Oracle® Data Guard

概要および管理

11*g* リリース 1（11.1）

部品番号 : E05755-03

2008 年 11 月

ORACLE®

Oracle Data Guard 概要および管理 , 11*g* リリース 1（11.1）

部品番号 : E05755-03

Oracle Data Guard Concepts and Administration, 11*g* Release 1 (11.1)

原本部品番号 : B28294-03

原本著者 : Kathy Rich

原本協力者 : Andy Adams, Beldalker Anand, Rick Anderson, Andrew Babb, Pam Bantis, Tammy Bednar, Barbara Benton, Chipper Brown, Larry Carpenter, George Claborn, Laurence Clarke, Jay Davison, Jeff Detjen, Ray Dutcher, B.G. Garin, Mahesh Girkar, Yosuke Goto, Ray Guzman, Susan Hillson, Mark Johnson, Rajeev Jain, Joydip Kundu, J. William Lee, Steve Lee, Steve Lim, Nitin Karkhanis, Steve McGee, Bob McGuirk, Joe Meeks, Steve Moriarty, Muthu Olagappan, Deborah Owens, Ashish Ray, Antonio Romero, Mike Schloss, Vivian Schupmann, Mike Smith, Vinay Srihali, Morris Tao, Lawrence To, Doug Utzig, Ric Van Dyke, Doug Voss, Ron Weiss, Jingming Zhang

目次

7 適用サービス

8 ロールの推移

9 フィジカルおよびスナップショット・スタンバイ・データベースの管理

10 ロジカル・スタンバイ・データベースの管理

11 Recovery Manager を使用したファイルのバックアップおよびリストア

12 SQL Apply を使用した Oracle Database のアップグレード

13 Data Guard の使用例

第 III 部　付録

A　Data Guard のトラブルシューティング

B　Data Guard 構成におけるデータベースのアップグレード

C ロジカル・スタンバイ・データベースでサポートされるデータ型および DDL

D Data Guard および Oralce Real Application Clusters

E カスケードされた宛先

F Recovery Manager を使用したスタンバイ・データベースの作成

G アーカイブ・トレースの設定

索引

例一覧

図一覧

表一覧

はじめに

Oracle Data Guard は、現在使用できる最も効率的なソリューションで、企業の中核となる資産、つまりデータを保護し、障害やその他の災害が発生しても、24 時間 365 日そのデータを使用できるようにします。このマニュアルでは、Oracle Data Guard テクノロジとその概要、スタンバイ・データベースの構成および実装方法について説明します。

対象読者

『Oracle Data Guard 概要および管理』は、Oracle データベース・システムのバックアップ、リストアおよびリカバリ操作を管理するデータベース管理者（DBA）を対象としています。

このマニュアルは、読者がリレーショナル・データベースの概念と、基本的なバックアップおよびリカバリ管理に精通していることを前提としています。また、Oracle ソフトウェアを実行するオペレーティング・システム環境をよく理解している必要があります。

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様にもオラクル社の製品、サービスおよびサポート・ドキュメントを簡単にご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト `http://www.oracle.com/accessibility/` を参照してください。

ドキュメント内のサンプル・コードのアクセシビリティについて

スクリーン・リーダーは、ドキュメント内のサンプル・コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかし JAWS は括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

Oracle サポート・サービスへの TTY アクセス

アメリカ国内では、Oracle サポート・サービスへ 24 時間年中無休でテキスト電話（TTY）アクセスが提供されています。TTY サポートについては、(800)446-2398 にお電話ください。アメリカ国外からの場合は、+1-407-458-2479 にお電話ください。

関連ドキュメント

『Oracle Data Guard 概要および管理』の読者は、次のマニュアルをすでに読んでいることが前提となります。

- 『Oracle Database 概要』の冒頭部分。Oracle データベースに関連する概念と用語の概要が説明されており、このマニュアルに記載されている詳細情報の基礎となっています。
- 『Oracle Database 管理者ガイド』の、制御ファイル、オンライン REDO ログ・ファイルおよびアーカイブ REDO ログ・ファイルの管理について説明している章。
- 『Oracle Database ユーティリティ』の、LogMiner テクノロジについて説明している章。
- 『Oracle Data Guard Broker』。Data Guard 構成の作成、メンテナンス、監視を自動化および集中管理化するグラフィカル・ユーザー・インタフェースとコマンドライン・インタフェースについて説明しています。
- Oracle Enterprise Manager のオンライン・ヘルプ・システム。

このマニュアルの説明では、次の各マニュアルも取り上げています。

- 『Oracle Database SQL 言語リファレンス』
- 『Oracle Database リファレンス』
- 『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』
- 『Oracle Database Net Services 管理者ガイド』
- 『SQL*Plus ユーザーズ・ガイドおよびリファレンス』
- 『Oracle Database 高可用性概要』

Oracle Streams および Streams のダウンストリーム取得データベースの詳細は、『Oracle Streams 概要および管理』も参照してください。Streams のダウンストリーム取得プロセスは、Oracle Data Guard の REDO 転送サービスを使用して、REDO データをリモート・データベース上のログ・ファイルに転送します。リモート・データベースでは、Streams の取得プロセスにより、リモート宛先でのアーカイブ REDO ログ・ファイルの変更点が取得されます。

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

サポートおよびサービス

次の各項に、各サービスに接続するための URL を記載します。

Oracle サポート・サービス

オラクル製品サポートの購入方法、および Oracle サポート・サービスへの連絡方法の詳細は、次の URL を参照してください。

```
http://www.oracle.com/lang/jp/support/index.html
```

製品マニュアル

製品のマニュアルは、次の URL にあります。

```
http://www.oracle.com/technology/global/jp/documentation/index.html
```

研修およびトレーニング

研修に関する情報とスケジュールは、次の URL で入手できます。

```
http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=3
```

その他の情報

オラクル製品やサービスに関するその他の情報については、次の URL から参照してください。

```
http://www.oracle.com/lang/jp/index.html
http://www.oracle.com/technology/global/jp/index.html
```

注意： ドキュメント内に記載されている URL や参照ドキュメントには、Oracle Corporation が提供する英語の情報も含まれています。日本語版の情報については、前述の URL を参照してください。

Oracle Data Guard の新機能

ここでは、Oracle Data Guard 11*g* リリース 1（11.1）に追加された新機能について説明するとともに、追加情報の記載場所も示します。Oracle Data Guard 11*g* リリース 1（11.1）には、ここで説明する新機能と拡張機能が追加されました。新機能を、次の主な内容にわけて説明します。

- REDO Apply と SQL Apply に共通する新機能
- REDO Apply およびフィジカル・スタンバイ・データベース固有の新機能
- SQL Apply およびロジカル・スタンバイ・データベース固有の新機能

REDO Apply と SQL Apply に共通する新機能

Oracle Data Guard 11*g* リリース 1（11.1）に対する次の拡張機能により、利便性、管理性およびパフォーマンスが向上し、障害時リカバリ機能を改善する新機能が追加されています。

- **Data Guard 構成でのネットワークを介した REDO トラフィックの圧縮**

  この機能により、REDO を圧縮してからネットワーク上に転送して REDO ギャップを解決すると、REDO 転送のパフォーマンスが向上します。

  関連項目： 15-5 ページの「COMPRESSION」属性

- **REDO 転送のレスポンス時間のヒストグラム**

  `V$REDO_DEST_RESP_HISTOGRAM` 動的パフォーマンス・ビューには、`SYNC` REDO 転送先ごとのレスポンス時間のヒストグラムが表示されます。このビューのデータを使用して、`LOG_ARCHIVE_DEST_`*n* NET_TIMEOUT 属性の適正値を簡単に決定することができます。

  関連項目： 15-14 ページの「NET_TIMEOUT」属性

- **ロールの推移の高速化**
- **REDO 転送ネットワーク・セッションの厳密認証**

  REDO 転送ネットワーク・セッションは、現在、SSL を使用して認証できます。これにより、厳密認証が行われ、Data Guard 構成でのリモート・ログイン・パスワード・ファイルの使用はオプションになります。

- **簡略化された Data Guard 管理インタフェース**

  Data Gurad 構成の管理に使用される SQL 文および初期化パラメータが、冗長な SQL 句および初期化パラメータが廃止することで簡略化されています。

関連項目：

- 廃止された句が含まれる文の詳細は、第 16 章「Data Guard に関連する SQL 文」を参照してください。
- 第 16 章で説明している SQL 文に関連する廃止された句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
- LOG_ARCHIVE_DEST_*n* パラメータの廃止された属性の詳細は、『Oracle Database リファレンス』を参照してください。

- **DB_UNIQUE_NAME に関連する拡張機能**

  現在、プライマリ・データベースの DB_UNIQUE_NAME は、V$DATABASE ビューの新しい PRIMARY_DB_UNIQUE_NAME 列を問い合せてスタンバイ・データベースから検出できます。また、Oracle Data Guard 11*g* リリース 1（11.1）では、各データベースの DB_UNIQUE_NAME が必ず異なります。11*g* にアップグレードすると、同じ DB_UNIQUE_NAME を持つデータベースはすべて相互に通信できません。

- **ローリング・アップグレードでのフィジカル・スタンバイ・データベースの使用**

  現在、フィジカル・スタンバイ・データベースでは、ロジカル・スタンバイによって提供されるローリング・アップグレード機能を利用できます。新しい KEEP IDENTITY 句オプションを SQL ALTER DATABASE RECOVER TO LOGICAL STANDBY 文に使用して、フィジカル・スタンバイ・データベースをローリング・アップグレードのために一時的にロジカル・スタンバイ・データベースに変換し、アップグレード後に、元の構成のプライマリ・データベースおよびフィジカル・スタンバイ・データベースに戻すことができます。

  関連項目： 第 12 章「SQL Apply を使用した Oracle Database のアップグレード」

- **異機種間の Data Guard 構成**

  この機能を使用すると、同じ Data Guard 構成でプライマリ・データベースとスタンバイ・データベースに Linux と Windows を組み合せることができます。

  注意： 強化されたブローカベースの管理フレームワーク（次の項目も含む）の詳細は、『Oracle Data Guard Broker』の新機能に関する項も参照してください。

  - Data Guard 構成の最大パフォーマンス・モードに対するファスト・スタート・フェイルオーバー
  - Data Guard 構成のファスト・スタート・フェイルオーバーをトリガーするユーザーの設定可能なイベント

- **ARCH REDO 転送モードの廃止**

  ARCH REDO 転送モードは廃止され、将来のリリースではサポートされません。ARCH 転送モードを現在使用している場合、ASYNC 転送モードに切り替えることをお薦めします。ASYNC 転送モードは、あらゆる面で ARCH 転送モードより優れた新しいデフォルトの転送モードです。

REDO Apply およびフィジカル・スタンバイ・データベース固有の新機能

次のリストに、Oracle Database 11*g* リリース 1（11.1）の REDO Apply およびフィジカル・スタンバイ・データベース固有の新機能を示します。

- **フィジカル・スタンバイのリアルタイム問合せ機能**

  この機能により、REDO Apply がアクティブである間、フィジカル・スタンバイ・データベースを問い合せることができます。

  関連項目： オープン・フィジカル・スタンバイ・データベースでプライマリ・データベースからの REDO データの受信および適用を続行する方法の詳細は、9-2 ページの 9.2 項「フィジカル・スタンバイ・データベースのオープン」を参照してください。

- **スナップショット・スタンバイ**

  スナップショット・スタンバイ・データベースは、新タイプの更新可能なスタンバイ・データベースで、プライマリ・データベースに完全なデータ保護を提供します。

  関連項目： 9.7 項「スナップショット・スタンバイ・データベースの管理」

- **フィジカル・スタンバイを使用した書込みの欠落の検出**

  書込みの欠落は、データベースに悪影響を及ぼす可能性がある重大なデータの破損です。実際には書込みが永続記憶域で発生しなかったのに対して、I/O サブシステムがデータベースのブロック書込みの完了を確認した場合に発生します。この機能を使用すると、プライマリ・データベースまたはフィジカル・スタンバイ・データベースに対する書込みの欠落をフィジカル・スタンバイ・データベースで検出できます。

  関連項目： 書込みの欠落のリカバリ例は 13.6 項「プライマリ・データベースでの書込みの欠落エラーからのリカバリ」を、書込みの欠落を検出できるようにする方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

- **改善された Recovery Manager との統合**

  Recovery Manager の多くの拡張機能により、カタログの使用時に、すべてのプライマリ・データベースおよびフィジカル・スタンバイ・データベース間のバックアップおよびリカバリの操作が簡略化されました。また、既存のデータベース・バックアップがなくても、Recovery Manager の `DUPLICATE` コマンドを使用して、ネットワーク上にフィジカル・スタンバイ・データベースを作成できます。

  関連項目：

  - 3.2 項「フィジカル・スタンバイ・データベースの作成手順」
  - 第 11 章「Recovery Manager を使用したファイルのバックアップおよびリストア」
  - 付録 F「Recovery Manager を使用したスタンバイ・データベースの作成」

SQL Apply およびロジカル・スタンバイ・データベース固有の新機能

次のリストに、Oracle Database 11*g* リリース 1（11.1）の SQL Apply およびロジカル・スタンバイ・データベースの新機能を示します。

- **追加のオブジェクト・データ型および PL/SQL パッケージのサポート**
  - `CLOB` として格納される XML

  関連項目： 付録 C「ロジカル・スタンバイ・データベースでサポートされるデータ型および DDL」

- **追加の PL/SQL パッケージのサポート**
  - DBMS_RLS（行レベル・セキュリティまたは仮想プライベート・データベース）
  - DBMS_FGA

    関連項目： 付録 C「ロジカル・スタンバイ・データベースでサポートされるデータ型および DDL」

- **透過的データ暗号化（TDE）のサポート**

  Data Guard SQL Apply を使用すると、透過的データ暗号化を有効に設定したプライマリ・データベースにデータ保護を提供できます。これにより、ロジカル・スタンバイ・データベースで、高度なセキュリティ要件を伴うアプリケーションにデータ保護を提供できます。

  関連項目：

  - 第 10 章「ロジカル・スタンバイ・データベースの管理」
  - C.2 項「透過的データ暗号化（TDE）のサポート」

- **Data Guard SQL Apply のパラメータの動的設定**

  現在、SQL Apply を再開しなくても特定の SQL Apply のパラメータを構成できます。DBMS_LOGSTDBY.APPLY_SET パッケージを使用して、動的に初期化パラメータを設定できるため、ロジカル・スタンバイ構成の管理性、稼働時間および自動化が向上します。

  さらに、APPLY_SET および APPLY_UNSET サブプログラムには、LOG_AUTO_DEL_RETENTION_TARGET および EVENT_LOG_DEST の 2 つの新しいパラメータが組み込まれています。

  関連項目： 『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_LOGSTDBY PL/SQL パッケージに関する項

- **ロジカル・スタンバイ・データベースに対して強化された RAC スイッチオーバーのサポート**

  ロジカル・スタンバイ・データベースにスイッチオーバーするとき、プライマリ・データベースまたはスタンバイ・データベースのいずれかが Oracle RAC を使用している場合は、プライマリ・データベースまたはロジカル・スタンバイ・データベースのいずれかで、インスタンスを一切停止せずに SWITCHOVER コマンドを使用できます。

- **Oracle Data Guard SQL Apply での強化された DDL の処理**

  SQL Apply では、パラレル DDL を（パラレル・サーバーの可用性に基づいて）パラレルで実行します。

- **PL/SQL DBMS_SCHEDULER パッケージを使用したスタンバイ・データベースでのスケジューラ・ジョブの作成**

  スケジューラ・ジョブは、PL/SQL DBMS_SCHEDULER パッケージを使用してスタンバイ・データベースで作成でき、適切なデータベース・ロールを関連付けて、予定された時期（たとえば、データベースがプライマリまたはスタンバイ、あるいはその両方であるとき）に実行されるようにできます。

第I部

概要および管理

この部は、次の章で構成されています。

1

Oracle Data Guard の概要

Oracle Data Guard では、企業データの高可用性、データ保護および障害時リカバリを保証します。Data Guard は、1 つ以上のスタンバイ・データベースの作成、メンテナンス、管理および監視など、一連の包括的なサービスを提供し、本番の Oracle データベースを障害およびデータ破損から保護します。Data Guard では、これらのスタンバイ・データベースを本番データベースのコピーとしてメンテナンスします。したがって、本番データベースが計画的または計画外の停止によって使用不可能になった場合は、スタンバイ・データベースを本番ロールに切り替えて、停止時間を最小限にできます。Data Guard を従来のバックアップ、リストアおよびクラスタ化の技法と連携して使用すると、高いレベルのデータ保護とデータ可用性を実現できます。

Data Guard を使用すると、リソース集中型のバックアップおよびレポート生成操作をスタンバイ・システムにオフロードすることで、管理者は本番データベースのパフォーマンスをオプションで改善できます。

この章では、Oracle Data Guard の概要を説明します。次の項目で構成されています。

1.1 Data Guard 構成

Data Guard 構成には、1 つの本番データベースと 1 つ以上のスタンバイ・データベースが含まれます。Data Guard 構成のデータベースは、Oracle Net で接続され、地理的に分散している場合があります。データベースの配置場所に制限はありませんが、相互に通信できる必要があります。たとえば、1 個のスタンバイ・データベースを本番データベースと同じシステム上に配置し、2 個のスタンバイ・データベースをリモート位置の別のシステム上に配置できます。

プライマリ・データベースとスタンバイ・データベースは、SQL コマンドライン・インタフェースまたは Data Guard Broker インタフェースを使用して管理できます。Data Guard Broker インタフェースには、コマンドライン・インタフェース（DGMGRL）や、Oracle Enterprise Manager に統合されたグラフィカル・ユーザー・インタフェースなどがあります。

1.1.1 プライマリ・データベース

Data Guard 構成には、1 個の本番データベースが含まれています。これはプライマリ・データベースとも呼ばれ、プライマリ・ロールで機能します。アプリケーションは主としてこのデータベースにアクセスします。

プライマリ・データベースは、単一インスタンスの Oracle データベースまたは Oracle Real Application Clusters（RAC）データベースのいずれかです。

1.1.2 スタンバイ・データベース

スタンバイ・データベースは、プライマリ・データベースのトランザクション一貫性のあるコピーです。プライマリ・データベースのバックアップ・コピーを使用すると、最大 9 個のスタンバイ・データベースを作成して、Data Guard 構成に組み込むことができます。スタンバイ・データベースが作成されると、Data Guard では、プライマリ・データベースからスタンバイ・データベースに REDO データを転送し、スタンバイ・データベースに REDO を適用することによって、各スタンバイ・データベースを自動的にメンテナンスします。

プライマリ・データベースと同様、スタンバイ・データベースは、単一インスタンスの Oracle データベースまたは Oracle RAC データベースのいずれかです。

スタンバイ・データベースのタイプは、次のとおりです。

- **フィジカル・スタンバイ・データベース**

  プライマリ・データベースと物理的に同一になるようコピーしたものです。ディスク上のデータベース構造は、ブロック単位でプライマリ・データベースと同一です。索引などのデータベース・スキーマも同一です。フィジカル・スタンバイ・データベースでは、プライマリ・データベースから受信した REDO データをリカバリし、REDO をフィジカル・スタンバイ・データベースに適用する **Redo Apply** によって、プライマリ・データベースとの同期を維持します。

  Oracle Database 11*g* リリース 1（11.1）現在、フィジカル・スタンバイ・データベースは、読取り専用アクセス用にオープンしている間は REDO を受信および適用できます。そのため、フィジカル・スタンバイ・データベースは、データ保護とレポート生成に同時に使用できます。

- **ロジカル・スタンバイ・データベース**

  本番データベースと同じ論理情報が格納されますが、データの物理的な構成と構造は異なる場合があります。ロジカル・スタンバイ・データベースでは、プライマリ・データベースから受信した REDO データを SQL 文に変換し、スタンバイ・データベースでその SQL 文を実行する **SQL Apply** によって、プライマリ・データベースとの同期を維持します。

  ロジカル・スタンバイ・データベースは、障害時リカバリ要件に加えて、その他のビジネス用途にも使用できます。このため、ロジカル・スタンバイ・データベースには、問合せやレポート生成の目的でいつでもアクセスできます。また、ロジカル・スタンバイ・データベースを使用すると、ほとんどの場合、データベースを停止することなく、Oracle Database ソフトウェアやパッチ・セットをアップグレードできます。つまり、ロジカル・スタンバイ・データベースは、データ保護、レポート生成およびデータベースのアップグレードに同時に使用できます。

- **スナップショット・スタンバイ・データベース**

  スナップショット・スタンバイ・データベースは、全面的に更新可能なスタンバイ・データベースで、フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースに変換して作成します。

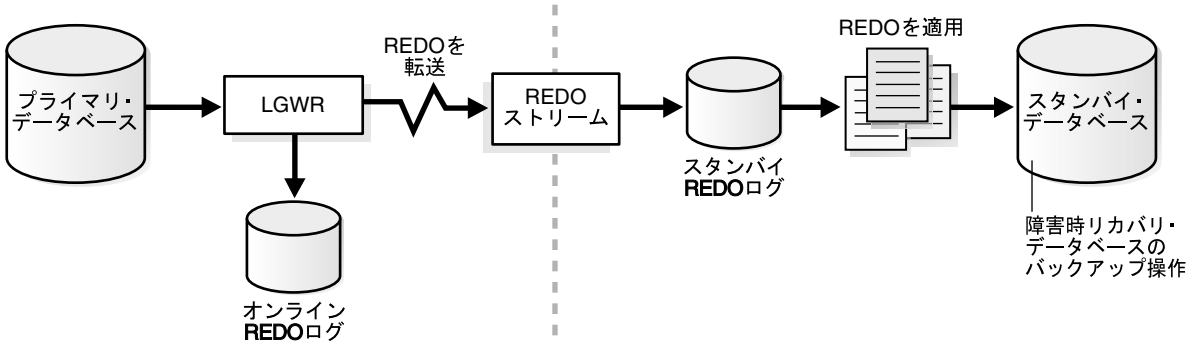
  フィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースと同様、スナップショット・スタンバイ・データベースは、プライマリ・データベースから REDO データを受信およびアーカイブします。しかし、フィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースとは異なり、スナップショット・スタンバイ・データベースは、受信した REDO データを適用しません。スナップショット・スタンバイ・データベースで受信された REDO データは、スナップショット・スタンバイ・データベースに対して行われたローカル更新がすべて破棄された後、スナップショット・スタンバイが変換されてフィジカル・スタンバイ・データベースに戻るまで適用されません。

  スナップショット・スタンバイ・データベースは、フィジカル・スタンバイ・データベースの一時的かつ更新可能なスナップショットが必要な場合に最もよく使用されます。スナップショット・スタンバイ・データベースで受信された REDO データは、フィジカル・スタンバイに変換されて戻るまで適用されないため、プライマリ・データベースの障害からリカバリするのに必要な時間は、適用する必要がある REDO データの量に正比例します。

1.1.3 構成例

図 1-1 は、REDO データをスタンバイ・データベースに転送するプライマリ・データベースが含まれている一般的な Data Guard 構成を示しています。スタンバイ・データベースは、障害時リカバリ操作とバックアップ操作に備えて、プライマリ・データベースから離れた位置に配置されます。スタンバイ・データベースはプライマリ・データベースと同じ位置にも構成できます。ただし、障害時リカバリに使用する場合は、スタンバイ・データベースを離れた位置に構成することをお薦めします。

図 1-1 一般的な Data Guard 構成



1.2 Data Guard サービス

次の各項では、Data Guard による REDO データの転送、REDO データの適用およびデータベース・ロールの変更の管理方法について説明します。

- REDO 転送サービス

  本番データベースから 1 つ以上のアーカイブ先に対する REDO データの自動転送を制御します。

- 適用サービス

  REDO データをスタンバイ・データベースに適用し、プライマリ・データベースとのトランザクションの同期を維持します。REDO データは、アーカイブ REDO ログ・ファイルから適用できます。また、リアルタイム適用が可能な場合は、スタンバイ・データベースで最初に REDO データをアーカイブしなくても、いっぱいになったときにスタンバイ REDO ログ・ファイルから直接適用することもできます。

- ロールの推移

  データベースのロールを、スイッチオーバー操作またはフェイルオーバー操作を使用して、スタンバイ・データベースからプライマリ・データベースに、またはプライマリ・データベースからスタンバイ・データベースに変更します。

1.2.1 REDO 転送サービス

REDO 転送サービスは、本番データベースから 1 つ以上のアーカイブ先に対する REDO データの自動転送を制御します。

REDO 転送サービスでは、次のタスクを実行します。

- REDO データを構成内のプライマリ・システムからスタンバイ・システムに転送します。
- ネットワーク障害により発生したアーカイブ REDO ログ・ファイル内のギャップを解決するプロセスを管理します。
- スタンバイ・システム上の欠落または破損しているアーカイブ REDO ログ・ファイルを自動的に検出し、プライマリ・データベースまたは別のスタンバイ・データベースからかわりのアーカイブ REDO ログ・ファイルを自動的に取得します。

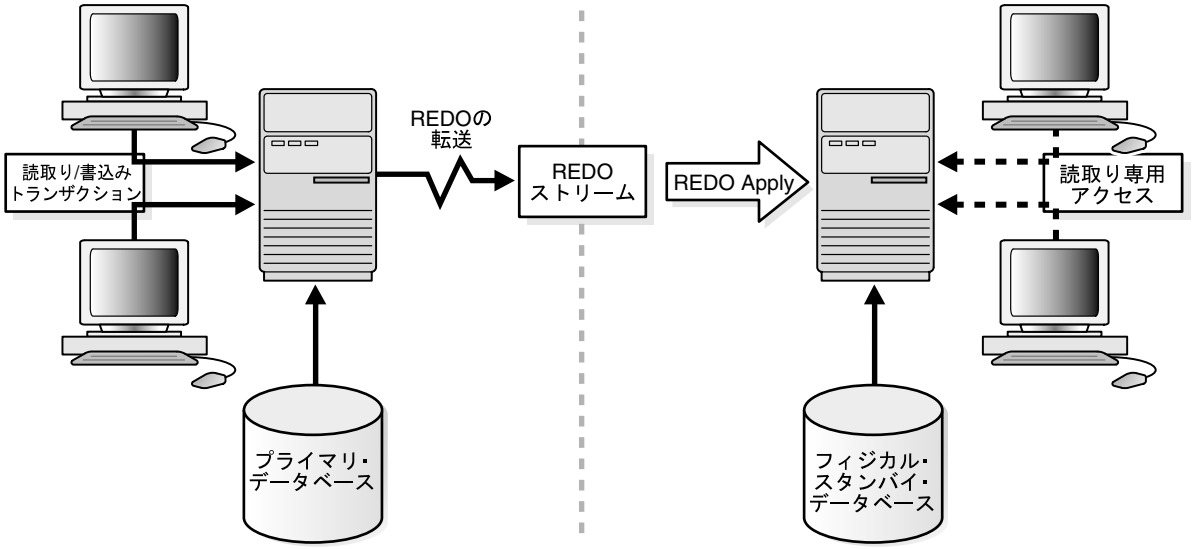
1.2.2 適用サービス

プライマリ・データベースから転送された REDO データは、スタンバイ・データベース上でスタンバイ REDO ログに書き込まれます。**適用サービス**は、REDO データをスタンバイ・データベースに自動的に適用し、プライマリ・データベースとの一貫性を維持します。データを読取り専用にすることも可能です。

フィジカル・スタンバイ・データベースとロジカル・スタンバイ・データベースの主な相違点は、適用サービスによるアーカイブ REDO データの適用方法にあります。

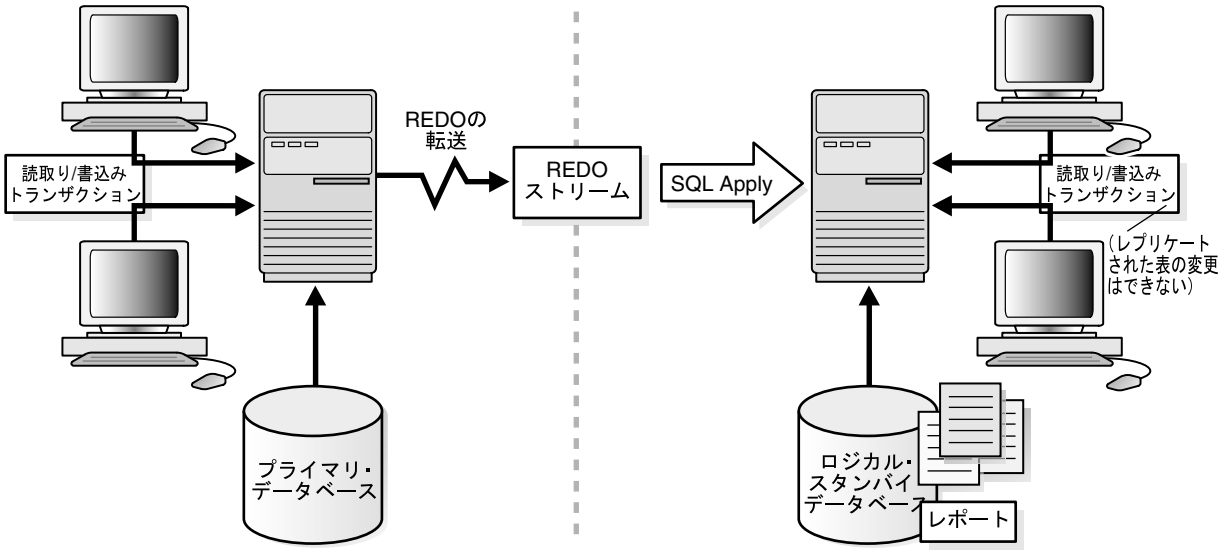
- フィジカル・スタンバイ・データベースの場合、Data Guard では **REDO Apply** テクノロジを使用します。このテクノロジでは、図 1-2 に示すように、Oracle データベースの標準リカバリ技法を使用して REDO データがスタンバイ・データベースに適用されます。

図 1-2 フィジカル・スタンバイ・データベースの自動更新



- ロジカル・スタンバイ・データベースの場合、Data Guard では **SQL Apply** テクノロジを使用します。このテクノロジでは、図 1-3 に示すように、まず、受信した REDO データが SQL 文に変換されてから、生成された SQL 文がロジカル・スタンバイ・データベース上で実行されます。

図 1-3 ロジカル・スタンバイ・データベースの自動更新



1.2.3 ロールの推移

Oracle データベースは、プライマリ・ロールまたはスタンバイ・ロールのいずれかで実行されます。Data Guard では、スイッチオーバー操作またはフェイルオーバー操作のいずれかを使用してデータベースのロールを変更できます。

スイッチオーバーとは、プライマリ・データベースとそのスタンバイ・データベースの 1 つとの間でロールを可逆的に推移させる操作です。スイッチオーバーにより、データ消失のない状態が保証されます。この操作は通常、プライマリ・システムの計画的なメンテナンスに対して実行します。スイッチオーバー時には、プライマリ・データベースがスタンバイ・ロールに、あるいはスタンバイ・データベースがプライマリ・ロールに推移します。

フェイルオーバーは、プライマリ・データベースが使用不可能な場合に発生します。フェイルオーバーは、プライマリ・データベースで障害が発生したときのみに実行され、スタンバイ・データベースをプライマリ・ロールに推移させます。データベース管理者は、データが消失しないように、Data Guard を構成できます。

このマニュアルで説明しているロールの推移は、SQL 文を使用して手動で行います。また、1.3 項で説明しているように、Oracle Data Guard Broker を使用してロールの推移を単純化させ、Oracle Enterprise Manager または DGMGRL コマンドライン・インタフェースを使用してフェイルオーバーを自動化することもできます。

1.3 Data Guard Broker

Data Guard Broker は分散管理フレームワークで、Data Guard 構成の作成、メンテナンスおよび監視を自動化します。Oracle Enterprise Manager のグラフィカル・ユーザー・インタフェース（GUI）または Data Guard コマンドライン・インタフェース（DGMGRL）を使用すると、次の操作が自動化および単純化されます。

- REDO 転送サービスや適用サービスの設定など、Data Guard 構成の作成と有効化。
- 構成内の任意のシステムによる Data Guard 構成全体の管理。
- Oracle RAC のプライマリ・データベースまたはスタンバイ・データベースが含まれる Data Guard 構成の管理と監視。
- Oracle Enterprise Manager でキー・クリックを 1 回する、あるいは DGMGRL コマンドライン・インタフェースで単一のコマンドを使用すると起動できるようにする、スイッチオーバーおよびフェイルオーバーの単純化。
- プライマリ・データベースが使用不可能になった場合に、自動的にフェイルオーバーを起動するための、ファスト・スタート・フェイルオーバーの有効化。ファスト・スタート・フェイルオーバーが有効になると、Data Guard Broker によりフェイルオーバーが必要かどうか決定され、指定のターゲット・スタンバイ・データベースに対して自動的にフェイルオーバーが開始されます。DBA による操作の必要はありません。

さらに、Oracle Enterprise Manager を使用すると、次の操作が自動化および単純化されます。

- プライマリ・データベースのバックアップ・コピーからのフィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースの作成
- Data Guard 構成への新規または既存のスタンバイ・データベースの追加
- ログ適用レートの監視、診断情報の獲得、および集中化したモニタリング・ツール、テスト・ツール、パフォーマンス・ツールなどを使用した問題の早期検出

関連項目： 詳細は、『Oracle Data Guard Broker』を参照してください。

1.3.1 Oracle Enterprise Manager Grid Control の使用

Oracle Enterprise Manager Grid Control（以下、このマニュアルでは Enterprise Manager と呼びます）は、Data Guard 構成内のプライマリ・データベースおよびスタンバイ・データベースを表示、監視および管理するための、Web ベースのインタフェースを提供します。Enterprise Manager の使いやすいインタフェースと、Broker による Data Guard 構成の集中化された管理および監視を組み合せることで、企業内の高可用性、サイト保護およびデータ保護を実現するための Data Guard ソリューションが強化されます。

Enterprise Manager Grid Control のセントラル・コンソールから、すべての管理操作をローカルまたはリモートで実行できます。プライマリ・データベースとスタンバイ・データベース、およびプライマリ・インスタンスとスタンバイ・インスタンスを含む、Oracle データベースのホーム・ページの表示、既存のスタンバイ・データベースの作成または追加、インスタンスの開始および停止、インスタンスのパフォーマンスの監視、イベントの表示、ジョブのスケジュール、バックアップ操作およびリカバリ操作の実行が可能です。

関連項目： Oracle Enterprise Manager Grid Control のブローカ・インタフェースの詳細は、『Oracle Data Guard Broker』を参照してください。

1.3.2 Data Guard コマンドライン・インタフェースの使用

Data Guard のコマンドライン・インタフェース（DGMGRL）を使用すると、DGMGRL プロンプトまたはスクリプト内から Data Guard 構成を制御および監視できます。DGMGRL を使用すると、構成でデータベースを管理および監視するために必要なアクティビティのほとんどを実行できます。DGMGRL のリファレンスの詳細と例は、『Oracle Data Guard Broker』を参照してください。

1.4 Data Guard の保護モード

場合によっては、状況に関係なくデータの消失が許されないビジネスがあります。また、起こりそうにない多重障害の場合に起こりうるデータの消失よりデータベースの可用性の方が重要な場合もあります。アプリケーションの中には、常に最大のデータベース・パフォーマンスが必要であるため、コンポーネントに障害が発生した場合に少量のデータの消失ならば許容できるものがあります。データ保護には、次の 3 つの異なるモードが用意されています。

最大可用性　この保護モードは、プライマリ・データベースの可用性を低下させない範囲で可能な最高レベルのデータ保護を提供します。トランザクションは、そのトランザクションのリカバリに必要なすべての REDO データが、オンライン REDO ログおよび少なくとも 1 つのスタンバイ・データベースに書き込まれるまでコミットされません。プライマリ・データベースは、REDO ストリームを少なくとも 1 つの同期化されたスタンバイ・データベースに書き込むことができない場合、REDO ストリームを同期化されたスタンバイ・データベースに再び書き込めるようになるまで、最大パフォーマンス・モードにあるかのように動作してプライマリ・データベースの可用性を維持します。

この保護モードは、特定の二重障害の場合（スタンバイ・データベースの障害が発生した後にプライマリ・データベースの障害が発生した場合など）を除いて、データ消失がないことを保証します。

最大パフォーマンス　これはデフォルトの保護モードです。この保護モードは、プライマリ・データベースのパフォーマンスに影響しない範囲で可能な最高レベルのデータ保護を提供します。これは、トランザクションによって生成されたすべての REDO データがオンライン・ログに書き込まれた直後に、そのトランザクションのコミットを可能にすることで実現されます。REDO データは、1 つ以上のスタンバイ・データベースにも書き込まれますが、トランザクション・コミットについて非同期で行われるため、プライマリ・データベースのパフォーマンスは、スタンバイ・データベースへの REDO データの書込み遅延による影響を受けません。

この保護モードは、最大可用性モードに比べてデータ保護が若干弱く、プライマリ・データベースのパフォーマンスへの影響を最小限に抑えます。

最大保護　この保護モードは、プライマリ・データベースに障害が発生した場合でも、データ消失がないことを保証します。このレベルの保護を提供するには、トランザクションがコミットされる前に、トランザクションのリカバリに必要な REDO データを、オンライン REDO ログおよび少なくとも 1 つのスタンバイ・データベースに書き込む必要があります。少なくとも 1 つのスタンバイ・データベースに REDO ストリームを書き込むことができない場合は、データ消失が発生しないように、プライマリ・データベースは停止し、トランザクションの処理を続行しません。

3 つの保護モードはいずれも、特定の REDO 転送オプションを使用して、REDO データが少なくとも 1 つのスタンバイ・データベースに送信されるようにする必要があります。プライマリ・データベースの保護モードの設定の詳細は、第 5 章「Data Guard の保護モード」を参照してください。

1.5 クライアント・フェイルオーバー

高可用性アーキテクチャには、データベースおよびデータベース・クライアントに対する高速フェイルオーバー機能が必要です。

クライアント・フェイルオーバーには、障害の通知、失効した接続のクリーンアップおよび新しいプライマリ・データベースへの透過的な再接続が含まれます。Oracle Database には、データベース・フェイルオーバーとフェイルオーバー・プロシージャを統合する機能があるため、クライアントは、データベース・フェイルオーバーから数秒以内に新しいプライマリ・データベースに自動的にリダイレクトされます。

クライアント・フェイルオーバーの詳細は、次の URL にアクセスし、Maximum Availability Architecture のクライアント・フェイルオーバーのベスト・プラクティスに関するホワイト・ペーパーを参照してください。

```
http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm
```

1.6 Data Guard と補完テクノロジ

Oracle Database では、Data Guard を補完する複数の固有のテクノロジを提供して、ビジネスに不可欠なシステムが、1 つのソリューションのみを利用する場合に比べ、はるかに高い可用性とデータ保護を実現しながら稼働できるようにします。Oracle の高可用性テクノロジには、次のものがあります。

- Oracle Real Application Clusters（RAC）

  Oracle RAC を使用すると、インターコネクトによってリンクされた複数の独立型サーバーで、Oracle データベースへのアクセスを共有し、障害の発生時に、高可用性、スケーラビリティおよび冗長性を実現できます。Oracle RAC と Data Guard を一緒に使用すると、システムレベル、サイトレベルおよびデータレベルのそれぞれの保護にメリットがあるため、データを消失することなく、高い可用性と障害時リカバリが実現します。

  - Oracle RAC は、ノード障害やインスタンスのクラッシュなどの障害から迅速かつ自動的にリカバリすることで、システムの障害に対処します。また、アプリケーションのスケーラビリティも改善されます。
  - Data Guard では、トランザクション的に一貫性のある、ディスクを共有しないプライマリ・データベースとスタンバイ・データベースを介して、サイト内の問題やデータ保護に対処し、サイトの障害やデータ破損からリカバリできるようにします。

  ローカル・サイトとリモート・サイトの使用、ロジカル・スタンバイ・データベースとフィジカル・スタンバイ・データベースの組合せとノードの使用によって、Oracle RAC および Data Guard を使用した様々なアーキテクチャを実現できます。Oralce RAC および Data Guard の統合の詳細は、付録 D「Data Guard および Oralce Real Application Clusters」および『Oracle Database 高可用性概要』を参照してください。

- フラッシュバック・データベース

  フラッシュバック・データベース機能により、論理データの破損やユーザー・エラーから迅速にリカバリできます。適切なときにフラッシュバックを許可することで、誤って変更または削除された可能性のある以前のバージョンのビジネス情報に再度アクセスできるようになります。この機能により、次のことが実現します。

  - バックアップをリストアして、エラーや破損が発生した時点までロールフォワード変更を行う必要がなくなります。かわりに、フラッシュバック・データベースでは、データファイルをリストアせずに、以前の時点まで Oracle データベースをロールバックできます。
  - REDO の適用を遅延してユーザー・エラーや論理的な破損から保護するための、代替手段を提供します。したがって、スタンバイ・データベースは、プライマリ・データベースとより密接に同期できるため、フェイルオーバーおよびスイッチオーバーの回数が少なくなります。

  - フェイルオーバー後に、元のプライマリ・データベースを完全に作成しなおす必要がなくなります。障害が発生したプライマリ・データベースを、フェイルオーバー前の時点にフラッシュ・バックして、新しいプライマリ・データベースのスタンバイ・データベースになるように変換できます。

  フラッシュバック・データベースの詳細は『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を、REDO データの適用の詳細は 7.2.2 項を参照してください。

- Recovery Manager（RMAN）

  Recovery Manager は、データベース・ファイルのバックアップ、リストアおよびリカバリを単純化する Oracle ユーティリティです。Data Guard と同様に、Recovery Manager は Oracle データベースの機能の 1 つであるため、個別にインストールする必要はありません。Data Guard と Recovery Manager は密接に統合されているため、次のことが実現されます。

  - Recovery Manager の `DUPLICATE` コマンドを使用して、プライマリ・データベースのバックアップからスタンバイ・データベースを作成できます。
  - 本番データベースではなくフィジカル・スタンバイ・データベースからバックアップを取得して、本番データベースの負荷を軽減し、スタンバイ・サイトのシステム・リソースを効率的に使用できます。また、フィジカル・スタンバイ・データベースが REDO を適用しているときに、バックアップを取得することもできます。
  - バックアップの実行後、入力に使用されたアーカイブ REDO ログ・ファイルを自動的に削除することで、アーカイブ REDO ログ・ファイルの管理を容易にします。

  付録 F「Recovery Manager を使用したスタンバイ・データベースの作成」および『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

1.7 Data Guard のメリットの要約

Data Guard には次のようなメリットがあります。

- 障害時リカバリ、データ保護および高可用性

  Data Guard は、効率のよい包括的な障害時リカバリおよび高可用性ソリューションを提供します。管理が容易なスイッチオーバー機能とフェイルオーバー機能を使用すると、プライマリ・データベースとスタンバイ・データベース間でロールを可逆的に推移できるため、計画的および計画外の停止によるプライマリ・データベースの停止時間が最小限になります。

- 完全なデータ保護

  Data Guard では、予期しない障害が発生した場合でもデータ消失がないことを保証します。スタンバイ・データベースには、データの破損やユーザーのミスに対する保護対策が用意されています。プライマリ・データベースから受信した REDO データはスタンバイ・データベースで検証されるため、プライマリ・データベースの記憶域レベルの物理破損は、スタンバイ・データベースに伝播されません。同様に、プライマリ・データベースの完全な破損の原因となった論理的な破損やユーザーのミスも解決できます。

- システム・リソースの効率的な使用

  プライマリ・データベースから受信した REDO データで更新されたスタンバイ・データベース表は、バックアップ、レポート生成、要約および問合せなどの他のタスクにも使用できます。したがって、これらのタスクの実行に必要なプライマリ・データベースのワークロードを低減し、貴重な CPU と I/O のサイクルを節約できます。

- 可用性とパフォーマンス要件とのバランスを保つことができるデータ保護の柔軟性

  Oracle Data Guard には、各企業がデータ可用性とシステム・パフォーマンス要件とのバランスを保つことができるように、最大保護、最大可用性および最大パフォーマンスの 3 つのモードが用意されています。

- ギャップの自動検出と自動解消

  ネットワークの問題などで、プライマリ・データベースと 1 つ以上のスタンバイ・データベースとの間の接続が失われた場合、プライマリ・データベース上に生成される REDO データは、宛先のスタンバイ・データベースに送信されません。接続が再度確立された時点で、Data Guard によって、欠落しているアーカイブ REDO ログ・ファイル（ギャップと呼ばれます）が自動的に検出され、それらのファイルがスタンバイ・データベースに自動的に転送されます。スタンバイ・データベースはプライマリ・データベースと同期化されるため、DBA による手動操作は不要です。

- 集中化された単純な管理

  Data Guard Broker は、グラフィカル・ユーザー・インタフェースとコマンドライン・インタフェースを提供し、Data Guard 構成の複数のデータベース全体の管理タスクと操作タスクを自動化します。また、ブローカは、単一の Data Guard 構成内のすべてのシステムを監視します。

- Oracle Database との統合

  Data Guard では Oracle Database Enterprise Edition の機能の 1 つであるため、個別にインストールする必要はありません。

- 自動ロールの推移

  ファスト・スタート・フェイルオーバーが有効になると、プライマリ・サイトで障害が発生した場合、Data Guard Broker により、DBA による操作の必要なしに同期化されたスタンバイ・サイトにフェイルオーバーされます。また、アプリケーションにロールの推移が自動的に通知されます。

2

Data Guard の概説

Data Guard 構成には、1 つのプライマリ・データベースと、それに対応付けられた最大 9 個のスタンバイ・データベースが含まれます。この章では、Data Guard の使用を開始するための次の考慮事項について説明します。

- スタンバイ・データベースのタイプ
- Data Guard 構成の管理のためのユーザー・インタフェース
- Data Guard の動作要件
- スタンバイ・データベースのディレクトリ構造に関する考慮事項

2.1 スタンバイ・データベースのタイプ

スタンバイ・データベースは、Oracle 本番データベースのトランザクション一貫性のあるコピーで、最初はプライマリ・データベースのバックアップ・コピーから作成されます。スタンバイ・データベースが作成および構成されると、Data Guard は、プライマリ・データベースの REDO データをスタンバイ・システムに転送し、スタンバイ・データベースに適用することによって、スタンバイ・データベースを自動的にメンテナンスします。

スタンバイ・データベースのタイプには、フィジカル・スタンバイ・データベース、ロジカル・スタンバイ・データベース、スナップショット・スタンバイ・データベースがあります。フィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースは、必要に応じてプライマリ・データベースのロールを担い、本番処理を引き継ぐことができます。Data Guard 構成には、これらのタイプのスタンバイ・データベースを任意に組み合せて含めることができます。

2.1.1 フィジカル・スタンバイ・データベース

フィジカル・スタンバイ・データベースは、プライマリ・データベースのブロック単位の完全コピーです。フィジカル・スタンバイ・データベースは、REDO Apply と呼ばれるプロセスにより完全コピーとしてメンテナンスされます。すなわち、データベース・リカバリ・メカニズムを使用して、プライマリ・データベースから受信された REDO データがフィジカル・スタンバイ・データベースに継続的に適用されます。

フィジカル・スタンバイ・データベースのメリット

フィジカル・スタンバイ・データベースには次のメリットがあります。

- 障害時リカバリと高可用性

  フィジカル・スタンバイ・データベースは、堅牢で効率的な障害時リカバリおよび高可用性のソリューションです。管理が容易なスイッチオーバー機能とフェイルオーバー機能を使用すると、プライマリ・データベースとフィジカル・スタンバイ・データベース間でロールを可逆的に推移できるため、計画的および計画外の停止によるプライマリ・データベースの停止時間が最小限になります。

- データ保護

  フィジカル・スタンバイ・データベースにより、予期しない障害が発生した場合でもデータ消失を防ぐことができます。フィジカル・スタンバイ・データベースでは、プライマリ・データベースでサポートされるすべてのデータ型、およびすべての DDL 操作と DML 操作がサポートされます。また、データの破損やユーザーのミスに対する保護対策も用意されています。プライマリ・データベースの記憶域レベルの物理破損は、スタンバイ・データベースに伝播されません。同様に、データ消失の原因になりかねない論理的な破損やユーザー・エラーも簡単に解決できます。

- プライマリ・データベースのワークロードの低減

  Oracle Recovery Manager（RMAN）は、フィジカル・スタンバイ・データベースを使用してプライマリ・データベースからバックアップをオフロードし、貴重な CPU と I/O のサイクルを節約できます。

  フィジカル・スタンバイ・データベースは、REDO Apply がアクティブであれば問い合せることもできるため、プライマリ・データベースからフィジカル・スタンバイ・データベースに問合せをオフロードでき、プライマリ・データベースのワークロードはさらに低減します。

- パフォーマンス

  フィジカル・スタンバイ・データベースで使用される REDO Apply テクノロジは、プライマリ・データベースで行われた変更でスタンバイ・データベースを常に更新することに関しては最も効率的なメカニズムです。これは、SQL レベルのコード・レイヤーをすべてバイパスする、低レベルのリカバリ・メカニズムを使用して変更を適用するためです。

2.1.2 ロジカル・スタンバイ・データベース

ロジカル・スタンバイ・データベースは、最初はプライマリ・データベースと同じ内容のコピーで作成されますが、後で異なる構造に変更できます。ロジカル・スタンバイ・データベースは、SQL 文を実行して更新されます。これにより、ユーザーは問合せとレポート生成の目的で、スタンバイ・データベースにいつでもアクセスできます。このように、ロジカル・スタンバイ・データベースは、データ保護操作とレポート生成操作に同時に使用できます。

Data Guard は、ログ・ファイルのデータを SQL 文に変換し、ロジカル・スタンバイ・データベースでその SQL 文を実行することによって、アーカイブ REDO ログ・ファイルまたはスタンバイ REDO ログ・ファイルの情報をロジカル・スタンバイ・データベースに自動的に適用します。ロジカル・スタンバイ・データベースは SQL 文を使用して更新されるため、オープン状態のままであることが必要です。ロジカル・スタンバイ・データベースは読取り / 書込みモードでオープンされますが、再生成された SQL に対するターゲット表は、読取り専用操作用にのみ使用可能です。更新中、これらの表は、レポート生成、要約、問合せなどの他のタスクで同時に使用できます。さらに、これらのタスクは、メンテナンスされている表に追加の索引やマテリアライズド・ビューを作成することによって最適化できます。

ロジカル・スタンバイ・データベースには、データ型、表のタイプ、DDL 操作および DML 操作のタイプに関していくつかの制限があります。ロジカル・スタンバイ・データベースのデータ型および DDL のサポートの詳細は、付録 C を参照してください。

ロジカル・スタンバイ・データベースのメリット

ロジカル・スタンバイ・データベースは、データ・リカバリ（DR）のメリットとともに高可用性（HA）を提供するには理想的です。フィジカル・スタンバイ・データベースと比較して、ロジカル・スタンバイ・データベースは、さらに次の重要な HA のメリットを提供します。

- 別の種類の障害からの保護

  ロジカル・スタンバイは REDO を分析してデータベースに対する論理的な変更を再構成するため、ブロック・レベルの変更を介してレプリケートされる可能性がある、プライマリでの特定の種類のハードウェア障害を検出し、それから保護することができます。Oracle では、同じプライマリ・サーバーに対してフィジカル・スタンバイとロジカル・スタンバイの両方を保持できます。

- リソースの効率的な使用

  ロジカル・スタンバイ・データベースは、プライマリでの変更をレプリケートする場合、読取り / 書込み用にオープンされます。そのため、ロジカル・スタンバイ・データベースを同時に使用して、他の多くのビジネス要件を満たすことができます。たとえば、プライマリのスループットに対して問題のあるワークロードのレポート生成を実行できます。プライマリのデータの完全で正確なコピーに対して、ソフトウェアの新しいリリースやある種のアプリケーションをテストするのにも使用できます。また、ローカル変更に対してプライマリからレプリケートされたデータを保護しつつ、他のアプリケーションや追加スキーマをホスト管理できます。特定の種類の物理的な再構築（パーティショニング・スキームへの変更など）による影響を評価するのにも使用できます。ロジカル・スタンバイはユーザー・トランザクションを識別し、バックグラウンド・システムの変更を除外しながら対象の変更のみをレプリケートするため、目的のトランザクションのみを効率的にレプリケートできます。

- ワークロードの分散

  ロジカル・スタンバイには、最新の一貫性のあるプライマリ・データベースのレプリカを作成するための、単純なターンキー・ソリューションがあります。このソリューションは、ワークロードの分散に使用できます。ワークロードの増加がレポートされた場合、プライマリ・サーバーのトランザクション・スループットに影響を与えずに透過的に負荷を分散して、ロジカル・スタンバイを追加作成できます。

- レポート生成要件および意思決定支援要件のための最適化

  ロジカル・スタンバイの主要なメリットは、重要な補助構造が作成してレポート生成のワークロードを最適化できることです。この構造は、プライマリのトランザクション・レスポンス時間にきわめて大きな影響を与えます。ロジカル・スタンバイは、異なるパーティショニングの様々な記憶域タイプにデータを物理的に再編成できます。また、多種多様な索引を保持し、オンデマンド・リフレッシュ・マテリアライズド・ビューを作成およびメンテナンスすることも可能です。さらに、データ・キューブおよび他の OLAP データ・ビューを作成するのにも使用できます。

- ソフトウェア・アップグレード時の停止時間の最短化

  ロジカル・スタンバイを使用すると、パッチ・セットやソフトウェアの新しいリリースの適用に関連する停止時間を大幅に短縮できます。ロジカル・スタンバイは、新しいリリースへのアップブレード後に切り替えて、アクティブなプライマリにすることができます。これにより、古いプライマリをロジカル・スタンバイに変換してパッチ・セットを適用する間、完全な可用性を可能にします。

2.1.3 スナップショット・スタンバイ・データベース

スナップショット・スタンバイ・データベースは、全面的に更新可能なスタンバイ・データベースで、フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースに変換して作成します。スナップショット・スタンバイ・データベースは、プライマリ・データベースから REDO データを受信およびアーカイブしますが、適用はしません。プライマリ・データベースから受信した REDO データは、スナップショット・スタンバイ・データベースへのローカル更新がすべて破棄された後、スナップショット・スタンバイ・データベースが変換されてフィジカル・スタンバイ・データベースに戻ると適用されます。

スナップショット・スタンバイ・データベースは、通常、時間とともにプライマリ・データベースとは異なってきます。これは、プライマリ・データベースからの REDO データが受信時に適用されないためです。スナップショット・スタンバイ・データベースに対するローカル更新が原因で、相違はさらに大きくなります。しかし、プライマリ・データベースのデータは完全に保護されます。これは、任意の時点でスナップショット・スタンバイを変換してフィジカル・スタンバイ・データベースに戻すことができ、その後にプライマリから受信した REDO データが適用されるためです。

スナップショット・スタンバイ・データベースのメリット

スナップショット・スタンバイ・データベースは、全面的に更新可能なスタンバイ・データベースで、フィジカル・スタンバイ・データベースと同様の、障害時リカバリおよびデータ保護のメリットがあります。スナップショット・スタンバイ・データベースは、プライマリ・データベースの一時的かつ更新可能なスナップショットを保持することにメリットがあれば、管理上の複雑さやプライマリ・データベースの障害からリカバリする時間が増えてもかまわない場合に最もよく使用されます。

スナップショット・スタンバイ・データベースの使用には、次のメリットがあります。

- 常にデータ保護を維持しながら、開発およびテスト用に本番データベースの正確なレプリカを提供します。
- フィジカル・スタンバイに変換して再同期化することで、現行の本番データが格納されるように簡単にリフレッシュできます。

スナップショット・スタンバイの作成、テスト、本番環境との再同期化に続いて、再びスナップショット・スタンバイの作成およびテストを実行できることは、必要に応じて何度も繰り返すことができる 1 サイクルです。同じプロセスを使用して、データへの読取り / 書込みアクセスが必要なレポート生成用にスナップショット・スタンバイを簡単に作成し、定期的に更新できます。

2.2 Data Guard 構成の管理のためのユーザー・インタフェース

Data Guard 構成の構成、実装および管理には、次のインタフェースを使用できます。

- Oracle Enterprise Manager

  Enterprise Manager では、Data Guard 環境の作成、構成および監視に関係するタスクの多くを自動化する、Data Guard Broker の GUI インタフェースが用意されています。GUI およびウィザードの詳細は、『Oracle Data Guard Broker』および Oracle Enterprise Manager のオンライン・ヘルプを参照してください。

- SQL*Plus コマンドライン・インタフェース

  いくつかの SQL*Plus 文では、`STANDBY` キーワードを使用して、スタンバイ・データベースに対する操作を指定します。他の SQL 文にはスタンバイ固有の構文はありませんが、スタンバイ・データベースで操作を実行するときに役立ちます。関連する文のリストは、第 16 章を参照してください。

- 初期化パラメータ

  Data Guard 環境の定義には、いくつかの初期化パラメータが使用されます。関連する初期化パラメータのリストは、第 14 章を参照してください。

- Data Guard Broker コマンドライン・インタフェース（DGMGRL）

  DGMGRL コマンドライン・インタフェースは、Enterprise Manager の代替手段です。DGMGRL コマンドライン・インタフェースは、ブローカを使用してバッチ・プログラムまたはスクリプトから Data Guard 構成を管理する場合に役立ちます。詳細は、『Oracle Data Guard Broker』を参照してください。

2.3 Data Guard の動作要件

次の各項で、Data Guard 使用時の動作要件を説明します。

- ハードウェアおよびオペレーティング・システムの要件
- Oracle ソフトウェア要件

2.3.1 ハードウェアおよびオペレーティング・システムの要件

Oracle Database 11*g* では、また MetaLink Note 4134841.1 に記載されている現在の制限により、Data Guard 構成に対する Data Guard の柔軟性が増し、プライマリ・システムおよびスタンバイ・システムに、異なる CPU アーキテクチャ、オペレーティング・システム（Windows や Linux など）、オペレーティング・システムのバイナリ（32 ビット /64 ビット）または Oracle データベースのバイナリ（32 ビット /64 ビット）が存在してもかまいません。混在するプラットフォームのサポートの詳細は、`https://metalink.oracle.com` から Oracle MetaLink Note 413484.1 を参照してください。

同じリリースの Oracle Database Enterprise Edition をプライマリ・データベースおよびすべてのスタンバイ・データベースにインストールする必要があります。ただし、ロジカル・スタンバイ・データベースを使用したローリング・データベース・アップグレード時を除きます。

関連項目：

- ローリング・データベース・アップグレードの詳細は、第 12 章「SQL Apply を使用した Oracle Database のアップグレード」を参照してください。

2.3.2 Oracle ソフトウェア要件

次のリストに、Data Guard 使用時の Oracle ソフトウェア要件を示します。

- Oracle Data Guard は、Oracle Database Enterprise Edition の機能としてのみ提供されています。Oracle Database Standard Edition には付属していません。

  注意： Oracle Database Standard Edition を実行しているデータベースでスタンバイ・データベース環境を再現することが可能です。これを行うには、オペレーティング・システムのコピー・ユーティリティ、または定期的にアーカイブ REDO ログ・ファイルをデータベースからデータベースに送信するカスタム・スクリプトを使用して、手動でアーカイブ REDO ログ・ファイルを転送します。したがって、この構成では、Data Guard によって提供される利便性、管理性、パフォーマンスおよび障害時リカバリ機能を得られません。

- Data Guard SQL Apply を使用して、Oracle Database ソフトウェアをパッチ・セット・リリース *n*（10.1.0.3 以上）から上位バージョンのパッチ・セットまたはメジャー・バージョン・リリースにローリング・アップグレードできます。ローリング・アップグレード時には、プライマリ・データベースおよびロジカル・スタンバイ・データベースを 1 つずつアップグレードする間、異なるリリースの Oracle Database を実行できます。詳細は、第 12 章「SQL Apply を使用した Oracle Database のアップグレード」および該当する Oracle Database 10*g* パッチ・セット・リリースの README ファイルを参照してください。
- Data Guard 構成のすべてのデータベースで、`COMPATIBLE` 初期化パラメータを同じ値に設定する必要があります。
- 現在 Oracle8*i* データベース・ソフトウェア上で Oracle Data Guard を実行している場合、Oracle Data Guard 11*g* にアップグレードする方法の詳細は、『Oracle Database アップグレード・ガイド』を参照してください。
- プライマリ・データベースは ARCHIVELOG モードで実行する必要があります。詳細は、『Oracle Database 管理者ガイド』を参照してください。
- プライマリ・データベースは、単一インスタンス・データベースまたは Oracle Real Application Clusters（RAC）データベースのいずれかです。スタンバイ・データベースは、単一インスタンス・データベースまたは Oracle RAC データベースで、フィジカル・タイプ、ロジカル・タイプおよびスナップショット・タイプを組み合せることができます。Oracle Data Guard を RAC とともに構成および使用する方法の詳細は、『Oracle Database 高可用性概要』を参照してください。
- プライマリ・データベースとスタンバイ・データベースには、それぞれ独自の制御ファイルが必要です。
- スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、スタンバイ・データベースのアーカイブ・ディレクトリは、プライマリ・データベースとは異なるディレクトリ構造を使用する必要があります。同じ構造を使用すると、スタンバイ・データベースがプライマリ・データベース・ファイルを上書きする可能性があります。
- プライマリ・データベースに対して、ログに記録されず、スタンバイ・データベースに伝播できないダイレクト書込みが行われるのを防ぐには、プライマリ・データベースで `FORCE LOGGING` をオンにした後で、スタンバイ作成用にデータファイルのバックアップを実行します。スタンバイ・データベースが必要な間は、データベースを `FORCE LOGGING` モードに保持してください。
- プライマリ・データベースとスタンバイ・データベースのインスタンスの管理に使用するユーザー・アカウントには、`SYSDBA` システム権限が必要です。
- Oracle Automatic Storage Management（ASM）および Oracle Managed Files（OMF）を Data Guard 構成でセットアップする際、プライマリおよびスタンバイ・データベースに対称的にセットアップすることをお薦めします。つまり、Data Guard 構成のいずれかのデータベースで ASM、OMF、またはその両方を使用する場合、構成内のすべてのデータベースでもそれぞれ ASM、OMF またはその両方を使用する必要があります。詳細は、13.5 項の使用例を参照してください。

注意： 更新の際に時間ベースのデータが使用される一部のアプリケーションでは、複数のタイムゾーンで入力されたデータを処理できないため、ロールの推移後もレコードの時間的順序が維持されるよう、プライマリおよびリモート・スタンバイ・システムに同じタイムゾーンを設定することをお薦めします。

2.4 スタンバイ・データベースのディレクトリ構造に関する考慮事項

各種スタンバイ・データベースのディレクトリ構造によってスタンバイ・データファイル、アーカイブ REDO ログ・ファイルおよびスタンバイ REDO ログ・ファイルのパス名が決まるため、ディレクトリ構造は重要です。可能であれば、プライマリ・システムとスタンバイ・システムでデータファイル、ログ・ファイルおよび制御ファイルの名前およびパス名を同一にし、Optimal Flexible Architecture（OFA）のネーミング規則を使用する必要があります。スタンバイ・データベースのアーカイブ・ディレクトリも、サイズおよび構造を含めてサイト間で同一であることが必要です。この方法により、バックアップ、スイッチオーバーおよびフェイルオーバーなどの他の操作を同じ手順で実行できるようになり、メンテナンスの複雑さが軽減されます。

関連項目： Optimal Flexible Architecture（OFA）の詳細は、オペレーティング・システム固有の Oracle ドキュメントを参照してください。

同一にしない場合は、ファイル名変換パラメータ（表 2-1 を参照）を設定するか、データファイルの名前を変更する必要があります。ディレクトリ構造が異なるシステムを使用する必要がある場合や、スタンバイ・データベースとプライマリ・データベースのシステムを同じにする必要がある場合は、管理作業を最小限に抑えるようにしてください。

図 2-1 に、3 つの基本構成オプションを示します。構成オプションは、次のとおりです。

- スタンバイ・データベースがプライマリ・データベースと同じシステムにあるが、プライマリ・システムとは異なるディレクトリ構造を使用する。これは、図 2-1 の `Standby1` です。

  スタンバイ・データベースをプライマリ・データベースと同じシステムに配置する場合は、異なるディレクトリ構造を使用する必要があります。同じ構造を使用すると、スタンバイ・データベースがプライマリ・データベース・ファイルを上書きしようとします。

- スタンバイ・データベースが別個のシステム上にあり、プライマリ・システムと同じディレクトリ構造を使用する。これは、図 2-1 の `Standby2` です。このオプションをお薦めします。

- スタンバイ・データベース別個のシステム上にあり、プライマリ・システムと異なるディレクトリ構造を使用する。これは、図 2-1 の `Standby3` です。

注意： Data Guard 構成のいずれかのデータベースで ASM、OMF、またはその両方を使用する場合、構成内のすべてのデータベースでもそれぞれ ASM、OMF またはその両方を使用する必要があります。Data Guard 構成での OMF のセットアップ方法を説明する使用例は、第 13 章を参照してください。

図 2-1 可能なスタンバイ構成

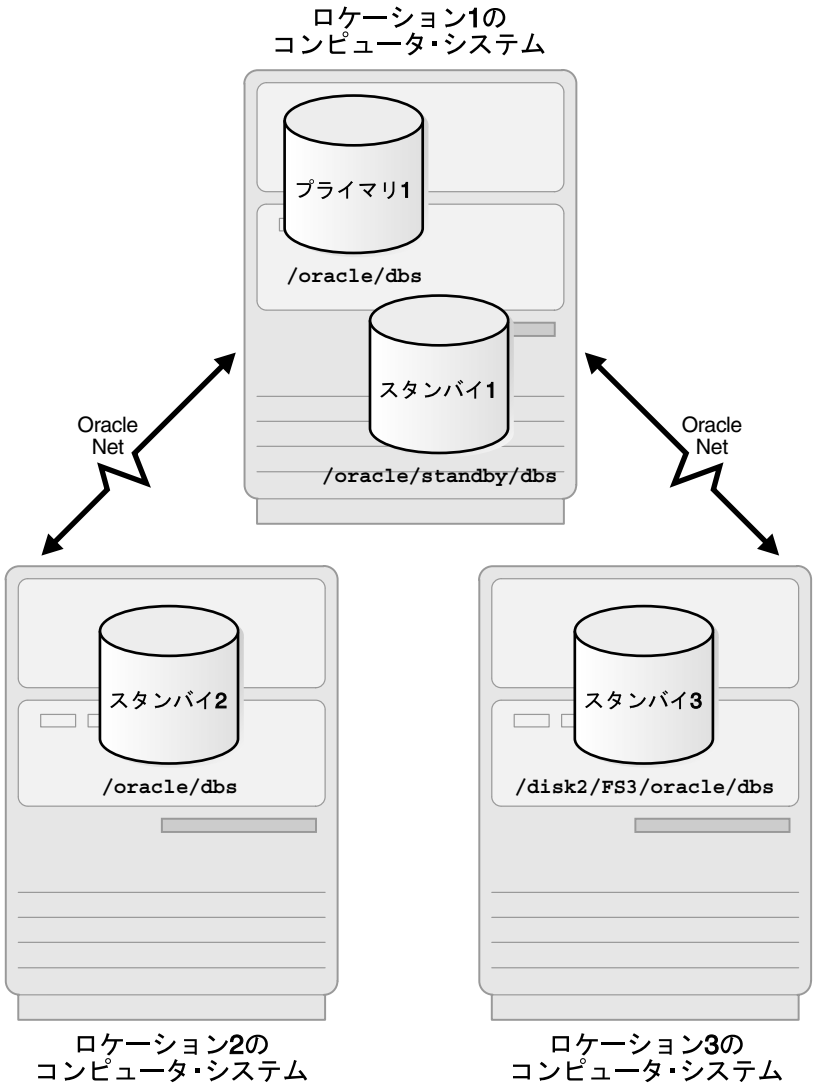


表 2-1 では、プライマリ・データベースとスタンバイ・データベースの可能な構成およびそれぞれの結果的な注意事項について説明します。Data Guard 構成の複数のメンバーが同じシステムに存在する場合は、DB_UNIQUE_NAME 初期化パラメータに一意の値を指定する必要があります。各データベースが異なるシステムに存在する場合も、DB_UNIQUE_NAME 初期化パラメータには常に一意の値を指定することをお薦めします。

表 2-1 スタンバイ・データベースの位置とディレクトリ・オプション

スタンバイ・システム	ディレクトリ構造	結果的な注意事項
プライマリ・システムと同じ	プライマリ・システムと異なる（必須）	■ DB_UNIQUE_NAME 初期化パラメータを設定する必要があります。 ■ ファイル名を手動で変更するか、スタンバイ・データベースで DB_FILE_NAME_CONVERT 初期化パラメータおよび LOG_FILE_NAME_CONVERT 初期化パラメータを設定すると、スタンバイ・データベース制御ファイル内のプライマリ・データベースのデータファイル、アーカイブ REDO ログ・ファイルおよびスタンバイ REDO ログ・ファイルのパス名を自動的に更新できます。（3.1.4 項を参照。） ■ スタンバイ・データベースは、プライマリ・データベースとスタンバイ・データベースが存在しているシステムを破壊するような障害に対する保護には役立ちませんが、計画的なメンテナンスに対するスイッチオーバー機能は提供します。
離れたシステム	プライマリ・システムと同じ	■ スタンバイ・データベース制御ファイル内のプライマリ・データベースのファイル名、アーカイブ REDO ログ・ファイルおよびスタンバイ REDO ログ・ファイル名を変更する必要はありません。ただし、新しいネーミング計画が必要な場合（複数のディスクにファイルを分散する場合など）は、変更しても構いません。 ■ スタンバイ・データベースを別の物理メディアに置くことにより、プライマリ・システムを破壊するような障害からプライマリ・データベースのデータを保護できます。
離れたシステム	プライマリ・システムと異なる	■ ファイル名を手動で変更するか、またはスタンバイ・データベースで DB_FILE_NAME_CONVERT 初期化パラメータおよび LOG_FILE_NAME_CONVERT 初期化パラメータを設定すると、データファイル名を自動的に変更できます（3.1.4 項を参照）。 ■ スタンバイ・データベースを別の物理メディアに置くことにより、プライマリ・システムを破壊するような障害からプライマリ・データベースのデータを保護できます。

3

フィジカル・スタンバイ・データベースの作成

この章では、フィジカル・スタンバイ・データベースを作成する手順について説明します。この章は、次の主な項目で構成されています。

- スタンバイ・データベースを作成するためのプライマリ・データベースの準備
- フィジカル・スタンバイ・データベースの作成手順
- 作成後の手順

この章で説明する手順では、スタンバイ・データベースが最大パフォーマンス・モードで構成されます。このモードは、デフォルトのデータ保護モードです。別のデータ保護モードの構成については、第 5 章を参照してください。

関連項目：

- サーバー・パラメータ・ファイルの作成方法および使用方法は、『Oracle Database 管理者ガイド』を参照してください。
- グラフィカル・ユーザー・インタフェースを使用してフィジカル・スタンバイ・データベースを自動的に作成する方法は、『Oracle Data Guard Broker』および Enterprise Manager のオンライン・ヘルプを参照してください。
- Recovery Manager（RMAN）を使用したスタンバイ・データベースの作成方法の詳細は、付録 F を参照してください。

3.1 スタンバイ・データベースを作成するためのプライマリ・データベースの準備

スタンバイ・データベースを作成する前に、プライマリ・データベースが正しく構成されていることを確認する必要があります。

表 3-1 は、フィジカル・スタンバイ・データベースを作成するための準備としてプライマリ・データベースで実行するタスクのチェックリストです。各タスクを詳細に説明している参照先の項も記載されています。

表 3-1 フィジカル・スタンバイ・データベースを作成するためのプライマリ・データベースの準備

参照先	タスク
3.1.1 項	強制ロギングの有効化
3.1.2 項	REDO 転送の認証の構成
3.1.3 項	REDO データを受信するためのプライマリ・データベースの構成
3.1.4 項	プライマリ・データベースの初期化パラメータの設定
3.1.5 項	アーカイブの有効化

注意： これらの準備のためのタスクは、1 回のみ実行してください。これらの手順を完了すると、データベースは、1 つ以上のスタンバイ・データベースに対するプライマリ・データベースとして機能する準備が整います。

3.1.1 強制ロギングの有効化

データベースの作成後、次の SQL 文を使用して、プライマリ・データベースを FORCE LOGGING モードにします。

```
SQL> ALTER DATABASE FORCE LOGGING;
```

この文は、完了までに非常に時間がかかる場合があります。これは、ログに記録されないダイレクト書込み I/O がすべて完了するまで待機するためです。

3.1.2 REDO 転送の認証の構成

Data Guard では、Oracle Net セッションを使用して Data Guard 構成のメンバー間で REDO データを転送し、メッセージを制御します。これらの REDO 転送セッションは、Secure Sockets Layer（SSL）プロトコルまたはリモート・ログイン・パスワード・ファイルのいずれかを使用して認証されます。

SSL は、次の場合に 2 つのデータベース間の REDO 転送セッションの認証に使用されます。

- データベースが同じ Oracle Internet Directory（OID）エンタープライズ・ドメインのメンバーであり、現行ユーザーのデータベース・リンクの使用を許可する場合
- データベースに対応する LOG_ARCHIVE_DEST_*n*、FAL_SERVER および FAL_CLIENT の各データベース初期化パラメータで、SSL 用に構成された Oracle Net 接続記述子を使用する場合
- 各データベースに、データベースの OID エントリの識別名（DN）と一致する DN が付けられたユーザー証明書を格納する Oracle ウォレットまたはサポートされるハードウェア・セキュリティ・モジュールがある場合

SSL 認証の要件が満たされない場合は、リモート・ログイン・パスワード・ファイルを使用するように Data Guard 構成の各メンバーを構成する必要があります。また、構成内のすべてのフィジカル・スタンバイ・データベースには、プライマリ・データベースのパスワード・ファイルの最新コピーが必要です。

SYSDBA 権限または SYSOPER 権限を付与または取り消すか、これらの権限を持つユーザーのログイン・パスワードを変更するたびに、構成内の各フィジカル・スタンバイ・データベースまたはスナップショット・スタンバイ・データベースで、パスワード・ファイルをプライマリ・データベースのパスワード・ファイルの最新コピーで置き換える必要があります。

関連項目：

- リモート・ログイン・パスワード・ファイルの詳細は、『Oracle Database 管理者ガイド』を参照してください。
- 『Oracle Database Advanced Security 管理者ガイド』
- 『Oracle Database Net Services 管理者ガイド』

3.1.3 REDO データを受信するためのプライマリ・データベースの構成

このタスクはオプションですが、Data Guard 構成の作成時に、REDO データを受信するようにプライマリ・データベースを構成することをお薦めします。このベスト・プラクティスに従うことで、プライマリ・データベースは、スタンバイ・ロールに速やかに推移して REDO データの受信を開始できるようになります。

REDO データを受信するようにデータベースを構成する方法の詳細は、6.2.3 項を参照してください。

3.1.4 プライマリ・データベースの初期化パラメータの設定

プライマリ・データベースでは、データベースがプライマリ・ロールで動作している間の REDO 転送サービスを制御する初期化パラメータを定義します。また、プライマリ・データベースがスタンバイ・ロールに推移したときに REDO データの受信と適用サービスを制御するパラメータを追加する必要があります。

例 3-1 に、プライマリ・データベースでメンテナンスする、プライマリ・ロールの初期化パラメータを示します。この例は、シカゴにプライマリ・データベースがあり、ボストンにフィジカル・スタンバイ・データベースが 1 つある Data Guard 構成を示しています。例 3-1 に示すパラメータは、シカゴのデータベースがプライマリ・データベース・ロールまたはスタンバイ・データベース・ロールで稼働している場合に有効です。この構成例では、次の表に示す名前を使用しています。

データベース	DB_UNIQUE_NAME	Oracle Net サービス名
プライマリ	chicago	chicago
フィジカル・スタンバイ	boston	boston

例 3-1 プライマリ・データベース：プライマリ・ロールの初期化パラメータ

```
DB_NAME=chicago
DB_UNIQUE_NAME=chicago
LOG_ARCHIVE_CONFIG='DG_CONFIG=(chicago,boston)'
CONTROL_FILES='/arch1/chicago/control1.ctl', '/arch2/chicago/control2.ctl'
LOG_ARCHIVE_DEST_1=
 'LOCATION=/arch1/chicago/
  VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=chicago'
LOG_ARCHIVE_DEST_2=
 'SERVICE=boston ASYNC
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
LOG_ARCHIVE_FORMAT=%t_%s_%r.arc
LOG_ARCHIVE_MAX_PROCESSES=30
```

これらのパラメータは、REDO 転送サービスが REDO データをスタンバイ・システムに転送する方法と、ローカル・ファイル・システムでの REDO データのアーカイブ処理を制御します。例では、LOG_ARCHIVE_DEST_2 初期化パラメータに REDO データを転送する非同期（ASYNC）ネットワーク転送を指定します。これらは推奨設定であり、スタンバイ REDO ログ・ファイルが必要です（3-3 ページの 3.1.3 項「REDO データを受信するためのプライマリ・データベースの構成」を参照）。

例 3-2 に、プライマリ・データベースで定義するスタンバイ・ロールの追加の初期化パラメータを示します。これらのパラメータは、プライマリ・データベースがスタンバイ・ロールに推移すると有効になります。

例 3-2 プライマリ・データベース：スタンバイ・ロールの初期化パラメータ

```
FAL_SERVER=boston
FAL_CLIENT=chicago
DB_FILE_NAME_CONVERT='boston','chicago'
LOG_FILE_NAME_CONVERT=
 '/arch1/boston/','/arch1/chicago/','/arch2/boston/','/arch2/chicago/'
STANDBY_FILE_MANAGEMENT=AUTO
```

例 3-2 に示す初期化パラメータを指定すると、プライマリ・データベースはギャップを解決して新しいプライマリ・データベースからの新規データファイルとログ・ファイルのパス名を変換するように設定され、このデータベースがスタンバイ・ロールで動作しているときの着信 REDO データがアーカイブされます。説明に従ってプライマリおよびスタンバイ・ロールについて初期化パラメータを設定した場合、ロールの推移後に変更が必要なパラメータはありません。

次の表に、例 3-1 および例 3-2 に示した各パラメータ設定に関する簡単な説明を示します。

パラメータ	推奨する設定
DB_NAME	8 文字の名前を指定します。すべてのスタンバイ・データベースに同じ名前を使用してください。
DB_UNIQUE_NAME	各データベースの一意の名前を指定します。この名前はデータベースと固定され、プライマリ・データベースとスタンバイ・データベースのロールが逆になっても変更されません。
LOG_ARCHIVE_CONFIG	このパラメータの DG_CONFIG 属性を指定して Data Guard 構成のプライマリ・データベースとスタンバイ・データベースの DB_UNIQUE_NAME をリストすると、Oracle RAC プライマリ・データベースが最大保護モードまたは最大可用性モードで稼働している Data Guard 構成に、スタンバイ・データベースを動的に追加できます。デフォルトでは、LOG_ARCHIVE_CONFIG パラメータを指定すると、データベースは REDO を送受信できます。
CONTROL_FILES	プライマリ・データベースの制御ファイルのパス名を指定します。例 3-1 は、2 つの制御ファイルのパス名を指定する方法を示しています。不良制御ファイルの位置に正常な制御ファイルをコピーした後でインスタンスを簡単に再起動できるように、制御ファイルの第 2 コピーを使用可能にしておくことをお薦めします。
LOG_ARCHIVE_DEST_n	REDO データがアーカイブされるプライマリ・システムおよびスタンバイ・システムの位置を指定します。例 3-1 では、次のようになっています。 ■ LOG_ARCHIVE_DEST_1 と指定すると、プライマリ・データベースにより生成された REDO データは、ローカルのオンライン REDO ログ・ファイルから /arch1/chicago/ にあるローカルのアーカイブ REDO ログ・ファイルにアーカイブされます。 ■ LOG_ARCHIVE_DEST_2 は、プライマリ・ロールにのみ有効です。この宛先を指定すると、REDO データはリモート・フィジカル・スタンバイ宛先 boston に転送されます。 **注意：**フラッシュ・リカバリ領域を（DB_RECOVERY_FILE_DEST 初期化パラメータを使用して）構成しており、LOCATION 属性でローカル・アーカイブ先を明示的に構成していない場合は、デフォルトのローカル・アーカイブ先として LOG_ARCHIVE_DEST_10 初期化パラメータが自動的に使用されます。また、LOG_ARCHIVE_DEST_*n* の詳細は、第 15 章を参照してください。
LOG_ARCHIVE_DEST_STATE_n	REDO 転送サービスが REDO データを指定した宛先に転送するのを許可するには、ENABLE を指定します。
REMOTE_LOGIN_PASSWORDFILE	リモート・ログイン・パスワード・ファイルを使用して管理ユーザーまたは REDO 転送セッションを認証する場合は、このパラメータを EXCLUSIVE または SHARED に設定する必要があります。
LOG_ARCHIVE_FORMAT	スレッド（%t）、順序番号（%s）およびリセットログ ID（%r）を使用して、アーカイブ REDO ログ・ファイルのフォーマットを指定します。
LOG_ARCHIVE_MAX_PROCESSES =integer	Oracle ソフトウェアが最初に起動するアーカイバ（ARC*n*）・プロセスの最大数（1 ～ 30）を指定します。デフォルト値は 4 です。
FAL_SERVER	FAL サーバー（通常はプライマリ・ロールで実行中のデータベース）の Oracle Net サービス名を指定します。シカゴのデータベースがスタンバイ・ロールで実行されている場合、ボストンのデータベースが欠落しているログ・ファイルを自動的に送信できなければ、ボストンのデータベースが FAL サーバーとして使用され、そこから欠落しているアーカイブ REDO ログ・ファイルがフェッチ（要求）されます。
FAL_CLIENT	シカゴのデータベースの Oracle Net サービス名を指定します。FAL サーバー（ボストン）は、欠落しているアーカイブ REDO ログ・ファイルをシカゴのスタンバイ・データベースにコピーします。

パラメータ	推奨する設定
DB_FILE_NAME_CONVERT	プライマリ・データベースのデータファイルのパス名とファイル名の位置を指定し、その後にスタンバイの位置を指定します。このパラメータにより、プライマリ・データベースのデータファイルのパス名がスタンバイ・データファイルのパス名に変換されます。スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、またはデータファイルが格納されているスタンバイ・サイトのディレクトリ構造がプライマリ・サイトと異なる場合は、このパラメータが必要です。このパラメータは、フィジカル・スタンバイ・データベースのパス名の変換にのみ使用されることに注意してください。このパラメータにより、複数のペアのパスを指定できます。
LOG_FILE_NAME_CONVERT	プライマリ・データベースのオンライン REDO ログ・ファイルの位置を指定し、その後にスタンバイの位置を指定します。このパラメータにより、プライマリ・データベースのログ・ファイルのパス名がスタンバイ・データベースのパス名に変換されます。スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、またはログ・ファイルが格納されているスタンバイ・システムのディレクトリ構造がプライマリ・システムと異なる場合は、このパラメータが必要です。このパラメータにより、複数のペアのパスを指定できます。
STANDBY_FILE_MANAGEMENT	データファイルがプライマリ・データベースに追加または削除された場合に、それに対応してスタンバイ・データベースも自動的に変更されるように、AUTO に設定します。

注意： 変更の必要性がある他のパラメータについては、初期化パラメータ・ファイルを調べてください。たとえば、ダンプ出力先パラメータの変更が必要な場合があります。これは、スタンバイ・データベースのディレクトリ位置がプライマリ・データベースで指定したディレクトリ位置と異なる場合に必要です。

3.1.5 アーカイブの有効化

アーカイブが有効になっていない場合は、次の文を発行して、プライマリ・データベースを ARCHIVELOG モードにし、自動アーカイブを有効にします。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER DATABASE OPEN;
```

アーカイブの詳細は、『Oracle Database 管理者ガイド』を参照してください。

3.2 フィジカル・スタンバイ・データベースの作成手順

この項では、フィジカル・スタンバイ・データベースの作成で実行するタスクについて説明します。

表 3-2 は、フィジカル・スタンバイ・データベースの作成で実行するタスク、および各タスクを実行するデータベースのチェックリストです。各タスクを詳細に説明している参照先の項も記載されています。

表 3-2 フィジカル・スタンバイ・データベースの作成

参照先	タスク	データベース
3.2.1 項	プライマリ・データベース・データファイルのバックアップ・コピーの作成	プライマリ
3.2.2 項	スタンバイ・データベース用の制御ファイルの作成	プライマリ
3.2.3 項	スタンバイ・データベース用の初期化パラメータ・ファイルの準備	プライマリ
3.2.4 項	プライマリ・システムからスタンバイ・システムへのファイルのコピー	プライマリ
3.2.5 項	スタンバイ・データベースをサポートする環境の設定	スタンバイ
3.2.6 項	フィジカル・スタンバイ・データベースの起動	スタンバイ
3.2.7 項	フィジカル・スタンバイ・データベースが正しく実行されているかどうかの確認	スタンバイ

3.2.1 プライマリ・データベース・データファイルのバックアップ・コピーの作成

データベースを完全にリカバリするのに必要なアーカイブ REDO ログ・ファイルがあるかぎり、プライマリ・データベースのバックアップ・コピーを使用して、フィジカル・スタンバイ・データベースを作成できます。Recovery Manager（RMAN）ユーティリティを使用することをお薦めします。

バックアップの推奨事項は『Oracle 高可用性アーキテクチャおよびベスト・プラクティス』を、データベースのバックアップ操作の実行方法は『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

3.2.2 スタンバイ・データベース用の制御ファイルの作成

バックアップ処理のために、プライマリ・データベースを停止する必要がある場合は、次の SQL*Plus 文を発行して、プライマリ・データベースを起動します。

```
SQL> STARTUP MOUNT;
```

スタンバイ・データベース用の制御ファイルを作成してから、ユーザー・アクセス用にプライマリ・データベースをオープンします。次に例を示します。

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/tmp/boston.ctl';
SQL> ALTER DATABASE OPEN;
```

注意： プライマリ・データベースとスタンバイ・データベースの両方に単一の制御ファイルを使用することはできません。

3.2.3 スタンバイ・データベース用の初期化パラメータ・ファイルの準備

次の手順を実行して、スタンバイの初期化パラメータ・ファイルを作成します。

手順 1 スタンバイ・データベースにプライマリ・データベース・パラメータ・ファイルをコピーする

プライマリ・データベースで使用されているサーバー・パラメータ・ファイル（SPFILE）から、テキストの初期化パラメータ・ファイル（PFILE）を作成します。テキストの初期化パラメータ・ファイルは、スタンバイの位置にコピーして変更できます。次に例を示します。

```
SQL> CREATE PFILE='/tmp/initboston.ora' FROM SPFILE;
```

このファイルを変更して、フィジカル・スタンバイ・データベースでの使用に適したパラメータ値を含めます。このファイルは、後述の 3.2.5 項で、サーバー・パラメータ・ファイルに変換しなおします。

手順 2 フィジカル・スタンバイ・データベースで初期化パラメータを設定する

プライマリ・システムからコピーしたテキストの初期化パラメータ・ファイルの初期化パラメータ設定の多くは、フィジカル・スタンバイ・データベースにも適していますが、一部を変更する必要があります。

例 3-3 は、スタンバイの初期化パラメータ・ファイルの一部です。この中の値は、フィジカル・スタンバイ・データベース用に変更されています。例 3-1 および例 3-2 と異なるパラメータ値は、太字で示されています。例 3-3 に示すパラメータは、ボストンのデータベースがプライマリ・データベース・ロールまたはスタンバイ・データベース・ロールで稼働している場合に有効です。

例 3-3 フィジカル・スタンバイ・データベース用の初期化パラメータの変更

```
.
.
.
DB_NAME=chicago
DB_UNIQUE_NAME=boston
LOG_ARCHIVE_CONFIG='DG_CONFIG=(chicago,boston)'
CONTROL_FILES='/arch1/boston/control1.ctl', '/arch2/boston/control2.ctl'
DB_FILE_NAME_CONVERT='chicago','boston'
LOG_FILE_NAME_CONVERT=
 '/arch1/chicago/','/arch1/boston/','/arch2/chicago/','/arch2/boston/'
LOG_ARCHIVE_FORMAT=log%t_%s_%r.arc
LOG_ARCHIVE_DEST_1=
 'LOCATION=/arch1/boston/
  VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_2=
 'SERVICE=chicago ASYNC
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=chicago'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
STANDBY_FILE_MANAGEMENT=AUTO
FAL_SERVER=chicago
FAL_CLIENT=boston
.
.
.
```

プライマリ・データベースとスタンバイ・データベースでは、COMPATIBLE 初期化パラメータを同じ値に設定する必要があります。値が異なる場合は、REDO 転送サービスが REDO データをプライマリ・データベースからスタンバイ・データベースに転送できない可能性があります。Data Guard 構成では、COMPATIBLE を少なくとも 9.2.0.1.0 に設定する必要があります。ただし、Oracle Database 11*g* の新機能を利用する場合は、COMPATIBLE パラメータを 11.0.0 に設定します。

SHOW PARAMETERS コマンドを使用して、他に変更を必要とするパラメータがないかどうかを確認することをお薦めします。

次の表に、例 3-3 に示したパラメータ設定のうち、プライマリ・データベースとは異なる設定に関する簡単な説明を示します。

パラメータ	推奨する設定
DB_UNIQUE_NAME	このデータベースの一意の名前を指定します。この名前はデータベースと固定され、プライマリ・データベースとスタンバイ・データベースのロールが逆になっても変更されません。
CONTROL_FILES	スタンバイ・データベースの制御ファイルのパス名を指定します。例 3-3 は、2 つの制御ファイルのパス名を指定する方法を示しています。不良制御ファイルの位置に正常な制御ファイルをコピーした後でインスタンスを簡単に再起動できるように、制御ファイルの第 2 コピーを使用可能にしておくことをお薦めします。
DB_FILE_NAME_CONVERT	プライマリ・データベースのデータファイルのパス名とファイル名の位置を指定し、その後にスタンバイの位置を指定します。このパラメータにより、プライマリ・データベースのデータファイルのパス名がスタンバイ・データファイルのパス名に変換されます。スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、またはデータファイルが格納されているスタンバイ・サイトのディレクトリ構造がプライマリ・サイトと異なる場合は、このパラメータが必要です。
LOG_FILE_NAME_CONVERT	プライマリ・データベースのオンライン REDO ログ・ファイルの位置を指定し、その後にスタンバイの位置を指定します。このパラメータにより、プライマリ・データベースのログ・ファイルのパス名がスタンバイ・データベースのパス名に変換されます。スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、またはログ・ファイルが格納されているスタンバイ・システムのディレクトリ構造がプライマリ・システムと異なる場合は、このパラメータが必要です。
LOG_ARCHIVE_DEST_*n*	REDO データのアーカイブ先を指定します。例 3-3 では、次のようになっています。 ■ LOG_ARCHIVE_DEST_1 を指定すると、プライマリ・データから受信した REDO データは /arch1/boston/ のアーカイブ REDO ログ・ファイルにアーカイブされます。 ■ LOG_ARCHIVE_DEST_2 はプライマリ・ロールのみに有効であるため、この宛先は現在は無視されます。スイッチオーバーが発生して、このインスタンスがプライマリ・データベースになる場合は、REDO データがリモートのシカゴの宛先に転送されます。 **注意 :** フラッシュ・リカバリ領域を（DB_RECOVERY_FILE_DEST 初期化パラメータを使用して）構成しており、LOCATION 属性でローカル・アーカイブ先を明示的に構成していない場合は、デフォルトのローカル・アーカイブ先として LOG_ARCHIVE_DEST_10 初期化パラメータが自動的に使用されます。また、LOG_ARCHIVE_DEST_*n* の詳細は、第 15 章を参照してください。
FAL_SERVER	FAL サーバー（通常はプライマリ・ロールで実行中のデータベース）の Oracle Net サービス名を指定します。ボストンのデータベースがスタンバイ・ロールで実行されている場合、シカゴのデータベースが欠落しているログ・ファイルを自動的に送信できなければ、シカゴのデータベースが FAL サーバーとして使用され、そこから欠落しているアーカイブ REDO ログ・ファイルがフェッチ（要求）されます。
FAL_CLIENT	ボストンのデータベースの Oracle Net サービス名を指定します。FAL サーバー（シカゴ）は、欠落しているアーカイブ REDO ログ・ファイルをボストンのスタンバイ・データベースにコピーします。

注意： 変更の必要性がある他のパラメータについては、初期化パラメータ・ファイルを調べてください。たとえば、ダンプ出力先パラメータの変更が必要な場合があります。これは、スタンバイ・データベースのディレクトリ位置がプライマリ・データベースで指定したディレクトリ位置と異なる場合に必要です。

3.2.4 プライマリ・システムからスタンバイ・システムへのファイルのコピー

オペレーティング・システムのコピー・ユーティリティを使用して、次のバイナリ・ファイルをプライマリ・システムからスタンバイ・システムにコピーします。

- 3.2.1 項で作成したバックアップ・データファイル
- 3.2.2 項で作成したスタンバイ制御ファイル
- 3.2.3 項で作成した初期化パラメータ・ファイル

3.2.5 スタンバイ・データベースをサポートする環境の設定

次の手順を実行して、Windows ベース・サービスの作成、パスワード・ファイルの作成、Oracle Net 環境の設定および SPFILE の作成を行います。

手順 1 Windows ベース・サービスを作成する

スタンバイ・データベースのホストが Windows システムである場合、ORADIM ユーティリティを使用して Windows サービスを作成します。次に例を示します。

```
WINNT> oradim -NEW -SID boston -STARTMODE manual
```

ORADIM ユーティリティの使用方法の詳細は、『Oracle Database プラットフォーム・ガイド for Microsoft Windows』を参照してください。

手順 2 リモート・ログイン・パスワード・ファイルをプライマリ・データベース・システムからスタンバイ・データベース・システムをコピーする

プライマリ・データベースにリモート・ログイン・パスワード・ファイルがある場合は、そのファイルをフィジカル・スタンバイ・データベース・システムの適切なディレクトリにコピーします。パスワード・ファイルは、SYSDBA 権限または SYSOPER 権限が付与または取り消されるか、これらの権限を持つユーザーのログイン・パスワードが変更されるたびに再コピーする必要があります。

この手順は、オペレーティング・システムの認証を管理ユーザーに使用する場合および SSL を REDO 転送の認証に使用する場合はオプションです。

手順 3 プライマリ・データベースとスタンバイ・データベースに対するリスナーを構成する

プライマリ・サイトとスタンバイ・サイトの両方で、Oracle Net Manager を使用して、各データベースに対するリスナーを構成します。

リスナーを再起動して新しい定義を読み込むには、プライマリ・システムとスタンバイ・システムの両方で次の LSNRCTL ユーティリティ・コマンドを入力します。

```
% lsnrctl stop
% lsnrctl start
```

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

手順 4 Oracle Net サービス名を作成する

プライマリ・システムとスタンバイ・システムの両方で、Oracle Net Manager を使用して、プライマリ・データベースとスタンバイ・データベースのネットワーク・サービス名を作成します。ネットワーク・サービス名は REDO 転送サービスで使用されます。

Oracle Net ネット・サービス名は、プライマリ・データベースとスタンバイ・データベースに対するリスナーの構成時に指定したものと同じプロトコル、ホスト・アドレス、ポートおよびサービスを使用する接続記述子に解析される必要があります。この接続記述子は、専用サーバーが使用されるように指定する必要もあります。

『Oracle Database Net Services 管理者ガイド』および『Oracle Database 管理者ガイド』を参照してください。

手順 5 スタンバイ・データベース用のサーバー・パラメータ・ファイルを作成する

アイドル状態のスタンバイ・データベースで、SQL CREATE 文を使用して、3-8 ページの手順 2 で編集したテキストの初期化パラメータ・ファイルから、スタンバイ・データベース用のサーバー・パラメータ・ファイルを作成します。次に例を示します。

```
SQL> CREATE SPFILE FROM PFILE='initboston.ora';
```

手順 6 プライマリ・データベースの暗号化ウォレットをスタンバイ・データベース・システムにコピーする

プライマリ・データベースにデータベース暗号化ウォレットがある場合は、スタンバイ・データベース・システムにコピーし、そのウォレットを使用するようにスタンバイ・データベースを構成します。

注意 : データベース暗号化ウォレットは、マスター暗号化キーが更新されるたびに、プライマリ・データベース・システムから各スタンバイ・データベース・システムにコピーする必要があります。

スタンバイ・データベースの暗号化されたデータには、プライマリ・データベースの現行のマスター暗号化キーを格納するデータベース暗号化ウォレットまたはハードウェア・セキュリティ・モジュールを指すように、スタンバイ・データベースが構成されるまでアクセスできません。

関連項目 : 透過的データベース暗号化の詳細は、『Oracle Database Advanced Security 管理者ガイド』を参照してください。

3.2.6 フィジカル・スタンバイ・データベースの起動

フィジカル・スタンバイ・データベースおよび REDO Apply を起動するには、次の手順を実行します。

手順 1 フィジカル・スタンバイ・データベースを起動する

スタンバイ・データベースで、次の SQL 文を発行してデータベースを起動およびマウントします。

```
SQL> STARTUP MOUNT;
```

手順 2 REDO データを受信するためにスタンバイ・データベースを準備する

REDO データをプライマリ・データベースから受信してアーカイブするように、6.2.3 項で説明する手順を実行して、スタンバイ・データベースを準備します。

手順 3 スタンバイ・データベースでオンライン REDO ログを作成する

この手順はオプションですが、スタンバイ・データベースの作成時に、オンライン REDO ログを作成することをお薦めします。このベスト・プラクティスに従うことで、スタンバイ・データベースは、プライマリ・データベース・ロールに速やかに推移できるようになります。

スタンバイ・データベースのオンライン REDO ログにおける REDO ログ・グループのサイズおよび数は、スタンバイ・データベースがプライマリ・ロールに推移した場合に十分に機能するように選択する必要があります。

手順 4 REDO Apply を開始する

スタンバイ・データベースで、次のコマンドを発行して REDO Apply を開始します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT
FROM SESSION;
```

この文には DISCONNECT FROM SESSION オプションが指定されているため、REDO Apply はバックグラウンド・セッションで実行されます。詳細は、7.3 項「REDO データのフィジカル・スタンバイ・データベースへの適用」を参照してください。

また、この文には USING CURRENT LOGFILE 句が指定されているため、REDO を受信後すぐに適用できます。詳細は、7.3.1 項「REDO Apply の開始」を参照してください。

3.2.7 フィジカル・スタンバイ・データベースが正しく実行されているかどうかの確認

フィジカル・スタンバイ・データベースを作成して REDO 転送サービスを設定した後、データベースの変更がプライマリ・データベースからスタンバイ・データベースに正常に転送されているかどうかの確認が必要な場合があります。

スタンバイ・データベースで REDO データが受信されていることを確認するには、最初に、スタンバイ・データベースの既存のアーカイブ REDO ログ・ファイルを識別し、ログ・スイッチを強制実行して、プライマリ・データベースのオンライン REDO ログ・ファイルをいくつかアーカイブし、スタンバイ・データベースを再度チェックする必要があります。このタスクの実行手順を次に示します。

手順 1 既存のアーカイブ REDO ログ・ファイルを識別する

スタンバイ・データベースで V$ARCHIVED_LOG ビューを問い合せて、アーカイブ REDO ログの既存のファイルを識別します。次に例を示します。

```
SQL> SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME
  2  FROM V$ARCHIVED_LOG ORDER BY SEQUENCE#;

 SEQUENCE# FIRST_TIME         NEXT_TIME
---------- ------------------ ------------------
         8 11-JUL-07 17:50:45 11-JUL-07 17:50:53
         9 11-JUL-07 17:50:53 11-JUL-07 17:50:58
        10 11-JUL-07 17:50:58 11-JUL-07 17:51:03

3 rows selected.
```

手順 2 ログ・スイッチを強制実行して現行のオンライン REDO ログ・ファイルをアーカイブする

プライマリ・データベースで、ALTER SYSTEM SWITCH LOGFILE 文を発行して、ログ・スイッチを強制実行し、現行のオンライン REDO ログ・ファイル・グループをアーカイブします。

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

手順 3 新しい REDO データがスタンバイ・データベースでアーカイブされたことを確認する

スタンバイ・データベースで V$ARCHIVED_LOG ビューを問い合せて、スタンバイ・データベースで REDO データが受信およびアーカイブされたことを確認します。

```
SQL> SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME
  2>  FROM V$ARCHIVED_LOG ORDER BY SEQUENCE#;

 SEQUENCE# FIRST_TIME         NEXT_TIME
---------- ------------------ ------------------
         8 11-JUL-07 17:50:45 11-JUL-07 17:50:53
         9 11-JUL-07 17:50:53 11-JUL-07 17:50:58
        10 11-JUL-07 17:50:58 11-JUL-07 17:51:03
        11 11-JUL-07 17:51:03 11-JUL-07 18:34:11
4 rows selected.
```

アーカイブ REDO ログ・ファイルは、フィジカル・スタンバイ・データベースに適用できます。

手順 4 受信した REDO の適用を確認する

スタンバイ・データベースで V$ARCHIVED_LOG ビューを問い合せて、受信した REDO が適用されたことを確認します。

```
SQL> SELECT SEQUENCE#,APPLIED FROM V$ARCHIVED_LOG
  2  ORDER BY SEQUENCE#;

SEQUENCE# APP
--------- ---
        8 YES
        9 YES
       10 YES
       11 IN-MEMORY
```

4 rows selected.

注意： 最後に受信したログ・ファイルの APPLIED 列の値は、そのログ・ファイルが適用されている場合、IN-MEMORY または YES のいずれかです。

3.3 作成後の手順

この時点で、フィジカル・スタンバイ・データベースが実行中であり、最大パフォーマンス・レベルのデータ保護を提供できます。次のリストに、フィジカル・スタンバイ・データベースに対して実行できるその他の準備について説明します。

- データ保護モードのアップグレード

  Data Guard 構成は、最初は最大パフォーマンス・モード（デフォルト）で設定されます。

- フラッシュバック・データベースの有効化

  フラッシュバック・データベースにより、フェイルオーバー後のプライマリ・データベースの再作成の必要がなくなります。フラッシュバック・データベースでは、データファイルをバックアップからリストアしたり REDO データを広範囲に適用する必要がないため、従来の Point-in-Time リカバリよりもはるかに高速でデータベースを過去の最新時点の状態に戻すことができます。フラッシュバック・データベースは、プライマリ・データベースまたはスタンバイ・データベース、あるいはその両方で有効化できます。Data Guard 環境でのフラッシュバック・データベースの使用例は、13.2 項および 13.3 項を参照してください。また、フラッシュバック・データベースの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

4

ロジカル・スタンバイ・データベースの作成

この章では、ロジカル・スタンバイ・データベースを作成する手順について説明します。この章は、次の主な項目で構成されています。

- ロジカル・スタンバイ・データベースの作成要件
- ロジカル・スタンバイ・データベースの作成手順
- 作成後の手順

関連項目：

- サーバー・パラメータ・ファイルの作成方法および使用方法は、『Oracle Database 管理者ガイド』を参照してください。
- グラフィカル・ユーザー・インタフェースを使用してロジカル・スタンバイ・データベースを自動的に作成する方法は、『Oracle Data Guard Broker』および Oracle Enterprise Manager のオンライン・ヘルプを参照してください。

4.1 ロジカル・スタンバイ・データベースの作成要件

ロジカル・スタンバイ・データベースを作成する前に、プライマリ・データベースが正しく構成されていることを最初に確認する必要があります。表 4-1 は、ロジカル・スタンバイ・データベースを作成するための準備としてプライマリ・データベースで実行するタスクのチェックリストです。

表 4-1 ロジカル・スタンバイ・データベースを作成するためのプライマリ・データベースの準備

参照先	タスク
4.1.1 項	表のデータ型および記憶域属性のサポートの判別
4.1.2 項	プライマリ・データベース内の表の行が一意に識別できることの確認

ロジカル・スタンバイ・データベースでは、スタンバイ REDO ログ（SRL）をプライマリ・データベースから受信した REDO に使用しますが、スタンバイ・データベースに変更を適用するため、オンライン REDO ログ（ORL）にも書き込みます。そのため、ロジカル・スタンバイ・データベースには、SRL および ORL を同時にアーカイブするために追加の ARC*n* プロセスが必要になることがあります。さらに、ORL のアーカイブは SRL のアーカイブに優先するため、ワークロードが非常に高いときは、ロジカル・スタンバイにより多くの SRL が必要になる可能性があります。

4.1.1 表のデータ型および記憶域属性のサポートの判別

ロジカル・スタンバイ・データベースを設定する前に、ロジカル・スタンバイ・データベースが、プライマリ・データベースのデータ型と表を保持できることを確認する必要があります。データ型および記憶域タイプに関する考慮事項の詳細リストは、付録 C を参照してください。

4.1.2 プライマリ・データベース内の表の行が一意に識別できることの確認

ロジカル・スタンバイ・データベースがプライマリ・データベースのバックアップ・コピーから作成されていても、ロジカル・スタンバイ・データベースの物理的な構成は、プライマリ・データベースの構成とは異なります。そのため、プライマリ・データベースによって生成された REDO レコードに含まれている ROWID は、ロジカル・スタンバイ・データベース内の対応する行を識別するためには使用できません。

Oracle では、ロジカル・スタンバイ・データベース内の変更された行を論理的に識別するために、主キーまたは一意制約 / 索引サプリメンタル・ロギングを使用します。また、データベース全体の主キーおよび一意制約 / 索引サプリメンタル・ロギングが使用可能になると、各 UPDATE 文により、ロジカル・スタンバイ・データベース内の変更された行を一意に識別するために、REDO ログに必要な列の値が書き込まれます。

- 表に主キーが定義されている場合、その主キーが、変更された行を識別するために UPDATE 文の一部として変更された列とともにログに記録されます。
- 主キーがない場合は、一番短く、NULL 値が許容されていない一意制約 / 索引が、変更された行を識別するために UPDATE 文の一部として変更された列とともにログに記録されます。
- 主キーも NULL 値が許容されていない一意制約 / 索引もない場合、バインドされているサイズのすべての列が、変更された行を識別するために UPDATE 文の一部としてログに記録されます。つまり、LONG、LOB、LONG RAW、オブジェクト型およびコレクション以外のすべての列が、ログに記録されます。
- ファンクション索引は、一意として宣言されている場合でも、変更された行を一意に識別するためには使用できません。しかし、ロジカル・スタンバイ・データベースでは、変更された行を一意に識別できるかぎり、ファンクション索引が定義された表をレプリケーションできます。

SQL Apply で、REDO データの更新をロジカル・スタンバイ・データベースに効率的に適用できるように、適切で可能な場合には、必ずプライマリ・データベースの表に主キーまたは NULL 値が許容されていない一意索引を追加することをお薦めします。

ロジカル・スタンバイ・データベース内のレプリケートされている各表の行を SQL Apply によって一意に識別できるかどうかを確認するには、次の手順を実行します。

手順 1 プライマリ・データベース内の一意ロジカル識別子のない表を検索する

SQL Apply で一意に識別できない可能性がある表のリストを表示するには、DBA_LOGSTDBY_NOT_UNIQUE ビューを問い合せます。次に例を示します。

```
SQL> SELECT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_NOT_UNIQUE
  2> WHERE (OWNER, TABLE_NAME) NOT IN
  3> (SELECT DISTINCT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED)
  4> AND BAD_COLUMN = 'Y'
```

手順 2 無効化された RELY 主キー制約を追加する

アプリケーションで、表内の行が一意であることが保証される場合は、表に無効化された RELY 主キー制約を作成できます。これによって、プライマリ・データベースでの主キーのメンテナンスに関するオーバーヘッドを回避できます。

無効化された RELY 制約をプライマリ・データベース表に作成するには、RELY DISABLE 句を指定して ALTER TABLE 文を使用します。次の例では、無効化された RELY 制約が mytab という表に作成されます。各行は、id 列と name 列を使用して一意に識別されます。

```
SQL> ALTER TABLE mytab ADD PRIMARY KEY (id, name) RELY DISABLE;
```

RELY 制約を指定すると、システムでは行が一意であると仮定されます。システムには情報に依存するように指示を出しますが、表が変更されるたびに検証は行わないため、表内の各行を一意に識別する RELY 制約が無効化されている場合は、十分に注意して列を選択する必要があります。このような一意性が存在しない場合、SQL Apply による表のメンテナンスが正しく行われません。

SQL Apply のパフォーマンスを改善するには、ロジカル・スタンバイ・データベースで行を識別するために一意制約 / 索引を列に追加します。追加できない場合は、SQL Apply によって表上で UPDATE または DELETE 文を実行中に、全表スキャンが実行されます。

関連項目：

- DBA_LOGSTDBY_NOT_UNIQUE ビューの詳細は、『Oracle Database リファレンス』を参照してください。
- ALTER TABLE 文の構文および RELY 制約の作成の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
- RELY 制約およびロジカル・スタンバイ・データベース上でパフォーマンスを向上させるために行うことができる処理の詳細は、10-27 ページの 10.7.1 項「主キー RELY 制約の作成」を参照してください。

4.2 ロジカル・スタンバイ・データベースの作成手順

この項では、ロジカル・スタンバイ・データベースの作成で実行するタスクについて説明します。

表 4-2 は、ロジカル・スタンバイ・データベースの作成で実行するタスクのチェックリストで、各タスクを実行するデータベースを指定します。各タスクを詳細に説明している参照先の項も記載されています。

表 4-2 ロジカル・スタンバイ・データベースの作成

参照先	タスク	データベース
4.2.1 項	フィジカル・スタンバイ・データベースの作成	プライマリ
4.2.2 項	フィジカル・スタンバイ・データベースでの REDO Apply の停止	スタンバイ
4.2.3 項	ロジカル・スタンバイ・データベースをサポートするためのプライマリ・データベースの準備	プライマリ
4.2.4 項	ロジカル・スタンバイ・データベースへの推移	スタンバイ
4.2.5 項	ロジカル・スタンバイ・データベースのオープン	スタンバイ
4.2.6 項	ロジカル・スタンバイ・データベースが正しく実行されているかどうかの確認	スタンバイ

4.2.1 フィジカル・スタンバイ・データベースの作成

ロジカル・スタンバイ・データベースを作成するには、次のように最初にフィジカル・スタンバイ・データベースを作成してから、ロジカル・スタンバイ・データベースへと推移させます。第 3 章「フィジカル・スタンバイ・データベースの作成」の指示に従ってフィジカル・スタンバイ・データベースを作成します。

4.2.2 フィジカル・スタンバイ・データベースでの REDO Apply の停止

ロジカル・スタンバイ・データベースに変換する前に、実行時間に関係なく、新規フィジカル・スタンバイ・データベースで REDO Apply を実行できます。ただし、ロジカル・スタンバイ・データベースに変換する前に、フィジカル・スタンバイ・データベースで REDO Apply を停止する必要があります。REDO Apply の停止は、LogMiner ディクショナリを含む REDO の後に変更が適用されることを避けるために必要です（4-6 ページの 4.2.3.2 項「REDO データでのディクショナリの構築」を参照）。

REDO Apply を停止するには、フィジカル・スタンバイ・データベースで次の文を発行します。データベースが複数のインスタンスから構成される Oracle RAC データベースの場合、この文を発行する前に、1 つを残してすべての Oracle RAC インスタンスを停止する必要があります。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

4.2.3 ロジカル・スタンバイ・データベースをサポートするためのプライマリ・データベースの準備

この項は、次の項目で構成されています。

- ロールの推移のためのプライマリ・データベースの準備
- REDO データでのディクショナリの構築

4.2.3.1 ロールの推移のためのプライマリ・データベースの準備

3-3 ページの 3.1.4 項「プライマリ・データベースの初期化パラメータの設定」では、プライマリ・データベースがフィジカル・スタンバイ・ロールに推移する場合に有効になるように、複数のスタンバイ・ロールの初期化パラメータを設定しました。

注意： この手順は、スイッチオーバーを実行する予定の場合のみ必要です。

プライマリ・データベースをロジカル・スタンバイ・ロールに推移させる場合は、ロールの推移後にパラメータを変更しなくてもよいように、例 4-1 に太字で示すパラメータを変更する必要もあります。

- 元の LOG_ARCHIVE_DEST_1 宛先の VALID_FOR 属性を、スタンバイ REDO ログからではなく、オンライン REDO ログからのみ REDO データをアーカイブするように変更します。
- LOG_ARCHIVE_DEST_3 宛先をプライマリ・データベースに含めます。このパラメータは、プライマリ・データベースがロジカル・スタンバイ・ロールに推移したときにのみ有効になります。

例 4-1 プライマリ・データベース：ロジカル・スタンバイ・ロールの初期化パラメータ

```
LOG_ARCHIVE_DEST_1=
 'LOCATION=/arch1/chicago/
  VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=chicago'
LOG_ARCHIVE_DEST_3=
 'LOCATION=/arch2/chicago/
  VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE)
  DB_UNIQUE_NAME=chicago'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

これらの初期化パラメータを動的に設定するには、SQL ALTER SYSTEM SET 文を使用し、変更が即時に有効になってデータベースを停止して再起動した後も存続するように SCOPE=BOTH 句を指定します。

次の表に、例 4-1 に示した変更後の初期化パラメータで定義されるアーカイブ・プロセスを示します。

	シカゴのデータベースがプライマリ・ロールで稼働している場合	シカゴのデータベースがロジカル・スタンバイ・ロールで稼働している場合
LOG_ARCHIVE_DEST_1	/arch1/chicago/ 内のローカル・アーカイブ REDO ログ・ファイルにローカル・オンライン REDO ログ・ファイルからプライマリ・データベースによって生成された REDO データをアーカイブするよう指示。	/arch1/chicago/ 内のローカル・アーカイブ REDO ログ・ファイルにローカル・オンライン REDO ログ・ファイルからロジカル・スタンバイ・データベースによって生成された REDO データをアーカイブするよう指示。
LOG_ARCHIVE_DEST_3	無視。LOG_ARCHIVE_DEST_3 は、chicago がスタンバイ・ロールで稼働している場合にのみ有効。	/arch2/chicago/ 内のローカル・アーカイブ REDO ログ・ファイルにスタンバイ REDO ログ・ファイルからの REDO データをアーカイブするよう指示。

4.2.3.2 REDO データでのディクショナリの構築

LogMiner ディクショナリは、SQL Apply の LogMiner コンポーネントによって REDO での変更が適切に解釈できるように、REDO データに構築される必要があります。LogMiner ディクショナリの構築の一部として、主キーおよび一意制約 / 索引列を記録するサプリメンタル・ロギングが自動的に設定されます。サプリメンタル・ロギング情報により、各更新に、文によって変更された各行を論理的に識別するための十分な情報が含まれていることが保証されます。

LogMiner ディクショナリを構築するには、次の文を発行します。

```
SQL> EXECUTE DBMS_LOGSTDBY.BUILD;
```

DBMS_LOGSTDBY.BUILD プロシージャでは、既存のトランザクションがすべて完了するのを待機します。プライマリ・データベースで長時間実行されているトランザクションは、このコマンドの適時性に影響を与えます。

関連項目：

- 『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_LOGSTDBY.BUILD PL/SQL パッケージに関する項
- 『Oracle Database リファレンス』の UNDO_RETENTION 初期化パラメータに関する項

4.2.4 ロジカル・スタンバイ・データベースへの推移

この項では、ロジカル・スタンバイ・データベースに推移するために、フィジカル・スタンバイ・データベースを準備する方法について説明します。この章は、次の項目で構成されています。

- ロジカル・スタンバイ・データベースへの変換
- ロジカル・スタンバイ・データベース用の初期化パラメータの調整

4.2.4.1 ロジカル・スタンバイ・データベースへの変換

REDO ログには、フィジカル・スタンバイ・データベースをロジカル・スタンバイ・データベースに変換するために必要な情報が含まれています。

注意： Oracle RAC フィジカル・スタンバイ・データベースがある場合、1 つのインスタンス以外はすべて停止して、CLUSTER_DATABASE を FALSE に設定し、次のようにしてスタンバイ・データベースを MOUNT EXCLUSIVE モードで単一インスタンスとして起動します。

```
SQL> ALTER SYSTEM SET CLUSTER_DATABASE=FALSE SCOPE=SPFILE;
SQL> SHUTDOWN ABORT;
SQL> STARTUP MOUNT EXCLUSIVE;
```

ロジカル・スタンバイ・データベースへの変換準備が完了するまで REDO データをフィジカル・スタンバイ・データベースに適用し続けるには、次の SQL 文を発行します。

```
SQL> ALTER DATABASE RECOVER TO LOGICAL STANDBY db_name;
```

db_name には、新しいロジカル・スタンバイ・データベースを識別するデータベース名を指定します。この文の発行時にサーバー・パラメータ・ファイル（spfile）を使用している場合、データベースは、新しいロジカル・スタンバイ・データベースに関する適切な情報でこのファイルを更新します。spfile を使用していない場合は、データベースの停止後に DB_NAME パラメータの名前を設定するよう求めるメッセージが発行されます。

注意： フィジカル・スタンバイ・データベースで Oracle ソフトウェアのローリング・アップグレードを実行する状況で、ロジカル・スタンバイ・データベースを作成するとき、次のコマンドをかわりに発行する必要があります。

```
SQL> ALTER DATABASE RECOVER TO LOGICAL STANDBY KEEP IDENTITY;
```

KEEP IDENTITY 句を指定して作成されたロジカル・スタンバイ・データベースは、プライマリ・データベースと同じ DB_NAME および DBID を保持します。このようなロジカル・スタンバイ・データベースは、1 回のスイッチオーバー操作にしか使用できないため、フィジカル・スタンバイ・データベースでローリング・アップグレードを実行する状況でのみ作成する必要があります。

KEEP IDENTITY 句は、アップグレードするデータベースで Oracle Database リリース 11.1 以降が稼働している場合にのみ、使用できます。

この文では、ログ・ファイル内に LogMiner ディクショナリが見つかるまで、REDO データを適用しながら待機します。4.2.3.2 項「REDO データでのディクショナリの構築」で生成された REDO がスタンバイ・データベースに推移されるのにかかる時間と、適用する必要のある REDO データの量にもよりますが、この作業には数分かかります。プライマリ・データベースでディクショナリ構築が正常に実行されるまで、このコマンドは完了しません。SQL 文を取り消すには、別の SQL セッションから ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL 文を発行します。

注意： 旧リリースでは、ロジカル・スタンバイ・データベースをオープンする前に新しいパスワード・ファイルを作成する必要がありました。現在は、その必要はありません。ロジカル・スタンバイ・データベースに新しいパスワード・ファイルを作成すると、REDO 転送サービスが正常に機能しなくなります。

4.2.4.2 ロジカル・スタンバイ・データベース用の初期化パラメータの調整

注意： Oracle RAC フィジカル・スタンバイ・データベースで開始した場合、次のように CLUSTER_DATABASE の設定を TRUE に戻します。

```
SQL> ALTER SYSTEM SET CLUSTER_DATABASE=TRUE SCOPE=SPFILE;
```

ロジカル・スタンバイ・データベースで、インスタンスを停止し、STARTUP MOUNT 文を発行してデータベースを起動およびマウントします。データベースをオープンしないでください。後続の作成プロセスまで、データベースはユーザー・アクセスに対してクローズの状態のままにしておく必要があります。次に例を示します。

```
SQL> SHUTDOWN;
SQL> STARTUP MOUNT;
```

LOG_ARCHIVE_DEST_*n* パラメータを変更する必要があります。これは、フィジカル・スタンバイ・データベースとは異なり、ロジカル・スタンバイ・データベースは REDO データを生成するオープン・データベースであり、複数のログ・ファイル（オンライン REDO ログ・ファイル、アーカイブ REDO ログ・ファイルおよびスタンバイ REDO ログ・ファイル）があるためです。次のものに対し、別々のローカル宛先を指定することをお薦めします。

- ロジカル・スタンバイ・データベースで生成された REDO データを格納するアーカイブ REDO ログ・ファイル。例 4-2 では、LOG_ARCHIVE_DEST_1=LOCATION=/arch1/boston 宛先として構成されています。
- プライマリ・データベースから受信した REDO データを格納するアーカイブ REDO ログ・ファイル。例 4-2 では、LOG_ARCHIVE_DEST_3=LOCATION=/arch2/boston 宛先として構成されています。

例 4-2 に、ロジカル・スタンバイ・データベース用に変更された初期化パラメータを示します。各パラメータは、ボストンのロジカル・スタンバイ・データベースがプライマリ・データベース・ロールまたはスタンバイ・データベース・ロールで実行されている場合に有効です。

例 4-2 ロジカル・スタンバイ・データベース用の初期化パラメータの変更

```
LOG_ARCHIVE_DEST_1=
  'LOCATION=/arch1/boston/
   VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES)
   DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_2=
  'SERVICE=chicago ASYNC
   VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
   DB_UNIQUE_NAME=chicago'
LOG_ARCHIVE_DEST_3=
  'LOCATION=/arch2/boston/
   VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE)
   DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

注意： データベースの互換性を 11.1 に設定した場合、フラッシュ・リカバリ領域を使用してリモート・アーカイブ・ログを格納することもできます。そのためには、次のパラメータを設定します
(すでに DB_RECOVERY_FILE_DEST および
DB_RECOVERY_FILE_DEST_SIZE は適切に設定されているとします)。

```
LOG_ARCHIVE_DEST_1=
  'LOCATION=USE_DB_RECOVERY_FILE_DEST
   VALID_FOR=(ONLINE_LOGFILES, ALL_ROLES)
   DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_3=
  'LOCATION=USE_DB_RECOVERY_FILE_DEST
   VALID_FOR=(STANDBY_LOGFILES, STANDBY_ROLE)
   DB_UNIQUE_NAME=boston'
```

次の表に、例 4-2 に示した初期化パラメータで定義されるアーカイブ・プロセスを示します。

	ボストンのデータベースがプライマリ・ロールで稼働している場合	ボストンのデータベースがロジカル・スタンバイ・ロールで稼働している場合
LOG_ARCHIVE_DEST_1	/arch1/boston/ 内のローカル・アーカイブ REDO ログ・ファイルにローカル・オンライン REDO ログ・ファイルからプライマリ・データベースによって生成された REDO データをアーカイブするよう指示。	/arch1/boston/ 内のローカル・アーカイブ REDO ログ・ファイルにローカル・オンライン REDO ログ・ファイルからロジカル・スタンバイ・データベースによって生成された REDO データをアーカイブするよう指示。
LOG_ARCHIVE_DEST_2	リモート・ロジカル・スタンバイ・データベース chicago に REDO データを転送するよう指示。	無視。LOG_ARCHIVE_DEST_2 は、boston がプライマリ・ロールで稼働している場合にのみ有効。
LOG_ARCHIVE_DEST_3	無視。LOG_ARCHIVE_DEST_3 は、boston がスタンバイ・ロールで稼働している場合にのみ有効。	/arch2/boston/ 内のローカル・アーカイブ REDO ログ・ファイルにプライマリ・データベースから受信した REDO データをアーカイブするよう指示。

注意： DB_FILE_NAME_CONVERT 初期化パラメータは、フィジカル・スタンバイ・データベースがロジカル・スタンバイ・データベースに変換された後は考慮されません。必要な場合は、スキップ・ハンドラを登録し、プライマリ・データベースのデータファイルのパス名をスタンバイ・データファイルのパス名に変換することで、実行対象のかわりとなる DDL 文字列を SQL Apply に提供します。SKIP プロシージャの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_LOGSTDBY パッケージに関する項を参照してください。

4.2.5 ロジカル・スタンバイ・データベースのオープン

新規ロジカル・スタンバイ・データベースをオープンするには、次の文を発行し、RESETLOGS オプションを使用してオープンする必要があります。

```
SQL> ALTER DATABASE OPEN RESETLOGS;
```

注意： Oracle RAC フィジカル・スタンバイ・データベースで開始した場合、その他のすべてのスタンバイ・インスタンスをこの時点で開始できます。

注意： ロジカル・スタンバイ・データベースをプライマリ・データベースと同じコンピュータ・システム上に配置している場合、SQL Apply を初めて開始する前に次の SQL 文を発行し、SQL Apply がプライマリ・データベースで実行されるファイル操作をスキップするようにする必要があります。この文の発行が必要な理由は、SQL Apply はプライマリ・データベースと同じディレクトリ構造にアクセスできるので、特定のファイル固有の操作を再実行しようとした場合に、プライマリ・データベースに属するデータファイルが破損する可能性があるためです。

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP('ALTER TABLESPACE');
```

フィジカル・データベースをプライマリ・データベースと同じシステム上に配置する際に設定した DB_FILENAME_CONVERT パラメータは、SQL Apply では無視されます。DBMS_LOGSTDBY.SKIP およびロジカル・スタンバイ・データベースの下での同等の動作の詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

データベースがオープンされるのはこれが初回であるため、データベースのグローバル名は、新しい DB_NAME 初期化パラメータと一致するように自動的に調整されます。

次の文を発行して、ロジカル・スタンバイ・データベースに対する REDO データの適用を開始します。次に例を示します。

```
SQL>  ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

4.2.6 ロジカル・スタンバイ・データベースが正しく実行されているかどうかの確認

ロジカル・スタンバイ・データベースが適切に実行されていることを検証するには、次の項を参照してください。

- 第 6 章「REDO 転送サービス」
- 第 10 章「ロジカル・スタンバイ・データベースの管理」

4.3 作成後の手順

この時点で、ロジカル・スタンバイ・データベースが実行中であり、最大パフォーマンス・レベルのデータ保護を提供できます。次のリストに、ロジカル・スタンバイ・データベースに対して実行できるその他の準備について説明します。

- データ保護モードのアップグレード

  Data Guard 構成は、最初は最大パフォーマンス・モード（デフォルト）で設定されます。

- フラッシュバック・データベースの有効化

  フラッシュバック・データベースにより、フェイルオーバー後のプライマリ・データベースの再作成の必要がなくなります。フラッシュバック・データベースでは、データファイルをバックアップからリストアしたり REDO データを広範囲に適用する必要がないため、従来の Point-in-Time リカバリよりもはるかに高速でデータベースを過去の最新時点の状態に戻すことができます。フラッシュバック・データベースは、プライマリ・データベースまたはスタンバイ・データベース、あるいはその両方で有効化できます。Data Guard 環境でのフラッシュバック・データベースの使用例は、13-5 ページの 13.2 項「フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換」、および 13-8 ページの 13.3 項「Open Resetlogs 文の発行後のフラッシュバック・データベースの使用」を参照してください。また、フラッシュバック・データベースの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

5

Data Guard の保護モード

この章は、次の項目で構成されています。

5.1 Data Guard の保護モード

この項では、Data Guard の保護モードについて説明します。

ここでの説明では、同期化されたスタンバイ・データベースは、構成されたデータ保護モードの最低要件を満たし、REDO ギャップがないデータベースとして作成されています。REDO ギャップの詳細は、6.3.3 項を参照してください。

最大可用性

この保護モードは、プライマリ・データベースの可用性を低下させない範囲で可能な最高レベルのデータ保護を提供します。トランザクションは、そのトランザクションのリカバリに必要なすべての REDO データが、オンライン REDO ログおよび少なくとも 1 つの同期化されたスタンバイ・データベースに書き込まれるまでコミットされません。プライマリ・データベースは、REDO ストリームを少なくとも 1 つの同期化されたスタンバイ・データベースに書き込むことができない場合、REDO ストリームを同期化されたスタンバイ・データベースに再び書き込めるようになるまで、最大パフォーマンス・モードにあるかのように動作してプライマリ・データベースの可用性を維持します。

このモードでは、プライマリ・データベースに障害が発生した場合、ただし、2 回目の障害でプライマリ・データベースから少なくとも 1 つのスタンバイ・データベースに REDO データの完全なセットが送信される場合にのみ、データが消失しないことを保証します。

最大パフォーマンス

この保護モードは、プライマリ・データベースのパフォーマンスに影響しない範囲で可能な最高レベルのデータ保護を提供します。これは、トランザクションによって生成されたすべての REDO データがオンライン・ログに書き込まれた直後に、そのトランザクションのコミットを可能にすることで実現されます。REDO データは、1 つ以上のスタンバイ・データベースにも書き込まれますが、トランザクション・コミットについて非同期で行われるため、プライマリ・データベースのパフォーマンスは、スタンバイ・データベースへの REDO データの書込み遅延による影響を受けません。

この保護モードは、最大可用性モードに比べてデータ保護が若干弱く、プライマリ・データベースのパフォーマンスへの影響を最小限に抑えます。

これはデフォルトの保護モードです。

最大保護

この保護モードは、プライマリ・データベースに障害が発生した場合でも、データ消失がないことを保証します。このレベルの保護を提供するには、トランザクションがコミットされる前に、トランザクションのリカバリに必要な REDO データを、オンライン REDO ログおよび少なくとも 1 つの同期化されたスタンバイ・データベースに書き込む必要があります。少なくとも 1 つの同期化されたスタンバイ・データベースに REDO ストリームを書き込むことができない場合は、データ消失が発生しないように、プライマリ・データベースは停止し、トランザクションの処理を続行しません。

このデータ保護モードは、プライマリ・データベースの可用性よりもデータ保護を優先するため、2 つ以上のスタンバイ・データベースを使用して、最大保護モードで稼働するプライマリ・データベースを保護し、1 つのスタンバイ・データベースの障害が原因でプライマリ・データベースが停止しないようにします。

5.2 プライマリ・データベースのデータ保護モードの設定

次の手順を実行して、プライマリ・データベースのデータ保護モードを変更します。

手順 1 可用性、パフォーマンスおよびデータ保護の要件を満たすデータ保護モードを選択する

選択可能なモードの詳細は、5.1 項を参照してください。

手順 2 1 つ以上のスタンバイ・データベースで REDO 転送が構成されていることを確認する

スタンバイ・データベースに対応する LOG_ARCHIVE_DEST_*n* データベース初期化パラメータの値に、移行後のデータ保護モードに対する REDO 転送の属性（表 5-1 を参照）が指定されている必要があります。

プライマリ・データベースに複数のスタンバイ・データベースがある場合、表 5-1 に示す REDO 転送の設定を使用する必要があるのは、それらのスタンバイ・データベースの中の 1 つのみです。

また、スタンバイ・データベースにはスタンバイ REDO ログが必要です。

REDO 転送およびスタンバイ REOD ログの構成の詳細は、第 6 章「REDO 転送サービス」を参照してください。

表 5-1 データ保護モードに対する REDO 転送の必須属性

最大可用性	最大パフォーマンス	最大保護
AFFIRM	NOAFFIRM	AFFIRM
SYNC	ASYNC	SYNC
DB_UNIQUE_NAME	DB_UNIQUE_NAME	DB_UNIQUE_NAME

手順 3 プライマリ・データベースおよびスタンバイ・データベースで DB_UNIQUE_NAME データベース初期化パラメータが一意の名前に設定されていることを確認する

たとえば、いずれのデータベースにも DB_UNIQUE_NAME パラメータが定義されていない場合、次の SQL 文を使用して各データベースに一意の名前を割り当てることができます。

次の SQL 文はプライマリ・データベースで実行します。

```
SQL> ALTER SYSTEM SET DB_UNIQUE_NAME='CHICAGO' SCOPE=SPFILE;
```

次の SQL 文はスタンバイ・データベースで実行します。

```
SQL> ALTER SYSTEM SET DB_UNIQUE_NAME='BOSTON' SCOPE=SPFILE;
```

手順 4 LOG_ARCHIVE_CONFIG データベース初期化パラメータがプライマリ・データベースおよびスタンバイ・データベースに定義されていて、プライマリ・データベースおよびスタンバイ・データベースの DB_UNIQUE_NAME を含む DG_CONFIG リストが、その値に指定されていることを確認する

たとえば、いずれのデータベースにも LOG_ARCHIVE_CONFIG パラメータが定義されていない場合、次の SQL 文を各データベースで実行して LOG_ARCHIVE_CONFIG パラメータを構成できます。

```
SQL> ALTER SYSTEM SET
  2> LOG_ARCHIVE_CONFIG='DG_CONFIG=(CHICAGO,BOSTON)';
```

手順 5 保護モードを最大保護に設定する、または最大パフォーマンスから最大可用性に変更する場合は、プライマリ・データベースを停止し、マウント済モードで再起動する。プライマリ・データベースが Oracle Real Application Clusters である場合は、すべてのインスタンスを停止し、1 つのインスタンスのみを起動およびマウントする

次に例を示します。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
```

手順 6 データ保護モードを設定する

次の SQL 文はプライマリ・データベースで実行します。

```
SQL> ALTER DATABASE
  2> SET STANDBY DATABASE TO MAXIMIZE {AVAILABILITY | PERFORMANCE | PROTECTION};
```

プライマリ・データベースが Oracle Real Application Clusters である場合は、この時点で手順 5 で停止したインスタンスを再起動できます。

手順 7 プライマリ・データベースをオープンする

手順 5 でデータベースを再起動した場合は、データベースをオープンします。

```
SQL> ALTER DATABASE OPEN;
```

手順 8 プライマリ・データベースが新しい保護モードで動作していることを確認する

次の問合せをプライマリ・データベースで実行し、新しい保護モードで動作していることを確認します。

```
SQL> SELECT PROTECTION_MODE FROM V$DATABASE;
```

6

REDO 転送サービス

この章では、Oracle REDO 転送サービスを構成および監視する方法について説明します。次の項目で構成されています。

- REDO 転送サービスの概要
- REDO 転送サービスの構成
- REDO 転送サービスの監視
- REDO 転送のチューニング

6.1 REDO 転送サービスの概要

REDO 転送サービスは、Oracle データベース間で REDO データの自動転送を実行します。次の REDO 転送先がサポートされています。

- Oracle Data Guard スタンバイ・データベース

  このマニュアルでは、スタンバイ・データベースを作成および管理する方法について説明します。

- アーカイブ・ログ・リポジトリ

  この宛先タイプは、アーカイブ REDO ログ・ファイルの一時オフサイト記憶域に使用されます。アーカイブ・ログ・リポジトリは、Oracle データベース・インスタンスとフィジカル・スタンバイ制御ファイルで構成されます。アーカイブ・ログ・リポジトリにはデータファイルが含まれないため、ロールの推移はサポートできません。

  アーカイブ・ログ・リポジトリの作成に使用される手順は、データファイルのコピーを除いて、フィジカル・スタンバイ・データベースの作成に使用される手順と同じです。

- Oracle Streams ダウンストリーム取得データベース

  Oracle Streams ダウンストリーム取得データベースの詳細は、『Oracle Streams 概要および管理』を参照してください。

- Oracle Change Data Capture ステージング・データベース

  Oracle Change Data Capture ステージング・データベースの詳細は、『Oracle Warehouse Builder ユーザーズ・ガイド』を参照してください。

Oracle データベースでは、REDO データを最大 9 個の REDO 転送先に送信できます。各 REDO 転送先は、次の 2 つの REDO 転送モードのいずれかにより REDO データを受信するように、個別に構成されます。

- 同期

  同期 REDO 転送モードは、REDO データをトランザクション・コミットに対して同期させて転送します。トランザクションによって生成されたすべての REDO が、同期 REDO 転送モードを使用する使用可能な REDO 転送先すべてに正常に送信されるまで、そのトランザクションはコミットできません。

  この転送モードは、最大保護および最大可用性の各データ保護モード（第 5 章「Data Guard の保護モード」を参照）で使用されます。

- 非同期

  非同期 REDO 転送モードは、REDO データをトランザクション・コミットに対して非同期で転送します。トランザクションによって生成されたすべての REDO が、非同期 REDO 転送モードを使用する REDO 転送先すべてに正常に送信されるまで待機しなくても、そのトランザクションはコミットできます。

  この転送モードは、最大パフォーマンス・データ保護モード（第 5 章「Data Guard の保護モード」を参照）で使用されます。

6.2 REDO 転送サービスの構成

この項では、REDO 転送サービスの構成方法について説明します。次の項目で構成されています。

- REDO 転送のセキュリティ
- REDO データを送信するための Oracle データベースの構成
- REDO データを受信するための Oracle データベースの構成

この項では、読者が次の項目（『Oracle Database 管理者ガイド』および『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照）を十分に理解していることを前提にしたレベルで説明します。

- データベース管理者の認証
- データベース初期化パラメータ
- REDO ログの管理
- アーカイブ REDO ログの管理
- フラッシュ・リカバリ領域

6.2.1 REDO 転送のセキュリティ

REDO 転送では、Oracle Net セッションを使用して REDO データを転送します。これらの REDO 転送セッションは、Secure Sockets Layer（SSL）プロトコルまたはリモート・ログイン・パスワード・ファイルのいずれかを使用して認証されます。

6.2.1.1 SSL を使用した REDO 転送の認証

Secure Sockets Layer（SSL）は、ネットワーク接続を保護するための業界標準プロトコルです。SSL では、RSA 公開鍵暗号化および対称鍵暗号化を使用して、認証、暗号化およびデータ整合性を提供します。SSL は、次の場合に 2 つの Oracle データベース間の REDO 転送の認証に自動的に使用されます。

- データベースが同じ Oracle Internet Directory（OID）エンタープライズ・ドメインのメンバーであり、そのドメインが現行ユーザーのデータベース・リンクの使用を許可する場合
- データベースに対応する `LOG_ARCHIVE_DEST_n`、`FAL_SERVER` および `FAL_CLIENT` の各データベース初期化パラメータで、SSL 用に構成された Oracle Net 接続記述子を使用する場合
- 各データベースに、データベースの OID エントリの識別名（DN）と一致する DN が付けられたユーザー証明書を格納する Oracle ウォレットまたはサポートされるハードウェア・セキュリティ・モジュールがある場合

関連項目：

- SSL 認証の構成の詳細は、『Oracle Database Advanced Security 管理者ガイド』を参照してください。
- エンタープライズ・ドメインの管理の詳細は、『Oracle Database エンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。
- Oracle Internet Directory の管理の詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

6.2.1.2 パスワード・ファイルを使用した REDO 転送の認証

SSL 認証の要件が満たされない場合、各データベースはリモート・ログイン・パスワード・ファイルを使用する必要があります。Data Guard 構成では、すべてのフィジカル・スタンバイ・データベースおよびスナップショット・スタンバイ・データベースは、プライマリ・データベースのパスワード・ファイルのコピーを使用する必要があります。このコピーは、SYSOPER 権限または SYSDBA 権限が付与または取り消されるたびに、あるいはこれらの権限を持つユーザーのパスワードが変更された後に、リフレッシュする必要があります。

REDO 転送の認証にパスワード・ファイルを使用する場合、REDO 転送の認証に使用されるユーザー・アカウントのパスワードは、REDO 転送セッションを開始したデータベースとターゲット・データベース間で比較されます。REDO 転送セッションを確立するには、パスワードが両方のデータベースで一致する必要があります。

デフォルトでは、パスワード・ファイルを使用する場合、SYS ユーザーのパスワードが REDO 転送セッションの認証に使用されます。REDO_TRANSPORT_USER データベース初期化パラメータを使用して、SYSOPER 権限が付与されたユーザー名を設定すると、REDO 転送の認証用に別のユーザー・パスワードを選択できます。管理しやすくするために、REDO_TRANSPORT_USER パラメータを REDO ソース・データベースおよび各 REDO 転送宛先で同じ値に設定することをお薦めします。

> **関連項目：** リモート・ログイン・パスワード・ファイルの作成およびメンテナンスの詳細は、『Oracle Database 管理者ガイド』を参照してください。

6.2.2 REDO データを送信するための Oracle データベースの構成

この項では、REDO データを REDO 転送先に送信するように Oracle データベースを構成する方法について説明します。

LOG_ARCHIVE_DEST_*n* データベース初期化パラメータ（*n* は 1 ～ 10 の整数）を使用して、ローカルのアーカイブ REDO ログの位置を指定するか、REDO 転送先を指定します。この項では、このパラメータの後者の使用について説明します。

各 LOG_ARCHIVE_DEST_*n* パラメータに対応する LOG_ARCHIVE_DEST_STATE_*n* データベース初期化パラメータ（*n* は 1 ～ 10 の整数）があります。このパラメータを使用すると、対応する REDO 転送先を有効または無効にできます。表 6-1 に、このパラメータに割り当てることができる有効な値を示します。

表 6-1 LOG_ARCHIVE_DEST_STATE_n 初期化パラメータの値

値	説明
ENABLE	REDO 転送サービスはこの宛先に REDO データを転送できる。これはデフォルトです。
DEFER	REDO 転送サービスはこの宛先に REDO データを転送しない。
ALTERNATE	この宛先は、関連付けられている宛先への通信に障害が発生すると使用可能になる。

REDO 転送先は、LOG_ARCHIVE_DEST_*n* パラメータを、1 つ以上の属性を含む文字列に設定して構成します。この項では、最も一般的に使用される属性について簡単に説明します。すべての LOG_ARCHIVE_DEST_*n* パラメータの属性の詳細は、第 15 章を参照してください。

SERVICE 属性は、REDO 転送先の必須属性で、属性リストの最初に指定する必要があります。SERVICE 属性は、REDO 転送先への接続に使用される Oracle Net サービス名の指定に使用されます。Oracle Net サービス名の詳細は、『Oracle Database Net Services 管理者ガイド』を参照してください。

SYNC 属性は、同期 REDO 転送モードを使用して REDO データを REDO 転送先に送信するように指定するために使用します。

ASYNC 属性は、非同期 REDO 転送モードを使用して REDO データを REDO 転送先に送信するように指定するために使用します。非同期 REDO 転送モードは、SYNC 属性と ASYNC 属性のどちらも指定されない場合に使用されます。

NET_TIMEOUT 属性は、LGWR プロセスが、同期 REDO 転送モードを使用する転送先で REDO データが正常に受信されたことの確認待ちをブロックする時間を指定するために使用します。確認が NET_TIMEOUT の秒数内に受信されない場合は、REDO 転送接続は終了し、エラーがログに記録されます。

同期 REDO 転送モードを使用する場合は常に NET_TIMEOUT 属性を指定して、REDO 転送障害による REDO ソース・データベースの最大停止期間を厳密に制御できるようにすることをお薦めします。同期 REDO 転送モードのレスポンス時間の監視の詳細は、6.3.2 項を参照してください。

AFFIRM 属性は、REDO ソース・データベースから受信された REDO が、スタンバイ REDO ログに書き込まれるまで確認されないように指定するために使用します。NOAFFIRM 属性は、受信された REDO が、スタンバイ REDO ログに書き込まれるのを待機せずに確認されるように指定するために使用します。

DB_UNIQUE_NAME 属性は、REDO 転送先の DB_UNIQUE_NAME を指定するために使用します。DB_UNIQUE_NAME 属性は、LOG_ARCHIVE_CONFIG データベース初期化パラメータが定義されていて、その値に DG_CONFIG リストが指定されている場合に指定する必要があります。

DB_UNIQUE_NAME 属性を指定する場合は、その値は DG_CONFIG リストの DB_UNIQUE_NAME 値の 1 つと一致する必要があります。また、REDO 転送先の DB_UNIQUE_NAME データベース初期化パラメータの値とも一致する必要があります。どちらかが一致しない場合は、エラーがログに記録され、その宛先には REDO 転送できません。

VALID_FOR 属性は、REDO 転送サービスが REDO データを REDO 転送先に転送する時期を指定するために使用します。Data Gurad 構成内のすべてのサイトで REDO 転送先ごとに VALID_FOR 属性を指定して、どのスタンバイ・データベースがプライマリ・ロールを担うかに関係なく、ロールの推移後も REDO 転送サービスがすべてのスタンバイ・データベースに引き続き REDO データを送信するようにすることをお薦めします。

REOPEN 属性は、前回のエラーにより非アクティブな REDO 転送先への自動再接続試行の最低間隔（秒数）を指定するために使用します。

COMPRESSION 属性は、REDO データ・ギャップの解決時に、REDO データを REDO 転送先に圧縮して転送するように指定するために使用します。REDO 転送を圧縮すると、帯域幅が小さく待機時間が長いネットワーク・リンクを REDO 転送に使用している場合、REDO ギャップの解決時間が大幅に短縮されます。REDO ギャップの解決の詳細は、6.3.3 項を参照してください。

次の例では、この項で説明した LOG_ARCHIVE_DEST_*n* の属性をすべて使用しています。2 つの REDO 転送先が定義され、使用可能になっています。最初の宛先は、非同期 REDO 転送モードを使用します。2 番目の宛先は、30 秒のタイムアウトを指定した同期 REDO 転送モードを使用します。DB_UNIQUE_NAME は、REDO ギャップの解決時に圧縮を使用するので、両方の宛先に指定されています。REDO 転送障害がいずれかの宛先で発生した場合、REDO 転送はその宛先への再接続を試行しますが、60 秒に 1 回より長い間隔で行います。

```
DB_UNIQUE_NAME=BOSTON
LOG_ARCHIVE_CONFIG='DG_CONFIG=(BOSTON,CHICAGO,DENVER)'
LOG_ARCHIVE_DEST_2='SERVICE=CHICAGO ASYNC NOAFFIRM VALID_FOR=(ONLINE_LOGFILE,
PRIMARY_ROLE) REOPEN=60 COMPRESSION=ENABLE  DB_UNIQUE_NAME=CHICAGO'
LOG_ARCHIVE_DEST_STATE_2='ENABLE'
LOG_ARCHIVE_DEST_3='SERVICE=DENVER SYNC AFFIRM NET_TIMEOUT=30 VALID_FOR=
(ONLINE_LOGFILE,PRIMARY_ROLE) REOPEN=60 COMPRESSION=ENABLE  DB_UNIQUE_NAME=DENVER'
LOG_ARCHIVE_DEST_STATE_3='ENABLE'
```

6.2.2.1 V$ARCHIVE_DEST ビューを使用した属性の表示

V$ARCHIVE_DEST ビューを問い合せると、REDO 転送先ごとに現行の設定およびステータスを確認できます。

6.2.3 REDO データを受信するための Oracle データベースの構成

この項では、REDO ソース・データベースから REDO データを受信およびアーカイブするように、REDO 転送先を構成する方法について説明します。

次の項目で構成されています。

- スタンバイ REDO ログの作成および管理
- スタンバイ REDO ログ・アーカイブの構成

6.2.3.1 スタンバイ REDO ログの作成および管理

同期および非同期の REDO 転送モードでは、REDO 転送先にスタンバイ REDO ログが必要です。スタンバイ REDO ログは、別の Oracle データベースから受信された REDO を格納するために使用します。スタンバイ REDO ログは、構造的には REDO ログと同じで、REDO ログの作成および管理に使用するのと同じ SQL 文を使用して作成および管理します。

別の Oracle データベースから REDO 転送を介して受信された REDO は、RFS バックグラウンド・プロセスにより、現行のスタンバイ REDO ログ・グループに書き込まれます。REDO ソース・データベースでログ・スイッチが発生すると、着信 REDO は次のスタンバイ REDO ログ・グループに書き込まれ、前に使用されたスタンバイ REDO ログ・グループは ARCn バックグラウンド・プロセスによってアーカイブされます。

REDO ソース・データベースで REDO ログ・ファイル・グループを順次いっぱいにしてアーカイブするプロセスは、スタンバイ REDO ログ・グループを順次いっぱいにしてアーカイブすることで、各 REDO 転送先でミラーリングされます。

各スタンバイ REDO ログ・ファイルは、少なくとも REDO ソース・データベースの REDO ログで最も大きな REDO ログ・ファイルと同じ大きさである必要があります。管理しやすくするために、REDO ソース・データベースの REDO ログの全 REDO ログ・ファイルと REDO 転送先のスタンバイ REDO ログは同じサイズにすることをお薦めします。

スタンバイ REDO ログには、REDO ソース・データベースの REDO ログより REDO ログ・グループが少なくとも 1 つ多く必要です。

REDO ソース・データベースで次の問合せを実行し、REDO ログの各ログ・ファイルのサイズおよびログ・グループ数を確認します。

```
SQL> SELECT GROUP#, BYTES FROM V$LOG;
```

REDO 転送先データベースで次の問合せを実行し、スタンバイ REDO ログの各ログ・ファイルのサイズおよびログ・グループ数を確認します。

```
SQL> SELECT GROUP#, BYTES FROM V$STANDBY_LOG;
```

スタンバイ REDO ログは、Data Guard 構成のプライマリ・データベースに作成して、スタンバイ・ロールへのスイッチオーバー後すぐに REDO データを受信できるようにすることをお薦めします。

`ALTER DATABASE ADD STANDBY LOGFILE` SQL 文を使用して、スタンバイ REDO ログを作成し、スタンバイ REDO ログ・グループを既存のスタンバイ REDO ログに追加します。

たとえば、REDO ソース・データベースの REDO ログに 2 つの REDO ログ・グループがあり、各グループには 500MB の REDO ログ・ファイルが 1 つずつ含まれているとします。この場合、REDO ソース・データベースの REDO ログより REDO ログ・グループが少なくとも 1 つ多く必要であるという要件を満たすため、スタンバイ REDO ログには、3 つ以上のスタンバイ REDO ログ・グループが必要です。

次の SQL 文を使用すると、この使用例に適したスタンバイ REDO ログを作成できます。

```
ALTER DATABASE ADD STANDBY LOGFILE
   ('/oracle/dbs/slog1.rdo') SIZE 500M;

ALTER DATABASE ADD STANDBY LOGFILE
   ('/oracle/dbs/slog2.rdo') SIZE 500M;

ALTER DATABASE ADD STANDBY LOGFILE
   ('/oracle/dbs/slog3.rdo') SIZE 500M;
```

注意： REDO ログ・グループを Oracle Data Guard 構成のプライマリ・データベースに追加した場合は常に、同期 REDO 転送モードを使用する構成の各スタンバイ・データベースで、スタンバイ REDO ログ・グループをスタンバイ REDO ログに追加する必要もあります。そうしないと、最大保護データ保護モードで稼働しているプライマリ・データベースは停止し、最大可用性データ保護モードで稼働しているプライマリ・データベースは最大パフォーマンス・データ保護モードに移行します。

6.2.3.2 スタンバイ REDO ログ・アーカイブの構成

この項では、スタンバイ REDO ログ・アーカイブの構成方法について説明します。

関連項目：

- アーカイブ REDO ログの管理の詳細は、『Oracle Database 管理者ガイド』を参照してください。
- フラッシュ・リカバリ領域の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

6.2.3.2.1 フラッシュ・リカバリ領域へのスタンバイ REDO ログ・アーカイブ 次の手順を実行して、スタンバイ REDO ログ・アーカイブをフラッシュ・リカバリ領域に設定します。

1. LOG_ARCHIVE_DEST_*n* パラメータの LOCATION 属性を USE_DB_RECOVERY_FILE_DEST に設定します。
2. 同じ LOG_ARCHIVE_DEST_*n* パラメータの VALID_FOR 属性をスタンバイ REDO ログ・アーカイブを許可する値に設定します。

次に、スタンバイ REDO ログをフラッシュ・リカバリ領域にアーカイブするようにフィジカル・スタンバイ・データベースを構成するために使用されるパラメータ値のサンプルをいくつか示します。

```
LOG_ARCHIVE_DEST_2 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(STANDBY_LOGFILE,STANDBY_ROLE)'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

アーカイブ REDO ログ・ファイルの管理が簡素化されるため、フラッシュ・リカバリ領域の使用をお薦めします。

6.2.3.2.2 ローカル・ファイル・システムの場所へのスタンバイ REDO ログ・アーカイブ 次の手順を実行して、スタンバイ REDO ログ・アーカイブをローカル・ファイル・システムの場所に設定します。

1. LOG_ARCHIVE_DEST_*n* パラメータの LOCATION 属性を有効なパス名に設定します。
2. 同じ LOG_ARCHIVE_DEST_*n* パラメータの VALID_FOR 属性をスタンバイ REDO ログ・アーカイブを許可する値に設定します。

次に、スタンバイ REDO ログをローカル・ファイル・システムの場所にアーカイブするようにフィジカル・スタンバイ・データベースを構成するために使用されるパラメータ値のサンプルをいくつか示します。

```
LOG_ARCHIVE_DEST_2 = 'LOCATION = /disk2/archive
VALID_FOR=(STANDBY_LOGFILE,STANDBY_ROLE)'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

6.3 REDO 転送サービスの監視

この項は、次の項目で構成されています。

6.3.1 REDO 転送ステータスの監視

この項では、REDO ソース・データベースで REDO 転送ステータスを監視するのに使用する手順について説明します。

手順 1 最新のアーカイブ REDO ログ・ファイルを判定する

REDO ソース・データベースで次の問合せを実行し、各スレッドの最新のアーカイブ順序番号を判定します。

```
SQL> SELECT MAX(SEQUENCE#), THREAD# FROM V$ARCHIVED_LOG GROUP BY THREAD#;
```

手順 2 各 REDO 転送先で最新のアーカイブ REDO ログ・ファイルを判定する

REDO ソース・データベースで次の問合せを実行し、各 REDO 転送先の最新のアーカイブ REDO ログ・ファイルを判定します。

```
SQL> SELECT DESTINATION, STATUS, ARCHIVED_THREAD#, ARCHIVED_SEQ#
  2> FROM V$ARCHIVE_DEST_STATUS
  3> WHERE STATUS <> 'DEFERRED' AND STATUS <> 'INACTIVE';

DESTINATION         STATUS  ARCHIVED_THREAD#  ARCHIVED_SEQ#
------------------  ------  ----------------  -------------
/private1/prmy/lad   VALID                 1            947
standby1             VALID                 1            947
```

最新のアーカイブ REDO ログ・ファイルは、各宛先で同じである必要があります。同じでない場合、その宛先へのアーカイブ操作の間に検出されたエラーは VALID 以外のステータスで表示されます。

手順 3 アーカイブ REDO ログ・ファイルが REDO 転送先で受信されたかどうかを確認する

REDO ソース・データベースで問合せを実行すると、アーカイブ REDO ログ・ファイルが特定の REDO 転送先で受信されたかどうかを確認できます。各宛先には対応付けられた ID 番号があります。データベースの V$ARCHIVE_DEST ビューの DEST_ID 列に問合せを発行して、各宛先の ID 番号を特定できます。

宛先 1 がローカル・アーカイブ REDO ログを指し、宛先 2 が REDO 転送先を指しているとします。REDO ソース・データベースで次の問合せを実行し、ログ・ファイルが REDO 転送先で欠落しているかどうかを確認します。

```
SQL> SELECT LOCAL.THREAD#, LOCAL.SEQUENCE# FROM
  2> (SELECT THREAD#, SEQUENCE# FROM V$ARCHIVED_LOG WHERE DEST_ID=1)
  3> LOCAL WHERE
  4> LOCAL.SEQUENCE# NOT IN
  5> (SELECT SEQUENCE# FROM V$ARCHIVED_LOG WHERE DEST_ID=2 AND
  6> THREAD# = LOCAL.THREAD#);

  THREAD#  SEQUENCE#
---------  ---------
  1        12
  1        13
  1        14
```

手順 4 REDO 転送先に転送された REDO の進捗状況をトレースする

REDO ソース・データベースおよび各 REDO 転送先で LOG_ARCHIVE_TRACE データベース初期化パラメータを設定し、REDO 転送の進捗状況をトレースします。詳細と例は、付録 G を参照してください。

6.3.2 同期 REDO 転送のレスポンス時間の監視

V$REDO_DEST_RESP_HISTOGRAM ビューには、REDO 転送先ごとのレスポンス時間データが格納されます。このレスポンス時間データは、同期 REDO 転送モードにより送信された REDO 転送メッセージに関して保持されます。

宛先ごとのデータは一連の行で構成されており、各レスポンス時間に 1 行ずつ割り当てられています。レコードの保存を簡素化するために、300 秒未満の場合、レスポンス時間は最も近い整数の秒数に切り上げられます。300 秒超のレスポンス時間は、600、1200、2400、4800 または 9600 秒にそれぞれ切り上げられます。

各行は、4 つの列 FREQUENCY、DURATION、DEST_ID および TIME で構成されます。

FREQUENCY 列には、特定のレスポンス時間が監視された回数が格納されます。DURATION 列は、レスポンス時間に相当します。DEST_ID 列は、宛先を特定します。TIME 列には、その行が最後に更新されたときに取得されたタイムスタンプが格納されます。

このビューのレスポンス時間データは、REDO ソース・データベースでのトランザクション・スループットに影響を与える可能性がある同期 REDO 転送モードのパフォーマンスの問題を特定するのに役立ちます。また、NET_TIMEOUT 属性のチューニングにも有効です。

次の 3 つの例では、LOG_ARCHIVE_DEST_2 パラメータに対応する宛先 2 への問合せのサンプルを示します。別の宛先についてレスポンス時間データを表示するには、問合せの DEST_ID を変更するだけです。

REDO ソース・データベースで次の問合せを実行すると、宛先 2 についてレスポンス時間のヒストグラムを表示できます。

```
SQL> SELECT FREQUENCY, DURATION FROM
2> V$REDO_DEST_RESP_HISTOGRAM WHERE DEST_ID=2 AND FREQUENCY>1;
```

REDO ソース・データベースで次の問合せを実行すると、宛先 2 について最も遅いレスポンス時間を表示できます。

```
SQL> SELECT max(DURATION) FROM V$REDO_DEST_RESP_HISTOGRAM
2> WHERE DEST_ID=2 AND FREQUENCY>1;
```

REDO ソース・データベースで次の問合せを実行すると、宛先 2 について最も速いレスポンス時間を表示できます。

```
SQL> SELECT min( DURATION) FROM V$REDO_DEST_RESP_HISTOGRAM
2> WHERE DEST_ID=2 AND FREQUENCY>1;
```

> **注意：** 宛先について最も頻繁に監視されるレスポンス時間は、その宛先に指定された NET_TIMEOUT 値の最高値を超えることはできません。これは、REDO 転送先が NET_TIMEOUT の秒数内に REDO 転送メッセージに応答しない場合、同期 REDO 転送モードのセッションが終了するためです。

6.3.3 REDO ギャップの検出および解決

REDO ギャップは、REDO 転送が割り込まれるたびに発生します。REDO 転送が再開されると、REDO 転送サービスは REDO ギャップを自動的に検出し、欠落した REDO を宛先に送信して解決します。

REDO ギャップの解決に必要な時間は、ギャップのサイズに正比例し、REDO ソース・データベースと REDO 転送先との間のネットワーク・リンクの実効スループットに反比例します。REDO 転送サービスには、パフォーマンスが低いネットワーク・リンクを使用している場合に REDO ギャップの解決時間を短縮するオプションが 2 つあります。

- REDO 転送の圧縮

  LOG_ARCHIVE_DEST_*n* パラメータの COMPRESSION 属性を使用すると、REDO 転送の圧縮を使用して、REDO ギャップを解決するために送信される REDO を圧縮するように指定できます。

- パラレル REDO 転送ネットワーク・セッション

  LOG_ARCHIVE_DEST_*n* パラメータの MAX_CONNECTIONS 属性を使用すると、複数のネットワーク・セッションを使用して、REDO ギャップを解決するために必要な REDO を送信するように指定できます。

COMPRESSION 属性と MAX_CONNECTIONS 属性の詳細は、第 15 章「LOG_ARCHIVE_DEST_n パラメータの属性」を参照してください。

6.3.3.1 手動によるギャップの解決

場合によっては、ギャップの解決を自動的に実行できず、手動による実行が必要になります。たとえば、プライマリ・データベースが使用できない場合には、ロジカル・スタンバイ・データベースで REDO ギャップの解決を手動で実行する必要があります。

フィジカル・スタンバイ・データベースで次の問合せを実行し、REDO ギャップがフィジカル・スタンバイ・データベースに存在するかどうかを判断します。

```
SQL> SELECT * FROM V$ARCHIVE_GAP;
    THREAD# LOW_SEQUENCE# HIGH_SEQUENCE#
----------  -------------  --------------
          1              7              10
```

この例の出力は、フィジカル・スタンバイ・データベースでスレッド 1 の順序番号 7 ～ 10 のログ・ファイルが現在欠落していることを示しています。

プライマリ・データベースで次の問合せを実行し、プライマリ・データベースでのアーカイブ REDO ログ・ファイルの位置を特定します（プライマリ・データベースのローカル・アーカイブ先は LOG_ARCHIVE_DEST_1 とします）。

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE THREAD#=1 AND
2> DEST_ID=1 AND SEQUENCE# BETWEEN 7 AND 10;
NAME
--------------------------------------------------------------------------------
/primary/thread1_dest/arcr_1_7.arc
/primary/thread1_dest/arcr_1_8.arc
/primary/thread1_dest/arcr_1_9.arc
```

注意： この問合せは、特定のスレッドについて連続する順序番号を戻します。その場合、実際のギャップはありませんが、関連するスレッドは、これらの 2 つのアーカイブ・ログの生成期間内に無効および有効に設定されました。また、この問合せは、特定のスレッドの最後に存在するギャップを識別しません。たとえば、プライマリ・データベースでスレッド 1 に対して順序番号 100 までアーカイブ・ログが生成され、ロジカル・スタンバイ・データベースがそのスレッドについて受信した最後のアーカイブ・ログが順序番号 77 である場合、この問合せは、順序番号 78 ～ 100 のアーカイブ・ログについてギャップがあるにもかかわらず、1 行も戻しません。

これらのログ・ファイルをフィジカル・スタンバイ・データベースにコピーし、コピーしたログを ALTER DATABASE REGISTER LOGFILE を使用して登録します。次に例を示します。

```
SQL> ALTER DATABASE REGISTER LOGFILE
'/physical_standby1/thread1_dest/arcr_1_7.arc';
SQL> ALTER DATABASE REGISTER LOGFILE
'/physical_standby1/thread1_dest/arcr_1_8.arc';
SQL> ALTER DATABASE REGISTER LOGFILE
'/physical_standby1/thread1_dest/arcr_1_9.arc';
```

注意： フィジカル・スタンバイ・データベースの V$ARCHIVE_GAP ビューが戻すのは、REDO Apply の続行を現在ブロックしているギャップのみです。そのギャップを解決した後、フィジカル・スタンバイ・データベースで V$ARCHIVE_GAP ビューを再度問い合せ、別のギャップ・シーケンスが存在するかどうかを判断します。この処理をギャップがなくなるまで繰り返します。

REDO ギャップがロジカル・スタンバイ・データベースに存在するかどうかを判断するには、ロジカル・スタンバイ・データベースで DBA_LOGSTDBY_LOG ビューを問い合せます。たとえば、次の問合せでは、ロジカル・スタンバイ・データベースの THREAD 1 に、2 つのファイルが表示されているため、アーカイブ REDO ログ・ファイルの順序番号にギャップがあることを示しています。（ギャップがない場合、問合せで表示されるのは、スレッドごとに 1 つのファイルのみです。）この出力は、登録されたファイルの最大の順序番号は 10 ですが、順序番号 6 で示されるファイルにギャップがあることを示しています。

```
SQL> COLUMN FILE_NAME FORMAT a55
SQL> SELECT THREAD#, SEQUENCE#, FILE_NAME FROM
DBA_LOGSTDBY_LOG L
  2> WHERE NEXT_CHANGE# NOT IN
  3> (SELECT FIRST_CHANGE# FROM DBA_LOGSTDBY_LOG WHERE L.THREAD# = THREAD#)
  4> ORDER BY THREAD#, SEQUENCE#;

   THREAD#  SEQUENCE# FILE_NAME
---------- ---------- -----------------------------------------------
         1          6 /disk1/oracle/dbs/log-1292880008_6.arc
         1         10 /disk1/oracle/dbs/log-1292880008_10.arc
```

順序番号 7、8 および 9 の欠落しているログ・ファイルをロジカル・スタンバイ・システムにコピーし、コピーしたログを ALTER DATABASE REGISTER LOGICAL LOGFILE 文を使用して登録します。次に例を示します。

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/disk1/oracle/dbs/log-1292880008_7.arc';
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/disk1/oracle/dbs/log-1292880008_8.arc';
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/disk1/oracle/dbs/log-1292880008_9.arc';
```

注意： ここで指定した、ロジカル・スタンバイ・データベースの DBA_LOGSTDBY_LOG ビューに基づいた問合せが戻すのは、SQL Apply の続行を現在ブロックしているギャップのみです。そのギャップを解決した後、ロジカル・スタンバイ・データベースで DBA_LOGSTDBY_LOG ビューを再度問い合せ、別のギャップ・シーケンスが存在するかどうかを判断します。この処理をギャップがなくなるまで繰り返します。

6.3.4 REDO 転送サービスの待機イベント

表 6-2 に、REDO ソース・データベースでの REDO 転送の待機時間を追跡管理するのに使用される Oracle 待機イベントをいくつか示します。これらの待機イベントは、V$SYSTEM_EVENT 動的パフォーマンス・ビューに表示されます。

REDO 転送で使用される Oracle 待機イベントの詳細リストは、次の URL から Oracle Maximum Availability Architecture（MAA）のホームページにアクセスし、『Oracle Data Guard Redo Transport and Network Best Practices』ホワイト・ペーパーを参照してください。

http://otn.oracle.com/deploy/availability/htdocs/maa.htm

表 6-2 REDO 転送の待機イベント

待機イベント	説明
LNS wait on ATTACH	すべての ASYNC および SYNC の REDO 転送先に対して REDO 転送セッションが確立するまでの待機に費やされた合計時間
LNS wait on SENDREQ	すべての ASYNC および SYNC の REDO 転送先に REDO データが書き込まれるまでの待機に費やされた合計時間
LNS wait on DETACH	すべての ASYNC および SYNC の REDO 転送先に対して REDO 転送の接続が終了されるまでの待機に費やされた合計時間

6.4 REDO 転送のチューニング

最高のパフォーマンスを得るために REDO 転送を最適化する方法は、『Oracle Data Guard Redo Transport and Network Configuration Best Practices』ホワイト・ペーパーを参照してください。このホワイト・ペーパーは、次の URL から Oracle Maximum Availability Architecture（MAA）のホームページにアクセスして入手することができます。

http://otn.oracle.com/deploy/availability/htdocs/maa.htm

7

適用サービス

この章では、REDO データをスタンバイ・データベースに適用する方法について説明します。この章は、次の項目で構成されています。

7.1 適用サービスの概要

適用サービスは、スタンバイ・データベースに REDO を自動的に適用して、プライマリ・データベースとの同期を維持します。また、データに対するトランザクションの一貫性のあるアクセスを可能にします。

デフォルトでは、適用サービスはいっぱいになったアーカイブ REDO ログ・ファイルがスタンバイ・データベースで受信されるのを待機してから、スタンバイ・データベースに適用します。ただし、スタンバイ REDO ログを使用している場合は、**リアルタイム適用**を使用可能にできます。リアルタイム適用により、Data Guard では、現在のスタンバイ REDO ログ・ファイルがいっぱいになったときに、現在のスタンバイ REDO ログ・ファイルから REDO データをリカバリできます。リアルタイム適用の詳細は、7.2.1 項を参照してください。

適用サービスは、次の方法でフィジカル・スタンバイ・データベースとロジカル・スタンバイ・データベースをメンテナンスします。

- REDO Apply（フィジカル・スタンバイ・データベースのみ）

  メディア・リカバリを使用して、プライマリ・データベースとフィジカル・スタンバイ・データベースの同期を維持します。

- SQL Apply（ロジカル・スタンバイ・データベースのみ）

  プライマリ・データベースから受信した REDO に基づいて SQL 文を再構成し、その SQL 文をロジカル・スタンバイ・データベースに対して実行します。

  ロジカル・スタンバイ・データベースは読取り / 書込みモードでオープンできますが、レポート生成のためにロジカル・スタンバイ・データベースでメンテナンスされているターゲット表は読取り専用モードでオープンします（データベース・ガードが適切に設定されている場合）。SQL Apply を使用すると、SQL 文が適用されている間でも、ロジカル・スタンバイ・データベースをレポート・アクティビティで使用できます。

この章の各項では、REDO Apply、SQL Apply、リアルタイム適用および適用の遅延について詳細に説明します。

7.2 適用サービスの構成オプション

この項は、次の項目で構成されています。

- リアルタイム適用による REDO データの即時適用
- アーカイブ REDO ログ・ファイルの適用に対する遅延時間の指定

7.2.1 リアルタイム適用による REDO データの即時適用

リアルタイム適用機能が使用可能になっている場合、適用サービスでは、現在のスタンバイ REDO ログ・ファイルがアーカイブされるのを待機せずに REDO データを受信したときに、REDO データを適用できます。これにより、スイッチオーバーおよびフェイルオーバーが高速化されます。これは、フェイルオーバーまたはスイッチオーバーが開始されるまでに、スタンバイ REDO ログ・ファイルがスタンバイ・データベースにすでに適用されているためです。

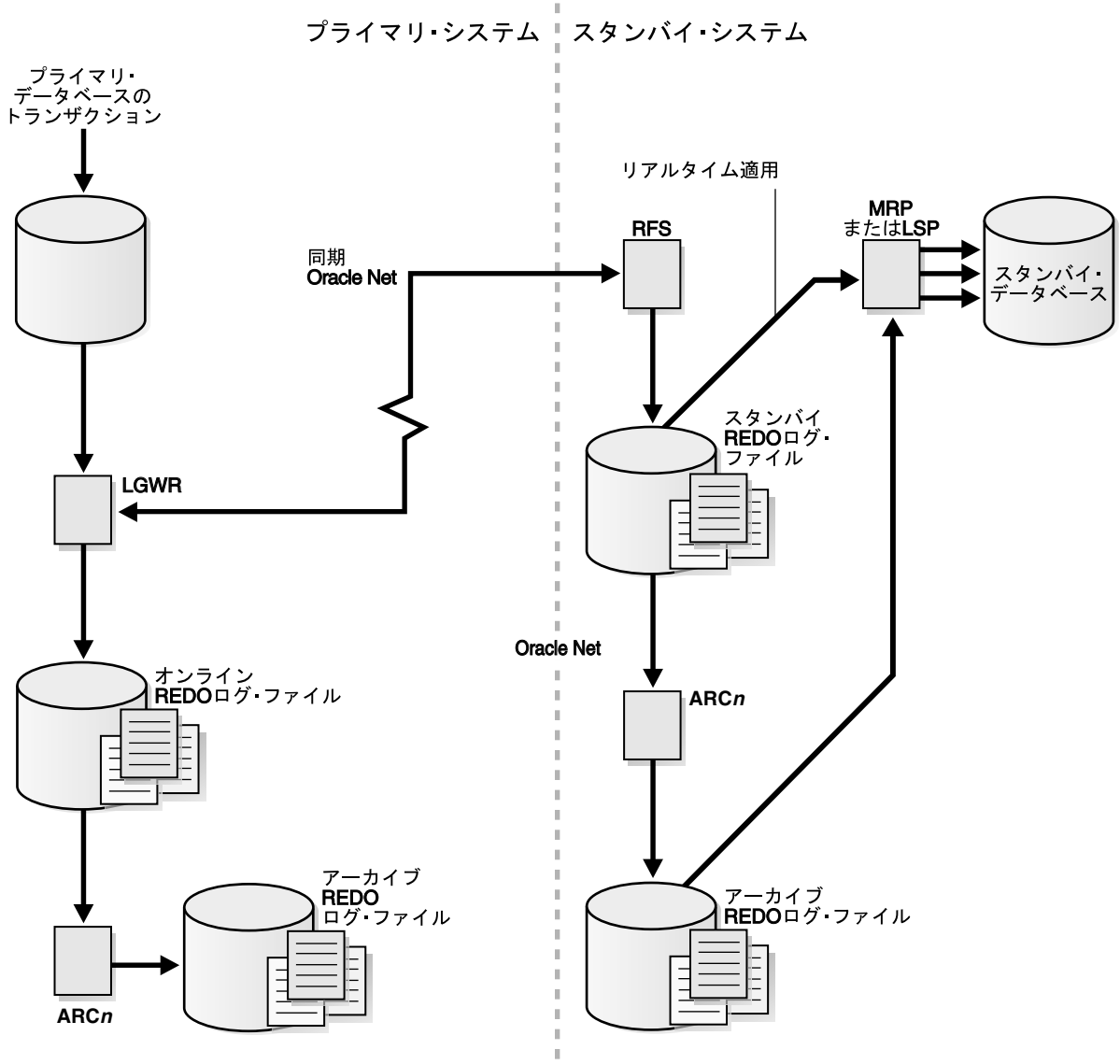
次のように、`ALTER DATABASE` 文を使用して、リアルタイム適用機能を使用可能にします。

- フィジカル・スタンバイ・データベースの場合は、`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE` 文を発行します。
- ロジカル・スタンバイ・データベースの場合は、`ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE` 文を発行します。

リアルタイム適用を使用するには、スタンバイ REDO ログ・ファイルが必要です。

図 7-1 に、ローカル宛先とスタンバイ宛先がある Data Guard 構成を示します。リモート・ファイル・サーバー（RFS）プロセスは、スタンバイ・データベース上のスタンバイ REDO ログ・ファイルに REDO データを書き込むため、適用サービスでは、いっぱいになったときにスタンバイ REDO ログ・ファイルから REDO をリカバリできます。

図 7-1 リアルタイム適用を使用したスタンバイ宛先への REDO データの適用



7.2.2 アーカイブ REDO ログ・ファイルの適用に対する遅延時間の指定

プライマリ・サイトから REDO データを受信した後、その REDO データをスタンバイ・データベースに適用するまでの間にタイム・ラグを作成することが必要な場合があります。時間間隔（分単位）を指定すると、破損したデータや誤ったデータがスタンバイ・サイトに適用されるのを防止できます。DELAY 間隔を設定すると、スタンバイ・データベースへの REDO データの転送が遅延することはありません。かわりに、指定したタイム・ラグは、REDO データがスタンバイ宛先で完全にアーカイブされたときに開始します。

注意： リアルタイム適用が使用可能になっている宛先に遅延を定義すると、その遅延は無視されます。

遅延時間の指定

LOG_ARCHIVE_DEST_*n* 初期化パラメータの DELAY=*minutes* 属性を使用してプライマリ・データベースとスタンバイ・データベースで遅延時間を設定し、スタンバイ・データベースへのアーカイブ REDO ログ・ファイルの適用を遅延させることができます。デフォルトでは、遅延時間は発生しません。値を指定せずに DELAY 属性を指定すると、デフォルトの遅延間隔は 30 分になります。

遅延時間の取消し

次のように、指定した遅延間隔を取り消すことができます。

- フィジカル・スタンバイ・データベースで、RECOVER MANAGED STANDBY DATABASE 句の NODELAY キーワードを使用します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NODELAY;
```

- ロジカル・スタンバイ・データベースで、次の SQL コマンドを指定します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NODELAY;
```

これらのコマンドにより、時間間隔が経過する前に、適用サービスで、スタンバイ・データベースへのアーカイブ REDO ログ・ファイルの適用が即時に開始されます。

7.2.2.1 遅延時間設定の代替策としてのフラッシュバック・データベースの使用

適用遅延を設定するかわりにフラッシュバック・データベースを使用して、スタンバイ・データベースへの破損データやエラー・データの適用からリカバリできます。フラッシュバック・データベースを使用すると、スタンバイ・データベースを任意の時点まで、すばやく容易にフラッシュ・バックできます。

Data Guard とフラッシュバック・データベースの併用方法の例は第 13 章を、フラッシュバック・データベースを有効にして使用する方法の詳細は『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

7.3 REDO データのフィジカル・スタンバイ・データベースへの適用

デフォルトでは、アーカイブ REDO ログ・ファイルからの REDO データが適用されます。REDO Apply を実行している場合、フィジカル・スタンバイ・データベースはリアルタイム機能を使用して、スタンバイ REDO ログ・ファイルが RFS プロセスによって書き込まれているときに、スタンバイ REDO ログ・ファイルから直接 REDO を適用できます。適用サービスは、フィジカル・スタンバイ・データベースが読取り専用でオープンしているときは、REDO データをデータベースに適用できないことに注意してください。

この項は、次の項目で構成されています。

7.3.1 REDO Apply の開始

フィジカル・スタンバイ・データベースで適用サービスを開始するには、フィジカル・スタンバイ・データベースが起動されマウントされていることを確認してから、SQL `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE` 文を使用して REDO Apply を開始します。

Redo Apply をフォアグラウンド・セッションまたはバックグラウンド・プロセスとして動作するように指定し、リアルタイム適用で使用可能にすることができます。

- REDO Apply をフォアグラウンドで開始するには、次の SQL 文を発行します。

  ```
  SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
  ```

  フォアグラウンド・セッションを開始すると、別のセッションでリカバリが取り消されるまで、制御がコマンド・プロンプトに戻りません。

- REDO Apply をバックグラウンドで開始するには、この SQL 文に `DISCONNECT` キーワードを挿入します。次に例を示します。

  ```
  SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;
  ```

  この文によって、分離されたサーバー・プロセスが起動し、即時に制御がユーザーに戻されます。管理リカバリ・プロセスがバックグラウンドでリカバリを実行している間、`RECOVER` 文を発行したフォアグラウンド・プロセスは、他のタスクの実行を継続できます。これによって、カレント SQL セッションが切断されることはありません。

- リアルタイム適用を開始するには、SQL 文に `USING CURRENT LOGFILE` 句を含めます。次に例を示します。

  ```
  SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE;
  ```

7.3.2 REDO Apply の停止

REDO Apply を停止するには、別のウィンドウで次の SQL 文を発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

7.3.3 フィジカル・スタンバイ・データベースでの REDO Apply の監視

フィジカル・スタンバイ・データベースでの適用サービスのステータスを監視するには、9.5.1 項を参照してください。スタンバイ・データベースは、Oracle Enterprise Manager を使用して監視することもできます。また、ビューの詳細は『Oracle Database リファレンス』を参照してください。

7.4 REDO データのロジカル・スタンバイ・データベースへの適用

SQL Apply は、アーカイブ REDO ログまたはスタンバイ REDO ログのデータを SQL 文に変換してから、その SQL 文をロジカル・スタンバイ・データベースで実行します。ロジカル・スタンバイ・データベースはオープン状態のままであるため、メンテナンスされる表は、レポート生成、要約、問合せなどの他のタスクで同時に使用できます。

この項は、次の項目で構成されています。

- SQL Apply の開始
- ロジカル・スタンバイ・データベースでの SQL Apply の停止
- ロジカル・スタンバイ・データベースでの SQL Apply の監視

7.4.1 SQL Apply の開始

SQL Apply を開始するには、ロジカル・スタンバイ・データベースを起動して次の文を発行します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

ロジカル・スタンバイ・データベースでリアルタイム適用を開始して、ロジカル・スタンバイ・データベースのスタンバイ REDO ログ・ファイルから REDO データを即時に適用するには、次の文に示すように、IMMEDIATE キーワードを含めます。

```
SQL>  ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

7.4.2 ロジカル・スタンバイ・データベースでの SQL Apply の停止

SQL Apply を停止するには、ロジカル・スタンバイ・データベースで次の文を発行します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
```

この文を発行すると、SQL Apply は適用対象プロセス内の完了トランザクションがすべてコミットされるまで待機します。つまり、このコマンドを実行しても、SQL Apply プロセスが即時に停止しない場合があります。

7.4.3 ロジカル・スタンバイ・データベースでの SQL Apply の監視

SQL Apply を監視するには、10.3 項を参照してください。スタンバイ・データベースは、Oracle Enterprise Manager を使用して監視することもできます。付録 A「Data Guard のトラブルシューティング」および『Oracle Data Guard Broker』を参照してください。

8

ロールの推移

Data Guard 構成は、プライマリ・ロールで機能する 1 つのデータベース、およびスタンバイ・ロールで機能する 1 つ以上のデータベースで構成されています。通常、各データベースのロールは変更されません。ただし、プライマリ・データベースが停止したときに Data Guard を使用してサービスを維持する場合は、構成内の現行のプライマリ・データベースと 1 つのスタンバイ・データベース間でロールの推移を開始する必要があります。データベースの現行のロールを確認するには、V$DATABASE ビューの DATABASE_ROLE 列を問い合せます。

Data Guard 構成内のスタンバイ・データベースの数、位置、タイプおよびプライマリ・データベースからの REDO データを各スタンバイ・データベースに伝播する方法に応じて、プライマリ・データベースの停止に対して設定できるロール管理オプションが決定します。

この章では、Data Guard 構成でロールの推移を管理する方法について説明します。この章は、次の項目で構成されています。

- ロールの推移の概要
- フィジカル・スタンバイ・データベースが関与するロールの推移
- ロジカル・スタンバイ・データベースが関与するロールの推移
- ロールの推移後のフラッシュバック・データベースの使用

この章で説明するロールの推移は、SQL 文を使用して手動で起動します。Oracle Data Guard Broker を使用し、ロールの推移を単純化してフェイルオーバーを自動化することもできます。

> **関連項目：** Oracle Data Guard Broker を次の目的で使用する方法の詳細は、『Oracle Data Guard Broker』を参照してください。
>
> - Oracle Enterprise Manager でキー・クリックを 1 回する、あるいは DGMGRL コマンドライン・インタフェースで単一のコマンドを使用すると起動できるようにする、スイッチオーバーおよびフェイルオーバーの単純化。
> - プライマリ・データベースが使用不可能になった場合に、**ファスト・スタート・フェイルオーバー**による自動的なフェイルオーバーを可能にします。ファスト・スタート・フェイルオーバーが有効になると、Data Guard Broker によりフェイルオーバーが必要かどうか決定され、指定のターゲット・スタンバイ・データベースに対して自動的にフェイルオーバーが開始されます。DBA による操作の必要はありません。

8.1 ロールの推移の概要

データベースは、相互に排他的な 2 つのロールのいずれかで実行されます。すなわち、**プライマリ**と**スタンバイ**です。Data Guard では、この章で説明する SQL 文を発行するか、Data Guard Broker の GUI またはコマンドライン・インタフェースを使用して、これらのロールを動的に変更できます。Oracle Data Guard は、次のロールの推移をサポートしています。

- **スイッチオーバー**

  プライマリ・データベースが、スタンバイ・データベースの 1 つを使用してロールを切り替えられるようにします。スイッチオーバー時には、データは消失しません。スイッチオーバーの完了後、各データベースは、新しいロールとともに Data Guard 構成に継続して含まれます。

- **フェイルオーバー**

  プライマリ・データベースの障害時に、スタンバイ・データベースをプライマリ・ロールに変更します。障害の前にプライマリ・データベースが最大保護モードまたは最大可用性モードで動作していなかった場合、データ消失が発生することがあります。フラッシュバック・データベースがプライマリ・データベースで使用可能になっている場合は、障害の原因が修正された後、元どおり新しいプライマリ・データベースのスタンバイに戻すことができます。

8-2 ページの 8.1.1 項「ロールの推移の準備」では、停止時間およびデータ消失のリスクを最小限にするために、どのロールの推移を選択するかについて説明します。8-4 ページの 8.1.3 項「スイッチオーバー」および 8-6 ページの 8.1.4 項「フェイルオーバー」では、それぞれスイッチオーバーとフェイルオーバーの詳細を説明します。

8.1.1 ロールの推移の準備

ロールの推移を開始する前に、次の準備を実行します。

- 各データベースが、担う予定のロールについて正しく構成されていることを確認します。プライマリ・データベースとスタンバイ・データベースでデータベース初期化パラメータ、アーカイブ・ログ・モード、スタンバイ REDO ログ、オンライン REDO ログを構成する方法の詳細は、第 3 章「フィジカル・スタンバイ・データベースの作成」および第 4 章「ロジカル・スタンバイ・データベースの作成」を参照してください。

  注意： スイッチオーバーまたはフェイルオーバーの発生時に、すべてのスタンバイ・サイトが新しいプライマリ・データベースから引き続き REDO データを受信するように、各スタンバイ・データベース上で `LOG_ARCHIVE_DEST_n` および `LOG_ARCHIVE_DEST_STATE_n` パラメータを定義する必要があります。

- スタンバイ・データベースに存在する一時ファイルが、プライマリ・データベースの一時ファイルと一致することを確認します。
- 新たにプライマリ・データベースになるスタンバイ・データベースで現在有効になっている REDO の適用遅延を解除します。
- Oracle RAC プライマリ・データベースからフィジカル・スタンバイ・データベースにスイッチオーバーを実行するには、1 つのプライマリ・データベース・インスタンス以外はすべて停止します。このとき停止したプライマリ・データベース・インスタンスは、スイッチオーバーの完了後に起動できます。

  Oracle RAC フィジカル・スタンバイ・データベースへのスイッチオーバーまたはフェイルオーバーを実行するには、1 つのスタンバイ・データベース・インスタンス以外はすべて停止します。このとき停止したスタンバイ・データベース・インスタンスは、ロールの推移の完了後に再起動できます。

8.1.2 ロールの推移のターゲット・スタンバイ・データベースの選択

複数のスタンバイ・データベースがある Data Guard 構成の場合、ロールの推移のターゲット・スタンバイ・データベースを選択する際には、多数の要因を考慮する必要があります。次のような要因があります。

- スタンバイ・データベースのローカリティ。
- スタンバイ・データベースの機能（CPU 数、使用可能な I/O 帯域幅などのハードウェア仕様）。
- ロールの推移の実行所要時間。これは、スタンバイ・データベースが REDO データの適用に関してどの程度遅れているか、およびアプリケーションの可用性とデータ消失をどの程度柔軟にトレードオフできるかに影響を受けます。
- スタンバイ・データベースのタイプ。

ロールの推移のターゲットとして選択されたスタンバイのタイプにより、構成内の他のスタンバイ・データベースがロールの推移後にどのように動作するか決定されます。ロールの推移前、新しいプライマリがフィジカル・スタンバイだった場合、構成内の他のスタンバイ・データベースは新しいプライマリのスタンバイになります。ロールの推移前、新しいプライマリがロジカル・スタンバイだった場合、構成内の他のすべてのロジカル・スタンバイは新しいプライマリのスタンバイになりますが、構成内のフィジカル・スタンバイは、古いプライマリのスタンバイのままであるため、新しいプライマリを保護しません。後者の場合、次回のスイッチオーバーまたはフェイルオーバーで元のプライマリ・データベースに戻ると、すべてのスタンバイは現行のプライマリのスタンバイとしての元のロールに戻ります。これらの理由により、フィジカル・スタンバイとロジカル・スタンバイの両方が含まれる構成内におけるロールの推移の最適なターゲットは、通常、フィジカル・スタンバイです。

注意： スナップショット・スタンバイは、ロールの推移のターゲットにはできません。

Data Guard には V$DATAGUARD_STATS ビューが用意されています。このビューを使用して、スタンバイ・データベースのデータの最新性から見た各スタンバイ・データベースと、使用可能な REDO データがすべてスタンバイ・データベースに適用された場合の、ロールの推移の実行所要時間を評価することができます。次に例を示します。

```
SQL> COLUMN NAME FORMAT A18
SQL> COLUMN VALUE FORMAT A16
SQL> COLUMN TIME_COMPUTED FORMAT A24
SQL> SELECT * FROM V$DATAGUARD_STATS;
NAME               VALUE             TIME_COMPUTED
------------------ ----------------  ------------------------
apply finish time  +00 00:00:02.4    15-MAY-2005 10:32:49
       second(1)
       interval
apply lag          +00 0:00:04       15-MAY-2005 10:32:49
       second(0)
       interval
transport lag      +00 00:00:00      15-MAY-2005 10:32:49
       second(0)
       interval
```

それぞれの統計の計算時刻は、TIME_COMPUTED 列に示されています。
V$DATATGUARD_STATS.TIME_COMPUTED 列は、V$DATATGUARD_STATS 行のメトリックが計算されるときに取得されるタイムスタンプです。この列は、関連付けられたメトリックの鮮度を示します。この例は、このスタンバイ・データベースではトランスポート・ラグがないこと、適用サービスでは過去 4 秒間に生成された REDO が適用されていないこと（apply lag）、および適用サービスで未適用の REDO の適用が完了するまでに 2.4 秒かかること（apply finish time）を示しています。APPLY LAG および TRANSPORT LAG メトリックは、プライマリ・データベースから受信される情報に基づいて計算されます。これらのメトリックは、プライマリ・データベースとスタンバイ・データベース間の通信が中断されると失効します。

APPLY LAG および TRANSPORT LAG メトリックについてこの列の値が変化しない場合、これらのメトリックが更新されていない（失効した）ことを示し、プライマリ・データベースとスタンバイ・データベース間の通信障害がその原因と考えられます。

8.1.3 スイッチオーバー

スイッチオーバーは、通常、オペレーティング・システムやハードウェアのアップグレード、または Oracle データベース・ソフトウェアとパッチ・セットのローリング・アップグレードなど、計画的な停止によるプライマリ・データベースの停止時間を短縮するために使用します（第 12 章「SQL Apply を使用した Oracle Database のアップグレード」を参照）。

スイッチオーバーは、2 つのフェーズで実行されます。最初のフェーズで、既存のプライマリ・データベースがスタンバイ・ロールに推移します。2 番目のフェーズで、スタンバイ・データベースがプライマリ・ロールに推移します。

図 8-1 は、データベースのロールを切り替える前の 2 つのサイトにある Data Guard 構成を示します。この構成では、プライマリ・データベースはサンフランシスコにあり、スタンバイ・データベースはボストンにあります。

図 8-1 スイッチオーバー前の Data Guard 構成

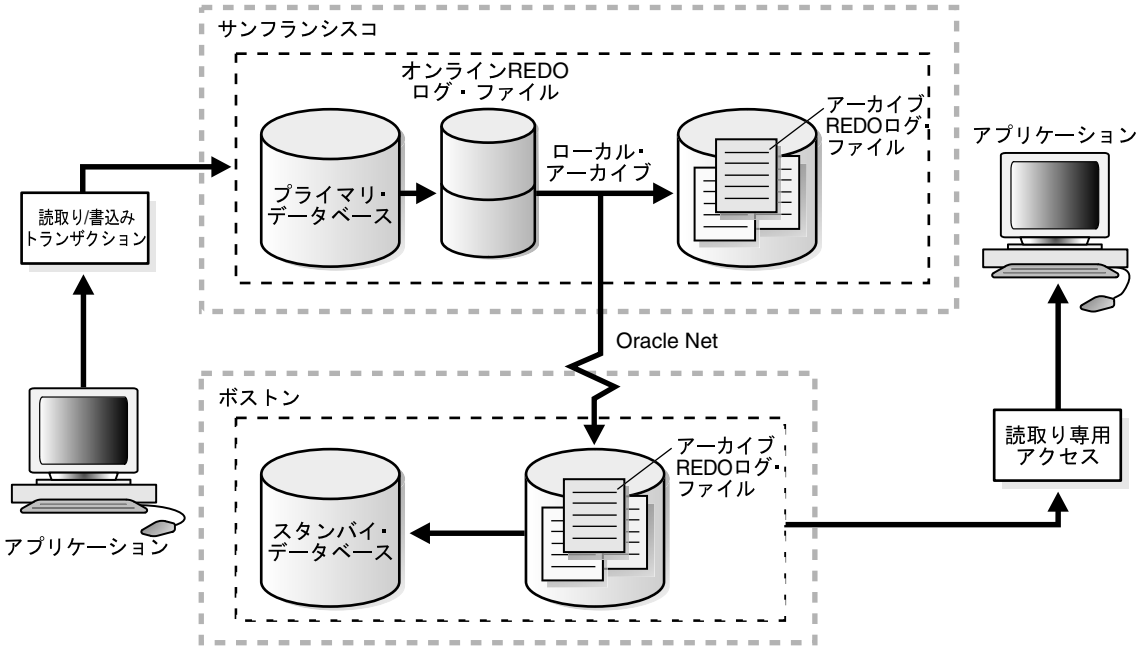


図 8-2 に、元のプライマリ・データベースがスタンバイ・データベースにスイッチオーバーされた後、元のスタンバイ・データベースがまだ新しいプライマリ・データベースになっていない Data Guard 環境を示します。このとき、Data Guard 構成には一時的に 2 つのスタンバイ・データベースが存在することになります。

図 8-2 新しいプライマリ・データベースへのスイッチオーバー前のスタンバイ・データベース

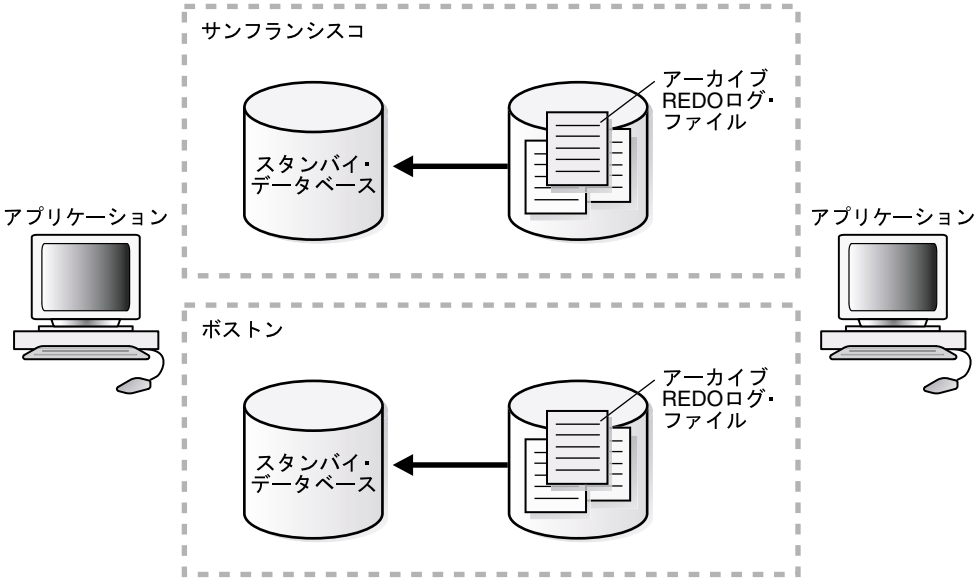
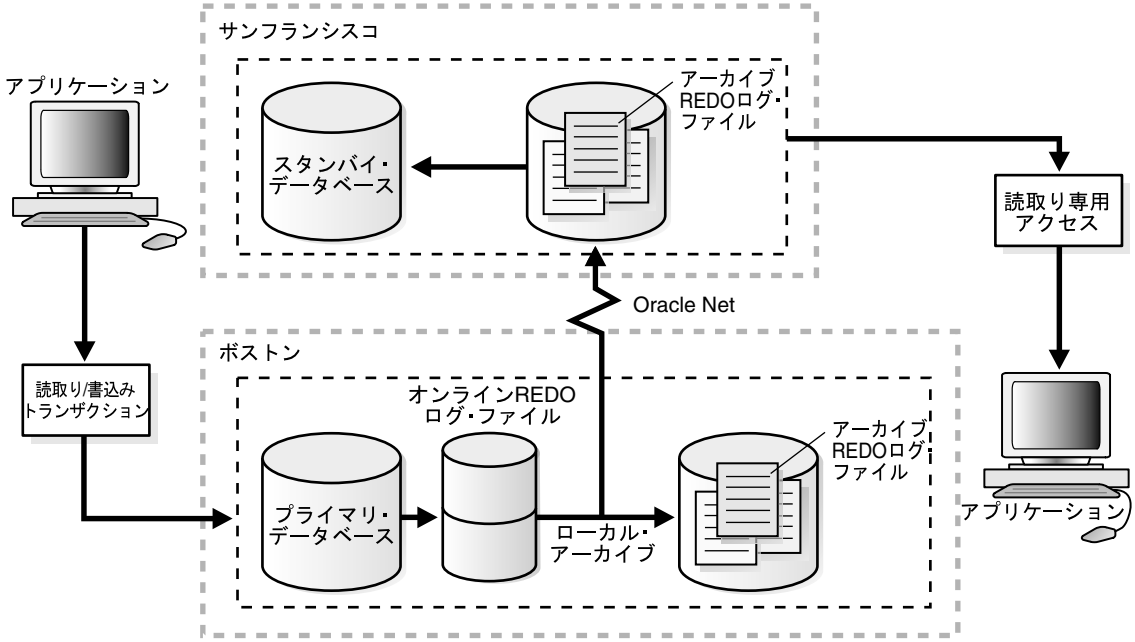


図 8-3 は、スイッチオーバー実行後の Data Guard 環境を示します。元のスタンバイ・データベースは新しいプライマリ・データベースになります。現在、プライマリ・データベースはボストンにあり、スタンバイ・データベースはサンフランシスコにあります。

図 8-3 スイッチオーバー後の Data Guard 環境



スイッチオーバーの準備

8.1.1 項に示した前提条件が満たされていることを確認します。また、スイッチオーバーの場合は、次の前提条件が満たされている必要があります。

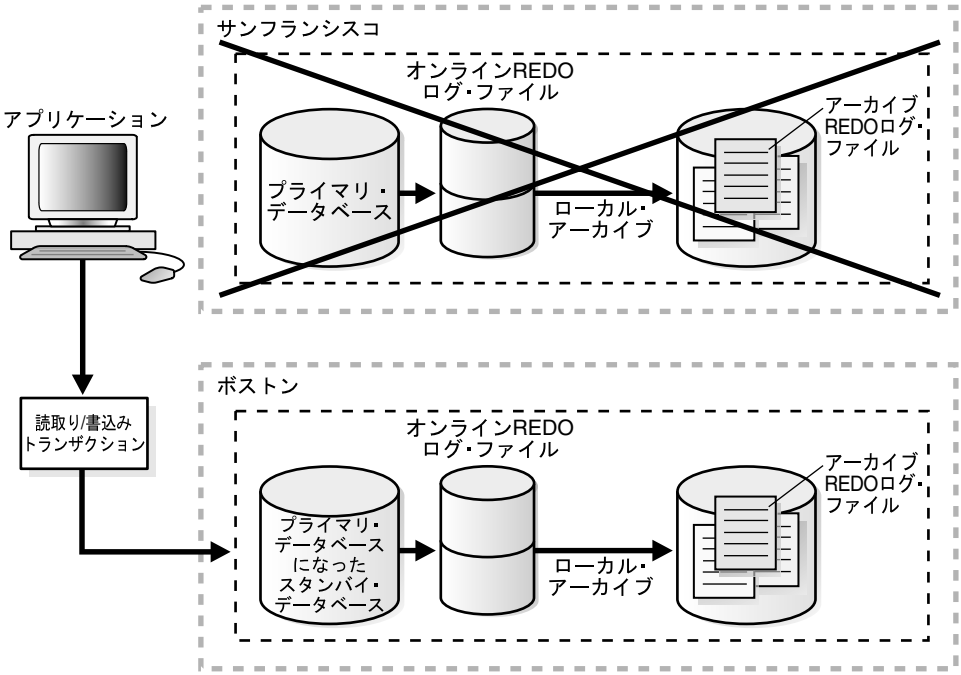
- フィジカル・スタンバイ・データベースが関与するスイッチオーバーの場合は、プライマリ・データベースがオープンし、REDO Apply がスタンバイ・データベースでアクティブであることを確認します。REDO Apply の詳細は、7.3 項「REDO データのフィジカル・スタンバイ・データベースへの適用」を参照してください。
- ロジカル・スタンバイ・データベースが関与するスイッチオーバーの場合は、プライマリおよびスタンバイ・データベース・インスタンスの両方がオープンし、SQL Apply がアクティブであることを確認します。SQL Apply の詳細は、7.4 項「REDO データのロジカル・スタンバイ・データベースへの適用」を参照してください。

8.1.4 フェイルオーバー

通常、フェイルオーバーは、プライマリ・データベースが使用不可能になり、適正な時間内にプライマリ・データベースをリストアしてサービスを再開できない場合にのみ使用します。フェイルオーバー時に実行するアクションは、フェイルオーバーに関与するスタンバイ・データベースがロジカルかフィジカルか、フェイルオーバー時の Data Guard 構成の状態、およびフェイルオーバーを開始するために使用する特定の SQL 文によって異なります。

図 8-4 に、サンフランシスコにあるプライマリ・データベースからボストンにあるフィジカル・スタンバイ・データベースへのフェイルオーバーの結果を示します。

図 8-4 スタンバイ・データベースへのフェイルオーバー



フェイルオーバーの準備

可能な場合は、フェイルオーバーを実行する前に、使用可能な未適用のプライマリ・データベース REDO データを可能なかぎりスタンバイ・データベースに転送する必要があります。

8-2 ページの 8.1.1 項「ロールの推移の準備」に示した前提条件が満たされていることを確認します。また、フェイルオーバーの場合は、次の前提条件が満たされている必要があります。

- 最大保護モードで実行中のスタンバイ・データベースがフェイルオーバーに関与する場合は、スタンバイ・データベースで次の文を発行して、データベースを最大パフォーマンス・モードにします。

  ```
  SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE PERFORMANCE;
  ```

  適切なスタンバイ・データベースが使用できる場合は、フェイルオーバーが完了した後に、新しいプライマリ・データベースで希望の保護モードをリセットできます。

  最大保護モードに設定されているスタンバイ・データベースはフェイルオーバーできないため、この作業が必要になります。さらに、最大保護モードのプライマリ・データベースがスタンバイ・データベースと通信中の場合は、スタンバイ・データベースを最大保護モードから最大パフォーマンス・モードに変更するために ALTER DATABASE 文を発行しても失敗します。フェイルオーバーを実行すると元のプライマリ・データベースは Data Guard 構成から削除されるため、この機能によって、最大保護モードで実行されているプライマリ・データベースを計画外のフェイルオーバーの影響から保護します。

  注意： スタンバイ・データベースが正しく更新されているかどうかをテストするために、スタンバイ・データベースにフェイルオーバーしないでください。かわりに、次のようにします。

  - 3.2.7 項「フィジカル・スタンバイ・データベースが正しく実行されているかどうかの確認」を参照してください。
  - 4.2.6 項「ロジカル・スタンバイ・データベースが正しく実行されているかどうかの確認」を参照してください。

8.1.5 ロールの推移のトリガー

ロールの推移が発生するたびに、DB_ROLE_CHANGE システム・イベントにシグナルが送信されます。このシステム・イベントには、ロールの推移が発生したときにデータベースがオープンしている場合は、すぐにシグナルが送信されます。あるいはロールの推移が発生したときにデータベースがクローズしている場合は、次回オープン時にシグナルが送信されます。

DB_ROLE_CHANGE システム・イベントを使用すると、ロールの推移が発生するたびに一連のアクションを実行するトリガーを起動できます。

8.2 フィジカル・スタンバイ・データベースが関与するロールの推移

この項では、フィジカル・スタンバイ・データベースに対するスイッチオーバーまたはフェイルオーバーの実行方法を説明します。

8.2.1 フィジカル・スタンバイ・データベースへのスイッチオーバーの実行

この項では、フィジカル・スタンバイ・データベースへのスイッチオーバーの実行方法を説明します。

スイッチオーバーは、プライマリ・データベースで開始し、ターゲット・スタンバイ・データベースで完了します。

手順 1 プライマリ・データベースがスタンバイ・ロールに切替え可能であることを確認する。

プライマリ・データベースで V$DATABASE ビューの SWITCHOVER_STATUS 列を問い合せます。

次に例を示します。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
 -----------------
 TO STANDBY
 1 row selected
```

値 TO STANDBY または SESSIONS ACTIVE は、プライマリ・データベースをスタンバイ・ロールに切替え可能であることを示します。これらのいずれの値も戻されない場合、REDO 転送が正しく構成されていないか、正しく機能していないため、スイッチオーバーは不可能です。REDO 転送の構成および監視の詳細は、第 6 章を参照してください。

手順 2 プライマリ・データベースでスイッチオーバーを開始する。

プライマリ・データベースで次の SQL 文を発行してスタンバイ・ロールに切り替えます。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY WITH
SESSION SHUTDOWN;
```

この文では、プライマリ・データベースをフィジカル・スタンバイ・データベースに変換します。現行の制御ファイルは、スイッチオーバーの前にカレント SQL セッション・トレース・ファイルにバックアップされます。これによって、必要に応じて現行の制御ファイルを再作成できるようになります。

注意： 前の手順で実行した問合せで TO STANDBY が戻された場合、WITH SESSION SHUTDOWN 句をスイッチオーバー文から省略できます。

手順 3 元のプライマリ・データベースを停止してマウントする。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
```

スイッチオーバー・プロセスのこの時点では、元のプライマリ・データベースがフィジカル・スタンバイ・データベースです（図 8-2 を参照）。

手順 4 スイッチオーバー・ターゲットでプライマリ・ロールに切り替える準備が完了していることを確認する。

スタンバイ・データベースで V$DATABASE ビューの SWITCHOVER_STATUS 列を問い合せます。

次に例を示します。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO_PRIMARY
1 row selected
```

値 TO STANDBY または SESSIONS ACTIVE は、スタンバイ・データベースでプライマリ・ロールに切り替える準備が完了していることを示します。これらのいずれの値も戻されない場合、REDO apply がアクティブで、REDO 転送が正しく構成されて機能していることを確認します。値 TO PRIMARY または SESSIONS ACTIVE が戻されるまで、この列の問合せを続行します。

手順 5 ターゲット・フィジカル・スタンバイ・データベース・ロールからプライマリ・ロールに切り替える。

次の SQL 文をターゲット・フィジカル・スタンバイ・データベースで発行します。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WITH SESSION SHUTDOWN;
```

> **注意：** 前の手順で実行した問合せで TO PRIMARY が戻された場合、WITH SESSION SHUTDOWN 句をスイッチオーバー文から省略できます。

手順 6 新しいプライマリ・データベースをオープンする。

```
SQL> ALTER DATABASE OPEN;
```

手順 7 新しいフィジカル・スタンバイ・データベースで REDO Apply を開始する。

次に例を示します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
 DISCONNECT FROM SESSION;
```

8.2.2 フィジカル・スタンバイ・データベースへのフェイルオーバーの実行

手順 1 REDO ギャップを識別して解決する。

V$ARCHIVE_GAP ビューを問い合せて、ターゲット・スタンバイ・データベースに REDO ギャップがあるかどうか判別します。

次に例を示します。

```
SQL> SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE# FROM V$ARCHIVE_GAP;
THREAD#    LOW_SEQUENCE# HIGH_SEQUENCE#
---------- ------------- --------------
         1            90             92
```

この例では、スレッド 1 の順序 90、91 および 92 のアーカイブ REDO ログ・ファイルがギャップです。

可能であれば、欠落しているすべてのアーカイブ REDO ログ・ファイルを、プライマリ・データベースからターゲット・スタンバイ・データベースにコピーして登録します。スレッドごとに実行する必要があります。

次に例を示します。

```
SQL> ALTER DATABASE REGISTER PHYSICAL LOGFILE 'filespec1';
```

手順 2 すべてのギャップが解決されるまで手順 1 を繰り返す。

手順 1 で実行する問合せでは、順序番号が最も大きいギャップに関する情報のみが表示されます。ギャップを解決した後、行が戻されなくなるまで問合せを繰り返します。

手順 3 欠落した他のアーカイブ REDO ログ・ファイルをコピーする。

欠落したアーカイブ REDO ログ・ファイルが他に存在するかどうかを判断するには、ターゲット・スタンバイ・データベースで V$ARCHIVED_LOG ビューを問い合せて、スレッドごとに最も大きい順序番号を取得します。

次に例を示します。

```
SQL> SELECT UNIQUE THREAD# AS THREAD, MAX(SEQUENCE#)
  2> OVER (PARTITION BY thread#) AS LAST from V$ARCHIVED_LOG;

    THREAD       LAST
---------- ----------
         1        100
```

可能であれば、ターゲット・スタンバイ・データベースで使用可能な最も大きい順序番号より大きい順序番号を含むアーカイブ REDO ログ・ファイルを、プライマリ・データベースからターゲット・スタンバイ・データベースにコピーして登録します。スレッドごとに実行する必要があります。

次に例を示します。

```
SQL> ALTER DATABASE REGISTER PHYSICAL LOGFILE 'filespec1';
```

欠落したアーカイブ REDO ログ・ファイルがターゲット・スタンバイ・データベースにコピーされたら、手順 1 に戻り、追加の REDO ギャップが発生していないかどうか確認します。

手順 1 から手順 3 を実行した後、アーカイブ REDO ログ・ファイルのすべてのギャップを解決できない場合（障害の発生したプライマリ・データベースのホスト・システムへのアクセス権がない場合など）、フェイルオーバー時にデータが消失する可能性があります。

手順 4 REDO Apply を停止する。

次の SQL 文を発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

手順 5 受信したすべての REDO データの適用を終了する。

次の SQL 文を発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
```

この文がエラーなしに終了した場合、手順 6 に進みます。

エラーが発生した場合、受信された REDO データに適用されていないものがあります。エラーの原因を解決し、文を再発行してから次の手順に進みます。

エラー状態を解決できない場合でも、次の SQL 文を発行してフェイルオーバーを実行できます（一部のデータが消失する可能性があります）。

```
SQL> ALTER DATABASE ACTIVATE PHYSICAL STANDBY DATABASE;
```

ACTIVATE 文が完了した後、手順 8 に進みます。

手順 6 ターゲット・スタンバイ・データベースがプライマリ・データベースになる準備が完了していることを確認する。

ターゲット・スタンバイ・データベースで V$DATABASE ビューの SWITCHOVER_STATUS 列を問い合せます。

次に例を示します。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO PRIMARY
1 row selected
```

値 TO STANDBY または SESSIONS ACTIVE は、スタンバイ・データベースでプライマリ・ロールに切り替える準備ができていることを示します。これらのいずれの値も戻されない場合、REDO apply がアクティブで、TO PRIMARY または SESSIONS ACTIVE のいずれかが戻されるまでこのビューを問い合せ続けることを確認します。

手順 7 フィジカル・スタンバイ・データベース・ロールからプライマリ・ロールに切り替える。

次の SQL 文を発行します。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WITH SESSION SHUTDOWN;
```

注意： 前の手順で実行された SWITCHOVER_STATUS 列の問合せで TO PRIMARY が戻された場合、WITH SESSION SHUTDOWN 句をスイッチオーバー文から省略できます。

手順 8 新しいプライマリ・データベースをオープンする。

```
SQL> ALTER DATABASE OPEN;
```

手順 9 新しいプライマリ・データベースをバックアップする。

新しいプライマリ・データベースの全体バックアップを作成することをお薦めします。

手順 10 障害の発生したプライマリ・データベースをリストアする（オプション）。

フェイルオーバーの完了後、元のプライマリ・データベースは 13.2 項または 13.7 項で説明する方法に従って、新しいプライマリ・データベースのフィジカル・スタンバイ・データベースに変換できます。あるいは、3.2 項で説明した方法に従って、新しいプライマリ・データベースのバックアップからフィジカル・スタンバイ・データベースとして再作成できます。

元のプライマリ・データベースをスタンバイ・ロールで実行したら、スイッチオーバーを実行してプライマリ・ロールにリストアできます。

8.3 ロジカル・スタンバイ・データベースが関与するロールの推移

この項では、ロジカル・スタンバイ・データベースが関与するスイッチオーバーおよびフェイルオーバーの実行方法を説明します。

8.3.1 ロジカル・スタンバイ・データベースへのスイッチオーバーの実行

プライマリ・データベースとロジカル・スタンバイ・データベースとの間でスイッチオーバーを実行する場合、スイッチオーバーは、常にプライマリ・データベースで開始し、ロジカル・スタンバイ・データベースで完了してください。次の手順を記載されているとおりの順序で実行しなかった場合、スイッチオーバーは成功しません。

手順 1 プライマリ・データベースでスイッチオーバーを実行できるかどうかを確認する。

スイッチオーバーを実行できるかどうかを確認するには、現行のプライマリ・データベースで V$DATABASE 固定ビューの SWITCHOVER_STATUS 列を問い合せます。

次に例を示します。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO STANDBY
1 row selected
```

SWITCHOVER_STATUS 列の値 TO STANDBY または SESSIONS ACTIVE は、プライマリ・データベースをロジカル・スタンバイ・ロールに切替え可能であることを示します。これらの値のいずれかが表示されない場合は、Data Guard 構成が正常に機能していることを確認してください（たとえば、LOG_ARCHIVE_DEST_*n* パラメータのすべての値が正しく指定されていることを確認します）。V$DATABASE ビューの SWITCHOVER_STATUS 列に対するその他の有効な値は、『Oracle Database リファレンス』を参照してください。

手順 2 現行のプライマリ・データベースのスイッチオーバーを準備する。

ロジカル・スタンバイ・データベース・ロール用に現行のプライマリ・データベースを準備するには、プライマリ・データベースで次の SQL 文を発行します。

```
SQL> ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY;
```

この文は、現行のプライマリ・データベースがロジカル・スタンバイ・ロールに切り替わり、新しいプライマリ・データベースから REDO データの受信を開始することを、現行のプライマリ・データベースに通知します。この手順は、現行のロジカル・スタンバイ・データベースの REDO ストリームで記録される LogMiner ディクショナリを受信するための準備中に、プライマリ・データベースで実行します。手順 3 を参照してください。

この操作に成功すると、V$DATABASE.SWITCHOVER_STATUS 列に値 PREPARING SWITCHOVER が表示されます。

手順 3 ターゲット・ロジカル・スタンバイ・データベースのスイッチオーバーを準備する。

スイッチオーバーのターゲットであるロジカル・スタンバイ・データベースで LogMiner ディクショナリを作成するには、次の文を使用します。

```
SQL> ALTER DATABASE PREPARE TO SWITCHOVER TO PRIMARY;
```

また、この文は、現行のプライマリ・データベース、および Data Guard 構成内の他のスタンバイ・データベースに REDO データの転送を開始するロジカル・スタンバイ・データベースで、REDO 転送サービスを開始します。このロジカル・スタンバイ・データベースから REDO データを受信するサイトは、REDO データを受け入れますが、適用はしません。

このスイッチオーバーは、処理量とデータベースのサイズによって完了までに時間がかかる場合があります。

ロジカル・スタンバイ・データベースの V$DATABASE.SWITCHOVER_STATUS には、LogMiner ディクショナリが REDO ストリームに記録されている間、最初は PREPARING DICTIONARY が表示されます。正常に完了すると、SWITCHOVER_STATUS 列に PREPARING SWITCHOVER が表示されます。

手順 4 現行のプライマリ・データベースで将来のプライマリ・データベースの REDO ストリームに対する準備が完了していることを確認する。

プライマリ・データベースからロジカル・スタンバイ・ロールへのロールの推移を完了する前に、プライマリ・データベースで V$DATABASE 固定ビューの SWITCHOVER_STATUS 列を問い合せ、プライマリ・データベースで LogMiner ディクショナリが受信されていることを確認します。LogMiner ディクショナリが受信されていない場合、スイッチオーバーは進行できません。これは、現行のプライマリ・データベースでは、将来のプライマリ・データベースから送信される REDO レコードを解析できないためです。SWITCHOVER_STATUS 列は、スイッチオーバーの進捗を示します。

問合せが TO LOGICAL STANDBY 値を戻した場合は、手順 5 に進むことができます。次に例を示します。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO LOGICAL STANDBY
1 row selected
```

注意： 次の文を示された順序で発行すると、スイッチオーバー操作を取り消すことができます。

1. プライマリ・データベースでスイッチオーバーを取り消します。

   ```
   SQL> ALTER DATABASE PREPARE TO SWITCHOVER CANCEL;
   ```

2. ロジカル・スタンバイ・データベースでスイッチオーバーを取り消します。

   ```
   SQL> ALTER DATABASE PREPARE TO SWITCHOVER CANCEL;
   ```

手順 5 プライマリ・データベースをロジカル・スタンバイ・データベース・ロールに切り替える。

プライマリ・データベースからロジカル・スタンバイ・データベース・ロールへのロールの推移を完了するには、次の SQL 文を発行します。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;
```

この文は、プライマリ・データベースで現在のトランザクションがすべて終了するまで待機し、新規ユーザーによる新規トランザクションの開始を防止して、スイッチオーバーがコミットされる時点を設定します。

この文を実行すると、ユーザーはロジカル・スタンバイ・データベースでメンテナンスされているデータの変更もできなくなります。スイッチオーバーを迅速に実行するには、スイッチオーバー文を発行する前に、プライマリ・データベースが静止状態で更新アクティビティが実行されていないことを確認してください（たとえば、すべてのユーザーをプライマリ・データベースから一時的にログオフします）。V$TRANSACTION ビューを問い合せると、この文の実行を遅延させる可能性のある現在進行中のトランザクションのステータス情報を取得できます。

この時点でプライマリ・データベースのロールが推移して、スタンバイ・データベース・ロールで実行されます。

プライマリ・データベースをロジカル・スタンバイ・データベース・ロールに推移した場合は、データベースを停止して再起動する必要はありません。

手順 6 使用可能なすべての REDO が、新規プライマリ・データベースになるターゲット・ロジカル・スタンバイ・データベースに適用されたことを確認する。

プライマリ・データベースからロジカル・スタンバイ・ロールへのロールの推移を完了し、構成内のスタンバイ・データベースがスイッチオーバー通知を受け取った後、ターゲット・スタンバイ・データベースで V$DATABASE 固定ビューの SWITCHOVER_STATUS 列を問い合せ、ターゲット・スタンバイ・データベースがスイッチオーバー通知を処理したかどうかを確認する必要があります。使用可能なすべての REDO レコードがロジカル・スタンバイ・データベースに適用されると、SQL Apply は予想されるロールの推移の準備を自動的に停止します。

スイッチオーバー中に SWITCHOVER_STATUS 値が更新され、進捗を示します。ステータスが TO PRIMARY の場合は、手順 7 に進むことができます。

次に例を示します。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO PRIMARY
1 row selected
```

V$DATABASE ビューの SWITCHOVER_STATUS 列に対するその他の有効な値は、『Oracle Database リファレンス』を参照してください。

手順 7 ターゲット・ロジカル・スタンバイ・データベースをプライマリ・データベース・ロールに切り替える。

プライマリ・ロールに切り替えるロジカル・スタンバイ・データベースで、次の SQL 文を使用し、ロジカル・スタンバイ・データベースをプライマリ・ロールに切り替えます。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

Data Guard 構成内のロジカル・スタンバイ・データベースは、停止して再起動する必要はありません。8-3 ページの 8.1.2 項で説明したように、構成内の他のすべてのロジカル・スタンバイは新しいプライマリのスタンバイになりますが、フィジカル・スタンバイ・データベースは、元のプライマリ・データベースのスタンバイのままです。

手順 8 新規のロジカル・スタンバイ・データベースで SQL Apply を開始する。

新しいロジカル・スタンバイ・データベースで SQL Apply を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

8.3.2 ロジカル・スタンバイ・データベースへのフェイルオーバーの実行

この項では、ロジカル・スタンバイ・データベースが関与するフェイルオーバーの実行方法を説明します。ロジカル・スタンバイ・データベースが関与するフェイルオーバーによるロールの推移では、障害の発生したプライマリ・データベースとすべてのロジカル・スタンバイ・データベースで対処措置を実行する必要があります。障害の発生したプライマリ・データベースでフラッシュバック・データベースが有効化されていない場合は、現行のプライマリ・データベースから作成したバックアップを使用してデータベースを再作成する必要があります。それ以外の場合は、13.2 項で説明する手順に従って、障害の発生したプライマリ・データベースを新しいプライマリ・データベースのロジカル・スタンバイ・データベースに変換できます。

構成に対する保護モードおよび REDO 転送サービスに対して選択した属性に従って、プライマリ・データベースの修正の一部またはすべてを自動的にリカバリすることも可能です。

手順 1 欠落したアーカイブ REDO ログ・ファイルを、新しいプライマリ・データベースの候補となるターゲット・ロジカル・スタンバイ・データベースにコピーし、登録する。

構成に含まれるコンポーネントの状態によっては、プライマリ・データベースのアーカイブ REDO ログ・ファイルにアクセスできる場合があります。アクセスする場合の手順は、次のとおりです。

1. ロジカル・スタンバイ・データベースでアーカイブ REDO ログ・ファイルが欠落しているかどうかを判断します。
2. 欠落しているログ・ファイルを、プライマリ・データベースからロジカル・スタンバイ・データベースにコピーします。
3. コピーしたログ・ファイルを登録します。

次の文を発行して、アーカイブ REDO ログ・ファイルをロジカル・スタンバイ・データベースに登録できます。次に例を示します。

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
  2> '/disk1/oracle/dbs/log-%r_%s_%t.arc';
Database altered.
```

手順 2 リモート宛先を使用可能にする。

ロールベースの宛先をまだ構成していない場合は、新しいプライマリ・データベースのリモート・ロジカル・スタンバイ宛先に対応する初期化パラメータを特定して、これらの宛先ごとに REDO データのアーカイブを手動で使用可能にします。

たとえば、LOG_ARCHIVE_DEST_2 パラメータで定義したリモート宛先のアーカイブを可能にするには、次の文を発行します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE SCOPE=BOTH;
```

後で新しいプライマリ・データベースを再起動した場合にもこの変更を保持するには、適切なテキスト初期化パラメータ・ファイルまたはサーバー・パラメータ・ファイルを更新します。通常、データベースがプライマリ・ロールで動作する場合はリモート宛先へのアーカイブを可能にし、データベースがスタンバイ・ロールで動作する場合はリモート宛先へのアーカイブを不可能にする必要があります。

手順 3 新しいプライマリ・データベースをアクティブにする。

ターゲット・ロジカル・スタンバイ・データベース（新規プライマリ・ロールに推移させるデータベース）で、次の文を発行します。

```
SQL> ALTER DATABASE ACTIVATE LOGICAL STANDBY DATABASE FINISH APPLY;
```

この文は RFS プロセスを停止し、ロジカル・スタンバイ・データベースがプライマリ・データベースになる前にスタンバイ REDO ログ・ファイルに残っている REDO データを適用し、SQL Apply を停止して、データベースをプライマリ・データベース・ロールでアクティブ化します。

FINISH APPLY 句が指定されていない場合、現行のスタンバイ REDO ログ・ファイルの未適用の REDO は、スタンバイ・データベースがプライマリ・データベースになるまで適用されません。

手順 4 フェイルオーバー後に他のスタンバイ・データベースをリカバリする。

13.1 項で説明する方法に従って、既存のロジカル・スタンバイ・データベースが新しいプライマリ・データベースを引き続き保護できるようにします。

手順 5 新しいプライマリ・データベースをバックアップする。

Data Guard のデータベース・フェイルオーバー直後に、新しいプライマリ・データベースのバックアップを作成します。バックアップを即時に実行することは、必要な安全策です。これは、データベースの完全なバックアップ・コピーを作成せずにフェイルオーバーを行うと、変更をリカバリできないためです。

手順 6 障害の発生したプライマリ・データベースをリストアする。

フェイルオーバーの完了後、元のプライマリ・データベースは 13.2 項で説明する方法に従って、新しいプライマリ・データベースのロジカル・スタンバイ・データベースに変換できます。あるいは、第 4 章で説明したように、新しいプライマリ・データベースのバックアップからロジカル・スタンバイ・データベースとして再作成できます。

元のプライマリ・データベースをスタンバイ・データベースに変換したら、スイッチオーバーを実行してプライマリ・ロールにリストアできます。

8.4 ロールの推移後のフラッシュバック・データベースの使用

ロールの推移後に、必要に応じて FLASHBACK DATABASE コマンドを使用して、データベースをロールの推移が発生する前の時点またはシステム変更番号（SCN）まで戻すことができます。

フィジカル・スタンバイ・データベース環境では、Data Guard 構成を維持するためにプライマリ・データベースとすべてのスタンバイ・データベースのフラッシュバックが必要になる場合があります。プライマリ・データベースを特定の SCN または時点までフラッシュバックする場合は、すべてのスタンバイ・データベースを同じ（またはそれ以前の）SCN または時点までフラッシュバックする必要があります。これにより、REDO Apply の開始後に、フィジカル・スタンバイ・データベースではプライマリ・データベースから受信した REDO データの適用が自動的に開始されます。

この方法でプライマリ・データベースまたはスタンバイ・データベースをフラッシュバックすると、過去のスイッチオーバーを認識する必要がありません。Oracle では、SCN または時点が過去のスイッチオーバーより前であれば、過去のスイッチオーバーにまたがって自動的にフラッシュバックできます。

注意： ロールの推移が発生する前に、データベースでフラッシュバック・データベースを有効化しておく必要があります。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

8.4.1 スイッチオーバー後のフラッシュバック・データベースの使用

スイッチオーバー後に FLASHBACK DATABASE コマンドを使用して、データベースをスイッチオーバー発生前の時点またはシステム変更番号（SCN）に戻すことができます。

スイッチオーバーにフィジカル・スタンバイ・データベースが関与していた場合、フラッシュバック操作中はプライマリおよびスタンバイ・データベース・ロールが保持されます。つまり、データベースが実行中のロールは、フラッシュバックした時点またはターゲット SCN までデータベースがフラッシュバックされても変化しません。スイッチオーバー後からフラッシュバックの前までフィジカル・スタンバイ・ロールで実行されていたデータベースは、フラッシュバック・データベース操作後もフィジカル・スタンバイ・データベースで実行されます。

スイッチオーバーにロジカル・スタンバイ・データベースが関与していた場合、フラッシュバックするとスタンバイ・データベースのロールはフラッシュバックした時点またはターゲット SCN でのロールに変更されます。

8.4.2 フェイルオーバー後のフラッシュバック・データベースの使用

フラッシュバック・データベースを使用して、障害の発生したプライマリ・データベースをフェイルオーバー発生前の時点に変換してから、スタンバイ・データベースに変換できます。詳細な手順は、13.2 項「フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換」を参照してください。

9

フィジカルおよびスナップショット・スタンバイ・データベースの管理

この章では、フィジカル・スタンバイ・データベースおよびスナップショット・スタンバイ・データベースの管理方法について説明します。次の項目で構成されています。

- フィジカル・スタンバイ・データベースの起動と停止
- フィジカル・スタンバイ・データベースのオープン
- フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更
- OPEN RESETLOGS 文を使用したリカバリ
- プライマリ、フィジカル・スタンバイおよびスナップショット・スタンバイ・データベースの監視
- REDO Apply のチューニング
- スナップショット・スタンバイ・データベースの管理

Data Guard Broker を使用してフィジカル・スタンバイ・データベースおよびスナップショット・スタンバイ・データベースの管理を簡素化する方法の詳細は、『Oracle Data Guard Broker』を参照してください。

9.1 フィジカル・スタンバイ・データベースの起動と停止

この項では、フィジカル・スタンバイ・データベースを起動および停止する方法について説明します。

9.1.1 フィジカル・スタンバイ・データベースの起動

フィジカル・スタンバイ・データベースを起動するには、SQL*Plus STARTUP コマンドを使用します。SQL*Plus STARTUP コマンドは、引数を指定せずに起動すると、フィジカル・スタンバイ・データベースを読取り専用モードで起動、マウントおよびオープンします。

フィジカル・スタンバイ・データベースは、マウントまたはオープンされると、プライマリ・データベースから REDO データを受信できます。

REDO Apply の詳細は 7.3 項を、フィジカル・スタンバイ・データベースを読取り専用モードでオープンする方法の詳細は 9.2 項を参照してください。

注意： プライマリ・データベースからの REDO データをまだ受信していないフィジカル・スタンバイ・データベースで REDO Apply を開始すると、ORA-01112 メッセージが戻されることがあります。このメッセージは、REDO Apply がメディア・リカバリ用の開始順序番号を判別できないことを示します。このエラーが発生した場合は、スタンバイ・データベースでアーカイブ REDO ログ・ファイルをプライマリ・データベースから手動で取得して登録するか、または REDO 転送が開始されるまで待機した後、REDO Apply を開始します。

9.1.2 フィジカル・スタンバイ・データベースの停止

REDO Apply を停止してフィジカル・スタンバイ・データベースを停止するには、SQL*Plus SHUTDOWN コマンドを使用します。停止処理が完了するまで、データベース停止を開始したセッションに制御が戻されません。

プライマリ・データベースが起動して稼働している場合は、フィジカル・スタンバイ・データベースを停止する前に、プライマリ・データベースでスタンバイ宛先を遅延し、ログ・スイッチを実行します。

9.2 フィジカル・スタンバイ・データベースのオープン

フィジカル・スタンバイ・データベースは、読取り専用アクセス用にオープンし、問合せをプライマリ・データベースからオフロードするために使用できます。

Oracle Active Data Guard オプションのライセンスを購入済の場合、REDO Apply がアクティブである間は、フィジカル・スタンバイ・データベースをオープンできます。この機能は、リアルタイム問合せと呼ばれます。詳細は、9.2.1 項を参照してください。

Oracle Active Data Guard オプションのライセンスを未購入の場合、REDO Apply がアクティブである間は、フィジカル・スタンバイ・データベースをオープンできません。そのため、フィジカル・スタンバイ・データベース・インスタンスのオープン時または REDO Apply の開始時には、次のルールに従う必要があります。

- REDO Apply は、フィジカル・スタンバイ・データベース・インスタンスをオープンする前に停止する必要がある。
- 1 つ以上のフィジカル・スタンバイ・インスタンスがオープンしている場合、REDO Apply を開始する前にこれらのインスタンスをクローズする必要がある。

注意： フィジカル・スタンバイ・データベースで分散問合せを実行する前に、SET TRANSACTION READ ONLY SQL 文を実行する必要があります。

9.2.1 リアルタイム問合せ

フィジカル・スタンバイ・データベースは、Oracle Active Data Guard オプションのライセンスを購入済の場合、REDO Apply がアクティブである間は読取り専用アクセス用にオープンできます。この機能は、リアルタイム問合せと呼ばれます。

フィジカル・スタンバイ・データベース・インスタンスは、REDO Apply がそのインスタンスまたは他のマウント済インスタンスでアクティブである場合、オープンできません。次の SQL 文を使用して REDO Apply を停止し、スタンバイ・インスタンスを読取り専用でオープンして REDO Apply を再開します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> ALTER DATABASE OPEN;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
  2> DISCONNECT;
```

> **注意：** REDO Apply がオープン・インスタンスでアクティブである場合は、REDO Apply を停止しなくても他のフィジカル・スタンバイ・データベース・インスタンスをオープンできます。

REDO Apply は、他のインスタンスがオープンされている場合、マウント済フィジカル・スタンバイ・インスタンスでは開始できません。REDO Apply を開始する予定のインスタンスは、REDO Apply の開始前にオープンしておく必要があります。

フィジカル・スタンバイ適用インスタンスが（強制終了やノード・クラッシュなどにより）異常終了すると、フィジカル・スタンバイ・データベースのオープンは「ORA-16004: バックアップ・データベースをリカバリしてください。」エラーとなります。このエラーが発生した場合は、REDO Apply を開始した後、フィジカル・スタンバイ・データベースを再度オープンする前に REDO Apply を停止する必要があります。

例 9-1 で、Boston というフィジカル・スタンバイ・データベース・インスタンスの障害およびリカバリと、リカバリ後に REDO Apply がアクティブであるときの Boston のオープン（リアルタイム問合せ）を説明します。

例 9-1 リアルタイム問合せ

この例の初めに、フィジカル・スタンバイ・インスタンスの Boston を REDO Apply がアクティブの状態でマウントします。次に、Boston は停電のためにクラッシュし、再起動されます。

```
SQL> STARTUP
ORACLE instance started.

Total System Global Area  234364928 bytes
Fixed Size                  1298908 bytes
Variable Size             209718820 bytes
Database Buffers           16777216 bytes
Redo Buffers                6569984 bytes
Database mounted.
ORA-16004: backup database requires recovery
ORA-01196: file 1 is inconsistent due to a failed media recovery session
ORA-01110: data file 1: '/scratch/datafiles/oracle/dbs/system1.f'
```

フィジカル・スタンバイ・データベースは、REDO Apply がアクティブであるときに Boston インスタンスがクラッシュしたため、一貫性が欠如しています。REDO Apply を開始して、データベースが一貫性のある SCN までリカバリできるようにします。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;
Database altered.
```

1 分ほど待機した後、REDO Apply を停止します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
Database altered.
```

アラート・ログには、通常予想される警告「ORA-16037: 管理リカバリ操作の取消がユーザーからリクエストされました。」がリストされます。

この時点で、Boston は一貫性のある SCN までリカバリされました。

Boston をオープンし、REDO Apply を開始します。

```
SQL> ALTER DATABASE OPEN;
Database altered.

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING
  2> CURRENT LOGFILE DISCONNECT;
Database altered.
```

この時点で、Boston はオープンされ、REDO Apply はアクティブです。つまり、リアルタイム問合せは有効です。

9.3 フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更

プライマリ・データベースに対して行われる構造の変更のほとんどは、REDO データによってフィジカル・スタンバイ・データベースに自動的に伝播します。表 9-1 に、フィジカル・スタンバイ・データベースで手動操作が必要なプライマリ・データベースの構造および構成の変更を示します。

表 9-1 フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更

参照先	プライマリ・データベースの変更	フィジカル・スタンバイ・データベースで必要なアクション
9.3.1 項	データファイルの追加または表領域の作成	STANDBY_FILE_MANAGEMENT データベース初期化パラメータが AUTO に設定されている場合、アクションは必要ない。このパラメータが MANUAL に設定されている場合は、新しいデータファイルをフィジカル・スタンバイ・データベースにコピーする必要がある。
9.3.2 項	表領域またはデータファイルの削除	DROP または DELETE コマンドを含む REDO データがフィジカル・スタンバイに適用された後に、プライマリ・データベースとフィジカル・スタンバイ・データベースからデータファイルを削除する。
9.3.3 項	トランスポータブル表領域の使用	プライマリ・データベースとフィジカル・スタンバイ・データベースの間で表領域を移動する。
9.3.4 項	データファイルの改名	フィジカル・スタンバイ・データベースでデータファイル名を変更する。
9.3.5 項	REDO ログ・ファイル・グループの追加または削除	フィジカル・スタンバイ・データベースで REDO ログおよびスタンバイ REDO ログの構成を評価し、必要に応じて調整する。
9.3.6 項	NOLOGGING または UNRECOVERABLE 句を使用した DML または DDL 操作の実行	ログに記録されていない変更を含むデータファイルをフィジカル・スタンバイ・データベースにコピーする。

表 9-1 フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更（続き）

参照先	プライマリ・データベースの変更	フィジカル・スタンバイ・データベースで必要なアクション
9.3.7 項	管理権限の付与または取消し、あるいは管理権限を持つユーザーのパスワードの変更	REMOTE_LOGIN_PASSWORDFILE 初期化パラメータが SHARED または EXCLUSIVE に設定されている場合は、フィジカル・スタンバイ・データベースのパスワード・ファイルをプライマリ・データベースのパスワード・ファイルの最新コピーで置き換える。
9.3.8 項	TDE マスター暗号化キーのリセット	フィジカル・スタンバイ・データベースのデータベース暗号化ウォレットをプライマリ・データベースのデータベース暗号化ウォレットの最新コピーで置き換える。
第 14 章	初期化パラメータの変更	フィジカル・スタンバイ・データベースで対応する変更を初期化パラメータに加える必要があるかどうかを評価する。

9.3.1 データファイルの追加または表領域の作成

STANDBY_FILE_MANAGEMENT データベース初期化パラメータは、プライマリ・データベースへのデータファイルの追加をフィジカル・スタンバイ・データベースに自動的に伝播するかどうかを制御します。

- フィジカル・スタンバイ・データベースの STANDBY_FILE_MANAGEMENT パラメータが AUTO に設定されている場合、プライマリ・データベースに作成された新しいデータファイルはフィジカル・スタンバイ・データベースに自動的に作成されます。
- フィジカル・スタンバイ・データベースの STANDBY_FILE_MANAGEMENT データベース・パラメータが MANUAL に設定されている場合、新しいデータファイルがプライマリ・データベースに追加されると、そのファイルは、プライマリ・データベースからフィジカル・スタンバイ・データベースに手動でコピーする必要があります。

別のデータベースの既存のデータファイルがプライマリ・データベースにコピーされた場合、同様にそのファイルをスタンバイ・データベースにコピーする必要があり、STANDBY_FILE_MANAGEMENT パラメータの設定に関係なくスタンバイ制御ファイルを再作成する必要があります。

9.3.1.1 RAW デバイスでの STANDBY_FILE_MANAGEMENT パラメータの使用

注意： 次の手順は、Oracle Managed Files を使用するデータベースには使用しないでください。また、RAW デバイスのパス名がプライマリ・サーバー上とスタンバイ・サーバー上で異なる場合は、DB_FILE_NAME_CONVERT データベース初期化パラメータを使用してパス名を変換してください。

STANDBY_FILE_MANAGEMENT パラメータを AUTO に設定すると、新しいデータファイルがプライマリ・データベースに追加または削除されるたびに、手動操作しなくても対応する変更がスタンバイ・データベースに加えられます。これは、スタンバイ・データベースでファイル・システムが使用されている場合です。スタンバイ・データベースでデータファイルに RAW デバイスが使用されている場合も STANDBY_FILE_MANAGEMENT パラメータは機能しますが、手動操作が必要です。この手動操作では、REDO Apply によって新しいデータファイルを作成する REDO データが適用される前に、RAW デバイスの存在を確認します。

プライマリ・データベースで、データファイルが RAW デバイスに常駐する新規表領域を作成します。それと同時に、スタンバイ・データベースでも同じ RAW デバイスを作成します。次に例を示します。

```
SQL> CREATE TABLESPACE MTS2 -
> DATAFILE '/dev/raw/raw100' size 1m;
Tablespace created.

SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
```

スタンバイ・データベースでは、RAW デバイスが存在するため自動的にデータファイルが追加されます。スタンバイ・アラート・ログには次のように示されます。

```
Fri Apr  8 09:49:31 2005
Media Recovery Log /u01/MILLER/flash_recovery_area/MTS_STBY/archivelog/2005_04_08/
o1_mf_1_7_15ffgt0z_.arc
Recovery created file /dev/raw/raw100
Successfully added datafile 6 to media recovery
Datafile #6: '/dev/raw/raw100'
Media Recovery Waiting for thread 1 sequence 8 (in transit)
```

ただし、プライマリ・システム上で RAW デバイスが作成されてもスタンバイ上で作成されなければ、REDO Apply はファイル作成エラーにより停止します。たとえば、プライマリ・データベースで次の文を発行します。

```
SQL> CREATE TABLESPACE MTS3 –
> DATAFILE '/dev/raw/raw101' size 1m;
Tablespace created.

SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
```

スタンバイ・システムには、RAW デバイス /dev/raw/raw101 が作成されていません。アーカイブをリカバリすると、スタンバイ・アラート・ログに次のメッセージが示されます。

```
Fri Apr  8 10:00:22 2005
Media Recovery Log /u01/MILLER/flash_recovery_area/MTS_STBY/archivelog/2005_04_08/o1_
mf_1_8_15ffjrov_.arc
File #7 added to control file as 'UNNAMED00007'.
Originally created as:
'/dev/raw/raw101'
Recovery was unable to create the file as:
'/dev/raw/raw101'
MRP0: Background Media Recovery terminated with error 1274
Fri Apr  8 10:00:22 2005
Errors in file /u01/MILLER/MTS/dump/mts_mrp0_21851.trc:
ORA-01274: cannot add datafile '/dev/raw/raw101' - file could not be created
ORA-01119: error in creating database file '/dev/raw/raw101'
ORA-27041: unable to open file
Linux Error: 13: Permission denied
Additional information: 1
Some recovered datafiles maybe left media fuzzy
Media recovery may continue but open resetlogs may fail
Fri Apr  8 10:00:22 2005
Errors in file /u01/MILLER/MTS/dump/mts_mrp0_21851.trc:
ORA-01274: cannot add datafile '/dev/raw/raw101' - file could not be created
ORA-01119: error in creating database file '/dev/raw/raw101'
ORA-27041: unable to open file
Linux Error: 13: Permission denied
Additional information: 1
Fri Apr  8 10:00:22 2005
MTS; MRP0: Background Media Recovery process shutdown
ARCH: Connecting to console port...
```

9.3.1.2 エラーのリカバリ

9.3.1.1 項で説明した問題を修正する手順は、次のとおりです。

1. スタンバイ・データベースに RAW デバイスを作成し、Oracle ユーザーに権限を割り当てます。
2. V$DATAFILE ビューを問い合せます。次に例を示します。

```
SQL> SELECT NAME FROM V$DATAFILE;

NAME.
 ----------------------------------------------------------------------------------
/u01/MILLER/MTS/system01.dbf
/u01/MILLER/MTS/undotbs01.dbf
/u01/MILLER/MTS/sysaux01.dbf
/u01/MILLER/MTS/users01.dbf
/u01/MILLER/MTS/mts.dbf
/dev/raw/raw100
/u01/app/oracle/product/10.1.0/dbs/UNNAMED00007

SQL> ALTER SYSTEM SET -
> STANDBY_FILE_MANAGEMENT=MANUAL;

SQL> ALTER DATABASE CREATE DATAFILE
2  '/u01/app/oracle/product/10.1.0/dbs/UNNAMED00007'
3  AS
4  '/dev/raw/raw101';
```

3. スタンバイ・アラート・ログに、次のような情報が表示されます。

```
Fri Apr  8 10:09:30 2005
alter database create datafile
'/dev/raw/raw101' as '/dev/raw/raw101'

Fri Apr  8 10:09:30 2005
Completed: alter database create datafile
'/dev/raw/raw101' a
```

4. スタンバイ・データベースで、STANDBY_FILE_MANAGEMENT を AUTO に設定して REDO Apply を再開します。

```
SQL> ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT=AUTO;
SQL> RECOVER MANAGED STANDBY DATABASE DISCONNECT;
```

この時点で、REDO Apply は新規の RAW デバイスのデータファイルを使用し、リカバリが続行されます。

9.3.2 表領域の削除とデータファイルの削除

プライマリ・データベースから表領域またはデータファイルを削除した場合、対応するデータファイルをフィジカル・スタンバイ・データベースから削除する必要があります。次の例に、表領域を削除する方法を示します。

```
SQL> DROP TABLESPACE tbs_4;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

削除されたデータファイルがデータベースに属していないことを確認するには、V$DATAFILE ビューを問い合せます。

過去の変更を含む REDO データがスタンバイ・データベースに適用された後、スタンバイ・システムで対応するデータファイルを削除します。次に例を示します。

```
% rm /disk1/oracle/oradata/payroll/s2tbs_4.dbf
```

削除した表領域の REDO 情報がスタンバイ・データベースで適用されたことを確認した後、プライマリ・データベースで表領域のデータファイルを削除できます。次に例を示します。

```
% rm /disk1/oracle/oradata/payroll/tbs_4.dbf
```

9.3.2.1 DROP TABLESPACE INCLUDING CONTENTS AND DATAFILES の使用

プライマリ・データベースで SQL DROP TABLESPACE INCLUDING CONTENTS AND DATAFILES 文を発行すると、プライマリ・データベースとスタンバイ・データベースの両方からデータファイルを削除できます。この文を使用するには、STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定する必要があります。たとえば、プライマリ・サイトで表領域を削除するには、次のように入力します。

```
SQL> DROP TABLESPACE INCLUDING CONTENTS –
> AND DATAFILES tbs_4;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

9.3.3 フィジカル・スタンバイ・データベースでのトランスポータブル表領域の使用

Oracle トランスポータブル表領域機能を使用すると、Oracle データベースのサブセットを移動して、別の Oracle データベースにプラグインできます。これにより、実際には表領域がデータベース間で移動します。

フィジカル・スタンバイの使用中に表領域セットをプライマリ・データベースに移動またはコピーする手順は、次のとおりです。

1. トランスポートする表領域セットのデータファイルとその表領域セットの構造情報を含むエクスポート・ファイルで構成される、トランスポータブル表領域セットを生成します。
2. 次の手順で表領域セットをトランスポートします。
   a. データファイルとエクスポート・ファイルをプライマリ・データベースにコピーします。
   b. データファイルをスタンバイ・データベースにコピーします。

   データファイルは、DB_FILE_NAME_CONVERT 初期化パラメータで定義したディレクトリにコピーする必要があります。DB_FILE_NAME_CONVERT が定義されていない場合は、トランスポータブル表領域を含む REDO データが適用されて失敗した後に、ALTER DATABASE RENAME FILE 文を発行してスタンバイ制御ファイルを変更します。STANDBY_FILE_MANAGEMENT 初期化パラメータは AUTO に設定する必要があります。

3. 表領域をプラグインします。

   データ・ポンプ・ユーティリティを起動して、表領域セットをプライマリ・データベースにプラグインします。スタンバイ・サイトで REDO データが生成されて適用され、表領域がスタンバイ・データベースにプラグインされます。

トランスポータブル表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。

9.3.4 プライマリ・データベースのデータファイルの改名

プライマリ・データベースで 1 つ以上のデータファイルを改名した場合、その変更はスタンバイ・データベースに伝播されません。STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定しても変更処理は自動的に実行されないため、スタンバイ・データベースにある同じデータファイルを改名する場合は、スタンバイ・データベースで同じ変更を手動で行う必要があります。

次の手順では、プライマリ・データベースでデータファイルを改名し、その変更をスタンバイ・データベースに手動で伝播する方法について説明します。

1. プライマリ・データベースでデータファイルを改名するには、表領域をオフラインにします。

   ```
   SQL> ALTER TABLESPACE tbs_4 OFFLINE;
   ```

2. SQL プロンプトを終了し、UNIX の mv コマンドなどのオペレーティング・システム・コマンドを発行して、プライマリ・システム上のデータファイルを改名します。

   ```
   % mv /disk1/oracle/oradata/payroll/tbs_4.dbf
   /disk1/oracle/oradata/payroll/tbs_x.dbf
   ```

3. プライマリ・データベース内のデータファイルを改名し、表領域をオンラインに戻します。

   ```
   SQL> ALTER TABLESPACE tbs_4 RENAME DATAFILE
     2> '/disk1/oracle/oradata/payroll/tbs_4.dbf'
     3>  TO '/disk1/oracle/oradata/payroll/tbs_x.dbf';
   SQL> ALTER TABLESPACE tbs_4 ONLINE;
   ```

4. スタンバイ・データベースに接続し、REDO Apply を停止します。

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   ```

5. スタンバイ・データベースを停止します。

   ```
   SQL> SHUTDOWN;
   ```

6. UNIX の mv コマンドなどのオペレーティング・システム・コマンドを使用して、スタンバイ・サイトでデータファイルを改名します。

   ```
   % mv /disk1/oracle/oradata/payroll/tbs_4.dbf /disk1/oracle/oradata/payroll/
   tbs_x.dbf
   ```

7. スタンバイ・データベースを起動し、マウントします。

   ```
   SQL> STARTUP MOUNT;
   ```

8. スタンバイ制御ファイルのデータファイルを改名します。STANDBY_FILE_MANAGEMENT 初期化パラメータは MANUAL に設定する必要があります。

   ```
   SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/tbs_4.dbf'
     2> TO '/disk1/oracle/oradata/payroll/tbs_x.dbf';
   ```

9. スタンバイ・データベースで、REDO Apply を再開します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
  2> DISCONNECT FROM SESSION;
```

スタンバイ・システムで対応するデータファイルを改名しないでスタンバイ・データベース制御ファイルをリフレッシュしようとすると、スタンバイ・データベースは改名されたデータファイルの使用を試みますが、改名されたデータファイルは見つかりません。したがって、アラート・ログに次のようなエラー・メッセージが表示されます。

```
ORA-00283: recovery session canceled due to errors
ORA-01157: cannot identify/lock datafile 4 - see DBWR trace file
ORA-01110: datafile 4: '/Disk1/oracle/oradata/payroll/tbs_x.dbf'
```

9.3.5 REDO ログ・ファイル・グループの追加または削除

プライマリ・データベースで REDO ログ・ファイル・グループを追加または削除した後に、フィジカル・スタンバイ・データベースの REDO ログおよびスタンバイ REDO ログの構成を再評価し、必要に応じて調整する必要があります。

次の手順を実行して、フィジカル・スタンバイ・データベースで REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループを追加または削除します。

1. REDO Apply を停止します。
2. STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定している場合は、値を MANUAL に変更します。
3. REDO ログ・ファイル・グループを追加または削除します。
4. STANDBY_FILE_MANAGEMENT 初期化パラメータおよび REDO Apply オプションを元の状態にリストアします。
5. REDO Apply を再開する。

9.3.6 ログに記録されていないまたはリカバリ不能な操作

NOLOGGING または UNRECOVERABLE 句を使用して DML または DDL 操作を実行するとスタンバイ・データベースは無効になるため、修正のために多大な DBA 管理アクティビティが必要になる場合があります。SQL ALTER DATABASE または SQL ALTER TABLESPACE 文で FORCELOGGING 句を指定すると、NOLOGGING 設定を上書きできます。ただし、この文はすでに無効になっているデータベースを修正しません。

NOLOGGING 句を使用した後のリカバリの詳細は、13.4 項を参照してください。

9.3.7 パスワード・ファイルのリフレッシュ

REMOTE_LOGIN_PASSWORDFILE データベース初期化パラメータが SHARED または EXCLUSIVE に設定されている場合、管理権限を付与または取り消すか、あるいは管理権限を持つユーザーのパスワードを変更した後に、フィジカル・スタンバイ・データベースのパスワード・ファイルをプライマリ・データベースの最新コピーで置き換える必要があります。

フィジカル・スタンバイ・データベースのパスワード・ファイルをリフレッシュできないと、REDO 転送セッションの認証や SYSDBA または SYSOPER としてのフィジカル・スタンバイ・データベースへの接続が失敗する可能性があります。

9.3.8 TDE マスター暗号化キーのリセット

フィジカル・スタンバイ・データベースのデータベース暗号化ウォレットは、プライマリ・データベースで TDE マスター暗号化キーがリセットされるたびに、プライマリ・データベースのデータベース暗号化ウォレットの最新コピーで置き換える必要があります。

フィジカル・スタンバイ・データベースのデータベース暗号化ウォレットをリフレッシュできないと、プライマリ・データベースでマスター暗号化キーがリセットされた後に変更された、フィジカル・スタンバイ・データベース上の暗号化された列にアクセスできません。

9.4 OPEN RESETLOGS 文を使用したリカバリ

Data Guard では、RESETLOGS オプションを使用してプライマリ・データベースがオープンされた後も、フィジカル・スタンバイ・データベースでリカバリを続行できます。プライマリ・データベースで ALTER DATABASE OPEN RESETLOGS 文を発行すると、データベースのインカネーションが変更され、REDO データの新規ブランチが作成されます。

フィジカル・スタンバイ・データベースが REDO データの新規ブランチを受信すると、REDO Apply ではそれが自動的に使用されます。フィジカル・スタンバイ・データベースの場合、スタンバイ・データベースにより新規リセットログの SCN より後（REDO データの新規ブランチの開始より後）の REDO データが適用されていなければ、手動による介入は必要ありません。次の表に、スタンバイ・データベースとプライマリ・データベースのブランチを再同期化する方法を示します。

スタンバイ・データベースの状態	操作	実行する手順
新規リセットログの SCN の後（REDO データの新規ブランチの開始より後）の REDO データが適用されていない場合	REDO Apply は自動的に REDO の新規ブランチを使用します。	手動による介入は必要ありません。MRP は、自動的にスタンバイ・データベースを REDO データの新規ブランチと再同期化します。
新規リセットログの SCN の後（REDO データの新規ブランチの開始より後）の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっている場合	スタンバイ・データベースは、REDO データの将来の新規ブランチでリカバリされます。	1. 13.3.1 項の手順に従ってフィジカル・スタンバイ・データベースをフラッシュ・バックします。 2. REDO Apply を再開し、新規リセットログ・ブランチへの REDO データの適用を続行します。 MRP は、自動的にスタンバイ・データベースを新規ブランチと再同期化します。
新規リセットログの SCN の後（REDO データの新規ブランチの開始より後）の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっていない場合	指定のプライマリ・データベース・ブランチで、プライマリ・データベースとスタンバイの内容にずれが生じます。	第 3 章の手順に従ってフィジカル・スタンバイ・データベースを再作成します。
REDO データの新規ブランチから介入するアーカイブ REDO ログ・ファイルが欠落している場合	MRP は欠落しているログ・ファイルが取得されるまで続行できません。	各ブランチから欠落しているアーカイブ REDO ログ・ファイルを検索して登録します。
REDO データの前のブランチの終わりからアーカイブ REDO ログ・ファイルが欠落している場合	MRP は欠落しているログ・ファイルが取得されるまで続行できません。	前のブランチから欠落しているアーカイブ REDO ログ・ファイルを検索して登録します。

データベース・インカネーション、OPEN RESETLOGS 操作を使用したリカバリおよびフラッシュバック・データベースの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

9.5 プライマリ、フィジカル・スタンバイおよびスナップショット・スタンバイ・データベースの監視

この項では、プライマリ・データベースおよびスタンバイ・データベースを監視するために役立つ情報の検索方法について説明します。

表 9-2 に、一般的なプライマリ・データベースの管理アクションと、そのアクションに関する情報の参照先を示します。

表 9-2 一般的なプライマリ・データベースの管理アクションに関する情報ソース

プライマリ・データベースのアクション	プライマリ・サイト情報	スタンバイ・サイト情報
REDO スレッドの有効化または無効化	■ アラート・ログ ■ V$THREAD	アラート・ログ
データベース・ロール、保護モード、保護レベル、スイッチオーバー・ステータス、ファスト・スタート・フェイルオーバー情報などの表示	V$DATABASE	V$DATABASE
REDO ログ・ファイル・グループの追加または削除	■ アラート・ログ ■ V$LOG ■ V$LOGFILE の STATUS 列	アラート・ログ
CREATE CONTROLFILE	アラート・ログ	アラート・ログ
REDO Apply の監視	■ アラート・ログ ■ V$ARCHIVE_DEST_STATUS	■ アラート・ログ ■ V$ARCHIVED_LOG ■ V$LOG_HISTORY ■ V$MANAGED_STANDBY
表領域ステータスの変更	■ V$RECOVER_FILE ■ DBA_TABLESPACES ■ アラート・ログ	■ V$RECOVER_FILE ■ DBA_TABLESPACES
データファイルまたは表領域の追加または削除	■ DBA_DATA_FILES ■ アラート・ログ	■ V$DATAFILE ■ アラート・ログ
データファイルの改名	■ V$DATAFILE ■ アラート・ログ	■ V$DATAFILE ■ アラート・ログ
ログに記録されていないまたはリカバリ不能な操作	■ V$DATAFILE ■ V$DATABASE	アラート・ログ
REDO 転送の監視	■ V$ARCHIVE_DEST_STATUS ■ V$ARCHIVED_LOG ■ V$ARCHIVE_DEST ■ アラート・ログ	■ V$ARCHIVED_LOG ■ アラート・ログ
OPEN RESETLOGS または CLEAR UNARCHIVED LOGFILES 文の発行	アラート・ログ	アラート・ログ
初期化パラメータの変更	アラート・ログ	アラート・ログ

9.5.1 プライマリ、フィジカルおよびスナップショット・スタンバイ・データベースを監視するためのビューの使用

この項では、動的パフォーマンス・ビューを使用してプライマリ・データベース、フィジカル・スタンバイ・データベースおよびスナップショット・スタンバイ・データベースを監視する方法について説明します。

次の動的パフォーマンス・ビューについて説明します。

- V$DATABASE
- V$MANAGED_STANDBY
- V$ARCHIVED_LOG
- V$LOG_HISTORY
- V$DATAGUARD_STATUS

> **関連項目：** 各ビューの詳細は、『Oracle Database リファレンス』を参照してください。

9.5.1.1 V$DATABASE

次の問合せは、プライマリ・データベース、フィジカル・スタンバイ・データベースまたはスナップショット・スタンバイ・データベースのデータ保護モード、データ保護レベル、データベース・ロールおよびスイッチオーバー・ステータスを表示します。

```
SQL> SELECT PROTECTION_MODE, PROTECTION_LEVEL, -
> DATABASE_ROLE ROLE, SWITCHOVER_STATUS -
> FROM V$DATABASE;
```

次の問合せは、ファスト・スタート・フェイルオーバーのステータスを表示します。

```
SQL> SELECT FS_FAILOVER_STATUS "FSFO STATUS", -
> FS_FAILOVER_CURRENT_TARGET TARGET, -
> FS_FAILOVER_THRESHOLD THRESHOLD, -
> FS_FAILOVER_OBSERVER_PRESENT "OBSERVER PRESENT" -
> FROM V$DATABASE;
```

9.5.1.2 V$MANAGED_STANDBY

次の問合せは、フィジカル・スタンバイ・データベースでの REDO Apply および REDO 転送ステータスを表示します。

```
SQL> SELECT PROCESS, STATUS, THREAD#, SEQUENCE#,-
> BLOCK#, BLOCKS FROM V$MANAGED_STANDBY;

PROCESS STATUS       THREAD#    SEQUENCE#  BLOCK#     BLOCKS
------- ------------ ---------- ---------- ---------- ----------
RFS     ATTACHED     1          947        72         72
MRP0    APPLYING_LOG 1          946        10         72
```

このサンプル出力は、RFS プロセスが REDO ログ・ファイル順序番号 947 のアーカイブを完了したことを示しています。また、REDO Apply がアーカイブ REDO ログ・ファイル順序番号 946 を適用していることも示しています。REDO Apply は現在、72 ブロックのアーカイブ REDO ログ・ファイルのブロック番号 10 をリカバリしている最中です。

9.5.1.3 V$ARCHIVED_LOG

次の問合せは、フィジカル・スタンバイ・データベースまたはスナップショット・スタンバイ・データベースでプライマリ・データベースから受信したアーカイブ REDO ログ・ファイルに関する情報を表示します。

```
SQL> SELECT THREAD#, SEQUENCE#, FIRST_CHANGE#, -
> NEXT_CHANGE# FROM V$ARCHIVED_LOG;

THREAD#    SEQUENCE#  FIRST_CHANGE# NEXT_CHANGE#
---------- ---------- ------------- ------------
1          945        74651         74739
1          946        74739         74772
1          947        74772         7474
```

このサンプル出力は、プライマリ・データベースから受信した 3 つのアーカイブ REDO ログ・ファイルを示しています。

9.5.1.4 V$LOG_HISTORY

次の問合せは、アーカイブ・ログの履歴情報を表示します。

```
SQL> SELECT THREAD#, SEQUENCE#, FIRST_CHANGE#, -
> NEXT_CHANGE# FROM V$LOG_HISTORY;
```

9.5.1.5 V$DATAGUARD_STATUS

次の問合せは、アラート・ログまたはサーバー・プロセス・トレース・ファイルにメッセージが書き込まれる原因となった Data Guard イベントによって生成されたメッセージを表示します。

```
SQL> SELECT MESSAGE FROM V$DATAGUARD_STATUS;
```

9.6 REDO Apply のチューニング

REDO Apply およびメディア・リカバリのパフォーマンスを最適化する方法は、『Oracle Data Guard Redo Apply and Media Recovery Best Practices』ホワイト・ペーパーを参照してください。このホワイト・ペーパーは、次の URL から Oracle Maximum Availability Architecture（MAA）のホームページにアクセスして入手することができます。

```
http://otn.oracle.com/deploy/availability/htdocs/maa.htm
```

関連項目：

Standby Statspack のインストールおよび使用方法の詳細は、`https://metalink.oracle.com` で、Oracle MetaLink Note 454848.1 を参照してください。Standby Statspack を使用すると、フィジカル・スタンバイ・データベースから REDO Apply のパフォーマンス・データを収集できます。

9.7 スナップショット・スタンバイ・データベースの管理

スナップショット・スタンバイ・データベースは、全面的に更新可能なスタンバイ・データベースで、フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースに変換して作成します。スナップショット・スタンバイ・データベースは、プライマリ・データベースから REDO データを受信およびアーカイブしますが、適用はしません。プライマリ・データベースから受信した REDO データは、スナップショット・スタンバイ・データベースへのローカル更新がすべて破棄された後、スナップショット・スタンバイ・データベースが変換されてフィジカル・スタンバイ・データベースに戻ると適用されます。

スナップショット・スタンバイ・データベースは、通常、時間とともにプライマリ・データベースとは異なってきます。これは、プライマリ・データベースからの REDO データが受信時に適用されないためです。スナップショット・スタンバイ・データベースに対するローカル更新が原因で、相違はさらに大きくなります。しかし、プライマリ・データベースのデータは完全に保護されます。これは、任意の時点でスナップショット・スタンバイを変換してフィジカル・スタンバイ・データベースに戻すことができ、その後にプライマリから受信した REDO データが適用されるためです。

スナップショット・スタンバイ・データベースには、フィジカル・スタンバイ・データベースと同様の、障害時リカバリおよびデータ保護のメリットがあります。スナップショット・スタンバイ・データベースは、プライマリ・データベースの一時的かつ更新可能なスナップショットを保持することにメリットがあれば、管理上の複雑さやプライマリ・データベースの障害からリカバリする時間が増えてもかまわない場合に最もよく使用されます。

9.7.1 スナップショット・スタンバイ・データベースへのフィジカル・スタンバイ・データベースの変換

次の手順を実行して、フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースに変換します。

1. REDO Apply がアクティブな場合は、停止します。
2. Oracle Real Application Clusters（RAC）データベースで、1 つのインスタンス以外はすべて停止します。
3. データベースがマウントされ、オープンしていないことを確認します。
4. 次の SQL 文を発行して変換を実行します。

```
SQL> ALTER DATABASE CONVERT TO SNAPSHOT STANDBY;
```

データベースは変換後にディスマウントされ、再起動する必要があります。

注意： Data Guard Broker で管理されるフィジカル・スタンバイ・データベースは、DGMGRL または Oracle Enterprise Manager のいずれかを使用してスナップショット・スタンバイ・データベースに変換できます。詳細は、『Oracle Data Guard Broker』を参照してください。

9.7.2 スナップショット・スタンバイ・データベースの使用

スナップショット・スタンバイ・データベースは読取り / 書込みモードでオープンされ、全体を更新できます。

スナップショット・スタンバイ・データベースには、次の特性があります。

- スナップショット・スタンバイ・データベースは、スイッチオーバーまたはフェイルオーバーのターゲットにはできません。スナップショット・スタンバイ・データベースは、ロールの推移を実行する前にフィジカル・スタンバイ・データベースに変換して戻す必要があります。
- スナップショット・スタンバイ・データベースは、最大保護の Data Guard 構成では唯一のスタンバイ・データベースになりません。

注意： フラッシュバック・データベースを使用して、スナップショット・スタンバイ・データベースを変換してフィジカル・スタンバイ・データベースに戻すことができます。フラッシュバック・データベース・テクノロジを使用して元に戻すことができない操作は、スナップショット・スタンバイを変換してフィジカル・スタンバイに戻すことを妨げます。

9.7.3 フィジカル・スタンバイ・データベースへのスナップショット・スタンバイ・データベースの変換

次の手順を実行して、スナップショット・スタンバイ・データベースをフィジカル・スタンバイ・データベースに変換します。

1. Oracle Real Application Clusters（RAC）データベースで、1 つのインスタンス以外はすべて停止します。
2. データベースがマウントされ、オープンしていないことを確認します。
3. 次の SQL 文を発行して変換を実行します。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

データベースは変換後にディスマウントされ、再起動する必要があります。

データベースがスナップショット・スタンバイ・データベースの間に受信した REDO データは、REDO Apply が開始されると自動的に適用されます。

注意： スナップショット・スタンバイ・データベースは、少なくとも 1 回は読取り / 書込みモードでオープンしてから、フィジカル・スタンバイ・データベースに変換する必要があります。

10

ロジカル・スタンバイ・データベースの管理

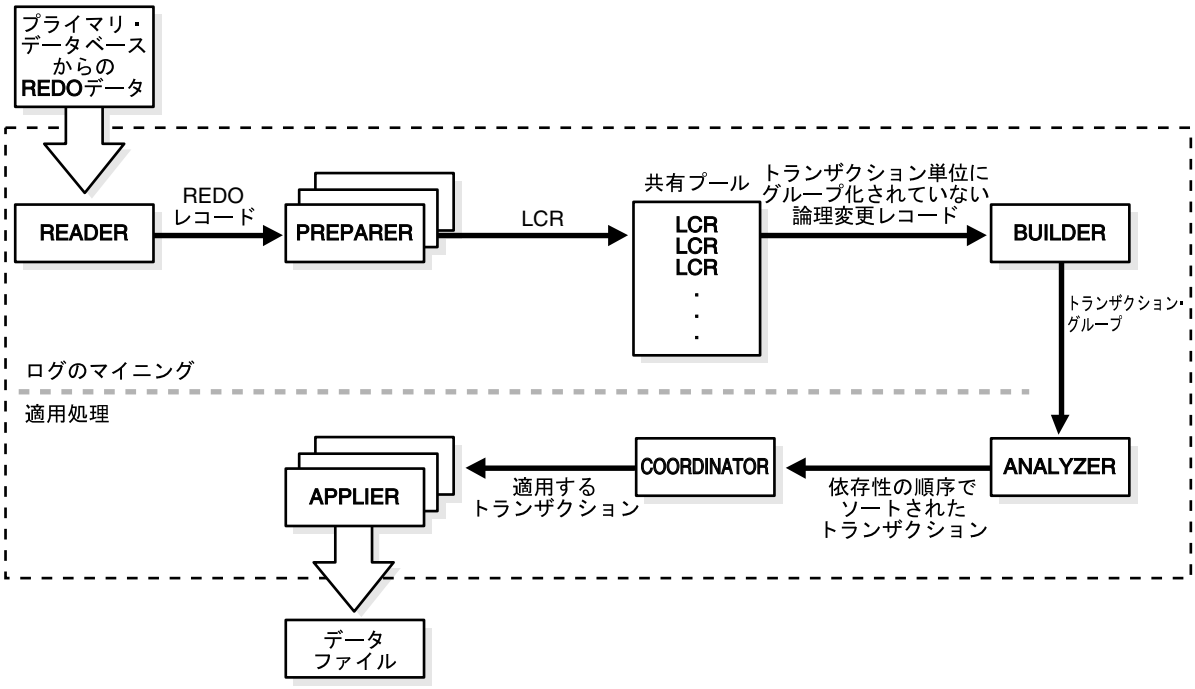
この章は、次の項目で構成されています。

10.1 SQL Apply アーキテクチャの概要

SQL Apply では、バックグラウンド・プロセスの集合を使用して、プライマリ・データベースの変更がロジカル・スタンバイ・データベースに適用されます。

図 10-1 に、情報の流れと各プロセスで実行されるロールを示します。

図 10-1 SQL Apply の処理



ログのマイニングおよび適用処理中は、次のように、様々なプロセスとその機能が関係します。

ログのマイニング中には、次のプロセスが実行されます。

- READER プロセスでは、REDO レコードがアーカイブ REDO ログ・ファイルまたはスタンバイ REDO ログ・ファイルから読み込まれます。
- PREPARER プロセスでは、REDO レコードに含まれるブロックの変更が論理変更レコード（LCR）に変換されます。特定の REDO ログ・ファイルに対して複数の PREPARER プロセスをアクティブにすることができます。LCR は、LCR キャッシュと呼ばれるシステム・グローバル領域（SGA）でステージングされます。
- BUILDER プロセスでは、LCR がトランザクション単位にグループ化されて、LCR キャッシュ内のメモリー管理、SQL Apply の再起動に関連するチェックポイント処理および重要でない変更のフィルタリングなど、その他のタスクが実行されます。

適用処理中には、次のプロセスが実行されます。

- ANALYZER プロセスでは、様々なトランザクション間の依存性が識別されます。
- COORDINATOR プロセス（LSP）では、トランザクションを様々な APPLIER に割り当て、トランザクション間の依存性が考慮されるように調整します。
- APPLIER プロセスでは、COORDINATOR プロセスの管理下でロジカル・スタンバイ・データベースにトランザクションを適用します。

V$LOGSTDBY_PROCESS ビューを問い合せると、SQL Apply プロセスのアクティビティを検査できます。現行のアクティビティに関する情報は、V$LOGSTDBY_STATS ビューでも提供されます。このビューには、SQL Apply アクティビティ中のロジカル・スタンバイ・データベースに関する統計、現在の状態およびステータス情報が表示されます。この 2 つのビューと他の関連ビューの詳細は、10.3 項「ロジカル・スタンバイ・データベースの管理および監視関連のビュー」を参照してください。

注意： SQL Apply プロセス（COORDINATOR プロセス lsp0 を含む）はすべてバックグラウンド・プロセスです。リソース・マネージャによる調整は行われません。そのため、ロジカル・スタンバイ・データベースでリソース・グループを作成しても、SQL Apply プロセスには影響ありません。

10.1.1 SQL Apply に関する各種の考慮事項

この項は、次の項目で構成されています。

- トランザクション・サイズの考慮事項
- ページアウトの考慮事項
- 再開の考慮事項
- DML 適用の考慮事項
- DDL 適用の考慮事項
- パスワード検証関数

10.1.1.1 トランザクション・サイズの考慮事項

SQL Apply では、トランザクションが大小 2 つのクラスに分類されます。

- 小さいトランザクション : SQL Apply は、REDO ログ・ファイル内で小さいトランザクションのコミット・レコードを検出すると、そのトランザクションに属する LCR の適用を開始します。
- 大きいトランザクション : SQL Apply は、大きいトランザクションをトランザクション・チャンクと呼ばれる小さい単位に分割し、そのトランザクションのコミット・レコードを REDO ログ・ファイル内で検出する前に、チャンクの適用を開始します。この処理が実行されるのは、LCR キャッシュのメモリー使用量を削減し、フェイルオーバー時間全体を短縮するためです。

  たとえば、小さく分割しない場合、SQL*Loader のロードが 1000 万行で、サイズがそれぞれ 100 バイトであれば、LCR キャッシュでは 1GB を超えるメモリーが使用されます。LCR キャッシュに割り当てられているメモリーが 1GB 未満の場合は、LCR キャッシュからのページアウトが発生します。

  メモリーの考慮事項とは別に、SQL Apply がトランザクションの COMMIT レコードを検出するまでに 1000 万行の SQL*Loader ロードに関連する変更の適用を開始しない場合、ログの推移が停止します。SQL Apply がロジカル・スタンバイ・データベース上でトランザクションを適用し終わるまで、そのトランザクションのコミット後に開始されたスイッチオーバーまたはフェイルオーバーは終了できません。

  トランザクション・チャンクの使用にもかかわらず、100 万行超を変更するトランザクションの処理時に SQL Apply のパフォーマンスが低下することがあります。トランザクションのサイズを 100 万行未満に制限することをお薦めします。SQL*Loader 表のロードが大規模な場合は、ROWS 句を使用してトランザクション内にロードされる行数を制限します。

すべてのトランザクションは、最初は小さいトランザクションとして分類されます。SQL Apply では、トランザクションが大きいトランザクションとして再分類されるタイミングは、LCR キャッシュに使用可能なメモリー量と、トランザクションに属する LCR のメモリー使用量に応じて決定します。

10.1.1.2 ページアウトの考慮事項

LCR キャッシュのメモリーがすべて使用され、SQL Apply を進行させるために領域の解放が必要になると、SQL Apply の下でページアウトが発生します。

たとえば、LCR キャッシュに 100MB のメモリーが割り当てられている場合に、SQL Apply でサイズが 300MB の LONG 列を含む表に対する INSERT トランザクションが検出されるとします。この場合、ログ・マイニング・コンポーネントは、LONG データの最初の部分をページアウトして列変更の後半部分を読み取ります。適切にチューニングされているロジカル・スタンバイ・データベースの場合、ページアウト・アクティビティはほとんど発生せず、システム全体のスループットには影響しません。

> **関連項目：** 問題のあるページアウトを識別して修正処理を実行する方法の詳細は、10.5 項「ロジカル・スタンバイ・データベースのカスタマイズ」を参照してください。

10.1.1.3 再開の考慮事項

トランザクションのコミット・レコードが REDO ログ・ファイルからマイニングされてロジカル・スタンバイ・データベースに適用されるまで、ロジカル・スタンバイ・データベースに対する変更は永続的な変更になりません。つまり、ユーザー指示の結果であるかシステム障害の結果であるかに関係なく、SQL Apply は停止されるたびに、コミットされていない最も古いトランザクションまで遡って再びマイニングする必要があります。

トランザクションの処理量が小さくても長時間オープン状態になっている場合は、SQL Apply を最初から再開すると非常に高コストになります。これは、SQL Apply ではコミットされていない少数のトランザクションの REDO データを読み取るだけでなく、多数のアーカイブ REDO ログ・ファイルを再びマイニングする操作が必要になるためです。これを軽減するために、SQL Apply はコミットされていない古いデータのチェックポイントを定期的に実行します。チェックポイントが実行される SCN は、V$LOGSTDBY_PROGRESS ビューの RESTART_SCN 列に反映されます。

再開時には、SQL Apply は RESTART_SCN 列に表示される値より大きい SCN で生成される REDO レコードのマイニングを開始します。再開に不要なアーカイブ REDO ログ・ファイルは、SQL Apply により自動的に削除されます。

多数の DDL トランザクション、パラレル DML 文（PDML）およびダイレクト・パス・ロードなど、ある種のワークロードがあると、RESTART_SCN はワークロードの存続期間中は進行しなくなります。

10.1.1.4 DML 適用の考慮事項

SQL Apply がロジカル・スタンバイ・データベースのスループットと待機時間に影響する DML トランザクションを適用する場合、次の特性があります。

- 1 つの文で複数の行が変更されるバッチ更新または削除をプライマリ・データベースで実行した場合、ロジカル・スタンバイ・データベースでは行の個別変更として適用されます。そのため、メンテナンスされている各表に一意索引または主キーが必要になります。詳細は、4.1.2 項「プライマリ・データベース内の表の行が一意に識別できることの確認」を参照してください。
- プライマリ・データベースで実行されたダイレクト・パス・インサートは、ロジカル・スタンバイ・データベースで従来型の INSERT 文を使用して適用されます。
- パラレル DML（PDML）トランザクションは、ロジカル・スタンバイ・データベースではパラレルに実行されません。

10.1.1.5 DDL 適用の考慮事項

SQL Apply がロジカル・スタンバイ・データベースのスループットと待機時間に影響する DDL トランザクションを適用する場合、次の特性があります。

- DDL トランザクションは、ロジカル・スタンバイ・データベースではシリアルに適用されます。そのため、プライマリ・データベースで同時に適用された DDL トランザクションは、ロジカル・スタンバイ・データベースでは一度に 1 つずつ適用されます。
- ロジカル・スタンバイ・データベース上で DML アクティビティ（CREATE TABLE AS SELECT（CTAS）文の一部）が抑制されるように、CTAS 文が実行されます。CTAS 文の一部として新規作成された表に挿入された行は、REDO ログ・ファイルからマイニングされ、INSERT 文を使用してロジカル・スタンバイ・データベースに適用されます。
- SQL Apply は、プライマリ・データベースで実行された DDL を再発行し、DDL 操作のターゲットである同じオブジェクトで、同じトランザクション内で発生する DML がロジカル・スタンバイ・データベースでレプリケートされないようにします。そのため、次の 2 つの場合にプライマリ・サイトとスタンバイ・サイトは互いに異なります。
  - DDL に、プライマリ・データベースの状態から導出される非リテラル値が含まれる場合。このような DDL の例を次に示します。

    ```
    ALTER TABLE hr.employees ADD (start_date date default sysdate);
    ```

    SQL Apply はロジカル・スタンバイで同じ DDL を再発行するため、sysdate() ファンクションはロジカル・スタンバイで再評価されます。そのため、start_date 列は、プライマリ・データベースでのデフォルト値とは異なるデフォルト値で作成されます。
  - DDL で、ターゲット表で定義されている DML トリガーを起動する場合。トリガーされる DML は DDL と同じトランザクションで発生し、DDL のターゲット表で動作するため、これらのトリガーされる DML は、ロジカル・スタンバイでレプリケートされません。

    たとえば、次のような表を作成するとします。

    ```
    create table HR.TEMP_EMPLOYEES (
    emp_id       number primary key,
    first_name   varchar2(64),
    last_name    varchar2(64),
    modify_date  timestamp);
    ```

    次に、表にトリガーを作成して、表が更新されるたびに、modify_date を更新して変更時間が反映されるようにします。

    ```
    CREATE OR REPLACE TRIGGER TRG_TEST_MOD_DT  BEFORE UPDATE ON HR.TEST_EMPLOYEES
    REFERENCING
    NEW  AS NEW_ROW  FOR EACH ROW
    BEGIN
    :NEW_ROW.MODIFY_DATE:= SYSTIMESTAMP;
    END;
    /
    ```

    この表は、通常の DML/DDL ワークロードで正しくメンテナンスされます。しかし、デフォルト値の列を表に追加すると、ADD COLUMN DDL はこの更新トリガーを起動し、表のすべての行の MODIFY_DATE 列を新しいタイムスタンプに変更します。MODIFY_DATE 列へのこのような変更は、ロジカル・スタンバイ・データベースでレプリケートされません。表に対する後続の DML では SQL Apply が停止します。これは、REDO ストリームに記録された MODIFY_DATE 列のデータがロジカル・スタンバイ。データベースに存在するデータと一致しないためです。

10.1.1.6 パスワード検証関数

パスワードの複雑性をチェックするパスワード検証関数は、SYS スキーマで作成する必要があります。SQL Apply は SYS スキーマで作成されたオブジェクトをレプリケートしないため、このような検証関数は、ロジカル・スタンバイ・データベースにレプリケートされません。ロジカル・スタンバイ・データベースでは、パスワード検証関数を手動で作成し、適切なプロファイルと関連付ける必要があります。

10.2 ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスの制御

SQL の ALTER DATABASE GUARD 文は、ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスを制御します。デフォルトでは、ロジカル・スタンバイ・データベースについてはデータベース・ガードが ALL に設定されています。

ALTER DATABASE GUARD 文では、次のキーワードを使用できます。

- ALL

  ALL を指定すると、SYS 以外のすべてのユーザーが、ロジカル・スタンバイ・データベース内のデータを変更できなくなります。

- STANDBY

  STANDBY を指定すると、SYS 以外のすべてのユーザーが、SQL Apply でメンテナンスされる表または順序に対して DML および DDL を変更できなくなります。

- NONE

  NONE を指定すると、データベース内の全データには通常のセキュリティが適用されます。

たとえば、SQL Apply によってメンテナンスされない表をユーザーが変更できるようにするには、次の文を使用します。

```
SQL> ALTER DATABASE GUARD STANDBY;
```

権限を持つユーザーは、ALTER SESSION DISABLE GUARD および ALTER SESSION ENABLE GUARD 文をそれぞれ使用して、現在のセッションでデータベース・ガードを一時的にオフまたはオンにできます。この文は、Oracle9*i* で同様の機能を行っていた DBMS_LOGSTDBY.GUARD_BYPASS PL/SQL プロシージャにかわるものです。ALTER SESSION [ENABLE|DISABLE] GUARD 文は、10.5.4 項に説明されているように、データベース・ガードを一時的に無効にしてデータベースを変更するときに役立ちます。

注意： データベース・ガードが無効のときに、プライマリおよびロジカル・スタンバイ・データベースの内容に相違を生じさせないようにしてください。

10.3 ロジカル・スタンバイ・データベースの管理および監視関連のビュー

次のパフォーマンス・ビューでは、ロジカル・スタンバイ・データベースを保守する SQL Apply の動作が監視されます。次の各項では、ロジカル・スタンバイ・データベースの監視に使用できる主なビューについて説明します。

- DBA_LOGSTDBY_EVENTS ビュー
- DBA_LOGSTDBY_LOG ビュー
- V$DATAGUARD_STATS ビュー
- V$LOGSTDBY_PROCESS ビュー
- V$LOGSTDBY_PROGRESS ビュー
- V$LOGSTDBY_STATE ビュー
- V$LOGSTDBY_STATS ビュー

関連項目： 各ビューの詳細は、『Oracle Database リファレンス』を参照してください。

10.3.1 DBA_LOGSTDBY_EVENTS ビュー

DBA_LOGSTDBY_EVENTS ビューには、SQL Apply 操作中に発生した重要なイベントが記録されます。デフォルトでは、最新の 10,000 のイベントが記録されます。ただし、記録されるイベントの数は、PL/SQL プロシージャ DBMS_LOGSTDBY.APPLY_SET() をコールして変更できます。SQL Apply が突然停止した場合は、このビューにその原因も記録されます。

注意： SQL Apply を停止させるエラーは、イベント表に記録されます。この種のイベントは ALERT.LOG ファイルにも記録され、テキストに LOGSTDBY キーワードが挿入されます。このビューを問い合せるときは、EVENT_TIME_STAMP、COMMIT_SCN、CURRENT_SCN の順に列を選択して、イベントが必要な順序になるようにします。

このビューは、適用された DDL トランザクションやスキップされた DDL トランザクションなどの他の情報も表示されるようにカスタマイズできます。次に例を示します。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT  = 'DD-MON-YY HH24:MI:SS';
Session altered.
SQL> COLUMN STATUS FORMAT A60
SQL> SELECT EVENT_TIME, STATUS, EVENT FROM DBA_LOGSTDBY_EVENTS
  2  ORDER BY EVENT_TIMESTAMP, COMMIT_SCN, CURRENT_SCN;

EVENT_TIME         STATUS
------------------------------------------------------------------------------
EVENT
------------------------------------------------------------------------------
23-JUL-02 18:20:12 ORA-16111: log mining and apply setting up
23-JUL-02 18:25:12 ORA-16128: User initiated shut down successfully completed
23-JUL-02 18:27:12 ORA-16112: log mining and apply stopping
23-JUL-02 18:55:12 ORA-16128: User initiated shut down successfully completed
23-JUL-02 18:57:09 ORA-16111: log mining and apply setting up
23-JUL-02 20:21:47 ORA-16204: DDL successfully applied
create table hr.test_emp (empno number, ename varchar2(64))
23-JUL-02 20:22:55 ORA-16205: DDL skipped due to skip setting
create database link link_to_boston connect to system identified by change_on_inst
7 rows selected.
```

この問合せは、SQL Apply の開始と停止が数回繰り返されたことを示しています。適用された DDL とスキップされた DDL も示しています。

10.3.2 DBA_LOGSTDBY_LOG ビュー

DBA_LOGSTDBY_LOG ビューには、SQL Apply で処理中のアーカイブ・ログに関する動的な情報が表示されます。

次に例を示します。

```
SQL> COLUMN DICT_BEGIN FORMAT A10;
SQL> SET NUMF 99999999
SQL> SELECT FILE_NAME, SEQUENCE# AS SEQ#, FIRST_CHANGE# AS F_SCN#, -
     NEXT_CHANGE# AS N_SCN#, TIMESTAMP, -
     DICT_BEGIN AS BEG, DICT_END AS END, -
     THREAD# AS THR#, APPLIED FROM DBA_LOGSTDBY_LOG -
     ORDER BY SEQUENCE#;
FILE_NAME                 SEQ# F_SCN    N_SCN TIMESTAM BEG END THR# APPLIED
------------------------- ---- ------- ------- -------- --- --- --- ---------
/oracle/dbs/hq_nyc_2.log  2     101579  101588 11:02:58 NO  NO  1     YES
/oracle/dbs/hq_nyc_3.log  3     101588  142065 11:02:02 NO  NO  1     YES
/oracle/dbs/hq_nyc_4.log  4     142065  142307 11:02:10 NO  NO  1     YES
/oracle/dbs/hq_nyc_5.log  5     142307  142739 11:02:48 YES YES 1     YES
/oracle/dbs/hq_nyc_6.log  6     142739  143973 12:02:10 NO  NO  1     YES
/oracle/dbs/hq_nyc_7.log  7     143973  144042 01:02:11 NO  NO  1     YES
/oracle/dbs/hq_nyc_8.log  8     144042  144051 01:02:01 NO  NO  1     YES
/oracle/dbs/hq_nyc_9.log  9     144051  144054 01:02:16 NO  NO  1     YES
/oracle/dbs/hq_nyc_10.log 10    144054  144057 01:02:21 NO  NO  1     YES
/oracle/dbs/hq_nyc_11.log 11    144057  144060 01:02:26 NO  NO  1  CURRENT
/oracle/dbs/hq_nyc_12.log 12    144060  144089 01:02:30 NO  NO  1  CURRENT
/oracle/dbs/hq_nyc_13.log 13    144089  144147 01:02:41 NO  NO  1       NO
```

BEG および END 列のエントリ YES は、LogMiner ディクショナリの作成がログ・ファイルの順序番号 5 で開始したことを示しています。最新のアーカイブ REDO ログ・ファイルは順序番号 13 で、ロジカル・スタンバイ・データベースでは 1 時 2 分 41 秒に受信されています。

APPLIED 列は、SQL Apply により SCN 144057 までの REDO がすべて適用されたことを示しています。トランザクションでは複数のアーカイブ・ログ・ファイルを使用している可能性があるため、複数のアーカイブ・ログ・ファイルの APPLIED 列に値 CURRENT が表示されることがあります。

10.3.3 V$DATAGUARD_STATS ビュー

このビューには、ロジカル・スタンバイ・データベースのフェイルオーバー特性に関連する次のような情報が表示されます。

- フェイルオーバー時間（apply finish time）
- ロジカル・スタンバイ・データベース内のコミット済データの最新性（apply lag）
- 障害時の潜在的なデータ損失（transport lag）

次に例を示します。

```
SQL> COL NAME FORMAT A20
SQL> COL VALUE FORMAT A12
SQL> COL UNIT FORMAT A30
SQL> SELECT NAME, VALUE, UNIT FROM V$DATAGUARD_STATS;

NAME                 VALUE        UNIT
-------------------- ------------ ------------------------------
apply finish time    +00 00:00:00 day(2) to second(1) interval
apply lag            +00 00:00:00 day(2) to second(0) interval
transport lag        +00 00:00:00 day(2) to second(0) interval
```

この出力は、プライマリ・データベースから生成されたすべての REDO を受信および適用したロジカル・スタンバイ・データベースからの出力です。

10.3.4 V$LOGSTDBY_PROCESS ビュー

このビューには、次のように、SQL Apply 関連の各種プロセスの現在の状態に関する情報が表示されます。

- 識別情報（sid | serial# | spid）
- SQL Apply プロセス：COORDINATOR、READER、BUILDER、PREPARER、ANALYZER または APPLIER（type）
- プロセスの現行のアクティビティのステータス（status_code | status）
- このプロセスで処理された最上位の REDO レコード（high_scn）

次に例を示します。

```
SQL> COLUMN SERIAL# FORMAT 9999
SQL> COLUMN SID FORMAT 9999
SQL> SELECT SID, SERIAL#, SPID, TYPE, HIGH_SCN FROM V$LOGSTDBY_PROCESS;

  SID   SERIAL#   SPID         TYPE            HIGH_SCN
 ----- -------   ----------- ---------------- ----------
   48        6    11074        COORDINATOR     7178242899
   56       56    10858        READER          7178243497
   46        1    10860        BUILDER         7178242901
   45        1    10862        PREPARER        7178243295
   37        1    10864        ANALYZER        7178242900
   36        1    10866        APPLIER         7178239467
   35        3    10868        APPLIER         7178239463
   34        7    10870        APPLIER         7178239461
   33        1    10872        APPLIER         7178239472

9 rows selected.
```

HIGH_SCN 列は、READER プロセスが他のすべてのプロセスに先行することと、PREPARER および BUILDER プロセスが残りのプロセスに先行することを示しています。

```
SQL> COLUMN STATUS FORMAT A40
SQL> SELECT TYPE, STATUS_CODE, STATUS FROM V$LOGSTDBY_PROCESS;

TYPE             STATUS_CODE STATUS
---------------- ----------- ----------------------------------------
COORDINATOR            16117 ORA-16117: processing
READER                 16127 ORA-16127: stalled waiting for additional
                             transactions to be applied
BUILDER                16116 ORA-16116: no work available
PREPARER               16116 ORA-16117: processing
ANALYZER               16120 ORA-16120: dependencies being computed for
                             transaction at SCN 0x0001.abdb440a
APPLIER                16124 ORA-16124: transaction 1 13 1427 is waiting
                             on another transaction
APPLIER                16121 ORA-16121: applying transaction with commit
                             SCN 0x0001.abdb4390
APPLIER                16123 ORA-16123: transaction 1 23  1231 is waiting
                             for commit approval
APPLIER                16116 ORA-16116: no work available
```

出力は、実行中の SQL Apply のスナップショットを示しています。マイニング側では、READER プロセスはさらに読み取れるように追加のメモリーが使用可能になるまで待機中で、PREPARER プロセスは REDO レコードの処理中、BUILDER プロセスに使用可能な作業はありません。適用側では、COORDINATOR は APPLIER プロセスにさらにトランザクションを割当中で、ANALYZER は SCN 7178241034 で依存性を計算中、APPLIER の 1 つは使用可能な作業がなく、2 つにはまだ満たされていない未処理の依存性があります。

関連項目： 出力例は 10.4.1 項「SQL Apply の進捗の監視」を参照してください。

10.3.5 V$LOGSTDBY_PROGRESS ビュー

このビューには、次のように、SQL Apply の進捗に関する詳細情報が表示されます。

- プライマリ・データベース上でコミットされた全トランザクションがロジカル・スタンバイ・データベースに適用された SCN および時刻（applied_scn, applied_time）
- 再開時に SQL Apply による REDO レコードの読取りが開始される SCN および時刻（restart_scn, restart_time）
- ロジカル・スタンバイ・データベースで受信された最新 REDO レコードの SCN および時刻（latest_scn, latest_time）
- BUILDER プロセスにより処理された最新レコードの SCN および時刻（mining_scn, mining_time）

次に例を示します。

```
SQL> SELECT APPLIED_SCN, LATEST_SCN, MINING_SCN, RESTART_SCN FROM V$LOGSTDBY_PROGRESS;

APPLIED_SCN  LATEST_SCN MINING_SCN RESTART_SCN
----------- ----------- ---------- -----------
 7178240496  7178240507 7178240507  7178219805
```

この出力は次のことを示しています。

- SQL Apply により、SCN 7178240496 以前にコミットされた全トランザクションが適用されました。
- ロジカル・スタンバイ・データベースで受信された最新 REDO レコードは、SCN 7178240507 で生成されました。
- マイニング・コンポーネントにより、SCN 7178240507 以前に生成された全 REDO レコードが処理されました。
- SQL Apply がなんらかの理由で停止されて再開されると、SCN 7178219805 以降に生成された REDO レコードのマイニングが開始されます。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='yy-mm-dd hh24:mi:ss';
Session altered

SQL> SELECT APPLIED_TIME, LATEST_TIME, MINING_TIME, RESTART_TIME FROM
V$LOGSTDBY_PROGRESS;

APPLIED_TIME      LATEST_TIME       MINING_TIME       RESTART_TIME
----------------- ----------------- ----------------- -----------------
05-05-12 10:38:21 05-05-12 10:41:53 05-05-12 10:41:21 05-05-12 10:09:30
```

この出力は次のことを示しています。

- SQL Apply により、時刻 05-05-12 10:38:21 以前にコミットされた全トランザクションが適用されました（APPLIED_TIME）。
- 最後の REDO は、プライマリ・データベースで時刻 05-05-12 10:41:53 に生成されました（LATEST_TIME）。
- マイニング・エンジンにより、05-05-12 10:41:21 以前に生成された全 REDO レコードが処理されました（MINING_TIME）。
- SQL Apply の再開時には、05-05-12 10:09:30 以後に生成された REDO レコードのマイニングが開始されます。

関連項目： 出力例は 10.4.1 項「SQL Apply の進捗の監視」を参照してください。

10.3.6 V$LOGSTDBY_STATE ビュー

このビューには、次のように、SQL Apply の現在の状態の概要が表示されます。

- プライマリ・データベースの DBID（primary_dbid）
- SQL Apply に割り当てられている LogMiner セッション ID（session_id）
- SQL Apply がリアルタイムで適用中かどうか（realtime_apply）

次に例を示します。

```
SQL> COLUMN REALTIME_APPLY FORMAT a15
SQL> COLUMN STATE FORMAT a16
SQL> SELECT * FROM V$LOGSTDBY_STATE;

PRIMARY_DBID SESSION_ID REALTIME_APPLY  STATE
------------ ---------- --------------- ----------------
  1562626987          1 Y               APPLYING
```

この出力は、SQL Apply がリアルタイム適用モードで実行中で、現在はプライマリ・データベースから受信した REDO データを適用中であり、プライマリ・データベースの DBID が 1562626987 で、SQL Apply セッションに関連付けられている LogMiner セッション ID が 1 であることを示しています。

> **関連項目：** 出力例は 10.4.1 項「SQL Apply の進捗の監視」を参照してください。

10.3.7 V$LOGSTDBY_STATS ビュー

V$LOGSTDBY_STATS ビューには、SQL Apply に関連する統計、現行の状態およびステータス情報が表示されます。SQL Apply が実行されていない場合、このビューからは 1 行も戻されません。このビューは、ロジカル・スタンバイ・データベースの下でのみ意味があります。

次に例を示します。

```
SQL> ALTER SESSION SET_NLS_DATE_FORMAT='dd-mm-yyyy hh24:mi:ss';
Session altered

SQL> SELECT SUBSTR(name, 1, 40) AS NAME, SUBSTR(value,1,32) AS VALUE FROM V$LOGSTDBY_STATS;

NAME                                     VALUE
---------------------------------------- --------------------------------
logminer session id                      1
number of preparers                      1
number of appliers                       5
server processes in use                  9
maximum SGA for LCR cache (MB)           30
maximum events recorded                  10000
preserve commit order                    TRUE
transaction consistency                  FULL
record skipped errors                    Y
record skipped DDLs                      Y
record applied DDLs                      N
record unsupported operations            N
realtime apply                           Y
apply delay (minutes)                    0
coordinator state                        APPLYING
coordinator startup time                 19-06-2007 09:55:47
coordinator uptime (seconds)             3593
txns received from logminer              56
txns assigned to apply                   23
txns applied                             22
txns discarded during restart            33
large txns waiting to be assigned        2
rolled back txns mined                   4
```

```
DDL txns mined                          40
CTAS txns mined                         0
bytes of redo mined                     60164040
bytes paged out                         0
pageout time (seconds)                  0
bytes checkpointed                      4845
checkpoint time (seconds)               0
system idle time (seconds)              2921
standby redo logs mined                 0
archived logs mined                     5
gap fetched logs mined                  0
standby redo log reuse detected         1
logfile open failures                   0
current logfile wait (seconds)          0
total logfile wait (seconds)            2910
thread enable mined                     0
thread disable mined                    0
.
40 rows selected.
```

10.4 ロジカル・スタンバイ・データベースの監視

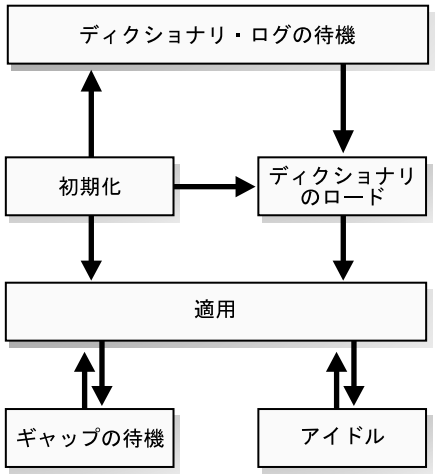
この項は、次の項目で構成されています。

- SQL Apply の進捗の監視
- ログ・ファイルの自動削除

10.4.1 SQL Apply の進捗の監視

SQL Apply には、6 つの進捗状況があります。SQL Apply の初期化、ディクショナリ・ログの待機、LogMiner ディクショナリのロード、(REDO データの) 適用、アーカイブ・ギャップの解決の待機およびアイドルです。図 10-2 に、これらの状態の流れを示します。

図 10-2 SQL Apply 処理中の進捗状態



次の各項では、それぞれの状態を詳細に説明します。

初期化中状態

ALTER DATABASE START LOGICAL STANDBY APPLY 文を発行して SQL Apply を開始すると、初期化中状態になります。

SQL Apply の現在の状態を判断するには、V$LOGSTDBY_STATE ビューを問い合せます。次に例を示します。

```
SQL> SELECT SESSION_ID, STATE FROM V$LOGSTDBY_STATE;

SESSION_ID    STATE
----------    -------------
1             INITIALIZING
```

SESSION_ID 列は、SQL Apply により作成された永続 LogMiner セッションを示します。このセッションで、プライマリ・データベースで生成されたアーカイブ REDO ログ・ファイルがマイニングされます。

ディクショナリ・ログ待機中

SQL Apply は、初回の開始時に、REDO ログ・ファイルで取得された LogMiner ディクショナリをロードする必要があります。SQL Apply は、LogMiner ディクショナリのロードに必要な REDO データをすべて受信するまで、WAITING FOR DICTIONARY LOGS 状態にとどまります。

ディクショナリ・ロード中状態

ディクショナリ・ロード中状態は、しばらく継続する場合があります。大規模データベースで LogMiner ディクショナリをロードするには、長時間かかることがあります。
V$LOGSTDBY_STATE ビューを問い合せると、ディクショナリのロード中は次の出力が戻されます。

```
SQL> SELECT SESSION_ID, STATE FROM V$LOGSTDBY_STATE;

SESSION_ID    STATE
----------    ------------------
1             LOADING DICTIONARY
```

LogMiner ディクショナリが完全にロードされるまでに起動されるのは、COORDINATOR プロセスとマイニング・プロセスのみです。したがって、この時点で V$LOGSTDBY_PROCESS を問い合せると、APPLIER プロセスは表示されません。次に例を示します。

```
SQL> SELECT SID, SERIAL#, SPID, TYPE FROM V$LOGSTDBY_PROCESS;

SID     SERIAL#     SPID       TYPE
------  ---------   ---------  ---------------------
47      3           11438      COORDINATOR
50      7           11334      READER
45      1           11336      BUILDER
44      2           11338      PREPARER
43      2           11340      PREPARER
```

V$LOGMNR_DICTIONARY_LOAD ビューを問い合せると、ディクショナリ・ロードの進捗に関する詳細情報を取得できます。ディクショナリ・ロードは、次の 3 つのフェーズで発生します。

1. LogMiner ディクショナリのロードに関連する REDO 変更を収集するために、関連アーカイブ REDO ログ・ファイルまたはスタンバイ REDO ログ・ファイルがマイニングされます。
2. 変更が処理され、データベース内のステージング表にロードされます。

3. 一連の DDL 文が発行されて、LogMiner ディクショナリ表がロードされます。

次に例を示します。

```
SQL> SELECT PERCENT_DONE, COMMAND
     FROM   V$LOGMNR_DICTIONARY_LOAD
     WHERE  SESSION_ID = (SELECT SESSION_ID FROM V$LOGSTDBY_STATE);

PERCENT_DONE     COMMAND
-------------    ------------------------------
40               alter table SYSTEM.LOGMNR_CCOL$ exchange partition
                 P101 with table SYS.LOGMNRLT_101_CCOL$ excluding
                 indexes without validation
```

PERCENT_DONE または COMMAND 列が長時間変化しない場合は、V$SESSION_LONGOPS ビューを問い合せ、問題の DDL トランザクションの進捗を監視してください。

適用中状態

この状態では、SQL Apply は LogMiner ディクショナリの初期スナップショットを正常にロードしており、現在はロジカル・スタンバイ・データベースに REDO データを適用中です。

SQL Apply の進捗の詳細情報を確認するには、次のように V$LOGSTDBY_PROGRESS ビューを問い合せます。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SQL> SELECT APPLIED_TIME, APPLIED_SCN, MINING_TIME, MINING_SCN,
     FROM V$LOGSTDBY_PROGRESS;

APPLIED_TIME            APPLIED_SCN   MINING_TIME           MINING_SCN
--------------------    -----------   --------------------  -----------
10-JAN-2005 12:00:05    346791023     10-JAN-2005 12:10:05  3468810134
```

プライマリ・データベース上で APPLIED_SCN（または APPLIED_TIME）以前に表示されるコミット済トランザクションは、すべてロジカル・スタンバイ・データベースに適用されています。マイニング・エンジンは、プライマリ・データベースで MINING_SCN（および MINING_TIME）以前に生成された REDO レコードをすべて処理しています。安定状態では、MINING_SCN（および MINING_TIME）は常に APPLIED_SCN（および APPLIED_TIME）よりも前の値を示します。

ギャップ待機中状態

この状態が発生するのは、SQL Apply が使用可能な全 REDO レコードのマイニングと適用を完了し、RFS プロセスによる新規ログ・ファイル（または欠落しているログ・ファイル）のアーカイブを待機している場合です。

```
SQL> SELECT STATUS FROM V$LOGSTDBY_PROCESS WHERE TYPE = 'READER';

STATUS
------------------------------------------------------------------------
ORA:01291 Waiting for logfile
```

アイドル状態

SQL Apply は、プライマリ・データベースで生成された REDO をすべて適用し終わると、この状態になります。

10.4.2 ログ・ファイルの自動削除

外部アーカイブ・ログには、プライマリ・データベースから送信された REDO が含まれています。外部アーカイブ・ログの格納には次の 2 つの方法があります。

- フラッシュ・リカバリ領域に格納
- フラッシュ・リカバリ領域外のディレクトリに格納

フラッシュ・リカバリ領域内に格納される外部アーカイブ・ログは、必ず SQL Apply によって管理されます。ログに含まれるすべての REDO レコードは、ロジカル・スタンバイ・データベースでの適用後、DB_FLASHBACK_RETENTION_TARGET パラメータで指定された期間（DB_FLASHBACK_RETENTION_TARGET が指定されていない場合は 1440 分間）保持されます。フラッシュ・リカバリ領域内に格納されている外部アーカイブ・ログの自動管理は上書きできません。

フラッシュ・リカバリ領域内に格納されない外部アーカイブ・ログは、デフォルトでは SQL Apply によって管理されます。自動管理下では、フラッシュ・リカバリ領域内に格納されない外部アーカイブ・ログは、ログに含まれるすべての REDO レコードがロジカル・スタンバイ・データベースで適用された後、LOG_AUTO_DEL_RETENTION_TARGET パラメータで指定された期間保持されます。フラッシュ・リカバリ領域内に格納されない外部アーカイブ・ログの自動管理は、次の PL/SQL プロシージャを実行すると上書きできます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE', FALSE);
```

注意： DBMS_LOGTSDBY.APPLY_SET プロシージャは、このパラメータの設定に使用します。LOG_AUTO_DEL_RETENTION_TARGET を明示的に指定しないと、ロジカル・スタンバイ・データベースに設定された DB_FLASHBACK_RETENTION_TARGET にデフォルトで設定されます。DB_FLASHBACK_RETENTION_TARGET が設定されていない場合は、1440 分に設定されます。

デフォルトの自動ログ削除機能を上書きする場合は、定期的に次の手順を実行して SQL Apply で不要になったアーカイブ REDO ログ・ファイルを識別し、削除します。

1. 不要になったメタデータのロジカル・スタンバイ・セッションを消去するには、次の PL/SQL 文を入力します。

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.PURGE_SESSION;
   ```

   この文では、不要になったアーカイブ REDO ログ・ファイルを表示する DBA_LOGMNR_PURGED_LOG ビューも更新されます。

2. DBA_LOGMNR_PURGED_LOG ビューを問い合せて、削除できるアーカイブ REDO ログ・ファイルのリストを表示します。

   ```
   SQL> SELECT * FROM DBA_LOGMNR_PURGED_LOG;

   FILE_NAME
   -----------------------------------
   /boston/arc_dest/arc_1_40_509538672.log
   /boston/arc_dest/arc_1_41_509538672.log
   /boston/arc_dest/arc_1_42_509538672.log
   /boston/arc_dest/arc_1_43_509538672.log
   /boston/arc_dest/arc_1_44_509538672.log
   /boston/arc_dest/arc_1_45_509538672.log
   /boston/arc_dest/arc_1_46_509538672.log
   /boston/arc_dest/arc_1_47_509538672.log
   ```

3. オペレーティング・システム固有のコマンドを使用して、問合せにより表示されたアーカイブ REDO ログ・ファイルを削除します。

10.5 ロジカル・スタンバイ・データベースのカスタマイズ

この項は、次の項目で構成されています。

- DBA_LOGSTDBY_EVENTS ビューでのイベントのロギングのカスタマイズ
- DBMS_LOGSTDBY.SKIP による特定のスキーマ・オブジェクトに対する変更の防止
- DDL 文のスキップ・ハンドラの設定
- ロジカル・スタンバイ・データベースの変更
- ロジカル・スタンバイ・データベースでの表の追加または再作成

関連項目：『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_LOGSTDBY パッケージに関する項

10.5.1 DBA_LOGSTDBY_EVENTS ビューでのイベントのロギングのカスタマイズ

DBA_LOGSTDBY_EVENTS ビューは、SQL Apply の下で発生する重要な最新イベントを含む循環型ログとみなすことができます。デフォルトでは、最新の 10,000 イベントがイベント・ビューに記録されます。ログに記録されるイベント数は、DBMS_LOGSTDBY.APPLY_SET プロシージャを起動して変更できます。たとえば、最新の 100,000 イベントを確実に記録するには、次の文を発行できます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET ('MAX_EVENTS_RECORDED', '100000');
```

SQL Apply の停止原因となったエラーは、SYSTEM 表領域に十分な領域があるかぎり、必ず DBA_LOGSTDBY_EVENTS ビューに記録されます。これらのイベントは常にアラート・ファイルにも格納され、テキストには LOGSTDBY というキーワードが含まれます。このビューを問い合せるときは、EVENT_TIME、COMMIT_SCN、CURRENT_SCN の順に列を選択してください。この順序によって、停止障害がビューの最後に表示されます。

次の例では、ビューに記録されるイベントを指定する DBMS_LOGSTDBY サブプログラムを示します。

例 1 DDL 文が適用されたかどうかを判断する

たとえば、適用済 DDL トランザクションを DBA_LOGSTDBY_EVENTS ビューに記録するには、次の文を発行します。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET ('RECORD_APPLIED_DDL', 'TRUE');
```

例 2 サポートされない操作がないか DBA_LOGSTDBY_EVENTS ビューをチェックする

プライマリ・データベースで実行されていて、ロジカル・スタンバイ・データベースでサポートされないトランザクションに関する情報を取得するには、次の文を発行します。

```
SQL> EXEC DBMS_LOGSTDBY.APPLY_SET('RECORD_UNSUPPORTED_OPERATIONS', 'TRUE');
```

次に、サポートされない操作がないか DBA_LOGSTDBY_EVENTS ビューをチェックします。通常、サポートされない表に対する操作は、SQL Apply によって警告なしで無視されます。しかし、ローリング・アップグレード中（スタンバイ・データベースが上位バージョンであり、下位バージョンのプライマリ・データベースによって生成された REDO をマイニングしているとき）に、サポートされない操作をプライマリ・データベースに対して実行した場合、ロジカル・スタンバイ・データベースにスイッチオーバーされない可能性があります。Data Guard は、表ごとに 1 つ以上のサポートされない操作を DBA_LOGSTDBY_EVENTS ビューに記録します。ローリング・アップグレードの詳細は、第 12 章「SQL Apply を使用した Oracle Database のアップグレード」を参照してください。

10.5.2 DBMS_LOGSTDBY.SKIP による特定のスキーマ・オブジェクトに対する変更の防止

デフォルトでは、プライマリ・データベース内でサポートされる表は、すべてロジカル・スタンバイ・データベースで複製されます。このデフォルト動作は、特定の表に対する変更の適用をスキップするルールを指定することで変更できます。たとえば、HR.EMPLOYEES 表に対する変更を省略するには、特定の表に対する DML 変更と DDL 変更の適用を防止するルールを指定できます。次に例を示します。

1. SQL Apply を停止します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
```

2. SKIP ルールを登録します。

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP (stmt => 'DML', schema_name => 'HR', -
           object_name => 'EMPLOYEES');
SQL> EXECUTE DBMS_LOGSTDBY.SKIP (stmt => 'SCHEMA_DDL', schema_name => 'HR', -
           object_name => 'EMPLOYEES');
```

3. SQL Apply を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

10.5.3 DDL 文のスキップ・ハンドラの設定

特定の DDL 文をインターセプトして元の DDL 文を別の DDL 文で置き換えるプロシージャを作成できます。たとえば、ロジカル・スタンバイ・データベース内のファイル・システムの編成がプライマリ・データベースとは異なる場合、DBMS_LOGSTDBY.SKIP プロシージャを作成し、ファイルを指定して DDL トランザクションを透過的に処理できます。

次のプロシージャでは、ファイル指定文字列に特定のネーミング規則を使用するかぎり、プライマリ・データベースとスタンバイ・データベース間で異なるファイル・システム編成を処理できます。

1. 次のように、表領域の DDL トランザクションを処理するスキップ・プロシージャを作成します。

```
CREATE OR REPLACE PROCEDURE SYS.HANDLE_TBS_DDL (
  OLD_STMT  IN  VARCHAR2,
  STMT_TYP  IN  VARCHAR2,
  SCHEMA    IN  VARCHAR2,
  NAME      IN  VARCHAR2,
  XIDUSN    IN  NUMBER,
  XIDSLT    IN  NUMBER,
  XIDSQN    IN  NUMBER,
  ACTION    OUT NUMBER,
  NEW_STMT  OUT VARCHAR2
) AS
BEGIN

-- All primary file specification that contains a directory
-- /usr/orcl/primary/dbs
-- should go to /usr/orcl/stdby directory specification


  NEW_STMT := REPLACE(OLD_STMT,
                      '/usr/orcl/primary/dbs',
                      '/usr/orcl/stdby');

  ACTION := DBMS_LOGSTDBY.SKIP_ACTION_REPLACE;

EXCEPTION
  WHEN OTHERS THEN
    ACTION := DBMS_LOGSTDBY.SKIP_ACTION_ERROR;
```

```
    NEW_STMT := NULL;
END HANDLE_TBS_DDL;
```

2. SQL Apply を停止します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
```

3. スキップ・プロシージャを SQL Apply に登録します。

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP (stmt => 'TABLESPACE', -
              proc_name => 'sys.handle_tbs_ddl');
```

4. SQL Apply を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

10.5.4 ロジカル・スタンバイ・データベースの変更

ロジカル・スタンバイ・データベースは、SQL 文が適用されている間でもレポート・アクティビティに使用できます。データベース・ガードはロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスを制御し、ALTER SESSION DISABLE GUARD 文はデータベース・ガードをバイパスしてロジカル・スタンバイ・データベース内の表に対する変更を可能にするために使用します。

注意： ロジカル・スタンバイ・データベースを使用して、独自に他の表を作成すると同時にプライマリ・データベースからレプリケートされるデータを処理する他のアプリケーションをホスト管理するには、データベース・ガードを STANDBY に設定する必要があります。このようなアプリケーションがシームレスに動作するには、PRESERVE_COMMIT_ORDER を TRUE（SQL Apply のデフォルト設定）に設定して実行する必要があります。（DBMS_LOGSTDBY PL/SQL パッケージの PRESERVE_COMMIT_ORDER パラメータの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。）

次の SQL 文を発行して、データベース・ガードを STANDBY に設定します。

```
SQL> ALTER DATABASE GUARD STANDBY;
```

このガード設定では、プライマリ・データベースからレプリケートされる表は、ユーザーによる変更から保護されますが、スタンバイ・データベースに作成された表は、ロジカル・スタンバイで実行されているアプリケーションで変更できます。

デフォルトでは、ロジカル・スタンバイ・データベースはデータベース・ガードが ALL（最も限定的な設定）に設定されている状態で作動し、ユーザーはデータベースに対して変更を実行できません。データベース・ガードを無視して、ロジカル・スタンバイ・データベースを変更可能にするには、ALTER SESSION DISABLE GUARD 文を実行します。権限を付与されているユーザーは、この文を発行して、現行のセッションでデータベース・ガードをオフにできます。

次の各項で、例を示します。これらの例では、データベース・ガードが ALL または STANDBY に設定されていると仮定しています。

10.5.4.1 ロジカル・スタンバイ・データベースでの DDL の実行

この項では、SQL Apply を介してメンテナンスされている表に制約を追加する方法を説明します。

デフォルトでは、SYS 権限を持つアカウントのみがデータベースを変更でき、データベース・ガードは ALL または STANDBY に設定されています。SYSTEM または権限を付与されている別のアカウントでログインした場合、最初はセッションに対するデータベース・ガードをバイパスしていないので、ロジカル・スタンバイ・データベースでは DDL 文を発行できません。

次の例は、SQL Apply を停止し、データベース・ガードをバイパスしてロジカル・スタンバイ・データベースで SQL 文を実行した後、データベース・ガードを再び使用可能にする方法を示しています。この例では、部分一致の問合せを高速化するために、soundex 索引が SCOTT.EMP の surname 列に追加されます。soundex 索引をプライマリ・サーバーで保持すると、非常に高コストになる場合があります。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered.

SQL> ALTER SESSION DISABLE GUARD;
PL/SQL procedure successfully completed.

SQL> CREATE INDEX EMP_SOUNDEX ON SCOTT.EMP(SOUNDEX(ENAME));
Table altered.

SQL> ALTER SESSION ENABLE GUARD;
PL/SQL procedure successfully completed.

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
Database altered.

SQL> SELECT ENAME,MGR FROM SCOTT.EMP WHERE SOUNDEX(ENAME) = SOUNDEX('CLARKE');

ENAME           MGR
----------      ----------
CLARK            7839
```

オラクル社では、データベース・ガードのバイパスが使用可能になっている間は、SQL Apply により保守される表に対して DML 操作を実行しないことをお薦めします。DML 操作を実行すると、プライマリ・データベースとスタンバイ・データベース間で違いが生じ、ロジカル・スタンバイ・データベースをメンテナンスできないようになります。

10.5.4.2 SQL Apply でメンテナンスされていない表の変更

場合によっては、レポート生成アプリケーションが、要約した結果を収集してその結果を一時的に格納したり、レポートが実行された回数を追跡する必要があります。アプリケーションの主な目的はレポート・アクティビティを実行することですが、ロジカル・スタンバイ・データベースに対して DML（挿入、更新および削除）操作の発行が必要な場合があります。さらに、アプリケーションで表の作成や削除が必要になることもあります。

データが SQL Apply を介してメンテナンスされていない場合は、レポート生成操作でデータを変更できるように、データベース・ガードを設定できます。次の設定を行う必要があります。

- DBMS_LOGSTDBY.SKIP プロシージャを実行して、アプリケーションによるデータの書込みを可能にする、ロジカル・スタンバイ・データベース上の表のセットを指定します。スキップされた表は、SQL Apply でメンテナンスされません。
- スタンバイ表のみを保護するようにデータベース・ガードを設定します。

次の例では、レポートによって書込みが行われる表がプライマリ・データベース上にあると仮定しています。

この例では、SQL Apply を停止し、表をスキップした後、SQL Apply を再開しています。この操作によって、レポート生成アプリケーションは、HR の TESTEMP% に書込みができるようになります。スキップされた表は、SQL Apply でメンテナンスされません。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered.

SQL> EXECUTE DBMS_LOGSTDBY.SKIP(stmt => 'SCHEMA_DDL',-
     schema_name => 'HR', -
     object_name => 'TESTEMP%');
PL/SQL procedure successfully completed.

SQL> EXECUTE DBMS_LOGSTDBY.SKIP('DML','HR','TESTEMP%');
PL/SQL procedure successfully completed.

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
Database altered.
```

SQL Apply では、開始後に、新しく指定されてスキップ・ルールに追加された表について、スタンバイ・データベースのメタデータを更新する必要があります。SQL Apply によりメタデータが更新される前に、新しくスキップされる表を変更しようとすると失敗します。追加したばかりの SKIP ルールが SQL Apply で正常に考慮されたかどうかは、次の問合せを発行すると確認できます。

```
SQL> SELECT VALUE FROM DBA_LOGSDTBY_PARAMETERS
     WHERE NAME = 'GUARD_STANDBY';

VALUE
---------------
Ready
```

VALUE 列に Ready と表示される場合、SQL Apply ではスキップされる表の関連メタデータがすべて正常に更新されており、その表を変更しても安全です。

> **関連項目：** C.11 項「ロジカル・スタンバイ・データベースでサポートされる DDL 文」および『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_LOGSTDBY パッケージに関する項

10.5.5 ロジカル・スタンバイ・データベースでの表の追加または再作成

通常、リカバリ不能な操作の後に表を再作成するには、DBMS_LOGSTDBY.INSTANTIATE_TABLE プロシージャを使用します。また、このプロシージャを使用して、以前はスキップしていた表に対して SQL Apply を使用可能にすることもできます。

表を作成する前に、4.1.2 項「プライマリ・データベース内の表の行が一意に識別できることの確認」に記載されている要件を満たす必要があります。その後、次の手順に従って表 HR.EMPLOYEES を再作成し、SQL Apply を再開できます。この手順は、プライマリ・データベースにアクセスするデータベース・リンク BOSTON が定義済であることを前提としています。

次のリストは、表を再作成し、その表に対して SQL Apply を再開する方法を示しています。

1. SQL Apply を停止します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
```

2. DBA_LOGSTDBY_SKIP ビューを問い合せ、問題の表で操作がスキップされていないことを確認します。

```
SQL> SELECT * FROM DBA_LOGSTDBY_SKIP;
ERROR  STATEMENT_OPT        OWNER          NAME                PROC
-----  -------------------  -------------  ----------------    -----
N      SCHEMA_DDL           HR             EMPLOYEES
N      DML                  HR             EMPLOYEES
N      SCHEMA_DDL           OE             TEST_ORDER
N      DML                  OE             TEST_ORDER
```

ロジカル・スタンバイ・データベース上で再作成する表には、すでにスキップ・ルールが関連付けられているため、最初にそのルールを削除する必要があります。そのためには、DBMS_LOGSTDBY.UNSKIP プロシージャをコールします。次に例を示します。

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP(stmt => 'DML', -
     schema_name => 'HR', -
     object_name => 'EMPLOYEES');

SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP(stmt => 'SCHEMA_DDL', -
     schema_name => 'HR', -
     object_name => 'EMPLOYEES');
```

3. DBMS_LOGSTDBY.INSTANTIATE_TABLE プロシージャを使用し、ロジカル・スタンバイ・データベースの全データを使用して表 HR.EMPLOYEES を再作成します。次に例を示します。

```
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE(schema_name => 'HR', -
     object_name => 'EMPLOYEES', -
     dblink => 'BOSTON');
```

4. SQL Apply を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

関連項目： DBMS_LOGSTDBY.UNSKIP および DBMS_LOGSTDBY.INSTANTIATE_TABLE プロシージャの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

新規にインスタンス化された表とデータベースの残りの部分にまたがってビューの一貫性が得られるように、SQL Apply がプライマリ・データベースと一致するまで待機してから、この表を問い合せます。手順は次のとおりです。

1. プライマリ・データベースで、V$DATABASE ビューを問い合せて現在の SCN を判別します。

```
SQL> SELECT CURRENT_SCN FROM V$DATABASE@BOSTON;
CURRENT_SCN
---------------------
345162788
```

2. 前の問合せで戻された CURRENT_SCN より前のコミット済のトランザクションがすべて、SQL Apply により適用されていることを確認します。

```
SQL> SELECT APPLIED_SCN FROM V$LOGSTDBY_PROGRESS;

APPLIED_SCN
--------------------------
345161345
```

この問合せで戻された APPLIED_SCN が最初の問合せで戻された CURRENT_SCN よりも大きい場合は、新規に再作成された表を問い合せても安全です。

10.6 ロジカル・スタンバイ・データベースの下での特定のワークロードの管理

この項は、次の項目で構成されています。

- プライマリ・データベースへのトランスポータブル表領域のインポート
- マテリアライズド・ビューの使用
- ロジカル・スタンバイ・データベースでのトリガーと制約の処理方法
- プライマリでの Point-in-Time リカバリの実行によるリカバリ

10.6.1 プライマリ・データベースへのトランスポータブル表領域のインポート

表領域をプライマリ・データベースにインポートする手順は、次のとおりです。

1. ロジカル・スタンバイ・データベースを変更できるように、ガード設定を無効にします。

```
SQL> ALTER DATABASE GUARD STANDBY;
```

2. ロジカル・スタンバイ・データベースで表領域をインポートします。
3. データベース・ガード設定を有効にします。

```
SQL> ALTER DATABASE GUARD ALL;
```

4. プライマリ・データベースで表領域をインポートします。

10.6.2 マテリアライズド・ビューの使用

ロジカル・スタンバイでは、マテリアライズド・ビューに関連した次の DDL 文を自動的にスキップします。

- CREATE、ALTER または DROP MATERIALIZED VIEW
- CREATE、ALTER または DROP MATERIALIZED VIEW LOG

ロジカル・スタンバイ・データベースの作成後にプライマリ・データベースで作成、変更または削除された新規マテリアライズド・ビューは、ロジカル・スタンバイ・データベースには反映されません。ただし、ロジカル・スタンバイ・データベースの作成前にプライマリ・データベースで作成されたマテリアライズド・ビューは、ロジカル・スタンバイ・データベースに存在します。

ロジカル・スタンバイでは、他の種類の補助データ構造に加えて、ロジカル・スタンバイ・データベースでの新規マテリアライズド・ビューのローカルな作成およびメンテナンスをサポートしています。たとえば、オンライン・トランザクション処理（OLTP）システムでは、高度に正規化された表を更新パフォーマンスに頻繁に使用しますが、これらの表により、複雑な意思決定支援の問合せに対するレスポンス時間が長くなることがあります。ロジカル・スタンバイ・データベースでより効率的な問合せが作成できるように、次のようにして、レプリケートされたデータを非正規化するマテリアライズド・ビューを作成できます。

```
SQL> ALTER SESSION DISABLE GUARD;

SQL> ALTER TABLE DEPT ADD (CONSTRAINT DEPT_PK PRIMARY KEY (DEPTNO));

SQL> ALTER TABLE  EMP ADD (CONSTRAINT EMP_FK FOREIGN KEY (DEPTNO)
  2  REFERENCES DEPT(DEPTNO));

SQL> CREATE MATERIALIZED VIEW LOG ON EMP
  2  WITH ROWID (EMPNO, ENAME, MGR, DEPTNO) INCLUDING NEW VALUES;

SQL> CREATE MATERIALIZED VIEW LOG ON DEPT
  2  WITH ROWID (DEPTNO, DNAME) INCLUDING NEW VALUES;

SQL> CREATE MATERIALIZED VIEW MANAGED_BY
  2  REFRESH ON DEMAND
  3  ENABLE QUERY REWRITE
  4  AS SELECT  E.ENAME, M.ENAME AS MANAGER
  5  FROM EMP E, EMP M WHERE E.MGR=M.EMPNO;

SQL> CREATE MATERIALIZED VIEW IN_DEPT
  2  REFRESH FAST ON COMMIT
  3  ENABLE QUERY REWRITE
  4  AS SELECT E.ROWID AS ERID, D.ROWID AS DRID, E.ENAME, D.DNAME
  5  FROM EMP E, DEPT D WHERE E.DEPTNO=D.DEPTNO;
```

ロジカル・スタンバイ・データベースの場合

- トランザクションのコミットが発生すると、ロジカル・スタンバイ・データベースの ON-COMMIT マテリアライズド・ビューが自動的にリフレッシュされます。
- ON-DEMAND マテリアライズド・ビューは自動的にリフレッシュされません。リフレッシュするには、DBMS_MVIEW.REFRESH プロシージャを実行する必要があります。

たとえば、次のコマンドを発行すると、前の例で作成された ON-DEMAND マテリアライズド・ビューがリフレッシュされます。

```
SQL> ALTER SESSION DISABLE GUARD;

SQL> EXECUTE DBMS_MVIEW.REFRESH (LIST => 'SCOTT.MANAGED_BY', METHOD => 'C');
```

DBMS_SCHEDULER ジョブを使用して ON-DEMAND マテリアライズド・ビューを定期的にリフレッシュする場合は、データベース・ガードを STANDBY に設定する必要があります。（ALTER SESSION DISABLE GUARD 文を PL/SQL ブロック内で使用して有効にすることはできません。）

10.6.3 ロジカル・スタンバイ・データベースでのトリガーと制約の処理方法

デフォルトでは、トリガーと制約はロジカル・スタンバイ・データベースで自動的に使用可能になり、処理されます。

SQL Apply でメンテナンスされる表にトリガーおよび制約がある場合 :

- 制約 : CHECK 制約はプライマリ・データベースで評価されます。ロジカル・スタンバイ・データベースでの再評価は不要です。
- トリガー : プライマリ・データベースで実行されたトリガーの影響は、スタンバイ・データベースでログに記録され、適用されます。このルールの例外は、`Fire_Once_Only` 起動プロパティが `FALSE` に設定されたトリガーです。

SQL Apply でメンテナンスされない表にトリガーおよび制約がある場合 :

- 制約は評価されます。
- トリガーは発行されます。

10.6.4 サポートされていない表のレプリケートでのトリガーの使用

単純なオブジェクト型の列のためにサポートされていない表を、Fire_Once_Only ではないトリガーを使用してレプリケートできます。通常の DML トリガーをプライマリで使用して、オブジェクト型を、サポート可能な表にフラット化できます。Fire_Once_Only ではないトリガーをロジカル・スタンバイで使用して、オブジェクト型を再作成し、サポートされていない表をトランザクション方式で更新できます。

関連項目 :

- `BMS_DDL.SET_TRIGGER_FIRING_PROPERTY` プロシージャおよび `DBMS_LOGSTDBY.IS_APPLY_SERVER` ファンクションの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

次の例は、トリガーを使用して単純なオブジェクト・タイプをレプリケートする方法を示しています。この例では、挿入処理の方法を示します。同じ原則を更新および削除に適用できます。ネストした表および VARRAY も、ネストしたデータの正規化ループという追加手順でこの方法を使用して、レプリケートできます。

```
-- simple object type
create or replace type Person as object
(
  FirstName    varchar2(50),
  LastName     varchar2(50),
  BirthDate    Date
)

-- unsupported object table
create table employees
(
  IdNumber     varchar2(10) ,
  Department   varchar2(50),
  Info         Person
)

-- supported table populated via trigger
create table employees_transfer
(
  t_IdNumber   varchar2(10),
  t_Department varchar2(50),
  t_FirstName  varchar2(50),
  t_LastName   varchar2(50),
  t_BirthDate  Date
)
```

```
--
-- create this trigger to flatten object table on the primary
-- this trigger will not fire on the standby
--
create or replace trigger flatten_employees
  after insert on employees for each row
declare
begin
  insert into employees_transfer
    (t_IdNumber, t_Department, t_FirstName, t_LastName, t_BirthDate)
  values
    (:new.IdNumber, :new.Department,
 :new.Info.FirstName,:new.Info.LastName, :new.Info.BirthDate);
end

--
-- create this trigger at the logical standby database
-- to populate object table on the standby
-- this trigger only fires when apply replicates rows to the standby
--
create or replace trigger reconstruct_employees
  after insert on employees_transfer for each row
begin
  if dbms_logstdby.is_apply_server() then
    insert into employees (IdNumber, Department, Info)
    values (:new.t_IdNumber, :new.t_Department,
          Person(:new.t_FirstName, :new.t_LastName, :new.t_BirthDate));
  end if;
end

-- set this trigger to fire from the apply server
execute dbms_ddl.set_trigger_firing_property(trig_owner=> 'scott', trig_name=>
'reconstruct_employees', fire_once => FALSE);
```

10.6.5 プライマリでの Point-in-Time リカバリの実行によるリカバリ

ロジカル・スタンバイ・データベースが REDO データの新規ブランチを受信すると、SQL Apply ではそれが自動的に使用されます。ロジカル・スタンバイ・データベースの場合、スタンバイ・データベースにより新規リセットログの SCN より後（REDO データの新規ブランチの開始より後）の REDO データが適用されていなければ、手動による介入は必要ありません。

次の表に、スタンバイ・データベースとプライマリ・データベースのブランチを再同期化する方法を示します。

スタンバイ・データベースの状態	操作	実行する手順
新規リセットログの SCN の後（REDO データの新規ブランチの開始より後）の REDO データが適用されていない場合	SQL Apply は、自動的に REDO データの新規ブランチを使用します。	手動による介入は必要ありません。SQL Apply は、自動的にスタンバイ・データベースを REDO データの新規ブランチと再同期化します。
新規リセットログの SCN の後（REDO データの新規ブランチの開始より後）の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっている場合	スタンバイ・データベースは、REDO データの将来の新規ブランチでリカバリされます。	1. 13.3.2 項「特定時点へのロジカル・スタンバイ・データベースのフラッシュバック」の手順に従ってロジカル・スタンバイ・データベースをフラッシュ・バックします。 2. SQL Apply を再開し、新規リセットログ・ブランチへの REDO の適用を続行します。 SQL Apply は、自動的にスタンバイ・データベースを新規ブランチと再同期化します。
新規リセットログの SCN の後（REDO データの新規ブランチの開始より後）の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっていない場合	指定のプライマリ・データベース・ブランチで、プライマリ・データベースとスタンバイの内容にずれが生じます。	第 4 章「ロジカル・スタンバイ・データベースの作成」の手順に従ってロジカル・スタンバイ・データベースを再作成します。
REDO データの前のブランチの終わりからアーカイブ REDO ログ・ファイルが欠落している場合	SQL Apply は欠落しているログ・ファイルが取得されるまで続行できません。	前のブランチから欠落しているアーカイブ REDO ログ・ファイルを検索して登録します。

データベース・インカネーション、`OPEN RESETLOGS` 操作を使用したリカバリおよびフラッシュバック・データベースの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

10.7 ロジカル・スタンバイ・データベースのチューニング

この項は、次の項目で構成されています。

- 主キー RELY 制約の作成
- コストベース・オプティマイザの統計の収集
- プロセス数の調整
- LCR キャッシュ用メモリーの調整
- ロジカル・スタンバイ・データベースにおけるトランザクションの適用方法の調整

10.7.1 主キー RELY 制約の作成

プライマリ・データベースで、表に主キーまたは一意索引がなく、実際には一意である行がわかっている場合は、主キーの RELY 制約を作成します。ロジカル・スタンバイ・データベースでは、主キーを構成する列に索引を作成します。次の問合せは、ロジカル・スタンバイ・データベースで行を一意に識別するために適用できる索引情報がない表のリストを生成します。次の表に索引を作成することによって、パフォーマンスが大幅に向上する可能性があります。

```
SQL> SELECT OWNER, TABLE_NAME FROM DBA_TABLES
  2> WHERE OWNER NOT IN (SELECT OWNER FROM DBA_LOGSTDBY_SKIP
  3> WHERE STATEMENT_OPT = 'INTERNAL SCHEMA')
  4> MINUS
  5> SELECT DISTINCT TABLE_OWNER, TABLE_NAME FROM DBA_INDEXES
  6> WHERE INDEX_TYPE NOT LIKE ('FUNCTION-BASED%')
  7> MINUS
  8> SELECT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED;
```

プライマリ・データベースの表に主キーの RELY 制約を追加する手順は、次のとおりです。

1. プライマリ・データベースで主キーの RELY 制約を追加します。

   ```
   SQL> ALTER TABLE HR.TEST_EMPLOYEES ADD PRIMARY KEY (EMPNO) RELY DISABLE;
   ```

   これにより、HR.TEST_EMPLOYEES 表の各行を一意に識別するために使用可能な EMPNO 列が、その表に対して実行される更新の一部としてサプリメンタル・ロギングで確実に記録されるようになります。

   HR.TEST_EMPLOYEES 表には、ロジカル・スタンバイ・データベースで指定された一意索引がまだないことに注意してください。このため、UPDATE 文ではロジカル・スタンバイ・データベースに対して全表スキャンが実行される場合があります。これを避けるには、ロジカル・スタンバイ・データベースで EMPNO 列に一意索引を追加します。

   RELY 制約の詳細は、4.1.2 項「プライマリ・データベース内の表の行が一意に識別できることの確認」および『Oracle Database SQL 言語リファレンス』を参照してください。

   残りの手順をロジカル・スタンバイ・データベースで実行します。

2. SQL Apply を停止します。

   ```
   SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   ```

3. ロジカル・スタンバイ・データベースでメンテナンスされる表を変更できるように、ガードを無効にします。

   ```
   SQL> ALTER SESSION DISABLE GUARD;
   ```

4. EMPNO 列に一意索引を追加します。

   ```
   SQL> CREATE UNIQUE INDEX UI_TEST_EMP ON HR.TEST_EMPLOYEES (EMPNO);
   ```

5. ガードを有効にします。

   ```
   SQL> ALTER SESSION ENABLE GUARD;
   ```

6. SQL Apply を開始します。

   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
   ```

10.7.2 コストベース・オプティマイザの統計の収集

コストベース・オプティマイザ（CBO）で最適の問合せ実行パスを判別するために使用されるため、スタンバイ・データベースで統計を収集する必要があります。以前の統計情報が正確でなくなるような変更をスキーマ・オブジェクトのデータまたは構造に対して実行した場合は、その後に新しい統計を収集してください。たとえば、多数の行を表に挿入または削除した後は、行数に関する新しい統計を収集します。

プライマリ・データベースでの DML および DDL 操作はワークロードの機能として実行されるため、統計はスタンバイ・データベースで収集してください。スタンバイ・データベースがプライマリ・データベースと論理的に等しい場合、SQL Apply ではワークロードを異なる方法で実行する場合があります。そのため、ロジカル・スタンバイ・データベースで STATS パッケージおよび V$SYSSTAT ビューを使用すると、リソースおよび表スキャンを最も多く使用している表を判断する際に役立ちます。

関連項目：

- 4.1.2 項「プライマリ・データベース内の表の行が一意に識別できることの確認」
- RELY 制約の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

10.7.3 プロセス数の調整

次の各項で、次の内容を説明します。

- APPLIER プロセス数の調整
- PREPARER プロセス数の調整

SQL Apply に割り当てられるプロセス数を制御するために変更できるパラメータには、MAX_SERVERS、APPLY_SERVERS および PREPARE_SERVERS の 3 つがあります。常に、次の関係に該当する必要があります。

- `APPLY_SERVERS + PREPARE_SERVERS = MAX_SERVERS - 3`

  これは、SQL Apply が常に READER、BUILDER、ANALYZER の各ロールに対して 1 つずつプロセスを割り当てるためです。

- デフォルトでは、MAX_SERVERS は 9 に、PREPARE_SERVERS は 1 に、APPLY_SERVERS は 5 に設定されます。

- DBMS_LOGSTDBY.APPLY_SET プロシージャを使用して MAX_SERVERS パラメータのみを変更し、SQL Apply でサーバー・プロセスを準備プロセスと適用プロセスの間に適当に分散できようにすることをお薦めします。

- SQL Apply ではプロセス割当てアルゴリズムを使用して、MAX_SERVER で指定されるように、SQL Apply に割り当てられる 20 サーバー・プロセスごとに 1 つの PREPARE_SERVER を割り当て、PREPARE_SERVERS の数を 5 に制限します。そのため、MAX_SERVERS を 1 ～ 20 の任意の値に設定すると、SQL Apply は PREPARER として機能するようにサーバー・プロセスを 1 つ割り当て、残りのプロセスを APPLIER として割り当てて前述の関係を満たします。同様に、MAX_SERVERS を 21 ～ 40 の値に設定すると、SQL Apply は PREPARER として機能するようにサーバー・プロセスを 2 つ割り当て、残りを APPLIER として割り当てて前述の関係を満たします。この内部プロセス割当てアルゴリズムは、前述の関係を満たしているならば、APPLY_SERVERS および PREPARE_SERVERS を直接設定することで上書きできます。

10.7.3.1 APPLIER プロセス数の調整

APPLIER プロセスの数を調整することでスループットを改善できるかどうかを判断する手順は、次のとおりです。

1. 次の問合せを発行して、APPLIER プロセスがビジーかどうかを判断します。

```
SQL> SELECT COUNT(*) AS IDLE_APPLIER
     FROM V$LOGSTDBY_PROCESS
     WHERE TYPE = 'APPLIER' and status_code = 16166;

IDLE_APPLIER
------------------------
0
```

2. アイドル状態の APPLIER プロセスがないことを確認した後、次の問合せを発行し、APPLIER の数を調整した場合に追加分の APPLIER プロセスに十分な作業が使用可能かどうかを確認します。

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME LIKE
'transactions%';
```

この 2 つの統計では、APPLIER プロセスによる適用準備が完了しているトランザクション数とすでに適用済のトランザクション数の累計が維持されます。

この数値（transactions mined - transactions applied）が使用可能な APPLIER プロセス数の 2 倍を超えている場合は、APPLIER プロセス数を増やすことでスループットを改善できます。

> **注意：** この数値は準備作業の概算です。ワークロードによっては、準備が完了したトランザクション間の依存性が、使用可能な追加の APPLIER プロセスによる適用の妨げになる場合があります。たとえば、適用準備の完了しているトランザクションの大多数が DDL トランザクションの場合は、さらに APPLIER プロセスを追加してもスループットは改善されません。

APPLIER プロセス数をデフォルト値の 5 から 20 に調整し、PREPARER プロセス数は 1 のままとします。これは、次の等式を満たす必要があるためです。

```
APPLY_SERVERS + PREPARE_SERVERS = MAX_SERVERS - 3
```

まず、MAX_SERVERS を 24 に設定する必要があります。設定したら、次のようにして、APPLY_SERVERS の数を 20 に設定できます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SERVERS', 24);
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('APPLY_SERVERS', 20);
```

10.7.3.2 PREPARER プロセス数の調整

PREPARER プロセスの数の調整が必要になることは、ほとんどありません。PREPARER プロセスの数を増やす前に、次の条件に該当しているかどうかを確認してください。

- すべての PREPARER プロセスがビジーである。
- 適用準備が完了しているトランザクションの数が、使用可能な APPLIER プロセスの数よりも少ない。
- アイドル状態の APPLIER プロセスがある。

前述の条件に該当しているかどうかを判断する手順は、次のとおりです。

1. すべての PREPARER プロセスがビジーであることを確認します。

```
SQL> SELECT COUNT(*) AS IDLE_PREPARER
     FROM V$LOGSTDBY_PROCESS
     WHERE TYPE = 'PREPARER' and status_code = 16166;
IDLE_PREPARER
-------------
0
```

2. 適用準備が完了しているトランザクションの数が APPLIER プロセスの数よりも少ないことを確認します。

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS
     WHERE NAME LIKE 'transactions%';
NAME                          VALUE
----------------------        -------
transactions ready             27896
transactions applied           27892

SQL> SELECT COUNT(*) AS APPLIER_COUNT
     FROM V$LOGSTDBY_PROCESS WHERE TYPE = 'APPLIER';
APPLIER_COUNT
-------------
20
```

   注意 : これが一時イベントでないことを確認するために、この問合せは数回発行してください。

3. アイドル状態の APPLIER プロセスがあることを確認します。

```
SQL> SELECT COUNT(*) AS IDLE_APPLIER
     FROM V$LOGSTDBY_PROCESS
     WHERE TYPE = 'APPLIER' and status_code = 16166;
IDLE_APPLIER
------------------------
19
```

この例では、PREPARER プロセス数の増加に必要な 3 つの条件がすべて満たされています。APPLIER プロセス数の設定を 20 のままにし、PREPARER プロセス数を 1 から 3 に増やすとします。これは、常に次の等式を満たす必要があるためです。

```
APPLY_SERVERS + PREPARE_SERVERS = MAX_SERVERS - 3
```

まず、増加後の PREPARER プロセス数に対応するように、MAX_SERVERS の数を 24 から 26 に増やす必要があります。次に、PREPARER プロセス数を次のようにして増やすことができます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SERVERS', 26);
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('PREPARE_SERVERS', 3);
```

10.7.4 LCR キャッシュ用メモリーの調整

ある程度のワークロードの場合、SQL Apply は多数のページアウト操作を使用することがあるため、システム全体のスループットが低下します。LCR キャッシュに割当済のメモリーを増やした場合にスループットを改善できるかどうかを判断する手順は、次のとおりです。

1. 次の問合せを発行して、ページアウト・アクティビティのスナップショットを取得します。

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME LIKE '%page%'
OR NAME LIKE '%uptime%' OR NAME LIKE '%idle%';

NAME                          VALUE
---------------------------   ---------------
coordinator uptime in secs              894856
bytes paged out                          20000
seconds spent in pageout                     2
system idle time in secs                  1000
```

2. この問合せを 5 分以内に再発行します。

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME LIKE '%page%'
OR NAME LIKE '%uptime%' OR NAME LIKE '%idle%';

NAME                          VALUE
---------------------------   ---------------
coordinator uptime in secs              895156
bytes paged out                        1020000
seconds spent in pageout                   100
system idle time in secs                  1000
```

3. 正規化されたページアウト・アクティビティを計算します。次に例を示します。

```
Change in coordinator uptime (C)= (895156 - 894856) = 300 secs
Amount of additional idle time (I)= (1000 - 1000) = 0
Change in time spent in pageout (P) = (100 - 2) = 98 secs
Pageout time in comparison to uptime = P/(C-I) = 98/300 ~ 32.67%
```

ページアウト・アクティビティによる消費量が稼働時間全体の 5% 以下であれば理想的です。間隔を長くして引き続きスナップショットを取得し、ページアウト・アクティビティが引き続き適用時間の大部分を占めていることがわかった場合は、メモリー・サイズを大きくするとスループットが改善される可能性があります。SQL Apply に割当済のメモリーは、LCR キャッシュに割り当てられるメモリーを設定する（この例では SGA を 1GB に設定）と増やすことができます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SGA', 1024);
PL/SQL procedure successfully completed
```

10.7.5 ロジカル・スタンバイ・データベースにおけるトランザクションの適用方法の調整

デフォルトでは、トランザクションは、プライマリ・データベースでコミットされた順序に正確に従って、ロジカル・スタンバイ・データベースに適用されます。厳密なデフォルト順序でのトランザクションのコミットにより、アプリケーションをロジカル・スタンバイ・データベースで透過的に実行できます。

ただし、多くのアプリケーションでは、すべてのトランザクションの間でそのような厳密な順序が必要なわけではありません。そのようなアプリケーションでは、オーバーラップしない一連の行を含むトランザクションを、プライマリ・データベースでコミットしたときと同じ順序でコミットする必要がありません。一般に、厳密でない順序により、ロジカル・スタンバイ・データベースでの適用率は高くなります。次の手順で、トランザクションをコミットするデフォルトの順序を変更できます。

1. SQL Apply を停止します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered
```

2. 次の文を発行して、プライマリ・データベースでコミットされた順序と異なる順序でトランザクションを適用できるようにします。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('PRESERVE_COMMIT_ORDER', 'FALSE');
PL/SQL procedure successfully completed
```

3. SQL Apply を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
Database altered
```

プライマリ・データベースと一致し（V$LOGSTDBY_STATS ビューを問い合せて確認し）、ロジカル・スタンバイ・データベースをレポート・アプリケーションに対してオープンする準備が完了した時点で、次のように適用モードを変更できます。

1. SQL Apply を停止します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered
```

2. PRESERVE_COMMIT_ORDER パラメータのデフォルト値をリストアします。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_UNSET('PRESERVE_COMMIT_ORDER');
PL/SQL procedure successfully completed
```

3. SQL Apply を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
Database altered
```

一般的なオンライン・トランザクション処理（OLTP）のワークロードでは、デフォルト以外のモードにすると、デフォルトの適用モードに比べてスループットを 50% 以上改善できます。

10.8 ロジカル・スタンバイ・データベースの下でのバックアップおよびリカバリ

ロジカル・スタンバイ・データベースは、従来の方法を使用してバックアップでき、データベース・バックアップをリストアしてアーカイブ・ログに対してメディア・リカバリを実行すると、バックアップと同時にリカバリができます。次の各項目は、ロジカル・スタンバイ・データベースの場合に該当します。

Recovery Manager のローカル・リカバリ・カタログの作成および使用時の考慮事項

Recovery Manager のリカバリ・カタログを作成する、またはカタログを変更する Recovery Manager のアクティビティを実行する場合、ロジカル・スタンバイ・データベースでは、GUARD を STANDBY に設定して実行する必要があります。

ローカル・リカバリ・カタログがロジカル・スタンバイ制御ファイルにのみ保持されている場合は、GUARD を ALL に設定したままにできます。

制御ファイルのバックアップに関する考慮事項

ロジカル・スタンバイ・データベースのインスタンス化の直後に、制御ファイルのバックアップを取得することをお薦めします。

Point-in-Time リカバリに関する考慮事項

Point-in-Time リカバリの後に初めて SQL Apply が開始されるとき、必要なアーカイブ・ログをローカル・システムで検出するか、プライマリ・データベースからフェッチできる必要があります。V$LOGSTDBY_PROCESS ビューを使用して、アーカイブ・ログをプライマリ・データベースでリストアする必要があるかどうかを判断します。

表領域の Point-in-Time リカバリに関する考慮事項

ロジカル・スタンバイ・データベースの表領域に対して Point-in-Time リカバリを実行する場合、次のいずれかを実行する必要があります。

- 表領域に SQL Apply プロセスでメンテナンスされている表またはパーティションが含まれないことを確認する。
- 表領域に SQL Apply プロセスでメンテナンスされている表またはパーティションが含まれる場合は、ロジカル・スタンバイ・データベースで、DBMS_LOGSTDBY.INSTANTIATE_TABLE プロシージャを使用して、リカバリされる表領域に含まれるメンテナンス対象の表をすべて再度インスタンス化するか、または DBMS_LOGSTDBY.SKIP プロシージャを使用してリカバリされる表領域に含まれる表をすべて登録し、メンテナンス対象の表のリストからスキップされるようにする。

11

Recovery Manager を使用したファイルのバックアップおよびリストア

この章では、Data Guard およびスタンバイ・データベースとともに Oracle Recovery Manager（RMAN）を使用するバックアップ対策について説明します。Recovery Manager を使用すると、最小限の労力でプライマリ・データベースのバックアップを実行し、個々のデータファイルの消失またはデータベース全体をすばやくリカバリできます。Recovery Manager と Data Guard を併用すると、Data Guard 構成の管理作業を簡素化できます。

この章は、次の項目で構成されています。

注意： ロジカル・スタンバイ・データベースはプライマリ・データベースのブロック単位のコピーではないため、ロジカル・スタンバイ・データベースを使用してプライマリ・データベースをバックアップすることはできません。

関連項目：

- Recovery Manager の概念および Data Guard 環境での Recovery Manager の使用方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- この章で使用される Recovery Manager の全コマンドの詳細は、『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

11.1 Data Guard 構成における Recovery Manager ファイル管理の概要

Recovery Manager では、リカバリ・カタログを使用して、Data Guard 環境のすべてのデータベース・ファイルについてファイル名を追跡管理します。リカバリ・カタログは、1 つ以上の Oracle データベースに関するメタデータを格納するために Recovery Manager で使用されるデータベース・スキーマです。また、カタログには、オンライン REDO ログ、スタンバイ REDO ログ、テンプレート、アーカイブ REDO ログ、バックアップ・セットおよびイメージ・コピーが作成される場所も記録されます。

11.1.1 Data Guard 環境におけるバックアップの互換性

Recovery Manager コマンドは、リカバリ・カタログのメタデータを使用して、Data Guard 環境の様々なフィジカル・データベースにわたって透過的に動作します。たとえば、フィジカル・スタンバイ・データベースで表領域のバックアップを作成し、プライマリ・データベースでリストアおよびリカバリできます。同様に、プライマリ・データベースで表領域のバックアップを作成し、フィジカル・スタンバイ・データベースでリストアおよびリカバリできます。

注意： ロジカル・スタンバイ・データベースのバックアップは、プライマリ・データベースでは使用できません。

スタンバイ制御ファイルとスタンバイ以外の制御ファイルのバックアップには互換性があります。たとえば、スタンバイ制御ファイルをプライマリ・データベースでリストアしたり、プライマリ制御ファイルをフィジカル・スタンバイ・データベースでリストアできます。この互換性は、制御ファイルのバックアップを Data Guard 環境の 1 つのデータベースにオフロードできるということです。Recovery Manager は、データベースでのリストアおよびリカバリ時に、データベース・ファイルのファイル名を自動的に更新します。

11.1.2 Data Guard 環境におけるバックアップの関連付け

リカバリ・カタログでは、すべてのデータベース・ファイル（バックアップ・ファイル）を DB_UNIQUE_NAME に関連付けて、Data Guard 環境のファイルを追跡管理します。ファイルを作成するデータベースは、そのファイルに関連付けられます。たとえば、Recovery Manager で一意の名前 standby1 が付けられたデータベースのバックアップが作成されると、standby1 はこのバックアップに関連付けられます。バックアップは、CHANGE ... RESET DB_UNIQUE_NAME を使用して別のデータベースに関連付けないかぎり、作成元のデータベースに関連付けられたままです。

11.1.3 Data Guard 環境におけるバックアップのアクセス可能性

バックアップのアクセス可能性は、関連付けとは異なります。Data Guard 環境では、リカバリ・カタログは、ディスク・バックアップは関連付けられたデータベースでのみアクセス可能であるとみなします。これに対して、あるデータベースで作成されたテープ・バックアップは、すべてのデータベースでアクセス可能です。バックアップ・ファイルがデータベースに関連付けられていない場合は、リカバリ・カタログ・ビューでそれを示す行の SITE_KEY 列に null と表示されます。デフォルトでは、SITE_KEY が null のファイルは、Recovery Manager によってターゲット・データベースに関連付けられます。

BACKUP、RESTORE、CROSSCHECK などの Recovery Manager コマンドは、アクセス可能なバックアップに対して作用します。たとえば、RECOVER COPY 操作の場合、Recovery Manager はデータベースに関連付けられているイメージ・コピーのみをリカバリ対象とみなします。また、ディスクおよびテープ上の増分バックアップをイメージ・コピーのリカバリ対象とみなします。データベース・リカバリでは、Recovery Manager は、データベースに関連付けられたディスク・バックアップのみとテープ上のすべてのファイルをリストアの対象とみなします。

バックアップのアクセス可能性における差異を示すため、データベース prod および standby1 が異なるホスト上に存在すると仮定します。Recovery Manager は、prod のデータファイル 1 のバックアップを本番ホスト上にある /prmhost/disk1/df1.dbf およびテープに作成します。また、standby1 のデータファイル 1 のバックアップをスタンバイ・ホスト上にある /sbyhost/disk2/df1.dbf およびテープに作成します。Recovery Manager がデータベース prod に接続されると、Recovery Manager コマンドを使用してスタンバイ・ホスト上にある /sbyhost/disk2/df1.dbf バックアップで操作を実行することはできません。しかし、Recovery Manager は standby1 で作成したテープ・バックアップをリストア対象とみなします。

注意： スタンバイ・ホストからプライマリ・ホスト（またはその逆）にバックアップを FTP で送信し、TARGET としてそのホスト上のデータベースに接続すると、そのバックアップに CATALOG を実行できます。ファイルは、ターゲット・データベースによってカタログに追加された後に、ターゲット・データベースに関連付けられます。

11.2 Data Guard 環境における Recovery Manager 構成の概要

Data Guard 構成では、制御ファイル、データファイルおよびアーカイブ・ログのバックアップ処理をスタンバイ・システムにオフロードできるため、バックアップが本番システムに与える影響は最小限に抑えられます。これらのバックアップを使用すると、プライマリ・データベースまたはスタンバイ・データベースをリカバリできます。

Recovery Manager では、DB_UNIQUE_NAME 初期化パラメータを使用してデータベース・サイトどうしを区別します。そのため、DB_UNIQUE_NAME の一意性が Data Guard 構成で維持されることは不可欠なことです。

Recovery Manager の REGISTER DATABASE コマンドを使用して明示的に登録する必要があるのは、プライマリ・データベースのみです。この登録は、Recovery Manager をリカバリ・カタログに接続し、ターゲットとしてプライマリ・データベースに接続した後に行います。

Recovery Manager 構成を設定するには、Recovery Manager の CONFIGURE コマンドを使用します。CONFIGURE コマンドが FOR DB_UNIQUE_NAME オプションを指定して使用されると、指定した DB_UNIQUE_NAME のデータベースに対して Recovery Manager サイト固有の構成が設定されます。

たとえば、リカバリ・カタログへの接続後、次のコマンドを Recovery Manager プロンプトで使用して、DBID が 1625818158 の BOSTON データベースに対してデフォルト・デバイス・タイプを SBT に設定できます。Recovery Manager の SET DBID コマンドは、ターゲットとしてデータベースに接続していない場合にのみ必要です。

```
SET DBID 1625818158;
CONFIGURE DEFAULT DEVICE TYPE TO SBT FOR DB_UNIQUE_NAME BOSTON;
```

11.3 推奨される Recovery Manager 構成および Oracle Database 構成

この項では、次の Recovery Manager 構成および Oracle Database 構成について説明します。各構成によってバックアップ操作およびリカバリ操作を簡略化できます。

- プライマリおよびスタンバイ・データベースでの Oracle Database 構成
- プライマリ・データベースでの Recovery Manager 構成
- バックアップが実行されるスタンバイ・データベースでの Recovery Manager 構成
- バックアップが実行されないスタンバイでの Recovery Manager 構成

構成の前提

この項で説明する構成は、次の前提に基づいています。

- スタンバイ・データベースはフィジカル・スタンバイ・データベースで、バックアップはスタンバイ・データベースでのみ作成されます。プライマリ・データベースとスタンバイ・データベースの両方でバックアップを作成する場合の手順の変更は、11.9.1 項を参照してください。
- Recovery Manager のリカバリ・カタログは、あるデータベース・サーバーで作成されたバックアップを別のデータベース・サーバーにリストアできるようにするために必要です。制御ファイルのみを Recovery Manager のリポジトリとして使用するのは不十分です。これは、プライマリ・データベースがスタンバイ・データベースで作成されたバックアップを認識しないためです。

  Recovery Manager のリカバリ・カタログは、バックアップ履歴および他のリカバリ関連のメタデータを一元化された場所に編成します。リカバリ・カタログは、データベース内に構成され、バックアップ・メタデータが格納されます。リカバリ・カタログには、制御ファイルに関する領域の制限がないため、バックアップに関する履歴データをより多く格納できます。

  カタログ・サーバーは、プライマリ・サイトおよびスタンバイ・サイトと物理的に切り離されていますが、Data Guard 構成内に置くことをお薦めします。これは、いずれのサイトで障害が発生しても最新のバックアップをリカバリする機能に影響しないためです。

  > **関連項目：** リカバリ・カタログの管理の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

- 構成内のすべてのデータベースには、Oracle Database 11*g* リリース 1（11.1）を使用します。
- Oracle Secure Backup ソフトウェアまたはサード・パーティのメディア管理ソフトウェアは、バックアップをテープに取るように Recovery Manager で構成します。

11.3.1 プライマリおよびスタンバイ・データベースでの Oracle Database 構成

次の Oracle Database 構成は、Data Guard 環境におけるすべてのプライマリ・データベースおよびスタンバイ・データベースで推奨されます。

- データベースごとにフラッシュ・リカバリ領域を構成する（リカバリ領域はデータベースに対してローカル）。

  フラッシュ・リカバリ領域は、ファイル・システムまたは自動ストレージ管理（ASM）ディスク・グループ上にある、リカバリに必要なすべてのファイルが存在する唯一の格納場所です。これらのファイルには、制御ファイル、アーカイブ・ログ、オンライン REDO ログ、フラッシュバック・ログおよび Recovery Manager バックアップがあります。新しいバックアップおよびアーカイブ・ログがフラッシュ・リカバリ領域に作成されると、古いファイル（保存期間を超過しているまたは三次記憶域にバックアップされている）は自動的に削除され、新しいファイル用の領域が作成されます。さらに、フラッシュ・リカバリ領域の領域使用量が事前に定義された制限に近づいている場合に、DBA にアラートを送信するように通知を設定できます。DBA は、リカバリ領域の領域制限を上げる、ディスク・ハードウェアを追加する、あるいは保存期間を短縮するなどの処置を取ることができます。

  次の初期化パラメータを設定して、フラッシュ・リカバリ領域を構成します。

  ```
  DB_RECOVERY_FILE_DEST = <mount point or ASM Disk Group>
  DB_RECOVERY_FILE_DEST_SIZE = <disk space quota>
  ```

  > **関連項目：** フラッシュ・リカバリ領域の構成の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

- サーバー・パラメータ・ファイル（SPFILE）を使用して、インスタンス・パラメータがバックアップに保存されるようにバックアップを作成できるようにする。
- プライマリ・データベースおよびスタンバイ・データベースでフラッシュバック・データベースを有効にする。

  フラッシュバック・データベースを有効に設定すると、Oracle Database によってフラッシュバック・ログがフラッシュ・リカバリ領域に保持されます。これらのログを使用すると、完全にリストアしなくても、前の時点までデータベースをロールバックできます。

  > **関連項目：** フラッシュバック・データベースを有効に設定する方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

11.3.2 プライマリ・データベースでの Recovery Manager 構成

Recovery Manager の継続的な使用を簡略化するために、多くの永続的な構成設定を Data Guard 環境のデータベースごとに設定できます。これらの設定により、Recovery Manager の動作を多面的に制御します。たとえば、バックアップの保存ポリシー、デフォルトのバックアップ先（テープまたはディスク）、デフォルトのバックアップ・デバイス・タイプなどを構成できます。CONFIGURE コマンドを使用すると、Recovery Manager 構成を設定および変更できます。次の Recovery Manager 構成は、プライマリ・データベースにおいて推奨されます。

1. Recovery Manager をプライマリ・データベースおよびリカバリ・カタログに接続します。
2. データベースの保存ポリシーを *n* 日間として構成します。

   ```
   CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF <n> DAYS;
   ```

   この構成により、任意の時点までのデータベース・リカバリの実行に必要なバックアップを指定した日数の間保存できます。

   DELETE OBSOLETE コマンドを使用して、（所定の保存ポリシーに従って）指定した日数以内でリカバリを実行するのに必要ではなくなったバックアップを削除します。

3. アーカイブ・ログを削除できる時期を CONFIGURE ARCHIVELOG DELETION POLICY コマンドを使用して指定します。たとえば、すべての宛先への送信を確認した後にログを削除する場合は、次の構成を使用します。

```
CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO ALL STANDBY;
```

すべてのスタンバイ宛先への適用を確認した後にログを削除する場合は、次の構成を使用します。

```
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
```

4. RESYNC CATALOG FROM DB_UNIQUE_NAME コマンドの使用時に Recovery Manager がリモート接続して再同期化を実行できるように、プライマリ・データベースおよびすべてのスタンバイ・データベースについて接続文字列を構成します。ターゲット・インスタンスに接続するとき、ネット・サービス名を指定する必要があります。この要件は、再同期化が実行される他のデータベース・インスタンスがローカル・ホスト上にある場合にも当てはまります。ターゲット・インスタンスおよびリモート・インスタンスは、同じパスワード SYSDBA を使用する必要があります。すなわち、どちらのインスタンスにもパスワード・ファイルがすでに存在している必要があります。パスワード・ファイルは単一のパスワードで作成できるため、すべてのデータベース・インスタンスはそのパスワード・ファイルを使用して開始できます。たとえば、ボストンのスタンバイに接続する TNS 別名が boston_conn_str の場合、次のコマンドを使用して BOSTON データベース・サイトについて接続識別子を構成できます。

```
CONFIGURE DB_UNIQUE_NAME BOSTON CONNECT IDENTIFIER 'boston_conn_str';
```

'boston_conn_str' には、ユーザー名およびパスワードが含まれていないことに注意してください。任意のデータベース・サイトから BOSTON データベース・サイトに接続するために使用できる Oracle Net サービス名のみが含まれます。

すべてのスタンバイ・データベースについて接続識別子が構成されたら、LIST DB_UNIQUE_NAME OF DATABASE コマンドを使用してスタンバイのリストを確認できます。

関連項目：

- Recovery Manager 構成の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- Recovery Manager の CONFIGURE コマンドの詳細は、『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

11.3.3 バックアップが実行されるスタンバイ・データベースでの Recovery Manager 構成

次の Recovery Manager 構成は、バックアップが実行されるスタンバイ・データベースで推奨されます。

1. Recovery Manager をターゲットとして（バックアップが実行される）スタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. 制御ファイルおよびサーバー・パラメータ・ファイルの自動バックアップを有効にします。

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

3. 同じチェックポイントで有効なバックアップがすでに存在するデータファイルのバックアップをスキップします。

```
CONFIGURE BACKUP OPTIMIZATION ON;
```

4. メディア管理ソフトウェアによる要求に応じて、バックアップを作成するためのテープ・チャネルを構成します。

```
CONFIGURE CHANNEL DEVICE TYPE SBT PARMS '<channel parameters>';
```

5. アーカイブ・ログを削除できる時期を CONFIGURE ARCHIVELOG DELETION POLICY コマンドを使用して指定します。

   ログはスタンバイ・サイトでバックアップされるため、ログ削除ポリシーに対して BACKED UP オプションを構成することをお薦めします。

   関連項目： アーカイブ REDO ログの削除ポリシーを有効に設定する方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

11.3.4 バックアップが実行されないスタンバイでの Recovery Manager 構成

次の Recovery Manager 構成は、バックアップが実行されないスタンバイ・データベースで推奨されます。

1. Recovery Manager をターゲットとしてスタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. スタンバイ・データベースで適用された後のアーカイブ・ログの自動削除を有効にします。

```
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
```

11.4 バックアップ手順

この項では、Data Guard 構成内の Oracle Database のバックアップに使用される Recovery Manager スクリプトおよび手順について説明します。次の項目で構成されています。

- テープ・バックアップ用キャッシュとしてのディスクの使用
- テープへの直接バックアップの実行

注意： Oracle の Maximum Availability Architecture（MAA）ベスト・プラクティスでは、プライマリ・データベースとスタンバイ・データベースの両方が停止した場合や、スイッチオーバーおよびフェイルオーバーについてサイトのプラクティスを新たに導入しなくてもよいようにする場合に、プライマリ・データベースとスタンバイ・データベースの両方でバックアップを作成して MTTR を短縮することをお薦めしています。

サーバー・パラメータ・ファイルのバックアップ

Oracle Database 11*g* までは、サーバー・パラメータ・ファイル（SPFILE）のバックアップは、他のスタンバイ・データベースで使用できると想定されていました。しかし、実際には、すべてのスタンバイ・データベースで同じ SPFILE を使用することは不可能です。この問題に対応するため、Recovery Manager では、あるデータベース・サイトで作成された SPFILE バックアップを別のデータベース・サイトで使用できないようにしています。この制約は、COMPATIBLE 初期化パラメータが 11.0.0 に設定されている場合にのみ実施されます。

スタンバイ・データベースでは、SPFILE のバックアップを除いたすべてのバックアップ操作をある特定のスタンバイ・データベースにオフロードできます。しかし、COMPATIBLE 初期化パラメータが 11.0.0 に設定されている場合、SPFILE をディスクにバックアップし、バックアップがテープに書き込まれるスタンバイ・サイトでは手動でカタログに追加できます。SPFILE バックアップ・セットに格納される追加メタデータにより、どのデータベースの SPFILE がどのバックアップ・セットに含まれるか Recovery Manager で識別できます。そのため、テープからリストアする際に、適切な SPFILE バックアップが選択されます。

11.4.1 テープ・バックアップ用キャッシュとしてのディスクの使用

スタンバイ・データベースのフラッシュ・リカバリ領域は、テープ・バックアップ用のディスク・キャッシュとして機能できます。ディスクはバックアップ用の一次記憶域として使用され、テープは長期間のアーカイブ記憶域となります。テープの増分バックアップは毎日、テープの全体バックアップは毎週作成されます。これらのバックアップの実行に使用されるコマンドは、次の各項を参照してください。

11.4.1.1 ディスクをキャッシュとして使用する日次テープ・バックアップ用のコマンド

バックアップ計画を決定する際に、日次増分バックアップを利用することをお薦めします。データファイル・イメージ・コピーは、最新の増分バックアップを使用してロールフォワードできるため、常に最新のデータファイル・イメージ・コピーを提供できます。Recovery Manager では、そのシステム変更番号（SCN）で取得された全体イメージ・コピーを使用するように、結果のイメージ・コピーをメディア・リカバリに使用するため、データベースの全体イメージ・コピーを毎日実行するオーバーヘッドはありません。もう 1 つのメリットは、イメージ・コピーが最新のブロック変更で更新され、現在の状態にデータベースを戻すのに必要な REDO ログが少なくなるため、リカバリ時間が短縮されることです。

日次増分バックアップを実施するには、1 日目にデータベースの全体バックアップを作成し、2 日目に増分バックアップを作成します。アーカイブ REDO ログを使用すると、いずれの日のどの時点についてもデータベースをリカバリできます。3 日目以降は、前日の増分バックアップをデータファイル・コピーとマージし、当日の増分バックアップを作成して、最終日までのどの時点についても高速リカバリを可能にします。REDO ログを使用すると、当日のどの時点についてもデータベースをリカバリできます。

日次バックアップを実行するスクリプトは次のとおりです（最終行 DELETE ARCHIVELOG ALL は、フラッシュ・リカバリ領域をログの格納に使用しない場合にのみ必要です）。

```
RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;
RECOVER COPY OF DATABASE WITH TAG 'OSS';
BACKUP DEVICE TYPE DISK INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG 'OSS'
DATABASE;
BACKUP DEVICE TYPE SBT ARCHIVELOG ALL;
BACKUP BACKUPSET ALL;
DELETE ARCHIVELOG ALL;
```

スタンバイ制御ファイルは、制御ファイルの自動バックアップが有効に設定されているため、バックアップ操作の最後に自動的にバックアップされます。

次に、スクリプトの各コマンドの動作について説明します。

- RESYNC CATALOG FROM DB_UNIQUE_NAME ALL

  リカバリ・カタログに認識されている、Data Guard 設定内のその他すべてのデータベース・サイト（プライマリ・データベースおよび他のスタンバイ・データベース）の情報を再同期化します。RESYNC CATALOG FROM DB_UNIQUE_NAME が機能するには、Oracle Net サービス名を使用して Recovery Manager をターゲットに接続する必要があります。また、すべてのデータベースで同じパスワード・ファイルを使用する必要があります。

- RECOVER COPY OF DATABASE WITH TAG 'OSS'

  前日に取得したレベル 1 の増分バックアップを適用して、データベースのレベル 0 のコピーをロールフォワードします。この例のスクリプトでは、前日のレベル 1 の増分には OSS タグが付けられています。その増分は、BACKUP DEVICE TYPE DISK ... DATABASE コマンドによって生成されます。このコマンドが実行された 1 日目には、レベル 1 の増分がまだないため、ロールフォワードはありません。レベル 0 の増分が、BACKUP DEVICE TYPE DISK ... DATABASE コマンドによって生成されます。2 日目にも、レベル 0 の増分しかないため、ロールフォワードはありません。OSS タグが付けられたレベル 1 の増分が、BACKUP DEVICE TYPE DISK ... DATABASE コマンドによって生成されます。3 日目以降には、前日に作成された OSS タグ付きのレベル 1 の増分を使用して、ロールフォワードが実行されます。

- `BACKUP DEVICE TYPE DISK INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG 'OSS' DATABASE`

  新しいレベル 1 の増分バックアップを作成します。このコマンドが実行された 1 日目は、レベル 0 の増分となります。2 日目以降は、レベル 1 の増分となります。

- `BACKUP DEVICE TYPE SBT ARCHIVELOG ALL`

  所定の削除ポリシーに従って、アーカイブ・ログをテープにバックアップします。

- `BACKUP BACKUPSET ALL`

  増分バックアップの作成結果として作成されるバックアップ・セットをバックアップします。

- `DELETE ARCHIVELOG ALL`

  `CONFIGURE ARCHIVELOG DELETION POLICY` コマンドで設定されたログ削除ポリシーに従って、アーカイブ・ログを削除します。アーカイブ・ログがフラッシュ・リカバリ領域にある場合は、空きディスク領域がさらに必要になると自動的に削除されます。そのため、毎日ログを明示的に削除する場合にのみ、このコマンドを使用する必要があります。

11.4.1.2 ディスクをキャッシュとして使用する週次テープ・バックアップ用のコマンド

すべてのリカバリ関連ファイルをテープにバックアップするには、次のコマンドを週 1 回使用します。

```
BACKUP RECOVERY FILES;
```

これにより、ディスク上にある最新の増分バックアップ、イメージ・コピー・バックアップおよびアーカイブ・ログ・バックアップがすべてテープにバックアップされます。

11.4.2 テープへの直接バックアップの実行

Oralce の Media Management Layer（MML）API により、サード・パーティ・ベンダーは、メディア・マネージャ、Recovery Manager と連動して機能するソフトウェア、テープ・ドライブなどの順次メディア・デバイスへのバックアップを可能にするベンダーのハードウェアを構築できます。メディア・マネージャは、テープなどの順次メディアのロード、アンロードおよびラベル付けを処理します。Recovery Manager を順次メディア・デバイスとともに使用するには、Oracle Secure Backup またはサード・パーティのメディア管理ソフトウェアをインストールする必要があります。

デフォルトでは、次の手順を実行してテープへの直接バックアップを実行します。

1. Recovery Manager を（ターゲット・データベースとして）スタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. 次のように、`CONFIGURE` コマンドを実行します。

   ```
   CONFIGURE DEFAULT DEVICE TYPE TO SBT;
   ```

この例では、スタンバイ・データベースで全体バックアップが毎週、増分バックアップが毎日作成されます。

> **関連項目：** メディア・マネージャとともに使用するために Recovery Manager を構成する方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

11.4.2.1 テープに直接実行する日次バックアップ用のコマンド

次の手順を実行して、テープに日次バックアップを直接実行します。

1. Recovery Manager を（ターゲット・データベースとして）スタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. 次の Recovery Manager コマンドを実行します。

```
RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;
BACKUP AS BACKUPSET INCREMENTAL LEVEL 1 DATABASE PLUS ARCHIVELOG;
DELETE ARCHIVELOG ALL;
```

これらのコマンドは、Data Guard 環境内のその他すべてのデータベースの情報を再同期化します。また、すべてのアーカイブ・ログを含むデータベースのレベル 1 の増分バックアップを作成します。このスクリプトが実行された 1 日目に、レベル 0 のバックアップが検出されなければ、レベル 0 のバックアップが作成されます。

DELETE ARCHIVELOG ALL コマンドは、すべてのアーカイブ・ログ・ファイルがフラッシュ・リカバリ領域にない場合にのみ必要です。

11.4.2.2 テープに直接実行する週次バックアップ用のコマンド

週 1 日、次の手順を実行してテープに週次バックアップを直接実行します。

1. Recovery Manager を（ターゲット・データベースとして）スタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. 次の Recovery Manager コマンドを実行します。

```
BACKUP AS BACKUPSET INCREMENTAL LEVEL 0 DATABASE PLUS ARCHIVELOG;
DELETE ARCHIVELOG ALL;
```

これらのコマンドは、Data Guard 環境内のその他すべてのデータベースの情報を再同期化し、すべてのアーカイブ・ログを含むレベル 0 のデータベース・バックアップを作成します。

DELETE ARCHIVELOG ALL コマンドは、すべてのアーカイブ・ログ・ファイルがフラッシュ・リカバリ領域にない場合にのみ必要です。

11.5 Data Guard 環境でのデータベースの登録および登録解除

REGISTER DATABASE コマンドを使用して明示的に登録する必要があるのは、プライマリ・データベースのみです。この登録は、Recovery Manager をリカバリ・カタログに接続し、TARGET としてプライマリ・データベースに接続した後に行います。

新しいスタンバイは、スタンバイ・データベースに接続するか、または CONFIGURE DB_UNIQUE_NAME コマンドを使用して接続識別子を構成すると、リカバリ・カタログに自動的に登録されます。

特定のスタンバイ・データベースに関する情報を登録解除するには、UNREGISTER DB_UNIQUE_NAME コマンドを使用できます。スタンバイ・データベースを Data Guard 環境から完全に削除すると、同じ Data Guard 環境内の別のデータベースに接続した後に、リカバリ・カタログのデータベース情報も削除できます。登録解除されたデータベースと関連付けられたバックアップは、そのまま他のデータベースで使用できます。これらのバックアップは、CHANGE BACKUP RESET DB_UNIQUE_NAME コマンドを使用して既存の他のデータベースに関連付けることができます。

INCLUDING BACKUPS オプションを指定して UNREGISTER DB_UNIQUE_NAME コマンドを使用すると、削除されるデータベースに関連付けられたすべてのバックアップ・ファイルのメタデータもリカバリ・カタログから削除されます。

11.6 Data Guard 環境におけるレポート生成

FOR DB_UNIQUE_NAME 句を指定して Recovery Manager の LIST、REPORT および SHOW の各コマンドを使用すると、特定のデータベースに関する情報を表示できます。

たとえば、リカバリ・カタログに接続した後、次のコマンドを使用して DBID が 1625818158 のデータベースに関する情報を表示し、Data Guard 環境内のデータベースを出力できます。SET DBID コマンドは、TARGET としてデータベースに接続していない場合にのみ必要です。最後の 3 つのコマンドは、DB_UNIQUE_NAME が BOSTON のデータベースについて、アーカイブ・ログ、データベース・ファイル名および Recovery Manager の構成情報を出力します。

```
SET DBID 1625818158;
LIST DB_UNIQUE_NAME OF DATABASE;
LIST ARCHIVELOG ALL FOR DB_UNIQUE_NAME BOSTON;
REPORT SCHEMA FOR DB_UNIQUE_NAME BOSTON;
SHOW ALL FOR DB_UNIQUE_NAME BOSTON;
```

11.7 Data Guard 環境におけるバックアップ・メンテナンスの実行

Data Guard 環境のファイル（データファイル、アーカイブ・ログ、バックアップ・ピース、イメージ・コピー、プロキシ・コピー）は、DB_UNIQUE_NAME パラメータを使用してデータベースに関連付けられます。そのため、Data Guard 環境のデータベースごとに、DB_UNIQUE_NAME に指定された値は一意であることが重要です。この情報は、ファイル共有属性とともに、Recovery Manager の様々な操作の際にアクセスできるファイルを判別するのに使用されます。

ファイル共有属性は、ディスク上のファイルが関連付けられているデータベースでのみアクセスできることを指定します。これに対して、テープ上のファイルはすべて、全データベースによってアクセスできると想定されます。BACKUP や RESTORE などの Recovery Manager コマンドや他のメンテナンス・コマンドは、この前提に従って動作します。たとえば、データベースでイメージ・コピーのロールフォワード操作中、そのデータベースに関連付けられたイメージ・コピーのみがロールフォワードされます。さらに、イメージ・コピーのロールフォワードには、ディスク上のすべての増分バックアップとテープ上のすべての増分バックアップが使用されます。同様に、リカバリ操作中、データベースに関連付けられたディスク・バックアップのみとテープ上のファイルが、バックアップのソースとみなされます。

> **関連項目：** Recovery Manager コマンドの詳細は、『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

11.7.1 リカバリ・カタログのメタデータの変更

様々なオペランドを指定して Recovery Manager の CHANGE コマンドを使用すると、リカバリ・カタログのメタデータを変更できます。詳細は、次の各項を参照してください。

スタンバイ・データベース間でのファイルの関連付けの変更

RESET DB_UNIQUE_NAME オプションを指定して CHANGE コマンドを使用すると、Data Guard 環境内のあるデータベースから別のデータベースにファイルの関連付けを変更できます。CHANGE コマンドは、ディスク・バックアップまたはアーカイブ・ログをあるデータベースから別のデータベースに転送し、転送先のデータベースでそれらのファイルを使用するときに役立ちます。また、CHANGE コマンドは、FOR DB_UNIQUE_NAME オプションと RESET DB_UNIQUE_NAME TO オプションを使用すると、いずれのデータベースに直接接続しなくてもデータベース間でファイルの関連付けを変更できます。

データベースの DB_UNIQUE_NAME の変更

データベースの DB_UNIQUE_NAME 初期化パラメータの値を変更すると、Data Guard 環境でも同じ変更を行う必要があります。そのデータベース・インスタンスへの接続後、Recovery Manager のリカバリ・カタログは DB_UNIQUE_NAME の新旧両方の値を認識します。リカバリ・カタログ・スキーマ内で新旧の値に関する情報をマージするには、Recovery Manager の CHANGE DB_UNIQUE_NAME コマンドを使用する必要があります。Recovery Manager が変更後の DB_UNIQUE_NAME パラメータでインスタンスに接続されない場合は、CHANGE DB_UNIQUE_NAME コマンドを使用してリカバリ・カタログ・スキーマの DB_UNIQUE_NAME の名前を変更することもできます。たとえば、データベースのインスタンス・パラメータ値が BOSTON_A から BOSTON_B に変更された場合、ターゲット・データベースおよびリカバリ・カタログに接続した後に、Recovery Manager のプロンプトで次のコマンドを実行する必要があります。

```
CHANGE DB_UNIQUE_NAME FROM BOSTON_A TO BOSTON_B;
```

バックアップの使用不可化またはメタデータの削除

AVAILABLE、UNAVAILABLE、KEEP、UNCATALOG などのオプションを指定して CHANGE コマンドを使用すると、バックアップをリストアおよびリカバリの目的で使用可能または使用不可能にしたり、そのメタデータを保存または削除することができます。

> **関連項目：** Recovery Manager の CHANGE コマンドの詳細は、『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

11.7.2 アーカイブ・ログまたはバックアップの削除

DELETE コマンドを使用して、バックアップ・セット、イメージ・コピー、アーカイブ・ログ、プロキシ・コピーを削除します。特定のデータベースに関連付けられているファイルのみを削除するには、FOR DB_UNIQUE_NAME オプションを DELETE コマンドとともに使用する必要があります。

ファイルのメタデータは、現行のターゲット・データベースに関連付けられ、正常に削除されたすべてのファイル（またはどの既知のデータベースにも関連付けられていないファイル）について削除されます。ファイルが正常に削除できなかった場合は、FORCE オプションを使用してファイルのメタデータを削除します。

別のデータベースに関連付けられているファイルが正常に削除された場合、リカバリ・カタログのそのメタデータも削除されます。他のデータベースに関連付けられているファイルおよび正常に削除できなかったファイルは、DELETE コマンドの完了時に、ファイルが関連付けられているデータベースで同じ操作を実行するように求める指示とともに出力されます（ファイルはデータベースごとにグループ化されます）。FORCE オプションを使用して、この動作を上書きすることはできません。削除不可のファイルのメタデータを削除しても問題ないことが明確である場合は、CHANGE RESET DB_UNIQUE_NAME コマンドを使用してデータベースとのファイルの関連付けに対するメタデータを変更し、FORCE オプションを指定して DELETE コマンドを使用し、ファイルのメタデータを削除します。

> **関連項目：** Recovery Manager の DELETE コマンドの詳細は、『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

11.7.3 リカバリ・カタログのメタデータの検証

CROSSCHECK コマンドを使用してリカバリ・カタログ・スキーマのファイル・ステータスを検証および更新します。特定のデータベースに関連付けられたファイルを検証するには、FOR DB_UNIQUE_NAME オプションを CROSSCHECK コマンドとともに使用します。

現行のターゲット・データベースに関連付けられたすべてのファイル（またはどのデータベースにも関連付けられていないあらゆるファイル）のメタデータは、CROSSCHECK 操作の結果に応じて AVAILABLE または EXPIRED のマークが付けられます。

別のデータベースに関連付けられているファイルが正常に検査された場合、リカバリ・カタログのそのメタデータも AVAILABLE に変更されます。他のデータベースに関連付けられているファイルおよび正常に検査できなかったファイルは、CROSSCHECK コマンドの完了時に、ファイルが関連付けられているデータベースで同じ操作を実行するように求める指示とともに出力されます（ファイルはサイトごとにグループ化されます）。確認し、それでも使用不可のファイルのステータス・メタデータを変更する場合は、CHANGE RESET DB_UNIQUE_NAME コマンドを使用してデータベースとのファイルの関連付けに対するメタデータを変更し、CROSSCHECK コマンドを実行してステータス・メタデータを EXPIRED に更新します。

> **関連項目：** Recovery Manager の CROSSCHECK コマンドの詳細は、『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

11.8 Data Guard 環境におけるリカバリ例

次の各項の例では、バックアップを作成したのと同じシステムにテープからファイルをリストアすると仮定しています。ファイルを異なるシステムにリストアする必要がある場合は、そのシステムに対してチャネルを構成してから、リストア・コマンドおよびリカバリ・コマンドを実行する必要があります。存在しないデータベースの構成を設定するには、SET DBID コマンドおよび FOR DB_UNIQUE_NAME を指定した CONFIGURE コマンドを使用します。異なるシステムから Recovery Manager バックアップにアクセスする方法の詳細は、メディア管理のドキュメントを参照してください。

この項では、次の使用例について説明します。

- プライマリ・データベースのデータファイル消失からのリカバリ
- スタンバイ・データベースのデータファイル消失からのリカバリ
- スタンバイ制御ファイルの消失からのリカバリ
- プライマリ制御ファイルの消失からのリカバリ
- オンライン REDO ログ・ファイルの消失からのリカバリ
- プライマリ・データベースの不完全リカバリ

11.8.1 プライマリ・データベースのデータファイル消失からのリカバリ

プライマリ・データベースのデータファイル消失からは、バックアップを使用するか、スタンバイ・データベースのファイルを使用してリカバリできます。詳細は、次の各項を参照してください。

バックアップの使用

次の Recovery Manager コマンドを発行して、データファイルをリストアおよびリカバリします。プライマリ・データベースとリカバリ・カタログ・データベースの両方に接続する必要があります。

```
RESTORE DATAFILE n,m...;
RECOVER DATAFILE n,m...;
```

次の Recovery Manager コマンドを発行して、表領域をリストアおよびリカバリします。プライマリ・データベースとリカバリ・カタログ・データベースの両方に接続する必要があります。

```
RESTORE TABLESPACE tbs_name1, tbs_name2, ...
RECOVER TABLESPACE tbs_name1, tbs_name2, ...
```

スタンバイ・データベースのファイルの使用

Oracle 11*g* 現在、スタンバイ・データベースのファイルを使用して消失したファイルをリカバリできます。これは、スタンバイが最新の状態で、ネットワーク接続がスタンバイとプライマリ間のファイル・コピーをサポートするのに十分足りる場合に有効に機能します。

Recovery Manager を起動し、次の手順を実行してスタンバイからプライマリにデータファイルをコピーします。

1. ターゲット・データベースとしてスタンバイ・データベースに接続します。

   ```
   CONNECT TARGET sys@standby
   ```

   パスワードの入力を求められます。

   ```
   target database Password: password
   ```

2. 補助データベースとしてプライマリ・データベースに接続します。

   ```
   CONNECT AUXILIARY sys@primary
   ```

   パスワードの入力を求められます。

   ```
   target database Password: password
   ```

3. ネットワークを介してスタンバイ・ホストのデータファイルをプライマリ・ホスト上の場所にバックアップします。たとえば、/disk1/df2.dbf がスタンバイ・ホストのデータファイル 2 の名前だとします。/disk8/datafile2.dbf がプライマリ・ホストのデータファイル 2 の名前だとします。次のコマンドは、ネットワークを介してデータファイル 2 を /disk9/df2copy.dbf にコピーします。

   ```
   BACKUP AS COPY DATAFILE 2 AUXILIARY FORMAT '/disk9/df2copy.dbf';
   ```

4. 次のようにして、Recovery Manager クライアントを終了します。

   ```
   EXIT;
   ```

5. Recovery Manager を起動し、ターゲットとしてプライマリ・データベースに接続し、リカバリ・カタログに接続します。

   ```
   CONNECT TARGET sys@primary;
   target database Password: password

   CONNECT CATALOG rman@catdb;
   recovery catalog database Password: password
   ```

6. CATALOG DATAFILECOPY コマンドを使用して、Recovery Manager で使用できるようにこのデータファイル・コピーをカタログに追加します。

```
CATALOG DATAFILECOPY '/disk9/df2copy.dbf';
```

次に、SWITCH DATAFILE コマンドを使用してデータファイル・コピーを切り替え、/disk9/df2copy.dbf が現行のデータファイルになるようにします。

```
RUN {
  SET NEWNAME FOR DATAFILE 2 TO '/disk9/df2copy.dbf';
  SWITCH DATAFILE 2;
}
```

11.8.2 スタンバイ・データベースのデータファイル消失からのリカバリ

1 つ以上のデータファイルが消失した後にスタンバイ・データベースをリカバリするには、Recovery Manager の RESTORE DATAFILE コマンドを使用して、消失したファイルをバックアップからスタンバイ・データベースにリストアする必要があります。破損ファイルのリカバリに必要なアーカイブ REDO ログ・ファイルすべてにスタンバイ・データベースがディスク上でアクセス可能な場合は、REDO Apply を再開します。

リカバリに必要なアーカイブ REDO ログ・ファイルにディスク上でアクセスできない場合は、次の手順に従って Recovery Manager を使用し、リストアしたデータファイルをスタンバイ・データベースに最後に適用されたログよりも大きい SCN/ ログ順序番号までリカバリし、REDO Apply を再開して REDO データの適用を続行します。

1. SQL*Plus をスタンバイ・データベースに接続します。
2. SQL ALTER DATABASE ... 文を使用して、REDO Apply を停止します。
3. 別の端末で、Recovery Manager を起動し、スタンバイ・データベースとリカバリ・カタログ・データベースの両方に接続します（スタンバイ・インスタンスへの接続には TARGET キーワードを使用します）。
4. 次の Recovery Manager コマンドを発行して、スタンバイ・データベースでデータファイルをリストアおよびリカバリします。

```
RESTORE DATAFILE <n,m,...>;
RECOVER DATABASE;
```

表領域をリストアするには、Recovery Manager の 'RESTORE TABLESPACE *tbs_name1, tbs_name2, ...*' コマンドを使用します。

5. SQL*Plus のプロンプトで SQL ALTER DATABASE ... 文を使用して、REDO Apply を再開します。

関連項目： REDO Apply の開始および停止の詳細は、7.3 項および 7.4 項を参照してください。

11.8.3 スタンバイ制御ファイルの消失からのリカバリ

Oracle ソフトウェアでは、スタンバイ制御ファイルを多重化できます。スタンバイ制御ファイルが多重化されているかどうかを確認するには、次のように CONTROL_FILES 初期化パラメータを調べます。

```
SQL> SHOW PARAMETER CONTROL_FILES;
NAME                                 TYPE        VALUE
 ------------------------------------ ----------- ------------------------------
control_files                        string      <cfilepath1>,<cfilepath2>
```

多重スタンバイ制御ファイルの 1 つが消失しているかアクセスできないと、Oracle ソフトウェアはインスタンスを停止し、アラート・ログに次のメッセージを書き込みます。

```
ORA-00210: cannot open the specified controlfile
ORA-00202: controlfile: '/disk1/oracle/dbs/scf3_2.f'
ORA-27041: unable to open file
```

制御ファイルの元のコピーを消失したコピーに上書きコピーし、次の SQL 文を使用してスタンバイ・インスタンスを再起動できます。

```
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

RESTORE CONTROLFILE コマンドに続いて RECOVER DATABASE コマンドを実行すると、バックアップから制御ファイルをリストアできます。RECOVER DATABASE コマンドは、制御ファイル内のファイル名をそのデータベースに存在するファイルと一致するように自動的に修正し、データベースで最後に受信したログ順序番号までデータベースをリカバリします。

他の方法として、プライマリ・データベースから新しい制御ファイルを作成し、すべての多重化位置にコピーしてデータファイル名を手動で変更し、ディスク上に存在するファイルと一致させます。

11.8.4 プライマリ制御ファイルの消失からのリカバリ

Oracle ソフトウェアでは、プライマリ・データベースの制御ファイルを多重化できます。プライマリ・データベースで制御ファイルの 1 つを更新できない場合は、プライマリ・データベース・インスタンスが自動的に停止します。

RESTORE CONTROLFILE コマンドに続いて RECOVER DATABASE コマンドを実行すると、バックアップから制御ファイルをリストアできます。RECOVER DATABASE コマンドは、制御ファイル内のファイル名をそのデータベースに存在するファイルと一致するように自動的に修正し、データベースをリカバリします。

他の方法として、CREATE CONTROLFILE SQL コマンドを使用して、新しい制御ファイルを作成します。すべてのデータファイルおよびオンライン・ログが消失していなければ、制御ファイルの再作成は可能です。

> **関連項目：** Recovery Manager を使用して制御ファイルの消失からリカバリする方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

11.8.5 オンライン REDO ログ・ファイルの消失からのリカバリ

オンライン REDO ログ・ファイルは多重化することをお薦めします。オンライン REDO ログ・グループのメンバーがすべて消失すると、Oracle ソフトウェアはインスタンスを終了します。書込みできないのがログ・ファイル・グループの一部のメンバーのみの場合、そのメンバーはアクセス可能になるまで使用されません。V$LOGFILE および V$LOG ビューには、プライマリ・データベース・インスタンスのログ・ファイル・メンバーの現行のステータスに関する詳細情報が含まれます。

Oracle ソフトウェアがオンライン REDO ログ・ファイルのメンバーの 1 つに書込みできない場合は、次のアラート・メッセージが戻されます。

```
ORA-00313: open failed for members of log group 1 of thread 1
ORA-00312: online log 1 thread 1: '/disk1/oracle/dbs/t1_log1.f'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
```

アクセスの問題がハードウェア・エラーによる一時的なものの場合は、問題を解決すると処理が自動的に続行されます。消失が永続的な場合は、新規メンバーを追加して古いメンバーをグループから削除できます。

REDO ログ・グループに新規メンバーを追加するには、次の文を発行します。

```
SQL> ALTER DATABASE ADD LOGFILE MEMBER 'log_file_name' REUSE TO GROUP n;
```

この文は、データベースがオープン状態の場合にも、データベースの可用性に影響を与えずに発行できます。

アーカイブ済の非アクティブ・グループのメンバーがすべて消失した場合は、そのグループを削除して再作成できます。

他のすべての場合（現行の ACTIVE グループまたはアーカイブされていない非アクティブ・グループのオンライン・ログ・メンバーすべてが消失した場合）は、スタンバイ・データベースにフェイルオーバーする必要があります。フェイルオーバー手順の詳細は第 8 章を参照してください。

11.8.6 プライマリ・データベースの不完全リカバリ

プライマリ・データベースの不完全リカバリを実行するのは、通常、データベースが論理的に（ユーザーまたはアプリケーションが原因で）破損した場合や、表領域またはデータファイルがデータベースから意図せずに削除された場合などです。

スタンバイ・データベース・インスタンス上の現在のデータベース・チェックポイント SCN に応じて、次のいずれかの手順でプライマリ・データベースの不完全リカバリを実行できます。すべての手順は、最も所要時間が短いものから優先順位順になっています。

フラッシュバック・データベースの使用　フラッシュバック・データベースの使用は、プライマリ・データベースでフラッシュバック・データベース機能が使用可能で、データベース・ファイルが 1 つも消失しておらず、Point-in-Time リカバリが最も古いフラッシュバック SCN より大きいか、最も早いフラッシュバック時間よりも後の場合の推奨手順です。フラッシュバック・データベースを使用して Point-in-Time リカバリを実行する手順の詳細は、13.3 項を参照してください。

スタンバイ・データベース・インスタンスの使用　これは、スタンバイ・データベースが必要な不完全リカバリ時間より後のもので、プライマリまたはスタンバイ・データベースでフラッシュバック・データベースが使用可能でない場合の推奨手順です。

1. スタンバイ・データベースを必要な時点までリカバリします。

   ```
   RECOVER DATABASE UNTIL TIME 'time';
   ```

   あるいは、SCN またはログ順序番号を使用して不完全リカバリ時間を指定できます。

   ```
   RECOVER DATABASE UNTIL SCN incomplete recovery SCN';
   RECOVER DATABASE UNTIL LOGSEQ incomplete recovery log sequence number THREAD thread
   number;
   ```

2. スタンバイ・データベースを読取り専用モードでオープンし、データベースの状態を確認します。

   必要な状態になっていない場合は、LogMiner ユーティリティを使用してアーカイブ REDO ログ・ファイルを調べ、不完全リカバリに適切なターゲット時刻または SCN を確認します。または、ターゲット時刻より前の判明している時点までスタンバイ・データベースをリカバリしてから、データベースを読取り専用モードでオープンし、データの状態を検査することもできます。データベースの状態が正常であると確認されるまで、このプロセスを繰り返します。データベースのリカバリ終了時点が遅すぎると（つまり、エラーが発生した SCN より後までリカバリすると）、それより前の SCN に戻せないことに注意してください。

3. SQL `ALTER DATABASE ACTIVATE STANDBY DATABASE` 文を使用してスタンバイ・データベースをアクティブ化します。これにより、スタンバイ・データベースがプライマリ・データベースに変換されて、新しいリセットログ・ブランチが作成され、データベースがオープンします。新規リセットログ・ブランチへのスタンバイ・データベースの対応の詳細は、9.4 項を参照してください。

プライマリ・データベース・インスタンスの使用　すべてのスタンバイ・データベース・インスタンスがすでに必要な時点より後までリカバリされており、プライマリまたはスタンバイ・データベースでフラッシュバック・データベースが使用できない場合は、この方法で操作する必要があります。

次の手順に従ってプライマリ・データベースで不完全リカバリを実行します。

1. LogMiner または他の手段を使用して、データベースのすべてのデータが正常と判明している時点または SCN を識別します。

2. その時点または SCN を使用して次の Recovery Manager コマンドを発行し、不完全データベース・リカバリを実行し、`RESETLOGS` オプションを使用して（カタログ・データベースと `MOUNT` 状態のプライマリ・インスタンスに接続した後で）データベースをオープンします。

   ```
   RUN
   {
   SET UNTIL TIME 'time';
   RESTORE DATABASE;
   RECOVER DATABASE;
   }
   ALTER DATABASE OPEN RESETLOGS;
   ```

この処理の後に、Data Guard 構成内ですべてのスタンバイ・データベース・インスタンスを再確立する必要があります。

11.9 その他のバックアップ状況

次の各項では、スタンバイ・データベースとプライマリ・データベースでバックアップ・ファイルを共有できない場合、REDO ログ・ファイルのリモート・アーカイブにスタンバイ・インスタンスのみが使用される場合、またはスタンバイ・データベースのファイル名がプライマリ・データベースとは異なる場合など、その他の構成にあわせてバックアップ処理を変更する方法について説明します。

11.9.1 スタンバイ・データベースが地理的に離れすぎているためにバックアップを共有できない場合

スタンバイ・データベース間の距離が離れている場合、プライマリ・システムや他のスタンバイ・システムからは、これらの上で作成されたバックアップに簡単にアクセスできません。すべてのシステム上でデータベースの完全バックアップを実行して、リカバリ操作を実行します。フラッシュ・リカバリ領域は、プライマリおよびスタンバイ・システム上にローカルに置くことができます（つまり、プライマリ・データベースとスタンバイ・データベースでフラッシュ・リカバリ領域が同一である必要がありません）。

この使用例でも、11.8 項で説明した一般的な対策を使用できますが、次の例外があります。

- Recovery Manager で作成されるバックアップ・ファイルには、タグとしてローカル・システム名を指定し、そのタグを RESTORE 操作で使用して、Recovery Manager による同じホスト上で作成されたバックアップの選択を制限する必要があります。つまり、バックアップの作成時に、BACKUP コマンドでは TAG *system name* オプションを、RESTORE コマンドでは FROM TAG *system name* オプションを、RECOVER コマンドでは FROM TAG *system name* ARCHIVELOG TAG *system name* オプションをそれぞれ使用する必要があります。
- スタンバイ・サイトの障害時リカバリを実行する手順は、次のとおりです。

  1. スタンバイの操作に使用されていたのと同じパラメータ・ファイルを使用して、スタンバイ・インスタンスを NOMOUNT 状態で起動します。
  2. プライマリ・インスタンスで、SQL ALTER DATABASE CREATE STANDBY CONTROLFILE AS *filename* 文を使用してスタンバイ制御ファイルを作成し、作成された制御ファイルを使用してスタンバイ・インスタンスをマウントします。
  3. 次の Recovery Manager コマンドを発行し、データベース・ファイルをリストアおよびリカバリします。

     ```
     RESTORE DATABASE FROM TAG 'system name';
     RECOVER DATABASE FROM TAG 'system name' ARCHIVELOG TAG 'system name';
     ```

  4. REDO Apply を再開する。

スタンバイ・インスタンスにより残りのアーカイブ REDO ログ・ファイルがフェッチされます。

11.9.2 FAL サーバーとして使用されるスタンバイ・データベースにデータファイルが含まれていない場合

11.4 項で説明した手順を使用しますが、データベース・ファイルをバックアップする Recovery Manager コマンドは FAL サーバーに対して実行できません。FAL サーバーは、すべてのアーカイブ REDO ログ・ファイルのバックアップ・ソースとして使用できるため、アーカイブ REDO ログ・ファイルのバックアップを FAL サーバーにオフロードします。

11.9.3 スタンバイ・データベースのファイル名がプライマリ・データベースとは異なる場合

注意： Oracle Database 11*g* 現在、リカバリ・カタログは各スタンバイ・データベース・サイトのファイル名を再同期化できます。しかし、スタンバイ・データベースのファイル名がなんらかの理由でまったく再同期化されていなかった場合は、この項で説明する手順に従って再同期化することが可能です。

まったく再同期化されていなかったプライマリ・データベースとスタンバイ・データベースでデータベース・ファイル名が異なる場合は、使用する RESTORE および RECOVER コマンドが若干異なります。スタンバイ・データベースで実際のデータファイル名を取得するには、V$DATAFILE ビューを問い合せ、データベースのすべてのデータファイルに対して SET NEWNAME オプションを指定します。

```
RUN
{
SET NEWNAME FOR DATAFILE 1 TO 'existing file location for file#1 from V$DATAFILE';
SET NEWNAME FOR DATAFILE 2 TO 'existing file location for file#2 from V$DATAFILE';
…
…
 SET NEWNAME FOR DATAFILE n TO 'existing file location for file#n from V$DATAFILE';
 RESTORE {DATAFILE <n,m,…> | TABLESPACE tbs_name_1, 2, …| DATABASE;
SWITCH DATAFILE ALL;
RECOVER DATABASE {NOREDO};
}
```

同様に、Recovery Manager の DUPLICATE コマンドでも SET NEWNAME オプションを使用し、スタンバイ・データベースの作成中に新規ファイル名を指定する必要があります。あるいは、LOG_FILE_NAME_CONVERT および DB_FILE_NAME_CONVERT パラメータを設定できます。

11.10 Recovery Manager の増分バックアップによるフィジカル・スタンバイ・データベースのロールフォワード

Recovery Manager の増分バックアップを使用して、フィジカル・スタンバイ・データベースをプライマリ・データベースと同期化できる場合があります。Recovery Manager の BACKUP INCREMENTAL FROM SCN コマンドを使用すると、プライマリ・データベースでスタンバイの現行 SCN から始まるバックアップを作成し、それを使用してスタンバイ・データベースをロールフォワードできます。

この項で説明する手順は、フィジカル・スタンバイ・データベースが次のいずれかに該当するため、Recovery Manager の増分バックアップが役立つ状況に適用します。

- プライマリ・データベースから大幅に遅れている
- 広範囲にロギングなしの変更がある
- データファイルのサブセットにロギングなしの変更がある

注意： この操作の実行時にはリカバリ・カタログの使用をお薦めします。これらの手順は、リカバリ・カタログがなくても可能ですが、リストアされた制御ファイルのファイル名を慎重に修正する必要があります。

関連項目： Recovery Manager の増分バックアップの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

11.10.1 Recovery Manager の増分バックアップを使用するための手順

特に記載していないかぎり、次の手順は前述の 3 つの状況すべてに適用されます。

1. スタンバイ・データベースで REDO Apply を停止します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

2. スタンバイ・データベースで、増分バックアップの FROM SCN を計算します。この計算は、状況によって異なります。

   - プライマリ・データベースから大幅に遅れているスタンバイでは、V$DATABASE ビューを問い合せてスタンバイ・データベースの現行 SCN を記録します。

```
SQL> SELECT CURRENT_SCN FROM V$DATABASE;
CURRENT_SCN
-----------
     233995
```

   - 広範囲にロギングなしの変更があるスタンバイでは、V$DATAFILE ビューを問い合せて最小の FIRST_NONLOGGED_SCN を記録します。

```
SQL> SELECT MIN(FIRST_NONLOGGED_SCN) FROM V$DATAFILE
  2> WHERE FIRST_NONLOGGED_SCN>0;

MIN(FIRST_NONLOGGED_SCN)
------------------------
                  223948
```

   - データファイルのサブセットにロギングなしの変更があるスタンバイでは、次のように V$DATAFILE ビューを問い合せます。

```
SQL> SELECT FILE#, FIRST_NONLOGGED_SCN FROM V$DATAFILE
  2> WHERE FIRST_NONLOGGED_SCN > 0;

FILE#      FIRST_NONLOGGED_SCN
---------- -------------------
         4              225979
         5              230184
```

3. Recovery Manager ターゲットとしてプライマリ・データベースに接続し、手順 2 で記録したスタンバイ・データベースの現行 SCN（プライマリから大幅に遅れているスタンバイの場合）または最小の FIRST_NONLOGGED_SCN（広範囲にロギングなしの変更があるスタンバイの場合）から増分バックアップを作成します。

```
RMAN> BACKUP INCREMENTAL FROM SCN 233995 DATABASE FORMAT '/tmp/ForStandby_%U' tag
'FORSTANDBY';
```

   データファイルのサブセットにロギングなしの変更があるスタンバイの場合は、次のように、FIRST_NONLOGGED_SCN 列に表示されたデータファイル（手順 2 で記録）ごとに増分バックアップを作成します。

```
RMAN> BACKUP INCREMENTAL FROM SCN 225979 DATAFILE 4 FORMAT '/tmp/ForStandby_%U' TAG
'FORSTANDBY';
RMAN> BACKUP INCREMENTAL FROM SCN 230184 DATAFILE 5 FORMAT '/tmp/ForStandby_%U' TAG
'FORSTANDBY';
```

4. バックアップが共有記憶域に書き込まれた場合は、この手順をスキップします。それ以外の場合は、プライマリ・システムに作成されたすべてのバックアップ・セットを、スタンバイ・システムに転送し、カタログに追加します。複数のバックアップ・ファイルが作成されている場合があります。次の例では、オペレーティング・システムのプロンプトで、scp コマンドを使用してファイルをコピーしています。

   ```
   scp /tmp/ForStandby_* standby:/tmp
   ```

   次に、Recovery Manager のプロンプトで、次のコマンドを入力してカタログに追加します。

   ```
   RMAN> CATALOG START WITH '/tmp/ForStandby';
   ```

5. Recovery Manager ターゲットとしてスタンバイ・データベースに接続し、REPORT SCHEMA 文を実行してスタンバイ・データベース・サイトが自動的に登録され、スタンバイ・サイトにファイル名が表示されることを確認します。

   ```
   RMAN> REPORT SCHEMA;
   ```

6. Recovery Manager ターゲットとしてスタンバイ・データベースに接続し、増分バックアップを適用します。次のように実行します。

   ```
   RMAN> STARTUP FORCE NOMOUNT;
   RMAN> RESTORE STANDBY CONTROLFILE FROM TAG 'FORSTANDBY';
   RMAN> ALTER DATABASE MOUNT;
   RMAN> RECOVER DATABASE NOREDO;
   ```

   注意： リカバリ・カタログの使用をお薦めしますが、使用しない場合は、RECOVER コマンドを発行する直前に制御ファイル内のファイル名を編集するか、Recovery Manager の SET NEWNAME コマンドを使用してデータファイル名を割り当てる必要があります。

7. 広範囲にロギングなしの変更があるスタンバイまたはデータファイルのサブセットにロギングなしの変更があるスタンバイで、V$DATAFILE ビューを問い合せ、ロギングなしの変更が含まれるデータファイルがないことを確認します。次の問合せでは 0（ゼロ）行が戻されます。

   ```
   SQL> SELECT FILE#, FIRST_NONLOGGED_SCN FROM V$DATAFILE
     2> WHERE FIRST_NONLOGGED_SCN > 0;
   ```

   注意： 増分バックアップは 7 日で不要になります。あるいは、この時点で Recovery Manager の DELETE コマンドを使用して削除できます。

8. スタンバイ制御ファイルを再作成します。
9. フィジカル・スタンバイ・データベースで REDO Apply を開始します。

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
     2> USING CURRENT LOGFILE DISCONNECT FROM SESSION;
   ```

12

SQL Apply を使用した Oracle Database のアップグレード

Oracle Database 10*g* リリース 1（10.1.0.3）からは、ロジカル・スタンバイ・データベースを使用して、Oracle Database 10*g* ソフトウェアのローリング・アップグレードを実行できます。ローリング・アップグレード中は、プライマリ・データベースとロジカル・スタンバイ・データベースで異なるリリースの Oracle データベースを実行しながら、プライマリ・データベースの停止時間を最短に抑えて各リリースを一度に 1 つずつアップグレードできます。

注意： この章では、より停止時間が長い通常のアップグレード手順（付録 B「Data Guard 構成におけるデータベースのアップグレード」を参照）にかわる方法について説明します。この章で説明する方法の手順と、付録 B の手順を組み合せようとしないでください。

この章の各手順では、Oracle データベースのアップグレード中の停止時間を最短に抑える方法について説明します。この章は、次の項目で構成されています。

- SQL Apply を使用するローリング・アップグレードのメリット
- SQL Apply を使用してローリング・アップグレードを実行するための要件
- アップグレード手順に使用する図と表記規則
- ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行
- 既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行
- 既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行

12.1 SQL Apply を使用するローリング・アップグレードのメリット

SQL Apply を使用してローリング・アップグレードを実行する方法には、次のように複数のメリットがあります。

- データベースの停止時間が最短で済みます。停止時間全体がスイッチオーバーの実行所要時間と同様に短時間です。
- PL/SQL の再コンパイルによるアプリケーションの停止時間は発生しません。
- プライマリ・データベースに影響を与えずに、アップグレード後のデータベース・リリースを検証できます。

12.2 SQL Apply を使用してローリング・アップグレードを実行するための要件

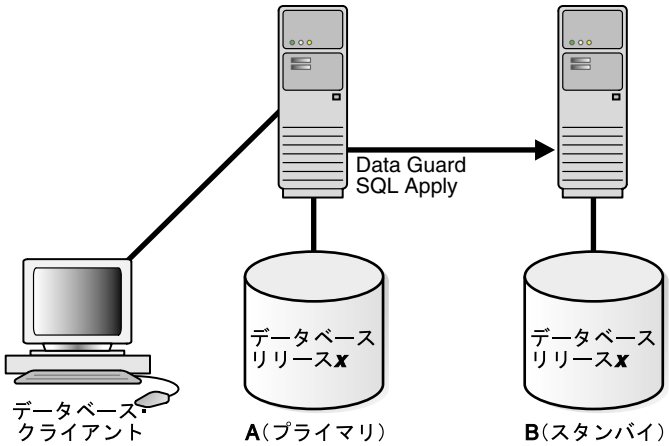
ローリング・アップグレード手順の要件は、次のとおりです。

- Oracle Database リリース *x* を実行中のプライマリ・データベースと Oracle Database リリース *y* を実行中のロジカル・スタンバイ・データベース。
- データベースが Data Guard Broker 構成に含まれていないこと。Broker 構成からデータベースを削除する方法の詳細は、『Oracle Data Guard Broker』を参照してください。
- Data Guard の保護モードが最大可用性モードまたは最大パフォーマンス・モードに設定されていること。現在の保護モード設定を確認するには、V$DATABASE ビューの PROTECTION_LEVEL 列を問い合せます。
- ロジカル・スタンバイ・データベースのアップグレード中もプライマリ・データベースで処理を進行できるようにするために、ロジカル・スタンバイ・データベースの宛先の LOG_ARCHIVE_DEST_*n* 初期化パラメータが MANDATORY に設定されていないこと。
- COMPATIBLE 初期化パラメータがアップグレード前のソフトウェア・リリースと一致していること。つまり、リリース *x* からリリース *y* へのローリング・アップグレードでは、プライマリ・データベースとスタンバイ・データベースの両方で COMPATIBLE 初期化パラメータをリリース *x* に設定する必要があります。

12.3 アップグレード手順に使用する図と表記規則

図 12-1 に、アップグレード開始前の Data Guard 構成を示します。この例では、プライマリ・データベースとロジカル・スタンバイ・データベースで同じリリースの Oracle Database ソフトウェアが実行されています。

図 12-1 アップグレード前の Data Guard 構成



アップグレード処理中に、Data Guard 構成は何度か混合データベース・リリースで動作します。リリース間にまたがるデータ保護は使用できません。これらの手順では、データ保護を提供するために Data Guard 構成に第 2 のスタンバイ・データベースがある場合を考えます。

アップグレード手順と図では、データベースを「プライマリ・データベース」と「スタンバイ・データベース」ではなく「データベース A」および「データベース B」と呼びます。これは、アップグレード手順の実行中にデータベースのロールが切り替わるためです。最初は、図 12-1 に示すようにデータベース A がプライマリ・データベースでデータベース B がロジカル・スタンバイ・データベースです。

次の各項では、SQL Apply のローリング・アップグレード手順を使用できる例について説明します。

- 12-4 ページの 12.4 項「ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行」
- 12-6 ページの 12.5 項「既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行」
- 12-13 ページの 12.6 項「既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行」

12.4 ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行

この使用例では、既存の Data Guard 構成がなく、Oracle Database のローリング・アップグレードを実行するためだけにロジカル・スタンバイ・データベースを作成することを前提にしています。

プライマリ・データベースとスタンバイ・データベースをアップグレードできるように準備する手順は、次のとおりです。

手順 1 サポートされないデータ型および記憶域属性を識別する

プライマリ・データベースでサポートされないデータベース・オブジェクトを識別し、その処理方法を決定する手順は、次のとおりです。

1. 表についてサポートされないデータ型および記憶域属性を識別します。
   - 付録 C「ロジカル・スタンバイ・データベースでサポートされるデータ型および DDL」を参照し、サポートされるデータ型および記憶域属性のリストを確認します。
   - プライマリ・データベースで DBA_LOGSTDBY_UNSUPPORTED および DBA_LOGSTDBY_SKIP ビューを問い合せます。プライマリ・データベースで表示された表およびスキーマに対して行われた変更は、ロジカル・スタンバイ・データベースに適用されません。次の問合せを実行すると、サポートされない表のリストの例が表示されます。

```
SQL> SELECT DISTINCT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED;
OWNER      TABLE_NAME
---------- -----------------
OE         CATEGORIES_TAB
OE         CUSTOMERS
OE         WAREHOUSES
PM         ONLINE_MEDIA
PM         PRINT_MEDIA
SCOTT      MYCOMPRESS
SH         MVIEW$_EXCEPTIONS
7 rows selected.

SQL>
SQL> SELECT OWNER FROM DBA_LOGSTDBY_SKIP
  2   WHERE STATEMENT_OPT = 'INTERNAL SCHEMA';

OWNER
------------------------------
CTXSYS
DBSNMP
DIP
ORDPLUGINS
ORDSYS
OUTLN
SI_INFORMTN_SCHEMA
SYS
SYSTEM
WMSYS
10 rows selected.
```

2. サポートされない表の処理方法を決定します。

   サポートされないオブジェクトをプライマリ・データベースで変更している場合、アップグレード手順の実行所要時間中、サポートされない表に対する変更を一時的に停止することで、アップグレードを実行できるようになる可能性があります。

   サポートされない表に対する変更を防止できる場合は、SQL Apply を使用してアップグレード手順を実行できる場合があります。この方法では、ロジカル・スタンバイ制御ファイルの作成時からアップグレード完了時までは、サポートされない表に対するユーザー変更を禁止する必要があります。たとえば、給与管理部門はオブジェクト表を更新しますが、この部門がデータベースを更新するのは月曜から金曜のみであるとします。一方、カスタマ・サービス部門は 1 日 24 時間、1 週 7 日間のデータベース・アクセスを必要としますが、使用するのはサポートされるデータ型および表のみであるとします。この使用例では、週末にアップグレードを実行できます。DBA_LOGSTDBY_EVENTS ビューでトランザクション・アクティビティを監視し、最初のスイッチオーバーの実行時までアップグレードを中断できます（必要な場合）。

   サポートされない表に対する変更をアップグレード中に防止できない場合、発生したサポートされないトランザクションはロジカル・スタンバイ・データベースの DBA_LOGSTDBY_EVENTS 表に記録されます。アップグレードの完了後に、Oracle Data Pump またはエクスポート / インポート・ユーティリティを使用して、変更された表をアップグレード後のデータベースにインポートできます。

   変更された表のサイズにより、データベース操作が使用できなくなる期間が決定するため、データをエクスポートしてスタンバイ・データベースにインポートするには表が大きすぎないかどうかを判断する必要があります。たとえば、4TB の表は、エクスポート / インポート処理に適した候補ではありません。

   注意： アプリケーションのデータ型がサポートされないためにロジカル・スタンバイ・データベースを使用できない場合は、『Oracle Database アップグレード・ガイド』の説明に従ってアップグレードを実行してください。

手順 2 ロジカル・スタンバイ・データベースを作成する

ロジカル・スタンバイ・データベースを作成するには、第 4 章の手順に従います。

停止時間を最短に抑えるために、ロジカル・スタンバイ・データベースでスタンバイ REDO ログを構成することをお薦めします。

手順 3 ローリング・アップグレードを実行する

ロジカル・スタンバイ・データベースを作成したので、12.5 項「既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行」で説明している手順を実行できます。この手順では、ロジカル・スタンバイで同じ Oracle ソフトウェアを実行していることを前提としています。

12.5 既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行

この項では、ロジカル・スタンバイ・データベースとプライマリ・データベースをアップグレードする手順について説明します。表 12-1 に手順を示します。

表 12-1 既存のロジカル・スタンバイを使用したローリング・アップグレードの実行手順

手順	説明
1	ローリング・アップグレードの準備をする
2	ロジカル・スタンバイ・データベースをアップグレードする
3	サポートされない表に関する情報を取得する
4	アップグレード後のロジカル・スタンバイ・データベースで SQL Apply を再開する
5	アップグレード後のスタンバイ・データベースでイベントを監視する
6	スイッチオーバーを開始する
7	アップグレード中に変更されたすべての表をインポートする
8	スイッチオーバーを完了してユーザー・アプリケーションをアクティブ化する
9	古いプライマリ・データベースをアップグレードする
10	古いプライマリ・データベースで SQL Apply を開始する
11	必要な場合は、両方のデータベースの互換性レベルを上げる
12	新規ロジカル・スタンバイ・データベースでイベントを監視する
13	必要な場合は、再度スイッチオーバーを実行する

手順 1 ローリング・アップグレードの準備をする

次の手順に従って、Oracle ソフトウェアのローリング・アップグレードの実行準備をします。

1. ロジカル・スタンバイ・データベース（データベース B）で次の文を発行し、SQL Apply を停止します。

   ```
   SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   ```

2. 必要に応じて、互換性を最高値に設定します。

   COMPATIBLE 初期化パラメータに、アップグレード前のプライマリ・データベースで実行されている Oracle Database ソフトウェアのリリース番号が指定されていることを確認します。

   たとえば、プライマリ・データベースでリリース 10.1 が実行されている場合は、COMPATIBLE 初期化パラメータを両方のデータベースで 10.1 に設定します。COMPATIBLE 初期化パラメータは、必ずスタンバイ・データベースで設定してからプライマリ・データベースで設定してください。

手順 2 ロジカル・スタンバイ・データベースをアップグレードする

ロジカル・スタンバイ・データベース（データベース B）の Oracle Database ソフトウェアをリリース *y* にアップグレードします。ロジカル・スタンバイ・データベースでは、アップグレード中、プライマリ・データベースからの REDO データを受け入れません。

手順 3 サポートされない表に関する情報を取得する

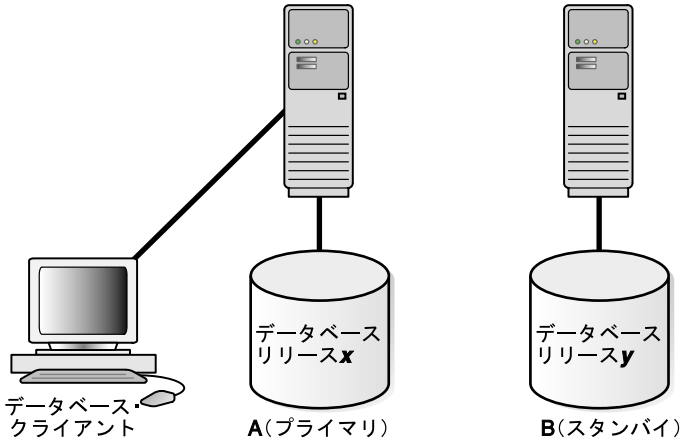
データベース B で、PL/SQL プロシージャ DBMS_LOGSTDBY を使用して、プライマリ・データベースで実行中の、ロジカル・スタンバイ・データベースではサポートされないトランザクションに関する情報を取得します。次のプロシージャを実行して情報を取得し、DBA_LOGSTDBY_EVENTS 表にイベントとして記録します。

```
EXEC DBMS_LOGSTDBY.APPLY_SET('MAX_EVENTS_RECORDED',DBMS_LOGSTDBY.MAX_EVENTS);
EXEC DBMS_LOGSTDBY.APPLY_SET('RECORD_UNSUPPORTED_OPERATIONS', 'TRUE');
```

Oracle Database ソフトウェアをアップグレードするには、該当する Oracle Database リリースの『Oracle Database アップグレード・ガイド』を参照してください。

図 12-2 に、リリース *x* を実行中のデータベース A とリリース *y* を実行中のデータベース B を示します。アップグレード中は、REDO データがプライマリ・システムに蓄積されます。

図 12-2 ロジカル・スタンバイ・データベースのリリースのアップグレード



関連項目：

- DBA_LOGSTDBY_EVENTS ビューの詳細は、10-16 ページの 10.5.1 項「DBA_LOGSTDBY_EVENTS ビューでのイベントのロギングのカスタマイズ」を参照してください。
- DBMS_LOGSTDBY パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

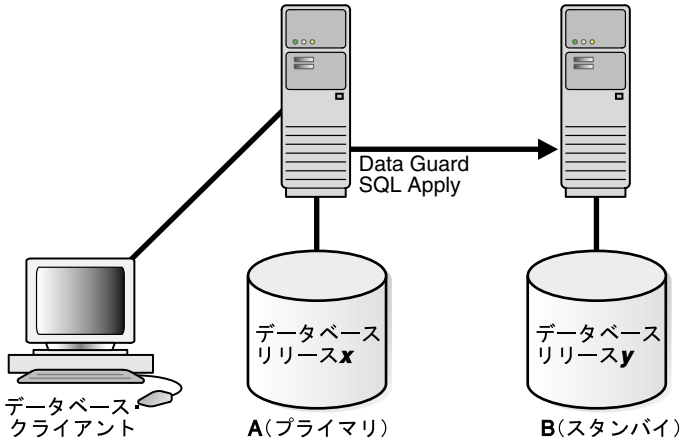
手順 4 アップグレード後のロジカル・スタンバイ・データベースで SQL Apply を再開する

SQL Apply を再開し、データベース A ではリリース *x*、データベース B ではリリース *y* で動作させます。SQL Apply を開始するには、データベース B で次の文を発行します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

プライマリ・システムに蓄積された REDO データは、新しくアップグレードしたロジカル・スタンバイ・データベースに自動的に転送され、適用されます。Data Guard 構成では、アップグレード後の Oracle Database ソフトウェア・リリースが本番環境で正常に実行されるかどうかを確認するために、任意の期間だけ図 12-3 に示す混合リリースを実行できます。

図 12-3 混合リリースの実行



データベース B がデータベース A とどの程度迅速に一致するかを監視するには、次のようにデータベース B で V$LOGSTDBY_PROGRESS ビューを問い合せます。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YY HH24:MI:SS';
Session altered.

SQL> SELECT SYSDATE, APPLIED_TIME FROM V$LOGSTDBY_PROGRESS;

SYSDATE            APPLIED_TIME
------------------ ------------------
27-JUN-05 17:07:06 27-JUN-05 17:06:50
```

手順 5 アップグレード後のスタンバイ・データベースでイベントを監視する

頻繁に DBA_LOGSTDBY_EVENTS ビューを問い合せ、データベース B に適用されなかった DDL 文と DML 文があるかどうかを確認する必要があります。例 12-1 に、イベントの監視により 2 つのデータベースにおける潜在的な差異を把握する方法を示します。

例 12-1 DBA_LOGSTDBY_EVENTS を使用したイベントの監視

```
SQL> SET LONG 1000
SQL> SET PAGESIZE 180
SQL> SET LINESIZE 79
SQL> SELECT EVENT_TIMESTAMP, EVENT, STATUS FROM DBA_LOGSTDBY_EVENTS
ORDER BY EVENT_TIMESTAMP;

EVENT_TIMESTAMP
-------------------------------------------------------------------------------
EVENT
-------------------------------------------------------------------------------
STATUS
-------------------------------------------------------------------------------
…
24-MAY-05 05.18.29.318912 PM
CREATE TABLE SYSTEM.TST (one number)
ORA-16226: DDL skipped due to lack of support

24-MAY-05 05.18.29.379990 PM
"SYSTEM"."TST"
ORA-16129: unsupported dml encountered
```

前述の例で、各エラーの意味は次のとおりです。

- ORA-16226 エラーは、サポートできない DDL 文を示します。この場合は、内部スキーマに属しているためにサポートできません。
- ORA-16129 エラーは、適用されなかった DML 文を示します。

この種のエラーは、データベース A で発生した変更の一部がデータベース B に適用されなかったことを示します。この時点で、アップグレード手順を続行するかどうかを判断する必要があります。ロジカル・スタンバイ・データベースとプライマリ・データベース間で、この差異が許容可能であることが確実な場合は、アップグレード手順を続行します。不確実な場合は、アップグレード手順を中断してデータベース B を再インスタンス化し、別の時点でアップグレード手順を実行します。

手順 6 スイッチオーバーを開始する

アップグレード後のデータベース・ソフトウェアが正常に動作していることを確認した後、データベース A で次の文を発行してスイッチオーバーを実行し、データベース・ロールを元に戻します。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;
```

この文は、既存のトランザクションが完了まで待機する必要があります。スイッチオーバーの完了所要時間を最短に抑えるには、データベース A にまだ接続されているユーザーがすぐにログオフし、データベース B に再接続する必要があります。

注意： 通常の 2 フェーズ準備済スイッチオーバーは、同じバージョンの Oracle ソフトウェアの実行にプライマリ・データベースとスタンバイ・データベースの両方が必要であり、この時点でプライマリ・データベースは下位バージョンの Oracle ソフトウェアを実行しているため、使用できません。かわりに、前述の 1 フェーズ準備なしスイッチオーバーの手順を使用します。準備なしスイッチオーバーは、ロジカル・スタンバイ・データベースを使用するローリング・アップグレードの場合にのみ使用してください。

注意： データベース A がプライマリ・データベースのときに、そこでサポートされない表またはパッケージに対するアクティビティを一時停止した場合、最終的にデータベース A に切り替える計画であれば、データベース B がプライマリ・データベースになっている間にデータベース B でも引き続き同じアクティビティを一時停止する必要があります。

手順 7 アップグレード中に変更されたすべての表をインポートする

手順 5 の「アップグレード後のスタンバイ・データベースでイベントを監視する」では、変更中のサポートされない表をリストする方法について説明しました。プライマリ・データベースでサポートされない DML 文が発行された場合は（例 12-1 を参照）、Oracle Data Pump などのインポート・ユーティリティを使用して、各表の最新バージョンをインポートします。

たとえば、次のインポート・コマンドは scott.emp 表を切り捨て、前のプライマリ・データベース（A）と一致するデータを移入します。

```
IMPDP SYSTEM NETWORK_LINK=DATABASEA TABLES=SCOTT.EMP TABLE_EXIST_ACTION=TRUNCATE
```

このコマンドは、実行前に impdp パスワードの入力を求めてくることに注意してください。

手順 8 スイッチオーバーを完了してユーザー・アプリケーションをアクティブ化する

アップグレード後のデータベース・ソフトウェアが正常に動作していることを確認した後、スイッチオーバーを完了してデータベース・ロールを元に戻します。

1. データベース B で、次のように V$DATABASE ビューの SWITCHOVER_STATUS 列を問い合せます。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;

SWITCHOVER_STATUS
--------------------
TO PRIMARY
```

2. SWITCHOVER_STATUS 列に TO PRIMARY が表示される場合は、データベース B で次の文を発行してスイッチオーバーを完了します。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL PRIMARY;
```

注意： 通常の 2 フェーズ準備済スイッチオーバーは、同じバージョンの Oracle ソフトウェアの実行にプライマリ・データベースとスタンバイ・データベースの両方が必要であり、この時点でプライマリ・データベースは下位バージョンの Oracle ソフトウェアを実行しているため、使用できません。かわりに、前述の 1 フェーズ準備なしスイッチオーバーの手順を使用します。準備なしスイッチオーバーは、ロジカル・スタンバイ・データベースを使用するローリング・アップグレードの場合にのみ使用してください。

---

3. 現在はプライマリ・データベース・ロールで実行中のデータベース B で、ユーザー・アプリケーションとサービスをアクティブ化します。

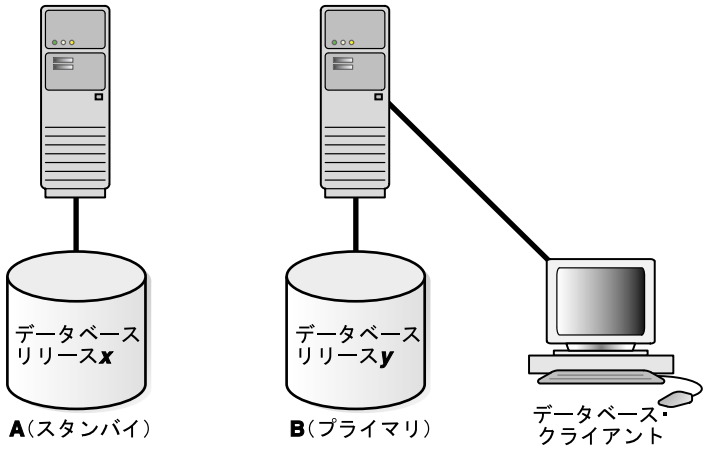
スイッチオーバー後は、新規データベース・ソフトウェア・リリースを実行中の新規プライマリ・データベース（B）から旧ソフトウェア・リリースを実行中の新規スタンバイ・データベース（A）には、REDO データを送信できません。これは次のことを意味します。

- REDO データは、新規プライマリ・データベースに蓄積されます。
- 新規プライマリ・データベースは、この時点では保護されていません。

図 12-4 に、前はスタンバイ・データベース（リリース *y* を実行）で現在はプライマリ・データベースになっているデータベース B と、前はプライマリ・データベース（リリース *x* を実行）で現在はスタンバイ・データベースになっているデータベース A を示します。ユーザーはデータベース B に接続されます。

データベース B がプライマリ・データベースとして適切に機能でき、ビジネスにプライマリ・データベースをサポートするためのロジカル・スタンバイ・データベースが不要な場合、これでローリング・アップグレード処理は完了です。ユーザーにデータベース B へのログインとそこでの作業開始を許可し、問題のない時点でデータベース A を破棄します。それ以外の場合は、手順 9 に進みます。

図 12-4 スイッチオーバー後



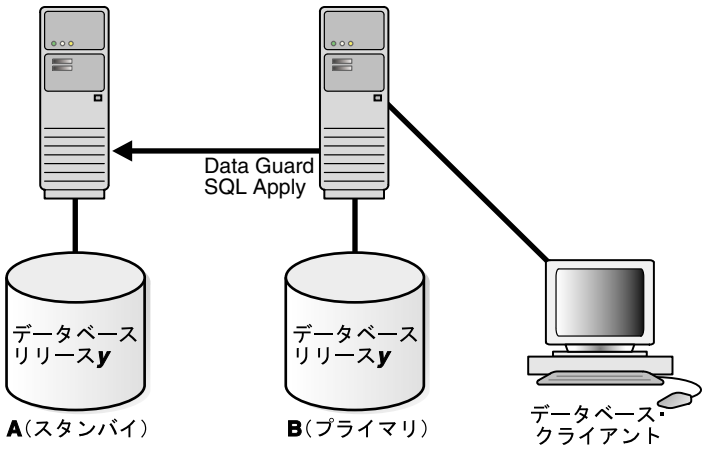
手順 9 古いプライマリ・データベースをアップグレードする

データベース A では引き続きリリース *x* が実行されており、アップグレードして SQL Apply を開始するまでは、データベース B からの REDO データを適用できません。

Oracle Database ソフトウェアのアップグレードの詳細は、該当する Oracle Database リリースの『Oracle Database アップグレード・ガイド』を参照してください。

図 12-5 に、両方のデータベースがアップグレードされた後のシステムを示します。

図 12-5 両方のデータベースがアップグレードされた後



手順 10 古いプライマリ・データベースで SQL Apply を開始する

データベース A で次の文を発行して SQL Apply を開始し、必要に応じてデータベース B へのデータベース・リンクを作成します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE NEW PRIMARY db_link_to_b;
```

注意： データベース・リンクを作成し（まだ設定されていない場合）、NEW PRIMARY 句を使用する必要があります。これは、手順 4 で 1 フェーズ準備なしスイッチオーバーを使用してデータベース A をスタンバイ・データベースに切り替えたためです。

SYS ユーザーとして接続するか、データベース・リンクに対して同様のレベルの権限を持つアカウントで接続する必要があります。

データベース A で SQL Apply を開始すると、プライマリ・データベース（B）に蓄積された REDO データがロジカル・スタンバイ・データベース（B）に送信されます。すべての REDO データがスタンバイ・データベースで使用可能になると、プライマリ・データベースはデータ消失から保護されます。

手順 11 必要な場合は、両方のデータベースの互換性レベルを上げる

COMPATIBLE 初期化パラメータを設定して、両方のデータベースの互換性レベルを上げます。互換性レベルは、ロジカル・スタンバイ・データベースで上げてから、プライマリ・データベースで上げる必要があります。COMPATIBLE パラメータは、スタンバイ・データベースで設定してからプライマリ・データベースで設定します。COMPATIBLE 初期化パラメータの詳細は、『Oracle Database リファレンス』を参照してください。

手順 12 新規ロジカル・スタンバイ・データベースでイベントを監視する

データベース B で実行されるすべての変更がロジカル・スタンバイ・データベース（A）に適切に適用されることを確認するために、手順 5 でデータベース A に対して実行した場合と同様に、DBA_LOGSTDBY_EVENTS ビューを頻繁に問い合せる必要があります。（例 12-1 を参照。）

変更が行われたためにデータベース A が既存のプライマリ・データベースのコピーとして無効になった場合は、データベース A を破棄し、かわりに新規のロジカル・スタンバイ・データベースを作成できます。詳細は、第 4 章「ロジカル・スタンバイ・データベースの作成」を参照してください。

手順 13 必要な場合は、再度スイッチオーバーを実行する

必要な場合は、データベース A がプライマリ・データベース・ロールで再実行されるように（図 12-1 を参照）、データベースのスイッチオーバーを再度実行します。

注意： この時点で、データベース A とデータベース B はいずれも同じバージョンの Oracle ソフトウェアを実行しているため、2 フェーズ準備済スイッチオーバーを使用します。

関連項目： 8.3.1 項「ロジカル・スタンバイ・データベースへのスイッチオーバーの実行」

12.6 既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行

この項の手順では、Oracle ソフトウェアのローリング・アップグレードを実行した後に元の構成に戻し、A がプライマリ・データベース、B がフィジカル・データベースで、いずれのデータベースもアップグレード後の Oracle ソフトウェアを実行している状態にする方法について説明します。

注意： この項の手順では、プライマリ・データベース（A）とフィジカル・スタンバイ・データベース（B）が設定済で、Oracle Database リリース 11.1 以降を使用していることを前提にしています。

表 12-2 に手順を示します。

表 12-2 既存のフィジカル・スタンバイを使用したローリング・アップグレードの実行手順

手順	説明
1	ローリング・アップグレードのためにプライマリ・データベースを準備する（データベース A でこの手順を実行する）
2	フィジカル・スタンバイ・データベースをロジカル・スタンバイ・データベースに変換する（データベース B でこの手順を実行する）
3	ロジカル・スタンバイ・データベースをアップグレードしてプライマリ・データベースと一致させる（データベース B でこの手順を実行する）
4	データベース A を保証付きリストア・ポイントにフラッシュバックする（データベース A でこの手順を実行する）
5	データベース A を新しいバージョンの Oracle ソフトウェアを使用するデータベース B のロジカル・スタンバイにする
6	データベース A を変換してフィジカル・スタンバイに戻す
7	データベース A で管理リカバリを開始する
8	スイッチオーバーを実行してデータベース A をプライマリ・データベースにする

手順 1 ローリング・アップグレードのためにプライマリ・データベースを準備する（データベース A でこの手順を実行する）

1. まだ有効でない場合、フラッシュバック・データベースを有効にします。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE FLASHBACK ON;
SQL> ALTER DATABASE OPEN;
```

2. 保証付きリストア・ポイントを作成します。

```
SQL> CREATE RESTORE POINT pre_upgrade GUARANTEE FLASHBACK DATABASE;
```

手順 2 フィジカル・スタンバイ・データベースをロジカル・スタンバイ・データベースに変換する（データベース B でこの手順を実行する）

1. 第 4 章「ロジカル・スタンバイ・データベースの作成」で説明されている手順を実行します。ただし、次の相違点を除きます。4.2.4.1 項「ロジカル・スタンバイ・データベースへの変換」とは異なるコマンドを使用してロジカル・スタンバイ・データベースを変換する必要があります。ALTER DATABASE RECOVER TO LOGICAL STANDBY db_name のかわりに、次のコマンドを発行します。

```
SQL> ALTER DATABASE RECOVER TO LOGICAL STANDBY KEEP IDENTITY;
SQL> ALTER DATABASE OPEN;
```

2. ロジカル・スタンバイ・データベースでの外部アーカイブ・ログの自動削除を無効にした後、SQL Apply を初めて開始します。

```
SQL> execute DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE', 'FALSE');
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

注意： ロジカル・スタンバイ・データベース（データベース B）で処理されたリモート・アーカイブ・ログは削除しないでください。これらのリモート・アーカイブ・ログは、後でローリング・アップグレード処理の際に必要になります。リカバリ領域を使用してリモート・アーカイブ・ログを格納している場合は、ロジカル・スタンバイ・データベースの通常動作を妨げることなくこれらのログを収容するのに十分な領域があることを確認してください。

手順 3 ロジカル・スタンバイ・データベースをアップグレードしてプライマリ・データベースと一致させる（データベース B でこの手順を実行する）

これで、12.5 項「既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行」に示した手順 1 ～ 6 を実行できます。これらの手順が完了すると、データベース B はアップグレードされたバージョンの Oracle ソフトウェアを実行するプライマリ・データベースになり、データベース A はロジカル・スタンバイ・データベースになります。

次の手順に進んで、データベース A をデータベース B のフィジカル・スタンバイにします。

手順 4 データベース A を保証付きリストア・ポイントにフラッシュバックする（データベース A でこの手順を実行する）

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> FLASHBACK DATABASE TO RESTORE POINT pre_upgrade;
SQL> SHUTDOWN IMMEDIATE;
```

手順 5 データベース A を新しいバージョンの Oracle ソフトウェアを使用するデータベース B のロジカル・スタンバイにする

この時点で、上位バージョンの Oracle ソフトウェアを使用できるように、データベース A で Oracle バイナリを切り替える必要があります。データベース A はフィジカル・スタンバイに切り替えられており、データベース B によって生成された REDO データを適用して自動的にアップグレードされるため、アップグレード・スクリプトは実行しないでください。

次のようにして、データベース A をマウントします。

```
SQL> STARTUP MOUNT;
```

手順 6 データベース A を変換してフィジカル・スタンバイに戻す

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
SQL> SHUTDOWN IMMEDIATE;
```

手順 7 データベース A で管理リカバリを開始する

データベース A は、データベース B により生成された REDO データを適用して、自動的にアップグレードされます。管理リカバリは、プライマリからの新しいインカネーション・ブランチが登録されるまで待機した後、REDO の適用を開始します。

```
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT
FROM SESSION;
```

手順 8 スイッチオーバーを実行してデータベース A をプライマリ・データベースにする

この時点で、データベース B はプライマリ・データベース、データベース A はフィジカル・スタンバイです。どちらも上位バージョンの Oracle ソフトウェアを実行しています。データベース A をプライマリ・データベースにするには、8-8 ページの 8.2.1 項「フィジカル・スタンバイ・データベースへのスイッチオーバーの実行」の手順に従います。

13

Data Guard の使用例

この章では、Data Guard 構成の管理中に遭遇する可能性がある例を説明します。使用例はそれぞれ、ユーザー固有の環境に適応できます。表 13-1 に、この章で説明する使用例を示します。

表 13-1 Data Guard の使用例

参照先	使用例
13.1 項	フェイルオーバー後のロジカル・スタンバイ・データベースの構成
13.2 項	フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換
13.3 項	Open Resetlogs 文の発行後のフラッシュバック・データベースの使用
13.4 項	NOLOGGING 句を指定した後のリカバリ
13.5 項	OMF または ASM を使用するスタンバイ・データベースの作成
13.6 項	プライマリ・データベースでの書込みの欠落エラーからのリカバリ
13.7 項	Recovery Manager バックアップを使用した障害が発生したプライマリのスタンバイ・データベースへの変換

13.1 フェイルオーバー後のロジカル・スタンバイ・データベースの構成

この項では、プライマリ・データベースを別のスタンバイ・データベースにフェイルオーバーした後に、ロジカル・スタンバイ・データベースで実行する必要のある手順について説明します。フェイルオーバーの発生後、ロジカル・スタンバイ・データベースは、元のプライマリ・データベースからの最後の REDO が適用されるまで、新規プライマリ・データベースのスタンバイ・データベースとして機能できません。これは、フェイルオーバー中に新規のプライマリ・データベースで最後の REDO が適用される場合と同じです。実行する必要のある手順は、新規プライマリ・データベースがフェイルオーバー前にフィジカル・スタンバイ・データベースであったかロジカル・スタンバイ・データベースであったかに応じて異なります。

- 13.1.1 項「新規プライマリ・データベースがフィジカル・スタンバイ・データベースだった場合」
- 13.1.2 項「新規プライマリ・データベースがロジカル・スタンバイ・データベースだった場合」

13.1.1 新規プライマリ・データベースがフィジカル・スタンバイ・データベースだった場合

この使用例では、プライマリ・ロールを担う前はフィジカル・スタンバイ・データベースだった新規プライマリ・データベースをサポートするように、ロジカル・スタンバイ・データベースを構成する方法について説明します。この使用例では、SAT はロジカル・スタンバイ・データベース、NYC はプライマリ・データベースです。

手順 1 プライマリ・データベースからのアーカイブを使用不可にする。

NYC データベースで、次の文を発行します（LOG_ARCHIVE_DEST_4 が SAT データベースにアーカイブするように構成されているとします）。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4=DEFER;
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

手順 2 ロジカル・スタンバイ・データベースが新規プライマリ・データベースのスタンバイ・データベースとして機能できることを確認する。

SAT データベースで、次の文を発行します。

```
SQL> EXECUTE DBMS_LOGSTDBY.PREPARE_FOR_NEW_PRIMARY(-
     former_standby_type => 'PHYSICAL' -
     dblink => 'nyc_link');
```

注意： ORA-16109 メッセージが戻され、alert.log に「LOGSTDBY: prepare_for_new_primary failure -- applied too far, flashback required.」という警告が書き込まれる場合は、次の手順を実行します。

1. データベースを警告に示された SCN までフラッシュバックします。
2. 続行する前に、この手順を繰り返します。

ロジカル・スタンバイ・データベースを Apply SCN までフラッシュバックする方法の例は、13.2.3 項を参照してください。

手順 3 プライマリ・データベースでアーカイブを使用可能にする。

NYC データベースで、次の文を発行します（LOG_ARCHIVE_DEST_4 が SAT データベースにアーカイブするように構成されているとします）。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4=ENABLE;
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

手順 4 新規プライマリ・データベースを問い合せ、ロジカル・スタンバイ・データベースでリアルタイム適用を有効化できる SCN を判断する。

NYC データベースで、次の問合せを発行して必要な SCN を判断します。

```
SQL> SELECT MAX(NEXT_CHANGE#) -1 AS WAIT_FOR_SCN FROM V$ARCHIVED_LOG;
```

手順 5 SQL Apply を開始する。

SAT データベースで、次の文を発行します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

この文は、常にリアルタイム適用オプションを指定せずに発行する必要があることに注意してください。手順 4 で戻された過去の WAIT_FOR_SCN が SQL Apply により適用されるまで待機した後で、リアルタイム適用を有効化する必要があります。ロジカル・スタンバイ・データベースでリアルタイム適用を安全に再開できる時点を判断するには、V$LOGSTDBY_PROGRESS ビューを監視します。

```
SQL> SELECT APPLIED_SCN FROM V$LOGSTDBY_PROGRESS;
```

戻される値が手順 4 で戻された WAIT_FOR_SCN 値以上の場合は、SQL Apply を停止し、リアルタイム適用オプションを指定して再開できます。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

13.1.2 新規プライマリ・データベースがロジカル・スタンバイ・データベースだった場合

この使用例では、プライマリ・ロールを担う前はロジカル・スタンバイ・データベースだった新規プライマリ・データベースをサポートするように、ロジカル・スタンバイ・データベースを構成する方法について説明します。この使用例では、SAT はロジカル・スタンバイ・データベース、NYC はプライマリ・データベースです。

手順 1 新規プライマリ・データベースで、ロジカル・スタンバイ・データベースのサポート準備が完了していることを確認する。

NYC データベースで、次の問合せにより値 READY が戻されることを確認します。それ以外の場合、LSP1 バックグラウンド・プロセスの処理が完了しておらず、このロジカル・スタンバイ・データベースの構成は待機する必要があります。次に例を示します。

```
SQL> SELECT VALUE FROM SYSTEM.LOGSTDBY$PARAMETERS
2>   WHERE NAME = 'REINSTATEMENT_STATUS';
```

> **注意：** VALUE 列に NOT POSSIBLE が含まれている場合は、新規プライマリ・データベースでロジカル・スタンバイ・データベースを構成できず、データベースを元に戻す必要があることを意味します。

手順 2 プライマリ・データベースからのアーカイブを使用不可にする。

NYC データベースで、次の文を発行します（LOG_ARCHIVE_DEST_4 が SAT データベースにアーカイブするように構成されているとします）。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4=DEFER;
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

手順 3 ロジカル・スタンバイ・データベースが新規プライマリ・データベースのスタンバイ・データベースとして機能できることを確認する。

SAT データベースで、次の文を発行します。

```
SQL> EXECUTE DBMS_LOGSTDBY.PREPARE_FOR_NEW_PRIMARY(-
     former_standby_type => 'LOGICAL' -
     dblink => 'nyc_link');
```

注意： ORA-16109 メッセージが戻され、alert.log ファイルに「LOGSTDBY: prepare_for_new_primary failure -- applied too far, flashback required.」という警告が書き込まれる場合は、次の手順を実行します。

1. データベースを警告に示された SCN までフラッシュバックします。
2. 続行する前に、この手順を繰り返します。

ロジカル・スタンバイ・データベースを Apply SCN までフラッシュバックする方法の例は、13.2.3 項を参照してください。

手順 4 ローカル・システムにコピーする必要のあるログ・ファイルを判別する。

SAT データベースで、DBMS_LOGSTDBY.PREPARE_FOR_NEW_PRIMARY プロシージャの出力を調べます。このプロシージャでは、ローカル・システムにコピーする必要のあるログ・ファイルが識別されます。手順 3 でフェイルオーバーを非データ消失フェイルオーバーとして識別した場合は、表示されたログ・ファイルを新規プライマリ・データベースからコピーする必要があります。他のロジカル・スタンバイ・データベースまたは前のプライマリ・データベースからは取得しないでください。たとえば、Linux システムでは、grep コマンドを入力します。

```
%grep 'LOGSTDBY: Terminal log' alert_sat.log
LOGSTDBY: Terminal log: [/oracle/dbs/hq_nyc_13.log]
```

注意： 前の手順を複数回実行した場合、関連があるのは最後の試行で得られた出力のみです。ファイル・パスは新規プライマリ・データベースとの相対パスであり、ローカル・ファイル・システムでは解決できない場合があります。

手順 5 ログ・ファイルをローカル・システムにコピーする。

SAT データベースで、端末のログ・ファイルをローカル・システムにコピーします。次の例に、この操作に Linux コマンドを使用する方法を示します。

```
%cp /net/nyc/oracle/dbs/hq_nyc_13.log
/net/sat/oracle/dbs/hq_sat_13.log
```

手順 6 端末のログをロジカル・スタンバイ・データベースに登録する。

SAT データベースで、次の文を発行します。

```
SQL> ALTER DATABASE REGISTER OR REPLACE LOGICAL LOGFILE -
     '/net/sat/oracle/dbs/hq_sat_13.log';
```

手順 7 SQL Apply を開始する。

SAT データベースで、次の文を発行します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY nyc_link;
```

この文は、常にリアルタイム適用オプションを指定せずに発行する必要があることに注意してください。ロジカル・スタンバイ・データベースでリアルタイム適用を使用可能にする必要がある場合は、前述の文が正常終了するまで待機してから、次の文を発行します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

手順 8 プライマリ・データベースでロジカル・スタンバイ・データベースへのアーカイブを使用可能にする。

NYC データベースで、次の文を発行します（LOG_ARCHIVE_DEST_4 が SAT データベースにアーカイブするように構成されているとします）。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4=ENABLE;
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

13.2 フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換

フェイルオーバーの発生後、元のプライマリ・データベースは、修正され、新しい構成のスタンバイ・データベースとして確立されるまで、Data Guard 構成に含められません。これを行うには、フラッシュバック・データベース機能を使用して、障害の発生したプライマリ・データベースを障害発生前の時点にフラッシュバックし、その後新しい構成内のフィジカルまたはロジカル・スタンバイ・データベースに変換します。次の各項で、次の内容を説明します。

- 障害が発生したプライマリ・データベースのフィジカル・スタンバイ・データベースへのフラッシュバック
- 障害が発生したプライマリ・データベースのロジカル・スタンバイ・データベースへのフラッシュバック

注意： フェイルオーバー前に、元のプライマリ・データベースでフラッシュバック・データベースが使用可能になっている必要があります。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

- 特定の適用済 SCN へのロジカル・スタンバイ・データベースのフラッシュバック

関連項目： 障害の発生したプライマリ・データベースを（フラッシュバック・データベースを使用するかわりに）自動的に新規スタンバイ・データベースに戻す方法は、『Oracle Data Guard Broker』を参照してください。

13.2.1 障害が発生したプライマリ・データベースのフィジカル・スタンバイ・データベースへのフラッシュバック

次の手順では、フィジカル・スタンバイ・データベースに対してフェイルオーバーが実行され、ファイルオーバーの際に古いプライマリ・データベースでフラッシュバック・データベースが使用可能にされていることを前提としています。この手順では、古いプライマリ・データベースをフィジカル・スタンバイ・データベースとして Data Guard 構成に戻します。

手順 1 古いスタンバイ・データベースがプライマリ・データベースになった SCN を判別する。

新しいプライマリ・データベース上で次の問合せを発行し、古いスタンバイ・データベースが新しいプライマリ・データベースになった SCN を判別します。

```
SQL> SELECT TO_CHAR(STANDBY_BECAME_PRIMARY_SCN) FROM V$DATABASE;
```

手順 2 障害の発生したプライマリ・データベースをフラッシュバックする。

必要に応じて古いプライマリ・データベースを停止し、マウントして、手順 1 で判別した STANDBY_BECAME_PRIMARY_SCN の値にフラッシュバックします。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> FLASHBACK DATABASE TO SCN standby_became_primary_scn;
```

手順 3 データベースからフィジカル・スタンバイ・データベースに変換する。

古いプライマリ・データベースで次の手順を実行します。

1. 古いプライマリ・データベースで次の文を発行します。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

制御ファイルからスタンバイ制御ファイルへ正常に変換後、この文はデータベースをマウントしません。

2. データベースを停止して、再開します。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
```

手順 4 新しいフィジカル・スタンバイ・データベースへの REDO 転送を開始する。

新しいプライマリ・データベースで次の手順を実行します。

1. 次の問合せを発行し、アーカイブ宛先の現在の状態を調べます。

```
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, ERROR,SRL
  2> FROM V$ARCHIVE_DEST_STATUS;
```

2. 必要に応じて、宛先を使用可能にします。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
```

3. スタンバイ・データベースが新しいプライマリ・データベースからの REDO データの受信を確実に開始するよう、ログ・スイッチを実行し、これが成功したことを確認します。次の SQL 文を新しいプライマリ・データベースで発行します。

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, ERROR,SRL
  2> FROM V$ARCHIVE_DEST_STATUS;
```

新しいスタンバイ・データベースで、REDO 転送サービスが REDO データを別のデータベースに転送しないよう、LOG_ARCHIVE_DEST_*n* 初期化パラメータも変更する必要があります。

手順 5 新しいフィジカル・スタンバイ・データベースで REDO Apply を開始する。

次の SQL 文を新しいフィジカル・スタンバイ・データベースで発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
 2> USING CURRENT LOGFILE DISCONNECT;
```

REDO Apply は、ロールの推移により生成される REDO レコードが検出されるたびに自動的に停止するため、新しいプライマリ・データベースがプライマリ・データベースとなった SCN を超えて適用されるまで、1 回以上再起動する必要があります。障害が発生したプライマリ・データベースがリストアされてスタンバイ・ロールで稼働し始めた後、オプションでスイッチオーバーを実行し、それぞれのデータベースを元の（障害発生前の）ロールに推移できます。詳細は、8.2.1 項「フィジカル・スタンバイ・データベースへのスイッチオーバーの実行」を参照してください。

13.2.2 障害が発生したプライマリ・データベースのロジカル・スタンバイ・データベースへのフラッシュバック

次の手順では、Data Guard 構成ですでにロジカル・スタンバイ・データベースを使用したフェイルオーバーが実行済で、古いプライマリ・データベースでフラッシュバック・データベースが使用可能にされていることを前提としています。この手順は、新しいプライマリ・データベースから古いプライマリ・データベースを正式にインスタンス化せずに、古いプライマリ・データベースを新しいロジカル・スタンバイ・データベースとして Data Guard 構成に戻します。

手順 1 フラッシュバック SCN およびリカバリ SCN を判別する。

フラッシュバック SCN は、障害が発生したプライマリ・データベースのフラッシュバック先となる SCN です。リカバリ SCN は、障害が発生したプライマリ・データベースのリカバリ先となる SCN です。新しいプライマリで次の問合せを発行し、これらの SCN を特定します。

```
SQL> SELECT merge_change# AS FLASHBACK_SCN, processed_change# AS RECOVERY_SCN
 2> FROM DBA_LOGSTDBY_HISTORY
 3> WHERE stream_sequence# = (SELECT MAX(stream_sequence#)-1
 4> FROM DBA_LOGSTDBY_HISTORY);
```

手順 2 障害が発生したプライマリ・データベースを手順 1 で特定したフラッシュバック SCN までフラッシュバックする。

```
SQL> FLASHBACK DATABASE TO SCN flashback_scn;
```

手順 3 障害が発生したプライマリをフィジカル・スタンバイに変換し、リカバリの準備としてスタンバイ・データベースを再度マウントする。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
```

手順 4 新しいプライマリで範囲 [フラッシュバック SCN, リカバリ SCN] 内の REDO を含むログ・ファイルを特定する。

次の問合せによって特定されるログ・ファイルのみが、障害が発生したプライマリ・データベースを安全にリカバリできるアーカイブ・ログ・ファイルであるため、これらのファイルは非常に重要です。次の問合せから戻されるログ・ファイルが手順 5 で登録できない場合、障害が発生したプライマリをロジカル・スタンバイとして回復することはできません。そのような場合は、ロジカル・スタンバイを新しいプライマリから作成する必要があります。

```
SQL> SELECT file_name FROM DBA_LOGSTDBY_LOG
  2> WHERE first_change# <= recovery_scn
  3> AND next_change#  > flashback_scn;
```

手順 5 手順 4 から戻されたすべてのログ・ファイルをフィジカル・スタンバイ（障害が発生したプライマリ）に登録する。

```
SQL> ALTER DATABASE REGISTER LOGFILE 'files_from_step _4';
```

手順 6 手順 1 で特定したリカバリ SCN までリカバリする。

```
SQL> RECOVER MANAGED STANDBY DATABASE UNTIL CHANGE recovery_scn;
```

手順 7 データベース・ガードを有効にする。

```
SQL> ALTER DATABASE GUARD ALL;
```

手順 8 フィジカル・スタンバイをアクティブにしてプライマリ・データベースにする。

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
```

手順 9 データベースをオープンする。

```
SQL> ALTER DATABASE OPEN;
```

手順 10 新しいプライマリへのデータベース・リンクを作成して SQL Apply を開始する。

```
SQL> CREATE PUBLIC DATABASE LINK mylink
  2> CONNECT TO system IDENTIFIED BY password
  3> USING 'service_name_of_new_primary_database';

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY mylink;
```

これで、ロールの可逆的な推移が完了しました。

13.2.3 特定の適用済 SCN へのロジカル・スタンバイ・データベースのフラッシュバック

スタンバイ・データベースのメリットの 1 つは、プライマリ・データベース・サービスに影響を与えずに、スタンバイ・データベースでフラッシュバック・データベースを実行できることです。データベースを特定の時点までフラッシュバックするタスクは簡単ですが、ロジカル・スタンバイ・データベースでは、既知のトランザクションがコミットされる直前の時点までフラッシュバックする操作が必要になる場合があります。このような必要が生じるのは、フェイルオーバー後に新しいプライマリ・データベースを使用してロジカル・スタンバイ・データベースを構成する場合です。

次の手順では、フラッシュバック・データベースと SQL Apply を使用して既知の適用済 SCN までリカバリする方法について説明します。

手順 1 プライマリでの既知の SCN（APPLIED_SCN）の特定後に、フラッシュバック操作に使用するためにロジカル・スタンバイ・データベースで次の問合せを発行して対応する SCN を判別する。

```
SQL> SELECT DBMS_LOGSTDBY.MAP_PRIMARY_SCN (PRIMARY_SCN => APPLIED_SCN)
  2> AS TARGET_SCN FROM DUAL;
```

手順 2 戻された TARGET_SCN までロジカル・スタンバイをフラッシュバックする。

次の SQL 文を発行し、ロジカル・スタンバイ・データベースを指定した SCN までフラッシュバックし、RESETLOGS オプションを指定してロジカル・スタンバイ・データベースをオープンします。

```
SQL> SHUTDOWN;
SQL> STARTUP MOUNT EXCLUSIVE;
SQL> FLASHBACK DATABASE TO SCN <TARGET_SCN>;
SQL> ALTER DATABASE OPEN RESETLOGS;
```

手順 3 SQL Apply が APPLIED_SCN まで適用されたことを確認する。

次の問合せを発行します。

```
SQL> SELECT APPLIED_SCN FROM V$LOGSTDBY_PROGRESS;
```

13.3 Open Resetlogs 文の発行後のフラッシュバック・データベースの使用

スタンバイ・データベースがリアルタイム適用を使用している Data Guard 構成で、プライマリ・データベースでエラーが発生したとします。この場合、同じエラーがスタンバイ・データベースにも適用されます。

ただし、フラッシュバック・データベースが使用可能になっている場合、プライマリおよびスタンバイ・データベースをエラー前の状態に戻せます。これには、適用サービスを再開する前に、プライマリ・データベースで FLASHBACK DATABASE および OPEN RESETLOGS 文を発行し、次に同様の FLASHBACK STANDBY DATABASE 文をスタンバイ・データベースで発行します。（フラッシュバック・データベースが使用可能でない場合、第 3 章および第 4 章で説明されているように、プライマリ・データベースで Point-in-Time リカバリが実行された後、スタンバイ・データベースを再作成する必要があります。）

13.3.1 特定時点へのフィジカル・スタンバイ・データベースのフラッシュバック

次の手順では、プライマリ・データベースで OPEN RESETLOGS 文を発行した後、フィジカル・スタンバイ・データベースの再作成を回避する方法を説明します。

手順 1 RESETLOGS 操作が発生した前の SCN を判別する。

プライマリ・データベースで、次の問合せを使用して RESETLOGS 操作がプライマリ・データベースで発生したよりも 2 つ前のシステム変更番号（SCN）の値を取得します。

```
SQL> SELECT TO_CHAR(RESETLOGS_CHANGE# - 2) FROM V$DATABASE;
```

手順 2 スタンバイ・データベースの現行の SCN を取得する。

スタンバイ・データベースで、次の問合せを使用して現行の SCN を取得します。

```
SQL> SELECT TO_CHAR(CURRENT_SCN) FROM V$DATABASE;
```

手順 3 データベースをフラッシュバックする必要があるかどうかを判別する。

CURRENT_SCN の値が resetlogs_change# - 2 の値より大きい場合、次の文を発行してスタンバイ・データベースをフラッシュバックします。

```
SQL> FLASHBACK STANDBY DATABASE TO SCN resetlogs_change# -2;
```

- CURRENT_SCN の値が resetlogs_change# - 2 の値より小さい場合、手順 4 に進みます。
- スタンバイ・データベースの SCN がプライマリ・データベースの SCN より大幅に小さい場合、適用サービスは停止せずに OPEN RESETLOGS 文中も継続して動作できます。この場合、適用サービスは REDO データ内の OPEN RESETLOGS 文に到達しても停止しないため、データベースのフラッシュバックは不要です。

手順 4 REDO Apply を再開する。

フィジカル・スタンバイ・データベースで REDO Apply を開始するには、次の文を発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
  2> USING CURRENT LOGFILE DISCONNECT;
```

スタンバイ・データベースはプライマリ・データベースから REDO を受信し、適用できます。

13.3.2 特定時点へのロジカル・スタンバイ・データベースのフラッシュバック

次の手順では、プライマリ・データベースをフラッシュバックし、OPEN RESETLOGS 文を発行してオープンした後、ロジカル・スタンバイ・データベースの再作成を回避する方法を説明します。

注意： SQL Apply は、プライマリ・データベースでリセットログ操作の発生を検出すると、ロジカル・スタンバイ・データベースをフラッシュバックしなくてもマイニングできる場合は、REDO の正しいブランチを自動的にマイニングします。それ以外の場合、SQL Apply は「ORA-1346: 指定されたリセット・ログ scn を超えて LogMiner が REDO を処理しました」エラーで停止します。この項では、SQL Apply はこのエラーですでに停止していると仮定しています。

手順 1 プライマリ・データベースの SCN を判別する。

プライマリ・データベースで、次の問合せを使用して RESETLOGS 操作がプライマリ・データベースで発生したよりも 2 つ前のシステム変更番号（SCN）の値を取得します。

```
SQL> SELECT TO_CHAR(RESETLOGS_CHANGE# - 2) AS FLASHBACK_SCN FROM V$DATABASE;
```

手順 2 ロジカル・スタンバイでのフラッシュバック操作のターゲット SCN を判別する。

```
SQL> SELECT DBMS_LOGSTDBY.MAP_PRIMARY_SCN (PRIMARY_SCN => FLASHBACK_SCN)
  2> AS TARGET_SCN FROM DUAL;
```

手順 3 戻された TARGET_SCN までロジカル・スタンバイをフラッシュバックする。

次の SQL 文を発行し、ロジカル・スタンバイ・データベースを指定した SCN までフラッシュバックし、RESETLOGS オプションを指定してロジカル・スタンバイ・データベースをオープンします。

```
SQL> SHUTDOWN;
SQL> STARTUP MOUNT EXCLUSIVE;
SQL> FLASHBACK DATABASE TO SCN <TARGET_SCN>;
SQL> ALTER DATABASE OPEN RESETLOGS;
```

手順 4 SQL Apply を開始する。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

13.4 NOLOGGING 句を指定した後のリカバリ

一部の SQL 文には、NOLOGGING 句を指定するオプションがあります。このオプションによって、データベースに関する操作をオンライン REDO ログ・ファイルに記録しないように指定できます。ユーザーがこの句を指定しても、REDO レコードはオンライン REDO ログ・ファイルに書き込まれます。ただし、このレコードに関連したデータはありません。これが結果的に、スタンバイ・サイトでのログ適用エラーやデータ・アクセス・エラーの原因となり、ログ・ファイルの適用の再開で、手動によるリカバリが必要となります。

注意： この問題を回避するには、CREATE DATABASE または ALTER DATABASE 文に FORCE LOGGING 句を常に指定することをお薦めします。『Oracle Database 管理者ガイド』を参照してください。

13.4.1 ロジカル・スタンバイ・データベースのリカバリ手順

ロジカル・スタンバイ・データベースの場合は、SQL Apply で NOLOGGING 句を使用して対象の表で実行された操作の REDO レコードが検出されると、「ORA-16211 サポートされていないレコードが、アーカイブ REDO ログで見つかりました。」エラーで停止します。

NOLOGGING 句を指定した後のリカバリには、10.5.5 項に説明されているように、プライマリ・データベースから 1 つ以上の表を再作成します。

注意： 通常、NOLOGGING 句の使用はお薦めしません。また、プライマリ・データベース内の特定の表で NOLOGGING 句を使用する操作が実行されることが事前にわかっている場合は、DBMS_LOGSTDBY.SKIP プロシージャを使用して、これらの表に関連付けられている SQL 文をロジカル・スタンバイ・データベースに適用しないようにできます。

13.4.2 フィジカル・スタンバイ・データベースのリカバリ手順

スタンバイ・サイトにコピーされたアーカイブ REDO ログ・ファイルがフィジカル・スタンバイ・データベースに適用されると、データファイルの一部が使用不可能となり、リカバリ不能のマークが付けられます。フィジカル・スタンバイ・データベースにフェイルオーバーするか、読取り専用アクセスでスタンバイ・データベースをオープンし、UNRECOVERABLE のマークが付けられたブロックの範囲を読み取ろうとすると、次のようなエラー・メッセージが表示されます。

```
ORA-01578: ORACLE data block corrupted (file # 1, block # 2521)
ORA-01110: data file 1: '/oracle/dbs/stdby/tbs_1.dbf'
ORA-26040: Data block was loaded using the NOLOGGING option
```

NOLOGGING 句が指定された後でリカバリするには、欠落している REDO データが含まれているデータファイルをプライマリ・サイトからフィジカル・スタンバイ・サイトにコピーする必要があります。次の手順に従ってください。

手順 1 コピーするデータファイルを判別する。

次の手順に従います。

1. プライマリ・データベースを問い合せます。

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
NAME                                               UNRECOVERABLE
-------------------------------------------------- -------------
/oracle/dbs/tbs_1.dbf                                       5216
/oracle/dbs/tbs_2.dbf                                          0
/oracle/dbs/tbs_3.dbf                                          0
/oracle/dbs/tbs_4.dbf                                          0
4 rows selected.
```

2. スタンバイ・データベースを問い合せます。

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
NAME                                               UNRECOVERABLE
-------------------------------------------------- -------------
/oracle/dbs/stdby/tbs_1.dbf                                 5186
/oracle/dbs/stdby/tbs_2.dbf                                    0
/oracle/dbs/stdby/tbs_3.dbf                                    0
/oracle/dbs/stdby/tbs_4.dbf                                    0
4 rows selected.
```

3. プライマリ・データベースの問合せ結果とスタンバイ・データベースの問合せ結果を比較します。

   両方の問合せ結果の UNRECOVERABLE_CHANGE# 列の値を比較します。プライマリ・データベースの UNRECOVERABLE_CHANGE# 列の値の方が、スタンバイ・データベースの同じ列の値より大きい場合は、データファイルをプライマリ・サイトからスタンバイ・サイトへコピーする必要があります。

   この例では、tbs_1.dbf データファイルのプライマリ・データベースにある UNRECOVERABLE_CHANGE# の値の方が大きいため、tbs_1.dbf データファイルをスタンバイ・サイトへコピーする必要があります。

手順 2 プライマリ・サイトで、スタンバイ・サイトにコピーする必要があるデータファイルのバックアップを作成する。

次の SQL 文を発行します。

```
SQL> ALTER TABLESPACE system BEGIN BACKUP;
SQL> EXIT;
% cp tbs_1.dbf /backup
SQL> ALTER TABLESPACE system END BACKUP;
```

手順 3 データファイルをスタンバイ・データベースにコピーする。
欠落している REDO データが含まれているデータファイルを、プライマリ・サイトからリカバリ関連のファイルが格納されているフィジカル・スタンバイ・サイトにコピーします。

手順 4 スタンバイ・データベースで、REDO Apply を再開する。
次の SQL 文を発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

REDO Apply を再開しようとすると、次のエラー・メッセージが（通常アラート・ログに）表示されることがあります。

```
ORA-00308: cannot open archived log 'standby1'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error below
ORA-01152: file 1 was not restored from a sufficiently old backup
ORA-01110: data file 1: '/oracle/dbs/stdby/tbs_1.dbf'
```

ORA-00308 エラーが表示され、REDO Apply が自動的に終了しない場合は、別の端末のウィンドウから次の文を発行して、リカバリを取り消すことができます。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

これらのエラー・メッセージは、アーカイブ・ギャップ内にある 1 つ以上のログ・ファイルが正常に適用されなかった場合に戻されます。これらのエラーを受け取った場合は、ギャップを手動で解決し、手順 4 を繰り返します。アーカイブ・ギャップの手動による解決方法の詳細は、6.3.3.1 項を参照してください。

13.4.3 リカバリ不能処理後にバックアップが必要かどうかの判断

プライマリ・データベースでリカバリ不能な操作を実行した場合は、次の手順に従って新しいバックアップ操作が必要かどうかを判断します。

1. プライマリ・データベースで V$DATAFILE ビューを問い合せ、**システム変更番号（SCN）**、または Oracle データベースが無効な REDO データを生成した最新の時刻を判別します。
2. プライマリ・データベースで次の SQL 文を発行して、新たにバックアップを実行する必要があるかどうかを判断します。

   ```
   SELECT UNRECOVERABLE_CHANGE#,
          TO_CHAR(UNRECOVERABLE_TIME, 'mm-dd-yyyy hh:mi:ss')
   FROM   V$DATAFILE;
   ```

3. 前の手順での問合せで、データファイルが最後にバックアップされた時刻より後のデータファイル・リカバリ不能時刻が報告された場合は、問題となっているデータファイルのバックアップを新たに作成します。

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

13.5 OMF または ASM を使用するスタンバイ・データベースの作成

第 3 章および第 4 章では、フィジカル・スタンバイ・データベースおよびロジカル・スタンバイ・データベースの作成方法を説明しました。この項では、プライマリ・データベースで Oracle Managed Files（OMF）または自動ストレージ管理（ASM）を使用する場合に実行する必要のある追加手順を説明し、これらの章を補完します。

注意： この項の説明は、読者がすでにフィジカル・スタンバイ・データベースの作成方法を理解しており、Recovery Manager、OMF および ASM 機能に習熟していることを前提としています。詳細は、次の資料を参照してください。

- フィジカル・スタンバイ・データベースおよびロジカル・スタンバイ・データベースの作成方法の詳細は、第 3 章、第 4 章および付録 F を参照してください。
- OMF および ASM の詳細は、『Oracle Database 管理者ガイド』を参照してください。
- Recovery Manager の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』および『Oracle Database バックアップおよびリカバリ・リファレンス』を参照してください。

スタンバイ・データベースを作成する前に、次の準備作業を実行します。

1. プライマリ・データベースで強制ロギングを使用可能にします。
2. プライマリ・データベースでアーカイブを使用可能にします。
3. プライマリ・データベースで必要な初期化パラメータをすべて設定します。
4. スタンバイ・データベース用に初期化パラメータ・ファイルを作成します。
5. プライマリ・データベースが OMF を使用するよう構成されている場合、スタンバイ・データベースも OMF を使用するよう構成することをお薦めします。これを行うには、`DB_CREATE_FILE_DEST` および `DB_CREATE_ONLINE_LOG_DEST_n` 初期化パラメータに適切な値を設定します。プライマリおよびスタンバイ・データベースの両方で同じディスク・グループ名が使用されている場合、メンテナンスや後のロールの推移が簡易化されます。
6. `STANDBY_FILE_MANAGEMENT` 初期化パラメータに `AUTO` を設定します。
7. スタンバイ・データベースに接続するために、必要に応じて Oracle Net を構成します。
8. REDO 転送の認証を構成します。詳細は、3.1.2 項「REDO 転送の認証の構成」を参照してください。
9. 制御ファイルをマウントせずにスタンバイ・データベース・インスタンスを起動します。

スタンバイ・データベースを作成するには、次の作業を実行します。

1. スタンバイ・データベースで ASM を使用する場合、スタンバイ・データベース・システムに ASM インスタンスが存在していなければ、インスタンスを作成します。
2. Recovery Manager の BACKUP コマンドを使用して、プライマリ・データベースのデータファイル、アーカイブ・ログ・ファイルおよびスタンバイ制御ファイルのコピーが含まれているバックアップ・セットを作成します。
3. Recovery Manager の DUPLICATE FOR STANDBY コマンドを使用して、バックアップ・セットのデータファイル、アーカイブ REDO ログ・ファイルおよびスタンバイ制御ファイルをスタンバイ・データベースの格納場所にコピーします。

   DUPLICATE FOR STANDBY コマンドは、スタンバイ・インスタンスでの実際のデータ移動を実行します。バックアップ・セットがテープ上に存在する場合、スタンバイ・インスタンスがバックアップ・セットを読み取れるよう、メディア・マネージャを構成する必要があります。バックアップ・セットがディスク上に存在する場合、バックアップ・ピースはスタンバイ・インスタンスによって読み取ることができる必要があります。これには、プライマリ・パス名を NFS を介して取得可能にするか、これらをスタンバイ・システムにコピーし、Recovery Manager の CATALOG BACKUPPIECE コマンドを使用してバックアップ・ピースをリストア前にカタログに追加します。

これらの手順を完了した後、3.2.7 項の手順を実行してフィジカル・スタンバイ・データベースの構成を確認します。

ロジカル・スタンバイ・データベースを作成するには、第 4 章で説明されているスタンバイ・データベースの作成プロセスを実行しますが、次の点を変更します。

1. ロジカル・スタンバイ・データベースの場合、DB_CREATE_FILE_DEST パラメータを設定しても、OMF ファイル名を強制的に作成しません。ただし、このパラメータをプライマリ・データベースで設定する場合、スタンバイ・データベースでも設定する必要があります。
2. プライマリ・システムでロジカル・スタンバイ制御ファイルを作成した後、このファイルをスタンバイ・システムにコピーするために、オペレーティング・システムのコマンドを使用しないでください。かわりに Recovery Manager の RESTORE CONTROLFILE コマンドを使用して、ロジカル・スタンバイ制御ファイルのコピーをスタンバイ・システムにリストアします。
3. プライマリ・データベースで OMF ファイルを使用する場合、スタンバイ・データベースに作成された新しい OMF ファイルをスタンバイ・データベース制御ファイルが使用するよう、Recovery Manager を使用してスタンバイ・データベース制御ファイルを更新します。この操作を実行するには、次の例に示すように、スタンバイ・データベースにのみ接続します。

```
> RMAN TARGET sys@lstdby

target database Password: password

RMAN> CATALOG START WITH '+stby_diskgroup';
RMAN> SWITCH DATABASE TO COPY;
```

これらの手順を完了した後、4.2.5 項の手順を実行してロジカル・スタンバイ・データベースの起動、リカバリおよび検証を行います。

13.6 プライマリ・データベースでの書込みの欠落エラーからのリカバリ

Data Guard 構成でのメディア・リカバリの実行中に、フィジカル・スタンバイ・データベースを使用してプライマリ・データベースで書込みの欠落によるデータ破損エラーを検出できます。これは、プライマリ・データベースの REDO ログに格納されているブロックの SCN をフィジカル・スタンバイ・データベースのブロックの SCN と比較して行います。プライマリ・データベースのブロックの SCN がスタンバイ・データベースの SCN よりも小さい場合は、プライマリ・データベースで書込みの欠落が発生しています。

注意： 書込みの欠落エラーが検出されるのは、プライマリによってブロックがキャッシュに読み込まれ、その後対応する REDO がスタンバイのブロックと比較されるときのみのため、読取りおよび検証が行われていないプライマリとスタンバイの両方に未検出の失効したブロックが存在する可能性があります。これらの失効したブロックは現行のデータベースの動作に影響しません。これは、これらのブロックが読み取られるまでに、スタンバイで現在適用されている REDO の SCN までの、問合せまたは更新の実行に使用されたブロックはすべて、スタンバイによる検証が済んでいるためです。

プライマリの書込みの欠落エラーがスタンバイで検出されると、失効したブロックごとに次のようなブロック・エラー・メッセージが 1 つ以上スタンバイ・データベースのアラート・ファイルに出力されます。

```
Tue Dec 12 19:09:48 2006
STANDBY REDO APPLICATION HAS DETECTED THAT THE PRIMARY DATABASE
LOST A DISK WRITE OF BLOCK 26, FILE 7
NO REDO AT OR AFTER SCN 389667 CAN BE USED FOR RECOVERY.
.
.
.
```

次に、アラート・ファイルは、ORA-00752 エラーがスタンバイ・データベースに発生し、管理リカバリが取り消されたことを示します。

```
Slave exiting with ORA-752 exception
Errors in file /oracle/log/diag/rdbms/dgstwrite2/stwrite2/trace/
stwrite2_pr00_23532.trc:
ORA-00752: recovery detected a lost write of a data block
ORA-10567: Redo is inconsistent with data block (file# 7, block# 26)
ORA-10564: tablespace TBS_2
ORA-01110: data file 7: '/oracle/dbs/btbs_21.f'
ORA-10561: block type 'TRANSACTION MANAGED DATA BLOCK', data object# 57503
.
.
.
```

次に、スタンバイ・データベースは、アラート・ファイルに出力された SCN 時点で、このエラーによるデータファイルの破損がない、一貫した状態にリカバリされます。

```
Recovery interrupted!
Recovered data files to a consistent state at change 389569
```

この最後のメッセージは、アラート・ファイルのかなり後の方に表示されることがあり、ブロック・エラー・メッセージより小さい SCN の場合もあります。また、プライマリ・データベースは、すでにデータファイルに破損があってもエラーが表示されずに動作している場合があります。

このようなエラーからリカバリするための推奨手順は、次の手順に示すように、フィジカル・スタンバイにフェイルオーバーします。

プライマリで書込みの欠落が検出された後のフィジカル・スタンバイへのフェイルオーバーの手順

1. プライマリ・データベースを停止します。ブロック・エラー・メッセージに出力されたSCN 以降のデータがすべて消失します。
2. スタンバイ・データベースで次の SQL 文を発行してプライマリに変換します。

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;

Database altered.

Tue Dec 12 19:15:23 2006
alter database activate standby database
ALTER DATABASE ACTIVATE [PHYSICAL] STANDBY DATABASE (stwrite2)
RESETLOGS after incomplete recovery UNTIL CHANGE 389569
Resetting resetlogs activation ID 612657558 (0x24846996)
Online log /oracle/dbs/bt_log1.f: Thread 1 Group 1 was previously cleared
Online log /oracle/dbs/bt_log2.f: Thread 1 Group 2 was previously cleared
Standby became primary SCN: 389567
Tue Dec 12 19:15:23 2006
Setting recovery target incarnation to 3
Converting standby mount to primary mount.
ACTIVATE STANDBY: Complete - Database mounted as primary (stwrite2)
Completed: alter database activate standby database
```

3. 新しいプライマリをバックアップします。バックアップを即時に実行することは、必要な安全策です。これは、データベースの完全なバックアップ・コピーを作成せずにフェイルオーバーを行うと、変更をリカバリできないためです。フェイルオーバーの結果、元のプライマリ・データベースは Data Guard 構成に含まれなくなり、他のすべてのスタンバイ・データベースは新しいプライマリ・データベースから REDO データを受信して適用します。
4. 新しいプライマリ・データベースをオープンします。
5. 必要に応じて、障害が発生したプライマリをフィジカル・スタンバイとして再作成します。これは、手順 3 で新しいプライマリで作成したデータベース・バックアップを使用して行います（この状況では、フラッシュバック・データベースまたは Data Guard Broker を使用して古いプライマリ・データベースを再インスタンス化することはできません）。

   新しいプライマリから作成したバックアップを使用して作成したフィジカル・スタンバイは、古いスタンバイと同じデータファイルを保持することに注意してください。そのため、新しいスタンバイでは同じブロックを比較するので、古いスタンバイがアクティブ化される前に存在していた未検出の書込みの欠落は、新しいスタンバイによって検出されません。プライマリまたはスタンバイで発生した新しい書込みの欠落はすべての検出されます。

   関連項目： 書込みの欠落の検出を有効にする方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

13.7 Recovery Manager バックアップを使用した障害が発生したプライマリのスタンバイ・データベースへの変換

障害が発生したプライマリ・データベースを変換するには、プライマリでフラッシュバック・データベース機能を有効にし、13.2.1 項または 13.2.2 項で説明した手順に従うことをお薦めします。これらの項の手順では、障害が発生したプライマリをフィジカル・スタンバイまたはロジカル・スタンバイに変換する最も速い方法を説明しています。しかし、障害が発生したプライマリでフラッシュバック・データベースが無効になっている場合でも、次の項で説明するように、障害が発生したプライマリのローカル・バックアップを使用して、障害が発生したプライマリをフィジカル・スタンバイまたはロジカル・スタンバイに変換できます。

- Recovery Manager バックアップを使用した、障害が発生したプライマリのフィジカル・スタンバイへの変換
- Recovery Manager バックアップを使用した障害が発生したプライマリのロジカル・スタンバイへの変換

13.7.1 Recovery Manager バックアップを使用した、障害が発生したプライマリのフィジカル・スタンバイへの変換

この項の手順では、Recovery Manager バックアップを使用して障害が発生したプライマリをフィジカル・スタンバイに変換する方法を説明します。この手順には、古いプライマリの COMPATIBLE 初期化パラメータが 11.0.0 以上に設定されている必要があります。

手順 1 古いスタンバイ・データベースがプライマリ・データベースになった SCN を判別する。

新しいプライマリ・データベース上で次の問合せを発行し、古いスタンバイ・データベースが新しいプライマリ・データベースになった SCN を判別します。

```
SQL> SELECT TO_CHAR(STANDBY_BECAME_PRIMARY_SCN) FROM V$DATABASE;
```

手順 2 データベース全体をリストアおよびリカバリする。

スタンバイが新しいプライマリになった SCN（standby_became_primary_scn）に古いプライマリが到達する前に作成したバックアップを使用して、データベースをリストアします。次に、Point-in-Time リカバリを実行して、古いプライマリをそれと同じ時点までリカバリします。

次の Recovery Manager コマンドを発行します。

```
RMAN> RUN
    {
      SET UNTIL SCN <standby_became_primary_scn + 1>;
      RESTORE DATABASE;
      RECOVER DATABASE;
     }
```

ユーザー管理リカバリでは、先にデータベースを手動でリストアできます。通常、フェイルオーバー前の、数時間を要したバックアップは古くなっています。そのため、障害が発生したプライマリは次のコマンドを使用してリカバリできます。

```
SQL> RECOVER DATABASE USIING BACKUP CONTROLFILE UNTIL CHANGE
<standby_became_primary_scn + 1>;
```

フラッシュバック・データベースを使用する再インスタンス化とは異なり、この手順では standby_became_primary_scn に 1 を加算します。データファイルの場合、SCN までフラッシュバックすることは、その SCN に 1 を加算するまでリカバリすることと同義です。

手順 3 データベースからフィジカル・スタンバイ・データベースに変換する。
古いプライマリ・データベースで次の手順を実行します。

1. 古いプライマリ・データベースで次の文を発行します。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

制御ファイルからスタンバイ制御ファイルへ正常に変換後、この文はデータベースをマウントしません。

2. データベースを停止して、再開します。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
```

手順 4 データベースを読取り専用でオープンする。
次のコマンドを発行します。

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

この手順の目的は、ディクショナリ・チェックを使用して制御ファイルをデータベースと同期させることです。このコマンドの後に、ディクショナリ・チェックで示されたアクションがないかアラート・ログをチェックします。通常、フェイルオーバー中に古いプライマリがデータファイルを追加または削除している途中でなければ、ユーザー・アクションは必要ありません。

手順 5（オプション）必要に応じて、スタンバイを再度マウントする。
フィジカル・スタンバイは、読取り専用でオープンされている間、REDO を適用できます。しかし、読取り専用でオープンせずにフィジカル・スタンバイをリカバリする場合、次のように、必要に応じてフィジカル・スタンバイを停止し、再度マウントしてもかまいません。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
```

手順 6 新しいフィジカル・スタンバイ・データベースへの REDO 転送を再開する。
新しいスタンバイ・データベースの作成前に、新しいプライマリ・データベースは、リモート宛先への REDO の転送を停止しているはずです。REDO 転送サービスを再開するには、新しいプライマリ・データベースで次の手順を実行します。

1. 次の問合せを発行し、アーカイブ宛先の現在の状態を調べます。

```
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, ERROR,SRL
  2> FROM V$ARCHIVE_DEST_STATUS;
```

2. 必要に応じて、宛先を使用可能にします。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
```

3. スタンバイ・データベースが新しいプライマリ・データベースからの REDO データの受信を確実に開始するよう、ログ・スイッチを実行し、これが成功したことを確認します。

注意： これは、新しいプライマリに続いて、古いプライマリが新しいスタンバイになるために重要な手順です。この手順が実行されないと、古いプライマリは誤ったデータベース・ブランチにリカバリされます。問題を修正する唯一の方法は、古いプライマリを再度変換することです。

SQL プロンプトで次の文を入力します。

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, ERROR,SRL
  2> FROM V$ARCHIVE_DEST_STATUS;
```

新しいスタンバイ・データベースで、REDO 転送サービスが REDO データを別のデータベースに転送しないよう、LOG_ARCHIVE_DEST_*n* 初期化パラメータも変更する必要があります。1 つのサーバー・パラメータ・ファイル（SPFILE）でプライマリおよびスタンバイ・データベース・ロールの両方が VALID_FOR 属性で設定されている場合、この手順を実行する必要はありません。これにより、Data Guard 構成は、ロールの推移後も正しく動作します。

手順 7 REDO Apply を開始する。

次のようにして、新しいフィジカル・スタンバイ・データベースで REDO Apply を開始します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
  2> USING CURRENT LOGFILE DISCONNECT;
```

障害が発生したプライマリ・データベースがリストアされてスタンバイ・ロールで稼働し始めた後、オプションでスイッチオーバーを実行し、それぞれのデータベースを元の（障害発生前の）ロールに推移できます。詳細は、8.2.1 項「フィジカル・スタンバイ・データベースへのスイッチオーバーの実行」を参照してください。

13.7.2 Recovery Manager バックアップを使用した障害が発生したプライマリのロジカル・スタンバイへの変換

この項の手順では、Recovery Manager バックアップを使用して障害が発生したプライマリをロジカル・スタンバイに変換する方法を説明します。

手順 1 障害が発生したプライマリ・データベースのリカバリ先 SCN を判別する。

新しいプライマリ・データベースで次の問合せを発行し、障害が発生したプライマリ・データベースをリカバリする SCN を判別します。

```
SQL> SELECT APPLIED_SCN RECOVERY_SCN FROM V$LOGSTDBY_PROGRESS;
```

また、新しいプライマリ・データベースで、次のようにして、アーカイブ・ログの処理に使用する SCN を判別します。

1. すべてのスタンバイ REDO ログを必ずアーカイブしておきます。次の問合せを発行し、戻される READY の値を検索します。データベースのサイズおよびアーカイブする必要があるログの数によって、READY のステータスが戻されるまでしばらく時間がかかることがあります。

   ```
   SQL> SELECT VALUE FROM SYSTEM.LOGSTDBY$PARAMETERS WHERE
   NAME='REINSTATEMENT_STATUS';
   ```

2. READY のステータスが戻されたら、次の問合せを実行し、このリカバリの一環としてアーカイブ・ログの処理のための SCN を取得します。

   ```
   SQL> SELECT VALUE ARCHIVE_SCN FROM SYSTEM.LOGSTDBY$PARAMETERS
     2> WHERE NAME='STANDBY_BECAME_PRIMARY_SCN';
   ```

手順 2 不一致のアーカイブ・ログを障害が発生したプライマリ・データベースから削除する。

フェイルオーバー操作以後に作成されたアーカイブ・ログをすべて、障害が発生したプライマリ・データベースから削除します。障害が発生したプライマリ・データベースがスタンバイから分離されていた場合は、現行のプライマリ・データベースと一貫性がない不一致のアーカイブ・ログが存在する可能性があります。これらの不一致のアーカイブ・ログを適用しないようにするには、バックアップおよびフラッシュ・リカバリ領域から削除する必要があります。次の Recovery Manager コマンドを使用すると、関連アーカイブ・ログをフラッシュ・リカバリ領域から削除できます。

```
RMAN> DELETE ARCHIVELOG FROM SCN ARCHIVE_SCN;
```

削除されると、これらの不一致のログおよび後続のトランザクションはリカバリできなくなります。

手順 3 障害が発生したプライマリ・データベースにコピーするログ・ファイルを判別する。

新しいプライマリ・データベースで、次の問合せを発行し、バックアップからリカバリする前に、障害が発生したプライマリ・データベースにコピーする必要があるログ・ファイルの最小セットを判別します。

```
SQL> SELECT file_name FROM DBA_LOGSTDBY_LOG WHERE next_change# > ARCHIVE_SCN;
```

必要なスタンバイ・ログを取得し、バックアップ・セットを新しいスタンバイにコピーして新しいスタンバイ・フラッシュ・リカバリ領域にリストアします。これらのログはスタンバイ REDO ログから得られるため、スタンバイの標準アーカイブの一部ではありません。Recovery Manager ユーティリティでは、部分的なファイル名を使用して正しい場所からファイルを取得できます。

次に、Recovery Manager の BACKUP コマンドの使用例を示します。

```
RMAN> BACKUP AS COPY DEVICE TYPE DISK FORMAT '/tmp/test/%U'
> ARCHIVELOG LIKE '<partial file names from above>%';
```

次に、Recovery Manager の RESTORE コマンドの使用例を示します。

```
RMAN> CATALOG START WITH '/tmp/test';
RMAN> RESTORE ARCHIVELOG FROM SEQUENCE 33 UNTIL SEQUENCE 35;
```

手順 4 バックアップをリストアしてデータベースをリカバリする。

元のプライマリの全データファイルのバックアップをリストアし、RECOVERY_SCN + 1 までリカバリします。現行の制御ファイルを利用することをお薦めします。

1. データベースを制限モードで起動して、データベースのオープン後に GUARD ALL コマンドが発行されるまでログ・トランザクションから保護します。
2. バックアップを使用して障害が発生したプライマリ・データベースのデータファイルをリストアします。
3. フラッシュバック・データベースが有効な場合は、無効にします（USING BACKUP CONTROLFILE 句に必要）。
4. SQL*Plus で、RECOVERY_SCN +1 まで Point-in-Time リカバリを実行します。

現行の制御ファイルを使用してもバックアップ制御ファイルを使用しても、USING BACKUP CONTROLFILE 句を指定してリストアされるアーカイブ・ログを指すことができるようにする必要があります。指定しないと、リカバリ・プロセスでは、手順 3 で取得したログではなく、オンライン REDO ログへのアクセスが試行されることがあります。手順 3 で取得した順序の入力を求められた場合は、次のようにリストアされるアーカイブ・ログのコピーのファイル名を指定します。

```
SQL> RECOVER DATABASE UNTIL CHANGE RECOVERY_SCN + 1 USING BACKUP CONTROLFILE;
```

手順 5 RESETLOGS オプションでデータベースをオープンする。

```
SQL> ALTER DATABASE OPEN RESETLOGS;
```

手順 6 データベース・ガードを有効にする。

```
SQL> ALTER DATABASE GUARD ALL;
```

手順 7 新しいプライマリ・データベースへのデータベース・リンクを作成して SQL Apply を開始する。

```
SQL> CREATE PUBLIC DATABASE LINK myLink
  2> CONNECT TO SYSTEM IDENTIFIED BY password
  3> USING 'service name of new primary database';

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY myLink;
```

この時点で、制限セッション（ALTER SYSTEM DISABLE RESTRICTED SESSION）を無効にできます。つまり、データベースを再起動して手順 4.3 のフラッシュバックを再度有効にする必要がある場合は、RESTRICTED SESSION をオフにして再起動します。

第 II 部

参照先

この部では、Oracle Data Guard のスタンバイ・データベースの特長について参考となる資料を提供します。

この部は、次の章で構成されています。

- 第 14 章「初期化パラメータ」
- 第 15 章「LOG_ARCHIVE_DEST_n パラメータの属性」
- 第 16 章「Data Guard に関連する SQL 文」
- 第 17 章「Oracle Data Guard に関連するビュー」

14

初期化パラメータ

この章では、Data Guard 環境のデータベースに影響を与える初期化パラメータについて説明します。

表 14-1 に初期化パラメータと、そのパラメータがプライマリ・データベース・ロール、スタンバイ・データベース・ロール、あるいはその両方のいずれに適用されるかを示します。また、これらのパラメータを Data Guard 環境で設定する場合の固有の注意点および推奨事項も記載します。初期化パラメータに関する詳細は、『Oracle Database リファレンス』を参照してください。これには、初期化パラメータを更新する方法として、ALTER SYSTEM SET 文（たとえば ALTER SYSTEM SET LOG_ARCHIVE_TRACE）を発行する方法および初期化パラメータ・ファイルを編集する方法の説明も含まれています。初期化パラメータの設定方法の詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

表 14-1 Data Guard 構成内のインスタンスの初期化パラメータ

パラメータ	適用可能先	注意点および推奨事項
COMPATIBLE = *release_number*	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	スイッチオーバーを実行すると思われる場合は、同じ値をプライマリ・データベースおよびスタンバイ・データベースに指定します。値が異なる場合は、REDO 転送サービスが REDO データをプライマリ・データベースからスタンバイ・データベースに転送できない可能性があります。3.2.3 項の例を参照してください。 SQL Apply を使用したローリング・アップグレードの場合は、このパラメータを 12.4 項「ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行」で説明したガイドラインに従って設定します。
CONTROL_FILE_RECORD_KEEP_TIME = *number_of_days*	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。このパラメータを使用して、指定された日数（0 ～ 365）内に、制御ファイル内（アーカイブ REDO ログ・ファイルなどの必要な情報が含まれている）の再使用可能レコードが上書きされないようにします。
(CONTROL_FILES = '*control_file_name*' , '*control_file_name*', '...')	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	必須。1 つ以上の制御ファイルのパス名とファイル名を指定します。制御ファイルはすでにデータベースに存在している必要があります。2 つの制御ファイルを使用することをお薦めします。現行の制御ファイルの別のコピーが存在していれば、問題のない制御ファイルを問題のある制御ファイルの場所にコピーした後、容易にインスタンスを再開できます。3.2.3 項の例を参照してください。

表 14-1 Data Guard 構成内のインスタンスの初期化パラメータ（続き）

パラメータ	適用可能先	注意点および推奨事項
DB_FILE_NAME_CONVERT = ('*location_of_primary_database_datafile*' , '*location_of_standby_database_datafile_name*' , '...')	フィジカル・スタンバイ スナップショット・スタンバイ	スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、またはデータファイルが格納されているスタンバイ・システムのディレクトリがプライマリ・システムと異なる場合は必須。このパラメータには、文字列をペアで指定する必要があります。最初の文字列は、プライマリ・データベース・ファイル名の中で検索される文字列です。その文字列が一致すると、スタンバイ・データベース・ファイル名を構成する 2 番目の文字列に置き換えられます。複数のペアのファイル名を指定できます。例 3-1 も参照してください。
DB_UNIQUE_NAME = *Unique name for the database*	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	推奨。ただし、LOG_ARCHIVE_CONFIG パラメータを指定している場合は必須。このデータベースの一意の名前を指定します。プライマリ・データベースとスタンバイ・データベースのロールが交代しても、この名前は変更されません。DB_UNIQUE_NAME パラメータのデフォルトは、DB_NAME パラメータの値です。
FAL_CLIENT = *Oracle_Net_service_name*	フィジカル・スタンバイ スナップショット・スタンバイ	FAL_SERVER パラメータが指定されている場合は必須。FAL サーバー（通常はプライマリ・データベース）が FAL クライアント（スタンバイ・データベース）への接続に使用する Oracle Net サービス名を指定します。
FAL_SERVER = *Oracle_Net_service_name*	フィジカル・スタンバイ スナップショット・スタンバイ	FAL_CLIENT パラメータが指定されている場合は必須。欠落しているアーカイブ REDO ログ・ファイルをこのスタンバイ・データベースがフェッチ（要求）できるフェッチ元データベースの Oracle Net サービス名を 1 つ以上指定します。
INSTANCE_NAME	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。このパラメータが定義されており、プライマリ・データベースとスタンバイ・データベースが同じホストに存在する場合は、スタンバイ・データベースに対してプライマリ・データベースとは異なる名前を指定します。3.2.3 項の例を参照してください。
LOG_ARCHIVE_CONFIG= 'DG_CONFIG=(*db_unique_name, db_unique_name, ...*)'	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	推奨。Data Guard 構成内のプライマリ・データベースおよび各スタンバイ・データベースの DB_UNIQUE_NAME を識別する場合、DG_CONFIG 属性を指定します。Data Guard 構成内で Oracle Real Application Clusters（RAC）プライマリ・データベースが最大保護モードまたは最大可用性モードで稼働している場合、この Data Guard 構成へのスタンバイ・データベースの動的追加を使用可能するには、DG_CONFIG 属性を設定する必要があります。
LOG_ARCHIVE_DEST_*n* = {LOCATION=*path_name* | SERVICE=*service_name, attribute, attribute, ...* }	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	必須。最大 10（*n* = 1, 2, 3, ... 10）の宛先を定義します。いずれも LOCATION または SERVICE 属性を指定する必要があります。各 LOG_ARCHIVE_DEST_*n* パラメータに対し、それぞれ対応する LOG_ARCHIVE_DEST_STATE_*n* パラメータを指定します。
LOG_ARCHIVE_DEST_STATE_*n* = {ENABLE | DEFER | ALTERNATE}	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	必須。LOG_ARCHIVE_DEST_STATE_*n* パラメータを指定して、指定された（または代替の）宛先への REDO 転送サービスによる REDO データの転送を有効または無効に設定します。各 LOG_ARCHIVE_DEST_*n* パラメータに対し、LOG_ARCHIVE_DEST_STATE_*n* パラメータを定義します。第 15 章も参照してください。

表 14-1 Data Guard 構成内のインスタンスの初期化パラメータ（続き）

パラメータ	適用可能先	注意点および推奨事項
LOG_ARCHIVE_FORMAT= *log%d_%t_%s_%r*.arc	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	LOG_ARCHIVE_FORMAT および LOG_ARCHIVE_DEST_n パラメータが連結され、データベースに完全修飾されたアーカイブ REDO ログ・ファイル名が生成されます。
LOG_ARCHIVE_LOCAL_FIRST =[TRUE|FALSE]	プライマリ スナップショット・スタンバイ	このパラメータは廃止されており、下位互換性のためにのみ保持されています。このパラメータを明示的に設定する場合は、TRUE にのみ設定することをお薦めします。
LOG_ARCHIVE_MAX_PROCESSES =*integer*	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。Oracle ソフトウェアが最初に起動するアーカイバ（ARC*n*）プロセスの数（1 ～ 30）を指定します。デフォルト値は 4 です。
LOG_ARCHIVE_TRACE=*integer*	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。スタンバイ・サイトへの REDO データの転送をトレースするには、このパラメータを設定します。有効な整数値は、付録 G を参照してください。
LOG_FILE_NAME_CONVERT ='*location_of_primary_database_redo_logs*','*location_of_standby_database_redo_logs*'	ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、またはログ・ファイルが存在するスタンバイ・サイトのディレクトリ構造がプライマリ・サイトと異なる場合は必須。このパラメータは、プライマリ・データベースのオンライン REDO ログ・ファイルのパス名をスタンバイ・データベースのパス名に変換します。3.2.3 項の例を参照してください。
REMOTE_LOGIN_PASSWORDFILE= {EXCLUSIVE|SHARED}	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オペレーティング・システムの認証を管理ユーザーに使用する場合および SSL を REDO 転送の認証に使用する場合はオプション。それ以外の場合は、Data Guard 構成内の各データベースに対して、EXCLUSIVE または SHARED に設定する必要があります。

表 14-1 Data Guard 構成内のインスタンスの初期化パラメータ（続き）

パラメータ	適用可能先	注意点および推奨事項
SHARED_POOL_SIZE = *bytes*	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。オンライン REDO ログ・ファイルから読み込んだ情報を処理する System Global Area（SGA）の指定に使用します。使用可能な SGA が大きいと、処理できる情報量も大きくなります。
STANDBY_ARCHIVE_DEST= *filespec*	ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	このパラメータは廃止されており、下位互換性のためにのみ保持されています。
STANDBY_FILE_MANAGEMENT = {AUTO|MANUAL}	プライマリ フィジカル・スタンバイ スナップショット・スタンバイ	データファイルがプライマリ・データベースに追加または削除された場合に、手動で変更しなくてもスタンバイ・データベースに対応する変更が加えられるよう、STANDBY_FILE_MANAGEMENT パラメータを AUTO に設定します。プライマリ・データベースとスタンバイ・データベースのディレクトリ構造が異なる場合は、DB_FILE_NAME_CONVERT 初期化パラメータも設定して、プライマリ・データベースのデータファイルの 1 つ以上のセットのファイル名を（フィジカル）スタンバイ・データベースのファイル名に変換する必要があります。詳細および例は、例 3-1 を参照してください。

15

LOG_ARCHIVE_DEST_n パラメータの属性

この章では、LOG_ARCHIVE_DEST_*n* 初期化パラメータの属性のリファレンス情報を提供します。属性には次のものがあります。

AFFIRM および NOAFFIRM
ALTERNATE
COMPRESSION
DB_UNIQUE_NAME
DELAY
LOCATION および SERVICE
MANDATORY
MAX_CONNECTIONS
MAX_FAILURE
NET_TIMEOUT
NOREGISTER
REOPEN
SYNC および ASYNC
VALID_FOR

各 LOG_ARCHIVE_DEST_*n* 宛先には、ローカル・ディスクのディレクトリまたはリモートからアクセスするデータベースを指定する、LOCATION または SERVICE 属性が含まれている必要があります。他の属性はすべてオプションです。

注意： LOG_ARCHIVE_DEST_*n* 初期化パラメータの複数の属性が廃止になりました。これらの属性は下位互換性のためにのみサポートされています。詳細は、『Oracle Database リファレンス』を参照してください。

関連項目： LOG_ARCHIVE_DEST_*n* 宛先を定義して REDO 転送サービスを設定する方法の詳細は、第 6 章を参照してください。

AFFIRM および NOAFFIRM

REDO 転送先で受信した REDO データをスタンバイ REDO ログに書き込む前に確認するか、後に確認するかを制御します。

- AFFIRM: REDO 転送先で受信した REDO データをスタンバイ REDO ログに書き込んだ後に確認するように指定します。
- NOAFFIRM: REDO 転送先で受信した REDO データをスタンバイ REDO ログに書き込む前に確認するように指定します。

カテゴリ	AFFIRM	NOAFFIRM
データ型	キーワード	キーワード
有効な値	該当なし	該当なし
デフォルト値	該当なし	該当なし
必須属性	SERVICE	SERVICE
属性との競合	NOAFFIRM	AFFIRM
対応先	V$ARCHIVE_DEST ビューの AFFIRM 列	V$ARCHIVE_DEST ビューの AFFIRM 列

使用方法

- AFFIRM 属性および NOAFFIRM 属性が指定されていない場合、デフォルトは、SYNC 属性が指定されているときは AFFIRM、ASYNC 属性が指定されているときは NOAFFIRM です。
- SYNC 属性を使用しない AFFIRM 属性の指定は廃止されており、今後のリリースではサポートされません。

関連項目： 15-17 ページの「SYNC および ASYNC」属性

例

次の例は、リモート宛先の AFFIRM 属性を示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 SYNC AFFIRM'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

ALTERNATE

元のアーカイブ先で障害が発生したときに使用する代替アーカイブ先を指定します。

カテゴリ	ALTERNATE=LOG_ARCHIVE_DEST_*n*
データ型	文字列
有効な値	LOG_ARCHIVE_DEST_*n* 宛先
デフォルト値	なし。代替アーカイブ先が指定されていない場合、REDO 転送サービスでは自動的に別のアーカイブ先に変更されません。
必須属性	該当なし
属性との競合	なし[1]
対応先	V$ARCHIVE_DEST ビューの ALTERNATE および STATUS 列

[1] REOPEN 属性が 0（ゼロ）ではない値で指定されている場合、ALTERNATE 属性は無視されます。MAX_FAILURE 属性に 0（ゼロ）ではない値が指定され、障害件数が指定した障害しきい値を超過する場合は、ALTERNATE 宛先が有効になります。このため、ALTERNATE 属性は 0（ゼロ）ではない REOPEN 属性値と競合しません。

使用方法

- ALTERNATE 属性はオプションです。代替アーカイブ先が指定されていない場合、元のアーカイブ先に障害が発生しても、REDO 転送サービスでは自動的に別のアーカイブ先に変更されません。
- 指定できる代替アーカイブ先は LOG_ARCHIVE_DEST_*n* パラメータごとに 1 つのみですが、複数の有効なアーカイブ先で同じ代替アーカイブ先を共有できます。
- 代替アーカイブ先に次のいずれかを指定することが理想的です。
  - 同じローカル・スタンバイ・データベース・システム上の異なるディスクの場所（15-4 ページの例 15-1 を参照）
  - 同じスタンバイ・データベース・システムへの異なるネットワーク・ルート（15-4 ページの例 15-2 を参照）
  - 有効なアーカイブ先を厳密に反映するリモート・スタンバイ・データベース・システム
- 代替アーカイブ先を参照する有効な宛先がない場合は、代替アーカイブ先が保留されていることを意味します。これは、代替アーカイブ先を有効にする自動手段がないためです。ただし、ALTER SYSTEM を使用すると、代替アーカイブ先を実行時に有効にする（または遅延させる）ことができます。
- 代替アーカイブ先に指定できる宛先には、次の制限事項があります。
  - ローカルの必須の宛先を少なくとも 1 つは有効にする。
  - 有効な宛先数は、LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータでの定義と一致させる必要がある。
  - 宛先をそれ自身の代替先にすることはできない。
- 有効な宛先数を増加させると、利用可能な代替アーカイブ先の数が減少します。
- 宛先に障害が発生すると、次のアーカイブ操作時に代替アーカイブ先が有効になります。アーカイブ操作の途中で、代替アーカイブ先を有効にすることはできません。これは、すでに処理済のブロックを再度読み込む必要が生じるためです。これは、REOPEN 属性の機能と同じです。

- REOPEN 属性が 0（ゼロ）ではない値で指定されている場合、MAX_FAILURE 属性が 0（ゼロ）ではない値で指定されていなければ ALTERNATE 属性は無視されます。MAX_FAILURE 属性と REOPEN 属性の値が 0（ゼロ）ではない場合は、障害件数が指定の障害しきい値を超過すると、ALTERNATE 宛先が有効になります。このため、ALTERNATE 属性は 0（ゼロ）ではない REOPEN 属性値と競合しません。

例

例 15-1 のサンプル初期化パラメータ・ファイルでは、エラーが発生したり、デバイスがいっぱいになった場合、LOG_ARCHIVE_DEST_1 は、LOG_ARCHIVE_DEST_2 へのフェイルオーバーを次回のアーカイブ操作で自動的に実行します。

例 15-1 代替アーカイブ先への自動フェイルオーバー

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

このローカル宛先の例では、LOG_ARCHIVE_DEST_STATE_*n* 初期化パラメータの指定に従って、宛先を ALTERNATE 状態にすることができることもわかります。ALTERNATE 状態では、この宛先への REDO 転送サービスによる REDO データの転送は、別の宛先の障害によって自動的にこの宛先が有効になるまで遅延します。

例 15-2 同じスタンバイ・データベースに対する代替 Oracle Net サービス名の定義

この例は、同じスタンバイ・データベースに代替 Oracle Net サービス名を定義する方法を示しています。

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='SERVICE=stby1_path1 ALTERNATE=LOG_ARCHIVE_DEST_3'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=stby1_path2'
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

COMPRESSION

COMPRESSION 属性は、REDO データ・ギャップの解決時に、REDO データを REDO 転送先に圧縮して転送するか、圧縮しないで転送するかを指定するために使用します。

> **注意：** REDO 転送の圧縮は、Oracle Advanced Compression オプションの機能です。REDO 転送の圧縮機能を使用する前に、このオプションのライセンスを購入する必要があります。

カテゴリ	COMPRESSION=ENABLE または DISABLE
データ型	ブール
有効な値	ENABLE または DISABLE
デフォルト値	DISABLE
必須属性	なし
属性との競合	なし
対応先	V$ARCHIVE_DEST ビューの COMPRESSION 列

使用方法

- COMPRESSION 属性はオプションです。指定されていない場合、デフォルトの圧縮動作は DISABLE です。

例

次の例は、COMPRESSION 属性が指定されている LOG_ARCHIVE_DEST_*n* パラメータを示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=denver SYNC COMPRESSION=ENABLE'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

DB_UNIQUE_NAME

この宛先にあるデータベースの一意の名前を指定します。

カテゴリ	DB_UNIQUE_NAME=*name*
データ型	文字列
有効な値	name は、DB_UNIQUE_NAME パラメータを使用してこのデータベースに定義された値と一致している必要があります。
デフォルト値	なし
必須属性	なし
属性との競合	なし
対応先	V$ARCHIVE_DEST ビューの DB_UNIQUE_NAME 列

使用方法

- この属性は、次の場合にはオプションです。
  - LOG_ARCHIVE_CONFIG=DG_CONFIG 初期化パラメータが指定されていない場合
  - これがローカル宛先（LOCATION 属性で指定）の場合
- この属性が必須となるのは、LOG_ARCHIVE_CONFIG=DG_CONFIG 初期化パラメータが指定されており、かつ、これがリモート宛先（SERVICE 属性で指定）の場合です。
- プライマリ・データベースとスタンバイ・データベースの関係を明確に識別するには、DB_UNIQUE_NAME 属性を使用します。この属性は、Data Guard 構成に複数のスタンバイ・データベースが存在する場合に特に有用です。
- DB_UNIQUE_NAME で指定する名前は、DG_CONFIG リストの DB_UNIQUE_NAME 値の 1 つと一致している必要があります。REDO 転送サービスにより、指定された宛先のデータベースの DB_UNIQUE_NAME 属性が DB_UNIQUE_NAME 属性と一致していること、またはその宛先への接続が拒否されていることが検証されます。
- DB_UNIQUE_NAME 属性で指定する名前は、宛先で識別されるデータベースの DB_UNIQUE_NAME 初期化パラメータで指定されている名前と一致している必要があります。

例

次の例では、DB_UNIQUE_NAME パラメータでは boston（DB_UNIQUE_NAME=boston）が指定されており、これは LOG_ARCHIVE_DEST_1 パラメータの DB_UNIQUE_NAME 属性にも指定されています。LOG_ARCHIVE_DEST_2 パラメータの DB_UNIQUE_NAME 属性は、宛先として chicago を指定しています。boston と chicago は、どちらも LOG_ARCHIVE_CONFIG=DG_CONFIG パラメータにリストされています。

```
DB_UNIQUE_NAME=boston
LOG_ARCHIVE_CONFIG='DG_CONFIG=(chicago,boston,denver)'
LOG_ARCHIVE_DEST_1='LOCATION=/arch1/
  VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_2='SERVICE=Sales_DR
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=chicago'
```

DELAY

スタンバイ・サイトで REDO データがアーカイブされてから、スタンバイ・データベースにアーカイブ REDO ログ・ファイルが適用されるまでのタイム・ラグを指定します。

カテゴリ	DELAY[=*minutes*]
データ型	数値
有効な値	0 分以上
デフォルト値	30 分
必須属性	SERVICE
属性との競合	LOCATION
対応先	V$ARCHIVE_DEST ビューの DELAY_MINS および DESTINATION 列

使用方法

- DELAY 属性はオプションです。デフォルトでは、遅延は発生しません。
- DELAY 属性は、スタンバイ宛先のアーカイブ REDO ログ・ファイルが、指定した時間が経過するまでリカバリに利用されないことを示します。時間間隔は分で表され、転送された REDO データがスタンバイ・サイトに正常に到達した時点から計測されます。
- DELAY 属性を使用すると、破損したプライマリ・データまたは誤ったプライマリ・データからスタンバイ・データベースを保護できます。ただし、フェイルオーバー中は、破損が発生した時点までのすべての REDO を適用するための所要時間が長くなるため、トレードオフを伴います。
- DELAY 属性は、REDO データのスタンバイ宛先への転送に影響しません。
- リアルタイム適用が使用可能になっている場合、設定した遅延は無視されます。
- DELAY 属性の変更は、次に REDO データをアーカイブすると（ログ・スイッチ後に）有効になります。進行中のアーカイブ操作には、影響しません。
- 指定した遅延間隔は、スタンバイ・サイトで次のように上書きできます。
  - フィジカル・スタンバイ・データベース用

    ```
    SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NODELAY;
    ```

  - ロジカル・スタンバイ・データベース用

    ```
    SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NODELAY;
    ```

    関連項目： これらの ALTER DATABASE 文の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

例

DELAY 属性を使用すれば、プライマリ・データベースと様々なレベルで同期させる複数のスタンバイ・データベースを維持する構成を設定できます。ただし、REDO Apply で破損時点までのすべての REDO を適用するまでにかかる時間が長くなるため、フェイルオーバー中は、この保護はなんらかのオーバーヘッドを伴います。

たとえば、プライマリ・データベース A にはスタンバイ・データベース B および C がある場合を考えます。スタンバイ・データベース B は、障害時リカバリ・データベースとして設定するため、タイム・ラグは設定しません。スタンバイ・データベース C は 2 時間の遅延に設定されています。これは、スタンバイ・データベースに伝播する前にユーザー・エラーを検出するには十分な時間です。

次の例は、この構成用に DELAY 属性を指定する方法を示しています。

```
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/dbs/'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='SERVICE=stbyB SYNC AFFIRM'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=stbyC DELAY=120'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

注意： または、十分なフラッシュバック・ログ・データがあれば、フラッシュバック・データベースを使用して、データベースを障害発生時点または他のデータベース・インカネーションの SCN まで戻すこともできます。フラッシュバック・データベースの使用方法は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

LOCATION および SERVICE

各宛先には LOCATION 属性または SERVICE 属性を指定して、REDO 転送サービスがローカル・ディスクのディレクトリまたはリモート・データベースの宛先のどちらに REDO データを転送できるかを明示する必要があります。

カテゴリ	LOCATION=*local_disk_directory* または USE_DB_RECOVERY_FILE_DEST	SERVICE=*net_service_name*
データ型	文字列値	文字列値
有効な値	該当なし	該当なし
デフォルト値	なし	なし
必須属性	該当なし	該当なし
属性との競合	SERVICE、DELAY、NOREGISTER、SYNC、ASYNC、NET_TIMEOUT、AFFIRM、NOAFFIRM、COMPRESSION、MAX_CONNECTIONS	LOCATION
対応先	V$ARCHIVE_DEST ビューの DESTINATION および TARGET 列	V$ARCHIVE_DEST ビューの DESTINATION および TARGET 列

使用方法

- LOCATION または SERVICE 属性を指定する必要があります。デフォルトはありません。
- 複数の属性を指定している場合は、属性のリストの最初に LOCATION 属性または SERVICE 属性を指定します。
- LOCATION 属性で少なくとも 1 つのローカル・ディスクのディレクトリを指定する必要があります。これにより、データベースのメディア・リカバリが必要な場合は、ローカルのアーカイブ REDO ログ・ファイルに確実にアクセスできるようになります。ローカルまたはリモートの追加宛先は最大 9 個まで指定できます。
- LOCATION 属性には、次のいずれかを指定できます。
  - LOCATION=*local_disk_directory*

    これは、データベースを置くシステム上のディスク・ディレクトリの一意のディレクトリ・パス名を指定します。これは、アーカイブ REDO ログ・ファイルのローカル宛先です。
  - LOCATION=USE_DB_RECOVERY_FILE_DEST

    フラッシュ・リカバリ領域を構成するには、DB_RECOVERY_FILE_DEST 初期化パラメータを使用して、フラッシュ・リカバリ領域として機能するディレクトリまたは Oracle Storage Manager のディスク・グループを指定します。フラッシュ・リカバリ領域の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- SERVICE 属性を指定する場合は、次のようにします。
  - SERVICE 属性と、REDO データの送信先となるリモート Oracle データベース・インスタンスを識別する有効な Oracle Net サービス名（SERVICE=*net_service_name*）を指定して、リモート宛先を識別します。

    SERVICE 属性で指定する Oracle Net サービス名は、リモート・データベースへの接続に必要な情報を含む接続記述子に変換されます。

    関連項目： Oracle Net サービス名の設定方法は、『Oracle Database Net Services 管理者ガイド』を参照してください。

  - リモート宛先に REDO データを転送するには、着信するアーカイブ REDO データを受け取るために、ネットワーク接続と、リモート宛先に対応付けられた Oracle データベース・インスタンスが必要です。

- LOCATION および SERVICE 属性の現行の設定を確認するには、V$ARCHIVE_DEST 固定ビューを問い合せます。
  - TARGET 列は、宛先がプライマリ・データベースにとってローカルかリモートかを示します。
  - DESTINATION 列は、宛先に指定されている値を示します。たとえば、宛先パラメータ値は、アーカイブ REDO ログ・ファイルが配置されているリモートの Oracle インスタンスを示す Oracle Net サービス名を指定します。

例

例 1 LOCATION 属性の指定

```
LOG_ARCHIVE_DEST_2='LOCATION=/disk1/oracle/oradata/payroll/arch/'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

例 2 SERVICE 属性の指定

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

MANDATORY

いっぱいになったオンライン・ログ・ファイルが再利用可能になるには、宛先へのアーカイブが成功する必要があることを指定します。

カテゴリ	MANDATORY
データ型	キーワード
有効な値	該当なし
デフォルト値	該当なし
必須属性	該当なし
属性との競合	オプション
対応先	V$ARCHIVE_DEST ビューの BINDING 列

使用方法

- MANDATORY が指定されていない場合、デフォルトでは、宛先はオプションとみなされます。

  すべての宛先がオプションである場合にも、最低 1 つの宛先は成功する必要があります。オプションの宛先へのアーカイブが失敗した場合、オンライン REDO ログ・ファイルは再利用可能なままで、最終的に上書きされる可能性があります。しかし、必須の宛先へのアーカイブ操作が失敗した場合、オンライン REDO ログ・ファイルは上書きされません。

- LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* パラメータ（*n* は 1 ～ 10 の整数）は、オンライン REDO ログ・ファイルを上書きできるようになる前に、正常にアーカイブしておく必要がある宛先の数を指定します。

  LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* の件数は、すべての MANDATORY の宛先とオプションのローカル宛先によって満たされます。LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータに設定した値が一致すると、オンライン REDO ログ・ファイルは再利用可能になります。たとえば、次のようにパラメータを設定できます。

  ```
  # Database must archive to at least two locations before
  # overwriting the online redo log files.
  LOG_ARCHIVE_MIN_SUCCEED_DEST = 2
  ```

- 1 つ以上のローカル宛先が必要であり、それらに MANDATORY を宣言するか、またはオプションのままにしておくことができます。

  LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータの最小値は 1 のため、最低 1 つのローカル宛先が操作上必須です。

- 必須の宛先のいずれかに障害が発生すると、LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータは不適切になります。

- LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータ値を、必須の宛先数にオプションのローカル宛先数を加えた数よりも大きく設定することはできません。

- V$ARCHIVE_DEST 固定ビューの BINDING 列は、アーカイブ操作に障害がどのように影響するのかを指定します。

例

次の例は、MANDATORY 属性を示しています。

```
LOG_ARCHIVE_DEST_1='LOCATION=/arch/dest MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=denver MANDATORY'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

MAX_CONNECTIONS

アーカイブ REDO ログ・ファイルを REDO 転送先に送信する際に、複数のネットワーク接続を使用できるようにします。複数のネットワーク接続を使用すると、待機時間が長いネットワーク・リンクを介した REDO 転送のパフォーマンスを向上させることができます。

カテゴリ	説明
データ型	整数
有効な値	1 ～ 5
デフォルト値	1
必須属性	なし
属性との競合	なし
対応先	プライマリ・データベースの V$ARCHIVE_DEST ビューの MAX_CONNECTIONS 列

使用方法

- MAX_CONNECTIONS 属性はオプションです。指定した場合は、REDO 転送サービスで ARC*n* プロセスをアーカイブに使用する場合にのみ使用されます。
  - MAX_CONNECTIONS を 1（デフォルト）に設定すると、REDO 転送サービスでは、1 つの ARC*n* プロセスを使用してリモート宛先に REDO データを転送します。
  - MAX_CONNECTIONS を 1 より大きな値に設定すると、REDO 転送サービスでは、複数の ARC*n* プロセスを使用してリモート宛先のアーカイブ REDO ログ・ファイルにパラレルで REDO を転送します。各アーカイバ（ARC*n*）プロセスでは、個別のネットワーク接続を使用します。
- 複数の ARC*n* プロセスを使用すると、REDO 転送はパラレルで行われるため、REDO がリモート宛先に転送される速度は向上します。
- アーカイバ（ARC*n*）プロセスを使用するスタンバイ・データベースでは、MAX_CONNECTIONS 属性が指定されると、スタンバイ REDO ログを使用しません。そのため、このような宛先では次のことができません。
  - リアルタイム適用の使用
  - REDO 転送先としての構成
- 常に使用されているアーカイバ・プロセスの実際の数は、アーカイバのワークロードおよび LOG_ARCHIVE_MAX_PROCESSES 初期化パラメータの値によって異なります。たとえば、すべての宛先の MAX_CONNECTIONS 属性の合計が LOG_ARCHIVE_MAX_PROCESSES の値を超えている場合、Data Guard ではできるだけ多数の ARC*n* プロセスが使用されますが、その数は MAX_CONNECTIONS 属性に指定されている値より少ない場合があります。
- Oracle RAC 環境で複数の ARC*n* プロセスを使用する場合は、1 つのスタンバイ・データベース・インスタンスに REDO データを転送するようにプライマリ・インスタンスを構成します。REDO 転送サービスがそのように構成されていないと、アーカイブはリモート・アーカイブのデフォルトの動作に戻ります。つまり、1 つの ARC*n* プロセスを使用して REDO データを転送します。

例

次の例は、MAX_CONNECTIONS 属性を示しています。

```
LOG_ARCHIVE_DEST_1='LOCATION=/arch/dest'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=denver MAX_CONNECTIONS=3'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

MAX_FAILURE

プライマリ・データベースが失敗した宛先を放棄する前に、REDO 転送サービスが通信を再確立してその宛先への REDO データを転送する連続試行回数を制御します。

カテゴリ	MAX_FAILURE=*count*
データ型	数値
有効な値	0 以上
デフォルト値	なし
必須属性	REOPEN
属性との競合	なし
対応先	V$ARCHIVE_DEST ビューの MAX_FAILURE、FAILURE_COUNT および REOPEN_SECS 列

使用方法

- MAX_FAILURE 属性はオプションです。デフォルトでは、失敗した宛先に対するアーカイブの試行回数に制限はありません。
- この属性は、失敗後に回数制限付きで REDO データの転送を再試行するように、宛先に対する障害の解決方法を指定するときに役立ちます。
- MAX_FAILURE 属性を指定した場合は、REOPEN 属性も設定する必要があります。指定した連続試行回数を超過すると、その宛先は REOPEN 属性が指定されていないものとして処理されます。
- V$ARCHIVE_DEST 固定ビューの FAILURE_COUNT 列で障害件数を確認できます。関連する REOPEN_SECS 列は、REOPEN 属性値を示します。

> **注意：** 宛先の障害件数が、指定した MAX_FAILURE 属性の値に達した場合、その宛先を再利用するには、MAX_FAILURE 属性値または他の属性を修正する必要があります。この結果、障害件数がゼロ（0）にリセットされます。

- ALTER SYSTEM SET 文によって宛先が変更されると、障害件数は 0（ゼロ）にリセットされます。このため、現在の障害件数の値よりも小さな値に MAX_FAILURE 属性を設定する問題が回避されます。
- 障害件数が MAX_FAILURE 属性に設定された値以上になると、REOPEN 属性値は暗黙的に 0（ゼロ）に設定されます。これによって、REDO 転送サービスは、次のアーカイブ操作では REDO データを代替アーカイブ先（ALTERNATE 属性で指定）に転送するようになります。
- MAX_FAILURE 属性を指定せずに（または MAX_FAILURE=0 を指定し）、REOPEN 属性に 0（ゼロ）ではない値を指定すると、REDO 転送サービスは、失敗した宛先に対して無制限にアーカイブを試行します。宛先が MANDATORY 属性を持つ場合、オンライン REDO ログ・ファイルはこの宛先にアーカイブされるまで再利用できません。

例

次の例は、REDO 転送サービスが、宛先 arc_dest に対して、アーカイブ操作を連続最高 3 回、5 秒ごとに試行できる指定を示しています。3 回目の試行後、アーカイブ操作に失敗すると、その宛先は REOPEN 属性が指定されていないものとして扱われます。

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=5 MAX_FAILURE=3'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

NET_TIMEOUT

LGWR バックグラウンド・プロセスが、REDO 転送先に送信された REDO データの確認待ちをブロックする時間を秒数で指定します。確認が NET_TIMEOUT の秒数内に受信されない場合は、エラーがログに記録され、その宛先への REDO 転送セッションは終了します。

カテゴリ	NET_TIMEOUT=*seconds*
データ型	数値
有効な値	1[1] ～ 1200
デフォルト値	30 秒
必須属性	SYNC
属性との競合	ASYNC（ASYNC 属性を指定しても、REDO 転送サービスでは無視され、エラーは戻されません。）
対応先	プライマリ・データベースの V$ARCHIVE_DEST ビューの NET_TIMEOUT 列

[1] 最小値は 1 秒に設定できますが、一時的なネットワーク・エラーによるスタンバイ・データベースからの切断を回避するために、最小値は 8 ～ 10 秒に設定することをお薦めします。

使用方法

- NET_TIMEOUT 属性はオプションです。ただし、NET_TIMEOUT 属性を指定しないと 30 秒に設定されますが、プライマリ・データベースが停止する可能性があります。この状況を回避するには、NET_TIMEOUT 属性に 0（ゼロ）以外の小さい値を指定することによって、ネットワーク・サーバーからのステータスを待機する際、ユーザーが指定したタイムアウト時間の期限が切れた後に、プライマリ・データベースが操作を続行できます。

例

次の例は、NET_TIMEOUT 属性を使用して、プライマリ・データベースのネットワーク・タイムアウト値を 40 秒に指定する方法を示しています。

```
LOG_ARCHIVE_DEST_2='SERVICE=stby1 SYNC NET_TIMEOUT=40'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

NOREGISTER

アーカイブ REDO ログ・ファイルの位置が、対応する宛先に記録されてはいけないことを示します。

カテゴリ	NOREGISTER
データ型	キーワード
有効な値	該当なし
デフォルト値	該当なし
必須属性	SERVICE
属性との競合	LOCATION
対応先	V$ARCHIVE_DEST ビューの DESTINATION および TARGET 列

使用方法

- NOREGISTER 属性は、スタンバイ・データベースの宛先が Data Guard 構成の一部である場合はオプションです。
- NOREGISTER 属性は、宛先が Data Guard 構成の一部でない場合は必須です。
- これは、リモートの宛先のみの属性です。各アーカイブ REDO ログ・ファイルの位置は、常にプライマリ・データベースの制御ファイルに記録されます。

例

次の例は、NOREGISTER 属性を示しています。

```
LOG_ARCHIVE_DEST_5='NOREGISTER'
```

REOPEN

REDO 転送サービスが失敗した宛先の再オープンを試行するまでの最小秒数を指定します。

カテゴリ	REOPEN [=*seconds*]
データ型	数値
有効な値	0 秒以上
デフォルト値	300 秒
必須属性	なし
属性との競合	該当なし
対応先	V$ARCHIVE_DEST ビューの REOPEN_SECS および MAX_FAILURE 列

使用方法

- REOPEN 属性はオプションです。
- REDO 転送サービスは、失敗した宛先の再オープンをログ・スイッチ時に試行します。
- REDO 転送サービスは、最後のエラーの時刻に REOPEN 間隔を加算した時刻が現在の時刻に達していないかどうかをチェックします。達していない場合、REDO 転送サービスは宛先の再オープンを試行します。
- REOPEN は、接続障害のみでなく、すべてのエラーに適用されます。エラーには、ネットワーク障害、ディスク・エラー、クオータ例外などがありますが、これらに制限されません。
- オプションの宛先に REOPEN を指定すると、エラーが発生した場合に Oracle データベースでオンライン REDO ログ・ファイルを上書きできます。MANDATORY 宛先に REOPEN を指定すると、REDO 転送サービスは、REDO データを正常に転送できない場合にプライマリ・データベースを停止します。この場合には、次のオプションを考慮します。
  - 宛先を遅延させる、宛先を OPTIONAL として指定する、または SERVICE 属性値を変更することで宛先を変更
  - 代替宛先の指定
  - 宛先の無効化

例

次の例は、REOPEN 属性を示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 MANDATORY REOPEN=60'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

SYNC および ASYNC

使用する REDO 転送モードが同期（SYNC）なのか、非同期（ASYNC）なのかを指定します。

カテゴリ	SYNC	ASYNC
データ型	キーワード	キーワード
有効な値	該当なし	該当なし
デフォルト値	該当なし	なし
必須属性	なし	なし
属性との競合	ASYNC、LOCATION	SYNC、LOCATION
対応先	V$ARCHIVE_DEST ビューの TRANSMIT_MODE 列	V$ARCHIVE_DEST ビューの TRANSMIT_MODE 列

使用方法

- トランザクションをコミットするには、SYNC 属性が指定されている使用可能な各宛先で、そのトランザクションによって生成された REDO データを受信しておく必要があります。
- トランザクションをコミットするのに、ASYNC 属性が指定されている宛先で、そのトランザクションによって生成された REDO データを受信しておく必要はありません。これは、SYNC および ASYNC がいずれも指定されていない場合のデフォルトの動作です。

例

次の例は、SYNC 属性が指定されている LOG_ARCHIVE_DEST_*n* パラメータを示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 SYNC'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

VALID_FOR

次の要因に基づき、REDO データを宛先に書き込むかどうかを指定します。

- データベースがプライマリ・ロールとスタンバイ・ロールのどちらで現在実行されているか
- オンライン REDO ログ・ファイルまたはスタンバイ REDO ログ・ファイル（あるいはその両方）が、現在この宛先のデータベースでアーカイブされているかどうか

カテゴリ	VALID_FOR=(*redo_log_type, database_role*)
データ型	文字列値
有効な値	該当なし
デフォルト値	VALID_FOR=(ALL_LOGFILES, ALL_ROLES)
必須属性	なし
属性との競合	なし
対応先	V$ARCHIVE_DEST ビューの VALID_NOW、VALID_TYPE および VALID_ROLE 列

使用方法

- VALID_FOR 属性はオプションです。ただし、Data Guard 構成内の各データベースで REDO 転送先ごとに VALID_FOR 属性を指定し、構成内のスタンバイ・データベースへのロール推移後も REDO 転送が続行されるようにすることをお薦めします。
- LOG_ARCHIVE_DEST_*n* 宛先ごとにこれらの要因を構成するには、キーワードのペア（VALID_FOR=(*redo_log_type*,*database_role*)）を使用してこの属性を指定します。
  - *redo_log_type* キーワードは、次のいずれかをアーカイブする場合に有効な宛先を識別します。
    - ONLINE_LOGFILE: この宛先は、オンライン REDO ログ・ファイルをアーカイブする場合のみ有効です。
    - STANDBY_LOGFILE: この宛先は、スタンバイ REDO ログ・ファイルをアーカイブする場合のみ有効です。
    - ALL_LOGFILES: この宛先は、オンライン REDO ログ・ファイルまたはスタンバイ REDO ログ・ファイルをアーカイブする場合に有効です。
  - *database_role* キーワードは、この宛先がアーカイブに有効なロールを識別します。
    - PRIMARY_ROLE: この宛先は、データベースがプライマリ・ロールで実行されている場合のみ有効です。
    - STANDBY_ROLE: この宛先は、データベースがスタンバイ・ロールで実行されている場合のみ有効です。
    - ALL_ROLES: この宛先は、データベースがプライマリ・ロールまたはスタンバイ・ロールで実行されている場合に有効です。
- 宛先に VALID_FOR 属性を指定しないと、データベースがプライマリ・ロールまたはスタンバイ・ロールで稼働しているかどうかに関係なく、デフォルトで、オンライン REDO ログ・ファイルとスタンバイ REDO ログ・ファイルのアーカイブが宛先で有効になります。このデフォルトの動作は、VALID_FOR 属性で（ALL_LOGFILES,ALL_ROLES）キーワード・ペアを設定した場合と同様です。
- VALID_FOR 属性を使用すると、同じ初期化パラメータ・ファイルをプライマリ・ロールとスタンバイ・ロールの両方に使用できます。

例

次の例は、VALID_FOR のデフォルトのキーワードのペアを示します。

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata VALID_FOR=(ALL_LOGFILES, ALL_ROLES)'
```

このデータベースがプライマリ・ロールまたはスタンバイ・ロールで稼働している場合、宛先 1 は、すべてのログ・ファイルをローカル・ディレクトリ /disk1/oracle/oradata にアーカイブします。

16

Data Guard に関連する SQL 文

この章では、Data Guard 環境のスタンバイ・データベースで操作を実行するときに役立つ SQL および SQL*Plus 文の概要について説明します。この章は、次の項目で構成されています。

- ALTER DATABASE 文
- ALTER SESSION 文

この章では、特定の SQL 文について、その構文と簡単な概要を示します。これらの SQL 文およびその他の SQL 文の完全な構文および説明は、『Oracle Database SQL 言語リファレンス』を参照してください。

`ALTER SYSTEM SET` 文を使用して設定および動的な更新を行うことができる初期化パラメータのリストは、第 14 章を参照してください。

16.1 ALTER DATABASE 文

表 16-1 で、Data Guard に関連のある ALTER DATABASE 文について説明します。

表 16-1 Data Guard 環境で使用される ALTER DATABASE 文

ALTER DATABASE 文	説明
`ADD [STANDBY] LOGFILE [THREAD` *integer*`] [GROUP` *integer*`]` *filespec*	指定したスレッドに 1 つ以上のオンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループを追加し、スレッドが割り当てられたインスタンスでログ・ファイルを使用できるようにします。 この文の例は、9.3.5 項を参照してください。
`ADD [STANDBY] LOGFILE MEMBER '`*filename*`' [REUSE] TO` *logfile-descriptor*	既存のオンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループに新しいメンバーを追加します。
`[ADD|DROP] SUPPLEMENTAL LOG DATA {PRIMARY KEY|UNIQUE INDEX} COLUMNS`	この文は、ロジカル・スタンバイ・データベース専用です。 ロジカル・スタンバイ・データベースを作成する前に、サプリメンタル・ロギングを使用可能にするために使用します。サプリメンタル・ロギングがロジカル・スタンバイ・データベースに対する変更のソースであるため、このようにする必要があります。完全なサプリメンタル・ロギングを実装するには、この文で PRIMARY KEY COLUMNS または UNIQUE INDEX COLUMNS キーワードを指定する必要があります。 詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
`COMMIT TO SWITCHOVER TO [[PRIMARY] | [[PHYSICAL|LOGICAL] [STANDBY]] [WITH | WITHOUT] SESSION SHUTDOWN] [WAIT | NOWAIT]`	次のスイッチオーバーを実行します。 ■ カレント・プライマリ・データベースをスタンバイ・データベース・ロールに変更する ■ 1 つのスタンバイ・データベースをプライマリ・データベース・ロールに変更する **注意：**ロジカル・スタンバイ・データベースでは、ALTER DATABASE COMMIT TO SWITCHOVER 文を発行する前に、ALTER DATABASE PREPARE TO SWITCHOVER 文を発行し、スイッチオーバーのためにデータベースを準備します。 この文の例は、8.2.1 項および 8.3.1 項を参照してください。
`CONVERT TO [[PHYSICAL|SNAPSHOT] STANDBY] DATABASE`	フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースに、またはその逆に変換します。
`CREATE [PHYSICAL|LOGICAL] STANDBY CONTROLFILE AS '`*filename*`' [REUSE]`	フィジカルまたはロジカル・スタンバイ・データベースの維持に使用される制御ファイルを作成します。この文はプライマリ・データベースで発行します。 この文の例は、3.2.2 項を参照してください。
`DROP [STANDBY] LOGFILE` *logfile_descriptor*	オンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループのすべてのメンバーを削除します。 この文の例は、9.3.5 項を参照してください。
`DROP [STANDBY] LOGFILE MEMBER` *'filename'*	オンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループの 1 つ以上のメンバーを削除します。

表 16-1 Data Guard 環境で使用される ALTER DATABASE 文（続き）

ALTER DATABASE 文	説明
[NO] FORCE LOGGING	一時表領域および一時セグメントに対する変更を除くデータベースに対するすべての変更を、Oracle データベースが記録するかどうかを制御します。[NO] FORCE LOGGING 句は、スタンバイ・データベース間の一貫性の欠如を防止するために必要です。 この文を発行する場合、プライマリ・データベースがマウントされている必要があります。ただし、オープンする必要はありません。この文の例は、3.1.1 項を参照してください。
GUARD	ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスを制御します。可能な値は、ALL、STANDBY および NONE です。詳細は、10.2 項を参照してください。
MOUNT [STANDBY DATABASE]	スタンバイ・データベースをマウントし、スタンバイ・インスタンスが REDO データをプライマリ・インスタンスから受信できるようにします。
OPEN	すでに起動され、マウントされたデータベースをオープンします。 ■ フィジカル・スタンバイ・データベースは読取り専用モードでオープンされ、ユーザーを読取り専用トランザクションに制限し、REDO データの生成を防ぎます。 ■ ロジカル・スタンバイ・データベースは読取り / 書込みモードでオープンされます。
PREPARE TO SWITCHOVER TO [PRIMARY] | [[PHYSICAL|LOGICAL] [STANDBY]] [WITH | WITHOUT] SESSION SHUTDOWN] [WAIT | NOWAIT]	この文は、ロジカル・スタンバイ・データベース専用です。 スイッチオーバーが実行される前に、LogMiner ディクショナリを構築することにより、プライマリ・データベースおよびロジカル・スタンバイ・データベースをスイッチオーバー用に準備します。ディクショナリ構築の完了後、ALTER DATABASE COMMIT TO SWITCHOVER 文を発行し、プライマリおよびロジカル・スタンバイ・データベースのロールを切り替えます。 この文の例は、8.3.1 項を参照してください。
RECOVER MANAGED STANDBY DATABASE [{ DISCONNECT [FROM SESSION] | USING CURRENT LOGFILE | NODELAY | UNTIL CHANGE integer }...]	この文は、フィジカル・スタンバイ・データベースに対する REDO Apply を開始して制御します。RECOVER MANAGED STANDBY DATABASE 句は、マウント、オープンまたはクローズされているフィジカル・スタンバイ・データベースで使用できます。例は、3.2.6 項の手順 4 および 7.3 項を参照してください。 **注意：**複数の句およびキーワードが廃止され、下位互換性のためにのみサポートされています。これらの句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
RECOVER MANAGED STANDBY DATABASE CANCEL	CANCEL 句は、現在のアーカイブ REDO ログ・ファイルを適用した後に、フィジカル・スタンバイ・データベースで REDO Apply を取り消します。 **注意：**複数の句およびキーワードが廃止され、下位互換性のためにのみサポートされています。これらの句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
RECOVER MANAGED STANDBY DATABASE FINISH	FINISH 句は、ターゲット・フィジカル・スタンバイ・データベースでフェイルオーバーを開始し、現行のスタンバイ REDO ログ・ファイルをリカバリします。FINISH 句を使用するのは、プライマリ・データベースに障害が発生した場合のみです。この句により、指定の遅延間隔がオーバーライドされます。 例は、8.2.2 項の手順 4 を参照してください。 **注意：**複数の句およびキーワードが廃止され、下位互換性のためにのみサポートされています。これらの句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

表 16-1 Data Guard 環境で使用される ALTER DATABASE 文（続き）

ALTER DATABASE 文	説明
`REGISTER [OR REPLACE] [PHYSICAL|LOGICAL] LOGFILE` *filespec*	手動でアーカイブ REDO ログ・ファイルを登録できます。
`RECOVER TO LOGICAL STANDBY` *new_database_name*	適用サービスに対して、データベースをロジカル・スタンバイ・データベースに変換するコマンドを発行するまでは、引き続き変更をフィジカル・スタンバイ・データベースに適用するように指示します。詳細は、4.2.4.1 項を参照してください。
`RESET DATABASE TO INCARNATION` *integer*	データベースのターゲット・リカバリ・インカネーションを、現在のインカネーションから別のインカネーションにリセットします。
`SET STANDBY DATABASE TO MAXIMIZE {PROTECTION|AVAILABILITY|PERFORMANCE}`	この句を使用して、Data Guard 構成におけるデータの保護レベルを指定します。この句は、マウント済でオープンされていないプライマリ・データベースから指定します。
`START LOGICAL STANDBY APPLY INITIAL [`*scn-value*`] ] [NEW PRIMARY` *dblink*`]`	この文は、ロジカル・スタンバイ・データベース専用です。 ロジカル・スタンバイ・データベースで SQL Apply を起動します。この文の例は、7.4.1 項を参照してください。
`{STOP|ABORT} LOGICAL STANDBY APPLY`	この文は、ロジカル・スタンバイ・データベース専用です。 ロジカル・スタンバイ・データベースで SQL Apply を正しく停止するには、`STOP` 句を使用します。SQL Apply を強制的に停止するには、`ABORT` 句を使用します。この文の例は、8.3.2 項を参照してください。
`ACTIVATE [PHYSICAL|LOGICAL] STANDBY DATABASE FINISH APPLY]`	フェイルオーバーを実行します。スタンバイ・データベースは、マウント後に、この文を使用してアクティブ化する必要があります。 **注意：**`ALTER DATABASE ACTIVATE STANDBY DATABASE` 文をフェイルオーバーに使用しないでください。データ消失が発生します。かわりに、次のベスト・プラクティスに従ってください。 ■ フィジカル・スタンバイ・データベースの場合、`ALTER DATABASE RECOVER MANAGED STANDBY DATABASE` 文で `FINISH` キーワードを使用します。これにより、ロールの推移がすばやく実行されるため、データ消失は防止されるかまたは最小限に抑えられ、他のスタンバイ・データベースが使用できなくなることはありません。 **注意：**フェイルオーバー操作は、最後にアーカイブされるログ・ファイルのヘッダーに end-of-redo マーカーを追加し、プライマリ・ロールに指定できるすべての有効な宛先（`VALID_FOR=(PRIMARY_ROLE, *_LOGFILES)` または `VALID_FOR=(ALL_ROLES, *_LOGFILES)` 属性で指定）に REDO を送信します。 ■ ロジカル・スタンバイ・データベースの場合、`ALTER DATABASE PREPARE TO SWITCHOVER` および `ALTER DATABASE COMMIT TO SWITCHOVER` 文を使用します。

16.2 ALTER SESSION 文

表 16-2 で、Data Guard に関連のある ALTER SESSION 文について説明します。

表 16-2 Data Guard 環境で使用される ALTER SESSION 文

ALTER SESSION 文	説明
ALTER SESSION [ENABLE|DISABLE] GUARD	この文は、ロジカル・スタンバイ・データベース専用です。 権限を付与されているユーザーは、この文を使用して、現在のセッションでデータベース・ガードを選択または解除できます。 詳細は、10.5.4 項を参照してください。

17

Oracle Data Guard に関連するビュー

この章では、Data Guard 環境で重要であるビューについて説明します。この章で説明されているビューは、Oracle データベースで使用可能なビューのサブセットです。

表 17-1 では、各ビューについて説明し、そのビューがフィジカル・スタンバイ・データベース、ロジカル・スタンバイ・データベース、スナップショット・スタンバイ・データベース、プライマリ・データベースのいずれに適用されるのかを示します。各ビューの詳細は、『Oracle Database リファレンス』を参照してください。

表 17-1 Data Guard 構成に関連のあるビュー

ビュー	データベース	説明
DBA_LOGSTDBY_EVENTS	ロジカルのみ	ロジカル・スタンバイ・データベースのアクティビティに関する情報が格納されます。このビューは、SQL Apply によって REDO がロジカル・スタンバイ・データベースに適用されるときに発生する障害の原因究明に役立ちます。
DBA_LOGSTDBY_HISTORY	ロジカルのみ	Data Guard 構成内のロジカル・スタンバイ・データベースに関するスイッチオーバーとフェイルオーバーの履歴を表示します。そのために、すべてのロールの推移間で、ローカル・システム上で処理または作成された REDO ログ・ストリームの順序全体が表示されます。（ロールの推移後は新規のログ・ストリームが開始され、新しいプライマリ・データベースによりログ・ストリーム順序番号が増分されます。）
DBA_LOGSTDBY_LOG	ロジカルのみ	ロジカル・スタンバイ・データベースにレジストリされているログ・ファイルを表示します。
DBA_LOGSTDBY_NOT_UNIQUE	ロジカルのみ	主キー制約または NOT NULL の一意制約がない表を識別します。
DBA_LOGSTDBY_PARAMETERS	ロジカルのみ	SQL Apply で使用されるパラメータのリストが格納されます。
DBA_LOGSTDBY_SKIP	ロジカルのみ	SQL Apply によってスキップされる表をリスト表示します。
DBA_LOGSTDBY_SKIP_TRANSACTION	ロジカルのみ	選択されているスキップ設定をリスト表示します。
DBA_LOGSTDBY_UNSUPPORTED	ロジカルのみ	サポートされないデータ型が含まれているスキーマおよび表（およびその表の列）を識別します。このビューは、ロジカル・スタンバイ・データベースの作成を準備する際に使用します。
V$ARCHIVE_DEST	プライマリ、フィジカル、スナップショットおよびロジカル	Data Guard 構成内のすべての宛先の現在の値、モードおよび状況を含めた説明を表示します。 **注意：**このビューに表示される情報は、インスタンスの停止後には保存されません。

表 17-1 Data Guard 構成に関連のあるビュー（続き）

ビュー	データベース	説明
V$ARCHIVE_DEST_STATUS	プライマリ、フィジカル、スナップショットおよびロジカル	アーカイブ REDO ログ宛先の実行時および構成用の情報を表示します。 **注意：**このビューに表示される情報は、インスタンスの停止後には保存されません。
V$ARCHIVE_GAP	フィジカル、スナップショットおよびロジカル	アーカイブ REDO ログ・ファイルのギャップの識別に役立つ情報を表示します。
V$ARCHIVED_LOG	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルのアーカイブ REDO ログ情報を、アーカイブ REDO ログ・ファイル名を含めて表示します。
V$DATABASE	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルからのデータベース情報を表示します。ファスト・スタート・フェイルオーバー（Data Guard Broker でのみ使用可能）に関する情報が含まれます。
V$DATABASE_INCARNATION	プライマリ、フィジカル、スナップショットおよびロジカル	すべてのデータベース・インカネーションに関する情報を表示します。Oracle Database では、RESETLOGS オプションを使用してデータベースがオープンされるたびに、新しいインカネーションが作成されます。V$DATABASE ビューには、現在のインカネーションのみでなく、前のインカネーションのレコードも表示されます。
V$DATAFILE	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルからのデータファイル情報を表示します。
V$DATAGUARD_CONFIG	プライマリ、フィジカル、スナップショットおよびロジカル	DB_UNIQUE_NAME および LOG_ARCHIVE_CONFIG 初期化パラメータで定義された一意のデータベース名をリスト表示します。
V$DATAGUARD_STATS	プライマリ、フィジカル、スナップショットおよびロジカル	プライマリ・データベースによって生成された REDO データのうち、スタンバイ・データベースで使用できないデータの量を表示します。すなわち、このビューを問い合せた時点でプライマリ・データベースがクラッシュした場合に失われる REDO データの量を示します。このビューは、Data Guard 構成のスタンバイ・データベースのインスタンスで問い合せることができます。このビューをプライマリ・データベースで問い合せると、列の値は消去されます。例および詳細は、8-3 ページの 8.1.2 項を参照してください。
V$DATAGUARD_STATUS	プライマリ、フィジカル、スナップショットおよびロジカル	通常はアラート・ログまたはサーバー・プロセスのトレース・ファイルへのメッセージによってトリガーされるイベントが表示および記録されます。
V$FS_FAILOVER_STATS	プライマリ	システムで発生したファスト・スタート・フェイルオーバーに関する統計を表示します。
V$LOG	プライマリ、フィジカル、スナップショットおよびロジカル	オンライン REDO ログ・ファイルのログ・ファイル情報が格納されます。
V$LOGFILE	プライマリ、フィジカル、スナップショットおよびロジカル	オンライン REDO ログ・ファイルとスタンバイ REDO ログ・ファイルに関する情報が格納されます。

表 17-1 Data Guard 構成に関連のあるビュー（続き）

ビュー	データベース	説明
V$LOG_HISTORY	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルからのログ履歴情報が格納されます。
V$LOGSTDBY_PROCESS	ロジカルのみ	SQL Apply の動作に関する動的な情報を表示します。このビューは、ロジカル・スタンバイ・データベースで SQL Apply を実行しているときのパフォーマンス上の問題を診断する場合に非常に有用です。このビューはまた、その他の問題の解決にも役立ちます。
V$LOGSTDBY_PROGRESS	ロジカルのみ	ロジカル・スタンバイ・データベースでの SQL Apply の進捗状況を表示します。
V$LOGSTDBY_STATE	ロジカルのみ	SQL Apply とロジカル・スタンバイ・データベースの実行状態に関して、V$LOGSTDBY_PROCESS および V$LOGSTDBY_STATS ビューの情報を連結します。
V$LOGSTDBY_STATS	ロジカルのみ	LogMiner 統計、現行のステータスおよび SQL Apply 時のロジカル・スタンバイ・データベースのステータス情報を表示します。SQL Apply が実行されていない場合、統計の値は消去されます。
V$LOGSTDBY_TRANSACTION	ロジカルのみ	ロジカル・スタンバイ・データベースで SQL Apply により処理中のすべてのアクティブ・トランザクションに関する情報を表示します。
V$MANAGED_STANDBY	フィジカルおよびスナップショット	フィジカル・スタンバイ・データベースに関連する Oracle データベース・プロセスの現在の状態を表示します。 **注意：**このビューに表示される情報は、インスタンスの停止後には保存されません。
V$REDO_DEST_RESP_HISTOGRAM	プライマリ	SYNC 転送用に構成された宛先に関するレスポンス時間の情報が格納されます。
V$STANDBY_LOG	フィジカル、スナップショットおよびロジカル	スタンバイ REDO ログ・ファイルのログ・ファイル情報が格納されます。

第 III 部

付録

この部は、次の付録で構成されています。

- 付録 A「Data Guard のトラブルシューティング」
- 付録 B「Data Guard 構成におけるデータベースのアップグレード」
- 付録 C「ロジカル・スタンバイ・データベースでサポートされるデータ型および DDL」
- 付録 D「Data Guard および Oralce Real Application Clusters」
- 付録 E「カスケードされた宛先」
- 付録 F「Recovery Manager を使用したスタンバイ・データベースの作成」
- 付録 G「アーカイブ・トレースの設定」

A

Data Guard のトラブルシューティング

この付録は、スタンバイ・データベースのトラブルシューティングのヘルプとしてご利用いただけます。この項は、次の項目で構成されています。

A.1 一般的な問題

スタンバイ・データベースの使用時に発生する問題の原因には、次のようなものがあります。

- ALTER DATABASE 文によるデータファイル名の変更
- スタンバイ・データベースがプライマリ・データベースから REDO データを受信しない
- フィジカル・スタンバイ・データベースをマウントできない

A.1.1 ALTER DATABASE 文によるデータファイル名の変更

STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定すると、スタンバイ・サイトのデータファイルを改名できません。STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定した場合、次の SQL 文は使用できなくなります。

- ALTER DATABASE RENAME
- ALTER DATABASE ADD/DROP LOGFILE
- ALTER DATABASE ADD/DROP STANDBY LOGFILE MEMBER
- ALTER DATABASE CREATE DATAFILE AS

スタンバイ・データベースでこれらの文のいずれかを使用しようとすると、エラーが表示されます。次に例を示します。

```
SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/t_db2.log' to 'dummy';
alter database rename file '/disk1/oracle/oradata/payroll/t_db2.log' to 'dummy'
*
ERROR at line 1:
ORA-01511: error in renaming log/datafiles
ORA-01270: RENAME operation is not allowed if STANDBY_FILE_MANAGEMENT is auto
```

フィジカル・スタンバイ・データベースにデータファイルを追加する方法は、9.3.1 項を参照してください。

A.1.2 スタンバイ・データベースがプライマリ・データベースから REDO データを受信しない

スタンバイ・サイトが REDO データを受信していない場合は、V$ARCHIVE_DEST ビューを問い合せて、エラー・メッセージをチェックします。たとえば、次のような問合せを入力します。

```
SQL> SELECT DEST_ID "ID",
  2> STATUS "DB_status",
  3> DESTINATION "Archive_dest",
  4> ERROR "Error"
  5> FROM V$ARCHIVE_DEST WHERE DEST_ID <=5;

ID DB_status Archive_dest                  Error
-- --------- ------------------------------ -----------------------------------
 1  VALID    /vobs/oracle/work/arc_dest/arc
 2  ERROR    standby1                       ORA-16012: Archivelog standby database identifier mismatch
 3  INACTIVE
 4  INACTIVE
 5  INACTIVE
5 rows selected.
```

問合せの出力によって解決しない場合は、次の問題リストを確認します。次のいずれかの条件に該当する場合、REDO 転送サービスはスタンバイ・データベースに REDO データを転送できません。

- プライマリ・データベースの tnsnames.ora ファイル内でスタンバイ・インスタンスのサービス名が正しく構成されていない場合。
- プライマリ・データベースの LOG_ARCHIVE_DEST_*n* パラメータで指定された Oracle Net サービス名が不適切な場合。

- スタンバイ・データベースの LOG_ARCHIVE_DEST_STATE_*n* パラメータが値 ENABLE に設定されていない場合。
- listener.ora ファイルがスタンバイ・データベース用に正しく構成されていない場合。
- スタンバイ・サイトでリスナーが開始されていない場合。
- スタンバイ・インスタンスが起動されていない場合。
- スタンバイ・アーカイブ先をプライマリ SPFILE またはテキスト初期化パラメータ・ファイルに追加したが、その変更をまだ使用可能にしていない場合。
- REDO 転送の認証が正しく構成されていない場合。REDO 転送の認証の構成要件は、3.1.2 項を参照してください。
- スタンバイ・データベースのベースとして無効なバックアップを使用した場合（たとえば、間違ったデータベースからのバックアップを使用、または正しい方法でスタンバイ制御ファイルを作成しなかったなど）。

A.1.3 フィジカル・スタンバイ・データベースをマウントできない

スタンバイ制御ファイルが、ALTER DATABASE CREATE [LOGICAL] STANDBY CONTROLFILE ... 文または Recovery Manager コマンドを使用して作成されていない場合、スタンバイ・データベースをマウントすることはできません。次のタイプの制御ファイル・バックアップは使用できません。

- オペレーティング・システムで作成されたバックアップ
- PHYSICAL STANDBY または LOGICAL STANDBY オプションのない ALTER DATABASE 文を使用して作成されたバックアップ

A.2 ログ・ファイル宛先の障害

MANDATORY 宛先に REOPEN を指定した場合、REDO 転送サービスは、REDO データを正常に転送できなかったとき、プライマリ・データベースを停止します。

MAX_FAILURE 属性を使用する場合は、REOPEN 属性は必須です。例 A-1 に、再試行時間を 5 秒に設定し、再試行回数を 3 回に制限する方法を示します。

例 A-1 再試行時間と制限の設定

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=5 MAX_FAILURE=3'
```

LOG_ARCHIVE_DEST_*n* パラメータの ALTERNATE 属性を使用して代替アーカイブ先を指定します。スタンバイ・データベースへの REDO データの転送に失敗した場合は、代替アーカイブ先を使用できます。転送に失敗したときに、REOPEN 属性が指定されていなかった場合、または MAX_FAILURE 属性のしきい値を超えた場合、REDO 転送サービスは、次のアーカイブ操作時に代替先への REDO データの転送を試みます。

元のアーカイブ先で障害が発生したときに、元のアーカイブ先が自動的に代替アーカイブ先に変更されることを防ぐには、NOALTERNATE 属性を使用します。

例 A-2 は、エラー発生時に、単一、必須のローカル宛先が異なる宛先へ自動的にフェイルオーバーするように初期化パラメータを設定する方法を示します。

例 A-2 代替宛先の指定

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

LOG_ARCHIVE_DEST_1 の宛先で障害が発生した場合、アーカイブ・プロセスは、プライマリ・データベースでの次のログ・ファイル・スイッチで LOG_ARCHIVE_DEST_2 の宛先に自動的に切り替えます。

A.3 ロジカル・スタンバイ・データベース障害の処理

ロジカル・スタンバイ・データベース障害を処理するための重要なツールは、DBMS_LOGSTDBY.SKIP_ERROR プロシージャです。表の重要度に基づいて、次のいずれかを実行できます。

- 表や特定の DDL に関する障害を無視する。
- ストアド・プロシージャをフィルタに関連付ける。これによって、文のスキップ、文の実行または代用文の実行について実行時に決定できます。

これらの処理のいずれかを実行すると、SQL Apply の停止を回避できます。存在している問題は、後で DBA_LOGSTDBY_EVENTS ビューを問い合せて検索し、訂正できます。DBMS_LOGSTDBY パッケージを PL/SQL コールアウト・プロシージャとともに使用する方法の詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

A.4 フィジカル・スタンバイ・データベースへのスイッチオーバーの問題

通常、第 8 章で説明した手順に従うと、スイッチオーバーは正常に行われます。ただし、スイッチオーバーが失敗した場合でも、次の項を読むと、問題の解決に役立ちます。

- REDO データが転送されていないためスイッチオーバーできない
- SQL セッションがアクティブなためスイッチオーバーできない
- ユーザー・セッションがアクティブなためスイッチオーバーできない
- ORA-01102 エラーによりスイッチオーバーできない
- スイッチオーバー後に REDO データが適用されない
- 失敗したスイッチオーバーをロールバックして最初からやりなおす

A.4.1 REDO データが転送されていないためスイッチオーバーできない

スイッチオーバーが正常に完了しない場合は、V$ARCHIVED_LOG ビューの SEQUENCE# 列を問い合せて、元のプライマリ・データベースから転送された最後の REDO データが、スタンバイ・データベースに適用されているかどうかを確認します。最後の REDO データがスタンバイ・データベースに転送されていない場合は、その REDO データが含まれるアーカイブ REDO ログ・ファイルを元のプライマリ・データベースから古いスタンバイ・データベースに手動でコピーし、SQL ALTER DATABASE REGISTER LOGFILE *file_specification* 文を使用して登録します。次に、適用サービスを開始すると、アーカイブ REDO ログ・ファイルが自動的に適用されます。V$DATABASE ビューの SWITCHOVER_STATUS 列を問い合せます。SWITCHOVER_STATUS 列の値 TO PRIMARY は、プライマリ・ロールへのスイッチオーバーが可能になったことを示します。

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
-----------------
TO PRIMARY
1 row selected
```

V$DATABASE ビューの SWITCHOVER_STATUS 列に対するその他の有効な値の詳細は、第 17 章を参照してください。

スイッチオーバーを続行するには、8.2.1 項（フィジカル・スタンバイ・データベースの場合）または 8.3.1 項（ロジカル・スタンバイ・データベースの場合）の説明に従って、ターゲットのスタンバイ・データベースをプライマリ・ロールに切り替える操作を再度実行します。

A.4.2 SQL セッションがアクティブなためスイッチオーバーできない

ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY文にWITH SESSION SHUTDOWN 句を組み込んでいない場合、アクティブな SQL セッションがあると、スイッチオーバーを継続できません。アクティブな SQL セッションには、他の Oracle Database プロセスが含まれていることがあります。

セッションがアクティブのときは、スイッチオーバーの試行が次のエラー・メッセージを伴って失敗します。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY;
ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY *
ORA-01093: ALTER DATABASE CLOSE only permitted with no sessions connected
```

処置 : V$SESSION ビューを問い合せ、エラーの原因となっているプロセスを判断します。次に例を示します。

```
SQL> SELECT SID, PROCESS, PROGRAM FROM V$SESSION
  2> WHERE TYPE = 'USER'
  3>  AND SID <> (SELECT DISTINCT SID FROM V$MYSTAT);
SID        PROCESS   PROGRAM
---------  --------  ------------------------------------------------
        7      3537  oracle@nhclone2 (CJQ0)
       10
       14
       16
       19
       21
 6 rows selected.
```

この例では、JOB_QUEUE_PROCESSES パラメータが CJQ0 プロセス・エントリに対応しています。ジョブ・キュー・プロセスはユーザー・プロセスなので、スイッチオーバーの発生を妨げるのは SQL セッションであると考えられます。プロセスまたはプログラム情報のないエントリは、ジョブ・キュー・コントローラによって開始されるスレッドです。

次の SQL 文を使用して、JOB_QUEUE_PROCESSES パラメータが設定されていることを確認してください。

```
SQL> SHOW PARAMETER JOB_QUEUE_PROCESSES;
NAME                          TYPE      VALUE
----------------------------- -------   --------------------
job_queue_processes           integer   5
```

さらに、パラメータを 0（ゼロ）に設定してください。次に例を示します。

```
SQL> ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
Statement processed.
```

JOB_QUEUE_PROCESSES は動的パラメータであるため、値を変更した際にインスタンスを再起動しなくても、変更を即時に有効にできます。これで、スイッチオーバー手順を再試行できます。

初期化パラメータ・ファイル内のパラメータは変更しないでください。インスタンスを停止し、スイッチオーバーが完了した後で再起動すると、パラメータが元の値にリセットされます。これは、プライマリ・データベースとフィジカル・スタンバイ・データベースの両方に適用されます。

表 A-1 に、スイッチオーバーを妨げる共通のプロセスと、実行する必要のある対処措置をまとめます。

表 A-1 スイッチオーバーを妨げる共通のプロセス

プロセスのタイプ	プロセスの説明	対処措置
CJQ0	Job Queue Scheduler Process	JOB_QUEUE_PROCESSES 動的パラメータを値 0（ゼロ）に変更してください。変更は、インスタンスを再起動しなくても即時に有効になります。
QMN0	Advanced Queue Time Manager	AQ_TM_PROCESSES 動的パラメータを値 0（ゼロ）に変更してください。変更は、インスタンスを再起動しなくても即時に有効になります。
DBSNMP	Oracle Enterprise Manager Management Agent	オペレーティング・システムのプロンプトから、emctl stop agent コマンドを発行してください。

A.4.3 ユーザー・セッションがアクティブなためスイッチオーバーできない

スイッチオーバーに失敗し、エラー「ORA-01093: ALTER DATABASE CLOSE は接続中のセッションがない場合にのみ実行できます」が戻された場合、通常これは、ALTER DATABASE COMMIT TO SWITCHOVER 文が暗黙的にデータベースのクローズを実行したときに、その他のユーザー・セッションがデータベースに接続されていてクローズできなかったことが原因と考えられます。

このエラーが発生した場合は、データベースに接続されているユーザー・セッションをすべて切断します。まだアクティブなセッションを確認するには、次のように V$SESSION ビューを問い合せます。

```
SQL> SELECT SID, PROCESS, PROGRAM FROM V$SESSION;
```

A.4.4 ORA-01102 エラーによりスイッチオーバーできない

スタンバイ・データベースとプライマリ・データベースが同じサイトに存在するとします。ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY 文および ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY 文の両方が正常に実行された後、フィジカル・スタンバイ・データベースとプライマリ・データベースを停止し、再起動します。

注意： フィジカル・スタンバイ・データベースが起動後に読取り専用モードでオープンされていない場合、停止して再起動する必要はありません。

ただし、2 つ目のデータベースの起動はエラー・メッセージ「ORA-01102 データベースを排他モードでマウントすることができません。」を伴って失敗します。

スタンバイ・データベース（つまり元のプライマリ・データベース）で使用される初期化パラメータ・ファイルに DB_UNIQUE_NAME パラメータを設定していないと、スイッチオーバー中にこの現象が発生することがあります。スタンバイ・データベースの DB_UNIQUE_NAME パラメータが設定されていない場合、スタンバイ・データベースとプライマリ・データベースの両方が同じマウント・ロックを使用するため、2 つ目のデータベースの起動時に ORA-01102 エラーが発生します。

処置：DB_UNIQUE_NAME=*unique_database_name* をスタンバイ・データベースが使用する初期化パラメータ・ファイルに追加して、スタンバイ・データベースとプライマリ・データベースを停止し、再起動します。

A.4.5 スイッチオーバー後に REDO データが適用されない

スイッチオーバー後にアーカイブ REDO ログ・ファイルが新しいスタンバイ・データベースに適用されません。

これは、スイッチオーバー後、環境パラメータまたは初期化パラメータが適切に設定されていなかったことが原因と考えられます。

処置 :

- 新しいプライマリ・サイトの tnsnames.ora ファイルおよび新しいスタンバイ・サイトの listener.ora ファイルを調べます。スタンバイ・サイトにはリスナーのエントリ、プライマリ・サイトにはそれに対応するサービス名が必要です。
- リスナーがまだ起動されていない場合は、スタンバイ・サイトで起動します。
- プライマリ・サイトからスタンバイ・サイトに REDO データを正しく転送するための、LOG_ARCHIVE_DEST_*n* 初期化パラメータが設定されているかどうかを調べます。たとえば、プライマリ・サイトで V$ARCHIVE_DEST 固定ビューを次のように問い合せます。

  ```
  SQL> SELECT DEST_ID, STATUS, DESTINATION FROM V$ARCHIVE_DEST;
  ```

  スタンバイ・サイトに対応するエントリが見つからない場合、LOG_ARCHIVE_DEST_*n* 初期化パラメータおよび LOG_ARCHIVE_DEST_STATE_*n* 初期化パラメータを設定する必要があります。
- スタンバイ・サイトに STANDBY_ARCHIVE_DEST 初期化パラメータおよび LOG_ARCHIVE_FORMAT 初期化パラメータを正しく設定し、アーカイブ REDO ログ・ファイルが適切な場所に適用されるようにします。(STANDBY_ARCHIVE_DEST パラメータは廃止されており、下位互換性のためにのみサポートされています。)
- スタンバイ・サイトで DB_FILE_NAME_CONVERT 初期化パラメータおよび LOG_FILE_NAME_CONVERT 初期化パラメータを設定します。プライマリ・サイトで作成された新しいデータファイルがスタンバイ・サイトに自動的に追加されるようにするには、STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定します。

A.4.6 失敗したスイッチオーバーをロールバックして最初からやりなおす

フィジカル・スタンバイ・データベースでエラーが発生し、スイッチオーバーを続行できない場合は、次の手順に従って、新しいフィジカル・スタンバイ・データベースをプライマリ・ロールに戻すことができます。

1. 新しいスタンバイ・データベース（元のプライマリ）に接続し、次の文を発行して、このスタンバイ・データベースをプライマリ・ロールに戻します。

   ```
   SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
   ```

   この文が正常に実行された場合は、必要に応じてデータベースを停止し、再起動します。再起動すると、データベースはプライマリ・データベース・ロールで実行されるため、それ以降の手順を実行する必要はありません。

   この文が正常に実行されなかった場合は、手順 3 に進んでください。

2. プライマリからフィジカル・スタンバイにロールを変更するスイッチオーバーを開始したときに、トレース・ファイルがログ・ディレクトリに書き込まれています。このトレース・ファイルには、元のプライマリ制御ファイルを再作成するために必要な SQL 文が含まれています。トレース・ファイルの位置を特定し、SQL 文を一時ファイルに抽出します。SQL*Plus から一時ファイルを実行します。これによって、新しいスタンバイ・データベースはプライマリ・ロールに戻されます。
3. 元のフィジカル・スタンバイ・データベースを停止します。
4. 新しいスタンバイ制御ファイルを作成します。このファイルは、プライマリ・データベースとフィジカル・スタンバイ・データベースの再同期化に必要です。フィジカル・スタンバイ制御ファイルを元のフィジカル・スタンバイ・システムにコピーします。フィジカル・スタンバイ制御ファイルの作成方法は、3.2.2 項を参照してください。

5. 元のフィジカル・スタンバイ・インスタンスを再起動します。

   この手順が正常終了し、アーカイブ・ギャップ管理が使用可能になると、FAL プロセスが起動し、欠落したアーカイブ REDO ログ・ファイルをフィジカル・スタンバイ・データベースに再アーカイブします。プライマリ・データベース上でログ・スイッチを強制実行し、プライマリ・データベースとフィジカル・スタンバイ・データベースの両方のアラート・ログを調べて、アーカイブ REDO ログ・ファイルの順序番号が正しいことを確認します。

   アーカイブ・ギャップ管理の詳細は 6.3.3.1 項を、トレース・ファイルの場所については付録 G を参照してください。

6. スイッチオーバーを再度実行します。

   この時点で、Data Guard 構成は初期状態にロールバックされています。最初に失敗したスイッチオーバーの問題をすべて解決した後、スイッチオーバー操作を再度実行できます。

A.5 ロジカル・スタンバイ・データベースへのスイッチオーバーの問題

ロジカル・スタンバイ・データベースに関連するスイッチオーバー操作には、通常、準備とコミットの 2 つのフェーズがあります。これに対する例外は、ロジカル・スタンバイ・データベースを使用した Oracle ソフトウェアのローリング・アップグレードの場合、または Data Guard Broker を使用している場合です。ロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行中、または Data Guard Broker で開始されたスイッチオーバー操作の際に障害が発生した場合は、A.5.2 項に直接進んでください。

注意： Data Guard 構成のすべてのデータベースで、フラッシュバック・データベースを有効にすることをお薦めします。この項の手順では、使用する Data Guard 構成内のすべてのデータベースで、フラッシュバック・データベースが有効にされていると想定しています。

A.5.1 スイッチオーバー操作の準備フェーズ中の障害

スイッチオーバー操作の準備フェーズ中に障害が発生した場合、スイッチオーバーを取り消して、スイッチオーバー操作を再度初めから行います。

A.5.1.1 プライマリ・データベースの準備中の障害

`ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY` 文の実行中に障害が発生した場合、プライマリ・データベースで次の SQL 文を発行してスイッチオーバーの準備フェーズを取り消します。

```
SQL> ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY CANCEL;
```

これで、スイッチオーバー操作を再度初めから行うことができます。

A.5.1.2 ロジカル・スタンバイ・データベースの準備中の障害

`ALTER DATABASE PREPARE TO SWITCHOVER TO PRIMARY` 文の実行中に障害が発生した場合、プライマリ・データベースおよびターゲット・スタンバイ・データベースで準備操作を取り消す必要があります。次の手順に従ってください。

1. プライマリ・データベースで、スイッチオーバーの準備のために発行した文を取り消します。

   ```
   SQL> ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY CANCEL;
   ```

2. スイッチオーバーのターゲットであるロジカル・スタンバイ・データベースで、スイッチオーバーの準備のために発行した文を取り消します。

```
SQL> ALTER DATABASE PREPARE TO SWITCHOVER TO PRIMARY CANCEL;
```

これで、スイッチオーバー操作を再度初めから行うことができます。

A.5.2 スイッチオーバー操作のコミット・フェーズ中の障害

スイッチオーバーのコミットでは単一の SQL 文のみを使用しますが、内部的には多数の操作が実行されます。実行する必要がある修正処理は、エラー発生時のスイッチオーバー操作のコミットの状態によって異なります。

A.5.2.1 元のプライマリ・データベースの変換での障害

ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY 文の実行中に障害が発生した場合、次の手順を実行します。

1. 元のプライマリ・データベースの V$DATABASE 固定ビューの DATABASE_ROLE 列をチェックします。

```
SQL> SELECT DATABASE_ROLE FROM V$DATABASE;
```

   - 列に LOGICAL STANDBY の値が含まれる場合、スイッチオーバー操作は完了していますが、スイッチオーバー後のタスクで障害が発生しています。この場合、データベースを停止して再オープンすることをお薦めします。
   - 列に PRIMARY の値が含まれる場合、手順 2 に進みます。

2. 元のプライマリで次の問合せを実行します。

```
SQL> SELECT COUNT(*) FROM SYSTEM.LOGSTDBY$PARAMETERS
  2> WHERE NAME = 'END_PRIMARY';
```

   - 問合せで 0 が戻される場合、プライマリの状態は、スイッチオーバーのコミット・コマンドが発行される前と同一です。修正処理を実行する必要はありません。スイッチオーバー操作のコミットに進むか、A.5.1.2 項で説明するようにスイッチオーバー操作を取り消します。
   - 問合せで 1 が戻される場合、プライマリは一貫性がない状態のため、手順 3 に進む必要があります。

3. インスタンス化された既存のまたは新しいロジカル・スタンバイ・データベースにより保護されるための機能が維持されるよう、元のプライマリ・データベースで対処措置を実行します。

   スイッチオーバー操作のコミット中に発生したエラーの基となる原因を修正して SQL 文（ALTER DTABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY）を再発行します。次の手順を実行することもできます。

   a. スイッチオーバー・コマンドのコミットを開始したインスタンスのアラート・ログから、元のプライマリへのフラッシュバックに必要な SCN を判別します。この情報は、SQL 文 ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY の後に表示されます。

```
LOGSTDBY: Preparing the COMMIT TO SWITCHOVER TO LOGICAL STANDBY DDL at scn
[flashback_scn].
```

   b. プライマリ・データベースのすべてのインスタンスを停止します。

```
SQL> SHUTDOWN IMMEDIATE;
```

   c. プライマリ・データベースを排他モードでマウントします。

```
SQL> STARTUP MOUNT;
```

d. データベースをアラート・ログから取得した SCN までフラッシュバックします。

```
SQL> FLASHBACK DATABASE TO BEFORE SCN <flashback_scn>;
```

e. プライマリ・データベースをオープンします。

```
SQL> STARTUP;
```

f. 元のプライマリ・データベースのデータベース・ガードを低くします。

```
SQL> ALTER DATABASE GUARD NONE;
```

この時点で、プライマリの状態はスイッチオーバーのコミット・コマンドが発行される前と同一です。修正処理を実行する必要はありません。スイッチオーバー操作のコミットに進むか、A.5.1.1 項で説明するようにスイッチオーバー操作を取り消します。

A.5.2.2 ターゲット・ロジカル・スタンバイ・データベースの変換での障害

ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY 文の実行中に障害が発生した場合、次の手順を実行します。

1. ターゲット・スタンバイ・データベースの V$DATABASE 固定ビューの DATABASE_ROLE 列をチェックします。

   ```
   SQL> SELECT DATABASE_ROLE FROM V$DATABASE;
   ```

   - 列に PRIMARY の値が含まれる場合、スイッチオーバー操作は完了していますが、スイッチオーバー後のタスクで障害が発生しています。この場合、次の手順を実行します。
     a. データベースを停止して、再オープンします。
     b. ALTER DATABASE GUARD NONE コマンドを発行して、データベースに対する書込み制限を削除します。
   - 列に LOGICAL STANDBY の値が含まれる場合、手順 2 に進みます。

2. ターゲット・ロジカル・スタンバイで次の問合せを実行します。

   ```
   SQL> SELECT COUNT(*) FROM SYSTEM.LOGSTDBY$PARAMETERS
     2> WHERE NAME = 'BEGIN_PRIMARY';
   ```

   - 問合せで 0 が戻される場合、ロジカル・スタンバイの状態はスイッチオーバーのコミット・コマンドが発行される前と同一です。修正処理を実行する必要はありません。スイッチオーバー操作のコミットに進むか、A.5.1.2 項で説明するようにスイッチオーバー操作を取り消します。
   - 問合せで 1 が戻される場合、ロジカル・スタンバイは一貫性がない状態のため、手順 3 に進む必要があります。

3. 新しいプライマリまたは別の新しいプライマリに対するバイスタンダになるための機能が維持されるよう、ロジカル・スタンバイで対処措置を実行します。

   スイッチオーバー操作のコミット中に発生したエラーの基となる原因を修正して SQL 文（ALTER DTABASE COMMIT TO SWITCHOVER TO PRIMARY）を再発行します。次の手順を実行して、スイッチオーバーのコミットを試行する直前の一貫性の時点までロジカル・スタンバイ・データベースをフラッシュバックすることもできます。

   a. スイッチオーバーのコミット・コマンドを開始したインスタンスのアラート・ログから、ロジカル・スタンバイへのフラッシュバックに必要な SCN を判別します。この情報は、SQL 文 ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY の後に表示されます。

   ```
   LOGSTDBY: Preparing the COMMIT TO SWITCHOVER TO PRIMARY DDL at scn
   [flashback_scn].
   ```

b. ターゲット・スタンバイ・データベースのすべてのインスタンスを停止します。

```
SQL> SHUTDOWN IMMEDIATE;
```

c. ターゲット・ロジカル・スタンバイ・データベースをマウントします。

```
SQL> STARTUP MOUNT;
```

d. 必要な SCN までターゲット・ロジカル・スタンバイをフラッシュバックします。

```
SQL> FLASHBACK DATABASE TO BEFORE SCN <flashback_scn>;
```

e. データベースをオープンします（Oracle RAC の場合、すべてのインスタンスをオープンします）。

```
SQL> STARTUP OPEN;
```

この時点で、ターゲット・スタンバイの状態はスイッチオーバーのコミット・コマンドが発行される前と同一です。対処措置を実行する必要はありません。スイッチオーバー操作のコミットに進みます。

A.6 SQL Apply が停止した場合の処置

適用サービスでは、サポートされない DML 文、DDL 文およびオラクル社が提供するパッケージを、SQL Apply が実行されているロジカル・スタンバイ・データベースに適用できません。

サポートされない文やパッケージが検出されると、SQL Apply は停止します。表 A-2 で説明する処理を実行して問題を解決した後、ロジカル・スタンバイ・データベースで SQL Apply を再開してください。

表 A-2 一般的な SQL Apply エラーの解決方法

エラー	解決方法
サポートされない文またはオラクル社が提供するパッケージが検出された可能性を示すエラー	DBA_LOGSTDBY_EVENTS ビューで、最後の文を検索してください。この結果、SQL Apply の失敗の原因となった文とエラーが表示されます。不適切な SQL 文が原因で SQL Apply に失敗した場合は、その文とエラーに関する情報とともにトランザクション情報を表示できます。このトランザクション情報は、問題の原因を正確に判断するために LogMiner ツールで使用できます。
データベース管理を要求するエラー（特定の表領域内の領域不足など）	問題を解決した後、ALTER DATABASE START LOGICAL STANDBY APPLY 文を使用して SQL Apply を再開してください。
不適切な SQL 文の入力によるエラー（不適切なスタンバイ・データベースのファイル名が表領域文に入力された場合など）	適切な SQL 文を入力した後、DBMS_LOGSTDBY.SKIP_TRANSACTION プロシージャを使用して、次回 SQL Apply が実行されたときに不適切な文が無視されるようにしてください。その後、ALTER DATABASE START LOGICAL STANDBY APPLY 文を使用して SQL Apply を再開してください。
不適切な SKIP パラメータの設定によるエラー（特定の表で全 DML がスキップされるように指定したが、CREATE、ALTER および DROP TABLE の各文がスキップされるように指定していないなど）	DBMS_LOGSTDBY.SKIP('TABLE','schema_name', 'table_name',null) プロシージャを発行した後、SQL Apply を再開してください。

障害の原因を特定するために DBA_LOGSTDBY_EVENTS ビューを問い合せる方法は、第 17 章を参照してください。

A.7 REDO データ転送のネットワーク調整

最適なパフォーマンスを得るには、REDO 転送サービスで使用される各 Oracle Net 接続記述子で、Oracle Net SDU パラメータを 32KB に設定します。

次の例は、データベース初期化パラメータ・ファイル内のリモート宛先 netserv を定義する部分を示します。

```
LOG_ARCHIVE_DEST_3='SERVICE=netserv'
```

次の例は、tnsnames.ora ファイル内のサービス名の定義を示します。

```
netserv=(DESCRIPTION=(SDU=32768)(ADDRESS=(PROTOCOL=tcp)(HOST=host) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=srvc)))
```

次の例は、listener.ora ファイル内の定義を示します。

```
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
(HOST=host)(PORT=1521))))

SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(SDU=32768)(SID_NAME=sid)
(GLOBALDBNAME=srvc)(ORACLE_HOME=/oracle)))
```

待機時間が長いかまたは帯域幅が大きいネットワーク・リンクを使用してリモート・サイトへのアーカイブを行う場合は、Oracle Net プロファイル・パラメータの SQLNET.SEND_BUF_SIZE および SQLNET.RECV_BUF_SIZE を使用して、ネットワークの送受信 I/O バッファのサイズを大きくすることでパフォーマンスを改善できます。

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

A.8 スタンバイ・データベースのディスクのパフォーマンスが遅い

ファイル・システム上の非同期 I/O がパフォーマンス上の問題を示している場合は、ダイレクト I/O オプションを使用してファイル・システムをマウントするか、または FILESYSTEMIO_OPTIONS=SETALL 初期化パラメータを設定します。最大 I/O のサイズ設定は、1MB です。

A.9 プライマリ・データベースの停止を回避するにはログ・ファイルを一致させる必要がある

構成内の 1 つ以上のスタンバイ・データベースでスタンバイ REDO ログを構成した場合は、各スタンバイ・データベースのスタンバイ REDO ログ・ファイルのサイズが、プライマリ・データベースのオンライン REDO ログ・ファイルのサイズと正確に一致していることを確認してください。

ログ・スイッチが発生したときに、プライマリ・データベースに新しい現行のオンライン REDO ログ・ファイルのサイズと一致する使用可能なスタンバイ REDO ログ・ファイルがない場合、次のようになります。

- プライマリ・データベースが最大保護モードで作動しているときは停止します。

  または

- スタンバイ・データベース上の RFS プロセスにより、スタンバイ・データベースにアーカイブ REDO ログ・ファイルが作成され、次のメッセージがアラート・ログに書き込まれます。

  ```
  No standby log files of size <#> blocks available.
  ```

たとえば、プライマリ・データベースで、ログ・ファイルが 100K のオンライン REDO ログ・グループを 2 つ使用する場合、スタンバイ・データベースは、ログ・ファイル・サイズが 100K のスタンバイ REDO ログ・グループを 3 つ持ちます。

また、REDO ログ・グループをプライマリ・データベースに追加した場合は、対応するスタンバイ REDO ログ・グループをスタンバイ・データベースに追加する必要があります。これにより、プライマリ・データベースが悪影響を受ける可能性が低くなります。これは、ログ・スイッチが発生したときに、必要なサイズのスタンバイ REDO ログ・ファイルが使用できないためです。

A.10 ロジカル・スタンバイ・データベースのトラブルシューティング

この項は、次の項目で構成されています。

A.10.1 エラーのリカバリ

ロジカル・スタンバイ・データベースでは、ユーザー表、順序およびジョブがメンテナンスされます。他のオブジェクトをメンテナンスするには、REDO データ・ストリームにある DDL 文を再発行する必要があります。

SQL Apply に障害が発生した場合、エラーは DBA_LOGSTDBY_EVENTS 表に記録されます。次の各項では、このようなエラーのリカバリ方法を説明します。

A.10.1.1 ファイル仕様が含まれている DDL トランザクション

DDL 文は、プライマリ・データベースとロジカル・スタンバイ・データベースでは同じ方法で実行されます。基礎となるファイル構造が両方のデータベースで同じ場合、DDL はスタンバイ・データベース上で予想どおりに実行されます。

エラーの原因が、ロジカル・スタンバイ・データベース環境に適合しないファイル仕様を含んだ DDL トランザクションにある場合は、次の手順を実行して問題を解決してください。

1. ALTER SESSION DISABLE GUARD 文を使用してデータベース・ガードをバイパスし、ロジカル・スタンバイ・データベースへの変更を可能にします。

   ```
   SQL> ALTER SESSION DISABLE GUARD;
   ```

2. 正しいファイル仕様を使用して DDL 文を実行し、データベース・ガードを再び使用可能にします。次に例を示します。

   ```
   SQL> ALTER TABLESPACE t_table ADD DATAFILE '/dbs/t_db.f' SIZE 100M REUSE;
   SQL> ALTER SESSION ENABLE GUARD;
   ```

3. ロジカル・スタンバイ・データベースで SQL Apply を開始し、障害が発生したトランザクションをスキップします。

   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE
     2> SKIP FAILED TRANSACTION;
   ```

トランザクション障害の原因となった問題を修正し、トランザクションをスキップせずに SQL Apply を再開できる場合があります。たとえば、使用可能領域がすべて使用された場合などです。(DDL トランザクションをスキップするときに、プライマリおよびロジカル・スタンバイ・データベースの内容にずれを生じさせないようにしてください。可能な場合は、スキップされたトランザクションのかわりに補足用トランザクションを手動で実行する必要があります。)

次の例に、SQL Apply の停止、エラーの修正および SQL Apply の再開を示します。

```
SQL> SET LONG 1000
SQL> ALTER SESSION SET NLS_DATE_FORMAT  = 'DD-MON-YY HH24:MI:SS';

Session altered.

SQL> SELECT EVENT_TIME, COMMIT_SCN, EVENT, STATUS FROM DBA_LOGSTDBY_EVENTS;

EVENT_TIME              COMMIT_SCN
------------------ ---------------
EVENT
--------------------------------------------------------------------------------
STATUS
--------------------------------------------------------------------------------
22-OCT-03 15:47:58

ORA-16111: log mining and apply setting up

22-OCT-03 15:48:04          209627
insert into "SCOTT"."EMP"
values
   "EMPNO" = 7900,
   "ENAME" = 'ADAMS',
   "JOB" = 'CLERK',
   "MGR" IS NULL,
   "HIREDATE" = TO_DATE('22-OCT-03', 'DD-MON-RR'),
   "SAL" = 950,
   "COMM" IS NULL,
   "DEPTNO" IS NULL
ORA-01653: unable to extend table SCOTT.EMP by %200 bytes in tablespace T_TABLE
```

この例で、ORA-01653 メッセージは、表領域がいっぱいで表を拡張できないことを示しています。問題を修正するには、新しいデータファイルを表領域に追加します。次に例を示します。

```
SQL> ALTER TABLESPACE t_table ADD DATAFILE '/dbs/t_db.f' SIZE 60M;
Tablespace altered.
```

次に、SQL Apply を再開します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
Database altered.
```

SQL Apply を再開すると、障害が発生したトランザクションが再実行され、ロジカル・スタンバイ・データベースに適用されます。

A.10.1.2 DML 障害のリカバリ

DML 障害のフィルタ処理には、SKIP_TRANSACTION プロシージャを使用しないでください。また、スキップされたイベント表にある DML のみでなく、トランザクションに関連しているすべての DML についても注意が必要です。これが複数の表の原因となります。

DML 障害は通常、特定の表に関する問題を示しています。たとえば、障害が即時に解決できない記憶域の不足に関するエラーであるとします。次の手順は、この問題に応答する 1 つの方法を示しています。

1. 表をスキップ・リストに追加することによって、トランザクションではなく、表をバイパスします。

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.SKIP('DML','SCOTT','EMP');
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
   ```

   この時点から、SCOTT.EMP 表に関する DML アクティビティは適用されません。表は、記憶域の問題を解決した後で修正できます。ただし、表を修正するには、DBMS_LOGSTDBY パッケージ内のプロシージャを実行するための管理者権限がある、プライマリ・データベースへのデータベース・リンクを設定していることが必要です。

2. プライマリ・データベースへのデータベース・リンクを使用して、ローカルな SCOTT.EMP 表を削除し、表を再作成して、データをスタンバイ・データベースに転送します。

   ```
   SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE('SCOTT','EMP','PRIMARYDB');
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
   ```

3. 新規にインスタンス化された表とデータベースの残りの部分にまたがってビューの一貫性が得られるように、SQL Apply がプライマリ・データベースと一致するまで待機してから、この表を問い合せます。詳細な例は、10.5.5 項「ロジカル・スタンバイ・データベースでの表の追加または再作成」を参照してください。

A.10.2 SQL*Loader セッションのトラブルシューティング

Oracle SQL*Loader には、様々なソースから Oracle Database にデータをロードする方法が用意されています。この項では、SQL*Loader ユーティリティの一部の SQL Apply 関連機能について説明します。

選択したデータ・ロード方法に関係なく、SQL*Loader 制御ファイルには、APPEND および REPLACE キーワードを介して、新規データのロード先となる Oracle 表の現在の内容に対して実行する作業を示す指示が含まれています。次の例に、これらのキーワードを LOAD_STOK という表に使用する方法を示します。

- APPEND キーワードを使用すると、新規にロードされるデータは LOAD_STOK 表の内容に追加されます。

  ```
  LOAD DATA
  INTO TABLE LOAD_STOK APPEND
  ```

- REPLACE キーワードを使用すると、LOAD_STOK 表の内容が削除されてから新規のデータがロードされます。SQL*Loader では、DELETE 文を使用して表の内容が 1 つのトランザクションでパージされます。

  ```
  LOAD DATA
  INTO TABLE LOAD_STOK REPLACE
  ```

SQL*Loader スクリプトで REPLACE キーワードを使用するかわりに、データをロードする前にプライマリ・データベースの表に対して SQL*Plus の TRUNCATE TABLE コマンドを発行することをお薦めします。これは、プライマリ・データベースとスタンバイ・データベースの両方から高速かつ効率的な方法で表のコピーをパージするのと同じ効果があります。これは、TRUNCATE TABLE コマンドはオンライン REDO ログ・ファイルに記録され、ロジカル・スタンバイ・データベースで SQL Apply により発行されるためです。

SQL*Loader スクリプトに引き続き REPLACE キーワードを挿入することはできますが、プライマリ・データベースではオブジェクトからの 0（ゼロ）行の DELETE が試行されます。プライマリ・データベースから行が削除されていないため、REDO ログ・ファイルに REDO は記録されません。したがって、ロジカル・スタンバイ・データベースに対して DELETE 文は発行されません。

SQL 文 TRUNCATE TABLE を使用せずに REPLACE キーワードを発行すると、トランザクションをロジカル・スタンバイ・データベースに適用する必要がある場合に、SQL Apply に次のような問題が発生する可能性があります。

- 現在、表に多数の行が含まれている場合は、その行をスタンバイ・データベースから削除する必要があります。SQL Apply では文の元の構文を判別できないため、プライマリ・データベースからパージする行ごとに DELETE 文を発行する必要があります。

  たとえば、プライマリ・データベースの表に最初は 10,000 行含まれていた場合、Oracle SQL*Loader は 1 つの DELETE 文を発行して 10,000 行をパージします。スタンバイ・データベースでは、SQL Apply はすべての行がパージされることを認識しないかわりに、10,000 の DELETE 文を個別に発行し、それぞれの文で 1 行ずつパージする必要があります。

- スタンバイ・データベースの表に SQL Apply で使用可能な索引が含まれていない場合、DELETE 文では情報をパージするために全表スキャンが発行されます。

  前述の例では、SQL Apply は 10,000 の DELETE 文を個別に発行しているため、スタンバイ・データベースに対して 10,000 の全表スキャンが発行される可能性があります。

A.10.3 長時間実行トランザクションのトラブルシューティング

SQL Apply 環境における長時間実行トランザクションの主な原因の 1 つは、全表スキャンです。また、索引の作成時や再作成時のように、SQL 文がスタンバイ・データベースに複製されるために、長時間実行トランザクションが発生することもあります。

長時間実行トランザクションの識別

SQL Apply で 1 つの SQL 文が長時間実行されている場合は、SQL Apply インスタンスのアラート・ログに次のような警告メッセージがレポートされます。

```
Mon Feb 17 14:40:15 2003
WARNING: the following transaction makes no progress
WARNING: in the last 30 seconds for the given message!
WARNING: xid =
0x0016.007.000017b6 cscn = 1550349, message# = 28, slavid = 1
knacrb: no offending session found (not ITL pressure)
```

この警告メッセージの注意事項は次のとおりです。

- この警告は、Interested Transaction List（ITL）が不十分な場合に戻される警告メッセージに似ていますが、最終行が knacrb で始まります。この最終行は、次のことを示しています。
  - 全表スキャンが発生している可能性がある
  - この発行では、Interested Transaction List（ITL）が不十分な状態に対して何も実行されない
- この警告メッセージがレポートされるのは、1 つの文の実行時間が 30 秒を超える場合のみです。

長時間実行文で実行されている SQL 文を判別できない場合がありますが、次の SQL 文を発行すると、SQL Apply が作業中のデータベース・オブジェクトを識別できます。

```
SQL> SELECT SAS.SERVER_ID
  2       , SS.OWNER
  3       , SS.OBJECT_NAME
  4       , SS.STATISTIC_NAME
  5       , SS.VALUE
  6    FROM V$SEGMENT_STATISTICS SS
  7       , V$LOCK L
  8       , V$STREAMS_APPLY_SERVER SAS
  9   WHERE SAS.SERVER_ID = &SLAVE_ID
 10     AND L.SID = SAS.SID
 11     AND L.TYPE = 'TM'
 12     AND SS.OBJ# = L.ID1;
```

また、次の SQL 文を発行すると、実行ごとに多数のディスク読取りを発生させた SQL 文を識別できます。

```
SQL> SELECT SUBSTR(SQL_TEXT,1,40)
  2       , DISK_READS
  3       , EXECUTIONS
  4       , DISK_READS/EXECUTIONS
  5       , HASH_VALUE
  6       , ADDRESS
  7    FROM V$SQLAREA
  8   WHERE DISK_READS/GREATEST(EXECUTIONS,1) > 1
  9     AND ROWNUM < 10
 10   ORDER BY DISK_READS/GREATEST(EXECUTIONS,1) DESC
```

すべての表に主キー制約を定義することをお薦めします。これにより自動的に列が NOT NULL として定義されます。主キー制約を定義できない表の場合は、NOT NULL として定義されている適切な列に索引を定義する必要があります。表に適切な列が存在しない場合はその表を見直し、可能な場合は SQL Apply でスキップしてください。

SCOTT スキーマの FTS 表に対して発行された DML 文をすべてスキップする手順は、次のとおりです。

1. SQL Apply を停止します。

   ```
   SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   Database altered
   ```

2. SCOTT.FTS 表について、すべての DML トランザクションのスキップ・プロシージャを構成します。

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.SKIP(stmt => 'DML' , -
        schema_name => 'SCOTT' , -
        object_name => 'FTS');
   PL/SQL procedure successfully completed
   ```

3. SQL Apply を開始します。

   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
   Database altered
   ```

ITL が不十分な状態のトラブルシューティング

Interested Transaction List（ITL）が不十分な状態は、SQL Apply インスタンスのアラート・ログでレポートされます。例 A-3 に、警告メッセージの例を示します。

例 A-3 ITL が不十分な状態に関してレポートされる警告メッセージ

```
Tue Apr 22 15:50:42 2003
WARNING: the following transaction makes no progress
WARNING: in the last 30 seconds for the given message!
WARNING: xid =
0x0006.005.000029fa cscn = 2152982, message# = 2, slavid = 17
```

リアルタイム適用の分析

例 A-3 のメッセージは、SQL Apply プロセス（slavid）#17 が過去 30 秒にわたって進捗がなかったことを示しています。適用プロセスにより発行されている SQL 文を判別するには、次の問合せを発行します。

```
SQL> SELECT SA.SQL_TEXT
  2    FROM V$SQLAREA SA
  3       , V$SESSION S
  4       , V$STREAMS_APPLY_SERVER SAS
  5   WHERE SAS.SERVER_ID = &SLAVEID
  6     AND S.SID = SAS.SID
  7   AND SA.ADDRESS = S.SQL_ADDRESS
SQL_TEXT
------------------------------------------------------------
insert into "APP"."LOAD_TAB_1" p("PK","TEXT")values(:1,:2)
```

ITL が不十分な状態を識別するには、次の例に示すように V$LOCK ビューを問い合せる方法もあります。TX ロックのリクエスト値が 4 のセッションは、ITL が使用可能になるまで待機中です。

```
SQL> SELECT SID,TYPE,ID1,ID2,LMODE,REQUEST
  2    FROM V$LOCK
  3   WHERE TYPE = 'TX'

SID        TY ID1        ID2        LMODE      REQUEST
---------- -- ---------- ---------- ---------- ----------
         8 TX     327688         48          6          0
        10 TX     327688         48          0          4
```

この例では、SID 10 は SID 8 が保持している TX ロックを待機中です。

障害後のレビュー

セグメントの ITL が不十分な状態が長時間続くことはまずありません。また、この状態の継続時間が 30 秒未満の場合、スタンバイ・データベースのアラート・ログではレポートされません。したがって、ITL が不十分な状態になっているオブジェクトを判別するには、次の文を発行します。

```
SQL> SELECT SEGMENT_OWNER, SEGMENT_NAME, SEGMENT_TYPE
  2    FROM V$SEGMENT_STATISTICS
  3   WHERE STATISTIC_NAME = 'ITL WAITS'
  4     AND VALUE > 0
  5   ORDER BY VALUE
```

この文では、インスタンスが前回起動された後に ITL が不十分な状態になったことのあるデータベース・セグメントがすべてレポートされます。

注意： この SQL 文は、Data Guard 環境のロジカル・スタンバイ・データベースに限定されません。すべての Oracle データベースに適用可能です。

ITL が不十分な状態の解消

特定のデータベース・オブジェクトの INITRANS の整数を増やすには、最初に SQL Apply を停止する必要があります。

> **関連項目：** INITRANS の整数を指定する方法の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。この整数は、データベース・オブジェクトに割り当てられた各データ・ブロック内で割り当てられる初期の同時トランザクション・エントリ数です。

次の例に、app スキーマ内の表 load_tab_1 の INITRANS を増やすための手順を示します。

1. SQL Apply を停止します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered.
```

2. データベース・ガードを一時的にバイパスします。

```
SQL> ALTER SESSION DISABLE GUARD;
Session altered.
```

3. スタンバイ・データベースで INITRANS を増やします。次に例を示します。

```
SQL> ALTER TABLE APP.LOAD_TAB_1 INITRANS 30;
Table altered
```

4. データベース・ガードを再び使用可能にします。

```
SQL> ALTER SESSION ENABLE GUARD;
Session altered
```

5. SQL Apply を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
Database altered.
```

また、スイッチオーバー時に新しいスタンバイ・データベースでエラーが発生しないように、プライマリ・データベースでデータベース・オブジェクトを変更することを考慮してください。

A.10.4 フラッシュバック・トランザクションでの ORA-1403 エラーのトラブルシューティング

SQL Apply が「ORA-1403: データが見つかりません。」エラーを戻す場合は、フラッシュバック・トランザクションを使用して欠落しているデータを再構成できる場合があります。これは、スタンバイ・データベース・インスタンスで指定されている UNDO_RETENTION 初期化パラメータに依存します。

通常、ロジカル・スタンバイ・データベース環境で ORA-1403 エラーは表示されません。このエラーが発生するのは、SQL Apply の管理対象である表のデータがスタンバイ・データベースで直接変更された後、同じデータがプライマリ・データベースで変更された場合です。

変更されたデータがプライマリ・データベースで更新されてからロジカル・スタンバイ・データベースで受信されると、SQL Apply はデータのオリジナル・バージョンがスタンバイ・データベースに存在することを確認してから、レコードを更新します。この確認に失敗すると、「ORA-1403: データが見つかりません。」エラーが戻されます。

初期エラー

SQL Apply が検証に失敗すると、ロジカル・スタンバイ・データベースのアラート・ログでエラー・メッセージがレポートされ、DBA_LOGSTDBY_EVENTS ビューにレコードが挿入されます。

アラート・ログ内の情報は切り捨てられますが、エラー全体がデータベース・ビューでレポートされます。次に例を示します。

```
LOGSTDBY stmt: UPDATE "SCOTT"."MASTER"
  SET
    "NAME" = 'john'
  WHERE
    "PK" = 1 and
    "NAME" = 'andrew' and
    ROWID = 'AAAAAAAAEAAAAAPAAA'
LOGSTDBY status: ORA-01403: no data found
LOGSTDBY PID 1006, oracle@staco03 (P004)
LOGSTDBY XID 0x0006.00e.00000417, Thread 1, RBA 0x02dd.00002221.10
```

調査

最初のステップは、エラーの原因となった表の履歴データを分析することです。そのためには、SELECT 文の VERSIONS 句を使用します。たとえば、プライマリ・データベースで次の問合せを発行できます。

```
SELECT VERSIONS_XID
      , VERSIONS_STARTSCN
      , VERSIONS_ENDSCN
      , VERSIONS_OPERATION
      , PK
      , NAME
   FROM SCOTT.MASTER
        VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
  WHERE PK = 1
  ORDER BY NVL(VERSIONS_STARTSCN,0);

VERSIONS_XID     VERSIONS_STARTSCN VERSIONS_ENDSCN V  PK NAME
---------------- ----------------- --------------- - --- -------
03001900EE070000           3492279         3492290 I   1 andrew
02000D00E4070000           3492290                 D   1 andrew
```

データベースが保持するように構成されている UNDO 保存の量（UNDO_RETENTION）と表に対するアクティビティによっては、戻される情報が広範囲にわたり、その量を制限するために構文間でバージョンの変更が必要になる場合があります。

戻された情報から、レコードが最初に SCN 3492279 で挿入され、SCN 3492290 でトランザクション ID 02000D00E4070000 の一部として削除されたことがわかります。

このトランザクション ID を使用してデータベースを問い合せ、トランザクションの有効範囲を確認する必要があります。そのためには FLASHBACK_TRANSACTION_QUERY ビューを問い合せます。

```
SELECT OPERATION
     , UNDO_SQL
  FROM FLASHBACK_TRANSACTION_QUERY
 WHERE XID = HEXTORAW('02000D00E4070000');

OPERATION  UNDO_SQL
---------- ------------------------------------------------
DELETE     insert into "SCOTT"."MASTER"("PK","NAME") values
           ('1','andrew');
BEGIN
```

トランザクションの開始を表す 1 行が必ず戻されることに注意してください。このトランザクションでは、マスター表から削除されたのは 1 行のみです。実行時の UNDO_SQL 列では、元のデータが表にリストアされます。

```
SQL> INSERT INTO "SCOTT"."MASTER"("PK","NAME") VALUES ('1','ANDREW');
SQL> COMMIT;
```

SQL Apply を再開すると、トランザクションがスタンバイ・データベースに適用されます。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

B

Data Guard 構成におけるデータベースのアップグレード

この付録の手順では、構成にフィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースが存在する場合に、Oracle Database 11*g* リリース 1（11.1）にアップグレードする方法を説明します。

この付録は、次の項目で構成されています。

B.1 Oracle Database ソフトウェアをアップグレードする前の注意事項

Oracle Database ソフトウェアのアップグレードを開始する前に、次のことに注意してください。

- 構成の管理に Data Guard Broker を使用している場合、Broker 構成を削除または無効にする方法の詳細は、『Oracle Data Guard Broker』の指示を参照してください。
- この付録で説明する手順は、『Oracle Database アップグレード・ガイド』に記載されている手順とともに使用します。
- この付録で説明する手順では、Database Upgrade Assistant（DBUA）を使用してアップグレードを実行します。アップグレードを手動で実行する手順は、『Oracle Database アップグレード・ガイド』を参照してください。DBUA の使用を言及している場合にも、記載されている手動のアップグレード手順は実行する必要があります。
- NOLOGGING 操作をチェックします。NOLOGGING 操作が実行されていた場合は、スタンバイ・データベースを更新する必要があります。詳細は、13.4 項「NOLOGGING 句を指定した後のリカバリ」を参照してください。
- `OFFLINE IMMEDIATE` が原因でリカバリを必要とする表領域またはデータファイルを書き留めておきます。アップグレード前に表領域またはデータファイルをリカバリし、オンラインまたはオフラインにする必要があります。

B.2 フィジカル・スタンバイ・データベースが存在する場合の Oracle Database のアップグレード

構成にフィジカル・スタンバイ・データベースが存在する場合に、Oracle Database 11*g* リリース 1（11.1）にアップグレードする手順は、次のとおりです。

1. 『Oracle Database アップグレード・ガイド』の「アップグレードの準備」に記載されている手順を確認して実行します。
2. プライマリ・データベースを停止します。
3. フィジカル・スタンバイ・データベースを停止します。
4. フィジカル・スタンバイ・システムで新しいリリースの Oracle ソフトウェアを新しい Oracle ホームにインストールします。詳細は、『Oracle Database アップグレード・ガイド』を参照してください。
5. フィジカル・スタンバイ・データベースをマウントします。

   注意： プライマリ・データベースのアップグレードが完了してから、スタンバイ・データベースをオープンします。

6. フィジカル・スタンバイ・データベースで REDO Apply を開始します。
7. プライマリ・データベース・システムで新しいリリースの Oracle ソフトウェアを新しい Oracle ホームにインストールします。詳細は、『Oracle Database アップグレード・ガイド』を参照してください。
8. プライマリ・データベースをアップグレードします。詳細は、『Oracle Database アップグレード・ガイド』を参照してください。フィジカル・スタンバイ・データベースは、プライマリ・データベースでアップグレード時に生成された REDO を適用するとアップグレードされます。
9. アップグレード後のプライマリ・データベースをオープンします。
10. アップグレードの前に Active Data Guard が使用されている場合、これをアップグレード後に再び有効にする方法は9.2.1 項を参照してください。

B.3 ロジカル・スタンバイ・データベースが存在する場合の Oracle Database のアップグレード

注意： この付録では、ロジカル・スタンバイ・データベースが存在する場合に Oracle Database ソフトウェアをアップグレードする従来の方法を説明します。第 12 章「SQL Apply を使用した Oracle Database のアップグレード」にある 2 つ目の方法では、ロジカル・スタンバイ・データベースが存在する場合にローリング・アップグレードを行って停止時間を最短に抑える方法を説明します。完全なアップグレードを実行するには、どちらか一方の方法の手順のみを使用してください。アップグレードの実行に両方の方法を使用したり、2 つの方法の手順を組み合せようとしないでください。

この項で説明する手順では、プライマリ・データベースが MAXIMUM PERFORMANCE データ保護モードで稼働していると仮定しています。

構成にロジカル・スタンバイ・データベースが存在する場合に、Oracle Database 11*g* リリース 1（11.1）にアップグレードする手順は、次のとおりです。

1. 『Oracle Database アップグレード・ガイド』の「アップグレードの準備」に記載されている手順を確認して実行します。
2. 必要に応じて、プライマリ・データベースでデータ保護モードを MAXIMUM PERFORMANCE に設定します。

   ```
   SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE PERFORMANCE;
   ```

3. プライマリ・データベースで、すべてのユーザー・アクティビティを停止し、ロジカル・スタンバイ・データベースに関連付けられているリモート・アーカイブ先を遅延させます（この手順では、LOG_ARCHIVE_DEST_2 がロジカル・スタンバイ・データベースに関連付けられていると仮定しています）。

   ```
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=DEFER SCOPE=BOTH;
   SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
   ```

4. スタンバイ・データベースで SQL Apply を停止します。

   ```
   SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
   ```

5. プライマリ・データベースで新しいリリースの Oracle ソフトウェアをインストールします。詳細は、『Oracle Database アップグレード・ガイド』を参照してください。
6. ロジカル・スタンバイ・データベースで新しいリリースの Oracle ソフトウェアをインストールします。詳細は、『Oracle Database アップグレード・ガイド』を参照してください。

   注意： 手順 5 および手順 6 は、アップグレード手順中の停止時間を短縮するために、同時に実行できます（つまり、プライマリ・データベースとスタンバイ・データベースは同時にアップグレードできます）。

   ---

7. アップグレード後のロジカル・スタンバイ・データベースで SQL Apply を開始します。Oracle RAC を使用している場合は、他のスタンバイ・データベース・インスタンスを起動します。

   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
   ```

8. アップグレード後のプライマリ・データベースをオープンし、ユーザーが接続できるようにします。Oracle RAC を使用している場合は、他のプライマリ・データベース・インスタンスを起動します。

   また、次のようにして、アップグレード後のロジカル・スタンバイ・データベースへのアーカイブを有効にします。

   ```
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
   ```

9. 必要に応じて、手順 2 で元のデータ保護モードを変更した場合は、そのモードをリセットします。

C

ロジカル・スタンバイ・データベースでサポートされるデータ型および DDL

ロジカル・スタンバイ・データベースを設定する際、ロジカル・スタンバイ・データベースが、プライマリ・データベースのデータ型と表を保持できることを確認する必要があります。この付録では、様々なデータベース・オブジェクト、記憶域タイプおよび PL/SQL パッケージについて、ロジカル・スタンバイ・データベースでサポートされるものとサポートされないものを示します。この章は、次の項目で構成されています。

C.1 データ型に関する考慮事項

次の各項では、データベース・オブジェクトについて、サポートされるものとサポートされないものを示します。

- ロジカル・スタンバイ・データベースでサポートされるデータ型
- ロジカル・スタンバイ・データベースでサポートされないデータ型

C.1.1 ロジカル・スタンバイ・データベースでサポートされるデータ型

ロジカル・スタンバイ・データベースでは、次のデータ型がサポートされます。

BINARY_DOUBLE
BINARY_FLOAT
BLOB
CHAR
CLOB および NCLOB
DATE
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
LONG
LONG RAW
NCHAR
NUMBER
NVARCHAR2
RAW
TIMESTAMP
TIMESTAMP WITH LOCAL TIMEZONE
TIMESTAMP WITH TIMEZONE
VARCHAR2 および VARCHAR
CLOB として格納される XMLType

注意： 次のデータ型に対する SQL Apply サポートでは、プライマリ・データベースに対する互換性要件があります。

- マルチバイトの CLOB サポート（10.1 以上の互換性で動作するプライマリ・データベースが必要）
- LOB およびオーバーフローなしの IOT サポート（10.1 以上の互換性で動作するプライマリ・データベースが必要）
- LOB およびオーバーフローありの IOT サポート（10.2 以上の互換性で動作するプライマリ・データベースが必要）
- CLOB として格納される XMLType（11.1 以上の互換性で動作するプライマリ・データベースが必要）
- TDE サポート（11.1 以上の互換性で動作するプライマリ・データベースが必要）

C.1.2 ロジカル・スタンバイ・データベースでサポートされないデータ型

ロジカル・スタンバイ・データベースでは、次のデータ型がサポートされません。

BFILE
コレクション（VARRAYS およびネストした表を含む）
マルチメディア・データ型（Spatial、Image および Oracle Text を含む）
ROWID、UROWID
ユーザー定義型
セキュア・ファイルとして格納される LOB
オブジェクト・リレーショナルとして格納される XMLType
バイナリ XML

C.2 透過的データ暗号化（TDE）のサポート

Data Guard SQL Apply を使用すると、透過的データ暗号化（TDB）を有効に設定したプライマリ・データベースにデータ保護を提供できます。ロジカル・スタンバイ・データベースを使用し、高度なセキュリティ要件を伴うアプリケーションにデータ保護を提供する場合は、次のことを考慮してください。

- サーバーが保持する鍵を使用する透過的データ暗号化が指定された表は、プライマリ・データベースとスタンバイ・データベースの両方が 11.1 以上の互換性レベルで稼働している場合のロジカル・スタンバイ・データベースの下でサポートされます。
- ハードウェア・セキュリティ・モジュールの下での透過的データ暗号化は、11*g* リリース 1 のロジカル・スタンバイ・データベースの下ではサポートされません。

ロジカル・スタンバイ・データベースの下では、暗号化された列が含まれる表をレプリケートするときに、次の制限事項を考慮する必要があります。

1. 暗号化された REDO レコードを変換するには、透過的データ暗号化鍵を含むオープン・ウォレットへのアクセス権が SQL Apply に必要です。そのため、鍵を含むウォレットの作成後に、そのウォレットをプライマリ・データベースからスタンバイ・データベースにコピーする必要があります。
2. ウォレットは、マスター鍵が変更されるたびに、プライマリ・データベースからロジカル・スタンバイ・データベースにコピーする必要があります。
3. ロジカル・スタンバイ・データベースでは、プライマリ・データベースから暗号化された表をレプリケートしている間、マスター鍵を REKEY（鍵の変更）しないことをお薦めします。これは、暗号化された REDO レコードの検出時に SQL Apply が停止する原因になります。
4. ロジカル・スタンバイ・データベースでは、レプリケートされた表の暗号化鍵を REKEY できます。このためには、REKEY コマンドを発行する前にガード設定を NONE にして低くする必要があります。
5. レプリケートされた暗号化された表では、プライマリ・データベースで使用された暗号化方式とは異なる方式を列に対して使用できます。たとえば、HR.EMPLOYEES 表の SALARY 列がプライマリ・データベースで AES102 暗号化アルゴリズムを使用して暗号化される場合、ロジカル・スタンバイでは AES256 暗号化アルゴリズムを使用して暗号化することができます。あるいは、ロジカル・スタンバイ・データベースでは、SALARY 列を暗号化しないままにしておくことも可能です。

C.3 表領域の暗号化のサポート

Data Guard SQL Apply を使用すると、表領域の暗号化を有効に設定したプライマリ・データベースにデータ保護を提供できます。その場合、C.2 項「透過的データ暗号化（TDE）のサポート」に示した制限事項 1、2 および 3 が適用されます。

注意： 暗号化された表領域の表に対する変更に、SQL Apply が REDO レコードをマイニングおよび適用した際、暗号化されていない形式のユーザー・データのレコードが長時間維持される場合があります。これを許容できない場合、次のコマンドを発行して、暗号化された表領域への SQL Apply のコンポーネントのマイニングに関係するすべてのメタデータ表を移動します。

```
SQL> DBMS_LOGMNR_D.SET_TABLESPACE(NEW_TABLESPACE =>
'ENCRYPTED_LOGMNR_TS');
```

C.4 行レベル・セキュリティおよびファイングレイン監査のサポート

Oracle Database 11*g* 現在、ロジカル・スタンバイでは、DBMS_RLS および DBMS_FGA PL/SQL パッケージによって提供されるセキュリティ環境を自動的にレプリケートできます。このサポートにより、セキュリティ環境は透過的に維持されるため、サーバーがスタンバイにフェイルオーバーした場合のセキュリティに関する考慮事項が簡単に管理できるようになります。また、プライマリ・データに適用されるアクセス制御ポリシーをスタンバイに自動転送できるため、スタンバイ・データに同じレベルの保護を透過的に提供できます。11*g* を使用してスタンバイ・サーバーを新規作成する場合、このレプリケーションはデフォルトで有効に設定されます。それ以外の場合は、適宜 DBA が有効に設定する必要があります。

これらの PL/SQL パッケージのレプリケーションをサポートするには、プライマリとスタンバイの両方が 11.1 以上の互換性設定で稼働している必要があります。

また、参照される表は、ロジカル・スタンバイよって保持されるオブジェクトである必要があります。（たとえば、ROWID 列を含む表のデータがロジカル・スタンバイによって保持されない場合、その表を参照する DBMS_RLS および DBMS_FGA コールも保持されません。）

C.4.1 行レベルのセキュリティ

行レベルのセキュリティは、仮想プライベート・データベース（VPD）とも呼ばれるセキュリティ機能です。この機能を使用すると、表、ビューまたはシノニムに対して直接、セキュリティの粒度を FINE レベルにまで強化できます。VDP ポリシーで保護された表、ビューまたはシノニムに直接または間接的にユーザーがアクセスすると、サーバーはそのユーザーの SQL 文を動的に変更します。この変更では、セキュリティ・ポリシーを実装するファクションによって戻される WHERE 条件（述語）が作成されます。文は、ファンクションで表される条件あるいはファンクションによって戻される条件を使用して、ユーザーに対して透過的に、動的に変更されます。VPD ポリシーは、SELECT、INSERT、UPDATE、INDEX および DELETE 文に適用されます。VPD は、DBMS_RLS パッケージを使用してセキュリティ・ポリシーを適用すると実装されます。

DBMS_RLS プロシージャをプライマリで実行すると、追加情報が REDO に取得されるため、スタンバイでプロシージャ・コールを論理的に再構築して実行することができます。ロジカル・スタンバイでは、VPD の補助オブジェクト（コンテキスト、データベース・ログオン、トリガー、サポート対象パッケージなど）をレプリケートできます。これらのオブジェクトが保持されるスキーマに存在し、レプリケーションを停止させる DDL スキップが構成されていないことを確認する必要があります。

C.4.2 ファイングレイン監査

ファイングレイン監査は、SELECT 文を監査する方法です。DBMS_FGA パッケージを使用すると、表にアクセスするすべての SELECT 文をアクセスされたデータとともに取得できます。FGA ポリシーは、特定の列に適用できます。あるいは、指定した述語が TRUE を戻す行を返す SELECT 文に限定しても適用できます。

DBMS_FGA プロシージャをプライマリで実行すると、追加情報が REDO に取得されるため、スタンバイでプロシージャ・コールを論理的に再構築して実行することができます。

C.4.3 PL/SQL レプリケーションのスキップおよび有効化

PL/SQL は、DDL 文そのものとして skip および skip_error ルールを指定して構成できます。ただし、パッケージおよびプロシージャに対してワイルドカードは使用できません。たとえば、VPD を全面的にスキップするには、次のようにします。

```
DBMS_LOGSTDBY.Skip (
stmt => 'PL/SQL',
schema_name => 'SYS',
object_name =>'DBMS_RLS',
use_like => FALSE);
```

指定されたスキーマが、パッケージが定義されているスキーマであることに注意してください。パッケージの個々のプロシージャをスキップする場合の構文は、次のようになります。

```
DBMS_LOGSTDBY.Skip (
stmt => 'PL/SQL',
schema_name => 'SYS',
object_name =>'DBMS_RLS.Add_Policy',
use_like => FALSE);
```

特定のスキーマまたは表について VPD をスキップするには、スキップ・プロシージャを使用する必要があります。スキップ・プロシージャに、実行予定の完全修飾された PL/SQL 文を渡します。次に例を示します。

```
DBMS_RLS.Drop_Policy(
object_schema => 'SCOTT,
object_name  => 'EMP',
policy_name => 'MYPOLICY');
```

すると、プロシージャは、文を解析してスキップするか、適用するか、適用を中止して DBA に補正アクションを実行させるかを決定します。

DDL とは異なり、PL/SQL に対するスキップ・プロシージャは代用文を戻しません。

C.5 Oracle Label Security

ロジカル・スタンバイ・データベースでは、Oracle Label Security をサポートしません。Oracle Label Security をプライマリ・データベースにインストールすると、開始時に内部エラーが発生して SQL Apply はロジカル・スタンバイ・データベースで失敗します。

C.6 サポートされる表記憶域タイプ

ロジカル・スタンバイ・データベースでは、次の表記憶域タイプがサポートされます。

- クラスタ表（索引クラスタおよびヒープ・クラスタを含む）
- 索引構成表（オーバーフロー・セグメントを含むパーティション化および非パーティション化）
- ヒープ構成表（パーティション化および非パーティション化）

C.7 サポートされない表記憶域タイプ

ロジカル・スタンバイ・データベースでは、次の表記憶域タイプがサポートされません。

- セグメント圧縮が有効に設定されて格納される表
- セキュア・ファイルとして格納される LOB 列を含む表
- 仮想列を含む表

C.8 PL/SQL パッケージに関する考慮事項

この項では、PL/SQL パッケージに関する次の考慮事項について説明します。

- サポートされる PL/SQL パッケージ
- サポートされない PL/SQL パッケージ
- ロジカル・スタンバイでの XML および XDB PL/SQL パッケージの処理

> **関連項目：** オラクル社が提供する PL/SQL パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

C.8.1 サポートされる PL/SQL パッケージ

システムのメタデータやユーザー・データを変更しない、オラクル社が提供する PL/SQL パッケージは、アーカイブ REDO ログ・ファイルにフットプリントを残さないため、プライマリ・データベースで安全に使用できます。この種のパッケージの例には、DBMS_OUTPUT、DBMS_RANDOM、DBMS_PIPE、DBMS_DESCRIBE、DBMS_OBFUSCATION_TOOLKIT、DBMS_TRACE、DBMS_METADATA、DBMS_CRYPTO があります。

システムのメタデータは変更せず、ユーザー・データを変更することがある、オラクル社が提供する PL/SQL パッケージは、変更されるデータが C.1.1 項に示すサポートされるデータ型に属するかぎり、SQL Apply でサポートされます。この種のパッケージの例には、DBMS_LOB、DBMS_SQL、DBMS_TRANSACTION があります。

Data Guard のロジカル・スタンバイでは、DBMS_RLS と DBMS_FGA の 2 つのパッケージを使用して実行されるアクションのレプリケーションがサポートされます。

C.8.2 サポートされない PL/SQL パッケージ

システムのメタデータを変更する、オラクル社が提供する PL/SQL パッケージは、通常、SQL Apply ではサポートされないため、その影響はロジカル・スタンバイ・データベースでは参照できません。この種のパッケージの例には、DBMS_JAVA、DBMS_REGISTRY、DBMS_ALERT、DBMS_SPACE_ADMIN、DBMS_REFRESH、DBMS_REDEFINITION、DBMS_AQ があります。

DBMS_JOB に固有のサポートが用意されています。ジョブの実行はロジカル・スタンバイ・データベース上で一時停止され、スタンバイ・データベース上ではジョブを直接スケジュールできません。ただし、プライマリ・データベースで発行されたジョブは、スタンバイ・データベースにレプリケートされます。スイッチオーバーまたはフェイルオーバーが発生すると、元のプライマリ・データベースでスケジュールされていたジョブの実行は、新規のプライマリ・データベースで自動的に開始されます。

DBMS_SCHEDULER に固有のサポートが用意されているため、スタンバイ・データベースでジョブを実行できます。11*g* では、database_role というスケジューラ・ジョブの新しい属性が作成されています。この属性の内容は、V$DATABASE の database_role 属性と同じです。スケジューラ・ジョブが作成されると、デフォルトでローカル・ロールに設定されます（すなわち、スタンバイで作成されたジョブはデフォルトで LOGICAL STANDBY の database_role に設定されます）。ジョブ・スケジューラは、現行ロールに固有のジョブのみを実行します。スイッチオーバーまたはフェイルオーバーでは、スケジューラは、新しいロールに固有のジョブを実行するように自動的に切り替えます。

スケジューラ・ジョブはスタンバイにレプリケートされません。しかし、既存のジョブは DBMS_SCHEDULER.Set_Attribute プロシージャを使用して、新しいロールの下でアクティブ化できます。あるいは、両方のロールで実行する必要があるジョブをクローニングし、コピーを他のロールに固有のものにすることが可能です。DBA_SCHEDULER_JOB_ROLES ビューには、どのジョブがどのロールに固有かが表示されます。

スケジューラ・ジョブは、ロジカル・スタンバイ・データベースで実行される場合、データベース・ガードに準拠します。そのため、メンテナンスされていない表を変更する必要があるジョブを実行するには、データベース・ガードを STANDBY に設定する必要があります。(ALTER SESSION DISABLE GUARD 文を PL/SQL ブロック内で使用して有効にすることはできません。)

> **関連項目：** 特定のパッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

C.8.3 ロジカル・スタンバイでの XML および XDB PL/SQL パッケージの処理

Oracle Database 11*g* リリース 1（11.1）では、XML が CLOB 形式で格納される場合は、ロジカル・スタンバイで XML をサポートします。しかし、完全にはサポートされない XML とともに使用される PL/SQL パッケージがいくつかあります。

ロジカル・スタンバイでサポートされる PL/SQL パッケージおよびプロシージャは、メモリー内構造のみを変更します。データベースに格納されたデータは変更しません。これらのパッケージは、REDO を生成しないため、ロジカル・スタンバイにはレプリケートされません。

ロジカル・スタンバイでサポートされなくても、レプリケーション・アクティビティの続行にロジカル・スタンバイ・データベースで対応する起動が必要な XML および XDB に関連する特定の PL/SQL パッケージおよびプロシージャは、プライマリ・データベースでこれらのプロシージャが起動することにより、プロシージャの起動を示す追加の REDO レコードが生成されるように装備されています。SQL Apply でこのような REDO レコードが検出されると、SQL Apply は停止し、プロシージャ名を示すエラー・メッセージが DBA_LOGSTDBY_EVENTS 表に書き込まれます。これにより、DBA は対応するプロシージャを適切な時期にロジカル・スタンバイ・データベースで起動し、プライマリ・データベースで生成された後続の REDO レコードをロジカル・スタンバイ・データベースに正常に適用することができます。これらのサポートされないプロシージャの処理の詳細は、C.8.3.1 項～ C.8.3.6 項を参照してください。

次のパッケージには、サポートされないプロシージャが含まれます。

- DBMS_XMLSCHEMA
- DBMS_XMLINDEX

これらのパッケージに加えて、ロジカル・スタンバイでは XDB スキーマへの変更もサポートしません。XDB スキーマ内のオブジェクトは、システム・メタデータとみなされ、直接の変更はレプリケートされません。

Oracle XML DB リポジトリで管理される表は、階層対応表とも呼ばれますが、ロジカル・スタンバイではサポートされません。これらの表は、XML データの格納に使用され、通常の SQL アクセスに加えて FTP および HTTP プロトコルを使用してアクセスできます。これらの表の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

C.8.3.1 DBMS_XMLSCHEMA スキーマ

DBMS_XMLSCHEMA パッケージ内の次のプロシージャは、ロジカル・スタンバイではサポートされず、レプリケートできません。ロジカル・スタンバイは、これらのプロシージャへのコールを検出すると、これらのコールに対してユーザーが補正アクションを実行できるように停止します。次のサポートされないプロシージャの処理に使用できる代替方法の詳細は、C.8.3.3 項～ C.8.3.6 項を参照してください。

- REGISTERSCHEMA
- REGISTERURI
- DELETESCHEMA
- PURGESCHEMA
- COPYEVOLVE
- INPLACEEVOLVE
- COMPILESCHEMA

XDB スキーマは、Oracle 管理スキーマです。このスキーマに対する変更は、ロジカル・スタンバイによって自動的にスキップされます。次のプロシージャは、XDB スキーマを変更しますが、レプリケートされません。

- GENERATEBEAN

次のプロシージャおよびファンクションは、REDO を生成しないため、ロジカル・スタンバイは停止しません。

- GENERATESCHEMAS
- GENERATESCHEMA

C.8.3.2 DBMS_XMLINDEX パッケージ

DBMS_XMLINDEX パッケージ内の SYNCINDEX プロシージャは、ロジカル・スタンバイではサポートされず、レプリケートできません。ロジカル・スタンバイは、このプロシージャへのコールを検出すると停止します。

次のファンクションおよびプロシージャは、REDO を生成しないため、ロジカル・スタンバイは停止しません。

- NODEREFGETREF
- NODEREFGETVALUE
- NODEREFGETPARENTREF
- NODEREFGETNAME
- NODEREFGETNAMESPACE

C.8.3.3 サポートされない PL/SQL プロシージャの処理

サポートされない PL/SQL プロシージャの処理には、2 つのオプションがあります。1 つ目のオプションは、ロジカル・スタンバイの適用プロセスが停止してなんらかの補正アクションを手動で実行できるようにする方法です。2 つ目のオプションは、事前対応アクションを実行し、さらにロジカル・スタンバイのスキップ・プロシージャを使用してサポートされない PL/SQL をスキップする方法です。次の各項で、これらのオプションについてそれぞれ説明します。

C.8.3.4 サポートされない PL/SQL の手動による補正

ロジカル・スタンバイでサポートされないものが検出されると、適用プロセスは停止し、エラーが DBA_LOGSTBDY_EVENTS 表に記録されます。この表を問い合せると、スタンバイが停止した原因となるアクションと、該当する場合は補正のために必要なアクションを確認できます。

次の例に、問合せ内容のサンプルとその出力を示します。

```
select status, event from dba_logstdby_events
          where commit_scn >= (select applied_scn from dba_logstdby_progress) and
          status_code = 16265
          order by commit_scn desc;

STATUS
--------------------------------------------------------------------------------
EVENT
--------------------------------------------------------------------------------
ORA-16265: Unsupported PL/SQL procedure encountered
begin
 "XDB"."DBMS_XMLSCHEMA"."REGISTERSCHEMA" (
   "SCHEMAURL" => 'xmlplsqlsch2

ORA-16265: Unsupported PL/SQL procedure encountered
begin
 "XDB"."DBMS_XMLSCHEMA"."REGISTERSCHEMA" (
   "SCHEMAURL" => 'xmlplsqlsch2


2 rows selected.
```

同じ情報で 2 行が戻されます。これは、ロジカル・スタンバイが失敗したトランザクションを自動的に再試行するためです。この結果は、xmlplsqlsch2 スキーマについて DBMS_XMLSCHEMA.REGISTERSCHEMA へのコールが検出されたときに、スタンバイが停止したことを示しています。この情報を使用して必要なファイルをプライマリから転送し、スタンバイでこのスキーマを登録します。

スタンバイでこのスキーマが正常に登録されると、ロジカル・スタンバイで適用プロセスを再開できます。これは、SKIP FAILED TRANSACTION オプションを使用して実行する必要があります。次に例を示します。

```
alter database start logical standby apply skip failed transaction'
```

ロジカル・スタンバイは、問題となるトランザクションをスキップし、プライマリからの REDO の適用を続行します。

サポートされない PL/SQL を手動でレプリケートするための一般的な手順は、次のとおりです。

1. サポートされない PL/SQL をプライマリ・データベースで実行します。
2. スタンバイ・データベースでサポートされない PL/SQL が検出され、適用が停止します。
3. DBA_LOGSTBDY_EVENTS 表を調べ、適用が停止した原因を確認します。
4. スタンバイで、サポートされない PL/SQL についてなんらかの補正アクションを実行します。
5. スタンバイで適用を再開します。

C.8.3.5 サポートされない PL/SQL の事前対応的な補正

プライマリ・データベースで実行する予定のアクションが、スタンバイの停止原因となることがわかっている場合があります。その場合、事前にアクションを実行してスタンバイが REDO を適用していない時間を最小限に抑えるか、なくなるようにすることが可能です。

たとえば、新しいアプリケーションがインストールされる予定であることがわかっているとします。インストールの一貫として、大量の XML スキーマを登録する必要があります。スタンバイでこれらのスキーマを登録した後に、プライマリで登録することができます。また、スタンバイで DBMS_XMLSCHEMA.REGISTERSCHEMA プロシージャ用にスキップ・プロシージャをインストールすることもできます。このプロシージャは、XML スキーマが登録されているかどうかチェックし、登録されている場合はロジカル・スタンバイにその PL/SQL コールをスキップするように指示します。

この方法は、他のサポートされていない PL/SQL プロシージャの一部にも使用できます。たとえば、DBMS_XMLSCHEMA.DELETESCHEMA は、同様の方法で処理できます。スタンバイにスキーマがインストールされているかどうかを確認し、インストールされていない場合は、スタンバイには大きな影響がまったくないので、その PL/SQL が安全にスキップされるように、スキップ・プロシージャを作成できます。

C.8.3.6 順序に依存するサポートされない PL/SQL の補正

前述の方法は役に立ちますが、どんな場合にも使用できるわけではありません。安全に使用できるのは、PL/SQL が実行される時期が他のトランザクションに対して重要ではない場合に限られます。この方法を使用してはいけない事例の 1 つは、DBMS_XMLSCHEMA.copyEvolve です。

このプロシージャは、スキーマを発展（変更）させ、列を追加または削除（あるいはその両方）して表を変更できます。また、XML 文書が有効か無効かを変更することもできます。ロジカル・スタンバイでこのプロシージャを実行する時期は重要です。安全が保証される唯一の時期は、ロジカル・スタンバイで、このプロシージャがプライマリ・データベースで実行されたことを認識し、適用が停止しているときです。

スキーマを発展させる前に、プライマリでそのスキーマを使用するトラフィックをすべて QUIESCE（静止）させることも重要です。そうしないと、2 つのトランザクション間の依存性がロジカル・スタンバイには識別できないため、プライマリでは evolveSchema に間に合うようにクローズされるトランザクションが、ロジカル・スタンバイでは異なる順序で実行される可能性があります。そのため、順序に依存する PL/SQL が含まれる場合は、次の手順に従う必要があります。

1. プライマリで依存表への変更を QUIESCE させます。
2. プライマリで CopyEvolve を実行します。
3. スタンバイでの CopyEvolve PL/SQL の停止を待機します。
4. スタンバイで補正 CopyEvolve を適用します。
5. スタンバイで適用を再開します。

例 C-1 に、RegisterSchema コールの処理方法を決定するために使用されるプロシージャのサンプルを示します。

例 C-1 RegisterSchema 用の PL/SQL スキップ・プロシージャ

```
-- Procedures to determine how to handle registerSchema calls

-- This procedure extracts the schema URL, or name, from the statement
-- string that is passed into the skip procedure.

Create or replace procedure sec_mgr.parse_schema_str(
  statement             in varchar2,
  schema_name      out varchar2)
Is
  pos1 number;
  pos2 number;
  workingstr   varchar2(32767);
Begin

-- Find the correct argument
pos1 := instr(statement, '"SCHEMAURL" => ''');
workingstr := substr(statement, pos1 + 16);

-- Find the end of the schema name
pos1 := instr(workingstr, '''');

-- Get just the schema name
workingstr := substr(workingstr, 1, pos1 - 1);

schema_name := workingstr;

End parse_schema_str;
/
show errors


-- This procedure checks if a schema is already registered. If so,
-- it returns the value DBMS_LOGSTDBY.SKIP_ACTION_SKIP to indicate that
-- the PL/SQL should be skipped. Otherwise, the value
-- DBMS_LOGSTDBY.SKIP_ACTION_SKIP is returned and Logical Standby apply
-- will halt to allow the DBA to deal with the registerSchema call.

Create or replace procedure sec_mgr.skip_registerschema(
  statement          in varchar2,
  package_owner            in varchar2,
  package_name             in varchar2,
  procedure_name          in varchar2,
  current_user              in varchar2,
  xidusn             in number,
  xidslt               in number,
  xidsqn             in number,
  exit_status           in number,
  skip_action      out number)
Is
  schema_exists number;
  schemastr varchar2(2000);
Begin

  skip_action := DBMS_LOGSTDBY.SKIP_ACTION_SKIP;

  -- get the schame name from statement
  parse_schema_str(statement, schemastr);
```

```
  -- see if the schema is already registered
  select count(*) into schema_exists from sys.all_xml_schemas s
                                  where s.schema_url = schemastr and
                                        s.owner = current_user;

  IF schema_exists = 0 THEN
      -- if the schema is not  registered, then we must stop apply
      skip_action := DBMS_LOGSTDBY.SKIP_ACTION_APPLY;
  ELSE
      -- if the schema is already registered, then we can skip this statement
      skip_action := DBMS_LOGSTDBY.SKIP_ACTION_SKIP;
  END IF;

End skip_registerschema;
/
show errors

-- Register the skip procedure to deal with the unsupported registerSchema
-- PL/SQL.
Begin
   sys.dbms_logstdby.skip(stmt => 'PL/SQL',
        schema_name => 'XDB',
               object_name   => 'DBMS_XMLSCHEMA.REGISTERSCHEMA',
                        proc_name     => 'SEC_MGR.SKIP_REGISTERSCHEMA',
        use_like         => FALSE );
                End;
    /
show errors
```

C.9 サポートされない表

ロジカル・スタンバイ・データベースを作成する前に、プライマリ・データベースでサポートされないデータベース・オブジェクトを識別することが重要です。これは、プライマリ・データベースでサポートされないデータ型および表に対して行われた変更が、ロジカル・スタンバイ・データベースでは SQL Apply によって自動的にスキップされるためです。さらに、エラー・メッセージは戻されません。

ロジカル・スタンバイのサポートの観点から、データベースには次の 3 種類のオブジェクトがあります。

- SQL Apply によって明示的にメンテナンスされるオブジェクト
- SQL Apply によって暗黙的にメンテナンスされるオブジェクト
- SQL Apply によってメンテナンスされないオブジェクト

Oracle データベースに同梱される一部のスキーマ（SYSTEM など）には、SQL Apply によって暗黙的にメンテナンスされるオブジェクトが含まれます。しかし、ユーザー定義の表を SYSTEM に入れると、サポートされるデータ型の列が含まれている場合でもその表はメンテナンスされません。SQL Apply によってメンテナンスされないオブジェクトを検出するには、次の 2 つの問合せを実行する必要があります。1 つ目の問合せは次のとおりです。

```
SQL> SELECT OWNER FROM DBA_LOGSTDBY_SKIP WHERE STATEMENT_OPT = 'INTERNAL SCHEMA';
```

この問合せは、内部スキーマとみなされるスキーマをすべて戻します。これらのスキーマにあるユーザー表は、ロジカル・スタンバイ・データベースでレプリケートされず、DBA_LOGSTDBY_UNSUPPORTED ビューにも表示されません。スキーマに実装された機能がロジカル・スタンバイの下でサポートされる場合、オラクル社で作成されたこれらのスキーマ内の表は、ロジカル・スタンバイで保持されます。

実行する必要がある 2 つ目の問合せは次のとおりです。この問合せは、内部スキーマに属さず、サポートされないデータ型が含まれるために SQL Apply によってメンテナンスされない表を戻します。

```
SQL> SELECT DISTINCT OWNER,TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED
  2> ORDER BY OWNER,TABLE_NAME;

OWNER        TABLE_NAME
-----------  -------------------------
HR           COUNTRIES
OE           ORDERS
OE           CUSTOMERS
OE           WAREHOUSES
```

前述の問合せでリストされたいずれかの表の列名とデータ型を表示するには、次のような SELECT 文を使用します。

```
SQL> SELECT COLUMN_NAME,DATA_TYPE FROM DBA_LOGSTDBY_UNSUPPORTED
  2> WHERE OWNER='OE' AND TABLE_NAME = 'CUSTOMERS';

COLUMN_NAME                    DATA_TYPE
-----------------------------  -------------------
CUST_ADDRESS                   CUST_ADDRESS_TYP
PHONE_NUMBERS                  PHONE_LIST_TYP
CUST_GEO_LOCATION              SDO_GEOMETRY
```

プライマリ・データベースにサポートされない表が含まれている場合、REDO データをロジカル・スタンバイ・データベースに適用すると、SQL Apply サービスによって、その表が自動的に除外されます。

注意： プライマリ・データベースの重要な表がロジカル・スタンバイ・データベースでサポートされない場合は、フィジカル・スタンバイ・データベースの使用を考慮することもできます。フィジカル・スタンバイ・データベースには、このようなデータ型の制約はありません。

C.10 ロジカル・スタンバイ・データベースでスキップされる SQL 文

デフォルトでは、次の SQL 文が SQL Apply により自動的にスキップされます。

```
ALTER DATABASE
ALTER MATERIALIZED VIEW
ALTER MATERIALIZED VIEW LOG
ALTER SESSION
ALTER SYSTEM
CREATE CONTROL FILE
CREATE DATABASE
CREATE DATABASE LINK
CREATE PFILE FROM SPFILE
CREATE MATERIALIZED VIEW
CREATE MATERIALIZED VIEW LOG
CREATE SCHEMA AUTHORIZATION
CREATE SPFILE FROM PFILE
DROP DATABASE LINK
DROP MATERIALIZED VIEW
DROP MATERIALIZED VIEW LOG
EXPLAIN
LOCK TABLE
SET CONSTRAINTS
SET ROLE
SET TRANSACTION
```

プライマリ・データベースで実行されるこの他の SQL 文はすべて、ロジカル・スタンバイ・データベースに適用されます。

C.11 ロジカル・スタンバイ・データベースでサポートされる DDL 文

表 C-1 に、DBMS_LOGSTDBY.SKIP プロシージャの stmt パラメータについて、サポートされる値を示します。表の左側の列には、キーワードの右側にある SQL 文セットの識別に使用される可能性のあるキーワードを示します。キーワードは、通常、データベース・オブジェクトによって定義されます。

> **関連項目：** DBMS_LOGSTDBY パッケージの詳細は『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。また、10.5.3 項「DDL 文のスキップ・ハンドラの設定」も参照してください。

表 C-1 DBMS_LOGSTDBY.SKIP プロシージャの stmt パラメータの値

キーワード	関連する SQL 文
このグループの SQL 文に対するキーワードはない。	GRANT OBJECT REVOKE OBJECT SYSTEM GRANT SYSTEM REVOKE
CLUSTER	AUDIT CLUSTER CREATE CLUSTER DROP CLUSTER TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT DROP CONTEXT
DATABASE LINK	CREATE DATABASE LINK CREATE PUBLIC DATABASE LINK DROP DATABASE LINK DROP PUBLIC DATABASE LINK
DIMENSION	ALTER DIMENSION CREATE DIMENSION DROP DIMENSION
DIRECTORY	CREATE DIRECTORY DROP DIRECTORY
DML	表内の DML 文を含む（例：INSERT、UPDATE、DELETE）
INDEX	ALTER INDEX CREATE INDEX DROP INDEX
NON_SCHEMA_DDL	特定のスキーマに関連付けられていないすべての DDL **注意：** SCHEMA_NAME および OBJECT_NAME は NULL である必要があります。

表 C-1 DBMS_LOGSTDBY.SKIP プロシージャの stmt パラメータの値（続き）

キーワード	関連する SQL 文
PROCEDURE[1]	ALTER FUNCTION ALTER PACKAGE ALTER PACKAGE BODY ALTER PROCEDURE CREATE FUNCTION CREATE LIBRARY CREATE PACKAGE CREATE PACKAGE BODY CREATE PROCEDURE DROP FUNCTION DROP LIBRARY DROP PACKAGE DROP PACKAGE BODY DROP PROCEDURE
PROFILE	ALTER PROFILE CREATE PROFILE DROP PROFILE
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM
ROLE	ALTER ROLE CREATE ROLE DROP ROLE SET ROLE
ROLLBACK STATEMENT	ALTER ROLLBACK SEGMENT CREATE ROLLBACK SEGMENT DROP ROLLBACK SEGMENT
SCHEMA_DDL	スキーマ・オブジェクト（例：表、索引、列）の作成、変更または削除を行うすべての DDL 文 **注意：** SCHEMA_NAME および OBJECT_NAME は NULL 以外である必要があります。
SEQUENCE	ALTER SEQUENCE CREATE SEQUENCE DROP SEQUENCE
SYNONYM	CREATE PUBLIC SYNONYM CREATE SYNONYM DROP PUBLIC SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT *SQL_statements* NOAUDIT *SQL_statements*
TABLE	CREATE TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE DROP TABLESPACE ALTER TABLESPACE

表 C-1 DBMS_LOGSTDBY.SKIP プロシージャの stmt パラメータの値（続き）

キーワード	関連する SQL 文
TRIGGER	ALTER TRIGGER CREATE TRIGGER DISABLE ALL TRIGGERS DISABLE TRIGGER DROP TRIGGER ENABLE ALL TRIGGERS ENABLE TRIGGER
TYPE	ALTER TYPE ALTER TYPE BODY CREATE TYPE CREATE TYPE BODY DROP TYPE DROP TYPE BODY
USER	ALTER USER CREATE USER DROP USER
VIEW	CREATE VIEW DROP VIEW

[1] Java スキーマ・オブジェクト（ソース、クラス、リソース）は、SQL 文をスキップ（無視）するためのプロシージャと同等とみなされます。

関連項目： SKIP および UNSKIP オプションの使用例を示した次の各項

- 10.5.2 項「DBMS_LOGSTDBY.SKIP による特定のスキーマ・オブジェクトに対する変更の防止」
- 10.5.3 項「DDL 文のスキップ・ハンドラの設定」
- 10.5.4 項「ロジカル・スタンバイ・データベースの変更」
- 10.5.5 項「ロジカル・スタンバイ・データベースでの表の追加または再作成」

C.11.1 DBLINK を使用する DDL 文

SQL Apply は、データベース・リンクを参照する次のような DDL 文を正しく適用できない場合があります。

```
CREATE TABLE tablename AS SELECT * FROM bar@dblink
```

これは、ロジカル・スタンバイ・データベースで *dblink* がプライマリ・データベースと同じデータベースを指さないことがあるためです。このような DDL 文の実行中に SQL Apply に障害が発生した場合は、表の作成に DBMS_LOGSTDBY.INSTANTIATE_TABLE プロシージャを使用して、SQL APPLY の操作を再開する必要があります。

C.11.2 ロジカル・スタンバイ・プロセス上の AUD$ および FGA_LOG$ のレプリケーション

ロジカル・スタンバイでは、監査およびファイングレイン監査がサポートされています。プライマリ・データベースでの AUD$ および FGA_AUD$ 表に対する変更は、ロジカル・スタンバイでレプリケートされます。

AUD$ 表および FGA_AUD$ 表の両方に、DBID 列があります。DBID 値がプライマリ・データベースの値の場合、その行は、プライマリでのアクティビティに基づいてロジカル・スタンバイにレプリケートされています。DBID 値がロジカル・スタンバイ・データベースの値の場合、その行は、ロジカル・スタンバイのローカル・アクティビティの結果として挿入されています。

ロジカル・スタンバイ・データベースで、プライマリ・ロールをロールの推移（スイッチオーバーまたはフェイルオーバー）の結果とみなした後、新しいプライマリ（元のロジカル・スタンバイ）および新しいロジカル・スタンバイ（元のプライマリ）の AUD$ 表および FGA_AUD$ 表は、必ず同期されるとはかぎりません。このため、新しいプライマリ・データベースの AUD$ 表または FGA_AUD$ 表のすべての行が、新しいロジカル・スタンバイ・データベースに存在するとはかぎりません。ただし、データベースがプライマリ・ロールの際に挿入された AUD$ および FGA_LOG$ のすべての行はレプリケートされ、ロジカル・スタンバイ・データベースに存在します。

D

Data Guard および Oralce Real Application Clusters

Oracle Data Guard 構成は、単一インスタンスおよび複数インスタンスの Oralce Real Application Clusters（RAC）データベースの組合せで構成されます。この章では、Oracle Data Guard を Oracle RAC データベースとともに使用する場合に適用される構成要件と考慮事項の概要を説明します。次の項目で構成されています。

D.1 Oracle RAC 環境でのスタンバイ・データベースの構成

スタンバイ・データベースを構成すると、Oracle RAC を使用しているプライマリ・データベースを保護できます。次の表では、プライマリ・データベースとスタンバイ・データベースのインスタンスの可能な組合せについて説明します。

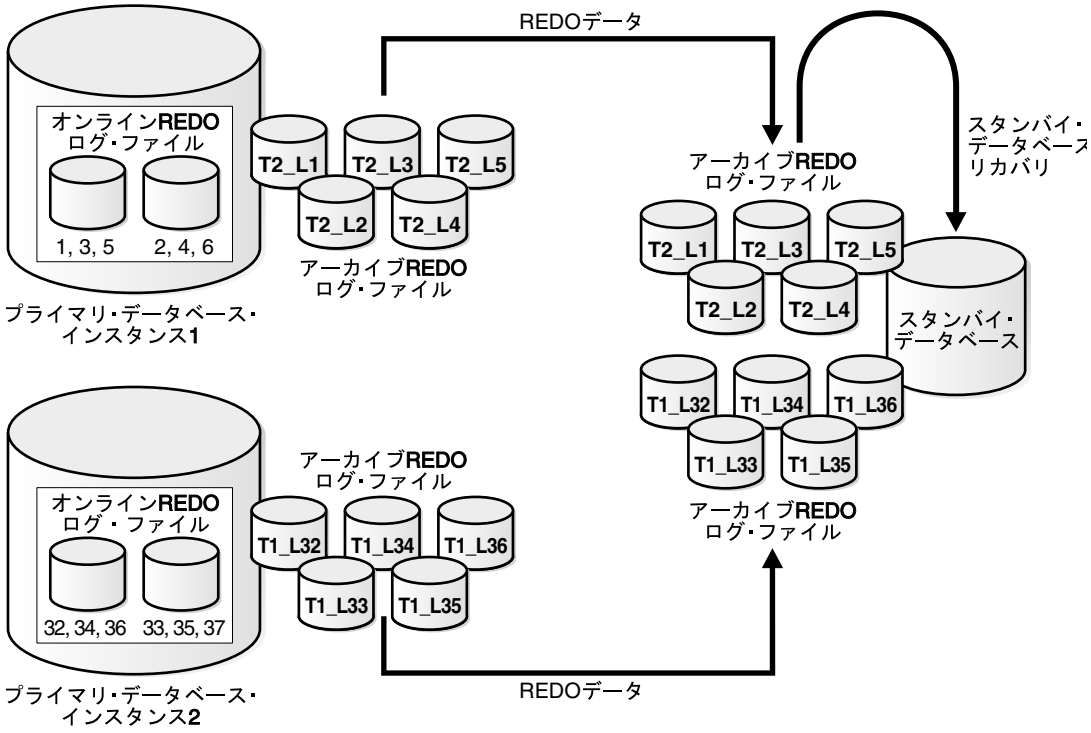
インスタンスの組合せ	単一インスタンス・スタンバイ・データベース	複数インスタンス・スタンバイ・データベース
単一インスタンス・プライマリ・データベース	可	可
複数インスタンス・プライマリ・データベース	可	可

それぞれの組合せでは、プライマリ・データベースの各インスタンスが、REDO データをスタンバイ・データベースのインスタンスへ転送します。

D.1.1 複数インスタンス・プライマリ・データベースと単一インスタンス・スタンバイ・データベースの設定

図 D-1 では、プライマリ・データベース・インスタンスが 2 つある Oracle RAC データベース（複数インスタンス・プライマリ・データベース）が、単一インスタンス・スタンバイ・データベースへ REDO データを転送しています。

図 D-1 複数インスタンス・プライマリ・データベースからの REDO データの転送



この場合は、プライマリ・データベースのインスタンス 1 がローカル・アーカイブ REDO ログ・ファイル 1、2、3、4、5 に REDO データをアーカイブし、スタンバイ・データベース宛先に REDO データを転送するのに対し、インスタンス 2 がローカル・アーカイブ REDO ログ・ファイル 32、33、34、35、36 に REDO データをアーカイブし、同じスタンバイ・データベース宛先に REDO データを転送します。スタンバイ・データベースでは、アーカイブ REDO ログ・ファイルを適用する正しい順序が自動的に判断されます。

Oracle RAC 環境でプライマリ・データベースを設定する手順

各プライマリ・インスタンスを構成するには、第 3 章（フィジカル・スタンバイ・データベース作成の場合）、または第 4 章（ロジカル・スタンバイ・データベース作成の場合）の説明に従ってください。

単一インスタンス・スタンバイ・データベースを設定する手順

LOG_ARCHIVE_DEST_n パラメータおよび LOG_ARCHIVE_FORMAT パラメータを定義して、アーカイブ REDO ログ・ファイルおよびスタンバイ REDO ログ・ファイルの場所を指定するには、第 3 章（フィジカル・スタンバイ・データベース作成の場合）、または第 4 章（ロジカル・スタンバイ・データベース作成の場合）の説明に従ってください。

D.1.2 Oracle RAC プライマリおよびスタンバイ・データベースの設定

この項では、Oracle RAC スタンバイ・データベースに REDO データを送信するように Oracle RAC プライマリ・データベースを構成する方法について説明します。

D.1.2.1 REDO データを受信するための Oracle RAC スタンバイ・データベースの構成

次の手順を実行して、プライマリ・データベースから REDO データを受信するように Oracle RAC スタンバイ・データベースを構成します。

1. スタンバイ・データベースで、スタンバイ REDO ログを作成します。スタンバイ REDO ログの REDO ログ・ファイルは、すべてのスタンバイ・データベース・インスタンスがアクセスできる場所（クラスタ・ファイル・システムや ASM インスタンスなど）に存在する必要があります。スタンバイ REDO ログの作成の詳細は、6.2.3.1 項を参照してください。
2. 各スタンバイ・データベース・インスタンスでスタンバイ REDO ログ・アーカイブを構成します。スタンバイ REDO ログは、すべてのスタンバイ・データベース・インスタンスがアクセスできる場所にアーカイブする必要があります。また、すべてのスタンバイ・データベース・インスタンスは、スタンバイ REDO ログを同じ場所にアーカイブするように構成する必要があります。スタンバイ REDO ログ・アーカイブの構成の詳細は、6.2.3.2 項を参照してください。

D.1.2.2 REDO データを送信するための Oracle RAC プライマリ・データベースの構成

REDO データを RAC スタンバイ・データベースに送信するように、RAC プライマリ・データベースの各インスタンスを構成します。REDO データを別のデータベースに送信するように Oracle データベース・インスタンスを構成する方法は、6.2.2 項を参照してください。

REDO データを Oracle RAC スタンバイ・データベースに送信するように Oracle RAC プライマリ・データベースを構成する際には、次のベスト・プラクティスをお薦めします。

1. 各プライマリ・データベース・インスタンスで同じ LOG_ARCHIVE_DEST_n パラメータを使用して、REDO データを特定のスタンバイ・データベースに送信する。
2. 特定のスタンバイ・データベースに対応する各 LOG_ARCHIVE_DEST_n パラメータの SERVICE 属性を同じネット・サービス名に設定する。
3. ネット・サービス名は、アドレス・リストを含む Oracle Net 接続記述子に解決する必要があり、そのアドレス・リストには、各スタンバイ・データベース・インスタンスの接続データが含まれる必要がある。

D.2 Oracle RAC 環境での構成に関する考慮事項

この項では、Oracle RAC 環境に固有の Data Guard 構成情報を示します。この章は、次の項目で構成されています。

- アーカイブ REDO ログ・ファイル名の形式
- データ保護モード
- ロールの推移

D.2.1 アーカイブ REDO ログ・ファイル名の形式

アーカイブ REDO ログ・ファイル名は、log_%*parameter* の形式になります。%*parameter* には、表 D-1 のパラメータを 1 つ以上含めることができます。

表 D-1 LOG_ARCHIVE_FORMAT 初期化パラメータのディレクティブ

ディレクティブ	説明
%a	データベース・アクティブ ID
%A	0（ゼロ）を埋め込んだデータベース・アクティブ ID
%d	データベース ID
%D	0（ゼロ）を埋め込んだデータベース ID
%t	インスタンス・スレッド番号
%T	0（ゼロ）を埋め込んだインスタンス・スレッド番号
%s	ログ・ファイル順序番号
%S	0（ゼロ）を埋め込んだログ・ファイル順序番号
%r	リセットログ ID
%R	0（ゼロ）を埋め込んだリセットログ ID

次に例を示します。

```
LOG_ARCHIVE_FORMAT = log%d_%t_%s_%r.arc
```

Oracle RAC が、LOG_ARCHIVE_FORMAT パラメータでアーカイブ REDO ログ・ファイルを一意に識別するためには、スレッド・パラメータ %t または %T が必須です。

D.2.2 データ保護モード

最大保護モードまたは最大可用性モードで稼働している Oracle RAC 構成では、インスタンスがスタンバイ宛先との接続を失うと、他のすべてのインスタンスがその宛先へのデータ送信を停止します（これにより、その宛先に転送されたデータの整合性が維持されます）。

障害のあったスタンバイ宛先が復旧すると、Data Guard は、ギャップなしになるまで、そのサイトを再同期化モードで実行します。これによって、そのスタンバイ宛先は Data Guard 構成に再び含まれます。

次に、Oracle RAC 環境における保護モードの動作を説明します。

- 最大保護の構成

  接続を失った宛先が最後の SYNC 宛先の場合、インスタンスは接続を失い、停止します。Oracle RAC 構成内でスタンバイ宛先への接続を維持しているインスタンスは、接続を失ったインスタンスをリカバリし、スタンバイ宛先への送信を継続します。Oracle RAC 構成内のすべてのインスタンスが最後のスタンバイ宛先への接続を失った場合にのみ、プライマリ・データベースが停止します。

D.2.3 ロールの推移

この項は、次の項目で構成されています。

- スイッチオーバー
- フェイルオーバー

D.2.3.1 スイッチオーバー

Oracle RAC データベースの場合は、ターゲット・データベースがフィジカル・スタンバイであるスイッチオーバー時にアクティブにできるのは、1 つのプライマリ・インスタンスと 1 つのスタンバイ・インスタンスのみです。したがって、フィジカル・スタンバイ・データベースへのスイッチオーバーの前に、1 つのプライマリ・インスタンスと 1 つのスタンバイ・インスタンス以外はすべて停止します。スイッチオーバーの完了後、スイッチオーバーの実行時に停止したプライマリ・インスタンスとスタンバイ・インスタンスを再起動します。この制限は、ロジカル・スタンバイ・データベースの場合には存在しません。

注意： SQL ALTER DATABASE 文を使用してスイッチオーバーを実行すると、REDO ログ・ファイルが存在していない場合は自動的に作成されます。これによって、COMMIT 操作の完了に必要な時間が大幅に長くなる場合があるため、フィジカル・スタンバイ・データベースを作成するときは、REDO ログ・ファイルを手動で追加することをお薦めします。

D.2.3.2 フェイルオーバー

Oracle RAC スタンバイ・データベースへのフェイルオーバーを実行するには、まず、1 つのスタンバイ・インスタンス以外はすべて停止します。フェイルオーバーの完了後、停止したインスタンスを再起動します。

D.3 トラブルシューティング

この項は、Oracle RAC で発生する問題のトラブルシューティングのヘルプとして利用できます。

D.3.1 Oralce RAC 構成でスイッチオーバーできない

データベースで Oracle RAC を使用すると、アクティブ・インスタンスによってスイッチオーバーの実行が妨げられます。他のインスタンスがアクティブの場合は、スイッチオーバーの試行が次のエラー・メッセージを伴って失敗します。

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY;
ALTER DATABASE COMMIT TO SWITCHOVER TO STANDBY *
ORA-01105: mount is incompatible with mounts by other instances
```

処置：次のように GV$INSTANCE ビューを問い合せ、問題の原因となっているインスタンスを判断します。

```
SQL> SELECT INSTANCE_NAME, HOST_NAME FROM GV$INSTANCE
  2> WHERE INST_ID <> (SELECT INSTANCE_NUMBER FROM V$INSTANCE);
INSTANCE_NAME HOST_NAME
------------- ---------
INST2         standby2
```

この例では、識別されたインスタンスを手動で停止しないと、スイッチオーバーを継続できません。識別されたインスタンスには使用中のインスタンスから接続でき、SHUTDOWN 文をリモートで発行できます。次に例を示します。

```
SQL> CONNECT SYS@standby2 AS SYSDBA
Enter Password:
SQL> SHUTDOWN;
SQL> EXIT
```

E

カスケードされた宛先

プライマリ・システムの負荷を軽減するため、あるいは Wide Area Network（WAN）を介してスタンバイ・データベースとプライマリ・データベースが分散している場合の帯域幅の要件を緩和するために、カスケードされた宛先を実装できます。これにより、スタンバイ・データベースは、REDO データをプライマリ・データベースから直接受信するのではなく、別のスタンバイ・データベースから受信します。

カスケードされた宛先を使用する Data Guard 構成では、フィジカル・スタンバイ・データベースはプライマリ・データベースから受信した REDO データを別のスタンバイ・データベースに転送できます。REDO データを別のスタンバイ・データベースに転送するように構成できるのは、フィジカル・スタンバイ・データベースのみです。ロジカル・スタンバイ・データベースは、REDO を別のスタンバイ・データベースに転送できません。

> **注意：** プライマリ・データベースが Oracle Real Application Clusters（RAC）環境または Data Guard Broker 環境の一部である場合は、REDO を転送するようにフィジカル・スタンバイを設定できません。

カスケードされた宛先を使用する次の Data Guard 構成がサポートされます。

- プライマリ・データベース→カスケードされた宛先が定義されているフィジカル・スタンバイ・データベース→フィジカル・スタンバイ・データベース
- プライマリ・データベース→カスケードされた宛先が定義されているフィジカル・スタンバイ・データベース→ロジカル・スタンバイ・データベース

フィジカル・スタンバイ・データベースでは、最大 9 個のリモート宛先をサポートできます。フィジカル・スタンバイ・データベースでカスケードされた宛先を定義すると、フィジカル・スタンバイは、スタンバイ REDO ログがいっぱいになってアーカイブされた後に、プライマリから受信した REDO を 2 つ目のスタンバイ・データベースに転送します。そのため、カスケードされた宛先の結果として転送された REDO を受信する 2 つ目のスタンバイ・データベースは、必然的にプライマリ・データベースから遅れます。カスケードされた宛先は、プライマリ・システムからまったく遅れていないデータにアクセスする必要がないオフロードのレポート生成やアプリケーションにのみ使用することをお薦めします。これは、カスケードされた宛先の性質そのものが、エンド・ポイントであるスタンバイ・データベースがプライマリ・データベースから遅れている 1 つ以上のログ・ファイルであるということを意味するためです。また、プライマリ・ロールがロールの推移の対象となるスタンバイ・データベースは、REDO データをプライマリ・データベースから直接受信することをお薦めします。

この付録では、次の情報を示します。

E.1 カスケードされた宛先の構成

フィジカル・スタンバイ・データベースを有効にして着信 REDO データをカスケードされた宛先に転送するには、次の手順を実行します。

1. フィジカル・スタンバイ・データベースでスタンバイ REDO ログ・ファイルを作成します（未作成の場合）。
2. スタンバイ REDO ログ・ファイルが未定義の場合は、スタンバイ・データベースで動的に定義できます。スタンバイ・データベースは、プライマリ・データベースで次回ログ・スイッチが発生した後に、スタンバイ REDO ログ・ファイルの使用を開始します。
3. プライマリ・データベースで LOG_ARCHIVE_DEST_*n* 初期化パラメータを定義して、REDO をカスケードされた宛先に転送するフィジカル・スタンバイ・データベースを設定します。次の属性を使用する宛先を定義します。
   - ASYNC または SYNC
   - 必要に応じて、元のプライマリ・データベースと REDO を転送する中間スタンバイ・データベース間でロールの推移が発生した後でも REDO 転送が有効であるように、VALID_FOR 属性を設定します。この設定は、データベースが Wide Area Network 上に分散している場合に意味があります。
4. カスケードされた宛先が定義されているフィジカル・スタンバイ・データベース（REDO を転送するスタンバイ・データベース）でアーカイブが有効であることを確認します。
5. カスケードされた宛先ごとに（REDO データを転送するフィジカル・スタンバイで）LOG_ARCHIVE_DEST_*n* パラメータを構成します。

例 E-1 に、Boston というプライマリ・データベースの初期化パラメータを示します。Boston データベースは Chicago というフィジカル・スタンバイ・データベースに REDO を送信し、Chicago データベースは Denver というカスケードされたスタンバイ・データベースに受信した REDO を転送します。この例では、Denver データベースはロジカル・スタンバイ・データベースですが、フィジカル・スタンバイ・データベースは、フィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースのいずれにも REDO を転送できることに注意してください。

注意： カスケードされた宛先がロジカル・スタンバイ・データベースである場合、ロジカル・スタンバイがプライマリ・データベースに直接接続されるかのように作成することに注意してください。詳細は、第 4 章「ロジカル・スタンバイ・データベースの作成」を参照してください。

例 E-1 カスケードされた宛先での初期化パラメータの使用例

Boston データベース（プライマリ・ロール）

```
DB_UNIQUE_NAME=boston
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE

LOG_ARCHIVE_CONFIG='DG_CONFIG=(chicago,boston,denver)'

LOG_ARCHIVE_DEST_1='LOCATION=/arch1/boston/ VALID_FOR=(ALL_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=boston'

LOG_ARCHIVE_DEST_2= 'SERVICE=denver  VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE)
DB_UNIQUE_NAME=denver'

LOG_ARCHIVE_DEST_3=  'SERVICE=chicago  VALID_FOR=  (ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=chicago'
```

Chicago データベース（スタンバイ・ロール）

```
DB_UNIQUE_NAME=chicago
LOG_ARCHIVE_CONFIG= 'DG_CONFIG=(chicago,boston,denver)'
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE

LOG_ARCHIVE_DEST_1= 'LOCATION=/arch1/chicago/  VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
DB_UNIQUE_NAME=chicago'

LOG_ARCHIVE_DEST_2= 'SERVICE=denver  VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE)
DB_UNIQUE_NAME=denver'

LOG_ARCHIVE_DEST_3=  'SERVICE=boston  VALID_FOR=  (ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=boston'
```

Denver データベース（スタンバイ・ロール）

```
DB_UNIQUE_NAME=denver
LOG_ARCHIVE_CONFIG= 'DG_CONFIG=(chicago,boston,denver)'
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE

LOG_ARCHIVE_DEST_1= 'LOCATION=/arch1/denver/  VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES)
DB_UNIQUE_NAME=denver'

LOG_ARCHIVE_DEST_2= 'LOCATION=/arch2/denver/  VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE)
DB_UNIQUE_NAME=denver'
```

Boston プライマリ・データベースおよび Chicago フィジカル・スタンバイ・データベースでは、LOG_ARCHIVE_DEST_2 初期化パラメータを SERVICE=denver VALID_FOR=(STANDBY_LOGFILES, STANDBY_ROLE) として定義しています。そのため、Boston データベースと Chicago データベースでロールが切り替えられても、REDO データは Denver データベースに引き続き転送されます。フィジカル・スタンバイ・データベースの元の設定の一部として、フィジカル・スタンバイ・データベースがプライマリ・ロールに推移した際にローカル・アーカイブに使用する、ローカルの宛先 VALID_FOR=(ALL_LOGFILES, PRIMARY_ROLE) を定義する必要があることに注意してください。

E.2 カスケードされた宛先を使用したロールの推移

障害時リカバリを主目的とするスタンバイ・データベースでは、REDO データをプライマリ・データベースから直接受信することをお薦めします。これは、最高レベルのデータ保護となります。カスケードされた宛先は、第 2 防御ラインとして利用できますが、本質的に REDO をプライマリから直接受信するスタンバイ・データベースより常に大幅に遅れます。

E.3 カスケードされた宛先の使用例

この項では、次の使用例について説明します。各使用例では、カスケードされた宛先の構成オプションとその使用方法を示します。

- フィジカル・スタンバイによるリモート・フィジカル・スタンバイへの REDO の転送
- フィジカル・スタンバイによるローカル・スタンバイへの REDO の転送

E.3.1 フィジカル・スタンバイによるリモート・フィジカル・スタンバイへの REDO の転送

会社のオフィスにプライマリ・データベースがあり、首都圏内の別の施設にスタンバイ・データベースを作成して、プライマリ・サイトで障害が発生した場合にデータ消失がないように保護するとします。ローカル・スタンバイに加えて、地理的に 2000 マイル離れた障害リカバリ・サイトにリモート・スタンバイ・データベースを保持するとします。リモート・スタンバイへのフェイルオーバーが必要な場合、少量のデータ消失は許容できます（大規模な地理的領域に影響を与える可能性があり、プライマリ・サイトとローカル・スタンバイ・データベースの両方に障害が発生する原因となるイベントに対する特別な保護と引き換えに許容できる交換条件です）。また、リモート・スタンバイ・データベースは、ローカル・スタンバイ・データベースへのフェイルオーバー後も引き続きデータ保護を提供し、バックアップを離れた場所に作成して格納できるようにしてオフサイトにテープを発送する必要をなくし、セキュリティを向上させます。

プライマリ・データベースは、REDO を両方のスタンバイ・データベースに直接送信するように構成できますが、WAN を介して REDO を 2 つ目のスタンバイ・データベースに送信するプライマリ・データベースの潜在的なオーバーヘッドをなくす必要があります。この問題は、1 つ目のフィジカル・スタンバイを首都圏内のローカル施設に作成し、SYNC ネットワーク・トランスポートを使用してデータ消失がないように保護することで解決します。カスケードされた宛先は、ASYNC ネットワーク転送を使用して、プライマリから受信した REDO をリモート・スタンバイ・データベースに転送するローカル・フィジカル・スタンバイ・データベースで定義します。これは、ローカル・スタンバイでカスケードされた宛先によるリモート・スタンバイとの通信をすべて管理するためで、プライマリ・データベースには、2 つ目のスタンバイの保持による影響はまったくありません。

E.3.2 フィジカル・スタンバイによるローカル・スタンバイへの REDO の転送

この使用例では、米国の都市にプライマリ・データベースがあり、ヨーロッパの 3 つの異なる工業プラントに、エンドユーザーの問合せやレポート生成に使用するために、このデータベースの 3 つの完全なレプリカをデプロイするとします。目的は、ヨーロッパのユーザーおよびアプリケーションが米国にあるデータにアクセスする必要をなくし、ネットワークの分断によってデータがローカル・アクセスできなくなるを防ぐことです。プライマリで行われる更新の時期とその更新を 3 つのヨーロッパ・サイト全部にレプリケートする時期の間に多少の待機時間があることは許容できますが、データはできるだけ最新の状態にして問合せやレポート生成の実行に使用できるようにする必要があります。アプリケーションには完全に透過的で、必要に応じてヨーロッパのサイトにレプリカを追加デプロイできる解決策が必要です。最後の要件は、米国とヨーロッパの施設間で、帯域幅が制限され、ネットワーク待機時間が非常に長いネットワーク接続を使用して、この作業を実行する必要があるということです。

これらの要件に対処するため、まず、米国にあるプライマリ・データベース用のフィジカル・スタンバイ・データベースをヨーロッパに作成します。次に、各ヨーロッパ・プラントに 1 つずつ、計 3 つのロジカル・スタンバイ・データベースを作成し、各ロジカル・スタンバイをカスケードされた宛先としてフィジカル・スタンバイ・データベースに定義します。REDO のコピーの 1 つが、大西洋を隔てたリンクを介して米国からヨーロッパのフィジカル・スタンバイに送信されます。ヨーロッパのフィジカル・スタンバイは、ヨーロッパの工業プラントにある 3 つのロジカル・スタンバイ・データベースに REDO を転送し、エンドユーザーの問合せやレポート生成のために企業データにローカル・アクセスできるようにします。今後の成長のための余地は組み込まれています。追加のスタンバイ・データベースは、アプリケーションの変更、プライマリ・システムでの追加オーバーヘッド、大西洋を隔てた帯域幅の追加消費を必要とせずにヨーロッパにデプロイできます。

フィジカル・スタンバイ・データベースは、前述の例のように、各工業サイトにあるロジカル・スタンバイ・データベースに REDO データを転送するように構成します。例 E-1 のパラメータとの唯一の相違点は、2 つの `LOG_ARCHIVE_DEST_n` パラメータをフィジカル・スタンバイに追加定義して、REDO が 3 つのロジカル・スタンバイ・データベースすべてに転送されるようにしていることです。

F

Recovery Manager を使用したスタンバイ・データベースの作成

この付録では、Oracle Recovery Manager を使用してスタンバイ・データベースを作成する方法を説明します。この付録は、次の項目で構成されています。

F.1 前提条件

この付録では、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』のデータベースの複製に関する章を読んでいることを前提としています。DUPLICATE コマンドを使用して Recovery Manager でスタンバイ・データベースを作成するため、『Oracle Database バックアップおよびリカバリ・リファレンス』の DUPLICATE コマンドに関する記載を理解しておく必要があります。

この章で説明する Recovery Manager での作成手順を実行する前に、第 3 章「フィジカル・スタンバイ・データベースの作成」および第 4 章「ロジカル・スタンバイ・データベースの作成」のスタンバイ・データベースの作成方法を理解しておいてください。

F.2 Recovery Manager を使用したスタンバイ・データベースの作成の概要

この項では、Recovery Manager を使用してスタンバイ・データベースを作成する目的および基本概念について説明します。

F.2.1 Recovery Manager を使用したスタンバイ・データベースの作成の目的

プライマリ・データベースのバックアップからスタンバイ・データベースを作成するには、手動で行うか、Recovery Manager の DUPLICATE コマンドを使用します。Recovery Manager を使用してスタンバイ・データベースを作成すると、手動による作成と比べて次のメリットがあります。

- プライマリ・データベースで現在使用しているファイルをコピーしてスタンバイ・データベースを作成できます。バックアップは必要ありません。
- プライマリ・データベースのバックアップをスタンバイ・サイトにリストアしてスタンバイ・データベースを作成できます。したがって、スタンバイ・データベースの作成中にプライマリ・データベースが影響を受けることはありません。
- Oracle Managed Files（OMF）などのファイルやディレクトリ構造の名前の変更が自動化されます。
- アーカイブ REDO ログ・ファイルをバックアップからリストアされ、スタンバイ・データベースとプライマリ・データベースが同期化されるようにメディア・リカバリが実行されます。

F.2.2 Recovery Manager を使用したスタンバイ・データベースの作成の基本概念

Recovery Manager を使用してスタンバイ・データベースを作成する手順は、複製データベースの作成とほぼ同じです。スタンバイ・データベース固有の問題に対処するために、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』で説明されている複製手順を変更する必要があります。

DUPLICATE コマンドを使用してスタンバイ・データベースを作成するには、ターゲットとしてプライマリ・データベースに接続し、FOR STANDBY オプションを指定する必要があります。スタンバイ・データベースに接続してスタンバイ・データベースを追加作成することはできません。Recovery Manager は、制御ファイルをリストアおよびマウントして、スタンバイ・データベースを作成します。Recovery Manager ではプライマリ・データベースの制御ファイルの既存のバックアップを使用できるので、スタンバイ・データベース専用に制御ファイルのバックアップを作成する必要はありません。

FOR STANDBY オプションを指定せずに DUPLICATE コマンドを使用して作成された複製データベースと異なり、スタンバイ・データベースには新しい DBID が設定されません。そのため、スタンバイ・データベースをリカバリ・カタログに登録しないでください。

F.2.2.1 アクティブ・データベース複製とバックアップベース複製

アクティブ複製とバックアップベース複製のいずれかを選択する必要があります。FROM ACTIVE DATABASE を指定すると、Recovery Manager はデータファイルをプライマリ・データベースからスタンバイ・データベースに直接コピーします。プライマリ・データベースをマウントまたはオープンする必要があります。

FROM ACTIVE DATABASE を指定しないと、Recovery Manager はバックアップベース複製を実行します。プライマリ・データファイルのバックアップがスタンバイ・データベースにリストアされます。スタンバイ・データベースの作成およびリカバリに必要なバックアップおよびアーカイブ REDO ログ・ファイルはすべて、スタンバイ・ホストのサーバー・セッションでアクセスできる必要があります。SET UNTIL コマンドを実行しないかぎり、最新のデータファイルがリストアされます。

F.2.2.2 Recovery Manager 環境における DB_UNIQUE_NAME の値

FOR STANDBY オプションを指定せずに DUPLICATE コマンドを使用して作成された複製データベースと異なり、スタンバイ・データベースには新しい DBID が設定されません。Data Guard 環境で Recovery Manager を使用する場合、常にリカバリ・カタログに接続する必要があります。リカバリ・カタログには、環境内のすべてのプライマリ・データベースとスタンバイ・データベースに関するメタデータを格納できます。スタンバイ・データベースは、リカバリ・カタログに明示的に登録しないでください。

Data Guard 環境内のデータベースは、初期化パラメータ・ファイルの DB_UNIQUE_NAME パラメータによって一意に識別する必要があります。DB_UNIQUE_NAME は、Recovery Manager が Data Guard 環境で正しく機能するために、同じ DBID を持つすべてのデータベース間で一意である必要があります。

> **関連項目：** Data Guard 環境での Recovery Manager の動作の概要は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

F.2.2.3 スタンバイ・データベースのリカバリ

デフォルトで、Recovery Manager はスタンバイ・データベースの作成後、リカバリを実行しません。Recovery Manager はスタンバイ・データベースをマウントしたままにしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。Recovery Manager は接続を切断し、スタンバイ・データベースのメディア・リカバリを実行しません。

Recovery Manager でスタンバイ・データベースの作成後にリカバリを行う場合、リカバリに対してスタンバイ制御ファイルが使用可能である必要があります。次の条件を満たしている必要があります。

- スタンバイ・データベースのリカバリ終了時刻は、スタンバイ制御ファイルのチェックポイント SCN 以上である必要があります。
- スタンバイ制御ファイルのチェックポイント SCN が含まれているアーカイブ REDO ログ・ファイルは、リカバリ用にスタンバイ・サイトから使用可能である必要があります。

これらの条件が満たされていることを確認するには、プライマリ・データベースで制御ファイルをバックアップした後に ALTER SYSTEM ARCHIVE LOG CURRENT 文を発行する方法があります。この文は、プライマリ・データベースのオンライン REDO ログ・ファイルをアーカイブします。その後、最新のアーカイブ REDO・ログ・ファイルを Recovery Manager でバックアップするか、またはアーカイブ REDO ログ・ファイルをスタンバイ・サイトに移動します。

DUPLICATE コマンドの DORECOVER オプションを使用して、Recovery Manager でスタンバイ・データベースをリカバリするように指定します。Recovery Manager は、スタンバイ・データベース・ファイルの作成後に次の手順を実行します。

1. Recovery Manager はメディア・リカバリを開始します。アーカイブ REDO ログ・ファイルを必要とするリカバリで、そのログ・ファイルがディスクにない場合、Recovery Manager はバックアップのリストアを試みます。
2. Recovery Manager は、指定された時間、システム変更番号（SCN）またはログ・ファイル順序番号にスタンバイ・データベースをリカバリします。そのいずれも指定されていない場合は、最新の生成済アーカイブ REDO ログ・ファイルにリカバリします。
3. Recovery Manager は、メディア・リカバリの完了後、スタンバイ・データベースをマウントしたままにしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。

F.2.2.4 スタンバイ・データベース REDO ログ・ファイル

Recovery Manager は、スタンバイ・データベースでスタンバイ REDO ログ・ファイルを自動的に作成します。ログ・ファイルは作成後、スタンバイ・データベースによってログ・ファイルの通常のルールに従って管理およびアーカイブされます。

バックアップベース複製を使用した場合、スタンバイ・データベースでスタンバイ REDO ログ・ファイルに名前を付けるときの唯一のオプションは、スタンバイ制御ファイルで指定されるログ・ファイルのファイル名です。スタンバイでのログ・ファイル名をプライマリのファイル名と異なる名前にする必要がある場合は、スタンバイ初期化パラメータ・ファイルで LOG_FILE_NAME_CONVERT を設定して、スタンバイ REDO ログのファイル名を指定する方法を選択できます。

スタンバイ・データベースでスタンバイ REDO ログ・ファイルのファイル名を指定する際、次の制限事項に注意してください。

- プライマリ・データベースおよびスタンバイ・データベースでログ・ファイルに異なるネーミング規則を使用する場合、スタンバイ REDO ログ・ファイルには LOG_FILE_NAME_CONVERT パラメータを使用して名前を付ける必要があります。
- スタンバイ REDO ログ・ファイルの名前の変更には、SET NEWNAME または CONFIGURE AUXNAME コマンドを使用できません。
- スタンバイ REDO ログ・ファイルのファイル名の指定には、DUPLICATE コマンドの LOGFILE 句を使用できません。
- スタンバイ・データベースでのスタンバイ REDO ログ・ファイル名をプライマリ REDO ログ・ファイル名と同じ名前にする場合、DUPLICATE コマンドの NOFILENAMECHECK 句を指定する必要があります。指定しない場合、スタンバイ・データベースが異なるホスト上で作成されていても、Recovery Manager によりエラーが発生します。

F.2.2.5 スタンバイ・データベースのパスワード・ファイル

アクティブ・データベース複製を使用すると、スタンバイ・データベースのパスワード・ファイルはターゲット・データベースのパスワード・ファイルの完全コピーである必要があるため、Recovery Manager は常にパスワード・ファイルをスタンバイ・ホストにコピーします。その場合、PASSWORD FILE 句は必要ありません。補助インスタンスの既存のパスワード・ファイルはすべて上書きされます。バックアップベース複製では、Data Guard によるログの送信のために、プライマリで使用されているパスワード・ファイルをスタンバイにコピーする必要があります。

F.3 DUPLICATE コマンドを使用したスタンバイ・データベースの作成

スタンバイ・データベースの作成手順は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』で説明している複製手順と基本的に同じです。

F.3.1 アクティブ・データベース複製によるスタンバイ・データベースの作成

プライマリ・データベースのアクティブなファイルからスタンバイ・データベースを作成するには、FOR STANDBY と FROM ACTIVE DATABASE の両方を指定します。必要に応じて、DORECOVER オプションを指定し、スタンバイの作成後にデータベースをリカバリします。

この使用例では、スタンバイ・ホストおよびプライマリ・データベース・ホストのディレクトリ構造は同じであることを前提にしています。

スタンバイ・データベースをアクティブなデータベース・ファイルから作成するには、次のようにします。

1. 補助データベース・インスタンスを準備します。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

   アクティブ・データベース複製を使用するため、補助インスタンス用にパスワード・ファイルを作成し、Oracle Net 接続を確立する必要があります。これは、複製操作中に上書きされるので、一時的なパスワード・ファイルです。

2. スタンバイ制御ファイル、データファイル、オンライン REDO ログおよび一時ファイルの名前の指定方法を決定します。この手順の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

   この使用例では、スタンバイ・データベース・ファイルには、プライマリ・データベース・ファイルと同じ名前を付けます。

3. Recovery Manager を起動して構成します。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

4. DUPLICATE コマンドを実行します。

   次の例に、アクティブ複製での DUPLICATE の使用方法を示します。この例では、プライマリ・データベース・ファイルがスタンバイ・データベース・ファイルと同じ名前であるため、NOFILENAMECHECK オプションが必要です。SPFILE の SET 句は、ログの送信が正常に行われるために必要です。db_unique_name を設定して、カタログおよび Data Guard でこのデータベースがプライマリとは異なるものであると識別できるようにする必要があります。

```
DUPLICATE TARGET DATABASE
  FOR STANDBY
  FROM ACTIVE DATABASE
  DORECOVER
  SPFILE
    SET "db_unique_name"="foou" COMMENT ''Is a duplicate''
    SET LOG_ARCHIVE_DEST_2="service=inst3 ASYNC REGISTER
     VALID_FOR=(online_logfile,primary_role)"
    SET FAL_CLIENT="inst3" COMMENT "Is standby"
    SET FAL_SERVER="inst1" COMMENT "Is primary"
  NOFILENAMECHECK;
```

   Recovery Manager は、サーバー・パラメータ・ファイルをスタンバイ・ホストに自動的にコピーし、そのサーバー・パラメータ・ファイルを使用して補助インスタンスを起動してバックアップ制御ファイルをリストアし、必要なデータベース・ファイルおよびアーカイブ REDO ログをすべてスタンバイ・ホストにネットワークを介してコピーします。Recovery Manager はスタンバイ・データベースをリカバリしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。

F.3.2 バックアップベース複製によるスタンバイ・データベースの作成

スタンバイ・データベースをバックアップから作成するには、FOR STANDBY を指定しますが、FROM ACTIVE DATABASE は指定しません。必要に応じて、DORECOVER オプションを指定し、スタンバイの作成後にデータベースをリカバリします。

この使用例では、スタンバイ・ホストおよびプライマリ・データベース・ホストのディレクトリ構造は同じであることを前提にしています。

スタンバイ・データベースをバックアップから作成するには、次のようにします。

1. データベース・バックアップおよびアーカイブ REDO ログを複製ホストの補助インスタンスで使用できるようにします。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
2. 補助データベース・インスタンスを準備します。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
3. スタンバイ制御ファイル、データファイル、オンライン REDO ログおよび一時ファイルの名前の指定方法を決定します。この手順の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

   この使用例では、スタンバイ・データベース・ファイルには、プライマリ・データベース・ファイルと同じ名前を付けます。
4. Recovery Manager を起動して構成します。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
5. DUPLICATE コマンドを実行します。

   次の例に、バックアップベース複製での DUPLICATE の使用方法を示します。この例では、プライマリ・データベース・ファイルがスタンバイ・データベース・ファイルと同じ名前であるため、NOFILENAMECHECK オプションが必要です。

```
DUPLICATE TARGET DATABASE
  FOR STANDBY
  DORECOVER
  SPFILE
    SET "db_unique_name"="foou" COMMENT ''Is a duplicate''
    SET LOG_ARCHIVE_DEST_2="service=inst3 ASYNC REGISTER
     VALID_FOR=(online_logfile,primary_role)"
    SET FAL_CLIENT="inst3" COMMENT "Is standby"
    SET FAL_SERVER="inst1" COMMENT "Is primary"
  NOFILENAMECHECK;
```

   Recovery Manager は、サーバー・パラメータ・ファイルをスタンバイ・ホストに自動的にコピーし、そのサーバー・パラメータ・ファイルを使用して補助インスタンスを起動し、必要なデータベース・ファイルおよびアーカイブ REDO ログをすべてスタンバイ・ホストにリストアします。Recovery Manager はスタンバイ・データベースをリカバリしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。

G

アーカイブ・トレースの設定

Oracle データベースは、LOG_ARCHIVE_TRACE パラメータを使用して、ログ・アーカイブおよび REDO 転送アクティビティに関する包括的なトレース情報の生成を可能にし、制御します。このトレース情報は、Automatic Diagnostic Repository に書き込まれます。

この項は、次の項目で構成されています。

- LOG_ARCHIVE_TRACE 初期化パラメータの設定
- 整数値の選択

> **関連項目：** Automatic Diagnostic Repository の詳細は、『Oracle Database 管理者ガイド』を参照してください。

G.1 LOG_ARCHIVE_TRACE 初期化パラメータの設定

アーカイブ・トレース・パラメータの書式は次のとおりです。*trace_level* は整数です。

```
LOG_ARCHIVE_TRACE=trace_level
```

フィジカル・スタンバイ・データベースで LOG_ARCHIVE_TRACE パラメータの有効化、無効化または変更を行うには、次のような SQL 文を発行します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_TRACE=15;
```

この例では、LOG_ARCHIVE_TRACE パラメータを 15 に設定したため、G.2 項で説明するように、トレース・レベルが 1、2、4 および 8 に設定されます。

次のアーカイブ REDO ログ・ファイルをプライマリ・データベースから受信したときに、リモート・ファイル・サーバー（RFS）および ARC*n* プロセスによって生成されたトレース出力に作用するように、別のスタンバイ・セッションから ALTER SYSTEM 文を発行します。たとえば、次のように入力します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_TRACE=32;
```

G.2 整数値の選択

LOG_ARCHIVE_TRACE パラメータの整数値は、データのトレース・レベルを表します。一般的には、レベルが高いほど、情報が詳細になります。次の整数値を使用できます。

レベル	意味
0	アーカイブ REDO ログのトレースを使用不可能にします（デフォルト設定）。
1	ログ・ファイルのアーカイブを追跡します。
2	アーカイブ・ログ・ファイルの宛先ごとにアーカイブ状況を追跡します。
4	アーカイブ操作のフェーズを追跡します。
8	アーカイブ・ログの宛先のアクティビティを追跡します。
16	アーカイブ・ログの宛先の詳細なアクティビティを追跡します。
32	アーカイブ・ログの宛先のパラメータ修正を追跡します。
64	ARC*n* プロセスの状態のアクティビティを追跡します。
128	FAL サーバー・プロセスのアクティビティを追跡します。
256	RFS の論理的なクライアントを追跡します。
512	LGWR の REDO 転送ネットワーク・アクティビティを追跡します。
1024	RFS の物理的なクライアントを追跡します。
2048	RFS/ARC*n* の ping のハートビートを追跡します。
4096	リアルタイム適用アクティビティを追跡します。
8192	REDO Apply アクティビティ（メディア・リカバリまたはフィジカル・スタンバイ）を追跡します。

LOG_ARCHIVE_TRACE パラメータの値をいくつかのトレース・レベルを合計した値に設定すると、トレース・レベルを組み合せることができます。たとえば、パラメータを 6 に設定すると、レベル 2 とレベル 4 のトレース出力が生成されます。

次に、ログ・ファイル 387 を 2 つの異なる宛先（サービス standby1 およびローカル・ディレクトリ /oracle/dbs）にアーカイブすることによってプライマリ・サイトに生成される ARC0 トレース・データの例を示します。

注意： レベルの数値は実際のトレース出力には表示されません。ここでは説明の目的で示してあります。

```
Level   Corresponding entry content (sample)
-----   -------------------------------
( 1)    ARC0: Begin archiving log# 1 seq# 387 thrd# 1
( 4)    ARC0: VALIDATE
( 4)    ARC0: PREPARE
( 4)    ARC0: INITIALIZE
( 4)    ARC0: SPOOL
( 8)    ARC0: Creating archive destination 2 : 'standby1'
(16)    ARC0:  Issuing standby Create archive destination at 'standby1'
( 8)    ARC0: Creating archive destination 1 : '/oracle/dbs/d1arc1_387.log'
(16)    ARC0:  Archiving block 1 count 1 to : 'standby1'
(16)    ARC0:  Issuing standby Archive of block 1 count 1 to 'standby1'
(16)    ARC0:  Archiving block 1 count 1 to :  '/oracle/dbs/d1arc1_387.log'
( 8)    ARC0: Closing archive destination 2  : standby1
(16)    ARC0:  Issuing standby Close archive destination at 'standby1'
( 8)    ARC0: Closing archive destination 1  :  /oracle/dbs/d1arc1_387.log
( 4)    ARC0: FINISH
( 2)    ARC0: Archival success destination 2 : 'standby1'
( 2)    ARC0: Archival success destination 1 : '/oracle/dbs/d1arc1_387.log'
( 4)    ARC0: COMPLETE, all destinations archived
(16)    ARC0: ArchivedLog entry added: /oracle/dbs/d1arc1_387.log
(16)    ARC0: ArchivedLog entry added: standby1
( 4)    ARC0: ARCHIVED
( 1)    ARC0: Completed archiving log# 1 seq# 387 thrd# 1

(32)  Propagating archive 0 destination version 0 to version 2
        Propagating archive 0 state version 0 to version 2
        Propagating archive 1 destination version 0 to version 2
        Propagating archive 1 state version 0 to version 2
        Propagating archive 2 destination version 0 to version 1
        Propagating archive 2 state version 0 to version 1
        Propagating archive 3 destination version 0 to version 1
        Propagating archive 3 state version 0 to version 1
        Propagating archive 4 destination version 0 to version 1
        Propagating archive 4 state version 0 to version 1

(64) ARCH: changing ARC0 KCRRNOARCH->KCRRSCHED
       ARCH: STARTING ARCH PROCESSES
       ARCH: changing ARC0 KCRRSCHED->KCRRSTART
       ARCH: invoking ARC0
       ARC0: changing ARC0 KCRRSTART->KCRRACTIVE
       ARCH: Initializing ARC0
       ARCH: ARC0 invoked
       ARCH: STARTING ARCH PROCESSES COMPLETE
       ARC0 started with pid=8
       ARC0: Archival started
```

次に示すトレース・データは、スタンバイ・サイトで、ディレクトリ /stby にアーカイブ REDO ログ・ファイル 387 を受信し、それをスタンバイ・データベースに適用する RFS プロセスによって生成されたものです。

```
level    trace output (sample)
----    ------------------
( 4)      RFS: Startup received from ARCH pid 9272
( 4)      RFS: Notifier
( 4)      RFS: Attaching to standby instance
( 1)      RFS: Begin archive log# 2 seq# 387 thrd# 1
(32)      Propagating archive 5 destination version 0 to version 2
(32)      Propagating archive 5 state version 0 to version 1
( 8)      RFS: Creating archive destination file: /stby/parc1_387.log
(16)      RFS:  Archiving block 1 count 11
( 1)      RFS: Completed archive log# 2 seq# 387 thrd# 1
( 8)      RFS: Closing archive destination file: /stby/parc1_387.log
(16)      RFS: ArchivedLog entry added: /stby/parc1_387.log
( 1)      RFS: Archivelog seq# 387 thrd# 1 available 04/02/99 09:40:53
( 4)      RFS: Detaching from standby instance
( 4)      RFS: Shutdown received from ARCH pid 9272
```

索引

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

き

け

こ

さ

し

す

と

ね

は

ひ

ふ

へ

ほ

ま

む

め