成
  分散データベース・システム全体にわたってデータおよびスキーマ・オブジェクトへのアプリケーション・アクセスをサポートするために、すべての必要なデータベース・リンクを作成する必要があります。
- 共有データベース・リンクの使用
  標準のデータベース・リンクを使用してリモート・サーバーを参照するすべてのアプリケーションは、ローカル・データベースとリモート・データベース間に接続を確立します。多くのユーザーがアプリケーションを同時に実行した場合、ローカル・データベースとリモート・データベースの間で多数の接続が発生する可能性があります。共有データベース・リンクを使用すると、ローカル・サーバーとリモート・サーバーの間で必要なネットワーク接続の数を制限できます。
- データベース・リンクの管理
  データベース・リンクの管理には、それらのクローズ、削除、およびアクティブな接続数の制限などのタスクが含まれます。
- データベース・リンク情報の表示
  各データベースのデータ・ディクショナリには、データベース内のすべてのデータベース・リンクの定義が格納されています。データ・ディクショナリ表およびビューを使用して、リンクに関する情報を取得できます。
- 位置の透過性の作成
  必要なデータベース・リンクを構成した後、データベース・システムの分散特性がユーザーに見えないようにするために、様々なツールを使用できます。言い換えれば、ユーザーはリモート・オブジェクトに対してあたかもローカル・オブジェクトであるかのようにアクセスできるようになります。
- 文の透過性の管理
  分散データベースでは、いくつかのSQL文がリモートの表を参照できます。
- 分散データベースの管理: 例
  データベース・リンクの管理の例を示します。

親トピック: 分散データベースの管理

32.1 分散システムでのグローバル名の管理

分散データベース・システムでは、各データベースが一意のグローバル・データベース名を持ちます。グローバル・データベース名は、システム内のデータベースを一意に識別します。分散データベースでの主な管理作業として、グローバル・データベース名の作成と変更の管理があります。

- グローバル・データベース名の形成方法の理解
  グローバル・データベース名は、データベース名とドメインの2つの構成要素から形成されています。
- グローバル・ネーミング施工の判断
  ローカル・データベースのリンクに付ける名前は、ローカル・データベースがグローバル・ネーミングを施行するかどうかによって異なります。
- グローバル・データベース名の参照

データベースのグローバル名を参照するには、データ・ディクショナリ・ビューGLOBAL_NAMEを使用します。

- グローバル・データベース名のドメインの変更
  データベースのグローバル名のドメインを変更するには、ALTER DATABASE文を使用します。
- グローバル・データベース名の変更: シナリオ
  グローバル・データベース名の変更を示すシナリオです。

親トピック: 分散データベースの管理

32.1.1 グローバル・データベース名の書式の理解

グローバル・データベース名は、データベース名とドメインの2つの構成要素から構成されます。

データベース名とドメイン名は、データベースの作成時に次の初期化パラメータによって決まります。

コンポーネント	パラメータ	要件	例
データベース名	DB_NAME	30 文字以下である必要があります。	sales
データベースが存在するドメイン	DB_DOMAIN	インターネットの標準規則に従う必要があります。ドメイン名のレベルはドットで区切る必要があり、ドメイン名の順序は、左から右に向かってリーフからルートの順になります。	us.example.com

次に、有効なグローバル・データベース名の例を示します。

DB_NAME	DB_DOMAIN	グローバル・データベース名
sales	example.com	sales.example.com
sales	us.example.com	sales.us.example.com
mktg	us.example.com	mktg.us.example.com
payroll	example.org	payroll.example.org

DB_DOMAIN初期化パラメータが重要になるのは、データベースの作成時に、この初期化パラメータを(DB_NAMEパラメータと一緒に)使用して、データベースのグローバル名を構成するときだけです。この時点で、データベースのグローバル名はデータ・ディクショナリに格納されます。グローバル名を変更するには、初期化パラメータ・ファイル内のDB_DOMAINパラメータを変更するのではなく、ALTER DATABASE 文を使用する必要があります。ただし、次回のデータベースの起動を行う前にDB_DOMAINパラメータを変更してドメイン名の変更を反映することをお薦めします。

親トピック: 分散システムでのグローバル名の管理

32.1.2 グローバル・ネーミング施行の判断

ローカル・データベースのリンクに付ける名前は、ローカル・データベースがグローバル・ネーミングを施行するかどうかによって異なります。

ローカル・データベースがグローバル・ネーミングを施行する場合は、リモート・データベースのグローバル・データベース名をリンクの名前として使用する必要があります。たとえば、ローカルのhqサーバーに接続していて、リモートのmfgデータベースへのリンクを作成する場合は、ローカル・データベースがグローバル・ネーミングを施行していれば、mfgのグローバル・データベース名をリンク名として使用する必要があります。

データベース・リンク名の一部としてサービス名を使用することもできます。たとえば、サービス名sn1およびsn2を使用してデータベースhq.example.comに接続する場合、グローバル・ネーミングが施行されていれば、次のようなhqへのリンク名を作成できます。

- HQ.EXAMPLE.COM@SN1
- HQ.EXAMPLE.COM@SN2

関連項目:

リンク名でのサービス名の使用の詳細は、「リンク名に含まれるサービス名を指定するための接続修飾子の使用」を参照してください

グローバル・ネーミングがデータベースで施行されているかどうかを判断するには、データベースの初期化パラメータ・ファイルを調べるか、またはV$PARAMETERビューを問い合せます。たとえば、mfgでグローバル・ネーミングが施行されているかどうかを判断するには、mfgのセッションを開始して、次のglobalnames.sqlスクリプトを作成し、実行できます(出力例も含まれています)。

```
COL NAME FORMAT A12
COL VALUE FORMAT A6
SELECT NAME, VALUE FROM V$PARAMETER
   WHERE NAME = 'global_names'
/
SQL> @globalnames
NAME          VALUE
------------ ------
global_names FALSE
```

親トピック: 分散システムでのグローバル名の管理

32.1.3 グローバル・データベース名の参照

データベースのグローバル名を参照するには、データ・ディクショナリ・ビューGLOBAL_NAMEを使用します。

たとえば、次の文を発行します。

```
SELECT * FROM GLOBAL_NAME;
GLOBAL_NAME
--------------------------------------------------------------------------------
SALES.EXAMPLE.COM
```

親トピック: 分散システムでのグローバル名の管理

32.1.4 グローバル・データベース名のドメインの変更

データベースのグローバル名のドメインを変更するには、ALTER DATABASE文を使用します。

データベースを作成した後に初期化パラメータDB_DOMAINを変更しても、グローバル・データベース名やデータベース・リンク名の解決には効果がありません。

次の例は、名前変更文の構文を示しています。ここで、databaseはデータベース名、domainはネットワーク・ドメインを表します。

```
ALTER DATABASE RENAME GLOBAL_NAME TO database.domain;
```

グローバル・データベース名のドメインを変更するには、次の手順を実行します。

1. 現行のグローバル・データベース名を確認します。たとえば、次のように入力します。

```
SELECT * FROM GLOBAL_NAME;
GLOBAL_NAME
--------------------------------------------------------------------------------
SALES.EXAMPLE.COM
```

2. ALTER DATABASE文を使用して、グローバル・データベース名を変更します。たとえば、次のように入力します。

```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.us.example.com;
```

3. GLOBAL_NAME表を問い合せて、新しい名前を確認します。たとえば、次のように入力します。

```
SELECT * FROM GLOBAL_NAME;
GLOBAL_NAME
--------------------------------------------------------------------------------
SALES.US.EXAMPLE.COM
```

親トピック: 分散システムでのグローバル名の管理

32.1.5 グローバル・データベース名の変更: 使用例

グローバル・データベース名の変更を示すシナリオです。

この例では、ローカル・データベースのグローバル・データベース名のドメイン部分を変更します。また、部分指定のグローバル名を使用してデータベース・リンクを作成し、Oracle Databaseがその名前をどのように解決するのかを調べます。データベースは、初期化パラメータDB_DOMAINの値ではなく、ローカル・データベースの現行のグローバル・データベース名のドメイン部分を使用して部分名を解決することがわかります。

1. SALES.US.EXAMPLE.COMに接続してGLOBAL_NAMEデータ・ディクショナリ・ビューを問い合せ、現行のデータベース・グローバル名を確認します。

```
CONNECT SYSTEM@sales.us.example.com
SELECT * FROM GLOBAL_NAME;
GLOBAL_NAME
--------------------------------------------------------------------------------
SALES.US.EXAMPLE.COM
```

2. V$PARAMETERビューを問い合せ、DB_DOMAIN初期化パラメータの現在の設定を調べます。

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'db_domain';
NAME       VALUE
---------  -----------
db_domain  US.EXAMPLE.COM
```

3. 部分指定のグローバル名のみを使用して、hqデータベースへのデータベース・リンクを作成します。

```
CREATE DATABASE LINK hq USING 'sales';
```

データベースは、リンクで指定されたデータベース名にローカル・データベースのグローバル・データベース名のドメイン部分を追加して、このリンクのグローバル・データベース名を拡張します。

4. USER_DB_LINKSを問い合せて、データベースが部分指定のグローバル・データベース名を解決するために使用したドメイン名を確認します。

```
SELECT DB_LINK FROM USER_DB_LINKS;
DB_LINK
------------------
HQ.US.EXAMPLE.COM
```

この結果は、ローカル・データベースのグローバル・データベース名のドメイン部分がus.example.comであることを示しています。データベースは、データベース・リンク作成時の部分的なデータベース・リンク名を解決するために、このドメインを使用します。

5. salesデータベースが日本に移動するという通知を受けたので、salesデータベースをsales.jp.example.comに変更します。

```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.jp.example.com;
SELECT * FROM GLOBAL_NAME;
GLOBAL_NAME
--------------------------------------------------------------------------------
SALES.JP.EXAMPLE.COM
```

6. 再度V$PARAMETERを問い合せて、グローバル・データベース名のドメイン部分の名前を変更しても、DB_DOMAINの値が変更されていないことを確認します。

```
SELECT NAME, VALUE FROM V$PARAMETER
  WHERE NAME = 'db_domain';
NAME       VALUE
---------  -----------
db_domain  US.EXAMPLE.COM
```

この結果は、DB_DOMAIN初期化パラメータの値がALTER DATABASE RENAME GLOBAL_NAME文とは関係がないことを示しています。ALTER DATABASE文は、DB_DOMAIN初期化パラメータではなく、グローバル・データベース名のドメインを決定します(それでも、新しいドメイン名が反映されるようにDB_DOMAINを変更することには意味があります)。

7. データベースsupplyへの別のデータベース・リンクを作成してから、USER_DB_LINKSを問い合せて、データベースがsupplyのグローバル・データベース名のドメイン部分をどのように解決するのかを確認します。

```
CREATE DATABASE LINK supply USING 'supply';
SELECT DB_LINK FROM USER_DB_LINKS;
DB_LINK
------------------
HQ.US.EXAMPLE.COM
SUPPLY.JP.EXAMPLE.COM
```

この結果は、データベースがドメインjp.example.comを使用して、部分指定のリンク名を解決していることを示しています。このドメインは、ローカル・データベースのグローバル・データベース名のドメイン部分であることから、リンクを作成するときに使用されています。データベースは部分的なリンク名を解決するときに、DB_DOMAIN初期化パラメータの設定は使用しません。

8. 次に、前の情報が誤りで、salesはjp.example.comドメインではなくasia.jp.example.comドメインにあるという通知を受けました。そのため、グローバル・データベース名を次のように変更します。

```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.asia.jp.example.com;
SELECT * FROM GLOBAL_NAME;
```

```
GLOBAL_NAME
--------------------------------------------------------------------------------
SALES.ASIA.JP.EXAMPLE.COM
```

9. 再びV$PARAMETERを問い合せて、パラメータDB_DOMAINの設定を確認します。

```
SELECT NAME, VALUE FROM V$PARAMETER
  WHERE NAME = 'db_domain';
NAME         VALUE
----------   -----------
db_domain    US.EXAMPLE.COM
```

この結果は、パラメータ・ファイル内のドメイン設定が、2つのALTER DATABASE RENAME文の発行前の時点の設定と同じであることを示しています。

10. 最後に、warehouseデータベースへのリンクを作成し、再度USER_DB_LINKSを問い合せて、データベースが部分指定のグローバル名をどのように解決するのかを調べます。

```
CREATE DATABASE LINK warehouse USING 'warehouse';
SELECT DB_LINK FROM USER_DB_LINKS;
DB_LINK
------------------
HQ.US.EXAMPLE.COM
SUPPLY.JP.EXAMPLE.COM
WAREHOUSE.ASIA.JP.EXAMPLE.COM
```

ここでもデータベースは、リンクの作成時にローカル・データベースのグローバル・データベース名のドメイン部分を使用して、部分的なリンク名を拡張していることがわかります。

ノート:

supply データベース・リンクを訂正するには、必ずこのデータベース・リンクを削除してから再作成してください。

関連項目:

- DB_NAME初期化パラメータの指定の詳細は、『Oracle Databaseリファレンス』を参照してください
- DB_DOMAIN初期化パラメータの指定の詳細は、『Oracle Databaseリファレンス』を参照してください

親トピック: 分散システムでのグローバル名の管理

32.2 データベース・リンクの作成

アプリケーションによるデータおよびスキーマ・オブジェクトへのアクセスを分散データベース・システム全体にわたってサポートするために、必要なデータベース・リンクをすべて作成する必要があります。

- データベース・リンクの作成に必要な権限の取得
  データベース・リンクは、リモート・データベース上のオブジェクトにアクセスできるようにするローカル・データベース内のポインタです。プライベート・データベース・リンクを作成するには、あらかじめ適切な権限が付与されている必要があります。
- リンク・タイプの指定
  データベース・リンクを作成するときは、そのデータベース・リンクにアクセスするユーザーを決める必要があります。
- リンク・ユーザーの指定

データベース・リンクは、データベース間の通信パスを定義します。アプリケーションがデータベース・リンクを使用してリモート・データベースに接続するとき、Oracle Databaseはローカル・アプリケーションの要求にかわってリモート・データベース内でデータベース・セッションを確立します。プライベートまたはパブリックのデータベース・リンクを作成する際、固定ユーザー、現行ユーザーおよび接続ユーザーのデータベース・リンクを作成することで、リモート・データベースのどのスキーマにリンクが接続を確立するのかを指定できます。

- リンク名に含まれるサービス名を指定するための接続修飾子の使用
  いくつかの状況では、同じリモート・データベースを指す同じタイプ(パブリックなど)の複数のデータベース・リンクが必要であるが、異なる通信経路を使用してリモート・データベースへの接続を確立できるようにする場合があります。

親トピック: 分散データベースの管理

32.2.1 データベース・リンクの作成に必要な権限の取得

データベース・リンクは、リモート・データベースのオブジェクトへのアクセスを可能にするローカル・データベース内のポインタです。プライベート・データベース・リンクを作成するには、あらかじめ適切な権限が付与されている必要があります。

次の表は、どのリンク・タイプのデータベースに対してどの権限が必要なのかを示しています。

権限	データベース	この権限を必要とする操作
CREATE DATABASE LINK	ローカル	プライベート・データベース・リンクの作成。
CREATEPUBLICDATABASELINK	ローカル	パブリック・データベース・リンクの作成。
CREATE SESSION	リモート	任意タイプのデータベース・リンクの作成。

現在使用可能な権限を確認するには、ROLE_SYS_PRIVSを問い合せます。たとえば、次のprivs.sqlスクリプトを作成し、実行できます(出力例も含まれています)。

```
SELECT DISTINCT PRIVILEGE AS "Database Link Privileges"
FROM ROLE_SYS_PRIVS
WHERE PRIVILEGE IN ( 'CREATE SESSION','CREATE DATABASE LINK',
                     'CREATE PUBLIC DATABASE LINK')
/
SQL> @privs
Database Link Privileges
----------------------------------------
CREATE DATABASE LINK
CREATE PUBLIC DATABASE LINK
CREATE SESSION
```

親トピック: データベース・リンクの作成

32.2.2 リンク・タイプの指定

データベース・リンクを作成するときは、そのデータベース・リンクにアクセスするユーザーを決める必要があります。

- プライベート・データベース・リンクの作成
  プライベート・データベース・リンクを作成するには、CREATE DATABASE LINK文を使用します。
- パブリック・データベース・リンクの作成
  パブリック・データベース・リンクを作成するには、CREATE PUBLIC DATABASE LINK文を使用します。

- グローバル・データベース・リンクの作成
  ネット・サービス名によってデータベースを識別するディレクトリ・サーバーを使用できます。このマニュアルでは、ディレクトリ・サーバーでのネット・サービス名によるデータベースの識別をグローバル・データベース・リンクと呼びます。

親トピック: データベース・リンクの作成

32.2.2.1 プライベート・データベース・リンクの作成

プライベート・データベース・リンクを作成するには、CREATE DATABASE LINK文を使用します。

プライベート・データベース・リンクを作成するには、次の文を指定します。ここで、link_nameはグローバル・データベース名または任意のリンク名を表します。

```
CREATE DATABASE LINK link_name ...;
```

プライベート・データベース・リンクの例を次に示します。

SQL文	結果
`CREATE DATABASE LINK supply.us.example.com;`	リモートの supply データベースへの、グローバル・データベース名を使用したプライベート・リンク。 リンクは、接続ユーザーのユーザーID/パスワードを使用します。そのため、scott(パスワード: password)が問合せの中でこのリンクを使用すると、リモート・データベースへの接続が scott/password として確立されます。
`CREATE DATABASE LINK link_2 CONNECT TO jane IDENTIFIED BY password USING 'us_supply';`	サービス名 us_supply を持つデータベースへの、link_2 という名前のプライベート固定ユーザー・リンク。このリンクは、接続ユーザーとは無関係に、jane/password というユーザーID/パスワードでリモート・データベースに接続します。
`CREATE DATABASE LINK link_1 CONNECT TO CURRENT_USER USING 'us_supply';`	サービス名 us_supply を持つデータベースへの、link_1 という名前のプライベート・リンク。このリンクは、現行ユーザーのユーザーID/パスワードを使用してリモート・データベースにログインします。 ノート: 現行ユーザーと接続ユーザーは異なる場合があります。また、現行ユーザーは、リンクに関係している両方のデータベースのグローバル・ユーザーであることが必要です(「データベース・リンクのユーザー」を参照)。現行ユーザー・リンクは、Oracle Advanced Security オプションの一部です。

関連項目:

CREATE DATABASE LINKの構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: リンク・タイプの指定

32.2.2.2 パブリック・データベース・リンクの作成

パブリック・データベース・リンクを作成するには、CREATE PUBLIC DATABASE LINK文を使用します。

パブリック・データベース・リンクを作成するには、キーワードPUBLICを使用します。ここで、link_nameはグローバル・データベース名または任意のリンク名を表します。

```
CREATE PUBLIC DATABASE LINK link_name ...;
```

パブリック・データベース・リンクの例を次に示します。

SQL文	結果
`CREATE PUBLIC DATABASE LINK supply.us.example.com;`	リモートの supply データベースへのパブリック・リンク。リンクは、接続ユーザーのユーザーID/パスワードを使用します。そのため、scott(パスワード: password)が問合せの中でこのリンクを使用すると、リモート・データベースへの接続が scott/password として確立されます。
`CREATE PUBLIC DATABASE LINK pu_link CONNECT TO CURRENT_USER USING 'supply';`	サービス名 supply を持つデータベースへの、pu_link という名前のパブリック・リンク。このリンクは、現行ユーザーのユーザーID/パスワードを使用してリモート・データベースにログインします。 ノート: 現行ユーザーと接続ユーザーは異なる場合があります。また、現行ユーザーは、リンクに関係している両方のデータベースのグローバル・ユーザーであることが必要です(「データベース・リンクのユーザー」を参照)。
`CREATE PUBLIC DATABASE LINK sales.us.example.com CONNECT TO jane IDENTIFIED BY password;`	リモートの sales データベースへのパブリック固定ユーザー・リンク。このリンクは、jane/password というユーザーID/パスワードでリモート・データベースに接続します。

関連項目:

CREATE PUBLIC DATABASE LINKの構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: リンク・タイプの指定

32.2.2.3 グローバル・データベース・リンクの作成

データベースがネット・サービス名によって識別されるディレクトリ・サーバーを使用できます。このマニュアルでは、ディレクトリ・サーバーでのネット・サービス名によるデータベースの識別をグローバル・データベース・リンクと呼びます。

グローバル・データベース・リンクとして動作するディレクトリ・エントリの作成方法を学習するには、『Oracle Database Net Services管理者ガイド』を参照してください。

親トピック: リンク・タイプの指定

32.2.3 リンク・ユーザーの指定

データベース・リンクは、あるデータベースから別のデータベースへの通信経路を定義します。アプリケーションがデータベース・リンクを使用してリモート・データベースに接続するとき、Oracle Databaseはローカル・アプリケーションの要求にかわってリモート・データベース内でデータベース・セッションを確立します。プライベートまたはパブリックのデータベース・リンクを作成する際、固定ユーザー、現行ユーザーおよび接続ユーザーのデータベース・リンクを作成することで、リモート・データベースのどのスキーマにリンクが接続を確立するのかを指定できます。

- 固定ユーザー・データベース・リンクの作成
  アプリケーションで固定ユーザー・データベース・リンクを使用するとき、ローカル・サーバーは必ずリモート・データベース内の固定リモート・スキーマへの接続を確立します。また、ローカル・サーバーはアプリケーションがリンクを使用してリモート・データベースにアクセスするときに、ネットワークを介して固定ユーザーの資格証明を送信します。
- 接続ユーザーおよび現行ユーザー・データベース・リンクの作成
  接続ユーザーおよび現行ユーザー・データベース・リンクでは、リンク定義に資格証明が含まれません。リモート・データベースへの接続に使用する資格証明は、データベース・リンクを参照するユーザーや、アプリケーションが実行する操作によって異なります。

親トピック: データベース・リンクの作成

32.2.3.1 固定ユーザー・データベース・リンクの作成

アプリケーションで固定ユーザー・データベース・リンクを使用するとき、ローカル・サーバーは必ずリモート・データベース内の固定リモート・スキーマへの接続を確立します。また、ローカル・サーバーはアプリケーションがリンクを使用してリモート・データベースにアクセスするときに、ネットワークを介して固定ユーザーの資格証明を送信します。

固定ユーザー・データベース・リンクを作成するには、リモート・データベースへのアクセスに必要な資格証明(この場合はユーザー名とパスワード)をリンク定義に埋め込みます。

```
CREATE DATABASE LINK ... CONNECT TO username IDENTIFIED BY password ...;
```

固定ユーザー・データベース・リンクの例を次に示します。

SQL文	結果
`CREATE PUBLIC DATABASE LINK supply.us.example.com CONNECT TO scott IDENTIFIED BY password;`	リモートの `supply` データベースへの、グローバル・データベース名を使用したパブリック・リンク。このリンクは、`scott/password` というユーザーID/パスワードでリモート・データベースに接続します。
`CREATE DATABASE LINK foo CONNECT TO jane IDENTIFIED BY password USING 'finance';`	サービス名 `finance` を持つデータベースへの、`foo` という名前のプライベート固定ユーザー・リンク。このリンクは、`jane/password` というユーザーID/パスワードでリモート・データベースに接続します。

親トピック: リンク・ユーザーの指定

32.2.3.2 接続ユーザーおよび現行ユーザー・データベース・リンクの作成

接続ユーザーおよび現行ユーザー・データベース・リンクでは、リンク定義に資格証明が含まれません。リモート・データベースへの接続に使用する資格証明は、データベース・リンクを参照するユーザーや、アプリケーションが実行する操作によって異なります。

ノート:

多くの分散アプリケーションでは、ユーザーがリモート・データベースでの権限を持つ必要はありません。これを実現する簡単な方法の 1 つは、固定ユーザーまたは現行ユーザーのデータベース・リンクを含むプロシージャを作成することです。この方法では、プロシージャにアクセスするユーザーに対して第三者の権限が一時的に付与されます。

- 接続ユーザー・データベース・リンクの作成
  接続ユーザー・データベース・リンクを作成するには、CREATE DATABASE LINK文のCONNECT TO句を省略します。
- 現行ユーザー・データベース・リンクの作成
  現行ユーザー・データベース・リンクを作成するには、CREATE DATABASE LINK文でCONNECT TO CURRENT_USER句を使用します。

関連トピック

- データベース・リンクのユーザー

親トピック: リンク・ユーザーの指定

32.2.3.2.1 接続ユーザー・データベース・リンクの作成

接続ユーザー・データベース・リンクを作成するには、CREATE DATABASE LINK文のCONNECT TO句を省略します。

次の構文は、接続ユーザー・データベース・リンクを作成し、ここでdblinkはリンクの名前、net_service_nameはオプションの接続文字列を表します。

```
CREATE [SHARED] [PUBLIC] DATABASE LINK dblink ... [USING 'net_service_name'];
```

たとえば、接続ユーザー・データベース・リンクを作成するために、次の構文を使用します。

```
CREATE DATABASE LINK sales.division3.example.com USING 'sales';
```

親トピック: 接続ユーザーおよび現行ユーザー・データベース・リンクの作成

32.2.3.2.2 現行ユーザー・データベース・リンクの作成

現行ユーザー・データベース・リンクを作成するには、CREATE DATABASE LINK文でCONNECT TO CURRENT_USER句を使用します。

現行ユーザー・リンクは、Oracle Advanced Securityオプションでのみ使用できます。

次の構文は、現行ユーザー・データベース・リンクを作成します。ここで、dblinkはリンクの名前、net_service_nameはオプションの接続文字列を表します。

```
CREATE [SHARED] [PUBLIC] DATABASE LINK dblink CONNECT TO CURRENT_USER
[USING 'net_service_name'];
```

たとえば、salesデータベースへの現行ユーザー・データベース・リンクを作成するには、次の構文を使用します。

```
CREATE DATABASE LINK sales CONNECT TO CURRENT_USER USING 'sales';
```

ノート:

現行ユーザー・データベース・リンクを使用するには、リンクに関係している両方のデータベースで現行ユーザーがグ

ローバル・ユーザーであることが必要です。

関連項目:

データベース・リンクの作成に関する構文情報の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 接続ユーザーおよび現行ユーザー・データベース・リンクの作成

32.2.4 リンク名に含まれるサービス名を指定するための接続修飾子の使用

場合によっては、同じリモート・データベースを指すものの、異なる通信経路を使用してリモート・データベースへの接続を確立する同じタイプ(パブリックなど)のデータベース・リンクを複数作成する必要があります。

次のような場合に、この方法が役立ちます。

- リモート・データベースがOracle Real Application Clusters構成の一部であり、そのためリモート・データベースの特定のインスタンスに接続を確立できるようにローカル・ノードで複数のパブリック・データベース・リンクを定義する場合。
- Oracle Databaseサーバーへの接続に、TCP/IPを使用するクライアントと、DECNETを使用するクライアントがある場合。

このような機能を容易にするため、データベースでは、データベース・リンクにオプションのサービス名を付加したデータベース・リンク名を作成できます。データベース・リンクを作成するときに、サービス名を@記号で区切って(たとえば、@sales)、データベース・リンク名の末尾の部分として指定します。この文字列は接続修飾子と呼ばれます。

たとえば、リモート・データベースhq.example.comがOracle Real Application Clusters環境で管理されているとします。hqデータベースには、hq_1およびhq_2という名前の2つのインスタンスがあります。ローカル・データベースで次のパブリック・データベース・リンクを作成して、hqデータベースのリモート・インスタンスへの経路を定義できます。

```
CREATE PUBLIC DATABASE LINK hq.example.com@hq_1
  USING 'string_to_hq_1';
CREATE PUBLIC DATABASE LINK hq.example.com@hq_2
  USING 'string_to_hq_2';
CREATE PUBLIC DATABASE LINK hq.example.com
  USING 'string_to_hq';
```

最初の2つの例では、サービス名は単なるデータベース・リンク名の一部であることに注目してください。サービス名のテキストは必ずしも接続の確立方法を示しているわけではないため、この情報はUSING句のサービス名で指定されています。また、3番目の例では、サービス名がリンク名の一部として指定されていないことに注目してください。この場合も、サービス名がリンク名の一部として指定されたときと同じように、インスタンスはUSING文字列によって決定されます。

サービス名を使用して特定のインスタンスを指定するには、サービス名をグローバル・オブジェクト名の末尾に付けます。

```
SELECT * FROM scott.emp@hq.example.com@hq_1
```

この例では、2つのアットマーク(@)があることに注意してください。

親トピック: データベース・リンクの作成

32.3 共有データベース・リンクの使用

標準のデータベース・リンクを使用してリモート・サーバーを参照するアプリケーションはすべて、ローカル・データベースとリモート・データベースの間で接続を確立します。多くのユーザーがアプリケーションを同時に実行した場合、ローカル・データベースとリモー

ト・データベースの間で多数の接続が発生する可能性があります。共有データベース・リンクを使用すると、ローカル・サーバーとリモート・サーバーの間で必要なネットワーク接続の数を制限できます。

- 共有データベース・リンクの使用の決定
  アプリケーションおよび共有サーバーの構成を綿密に検討して、共有リンクを使用するかどうかを判断してください。簡単なガイドラインとして、データベース・リンクにアクセスするユーザー数がローカル・データベース内のサーバー・プロセス数よりもかなり多いと予測されるときは、共有データベース・リンクを使用します。
- 共有データベース・リンクの作成
  共有データベース・リンクを作成するには、CREATE DATABASE LINK文でキーワードSHAREDを使用します。
- 共有データベース・リンクの構成
  共有データベース・リンクは、様々な方法で構成できます。

関連項目:

共有データベース・リンクの概念の概要は、「共有データベース・リンクの概要」

親トピック: 分散データベースの管理

32.3.1 共有データベース・リンクの使用の判断

アプリケーションおよび共有サーバーの構成を綿密に検討して、共有リンクを使用するかどうかを判断してください。簡単なガイドラインとして、データベース・リンクにアクセスするユーザー数がローカル・データベース内のサーバー・プロセス数よりもかなり多いと予測されるときは、共有データベース・リンクを使用します。

データベース・リンクに関して可能な3つの構成を次の表に示します。

リンク・タイプ	サーバー・モード	影響
非共有	専用/共有サーバー	アプリケーションで標準のパブリック・データベース・リンクを使用していて、100 人のユーザーが同時に接続を要求した場合は、リモート・データベースへの直接ネットワーク接続が 100 必要になります。
共有	共有サーバー	ローカルの共有サーバー・モード・データベースに 10 の共有サーバー・プロセスが存在している場合、同じデータベース・リンクを使用するユーザーが 100 人いるときに必要となるリモート・サーバーへのネットワーク接続数は 10 以下になります。ローカルの各共有サーバー・プロセスが必要とするリモート・サーバーへの接続は、それぞれ 1 つで済む場合があります。
共有	専用	10 のクライアントがローカルの専用サーバーに接続していて、各クライアントが同じ接続のセッションを 10 持っており(したがって全部で 100 のセッションを確立している)、各セッションで同じリモート・データベースを参照している場合、必要な接続は 10 で済みます。非共有データベース・リンクでは、100 の接続が必要になります。

共有データベース・リンクは、必ずしもすべての状況で役立つとはかぎりません。たとえば、リモート・サーバーに接続するユーザー

が1人のみの場合を考えます。このユーザーが共有データベース・リンクを定義し、ローカル・データベースに10個の共有サーバー・プロセスが存在する場合、このユーザーはリモート・サーバーへのネットワーク接続を最大10個必要とする可能性があります。ユーザーは個々の共有サーバー・プロセスを使用できるため、各プロセスはそれぞれリモート・サーバーへの接続を確立できます。

この場合、ネットワーク接続は1つしか必要ないため、明らかに非共有データベース・リンクの方が適しています。シングルユーザー環境では、共有データベース・リンクの使用はネットワーク接続の増加につながるため、共有リンクは、複数のユーザーが同じリンクを使用する必要がある場合にのみ使用します。通常、共有リンクはパブリック・データベース・リンクに使用しますが、複数のクライアントが同じローカル・スキーマにアクセスする場合(その結果、同じプライベート・データベース・リンクにアクセスする場合)はプライベート・データベース・リンクにも使用できます。

ノート:

複数階層の環境では、共有データベース・リンクを使用してリモート・データベースに接続した場合、そのリモート・データベースは、移行不可能なデータベース・リンクを使用して別のデータベースにリンクできないという制約があります。このリンクには共有サーバーを使用するか、またはリンク自体が別の共有データベース・リンクである必要があります。

親トピック: 共有データベース・リンクの使用

32.3.2 共有データベース・リンクの作成

共有データベース・リンクを作成するには、CREATE DATABASE LINK文でキーワードSHAREDを使用します。

共有データベース・リンクを作成するには、次の構文を使用します。

```
CREATE SHARED DATABASE LINK dblink_name
[CONNECT TO username IDENTIFIED BY password]|[CONNECT TO CURRENT_USER]
AUTHENTICATED BY schema_name IDENTIFIED BY password
[USING 'service_name'];
```

キーワードSHAREDを使用するときは、必ずAUTHENTICATED BY句が必要です。AUTHENTICATED BYで指定されたスキーマがリモート・データベースに存在し、少なくとも、CREATE SESSION権限が付与されている必要があります。このスキーマの資格証明は、ローカル・データベースとリモート・データベース間での認証方式と考えることができます。これらの資格証明は、データベース・リンクのユーザーになりすまして情報に不当にアクセスしようとするクライアントから、リモート共有サーバー・プロセスを保護するために必要です。

共有データベース・リンクとの接続後、リモート・データベース上での操作は、AUTHENTICATED BYスキーマではなく、CONNECT TOユーザー、またはCURRENT_USERの権限で実行されます。

次の例では、scottとして接続し、linkuserとして認証される、salesデータベースへの固定ユーザー共有リンクを作成しています。

```
CREATE SHARED DATABASE LINK link2sales
CONNECT TO scott IDENTIFIED BY password
AUTHENTICATED BY linkuser IDENTIFIED BY ostrich
USING 'sales';
```

関連項目:

CREATE DATABASE LINK文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 共有データベース・リンクの使用

32.3.3 共有データベース・リンクの構成

共有データベース・リンクは、様々な方法で構成できます。

- 専用サーバーへの共有リンクの作成
  ローカル・サーバーの共有サーバー・プロセスは、専用リモート・サーバー・プロセスを所有できます。
- 共有サーバーへの共有リンクの作成
  リモート・サーバー上の共有サーバー・プロセスを使用して共有リンクを作成できます。

親トピック: 共有データベース・リンクの使用

32.3.3.1 専用サーバーへの共有リンクの作成

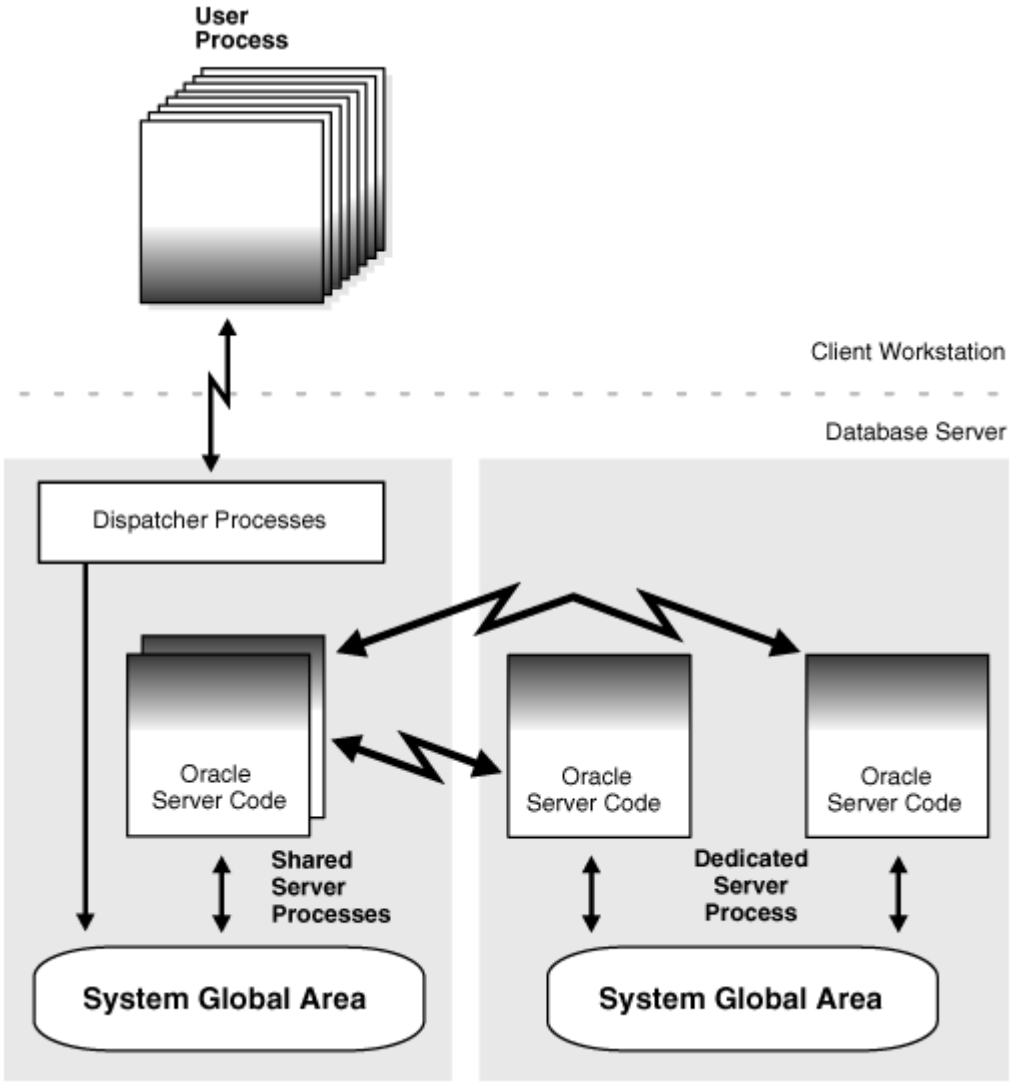
ローカル・サーバーの共有サーバー・プロセスは、専用リモート・サーバー・プロセスを所有できます。

この構成の利点は、ローカルの共有サーバーとリモートの専用サーバーとの間に直接的なネットワーク・トランスポートが存在することです。ただし、余分なバックエンド・サーバー・プロセスが必要になるというデメリットもあります。

ノート:

リモート・サーバーは、共有サーバーまたは専用サーバーのどちらか一方にできます。ローカル・サーバーとリモート・サーバーの間には専用の接続が存在します。リモート・サーバーが共有サーバーのときは、サービス名の定義にSERVER=DEDICATED 句を使用することで、専用サーバー接続を強制できます。

図32-1 専用サーバー・プロセスへの共有データベース・リンク



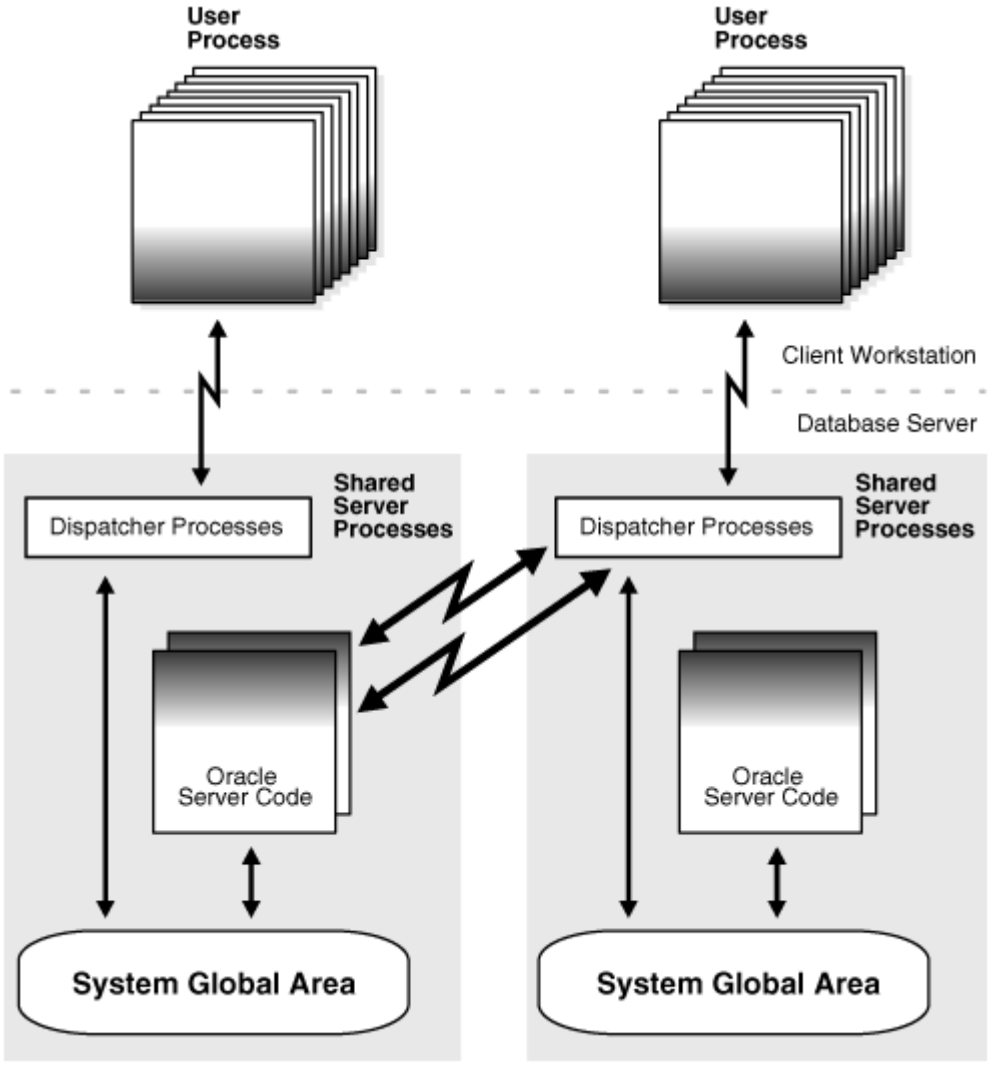
親トピック: 共有データベース・リンクの構成

32.3.3.2 共有サーバーへの共有リンクの作成

リモート・サーバー上の共有サーバー・プロセスを使用して共有リンクを作成できます。

この構成では、多数の専用サーバー・プロセスは必要ありませんが、リモート・サーバーのディスパッチャを経由する接続が必要になります。この場合、ローカル・サーバーとリモート・サーバーの両方を共有サーバーとして構成する必要があります。

図32-2 共有サーバーへの共有データベース・リンク



関連項目:

共有サーバー・オプションの詳細は、「共有サーバー・プロセス」

親トピック: 共有データベース・リンクの構成

32.4 データベース・リンクの管理

データベース・リンクの管理には、クローズ、削除、およびアクティブな接続数の制限などのタスクが含まれます。

- データベース・リンクのクローズ
  セッション中にデータベース・リンクにアクセスした場合、そのセッションをクローズするまでリンクはオープンしたままです。手動でデータベース・リンクをクローズするには、`ALTER SESSION CLOSE DATABASE LINK`文を使用します。
- データベース・リンクの削除
  データベース・リンクは、表またはビューと同様に削除できます。リンクがプライベートの場合は、自分のスキーマ内に存在する必要があります。リンクがパブリックの場合は、`DROP PUBLIC DATABASE LINK`システム権限が必要です。
- アクティブ・データベース・リンクの接続数の制限
  静的な初期化パラメータ`OPEN_LINKS`を使用して、ユーザー・プロセスからリモート・データベースへの接続数を制限できます。

親トピック: 分散データベースの管理

32.4.1 データベース・リンクのクローズ

あるセッションでデータベース・リンクにアクセスする場合、そのセッションをクローズするまでリンクはオープンしたままです。手動でデータベース・リンクをクローズするには、ALTER SESSION CLOSE DATABASE LINK文を使用します。

リンクがオープンしているということは、そのリンクを介してアクセスしている各リモート・データベースのプロセスがアクティブであることを意味します。この状況では、次のような結果が生じます。

- 20人のユーザーがセッションをオープンしてローカル・データベースの同じパブリック・リンクにアクセスする場合、20のデータベース・リンク接続がオープンします。
- 20人のユーザーがセッションをオープンして各ユーザーがプライベート・リンクにアクセスする場合、20のデータベース・リンク接続がオープンします。
- 1人のユーザーがセッションを開始して20の異なるリンクにアクセスする場合、20のデータベース・リンク接続がオープンします。

セッションをクローズした後、セッションでアクティブだったリンクは自動的にクローズされます。ユーザーがリンクを手動でクローズする場合もあります。たとえば、次のようなときにリンクをクローズします。

- リンクによって確立されたネットワーク接続がアプリケーションでそれほど頻繁に使用されないとき。
- ユーザー・セッションを終了する必要があるとき。

リンクをクローズするには次の文を発行しますが、ここでlinknameはリンクの名前を表します。

```
ALTER SESSION CLOSE DATABASE LINK linkname;
```

この文は、現行セッションでアクティブなリンクのみをクローズします。

親トピック: データベース・リンクの管理

32.4.2 データベース・リンクの削除

データベース・リンクは、表やビューと同様に削除できます。リンクがプライベートの場合は、自分のスキーマ内に存在する必要があります。リンクがパブリックの場合は、DROP PUBLIC DATABASE LINKシステム権限が必要です。

構文は次のとおりです。ここで、dblinkはリンクの名前を表します。

```
DROP [PUBLIC] DATABASE LINK dblink;
```

- プライベート・データベース・リンクの削除
  プライベート・データベース・リンクを削除するには、DROP DATABASE LINK文を使用します。
- パブリック・データベース・リンクの削除
  パブリック・データベース・リンクを削除するには、DROP PUBLIC DATABASE LINK文を使用します。

親トピック: データベース・リンクの管理

32.4.2.1 プライベート・データベース・リンクの削除

プライベート・データベース・リンクを削除するには、DROP DATABASE LINK文を使用します。

1. SQL*Plusを使用して、ローカル・データベースに接続します。たとえば、次のように入力します。

   ```
   CONNECT scott@local_db
   ```

2. USER_DB_LINKSを問い合せて、所有しているリンクを確認します。たとえば、次のように入力します。

```
SELECT DB_LINK FROM USER_DB_LINKS;
DB_LINK
----------------------------------
SALES.US.EXAMPLE.COM
MKTG.US.EXAMPLE.COM
2 rows selected.
```

3. DROP DATABASE LINK文を使用して、目的のリンクを削除します。たとえば、次のように入力します。

```
DROP DATABASE LINK sales.us.example.com;
```

親トピック: データベース・リンクの削除

32.4.2.2 パブリック・データベース・リンクの削除

パブリック・データベース・リンクを削除するには、DROP PUBLIC DATABASE LINK文を使用します。

1. ローカル・データベースにDROP PUBLIC DATABASE LINK権限を持つユーザーとして接続します。たとえば、次のように入力します。

```
CONNECT SYSTEM@local_db AS SYSDBA
```

2. DBA_DB_LINKSを問い合せて、パブリック・リンクを確認します。たとえば、次のように入力します。

```
SELECT DB_LINK FROM DBA_DB_LINKS
  WHERE OWNER = 'PUBLIC';
DB_LINK
----------------------------------
DBL1.US.EXAMPLE.COM
SALES.US.EXAMPLE.COM
INST2.US.EXAMPLE.COM
RMAN2.US.EXAMPLE.COM
4 rows selected.
```

3. DROP PUBLIC DATABASE LINK文を使用して、目的のリンクを削除します。たとえば、次のように入力します。

```
DROP PUBLIC DATABASE LINK sales.us.example.com;
```

親トピック: データベース・リンクの削除

32.4.3 アクティブ・データベース・リンクの接続数の制限

静的な初期化パラメータOPEN_LINKSを使用すると、ユーザー・プロセスからリモート・データベースへの接続数を制限できます。

このパラメータは、分散トランザクションにおいて1つのユーザー・セッションが同時に使用できるリモート接続数を制御します。

このパラメータを設定する際は、次の点を考慮してください。

- 設定値は、複数のデータベースを参照する1つのSQL文によって参照されるデータベースの数以上にします。
- 複数の分散データベースに継続してアクセスする場合は、設定値を大きくします。たとえば、3つのデータベースに定期的にアクセスする場合は、OPEN_LINKSを3以上に設定します。
- OPEN_LINKSのデフォルト値は4です。OPEN_LINKSを0(ゼロ)に設定した場合、分散トランザクションは許可されません。

関連項目:

OPEN_LINKS初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

32.5 データベース・リンク情報の表示

各データベースのデータ・ディクショナリには、そのデータベース内にあるデータベース・リンクの定義がすべて格納されています。データ・ディクショナリ表およびビューを使用して、リンクに関する情報を取得できます。

- データベース内のリンクの判断
  ローカル・データベースに定義され、データ・ディクショナリに格納されているデータベース・リンクは、ビュー・セットに表示されます。
- オープンしているリンク接続の判断
  セッション内で現在オープンしているデータベース・リンク接続がわかれば役に立つ場合があります。
- 送信データベース・リンクのホストの判断
  送信データベース・リンクのホスト名を判断するには、DBMS_TNSパッケージのRESOLVE_TNSNAMEファンクションを使用できます。
- 受信データベース・リンクの情報の判断
  受信データベース・リンクに関する情報を判断するには、DBA_DB_LINK_SOURCESビューを使用できます。
- 受信データベース・リンクの高SCNアクティビティ・ソースの判断
  受信データベース・リンクで高いシステム変更番号(SCN)アクティビティのソースを判断するには、次のビューを使用できます: DBA_EXTERNAL_SCN_ACTIVITY、DBA_DB_LINK_SOURCESおよびDBA_DB_LINK。

32.5.1 データベース内のリンクの判断

ローカル・データベースに定義され、データ・ディクショナリに格納されているデータベース・リンクは、ビュー・セットに表示されます。

:

ビュー	用途
DBA_DB_LINKS	データベース内のデータベース・リンクがすべてリストされます。
ALL_DB_LINKS	接続ユーザーがアクセス可能なデータベース・リンクがすべてリストされます。
USER_DB_LINKS	接続ユーザーが所有しているデータベース・リンクがすべてリストされます。

これらのデータ・ディクショナリ・ビューには、データベース・リンクに関する同じ基本情報が含まれていますが、いくつか例外もあります。

列	対象になるビュー	説明
OWNER	USER_*を除くすべて	データベース・リンクを作成したユーザー。リンクがパブリックの場合、ユーザーは PUBLIC としてリストされます。
DB_LINK	すべて	データベース・リンクの名前。

列	対象になるビュー	説明
USERNAME	すべて	リンク定義に固定ユーザーが含まれている場合、この列には固定ユーザーのユーザー名が表示されます。固定ユーザーが含まれていない場合、この列は NULL になります。
PASSWORD	USER_*のみ	使用されません。下位互換性のためにのみ維持されています。
HOST	すべて	リモート・データベースへの接続に使用されるネット・サービス名。
CREATED	すべて	データベース・リンクの作成日時。

どのユーザーでもUSER_DB_LINKSを問い合せることで、そのユーザーが使用できるデータベース・リンクを判断できます。ALL_DB_LINKSビューまたはDBA_DB_LINKSビューを使用できるのは、追加の権限を持つユーザーのみです。

次のスクリプトは、DBA_DB_LINKSビューを問い合せて、リンク情報にアクセスします。

```
COL OWNER FORMAT a10
COL USERNAME FORMAT A8 HEADING "USER"
COL DB_LINK FORMAT A30
COL HOST FORMAT A7 HEADING "SERVICE"
SELECT * FROM DBA_DB_LINKS
/
```

ここでスクリプトが起動され、その出力結果が表示されます。

```
SQL>@link_script
OWNER      DB_LINK                        USER     SERVICE CREATED
---------- ------------------------------ -------- ------- ----------
SYS        TARGET.US.EXAMPLE.COM             SYS      inst1   23-JUN-99
PUBLIC     DBL1.UK.EXAMPLE.COM               BLAKE    ora51   23-JUN-99
PUBLIC     RMAN2.US.EXAMPLE.COM                       inst2   23-JUN-99
PUBLIC     DEPT.US.EXAMPLE.COM                        inst2   23-JUN-99
JANE       DBL.UK.EXAMPLE.COM                BLAKE    ora51   23-JUN-99
SCOTT      EMP.US.EXAMPLE.COM                SCOTT    inst2   23-JUN-99
6 rows selected.
```

親トピック: データベース・リンク情報の表示

32.5.2 オープンしているリンク接続の判断

自分のセッションで現在オープンしているデータベース・リンク接続がわかれば役に立つ場合があります。

しかし、SYSDBAとして接続している場合には、ビューを問い合せてすべてのセッションでオープンしているすべてのリンクを判断することはできず、現在作業中のセッションのリンク情報にアクセスすることしかできません。

次のビューには、現行セッションで現在オープンしているデータベース・リンク接続が表示されます。

ビュー	用途
V$DBLINK	自分のセッションでオープンしているすべてのデータベース・リンク、つまり、IN_TRANSACTION 列が YES に設定されているすべてのデータベース・リンク

ビュー	用途
	がリストされます。
GV$DBLINK	自分のセッションでオープンしているすべてのデータベース・リンクと対応するインスタンスがリストされます。このビューは、Oracle Real Application Clusters構成の使用時に役立ちます。

これらのデータ・ディクショナリ・ビューには、データベース・リンクに関する同じ基本情報が含まれていますが、1つ例外があります。

列	対象になるビュー	説明
DB_LINK	すべて	データベース・リンクの名前。
OWNER_ID	すべて	データベース・リンクの所有者。
LOGGED_ON	すべて	データベース・リンクが現在ログインされているかどうか。
HETEROGENEOUS	すべて	データベース・リンクが同機種間(NO)と異機種間(YES)のどちらであるか。
PROTOCOL	すべて	データベース・リンクの通信プロトコル。
OPEN_CURSORS	すべて	データベース・リンクでカーソルがオープンされているかどうか。
IN_TRANSACTION	すべて	コミットまたはロールバックがまだ完了していないトランザクションで、データベース・リンクがアクセスされているかどうか。
UPDATE_SENT	すべて	データベース・リンクで更新があったかどうか。
COMMIT_POINT_STRENGTH	すべて	データベース・リンクを使用しているトランザクションのコミット・ポイント強度。
INST_ID	GV$DBLINK のみ	ビュー情報が取得されたインスタンス。

たとえば、次のスクリプトを作成し、実行することで、オープンされているリンクを判断できます(出力例も含まれています)。

```
COL DB_LINK FORMAT A25
COL OWNER_ID FORMAT 99999 HEADING "OWNID"
COL LOGGED_ON FORMAT A5 HEADING "LOGON"
COL HETEROGENEOUS FORMAT A5 HEADING "HETER"
COL PROTOCOL FORMAT A8
COL OPEN_CURSORS FORMAT 999 HEADING "OPN_CUR"
COL IN_TRANSACTION FORMAT A3 HEADING "TXN"
COL UPDATE_SENT FORMAT A6 HEADING "UPDATE"
COL COMMIT_POINT_STRENGTH FORMAT 99999 HEADING "C_P_S"
```

```
SELECT * FROM V$DBLINK
/
SQL> @dblink
DB_LINK                    OWNID LOGON HETER PROTOCOL OPN_CUR TXN UPDATE  C_P_S
-------------------------- ------ ----- ----- -------- ------- --- ------ ------
INST2.EXAMPLE.COM                 0 YES   YES    UNKN          0 YES YES       255
```

親トピック: データベース・リンク情報の表示

32.5.3 送信データベース・リンクのホストの確認

送信データベース・リンクのホスト名を判断するには、DBMS_TNSパッケージのRESOLVE_TNSNAMEファンクションを使用できます。

送信データベース・リンクでは、ネットワーク・サービス名などの接続識別子によってデータベース・リンクが作成されている場合、DBA_DB_LINKSビューのHOST列の値は、接続データを解決しません。結果的に、ホスト名などの接続データの一部を必要とするツールは、情報を見つけるためにtnsnames.oraファイルをチェックする必要があります。ただし、ツールがtnsnames.oraファイルにアクセスできない場合もあります。

1. DBMS_TNSパッケージ内のサブプログラムを実行できるユーザーとして送信データベース・リンクが含まれるデータベースに接続します。
2. DBA_DB_LINKSビューを問い合せます。

例32-1 送信データベース・リンクのホストの確認

この問合せ結果には、各送信データベース・リンクのホスト名を含む接続データが表示されます。

```
SELECT DB_LINK, DBMS_TNS.RESOLVE_TNSNAME(HOST) FROM DBA_DB_LINKS;
```

関連項目:

- 『Oracle Databaseリファレンス』
- RESOLVE_TNSNAMEファンクションの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

親トピック: データベース・リンク情報の表示

32.5.4 受信データベース・リンクについての情報の確認

受信データベース・リンクに関する情報を判断するには、DBA_DB_LINK_SOURCESビューを使用できます。

データベースは、受信データベース・リンクの一意の接続に関する詳細を永続表およびDBA_DB_LINK_SOURCESビューに記録します。このビューに問い合せると、データベース・リンクに対する受信接続に関する情報を取得できます。

1. DBA_DB_LINK_SOURCESビューの問合せを実行できるユーザーとしてデータベースに接続します。
2. DBA_DB_LINK_SOURCESビューを問い合せます。

例32-2 DBA_DB_LINK_SOURCESビューの問合せ

この例は、各受信データベース・リンクのデータベース名、ホスト名を返します。また、データベース・リンクによる現在のデータベースへの最初と最後のログイン時間も返します。

```
SELECT DB_NAME, HOST_NAME, FIRST_LOGON_TIME, LAST_LOGON_TIME
   FROM DBA_DB_LINK_SOURCES;
```

関連項目:

『Oracle Databaseリファレンス』

親トピック: データベース・リンク情報の表示

32.5.5 受信データベース・リンクのSCNの高アクティビティの原因の確認

受信データベース・リンクで高いシステム変更番号(SCN)アクティビティのソースを判断するには、次のビューを使用できます: DBA_EXTERNAL_SCN_ACTIVITY、DBA_DB_LINK_SOURCESおよびDBA_DB_LINK。

分散データベース環境での分散トランザクションと分散読取りの一貫性を実現するために、データベース・リンクを介して呼出しが行われた場合、データベースはSCNを同期します。SCNの増加率が高いと、データベースがエラーを返す可能性があります。

多くの場合、分散データベース操作から生じるSCNの異常な増加の原因を判別するのは困難です。DBA_EXTERNAL_SCN_ACTIVITYビューにより、過剰なSCNの増加を引き起こしているデータベースまたはクライアントを見つけ出すことができます。問合せでこのビューをDBA_DB_LINK_SOURCES およびDBA_DB_LINKビューと結合することにより、情報が返されます。

1. DBA_EXTERNAL_SCN_ACTIVITY、DBA_DB_LINK_SOURCESおよびDBA_DB_LINKビューの問合せを実行できるユーザーとしてデータベースに接続します。
2. 最近のSCN増加の履歴とそれらのソースを表示するために、次の問合せを実行します。

```
(SELECT RESULT, OPERATION_TIMESTAMP, EXTERNAL_SCN, SCN_ADJUSTMENT, HOST_NAME,
DB_NAME, SESSION_ID, SESSION_SERIAL#
   FROM DBA_EXTERNAL_SCN_ACTIVITY a, DBA_DB_LINK_SOURCES s
   WHERE a.INBOUND_DB_LINK_SOURCE_ID = s.SOURCE_ID)
UNION
(SELECT RESULT, OPERATION_TIMESTAMP, EXTERNAL_SCN, SCN_ADJUSTMENT,
dbms_tns.resolve_tnsname(HOST) HOST_NAME, NULL DB_NAME, SESSION_ID,
SESSION_SERIAL#
   FROM DBA_EXTERNAL_SCN_ACTIVITY a, DBA_DB_LINKS o
   WHERE a.OUTBOUND_DB_LINK_NAME = o.DB_LINK AND
         a.OUTBOUND_DB_LINK_OWNER = o.OWNER)
UNION
(SELECT RESULT, OPERATION_TIMESTAMP, EXTERNAL_SCN,   SCN_ADJUSTMENT,  s.MACHINE
HOST_NAME, NULL DB_NAME, SESSION_ID, SESSION_SERIAL#
   FROM DBA_EXTERNAL_SCN_ACTIVITY a, V$SESSION s
   WHERE a.SESSION_ID = s.SID AND
         a.SESSION_SERIAL#=s.SERIAL# AND
         INBOUND_DB_LINK_SOURCE_ID IS NULL AND
         OUTBOUND_DB_LINK_NAME IS NULL AND
         OUTBOUND_DB_LINK_OWNER IS NULL);
```

ノート:

高 SCN アクティビティが DBA_EXTERNAL_SCN_ACTIVITY ビューに記録されていない場合、この問合せは結果を返しません。

関連項目:

『Oracle Databaseリファレンス』

親トピック: データベース・リンク情報の表示

32.6 位置の透過性の作成

必要なデータベース・リンクの構成が完了した後、各種のツールを使用してデータベース・システムの分散的な性質をユーザーから隠すことができます。言い換えれば、ユーザーはリモート・オブジェクトに対してあたかもローカル・オブジェクトであるかのようにアクセスできるようになります。

- ビューを使用した位置の透過性の作成
  ローカル・ビューは、分散データベース・システムにおいてローカルおよびリモートの表に対する位置の透過性を提供できます。
- シノニムを使用した位置の透過性の作成
  シノニムは、分散データベース・システムでの位置など、基礎となるオブジェクトの個別情報を隠すので、分散環境と非分散環境の両方で役立ちます。基礎になるオブジェクトを名前変更または移動する必要がある場合でも、シノニムを再定義するだけで済み、シノニムをベースとするアプリケーションは引き続き通常どおり動作します。また、分散データベース・システムのユーザーが使用するSQL文も、シノニムによって単純化されます。
- プロシージャを使用した位置の透過性の作成
  プロシージャと呼ばれるPL/SQLプログラム・ユニットは、位置の透過性を提供できます。

親トピック: 分散データベースの管理

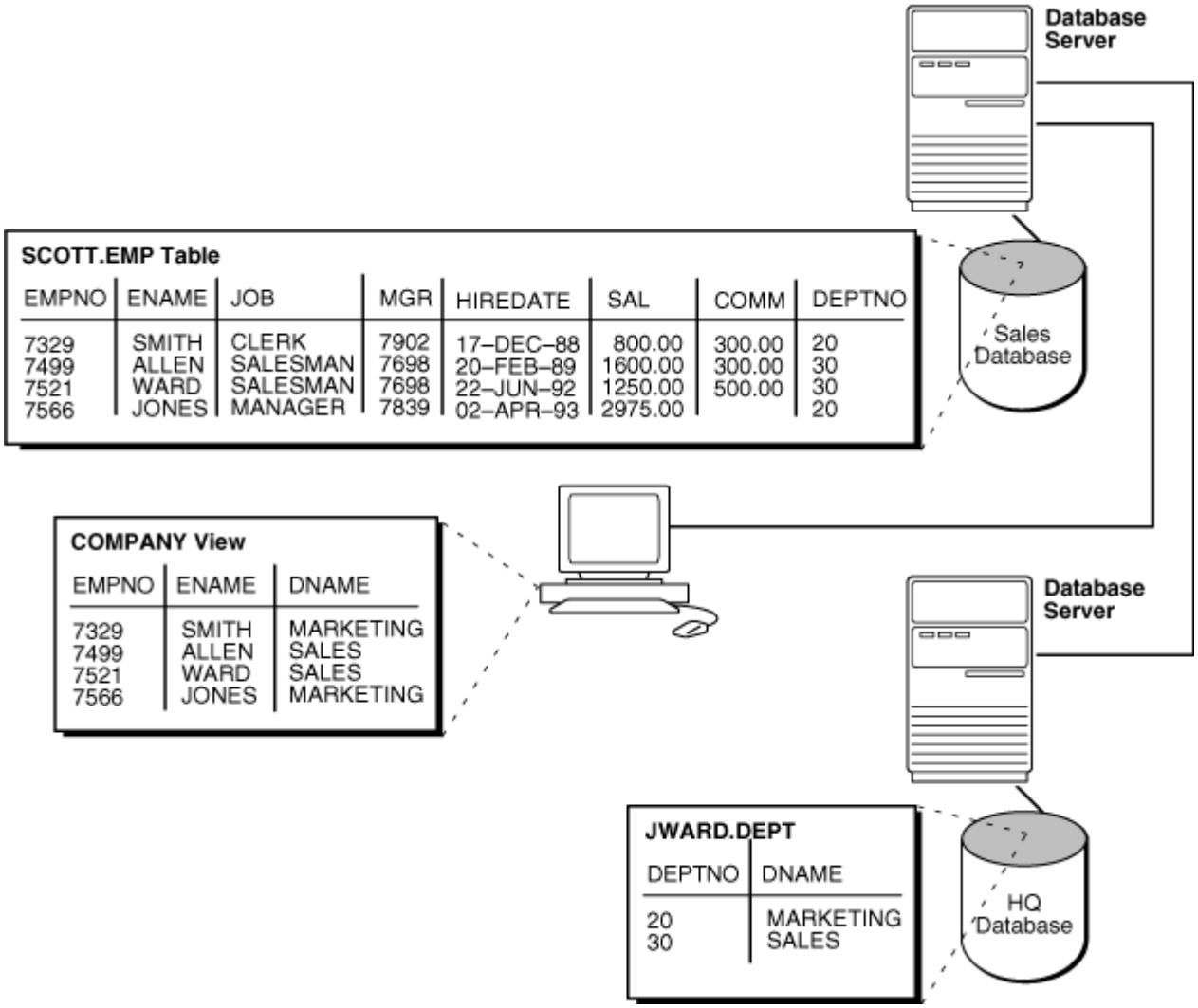
32.6.1 ビューを使用した位置の透過性の作成

ローカル・ビューは、分散データベース・システムにおいてローカルおよびリモートの表に対する位置の透過性を提供できます。

たとえば、emp表がローカル・データベースに、dept表がリモート・データベースに、それぞれ格納されているとします。システムのユーザーに対してこれらの表を透過的にするために、ローカル・データとリモート・データを結合するビューをローカル・データベースに作成できます。

```
CREATE VIEW company AS
   SELECT a.empno, a.ename, b.dname
   FROM scott.emp a, jward.dept@hq.example.com b
   WHERE a.deptno = b.deptno;
```

図32-3 ビューと位置の透過性



ユーザーはこのビューにアクセスするときに、データが物理的に格納されている場所や、複数の表のデータにアクセスしているかどうかについて知る必要はありません。したがって、ユーザーは必要な情報をより簡単に取得できます。たとえば、次の問合せは、ローカルおよびリモートの両方のデータベース表からのデータを提供します。

```
SELECT * FROM company;
```

ローカル・ビューの所有者は、リモート・ユーザーによってすでに付与されているローカル・ビューのオブジェクト権限のみを付与できます。(リモート・ユーザーはデータベース・リンクのタイプによって示されます)。これは、ローカル・データを参照するビューの権限の管理と似ています。

親トピック: 位置の透過性の作成

32.6.2 シノニムを使用した位置の透過性の作成

シノニムは、分散データベース・システム内での位置など、基礎になるオブジェクトの個別情報を隠すため、分散環境と非分散環境の両方で役立ちます。基礎になるオブジェクトを名前変更または移動する必要がある場合でも、シノニムを再定義するだけで済み、シノニムをベースとするアプリケーションは引き続き通常どおり動作します。また、分散データベース・システムのユーザーが使用するSQL文も、シノニムによって単純化されます。

- シノニムの作成
  すべてのシノニムは、それらが作成されるデータベースのデータ・ディクショナリに格納されるスキーマ・オブジェクトです。シノニムでは、データベース・リンクを介したリモート表へのアクセスを簡単にするために、単一ワードでリモート・データにアクセスでき、それによって特定のオブジェクト名や位置をシノニムのユーザーから隠すことができます。
- 権限とシノニムの管理

シノニムは、実際のオブジェクトへの参照です。特定のスキーマ・オブジェクトのシノニムにアクセスするユーザーは、元のスキーマ・オブジェクトそのものに対する権限を持っている必要があります。

親トピック: 位置の透過性の作成

32.6.2.1 シノニムの作成

シノニムはすべて、作成するデータベースのデータ・ディクショナリに格納されるスキーマ・オブジェクトです。シノニムでは、データベース・リンクを介したリモート表へのアクセスを簡単にするために、単一ワードでリモート・データにアクセスでき、それによって特定のオブジェクト名や位置をシノニムのユーザーから隠すことができます。

次のもののシノニムを作成できます。

- 表
- タイプ
- ビュー
- マテリアライズド・ビュー
- 順序
- プロシージャ
- ファンクション
- パッケージ

シノニムを作成するための構文は、次のとおりです。

```
CREATE [PUBLIC] SYNONYM synonym_name
FOR [schema.]object_name[@database_link_name];
```

ここで:

- PUBLICは、すべてのユーザーがこのシノニムを使用できることを指定するキーワードです。このパラメータを省略するとシノニムがプライベートになり、作成者のみ使用可能になります。パブリック・シノニムは、CREATE PUBLIC SYNONYMシステム権限を持つユーザーのみが作成できます。
- synonym_nameには、ユーザーおよびアプリケーションが参照する代替オブジェクト名を指定します。
- schemaには、object_nameで指定されたオブジェクトのスキーマを指定します。このパラメータを省略すると、オブジェクトのスキーマとして作成者のスキーマが使用されます。
- object_nameには、表、ビュー、順序、マテリアライズド・ビュー、型、プロシージャ、ファンクションまたはパッケージのいずれかを適宜指定します。
- database_link_nameには、object_nameで指定されたオブジェクトが存在するリモート・データベースおよびスキーマを識別するデータベース・リンクを指定します。

シノニムには、そのスキーマ内に存在するオブジェクトの中で一意の名前を付ける必要があります。スキーマにスキーマ・オブジェクトがあり、同じ名前のパブリック・シノニムが存在する場合、データベースはスキーマを所有しているユーザーがその名前を参照するときに、常にスキーマ・オブジェクトの方を検索します。

例: パブリック・シノニムの作成

分散データベース・システム内のすべてのデータベースにおいて、hqデータベースに格納されているscott.emp表のパブリック・シ

ノニムが定義されているとします。

```
CREATE PUBLIC SYNONYM emp FOR scott.emp@hq.example.com;
```

表scott.emp@hq.example.comの位置はパブリック・シノニムによって隠されているので、使用場所に依存しない従業員管理アプリケーションを設計できます。アプリケーション内のSQL文は、パブリック・シノニムempを参照して表にアクセスできます。

また、emp表をhqデータベースからhrデータベースに移動する場合も、システムのノードでパブリック・シノニムを変更するだけで済みます。従業員管理アプリケーションは、すべてのノードで引き続き正常に動作します。

親トピック: シノニムを使用した位置の透過性の作成

32.6.2.2 権限とシノニムの管理

シノニムは、実際のオブジェクトへの参照です。特定のスキーマ・オブジェクトのシノニムにアクセスするユーザーは、元のスキーマ・オブジェクトそのものに対する権限を持っている必要があります。

たとえば、ユーザーがシノニムにアクセスしようとして、そのシノニムが識別する表に対してユーザーが権限を持っていなければ、表またはビューが存在しないというエラーが発生します。

scottが、リモート・オブジェクトscott.emp@sales.example.comの別名としてローカル・シノニムempを作成したとします。scottは、このシノニムのオブジェクト権限を別のローカル・ユーザーに付与できません。この操作はsalesデータベースのリモートのemp表の権限の付与に相当しますが、これは許可されていないので、scottはこのシノニムのローカル権限を付与することはできません。この動作は、ローカルの表またはビューの別名であるシノニムの権限管理とは異なります。

したがって、シノニムを位置の透過性のために使用する場合、ローカル権限は管理できません。ベース・オブジェクトのセキュリティは、リモート・ノードですべて管理されます。たとえば、ユーザーadminはemp_synシノニムのオブジェクト権限を付与することはできません。

ビューやプロシージャの定義で参照されるデータベース・リンクとは異なり、シノニムで参照されるデータベース・リンクを解決する際は、シノニムへの参照が解析される時点で有効なスキーマが所有しているプライベート・リンクが最初に検索されます。したがって、オブジェクトの適切な解決を保証するには、基礎になるオブジェクトのスキーマをシノニムの定義の中で指定することが特に重要になります。

親トピック: シノニムを使用した位置の透過性の作成

32.6.3 プロシージャを使用した位置の透過性の作成

プロシージャと呼ばれるPL/SQLプログラム・ユニットは、位置の透過性を提供できます。

- ローカル・プロシージャを使用したリモート・データの参照
  プロシージャおよび関数(スタンドアロンまたはパッケージ)に、リモート・データを参照するSQL文を含めることができます。
- ローカル・プロシージャを使用したリモート・プロシージャのコール
  ローカル・プロシージャを使用してリモート・プロシージャをコールできます。リモート・プロシージャでは、要求されたデータ操作言語(DML)を実行できます。
- ローカル・シノニムを使用したリモート・プロシージャの参照
  ローカル・シノニムを使用してリモート・プロシージャを参照できます。
- プロシージャと権限の管理
  ローカル・プロシージャに、リモート表またはリモート・ビューを参照する文が含まれているとします。ローカル・プロシージャの所有者はexecute権限をすべてのユーザーに付与できるため、任意のユーザーがプロシージャを実行して間接的にリモート・データにアクセスできるようにすることができます。

親トピック: 位置の透過性の作成

32.6.3.1 ローカル・プロシージャを使用したリモート・データの参照

プロシージャおよびファンクションには、スタンドアロンとパッケージのどちらの場合でも、リモート・データを参照するSQL文を含めることができます。

たとえば、次の文で作成されるプロシージャを考えます。

```
CREATE PROCEDURE fire_emp (enum NUMBER) AS
BEGIN
  DELETE FROM emp@hq.example.com
  WHERE empno = enum;
END;
```

ユーザーまたはアプリケーションがfire_empプロシージャをコールするときに、リモート表が変更されようとしていることは見かけ上わかりません。

プロシージャ内の文でローカルのプロシージャ、ビューまたはシノニムを使用して間接的にリモート・データを参照するときは、2層目の位置の透過性が可能です。たとえば、次の文はローカル・シノニムを定義します。

```
CREATE SYNONYM emp FOR emp@hq.example.com;
```

このシノニムがあれば、次の文を使用してfire_empプロシージャを作成できます。

```
CREATE PROCEDURE fire_emp (enum NUMBER) AS
BEGIN
  DELETE FROM emp WHERE empno = enum;
END;
```

表emp@hq.acme.comを名前変更または移動した場合でも、表を参照するローカル・シノニムを変更するだけで済みます。このプロシージャをコールするプロシージャやアプリケーションを変更する必要はありません。

親トピック: プロシージャを使用した位置の透過性の作成

32.6.3.2 ローカル・プロシージャを使用したリモート・プロシージャのコール

ローカル・プロシージャを使用してリモート・プロシージャをコールできます。リモート・プロシージャでは、要求されたデータ操作言語(DML)を実行できます。

たとえば、scottがlocal_dbに接続して次のプロシージャを作成したとします。

```
CONNECT scott@local_db
CREATE PROCEDURE fire_emp (enum NUMBER)
AS
BEGIN
  EXECUTE term_emp@hq.example.com;
END;
```

ここで、scottは、リモート・データベースに接続して次のリモート・プロシージャを作成するとします。

```
CONNECT scott@hq.example.com
CREATE PROCEDURE term_emp (enum NUMBER)
AS
BEGIN
  DELETE FROM emp WHERE empno = enum;
END;
```

ユーザーまたはアプリケーションがlocal_dbに接続してfire_empプロシージャをコールすると、このプロシージャはさらにhq.example.comにあるリモートのterm_empプロシージャをコールします。

親トピック: プロシージャを使用した位置の透過性の作成

32.6.3.3 ローカル・シノニムを使用したリモート・プロシージャの参照

ローカル・シノニムを使用してリモート・プロシージャを参照できます。

たとえば、scottがローカルのsales.example.comデータベースに接続して次のプロシージャを作成したとします。

```
CREATE PROCEDURE fire_emp (enum NUMBER) AS
BEGIN
DELETE FROM emp@hq.example.com
WHERE empno = enum;
END;
```

この後、ユーザーpeggyがsupply.example.comデータベースに接続し、scottがリモートのsalesデータベースで作成したプロシージャに対する次のシノニムを作成します。

```
SQL> CONNECT peggy@supply
SQL> CREATE PUBLIC SYNONYM emp FOR scott.fire_emp@sales.example.com;
```

supplyのローカル・ユーザーは、このシノニムを使用してsalesのプロシージャを実行できます。

親トピック: プロシージャを使用した位置の透過性の作成

32.6.3.4 プロシージャと権限の管理

ローカル・プロシージャに、リモート表またはリモート・ビューを参照する文が含まれているとします。ローカル・プロシージャの所有者はexecute権限をすべてのユーザーに付与できるため、任意のユーザーがプロシージャを実行して間接的にリモート・データにアクセスできるようにすることができます。

一般に、プロシージャはセキュリティの面で役に立ちます。プロシージャの中で参照されるオブジェクトの権限を明示的にコール側のユーザーに付与する必要はありません。

親トピック: プロシージャを使用した位置の透過性の作成

32.7 文の透過性の管理

分散データベースでは、いくつかのSQL文がリモートの表を参照できます。

データベースでは、次に示す標準のDML文でリモート表を参照することが可能です。

- SELECT (問合せ)
- INSERT
- UPDATE
- DELETE
- SELECT...FOR UPDATE(異機種間システムでは常にサポートされるとはかぎらない)
- LOCK TABLE

結合、集計、副問合せおよびSELECT...FOR UPDATEを含む問合せでは、ローカルおよびリモートの表およびビューをいくつでも参照できます。たとえば、次の問合せは2つのリモート表の情報を結合します。

```
SELECT e.empno, e.ename, d.dname
  FROM scott.emp@sales.division3.example.com e, jward.dept@hq.example.com d
  WHERE e.deptno = d.deptno;
```

同機種環境では、UPDATE、INSERT、DELETEおよびLOCK TABLE文は、ローカルおよびリモートの両方の表を参照できま

す。リモート・データを更新するためのプログラミングは不要です。たとえば、次の文はローカル・データベースのjwardスキーマのemp表から行を選択して、scott.salesスキーマのリモート表empに新しい行を挿入します。

```
INSERT INTO scott.emp@sales.division3.example.com
  SELECT * FROM jward.emp;
```

文の透過性に関する制限事項

文の透過性には、いくつかの制限事項が適用されます。

- リモートのOracle Database以外のシステム上のオブジェクトを更新するデータ操作言語文では、ローカルのOracle Databaseのオブジェクトを参照できません。たとえば、次のような文を実行するとエラーになります。

  ```
  INSERT INTO remote_table@link as SELECT * FROM local_table;
  ```

- 1つのSQL文において、参照されるすべてのLONGおよびLONG RAWの列、順序、更新済の表およびロック済の表は、同じノードに存在する必要があります。
- データベースでは、同機種システムにおけるリモートのデータ定義言語(DDL)文(CREATE、ALTER、DROPなど)は使用できません。ただし、次の例のようにDBMS_SQLパッケージのプロシージャのリモート実行を使用する場合を除きます。

  ```
  DBMS_SQL.PARSE@link_name(crs, 'drop table emp', v7);
  ```

  異機種間システムでは、パススルー機能によってDDLの実行が可能です。
- ANALYZE文のLIST CHAINED ROWS句は、リモート表を参照できません。
- 分散データベース・システムでは、SYSDATE、USER、UID、USERENVなどの環境に依存したSQLファンクションは、データベースによって、その文(または文の一部)が実行される場所にかかわらず、常にローカル・サーバーについて評価されます。

  ノート:

  Oracle Database は、USERENV ファンクションを問合せの用途でのみサポートしています。
- リモート・オブジェクトのアクセスに関連して、次のようなパフォーマンス上の制限事項があります。
  - リモート・ビューは統計データを持ちません。
  - パーティション表に対する問合せは、最適化されない場合があります。
  - リモート表で考慮される索引の数は20以下です。
  - コンポジット索引で使用される列の数は20以下です。
- Oracle Databaseの分散読込み一貫性の実装には制限があり、これによってあるノードが別のノードに対して古い状態になる可能性があります。問合せを実行したときに、読込み一貫性に従ってデータが取得されているにもかかわらず、そのデータが古い場合があります。この問題の管理方法を学習するには、「読込み一貫性の管理」を参照してください。

関連項目:

DBMS_SQLパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: 分散データベースの管理

32.8 分散データベースの管理: 例

データベース・リンクの管理の例を示します。

- 例1: パブリック固定ユーザーのデータベース・リンクの作成
  パブリック固定ユーザーのデータベース・リンクの作成を示す例です。
- 例2: パブリック固定ユーザーの共有データベース・リンクの作成
  パブリック固定ユーザーの共有データベース・リンクの作成を示す例です。
- 例3: パブリック接続ユーザーのデータベース・リンクの作成
  パブリック接続ユーザーのデータベース・リンクの作成を示す例です。
- 例4: パブリック接続ユーザーの共有データベース・リンクの作成
  パブリック接続ユーザーの共有データベース・リンクの作成を示す例です。
- 例5: パブリック現行ユーザーのデータベース・リンクの作成
  パブリック現行ユーザーのデータベース・リンクの作成を示す例です。

親トピック: 分散データベースの管理

32.8.1 例1: パブリック固定ユーザーのデータベース・リンクの作成

パブリック固定ユーザーのデータベース・リンクの作成を示す例です。

次の文では、ローカル・データベースにjaneとして接続し、データベースsalesに対してパブリック固定ユーザーscottのデータベース・リンクを作成しています。データベースには、ネット・サービス名sldbを介してアクセスします。

```
CONNECT jane@local
CREATE PUBLIC DATABASE LINK sales.division3.example.com
  CONNECT TO scott IDENTIFIED BY password
  USING 'sldb';
```

この文の実行後は、ローカル・データベースに接続する任意のユーザーが、sales.division3.example.comデータベース・リンクを使用してリモート・データベースに接続できます。各ユーザーは、リモート・データベースのスキーマscottに接続します。

ユーザーは、次のSQL問合せを発行して、scottのリモート・スキーマの表empにアクセスできます。

```
SELECT * FROM emp@sales.division3.example.com;
```

各アプリケーションまたはユーザー・セッションは、サーバーの共通アカウントへの接続をそれぞれ別々に作成します。リモート・データベースへの接続は、アプリケーションの実行中またはユーザー・セッションの継続期間中オープンしたままです。

親トピック: 分散データベースの管理: 例

32.8.2 例2: パブリック固定ユーザーの共有データベース・リンクの作成

パブリック固定ユーザーの共有データベース・リンクの作成を示す例です。

次の例では、ローカル・データベースにdanaとして接続し、ネット・サービス名sldbを使用してsalesデータベースへのパブリック・リンクを作成しています。このリンクによって、リモート・データベースにscottとして接続でき、このユーザーがscottとして認証されます。

```
CONNECT dana@local
CREATE SHARED PUBLIC DATABASE LINK sales.division3.example.com
  CONNECT TO scott IDENTIFIED BY password
```

```
  AUTHENTICATED BY scott IDENTIFIED BY password
  USING 'sldb';
```

ローカルの共有サーバーに接続する任意のユーザーがこのデータベース・リンクを使用し、共有サーバー・プロセスを介してリモートのsalesデータベースに接続できます。その後ユーザーは、scottスキーマの表を問い合せることができます。

前述の例では、ローカルの共有サーバーはリモート・サーバーへの接続を1つずつ確立できます。ローカルの共有サーバー・プロセスでsales.division3.example.comデータベース・リンクを介したリモート・サーバーへのアクセスが必要になると、そのたびにローカルの共有サーバー・プロセスは確立済のネットワーク接続を再利用します。

親トピック: 分散データベースの管理: 例

32.8.3 例3: パブリック接続ユーザーのデータベース・リンクの作成

パブリック接続ユーザーのデータベース・リンクの作成を示す例です。

次の例では、ローカル・データベースにlarryとして接続し、ネット・サービス名sldbを使用してデータベースへのパブリック・リンクを作成しています。

```
CONNECT larry@local
CREATE PUBLIC DATABASE LINK redwood
  USING 'sldb';
```

ローカル・データベースに接続する任意のユーザーが、redwoodデータベース・リンクを使用できます。データベース・リンクを使用するローカル・データベースの接続ユーザーによって、リモート・スキーマが決まります。

接続ユーザーscottがデータベース・リンクを使用した場合、データベース・リンクはリモート・スキーマscottに接続します。接続ユーザーfoxがデータベース・リンクを使用した場合、データベース・リンクはリモート・スキーマfoxに接続します。

リモート・スキーマfoxがempスキーマ・オブジェクトを解決できない場合は、ローカル・データベースでローカル・ユーザーfoxが次の文を実行すると失敗します。つまり、sales.division3.example.comのfoxスキーマにempが表、ビューまたは(パブリック)シノニムとして存在しない場合は、エラーが返されます。

```
CONNECT fox@local
SELECT * FROM emp@redwood;
```

親トピック: 分散データベースの管理: 例

32.8.4 例4: パブリック接続ユーザーの共有データベース・リンクの作成

パブリック接続ユーザーの共有データベース・リンクの作成を示す例です。

次の例では、ローカル・データベースにneilとして接続し、ネット・サービス名sldbを使用してsalesデータベースへの共有パブリック・リンクを作成しています。ユーザーは、crazy/horseというユーザーID/パスワードによって認証されます。次の文は、パブリック接続ユーザーの共有データベース・リンクを作成します。

```
CONNECT neil@local
CREATE SHARED PUBLIC DATABASE LINK sales.division3.example.com
  AUTHENTICATED BY crazy IDENTIFIED BY horse
  USING 'sldb';
```

ローカル・サーバーに接続する各ユーザーは、この共有データベース・リンクを使用してリモート・データベースに接続し、対応するリモート・スキーマの表を問い合せることができます。

ローカルの共有サーバー・プロセスは、それぞれリモート・サーバーへの接続を1つずつ確立します。ローカルのサーバー・プロセスでsales.division3.example.comデータベース・リンクを介したリモート・サーバーへのアクセスが必要になると、たとえ接

続ユーザーが異なるユーザーであっても、そのたびにローカルの共有サーバー・プロセスは確立済のネットワーク接続を再利用します。

このデータベース・リンクが頻繁に使用されると、最終的にローカル・データベースのすべての共有サーバーがリモート接続を持つことになります。この時点で、新しいユーザーがこの共有データベース・リンクを使用しても、リモート・サーバーへの物理的な接続は新たに確立されません。

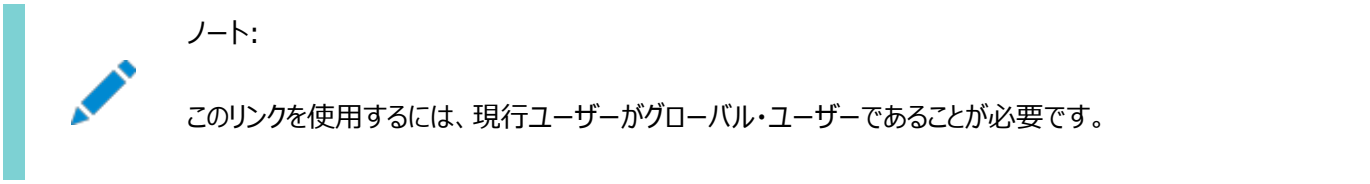
親トピック: 分散データベースの管理: 例

32.8.5 例5: パブリック現行ユーザーのデータベース・リンクの作成

パブリック現行ユーザーのデータベース・リンクの作成を示す例です。

次の例では、ローカル・データベースに接続ユーザーとして接続し、ネット・サービス名sldbを使用してsalesデータベースへのパブリック・リンクを作成しています。次の文は、パブリック現行ユーザーのデータベース・リンクを作成します。

```
CONNECT bart@local
CREATE PUBLIC DATABASE LINK sales.division3.example.com
  CONNECT TO CURRENT_USER
  USING 'sldb';
```



ノート:

このリンクを使用するには、現行ユーザーがグローバル・ユーザーであることが必要です。

このデータベース・リンクの結果は、次のとおりです。

scottがリモートのemp表から1行を削除するローカル・プロシージャfire_empを作成し、fire_empの実行権限をfordに付与したとします。

```
CONNECT scott@local_db
CREATE PROCEDURE fire_emp (enum NUMBER)
AS
BEGIN
   DELETE FROM emp@sales.division3.example.com
   WHERE empno=enum;
END;
GRANT EXECUTE ON fire_emp TO ford;
```

ここで、fordはローカル・データベースに接続してscottのプロシージャを実行したとします。

```
CONNECT ford@local_db
EXECUTE PROCEDURE scott.fire_emp (enum 10345);
```

fordがプロシージャscott.fire_empを実行するとき、プロシージャはscottの権限のもとで実行されます。現行ユーザー・データベース・リンクが使用されているため、接続はfordのリモート・スキーマではなく、scottのリモート・スキーマに確立されます。scottはグローバル・ユーザーである必要がありますが、fordはグローバル・ユーザーでなくてもかまいません。

ノート:

かわりに接続ユーザー・データベース・リンクが使用された場合、接続は ford のリモート・スキーマに確立されます。実行者権限と権限の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

scottのリモート・スキーマへの固定ユーザー・データベース・リンクを使用しても、同じ結果を得ることができます。

親トピック: 分散データベースの管理: 例

33 分散データベース・システムのアプリケーション開発

分散データベース・システムのアプリケーション開発には、アプリケーション・データの分布の管理、データベース・リンクによって確立される接続の制御、参照整合性の維持、分散問合せのチューニング、およびリモート・プロシージャのエラー処理などのタスクが含まれます。

- アプリケーション・データの分布の管理
  分散データベース環境では、データベース管理者と共同でデータの最適な格納場所を決めます。
- データベース・リンクにより確立される接続の制御
  SQL文やリモート・プロシージャ・コールの中でグローバル・オブジェクト名が参照されると、ローカル・ユーザーにかわってデータベース・リンクがリモート・データベース内のセッションへの接続を確立します。
- 分散システムの参照整合性の維持
  分散更新の一部が失敗したことを示すエラー・メッセージが返されていないかどうかをチェックするようにアプリケーションを設計します。失敗を検出した場合は、トランザクション全体をロールバックしてからアプリケーションの処理を進めるようにしてください。
- 分散問合せのチューニング
  ローカルのOracle Databaseサーバーは、分散問合せをそれに合った数のリモート問合せに分割し、次に、その問合せを実行するためにリモート・ノードに送信します。リモート・ノードは問合せを実行し、その結果をローカル・ノードに返します。その後、ローカル・ノードは必要な後処理を行い、ユーザーまたはアプリケーションに結果を戻します。
- リモート・プロシージャのエラー処理
  データベースがプロシージャを実行するとき、エラーが発生する可能性があります。

関連項目:

Oracle Database環境でのアプリケーション開発の詳細は、『Oracle Database開発ガイド』を参照してください

親トピック: 分散データベースの管理

33.1 アプリケーション・データの分散の管理

分散データベース環境では、データベース管理者と共同でデータの最適な格納場所を決めます。

その際、次の点について考慮します。

- 個々の場所から転送されるトランザクション数
- 各ノードで使用されるデータ(表の部分)の量
- パフォーマンス特性とネットワークの信頼性
- 各種ノードの速度とディスクの容量
- ノードまたはリンクが使用できないときの重要度
- 表間の参照整合性の必要性

親トピック: 分散データベース・システムのアプリケーション開発

33.2 データベース・リンクにより確立される接続の制御

SQL文やリモート・プロシージャ・コールの中でグローバル・オブジェクト名が参照されると、ローカル・ユーザーにかわってデータベース・リンクがリモート・データベース内のセッションへの接続を確立します。

リモートの接続とセッションは、それまでにローカルのユーザー・セッションに対して接続が確立されていない場合にのみ作成されます。

リモート・データベースとの間に確立された接続とセッションは、アプリケーションやユーザーによって明示的に終了されないかぎり、ローカル・ユーザーのセッションの間存続します。データベース・リンクを経由してSELECT文を発行すると、UNDOセグメントにトランザクション・ロックが設定されます。セグメントを再び解放するには、COMMIT文またはROLLBACK文を発行する必要があります。

アプリケーションで不要になった高コストの接続を切断するには、データベース・リンクを使用して確立されたリモート接続を終了することが有効です。リモートの接続とセッションを終了するには、CLOSE DATABASE LINK句を指定したALTER SESSION文を使用します。たとえば、次のトランザクションを発行した場合を考えます。

```
SELECT * FROM emp@sales;
COMMIT;
```

次の文は、salesデータベース・リンクが指すリモート・データベース内のセッションを終了します。

```
ALTER SESSION CLOSE DATABASE LINK sales;
```

ユーザー・セッションのデータベース・リンク接続をクローズするには、ALTER SESSIONシステム権限が必要です。

ノート:

データベース・リンクをクローズする前に、まずそのリンクを使用しているカーソルをすべてクローズし、次にそのリンクを使用している現行トランザクションがあればそのトランザクションを終了してください。

関連項目:

ALTER SESSION文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: 分散データベース・システムのアプリケーション開発

33.3 分散システムの参照整合性の維持

返されたエラー・メッセージをすべてチェックするようにアプリケーションを設計し、分散更新の一部が失敗したことを示すエラー・メッセージがないかを確認します。失敗を検出した場合は、トランザクション全体をロールバックしてからアプリケーションの処理を進めるようにしてください。

たとえば、整合性制約違反など、分散型の文の一部が失敗した場合、データベースはエラー番号ORA-02055を返します。以降の文またはプロシージャ・コールは、ROLLBACKまたはROLLBACK TO SAVEPOINTが発行されるまで、エラー番号ORA-02067を返します。

データベースでは、宣言参照整合性の制約を分散システムのノード間で定義することは許可されていません。つまり、1つの表での宣言参照整合性の制約では、リモート表の主キーまたは一意キーを参照する外部キーを指定することはできません。ただし、トリガーを使用してノード間の親子の表関係を維持することは可能です。

トリガーを使用して分散データベースの複数のノードにまたがる参照整合性を定義する場合、ネットワーク障害により、親表へのアクセスだけでなく子表へのアクセスも制限される可能性があることに注意してください。たとえば、子表がsalesデータベースに存在し、親表がhqデータベースに存在すると想定します。2つのデータベース間のネットワーク接続に障害が発生すると、参照整合性トリガーがhqデータベース内の親表へアクセスする必要があるため、子表に対するDML文が処理(子表に行を挿入したり、子表の外部キーの値を更新するような処理)を進めることができない場合があります。

関連項目:

トリガーを使用した参照整合性の規定の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

親トピック: 分散データベース・システムのアプリケーション開発

33.4 分散問合せのチューニング

ローカルのOracle Databaseサーバーは、分散問合せを対応する数のリモート問合せに分割し、次に、その問合せを実行するためにリモート・ノードに送信します。リモート・ノードは問合せを実行し、その結果をローカル・ノードに返します。その後、ローカル・ノードは必要な後処理を行い、ユーザーまたはアプリケーションに結果を戻します。

ノート:

SQL プロファイル、SQL 計画ベースライン、SQL パッチおよび格納されたアウトラインなどの SQL 管理オブジェクトは、問合せでデータベース・リンクを使用してリモート表を参照する場合に、想定したとおりに動作しないことがあります。たとえば、SQL 計画管理で、問合せに SQL 計画ベースラインを使用する場合、リモートで実行される問合せの部分は、SQL 計画ベースラインが作成されたときとは違う計画を使用することがあります。

- 連結インライン・ビューの使用
  分散問合せを最適化する最も効果的な方法は、リモート・データベースへのアクセスをできるだけ抑えて必要なデータのみを取得することです。
- コストベース最適化の使用
  コストベース最適化の使用には、クエリー・リライトやコストベース最適化の設定などのタスクの完了が含まれます。
- ヒントの使用
  ヒントにより、コストベース最適化の機能を拡張できます。
- 実行計画の分析
  分散問合せのチューニングの際に重要なこととして、実行計画の分析があります。

親トピック: 分散データベース・システムのアプリケーション開発

33.4.1 連結インライン・ビューの使用

分散問合せを最適化する最も効果的な方法は、リモート・データベースへのアクセスをできるだけ抑えて必要なデータのみを取得することです。

たとえば、分散問合せで5つのリモート表を2つの異なるリモート・データベースから参照し、複合フィルタ(`WHERE r1.salary + r2.salary > 50000`など)を使用するとします。この場合、リモート・データベースへのアクセスを1回にし、フィルタをリモート・サイトで適用するように問合せをリライトすることで、問合せのパフォーマンスを改善できます。このリライトにより、問合せを実行するサイトに転送されるデータの量が少なくなります。

リモート・データベースへのアクセスが1回になるように問合せをリライトするには、連結インライン・ビューを使用します。用語の定

義は次のとおりです。

- 連結

  同じデータベースにある2つ以上の表。

- インライン・ビュー

  親のSELECT文の表に置き換えられるSELECT文。次のカッコで囲まれた部分の埋込みSELECT文がインライン・ビューの例です。

```
SELECT e.empno,e.ename,d.deptno,d.dname
   FROM (SELECT empno, ename from
         emp@orc1.world) e, dept d;
```

- 連結インライン・ビュー

  1つのデータベースのみの複数の表からデータを選択するインライン・ビュー。これを使用すると、リモート・データベースにアクセスする時間が短縮されるため、分散問合せのパフォーマンスが向上します。

連結インライン・ビューを使用して分散問合せを構成し、分散問合せのパフォーマンスを改善することをお薦めします。Oracle Databaseのコストベース最適化を使用すると、分散問合せの多くが透過的にリライトされ、連結インライン・ビューによるパフォーマンスの向上が得られます。

親トピック: 分散問合せのチューニング

33.4.2 コストベース最適化の使用

コストベース最適化の使用には、クエリー・リライトやコストベース最適化の設定などのタスクの完了が含まれます。

- コストベース最適化の動作の仕組み
  最適化の主要タスクは、連結インライン・ビューを使用するように分散問合せをリライトすることです。
- コストベース最適化のためのクエリー・リライト
  連結インライン・ビューによるクエリー・リライトに加えて、コストベース最適化の方法を使用すると、参照先の表から収集される統計とオプティマイザが実行する計算に従って分散問合せが最適化されます。
- コストベース最適化の設定
  分散問合せのパフォーマンス改善のためにコストベース最適化を使用するようにシステムを設定すると、その処理がユーザーに対して透過的になります。つまり、問合せの発行時に自動的に最適化が実行されます。

親トピック: 分散問合せのチューニング

33.4.2.1 コストベース最適化の動作の仕組み

オプティマイザの主なタスクは、連結インライン・ビューを使用するように分散問合せをリライトすることです。

この最適化は、次の3つのステップで実行されます。

1. マージ可能なビューがすべてマージされます。
2. オプティマイザが連結問合せブロックのテストを実行します。
3. オプティマイザが連結インライン・ビューを使用して問合せをリライトします。

問合せがリライトされた後、その問合せが実行され、データ・セットがユーザーに返されます。

コストベース最適化をユーザーに対して透過的に実行する場合は、複数の分散問合せのパフォーマンスを改善することはできません。特に、次のものが分散問合せに含まれている場合、コストベース最適化は効果がありません。

- 集計
- 副問合せ
- 複合SQL

これらの要素のいずれかが分散問合せに含まれている場合は、問合せの修正方法と、分散問合せのパフォーマンスを改善するためのヒントの使用を学習するには、「ヒントの使用」を参照してください。

親トピック: コストベース最適化の使用

33.4.2.2 コストベース最適化の問合せのリライト

連結インライン・ビューによるクエリー・リライトに加えて、コストベース最適化の方法を使用すると、参照先の表から収集される統計とオプティマイザが実行する計算に従って分散問合せが最適化されます。

たとえば、コストベースの最適化によって次の問合せを分析します。この例では、表統計が使用できると仮定しています。CREATE TABLE文内の問合せが分析されます。

```
CREATE TABLE AS (
                  SELECT l.a, l.b, r1.c, r1.d, r1.e, r2.b, r2.c
                  FROM local l, remote1 r1, remote2 r2
                     WHERE l.c = r.c
                     AND r1.c = r2.c
                     AND r.e > 300
                 );
```

この文は、次のようにリライトされます。

```
CREATE TABLE AS (
                  SELECT l.a, l.b, v.c, v.d, v.e
                  FROM (
                         SELECT r1.c, r1.d, r1.e, r2.b, r2.c
                          FROM remote1 r1, remote2 r2
                          WHERE r1.c = r2.c
                          AND r1.e > 300
                       ) v, local l
                  WHERE l.c = r1.c
                 );
```

インライン・ビューに別名vが割り当てられ、前述のSELECT文で表としてこの別名を参照できます。連結インライン・ビューを作成すると、リモート・サイトで実行される問合せの量が減少するため、コストのかかるネットワーク通信量を削減できます。

親トピック: コストベース最適化の使用

33.4.2.3 コストベース最適化の設定

分散問合せのパフォーマンス改善のためにコストベース最適化を使用するようにシステムを設定すると、その処理がユーザーに対して透過的になります。つまり、問合せの発行時に自動的に最適化が実行されます。

- 環境の設定
  OPTIMIZER_MODE初期化パラメータを設定し、インスタンスに最適化アプローチを選択するためのデフォルト動作を設定します。
- 表の分析
  コストベース最適化で分散問合せのための最も効率的なパスが選択されるようにするには、問合せに関係する表の正確な統計を提供する必要があります。この操作には、DBMS_STATSパッケージを使用します。

親トピック: コストベース最適化の使用

33.4.2.3.1 環境を設定する

OPTIMIZER_MODE初期化パラメータを設定し、インスタンスに最適化アプローチを選択するためのデフォルト動作を設定します。

このパラメータは次の方法で設定できます。

- 初期化パラメータ・ファイルのOPTIMIZER_MODEパラメータを変更します。
- ALTER SESSION文を発行してセッション・レベルで設定します。

関連項目:

パラメータ・ファイルのOPTIMIZER_MODE初期化パラメータの設定と、コストベース最適化方法を使用するためのシステムの構成の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

親トピック: コストベース最適化の設定

33.4.2.3.2 表の分析

コストベース最適化で分散問合せのための最も効率的なパスが選択されるようにするには、問合せに関係する表の正確な統計を提供する必要があります。この操作には、DBMS_STATSパッケージを使用します。

ノート:

DBMS_STATS プロシージャを実行するには、表に対してローカルに接続する必要があります。

まずリモート・サイトに接続し、次に DBMS_STATS プロシージャを実行する必要があります。

次のDBMS_STATSプロシージャを使用すると、特定のクラスのオプティマイザ統計を収集できます。

- GATHER_INDEX_STATS
- GATHER_TABLE_STATS
- GATHER_SCHEMA_STATS
- GATHER_DATABASE_STATS

たとえば、分散トランザクションが日常的にscott.dept表にアクセスするとします。コストベースのオプティマイザが引き続き最適な方法を確実に選択するように、次の文を実行します。

```
BEGIN
   DBMS_STATS.GATHER_TABLE_STATS ('scott', 'dept');
END;
```

関連項目:

- 統計生成の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- DBMS_STATSパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: コストベース最適化の設定

33.4.3 ヒントの使用方法

ヒントにより、コストベース最適化の機能を拡張できます。

- ヒントの使用について
  文が十分に最適化されない場合は、ヒントを使用してコストベース最適化の機能を拡張できます。特に、独自の問合せを記述して連結インライン・ビューを使用する場合は、分散問合せがリライトされないようにコストベース・オプティマイザに指示を与えます。
- NO_MERGEヒントの使用
  NO_MERGEヒントは、データベースで連結されない可能性のあるSQL文にインライン・ビューがマージされないようにします。
- DRIVING_SITEヒントの使用
  DRIVING_SITEヒントを使用すると、問合せが実行されるサイトを指定できます。

親トピック: 分散問合せのチューニング

33.4.3.1 ヒントの使用について

文が十分に最適化されない場合は、ヒントを使用してコストベース最適化の機能を拡張できます。特に、独自の問合せを記述して連結インライン・ビューを使用する場合は、分散問合せがリライトされないようにコストベース・オプティマイザに指示を与えます。

また、データベース環境に関する特別な情報(統計、負荷、ネットワークおよびCPUの制限事項、分散問合せなど)を持っている場合は、ヒントを指定してコストベース最適化を適切に誘導できます。たとえば、データベース環境の情報に基づく連結インライン・ビューを使用して、独自に最適化した問合せを記述した場合は、NO_MERGEヒントを指定することにより、オプティマイザが問合せをリライトしないようにできます。

この手法は、分散問合せに集計、副問合せまたは複合SQLが含まれている場合に特に役立ちます。このタイプの分散問合せはオプティマイザによってリライトできないため、NO_MERGEを指定すると、オプティマイザは「コストベース最適化の動作の仕組み」で説明されているステップをスキップします。

DRIVING_SITEヒントを使用すると、リモート・サイトを問合せ実行サイトとして機能するように定義できます。この方法では、問合せがリモート・サイトで実行され、データがローカル・サイトに返されます。リモート・サイトにデータの大部分が格納されているときは、このヒントが特に役立ちます。

関連項目:

ヒントの使用の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

親トピック: ヒントの使用

33.4.3.2 NO_MERGEヒントの使用

NO_MERGEヒントは、データベースで連結されない可能性のあるSQL文にインライン・ビューがマージされないようにします。

このヒントはSELECT文に埋め込み、インライン・ビューを引数として使用するSELECT文の先頭、またはインライン・ビューを定義する問合せブロック内のいずれかに記述できます。

```
/* with argument */
SELECT /*+NO_MERGE(v)*/ t1.x, v.avg_y
   FROM t1, (SELECT x, AVG(y) AS avg_y FROM t2 GROUP BY x) v,
   WHERE t1.x = v.x AND t1.y = 1;
/* in query block */
```

```
SELECT t1.x, v.avg_y
   FROM t1, (SELECT /*+NO_MERGE*/ x, AVG(y) AS avg_y FROM t2 GROUP BY x) v,
   WHERE t1.x = v.x AND t1.y = 1;
```

通常、このヒントは、データベース環境の情報に基づいて最適化した問合せを作成したときに使用します。

関連トピック

- ヒントの使用

親トピック: ヒントの使用

33.4.3.3 DRIVING_SITEヒントの使用

DRIVING_SITEヒントを使用すると、問合せが実行されるサイトを指定できます。

コストベース最適化によって実行が行われる場所を決定することをお薦めしますが、オプティマイザを無効にする場合は、手動で実行サイトを指定できます。

DRIVING_SITEヒントを使用したSELECT文の例を次に示します。

```
SELECT /*+DRIVING_SITE(dept)*/ * FROM emp, dept@remote.com
   WHERE emp.deptno = dept.deptno;
```

関連トピック

- ヒントの使用

親トピック: ヒントの使用

33.4.4 実行計画の分析

分散問合せのチューニングの際に重要なこととして、実行計画の分析があります。

分析結果から得られるフィードバックは、データベースのテストと検証を行う上で重要な要素になります。計画を比較するときは、検証が特に重要になります。たとえば、コストベースの最適化によって分散問合せを最適化する実行計画を、ヒント、連結インライン・ビューおよびその他の方法を使用して問合せを手動で最適化する計画と比較します。

- 実行計画の生成
  実行計画を格納するデータベースの準備ができたら、指定した問合せの計画を参照できるようになります。SQL文を直接実行するかわりに、EXPLAIN PLAN FOR句に文を追加します。
- 実行計画の表示
  前述のSQL文を実行すると、実行計画がPLAN_TABLEに一時的に格納されます。

関連項目:

実行計画、EXPLAIN PLAN文およびその結果の解釈方法の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

親トピック: 分散問合せのチューニング

33.4.4.1 実行計画の生成

データベース内に実行計画を格納する場所を準備すると、指定した問合せの計画を表示するための準備が完了します。SQL文を直接実行するかわりに、EXPLAIN PLAN FOR句に文を追加します。

たとえば、次のような文を実行できます。

```
EXPLAIN PLAN FOR
   SELECT d.dname
   FROM dept d
      WHERE d.deptno
      IN (SELECT deptno
          FROM emp@orc2.world
          GROUP BY deptno
          HAVING COUNT (deptno) >3
          )
/
```

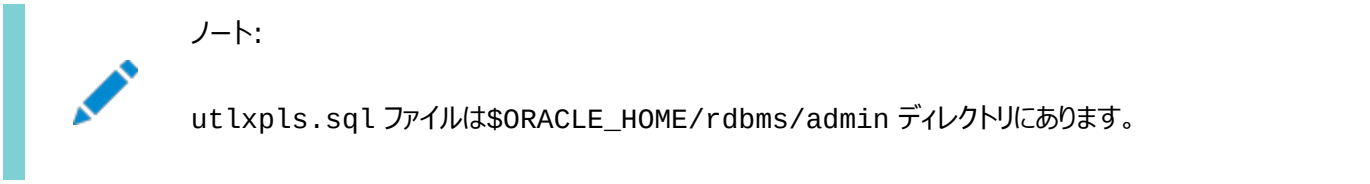
親トピック: 実行計画の分析

33.4.4.2 実行計画の表示

前述のSQL文を実行すると、実行計画がPLAN_TABLEに一時的に格納されます。

実行計画の結果を表示するには、次のスクリプトを実行します。

```
@utlxpls.sql
```



ノート:

utlxpls.sql ファイルは$ORACLE_HOME/rdbms/admin ディレクトリにあります。

utlxpls.sqlスクリプトを実行すると、指定したSELECT文の実行計画が表示されます。結果は、次のように書式化されます。

```
Plan Table
--------------------------------------------------------------------------------
| Operation                 |  Name    |  Rows | Bytes|  Cost  | Pstart| Pstop |
--------------------------------------------------------------------------------
| SELECT STATEMENT          |          |       |      |        |       |       |
|  NESTED LOOPS             |          |       |      |        |       |       |
|   VIEW                    |          |       |      |        |       |       |
|    REMOTE                 |          |       |      |        |       |       |
|   TABLE ACCESS BY INDEX RO|DEPT      |       |      |        |       |       |
|    INDEX UNIQUE SCAN      |PK_DEPT   |       |      |        |       |       |
--------------------------------------------------------------------------------
```

独自の連結インライン・ビューを記述するか、またはヒントを使用して、分散問合せを手動で最適化しようとする場合は、手動最適化の前および後に実行計画を生成するのが最適です。これら両方の実行計画を使用することで、手動最適化の効果を比較し、必要に応じて変更を加え、分散問合せのパフォーマンスを改善できます。

リモート・サイトで実行されるSQL文を表示するには、次のSELECT文を実行します。

```
SELECT OTHER
FROM PLAN_TABLE
   WHERE operation = 'REMOTE';
```

出力例を次に示します。

```
SELECT DISTINCT "A1"."DEPTNO" FROM "EMP" "A1"
   GROUP BY "A1"."DEPTNO" HAVING COUNT("A1"."DEPTNO")>3
```

ノート:

OTHER 列の内容全体がうまく表示されない場合は、次の SQL*Plus コマンドを実行してください。

```
SET LONG 9999999
```

親トピック: 実行計画の分析

33.5 リモート・プロシージャのエラー処理

データベースがプロシージャを実行するとき、エラーが発生する可能性があります。

データベースがプロシージャをローカルまたはリモートの位置で実行するときには、次の4種類の例外が発生する可能性があります。

- PL/SQLのユーザー定義例外。この例外は、EXCEPTIONキーワードを使用して宣言する必要があります。
- PL/SQLの事前定義例外。NO_DATA_FOUNDキーワードなど。
- SQLエラー。ORA-00900やORA-02015など。
- RAISE_APPLICATION_ERROR()プロシージャを使用して生成されるアプリケーション例外。

ローカル・プロシージャを使用するときは、次のような例外ハンドラを記述して、これらのメッセージをトラップできます。

```
BEGIN
  ...
EXCEPTION
   WHEN ZERO_DIVIDE THEN
   /* ... handle the exception */
END;
```

WHEN句には例外名が必要です。たとえば、RAISE_APPLICATION_ERRORで生成された例外など、例外が名前を持っていない場合は、PRAGMA_EXCEPTION_INITを使用して名前を割り当てることができます。たとえば:

```
DECLARE
  null_salary EXCEPTION;
  PRAGMA EXCEPTION_INIT(null_salary, -20101);
BEGIN
  ...
  RAISE_APPLICATION_ERROR(-20101, 'salary is missing');
...
EXCEPTION
  WHEN null_salary THEN
  ...
END;
```

リモート・プロシージャのコール中に、例外の処理をローカル・プロシージャの例外ハンドラで行うことができます。リモート・プロシージャはローカルのコール側プロシージャにエラー番号を返す必要があり、返すと前述の例に示すようにコール側プロシージャで例外が処理されます。PL/SQLのユーザー定義例外では、常に、ローカル・プロシージャにORA-06510が返されることに注意してください。

したがって、2つの異なるユーザー定義例外をエラー番号で区別することはできません。その他のリモート例外はすべて、ローカル例外と同じ方法で処理できます。

関連項目:

PL/SQLプロシージャの詳細は、『Oracle Database PL/SQL言語リファレンス』

親トピック: 分散データベース・システムのアプリケーション開発

34 分散トランザクションの概念

分散トランザクションは、分散データベースの2つ以上の異なるノード上のデータを更新します。

- 分散トランザクションの概要
  分散トランザクションには1つ以上の文が含まれており、それらは個別に、またはグループとして、分散データベースの2つ以上の異なるノード上のデータを更新します。
- 分散トランザクションのセッション・ツリー
  セッション・ツリーとは、セッション間の関係とセッションのロールを表す階層モデルです。
- 2フェーズ・コミット・メカニズム
  分散データベース環境では、データベースは自己完結ユニットとして分散トランザクションでの変更のコミットまたはロールバックを調整する必要があります。
- インダウト・トランザクション
  2フェーズ・コミット・メカニズムが失敗すると、トランザクションはインダウトになります。
- 分散トランザクション処理: 事例
  事例は、分散トランザクション処理を示しています。

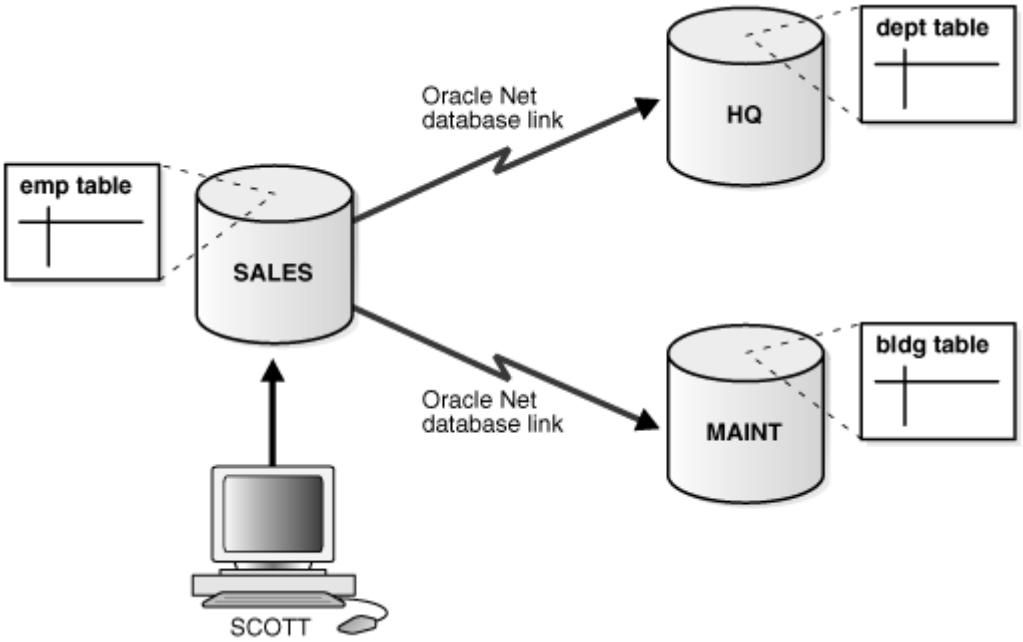
親トピック: 分散データベースの管理

34.1 分散トランザクションの概要

分散トランザクションは1つ以上の文からなり、それらが個別に、またはグループとして、分散データベースの複数ノードのデータを更新します。

たとえば、図34-1に示すデータベース構成を考えます。

図34-1 分散システム



scottによって実行される次の分散トランザクションは、ローカルのsalesデータベース、リモートのhqデータベース、およびリモートのmaintデータベースを更新します。

```
UPDATE scott.dept@hq.us.example.com
  SET loc = 'REDWOOD SHORES'
  WHERE deptno = 10;
```

```
UPDATE scott.emp
  SET deptno = 11
  WHERE deptno = 10;
UPDATE scott.bldg@maint.us.example.com
  SET room = 1225
  WHERE room = 1163;
COMMIT;
```

ノート:

トランザクションのすべての文が 1 つのリモート・ノードのみを参照している場合、そのトランザクションは分散トランザクションではなくリモート・トランザクションです。

分散トランザクションで許容される2つのタイプの操作は、DMLとDDLトランザクション、およびトランザクション制御文です。

- DMLおよびDDLトランザクション
  分散トランザクションでは、いくつかのDMLおよびDDL操作がサポートされます。
- トランザクション制御文
  分散トランザクションでは、いくつかのトランザクション制御文がサポートされます。

親トピック: 分散トランザクションの概念

34.1.1 DMLおよびDDLトランザクション

分散トランザクションでは、いくつかのDMLおよびDDL操作がサポートされます。

分散トランザクションでサポートされているデータ操作言語(DML)およびデータ定義言語(DDL)操作は、次のとおりです。

- CREATE TABLE AS SELECT
- DELETE
- INSERT(デフォルトおよびダイレクト・ロード)
- UPDATE
- LOCK TABLE
- SELECT
- SELECT FOR UPDATE

DML文およびDDL文はパラレルに実行でき、ダイレクト・ロード・インサート文はシリアルに実行できます。ただし、次の制限に注意してください。

- リモート操作はすべてSELECT文である必要があります。
- これらの文は、別の分散トランザクション内の句であってはいけません。
- INSERT、UPDATEまたはDELETE文のtable_expression_clauseで参照される表がリモートの場合、実行はパラレルではなくシリアルになります。
- パラレルDML/DDLまたはダイレクト・ロード・インサートの発行後にリモート操作を実行することはできません。
- XAまたはOCIを使用してトランザクションを開始した場合、そのトランザクションはシリアルに実行されます。
- パラレル操作の実行元であるトランザクションで、ループバック操作を実行することはできません。たとえば、実際はローカル・オブジェクトのシノニムであるリモート・オブジェクトを参照することはできません。

- トランザクションでSELECT以外の分散操作を実行する場合、DMLはパラレル化されません。

親トピック: 分散トランザクションの概要

34.1.2 トランザクション制御文

分散トランザクションでは、いくつかのトランザクション制御文がサポートされます。

サポートされているトランザクション制御文は、次のとおりです。

- `COMMIT`
- `ROLLBACK`
- `SAVEPOINT`

関連項目:

これらのSQL文の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: 分散トランザクションの概要

34.2 分散トランザクションのセッション・ツリー

セッション・ツリーとは、セッション間の関係とセッションのロールを表す階層モデルです。

- 分散トランザクションのセッション・ツリーについて
  分散トランザクションで文が発行されると、データベースはトランザクションに参加しているすべてのノードのセッション・ツリーを定義します。
- クライアント
  ノードは情報を別のノードのデータベースから参照するとき、クライアントとして機能します。
- データベース・サーバー
  データベース・サーバーは、クライアントがデータを要求するデータベースをホストするノードです。
- ローカル・コーディネータ
  分散トランザクションにおいて自身のトランザクション部分を完了するために別のノードのデータを参照する必要があるノードは、ローカル・コーディネータと呼ばれます。
- グローバル・コーディネータ
  分散トランザクションの実行元であるノードのことを、グローバル・コーディネータと呼びます。
- コミット・ポイント・サイト
  システム管理者は、常に1つのノードをコミット・ポイント・サイトとして指定します。

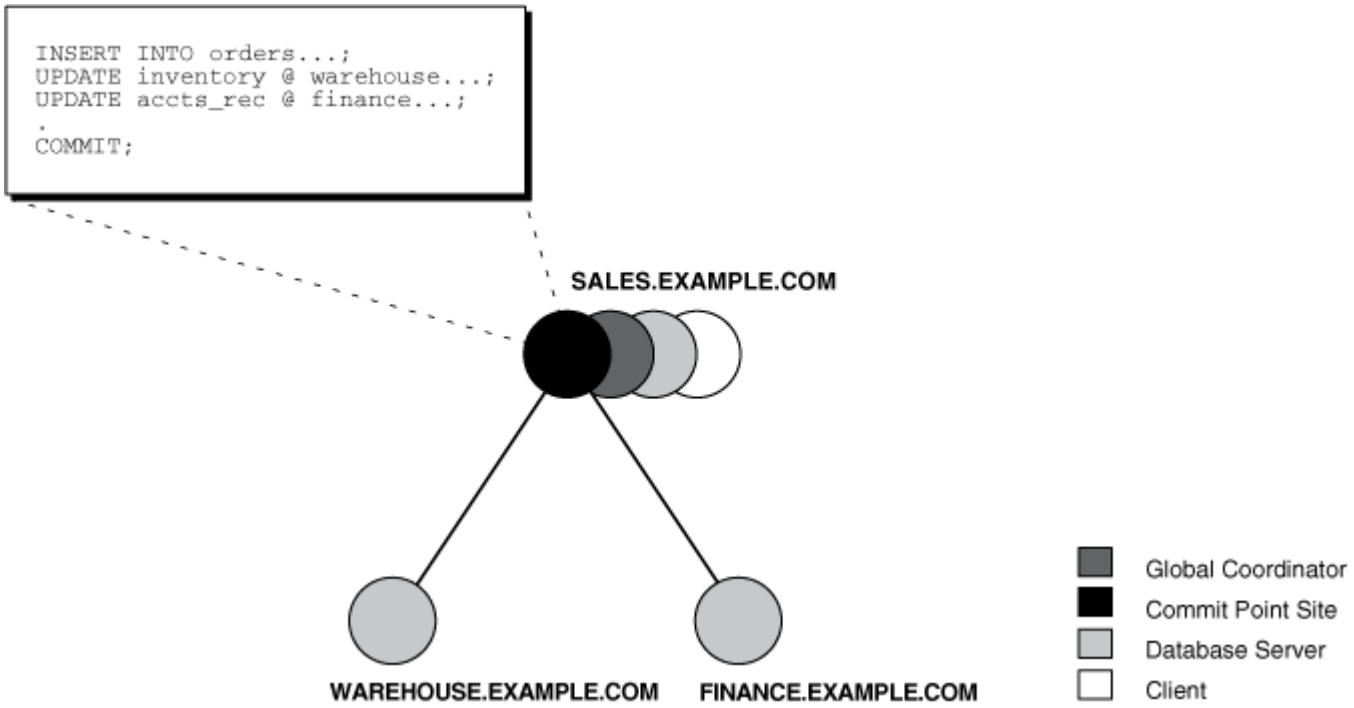
親トピック: 分散トランザクションの概念

34.2.1 分散トランザクションのセッション・ツリーについて

分散トランザクションで文が発行されると、データベースはトランザクションに参加しているすべてのノードのセッション・ツリーを定義します。

セッション・ツリーとは、セッション間の関係とセッションのロールを表す階層モデルです。セッション・ツリーの例を図34-2に示します。

図34-2 セッション・ツリーの例



分散トランザクションのセッション・ツリーに参加しているすべてのノードは、次に示すロールを1つ以上持ちます。

ロール	説明
クライアント	異なるノードに属するデータベース内の情報を参照するノード。
データベース・サーバー	別のノードからの情報の要求を受け取るノード。
グローバル・コーディネータ	分散トランザクションの実行元ノード。
ローカル・コーディネータ	他のノードのデータを強制的に参照して、自身のトランザクション部分を完了するノード。
コミット・ポイント・サイト	グローバル・コーディネータの指示に従ってトランザクションをコミットまたはロールバックするノード。

分散トランザクションのノードが果たすロールは、次の条件によって決まります。

- トランザクションがローカルとリモートのどちらであるか。
- ノードのコミット・ポイント強度(「コミット・ポイント・サイト」を参照)。
- 要求されたすべてのデータがノードで使用可能か、またはトランザクションを完了するために他のノードを参照する必要があるか。
- ノードが読取り専用かどうか。

親トピック: 分散トランザクションのセッション・ツリー

34.2.2 クライアント

情報を別のノードのデータベースから参照するとき、ノードはクライアントとして機能します。

参照先のノードはデータベース・サーバーです。図34-2のノードsalesは、warehouseデータベースおよびfinanceデータ

ベースが稼働しているノードのクライアントです。

親トピック: 分散トランザクションのセッション・ツリー

34.2.3 データベース・サーバー

データベース・サーバーは、クライアントがデータを要求する要求先データベースが稼働しているノードです。

図34-2では、salesノードのアプリケーションは、warehouseノードおよびfinanceノードのデータにアクセスする分散トランザクションを開始します。したがって、sales.example.comはクライアント・ノードのロールを持ち、warehouseおよびfinanceはどちらもデータベース・サーバーのロールを持ちます。この例では、アプリケーションでsalesデータベース内のデータも変更されているため、salesはデータベース・サーバーおよびクライアントです。

親トピック: 分散トランザクションのセッション・ツリー

34.2.4 ローカル・コーディネータ

分散トランザクションにおいて自身のトランザクション部分を完了するために別のノードのデータを参照する必要があるノードは、ローカル・コーディネータと呼ばれます。

図34-2で、salesは、直接参照しているノードwarehouseおよびfinanceを調整しているため、ローカル・コーディネータです。また、ノードsalesは、トランザクションに関係しているすべてのノードを調整しているため、グローバル・コーディネータでもあります。

ローカル・コーディネータは、自身が直接やり取りするノードの間で次のようにトランザクションを調整する役目を果たします。

- これらのノード間でトランザクションのステータス情報を受け渡します。
- これらのノードに問合せを渡します。
- これらのノードから問合せを受け取り、他のノードに渡します。
- 問合せの結果を開始元のノードに返します。

親トピック: 分散トランザクションのセッション・ツリー

34.2.5 グローバル・コーディネータ

分散トランザクションの実行元であるノードのことを、グローバル・コーディネータと呼びます。

分散トランザクションを発行するデータベース・アプリケーションは、グローバル・コーディネータとして機能しているノードに直接接続します。たとえば、図34-2では、ノードsalesで発行されたトランザクションは、データベース・サーバーwarehouseおよびfinanceの情報を参照します。したがって、sales.example.comは、この分散トランザクションのグローバル・コーディネータになります。

グローバル・コーディネータは、セッション・ツリーの親またはルートになります。グローバル・コーディネータは、分散トランザクション処理中に次の操作を実行します。

- 分散トランザクションのすべてのSQL文やリモート・プロシージャ・コールなどを参照先のノードに直接送り、それによってセッション・ツリーを構成します。
- コミット・ポイント・サイト以外のすべての直接参照先ノードに対して、トランザクションの準備をするように指示します。
- すべてのノードの準備が正常に完了した場合に、トランザクションのグローバル・コミットを開始するようにコミット・ポイント・サイトに対して指示します。

- 終了の応答があった場合に、トランザクションのグローバル・ロールバックを開始するようにすべてのノードに対して指示します。

親トピック: 分散トランザクションのセッション・ツリー

34.2.6 コミット・ポイント・サイト

システム管理者は、常に1つのノードをコミット・ポイント・サイトとして指定します。

- コミット・ポイント・サイトについて
  コミット・ポイント・サイトの役割は、グローバル・コーディネータの指示に従ってコミットまたはロールバックの操作を開始することです。
- 分散トランザクションのコミットの仕組み
  分散トランザクションがコミットされたとみなされるのは、コミット・ポイント以外のすべてのサイトで準備が完了し、実際にトランザクションがコミット・ポイント・サイトでコミットされた後です。
- コミット・ポイント強度
  データベース・サーバーには、必ずコミット・ポイント強度を割り当てる必要があります。データベース・サーバーが分散トランザクション内で参照される場合、そのコミット・ポイント強度の値によって、2フェーズ・コミットにおける役割が決まります。

親トピック: 分散トランザクションのセッション・ツリー

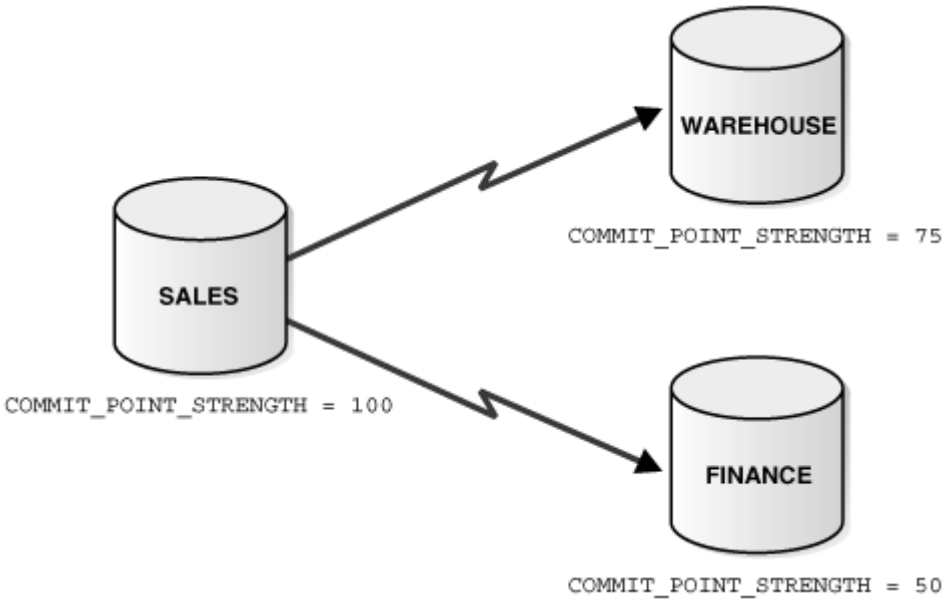
34.2.6.1 コミット・ポイント・サイトについて

コミット・ポイント・サイトの役割は、グローバル・コーディネータの指示に従ってコミットまたはロールバックの操作を開始することです。

システム管理者は、すべてのノードにコミット・ポイント強度を割り当てることで、セッション・ツリー内のノードの1つをコミット・ポイント・サイトとして必ず指定します。コミット・ポイント・サイトには、最も重要なデータを格納するノードを選択してください。

図34-3は、salesがコミット・ポイント・サイトとして機能している分散システムの例です。

図34-3 コミット・ポイント・サイト



コミット・ポイント・サイトは、分散トランザクションに関係する他のすべてのノードと次の点で区別されます。

- コミット・ポイント・サイトが準備完了状態に入ることはありません。そのため、コミット・ポイント・サイトに最も重要なデータが格納されている場合は、たとえ障害が起きたとしても、データがインダウトのままになることはありません。障害が発生すると、障害を起こしたノードは準備完了状態のままになり、インダウト・トランザクションが解決されるまで必要なロック

が保持されます。

- コミット・ポイント・サイトは、トランザクションに関係している他のノードよりも先にコミットします。実際は、コミット・ポイント・サイトでの分散トランザクションの結果によって、すべてのノードでのトランザクションがコミットされるかロールバックされるかが決まり、他のノードはコミット・ポイント・サイトの指示に従います。グローバル・コーディネータは、すべてのノードでコミット・ポイント・サイトと同様にトランザクションが完了することを保証します。

親トピック: コミット・ポイント・サイト

34.2.6.2 分散トランザクションのコミットの仕組み

分散トランザクションは、コミット・ポイント以外のすべてのサイトで準備が完了した後、コミットされたとみなされますが、実際には、トランザクションはコミット・ポイント・サイトで先にコミットされています。

コミット・ポイント・サイトのREDOログは、このノードで分散トランザクションがコミットされるとただちに更新されます。

コミット・ポイント・ログにはコミットの記録があります。そのため、たとえ参加中のノードの一部がまだ準備完了状態であり、それらのノードで実際にトランザクションがコミットされていない場合であっても、トランザクションはコミットされたとみなされます。同様に、コミット・ポイント・サイトでコミットが記録されていない場合は、分散トランザクションはコミットされていないとみなされます。

親トピック: コミット・ポイント・サイト

34.2.6.3 コミット・ポイント強度

データベース・サーバーには、必ずコミット・ポイント強度を割り当てる必要があります。データベース・サーバーが分散トランザクション内で参照される場合、そのコミット・ポイント強度の値によって、2フェーズ・コミットにおける役割が決まります。

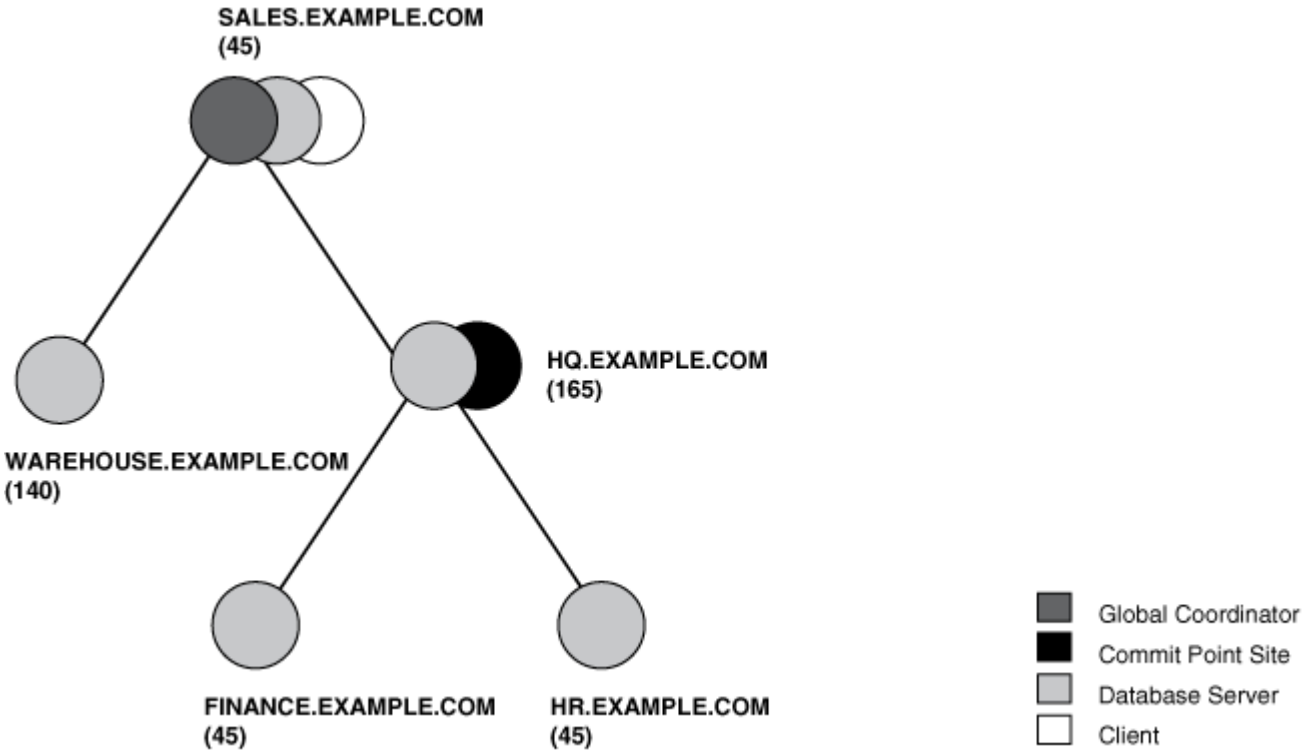
具体的には、このコミット・ポイント強度によって、どのノードが分散トランザクションのコミット・ポイント・サイトになり、他のすべてのノードより前にコミットするかが決まります。この値を指定するには、初期化パラメータ`COMMIT_POINT_STRENGTH`を使用します。ここでは、データベースがコミット・ポイント・サイトを決定する仕組みについて説明します。

準備フェーズの冒頭で決定されるコミット・ポイント・サイトは、トランザクションに参加しているノードの中からのみ選択されます。次に示す一連のイベントが発生します。

1. データベースは、グローバル・コーディネータが直接参照しているノードの中で、最も高いコミット・ポイント強度を持つノードをコミット・ポイント・サイトとして選択します。
2. 最初に選択されたノードは、このトランザクションの情報を取得する必要のあるノードの中で、自身よりも高いコミット・ポイント強度を持っているノードがないかを判断します。
3. トランザクションで直接参照しているノードの中で最も高いコミット・ポイント強度を持つノードか、またはそのノードのサーバーの中でより高いコミット・ポイント強度を持つもののどちらかが、コミット・ポイント・サイトになります。
4. 最終的なコミット・ポイント・サイトが決定した後、グローバル・コーディネータは、トランザクションに参加しているすべてのノードに準備応答を送ります。

図34-4は、各ノードのコミット・ポイント強度(カッコ内の値)を示したセッション・ツリーの例です。また、コミット・ポイント・サイトとして選択されたノードも示しています。

図34-4 コミット・ポイント強度とコミット・ポイント・サイトの決定



コミット・ポイント・サイトを決定するときは、次の条件が適用されます。

- 読取り専用ノードは、コミット・ポイント・サイトにはなれません。
- グローバル・コーディネータが直接参照している複数のノードが同じコミット・ポイント強度を持っている場合、データベースはそれらのうちの1つをコミット・ポイント・サイトとして指定します。
- 分散トランザクションがロールバックで終了する場合は、準備フェーズとコミット・フェーズは不要です。したがって、データベースはコミット・ポイント・サイトを決定しません。そのかわりに、グローバル・コーディネータはROLLBACK文をすべてのノードに送り、分散トランザクションの処理を終了します。

図34-4のように、コミット・ポイント・サイトとグローバル・コーディネータがセッション・ツリーの異なるノードになることもあります。各ノードのコミット・ポイント強度は、最初に接続が確立されたときにコーディネータに渡されます。コーディネータは、2フェーズ・コミット時のコミット・ポイント・サイトを効率的に選択するために、自身が直接やり取りしている各ノードのコミット・ポイント強度を保持しています。そのため、コミットが発生するたびにコーディネータとノード間でコミット・ポイント強度を交換する必要がありません。

関連項目:

- ノードのコミット・ポイント強度の設定方法を学習するには、「ノードのコミット・ポイント強度の指定」
- 初期化パラメータCOMMIT_POINT_STRENGTHの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: コミット・ポイント・サイト

34.3 2フェーズ・コミット・メカニズム

分散データベース環境では、データベースは自己完結ユニットとして分散トランザクションでの変更のコミットまたはロールバックを調整する必要があります。

- 2フェーズ・コミット・メカニズムについて
  ローカル・データベースのトランザクションとは異なり、分散トランザクションには複数のデータベースでのデータの変更が伴います。そのため、データベースは、トランザクションの変更のコミットまたはロールバックを自己完結単位として調整する

必要があり、分散トランザクションの処理はより複雑になります。言い換えれば、トランザクション全体がコミットするか、またはトランザクション全体がロールバックするかのどちらかの結果になります。

- 準備フェーズ
  準備フェーズは、分散トランザクションのコミットにおける最初のフェーズです。
- コミット・フェーズ
  コミット・フェーズは、分散トランザクションのコミットにおける2番目のフェーズです。このフェーズになる前に、分散トランザクションで参照されている、コミット・ポイント・サイト以外のすべてのノードが準備完了していること、つまり、トランザクションのコミットに必要なリソースを確保していることが保証されます。
- 情報消去フェーズ
  参加ノードが自身のコミットの完了をコミット・ポイント・サイトに通知した後、コミット・ポイント・サイトは、トランザクションに関する情報を消去できます。

親トピック: 分散トランザクションの概念

34.3.1 2フェーズ・コミット・メカニズムについて

ローカル・データベースのトランザクションとは異なり、分散トランザクションには複数のデータベースでのデータの変更が伴います。そのため、データベースは、トランザクションの変更のコミットまたはロールバックを自己完結単位として調整する必要があり、分散トランザクションの処理はより複雑になります。言い換えれば、トランザクション全体がコミットするか、またはトランザクション全体がロールバックするかのどちらかの結果になります。

データベースは、2フェーズ・コミット・メカニズムを使用することで、分散トランザクションにおけるデータの整合性を保証します。準備フェーズでは、トランザクション内の開始ノードが他の参加ノードに対して、トランザクションをコミットまたはロールバックすることを確約するように要求します。コミット・フェーズでは、開始ノードがすべての参加ノードに対して、トランザクションをコミットするように要求します。この結果が達成できない場合、すべてのノードはロールバックするように要求されます。

分散トランザクションに参加しているすべてのノードは、必ず同じ動作を実行します。つまり、ノードすべてがトランザクションをコミットするか、またはノードすべてがトランザクションをロールバックします。データベースは、分散トランザクションのコミットまたはロールバックを自動的に制御および監視しており、2フェーズ・コミット・メカニズムを使用してグローバル・データベース(トランザクションに参加しているデータベースの集まり)の整合性を維持します。このメカニズムは完全に透過的であり、ユーザーやアプリケーション開発者の側でプログラミングを行う必要はまったくありません。

コミット・メカニズムは、次の各フェーズで構成されています。データベースは、ユーザーが分散トランザクションをコミットすると、必ずこれらのフェーズを自動的に実行します。

フェーズ	説明
準備フェーズ	グローバル・コーディネータと呼ばれる開始ノードは、コミット・ポイント・サイト以外の参加ノードに対して、たとえ障害が起きた場合でもトランザクションをコミットまたはロールバックすることを確約するように要求します。準備ができないノードがある場合、トランザクションはロールバックされます。
コミット・フェーズ	すべての参加ノードが準備完了の応答をコーディネータに伝えると、コーディネータは、コミット・ポイント・サイトにコミットを要求します。コミット・ポイント・サイトのコミット後、コーディネータは、トランザクションをコミットするように他のすべてのノードに要求します。

フェーズ	説明
情報消去フェーズ	グローバル・コーディネータは、トランザクションに関する情報を消去します。

親トピック: 2フェーズ・コミット・メカニズム

34.3.2 準備フェーズ

準備フェーズは、分散トランザクションのコミットにおける最初のフェーズです。

- 準備フェーズについて
  分散トランザクションのコミットにおける最初のフェーズは、準備フェーズです。
- 準備フェーズでの応答のタイプ
  準備するように指示されたノードは、異なる方法で応答できます。
- 準備フェーズのステップ
  2フェーズ・コミット・プロセスの準備フェーズには、特定のステップが含まれています。

親トピック: 2フェーズ・コミット・メカニズム

34.3.2.1 準備フェーズについて

準備フェーズは、分散トランザクションのコミットにおける1番目のフェーズです。

このフェーズでは、データベースがトランザクションを実際にコミットまたはロールバックすることはありません。ここでは、分散トランザクションで参照されているすべてのノード(「コミット・ポイント・サイト」で説明されているコミット・ポイント・サイトを除く)がコミットの準備を指示されます。ノードは、準備を完了するために次の処理を実行します。

- REDOログの情報を記録し、それ以降障害が発生してもトランザクションをコミットまたはロールバックできるようにします。
- 変更済の表に分散ロックを設定し、読取りを防ぎます。

各ノードは、コミットの準備が完了したという応答をグローバル・コーディネータに伝えることによって、その後トランザクションをコミットまたはロールバックすることを確約しますが、トランザクションをコミットまたはロールバックするという決定を各ノードが一方的に下すわけではありません。この確約の意味は、この時点でインスタンス障害が発生した場合、ノードはオンライン・ログのREDOレコードを使用してデータベースをリカバリし、準備フェーズに戻すことができるということです。

ノート:

ノードの準備完了後に発行した問合せは、すべてのフェーズが完了するまで、関連するロック済データにアクセスできません。この時間は、障害が発生しないかぎり問題にはなりません(「インダウト・トランザクションの処理方法の決定」を参照)。

親トピック: 準備フェーズ

34.3.2.2 準備フェーズでの応答のタイプ

準備を指示されたノードは、様々な方法で応答できます。

レスポンス	意味

レスポンス	意味
準備完了	ノードのデータの変更が分散トランザクション内の文によって完了しており、ノードの準備が正常に完了しています。
読取り専用	ノードで変更されるデータがない(問合せのみ)か、または変更できないので、準備は不要です。
中止	ノードが正常に準備できません。

- 準備完了応答
  ノードの準備が正常に完了すると、ノードは準備完了メッセージを発行します。
- 読取り専用応答
  ノードが準備を要求され、データベースに影響を及ぼすSQL文がノードのデータを変更しない場合、ノードは読取り専用メッセージで応答します。
- 中止応答
  中止メッセージにより、特有な処理が生じます。

親トピック: 準備フェーズ

34.3.2.2.1 準備応答

ノードの準備が正常に完了すると、ノードは準備完了メッセージを発行します。

このメッセージは、ノードの変更レコードがオンライン・ログに格納されており、ノードでコミットまたはロールバックのどちらかを実行できる準備が整っていることを示します。また、このメッセージによって、トランザクションに対して保持されているロックが障害発生時にも残ることが保証されます。

親トピック: 準備フェーズでの応答のタイプ

34.3.2.2.2 読取り専用応答

ノードが準備を要求されたときに、データベースにアクセスするSQL文がノードのデータを変更しない場合、ノードは読取り専用メッセージで応答します。

このメッセージは、ノードがコミット・フェーズに参加しないことを示します。

分散トランザクションの全部または一部が読取り専用になるケースとして、次の3つがあります。

ケース	条件	結果
一部読取り専用	次のいずれかが発生した場合 ● 1 つ以上のノードで問合せのみが発行された。 ● データが変更されない。 ● トリガーの起動または制約違反のために変更がロールバックされ	読取り専用ノードは、準備を要求されたときに自身のステータスを認識します。読取り専用ノードは、読取り専用応答をローカル・コーディネータに伝えます。この結果、データベースは読取り専用ノードを以降の処理から除外するので、コミット・フェーズがより高速に完了します。

ケース	条件	結果
	た。	
準備フェーズでの完全な読取り専用	次のすべてが発生した場合 ● データが変更されない。 ● トランザクションが SET TRANSACTION READ ONLY 文で開始されていない。	準備フェーズ時にすべてのノードが読取り専用であることを認識するので、コミット・フェーズは不要になります。グローバル・コーディネータにはすべてのノードが読取り専用かどうかわからないため、グローバル・コーディネータは引き続き準備フェーズを実行する必要があります。
2 フェーズ・コミットなしの完全な読取り専用	次のすべてが発生した場合 ● データが変更されない。 ● トランザクションが SET TRANSACTION READ ONLY 文で開始されている。	このトランザクションでは問合せのみが許可されるため、グローバル・コーディネータは、2 フェーズ・コミットを実行する必要がありません。また、システム変更番号(SCN)の調整がノード間でグローバルに行われるため、他のトランザクションによる変更によってグローバル・トランザクション・レベルの読込み一貫性が損なわれることはありません。このトランザクションでは、UNDO セグメントは使用されません。

分散トランザクションが読取り専用に設定されている場合、そのトランザクションでUNDOセグメントは使用されません。多数のユーザーがデータベースに接続していて、ユーザーのトランザクションがREAD ONLYに設定されていない場合は、それらのトランザクションが問合せを実行するのみであっても、UNDO領域が割り当てられます。

親トピック: 準備フェーズでの応答のタイプ

34.3.2.2.3 中止応答

中止メッセージにより、特有な処理が生じます。

ノードは、正常に準備できないときに次の処理を実行します。

1. トランザクションが現在保持しているリソースを解放し、トランザクションのローカル部分をロールバックします。
2. 分散トランザクション内で自身を参照しているノードに対し、異常終了メッセージで応答します。

これらの処理は、分散トランザクションに関係している他のノードに伝播します。これにより、他のノードはトランザクションをロールバックできるので、グローバル・データベースのデータの整合性が保証されます。この応答によって、「トランザクションに関係しているすべてのノードが同じ物理時間でトランザクションをすべてコミットするか、またはすべてロールバックする」という分散トランザクションの基本原則が守られます。

親トピック: 準備フェーズでの応答のタイプ

34.3.2.3 準備フェーズのステップ

2フェーズ・コミット・プロセスの準備フェーズには、特定のステップが含まれています。

準備フェーズを完了するために、コミット・ポイント・サイトを除く各ノードは次のステップを実行します。

1. ノードは、自身の子(以降参照する各ノード)に対して、コミットの準備をするように要求します。

2. ノードは、トランザクションによって自分自身のデータまたは子のデータが変更されるかどうかをチェックします。データが変更されない場合、ノードは残りのステップを省略し、読取り専用応答を返します(「読取り専用応答」を参照)。
3. データが変更される場合、ノードはトランザクションのコミットに必要なリソースを割り当てます。
4. ノードは、トランザクションによる変更に対応するREDOレコードをREDOログに保存します。
5. ノードは、トランザクションに対して保持されているロックが障害発生時にも残ることを保証します。
6. ノードは、開始ノードに準備完了応答を返すか(「準備完了応答」を参照)、それ自体またはいずれかの子が準備の試行に失敗した場合は、中止応答を返します(「中止応答」を参照)。

これらの処理によって、ノードが後で自身のトランザクションをコミットまたはロールバックできることが保証されます。この後、準備完了ノードは、グローバル・コーディネータからCOMMITまたはROLLBACK要求を受け取るまで待機します。

ノードの準備が完了すると、分散トランザクションはインダウトと呼ばれる状態になります(「インダウト・トランザクション」を参照)。すべての変更がコミットまたはロールバックされるまで、インダウト状態のままです。

親トピック: 準備フェーズ

34.3.3 コミット・フェーズ

コミット・フェーズは、分散トランザクションのコミットにおける2番目のフェーズです。このフェーズになる前に、分散トランザクションで参照されている、コミット・ポイント・サイト以外のすべてのノードが準備完了していること、つまり、トランザクションのコミットに必要なリソースを確保していることが保証されます。

- コミット・フェーズのステップ
  2フェーズ・コミット・プロセスのコミット・フェーズには、固有のステップが含まれています。
- グローバル・データベースの一貫性の保証
  コミットされた各トランザクションには、そのトランザクション内部のSQL文によって行われた変更を一意に識別するためのシステム変更番号(SCN)が対応付けられます。

親トピック: 2フェーズ・コミット・メカニズム

34.3.3.1 コミット・フェーズのステップ

2フェーズ・コミット・プロセスのコミット・フェーズには、固有のステップが含まれています。

コミット・フェーズは次のステップで構成されています。

1. グローバル・コーディネータは、コミット・ポイント・サイトにコミットを指示します。
2. コミット・ポイント・サイトは、コミットを実行します。
3. コミット・ポイント・サイトは、コミットが完了したことをグローバル・コーディネータに伝えます。
4. グローバル・コーディネータおよびローカル・コーディネータは、すべてのノードに対してトランザクションをコミットするように指示するメッセージを送ります。
5. 各ノードで、データベースは分散トランザクションのローカル部分をコミットし、ロックを解放します。
6. 各ノードで、データベースは、トランザクションのコミットが完了したことを示す追加のREDOエントリをローカルREDOログに記録します。
7. 各参加ノードは、自身のコミットが完了したことをグローバル・コーディネータに伝えます。

コミット・フェーズが完了するときは、分散システムの全ノードのデータについて一貫性が保たれています。

親トピック: コミット・フェーズ

34.3.3.2 グローバル・データベースの一貫性の保証

コミットされた各トランザクションには、そのトランザクション内部のSQL文によって行われた変更を一意に識別するためのSCNが対応付けられます。

SCNは、データベースのコミット済バージョンを一意に識別する内部的なタイムスタンプの役割を持ちます。

分散システムでは、次の処理がすべて発生したときに、通信中のノードのSCNが調整されます。

- 1つ以上のデータベース・リンクによって表されるパスを使用した接続の確立
- 分散SQL文の実行
- 分散トランザクションのコミット

特に、分散システムのノード間でSCNが調整されることにより、文とトランザクションの両方のレベルでグローバルな読込み一貫性が保証されるという利点があります。必要であれば、グローバルな時間ベースのリカバリを実行することもできます。

準備フェーズ時に、データベースは、トランザクションに関係しているすべてのノードで最も高いSCNを判断します。次に、コミット・ポイント・サイトにおいて最も高いSCNでトランザクションをコミットします。さらに、すべての準備完了ノードに対して、コミットの指示とともにコミットSCNを送ります。

関連項目:

読込み一貫性におけるタイム・ラグ問題の管理の詳細は、「読込み一貫性の管理」を参照してください

親トピック: コミット・フェーズ

34.3.4 情報消去フェーズ

参加ノードが自身のコミットの完了をコミット・ポイント・サイトに通知した後、コミット・ポイント・サイトは、トランザクションに関する情報を消去できます。

次のステップが発生します。

1. すべてのノードのコミットが完了したことをグローバル・コーディネータから通知された後、コミット・ポイント・サイトは、このトランザクションに関するステータス情報を消去します。
2. コミット・ポイント・サイトは、ステータス情報を消去したことをグローバル・コーディネータに伝えます。
3. グローバル・コーディネータは、トランザクションに関する自分自身の情報を消去します。

親トピック: 2フェーズ・コミット・メカニズム

34.4 インダウト・トランザクション

2フェーズ・コミット・メカニズムが失敗すると、トランザクションはインダウトになります。

- インダウト・トランザクションについて
  2フェーズ・コミット・メカニズムは、すべてのノードが一斉にコミットするか、またはロールバックを実行することを保証します。システムやネットワークのエラーのために、3つのフェーズのいずれかが失敗した場合はどうなるのでしょうか。この場合、トランザクションはインダウトになります。
- インダウト・トランザクションの自動解決

ほとんどの場合、インダウト・トランザクションはデータベースにより自動的に解決されます。たとえば、次の使用例においてlocalとremoteの2つのノードがあるとします。ローカル・ノードはコミット・ポイント・サイトです。ユーザーscottは、localに接続してlocalとremoteを更新する分散トランザクションを実行し、コミットします。

- インダウト・トランザクションの手動解決
  場合によっては、インダウト・トランザクションを手動で解決する必要があります。
- インダウト・トランザクションのシステム変更番号の関連性
  システム変更番号(SCN)は、データベースのコミット済バージョンの内部タイムスタンプです。Oracle Databaseサーバーは、SCNクロック値を使用することでトランザクションの一貫性を保証します。

親トピック: 分散トランザクションの概念

34.4.1 インダウト・トランザクションについて

2フェーズ・コミット・メカニズムは、すべてのノードがコミットされるかまたはロールバックを一斉に実行することを保証します。システムやネットワークのエラーのために、3つのフェーズのいずれかが失敗した場合はどうなるのでしょうか。この場合、トランザクションはインダウトになります。

分散トランザクションは、次の要因によってインダウトになる可能性があります。

- Oracle Databaseソフトウェアを実行しているサーバー・システムがクラッシュした。
- 分散処理に関係している複数のOracle Database間のネットワーク接続が切断された。
- 未処理のソフトウェア・エラーが発生した。

システム、ネットワークまたはソフトウェアの問題が解決されると、RECOプロセスによってインダウト・トランザクションが自動的に解決されます。RECOがトランザクションを解決できるまで、データは読取りおよび書込みの両方についてロックされます。読取りをブロックするのは、データベースが問合せに対してどのバージョンのデータを表示すればよいかを判断できないためです。

親トピック: インダウト・トランザクション

34.4.2 インダウト・トランザクションの自動解決

データベースでは、多くの場合、インダウト・トランザクションは自動的に解決されます。たとえば、次の使用例においてlocalとremoteの2つのノードがあるとします。ローカル・ノードはコミット・ポイント・サイトです。ユーザーscottは、localに接続してlocalとremoteを更新する分散トランザクションを実行し、コミットします。

- 準備フェーズ中の障害
  例は、2フェーズ・トランザクションの準備フェーズ中に障害が発生した場合に踏まれるステップを示しています。
- コミット・フェーズ中の障害
  例は、2フェーズ・トランザクションのコミット・フェーズ中に障害が発生した場合に踏まれるステップを示しています。

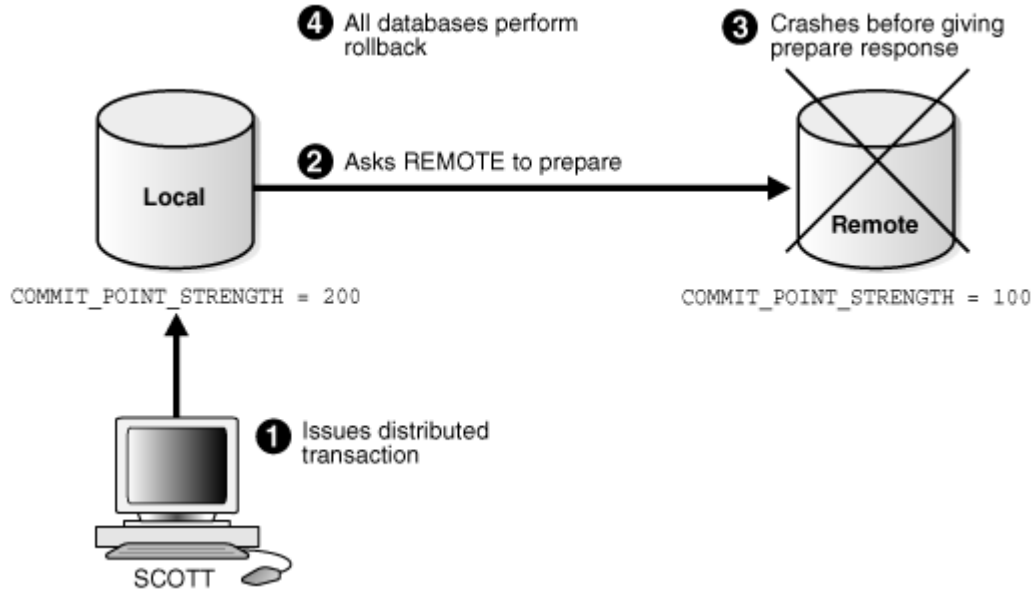
親トピック: インダウト・トランザクション

34.4.2.1 準備フェーズ中の障害

例は、2フェーズ・トランザクションの準備フェーズ中に障害が発生した場合に踏まれるステップを示しています。

図34-5は、分散トランザクションの準備フェーズ中に障害が発生したときの一連のイベントを示しています。

図34-5 準備フェーズ中の障害



次のステップが発生します。

1. ユーザーSCOTTはLocalに接続して分散トランザクションを実行します。
2. グローバル・コーディネータ(この例ではコミット・ポイント・サイトを兼務)は、コミット・ポイント・サイト以外のすべてのデータベースに対して、コミットまたはロールバックの指示があったときにその動作を実行することを確約するように要求します。
3. remoteデータベースが、localに準備応答を発行する前にクラッシュします。
4. トランザクションは、リモート・サイトのリストア時に、RECOプロセスによって各データベースで最終的にロールバックされます。

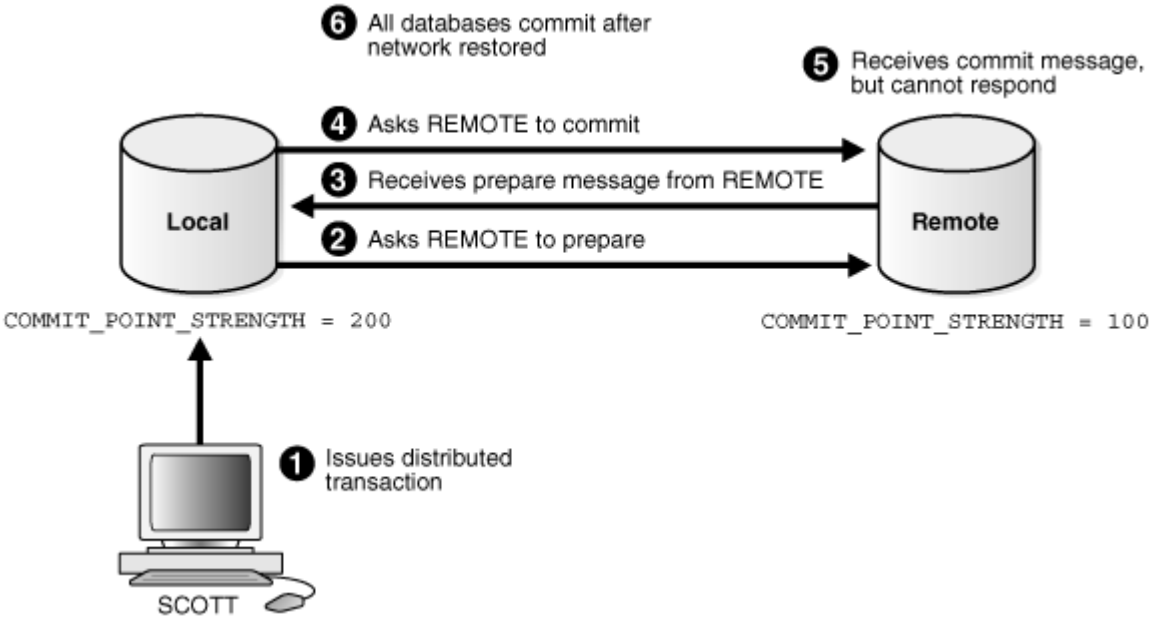
親トピック: インダウト・トランザクションの自動解決

34.4.2.2 コミット・フェーズ中の障害

例は、2フェーズ・トランザクションのコミット・フェーズ中に障害が発生した場合に踏まれるステップを示しています。

図34-6は、分散トランザクションのコミット・フェーズ中に障害が発生したときの一連のイベントを示しています。

図34-6 コミット・フェーズ中の障害



次のステップが発生します。

1. ユーザーScottはlocalに接続して分散トランザクションを実行します。
2. グローバル・コーディネータ(この場合はコミット・ポイント・サイトを兼務)は、コミット・ポイント・サイト以外のすべてのデータベースに対して、コミットまたはロールバックの指示があったときにその動作を実行することを確約するように要求します。
3. コミット・ポイント・サイトは、コミットの確約を示す準備完了メッセージをremoteから受け取ります。
4. コミット・ポイント・サイトはトランザクションをローカルにコミットし、それからコミット・メッセージをremoteに送ってコミットを要求します。
5. remoteデータベースはコミット・メッセージを受け取りましたが、ネットワーク障害のために応答できません。
6. トランザクションは、ネットワークがリストアされた後、RECOプロセスによってリモート・データベースで最終的にコミットされます。

関連項目:

障害状況の説明と、2フェーズ・コミット中に障害が発生した場合のデータベースによる解決方法の詳細は、「インダウト・トランザクションの処理方法の決定」

親トピック: インダウト・トランザクションの自動解決

34.4.3 インダウト・トランザクションの手動解決

場合によっては、インダウト・トランザクションを手動で解決する必要があります。

次の場合のみ、インダウト・トランザクションを手動で解決する必要があります。

- インダウト・トランザクションが重要なデータまたはUNDOセグメントをロックしている場合。
- システム、ネットワークまたはソフトウェアの障害の原因をすぐに修復できない場合。

インダウト・トランザクションの解決が複雑になる場合があります。その場合は、次の手順を実行する必要があります。

- インダウト・トランザクションのトランザクション識別番号を特定します。
- DBA_2PC_PENDINGビューおよびDBA_2PC_NEIGHBORSビューを問い合せ、トランザクションに関係しているデータベースのコミットが完了しているかどうかを確認します。
- 必要であれば、COMMIT FORCE文を使用してコミットを強制実行するか、またはROLLBACK FORCE文を使用してロールバックを強制実行します。

関連項目:

インダウト・トランザクションの解決方法の詳細は、次の項を参照してください。

- 「インダウト・トランザクションの処理方法の決定」
- 「手動によるインダウト・トランザクションのオーバーライド」

親トピック: インダウト・トランザクション

34.4.4 インダウト・トランザクションのシステム変更番号の関連性

SCNは、コミット済バージョンのデータベースの内部的なタイムスタンプです。Oracle Databaseサーバーは、SCNクロック値を

使用することでトランザクションの一貫性を保証します。

たとえば、ユーザーがトランザクションをコミットするとき、データベースはそのコミットのSCNをREDOログに記録します。

データベースは、SCNを使用して、異なるデータベース間での分散トランザクションを調整します。たとえば、データベースは、次のときにSCNを使用します。

1. アプリケーションがデータベース・リンクを使用して接続を確立するとき。
2. 分散トランザクションが、それに関係しているすべてのデータベースの中で最も高いグローバルSCNでコミットされるとき。
3. コミット・グローバルSCNが、トランザクションに関係しているすべてのデータベースに送られるとき。

SCNは、トランザクションが失敗した場合も含め、トランザクションの同期化されたコミット・タイムスタンプとして機能するため、分散トランザクションにとって非常に重要な存在です。トランザクションがインダウトになった場合、管理者はこのSCNを使用して、グローバル・データベースへの変更を調整できます。トランザクション・コミットのグローバルSCNは、分散リカバリの実行時など、後でトランザクションの識別に使用することもできます。

親トピック: インダウト・トランザクション

34.5 分散トランザクション処理: 事例

事例は、分散トランザクション処理を示しています。

- 分散トランザクション処理の事例について
  この使用例では、ある会社が`sales.example.com`および`warehouse.example.com`という異なるOracle Databaseサーバーを持っています。ユーザーが売上レコードを`sales`データベースに挿入すると、対応付けられたレコードが`warehouse`データベースで更新されます。
- 第1段階: クライアント・アプリケーションによるDML文の発行
  例は、分散トランザクション処理の第1段階を示しています。
- 第2段階: Oracle Databaseによるコミット・ポイント・サイトの判別
  例は、分散トランザクション処理の第2段階を示しています。
- 第3段階: グローバル・コーディネータによる準備応答の送信
  例は、分散トランザクション処理の第3段階を示しています。
- 第4段階: コミット・ポイント・サイトによるコミット
  例は、分散トランザクション処理の第4段階を示しています。
- 第5段階: コミット・ポイント・サイトによるグローバル・コーディネータへのコミットの通知
  例は、分散トランザクション処理の第5段階を示しています。
- 第6段階: グローバルおよびローカル・コーディネータによる全ノードへのコミットの要求
  例は、分散トランザクション処理の第6段階を示しています。
- 第7段階: グローバル・コーディネータとコミット・ポイント・サイトによるコミットの完了
  例は、分散トランザクション処理の第7段階を示しています。

親トピック: 分散トランザクションの概念

34.5.1 分散トランザクション処理の事例について

この使用例では、ある会社が`sales.example.com`および`warehouse.example.com`という異なるOracle Databaseサーバーを持っています。ユーザーが売上レコードを`sales`データベースに挿入すると、対応付けられたレコードが`warehouse`データベースで更新されます。

この分散処理の事例では、次のことを示します。

- セッション・ツリーの定義。
- コミット・ポイント・サイトがどのように決定されるか。
- 準備メッセージがいつ送られるか。
- トランザクションがいつ実際にコミットされるか。
- トランザクションに関するどのような情報がローカルに格納されるか。

親トピック: 分散トランザクション処理: 事例

34.5.2 第1段階: クライアント・アプリケーションによるDML文の発行

例は、分散トランザクション処理の第1段階を示しています。

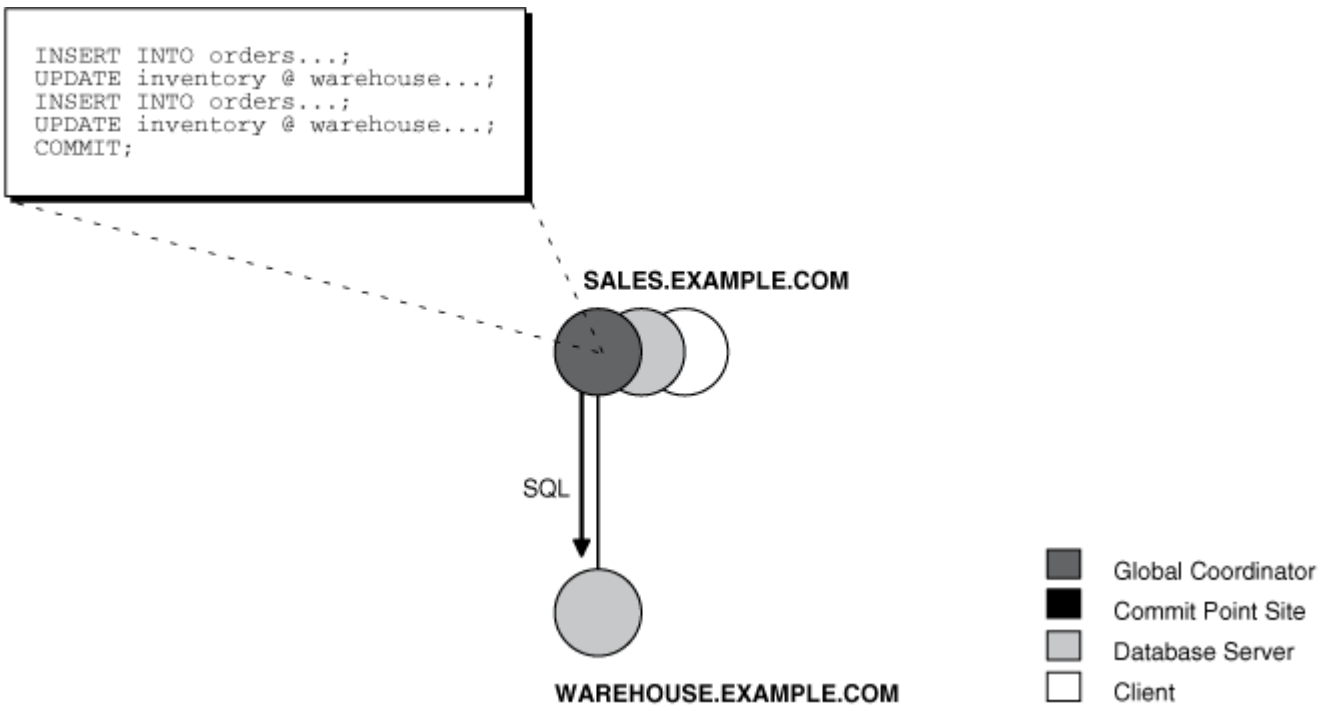
営業部門で、営業担当がSQL*Plusを使用して売上注文を入力し、コミットします。アプリケーションは次のようなSQL文を発行し、salesデータベースに注文を入力して、warehouseデータベースのinventory表を更新します。

```
CONNECT scott@sales.example.com ...;
INSERT INTO orders ...;
UPDATE inventory@warehouse.example.com ...;
INSERT INTO orders ...;
UPDATE inventory@warehouse.example.com ...;
COMMIT;
```

これらのSQL文は単一の分散トランザクションの一部であり、発行されるすべてのSQL文がユニットとして成功または失敗することが保証されています。文をユニットとして扱うことにより、注文が設定されていても、注文を反映するように在庫が更新されていないという事態を防ぐことができます。実際は、トランザクションによってグローバル・データベースのデータの一貫性が保証されています。

トランザクションの各SQL文が実行されると、図34-7のようにセッション・ツリーが定義されます。

図34-7 セッション・ツリーの定義



トランザクションの次の点に注意してください。

- トランザクションを開始するのは、salesデータベースで実行されている注文入力アプリケーションです。したがって、分散トランザクションのグローバル・コーディネータは、sales.example.comになります。
- 注文入力アプリケーションは、新しい売上レコードをsalesデータベースに挿入し、warehouseデータベースのinventory表を更新します。したがって、ノードsales.example.comとwarehouse.example.comはどちらもデータベース・サーバーになります。
- sales.example.comはinventory表を更新するため、warehouse.example.comのクライアントになります。

この段階では、この分散トランザクションのセッション・ツリーの定義が完了します。ツリー内の各ノードは、必要なデータ・ロックを獲得して、ローカル・データを参照するSQL文を実行します。これらのロックは、SQL文の実行が完了した後も、2フェーズ・コミットが完了するまで保持されます。

親トピック: 分散トランザクション処理: 事例

34.5.3 第2段階: Oracle Databaseによるコミット・ポイント・サイトの判別

例は、分散トランザクション処理の第2段階を示しています。

データベースは、COMMIT文の直後にコミット・ポイント・サイトを判別します。図34-8のように、グローバル・コーディネータのsales.example.comがコミット・ポイント・サイトとして判別されます。

関連項目:

コミット・ポイント・サイトの判別の詳細は、「コミット・ポイント強度」

図34-8 コミット・ポイント・サイトの判別

親トピック: 分散トランザクション処理: 事例

34.5.4 第3段階: グローバル・コーディネータによる準備応答の送信

例は、分散トランザクション処理の第3段階を示しています。

準備段階では、次のステップが実行されます。

1. データベースがコミット・ポイント・サイトを判別した後、グローバル・コーディネータは、コミット・ポイント・サイトを除くセッション・ツリーの直接参照先のノードすべてに準備メッセージを送ります。この例では、準備を要求されるノードはwarehouse.example.comのみです。
2. ノードwarehouse.example.comは準備を試みます。トランザクション内のローカルに独立した部分をコミットし、自身のローカルREDOログにコミット情報を記録できることをノードが保証できれば、そのノードは正常に準備できます。こ

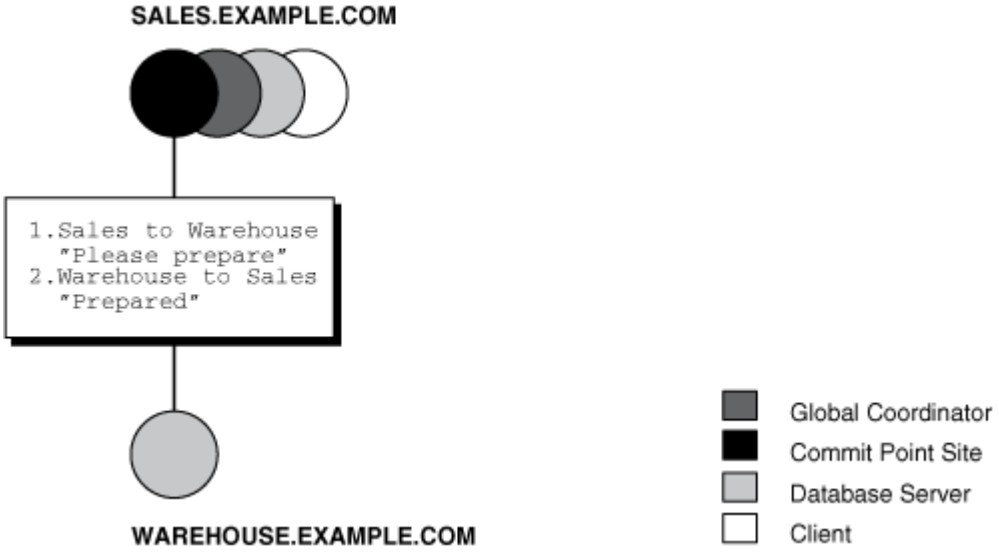
の例では、sales.example.comがコミット・ポイント・サイトなので、warehouse.example.comのみが準備メッセージを受け取ります。

3. ノードwarehouse.example.comは、準備完了メッセージでsales.example.comに応答します。

各ノードが準備を終えると、準備を要求した側のノードに応答メッセージが送り返されます。応答に応じて、次のいずれかの動作が発生します。

- 準備を要求されたノードのいずれかが、グローバル・コーディネータに対して異常終了メッセージで応答した場合、グローバル・コーディネータはすべてのノードに対してトランザクションのロールバックを指示し、操作が完了します。
- 準備を要求されたすべてのノードがグローバル・コーディネータに対して準備完了メッセージまたは読取り専用メッセージで応答した場合、つまり、正常に準備できた場合は、グローバル・コーディネータがコミット・ポイント・サイトに対してトランザクションのコミットを要求します。

図34-9 準備メッセージの送信と確認



親トピック: 分散トランザクション処理: 事例

34.5.5 第4段階: コミット・ポイント・サイトによるコミット

例は、分散トランザクション処理の第4段階を示しています。

コミット・ポイント・サイトによるトランザクションのコミットでは、次のステップが実行されます。

1. ノードsales.example.comは、warehouse.example.comの準備が完了しているという確認を受け取り、トランザクションをコミットするようにコミット・ポイント・サイトに指示します。
2. この時点で、コミット・ポイント・サイトはトランザクションをローカルにコミットし、この操作を自身のローカルREDOログに記録します。

warehouse.example.comがまだコミットを完了していない場合でも、このトランザクションの結果は事前に決まっています。言い換えれば、指定したノードのコミット機能が遅延したとしても、トランザクションはすべてのノードでコミットされます。

親トピック: 分散トランザクション処理: 事例

34.5.6 第5段階: コミット・ポイント・サイトによるグローバル・コーディネータへのコミットの通知

例は、分散トランザクション処理の第5段階を示しています。

この段階では、次のステップが実行されます。

1. コミット・ポイント・サイトは、トランザクションがコミットされたことをグローバル・コーディネータに伝えます。この例ではコミット・ポイント・サイトとグローバル・コーディネータが同一のノードであるため、操作は必要ありません。コミット・ポイント・サイトは、トランザクションがコミットされたことを自身のオンライン・ログに記録しているため、この事実を把握しています。
2. グローバル・コーディネータは、分散トランザクションに関係している他のすべてのノードでトランザクションがコミットされたことを確認します。

親トピック: 分散トランザクション処理: 事例

34.5.7 第6段階: グローバルおよびローカル・コーディネータによる全ノードへのコミットの要求

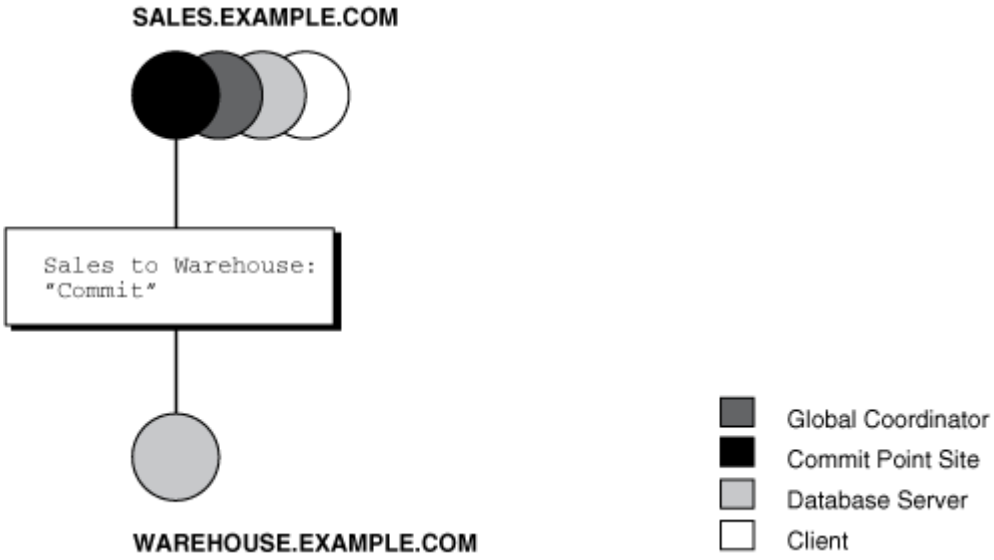
例は、分散トランザクション処理の第6段階を示しています。

トランザクション内のすべてのノードによるトランザクションのコミットでは、次のステップが実行されます。

1. コミット・ポイント・サイトでのコミットがグローバル・コーディネータに通知された後、グローバル・コーディネータは、直接参照している他のすべてのノードに対してコミットを指示します。
2. コミットの指示を受けたローカル・コーディネータは、次に自身のサーバーに対してコミットを指示し、以下同様にコミットの指示が伝播されます。
3. グローバル・コーディネータを含む各ノードは、トランザクションをコミットし、該当するREDOログ・エントリをローカルに記録します。各ノードのコミットが完了すると、そのトランザクションのためにローカルに保持されていたリソース・ロックが解放されます。

図34-10では、コミット・ポイント・サイトとグローバル・コーディネータを兼務しているsales.example.comが、トランザクションのローカルのコミットをすでに完了しています。この時点でsalesはwarehouse.example.comに対してトランザクションのコミットを指示します。

図34-10 ノードへのコミットの指示



親トピック: 分散トランザクション処理: 事例

34.5.8 第7段階: グローバル・コーディネータとコミット・ポイント・サイトによるコミットの完了

例は、分散トランザクション処理の第7段階を示しています。

トランザクションのコミットの完了は、次のステップで実行されます。

1. すべての参照先ノードとグローバル・コーディネータがトランザクションのコミットを完了した後、グローバル・コーディネータはこれをコミット・ポイント・サイトに通知します。
2. このメッセージを待っていたコミット・ポイント・サイトは、この分散トランザクションに関するステータス情報を消去します。
3. コミット・ポイント・サイトは、完了したことをグローバル・コーディネータに伝えます。つまり、コミット・ポイント・サイトは、分散トランザクションのコミットに関する情報を保持しません。2フェーズ・コミットに関係しているすべてのノードでトランザクションのコミットが正常に完了すると、それらのノードで今後ノード自身のステータスを判断する必要はないため、このような処理が許可されます。
4. グローバル・コーディネータは、トランザクション自体に関する情報を消去することでトランザクションを完了します。

COMMITフェーズの完了後、分散トランザクションそのものが完了します。これまで説明したステップは、1秒以内に自動的に実行されます。

親トピック: 分散トランザクション処理: 事例

35 分散トランザクションの管理

分散トランザクションの管理には、ノードのコミット・ポイント強度の指定、トランザクションの命名、インダウト・トランザクションの管理などのタスクが含まれます。

- ノードのコミット・ポイント強度の指定
  分散トランザクションでどのノードが最初にコミットされるかは、最も高いコミット・ポイント強度を持つデータベースによって決まります。
- トランザクションの命名
  トランザクションに名前を付けることができます。この機能は、特定の分散トランザクションを識別する際に便利であり、同じ目的のCOMMIT COMMENT文にかわるものです。
- 分散トランザクション情報の表示
  各データベースのデータ・ディクショナリには、オープンしているすべての分散トランザクションに関する情報が格納されています。データ・ディクショナリの表とビューを使用すると、トランザクションに関する情報を取得できます。
- インダウト・トランザクションの処理方法の決定
  2フェーズ・コミットのいずれかの側面で障害が発生すると、トランザクションはインダウトになります。Oracle Database ソフトウェアを実行しているサーバー・システムのクラッシュ、分散処理に含まれる2つ以上のOracle Database間のネットワーク接続の切断、または未処理のソフトウェア・エラーの発生により、分散トランザクションはインダウトになります。
- 手動によるインダウト・トランザクションのオーバーライド
  FORCEオプションを含むCOMMITまたはROLLBACK文を使用し、コミットするインダウト・トランザクションのローカルまたはグローバル・トランザクションIDを示すテキスト文字列を含めます。
- データ・ディクショナリからの保留中の行のパージ
  インダウト・トランザクション用のデータ・ディクショナリから保留中の行をパージできます。
- インダウト・トランザクションの手動コミット: 例
  手動でインダウト・トランザクションをコミットする例を示します。
- ロックによるデータ・アクセスの障害
  SQL文を発行すると、データベースは文を正常に実行するために必要なリソースのロックを試みます。しかし、要求されたデータが、コミットされていない他のトランザクションの文によって現在保持されていて、長時間ロックされたままになっていると、タイムアウトが発生します。
- 分散トランザクション障害のシミュレーション
  自動的にトランザクションのローカル部分を解決するRECOを観察するため、または手動でインダウト分散トランザクションを解決してその結果を観察するために、分散トランザクションの障害を強制できます。
- 読込み一貫性の管理
  Oracle Databaseの分散読込み一貫性の実装には重要な制限事項があります。

親トピック: 分散データベースの管理

35.1 ノードのコミット・ポイント強度の指定

分散トランザクションでどのノードが最初にコミットされるかは、最も高いコミット・ポイント強度を持つデータベースによって決まります。

各ノードのコミット・ポイント強度を指定するときは、準備フェーズまたはコミット・フェーズ中に障害が発生した場合に最も重要なサーバーがブロックされないようにしてください。ノードのコミット・ポイント強度は、COMMIT_POINT_STRENGTH初期化パラメータによって指定します。

デフォルト値は、オペレーティング・システムによって異なります。値の範囲は、0(ゼロ)から255までの任意の整数です。たとえば、データベースのコミット・ポイント強度を200に設定するには、そのデータベースの初期化パラメータ・ファイルに次の行を追加します。

```
COMMIT_POINT_STRENGTH = 200
```

コミット・ポイント強度は、分散トランザクションでコミット・ポイント・サイトを決定する際にのみ使用されます。

データベースのコミット・ポイント強度を設定するときは、次の点を考慮してください。

- コミット・ポイント・サイトにはトランザクションのステータスに関する情報が格納されます。他のノードがトランザクションのステータスに関する情報を必要とする場合に備えて、コミット・ポイント・サイトには、信頼性が低下したり、使用できなくなったりする頻度の高いノードを使用しないでください。
- データベースのコミット・ポイント強度は、データベース内の重要な共有データの量に比例するように設定してください。たとえば、メインフレーム・コンピュータのデータベースは、一般にPCのデータベースよりも多くのデータをユーザー間で共有しています。したがって、メインフレームのコミット・ポイント強度は、PCのコミット・ポイント強度よりも高い値に設定します。

関連項目:

コミット・ポイントの概要については、「コミット・ポイント・サイト」

親トピック: 分散トランザクションの管理

35.2 トランザクションの命名

トランザクションの命名が可能です。この機能は、特定の分散トランザクションを識別する際に便利であり、同じ目的のCOMMIT COMMENT文にかわるものです。

トランザクションに命名するにはSET TRANSACTION...NAME文を使用します。たとえば:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
     NAME 'update inventory checkpoint 0';
```

この例は、SERIALIZABLEに等しい分離レベルを持つ新しいトランザクションをユーザーが開始し、それに'update inventory checkpoint 0'という名前を付けたことを示しています。

分散トランザクションでは、トランザクションがコミットされるときに、名前が参加サイトに送られます。トランザクション名が存在する場合は、COMMIT COMMENTがあっても無視されます。

トランザクション名は、V$TRANSACTIONビューのNAME列に表示され、トランザクションがコミットされると、DBA_2PC_PENDINGビューのTRAN_COMMENTフィールドに表示されます。

親トピック: 分散トランザクションの管理

35.3 分散トランザクション情報の表示

各データベースのデータ・ディクショナリには、オープンしているすべての分散トランザクションに関する情報が格納されています。データ・ディクショナリの表とビューを使用すると、トランザクションに関する情報を取得できます。

- 準備完了トランザクションのID番号と状態の判断
  特定のトランザクションIDのグローバル・コミット番号を確認するには、DBA_2PC_PENDINGビューを使用します。このグローバル・コミット番号は、インダウト・トランザクションを手動で解決するときに使用できます。

- インダウト・トランザクションのセッション・ツリーのトレース
  DBA_2PC_NEIGHBORSビューには、リモート・クライアントからの受信インダウト・トランザクションと、リモート・サーバーへの送信インダウト・トランザクションが表示されます。

親トピック: 分散トランザクションの管理

35.3.1 準備完了トランザクションのID番号と状態の判断

特定のトランザクションIDのグローバル・コミット番号を確認するには、DBA_2PC_PENDINGビューを使用します。このグローバル・コミット番号は、インダウト・トランザクションを手動で解決するときに使用できます。

次のビューには、ローカル・データベースで定義され、データ・ディクショナリに格納されているデータベース・リンクが表示されます。

ビュー	用途
DBA_2PC_PENDING	インダウト分散トランザクションがすべてリストされます。このビューには、インダウト・トランザクションが移入されるまで何もデータが入っていません。トランザクションが解決された後、ビューはパージされます。

関連性の最も高い列を次の表に示します(ビューのすべての列の詳細は、『Oracle Databaseリファレンス』を参照してください)。

表35-1 DBA_2PC_PENDING

列	説明
LOCAL_TRAN_ID	「integer.integer.integer」という書式のローカル・トランザクション識別子。 ノート: 接続の LOCAL_TRAN_ID と GLOBAL_TRAN_ID が同じである場合、そのノードはトランザクションのグローバル・コーディネータです。
GLOBAL_TRAN_ID	global_db_name.db_hex_id.local_tran_id という書式のグローバル・データベース識別子で、db_hex_id はデータベースを一意に識別するために使用する 8 文字の 16 進数値です。この共通のトランザクション ID は、分散トランザクションのすべてのノードで同一です。 ノート: 接続の LOCAL_TRAN_ID と GLOBAL_TRAN_ID が同じである場合、そのノードはトランザクションのグローバル・コーディネータです。
STATE	STATE に可能な値は、次のとおりです。 ● Collecting 通常、このカテゴリは、グローバル・コーディネータまたはローカル・コーディネータにのみ適用されます。ノードは現在他のデータベース・サーバーから情報を収集中であり、その後ノード自身が準備できるかどうかが判断されます。

列	説明
	● 準備完了 ノードは準備を完了していますが、そのことをローカル・コーディネータに準備完了メッセージで通知しているかどうかは不明です。しかし、コミット要求はまだ受け取っていません。ノードは準備完了状態にあり、トランザクションのコミットに必要なローカル・リソースのロックをすべて保持しています。 ● Committed ノード(任意のタイプ)はトランザクションのコミットを完了していますが、トランザクションに関係している他のノードが同じように完了していない可能性があります。つまり、トランザクションは 1 つ以上のノードでまだ保留しています。 ● Forced Commit ペンディング・トランザクションは、データベース管理者の判断で強制的にコミットできます。このエントリは、ローカル・ノードでトランザクションが手動でコミットされた場合に表示されます。 ● Forced termination (rollback) ペンディング・トランザクションは、データベース管理者の判断で強制的にロールバックできます。このエントリは、ローカル・ノードでこのトランザクションが手動でロールバックされた場合に表示されます。
MIXED	YES は、トランザクションの一部があるノードではコミットされ、別のノードではロールバックされたことを示します。
TRAN_COMMENT	トランザクションがコミットされると、トランザクション・コメントまたはトランザクション名(トランザクションに命名している場合)がこの列に設定されます。
HOST	ホスト・システムの名前。
COMMIT#	コミットされたトランザクションのグローバル・コミット番号。

DBA_2PC_PENDINGの関連情報を問い合せるには、次のスクリプトpending_txn_scriptを実行します(出力例も含まれています)。

```
COL LOCAL_TRAN_ID FORMAT A13
COL GLOBAL_TRAN_ID FORMAT A30
COL STATE FORMAT A8
COL MIXED FORMAT A3
COL HOST FORMAT A10
COL COMMIT# FORMAT A10
SELECT LOCAL_TRAN_ID, GLOBAL_TRAN_ID, STATE, MIXED, HOST, COMMIT#
   FROM DBA_2PC_PENDING
```

```
/
SQL> @pending_txn_script
LOCAL_TRAN_ID GLOBAL_TRAN_ID                   STATE    MIX HOST       COMMIT#
------------- ------------------------------ -------- --- ---------- ----------
1.15.870      HQ.EXAMPLE.COM.ef192da4.1.15.870  commit   no  dlsun183   115499
```

この出力は、ローカル・トランザクション1.15.870がこのノードではコミットを完了しているものの、他の1つ以上のノードで保留している可能性があることを示しています。LOCAL_TRAN_IDとGLOBAL_TRAN_IDのローカル部分が同じなので、このノードはトランザクションのグローバル・コーディネータです。

親トピック: 分散トランザクション情報の表示

35.3.2 インダウト・トランザクションのセッション・ツリーのトレース

DBA_2PC_NEIGHBORSビューには、リモート・クライアントからの受信インダウト・トランザクションと、リモート・サーバーへの送信インダウト・トランザクションが表示されます。

ビュー	用途
DBA_2PC_NEIGHBORS	(リモート・クライアントからの)受信および(リモート・サーバーへの)送信インダウト分散トランザクションすべてがリストされます。また、ローカル・ノードがトランザクション内のコミット・ポイント・サイトであるかどうかも示します。 このビューには、インダウト・トランザクションが移入されるまで何もデータが入っていません。トランザクションが解決された後、ビューはパージされます。

トランザクションがインダウトのときは、セッション・ツリー内のどのノードがどのロールを実行しているのかを判断することが必要な場合があります。このビューを使用すると、次のことが判断できます。

- 特定のトランザクションのすべての受信接続と送信接続。
- ノードが特定のトランザクションのコミット・ポイント・サイトかどうか。
- ノードが特定のトランザクションのグローバル・コーディネータかどうか(ノードのローカル・トランザクションIDとグローバル・トランザクションIDが同じかどうかで判断する)。

関連性の最も高い列を次の表に示します(ビューのすべての列の詳細は、『Oracle Databaseリファレンス』を参照してください)。

表35-2 DBA_2PC_NEIGHBORS

列	説明
LOCAL_TRAN_ID	「integer.integer.integer」という書式のローカル・トランザクション識別子。 ノート: 接続の LOCAL_TRAN_ID と GLOBAL_TRAN_ID.DBA_2PC_PENDING が同じである場合、そのノードはトランザクションのグローバル・コーディネータです。
IN_OUT	受信トランザクションの場合は IN、送信トランザクションの場合は OUT です。
DATABASE	受信トランザクションの場合は、このローカル・ノードから情報を要求したクライアント・データ

列	説明
	ベースの名前です。送信トランザクションの場合は、リモート・サーバーの情報へのアクセスに使用されたデータベース・リンクの名前です。
DBUSER_OWNER	受信トランザクションの場合は、リモート・データベース・リンクによる接続で使用されるローカル・アカウントです。送信トランザクションの場合は、データベース・リンクの所有者です。
INTERFACE	C はコミット・メッセージ、N は準備完了状態を示すメッセージまたは読取り専用コミットの要求のどちらかです。 IN_OUT が OUT の場合、C は、接続のリモート側の子がコミット・ポイント・サイトであり、コミットと終了のどちらを実行するかを知っていることを意味します。N は、ローカル・ノードの準備が完了したことをリモート・ノードに通知中であることを意味します。 IN_OUT が IN の場合、C は、送信接続のリモート側のローカル・ノードまたはデータベースがコミット・ポイント・サイトであることを表します。N は、リモート・ノードの準備が完了したことをローカル・ノードに通知中であることを意味します。

DBA_2PC_PENDINGの関連情報を問い合せるには、次のスクリプトneighbors_scriptを実行します(出力例も含まれています)。

```
COL LOCAL_TRAN_ID FORMAT A13
COL IN_OUT FORMAT A6
COL DATABASE FORMAT A25
COL DBUSER_OWNER FORMAT A15
COL INTERFACE FORMAT A3
SELECT LOCAL_TRAN_ID, IN_OUT, DATABASE, DBUSER_OWNER, INTERFACE
   FROM DBA_2PC_NEIGHBORS
/
SQL> CONNECT SYS@hq.example.com AS SYSDBA
SQL> @neighbors_script
LOCAL_TRAN_ID IN_OUT DATABASE                  DBUSER_OWNER    INT
------------- ------ ------------------------ --------------- ---
1.15.870      out    SALES.EXAMPLE.COM           SYS             C
```

この出力は、ローカル・ノードからリモート・サーバーsalesに、トランザクション1.15.870をコミットするように送信リクエストが送られたことを示しています。salesがトランザクションをコミットし、他にコミットしたノードがない場合は、常に最初にコミットするのがコミット・ポイント・サイトであるため、salesがコミット・ポイント・サイトになります。

親トピック: 分散トランザクション情報の表示

35.4 インダウト・トランザクションの処理方法の決定

2フェーズ・コミットの間に障害が発生すると、トランザクションはインダウトになります。Oracle Databaseソフトウェアを実行しているサーバー・システムのクラッシュ、分散処理に含まれる2つ以上のOracle Database間のネットワーク接続の切断、または未処理のソフトウェア・エラーの発生により、分散トランザクションはインダウトになります。

ローカルのインダウト分散トランザクションは、手動で強制的にコミットまたはロールバックできます。この操作では一貫性の問題が生じる可能性があるため、特定の条件が成り立つとき以外は実行しないでください。

- 2フェーズ・コミットに関する問題の検出
  分散トランザクションで問題が発生した場合は、エラー・メッセージによってアプリケーションに通知されます。
- 手動オーバーライドを実行するかどうかの判断
  インダウト・トランザクションのオーバーライドは、特定の条件下でのみ実行する必要があります。
- トランザクション・データの分析
  トランザクションの完了を強制する場合は、入手可能な情報を分析します。

関連項目:

インダウト・トランザクション

親トピック: 分散トランザクションの管理

35.4.1 2フェーズ・コミットに関する問題の検出

分散トランザクションで問題が発生した場合は、エラー・メッセージによってアプリケーションに通知されます。

分散トランザクションをコミットするユーザー・アプリケーションには、次のいずれかのエラー・メッセージによって問題が通知されます。

```
ORA-02050: transaction ID rolled back,
           some remote dbs may be in-doubt
ORA-02053: transaction ID committed,
           some remote dbs may be in-doubt
ORA-02054: transaction ID in-doubt
```

アプリケーションでこれらのエラーを受け取った場合は、トランザクションに関する情報を保存するようにしてください。この情報は、後で分散トランザクションの手動リカバリが必要になった場合に使用できます。

ネットワークまたはシステムの障害のために、任意のノードでインダウト分散トランザクションが1つ以上発生した場合でも、そのノードの管理者がなんらかの処理を実行する必要はありません。ネットワークやシステムの障害が解決した後、データベースの自動リカバリ機能によって、セッション・ツリーのすべてのノードが同じ結果になるように(つまり、すべてコミットされるかすべてロールバックされる)、すべてのインダウト・トランザクションの処理が透過的に完了します。

ただし、障害が長期にわたる場合には、トランザクションを強制的にコミットまたはロールバックすることで、ロックされたデータをすべて解放できます。アプリケーションは、この操作を想定しておく必要があります。

親トピック: インダウト・トランザクションの処理方法の決定

35.4.2 手動上書きを実行するかどうかの判断

インダウト・トランザクションのオーバーライドは、特定の条件下でのみ実行する必要があります。

特定のインダウト・トランザクションを手動でオーバーライドするのは、次のいずれかの条件が存在する場合のみです。

- インダウト・トランザクションが他のトランザクションに必要なデータをロックしている場合。この状況は、ORA-01591エラー・メッセージによってユーザーのトランザクションが妨げられたときに起こります。
- インダウト・トランザクションは、UNDOセグメントのエクステントが他のトランザクションによって使用されることを防ぎます。インダウト分散トランザクションのローカル・トランザクションIDの最初の部分は、データ・ディクショナリ・ビューDBA_2PC_PENDINGでリストされるUNDOセグメントのIDに対応します。
- 2フェーズ・コミットの各フェーズの完了を妨げている障害を許容される時間範囲内で訂正できない場合。このような例としては、通信ネットワークやデータベースの損傷など、リカバリに長い時間を要する障害があります。

通常、インダウト分散トランザクションをローカルで強制する場合は、別の場所の管理者と相談した上で決断する必要があります。判断を誤るとデータベースの一貫性が損われ、トレースが困難で手動での修正が必要になる恐れがあります。

これらの条件のいずれも当てはまらない場合、データベースの自動リカバリ機能を常に許可し、トランザクションを完了してください。しかし、前述のいずれかの条件に当てはまる場合には、インダウト・トランザクションのローカルでの修正を検討してください。

親トピック: インダウト・トランザクションの処理方法の決定

35.4.3 トランザクション・データの分析

トランザクションの完了を強制する場合は、入手可能な情報を分析します。

- コミットまたはロールバックされたノードの特定
  DBA_2PC_PENDINGビューを使用し、トランザクションをコミットまたはロールバックしたノードを探します。
- トランザクション・コメントの確認
  DBA_2PC_PENDINGのTRAN_COMMENT列に、対象の分散トランザクションに関するなんらかの情報が与えられていないかを確認します。
- トランザクション・アドバイスの確認
  DBA_2PC_PENDINGのADVICE列に、対象の分散トランザクションに関するなんらかの情報が与えられていないかを確認します。

親トピック: インダウト・トランザクションの処理方法の決定

35.4.3.1 コミットまたはロールバックされたノードの特定

DBA_2PC_PENDINGビューを使用し、トランザクションをコミットまたはロールバックしたノードを探します。

トランザクションをすでに解決しているノードが見つかった場合は、そのノードで実行された処理に従うことができます。

親トピック: トランザクション・データの分析

35.4.3.2 トランザクション・コメントの確認

DBA_2PC_PENDINGのTRAN_COMMENT列に、対象の分散トランザクションに関するなんらかの情報が与えられていないかを確認します。

COMMIT文のCOMMENT句にコメントが含まれているか、またはトランザクションに命名している場合は、トランザクションがコミットされると、トランザクション名がTRAN_COMMENTフィールドに設定されます。

たとえば、インダウト分散トランザクションのコメントで、トランザクションの開始元とタイプを示すことができます。

```
COMMIT COMMENT 'Finance/Accts_pay/Trans_type 10B';
```

また、この情報をトランザクション名として提供するために、SET TRANSACTION...NAME文を使用することも可能です。

関連項目:

「トランザクションの命名」

親トピック: トランザクション・データの分析

35.4.3.3 トランザクション・アドバイスの確認

DBA_2PC_PENDINGのADVICE列に、対象の分散トランザクションに関するなんらかの情報が与えられていないかを確認しま

す。

アプリケーションでALTER SESSION文のADVISE句を使用すれば、分散トランザクションの別々の部分を強制的にコミットまたはロールバックする際のアドバイスを規定できます。

準備フェーズ中に各ノードに送られたアドバイスは、現行トランザクション内でそのデータベースに対し最新のDML文が実行されたときに有効なアドバイスです。

たとえば、あるノードのemp表から別のノードのemp表に従業員レコードを移動する分散トランザクションを考えます。次の一連のSQL文を追加することによって、管理者が各ノードで個別にインダウト・トランザクションを強制的に完了した場合でも、トランザクションでレコードを保護できます。

```
ALTER SESSION ADVISE COMMIT;
INSERT INTO emp@hq ... ;    /*advice to commit at HQ */
ALTER SESSION ADVISE ROLLBACK;
DELETE FROM emp@sales ... ; /*advice to roll back at SALES*/
ALTER SESSION ADVISE NOTHING;
```

与えられたアドバイスに従ってインダウト・トランザクションを手動で強制的に完了すると、最悪の場合、各ノードに従業員レコードがコピーされ、消去できなくなる可能性があります。

親トピック: トランザクション・データの分析

35.5 インダウト・トランザクションの手動上書き

COMMIT文またはROLLBACK文に、FORCEオプションと、コミットまたはロールバックするインダウト・トランザクションのローカル・トランザクションIDまたはグローバル・トランザクションIDを示すテキスト文字列を指定します。

ノート:

以降のすべての例で、トランザクションはローカル・ノードでコミットまたはロールバックされます。ローカルのペンディング・トランザクション表では、このトランザクションの行の STATE 列に強制コミットまたは強制終了の値が記録されます。

- インダウト・トランザクションの手動コミット
  トランザクションIDまたはSCNを使用して、手動でインダウト・トランザクションをコミットできます。
- インダウト・トランザクションの手動ロールバック
  トランザクションIDを使用して、インダウト・トランザクションをロールバックできます。

親トピック: 分散トランザクションの管理

35.5.1 インダウト・トランザクションの手動コミット

トランザクションIDまたはSCNを使用して、手動でインダウト・トランザクションをコミットできます。

インダウト・トランザクションのコミットに必要な権限
トランザクションのコミットを試みる前に、適切な権限があるかどうかを確認してください。

トランザクションIDのみを使用したコミット
トランザクションIDを使用してインダウト・トランザクションをコミットできます。

SCNを使用したコミット
SCNを使用してインダウト・トランザクションをコミットできます。

親トピック: インダウト・トランザクションの手動上書き

35.5.1.1 インダウト・トランザクションのコミットに必要な権限

トランザクションのコミットを試みる前に、適正な権限を持っていることを確認してください。

次の要件に注意してください。

トランザクションをコミットするユーザー	必要な権限
管理者	FORCE TRANSACTION
別のユーザー	FORCE ANY TRANSACTION

親トピック: インダウト・トランザクションの手動コミット

35.5.1.2 トランザクションIDのみを使用したコミット

トランザクションIDを使用して、インダウト・トランザクションをコミットできます。

次のSQL文は、インダウト・トランザクションをコミットします。

```
COMMIT FORCE 'transaction_id';
```

変数transaction_idは、DBA_2PC_PENDINGデータ・ディクショナリ・ビューのLOCAL_TRAN_ID列またはGLOBAL_TRAN_ID列のどちらかで指定されているトランザクション識別子です。

たとえば、DBA_2PC_PENDINGを問い合せて、分散トランザクションのLOCAL_TRAN_IDが1:45.13であることを確認するとします。

そして、次のSQL文を発行し、このインダウト・トランザクションのコミットを強制実行します。

```
COMMIT FORCE '1.45.13';
```

親トピック: インダウト・トランザクションの手動コミット

35.5.1.3 システム変更番号(SCN)を使用したコミット

SCNを使用して、インダウト・トランザクションをコミットできます。

必要であれば、トランザクションを強制的にコミットするときに、トランザクションのSCNを指定することもできます。この機能を使用すると、他のノードでトランザクションがコミットされたときに割り当てられたSCNを使用して、インダウト・トランザクションをコミットできます。

これにより、障害が発生した場合でも、分散トランザクションのコミット時間の同期を保つことができます。SCNは、別のノードですでにコミットされた同じトランザクションのSCNが判別できるときのみ指定してください。

たとえば、次のグローバル・トランザクションIDを使用してトランザクションを手動でコミットするとします。

```
SALES.EXAMPLE.COM.55d1c563.1.93.29
```

まず、対象のトランザクションに関係しているリモート・データベースのDBA_2PC_PENDINGビューを問い合せます。次に、そのノードでトランザクションのコミットに使用されたSCNをノートにとります。そして、ローカル・ノードでトランザクションをコミットするときにこのSCNを指定します。たとえば、SCNが829381993の場合は、次の文を発行します。

```
COMMIT FORCE 'SALES.EXAMPLE.COM.55d1c563.1.93.29', 829381993;
```

関連項目:

COMMIT文の使用の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: インダウト・トランザクションの手動コミット

35.5.2 インダウト・トランザクションの手動ロールバック

トランザクションIDを使用して、インダウト・トランザクションをロールバックできます。

インダウト分散トランザクションのロールバックを試みる前に、適正な権限を持っていることを確認してください。次の要件に注意してください。

トランザクションをコミットするユーザー	必要な権限
管理者	FORCE TRANSACTION
別のユーザー	FORCE ANY TRANSACTION

次のSQL文は、インダウト・トランザクションをロールバックします。

```
ROLLBACK FORCE 'transaction_id';
```

変数transaction_idは、DBA_2PC_PENDINGデータ・ディクショナリ・ビューのLOCAL_TRAN_ID列またはGLOBAL_TRAN_ID列のどちらかで指定されているトランザクション識別子です。

たとえば、ローカル・トランザクションIDが2.9.4のインダウト・トランザクションをロールバックするには、次の文を使用します。

```
ROLLBACK FORCE '2.9.4';
```

ノート:

インダウト・トランザクションをセーブポイントまでロールバックすることはできません。

関連項目:

ROLLBACK文の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: インダウト・トランザクションの手動上書き

35.6 データ・ディクショナリからの保留行のパージ

インダウト・トランザクション用のデータ・ディクショナリから保留中の行をパージできます。

- データ・ディクショナリからの保留中の行のパージについて
  DBMS_TRANSACTION.PURGE_MIXEDプロシージャまたはDBMS_TRANSACTION.PURGE_LOST_DB_ENTRYプロシージャを使用して、インダウト・トランザクション用のデータ・ディクショナリから保留中の行をパージできます。

- PURGE_LOST_DB_ENTRYプロシージャの実行
  データ・ディクショナリのインダウト・トランザクション・エントリをクリーン・アップするには、DBMS_TRANSACTION.PURGE_LOST_DB_ENTRYプロシージャを使用します。
- DBMS_TRANSACTIONを使用する時期の判断
  通常、分散トランザクションの状態に基づいて特定のアクションを実行する必要があります。

親トピック: 分散トランザクションの管理

35.6.1 データ・ディクショナリからの保留行のパージについて

DBMS_TRANSACTION.PURGE_MIXEDプロシージャまたはDBMS_TRANSACTION.PURGE_LOST_DB_ENTRYプロシージャを使用して、インダウト・トランザクション用のデータ・ディクショナリから保留中の行をパージできます。

インダウト・トランザクションは、RECO(リカバラ・プロセス)によってリカバリされる前に、DBA_2PC_PENDING.STATE列にCOLLECTING、COMMITTEDまたはPREPAREDとして表示されます。COMMIT FORCEまたはROLLBACK FORCEを使用してインダウト・トランザクションを強制的に完了すると、FORCED COMMITまたはFORCED ROLLBACKの状態になることがあります。

自動リカバリでは通常、これらの状態にあるエントリが削除されます。唯一の例外として、リカバリ時に、トランザクション内の他のサイトと一貫性のない状態にある強制トランザクションが検出されるときがあります。この場合は、エントリが表内に残る可能性があり、DBA_2PC_PENDINGのMIXED列の値がYESになります。これらのエントリは、DBMS_TRANSACTION.PURGE_MIXEDプロシージャを使用してクリーン・アップできます。

リモート・データベースが永続的に失われたために自動リカバリが不可能な場合は、データベースを再作成しても再作成時に新しいデータベースIDが割り当てられるため、リカバリではそのデータベースを識別できません。このような場合、DBMS_TRANSACTIONパッケージのPURGE_LOST_DB_ENTRYプロシージャを使用してエントリをクリーン・アップする必要があります。このエントリによってデータベース・リソースが保持されることはないため、エントリのクリーン・アップを急ぐ必要はありません。

関連項目:

DBMS_TRANSACTIONパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: データ・ディクショナリからの保留行のパージ

35.6.2 PURGE_LOST_DB_ENTRYプロシージャの実行

データ・ディクショナリのインダウト・トランザクション・エントリをクリーンアップするには、DBMS_TRANSACTION.PURGE_LOST_DB_ENTRYプロシージャを使用します。

エントリをデータ・ディクショナリから手動で削除するには、次の構文を使用します(ここで、trans_idはトランザクションの識別子を表します)。

```
DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY('trans_id');
```

たとえば、保留中の分散トランザクション1.44.99をパージするには、SQL*Plusで次の文を入力します。

```
EXECUTE DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY('1.44.99');
```

このプロシージャは、自動リカバリによってトランザクションを解決できないほどの大幅な再構成を行った場合にのみ実行してください。たとえば、次のような場合です。

- リモート・データベースが完全に失われた場合
- ソフトウェアを再構成した結果、2フェーズ・コミットの機能がなくなった場合
- TPMonitorなどの外部トランザクション・コーディネータからの情報が失われた場合

親トピック: データ・ディクショナリからの保留行のページ

35.6.3 DBMS_TRANSACTIONを使用する時期の判断

通常、分散トランザクションの状態に基づいて特定のアクションを実行する必要があります。

次の表は、分散トランザクションに関する各種の状態と、それに対して管理者が実行すべき処理を示したものです。

STATE列	グローバル・トランザクションの状態	ローカル・トランザクションの状態	通常の処理	代替処理
Collecting	ロールバック	ロールバック	なし	PURGE_LOST_DB_ENTRY (自動リカバリでトランザクションを解決できない場合のみ)
Committed	Committed	Committed	なし	PURGE_LOST_DB_ENTRY (自動リカバリでトランザクションを解決できない場合のみ)
準備完了	不明	準備完了	なし	強制コミットまたは強制ロールバック
Forced commit	不明	Committed	なし	PURGE_LOST_DB_ENTRY (自動リカバリでトランザクションを解決できない場合のみ)
Forced rollback	不明	ロールバック	なし	PURGE_LOST_DB_ENTRY (自動リカバリでトランザクションを解決できない場合のみ)
Forced commit	混在	Committed	非一貫性部分を手動で削除してから PURGE_MIXED を使用する	-
Forced rollback	混在	ロールバック	非一貫性部分を手動で削除してから PURGE_MIXED を使用する	-

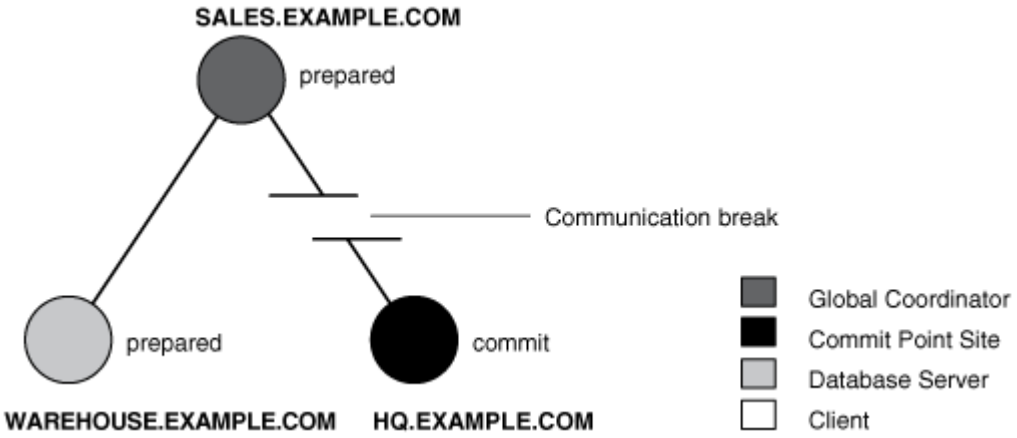
親トピック: データ・ディクショナリからの保留行のページ

35.7 インダウト・トランザクションの手動コミット: 例

例は、手動によるインダウト・トランザクションのコミットを示しています。

図35-1は、分散トランザクションのコミット中の障害を示しています。この障害例では、準備フェーズは完了しています。しかし、コミット・フェーズ時に、コミット・ポイント・サイトがトランザクションをコミットしていても、コミット・ポイント・サイトのコミット確認がグローバル・コーディネータに到達しません。インダウト・トランザクションは他のトランザクションにとっても重要なものであるため、在庫データはロックされており、アクセスできません。また、インダウト・トランザクションがコミットまたはロールバックされるまで、ロックが必ず保持されます。

図35-1 インダウト分散トランザクションの例



手動により、インダウト・トランザクションのローカル部分を強制できます。

- ステップ1: ユーザーからのフィードバックの記録
  例は、インダウト・トランザクションのユーザーからのフィードバックの記録を示しています。
- ステップ2: DBA_2PC_PENDINGの問合せ
  例は、DBA_2PC_PENDINGデータ・ディクショナリ・ビューのインダウト・トランザクションに関する情報の問合せを示しています。
- ステップ3: ローカル・ノードでのDBA_2PC_NEIGHBORSの問合せ
  このステップの目的は、セッション・ツリーを探索してコーディネータを見つけ、最終的にグローバル・コーディネータに到達することです。
- ステップ4: 全ノードでのデータ・ディクショナリ・ビューの問合せ
  この時点で、各設置ノードの管理者に連絡を取り、グローバル・トランザクションIDを使用してステップ2および3を繰り返すように各管理者に依頼できます。
- ステップ5: インダウト・トランザクションのコミット
  グローバルIDを使用してインダウト・トランザクションをコミットします。
- ステップ6: DBA_2PC_PENDINGを使用したMIXED結果のチェック
  トランザクションを手動で強制的にコミットまたはロールバックした後も、ペンディング・トランザクション表の対応する行はそのまま残っています。トランザクションの状態は、トランザクションをどのように強制完了したかに応じて変わります。

親トピック: 分散トランザクションの管理

35.7.1 ステップ1: ユーザーからのフィードバックの記録

例は、インダウト・トランザクションのユーザーからのフィードバックの記録を示しています。

インダウト・トランザクションのロックと競合を起こしているローカル・データベース・システムのユーザーは、次のエラー・メッセージを受け取ります。

```
ORA-01591: lock held by in-doubt distributed transaction 1.21.17
```

この場合、1.21.17はインダウト分散トランザクションのローカル・トランザクションIDです。どのインダウト・トランザクションを強制処理すればよいかを特定するために、問題を報告したユーザーからこのID番号を聞き、記録しておきます。

親トピック: インダウト・トランザクションの手動コミット: 例

35.7.2 ステップ2: DBA_2PC_PENDINGの問合せ

例は、DBA_2PC_PENDINGデータ・ディクショナリ・ビューのインダウト・トランザクションに関する情報の問合せを示しています。

SQL*Plusを使用してwarehouseに接続した後、ローカルのDBA_2PC_PENDINGデータ・ディクショナリ・ビューを問い合せて、インダウト・トランザクションに関する情報を取得します。

```
CONNECT SYS@warehouse.example.com AS SYSDBA
SELECT * FROM DBA_2PC_PENDING WHERE LOCAL_TRAN_ID = '1.21.17';
```

データベースは次の情報を返します。

```
Column Name             Value
----------------------- -------------------------------------
LOCAL_TRAN_ID           1.21.17
GLOBAL_TRAN_ID          SALES.EXAMPLE.COM.55d1c563.1.93.29
STATE                   prepared
MIXED                   no
ADVICE
TRAN_COMMENT            Sales/New Order/Trans_type 10B
FAIL_TIME               31-MAY-91
FORCE_TIME
RETRY_TIME              31-MAY-91
OS_USER                 SWILLIAMS
OS_TERMINAL             TWA139:
HOST                    system1
DB_USER                 SWILLIAMS
COMMIT#
```

- グローバル・トランザクションIDの判断
  グローバル・トランザクションIDは、分散トランザクションのすべてのノードで同一である共通のトランザクションIDです。
- トランザクションの状態の判断
  DBA_2PC_PENDINGデータ・ディクショナリ・ビューのSTATE列は、トランザクションの状態を示します。
- コメントまたはアドバイスの確認
  DBA_2PC_PENDINGデータ・ディクショナリ・ビューのTRANS_COMMENT列は、トランザクションのために含まれているコメントを示し、ADVICE列はアドバイスを提供します。

親トピック: インダウト・トランザクションの手動コミット: 例

35.7.2.1 グローバル・トランザクションIDの判断

グローバル・トランザクションIDは、分散トランザクションのすべてのノードで同一である共通のトランザクションIDです。

次のような書式で記録されています。

```
global_database_name.hhhhhhhh.local_transaction_id
```

ここで:

- global_database_nameは、グローバル・コーディネータのデータベース名です。
- hhhhhhhhは、グローバル・コーディネータの内部データベース識別子(16進値)です。

- local_transaction_idは、グローバル・コーディネータで割り当てられた、対応するローカル・トランザクションIDです。

グローバル・コーディネータでは、グローバル・トランザクションIDの最後の部分とローカル・トランザクションIDが一致します。この例では、これらの番号が一致していないため、warehouseがグローバル・コーディネータではないことがわかります。

```
LOCAL_TRAN_ID          1.21.17
GLOBAL_TRAN_ID         ... 1.93.29
```

親トピック: ステップ2: DBA_2PC_PENDINGの問合せ

35.7.2.2 トランザクションの状態の判断

DBA_2PC_PENDINGデータ・ディクショナリ・ビューのSTATE列は、トランザクションの状態を示します。

このノードのトランザクションは、準備完了状態です。

```
STATE          prepared
```

したがって、warehouseは、自身のコーディネータからコミット要求またはロールバック要求が送られてくるのを待っています。

親トピック: ステップ2: DBA_2PC_PENDINGの問合せ

35.7.2.3 コメントまたはアドバイスの確認

DBA_2PC_PENDINGデータ・ディクショナリ・ビューのTRANS_COMMENT列は、トランザクションのために含まれているコメントを示し、ADVICE列はアドバイスを提供します。

トランザクションのコメントまたはアドバイスに、このトランザクションに関する情報が含まれている可能性があります。トランザクションに関する情報が含まれていれば、そのコメントを利用します。この例では、トランザクションのコメントに開始元とトランザクション・タイプが含まれています。

```
TRAN_COMMENT           Sales/New Order/Trans_type 10B
```

また、SET TRANSACTION...NAME文によって、トランザクション名として提供されている可能性もあります。

この情報から、トランザクションのローカル部分をコミットすべきか、またはロールバックすべきかを決める際に役立つ情報を得ることができます。インダウト・トランザクションに有益なコメントが付けられていない場合は、その他の管理作業を実行し、セッション・ツリーをトレースして、トランザクションをすでに解決しているノードを探す必要があります。

親トピック: ステップ2: DBA_2PC_PENDINGの問合せ

35.7.3 ステップ3: ローカル・ノードでのDBA_2PC_NEIGHBORSの問合せ

このステップの目的は、セッション・ツリーを探索してコーディネータを見つけ、最終的にグローバル・コーディネータに到達することです。

それと同時に、トランザクションをすでに解決しているコーディネータが見つかる場合もあります。見つからない場合は、最終的にコミット・ポイント・サイトを探し出します(コミット・ポイント・サイトでは、常にインダウト・トランザクションが解決されています)。セッション・ツリーをトレースするには、各ノードのDBA_2PC_NEIGHBORSビューを問い合せます。

この例では、warehouseデータベースでこのビューを問い合せます。

```
CONNECT SYS@warehouse.example.com AS SYSDBA
SELECT * FROM DBA_2PC_NEIGHBORS
  WHERE LOCAL_TRAN_ID = '1.21.17'
  ORDER BY SESS#, IN_OUT;
Column Name            Value
```

```
---------------------- ------------------------------------
LOCAL_TRAN_ID          1.21.17
IN_OUT                 in
DATABASE               SALES.EXAMPLE.COM
DBUSER_OWNER           SWILLIAMS
INTERFACE              N
DBID                   000003F4
SESS#                  1
BRANCH                 0100
```

- データベースのロールとデータベース・リンク情報の取得
  DBA_2PC_NEIGHBORSビューは、インダウト・トランザクションに対応付けられた接続に関する情報を提供します。
- コミット・ポイント・サイトの判別
  INTERFACE列は、ローカル・ノードまたは下位ノードがコミット・ポイント・サイトかどうかを示します。

親トピック: インダウト・トランザクションの手動コミット: 例

35.7.3.1 データベースのロールとデータベース・リンク情報の取得

DBA_2PC_NEIGHBORSビューは、インダウト・トランザクションに対応付けられた接続に関する情報を提供します。

情報は、接続が受信(IN_OUT = in)か送信(IN_OUT = out)かによって異なります。

IN_OUT	意味	DATABASE	DBUSER_OWNER
in	このノードは別のノードのサーバーです。	このノードに接続しているクライアント・データベースの名前がリストされます。	インダウト・トランザクションに対応するデータベース・リンク接続のローカル・アカウントがリストされます。
out	このノードは他のサーバーのクライアントです。	リモート・ノードに接続しているデータベース・リンクの名前がリストされます。	インダウト・トランザクションのデータベース・リンクの所有者がリストされます。

この例では、IN_OUT列によって、warehouseデータベースがsalesクライアント(DATABASE列に示されている)のサーバーであることがわかります。

```
IN_OUT                 in
DATABASE               SALES.EXAMPLE.COM
```

warehouseへの接続は、swilliamsアカウント(DBUSER_OWNER列に示されている)からのデータベース・リンクを介して確立されています。

```
DBUSER_OWNER           SWILLIAMS
```

親トピック: ステップ3: ローカル・ノードでのDBA_2PC_NEIGHBORSの問合せ

35.7.3.2 コミット・ポイント・サイトの判別

INTERFACE列は、ローカル・ノードまたは下位ノードがコミット・ポイント・サイトかどうかを示します。

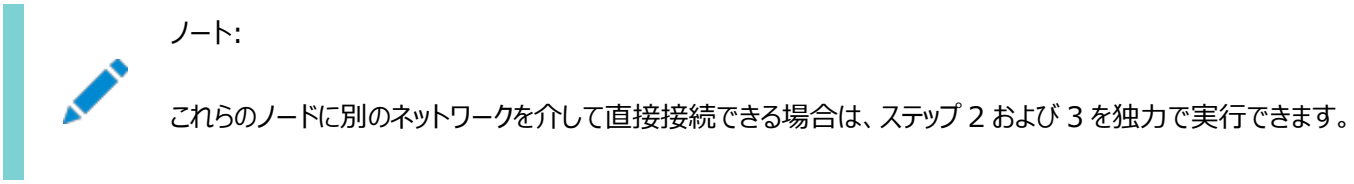
```
INTERFACE              N
```

INTERFACE列が示すように、warehouseもその子もコミット・ポイント・サイトではありません。

親トピック: ステップ3: ローカル・ノードでのDBA_2PC_NEIGHBORSの問合せ

35.7.4 ステップ4: 全ノードでのデータ・ディクショナリ・ビューの問合せ

この時点で、各設置ノードの管理者に連絡を取り、グローバル・トランザクションIDを使用してステップ2および3を繰り返すように各管理者に依頼できます。



ノート:

これらのノードに別のネットワークを介して直接接続できる場合は、ステップ 2 および 3 を独力で実行できます。

たとえば、salesおよびhqでステップ2および3を実行すると、次の結果が返されます。

- salesでのペンディング・トランザクションの状態の確認
  この段階では、salesの管理者がDBA_2PC_PENDINGデータ・ディクショナリ・ビューを問い合せます。
- salesでのペンディング・トランザクションの状態の確認
  次に、salesの管理者はDBA_2PC_NEIGHBORSを問い合せて、グローバル・コーディネータ、ローカル・コーディネータおよびコミット・ポイント・サイトを判別します。
- HQでのペンディング・トランザクションの状態の確認
  この段階では、hqの管理者がDBA_2PC_PENDINGデータ・ディクショナリ・ビューを問い合せます。

親トピック: インダウト・トランザクションの手動コミット: 例

35.7.4.1 salesでのペンディング・トランザクションの状態の確認

この段階では、salesの管理者がDBA_2PC_PENDINGデータ・ディクショナリ・ビューを問い合せます。

```
SQL> CONNECT SYS@sales.example.com AS SYSDBA
SQL> SELECT * FROM DBA_2PC_PENDING
   > WHERE GLOBAL_TRAN_ID = 'SALES.EXAMPLE.COM.55d1c563.1.93.29';
Column Name             Value
----------------------- -------------------------------------
LOCAL_TRAN_ID           1.93.29
GLOBAL_TRAN_ID          SALES.EXAMPLE.COM.55d1c563.1.93.29
STATE                   prepared
MIXED                   no
ADVICE
TRAN_COMMENT            Sales/New Order/Trans_type 10B
FAIL_TIME               31-MAY-91
FORCE_TIME
RETRY_TIME              31-MAY-91
OS_USER                 SWILLIAMS
OS_TERMINAL             TWA139:
HOST                    system1
DB_USER                 SWILLIAMS
COMMIT#
```

親トピック: ステップ4: 全ノードでのデータ・ディクショナリ・ビューの問合せ

35.7.4.2 salesでのコーディネータとコミット・ポイント・サイトの判別

次に、salesの管理者はDBA_2PC_NEIGHBORSを問い合せて、グローバル・コーディネータ、ローカル・コーディネータおよびコミット・ポイント・サイトを判別します。

```
SELECT * FROM DBA_2PC_NEIGHBORS
   WHERE GLOBAL_TRAN_ID = 'SALES.EXAMPLE.COM.55d1c563.1.93.29'
   ORDER BY SESS#, IN_OUT;
```

この問合せは、次の3行を返します。

- warehouseへの接続
- hqへの接続
- ユーザーが確立した接続

warehouse接続の行に対応する情報を、書式を変えて示すと次のようになります。

```
Column Name          Value
-------------------- ------------------------------------
LOCAL_TRAN_ID        1.93.29
IN_OUT               OUT
DATABASE             WAREHOUSE.EXAMPLE.COM
DBUSER_OWNER         SWILLIAMS
INTERFACE            N
DBID                 55d1c563
SESS#                1
BRANCH               1
```

hq接続の行に対応する情報を、書式を変えて示すと次のようになります。

```
Column Name          Value
-------------------- ------------------------------------
LOCAL_TRAN_ID        1.93.29
IN_OUT               OUT
DATABASE             HQ.EXAMPLE.COM
DBUSER_OWNER         ALLEN
INTERFACE            C
DBID                 00000390
SESS#                1
BRANCH               1
```

前の問合せから得られた情報によって、次のことがわかります。

- salesは、ローカル・トランザクションIDとグローバル・トランザクションIDが一致しているため、グローバル・コーディネータです。
- このノードからは送信接続が2つ確立されていますが、受信接続は確立されていません。したがって、salesは別のノードのサーバーではありません。
- コミット・ポイント・サイトは、hqまたはそのサーバーの1つです。

親トピック: ステップ4: 全ノードでのデータ・ディクショナリ・ビューの問合せ

35.7.4.3 HQでのペンディング・トランザクションの状態の確認

この段階では、hqの管理者がDBA_2PC_PENDINGデータ・ディクショナリ・ビューを問い合せます。

```
SELECT * FROM DBA_2PC_PENDING@hq.example.com
   WHERE GLOBAL_TRAN_ID = 'SALES.EXAMPLE.COM.55d1c563.1.93.29';
Column Name          Value
-------------------- ------------------------------------
LOCAL_TRAN_ID        1.45.13
GLOBAL_TRAN_ID       SALES.EXAMPLE.COM.55d1c563.1.93.29
STATE                COMMIT
MIXED                NO
ACTION
TRAN_COMMENT         Sales/New Order/Trans_type 10B
FAIL_TIME            31-MAY-91
FORCE_TIME
RETRY_TIME           31-MAY-91
OS_USER              SWILLIAMS
```

```
OS_TERMINAL             TWA139:
HOST                    SYSTEM1
DB_USER                 SWILLIAMS
COMMIT#                 129314
```

この時点で、トランザクションを解決しているノードが見つかりました。ビューが示しているように、このノードではコミットが完了しており、コミットID番号が割り当てられています。

```
STATE                   COMMIT
COMMIT#                 129314
```

したがって、自分のローカル・データベースでインダウト・トランザクションを強制的にコミットできます。この調査結果を他の管理者に伝えれば、他の場所での解決に役立つ可能性があります。

親トピック: ステップ4: 全ノードでのデータ・ディクショナリ・ビューの問合せ

35.7.5 ステップ5: インダウト・トランザクションのコミット

グローバルIDを使用してインダウト・トランザクションをコミットします。

salesデータベースの管理者と連絡を取り、グローバルIDを使用してインダウト・トランザクションを手動でコミットするように依頼します。

```
SQL> CONNECT SYS@sales.example.com AS SYSDBA
SQL> COMMIT FORCE 'SALES.EXAMPLE.COM.55d1c563.1.93.29';
```

同時に、warehouseデータベースの管理者として、グローバルIDを使用してインダウト・トランザクションを手動でコミットします。

```
SQL> CONNECT SYS@warehouse.example.com AS SYSDBA
SQL> COMMIT FORCE 'SALES.EXAMPLE.COM.55d1c563.1.93.29';
```

親トピック: インダウト・トランザクションの手動コミット: 例

35.7.6 ステップ6: DBA_2PC_PENDINGを使用したMIXED結果のチェック

トランザクションを手動で強制的にコミットまたはロールバックした後も、ペンディング・トランザクション表の対応する行はそのまま残っています。トランザクションの状態は、トランザクションをどのように強制完了したかに応じて変わります。

Oracle Databaseには、必ずペンディング・トランザクション表があります。これは、2フェーズ・コミットの各フェーズの進行に従って分散トランザクションに関する情報を格納する特別な表です。データベースの保留中のトランザクション表は、DBA_2PC_PENDINGデータ・ディクショナリ・ビューを介して問い合せることができます(準備完了トランザクションのID番号と状態の判断を参照)。

また、ペンディング・トランザクション表で特に重要なものとして、DBA_2PC_PENDING.MIXEDに示されるMIXED結果フラグがあります。ペンディング・トランザクションを強制的にコミットまたはロールバックする場合は、選択を誤る可能性があります。たとえば、ローカル管理者がトランザクションをロールバックしたにもかかわらず、他のノードではそのトランザクションをコミットしてしまうといったことが考えられます。このような誤った判断は自動的に検出され、対応するペンディング・トランザクションのレコードに損傷フラグが設定されます(MIXED=yes)。

RECOバックグラウンド・プロセスは、ペンディング・トランザクション表の情報を使用して、インダウト・トランザクションの状態を最終決定します。また、管理者がペンディング・トランザクション表の情報を使用して、保留中の分散トランザクションの自動リカバリ手順を手動で上書きすることもできます。

RECOによって自動解決されたトランザクションはすべて、保留中のトランザクション表から削除される。また、管理者によって正しく解決されたインダウト・トランザクションに関する情報も、RECOが通信を再確立したときにチェックされて、すべてペンディング・

トランザクション表から自動的に削除されます。ただし、管理者が解決したことによってノード間にまたがってMIXEDの結果が生じた行は、DBMS_TRANSACTIONS.PURGE_MIXEDを使用して手動で削除しないかぎり、関係しているすべてのノードのペンディング・トランザクション表にそのまま残ります。

親トピック: インダウト・トランザクションの手動コミット: 例

35.8 ロックによるデータ・アクセスの障害

SQL文を発行すると、データベースは文を正常に実行するために必要なリソースのロックを試みます。しかし、要求されたデータが、コミットされていない他のトランザクションの文によって現在保持されていて、長時間ロックされたままになっていると、タイムアウトが発生します。

- トランザクションのタイムアウト
  リモート・データベースのロックを必要とするDML文は、要求したデータのロックを別のトランザクションが所有している場合にブロックされる可能性があります。
- インダウト・トランザクションによるロック
  ローカル・データベースのロックを必要とする問合せまたはDML文は、インダウト分散トランザクションによるリソースのロックのために無期限にブロックされる可能性があります。

親トピック: 分散トランザクションの管理

35.8.1 トランザクションのタイムアウト

リモート・データベースのロックを必要とするDML文は、要求したデータのロックを別のトランザクションが所有している場合にブロックされる可能性があります。

このようなロックによってSQL文の要求がブロックされ続けると、次の一連のイベントが発生します。

1. タイムアウトが発生します。
2. データベースは文をロールバックします。
3. データベースは次のエラー・メッセージをユーザーに返します。

```
ORA-02049: time-out: distributed transaction waiting for lock
```

トランザクションによってデータは変更されていないので、タイムアウトの結果として必要な処理はありません。アプリケーションでは、デッドロックが発生したものとして処理を継続してください。文を実行したユーザーは、後で同じ文の再実行を試みることができます。それでもロックが続く場合には、ユーザーは管理者に連絡して問題を報告してください。

親トピック: ロックによるデータ・アクセスの障害

35.8.2 インダウト・トランザクションによるロック

ローカル・データベースのロックを必要とする問合せまたはDML文は、インダウト分散トランザクションによるリソースのロックのために無期限にブロックされる可能性があります。

この場合、データベースは次のエラー・メッセージを発行します。

```
ORA-01591: lock held by in-doubt distributed transaction identifier
```

この場合、データベースはSQL文をただちにロールバックします。文を実行したユーザーは、後で同じ文の再実行を試みることができます。ロックが持続する場合、ユーザーは管理者に連絡して、インダウト分散トランザクションのIDなども含めて問題を報告

する必要があります。

2フェーズ・コミットの重要な部分の処理中に障害が発生する確率は低いため、このような状況が起こる可能性はほとんどありません。仮にこのような障害が発生した場合でも、ネットワーク障害やシステム障害から迅速にリカバリできれば、手動で介入しなくても問題は自動的に解決されます。したがって、この種の問題は、ユーザーやデータベース管理者に検出される前に解決されるのが普通です。

親トピック: ロックによるデータ・アクセスの障害

35.9 分散トランザクション障害のシミュレーション

自動的にトランザクションのローカル部分を解決するRECOを観察するため、または手動でインダウト分散トランザクションを解決してその結果を観察するために、分散トランザクションの障害を強制できます。

- 分散トランザクションの強制障害
  COMMIT文のCOMMENTパラメータにコメントを含めることができます。
- RECOの有効化と無効化
  Oracle DatabaseインスタンスのRECOバックグラウンド・プロセスは、分散トランザクションに伴う障害を自動的に解決します。ノードのRECOバックグラウンド・プロセスは、時間間隔を指数関数的に広げながら、インダウト分散トランザクションのローカル部分のリカバリを試みます。

親トピック: 分散トランザクションの管理

35.9.1 分散トランザクションの強制障害

COMMIT文のCOMMENTパラメータには、コメントを挿入できます。

分散トランザクションの2フェーズ・コミットのフェーズ時に意図して障害を発生させるには、COMMENTパラメータに、次のコメントを挿入します。

```
COMMIT COMMENT 'ORA-2PC-CRASH-TEST-n';
```

コメント内のnには、次の整数のいずれかが入ります。

n	影響
1	収集後コミット・ポイントをクラッシュ
2	収集後コミット・ポイント・サイト以外をクラッシュ
3	準備前にクラッシュ(コミット・ポイント・サイト以外)
4	準備後にクラッシュ(コミット・ポイント・サイト以外)
5	コミット前にコミット・ポイント・サイトをクラッシュ
6	コミット後にコミット・ポイント・サイトをクラッシュ

n	影響
7	コミット前にコミット・ポイント・サイト以外をクラッシュ
8	コミット後にコミット・ポイント・サイト以外をクラッシュ
9	情報消去前にコミット・ポイント・サイトをクラッシュ
10	情報消去前にコミット・ポイント・サイト以外をクラッシュ

たとえば、ローカルのコミット・ポイント強度がリモートのコミット・ポイント強度よりも高く、双方のノードが更新されている場合、次の文では、次のメッセージが返されます。

```
COMMIT COMMENT 'ORA-2PC-CRASH-TEST-7';
ORA-02054: transaction 1.93.29 in-doubt
ORA-02059: ORA_CRASH_TEST_7 in commit comment
```

この時点で、インダウト分散トランザクションがDBA_2PC_PENDINGビューに表示されます。対応している場合は、RECOがそのトランザクションを自動的に解決します。

親トピック: 分散トランザクション障害のシミュレーション

35.9.2 RECOの有効化と無効化

Oracle DatabaseインスタンスのRECOバックグラウンド・プロセスは、分散トランザクションに伴う障害を自動的に解決します。ノードのRECOバックグラウンド・プロセスは、時間間隔を指数関数的に広げながら、インダウト分散トランザクションのローカル部分のリカバリを試みます。

RECOは、障害を起こしたトランザクションに関係している他のノードに対して、既存の接続を使用するか、または新しい接続を確立できます。接続が確立されると、RECOはすべてのインダウト・トランザクションを自動的に解決します。各データベースのペンディング・トランザクション表内にインダウト・トランザクションに対応している行があれば、自動的に削除されます。

ENABLE/DISABLE DISTRIBUTED RECOVERYオプションを指定してALTER SYSTEM文を使用すると、RECOを使用可能または使用禁止にできます。たとえば、RECOを一時的に使用禁止にして2フェーズ・コミットの障害を強制的に発生させ、インダウト・トランザクションを手動で解決できます。

次の文は、RECOを使用禁止にします。

```
ALTER SYSTEM DISABLE DISTRIBUTED RECOVERY;
```

一方、次の文はRECOを使用可能にし、インダウト・トランザクションが自動的に解決されるようにします。

```
ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY;
```

親トピック: 分散トランザクション障害のシミュレーション

35.10 読込み一貫性の管理

Oracle Databaseの分散読込み一貫性の実装には重要な制限事項があります。

問題の原因は、各システムには独自のSCNがあり、データベースの内部的なタイムスタンプとして表示できることです。Oracle

Databaseサーバーは、SCNを使用して、どのバージョンのデータが問合せから返されるかを判断します。

分散トランザクションのSCNは、各リモートSQL文の最後、および各トランザクションの最初と最後に同期化されます。2つのノード間の通信量が多く、特に分散更新がある場合には、この同期化が頻繁に実行されます。しかし、分散システムのSCNの絶対的な同期を保つための実用的な手段は存在しません。つまり、あるノードのSCNが別のノードのSCNと比べて幾分古くなるというような状況が常に存在します。

このようなSCNの食い違いから、実行した問合せがわずかに古いスナップショットを使用する可能性があり、その問合せ結果には、リモート・データベースに対する最新の変更が含まれていません。そのため、問合せを実行したときに、読込み一貫性に従ってデータが取得されているにもかかわらず、そのデータが古い場合があります。そのような問合せによって取得したデータはすべて古いSCNに基づいているため、ローカルに実行した更新トランザクションによってリモート・ノードの2つの表が更新された場合、次回のリモート・アクセスで両方の表から選択したデータには、更新前のデータが含まれることになります。

SCNが食い違っているために起こりうる結果の1つとして、SELECT文が2つ連続している場合に、それら2つの文の間でDMLをまったく実行していなくても各文で異なるデータが取得されることがあります。たとえば、UPDATE文を発行して、リモート・データベースで更新をコミットしたとします。このリモート表に基づくビューに対してSELECT文を発行すると、ビューには更新された行が表示されません。次にSELECT文を発行するときは、更新された行が表示されます。

次の手法を使用すると、問合せの直前に2つのシステムのSCNが必ず同期化されます。

- SCNはリモート問合せの最後に同期化されるので、各リモート問合せの前に、同じサイトで`SELECT * FROM DUAL@REMOTE`というようなダミーのリモート問合せを実行します。
- SCNはすべてのリモート・トランザクションの最初に同期化されるので、リモート問合せを発行する前に、現行トランザクションをコミットまたはロールバックします。

親トピック: 分散トランザクションの管理

第VI部 読取り専用マテリアライズド・ビューの管理

読取り専用マテリアライズド・ビューは、マスター表のデータへの読取り専用アクセスを提供します。読取り専用マテリアライズド・ビューを作成および管理できます。

- 読取り専用マテリアライズド・ビューの概念
  読取り専用マテリアライズド・ビューに関連する概念を理解します。
- 読取り専用マテリアライズド・ビューのアーキテクチャ
  マテリアライズド・ビューのレプリケーションに複数のオブジェクトが使用されています。これらのオブジェクトの一部はオプションで、作成したマテリアライズド・ビュー環境のサポートに必要な場合にのみ使用します。たとえば、高速リフレッシュができない複合マテリアライズド・ビューがある場合は、マスター・データベースにマテリアライズド・ビュー・ログがない可能性があります。
- 読取り専用マテリアライズド・ビューの計画
  読取り専用マテリアライズド・ビューの計画を開始する前に、マテリアライズド・ビューに関連する概念およびアーキテクチャを理解するのは重要なことです。読取り専用マテリアライズド・ビューの概念およびアーキテクチャを理解した後、読取り専用マテリアライズド・ビュー環境の計画に関する重要な検討事項があります。
- 読取り専用マテリアライズド・ビューの作成と管理
  読取り専用マテリアライズド・ビューおよびリフレッシュ・グループを作成および管理できます。また、マテリアライズド・ビューをリフレッシュすることもできます。
- 読取り専用マテリアライズド・ビューの問題のトラブルシューティング
  データベース・リンク、マテリアライズド・ビューの作成、およびマテリアライズド・ビューのリフレッシュに関する問題を診断および解決できます。

36 読取り専用マテリアライズド・ビューの概念

読取り専用マテリアライズド・ビューに関連する概念を理解します。

- レプリケーション・データベース
  レプリケーションとは、分散データベース・システムを構成する複数のデータベースのデータベース・オブジェクト(表など)をコピーおよびメンテナンスするプロセスです。レプリケーションをサポートする1つの方法は、読取り専用マテリアライズド・ビューです。
- 読取り専用マテリアライズド・ビュー
  読取り専用マテリアライズド・ビューには、ある一時点におけるターゲット・マスター表の全体または一部のコピーが含まれています。部分的コピーには、行のサブセット、列のサブセット、あるいはその両方が含まれる場合があります。
- マテリアライズド・ビューの使用
  マテリアライズド・ビューは、ネットワーク負荷の軽減、データのサブセット化やモバイル・コンピューティングなどの目標を達成するために使用できます。
- 使用可能なマテリアライズド・ビュー
  使用可能なマテリアライズド・ビューには、主キー・マテリアライズド・ビュー、オブジェクト・マテリアライズド・ビュー、ROWIDマテリアライズド・ビューおよび複合マテリアライズド・ビューがあります。
- マテリアライズド・ビュー関連のユーザーおよび権限
  マテリアライズド・ビューに関連するユーザーには、作成者、リフレッシャおよび所有者があります。マテリアライズド・ビューでの操作を実行するために必要な権限は、操作を実行するユーザーのタイプに応じて異なります。
- マテリアライズド・ビューを使用したデータのサブセット化
  マスター表内のデータのサブセットを反映するマテリアライズド・ビューを構成するために、行のサブセットおよび列のサブセットを使用できます。
- マテリアライズド・ビューのリフレッシュ
  マテリアライズド・ビューとマスター表との一貫性を確保するには、マテリアライズド・ビューを定期的に**リフレッシュ**する必要があります。
- リフレッシュ・グループ
  マテリアライズド・ビュー間のトランザクションの一貫性を保つことが重要なときは、マテリアライズド・ビューを**リフレッシュ・グループ**に編成できます。
- マテリアライズド・ビュー・ログ
  マテリアライズド・ビュー・ログは、マテリアライズド・ビューのマスター表が含まれているデータベースの表です。マスター表へのすべてのDML変更が記録されます。
- マテリアライズド・ビューおよびユーザー定義のデータ型
  ユーザー定義データ型を含むマテリアライズド・ビューには、特別な考慮事項があります。
- マスター・データベースでのマテリアライズド・ビューの登録
  マスター・データベースでは、マスター表に基づいてマテリアライズド・ビューの情報がOracleデータベースにより自動的に登録されます。

親トピック: 読取り専用マテリアライズド・ビューの管理

36.1 レプリケーション・データベース

レプリケーションとは、分散データベース・システムを構成する複数のデータベースのデータベース・オブジェクト(表など)をコピーおよびメンテナンスするプロセスです。レプリケーションをサポートする1つの方法は、読取り専用マテリアライズド・ビューです。

レプリケーション環境では、**マスター・データベース**および**マテリアライズド・ビュー・データベース**の2つの基本的なデータベース・タイプがサポートされます。マテリアライズド・ビュー・データベースには、ある一時点における表データのイメージ、すなわちマテリアライズド・ビューが含まれています。マテリアライズド・ビューが定義されている表は、マテリアライズド・ビューのマスター表です。通常、マテリアライズド・ビューは、マスター表と同期するために定期的にリフレッシュされます。

マテリアライズド・ビューは**リフレッシュ・グループ**に編成できます。リフレッシュ・グループ内のすべてのマテリアライズド・ビューのデータが同じ時点でのトランザクションの一貫性に対応できるように、リフレッシュ・グループ内のマテリアライズド・ビューは、同時にリフレッシュされます。

ノート:

Oracle GoldenGate は、レプリケーションのためのフル機能を備えた Oracle のソリューションです。詳細は、Oracle GoldenGate のドキュメントを参照してください。

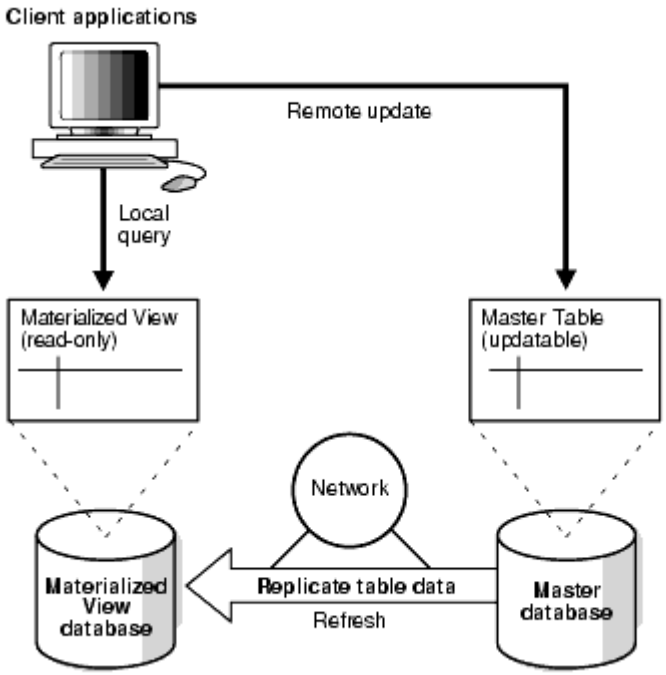
親トピック: 読取り専用マテリアライズド・ビューの概念

36.2 読取り専用マテリアライズド・ビュー

読取り専用マテリアライズド・ビューには、ある一時点におけるターゲット・マスター表の全体または一部のコピーが含まれています。部分的コピーには、行のサブセット、列のサブセット、あるいはその両方が含まれる場合があります。

読取り専用マテリアライズド・ビューは、マスター表のデータへの読取り専用アクセスを提供します。ネットワークの可用性にかかわらず、アプリケーションは読取り専用マテリアライズド・ビューにデータを問い合せて、マスター・データベースへのネットワーク・アクセスを回避できます。ただし、アプリケーションでデータ操作言語(DML)による変更を実行する場合は、全システムのアプリケーションはマスター・データベースのデータにアクセスする必要があります。図36-1は、基本的な読取り専用レプリケーションを示しています。

図36-1 読取り専用マテリアライズド・ビュー



マテリアライズド・ビューには、次の利点があります。

- ローカル・アクセスが可能なため、応答時間を短縮し、可用性を向上できます。

- マテリアライズド・ビューがそのソースとは異なるデータベース内にある場合、ユーザーはローカルのマテリアライズド・ビューへの問合せができるので、マスター・データベースを問合せの負荷から開放します
- ターゲット・マスターのデータ・セットのうち、選択したサブセットのみをレプリケート可能にすることにより、データ・セキュリティを高めることができます。

ユーザーは必要に応じて読取り専用マテリアライズド・ビューを同期(リフレッシュ)できます。ユーザーはマテリアライズド・ビューをリフレッシュするとき、最後のリフレッシュ以降にマスター表で発生した変更を受け取ります。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.3 マテリアライズド・ビューの使用

マテリアライズド・ビューは、ネットワーク負荷の軽減、データのサブセット化やモバイル・コンピューティングなどの目標を達成するために使用できます。

- ネットワーク負荷の軽減
  ネットワーク負荷の軽減が目的の1つである場合は、マテリアライズド・ビューを使用することによって、地域のデータベースに企業データベースを分布させることができます。
- データ・サブセッティングを可能にする
  マテリアライズド・ビューでは、列および行レベルのサブセッティングに基づいてデータをレプリケートできます。
- モバイル・コンピューティングを可能にする
  マテリアライズド・ビューは、専用のネットワーク接続を必要としません。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.3.1 ネットワーク負荷の軽減

ネットワーク負荷の軽減が目的の1つである場合は、マテリアライズド・ビューを使用することによって、地域のデータベースに企業データベースを分布させることができます。

会社全体で1つのデータベース・サーバーにアクセスするのではなく、ユーザーの負荷を複数のデータベース・サーバーに分散します。レプリケートするデータ量を減らすために、マテリアライズド・ビューをマスター表のサブセットにすることができます。

親トピック: マテリアライズド・ビューの使用

36.3.2 データのサブセット化の有効化

マテリアライズド・ビューでは、列および行レベルのサブセッティングに基づいてデータをレプリケートできます。

データのサブセット化により、特定のデータベースに関する情報のみをレプリケートできるようになります。たとえば、各地の営業所ではその地域に必要なデータのみをレプリケートし、不要なネットワークの通信量を減らすことができます。

行および列の両方のサブセットによって、マスター表のデータの一部を含むマテリアライズド・ビューを作成できます。行のサブセットでは、WHERE句を使用して、マスター内の必要な行のみをマテリアライズド・ビューに含めることができます。列のサブセットでは、マスター内の必要な列のみをマテリアライズド・ビューに含めることができます。マテリアライズド・ビューを作成する際に、SELECT文で特定の列を指定してこれらを実行します。

親トピック: マテリアライズド・ビューの使用

36.3.3 モバイル・コンピューティングの有効化

マテリアライズド・ビューは専用のネットワーク接続を必要としません。

ジョブをスケジュールしてリフレッシュ処理を自動化するオプションもありますが、必要に応じて手動でリフレッシュできるマテリアライズド・ビューは、ラップトップ・コンピュータ上で実行される問合せのための理想的なソリューションです。

親トピック: マテリアライズド・ビューの使用

36.4 使用可能なマテリアライズド・ビュー

使用可能なマテリアライズド・ビューには、主キー・マテリアライズド・ビュー、オブジェクト・マテリアライズド・ビュー、ROWIDマテリアライズド・ビューおよび複合マテリアライズド・ビューがあります。

- 使用可能なマテリアライズド・ビューについて
  Oracleでは、多数の異なるレプリケーション(および非レプリケーション)状況のニーズを満たすために、いくつかのタイプの読取り専用マテリアライズド・ビューを提供しています。
- 主キー・マテリアライズド・ビュー
  主キー・マテリアライズド・ビューがマテリアライズド・ビューのデフォルト・タイプです。
- オブジェクト・マテリアライズド・ビュー
  マテリアライズド・ビューがオブジェクト表に基づいてOF type句を使用して作成されている場合、このマテリアライズド・ビューはオブジェクト・マテリアライズド・ビューと呼ばれます。
- ROWIDマテリアライズド・ビュー
  ROWIDマテリアライズド・ビューは、マスター表の行の物理的な行ID (ROWID)に基づいています。
- 複合マテリアライズド・ビュー
  複合マテリアライズド・ビューは、高速リフレッシュできません。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.4.1 使用可能なマテリアライズド・ビューについて

Oracleでは、多数の異なるレプリケーション(および非レプリケーション)状況のニーズを満たすために、いくつかのタイプの読取り専用マテリアライズド・ビューを提供しています。

どのタイプのマテリアライズド・ビューを作成する場合でも、マテリアライズド・ビューの問合せで表所有者のスキーマ名を指定します。たとえば、次のCREATE MATERIALIZED VIEW文について考えます。

```
CREATE MATERIALIZED VIEW hr.employees
  AS SELECT * FROM hr.employees@orc1.example.com;
```

この文では、問合せにスキーマhrが指定されています。

親トピック: 使用可能なマテリアライズド・ビュー

36.4.2 主キー・マテリアライズド・ビュー

主キー・マテリアライズド・ビューがマテリアライズド・ビューのデフォルト・タイプです。

主キー・マテリアライズド・ビューを作成するためのSQL文の例を次に示します。

```
CREATE MATERIALIZED VIEW oe.customers WITH PRIMARY KEY
  AS SELECT * FROM oe.customers@orc1.example.com;
```

主キー・マテリアライズド・ビューがデフォルトであるため、次の文の結果も主キー・マテリアライズド・ビューになります。

```
CREATE MATERIALIZED VIEW oe.customers
  AS SELECT * FROM oe.customers@orc1.example.com;
```

リモート・マテリアライズド・ビュー・データベースで行のサブセットを作成できるように、主キー・マテリアライズド・ビューに副問合せを含めることができます。副問合せは主問合せに埋め込まれた問合せで、CREATE MATERIALIZED VIEW文に複数のSELECT文が使用されます。この副問合せには、基本的なWHERE句のように単純なものと、マルチレベルのWHERE EXISTS句のように複雑なものとがあります。参照される各マスターにマテリアライズド・ビュー・ログがある場合は、選択されたクラスの副問合せを含む主キー・マテリアライズド・ビュー・サイトに高速リフレッシュを実行できます。高速リフレッシュでは、マテリアライズド・ビュー・ログを使用して、最後のリフレッシュ以降に変更が加えられた行のみを更新します。

次のマテリアライズド・ビューは、副問合せを含むWHERE句を使用して作成されています。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
 SELECT * FROM oe.orders@orc1.example.com o
 WHERE EXISTS
   (SELECT * FROM oe.customers@orc1.example.com c
    WHERE o.customer_id = c.customer_id AND c.credit_limit > 10000);
```

このタイプのマテリアライズド・ビューは、副問合せマテリアライズド・ビューと呼ばれます。

ノート:

この oe.orders マテリアライズド・ビューを作成するには、マスター表のマテリアライズド・ビュー・ログに credit_limit がログされている必要があります。詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

関連項目:

- 副問合せを含むマテリアライズド・ビューの詳細は、「副問合せを含むマテリアライズド・ビュー」
- 高速リフレッシュの詳細は、「リフレッシュ・タイプ」
- マテリアライズド・ビュー・ログの詳細は、「マテリアライズド・ビュー・ログ」
- 副問合せの詳細は、『Oracle Database SQL言語リファレンス』

親トピック: 使用可能なマテリアライズド・ビュー

36.4.3 オブジェクト・マテリアライズド・ビュー

マテリアライズド・ビューがオブジェクト表に基づいてOF type句を使用して作成されている場合、このマテリアライズド・ビューはオブジェクト・マテリアライズド・ビューと呼ばれます。

オブジェクト・マテリアライズド・ビューは、オブジェクト表と同じ構造です。つまり、オブジェクト・マテリアライズド・ビューは行オブジェクトから構成され、各行オブジェクトはオブジェクト識別子(OID)列により識別されます。

関連項目:

- 「オブジェクト表に基づいたマテリアライズド・ビュー」

- オブジェクト・マテリアライズド・ビューを作成する例は、「読取り専用マテリアライズド・ビューの作成」

親トピック: 使用可能なマテリアライズド・ビュー

36.4.4 ROWIDマテリアライズド・ビュー

ROWIDマテリアライズド・ビューは、マスター表の行の物理的な行ID(ROWID)に基づいています。

ROWIDマテリアライズド・ビューを使用するのは、主キーを持たないマスター表に基づくマテリアライズド・ビュー、またはマスター表の一部の主キー列を含むマテリアライズド・ビューの場合です。

ROWIDマテリアライズド・ビューを作成するCREATE MATERIALIZED VIEW文の例を次に示します。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH WITH ROWID AS
 SELECT * FROM oe.orders@orc1.example.com;
```

関連項目:

- ROWIDマテリアライズド・ビューと主キー・マテリアライズド・ビューの相違点の詳細は、「マテリアライズド・ビュー・ログ」
- CREATE MATERIALIZED VIEW文のWITH ROWID句の詳細は、『Oracle Database SQL言語リファレンス』

親トピック: 使用可能なマテリアライズド・ビュー

36.4.5 複合マテリアライズド・ビュー

複合マテリアライズド・ビューは、高速リフレッシュできません。

- 複合マテリアライズド・ビューについて
  高速リフレッシュを実行するには、マテリアライズド・ビューの定義問合せで一定の制限事項を守る必要があります。
- 単純マテリアライズド・ビューと複合マテリアライズド・ビューの比較
  特定のアプリケーションでは、複合マテリアライズド・ビューの使用を検討する必要があります。

親トピック: 使用可能なマテリアライズド・ビュー

36.4.5.1 複合マテリアライズド・ビューについて

高速リフレッシュを実行するには、マテリアライズド・ビューの定義問合せで一定の制限事項を守る必要があります。

マテリアライズド・ビューの定義問合せが汎用的なものであり、制限事項を守ることができない場合は、そのマテリアライズド・ビューは複合マテリアライズド・ビューであり、高速リフレッシュを実行できません。

具体的には、マテリアライズド・ビューの定義問合せに次のいずれかが含まれる場合、そのマテリアライズド・ビューは複合マテリアライズド・ビューとみなされます。

- CONNECT BY句

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

  ```
  CREATE MATERIALIZED VIEW hr.emp_hierarchy AS
    SELECT LPAD(' ', 4*(LEVEL-1))||email USERNAME
      FROM hr.employees@orc1.example.com START WITH manager_id IS NULL
      CONNECT BY PRIOR employee_id = manager_id;
  ```

- INTERSECT、MINUSまたはUNION ALL集合演算

  たとえば、次の文にはUNION ALL集合演算が含まれているため、複合マテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW hr.mview_employees AS
  SELECT employees.employee_id, employees.email
  FROM hr.employees@orc1.example.com
UNION ALL
  SELECT new_employees.employee_id, new_employees.email
  FROM hr.new_employees@orc1.example.com;
```

- DISTINCTまたはUNIQUEキーワード

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW hr.employee_depts AS
  SELECT DISTINCT department_id FROM hr.employees@orc1.example.com
  ORDER BY department_id;
```

- 集計関数(場合による)。集計関数を定義問合せに指定して、単純マテリアライズド・ビューを作成することもできます。

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW hr.average_sal AS
  SELECT AVG(salary) "Average" FROM hr.employees@orc1.example.com;
```

- 副問合せ以外で指定される結合(場合による)。結合を定義問合せに指定して、単純マテリアライズド・ビューを作成することもできます。

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW hr.emp_join_dep AS
  SELECT last_name
  FROM hr.employees@orc1.example.com e, hr.departments@orc1.example.com d
  WHERE e.department_id = d.department_id;
```

- UNION演算子(場合による)。

  具体的には、次の条件のいずれかが満たされた場合に、UNION演算子を指定したマテリアライズド・ビューは複合マテリアライズド・ビューになります。

  - UNION内の問合せが複合問合せの場合。ここまでの箇条書きの項目は、問合せによりマテリアライズド・ビューが複合マテリアライズド・ビューになる場合を示します。
  - 最も外側のSELECTリスト列が、UNION内の問合せに一致しない場合。次の例で、最初の問合せの最も外側のSELECTリストにはorder_totalのみが含まれ、2番目の問合せの最も外側のSELECTリストにはcustomer_idが含まれています。したがって、マテリアライズド・ビューは複合マテリアライズド・ビューです。

```
CREATE MATERIALIZED VIEW oe.orders AS
  SELECT order_total
  FROM oe.orders@orc1.example.com o
  WHERE EXISTS
    (SELECT cust_first_name, cust_last_name
       FROM oe.customers@orc1.example.com c
       WHERE o.customer_id = c.customer_id
       AND c.credit_limit > 50)
UNION
  SELECT customer_id
  FROM oe.orders@orc1.example.com o
  WHERE EXISTS
    (SELECT cust_first_name, cust_last_name
       FROM oe.customers@orc1.example.com c
       WHERE o.customer_id = c.customer_id
       AND c.account_mgr_id = 30);
```

  最も内側のSELECTリストは、マテリアライズド・ビューが複合マテリアライズド・ビューであるかどうかには関係し

ません。この例では、UNION内の両方の問合せで、最も内側のSELECTリストはcust_first_nameとcust_last_nameです。

- 「副問合せを使用するマテリアライズド・ビューでの制限事項」で説明されている要件に適合しない句

> ノート:
>
> 複合マテリアライズド・ビューは高速リフレッシュできないので、パフォーマンスが低下する可能性があるため、可能な場合は、使用を避けてください。

関連項目:

- 「リフレッシュ処理」
- 集計関数と結合を含むマテリアライズド・ビューの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』
- CONNECT BY句、集合演算、DISTINCTキーワードおよび集計関数の詳細は、『Oracle Database SQL言語リファレンス』

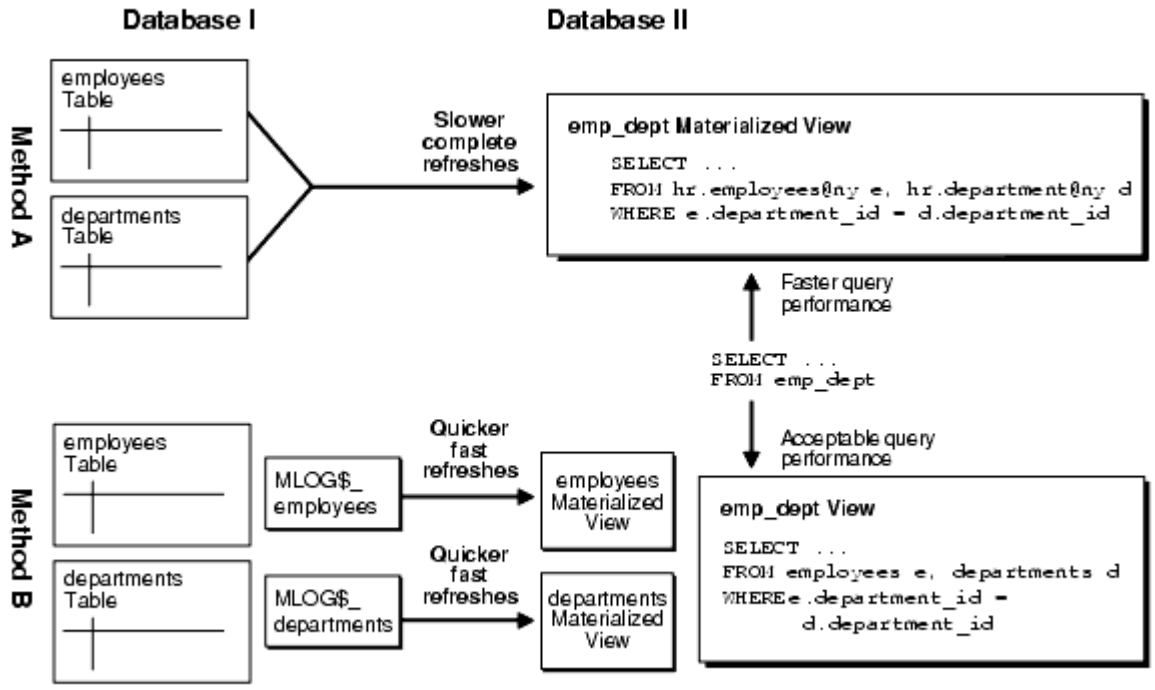
親トピック: 複合マテリアライズド・ビュー

36.4.5.2 単純マテリアライズド・ビューと複合マテリアライズド・ビューの比較

アプリケーションによっては、複合マテリアライズド・ビューの使用を考慮することが必要な場合があります。

図36-2とその後に続く本文では、考慮の必要な場合について説明します。

図36-2 単純マテリアライズド・ビューと複合マテリアライズド・ビューの比較



- 複合マテリアライズド・ビュー: 図36-2の方法Aは、複合マテリアライズド・ビューを示しています。データベースII内のマテリアライズド・ビューでは、マテリアライズド・ビューのリフレッシュ中に結合操作が完了しているので、非常に効率的に問

合せが行われます。ただし、複合マテリアライズド・ビューであるため完全リフレッシュを実行する必要があり、多くの場合、完全リフレッシュは、高速リフレッシュに比べ実行速度が遅くなります。

- 結合ビューを含む単純マテリアライズド・ビュー: 図36-2の方法Bは、データベースII内の2つの単純マテリアライズド・ビューと、マテリアライズド・ビューのデータベース内で結合を実行するビューを示しています。ビューに対する問合せのパフォーマンスは、方法Aの複合マテリアライズド・ビューに対する問合せのパフォーマンスに比べよくありません。ただし、単純マテリアライズド・ビューの方が、高速リフレッシュとマテリアライズド・ビュー・ログを使用してより高速にリフレッシュできます。

要約すると、使用する方法は次のようにして判断できます。

- リフレッシュを実行する頻度が少なく、より高速な問合せのパフォーマンスを優先する場合は、方法A(複合マテリアライズド・ビュー)を使用します。
- 頻繁にリフレッシュを実行し、問合せのパフォーマンスを優先する必要がない場合は、方法B(単純マテリアライズド・ビュー)を使用します。

親トピック: 複合マテリアライズド・ビュー

36.5 マテリアライズド・ビュー関連のユーザーおよび権限

マテリアライズド・ビューに関連するユーザーには、作成者、リフレッシャおよび所有者があります。マテリアライズド・ビューでの操作を実行するために必要な権限は、操作を実行するユーザーのタイプに応じて異なります。

- マテリアライズド・ビューの操作に必要な権限
  3つの異なるタイプのユーザーがマテリアライズド・ビューの操作を実行します。
- 作成者が所有者である場合
  マテリアライズド・ビューの作成者がそのマテリアライズド・ビューを所有している場合、このユーザーはマテリアライズド・ビューを作成するために必要な権限を持つ必要があります。
- 作成者が所有者でない場合
  マテリアライズド・ビューの作成者が所有者でない場合は、マテリアライズド・ビューを作成するために作成者と所有者に特定の権限を付与する必要があります。
- リフレッシャが所有者である場合
  マテリアライズド・ビューのリフレッシャがそのマテリアライズド・ビューを所有している場合、このユーザーはマテリアライズド・ビューを作成するために必要な権限を持つ必要があります。
- リフレッシャが所有者でない場合
  マテリアライズド・ビューのリフレッシャが所有者でない場合は、リフレッシャと所有者に特定の権限を付与する必要があります。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.5.1 マテリアライズド・ビューの操作に必要な権限

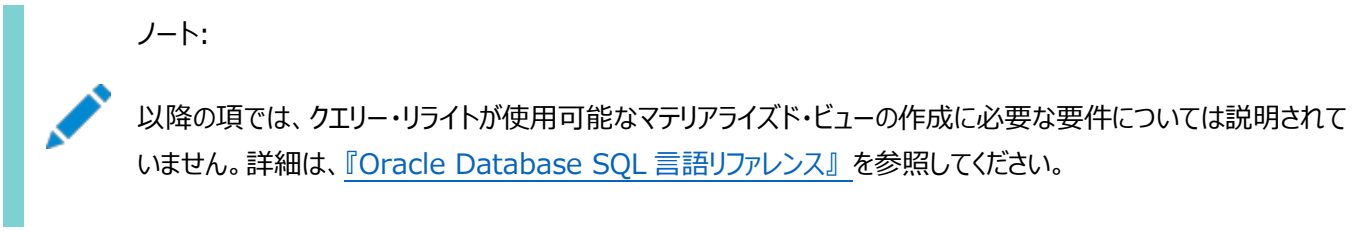
3つの異なるタイプのユーザーがマテリアライズド・ビューの操作を実行します。

これらのユーザーは、次のとおりです。

- 作成者: マテリアライズド・ビューを作成するユーザー。
- リフレッシャ: マテリアライズド・ビューをリフレッシュするユーザー。
- 所有者: マテリアライズド・ビューを所有するユーザー。マテリアライズド・ビューはこのユーザーのスキーマに常駐します。

1人のユーザーが特定のマテリアライズド・ビューに対してこのすべての操作を実行できます。ただし、レプリケーション環境によっては、特定のマテリアライズド・ビューに対して別々のユーザーがこの操作を実行します。これらの操作の実行に必要な権限は、1人のユーザーが操作を実行するか別々のユーザーが実行するかにより異なります。

マテリアライズド・ビュー・データベースのマテリアライズド・ビューの所有者にマスター・データベースへのプライベート・データベース・リンクがある場合、データベース・リンクはマスター・データベースのマスター表の所有者に接続します。それ以外の場合は、データベース・リンク経由の接続に関する通常の規則が適用されます。



ノート:

以降の項では、クエリー・リライトが使用可能なマテリアライズド・ビューの作成に必要な要件については説明されていません。詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

関連項目:

以降の項ではデータベース・リンクについて説明しています。データベース・リンクの使用の詳細は、「分散データベースの概念」を参照してください。

親トピック: マテリアライズド・ビュー関連のユーザーおよび権限

36.5.2 作成者が所有者である場合

マテリアライズド・ビューの作成者がそのマテリアライズド・ビューを所有している場合、このユーザーはマテリアライズド・ビューを作成するために必要な権限を持つ必要があります。

次の権限は、ロールを介してではなく、明示的に付与する必要があります。

- CREATE MATERIALIZED VIEWまたはCREATE ANY MATERIALIZED VIEW
- CREATE TABLEまたはCREATE ANY TABLE
- マスター表およびマスター表のマテリアライズド・ビュー・ログに対するREADまたはSELECTオブジェクト権限、またはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限

  マスター・データベースがリモートの場合は、マテリアライズド・ビュー・データベースのユーザーがデータベース・リンクを介して接続するマスター・データベースのユーザーに、READまたはSELECTオブジェクト権限を付与する必要があります。

親トピック: マテリアライズド・ビュー関連のユーザーおよび権限

36.5.3 作成者が所有者でない場合

マテリアライズド・ビューの作成者が所有者でない場合は、マテリアライズド・ビューを作成するために作成者と所有者に特定の権限を付与する必要があります。

作成者の権限と所有者の権限は、いずれもロールを介してではなく、明示的に付与する必要があります。

表36-1は、マテリアライズド・ビューの作成者が所有者でない場合に必要な権限を示します。

表36-1 マテリアライズド・ビューの作成に必要な権限(作成者が所有者でない場合)

作成者	所有者

作成者	所有者
CREATE ANY MATERIALIZED VIEW	CREATE TABLE または CREATE ANY TABLE マスター表およびマスター表のマテリアライズド・ビュー・ログに対する READ または SELECT オブジェクト権限、または READ ANY TABLE または SELECT ANY TABLE システム権限 マスター・データベースがリモートの場合は、マテリアライズド・ビュー・データベースのユーザーがデータベース・リンクを介して接続するマスター・データベースのユーザーに、READ または SELECT オブジェクト権限を付与する必要があります。

親トピック: マテリアライズド・ビュー関連のユーザーおよび権限

36.5.4 リフレッシャが所有者である場合

マテリアライズド・ビューのリフレッシャがマテリアライズド・ビューを所有している場合、このユーザーはマテリアライズド・ビューを作成するために必要な権限を持つ必要があります。

具体的には、このユーザーは、マスター表およびマスター表のマテリアライズド・ビュー・ログに対するREADまたはSELECTオブジェクト権限、またはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限を持つ必要があります。マスター・データベースがリモートの場合は、マテリアライズド・ビュー・データベースのユーザーがデータベース・リンクを介して接続するマスター・データベースのユーザーに、READまたはSELECTオブジェクト権限を付与する必要があります。この権限は、明示的にまたはロールを介して付与できます。

親トピック: マテリアライズド・ビュー関連のユーザーおよび権限

36.5.5 リフレッシャが所有者でない場合

マテリアライズド・ビューのリフレッシャが所有者でない場合は、リフレッシャと所有者に権限を付与する必要があります。

これらの権限は、明示的にまたはロールを介して付与できます。

表36-2は、マテリアライズド・ビューのリフレッシャが所有者でない場合に必要な権限を示します。

表36-2 マテリアライズド・ビューのリフレッシュに必要な権限(リフレッシャが所有者でない場合)

リフレッシャ	所有者
ALTER ANY MATERIALIZED VIEW	マスター・データベースがローカルの場合、マスター表およびマスター表のマテリアライズド・ビュー・ログに対する READ または SELECT オブジェクト権限、または READ ANY TABLE または SELECT ANY TABLE システム権限を付与する必要があります。 マスター・データベースがリモートの場合は、マテリアライズド・ビュー・データベースのユーザーがデータベース・リンクを介して接続するマスター・データベースのユーザーに、READ または SELECT オブジェクト権限を付与する必要があり

リフレッシャ	所有者
	ます。

親トピック: マテリアライズド・ビュー関連のユーザーおよび権限

36.6 マテリアライズド・ビューを使用したデータのサブセット化

マスター表内のデータのサブセットを反映するマテリアライズド・ビューを構成するために、行のサブセットおよび列のサブセットを使用できます。

- マテリアライズド・ビューを使用したデータのサブセット化について
  特定の状況では、マスター表のデータのサブセットをマテリアライズド・ビューに反映することができます。
- 副問合せを使用するマテリアライズド・ビュー
  複数の表の情報に基づいてデータをレプリケートする場合、それらのマテリアライズド・ビューのメンテナンスおよび定義は困難である可能性があります。
- 副問合せを使用するマテリアライズド・ビューでの制限事項
  副問合せを使用するマテリアライズド・ビューの定義問合せには、マテリアライズド・ビューの高速リフレッシュ機能を保つためのいくつかの制限事項があります。
- 副問合せを含むUNIONを使用するマテリアライズド・ビューでの制限事項
  副問合せを含むUNIONを使用する高速リフレッシュのマテリアライズド・ビューには、制限事項があります。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.6.1 マテリアライズド・ビューを使用したデータのサブセット化について

特定の状況では、マスター表のデータのサブセットをマテリアライズド・ビューに反映することができます。

行のサブセット化では、WHERE句を使用して、マスター表の必要な行のみをマテリアライズド・ビューに含めることができます。列のサブセット化では、マスター表の必要な列のみをマテリアライズド・ビューに含めることができます。これを行うには、マテリアライズド・ビューの作成中にSELECT文でいくつかのSELECT列を指定します。

データのサブセット化の使用目的には、次のようなものがあります。

- ネットワークの通信量の軽減: 列のサブセットを含むマテリアライズド・ビューでは、マテリアライズド・ビューの定義問合せのWHERE句を満たす変更のみがマテリアライズド・ビュー・データベースに適用されるため、転送データ量が減少し、ネットワーク通信量が軽減されます。
- 機密データの保護: マテリアライズド・ビューの定義問合せを満たすデータ以外は表示できません。
- リソース要件の削減: マテリアライズド・ビューがラップトップに格納される場合は、一般にハード・ディスクは会社のサーバーよりもはるかに小さくなります。サブセット化したマテリアライズド・ビューにより、使用する記憶領域をかなり小さくすることができます。
- リフレッシュ時間の改善: マテリアライズド・ビューに適用されるデータが少量であるため、リフレッシュ処理が高速になり、これはラップトップからネットワーク接続を使用してマテリアライズド・ビューをリフレッシュする必要のあるユーザーにとって重要なことです。

たとえば、次の文では、`oe.orders@orc1.example.com`マスター表に基づいたマテリアライズド・ビューが作成され、`sales_rep_id`番号が173の営業担当員の行のみが含まれます。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
 SELECT * FROM oe.orders@orc1.example.com
 WHERE sales_rep_id = 173;
```

受注表でsales_rep_id番号が173以外の行は、このマテリアライズド・ビューからは除外されます。

親トピック: マテリアライズド・ビューを使用したデータのサブセット化

36.6.2 副問合せを使用するマテリアライズド・ビュー

複数の表の情報に基づいてデータをレプリケートする場合、それらのマテリアライズド・ビューのメンテナンスおよび定義は困難である可能性があります。

- 多対1副問合せ
  多対1の関係を持つ副問合せを含むマテリアライズド・ビューを作成できます。
- 1対多副問合せ
  1対多の関係を持つ副問合せを含むマテリアライズド・ビューを作成できます。
- 多対多副問合せ
  多対多の関係を持つ副問合せを含むマテリアライズド・ビューを作成できます。
- 副問合せおよびUNIONを使用するマテリアライズド・ビュー
  副問合せおよびUNIONを含むマテリアライズド・ビューを作成できます。

親トピック: マテリアライズド・ビューを使用したデータのサブセット化

36.6.2.1 多対1副問合せ

多対1の関係を持つ副問合せを含むマテリアライズド・ビューを作成できます。

oeスキーマにcustomers表とorders表が含まれている場合に、orders表とcustomers表の両方のデータに基づいたorders表のマテリアライズド・ビューを作成するとします。たとえば、営業担当員が、信用限度額が$10,000を超える顧客の注文をすべて調べる必要があるとします。この場合、ordersマテリアライズド・ビューを作成するCREATE MATERIALIZED VIEW文には、多対1関係の副問合せが含まれますが、これは、それぞれの顧客が多数の注文をしている可能性があるためです。

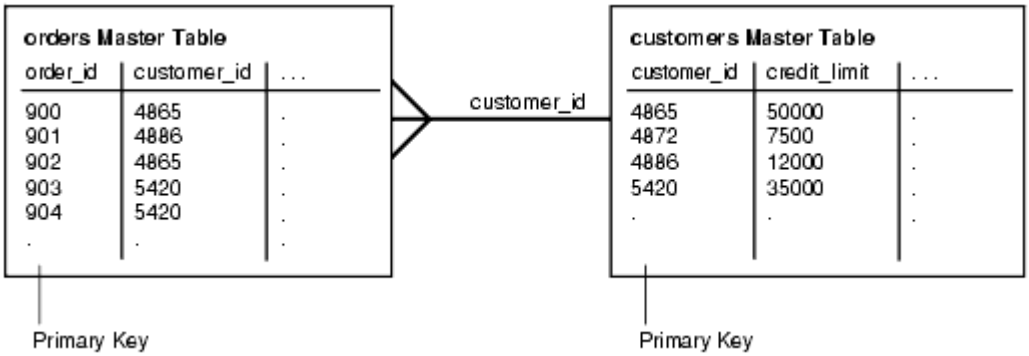
図36-3では、customers表とorders表はcustomer_id列を介して関係していることが示されています。営業担当員の目的は、次の文で達成されます。つまり、次の文では、信用限度額が$10,000より大きい顧客の注文が含まれたマテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
  SELECT * FROM oe.orders@orc1.example.com o
  WHERE EXISTS
    (SELECT * FROM oe.customers@orc1.example.com c
     WHERE o.customer_id = c.customer_id AND c.credit_limit > 10000);
```

ノート:

この oe.orders マテリアライズド・ビューを作成するには、マスター表のマテリアライズド・ビュー・ログに credit_limit がログされている必要があります。詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

図36-3 多対1副問合せを使用した行のサブセット化



この文により作成されるマテリアライズド・ビューは、高速リフレッシュが可能です。信用限度額が10,000ドルを超える新規顧客が発生した場合は、後続のリフレッシュ処理中に新規データがこのマテリアライズド・ビュー・データベースに伝播されます。同様に、顧客の信用限度額が$10,000以下になった場合は、後続のリフレッシュ処理中に、その顧客のデータがこのマテリアライズド・ビューから削除されます。

親トピック: 副問合せを使用するマテリアライズド・ビュー

36.6.2.2 1対多副問合せ

1対多の関係を持つ副問合せを含むマテリアライズド・ビューを作成できます。

oeスキーマにcustomers表とorders表が含まれている場合に、customers表とorders表の両方のデータに基づいたcustomers表のマテリアライズド・ビューを作成するとします。たとえば、営業担当員が、注文合計額が20,000ドルを超える注文を持つすべての顧客を調べる必要があるとします。この場合は、マテリアライズド・ビューの定義問合せに1対多の副問合せを含むマテリアライズド・ビューを作成するのが最も効率的な方法です。

customers表に対するCREATE MATERIALIZED VIEW文の定義問合せには、1対多関係を持つ副問合せが含まれます。つまり、それぞれの顧客が多数の注文をしている可能性があります。

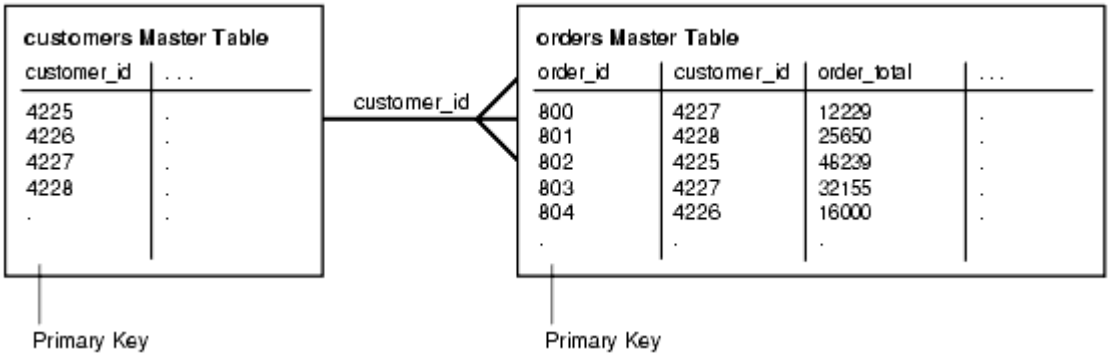
図36-4では、orders表とcustomers表はcustomer_id列を介して関係していることが示されています。営業担当員の目的は、次の文で達成されます。つまり、次の文では、受注合計額が$20,000より大きい全顧客が含まれたマテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW oe.customers REFRESH FAST AS
  SELECT * FROM oe.customers@orc1.example.com c
  WHERE EXISTS
    (SELECT * FROM oe.orders@orc1.example.com o
     WHERE c.customer_id = o.customer_id AND o.order_total > 20000);
```

ノート:

このoe.customersマテリアライズド・ビューを作成するには、ordersマスター表のマテリアライズド・ビュー・ログにcustomer_idとorder_totalをロギングする必要があります。詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

図36-4 1対多副問合せを使用した行のサブセット化



この文により作成されるマテリアライズド・ビューは、高速リフレッシュが可能です。受注合計額が20,000ドルを超える新規顧客が発生した場合は、後続のリフレッシュ処理中に新規データがこのマテリアライズド・ビュー・データベースに伝播されます。同様に、$20,000を超える受注合計額に含まれている注文を顧客が取り消して、受注合計額が$20,000より低くなった場合は、後続のリフレッシュ処理中に、その顧客のデータがこのマテリアライズド・ビューから削除されます。

親トピック: 副問合せを使用するマテリアライズド・ビュー

36.6.2.3 多対多副問合せ

多対多の関係を持つ副問合せを含むマテリアライズド・ビューを作成できます。

oeスキーマにorder_items表とinventories表が含まれている場合に、inventories表とorder_items表の両方のデータに基づいたinventories表のマテリアライズド・ビューを作成するとします。たとえば、営業担当員が、現在のインベントリ数が1以上の各製品で、製品のproduct_idがorder_items表に含まれている全製品のインベントリを調べる必要があるとします。つまり、営業担当員は、顧客が注文した全製品について、インベントリが1以上の製品を調べる必要があります。この場合、インベントリとは、特定の倉庫にある製品の数です。このため、特定の製品は多数の受注項目と多数のインベントリに含まれる可能性があります。

この営業担当員の目的を達成するには、order_items表とinventories表との間の多対多関係に対する副問合せを使用したマテリアライズド・ビューを作成します。

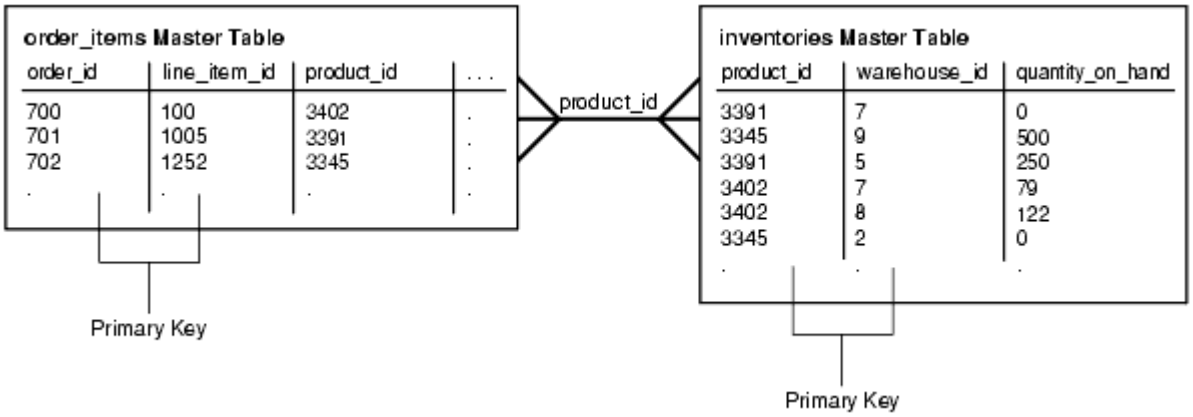
inventoriesマテリアライズド・ビューを作成する場合は、order_items表に含まれている製品に関して、現在の数量が1以上のインベントリを取り出します。図36-5では、inventories表とorder_items表はproduct_id列を介して関係していることが示されています。次の文によりこのマテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW oe.inventories REFRESH FAST AS
  SELECT * FROM oe.inventories@orc1.example.com i
  WHERE i.quantity_on_hand > 0 AND EXISTS
    (SELECT * FROM oe.order_items@orc1.example.com o
     WHERE i.product_id = o.product_id);
```

ノート:

この oe.inventories マテリアライズド・ビューを作成するには、マスター表のマテリアライズド・ビュー・ログに order_items マスター表の product_id 列がログされている必要があります。詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

図36-5 多対多副問合せを使用した行のサブセット化



この文により作成されるマテリアライズド・ビューは、高速リフレッシュが可能です。製品に対して1以上のインベントリが`order_items`表で見つかった場合は、後続のリフレッシュ処理中に新規データがこのマテリアライズド・ビュー・データベースに伝播されます。同様に、顧客が製品の注文を取り消して、その製品に対する他の注文が`order_items`表にない場合は、後続のリフレッシュ処理中に、その製品のインベントリがこのマテリアライズド・ビューから削除されます。

親トピック: 副問合せを使用するマテリアライズド・ビュー

36.6.2.4 副問合せおよびUNIONを使用するマテリアライズド・ビュー

副問合せおよびUNIONを含むマテリアライズド・ビューを作成できます。

1つのマテリアライズド・ビューに、複数の問合せの結果に完全に一致するデータを含める場合は、UNION演算子を使用できます。UNION演算子を使用してマテリアライズド・ビューを作成する場合は、UNION演算子の前後に1つずつSELECT文を使用します。結果のマテリアライズド・ビューには、いずれかの問合せにより選択された行が含まれます。

UNION演算子は、副問合せのWHERE句にOR式を使用せずに「OR」条件を満たす高速リフレッシュ可能なマテリアライズド・ビューを作成する方法として使用できます。副問合せのWHERE句にOR式を使用すると、場合によっては結果のマテリアライズド・ビューが複合マテリアライズド・ビューになり、高速リフレッシュができなくなることがあります。

関連項目:

副問合せでのOR式の詳細は、「副問合せを使用するマテリアライズド・ビューでの制限事項」

たとえば、営業担当員が、特定の`category_id`の製品で、カリフォルニアの倉庫にある製品、または翻訳された製品説明(フランス語翻訳用)に「Rouge」という単語が含まれている製品の製品情報を求めているとします。次の文では、UNION演算子と副問合せを使用して、`category_id` 29の製品について、マテリアライズド・ビューにこのデータを取り出します。

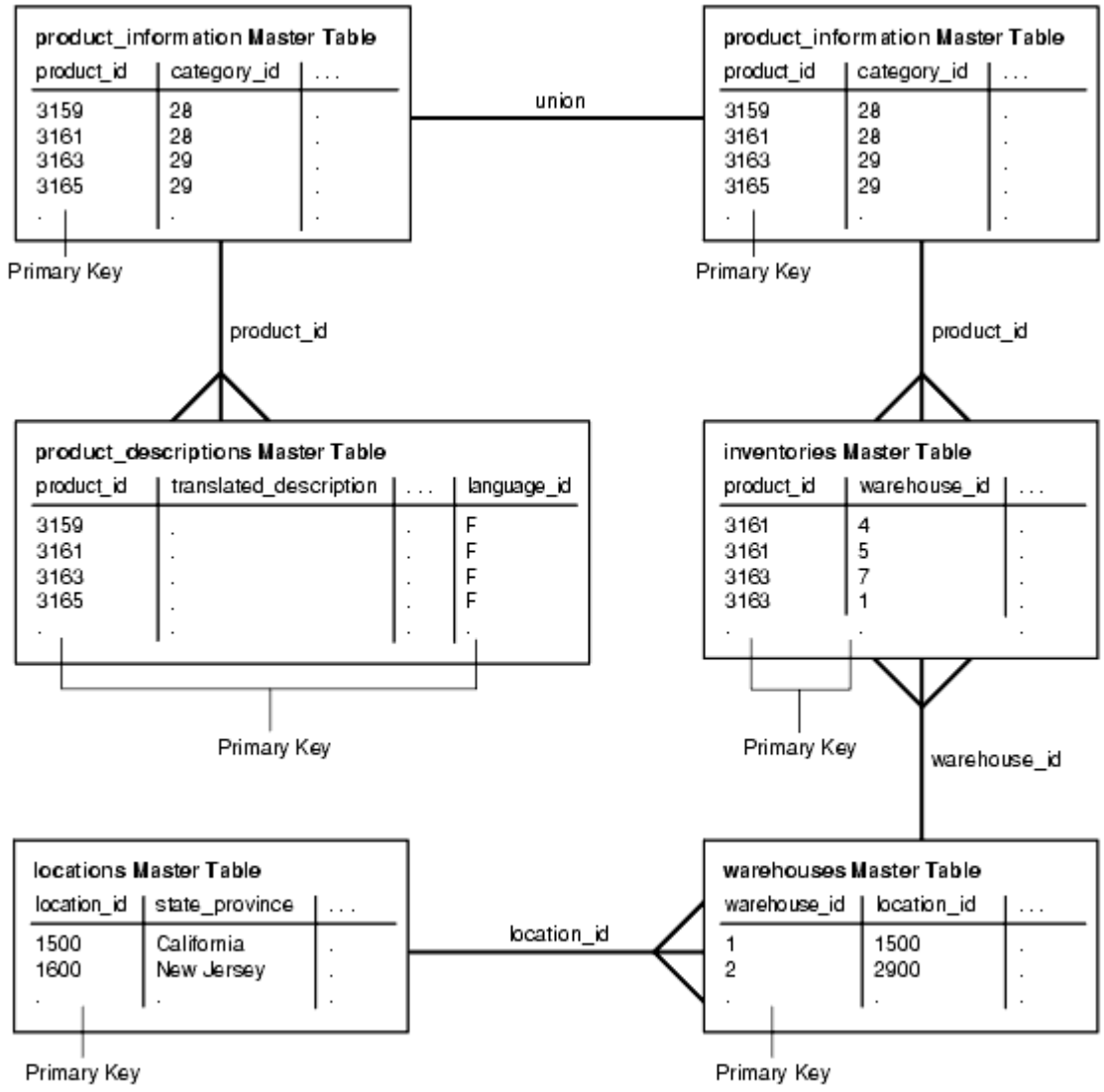
```
CREATE MATERIALIZED VIEW oe.product_information REFRESH FAST AS
 SELECT * FROM oe.product_information@orc1.example.com pi
 WHERE pi.category_id = 29 AND EXISTS
  (SELECT * FROM oe.product_descriptions@orc1.example.com pd
  WHERE pi.product_id = pd.product_id AND
        pd.translated_description LIKE '%Rouge%')
UNION
 SELECT * FROM oe.product_information@orc1.example.com pi
 WHERE pi.category_id = 29 AND EXISTS
  (SELECT * FROM oe.inventories@orc1.example.com i
  WHERE pi.product_id = i.product_id AND EXISTS
    (SELECT * FROM oe.warehouses@orc1.example.com w
    WHERE i.warehouse_id = w.warehouse_id AND EXISTS
      (SELECT * FROM hr.locations@orc1.example.com l
       WHERE w.location_id = l.location_id
       AND l.state_province = 'California')));
```

ノート:

oe.product_information マテリアライズド・ビューを作成するには、各マスターのマテリアライズド・ビュー・ログに、oe.product_descriptions マスター表の translated_description、hr.locations マスター表の state_province および oe.warehouses マスター表の location_id 列をロギングする必要があります。詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

図36-6に、この文に含まれているマスター表の関係を示します。

図36-6 副問合せおよびUNIONを使用した行のサブセット化



この文には、UNION演算子の他に次の副問合せが含まれています。

- product_information表とproduct_descriptions表を参照する副問合せ。1つの製品には複数の製品説明(異なる言語用)があるため、これは1対多副問合せです。
- product_information表とinventories表を参照する副問合せ。1つの製品は多数のインベントリに含まれることがあるため、これは1対多副問合せです。
- inventories表とwarehouses表を参照する副問合せ。多数のインベントリを1つの倉庫に格納できるため、これは多対1副問合せです。

- warehouses表とlocations表を参照する副問合せ。多数の倉庫を1つの場所に配置できるため、これは多対1副問合せです。

この文により作成されるマテリアライズド・ビューは、高速リフレッシュが可能です。カリフォルニアの倉庫に格納されているか、翻訳された製品説明に文字列「Rouge」を含む新製品が追加されると、後続のリフレッシュ処理中に新規データがproduct_informationマテリアライズド・ビューに伝播されます。

親トピック: 副問合せを使用するマテリアライズド・ビュー

36.6.3 副問合せを使用するマテリアライズド・ビューでの制限事項

副問合せを使用するマテリアライズド・ビューの定義問合せには、マテリアライズド・ビューの高速リフレッシュ機能を保つためのいくつかの制限事項があります。

副問合せを使用する高速リフレッシュ・マテリアライズド・ビューの制限事項は、次のとおりです。

- マテリアライズド・ビューは、主キー・マテリアライズド・ビューである必要があります。
- マスター表のマテリアライズド・ビュー・ログに、副問合せで参照されている特定の列が含まれている必要があります。含める列の詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。
- 副問合せが多対多または1対多の場合は、主キーの一部ではない結合列をマスター表のマテリアライズド・ビュー・ログに含める必要があります。この制限は、多対1の副問合せには適用されません。
- 副問合せは、肯定副問合せであることが必要です。たとえば、EXISTS条件は使用できますが、NOT EXISTS条件は使用できません。
- 副問合せは、EXISTSを使用して各ネスト・レベルを接続する必要があります(INは使用できません)。
- 各表は1つのEXISTS式にのみ含められます。
- 結合式では、完全一致または等価比較(等価結合)を使用する必要があります。
- 各表は副問合せ内で1回のみ結合できます。
- 各ネスト・レベルの各表に、主キーが存在している必要があります。
- 各ネスト・レベルでは、上位のレベルの表のみを参照できます。
- 副問合せにはAND条件を含められますが、各OR条件は1行に含まれている列のみを参照できます。1つの副問合せ内の複数のOR条件は、AND条件を使用して接続できます。
- 副問合せ内で参照されるすべての表は、同じマスター・データベース内に存在する必要があります。

ノート:

CREATE MATERIALIZED VIEW 文に ON PREBUILT TABLE 句および副問合せが含まれている場合、この副問合せは多対多として扱われます。したがって、この場合は、結合列をマテリアライズド・ビュー・ログに記録する必要があります。CREATE MATERIALIZED VIEW 文の ON PREBUILT TABLE 句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

関連項目:

- 主キー・マテリアライズド・ビューの詳細は、「主キー・マテリアライズド・ビュー」
- 「マテリアライズド・ビューの高速リフレッシュ機能に関する判断」

親トピック: マテリアライズド・ビューを使用したデータのサブセット化

36.6.4 副問合せを含むUNIONを使用するマテリアライズド・ビューでの制限事項

副問合せを含むUNIONを使用する高速リフレッシュのマテリアライズド・ビューには、制限事項があります。

副問合せを含むUNIONを使用する高速リフレッシュ・マテリアライズド・ビューには、次の制限事項があります。

- 「副問合せを使用するマテリアライズド・ビューでの制限事項」で説明したすべての制限事項が、各UNIONブロック内の副問合せに適用されます。
- 副問合せが多対1の場合でも、すべての結合列をマスター表のマテリアライズド・ビュー・ログに含める必要があります。
- 「複合マテリアライズド・ビュー」で説明したすべての制限事項が、UNIONSを含む句に適用されます。
- 副問合せを含むUNIONSを使用するマテリアライズド・ビューの例
  副問合せを含むUNIONSを使用するマテリアライズド・ビューを作成する例を示します。

親トピック: マテリアライズド・ビューを使用したデータのサブセット化

36.6.4.1 副問合せを含むUNIONを使用するマテリアライズド・ビューの例

副問合せを含むUNIONを使用するマテリアライズド・ビューを作成する例を示します。

次の文は、oe.ordersマテリアライズド・ビューを作成します。各UNIONブロック内の副問合せが、「副問合せを使用するマテリアライズド・ビューでの制限事項」で説明した副問合せに関する制限事項を満たしているため、このマテリアライズド・ビューは高速リフレッシュができます。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
  SELECT * FROM oe.orders@orc1.example.com o
  WHERE EXISTS
    (SELECT * FROM oe.customers@orc1.example.com c
     WHERE o.customer_id = c.customer_id
     AND c.credit_limit > 50)
UNION
  SELECT *
  FROM oe.orders@orc1.example.com o
  WHERE EXISTS
    (SELECT * FROM oe.customers@orc1.example.com c
     WHERE o.customer_id = c.customer_id
     AND c.account_mgr_id = 30);
```

副問合せには、各表は1つのEXISTS式にしか含められないという制限事項があります。ここでは、customers表が2つのEXISTS式で使用されていますが、これらのEXISTS式は別々のUNIONブロックにあります。「副問合せを使用するマテリアライズド・ビューでの制限事項」で説明した制限事項は、各UNIONブロックに対してのみ適用されるもので、CREATE MATERIALIZED VIEW文全体に適用されるものではないため、このマテリアライズド・ビューは高速リフレッシュが可能です。

これに対して、次の文で作成されたマテリアライズド・ビューは高速リフレッシュができません。これは、orders表が同一UNIONブロック内の2つのEXISTS式で参照されているためです。

```
CREATE MATERIALIZED VIEW oe.orders AS
  SELECT * FROM oe.orders@orc1.example.com o
  WHERE EXISTS
    (SELECT * FROM oe.customers@orc1.example.com c
     WHERE o.customer_id = c.customer_id  -- first reference to orders table
     AND c.credit_limit > 50
```

```
     AND EXISTS
        (SELECT * FROM oe.orders@orc1.example.com o
         WHERE order_total > 5000
         AND o.customer_id = c.customer_id)) -- second reference to orders table
UNION
  SELECT *
  FROM oe.orders@orc1.example.com o
  WHERE EXISTS
    (SELECT * FROM oe.customers@orc1.example.com c
     WHERE o.customer_id = c.customer_id
     AND c.account_mgr_id = 30);
```

関連項目:

「マテリアライズド・ビューの高速リフレッシュ機能に関する判断」

親トピック: 副問合せを含むUNIONを使用するマテリアライズド・ビューでの制限事項

36.7 マテリアライズド・ビューのリフレッシュ

マテリアライズド・ビューとマスター表との一貫性を確保するには、マテリアライズド・ビューを定期的に**リフレッシュ**する必要があります。

マテリアライズド・ビューをリフレッシュするには、次の3つの方法があります。

- マテリアライズド・ビュー・ログを使用して、最後のリフレッシュ以降に変更されている行のみを更新する**高速リフレッシュ**。
- マテリアライズド・ビュー全体を更新する**完全リフレッシュ**。
- 可能なときに高速リフレッシュを実行する**強制リフレッシュ**。高速リフレッシュが実行できない場合は、完全リフレッシュが実行されます。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.8 リフレッシュ・グループ

マテリアライズド・ビュー間のトランザクションの一貫性を保つことが重要なときは、マテリアライズド・ビューを**リフレッシュ・グループ**に編成できます。

リフレッシュ・グループをリフレッシュすることで、リフレッシュ・グループ内のすべてのマテリアライズド・ビューのデータが、トランザクションの一貫性を保つ特定の時点のデータと確実に一致します。リフレッシュ・グループ内のマテリアライズド・ビューを個別にリフレッシュすることも可能ですが、リフレッシュ・グループ内のその他のマテリアライズド・ビューはリフレッシュされないため、リフレッシュ・グループを編成する意味がなくなります。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.9 マテリアライズド・ビュー・ログ

マテリアライズド・ビュー・ログは、マテリアライズド・ビューのマスター表が含まれているデータベースの表です。マスター表へのすべてのDML変更が記録されます。

マテリアライズド・ビュー・ログは単一のマスター表に関連付けられ、マスター表からリフレッシュされるマテリアライズド・ビューの数とは無関係に、各マスター表のマテリアライズド・ビュー・ログは1つのみです。マテリアライズド・ビューの高速リフレッシュは、マテリアライズド・ビューのマスター表にマテリアライズド・ビュー・ログが存在する場合のみ可能です。マテリアライズド・ビューを高速リフレッ

シュすると、マテリアライズド・ビューと対応付けられたマテリアライズド・ビュー・ログのエントリで、最後のリフレッシュ以降に追加されたエントリがマテリアライズド・ビューに適用されます。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.10 マテリアライズド・ビューおよびユーザー定義のデータ型

ユーザー定義データ型を含むマテリアライズド・ビューには、特別な考慮事項があります。

- オブジェクト型およびコレクションでのマテリアライズド・ビューの動作方法
  レプリケーション環境で、マスター・データベースとマテリアライズド・ビュー・データベース間でオブジェクト型およびオブジェクトをレプリケートできます。
- レプリケーション・データベースでの型の一致
  ユーザー定義型には、オブジェクト、ネストした表、VARRAY、索引タイプなど、CREATE TYPE文で作成された型がすべて含まれます。ユーザー定義型に基づいてスキーマ・オブジェクトをレプリケートするには、マスター・データベースおよびマテリアライズド・ビュー・データベースに、それらのユーザー定義型が存在し、それらが同一である必要があります。
- 列オブジェクトを使用したマスターの列のサブセット化
  読取り専用マテリアライズド・ビューでは、その他の属性はレプリケートせずに列オブジェクトの特定の属性のみをレプリケートできます。
- オブジェクト表に基づいたマテリアライズド・ビュー
  オブジェクト表に基づいてビューを作成できます。
- コレクション列を含むマテリアライズド・ビュー
  コレクション列は、varrayおよびネストされた表データのタイプに基づいた列です。Oracleではコレクション列を含むマテリアライズド・ビューの作成をサポートしています。
- REF列を含むマテリアライズド・ビュー
  マテリアライズド・ビューにREF列を含めることができます。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.10.1 オブジェクト型およびコレクションでのマテリアライズド・ビューの動作方法

レプリケーション環境で、マスター・データベースとマテリアライズド・ビュー・データベース間でオブジェクト型およびオブジェクトをレプリケートできます。

Oracleのオブジェクト型はユーザー定義データ型で、これを使用すると現実の複雑なエンティティ(顧客や注文など)をデータベース内に単一エンティティ(オブジェクト)としてモデル化できます。オブジェクト型は、CREATE TYPE . . . AS OBJECT文を使用して作成します。

表の1列を占有するOracleオブジェクトを、列オブジェクトと呼びます。一般に、列オブジェクトが含まれている表にはその他の列も含まれており、このような列のデータ型は組込みデータ型(VARCHAR2やNUMBERなど)の場合があります。オブジェクト表は、すべての行がオブジェクトを表す特殊な表です。オブジェクト表の各行は行オブジェクトです。

コレクションをレプリケートすることもできます。コレクションとは、VARRAYデータ型およびネストした表のデータ型に基づいたユーザー定義データ型です。VARRAYはCREATE TYPE . . . AS VARRAY文を、ネストした表はCREATE TYPE . . . AS TABLE文を使用して作成します。


ノート:

- ユーザー定義型またはOracle 提供の型を含むマスター表に基づいてコミット時にリフレッシュするマテリアライズド・ビューを作成することはできません。ON COMMIT オプションが指定されたマテリアライズド・ビューとは、CREATE MATERIALIZED VIEW 文で ON COMMIT REFRESH 句を使用して作成するマテリアライズド・ビューです。

- タイプ継承および NOT FINAL 句で作成されたタイプはサポートされません。

関連項目:

- ユーザー定義型、Oracleオブジェクトおよびコレクションの詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。この項では、これらの概念の基本的な理解があることを前提としています。
- ユーザー定義型およびオラクル社が提供する型の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: マテリアライズド・ビューおよびユーザー定義のデータ型

36.10.2 レプリケーション・データベースでの型の一致

ユーザー定義型には、オブジェクト、ネストした表、VARRAY、索引タイプなど、CREATE TYPE文で作成された型がすべて含まれます。ユーザー定義型に基づいてスキーマ・オブジェクトをレプリケートするには、マスター・データベースおよびマテリアライズド・ビュー・データベースに、それらのユーザー定義型が存在し、それらが同一である必要があります。

ユーザー定義型とそれに基づいたスキーマ・オブジェクトをレプリケートする場合は、次の条件が適用されます。

- マスター・データベースとマテリアライズド・ビュー・データベースでレプリケートされるユーザー定義型は、これらの型に依存するマテリアライズド・ビューを作成する前にマテリアライズド・ビュー・データベースで作成する必要があります。
- ユーザー定義型を含むマテリアライズド・ビューを作成するには、マテリアライズド・ビューの基になるすべてのマスター表が同一のマスター・データベースに存在する必要があります。
- ユーザー定義型は、すべてのデータベースで同一である必要があります。
  - レプリケートされる各ユーザー定義型について、オブジェクト識別子(OID)、スキーマ所有者および型の名前を全レプリケーション・データベースで同一にする必要があります。
  - ユーザー定義型がオブジェクト型の場合は、オブジェクト型の属性の順序とデータ型がすべてのデータベースで一致している必要があります。属性の順序とデータ型は、オブジェクト型を作成するときに設定します。たとえば、次のようなオブジェクト型の場合を考えます。

```
CREATE TYPE cust_address_typ AS OBJECT
     (street_address     VARCHAR2(40),
      postal_code        VARCHAR2(10),
      city               VARCHAR2(30),
      state_province     VARCHAR2(10),
      country_id         CHAR(2));
/
```

すべてのデータベースで、street_addressはこの型の最初の属性でVARCHAR2(40)、postal_codeは2番目の属性でVARCHAR2(10)、cityは3番目の属性でVARCHAR2(30)、というようになっている必要があります。

  - すべてのデータベースで、ユーザー定義型のハッシュコードが一致している必要があります。Oracleでは、ユー

ザー定義型が検査され、ハッシュコードが割り当てられます。この検査では、型の属性、属性の順序および型の名前が調べられます。複数の型についてこれらの項目がすべて同じである場合、型のハッシュコードは同じです。型のハッシュコードは、DBA_TYPE_VERSIONSデータ・ディクショナリ・ビューを問い合せて表示できます。

型を作成するには、マテリアライズド・ビュー・データベースでCREATE TYPE文を使用できます。型を使用する読取り専用マテリアライズド・ビュー・を作成するには、これを行うことが必要な場合があります。

この方法を選択する場合は、次のことを確認する必要があります。

- マテリアライズド・ビュー・データベースとマスター・データベースで型が同一スキーマ内にあること。
- マテリアライズド・ビュー・データベースとマスター・データベースで型の属性および順序が同一であること。
- マテリアライズド・ビュー・データベースとマスター・データベースで型の各属性のデータ型が同一であること。
- マテリアライズド・ビュー・データベースとマスター・でたべで型のオブジェクト識別子が同一であること。

型のオブジェクト識別子は、DBA_TYPESデータ・ディクショナリ・ビューを問い合せると検出できます。たとえば、cust_address_typのオブジェクト識別子(OID)を検出するには、次の問合せを入力します。

```
SELECT TYPE_OID FROM DBA_TYPES WHERE TYPE_NAME = 'CUST_ADDRESS_TYP';
TYPE_OID
--------------------------------
6F9BC33653681B7CE03400400B40A607
```

たとえば、マスター・データベースでの型のOIDを知ることができたら、次のステップを実行して、マテリアライズド・ビュー・サイトでこの型を作成できます。

1. マスター・データベースでこの型を所有するユーザーとして、マテリアライズド・ビュー・データベースにログインします。このユーザーがマテリアライズド・ビュー・データベースに存在しない場合は、ユーザーを作成します。
2. CREATE TYPE文を発行し、OIDを指定します。

```
CREATE TYPE oe.cust_address_typ OID '6F9BC33653681B7CE03400400B40A607'
     AS OBJECT (
     street_address     VARCHAR2(40),
     postal_code        VARCHAR2(10),
     city               VARCHAR2(30),
     state_province     VARCHAR2(10),
     country_id         CHAR(2));
/
```

これで、マテリアライズド・ビュー・データベースでこの型を使用できるようになりました。

親トピック: マテリアライズド・ビューおよびユーザー定義のデータ型

36.10.3 列オブジェクトを使用したマスターの列のサブセット化

読取り専用マテリアライズド・ビューでは、その他の属性はレプリケートせずに列オブジェクトの特定の属性のみをレプリケートできます。

たとえば、前述のcust_address_typユーザー定義データ型を使用して、customers_subマスター表がマスター・データベースorc1.example.comに作成されているとします。

```
CREATE TABLE oe.customers_sub (
      customer_id        NUMBER(6)  PRIMARY KEY,
      cust_first_name    VARCHAR2(20),
      cust_last_name     VARCHAR2(20),
      cust_address       oe.cust_address_typ);
```

次の読取り専用マテリアライズド・ビューをリモート・マテリアライズド・ビュー・データベースに作成できます。

```
CREATE MATERIALIZED VIEW oe.customers_mv1 AS
   SELECT customer_id, cust_last_name, c.cust_address.postal_code
   FROM oe.customers_sub@orc1.example.com c;
```

ここでは、postal_code属性がcust_address列オブジェクト内に指定されています。

親トピック: マテリアライズド・ビューおよびユーザー定義のデータ型

36.10.4 オブジェクト表に基づいたマテリアライズド・ビュー

オブジェクト表に基づいてマテリアライズド・ビューを作成できます。

- オブジェクト表に基づいたマテリアライズド・ビューについて
  マテリアライズド・ビューがオブジェクト表に基づいてOF type句を使用して作成されている場合、このマテリアライズド・ビューはオブジェクト・マテリアライズド・ビューと呼ばれます。
- オブジェクト表に基づいてOF type句を使用せずに作成したマテリアライズド・ビュー
  OF type句を使用せずに、オブジェクト表に基づいてマテリアライズド・ビューを作成すると、マテリアライズド・ビューでは、基になるオブジェクト表のオブジェクト・プロパティが失われます。
- オブジェクト・マテリアライズド・ビューでのOIDの保持
  オブジェクト・マテリアライズド・ビューは、マスターのオブジェクト識別子(OID)の仕様を継承します。

親トピック: マテリアライズド・ビューおよびユーザー定義のデータ型

36.10.4.1 オブジェクト表に基づいたマテリアライズド・ビューについて

マテリアライズド・ビューがオブジェクト表に基づいてOF type句を使用して作成されている場合、このマテリアライズド・ビューはオブジェクト・マテリアライズド・ビューと呼ばれます。

オブジェクト・マテリアライズド・ビューは、オブジェクト表と同じ構造です。つまり、オブジェクト・マテリアライズド・ビューは行オブジェクトで構成されます。

オブジェクト表に基づいたマテリアライズド・ビューをOF type句を使用せずに作成した場合、そのマテリアライズド・ビューはオブジェクト・マテリアライズド・ビューではありません。つまり、このようなマテリアライズド・ビューには、行オブジェクトではなく通常の行が含まれています。

オブジェクト表に基づいたマテリアライズド・ビューを作成するには、マテリアライズド・ビューが依存する型がマテリアライズド・ビュー・データベースに存在する必要があり、それぞれの型はマスター・データベースと同じオブジェクト識別子を持つ必要があります。

関連項目:

オブジェクト・マテリアライズド・ビューを作成する例は、「読取り専用マテリアライズド・ビューの作成」

親トピック: オブジェクト表に基づいたマテリアライズド・ビュー

36.10.4.2 オブジェクト表に基づいてOF type句を使用せずに作成したマテリアライズド・ビュー

OF type句を使用せずに、オブジェクト表に基づいてマテリアライズド・ビューを作成すると、マテリアライズド・ビューでは、基になるオブジェクト表のオブジェクト・プロパティが失われます。

つまり、作成された読取り専用マテリアライズド・ビューにはマスター表の列が1つ以上含まれますが、それぞれの行はリレーショナル表の行として機能します。行は行オブジェクトではありません。

たとえば、次のSQL文を使用して、categories_tabマスター表に基づいたマテリアライズド・ビューを作成できます。

```
CREATE MATERIALIZED VIEW oe.categories_relmv
   AS SELECT * FROM oe.categories_tab@orc1.example.com;
```

この場合、このマテリアライズド・ビューの行は、リレーショナル表の行と同様に機能します。

親トピック: オブジェクト表に基づいたマテリアライズド・ビュー

36.10.4.3 オブジェクト・マテリアライズド・ビューでのOIDの保持

オブジェクト・マテリアライズド・ビューは、マスターのオブジェクト識別子(OID)指定を継承します。

マスター表に主キー・ベースのOIDがある場合、マテリアライズド・ビューの行オブジェクトのOIDは主キー・ベースになります。マスター表にシステム生成されたOIDがある場合、マテリアライズド・ビューの行オブジェクトのOIDはシステム生成されたOIDです。また、オブジェクト・マテリアライズド・ビューの各行のOIDは、マスター表内の同じ行のOIDと一致し、マテリアライズド・ビューのリフレッシュ中、OIDは保持されます。その結果、オブジェクト表の行へのREFは、マテリアライズド・ビュー・データベースで有効なままです。

親トピック: オブジェクト表に基づいたマテリアライズド・ビュー

36.10.5 コレクション列を含むマテリアライズド・ビュー

コレクション列とは、VARRAYデータ型およびネストした表のデータ型に基づいた列です。Oracleではコレクション列を含むマテリアライズド・ビューの作成をサポートしています。

コレクション列がネストした表の場合は、マテリアライズド・ビューの作成中にオプションとしてnested_table_storage_clauseを指定できます。nested_table_storage_clauseを使用すると、ネストした表用の記憶表の名前をマテリアライズド・ビューに指定できます。

たとえば、マスター・データベースorc1.example.comにマスター表people_reltabを作成し、この表にネストした表phones_ntabが含まれているとします。

```
CREATE TYPE oe.phone_typ AS OBJECT (
   location    VARCHAR2(15),
   num         VARCHAR2(14));
/
CREATE TYPE oe.phone_ntabtyp AS TABLE OF oe.phone_typ;
/
CREATE TABLE oe.people_reltab (
   id               NUMBER(4) CONSTRAINT pk_people_reltab PRIMARY KEY,
   first_name       VARCHAR2(20),
   last_name        VARCHAR2(20),
   phones_ntab      oe.phone_ntabtyp)
   NESTED TABLE phones_ntab STORE AS phone_store_ntab
   ((PRIMARY KEY (NESTED_TABLE_ID, location)));
```

このSQL文の最後の行のPRIMARY KEY指定に注意してください。親表に基づいてマテリアライズド・ビューを作成する場合は、記憶表に主キーを指定する必要があります。この例では、記憶表はphone_store_ntabで、親表はpeople_reltabです。

高速リフレッシュ可能なマテリアライズド・ビューを作成する場合は、親表と記憶表の両方にマテリアライズド・ビュー・ログを作成し、ネストした表の列を親表のマテリアライズド・ビュー・ログのフィルタ列として指定します。

```
CREATE MATERIALIZED VIEW LOG ON oe.people_reltab;
ALTER MATERIALIZED VIEW LOG ON oe.people_reltab ADD(phones_ntab);
CREATE MATERIALIZED VIEW LOG ON oe.phone_store_ntab WITH PRIMARY KEY;
```

マテリアライズド・ビュー・データベースで、各型のオブジェクト識別子がマスター・データベースのオブジェクト識別子と同じになるように、必要な型を作成します。この後、people_reltabに基づいてマテリアライズド・ビューを作成し、次の文を使用してその記憶表を指定できます。

```
CREATE MATERIALIZED VIEW oe.people_reltab_mv
   NESTED TABLE phones_ntab STORE AS phone_store_ntab_mv
   REFRESH FAST AS SELECT * FROM oe.people_reltab@orc1.example.com;
```

この場合、nested_table_storage_clauseは前述の例の「NESTED TABLE」で始まる行であり、これによって記憶表の名前がphone_store_ntab_mvであることを指定します。nested_table_storage_clauseはオプションです。この句を指定しない場合、Oracleデータベースにより自動的に記憶域表の名前が付けられます。記憶表の名前を表示するには、DBA_NESTED_TABLESデータ・ディクショナリ表を問い合せます。

記憶表には次の特性があります。

- 個別のセカンダリ・マテリアライズド・ビューです。
- ネストした表を含むマテリアライズド・ビューをリフレッシュすると、自動的にリフレッシュされます。
- ネストした表を含むマテリアライズド・ビューを削除すると、自動的に削除されます。
- マスターの記憶表の主キー制約を継承します。

記憶表はマスター表の記憶表の主キー制約を継承するため、STORE AS句にはPRIMARY KEYを指定しないでください。

マテリアライズド・ビュー内のネストした表の記憶表に対して次のアクションを直接実行することはできません。

- 記憶表のリフレッシュ
- 記憶表の変更
- 記憶表の削除
- 記憶表に対するレプリケーション・サポートの生成

これらのアクションは、ネストした表を含むマテリアライズド・ビューに対する実行時に間接的に発生します。さらに、記憶表内の列のサブセットはレプリケートできません。

- コレクション列を含むマテリアライズド・ビューに関する制限事項
  コレクション列を含むマテリアライズド・ビューに制限が適用されます。

関連項目:

nested_table_col_propertiesの詳細は、『Oracle Database SQL言語リファレンス』のCREATE TABLE文に関する説明を参照してください。

親トピック: マテリアライズド・ビューおよびユーザー定義のデータ型

36.10.5.1 コレクション列を含むマテリアライズド・ビューに関する制限事項

コレクション列を含むマテリアライズド・ビューに制限が適用されます。

次の制限があります。

- コレクション列の行のサブセット化はできません。ただし、ネストした表の親表に対しては行のサブセット化を使用でき、その結果、マスター・マテリアライズド・ビューのネストした表がサブセット化されます。

- コレクション列の列のサブセット化はできません。
- ネストした表の記憶表には主キーが必要です。
- ネストした表の親表で高速リフレッシュ可能にするには、親表とネストした表の両方の記憶表にマテリアライズド・ビュー・ログが必要です。

親トピック: コレクション列を含むマテリアライズド・ビュー

36.10.6 REF列を含むマテリアライズド・ビュー

マテリアライズド・ビューにREF列を含めることができます。

- REF列を含むマテリアライズド・ビューについて
  REF列を含むマテリアライズド・ビューを作成できます。REFとはOracleの組込みデータ型で、オブジェクト表内の行オブジェクトを指す論理的なポインタです。
- 有効範囲付きREF列
  有効範囲付きREF列を含むマスター表に基づいてマテリアライズド・ビューを作成する場合は、マテリアライズド・ビュー・データベースの別のオブジェクト表またはオブジェクト・マテリアライズド・ビューにREFの有効範囲を再指定できます。
- 有効範囲なしのREF列
  有効範囲なしのREF列を含むリモート・マスター表に基づいてマテリアライズド・ビューを作成した場合、マテリアライズド・ビューにREF列は作成されますが、REFはリモート・データベースを指すため参照先がないとみなされます。
- マテリアライズド・ビュー・ログへのREF列のロギング
  必要な場合、マテリアライズド・ビュー・ログにREF列を記録できます。
- WITH ROWID句を使用して作成されたREF
  REF列にWITH ROWID句が指定されている場合、Oracle DatabaseはREFで参照されているオブジェクトの行IDを維持します。

親トピック: マテリアライズド・ビューおよびユーザー定義のデータ型

36.10.6.1 REF列を含むマテリアライズド・ビューについて

REF列を含むマテリアライズド・ビューを作成できます。REFとはOracleの組込みデータ型で、オブジェクト表内の行オブジェクトを指す論理的なポインタです。

有効範囲付きREFは、指定されたオブジェクト表のみへの参照を含めることができるREFで、有効範囲なしREFは、データベース内で対応するオブジェクト型に基づいたすべてのオブジェクト表に対する参照を含めることができます。有効範囲付き REFは、有効範囲なしREFと比べて必要とする記憶領域が少なく、より効率的なアクセスを提供します。

マテリアライズド・ビューの作成時に、マテリアライズド・ビュー・データベースのローカル・マテリアライズド・ビューまたは表にREF列の有効範囲を再指定できます。有効範囲を再指定しない場合、REF列はリモートのマスター表を指し続けます。有効範囲なしのREF列は、常にマスター表を指し続けます。マテリアライズド・ビュー・データベースのREF列がリモート・マスター表を指す場合、REFは参照先がないとみなされます。SQLでは、参照先がないREFを参照解除すると、NULLが返されます。また、PL/SQLではUTL_OBJECTパッケージを使用したREFの参照解除のみがサポートされ、参照先がないREFに対しては例外が発生します。

親トピック: REF列を含むマテリアライズド・ビュー

36.10.6.2 有効範囲付きREF列

有効範囲付きREF列を含むマスター表に基づいてマテリアライズド・ビューを作成する場合は、マテリアライズド・ビュー・データベースの別のオブジェクト表またはオブジェクト・マテリアライズド・ビューにREFの有効範囲を再指定できます。

通常、REF列の有効範囲は、元のリモート・オブジェクト表ではなくローカル・オブジェクト・マテリアライズド・ビューに変更します。マテリアライズド・ビューの有効範囲を変更するには、CREATE MATERIALIZED VIEW文にSCOPE FOR句を使用するか、マテリアライズド・ビューを作成した後でALTER MATERIALIZED VIEW文を使用します。REF列の有効範囲を変更しないと、マテリアライズド・ビューはマスター表のREF列をそのまま保持します。

たとえば、次の文を使用してorc1.example.comマスター・データベースにcustomers_with_refマスター表を作成するとします。

```
-- Create the user-defined data type cust_address_typ.
CREATE TYPE oe.cust_address_typ AS OBJECT
   (street_address     VARCHAR2(40),
    postal_code        VARCHAR2(10),
    city               VARCHAR2(30),
    state_province     VARCHAR2(10),
    country_id         CHAR(2));
/
-- Create the object table cust_address_objtab.
CREATE TABLE oe.cust_address_objtab OF oe.cust_address_typ;
-- Create table with REF to cust_address_typ.
CREATE TABLE oe.customers_with_ref (
         customer_id        NUMBER(6) PRIMARY KEY,
     cust_first_name    VARCHAR2(20),
     cust_last_name     VARCHAR2(20),
     cust_address       REF oe.cust_address_typ
                          SCOPE IS oe.cust_address_objtab);
```

マスター・データベースの型と同じオブジェクト識別子を持つcust_address_typがマテリアライズド・ビュー・データベースにあると仮定した場合、次の文を使用するとcust_address_objtab_mvオブジェクト・マテリアライズド・ビューを作成できます。

```
CREATE MATERIALIZED VIEW oe.cust_address_objtab_mv OF oe.cust_address_typ AS
   SELECT * FROM oe.cust_address_objtab@orc1.example.com;
```

これで、次の文を使用してcustomers_with_refマスター表のマテリアライズド・ビューを作成し、REFの有効範囲をcust_address_objtab_mvマテリアライズド・ビューに変更できます。

```
CREATE MATERIALIZED VIEW oe.customers_with_ref_mv
   (SCOPE FOR (cust_address) IS oe.cust_address_objtab_mv)
   AS SELECT * FROM oe.customers_with_ref@orc1.example.com;
```

マテリアライズド・ビューを作成するときにSCOPE FOR句を使用する場合は、SCOPE FOR句に指定するマテリアライズド・ビューまたは表を先に作成する必要があります。そうしないと、マテリアライズド・ビューの作成中にSCOPE FOR句を指定できません。たとえば、cust_address_objtab_mvマテリアライズド・ビューを作成する前に、customers_with_ref_mvマテリアライズド・ビューを作成すると、customers_with_ref_mvマテリアライズド・ビューを作成するときにSCOPE FOR句を使用できません。この場合、REFはリモート・マスター・データベースのオブジェクト表を指すため、参照先がないとみなされます。

ただし、マテリアライズド・ビューを作成するときにSCOPE FOR句を使用しない場合でも、SCOPE FOR句を指定するようにマテリアライズド・ビューを変更できます。たとえば、次の文でcustomers_with_ref_mvマテリアライズド・ビューを変更できます。

```
ALTER MATERIALIZED VIEW oe.customers_with_ref_mv
   MODIFY SCOPE FOR (cust_address) IS oe.cust_address_objtab_mv;
```

親トピック: REF列を含むマテリアライズド・ビュー

36.10.6.3 有効範囲なしのREF列

有効範囲なしのREF列を含むリモート・マスター表に基づいてマテリアライズド・ビューを作成した場合、マテリアライズド・ビューにREF列は作成されますが、REFはリモート・データベースを指すため参照先がないとみなされます。

親トピック: REF列を含むマテリアライズド・ビュー

36.10.6.4 マテリアライズド・ビュー・ログへのREF列のロギング

必要な場合、マテリアライズド・ビュー・ログにREF列を記録できます。

関連項目:

「マテリアライズド・ビュー・ログへの列のロギング」

親トピック: REF列を含むマテリアライズド・ビュー

36.10.6.5 WITH ROWID句を使用して作成されたREF

REF列にWITH ROWID句が指定されている場合、Oracle DatabaseはREFで参照されているオブジェクトの行IDを維持します。

Oracleデータベースでは、REFに含まれている行IDを使用して、参照されているオブジェクトを直接検出できます(OID索引から行IDをフェッチする必要はありません)。したがって、WITH ROWID句は行IDヒントを指定するために使用します。WITH ROWID句は、有効範囲付きREFに対してはサポートされていません。

WITH ROWID句を使用して作成されたREFをレプリケートした場合、REFが最初に作成または変更されたデータベースを除き、各レプリケーション・データベースの行IDヒントは正しくありません。他のデータベースではREFのROWID情報は無意味であり、Oracleデータベースでは行IDヒントは自動的に修正されません。無効な行IDヒントは、パフォーマンス上の問題の原因になります。この場合は、ANALYZE TABLE文のVALIDATE STRUCTUREオプションを使用して、各レプリケーション・データベースの正しくない行IDヒントを判断できます。

関連項目:

ANALYZE TABLE文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: REF列を含むマテリアライズド・ビュー

36.11 マスター・データベースでのマテリアライズド・ビューの登録

マスター・データベースでは、マスター表に基づいてマテリアライズド・ビューの情報がOracleデータベースにより自動的に登録されます。

- 登録されたマテリアライズド・ビューに関する情報の表示
  マテリアライズド・ビューは、そのマスター・データベースに登録されます。
- 内部メカニズム
  Oracleデータベースでは、マテリアライズド・ビューが作成されると自動的にマスター・データベースに登録され、マテリアライズド・ビューが削除されると登録解除されます。
- マテリアライズド・ビューの手動による登録
  必要な場合は、登録を手動で維持できます。

親トピック: 読取り専用マテリアライズド・ビューの概念

36.11.1 登録されたマテリアライズド・ビューに関する情報の表示

マテリアライズド・ビューは、そのマスター・データベースに登録されます。

マスター・データベースのDBA_REGISTERED_MVIEWSデータ・ディクショナリ・ビューの問合せにより、リモート・マテリアライズド・ビューに関する次の情報が表示されます。

- 所有者、名前およびマテリアライズド・ビューを含むデータベース
- マテリアライズド・ビューの定義問合せ
- マテリアライズド・ビューのその他の特性(リフレッシュ方式など)

また、マスター・データベースのDBA_MVIEW_REFRESH_TIMESビューの問合せにより、各マテリアライズド・ビューの最後のリフレッシュ時間を取得できます。管理者は、この情報を使用してマテリアライズド・ビューのアクティビティを監視し、マスター表またはマスター・マテリアライズド・ビューの削除、変更または再配置が必要な場合にマテリアライズド・ビュー・データベースへの変更を調整できます。

親トピック: マスター・データベースでのマテリアライズド・ビューの登録

36.11.2 内部メカニズム

Oracleデータベースでは、マテリアライズド・ビューが作成されると自動的にマスター・データベースに登録され、マテリアライズド・ビューが削除されると登録解除されます。

ノート:

Oracle データベースでは、マテリアライズド・ビューの作成時または削除時に、マスター・データベースでマテリアライズド・ビューが登録または登録解除されることを保証できません。作成時に Oracle データベースがマテリアライズド・ビューを正常に登録できない場合は、DBMS_MVIEW パッケージ内の REGISTER_MVIEW プロシージャを使用して手動で登録を完了する必要があります。マテリアライズド・ビューの削除時に、Oracle データベースがマテリアライズド・ビューを正常に登録解除できない場合は、手動で登録を解除するまで、マスター・データベースにマテリアライズド・ビューの登録情報が保持され続けます。複合マテリアライズド・ビューは登録されない場合があります。

親トピック: マスター・データベースでのマテリアライズド・ビューの登録

36.11.3 マテリアライズド・ビューの手動による登録

必要な場合は、登録を手動でメンテナンスできます。

マテリアライズド・ビューの登録情報を追加、変更または削除するには、マスター・データベースのDBMS_MVIEWパッケージ内のREGISTER_MVIEWおよびUNREGISTER_MVIEWプロシージャを使用します。

関連項目:

REGISTER_MVIEWプロシージャおよびUNREGISTER_MVIEWプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

親トピック: マスター・データベースでのマテリアライズド・ビューの登録

37 読取り専用マテリアライズド・ビューのアーキテクチャ

マテリアライズド・ビューのレプリケーションに複数のオブジェクトが使用されています。これらのオブジェクトの一部はオプションで、作成したマテリアライズド・ビュー環境のサポートに必要な場合にのみ使用します。たとえば、高速リフレッシュができない複合マテリアライズド・ビューがある場合は、マスター・データベースにマテリアライズド・ビュー・ログがない可能性があります。

- マスター・データベースのメカニズム
  マスター・データベースでマテリアライズド・ビューをサポートするメカニズムがあります。
- マテリアライズド・ビュー・データベースのメカニズム
  マテリアライズド・ビューを作成すると、そのマテリアライズド・ビューをサポートするための追加のメカニズムがマテリアライズド・ビュー・データベースで作成されます。具体的には、1つ以上の索引が作成されます。
- 編成メカニズム
  いくつかのメカニズムは、マテリアライズド・ビュー・データベースでマテリアライズド・ビューを編成します。これらのメカニズムは、マテリアライズド・ビュー・データベースとそのマスター・データベース間の編成上の一貫性を維持します。
- リフレッシュ処理
  マテリアライズド・ビューとマスター表との一貫性を確保するには、マテリアライズド・ビューを定期的にリフレッシュする必要があります。

親トピック: 読取り専用マテリアライズド・ビューの管理

37.1 マスター・データベースのメカニズム

マスター・データベースでマテリアライズド・ビューをサポートするメカニズムがあります。

- マスター・データベース・オブジェクト
  マテリアライズド・ビューの高速リフレッシュをサポートするために、マスター・データベースで特定のデータベース・オブジェクトが必要です。
- マスター表
  マスター表は、マテリアライズド・ビューの基礎です。
- マテリアライズド・ビュー・ログの内部トリガー
  DMLを使用してマスター表に変更が加えられると、内部トリガーにより、影響を受けた行に関する情報がマテリアライズド・ビュー・ログに記録されます。
- マテリアライズド・ビュー・ログ
  マテリアライズド・ビュー・ログは、マテリアライズド・ビューの高速リフレッシュを可能にします。

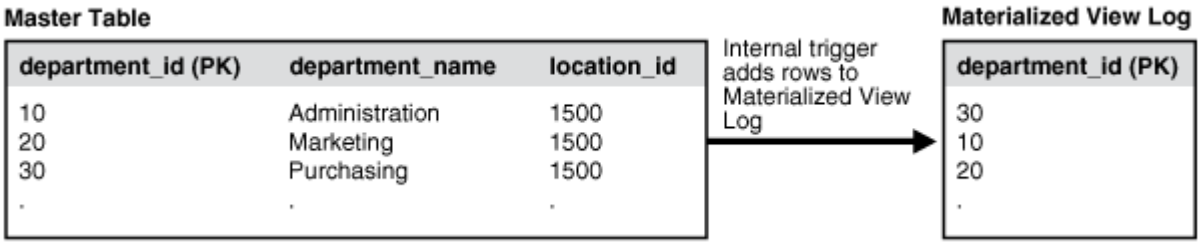
親トピック: 読取り専用マテリアライズド・ビューのアーキテクチャ

37.1.1 マスター・データベース・オブジェクト

マテリアライズド・ビューの高速リフレッシュをサポートするために、マスター・データベースで特定のデータベース・オブジェクトが必要です。

マテリアライズド・ビューの高速リフレッシュをサポートするために、図37-1に表示された3つのデータベース・オブジェクトがマスター・データベースで必要です。

図37-1 マスター・データベース・オブジェクト



親トピック: マスター・データベースのメカニズム

37.1.2 マスター表

マスター表は、マテリアライズド・ビューの基礎です。

マスター表は、ターゲット・マスター・データベースにあります。マテリアライズド・ビューは、1つのマスター・データベースのみを指します。マテリアライズド・ビュー・ログに記録されたマスター表へのデータ操作言語(DML)の変更は、リフレッシュ・プロセス中にマテリアライズド・ビューに伝播されます。

ノート:

高速リフレッシュ可能なマテリアライズド・ビューは、マスター表またはマスター表のシノニムに基づいている必要があります。完全リフレッシュは、ビューに基づいたマテリアライズド・ビューで使用する必要があります。

親トピック: マスター・データベースのメカニズム

37.1.3 マテリアライズド・ビュー・ログの内部トリガー

DMLを使用してマスター表に変更が加えられると、内部トリガーにより、影響を受けた行に関する情報がマテリアライズド・ビュー・ログに記録されます。

この情報には、主キー、行IDまたはオブジェクト識別子(あるいはこれらすべて)の値のみでなく、マテリアライズド・ビュー・ログにロギングされる他の列の値も含まれます。これは、ターゲット・マスター表のマテリアライズド・ビュー・ログを作成するときに自動的にアクティブになるAFTER ROW内部トリガーです。このトリガーは、INSERT文、UPDATE文またはDELETE文により表のデータが変更されるたびに、マテリアライズド・ビュー・ログに行を挿入します。このトリガーは、常に最後に起動されるトリガーです。

ノート:

マテリアライズド・ビューに副問合せが含まれている場合は、副問合せで参照される列のログが必要になることがあります。副問合せのマテリアライズド・ビューの詳細は、「マテリアライズド・ビューを使用したデータのサブセット化」、ログを必要とする列の詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

親トピック: マスター・データベースのメカニズム

37.1.4 マテリアライズド・ビュー・ログ

マテリアライズド・ビュー・ログは、マテリアライズド・ビューの高速リフレッシュを可能にします。

- マテリアライズド・ビュー・ログについて
  マスター表に基づいてマテリアライズド・ビューの高速リフレッシュを実行する場合は、マスター表のマテリアライズド・

ビュー・ログが必要です。

- マテリアライズド・ビュー・ログに記録される列
  マテリアライズド・ビュー・ログを作成する場合は、必要に応じてログに列を追加できます。
- 異なるスキーマへのマテリアライズド・ビュー・ログのインポートに関する制限事項
  マテリアライズド・ビュー・ログは、DDL文に明示的に指定されたスキーマ名でエクスポートされます。そのため、マテリアライズド・ビュー・ログは、エクスポート元のスキーマとは異なるスキーマにはインポートできません。

親トピック: マスター・データベースのメカニズム

37.1.4.1 マテリアライズド・ビュー・ログについて

マスター表に基づいてマテリアライズド・ビューの高速リフレッシュを実行する場合は、マスター表のマテリアライズド・ビュー・ログが必要です。

マスター表のマテリアライズド・ビュー・ログを作成すると、Oracleデータベースにより基になる表がマテリアライズド・ビュー・ログとして作成されます。マテリアライズド・ビュー・ログには、マスター表で更新されている行の主キー、行IDまたはオブジェクト識別子を含めることができます。マテリアライズド・ビュー・ログには、副問合せを含むマテリアライズド・ビューの高速リフレッシュをサポートする他の列を含めることもできます。

マテリアライズド・ビュー・ログの表の名前は、MLOG$_master_table_nameです。マテリアライズド・ビュー・ログは、ターゲット・マスター表と同じスキーマに作成されます。1つのマテリアライズド・ビュー・ログは、マスター表の複数のマテリアライズド・ビューをサポートできます。前の項で説明したように、ターゲット・マスター表に対してDMLトランザクションが発生するたびに、内部トリガーによってマテリアライズド・ビュー・ログに変更情報が追加されます。

マテリアライズド・ビュー・ログには次のタイプがあります。

- 主キー: マテリアライズド・ビューは、影響を受けた行の主キーに基づいてマスター表への変更内容を記録します。
- 行ID: マテリアライズド・ビューは、影響を受けた行の行IDに基づいてマスター表への変更内容を記録します。
- オブジェクトID: マテリアライズド・ビューは、影響を受けた行オブジェクトのオブジェクト識別子に基づいてマスター・オブジェクト表への変更内容を記録します。
- 組合せ: マテリアライズド・ビューは、3つのオプションの任意の組合せに基づいてマスター表への変更を記録します。主キー、ROWID、影響を受ける行のオブジェクト識別子に基づいて変更を記録できます。このようなマテリアライズド・ビュー・ログは、主キー、ROWIDおよびオブジェクト・マテリアライズド・ビューをサポートし、マスター表に基づく3つのすべてのタイプのマテリアライズド・ビューのある環境で役立ちます。

組合せマテリアライズド・ビュー・ログは、1種類のタイプの値を追跡するマテリアライズド・ビュー・ログと同じように機能しますが、複数のタイプの値が記録されるという点のみが異なります。たとえば、組合せマテリアライズド・ビュー・ログは、影響を受けた行の主キーと行IDの両方を追跡できます。

主キーに基づいたマテリアライズド・ビュー・ログと行IDに基づいたマテリアライズド・ビュー・ログの間に大きな違いはありませんが(影響を受けた行を記録する際に、主キーと物理行IDのどちらを使用するかという点のみが異なります)、実際に与える影響は非常に大きくなります。ROWIDマテリアライズド・ビューとマテリアライズド・ビュー・ログを使用すると、ROWIDマテリアライズド・ビューを高速リフレッシュできなくなるため、マスター表の再編成および切捨てが難しくなります。マスター表の再編成または切捨てを実行した場合はマスター表の行IDが変更されるため、ROWIDマテリアライズド・ビューは完全リフレッシュする必要があります。


ノート:

- マスター表の再編成には、DBMS_MVIEW パッケージの BEGIN_TABLE_REORGANIZATION および END_TABLE_REORGANIZATION プロシージャを使用します。詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

- マスター表を再編成するもう 1 つの方法としては、表のオンライン再定義がありますが、オンライン再定義は、マテリアライズド・ビュー・ログおよびマテリアライズド・ビューを持つマスター表では使用できません。オンライン再定義は、マテリアライズド・ビュー・ログを持たないマスター表に対して実行できます。「表のオンライン再定義」を参照してください。

関連項目:

「マテリアライズド・ビュー・ログの作成」

親トピック: マテリアライズド・ビュー・ログ

37.1.4.2 マテリアライズド・ビュー・ログに記録される列

マテリアライズド・ビュー・ログを作成する場合は、必要に応じてログに列を追加できます。

マテリアライズド・ビューを高速リフレッシュするには、マテリアライズド・ビュー・ログに次のタイプの列を追加する必要があります。

- 副問合せのWHERE句で参照される列で、等価結合の一部ではなく、主キー列でもない列。このような列は、フィルタ列と呼ばれます。
- 副問合せが多対多または1対多のいずれかの関係の場合に、等価結合内の列で主キー列ではない列。副問合せが多対1の場合は、マテリアライズド・ビュー・ログに結合列を追加する必要はありません。

コレクション列はマテリアライズド・ビュー・ログに追加できません。また、完全リフレッシュを使用するマテリアライズド・ビューの場合は、マテリアライズド・ビュー・ログは不要です。

たとえば、次のようなDDLを考えてみます。

```
1) CREATE MATERIALIZED VIEW oe.customers REFRESH FAST AS
2)  SELECT * FROM oe.customers@orc1.example.com c
3)  WHERE EXISTS
4)    (SELECT * FROM oe.orders@orc1.example.com o
5)     WHERE c.customer_id = o.customer_id AND o.order_total > 20000);
```

このDDLの5行目では、WHERE句内で3つの列が参照されています。列orders.customer_idとcustomers.customer_idは等価結合句の一部として参照されています。customers.customer_idは主キー列であるため、この列はデフォルトでロギングされますが、orders.customer_idは主キー列ではないため、マテリアライズド・ビュー・ログに追加する必要があります。また、列orders.order_totalは追加フィルタ列であるため、これもロギングする必要があります。

したがって、orders.customer_idとorders.order_totalをoe.orders表のマテリアライズド・ビュー・ログに追加します。

作成するマテリアライズド・ビューの定義問合せを分析し、マテリアライズド・ビュー・ログに追加する必要がある列を識別することをお薦めします。マテリアライズド・ビュー・ログに列を追加せずに、追加列を必要とするマテリアライズド・ビューを作成またはリフレッシュしようとした場合、マテリアライズド・ビューの作成またはリフレッシュは失敗します。

ノート:

マテリアライズド・ビューを高速リフレッシュするには、副問合せ内の結合列をマテリアライズド・ビュー・ログに追加する必要があります(結合列が主キー列ではなく、副問合せが多対多または 1 対多のいずれかの関係の場合)。副問合せが多対 1 の場合は、マテリアライズド・ビュー・ログに結合列を追加する必要はありません。

関連項目:

- マテリアライズド・ビュー・ログの作成方法については、「マテリアライズド・ビュー・ログの作成」
- 「マテリアライズド・ビュー・ログへの列のロギング」
- 副問合せを含むマテリアライズド・ビューについては、「マテリアライズド・ビューを使用したデータのサブセット化」
- 副問合せを含むマテリアライズド・ビューに関する制限については、「副問合せを使用するマテリアライズド・ビューでの制限事項」を参照してください

親トピック: マテリアライズド・ビュー・ログ

37.1.4.3 異なるスキーマへのマテリアライズド・ビュー・ログのインポートに関する制限事項

マテリアライズド・ビュー・ログは、DDL文で明示的に指定されたスキーマ名でエクスポートされます。そのため、マテリアライズド・ビュー・ログは、エクスポート元のスキーマとは異なるスキーマにはインポートできません。

指定したスキーマのマテリアライズド・ビュー・ログを含むエクスポート・ダンプ・ファイルをインポートするために、REMAP_SCHEMAインポート・パラメータを指定するデータ・ポンプ・インポート・ユーティリティを使用してインポートを試行すると、インポート・ログ・ファイルにエラーが書き込まれ、項目はインポートされません。

親トピック: マテリアライズド・ビュー・ログ

37.2 マテリアライズド・ビュー・データベースのメカニズム

マテリアライズド・ビューを作成すると、そのマテリアライズド・ビューをサポートするための追加のメカニズムがマテリアライズド・ビュー・データベースで作成されます。具体的には、1つ以上の索引が作成されます。

ノート:

マテリアライズド・ビュー名のサイズの上限は 30 バイトです。31 バイト以上の名前を付けてマテリアライズド・ビューを作成しようとすると、Oracle データベースがエラーを返します。

- マテリアライズド・ビューの索引
  主キーおよびROWIDのマテリアライズド・ビューのそれぞれに対して、リモート・マテリアライズド・ビュー・データベースに1つ以上の索引が作成されます。

親トピック: 読取り専用マテリアライズド・ビューのアーキテクチャ

37.2.1 マテリアライズド・ビューの索引

主キーおよびROWIDのマテリアライズド・ビューのそれぞれに対して、リモート・マテリアライズド・ビュー・データベースに1つ以上の索

引が作成されます。

主キーのマテリアライズド・ビューの場合、索引は、ターゲット・マスター表の主キーに対応し、名前に_PKが含まれます。マテリアライズド・ビュー・データベースに同じ名前の索引がすでに存在する場合は、番号が追加されます。ROWIDマテリアライズド・ビューの場合、索引はROWID列に対応し、名前にI_SNAP$_が含まれます。副問合せを含むマテリアライズド・ビューの高速リフレッシュをサポートするために、リモート・マテリアライズド・ビュー・データベースに追加の索引が作成されることもあります。

親トピック: マテリアライズド・ビュー・データベースのメカニズム

37.3 編成メカニズム

いくつかのメカニズムは、マテリアライズド・ビュー・データベースでマテリアライズド・ビューを編成します。これらのメカニズムは、マテリアライズド・ビュー・データベースとそのマスター・データベース間の編成上の一貫性を維持します。

- リフレッシュ・グループ
  複数のマテリアライズド・ビュー間の参照整合性およびトランザクション(読取り)一貫性を保つために、Oracle Databaseでは、個々のマテリアライズド・ビューをリフレッシュ・グループの一部としてリフレッシュできます。
- リフレッシュ・グループのサイズ
  リフレッシュ・グループのサイズを決定するときは、サイズの違いによるメリットとデメリットの両方を考慮する必要があります。

親トピック: 読取り専用マテリアライズド・ビューのアーキテクチャ

37.3.1 リフレッシュ・グループ

複数のマテリアライズド・ビュー間の参照整合性およびトランザクション(読取り)一貫性を保つために、Oracle Databaseでは、個々のマテリアライズド・ビューをリフレッシュ・グループの一部としてリフレッシュできます。

リフレッシュ・グループ内のすべてのマテリアライズド・ビューのリフレッシュが終了すると、グループ内のすべてのマテリアライズド・ビューのデータはトランザクションの一貫性を保つ特定の時点のデータに対応します。

親トピック: 編成メカニズム

37.3.2 リフレッシュ・グループのサイズ

リフレッシュ・グループのサイズを決定するときは、サイズの違いによるメリットとデメリットの両方を考慮する必要があります。

Oracleデータベースは、大規模なリフレッシュ・グループのために最適化されています。したがって、大規模リフレッシュ・グループと小規模リフレッシュ・グループを比較した場合、グループ内のマテリアライズド・ビューが類似していると想定すると、大規模リフレッシュ・グループの方が同じ数のマテリアライズド・ビューを高速にリフレッシュできます。たとえば、マテリアライズド・ビューを20ずつ含む5つのリフレッシュ・グループよりも、100のマテリアライズド・ビューを含む1つのリフレッシュ・グループの方が高速にリフレッシュできます。また、大規模なリフレッシュ・グループでは、1回のPL/SQLサブプログラムの呼出しにより、多数のマテリアライズド・ビューをリフレッシュできます。

リフレッシュの実行時は、ネットワーク接続を維持する必要があります。リフレッシュ実行時にネットワーク接続が切断または妨害されると、データベースの整合性を保つためにすべての変更がロールバックされます。したがって、ネットワーク接続を維持することが困難な場合は、リフレッシュ・グループの規模を小さくします。

NULLリフレッシュも最適化されます。つまり、特定のマテリアライズド・ビューの最後のリフレッシュ以降にマスター表が変更されなかった場合、リフレッシュ時にマテリアライズド・ビューのための追加の時間はほとんどかかりません。

親トピック: 編成メカニズム

37.4 リフレッシュ処理

マテリアライズド・ビューとマスター表との一貫性を確保するには、マテリアライズド・ビューを定期的にリフレッシュする必要があります。

- リフレッシュ処理について
  マテリアライズド・ビューのリフレッシュは、最新のマスター表の状態をマテリアライズド・ビューに反映させる効果的なバッチ操作です。
- リフレッシュ・タイプ
  Oracleデータベースでは、高速リフレッシュ、完全リフレッシュまたは強制リフレッシュのいずれかを使用して、マテリアライズド・ビューをリフレッシュできます。
- リフレッシュの開始
  リフレッシュ・グループを作成するときに、スケジュールした間隔でグループのマテリアライズド・ビューが自動的にリフレッシュされるようにグループを構成できます。逆に、リフレッシュ・グループを手動または必要に応じてリフレッシュするように、スケジュール情報を省略することもできます。ネットワークに常時接続されていないシステムでリフレッシュを実行する場合は、手動リフレッシュが理想的なソリューションです。
- 制約およびリフレッシュ
  マテリアライズド・ビューのリフレッシュ中の整合性制約違反を回避するために、各マテリアライズド・ビューの主キー以外の整合性制約を遅延可能にします。

親トピック: 読取り専用マテリアライズド・ビューのアーキテクチャ

37.4.1 リフレッシュ処理について

マテリアライズド・ビューのリフレッシュは、最新のマスター表の状態をマテリアライズド・ビューに反映させる効果的なバッチ操作です。

マテリアライズド・ビューのデータは、現在のマスター表のデータといつも一致しているとはかぎりません。マテリアライズド・ビューは、ある一時点(作成時またはリフレッシュ時)に存在していたデータとして、トランザクション(読取り)一貫性を保ってマスター表を反映したものです。マテリアライズド・ビューのデータをマスター表のデータとほぼ一致させるには、マテリアライズド・ビューを定期的にリフレッシュする必要があります。

マスター表の1行は、マテリアライズド・ビューのリフレッシュとリフレッシュの間に何度も更新される場合がありますが、リフレッシュでは、その行はマテリアライズド・ビュー内で現行のデータを使用して1回のみ更新されます。たとえば、マテリアライズド・ビューの前回のリフレッシュ以降に、マスター表の1行が10回更新されても、次回のリフレッシュではマテリアライズド・ビュー内の対応する行は1度のみ更新されます。

より新しいマスター・データを反映するために、各マテリアライズド・ビューのリフレッシュ間隔およびリフレッシュ方法を決定します。たとえば、アプリケーションで頻繁に更新されるマスター表に基づいたマテリアライズド・ビューは、頻繁にリフレッシュする必要があります。これに対して、比較的静的なマスター表に基づいたマテリアライズド・ビューは、あまり頻繁にリフレッシュする必要はありません。つまり、アプリケーションの特性および要件を分析して、マテリアライズド・ビューの適切なリフレッシュ間隔を判断します。

マテリアライズド・ビューのリフレッシュのために、Oracleデータベースではいくつかのリフレッシュ・タイプとリフレッシュ開始方法がサポートされています。

親トピック: リフレッシュ処理

37.4.2 リフレッシュ・タイプ

Oracleデータベースでは、高速リフレッシュ、完全リフレッシュまたは強制リフレッシュのいずれかを使用して、マテリアライズド・ビューをリフレッシュできます。

- 完全リフレッシュ
  マテリアライズド・ビューの**完全リフレッシュ**を実行するには、マテリアライズド・ビューを管理するサーバーがマテリアライズド・ビューの定義問合せを実行します(これは基本的に、マテリアライズド・ビューを再作成する処理です)。
- 高速リフレッシュ
  高速リフレッシュを実行するときは、まずマテリアライズド・ビューを管理するマスター表が、マテリアライズド・ビューの最後のリフレッシュ以降にマスター表で発生した変更を識別し、次にそれらの変更をマテリアライズド・ビューに適用します。
- 強制リフレッシュ
  マテリアライズド・ビューの**強制リフレッシュ**を実行するために、マテリアライズド・ビューを管理するサーバーは高速リフレッシュの実行を試みます。高速リフレッシュが実行できない場合は、完全リフレッシュが実行されます。

親トピック: リフレッシュ処理

37.4.2.1 完全リフレッシュ

マテリアライズド・ビューの**完全リフレッシュ**を実行するには、マテリアライズド・ビューを管理するサーバーがマテリアライズド・ビューの定義問合せを実行します(これは基本的に、マテリアライズド・ビューを再作成する処理です)。

マテリアライズド・ビューをリフレッシュするために、問合せの結果セットが現在のマテリアライズド・ビュー・データを置き換えます。Oracleデータベースでは、任意のマテリアライズド・ビューの完全リフレッシュを実行できます。定義問合せを満たすデータの量によっては、完全リフレッシュの実行に高速リフレッシュの場合よりもかなり長い時間がかかる場合があります。

ノート:

マテリアライズド・ビューで完全リフレッシュを実行する場合は、効率を最大化するために、PCTFREE を 0 に、PCTUSED を 99 に設定します。

親トピック: リフレッシュ・タイプ

37.4.2.2 高速リフレッシュ

高速リフレッシュを実行するときは、まずマテリアライズド・ビューを管理するマスター表が、マテリアライズド・ビューの最後のリフレッシュ以降にマスター表で発生した変更を識別し、次にそれらの変更をマテリアライズド・ビューに適用します。

高速リフレッシュは、関係する各サーバーおよびネットワークでレプリケートするデータが少なくて済むため、マスター表」に加えられた変更が少ないときは完全リフレッシュよりも効率的です。マテリアライズド・ビューの高速リフレッシュは、マスター表にマテリアライズド・ビュー・ログがある場合にのみ可能です。また、高速リフレッシュを完全リフレッシュより高速にするには、CREATE MATERIALIZED VIEW文の各結合列に対して索引が必要です。

SQL*Loaderを使用してマスター表でダイレクト・パス・ロードを行った後、高速リフレッシュではダイレクト・パス・ロード中に発生した変更は適用されません。また、高速リフレッシュでは、マスター表へのその他のバルク・ロード操作によって生じた変更も適用されません。このような操作の例としては、APPENDヒントを指定したINSERT文と、INSERT ... SELECT * FROM文などがあります。

パーティション化されたマスター表に基づくマテリアライズド・ビューがある場合、パーティション・チェンジ・トラッキング(PCT)を使用

して、特定のパーティションに対応しているマテリアライズド・ビューの行を識別できます。また、PCTは、マテリアライズド・ビューのマスター表のパーティション・メンテナンス操作後、高速リフレッシュをサポートするために使用されます。マテリアライズド・ビューでのPCTに基づくリフレッシュは、様々な条件が満たされた場合のみ可能になります。

関連項目:

PCTおよびPCTに基づいたリフレッシュの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

親トピック: リフレッシュ・タイプ

37.4.2.3 強制リフレッシュ

マテリアライズド・ビューの**強制リフレッシュ**を実行するために、マテリアライズド・ビューを管理するサーバーは高速リフレッシュの実行を試みます。高速リフレッシュが実行できない場合は、完全リフレッシュが実行されます。

高速リフレッシュが実行できない場合にマテリアライズド・ビューのリフレッシュを実行するには、強制リフレッシュの設定を使用します。

親トピック: リフレッシュ・タイプ

37.4.3 リフレッシュの開始

リフレッシュ・グループを作成するときに、スケジュールした間隔でグループのマテリアライズド・ビューが自動的にリフレッシュされるようにグループを構成できます。逆に、リフレッシュ・グループを手動または必要に応じてリフレッシュするように、スケジュール情報を省略することもできます。ネットワークに常時接続されていないシステムでリフレッシュを実行する場合は、手動リフレッシュが理想的なソリューションです。

- スケジュールしたリフレッシュ
  自動リフレッシュのためのリフレッシュ・グループを作成するには、作成プロセスで、スケジュールしたリフレッシュ間隔をグループに指定する必要があります。
- 必要に応じたリフレッシュ
  必要に応じたリフレッシュは、明示的なプロシージャ・コールによりマテリアライズド・ビューをリフレッシュすることを意味します。

親トピック: リフレッシュ処理

37.4.3.1 スケジュールしたリフレッシュ

自動リフレッシュのためのリフレッシュ・グループを作成するには、作成プロセスで、スケジュールしたリフレッシュ間隔をグループに指定する必要があります。

グループのリフレッシュ間隔を設定するときは、次の特性を考慮してください。

- リフレッシュ間隔を指定する日付または日付式の値は、将来の特定の時点である必要があります。
- リフレッシュ間隔は、リフレッシュの実行に必要な時間よりも長くする必要があります。
- 相対日付式の値は、最新のリフレッシュ日付から計算される特定の時点である必要があります。ネットワークまたはシステムの障害により、グループのリフレッシュがスケジュールどおりに実行されない場合、相対日付式の結果は状況に応じて変更されることがあります。
- 明示的日付式の値は、最新のリフレッシュ日付に関係なく特定の時点である必要があります。

- 古いデータに対する環境の許容度を考慮してください。許容度が低い場合はリフレッシュ間隔を小さくし、許容度が高い場合はリフレッシュ間隔を大きくします。

リフレッシュ間隔の指定に使用できる単純な日付式の例を、次に示します。

- 1時間間隔の指定:

```
SYSDATE + 1/24
```

- 7日間隔の指定:

```
SYSDATE + 7
```

関連項目:

日付算術の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: リフレッシュの開始

37.4.3.2 必要に応じたリフレッシュ

必要に応じたリフレッシュは、明示的なプロシージャ・コールによりマテリアライズド・ビューをリフレッシュすることを意味します。

環境や状況によっては、スケジュールしたマテリアライズド・ビュー・リフレッシュの実行が適切でない場合もあります。たとえば、マスター表にバルク・データをロードした直後は、依存マテリアライズド・ビューにマスター表のデータは反映されていません。

マテリアライズド・ビューが非接続環境のラップトップの営業支援システムと統合されている場合は、マテリアライズド・ビューを必要に応じてリフレッシュすることもできます。営業支援ソフトウェアを設計する開発者がアプリケーション・コントロール(ボタンなど)を作成し、営業担当員は、1日の受注をサーバーに転送するときに、ネットワーク接続を確立した後、アプリケーション・コントロールを使用してマテリアライズド・ビューをリフレッシュできます。

hr_refgリフレッシュ・グループを必要に応じてリフレッシュする例を次に示します。

```
EXECUTE DBMS_REFRESH.REFRESH('hr_refg');
```

ノート:

レプリケーション環境で使用するマテリアライズド・ビューは、DBMS_MVIEW.REFRESH_ALL_MVIEWS または DBMS_MVIEW.REFRESH_DEPENDENT プロシージャを使用してリフレッシュしないでください。レプリケーション環境のマテリアライズド・ビューは、DBMS_REFRESH.REFRESH または DBMS_MVIEW.REFRESH プロシージャを使用してリフレッシュします。

親トピック: リフレッシュの開始

37.4.4 制約およびリフレッシュ

マテリアライズド・ビューのリフレッシュ中の整合性制約違反を回避するために、各マテリアライズド・ビューの主キー以外の整合性制約を遅延可能にします。

これには、NOT NULL制約を持つLOB列が必要です。また、外部キー制約によって関連付けられているすべてのマテリアライズド・ビューは、まとめてリフレッシュするか、同じリフレッシュ・グループでリフレッシュする必要があります。

ノート:

マテリアライズド・ビューの主キー制約は、遅延可能である場合とない場合があります。

関連項目:

制約を遅延可能にする方法の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: リフレッシュ処理

38 読取り専用マテリアライズド・ビューの計画

読取り専用マテリアライズド・ビューの計画を開始する前に、マテリアライズド・ビューに関連する概念およびアーキテクチャを理解するのは重要なことです。読取り専用マテリアライズド・ビューの概念およびアーキテクチャを理解した後、読取り専用マテリアライズド・ビュー環境の計画に関する重要な検討事項があります。

- マスター表に関する考慮事項
  マスター表では、主キー、外部キーおよびデータ型を考慮する必要があります。
- マスター・データベースとマテリアライズド・ビュー・データベースの計画
  読取り専用マテリアライズド・ビュー環境のデータベース計画には、マテリアライズド・ビューの準備およびマテリアライズド・ビュー・ログの構成が含まれます。

親トピック: 読取り専用マテリアライズド・ビューの管理

38.1 マスター表に関する考慮事項

マスター表では、主キー、外部キーおよびデータ型を考慮する必要があります。

- 主キーとマスター表
  可能であれば、各マスター表が主キーを持つ必要があります。
- 外部キーとマスター表
  外部キー参照制約を持つ表をレプリケートする場合は、親表で更新および削除が許可されていない場合を除き、常に外部キー列に索引を付けて、これらの索引をレプリケートすることをお薦めします。索引は自動でレプリケートされません。
- マスター表のデータ型に関する考慮事項
  データ型とマスター表に関するいくつかの考慮事項があります。
- サポートされていない表タイプ
  いくつかの表タイプは、マテリアライズド・ビューのベースとして使用できません。

親トピック: 読取り専用マテリアライズド・ビューの計画

38.1.1 主キーとマスター表

可能であれば、各マスター表が主キーを持つ必要があります。

主キーを設定できない場合、各マスター表に、表の各行の一意の識別子として使用できる列のセットが含まれている必要があります。レプリケーション環境で使用する表に主キーまたは一意の列のセットがない場合は、レプリケート表を必要に応じて修正します。さらに、マスター表に基づいて主キー・マテリアライズド・ビューを作成する場合は、そのマスターに主キーを設定する必要があります。

親トピック: マスター表に関する考慮事項

38.1.2 外部キーとマスター表

親表の更新や削除が許可されていない場合を除き、外部キー参照制約がある表のレプリケーションでは、常に外部キーに索引を作成してからこれらの索引をレプリケートすることをお薦めします。索引は自動でレプリケートされません。

親トピック: マスター表に関する考慮事項

38.1.3 マスター表のデータ型に関する考慮事項

データ型とマスター表に関するいくつかの考慮事項があります。

次のデータ型を使用する列を含むマスター表に基づいて読取り専用マテリアライズド・ビューを作成できます。

- VARCHAR2
- NVARCHAR2
- NUMBER
- DATE
- TIMESTAMP
- TIME WITH TIME ZONE
- TIMESTAMP LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- RAW
- ROWID
- CHAR
- NCHAR
- BASICFILE記憶域を持つCLOB
- BASICFILE記憶域を持つNCLOB
- BASICFILE記憶域を持つBLOB
- CLOBとして格納されたXMLType
- 型継承または型進化を使用しないユーザー定義型
- 型継承または型進化を使用しないOracle提供の型

ノート:

XMLType を CLOB として格納することは非推奨です。

マテリアライズド・ビューの定義問合せのWHERE句でLOB列を参照することはできません。

ユーザー定義型(列オブジェクト、オブジェクト表、REF、VARRAY、ネストした表など)を使用するマテリアライズド・ビューを作成できます。

次のデータ型を使用する列を含むマスター表に基づいてマテリアライズド・ビューを作成することはできません。

- FLOAT
- BINARY_FLOAT
- BINARY_DOUBLE
- LONG
- LONG RAW
- SECUREFILE記憶域を持つCLOB

- SECUREFILE記憶域を持つNCLOB
- SECUREFILE記憶域を持つBLOB
- BFILE
- オブジェクト・リレーショナルまたはバイナリXMLとして格納されているXMLType
- Expression型
- 型継承または型進化を使用するユーザー定義型
- 型継承または型進化を使用するOracle提供の型

LONGデータ型は、BASICFILE記憶域を持つLOBに変換する必要があります。

関連項目:

データ型の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

親トピック: マスター表に関する考慮事項

38.1.4 サポートされていない表タイプ

いくつかの表タイプは、マテリアライズド・ビューのベースとして使用できません。

これらの表タイプに基づいてマテリアライズド・ビューを作成することはできません:

- 表圧縮機能によって圧縮された表
- 仮想列を含む表
- 一時表
- フラッシュバック・データ・アーカイブの表

親トピック: マスター表に関する考慮事項

38.2 マスター・データベースとマテリアライズド・ビュー・データベースの計画

読取り専用マテリアライズド・ビュー環境のデータベース計画には、マテリアライズド・ビューの準備およびマテリアライズド・ビュー・ログの構成が含まれます。

- マスター・データベースとマテリアライズド・ビュー・データベースの特性
  レプリケーション環境を計画するときは、レプリケーション環境に含める各データベースを、マスター・データベースまたはマテリアライズド・ビュー・データベースにするかどうかを決定する必要があります。
- マスター・データベースの利点
  マスター・データベースには、いくつかの利点があります。
- マテリアライズド・ビュー・データベースの利点
  マテリアライズド・ビュー・データベースには、いくつかの利点があります。
- マテリアライズド・ビューの準備
  マテリアライズド・ビュー・レプリケーションに関する問題の多くは、環境の準備が正しく行われていないことによるものです。
- マテリアライズド・ビュー・ログの作成
  マスター表に基づいたマテリアライズド・ビューの高速リフレッシュを可能にするために、マスター表にマテリアライズド・

ビュー・ログを作成します。

- マテリアライズド・ビュー・ログの列のロギング
  マテリアライズド・ビュー・ログを作成するときに、マテリアライズド・ビューの高速リフレッシュを可能にするために、ログに列を追加できます。

親トピック: 読取り専用マテリアライズド・ビューの計画

38.2.1 マスター・データベースおよびマテリアライズド・ビュー・データベースの特性

レプリケーション環境を計画するときは、レプリケーション環境に含める各データベースを、マスター・データベースまたはマテリアライズド・ビュー・データベースにするかどうかを決定する必要があります。

環境内の特定のデータベースをマスター・データベースまたはマテリアライズド・ビュー・データベースにするかどうかを決定するときは、両方のデータベース・タイプの特性と利点を検討してください。マスター・データベースとマテリアライズド・ビュー・データベースの両方を1つのレプリケーション環境でサポートできます。

表38-1 マスター・データベースおよびマテリアライズド・ビュー・データベースの特性

マスター・データベース	マテリアライズド・ビュー・データベース
多数のマテリアライズド・ビュー・データベースとの通信の可能性	1 つのマスター・データベースとの通信
大量のデータを含む	マスター・データベースのデータのサブセットであることが可能な少量のデータを含む

親トピック: マスター・データベースとマテリアライズド・ビュー・データベースの計画

38.2.2 マスター・データベースの利点

マスター・データベースには、いくつかの利点があります。

マスター・データベースには、次の利点があります。

- リモート・データベースによる可用性の高いデータのサポート
- データ操作言語(DML)の変更へのサポートを提供
- フェイルオーバーによる保護を提供できます。

親トピック: マスター・データベースとマテリアライズド・ビュー・データベースの計画

38.2.3 マテリアライズド・ビュー・データベースの利点

マテリアライズド・ビュー・データベースには、いくつかの利点があります。

マテリアライズド・ビュー・データベースには、次の利点があります。

- モバイル・コンピューティングのサポート
- マスター・データベースのデータのサブセットを含めることが可能

親トピック: マスター・データベースとマテリアライズド・ビュー・データベースの計画

38.2.4 マテリアライズド・ビューの準備

マテリアライズド・ビュー・レプリケーションに関する問題の多くは、環境の準備が正しく行われていないことによるものです。

マテリアライズド・ビュー環境を作成する前に、次の前提条件が満たされていることを確認します。

- 必要なスキーマが存在することを確認します。
- 必要なデータベース・リンクが存在することを確認します。
- 必要な権限が付与されていることを確認します。
- 十分なジョブ・プロセスが存在することを確認します。

- マテリアライズド・ビュー・データベースに必要なスキーマ
  リモート・データベースのマテリアライズド・ビューを含むスキーマは、マスター・データベースのマスター表を含むスキーマに対応している必要があります。
- マテリアライズド・ビューに必要なデータベース・リンク
  マテリアライズド・ビューの定義問合せは、リモート表のデータを参照するために1つ以上のデータベース・リンクを使用することができます。
- 必要な権限
  マテリアライズド・ビューの作成者と所有者は、ともにマテリアライズド・ビューの定義SELECT文を発行できる必要があります。
- 十分なジョブ・プロセス
  レプリケーション環境を自動化できる十分なジョブ・プロセスが割り当てられていることが重要です。ジョブ・プロセスは、自動的にマテリアライズド・ビューをリフレッシュします。

親トピック: マスター・データベースとマテリアライズド・ビュー・データベースの計画

38.2.4.1 マテリアライズド・ビュー・データベースに必要なスキーマ

リモート・データベースのマテリアライズド・ビューを含むスキーマは、マスター・データベースのマスター表を含むスキーマに対応している必要があります。

したがって、マテリアライズド・ビューを使用してレプリケートするマスター表を含むスキーマを識別してください。マスター・データベースのターゲット・スキーマを識別した後、リモート・データベースで、対応するアカウントを同じ名前で作成します。たとえば、すべてのマスター表が`ny.example.com`データベースの`sales`スキーマ内にある場合は、対応する`sales`スキーマをマテリアライズド・ビュー・データベース`sf.example.com`に作成します。

関連項目:

「マテリアライズド・ビューの操作に必要な権限」

親トピック: マテリアライズド・ビューの準備

38.2.4.2 マテリアライズド・ビューに必要なデータベース・リンク

マテリアライズド・ビューの定義問合せでは、1つ以上のデータベース・リンクを使用してリモート表のデータを参照します。

マテリアライズド・ビューを作成するには、使用する予定のデータベース・リンクが使用可能になっている必要があります。さらに、リモート・データベースにアクセスするためにデータベース・リンクで使用するアカウントには、セキュリティ・コンテキストが定義され、そのコンテキストに基づいてマテリアライズド・ビューが作成され、リフレッシュされます。

正常な動作を保証するために、マテリアライズド・ビューの定義問合せでは、ユーザー名とパスワードが定義に埋め込まれているデータベース・リンクを使用する必要があります(マテリアライズド・ビューを作成するときは、パブリック・データベース・リンクを使用できません)。ユーザー名とパスワードが埋め込まれているデータベース・リンクは常に、指定されたアカウントを使用してリモート・データベースとの接続を確立します。また、リンクで使用するリモート・アカウントには、マテリアライズド・ビューの定義問合せの中で参照されるデータにアクセスするために必要なSELECT権限が必要です。

マテリアライズド・ビューを作成するには、いくつかの管理データベース・リンクを作成する必要があります。具体的には、マテリアライズド・ビュー・データベースからマスター・データベースへのPUBLICデータベース・リンクを作成する必要があります。これを作成すると、各プライベート・データベース・リンク内にUSING句を含める必要がなくなるため、プライベート・データベース・リンクを定義しやすくなります。

たとえば、次の文は、マテリアライズド・ビュー・データベースからマスター・データベースへのパブリック・データベース・リンクを作成します。

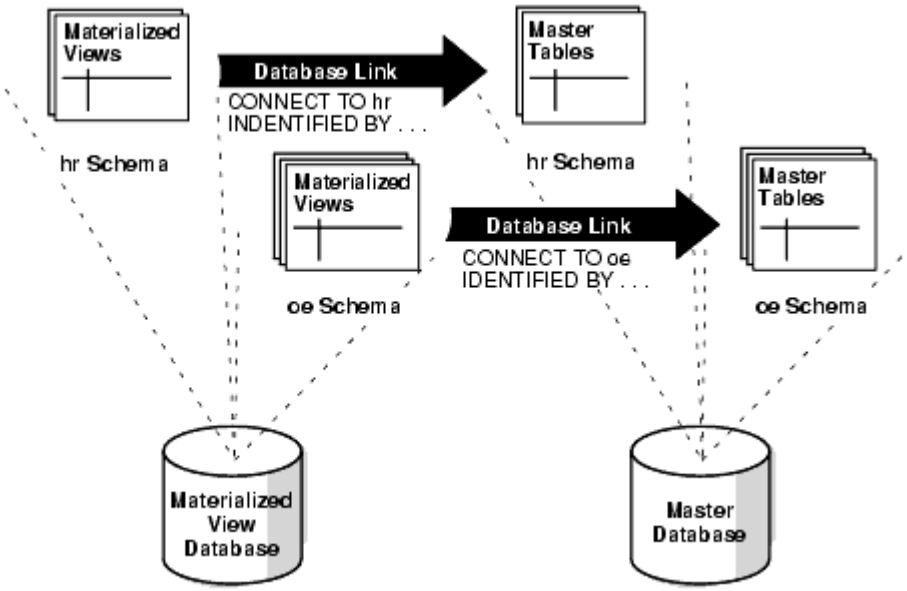
```
CREATE PUBLIC DATABASE LINK orc1.example.com USING 'orc1.example.com';
```

管理データベース・リンクを作成した後、マテリアライズド・ビュー・データベースの各レプリケート・マテリアライズド・ビュー・スキーマをマスター・データベースの対応するスキーマに接続するプライベート・データベース・リンクを作成する必要があります。マテリアライズド・ビュー・データベースの各プライベート・データベース・リンクには、対応付けられたマスター・データベースのアカウント情報を埋め込むことが必要です。たとえば、マテリアライズド・ビュー・データベースのhrスキーマには、hrというユーザー名とパスワードを使用してマスター・データベースに接続するプライベート・データベース・リンクが必要です。

たとえば、次の文は、マテリアライズド・ビュー・データベースからマスター・データベースへのプライベート・データベース・リンクを作成します。

```
CREATE DATABASE LINK orc1.example.com
   CONNECT TO myuser IDENTIFIED BY password;
```

図38-1 推奨されるスキーマおよびデータベース・リンク構成



マテリアライズド・ビューの定義問合せは、仮想プライベート・データベース(VPD)によって変更できません。VPDは、マテリアライズド・ビューの作成とリフレッシュの両方を実行するスキーマに対してNULLポリシーを戻す必要があります。USING TRUSTED CONSTRAINTS句を指定した場合は、NULL以外のVPDポリシーを持つマテリアライズド・ビューを作成できます。この場合、マテリアライズド・ビューが正しく動作することを確認します。マテリアライズド・ビューの結果は、VPDポリシーによってフィルタ処理された行と列に基づいて計算されます。したがって、正しい結果を得るには、マテリアライズド・ビュー定義をVPDポリシーで調整す

る必要があります。

関連項目:

- データベース・リンクの詳細は、「分散データベースの概念」
- VPDの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- Oracle Label Securityの詳細は、『Oracle Label Security管理者ガイド』

親トピック: マテリアライズド・ビューの準備

38.2.4.3 必要な権限

マテリアライズド・ビューの作成者と所有者は、ともにマテリアライズド・ビューの定義SELECT文を発行できる必要があります。

マテリアライズド・ビューの所有者とは、マテリアライズド・ビューが含まれているスキーマのことです。

関連項目:

「マテリアライズド・ビューの操作に必要な権限」

親トピック: マテリアライズド・ビューの準備

38.2.4.4 十分なジョブ・プロセス

レプリケーション環境を自動化できる十分なジョブ・プロセスが割り当てられていることが重要です。ジョブ・プロセスは、自動的にマテリアライズド・ビューをリフレッシュします。

マテリアライズド・ビュー・レプリケーションの特性として、マテリアライズド・ビュー・データベースには、通常、マスター・データベースへのスケジュール・リンクが1つあり、少なくとも1つのジョブ・プロセスが必要です。マテリアライズド・ビュー・データベースで必要なジョブ・プロセスの数は、パージ・スケジュール、ユーザー定義ジョブおよびスケジュール・リンクに依存しますが、通常は1つから3つを必要とします。また、それぞれの並列度に対して少なくとも1つのジョブ・プロセスが必要です。

ユーザーがアプリケーション・インタフェースを使用してマテリアライズド・ビューを手動でリフレッシュする場合は、スケジュール・リンクを作成する必要がなくなり、マテリアライズド・ビュー・データベースで必要なジョブ・プロセスが1つ減ります。

ジョブ・プロセスは、JOB_QUEUE_PROCESSES初期化パラメータを使用して定義します。この初期化パラメータは、変更可能です。そのため、インスタンス実行中に変更できます。Oracleデータベースにより、ジョブ・プロセスの間隔が自動的に決定されます。つまり、Oracleデータベースが、ジョブを実行するためにジョブ・プロセスを「起動する」タイミングを決定します。

関連項目:

Oracle Databaseリファレンス

親トピック: マテリアライズド・ビューの準備

38.2.5 マテリアライズド・ビュー・ログの作成

マスター表に基づいたマテリアライズド・ビューの高速リフレッシュを可能にするために、マスター表にマテリアライズド・ビュー・ログを作成します。

リモート・マテリアライズド・ビュー・データベースのマテリアライズド・ビューを作成する前に、マスター・データベースに必要なマテリアライズド・ビュー・ログを作成する必要があります。マテリアライズド・ビュー・ログは、高速リフレッシュ機能付きのマテリアライズド・ビューを1つでもサポートするマスター表に必要です。

マテリアライズド・ビュー・ログを作成するには、次の権限が必要です。

- CREATE ANY TABLE
- CREATE ANY TRIGGER
- SELECT(マテリアライズド・ビュー・ログのマスター表)
- COMMENT ANY TABLE

マテリアライズド・ビューを作成するには:

1. 目的の表にマテリアライズド・ビュー・ログを作成するために、必要な権限を持つユーザーとしてマスター・データベースに接続します。
2. CREATE MATERIALIZED VIEW LOG文を実行します。

   オブジェクト表に基づいてマテリアライズド・ビューを作成するときは、WITH OBJECT ID句を指定してオブジェクト識別子をロギングする必要がありますが、オブジェクト識別子が主キー・ベースの場合は、主キーもロギングされるように指定できます。

例38-1 マテリアライズド・ビュー・ログの作成

```
CREATE MATERIALIZED VIEW LOG ON hr.employees;
```

例38-2 オブジェクト表のマテリアライズド・ビュー・ログの作成

次のSQL文は、categories_typユーザー定義型を作成します。

```
CREATE TYPE oe.category_typ AS OBJECT
   (category_name           VARCHAR2(50),
    category_description    VARCHAR2(1000),
    category_id             NUMBER(2));
/
```

この型に基づいてオブジェクト表を作成するとき、オブジェクト識別子がシステム生成のものか主キーに基づいたものかを指定できます。

```
CREATE TABLE oe.categories_tab_sys OF oe.category_typ
    (category_id    PRIMARY KEY)
    OBJECT ID SYSTEM GENERATED;
CREATE TABLE oe.categories_tab_pkbased OF oe.category_typ
    (category_id    PRIMARY KEY)
    OBJECT ID PRIMARY KEY;
```

たとえば、次の文ではcategories_tab_sysオブジェクト表のマテリアライズド・ビュー・ログを作成し、オブジェクト識別子列をロギングするように指定します。

```
CREATE MATERIALIZED VIEW LOG ON oe.categories_tab_sys
   WITH OBJECT ID;
```

次の文では、categories_tab_pkbasedオブジェクト表のマテリアライズド・ビュー・ログを作成し、オブジェクト識別子列とともに主キー列をロギングするように指定します。

```
CREATE MATERIALIZED VIEW LOG ON oe.categories_tab_pkbased
   WITH OBJECT ID, PRIMARY KEY;
```

関連項目:

- 「マテリアライズド・ビュー・ログ」
- Oracle Database SQL言語リファレンス

親トピック: マスター・データベースとマテリアライズド・ビュー・データベースの計画

38.2.6 マテリアライズド・ビュー・ログの列のロギング

マテリアライズド・ビュー・ログを作成するときに、マテリアライズド・ビューの高速リフレッシュを可能にするために、ログに列を追加できます。

1. 目的の表にマテリアライズド・ビュー・ログを作成または変更するために、必要な権限を持つユーザーとしてマスター・データベースに接続します。
2. 次のいずれかを行います:
   - `CREATE MATERIALIZED VIEW LOG`文を実行し、ログする列を指定します。
   - `ALTER MATERIALIZED VIEW LOG`文を実行し、ログする列を指定します。

例38-3 マテリアライズド・ビューの作成時の列のロギング

`orders.customer_id`および`orders.order_total`列が追加された`oe.orders`表に対するマテリアライズド・ビュー・ログを作成するには、次の文を発行します。

```
CREATE MATERIALIZED VIEW LOG ON oe.orders
  WITH PRIMARY KEY (customer_id,order_total);
```

例38-4 既存マテリアライズド・ビューの列のロギング

次の文を発行することで、`oe.orders`表に対するマテリアライズド・ビュー・ログに`orders.customer_id`および`orders.order_total`列を追加できます。

```
ALTER MATERIALIZED VIEW LOG ON oe.orders ADD (customer_id,order_total);
```

例38-5 列オブジェクトの属性のログ

ユーザー定義データ型を使用している場合は、列オブジェクトの属性をマテリアライズド・ビュー・ログにロギングできます。たとえば、`oe.customers`表に`cust_address.postal_code`属性がある場合、次の文を発行すると、この属性をマテリアライズド・ビュー・ログにロギングできます。

```
ALTER MATERIALIZED VIEW LOG ON oe.customers ADD (cust_address.postal_code);
```

関連項目:

- 「マテリアライズド・ビュー・ログにログされる列」
- Oracle Database SQL言語リファレンス

親トピック: マスター・データベースとマテリアライズド・ビュー・データベースの計画

39 読取り専用マテリアライズド・ビューの作成および管理

読取り専用マテリアライズド・ビューおよびリフレッシュ・グループを作成および管理できます。また、マテリアライズド・ビューをリフレッシュすることもできます。

- 読取り専用マテリアライズド・ビューの作成
  マテリアライズド・ビュー・データベースのマスター表のデータをレプリケートするには、読取り専用マテリアライズド・ビューを作成します。
- リフレッシュ・グループの作成
  リフレッシュ・グループ内の関連のあるマテリアライズド・ビュー間のトランザクション一貫性を確保するために、リフレッシュ・グループにマテリアライズド・ビューを追加します。
- マテリアライズド・ビューのリフレッシュ
  マテリアライズド・ビューのリフレッシュにより、マテリアライズド・ビューのマスターのデータとマテリアライズド・ビューのデータが同期化されます。
- マテリアライズド・ビューの高速リフレッシュ機能に関する判断
  REFRESH FAST句を使用してマテリアライズド・ビューの作成を試みるか、DBMS_MVIEW.EXPLAIN_MVIEWプロシージャを使用することにより、マテリアライズド・ビューが高速リフレッシュ可能かどうかを判断できます。
- 新しいマテリアライズド・ビュー・データベースの追加
  1つ以上のマテリアライズド・ビュー・データベースを含むマテリアライズド・ビュー環境を作成した後、新しいマテリアライズド・ビュー・データベースの追加が必要な場合があります。
- マテリアライズド・ビュー・ログの監視
  マスター・データベースのマテリアライズド・ビュー・ログの情報を表示するために、問合せを実行できます。
- マテリアライズド・ビューの監視
  マテリアライズド・ビューおよびリフレッシュ・グループの情報を表示するために、問合せを実行できます。

親トピック: 読取り専用マテリアライズド・ビューの管理

39.1 読取り専用マテリアライズド・ビューの作成

マテリアライズド・ビュー・データベースのマスター表のデータをレプリケートするには、読取り専用マテリアライズド・ビューを作成します。

マスター・データベースとマテリアライズド・ビュー・データベース間でデータをレプリケートするためのマテリアライズド・ビューを作成する前に、使用する予定のデータベース・リンクが使用可能である必要があります。

1. マテリアライズド・ビューを作成するための必要な権限を持つユーザーとしてデータベースに接続します。
2. CREATE MATERIALIZED VIEW文を実行します。

例39-1 主キー・マテリアライズド・ビューの作成

```
CREATE MATERIALIZED VIEW hr.employees_mv1 WITH PRIMARY KEY
  AS SELECT * FROM hr.employees@orc1.example.com;
```

例39-2 ROWIDマテリアライズド・ビューの作成

```
CREATE MATERIALIZED VIEW oe.orders REFRESH WITH ROWID AS
  SELECT * FROM oe.orders@orc1.example.com;
```

例39-3 オブジェクト・マテリアライズド・ビューの作成

必要な型をマテリアライズド・ビュー・データベースに作成した後、OF type句を指定してオブジェクト・マテリアライズド・ビューを作成できます。

たとえば、次のSQL文でorc1.example.comマスター・データベースにoe.categories_tabオブジェクト表を作成するとします。

```
CREATE TYPE oe.category_typ AS OBJECT
   (category_name           VARCHAR2(50),
    category_description    VARCHAR2(1000),
    category_id             NUMBER(2));
/
CREATE TABLE oe.categories_tab OF oe.category_typ
    (category_id    PRIMARY KEY);
```

oe.categories_tabマスター表に基づいて高速リフレッシュが可能なマテリアライズド・ビューを作成するには、この表のマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON oe.categories_tab WITH OBJECT ID;
```

オブジェクト表に対するマテリアライズド・ビュー・ログを作成する場合には、WITH OBJECT ID句が必要です。

マスター・データベースの同じ型と同じオブジェクト識別子を持つoe.category_typ型をマテリアライズド・ビュー・データベースに作成した後、次のSQL文のようにOF type句を使用して、oe.categories_tabオブジェクト表に基づくオブジェクト・マテリアライズド・ビューを作成できます。

```
CREATE MATERIALIZED VIEW oe.categories_objmv OF oe.category_typ
   REFRESH FAST
   AS SELECT * FROM oe.categories_tab@orc1.example.com;
```

この場合、typeはoe.category_typです。

ノート:

型は、マテリアライズド・ビュー・データベースとマスター・データベースで同じである必要があります。詳細は、「レプリケーション・データベースでの型の一致」を参照してください。

関連項目:

- マテリアライズド・ビューを作成するいくつかの例については、「読取り専用マテリアライズド・ビューの概念」
- マテリアライズド・ビューを作成するために必要な権限については、「マテリアライズド・ビューの操作に必要な権限」
- 「マテリアライズド・ビューのために必要なデータベース・リンク」
- 「オブジェクト表に基づいたマテリアライズド・ビュー」
- Oracle Database SQL言語リファレンス

親トピック: 読取り専用マテリアライズド・ビューの作成および管理

39.2 リフレッシュ・グループの作成

リフレッシュ・グループ内の関連のあるマテリアライズド・ビュー間のトランザクション一貫性を確保するために、リフレッシュ・グループにマテリアライズド・ビューを追加します。

リフレッシュ・グループがリフレッシュされると、特定のリフレッシュ・グループに追加されているすべてのマテリアライズド・ビューが同時にリフレッシュされます。

1. リフレッシュ・グループを作成し、それにマテリアライズド・ビューを追加するために、必要な権限を持つ管理ユーザーとしてマテリアライズド・ビュー・データベースに接続します。
2. リフレッシュ・グループを作成するために、DBMS_REFRESH.MAKEプロシージャを実行します。
3. リフレッシュ・グループにマテリアライズド・ビューを追加するために、DBMS_REFRESH.ADDプロシージャを1回以上実行します。

例39-4 リフレッシュ・グループの作成

この例では、リフレッシュ・グループを作成し、それに2つのマテリアライズド・ビューを追加します。

```
BEGIN
   DBMS_REFRESH.MAKE (
      name => 'mviewadmin.hr_refg',
      list => '',
      next_date => SYSDATE,
      interval => 'SYSDATE + 1/24',
      implicit_destroy => FALSE,
      rollback_seg => '',
      push_deferred_rpc => TRUE,
      refresh_after_errors => FALSE);
END;
/
BEGIN
    DBMS_REFRESH.ADD (
      name => 'mviewadmin.hr_refg',
      list => 'hr.countries_mv1',
      lax => TRUE);
END;
/
BEGIN
    DBMS_REFRESH.ADD (
      name => 'mviewadmin.hr_refg',
      list => 'hr.departments_mv1',
      lax => TRUE);
END;
/
```

関連項目:

「リフレッシュ・グループ」

親トピック: 読取り専用マテリアライズド・ビューの作成および管理

39.3 マテリアライズド・ビューのリフレッシュ

マテリアライズド・ビューのリフレッシュにより、マテリアライズド・ビューのマスターのデータとマテリアライズド・ビューのデータが同期化されます。

リフレッシュ・グループのすべてのマテリアライズド・ビューを一度にリフレッシュすることも、マテリアライズド・ビューを個別にリフレッシュすることもできます。マテリアライズド・ビュー・データベースで複数のマテリアライズド・ビューに依存するアプリケーションがある場合、アプリケーションで使用されるすべてのマテリアライズド・ビューでデータのトランザクション一貫性を維持するために、リフレッシュ・グループの使用をお薦めします。

1. リフレッシュ・グループまたは個々のマテリアライズド・ビューをリフレッシュするために、必要な権限を持つユーザーとしてマ

テリアライズド・ビュー・データベースに接続します。

2. 次のいずれかを行います:
   - リフレッシュ・グループをリフレッシュするために、DBMS_REFRESH.REFRESHプロシージャを実行します。
   - 個々のマテリアライズド・ビューをリフレッシュするために、DBMS_MVIEW.REFRESH プロシージャを実行します。

例39-5 リフレッシュ・グループのリフレッシュ

次の例では、hr_refgリフレッシュ・グループがリフレッシュされます。

```
EXECUTE DBMS_REFRESH.REFRESH ('hr_refg');
```

例39-6 個々のマテリアライズド・ビューのリフレッシュ

次の例では、hr.departments_mvマテリアライズド・ビューがリフレッシュされます。

```
BEGIN
   DBMS_MVIEW.REFRESH (
     list   =>  'hr.departments_mv',
     method =>  '?');
END;
/
```

ノート:

マテリアライズド・ビューをリフレッシュするために DBMS_MVIEW.REFRESH_ALL_MVIEWS または DBMS_MVIEW.REFRESH_DEPENDENT プロシージャを使用しないでください。レプリケーション環境のマテリアライズド・ビューは、DBMS_REFRESH.REFRESH または DBMS_MVIEW.REFRESH プロシージャを使用してリフレッシュします。

関連項目:

- マテリアライズド・ビューを作成するために必要な権限については、「マテリアライズド・ビューの操作に必要な権限」
- DBMS_MVIEWパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』

親トピック: 読取り専用マテリアライズド・ビューの作成および管理

39.4 マテリアライズド・ビューの高速リフレッシュ機能に関する判断

REFRESH FAST句を使用してマテリアライズド・ビューの作成を試みるか、DBMS_MVIEW.EXPLAIN_MVIEWプロシージャを使用することにより、マテリアライズド・ビューが高速リフレッシュ可能かどうかを判断できます。

高速リフレッシュでは、マテリアライズド・ビュー・ログを使用して、最後のリフレッシュ以降に変更が加えられた行のみを更新します。マテリアライズド・ビューが高速リフレッシュ可能かどうかを判断するには、REFRESH FAST句を使用してマテリアライズド・ビューを作成します。マテリアライズド・ビューが副問合せマテリアライズド・ビューの制限に違反した場合、Oracleデータベースはエラーを返します。強制リフレッシュを指定した場合、Oracleデータベースでは高速リフレッシュが実行できなければ完全リフレッシュが自動的に実行されるため、エラーが返されない場合があります。

また、DBMS_MVIEWパッケージ内のEXPLAIN_MVIEWプロシージャを使用して、既存のマテリアライズド・ビュー、あるいはまだ作成されていないマテリアライズド・ビューに関して次の情報を判断できます。

- マテリアライズド・ビューの機能
- 各機能を使用できるかどうか
- 機能を使用できない場合にはできない理由

この情報は、VARRAYまたはMV_CAPABILITIES_TABLEに格納できます。この表に情報を格納する場合は、EXPLAIN_MVIEWプロシージャを実行する前に、Oracle_home/rdbms/adminディレクトリにあるutlxmv.sqlスクリプトを実行してこの表を作成する必要があります。

マテリアライズド・ビューの高速リフレッシュ機能を判断するには:

1. マテリアライズド・ビュー・データベースに管理ユーザーとして接続します。
2. 次のいずれかを行います:
   - REFRESH FAST句を使用してマテリアライズド・ビューを作成します。
   - DBMS_MVIEW.EXPLAIN_MVIEWプロシージャを実行します。

例39-7 FAST REFRESH句を使用したマテリアライズド・ビューの作成

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
 SELECT * FROM oe.orders@orc1.example.com o
 WHERE EXISTS
   (SELECT * FROM oe.customers@orc1.example.com c
    WHERE o.customer_id = c.customer_id AND c.credit_limit > 10000);
```

例39-8 既存マテリアライズド・ビューのリフレッシュ機能の判断

たとえば、oe.ordersマテリアライズド・ビューの機能を判断するには、次のように入力します。

```
EXECUTE DBMS_MVIEW.EXPLAIN_MVIEW ('oe.orders');
```

例39-9 まだ存在しないマテリアライズド・ビューのリフレッシュ機能の判断

または、マテリアライズド・ビューがまだ作成されていない場合は、その作成に使用する問合せを指定できます。

```
BEGIN
  DBMS_MVIEW.EXPLAIN_MVIEW ('SELECT * FROM oe.orders@orc1.example.com o
    WHERE EXISTS (SELECT * FROM oe.customers@orc1.example.com c
    WHERE o.customer_id = c.customer_id AND c.credit_limit > 500)');
END;
/
```

MV_CAPABILITIES_TABLEを問い合せて結果を表示します。

MV_CAPABILITIES_TABLEを問い合せて結果を表示します。

ノート:

MV_CAPABILITIES_TABLE は、事前作成されたコンテナ表に依存するマテリアライズド・ビューのリフレッシュ機能を示しません。たとえば、事前作成されたコンテナ表でのパーティション・メンテナンス操作の後に完全なリフレッシュが必要ですが、MV_CAPABILITIES_TABLE ではこの制限が示されません。

関連項目:

- 「副問合せを使用するマテリアライズド・ビューでの制限事項」

- 「マテリアライズド・ビュー・ログ」
- EXPLAIN_MVIEWプロシージャの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』

親トピック: 読取り専用マテリアライズド・ビューの作成および管理

39.5 新規マテリアライズド・ビュー・データベースの追加

1つ以上のマテリアライズド・ビュー・データベースを含むマテリアライズド・ビュー環境を作成した後、新しいマテリアライズド・ビュー・データベースの追加が必要な場合があります。

次の2つの条件がどちらも成立する場合は、新規マテリアライズド・ビュー・データベースに作成したマテリアライズド・ビューを高速リフレッシュしようとしたときに問題が発生することがあります。

- 新規マテリアライズド・ビュー・データベースのマテリアライズド・ビューと、他のマテリアライズド・ビュー・データベースの既存マテリアライズド・ビューが、同一のマスター表に基づいている。
- 新規マテリアライズド・ビュー・データベースに新規マテリアライズド・ビューを作成中に、既存のマテリアライズド・ビューがリフレッシュされる可能性がある。

問題が発生するのは、新規マテリアライズド・ビューが最初の高速リフレッシュを実行する前に、マスター表のマテリアライズド・ビュー・ログがパージされたときです。このようなときに新規マテリアライズド・ビュー・データベースでマテリアライズド・ビューを高速リフレッシュしようとすると、次のエラーが発生することがあります。

```
ORA-12004 REFRESH FAST cannot be used for materialized view materialized_view_name
ORA-12034 materialized view log on materialized_view_name younger than last refresh
```

これらのエラーを受け取った場合の解決策は、新規マテリアライズド・ビューの完全リフレッシュを実行することのみです。この問題を回避するには、本番マテリアライズド・ビューを作成する前に、新規マテリアライズド・ビュー・データベースにダミー・マテリアライズド・ビューを作成します。ダミーのマテリアライズド・ビューを作成することにより、本番のマテリアライズド・ビューを作成中にマテリアライズド・ビュー・ログがパージされなくなります。

マテリアライズド・ビュー・データベースにダミー・マテリアライズド・ビューを作成することを選択する場合は、次のステップを実行します。

1. マスター表に基づいてdummy_mviewというダミー・マテリアライズド・ビューを作成します。たとえば、salesというマスター表に基づいてダミー・マテリアライズド・ビューを作成するには、新規マテリアライズド・ビュー・データベースで次の文を発行します。

   ```
   CREATE MATERIALIZED VIEW dummy_mview REFRESH FAST AS
     SELECT * FROM pr.sales@orc1.example.com WHERE 1=0;
   ```

2. 新規マテリアライズド・ビュー・データベースに本番マテリアライズド・ビューを作成します。
3. 新規マテリアライズド・ビュー・データベースで本番マテリアライズド・ビューの高速リフレッシュを実行します。
4. ダミー・マテリアライズド・ビューを削除します。

親トピック: 読取り専用マテリアライズド・ビューの作成および管理

39.6 マテリアライズド・ビュー・ログの監視

マスター・データベースのマテリアライズド・ビュー・ログの情報を表示するために、問合せを実行できます。

- マスター・データベースのマテリアライズド・ビュー・ログの情報のリスト
  マテリアライズド・ビュー・ログにより、マスターに基づいてマテリアライズド・ビューの高速リフレッシュを実行できます。マス

ターには、マスター表とマスター・マテリアライズド・ビューがあります。

- マテリアライズド・ビュー・ログを使用するマテリアライズド・ビューのリスト
  複数のマテリアライズド・ビューがマテリアライズド・ビュー・ログを使用できます。

親トピック: 読取り専用マテリアライズド・ビューの作成および管理

39.6.1 マスター・データベースにあるマテリアライズド・ビュー・ログの情報のリスト表示

マテリアライズド・ビュー・ログにより、マスターをベースとするマテリアライズド・ビューで高速リフレッシュを実行できます。マスターには、マスター表とマスター・マテリアライズド・ビューがあります。

マスターをベースとするマテリアライズド・ビュー・ログがある場合は、この項の問合せを使用して、このログに関する次の情報をリストできます。

- マテリアライズド・ビュー・ログ・データが格納される各ログ表の名前
- 各マテリアライズド・ビュー・ログの所有者
- 各マテリアライズド・ビュー・ログがベースとするマスター
- マテリアライズド・ビュー・ログが行IDマテリアライズド・ビュー・ログかどうか
- マテリアライズド・ビュー・ログが主キー・マテリアライズド・ビュー・ログかどうか
- マテリアライズド・ビュー・ログがオブジェクトIDマテリアライズド・ビュー・ログかどうか
- マテリアライズド・ビュー・ログにフィルタ列があるかどうか

この情報を表示するには、次のステップを実行します。

1. マスター・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行します。

```
COLUMN LOG_TABLE HEADING 'Log Table' FORMAT A20
COLUMN LOG_OWNER HEADING 'Log|Owner' FORMAT A5
COLUMN MASTER HEADING 'Master' FORMAT A15
COLUMN ROWIDS HEADING 'Row|ID?' FORMAT A3
COLUMN PRIMARY_KEY HEADING 'Primary|Key?' FORMAT A7
COLUMN OBJECT_ID HEADING 'Object|ID?' FORMAT A6
COLUMN FILTER_COLUMNS HEADING 'Filter|Columns?' FORMAT A8

SELECT DISTINCT LOG_TABLE,
       LOG_OWNER,
       MASTER,
       ROWIDS,
       PRIMARY_KEY,
       OBJECT_ID,
       FILTER_COLUMNS
    FROM DBA_MVIEW_LOGS
    ORDER BY 1;
```

出力は次のようになります。

```
                     Log                    Row Primary Object Filter
Log Table            Owner Master           ID? Key?    ID?    Columns?
-------------------- ----- ---------------- --- ------- ------ --------
MLOG$_COUNTRIES      HR    COUNTRIES        NO  YES     NO     NO
MLOG$_DEPARTMENTS    HR    DEPARTMENTS      NO  YES     NO     NO
MLOG$_EMPLOYEES      HR    EMPLOYEES        NO  YES     NO     NO
MLOG$_JOBS           HR    JOBS             NO  YES     NO     NO
MLOG$_JOB_HISTORY    HR    JOB_HISTORY      NO  YES     NO     NO
MLOG$_LOCATIONS      HR    LOCATIONS        NO  YES     NO     NO
```

```
MLOG$_REGIONS          HR     REGIONS           NO   YES      NO      NO
```

親トピック: マテリアライズド・ビュー・ログの監視

39.6.2 マテリアライズド・ビュー・ログを使用するマテリアライズド・ビューのリスト表示

あるマテリアライズド・ビュー・ログを複数のマテリアライズド・ビューが使用する場合があります。

マスターをベースとするマテリアライズド・ビュー・ログがある場合は、この項の問合せを使用して、各ログを使用するマテリアライズド・ビューに関する次の情報をリストできます。

- マテリアライズド・ビュー・ログ・データが格納される各ログ表の名前
- 各マテリアライズド・ビュー・ログの所有者
- 各マテリアライズド・ビュー・ログがベースとするマスター
- マテリアライズド・ビュー・ログを使用する各マテリアライズド・ビューの、マテリアライズド・ビュー識別番号
- マテリアライズド・ビュー・ログを使用する各マテリアライズド・ビューの名前

この情報を表示するには、次のステップを実行します。

1. マスター・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行します。

```
COLUMN LOG_TABLE HEADING 'Mview|Log Table' FORMAT A20
COLUMN LOG_OWNER HEADING 'Mview|Log Owner' FORMAT A10
COLUMN MASTER HEADING 'Master' FORMAT A20
COLUMN MVIEW_ID HEADING 'Mview|ID' FORMAT 9999
COLUMN NAME HEADING 'Mview Name' FORMAT A20

SELECT L.LOG_TABLE, L.LOG_OWNER, B.MASTER, B.MVIEW_ID, R.NAME
FROM ALL_MVIEW_LOGS L, ALL_BASE_TABLE_MVIEWS B, ALL_REGISTERED_MVIEWS R
WHERE B.MVIEW_ID = R.MVIEW_ID
AND B.OWNER = L.LOG_OWNER
AND B.MASTER = L.MASTER;
```

出力は次のようになります。

```
Mview                Mview                              Mview
Log Table            Log Owner  Master                     ID Mview Name
-------------------- ---------- -------------------- ----- --------------------
MLOG$_COUNTRIES      HR         COUNTRIES               21 COUNTRIES_MV1
MLOG$_DEPARTMENTS    HR         DEPARTMENTS             22 DEPARTMENTS_MV1
MLOG$_EMPLOYEES      HR         EMPLOYEES               23 EMPLOYEES_MV1
MLOG$_JOBS           HR         JOBS                    24 JOBS_MV1
MLOG$_JOB_HISTORY    HR         JOB_HISTORY             25 JOB_HISTORY_MV1
MLOG$_LOCATIONS      HR         LOCATIONS               26 LOCATIONS_MV1
MLOG$_REGIONS        HR         REGIONS                 27 REGIONS_MV1
```

親トピック: マテリアライズド・ビュー・ログの監視

39.7 マテリアライズド・ビューの監視

マテリアライズド・ビューおよびリフレッシュ・グループの情報を表示するために、問合せを実行できます。

- マテリアライズド・ビューの情報のリスト表示
  マテリアライズド・ビューの情報を表示するために、問合せを実行できます。
- マテリアライズド・ビュー・データベースのリフレッシュ・グループの情報のリスト

マテリアライズド・ビュー・データベースの各リフレッシュ・グループは、設定された間隔でリフレッシュ・グループ内のマテリアライズド・ビューをリフレッシュするリフレッシュ・ジョブに関連付けられています。

- マテリアライズド・ビュー・データベースの各リフレッシュ・ジョブのジョブIDの判断
  マテリアライズド・ビュー・データベースの各リフレッシュ・ジョブのジョブ識別番号を判断するには、DBA_REFRESHおよびDBA_JOBSビューの問合せを実行します。
- 現在リフレッシュ中のマテリアライズド・ビューの判断
  現在リフレッシュ中のマテリアライズド・ビューを判断するには、V$MVREFRESHビューの問合せを実行します。

親トピック: 読取り専用マテリアライズド・ビューの作成および管理

39.7.1 マテリアライズド・ビューの情報のリスト表示

マテリアライズド・ビューの情報を表示するために、問合せを実行できます。

- マテリアライズド・ビューのマスター・データベース情報のリスト
  マテリアライズド・ビューのマスター・データベース情報のリストを表示するには、DBA_MVIEWSビューの問合せを実行します。
- マテリアライズド・ビューのプロパティのリスト表示
  マテリアライズド・ビューのプロパティのリストを表示するには、DBA_MVIEWSビューの問合せを実行します。

親トピック: マテリアライズド・ビューの監視

39.7.1.1 マテリアライズド・ビューのマスター・データベース情報のリスト表示

マテリアライズド・ビューのマスター・データベース情報のリストを表示するには、DBA_MVIEWSビューの問合せを実行します。

レプリケーション・データベースの各マテリアライズド・ビューのマスター・データベースを表示し、マテリアライズド・ビューが高速リフレッシュ可能かどうかを確認するには、次のステップを実行します。

1. マテリアライズド・ビュー・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行します。

```
COLUMN MVIEW_NAME HEADING 'Materialized|View Name' FORMAT A15
COLUMN OWNER HEADING 'Owner' FORMAT A10
COLUMN MASTER_LINK HEADING 'Master Link' FORMAT A30
COLUMN Fast_Refresh HEADING 'Fast|Refreshable?' FORMAT A16
SELECT MVIEW_NAME,
       OWNER,
       MASTER_LINK,
       DECODE(FAST_REFRESHABLE,
              'NO', 'NO',
              'DML', 'YES',
              'DIRLOAD', 'DIRECT LOAD ONLY',
              'DIRLOAD_DML', 'YES',
              'DIRLOAD_LIMITEDDML', 'LIMITED') Fast_Refresh
    FROM DBA_MVIEWS;
```

出力は次のようになります。

```
Materialized                                                    Fast
View Name       Owner      Master Link                          Refreshable?
--------------- ---------- ------------------------------ ----------------
COUNTRIES_MV1   HR         @ORC1.EXAMPLE.COM YES
DEPARTMENTS_MV1 HR         @ORC1.EXAMPLE.COM YES
EMPLOYEES_MV1   HR         @ORC1.EXAMPLE.COM YES
JOBS_MV1        HR         @ORC1.EXAMPLE.COM YES
JOB_HISTORY_MV1 HR         @ORC1.EXAMPLE.COM YES
LOCATIONS_MV1   HR         @ORC1.EXAMPLE.COM YES
```

```
REGIONS_MV1     HR          @ORC1.EXAMPLE.COM YES
```

親トピック: マテリアライズド・ビューの情報のリスト表示

39.7.1.2 マテリアライズド・ビューのプロパティのリスト表示

マテリアライズド・ビューのプロパティのリストを表示するには、DBA_MVIEWSビューの問合せを実行します。

この項の問合せを使用して、カレント・レプリケーション・データベースにあるマテリアライズド・ビューに関する次の情報をリストできます。

- 各マテリアライズド・ビューの名前
- 各マテリアライズド・ビューの所有者
- 各マテリアライズド・ビューが使用するリフレッシュ方法がCOMPLETE、FORCE、FASTまたはNEVERのいずれであるか
- 各マテリアライズド・ビューがリフレッシュされた最後の日付

この情報を表示するには、次のステップを実行します。

1. マテリアライズド・ビュー・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行し、この情報をリストします。

この情報を表示するには、次のステップを実行します。

1. マテリアライズド・ビュー・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行します。

```
COLUMN MVIEW_NAME HEADING 'Materialized|View Name' FORMAT A15
COLUMN OWNER HEADING 'Owner' FORMAT A10
COLUMN REFRESH_METHOD HEADING 'Refresh|Method' FORMAT A10
COLUMN LAST_REFRESH_DATE HEADING 'Last|Refresh|Date'
COLUMN LAST_REFRESH_TYPE HEADING 'Last|Refresh|Type' FORMAT A15
SELECT MVIEW_NAME,
       OWNER,
       REFRESH_METHOD,
       LAST_REFRESH_DATE,
       LAST_REFRESH_TYPE
    FROM DBA_MVIEWS;
```

出力は次のようになります。

```
                                      Last      Last
Materialized                Refresh   Refresh   Refresh
View Name       Owner       Method    Date      Type
--------------- ----------- --------- --------- ---------------
COUNTRIES_MV1   HR          FAST      21-OCT-03 FAST
DEPARTMENTS_MV1 HR          FAST      21-OCT-03 FAST
EMPLOYEES_MV1   HR          FAST      21-OCT-03 FAST
JOBS_MV1        HR          FAST      21-OCT-03 FAST
JOB_HISTORY_MV1 HR          FAST      21-OCT-03 FAST
LOCATIONS_MV1   HR          FAST      21-OCT-03 FAST
REGIONS_MV1     HR          FAST      21-OCT-03 FAST
```

親トピック: マテリアライズド・ビューの情報のリスト表示

39.7.2 マテリアライズド・ビュー・データベースにあるリフレッシュ・グループの情報のリスト表示

マテリアライズド・ビュー・データベースの各リフレッシュ・グループは、設定された間隔でリフレッシュ・グループ内のマテリアライズド・ビューをリフレッシュするリフレッシュ・ジョブに関連付けられています。

DBA_REFRESHデータ・ディクショナリ・ビューの問合せにより、マテリアライズド・ビュー・データベースのリフレッシュ・ジョブに関する次の情報をリストできます。

- リフレッシュ・グループの名前。
- リフレッシュ・グループの所有者。
- リフレッシュ・ジョブが中断されているかどうか。
- 次回リフレッシュ・ジョブが実行される日付および時刻。
- リフレッシュ・ジョブの現在の間隔設定。間隔設定では、ジョブの開始と同じジョブの次回の開始の間の、時間の長さを指定します。

この情報を表示するには、次のステップを実行します。

1. マテリアライズド・ビュー・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行します。

```
COLUMN RNAME HEADING 'Refresh|Group|Name' FORMAT A10
COLUMN ROWNER HEADING 'Refresh|Group|Owner' FORMAT A10
COLUMN BROKEN HEADING 'Broken?' FORMAT A7
COLUMN next_refresh HEADING 'Next Refresh'
COLUMN INTERVAL HEADING 'Interval' FORMAT A20
SELECT RNAME,
                ROWNER,
       BROKEN,
       TO_CHAR(NEXT_DATE, 'DD-MON-YYYY HH:MI:SS AM') next_refresh,
       INTERVAL
    FROM DBA_REFRESH
    ORDER BY 1;
```

出力は次のようになります。

```
Refresh    Refresh
Group      Group
Name       Owner      Broken? Next Refresh            Interval
---------- ---------- ------- ---------------------- --------------------
HR_REFG    MVIEWADMIN N       24-OCT-2003 07:18:44 AM SYSDATE + 1/24
```

Broken?列のNは、ジョブが中断されていないことを意味します。したがって、リフレッシュ・ジョブは次回開始時刻に実行されます。この列のYは、ジョブが中断されていることを意味します。

親トピック: マテリアライズド・ビューの監視

39.7.3 マテリアライズド・ビュー・データベースにある各リフレッシュ・ジョブのジョブIDの判定

マテリアライズド・ビュー・データベースの各リフレッシュ・ジョブのジョブ識別番号を判断するには、DBA_REFRESHおよびDBA_JOBSビューの問合せを実行します。

問合せを実行して、マテリアライズド・ビュー・データベースのリフレッシュ・ジョブに関する次の情報をリストできます。

- 各リフレッシュ・ジョブのジョブ識別番号。Oracle Schedulerによって作成された各ジョブには、一意の識別番号が割り当てられます。
- 権限スキーマ。権限スキーマとは、ジョブに適用されるデフォルトの権限を所有するスキーマです。
- 各リフレッシュ・ジョブを所有するスキーマ。
- ジョブがリフレッシュするリフレッシュ・グループの名前。
- リフレッシュ・ジョブのステータスが通常か中断か。

この情報を表示するには、次のステップを実行します。

1. マテリアライズド・ビュー・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行します。

```
COLUMN JOB HEADING 'Job ID' FORMAT 999999
COLUMN PRIV_USER HEADING 'Privilege|Schema' FORMAT A10
COLUMN RNAME HEADING 'Refresh|Group|Name' FORMAT A10
COLUMN ROWNER HEADING 'Refresh|Group|Owner' FORMAT A10
COLUMN BROKEN HEADING 'Broken?' FORMAT A7
SELECT J.JOB,
       J.PRIV_USER,
       R.ROWNER,
       R.RNAME,
       J.BROKEN
    FROM DBA_REFRESH R, DBA_JOBS J
    WHERE R.JOB = J.JOB
    ORDER BY 1;
```

出力は次のようになります。

```
                    Refresh    Refresh
        Privilege   Group      Group
 Job ID Schema      Owner      Name       Broken?
------- ---------- ---------- ---------- -------
     21 MVIEWADMIN MVIEWADMIN HR_REFG    N
```

Broken?列のNは、ジョブが中断されていないことを意味します。したがって、ジョブは次回開始時刻に実行されます。この列のYは、ジョブが中断されていることを意味します。

親トピック: マテリアライズド・ビューの監視

39.7.4 現在リフレッシュしているマテリアライズド・ビューの判定

現在リフレッシュ中のマテリアライズド・ビューを判断するには、V$MVREFRESHビューの問合せを実行します。

現在リフレッシュ中のマテリアライズド・ビューを表示するには、次のステップを実行します。

1. マテリアライズド・ビュー・データベースに管理ユーザーとして接続します。
2. 次の問合せを実行します。

```
COLUMN SID HEADING 'Session|Identifier' FORMAT 9999
COLUMN SERIAL# HEADING 'Serial|Number' FORMAT 999999
COLUMN CURRMVOWNER HEADING 'Owner' FORMAT A15
COLUMN CURRMVNAME HEADING 'Materialized|View' FORMAT A25
SELECT * FROM V$MVREFRESH;
```

出力は次のようになります。

```
   Session  Serial                 Materialized
Identifier  Number Owner           View
```

```
---------- ------- --------------- -------------------------
        19     233 HR              COUNTRIES_MV
         5     647 HR              EMPLOYEES_MV
```

親トピック: マテリアライズド・ビューの監視

40 読取り専用マテリアライズド・ビューの問題のトラブルシューティング

データベース・リンク、マテリアライズド・ビューの作成、およびマテリアライズド・ビューのリフレッシュに関する問題を診断および解決できます。

- データベース・リンクの問題の診断
  データベース・リンクが正常に機能していないと考えられる場合、Oracle Enterprise Manager Cloud Control、SQL*Plusまたは別のツールを使用してデータベース・リンクを削除し、再作成できます。
- マテリアライズド・ビューの作成の問題
  マテリアライズド・ビューの作成に問題がある場合は、チェックする項目があります。
- リフレッシュに関する問題
  一般的なリフレッシュの問題を診断し、解決できます。
- リフレッシュに関する問題の高度なトラブルシューティング
  マテリアライズド・ビューのリフレッシュに問題がある場合は、いくつかの項目をチェックできます。

親トピック: 読取り専用マテリアライズド・ビューの管理

40.1 データベース・リンクに関する問題の診断方法

データベース・リンクが正常に機能していないと考えられる場合、Oracle Enterprise Manager Cloud Control、SQL*Plusまたは別のツールを使用してデータベース・リンクを削除し、再作成します。

- データベース・リンク名が、ターゲット・データベースのグローバル名と同じであることを確認します。
- スケジュールした間隔が意図したとおりであることを確認します。
- スケジュールした間隔が、必要な実行時間よりも短くないことを確認します。

指定されたデータベースへのデータベース・リンクに接続修飾子が使用されている場合は、そのデータベースにリンクする他のデータベースにも同じ接続修飾子が必要です。たとえば、次のようにデータベース・リンクを作成するとします。

```
CREATE DATABASE LINK dbs1.example.com@myethernet CONNECT TO myadmin
  IDENTIFIED BY password USING 'connect_string_myethernet';
```

マスター・データベースかマテリアライズド・ビュー・データベースかにかかわらず、`dbs1.example.com@myethernet`に対応付けられたすべてのデータベースで、`myethernet`を接続修飾子として使用する必要があります。

関連項目:

- データベース・リンクおよび接続修飾子の詳細は、「リンク名に含まれるサービス名を指定するための接続修飾子の使用」
- 「マテリアライズド・ビューのために必要なデータベース・リンク」

親トピック: 読取り専用マテリアライズド・ビューの問題のトラブルシューティング

40.2 マテリアライズド・ビューの作成に関する問題

マテリアライズド・ビューの作成に問題がある場合は、チェックする項目があります。

マテリアライズド・ビューの作成に失敗した場合は、次の処理を試行します。

- マテリアライズド・ビューを作成するための必要な権限があることを確認します。マスター表とそのマテリアライズド・ビュー・ログに対するSELECT権限が必要です。詳細は、「必要な権限」を参照してください。
- 高速リフレッシュの主キーまたは副問合せのマテリアライズド・ビューを作成しようとしている場合は、マスター表のマテリアライズド・ビュー・ログに主キーがログされることを確認します。
- 高速リフレッシュを実行するROWIDマテリアライズド・ビューを作成する場合は、マスター表のマテリアライズド・ビュー・ログにROWIDがロギングされていることを確認します。
- 副問合せマテリアライズド・ビューの場合は、必要な列がマテリアライズド・ビュー・ログに追加されていることを確認します。詳細は、「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。
- 高速リフレッシュを実行するマテリアライズド・ビューと関連するすべての表に対して、マテリアライズド・ビュー・ログが存在することを確認します。マテリアライズド・ビューに副問合せが含まれている場合は、副問合せで参照される表のそれぞれに対してマテリアライズド・ビュー・ログが必要です。

親トピック: 読取り専用マテリアライズド・ビューの問題のトラブルシューティング

40.3 リフレッシュに関する問題

一般的なリフレッシュの問題を診断し、解決できます。

- リフレッシュに関する一般的な問題
  いくつかの共通の要因が、マテリアライズド・ビューのグループの自動リフレッシュを妨げる可能性があります。
- 自動リフレッシュの再試行
  Oracleデータベースが自動的にリフレッシュ・グループをリフレッシュできなかった場合、リフレッシュ・グループはリフレッシュの完了が予定されたままになります。
- 新しいマテリアライズド・ビュー・データベースでの高速リフレッシュ・エラー
  場合によっては、新規マテリアライズド・ビュー・データベースでマテリアライズド・ビューの作成中に、マスター表のマテリアライズド・ビュー・ログがパージされることがあります。
- マテリアライズド・ビューが繰り返しリフレッシュされる場合
  マテリアライズド・ビュー・グループのリフレッシュが継続的に繰り返される場合、グループのリフレッシュ間隔をチェックします。
- マテリアライズド・ビュー・ログが大きくなりすぎる場合
  マスター・データベースでマテリアライズド・ビュー・ログが大きくなりすぎる場合は、ネットワークまたはデータベース障害により、マスター・データベースがマテリアライズド・ビューの削除を認識できなかったかどうかを確認してください。

親トピック: 読取り専用マテリアライズド・ビューの問題のトラブルシューティング

40.3.1 リフレッシュに関する一般的な問題

いくつかの共通の要因が、マテリアライズド・ビューのグループの自動リフレッシュを妨げる可能性があります。

これらの要因には次のものがあります。

- マテリアライズド・ビュー・データベースでのジョブ・スレーブの欠落

- ネットワークまたはサーバー障害の影響
- サーバー・シャットダウンの影響

リフレッシュ・グループの問題が発生したときは、前述の状況のいずれかが、Oracleデータベースでのグループのリフレッシュの完了を妨げていないことを確認します。

親トピック: リフレッシュに関する問題

40.3.2 自動リフレッシュの再試行

Oracleデータベースが自動的にリフレッシュ・グループをリフレッシュできなかった場合、リフレッシュ・グループはリフレッシュの完了が予定されたままになります。

Oracleデータベースでは、次の動作によりグループの自動リフレッシュを再試行します。

- リフレッシュ・グループのリフレッシュの再試行を、初回は1分後、次は2分後、その次は4分後というように、失敗するたびに再試行間隔を倍にします。
- 再試行間隔が、リフレッシュ間隔より長くなることは許可されません。
- Oracleデータベースは、最高16回まで自動リフレッシュを再試行します。

リフレッシュ・グループのリフレッシュを16回再試行した後、エラーが引続き発生する場合は、グループは破損しているとみなされます。USER_REFRESHおよびUSER_REFRESH_CHILDRENデータ・ディクショナリ・ビューのBROKEN列を問い合せて、リフレッシュ・グループの現在の状態を確認できます。

リフレッシュ・グループが破損したとみなされる原因となったエラーは、トレース・ファイルに記録されます。リフレッシュ・グループが正常にリフレッシュされない問題を修正した後、リフレッシュ・グループを手動でリフレッシュする必要があります。これにより、自動リフレッシュが再度発生できるように、破損したフラグがリセットされます。

関連項目:

マテリアライズド・ビュー・トレース・ファイルの名前の形式はjnで、nはオペレーティング・システム固有です。各システムにおける名前は、オペレーティング・システム固有のOracleドキュメントを参照してください。

親トピック: リフレッシュに関する問題

40.3.3 新規マテリアライズド・ビュー・データベースでの高速リフレッシュ・エラー

場合によっては、新規マテリアライズド・ビュー・データベースでマテリアライズド・ビューの作成中に、マスター表のマテリアライズド・ビュー・ログがパージされることがあります。

これが発生すると、次のエラーが検出される場合があります。

```
ORA-12004 REFRESH FAST cannot be used for materialized view materialized_view_name
ORA-12034 materialized view log on materialized_view_name younger than last refresh
```

関連項目:

この問題を回避する方法の詳細は、「新しいマテリアライズド・ビュー・データベースの追加」を参照してください。

親トピック: リフレッシュに関する問題

40.3.4 マテリアライズド・ビューが繰り返しリフレッシュされる場合

マテリアライズド・ビュー・グループのリフレッシュが継続的に繰り返される場合、グループのリフレッシュ間隔をチェックします。

Oracleデータベースは、リフレッシュを開始する前に、リフレッシュ・グループの自動リフレッシュ間隔を評価します。リフレッシュ・グループのリフレッシュ間隔がグループ内のすべてのマテリアライズド・ビューのリフレッシュに要する時間よりも短い場合、ジョブ・スレーブが未処理ジョブのキューをチェックするたびに、リフレッシュ・グループのリフレッシュが繰り返し開始されます。

親トピック: リフレッシュに関する問題

40.3.5 マテリアライズド・ビュー・ログが大きくなりすぎる場合

マスター・データベースでマテリアライズド・ビュー・ログが大きくなりすぎる場合は、ネットワークまたはデータベース障害により、マスター・データベースがマテリアライズド・ビューの削除を認識できなかったかどうかを確認してください。

マテリアライズド・ビュー・ログの部分的なパージ、または未使用のマテリアライズド・ビュー・データベースの登録解除が必要になる場合があります。

親トピック: リフレッシュに関する問題

40.4 リフレッシュに関する問題の高度なトラブルシューティング

マテリアライズド・ビューのリフレッシュに問題がある場合は、いくつかの項目をチェックできます。

マテリアライズド・ビューのリフレッシュに関する問題が発生している場合、次の処理を実行します。

- DBA_REFRESH_CHILDRENビューに表示されているNEXT_DATE値をチェックして、リフレッシュがスケジュールされているかどうかを判断します。
- リフレッシュ間隔が経過している場合は、DBA_REFRESHビューで、マテリアライズド・ビューのリフレッシュが対応付けられたジョブ番号をチェックして、ジョブ・キューに関する問題の診断を行います。
- 実行中のジョブ・スレーブがあるかどうかをチェックします。JOB_QUEUE_PROCESSES初期化パラメータをチェックし、DBA_JOBS_RUNNINGビューを問い合せ、オペレーティング・システムを使用してジョブ・スレーブが稼働中かどうかをチェックします。
- 2つのマテリアライズド・ビュー間のマスター・ディテール関係を定義すると、エラーが発生することもあります。マスター・ディテール関係は、宣言参照整合性の制約を使用して、マスター表に対してのみ定義します。その後、関連するマテリアライズド・ビューを同じリフレッシュ・グループ内に入れて、この関係を保ちます。ただし、遅延(遅延可能)制約はマテリアライズド・ビューに対して定義します。
- 同じリフレッシュ・グループ内のマテリアライズド・ビューでは、行は1つのトランザクションで更新されます。このようなトランザクションは非常に大きくなる可能性があるので、大きなロールバック・セグメント(リフレッシュ中に使用するように指定したロールバック・セグメントを含む)がマテリアライズド・ビュー・データベースで必要になるか、または、リフレッシュをより頻繁に実行してトランザクションのサイズを小さくする必要があります。
- OracleエラーORA-12004が発生する場合は、マテリアライズド・ビュー・ログを維持しようとしたときにマスター・データベースでロールバック・セグメントが不足したか、マテリアライズド・ビュー・ログの情報が古い可能性があります。たとえば、マテリアライズド・ビュー・ログがパージまたは再作成された可能性があります。
- 1つのマテリアライズド・ビューの完全リフレッシュでは、内部的にTRUNCATE機能が使用されるので、処理速度が向上し、ロールバック・セグメントの必要量が少なくなります。ただし、マテリアライズド・ビューが完全リフレッシュされるまで、マテリアライズド・ビューには一時的にデータが表示されないことがあります。複数のマテリアライズド・ビュー(リフレッシュ・グ

ループなど)をリフレッシュするときは、TRUNCATE機能は使用されません。

- マスター表を再編成するとき(たとえば、システム・リソースの再利用などの目的で)は、マスター表にTRUNCATE機能を実行して、ROWIDマテリアライズド・ビューの完全リフレッシュを実行する必要があります。そうしないと、マテリアライズド・ビューがマスター表への誤ったROWIDを持つことになります。マスター表の再編成には、DBMS_MVIEWパッケージのBEGIN_TABLE_REORGANIZATIONおよびEND_TABLE_REORGANIZATIONプロシージャを使用します。
- リフレッシュの実行中にエラーORA-00942(表またはビューが存在しません)が発生した場合は、データベース・リンクをチェックし、マスター表とマテリアライズド・ビュー・ログに対する必要な権限があるかどうかを確認します。
- 高速リフレッシュが成功した後で障害が発生した場合、次の点をチェックします。
    - マテリアライズド・ビュー・ログの切捨て、パージまたは削除が実行されたかどうか
    - マテリアライズド・ビュー・ログに対する必要な権限を持っているかどうか
- 強制リフレッシュに非常に長い時間を要する場合は、リフレッシュ時に使用するマテリアライズド・ビュー・ログが削除されていないかチェックします。
- BUILD DEFERREDを使用して作成したマテリアライズド・ビューの最初の高速リフレッシュが失敗した場合は、その他の問題をチェックする前に、前の完全リフレッシュが成功していたことを確認します。

親トピック: 読取り専用マテリアライズド・ビューの問題のトラブルシューティング

付録

付録には、このドキュメントの補足資料が含まれています。

- DBMS_JOBのサポート
  DBMS_JOBパッケージは引き続きサポートされます。ただし、DBMS_JOBジョブを発行するデータベース・スキーマにCREATE　JOB権限を付与する必要があります。
- ブロックチェーン表参照
  行内容を使用して、行のハッシュ値および署名を独立して検証できます。

A DBMS_JOBのサポート

DBMS_JOBパッケージは引き続きサポートされます。ただし、DBMS_JOBジョブを発行するデータベース・スキーマにCREATE JOB権限を付与する必要があります。

Oracle Schedulerは、DBMS_JOBパッケージを置換します。DBMS_JOBは下位互換性のために引き続きサポートされますが、DBMS_JOBからOracle Schedulerに切り替えることをお薦めします。

Oracle Database 19c以降のリリースでは、アップグレードでDBMS_SCHEDULERを使用して既存のDBMS_JOBジョブを再作成できる場合、下位互換性のために、アップグレード後も引き続きDBMS_JOBはDBMS_SCHEDULERジョブに対するレガシー・インタフェースとして機能します。メタデータの問題のため、DBMS_SCHEDULERを使用して既存のジョブを再作成できない場合は、アップグレードの事前チェックを実行した際にJOB_TABLE_INTEGRITY警告が表示されます。その場合、次の3つのオプションがあります。

- メタデータを修正します。アップグレード後も引き続きDBMS_JOBSをインタフェースとして使用して実行し、DBMS_SCHEDULERジョブとして実行します。
- 不要な場合は、ジョブを削除します。
- DBMS_JOBSジョブを削除し、DBMS_SCHEDULERを使用して手動でジョブを再作成します。

アップグレード時に再作成されるDBMS_JOBで作成された既存のジョブの場合、DBMS_JOBレガシー・ジョブは引き続きインタフェースとして存在しますが、これを使用すると常にDBMS_SCHEDULERエントリが作成されます。インタフェースは別として、ジョブはDBMS_SCHEDULERジョブとして実行されます。その後、アップグレード前に作成されたDBMS_JOBジョブを無効にすると、DBMS_SCHEDULERジョブも無効になります。この動作を回避するには、レガシー・ジョブを削除し、DBMS_SCHEDULERジョブで置換します。

すべての新しいジョブには、DBMS_SCHEDULERを使用します。

- DBMS_JOBからOracle Schedulerへの置換え
  Oracle Database 11g リリース2 (11.2)以降では、DBMS_JOBがOracle Schedulerに置き換えられています。Oracle Schedulerは、ジョブのスケジュールに使用されるパッケージであるDBMS_JOBよりも、強力で柔軟性があります。DBMS_JOBは下位互換性のために引き続きサポートされますが、DBMS_JOBからOracle Schedulerに切り替えることをお薦めします。
- DBMS_JOBからOracle Schedulerへの移行
  この項では、DBMS_JOBパッケージで作成されたジョブを取得し、DBMS_SCHEDULERパッケージで構成して制御するOracle Schedulerを使用してそれらをリライトする方法の例をいくつか示します。

親トピック: 付録

A.1 DBMS_JOBからOracle Schedulerへの置換え

Oracle Database 11g リリース2 (11.2)以降では、DBMS_JOBがOracle Schedulerに置き換えられています。Oracle Schedulerは、ジョブのスケジュールに使用されるパッケージであるDBMS_JOBよりも、強力で柔軟性があります。DBMS_JOBは下位互換性のために引き続きサポートされますが、DBMS_JOBからOracle Schedulerに切り替えることをお薦めします。

- DBMS_JOBの構成
  JOB_QUEUE_PROCESSES初期化パラメータでは、ジョブの実行用に作成できるプロセスの最大数を指定します。
- DBMS_JOBとOracle Schedulerの使用
  DBMS_JOBとOracle Scheduler(スケジューラ)は、同じジョブ・コーディネータを使用して、ジョブ・スレーブを起動しま

す。

親トピック: DBMS_JOBのサポート

A.1.1 DBMS_JOBの構成

JOB_QUEUE_PROCESSES初期化パラメータには、ジョブの実行に対して作成可能なプロセスの最大数を指定する。

Oracle Database 12cリリース2以降では、JOB_QUEUE_PROCESSESのデフォルトが4000になっています。ジョブ・コーディネータ・プロセスは、実行されるジョブ数と使用可能なリソースに基づいて、必要な数のジョブ・キュー・プロセスのみを起動します。JOB_QUEUE_PROCESSESにより低い数字を設定すると、ジョブ・キュー・プロセスの数を制限できます。

JOB_QUEUE_PROCESSESを0に設定すると、DBMS_JOBジョブおよびDBMS_SCHEDULERジョブが使用不可になります。

関連項目:

JOB_QUEUE_PROCESSES初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: DBMS_JOBからOracle Schedulerへの置換え

A.1.2 DBMS_JOBとOracle Schedulerの使用

DBMS_JOBとOracle Scheduler(スケジューラ)は、同じジョブ・コーディネータを使用して、ジョブ・スレーブを起動します。

JOB_QUEUE_PROCESSES初期化パラメータを使用すると、DBMS_JOBとスケジューラの両方のジョブ・スレーブ数を制限できます。

JOB_QUEUE_PROCESSESが0の場合、DBMS_JOBとOracle Schedulerジョブの両方が無効になります。

関連項目:

- Oracle Schedulerを使用したジョブのスケジューリング
- 「スケジューラのプリファレンスの設定」
- JOB_QUEUE_PROCESSES初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

親トピック: DBMS_JOBからOracle Schedulerへの置換え

A.2 DBMS_JOBからOracle Schedulerへの移行

この項では、DBMS_JOBパッケージで作成されたジョブを取得し、DBMS_SCHEDULERパッケージを構成して制御するOracle Schedulerを使用してそれらをリライトする方法の例をいくつか示します。

- ジョブの作成
  DBMS_JOBパッケージおよびDBMS_SCHEDULERパッケージを使用したジョブの作成を例で示します。
- ジョブの変更
  DBMS_JOBパッケージおよびDBMS_SCHEDULERパッケージを使用したジョブの変更を例で示します。
- ジョブ・キューからのジョブの削除
  DBMS_JOBパッケージおよびDBMS_SCHEDULERパッケージを使用したジョブの削除を例で示します。

親トピック: DBMS_JOBのサポート

A.2.1 ジョブの作成

DBMS_JOBパッケージおよびDBMS_SCHEDULERパッケージを使用したジョブの作成を例で示します。

次の例では、DBMS_JOBを使用してジョブを作成します。

```
VARIABLE jobno NUMBER;
BEGIN
 DBMS_JOB.SUBMIT(:jobno, 'INSERT INTO employees VALUES (7935, ''SALLY'',
   ''DOGAN'', ''sally.dogan@examplecorp.com'', NULL, SYSDATE, ''AD_PRES'', NULL,
    NULL, NULL, NULL);', SYSDATE, 'SYSDATE+1');
 COMMIT;
END;
/
```

DBMS_SCHEDULERを使用した等価の文は、次のとおりです。

```
BEGIN
 DBMS_SCHEDULER.CREATE_JOB(
   job_name          =>  'job1',
   job_type          =>  'PLSQL_BLOCK',
   job_action        =>  'INSERT INTO employees VALUES (7935, ''SALLY'',
     ''DOGAN'', ''sally.dogan@examplecorp.com'', NULL, SYSDATE,''AD_PRES'', NULL,
      NULL, NULL, NULL);',
   start_date        =>  SYSDATE,
   repeat_interval   =>  'FREQ = DAILY; INTERVAL = 1');
END;
/
```

親トピック: DBMS_JOBからOracle Schedulerへの移行

A.2.2 ジョブの変更

DBMS_JOBパッケージおよびDBMS_SCHEDULERパッケージを使用したジョブの変更を例で示します。

次の例では、DBMS_JOBを使用してジョブを変更します。

```
BEGIN
 DBMS_JOB.WHAT(31, 'INSERT INTO employees VALUES (7935, ''TOM'', ''DOGAN'',
   ''tom.dogan@examplecorp.com'', NULL, SYSDATE,''AD_PRES'', NULL,
   NULL, NULL, NULL);');
 COMMIT;
END;
/
```

この文では、別の値を挿入するようにJOB1の処理が変更されます。

DBMS_SCHEDULERを使用した等価の文は、次のとおりです。

```
BEGIN
 DBMS_SCHEDULER.SET_ATTRIBUTE(
   name          => 'JOB1',
   attribute     => 'job_action',
   value         => 'INSERT INTO employees VALUES (7935, ''TOM'', ''DOGAN'',
      ''tom.dogan@examplecorp.com'', NULL, SYSDATE, ''AD_PRES'', NULL,
      NULL, NULL, NULL);');
END;
/
```

親トピック: DBMS_JOBからOracle Schedulerへの移行

A.2.3 ジョブ・キューからのジョブの削除

DBMS_JOBパッケージおよびDBMS_SCHEDULERパッケージを使用したジョブの削除を例で示します。

次の例では、DBMS_JOBを使用してジョブを削除します。14144は実行されているジョブの番号です。

```
BEGIN
   DBMS_JOB.REMOVE(14144);
COMMIT;
END;
/
```

DBMS_SCHEDULERを使用して、かわりに次のような文を発行します。

```
BEGIN
   DBMS_SCHEDULER.DROP_JOB('myjob1');
END;
/
```

関連項目:

- DBMS_SCHEDULERパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- Oracle Schedulerを使用したジョブのスケジューリング

親トピック: DBMS_JOBからOracle Schedulerへの移行

B ブロックチェーン表参照

行内容を使用して、行のハッシュ値および署名を独立して検証できます。

行内容と列内容のデータ形式を使用して、行のハッシュ値とユーザー署名を検証するプロシージャまたは関数を作成します。

- ブロックチェーン表の列内容
  行の列内容は、列メタデータおよび列データで構成されます。
- ブロックチェーン表の行内容
  事前定義された形式を使用して、ブロックチェーン表の行の行内容を計算します。
- ブロックチェーン表の署名ダイジェストの形式
  署名ダイジェストは、ブロックチェーン表の各チェーンの最後の行に関するメタデータとデータで構成されます。

親トピック: 付録

B.1 ブロックチェーン表の列内容

行の列内容は、列メタデータおよび列データで構成されます。

列内容のデータ形式は、事前定義された形式のバイトのプラットフォームに依存しないシーケンスで、列メタデータおよび列データに基づいています。列コンテンツのデータ形式の計算方法を理解するには、bctab表の行を検討します。この表には、bankとamountの2つの列があります。この表の行には値「Chase」と1000がそれぞれ含まれます。

列データのデータ形式

データ形式は、ユーザー定義列と非表示列の両方について計算されます。列データはプラットフォームに依存せず、SQL DUMP関数を使用して取得できます。

次の例では、データベース文字セットがANSI ASCIIであると仮定し、列データのデータ形式を取得します。

```
SELECT REGEXP_REPLACE(REGEXP_SUBSTR(DUMP(bank_name, 16), '[^ ]+',1, 3), ',', '')
"Data Value"
    FROM examples.bank_ledger WHERE bank_name = 'MyBank';
Data Value
--------------------------------------------------------------------------------
4368617365
```

列メタデータのデータ形式

列メタデータは、次の形式の20バイトの構造です。

```
typedef struct col_meta_data
{
     ub2  version;                 /* VALUE IS ALWAYS 1/
     ub2  col_position;
     ub2  col_type;
     ub1  is_col_null;
     ub1  reserved1;               /*VALUE IS ALWAYS 0*/
     ub8  col_len;
     ub4  spare;
} col_meta_data;
```

ここで:

- ub1は、符号なしのシングルバイト値です
- ub2は、符号なしshort型で2バイトの値です

- ub4は、符号なしlong型で4バイトの値です
- ub8は、符号なしlong long型で8バイトの値です

ub2、ub4およびub8は、リトル・エンディアン形式を使用します。

col_meta_data構造の属性では、表内の位置、データ型、列値がNULLかどうか、および列データの長さ(バイト)など、列に関する情報を記述します。

この例では、ASCIIの「Chase」値の列メタデータを、前述の構造を使用して表します。

```
01 00
01 00
01 00
00
00
05 00 00 00 00 00 00 00
00 00 00 00
```

最初の行01 00は、データ形式バージョンの2バイト表現です(このリリースでは1)。2行目の01 00は、BANK_NAMEの列位置の2バイト表現(1)です。3行目の01 00は、BANK_NAME列のデータ型の内部コードの2バイト表現です。データ型はVARCHAR2、内部コードは1です。4行目の00には、列の値がNULL (01)かNULLでない(00)かを指定するバイトがあります。列値は「Chase」です。そのためNULLではありません。5行目の00は予約済バイトであり、このリリースではゼロである必要があります。6行目は05 00 00 00 00 00 00 00です。8バイトの長さです。値「Chase」はデータベース文字セットで5バイトです。7行目の00 00 00 00は4つの予約済バイトであり、このリリースでは、それぞれがゼロである必要があります。

列値「Chase」の列内容のデータ書式は、その列メタデータおよび列データからのバイトを連結したものです。したがって、列内容のデータ形式は次の値になります。

```
01 00
01 00
01 00
00
00
05 00 00 00 00 00 00 00
00 00 00 00
43 68 61 73 65
```

例B-1 列メタデータ値を構成するためのユーザー定義ファンクションの作成

プロシージャlittle_endian_ubxは、便利なユーティリティ・プロシージャです。数値をリトル・エンディアン形式のub2、ub4またはub8値に変換します。パラメータxは、列のデータ型を表します。ub2入力には値2を、ub4入力には4を、ub8入力には8を使用します。

```
CREATE OR REPLACE FUNCTION little_endian_ubx(value IN NUMBER, x IN NUMBER) RETURN RAW
IS
  format VARCHAR2(16);
  string VARCHAR2(16);
  result RAW(8);
BEGIN
  format := RPAD('0', 2*x-1, '0') || 'X';       -- conversion format is all zeroes
and a final X
  string := SUBSTR(TO_CHAR(value, format), 2);  -- use SUBSTR to strip leading space
  dbms_output.put_line('string is ' || string);
  result := utl_raw.reverse(HEXTORAW(string));
  dbms_output.put_line('result is ' || RAWTOHEX(result));
  RETURN result;
END little_endian_ubx;
```

親トピック: ブロックチェーン表参照

B.2 ブロックチェーン表の行内容

事前定義された形式を使用して、ブロックチェーン表の行の行内容を計算します。

行の行内容は、行データとチェーン内の前の行のハッシュ値に基づく、バイトの連続したシーケンスです。行内容のデータ形式は、バイトの順序とシーケンスを定義します。

ハッシュ値を計算するときと、行の署名を計算するときでは、行内容のデータ形式が異なります。

ハッシュ値を計算するときの行内容のデータ形式

ハッシュ値の行内容のデータ形式は、ユーザー列、一部の非表示列、およびチェーン内の前の行のハッシュ値に基づいて計算されます。

ハッシュ値を計算するときに行内容のデータ形式を理解するには、bankとamountという2列を含むブロックチェーン表bctabを考えます。この表の2行目には値「Chase」と1000がそれぞれ含まれます。行のハッシュ値を計算する際、行内容のデータ形式は、次に示すバイト値をこの順序で連結することで取得されます。

- 「Chase」値が含まれるbank列のデータ形式
- 値1000が含まれるamount列のデータ形式
- 非表示列ORABCTAB_INST_ID$のデータ形式
- 非表示列ORABCTAB_CHAIN_ID$のデータ形式
- 非表示列ORABCTAB_SEQ_NUM$のデータ形式
- 非表示列ORABCTAB_CREATION_TIME$のデータ形式
- 非表示列ORABCTAB_USER_NUMBER$のデータ形式
- 表の最初の行のハッシュ値のデータ形式(両方の行が同じチェーンにあると仮定した場合)

列の順序は、ビューALL_TAB_COLSのINTERNAL_COLUMN_ID列によって決定されます。

行の署名を計算するときの行内容のデータ形式

行署名の行内容のデータ形式は、行のハッシュ値に基づいて計算されます。

親トピック: ブロックチェーン表参照

B.3 ブロックチェーン表の署名ダイジェストの形式

署名ダイジェストは、ブロックチェーン表の各チェーンの最後の行に関するメタデータとデータで構成されます。

署名ダイジェストのデータ形式には、ヘッダーと行情報の配列が含まれます。

ヘッダーの構造は、次のとおりです。

```
{
    ub1    version;
                /* 1 in 19.x */
    ub1    reserved_1;
    ub1    reserved_2;
    ub1    reserved_3;
    ub4    reserved_4;
    ub8    total_length;
                /* total length of signature content buffer, except version,
reserved_% and total_length fields */
    ub1    pdb_guid[16];
                /* 16 bytes long PDB GUID */
    ub4    owner_schema_objn;
    ub4    blockchain_table_objn;
    ub4    signature_algorithm;
```

```
    ub4    number_of_rows;
 }
```

行情報の構造は、次のとおりです。

```
{
    ub4   instance_id ;
    ub4   chain_id
    ub8   sequence_number;
    ub4   user_number;
    ub1   row_creation_time[16];
              /* UTC format that Oracle uses has 13 bytes; padded 3 bytes */
    ub4   crypto_hash_len;
    ub1  *crypto_hash;
              /* padded to 4 byte boundary */
    ub4   user_columns_count;
              /* always 0 in 19.x
               * padded to 8 byte boundary */
    ub8   user_columns_data_len;
              /* always 0 in 19.x */
}
```

ここで:

- ub1は、符号なしのシングルバイト値です
- ub4は、符号なしlong型で4バイトの値です
- ub8は、符号なしlong型で8バイトの値です

ub4およびub8は、リトル・エンディアン形式を使用します。

パディングは、2進ゼロを追加することによって行われます。

親トピック: ブロックチェーン表参照

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z