Oracle9*i* Data Guard

概要および管理

リリース 2（9.2）

2003 年 2 月

部品番号：J06272-02

ORACLE®

Oracle9*i* Data Guard 概要および管理 , リリース 2（9.2）

部品番号：J06272-02

原本名：Oracle Data Guard Concepts and Administration, Release 2 (9.2)

原本部品番号：A96653-02

目次

4 ロジカル・スタンバイ・データベースの作成

5 ログ転送サービス

6 ログ適用サービス

7 ロール管理

8 フィジカル・スタンバイ・データベースの管理

9 ロジカル・スタンバイ・データベースの管理

10 Data Guard の使用例

第 II 部　リファレンス

11 初期化パラメータ

12 LOG_ARCHIVE_DEST_*n* パラメータの属性

13 SQL 文

14 ビュー

C Real Application Clusters でのスタンバイ・データベースのサポート

D カスケードされた REDO ログ宛先

例

図

表

はじめに

スタンバイ・データベースは、プライマリ・データベースが使用不可能になった場合に本番システムの実行に使用できるため、最も効率的な Oracle データベース用障害時リカバリ・ソリューションとなります。スタンバイ・データベースは、ユーザーのミス、データ破損およびその他の操作時障害によって生じた問題の回復にも使用できます。

このマニュアルでは、Oracle Data Guard の概要と、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。

「はじめに」は、次の項目で構成されています。

対象読者

『Oracle9*i* 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 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。

このマニュアルの構成

このマニュアルの構成は次のとおりです。

第 I 部「概要および管理」

第 1 章「Oracle Data Guard の概要」

Oracle9*i* Data Guard のアーキテクチャの概要について説明します。

第 2 章「Data Guard スタート・ガイド」

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・データベースおよびカスケード・スタンバイ・データベースについて説明します。

第 3 章「フィジカル・スタンバイ・データベースの作成」

フィジカル・スタンバイ・データベースを作成し、REDO ログの適用を開始する方法について説明します。

第 4 章「ロジカル・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し、REDO ログの適用を開始する方法について説明します。

第 5 章「ログ転送サービス」

ログ転送サービスについて紹介します。本番データベースが予定外に停止した場合のデータ消失から保護するためのデータ保護モードについて説明します。また、プライマリ・データベースとスタンバイ・データベースでログ転送サービスを構成するための手順とガイドラインを示します。

第 6 章「ログ適用サービス」

ログ適用サービスについて説明します。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・データベース用のログ適用サービスを管理するためのガイドラインを示します。

第 7 章「ロール管理」

ロール管理サービスについて説明します。フェイルオーバーおよびスイッチオーバー時のデータベース・ロールの推移に関する情報を提供します。

第 8 章「フィジカル・スタンバイ・データベースの管理」

フィジカル・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。データベース・ロールに影響を与えるイベントの監視と応答に関する情報を提供します。

第 9 章「ロジカル・スタンバイ・データベースの管理」

ロジカル・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。REDO ログの適用、システム・チューニングおよび表領域管理に関する情報を提供します。

第 10 章「Data Guard の使用例」

スタンバイおよびプライマリ・データベースの作成、リカバリ、フェイルオーバー、切替え、構成およびバックアップなどの共通のデータベース操作について説明します。

第 II 部「リファレンス」

第 11 章「初期化パラメータ」

Data Guard 環境のプライマリ・データベースと各スタンバイ・データベースを含む、各 Oracle インスタンスの初期化パラメータについて説明します。

第 12 章「LOG_ARCHIVE_DEST_n パラメータの属性」

`LOG_ARCHIVE_DEST_n` 初期化パラメータの属性の構文と例を示します。

第 13 章「SQL 文」

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 文を紹介します。

第 14 章「ビュー」

Data Guard 環境の監視に役立つ情報を含むビューをリストします。各ビューに表示される列をまとめ、それぞれの列について説明します。

第 III 部「付録および用語集」

付録 A「スタンバイ・データベースのトラブルシューティング」

スタンバイ・データベースのトラブルシューティングについて説明します。

付録 B「手動リカバリ」

手動リカバリ・モードでのフィジカル・スタンバイ・データベースの管理について説明します。アーカイブ・ギャップを手動で解決する手順や、変換パラメータによって獲得されないスタンバイ・ファイルの名前を変更する手順を紹介します。

付録 C「Real Application Clusters でのスタンバイ・データベースのサポート」

Real Application Clusters 環境のプライマリおよびスタンバイ・データベースの構成について説明します。

付録 D「カスケードされた REDO ログ宛先」

カスケードされた REDO ログ宛先の実装方法について説明します。これによって、スタンバイ・データベースは、REDO ログをプライマリ・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。

付録 E「障害時リカバリに関する ReadMe ファイルのサンプル」

障害時リカバリを行う担当者が、どのスタンバイ・データベースをフェイルオーバー操作のターゲットとするかを決定する上で必要な情報が含まれた、ReadMe ファイルのサンプルを提供します。

用語集

関連文書

このマニュアルの読者は、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。

- 『Oracle9*i* データベース概要』の冒頭部分。Oracle データベース・サーバーに関連する概念と用語の概要、およびこのマニュアルに記載されている詳細情報の基礎となる内容が記載されています。その後の部分では、Oracle のアーキテクチャと特性について詳細に説明しています。
- 『Oracle9*i* データベース管理者ガイド』の中で、制御ファイル、オンライン REDO ログおよびアーカイブ REDO ログの管理について説明している章。

次の各ガイドも、必要に応じて参照します。

- 『Oracle9*i* Data Guard Broker』
- 『Oracle9*i* SQL リファレンス』
- 『Oracle9*i* データベース・リファレンス』
- 『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』
- 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

- 『Oracle9*i* 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』

既存のスタンバイ・データベースを現行の Oracle9*i* リリースに移行する場合の詳細な手順は、『Oracle9*i* データベース移行ガイド』を参照してください。障害時リカバリとデータの高可用性ソリューションを提供するその他の Oracle 製品と特性に関する情報は、『Oracle9*i* データベース概要』を参照してください。

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

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

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

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

表記規則

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

- 本文の表記規則
- コード例の表記規則

本文の表記規則

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

表記	意味	例
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。	この句を指定する場合は、**索引構成表**を作成します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージとメソッド、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は、NUMBER 列に対してのみ指定できます。 BACKUP コマンドを使用すると、データベースのバックアップを作成できます。 USER_TABLES データ・ディクショナリ・ビューの TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。

表記	意味	例
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネットワーク・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニット、およびパラメータ値があります。	sqlplus と入力して SQL*Plus をオープンします。 パスワードは orapwd ファイルに指定されています。 データ・ファイルと制御ファイルのバックアップを /disk/oracle/dbs ディレクトリに作成します。 department_id、department_name および location_id の各列は、hr.departments 表にあります。 初期化パラメータ QUERY_REWRITE_ENABLED を TRUE に設定します。
	注意： 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。	oe ユーザーで接続します。 これらのメソッドは JRepUtil クラスに実装されます。
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。*old_release* は、アップグレード前にインストールしたリリースです。

コード例の表記規則

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

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

次の表に、コード例の記載上の表記規則と使用例を示します。

表記	意味	例
[ ]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (digits [, precision])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]

表記	意味	例
. . .	水平の省略記号は、次のどちらかを示します。	
	■ 例に直接関係のないコード部分が省略されていること。	CREATE TABLE ...AS *subquery*;
	■ コードの一部が繰返し可能であること。	SELECT *col1*, *col2*, ..., *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fs1/dbs/tbs_01/dbf /fs1/dbs/tbs_02/dbf . . . /fs1/dbs/tbs_09/dbf 9 rows selected.
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、示されているとおりに入力してください。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、小文字でも入力できます。	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。 **注意 :** 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Data Guard の新機能

ここでは、Oracle9*i* Data Guard リリース 2（9.2）の新機能について説明するとともに、関連項目の記載箇所も示します。現行リリースへアップグレードするユーザーのために、旧リリースの新機能情報についても説明します。

次の項目があります。

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

Oracle9*i* リリース 2（9.2）Data Guard の新機能

Oracle9*i* リリース 2（9.2）の Data Guard には、この項で説明する新機能と拡張機能が追加されました。

- **ロジカル・スタンバイ・データベース**

  以前のリリースでは、フィジカル・スタンバイ・データベースのみが実装されており、管理リカバリ操作または読取り専用操作のいずれか一方を実行できました。フィジカル・スタンバイ・データベースは、プライマリ・データベースと物理的に同一です。フィジカル・スタンバイ・データベースは、REDO ログを適用しているときはレポート生成用にオープンできず、レポート生成用にオープンしているときは REDO ログを適用できません。ロジカル・スタンバイ・データベースには、プライマリ・データベースと論理的に同じスキーマがありますが、追加の索引など、物理的に異なるオブジェクトが含まれています。ロジカル・スタンバイ・データベースを使用すると、レポート生成と REDO ログの適用を同時に実行できます。

- **データ保護モード**

  データベース管理者（DBA）は、データベースを次のいずれかのモードにすることができます。

  – 最大保護

  – 最大可用性

  – 最大パフォーマンス

  これらのモードは、Oracle9*i* リリース 1（9.0.1）で使用可能な、保証付き、即時、迅速および遅延の各データ保護モードに置き換わるものです。

  関連項目： 第 5 章「ログ転送サービス」および 第 13 章「SQL 文」

- **カスケードされた REDO ログ宛先**

  カスケードされた REDO ログ宛先は、REDO データを元のプライマリ・データベースではなく、別のスタンバイ・データベースから受信するスタンバイ・データベースです。フィジカルまたはロジカル・スタンバイ・データベースを設定すると、1 レベルまでのリダイレクションで、プライマリ・データベースと同じ方法で着信 REDO データを他のリモートの宛先に送信できます。

  関連項目： 付録 D「カスケードされた REDO ログ宛先」

- **Oracle9*i* Data Guard Broker**

  Oracle9*i* Data Guard Broker は、次の内容をサポートします。

  - 最大 9 箇所のフィジカルまたはロジカル・スタンバイ宛先
  - フェイルオーバー操作とスイッチオーバー操作

  関連項目： 『Oracle9*i* Data Guard Broker』

- **`REMOTE_ARCHIVE_ENABLE` 初期化パラメータの新規キーワード**

  - `SEND`
  - `RECEIVE`

  関連項目： 5-5 ページ「REDO ログをアーカイブするための権限の設定」

- **`LOG_ARCHIVE_DEST_n` 初期化パラメータの新規属性**

  - `[NO]TEMPLATE`

    アーカイブ REDO ログのスタンバイ宛先のディレクトリ指定と形式を定義します。

  - `[NO]NET_TIMEOUT`

    ログ・ライター・プロセスが、発行したネットワーク操作に関するネットワーク・サーバーからのステータスを待機する秒数を指定します。

  - `SYNC` 属性に対する `PARALLEL` 修飾子

    複数の宛先への I/O 操作をパラレルまたはシリアルのいずれで実行するかを示します。

  関連項目： 第 12 章「LOG_ARCHIVE_DEST_n パラメータの属性」

- **`ALTER DATABASE` 文に追加された新規構文**

  - `ACTIVATE [PHYSICAL | LOGICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]]`
  - `COMMIT TO SWITCHOVER TO {PHYSICAL | LOGICAL} {PRIMARY | STANDBY} [[WITH | WITHOUT] SESSION SHUTDOWN [WAIT | NOWAIT]]`
  - `[NO]FORCE LOGGING`
  - `RECOVER MANAGED STANDBY DATABASE [FINISH [SKIP [STANDBY LOGFILE] [WAIT | NOWAIT]]`
  - `RECOVER MANAGED STANDBY DATABASE [THROUGH {ALL | NEXT | LAST} SWITCHOVER]`

- RECOVER MANAGED STANDBY DATABASE [THROUGH ALL ARCHIVELOG | [THREAD *n*] SEQUENCE *n*]
- REGISTER [OR REPLACE] [PHYSICAL | LOGICAL] LOGFILE *filespec*
- SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE}
- START LOGICAL STANDBY APPLY
- {STOP | ABORT} LOGICAL STANDBY APPLY

関連項目： 第 13 章「SQL 文」

■ **新規ビューの追加**

- DBA_LOGSTDBY_EVENTS
- DBA_LOGSTDBY_LOG
- DBA_LOGSTDBY_NOT_UNIQUE
- DBA_LOGSTDBY_PARAMETERS
- DBA_LOGSTDBY_PROGRESS
- DBA_LOGSTDBY_SKIP
- DBA_LOGSTDBY_SKIP_TRANSACTION
- DBA_LOGSTDBY_UNSUPPORTED
- V$DATAGUARD_STATUS
- V$LOGSTDBY
- V$LOGSTDBY_STATS

関連項目： 第 14 章「ビュー」

■ **既存の固定ビューへの新規の列の追加**

- V$ARCHIVE_DEST ビュー
  - NET_TIMEOUT
  - TYPE
- V$ARCHIVE_DEST_STATUS ビュー
  - PROTECTION_MODE
  - SRL

- V$DATABASE ビュー
  - GUARD_STATUS
  - SUPPLEMENTAL_LOG_DATA_MIN
  - SUPPLEMENTAL_LOG_DATA_PK
  - SUPPLEMENTAL_LOG_DATA_UI
  - FORCE_LOGGING
  - PROTECTION_LEVEL

  関連項目： 第 14 章「ビュー」

- **既存の固定ビュー内の既存列の名称変更**
  - V$ARCHIVE_DEST ビュー
    - MANIFEST は REGISTER に名称変更され、値は YES と NO に変更されました。
    - REGISTER は REMOTE_TEMPLATE に名称変更されました。
  - V$DATABASE ビュー
    - STANDBY_MODE は PROTECTION_MODE に名称変更され、MAXIMUM PROTECTED、MAXIMUM AVAILABILITY、RESYNCHRONIZATION、MAXIMUM PERFORMANCE および UNPROTECTED の値が追加されました。

  関連項目： 第 14 章「ビュー」

- **既存の固定ビューの既存列への新規の値の追加**
  - V$ARCHIVE_DEST ビューの TRANSMIT_MODE 列
    - PARALLELSYNC
    - SYNCHRONOUS
    - ASYNCHRONOUS
  - V$DATABASE ビューの REMOTE_ARCHIVE 列
    - SEND
    - RECEIVE

  関連項目： 第 14 章「ビュー」

- **LOG_ARCHIVE_TRACE パラメータに使用する新規の整数値の追加**
  - 1024: RFS の物理クライアント・トラッキング
  - 2048: ARC*n* または RFS のハートビート・トラッキング

  関連項目： 6-27 ページ「整数値の選択」

Oracle9*i* リリース 1（9.0.1）Data Guard の新機能

Oracle9*i* リリース 1（9.0.1）の Data Guard には、この項で説明する新機能と拡張機能が追加されました。

- **Oracle9*i* Data Guard**

  Oracle8*i* スタンバイ・データベースは、Oracle9*i* Data Guard に名称変更されました。
- **Oracle9*i* Data Guard Broker**
- **非データ消失**
- **データベースのスイッチオーバー**
- **アーカイブ・ギャップを自動的に検出および解消**
- **新しいデータ・ファイルをプライマリ・データベースに追加可能。対応するデータ・ファイルをスタンバイ・データベースに手動で追加する必要はありません。**
- **バックグラウンド管理リカバリ・モード**
- **パラレル・リカバリにより、フィジカル・スタンバイ・データベースでのリカバリを高速化**
- **アーカイブ先を 10 箇所まで指定可能**
- **LOG_ARCHIVE_DEST_*n* 初期化パラメータの個々の属性を段階的に変更可能**
- **スタンバイ REDO ログ**
- **フィジカル・スタンバイ・データベースのアーカイバ・プロセス（ARC*n*）によるスタンバイ REDO ログのアーカイブが可能**
- **カレント REDO ログまたはオンライン REDO ログのアーカイブ。データベースがマウントされたがオープンされていないときに、システム変更番号（SCN）値に基づいてアーカイブできます。また、バックアップ制御ファイルが使用されているときにオンライン REDO ログをアーカイブできます。従来のリリースでは、現行の制御ファイルが必要でした。**
- **新規制御オプション：DELAY、DISCONNECT、EXPIRE、FINISH、NEXT、NODELAY**
- **Real Application Clusters でのスタンバイ・データベースのサポート**

- **新規アーカイブ・ログ・リポジトリ。このリポジトリは、スタンドアロンのスタンバイ・データベースです。**
- **アーカイブ REDO ログとアーカイブ先の間に定義された関係**
- **新規の初期化パラメータ : REMOTE_ARCHIVE_ENABLE、FAL_CLIENT、FAL_SERVER、STANDBY_FILE_MANAGEMENT、ARCHIVE_LAG_TARGET**
- **LOG_ARCHIVE_DEST_*n* 初期化パラメータの新規属性**
  - ARCH | LGWR
  - [NO]AFFIRM
  - [NO]ALTERNATE
  - [NO]DELAY
  - [NO]DEPENDENCY
  - [NO]MAX_FAILURE
  - [NO]QUOTA_SIZE
  - [NO]QUOTA_USED
  - [NO]REGISTER | REGISTER [=*location_format*]
  - NOREOPEN
  - SYNC | ASYNC
- **LOG_ARCHIVE_DEST_STATE_*n* 初期化パラメータに使用する値の範囲および ALTERNATE キーワードの追加**
- **1 ～ 10 までのすべてのアーカイブ先（Oracle8*i* では 1 ～ 5 箇所）が正常にアーカイブしてから、ログ・ライター・プロセス（LGWR）によるオンライン REDO ログの上書きが可能**
- **LOG_ARCHIVE_TRACE 初期化パラメータに使用する新規トレース・レベル（128、256 および 512）の追加**
- **ALTER DATABASE 文に使用する新規の句の追加**
  - ACTIVATE [PHYSICAL] STANDBY DATABASE
    [SKIP [STANDBY LOGFILE]]
  - ADD [STANDBY] LOGFILE TO [THREAD *integer*]
    [GROUP *integer*] *filespec*
  - ADD [STANDBY] LOGFILE MEMBER '*filename*' [REUSE] TO
    '*logfile-descriptor*'
  - COMMIT TO SWITCHOVER TO [PHYSICAL]
    {PRIMARY | STANDBY} [[NO]WAIT]

- REGISTER [PHYSICAL] LOGFILE *filespec*
- SET STANDBY DATABASE {PROTECTED | UNPROTECTED}

注意： Oracle9*i* リリース 2（9.2）では、この構文はさらに変更されました。「Oracle9i リリース 2（9.2）Data Guard の新機能」の項を参照してください。

■ **RECOVER MANAGED STANDBY DATABASE 句に使用する新規キーワードの追加**

- NODELAY
- CANCEL [IMMEDIATE] [NOWAIT]
- [DISCONNECT [FROM SESSION]]
- [FINISH [NOWAIT]]
- [PARALLEL [*integer*]]
- NEXT
- EXPIRE
- DELAY

注意： Oracle9*i* リリース 2（9.2）では、この構文はさらに変更されました。「Oracle9i リリース 2（9.2）Data Guard の新機能」の項を参照してください。

■ **新規固定ビューの追加**

- V$ARCHIVE_DEST_STATUS
- V$ARCHIVE_GAP
- V$MANAGED_STANDBY
- V$STANDBY_LOG

関連項目： 第 14 章「ビュー」

- **既存の固定ビューへの新規の列の追加**
  - V$ARCHIVE_DEST ビュー
    - AFFIRM
    - ALTERNATE
    - ARCHIVER
    - ASYNC_BLOCKS
    - DELAY_MINS
    - DEPENDENCY
    - FAILURE_COUNT
    - LOG_SEQUENCE
    - MANIFEST（Oracle9*i* リリース 1（9.0.1）で新規に追加されたこの列は、Oracle9*i* リリース 2（9.2）で REGISTER に名称変更されました。）
    - MAX_FAILURE
    - MOUNTID
    - PROCESS
    - QUOTA_SIZE
    - QUOTA_USED
    - REGISTER（Oracle9*i* リリース 1（9.0.1）で新規に追加されたこの列は、Oracle9*i* リリース 2（9.2）で REMOTE_TEMPLATE に名称変更されました。）
    - SCHEDULE
    - TRANSMIT_MODE
    - TYPE
    - STATUS 列に追加された新規の値 ALTERNATE および FULL
  - V$ARCHIVED_LOG ビュー
    - APPLIED
    - BACKUP_COUNT
    - COMPLETION_TIME
    - CREATOR
    - DELETED
    - DEST_ID

    - DICTIONARY_BEGIN
    - DICTIONARY_END
    - REGISTRAR
    - STANDBY_DEST
    - STATUS
    - END_OF_REDO
    - ARCHIVAL_THREAD#
- V$LOG ビュー
    - STATUS 列に追加された新規の値 INVALIDATED
- V$LOGFILE ビュー
    - TYPE
- V$DATABASE ビュー
    - ACTIVATION#
    - ARCHIVELOG_CHANGE#
    - DATABASE_ROLE
    - REMOTE_ARCHIVE
    - STANDBY_MODE
    - SWITCHOVER_STATUS
- V$ARCHIVE_DEST_STATUS ビュー
    - STANDBY_LOGFILE_COUNT
    - STANDBY_LOGFILE_ACTIVE

注意： Oracle9*i* リリース 2（9.2）では、V$DATABASE および V$ARCHIVE_DEST 固定ビューで新規の値が追加され、既存の列の名前が変更されました。「Oracle9i リリース 2（9.2）Data Guard の新機能」の項を参照してください。

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、スタンバイ・データベースを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ることによって、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。

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。

Data Guard 構成

Data Guard 構成には、1 個の本番データベースと最大 9 個のスタンバイ・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、コマンドライン・インタフェースまたは Data Guard Broker を使用して管理できます。ブローカには、Oracle Data Guard Manager というグラフィカル・ユーザー・インタフェースが含まれています。

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、シングル・インスタンスの Oracle データベースまたは Oracle Real Application Clusters データベースのいずれかです。

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。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。

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、シングル・インスタンスの Oracle データベースまたは Oracle Real Application Clusters 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、次のフィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースのいずれかです。

- **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のコピーで、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 データをリカバリすることによって、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**

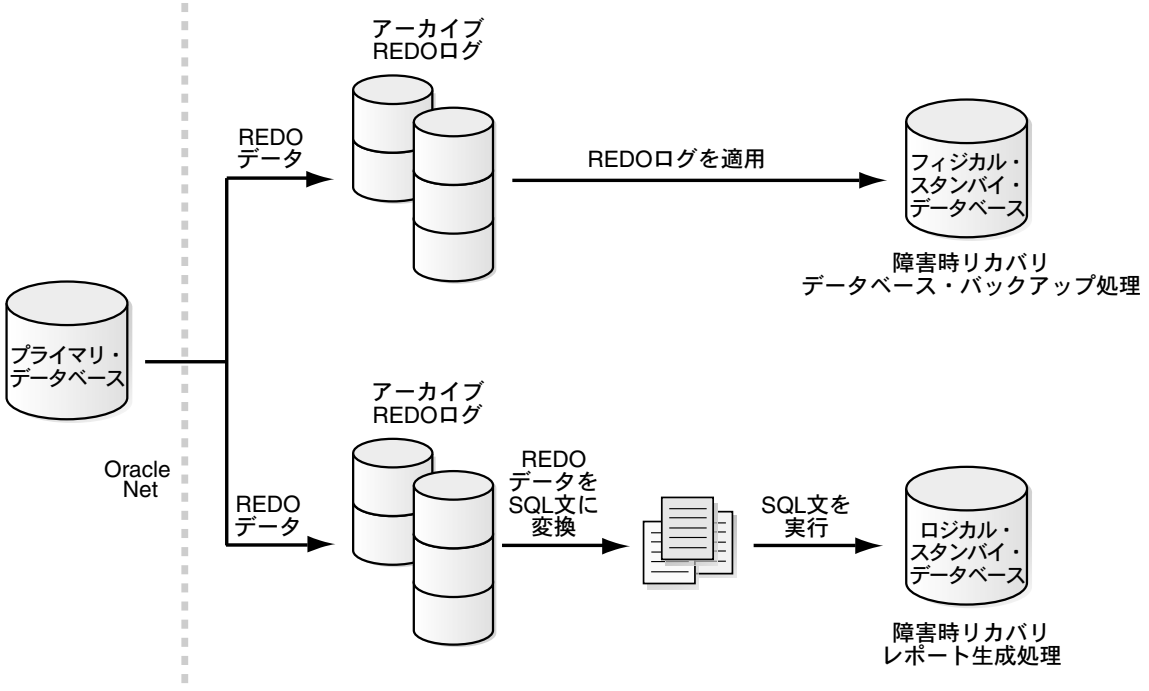
  本番データベースと同じ論理情報が格納されますが、データの物理的な構成と構造は異なる場合があります。ロジカル・スタンバイ・データベースでは、プライマリ・データベースから受信した REDO ログのデータを SQL 文に変換し、スタンバイ・データベース上でその SQL 文を実行することによって、プライマリ・データベースとの同期を維持します。ロジカル・スタンバイ・データベースは、障害時リカバリ要件に加えて、その他のビジネス用途にも使用できます。このため、ロジカル・スタンバイ・データベースには、問合せやレポート生成の目的でいつでもアクセスできます。つまり、ロジカル・スタンバイ・データベースは、データ保護とレポート生成に同時に使用できます。

構成例

図 1-1 は、プライマリ・データベース・インスタンスから離れた位置にあるフィジカル・スタンバイ・データベースとロジカル・スタンバイ・データベースに、REDO データを転送するプライマリ・データベース・インスタンスが含まれた Data Guard 構成を示しています。この構成では、フィジカル・スタンバイ・データベースは障害時リカバリとバックアップ操作用に構成されています。ロジカル・スタンバイ・データベースは主にレポート生成用に構成されていますが、障害時リカバリ用にも使用できます。これらのスタンバイ・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 は、アーカイブ REDO ログをフィジカル・スタンバイ・データベースとロジカル・スタンバイ・データベースの両方に適用する一般的な Data Guard 構成を示しています。

図 1-1 一般的な Data Guard 構成



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 ログ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。

- ログ転送サービス

  Data Guard 構成内で REDO データの自動転送を制御します。

- ログ適用サービス

  アーカイブ REDO ログをスタンバイ・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。

- ロール管理サービス

  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。

ログ転送サービス

ログ転送サービスは、Data Guard 構成内で REDO データの自動転送を制御します。

ログ転送サービスでは、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。
- データベース保護モード（「Data Guard 保護モード」を参照）を施行します。

ログ適用サービス

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・システム上でアーカイブ REDO ログの形式でアーカイブされます。**ログ適用サービス**は、アーカイブ REDO ログをスタンバイ・データベースに自動的に適用して、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。また、データに対するトランザクション一貫性のある読取り専用アクセスを可能にします。

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、ログ適用サービスによるアーカイブ REDO ログ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 適用**テクノロジを使用します。このテクノロジでは、Oracle データベース・サーバーの標準リカバリ技法を使用して REDO データをスタンバイ・データベースに適用します。
- ロジカル・スタンバイ・データベースの場合、Data Guard では **SQL 適用**テクノロジを使用します。このテクノロジでは最初に、受信した REDO データを SQL 文に変換した後、生成された SQL 文をロジカル・スタンバイ・データベース上で実行します。

ログ適用サービスでは、次のタスクを実行します。

- アーカイブ REDO ログをスタンバイ・データベースに自動的に適用します。
- スタンバイ・システム上の欠落している REDO ログを自動的に検出したり、プライマリ・データベースまたは別のスタンバイ・データベースから欠落している REDO ログを自動的に取得します。

ロール管理サービス

Oracle データベースは、次の 2 つのロールのいずれかで実行されます。2 つのロールとは、プライマリとスタンバイです。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。このロールを制御するサービスは、**ロール管理サービス**と呼ばれます。

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。スイッチオーバー操作では、データ消失がないことが保証されます。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。推移は、いずれのデータベースも再作成せずに実行されます。

フェイルオーバーとは、スタンバイ・データベースをプライマリ・ロールに不可逆的に推移させる操作です。この操作を実行するのは、プライマリ・データベースで災害などの障害が起きたときのみです。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。

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および集中化します。次に、ブローカによる自動化または単純化の操作について説明します。

- ログ転送サービスやログ適用サービスの設定なども含めて、1 つ以上の Data Guard 構成を作成し有効化します。
- 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・データベースを作成します。
- 新規または既存のスタンバイ・データベースを既存の Data Guard 構成に追加します。
- 構成内の任意のシステムから Data Guard 構成全体を管理します。
- ログ適用レートの監視、診断情報の獲得、および集中化した監視ツール、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を行います。

Data Guard 保護モード

ビジネスによっては、どのような状況でもデータの消失を避けたい企業があります。また、データの消失よりデータベースの可用性のほうが重要な企業もあります。アプリケーションの中には、データベース・パフォーマンスを最大化することが必要で、データ消失の可能性を許容できるものがあります。

Data Guard には、次の 3 つの異なるデータ保護モードが用意されています。

- **最大保護**

  このモードは、最も高いレベルのデータ保護を提供します。データは、プライマリ・データベースからスタンバイ・データベースに同期して転送されます。また、REDO データがこのモードで構成されたスタンバイ・データベースの少なくとも 1 つで使用可能にならないかぎり、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ません。このモードで構成された最後のスタンバイ・データベースが使用不可能になると、プライマリ・データベースでの処理が停止します。このモードでは、データ消失がないことが保証されます。

- **最大可用性**

  このモードは、データ消失がないことが保証される点など、最大保護モードと似ています。ただし、スタンバイ・データベースがネットワーク接続の問題などで使用不可能になった場合でも、プライマリ・データベースでの処理は続行されます。障害が解決されると、スタンバイ・データベースはプライマリ・データベースと再同期化されます。スタンバイ・データベースの再同期化前にフェイルオーバーが必要な場合は、一部のデータが消失する可能性があります。

- 最大パフォーマンス

  このモードでは、プライマリ・データベースでのデータ保護のレベルが少し低下します。ただし、パフォーマンス・レベルは最大可用性モードよりも高くなります。このモードでは、プライマリ・データベースがトランザクションを処理すると、REDO データは非同期でスタンバイ・データベースに送り出されます。プライマリ・データベースでのコミット操作は、スタンバイ・データベースによる REDO データの受信を確認するまで待機せずに、プライマリ・データベースでの書込み操作を完了します。スタンバイ宛先が使用不可能になった場合でも、プライマリ・データベースでの処理は続行されます。したがって、プライマリ・データベースのパフォーマンスへの影響はほとんどありません。

Data Guard のメリットの要約

Data Guard には、各種のスタンバイ・データベースにあるメリットの他に、全体的なメリットが多数あります。Data Guard 構成の設計時には、各タイプのスタンバイ・データベース固有のメリットも含めて、すべてのメリットを考慮する必要があります。

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、データ保護および高可用性を提供します。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

  スタンバイ・データベースを使用することで、予期しない障害が発生した場合でもデータが消失しないことが保証されます。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。最後に、REDO データが、スタンバイ・データベースへの適用時に検証されます。

- 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 ログで更新されたスタンバイ・データベース表は、バックアップ操作、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。ロジカル・スタンバイ・データベースを使用すると、プライマリ・データベースに基づいて更新されていないスキーマ内の各表で通常のデータ操作を実行できます。ロジカル・スタンバイ・データベースは、表をプライマリ・データベースから更新する間、オープン状態のままにしておくことができるため、表は読取り専用アクセスに同時に使用できます。最後に、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、問合せパフォーマンスを改善し、特定のビジネス要件を満たすことができます。

- 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。

- 集中化された単純な管理

  Data Guard Broker は、Data Guard Manager のグラフィカル・ユーザー・インタフェースと、Data Guard 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。

- 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、欠落しているログ順序番号（またはギャップ）が自動的に検出され、必要な REDO ログがスタンバイ・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・データベースと再同期化されるため、DBA による手動操作は不要です。

2

Data Guard スタート・ガイド

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の選択
- Data Guard のユーザー・インタフェースの選択
- 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

スタンバイ・データベースのタイプの選択

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（ここで REDO データがアーカイブされます）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・データベースには、**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 2 つのタイプがあります。いずれのタイプのスタンバイ・データベースも、必要に応じてプライマリ・データベースのロールを引き継ぎ、本番処理を継続できます。Data Guard 構成には、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、または両方のタイプの組合せを含めることができます。

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・データベースと物理的に同一で、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・スキーマは同じであることが必要です。

Data Guard は、管理リカバリ操作を実行することによって、フィジカル・スタンバイ・データベースをメンテナンスします。管理リカバリ操作が実行されていないときは、フィジカル・スタンバイ・データベースを読取り専用操作用にオープンできます。

- **管理リカバリ**

  フィジカル・スタンバイ・データベースは、Oracle リカバリ・メカニズムを使用して、スタンバイ・システムでアーカイブ REDO ログを適用することによってメンテナンスされます。リカバリ操作では、物理 ROWID を使用してブロックごとに変更が適用されます。REDO データの適用中は、データベースを読取り操作用または読取り / 書込み操作用にオープンすることはできません。

- **読取り専用オープン**

  フィジカル・スタンバイ・データベースは読取り専用操作用にオープンできるため、問合せを実行できます。読取り専用操作用にオープンしている間、スタンバイ・データベースは REDO ログの受信を継続できますが、ログのデータの適用は、データベースで管理リカバリ操作が再開されるまで遅延されます。

フィジカル・スタンバイ・データベースでは、管理リカバリ操作と読取り専用操作を同時に実行することはできませんが、これらの操作を切り替えることはできます。たとえば、フィジカル・スタンバイ・データベースで管理リカバリ操作を実行し、次にアプリケーションによるレポート用の読取り専用操作のためにそのデータベースをオープンした後、管理リカバリ操作に戻って未処理のアーカイブ REDO ログを適用できます。このサイクルを繰り返し、管理リカバリ操作と読取り専用操作を必要に応じて交互に実行できます。

いずれの場合も、フィジカル・スタンバイ・データベースはバックアップ操作の実行に使用できます。さらに、フィジカル・スタンバイ・データベースは、その時点で適用されない場合でも、REDO ログの受信を継続します。

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・データベースによって、堅牢で効率的な障害時リカバリと可用性の高いソリューションを実現できます。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・データベースを使用することで、予期しない障害が発生した場合でもデータが消失しないことが保証されます。フィジカル・スタンバイ・データベースでは、プライマリでサポートされるすべてのデータ型、および 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、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。最後に、REDO データが、スタンバイ・データベースへの適用時に検証されます。

- プライマリ・データベースのワークロードの低減

  Oracle Recovery Manager 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・データベースを読取り専用モードでオープンすると、レポート生成および問合せを実行することもできます。

- 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 適用テクノロジは、低レベルのリカバリ・メカニズムを使用して変更を適用します。このメカニズムは、SQL レベルのコード・レイヤーをすべてバイパスするため、変更の適用に関しては最も効率的です。このため、REDO 適用テクノロジは、データベース間で変更を伝播する非常に効率的なメカニズムです。

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 文を適用して更新されます。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、REDO ログのデータを 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 / 書込み操作用に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、表のタイプおよびデータ定義言語（DDL）操作やデータ操作言語（DML）操作のタイプに関していくつかの制限があります。サポートされないデータ型と表については、4-4 ページの「データ型または表のサポートの判別」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、高可用性およびデータ保護のメリットがあります。その他に、次のメリットもあります。

- スタンバイ・ハードウェア資源の効率的な使用

  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。Data Guard 構成内で保護されているデータベース・スキーマ以外に別のデータベース・スキーマをホスティングできるため、ユーザーは、これらのスキーマ上で通常の DDL 操作または DML 操作をいつでも実行できます。Data Guard で保護されているロジカル・スタンバイ表は、プライマリ・データベースとは異なる物理的なレイアウトで格納できるため、追加の索引やマテリアライズド・ビューを作成して、問合せのパフォーマンスを改善したり、特定のビジネス要件にあわせることができます。

- プライマリ・データベースのワークロードの低減

  ロジカル・スタンバイ・データベースは、その表をプライマリ・データベースから更新するとき、オープン状態のままにしておくことができるため、これらの表は読取りアクセスに同時に使用できます。このことによって、ロジカル・スタンバイ・データベースは、問合せ、要約およびレポート生成の各アクティビティを実行する際の優れたデータベースとなり、これらのタスクからプライマリ・データベースがオフロードされ、貴重な CPU と I/O のサイクルが節約されます。

Data Guard のユーザー・インタフェースの選択

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。

- コマンドライン・インタフェース :
  - SQL*Plus

    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。

    関連項目： 関連する文については、第 13 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。

    関連項目： 関連する初期化パラメータについては、11-4 ページの「Data Guard 構成内のインスタンスの初期化パラメータ」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

  Data Guard Broker のコマンドライン・インタフェースは、Oracle Data Guard Manager のグラフィカル・ユーザー・インタフェース（GUI）の代替手段です。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。

  関連項目： 『Oracle9*i* Data Guard Broker』

- Oracle Data Guard Manager

  Oracle Data Guard Manager 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 GUI です。

  関連項目： Data Guard Manager の GUI および Oracle9*i* Data Guard Manager Wizard については、『Oracle9*i* Data Guard Broker』および Oracle9*i* Data Guard Manager のオンライン・ヘルプを参照してください。

このマニュアルの説明と例では、Data Guard Broker のコマンドライン・インタフェースが使用されます。

Data Guard の動作要件

Data Guard 使用時の動作要件は次のとおりです。

- Data Guard 構成のすべてのシステムに、同じバージョンの Oracle Enterprise Edition がインストールされている必要があります。
- 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。
- 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、同じリリースの Oracle ソフトウェアを使用する必要があります。プライマリ・ロケーションとスタンバイ・ロケーションのオペレーティング・システムは同じであることが必要ですが、同じリリースである必要はありません。また、スタンバイ・データベースではプライマリ・データベースと異なるディレクトリ構造を使用できます。
- プライマリ・ロケーションとスタンバイ・ロケーションのハードウェアおよびオペレーティング・システムのアーキテクチャは同じであることが必要です。たとえば、32 ビットの Sun システムにプライマリ・データベースが存在している Data Guard 構成の場合、そのスタンバイ・データベースは 32 ビットの Sun システムに構成されている必要があります。同様に、64 ビットの HP-UX システムに存在するプライマリ・データベースは、64 ビットの HP-UX システム上にあるスタンバイ・データベースとともに構成される必要があり、32 ビットの Intel システムの Linux に存在するプライマリ・データベースは、32 ビットの Intel システムの Linux 上にあるスタンバイ・データベースとともに構成されている必要があり、その他の場合も同様です。
- プライマリ・データベースは、シングル・インスタンス・データベースまたは複数インスタンスの Real Application Clusters データベースです。スタンバイ・データベースは、シングル・インスタンス・データベースまたは複数インスタンスの Real Application Clusters データベースで、フィジカル・タイプとロジカル・タイプを組み合せることができます。
- ハードウェア（CPU の数、メモリー・サイズ、記憶域構成など）は、プライマリ・システムとスタンバイ・システムで異なっていても構いません。スタンバイ・システムがプライマリ・システムより小規模な場合は、スイッチオーバー操作またはフェイルオーバー操作後にスタンバイ・システムで実行可能な作業を制限する必要があります。さらに、スタンバイ・システムには、プライマリ・データベースからすべての REDO データを受信して適用するための十分なリソースが必要です。ロジカル・スタンバイ・データベースには、REDO データを SQL 文に変換し、その SQL をロジカル・スタンバイ・データベースで実行するための追加のリソースが必要です。
- プライマリ・データベースとスタンバイ・データベースには、それぞれ独自の制御ファイルが必要です。
- プライマリ・データベースとスタンバイ・データベースが同じシステム上にある場合、初期化パラメータを正しく調整する必要があります。

- 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操作を実行します。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。
- 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、Oracle9*i* データベース・ソフトウェア上で Oracle Data Guard を実行するようにアップグレードするための情報は、『Oracle9*i* データベース移行ガイド』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。

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・ファイルと REDO ログのパス名が決まるため、ディレクトリ構造は重要です。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・データベースの場合は、可能なかぎり、スタンバイ・ファイルのパス名を同じにしてください。同じパス名にしない場合は、ファイル名変換パラメータを設定する必要があります（表 2-1 を参照）。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（Standby1）。
- スタンバイ・データベースが離れたシステムにあるが、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（Standby2）。このオプションをお薦めします。
- スタンバイ・データベースが離れたシステムにあり、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（Standby3）。

図 2-1 可能なスタンバイ構成

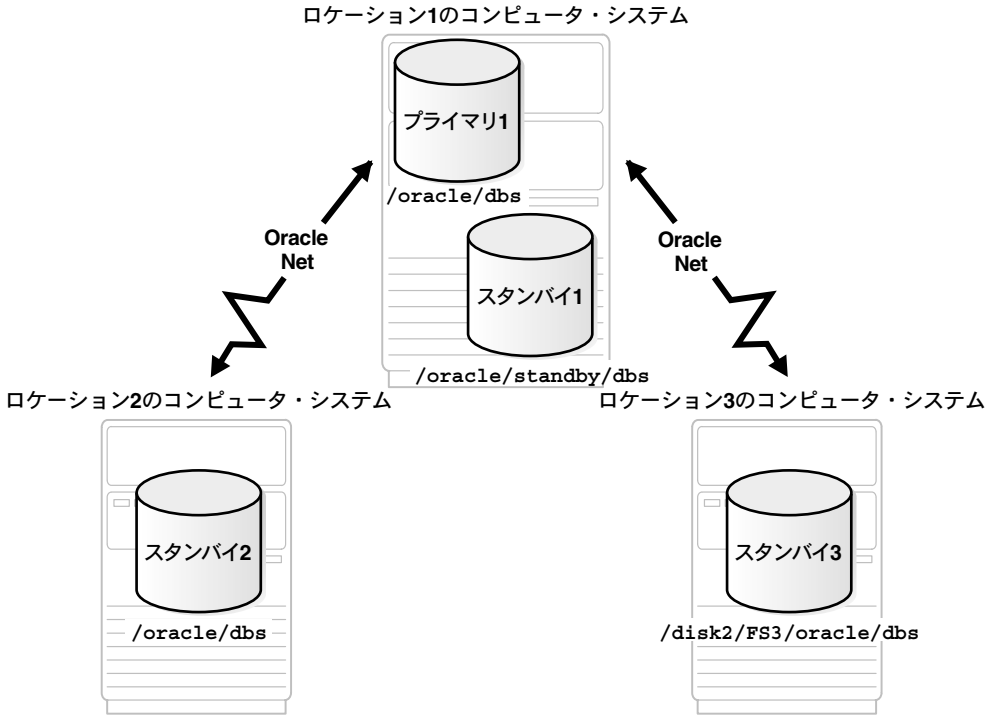


表 2-1 で、プライマリ・データベースとスタンバイ・データベースの可能な構成、およびそれぞれの結果的な注意事項について説明します。

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）	■ LOCK_NAME_SPACE 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・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・ファイル名と REDO ログ名を手動で変更する必要があります（付録 B を参照）。フィジカル・スタンバイ・データベースの場合は、データ・ファイル名が自動的に変更されるように、スタンバイ・データベースで DB_FILE_NAME_CONVERT および LOG_FILE_NAME_CONVERT 初期化パラメータを設定する方法もあります（6-7 ページの「データ・ファイルの管理」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・データベースの両方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と REDO ログ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。 ■ データベースに個別の物理メディアを使用すると、プライマリ・データの保護対策が行われます。
離れたシステム	プライマリ・システムと異なる	■ 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・ファイル名と REDO ログ名を手動で変更する必要があります（付録 B を参照）。フィジカル・スタンバイ・データベースの場合は、データ・ファイル名が自動的に変更されるように、スタンバイ・データベースで DB_FILE_NAME_CONVERT および LOG_FILE_NAME_CONVERT 初期化パラメータを設定する方法もあります（6-7 ページの「データ・ファイルの管理」を参照）。 ■ データベースに個別の物理メディアを使用すると、プライマリ・データの保護対策が行われます。

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
- フィジカル・スタンバイ・データベースの確認

この章の説明は、従来のテキストの初期化パラメータ・ファイル（PFILE）ではなく、サーバー・パラメータ・ファイル（SPFILE）に初期化パラメータを指定することを前提としています。サーバー・パラメータ・ファイルの作成方法と使用方法については、『Oracle9*i* データベース管理者ガイド』を参照してください。

> **関連項目：** Data Guard Manager のグラフィカル・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については、『Oracle9*i* Data Guard Broker』および Oracle Data Guard Manager のオンライン・ヘルプ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
3-2 ページ	強制ロギングの有効化
3-2 ページ	アーカイブの有効化とローカルのアーカイブ先の定義

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 操作がすべて完了するまで待機するためです。

アーカイブの有効化とローカルのアーカイブ先の定義

プライマリ・データベースが ARCHIVELOG モードであること、自動アーカイブが使用可能であること、およびローカルのアーカイブ先が定義済みであることを確認します。

次の SQL 文を使用して、ローカルのアーカイブ先を設定します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll
  2> MANDATORY' SCOPE=BOTH;
```

> **関連項目：** アーカイブの詳細は『Oracle9*i* データベース管理者ガイド』を、初期化パラメータの詳細は、第 11 章および『Oracle9*i* データベース・リファレンス』を参照してください。

フィジカル・スタンバイ・データベースの作成

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
3-4 ページ	プライマリ・データベースのデータ・ファイルの識別	プライマリ
3-4 ページ	プライマリ・データベースのコピーの作成	プライマリ
3-5 ページ	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
3-5 ページ	スタンバイ・データベースにコピーする初期化パラメータ・ファイルの準備	プライマリ
3-5 ページ	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
3-6 ページ	フィジカル・スタンバイ・データベースでの初期化パラメータの設定	スタンバイ
3-8 ページ	Windows サービスの作成	スタンバイ
3-8 ページ	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	プライマリとスタンバイ
3-9 ページ	スタンバイ・システムでの使用不能接続の検出の有効化	スタンバイ
3-9 ページ	Oracle Net サービス名の作成	プライマリとスタンバイ
3-9 ページ	スタンバイ・データベース用のサーバー・パラメータ・ファイルの作成	スタンバイ
3-9 ページ	フィジカル・スタンバイ・データベースの起動	スタンバイ
3-10 ページ	ログ適用サービスの開始	スタンバイ
3-10 ページ	フィジカル・スタンバイ・データベースへのアーカイブの有効化	プライマリ

プライマリ・データベースのデータ・ファイルの識別

プライマリ・データベースで、V$DATAFILE ビューを問い合せて、フィジカル・スタンバイ・データベースの作成に使用するファイルをリストします。次に例を示します。

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
--------------------------------------------------------------------------------
/disk1/oracle/oradata/payroll/system01.dbf
/disk1/oracle/oradata/payroll/undotbs01.dbf
/disk1/oracle/oradata/payroll/cwmlite01.dbf
.
.
.
```

プライマリ・データベースのコピーの作成

プライマリ・データベースで、次の手順を実行して、プライマリ・データベースのクローズ状態のバックアップ・コピーを作成します。

手順 1 プライマリ・データベースを停止する

次の SQL*Plus 文を発行して、プライマリ・データベースを停止します。

```
SQL> SHUTDOWN IMMEDIATE;
```

手順 2 データ・ファイルを一時的な位置にコピーする

「プライマリ・データベースのデータ・ファイルの識別」で識別したデータ・ファイルを、オペレーティング・システム・ユーティリティの copy コマンドを使用して一時的な位置にコピーします。次の例では、UNIX の cp コマンドを使用しています。

```
cp /disk1/oracle/oradata/payroll/system01.dbf
/disk1/oracle/oradata/payroll/standby/system01.dbf
```

データ・ファイルを一時的な位置にコピーすると、プライマリ・データベースの停止時間が短縮されます。

手順 3 プライマリ・データベースを再起動する

次の SQL*Plus 文を発行して、プライマリ・データベースを再起動します。

```
SQL> STARTUP;
```

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・データベースで、スタンバイ・データベース用の制御ファイルを作成します。次に例を示します。

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS
  2> '/disk1/oracle/oradata/payroll/standby/payroll2.ctl';
```

> **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。

作成したスタンバイ制御ファイルのファイル名は、プライマリ・データベースの現行の制御ファイルとは異なるファイル名にする必要があります。また、この制御ファイルは、バックアップ・データ・ファイルの最新タイムスタンプよりも後に作成する必要があります。

スタンバイ・データベースにコピーする初期化パラメータ・ファイルの準備

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・ファイルから、従来のテキストの初期化パラメータ・ファイルを作成します。従来の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='/disk1/oracle/dbs/initpayroll2.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「スタンバイ・データベース用のサーバー・パラメータ・ファイルの作成」で、サーバー・パラメータ・ファイルに変換し直します。

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・システムで、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。

- 「プライマリ・データベースのコピーの作成」で作成したバックアップ・データ・ファイル
- 「スタンバイ・データベース用の制御ファイルの作成」で作成したスタンバイ制御ファイル
- 「スタンバイ・データベースにコピーする初期化パラメータ・ファイルの準備」で作成した初期化パラメータ・ファイル

フィジカル・スタンバイ・データベースでの初期化パラメータの設定

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 3-1 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。変更されたパラメータ値は、太字で示されています。

例 3-1 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=PAYROLL
compatible=9.2.0.1.0
control_files='/disk1/oracle/oradata/payroll/standby/payroll2.ctl'
log_archive_start=TRUE
standby_archive_dest='/disk1/oracle/oradata/payroll/standby'
db_file_name_convert=('/disk1/oracle/oradata/payroll/',
'/disk1/oracle/oradata/payroll/standby/')
log_file_name_convert=('/disk1/oracle/oradata/payroll/',
'/disk1/oracle/oradata/payroll/standby/')
log_archive_format=log%d_%t_%s.arc
log_archive_dest_1=('LOCATION=/disk1/oracle/oradata/payroll/standby/')
standby_file_management=AUTO
remote_archive_enable=TRUE
instance_name=PAYROLL2
# The following parameter is required only if the primary and standby databases
# are located on the same system.
lock_name_space=PAYROLL2
.
.
.
```

次のリストは、例 3-1 で示したパラメータ設定に関する簡単な説明です。

- db_name - 変更なし。プライマリ・データベースと同じ名前です。
- compatible - 変更なし。プライマリ・データベースと同じ 9.2.0.1.0 です。
- control_files - スタンバイ制御ファイルのパス名とファイル名を指定します。
- log_archive_start - 変更なし。プライマリ・データベースの設定と同じ TRUE です。
- standby_archive_dest - プライマリ・データベースから受信するアーカイブ REDO ログの位置を指定します。

- db_file_name_convert - プライマリ・データベースのデータ・ファイルの位置を指定し、その後にスタンバイのデータ・ファイルの位置を指定します。このパラメータは、プライマリ・データベースのデータ・ファイルのディレクトリ・パスをスタンバイ・データ・ファイルのディレクトリ・パスに変換します。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。プライマリ・データベースのデータ・ファイルの位置については、3-4 ページの「プライマリ・データベースのデータ・ファイルの識別」を参照してください。
- log_file_name_convert - プライマリ・データベースのログの位置を指定し、その後にスタンバイのログの位置を指定します。このパラメータは、プライマリ・データベースのログのディレクトリ・パスをスタンバイ・ログのディレクトリ・パスに変換します。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。プライマリ・データベースのログの位置については、3-4 ページの「プライマリ・データベースのデータ・ファイルの識別」を参照してください。
- log_archive_format - DBID（%d）、スレッド（%t）および順序番号（%s）を使用して、アーカイブ REDO ログのフォーマットを指定します。
- log_archive_dest_1 - REDO ログがアーカイブされるスタンバイ・システムの位置を指定します（スイッチオーバーが発生し、このインスタンスがプライマリ・データベースになった場合、このパラメータは、オンライン REDO ログがアーカイブされる位置を指定します）。
- standby_file_management - AUTO に設定します。
- remote_archive_enable - TRUE に設定します。
- instance_name - このパラメータが定義されている場合は、プライマリ・データベースとスタンバイ・データベースが同じホストに存在するときに、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値を指定します。
- lock_name_space - スタンバイ・データベースのインスタンス名を指定します。

  このパラメータは、フィジカル・スタンバイ・データベースをプライマリ・データベースと同じシステムに作成するときに使用します。INSTANCE_NAME パラメータをプライマリ・データベースと異なる値に変更し、この LOCK_NAME_SPACE 初期化パラメータを、スタンバイ・データベースの INSTANCE_NAME 初期化パラメータに指定した値と同じ値に設定します。

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（background_dump_dest、core_dump_dest、user_dump_dest）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。さらに、スタンバイ・システムにまだ存在していないディレクトリがある場合は、そのディレクトリを作成する必要があります。

関連項目： Data Guard 環境の変更に使用できるすべての初期化パラメータの説明は、第 11 章を参照してください。

Windows サービスの作成

スタンバイ・システムが Windows システムで実行されている場合は、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 payroll2 -STARTMODE manual
```

関連項目： ORADIM ユーティリティの使用方法の詳細は、『Oracle9*i* Database for Windows 管理者ガイド』を参照してください。

プライマリ・データベースとスタンバイ・データベースに対するリスナーの構成

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。Data Guard Broker を使用して構成を管理する場合は、TCP/IP プロトコルを使用するようにリスナーを構成し、データベース・インスタンスの SID を使用して、各データベースのサービス情報を静的に登録する必要があります。

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
```

関連項目： 『Oracle9*i* Net Services 管理者ガイド』

スタンバイ・システムでの使用不能接続の検出の有効化

スタンバイ・システムの SQLNET.ORA パラメータ・ファイルで SQLNET.EXPIRE_TIME パラメータを 2 に設定して、使用不能接続の検出を使用可能にします。次に例を示します。

```
SQLNET.EXPIRE_TIME=2
```

Oracle Net サービス名の作成

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・サービス名はログ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のと同じプロトコル、ホスト・アドレス、ポートおよび SID 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。

> **関連項目：** 『Oracle9*i* Net Services 管理者ガイド』および『Oracle9*i* データベース管理者ガイド』

スタンバイ・データベース用のサーバー・パラメータ・ファイルの作成

アイドル状態のスタンバイ・データベースで、SQL の CREATE 文を使用して、「フィジカル・スタンバイ・データベースでの初期化パラメータの設定」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='initpayroll2.ora';
```

フィジカル・スタンバイ・データベースの起動

スタンバイ・データベースで、次の SQL 文を発行して、データベースをスタンバイ・モードで起動し、マウントします。

```
SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

ログ適用サービスの開始

スタンバイ・データベースで、ログ適用サービスを開始します。次に例を示します。

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

この例には DISCONNECT FROM SESSION オプションが指定されているため、ログ適用サービスはバックグラウンド・セッションで実行されます。

> **関連項目：** 6-3 ページ「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」

フィジカル・スタンバイ・データベースへのアーカイブの有効化

この項では、フィジカル・スタンバイ・データベースへのアーカイブを設定し、使用可能にするために、プライマリ・データベースで最低限実行する必要がある作業について説明します。

> **関連項目：** ログ転送サービスについては第 5 章を参照し、LOG_ARCHIVE_DEST_*n* 初期化パラメータで設定できる他の属性については、第 12 章を参照してください。

手順 1 アーカイブを定義する初期化パラメータを設定する

プライマリ・データベースからスタンバイ・サイトへのアーカイブ・ロギングを構成するには、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。

次は、スタンバイ・サイトに対するアーカイブ・ロギングを使用可能にするために必要な初期化パラメータの設定の例です。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=payroll2' SCOPE=BOTH;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE SCOPE=BOTH;
```

手順 2 リモート・アーカイブを開始する

リモート・スタンバイ・ロケーションへの REDO ログのアーカイブは、ログ・スイッチ後まで行われません。デフォルトでは、ログ・スイッチはオンライン REDO ログがいっぱいになったときに発生します。カレント REDO ログのアーカイブを即時に強制実行するには、プライマリ・データベースで SQL の ALTER SYSTEM 文を使用します。次に例を示します。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

> **関連項目：** 6-3 ページ「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」

フィジカル・スタンバイ・データベースの確認

フィジカル・スタンバイ・データベースを作成し、ログ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送信されていることの確認が必要な場合があります。

スタンバイ・データベースで受信された新しいアーカイブ REDO ログを調べるには、最初に、スタンバイ・データベースの既存のアーカイブ REDO ログを識別し、プライマリ・データベースのログをいくつかアーカイブして、スタンバイ・データベースを再度チェックします。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、既存のアーカイブ REDO ログを識別します。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-02 17:50:45 11-JUL-02 17:50:53
         9 11-JUL-02 17:50:53 11-JUL-02 17:50:58
        10 11-JUL-02 17:50:58 11-JUL-02 17:51:03

3 rows selected.
```

手順 2 カレント・ログをアーカイブする

プライマリ・データベースで、次の SQL 文を使用してカレント・ログをアーカイブします。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

手順 3 新規アーカイブ REDO ログの受信を確認する

スタンバイ・データベースで、V$ARCHIVED_LOG ビューを問い合せて、REDO ログが受信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-02 17:50:45 11-JUL-02 17:50:53
         9 11-JUL-02 17:50:53 11-JUL-02 17:50:58
        10 11-JUL-02 17:50:58 11-JUL-02 17:51:03
        11 11-JUL-02 17:51:03 11-JUL-02 18:34:11

4 rows selected.
```

ログがログ適用サービスで使用できるようになり、REDO データをスタンバイ・データベースに適用できます。

手順 4 新規アーカイブ REDO ログの適用を確認する

スタンバイ・データベースで、V$ARCHIVED_LOG ビューを問い合せて、アーカイブ REDO ログ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 YES

4 rows selected.
```

関連項目： ログ転送サービスとログ適用サービスの両方が正常に動作していることの確認方法は、5-32 ページの「REDO ログ・アーカイブ情報の監視」および 6-16 ページの「フィジカル・スタンバイ・データベースに関するログ適用サービスの監視」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してから、SQL 適用テクノロジを使用して、ログ適用サービスを構成し、スタンバイ・データベースをメンテナンスするまでの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
- ロジカル・スタンバイ・データベースの作成
- ロジカル・スタンバイ・データベースの確認

この章で説明する手順では、ロジカル・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：** Data Guard Manager のグラフィカル・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については、『Oracle9*i* Data Guard Broker』および Oracle Data Guard Manager のオンライン・ヘルプ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ためのプライマリ・データベースの準備

この章で説明するタスクを実行する前に、ロジカル・スタンバイ・データベース作成時にプライマリ・データベースで使用するユーザー・アカウントが、次のデータベース・ロールを含むように構成されていることを確認する必要があります。

- LOGSTDBY_ADMINISTRATOR ロール（ロジカル・スタンバイ機能を使用するために必要）
- SELECT_CATALOG_ROLE ロール（すべてのデータ・ディクショナリ・ビューに対する SELECT 権限を持つために必要）

また、この章の説明は、テキストの初期化パラメータ・ファイル（PFILE）ではなく、サーバー・パラメータ・ファイル（SPFILE）に初期化パラメータを指定することを前提としています。

> **関連項目：** サーバー・パラメータ・ファイルの作成方法と使用方法については、『Oracle9*i* データベース管理者ガイド』を参照してください。

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。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 4-1 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
4-3 ページ	強制ロギングの有効化
4-3 ページ	アーカイブの有効化とローカルのアーカイブ先の定義
4-3 ページ	LOG_PARALLELISM 初期化パラメータの確認
4-4 ページ	データ型または表のサポートの判別
4-7 ページ	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
4-9 ページ	サプリメンタル・ロギングが使用可能であることの確認
4-10 ページ	代替表領域の作成

> **注意：** 表 4-1 にリストされている手順は、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 つ以上のロジカル・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。

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 操作がすべて完了するまで待機するためです。

アーカイブの有効化とローカルのアーカイブ先の定義

プライマリ・データベースが ARCHIVELOG モードであること、自動アーカイブが使用可能であること、およびローカルのアーカイブ先が定義済みであることを確認します。

次の SQL 文を使用して、ローカルのアーカイブ先を設定します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll
  2> MANDATORY' SCOPE=BOTH;
```

> **関連項目：** アーカイブの詳細は『Oracle9*i* データベース管理者ガイド』を、初期化パラメータの詳細は第 11 章および『Oracle9*i* データベース・リファレンス』を参照してください。

LOG_PARALLELISM 初期化パラメータの確認

プライマリ・データベースで、SHOW PARAMETER *parameter_name* 文を使用して、LOG_PARALLELISM 初期化パラメータの現在の値を判別します。ロジカル・スタンバイ・データベースでは、この初期化パラメータをデフォルト値の 1 に設定する必要があります。LOG_PARALLELISM 初期化パラメータがすでに 1 に設定されている場合は、「データ型または表のサポートの判別」に進んでください。1 に設定されていない場合は、プライマリ・データベースで SQL の ALTER SYSTEM SET 文を発行して LOG_PARALLELISM=1 を設定し、値がサーバー・パラメータ・ファイルで確実に更新されるように SCOPE=SPFILE 句を指定します。次に例を示します。

```
SQL> ALTER SYSTEM SET LOG_PARALLELISM=1 SCOPE=SPFILE;
```

LOG_PARALLELISM 初期化パラメータを変更した場合は、初期化パラメータの新しい値が有効になるように、プライマリ・データベースを停止して再起動する必要があります。次に例を示します。

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

> **関連項目：** ALTER SET 文の詳細は『Oracle9*i* SQL リファレンス』を、初期化パラメータの詳細は第 11 章および『Oracle9*i* データベース・リファレンス』を参照してください。

データ型または表のサポートの判別

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メンテナンスできることを確認する必要があります。

次に、様々なデータベース・オブジェクトについて、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。

サポートされるデータ型

CHAR
NCHAR
VARCHAR2 および VARCHAR
NVARCHAR2
NUMBER
DATE
TIMESTAMP
TIMESTAMP WITH TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
RAW
CLOB
BLOB

サポートされないデータ型

NCLOB
LONG
LONG RAW
BFILE
ROWID
UROWID
ユーザー定義型
オブジェクト型 REF
VARRAY
ネストした表

サポートされない表、順序およびビュー

SYS スキーマ内のユーザー定義表と順序
サポートされないデータ型を使用した表
データ・セグメント圧縮を使用した表
索引構成表

プライマリ・データベースにサポートされないオブジェクトが含まれているかどうかを判断するには、DBA_LOGSTDBY_UNSUPPORTED ビューを問い合せます。たとえば、ロジカル・スタンバイ・データベースでサポートされないプライマリ・データベース表のスキーマと表名をリストするには、プライマリ・データベースで次の問合せを使用します。

```
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 ログをロジカル・スタンバイ・データベースに適用すると、ログ適用サービスによって、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・データベースの作成方法については、第 3 章を参照してください。

関連項目： DBA_LOGSTDBY_UNSUPPORTED ビューに関する情報は、第 14 章「ビュー」を参照してください。

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、SQL 文がプライマリ・データベースで実行された場合、次のリスト内の SQL 文以外はすべて、ロジカル・スタンバイ・データベースに適用されます。

```
ALTER DATABASE
ALTER SESSION
ALTER SNAPSHOT
ALTER SNAPSHOT LOG
ALTER SYSTEM SWITCH LOG
CREATE CONTROL FILE
CREATE DATABASE
CREATE DATABASE LINK
CREATE PFILE FROM SPFILE
CREATE SCHEMA AUTHORIZATION
CREATE SNAPSHOT
CREATE SNAPSHOT LOG
CREATE SPFILE FROM PFILE
CREATE TABLE AS SELECT FROM A CLUSTER TABLE
DROP DATABASE LINK
DROP SNAPSHOT
DROP SNAPSHOT LOG
EXPLAIN
LOCK TABLE
RENAME
SET CONSTRAINTS
SET ROLE
SET TRANSACTION
```

オブジェクトおよび操作のサポートの判別

メタデータを変更する PL/SQL プロシージャはスタンバイ・データベースに適用されないため、その結果はスタンバイ・データベースからは参照できません。この例の 1 つに DBMS_AQADM アドバンスト・キューイング・パッケージがあります。このパッケージはロジカル・スタンバイ・データベースではサポートされません。もう 1 つの例は DBMS_MVIEW_REFRESH です。このパッケージは、プライマリ・データベースで実行された場合、スタンバイ・データベースでの SQL 適用操作ではメンテナンスされません。

注意： DBMS_MVIEW_REFRESH ルーチンは、マテリアライズド・ビューをリフレッシュする必要があるスタンバイ・データベースで起動する必要があります。

唯一の例外は DBMS_JOB パッケージです。ジョブのメタデータはロジカル・スタンバイ・データベースに適用されます。ただし、ジョブは実行されません。

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・データベースの ROWID はプライマリ・データベースの ROWID と異なる可能性があるため、プライマリ・データベースで更新した行をスタンバイ・データベースの対応する行と照合するために、別の方法を使用する必要があります。次のいずれかを使用して、対応する行を照合します。

- 主キー
- 一意索引

SQL 適用操作で、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・データベースに主キーまたは一意索引を追加することをお薦めします。

ログ適用サービスで表の行を必ず一意に識別できるようにするには、「プライマリ・データベース内の一意識別子のない表の検索」および「無効化された RELY 主キー制約の追加」で説明する処理を実行してください。

プライマリ・データベース内の一意識別子のない表の検索

プライマリ・データベース内の主キーまたは一意索引のない表を識別するには、DBA_LOGSTDBY_NOT_UNIQUE ビューを問い合せます。次の問合せでは、SQL 適用操作で一意に識別できない可能性がある表のリストが表示されます。

```
SQL> SELECT OWNER, TABLE_NAME,BAD_COLUMN FROM DBA_LOGSTDBY_NOT_UNIQUE
  2> WHERE TABLE_NAME NOT IN (SELECT TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED);
```

DBA_LOGSTDBY_NOT_UNIQUE ビューに表示される表の一部は、サポートされている場合もあります。これは、**サプリメンタル・ロギング**（「サプリメンタル・ロギングが使用可能であることの確認」で使用可能にします）によって、REDO ログ内の行を一意に識別する情報が追加されるためです。主キーまたは一意索引が存在するかどうかが、サプリメンタル・ロギングに次のような影響を与える場合があります。

- 表に主キーまたは一意索引がある場合、サプリメンタル・ロギング時に REDO ログに追加される情報の量は最小限となります。
- 表に主キーまたは一意索引がない場合、サプリメンタル・ロギングは、各行に対するすべてのスカラー値を REDO ログに記録します。

BAD_COLUMN 列の値は Y または N です。次にその内容を説明します。

- Y

  表の列が、CLOB または BLOB などのバインドされないデータ型を使用して定義されていることを示します。SQL 適用操作はこれらの表のメンテナンスを試みますが、ユーザーは、バインドされていない列以外でアプリケーションが一意性を提供するように配慮する必要があります。表内の 2 つの行が、LOB 列以外で一致している場合、その表は適切にメンテナンスできません。

- N

  ロジカル・スタンバイ・データベース内の表をメンテナンスするための十分な列情報がその表にあることを示します。

無効化された RELY 主キー制約の追加

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。

> **関連項目：** ALTER TABLE 文の構文と使用情報については、『Oracle9*i* SQL リファレンス』を参照してください。

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、次の処理を実行します。

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 制約は、行が一意であると仮定するようシステムに指示します。行を一意に識別する、無効化された RELY 制約用に列を選択する場合は、注意が必要です。RELY 制約用に選択された列で行が一意に識別されない場合、ログ適用サービスは、REDO ログからロジカル・スタンバイ・データベースへのデータの適用に失敗します。

SQL 適用操作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一意に識別する列に索引を追加します。追加しない場合は、全表スキャンを実行する必要があります。

> **関連項目：** DBA_LOGSTDBY_NOT_UNIQUE ビューの詳細は第 14 章「ビュー」を、RELY 制約の作成方法の詳細は『Oracle9*i* SQL リファレンス』を、また、RELY 制約およびロジカル・スタンバイ・データベースでのパフォーマンスを向上させるための処置については、9-18 ページの「ロジカル・スタンバイ・データベースのチューニング」を参照してください。

サプリメンタル・ロギングが使用可能であることの確認

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・データベースでサプリメンタル・ロギングを使用可能にする必要があります。Oracle では変更された列のみがログに記録されるため、変更された行を一意に識別するのに十分でない場合があります。このため、REDO ログには追加の（補足）情報を記録する必要があります。REDO ログに追加される補足情報によって、ログ適用サービスは、ロジカル・スタンバイ・データベースの表を正しく識別し、メンテナンスできます。

サプリメンタル・ロギングがプライマリ・データベースで使用可能かどうかを判別するには、V$DATABASE 固定ビューを問い合せます。次に例を示します。

```
SQL> SELECT SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI FROM V$DATABASE;
SUP SUP
--- ---
NO NO
```

この例で、NO という値は、サプリメンタル・ロギングがプライマリ・データベースで使用可能でないことを示しています。

サプリメンタル・ロギングが使用可能である場合は、「代替表領域の作成」に進んでください。サプリメンタル・ロギングが使用可能でない場合は、次の項の手順を実行してサプリメンタル・ロギングを使用可能にします。

サプリメンタル・ロギングの有効化

プライマリ・データベースで、次の文を発行して、主キーと一意索引の情報をアーカイブ REDO ログに追加します。

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE INDEX) COLUMNS;
```

この SQL 文によって、プライマリ・データベースで変更された行を一意に識別する情報が追加されるため、ログ適用サービスは、スタンバイ・データベースの同じ行を正しく識別し、メンテナンスできます。

新規 REDO ログへの切替え

プライマリ・データベースで、次の文を発行して新規 REDO ログに切り替えます。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

新規ログ・ファイルに切り替えることによって、サプリメンタルと非サプリメンタルの両方のログ・データが REDO ログに含まれないようにします。ロジカル・スタンバイ・データベースは、サプリメンタルと非サプリメンタルの両方のログ・データが含まれた REDO ログを使用できません。

サプリメンタル・ロギングが使用可能であることの確認

プライマリ・データベースで、前述の問合せと同じ問合せを発行して、サプリメンタル・ロギングが使用可能であることを確認します。次に例を示します。

```
SQL> SELECT SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI FROM V$DATABASE;
SUP SUP
--- ---
YES YES
```

この例で、YES という値は、サプリメンタル・ロギングがプライマリ・データベースで使用可能であることを示しています。主キー（SUPPLEMENTAL_LOG_DATA_PK）または一意索引（SUPPLEMENTAL_LOG_DATA_UI）が設定されている表の場合はすべて、更新操作の実行時に、主キーと一意索引のすべての列が REDO ログに記録されます。

注意： フィジカル・スタンバイ・データベースがすでに含まれている Data Guard 構成内のプライマリ・データベースでサプリメンタル・ロギングを使用可能にする場合は、スイッチオーバー操作が正しく実行されるように、各フィジカル・スタンバイ・データベースで ALTER DATABASE ADD SUPPLEMENTAL LOG DATA 文を発行する必要があります。

関連項目： V$DATABASE ビューの詳細は第 14 章「ビュー」を、ALTER DATABASE ADD SUPPLEMENTAL LOG DATA 文の詳細は『Oracle9*i* SQL リファレンス』を参照してください。

代替表領域の作成

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操作を実行する場合は、プライマリ・データベースに代替表領域を作成し、その別の表領域にロジカル・スタンバイのシステム表を移す必要があります。

ロジカル・スタンバイ・データベースでは、SYS スキーマと SYSTEM スキーマに定義されている多数の表が使用されます。これらの表は、SYSTEM 表領域内にデフォルトで作成されます。これらの表のうちいくつかは急速に大きくなる場合があります。事前に代替表領域を準備し、ロジカル・スタンバイのシステム表を別の表領域に移しておくことによって、SYSTEM 表領域全体がこれらの表でいっぱいになるのを防止できます。新しい表領域への表の移動は、「ロジカル・スタンバイ・データベースの作成」で説明するロジカル・スタンバイ作成プロセスでその表にデータが移入される前に行ってください。

ロジカル・スタンバイ表用に新規表領域を作成するには、SQL の CREATE TABLESPACE 文を発行します。次に、DBMS_LOGMNR_D.SET_TABLESPACE プロシージャを使用して、プライマリ・データベース上の新規表領域に表を移します。たとえば、次の文では、logmnrts という名前の新規表領域が作成され、その表領域に LogMiner 表が移されます。

```
SQL> CREATE TABLESPACE logmnrts DATAFILE '/disk1/oracle/dbs/logmnrts.dbf'
  2> SIZE 25M AUTOEXTEND ON MAXSIZE UNLIMITED;
SQL> EXECUTE DBMS_LOGMNR_D.SET_TABLESPACE('logmnrts');
```

代替表領域の作成では、完了までに数分かかる場合があります。

プライマリ・データベースに代替表領域を作成することによって、プライマリ・データベースから作成されたすべてのスタンバイ・データベースに新規表領域が含まれます。後でプライマリ・データベースがスタンバイ・データベースになった場合、これらの表領域は正しく設定されます。

> **関連項目：** CREATE TABLESPACE 文の詳細は『Oracle9*i* SQL リファレンス』を、DBMS_LOGMNR_D 提供パッケージの詳細は『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

ロジカル・スタンバイ・データベースの作成

この項では、ロジカル・スタンバイ・データベースの設定と作成で実行する必要のあ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、および各手順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 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
4-13 ページ	プライマリ・データベースのデータ・ファイルとログ・ファイルの識別	プライマリ
4-13 ページ	プライマリ・データベースのコピーの作成	プライマリ
4-15 ページ	スタンバイ・システムにコピーする初期化パラメータ・ファイルの準備	プライマリ
4-16 ページ	プライマリ・データベースの位置からスタンバイの位置へのファイルのコピー	プライマリ
4-16 ページ	ロジカル・スタンバイ・データベースでの初期化パラメータの設定	スタンバイ
4-18 ページ	Windows サービスの作成	スタンバイ
4-18 ページ	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対するリスナーの構成	プライマリとスタンバイ
4-18 ページ	スタンバイ・システムでの使用不能接続の検出の有効化	スタンバイ
4-18 ページ	Oracle Net サービス名の作成	プライマリとスタンバイ
4-19 ページ	ロジカル・スタンバイ・データベースの起動とマウント	スタンバイ
4-19 ページ	ロジカル・スタンバイ・データベースでのデータ・ファイル名の変更	スタンバイ
4-19 ページ	ロジカル・スタンバイ・データベースでのオンライン REDO ログ名の変更	スタンバイ
4-20 ページ	データベース・ガードをオンに変更	スタンバイ
4-20 ページ	ロジカル・スタンバイ・データベースのデータベース名のリセット	スタンバイ
4-21 ページ	パラメータ・ファイルでのデータベース名の変更	スタンバイ
4-22 ページ	ロジカル・スタンバイ・データベースに対する新規一時ファイルの作成	スタンバイ
4-23 ページ	アーカイブ REDO ログの登録と SQL 適用操作の開始	スタンバイ
4-23 ページ	ロジカル・スタンバイ・データベースへのアーカイブの有効化	プライマリ

注意： 作成する各ロジカル・スタンバイ・データベースに対して、表 4-2 の手順を実行してください。

プライマリ・データベースのデータ・ファイルとログ・ファイルの識別

プライマリ・データベースで、V$DATAFILE ビューを問い合せて、ロジカル・スタンバイ・データベースの作成に使用するファイルをリストします。次に例を示します。

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
--------------------------------------------------------------------------------
/disk1/oracle/oradata/payroll/system01.dbf
/disk1/oracle/oradata/payroll/undotbs01.dbf
/disk1/oracle/oradata/payroll/cwmlite01.dbf
.
.
.
```

プライマリ・データベースで、V$LOGFILE ビューを問い合せて、プライマリ・データベースのログをリストします （この情報は後の手順で使用します）。次に例を示します。

```
SQL> SELECT GROUP#,TYPE,MEMBER FROM V$LOGFILE;
GROUP#            TYPE       MEMBER
--------------------------------------------------------------------------------
1          ONLINE           /disk1/oracle/oradata/payroll/redo01.log
2          ONLINE           /disk1/oracle/oradata/payroll/redo02.log
3          ONLINE           /disk1/oracle/oradata/payroll/redo03.log
.
.
.
```

プライマリ・データベースのコピーの作成

次の手順を実行して、プライマリ・データベースのクローズ状態のバックアップ・コピーを作成します。

手順 1 プライマリ・データベースを停止する

プライマリ・データベースで、SQL*Plus 文を使用してプライマリ・データベースを停止します。

```
SQL> SHUTDOWN IMMEDIATE;
```

手順 2 データ・ファイルを一時的な位置にコピーする

プライマリ・データベースで、「プライマリ・データベースのデータ・ファイルとログ・ファイルの識別」で識別したデータ・ファイルを、オペレーティング・システム・ユーティリティの copy コマンドを使用して一時的な位置にコピーします。次の例では、UNIX の cp コマンドを使用しています。

```
cp /disk1/oracle/oradata/payroll/system01.dbf
/disk1/oracle/oradata/payroll/standby/system01.dbf

cp /disk1/oracle/oradata/payroll/undotbs01.dbf
/disk1/oracle/oradata/payroll/standby/undotbs01.dbf

cp /disk1/oracle/oradata/payroll/cwmlite01.dbf
/disk1/oracle/oradata/payroll/standby/cwmlite01.dbf
.
.
.
```

データ・ファイルを一時的な位置にコピーすると、プライマリ・データベースの停止時間が短縮されます。

手順 3 プライマリ・データベースを再起動する

プライマリ・データベースで、次の SQL*Plus コマンドを使用してプライマリ・データベースを再起動し、マウントします。

```
SQL> STARTUP MOUNT;
```

手順 4 スタンバイ・データベースの制御ファイルのバックアップ・コピーを作成する

プライマリ・データベースで、スタンバイ・データベースの制御ファイルのバックアップ・コピーを作成します。

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO
  2> '/disk1/oracle/oradata/payroll/standby/payroll3.ctl';
```

手順 5 プライマリ・データベースで制限付きセッション・モードを使用可能にする

プライマリ・データベースで、制限付きセッション・モードを使用可能にして、ユーザーまたはアプリケーションが DML 操作または DDL 操作を実行する可能性を低減します。

注意： 制限付きセッション・モードが手順 7 で無効にされるまで、DML 操作または DDL 操作は禁止してください。

次の文で、制限付きセッション・モードを使用可能にします。

```
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
```

手順 6 LogMiner ディクショナリを作成する

ロジカル・スタンバイ・データベースを作成するには、ロジカル・スタンバイ・データベースのディクショナリを手動で作成する必要があります。プライマリ・データベースで、次の文を発行して LogMiner ディクショナリを作成します。

```
SQL> ALTER DATABASE OPEN;
SQL> EXECUTE DBMS_LOGSTDBY.BUILD;
```

手順 7 プライマリ・データベースで制限付きセッション・モードを使用不可能にする

プライマリ・データベースで、次の SQL 文を使用して制限付きセッション・モードを使用不可能にします。

```
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

手順 8 最新のアーカイブ REDO ログを識別する

ロジカル・スタンバイ・データベース作成の開始点を取得するために、V$ARCHIVED_LOG ビューを問い合せて、最新のアーカイブ REDO ログを識別し、後続の作成プロセスで使用するためにその名前を記録します。次の問合せには、新規ディクショナリの名前を検索する DICTIONARY_BEGIN 句、およびローカルの宛先に対してのみ情報を示す STANDBY_DEST 句が含まれています （STANDBY_DEST 句を指定しない場合、情報にはローカルとスタンバイの両方の宛先に対する出力が含まれます)。次に例を示します。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
SQL> SELECT NAME FROM V$ARCHIVED_LOG
  2> WHERE (SEQUENCE#=(SELECT MAX(SEQUENCE#) FROM V$ARCHIVED_LOG
  3> WHERE DICTIONARY_BEGIN = 'YES' AND STANDBY_DEST= 'NO'));

NAME
----------------------------------------
/disk1/oracle/oradata/payroll/arc0004.001
```

後続の作成プロセスで使用するためにアーカイブ REDO ログの名前を必ず記録してください。

スタンバイ・システムにコピーする初期化パラメータ・ファイルの準備

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・ファイルから、テキストの初期化パラメータ・ファイルを作成します。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。次の例は、SPFILE からテキストの初期化パラメータ・ファイルを作成するためにプライマリ・データベースで使用する文です。

```
SQL> CREATE PFILE='/disk1/oracle/dbs/initpayroll3.ora' FROM SPFILE;
```

このファイルを、ロジカル・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が含まれるように変更します。その後、「パラメータ・ファイルでのデータベース名の変更」で、サーバー・パラメータ・ファイルに変換し直します。

プライマリ・データベースの位置からスタンバイの位置へのファイルのコピー

プライマリ・データベースで、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・データベース・サイトからスタンバイ・サイトにコピーします。

- 「プライマリ・データベースのコピーの作成」で作成したバックアップ・データ・ファイルと制御ファイル
- 「プライマリ・データベースのコピーの作成」の手順 8 で識別された最新のアーカイブ REDO ログ
- 「スタンバイ・システムにコピーする初期化パラメータ・ファイルの準備」で作成したデータベース初期化パラメータ・ファイル

ロジカル・スタンバイ・データベースでの初期化パラメータの設定

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、ロジカル・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 4-1 は、スタンバイのテキスト初期化パラメータ・ファイルの一部です。この中の値は、ロジカル・スタンバイ・データベース用に変更されています。変更されたパラメータ値は、太字で示されています。

例 4-1 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

```
.
.
.
db_name=PAYROLL
compatible=9.2.0.1.0
control_files='/disk1/oracle/oradata/payroll/standby/payroll3.ctl'
log_archive_start=TRUE
standby_archive_dest='/disk1/oracle/oradata/payroll/standby'
log_archive_format=log%d_%t_%s.arc
log_archive_dest_1='LOCATION=/disk1/oracle/oradata/payroll/arch/'
log_parallelism=1
parallel_max_servers=9
instance_name=PAYROLL3
# The following parameter is required only if the primary and standby databases
# are located on the same system.
lock_name_space=PAYROLL3
.
.
.
```

次のリストは、例 4-1 で示したパラメータの設定に関する簡単な説明です。

- db_name - 変更なし。プライマリ・データベースと同じ名前です。
- compatible - 変更なし。プライマリ・データベースと同じ 9.2.0.1.0 です。
- control_files - スタンバイ制御ファイルのパス名とファイル名を指定します。
- log_archive_start - 変更なし。プライマリ・データベースの設定と同じ TRUE です。
- standby_archive_dest - プライマリ・データベースから受信するアーカイブ REDO ログの位置を指定します。
- log_archive_format - DBID（%d）、スレッド（%t）および順序番号（%s）を使用して、アーカイブ REDO ログのフォーマットを指定します。
- log_archive_dest_1 - REDO ログがアーカイブされる位置を指定します。
- log_parallelism - 変更なし。プライマリ・データベースと同じ値です。
- parallel_max_servers - 9 に設定します。
- instance_name - このパラメータが定義されている場合は、プライマリ・データベースとスタンバイ・データベースが同じホストに存在するときに、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値を指定します。
- lock_name_space - スタンバイ・データベースのインスタンス名を指定します。

  このパラメータは、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システムに作成するときに使用します。INSTANCE_NAME パラメータをプライマリ・データベースと異なる値に変更し、この LOCK_NAME_SPACE 初期化パラメータを、スタンバイ・データベースの INSTANCE_NAME 初期化パラメータに指定した値と同じ値に設定します。

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（background_dump_dest、core_dump_dest、user_dump_dest）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。さらに、スタンバイ・システムにまだ存在していないディレクトリがある場合は、そのディレクトリを作成する必要があります。

関連項目： Data Guard 環境の変更に使用できるすべての初期化パラメータの説明は、第 11 章を参照してください。

Windows サービスの作成

スタンバイ・システムが Windows システムで実行されている場合は、Windows サービスを作成する必要があります。ORADIM ユーティリティを実行して、Windows サービスとパスワード・ファイルの両方を作成します。次に例を示します。

```
WINNT> oradim -NEW -SID payroll3 -STARTMODE auto
```

関連項目： ORADIM ユーティリティの使用方法の詳細は、『Oracle9*i* Database for Windows 管理者ガイド』を参照してください。

プライマリ・データベースとスタンバイ・データベースの両方に対するリスナーの構成

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。Data Guard Broker を使用して構成を管理する場合は、TCP/IP プロトコルを使用するようにリスナーを構成し、データベース・インスタンスの SID を使用して、各データベースのサービス情報を静的に登録する必要があります。

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
```

関連項目：『Oracle9*i* Net Services 管理者ガイド』

スタンバイ・システムでの使用不能接続の検出の有効化

スタンバイ・システムの SQLNET.ORA パラメータ・ファイルで SQLNET.EXPIRE_TIME パラメータを 2 に設定して、使用不能接続の検出を使用可能にします。次に例を示します。

```
SQLNET.EXPIRE_TIME=2
```

Oracle Net サービス名の作成

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・サービス名はログ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のと同じプロトコル、ホスト・アドレス、ポートおよび SID 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。

関連項目：『Oracle9*i* Net Services 管理者ガイド』

ロジカル・スタンバイ・データベースの起動とマウント

STARTUP 文を使用して、ロジカル・スタンバイ・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> STARTUP MOUNT PFILE=initpayroll3.ora;
```

ロジカル・スタンバイ・データベースでのデータ・ファイル名の変更

ロジカル・スタンバイ・データベースで、「プライマリ・データベースのデータ・ファイルとログ・ファイルの識別」で識別され、プライマリ・データベースからコピーされたすべてのデータ・ファイルの名前を変更します。次に例を示します。

```
SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/system01.dbf'
  2> TO '/disk1/oracle/oradata/payroll/standby/system01.dbf';

SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/undotbs01.dbf'
  2> TO '/disk1/oracle/oradata/payroll/standby/undotbs01.dbf'

SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/cwmlite01.dbf'
  2> TO '/disk1/oracle/oradata/payroll/standby/cwmlite01.dbf'
.
.
.
```

これらの文は、制御ファイルにあるデータ・ファイル名を指定します。実際のデータ・ファイルの名前は変更しません。

スタンバイ・データベースで、V$DATAFILE ビューの NAME 列を問い合せて、すべてのデータ・ファイルの位置が正しいことを確認します（これは、「プライマリ・データベースのデータ・ファイルとログ・ファイルの識別」で示した問合せと同じです）。

ロジカル・スタンバイ・データベースでのオンライン REDO ログ名の変更

オンライン REDO ログはプライマリ・データベースからコピーされていませんが、制御ファイル内のポインタが正しい位置を示すために更新されるように、名前をすべて変更する必要があります。オンライン REDO ログの位置と名前は、「プライマリ・データベースのデータ・ファイルとログ・ファイルの識別」で識別されたものです。次に例を示します。

```
SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/redo01.log'
  2> TO '/disk1/oracle/oradata/payroll/standby/redo01.log';
```

スタンバイ・データベースで、V$DATAFILE ビューの NAME 列を問い合せて、すべてのログの位置が正しいことを確認します（これは、「プライマリ・データベースのデータ・ファイルとログ・ファイルの識別」で示した問合せと同じです）。

データベース・ガードをオンに変更

ユーザーがロジカル・スタンバイ・データベースのオブジェクトを更新しないように、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・ガードをオンにします。

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

ロジカル・スタンバイ・データベースのデータベース名のリセット

Oracle の DBNEWID（nid）ユーティリティを実行して、ロジカル・スタンバイ・データベースのデータベース名を変更します。名前の変更によって、このプライマリ・データベースのコピーと元のプライマリ・データベース間での相互作用を防止します。

DBNEWID（nid）ユーティリティを実行する前に、データベースを停止し、その後再起動してマウントする必要があります。次に例を示します。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT PFILE=initpayroll3.ora;
```

ここで、スタンバイ・データベースで Oracle の DBNEWID ユーティリティを実行し、データベース名を変更してデータベースを停止します。

```
nid TARGET=SYS/password@PAYROLL3 DBNAME=PAYROLL3
Connected to database PAYROLL (DBID=1456557175)

Control Files in database:
    /disk1/oracle/oradata/payroll/standby/stdby.ctl
Change database ID and database name PAYROLL to PAYROLL3? (Y/[N]) => y

Proceeding with operation
Changing database ID from 1456557175 to 416458362
Changing database name from PAYROLL to PAYROLL3
    Control File /disk1/oracle/oradata/payroll/standby/payroll3.ctl - modified
    Datafile /disk1/oracle/oradata/payroll/standby/system01.dbf - dbid changed,
wrote new name
    Datafile /disk1/oracle/oradata/payroll/standby/undotbs01.dbf -dbid changed,
wrote new name
    .
    .
    .
    Control File /disk1/oracle/oradata/payroll/standby/payroll3.ctl-dbid changed,
wrote new name

Database name changed to PAYROLL3.
Modify parameter file and generate a new password file before restarting.
Database ID for database PAYROLL3 change to 416458362.
All previous backups and archived redo logs for this database are unusable.
```

```
Shut down database and open with RESETLOGS option.
Successfully changed database name and ID.
DBNEWID - Completed successfully.
```

認証にパスワード・ファイルを使用する場合は、Oracle の DBNEWID（nid）ユーティリティを実行した後で、パスワード・ファイルを再作成する必要があります。

パラメータ・ファイルでのデータベース名の変更

DBNEWID ユーティリティの出力に、初期化パラメータ・ファイルの更新が必要であることが示されます。次の手順で、このタスクの実行方法を説明します。

手順 1 DB_NAME パラメータを変更する

テキスト初期化パラメータ・ファイルの DB_NAME 初期化パラメータを、新規名と一致するように設定します。

```
.
.
.
db_name=PAYROLL3
.
.
.
```

手順 2 スタンバイ・データベースを停止する

スタンバイ・データベースで、次の SQL 文を発行します。

```
SQL> SHUTDOWN IMMEDIATE;
```

手順 3 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

スタンバイ・データベースのアイドル状態のインスタンスに接続し、「ロジカル・スタンバイ・データベースでの初期化パラメータの設定」と、この項の手順 1 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=initpayroll3.ora;
```

手順 4 ロジカル・スタンバイ・データベースを再起動する

次のように、ユーザー・アクセス用にデータベースを起動してオープンします。

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

ロジカル・スタンバイ・データベースに対する新規一時ファイルの作成

プライマリ・データベースでのクローズ状態のバックアップ操作の一部に含まれていた一時ファイルは、ロジカル・スタンバイ・データベースでは使用できません（一時ファイルはプライマリ・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はありません）。

不要な一時ファイルを識別して削除するには、ロジカル・スタンバイ・データベースで次の手順を実行します。

手順 1 カレント一時ファイルを識別する

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、スタンバイ・データベースのカレント一時ファイルを識別します。

```
SQL> SELECT * FROM V$TEMPFILE;
no rows selected
```

例に示されているように、この問合せの結果が「no rows selected」の場合は、手順 2 をスキップし、手順 3 に進んでください。

手順 2 スタンバイ・データベースからカレント一時ファイルを削除する

スタンバイ・データベースからカレント一時ファイルを削除します。次に例を示します。

```
SQL> ALTER DATABASE TEMPFILE 'tempfilename' DROP;
```

手順 3 新規一時ファイルを追加する

ロジカル・スタンバイ・データベースで、次のタスクを実行して、新規一時ファイルを表領域に追加します。

1. 一時ファイルを格納する表領域を識別します。次に例を示します。

```
SQL> SELECT TABLESPACE_NAME FROM DBA_TABLESPACES WHERE
  2> CONTENTS ='TEMPORARY';

TABLESPACE_NAME
--------------------------------
TEMP
```

2. 新規一時ファイルを追加します。次に例を示します。

```
SQL> ALTER TABLESPACE TEMP ADD TEMPFILE
  2> '/disk1/oracle/oradata/payroll/standby/temp01.dbf'
  3> SIZE 40M REUSE;
```

アーカイブ REDO ログの登録と SQL 適用操作の開始

最新のアーカイブ REDO ログを登録し、REDO ログからスタンバイ・データベースへのデータの適用を開始するには、次の手順を実行します。

手順 1 ログ適用サービスに最新アーカイブ REDO ログを登録する

「プライマリ・データベースのコピーの作成」の手順 8 で識別されたアーカイブ REDO ログを登録します。次の例は、ロジカル・スタンバイ・サイトにコピーした最新のアーカイブ REDO ログのファイル名と位置を指定します。

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
  2> '/disk1/oracle/oradata/payroll/standby/arc0004.001';
```

手順 2 ロジカル・スタンバイ・データベースに対する REDO ログの適用を開始する

次の SQL 文を指定して、ロジカル・スタンバイ・データベースに対する REDO ログの適用を開始します。次に例を示します。

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

注意： INITIAL キーワードを指定するのは、REDO ログからスタンバイ・データベースへのデータ適用を初めて開始するときのみです。たとえば次の文は、その後の SQL 適用操作の停止および開始方法を示しています。

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

ロジカル・スタンバイ・データベースへのアーカイブの有効化

この項では、ロジカル・スタンバイ・データベースでのアーカイブを設定し、使用可能にするために、プライマリ・データベースで最低限実行する必要がある作業について説明します。

関連項目： ログ転送サービスについては第 5 章を参照し、LOG_ARCHIVE_DEST_*n* 初期化パラメータで設定できる他の属性については、第 12 章を参照してください。

手順 1 アーカイブを定義する初期化パラメータを設定する

プライマリ・データベースからスタンバイ・サイトへのアーカイブ・ロギングを構成するには、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。

次は、スタンバイ・サイトでのアーカイブ・ロギングを使用可能にするために必要な初期化パラメータの設定の例です。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_3='SERVICE=payroll3' SCOPE=BOTH;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_3=ENABLE SCOPE=BOTH;
```

手順 2 リモート・アーカイブを開始する

リモート・スタンバイ・ロケーションへの REDO ログのアーカイブは、ログ・スイッチ後まで行われません。デフォルトでは、ログ・スイッチはオンライン REDO ログがいっぱいになったときに発生します。カレント REDO ログのアーカイブを即時に強制実行するには、プライマリ・データベースで SQL の ALTER SYSTEM 文を使用します。次に例を示します。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

> **関連項目：** ロジカル・スタンバイ・データベースをスイッチオーバー操作のターゲットにする場合は、第 7 章を参照してください。この章では、スイッチオーバー操作時に使用するプライマリ・データベースへのデータベース・リンクの定義方法を説明しています。

ロジカル・スタンバイ・データベースの確認

ロジカル・スタンバイ・データベースを作成し、ログ転送サービスとログ適用サービスを設定した後は、REDO ログがプライマリ・データベースから転送され、スタンバイ・データベースに適用されていることを確認できます。これをチェックするには、次の手順を実行します。

手順 1 REDO ログが登録されたことを確認する

REDO ログがロジカル・スタンバイ・システムに登録されたことを確認するには、ロジカル・スタンバイ・データベースに接続し、DBA_LOGSTDBY_LOG ビューを問い合せます。次に例を示します。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT  = 'DD-MON-YY HH24:MI:SS';
Session altered.

SQL> SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME, DICT_BEGIN, DICT_END
  2> FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

 SEQUENCE# FIRST_TIME         NEXT_TIME          DIC DIC
---------- ------------------ ------------------ --- ---
        24 23-JUL-02 18:19:05 23-JUL-02 18:19:48 YES YES
        25 23-JUL-02 18:19:48 23-JUL-02 18:19:51 NO  NO
        26 23-JUL-02 18:19:51 23-JUL-02 18:19:54 NO  NO
        27 23-JUL-02 18:19:54 23-JUL-02 18:19:59 NO  NO
        28 23-JUL-02 18:19:59 23-JUL-02 18:20:03 NO  NO
        29 23-JUL-02 18:20:03 23-JUL-02 18:20:13 NO  NO
```

```
        30 23-JUL-02 18:20:13 23-JUL-02 18:20:18 NO  NO
        31 23-JUL-02 18:20:18 23-JUL-02 18:20:21 NO  NO

8 rows selected.
```

手順 2 一部の REDO ログをアーカイブする

プライマリ・データベースに接続し、一部の REDO ログをアーカイブします。次に例を示します。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
System altered.

SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
System altered.
```

手順 3 DBA_LOGSTDBY_LOG ビューを再度問い合せる

ロジカル・スタンバイ・データベースに接続し、DBA_LOGSTDBY_LOG ビューを再度問い合せます。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT  = 'DD-MON-YY HH24:MI:SS';
Session altered.

SQL> SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME, DICT_BEGIN, DICT_END
  2  FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

 SEQUENCE# FIRST_TIME         NEXT_TIME          DIC DIC
---------- ------------------ ------------------ --- ---
        24 23-JUL-02 18:19:05 23-JUL-02 18:19:48 YES YES
        25 23-JUL-02 18:19:48 23-JUL-02 18:19:51 NO  NO
        26 23-JUL-02 18:19:51 23-JUL-02 18:19:54 NO  NO
        27 23-JUL-02 18:19:54 23-JUL-02 18:19:59 NO  NO
        28 23-JUL-02 18:19:59 23-JUL-02 18:20:03 NO  NO
        29 23-JUL-02 18:20:03 23-JUL-02 18:20:13 NO  NO
        30 23-JUL-02 18:20:13 23-JUL-02 18:20:18 NO  NO
        31 23-JUL-02 18:20:18 23-JUL-02 18:20:21 NO  NO
        32 23-JUL-02 18:20:21 23-JUL-02 18:32:11 NO  NO
        33 23-JUL-02 18:32:11 23-JUL-02 18:32:19 NO  NO

10 rows selected.
```

スタンバイ・データベースでファイルをチェックし、REDO ログをいくつかアーカイブして、再度スタンバイ・データベースをチェックすることによって、新しい REDO ログが登録されたことを確認できます。これらのログは、ログ適用サービスがログの適用を開始するために使用できます。

手順 4 REDO ログのデータが正しく適用されていることを確認する

ロジカル・スタンバイ・データベースで、DBA_LOGSTDBY_STATS ビューを問い合せて、REDO データが正しく適用されていることを確認します。次に例を示します。

```
SQL> COLUMN NAME FORMAT A30
SQL> COLUMN VALUE FORMAT A30
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME = 'coordinator state';

NAME                           VALUE
------------------------------ ------------------------------
coordinator state              INITIALIZING
```

例では、DBA_LOGSTDBY_STATS ビューの出力は、コーディネータ・プロセスが初期化状態であることを示しています。コーディネータ・プロセスが初期化中の場合、ログ適用サービスは SQL 適用操作の開始準備中ですが、REDO ログのデータはロジカル・スタンバイ・データベースに適用されていません。

注意： ログ適用サービスを初めて開始するときは、データベースの初期化と準備に非常に時間がかかる場合があります。ロジカル・スタンバイ・データベースに多数の表があると、初期化と準備に数時間かかる場合もあります。ただし、最初の準備アクティビティ後に行う再起動では迅速になります。

コーディネータ・プロセスの状態を認識することは特に重要です。これは、その他のすべてのロジカル・スタンバイ・プロセスに指示する LSP バックグラウンド・プロセスであるためです。

手順 5 V$LOGSTDBY ビューを表示して現在の SQL 適用アクティビティを確認する

ロジカル・スタンバイ・データベースで、V$LOGSTDBY ビューを問い合せて、SQL 適用アクティビティの現在のスナップショットを確認します。変更の読込みおよび適用に関係する各プロセスの現在のアクティビティを示すテキスト・メッセージが表示されます。

例 4-2 は、初期化フェーズ時の一般的な出力です。

例 4-2 初期化フェーズ時の V$LOGSTDBY 出力

```
SQL> COLUMN STATUS FORMAT A50

SQL> COLUMN TYPE FORMAT A12

SQL> SELECT TYPE, HIGH_SCN, STATUS FROM V$LOGSTDBY;
TYPE           HIGH_SCN STATUS
------------ ---------- --------------------------------------------------
COORDINATOR             ORA-16115: Log Miner ディクショナリ・データをロードしています
READER                  ORA-16127: 追加のトランザクションが適用されるまで待機して停止しました。
```

```
                        ions to be applied
BUILDER                 ORA-16117: 処理中です。
PREPARER                ORA-16116: 使用できる作業はありません

SQL> SELECT TYPE, HIGH_SCN, STATUS FROM V$LOGSTDBY;
TYPE           HIGH_SCN STATUS
------------ ---------- --------------------------------------------------
COORDINATOR             ORA-16126: 表または順序オブジェクト番号 %d のロード中
READER                  ORA-16116: 使用できる作業はありません
BUILDER                 ORA-16116: 使用できる作業はありません
PREPARER                ORA-16116: 使用できる作業はありません
```

コーディネータ・プロセスがロジカル・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すると、V$LOGSTDBY ビューは、APPLYING ステートを表示して適用開始を示します。

例 4-3 は、適用フェーズ時の一般的な出力です。HIGH_SCN 列の値が増加していることに注意してください。この列の数は、変更が適用されているかぎり増加し続けます。HIGH_SCN 列は、進捗のインジケータとしてのみ機能します。

例 4-3 適用フェーズ時の V$LOGSTDBY 出力

```
SQL> COLUMN NAME FORMAT A30
SQL> COLUMN VALUE FORMAT A30
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME = 'coordinator state';
NAME                           VALUE
------------------------------ ------------------------------
coordinator state              APPLYING

SQL> COLUMN STATUS FORMAT A50
SQL> COLUMN TYPE FORMAT A12
SQL> SELECT TYPE, HIGH_SCN, STATUS FROM V$LOGSTDBY;
TYPE           HIGH_SCN STATUS
------------ ---------- --------------------------------------------------
COORDINATOR             ORA-16117: 処理中です。
READER                  ORA-16127: 追加のトランザクションが適用されるまで待機して
                        停止しました。

BUILDER          191896 ORA-16116: 使用できる作業はありません
PREPARER         191902 ORA-16117: 処理中です。
ANALYZER         191820 ORA-16120: SCN 0x0000.0002ed4e でのトランザクションに対する
                        依存性を計算中です。

APPLIER          191209 ORA-16124: トランザクション 1 16 1598 は待機中です。
```

```
APPLIER          191205 ORA-16116: 使用できる作業はありません
APPLIER          191206 ORA-16124: トランザクション 1 5 1603 は待機中です。

APPLIER          191213 ORA-16117: 処理中です。
APPLIER          191212 ORA-16124: トランザクション 1 20 1601 は待機中です。

APPLIER          191216 ORA-16124: トランザクション 1 4 1602 は待機中です。

11 rows selected.
```

手順 6 ログ適用サービス全体の進捗をチェックする

ログ適用サービス全体の進捗をチェックするには、スタンバイ・データベースで DBA_LOGSTDBY_PROGRESS ビューを問い合せます。次に例を示します。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     180702     180702
```

この問合せ例に示すように、APPLIED_SCN 列と NEWEST_SCN 列の数値が同じ場合は、REDO ログの使用可能なデータがすべて適用されたことを示します。これらの値を DBA_LOGSTDBY_LOG ビューの FIRST_CHANGE# 列の値と比較すると、適用されたログ情報量と未適用のログ情報量を確認できます。

> **関連項目：** ログ転送サービスとログ適用サービスの両方が正常に動作していることの確認方法は、5-32 ページの「REDO ログ・アーカイブ情報の監視」および 6-16 ページの「フィジカル・スタンバイ・データベースに関するログ適用サービスの監視」を参照してください。

5

ログ転送サービス

この章では、ログ転送サービスの概要、およびこのサービスを使用してスタンバイ・データベースへの REDO データの転送を制御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。

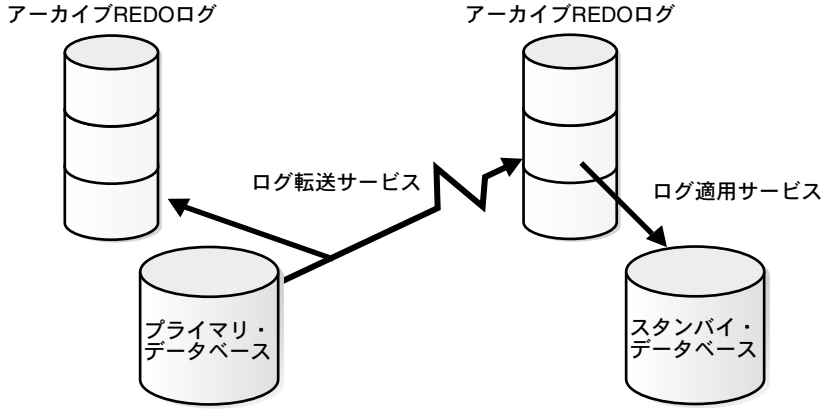
ログ転送サービスの概要

ログ転送サービスは、Data Guard 構成内で REDO データの自動転送を制御します。

ログ転送サービスは、データベースのデータ保護のレベルも制御します。データ保護および可用性とパフォーマンスとのバランスを保つように、ログ転送サービスを構成できます。Data Guard 環境では、ログ転送サービスはログ適用サービスおよびロール管理サービスと協調して、スイッチオーバーおよびフェイルオーバー操作を実行します。

図 5-1 に、ログ転送サービスを使用して REDO ログをプライマリ・データベースからローカルの宛先とリモート・スタンバイ・データベースの宛先にアーカイブする単純な Data Guard 構成を示します。

図 5-1 REDO ログのアーカイブ



次に、ログ転送サービスを理解するために重要な項目を説明します。

- REDO ログ

  REDO ログには、データベースのリカバリに必要なデータが含まれています。REDO ログは、データの更新をスタンバイ・データベースに適用するために、スタンバイ・システムでも使用します。

- REDO ログ宛先

  REDO ログ宛先は、REDO ログの位置とタイプ、およびそれらを管理するためのポリシーを指定します。

- REDO ログの転送と受信

  ログ転送サービスは、REDO データの転送と受信を行います。これには、Data Guard 構成内での REDO データの転送、および REDO ログのデータをディスクにコミットする処理が含まれます。

- データ保護

  アーカイブ先属性とログ転送サービス・オプションを設定して、3 種類のデータ保護モードのいずれかを施行できます。

要約すると、ログ転送サービスは、REDO ログを様々な宛先に転送し、REDO データをアーカイブ REDO ログに書き込みます。

データ保護モード

Data Guard 構成は、常に最大保護、最大可用性または最大パフォーマンスのいずれかのデータ保護モードで実行されます。これら 3 種類の保護モードは、データ保護、データ可用性およびプライマリ・データベースのパフォーマンスに関する異なるバランスをそれぞれ提供します。ビジネス・ニーズに対して最適な保護モードを選択するには、データ保護要件とパフォーマンスに対するユーザーの要求を慎重に検討する必要があります。

最大保護モードは、最も高いレベルのデータ保護を提供します。プライマリ・データベース・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 REDO データが、このモードの最低要件を満たす少なくとも 1 つのフィジカル・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。このような少なくとも 1 つのスタンバイ・データベースに、プライマリ・データベースが REDO データを書き込むことができない場合、そのプライマリ・データベースは、保護されないデータの生成を防ぐために停止します。この保護モードはデータ消失がないことを保証しますが、プライマリ・データベースのパフォーマンスと可用性に非常に大きな影響を与える可能性があります。

最大可用性モードは、2 番目に高いレベルのデータ保護を提供します。プライマリ・データベース・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 REDO データが、このモードの最低要件を満たす少なくとも 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。最大保護モードとは異なり、このような少なくとも 1 つのスタンバイ・データベースに、プライマリ・データベースが REDO データを書き込むことができない場合でも、そのプライマリ・データベースは停止しません。ただし、障害が解決されてスタンバイ・データベースがプライマリ・データベースと同じ状態になるまで、保護モードは、最大パフォーマンス・モードまで一時的に低下します。このモードは、最大パフォーマンス・モードである間、プライマリ・データベースに障害が発生しないかぎり、データ消失がないことを保証します。この保護モードは、プライマリ・データベースの可用性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。

最大パフォーマンス・モードは、デフォルトの保護モードです。プライマリ・データベース・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 REDO データがスタンバイ・データベースに書き込まれるまでコミットを待機しません。したがって、プライマリ・データベースに障害が発生し、コミットされたトランザクションのリカバリに必要な REDO データがどのスタンバイ・データベースにもない場合は、一部のデータが消失する可能性があります。このモードは、プライマリ・データベースのパフォーマンスや可用性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。

これら 3 種類のデータ保護モードでは、構成内の少なくとも 1 つのスタンバイ・データベースがログ転送サービスの特定の属性セットを使用している必要があります。この章では、その属性について詳しく説明します。属性を理解することによって、ビジネスに適した Data Guard 保護モードをサポートできるように、構成を適切に変更できます。

Data Guard 構成のデータ保護モードの設定に使用する SQL 文については、5-26 ページの「Data Guard 構成のデータ保護モードの設定」を参照してください。

REDO データの転送

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、その REDO ログをスタンバイ・データベースに適用して、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。この項では、Data Guard 構成で次のタイプの REDO ログを使用する方法について説明します。

- オンライン REDO ログ
- アーカイブ REDO ログ
- スタンバイ REDO ログ

オンライン REDO ログ

オンライン REDO ログは、Oracle データ・ファイルと制御ファイルに対して行われたすべての変更を記録する、2 つ以上のファイルのセットです。データベースに変更が行われると、Oracle データベース・サーバーは必ずそのデータを書き込み、REDO バッファに REDO レコードを生成します。REDO バッファの内容は、ログ・ライター・プロセスによってオンライン REDO ログにフラッシュされます。

現行のオンライン REDO ログとは、ログ・ライター・プロセスによる書込みが行われているオンライン REDO ログです。ログ・ライター・プロセスは、ファイルの終わりに到達するとログ・スイッチを実行し、新規ログ・ファイルへの書込みを開始します。データベースを ARCHIVELOG モードで実行する場合は、アーカイバ・プロセスがオンライン REDO ログをアーカイブ REDO ログにコピーします。

REDO ログ・グループは、冗長性のために多重化された 2 つ以上の REDO ログのセットです。ログ・ライター・プロセスは、同じ REDO データを 1 つのグループ内のすべての REDO ログに書き込みます。1 つのログで書込みエラーが発生しても、その REDO データは、グループ内の他の REDO ログに対して使用可能です。

プライマリ・サイトでのアーカイブ REDO ログの生成には、オンライン REDO ログのサイズとそれらが切り替わる頻度の両方が影響します。一般的に、オンライン REDO ログのサイズを決定する際に最も重要な要素は、データベースのフェイルオーバー操作時にスタンバイ・データベースに適用する必要のあるアプリケーション・データの量です。オンライン REDO ログのサイズが大きいほど、プライマリ・データベースとの一貫性を保つためにスタンバイ・データベースに適用する必要のあるデータ量が増えます。

Oracle データベース・サーバーは各ログ・スイッチでチェックポイントを試行します。したがって、オンライン REDO ログのサイズが小さすぎる場合は、ログ・スイッチの発生頻度が高いために、チェックポイントの発生頻度も高くなり、スタンバイ・データベースのシステム・パフォーマンスに悪影響を及ぼします。

> **関連項目：** オンライン REDO ログとオンライン REDO ログ・グループの構成の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

アーカイブ REDO ログ

アーカイブ REDO ログは、いっぱいになったオンライン REDO ログ・グループのメンバーのコピーで、データベースが ARCHIVELOG モードのときに作成されます。LGWR プロセスが各オンライン REDO ログを REDO レコードで満たすと、アーカイバ・プロセスがこのログを 1 つ以上のアーカイブ・ログ宛先にコピーします。

いっぱいになったオンライン REDO ログをアーカイブすることによって、メディア・リカバリなどの操作に必要な古い REDO ログ・データが保持されます。一方、事前割当てのオンライン REDO ログは、データベースの最新の変更を保存するために継続して再利用されます。スタンバイ・システムでは、アーカイブ REDO ログを使用して、プライマリ・データベースの変更をスタンバイ・データベースに適用します。

REDO ログをアーカイブするための権限の設定

リモート宛先にオンライン REDO ログをアーカイブするための権限は、REMOTE_ARCHIVE_ENABLE 初期化パラメータを使用して指定します。このパラメータのオプションは、TRUE、FALSE、SEND および RECEIVE です。Data Guard 環境では、ほとんどの場合、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このパラメータを TRUE に設定する必要があります。リモート・アーカイブ REDO ログの送受信を個別に可能または不可能にするには、SEND 値と RECEIVE 値を使用します。

たとえば、プライマリ・データベースが誤ってアーカイブ REDO ログを受信しないようにするには、プライマリ・データベースの REMOTE_ARCHIVE_ENABLE 初期化パラメータを SEND に設定します。逆に、スタンバイ・データベースがスタンバイ REDO ログをリモートでアーカイブしないように、スタンバイ・データベースの REMOTE_ARCHIVE_ENABLE 初期化パラメータを RECEIVE に設定することもできます。

> **関連項目：** ロールの推移の操作に関する初期化パラメータの設定については、第 7 章を参照してください。

アーカイブ REDO ログの再利用の制御

CONTROL_FILE_RECORD_KEEP_TIME 初期化パラメータは、制御ファイル内の再使用可能レコードが再使用可能になるまでの最小日数を指定します。このパラメータを設定すると、ログ転送サービスによる制御ファイル内の再使用可能レコードの上書きを防ぐことができます（このパラメータが適用されるのは、制御ファイル内の逐次再使用可能なレコードのみです)。このパラメータによって、スタンバイ・データベースで確実にアーカイブ REDO ログ情報が使用可能になります。これは、スタンバイ・データベースに適用遅延を指定している場合に特に重要です。このパラメータの値の範囲は 0 ～ 365 日です。デフォルト値は 7 日です。

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

REDO ログの適用に対するタイム・ラグの指定

プライマリ・サイトの REDO ログのアーカイブとスタンバイ・サイトの REDO ログの適用の間にタイム・ラグを作成することが必要な場合があります。タイム・ラグによって、破損したデータや誤ったデータがプライマリ・サイトからスタンバイ・サイトに適用されるのを防止できます。

LOG_ARCHIVE_DEST_*n* 初期化パラメータの DELAY=*minutes* 属性を使用して、スタンバイ・サイトで REDO ログを適用するためのタイム・ラグを指定します。DELAY 属性で指定する遅延間隔は、宛先でのアーカイブ REDO ログの完了に関連しています。この属性によってスタンバイ・データベースへの REDO ログの転送が遅延することはありません。この属性のデフォルト設定は、NODELAY です。DELAY 属性に値が指定されていない場合、この属性の値は 30 分となります。

> **関連項目：**
>
> - 10-16 ページ「タイム・ラグのあるフィジカル・スタンバイ・データベースの使用」
> - ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DELAY 文で DELAY 制御オプションを使用してプライマリ・データベースに指定された適用遅延間隔より優先させるフィジカル・スタンバイ・データベースについては、「ALTER DATABASE RECOVER MANAGED STANDBY DATABASE」を参照してください。
> - DBMS_LOGSTDBY.APPLY_SET プロシージャを使用してプライマリ・データベースに指定された適用遅延間隔より優先させるロジカル・スタンバイ・データベースについては、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

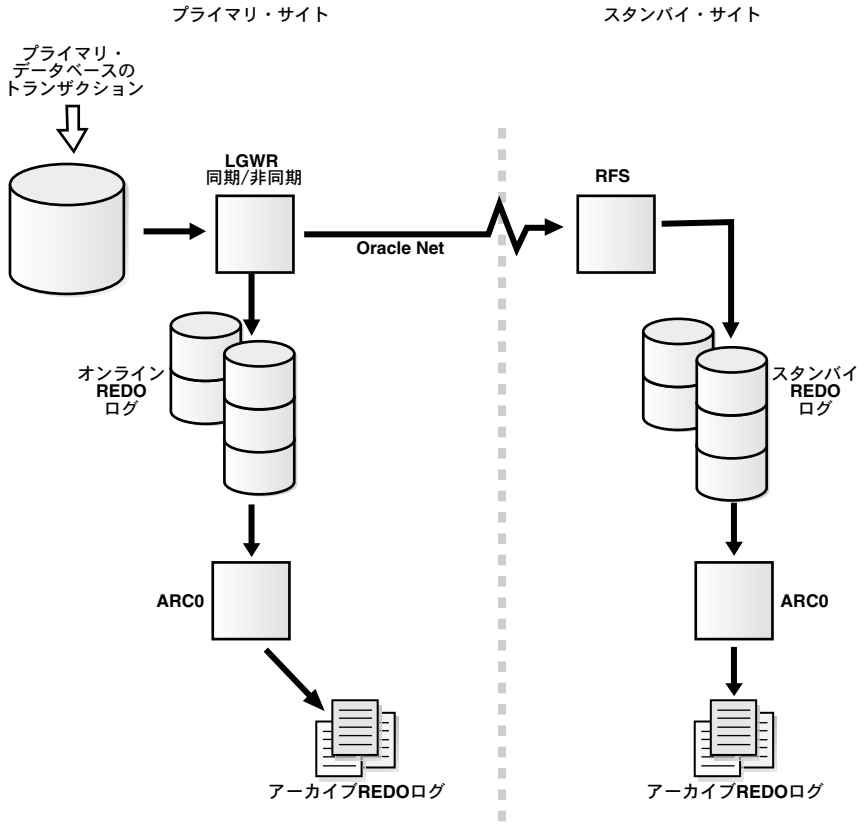
スタンバイ REDO ログ

注意： この項の内容は、フィジカル・スタンバイ・データベースにのみ適用されます。

スタンバイ REDO ログはオンライン REDO ログに似ており、フィジカル・スタンバイ・データベースが最大保護モードと最大可用性モードで実行されている必要があります。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・システムのリモート・ファイル・サーバー・プロセス（RFS）によって受信されます。RFS プロセスは、この REDO データをスタンバイ REDO ログまたはアーカイブ REDO ログのいずれかに書き込みます。

スタンバイ REDO ログは、ログ・ファイル・グループの個別のプールを構成します。フェイルオーバー操作時にスタンバイ REDO ログを使用すると、Data Guard はアーカイブ REDO ログのみを使用する場合よりも多くの REDO データを適用できます。スタンバイ REDO ログは、データがスタンバイ・データベースに適用される前にアーカイブする必要があるため、アーカイバ・プロセスはスタンバイ・データベースで起動する必要があります。図 5-2 に Data Guard 構成を示します。この構成では、RFS プロセスがログ・ライター・プロセスから受信した REDO データをスタンバイ REDO ログに書き込みます。プライマリ・データベースのログ・スイッチによって、スタンバイ・データベースのログ・スイッチがトリガーされると、アーカイバ・プロセスは、スタンバイ REDO ログをスタンバイ・データベースのアーカイブ REDO ログにアーカイブします。

図 5-2 REDO ログ受信オプション



スタンバイ REDO ログのサイズと数

スタンバイ REDO ログのサイズは、プライマリ・データベースのオンライン REDO ログと正確に一致している必要があります。たとえば、プライマリ・データベースが、ログ・サイズが 100K および 200K の 2 つのオンライン REDO ログ・グループをそれぞれ使用する場合、スタンバイ・データベースは、サイズが同じスタンバイ REDO ログ・グループを持ちます。

スタンバイ REDO ログ・グループの数 最小の構成では、プライマリ・データベースの数より 1 つ以上多いスタンバイ REDO ログ・グループが必要です。

スタンバイ REDO ログが RFS プロセスで再利用される前にアーカイブ操作を完了する時間を確保するために、フィジカル・スタンバイ・データベースに追加のスタンバイ・ログ・グループを作成する必要があります。プライマリ・データベースが最大保護モードで実行されていて、スタンバイ REDO ログの割当てができなかった場合、そのプライマリ・データ

ベースのインスタンスはただちに停止します。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実行されている場合、プライマリ・データベースはスタンバイ REDO ログが使用可能になるまで待機します。このため、必ず適切な数のスタンバイ REDO ログを割り当てるようにしてください。

注意： オンライン REDO ログをプライマリ・データベースに追加した場合は、対応するスタンバイ REDO ログをスタンバイ・データベースに追加する必要があります。スタンバイ REDO ログをスタンバイ・データベースに追加しない場合は、プライマリ・データベースが停止する可能性があります。

テスト中、現在のスタンバイ・ログ構成が適切であるかどうかを判断する最も簡単な方法は、RFS プロセスのトレース・ファイルとデータベースのアラート・ログの内容を調べることです。アーカイブが完了しないために RFS プロセスが頻繁にグループを待つ必要があることを示すメッセージがあった場合は、スタンバイ・ログ・グループを追加します。

Real Application Clusters を使用するときは、各種スタンバイ REDO ログが各種プライマリ・データベースのインスタンス間で共有されます。スタンバイ REDO ログ・グループは、特定のプライマリ・データベース・スレッドに専有されることはありません。

スタンバイ REDO ログ・グループのガイドライン スタンバイ REDO ログ・グループの構成を設定または変更する前に、スタンバイ REDO ログ・グループの数を制限するようにデータベース・パラメータを考慮します。次のパラメータは、データベースに追加できるスタンバイ REDO ログ・グループの数を制限します。

- プライマリ・データベースで実行する `CREATE DATABASE` 文の `MAXLOGFILES` 句は、1 つのフィジカル・スタンバイ・データベースに作成することができるスタンバイ REDO ログ・グループの最大数を指定します。この制限を変更する唯一の方法は、プライマリ・データベースまたは制御ファイルを再作成することです。
- `LOG_FILES` 初期化パラメータによって、現在のインスタンスが継続している間、一時的にスタンバイ REDO ログのグループの最大数を減らすことができます。
- プライマリ・データベースで使用する `CREATE DATABASE` 文の `MAXLOGMEMBERS` 句は、グループ当たりのメンバーの最大数を指定します。この制限を変更する唯一の方法は、プライマリ・データベースまたは制御ファイルを再作成することです。

関連項目：『Oracle9*i* SQL リファレンス』

スタンバイ REDO ログの作成

スタンバイ REDO ログは、ALTER DATABASE 文の ADD STANDBY LOGFILE 句を使用して作成します。

スタンバイ REDO ログが作成されていることを確認するには、V$STANDBY_LOG ビュー（スタンバイ REDO ログのステータスとして ACTIVE または INACTIVE を表示）または V$LOGFILE ビューを問い合せます。次の例では、V$LOGFILE ビューを問い合せます。

```
SQL> SELECT * FROM V$LOGFILE WHERE TYPE = 'STANDBY';
```

スタンバイ REDO ログ・グループの作成

> **注意：** スタンバイ REDO ログが使用されるのは、データベースがフィジカル・スタンバイ・ロールで実行されているときのみですが、プライマリ・データベースにスタンバイ REDO ログを作成して、プライマリ・データベースが DBA による介入なしにスタンバイ・ロールに迅速に切り替えられるようにすることをお薦めします。

オンライン REDO ログの多重化と同様に、スタンバイ REDO ログを多重化して REDO ログの可用性を高めることができます。スタンバイ・データベースをインスタンス化した後、データベースのスタンバイ REDO ログ構成を計画して、必要なすべてのグループとグループのメンバーを作成します。新しいスタンバイ REDO ログ・グループとメンバーの作成には、ALTER DATABASE システム権限が必要です。データベースには、SQL CREATE DATABASE 文の MAXLOGFILES 句で指定した値と同じ数のグループを指定できます。

スタンバイ REDO ログの新しいグループを作成するには、ALTER DATABASE 文の ADD STANDBY LOGFILE 句を使用します。

次の文は、スタンバイ REDO ログの新しいグループをフィジカル・スタンバイ・データベースに追加します。

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE
  2> ('/oracle/dbs/log1c.rdo','/oracle/dbs/log2c.rdo') SIZE 500K;
```

GROUP オプションを使用して、グループを識別する番号を指定することもできます。

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 10
  2> ('/oracle/dbs/log1c.rdo','/oracle/dbs/log2c.rdo') SIZE 500K;
```

グループ番号を使用すれば、スタンバイ REDO ログ・グループの管理が容易になります。ただし、グループ番号は 1 と MAXLOGFILES 初期化パラメータの値の間であることが必要です。REDO ログ・ファイル・グループ番号はスキップしないでください（つまり、グループに 10、20、30 などの番号を付けないでください）。スキップすると、フィジカル・スタンバイ・データベースの制御ファイルの領域が余分に使用されます。

フィジカル・スタンバイ・データベースは、プライマリ・データベースで次回ログ・スイッチが発生したときに、新規に作成されたスタンバイ REDO ログの使用を開始します。スタンバイ REDO ログ・グループが作成されて正しく実行していることを確認するには、プライマリ・データベースでログ・スイッチを起動してから、フィジカル・スタンバイ・データベースで V$STANDBY_LOG ビューを問い合せます。

```
SQL> SELECT GROUP#,THREAD#,SEQUENCE#,ARCHIVED,STATUS FROM V$STANDBY_LOG;

GROUP#     THREAD#    SEQUENCE#  ARC STATUS
---------- ---------- ---------- --- ----------
         3          1         16 NO  ACTIVE
         4          0          0 YES UNASSIGNED
         5          0          0 YES UNASSIGNED
```

スタンバイ REDO ログ・メンバーを既存のグループに追加

スタンバイ REDO ログの完全なグループを作成する必要がない場合もあります。グループがすでに存在するものの、1 つ以上のメンバーが削除されているため完全ではないことがあります（たとえば、ディスク障害のために）。この場合には、新しいメンバーを既存のグループに追加できます。

新しいスタンバイ REDO ログ・グループのメンバーを追加するには、ALTER DATABASE 文と ADD STANDBY LOGFILE MEMBER パラメータを使用します。次の文は、新しいメンバーを REDO ログ・グループの番号 2 に追加します。

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE MEMBER '/disk1/oracle/dbs/log2b.rdo'
  2> TO GROUP 2;
```

ファイルの作成場所を示すために新しいログ・メンバーの完全修飾されたファイル名を使用します。完全修飾されたファイル名を使用しないと、データベース・サーバーのデフォルト・ディレクトリまたはカレント・ディレクトリのいずれかにファイルが作成されます。

宛先パラメータと属性

ログ転送サービスは、REDO データを最大 10 箇所の REDO ログ宛先に転送します。プライマリ・データベースでアーカイブを実行するように構成するには、LOG_ARCHIVE_DEST_*n*（*n* は 1 ～ 10 の整数）初期化パラメータおよび対応する LOG_ARCHIVE_DEST_STATE_*n*（*n* は 1 ～ 10 の整数）初期化パラメータを使用します。これらの初期化パラメータは、カスケード・スタンバイ・データベースの設定にも使用できます。詳細は、付録 D を参照してください。

宛先の構成に使用する初期化パラメータがいくつかあります。LOG_ARCHIVE_DEST_*n* などの一部のパラメータには、詳細を設定するための属性があります。

アーカイブ先属性を使用して、位置のみでなく宛先に関するすべての詳細を指定します。特に、次のプロパティを指定します。

- 宛先の位置。
- REDO ログ送受信に関する宛先の特性。
- 依存性など宛先間の関係。
- 宛先の重要度。たとえば、宛先がオプションかどうかを指定します。
- スタンバイ・データベースへの REDO ログ適用のタイム・ディレイ。
- エラー処理および REDO ログの再転送。

アーカイブ先に関連するパラメータは次のとおりです。

- LOG_ARCHIVE_DEST_*n*

  宛先のほとんどの動作とプロパティを制御します。このパラメータには多数の属性があります。LOG_ARCHIVE_DEST_*n* のすべての属性の詳細は、第 12 章を参照してください。

- LOG_ARCHIVE_DEST_STATE_*n*

  宛先の状態を制御します。LOG_ARCHIVE_DEST_*n* パラメータには、それぞれ対応する LOG_ARCHIVE_DEST_STATE_*n* パラメータがあります。

- STANDBY_ARCHIVE_DEST

  スタンバイ・データベース上のアーカイブ REDO ログの位置を指定します。

- LOG_ARCHIVE_FORMAT

  アーカイブ REDO ログのファイル名の形式を指定します。STANDBY_ARCHIVE_DEST と LOG_ARCHIVE_FORMAT が連結して、スタンバイ・データベースのアーカイブ REDO ログの完全修飾されたファイル名が生成されます。

- LOG_ARCHIVE_MIN_SUCCEED_DEST

  プライマリ・データベースのログ・ライター・プロセスがオンライン REDO ログを再利用する前に REDO ログを正常に受信する必要がある、ローカル宛先の最小数を定義します。

- REMOTE_ARCHIVE_ENABLE

  リモート宛先への REDO ログの送信、およびリモート REDO ログの受信を可能または不可能にします。

REDO ログのアーカイブ先の指定

ARCHIVELOG モードで実行するプライマリ・データベースの設定に加えて、宛先および関連する状態を設定することによって、REDO ログをアーカイブするようにプライマリ・データベースを構成する必要があります。この構成には、LOG_ARCHIVE_DEST_*n* 初期化パラメータおよび対応する LOG_ARCHIVE_DEST_STATE_*n* パラメータを使用します。

LOG_ARCHIVE_DEST_STATE_*n*（*n* は 1 ～ 10 の整数）初期化パラメータは、LOG_ARCHIVE_DEST_*n* 初期化パラメータ（*n* は同じ整数）が示す宛先の状態を指定します。たとえば、LOG_ARCHIVE_DEST_STATE_3 パラメータは、LOG_ARCHIVE_DEST_3 宛先の状態を指定します。

表 5-1 で、LOG_ARCHIVE_DEST_STATE_*n* パラメータの属性を説明します。

表 5-1 LOG_ARCHIVE_DEST_STATE_n 初期化パラメータの属性

属性	説明
ENABLE	ログ転送サービスはこの宛先で REDO ログをアーカイブできる。
DEFER	ログ転送サービスはこの宛先には REDO ログをアーカイブしない。これは使用されない宛先である。
ALTERNATE	この宛先は使用可能ではないが、別の宛先への通信に障害が起きると使用可能になる。

ログ転送サービスを設定して、リモート・ノード上にある payroll2 という名前のスタンバイ・データベースに REDO ログをアーカイブするには、プライマリ・データベース初期化パラメータ・ファイルを次のように修正してください。これらの修正は、次回のログ・スイッチ後に適用されます。次に例を示します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=payroll2';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;
```

アーカイブ REDO ログとスタンバイ REDO ログの格納場所の指定

スタンバイ・データベースで STANDBY_ARCHIVE_DEST 初期化パラメータを使用し、アーカイブ REDO ログを格納するディレクトリを指定します。ログ転送サービスでは、LOG_ARCHIVE_FORMAT パラメータと共にこの値を使用してスタンバイ・サイトでアーカイブ REDO ログのファイル名を生成します。

パラメータ	指示	例
STANDBY_ARCHIVE_DEST	アーカイブ・オンライン REDO ログを配置するディレクトリ	STANDBY_ARCHIVE_DEST= /arc_dest/
LOG_ARCHIVE_FORMAT	アーカイブ・オンライン REDO ログのファイル名の形式	LOG_ARCHIVE_FORMAT = "log%d_%t_%s.arc" **注意 :** %d はデータベース ID に、%s は順序番号に対応しています。Real Application Clusters の構成に必要な %t は、スレッドに対応しています。

ログ転送サービスは、完全修飾されたファイル名をスタンバイ制御ファイルに格納します。ログ適用サービスは、この情報を使用してスタンバイ・データベースでリカバリ操作を実行します。次の例では、LOG_ARCHIVE_FORMAT 初期化パラメータの設定方法を示します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_FORMAT='log%d_%t_%s.arc';
```

プライマリ・データベースで次の問合せを発行し、スタンバイ・システム上のアーカイブ REDO ログのリストを表示します。

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG;
NAME
--------------------------------------------------------------------------------
/arc_dest/log_1_771.arc
/arc_dest/log_1_772.arc
/arc_dest/log_1_773.arc
/arc_dest/log_1_774.arc
/arc_dest/log_1_775.arc
```

スタンバイ REDO ログを使用する場合は、スタンバイ・データベースの LOG_ARCHIVE_DEST_*n* 初期化パラメータ（*n* は 1 ～ 10 の値）によって、スタンバイ REDO ログをアーカイブするディレクトリが指定されます。

パラメータ	指示	例
LOG_ARCHIVE_DEST_*n*	スタンバイ・サイトのアーカイブ REDO ログの格納ディレクトリ	LOG_ARCHIVE_DEST_1= 'LOCATION=/oracle/stby/arc/' **注意 :** このパラメータを定義しない場合は、STANDBY_ARCHIVE_DEST パラメータの値が使用されます。
LOG_ARCHIVE_FORMAT	アーカイブ・オンライン REDO ログのファイル名の形式	LOG_ARCHIVE_FORMAT = "log%d_%t_%s.arc" **注意 :** %d はデータベース ID に、%s は順序番号に対応しています。Real Application Clusters の構成に必要な %t は、スレッドに対応しています。

注意 : スタンバイ REDO ログを使用する場合は、スタンバイ・データベースでアーカイバ・プロセス（ARC*n*）を使用可能にする必要があります。スタンバイ・データベースの LOG_ARCHIVE_START 初期化パラメータは、常に TRUE に設定することをお薦めします。

必須およびオプションの宛先の指定

LOG_ARCHIVE_DEST_*n* パラメータに OPTIONAL または MANDATORY 属性を使用して、オンライン REDO ログを再利用するためのポリシーを指定できます。リモート宛先は OPTIONAL に設定することをお薦めします （これがデフォルトです）。OPTIONAL の宛先へのアーカイブ操作に障害が発生した場合、オンライン REDO ログが上書きされます。必須の宛先へのアーカイブ操作に障害が発生した場合、オンライン REDO ログは上書きされません。

デフォルトでは、すべてのローカル宛先を OPTIONAL に指定しても、1 つの宛先は MANDATORY になります。

例 5-1 に、必須のローカル・アーカイブ先を設定してその宛先を使用可能にする方法を示します。

例 5-1 必須のアーカイブ先の設定

```
LOG_ARCHIVE_DEST_3 = 'LOCATION=/arc_dest MANDATORY'
```

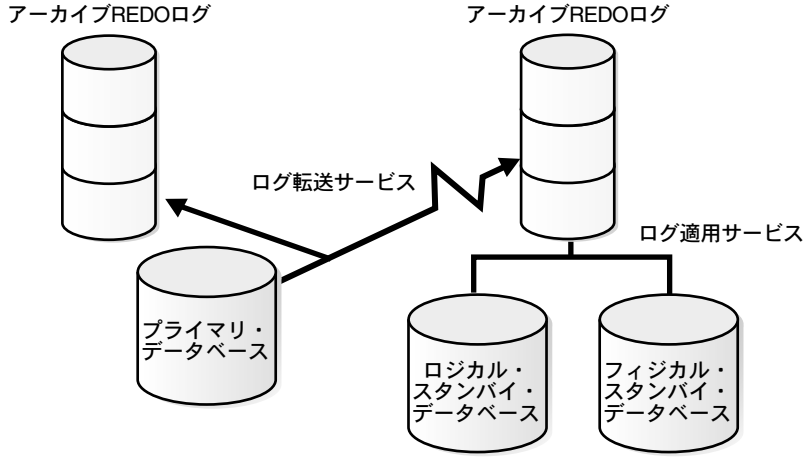
複数のスタンバイ・データベース間でのログ・ファイル宛先の共有

リモート・データベースへの REDO ログのアーカイブは、別の宛先に対するアーカイブ操作が成功したか失敗したかに依存するように定義できます。これは、依存する宛先と呼ばれます。

依存する宛先の定義には、LOG_ARCHIVE_DEST_*n* 初期化パラメータの DEPENDENCY 属性を使用します。この属性は、この宛先が親宛先に対するアーカイブ操作の正常な完了に依存することを示しています。

図 5-3 に示す Data Guard 構成では、プライマリ・データベースは 1 つのアーカイブ先に REDO データを転送します。このアーカイブ先は、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・データベースの共有の宛先として使用されます。

図 5-3 依存する宛先を持つ Data Guard 構成



宛先依存性の指定は次の場合に役立ちます。

- 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・データベースを同じノード上に構成する場合。
- スタンバイ・データベースとプライマリ・データベースが同じノード上にある場合。したがって、スタンバイ・データベースは暗黙的にアーカイブ REDO ログへアクセスできます。
- プライマリ・データベースのアーカイブ REDO ログへのアクセスをリモート・スタンバイ・データベースに提供する OS（オペレーティング・システム）固有のネットワーク・ファイル・システムが使用されている場合。

- 地理的に離れた位置に対する透過的ネットワーク・サポートを使用して、ミラー化ディスク・テクノロジを使用する場合。
- 同じリモート・ノード上に複数のスタンバイ・データベースがあり、交互に発生する管理リカバリ操作のために、共通のアーカイブ REDO ログへのアクセスを共有している場合。

これらの場合、物理的なアーカイブ操作は不要ですが、スタンバイ・データベースでは、アーカイブ REDO ログの位置を認識している必要があります。この認識によって、スタンバイ・データベースは、アーカイブ REDO ログがログ適用サービスによる適用で使用可能になると、そのアーカイブ REDO ログにアクセスできます。別（親）の宛先へのアーカイブが成功したか失敗したかに依存するようにアーカイブ先を指定する必要があります。

アーカイブ障害ポリシーの指定

LOG_ARCHIVE_DEST_*n* 初期化パラメータの REOPEN 属性と MAX_FAILURES 属性を使用して、宛先へのアーカイブに障害が起きた場合に実行するアクションを指定します。該当するアクションは、次のとおりです。

- 指定時間が経過した後、障害が起きた宛先へのアーカイブ操作を、制限回数まで再試行する
- 代替または置換宛先を使用する

エラーにより障害が起きた宛先に対する REDO ログの再アーカイブを、アーカイバ・プロセスまたはログ・ライター・プロセスで試行するかどうかを決定し、その時期を決定するには、LOG_ARCHIVE_DEST_*n* パラメータの REOPEN 属性を使用します。

REOPEN=*seconds* 属性を使用して、エラー発生後から障害が起きた宛先に対しアーカイブ処理を再実行するまでの経過時間の最小秒数を指定します。デフォルト値は 300 秒です。REOPEN 属性の値セットは、接続障害だけでなくすべてのエラーに適用されます。このオプションは、NOREOPEN を指定してオフにできます。これによって、障害が起きた宛先に対するアーカイブ処理は再実行されません。

REOPEN 属性を MAX_FAILURE 属性とともに使用して、障害が起きた宛先との通信の再確立を連続して試みる回数を制限できます。指定した連続試行数を超えると、その宛先は NOREOPEN 属性が指定されているものとして扱われます。

MAX_FAILURE 属性を使用する場合は、REOPEN 属性は必須です。例 5-2 に、再試行時間を 60 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 5-2 再試行時間の設定と制限

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=60 MAX_FAILURE=3'
```

その他の宛先タイプ

リモート宛先には、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、アーカイブ・ログ・リポジトリおよびインスタンス間アーカイブ・データベース環境の 4 つのタイプがあります。第 1 章で説明したフィジカル・スタンバイ・データベースとロジカル・スタンバイ・データベースは、最もよく使用される宛先です。次に、これ以外の宛先について説明します。

- アーカイブ・ログ・リポジトリ

  このタイプのアーカイブ先では、REDO ログのアーカイブがオフサイトで可能です。アーカイブ・ログ・リポジトリは、フィジカル・スタンバイ制御ファイルを使用し、インスタンスを起動して、データベースをマウントすることによって作成されます。このデータベースにはデータ・ファイルは含まれず、プライマリ・データベースのリカバリ用には使用できません。短期間（たとえば 1 日）REDO ログを保持し、その後ログを削除する方法として便利です。これによって、他の完全に構成されたスタンバイ・データベースの格納と処理の手間をほとんど省くことができます。

- インスタンス間アーカイブ・データベース環境

  インスタンス間アーカイブ・データベース環境は、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作成可能です。Real Application Clusters 環境内では、各インスタンスがそのアーカイブ REDO ログをクラスタの単一インスタンスに転送します。このインスタンスは、**リカバリ・インスタンス**と呼ばれ、一般的には管理リカバリが実行されるインスタンスです。リカバリ・インスタンスには一般的に、Recovery Manager によるバックアップとリストア・サポートに使用できるテープ・ドライブがあります。

REDO データの転送と受信

ログ転送サービスは、Data Guard 構成内のすべての REDO ログを自動的に転送および受信します。転送と受信の特性を調整して、データ保護レベルとパフォーマンスとのバランスを保つことができます。

REDO ログをリモートの宛先へアーカイブするには、Oracle Net を介した接続が中断されないことが必要です。宛先がリモートのフィジカル・スタンバイ・データベースの場合、そのフィジカル・スタンバイ・データベースはマウントされているか、あるいはアーカイブ REDO ログを受信するために読取り専用モードでオープンしている必要があります。また、ロジカル・スタンバイ・データベースはオープンしている必要があります。

ログ転送サービスに関する次の事項を指定できます。

- REDO ログを転送するプロセス
- REDO ログのネットワーク転送モード
- REDO ログのデータをディスクに書き込む方法

REDO データを転送するプロセスの指定

プライマリ・データベースに障害が発生した場合のデータ消失を最小限にするため、データの生成時に、プライマリ・データベースからスタンバイ・データベースにそのデータをコピーできます。REDO ログを宛先に転送するプロセスには、ログ・ライター・プロセスまたはアーカイバ・プロセスのいずれかを選択できます。

REDO データの転送プロセスを指定するには、LOG_ARCHIVE_DEST_*n* 初期化パラメータの ARCH または LGWR 属性のいずれかを使用します。

属性	例	デフォルト
{ARCH | LGWR}	LOG_ARCHIVE_DEST_3='SERVICE=stby1 LGWR'	ARCH

LGWR と ARCH 属性は相互に排他的です。したがって、同じ宛先に対し両方の属性を指定することはできません。ただし、個々の宛先には、LGWR または ARCH 属性を指定できます。このため、アーカイバ・プロセスが REDO データをある宛先に転送している間に、ログ・ライター・プロセスが別の宛先に REDO データを転送するように指定できます。

ARCH 属性を選択した場合は、プライマリ・データベースでログ・スイッチが発生すると、アーカイバ・プロセス（ARC*n*）がカレント REDO ログを対応付けられた宛先にアーカイブします。これはデフォルトの設定値です。

LGWR 属性を選択した場合は、ログ・ライター・プロセス（LGWR）が、REDO データの生成時に REDO データを対応付けられた宛先に転送します。プライマリ・データベース用に生成された REDO は、スタンバイ・システムにも伝播され、RFS プロセスがその REDO をスタンバイ REDO ログまたはスタンバイ・アーカイブ REDO ログのいずれかに書き込みます。

ネットワーク転送モードの指定

データを消失しない唯一の方法は、REDO データをプライマリ・データベースでコミットする前にスタンバイ・データベースに書き込むことです。SYNC 属性を指定すると、すべてのネットワーク I/O 操作が、オンライン REDO ログへの各書込み操作と同期して実行されます。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 REDO データが宛先で受信されるまで、プライマリ・データベースでコミットされません。

ログ・ライター・プロセスを使用して REDO ログをアーカイブするときは、アーカイブ先に対して SYNC または ASYNC 属性を使用することで、REDO ログの同期（SYNC）または非同期（ASYNC）ネットワーク転送を指定できます。SYNC 属性または ASYNC 属性のいずれも指定しない場合、デフォルトは SYNC ネットワーク転送モードです。次に、各転送モードについて説明します。

- SYNC ネットワーク転送方法

  SYNC 属性によって、プライマリ・データベースのパフォーマンスが悪影響を受ける可能性がありますが、宛先サイトにおけるデータ保護のレベルは最も高くなります。同期転送は、非データ消失環境で必要です。

- ASYNC ネットワーク転送方法

  ASYNC 属性を指定すると、すべてのネットワーク I/O 操作が非同期で実行され、実行中のアプリケーションまたはユーザーに即時に制御が戻されます。ブロック・カウントを指定して、使用する SGA ネットワーク・バッファのサイズを決めることができます。ブロック・カウントは、0 ～ 20,480 まで許されます。この属性では、オプションの接尾辞の値 K で 1,000 を表すことができます（値 1K は 1,000 個の 512 バイト・ブロックを示します）。速度の遅いネットワーク接続では、通常、大きなブロック・カウントを使用します。

関連項目： 第 12 章「LOG_ARCHIVE_DEST_n パラメータの属性」

REDO データのディスク書込み

LOG_ARCHIVE_DEST_*n* 初期化パラメータの [NO]AFFIRM 属性を使用して、ログ・アーカイブのディスク書込み I/O 操作を、同期または非同期のいずれで実行するのかを指定します。

注意： AFFIRM および NOAFFIRM 属性の適用先は、オンライン・アーカイブ・ログ宛先のみであるため、オンライン REDO ログ・ディスク I/O 操作には影響を与えません。

サンプル構成でのログ転送サービス

ログ転送サービスは、REDO データを Data Guard 構成内のシステムに転送します。REDO データの転送に使用されるプロセスは、次のとおりです。

- ログ・ライター（LGWR）

  ログ・ライター・プロセスは、プライマリ・データベースのトランザクション REDO データを収集し、オンライン REDO ログを更新します。さらに、LGWR は、オンライン REDO データを直接スタンバイ・システムに転送できます。

- アーカイバ（ARCn）

  アーカイバ・プロセスは、オンライン REDO ログとスタンバイ REDO ログの両方をアーカイブ先にコピーします。ローカルまたはリモートのアーカイブ先があります。アーカイバ・プロセスは、プライマリ・システムとスタンバイ・システムの両方で実行されます。

- リモート・ファイル・サーバー（RFS）

  リモート・ファイル・サーバーはスタンバイ・システムで実行され、ネットワーク上で LGWR と ARC*n* の両方から REDO データを受信します。RFS プロセスは、REDO データをスタンバイ REDO ログまたはスタンバイ・アーカイブ REDO ログのいずれかに書き込みます。

- フェッチ・アーカイブ・ログ（FAL）

  フェッチ・アーカイブ・ログ（FAL）プロセスは、アーカイブ REDO ログのギャップの解決に役立ちます。フィジカル・スタンバイ・データベースに REDO ログの欠落がある場合は、ローカルの FAL クライアントがそのログをフェッチします。

次の各図に、様々な構成でのログ転送サービスの作動方法を示します。

図 5-4 に、単一のローカル宛先がある最も単純な構成を示します。ログ・ライター・プロセスは、REDO データをオンライン REDO ログに書き込みます。各オンライン REDO ログがいっぱいになると、ログ・スイッチが発生し、アーカイバ・プロセスは、いっぱいになったオンライン REDO ログをアーカイブ REDO ログにアーカイブします。これによって、いっぱいになったオンライン REDO ログが再利用できます。

図 5-4 スタンバイ・データベースがない場合のプライマリ・データベースのアーカイブ処理

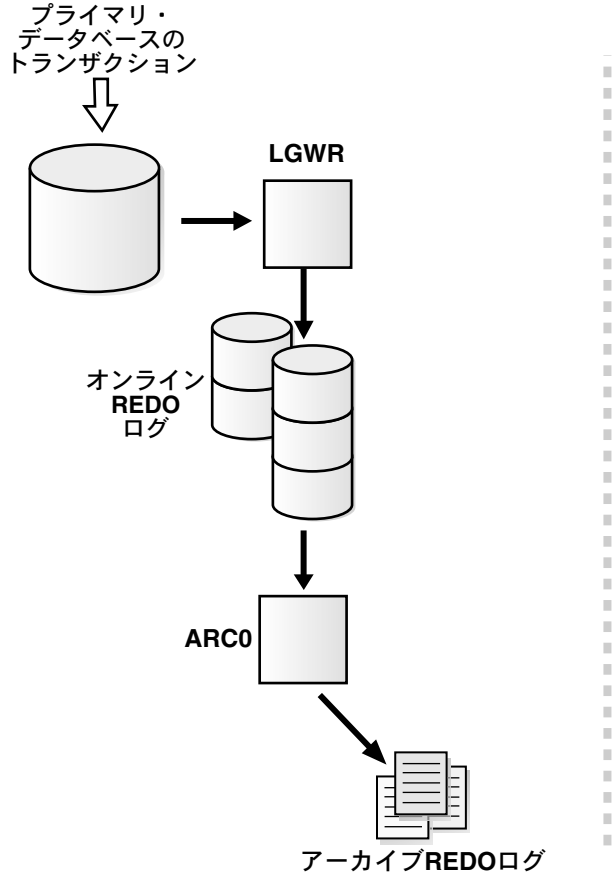


図 5-5 に、ローカル宛先とスタンバイ宛先がある Data Guard 構成を示します。ログ・スイッチが発生すると、アーカイバ・プロセスはローカル宛先とスタンバイ宛先の両方にアーカイブします。アーカイバ・プロセスは、Oracle Net を使用し、ネットワークを介して REDO データを RFS プロセスに送信します。RFS プロセスは、その REDO データをスタンバイ・データベースのアーカイブ REDO ログに書き込みます。この図では、管理リカバリ・プロセス（MRP）またはロジカル・スタンバイ・プロセス（LSP）が、REDO ログをスタンバイ・データベースに適用するために使用されています。

関連項目： MRP および LSP プロセスについては、第 6 章を参照してください。

図 5-5 基本的な Data Guard 構成

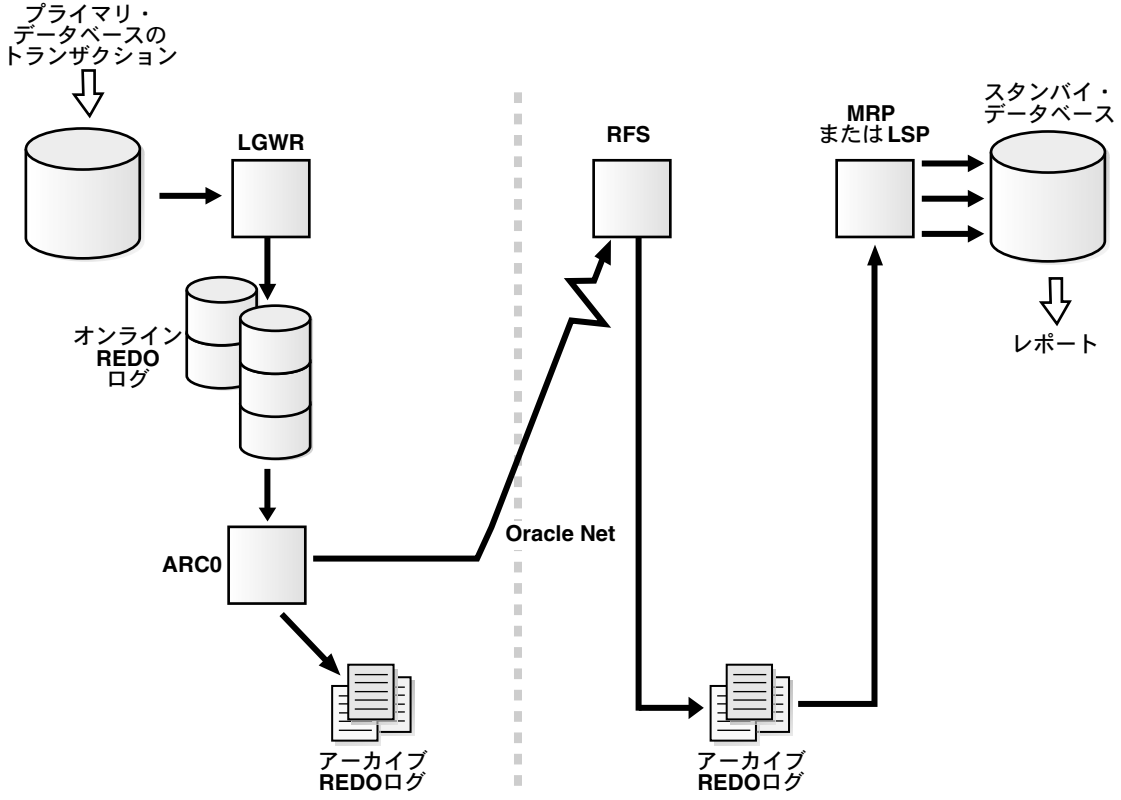


図 5-6 に、ローカル宛先とスタンバイ宛先がある Data Guard 構成を示します。この構成では、プライマリ・システムのアーカイバがアーカイブする先はローカル宛先のみです。ログ・ライター・プロセスは、REDO データをオンライン REDO ログに書き込むと同時に、そのデータをスタンバイ・システムに送信します。RFS プロセスは、その REDO データをスタンバイ・データベースのオンライン REDO ログに書き込みます。プライマリ・データベースのログ・スイッチによって、スタンバイ・データベースのログ・スイッチがトリガーされると、スタンバイ・データベースのアーカイバ・プロセスは、REDO ログをスタンバイ・データベースのアーカイブ REDO ログにアーカイブします。この構成は、最高レベルのデータ保護に対する前提です。フィジカル・スタンバイ・データベースの場合は、スタンバイ REDO ログを使用することをお薦めします。スタンバイ REDO ログは、ロジカル・スタンバイ・データベースではサポートされていません。

図 5-6 ログ・ライター・プロセスを使用したフィジカル・スタンバイ宛先へのアーカイブ処理

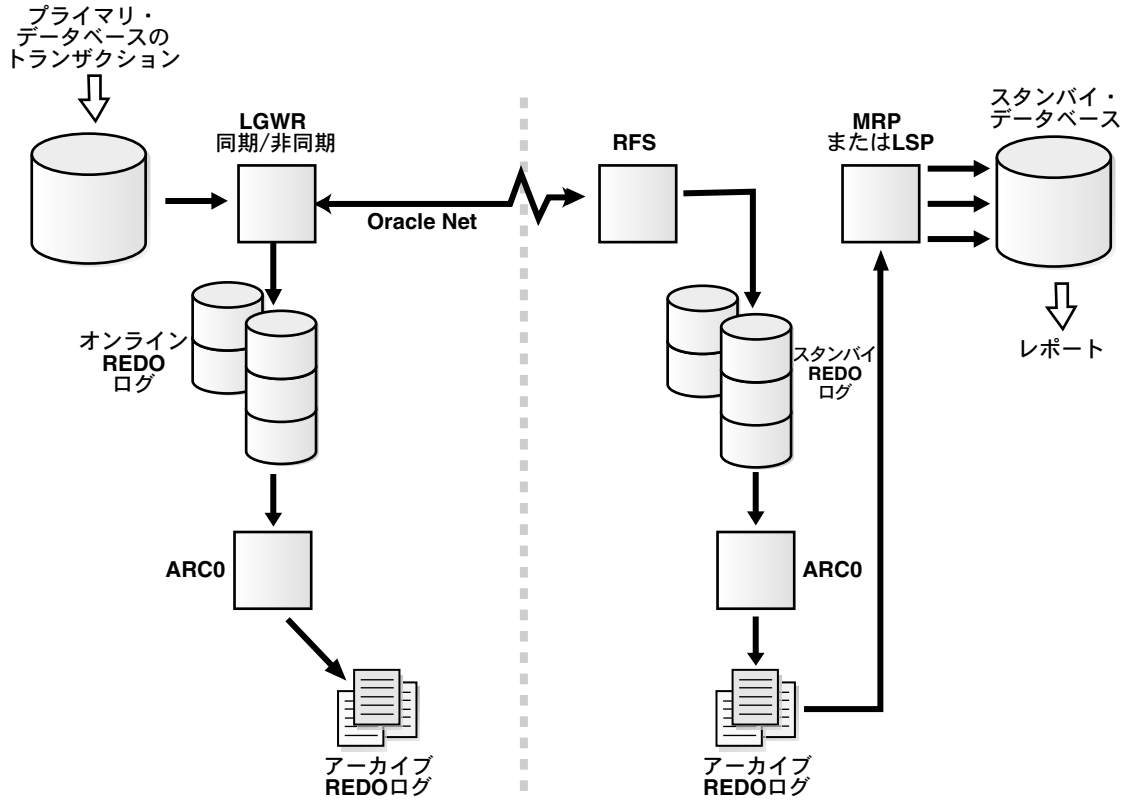
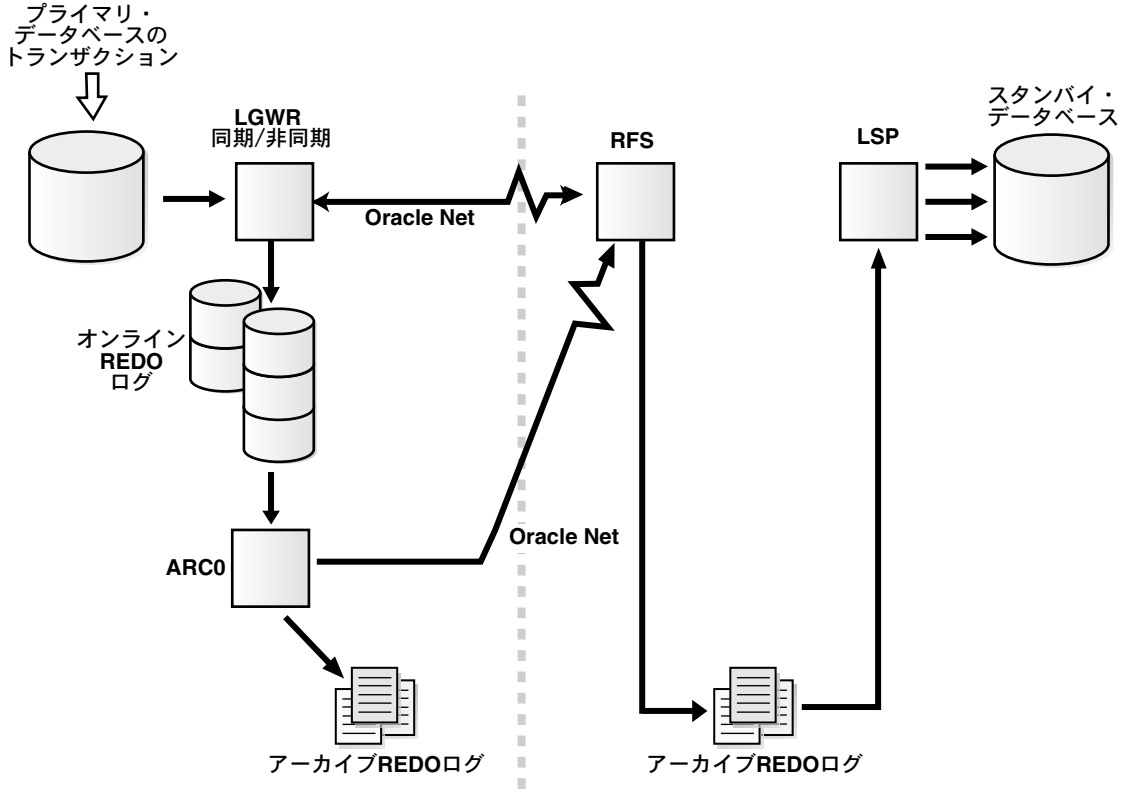


図 5-7 に、ローカル宛先とリモート・ロジカル・スタンバイ宛先がある Data Guard 構成を示します。この構成では、プライマリ・システムのアーカイバがアーカイブする先はローカル宛先のみです。ログ・ライター・プロセスは、REDO データをオンライン REDO ログに書き込むと同時に、そのデータをスタンバイ・システムに送信します。RFS プロセスは、その REDO データを受信してスタンバイ・データベースのアーカイブ REDO ログに書き込みます。プライマリ・データベースのログ・スイッチによって、スタンバイ・データベースのログ・スイッチがトリガーされると、スタンバイ・データベースのアーカイバ・プロセスは、REDO ログをスタンバイ・データベースでアーカイブします。

図 5-7 ログ・ライター・プロセスを使用したロジカル・スタンバイ宛先へのアーカイブ処理



Data Guard 構成のデータ保護モードの設定

Data Guard の各データ保護モードでは、構成内の少なくとも 1 つのスタンバイ・データベースが表 5-2 に記載された一連の最低要件を満たしている必要があります。

表 5-2 データ保護モードの要件

	最大保護	最大可用性	最大パフォーマンス
REDO アーカイブ・プロセス	LGWR	LGWR	LGWR または ARCH
ネットワーク転送モード	SYNC	SYNC	LGWR プロセスを使用する場合は ASYNC。ARCH プロセスを使用する場合は該当しない。
ディスク書込みオプション	AFFIRM	AFFIRM	NOAFFIRM
スタンバイ REDO ログの必要性	必須	フィジカル・スタンバイ・データベースでのみ必須	LGWR プロセスを使用するフィジカル・スタンバイ・データベースで必須
データベースのタイプ	フィジカル	フィジカルおよびロジカル	フィジカルおよびロジカル

指定モードの最低要件を満たすために使用したスタンバイ・データベースは、その指定モードに切り替える前に使用可能にし、REDO データをプライマリ・データベースから受信する準備が整っている必要があります。

構成のデータ保護モードを変更する前に、3 つのデータ保護モードの説明を見直してください。ビジネスで必要なデータ保護レベル、およびその保護レベルを提供するモードで操作を行った場合のパフォーマンスと可用性への影響を慎重に検討してください。

注意： 最大保護モードで実行する Data Guard 構成には、表 5-2 に記載された要件を満たす少なくとも 2 つのフィジカル・スタンバイ・データベースを含めることをお薦めします。これによって、1 つのフィジカル・スタンバイ・データベースがプライマリ・データベースから REDO データを受信できない場合でも、プライマリ・データベースは処理を継続できます。

Data Guard 構成が、使用する保護モードの最低要件を満たしていることを確認してから、ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE 文を使用してそのモードに切り替えます。この文の構文は次のとおりです。

```
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE}
```

関連項目： SQL 文については、第 13 章および『Oracle9*i* SQL リファレンス』を参照してください。

ログ転送サービスの管理

次の各項では、データベース初期化パラメータを使用してログ転送サービスのオプションを制御する方法について説明します。

データベース初期化パラメータ

プライマリ・データベースおよびスタンバイ・データベースの初期化パラメータはほとんどが同じですが、CONTROL_FILES パラメータや DB_FILE_NAME_CONVERT パラメータなどの一部の初期化パラメータは異なります。パラメータ値を変更するのは、スタンバイ・データベースの機能やファイル名変換で必要な場合のみです。

関連項目： スタンバイ・データベースの構成で重要な役割を果たす初期化パラメータのリストは、第 11 章を参照してください。

初期化パラメータ・ファイルでのログ転送パラメータの設定

ログ転送サービスを設定するには、データベース・インスタンスの起動前に、データベースの初期化パラメータ・ファイルを変更します。従来のテキストの初期化パラメータ・ファイルを使用する場合は、すべてのパラメータを 1 行で指定する必要があります。

注意： この項の例では、パラメータと変更は異なる方法で指定できることが理解できるように、従来のテキストの初期化パラメータ・ファイルを使用しています。

例 5-3 に、1 行に 1 つの属性を持つパラメータを指定する方法を示します。

例 5-3 1 行に 1 つの属性を指定

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll'
```

例 5-4 に、1 行に複数の属性を持つパラメータを指定する方法を示します。

例 5-4 1 行に複数の属性を指定

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll LGWR'
```

ログ転送サービスのほとんどの初期化パラメータでは、新規の値を指定すると、前に指定した値は新規の値に完全に置き換えられます。

SQL 文を使用して実行時にログ転送パラメータを設定

実行時には、ALTER SYSTEM SET 文を使用して LOG_ARCHIVE_DEST_*n* 初期化パラメータを変更できます。属性は、1 つ以上の文字列で 1 つの文に指定できます。SCOPE=MEMORY 句がある ALTER SYSTEM SET 文を使用して行う変更は、永続的ではありません。

例 5-5 に、1 行に 1 つの属性を持つパラメータを指定する方法を示します。

例 5-5 1 行に 1 つの属性を指定

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll';
```

ログ転送サービスのほとんどの初期化パラメータでは、新規の値を指定すると、前に指定した値は新規の値に完全に置き換えられます。

ロールの推移に関する初期化パラメータの準備

ロールの推移後に Data Guard 構成が正しく動作するように、スイッチオーバーやフェイルオーバーの操作を実行する前には、プライマリおよびスタンバイ・データベースの両方で特定の初期化パラメータを構成する必要があります。

作成プロセスで構成した内容は、次のとおりです。

- プライマリ・データベースの初期化パラメータ。スタンバイ・データベースへの REDO データの転送に関連しています。
- スタンバイ・データベースの初期化パラメータ。スタンバイ・データベースで REDO データを受信する準備をします。
- スタンバイ・データベースに対するプライマリ・システムのネットワーク・アドレス。
- プライマリ・データベースに対するスタンバイ・システムのネットワーク・アドレス。

ここでは、プライマリ・データベースをスタンバイ・ロールで動作するように設定し、スタンバイ・データベースをプライマリ・ロールで動作するように設定する必要があります。

次の各項では、ログ転送サービスとログ適用サービスに関連した初期化パラメータについてのみ説明します。プライマリおよびスタンバイ・データベースの作成時に設定する他のパラメータについては説明していません。次の各項で説明していないパラメータは、そのまま変更しないか（DATABASE_NAME パラメータなど）、必要かつ可能な場合にのみ要件にあわせて変更できます（LOCK_NAME_SPACE パラメータや SHARED_POOL パラメータなど）。

プライマリ・データベースの初期化パラメータ

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ログ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 データの受信とログ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 5-6 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。

例 5-6 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

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll/'
LOG_ARCHIVE_DEST_2='SERVICE=sales1'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_FORMAT=%d_%t_%s.arc
REMOTE_ARCHIVE_ENABLE=SEND
```

これらのパラメータは、ログ転送サービスが REDO データをスタンバイ・システムに送信する方法と、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。最後のパラメータの REMOTE_ARCHIVE_ENABLE=SEND によって、プライマリ・データベースは、REDO データをスタンバイ・データベースに送信できますが、別のシステムからは REDO データを受信できません。

例 5-7 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 5-7 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=sales1
FAL_CLIENT=sales
DB_FILE_NAME_CONVERT=('/standby','/primary')
LOG_FILE_NAME_CONVERT=('/standby','/primary')
STANDBY_ARCHIVE_DEST=/disk1/oracle/oradata/payroll/
STANDBY_FILE_MANAGEMENT=AUTO
```

例 5-7 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とログ・ファイル・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間は着信 REDO データがアーカイブされます。

スタンバイ・データベースの初期化パラメータ

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

例 5-8 に、スタンバイ・データベースでメンテナンスする、スタンバイ・ロールの初期化パラメータを示します。

例 5-8 スタンバイ・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=sales
FAL_CLIENT=sales1
DB_FILE_NAME_CONVERT=("/primary","/standby")
LOG_FILE_NAME_CONVERT=("/primary","/standby")
STANDBY_ARCHIVE_DEST=/disk1/oracle/oradata/payroll/standby/arc
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll/standby/arc/'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_FORMAT=%d_%t_%s.arc
STANDBY_FILE_MANAGEMENT=AUTO
REMOTE_ARCHIVE_ENABLE=RECEIVE
```

これらの初期化パラメータによって、スタンバイ・データベースで次の処理を実行できます。

- ギャップを解決し、プライマリ・データベースからの新規データとログ・ファイル・パス名を変換します。
- プライマリ・データベースからREDOデータを受信してアーカイブするのは、データベースがスタンバイ・ロールで実行されている間のみです。

最後のパラメータによって、スタンバイ・データベースは、プライマリ・データベースからREDOデータを受信できますが、送信することはできません。

例 5-9 に、スタンバイ・データベースに追加するプライマリ・ロールのパラメータを示します。これらのパラメータは、スタンバイ・データベースがプライマリ・ロールに推移すると有効になります。

例 5-9 スタンバイ・データベース : プライマリ・ロールの初期化パラメータ

```
LOG_ARCHIVE_DEST_2='SERVICE=sales'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

これらの追加パラメータによって、ログ転送サービスがREDOデータを新しいスタンバイ・システムに送信する方法を制御します。

ロールの推移時の初期化パラメータの有効化

「プライマリ・データベースの初期化パラメータ」および「スタンバイ・データベースの初期化パラメータ」の説明に従ってプライマリおよびスタンバイ・データベースに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のは、REMOTE_ARCHIVE_ENABLE パラメータのみです。このパラメータは、元のプライマリ・データベースとプライマリ・ロールに推移するスタンバイ・データベースの両方で変更します。

元のプライマリ・データベース（新しいスタンバイ）でこのパラメータを設定し、新しいプライマリ・データベースから REDO を受信できるようにします。次に例を示します。

```
SQL> ALTER SYSTEM SET REMOTE_ARCHIVE_ENABLE=RECEIVE SCOPE=MEMORY;
```

新しいプライマリ・データベース（元のスタンバイ）でこのパラメータを設定し、スタンバイ・データベースに REDO を送信できるようにします。

```
SQL> ALTER SYSTEM SET REMOTE_ARCHIVE_ENABLE=SEND SCOPE=MEMORY;
```

SCOPE=MEMORY 句を使用して初期化パラメータを設定すると、ロールが再度推移したとき、2 つのデータベースは元の設定に戻り、元のロールで再開されます。2 つのデータベースがロールの推移を実行せずに、ある時点で再起動する場合は、SCOPE=MEMORY を SCOPE=BOTH に置換します。この場合、この初期化パラメータは、新たにロールが推移した後に再度手動でリセットする必要があります。

ロジカル・スタンバイ・データベースに関する考慮事項

「プライマリ・データベースの初期化パラメータ」および「スタンバイ・データベースの初期化パラメータ」の例で示したパラメータ値は、フィジカル・スタンバイとロジカル・スタンバイの両方の構成で使用できます。プライマリ・データベースとロジカル・スタンバイ・データベースの間でロールの推移を実行する場合、設定するアーカイブ先は、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（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のために）とで異なる必要があります。

ロジカル・スタンバイ・データベースを作成するときは、例 5-10 に示すように、初期化パラメータ用に異なるディレクトリを指定する必要があります。

例 5-10 ロジカル・スタンバイ・データベース : スタンバイ・ロールの初期化パラメータ

```
STANDBY_ARCHIVE_DEST=/disk1/oracle/oradata/payroll/standby/incoming
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll/standby/arc/'
```

これらのパラメータは、スタンバイ・データベースがプライマリ・ロールで動作している間は、適切に機能し続けます。

また、ロールの推移後にプライマリ・データベースがロジカル・スタンバイ・ロールで動作する場合は、ロジカル・スタンバイ・データベースのアーカイブ先とは異なる場所に着信 REDO データを転送するように、ローカル・アーカイブ・パラメータを構成する必要があります。

例 5-11 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

```
STANDBY_ARCHIVE_DEST=/disk1/oracle/oradata/payroll/incoming
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll/arc/'
```

例 5-11 のパラメータ値を設定すると、プライマリ・データベースから送信された REDO データ・ストリームは、ロジカル・スタンバイ・ロールで動作中のデータベースによって生成されたアーカイブ・ログとは異なる場所にアーカイブされます。

REDO ログ・アーカイブ情報の監視

この項では、プライマリ・データベースの REDO ログ・アーカイブ・アクティビティの監視を手動で行う方法を説明します。

> **関連項目：** Data Guard 環境の監視に関係する多数のタスクを自動化する Oracle9*i* Data Guard Manager のグラフィカル・ユーザー・インタフェースの詳細は、『Oracle9*i* Data Guard Broker』および Data Guard Manager のオンライン・ヘルプを参照してください。

手順 1 カレント REDO ログ順序番号を判定する

プライマリ・データベースで次の問合せを入力して、カレント REDO ログ順序番号を判定します。

```
SQL> SELECT THREAD#, SEQUENCE#, ARCHIVED, STATUS FROM V$LOG;

THREAD#   SEQUENCE#  ARC  STATUS
--------  ---------  ---  ------
       1        947  YES   ACTIVE
       1        948  NO    CURRENT
```

手順 2 最新のアーカイブ REDO ログを判定する

プライマリ・データベースで次の問合せを入力して、最新のアーカイブ REDO ログ・ファイルを判定します。

```
SQL> SELECT MAX(SEQUENCE#) FROM V$ARCHIVED_LOG;

MAX(SEQUENCE#)
--------------
           947
```

手順 3 各アーカイブ先で最新のアーカイブ REDO ログ・ファイルを判定する

プライマリ・データベースで次の問合せを入力して、アーカイブ先ごとに、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、リストされた各アーカイブ先で同じである必要があります。同じでない場合は、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 4 ログが特定のサイトで受信されたかどうかを確認する

ログが特定のサイトに送信されなかったかどうかは、プライマリ・データベースで問合せを発行して確認できます。各アーカイブ先には対応付けられた ID 番号があります。プライマリ・データベースの V$ARCHIVE_DEST 固定ビューの DEST_ID 列に問合せを発行して、アーカイブ先の ID を特定できます。

カレント・ローカル・アーカイブ先が 1 で、リモート・スタンバイ・アーカイブ先の ID の 1 つが 2 とします。この場合、どのログがこのスタンバイ宛先で受信されなかったのかを特定するために、次の問合せを発行します。

```
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
```

関連項目： プライマリ・データベースのアーカイブ・ステータス監視の詳細は、付録 A「スタンバイ・データベースのトラブルシューティング」を参照してください。

手順 5 スタンバイ・サイトでアーカイブ REDO ログの進行状況をトレースする

スタンバイ・サイトへの REDO ログのアーカイブの進行状況を確認するには、プライマリおよびスタンバイ初期化パラメータ・ファイルに LOG_ARCHIVE_TRACE パラメータを設定します。

関連項目： 詳細と例は、6-25 ページの「アーカイブ・トレースの設定」を参照してください。

6

ログ適用サービス

この章では、REDO ログをスタンバイ・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。

- ログ適用サービスの概要
- 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
- REDO データのロジカル・スタンバイ・データベースへの適用
- アーカイブ・ギャップの管理
- フィジカル・スタンバイ・データベースに関するログ適用サービスの監視
- ロジカル・スタンバイ・データベースに関するログ適用サービスの監視
- アーカイブ・トレースの設定

ログ適用サービスの概要

ログ適用サービスは、アーカイブ REDO ログ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。アーカイブされた REDO データをログ適用サービスで使用できるのは、プライマリ・データベースでログ・スイッチが発生した後です。

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、ログ適用サービスによるアーカイブ REDO ログ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、ログ適用サービスは、管理リカバリ操作を実行してスタンバイ・データベースをメンテナンスします。ロジカル・スタンバイ・データベースの場合、ログ適用サービスは、SQL 文を実行してスタンバイ・データベースをメンテナンスします。次に、これらの操作をまとめて示します。

- 管理リカバリ操作（フィジカル・スタンバイ・データベースのみ）

  このモードでは、ログ転送サービスがスタンバイ・サイトに REDO データを転送し、ログ適用サービスがその REDO ログを自動的に適用します。

  > **注意：** ユーザーがレポート生成のためにスタンバイ・データベースを問合せできるように、フィジカル・スタンバイ・データベースを読取り専用操作でオープンすることもできます。ただし、読取り専用アクセスでオープンしているスタンバイ・データベースは、プライマリ・データベースとのトランザクションの同期が維持されないため、障害時リカバリで必要な場合は、フェイルオーバーやスイッチオーバーの操作に時間がかかります。詳細は、8-3 ページの「読取り専用アクセス用にオープンしたスタンバイ・データベースの使用」を参照してください。

- SQL 適用操作（ロジカル・スタンバイ・データベースのみ）

  ログ適用サービスは、SQL 文を実行してロジカル・スタンバイ・データベースを管理します。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。SQL 適用モードでは、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、管理リカバリ操作、SQL 適用操作およびログ適用サービスについて詳細に説明します。

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

フィジカル・スタンバイ・データベースでは、いくつかのプロセスを使用して、スタンバイ・データベースでの REDO データのアーカイブと REDO ログのリカバリを自動化します。スタンバイ・データベースでは、ログ適用サービスが次のプロセスを使用します。

- リモート・ファイル・サーバー（RFS）

  リモート・ファイル・サーバー（RFS）・プロセスは、プライマリ・データベースから REDO データをアーカイブ REDO ログまたはスタンバイ REDO ログのいずれかの形式で受信します。

- アーカイバ（ARC*n*）

  スタンバイ REDO ログを使用する場合、ARC*n* プロセスは、管理リカバリ・プロセス（MRP）で適用されるスタンバイ REDO ログをアーカイブします。

- 管理リカバリ・プロセス（MRP）

  管理リカバリ・プロセス（MRP）は、アーカイブ REDO ログからの情報をスタンバイ・データベースに適用します。管理リカバリ操作を実行すると、ログ適用サービスは、アーカイブ REDO ログを自動的に適用し、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。

ログ適用サービスは、フィジカル・スタンバイ・データベースがリカバリを実行しているときは、ログをデータベースに適用できますが、読取り専用操作でデータベースがオープンしているときは適用できません。フィジカル・スタンバイ・データベースでは、次のいずれかの操作を実行できます。

- 管理リカバリ操作
- 読取り専用操作

表 6-1 に、ログ適用サービスを構成および監視するための基本タスクをまとめます。

表 6-1 タスク・リスト : フィジカル・スタンバイ・データベースに関するログ適用サービスの構成

手順	タスク	参照先
1	スタンバイ・インスタンスを起動し、スタンバイ・データベースをマウントする。	6-4 ページ「フィジカル・スタンバイ・インスタンスの起動」
2	管理リカバリ操作または読取り専用操作を使用可能にする。	6-5 ページ「ログ適用サービスの開始」または 8-3 ページ「読取り専用アクセス用にオープンしたスタンバイ・データベースの使用」
3	管理リカバリ操作を実行する場合は、初期化パラメータを設定してアーカイブ・ギャップを自動的に解決する。	6-12 ページ「アーカイブ・ギャップの管理」および『Oracle9*i* Net Services 管理者ガイド』
4	ログ適用サービスを監視する。	6-16 ページ「フィジカル・スタンバイ・データベースに関するログ適用サービスの監視」

フィジカル・スタンバイ・インスタンスの起動

スタンバイ・インスタンスは、必要なパラメータとネットワーク・ファイルをすべて構成した後で起動できます。スタンバイ・インスタンスが起動およびマウントされていない場合、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 データを受信できません。

フィジカル・スタンバイ・データベースのインスタンスを起動する手順は、次のとおりです。

1. データベースをマウントせずにフィジカル・スタンバイ・インスタンスを起動します。

```
SQL> STARTUP NOMOUNT;
```

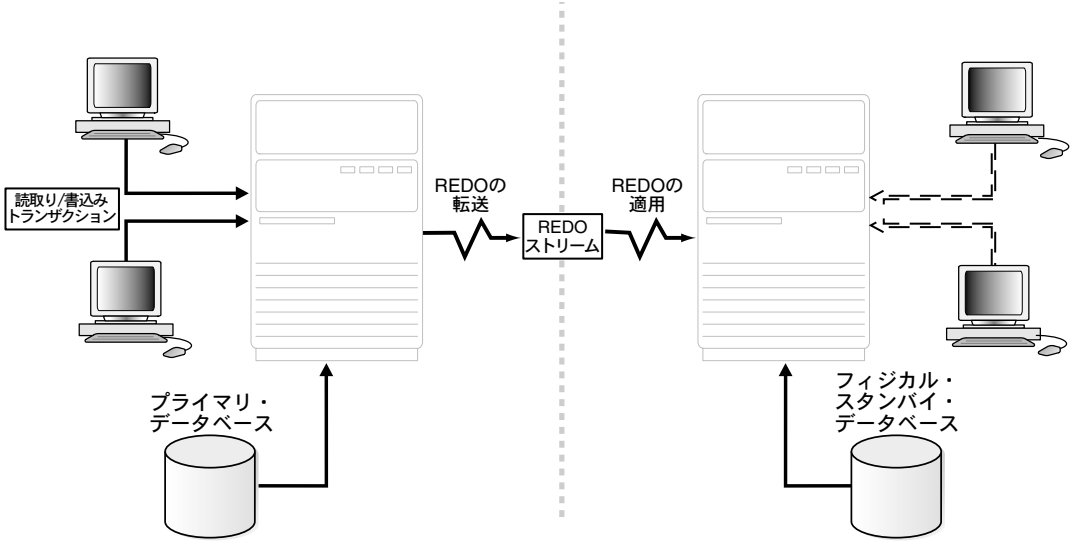
2. フィジカル・スタンバイ・データベースをマウントします。次に例を示します。

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

管理リカバリ操作の開始

ログ適用サービスは、図 6-1 に示すように、アーカイブ REDO ログをスタンバイ・データベースに自動的に適用することによって、スタンバイ・データベースとプライマリ・データベースの同期を維持します。

図 6-1 フィジカル・スタンバイ・データベースの自動更新



ログ適用サービスの開始

ログ適用サービスをフォアグラウンド・セッションとして実行するか、バックグラウンド・プロセスとして実行するかを指定できます。

- フォアグラウンド・セッションを開始するには、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、デフォルトでは、制御がコマンド・プロンプトに戻りません。

- バックグラウンド・セッションを開始するには、SQL 文で DISCONNECT キーワードを使用する必要があります。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。

- ログ適用サービスを、分離されたサーバー・プロセスとして開始しなかった場合は、次の SQL 文を別のウィンドウで発行してログ適用サービスを停止できます。

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

関連項目： 6-7 ページ「REDO 適用操作の制御」および 第 13 章

リカバリ処理の監視

次のように、ビューを問い合せてログ適用サービスを監視できます。

1. ログ適用サービスを正常に開始したかどうかを確認するには、スタンバイ・データベースで V$MANAGED_STANDBY 固定ビューを問い合せます。このビューは、管理リカバリ・モードでのスタンバイ・データベースの進捗を監視します。次に例を示します。

```
SQL> SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, BLOCK#, BLOCKS
  2> FROM V$MANAGED_STANDBY;

PROCESS STATUS       THREAD#    SEQUENCE#  BLOCK#     BLOCKS
------- ------------ ---------- ---------- ---------- ----------
MRP0    APPLYING_LOG 1          946        10         1001
```

   サーバー・プロセスをバックグラウンドで起動しなかった場合は、別の SQL セッションからこの問合せを実行する必要があります。

2. スタンバイ・データベースでのアクティビティを監視するには、V$ARCHIVE_DEST_STATUS 固定ビューを問い合せます。

   関連項目： 6-16 ページ「フィジカル・スタンバイ・データベースに関するログ適用サービスの監視」

REDO 適用操作の制御

この SQL ALTER DATABASE RECOVER MANAGED STANDBY DATABASE 文に追加の句は不要ですが、REDO 適用プロセスの制御に役立つキーワードが多数用意されています。

> **関連項目：** SQL 文の構文の詳細は、13-10 ページの「ALTER DATABASE RECOVER MANAGED STANDBY DATABASE」および『Oracle9*i* SQL リファレンス』を参照してください。

データ・ファイルの管理

プライマリ・データベースでデータ・ファイルを作成したときに、フィジカル・スタンバイ・データベースでデータ・ファイルを自動作成するには、STANDBY_FILE_MANAGEMENT 初期化パラメータを定義する必要があります。

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。

STANDBY_FILE_MANAGEMENT 初期化パラメータの設定

STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定すると、プライマリ・データベースで新規に作成されたデータ・ファイルが、プライマリ・データベースで指定した名前と同じ名前で、スタンバイ・データベースで自動的に作成されます。

STANDBY_FILE_MANAGEMENT 初期化パラメータは、DB_FILE_NAME_CONVERT パラメータと連携して、データ・ファイルの位置をプライマリ・サイトからスタンバイ・サイトに変換します。

DB_FILE_NAME_CONVERT 初期化パラメータの設定

新規データ・ファイルがプライマリ・データベースに追加されると、同じデータ・ファイルがスタンバイ・データベースに作成されます。DB_FILE_NAME_CONVERT パラメータは、プライマリ・データベースのデータ・ファイル名をスタンバイ・データベースのデータ・ファイル名に変換するために使用されます。このパラメータの動作は、STANDBY_FILE_MANAGEMENT 初期化パラメータの設定が AUTO の場合も MANUAL の場合も同じです。

DB_FILE_NAME_CONVERT 初期化パラメータには、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。次に例を示します。

```
DB_FILE_NAME_CONVERT= "/disk1/oracle/oradata/payroll/df1", \
"/disk1/oracle/oradata/payroll/standby/df1", \
"/disk1/oracle/oradata/payroll", "/disk1/oracle/oradata/payroll/standby/"
STANDBY_FILE_MANAGEMENT=AUTO
```

注意： ファイルをペアで指定する場合は、例に示すように、最も詳細に指定されたパス名から順に指定してください。

ALTER DATABASE 操作の制限事項

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。

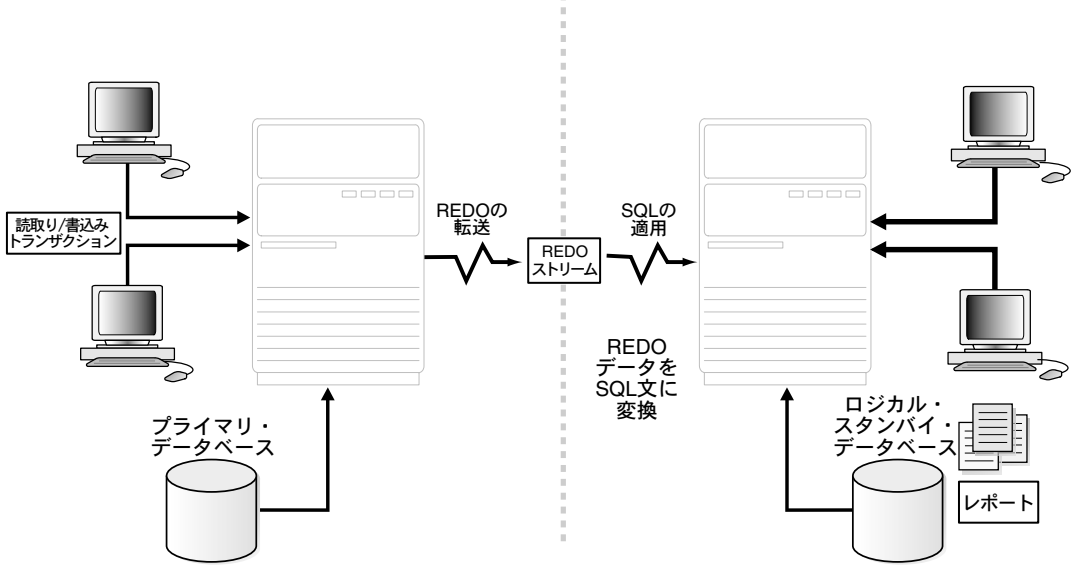
```
SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/t_db2.log' to 'dummy';
alter database rename file '/disk1/oracle/oradata/payroll/t_db2.log' to 'dummy'
*
ERROR at line 1:
ORA-01511: ログ / データ・ ファイルの名前の変更中にエラーが発生しました。
ORA-01270:STANDBY_FILE_MANAGEMENT が auto の場合、RENAME 操作はできません。
```

関連項目： データベースへデータ・ファイルを追加する方法は、8-11 ページの「データ・ファイルの追加または表領域の作成」を参照してください。

REDO データのロジカル・スタンバイ・データベースへの適用

ログ適用サービスは、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、メンテナンスされている表は、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。図 6-2 に、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ログ適用サービスを示します。

図 6-2 ロジカル・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・データベースでは、次のプロセスを使用します。

- リモート・ファイル・サーバー（RFS）

  リモート・ファイル・サーバー・プロセスは、プライマリ・データベースから REDO データを受信します。RFS プロセスは、ロジカル・スタンバイ・プロセス（LSP）と通信して、受信したファイルを調整して記録します。

- ロジカル・スタンバイ・プロセス（LSP）

  ロジカル・スタンバイ・プロセスは、アーカイブ REDO ログからの完了した SQL トランザクションの読込み、準備、作成、分析および適用を同時に行うコーディネータ・プロセスです。LSP もデータベースのメタデータをメンテナンスします。

表 6-2 に、ログ適用サービスを構成するための基本タスクをまとめます。

表 6-2 タスク・リスト：ロジカル・スタンバイ・データベースに関するログ適用サービスの構成

手順	タスク	参照先
1	ログ適用サービスを開始する。	6-10 ページ「ログ適用サービスの開始と停止」
2	REDO ログが適用されていることを確認する。	6-10 ページ「REDO ログの適用の確認」
3	SQL 適用操作を管理する。	9-2 ページ「ロジカル・スタンバイ・データベースの構成と管理」

次の各項では、表 6-2 に示すタスクの詳細に加えて、アーカイブ REDO ログの適用を遅延する方法についても説明します。

ログ適用サービスの開始と停止

ログ適用サービスを開始するには、ロジカル・スタンバイ・データベースを起動して、次の文を使用します （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方法で起動します)。

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

ログ適用サービスを停止するには、次の文を使用します。

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

REDO ログの適用の確認

REDO ログは、スタンバイ・サイトで受信されたときではなく、ログ・スイッチが発生したときにロジカル・スタンバイ・データベースに読み込まれて適用されます。次のビューを問い合せて、アーカイブ REDO ログの適用操作の状態を確認できます。

- V$LOGSTDBY

  このビューを使用して、アーカイブ REDO ログがスタンバイ・データベースに適用されていることを確認します。このビューには、REDO データを読み込んでいる処理およびアーカイブ REDO ログをロジカル・スタンバイ・データベースに適用している処理に関する情報が表示されます。たとえば、次の問合せは、初期化フェーズ時の一般的な出力を示しています。

```
SQL> COLUMN STATUS FORMAT A50
SQL> COLUMN TYPE FORMAT A12
SQL> SELECT TYPE, HIGH_SCN, STATUS FROM V$LOGSTDBY;
TYPE           HIGH_SCN STATUS
------------ ---------- --------------------------------------------------
COORDINATOR             ORA-16115: Log Miner ディクショナリ・データをロードしています
READER                  ORA-16127: 追加のトランザクションが適用されるまで待機して
                        停止しました。
BUILDER                 ORA-16117: 処理中です。
PREPARER                ORA-16116: 使用できる作業はありません

SQL> SELECT TYPE, HIGH_SCN, STATUS FROM V$LOGSTDBY;
TYPE           HIGH_SCN STATUS
------------ ---------- --------------------------------------------------
COORDINATOR             ORA-16126: 表または順序オブジェクト番号 %d のロード中
READER                  ORA-16116: 使用できる作業はありません
BUILDER                 ORA-16116: 使用できる作業はありません
PREPARER                ORA-16116: 使用できる作業はありません
```

- DBA_LOGSTDBY_PROGRESS

  このビューを使用して、ログ適用サービスの進捗に関する情報を確認します。このビューには、LSP の状態、およびロジカル・スタンバイ・データベースで実行された SQL トランザクションに関する情報が表示されます。次に例を示します。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     180702     180702
```

  この問合せ例のように、APPLIED_SCN 列と NEWEST_SCN 列の数値が同じ場合は、REDO ログの使用可能なデータがすべて適用されたことを示します。これらの値を DBA_LOGSTDBY_LOG ビューの FIRST_CHANGE# 列の値と比較すると、適用されたログ情報量と未適用のログ情報量を確認できます。

  関連項目： ロジカル・スタンバイ・データベースの管理については第 9 章を、Data Guard 環境で使用するビューの詳細は第 14 章を参照してください。

アーカイブ・ギャップの管理

Data Guard には、アーカイブ REDO ログのギャップを自動的に検出して解決する機能があり、1 つ以上のスタンバイ・データベースとプライマリ・データベースとの一時的な切断の原因となるネットワーク接続の問題を処理します。適切に構成された Data Guard の場合、このようなギャップの検出と解決に、DBA による手動操作は必要ありません。

次の各項では、ギャップの検出と解決について説明します。

アーカイブ・ギャップ

アーカイブ・ギャップとは、プライマリ・データベースによって生成された次のアーカイブ REDO ログを、スタンバイ・システムで受信できないときに作成されるアーカイブ REDO ログの範囲です。たとえば、アーカイブ・ギャップは、ネットワークが使用不可能になり、プライマリ・データベースからスタンバイ・データベースへの自動アーカイブが停止すると発生します。ネットワークが再度使用可能になると、プライマリ・データベースからスタンバイ・データベースへの REDO データの自動転送が再開します。

欠落しているアーカイブ REDO ログがギャップです。このギャップは自動的に検出および解決されます。

アーカイブ・ギャップの検出時期

アーカイブ・ギャップは、プライマリ・データベースがアーカイブしたログが、スタンバイ・サイトにはアーカイブされなかった場合に発生する可能性があります。プライマリ・データベースは、1 分ごとにスタンバイ・データベースをポーリングして、アーカイブ REDO ログの順序番号にギャップがあるかどうかを確認します。プライマリ・データベースとスタンバイ・データベースとの間のポーリングは、ハートビートと呼ばれることがあります。プライマリ・データベースは、スタンバイ・データベースを逐次的にポーリングします。

アーカイブ・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の判断

次の各項では、適切なビューを問い合せて、スタンバイ・データベースで欠落しているログを判断する方法を説明します。

フィジカル・スタンバイ・データベースの場合

フィジカル・スタンバイ・データベースにアーカイブ・ギャップが存在するかどうかを判断するには、次の例に示すように、V$ARCHIVE_GAP ビューを問い合せます。

```
SQL> SELECT * FROM V$ARCHIVE_GAP;

    THREAD#  LOW_SEQUENCE#  HIGH_SEQUENCE#
----------  -------------  --------------
          1              7              10
```

この出力例は、現在フィジカル・スタンバイ・データベースでスレッド 1 の順序番号 7 ～ 10 のログが欠落していることを示しています。ギャップを識別した後、プライマリ・データベースで次の SQL 文を発行し、プライマリ・データベースのアーカイブ REDO ログ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 DEST_ID=1 AND
  2> 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
```

これらのログをフィジカル・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 文を使用してフィジカル・スタンバイ・データベースに登録します。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';
                                        :
                                        :
```

ログをフィジカル・スタンバイ・データベースに登録した後は、管理リカバリ操作を再開できます。

注意： フィジカル・スタンバイ・データベースの V$ARCHIVE_GAP 固定ビューが戻すのは、管理リカバリの続行をブロックしている次のギャップのみです。識別したギャップを解決して管理リカバリを開始した後、V$ARCHIVE_GAP 固定ビューをフィジカル・スタンバイ・データベースで再度問い合せて、次のギャップ・シーケンス（存在する場合）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。

ロジカル・スタンバイ・データベースの場合

アーカイブ・ギャップが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
```

欠落しているログをロジカル・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 LOGFILE 文を使用してロジカル・スタンバイ・データベースに登録します。次に例を示します。

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE /disk1/oracle/dbs/log-1292880008_10.arc;
```

ログをロジカル・スタンバイ・データベースに登録した後は、ログ適用サービスを再開できます。

注意： ロジカル・スタンバイ・データベースの DBA_LOGSTDBY_LOG ビューが戻すのは、SQL 適用操作の続行をブロックしている次のギャップのみです。識別したギャップを解決してログ適用サービスを開始した後、DBA_LOGSTDBY_LOG ビューをロジカル・スタンバイ・データベースで再度問い合せて、次のギャップ・シーケンス（存在する場合）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。

ギャップの解決方法

Data Guard は、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・データベースの両方で、ギャップの検出と解決を自動的に実行します。追加の構成設定は必要ありません。ただし、フィジカル・スタンバイ・データベースの場合は、フィジカル・スタンバイ・データベースで発生したアーカイブ・ギャップもログ適用サービスで自動的に解決するように、初期化パラメータを設定できます。

次の各項では、フィジカル・スタンバイ・データベースでギャップ・リカバリを実行するための初期化パラメータの設定方法、およびロジカル・スタンバイ・データベースでのギャップ・リカバリの処理方法について説明します。

フィジカル・スタンバイ・データベースの場合

フィジカル・スタンバイ・データベースで発生したアーカイブ・ギャップを、ログ適用サービスが自動的に識別して解決するように、初期化パラメータを設定できます。

初期化パラメータ・ファイルの FAL_CLIENT および FAL_SERVER 初期化パラメータを、フィジカル・スタンバイ・データベースに対してのみ次のように定義してください。

パラメータ	機能	構文
FAL_CLIENT	このパラメータは、FAL サーバーがスタンバイ・データベースへの接続に使用するネットワーク・サービス名を指定します。	**構文** FAL_CLIENT=*net_service_name* **例** FAL_CLIENT=standby1_db
FAL_SERVER	このパラメータは、スタンバイ・データベースが FAL サーバーへの接続に使用するネットワーク・サービス名を指定します。	**構文** FAL_SERVER=*net_service_name* **例** FAL_SERVER=my_primary_db, my_standby_db

FAL サーバーは、FAL クライアントからの着信要求をサービスするバックグラウンド Oracle プロセスです。多くの場合、FAL サーバーはプライマリ・データベース上にあります。しかし、別のスタンバイ・データベース上にある場合もあります。

ログ適用サービスがアーカイブ・ギャップを自動的に識別し解決するには、次のようにします。

1. スタンバイ・システム上で、Oracle Net Manager を使用してリスナーを構成します。TCP/IP プロトコルを使用し、サービス名を使用してスタンバイ・データベース・サービスをリスナーに静的に登録します。このサービス名は、FAL クライアントとして使用されます。
2. Oracle Net Manager を使用して、スタンバイ・データベースが FAL サーバーへの接続に使用できるネットワーク・サービス名を作成します。ネットワーク・サービス名は、FAL サーバー・システム（通常はプライマリ・システム）でのリスナー構成時に指定した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名を使用する接続記述子に解析される必要があります。これらのパラメータに使用する値が不明の場合は、FAL サーバー・システムで Oracle Net Manager を実行し、リスナー構成を表示します。
3. スタンバイ・データベースの初期化パラメータ・ファイルで、手順 1 で作成したネットワーク・サービス名を FAL_CLIENT 初期化パラメータに割り当て、手順 2 で作成したネットワーク・サービス名を FAL_SERVER 初期化パラメータに割り当てます。
4. FAL サーバー・システムで、Oracle Net Manager を使用して、FAL サーバーがスタンバイ・データベースへの接続に使用できるネットワーク・サービス名を作成します。ネットワーク・サービス名は、手順 1 と同じプロトコル、ホスト・アドレス、ポートおよび SID を使用する接続記述子に解析される必要があります。

ALTER DATABASE RECOVER MANAGED STANDBY DATABASE 文で管理リカバリを使用可能にすると、存在するギャップをログ適用サービスが自動的に検出し、プライマリ・データベースで実行中の FAL サーバー・プロセスが解決を試みます。

> **関連項目：** 手動による手順の説明は B-5 ページの「手動によるアーカイブ・ギャップの解決」を、Oracle Net に関する情報は『Oracle9*i* Net Services 管理者ガイド』を参照してください。

ロジカル・スタンバイ・データベースの場合

ロジカル・スタンバイ・データベースでのギャップ・リカバリは、ハートビート・メカニズムを使用して処理されます。ここで重要なことは、自動ギャップ・リカバリの実行には、プライマリ・データベースの可用性が不可欠であることです。フェイルオーバーの発生などでプライマリ・データベースが使用不可能な場合、自動ギャップ・リカバリは実行されません。

フィジカル・スタンバイ・データベースに関するログ適用サービスの監視

フィジカル・スタンバイ・データベースのアーカイブ REDO ログの状況を監視し、ログ適用サービスに関する情報を取得するには、この項で説明する固定ビューを問い合せます。スタンバイ・データベースは、Data Guard Manager を使用して監視することもできます。

> **関連項目：** 付録 A「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。

- V$MANAGED_STANDBY 固定ビューへのアクセス
- V$ARCHIVE_DEST_STATUS 固定ビューへのアクセス
- V$ARCHIVED_LOG 固定ビューへのアクセス
- V$LOG_HISTORY 固定ビューへのアクセス
- V$DATAGUARD_STATUS 固定ビューへのアクセス

> **関連項目：** これらのビューの詳細は、第 14 章を参照してください。

V$MANAGED_STANDBY 固定ビューへのアクセス

スタンバイ・サイトでログ適用サービスおよびログ転送サービスのアクティビティを監視するには、フィジカル・スタンバイ・データベースを問い合せます。

```
SQL> SELECT PROCESS, STATUS, THREAD#, SEQUENCE#, BLOCK#, BLOCKS
  2> FROM V$MANAGED_STANDBY;

PROCESS STATUS       THREAD#    SEQUENCE#  BLOCK#     BLOCKS
------- ------------ ---------- ---------- ---------- ----------
RFS     ATTACHED     1          947        72         72
MRP0    APPLYING_LOG 1          946        10         72
```

この問合せ出力は、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、この出力は、アーカイブ REDO ログ順序番号 946 を適用している管理リカバリ操作も示しています。このリカバリ操作では現在、72 ブロックのアーカイブ REDO ログのブロック番号 10 をリカバリしている最中です。

V$ARCHIVE_DEST_STATUS 固定ビューへのアクセス

スタンバイ・データベースの同期のレベルを迅速に判断するには、フィジカル・スタンバイ・データベースで次の問合せを発行します。

```
SQL> SELECT ARCHIVED_THREAD#, ARCHIVED_SEQ#, APPLIED_THREAD#, APPLIED_SEQ#
  2> FROM V$ARCHIVE_DEST_STATUS;

ARCHIVED_THREAD# ARCHIVED_SEQ# APPLIED_THREAD# APPLIED_SEQ#
---------------- -------------- ---------------- ------------
1                947            1                945
```

この問合せ出力は、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 ログの適用がアーカイブ・ログ 2 つ分遅れていることを示しています。これは、単一のリカバリ処理では、受信しているアーカイブ REDO ログのボリュームに追いつくことができないことを示している可能性があります。これを解決するには、PARALLEL オプションを使用します。

V$ARCHIVED_LOG 固定ビューへのアクセス

フィジカル・スタンバイ・データベース上の V$ARCHIVED_LOG 固定ビューには、プライマリ・データベースから受信したすべてのアーカイブ REDO ログが表示されます。このビューが役に立つのは、スタンバイ・サイトがログの受信を開始した後のみです。それ以前のビューは、プライマリ制御ファイルから生成された旧アーカイブ REDO レコードによって移入されたものであるためです。たとえば、次の SQL*Plus 文を実行できます。

```
SQL> SELECT REGISTRAR, CREATOR, THREAD#, SEQUENCE#, FIRST_CHANGE#,
  2> NEXT_CHANGE# FROM V$ARCHIVED_LOG;

REGISTRAR CREATOR THREAD#    SEQUENCE#  FIRST_CHANGE# NEXT_CHANGE#
--------- ------- ---------- ---------- ------------- ------------
RFS       ARCH    1          945        74651         74739
RFS       ARCH    1          946        74739         74772
RFS       ARCH    1          947        74772         74774
```

この問合せ出力は、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 ログを示しています。

関連項目： 第 14 章の「V$ARCHIVED_LOG」

V$LOG_HISTORY 固定ビューへのアクセス

フィジカル・スタンバイ・データベースで V$LOG_HISTORY 固定ビューを問い合せると、適用されたすべてのアーカイブ REDO ログが表示されます。

```
SQL> SELECT THREAD#, SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE#
  2> FROM V$LOG_HISTORY;

THREAD#    SEQUENCE#  FIRST_CHANGE# NEXT_CHANGE#
---------- ---------- ------------- ------------
1          945        74651         74739
```

この問合せ出力は、最も最近適用されたアーカイブ REDO ログが順序番号 945 であったことを示しています。

V$DATAGUARD_STATUS 固定ビューへのアクセス

V$DATAGUARD_STATUS 固定ビューには、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されます。

次の例は、プライマリ・データベースでの V$DATAGUARD_STATUS ビューの出力を示します。

```
SQL> SELECT MESSAGE FROM V$DATAGUARD_STATUS;

MESSAGE
--------------------------------------------------------------------------------

ARC0: Archival started
ARC1: Archival started
Archivelog destination LOG_ARCHIVE_DEST_2 validated for no-data-loss
recovery
Creating archive destination LOG_ARCHIVE_DEST_2: 'dest2'
ARCH: Transmitting activation ID 0
LGWR: Completed archiving log 3 thread 1 sequence 11
Creating archive destination LOG_ARCHIVE_DEST_2: 'dest2'
LGWR: Transmitting activation ID 6877c1fe
LGWR: Beginning to archive log 4 thread 1 sequence 12
ARC0: Evaluating archive   log 3 thread 1 sequence 11
ARC0: Archive destination LOG_ARCHIVE_DEST_2: Previously completed
ARC0: Beginning to archive log 3 thread 1 sequence 11
Creating archive destination LOG_ARCHIVE_DEST_1:
'/oracle/arch/arch_1_11.arc'

ARC0: Completed archiving  log 3 thread 1 sequence 11
ARC1: Transmitting activation ID 6877c1fe

15 rows selected.
```

次の例は、フィジカル・スタンバイ・データベースでの V$DATAGUARD_STATUS ビューの内容を示します。

```
SQL> SELECT MESSAGE FROM V$DATAGUARD_STATUS;

MESSAGE
--------------------------------------------------------------------------------

ARC0: Archival started
ARC1: Archival started
RFS: Successfully opened standby logfile 6: '/oracle/dbs/sorl2.log'
```

```
ARC1: Evaluating archive   log 6 thread 1 sequence 11
ARC1: Beginning to archive log 6 thread 1 sequence 11
Creating archive destination LOG_ARCHIVE_DEST_1:
'/oracle/arch/arch_1_11.arc'

ARC1: Completed archiving  log 6 thread 1 sequence 11
RFS: Successfully opened standby logfile 5: '/oracle/dbs/sorl1.log'

Attempt to start background Managed Standby Recovery process
Media Recovery Log /oracle/arch/arch_1_9.arc

10 rows selected.
```

関連項目： 第 14 章の「V$DATAGUARD_STATUS」

ロジカル・スタンバイ・データベースに関するログ適用サービスの監視

ロジカル・スタンバイ・データベースのアーカイブ REDO ログの状況を監視し、ログ適用サービスに関する情報を取得するには、この項で説明する固定ビューを問い合せます。スタンバイ・データベースは、Data Guard Manager を使用して監視することもできます。

関連項目： 付録 A「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。

- DBA_LOGSTDBY_EVENTS ビューへのアクセス
- DBA_LOGSTDBY_LOG ビューへのアクセス
- DBA_LOGSTDBY_PROGRESS ビューへのアクセス
- V$LOGSTDBY 固定ビューへのアクセス
- V$LOGSTDBY_STATS 固定ビューへのアクセス

DBA_LOGSTDBY_EVENTS ビューへのアクセス

ログ適用サービスが突然停止した場合は、このビューにその原因が表示されます。

> **注意：** SQL 適用操作の停止原因となったエラーは、システム表領域に十分な領域があるかぎり、必ずイベント表に記録されます。これらのイベントは、常に ALERT.LOG ファイルにも格納され、テキストには、'LOGSTDBY event' という句が含まれます。このビューを問い合せたときは、EVENT_TIME、COMMIT_SCN、CURRENT_SCN の順で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。

このビューには、適用された DDL 文やスキップされた DDL 文などの他の情報も表示されます。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_TIME, COMMIT_SCN;

EVENT_TIME         STATUS
------------------------------------------------------------------------------
EVENT

--------------------------------------------------------------------------------
23-JUL-02 18:20:12 ORA-16111: ログのマイニングと設定の適用
23-JUL-02 18:20:12 ORA-16128: ユーザーが開始した停止は、正常に終了しました。
23-JUL-02 18:20:12 ORA-16112: ログのマイニングと停止の適用
23-JUL-02 18:20:23 ORA-16111: ログのマイニングと設定の適用
23-JUL-02 18:55:12 ORA-16128: ユーザーが開始した停止は、正常に終了しました。
23-JUL-02 18:57:09 ORA-16111: ログのマイニングと設定の適用
23-JUL-02 20:21:47 ORA-16204: DDL は正常に適用されました
create table mytable (one number, two varchar(30))
23-JUL-02 20:22:55 ORA-16205: DDL はスキップ設定のためスキップされました create database
link mydblink

8 rows selected.
```

この問合せは、ログ適用サービス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。ログ適用サービスが停止した場合は、問合せの最後のレコードに問題の原因が表示されます。

DBA_LOGSTDBY_LOG ビューへのアクセス

DBA_LOGSTDBY_LOG ビューには、ログ適用サービスの処理状況に関する動的な情報が表示されます。このビューは、アーカイブ REDO ログをロジカル・スタンバイ・データベースに適用するログ適用サービスのパフォーマンスに関する問題の診断に役立ちます。また、他の問題にも役立ちます。

次に例を示します。

```
SQL> SELECT FILE_NAME, SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE#,
  2> TIMESTAMP, DICT_BEGIN, DICT_END, THREAD# FROM DBA_LOGSTDBY_LOG
  3> ORDER BY SEQUENCE#;

FILE_NAME                 SEQ# FIRST_CHANGE# NEXT_CHANGE# TIMESTAM BEG END THR#
------------------------- ---- ------------- ------------ -------- --- --- ----
/oracle/dbs/hq_nyc_2.log  2           101579       101588 11:02:58 NO  NO  1
/oracle/dbs/hq_nyc_3.log  3           101588       142065 11:02:02 NO  NO  1
/oracle/dbs/hq_nyc_4.log  4           142065       142307 11:02:10 NO  NO  1
/oracle/dbs/hq_nyc_5.log  5           142307       142739 11:02:48 YES YES 1
/oracle/dbs/hq_nyc_6.log  6           142739       143973 12:02:10 NO  NO  1
/oracle/dbs/hq_nyc_7.log  7           143973       144042 01:02:11 NO  NO  1
/oracle/dbs/hq_nyc_8.log  8           144042       144051 01:02:01 NO  NO  1
/oracle/dbs/hq_nyc_9.log  9           144051       144054 01:02:16 NO  NO  1
/oracle/dbs/hq_nyc_10.log 10          144054       144057 01:02:21 NO  NO  1
/oracle/dbs/hq_nyc_11.log 11          144057       144060 01:02:26 NO  NO  1
/oracle/dbs/hq_nyc_12.log 12          144060       144089 01:02:30 NO  NO  1
/oracle/dbs/hq_nyc_13.log 13          144089       144147 01:02:41 NO  NO  1
```

この問合せ出力は、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・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。

DBA_LOGSTDBY_PROGRESS ビューへのアクセス

すべてのログ・ファイル情報が適用されたかどうかを迅速に判断するには、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ます。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM D BA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     211301     211357
```

APPLIED_SCN が NEWEST_SCN と一致している場合は、使用可能なすべてのログ情報が適用されています。使用可能なログ全体の進捗状況を調べるには、次の例に示すように、DBA_LOGSTDBY_PROGRESS ビューと DBA_LOGSTDBY_LOG ビューを結合します。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT  = 'DD-MON-YY HH24:MI:SS';
Session altered.

SQL> SELECT L.SEQUENCE#, L.FIRST_TIME,
  2    (CASE WHEN L.NEXT_CHANGE# < P.READ_SCN THEN 'YES'
  3          WHEN L.FIRST_CHANGE# < P.APPLIED_SCN THEN 'CURRENT'
  4          ELSE 'NO' END) APPLIED
  5  FROM DBA_LOGSTDBY_LOG L, DBA_LOGSTDBY_PROGRESS P
  6  ORDER BY SEQUENCE#;

 SEQUENCE# FIRST_TIME         APPLIED
---------- ------------------ -------
        24 23-JUL-02 18:19:05 YES
        25 23-JUL-02 18:19:48 YES
        26 23-JUL-02 18:19:51 YES
        27 23-JUL-02 18:19:54 YES
        28 23-JUL-02 18:19:59 YES
        29 23-JUL-02 18:20:03 YES
        30 23-JUL-02 18:20:13 YES
        31 23-JUL-02 18:20:18 YES
        32 23-JUL-02 18:20:21 YES
        33 23-JUL-02 18:32:11 YES
        34 23-JUL-02 18:32:19 CURRENT
        35 23-JUL-02 19:13:20 CURRENT
        36 23-JUL-02 19:13:43 CURRENT
        37 23-JUL-02 19:13:46 CURRENT
        38 23-JUL-02 19:13:50 CURRENT
        39 23-JUL-02 19:13:54 CURRENT
        40 23-JUL-02 19:14:01 CURRENT
        41 23-JUL-02 19:15:11 NO
        42 23-JUL-02 19:15:54 NO

19 rows selected.
```

この問合せでは、計算結果列の APPLIED 列に、YES、CURRENT または NO が表示されます。YES が表示されたログは完全に適用されているため、そのログ・ファイルはロジカル・スタンバイ・データベースでは不要です。CURRENT が表示されたログには、処理中の情報が含まれています。ロジカル・スタンバイがトランザクションを適用し、トランザクションが複数のログを使用しているため、通常、ログ適用サービスでは複数のログの変更を適用します。NO が表示されたログの場合、ログ・ファイルの情報は適用されていません。ただし、ファイルはオープンして読み込まれている可能性があります。

V$LOGSTDBY 固定ビューへのアクセス

SQL 適用操作のプロセス・アクティビティを調べるには、ロジカル・スタンバイ・データベースで V$LOGSTDBY 固定ビューを問い合せます。次に例を示します。

```
SQL> COLUMN STATUS FORMAT A50
SQL> COLUMN TYPE FORMAT A12
SQL> SELECT TYPE, HIGH_SCN, STATUS FROM V$LOGSTDBY;

TYPE           HIGH_SCN STATUS
------------ ---------- --------------------------------------------------
COORDINATOR             ORA-16117: 処理中です。
READER                  ORA-16127: 追加のトランザクションが適用されるまで待機して
                        停止しました。

BUILDER          191896 ORA-16116: 使用できる作業はありません
PREPARER         191902 ORA-16117: 処理中です。
ANALYZER         191820 ORA-16120: SCN 0x0000.0002ed4e でのトランザクションに対する
                        依存性を計算中です。

APPLIER          191209 ORA-16124: トランザクション 1 16 1598 は待機中です。

APPLIER          191205 ORA-16116: 使用できる作業はありません
APPLIER          191206 ORA-16124: トランザクション 1 5 1603 は待機中です。

APPLIER          191213 ORA-16117: 処理中です。
APPLIER          191212 ORA-16124: トランザクション 1 20 1601 は待機中です。

APPLIER          191216 ORA-16124: トランザクション 1 4 1602 は待機中です。

11 rows selected.
```

この問合せでは、REDO ログの読込みと適用に関係するプロセスごとに 1 行が表示されます。TYPE 列に示されているように、プロセスごとに異なる機能を実行します。HIGH_SCN 列は、進捗状況を示すインジケータです。このインジケータが問合せごとに変化している間は、処理が進行しています。STATUS 列には、アクティビティの説明テキストが表示されます。

V$LOGSTDBY_STATS 固定ビューへのアクセス

V$LOGSTDBY_STATS 固定ビューには、ログ適用サービスに関する一連の状態と統計情報が表示されます。ほとんどのオプションにはデフォルト値があり、このビューには現在使用されている値が表示されます。進捗を示す統計情報も表示されます。データベースの状態情報を表示するには、次の問合せを発行します。

```
SQL> COLUMN NAME FORMAT A35
SQL> COLUMN VALUE FORMAT A35
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS
  2> WHERE NAME LIKE 'coordinator%' or NAME LIKE 'transactions%';

NAME                                VALUE
----------------------------------- -----------------------------------
coordinator state                   APPLYING
transactions ready                  7821
transactions applied                7802
coordinator uptime                  73
```

この問合せでは、SQL 適用操作の実行時間とその間に適用されたトランザクションの数が表示されます。適用可能なトランザクションの数も表示され、さらに作業が必要なことを示しています。

アーカイブ・トレースの設定

スタンバイ・サイトへの REDO ログのアーカイブの進行状況を確認するには、プライマリおよびスタンバイ初期化パラメータ・ファイルに LOG_ARCHIVE_TRACE パラメータを設定します。LOG_ARCHIVE_TRACE パラメータを設定すると、Oracle データベース・サーバーは、次のように監査証跡をトレース・ファイルに書き込みます。

- プライマリ・データベースの場合

  Oracle データベース・サーバーは、プライマリ・データベース上のアーカイブ・プロセス・アクティビティ（ARC*n* プロセスとフォアグラウンド・プロセス）の監査証跡をトレース・ファイルに書き込みます。このトレース・ファイルのファイル名は、USER_DUMP_DEST 初期化パラメータで指定されます。

- スタンバイ・データベースの場合

  Oracle データベース・サーバーは、スタンバイ・データベース上のアーカイブ REDO ログに関連する RFS プロセスと ARC*n* プロセス・アクティビティの監査証跡をトレース・ファイルに書き込みます。このトレース・ファイルのファイル名は、USER_DUMP_DEST 初期化パラメータで指定されます。

トレース・ファイルの位置の判別

データベースのトレース・ファイルは、初期化パラメータ・ファイル内で USER_DUMP_DEST パラメータで指定されたディレクトリ内にあります。位置を判別するには、SQL*Plus を使用してプライマリおよびスタンバイ・インスタンスに接続し、SHOW 文を発行します。次に例を示します。

```
SQL> SHOW PARAMETER user_dump_dest
NAME                                 TYPE    VALUE
------------------------------------ ------- ------------------------------
user_dump_dest                       string  ?/rdbms/log
```

ログ・トレース・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
```

プライマリ・データベースで LOG_ARCHIVE_TRACE パラメータを使用可能、使用不可能にするか、または変更するには、次のいずれかを行います。

- プライマリ・データベースを停止し、初期化パラメータ・ファイルを修正して、データベースを再起動します。
- データベースがオープンまたはマウントされている間に、ALTER SYSTEM SET LOG_ARCHIVE_TRACE=*trace_level* 文を発行します。

読取り専用操作または管理リカバリ操作を実行しているフィジカル・スタンバイ・データベースで LOG_ARCHIVE_TRACE パラメータを使用可能、使用不可能または変更するには、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 に設定したので、6-27 ページの「整数値の選択」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・ログをプライマリ・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影響するように、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;
```

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	REDO ログ・ファイルのアーカイブを追跡します。
2	アーカイブ REDO ログ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 REDO 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 REDO 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 REDO 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	将来のリリースでサポートされます。
512	非同期 LGWR 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	ARC*n* または RFS のハートビートを追跡します。

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、REDO ログ 387 を 2 つの異なる宛先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。2 つの宛先とは、サービス standby1 とローカル・ディレクトリ /oracle/dbs です。

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、スタンバイ・サイトで、ディレクトリ /stby にアーカイブ・ログ 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るトレース・データを示します。

```
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
```

7

ロール管理

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。しかし、プライマリ・データベースが使用不可能になった場合、またはハードウェアやソフトウェアのメンテナンス操作の実行が必要になった場合は、構成内の 1 つ以上のデータベースのロールを変更する必要があります。

Data Guard 構成内のスタンバイ・データベースの数、位置、タイプ（フィジカルまたはロジカル）、およびプライマリ・データベースへの変更を各スタンバイ・データベースに伝播する方法に応じて、計画的および計画外のプライマリ・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事前に決まります。

この章では、Data Guard 構成内でデータベースのロールを変更および管理するための Data Guard のロール管理サービスと操作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

データベースは、相互に排他的な次の 2 つのロールのいずれかで実行されます。2 つのロールとは、**プライマリ**と**スタンバイ**です。Data Guard ではこれらのロールを、この章で説明する SQL コマンド、Data Guard Manager、あるいは Oracle Data Guard Broker のコマンドライン・インタフェース（『Oracle9*i* Data Guard Broker』を参照）を使用して動的に変更できます。

Oracle Data Guard は、ロールを推移するための 2 つの操作をサポートしています。1 つは**スイッチオーバー**操作で、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します。もう 1 つは**フェイルオーバー**操作で、プライマリ・データベース障害時にスタンバイ・データベースをプライマリ・ロールに推移します。

それぞれの操作の詳細は、7-4 ページの「スイッチオーバー操作」および 7-8 ページの「フェイルオーバー操作」で説明します。7-2 ページの「使用するロールの推移の選択」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、どのロールの推移操作を選択するかについて説明します。

注意： Oracle Data Guard のスイッチオーバー操作とフェイルオーバー操作は、自動的に起動しません。スイッチオーバー操作またはフェイルオーバー操作は、SQL 文または Data Guard Broker インタフェースを使用して手動で開始する必要があります。

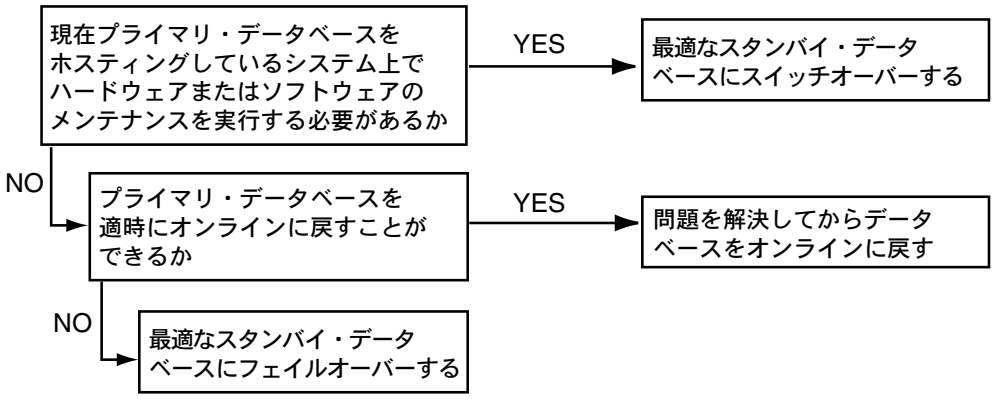
使用するロールの推移の選択

ロールを推移するとき、操作の完了までに必要な停止時間、データ消失の可能性、および構成内の他のスタンバイ・データベースへの影響は次の要因から判断します。

- 推移直前のプライマリ・データベースの状態
- ロールの推移の対象として選択されたスタンバイ・データベースの推移時の状態
- 選択されたスタンバイ・データベースがフィジカル・スタンバイ・データベースとして構成されているか、またはロジカル・スタンバイ・データベースとして構成されているか
- ロールの推移がスイッチオーバーか、またはフェイルオーバーか

目標は、データをまったく消失せず、可能なかぎり迅速にロールの推移を実行することです。図 7-1 に示す意思決定ツリーは、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を決定するのに役立ちます。

図 7-1 ロールの推移を選択するための意思決定ツリー



通常は、プライマリ・データベースの修正とロールの推移のどちらが速いかを検討します。プライマリ・データベースを修正できる場合は、クライアント・アプリケーションを新しいデータベースに接続するように再構成する必要はありません。ただし、修正操作によってデータが消失した場合は、修正したプライマリ・データベースのバックアップ・コピーから、構成内の他のすべてのスタンバイ・データベースを再作成する必要があります。

ロールの推移が必要で、構成内にフィジカル・スタンバイ・データベースが含まれる場合は、最適なフィジカル・スタンバイ・データベースを使用したロールの推移をお薦めします。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では、次の点を考慮してください。

- ロジカル・スタンバイ・データベースがプライマリ・データベースに存在するデータのサブセットのみメンテナンス対象にする構成の場合は、データが消失する可能性があります。
- 既存のフィジカル・スタンバイ・データベースは、ロールの推移後も Data Guard 構成に継続して含まれるように、新しいプライマリ・データベースのコピーから再作成する必要があります。

> **関連項目：** 最適なフィジカルまたはロジカル・スタンバイ・データベースを選択する方法については、10-2 ページの「ロールの推移に最適なスタンバイ・データベースの選択」を参照してください。

実行するロールの推移のタイプを決定した後、次のいずれかの項に進んでください。

- スイッチオーバーの場合は、7-4 ページの「スイッチオーバー操作」を参照してください。
- フェイルオーバーの場合は、7-8 ページの「フェイルオーバー操作」を参照してください。

スイッチオーバー操作

スイッチオーバー操作中、データの消失はなく、古いプライマリ・データベースは構成内にスタンバイ・データベースとして残ります。スイッチオーバーは、通常、計画的な停止（オペレーティング・システムまたはハードウェアのアップグレードなど）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。スイッチオーバー操作は、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 7-2 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 7-2 スイッチオーバー操作前の Data Guard 構成

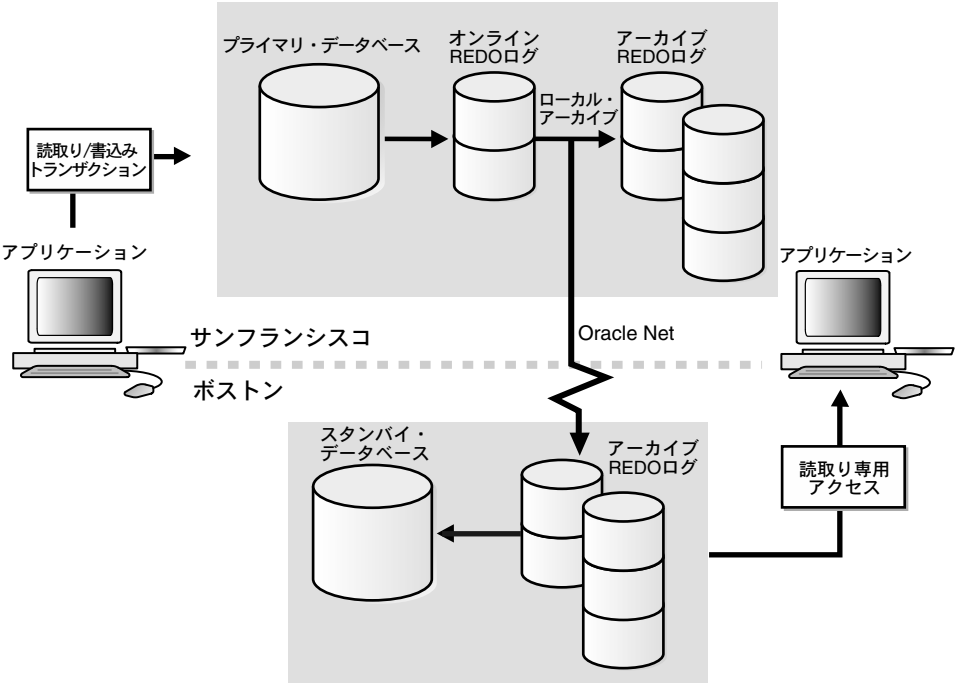


図 7-3 に、元のプライマリ・データベースがスタンバイ・データベースにスイッチオーバーされた後、元のスタンバイ・データベースがまだ新しいプライマリ・データベースになっていない Data Guard 環境を示します。このとき、Data Guard 構成には一時的に 2 つのスタンバイ・データベースが存在することになります。

図 7-3 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

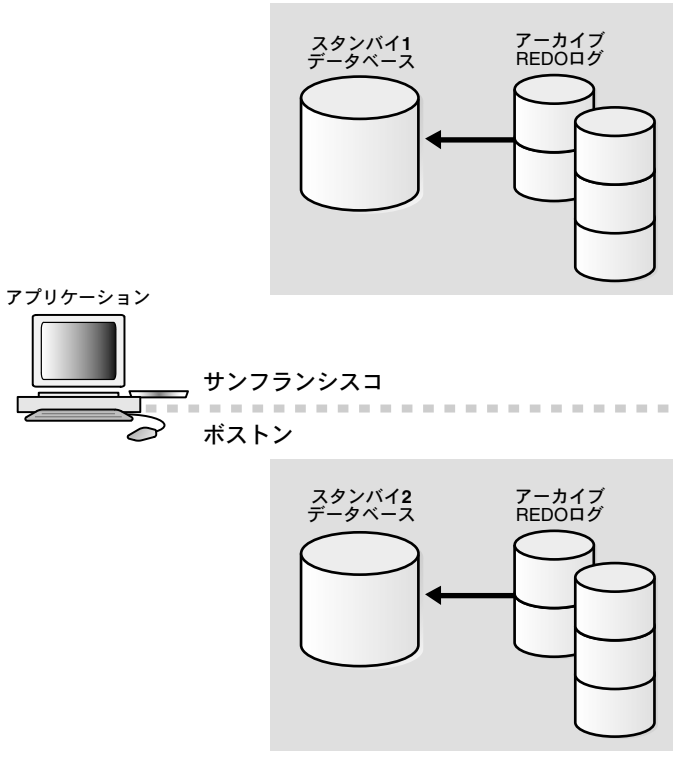
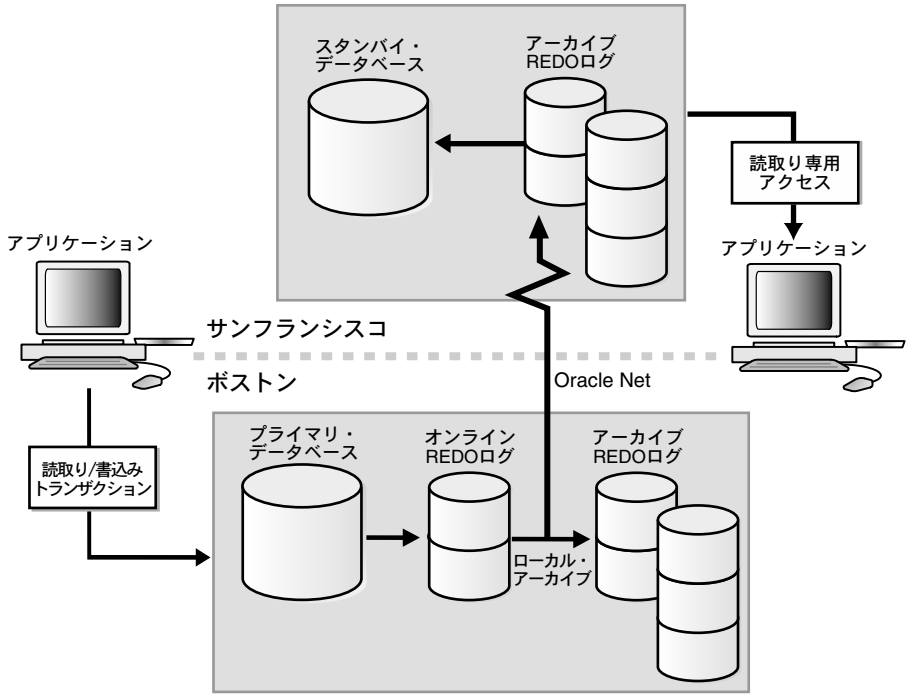


図 7-4 は、スイッチオーバー実行後の Data Guard 環境を示します。元のスタンバイ・データベースは新しいプライマリ・データベースになります。現在、プライマリ・データベースはボストンにあり、スタンバイ・データベースはサンフランシスコにあります。

図 7-4 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

スイッチオーバー操作は、Data Guard 構成内のプライマリ・データベースとロジカルまたはフィジカル・スタンバイ・データベースとの間で実行できますが、7-2 ページの「使用するロールの推移の選択」で説明したように、フィジカル・スタンバイ・データベースとの間で実行することをお薦めします。停止時間を最短にするために、各スイッチオーバー操作を慎重に計画して、スイッチオーバー操作に関与するプライマリおよびスタンバイ・データベースのトランザクション上の遅延が最小限になるようにします。

スイッチオーバー操作を開始する前に、次の作業を行ってください。

- ロールの推移を実行するために変更が必要な初期化パラメータを識別します。

注意： Data Guard Broker を使用しない場合は、すべてのスタンバイ・サイトで 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。これによって、スイッチオーバー操作またはフェイルオーバー操作が発生したとき、すべてのスタンバイ・サイトで新しいプライマリ・データベースからのログの受信を継続できます。Data Guard Broker のコマンドライン・インタフェースまたは Data Guard Manager で設定した構成では、LOG_ARCHIVE_DEST_*n* パラメータと LOG_ARCHIVE_DEST_STATE_*n* パラメータが自動的に定義されます。これには、プライマリ・データベースおよび他のすべてのスタンバイ・データベースを指し示すための LOG_ARCHIVE_DEST_*n* パラメータの定義も含まれます。

ロールの推移後に Data Guard 構成が正しく動作するように、5-28 ページの「ロールの推移に関する初期化パラメータの準備」を参照して、プライマリおよびスタンバイ・データベースの両方で初期化パラメータを構成します。

- プライマリ・データベースとスタンバイ・データベースの間にネットワーク接続があることを確認します。

  Data Guard 構成内の各位置で、Oracle Net を介してプライマリ・データベースおよび関連するすべてのスタンバイ・データベースに接続できる必要があります。

- データベースに接続しているアクティブなユーザーがいないことを確認します。
- Real Application Clusters 構成の場合は、1 つのプライマリ・インスタンスと 1 つのスタンバイ・インスタンスを除いて、構成内のすべてのインスタンスが停止していることを確認します。

  Real Application Clusters データベースの場合は、1 つのプライマリ・インスタンスと 1 つのスタンバイ・インスタンスのみがスイッチオーバー操作を実行できます。スイッチオーバー操作の前に、他のすべてのインスタンスを停止しておいてください。

- 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操作の場合は、プライマリ・データベース・インスタンスがオープンし、スタンバイ・データベース・インスタンスがマウントされていることを確認します。

  スイッチオーバー操作の開始前に、プライマリ・ロールに推移するスタンバイ・データベースをマウントする必要があります。また、データベースのロールが切り替えられたとき、フィジカル・スタンバイ・データベースがアーカイブ REDO ログをリカバリするのが理想的です。フィジカル・スタンバイ・データベースが読取り専用アクセスのためにオープンされている場合、スイッチオーバー操作は実行されますが時間がかかります。

  関連項目： 管理リカバリ・モードの詳細は、6-5 ページの「管理リカバリ操作の開始」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操作の場合は、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てい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・データベースを ARCHIVELOG モードにします。
- スタンバイ・データベースで現在有効になっている REDO データの適用遅延を解除します。

注意： プライマリ・データベースとスタンバイ・データベースのリリースは常に同じであるため、スイッチオーバー操作は、Oracle データベース・ソフトウェアのローリング・アップグレードを実行するために使用しないでください。ただし、システム・ハードウェアのローリング・アップグレードを実行するためにスイッチオーバー操作を使用することはできます。

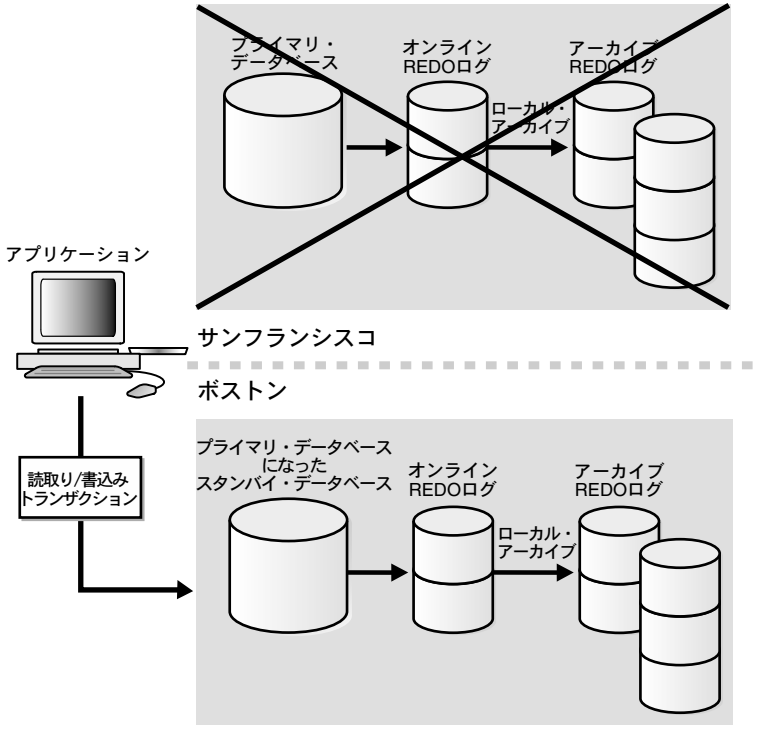
フィジカル・スタンバイ・データベースが関与するスイッチオーバー操作については、7-12 ページの「フィジカル・スタンバイ・データベースが関与するスイッチオーバー操作」を参照してください。ロジカル・スタンバイ・データベースが関与するスイッチオーバー操作については、7-20 ページの「ロジカル・スタンバイ・データベースが関与するスイッチオーバー操作」を参照してください。Oracle Data Guard Broker の分散管理フレームワークを構成している場合、Oracle Data Guard Manager の Switchover Wizard を使用してスイッチオーバー・プロセスを自動化する方法については、『Oracle9*i* Data Guard Broker』を参照してください。

フェイルオーバー操作

フェイルオーバー操作時、スタンバイ・データベースはプライマリ・ロールに推移し、古いプライマリ・データベースは構成に含まれなくなります。フェイルオーバー前に古いプライマリ・データベースがどの保護モードで実行されていたかによって、フェイルオーバー時に一部のデータが消失する場合があります。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、フェイルオーバー操作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 コマンドによって異なります。

図 7-5 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操作の結果を示します。

図 7-5 スタンバイ・データベースへのフェイルオーバー



注意： フェイルオーバー操作の実行後、次の手順を実行して、Data Guard 構成を元の状態にリストアすることもできます。

1. 新しいプライマリ・データベースのコピーを使用して、障害が発生したプライマリ・データベースを新しいスタンバイ・データベースとして再作成します。
2. 再作成したデータベースを新しいスタンバイ・データベースとして構成に追加します。
3. スイッチオーバーを実行してそのデータベースをプライマリ・ロールに推移させ、構成を障害が発生する前の元の状態にリストアします。

フェイルオーバーの準備

通常、フェイルオーバー操作を実行する前に、この項で説明する手順に従って、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。

フェイルオーバー操作を開始する前に、次の作業を行ってください。

- ロールの推移を実行するために変更が必要なパラメータを識別します。

  注意： Data Guard Broker を使用しない場合は、すべてのスタンバイ・サイトで 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。これによって、スイッチオーバー操作またはフェイルオーバー操作が発生したとき、すべてのスタンバイ・サイトで新しいプライマリ・データベースからのログの受信を継続できます。Data Guard コマンドライン・インタフェースまたは Data Guard Manager で設定した構成では、LOG_ARCHIVE_DEST_*n* パラメータと LOG_ARCHIVE_DEST_STATE_*n* パラメータが自動的に定義されます。これには、プライマリ・データベースおよび他のすべてのスタンバイ・データベースを指し示すための LOG_ARCHIVE_DEST_*n* パラメータの定義も含まれます。

  関連項目： プライマリ・データベースとスタンバイ・データベースの初期化パラメータ・ファイルの例は、5-28 ページの「ロールの推移に関する初期化パラメータの準備」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・データベースの間にネットワーク接続があることを確認します。

  Data Guard 構成内の各位置で、Oracle Net を介してプライマリ・データベースおよび関連するすべてのスタンバイ・データベースに接続できる必要があります。

- Real Application Clusters 構成の場合は、1 つのスタンバイ・インスタンスを除いて、構成内のすべてのインスタンスが停止していることを確認します。

  Real Application Clusters データベースの場合、フェイルオーバー操作中は 1 つのスタンバイ・インスタンスのみアクティブにできます。フェイルオーバー操作の前に、他のすべてのインスタンスを停止しておいてください。

- 最大保護モードで実行中のフィジカル・スタンバイ・データベースがフェイルオーバー操作に関与する場合は、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;
  ```

  最大保護モードに設定されているフィジカル・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。フェイルオーバー操作を実行すると、元のプライマリ・データベースは Data Guard 構成から削除されて元に戻せないため、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操作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、プライマリ・データベースをフィジカル・スタンバイ・データベースにフェイルオーバーしないでください。かわりに、読取り専用モードでスタンバイ・データベースをオープンし、データベースを問い合せて、プライマリ・データベースに対する更新がスタンバイ・データベースに伝播されていることを確認してください。

フィジカル・スタンバイ・データベースが関与するフェイルオーバー操作を実行する場合は、7-15 ページの「フィジカル・スタンバイ・データベースが関与するフェイルオーバー操作」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フェイルオーバー操作を実行する場合は、7-23 ページの「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操作」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・データベースが関与するスイッチオーバー操作とフェイルオーバーの実行方法、および失敗したスイッチオーバーのリカバリ方法を説明します。

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操作

この項では、プライマリ・データベースとフィジカル・スタンバイ・データベースの間でロールを変更するスイッチオーバー操作の実行方法を説明します。スイッチオーバー操作は、常にプライマリ・データベースで開始し、フィジカル・スタンバイ・データベースで完了してください。次に、スイッチオーバー操作を実行する手順を説明します。

現行のプライマリ・データベースでの手順

手順 1 スイッチオーバー操作を実行できるかどうかを確認する

スイッチオーバー操作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 は、プライマリ・データベースのスタンバイ・ロールへの切替えが可能であることを示します。値 TO STANDBY 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）。

> **関連項目：** V$DATABASE ビューの SWITCHOVER_STATUS 列に対するその他の有効な値については、第 14 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操作を開始する

現行のプライマリ・データベースをフィジカル・スタンバイ・データベース・ロールに推移するには、プライマリ・データベースで次の SQL 文を使用します。

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

この文が完了すると、プライマリ・データベース・ロールはスタンバイ・データベースに変換されます。現行の制御ファイルは、スイッチオーバー操作の前にカレント 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。

手順 3 元のプライマリ・インスタンスを停止して再起動する

元のプライマリ・インスタンスを停止し、データベースをマウントせずに再起動します。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP NOMOUNT;
```

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マウントします。

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

スイッチオーバー・プロセスのこの時点では、両方のデータベースがスタンバイ・データベースとして構成されています（図 7-3 を参照）。

ターゲット・フィジカル・スタンバイ・データベースでの手順

手順 4 スイッチオーバーの状態を V$DATABASE ビューで確認する

プライマリ・データベースをフィジカル・スタンバイ・ロールに推移させ、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 列を問い合せて、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。

次に例を示します。

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

SWITCHOVER_STATUS 列の値 TO PRIMARY は、スタンバイ・データベースがスタンバイ・ロールからプライマリ・ロールへすぐに切り替えられることを示します。値 TO PRIMARY 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）。

> **関連項目：** V$DATABASE ビューの SWITCHOVER_STATUS 列に対するその他の有効な値については、第 14 章を参照してください。

手順 5 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

フィジカル・スタンバイ・データベースをスタンバイ・ロールからプライマリ・ロールに切り替えることができるのは、そのスタンバイ・データベースのインスタンスが管理リカバリ・モードでマウントされているか、あるいは読取り専用アクセスのためにオープンされているときです。フィジカル・スタンバイ・データベースは、いずれかのモードでマウントする必要があります。これによって、プライマリ・データベースのスイッチオーバー操作要求を調整できます。

SQL ALTER DATABASE 文を使用してスイッチオーバーを実行すると、オンライン REDO ログ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ります。したがって、ターゲット・スタンバイ・データベースを作成するときは、常にオンライン REDO ログをターゲット・スタンバイ・データベースに手動で追加しておくことをお薦めします。オンライン REDO ログが存在しない場合は、次のいずれかの方法で手動で追加します。

- 既存のオンライン REDO ログを元のプライマリ・データベース・サイトからターゲット・スタンバイ・データベース・サイトにコピーし、スタンバイ・サイトのパス名を新しいオンライン REDO ログに正しく対応付けるように LOG_FILE_NAME_CONVERT 初期化パラメータを定義します（3-6 ページの「フィジカル・スタンバイ・データベースでの初期化パラメータの設定」を参照）。
- ターゲット・スタンバイ・サイトの既存のオンライン REDO ログを削除し、ALTER DATABASE ADD STANDBY LOGFILE 文を使用して新しいオンライン REDO ログを作成します。

手動でオンライン REDO ログを追加した後、プライマリ・ロールに推移するフィジカル・スタンバイ・データベースで次の SQL 文を使用します。

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

手順 6 新しいプライマリ・データベースを停止して再起動する

ターゲット・スタンバイ・インスタンスを停止し、プライマリ・ロールの適切な初期化パラメータを使用して再起動します。

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

ターゲット・フィジカル・スタンバイ・データベースがプライマリ・データベース・ロールに推移します。

注意： スイッチオーバー操作時にオンラインでもスイッチオーバーに関与しない他のスタンバイ・データベースは、停止して再起動する必要はありません。これらのスタンバイ・データベースは、スイッチオーバーの完了後も正常に機能し続けます。

新しいフィジカル・スタンバイ・データベースでの手順

手順 7 管理リカバリ操作とログ適用サービスを開始する

次の文を発行して、新しいフィジカル・スタンバイ・データベースで管理リカバリ操作を開始します。

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

新しいプライマリ・データベースでの手順

手順 8 スタンバイ・データベースへの REDO データの送信を開始する

新しいプライマリ・データベースで次の文を発行します。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

フィジカル・スタンバイ・データベースが関与するフェイルオーバー操作

この項では、フィジカル・スタンバイ・データベースが関与するフェイルオーバー操作の実行方法を説明します。

フィジカル・スタンバイ・データベースが関与するフェイルオーバー操作時に、次の処理が実行されます。

- すべての場合、元のプライマリ・データベースは Data Guard 構成から削除されます。
- ほとんどの場合、フェイルオーバー操作に直接関与しない他のロジカルまたはフィジカル・スタンバイ・データベースは構成内に残り、停止して再起動する必要はありません。
- 新しいプライマリ・データベースを構成した後、すべてのスタンバイ・データベースの再作成が必要な場合があります。

フェイルオーバー操作を開始する前に、7-10 ページの「フェイルオーバーの準備」で説明した手順を可能なかぎり実行し、選択したスタンバイ・データベースのフェイルオーバー操作の準備をしてから、「フェイルオーバーの手順」に進んでください。

フェイルオーバーの手順

次に、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を実行する手順を説明します。使用しているデータ保護モードに従って、コミットされたすべてのトランザクションを自動的にリカバリすることも可能です。

この項で説明する手順では、選択したフィジカル・スタンバイ・データベースがプライマリ・ロールに推移するため、構成内の他のフィジカルまたはロジカル・スタンバイ・データベースは構成内に残り、停止して再起動する必要はありません。

手順 1 アーカイブ REDO ログのギャップを識別して解決する

ターゲット・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 ログにギャップが存在するかどうかを判断するには、V$ARCHIVE_GAP ビューを問い合せます。このビューには、各スレッドで欠落しているアーカイブ・ログの順序番号が表示されます。戻されるデータは、順序番号が最も大きいギャップのみです （手順 3 で、他のギャップの解決方法を説明します)。

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 のアーカイブ・ログ 90、91 および 92 がギャップです。可能であれば、欠落が識別されたすべてのアーカイブ REDO ログを、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。この処理は、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 欠落した他のアーカイブ REDO ログをコピーする

欠落したアーカイブ REDO ログが他に存在するかどうかを判断するには、構成内で使用可能なすべてのデータベースで 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
```

ターゲット・スタンバイ・データベースで最も大きい順序番号より大きい順序番号を含む別の使用可能なデータベースから、アーカイブ REDO ログをターゲット・スタンバイ・データベースにコピーして登録します。この処理は、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';
```

注意： 前述の手順を実行すると、登録されるのは部分的なアーカイブ REDO ログになる場合があります。部分的なアーカイブ REDO ログには、プライマリ・データベースに障害が発生したときにスタンバイ・データベースがプライマリ・データベースから受信したすべての REDO データが含まれますが、アーカイブ REDO ログはスタンバイ・データベースに自動的には登録されません。

部分的なアーカイブ・ログを登録すると、スタンバイ REDO ログのリカバリができなくなります。したがって、部分的なアーカイブ・ログを登録するかどうかによって、使用するフェイルオーバー・コマンドが決まります。部分的なアーカイブ・ログを手動で登録する場合は、次のメッセージがアラート・ログに表示されます。

```
Register archivelog 'filespec1' was created due to a
network disconnect; archivelog contents are valid but
missing subsequent data
```

手順 3 手順 1 ～ 2 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。そのギャップを解決した後、手順 1 の問合せで行が戻されなくなるまで、手順 1 ～ 2 を繰り返します。

手順 4 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操作を開始する

ターゲット・スタンバイ・データベースにスタンバイ REDO ログが構成されていて、部分的なアーカイブ REDO ログを手動で登録していない場合は、次の文を発行します。

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

それ以外の場合は、次の文を発行する必要があります。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH
  2> SKIP STANDBY LOGFILE;
```

手順 5 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・ロールに変換する

SQL ALTER DATABASE RECOVER MANAGED STANDBY DATABASE...FINISH 文が正常に完了した後、次の SQL 文を発行して、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・ロールに推移させます。

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

この SQL 文を発行した後、このデータベースはスタンバイ・データベースとして使用できなくなり、元のプライマリ・データベースからの後続の REDO ログは適用できません。スタンバイ REDO ログをアーカイブした後、元のプライマリ・データベースから作成した他のすべてのスタンバイ・データベースにコピーして登録し、リカバリしてください。この処理は、新しいプライマリ・データベースでスタンバイ宛先を適切に定義すると、自動的に実行されます。

フェイルオーバー操作に関与しない構成内の他のスタンバイ・データベースは、停止して再起動する必要はありません。フェイルオーバー時に、元のプライマリ・データベースは削除されて構成に含まれなくなります。古いプライマリ・データベースを新しい構成内で再利用するには、新しいプライマリ・データベースのバックアップ・コピーを使用し、スタンバイ・データベースとして再作成する必要があります。

プライマリ・データベースおよび残りの全スタンバイ・データベースでの手順

手順 6 新しいプライマリ・データベースから REDO ログを受信する準備をする

アーカイブ・スタンバイ REDO ログをすべてのスタンバイ宛先で受信してリカバリすると、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・データベースから REDO ログを受信する準備ができたことになります。新しいプライマリ・データベースで他のスタンバイ・データベースのアーカイブ・ログの宛先が定義されていない場合は、アーカイブ・ログ宛先を定義して使用可能にする必要があります。さらに、構成内の残りのスタンバイ・データベースごとに、フェイルオーバーの結果として生じるアーカイブ REDO ログを手動でコピーして登録する必要があります。

自動的に適用されなかったアーカイブ REDO ログをコピーした場合、コピーしたアーカイブ REDO ログを手動で登録するには、コピーしたファイルごとに各スタンバイ・データベースで次の文を発行します。

```
SQL> ALTER DATABASE REGISTER LOGFILE 'filespec';
```

新しいプライマリ・データベースでの手順

手順 7 新しいプライマリ・データベースを停止して再起動する

フェイルオーバー操作を完了するには、プライマリ・ロールに対して適切な従来の初期化パラメータ・ファイル（またはサーバー・パラメータ・ファイル）を使用して、新しいプライマリ・データベースを停止して読取り / 書込みモードで再起動する必要があります。

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

> **関連項目：** ロールの推移後に Data Guard 構成が正しく動作するように、プライマリおよびスタンバイ・データベースの両方で初期化パラメータを構成する方法については、5-28 ページの「ロールの推移に関する初期化パラメータの準備」を参照してください。

手順 8 新しいプライマリ・データベースをバックアップする（オプション）

STARTUP 文を発行する前に、必要に応じて新しいプライマリ・データベースをバックアップできます。この作業は、必要がない場合にも推奨される安全な手段です。これは、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。

フェイルオーバー操作の結果、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。

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操作およびフェイルオーバー操作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スイッチオーバー操作

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操作を実行する場合、スイッチオーバー操作は、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。次に、スイッチオーバー操作を実行する手順を説明します。

注意： 現在、V$DATABASE ビューの SWITCHOVER_STATUS 列は、フィジカル・スタンバイ・データベースで使用する場合のみサポートされているため、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操作時に問い合せることはできません。

元のプライマリ・データベースでの手順

手順 1 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・データベースをロジカル・スタンバイ・データベース・ロールに推移するには、次の SQL 文を発行します。

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

この文によって、プライマリ・データベースでの現在の更新処理が終了するまで待機し、新規ユーザーによる更新処理の開始を防止します。また、ロジカル・スタンバイ・データベース処理の同期点を示すために、REDO ログ・ファイルにマーカーを付けます。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）。

プライマリ・データベースが推移して、スタンバイ・データベース・ロールが実行されます。

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 2 アーカイブ REDO ログの宛先を変更する

すべてのリモート・データベースのアーカイブ REDO ログの宛先を変更します。

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

後で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 REDO ログ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 REDO ログのアーカイブを不可能にする必要があります。

元のロジカル・スタンバイ・データベースでの手順

手順 3 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・ロールにするロジカル・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。他の既存のロジカル・スタンバイ・データベースは、スイッチオーバー操作の完了後も正常に機能し続けます。ただし、既存のすべてのフィジカル・スタンバイ・データベースは、スイッチオーバー後は Data Guard 構成に含まれなくなります。

手順 4 アーカイブ REDO ログを使用可能にする

新しいロジカル・スタンバイ・データベースおよび他のすべてのリモート・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識別し、宛先ごとに REDO ログのアーカイブを可能にします。

> **関連項目：** ロールの推移後もすべてのスタンバイ・ロケーションで REDO データの受信を継続するように、各データベースの LOG_ARCHIVE_DEST_*n* および LOG_ARCHIVE_DEST_STATE_*n* 初期化パラメータを指定する方法については、7-6 ページの「スイッチオーバーの準備」を参照してください。

たとえば、LOG_ARCHIVE_DEST_2 パラメータで定義したリモート宛先への REDO ログ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、適切な従来の初期化パラメータ・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 REDO ログ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 REDO ログのアーカイブを不可能にする必要があります。

すべてのロジカル・スタンバイ・データベースでの手順

手順 5 新しいプライマリ・データベースへのデータベース・リンクを作成する

各ロジカル・スタンバイ・データベース（元のプライマリ・データベースおよびすべての既存のロジカル・スタンバイ・データベースを含む）で、次の手順に従って、新しいプライマリ・データベースへのデータベース・リンクを定義します。

1. 各ロジカル・スタンバイ・データベースで、新しいプライマリ・データベースを指すデータベース・リンクを作成します （この手順の例では、データベース・リンク `location1` を使用します）。

   `DBMS_LOGSTDBY.GUARD_BYPASS_ON` プロシージャ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・データベース内の表に対する変更を可能にします。次に例を示します。

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
   SQL> CREATE DATABASE LINK location1
     2> CONNECT TO user-name IDENTIFIED BY password USING 'location1';
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
   ```

   `CREATE DATABASE LINK` 文で指定したデータベース・ユーザー・アカウントには、新しいプライマリ・データベースに対する `SELECT_CATALOG_ROLE` ロールが付与されている必要があります。

   > **関連項目：** `DBMS_LOGSTDBY` パッケージの詳細は『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を、データベース・リンクの作成方法の詳細は『Oracle9*i* データベース管理者ガイド』を参照してください。

2. データベース・リンクを確認します。

   各ロジカル・スタンバイ・データベースで、データベース・リンクを使用して次の問合せを実行し、データベース・リンクが正しく構成されていることを確認します。

   ```
   SQL> SELECT * FROM DBA_LOGSTDBY_PARAMETERS@location1;
   ```

   問合せが成功した場合は、手順 1 で作成したデータベース・リンクを使用してスイッチオーバーを完了できます。

手順 6 SQL 適用操作を開始する

新しいロジカル・スタンバイ・データベース（元のプライマリ・データベース）および他の既存のロジカル・スタンバイ宛先で、SQL 適用操作を開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY location1;
```

この例では、location1 が新しいプライマリ・データベースへのデータベース・リンクです。

新しいプライマリ・データベースでの手順

手順 7 すべてのスタンバイ・データベースで REDO ログの受信を開始する

すべてのスタンバイ・データベースで REDO ログの受信を開始できるように、新しいプライマリ・データベースで次の SQL 文を実行して、アーカイブ・ロギングを使用可能にし、ログを切り替えます。

```
SQL> ALTER SYSTEM ARCHIVE LOG START;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

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操作

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操作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操作時に、次の処理が実行されます。

- すべての場合、元のプライマリ・データベースおよびすべてのフィジカル・スタンバイ・データベースは Data Guard 構成から削除されます。
- ほとんどの場合、フェイルオーバー操作に直接関与しない他のロジカル・スタンバイ・データベースは構成内に残り、停止して再起動する必要はありません。
- 新しいプライマリ・データベースを構成した後、すべてのスタンバイ・データベースの再作成が必要な場合があります。

フェイルオーバー操作を開始する前に、7-10 ページの「フェイルオーバーの準備」で説明した手順を可能なかぎり実行し、選択したスタンバイ・データベースのフェイルオーバー操作の準備をしてください。構成に対する保護モードおよびログ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。

フェイルオーバー操作を開始するには、次の手順を実行します。

プライマリ・ロールに推移するロジカル・スタンバイ・データベースでの手順

手順 1 欠落したアーカイブ REDO ログをコピーして登録する

緊急時の状態によって、プライマリ・データベースまたは他のスタンバイ・データベースのアーカイブ REDO ログにアクセスします。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. 欠落しているログを、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. コピーしたログを登録します。

ロジカル・スタンバイ・データベースで DBA_LOGSTDBY_LOG ビューを問い合せて、欠落しているログを判断して登録します。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
```

アーカイブ REDO ログのギャップを解決するには、順序番号 7 と 11 のアーカイブ REDO ログをコピーします（問合せで表示された各行より順序番号が 1 つ後のファイルをコピーします)。次に、それらのアーカイブ REDO ログをロジカル・スタンバイ・データベースに登録します。次に例を示します。

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

SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
  2> '/disk1/oracle/dbs/log-1292880008_11.arc';
Database altered.
```

欠落したアーカイブ REDO ログをコピーしてロジカル・スタンバイ・システムに登録した後、DBA_LOGSTDBY_LOG ビューを再度問い合せて、ギャップがないこと、およびロジカル・スタンバイ・データベースで必要な次のスレッドと順序番号が存在しないことを確認します。

手順 2 オンライン REDO ログをプライマリ・データベースからコピーして登録する

緊急時の状態によっては、オンライン REDO ログがプライマリ・データベースで使用可能なままになっています。その場合、欠落したオンライン REDO ログをプライマリ・データベースからコピーして登録し、次にロジカル・スタンバイ・データベースに適用します。

すでにアーカイブされてロジカル・スタンバイ・データベースに登録されているオンライン REDO ログを登録すると、ORA-01289 メッセージが表示されます。このエラー・メッセージは無視しても安全です。次に例を示します。

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
  2> '/disk1/oracle/dbs/online_log1.log';
ALTER DATABASE REGISTER LOGICAL LOGFILE '/disk1/oracle/dbs/online_log1.log'
*
ERROR at line 1:
ORA-01289: 重複するログ・ファイルを追加できません。

SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
  2> '/disk1/oracle/dbs/online_log2.log';
Database altered.
```

手順 3 部分的なアーカイブ REDO ログを登録する（存在する場合）

緊急時の状態によっては、プライマリ・データベースのファイルにアクセスできないこともあります。部分的なアーカイブ REDO ログを検索するには、他のスタンバイ REDO ログが存在するディレクトリのロジカル・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。部分的なアーカイブ REDO ログが存在する場合、その順序番号は、最後に登録されたアーカイブ REDO ログより 1 つ大きい順序番号になります。次に例を示します。

```
SQL> COLUMN FILE_NAME FORMAT a55
SQL> SELECT THREAD#, SEQUENCE#, FILE_NAME FROM DBA_LOGSTDBY_LOG L
  2> ORDER BY THREAD#,SEQUENCE#;

   THREAD#  SEQUENCE# FILE_NAME
---------- ---------- -------------------------------------------------------
         1          3 /disk1/oracle/dbs/db1loga-1292880008_3.arc
         1          4 /disk1/oracle/dbs/archlogb-1292880008_4.arc
         1          5 /disk1/oracle/dbs/archlogb-1292880008_5.arc
         1          6 /disk1/oracle/dbs/archlogb-1292880008_6.arc
         1          7 /disk1/oracle/dbs/archlogb-1292880008_7.arc
         1          8 /disk1/oracle/dbs/archlogb-1292880008_8.arc
```

```
         1          9 /disk1/oracle/dbs/archlogb-1292880008_9.arc
         1         10 /disk1/oracle/dbs/archlogb-1292880008_10.arc

8 rows selected.
```

部分的なアーカイブ・ログが存在する場合は、そのアーカイブ・ログを登録します。次に例を示します。

```
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE
  2> '/disk1/oracle/dbs/log-1292880008_11.arc';

Database altered.
```

手順 4 適用遅延間隔をオフにする

適用遅延間隔をオフにするには、ログ適用サービスを停止し、DBMS_LOGSTDBY.APPLY_UNSET プロシージャを実行します。次に、ログ適用サービスを再開します。

当初は適用遅延間隔をプライマリ・データベースで設定しましたが、プライマリ・データベースが使用不可能になったため、次の文をロジカル・スタンバイ・データベースで発行して適用遅延間隔を無効にする必要があります。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered.

SQL> EXECUTE DBMS_LOGSTDBY.APPLY_UNSET('APPLY_DELAY');
PL/SQL procedure successfully completed.

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
Database altered.
```

手順 5 すべての REDO ログが適用されたことを確認する

プライマリ・ロールに推移させるロジカル・スタンバイ・データベースで DBA_LOGSTDBY_PROGRESS ビューを問い合せて、残りのアーカイブ REDO ログが適用されたことを確認します。次に例を示します。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     190725     190725
```

APPLIED_SCN と NEWEST_SCN の値が等しい場合は、取得可能なすべてのデータが適用され、ロジカル・スタンバイ・データベースにはプライマリ・データベースからのデータが可能なかぎり含まれています。

> **関連項目：** DBA_LOGSTDBY_PROGRESS ビューについては、第 9 章および第 10 章を参照してください。

手順 6 新しいプライマリ・データベースをアクティブにする

新しいプライマリ・ロールに推移させるロジカル・スタンバイ・データベースで次の文を発行して SQL 適用操作を停止し、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にします。

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

他のすべてのロジカル・スタンバイ・データベースでの手順

手順 1 他のスタンバイ・データベースをリカバリする

新しいプライマリ・データベースに適用された REDO データの量に従って、既存のロジカル・スタンバイ・データベースを Data Guard 構成に再度追加して新しいプライマリ・データベースのスタンバイ・データベースとして使用できる場合とできない場合があります。

手順 2 他のスタンバイ・データベースから新しいプライマリ・データベースへのデータベース・リンクを作成する

次の手順に従って、将来のスイッチオーバー操作時に使用する新しいプライマリ・データベースへのデータベース・リンクを定義します。

1. 各ロジカル・スタンバイ・データベースでデータベース・リンクを作成します。

   DBMS_LOGSTDBY.GUARD_BYPASS_ON プロシージャ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・データベース内の表に対する変更を可能にします。次に例を示します。

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
   SQL> CREATE DATABASE LINK location1
     2> CONNECT TO <user-name> IDENTIFIED BY <password> USING 'location1';
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
   ```

   CREATE DATABASE LINK 文で指定したデータベース・ユーザー・アカウントには、プライマリ・データベースに対する SELECT_CATALOG_ROLE ロールが付与されている必要があります。

   > **関連項目：** DBMS_LOGSTDBY パッケージの詳細は『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を、データベース・リンクの作成方法の詳細は『Oracle9*i* データベース管理者ガイド』を参照してください。

2. データベース・リンクを確認します。

   ロジカル・スタンバイ・データベースで、データベース・リンクを使用して次の問合せを実行し、データベース・リンクが正しく構成されていることを確認します。

   ```
   SQL> SELECT * FROM DBA_LOGSTDBY_PARAMETERS@location1;
   ```

   問合せが成功した場合は、手順 1 で作成したデータベース・リンクを使用してスイッチオーバーを実行できることを示します。

新しいプライマリ・データベースでの手順

すべてのロジカル・スタンバイのリモート宛先への REDO ログのアーカイブを可能にします。次に例を示します。

```
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 REDO ログ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 REDO ログのアーカイブを不可能にする必要があります。

すべてのロジカル・スタンバイ・データベースでの手順

すべてのロジカル・スタンバイ・データベースで次の SQL 文を発行して、ログ適用サービスを開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY location1;
```

この文が完了すると、残りのすべてのアーカイブ REDO ログが適用されます。この操作は、処理量によって完了までに時間がかかる場合があります。

ORA-16109 エラーが表示された場合は、新しいプライマリ・データベースのバックアップ・コピーからロジカル・スタンバイ・データベースを再作成して Data Guard 構成に追加する必要があります。

次に、新しい構成内のロジカル・スタンバイ・データベースでログ適用サービスの開始に失敗した例を示します。ここで、`location1` は新しいプライマリ・データベースを指しています。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY location1;
ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY location1
                                                       *
ERROR at line 1:
ORA-16109: 以前のプライマリからのログ・データの適用に失敗しました
```

8

フィジカル・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・データベースの管理方法について説明します。Data Guard は、多くの方法でフィジカル・スタンバイ・データベースを容易に管理、操作および変更する手段を提供します。

この章は、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

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手順を説明します。

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、管理者権限で SQL*Plus を使用してデータベースに接続し、SQL*Plus STARTUP コマンドに NOMOUNT オプションを指定して使用します （スタンバイ・データベースでは NOMOUNT オプションを使用する必要があります）。

プライマリおよびスタンバイ・データベースの両方がオフラインの場合は、常に（可能なかぎり）プライマリ・データベースを起動する前にスタンバイ・データベースを起動します。

データベースの起動後、そのデータベースをスタンバイ・データベースとしてマウントします。マウントされたデータベースは、プライマリ・データベースからアーカイブ REDO データを受信できます。

次に、管理リカバリ操作を開始するか、またはデータベースを読取り専用アクセス用にオープンします。通常は、管理リカバリ操作を開始します。次の例で、スタンバイ・データベースの起動方法を示します。

1. データベースを起動します。

```
SQL> STARTUP NOMOUNT;
```

2. 次のように入力してスタンバイ・データベースをマウントします。

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

3. 管理リカバリ操作を開始します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
  2> DISCONNECT FROM SESSION;
```

データベースで管理リカバリを実行すると、ログ適用サービスによってアーカイブ REDO ログがスタンバイ・データベースに適用されます。

> **関連項目：** 管理リカバリについては 6-5 ページの「管理リカバリ操作の開始」を、スタンバイ・データベースを読取り専用アクセス用にオープンする方法については 8-3 ページの「読取り専用アクセス用にオープンしたスタンバイ・データベースの使用」を参照してください。

フィジカル・スタンバイ・データベースの停止

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。データベースが管理リカバリを実行している場合は、SHUTDOWN コマンドを発行する前に管理リカバリ操作を取り消す必要があります。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実行中の場合は、スタンバイ・データベースを停止する前に、プライマリ・データベースでアーカイブ・ログの宛先を遅延し、遅延操作を有効にするためにログ・スイッチ操作を実行します。これを実行しないと、ログ転送サービスは REDO データをこのスタンバイ・サイトに転送できなくなります。

スタンバイ・データベースを停止するには、次の手順を実行します。

1. スタンバイ・データベースが管理リカバリを実行しているかどうかを識別します。MRP0 または MRP プロセスが存在する場合は、スタンバイ・データベースが管理リカバリを実行しています。

```
SQL> SELECT PROCESS, STATUS FROM V$MANAGED_STANDBY;
```

2. 管理リカバリ操作を取り消します。

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

3. スタンバイ・データベースを停止します。

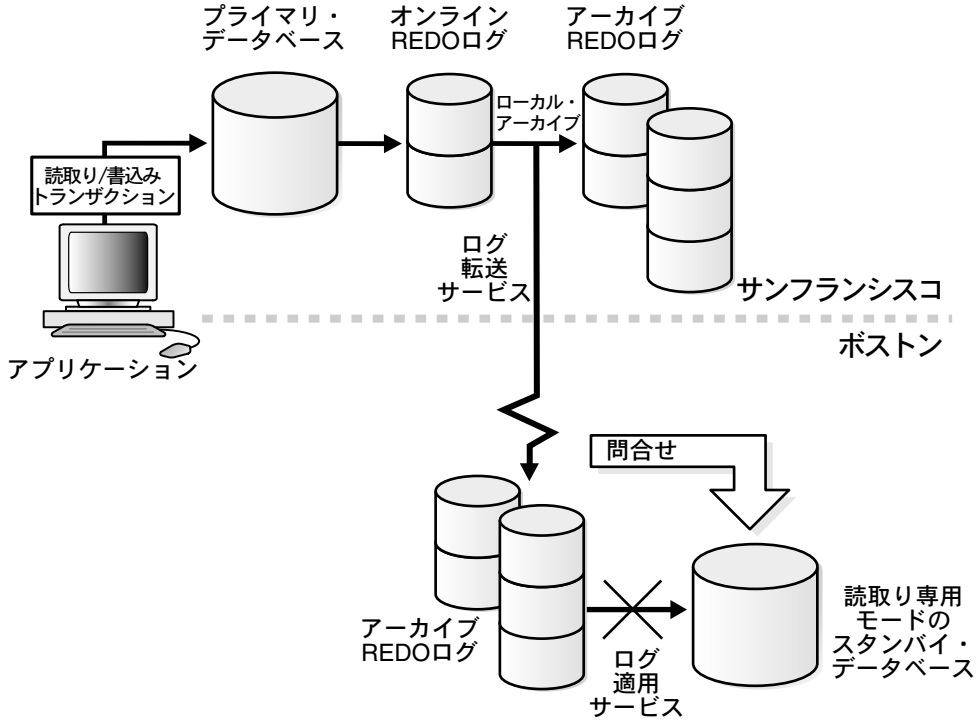
```
SQL> SHUTDOWN IMMEDIATE;
```

読取り専用アクセス用にオープンしたスタンバイ・データベースの使用

スタンバイ・データベースが読取り専用アクセス用にオープンしている場合、ユーザーは、オンライン・データ変更が行われることなく、スタンバイ・データベースを問い合せることができます。レポートを生成するためにスタンバイ・データベースを使用すると、プライマリ・データベース負荷を低減できます。スタンバイ・データベースを読取り専用アクセス用に定期的にオープンし、ログ適用サービスがスタンバイ・データベースを適切に更新していることを確認するために、非定型問合せを実行できます。

図 8-1 に、読取り専用アクセス用にオープンしたスタンバイ・データベースを示します。

図 8-1 読取り専用アクセス用にオープンしたスタンバイ・データベース



この項は、次の項目で構成されています。

- 読取り専用アクセス用にスタンバイ・データベースをオープンするかどうかの評価
- 読取り専用アクセス用にスタンバイ・データベースをオープン

読取り専用アクセス用にスタンバイ・データベースをオープンするかどうかの評価

フィジカル・スタンバイ・データベースを読取り専用アクセス用にオープンするかどうかを判断する場合は、次のことを考慮してください。

- 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おくと、管理リカバリ操作には使用できなくなります。アーカイブ REDO データはスタンバイ・データベースで受信されますが、REDO ログは適用されません。したがって、読取り専用アクセス用にオープンしたスタンバイ・データベースは、プライマリ・データベースとのトランザクションの同期が維持されません。スタンバイ・データベースで管理リカバリを再開し、アーカイブ REDO ログを適用して、スタンバイ・データベースをプライマリ・データベースと再同期化させることが必要となる場合があります。スタンバイ・データベースを読取り専用アクセス用にオープンしておくと、障害時リカバリでスタンバイ・データベースが必要なときにフェイルオーバーまたはスイッチオーバー操作に時間がかかる場合があります。
- 障害に対する保護およびレポート生成の目的でスタンバイ・データベースが必要な場合は、複数のスタンバイ・データベースの一部を読取り専用アクセス用にオープンし、残りのスタンバイ・データベースで管理リカバリ（スタンバイ・データベースにアーカイブ REDO ログを自動的に適用します）を実行できます。ただし、プライマリ・データベースの最新の変更をスタンバイ・データベースに適用するために、管理リカバリをすべてのスタンバイ・データベースで定期的に実行する必要があります。管理リカバリを実行するフィジカル・スタンバイ・データベースは、即時に障害に対する保護を提供します。

注意： ビジネスで、障害時リカバリ要件を満たすとともに、問合せとレポート生成の目的でスタンバイ・データベースを使用する必要がある場合は、ロジカル・スタンバイ・データベースの使用を検討してください。

読取り専用アクセス用にスタンバイ・データベースをオープン

スタンバイ・データベースは、次の手順に従って、読取り専用アクセス用オープンから管理リカバリ実行に（またはその逆に）変更できます。

スタンバイ・データベースが停止しているときに読取り専用アクセス用にオープンする方法

1. スタンバイ・データベースをマウントせずに、データベースで Oracle インスタンスを起動します。

```
SQL> STARTUP NOMOUNT;
```

2. 次のように入力してスタンバイ・データベースをマウントします。

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

3. 次のように入力してデータベースを読取り専用アクセス用にオープンします。

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

スタンバイ・データベースが管理リカバリを実行しているときに読取り専用アクセス用にオープンする方法

1. 次のように入力してログ適用サービスを取り消します。

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

2. 次のように入力してデータベースを読取り専用アクセス用にオープンします。

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

スタンバイ・データベースを読取り専用アクセス用オープンから管理リカバリ実行に変更する方法

1. スタンバイ・データベースのすべてのアクティブ・ユーザー・セッションを終了します。

2. 次のように入力してログ適用サービスを再開します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
 2> DISCONNECT FROM SESSION;
```

読取り専用アクセス用にオープンしたスタンバイ・データベースのソートに関する考慮事項

スタンバイ・データベースを読取り専用アクセス用にオープンする前に、ソート操作に関する次の項目を考慮してください。

- データベースが読取り専用アクセス用にオープンしているときのソート操作
- 一時表領域を使用しないソート操作

データベースが読取り専用アクセス用にオープンしているときのソート操作

読取り専用アクセス用にオープンしているスタンバイ・データベースで大量のデータをソートする問合せを実行するには、Oracle データベース・サーバーがディスク・ソート処理を実行できることが必要です。Oracle ソフトウェアによるデータ・ディクショナリへの書込みが発生するため、ソート処理のための領域を表領域に割り当てることはできません。

一時表領域を使用すると、問合せ作成の目的でデータベースが読取り専用アクセス用にオープンしているときに、ディクショナリ・ファイルまたは REDO 項目の生成に影響を与えずに `tempfile` 項目を追加できます。したがって、一時表領域を作成するための次の要件に従うかぎり、一時表領域を使用できます。

- 表領域は一時的なもので、ローカルで管理される必要があり、一時ファイルのみが格納されている必要があります。
- ユーザー・レベルの割当てと、ローカルで管理される一時表領域を使用する許可が、プライマリ・データベース上の適所に存在する必要があります。これらの設定は、スタンバイ・データベースでは変更できません。
- 一時表領域の一時ファイルをスタンバイ・データベースで作成して関連付ける必要があります。

読取り専用フィジカル・スタンバイ・データベースで使用するために一時表領域を作成する方法

フィジカル・スタンバイ・データベースを作成したときに一時表領域がプライマリ・データベースに存在しない場合は、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 SQL 文を入力します。

```
SQL> CREATE TEMPORARY TABLESPACE temp1
     TEMPFILE '/disk1/oracle/dbs/temp1.dbf'
     SIZE 20M REUSE
     EXTENT MANAGEMENT LOCAL UNIFORM SIZE 16M;
```

2. ログを切り替えて、REDO データをスタンバイ・データベースに送信します。

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

読取り専用フィジカル・スタンバイ・データベースで一時ファイルを作成して一時表領域に関連付ける方法

プライマリ・データベースで生成された REDO データは、アーカイブ REDO ログがフィジカル・スタンバイ・データベースに適用された後、スタンバイ制御ファイル内に一時表領域を自動的に作成します。ただし、フィジカル・スタンバイ・データベースを作成する前にプライマリ・データベースに一時表領域が存在している場合でも、実際にスタンバイ・データベースにディスク・ファイルを作成するには ADD TEMPFILE 句を使用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、次の手順を実行します。

1. 必要に応じて管理リカバリを起動し、次の SQL 文を入力してアーカイブ REDO ログを適用します。

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

2. 次の SQL 文を使用して、管理リカバリを取り消し、フィジカル・スタンバイ・データベースを読取り専用アクセス用にオープンします。

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   SQL> ALTER DATABASE OPEN READ ONLY;
   ```

   フィジカル・スタンバイ・データベースを読取り専用アクセス用にオープンすると、一時ファイルを追加できます。一時ファイルを追加しても REDO データは生成されないため、データベースを読取り専用アクセス用にオープンできます。

3. 一時表領域の一時ファイルを作成します。一時ファイルのサイズと名前は、プライマリ・データベースと異なっていても構いません。次に例を示します。

   ```
   SQL> ALTER TABLESPACE temp1
          ADD TEMPFILE '/disk1/oracle/dbs/s_temp1.dbf'
          SIZE 10M REUSE;
   ```

   関連項目： CREATE TEMPORARY TABLESPACE の構文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

一時表領域を使用しないソート操作

一時ファイルがスタンバイ・データベースに存在しない場合、またはスタンバイ・データベースがオープンしていない場合に大量のデータのソートを試行すると、次の例に示すように、エラーが戻されます。

```
SQL> SELECT * FROM V$PARAMETER;

select * from v$parameter

              *

ERROR at line 1:

ORA-01220: データベースのオープン前のファイル・ベースのソートは無効です。
```

ただし、サーバー・パラメータ・ファイルで SORT_AREA_SIZE パラメータを十分な値に設定した場合は、少量のデータをソートできます （SORT_AREA_SIZE パラメータは静的パラメータです)。

フィジカル・スタンバイ・データベースを使用したプライマリ・データベースのバックアップ・ファイルの作成

フィジカル・スタンバイ・データベースはプライマリ・データベースのコピーであるため、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・データベースからデータベース・バックアップ操作をオフロードできます。スタンバイ・サイトで Recovery Manager を使用すると、スタンバイ・データベースが管理リカバリを実行している間に、データ・ファイルとアーカイブ REDO ログをバックアップできます。これらのバックアップは、後から Recovery Manager を使用してプライマリ・データベースにリストアできます。

注意： 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 によるスタンバイ・データベースを使用したプライマリ・データベースのバックアップおよびリカバリの詳細は、『Oracle9*i* Recovery Manager ユーザーズ・ガイド』を参照してください。

スタンバイ・データベースに影響を与えるプライマリ・データベース・イベントの管理

問題が起きないようにするには、スタンバイ・データベースに影響するプライマリ・データベースのイベントを認識し、それらに応答する方法を見極める必要があります。この項では、それらのイベントについて説明し、イベントに対して推奨される応答についても説明します。

プライマリ・データベースで発生するイベントまたは変更の一部は、アーカイブ REDO ログを介してスタンバイ・データベースに自動的に伝播されるため、スタンバイ・データベースでそれ以上のアクションは不要です。それ以外の場合は、スタンバイ・データベースでメンテナンス・タスクを実行する必要があります。

プライマリ・データベースで行われた変更をスタンバイ・データベースへ伝播するためには、データベース管理者（DBA）の介入が必要かどうかを表 8-1 に示します。また、これらのイベントに応答する方法についても簡単に説明します。応答の詳細は、表に示されている参照先の項で説明します。

注意： ALTER DATABASE CLEAR UNARCHIVED LOGFILE 文を発行することによってプライマリ・データベースでログを消去したり、RESETLOGS オプションを使用してプライマリ・データベースをオープンすると、スタンバイ・データベースが無効になります。これらの操作はどちらもプライマリ・ログの順序番号を 1 にリセットするので、プライマリ・データベースによって生成されたアーカイブ REDO ログを適用できるようにするには、スタンバイ・データベースを再作成する必要があります。

次のイベントは、ログ転送サービスおよびログ適用サービスによって自動的に管理されるため、データベース管理者の介入は不要です。

- ENABLE THREAD または DISABLE THREAD 句を使用した SQL ALTER DATABASE 文の発行
- 表領域の状態の変更（読取り / 書込みまたは読取り専用に変更され、オンラインまたはオフラインにされる）
- STANDBY_FILE_MANAGEMENT 初期化パラメータが AUTO に設定されている場合のデータ・ファイルの追加または表領域の作成

表 8-1 プライマリ・データベースでの変更後にスタンバイ・データベースで必要なアクション

参照先	プライマリ・データベースで行われた変更	スタンバイ・データベースで必要なアクション
8-11 ページ	データ・ファイルの追加または表領域の作成	STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定していない場合は、新しいデータ・ファイルをスタンバイ・データベースにコピーする必要がある。
8-14 ページ	表領域またはデータ・ファイルの削除	アーカイブ REDO ログの適用後に、対応するデータ・ファイルを削除する。
8-15 ページ	データ・ファイルの改名	スタンバイ・データベースでデータ・ファイルを改名する。
8-16 ページ	オンライン REDO ログの追加または削除	変更をスタンバイ・データベースで同期化する。
8-17 ページ	プライマリ・データベース制御ファイルの変更（SQL ALTER DATABASE CREATE CONTROLFILE 文を使用）	変更に応じて、スタンバイ制御ファイルまたはスタンバイ・データベースを再作成する。
8-17 ページ	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・ファイルをスタンバイ・データベースに送信する。
第 11 章	初期化パラメータの変更	スタンバイ・パラメータを動的に変更するか、またはスタンバイ・データベースを停止し、初期化パラメータ・ファイルを更新する。

データ・ファイルの追加または表領域の作成

STANDBY_FILE_MANAGEMENT 初期化パラメータを使用して、次のように、プライマリ・データベースへのデータ・ファイルの追加がスタンバイ・データベースに自動的に伝播されるかどうかを制御できます。

- スタンバイ・データベースのサーバー・パラメータ・ファイルで STANDBY_FILE_MANAGEMENT 初期化パラメータを AUTO に設定した場合は、プライマリ・データベースで作成された新しいデータ・ファイルがスタンバイ・データベースでも自動的に作成されます。
- STANDBY_FILE_MANAGEMENT 初期化パラメータを指定していない場合、またはこのパラメータを MANUAL に設定した場合は、新しいデータ・ファイルをプライマリ・データベースに追加するときに、そのデータ・ファイルをスタンバイ・データベースに手動でコピーする必要があります。

既存のデータ・ファイルを別のデータベースからプライマリ・データベースへコピーする場合は、STANDBY_FILE_MANAGEMENT 初期化パラメータの設定に関係なく、新しいデータ・ファイルをスタンバイ・データベースにコピーし、スタンバイ制御ファイルを再作成する必要があります。

次の各項では、STANDBY_FILE_MANAGEMENT 初期化パラメータが AUTO または MANUAL に設定されている場合に、データ・ファイルをプライマリおよびスタンバイ・データベースに追加する例を示します。

STANDBY_FILE_MANAGEMENT が AUTO に設定されている場合の表領域およびデータ・ファイルの追加

次の例は、STANDBY_FILE_MANAGEMENT 初期化パラメータが AUTO に設定されている場合に、新しいデータ・ファイルをプライマリおよびスタンバイ・データベースに追加する手順を示しています。

1. プライマリ・データベースに新しい表領域を追加します。

```
SQL> CREATE TABLESPACE new_ts DATAFILE 't_db2.dbf'
  2> SIZE 1m AUTOEXTEND ON MAXSIZE UNLIMITED;
```

2. スタンバイ・データベースにコピーされるようにカレント REDO ログをアーカイブします。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

3. 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たかどうかを確認します。

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
--------------------------------------------------------------------------
/disk1/oracle/dbs/t_db1.dbf
/disk1/oracle/dbs/t_db2.dbf
```

4. 新しいデータ・ファイルがスタンバイ・データベースに追加されたかどうかを確認します。

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME
--------------------------------------------------------------------------
/disk1/oracle/dbs/s2t_db1.dbf
/disk1/oracle/dbs/s2t_db2.dbf
```

STANDBY_FILE_MANAGEMENT が MANUAL に設定されている場合の表領域およびデータ・ファイルの追加

次の例は、STANDBY_FILE_MANAGEMENT 初期化パラメータが MANUAL に設定されている場合に、新しいデータ・ファイルをプライマリおよびスタンバイ・データベースに追加する手順を示しています。スタンバイ・データ・ファイルが RAW デバイスに存在する場合は、STANDBY_FILE_MANAGEMENT 初期化パラメータを MANUAL に設定する必要があります。

1. プライマリ・データベースに新しい表領域を追加します。

```
SQL> CREATE TABLESPACE new_ts DATAFILE 't_db2.dbf'
  2> SIZE 1m AUTOEXTEND ON MAXSIZE UNLIMITED;
```

2. 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たかどうかを確認します。

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME

------------------------------------------------------------------------
/disk1/oracle/dbs/t_db1.dbf
/disk1/oracle/dbs/t_db2.dbf
```

3. 次の手順を実行して、表領域をリモート・スタンバイ・ロケーションにコピーします。

   a. 新しい表領域をオフラインにします。

   ```
   SQL> ALTER TABLESPACE new_ts OFFLINE;
   ```

   b. オペレーティング・システム・ユーティリティの copy コマンドを使用して、新しい表領域をローカルの一時的な位置にコピーします。ファイルを一時的な位置にコピーすると、表領域をオフラインにする時間が短縮されます。次の例では、UNIX の cp コマンドを使用して表領域をコピーします。

   ```
   % cp t_db2.dbf s2t_db2.dbf
   ```

   c. 新しい表領域をオンラインに戻します。

   ```
   SQL> ALTER TABLESPACE new_ts ONLINE;
   ```

   d. オペレーティング・システム・ユーティリティのコマンドを使用して、表領域のローカル・コピーをリモート・スタンバイ・ロケーションにコピーします。次の例では、UNIX の rcp コマンドを使用しています。

   ```
   %rcp s2t_db2.dbf standby_location
   ```

4. スタンバイ・データベースにコピーされるように、プライマリ・データベースでカレント REDO ログをアーカイブします。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

5. 次の問合せを使用して、管理リカバリが実行中であることを確認します。MRP または MRP0 プロセスが戻された場合、管理リカバリは実行されています。

```
SQL> SELECT PROCESS, STATUS FROM V$MANAGED_STANDBY;
```

6. REDO ログがスタンバイ・データベースに適用された後、データ・ファイルがスタンバイ・データベースに追加されたかどうかを確認します。

```
SQL> SELECT NAME FROM V$DATAFILE;
NAME

----------------------------------------------------------------------
/disk1/oracle/dbs/s2t_db1.dbf
/disk1/oracle/dbs/s2t_db2.dbf
```

プライマリ・データベースの表領域の削除

プライマリ・データベースで 1 つ以上のデータ・ファイルまたは表領域を削除した場合は、次の手順に従って、スタンバイ・データベースでも対応するデータ・ファイルを削除する必要があります。

1. プライマリ・サイトで表領域を削除します。

```
SQL> DROP TABLESPACE tbs_4;
SQL> ALTER SYSTEM SWITCH LOGFILE;
% rm tbs_4.dbf
```

2. 管理リカバリがオンになっていることを確認します（オンの場合は変更がスタンバイ・データベースに適用されます）。次の問合せが MRP または MRP0 プロセスを戻した場合、管理リカバリはオンになっています。

```
SQL> SELECT PROCESS, STATUS FROM V$MANAGED_STANDBY;
```

3. アーカイブ REDO ログがスタンバイ・データベースに適用された後、スタンバイ・サイトで対応するデータ・ファイルを削除します。次に例を示します。

```
% rm tbs_4.dbf
```

4. 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 tbs_4.dbf
```

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。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物理構造にしない場合は、次の手順を実行する必要はありません。

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 tbs_4.dbf 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 'tbs_4.dbf'
  2>                                     TO '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てすべてのログが適用されたことを確認した後、管理リカバリ操作を停止します。

```
SQL> SELECT NAME, SEQUENCE#, ARCHIVED, APPLIED
  2> FROM V$ARCHIVED_LOG;
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 tbs_4.dbf tbs_x.dbf
```

7. 新しい制御ファイルを使用してスタンバイ・データベースを起動してマウントします。

```
SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

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 'tbs_4.dbf'
  2>                         TO 'tbs_x.dbf';
```

9. スタンバイ・データベースで、管理リカバリ操作を再開します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
  2> DISCONNECT FROM SESSION;
```

スタンバイ・サイトで対応するデータ・ファイルを改名せず、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: エラーによってリカバリ・セッションは取り消されました。
ORA-01157: データ・ファイル 4 を識別 / ロックできません－ DBWR トレース・ファイルを参照してください
ORA-01110: データ・ファイル 4: '/disk1/oracle/dbs/tbs_x.dbf'
```

オンライン REDO ログの追加または削除

データベースをチューニングするために、オンライン REDO ログのサイズと数を変更する場合があります。スタンバイ・データベースへの影響なしに、プライマリ・データベースへ REDO ログ・ファイル・グループまたはメンバーを追加することができます。同様に、スタンバイ・データベースへの影響なしに、プライマリ・データベースからログ・ファイル・グループまたはメンバーを削除できます。 ただし、これらの変更は、スイッチオーバー後にスタンバイ・データベースのパフォーマンスに影響します。

たとえば、REDO ログがプライマリ・データベースに 10 個、スタンバイ・データベースに 2 個ある場合、新しいプライマリ・データベースとして機能するようにスタンバイ・データベースにスイッチオーバーすると、新しいプライマリ・データベースは元のプライマリ・データベースより頻繁にアーカイブするように強制されます。

したがって、プライマリ・サイトでオンライン REDO ログを追加または削除した場合は、次の手順に従って、スタンバイ・データベースで変更を同期化することが重要です。

1. 管理リカバリがオンの場合は、ログを変更する前に管理リカバリを取り消す必要があります。
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. オンライン REDO ログを追加または削除します。
   - オンライン REDO ログを追加するには、次のような SQL 文を使用します。

     ```
     SQL> ALTER DATABASE ADD STANDBY LOGFILE 'prmy3.log' SIZE 100K;
     ```

   - オンライン REDO ログを削除するには、次のような SQL 文を使用します。

     ```
     SQL> ALTER DATABASE DROP STANDBY LOGFILE 'prmy3.log';
     ```

4. 手順 3 で使用した文を各スタンバイ・データベースで繰り返します。
5. STANDBY_FILE_MANAGEMENT 初期化パラメータおよび管理リカバリ・オプションを元の状態にリストアします。

プライマリ・データベースの制御ファイルの変更

プライマリ・データベースで SQL CREATE CONTROLFILE 文を使用して RESETLOGS オプションを指定すると、プライマリ・データベースを次にオープンしたときにオンライン・ログが強制的にリセットされるので、スタンバイ・データベースが無効になります。

スタンバイ・データベースの制御ファイルを無効にした場合は、3-5 ページの「スタンバイ・データベース用の制御ファイルの作成」の手順に従ってファイルを再作成してください。

スタンバイ・データベースを無効にした場合は、第 3 章の手順に従ってスタンバイ・データベースを再作成する必要があります。

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、この文は無効になったデータベースを修正しません。

リカバリ不能処理（ダイレクト・パス・ロードなど）を実行するとプライマリ・データベースのパフォーマンスは改善されますが、スタンバイ・データベースでは対応するリカバリ処理のパフォーマンスの改善はなく、データを手動でスタンバイ・データベースに移動する必要があります。

> **関連項目：** NOLOGGING 句を使用した後のリカバリについては、10-22 ページの「NOLOGGING 句を指定した後のリカバリ」を参照してください。

プライマリおよびスタンバイ・データベースの監視

この項では、Data Guard 環境のプライマリおよびスタンバイ・データベースを監視するための情報の検索方法について概要を説明します。

この項は、次の項目で構成されています。

- アラート・ログ
- 動的パフォーマンス・ビュー（固定ビュー）
- リカバリの進捗の監視

表 8-2 に、プライマリ・データベースで発生する共通イベント、およびプライマリ・サイトとスタンバイ・サイトでこれらのイベントを監視できるファイルとビューに関する情報をまとめます。

表 8-2 プライマリ・データベースの共通アクションを監視できる場所

プライマリ・データベース・イベント	プライマリ・サイト情報	スタンバイ・サイト情報
ENABLE THREAD または DISABLE THREAD 句を指定した SQL ALTER DATABASE 文の発行	■ アラート・ログ ■ V$THREAD ビュー	アラート・ログ
REDO ログの変更	■ アラート・ログ ■ V$LOG ビュー ■ V$LOGFILE ビューの STATUS 列	アラート・ログ
CREATE CONTROLFILE 文の発行	アラート・ログ	アラート・ログ[1]
管理リカバリの実行	アラート・ログ	アラート・ログ
表領域の状態の変更（読取り / 書込みまたは読取り専用にされ、オンラインまたはオフラインにされる）	■ DBA_TABLESPACES ビュー ■ アラート・ログ	V$RECOVER_FILE ビュー
データ・ファイルの追加または表領域の作成	■ DBA_DATA_FILES ビュー ■ アラート・ログ	V$DATAFILE ビュー アラート・ログ
表領域の削除	■ DBA_DATA_FILES ビュー ■ アラート・ログ	V$DATAFILE ビュー アラート・ログ
表領域またはデータ・ファイルをオフラインにする、またはデータ・ファイルをオフラインで削除する	■ V$RECOVER_FILE ビュー ■ アラート・ログ	V$RECOVER_FILE ビュー
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

表 8-2 プライマリ・データベースの共通アクションを監視できる場所（続き）

プライマリ・データベース・イベント	プライマリ・サイト情報	スタンバイ・サイト情報
ログに記録されていないまたはリカバリ不能な操作	■ V$DATAFILE ビュー ■ V$DATABASE ビュー	アラート・ログ
リカバリの進捗	■ V$ARCHIVE_DEST_STATUS ビュー ■ アラート・ログ	V$ARCHIVED_LOG ビュー V$LOG_HISTORY ビュー V$MANAGED_STANDBY ビュー アラート・ログ
データ・ファイルの自動拡張	アラート・ログ	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

[1] CREATE CONTROLFILE 文をプライマリ・データベースで発行した場合、スタンバイ・データベースは初期化パラメータに応じて、REDO データを検出しないかぎり正常に機能します。

アラート・ログ

データベースのアラート・ログは、メッセージとエラーに関する時系列のレコードです。アラート・ログには、Oracle データベースに関する情報以外に、次のような Data Guard 固有の操作に関する情報も含まれています。

- SQL 文の ALTER DATABASE RECOVER MANAGED STANDBY、STARTUP、SHUTDOWN、ARCHIVE LOG、RECOVER などの管理操作に関連するメッセージ
- ARC0、MRP0、RFS、LGWR などのバックグラウンド・プロセスでレポートされる管理操作に関連するエラー
- 管理操作の完了タイムスタンプ

アラート・ログは、特定のプロセスで生成されたトレース・ファイルまたはダンプ・ファイルに関する情報も提供します。

動的パフォーマンス・ビュー（固定ビュー）

Oracle データベース・サーバーには、サーバーが維持するビューのセットがあります。これらのビューは、データベースがオープン状態および使用中の場合に連続的に更新されるため、**動的パフォーマンス・ビュー**と呼ばれることがあります。その内容は主にパフォーマンスに関連しています。また、これらのビューは、データベース管理者が変更または削除できないので、**固定ビュー**と呼ばれることもあります。

これらのビューの名前には、V$ または GV$ という接頭辞が付き、たとえば `V$ARCHIVE_DEST` や `GV$ARCHIVE_DEST` のようになります。

標準の動的パフォーマンス・ビュー（V$ 固定ビュー）には、ローカル・インスタンスの情報が格納されます。それに対して、グローバル動的パフォーマンス・ビュー（GV$ 固定ビュー）には、すべてのオープン・インスタンスの情報が格納されます。各 V$ 固定ビューには対応する GV$ 固定ビューがあります。

> **関連項目：** ビューの列の詳細は、第 14 章「ビュー」および『Oracle9*i* データベース・リファレンス』を参照してください。

リカバリの進捗の監視

この項では、8-20 ページの「動的パフォーマンス・ビュー（固定ビュー）」で説明した、Data Guard 環境でリカバリの進捗を監視するビューの例をいくつか示します。この項は、次の例で構成されています。

- プロセス・アクティビティの監視
- 管理リカバリ操作の進捗の確認
- アーカイブ REDO ログの位置および作成者の確認
- アーカイブ・ログ履歴の表示
- スタンバイ・データベースに適用されたログの確認
- スタンバイ・サイトが受信しなかったログの確認

プロセス・アクティビティの監視

次のプロセスを実行してアクティビティを監視することにより、スタンバイ・データベースでの管理リカバリ操作に関する情報を取得できます。

- ARC0
- MRP/MRP0
- RFS

スタンバイ・データベース・サイトの V$MANAGED_STANDBY ビューには、Data Guard 環境内でログ転送プロセスおよびログ適用プロセスの両方で実行されたアクティビティが表示されます。次の問合せ出力の CLIENT_P 列で、対応するプライマリ・データベース・プロセスを識別します。

```
SQL> SELECT PROCESS, CLIENT_PROCESS, SEQUENCE#, STATUS FROM V$MANAGED_STANDBY;

PROCESS CLIENT_P  SEQUENCE# STATUS
------- -------- ---------- ------------
ARCH    ARCH              0 CONNECTED
ARCH    ARCH              0 CONNECTED
MRP0    N/A             204 WAIT_FOR_LOG
RFS     LGWR            204 WRITING
RFS     N/A               0 RECEIVING
```

管理リカバリ操作の進捗の確認

プライマリまたはスタンバイ・データベース・サイトのいずれかの V$ARCHIVE_DEST_STATUS ビューには、アーカイブされた REDO ログ、適用されたアーカイブ REDO ログ、それぞれのログ順序番号などの情報が表示されます。次の問合せ出力は、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 ログの適用がアーカイブ・ログ 2 つ分プライマリ・データベースから遅れていることを示しています。

```
SQL> SELECT ARCHIVED_THREAD#, ARCHIVED_SEQ#, APPLIED_THREAD#, APPLIED_SEQ#
  2> FROM V$ARCHIVE_DEST_STATUS;

ARCHIVED_THREAD# ARCHIVED_SEQ# APPLIED_THREAD# APPLIED_SEQ#
---------------- ------------- --------------- ------------
1                947           1               945
```

アーカイブ REDO ログの位置および作成者の確認

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 ログに関する追加情報を検索できます。このビューから、アーカイブ REDO ログの位置、アーカイブ REDO ログを作成したプロセス、各アーカイブ REDO ログの REDO ログ順序番号、ログがアーカイブされた時期、アーカイブ REDO ログが適用されたかどうかなどの情報を取得できます。次に例を示します。

```
SQL> SELECT NAME, CREATOR, SEQUENCE#, APPLIED, COMPLETION_TIME
  2> FROM V$ARCHIVED_LOG;

NAME                                            CREATOR SEQUENCE# APP COMPLETIO
----------------------------------------------  ------- --------- --- ---------
H:\ORACLE\ORADATA\PAYROLL\STANDBY\ARC00198.001  FGRD          198 YES 30-MAY-02
H:\ORACLE\ORADATA\PAYROLL\STANDBY\ARC00199.001  FGRD          199 YES 30-MAY-02
H:\ORACLE\ORADATA\PAYROLL\STANDBY\ARC00200.001  FGRD          200 YES 30-MAY-02
H:\ORACLE\ORADATA\PAYROLL\STANDBY\ARC00201.001  LGWR          201 YES 30-MAY-02
H:\ORACLE\ORADATA\PAYROLL\STANDBY\ARC00202.001  FGRD          202 YES 30-MAY-02
H:\ORACLE\ORADATA\PAYROLL\STANDBY\ARC00203.001  LGWR          203 YES 30-MAY-02

6 rows selected.
```

アーカイブ・ログ履歴の表示

フィジカル・スタンバイ・サイトの V$LOG_HISTORY ビューには、最初のエントリの時間、ログ内の最小の SCN、ログ内の最大の SCN、アーカイブ・ログの順序番号など、アーカイブ・ログの完全な履歴が表示されます。

```
SQL> SELECT FIRST_TIME, FIRST_CHANGE#, NEXT_CHANGE#, SEQUENCE# FROM V$LOG_HISTORY;

FIRST_TIM FIRST_CHANGE# NEXT_CHANGE#  SEQUENCE#
--------- ------------- ------------ ----------
13-MAY-02        190578       214480          1
13-MAY-02        214480       234595          2
13-MAY-02        234595       254713          3
.
.
.
30-MAY-02       3418615      3418874        201
30-MAY-02       3418874      3419280        202
30-MAY-02       3419280      3421165        203

203 rows selected.
```

スタンバイ・データベースに適用されたログの確認

スタンバイ・データベースで V$LOG_HISTORY ビューを問い合せてください。このビューには、適用された最新のログ順序番号が記録されています。たとえば、次のような問合せを発行します。

```
SQL> SELECT THREAD#, MAX(SEQUENCE#) AS "LAST_APPLIED_LOG"
  2> FROM V$LOG_HISTORY
  3> GROUP BY THREAD#;

THREAD# LAST_APPLIED_LOG
------- ----------------
      1              967
```

この例では、ログ順序番号 967 のアーカイブ REDO ログが最新の適用済みログです。

また、スタンバイ・データベースで V$ARCHIVED_LOG 固定ビューの APPLIED 列を使用すると、スタンバイ・データベースに適用されるログを識別できます。適用されたログの場合は列に YES と表示されます。次に例を示します。

```
SQL> SELECT THREAD#, SEQUENCE#, APPLIED FROM V$ARCHIVED_LOG;

   THREAD#  SEQUENCE# APP
---------- ---------- ---
         1          2 YES
         1          3 YES
         1          4 YES
         1          5 YES
         1          6 YES
         1          7 YES
         1          8 YES
         1          9 YES
         1         10 YES
         1         11 NO

10 rows selected.
```

スタンバイ・サイトが受信しなかったログの確認

各アーカイブ先には、割り当てられた宛先 ID があります。V$ARCHIVE_DEST 固定ビューの DEST_ID 列を問い合せると、宛先 ID を識別できます。次に、プライマリ・データベースでの問合せにこの宛先 ID を使用すると、特定のスタンバイ・サイトに送信されなかったログを識別できます。

たとえば、プライマリ・データベースのカレント・ローカル・アーカイブ先 ID が 1 で、リモート・スタンバイ・データベースの 1 つの宛先 ID が 2 であるとします。このスタンバイ宛先が受信しなかったログを識別するには、プライマリ・データベースで次の問合せを発行します。

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

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

この例には、スタンバイ宛先 2 が受信しなかったログが示されています。

9

ロジカル・スタンバイ・データベースの管理

この章では、ロジカル・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。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・データベースの構成と管理
- ロジカル・スタンバイ・データベースのチューニング

この章の各項では、ロジカル・スタンバイ・データベースを管理するための SQL 文、初期化パラメータ、ビューおよび PL/SQL パッケージ DBMS_LOGSTDBY の使用方法について、それぞれ説明します。

> **関連項目：** この章で説明する管理タスクを自動化するために Data Guard Broker を使用する場合は、『Oracle9*i* Data Guard Broker』を参照してください。

ロジカル・スタンバイ・データベースの構成と管理

この PL/SQL パッケージ DBMS_LOGSTDBY には、ロジカル・スタンバイ・データベースの構成と管理に役立つプロシージャが用意されています。PL/SQL パッケージ DBMS_LOGSTDBY を使用すると、ロジカル・スタンバイ・データベースに関する次のような管理タスクを実行できます。

- SQL 適用操作の管理
- 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
- 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・データベースでのトリガーと制約の処理
- ロジカル・スタンバイ・データベースでの SQL 適用操作のスキップ
- 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
- ロジカル・スタンバイ・イベントの表示と制御
- SQL 適用操作アクティビティの表示
- アーカイブ REDO ログの適用の遅延
- 適用された REDO ログ・データ量の確認
- エラーのリカバリ
- マテリアライズド・ビューのリフレッシュ

注意： DBMS_LOGSTDBY パッケージへのアクセスが必要なユーザーには、LOGSTDBY_ADMINISTRATOR ロールを付与する必要があります。

SQL 適用操作の管理

DBMS_LOGSTDBY PL/SQL パッケージには、ロジカル・スタンバイ・データベースでの SQL 適用操作の管理に役立つプロシージャが含まれています。このパッケージを使用すると、次のことができます。

- スタンバイ・データベース内の選択した表またはスキーマ全体へのアーカイブ REDO ログの適用をスキップする方法を提供する
- ログ適用サービスで使用する初期化パラメータを管理する
- サプリメンタル・ロギングが正しく使用できることを保証する
- ロジカル・スタンバイ・データベースには適用の必要がない一連の操作を説明する
- DML または DDL の変更を一時表に適用しない
- CREATE、ALTER または DROP INDEX の各操作を適用しない

- DDL 文の適用時にエラーが発生したときは、そのエラーを記録し、ロジカル・スタンバイ・データベースへのアーカイブ REDO ログの適用を継続する
- エラーが DDL 文で発生したときは、ログ適用サービスを停止し、DBA による処置の指定を待機する

> **関連項目：** DBMS_LOGSTDBY パッケージの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を、DBMS_LOGSTDBY PL/SQL パッケージによって提供されるプロシージャの要約は、第 9 章を参照してください。

表 9-1 に、PL/SQL パッケージ DBMS_LOGSTDBY のプロシージャをまとめます。

表 9-1 PL/SQL パッケージ DBMS_LOGSTDBY のプロシージャ

サブプログラム	説明
APPLY_SET	特定の初期化パラメータの値を設定して、SQL 適用操作の構成およびメンテナンスを可能にする。
APPLY_UNSET	特定の初期化パラメータの値をシステムのデフォルト値にリセットする。
BUILD	サプリメンタル・ロギングが正しく使用できることを保証し、LogMiner ディクショナリを作成する。
GUARD_BYPASS_OFF	以前に GUARD_BYPASS_ON プロシージャを使用して無効にしたデータベース・ガードを再び使用可能にする。
GUARD_BYPASS_ON	現在のセッションでデータベース・ガードを無効にする。これによって、ロジカル・スタンバイ・データベース内の表が変更可能になる。
INSTANTIATE_TABLE	プライマリ・データベース内の対応する表から、スタンバイ・データベースに表を作成して移入する。
SKIP	プライマリ・データベースで完了したデータベース操作の中で、ロジカル・スタンバイ・データベースに適用しない操作を指定できる。
SKIP_ERROR	エラーが発生した場合に従う条件を指定する。SQL 適用操作を停止するか、またはエラーを無視できる。
SKIP_TRANSACTION	特定の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ときにスキップ（無視）するトランザクション識別情報を指定する。このサブプログラムでは、代替文も実行できる。
UNSKIP	SKIP プロシージャで設定したオプションを変更する。

表 9-1 PL/SQL パッケージ DBMS_LOGSTDBY のプロシージャ（続き）

サブプログラム	説明
UNSKIP_ERROR	SKIP_ERROR プロシージャで設定したオプションを変更する。
UNSKIP_TRANSACTION	SKIP_TRANSACTION プロシージャで設定したオプションを変更する。

関連項目： DBMS_LOGSTDBY パッケージの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

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

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・アクセスは、SQL 文 ALTER DATABASE GUARD によって制御されます。ロジカル・スタンバイ・データベースでログ適用サービスを初めて開始するまで、ユーザーはロジカル・スタンバイ・データベースを変更できます。ただし、ログ適用サービスを開始すると、データベース・ガードがデフォルトで ALL に設定されます。

ALTER DATABASE GUARD 文で使用できるキーワードは、次のとおりです。

- 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いように保護されます。

- STANDBY

  STANDBY を指定すると、SYS 以外のすべてのユーザーが、SQL 適用操作を介してメンテナンスされている表または順序を DML および DDL によって変更できないように保護されます。

- NONE

  データベース内の全データに対して通常のセキュリティが必要な場合は、NONE を指定します。

たとえば、データベース・ガードを使用可能にしてロジカル・スタンバイ・データベース内の表へのユーザー・アクセスを保護するには、次の文を使用します。

```
SQL> ALTER DATABASE GUARD ALL;
```

ロジカル・スタンバイ・データベースの変更

データベース・ガードを一時的に無視して、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、DBMS_LOGSTDBY.GUARD_BYPASS_ON プロシージャを実行します。次の各項では、ロジカル・スタンバイ・データベースを変更するためにデータベース・ガードを一時的にバイパスすることが有用と考えられる 2 つの例を示します。

- ロジカル・スタンバイ・データベースでの DDL の実行
- SQL 適用でメンテナンスされていない表の変更

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。

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 適用操作を介してメンテナンスされている表に索引を追加する方法について説明します。

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、ログ適用サービスを停止し、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。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered.

SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
PL/SQL procedure successfully completed.

SQL> ALTER TABLE SCOTT.EMP ADD CONSTRAINT EMPID UNIQUE (EMPNO);
Table altered.

SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
PL/SQL procedure successfully completed.

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
Database altered.
```

このサンプル・プロシージャは、その他の DDL 文の実行にも使用できます。オラクル社では、データベース・ガードのバイパスが有効になっている間は 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。多くの場合、ロジカル・スタンバイ・データベースで行が段階的にメンテナンスされるのと同じ方法で、表の行を変更することはできません。

SQL 適用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 適用操作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 適用操作を介して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。この設定によって、ロジカル・スタンバイ・データベースがメンテナンスしている表のリストが記述されます。このリストには、アプリケーションによって書込みが行われる表を含めることはできません。

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、変更をロジカル・スタンバイ・データベースに適用できるように、SQL 適用操作を停止し、表をスキップした後、SQL 適用操作を再開しています。この操作によって、レポート生成アプリケーションは、MYSCHEMA の MYTABLES% 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 適用操作を介して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('SCHEMA_DDL','MYSCHEMA','MYTABLES%');
PL/SQL procedure successfully completed.

SQL> EXECUTE DBMS_LOGSTDBY.SKIP('DML','MYSCHEMA','MYTABLES%');
PL/SQL procedure successfully completed.

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
Database altered.
```

次に、DBA_LOGSTDBY_PARAMETERS ビューを問い合せ、ロジカル・スタンバイ・データベースが更新されることを確認します。次の例のように、列が戻らなくなるまで問合せを繰り返す必要があるため、確認に時間がかかることがあります。

```
SQL> SELECT NAME FROM DBA_LOGSTDBY_PARAMETERS WHERE NAME = 'EVALUATE_SKIP';
no rows selected
```

最後に、表を更新できるようにデータベース・ガードを設定します。

```
SQL> ALTER DATABASE GUARD STANDBY;
Database altered.
```

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

スタンバイ・データベースの場合、トリガーと制約は使用可能ですが実行されません。SQL 適用操作を介してメンテナンスされている表に対するトリガーと制約の場合、制約はプライマリ・データベースで評価されるので、ロジカル・スタンバイ・データベースで再評価する必要はありません。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。SQL 適用操作を介してメンテナンスされていない表の場合は、トリガーが発行され、制約が評価されます。

ロジカル・スタンバイ・データベースでの SQL 適用操作のスキップ

スタンバイ・データベース上で関連しているものが、プライマリ・データベースのアクティビティのサブセットのみの場合は、DBMS_LOGSTDBY.SKIP プロシージャを使用して、ログ適用サービスがロジカル・スタンバイ・データベースで SQL 文を発行しないように保護するフィルタを定義します（自動的にスキップされる SQL 文の詳細は、4-4 ページの「データ型または表のサポートの判別」を参照してください）。

サポートされないデータ型または文がフィルタ処理で自動的に除外された後、表での SQL 文の適用が続行されます。ただし、ロジカル・スタンバイ・データベースに適用しない表をスキップするには、DBMS_LOGSTDBY.SKIP プロシージャを使用する必要があります。次のリストは、ロジカル・スタンバイ・データベースに適用されないようにフィルタ処理またはスキップできる一般的な SQL 文の例を示しています。

- 表に対する DML または DDL 変更
- CREATE、ALTER または DROP INDEX DDL 文
- CREATE、ALTER、DROP または TRUNCATE TABLE 文
- CREATE、ALTER または DROP TABLESPACE 文
- CREATE または DROP VIEW 文

例 9-1 は、ロジカル・スタンバイ・データベース内の EMP 表を参照する SQL 適用操作をすべてスキップする方法を示しています。

例 9-1 ロジカル・スタンバイ・データベース内の表のスキップ

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> EXECUTE DBMS_LOGSTDBY.SKIP('SCHEMA_DDL', 'SCOTT', 'EMP', NULL);
SQL> EXECUTE DBMS_LOGSTDBY.SKIP('DML', 'SCOTT', 'EMP', NULL);
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

スキーマおよび非スキーマ操作のための DML 文と DDL 文のスキップに加え、特定の DML 操作と DDL 操作もスキップできます。例 9-2 は、非スキーマの DDL 操作を行う ALTER TABLESPACE および CREATE TABLESPACE をスキップする方法を示しています。

例 9-2 ALTER または CREATE TABLESPACE 文のスキップ

```
SQL> EXEC DBMS_LOGSTDBY.SKIP('CREATE TABLESPACE', NULL, NULL, NULL);
SQL> EXEC DBMS_LOGSTDBY.SKIP('ALTER TABLESPACE', NULL, NULL, NULL);

SQL> COLUMN ERROR FORMAT a5;
SQL> COLUMN STATEMENT_OPT FORMAT a20;
SQL> COLUMN OWNER FORMAT a10
SQL> COLUMN NAME FORMAT a15;
SQL> COLUMN PROC FORMAT a20;
SQL> SELECT * FROM DA_LOGSTDBY_SKIP;

ERROR STATEMENT_OPT     OWNER      NAME            PROC
----- ----------------- ---------- --------------- --------------------
N     CREATE TABLESPACE
N     ALTER TABLESPACE
```

特に DDL 文をスキップする場合は、SKIP プロシージャを慎重に使用してください。たとえば、CREATE TABLE 文がスキップされる場合は、その表を参照する他の DDL 文もスキップする必要があります。スキップしない場合、これらの文は失敗となり、例外が発生します。例外が発生した場合は、SQL 適用サービスの実行を停止して、手動で再開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、表のインスタンス化を使用します。また、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する SQL 適用操作を使用可能にすることもできます。

表を作成する前に、4-4 ページの「データ型または表のサポートの判別」および 4-7 ページの「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。これらの項では、次の方法が説明されています。

- 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や表がプライマリ・データベースに含まれているかどうかを判断する方法
- 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を確認する方法

注意： ロジカル・スタンバイ・データベースのオブジェクトで BLOB データ型がサポートされている場合でも、DBMS_LOGSTDBY PL/SQL パッケージでは、BLOB データ型をサポートしません。

次のリストおよび例 9-3 は、表を再作成し、その表に対して SQL 適用操作を再開する方法を示しています。

1. ログ適用サービスを停止して、SQL 文のデータベースへの適用を停止します。
2. DBA_LOGSTDBY_SKIP ビューを問い合せて、操作がスキップされていないことを確認します。

   操作がスキップされている場合は、DBMS_LOGSTDBY.UNSKIP プロシージャを使用して、現在スキップされている各操作の適用を再開します。表に対して複数のフィルタが作成されている場合は、プロシージャを複数回実行する必要があります。
3. DBMS_LOGSTDBY.INSTANTIATE_TABLE プロシージャを使用して、ロジカル・スタンバイ・データベースに表を再作成します。表の作成の他に、データベース・リンクを使用してプライマリ表からのデータのインポートも実行されます。このプロシージャに提供されるリンクには、プライマリ・データベースに対する LOGSTDBY_ADMINISTRATOR ロールが付与されている必要があります。
4. ログ適用サービスを再開します。

新しく追加した表のデータにアクセスする前に、プライマリ・データベースのカレント REDO ログをアーカイブし、その内容がロジカル・スタンバイ・データベースに適用されていることを確認する必要があります。

例 9-3 は、EMP 表をロジカル・スタンバイ・データベースに追加する方法を示しています。

例 9-3 ロジカル・スタンバイ・データベースへの表の追加

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> SELECT * FROM DBA_LOGSTDBY_SKIP;

ERROR  STATEMENT_OPT         OWNER          NAME            PROC
-------------------------------------------------------------------
N      SCHEMA_DDL            SCOTT          EMP
N      DML                   SCOTT          EMP

SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP('SCHEMA_DDL','SCOTT','EMP');
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE('SCOTT','EMP','DBLINK');
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP('DML','SCOTT','EMP');
```

プライマリ・データベースにログオンし、次の文を発行します。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
SQL> SELECT FIRST_CHANGE# FROM V$LOG WHERE STATUS = 'CURRENT';
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

DBA_LOGSTDBY_PROGRESS.APPLIED_SCN プロシージャで戻された値が、V$LOG ビューの問合せで選択された値を超えている場合、そのデータベースには一貫性があるため、レポートは安全に再実行できます。

ロジカル・スタンバイ・イベントの表示と制御

DBA_LOGSTDBY_EVENTS ビューを問い合せると、SQL 適用操作のアクティビティが含まれたイベント表が表示されます。このイベント表には、特に、DDL の実行やエラーの原因となった実行が記録されています。イベント表に記録するアクティビティの内容と量は、制御できます。デフォルトでは、この表に格納できるレコード数は 100 件ですが、レコード数は増加できます。次に例を示します。

```
SQL> DBMS_LOGSTDBY.APPLY_SET('MAX_EVENTS_RECORDED', '200');
```

さらに、記録するイベントのタイプを設定できます。デフォルトでは、すべてのタイプが表に記録されます。ただし、RECORD_SKIP_DDL、RECORD_SKIP_ERRORS および RECORD_APPLIED_DDL の各パラメータを FALSE に設定して、これらのイベントの記録を回避できます。

SQL 適用操作の停止原因となったエラーは、システム表領域に十分な領域があるかぎり、必ずイベント表に記録されます。これらのイベントは、常に ALERT.LOG ファイルにも格納され、テキストには、'LOGSTDBY event' という句が含まれます。このビューを問い合せたときは、EVENT_TIME、COMMIT_SCN、CURRENT_SCN の順で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。

SQL 適用操作アクティビティの表示

ロジカル・スタンバイ・データベースに対する SQL 適用操作では、パラレル実行サーバーと多様なタスクを実行するバックグラウンド・プロセスの集合が使用されます。V$LOGSTDBY ビューには、各プロセスで現在実行されている内容が表示され、TYPE 列には、実行中のタスクに関する説明が表示されます。

- COORDINATOR プロセス（LSP）は、他のプロセスを開始してトランザクションをスケジューリングするバックグラウンド・プロセスです。
- READER プロセスでは、REDO レコードがアーカイブ REDO ログから読み込まれます。
- PREPARER プロセスでは、ブロックの変更を表の変更に変換するための重要な計算が行われます。
- BUILDER プロセスでは、完了したトランザクションのアセンブルが行われます。
- ANALYZER プロセスでは、レコードが検査され、通常は、トランザクションの排除と依存性に関する計算が行われます。
- APPLIER プロセスでは、完了した SQL トランザクションが生成および実行されます。

V$LOGSTDBY ビューを問い合せたときは、特に HIGH_SCN 列に注意してください。この列は、アクティビティのインジケータです。V$LOGSTDBY ビューを問い合せるたびに、この列の内容が変化している間は、作業が進行しています。STATUS 列には、現行のアクティビティの説明テキストが表示されます。次に例を示します。

```
SQL> COLUMN NAME FORMAT A30
SQL> COLUMN VALUE FORMAT A30
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME = 'coordinator state';
NAME                           VALUE
------------------------------ ------------------------------
coordinator state              APPLYING

SQL> COLUMN STATUS FORMAT A50
SQL> COLUMN TYPE FORMAT A12
SQL> SELECT TYPE, HIGH_SCN, STATUS FROM V$LOGSTDBY;
TYPE           HIGH_SCN STATUS
------------ ---------- --------------------------------------------------
COORDINATOR             ORA-16117: 処理中です。
READER                  ORA-16127: 追加のトランザクションが適用されるまで待機して
                        停止しました。

BUILDER          191896 ORA-16116: 使用できる作業はありません
PREPARER         191902 ORA-16117: 処理中です。
ANALYZER         191820 ORA-16120: SCN 0x0000.0002ed4e でのトランザクションに対する
                        依存性を計算中です。
```

```
APPLIER          191209 ORA-16124: トランザクション 1  16  1598 は待機中です。
 .
 .
 .
```

現行のアクティビティに関する情報を取得するもう 1 つの場所は V$LOGSTDBY_STATS ビューです。このビューには、状態とステータスに関する情報が表示されます。DBMS_LOGSTDBY.APPLY_SET プロシージャに関するすべてのオプションにはデフォルト値があります。これらの値（デフォルトまたは設定済み）は、V$LOGSTDBY_STATS ビューで参照できます。さらに、適用されたトランザクションや準備が完了したトランザクションの件数によって、トランザクションが読込みと同時に適用されているかどうかを知ることができます。他の統計値には、システムのすべての部分に関する情報が含まれています。次に例を示します。

```
SQL> COLUMN NAME FORMAT A35
SQL> COLUMN VALUE FORMAT A35
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS
  2> WHERE NAME LIKE 'coordinator%' or NAME LIKE 'transactions%';

NAME                                VALUE
----------------------------------- -----------------------------------
coordinator state                   APPLYING
transactions ready                  7821
transactions applied                7802
coordinator uptime                  73
```

この問合せでは、SQL 適用操作の実行時間とその間に適用されたトランザクションの数が表示されます。適用可能なトランザクションの数も表示され、さらに作業が必要なことを示しています。

アーカイブ REDO ログの適用の遅延

プライマリ・データベースでの適用遅延間隔（秒単位）の指定は、5-6 ページの「REDO ログの適用に対するタイム・ラグの指定」に説明されているように、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・データベースでは同じです。ただし、ロジカル・スタンバイ・データベースでは、プライマリ・データベースが使用不可能になった場合に、次の PL/SQL コマンドを指定して適用遅延間隔を取り消すことができます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_UNSET('APPLY_DELAY');
```

適用された REDO ログ・データ量の確認

REDO ストリームのトランザクション・データは、複数の REDO ログを使用する場合があります。このため、ロジカル・スタンバイ・データベースでは、SQL 適用操作の進捗をレポートするために、個々のアーカイブ REDO ログではなく、REDO データの SCN 範囲が使用されます。

DBA_LOGSTDBY_PROGRESS ビューには、APPLIED_SCN、NEWEST_SCN および READ_SCN に関する情報が表示されます。APPLIED_SCN は、その SCN 以下のコミット済みトランザクションが適用されたことを示します。NEWEST_SCN は、追加のログを受信しなかった場合にデータを適用できる最大 SCN です。この値は通常、リスト内にギャップがない場合、DBA_LOGSTDBY_LOG の MAX(NEXT_CHANGE#)-1 です。

READ_SCN 未満の NEXT_CHANGE# を持つログは不要です。これらのログの情報は、すでに適用されたか、あるいはデータベースに永続的に格納された情報です。これらの SCN の値に関連付けられた時間の値は、ログ時間に基づいた単なる推測値です。これらの値は、SCN の値がプライマリ・データベースに書き込まれた正確な時間を示していません。

次の問合せを使用すると、適用されたログまたは適用されなかったログを確認できます。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT  = 'DD-MON-YY HH24:MI:SS';
Session altered.

SQL> SELECT L.SEQUENCE#, L.FIRST_TIME,
  2    (CASE WHEN L.NEXT_CHANGE# < P.READ_SCN THEN 'YES'
  3          WHEN L.FIRST_CHANGE# < P.APPLIED_SCN THEN 'CURRENT'
  4          ELSE 'NO' END) APPLIED
  5  FROM DBA_LOGSTDBY_LOG L, DBA_LOGSTDBY_PROGRESS P
  6  ORDER BY SEQUENCE#;

 SEQUENCE# FIRST_TIME         APPLIED
---------- ------------------ -------
        24 23-JUL-02 18:19:05 YES
        25 23-JUL-02 18:19:48 YES
        26 23-JUL-02 18:19:51 YES
        27 23-JUL-02 18:19:54 YES
        28 23-JUL-02 18:19:59 YES
        29 23-JUL-02 18:20:03 YES
        30 23-JUL-02 18:20:13 YES
        31 23-JUL-02 18:20:18 YES
        32 23-JUL-02 18:20:21 YES
        33 23-JUL-02 18:32:11 YES
        34 23-JUL-02 18:32:19 CURRENT
        35 23-JUL-02 19:13:20 CURRENT
        36 23-JUL-02 19:13:43 CURRENT
        37 23-JUL-02 19:13:46 CURRENT
        38 23-JUL-02 19:13:50 CURRENT
        39 23-JUL-02 19:13:54 CURRENT
```

```
        40 23-JUL-02 19:14:01 CURRENT
        41 23-JUL-02 19:15:11 NO
        42 23-JUL-02 19:15:54 NO

19 rows selected.
```

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。SYS スキーマ内の表がメンテナンスされることはありません。これは、この SYS スキーマでメンテナンスされるのは、Oracle のメタデータのみであるためです。

SQL 適用操作に障害が発生した場合のエラーは、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。

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。ただし、スタンバイ・システムのファイル・システム構造がプライマリ・システムのファイル・システム構造と異なる場合、DB_FILE_NAME_CONVERT では、プライマリ・データベース上の 1 つ以上のセットのデータ・ファイルのファイル名が、ロジカル・スタンバイ・データベース用のスタンバイ・データベース上のファイル名に変換されないため、エラーが発生する可能性があります。

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. DBMS_LOGSTDBY.GUARD_BYPASS_ON プロシージャを使用してデータベース・ガードを無効にし、ロジカル・スタンバイ・データベースを変更できるようにします。

   ```
   SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
   ```

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> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
   ```

3. DBA_LOGSTDBY_EVENTS ビューを問い合せて、障害が発生した DDL の XIDUSN、XIDSLT および XIDSQN の各値を検索します。次に、これらの値を DBMS_LOGSTDBY.SKIP_TRANSACTION プロシージャに渡します。障害が発生した DDL 文は、常に最後のトランザクションとなります。次に例を示します。

```
SQL> SELECT XIDUSN, XIDSLT, XIDSQN FROM DBA_LOGSTDBY_EVENTS
  2> WHERE EVENT_TIME = (SELECT MAX(EVENT_TIME) FROM DBA_LOGSTDBY_EVENTS);
SQL> EXECUTE DBMS_LOGSTDBY.SKIP_TRANSACTION( /*xidusn*/, /*xidslt*/,
/*xidsqn*/);
```

4. ロジカル・スタンバイ・データベースでログ適用サービスを開始します。

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

ログ適用サービスを再開すると、障害が発生したトランザクションの再実行が試みられます。障害が発生したトランザクションを再実行しないようにするには、値を DBMS_LOGSTDBY.SKIP_TRANSACTION プロシージャに渡して（例の手順 3 を参照）トランザクションをスキップします。

状況によっては、トランザクションの障害原因となる問題を修正して、トランザクションをスキップせずにログ適用サービスを再開できる場合もあります。この例として、使用可能な領域を使い果たした場合があります。次の例には、ログ適用サービスの停止、エラーの修正方法およびログ適用サービスの再開が記述されています。

```
SQL> SELECT * FROM DBA_LOGSTDBY_EVENTS;
EVENT_TIM CURRENT_SCN COMMIT_SCN     XIDUSN     XIDSLT     XIDSQN
--------- ----------- ---------- ---------- ---------- ----------
EVENT

--------------------------------------------------------------------------------
STATUS_CODE
-----------
STATUS

--------------------------------------------------------------------------------
30-JUL-02

      16111
ORA-16111: ログのマイニングと設定の適用

30-JUL-02      200240     200243          1          2     2213
create table bar (x number, y number) tablespace foo
      16204
ORA-16204: DDL は正常に適用されました
```

```
30-JUL-02      200695     200735          1         11     2215
SCOTT.BAR (Oper=INSERT)
       1653
ORA-01653: 表 SCOTT.BAR を拡張できません（%d 分、表領域 %s）

30-JUL-02      200812     200864          1         11     2215
SCOTT.BAR (Oper=INSERT)
       1653
ORA-01653: 表 SCOTT.BAR を拡張できません（%d 分、表領域 %s）
```

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、ログ適用サービスを再開します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
Database altered.
```

ログ適用サービスを再開すると、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。

DML 障害のリカバリ

SKIP_TRANSACTION プロシージャは便利なプロシージャですが、DML 障害のフィルタ処理に使用する場合は注意が必要です。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。このような処理によって、複数の表が破損する場合があります。

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;
   ```

   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。このプロシージャに提供されるリンクには、プライマリ・データベースに対する LOGSTDBY_ADMINISTRATOR ロールが付与されている必要があります。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE('SCOTT','EMP','PRIMARYDB');
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY;
```

3. SCOTT.EMP 表には、INSTANTIATE_TABLE プロシージャ実行時のレコードが格納される（手順 2）ため、SCOTT.DEPT 表にはない部門に関するレコードが格納される可能性があります。

マテリアライズド・ビューのリフレッシュ

プライマリ・データベースでリフレッシュされたマテリアライズド・ビューが、ロジカル・スタンバイ・データベースでそれぞれ自動的にリフレッシュされることはありません。ロジカル・スタンバイ・データベース上のマテリアライズド・ビューをリフレッシュするには、DBMS_LOGSTDBY パッケージの GUARD_BYPASS_ON および GUARD_BYPASS_OFF プロシージャを使用します。次に例を示します。

```
EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
EXECUTE DBMS_MVIEW.REFRESH ( 'BMVIEW', 'F', '',TRUE,FALSE,0,0,0,FALSE);
EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
```

> **関連項目：** DBMS_LOGSTDBY パッケージの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

DBMS_LOGSTDBY.APPLY_SET プロシージャを使用し、TRANSACTION_CONSISTENCY パラメータにデフォルト値 FULL を使用していない場合は、ロジカル・スタンバイ・データベース上のマテリアライズド・ビューをリフレッシュする前に、SQL 適用操作を停止してください。

ロジカル・スタンバイ・データベースのチューニング

次の処理によって、システム・パフォーマンスを向上させます。

- プライマリ・データベースで、表に主キーまたは一意の索引がない場合は、RELY 主キー制約を作成します。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('SYS','SYSTEM','OUTLN','DBSNMP')
  3> MINUS
  3> SELECT DISTINCT TABLE_OWNER, TABLE_NAME FROM DBA_INDEXES
  4> WHERE INDEX_TYPE NOT LIKE ('FUNCTION-BASED%')
  5> MINUS
  6> SELECT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED;
```

次の例は、EMP 表に索引を作成する文を示しています。これらの文は、前述の問合せで戻されたすべての表に対して実行してください。

```
SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
SQL> CREATE INDEX EMPI ON EMP (EMPNO);
SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
```

関連項目： RELY 制約の詳細は、4-3 ページの「アーカイブの有効化とローカルのアーカイブ先の定義」および『Oracle9*i* SQL リファレンス』を参照してください。

- ロジカル・スタンバイ・データベースで、オブジェクトに関するコストベース・オプティマイザ（CBO）の統計情報を定期的に収集します。統計情報は、データ量の変更または列値の変更によって時間の経過とともに古くなります。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/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 適用操作ではワークロードを異なる方法で実行できる場合があります。そのため、ロジカル・スタンバイ・データベースで DBMS_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操作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：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

- トランザクションの一貫性を調整します。

  DBMS_LOGSTDBY.APPLY_SET プロシージャの TRANSACTION_CONSISTENCY パラメータを使用して、ロジカル・スタンバイ・データベースへのトランザクションの適用方法を制御します。デフォルト設定は FULL です。この設定では、プライマリ・データベースでコミットされた順序で、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ます。

  次の値のいずれかを指定します。

  - FULL

    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。このオプションを指定すると、パフォーマンスは最低になります。この値は、デフォルトのパラメータ設定値です。

  - READ_ONLY

    トランザクションは、プライマリ・データベースでのコミット順序に関係なく適用されます。READ_ONLY オプションによって、FULL の値よりも高いパフォーマンスが提供され、SQL の SELECT 文によって、読込み一貫性のある結果が戻されます。この値は、レポートの生成にロジカル・スタンバイ・データベースを使用している場合、特に役立ちます。

  - NONE

    トランザクションは、プライマリ・データベースでのコミット順序に関係なく適用され、読込み一貫性のある結果も提供されません。この値の結果は、3 つの値の中で最高のパフォーマンスとなります。ロジカル・スタンバイ・データベースを読み込んでいるアプリケーションに、トランザクションの順序についての条件がない場合、このオプションは効果があります。

    注意： READ_ONLY および NONE オプションは、ALTER DATABASE GUARD ALL が設定されている場合にのみ使用してください。

- パラレル実行処理の最大数を調整します。

  PARALLEL_MAX_SERVERS 初期化パラメータを使用して、インスタンスに対するパラレル実行処理とパラレル・リカバリ処理の最大数を調整します。このパラメータのデフォルト値は、CPU_COUNT、PARALLEL_AUTOMATIC_TUNING および PARALLEL_ADAPTIVE_MULTI_USER の各初期化パラメータの値から導出されます。ロジカル・スタンバイ・データベースでは、このパラメータの値は、5 未満に設定しないでください。

DBMS_LOGSTDBY.APPLY_SET プロシージャの MAX_SERVERS パラメータを使用して、ログ適用サービスが使用するパラレル・サーバーの数を制限できます。このパラメータのデフォルト値は、PARALLEL_MAX_SERVERS 初期化パラメータの値と同じ値に設定されます。このパラメータを明示的に設定する場合は、5 未満の値または PARALLEL_MAX_SERVERS 初期化パラメータの値を超える値に設定しないでください。

インスタンスに対するパラレル実行処理とパラレル・リカバリ処理の最大数を増加すると、実行およびリカバリ操作を高速化できます。ただし、この改善は、プロセスごとの追加システム・リソースの消費とバランスをとる必要があります。

- ロジカル・スタンバイ・データベースでのメモリー使用量を制御します。

  DBMS_LOGSTDBY.APPLY_SET プロシージャの MAX_SGA パラメータを使用して、ログ適用サービスが REDO キャッシュに使用する共有プールの最大領域を設定できます。デフォルトでは、ログ適用サービスが使用する共有プールは、全体の 25 パーセントです。一般的には、ログ適用サービスが使用する共有プールのサイズまたは領域の増加によって、ロジカル・スタンバイ・データベースのパフォーマンスは向上します。

10

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一般的な例を説明します。使用例は、ユーザー固有の環境に適応できるように、それぞれ手順に従った詳細例として示します。表 10-1 は、この章で説明する使用例の一覧です。

表 10-1 Data Guard の使用例

参照先	使用例
10-2 ページ	ロールの推移に最適なスタンバイ・データベースの選択
10-16 ページ	タイム・ラグのあるフィジカル・スタンバイ・データベースの使用
10-19 ページ	タイム・ラグのあるフィジカル・スタンバイ・データベースへのスイッチオーバー
10-21 ページ	ネットワーク障害のリカバリ
10-22 ページ	NOLOGGING 句を指定した後のリカバリ

ロールの推移に最適なスタンバイ・データベースの選択

各スタンバイ・データベースは、1 つのプライマリ・データベースのみと対応付けられています。ただし、各プライマリ・データベースは、複数のフィジカルまたはロジカル・スタンバイ・データベースをサポートできます。この使用例では、フェイルオーバーまたはスイッチオーバー操作に最適なスタンバイ・データベースの選択に必要な情報の判断方法を示します。

ほとんどのロールの推移（フェイルオーバーまたはスイッチオーバー）では、構成にフィジカル・スタンバイ・データベースが含まれている場合、最適なフィジカル・スタンバイ・データベースを使用したロールの推移をお薦めします。これには次の理由があります。

- ロジカル・スタンバイ・データベースには、プライマリ・データベース内にあるデータのサブセットのみが含まれている可能性があります。
- 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では、既存のフィジカル・スタンバイ・データベースは、ロールの推移が完了した後も Data Guard 構成に継続して含まれるように、新しいプライマリ・データベースのコピーから再作成する必要があります。

これらの制限があるため、ロジカル・スタンバイ・データベースをロールの推移のターゲットとして考慮するのは、次のような特別な状況の場合に限定してください。

- 構成にロジカル・スタンバイ・データベースのみが含まれている場合
- スタンバイ・データベースのプライマリ・ロールへの迅速なフェイルオーバーが重要で、構成内の最新のロジカル・スタンバイ・データベースが構成内の最新のフィジカル・スタンバイ・データベースに比較して大幅に更新されている場合

フィジカルまたはロジカル・スタンバイ・データベースの使用を決定すると、ロールの推移のターゲットとして選択する特定のスタンバイ・データベースは、スタンバイ・ロケーションにある最新のプライマリ・データベースの変更量とそれらの変更がスタンバイ・データベースに適用された量によって判断されます。プライマリ・データベースはスイッチオーバー操作中もアクセス可能な状態であるため、データの消失はありません。また、スイッチオーバー中に使用されるスタンバイ・データベースの選択によって影響を受けるのは、スイッチオーバーの完了に必要な時間のみです。ただし、フェイルオーバーの場合、スタンバイ・データベースの選択では、データ消失に関するリスクの増加と、スタンバイ・データベースのプライマリ・ロールへの推移に必要な時間との間でトレードオフが必要となります。

例 : フェイルオーバー操作に最適なフィジカル・スタンバイ・データベース

障害時に DBA にとって最も重要な作業は、より迅速でかつ安全なのはプライマリ・データベースの修正か、スタンバイ・データベースへのフェイルオーバーかを判断することです。フェイルオーバー操作が必要と判断し、複数のフィジカル・スタンバイ・データベースが構成されている場合は、フェイルオーバー操作のターゲットとして最適なフィジカル・スタンバイ・データベースを選択する必要があります。最適なスタンバイ・データベースの判断に影響を与える環境上の要因は様々ですが、この使用例では、データ消失の査定を明確にするために、これらの要因が等しいと仮定します。

この使用例では、HQ プライマリ・データベースと 2 つのフィジカル・スタンバイ・データベース（SAT と NYC）で構成した Data Guard 構成を前提に説明します。HQ データベースは最大可用性保護モードで動作し、スタンバイ・データベースはそれぞれ 3 つのスタンバイ REDO ログで構成されています。

> **関連項目 :** フィジカル・スタンバイ・データベースの最大可用性保護モードの詳細は、5-3 ページの「データ保護モード」を参照してください。

表 10-2 に、この使用例で使用されているデータベースに関する情報を示します。

表 10-2 フィジカル・スタンバイ・データベースの例で使用される識別子

識別子	HQ データベース	SAT データベース	NYC データベース
場所	サンフランシスコ	シアトル	ニューヨーク市
データベース名	HQ	HQ	HQ
インスタンス名	HQ	SAT	NYC
初期化パラメータ・ファイル	hq_init.ora	sat_init.ora	nyc_init.ora
制御ファイル	hq_cf1.f	sat_cf1.f	nyc_cf1.f
データ・ファイル	hq_db1.f	sat_db1.f	nyc_db1.f
オンライン REDO ログ・ファイル 1	hq_log1.f	sat_log1.f	nyc_log1.f
オンライン REDO ログ・ファイル 2	hq_log2.f	sat_log2.f	nyc_log2.f
スタンバイ REDO ログ・ファイル 1	hq_srl1.f	sat_srl1.f	nyc_srl1.f
スタンバイ REDO ログ・ファイル 2	hq_srl2.f	sat_srl2.f	nyc_srl2.f
スタンバイ REDO ログ・ファイル 3	hq_srl3.f	sat_srl3.f	nyc_srl3.f
プライマリ保護モード	最大可用性	該当なし	該当なし
スタンバイ保護モード	該当なし	最大可用性（同期）	最大パフォーマンス（非同期）

表 10-2 フィジカル・スタンバイ・データベースの例で使用される識別子（続き）

識別子	HQ データベース	SAT データベース	NYC データベース
ネットワーク・サービス名（クライアント定義）	hq_net	sat_net	nyc_net
リスナー	hq_listener	sat_listener	nyc_listener

注意： ピーク・ワークロード時に HQ から NYC に REDO データを同期させて送信することは、プライマリ・データベースのパフォーマンスに影響を与える可能性があるため、ニューヨーク市のデータベースは最大パフォーマンス・モードで動作しています。ただし、ニューヨーク市のスタンバイ・データベースは、スタンバイ REDO ログを使用しているため、フェイルオーバー操作のための有効な候補とみなされています。

プライマリ・サイトがあるサンフランシスコでイベントが発生し、適時に修正できないような被害を受けたと仮定します。スタンバイ・データベースの 1 つにフェイルオーバーする必要があります。複数のスタンバイ・データベース構成を設定した DBA が、どのスタンバイ・データベースにフェイルオーバーするのが適切であるかを必ずしも決定できる状態にあるとは限りません。したがって、各スタンバイ・サイトで、プライマリ・サイト同様に障害時リカバリ計画を立てることが必要になります。障害時リカバリ・チームの各メンバーは、障害時リカバリ計画を知っており、実行する手順を認識している必要があります。この使用例では、フェイルオーバー操作のターゲットとなるスタンバイ・データベースの判断に必要な情報を識別します。

障害時リカバリ・チームに情報を提供する 1 つの方法は、各スタンバイ・サイトに ReadMe ファイルを用意しておく方法です。この ReadMe ファイルは DBA によって作成およびメンテナンスされ、次の方法を記述している必要があります。

- DBA としてのローカル・データベース・サーバーへのログオン方法
- スタンバイ・データベースが存在する各システムへのログオン方法
- システム間にはファイアウォールが存在する可能性があるため、ファイアウォールを通過する手順の取得方法
- DBA として他のデータベース・サーバーへログオンする方法
- 最も新しいログが適用されているスタンバイ・データベースの識別方法
- スタンバイ・データベース・フェイルオーバー処理の実行方法
- クライアント・アプリケーションが、元のプライマリ・データベースではなく、新しいプライマリ・データベースにアクセスできるようなネットワーク設定の構成方法

関連項目： ReadMe ファイルのサンプルは、付録 E を参照してください。

スタンバイ・データベースの選択に際しては、2 つの重要な考慮事項があります。1 つは最新の REDO データを受信したスタンバイ・データベース、もう 1 つは最多の REDO ログを適用したスタンバイ・データベースです。

次の手順に従って、フィジカル・スタンバイ・データベースのみで構成されている場合に、どのスタンバイ・データベースがフェイルオーバーに最適な候補であるかを決定します。常に、最高の保護レベルを提供しているスタンバイ・データベースから考慮します。この使用例では、シアトルのスタンバイ・データベースが最大可用性保護レベルで動作しているため、このデータベースが最高の保護レベルを提供しています。

手順 1 SAT フィジカル・スタンバイ・データベースに接続する

次のような SQL 文を発行します。

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL AS SYSDBA;
```

手順 2 REDO ログで使用可能なカレント REDO データの量を判断する

次のように、V$MANAGED_STANDBY ビューの列を問い合せます。

```
SQL> SELECT THREAD#, SEQUENCE#, BLOCK#, BLOCKS
     2> FROM V$MANAGED_STANDBY WHERE STATUS='RECEIVING';
   THREAD#  SEQUENCE#     BLOCK#     BLOCKS
---------- ---------- ---------- ----------
         1         14        234         16
```

このスタンバイ・データベースは、プライマリ・データベースから 249 ブロックの REDO データを受信しています。受信したブロック数を計算するには、BLOCKS 列の値を BLOCK# 列の値に加えて 1 を引きます。1 を引く理由は、ブロック番号 234 が受信した 16 ブロックに含まれているためです。

注意： プライマリ・データベースが使用不可能であった期間によっては、前述の問合せで指定した行が戻らない場合があります。これは、RFS プロセスがネットワークの切断を検出し、プロセス自体が終了する場合があるためです。この場合の最善策は、REDO データを同期モードで受信するように構成されたスタンバイ・データベースを常に選択することです。

手順 3 SAT データベースに適用した、または適用を保留しているアーカイブ REDO ログのリストを取得する

V$ARCHIVED_LOG ビューを問い合せます。

```
SQL> SELECT SUBSTR(NAME,1,25) FILE_NAME, SEQUENCE#, APPLIED
  2> FROM V$ARCVHIVED_LOG ORDER BY SEQUENCE#;
FILE_NAME                  SEQUENCE# APP
------------------------- ---------- ---
/oracle/dbs/hq_sat_2.log           2 YES
/oracle/dbs/hq_sat_3.log           3 YES
/oracle/dbs/hq_sat_4.log           4 YES
/oracle/dbs/hq_sat_5.log           5 YES
/oracle/dbs/hq_sat_6.log           6 YES
/oracle/dbs/hq_sat_7.log           7 YES
/oracle/dbs/hq_sat_8.log           8 YES
/oracle/dbs/hq_sat_9.log           9 YES
/oracle/dbs/hq_sat_10.log         10 YES
/oracle/dbs/hq_sat_11.log         11 YES
/oracle/dbs/hq_sat_13.log         13  NO
```

この出力は、アーカイブ REDO ログ 11 がスタンバイ・データベースに完全に適用されたことを示しています（出力例では、ログ 11 の行を確認しやすいように太字で示していますが、実際は太字ではありません）。

SEQUENCE# 列の順序番号にギャップがあることにも注意してください。この例の場合は、SAT スタンバイ・データベースでアーカイブ REDO ログ番号 12 が欠落していることを示しています。

手順 4 NYC データベースに接続して、NYC データベースが SAT スタンバイ・データベースよりも新しいかどうかを判断する

次のような SQL 文を発行します。

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL AS SYSDBA;
```

手順 5 REDO ログで使用可能なカレント REDO データの量を判断する

次のように、V$MANAGED_STANDBY ビューの列を問い合せます。

```
SQL> SELECT THREAD#, SEQUENCE#, BLOCK#, BLOCKS
     2> FROM V$MANAGED_STANDBY WHERE STATUS='RECEIVING';
   THREAD#  SEQUENCE#     BLOCK#     BLOCKS
---------- ---------- ---------- ----------
         1         14        157         93
```

このスタンバイ・データベースも、プライマリ・データベースから 249 ブロックの REDO 情報を受信しています。受信したブロック数を計算するには、BLOCKS 列の値を BLOCK# 列の値に加えて 1 を引きます。1 を引く理由は、ブロック番号 157 が受信した 93 ブロックに含まれているためです。

手順 6 NYC データベースに適用した、または適用を保留しているアーカイブ REDO ログのリストを取得する

V$ARCHIVED_LOG ビューを問い合せます。

```
SQL> SELECT SUBSTR(NAME,1,25) FILE_NAME, SEQUENCE#, APPLIED
  2> FROM V$ARCVHIVED_LOG ORDER BY SEQUENCE#;
FILE_NAME                  SEQUENCE# APP
------------------------- ---------- ---
/oracle/dbs/hq_nyc_2.log            2 YES
/oracle/dbs/hq_nyc_3.log            3 YES
/oracle/dbs/hq_nyc_4.log            4 YES
/oracle/dbs/hq_nyc_5.log            5 YES
/oracle/dbs/hq_nyc_6.log            6 YES
/oracle/dbs/hq_nyc_7.log            7 YES
/oracle/dbs/hq_nyc_8.log            8  NO
/oracle/dbs/hq_nyc_9.log            9  NO
/oracle/dbs/hq_nyc_10.log          10  NO
/oracle/dbs/hq_nyc_11.log          11  NO
/oracle/dbs/hq_nyc_12.log          12  NO
/oracle/dbs/hq_nyc_13.log          13  NO
```

この出力は、アーカイブ REDO ログ 7 がスタンバイ・データベースに完全に適用されたことを示しています（出力例では、ログ 7 の行を確認しやすいように太字で示していますが、実際は太字ではありません）。

この場所で受信した REDO データはシアトルの場合よりも多数ですが、スタンバイ・データベースに適用されたデータはシアトルの場合より少数です。

手順 7 最適なターゲット・スタンバイ・データベースを選択する

ほとんどの場合、フェイルオーバーのターゲットとして選択するフィジカル・スタンバイ・データベースは、データ消失のリスクと、ロールの推移の実行に必要な時間とのバランスを備えている必要があります。この情報を分析して、この使用例で最適なフェイルオーバー候補を決定するときに、次のことを考慮してください。

- フェイルオーバー操作時のデータ消失のリスクを最小にするには、最適なターゲット・スタンバイ・データベースとして NYC データベースを選択します。これは、手順 5 と 6 からわかるように、リカバリ可能な REDO ログが NYC サイトに多数存在するためです。

- フェイルオーバー操作時のプライマリ・データベース停止時間を最小にするには、最適なターゲット・スタンバイ・データベースとして SAT データベースを選択します。SAT データベースの方が適切な候補である理由は、手順 2 から 6 でわかるように、NYC データベースよりもアーカイブ REDO ログを 5 つ多く適用している点です。ただし、欠落しているアーカイブ・ログ（この例ではアーカイブ REDO ログ 12）のコピーの取得と適用が不可能な場合は、SAT データベースを NYC データベースと同じ最新の状態にすることはできません。したがって、未適用のデータ（この例ではログ 12、13 およびログ 14 の一部）は失われます。

最適なターゲット・スタンバイ・データベースは、ビジネス要件に基づいて選択します。

手順 8 選択したスタンバイ・データベースを最新の状態にする

ビジネス要件に基づいて最適なターゲットとして SAT を選択した場合は、次の手順を実行します。

1. オペレーティング・システムのユーティリティを使用して、欠落しているアーカイブ REDO ログを手動で取得します。この例では UNIX の cp コマンドを使用します。この使用例の SAT データベースでは、アーカイブ REDO ログ 12 が欠落しています。NYC データベースはこのアーカイブ REDO ログを受信しているため、次のように、SAT データベースにコピーできます。

   ```
   % cp /net/nyc/oracle/dbs/hq_nyc_12.log /net/sat/oracle/dbs/hq_sat_12.log
   ```

2. 次の順序番号に対して、部分的なアーカイブ REDO ログが存在しているかどうかを判断します。この例では、次の順序番号は 14 です。次の UNIX コマンドで SAT データベースのディレクトリを検索し、hq_sat_14.log というアーカイブ REDO ログが存在しているかどうかを調べます。

   ```
   % ls -l /net/sat/oracle/dbs/hq_sat_14.log
   /net/sat/oracle/dbs/hq_sat_14.log: No such file or directory
   ```

   SAT スタンバイ・データベースはスタンバイ REDO ログを使用しているため、部分的なアーカイブ REDO ログが存在していることはありません。

3. 取得したアーカイブ REDO ログを登録します（ログ適用サービスを停止する必要はありません）。

   ```
   SQL> ALTER DATABASE REGISTER PHYSICAL LOGFILE '/oracle/dbs/hq_sat_12.log';
   ```

4. V$ARCHIVED_LOG ビューを再度問い合せて、アーカイブ REDO ログが正常に適用されたことを確認します。

```
SQL> SELECT SUBSTR(NAME,1,25) FILE_NAME, SEQUENCE#, APPLIED
  2> FROM V$ARCVHIVED_LOG ORDER BY SEQUENCE#;

FILE_NAME                  SEQUENCE# APP
------------------------- ---------- ---
/oracle/dbs/hq_sat_2.log            2 YES
/oracle/dbs/hq_sat_3.log            3 YES
/oracle/dbs/hq_sat_4.log            4 YES
/oracle/dbs/hq_sat_5.log            5 YES
/oracle/dbs/hq_sat_6.log            6 YES
/oracle/dbs/hq_sat_7.log            7 YES
/oracle/dbs/hq_sat_8.log            8 YES
/oracle/dbs/hq_sat_9.log            9 YES
/oracle/dbs/hq_sat_10.log          10 YES
/oracle/dbs/hq_sat_11.log          11 YES
/oracle/dbs/hq_sat_12.log          12 YES
/oracle/dbs/hq_sat_13.log          13 YES
```

ビジネス要件に基づいて最適なターゲットとして NYC を選択した場合は、次の手順を実行します。

1. 次の順序番号に対して、部分的なアーカイブ REDO ログが存在しているかどうかを判断します。次の UNIX コマンドで NYC データベースのディレクトリを検索して、次の順序の名前が付いた hq_nyc_14 というアーカイブ REDO ログが存在しているかどうかを調べます。

```
% ls -l /net/nyc/oracle/dbs/hq_nyc_14.log
/net/nyc/oracle/dbs/hq_nyc_14.log: No such file or directory
```

   NYC スタンバイ・データベースはスタンバイ REDO ログを使用しているため、部分的なアーカイブ REDO ログが存在していることはありません。

2. ログ適用サービスを開始して、最新のログを適用します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
  2> DISCONNECT FROM SESSION;
```

3. V$ARCHIVED_LOG ビューを再度問い合せて、アーカイブ REDO ログが正常に適用されたことを確認します。

```
SQL> SELECT SUBSTR(NAME,1,25) FILE_NAME, SEQUENCE#, APPLIED
  2> FROM V$ARCVHIVED_LOG ORDER BY SEQUENCE#;
FILE_NAME                 SEQUENCE# APP
------------------------- ---------- ---
/oracle/dbs/hq_nyc_2.log           2 YES
/oracle/dbs/hq_nyc_3.log           3 YES
/oracle/dbs/hq_nyc_4.log           4 YES
/oracle/dbs/hq_nyc_5.log           5 YES
/oracle/dbs/hq_nyc_6.log           6 YES
/oracle/dbs/hq_nyc_7.log           7 YES
/oracle/dbs/hq_nyc_8.log           8 YES
/oracle/dbs/hq_nyc_9.log           9 YES
/oracle/dbs/hq_nyc_10.log         10 YES
/oracle/dbs/hq_nyc_11.log         11 YES
/oracle/dbs/hq_nyc_12.log         12  NO
/oracle/dbs/hq_nyc_13.log         13  NO
```

アーカイブ REDO ログの適用では、完了までに時間がかかる場合があります。したがって、次のようにすべてのアーカイブ REDO ログが指定されるまで待機する必要があります。

```
SQL> SELECT SUBSTR(NAME,1,25) FILE_NAME, SEQUENCE#, APPLIED
  2> FROM V$ARCVHIVED_LOG ORDER BY SEQUENCE#;

FILE_NAME                 SEQUENCE# APP
------------------------- ---------- ---
/oracle/dbs/hq_nyc_2.log           2 YES
/oracle/dbs/hq_nyc_3.log           3 YES
/oracle/dbs/hq_nyc_4.log           4 YES
/oracle/dbs/hq_nyc_5.log           5 YES
/oracle/dbs/hq_nyc_6.log           6 YES
/oracle/dbs/hq_nyc_7.log           7 YES
/oracle/dbs/hq_nyc_8.log           8 YES
/oracle/dbs/hq_nyc_9.log           9 YES
/oracle/dbs/hq_nyc_10.log         10 YES
/oracle/dbs/hq_nyc_11.log         11 YES
/oracle/dbs/hq_nyc_12.log         12 YES
/oracle/dbs/hq_nyc_13.log         13 YES
```

手順 9 フェイルオーバー操作を実行する

ログ適用サービスを停止し、選択したフィジカル・スタンバイ・データベースをプライマリ・ロールにフェイルオーバーする準備が整いました。

> **関連項目：** フィジカル・スタンバイ・データベースに対するフェイルオーバー操作の実行方法の詳細は、7-15 ページの「フィジカル・スタンバイ・データベースが関与するフェイルオーバー操作」を参照してください。

例：フェイルオーバー操作に最適なロジカル・スタンバイ・データベース

障害時にロジカル・スタンバイ・データベースのみが使用可能な場合、重要な作業は、どのロジカル・スタンバイ・データベースがフェイルオーバー操作に最適かを判断することです。最適なターゲット・スタンバイ・データベースの判断に影響を与える環境上の要因は様々ですが、この使用例では、データ消失の査定を明確にするために、これらの要因が等しいと仮定します。

> **関連項目：** ロジカル・スタンバイ・データベースの最大可用性保護モードの詳細は、5-3 ページの「データ保護モード」を参照してください。

この使用例では、HQ プライマリ・データベースと 2 つのロジカル・スタンバイ・データベース（SAT と NYC）で構成した Data Guard 構成を前提に説明します。表 10-3 に、これらのデータベースに関する情報を示します。

表 10-3 ロジカル・スタンバイ・データベースの例で使用される識別子

識別子	HQ データベース	SAT データベース	NYC データベース
場所	サンフランシスコ	シアトル	ニューヨーク市
データベース名	HQ	SAT	NYC
インスタンス名	HQ	SAT	NYC
初期化パラメータ・ファイル	hq_init.ora	sat_init.ora	nyc_init.ora
制御ファイル	hq_cf1.f	sat_cf1.f	nyc_cf1.f
データ・ファイル	hq_db1.f	sat_db1.f	nyc_db1.f
オンライン REDO ログ・ファイル 1	hq_log1.f	sat_log1.f	nyc_log1.f
オンライン REDO ログ・ファイル 2	hq_log2.f	sat_log2.f	nyc_log2.f
データベース・リンク（クライアント定義）	hq_link	sat_link	nyc_link

表 10-3 ロジカル・スタンバイ・データベースの例で使用される識別子（続き）

識別子	HQ データベース	SAT データベース	NYC データベース
ネットワーク・サービス名（クライアント定義）	hq_net	sat_net	nyc_net
リスナー	hq_listener	sat_listener	nyc_listener

次の手順に従って、ロジカル・スタンバイ・データベースのみで構成されている場合に、どのスタンバイ・データベースがフェイルオーバーに最適な候補であるかを決定します。

手順 1 SAT ロジカル・スタンバイ・データベースに接続する

次のような SQL 文を発行します。

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL AS SYSDBA;
```

手順 2 SAT データベースに適用された最大の SCN と適用可能な最大（最新）の SCN を判断する

DBA_LOGSTDBY_PROGRESS ビューで、次の列を問い合せます。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     144059     144059
```

手順 3 SAT データベースに適用した、または適用を保留しているアーカイブ REDO ログのリストを取得する

DBA_LOGSTDBY_LOG ビューを問い合せます。

```
SQL> SELECT SUBSTR(FILE_NAME,1,25) FILE_NAME, SUBSTR(SEQUENCE#,1,4) "SEQ#",
  2> FIRST_CHANGE#, NEXT_CHANGE#, TO_CHAR(TIMESTAMP, 'HH:MI:SS') TIMESTAMP,
  3> DICT_BEGIN BEG, DICT_END END, SUBSTR(THREAD#,1,4) "THR#"
  4> FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

FILE_NAME                 SEQ# FIRST_CHANGE# NEXT_CHANGE# TIMESTAM BEG END THR#
------------------------- ---- ------------- ------------ -------- --- --- ----
/oracle/dbs/hq_sat_2.log  2           101579       101588 11:02:57 NO  NO  1
/oracle/dbs/hq_sat_3.log  3           101588       142065 11:02:01 NO  NO  1
/oracle/dbs/hq_sat_4.log  4           142065       142307 11:02:09 NO  NO  1
/oracle/dbs/hq_sat_5.log  5           142307       142739 11:02:47 YES YES 1
/oracle/dbs/hq_sat_6.log  6           142739       143973 12:02:09 NO  NO  1
/oracle/dbs/hq_sat_7.log  7           143973       144042 01:02:00 NO  NO  1
/oracle/dbs/hq_sat_8.log  8           144042       144051 01:02:00 NO  NO  1
/oracle/dbs/hq_sat_9.log  9           144051       144054 01:02:15 NO  NO  1
/oracle/dbs/hq_sat_10.log 10          144054       144057 01:02:20 NO  NO  1
```

```
/oracle/dbs/hq_sat_11.log 11          144057       144060 01:02:25 NO  NO  1
/oracle/dbs/hq_sat_13.log 13          144089       144147 01:02:40 NO  NO  1
```

手順 2 に記載されているログ 11 の SCN144059 は、FIRST_CHANGE# 列の値 144057 と NEXT_CHANGE# 列の値 144060 の間にあります。これは、ログ 11 が現在適用中であることを示します（出力例では、ログ 11 の行を確認しやすいように太字で示していますが、実際は太字ではありません）。SEQ# 列の順序番号にギャップがあることにも注意してください。この例の場合は、SAT データベースでアーカイブ REDO ログ 12 が欠落していることを示しています。

手順 4 NYC データベースに接続する

次のような SQL 文を発行します。

```
SQL> CONNECT SYS/CHANGE_ON_INSTALL AS SYSDBA;
```

手順 5 NYC データベースに適用された最大の SCN と適用可能な最大の SCN を判断する

DBA_LOGSTDBY_PROGRESS ビューで、次の列を問い合せます。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;
APPLIED_SCN NEWEST_SCN
----------- ----------
     143970     144146
```

手順 6 NYC データベースで処理済みまたは現在処理が保留中のログのリストを取得する

次のような SQL 文を発行します。

```
SQL> SELECT SUBSTR(FILE_NAME,1,25) FILE_NAME, SUBSTR(SEQUENCE#,1,4) "SEQ#",
  2> FIRST_CHANGE#, NEXT_CHANGE#, TO_CHAR(TIMESTAMP, 'HH:MI:SS')  TIMESTAMP,
  3> DICT_BEGIN BEG, DICT_END END, SUBSTR(THREAD#,1,4) "THR#"
  4> FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;

FILE_NAME                 SEQ# FIRST_CHANGE# NEXT_CHANGE# TIMESTAM BEG END THR#
------------------------- ---- ------------- ------------ -------- --- --- ----
/oracle/dbs/hq_nyc_2.log  2           101579       101588 11:02:58 NO  NO  1
/oracle/dbs/hq_nyc_3.log  3           101588       142065 11:02:02 NO  NO  1
/oracle/dbs/hq_nyc_4.log  4           142065       142307 11:02:10 NO  NO  1
/oracle/dbs/hq_nyc_5.log  5           142307       142739 11:02:48 YES YES 1
/oracle/dbs/hq_nyc_6.log  6           142739       143973 12:02:10 NO  NO  1
/oracle/dbs/hq_nyc_7.log  7           143973       144042 01:02:11 NO  NO  1
/oracle/dbs/hq_nyc_8.log  8           144042       144051 01:02:01 NO  NO  1
/oracle/dbs/hq_nyc_9.log  9           144051       144054 01:02:16 NO  NO  1
/oracle/dbs/hq_nyc_10.log 10          144054       144057 01:02:21 NO  NO  1
/oracle/dbs/hq_nyc_11.log 11          144057       144060 01:02:26 NO  NO  1
/oracle/dbs/hq_nyc_12.log 12          144060       144089 01:02:30 NO  NO  1
/oracle/dbs/hq_nyc_13.log 13          144089       144147 01:02:41 NO  NO  1
```

手順 5 に記載されているログ 6 の SCN143970 は、FIRST_CHANGE# 列の値 142739 と NEXT_CHANGE# 列の値 143973 の間にあります。これは、ログ 6 が現在適用中であることを示します（出力例では、ログ 6 の行を確認しやすいように太字で示していますが、実際は太字ではありません）。処理が残っているログの順序番号にギャップがないことも注意してください。

手順 7 最適なターゲット・スタンバイ・データベースを選択する

ほとんどの場合、フェイルオーバーのターゲットとして選択するロジカル・スタンバイ・データベースは、データ消失のリスクと、ロールの推移の実行に必要な時間とのバランスを備えている必要があります。この情報を分析して、この使用例で最適なフェイルオーバー候補を決定するときに、次のことを考慮してください。

- フェイルオーバー操作時のデータ消失のリスクを最小にするには、最適なターゲット・スタンバイ・データベースとして NYC データベースを選択します。これは、手順 5 と 6 からわかるように、リカバリ可能な REDO ログが NYC サイトに多数存在するためです。
- フェイルオーバー操作時のプライマリ・データベース停止時間を最小にするには、最適なターゲット・スタンバイ・データベースとして SAT データベースを選択します。このデータベースの方が適切な候補である理由は、手順 2 から 6 の問合せでわかるように、SAT データベースの方がアーカイブ REDO ログを NYC データベースより 5 つ多く適用しているためです（ただし、NYC データベースによるアーカイブ REDO ログの受信にかかった遅延（ラグ）は 1 秒ほどです）。ただし、欠落しているアーカイブ・ログ（この例ではログ 12）のコピーの取得と適用が不可能な場合は、SAT データベースを NYC データベースと同じ最新の状態にすることはできません。したがって、リカバリされないデータ（この例ではログ 12、13 およびログ 14 の一部）は失われます。

最適なターゲット・スタンバイ・データベースは、ビジネス要件に基づいて選択します。

手順 8 選択したスタンバイ・データベースを最新の状態にする

ビジネス要件に基づいて最適なターゲットとして SAT を選択した場合は、次の手順を実行します。

1. オペレーティング・システムのユーティリティを使用して、欠落しているアーカイブ REDO ログを手動で取得します。この使用例の SAT データベースでは、アーカイブ REDO ログ 12 が欠落しています。NYC データベースはこのアーカイブ REDO ログを受信しているため、次のように、SAT データベースにコピーできます。

```
%cp /net/nyc/oracle/dbs/hq_nyc_12.log
/net/sat/oracle/dbs/hq_sat_12.log
```

2. 次の順序番号に対して、部分的なアーカイブ REDO ログが存在しているかどうかを判断します。この例では、次の順序番号は 14 です。次の UNIX コマンドで SAT データベースのディレクトリを表示して、hq_sat_14.log というアーカイブ REDO ログが存在しているかどうかを調べます。

```
%ls -l /net/sat/oracle/dbs/hq_sat_14.log
-rw-rw----   1 oracle    dbs  333280 Feb 12  1:03 hq_sat_14.log
```

3. ログ適用サービスを停止し、取得した欠落しているアーカイブ REDO ログと部分的なアーカイブ REDO ログの両方を登録します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/oracle/dbs/hq_sat_12.log';
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/oracle/dbs/hq_sat_14.log';
```

4. ログ適用サービスを開始して、最新のログを適用します。

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

5. DBA_LOGSTDBY_PROGRESS ビューを問い合せて SAT データベースで適用済みの最大の SCN を判断し、APPLIED_SCN 列の値が NEWEST_SCN 列の値と等しいかどうかを調べます。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     144205     144205
```

SCN の値が一致しているため、プライマリ・データベースの現在のログと、SAT データベースに適用された最後のログの間の遅延（ラグ）がなくなったことを確認できます。

ビジネス要件に基づいて最適なターゲットとして NYC を選択した場合は、次の手順を実行します。

1. 次の順序番号に対して、部分的なアーカイブ REDO ログが存在しているかどうかを判断します。この例では、次の順序番号は 14 です。次の UNIX コマンドで NYC データベースのディレクトリを表示して、hq_nyc_14 というアーカイブ REDO ログが存在しているかどうかを調べます。

```
%ls -l /net/nyc/oracle/dbs/hq_nyc_14.log
-rw-rw----   1 oracle    dbs  333330 Feb 12  1:03 hq_nyc_14.log
```

2. 部分的なアーカイブ REDO ログを NYC データベースに登録します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
SQL> ALTER DATABASE REGISTER LOGICAL LOGFILE '/oracle/dbs/hq_nyc_14.log';
```

3. ログ適用サービスを開始して、最新のログを適用します。

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

4. DBA_LOGSTDBY_PROGRESS ビューを問い合せて NYC データベースで適用済みの最大の SCN を判断し、APPLIED_SCN 列の値が NEWEST_SCN 列の値と等しいかどうかを調べます。

```
SQL> SELECT APPLIED_SCN, NEWEST_SCN FROM DBA_LOGSTDBY_PROGRESS;

APPLIED_SCN NEWEST_SCN
----------- ----------
     144205     144205
```

SCN の値が一致しているため、プライマリ・データベースの現在のログと、NYC データベースで受信され適用された最後のログの間の遅延（ラグ）がなくなったことを確認できます。

手順 9 フェイルオーバーを実行する

ログ適用サービスを停止し、選択したロジカル・スタンバイ・データベースをプライマリ・ロールにフェイルオーバーする準備が整いました。

> **関連項目：** フェイルオーバー操作の実行方法の詳細は、7-23 ページの「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操作」を参照してください。

タイム・ラグのあるフィジカル・スタンバイ・データベースの使用

スタンバイ・データベースで管理リカバリを実行している場合、デフォルトでは、スタンバイ・データベースがプライマリ・データベースから到着した REDO ログを自動的に適用します。しかし、ログをすぐに適用しないようにする場合もあります。これは、プライマリ・サイトでの REDO ログのアーカイブと、スタンバイ・サイトでのそのログの適用との間にタイム・ラグを設定する場合です。タイム・ラグによって、プライマリ・サイトからスタンバイ・サイトへ破損したまたは誤ったデータが送られてもスタンバイ・データベースを保護できます。

たとえば、毎晩プライマリ・データベースでバッチ・ジョブを実行するとします。誤ってバッチ・ジョブを 2 回実行してしまい、しかも 2 回の実行が完了するまでその誤操作に気づかなかったとします。理想的には、バッチ・ジョブを開始する前の状態までデータベースをロールバックする必要があります。タイム・ラグのあるスタンバイ・データベースを持つプライマリ・データベースを使用すると、リカバリに役立ちます。タイム・ラグのあるスタンバイ・データベースにフェイルオーバーして、それを新規のプライマリ・データベースとして使用できます。

タイム・ラグのあるスタンバイ・データベースを作成するには、プライマリ・データベースの初期化パラメータ・ファイルにある LOG_ARCHIVE_DEST_*n* 初期化パラメータの DELAY 属性を使用します。アーカイブ REDO ログはプライマリ・サイトからスタンバイ・サイトに自動的にコピーされますが、ログはすぐにはスタンバイ・データベースに適用されません。ログが適用されるのは、指定した時間が経過した後です。

この例では 4 時間のタイム・ラグが採用されています。この例に含まれる項目は、次のとおりです。

- フィジカル・スタンバイ・データベースでのタイム・ラグの設定
- タイム・ラグのあるフィジカル・スタンバイ・データベースへのフェイルオーバー
- タイム・ラグのあるフィジカル・スタンバイ・データベースへのスイッチオーバー

この例では、読者が通常のスタンバイ・データベースの作成手順を理解していることを前提にしています。したがって、この例で示す手順では詳細を省略しています。

> **関連項目：** フィジカル・データベースの作成方法の詳細は、第 3 章を参照してください。

フィジカル・スタンバイ・データベースでのタイム・ラグの設定

タイム・ラグのあるフィジカル・スタンバイ・データベースを作成するには、プライマリ・データベースの LOG_ARCHIVE_DEST_*n* 初期化パラメータを変更して、スタンバイ・データベースの遅延を設定します。次に、4 時間の遅延を LOG_ARCHIVE_DEST_*n* 初期化パラメータに追加する方法の例を示します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=stdby DELAY=240';
```

この DELAY 属性は、スタンバイ・サイトでは 4 時間が経過しないと、アーカイブ REDO ログがリカバリに使用されないことを示します。アーカイブ REDO ログがスタンバイ・サイトへ正常に転送されると、設定された時間間隔（分で表示）が始まります。REDO 情報は通常どおりスタンバイ・データベースに送信され、ディスクに書き込まれます。

> **関連項目：** DBMS_LOGSTDBY パッケージを使用してロジカル・スタンバイ・データベースでタイム・ラグを設定する方法については、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

タイム・ラグのあるフィジカル・スタンバイ・データベースへのフェイルオーバー

REDO ログの適用を遅延するように構成したスタンバイ・データベースは、プライマリ・データベースでのユーザーのミスやデータ破損のリカバリに使用できます。ほとんどの場合は、タイム・ディレイが設定されているスタンバイ・データベースを問い合せて、プライマリ・データベースの修正（誤って削除した表の内容のリカバリなど）に必要なデータを取得できます。プライマリ・データベースの被害が不明な場合やプライマリ・データベースの修正にかなりの時間が必要な場合は、タイム・ディレイが設定されているスタンバイ・データベースへのフェイルオーバーも考慮できます。

不注意によって、バックアップ・ファイルがプライマリ・データベースに 2 回適用され、プライマリ・データベースの修正にかなりの時間が必要であると仮定します。ここでは、REDO ログの適用が遅延されているフィジカル・スタンバイ・データベースへのフェイルオーバーを選択します。これによって、スタンバイ・データベースを、問題が発生する以前の時点のプライマリ・ロールに推移させますが、一部のデータは消失することになります。処理の手順は、次のとおりです。

1. タイム・ディレイが設定されているフィジカル・スタンバイ・データベースで適切な SQL 文を発行することで、フェイルオーバー操作を開始します。

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   SQL> ALTER DATABASE ACTIVATE PHYSICAL STANDBY DATABASE SKIP STANDBY LOGFILE;
   SQL> SHUTDOWN IMMEDIATE;
   SQL> STARTUP
   ```

   ACTIVATE 文は、スタンバイ・データベースをプライマリ・ロールに即時に推移させ、スタンバイ・ロケーションに存在している可能性のある他の REDO データの適用は行いません。このコマンドを使用する場合は、スタンバイ・ロケーションでのデータ消失のコストと、プライマリ・データベースの完全修正に必要な停止時間の延長のバランスを注意深く調整する必要があります。

2. この新しいプライマリ・データベースのコピーから、構成内にある他のすべてのスタンバイ・データベースを再作成します。

タイム・ラグのあるフィジカル・スタンバイ・データベースへのスイッチオーバー

すべての REDO ログは、使用可能になった時点でスタンバイ・サイトに送信されます。したがって、タイム・ディレイがスタンバイ・データベースに指定されている場合でも、SQL 文 ALTER DATABASE RECOVER MANAGED STANDBY を使用して遅延を無視することで、スタンバイ・データベースを最新の状態にできます。

注意： 論理的なエラーのリカバリには、スイッチオーバー操作ではなく、フェイルオーバー操作を実行する必要があります。

次の手順に従って、タイム・ディレイが設定されたフィジカル・スタンバイ・データベース（タイム・ラグをバイパスする）へのスイッチオーバーを実行する方法を説明します。この例では、プライマリ・データベースはニューヨークに、スタンバイ・データベースはボストンにあると仮定します。

手順 1 すべてのアーカイブ REDO ログを、タイム・ラグをバイパスしている元の（タイム・ディレイが設定された）スタンバイ・データベースに適用する

スタンバイ・データベースですべての REDO ログが適用されるまで、スイッチオーバーは開始しません。遅延を解除することによって、スタンバイ・データベースは適用操作の終了を待たずに処理を進めます。

遅延を解除するには、次の SQL 文を発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE NODELAY
  2> DISCONNECT FROM SESSION THROUGH LAST SWITCHOVER;
```

手順 2 プライマリ・データベースとスタンバイ・データベースでの読取りまたは更新アクティビティを停止する

スイッチオーバー操作を開始するには、排他的なデータベース・アクセス権限が必要です。ユーザーにプライマリおよびスタンバイ・データベースからログオフするように求めるか、V$SESSION ビューを問い合せてデータベースに接続しているユーザーを識別し、スイッチオーバー文を実行する SQL*Plus セッションを除くすべてのオープン・セッションをクローズします。

関連項目： ユーザーの管理の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

手順 3 プライマリ・データベースをフィジカル・スタンバイ・ロールにスイッチオーバーする

ニューヨークのプライマリ・データベースで、次の文を実行します。

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

この文は次のことを行います。

- プライマリ・データベースをクローズし、アクティブなセッションを終了します。
- アーカイブされていないログをアーカイブし、それらをボストンのスタンバイ・データベースに適用します。
- 最後にアーカイブされるログのヘッダーに end-of-redo マーカーを追加します。
- 現行の制御ファイルのバックアップを作成します。
- 現行の制御ファイルをスタンバイ制御ファイルに変換します。

手順 4 元のプライマリ・インスタンスを停止し、データベースをマウントせずに起動する

ニューヨークにある元のプライマリ・データベースで次の文を実行します。

```
SQL> SHUTDOWN NORMAL;
SQL> STARTUP NOMOUNT;
```

手順 5 元のプライマリ・データベースをフィジカル・スタンバイ・データベース・ロールでマウントする

次の文を実行して、ニューヨークにある元のプライマリ・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マウントします。

```
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

手順 6 元のスタンバイ・データベースをプライマリ・ロールに切り替える

次の SQL 文を発行します。

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

手順 7 新しいプライマリ・データベース・インスタンスを停止して再起動する

次の SQL 文を発行します。

```
SQL> SHUTDOWN;
SQL> STARTUP PFILE=Failover.ora;
```

ネットワーク障害のリカバリ

次の手順に従って、ネットワーク障害後のリカバリ方法を説明します。

手順 1 ネットワーク障害を識別する

V$ARCHIVE_DEST ビューは、ネットワーク・エラーを格納し、到達できないスタンバイ・データベースを識別します。プライマリ・データベースでは、ネットワーク障害が発生したアーカイブ・ログ宛先に対して次の SQL 文を実行します。次に例を示します。

```
SQL> SELECT DEST_ID, STATUS, ERROR FROM V$ARCHIVE_DEST WHERE DEST_ID = 2;

DEST_ID    STATUS    ERROR
---------- --------- -----------------------------------------------------
         2  ERROR     ORA-12224: TNS:リスナーがありません。
```

問合せの結果、スタンバイ・データベースへのアーカイブにエラーがあり、原因は「TNS:リスナーがありません。」であることがわかりました。スタンバイ・サイトでリスナーが起動されているかどうかを調べる必要があります。リスナーが停止している場合は、起動してください。

手順 2 プライマリ・データベースが停止するのを回避する

ネットワーク問題を迅速に解決できない場合、およびスタンバイ・データベースが必須の宛先として指定されている場合は、次のいずれかを実行して、データベースが停止しないようにしてください。

- 必須のアーカイブ先へのアーカイブを遅延します。

  ```
  SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = DEFER;
  ```

  ネットワーク問題が解決すると、アーカイブ先をもう一度使用可能にできます。

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

- 必須のアーカイブ先をオプションに変更します。

  ```
  SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1
    2> OPTIONAL REOPEN=60';
  ```

  ネットワーク問題が解決すると、アーカイブ先をオプションから必須に戻すことができます。

  ```
  SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1
    2> MANDATORY REOPEN=60';
  ```

手順 3 カレント REDO ログをアーカイブする

プライマリ・データベースで、カレント REDO ログをアーカイブします。

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

ネットワークが再開され、フィジカル・スタンバイ・データベースで管理リカバリ操作が再開されると、ログ適用サービスによるアーカイブ・ギャップの自動的な検出と解決が可能となります。

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 REDO ログ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 ログ・レコードは REDO ログに書き込まれます。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、ログ適用操作の再開で、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。『Oracle9*i* データベース管理者ガイド』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 適用操作で NOLOGGING 句を使用した操作の REDO ログ・レコードが検出されると、そのレコードはスキップされ、後続のレコードから変更の適用が続行されます。その後、NOLOGGING を実際に使用して更新されたレコードの 1 つにアクセスしようとすると、「ORA-01403 データが見つかりません」エラーが戻ります。

NOLOGGING 句を指定した後のリカバリには、9-9 ページの「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」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。

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 REDO ログがフィジカル・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 データ・ブロックに障害が発生しました（ファイル番号 1、ブロック番号 2521）
ORA-01110: データ・ファイル 1:'/oracle/dbs/stdby/tbs_1.dbf'
ORA-26040: データ・ブロックが NOLOGGING オプションを使用してロードされました。
```

NOLOGGING 句が指定された後でリカバリするには、記録されていない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# の値の方が大きいので、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・ファイルを、次のようにバックアップする

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 スタンバイ・データベースで、管理リカバリを再開する

次の SQL 文を発行します。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM
  2>SESSION;
```

管理リカバリ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: アーカイブ・ログ standby1 をオープンできません。
ORA-27037: ファイル・ステータスを取得できません。
SVR4 Error: 2: No such file or directory
Additional information: 3
ORA-01547: 警告 : RECOVER は成功しましたが OPEN RESETLOGS が次のエラーを受け取りました。
ORA-01152: ファイル 1 は十分に古いバックアップからリストアされていません。
ORA-01110: データ・ファイル 1:'/oracle/dbs/stdby/tbs_1.dbf'
```

ORA-00308 エラーが表示された場合は、別の端末のウィンドウから次の文を発行して、リカバリを取り消してください。

```
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・ギャップ内の 1 つ以上のログが正常に適用されなかった場合に戻ります。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 3 を繰り返します。

> **関連項目：** アーカイブ・ギャップの手動による解決方法については、B-5 ページの「手動によるアーカイブ・ギャップの解決」を参照してください。

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 ビューを問い合せて、**システム変更番号 (SCN)**、または Oracle データベース・サーバーが無効な 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。

> **関連項目：** V$DATAFILE ビューの詳細は、第 14 章の「V$DATAFILE」および『Oracle9*i* データベース・リファレンス』を参照してください。

第 II 部

リファレンス

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。詳細は、Oracle9*i* のマニュアルを参照してください。

この部は、次の章で構成されています。

11

初期化パラメータ

この章では、現行のデータベースに関する初期化パラメータの表示方法およびサーバー・パラメータ・ファイル（SPFILE）の変更方法について説明し、さらに Data Guard 構成のインスタンスに影響を与える初期化パラメータに関する参照情報も提供します。

すべてのデータベース初期化パラメータは、初期化パラメータ・ファイル（PFILE）またはサーバー・パラメータ・ファイル（SPFILE）に含まれています。パラメータは、初期化パラメータ・ファイルまたはサーバー・パラメータ・ファイルに指定するかわりに、実行時に `ALTER SYSTEM SET` 文または `ALTER SESSION SET` 文を使用して動的に変更することもできます。

注意： Data Guard Broker を使用する場合は、サーバー・パラメータ・ファイルを使用する必要があります。また、初期化パラメータ・ファイルまたはサーバー・パラメータ・ファイルに記録されない実行時のパラメータ変更は、永続的な設定ではないため、次回データベースを再起動すると消失します。

初期化パラメータの表示

次の表は、現行の初期化パラメータ設定を表示するために使用できる方法を示しています。

方法	説明
SHOW PARAMETERS SQL*Plus コマンド	このコマンドを発行すると、現在有効なパラメータ値が表示されます。
V$PARAMETER ビュー	このビューを問い合せると、現在有効なパラメータ値が表示されます。
V$PARAMETER2 ビュー	このビューを問い合せると、現在有効なパラメータ値が表示されます。このビューの出力は、V$PARAMETER ビューの出力と内容は同じですが、より確認しやすい構成になっています。
V$SPPARAMETER ビュー	このビューを問い合せると、サーバー・パラメータ・ファイルの現在の内容が表示されます。サーバー・パラメータ・ファイルがインスタンスによって使用されていない場合は、NULL 値が戻されます。

次の例では、CONTROL_FILES パラメータ設定について V$PARAMETER ビューを問い合せています。

```
SQL> SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'CONTROL_FILES';
```

サーバー・パラメータ・ファイルの変更

サーバー・パラメータ・ファイルはバイナリ・ファイルのため、手動では編集できません。サーバー・パラメータ・ファイルの値を変更するには、ファイルを編集可能なフォーマットにエクスポートして編集した後、サーバー・パラメータ・ファイルにインポートするか、ALTER SYSTEM SET 文を使用してサーバー・パラメータ値を変更する必要があります。これらの方法については、次の各項で説明します。

編集可能なファイルへのサーバー・パラメータ・ファイルのエクスポート

サーバー・パラメータ・ファイルを変更するには、次の手順を実行します。

1. SQL 文 CREATE PFILE を使用して、サーバー・パラメータ・ファイルをテキストの初期化パラメータ・ファイルにエクスポートします。後続の例 1 と例 2 を参照してください。

   初期化パラメータ・ファイルはテキスト・ファイルのため、手動で編集できます。この文を実行するには、SYSDBA または SYSOPER のシステム権限が必要です。エクスポート・ファイルはデータベース・サーバー・システムに作成されます。このファイルには、パラメータに関連するコメントが、パラメータ設定と同じ行に含まれています。

2. 初期化パラメータ・ファイルを編集します。

3. SQL 文 CREATE SPFILE を使用して、編集した初期化パラメータ・ファイルから新しいサーバー・パラメータ・ファイルを作成します。後続の例 3 と例 4 を参照してください。

   この文を実行するには、SYSDBA または SYSOPER のシステム権限が必要です。

例 1

この例では、ファイル名を指定せずに、サーバー・パラメータ・ファイルからテキストの初期化パラメータ・ファイルを作成します。

```
CREATE PFILE FROM SPFILE;
```

ファイルの名前が指定されていないため、オペレーティング・システム固有の初期化パラメータ・ファイル名が使用され、オペレーティング・システム固有のデフォルトのサーバー・パラメータ・ファイルから初期化パラメータ・ファイルが作成されます。

例 2

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

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

例 3

この例では、初期化パラメータ・ファイル /u01/oracle/dbs/test_init.ora からサーバー・パラメータ・ファイルを作成します。SPFILE 名が指定されていないため、ファイルは、オペレーティング・システム固有のデフォルトのサーバー・パラメータ・ファイル名およびデフォルトのファイルの位置を使用して作成されます。

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

例 4

この例では、サーバー・パラメータ・ファイルおよび初期化パラメータ・ファイルの両方の名前を指定して、サーバー・パラメータ・ファイルを作成します。

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

SQL ALTER SYSTEM SET によるサーバー・パラメータ・ファイルの変更

サーバー・パラメータ・ファイルをエクスポートし、編集した後でインポートするかわりに、前述のように SQL 文 ALTER SYSTEM SET を使用して初期化パラメータの値を変更できます。サーバー・パラメータ・ファイルに変更を適用するには、必ず SCOPE 句を使用します。

デフォルトでは、インスタンスの起動にサーバー・パラメータ・ファイルが使用された場合は有効範囲が BOTH に設定され、インスタンスの起動に初期化パラメータ・ファイルが使用された場合は有効範囲が MEMORY に設定されます。次の例では、ローカル・アーカイブ・ログの新しい宛先をサーバー・パラメータ・ファイルに追加しています。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_4=
  2> 'LOCATION=/disk1/oracle/oradata/payroll/',
  3> 'MANDATORY', 'REOPEN=2'  SCOPE=SPFILE;
```

Data Guard 構成内のインスタンスの初期化パラメータ

次のリストは、Data Guard 環境のインスタンスに影響を与える初期化パラメータを示しています。

- ARCHIVE_LAG_TARGET
- COMPATIBLE
- CONTROL_FILE_RECORD_KEEP_TIME
- CONTROL_FILES
- DB_FILE_NAME_CONVERT
- DB_FILES
- DB_NAME
- FAL_CLIENT
- FAL_SERVER
- LOCK_NAME_SPACE
- LOG_ARCHIVE_DEST_n
- LOG_ARCHIVE_DEST_STATE_n
- LOG_ARCHIVE_FORMAT
- LOG_ARCHIVE_MAX_PROCESSES
- LOG_ARCHIVE_MIN_SUCCEED_DEST
- LOG_ARCHIVE_START

- LOG_ARCHIVE_TRACE
- LOG_FILE_NAME_CONVERT
- LOG_PARALLELISM
- PARALLEL_MAX_SERVERS
- REMOTE_ARCHIVE_ENABLE
- SHARED_POOL_SIZE
- SORT_AREA_SIZE
- STANDBY_ARCHIVE_DEST
- STANDBY_FILE_MANAGEMENT
- USER_DUMP_DEST

次の項では、各パラメータについて説明し、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。スタンバイ・データベース・ロールに適用されるパラメータの場合、ほとんどのパラメータはフィジカル・スタンバイ・データベースおよびロジカル・スタンバイ・データベースの両方に関係があります。異なる点がある場合は示しています。

> **関連項目：** Data Guard 固有ではないパラメータの詳細や、これらの初期化パラメータの型、デフォルト値および構文については、『Oracle9*i* データベース・リファレンス』を参照してください。また、初期化パラメータの設定方法の詳細は、Oracle のオペレーティング・システム固有のマニュアルを参照してください。

ARCHIVE_LAG_TARGET

説明

データ消失量を制限し、指定した時間（秒単位）の経過後にログの切替えを強制して、スタンバイ・データベースの可用性を高めます。スタンバイ・データベースは、ARCHIVE_LAG_TARGET パラメータの値の時間範囲より前に生成された REDO ログを消失することがなくなります。

ロール

プライマリ・データベース・ロールに適用されます。

例

次の例では、ログ・スイッチ間隔を 30 分（一般的な値）に設定しています。

```
ARCHIVE_LAG_TARGET = 1800
```

COMPATIBLE

説明

データベースの互換性を制御します。Data Guard Broker、ロジカル・スタンバイ・データベースおよびフィジカル・スタンバイ・データベースの拡張機能を使用するには、9.0.0.0.0 以上に設定します。プライマリ・データベースとスタンバイ・データベースでは、このパラメータを常に同じ値に設定します。値が異なる場合は、プライマリ・データベースからスタンバイ・データベースへの REDO ログのアーカイブが実行できないことがあります。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、データベースの互換レベルを '9.2.0.0.0' に設定しています。

```
COMPATIBLE = '9.2.0.0.0'
```

CONTROL_FILE_RECORD_KEEP_TIME

説明

制御ファイル内の再使用可能レコードが再使用可能になるまでの最小日数を指定します。このパラメータを使用して、指定された期間内に、制御ファイル内（アーカイブ・ログ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。このパラメータの値の範囲は 0 ～ 365 日です。このパラメータが 0 に設定されている場合、再使用可能レコードは必要に応じて再使用されます。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、制御ファイル内の再使用可能レコードが再使用可能になるまでの最小日数を 20 日に設定しています。

```
CONTROL_FILE_RECORD_KEEP_TIME = 20
```

CONTROL_FILES

説明

制御ファイルの名前を指定します。複数の制御ファイル名を指定するには、各ファイル名をカンマで区切ります。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システム上にある場合、スタンバイ・データベースのこのパラメータは、常にプライマリ・データベースの CONTROL_FILES パラメータとは異なる値に設定します。スタンバイ・データベースの CONTROL_FILES パラメータで指定するファイル名は、スタンバイ・ロケーションに存在している必要があります。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、データベース・インスタンスに対して 2 つの制御ファイルを指定しています。

```
CONTROL_FILE = ("/disk1/oracle/oradata/payroll/control01.ctl",
"/disk1/oracle/oradata/payroll/control02.ctl")
```

DB_FILE_NAME_CONVERT

説明

プライマリ・データベース上にあるデータ・ファイルのファイル名をスタンバイ・データベース上のファイル名に変換します。スタンバイ・データベースの制御ファイルは、プライマリ・データベースの制御ファイルのコピーであるため、スタンバイ・データベースのファイル名がプライマリ・データベースのファイル名と異なる場合は、このパラメータを使用してファイル名を変換する必要があります。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上にある場合は、異なるパス名を使用する必要があります。

ロール

フィジカル・スタンバイ・データベース・ロールに適用されます。

例

次の例は、/dbs/t1/（プライマリ・データベース）から /dbs/t1/stdby（スタンバイ・データベース）、および dbs/t2/（プライマリ・データベース）から dbs/t2/stdby（スタンバイ・データベース）へのパスの変換を示しています。

```
DB_FILE_NAME_CONVERT = ('/dbs/t1/','/dbs/t1/stdby','dbs/t2/ ','dbs/t2/stdby')
```

DB_FILES

説明

このデータベース用にオープンできるデータベース・ファイルの最大数を指定します。このパラメータは、プライマリ・データベースとスタンバイ・データベースで同じ値を指定する必要があります。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、このデータベース・インスタンスに対して最大 300 のデータベース・ファイルをオープンできるように指定しています。

```
DB_FILES = 300
```

DB_NAME

説明

最高 8 文字のデータベース識別子を指定します。フィジカル・スタンバイ・データベースの場合は、DB_NAME パラメータをプライマリ・データベースの初期化ファイルに設定されている値と同じ値に設定します。ロジカル・スタンバイ・データベースの場合は、DB_NAME パラメータをプライマリ・データベースの初期化ファイルに設定されている値とは異なる値に設定します。ロジカル・スタンバイ・データベースのデータベース名は、4-20 ページの「ロジカル・スタンバイ・データベースのデータベース名のリセット」に説明されているように DBNEWID（nid）ユーティリティを使用して変更します。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、データベース名を Sales に指定しています。

```
DB_NAME = Sales
```

次の例は、DBNEWID ユーティリティを使用してロジカル・スタンバイ・データベース名を設定する方法を示しています。このコマンドを発行する前に、データベースをマウントする必要があります。

```
nid TARGET=SYS/CHANGE_ON_INSTALL@LogicalSDB DBNAME=SalesLSDB SETNAME=YES
```

FAL_CLIENT

説明

フェッチ・アーカイブ・ログ（FAL）サーバーが FAL クライアントを参照するときに使用する FAL クライアント名を割り当てます。このパラメータは、FAL サーバーがスタンバイ・データベースへの接続に使用する Oracle Net サービス名です。この Oracle Net サービス名は、FAL クライアントを指し示すように、FAL サーバー（プライマリ・データベース）で正しく構成されている必要があります。FAL_CLIENT パラメータは FAL_SERVER パラメータに依存しているため、この 2 つのパラメータは同時に構成または変更してください。このパラメータは、スタンバイ・サイトに設定します。

ロール

管理リカバリ・モードのフィジカル・スタンバイ・データベース・ロールに適用されます。

例

次の例では、FAL クライアントに Oracle Net サービス名 StandbyDB を割り当てています。

```
FAL_CLIENT = StandbyDB
```

FAL_SERVER

説明

スタンバイ・データベースがフェッチ・アーカイブ・ログ（FAL）サーバーへの接続に使用する Oracle Net サービス名を割り当てます。このパラメータは、スタンバイ・システムに設定します。

ロール

管理リカバリ・モードのフィジカル・スタンバイ・データベース・ロールに適用されます。

例

次の例では、FAL サーバーに Oracle Net サービス名 `PrimaryDB` を割り当てています。

```
FAL_SERVER = PrimaryDB
```

LOCK_NAME_SPACE

説明

分散ロック・マネージャ（DLM）がロック名を生成するために使用するネームスペースを指定します。スタンバイ・データベース名が同一システムまたは同一クラスタ上のプライマリ・データベース名と同じ場合は、このパラメータを各初期化パラメータ・ファイル内で一意の値に設定します。

> **注意：** スタンバイおよびプライマリ・データベースが同じシステムにある場合に LOCK_NAME_SPACE パラメータに同じ値を設定すると、ORA-1102 エラーを受け取ります。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、スタンバイ初期化パラメータ・ファイルの LOCK_NAME_SPACE を payroll2 に設定しています。

```
LOCK_NAME_SPACE = payroll2
```

LOG_ARCHIVE_DEST_*n*

説明

アーカイブ・ログの宛先とログ転送サービスの属性を定義します。このパラメータの詳細は、第 5 章および第 12 章を参照してください。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例は、スタンバイ・データベースへのリモート・アーカイブ・ログの宛先を示しています。

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=payroll2 OPTIONAL REOPEN=180'
```

LOG_ARCHIVE_DEST_STATE_*n*

説明

LOG_ARCHIVE_DEST_*n* パラメータで指定された宛先の状態を指定します。使用可能な値は、次のとおりです。

- ENABLE を指定すると、有効なログ・アーカイブ宛先を後続のアーカイブ操作（自動または手動）に使用できます。これがデフォルトです。
- DEFER を指定すると、有効な宛先情報および属性は保持されますが、ENABLE オプションでアーカイブを再度使用可能にするまで、この宛先はアーカイブ操作から除外されます。
- ALTERNATE を指定すると、宛先は使用可能になりませんが、別の宛先への通信に障害が発生した場合に、この宛先が使用可能になります。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、LOG_ARCHIVE_DEST_STATE_2 の状態を ENABLE に設定しています。

```
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
```

LOG_ARCHIVE_FORMAT

説明

アーカイブ REDO ログのファイル名の形式を指定します。STANDBY_ARCHIVE_DEST および LOG_ARCHIVE_FORMAT の初期化パラメータが連結して、スタンバイ・データベースのアーカイブ REDO ログの完全修飾されたファイル名が生成されます。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、データベース ID（%d）、スレッド（%t）および順序番号（%s）を使用して、アーカイブ REDO ログのファイル名の形式を指定しています。

```
LOG_ARCHIVE_FORMAT = 'log%d_%t_%s.arc'
```

LOG_ARCHIVE_MAX_PROCESSES

説明

データベース・サーバーが起動するアーカイバ・バックグラウンド・プロセスの数を指定します。LOG_ARCHIVE_START パラメータが TRUE の場合、この値はインスタンス起動時に評価されます。それ以外の場合、このパラメータはアーカイバ・プロセスが呼び出されたときに評価されます。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、ログ・アーカイバ・プロセスの数を 2 に設定しています。

```
LOG_ARCHIVE_MAX_PROCESSES = 2
```

LOG_ARCHIVE_MIN_SUCCEED_DEST

説明

プライマリ・データベースのログ・ライター・プロセスがオンライン REDO ログを再利用する前に REDO ログを正常に受信する必要がある、宛先の最小数を定義します。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、正常に受信する必要がある宛先の最小数を 2 に設定しています。

```
LOG_ARCHIVE_MIN_SUCCEED_DEST = 2
```

LOG_ARCHIVE_START

説明

インスタンス起動時に、アーカイブを自動にするか、手動にするかを示します。いっぱいになったログ・グループの自動アーカイブを使用可能にするには、初期化パラメータ・ファイルの LOG_ARCHIVE_START を TRUE に設定します。いっぱいになったオンライン REDO ログ・グループの自動アーカイブを使用不可能にするには、LOG_ARCHIVE_START を FALSE に設定します。このパラメータは、サーバー・パラメータ・ファイルには指定できません。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、LOG_ARCHIVE_START を TRUE に設定しています。

```
LOG_ARCHIVE_START = TRUE
```

LOG_ARCHIVE_TRACE

説明

ARC*n* プロセスと LGWR プロセス、プライマリ・データベースでのフォアグラウンド・プロセス、およびスタンバイ・データベースでの RFS プロセスと FAL サーバー・プロセスによって生成されるトレース出力を制御します。スタンバイ・サイトに対するアーカイブ REDO ログの進行状況を確認できます。Oracle データベース・サーバーは、プライマリ・データベースから受信した REDO ログの監査証跡をトレース・ファイルに書き込みます。トレース・ファイルの位置は、USER_DUMP_DEST パラメータを使用して指定します。使用可能な値は、次のとおりです。

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する（デフォルト設定）
1	REDO ログ・ファイルのアーカイブを追跡する
2	アーカイブ REDO ログの宛先ごとにアーカイブ状況を追跡する
4	アーカイブ操作のフェーズを追跡する
8	アーカイブ REDO ログの宛先のアクティビティを追跡する
16	アーカイブ REDO ログの宛先の詳細なアクティビティを追跡する
32	アーカイブ REDO ログの宛先のパラメータ修正を追跡する
64	ARC*n* プロセスの状態のアクティビティを追跡する
128	FAL サーバー・プロセスのアクティビティを追跡する
256	将来のリリースでサポート予定
512	非同期 LGWR アクティビティを追跡する
1024	RFS の物理的なクライアントを追跡する
2048	ARC*n* または RFS のハートビートを追跡する

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、LOG_ARCHIVE_TRACE を 1 に設定しています。

```
LOG_ARCHIVE_TRACE = 1
```

LOG_FILE_NAME_CONVERT

説明

プライマリ・データベース上にあるログのファイル名をスタンバイ・データベース上にあるログのファイル名に変換します。ログをプライマリ・データベースに追加した場合は、対応するログをスタンバイ・データベースに追加する必要があります。スタンバイ・データベースが更新されると、このパラメータを使用してプライマリ・データベースのログ・ファイル名からスタンバイ・データベースのログ・ファイル名に変換されます。このパラメータは、スタンバイ・データベースがプライマリ・データベースとは異なるパス名を使用している場合に必要です。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上にある場合は、異なるパス名を使用する必要があります。

ロール

フィジカル・スタンバイ・データベース・ロールに適用されます。

例

次の例は、2 つのパスの変換を示しています。/dbs/t1/（プライマリ・データベース）から /dbs/t1/stdby（スタンバイ・データベース）、および dbs/t2/（プライマリ・データベース）から dbs/t2/stdby（スタンバイ・データベース）に変換します。

```
LOG_FILE_NAME_CONVERT = ('/dbs/t1/','/dbs/t1/stdby','dbs/t2/ ','dbs/t2/stdby')
```

LOG_PARALLELISM

説明

REDO データをパラレルで生成できるように、REDO データ割当ての並行性レベルを指定します。プライマリ・データベースおよびすべてのロジカル・スタンバイ・データベースに対して、この値を 1 に設定してください。デフォルト値は 1 です。

ロール

ロジカル・スタンバイ・データベース・ロールにのみ適用されます。

例

次の例では、LOG_PARALLELISM パラメータを 1 に設定しています。

```
LOG_PARALLELISM = 1
```

PARALLEL_MAX_SERVERS

説明

このパラメータには、ロジカル・スタンバイ・データベースのログ適用サービスで動作できるパラレル・サーバーの最大数を指定します。このパラメータは、フィジカル・スタンバイ・データベースでは使用されません。

ログ適用サービスは、パラレル問合せプロセスを使用して処理を実行し、パラレル適用アルゴリズムを使用して高水準のデータベース適用パフォーマンスを維持します。ロジカル・スタンバイ・データベースには、少なくとも 5 つのパラレル問合せプロセスが必要です。したがって、PARALLEL_MAX_SERVERS パラメータの値は、5 以上の値に設定する必要があります。

ロール

プライマリ・データベース・ロールおよびロジカル・スタンバイ・データベース・ロールに適用されます。

例

次の例では、PARALLEL_MAX_SERVERS 初期化パラメータを 10 に設定しています。

```
PARALLEL_MAX_SERVERS = 10
```

REMOTE_ARCHIVE_ENABLE

説明

リモート宛先への REDO ログの送信、およびリモート REDO ログの受信を可能または不可能にします。

使用される値は次のとおりです。

- TRUE

  リモート宛先への REDO ログの送信またはリモート REDO ログの受信を可能にします。Data Guard 環境でのプライマリ・データベースおよびスタンバイ・データベースで、このパラメータを TRUE に設定すると、プライマリ・データベースが REDO ログをスタンバイ・データベースに送信したり、プライマリ・データベースからアーカイブする REDO ログをスタンバイ・データベースが受信できるようになります。

- FALSE

  REDO ログの送受信を不可能にします。

- SEND

  プライマリ・データベースによるスタンバイ・データベースへの REDO ログの送信を可能にします。

- RECEIVE

  スタンバイ・データベースによるプライマリ・データベースからの REDO ログの受信を可能にします。

リモート REDO ログの送受信を個別に可能または不可能にするには、SEND 値と RECEIVE 値を使用します。SEND 値と RECEIVE 値を両方使用すると、このパラメータを TRUE に設定する場合と同じになります。Oracle Real Application Clusters データベースのすべてのインスタンスには、同じ REMOTE_ARCHIVE_ENABLE 値が含まれている必要があります。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、リモート・ログの送受信を可能にしています。

```
REMOTE_ARCHIVE_ENABLE = true
```

SHARED_POOL_SIZE

説明

共有プールのサイズをバイト単位で指定します。ロジカル・スタンバイ・データベースのログ適用サービスは、共有プールの System Global Area（SGA）を使用して REDO ログから読み込んだ情報を処理します。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。デフォルトでは、SHARED_POOL_SIZE パラメータに設定した値の 4 分の 1 がログ適用サービスに使用されます。このデフォルト値は、PL/SQL プロシージャ DBMS_LOGSTDBY.APPLY_SET を使用して変更できます。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、共有プール・サイズを 33MB に設定しています。

```
SHARED_POOL_SIZE = 33554432
```

SORT_AREA_SIZE

説明

Oracle データベース・サーバーがソート操作に使用する最大メモリー量をバイト単位で指定します。このパラメータに値を設定すると、データベースがオープンしているときに SELECT * FROM V$PARAMETER 文を実行できます。このパラメータによって、データベースがオープンしていない場合に一時表領域なしでソートが試みられたときに発生するエラーを防止できます。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、ソート領域サイズを 65536 バイトに設定しています。

```
SORT_AREA_SIZE = 65536
```

STANDBY_ARCHIVE_DEST

説明

スタンバイ・データベースがこのパラメータを使用して、プライマリ・データベースから受信したオンライン REDO ログのアーカイブ位置を判断します。RFS プロセスは、この値を LOG_ARCHIVE_FORMAT 値と併用して、スタンバイ・データベースの REDO ログの完全修飾されたファイル名を生成します。生成されたファイル名は、LOG_ARCHIVE_DEST_*n* パラメータの TEMPLATE 属性によって上書きされることに注意してください。

このパラメータの値は、V$ARCHIVE_DEST データ・ディクショナリ・ビューを問い合せることによって確認できます。

ロール

スタンバイ・データベース・ロールに適用されます。

例

次の例では、スタンバイ・データベース上の REDO ログのファイル・パスを '/u01/oracle/oradata/archive' に指定しています。

```
STANDBY_ARCHIVE_DEST = '/u01/oracle/oradata/archive'
```

STANDBY_FILE_MANAGEMENT

説明

自動スタンバイ・ファイル管理を使用可能または使用不可能にします。

このパラメータに使用可能な値は、次のとおりです。

- MANUAL

  自動スタンバイ・ファイル管理を使用不可能にします。

- AUTO

  自動スタンバイ・ファイル管理を使用可能にします。

AUTO に設定すると、このパラメータは、プライマリ・サイトと同じファイル名を使用して、スタンバイ・サイトのデータ・ファイル名の作成および削除を自動化します。MANUAL に設定すると、データ・ファイルの作成と削除が自動的に実行されず、管理リカバリが終了する場合があります。

スタンバイ・データベースのファイル・パスがプライマリ・データベースとは異なる場合は、スタンバイ・サイトで確実に正しいファイルが作成されるように、このパラメータと DB_FILE_NAME_CONVERT 初期化パラメータを併用してください。このパラメータは、RAW デバイスのデータ・ファイルのファイル名をサポートしないことに注意してください。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、自動スタンバイ・ファイル管理を使用可能に設定しています。

```
STANDBY_FILE_MANAGEMENT = TRUE
```

USER_DUMP_DEST

説明

データベース・サーバーがユーザー・プロセスのかわりにデバッグ・トレース・ファイルを書き込む場合に、その書込み先ディレクトリ・パスを指定します。LOG_ARCHIVE_TRACE パラメータを使用して、トレース情報を制御できます。

ロール

プライマリ・データベース・ロールおよびスタンバイ・データベース・ロールに適用されます。

例

次の例では、データベース・トレース・ファイルの位置を '/u01/oracle/oradata/utrc' に指定しています。

```
USER_DUMP_DEST = '/u01/oracle/oradata/utrc'
```

12

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アーカイブ属性の構文、値および妥当性に関する情報について説明します。これらの属性には、次のものがあります。

AFFIRM および NOAFFIRM
ALTERNATE および NOALTERNATE
ARCH および LGWR
DELAY および NODELAY
DEPENDENCY および NODEPENDENCY
LOCATION および SERVICE
MANDATORY および OPTIONAL
MAX_FAILURE および NOMAX_FAILURE
NET_TIMEOUT および NONET_TIMEOUT
QUOTA_SIZE および NOQUOTA_SIZE
QUOTA_USED および NOQUOTA_USED
REGISTER および NOREGISTER
REGISTER=location_format
REOPEN および NOREOPEN
SYNC および ASYNC
TEMPLATE および NOTEMPLATE

さらに、次の項目についても説明します。

- LOG_ARCHIVE_DEST_n パラメータの属性
- アーカイブ先の属性の互換性

注意： 各宛先は、ローカル・ディスクのディレクトリまたはリモートにアクセスするデータベースのいずれかを示す必要があります。ログ転送サービスに関する追加情報およびこれらの初期化パラメータの使用方法は、第 5 章を参照してください。これらの初期化パラメータを使用してカスケードされた REDO ログを設定する方法の詳細は、付録 D を参照してください。

LOG_ARCHIVE_DEST_*n* パラメータの属性

LOG_ARCHIVE_DEST_*n*（*n* は 1 から 10 の整数）パラメータは、少なくとも 2 つ指定する必要があります。1 つは必須のローカル宛先用、もう 1 つはローカルまたはリモートの宛先用です。

次の項では、LOG_ARCHIVE_DEST_*n* 初期化パラメータの属性について説明します。追加情報については、第 5 章を参照してください。

すべての LOG_ARCHIVE_DEST_*n* パラメータには、少なくとも LOCATION または SERVICE のいずれかの属性が含まれている必要があります。また、定義した各宛先に対して LOG_ARCHIVE_DEST_STATE_*n* パラメータが必要です。

LOG_ARCHIVE_DEST_STATE_*n*（*n* は 1 から 10 の整数）初期化パラメータは、LOG_ARCHIVE_DEST_*n* 初期化パラメータによって示されている、対応する宛先の状態を指定します。たとえば、LOG_ARCHIVE_DEST_STATE_3 パラメータは、LOG_ARCHIVE_DEST_3 宛先の状態を指定します。

SQL 文を使用した宛先属性の変更

表 12-1 に、LOG_ARCHIVE_DEST_*n* 初期化パラメータに設定できる属性をリストし、その属性が ALTER SYSTEM 文または ALTER SESSION 文を使用して変更できるかどうかを示します。

表 12-1 SQL を使用した宛先属性の変更

属性	ALTER SYSTEM	ALTER SESSION
[NO]AFFIRM	可	可
[NO]ALTERNATE=*destination*	可	可
ARCH	可	可
ASYNC[=*blocks*]	可	不可
[NO]DELAY	可	可
[NO]DEPENDENCY=*destination*	可	不可
LGWR	可	不可
LOCATION=*local_disk_directory*	可	可
MANDATORY	可	可
[NO]MAX_FAILURE=*count*	可	不可
OPTIONAL	可	可
[NO]NET_TIMEOUT[=*seconds*]	可	不可

表 12-1 SQL を使用した宛先属性の変更（続き）

属性	ALTER SYSTEM	ALTER SESSION
[NO]QUOTA_SIZE=*blocks*	可	不可
[NO]QUOTA_USED=*blocks*	可	不可
[NO]REGISTER	可	可
REGISTER=location_format	可	可
[NO]REOPEN[=*seconds*]	可	可
SERVICE=*net_service_name*	可	可
SYNC[=PARALLEL|NOPARALLEL]	可	可
[NO]TEMPLATE=*filename_template*	可	可

LOG_ARCHIVE_DEST_*n* パラメータ設定の増分変更

ログ転送サービスの宛先 LOG_ARCHIVE_DEST_*n* 初期化パラメータは、その各パラメータに**属性**と呼ばれる複数の値が含まれているという点が特徴的です。LOCATION 属性と SERVICE 属性以外のすべての属性はオプションで、デフォルト値を持っています。

注意： 従来のテキストの初期化パラメータ・ファイルを使用している場合は、実行時に LOG_ARCHIVE_DEST_*n* パラメータを追加指定できます。これによって、パラメータ値全体を再指定せずに、1 つ以上の特定の属性を置換できます。5-27 ページの「データベース初期化パラメータ」の例で、従来のテキストの初期化パラメータ・ファイルの増分変更について確認してください。

等号（=）や空白などの埋込み文字を使用するネットワーク・サービス名を指定するには、サービス名を二重引用符（"）で囲みます。

例 12-1 に、LOG_ARCHIVE_DEST_1 パラメータの初期指定を置換する方法を示します。

例 12-1 宛先指定の置換

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll'
LOG_ARCHIVE_DEST_2='SERVICE=stdby REOPEN=60'
LOG_ARCHIVE_DEST_1='LOCATION=/disk3/oracle/oradata/payroll MANDATORY'
```

従来のテキストの初期化パラメータ・ファイルを使用している場合は、ALTER SYSTEM SET 文を使用して LOG_ARCHIVE_DEST_*n* 初期化パラメータを実行時に増分変更できます。これは、パラメータ値全体を再指定せずに 1 つ以上の特定の属性を変更できることを意味します。増分変更に指定できる属性は、LOCATION または SERVICE 以外の属性です。

注意： サーバー・パラメータ・ファイル（SPFILE）のパラメータに対しては増分変更できません。変更する場合は、SPFILE を従来のテキストの初期化パラメータ・ファイル（PFILE）に変換して、初期化パラメータを編集した後、PFILE を SPFILE に変換する必要があります。

例 12-2 に、複数の属性をそれぞれの行に設定する方法を示します。SERVICE または LOCATION 属性は第 1 行目に指定します。

例 12-2 複数の属性を増分して指定

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='OPTIONAL';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='REOPEN=5';
```

例 12-3 に、複数の宛先の属性を指定する方法を示します。LOG_ARCHIVE_DEST_*n* 初期化パラメータなどの増分パラメータは、相互に連続する必要はありません。

例 12-3 複数の宛先の複数の属性を指定

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=stdby REOPEN=60';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1= OPTIONAL';
```

LOCATION 属性または SERVICE 属性を指定すると、宛先の初期化パラメータがデフォルト値にリセットされます。例 12-4 に、増分変更とはみなされない LOG_ARCHIVE_DEST_1 に対する入力を示します。

例 12-4 置換された宛先指定

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll
REOPEN=60';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll';
```

パラメータ属性に NULL 値を含む文字列は、前に入力された宛先指定を消去します。例 12-5 に LOG_ARCHIVE_DEST_1 の定義を消去する方法を示します。

例 12-5 宛先指定の消去

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata/payroll'
LOG_ARCHIVE_DEST_2='SERVICE=stdby REOPEN=60'
LOG_ARCHIVE_DEST_1=''
```

宛先の初期化パラメータに対する現行設定の表示

SQL を使用して V$ARCHIVE_DEST などの固定ビューを問い合せて、LOG_ARCHIVE_DEST_*n* 初期化パラメータの現行設定を確認できます。たとえば、プライマリ・データベースで現行の宛先設定を表示するには、次の文を入力します。

```
SQL> SELECT DESTINATION FROM V$ARCHIVE_DEST;
```

注意： LOG_ARCHIVE_DEST_*n* 初期化パラメータの値の判断には、V$PARAMETER ビューを使用しないでください。V$PARAMETER ビューには、各パラメータに最後に指定された値のみが表示されるため、増分変更の場合は実際の LOG_ARCHIVE_DEST_*n* パラメータ値が表示されません。

AFFIRM および NOAFFIRM

カテゴリ	AFFIRM	NOAFFIRM
属性のデータ型	キーワード	キーワード
最小値	該当なし	該当なし
最大値	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	該当なし	該当なし
属性との競合	NOAFFIRM	AFFIRM
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	AFFIRM	AFFIRM
関連する V$ARCHIVE_DEST 列	ASYNC_BLOCKS	ASYNC_BLOCKS

目的

AFFIRM 属性および NOAFFIRM 属性を使用すると、アーカイブ REDO ログの内容をディスクに正しく書き込むことができます。この属性は、ローカル宛先およびリモート宛先の両方に適用されます。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに 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 NOAFFIRM です。

属性

AFFIRM

AFFIRM 属性は、リモート・スタンバイ・データベースの場合にも、すべてのアーカイブ REDO ログの I/O 操作が同期して実行されることを示します。この属性は、プライマリ・データベースのパフォーマンスに影響を与えることがあります。LGWR 属性と AFFIRM 属性を使用して、ログ・ライター・プロセスにローカルのアーカイブ REDO ログを同期させてディスクに書き込むように指定すると、ディスク I/O 操作が完了するまでは、制御がユーザーに戻されず、処理は続行されません。ARCH 属性と AFFIRM 属性を使用して、ARC*n* プロセスにアーカイブ REDO ログを同期させてディスクに書き込むように指定すると、アーカイブ操作に長時間かかり、アーカイブが完了するまでオンライン REDO ログが再使用できないことがあります。

AFFIRM 属性を使用しても、ASYNC 属性を使用する場合はパフォーマンスに影響しません。

V$ARCHIVE_DEST 固定ビューの AFFIRM 列を問合せると、AFFIRM 属性が、対応付けられた宛先に使用されるかどうかを確認できます。

次の表は、これらの属性の様々な組合せと、プライマリ・データベースのパフォーマンスやデータの可用性に対するそれぞれの影響を示します。たとえば、AFFIRM 属性を SYNC 属性や ASYNC 属性と併用すると、最高レベルのデータ保護が提供されますが、プライマリ・データベースのパフォーマンスは低下します。

ネットワーク I/O 属性	アーカイブ REDO ログ・ディスク I/O 属性	可能なプライマリ・データベースのパフォーマンス	スタンバイ・データベースのデータの保護
SYNC	AFFIRM	最低	最高
SYNC	NOAFFIRM	低	高
ASYNC	AFFIRM	高	低
ASYNC	NOAFFIRM	最高	最低

データの可用性が最高であっても、プライマリ・データベースのパフォーマンスは最低の場合があります。

注意： プライマリ・データベースが MAXIMIZE PROTECTION モードまたは MAXIMIZE AVAILABILITY モードである場合、ログ・ライター・プロセスによる REDO ログのアーカイブ先は、自動的に AFFIRM モードになります。

関連項目： 12-45 ページ「SYNC および ASYNC」

NOAFFIRM

NOAFFIRM 属性は、すべてのアーカイブ REDO ログのディスク I/O 操作が非同期で実行されることを示します。この場合、ログ・ライター・プロセスはディスク I/O が完了するまで待機せずに続行されます。

例

次の例は、AFFIRM 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 LGWR SYNC AFFIRM'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

ALTERNATE および NOALTERNATE

カテゴリ	ALTERNATE=*destination*	NOALTERNATE
属性のデータ型	文字列値	キーワード
最小値	該当なし	該当なし
最大値	該当なし	該当なし
デフォルト値	なし[1]	該当なし
必須属性	該当なし	該当なし
属性との競合	NOALTERNATE	ALTERNATE
属性クラス	ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	ALTERNATE	ALTERNATE
関連する V$ARCHIVE_DEST 列	STATUS	STATUS

[1] NOALTERNATE 属性が指定されている場合、または代替アーカイブ先が指定されていない場合、障害が発生しても宛先は自動的に別の宛先に変更されません。

目的

LOG_ARCHIVE_DEST_*n* パラメータの ALTERNATE 属性および NOALTERNATE 属性は、当初のアーカイブ先で障害が発生した場合の代替アーカイブ先を定義したり、代替アーカイブ先にアーカイブしないようにします。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに ALTERNATE 属性および NOALTERNATE 属性が指定されていない場合、デフォルトは NOALTERNATE です。

属性

ALTERNATE=*destination*

LOG_ARCHIVE_DEST_*n* パラメータの ALTERNATE 属性を使用して、当初のアーカイブ先へのアーカイブ操作に失敗した場合の代替アーカイブ先を定義します。

1 つのアーカイブ先に指定できる代替アーカイブ先は、最大でも 1 つです。代替アーカイブ先は、プライマリ・サイトからスタンバイ・サイトへのオンライン REDO ログの転送に失敗した場合に使用されます。アーカイブ操作に失敗し、REOPEN 属性の値に 0（ゼロ）が指定されている場合や NOREOPEN 属性が指定されている場合、Oracle データベース・サーバーは、次のアーカイブ操作で、オンライン REDO ログを代替アーカイブ先にアーカイブしようとします。

代替アーカイブ先には、ローカルまたはリモートのアーカイブ先を参照できます。代替アーカイブ先は、自己参照できません。

宛先は、ALTERNATE 状態にすることも可能です。この状態は、LOG_ARCHIVE_DEST_STATE_*n* 初期化パラメータを使用して指定します。ALTERNATE 状態の宛先の処理は、代替アーカイブ先属性が有効な場合、別の宛先の失敗によって、自動的にこの宛先が有効になるまで遅延します。LOG_ARCHIVE_DEST_STATE_*n* パラメータの詳細は、5-12 ページの「宛先パラメータと属性」を参照してください。

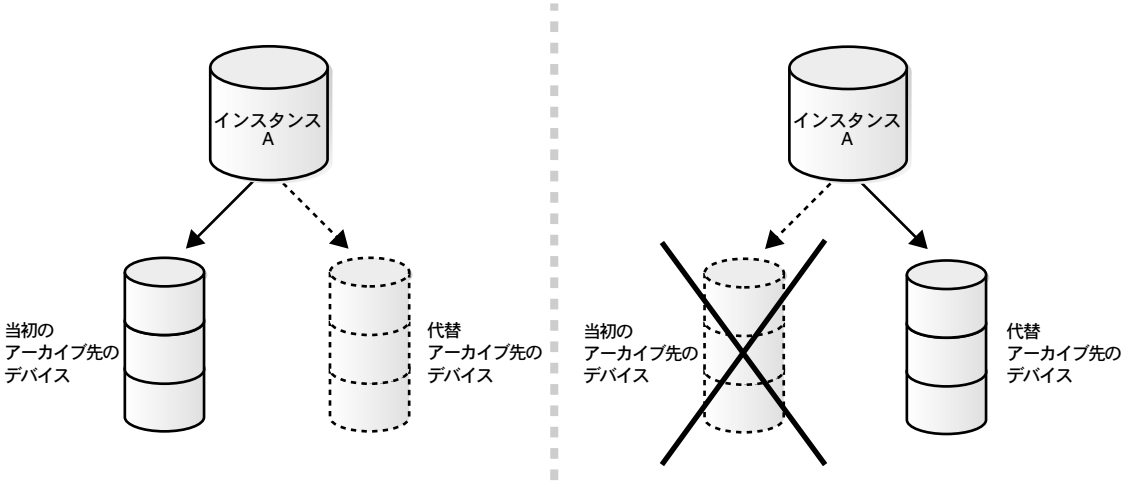
ALTERNATE 属性は、セッション・レベルでは修正できません。

次の例は、宛先 LOG_ARCHIVE_DEST_1 に失敗すると、アーカイブ・プロセスが自動的に宛先を LOG_ARCHIVE_DEST_2 に切り替える指定を示しています。

```
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
```

図 12-1 では、オンライン REDO ログがローカルのディスク・デバイスにアーカイブされています。当初のアーカイブ先のデバイスがいっぱいであったり、利用できない場合、アーカイブ操作は自動的に代替アーカイブ先のデバイスにリダイレクトされます。

図 12-1 代替アーカイブ先のデバイスへのアーカイブ操作



REOPEN 属性は、ALTERNATE 属性よりも優先されます。代替アーカイブ先は、次のいずれかの場合にのみ使用されます。

- NOREOPEN 属性が指定されている。
- REOPEN 属性に、値 0（ゼロ）が指定されている。
- 0（ゼロ）ではない REOPEN 属性で 0（ゼロ）ではない MAX_FAILURE の件数を超過している。

ALTERNATE 属性は、MANDATORY 属性よりも優先されます。これは、現在の宛先が必須であっても、有効な代替アーカイブ先に宛先をフェイルオーバーすることを意味します。

次の表は、アーカイブ REDO ログ宛先属性の優先順位を示します。

優先順位[1]	属性
1	MAX_FAILURE
2	REOPEN
3	ALTERNATE
4	MANDATORY

[1] 1 は最高優先順位、4 は最低優先順位を意味します。

スタンバイ・データベースを代替アーカイブ先のターゲットとして使用する場合は、注意が必要です。スタンバイの代替アーカイブ先は、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・ルートを指定するためにのみ使用するのが理想的です。

代替アーカイブ先を参照する有効な宛先がない場合、代替アーカイブ先を自動的に有効にする方法がないため、代替アーカイブ先は無効になることになります。

代替アーカイブ先は、実行時に手動で有効にできます。逆に言えば、代替アーカイブ先は、実行時に手動で無効にできます。実行時に SQL を使用した初期化パラメータ設定の変更の詳細は、5-28 ページの「SQL 文を使用して実行時にログ転送パラメータを設定」を参照してください。

代替アーカイブ REDO ログの宛先の一般プールはありません。データベース管理者が、すべての有効な宛先に対して、参照する宛先を厳密に反映する代替アーカイブ先を選択することが理想的です（ただし、代替アーカイブ先の指定は必須ではありません）。

有効な宛先は、それぞれ固有の代替アーカイブ先を持つことができます。逆に言えば、複数の有効な宛先が、同じ代替アーカイブ先を共有できます。これは、宛先のオーバーラップ・セットと呼ばれます。代替アーカイブ先を有効にすると、その宛先が属するセットが決まります。

有効な宛先数を増加させると、利用可能な代替 REDO ログのアーカイブ先数が減少します。

注意： 代替アーカイブ先は、次回に実行するアーカイブ操作で有効になります。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、すでに処理済みのブロックなどを、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。

代替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が、これによるエラーは発生しない。

SQL の ALTER SESSION 文を使用して定義した宛先は、システム・レベルで定義された代替アーカイブ先をアクティブ化しません。逆に言えば、システム定義の宛先は、セッション・レベルで定義された代替アーカイブ先をアクティブ化しません。

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。

NOALTERNATE

LOG_ARCHIVE_DEST_*n* パラメータの NOALTERNATE 属性を使用すると、当初の宛先で失敗した場合に、当初の宛先から代替アーカイブ先に自動的に変更されることを防止できます。

例

例 12-6 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 が、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 12-6 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 NOREOPEN ALTERNATE=LOG_ARCHIVE_DEST_2'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

例 12-7 のサンプル初期化パラメータ・ファイルは、同じスタンバイ・データベースに対して、代替の Oracle Net サービス名を定義する方法を示しています。

例 12-7 スタンバイ・データベースに対する代替の Oracle Net サービス名の定義

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2='SERVICE=stby1_path1 NOREOPEN OPTIONAL ALTERNATE=LOG_ARCHIVE_DEST_3'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=stby1_path2 NOREOPEN OPTIONAL'
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

ARCH および LGWR

カテゴリ	ARCH	LGWR
属性のデータ型	キーワード	キーワード
最小値	該当なし	該当なし
最大値	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	該当なし	該当なし
属性との競合	LGWR ASYNC、ASYNC、NET_TIMEOUT	ARCH
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SYSTEM のみ
対応する V$ARCHIVE_DEST 列	ARCHIVER	ARCHIVER
関連する V$ARCHIVE_DEST 列	PROCESS、SCHEDULE	PROCESS、SCHEDULE

目的

オプションの ARCH 属性および LGWR 属性を使用して、アーカイバまたはログ・ライター・プロセスのいずれかが、オンライン REDO ログをローカルおよびリモートのアーカイブ先に転送するように指定します。

アーカイブ先に ARCH 属性と LGWR 属性を使用し、アーカイブ操作を ARC*n* プロセスから LGWR プロセスに変更すると、LGWR プロセスは次のログ・スイッチ操作までアーカイブを開始しません。逆に、LGWR プロセスから ARC*n* プロセスにアーカイブ・プロセスを変更すると、LGWR プロセスは次のログ・スイッチ操作までアーカイブを継続します。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに ARCH 属性または LGWR 属性が指定されていない場合、デフォルトは ARCH です。

属性

ARCH

ARCH 属性は、アーカイブ操作中に REDO ログを宛先に転送することを示します。バックグラウンドのアーカイバ・プロセス（ARC*n*）またはフォアグラウンドのアーカイブ操作は、REDO ログ転送サービスとして機能します。

LGWR

LGWR 属性は、オンライン REDO ログが移入されると同時に、REDO ログが宛先に転送されることを示します。バックグラウンドのログ・ライター・プロセス（LGWR）は、REDO ログ転送サービスとして機能します。REDO ログをリモートの宛先に転送すると、LGWR プロセスは、宛先インスタンスにネットワーク接続を確立します。複数の REDO ログが同時に転送されるため、アーカイブ操作時に対応する宛先に再転送されることはありません。
LGWR の宛先に失敗すると、エラーが修正されるまで、宛先は自動的にアーカイバ・プロセス（ARCn）を使用します。

例

次の例は、LGWR 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 LGWR'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

DELAY および NODELAY

カテゴリ	DELAY[=*minutes*]	NODELAY
属性のデータ型	数値	キーワード
最小値	0 分	該当なし
最大値	無制限	該当なし
デフォルト値	30 分	該当なし
必須属性	SERVICE	該当なし
属性との競合	LOCATION、NODELAY	DELAY
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	DELAY_MINS	DELAY_MINS
関連する V$ARCHIVE_DEST 列	DESTINATION	DESTINATION

目的

スタンバイ・データベースが管理リカバリ・モードになっている場合は、プライマリ・データベースから受信した REDO ログが自動的に適用されます。ただし、破損または誤ったデータのプライマリ・サイトからスタンバイ・サイトへの転送を防止するために、プライマリ・サイトでの REDO ログのアーカイブと、スタンバイ・サイトでのアーカイブ REDO ログの適用との間にタイム・ラグを作成できます。

デフォルト

管理リカバリ・モードで、LOG_ARCHIVE_DEST_*n* パラメータに DELAY 属性および NODELAY 属性が指定されていない場合、デフォルトは NODELAY です。

時間間隔なしで DELAY 属性が指定されている場合、デフォルトの時間間隔は 30 分になります。

属性

DELAY[=*minutes*]
LOG_ARCHIVE_DEST_*n* 初期化パラメータの DELAY 属性を使用して、スタンバイ・サイトで REDO ログを適用する際のタイム・ラグを指定します。DELAY 属性は、REDO ログのスタンバイ・サイトへの転送に影響しません。

注意： DELAY 属性を変更すると、次回に実行するアーカイブ操作から有効になります。進行中のアーカイブ操作には、影響しません。

DELAY 属性は、スタンバイ・サイトのアーカイブ REDO ログが、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。時間間隔は分で示し、REDO ログがスタンバイ・サイトに正常に転送され到達したときから計測します。

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。たとえば、プライマリ・データベース A がスタンバイ・データベース B、C、D をサポートしている場合を考えます。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、論理的、または物理的な破損に対して保護するように設定され、2 時間の遅延があります。スタンバイ・データベース D には 4 時間の遅延があり、破損が拡大された場合の保護を提供します。

スタンバイ・サイトでの指定した遅延間隔は上書きできます。時間間隔が経過する前に、アーカイブ REDO ログをスタンバイ・データベースに即時に適用するには、RECOVER MANAGED STANDBY DATABASE 句の NODELAY キーワードを使用します。たとえば、次のようになります。

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

NODELAY
NODELAY 属性を指定し、スタンバイ・データベースが管理リカバリ・モードになっている場合は、プライマリ・データベースから受信した REDO ログが自動的に適用されます。

関連項目： 第 13 章「SQL 文」

例

次の例は、DELAY 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 DELAY=240'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

DEPENDENCY および NODEPENDENCY

カテゴリ	DEPENDENCY=*destination*	NODEPENDENCY
属性のデータ型	文字列値	キーワード
最小値	該当なし	該当なし
最大値	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、REGISTER	該当なし
属性との競合	NODEPENDENCY、LOCATION、NOREGISTER、QUOTA_SIZE、QUOTA_USED	DEPENDENCY
属性クラス	ALTER SYSTEM のみ	ALTER SYSTEM のみ
対応する V$ARCHIVE_DEST 列	DEPENDENCY	DEPENDENCY
関連する V$ARCHIVE_DEST 列	該当なし	該当なし

目的

リモート・データベースへの REDO ログのアーカイブは、別の宛先に対するアーカイブ操作の成功または失敗に依存して実行されるように定義できます。依存する宛先は、子宛先と呼ばれます。子が依存する宛先は、親宛先と呼ばれます。LOG_ARCHIVE_DEST_*n* パラメータに DEPENDENCY 属性を指定して、ローカルの宛先、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・データベースを定義します。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに DEPENDENCY 属性または NODEPENDENCY 属性が指定されていない場合、デフォルトは NODEPENDENCY です。

属性

DEPENDENCY=*destination*
宛先依存性を指定すると、次に示すような構成の場合に便利です。

- 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ノード上にある。このため、アーカイブ REDO ログが暗黙的にスタンバイ・データベースの影響を受けやすい状態にある。
- クラスタ化されたファイル・システムが、リモートのスタンバイ・データベースにプライマリ・データベースのアーカイブ REDO ログへのアクセスを提供する。
- オペレーティング・システム固有のネットワーク・ファイル・システムが、リモートのスタンバイ・データベースに、プライマリ・データベースのアーカイブ REDO ログへのアクセスを提供する。
- ミラー・ディスク・テクノロジによって、地理的に距離の離れた地点間の透過的なネットワーキング・サポートを提供する。
- 複数のスタンバイ・データベースが同じリモート・サイトに存在し、共通のアーカイブ REDO ログへのアクセスを共有する。

このような状況では、物理的なアーカイブ操作が、依存する宛先について発生することはありませんが、スタンバイ・データベースはアーカイブ REDO ログの位置を知っておく必要があります。このため、スタンバイ・データベースは、アーカイブ REDO ログが管理リカバリで利用可能になると、そのアーカイブ REDO ログにアクセスできます。データベース管理者は、親宛先の成功、または失敗に依存するように宛先を指定する必要があります。

2 つのノードがクラスタ化されている場合を考えます。この構成のプライマリ・ノードは、宛先へのアクセスを、ミラー化されたディスク・デバイスからスタンバイ・データベースと共有します。この構成は、ローカルにスタンバイ・データベースを維持するため、オフロードの非定型問合せとレポート機能に便利です。

プライマリ・データベースは、REDO ログをローカルにアーカイブします。正常に完了すると、アーカイブ REDO ログは即時にスタンバイ・データベースで利用できるようになり、管理リカバリを実行します。このとき、スタンバイ宛先には、物理的なリモートのアーカイブ操作を必要としません。この場合、2 つの宛先、つまりローカルのアーカイブ操作用の宛先とスタンバイ・サイトでのアーカイブ操作用の宛先が使用されます。スタンバイ宛先は、プライマリ宛先が成功しない場合は有効ではありません。このためスタンバイ宛先は、ローカルの宛先の成功、または失敗に依存します。

DEPENDENCY 属性には、次の制限事項があります。

- 依存性を持つことができるのはスタンバイ宛先のみである。
- 親宛先には、ローカルの宛先、またはスタンバイ宛先がなり得る。
- DEPENDENCY 属性は、セッション・レベルでは変更できない。
- REGISTER 属性が必要。

- SERVICE 属性が必要。
- 代替アーカイブ先は、自己参照できない。

1 つ以上の宛先が同じ親宛先に依存する場合、依存する宛先のすべての属性が、その宛先に適用されます。実際にはアーカイブ操作が 1 回のみ発生した場合でも、アーカイブ操作が各宛先に対して実行されたように見えます。

たとえば、2 つのスタンバイ・データベースが親宛先のアーカイブ REDO ログに依存する場合を考えます。各宛先について異なる DELAY 属性を指定できるため、プライマリ・データベースと各スタンバイ・データベース間で、タイム・ラグをずらして維持できます。

同じように、依存する宛先には、代替アーカイブ先を指定できますが、同じ親宛先に依存するようにもしないようにも指定できます。

注意： 依存する宛先は、スタンバイの非データ消失環境には含まれません。

NODEPENDENCY
アーカイブの実行が、別の宛先に対するアーカイブ操作の成功または失敗に依存しないことを指定します。

例

DEPENDENCY 属性を使用する理由の 1 つは、スタンバイ・データベースがプライマリ・データベースと同一のサイト上にある場合です。この構成を使用すると、REDO ログに必要なアーカイブは 1 度のみです。これは、スタンバイ・データベースがローカル・システム上に存在するので、同じ REDO ログにアクセスできるためです。次の例は、この構成を使用した場合の LOG_ARCHIVE_DEST_*n* パラメータを示しています。

```
# Set up the mandatory local destination:
#
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/dbs/ MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
#
# Set up the dependent standby database that resides on the local system:
#
LOG_ARCHIVE_DEST_2='SERVICE=dest2 DEPENDENCY=LOG_ARCHIVE_DEST_1 OPTIONAL'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

DEPENDENCY 属性を使用するもう 1 つの理由は、同一システム上に 2 つのスタンバイ・データベースがある場合です。親および子のスタンバイ・データベースには、フィジカル・スタンバイ・データベースとロジカル・スタンバイ・データベースを混在させることができます。次の例は、このような場合のパラメータ記述を示しています。

```
# Set up the mandatory local destination:
#
LOG_ARCHIVE_DEST_1='LOCATION=/oracle/dbs/ MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
#
# Set up the remote standby database that will receive the logs:
#
LOG_ARCHIVE_DEST_2='SERVICE=dest2 OPTIONAL'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
#
# Set up the remote standby database that resides on the same system as, and is
# dependent on, the first standby database:
#
LOG_ARCHIVE_DEST_3='SERVICE=dest3 DEPENDENCY=LOG_ARCHIVE_DEST_2 OPTIONAL'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

LOCATION および SERVICE

カテゴリ	LOCATION=*local_disk_directory*	SERVICE=*net_service_name*
属性のデータ型	文字列値	文字列値
最小値	該当なし	該当なし
最大値	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	該当なし	該当なし
属性との競合	SERVICE、DELAY、DEPENDENCY、REGISTER=*location_format*、NOREGISTER、ASYNC、TEMPLATE、NET_TIMEOUT	LOCATION、QUOTA_USED、QUOTA_SIZE
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	DESTINATION	DESTINATION
関連する V$ARCHIVE_DEST 列	TARGET	TARGET

目的

各宛先は、ローカル・ディスクのディレクトリまたはリモートにアクセスするデータベースのいずれかを示す必要があります。

LOCATION 属性または SERVICE 属性のいずれかを使用して、ログ転送サービスのアーカイブ REDO ログの宛先を指定します。各 Data Guard 構成に対して、REDO ログがアーカイブされるローカル・ディスクのディレクトリを最低でも 1 つ指定（LOCATION=*local_disk_directory*）する必要があります。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。有効な Oracle Net サービス名を指定（SERVICE=*net_service_name*）することによって、リモートの宛先を識別します。

注意： 複数の属性を指定している場合は、初期化パラメータ・ファイルの 1 行目に LOCATION 属性または SERVICE 属性のいずれかを指定します。

リモートのアーカイブ REDO ログを使用して、トランザクション一貫性のあるプライマリ・データベースのコピーを維持します。トランザクション一貫性のあるスタンバイ・データベースをメンテナンスする場合、ローカルのアーカイブ REDO ログは使用しません。ただし、ログ転送サービスを構成する場合は、少なくとも 1 つのローカル宛先を指定する必要があります。これによって、手動によるプライマリ・データベースのリカバリが必要な場合は、ローカルのアーカイブ REDO ログに確実にアクセスできます。

現行の設定を確認するには、V$ARCHIVE_DEST 固定ビューを問合せます。

- V$ARCHIVE_DEST 固定ビューの TARGET 列は、宛先がプライマリ・データベースにとってローカルかリモートかを示します。
- V$ARCHIVE_DEST 固定ビューの 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・ログ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。

デフォルト

これらの属性の 1 つを指定する必要があります。デフォルトはありません。

注意： 増分パラメータを変更する場合、デフォルトの属性値はないとみなされます。たとえば、LOG_ARCHIVE_DEST_1='SERVICE=stby1 LGWR' は、SYNC=PARALLEL オプションとみなされます。ARCH のデフォルトが SYNC=NOPARALLEL の場合でも、SYNC=PARALLEL は ARCH と競合するため、LOG_ARCHIVE_DEST_1='ARCH' は失敗となります。

属性

LOCATION=*local_disk_directory*

LOCATION 属性を使用する場合、プライマリ・データベースを置くシステム上のディスク・ディレクトリへの有効なパス名を指定します。LOCATION 属性を指定する各宛先は、一意のディレクトリ・パス名を示す必要があります。これは、アーカイブ REDO ログのローカル宛先です。

ローカル宛先は、アーカイブ REDO ログが、プライマリ・データベースで利用できるファイル・システム内に常駐していることを示します。ローカルのアーカイブ REDO ログは、物理的にはプライマリ・データベースのネームスペース内に残っています。宛先パラメータ値は、アーカイブ REDO ログがコピーされる、ローカルのファイル・システム・ディレクトリのパスを指定します。

SERVICE=*network_service_name*

SERVICE 属性を指定する場合は、有効な Oracle Net サービス名を指定します。

リモート宛先に REDO ログをアーカイブするには、着信するアーカイブ REDO ログを受け取るために、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。

宛先パラメータ値は、アーカイブ REDO ログのコピー先となるリモートの Oracle インスタンスを示す Oracle Net サービス名を指定します。

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 サービス名の設定の詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

例

次の例は、LOCATION 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_2='LOCATION=/arc_dest'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

次の例は、SERVICE 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'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

MANDATORY および OPTIONAL

カテゴリ	MANDATORY	OPTIONAL
属性のデータ型	キーワード	キーワード
最小値	該当なし	該当なし
最大値	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	該当なし	該当なし
属性との競合	OPTIONAL	MANDATORY
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	BINDING	BINDING
関連する V$ARCHIVE_DEST 列	該当なし	該当なし

目的

LOG_ARCHIVE_DEST_*n* パラメータに OPTIONAL または MANDATORY 属性を使用して、オンライン REDO ログを再利用するためのポリシーを指定できます。OPTIONAL の宛先へのアーカイブ操作には障害が発生する場合でも、オンライン REDO ログが上書きされます。必須の宛先へのアーカイブ操作に障害が発生した場合、オンライン REDO ログは上書きされません。

LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* パラメータ（*n* は 1 から 10 の整数）は、ログ・ライター・プロセスがオンライン REDO ログ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、すべての必須宛先とスタンバイではないオプションの宛先によって満たされます。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 logs.
LOG_ARCHIVE_MIN_SUCCEED_DEST = 2
```

パラメータを設定する際は、次のことに注意します。

- この属性は、宛先の保護モードに影響を与えません。
- 1 つ以上のローカル宛先が必要であり、それらに OPTIONAL または MANDATORY を宣言できます。

  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（必須のスタンバイ宛先を含む）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 値を、宛先の数および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。
- 必須の宛先を遅延させたり、オンライン・ログの上書きを、スタンバイ・サイトへ REDO ログを転送せずに実行する場合、REDO ログをスタンバイ・サイトに手動で転送する必要があります。

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。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに MANDATORY 属性および OPTIONAL 属性が指定されていない場合、デフォルトは OPTIONAL です。

すべての宛先がオプションに指定されている場合にも、最低 1 つの宛先は成功する必要があります。

属性

MANDATORY
REDO ログが再利用可能になる前に、宛先へのアーカイブに成功する必要があることを指定します。

OPTIONAL
REDO ログが再利用可能になる前に、宛先へのアーカイブに成功する必要がないことを指定します。成功したアーカイブが LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータに設定した必要な成功件数に一致すると、REDO ログには再利用のためのマークが設定されます。

例

次の例は、MANDATORY 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_1='LOCATION=/arc/dest MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_3='SERVICE=stby1 MANDATORY'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

MAX_FAILURE および NOMAX_FAILURE

カテゴリ	MAX_FAILURE=*count*	NOMAX_FAILURE
属性のデータ型	数値	キーワード
最小値	0	該当なし
最大値	なし	該当なし
デフォルト値	なし	該当なし
必須属性	REOPEN	該当なし
属性との競合	NOMAX_FAILURE	MAX_FAILURE
SQL 文による動的変更	ALTER SYSTEM	ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	MAX_FAILURE	該当なし
関連する V$ARCHIVE_DEST 列	FAILURE_COUNT、REOPEN_SECS	該当なし

目的

MAX_FAILURE 属性および NOMAX_FAILURE 属性によって、ログ転送サービスが障害の発生した宛先との通信を再確立してアーカイブ操作を再開するための試行回数を制御できます。

デフォルト

MAX_FAILURE 属性および NOMAX_FAILURE 属性をいずれも指定しない場合、デフォルトは NOMAX_FAILURE です。このデフォルトは、失敗した宛先へのアーカイブ REDO ログの転送を無制限に連続して試行します。

属性

MAX_FAILURE=*count*

MAX_FAILURE 属性は、ログ転送サービスが失敗した宛先へのアーカイブ操作を継続的に試行する最大回数を指定します。この属性を使用すると、失敗後に回数制限付きでアーカイブ操作を再試行するように、アーカイブ先に対する障害の解決方法を指定できます。MAX_FAILURE 属性を指定する場合は、REOPEN 属性も指定し、特定の宛先に対するアーカイブ操作の頻度を指定する必要があります。

アーカイブの試行回数を制限する設定 MAX_FAILURE 属性と REOPEN 属性の両方を 0（ゼロ）ではない値に設定すると、ログ転送サービスは、アーカイブ操作の試行回数を MAX_FAILURE 属性で指定されている回数に制限します。各宛先には、連続的に発生したアーカイブ障害の回数を追跡する内部障害カウンタがあります。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。

なんらかの理由でアーカイブ操作に失敗すると、次の状態になるまで障害件数が増加します。

- 障害が発生しなくなり、アーカイブ操作が再開されるまで
- 障害件数が MAX_FAILURE 属性に設定されている値以上になるまで

> **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 属性値またはその他の属性を修正する必要があります。

- ALTER SYSTEM SET 文を発行して、MAX_FAILURE 属性（または宛先のその他の属性）を動的に変更します。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。

> **注意：** QUOTA_USED 属性の変更など、実行時に宛先に対して行われる修正は、障害件数に影響しません。

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）の値に設定されます。これによって、ログ転送サービスは、次のアーカイブ操作ではアーカイブ REDO ログ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。

アーカイブ操作を無制限に試行する設定 MAX_FAILURE 属性を指定せずに（または MAX_FAILURE=0 または NOMAX_FAILURE 属性を指定し）、REOPEN 属性に 0（ゼロ）ではない値を指定すると、ログ転送サービスは、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。宛先に MANDATORY 属性があると、繰返し発生する障害の場合、オンライン REDO ログは再利用されません。

NOMAX_FAILURE

NOMAX_FAILURE 属性を指定すると、失敗した宛先に対するアーカイブ操作を無制限に試行できます。

NOMAX_FAILURE 属性は、MAX_FAILURE=0 に指定することと等価です。

例

次の例は、ログ転送サービスが、宛先 arc_dest に対して、アーカイブ操作を最高 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 NOREOPEN 属性が指定されている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 および NONET_TIMEOUT

カテゴリ	NET_TIMEOUT=*seconds*	NONET_TIMEOUT
属性のデータ型	数値	該当なし
最小値	15	該当なし
最大値	1200	該当なし
デフォルト値	該当なし	該当なし
必須属性	LGWR と SYNC=PARALLEL または LGWR と ASYNC > 0	該当なし
属性との競合	ARCH、 LOCATION、 NONET_TIMEOUT、 LGWR と SYNC=NOPARALLEL、LGWR と ASYNC=0	NET_TIMEOUT
属性クラス	ALTER SYSTEM	ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	NET_TIMEOUT	NET_TIMEOUT
関連する V$ARCHIVE_DEST 列	該当なし	該当なし

目的

LOG_ARCHIVE_DEST_*n* パラメータの NET_TIMEOUT 属性は、ネットワーク接続を終了する前に、ログ・ライター・プロセスがネットワーク・サーバーからのステータスを待機する時間を秒単位で指定します。NONET_TIMEOUT 属性は、以前に NET_TIMEOUT 属性で指定したタイムアウト値を中止または無効にします。

NET_TIMEOUT 属性を指定しない場合または NONET_TIMEOUT 属性を指定した場合、プライマリ・データベースは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 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。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに NET_TIMEOUT 属性および NONET_TIMEOUT 属性が指定されていない場合、デフォルトは NONET_TIMEOUT です。

属性

NET_TIMEOUT=*seconds*

NET_TIMEOUT 属性が使用されるのは、ログ・ライター・プロセスがネットワーク・サーバー・プロセスを使用してログをアーカイブする場合、および ASYNC 属性または SYNC=PARALLEL 属性のいずれかが指定されている場合のみです。ログ・ライター・プロセスは、指定時間が経過するまで、ネットワーク I/O 操作からのステータスの受信を待機します。ログ・ライター・プロセスが指定時間内に応答を受信しない場合は、プライマリ・データベースのネットワーク接続が終了します。

Data Guard 構成では、プライマリとスタンバイ間の各ネットワーク接続に対して同様の設定が必要なタイマーがあります。

- Data Guard 構成のプライマリ・データベースで NET_TIMEOUT 属性を設定します。
- Data Guard 構成の各スタンバイ・データベースで Oracle Net EXPIRE_TIME パラメータおよび TCP/IP keepalive パラメータを設定します。

プライマリ・データベースのネットワーク接続が終了した可能性がある場合にも、スタンバイ・データベースのネットワーク接続は、対応する TCP/IP ネットワーク・タイマーが期限切れになるまでアクティブなままです。このため、ネットワークの双方でタイマーを同等に設定する必要があります。ネットワーク・タイマーが適切に設定されていない場合、スタンバイ・データベースはタイムアウトしておらず、中断したネットワーク接続は依然として有効に見えるため、スタンバイ・データベースと連結しているプライマリ・データベースでのログ・ライター・プロセスによって行われる、その後の試行はエラーとなります。

注意： 通常、スタンバイ・システムのネットワーク・タイマーのパラメータは、プライマリ・システムの NET_TIMEOUT 属性に指定されているタイムアウト時間までに期限切れとなるように設定してください。次に例を示します。

- スタンバイ・システムで、keepalive パラメータなどの TCP/IP ネットワーク・タイマーを、NET_TIMEOUT 属性の対応する値セットよりも低い値に設定します。
- スタンバイ・システムで、Oracle Net EXPIRE_TIME パラメータを、NET_TIMEOUT 属性に設定した対応する値よりも低く設定します。EXPIRE_TIME パラメータは分単位で示します。

プライマリ・システムとスタンバイ・システムのタイムアウト・パラメータ値は慎重に調整してください。不適切な場合、プライマリ・システムでネットワーク上の問題が発生してネットワークが切断されたときに、スタンバイ・データベースでデフォルトのネットワーク・タイムアウト値が過度に高く設定されていると、スタンバイ・データベースがネットワークの切断を認識しない可能性があります。

関連項目：『Oracle9*i* Net Services 管理者ガイド』

ログ・ライター・プロセスがネットワークの切断を検出すると、ネットワーク・タイムアウトによって終了した切断であっても、ログ・ライター・プロセスはスタンバイ・データベースへの再接続を自動的に試行します。これは、ネットワークの停止およびネットワーク終了エラーを解決するためです。ネットワークが物理的に破損している場合を除くほとんどの場合、ログ・ライター・プロセスはネットワークに自動的に再接続できます。

ログ・ライター・プロセスは、プライマリ・データベースに現在設定されているデータ保護モードに応じた一定期間内であれば、スタンバイ・データベースへの再接続を継続的に試行します。ログ・ライター・プロセスがスタンバイ・データベースへの再接続を試行する期間のガイドラインとして、次の見積りを使用してください。

- 最大保護モードでは、ログ・ライター・プロセスはおよそ 5 分間再接続を試行します。
- 最大可用性モードでは、ログ・ライター・プロセスは再接続を約 2 分間試行します。
- 最大パフォーマンス・モードでは、ログ・ライター・プロセスはおよそ 30 秒間再接続を試行します。

注意： 最大限の保護モードで実行している場合は、適切な値を慎重に指定してください。不適切なネットワークの障害検出によって、プライマリ・インスタンスが停止する可能性があります。

ログ・ライター・プロセスが再接続を試行する実際の時間は、次の要因によって異なります。

- NET_TIMEOUT 属性の値。この値によって、接続のタイムアウトが発生するまでの時間が判断されます。
- スタンバイ・データベースの EXPIRE_TIME パラメータ値またはキープ・アライブ間隔の値。この値によって、プライマリ・データベースで再接続が行われるまでの最小時間が判断されます。
- プライマリ・データベースの保護モード。これによって、再接続が行われる最大時間が判断されます。

たとえば、NET_TIMEOUT 属性値が 60 秒、EXPIRE_TIME が 1 分に設定されているプライマリ・データベースが最大可用性保護モードで動作している場合、接続には 1 分、スタンバイ・データベースへの接続を終了するには最大 3 分の時間が実際には必要と考えられます。

NONET_TIMEOUT

NONET_TIMEOUT 属性は、ログ・ライター・プロセスが、システムに設定されているデフォルトのネットワーク・タイムアウト時間が経過するまで待機することを意味します。デフォルトのネットワーク・タイムアウト時間は、システムによって異なります。一部のシステムでは、デフォルトの TCP/IP ネットワーク・タイムアウトが 10 分から 15 分の間に設定されています。

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 LGWR NET_TIMEOUT=40 SYNC=PARALLEL'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

QUOTA_SIZE および NOQUOTA_SIZE

カテゴリ	QUOTA_SIZE=*blocks*	NOQUOTA_SIZE
属性のデータ型	数値	キーワード
最小値	0 ブロック	該当なし
最大値	無制限ブロック	該当なし
デフォルト値	該当なし	該当なし
必須属性	LOCATION	該当なし
属性との競合	NOQUOTA_SIZE、DEPENDENCY、SERVICE	QUOTA_SIZE
属性クラス	ALTER SYSTEM のみ	ALTER SYSTEM のみ
対応する V$ARCHIVE_DEST 列	QUOTA_SIZE	QUOTA_SIZE
関連する V$ARCHIVE_DEST 列	QUOTA_USED	QUOTA_USED

目的

LOG_ARCHIVE_DEST_*n* パラメータの QUOTA_SIZE 属性および NOQUOTA_SIZE 属性は、ローカル宛先で使用できるディスク・デバイス上の物理記憶域にある、512 バイトのブロックの最大数を示します。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに QUOTA_SIZE 属性または NOQUOTA_SIZE 属性が指定されていない場合、デフォルトは NOQUOTA_SIZE です。

属性

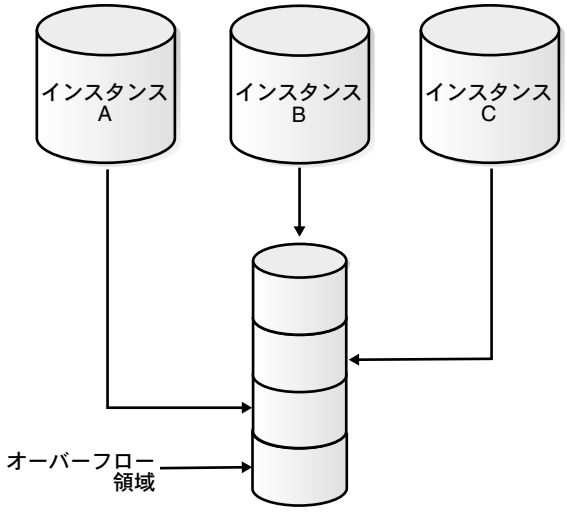
QUOTA_SIZE=*blocks*

QUOTA_SIZE 属性は、ローカル宛先で使用できるディスク・デバイス上の物理記憶域にある、512 バイトのブロックの最大数を示します。物理デバイスで異なるブロック・サイズが使用されている場合でも、この値は 512 バイトのブロック数で指定します。オプションの接尾辞の値、K、M、G はそれぞれ、千、百万、十億を表します（1K の値は 1,000 の 512 バイトのブロックを意味します）。

ローカルのアーカイブ先は、物理ディスクのすべて、または一部を占有できるように指定できます。たとえば、Real Application Clusters 環境では、Sun Clusters などで使用可能なクラスタ化ファイル・システムを介して、1 つの物理的なアーカイブ REDO ログ・ディスク・デバイスを 2 つ以上の個別のノードで共有できます。インスタンス間では、初期化パラメータの情報が共有されていないため、Real Application Clusters ノードは、アーカイブ REDO ログの物理ディスク・デバイスが他のインスタンスと共有されていることを認識していません。このため、宛先ディスク・ドライブがいっぱいになったときに、重大な問題が発生します。すなわち、すでにいっぱいになっているデバイスに対し、すべてのインスタンスがアーカイブを実行するまで、エラーは検出されません。これはデータベースの可用性に重大な影響を及ぼします。

たとえば、8GB ディスク・デバイス /dev/arc_dest について考えます。このデバイスはさらに、ノード固有のディレクトリ、node_a、node_b、node_c に副分割されます。データベース管理者は、これらの各インスタンスが最大 2GB を使用できるように指定できます。これによって、その他の目的に追加の 2GB が使用できるようになります。この例を図 12-2 に示します。

図 12-2 宛先へのディスク・クオータの指定



インスタンスは、割当て制限を超えるディスク・デバイスを使用することはありません。

クオータは、宛先のすべてのユーザーに共通のものです。フォアグラウンドのアーカイブ操作、アーカイバ・プロセス、ログ・ライター・プロセスなどで使用します。

オラクル社では、ALTERNATE 属性と QUOTA_SIZE 属性を一緒に使用することをお薦めします。ただし、これは必須ではありません。

関連項目： 12-8 ページ「ALTERNATE および NOALTERNATE」

NOQUOTA_SIZE

NOQUOTA_SIZE 属性、または値が 0（ゼロ）値の QUOTA_SIZE 属性を使用すると、この宛先によるディスク・デバイスの使用が無制限であることを示します。これはデフォルトの動作です。

例

次の例は、QUOTA_SIZE 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_4='QUOTA_SIZE=100K'
```

QUOTA_USED および NOQUOTA_USED

カテゴリ	QUOTA_USED=*blocks*	NOQUOTA_USED
属性のデータ型	数値	キーワード
最小値	0 ブロック	該当なし
最大値	無制限ブロック	該当なし
デフォルト値	該当なし	該当なし
必須属性	LOCATION	該当なし
属性との競合	NOQUOTA_USED、DEPENDENCY、SERVICE	QUOTA_USED
属性クラス	ALTER SYSTEM のみ	ALTER SYSTEM のみ
対応する V$ARCHIVE_DEST 列	QUOTA_USED	QUOTA_USED
関連する V$ARCHIVE_DEST 列	QUOTA_SIZE	QUOTA_SIZE

目的

LOG_ARCHIVE_DEST_*n* パラメータの QUOTA_USED 属性および NOQUOTA_USED 属性は、指定した宛先にアーカイブされたデータの 512 バイトのブロック数を示します。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに QUOTA_USED 属性または NOQUOTA_USED 属性が指定されていない場合、デフォルトは NOQUOTA_USED です。

QUOTA_USED 属性のリモートのアーカイブ先に対するデフォルト値は 0（ゼロ）です。

属性

QUOTA_USED=*blocks*

QUOTA_USED 属性は、指定のローカル宛先にアーカイブされたデータの 512 バイトのブロック数を示します。物理デバイスで異なるブロック・サイズが使用されている場合でも、この値は 512 バイトのブロック数で指定します。オプションの接尾辞の値、K、M、G はそれぞれ、千、百万、十億を表します（1K の値は 1,000 の 512 バイトのブロックを意味します）。

この属性は、セッション・レベルでは修正できません。

宛先に対して QUOTA_SIZE 属性の値を 0（ゼロ）よりも大きな値に指定し、データベース初期化パラメータ・ファイルの QUOTA_USED 属性を指定しない場合、QUOTA_USED 属性値はデータベースを最初にマウントしたときに自動的に決定されます。QUOTA_USED 属性値のデフォルトはローカルのアーカイブ先デバイス上にある実際のブロック数になります。計算した QUOTA_USED 属性値が QUOTA_SIZE 属性値を超過すると、QUOTA_SIZE 属性値は自動的に調整され、実際に使用した記憶域が反映されます。

このように自動的に計算された QUOTA_USED 値は、ローカルのアーカイブ先に対してのみ適用されます。

注意： QUOTA_USED 属性の実行時の値は、オンライン REDO ログのアーカイブ操作が開始されると自動的に変更されます。QUOTA_USED 属性値は、宛先のクオータ・サイズに対して前もって自動的に割り当てられます。この属性の値を変更する必要はありません。

実行時に QUOTA_SIZE 属性値が動的に修正され、QUOTA_USED 属性値が修正されない場合、QUOTA_USED 属性値は動的に再計算されません。

ローカル宛先の場合、QUOTA_USED 属性値はアーカイブ操作の開始時に増加されます。その結果の値が QUOTA_SIZE 属性値よりも大きい場合、宛先の状態が FULL に変化し、その宛先はアーカイブ操作の開始前に拒否されます。

QUOTA_SIZE と QUOTA_USED 属性は、一緒に使用すると、アーカイブ操作が始まる前にディスク領域の不足を検出できるため、非常に重要です。

QUOTA_SIZE 属性値が 100K、QUOTA_USED 属性値も 100K の場合を考えます。宛先の状態は、この時点では VALID です。しかし、1 ブロックをアーカイブすると、その結果 QUOTA_USED 属性値は 101K に変更されます。これは、QUOTA_SIZE 属性値を超過しています。このため、宛先の状態は、FULL に変更され、この宛先はアーカイブ操作が始まる前に拒否されます。

NOQUOTA_USED
指定の宛先でアーカイブできるデータのブロック数を制限しないことを指定します。

例

この値は Data Guard によって自動的に設定されます。QUOTA_USED 属性および NOQUOTA_USED 属性の値を変更する必要はありません。

REGISTER および NOREGISTER

カテゴリ	REGISTER	NOREGISTER
属性のデータ型	キーワード	キーワード
最小値	該当なし	該当なし
最大値	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	該当なし	SERVICE
属性との競合	NOREGISTER、NOALTERNATE	REGISTER、LOCATION
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	DESTINATION	DESTINATION
関連する $ARCHIVE_DEST 列	TARGET	TARGET

目的

LOG_ARCHIVE_DEST_*n* パラメータの REGISTER 属性および NOREGISTER 属性は、アーカイブ REDO ログの位置を宛先サイトに記録するかどうかを示します。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに REGISTER 属性および NOREGISTER 属性が指定されていない場合、デフォルトは REGISTER です。

属性

REGISTER

REGISTER 属性は、アーカイブ REDO ログの位置が、対応する宛先に記録されることを示します。

フィジカル・スタンバイ宛先の場合、アーカイブ REDO ログのファイル名は、接続先データベースの制御ファイルに記録され、管理リカバリ操作で使用されます。

ロジカル・スタンバイ・データベースの場合、アーカイブ REDO ログのファイル名は、ロジカル・スタンバイ・データベースの制御ファイルでメンテナンスされている表領域に記録され、SQL 適用操作で使用されます。

REGISTER 属性は、宛先が Data Guard のスタンバイ・データベースであることを意味します。

デフォルトでは、リモート宛先サイトでのアーカイブ REDO ログの位置は、宛先インスタンスの初期化パラメータ、STANDBY_ARCHIVE_DEST と LOG_ARCHIVE_FORMAT によって決まります。

注意： 各スタンバイ・データベースで SQL の ALTER DATABASE REGISTER LOGFILE *filespec* 文を実行して、REGISTER 属性を設定することもできます。この SQL 文の例は、7-15 ページの「フェイルオーバーの手順」を参照してください。

NOREGISTER

オプションの NOREGISTER 属性は、アーカイブ REDO ログの位置が、対応する宛先に記録されないことを示します。これは、リモートの宛先についてのみの設定です。各アーカイブ REDO ログの場所は、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。

NOREGISTER 属性は、宛先が Data Guard 構成の一部ではないスタンバイ・データベースである場合に必要です。

例

次の例は、REGISTER 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_5='REGISTER'
```

REGISTER=location_format

カテゴリ	REGISTER=*location_format*
属性のデータ型	文字列値
最小値	該当なし
最大値	該当なし
デフォルト値	該当なし
必須属性	DEPENDENCY
属性との競合	NOREGISTER、LOCATION、TEMPLATE
属性クラス	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	DESTINATION
関連する V$ARCHIVE_DEST 列	TARGET

目的

オプションの REGISTER=*location_format* 属性を使用すると、アーカイブ REDO ログのファイル名形式テンプレートを指定できます。これは、プライマリおよびスタンバイ・データベースの初期化パラメータ・ファイルに定義されている、デフォルトのファイル名形式テンプレートとは異なります。

この属性は、フィジカル・スタンバイ・データベースでのみ使用されます。

デフォルト

この属性にデフォルトはありません。

属性

REGISTER=*location_format*

オプションの REGISTER=*location_format* 属性を使用して、アーカイブ REDO ログに完全修飾されたファイル名形式テンプレートを指定します。このテンプレートは、プライマリおよびスタンバイ・データベースの初期化パラメータ・ファイルに定義されている、デフォルトのファイル名形式テンプレートとは異なります。デフォルトのファイル名形式テンプレートは、データベースの初期化パラメータ、STANDBY_ARCHIVE_DEST と LOG_ARCHIVE_FORMAT を組み合せたものです。

関連項目： 12-39 ページ「REGISTER および NOREGISTER」

REGISTER=*location_format* 属性は、リモートの宛先のみに有効です。

例

次の例は、REGISTER=*location_format* 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_4='REGISTER=/disk1/oracle/oradata/payroll/arc%d_%t_%s.arc'
```

REOPEN および NOREOPEN

カテゴリ	REOPEN [=*seconds*]	NOREOPEN
属性のデータ型	数値	キーワード
最小値	0 秒	該当なし
最大値	無制限秒数	該当なし
デフォルト値	300 秒	該当なし
必須属性	該当なし	該当なし
属性との競合	NOREOPEN	REOPEN
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	REOPEN_SECS	REOPEN_SECS
関連する V$ARCHIVE_DEST 列	MAX_FAILURE	MAX_FAILURE

目的

LOG_ARCHIVE_DEST_*n* パラメータの REOPEN 属性および NOREOPEN 属性は、アーカイバ・プロセス（ARC*n*、フォアグラウンドまたはログ・ライター・プロセス）が以前に失敗した宛先から再度アクセスしようとするまでの最小時間を秒単位で指定します。NOREOPEN を指定すると、この属性をオフにできます。

デフォルト

LOG_ARCHIVE_DEST_*n* パラメータに REOPEN 属性または NOREOPEN 属性が指定されていない場合、デフォルトは REOPEN です。
REOPEN 属性に整数値が指定されていない場合、デフォルトは 300 秒です。

属性

REOPEN[=*seconds*]
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。

OPTIONAL 宛先に REOPEN を指定した場合は、エラーが起こった場合でも、Oracle データベース・サーバーによるオンライン REDO ログの上書きが可能です。MANDATORY 宛先に REOPEN を指定した場合、ログ転送サービスは、REDO ログを正常にアーカイブできないと、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、OPTIONAL で宛先を指定する、またはサービス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。

REOPEN 属性を使用する場合は、次のことに注意します。

- アーカイバ・プロセスまたはログ・ライター・プロセスが宛先を再オープンするのは、ログの先頭からアーカイブ操作を開始するときのみで、現行の操作の途中で宛先を再オープンすることはありません。アーカイブ・プロセスでは常に先頭からログ・コピーが行われます。
- アーカイブ・プロセスは、記録されたエラーの時刻に REOPEN の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によって、REOPEN 属性に使用した値が指定した値であるかデフォルト値であるかをチェックします。現在の時刻に達していない場合は、その宛先に対するアーカイブ操作が再試行されます。
- LOG_ARCHIVE_DEST_*n* 初期化パラメータの MAX_FAILURE=*count* 属性を指定することで、ログ・アーカイブに障害が起きた後に宛先に対し再試行する回数を制御できます。

NOREOPEN

NOREOPEN を指定すると、障害が発生した宛先は、次の手順を実行するまで無効のままになります。

- 手動で宛先を再度使用可能にする。
- ALTER SYSTEM SET 文または ALTER SESSION SET 文を REOPEN 属性を指定して実行する。
- インスタンスを再起動する。

例

次の例は、REOPEN 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 MANDATORY REOPEN=60'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

SYNC および ASYNC

カテゴリ	SYNC[=*parallel_option*]	ASYNC[=*blocks*]
属性のデータ型	キーワード	数値
最小値	該当なし	0 ブロック
最大値	該当なし	20,480 ブロック
デフォルト値	該当なし	2,048
必須属性	該当なし	LGWR
属性との競合	ASYNC	SYNC、LOCATION、ARCH
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SYSTEM のみ
対応する V$ARCHIVE_DEST 列	TRANSMIT_MODE	TRANSMIT_MODE
関連する V$ARCHIVE_DEST 列	該当なし	ASYNC_BLOCKS

目的

LOG_ARCHIVE_DEST_*n* パラメータの SYNC 属性および ASYNC 属性は、ログ・ライター・プロセス（LGWR）を使用している場合にネットワーク I/O 操作が同期または非同期のいずれで実行されるかを指定します。

> **注意：** プライマリ・データベースが 3 種類の保護モードのいずれかである場合、ログ・ライター・プロセスによるスタンバイ REDO ログのアーカイブ先は、自動的に SYNC モードになります。

デフォルト

SYNC 属性または ASYNC 属性が指定されていない場合、デフォルトは SYNC です。宛先がデフォルトの SYNC に設定されている場合、または PARALLEL 修飾子を指定せずに SYNC 属性が指定された場合、PARALLEL 修飾子のデフォルトは、宛先に対して選択された転送プロセスによって決まります。LGWR 属性を指定すると、デフォルトのパラレル修飾子は PARALLEL になります。PARALLEL 修飾子は ARCH 属性と併用できないため、ARCH 属性を指定すると、デフォルトのパラレル修飾子は NOPARALLEL になります。

ASYNC 属性に整数値が指定されていない場合、デフォルトは 2048 ブロックです。

属性

SYNC=PARALLEL
SYNC=NOPARALLEL
SYNC 属性は、宛先に対するネットワーク I/O が同期して実行されることを指定します。これは、I/O が開始されると、アーカイブ・プロセスが I/O の完了を待ってから処理を続行することを意味します。SYNC 属性は非データ消失環境の設定に必要です。つまり、この属性によって、処理を継続する前に REDO レコードがスタンバイ・サイトに正常に転送されたことが確認されます。

ログ・ライター・プロセスが、SYNC 属性を使用する複数のスタンバイ宛先への転送を行うものと定義されている場合、ユーザーは、それらの各宛先に対して SYNC=PARALLEL または SYNC=NOPARALLEL を指定できます。

- SYNC=NOPARALLEL を使用すると、ログ・ライター・プロセスは各宛先にネットワーク I/O を連続して実行します。つまり、ログ・ライター・プロセスは、最初の宛先への I/O を開始すると、その終了を待たずに次の宛先への I/O を開始します。SYNC=NOPARALLEL 属性を指定することは、ASYNC=0 属性を指定することと同じです。
- SYNC=PARALLEL を使用すると、ネットワーク I/O は非同期に開始されます。したがって、複数の宛先への I/O をパラレルに開始できます。ただし、I/O が開始されると、ログ・ライター・プロセスは各 I/O 操作の完了を待ってから処理を続行します。事実上、これは同時に複数の同期 I/O 操作を実行することと同じです。SYNC=PARALLEL は、SYNC=NOPARALLEL よりも高い頻度で使用される可能性があります。

これは、PARALLEL 修飾子および NOPARALLEL 修飾子では、複数の宛先が関係している場合にのみ、相違が生じるためです。したがって、オラクル社ではすべての宛先に同じ値を使用することをお薦めします。

ASYNC[=*blocks*]
ASYNC 属性は、宛先に対するネットワーク I/O が非同期で実行されることを指定します。I/O が開始されると、ログ・ライターは、I/O の完了を待機せず、I/O の完了ステータスをチェックせずに次の要求の処理を続行します。ASYNC 属性を使用すると、プライマリ・データベースにほとんど影響を与えることなくスタンバイ環境をメンテナンスできます。オプションのブロック数によって、使用する SGA ネットワーク・バッファのサイズが決まります。一般に、ネットワーク接続の速度が遅いほど、ブロック数が大きくなります。また、ASYNC=0 属性を指定することは、SYNC=NOPARALLEL 属性を指定することと同じです。

ASYNC 属性の使用では、ネットワーク I/O を開始するためのイベントがいくつかあります。

- LGWR 要求が、現在使用可能なバッファ領域を超えた場合には、既存のバッファがスタンバイ・データベースへ転送されます。LGWR プロセスは、十分なバッファ領域が再度使用可能になるまで停止されます。
- プライマリ・データベースのログ・スイッチは、ログ・スイッチ操作が完了する前にバッファ内のすべての REDO ログをスタンバイ・データベースに転送します。

- プライマリ・データベースが正常に停止した場合。プライマリ・データベースを即時停止すると、バッファ内の REDO ログは廃棄されます。バッファ内の REDO ログは、スタンバイ・データベースの停止によっても廃棄されます。
- プライマリ・データベースが、一定期間 REDO アクティビティを行わなかったとき。データベースが非アクティブな状態の継続時間はシステムで決定され、ユーザーによる変更はできません。
- REDO 生成の速さが、実行時のネットワーク待機時間を超え、十分な領域が使用可能な場合、LGWR 要求はバッファに書き込まれます。領域が十分でない場合は、既存のバッファがスタンバイ・データベースへ転送されます。LGWR プロセスは、十分なバッファ領域が再度使用可能になるまで停止されます。

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 LGWR SYNC'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

TEMPLATE および NOTEMPLATE

カテゴリ	TEMPLATE=*filename_template*	NOTEMPLATE
属性のデータ型	文字列値	該当なし
最小値	該当なし	該当なし
最大値	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	該当なし
属性との競合	NOTEMPLATE、LOCATION、REGISTER=*location_format*	TEMPLATE
属性クラス	ALTER SESSION と ALTER SYSTEM	ALTER SESSION と ALTER SYSTEM
対応する V$ARCHIVE_DEST 列	REMOTE_TEMPLATE	REMOTE_TEMPLATE
関連する V$ARCHIVE_DEST 列	REGISTER	REGISTER

目的

LOG_ARCHIVE_DEST_*n* パラメータの TEMPLATE 属性および NOTEMPLATE 属性は、スタンバイ宛先でのディレクトリ仕様、およびアーカイブ REDO ログの形式テンプレートを定義します。この属性は、プライマリまたはスタンバイ初期化パラメータ・ファイルのいずれかに指定できます。ただし、属性が適用されるのは、アーカイブしているデータベース・ロールに対してのみです。

TEMPLATE 属性は、リモート・アーカイブ先の初期化パラメータ STANDBY_ARCHIVE_DEST および LOG_ARCHIVE_FORMAT の設定を上書きします。

TEMPLATE 属性および NOTEMPLATE 属性は、リモート・アーカイブ先でのみ有効です。

> **注意：** LGWR の宛先で使用される場合、ARC*n* プロセスによる再アーカイブでは、TEMPLATE 指定は使用されません。これは、保護されている宛先にとって重要です。

デフォルト

この属性にデフォルトはありません。

属性

TEMPLATE=*filename_template*
オプションの TEMPLATE 属性を使用して、スタンバイ宛先でのディレクトリ指定およびアーカイブ REDO ログの形式を定義します。この定義は、スタンバイ宛先の STANDBY_ARCHIVE_DEST および LOG_ARCHIVE_FORMAT 初期化パラメータで定義されたデフォルトのファイル名形式とは異なるファイル名の生成に使用されます。

TEMPLATE 属性の *filename_template* 値には、スレッドまたは順序番号のディレクティブを含める必要があります。次の表は、各ディレクティブの定義を示しています。

スレッドまたは順序番号のディレクティブ	説明
%a	データベース・アクティブ ID を代入します。
%A	0（ゼロ）を埋め込んだデータベース・アクティブ ID を代入します。
%d	データベース ID を代入します。
%D	0（ゼロ）を埋め込んだデータベース ID を代入します。
%t	インスタンス・スレッド番号を代入します。
%T	0（ゼロ）を埋め込んだインスタンス・スレッド番号を代入します。
%s	ログ・ファイル順序番号を代入します。
%S	0（ゼロ）を埋め込んだログ・ファイル順序番号を代入します。

filename_template 値は、スタンバイ宛先に転送され、そこで変換および検証された後、ファイル名が作成されます。

TEMPLATE 属性が指定されていない場合、設定は、REGISTER と同じになります。

NOTEMPLATE
オプションの NOTEMPLATE 属性を使用すると、初期化パラメータ STANDBY_ARCHIVE_DEST および LOG_ARCHIVE_FORMAT によって定義されたファイル名形式テンプレートを有効にできます。

例

次の例は、TEMPLATE 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_1='SERVICE=stby1 MANDATORY REOPEN=5
    TEMPLATE=/usr/oracle/prmy1/p1_%t_%s.dbf'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

prmy1 はリモート宛先で REDO ログをアーカイブします。stby1 は、p1_<thread#>_<sequence#>.dbf のファイル形式で、ディレクトリ /usr/oracle/prmy1 に配置されます。

アーカイブ先の属性の互換性

LOG_ARCHIVE_DEST_*n* 初期化パラメータには、多数の属性があります。これらの属性の中には、他の属性と競合するものがあります。また、属性の中には、定義すると必ず他の属性も定義する必要が生じるものがあります。表 12-2 は、サポートする属性と、それぞれの要件を示します。

表 12-2 LOG_ARCHIVE_DEST_*n* 属性の互換性

属性	必須	競合する属性
AFFIRM	該当なし	NOAFFIRM
NOAFFIRM	該当なし	AFFIRM
ALTERNATE=*destination*	該当なし	NOALTERNATE
NOALTERNATE	該当なし	ALTERNATE
ARCH	該当なし	LGWR ASYNC NET_TIMEOUT
ASYNC[=*blocks*]	LGWR	SYNC LOCATION ARCH
DELAY	SERVICE	LOCATION NODELAY
NODELAY	該当なし	DELAY
DEPENDENCY	SERVICE REGISTER	LOCATION NODEPENDENCY NOREGISTER QUOTA_SIZE QUOTA_USED

表 12-2 LOG_ARCHIVE_DEST_*n* 属性の互換性（続き）

属性	必須	競合する属性
NODEPENDENCY	該当なし	DEPENDENCY
LGWR	該当なし	ARCH
LOCATION	該当なし	SERVICE DEPENDENCY REGISTER=*location_format* NOREGISTER DELAY ASYNC NET_TIMEOUT TEMPLATE
MANDATORY	該当なし	OPTIONAL
MAX_FAILURE	REOPEN	NOMAX_FAILURE
NOMAX_FAILURE	該当なし	MAX_FAILURE
NET_TIMEOUT	LGWR と SYNC=PARALLEL または LGWR と ASYNC > 0	ARCH LOCATION NONET_TIMEOUT LGWR と SYNC=NOPARALLEL LGWR と ASYNC=0
NONET_TIMEOUT	該当なし	NET_TIMEOUT
OPTIONAL	該当なし	MANDATORY
QUOTA_SIZE	LOCATION	DEPENDENCY SERVICE NOQUOTA_SIZE
NOQUOTA_SIZE	該当なし	QUOTA_SIZE
QUOTA_USED	LOCATION	DEPENDENCY SERVICE NOQUOTA_USED
NOQUOTA_USED	該当なし	QUOTA_USED
REGISTER	該当なし	NOALTERNATE NOREGISTER
NOREGISTER	SERVICE	LOCATION REGISTER
REGISTER= *location_format*	DEPENDENCY	LOCATION NOREGISTER TEMPLATE
REOPEN	該当なし	NOREOPEN

表 12-2 LOG_ARCHIVE_DEST_*n* 属性の互換性（続き）

属性	必須	競合する属性
NOREOPEN	該当なし	REOPEN
SERVICE	該当なし	LOCATION QUOTA_USED QUOTA_SIZE
SYNC[= *parallel_option*]	該当なし	ASYNC
TEMPLATE	SERVICE	NOTEMPLATE LOCATION REGISTER=*location_format*
NOTEMPLATE	該当なし	TEMPLATE

13

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 文の概要について説明します。構成オプションは、次のとおりです。

```
ALTER DATABASE ACTIVATE STANDBY DATABASE
ALTER DATABASE ADD [STANDBY] LOGFILE
ALTER DATABASE ADD [STANDBY] LOGFILE MEMBER
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA
ALTER DATABASE COMMIT TO SWITCHOVER
ALTER DATABASE CREATE STANDBY CONTROLFILE AS
ALTER DATABASE DROP [STANDBY] LOGFILE
ALTER DATABASE DROP [STANDBY] LOGFILE MEMBER
ALTER DATABASE [NO]FORCE LOGGING
ALTER DATABASE MOUNT STANDBY DATABASE
ALTER DATABASE OPEN READ ONLY
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
ALTER DATABASE REGISTER LOGFILE
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE}
ALTER DATABASE START LOGICAL STANDBY APPLY
ALTER DATABASE {STOP | ABORT} LOGICAL STANDBY APPLY
```

関連項目： これらの SQL 文および他の SQL 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

ALTER DATABASE ACTIVATE STANDBY DATABASE

この文は、強制フェイルオーバー操作を実行します。これによって、プライマリ・データベースは Data Guard 環境から削除され、スタンバイ・データベースがプライマリ・データベースの役割を果たすと想定されます。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。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE ACTIVATE [PHYSICAL | LOGICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]];
```

表 13-1 は、この文のキーワードを示しています。

表 13-1 ACTIVATE STANDBY DATABASE 句のキーワード

キーワード	説明
PHYSICAL	フィジカル・スタンバイ・データベースをアクティブ化します。これがデフォルトです。
LOGICAL	ロジカル・スタンバイ・データベースをアクティブ化します。複数のロジカル・スタンバイ・データベースがある場合は、最初に、すべてのロジカル・スタンバイ・サイトで同一のログ・データが使用できるようにする必要があります。
SKIP [STANDBY LOGFILE] (フィジカル・スタンバイ・データベースのみ)	RECOVER MANAGED STANDBY DATABASE FINISH 句を使用してリカバリできるデータが、スタンバイ REDO ログに含まれている場合にも、フェイルオーバー操作の継続を強制します。SKIP [STANDBY LOGFILE] 句を使用すると、スタンバイ REDO ログの内容を廃棄しても構わないことが示されます。

注意： 強制フェイルオーバー操作ではなく、ALTER DATABASE RECOVER MANAGED STANDBY DATABASE 文で FINISH または FINISH SKIP のキーワードを使用して、できるかぎりフェイルオーバー操作を実行することをお薦めします。強制フェイルオーバー操作では、新規にアクティブ化されたプライマリ・データベースに対するスタンバイ・データベースとしての使用には適さない、フェイルオーバー操作に関係しない他のスタンバイ・データベースがレンダリングされます。

ALTER DATABASE ADD [STANDBY] LOGFILE

この文は、指定したスレッドに 1 つ以上の REDO ログのグループを追加し、スレッドに割り当てられたインスタンスがログを使用できるようにします。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE ADD [STANDBY] LOGFILE [THREAD integer] [GROUP integer] [REUSE] SIZE
filespec;
```

表 13-2 は、この文のキーワードを示しています。

表 13-2 ADD STANDBY LOGFILE 句のキーワード

キーワード	説明
STANDBY	作成された REDO ログを使用するのは、スタンバイ・データベースのみであることを示します。
THREAD *integer*	パラレル・モードで Real Application Clusters オプションを使用している場合にのみ適用可能です。*integer* 変数はスレッド番号です。
GROUP *integer*	すべてのスレッドの全グループの中で、REDO ログのグループを一意に識別します。1 から MAXLOGFILES 値の範囲で識別できます。同一の GROUP 値を持つ複数の REDO ログ・グループを追加することはできません。
filespec	1 人以上のメンバーが含まれる REDO ログ・グループを指定します。
REUSE	ログがすでに存在する場合は、REUSE を指定すると、Data Guard がログのヘッダー情報を上書きできます。
SIZE	ログのサイズをバイト単位で指定します。サイズを KB または MB で指定するには、K または M を使用します。

この SQL 文の詳細は、5-10 ページの「スタンバイ REDO ログ・グループの作成」を参照してください。

ALTER DATABASE ADD [STANDBY] LOGFILE MEMBER

この文は、新しいメンバーを既存の REDO ログのグループに追加します。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE ADD [STANDBY] LOGFILE MEMBER 'filename' [REUSE] TO logfile-descriptor;
```

表 13-3 は、この文のキーワードを示しています。

表 13-3 ADD STANDBY LOGFILE MEMBER 句のキーワード

キーワード	説明
STANDBY	ログのメンバーを使用するのは、スタンバイ・データベースのみであることを示します。STANDBY キーワードは必須ではありませんが、スクリプト内の対称性を保つために、必要に応じて、このキーワードを使用できます。
LOGFILE MEMBER *'filename'*	既存の REDO ログ・グループに新しいメンバーを追加します。新しい各メンバーを *'filename'* に指定します。
REUSE	*'filename'* に指定したファイルがすでに存在する場合、そのログのサイズは他のグループ・メンバーと同一であることが必要で、Oracle サーバーが既存のログを上書きできるように REUSE を指定する必要があります。ログが存在しない場合は、正しいサイズのログが作成されます。
logfile_descriptor	次のいずれかの方法を使用して、既存のログ・グループを指定します。 ■ GROUP *integer* パラメータを指定します。*integer* は、既存のログ・グループの番号です。ログ・グループがスタンバイ・データベースでの使用のために追加された場合、そのログ・グループのすべてのメンバーを使用するのは、スタンバイ・データベースのみです。 ■ REDO ログ・グループの各メンバーに対する完全ファイル名指定をリストします。オペレーティング・システムの表記規則に従ってファイル名を指定します。

この SQL 文の詳細は、5-11 ページの「スタンバイ REDO ログ・メンバーを既存のグループに追加」を参照してください。

ALTER DATABASE ADD SUPPLEMENTAL LOG DATA

この文は、ロジカル・スタンバイ・データベース専用です。

サプリメンタル・ロギングを使用可能にした後に、ロジカル・スタンバイ・データベースを作成する必要があります。これは、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です。サプリメンタル・ロギングを実装するには、PRIMARY KEY COLUMNS または UNIQUE INDEX COLUMNS を指定する必要があります。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA {PRIMARY KEY | UNIQUE INDEX} COLUMNS;
```

表 13-4 は、この文のキーワードを示しています。

表 13-4 ADD SUPPLEMENTAL LOG DATA 句のキーワード

キーワード	説明
PRIMARY KEY	主キーを持つすべての表に対して、更新操作が実行されるたびに主キーのすべての列を REDO ログに記録できるようにします。主キーが定義されていない場合は、行を一意に識別する一連の列が REDO ログに記録されます。
UNIQUE INDEX	一意索引を持つすべての表に対して、一意索引のいずれかの列が変更された場合に、その一意索引に属する他のすべての列も REDO ログに記録できるようにします。

この SQL 文の詳細は、4-9 ページの「サプリメンタル・ロギングが使用可能であることの確認」を参照してください。

ALTER DATABASE COMMIT TO SWITCHOVER

この文を使用してスイッチオーバー操作を実行し、現行のプライマリ・データベースをスタンバイ・データベース・ロールに変更し、1 つのスタンバイ・データベースをプライマリ・ロールに変更します。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ロジカル・スタンバイ・データベースのいずれのデータベースで文を発行するかによって、指定する SQL 文の句が異なります。

- フィジカル・スタンバイ・データベース・ロールで動作するように変更するプライマリ・データベースから文を発行する場合は、次の SQL 文の構文を使用します。

  ```
  ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY [[WITH | WITHOUT]
  SESSION SHUTDOWN ] [WAIT | NOWAIT];
  ```

- ロジカル・スタンバイ・データベース・ロールで動作するように変更するプライマリ・データベースから文を発行する場合は、次の SQL 文の構文を使用します。

  ```
  ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY [WAIT | NOWAIT];
  ```

- プライマリ・データベース・ロールで動作するように変更するフィジカル・スタンバイ・データベースから文を発行する場合は、次の SQL 文の構文を使用します。

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY [[WITH | WITHOUT] SESSION
SHUTDOWN ] [WAIT | NOWAIT];
```

- プライマリ・データベース・ロールで動作するように変更するロジカル・スタンバイ・データベースから文を発行する場合は、次の SQL 文の構文を使用します。

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY [WAIT | NOWAIT];
```

表 13-5 は、この文のキーワードを示しています。

表 13-5 COMMIT TO SWITCHOVER 句のキーワード

キーワード	説明
COMMIT TO SWITCHOVER TO PHYSICAL STANDBY	プライマリ・データベースをフィジカル・スタンバイ・データベースのロールで動作するように推移します。フィジカル・スタンバイ・データベースはマウントされている必要があり、READ ONLY モードでオープン状態で構いません。
COMMIT TO SWITCHOVER TO LOGICAL STANDBY	プライマリ・データベースをロジカル・スタンバイ・データベースのロールで動作するように推移します。このオプションの後に、ALTER DATABASE START LOGICAL STANDBY APPLY 文を指定する必要があります。
COMMIT TO SWITCHOVER TO [PHYSICAL | LOGICAL] PRIMARY	スタンバイ・データベースをプライマリ・データベースのロールで動作するように推移します。フィジカル・スタンバイ・データベースの場合のみ、スタンバイ・データベースはマウントされている必要があり、READ ONLY モードでオープン状態で構いません。（たとえば、スクリプトなどで）対称性を保つために、PHYSICAL または LOGICAL のパラメータを指定できますが、これらのキーワードは必須ではありません。
WITH SESSION SHUTDOWN （フィジカル・スタンバイ・データベースのみ）	オープン中のアプリケーション・セッションを停止し、この文の実行部分でコミットされていないトランザクションをロールバックします。 ロジカル・スタンバイ・データベースは、WITH SESSION SHUTDOWN オプションをサポートしていません。
WITHOUT SESSION SHUTDOWN （フィジカル・スタンバイ・データベースのみ）	アプリケーション・セッションがオープン中の場合は、スイッチオーバー操作が失敗します。これがデフォルトです。 ロジカル・スタンバイ・データベースは、WITHOUT SESSION SHUTDOWN オプションをサポートしていません。

表 13-5 COMMIT TO SWITCHOVER 句のキーワード（続き）

キーワード	説明
WAIT	スイッチオーバー操作の完了を待機してから制御をユーザーに戻します。これがデフォルトです。
NOWAIT	スイッチオーバー操作の完了を待たずに制御をユーザーに戻します。

この SQL 文の詳細は、7-12 ページの「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」および 7-20 ページの「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」を参照してください。

ALTER DATABASE CREATE STANDBY CONTROLFILE AS

この文は、フィジカル・スタンバイ・データベース専用です。

この文を使用して、スタンバイ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。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'filename' [REUSE];
```

表 13-6 は、この文のキーワードを示しています。

表 13-6 CREATE STANDBY CONTROLFILE AS 句のキーワード

キーワード	説明
CONTROLFILE AS *'filename'*	スタンバイ・データベースを保持するために作成および使用される制御ファイル名を指定します。
REUSE	*'filename'* パラメータで指定されている制御ファイルがすでに存在する場合は、Oracle サーバーが既存のファイルを上書きできるように REUSE を指定する必要があります。

この SQL 文の詳細は、3-5 ページの「スタンバイ・データベース用の制御ファイルの作成」を参照してください。

ALTER DATABASE DROP [STANDBY] LOGFILE

この句は、REDO ログ・グループのすべてのメンバーを削除します。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE DROP [STANDBY] LOGFILE logfile_descriptor;
```

表 13-7 は、この文のキーワードを示しています。

表 13-7 DROP [STANDBY] LOGFILE 句のキーワード

キーワード	説明
STANDBY	REDO ログ・グループのすべてのメンバーを削除します。対称性を保つために STANDBY を指定できますが、このキーワードは必須ではありません。
logfile_descriptor	次のいずれかの方法を使用して、既存の REDO ログ・グループを指定します。 ■ GROUP *integer* パラメータを指定します。*integer* は、既存のログ・グループの番号です。 ■ REDO ログ・グループに対する完全ファイル名指定をリストします。オペレーティング・システムの表記規則に従ってファイル名を指定します。

この SQL 文の使用例は、8-16 ページの「オンライン REDO ログの追加または削除」を参照してください。

ALTER DATABASE DROP [STANDBY] LOGFILE MEMBER

この文を使用して、1 つ以上の REDO ログのメンバーを削除します。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE DROP [STANDBY] LOGFILE MEMBER 'filename';
```

表 13-8 は、この文のキーワードを示しています。

表 13-8 DROP LOGFILE MEMBER 句のキーワード

キーワード	説明
STANDBY	1 つ以上のスタンバイ REDO ログのメンバーを削除します。対称性を保つために STANDBY を指定できますが、このキーワードは必須ではありません。
'filename'	ファイル名を指定します。複数のログ・メンバーを指定するには、各ファイル名をカンマで区切ります。各 *filename* には、オペレーティング・システムの表記規則に従って、完全修飾されたファイル指定を記述する必要があります。

ALTER DATABASE [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・サーバーが記録するかどうかを制御します。[NO] FORCE LOGGING 句の用途は、次のとおりです。

- フィジカル・スタンバイ・データベースの場合は、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。
- ロジカル・スタンバイ・データベースの場合は、スタンバイ・データベースでのデータの可用性を確実にするために、この指定を使用することをお薦めします。

注意： オラクル社では、FORCE LOGGING 句を設定した後に、バックアップ操作を実行してスタンバイ・データベースを作成し、スタンバイ・データベースがアクティブである間は強制ロギング・モードを維持することをお薦めします。また、強制ロギング・モードでのパフォーマンス低下を防止するために、データベースは ARCHIVELOG モードで実行してください。

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。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE [NO]FORCE LOGGING;
```

表 13-9 は、この文のキーワードを示しています。

表 13-9 [NO]FORCE LOGGING 句のキーワード

キーワード	説明
FORCE LOGGING	一時表領域および一時セグメントの変更を除く、データベース内のすべての変更を記録します。この設定は、個々の表領域に指定する NOLOGGING または FORCE LOGGING の設定、および個々のデータベース・オブジェクトに指定する NOLOGGING 設定よりも優先され、これらの設定との依存関係はありません。進行中でロギングされていないすべての操作を終了してから、強制ロギングを開始する必要があります。
NOFORCE LOGGING	強制ロギング・モードを取り消します。NOFORCE LOGGING がデフォルトです。

ALTER DATABASE MOUNT STANDBY DATABASE

フィジカル・スタンバイ・データベースをマウントし、スタンバイ・インスタンスがアーカイブ REDO ログをプライマリ・インスタンスから受信できるようにします。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE MOUNT STANDBY DATABASE;
```

ALTER DATABASE OPEN READ ONLY

この文は、フィジカル・スタンバイ・データベースの場合に必要です。ロジカル・スタンバイ・データベースにも使用できます。

読取り専用モードでフィジカル・スタンバイ・データベースをオープンします。この SQL 文は、ユーザーを読取り専用トランザクションに制限し、ユーザーが REDO ログを生成しないようにします。この句を使用すると、アーカイブ・ログがプライマリ・データベース・サイトからコピーされている間でも、フィジカル・スタンバイ・データベースを問合せに使用できるようになります。

フィジカル・スタンバイ・データベースは、マウントした後にオープンする必要があります。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE OPEN READ ONLY;
```

この SQL 文の詳細は、8-6 ページの「読取り専用アクセス用にスタンバイ・データベースをオープン」を参照してください。

ALTER DATABASE RECOVER MANAGED STANDBY DATABASE

この文は、フィジカル・スタンバイ・データベース専用です。

この文を使用して、管理リカバリ操作およびフィジカル・スタンバイ・データベースに対するログ適用サービスの開始、制御および取消しを行います。RECOVER MANAGED STANDBY DATABASE 句は、マウント、オープンまたはクローズしているデータベースで使用できます。この SQL 文には、その他の句は必須ではありませんが、管理リカバリ・プロセスを制御するために役立つ多くのオプションがあります。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [ startup_clause | modify_clause | cancel_
clause ];
```

startup_clause

管理リカバリ操作を開始すると、フォアグラウンドまたはバックグラウンド・セッションでログ適用サービスを開始できます。

- フォアグラウンド・セッションを開始するための SQL 文の構文は、次のとおりです。

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [TIMEOUT | NO TIMEOUT];
```

- バックグラウンド・セッションを開始するための SQL 文の構文は、次のとおりです。

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT [FROM SESSION] [NO TIMEOUT];
```

modify_clause

RECOVER MANAGED STANDBY DATABASE 句には、管理リカバリ・プロセス、スイッチオーバー操作およびフェイルオーバー操作を制御するための豊富なオプションがあります。一部の特定のフェイルオーバー操作とスイッチオーバー操作を除いて、管理リカバリ操作を開始したセッションがフォアグラウンドかバックグラウンドかに関係なく、これらのキーワードの動作は同じです。

FINISH キーワードを使用してフェイルオーバー操作を開始する場合を除いて、キーワードは、どのような順番で SQL 文に配置しても構いません。FINISH キーワードは、SQL 文の最後に指定する必要があります。

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE [
[ NO TIMEOUT | TIMEOUT [integer] ]
[ NODELAY | DELAY [integer] ]
[ DEFAULT DELAY ]
[ NO EXPIRE | EXPIRE [integer] ]
[ NEXT [integer] ]
[ NOPARALLEL | PARALLEL [integer]]
[ THROUGH { ALL | NEXT | LAST } SWITCHOVER ]
[ THROUGH ALL ARCHIVELOG  [ THREAD n ] SEQUENCE n ]
[ FINISH [ SKIP [STANDBY LOGFILE] [NOWAIT | WAIT] ] ]
]
```

cancel_clause

管理リカバリ・セッションを停止するための SQL 文の構文は、次のとおりです。

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL [IMMEDIATE] [NOWAIT];
```

表 13-10 は、すべてのキーワードを示しています。

表 13-10 RECOVER MANAGED STANDBY DATABASE 句のキーワード

キーワード	説明	互換性のないキーワード
CANCEL [IMMEDIATE] [NOWAIT]	管理リカバリを終了します。デフォルトでは、ログ適用サービスは現行のアーカイブ REDO ログの適用を終了してから停止します。 IMMEDIATE を指定すると、アーカイブ REDO ログから別のブロックを読み込む前、または次のアーカイブ REDO ログをオープンする前のいずれかのうち、先に発生したタイミングで、管理リカバリを終了します。 NOWAIT を指定すると、管理リカバリ・プロセスの終了を待たずに、CANCEL 文を発行したプロセスに制御が戻ります。	その他すべてのキーワード
DELAY *integer*	個々のアーカイブ REDO ログを適用するまでにログ適用サービスが待機する絶対遅延間隔（分単位）を指定します。適用遅延間隔は、アーカイブ REDO ログがリカバリのために選択された時点で計測が開始されます。	CANCEL、FINISH、NODELAY
DEFAULT DELAY	遅延間隔を、プライマリ・データベースの LOG_ARCHIVE_DEST_*n* 初期化パラメータに指定された時間（分単位）に戻します。	CANCEL、DELAY、FINISH、NODELAY
DISCONNECT [FROM SESSION]	管理リカバリ・プロセス（MRP）およびログ適用サービスをバックグラウンド・サーバー・プロセスで開始します。	CANCEL、TIMEOUT
NEXT *integer*	ログ転送サービスがアーカイブを終了した後、ログ適用サービスができるかぎり早く適用する必要がある遅延アーカイブ REDO ログの数を指定します。	CANCEL、FINISH
EXPIRE *integer*	管理リカバリ・プロセスが自動的に終了するまでの時間（分単位）を、現在の時刻に相対的に指定します。ログ適用サービスは、現行のアーカイブ REDO ログの適用を終了してから停止します。	CANCEL、FINISH、NO EXPIRE
FINISH [SKIP [STANDBY LOGFILE]] [NOWAIT | WAIT]	フェイルオーバー操作を起動します。この操作では、最初に使用可能なすべてのアーカイブ REDO ログを適用し、次に使用可能なスタンバイ REDO ログをリカバリします。 SKIP [STANDBY LOGFILE] を指定すると、スタンバイ REDO ログの内容の適用をスキップしても構わないことを示します。 NOWAIT を指定すると、リカバリが完了する前に制御がフォアグラウンド・プロセスに戻ります。 WAIT を指定すると、リカバリ完了後に制御が戻ります。	CANCEL、DELAY、EXPIRE、NEXT、THROUGH、TIMEOUT
NODELAY	以前に指定した DELAY オプションを使用禁止にして、ログ適用サービスが遅延なしでアーカイブ REDO ログをスタンバイ・データベースに適用します。	CANCEL、DELAY
NO EXPIRE	以前に指定した EXPIRE オプションを使用禁止にします。	CANCEL、EXPIRE
NO TIMEOUT	以前に指定した TIMEOUT オプションを使用禁止にします。	CANCEL

表 13-10 RECOVER MANAGED STANDBY DATABASE 句のキーワード（続き）

キーワード	説明	互換性のないキーワード
NOPARALLEL	以前に指定した PARALLEL オプションを使用禁止にします。その結果、ログ適用サービスが単一のプロセスを使用して後続のすべてのアーカイブ REDO ログを適用するようになります。これがデフォルトです。	CANCEL
PARALLEL [*integer*]	追加のパラレル・リカバリ処理を開始し、アーカイブ REDO ログを複数のスタンバイ・データ・ファイルに同時に適用するワークロードを分散します。デフォルトでは、すべてのアクティブ・インスタンスで使用可能な CPU の数と PARALLEL_THREADS_PER_CPU 初期化パラメータの値を乗算した数と同じ数のパラレル処理が選択されます。ただし、*integer* を指定すると、パラレル操作で使用するパラレル処理の数を示すことができます。各パラレル・スレッドでは、1 つまたは 2 つのパラレル実行サーバーを使用できます。	CANCEL
THROUGH [THREAD *n*] SEQUENCE *n*	リカバリするアーカイブ REDO ログのスレッド番号と順序番号を指定します。指定したアーカイブ REDO ログが適用されると、管理リカバリは終了します。THREAD *n* キーワードはオプションです。THREAD *n* を指定しない場合、デフォルトはスレッド 1 です。	CANCEL、FINISH
THROUGH ALL ARCHIVELOG	管理リカバリ・モードにデフォルト動作を指定します。これによって、管理リカバリ・モードが明示的に停止されるまで、管理リカバリは継続します。この句は、THROUGH THREAD *n* SEQUENCE *n* キーワードを使用して現在実行中である管理リカバリが、指定のアーカイブ REDO ログを適用した後に停止しないように変更する場合に役立ちます。	CANCEL、FINISH

表 13-10 RECOVER MANAGED STANDBY DATABASE 句のキーワード（続き）

キーワード	説明	互換性のないキーワード
THROUGH {ALL|NEXT|LAST} SWITCHOVER	スイッチオーバー操作後に、新規プライマリ・データベースから受信したアーカイブ REDO ログを適用する、ログ適用サービスを維持します。（デフォルトでは、アーカイブ REDO ログ内のスイッチオーバー（end-of-redo）・マーカーに到達すると、ログ適用サービスは停止します。） ALL －明示的に取り消すまで管理リカバリを継続します。管理リカバリは、スイッチオーバー（end-of-redo）・インジケータをすべて通過（無視）します。ALL オプションは、スイッチオーバー操作に関係のない他のスタンバイ・データベースがあり、各スタンバイ・データベースでのリカバリ処理を停止および再開しない場合に役に立ちます。 NEXT －スイッチオーバー（end-of-redo）・インジケータを最初に検出した時点で管理リカバリを停止します。これはデフォルトの動作です。 LAST －すべてのスイッチオーバー（end-of-redo）・インジケータを通過して管理リカバリを継続し、最後に受信したアーカイブ REDO ログで end-of-redo マーカーを検出した場合にのみ管理リカバリを停止します。	CANCEL、FINISH
TIMEOUT *integer*	管理リカバリ・プロセスが、プライマリ・データベースからの次回のアーカイブ REDO ログを待機する時間（分単位）を指定します。指定時間内に次のログが到着しない場合、ログ適用サービスは自動的に停止します。 フォアグラウンド・セッションで管理リカバリを開始している場合にのみ、TIMEOUT を指定します。	CANCEL、DISCONNECT、FINISH

関連項目： ログ適用サービスおよび管理リカバリ・プロセスの制御に関する詳細は、6-5 ページの「管理リカバリ操作の開始」を参照してください。

ALTER DATABASE REGISTER LOGFILE

この句を使用して、手動でアーカイブ REDO ログを登録できます。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE REGISTER [OR REPLACE] [PHYSICAL | LOGICAL] LOGFILE filespec;
```

表 13-11 は、この文のキーワードを示しています。

表 13-11 REGISTER LOGFILE 句のキーワード

キーワード	説明
OR REPLACE	スタンバイ・データベースの既存のアーカイブ REDO ログ・エントリを更新できます（たとえば、アーカイブ REDO ログの位置またはファイル指定を変更する場合）。エントリの SCN が厳密に一致し、元のエントリがログ転送サービスによって作成されていることが必要です。
PHYSICAL	アーカイブ REDO ログがフィジカル・スタンバイ・データベースの制御ファイルに登録されることを示します。
LOGICAL	アーカイブ REDO ログがロジカル・スタンバイ・データベースのディクショナリに登録されることを示します。
LOGFILE *filespec*	1 人以上のメンバーが含まれる REDO ログ・グループを指定します。新しい各メンバーを *filespec* に指定します。

この SQL 文の使用例は、7-15 ページの「フェイルオーバーの手順」を参照してください。

ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {PROTECTION | AVAILABILITY | PERFORMANCE}

この文を使用して、データベース環境のデータに対する保護レベルを指定します。次の保護レベルの 1 つを使用して、プライマリ・データベースのデータ消失およびデータ分岐を保護できます。SQL 文の構文は、次のとおりです。

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

この文はプライマリ・データベースで実行します。実行する際は、データベースが停止してマウント状態であることが必要です。表 13-12 は、この文のキーワードを示しています。

表 13-12 SET STANDBY TO MAXIMIZE 句のキーワード

キーワード	説明
PROTECTION (フィジカル・スタンバイ・データベースのみ)	最も高い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データが、SYNC ログ転送モードを使用するように構成された、少なくとも 1 つのフィジカル・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・データベースがこのような少なくとも 1 つのスタンバイ・データベースに REDO レコードを書き込みできない場合、プライマリ・データベースは停止します。このモードはデータ消失がないことを保証しますが、プライマリ・データベースのパフォーマンスと可用性に非常に大きな影響を与える可能性があります。
AVAILABILITY	2 番目に高いレベルのデータ保護を提供します。このモードは、ネットワークの停止からリカバリするまでに、プライマリ・データベースに障害が発生しないかぎり、プライマリ・サイトと構成内の少なくとも 1 つのスタンバイ・サイトとの間で、データ消失がないことを保証します。その後、サイトに送り出された最後のトランザクションに至るまで、データの消失はありません（ネットワークの停止後にプライマリ・サイトで続行されたトランザクションは消失する可能性があります)。プライマリ・データベースがこのような少なくとも 1 つのスタンバイ・データベースに REDO レコードを書き込みできない場合、最大保護モードとは異なり、プライマリ・データベースは停止しません。ただし、状況が修正されてスタンバイ・データベースがプライマリ・データベースと同じ状態になるまで、保護はパフォーマンス・モードまで低下します。このモードは、最大パフォーマンス・モードの間にプライマリ・データベースで障害が発生しないかぎり、データ消失がないことを保証します。最大可用性モードは、プライマリ・データベースの可用性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。
PERFORMANCE	これはデフォルトの保護モードです。プライマリ・データベースのトランザクションは、トランザクションのリカバリが必要なデータがスタンバイ・データベースに書き込まれる前にコミットします。したがって、プライマリ・データベースに障害が発生した場合は、一部のデータが消失する可能性があり、REDO レコードをプライマリ・データベースからリカバリできません。このモードは、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。

データ保護モードの追加情報は、5-3 ページの「データ保護モード」を参照してください。

ALTER DATABASE START 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。

この文を使用して、ロジカル・スタンバイ・データベースでログ適用サービスを開始します。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE START LOGICAL STANDBY APPLY [INITIAL [scn-value] ] [NEW PRIMARY dblink];
```

表 13-13 は、この文のキーワードを示しています。

表 13-13 START LOGICAL STANDBY APPLY 句のキーワード

キーワード	説明
INITIAL [*scn-value*]	ログをロジカル・スタンバイ・データベースに初めて適用するときに、このキーワードを指定します。システム変更番号（SCN）が整数で指定される直前の、トランザクションの一貫性が保たれている状態にデータベースがリカバリされます。
NEW PRIMARY *dblink*	データベースのスイッチオーバーが発生した後にログ適用サービスを開始します。この文によって、アーカイブ REDO ログのすべてのトランザクションをロジカル・スタンバイ・データベースに正しく適用できます。このキーワードは、ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY 文の後、あるいはロジカル・スタンバイ・データベースであるプライマリ・データベースからのログの処理が完了し、新規データベースがプライマリ・データベースになったときに指定します。コマンドラインで提供されるデータベース・リンク（dblink）を使用して、プライマリ・データベースの表にアクセスします。このリンクは、プライマリ・データベースの表の読込みとロックができる権限アカウントを参照する必要があります。

この SQL 文の詳細は、4-23 ページの「アーカイブ REDO ログの登録と SQL 適用操作の開始」を参照してください。

ALTER DATABASE {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。

この句を使用して、ロジカル・スタンバイ・データベースでログ適用サービスを停止します。SQL 文の構文は、次のとおりです。

```
ALTER DATABASE { STOP | ABORT } LOGICAL STANDBY APPLY;
```

表 13-14 は、この文のキーワードを示しています。

表 13-14 {STOP | ABORT} LOGICAL STANDBY APPLY 句のキーワード

キーワード	説明
STOP	ロジカル・スタンバイ設定の変更または計画的なメンテナンスを実行できるように、正しい方法でログ適用サービスを停止します。この句は、マテリアライズド・ビューまたはファンクション・ベース索引をリフレッシュするために役立ちます。ログ転送サービスは、ロジカル・スタンバイ・データベースへのアーカイブ REDO ログの送信を継続します。
ABORT	ログ適用サービスをただちに停止します。DBA_LOGSTDBY_PARAMETERS ビューに表示される TRANSACTION_CONSISTENCY オプションが READ_ONLY または NONE に設定されている場合、ABORT によって、トランザクションは一貫性のない状態になる可能性があります。ただちに停止する必要がある場合にのみ、ABORT キーワードを使用します。

14
ビュー

この章では、Data Guard 環境で使用するビューについて説明します。これはデータベースで使用できるビューのサブセットです。この章は、次の項目で構成されています。

- V$LOGSTDBY（ロジカル・スタンバイ・データベースのみ）
- V$LOGSTDBY_STATS（ロジカル・スタンバイ・データベースのみ）
- V$MANAGED_STANDBY（フィジカル・スタンバイ・データベースのみ）
- V$STANDBY_LOG

ビューの概要

Oracle データベースには、サーバーによってメンテナンスされ、データベース管理者がアクセス可能な基礎となるビューのセットが格納されています。これらの**固定ビュー**は、データベースがオープンおよび使用中の際に連続的に更新されるので、**動的パフォーマンス・ビュー**とも呼ばれます。その内容は主にパフォーマンス関連です。

これらのビューは通常のデータベース表のように見えますが、実際はそうではありません。これらのビューには、内部ディスク構造およびメモリー構造のデータが表示されます。これらのビューから選択することはできますが、その内容を更新したり変更することはできません。

固定ビューの名前には、V$ または GV$ という接頭辞が付き、たとえば V$ARCHIVE_DEST や GV$ARCHIVE_DEST のようになります。DBA_ の接頭辞が付いているビューには、データベース全体に関連するすべての情報が表示されます。標準の動的パフォーマンス・ビュー（V$ 固定ビュー）には、ローカル・インスタンスの情報が格納されます。それに対して、グローバル動的パフォーマンス・ビュー（GV$ 固定ビュー）には、すべてのオープン・インスタンスの情報が格納されます。各 V$ 固定ビューには対応する GV$ 固定ビューがあります。DBA_ のビューは、管理者専用です。SELECT_ANY_TABLE 権限を持つユーザーのみが、これらのビューにアクセスできます。（この権限は、システムに最初にインストールされるときに DBA のロールに割り当てられます。）

多くの場合、固定ビューに表示される使用可能な情報は、インスタンスが停止しても変更されません。ただし、特定の固定ビューの情報は、インスタンスが停止するとリセットされます。このようなビューについては、この章で取り上げます。

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

DBA_LOGSTDBY_EVENTS（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_EVENTS ビューには、ロジカル・スタンバイ・データベース・システム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、ログ適用サービスが REDO ログを適用する際に発生する障害の原因を判断するために役立ちます。このビューには次の列が表示されます。

列	データ型	説明
EVENT_TIME	DATE	イベントが記録された時間です。
CURRENT_SCN	NUMBER	イベントの変更ベクトル SCN です。障害が発生した場合は、この列を検査して、障害のソース（サポートされないレコードなど）が含まれるアーカイブ REDO ログを判断します。
COMMIT_SCN	NUMBER	変更がコミットされた SCN 値です。
XIDUSN	NUMBER	トランザクション ID の UNDO セグメント番号です。
XIDSLT	NUMBER	トランザクション ID のスロット番号です。
XIDSQN	NUMBER	トランザクション ID の順序番号です。
EVENT	CLOB	障害が発生したときに処理されていた文です。
STATUS_CODE	NUMBER	STATUS メッセージに属するステータス（つまり、Oracle エラー・コード）です。
STATUS	VARCHAR2(2000)	処理の現行アクティビティまたは適用操作の停止理由の説明です。

DBA_LOGSTDBY_LOG（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_LOG ビューは、ロジカル・スタンバイ・データベースに登録されているログを表示します。ビューには次の列が表示されます。

列	データ型	説明
THREAD#	NUMBER	アーカイブ REDO ログのスレッド ID です。シングル・インスタンスの THREAD 番号は 1 です。Real Application Clusters の場合、この列には異なる番号が格納されます。
SEQUENCE#	NUMBER	アーカイブ REDO ログの順序番号です。
FIRST_CHANGE#	NUMBER	現行のアーカイブ REDO ログの SCN です。
NEXT_CHANGE#	NUMBER	次のアーカイブ REDO ログの SCN です。
FIRST_TIME	DATE	現行のアーカイブ REDO ログの日付です。
NEXT_TIME	DATE	次のアーカイブ REDO ログの日付です。
FILE_NAME	VARCHAR2(3)	アーカイブ REDO ログの名前です。
TIMESTAMP	DATE	アーカイブ REDO ログが登録された時刻です。
DICT_BEGIN	VARCHAR2(3)	Y または N の値です。Y は、ディクショナリ作成の開始がこの特定のアーカイブ REDO ログにあることを示します。
DICT_END	VARCHAR2(3)	Y または N の値です。Y は、ディクショナリ作成の終了がこの特定のアーカイブ REDO ログにあることを示します。

注意： このビューの SCN 値は DBA_LOGSTDBY_PROGRESS（ロジカル・スタンバイ・データベースのみ）ビューの SCN 値と相互に関係があります。

DBA_LOGSTDBY_NOT_UNIQUE（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_NOT_UNIQUE ビューは、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。このビューの列には、ロジカル・スタンバイ・データベースでメンテナンスされる十分な情報が格納されるため、このビューに表示されるほとんどの表はサポートされます。ただし、表によっては、その列に必要な情報が含まれていないためにサポートできないことがあります。サポートされない表には、ほとんどの場合、サポートされないデータ型で定義された列が含まれています。このビューには次の列が表示されます。

列	データ型	説明
OWNER	VARCHAR2(30)	スキーマ名です。
TABLE_NAME	VARCHAR2(30)	表の名前です。
BAD_COLUMN	VARCHAR2(1)	この列には、Y または N の値が格納されます。 ■ Y は、表の列が LONG または BLOB などのバインドされないデータ型を使用して定義されていることを示します。表の 2 つの行が、それらの LOB 以外の列で一致する場合は、表を適切にメンテナンスできません。ログ適用サービスがこれらの表のメンテナンスを試みますが、ユーザーは、バインドされていない列でアプリケーションが一意性を許可しないように配慮することが必要です。 ■ N は、ロジカル・スタンバイ・データベースの表をメンテナンスするために十分な列情報があることを示します。ただし、ログ転送サービスとログ適用サービスは、主キーを追加すると、さらに効率的に動作します。無効化された RELY 制約をこれらの表に追加することを考慮してください。

DBA_LOGSTDBY_PARAMETERS（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_PARAMETERS ビューには、ロジカル・スタンバイ・データベースのログ適用サービスが使用するパラメータのリストがあります。このビューには次の列が表示されます。

列	データ型	説明
NAME	VARCHAR2 (30)	パラメータの名前です。使用される値は次のとおりです。 ■ MAX_SGA －ログ適用サービスのキャッシュ用に割り当てられている System Global Area（SGA）（メガバイト単位）を示します。 ■ MAX_SERVERS －ログ適用サービス用に特に予約されているパラレル問合せサーバーの数を示します。 ■ MAX_EVENTS_RECORDED － DBA_LOGSTDBY_EVENTS ビューに格納されているイベント数を示します。 ■ TRANSACTION_CONSISTENCY －保持されるトランザクションの一貫性のレベル、つまり FULL、READ_ONLY または NONE が表示されます。 ■ RECORD_SKIP_ERRORS －スキップされるレコードを示します。 ■ RECORD_SKIP_DDL －スキップされた DDL 文を示します。 ■ RECORD_APPLIED_DDL －適用された DDL 文を示します。 ■ FIRST_SCN －ログ転送サービスが REDO 情報の適用を開始する SCN を示します。 ■ PRIMARY －ログの適用先データベースのデータベース ID を示します。 ■ LMNR_SID － LogMiner セッション ID を示します。これは LogMiner セッションが使用中であることを示す内部値です。 ■ UNTIL_SCN －すべてのトランザクションが適用されるまでログ適用サービスが停止する SCN 値を示します。 ■ END_PRIMARY_SCN －この値は、スイッチオーバー時に、新規プライマリ・データベースが旧プライマリ・データベースから適用する最後の SCN を示します。 ■ NEW_PRIMARY_SCN －この値は、スイッチオーバー時に、新規プライマリ・データベースの開始 SCN 値を示します。 ■ COMPLETED_SESSION －ログ適用サービスのセッションが完了したことを示します。状況に応じて SWITCHOVER または FAILOVER の値を示します。
VALUE	VARCHAR2 (2000)	パラメータの値です。

DBA_LOGSTDBY_PROGRESS（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_PROGRESS ビューは、ロジカル・スタンバイ・データベースのログ適用サービスの進捗を示します。このビューには次の列が表示されます。

列	データ型	説明
APPLIED_SCN	NUMBER	すべての変更が適用された最新の SCN が表示されます。APPLIED_SCN 列と NEWEST_SCN 列の値は、使用可能な REDO ログ・データすべてが処理された場合に一致します。
APPLIED_TIME	DATE	APPLIED_SCN の予定日時です。
READ_SCN	NUMBER	この SCN より大きいすべてのログ・データが、読込みおよび保存されます。
READ_TIME	DATE	READ_SCN の予定日時です。
NEWEST_SCN	NUMBER	スタンバイ・システムで使用可能な最新の SCN です。スタンバイ・データベースに転送するログがこれ以上ない場合、この SCN に変更が適用されます。APPLIED_SCN 列と NEWEST_SCN 列の値は、使用可能な REDO ログ・データすべてが処理された場合に一致します。
NEWEST_TIME	DATE	NEWEST_SCN の予定日時です。

注意： このビューの SCN 値は DBA_LOGSTDBY_LOG（ロジカル・スタンバイ・データベースのみ）ビューの SCN 値と相互に関係があります。

DBA_LOGSTDBY_SKIP（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_SKIP ビューには、ログ適用サービスがスキップする表がリストされます。DBA_LOGSTDBY_SKIP ビューには次の列が表示されます。

列	データ型	説明
ERROR	BOOLEAN	文をスキップするか、文に対して単にエラーを戻すかを示します。
STATEMENT_OPT	VARCHAR(30)	スキップする文のタイプを指定します。このタイプとは、SYSTEM_AUDIT 文のオプションの 1 つであることが必要です。
SCHEMA	VARCHAR(30)	このスキップ・オプションを使用するスキーマの名前です。
NAME	VARCHAR(30)	このスキップ・オプションを使用するオプションの名前です。
PROC	VARCHAR(98)	スキップ・オプションの処理時に実行されるストアド・プロシージャの名前です。

DBA_LOGSTDBY_SKIP_TRANSACTION（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_SKIP_TRANSACTION ビューには、選択したスキップ設定がリストされます。このビューには次の列が表示されます。

列	データ型	説明
XIDUSN	NUMBER	トランザクション ID の UNDO セグメント番号です。
XIDSLT	NUMBER	トランザクション ID のスロット番号です。
XIDSQN	NUMBER	トランザクション ID の順序番号です。

DBA_LOGSTDBY_UNSUPPORTED（ロジカル・スタンバイ・データベースのみ）

DBA_LOGSTDBY_UNSUPPORTED ビューは、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。このビューには次の列が表示されます。

列	データ型	説明
OWNER	VARCHAR2(30)	サポートされない表のスキーマ名です。
TABLE_NAME	VARCHAR2(30)	サポートされない表の名前です。
COLUMN_NAME	VARCHAR2(30)	サポートされない列の名前です。
DATA_TYPE	VARCHAR2(106)	サポートされない列のデータ型です。

V$ARCHIVE_DEST

V$ARCHIVE_DEST ビューは、現行インスタンスについて、すべてのアーカイブ REDO ログの宛先、現在の値、モードおよび状態を記述します。

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_DEST ビューには次の列が表示されます。

列	説明
DEST_ID	ログのアーカイブ先のパラメータを識別します。
STATUS	宛先の現在の状態を識別します。使用される値は次のとおりです。 ■ VALID －初期化済みで使用可能 ■ INACTIVE －宛先情報なし ■ DEFERRED －ユーザーの手動設定により無効状態 ■ ERROR －オープン時またはコピー時にエラー発生 ■ DISABLED －エラーにより無効化 ■ BAD PARAM － LOG_ARCHIVE_DEST_*n* パラメータにエラーあり ■ ALTERNATE －代替状態の宛先 ■ FULL －宛先の割当て制限サイズを超過
BINDING	アーカイブ操作に対する障害の影響を指定します。使用される値は次のとおりです。 ■ OPTIONAL －正常なアーカイブ操作は不要 ■ MANDATORY －正常なアーカイブ操作が必要
NAME_SPACE	パラメータ設定の有効範囲を識別します。使用される値は次のとおりです。 ■ SYSTEM －システム定義 ■ SESSION －セッション定義
TARGET	アーカイブ先がプライマリ・データベースに対してローカルかリモートかを指定します。使用される値は次のとおりです。 ■ PRIMARY －ローカル ■ STANDBY －リモート

列	説明
ARCHIVER	問合せが発行されるデータベースに関連したアーカイバ・プロセスを識別します。使用される値は次のとおりです。 ■ ARC*n* ■ FOREGROUND ■ LGWR ■ RFS
SCHEDULE	この宛先のアーカイブが INACTIVE、PENDING、ACTIVE または LATENT のいずれであるかを示します。
DESTINATION	アーカイブ REDO ログをアーカイブする位置が表示されます。
LOG_SEQUENCE	アーカイブする最後のアーカイブ REDO ログの順序番号を識別します。
REOPEN_SECS	エラー後に再試行するまでの時間を秒単位で識別します。
DELAY_MINS	アーカイブ REDO ログがスタンバイ・データベースに自動的に適用されるまでの遅延間隔を、分単位で識別します。
PROCESS	問合せがスタンバイ・データベースで発行される場合でも、プライマリ・データベースに関連するアーカイバ・プロセスを識別します。使用される値は次のとおりです。 ■ ARC*n* ■ FOREGROUND ■ LGWR
REGISTER	アーカイブ REDO ログがリモートの宛先制御ファイルに登録されているかどうかを識別します。登録されている場合は、そのアーカイブ REDO ログを管理リカバリの操作に使用できます。使用される値は次のとおりです。 ■ YES ■ NO
FAIL_DATE	エラーの日時を示します。
FAIL_SEQUENCE	最後のエラーが発生した際にアーカイブされていたアーカイブ REDO ログの順序番号を示します。
FAIL_BLOCK	最後のエラーが発生した際にアーカイブされていたアーカイブ REDO ログのブロック番号を示します。
FAILURE_COUNT	宛先に対して連続的に発生した、アーカイブ操作の障害の現在までの回数を識別します。
MAX_FAILURE	ログ転送サービスが、障害が発生した宛先との通信を再確立してアーカイブ操作を再開するための試行回数を制御できます。
ERROR	エラーのテキストを表示します。
ALTERNATE	代替の宛先がある場合は、それを識別します。

列	説明
DEPENDENCY	依存アーカイブ先がある場合は、それを識別します。
REMOTE_TEMPLATE	記録される位置の導出に使用するテンプレートを表示します。
QUOTA_SIZE	バイト単位で表される、宛先の割当て制限を識別します。
QUOTA_USED	指定された宛先に現在あるすべてのアーカイブ REDO ログのサイズを識別します。
MOUNTID	インスタンスのマウント識別子を識別します。
AFFIRM	ディスク I/O モードを表示します。
ASYNC_BLOCKS	ASYNC 属性のブロック数を指定します。
TRANSMIT_MODE	ネットワーク転送モードが表示されます。使用される値は次のとおりです。 ■ PARALLELSYNC ■ SYNCHRONOUS ■ ASYNCHRONOUS
TYPE	アーカイブ・ログの宛先の定義が、PUBLIC と PRIVATE のいずれであるかを示します。PUBLIC の宛先のみ、実行時に ALTER SYSTEM SET 文または ALTER SESSION SET 文を使用して修正できます。デフォルトでは、すべてのアーカイブ・ログの宛先が PUBLIC になっています。
NET_TIMEOUT	ログ・ライター・プロセスが、発行したネットワーク操作に関するネットワーク・サーバーからのステータスを待機する秒数を指定します。

V$ARCHIVE_DEST_STATUS

V$ARCHIVE_DEST_STATUS ビューは、アーカイブ REDO ログ宛先の実行時および構成情報を表示します。

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_DEST_STATUS ビューには次の列が表示されます。

列	説明
DEST_ID	ログのアーカイブ先のパラメータを識別します。
STATUS	宛先の現在の状態を識別します。使用される値は次のとおりです。 ■ VALID －初期化済みで使用可能 ■ INACTIVE －宛先情報なし ■ DEFERRED －ユーザーの手動設定により無効状態 ■ ERROR －オープン時またはコピー時にエラー発生 ■ DISABLED －エラーにより無効化 ■ BAD PARAM － LOG_ARCHIVE_DEST_*n* パラメータにエラーあり ■ ALTERNATE －代替状態の宛先 ■ FULL －宛先の割当て制限サイズを超過
TYPE	アーカイブ先データベースのタイプを識別します。使用される値は次のとおりです。 ■ LOCAL －ローカルのプライマリ・インスタンス ■ PHYSICAL －フィジカル・スタンバイ・データベース ■ CROSS-INSTANCE －プライマリ・データベースのインスタンス
DATABASE_MODE	アーカイブ先データベースの現在のモードを識別します。使用される値は次のとおりです。 ■ STARTED －インスタンスは開始済み、マウントは未実行 ■ MOUNTED －マウント済み ■ MOUNTED-STANDBY －マウント済みスタンバイ ■ OPEN －読取り / 書込みでオープン ■ OPEN_READ-ONLY －読取り専用でオープン

列	説明
RECOVERY_MODE	アーカイブ先データベースのメディア・リカバリの現在のモードを識別します。使用される値は次のとおりです。 ■ IDLE －管理リカバリが非アクティブ ■ MANUAL －手動メディア・リカバリがアクティブ ■ MANAGED －管理リカバリがアクティブ
DESTINATION	アーカイブ REDO ログをアーカイブする位置が表示されます。
ARCHIVED_THREAD#	宛先で最後に受信されたアーカイブ REDO ログのスレッド番号を識別します。
ARCHIVED_SEQ#	宛先で最後に受信されたアーカイブ REDO ログのログ順序番号を識別します。
APPLIED_THREAD#	宛先で最後に適用された REDO ログのスレッド番号を識別します。
APPLIED_SEQ#	宛先で最後に適用された REDO ログのログ順序番号を識別します。
ERROR	エラーのテキストを表示します。
STANDBY_LOGFILE_COUNT	スタンバイ・データベースで作成されたスタンバイ REDO ログの合計数を示します。
STANDBY_LOGFILE_ACTIVE	プライマリ・データベースのオンライン REDO ログ情報を含むアクティブなスタンバイ・データベースにある、スタンバイ REDO ログの合計数を示します。
PROTECTION_MODE	データベースが保護されているかどうか、保護されている場合はその方法を示します。使用される値は次のとおりです。 ■ MAXIMUM PROTECTION ■ MAXIMUM AVAILABILITY ■ RESYNCHRONIZATION ■ MAXIMUM PERFORMANCE ■ UNPROTECTED
SRL	スタンバイ・データベースでのスタンバイ REDO ログの使用を示します。使用される値は次のとおりです。 ■ YES ■ NO

V$ARCHIVE_GAP

V$ARCHIVE_GAP ビューには、アーカイブ・ギャップの識別に役立つ情報が表示されます。V$ARCHIVE_GAP ビューには次の列が表示されます。

列	説明
THREAD#	スレッド番号を表示します。
LOW_SEQUENCE#	ログの番号の下限を指定します。
HIGH_SEQUENCE#	ログの番号の上限を指定します。

V$ARCHIVED_LOG

V$ARCHIVED_LOG ビューは、制御ファイルからのアーカイブ REDO ログ情報をアーカイブ・ログ名を含めて表示します。このビューには次の列が表示されます。

列	説明
RECID	アーカイブ・ログのレコード ID です。
STAMP	アーカイブ・ログのレコード・スタンプです。
NAME	アーカイブ・ログのファイル名です。NULL に設定されている場合、ログはアーカイブの前に消去されています。
DEST_ID	アーカイブ・ログの生成元の宛先です。宛先の識別子が使用できない場合、値は 0（ゼロ）になります。
THREAD#	REDO スレッド番号です。
SEQUENCE#	REDO ログ順序番号です。
RESETLOGS_CHANGE#	ログが記述された際の、データベースのリセットログ変更番号です。
RESETLOGS_TIME	ログが記述された際の、データベースのリセットログ・タイムです。
FIRST_CHANGE#	アーカイブ・ログ内の最初の変更番号です。
FIRST_TIME	最初の変更のタイムスタンプです。
NEXT_CHANGE#	次のログにおける最初の変更です。
NEXT_TIME	次の変更のタイムスタンプです。
BLOCKS	アーカイブ・ログのブロック単位のサイズです。
BLOCK_SIZE	REDO ログのブロックのサイズです。これはアーカイブ・ログの論理ブロックのサイズで、アーカイブ・ログのコピー元となったオンライン・ログの論理ブロックのサイズと同じです。オンライン・ログのブロックのサイズは、ユーザーが調整できないプラットフォーム固有の値です。
CREATOR	アーカイブ・ログの作成者を識別します。
REGISTRAR	エントリの登録者を識別します。
STANDBY_DEST	エントリがアーカイブ・ログの宛先かどうかを示します。

列	説明
ARCHIVED	オンライン REDO ログがアーカイブ済みであること、または Recovery Manager がログの検査のみを行い、将来のリカバリ時に適用する REDO ログのレコードを作成済みであることを示します。
APPLIED	アーカイブ・ログが対応するスタンバイ・データベースに適用済みかどうかを示します。
DELETED	Recovery Manager の DELETE コマンドが、アーカイブ REDO ログをディスクから物理的に削除し、同時に**ターゲット・データベース**の制御ファイルおよび**リカバリ・カタログ**から論理的に削除したかどうかを指定します。
STATUS	このアーカイブ・ログの状態です。使用される値は次のとおりです。 ■ A －使用可能 ■ D －削除済み ■ U －使用不能 ■ X －期限切れ
COMPLETION_TIME	アーカイブが完了した時刻を示します。
DICTIONARY_BEGIN	このログに LogMiner ディクショナリの開始が含まれるかどうかを示します。
DICTIONARY_END	このログに LogMiner ディクショナリの終了が含まれるかどうかを示します。
BACKUP_COUNT	このファイルがバックアップされた回数を示します。値の範囲は 0 から 15 です。ファイルが 15 回以上バックアップされている場合も、値は 15 のままです。
END_OF_REDO	このアーカイブ REDO ログに、プライマリ・データベースからの REDO 情報すべての終了が含まれるかどうかを示します。使用される値は YES および NO です。
ARCHIVAL_THREAD#	アーカイブ操作を実行したインスタンスの REDO スレッド番号を示します。この列が THREAD# 列と異なるのは、クローズされたスレッドが別のインスタンスによってアーカイブされている場合のみです。
ACTIVATION#	データベースのインスタンス化に割り当てられる番号を示します。

V$DATABASE

V$DATABASE ビューは、制御ファイルからのデータベース情報を表示します。このビューには次の列が表示されます。

列	説明
DBID	データベースの作成時に計算されるデータベースの識別子です。この識別子は、すべてのファイル・ヘッダーに格納されます。
NAME	データベースの名前です。
CREATED	作成の日付です。
RESETLOGS_CHANGE#	オープンしているリセットログの変更番号です。
RESETLOGS_TIME	オープンしているリセットログのタイムスタンプです。
PRIOR_RESETLOGS_CHANGE#	前回のリセットログの変更番号です。
PRIOR_RESETLOGS_TIME	前回のリセットログのタイムスタンプです。
LOG_MODE	アーカイブ・ログのモードです。
CHECKPOINT_CHANGE#	チェックポイント取得を実行した最後の SCN です。
ARCHIVE_CHANGE#	アーカイブした最後の SCN です。
CONTROLFILE_TYPE	制御ファイルのタイプです。使用される値は次のとおりです。 ■ STANDBY －データベースがスタンバイ・モードであることを示します。 ■ LOGICAL －データベースがロジカル・スタンバイ・データベースであることを示します。 ■ CLONE －クローン・データベースであることを示します。 ■ BACKUP | CREATED －データベースが、バックアップまたは作成した制御ファイルを使用してリカバリ中であることを示します。 ■ CURRENT －データベースが一般的に使用できることを示します。
CONTROLFILE_CREATED	制御ファイル作成時のタイムスタンプです。
CONTROLFILE_SEQUENCE#	制御ファイルのトランザクションごとに増加する、制御ファイルの順序番号です。
CONTROLFILE_CHANGE#	バックアップ制御ファイル内の最後の変更番号です。制御ファイルがバックアップでない場合は、NULL に設定されます。
CONTROLFILE_TIME	バックアップ制御ファイル内の最後のタイムスタンプです。制御ファイルがバックアップでない場合は、NULL に設定されます。

列	説明
OPEN_RESETLOGS	次のデータベースをオープンする際に、リセットログのオプションが許可されるかまたは必要とされるかどうかを示します。
VERSION_TIME	バージョン・タイムです。
OPEN_MODE	オープン・モード情報です。
PROTECTION_MODE	データベースが保護されているかどうか、保護されている場合はその方法を示します。使用される値は次のとおりです。 ■ MAXIMUM PROTECTION ■ MAXIMUM AVAILABILITY ■ RESYNCHRONIZATION ■ MAXIMUM PERFORMANCE ■ UNPROTECTED
PROTECTION_LEVEL	プライマリまたはスタンバイ・データベースで現在有効な保護モードの集約が表示されます。使用される値は次のとおりです。 ■ MAXIMUM PROTECTION ■ MAXIMUM AVAILABILITY ■ RESYNCHRONIZATION ■ MAXIMUM PERFORMANCE ■ UNPROTECTED
REMOTE_ARCHIVE	REMOTE_ARCHIVE_ENABLE 初期化パラメータの値です。使用される値は次のとおりです。 ■ TRUE ■ FALSE ■ SEND ■ RECEIVE
ACTIVATION#	データベースのインスタンス化に割り当てられる番号です。
DATABASE_ROLE	データベースの現在のロール（プライマリまたはスタンバイ）です。
ARCHIVELOG_CHANGE#	V$ARCHIVED_LOG ビューに表示される、アーカイブ・ログの最も大きい NEXT_CHANGE# です。

列	説明
SWITCHOVER_STATUS（フィジカル・スタンバイ・データベースのみ）	スイッチオーバーが許可されるかどうかを指定します。現在、この列はフィジカル・スタンバイ・データベースでのみサポートされています。使用される値は次のとおりです。 ■ NOT ALLOWED －これはスタンバイ・データベースでプライマリ・データベースが最初に切り替えられていないか、またはこれがプライマリ・データベースでスタンバイ・データベースがないかのいずれかです。 ■ SESSIONS ACTIVE －プライマリまたはスタンバイのデータベースに、スイッチオーバー操作を行う前にアクティブな SQL セッションがあることを示します。V$SESSION ビューに問い合せて、終了する必要のある特定のプロセスを識別します。 ■ SWITCHOVER PENDING －これはスタンバイ・データベースで、プライマリ・データベースのスイッチオーバー要求は受信されたものの、まだ処理されていないことを示します。 ■ SWITCHOVER LATENT －スイッチオーバーは保留モードにあったものの、完了せずにプライマリ・データベースに戻されたことを示します。 ■ TO PRIMARY －これはスタンバイ・データベースで、プライマリ・データベースへのスイッチオーバーが許可されていることを示します。 ■ TO STANDBY －これはプライマリ・データベースで、スタンバイ・データベースへのスイッチオーバーが許可されていることを示します。 ■ RECOVERY NEEDED －これはスタンバイ・データベースで、スイッチオーバー要求を受信していないことを示します。
GUARD_STATUS	データが変更されないように保護します。使用される値は次のとおりです。 ■ ALL － SYS 以外のすべてのユーザーが、データベース内のデータを変更しないように保護します。 ■ STANDBY － SYS 以外のすべてのユーザーが、ロジカル・スタンバイ・データベースで保持しているデータベース・オブジェクトに対して変更を行わないように保護します。 ■ NONE －データベース内の全データに対して標準的なセキュリティで保護することを意味します。
SUPPLEMENTAL_LOG_DATA_MIN	連鎖行や様々な記憶域割当て（クラスタ表など）をサポートするために、LogMiner が十分な情報を取得できるようにします。 ALTER DATABASE ADD SUPPLEMENTAL LOG DATA 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
SUPPLEMENTAL_LOG_DATA_PK	主キーを持つすべての表に対して、更新操作が実行されるたびに主キーのすべての列を REDO ログに記録できるようにします。 ALTER DATABASE ADD SUPPLEMENTAL LOG DATA 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

列	説明
SUPPLEMENTAL_LOG_DATA_UI	一意キーを持つすべての表に対して、一意キーのいずれかの列が変更された場合に、その一意キーに属するその他すべての列も REDO ログに記録できるようにします。 ALTER DATABASE ADD SUPPLEMENTAL LOG DATA 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
FORCE_LOGGING	NOLOGGING 操作に対しても REDO ログの生成を強制実行します。使用される値は次のとおりです。 ■ YES ■ NO
DATAGUARD_BROKER	Data Guard 構成がブローカによって管理されているかどうかを示します。使用される値は次のとおりです。 ■ ENABLED －構成がブローカの管理下にあることを示します。 ■ DISABLED －構成がブローカの管理下にないことを示します。

V$DATAFILE

V$DATAFILE ビューは、制御ファイルからのデータ・ファイル情報を表示します。このビューには次の列が表示されます。

列	説明
FILE#	ファイルの ID 番号です。
CREATION_CHANGE#	データ・ファイルが作成された際の変更番号です。
CREATION_TIME	データ・ファイル作成時のタイムスタンプです。
TS#	表領域番号です。
RFILE#	表領域の相対的なデータ・ファイル番号です。
STATUS	ファイルのタイプ（システムまたはユーザー）およびその状態です。使用される値は次のとおりです。 ■ OFFLINE －書込み不能 ■ ONLINE －書込み可能 ■ SYSTEM －システムのデータ・ファイル ■ RECOVER －リカバリ必要 ■ SYSOFF －オフラインのシステム
ENABLED	SQL からファイルへのアクセス可能性を説明します。使用される値は次のとおりです。 ■ DISABLED － SQL アクセスはできません。 ■ READ ONLY － SQL 更新はできません。 ■ READ WRITE －完全なアクセスが可能です。
CHECKPOINT_CHANGE#	最後のチェックポイントの SCN です。
CHECKPOINT_TIME	最後のチェックポイントのタイムスタンプです。
UNRECOVERABLE_ CHANGE#	このデータ・ファイルに作成された、前回のリカバリ不能な変更番号です。この列は、リカバリ不能な操作が完了すると必ず更新されます。
UNRECOVERABLE_TIME	前回のリカバリ不能な変更のタイムスタンプです。
LAST_CHANGE#	このデータ・ファイルに作成された、前回の変更番号です。データ・ファイルが変更されている場合は、NULL に設定します。
LAST_TIME	前回の変更のタイムスタンプです。
OFFLINE_CHANGE#	前回のオフライン範囲の、オフライン変更番号です。この列は、データ・ファイルがオンラインになった際にのみ更新されます。

列	説明
ONLINE_CHANGE#	前回のオフライン範囲の、オンライン変更番号です。
ONLINE_TIME	前回のオフライン範囲の、オンラインのタイムスタンプです。
BYTES	現行のデータ・ファイルのバイト単位によるサイズです。アクセス不能な場合は 0（ゼロ）になります。
BLOCKS	現行のデータ・ファイルのブロック単位によるサイズです。アクセス不能な場合は 0（ゼロ）になります。
CREATE_BYTES	バイト単位による作成時のサイズです。
BLOCK_SIZE	データ・ファイルのブロックのサイズです。
NAME	データ・ファイルの名前です。
PLUGGED_IN	表領域がプラグインされているかどうかを示します。表領域がプラグインされていても読取り / 書込み可能になっていない場合、値は 1 になり、そうでない場合は 0（ゼロ）になります。
BLOCK1_OFFSET	ファイルの冒頭から Oracle の一般情報が開始する箇所までのオフセットです。ファイルの厳密な長さは、BYTES + BLOCK1_OFFSET で計算できます。
AUX_NAME	このファイルに設定された補助の名前です。

V$DATAGUARD_STATUS

V$DATAGUARD_STATUS ビューは、通常、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をトリガーとするイベントを表示および記録します。

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$DATAGUARD_STATUS ビューには次の列が表示されます。

列	説明
INST_ID	イベントを検出したインスタンスの ID です。この列は、GV$DATAGUARD_STATUS ビューには表示されますが、V$DATAGUARD_STATUS ビューには表示されません。
FACILITY	イベントを検出した機能です。使用される値は次のとおりです。 ■ CRASH RECOVERY ■ LOG TRANSPORT SERVICES ■ LOG APPLY SERVICES ■ ROLE MANAGEMENT SERVICES ■ REMOTE FILE SERVER ■ FETCH ARCHIVE LOG ■ DATA GUARD ■ NETWORK SERVICES
SEVERITY	イベントの重大度です。使用される値は次のとおりです。 ■ INFORMATIONAL －連絡メッセージを示します。 ■ WARNING －警告メッセージを示します。 ■ ERROR －プロセスの失敗を示します。 ■ FATAL －プロセスまたはデータベース、あるいはその両方の障害を示します。 ■ CONTROL －アーカイブ、ログ・リカバリまたはスイッチオーバー操作の開始または完了など、予定上の状態の変化を示します。
DEST_ID	イベントが関係している宛先の ID 番号です。イベントが特定の宛先に関係していない場合、値は 0（ゼロ）です。
MESSAGE_NUM	発生順に増加する番号で、各イベントに対して一意の番号を指定します。
ERROR_CODE	イベントに関係するエラー ID です。

列	説明
CALLOUT	現行のエントリがコールアウト・イベントかどうかを示します。使用される値は次のとおりです。 ■ YES ■ NO YES の値は、このイベントに対して、データベース管理者がなんらかの処置を行う必要性があることを意味します。詳細は、ERROR_CODE および MESSAGE の列を確認してください。 NO の値は、通常、データベース管理者による処置を必要としない INFORMATIONAL または WARNING のイベントに相当します。
TIMESTAMP	エントリが作成された日時です。
MESSAGE	イベントを説明するテキスト・メッセージです。

V$LOG

V$LOG ビューには、オンライン REDO ログからのログ・ファイル情報が格納されます。このビューには次の列が表示されます。

列	説明
GROUP#	ログ・グループの番号です。
THREAD#	ログのスレッド番号です。
SEQUENCE#	ログ順序番号です。
BYTES	ログのバイト単位のサイズです。
MEMBERS	ログ・グループのメンバーの数です。
ARCHIVED	アーカイブ状態です。
STATUS	ログの状態を示します。使用される値は次のとおりです。 ■ UNUSED －オンライン REDO ログは一度も書き込まれていません。これは追加された直後の REDO ログか、あるいは RESETLOGS オプションを指定した直後（カレント REDO ログでない場合）の REDO ログのステータスです。 ■ CURRENT －これはカレント REDO ログです。これは REDO ログがアクティブであることを意味します。REDO ログはオープンまたはクローズの場合があります。 ■ ACTIVE －ログはアクティブですが、カレント・ログではありません。このログは障害リカバリのために必要です。ブロックのリカバリに使用される場合もあります。アーカイブされている場合も、されていない場合もあります。 ■ CLEARING －ログは ALTER DATABASE CLEAR LOGFILE 文の後の空のログとして再作成中です。ログが消去されると、状態は UNUSED に変わります。 ■ CLEARING_CURRENT －カレント・ログからクローズのスレッドを消去中です。切替えに、新しいログ・ヘッダーを記述する I/O エラーといったようななんらかの障害がある場合、ログはこの状態にとどまることができます。 ■ INACTIVE －ログはインスタンス・リカバリには不必要となりました。管理リカバリに使用される場合もあります。アーカイブされている場合も、されていない場合もあります。 ■ INVALIDATED －カレント REDO ログをログ・スイッチなしでアーカイブした場合です。
FIRST_CHANGE#	ログ内の最小の SCN です。
FIRST_TIME	ログ内の最初の SCN の時間です。

V$LOGFILE

V$LOGFILE ビューには、オンライン REDO ログの情報が格納されます。このビューには次の列が表示されます。

列	説明
GROUP#	REDO ログ・グループの識別子番号です。
STATUS	このログ・メンバーの状態です。使用される値は次のとおりです。 ■ INVALID －ファイルはアクセス不能です。 ■ STALE －内容が不完全です。 ■ DELETED －ファイルはすでに使用されていません。 ■ 空白（値なし）－ファイルは使用中です。
MEMBER	REDO ログのメンバー名です。
TYPE	ログがスタンバイ・ログかオンライン・ログかを指定します。使用される値は次のとおりです。 ■ STANDBY ■ ONLINE

V$LOG_HISTORY

V$LOG_HISTORY ビューには、制御ファイルからのログ履歴情報が格納されます。このビューには次の列が表示されます。

列	説明
RECID	制御ファイルのレコード ID です。
STAMP	制御ファイルのレコード・スタンプです。
THREAD#	アーカイブ・ログのスレッド番号です。
SEQUENCE#	アーカイブ・ログの順序番号です。
FIRST_CHANGE#	ログ内の最小の SCN です。
FIRST_TIME	ログ内の最初のエントリ（最小の SCN）の時間です。
NEXT_CHANGE#	ログ内の最大の SCN です。

V$LOGSTDBY（ロジカル・スタンバイ・データベースのみ）

V$LOGSTDBY ビューは、ログ適用サービスに発生している状況について動的な情報を提供します。スタンバイ・データベースに対するアーカイブ REDO ログのロジカル・アプリケーションで、パフォーマンス上の問題を診断している場合、このビューは非常に有用です。また、その他の問題についても有効な場合があります。V$LOGSTDBY ビューには次の列が表示されます。

列	データ型	説明
SERIAL#	NUMBER	SQL セッションのシリアル番号が格納されます。このデータは、このビューを V$SESSION ビューおよび V$PX_SESSION ビューと結合するときに使用されます。
LOGSTDBY_ID	NUMBER	パラレル問合せのスレーブ ID が格納されます。
PID	VARCHAR2(9)	プロセス ID が格納されます。
TYPE	VARCHAR2(30)	プロセス、つまり COORDINATOR、APPLIER、ANALYZER、READER、PREPARER、BUILDER が実行しているタスクを示します。
STATUS_CODE	NUMBER	STATUS メッセージに属するステータス番号（または Oracle エラー・コード）が格納されます。
STATUS	VARCHAR2(256)	プロセスの現行アクティビティの説明です。
HIGH_SCN	NUMBER	プロセスごとの最大の SCN が格納されます。この列は、個々のプロセスの進捗を確認するために使用されます。

V$LOGSTDBY_STATS（ロジカル・スタンバイ・データベースのみ）

V$LOGSTDBY_STATS ビューには、LogMiner 統計、現行のステータスおよび SQL 適用操作時のロジカル・スタンバイ・データベースのステータス情報が表示されます。ログ適用サービスが実行されていない場合、統計の値は消去されます。このビューには次の列が表示されます。

列	データ型	説明
NAME	VARCHAR2(64)	統計、状態またはステータスの名前です。 **注意：**次の統計の多くは、変更または削除の対象になります。プログラマは、統計の欠落または余分な統計を許容するように、アプリケーション・コードを記述してください。 ■ プリペアラの数 ■ アプライヤの数 ■ LCR キャッシュの最大 SGA ■ 使用しているパラレル・サーバー ■ トランザクションの一貫性 ■ コーディネータの状態 ■ スケジューリングされたトランザクション ■ 適用されたトランザクション ■ プリペアラ・メモリー割当て障害 ■ ビルダー・メモリー割当て障害 ■ ロー・メモリーに対する処理試行 ■ 成功したロー・メモリー・リカバリ ■ 回避したメモリー・スピル ■ ロールバックの試行 ■ 成功したロールバック ■ メモリー・スピル試行 ■ 成功したメモリー・スピル ■ プリペアラが無視したメモリー・ロー水位標 ■ ビルダーが無視したメモリー・ロー水位標 ■ 再開されたマイニング
VALUE	VARCHAR2(64)	統計または状態情報の値です。

V$MANAGED_STANDBY（フィジカル・スタンバイ・データベースのみ）

V$MANAGED_STANDBY ビューには、Data Guard 環境のフィジカル・スタンバイ・データベースに関連する Oracle データベース・サーバー・プロセスに関する現在の状態情報が表示されます。V$MANAGED_STANDBY ビューには、次の表にリストされている列があります。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。

列	説明
PROCESS	情報が報告されているプロセスのタイプです。使用される値は次のとおりです。 ■ ARCH －アーカイバ・プロセス ■ RFS －リモート・ファイル・サーバー ■ MRP0 －バックグラウンドの管理リカバリ・プロセス ■ MR(fg) －フォアグラウンドのリカバリ・セッション
PID	プロセスのオペレーティング・システム識別子です。
STATUS	現在のプロセスの状態です。使用される値は次のとおりです。 ■ UNUSED －アクティブなプロセスはありません。 ■ ALLOCATED －プロセスはアクティブですが、現在プライマリ・データベースに接続されていません。 ■ CONNECTED －プライマリ・データベースとネットワーク接続が確立されています。 ■ ATTACHED －プロセスはプライマリ・データベースと連結されて通信中です。 ■ IDLE －プロセスはアクティビティを実行していません。 ■ ERROR －プロセスは失敗しました。 ■ OPENING －プロセスはアーカイブ REDO ログをオープンしています。 ■ CLOSING －プロセスはアーカイブ操作を完了し、アーカイブ REDO ログをクローズしています。 ■ WRITING －プロセスはアーカイブ REDO ログのデータを記述しています。 ■ RECEIVING －プロセスはネットワーク通信を受信しています。 ■ ANNOUNCING －プロセスは依存アーカイブ REDO ログの存在の可能性を通知しています。 ■ REGISTERING －プロセスは完了した依存アーカイブ REDO ログの存在を登録しています。 ■ WAIT_FOR_LOG －プロセスはアーカイブ REDO ログの完了まで待機中です。 ■ WAIT_FOR_GAP －プロセスはアーカイブ・ギャップの解決まで待機中です。 ■ APPLYING_LOG －プロセスはアーカイブ REDO ログをスタンバイ・データベースに適用しています。

列	説明
CLIENT_PROCESS	対応するプライマリ・データベースのプロセスを識別します。使用される値は次のとおりです。 ■ ARCHIVAL ーフォアグラウンド（手動）のアーカイブ・プロセス（SQL） ■ ARCH ーバックグラウンドの ARC*n* プロセス ■ LGWR ーバックグラウンドの LGWR プロセス
CLIENT_PID	クライアント・プロセスのオペレーティング・システム識別子です。
CLIENT_DBID	プライマリ・データベースのデータベース識別子です。
GROUP#	スタンバイ REDO ログ・グループです。
THREAD#	アーカイブ REDO ログのスレッド番号です。
SEQUENCE#	アーカイブ REDO ログの順序番号です。
BLOCK#	最後に処理されたアーカイブ REDO ログのブロック番号です。
BLOCKS	アーカイブ REDO ログの、512 バイト・ブロック単位のサイズです。
DELAY_MINS	アーカイブ REDO ログの分単位による遅延間隔です。
KNOWN_AGENTS	アーカイブ REDO ログを処理中のスタンバイ・データベース・エージェントの合計数です。
ACTIVE_AGENTS	アーカイブ REDO ログを実際に処理中のスタンバイ・データベース・エージェントの数です。

V$STANDBY_LOG

V$STANDBY_LOG ビューには次の列が表示されます。

列	説明
GROUP#	ログ・グループの番号です。
THREAD#	ログのスレッド番号です。
SEQUENCE#	ログ順序番号です。
BYTES	ログのバイト単位のサイズです。
USED	ログに使用されているバイトの数です。
ARCHIVED	アーカイブ状態です。
STATUS	ログの状態を示します。使用される値は次のとおりです。 ■ UNUSED －オンライン REDO ログは一度も書き込まれていません。これは追加された直後の REDO ログか、あるいは RESETLOGS オプションを指定した直後（カレント REDO ログでない場合）の REDO ログのステータスです。 ■ CURRENT －これはカレント REDO ログです。これは REDO ログがアクティブであることを意味します。REDO ログはオープンまたはクローズの場合があります。 ■ ACTIVE －ログはアクティブですが、カレント・ログではありません。このログは障害リカバリのために必要です。ブロックのリカバリに使用される場合もあります。アーカイブされている場合も、されていない場合もあります。 ■ CLEARING －ログは ALTER DATABASE CLEAR LOGFILE 文の後の空のログとして再作成中です。ログが消去されると、状態は UNUSED に変わります。 ■ CLEARING_CURRENT －カレント・ログからクローズのスレッドを消去中です。切替えに、新しいログ・ヘッダーを記述する I/O エラーといったようななんらかの障害がある場合、ログはこの状態にとどまることができます。 ■ INACTIVE －ログはインスタンス・リカバリには不必要となりました。管理リカバリに使用される場合もあります。アーカイブされている場合も、されていない場合もあります。 ■ INVALIDATED －カレント REDO ログをログ・スイッチなしでアーカイブしました。
FIRST_CHANGE#	ログ内の最小の SCN です。
FIRST_TIME	ログ内の最初の SCN の時間です。
LAST_CHANGE#	このデータ・ファイルに作成された、最後の変更番号です。データ・ファイルが変更されている場合は、NULL に設定します。
LAST_TIME	最後の変更のタイムスタンプです。

第 III 部

付録および用語集

この部は、次の章で構成されています。

- 付録 A「スタンバイ・データベースのトラブルシューティング」
- 付録 B「手動リカバリ」
- 付録 C「Real Application Clusters でのスタンバイ・データベースのサポート」
- 付録 D「カスケードされた REDO ログ宛先」
- 付録 E「障害時リカバリに関する ReadMe ファイルのサンプル」
- 用語集

A

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。

- スタンバイ・データベースの準備における問題
- ログ宛先の障害
- ロジカル・スタンバイ・データベース障害の無視
- 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・データベースへの SQL 適用操作が停止した場合の処置
- REDO ログ転送のネットワーク調整
- Data Guard によるネットワーク・タイムアウトの管理

スタンバイ・データベースの準備における問題

スタンバイ・データベース準備中に発生する問題には次のようなものがあります。

- スタンバイ・アーカイブ宛先が適切に定義されない
- スタンバイ・サイトがプライマリ・データベースによってアーカイブされたログを受信しない
- 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

スタンバイ・アーカイブ宛先が適切に定義されない

STANDBY_ARCHIVE_DEST 初期化パラメータがスタンバイ・サイトの有効なディレクトリ名として定義されていない場合、Oracle データベース・サーバーは、アーカイブ REDO ログを格納するディレクトリを決定することができません。V$ARCHIVE_DEST ビューの DESTINATION 列および ERROR 列を調べます。たとえば、次のように入力します。

```
SQL> SELECT DESTINATION, ERROR FROM V$ARCHIVE_DEST;
```

宛先が有効であることを確認します。

スタンバイ・サイトがプライマリ・データベースによってアーカイブされたログを受信しない

スタンバイ・サイトがログを受信しない場合は、まず V$ARCHIVE_DEST ビューを問い合せて、プライマリ・データベースのアーカイブ状態について情報を取得します。特にエラー・メッセージについて調べます。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;

ID DB_status Archive_dest                   Error
-- --------- ------------------------------ ----------------------------------
 1  VALID    /vobs/oracle/work/arc_dest/arc
 2  ERROR    standby1                       ORA-16012: アーカイブ・ログのスタンバイ・データベース識別子が一致しません
 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、プライマリ・データベースはスタンバイ・サイトへのアーカイブを実行できません。

- プライマリ・サイトの 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。
- プライマリ初期化パラメータ・ファイルの LOG_ARCHIVE_DEST_*n* パラメータ内にリストされたサービス名が不適切な場合。
- スタンバイ・アーカイブ先の状態を指定する LOG_ARCHIVE_DEST_STATE_*n* パラメータが DEFER 値を持つ場合。
- listener.ora ファイルがスタンバイ・サイトで正しく構成されていない場合。
- 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初期化パラメータ・ファイルに追加したが、まだその変更を使用可能にしていない場合。
- 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）。

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

次のいずれかの条件に該当する場合、フィジカル・スタンバイ・データベースをマウントできません。

- スタンバイ・インスタンスが NOMOUNT モードで起動されていない場合。まずインスタンスを NOMOUNT で起動して次にデータベースをマウントする必要があります。
- ALTER DATABASE CREATE STANDBY CONTROLFILE ... 文または Recovery Manager でスタンバイ制御ファイルが作成されていない場合。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
  - 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

ログ宛先の障害

OPTIONAL 宛先に REOPEN を指定した場合は、エラーが発生した場合でも、Oracle データベース・サーバーによるオンライン REDO ログの再利用が可能です。MANDATORY 宛先に REOPEN を指定した場合は、ログ転送サービスのコンポーネントが REDO ログを正常にアーカイブできなかったとき、ログ転送サービスのコンポーネントによりプライマリ・データベースが停止されます。

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 再試行時間の設定と制限

```
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できます。スタンバイ・サイトへのオンライン REDO ログのアーカイブに障害が発生した場合に、代替アーカイブ先を使用します。アーカイブに障害が発生したときに、NOREOPEN 属性が指定されていなかった場合や MAX_FAILURE 属性のしきい値を超えた場合、ログ転送サービスは、次のアーカイブ操作で代替先への REDO ログのアーカイブを試行します。

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・ファイル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=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

宛先 LOG_ARCHIVE_DEST_1 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・スイッチで宛先を LOG_ARCHIVE_DEST_2 に自動的に切り替えます。

ロジカル・スタンバイ・データベース障害の無視

重要なスキップ・ツールに DBMS_LOGSTDBY.SKIP_ERROR があります。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のいずれを選択するかを実行時に決定できます。

> **関連項目：** DBMS_LOGSTDBY パッケージの PL/SQL コールアウト・プロシージャの使用方法については、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

これらの処理のいずれかを実行すると、SQL 適用操作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。

スタンバイ・データベースへのスイッチオーバーの問題

プライマリ・データベースからスタンバイ・データベースへのスイッチオーバーで発生する問題は次のいずれかの可能性があります。

- スイッチオーバーできない
- 失敗したスイッチオーバー操作をリカバリする
- 2 つ目のフィジカル・スタンバイ・データベースを起動できない
- アーカイブ REDO ログがスイッチオーバー後に適用されない
- SQL セッションがアクティブなときにスイッチオーバーできない

スイッチオーバーできない

ALTER DATABASE COMMIT TO SWITCHOVER がエラー・メッセージ 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」を伴って失敗しました。

このエラーは、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場合、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るためにクローズできないと発生します。

処置：すべてのユーザー・セッションを必ずデータベースから切断します。V$SESSION 固定ビューを問い合せると、まだ処理中のセッションを確認することができます。次に例を示します。

```
SQL> SELECT SID, PROCESS, PROGRAM FROM V$SESSION;

       SID PROCESS   PROGRAM
---------- --------- ------------------------------------------------
         1 26900     oracle@dbuser-sun (PMON)
         2 26902     oracle@dbuser-sun (DBW0)
         3 26904     oracle@dbuser-sun (LGWR)
         4 26906     oracle@dbuser-sun (CKPT)
         5 26908     oracle@dbuser-sun (SMON)
         6 26910     oracle@dbuser-sun (RECO)
         7 26912     oracle@dbuser-sun (ARC0)
         8 26897     sqlplus@dbuser-sun (TNS V1-V3)
        11 26917     sqlplus@dbuser-sun (TNS V1-V3)

9 rows selected.
```

前述の例で、最初の 7 つのセッションはすべてサーバーのバックグラウンド・プロセスです。2 つの SQL*Plus セッションの内、一方は問合せを発行中のカレントの SQL*Plus セッションで、他方はスイッチオーバー操作前に切断される必要のある余分のセッションです。

失敗したスイッチオーバー操作をリカバリする

通常、7-12 ページの「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操作」で説明した手順に従うと、スイッチオーバー操作は正常に行われます。ただし、最初のスイッチオーバー操作が失敗した場合は、次のリカバリ・オプションのいずれかを使用すると、スイッチオーバー操作を正常に完了できます。

オプション 1: 現行のスイッチオーバー操作を続行します。

スイッチオーバー操作が正常に完了しない場合は、V$ARCHIVED_LOG ビューの SEQUENCE# 列を問い合せて、最後のアーカイブ・ログが古いフィジカル・スタンバイ・データベースにアーカイブされ、適用されているかどうかを確認します。最後のログが古いフィジカル・スタンバイ・データベースにアーカイブされていない場合は、そのアーカイブ・ログを古いプライマリ・データベースから古いフィジカル・スタンバイ・データベースに手動でコピーし、SQL の ALTER DATABASE REGISTER LOGFILE filespec 文を使用して登録します。次に、管理リカバリ・プロセスを起動すると、アーカイブ・ログが自動的に適用されます。V$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であることを示します。

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

関連項目： V$DATABASE ビューの SWITCHOVER_STATUS 列に対するその他の有効な値については、第 14 章を参照してください。

スイッチオーバー操作を続行するには、7-12 ページの「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操作」の手順 5 に戻り、ターゲットのフィジカル・スタンバイ・データベースをプライマリ・ロールに切り替える操作を再度実行します。

オプション 2: 失敗したスイッチオーバー操作をロールバックして、最初からやり直します。
エラーが発生し、スイッチオーバー操作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. プライマリからスタンバイにロールを変更するスイッチオーバー操作を開始したときに、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、新しいフィジカル・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。
2. 元のフィジカル・スタンバイ・データベースを停止します。
3. 新しいフィジカル・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。スタンバイ制御ファイルを元のフィジカル・スタンバイ・サイトにコピーします。

   関連項目： スタンバイ制御ファイルの作成方法の詳細は、3-5 ページの「スタンバイ・データベース用の制御ファイルの作成」を参照してください。

4. 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 ログの順序番号が正しいことを確認します。

   関連項目： アーカイブ・ギャップ管理については、6-12 ページの「アーカイブ・ギャップの管理」を参照し、トレース・ファイルの位置の特定については、6-25 ページの「アーカイブ・トレースの設定」を参照してください。

5. スイッチオーバー操作を再度実行します。

   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操作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。

2 つ目のフィジカル・スタンバイ・データベースを起動できない

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。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 EXCLUSIVE 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・ファイル内に LOCK_NAME_SPACE パラメータを設定し忘れると、この現象がスイッチオーバー操作中に発生することがあります。スタンバイ・データベースの LOCK_NAME_SPACE パラメータが設定されていないと、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・ロックを使用し、2 つ目のデータベースの起動時に ORA-01102 エラーを発生させます。

処置：LOCK_NAME_SPACE=*unique_lock_name* をフィジカル・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・データベースの両方を停止し、再起動します。

アーカイブ REDO ログがスイッチオーバー後に適用されない

アーカイブ REDO ログがスイッチオーバー後にスタンバイ・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の後に、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いために発生することがあります。

処置：

- プライマリ・サイトの tnsnames.ora ファイルおよびスタンバイ・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 tnsname のエントリが必要です。
- 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・サイトにログを適切にアーカイブするための、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 ログが希望の場所に適用されるようにします。
- 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。

SQL セッションがアクティブなときに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 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 は接続中のセッションがない場合にのみ実行できます
```

処置：V$SESSION ビューを問い合せて、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 は動的パラメータなので、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した後、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 Intelligent Agent	オペレーティング・システム・プロンプトから agentctl stop コマンドを発行してください。

ロジカル・スタンバイ・データベースへの SQL 適用操作が停止した場合の処置

ログ適用サービスは、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 適用モードのロジカル・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 適用操作は停止します。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・データベースに対して SQL 文を再度適用してください。

表 A-2 一般的な SQL 適用操作エラーの解決方法

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 適用操作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 適用操作に失敗した場合は、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他の Oracle9*i* 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 適用操作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コマンド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 適用操作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 適用操作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) コールを発行した後、SQL 適用操作を再開してください。

関連項目： 障害の原因を判断するために DBA_LOGSTDBY_EVENTS ビューを問い合せる方法は、第 14 章を参照してください。

REDO ログ転送のネットワーク調整

REDO ログのアーカイブ・プロセスには、REDO ログからのバッファの読込みとアーカイブ・ログへの書込みが含まれます。宛先がリモートの場合は、Oracle Net サービスを使用するネットワークを介し、バッファはアーカイブ・ログへ書き込まれます。

デフォルトのアーカイブ・ログのバッファ・サイズは 1MB です。Oracle Net のデフォルトの転送バッファのサイズは 2KB です。このため、アーカイブ・ログ・バッファは転送のために、約 2KB のユニットに分割されます。これらのユニットは、基のネットワーク・インタフェースの最大転送ユニット（MTU）に応じてさらに分割されることがあります。

転送サイズを制御する Oracle Net のパラメータは、**セッション・データ・ユニット（SDU）**です。このパラメータは、転送されるネットワーク・パケット数を削減するために調整することができます。このパラメータは、512 バイト～ 32KB まで許されます。

最適なパフォーマンスを得るには、関連する SERVICE 宛先パラメータとして、Oracle Net SDU パラメータを 32KB に設定します。

次の例は、リモート宛先 netserv を定義するデータベース初期化パラメータ・ファイル・セグメントを示します。

```
LOG_ARCHIVE_DEST_3='SERVICE=netserv'
SERVICE_NAMES=srvc
```

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)(ORACLE_HOME=/oracle)))
```

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)))
```

遅延が少なく帯域幅の広いネットワー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、TCP の送受信ウィンドウ・サイズを増大させることでパフォーマンスを改善できます。ただし、これによりアーカイブと同じような特性を発揮できないなど、ネットワーク上のアプリケーションに悪影響を及ぼすことがあるため注意してください。この方法は、システム・リソースを大量に消費します。

関連項目：『Oracle9*i* Net Services 管理者ガイド』

また、カスケード・スタンバイ・データベースを使用して、プライマリ・データベースからスタンバイ・データベースにネットワーク処理をオフロードできます。詳細は、付録 D を参照してください。

Data Guard によるネットワーク・タイムアウトの管理

Oracle Data Guard ネットワーク接続を指定した場合、ネットワーク間の通信には 2 つのプロセスが存在します。ネットワーク接続が突然中断された場合、この 2 つのプロセスの対応は大きく異なります。次の説明は、ネットワーク接続が中断した場合に実際に発生する状態と、それが Data Guard 環境と構成に与える影響について述べています。この説明は、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・データベースの両方に適用されます。

Data Guard では、peer-to-peer 接続プロトコルを使用しています。このプロトコルによって、プライマリ・データベース・プロセスがログ・ライター（LGWR）またはアーカイバ（ARCH）の場合、スタンバイ・データベースへのネットワーク接続が確立されます。ネットワーク接続の要求があると、スタンバイ・サイトのリスナーは、スタンバイ・データベース上に個別のプロセスを作成します。このプロセスは、リモート・ファイル・サーバー（RFS）・プロセスと呼ばれます。この RFS プロセスでは、プライマリ・データベースのネットワーク・メッセージを使用します。つまり、ネットワークのメッセージを読み込んで要求を処理した後、確認メッセージをプライマリに返信します。

通常の Data Guard 操作で、REDO データがプライマリからスタンバイに転送される場合は、ネットワーク・メッセージがプライマリ・データベース（ネットワーク・クライアント）から開始され、常にスタンバイ・データベース（ネットワーク・サーバー）によって認識されます。この場合、LGWR プロセスと ARCH プロセスがネットワーク・クライアントで、RFS プロセスがネットワーク・サーバーです。

簡単な使用例として、プライマリ・システムとスタンバイ・システム間のネットワークが切断された場合を考えてみます。この結果、使用不能接続と呼ばれる状態が発生します。使用不能接続とは、物理的に接続が存在しない場合でも、各システム上のプロセスには接続が存在しているように見える状態を指します。

LGWR プロセスがこの使用不能接続を介して新しいメッセージを RFS プロセスに送信しようとすると、LGWR プロセスは、Oracle Net から TCP タイムアウト後に、接続が中断されたことを示すエラーを受け取ります。この方法で、LGWR はネットワーク接続が失われたことを確認した後、対処措置を取ることができます。Data Guard の属性 `[NO]MAX_FAILURE`、`[NO]REOPEN` および `[NO]NET_TIMEOUT` は、`LOG_ARCHIVE_DEST_`*n* パラメータのオプションです。これを使用すると、LGWR は応答していないネットワーク接続に関連付けられたタイムアウト間隔と再試行の回数を柔軟に制御できます。

LGWR プロセスとは対照的に、スタンバイ・データベースの RFS プロセスは、常に新しいメッセージがプライマリ・データベースから同期で着信するのを待機しています。ネットワークの読込み操作を実行する RFS プロセスは、データがその読込みバッファに着信するか、または基礎となるネットワーク・ソフトウェアが使用不能接続の無効化を決定するまでブロックされます。

Oracle Net では、定期的にネットワーク・プローブを送信して、クライアント / サーバー接続がアクティブであることを確認します。このため、クライアントの異常終了が原因で接続が無制限にオープン状態のままになることはありません。プローブは、使用不能接続または使用されなくなった接続を検出すると、エラーを戻し、RFS プロセスを終了させます。

Oracle Net のパラメータ SQLNET.EXPIRE_TIME を使用すると、ネットワーク・セッションがアクティブであることを確認するためにプローブを送信するタイム間隔を秒数で指定できます。このパラメータを小さい値に設定すると、使用不能接続が適時に検出される可能性が高くなります。このプローブ信号に応答しない接続は、切断されています。このパラメータは、将来スイッチオーバーする場合に備えて、プライマリ・データベースのみでなく、スタンバイ・データベースにも設定してください。

使用不能接続の検出機能を使用する場合の制限事項は、次のとおりです。

- プローブ・パケットは、小規模であっても、追加の通信量が発生します。ただし、プライマリ・データベースのワークロードをベースにする Data Guard によって発生するネットワーク通信量と比較すると、この追加パケットの通信量はごく少量です。
- 使用しているオペレーティング・システムによっては、サーバーで追加の処理を実行して、接続プローブ・イベントと、発生したその他のイベントを区別する必要があります。この処理は、ネットワークのパフォーマンスに影響を与える場合があります。

RFS プロセスは、使用不能ネットワーク接続の通知を受信すると、プロセスを終了します。ただし、RFS プロセスが終了するまでの時間、スタンバイ・サイトの ARCHIVELOG に関する情報、つまりプライマリ・データベースから受信中であった REDO 情報を含むスタンバイ REDO ログは、ロックされた状態のままです。この間、新しい RFS プロセスはすべて、同じアーカイブ REDO ログ（またはスタンバイ REDO ログ）の REDO 情報をプライマリ・データベースから受信できません。

使用不能ネットワーク接続の検出タイマーの期限切れの値は、TCP/IP keepalive パラメータでも制御できます。このパラメータを使用して、ネットワーク接続が有効であることを確認するまでの待機時間を秒数で指定します。ほとんどのシステムで、TCP/IP keepalive パラメータのデフォルト値は 2 時間に設定されています。これは、デフォルトでは、RFS プロセスが使用不能ネットワーク接続上でタイムアウトになるまで 2 時間も待機することを意味します。

したがって、オラクル社では、Oracle Net の SQLNET.EXPIRE_TIME パラメータと TCP/IP keepalive パラメータを 60 秒に設定することをお薦めします。この値は、ほとんどのシステムにとって適正な値です。パラメータを小さい値に設定しても本番システムに重大な影響を与えることはありません。

ネットワーク問題が解決し、プライマリ・データベースのプロセスがスタンバイ・データベースへのネットワーク接続を再度確立できるようになると、新しいネットワーク接続ごとに、新しい RFS プロセスがスタンバイ・データベース上で自動的に起動されます。これらの新しい RFS プロセスは、プライマリ・データベースからの REDO データの受信を再開します。

B

手動リカバリ

オラクル社では、フィジカル・スタンバイ・データベースに管理リカバリ・モードの使用をお薦めしていますが、**手動リカバリ・モード**の使用も可能です。手動リカバリ・モードの利点は次のとおりです。

- 手動リカバリ・モードでは、11 以上のスタンバイ・データベースを持つことができます。管理リカバリ操作で使用できるスタンバイ・データベースの数は 10 が限度です。
- プライマリおよびスタンバイ・データベースを Oracle Net を介さずに接続する場合、データベースを手動リカバリ・モードでオープンしてください。
- 管理リカバリ操作の実行中、なんらかの理由でその操作を使用できなくなった場合は、手動リカバリ・モードに切り替えることができます。

この付録では、手動リカバリ・モードでの操作方法を説明します。次の項目で構成されています。

手動リカバリ用スタンバイ・データベースの準備：基本タスク

表 B-1 に、手動リカバリ用のスタンバイ・データベースを設定する際の基本タスクをまとめます。この手順は、Oracle Net を介してスタンバイ・データベースに接続することを前提としています。Oracle Net をスタンバイ・データベースに接続しない場合は、手順 4 および手順 5 をスキップしてください。

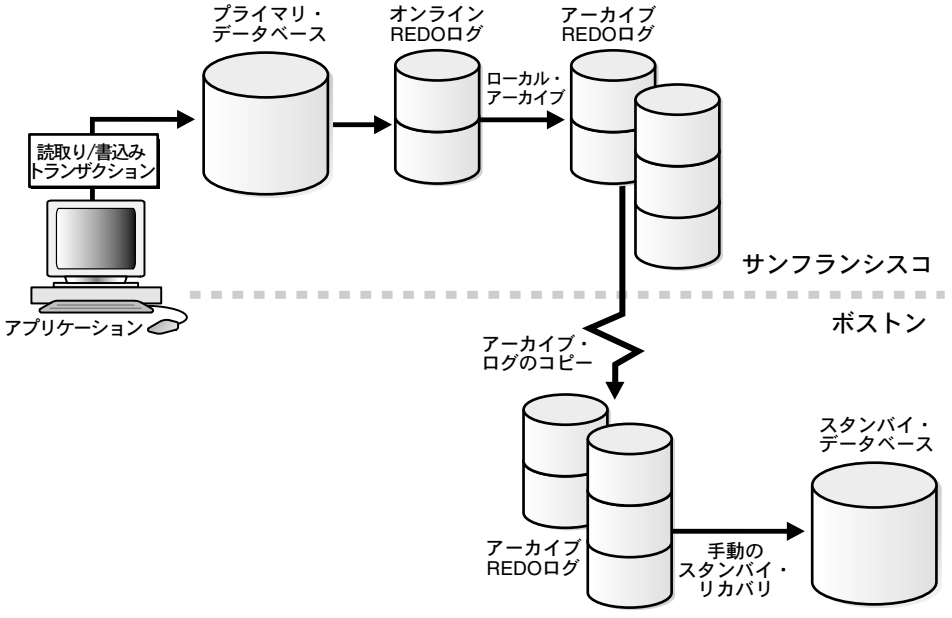
表 B-1 タスク・リスト：手動リカバリの準備

手順	タスク	手順の詳細
1	プライマリ・データベースのデータ・ファイルのバックアップを新しく作成するか、または旧バックアップを更新する。	3-4 ページ「プライマリ・データベースのコピーの作成」
2	プライマリ・データベースへ接続し、スタンバイ制御ファイルを作成する。	3-5 ページ「スタンバイ・データベース用の制御ファイルの作成」
3	準備したバックアップ・データ・ファイルとスタンバイ制御ファイルを、プライマリ・サイトからスタンバイ・サイトへコピーする。	3-5 ページ「スタンバイ・データベースにコピーする初期化パラメータ・ファイルの準備」および 3-5 ページ「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」
4	スタンバイ・データベースに Oracle Net 接続を作成する場合は、サービス名を作成する。	6-12 ページ「アーカイブ・ギャップの管理」および第 12 章の「LOCATION および SERVICE」
5	スタンバイ・データベースに Oracle Net 接続を作成する場合は、スタンバイ・サイトにリスナーを構築し、スタンバイ・インスタンスへの接続要求を受信できるようにする。	6-12 ページ「アーカイブ・ギャップの管理」
6	スタンバイ初期化パラメータ・ファイルをスタンバイ・サイトで作成し、スタンバイ・データベースの初期化パラメータを設定する。必要に応じて DB_FILE_NAME_CONVERT と LOG_FILE_NAME_CONVERT を設定し、スタンバイ制御ファイル内のプライマリ・ファイルを自動改名する。	5-27 ページ「ログ転送サービスの管理」
7	スタンバイ・インスタンスを起動し、スタンバイ・データベースをマウントする。	6-4 ページ「フィジカル・スタンバイ・インスタンスの起動」
8	手順 6 で DB_FILE_NAME_CONVERT と LOG_FILE_NAME_CONVERT によって自動的に改名されなかった全ファイルのスタンバイ制御ファイル内のプライマリ・データ・ファイルと REDO ログを手動で改名する。	B-13 ページ「スタンバイ・データベース・ファイルの手動による改名」

スタンバイ・データベースの手動リカバリ・モードへの設定

スタンバイ・データベースは、起動してマウントすると、手動リカバリ・モードにできます。スタンバイ・データベースを現行のものとしておくには、アーカイブ REDO ログをプライマリ・データベースからスタンバイ・データベースに手動で適用する必要があります。図 B-1 に、手動リカバリ・モードのデータベースを示します。

図 B-1 手動リカバリ・モードのスタンバイ・データベース



この項は、次の項目で構成されています。

- 手動リカバリ・モードの初期化
- 手動リカバリが必要な場合

手動リカバリ・モードの初期化

アーカイブ REDO ログは、次のいずれかの方法でスタンバイ・サイトに到着します。

- プライマリ・データベースは自動的にログをアーカイブします（ただし、Data Guard 環境をインストールしているときのみ）。
- オペレーティング・システムのユーティリティなどを利用して、手動でログを転送します。

スタンバイ・データベースは、アーカイブ・ログ・グループがスタンバイ初期化パラメータ・ファイル内の次のパラメータで指定した場所にあると考えます。

- LOG_ARCHIVE_DEST_*n* で指定した最初の有効なディスクの場所（*n* は 1 ～ 10 までの整数）。
- LOG_ARCHIVE_DEST_*n* で指定した場所

アーカイブ・ログが初期化パラメータ・ファイルで指定した場所にない場合は、RECOVER 文の FROM オプションを使用して別の場所を指定できます。

スタンバイ・データベースを手動リカバリ・モードにする方法

1. SQL*Plus を使用してスタンバイ・インスタンスに接続し、次にスタンバイ・データベースで Oracle インスタンスを起動します。たとえば、次のように入力します。

   ```
   STARTUP NOMOUNT pfile=initSTANDBY.ora
   ```

2. 次のように入力してスタンバイ・データベースをマウントします。

   ```
   ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

3. ログ転送サービスが、ログをスタンバイ・サイトに自動的にアーカイブしない場合は、オペレーティング・システムの適切なユーティリティで、スタンバイ・サイト上の希望の場所にログをコピーしてバイナリ・データを転送します。たとえば、次のように入力します。

   ```
   % cp /oracle/arc_dest/*.arc /standby/arc_dest
   ```

4. RECOVER 文を発行し、スタンバイ・データベースを手動リカバリ・モードにします。

注意： FROM 'location' オプションは、アーカイブ・ログ・グループがスタンバイ初期化パラメータ・ファイル内の LOG_ARCHIVE_DEST_*n* パラメータ（*n* は 1 ～ 10 までの整数）または LOG_ARCHIVE_DEST_*n* パラメータで指定した場所にない場合にのみ指定します。

たとえば、次の文のいずれかを実行します。

```
RECOVER STANDBY DATABASE # uses location for logs specified in
                         # initialization parameter file
RECOVER FROM '/logs' STANDBY DATABASE # specifies nondefault location
```

Oracle データベース・サーバーがアーカイブ REDO ログを生成しているときは、それらを継続的にコピーしてスタンバイ・データベースに適用し、ログを常にカレントの状態にする必要があります。

手動リカバリが必要な場合

手動リカバリ・モードは Data Guard 以外の環境で必要になります。Data Guard 以外の環境とは、以下のいずれかを手動で行う場合を指します。

- アーカイブ REDO ログをプライマリ・サイトからスタンバイ・サイトに転送します。
- スタンバイ・データベースにアーカイブ REDO ログを適用します。

Data Guard 環境を使用している場合も、ときによりスタンバイ・データベースで手動リカバリを実行できます。たとえば、手動リカバリ・モードを使用すると既存のアーカイブ・ギャップを手動で解決できます。

手動によるアーカイブ・ギャップの解決

アーカイブ・ギャップとは、アーカイブ REDO ログの特定の範囲を指します。この範囲は、プライマリ・データベースで生成される次のアーカイブ REDO ログをスタンバイ・データベースに適用できないときに発生します。この項は、次の項目で構成されています。

- アーカイブ・ギャップの発生原因
- アーカイブ・ギャップが存在するかどうかの判断
- スタンバイ・サイトへのアーカイブ・ギャップ・ログの手動による転送
- スタンバイ・データベースへのアーカイブ・ギャップ・ログの手動による適用

注意： 通常、アーカイブ・ギャップは手動操作を行うことなく自動的に解決されます。ログ適用サービスが REDO ログ内のギャップを自動的にリカバリする方法の詳細は、6-12 ページの「アーカイブ・ギャップの管理」を参照してください。

アーカイブ・ギャップの発生原因

アーカイブ・ギャップは、プライマリ・データベースがログをアーカイブしたにもかかわらず、スタンバイ・サイトにはアーカイブされなかった場合に発生する可能性があります。スタンバイ・データベースには REDO ログをログ順序番号順に適用する必要があるため、メディア・リカバリは、最初の欠落ログに遭遇すると停止します。

アーカイブ・ギャップは次の状況で発生することがあります。

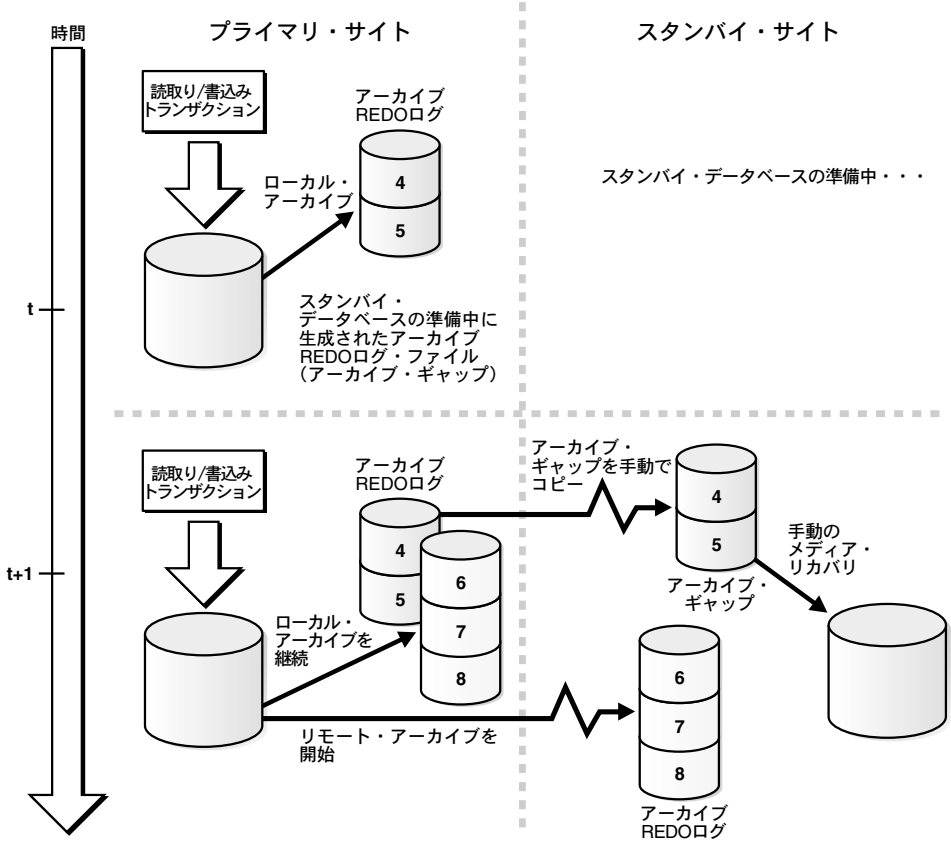
- スタンバイ・データベースの作成
- プライマリ・データベースがオープンしている間のスタンバイ・データベースの停止
- スタンバイ・サイトへのログのアーカイブを妨げるネットワーク障害

スタンバイ・データベースの作成

アーカイブ・ギャップは、スタンバイ・データベースを古いバックアップから作成すると発生します。たとえば、スタンバイ・データベースが、ログ 100 の変更を含むバックアップから作成されているとします。このときプライマリ・データベースに、現在ログ 150 の変更が含まれているとすると、スタンバイ・データベースは、ログ 101 からログ 150 までの適用を要求します。オープン・データベースのホット・バックアップからスタンバイ・データベースを生成する場合も、通常アーカイブ・ギャップが発生します。

たとえば、図 B-2 の使用例を想定します。

図 B-2 アーカイブ・ギャップ内のアーカイブ・ログの手動リカバリ



次の手順を実行します。

1. プライマリ・データベースのホット・バックアップを取得します。
2. ネットワーク・ファイルを構築している時間 *t* で、プライマリはログ順序の 4 および 5 をアーカイブします。
3. 時間 *t* + *1* で、スタンバイ・インスタンスを起動します。
4. プライマリは、ログ順序 6、7 および 8 をプライマリ・サイトとスタンバイ・サイトの両方にアーカイブします。

アーカイブ・ログの順序 4 および 5 はこれでアーカイブ・ギャップの一部となり、これらのログはスタンバイ・データベースに適用する必要があります。

プライマリ・データベースがオープンしている間のスタンバイ・データベースの停止

メンテナンス問題を解決するため、スタンバイ・データベースの停止が必要になることがあります。たとえば、プライマリ・データベース内で MAXDATAFILE のような制御ファイル・パラメータを変更するときには、スタンバイ・データベースを停止する必要があります。

アーカイブ・ギャップの発生を避けるために、次のルールを実行します。

- プライマリ・データベースを起動する前にスタンバイ・データベースを起動します。
- スタンバイ・データベースを停止する前にプライマリ・データベースを停止します。

これら 2 つのルールに違反すると、プライマリ・データベースがオープンし、アーカイブを実行しているときにスタンバイ・データベースが停止します。結果として、アーカイブ・ギャップが発生します。

注意： スタンバイ・サイトが、プライマリ初期化パラメータ・ファイルの LOG_ARCHIVE_DEST_*n* パラメータの 1 つにおいて MANDATORY と指定されている場合は、スタンバイ・データベースを停止する前に、それを動的に OPTIONAL に変更します。これを行わないと、プライマリ・データベースはオンライン REDO ログをアーカイブできないため停止します。

スタンバイ・サイトへのログのアーカイブを妨げるネットワーク障害

Data Guard 環境をメンテナンスしているときにネットワークが停止すると、プライマリ・データベースはディスクへのアーカイブを継続しますが、スタンバイ・サイトへのアーカイブは実行できません。この状況では、アーカイブされたログは通常どおり、プライマリ・サイトに蓄積されますが、スタンバイ・データベースはこれらのログを認識できません。

この問題を回避するには、スタンバイ・データベースの宛先を必須として指定します。アーカイブ先が必須の場合、プライマリ・データベースはログがスタンバイ・サイトにアーカイブできるようになるまで、ログのアーカイブを実行しません。たとえば、プライマリ初期化パラメータ・ファイルに次を設定して standby1 を必須のアーカイブ先にできます。

```
LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY'
```

この構成を行った場合、ネットワークの問題が解決されていないとプライマリ・データベースが停止します。これは、アーカイブされていないオンライン REDO ログには、プライマリ・データベースが切り替わらないためです。プライマリ・データベースにオンライン REDO ログが 2 つしかない場合、問題はさらに困難になります。

関連項目：

- スタンバイ・アーカイブにおける OPTIONAL 属性および MANDATORY 属性の重要性についての詳細は、5-15 ページの「必須およびオプションの宛先の指定」を参照してください。
- 関連する使用例については、10-21 ページの「ネットワーク障害のリカバリ」を参照してください。

アーカイブ・ギャップが存在するかどうかの判断

アーカイブ・ギャップが存在するかどうかを判断するには、V$ARCHIVED_LOG ビューと V$LOG ビューを問い合せます。アーカイブ・ギャップが存在する場合、問合せの出力により、アーカイブ・ギャップ内のすべてのログのスレッド番号および順序番号が特定されます。スレッドにアーカイブ・ギャップがない場合、問合せによって行は戻りません。

アーカイブ・ギャップ内のログの識別方法

スタンバイ・データベースの V$ARCHIVED_LOG ビューと V$LOG ビューを問い合せます。たとえば、次の問合せは、DEST_ID=2 で指定した宛先の RECD と SENT の順序番号に相違がある、つまりギャップがあることを示しています。

```
SQL> SELECT MAX(R.SEQUENCE#) LAST_SEQ_RECD, MAX(L.SEQUENCE#) LAST_SEQ_SENT FROM
  2> V$ARCHIVED_LOG R, V$LOG L WHERE
  3> R.DEST_ID=2 AND L.ARCHIVED='YES';

LAST_SEQ_RECD LAST_SEQ_SENT
------------- -------------
            7            10
```

次の問合せを使用して、ローカル・システム上にあるアーカイブ REDO ログの名前を判断します。このログはギャップが発生したスタンバイ・システムにコピーする必要があります。

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE THREAD#=1 AND DEST_ID=1 AND
  2> 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
/primary/thread1_dest/arcr_1_10.arc
```

スタンバイ・サイトへのアーカイブ・ギャップ・ログの手動による転送

アーカイブ・ギャップ内のログ順序番号を取得したら、プライマリ・サイトの V$ARCHIVED_LOG ビューを問い合せて、ファイル名を取得できます。スタンバイ・サイトのアーカイブ・ログ・ファイル名は、スタンバイ初期化パラメータ・ファイルの STANDBY_ARCHIVE_DEST パラメータおよび LOG_ARCHIVE_FORMAT パラメータによって生成されます。

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サイトにある場合、またはスタンバイ・データベースがプライマリ・データベースと異なるディレクトリ構造を持つリモート・サイトにある場合、スタンバイ・サイトのログのファイル名は、プライマリ・データベースでアーカイブされたログのファイル名と同じにはなりません。アーカイブ・ログをスタンバイ・サイトに転送する前に、スタンバイ・サイトでログの正しいファイル名を見極めてください。

スタンバイ・サイトにアーカイブ・ギャップ内のログをコピーする方法

1. 以前取得したアーカイブ・ギャップ・ログのリストを見直します。たとえば、次のアーカイブ・ギャップがあると想定します。

```
THREAD#    LOW_SEQUENCE#   HIGH_SEQUENCE#
---------- -------------   --------------
       1             460              463
       2             202              204
       3             100              100
```

   このビューに表示されているスレッドには、アーカイブ・ギャップが存在します。したがって、スレッド 1 ～ 3 のログをコピーする必要があります。

2. プライマリ・データベースによってアーカイブされたアーカイブ・ギャップのログのファイル名を判断します。プライマリ・データベースに接続した後、SQL 問合せを発行して各スレッドのログ名を取得します。たとえば、次の SQL 文を使用して、スレッド 1 のログのファイル名を取得します。

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE THREAD#=1 AND DEST_ID=1
  2> AND SEQUENCE# > 459 AND SEQUENCE# < 464;

NAME
--------------------------------------------------------------------------
/primary/thread1_dest/arcr_1_460.arc
/primary/thread1_dest/arcr_1_461.arc
/primary/thread1_dest/arcr_1_462.arc
/primary/thread1_dest/arcr_1_463.arc
4 rows selected
```

   スレッド 2 と 3 について、同様の問合せを実行します。

3. スタンバイ・サイトで、スタンバイ初期化パラメータ・ファイル内の STANDBY_ARCHIVE_DEST および LOG_ARCHIVE_FORMAT の設定を見直します。たとえば、次を発見したと想定します。

```
STANDBY_ARCHIVE_DEST = /standby/arc_dest/
LOG_ARCHIVE_FORMAT = log_%t_%s.arc
```

これらのパラメータの設定から、スタンバイ・サイトのアーカイブ REDO ログのファイル名を判断できます。

4. プライマリ・サイトで、プライマリ・サイトのアーカイブ・ギャップ・ログをスタンバイ・サイトにコピーします。この場合、ログの名前を STANDBY_ARCHIVE_DEST の値および LOG_ARCHIVE_FORMAT の値に応じて改名します。たとえば、次の copy コマンドを入力して、スレッド 1 に必要なアーカイブ・ギャップ・ログをコピーします。

```
% cp /primary/thread1_dest/arcr_1_460.arc /standby/arc_dest/log_1_460.arc
% cp /primary/thread1_dest/arcr_1_461.arc /standby/arc_dest/log_1_461.arc
% cp /primary/thread1_dest/arcr_1_462.arc /standby/arc_dest/log_1_462.arc
% cp /primary/thread1_dest/arcr_1_463.arc /standby/arc_dest/log_1_463.arc
```

同様の copy コマンドを使用して、スレッド 2 と 3 のアーカイブ・ギャップ・ログをコピーします。

5. スタンバイ・サイトで、LOG_ARCHIVE_DEST パラメータと STANDBY_ARCHIVE_DEST パラメータの値が異なる場合は、STANDBY_ARCHIVE_DEST ディレクトリのアーカイブ・ギャップ・ログを LOG_ARCHIVE_DEST ディレクトリにコピーします。これらのパラメータの値が同じ場合、この手順を実行する必要はありません。

たとえば、次のスタンバイ初期化パラメータが設定されているとします。

```
STANDBY_ARCHIVE_DEST = /standby/arc_dest/
LOG_ARCHIVE_DEST = /log_dest/
```

パラメータの値が異なるので、アーカイブ・ログを LOG_ARCHIVE_DEST の場所にコピーします。

```
% cp /standby/arc_dest/* /log_dest/
```

手動リカバリを開始すると、Oracle データベース・サーバーは、LOG_ARCHIVE_DEST 値を参照してログの位置を判断します。

これで、必要なログがすべて STANDBY_ARCHIVE_DEST ディレクトリにあるため、B-12 ページの「スタンバイ・データベースへのアーカイブ・ギャップ・ログの手動による適用」に進んで、アーカイブ・ギャップ・ログをスタンバイ・データベースに適用できます。

> **関連項目：** 6-18 ページの「V$ARCHIVED_LOG 固定ビューへのアクセス」および第 14 章の「V$ARCHIVED_LOG」

スタンバイ・データベースへのアーカイブ・ギャップ・ログの手動による適用

アーカイブ・ギャップ内のログをスタンバイ・サイトにコピーしたら、RECOVER AUTOMATIC 文を使用してそれらを適用できます。

アーカイブ・ギャップ内のアーカイブ REDO ログを適用する方法

1. スタンバイ・データベースを起動してマウントします（まだマウントされていない場合）。たとえば、次のように入力します。

```
SQL> STARTUP NOMOUNT PFILE=/oracle/admin/pfile/initSTBY.ora
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

2. AUTOMATIC オプションを使用してデータベースをリカバリします。

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

AUTOMATIC オプションは、リカバリ操作の継続に必要な次のアーカイブ REDO ログの名前を自動的に生成します。

利用可能なログをリカバリすると、Oracle データベース・サーバーは現在存在しないログの名前の入力を促します。これは、リカバリ・プロセスが、プライマリ・データベースによってスタンバイ・サイトにアーカイブされたログについて認識していないためです。たとえば、次のように表示されます。

```
ORA-00308: アーカイブ・ログ /oracle/standby/standby_logs/arcr_1_540.arc をオープンできません。
ORA-27037: ファイル・ステータスを取得できません。
SVR4 Error: 2: No such file or directory
Additional information: 3
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

3. Oracle データベース・サーバーが利用可能なログを適用したら、次の文を実行して（または CTRL+C キーを押下して）、リカバリを取り消します。

```
SQL> CANCEL
Media recovery cancelled.
```

リカバリ取消し後に出力される次のエラー・メッセージは許容できるもので、問題はありません。

```
ORA-01547: 警告 : RECOVER は成功しましたが OPEN RESETLOGS が次のエラーを受け取りました。
ORA-01194: ファイル 1 は一貫した状態にするためにさらにリカバリが必要です。
ORA-01110: データ・ファイル 1:'some_filename'
ORA-01112: メディア・リカバリが開始されていません
```

オラクル社は、ALTER DATABASE 文の RECOVER MANAGED STANDBY DATABASE 句を使用して、アーカイブ・ギャップ内のログを自動的に適用することをお薦めします。

> **関連項目：** 詳細は、6-12 ページの「アーカイブ・ギャップの管理」を参照してください。

スタンバイ・データベース・ファイルの手動による改名

プライマリ・データ・ファイルと REDO ログのすべてが、スタンバイ制御ファイル内で変換パラメータによって改名できないことがあります。たとえば、次のデータ・ファイルがデータベースに存在するとき、それらを次の表のように改名するとします。

プライマリ・ファイル名	スタンバイ・ファイル名
/oracle/dbs/df1.dbf	/standby/df1.dbf
/oracle/dbs/df2.dbf	/standby/df2.dbf
/data/df3.dbf	/standby/df3.dbf

次のように DB_FILE_NAME_CONVERT を設定して、最初の 2 つのデータ・ファイル用にファイル名を変換できます。

```
DB_FILE_NAME_CONVERT = '/oracle/dbs', '/standby'
```

これを行っても、このパラメータは /data/df3.dbf の改名を獲得しません。この場合、次のように SQL 文を発行し、スタンバイ・データベース制御ファイル内でデータ・ファイルを手動で改名する必要があります。

```
SQL> ALTER DATABASE RENAME FILE '/data/df3.dbf' to '/standby/df3.dbf';
```

データ・ファイルを手動で改名する方法

1. スタンバイ・データベースを起動し、マウントし（まだ起動されていない場合）、次にデータベースをマウントします。

   ```
   SQL> STARTUP NOMOUNT PFILE=initSTANDBY1.ora;
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

2. 改名を必要とする個々のデータ・ファイルに ALTER DATABASE 文を発行します。ここで *old_name* は制御ファイル内に記録されているデータ・ファイルの古い名前で、*new_name* はスタンバイ制御ファイル内に記録されるデータ・ファイルの新しい名前です。

   ```
   SQL> ALTER DATABASE RENAME FILE 'old_name' TO 'new_name';
   ```

DB_FILE_NAME_CONVERT パラメータによって獲得されないデータ・ファイルのすべてを手動で改名すると、スタンバイ・データベース制御ファイルは、リカバリ・プロセス中にログを正しく解釈します。

C

Real Application Clusters でのスタンバイ・データベースのサポート

Oracle9*i* は、プライマリおよびスタンバイのデータベースのいずれかまたは両方が Real Application Clusters 環境に存在するときに、プライマリ・データベースからスタンバイ・データベースへのデータベース・アーカイブを実行する機能を提供します。この章では、Oracle Data Guard を Oracle Real Application Clusters 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。

- Real Application Clusters 環境でのスタンバイ・データベースの構成
- Real Application Clusters 環境での構成に関する考慮事項
- トラブルシューティング

Real Application Clusters 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、Real Application Clusters 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。

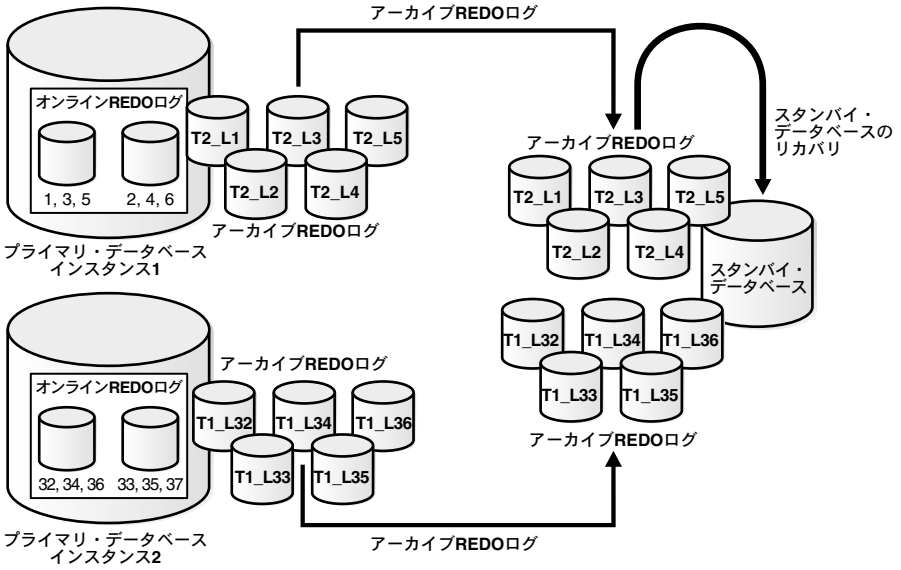
インスタンスの組合せ	単一インスタンス・スタンバイ・データベース	複数インスタンス・スタンバイ・データベース
単一インスタンス・プライマリ・データベース	可	可（読取り専用問合せの場合）
複数インスタンス・プライマリ・データベース	可	可

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、独自のオンライン REDO ログをスタンバイ・データベースへアーカイブします。

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 C-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 Real Application Clusters 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 ログをアーカイブしています。

図 C-1 複数インスタンス・プライマリ・データベースからの REDO ログのアーカイブ



この場合は、プライマリ・データベースのインスタンス 1 がログ 1、2、3、4、5 を転送するのに対し、インスタンス 2 がログ 32、33、34、35、36 を転送します。スタンバイ・データベースが管理リカバリ・モードになっている場合は、適用するアーカイブ REDO ログの正しい順序を自動的に判断します。

Real Application Clusters 環境でプライマリ・データベースを設定する手順

プライマリ・データベースでログ転送サービスを設定するには、次の手順を実行します。

1. すべてのインスタンスで ARCH または LGWR を、アーカイブ操作を実行するプロセスとして指定します。
2. スタンバイ・データベースを受信ノードとして指定します。これは、`LOG_ARCHIVE_DEST_n` 初期化パラメータの `SERVICE` 属性を使用して実行します。

また、スタンバイ・データベースは、管理リカバリによって受信するアーカイブ REDO ログを適用し、常にプライマリ・データベースと同じカレントの状態を保持します。

> **関連項目：** Real Application Clusters 用にデータベースを構成する方法については、『Oracle9*i* Real Application Clusters インストレーションおよび構成』を参照してください。

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

単一インスタンス・スタンバイ・データベースでログ転送サービスを設定するには、次の手順を実行します。

1. ログ転送サービスで LGWR プロセスが使用されている場合は、スタンバイ REDO ログを作成します。
2. LGWR プロセスが使用されている場合は、アーカイブ・ログ宛先をローカルでアーカイブするように定義します。これは、LOG_ARCHIVE_DEST_1 初期化パラメータの LOCATION 属性を使用して実行します。ログ転送サービスで ARCH プロセスが使用されている場合は、STANDBY_ARCHIVE_DEST および LOG_ARCHIVE_FORMAT を定義して、アーカイブ REDO ログの場所を指定します。
3. スタンバイ・データベースで MRP を開始します。

複数インスタンス・プライマリ・データベースと複数インスタンス・スタンバイ・データベースの設定

次の例は、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 Real Application Clusters 環境に存在する場合の構成を示しています。これによって、スタンバイ・データベースでログ転送サービスの処理とログ適用サービスの処理を分離できるため、プライマリとスタンバイの両方で、データベース全体のパフォーマンスが向上します。図 C-2 は、Real Application Clusters 環境でのスタンバイ・データベースの構成を示しています。

図 C-2 Real Application Clusters のスタンバイ・データベース

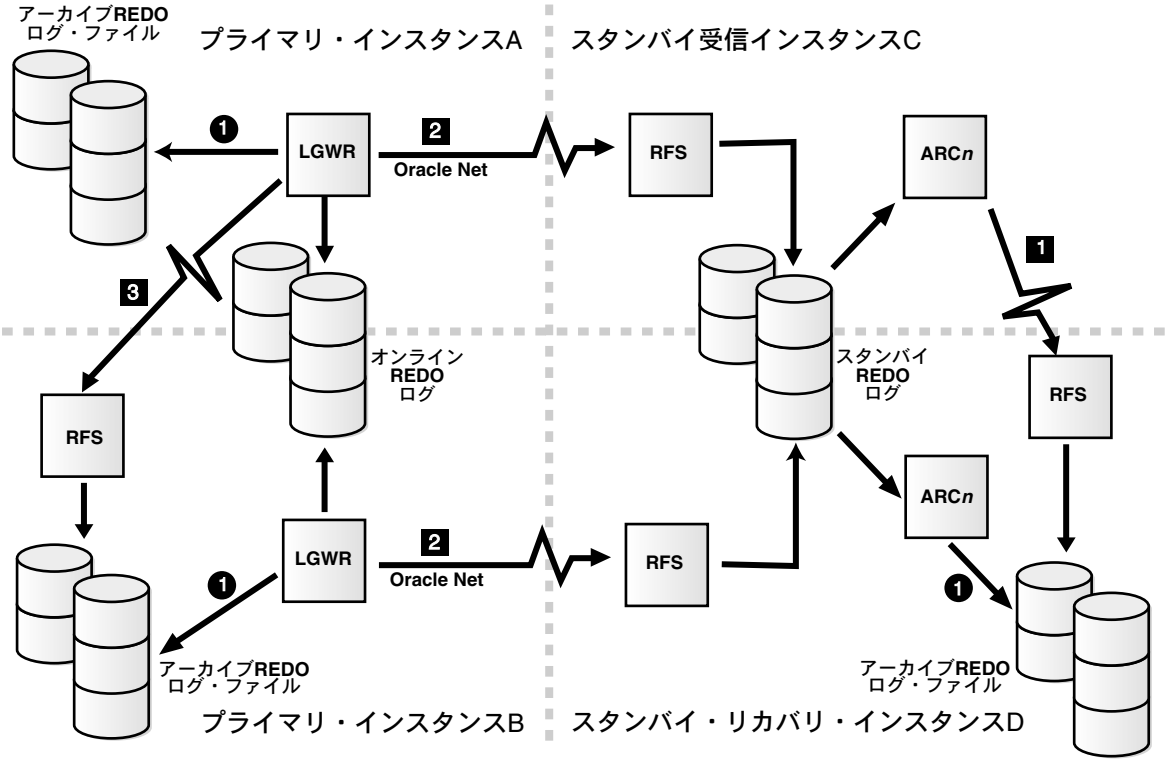


図 C-2 で、丸で囲まれた数字はローカル接続を示し、四角で囲まれた数字はリモート接続を示します。

Real Application Clusters 環境でスタンバイ・データベースを使用するときは、どのインスタンスも、アーカイブ・ログをプライマリ・データベースから受信できます。これは**受信インスタンス**と呼ばれます。しかし、アーカイブ・ログは、最終的には、管理リカバリ・プロセスが実行されるノードからアクセスが可能なディスク・デバイスに存在する必要があります。これは**リカバリ・インスタンス**と呼ばれます。スタンバイ・データベースのアーカイブ・ログの、受信インスタンスからリカバリ・インスタンスへの転送は、スタンバイ・データベースで実行されるインスタンス間アーカイブ操作を使用して実現されます。

スタンバイ・データベースのインスタンス間アーカイブ操作のためには、スタンバイ REDO ログを、プライマリ・データベースのアーカイブ・ログの一時リポジトリとして使用する必要があります。スタンバイ REDO ログを使用することにより、スタンバイ・データベースのパフォーマンスと信頼性が向上するだけでなく、インスタンス間アーカイブ操作の実行が可能になります。ただし、インスタンス間アーカイブ操作を行うためにはスタンバイ REDO

ログが必要なため、プライマリ・データベースは、ログ・ライター・プロセス（LGWR）を使用してプライマリ・データベースのアーカイブ操作を実行する必要があります。

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が Real Application Clusters 構成で、スタンバイ・データベースが管理リカバリ・モードである場合、スタンバイ・データベースの単一のインスタンスが、プライマリ・インスタンスによって転送されるログのすべてのセットを適用します。この場合、REDO を適用しないスタンバイ・インスタンスを、管理リカバリの進行時に読取り専用モードにすることはできません。ほとんどの場合、リカバリ不能インスタンスもマウントできますが、停止している必要があります。

Real Application Clusters 環境でスタンバイ・データベースを設定する手順

スタンバイ・データベースでログ転送サービスを設定するには、次の手順を実行します。

1. スタンバイ REDO ログを作成します。Real Application Clusters 環境では、スタンバイ REDO ログは、RAW デバイスなど、すべてのインスタンスに共有されるディスク・デバイスに存在する必要があります。
2. 管理リカバリ・プロセス（MRP）が稼働するリカバリ・インスタンスで、アーカイブ・ログ宛先をローカルでアーカイブするように定義します。これは、インスタンス間アーカイブが不要なためです。これは、`LOG_ARCHIVE_DEST_1` 初期化パラメータの `LOCATION` 属性を使用して実行します。
3. 受信インスタンスで、アーカイブ・ログ宛先を MRP が稼働するノードにアーカイブするように定義します。これは、`LOG_ARCHIVE_DEST_1` 初期化パラメータの `SERVICE` 属性を使用して実行します。
4. すべてのスタンバイ・データベース・インスタンスで ARC*n* プロセスを開始します。
5. リカバリ・インスタンスで MRP を開始します。

Real Application Clusters 環境でプライマリ・データベースを設定する手順

プライマリ・データベースでログ転送サービスを設定するには、次の手順を実行します。

1. すべてのインスタンスで LGWR を、アーカイブ操作を実行するプロセスとして指定します。
2. スタンバイ・データベースを受信ノードとして指定します。これは、`LOG_ARCHIVE_DEST_`*n* 初期化パラメータの `SERVICE` 属性を使用して実行します。

各プライマリ・データベース・インスタンスが、対応するスタンバイ・データベース・インスタンスにアーカイブできれば理想的です。ただし、これは必須ではありません。

インスタンス間アーカイブ・データベース環境の設定

インスタンス間アーカイブ・データベース環境を設定できます。Real Application Clusters 構成内では、各インスタンスがそのアーカイブ REDO ログをクラスタの単一インスタンスにダイレクトするように設定します。このインスタンスは、**リカバリ・インスタンス**と呼ばれ、一般的には管理リカバリが実行されるインスタンスです。このインスタンスには一般的に、Recovery Manager によりバックアップとリストア・サポートに使用できるテープ・ドライブがあります。例 C-1 に、複数インスタンスにわたって REDO ログをアーカイブするための LOG_ARCHIVE_DEST_*n* 初期化パラメータを設定する方法を示します。リカバリ・インスタンスを除くすべてのインスタンスでこの例を実行してください。

例 C-1 インスタンス間アーカイブの宛先の設定

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = 'LOCATION=archivelog MANDATORY REOPEN=120';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = enable;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=prmy1 MANDATORY REOPEN=300';
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = enable;
```

宛先 1 は、インスタンス・リカバリに必要なローカル・アーカイブ REDO ログが格納されているリポジトリです。これは必須の宛先です。失敗する原因としてディスク領域不足が予測されるので、再試行間隔は 2 分間です。これは、DBA が不要なアーカイブ REDO ログをパージするのに十分な時間です。宛先の障害の通知は、プライマリ・データベースのアラート・ログを手動で検索することによって実行されます。

宛先 2 は、ローカル・ディスク記憶域からテープへ、アーカイブ REDO ログをバックアップするために Recovery Manager が使用される、プライマリ・データベース上のリカバリ・インスタンスです。これは必須の宛先で、再接続のしきい値は 5 分間です。これは、ネットワーク関連の障害を修正するのに必要な時間です。宛先の障害の通知は、プライマリまたはスタンバイ・データベースのアラート・ログを手動で検索することによって実行されます。

インスタンス間アーカイブが使用できるのは、ARC*n* プロセスを使用したときのみです。インスタンス間アーカイブに LGWR プロセスを使用すると、RFS プロセスで障害が発生し、アーカイブ・ログの宛先がエラー状態になります。

Real Application Clusters 環境での構成に関する考慮事項

この項では、Real Application Clusters 環境に固有の Data Guard 構成情報を提供します。次の項目で構成されています。

- アーカイブ・ログ・ファイルの形式
- アーカイブ先の割当て制限
- データ保護モード
- ロールの推移

アーカイブ・ログ・ファイルの形式

アーカイブ・ログ・ファイル名の形式は通常、log_% パラメータの形式になります。% パラメータは次のいずれかです。

パラメータ	説明
%T	スレッド番号、左 0（ゼロ）埋込み
%t	スレッド番号、埋込みなし
%S	ログ順序番号、左 0（ゼロ）埋込み
%s	ログ順序番号、埋込みなし

たとえば、`LOG_ARCHIVE_FORMAT = "log_%t_%s.arc"` のようになります。Real Application Clusters が、`LOG_ARCHIVE_FORMAT` パラメータを持つアーカイブ REDO ログ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。

> **関連項目：** アーカイブ REDO ログの格納場所に関する詳細は、第 5 章「ログ転送サービス」を参照してください。

アーカイブ先の割当て制限

LOG_ARCHIVE_DEST_n 初期化パラメータの QUOTA_SIZE 属性を使用すると、アーカイブ先で使用できるディスク・デバイスの物理記憶域の量を指定できます。宛先に割り当てられた物理ディスクの全部または一部を占有できるようにアーカイブ先を指定できます。たとえば、Real Application Clusters 環境では、物理的なアーカイブ REDO ログ・ディスク・デバイスを 2 つ以上の別々のノードで共有できます。ただし、Sun Clusters で使用可能なクラスタ化ファイル・システムなどを介します。インスタンス間では、初期化パラメータの情報が共有されていないため、Real Application Clusters ノードは、アーカイブ REDO ログの物理ディスク・デバイスが他のインスタンスと共有されていることを認識していません。このため、宛先ディスク・デバイスがいっぱいになったときには重大な問題が発生します。すなわち、すべてのインスタンスがすでにいっぱいになったデバイスへのアーカイブを実行するまで、エラーは検出されません。これはデータベースの可用性に重大な影響を及ぼします。

データ保護モード

Real Application Clusters 構成では、ノードがスタンバイ宛先との接続を失うと、クラスタの他のすべてのメンバーがその宛先へのデータ送信を停止します（これによって、宛先に転送されたデータの整合性が維持され、リカバリが可能になります）。

障害のあったスタンバイ宛先が復旧すると、Data Guard は、プライマリ・データベースとスタンバイ・データベースが同じ（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、Real Application Clusters 環境での 3 つのデータ保護構成の影響について説明します。

- 最大保護の構成

  接続を失った宛先が最後のスタンバイ・サイトの場合、接続を失ったノード上のインスタンスは停止します。Real Application Clusters 構成内で最後のスタンバイ・サイトへの接続を維持しているノードは、接続を失ったインスタンスをリカバリし、スタンバイ・サイトへの送信を継続します。Real Application Clusters 構成内のすべてのノードが最後のスタンバイ・サイトへの接続を失った場合のみ、プライマリ・データベースも含めて構成全体が停止します。

注意： Real Application Clusters と Data Guard を最大保護モードで実行中に、ネットワークの停止時間が長い場合は、ネットワーク接続がリストアされるまで、プライマリ・データベースを最大可用性モードまたは最大パフォーマンス・モードのいずれかで実行するように変更することを考慮してください。プライマリ・データベースを一時的に最大可用性モードまたは最大パフォーマンス・モードに変更して実行する方法は、C-11 ページの「ネットワーク停止時に Real Application Clusters の停止時間を回避する」を参照してください。

最大保護の構成に含まれるサイトに対してフェイルオーバー操作が発生した場合、プライマリ・データベースでそれまでにコミットされたすべてのデータは、スタンバイ・サイトでリカバリされます。

- 最大可用性の構成

  最後のスタンバイ宛先との接続を失っても、プライマリ・データベースのインスタンスは停止しません。

  最大可用性の構成に含まれるサイトに対してフェイルオーバー操作が発生した場合、プライマリ・データベースでそれまでにコミットされ、スタンバイ・データベースに正常に送信されたすべてのデータは、スタンバイ・サイトでリカバリされます。

- 最大パフォーマンスの構成

  最後のスタンバイ宛先との接続を失っても、プライマリ・データベースのインスタンスは停止しません。

  スタンバイ・サイトに対してフェイルオーバー操作が発生した場合、プライマリ・データベースから受信したデータは、最後のトランザクションと一貫性のある状態になるまでスタンバイ・データベースでリカバリされます。これは、単一インスタンス構成の場合に、すべての受信データがリカバリされることを意味します。フェイルオーバーに際しては、転送前の 1 つ以上のログでトランザクションが失われる可能性があります。

ロールの推移

スイッチオーバー操作

Real Application Clusters データベースの場合は、1 つのプライマリ・インスタンスと 1 つのスタンバイ・インスタンスのみをスイッチオーバー操作時にアクティブにできます。したがって、スイッチオーバー操作の前に、1 つのプライマリ・データベースと 1 つのスタンバイ・データベース以外はすべて停止します。スイッチオーバー操作の完了後、スイッチオーバー操作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。

フェイルオーバー操作

Real Application Clusters 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、最初に 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。フェイルオーバー操作の完了後、停止したインスタンスを再起動します。

SQL 文の `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH SKIP STANDBY LOGFILE` を発行して、スタンバイ・データベースを強制的にプライマリ・ロールに切り替える場合、ログ適用サービスは、最初のアーカイブされていない REDO ログを検出するまでアーカイブ REDO ログを適用します。これ以降のすべてのアーカイブ REDO ログはリカバリされず、ログに含まれているすべてのデータは消失します。Real Application Clusters 環境では、`FINISH SKIP STANDBY LOGFILE` 句を使用すると、データの消失が増加する可能性があります。これは、複数のインスタンスが REDO ログに依存しているためです。

トラブルシューティング

この項は、Real Application Clusters 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ご利用いただけます。次の項目で構成されています。

Real Application Clusters 構成でスイッチオーバーできない

データベースが Real Application Clusters 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: マウントは別のインスタンスによるマウントと矛盾します。
```

処置：次のように GV$INSTANCE ビューを問い合せて、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/CHANGE_ON_INSTALL@standby2 AS SYSDBA
SQL> SHUTDOWN;
SQL> EXIT
```

ネットワーク停止時に Real Application Clusters の停止時間を回避する

Real Application Clusters 環境でプライマリ・データベースをサポートするように Data Guard を構成し、プライマリ・データベースが最大保護モードで稼働している場合、プライマリ・データベースとそのすべてのフィジカル・スタンバイ・データベース間のネットワークが停止すると、ネットワーク接続がリストアされるまで、プライマリ・データベースは使用不可能になります。最大保護モードでは、最後に使用したフィジカル・スタンバイ・データベースが使用不可能になった場合、プライマリ・データベースも停止します。

ネットワークの停止時間が長い場合は、ネットワーク接続がリストアされるまで、プライマリ・データベースを最大可用性モードまたは最大パフォーマンス・モードのいずれかで実行するように変更することを考慮してください。プライマリ・データベースを最大可用性モードに変更すると、プライマリ・データベースとスタンバイ・データベースの間にラグが発生する可能性があります。ただし、ネットワークの問題が解決するまで、プライマリ・データベースを使用することができます。

プライマリ・データベースを最大可用性モードに変更する場合は、次のプロシージャを使用してデータの破損を防止することが重要です。

ネットワークが停止し、Real Application Clusters 構成の保護モードを変更する場合は、次の手順を実行します。

1. フィジカル・スタンバイ・データベースを停止します。
2. 最大保護モードを最大可用性モードまたは最大パフォーマンス・モードのいずれかに変更する場合は、5-26 ページの「Data Guard 構成のデータ保護モードの設定」の説明に従ってください。また、ブローカを使用している場合は、『Oracle9*i* Data Guard Broker』を参照してください。
3. Real Application Clusters プライマリ・データベースをオープンし、通常にアクセスします。

後でネットワークが回復したときに、次の手順を実行して最大保護モードに戻ります。

1. Real Application Clusters プライマリ・データベースを停止した後、データベースをオープンせずにマウントして通常にアクセスします。
2. フィジカル・スタンバイ・データベースをマウントします。
3. Real Application Clusters プライマリ・データベースのモードを現行モード（最大可用性または最大パフォーマンス）から最大保護モードに変更します。
4. Real Application Clusters プライマリ・データベースをオープンします。

D

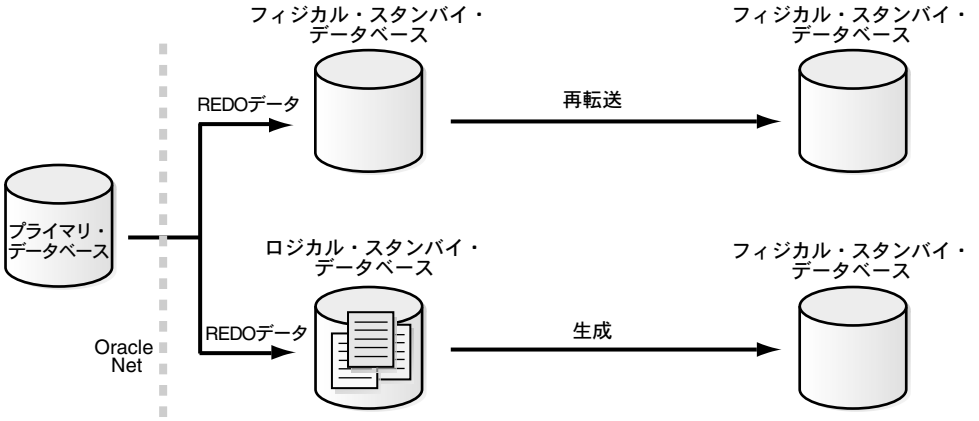
カスケードされた REDO ログ宛先

プライマリ・システムの負荷を軽減するために、**カスケードされた REDO ログ宛先**を実装できます。これによって、スタンバイ・データベースは、REDO ログをプライマリ・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。次のデータベースを構成できます。

- フィジカル・スタンバイ・データベース。プライマリ・データベースから受信した着信 REDO ログを、1 レベルまでのリダイレクションで、プライマリ・データベースと同じ方法で他のリモートの宛先に再転送します。
- ロジカル・スタンバイ・データベース。読取り / 書込みモードでオープンされるため、プライマリ・データベースから受信した REDO データのフィルタ処理と適用が完了した後に生成した REDO ログを、独自のフィジカルまたはロジカルのスタンバイ・データベースのセットに送信します。

図 D-1 は、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・データベースに送信されるカスケードされた REDO ログを示しています。

図 D-1 カスケードされた REDO ログ宛先の構成例



スタンバイ・データベースでは、REDO データを最大 9 個の宛先までカスケードできます。これは、Data Guard 構成に最大 90 個のスタンバイ・データベースを含めることができることを意味します。つまり、1 個のプライマリ・データベースに 9 個のスタンバイ・データベースを含め、さらにその各スタンバイ・データベースが他の 9 個のスタンバイ・データベースに REDO データをカスケードします。ただし、実際には、カスケードされた REDO データを受信するように構成するのは、主としてレポート生成操作またはバックアップ操作のオフロード専用のスタンバイ・データベースのみです。ロールの推移操作に関与する可能性のあるスタンバイ・データベースは、プライマリ・データベースから直接 REDO データを受信するように構成し、`LOG_ARCHIVE_DEST_`*n* パラメータと `LOG_ARCHIVE_DEST_STATE_`*n* パラメータを定義します。この結果、スイッチオーバー操作またはフェイルオーバー操作を実行した場合も、引き続き新しいプライマリ・データベースから直接ログを受信できます。

この項は、次の項目で構成されています。

- カスケードされた REDO ログ宛先の構成
- カスケードされた REDO ログ宛先の例

カスケードされた REDO ログ宛先の構成

次の各項では、カスケードされた REDO ログ宛先を使用できるように Data Guard 構成を設定する方法について説明します。

- フィジカル・スタンバイ・データベースに対するカスケードされた REDO ログ宛先の構成
- ロジカル・スタンバイ・データベースに対するカスケードされた REDO ログ宛先の構成

フィジカル・スタンバイ・データベースに対するカスケードされた REDO ログ宛先の構成

フィジカル・スタンバイ・データベースが着信 REDO ログを別の宛先セットに送信できるようにするには、次の項目を定義する必要があります。

- プライマリ・データベースの LOG_ARCHIVE_DEST_*n* 初期化パラメータを定義して、LGWR 転送方法を使用するカスケードの開始点となるフィジカル・スタンバイ・データベースを設定します。要件に応じて、SYNC または ASYNC ネットワーク・プロトコルを使用します。
- 受信側のフィジカル・スタンバイ・データベースで、十分なスタンバイ REDO ログ・ファイルを定義し、アーカイブが可能であることを確認します。

この時点で、カスケードのエンド・ポイントを定義するフィジカル・スタンバイ・データベースの LOG_ARCHIVE_DEST_*n* 初期化パラメータの定義を開始できます。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・アーカイブ先を定義していることに注意してください。たとえば、LOG_ARCHIVE_DEST_1 初期化パラメータを 'LOCATION=/physical1/arch' の位置に定義したとします。フィジカル・スタンバイ・データベースがロールを切り替えると、すべてのアーカイブ REDO ログは、LOG_ARCHIVE_FORMAT 初期化パラメータで定義した書式で、このディレクトリに格納されます。このローカル・アーカイブ先は、STANDBY_ARCHIVE_DEST パラメータで定義した宛先と同じにできますが、これは必須ではありません。

この構成の副作用は、スタンバイ・データベースのアーカイブ・プロセスが、カスケードのエンド・ポイントのみではなく、他のスタンバイ・データベースおよびプライマリ・データベース（定義済みで使用可能な場合）に対して、ログの送信を試行することです。受信側のデータベースが、同じログをすでに正常に受信しているプライマリ・データベースまたはスタンバイ・データベースの場合は、受信を拒否するため、これは問題にはなりません。宛先のスタンバイ・データベースがログを正常に受信していない場合、この副作用はアクティブなギャップ解消として機能します。この副作用を回避するには、カスケードに関与しないすべての宛先の状態を DEFER に設定します。ただし、スイッチオーバー操作またはフェイルオーバー操作を行う場合は、これらの宛先を再び使用可能にする必要があります。

1 つの初期化パラメータ・ファイルを使用して、カスケードされた REDO ログ宛先と元のプライマリ / スタンバイの宛先の両方を処理する場合は、カスケード・スタンバイ・データベースの宛先に加えて、プライマリ・データベースとその他のスタンバイ・データベースの宛先も定義します。ただし、リモート宛先の合計数は、ローカル・アーカイブ先も含めて 10 を超えることはできません。

スタンバイ・オンライン REDO ログのアーカイブ時に、REDO 情報をカスケードされた REDO ログ宛先に送信するのは、ログ・ライター・プロセスではなくアーカイバ・プロセスです。したがって、送信はプライマリ・データベースに直接接続されたスタンバイ・データベースごとに、1 つのカスケードされた REDO ログ宛先セットに限定されます。

ロジカル・スタンバイ・データベースに対するカスケードされた REDO ログ宛先の構成

プライマリ・データベースから直接 REDO データを受信するロジカル・スタンバイ・データベースは、プライマリ・データベースから受信した REDO データのフィルタ処理と適用が完了した後に生成した REDO データを、他のスタンバイ・データベースにカスケードするように構成できます。ロジカル・スタンバイ・データベースからカスケードされた REDO データは、プライマリ・データベースで生成された元の REDO データと同一ではないため、プライマリ・データベースから直接インスタンス化されたスタンバイ・データベースには適用できません。かわりに、ロジカル・スタンバイ・データベースからカスケードされた REDO データを受信するスタンバイ・データベースは、ロジカル・スタンバイ・データベースのコピーから作成する必要があります。この場合、次のようになります。

- ロジカル・スタンバイ・データベースから作成されたフィジカル・スタンバイ・データベースは、ロジカル・スタンバイ・データベースのブロック単位のコピーとなり、元のプライマリ・データベースの論理的なコピーとなります。
- ロジカル・スタンバイ・データベースから作成されたロジカル・スタンバイ・データベースは、親のロジカル・スタンバイ・データベースの論理的なコピーとなり、元のプライマリ・データベースに一部類似しています。これは、元のプライマリ・データベースのデータ以外に、親のロジカル・スタンバイ・データベースに格納されたすべての内容（異なる索引やマテリアライズド・ビューなどの変更も含む）が存在するためです。

ロジカル・スタンバイ・データベースからカスケードされた REDO データを受信するスタンバイ・データベースの場合は、プライマリ・データベースから直接 REDO データを受信するフィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースと同じ設定タスクを実行する必要があります。任意の転送モード（LGWR または ARCH）およびネットワーク・プロトコル（SYNC または ASYNC）が使用できます。LGWR ネットワーク・プロトコルを使用する場合は、必要に応じて、フィジカル・スタンバイ・データベースでスタンバイ・オンライン REDO ログを使用できます。

カスケードされた REDO ログ宛先の例

次の使用例では、カスケードされた REDO ログ宛先の構成オプションと使用方法を説明します。

使用例 1

本社のオフィスにプライマリ・データベースがあり、別のビルにスタンバイ・データベースを作成して、Local Area Network（LAN）で接続するとします。さらに、社内の保護要件により、REDO 情報とバックアップ・コピーを地理的に離れた位置にあるオフサイトで保管し、LAN ではなく Wide Area Network（WAN）で接続するとします。

この両方のサイトに REDO ログを転送できるように、プライマリ・データベースで 2 つの宛先を定義できますが、WAN を介した REDO ログ送信のネットワーク待機時間が原因で、プライマリ・データベースのスループットに余分なワークロードがかかります。

この問題を解決するために、LGWR および SYNC ネットワーク転送とスタンバイ・オンライン REDO ログを使用して、LAN 上のプライマリ・データベースとフィジカル・スタンバイ・データベースとの間に緊密な接続を定義できます。. これによって、プライマリ・データベースへの接続が失われないように保護し、プライマリ・データベースでメンテナンスが必要になった場合は、本番用の代替サイトを提供できます。WAN で接続されたセカンダリ・ロケーションには、フィジカル・スタンバイ・データベースがサービスを提供し、REDO 情報を確実にオフサイトで格納できるようにします。本番データベースの毎晩のバックアップ操作は、WAN を介してリモート・スタンバイ・データベースに移送できるため、オフサイトの格納場所にテープを発送する必要がなくなります。

プライマリ・データベースと LAN 上のフィジカル・スタンバイ・データベースの両方に接続できない最悪の状況が発生した場合は、最小限のデータ消失でリモート・スタンバイ・データベースにフェイルオーバーできます。元のスタンバイ・データベースからの、最後のスタンバイ・データベースのオンライン REDO ログにアクセスできた場合は、リモート・スタンバイ・データベースでリカバリできるため、データ消失はありません。

WAN を介した情報の送信によって問題が発生するのは、スイッチオーバー操作またはフェイルオーバー操作時（フィジカル・スタンバイ・データベースがプライマリ・ロールに推移したとき）のみです。しかし、この構成は企業の保護要件を満たしています。

使用例 2

離れた位置にプライマリ・データベースがあり、レポート生成のためにそのデータにローカルでアクセスする必要があるとします。プライマリ・データベースには、障害時の回復に備えて、すでに 1 つのスタンバイ・データベースが設定されています。このスタンバイ・データベースはオフサイトの位置にあり、LAN で接続されています。プライマリ・データベースで、このサイトに情報を送信するように宛先を指定すると、プライマリ・データベースのパフォーマンスに悪影響を与えます。

この問題の解決策は、使用例 1 で説明した解決策と似ています。ただし、すでに 1 つのフィジカル・スタンバイ・データベースが準備されているため、REDO ログはロジカル・スタンバイ・データベースに送信することにします。最初に、次の内容を確認します。

- フィジカル・スタンバイ・データベースが、プライマリ・データベースのログ・ライターから REDO ログを受信していること
- スタンバイ REDO ログが定義済みで使用されていること

  スタンバイ REDO ログが未定義の場合は、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 ログの使用を開始します。LGWR ネットワーク転送が使用されていない場合は、プライマリ・データベースでログ転送サービスを動的に設定できます。これによって、プライマリ・データベースは、次回のログ・スイッチの発生時にログ・ライターの使用を開始します。

次に、ロジカル・スタンバイ・データベースの通常の設定タスクを実行します。ロジカル・スタンバイ・データベースの使用準備に必要な手順は、すべて通常プライマリ・ロケーションで実行する必要があります。ロジカル・スタンバイ・データベースが起動して稼働した後、フィジカル・スタンバイ・データベースで宛先パラメータを定義し、REDO ログが WAN を介して送信され、ロジカル・スタンバイ・データベースに適用されるようにします。

使用例 3

製造サイトにあるプライマリ・データベースには、すでに 2 つのフィジカル・スタンバイ・データベースが構成されています。1 つのスタンバイ・データベースは、別のビルに配置され、LAN で接続されており、もう 1 つのスタンバイ・データベースは離れた位置にある本社のオフィスに配置され、WAN で接続されています。この 2 つのスタンバイ・データベースは、非データ消失モードで実行する必要があるため、WAN で接続されたスタンバイ・データベースにカスケード・スタンバイ・データベースは使用できません。また、マーケティング部門では、販売予測のために製造データにアクセスする必要があります。マーケティング部門は毎日データにアクセスし、販売データと製造データを照合して、販売時期に対する製造時期を正確に判断する必要があります。

解決策の 1 つとして、マーケティング部門が、本社にあるフィジカル・スタンバイ・データベースに読取り専用モードでアクセスできるようにする方法があります。しかし、スタンバイ・データベースを読取り専用モードに設定するには、管理リカバリ・プロセスを停止する必要があります。この結果、プライマリ・データベースの内容がフィジカル・スタンバイ・データベースに反映されるのが夜間のみになり、その間も、プライマリ・データベースは、

製造工場での第 2 シフトおよび第 3 シフト作業によるデータを受信していることになります。また、スタンバイ・データベースへの REDO ログの適用が常に、最短でも 12 時間遅れることになります。プライマリ・データベースに別の宛先を追加して、REDO ログを本社オフィス内の異なるロジカル・スタンバイ・データベースに送信することもできます。本社オフィスで使用されているシステムが、フィジカル・スタンバイ・データベースと計画済みのロジカル・スタンバイ・データベースとでは異なるため、スタンバイ宛先の定義時に DEPENDENCY 属性は使用できません。WAN を介して REDO ログを送信する必要があるため、REDO データを 2 回送信することは、プライマリ・データベースのパフォーマンスを許容できないレベルまで低下させると考えられます。

この問題は、カスケードされた REDO ログ宛先によって解決できます。カスケード・スタンバイ・データベースを設定するには、第 4 章の説明に従ってロジカル・スタンバイ・データベースを作成します。また、本社のフィジカル・スタンバイ・データベースが LAN を介してこの新しいロジカル・スタンバイ・データベースに REDO ログを転送するように設定する必要もあります。この方法では、プライマリ・データベースは WAN を介してデータを 1 回のみ送信します。また、ロジカル・スタンバイ・データベースは、新しいマテリアライズド・ビューを使用して変更できるため、マーケティング・グループは、データを効率的に管理できます。ロジカル・スタンバイ・データベースは、読取り / 書込み操作用にオープンされるため、マーケティング・グループは、プライマリ・データベースのパフォーマンス、あるいはフィジカル・スタンバイ・データベースの実行可能性や現行の状態に影響を与えずに、販売データに対する新規スキーマの追加やロードを実行できます。

使用例 4

世界各地に 5 つの販売オフィスがあり、各オフィスに独自のプライマリ・データベースが配置されているとします。すべてのオフィスに、障害時の回復対策を導入するとします。その際、各プライマリ・データベースへの影響を最小限に抑えながら、すべてのデータに適時にアクセスできる方法も考慮します。

この問題を解決するには、最初に、5 箇所の各オフィスに非データ消失環境を実装します。これを行うには、LGWR および SYNC 属性を使用して、各オフィスにローカルのフィジカル・スタンバイ・データベースを作成します。このフィジカル・スタンバイ・データベースは、LAN または WAN で接続できます。次に、5 個の各プライマリ・データベースからロジカル・スタンバイ・データベースを作成して、本社に配置します。ただし、REDO ログは、5 個の各プライマリ・データベースのログ転送サービスによって送信されるのではなく、5 個の各スタンバイ・データベースから WAN を介してロジカル・スタンバイ・データベースに送信されるように構成します。1 つまたはすべてのロジカル・スタンバイ・データベースで、別のロジカル・スタンバイ・データベースへのデータベース・リンクを定義し、このリンクによって、全販売データにアクセスできるようにします。5 個の各プライマリ・データベースの全情報ではなく、特定の表のみ必要な場合は、SKIP ルーチンを使用して、各ロジカル・スタンバイ・データベースに不要なデータの適用を停止できます。

使用例 5

現在、夜間のバックアップ操作のみで保護されているプライマリ・データベースがあるとします。優れた障害時リカバリ対策をすぐに導入する必要があります。社内に、同じハードウェア・タイプの別のシステムがありますが、フェイルオーバー用のスタンバイ・データベースとして使用するには処理能力が低く、データベース全体を格納するディスクも不足しています。データベース全体の格納に十分に対応できる唯一の別のシステムは、LAN で接続するには離れた場所にあり、WAN で接続すると極端に低速になります。この対策の導入期限は、いずれかのネットワークがアップグレードされるまでです。プライマリ・データベースで宛先を追加して、REDO ログをリモート・ロケーションに送信すると、パフォーマンスに重大な影響を与えます。

この問題の暫定的な解決策は、リモート・システムでフィジカル・スタンバイ・データベースを作成し、ローカルの小規模なシステムに分散リポジトリを作成することです。分散リポジトリは、スタンバイ制御ファイルとスタンバイ・データベースのオンライン REDO ログのみで構成され、データ・ファイルは格納されません。REDO 情報がローカルのリポジトリに送信されるようにプライマリ・データベースを構成するには、ログ・ライター・プロセス（LGWR）を同期モード（SYNC）で使用します。LAN で接続されるため、パフォーマンスへの影響は最小限で済みます。次に、リポジトリからデータが WAN を介して実際のスタンバイ・データベースに送信されるように構成します。

この構成のリスクは、プライマリ・データベースの障害時に、プライマリ・データベースからスタンバイ・データベースへの全データの転送が完了していても、リポジトリからリモート・スタンバイ・データベースへのデータ送信が完了していない可能性があることです。この環境では、両方のシステムに同時に障害が発生しないかぎり、リモート・スタンバイ・データベースでは、最後のログ・スイッチまでに送信された全データを受信できます。現行のオンライン REDO ログは、手動で送信する必要があります。

WAN がアップグレードされて、リモート・スタンバイ・データベースに直接接続できるようになった場合は、リポジトリの宛先を変更して、リモート・スタンバイ・データベースを直接指すようにするか、新たにリモート・スタンバイ・データベースの宛先を作成し、リポジトリへの転送は、アーカイブ・ログ・リポジトリとして継続します。

E

障害時リカバリに関する ReadMe ファイルのサンプル

複数のスタンバイ・データベース構成では、複数のスタンバイ・データベース構成を設定したデータベース管理者（DBA）が、どのスタンバイ・データベースにフェイルオーバーするのが適切であるかを必ずしも決定できる状態にあるとは限りません。したがって、各スタンバイ・サイトで、プライマリ・サイト同様に障害時リカバリ計画を立てることが必要になります。障害時リカバリ・チームの各メンバーは、障害時リカバリ計画を知っており、実行する手順を認識している必要があります。

例 E-1 では、どのスタンバイ・データベースをフェイルオーバー操作のターゲットとするかを決定する上で必要な情報を示しています。

ReadMe ファイルは、DBA によって作成およびメンテナンスされ、次の方法を記述している必要があります。

- DBA としてのローカル・データベース・サーバーへのログオン方法
- スタンバイ・データベースが存在する各システムへのログオン方法

  システム間にはファイアウォールが存在する可能性があります。ReadMe ファイルには、ファイアウォールを通過する手順も記載されている必要があります。
- DBA として他のデータベース・サーバーへログオンする方法
- 最も新しいログが適用されているスタンバイ・データベースの識別方法
- スタンバイ・データベース・フェイルオーバー処理の実行方法
- クライアント・アプリケーションが、元のプライマリ・データベースではなく、新しいプライマリ・データベースにアクセスできるようなネットワーク設定の構成方法

例 E-1 障害時リカバリに関する ReadMe ファイルのサンプル

```
----------------Standby Database Disaster Recovery ReadMe File----------------

Warning:
********************************************************************************
Perform the steps in this procedure only if you are responsible for failing over to
a standby database after the primary database fails.

If you perform the steps outlined in this file unnecessarily, you might corrupt the
entire database system.
********************************************************************************

Multiple Standby Database Configuration:

No.     Location       Type     IP Address
--- --------------- --------- --------------
 1   San Francisco   Primary   128.1.124.25
 2   San Francisco   Standby   128.1.124.157
 3   Boston          Standby   136.132.1.55
 4   Los Angeles     Standby   145.23.82.16
 5   San Francisco   Standby   128.1.135.24

You are in system No. 3, which is located in Boston.

Perform the following steps to fail over to the most up-to-date and available
standby database:

1. Log on to the local standby database as a DBA.

    a)  Log on with the following user name and password:

                username: Standby3
                password: zkc722Khn

    b)  Invoke SQL*Plus as follows:

        % sqlplus

    c)  Connect as the DBA as follows:

        CONNECT sys/s23LsdIc AS SYSDBA
```

2. Connect to as many remote systems as possible. You can connect to a maximum of four systems. System 4 does not have a firewall, so you can connect to it directly. Systems 1, 2, and 5 share the same firewall host. You need to go to the firewall host first and then connect to each system. The IP address for the firewall host is 128.1.1.100. Use the following user name and password:

```
username: Disaster
password: 82lhsIW32
```

3. Log on to as many remote systems as possible with the following user names and passwords:

Login information:

No.	Location	IP Address	username	password
1	San Francisco	128.1.124.25	Oracle9i	sdd290Ec
2	San Francisco	128.1.124.157	Standby2	ei23nJHb
3		(L o c a l)		
4	Los Angeles	145.23.82.16	Standby4	23HHoe2a
5	San Francisco	128.1.135.24	Standby5	snc#$dnc

4. Invoke SQL*Plus on each remote system you are able to log on to as follows:

```
% sqlplus
```

5. Connect to each remote database as follows:

```
CONNECT sys/password AS SYSDBA
```

The DBA passwords for each location are:

No.	Location	Password
1	San Francisco	x2dwlsd91
2	San Francisco	a239s1DAq
3	(L o c a l)	
4	Los Angeles	owKL(@as23
5	San Francisco	sad_KS13x

6. If you are able to log on to System 1, invoke SQL*Plus and execute the following statements:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP PFILE=PRMYinit.ora;
```

```
    Note: If you are able to execute the STARTUP statement successfully, the
          primary database has not been damaged.  Do not continue with this
          procedure.

7.  Execute the following SQL statements on each standby database (including the
    one on this system) that you were able to connect to:

    SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
    SQL> SELECT THREAD#, MAX(SEQUENCE#) FROM V$LOG_HISTORY GROUP BY THREAD#;

    Compare the query results of each standby database. Fail over to the
    standby database with the largest sequence number.

8.  Fail over to the standby database with the largest sequence number.

    On the standby database with the largest sequence number, invoke SQL*Plus
    and execute the following SQL statements:

    SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
      2> DISCONNECT FROM SESSION;
    SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
    SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
    SQL> SHUTDOWN IMMEDIATE;
    SQL> STARTUP PFILE=Failover.ora;

9.  Update the other standby databases with the new primary database information and
ensure the log transport and apply services are working correctly.

------------End of Standby Database Disaster Recovery ReadMe File-------------
```

用語集

ARCH

「アーカイバ・プロセス（archiver process: ARCn）」を参照。

ARCHIVELOG モード（ARCHIVELOG mode）

データベースのモードの 1 つ。このモードでは、いっぱいになったオンライン REDO ログをログ転送サービスがディスクにアーカイブする。このモードは、データベース作成時に指定するか、または SQL の ALTER DATABASE ARCHIVELOG 文を使用して指定する。自動アーカイブを使用可能にするには、SQL の ALTER SYSTEM ARCHIVE LOG START 文を動的に使用するか、または初期化パラメータ LOG_ARCHIVE_START を TRUE に設定する。

データベースを ARCHIVELOG モードで実行した場合は、NOARCHIVELOG モードと比較していくつかの利点がある。次の処理が可能となる。

- オープンされ、ユーザーがアクセスしているデータベースのバックアップ
- データベースの任意の時点までのリカバリ

障害発生時に ARCHIVELOG モードのデータベースを保護するため、アーカイブ・ログはバックアップすること。

「アーカイブ REDO ログ（archived redo log）」、「NOARCHIVELOG モード（NOARCHIVELOG mode）」および「REDO ログ（redo log）」も参照。

ARC*n*

「アーカイバ・プロセス（archiver process: ARCn）」を参照。

Data Guard

本番データベースまたは 1 つ以上のスタンバイ・データベースと連携して、管理、監視および自動化を行うソフトウェア。このソフトウェアによって、データベースを破壊する可能性のあるエラー、障害および破損からデータを保護する。

「プライマリ・データベース（primary database）」および「スタンバイ・データベース（standby database）」も参照。

FAL クライアント（FAL client）

「**FAL クライアント（Fetch Archive Log client）**」を参照。

FAL クライアント（Fetch Archive Log client）

バックグラウンドの Oracle データベース・サーバー・プロセス。FAL クライアントの初期化パラメータは、スタンバイ・データベースで設定される。FAL クライアントは、スタンバイ・データベースでアーカイブ・ギャップを検出すると、プライマリ・ロケーションからアーカイブ REDO ログをプルし、アーカイブ REDO ログの転送を自動的に開始および要求する。

「**FAL サーバー（Fetch Archive Log server）**」も参照。

FAL サーバー（FAL server）

「**FAL サーバー（Fetch Archive Log server）**」を参照。

FAL サーバー（Fetch Archive Log server）

バックグラウンドの Oracle データベース・サーバー・プロセスの 1 つ。プライマリ・データベースまたはその他のスタンバイ・データベースで実行され、FAL クライアントからの FAL 要求を処理する。FAL 要求の処理としては、たとえば、FAL サーバーを実行する Oracle データベース・サーバーへのキューイング要求（アーカイブ REDO ログを 1 つ以上のスタンバイ・データベースに送信するため）などがある。1 つのプライマリ・データベースで同時に複数の FAL サーバーを実行できる。受信した各 FAL 要求ごとに個別の FAL サーバーが作成される。FAL サーバーの初期化パラメータは、スタンバイ・データベースで設定される。

「**FAL クライアント（Fetch Archive Log client）**」も参照。

LGWR

「**ログ・ライター・プロセス（Log Writer Process: LGWR）**」を参照。

MRP

「**管理リカバリ・プロセス（managed recovery process: MRP）**」を参照。

NOARCHIVELOG モード（NOARCHIVELOG mode）

データベースのモードの 1 つ。このモードでは、ログ転送サービスは、いっぱいになったオンライン REDO ログがディスクにアーカイブされることを要求しない。このモードは、データベース作成時に指定するか、または SQL の `ALTER DATABASE` 文を使用して変更する。`NOARCHIVELOG` モードでは、失われたデータがリカバリされる可能性が非常に小さくなるため、オラクル社ではこのモードによる実行を推奨しない。

「**ARCHIVELOG モード（ARCHIVELOG mode）**」も参照。

Recovery Manager

Oracle データベースのバックアップ、リストアおよびリカバリを実行するユーティリティ。このユーティリティは、リカバリ・カタログと呼ばれる中央情報リポジトリとともに使用することも、またリカバリ・カタログなしで使用することもできる。リカバリ・カタログを使用しない場合、Recovery Manager は、データベースの制御ファイルを使用してバックアップおよびリカバリ処理に必要な情報を格納する。Recovery Manager とメディア・マネージャをあわせて使用すると、ファイルをテープなどの 3 次記憶装置にバックアップできる。

「**バックアップ・ピース（backup piece）**」、「**バックアップ・セット（backup set）**」および「**リカバリ・カタログ（recovery catalog）**」も参照。

REDO ログ（redo log）

REDO レコードを含むファイル。REDO ログには、オンライン REDO ログ、スタンバイ REDO ログおよびアーカイブ REDO ログの 3 つのタイプがある。

オンライン REDO ログは、データ・ファイルと制御ファイルに行われたすべての変更を記録する、2 つ以上のファイルのセットである。LGWR プロセスは、REDO レコードをこのログに記録する。現行のオンライン REDO ログとは、LGWR が現在書込みをしているオンライン REDO ログである。

スタンバイ REDO ログは、オプションの位置の 1 つで、スタンバイ・データベースがプライマリ・データベースから受信した REDO データを格納できる。この REDO データは、スタンバイ REDO ログまたはアーカイブ REDO ログを使用してスタンバイ・ロケーションに格納できる。

アーカイブ REDO ログは、オンライン REDO ログがオフラインの宛先にコピーされたもので、オフライン REDO ログとも呼ばれる。データベースが `ARCHIVELOG` モードになっている場合、いっぱいになったオンライン REDO ログは、1 つ以上の ARC*n* プロセスで、1 つ以上のアーカイブ・ログ宛先にコピーされる。

「**アーカイブ REDO ログ（archived redo log）**」、「**ARCHIVELOG モード（ARCHIVELOG mode）**」、「**現行のオンライン REDO ログ（current online redo log）**」、「**ログ・スイッチ（log switch）**」、「**オンライン REDO ログ（online redo log）**」および「**スタンバイ REDO ログ（standby redo log）**」も参照。

SQL 適用モード（SQL apply mode）

ログ適用サービスがアーカイブ REDO ログの情報をロジカル・スタンバイ・データベースに自動的に適用するモード。ログ適用サービスは、トランザクション情報を SQL 文に変換し、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する。

「**ログ適用サービス（log apply services）**」も参照。

TAF

「**透過的アプリケーション・フェイルオーバー（Transparent Application Failover: TAF）**」を参照。

アーカイバ・プロセス（archiver process: ARC*n*）

プライマリ・データベースがある位置で、スタンバイ・データベースのオンライン REDO ログのコピーを、ローカルまたはリモートで作成するプロセス。または、そのようなアーカイブ操作を実行する SQL セッション。スタンバイ・データベースがある位置で、ARC*n* プロセスは、管理リカバリ・プロセス（MRP）によって適用されるスタンバイ REDO ログをアーカイブする。

アーカイブ（archiving）

ARC*n* バックグラウンド・プロセスが、いっぱいになったオンライン REDO ログをオフラインの宛先にコピーする操作。REDO ログをアーカイブするには、プライマリ・データベースを ARCHIVELOG モードで実行する必要がある。

アーカイブ REDO ログ（archived redo log）

いっぱいになったオンライン REDO ログ・グループのメンバーのコピーで、データベースが ARCHIVELOG モードのときに作成される。LGWR プロセスが各オンライン REDO ログを REDO レコードで満たすと同時に、ログ転送サービスが、このログを 1 つ以上のオフラインのアーカイブ・ログ宛先にコピーする。このコピーがアーカイブ REDO ログであり、オフライン REDO ログとも呼ばれる。

「**ARCHIVELOG モード（ARCHIVELOG mode）**」、「**オンライン REDO ログ（online redo log）**」および「**REDO ログ（redo log）**」も参照。

アーカイブ・ギャップ（archive gap）

プライマリ・データベースによって生成された次のアーカイブ REDO ログを、スタンバイ・データベースに適用できないときに作成されるアーカイブ REDO ログの範囲。

宛先（destinations）

ログ転送サービスでは、宛先と呼ばれる、最大 10 箇所のローカルおよびリモートの位置に REDO ログをアーカイブするように、プライマリ・データベースを構成できる。

「**子宛先（child destination）**」、「**宛先依存性（destination dependency）**」および「**親宛先（parent destination）**」も参照。

宛先依存性（destination dependency）

指定された宛先への REDO ログのアーカイブが、別の宛先への REDO ログのアーカイブの成功または失敗に依存するように、ログ転送サービスが構成されていること。

「**子宛先（child destination）**」、「**宛先（destinations）**」および「**親宛先（parent destination）**」も参照。

一時表領域（temporary tablespace）

SQL 文の処理中に作成される一時的な表の表領域。この表領域により、問合せを作成するために、読取り専用モードで一時ファイルのエントリを追加できる。さらに、ディクショナ

リ・ファイルまたは REDO 項目の生成に影響を与えずに、読取り専用データベースでディスク上ソート処理を実行できる。

「**表領域（tablespace）**」および「**一時ファイル（tempfile）**」も参照。

一時ファイル（tempfile）

一時表領域にあり、TEMPFILE オプションで作成されるファイル。一時表領域には、表などの永続データベース・オブジェクトは格納できない。通常は、ソートに使用される。一時ファイルには永続オブジェクトが含まれないため、Recovery Manager によるバックアップは行われない。

「**一時表領域（temporary tablespace）**」も参照。

一貫性バックアップ（consistent backup）

メディア・リカバリを実行せずに RESETLOGS オプションでオープンできる、データベース全体のバックアップ（データベースに属する制御ファイルおよびすべてのデータ・ファイルのバックアップ）。つまり、このバックアップのデータ・ファイルには、一貫性を持たせるために REDO ログを適用する必要がない。一貫性バックアップのデータ・ファイルには、次の条件が必要である。

- ヘッダーに、同じチェックポイントの SCN を持つ。ただし、読取り専用の表領域のデータ・ファイル、または NORMAL モードでオフラインされた表領域のデータ・ファイルの場合を除く（このようなデータ・ファイルは、チェックポイントの SCN より前の正しい SCN を持つ）。
- チェックポイントの SCN より後の変更を含まない。
- 制御ファイルに格納されたデータ・ファイルのチェックポイント情報と一致する。

一貫性バックアップは、データベースを正しく停止した後にのみ実行できる。バックアップが完了するまで、データベースをオープンしないこと。

「**クローズ状態のバックアップ（closed backup）**」も参照。

インスタンス間アーカイブ環境（cross-instance archival environment）

Real Application Clusters の環境。各インスタンスがそのアーカイブ REDO ログをクラスタのシングル・インスタンスへ転送するように設定されている状態。このシングル・インスタンスは、リカバリ・インスタンスと呼ばれる。

「**リカバリ・インスタンス（recovery instance）**」も参照。

オープン状態のバックアップ（open backup）

データベースがオープンしている間に作成される、1 つ以上のデータ・ファイルのバックアップ。これはホット・バックアップとも呼ばれる。

オフライン REDO ログ（offline redo log）

「**アーカイブ REDO ログ（archived redo log）**」を参照。

親宛先（parent destination）

子宛先が対応付けられた、ログ転送サービスのアーカイブ先。

「**子宛先（child destination）**」、「**宛先依存性（destination dependency）**」および「**宛先（destinations）**」も参照。

オンライン REDO ログ（online redo log）

データ・ファイルと制御ファイルに行われたすべての変更を記録する、2 つ以上のファイルのセット。データベースに変更が行われたときは、Oracle データベース・サーバーは必ず REDO バッファに REDO レコードを生成する。REDO バッファの内容は、LGWR プロセスによってオンライン REDO ログにフラッシュされる。

どのデータベースにも、最低 2 つのオンライン REDO ログが含まれている必要がある。オンライン REDO ログ・ファイルを多重化する場合は、LGWR は同じ REDO データを同時に複数のファイルに書き込む。各ファイルは、オンライン REDO ログ・グループのメンバーと呼ばれる。

「**アーカイブ REDO ログ（archived redo log）**」、「**現行のオンライン REDO ログ（current online redo log）**」、「**REDO ログ（redo log）**」および「**スタンバイ REDO ログ（standby redo log）**」も参照。

カスケード・スタンバイ・データベース（cascading standby database）

REDO ログを元のプライマリ・データベースではなく、別のスタンバイ・データベースから受信するスタンバイ・データベース。

可用性（availability）

システムまたはリソースの、必要なときに目的のサービスを提供できる能力の度合い。デバイスを必要とする合計時間と、そのデバイスにアクセスできる時間との割合で測定される。コンピュータ・サービスが中断されないことが必須のビジネスでは、可用性目標は 100 パーセント、つまり 24 時間 365 日である。可用時間と停止時間の合計は 100 パーセントである。

「**停止時間（downtime）**」および「**スイッチオーバー（switchover）**」も参照。

完全なサプリメンタル・ロギング（full supplemental logging）

ログ・スイッチを実行してから `DBMS_LOGMNR_D.BUILD` プロシージャを起動することによって、サプリメンタル・ロギングが正しく設定されるようにすること。

「**サプリメンタル・ロギング（supplemental logging）**」も参照。

管理スタンバイ環境（managed standby environment）

「**スタンバイ・データベース環境（standby database environment）**」を参照。

管理リカバリ・プロセス（managed recovery process: MRP）

アーカイブ REDO ログ情報をスタンバイ・データベースに適用するプロセス。

管理リカバリ・モード（managed recovery mode）

プライマリ・データベースが REDO ログをスタンバイ・ロケーションに自動的にアーカイブする環境で、次の SQL 文が入力されると、スタンバイ・データベースが管理リカバリ・モードになる。

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

スタンバイ・データベースが管理リカバリ・モードで実行されているときは、プライマリ・データベースから受信した REDO ログを自動的に適用する。

クローズ状態のバックアップ（closed backup）

データベースがクローズしている間に作成される、1 つ以上のデータベース・ファイルのバックアップ。通常、クローズ状態のバックアップは、データベース全体のバックアップ（データベースに属する制御ファイルおよびすべてのデータ・ファイルのバックアップ）でもある。データベースが正常にクローズされた場合は、バックアップに含まれているファイルはすべて一貫性のある状態になっている。SHUTDOWN ABORT 文を使用してデータベースを停止した場合、またはインスタンスが異常終了した場合は、バックアップに一貫性がない。

「**一貫性バックアップ（consistent backup）**」も参照。

計画停止時間（planned downtime）

「**可用性（availability）**」、「**停止時間（downtime）**」および「**スイッチオーバー（switchover）**」を参照。

現行のオンライン REDO ログ（current online redo log）

現在 LGWR バックグラウンド・プロセスが REDO レコードを記録中の、オンライン REDO ログ。LGWR が書込みを行っていないログは、アクティブでないログと呼ばれる。

LGWR は、ログの終わりに到達するとログ・スイッチを実行し、新規ログへの書込みを開始する。データベースを ARCHIVELOG モードで実行する場合は、1 つ以上の ARC*n* プロセスが REDO データをアーカイブ REDO ログにコピーする。

「**オンライン REDO ログ（online redo log）**」および「**REDO ログ（redo log）**」も参照。

現行の制御ファイル（current control file）

ディスク上の、プライマリ・データベースの制御ファイル。これは最新の変更済み制御ファイルで、データベースの現行のインカネーションである。制御ファイルがリカバリ時に現行の制御ファイルとして認識されるには、バックアップからリストアされなかったことが必要である。

「**バックアップ制御ファイル（backup control file）**」および「**制御ファイル（control file）**」も参照。

コールド・バックアップ（cold backup）

「**クローズ状態のバックアップ（closed backup）**」を参照。

子宛先（child destination）

ログ転送サービスのアーカイブ先の 1 つ。プライマリ・データベースから REDO ログを受信するために構成され、親宛先のアーカイブ操作の正常な完了に依存する。

「**宛先依存性（destination dependency）**」、「**宛先（destinations）**」および「**親宛先（parent destination）**」も参照。

最大可用性モード（maximum availability mode）

ログ転送サービスのデータ保護モードの 1 つ。このモードに設定すると、現行のデータベース・トランザクションがコミットされる前に、スタンバイ・データベースで REDO ログを使用できる。

「**非データ消失（no data loss）**」、「**ログ転送サービス（log transport services）**」、「**最大パフォーマンス・モード（maximum performance mode）**」および「**最大保護モード（maximum protection mode）**」も参照。

最大パフォーマンス・モード（maximum performance mode）

ログ転送サービスのデータ保護モードの 1 つ。最低レベルのデータ保護を提供する。このモードでは、アーカイバ・プロセスは、可能なかぎりプライマリ・データベースのパフォーマンスに影響を与えずに、REDO ログ・データをスタンバイ・データベースに非同期で書き込む。フェイルオーバー操作では、未転送の 1 つ以上のログからデータが消失する可能性がある。これがデフォルトの保護モードである。

「**非データ消失（no data loss）**」、「**ログ転送サービス（log transport services）**」、「**最大可用性モード（maximum availability mode）**」および「**最大保護モード（maximum protection mode）**」も参照。

最大保護モード（maximum protection mode）

ログ転送サービスのデータ保護モードの 1 つ。このモードに設定すると、プライマリ・データベース処理が継続可能になる前に、スタンバイ・データベースで REDO ログを使用できる。このモードはスタンバイ REDO ログを必要とするため、ロジカル・スタンバイ・データベースでは使用できない。

「**非データ消失（no data loss）**」、「**ログ転送サービス（log transport services）**」、「**最大可用性モード（maximum availability mode）**」および「**最大パフォーマンス・モード（maximum performance mode）**」も参照。

サイト（site）

この用語は、Data Guard 構成では、プライマリ・データベースまたはスタンバイ・データベースのローカルの位置または地理的に離れたリモートの位置を示す場合がある。

Data Guard Broker 構成では、フェイルオーバーの管理単位を示す。

サプリメンタル・ロギング（supplemental logging）

追加情報を REDO ログ・ストリームに記録する機能。LogMiner によって、行変更に関連する REDO ストリームをログ・マイニング時にグループ化およびマージし、識別キーを使用して行を識別することもできる。

「**完全なサプリメンタル・ロギング（full supplemental logging）**」も参照。

システム変更番号（System Change Number: SCN）

コミットされたバージョンのデータベースを、ある時点で定義するスタンプ。Oracle データベース・サーバーは、コミットされたすべてのトランザクションに一意の SCN を割り当てる。

受信インスタンス（receiving instance）

Real Application Clusters 構成でスタンバイ・データベースを使用するときは、どのインスタンスも、アーカイブ・ログをプライマリ・データベースから受信できる。これを受信インスタンスという。

「**リカバリ・インスタンス（recovery instance）**」も参照。

手動リカバリ・モード（manual recovery mode）

プライマリ・データベースが、REDO ログをスタンバイ・ロケーションに自動的にアーカイブしない環境。この環境では、アーカイブ・ログをスタンバイ・ロケーションに手動で転送し、次の SQL 文を発行して手動で適用する必要がある。

```
ALTER DATABASE RECOVER STANDBY DATABASE;
```

このモードでは、スタンバイ・データベースを手動でリカバリできる。

信頼性（reliability）

コンピュータ・システムまたはソフトウェアの、障害なく稼働できる能力。

スイッチオーバー（switchover）

プライマリ・データベースとスタンバイ・データベースの 1 つとの間のロールの可逆的な推移。スイッチオーバー操作に関与するプライマリ・データベースとスタンバイ・データベースは、アプリケーション・データを消失することなくロールを交換する。また、構成内の他のスタンバイ・データベースを再起動または再作成する必要もない。スイッチオーバー操作は、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ために使用することはできない。ただし、ハードウェア・ベースのローリング・アップグレードを実行するために使用することはできる。

「**可用性（availability）**」、「**停止時間（downtime）**」、「**フェイルオーバー（failover）**」および「**ロールの推移（role transition）**」も参照。

スタンバイ REDO ログ（standby redo log）

スタンバイ REDO ログは、オプションのログ・セットの 1 つであり、ここにはスタンバイ・データベースがプライマリ・データベースから受信した REDO データを格納できる。（REDO データは、アーカイブ REDO ログを使用してスタンバイ・ロケーションにも格納できる。）スタンバイ REDO ログは、SQL の `ALTER DATABASE` 文の `ADD STANDBY LOGFILE` 句を使用して作成される。ログ・グループに後でメンバーを追加することにより、スタンバイ・ロケーションのディスク障害に対する信頼性のレベルを向上させることができる。最大保護モードを使用している場合、スタンバイ REDO ログが必須である。スタンバイ REDO ログは、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。

「**REDO ログ（redo log）**」も参照。

スタンバイ・データベース（standby database）

障害回復に使用可能な、プライマリ・データベースと同じ内容のコピー。スタンバイ・データベースは、プライマリ・データベースからのアーカイブ REDO ログを使用して更新することにより、最新の状態を維持できる。万一障害によりプライマリ・データベースが破壊された場合は、スタンバイ・データベースにフェイルオーバーし、これを新しいプライマリ・データベースにすることができる。スタンバイ・データベースには、独自の初期化パラメータ・ファイル、制御ファイルおよびデータ・ファイルがある。

「**ロジカル・スタンバイ・データベース（logical standby database）**」、「**フィジカル・スタンバイ・データベース（physical standby database）**」および「**プライマリ・データベース（primary database）**」も参照。

スタンバイ・データベース環境（standby database environment）

プライマリ・データベースおよびスタンバイ・データベースの物理的な構成。環境は、次の項目を含む多数の要因に依存する。

- プライマリ・データベースに対応付けられているスタンバイ・データベースの数
- データベースに使用されるホスト・システムの数
- データベースのディレクトリ構造
- ネットワーク構成

プライマリ・データベースが REDO ログをスタンバイ・ロケーションに自動的にアーカイブする構成は、管理スタンバイ環境である。管理リカバリ・モードのスタンバイ・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ログは自動的にスタンバイ・データベースに適用される。管理スタンバイ環境では、スタンバイ・データベースが管理リカバリ・モードではない場合も、プライマリ・データベースは、アーカイブ REDO ログの転送を継続する。

制御ファイル（control file）

データベースに対応付けられたバイナリ・ファイルの 1 つ。そのデータベース内のすべてのファイルの物理構造とタイムスタンプを維持する。Oracle データベース・サーバーは、データベースの使用中に制御ファイルを継続的に更新する。また、Oracle データベース・サーバーは、データベースがマウントまたはオープンされているときは、常に制御ファイルを書込み可能な状態にしておく必要がある。

「**バックアップ制御ファイル（backup control file）**」および「**現行の制御ファイル（current control file）**」も参照。

ターゲット・データベース（target database）

Recovery Manager で、バックアップまたはリストアするデータベース。

データ消失（data loss）

データ消失は、プライマリ・データベースからすべての REDO データを受信しなかったスタンバイ・データベースにフェイルオーバーした場合に発生する。

データ・ファイル（datafile）

Oracle データベース・サーバーによって作成され、表や索引などのデータ構造を含む、ディスク上の物理的なオペレーティング・システム・ファイル。データ・ファイルは、1 つのデータベースにのみ属することができる。

「**表領域（tablespace）**」も参照。

停止時間（downtime）

システムまたはリソースの、必要なときに目的のサービスを提供する能力の欠如の度合い。デバイスを必要とする合計時間と、そのデバイスにアクセスできない時間との割合で測定される。可用時間と停止時間の合計は 100 パーセントである。

ハードウェアやソフトウェアのアップグレードおよび付加価値サービスなどの、定期的なメンテナンス・タスクを実行するための一定の期間は、計画停止時間である。このような予定されたメンテナンス・タスクの間は、コンピュータ・システムを生産的な操作に使用できない。

「**可用性（availability）**」および「**スイッチオーバー（switchover）**」も参照。

透過的アプリケーション・フェイルオーバー（Transparent Application Failover: TAF）

フェイルオーバーが発生した後、データベースに自動的に再接続し、作業を再開するクライアント・アプリケーションの機能。

ノード（node）

「**サイト（site）**」を参照。

バックアップ制御ファイル（backup control file）

制御ファイルのバックアップ。このバックアップは、次の方法で作成する。

- Recovery Manager ユーティリティの `backup` または `copy` コマンドを使用する。バックアップ制御ファイルの作成には、オペレーティング・システム・コマンドを使用しないこと。
- SQL 文 `ALTER DATABASE BACKUP CONTROLFILE TO '`*`filename`*`'` を使用する。

通常、バックアップ制御ファイルをリストアするのは、現行の制御ファイルのコピーがすべて破損したときだが、ある種の Point-in-Time リカバリの実行前にリストアすることもある。

「**制御ファイル（control file）**」および「**現行の制御ファイル（current control file）**」も参照。

バックアップ・セット（backup set）

バックアップ・ピースと呼ばれる 1 つまたは複数の物理ファイルの、Recovery Manager 固有の論理的なグループ。Recovery Manager の `BACKUP` コマンドの出力がバックアップ・セットである。バックアップ・セットからファイルを抽出するには、Recovery Manager の `RESTORE` コマンドを使用する。バックアップ・セットには、ファイルを多重化できる。つまり、いくつかの入力ファイルのブロックを混合して 1 つのバックアップ・セットにすることができる。

バックアップ・セットには、次の 2 つのタイプがある。

- データ・ファイル・バックアップ・セット。これはデータ・ファイルまたは制御ファイルのバックアップである。このタイプのバックアップ・セットは圧縮される。つまり、使用されたデータ・ファイル・ブロックのみを含み、使用されなかったブロックは省略される。
- アーカイブ・ログ・バックアップ・セット。これはアーカイブ REDO ログのバックアップである。

「**バックアップ・ピース（backup piece）**」および「**Recovery Manager**」も参照。

バックアップ・ピース（backup piece）

Recovery Manager に固有の形式の物理ファイル。このファイルは 1 つのバックアップ・セットのみに属する。通常、バックアップ・セットは 1 つのバックアップ・ピースのみで構成される。Recovery Manager が複数のバックアップ・ピースを作成するのは、Recovery Manager の `ALLOCATE` または `CONFIGURE` コマンドの `MAXPIECESIZE` オプションを使用して、ピース・サイズを制限する場合のみである。

「**バックアップ・セット（backup set）**」および「**Recovery Manager**」も参照。

非管理リカバリ・モード（non-managed recovery mode）

「**手動リカバリ・モード（manual recovery mode）**」を参照。

非データ消失（no data loss）

ログ転送サービスのオプションの 1 つ。このオプションを構成すると、プライマリ・データベースに対して行われたデータ修正が、スタンバイ・データベースでも使用可能になるまで（ただし適用される必要はない）は、認識されない。

「**ログ転送サービス（log transport services）**」、「**最大可用性モード（maximum availability mode）**」および「**最大保護モード（maximum protection mode）**」も参照。

表領域（tablespace）

データベースの分割の単位になる、1 つ以上の論理的な記憶単位。各表領域には、その表領域にのみ対応付けられた物理的なデータ・ファイルが 1 つ以上格納される。

「**データ・ファイル（datafile）**」も参照。

フィジカル・スタンバイ・データベース（physical standby database）

プライマリ・データベースと物理的に同一であるスタンバイ・データベース。リカバリが物理 ROWID を使用してブロックごとに変更を適用する。

「**ロジカル・スタンバイ・データベース（logical standby database）**」および「**スタンバイ・データベース（standby database）**」も参照。

フェイルオーバー（failover）

ロールの不可逆的な推移。スタンバイ・データベースがプライマリ・ロールに推移し、古いプライマリ・データベースは、構成に含まれなくなる。フェイルオーバー操作前に古いプライマリ・データベースがどの保護モードで実行されていたかによって、フェイルオーバー時に一部のデータが消失する場合がある。通常、フェイルオーバーは、プライマリ・データベースがシステムやソフトウェアの障害などで使用不可能になり、適正な時間内でスイッチオーバーの実行またはプライマリ・データベースの完全な修正ができない場合にのみ使用される。

「**ロールの推移（role transition）**」および「**スイッチオーバー（switchover）**」も参照。

フェッチ・アーカイブ・ログ（fetch archive log: FAL）

「**FAL クライアント（Fetch Archive Log client）**」および「**FAL サーバー（Fetch Archive Log server）**」を参照。

プライマリ・データベース（primary database）

Data Guard 構成では、本番データベースはプライマリ・データベースと呼ばれる。プライマリ・データベースを使用して、スタンバイ・データベースが作成される。各スタンバイ・データベースは、1 つのプライマリ・データベースのみと対応付けられる。ただし、各プライマリ・データベースからは、複数のスタンバイ・データベースをサポートできる。

「**スタンバイ・データベース（standby database）**」も参照。

ブローカ（broker）

分散管理フレームワークの 1 つ。Data Guard 構成の作成、制御および監視に関するほとんどの操作を自動化し単純化する。ブローカのユーザー・インタフェースには、Oracle Data Guard Manager（GUI）と、Data Guard のコマンドライン・インタフェースの 2 つがある。

プロテクト・モード（protected mode）

「**最大保護モード（maximum protection mode）**」を参照。

ホット・バックアップ（hot backup）

「**オープン状態のバックアップ（open backup）**」を参照。

読取り専用データベース（read-only database）

SQL 文 `ALTER DATABASE OPEN READ ONLY` でオープンされるデータベース。その名称が示すとおり、読取り専用データベースは問合せ専用で、修正はできない。スタンバイ・データベースは読取り専用モードで実行できる。つまり、プライマリ・データベースの最新かつ緊急の代用として機能していても、問合せを実行できる。

「**読取り専用モード（read-only mode）**」も参照。

読取り専用モード（read-only mode）

フィジカル・スタンバイ・データベースのモードの 1 つ。次の SQL 文を発行することにより開始される。

```
ALTER DATABASE OPEN READ ONLY;
```

このモードでは、フィジカル・スタンバイ・データベースの問合せはできるが、変更はできない。

「**読取り専用データベース（read-only database）**」も参照。

リカバリ・インスタンス（recovery instance）

管理リカバリが実行されるノード。Real Application Clusters 構成では、各プライマリ・インスタンスが、そのアーカイブ REDO ログをスタンバイ・クラスタのこのノードへ方向指定する。

「**インスタンス間アーカイブ環境（cross-instance archival environment）**」および「**受信インスタンス（receiving instance）**」も参照。

リカバリ・カタログ（recovery catalog）

Recovery Manager が Oracle データベースに関する情報を格納するために使用する、表とビューのセット。Recovery Manager は、このデータを使用して Oracle データベースのバックアップ、リストアおよびリカバリを管理する。リカバリ・カタログを使用しない場合、Recovery Manager はターゲット・データベースの制御ファイルを使用する。リカバリ・カタログは、ターゲット・データベースに格納しないこと。

「**リカバリ・カタログ・データベース（recovery catalog database）**」および「**Recovery Manager**」も参照。

リカバリ・カタログ・データベース（recovery catalog database）

リカバリ・カタログのスキーマが格納されている Oracle データベース。

「**リカバリ・カタログ（recovery catalog）**」も参照。

リスナー（listener）

クライアントによる要求を受信し、適切なサーバーにリダイレクトするアプリケーション。

リモート・ファイル・サーバー（Remote File Server: RFS）

スタンバイ・ロケーションにあるリモート・ファイル・サーバー・プロセスは、プライマリ・データベースからアーカイブ REDO ログを受信する。

ローリング・アップグレード（rolling upgrade）

ソフトウェアのインストール・テクニックの 1 つ。このテクニックにより、クラスタ化されたシステムは、ソフトウェアが次のリリースにアップグレードされている間もサービスの提供を続行できる。このプロセスがローリング・アップグレードと呼ばれるのは、すべてのデータベースまたはシステムがアップグレードされるまで、クラスタ内の各データベースまたはシステムが順々にアップグレードされリブートされるためである。

ロール管理サービス（role management service）

データベース・ロールの変更に関する役割を担う、Data Guard 環境のコンポーネント。データベース・ロールの推移には、予定外の停止のためにプライマリ・データベースが使用不可能な場合は、スイッチオーバーとフェイルオーバーが含まれる。

「**ログ適用サービス（log apply services）**」および「**ログ転送サービス（log transport services）**」も参照。

ロールの推移（role transition）

データベースは、プライマリまたはスタンバイのいずれかのロールになる。この 2 つのロールは相互に排他的である。これらのロールは、計画された推移として動的に変更される場合（スイッチオーバー）と、予定外のデータベース障害の結果として変更される場合（フェイルオーバー）がある。

「**フェイルオーバー（failover）**」および「**スイッチオーバー（switchover）**」も参照。

ロギング（logging）

「**完全なサプリメンタル・ロギング（full supplemental logging）**」および「**サプリメンタル・ロギング（supplemental logging）**」を参照。

ログ・スイッチ（log switch）

LGWR がアクティブな REDO ログへの書込みを停止し、使用可能な次の REDO ログに切り替えるポイント。LGWR が切替えを実行するのは、アクティブなログが REDO レコードでいっぱいになったとき、または切替えを手動で強制されたときである。

データベースを ARCHIVELOG モードで実行する場合、ログ転送サービスは、アクティブでないログの REDO データをアーカイブ REDO ログにアーカイブする。ログ・スイッチが発生し、LGWR が古い REDO データに上書きを開始しても、アーカイブ REDO ログにはその古いデータが含まれるため、データが失われることはない。NOARCHIVELOG モードで実行する場合、ログ転送サービスは、ログ・スイッチの時点で古い REDO データをアーカイブせずに上書きする。したがって、古い REDO データはすべて失われる。

「**REDO ログ（redo log）**」も参照。

ログ適用サービス（log apply services）

Data Guard 環境のコンポーネント。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するために、スタンバイ・データベース上でアーカイブ REDO ログを適用する役割を担う。

「**ログ転送サービス（log transport services）**」、「**ロール管理サービス（role management service）**」および「**SQL 適用モード（SQL apply mode）**」も参照。

ログ転送サービス（log transport services）

Data Guard 環境のコンポーネント。プライマリ・データベースのオンライン REDO データの転送を自動化する役割を担う。ログ転送サービスは、アーカイブ REDO ログの許可、宛先、転送、受信および障害の解決を管理する。Data Guard 環境では、ログ転送サービスはログ適用サービスと共調しながら動作する。

「**ログ適用サービス（log apply services）**」、「**非データ消失（no data loss）**」および「**ロール管理サービス（role management service）**」も参照。

ログ・ライター・プロセス（Log Writer Process: LGWR）

トランザクション REDO を収集し、オンライン REDO ログを更新するバックグラウンド・プロセス。また、ログ・ライター・プロセスはローカルのアーカイブ REDO ログを作成したり、オンライン REDO ログをスタンバイ・データベースに転送できる。

ロジカル・スタンバイ・データベース（logical standby database）

プライマリ・データベースと論理的に同一で、プライマリ・データベースがオフラインになった場合に処理を引き継ぐスタンバイ・データベース。ロジカル・スタンバイ・データベースは、プライマリ・データベースと論理的に等しいため、同じスキーマ定義を共有する。

「**フィジカル・スタンバイ・データベース（physical standby database）**」および「**スタンバイ・データベース（standby database）**」も参照。

ロジカル・スタンバイ・プロセス（logical standby process: LSP）

LSP は、アーカイブ REDO ログ情報をロジカル・スタンバイ・データベースに適用する。

索引

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

あ

い

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

ね

は

ひ

ふ

へ

ほ

ま

む

め

ゆ

よ

り

れ

ろ