

Oracle® Data Guard

概要および管理

19c

F16122-04(原本部品番号:E96244-04)

2021年2月

タイトルおよび著作権情報

Oracle Data Guard概要および管理, 19c

F16122-04

Copyright © 1999, 2021, Oracle and/or its affiliates.

原著者: Padmaja Potineni, Kathy Rich

原協力者: Andy Adams, Beldalker Anand, Chipper Brown, Larry Carpenter, Jin-Jwei Chen, Laurence Clarke, Jeff Detjen, Ray Dutcher, David Gagne, B.G. Garin, Mahesh Girkar, Yuhong Gu, Joydip Kundu, Steven Lee, Steven Lim, Nitin Karkhanis, Goutam Kulkarni, Jonghyun Lee, Yunrui Li, Shashi Mangalat, Steven McGee, Bob McGuirk, Joe Meeks, Steve Moriarty, Muthu Olagappan, Ashish Ray, Mike Schloss, Mike Smith, Lawrence To, Stephen Vivian, Doug Voss, Hongjie Yang

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリバース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。

ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクル社までご連絡ください。

このソフトウェアまたは関連ドキュメントを、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供する場合は、次の通知が適用されます。

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

このソフトウェアもしくはハードウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアもしくはハードウェアは、危険が伴うアプリケーション(人的傷害を発生させる可能性があるアプリケーションを含む)への用途を目的として開発されていません。このソフトウェアもしくはハードウェアを危険が伴うアプリケーションで使用する際、安全に使用するために、適切な安全装置、バックアップ、冗長性(redundancy)、その他の対策を講じることは使用者の責任となります。このソフトウェアもしくはハードウェアを危険が伴うアプリケーションで使用したことに起因して損害が発生しても、オラクル社およびその関連会社は一切の責任を負いかねます。

OracleおよびJavaはOracle Corporationおよびその関連企業の登録商標です。その他の名称は、それぞれの所有者の商標または登録商標です。

IntelおよびIntel Insideは、Intel Corporationの商標または登録商標です。すべてのSPARCの商標はライセンスをもとに使用し、SPARC International, Inc.の商標または登録商標です。AMD、Epyc、AMDロゴは、Advanced Micro Devices, Inc.の商標または登録商標です。UNIXは、The Open Groupの登録商標です。

このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。お客様との間に適切な契約が定められている場合を除いて、オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。お客様との間に適切な契約が定められている場合を除いて、オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

目次

- [例一覧](#)
- [図一覧](#)
- [表一覧](#)
- [タイトルおよび著作権情報](#)
- [はじめに](#)
 - [対象読者](#)
 - [ドキュメントのアクセシビリティについて](#)
 - [関連ドキュメント](#)
 - [表記規則](#)
- [『Oracle Data Guard概要および管理』のこのリリースでの変更点](#)
 - [Oracle Databaseリリース19cにおける変更点](#)
 - [Oracle Databaseリリース18c \(バージョン18.1\)での変更点](#)
- [第I部 概要および管理](#)
 - [1 Oracle Data Guardの概要](#)
 - [1.1 Oracle Data Guardの構成](#)
 - [1.1.1 プライマリ・データベース](#)
 - [1.1.2 スタンバイ・データベース](#)
 - [1.1.3 遠隔同期インスタンス](#)
 - [1.1.4 Zero Data Loss Recovery Appliance](#)
 - [1.1.5 構成例](#)
 - [1.2 Oracle Data Guardのサービス](#)
 - [1.2.1 REDO転送サービス](#)
 - [1.2.2 適用サービス](#)
 - [1.2.3 ロールの推移](#)
 - [1.3 Oracle Data Guard Broker](#)
 - [1.3.1 Oracle Enterprise Manager Cloud Controlの使用](#)
 - [1.3.2 Oracle Data Guardコマンドライン・インタフェースの使用](#)
 - [1.4 Oracle Data Guardの保護モード](#)
 - [1.5 クライアント・フェイルオーバー](#)
 - [1.5.1 アプリケーション・コンティニューイティ](#)
 - [1.6 Oracle Data Guardと補完テクノロジー](#)
 - [1.7 Oracle Active Data GuardのOracle Shardingサポート](#)
 - [1.8 Oracle Data Guardのメリットの要約](#)
 - [2 Oracle Data Guardの開始](#)
 - [2.1 スタンバイ・データベースのタイプ](#)
 - [2.1.1 フィジカル・スタンバイ・データベース](#)
 - [2.1.2 ロジカル・スタンバイ・データベース](#)
 - [2.1.3 スナップショット・スタンバイ・データベース](#)
 - [2.2 Oracle Data Guard構成の管理のためのユーザー・インタフェース](#)
 - [2.3 Oracle Data Guardの動作要件](#)
 - [2.3.1 ハードウェアおよびオペレーティング・システムの要件](#)
 - [2.3.2 Oracleソフトウェア要件](#)

- [2.4 スタンバイ・データベースのディレクトリ構造に関する考慮事項](#)
 - [2.5 オンライン・データファイルの場所の移行](#)
 - [2.5.1 オンライン・データ・ファイルの場所の移行時の制限事項](#)
- [3 フィジカル・スタンバイ・データベースの作成](#)
 - [3.1 スタンバイ・データベースを作成するためのプライマリ・データベースの準備](#)
 - [3.1.1 適切なロギング・モードの有効化](#)
 - [3.1.2 REDO転送の認証の構成](#)
 - [3.1.3 REDOデータを受信するためのプライマリ・データベースの構成](#)
 - [3.1.4 プライマリ・データベースの初期化パラメータの設定](#)
 - [3.1.5 アーカイブの有効化](#)
 - [3.2 フィジカル・スタンバイ・データベースの作成ステップ](#)
 - [3.2.1 フィジカル・スタンバイの作成 タスク1: プライマリ・データベース・データファイルのバックアップ・コピーの作成](#)
 - [3.2.2 フィジカル・スタンバイの作成 タスク2: スタンバイ・データベース用の制御ファイルの作成](#)
 - [3.2.3 フィジカル・スタンバイの作成 タスク3: スタンバイ・データベース用のパラメータ・ファイルの作成](#)
 - [3.2.4 フィジカル・スタンバイの作成 タスク4: プライマリ・システムからスタンバイ・システムへのファイルのコピー](#)
 - [3.2.5 フィジカル・スタンバイの作成 タスク5: スタンバイ・データベースをサポートする環境の設定](#)
 - [3.2.6 フィジカル・スタンバイの作成 タスク6: フィジカル・スタンバイ・データベースの起動](#)
 - [3.2.7 フィジカル・スタンバイの作成 タスク7: フィジカル・スタンバイ・データベースが正しく実行されているかどうかの確認](#)
 - [3.3 フィジカル・スタンバイの作成: 作成後のステップ](#)
 - [3.4 DBCAを使用したData Guardスタンバイの作成](#)
 - [3.5 CDBのフィジカル・スタンバイの作成](#)
 - [3.6 プライマリ・データベースでのPDBの作成](#)
- [4 ロジカル・スタンバイ・データベースの作成](#)
 - [4.1 ロジカル・スタンバイ・データベースの作成要件](#)
 - [4.1.1 表のデータ型および記憶域属性のサポートの判別](#)
 - [4.1.2 プライマリ・データベース内の表の行が一意に識別できることの確認](#)
 - [4.2 ロジカル・スタンバイ・データベースの作成ステップ](#)
 - [4.2.1 ロジカル・スタンバイの作成 タスク1: フィジカル・スタンバイ・データベースの作成](#)
 - [4.2.2 ロジカル・スタンバイの作成 タスク2: フィジカル・スタンバイ・データベースでのREDO Applyの停止](#)
 - [4.2.3 ロジカル・スタンバイの作成 タスク3: ロジカル・スタンバイ・データベースをサポートするためのプライマリ・データベースの準備](#)
 - [4.2.3.1 ロールの推移のためのプライマリ・データベースの準備](#)
 - [4.2.3.2 REDOデータでのディクショナリの構築](#)
 - [4.2.4 ロジカル・スタンバイの作成 タスク4: ロジカル・スタンバイ・データベースへの推移](#)
 - [4.2.4.1 ロジカル・スタンバイ・データベースへの変換](#)
 - [4.2.4.2 ロジカル・スタンバイ・データベース用の初期化パラメータの調整](#)
 - [4.2.5 ロジカル・スタンバイの作成 タスク5: ロジカル・スタンバイ・データベースのオープン](#)
 - [4.2.6 ロジカル・スタンバイの作成 タスク6: ロジカル・スタンバイ・データベースが正しく実行](#)

されているかどうかの確認

- 4.3 ロジカル・スタンバイの作成: 作成後のステップ
- 4.4 CDBのロジカル・スタンバイの作成
- 5 遠隔同期インスタンスの使用
 - 5.1 遠隔同期インスタンスの作成
 - 5.1.1 遠隔同期インスタンスの作成および構成
 - 5.2 代替宛先
 - 5.2.1 グループへのログ・アーカイブ先の割当て
 - 5.2.2 グループ内のログ・アーカイブ先に対する優先順位の割当て
 - 5.2.3 グループ内の複数のアクティブな宛先への送信
 - 5.2.4 複数のログ・アーカイブ先グループの使用
 - 5.2.5 ログ・アーカイブ先の可用性ステータスの判断
 - 5.3 代替アーカイブ先の構成
 - 5.3.1 遠隔同期障害後の保護の低下
 - 5.3.2 遠隔同期インスタンスの高可用性
 - 5.3.3 ロール変更後の保護の維持
 - 5.4 遠隔同期インスタンスでサポートされる保護モード
 - 5.4.1 最大可用性モード構成での遠隔同期インスタンス
 - 5.4.2 最大パフォーマンス・モード構成での遠隔同期インスタンス
- 6 Oracle Data Guardの保護モード
 - 6.1 Oracle Data Guardの保護モード
 - 6.2 プライマリ・データベースのデータ保護モードの設定
- 7 REDO転送サービス
 - 7.1 REDO転送サービスの概要
 - 7.2 REDO転送サービスの構成
 - 7.2.1 REDO転送のセキュリティ
 - 7.2.1.1 SSLを使用したREDO転送の認証
 - 7.2.1.2 パスワード・ファイルを使用したREDO転送の認証
 - 7.2.2 REDOデータを送信するためのOracleデータベースの構成
 - 7.2.2.1 V\$ARCHIVE_DESTビューを使用した属性の表示
 - 7.2.3 REDOデータを受信するためのOracleデータベースの構成
 - 7.2.3.1 スタンバイREDOログの管理
 - 7.2.3.2 REDOがアーカイブREDOログ・ファイルに直接書き込まれる場合
 - 7.3 カスケードされたREDO転送先
 - 7.3.1 ターミナル宛先の構成
 - 7.3.2 カスケードのシナリオ
 - 7.3.2.1 フィジカル・スタンバイへのカスケード
 - 7.3.2.2 複数のフィジカル・スタンバイへのカスケード
 - 7.4 カスケード・スタンバイでのデータ保護の考慮点
 - 7.5 構成の検証
 - 7.6 REDO転送サービスの監視
 - 7.6.1 REDO転送ステータスの監視
 - 7.6.2 同期REDO転送のレスポンス時間の監視
 - 7.6.3 REDOギャップの検出および解決
 - 7.6.3.1 手動によるギャップの解決

- [7.6.4 REDO転送サービスの待機イベント](#)
 - [7.7 REDO転送のチューニング](#)
- [8 適用サービス](#)
 - [8.1 適用サービスの概要](#)
 - [8.2 適用サービスの構成オプション](#)
 - [8.2.1 リアルタイム適用によるREDOデータの即時適用](#)
 - [8.2.2 アーカイブREDOログ・ファイルの適用に対する遅延時間の指定](#)
 - [8.2.2.1 遅延時間設定の代替策としてのフラッシュバック・データベースの使用](#)
 - [8.3 REDOデータのフィジカル・スタンバイ・データベースへの適用](#)
 - [8.3.1 REDO Applyの開始](#)
 - [8.3.2 REDO Applyの停止](#)
 - [8.3.3 フィジカル・スタンバイ・データベースでのREDO Applyの監視](#)
 - [8.4 REDOデータのロジカル・スタンバイ・データベースへの適用](#)
 - [8.4.1 SQL Applyの開始](#)
 - [8.4.2 ロジカル・スタンバイ・データベースでのSQL Applyの停止](#)
 - [8.4.3 ロジカル・スタンバイ・データベースでのSQL Applyの監視](#)
 - [8.5 プライマリでのPDBの削除または名前の変更時のスタンバイの考慮点](#)
- [9 ロールの推移](#)
 - [9.1 ロールの推移の概要](#)
 - [9.1.1 ロールの推移の準備](#)
 - [9.1.2 ロールの推移のターゲット・スタンバイ・データベースの選択](#)
 - [9.1.3 スイッチオーバー](#)
 - [9.1.4 フェイルオーバー](#)
 - [9.1.5 ロールの推移のトリガー](#)
 - [9.2 フィジカル・スタンバイ・データベースが関与するロールの推移](#)
 - [9.2.1 フィジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)
 - [9.2.2 フィジカル・スタンバイ・データベースへのフェイルオーバーの実行](#)
 - [9.3 ロジカル・スタンバイ・データベースが関与するロールの推移](#)
 - [9.3.1 ロジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)
 - [9.3.2 ロジカル・スタンバイ・データベースへのフェイルオーバーの実行](#)
 - [9.4 ロールの推移後のフラッシュバック・データベースの使用](#)
 - [9.4.1 スイッチオーバー後のフラッシュバック・データベースの使用](#)
 - [9.4.2 フェイルオーバー後のフラッシュバック・データベースの使用](#)
- [10 フィジカルおよびスナップショット・スタンバイ・データベースの管理](#)
 - [10.1 フィジカル・スタンバイ・データベースの起動と停止](#)
 - [10.1.1 フィジカル・スタンバイ・データベースの起動](#)
 - [10.1.2 フィジカル・スタンバイ・データベースの停止](#)
 - [10.2 フィジカル・スタンバイ・データベースのオープン](#)
 - [10.2.1 リアルタイム問合せ](#)
 - [10.2.1.1 リアルタイム問合せ環境での適用ラグの監視](#)
 - [10.2.1.2 リアルタイム問合せ環境での適用ラグ許容差の構成](#)
 - [10.2.1.3 リアルタイム問合せ環境でのREDO Applyの強制同期](#)
 - [10.2.1.4 リアルタイム問合せの制限](#)
 - [10.2.1.5 自動ブロック・メディア・リカバリ](#)
 - [10.2.1.6 手動ブロック・メディア・リカバリ](#)

- [10.2.1.7 フィジカル・スタンバイ・データベースでの問合せのチューニング](#)
 - [10.2.1.8 フィジカル・スタンバイへの一時ファイルの追加](#)
- [10.2.2 Active Data GuardスタンバイでのSQLおよびPL/SQLの使用](#)
 - [10.2.2.1 Active Data Guardスタンバイ・データベースでのDML操作の実行](#)
 - [10.2.2.2 Active Data Guardスタンバイ・データベースでの最上位のPL/SQL操作の実行](#)
 - [10.2.2.3 変更されたPL/SQLオブジェクトの自動再コンパイル](#)
- [10.2.3 Active Data Guardインスタンスでの一時表の使用](#)
 - [10.2.3.1 Active Data Guardインスタンス上のグローバル一時表](#)
 - [10.2.3.2 Active Data Guardインスタンス上のプライベート一時表](#)
- [10.2.4 Active Data Guardスタンバイ環境でのIM列ストア](#)
- [10.2.5 Active Data Guard環境でのインメモリ外部表](#)
- [10.2.6 Oracle Active Data Guardでの順序の使用](#)
 - [10.2.6.1 セッション順序](#)
- [10.3 フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更](#)
 - [10.3.1 データファイルの追加または表領域の作成](#)
 - [10.3.2 表領域の削除とデータファイルの削除](#)
 - [10.3.2.1 DROP TABLESPACE INCLUDING CONTENTS AND DATAFILESの使用](#)
 - [10.3.3 フィジカル・スタンバイ・データベースでのトランSPORTABLE表領域の使用](#)
 - [10.3.4 プライマリ・データベースのデータファイルの改名](#)
 - [10.3.5 REDOログ・ファイル・グループの追加または削除](#)
 - [10.3.6 NOLOGGINGまたはリカバリ不能な操作](#)
 - [10.3.7 パスワード・ファイルのリフレッシュ](#)
 - [10.3.8 TDEマスター暗号化キーのリセット](#)
- [10.4 OPEN RESETLOGS文を使用したりリカバリ](#)
- [10.5 プライマリRESETLOGS操作の後のマウント済スタンバイの自動フラッシュバック](#)
- [10.6 プライマリ、フィジカル・スタンバイおよびスナップショット・スタンバイ・データベースの監視](#)
 - [10.6.1 プライマリ、フィジカルおよびスナップショット・スタンバイ・データベースを監視するためのビューの使用](#)
 - [10.6.1.1 V\\$DATABASE](#)
 - [10.6.1.2 V\\$DATAGUARD_PROCESS](#)
 - [10.6.1.3 V\\$MANAGED_STANDBY](#)
 - [10.6.1.4 V\\$ARCHIVED_LOG](#)
 - [10.6.1.5 V\\$LOG_HISTORY](#)
 - [10.6.1.6 V\\$DATAGUARD_STATUS](#)
 - [10.6.1.7 V\\$ARCHIVE_DEST](#)
- [10.7 プライマリからスタンバイへのリストア・ポイントのレプリケート](#)
- [10.8 REDO Applyのチューニング](#)
- [10.9 SQLチューニング・アドバイザによるActive Data Guard環境でのデータベースのチューニング](#)
- [10.10 Oracle Diagnostic Packを使用したOracle Active Data Guardのスタンバイのチューニング](#)
- [10.11 スナップショット・スタンバイ・データベースの管理](#)
 - [10.11.1 スナップショット・スタンバイ・データベースへのフィジカル・スタンバイ・データベースの変換](#)

- [10.11.2 スナップショット・スタンバイ・データベースの使用](#)
 - [10.11.3 フィジカル・スタンバイ・データベースへのスナップショット・スタンバイ・データベースの変換](#)
- [11 ロジカル・スタンバイ・データベースの管理](#)
 - [11.1 SQL Applyアーキテクチャの概要](#)
 - [11.1.1 SQL Applyに関する各種の考慮事項](#)
 - [11.1.1.1 トランザクション・サイズの考慮事項](#)
 - [11.1.1.2 ページアウトの考慮事項](#)
 - [11.1.1.3 再起動の考慮事項](#)
 - [11.1.1.4 DML適用の考慮事項](#)
 - [11.1.1.5 DDL適用の考慮事項](#)
 - [11.1.1.6 パスワード検証関数](#)
 - [11.2 ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスの制御](#)
 - [11.3 ロジカル・スタンバイ・データベースの管理および監視関連のビュー](#)
 - [11.3.1 DBA_LOGSTDBY_EVENTSビュー](#)
 - [11.3.2 DBA_LOGSTDBY_LOGビュー](#)
 - [11.3.3 V\\$DATAGUARD_STATSビュー](#)
 - [11.3.4 V\\$LOGSTDBY_PROCESSビュー](#)
 - [11.3.5 V\\$LOGSTDBY_PROGRESSビュー](#)
 - [11.3.6 V\\$LOGSTDBY_STATEビュー](#)
 - [11.3.7 V\\$LOGSTDBY_STATSビュー](#)
 - [11.4 ロジカル・スタンバイ・データベースの監視](#)
 - [11.4.1 SQL Applyの進捗の監視](#)
 - [11.4.2 ログ・ファイルの自動削除](#)
 - [11.5 ロジカル・スタンバイ・データベースのカスタマイズ](#)
 - [11.5.1 DBA_LOGSTDBY_EVENTSビューでのイベントのロギングのカスタマイズ](#)
 - [11.5.2 DBMS_LOGSTDBY.SKIPによる特定のスキーマ・オブジェクトに対する変更の防止](#)
 - [11.5.3 DDL文のスキップ・ハンドラの設定](#)
 - [11.5.4 ロジカル・スタンバイ・データベースの変更](#)
 - [11.5.4.1 ロジカル・スタンバイ・データベースでのDDLの実行](#)
 - [11.5.4.2 SQL Applyでメンテナンスされていない表の変更](#)
 - [11.5.5 ロジカル・スタンバイ・データベースでの表の追加または再作成](#)
 - [11.6 ロジカル・スタンバイ・データベースの下での特定のワークロードの管理](#)
 - [11.6.1 プライマリ・データベースへのトランSPORTABLE表領域のインポート](#)
 - [11.6.2 マテリアライズド・ビューの使用](#)
 - [11.6.3 ロジカル・スタンバイ・データベースでのトリガーと制約の処理方法](#)
 - [11.6.4 サポートされていない表のレプリケートでのトリガーの使用](#)
 - [11.6.5 プライマリでのPoint-in-Timeリカバリの実行によるリカバリ](#)
 - [11.6.6 ロジカル・スタンバイ・データベースでのOracle Streams取得プロセスの実行](#)
 - [11.7 ロジカル・スタンバイ・データベースのチューニング](#)
 - [11.7.1 主キー-RELY制約の作成](#)
 - [11.7.2 コストベース・オブティマイザの統計の収集](#)
 - [11.7.3 プロセス数の調整](#)
 - [11.7.3.1 APPLIERプロセス数の調整](#)

- [11.7.3.2 PREPARERプロセス数の調整](#)
 - [11.7.4 LCRキャッシュ用メモリの調整](#)
 - [11.7.5 ロジカル・スタンバイ・データベースにおけるトランザクションの適用方法の調整](#)
 - [11.8 ロジカル・スタンバイ・データベースの下でのバックアップおよびリカバリ](#)
- [12 RMANを使用したファイルのバックアップおよびリストア](#)
 - [12.1 Oracle Data Guard構成でのRMANファイル管理について](#)
 - [12.1.1 Oracle Data Guard環境でのバックアップの互換性](#)
 - [12.1.2 Oracle Data Guard環境でのバックアップの関連付け](#)
 - [12.1.3 Oracle Data Guard環境でのバックアップのアクセシビリティについて](#)
 - [12.2 Oracle Data Guard環境でのRMAN構成について](#)
 - [12.3 推奨されるRMAN構成およびOracle Database構成](#)
 - [12.3.1 プライマリおよびスタンバイ・データベースでのOracle Database構成](#)
 - [12.3.2 プライマリ・データベースでのRMAN構成](#)
 - [12.3.3 バックアップが実行されるスタンバイ・データベースでのRMAN構成](#)
 - [12.3.4 バックアップが実行されないスタンバイでのRMAN構成](#)
 - [12.4 バックアップ手順](#)
 - [12.4.1 テープ・バックアップ用キャッシュとしてのディスクの使用](#)
 - [12.4.1.1 ディスクをキャッシュとして使用する日次テープ・バックアップ用のコマンド](#)
 - [12.4.1.2 ディスクをキャッシュとして使用する週次テープ・バックアップ用のコマンド](#)
 - [12.4.2 テープへの直接バックアップの実行](#)
 - [12.4.2.1 テープに直接実行する日次バックアップ用のコマンド](#)
 - [12.4.2.2 テープに直接実行する週次バックアップ用のコマンド](#)
 - [12.5 Oracle Data Guard環境でのデータベースの登録および登録解除](#)
 - [12.6 Oracle Data Guard環境におけるレポート生成](#)
 - [12.7 Oracle Data Guard環境におけるバックアップ・メンテナンスの実行](#)
 - [12.7.1 リカバリ・カタログのメタデータの変更](#)
 - [12.7.2 アーカイブ・ログまたはバックアップの削除](#)
 - [12.7.3 リカバリ・カタログのメタデータの検証](#)
 - [12.8 Oracle Data Guard環境におけるリカバリ例](#)
 - [12.8.1 プライマリまたはスタンバイ・データベースでのファイルの損失からのリカバリ](#)
 - [12.8.2 オンラインREDOログ・ファイルの損失からのリカバリ](#)
 - [12.8.3 プライマリ・データベースの不完全リカバリ](#)
 - [12.8.4 プライマリでのTSPITRまたは表領域プラグイン後のスタンバイで必要なアクション](#)
 - [12.9 その他のバックアップ状況](#)
 - [12.9.1 スタンバイ・データベースが地理的に離れすぎているためにバックアップを共有できない場合](#)
 - [12.9.2 FALサーバーとして使用されるスタンバイ・データベースにデータファイルが含まれていない場合](#)
 - [12.9.3 スタンバイ・データベースのファイル名がプライマリ・データベースとは異なる場合](#)
 - [12.10 ネットワークを介したファイルのリストアおよびリカバリ](#)
 - [12.11 1つコマンドを使用したスタンバイのロールフォワード](#)
 - [12.12 Oracle Data Guard環境でのCDBに対するRMANサポート](#)
- [13 SQL Applyを使用したOracle Databaseのアップグレード](#)
 - [13.1 SQL Applyを使用するローリング・アップグレードのメリット](#)
 - [13.2 SQL Applyを使用してローリング・アップグレードを実行するための要件](#)

- [13.3 アップグレード手順に使用する図と表記規則](#)
 - [13.4 ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行](#)
 - [13.5 既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行](#)
 - [13.6 既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行](#)
- [14 DBMS_ROLLINGを使用したローリング・アップグレードの実行](#)
 - [14.1 ローリング・アップグレードの新しい概念](#)
 - [14.1.1 Data GuardブローカでのDBMS_ROLLINGアップグレードのサポート](#)
 - [14.2 DBMS_ROLLINGアップグレードとCDB](#)
 - [14.3 DBMS_ROLLINGの使用の概要](#)
 - [14.4 ローリング・アップグレードの計画](#)
 - [14.5 ローリング・アップグレードの実行](#)
 - [14.6 ローリング・アップグレードの監視](#)
 - [14.7 ローリング・アップグレードのロールバック](#)
 - [14.8 ローリング・アップグレード中に発生したロール変更の処理](#)
 - [14.9 ローリング・アップグレードの例](#)
- [15 Oracle Data Guardの使用例](#)
 - [15.1 フェイルオーバー後のロジカル・スタンバイ・データベースの構成](#)
 - [15.1.1 新規プライマリ・データベースがフィジカル・スタンバイ・データベースだった場合](#)
 - [15.1.2 新規プライマリ・データベースがロジカル・スタンバイ・データベースだった場合](#)
 - [15.2 フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換](#)
 - [15.2.1 障害が発生したプライマリ・データベースのフィジカル・スタンバイ・データベースへのフラッシュバック](#)
 - [15.2.2 障害が発生したプライマリ・データベースのロジカル・スタンバイ・データベースへのフラッシュバック](#)
 - [15.2.3 特定の適用済SCNへのロジカル・スタンバイ・データベースのフラッシュバック](#)
 - [15.3 Open Resetlogs文の発行後のフラッシュバック・データベースの使用](#)
 - [15.3.1 特定時点へのフィジカル・スタンバイ・データベースのフラッシュバック](#)
 - [15.3.2 特定時点へのロジカル・スタンバイ・データベースのフラッシュバック](#)
 - [15.4 NOLOGGING句を指定した後のリカバリ](#)
 - [15.4.1 ロジカル・スタンバイ・データベースのリカバリ・ステップ](#)
 - [15.4.2 フィジカル・スタンバイ・データベースのリカバリ・ステップ](#)
 - [15.4.3 リカバリ不能処理後にバックアップが必要かどうかの判断](#)
 - [15.4.4 フィジカル・スタンバイ・データベースの一部分のリカバリ・ステップ](#)
 - [15.5 OMFまたはOracle ASMを使用するスタンバイ・データベースの作成](#)
 - [15.6 プライマリ・データベースでの書込みの欠落エラーからのリカバリ](#)
 - [15.7 DBCOMPプロシージャを使用した欠落書込みおよびその他の不一致の検出](#)
 - [15.8 RMANバックアップを使用した障害が発生したプライマリのスタンバイ・データベースへの変換](#)
 - [15.8.1 RMANバックアップを使用した、障害が発生したプライマリのフィジカル・スタンバイへの変換](#)
 - [15.8.2 RMANバックアップを使用した障害が発生したプライマリのロジカル・スタンバイへの変換](#)
 - [15.9 フィジカル・スタンバイの再作成を伴わないプライマリの文字セットの変更](#)
 - [15.10 プライマリでのPDB PITRまたはPDBフラッシュバック後にスタンバイに必要なアクション](#)
- [第II部 リファレンス](#)

- [16 初期化パラメータ](#)
- [17 LOG_ARCHIVE_DEST_nパラメータの属性](#)
 - [17.1 AFFIRMおよびNOAFFIRM](#)
 - [17.2 ALTERNATE](#)
 - [17.3 COMPRESSION](#)
 - [17.4 DB_UNIQUE_NAME](#)
 - [17.5 DELAY](#)
 - [17.6 ENCRYPTION](#)
 - [17.7 GROUP](#)
 - [17.8 LOCATIONおよびSERVICE](#)
 - [17.9 MANDATORY](#)
 - [17.10 MAX_FAILURE](#)
 - [17.11 NET_TIMEOUT](#)
 - [17.12 NOREGISTER](#)
 - [17.13 PRIORITY](#)
 - [17.14 REOPEN](#)
 - [17.15 SYNCおよびASYNC](#)
 - [17.16 TEMPLATE](#)
 - [17.17 VALID_FOR](#)
- [18 Oracle Data Guardに関連するSQL文](#)
 - [18.1 ALTER DATABASE文](#)
 - [18.2 ALTER SESSION文](#)
 - [18.3 ALTER SYSTEM文](#)
- [19 Oracle Data Guardに関連するビュー](#)
- [付録](#)
 - [A Oracle Data Guardのトラブルシューティング](#)
 - [A.1 一般的な問題](#)
 - [A.1.1 ALTER DATABASE文を使用したデータファイルの改名](#)
 - [A.1.2 スタンバイ・データベースがプライマリ・データベースからREDOデータを受信しない](#)
 - [A.1.3 フィジカル・スタンバイ・データベースをマウントできない](#)
 - [A.2 ログ・ファイル宛先の障害](#)
 - [A.3 ロジカル・スタンバイ・データベース障害の処理](#)
 - [A.4 フィジカル・スタンバイ・データベースへのスイッチオーバーの問題](#)
 - [A.4.1 REDOデータが転送されていないためスイッチオーバーできない](#)
 - [A.4.2 ORA-01102エラーによりスイッチオーバーできない](#)
 - [A.4.3 スwitchオーバー後にREDOデータが適用されない](#)
 - [A.4.4 失敗したスイッチオーバーをロールバックして最初からやりなおす](#)
 - [A.5 ロジカル・スタンバイ・データベースへのスイッチオーバーの問題](#)
 - [A.5.1 スwitchオーバー操作の準備フェーズ中の障害](#)
 - [A.5.1.1 プライマリ・データベースの準備中の障害](#)
 - [A.5.1.2 ロジカル・スタンバイ・データベースの準備中の障害](#)
 - [A.5.2 スwitchオーバー操作のコミット・フェーズ中の障害](#)
 - [A.5.2.1 元のプライマリ・データベースの変換での障害](#)
 - [A.5.2.2 ターゲット・ロジカル・スタンバイ・データベースの変換での障害](#)
 - [A.6 SQL Applyが停止した場合の処置](#)

- [A.7 REDOデータ転送のネットワーク調整](#)
- [A.8 スタンバイ・データベースのディスクのパフォーマンスが遅い](#)
- [A.9 プライマリ・データベースの停止を回避するにはログ・ファイルを一致させる必要がある](#)
- [A.10 ロジカル・スタンバイ・データベースのトラブルシューティング](#)
 - [A.10.1 エラーのリカバリ](#)
 - [A.10.1.1 ファイル仕様が含まれているDDLトランザクション](#)
 - [A.10.1.2 DML障害のリカバリ](#)
 - [A.10.2 SQL*Loaderセッションのトラブルシューティング](#)
 - [A.10.3 長時間実行トランザクションのトラブルシューティング](#)
 - [A.10.4 フラッシュバック・トランザクションでのORA-1403エラーのトラブルシューティング](#)
- [B Oracle Data Guard構成におけるデータベースのパッチ適用、アップグレードおよびダウングレード](#)
 - [B.1 Oracle Databaseソフトウェアをパッチ適用またはアップグレードする前の注意事項](#)
 - [B.2 Standby-First Patchの適用を使用したOracle Databaseへのパッチ適用](#)
 - [B.3 フィジカル・スタンバイ・データベースが存在する場合のOracle Databaseのアップグレード](#)
 - [B.4 ロジカル・スタンバイ・データベースが存在する場合のOracle Databaseのアップグレード](#)
 - [B.5 アップグレード後のCOMPATIBLE初期化パラメータの変更](#)
 - [B.6 ロジカル・スタンバイが存在しない場合のOracle Databaseのダウングレード](#)
 - [B.7 ロジカル・スタンバイが存在する場合のOracle Databaseのダウングレード](#)
- [C ロジカル・スタンバイ・データベースでサポートされるデータ型およびDDL](#)
 - [C.1 データ型に関する考慮事項](#)
 - [C.1.1 ロジカル・スタンバイ・データベースでサポートされるデータ型](#)
 - [C.1.1.1 互換性要件](#)
 - [C.1.1.2 不透明型の制限](#)
 - [C.1.2 ロジカル・スタンバイ・データベースでサポートされないデータ型](#)
 - [C.2 ネイティブREDOサポートが欠如したデータ型のサポート](#)
 - [C.3 透過的データ暗号化\(TDE\)のサポート](#)
 - [C.4 表領域の暗号化のサポート](#)
 - [C.5 行レベル・セキュリティおよびファイングレイン監査のサポート](#)
 - [C.5.1 行レベルのセキュリティ](#)
 - [C.5.2 ファイングレイン監査](#)
 - [C.5.3 PL/SQLレプリケーションのスキップおよび有効化](#)
 - [C.6 Oracle Label Security](#)
 - [C.7 Oracle Database Vault](#)
 - [C.8 Oracle E-Business Suite](#)
 - [C.9 サポートされる表記憶域タイプ](#)
 - [C.10 サポートされない表記憶域タイプ](#)
 - [C.10.1 パーティション化の結果としてサポートされない表](#)
 - [C.11 PL/SQLパッケージに関する考慮事項](#)
 - [C.11.1 サポートされるPL/SQLパッケージ](#)
 - [C.11.2 サポートされないPL/SQLパッケージ](#)
 - [C.11.2.1 DBMS_JOBのサポート](#)
 - [C.11.2.2 DBMS_SCHEDULERのサポート](#)
 - [C.11.3 ロジカル・スタンバイでのXMLおよびXDB PL/SQLパッケージの処理](#)
 - [C.11.3.1 DBMS_XMLSCHEMAスキーマ](#)
 - [C.11.3.2 DBMS_XMLINDEXパッケージ](#)

- [C.11.3.3 サポートされないPL/SQLプロシージャの処理](#)
 - [C.11.3.4 サポートされないPL/SQLの手動による補正](#)
 - [C.11.3.5 順序に依存するサポートされないPL/SQLの補正](#)
- [C.12 サポートされない表](#)
 - [C.12.1 ローリング・アップグレード中にサポートされない表](#)
 - [C.12.2 PL/SQLファンクションで実行されたDMLの結果としてのサポートされない表](#)
- [C.13 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](#)
- [C.14 ロジカル・スタンバイ・データベースでサポートされるDDL文](#)
 - [C.14.1 DBLINKSを使用するDDL文](#)
 - [C.14.2 ロジカル・スタンバイ・プロセス上のAUD\\$およびFGA_LOG\\$のレプリケーション](#)
- [C.15 分散トランザクションおよびXAのサポート](#)
- [C.16 SecureFile LOBのサポート](#)
- [C.17 データベース・ファイル・システム\(DBFS\)のサポート](#)
- [C.18 文字セットに関する考慮事項](#)
- [C.19 DBMS_ROLLINGアップグレードの場合でのみ使用可能な追加のPL/SQLパッケージのサポート](#)
- [D Oracle Data GuardおよびOracle Real Application Clusters](#)
 - [D.1 Oracle RAC環境でのスタンバイ・データベースの構成](#)
 - [D.1.1 複数インスタンスのREDO Applyの設定](#)
 - [D.1.2 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](#)
 - [D.1.3 Oracle RACプライマリおよびスタンバイ・データベースの設定](#)
 - [D.1.3.1 REDOデータを受信するためのOracle RACスタンバイ・データベースの構成](#)
 - [D.1.3.2 REDOデータを送信するためのOracle RACプライマリ・データベースの構成](#)
 - [D.2 Oracle RAC環境での構成に関する考慮事項](#)
 - [D.2.1 アーカイブREDOログ・ファイル名の形式](#)
 - [D.2.2 データ保護モード](#)
- [E Recovery Managerを使用したスタンバイ・データベースの作成](#)
 - [E.1 前提条件](#)
 - [E.2 RMANを使用したスタンバイ・データベースの作成の概要](#)
 - [E.2.1 RMANを使用したスタンバイ・データベースの作成の目的](#)
 - [E.2.2 RMANを使用したスタンバイ・データベースの作成の基本概念](#)
 - [E.2.2.1 アクティブ・データベース複製とバックアップベース複製](#)
 - [E.2.2.2 RMAN環境におけるDB_UNIQUE_NAMEの値](#)
 - [E.2.2.3 スタンバイ・データベースのリカバリ](#)
 - [E.2.2.3.1 スタンバイ・データベースREDOログ・ファイル](#)
 - [E.2.2.4 スタンバイ・データベースのパスワード・ファイル](#)
 - [E.3 DUPLICATEコマンドを使用したスタンバイ・データベースの作成](#)
 - [E.3.1 アクティブ・データベース複製を使用したスタンバイ・データベースまたは遠隔同期インスタンスの作成](#)
 - [E.3.2 バックアップベース複製によるスタンバイ・データベースの作成](#)
- [F アーカイブ・トレースの設定](#)
 - [F.1 LOG_ARCHIVE_TRACE初期化パラメータの設定](#)

- [G 古い構文を使用したロールの推移の実行](#)
 - [G.1 フィジカル・スタンバイが関与するロールの推移のSQL構文](#)
 - [G.1.1 古い構文を使用する場合の新しい機能](#)
 - [G.2 フィジカル・スタンバイ・データベースが関与するロールの推移](#)
 - [G.2.1 古い構文を使用したフィジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)
 - [G.2.2 古い構文を使用したフィジカル・スタンバイ・データベースへのフェイルオーバーの実行](#)
 - [G.3 フィジカル・スタンバイ・データベースへのスイッチオーバーのトラブルシューティング](#)
 - [G.3.1 REDOデータが転送されていないためスイッチオーバーできない](#)
 - [G.3.2 ORA-01102エラーによりスイッチオーバーできない](#)
 - [G.3.3 スwitchオーバー後にREDOデータが適用されない](#)
 - [G.3.4 失敗したスイッチオーバーをロールバックして最初からやりなおす](#)
- [H ALTERNATE属性を使用したりモート代替宛先の構成](#)
 - [H.1 代替宛先の構成](#)
- [索引](#)

例の一覧

- [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](#)
- [4-1 プライマリ・データベース: ロジカル・スタンバイ・ロールの初期化パラメータ](#)
- [4-2 ロジカル・スタンバイ・データベース用の初期化パラメータの変更](#)
- [5-1 遠隔同期インスタンスに使用される初期化パラメータの一部](#)
- [5-2 単一宛先フェイルオーバーに対する構成](#)
- [5-3 複数スタンバイ・データベースのREDO宛先フェイルオーバーに対する構成](#)
- [5-4 高可用性遠隔同期インスタンスの設定に使用されるパラメータ](#)
- [5-5 ロール変更後の保護設定に使用されるパラメータ](#)
- [7-1 REDOのカスケード時に使用される初期化パラメータの一部](#)
- [10-1 フィジカル・スタンバイ・データベースでのDML操作の実行](#)
- [10-2 変更されたPL/SQLオブジェクトの自動再コンパイル](#)
- [例11-1 ロジカル・スタンバイ・データベースのV\\$DATAGUARD_STATSの内容](#)
- [14-1 適用ラグ要件を強制するスイッチオーバーの設定](#)
- [14-2 ログのデフォルト値へのリセット](#)
- [14-3 データベースのオプション参加者としての指定](#)
- [14-4 一時ロジカル・スタンバイを保護するデータベースの設定](#)
- [14-5 基本ローリング・アップグレード・ステップ](#)
- [14-6 2つのデータベース間のローリング・アップグレード](#)
- [14-7 3つのデータベース間のローリング・アップグレード](#)
- [14-8 4つのデータベース間のローリング・アップグレード](#)
- [14-9 リーダー・ファームでのローリング・アップグレード](#)
- [14-10 Application Testingのローリング・アップグレード](#)
- [14-11 新しいプライマリへのフェイルオーバー後のローリング・アップグレードの再開](#)
- [14-12 新しい一時ロジカルへのフェイルオーバー後のローリング・アップグレードの再開](#)
- [15-1 プライマリおよびすべてのスタンバイがマウントまたはオープンされており、DBCOMPがプライマリから実行される場合](#)
- [15-2 プライマリおよびすべてのスタンバイがマウントまたはオープンされており、DBCOMPがスタンバイから実行される場合](#)
- [15-3 プライマリがマウントまたはオープンされているが、スタンバイはすべてがマウントまたはオープンされているとはかぎらず、DBCOMPがプライマリから実行される場合](#)
- [15-4 プライマリがマウントまたはオープンされているが、スタンバイはすべてがマウントまたはオープンされているとはかぎらず、DBCOMPがスタンバイから実行される場合](#)
- [15-5 プライマリがマウントされていないが、複数のスタンバイがマウントまたはオープンされている場合](#)
- [15-6 プライマリがマウントまたはオープンされているが、マウントまたはオープンされているスタンバイがない場合](#)
- [17-1 代替ローカル宛先への自動フェイルオーバー](#)
- [17-2 自動ローカル代替フォールバック](#)
- [A-1 再試行時間と制限の設定](#)
- [A-2 代替宛先の指定](#)
- [C-1 RegisterSchema用のPL/SQLスキップ・プロシージャ](#)

図一覧

- [1-1 一般的なOracle Data Guard構成](#)
- [1-2 フィジカル・スタンバイ・データベースの自動更新](#)
- [1-3 ロジカル・スタンバイ・データベースの自動更新](#)
- [1-4 システム管理シャーディングとOracle Data Guardレプリケーション](#)
- [1-5 複合シャーディングとOracle Data Guardレプリケーション](#)
- [2-1 可能なスタンバイ構成](#)
- [8-1 リアルタイム適用を使用したスタンバイ宛先へのREDOデータの適用](#)
- [9-1 スイッチオーバー前のOracle Data Guard構成](#)
- [9-2 新しいプライマリ・データベースへのスイッチオーバー前のスタンバイ・データベース](#)
- [9-3 スイッチオーバー後のOracle Data Guard環境](#)
- [9-4 スタンバイ・データベースへのフェイルオーバー](#)
- [11-1 SQL Applyの処理](#)
- [11-2 SQL Apply処理中の進捗状態](#)
- [13-1 アップグレード前のOracle Data Guard構成](#)
- [13-2 ロジカル・スタンバイ・データベースのリリースのアップグレード](#)
- [13-3 混合リリースの実行](#)
- [13-4 スイッチオーバー後](#)
- [13-5 両方のデータベースがアップグレードされた後](#)
- [D-1 複数インスタンス・プライマリ・データベースからのREDOデータの転送](#)

表一覧

- [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](#)
- [3-1 フィジカル・スタンバイ・データベースの作成](#)
- [4-1 ロジカル・スタンバイ・データベースの作成](#)
- [6-1 データ保護モードに対するREDO転送の必須属性](#)
- [7-1 LOG_ARCHIVE_DEST_STATE_n初期化パラメータの値](#)
- [7-2 REDO転送の待機イベント](#)
- [10-1 フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更](#)
- [10-2 一般的なプライマリ・データベースの管理アクションに関する情報ソース](#)
- [13-1 ロジカル・スタンバイの新規作成によってローリング・アップグレードを実行するステップ](#)
- [13-2 既存のロジカル・スタンバイを使用したローリング・アップグレードの実行ステップ](#)
- [13-3 既存のフィジカル・スタンバイを使用したローリング・アップグレードの実行ステップ](#)
- [14-1 後続グループ・フィジカル\(TGP\)と先行グループ・フィジカル\(LGP\)](#)
- [14-2 DBMS_ROLLINGを使用したローリング・アップグレードの実行ステップ](#)
- [16-1 Oracle Data Guard構成内のインスタンスの初期化パラメータ](#)
- [17-1 TEMPLATE属性のディレクティブ](#)
- [18-1 Data Guard環境で使用されるALTER DATABASE文](#)
- [18-2 Oracle Data Guard環境で使用されるALTER SESSION文](#)
- [18-3 Oracle Data Guard環境で使用されるALTER SYSTEM文](#)
- [19-1 Oracle Data Guard構成に関連のあるビュー](#)
- [A-1 一般的なSQL Applyエラーの解決方法](#)
- [C-1 DBMS_LOGSTDBY.SKIPプロシージャのstmtパラメータの値](#)
- [D-1 LOG_ARCHIVE_FORMAT初期化パラメータのディレクティブ](#)

はじめに

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

対象読者

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

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

ドキュメントのアクセシビリティについて

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

Oracleサポートへのアクセス

サポートを購入したオラクル社のお客様は、My Oracle Supportを介して電子的なサポートにアクセスできます。詳細情報は (<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info>) か、聴覚に障害のあるお客様は (<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs>) を参照してください。

関連ドキュメント

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

- 『[Oracle Database概要](#)』の冒頭部分は、Oracleデータベースに関する概念と用語を提供し、このガイドに記載された詳細な情報の基礎として機能します。
- 『[Oracle Database管理者ガイド](#)』の制御ファイル、REDOログ・ファイル、アーカイブREDOログ・ファイルの管理に関する章。
- 『[Oracle Databaseユーティリティ](#)』のLogMiner技術について説明している章。
- 『[Oracle Data Guard Broker](#)』。Oracle Data Guard構成の作成、メンテナンス、監視を自動化および集中管理するためのグラフィカル・ユーザー・インタフェースとコマンドライン・インタフェースについて説明しています。
- Oracle Enterprise Manager Cloud Controlオンライン・ヘルプ・システム

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。

規則	意味
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

『Oracle Data Guard概要および管理』のこのリリースでの変更点

この序文では『Oracle Data Guard概要および管理』の変更をリストします。

Oracle Databaseリリース19cにおける変更点

Oracle Databaseリリース19cのOracle Data Guard概要および管理の変更点は、次のとおりです。

新機能

- プライマリで取得された時点までフィジカル・スタンバイをフラッシュバックするプロセスが、リストア・ポイントをプライマリからスタンバイに自動的にレプリケートすることによって簡略化されました。[プライマリからスタンバイへのリストア・ポイントのレプリケート](#)を参照してください。
- プライマリ・データベースでフラッシュバックまたはPoint-in-Timeリカバリが実行されると、マウント・モードのスタンバイは、プライマリで実行される同じリカバリ手順に従います。[プライマリRESETLOGS操作の後のマウント済スタンバイの自動フラッシュバック](#)を参照してください。
- Active Data Guardスタンバイ・データベースで、Oracle Databaseのインメモリー列ストアおよびData Guardの複数インスタンスのREDO Applyを同時に有効にできます。
- DML操作は、Active Data Guardスタンバイ・インスタンスで実行できます。無効なPL/SQLオブジェクトがADGスタンバイ・データベースで実行されると、そのオブジェクトは自動的に再コンパイルされます。[Active Data Guardスタンバイ・データベースでのDML操作の実行](#) and [Active Data Guardスタンバイ・データベースでの最上位のPL/SQL操作の実行](#)を参照してください。

サポート対象外機能

- LOG_ARCHIVE_DEST_n初期化パラメータのMAX_CONNECTIONS属性はサポートされなくなりました。
- 拡張データ型サポート(EDS)はサポートされなくなりました。EDSでサポートされているすべてのOracleデータ型がロジカル・スタンバイまたはOracle GoldenGateでネイティブにサポートされるようになりました。

Oracle Databaseリリース18c (バージョン18.1)での変更点

Oracle Databaseリリース18c (バージョン18.1)での変更点に対応するOracle Data Guard概要および管理の変更点は、次のとおりです。

- ロールの変更時にOracle Active Data Guardスタンバイでデータベース・バッファ・キャッシュの状態が保持されるようになりました。[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](#)を参照してください。
- Oracle Active Data Guardスタンバイ・データベースでグローバル一時表を動的に作成できるようになりました。[Active Data Guardインスタンス上のグローバル一時表](#)を参照してください。
- 新しい初期化パラメータADG_ACCOUNT_INFO_TRACKINGによって、ログイン攻撃に対するユーザー・アカウント・セキュリティの制御が本番データベースとすべてのOracle Active Data Guardスタンバイ・データベースに拡張されます。[『Oracle Databaseリファレンス』](#)を参照してください。
- 新しいビューV\$DATAGUARD_PROCESS (V\$MANAGED_STANDBYの置換え)に表示される情報を問い合わせることで、REDOがプライマリ・データベースから送信され、スタンバイ・データベースに適用されることを確認できます。[フィジカル・スタンバイの作成 タスク7: フィジカル・スタンバイ・データベースが正しく実行されているかどうかの確認](#)を参照してください。

- プライベート一時表(ローカル一時表とも呼ばれる)のメタデータをメモリーに格納できます。これにより、読取り専用データベースでプライベート一時テーブルを有効化できるため、Oracle Active Data Guardスタンバイ・データベースでレポート・アプリケーションを実行できます。[Active Data Guardインスタンス上のプライベート一時表](#)を参照してください。
- データベースのNOLOGGINGが、Standby Nologging for Load PerformanceとStandby Nologging for Data Availabilityの2つの新しいモードで拡張されました。これらのモードにより、Oracle Active Data Guard環境で生成されるREDOの量を大幅に増やすことなくNOLOGGINGを使用できるようになりました。[適切なロギング・モードの有効化](#)を参照してください。
- 1つのRMANコマンドRECOVER STANDBY DATABASEを使用してネットワーク経由でスタンバイ・データベースをリフレッシュできるようになりました。[1つコマンドを使用したスタンバイのロールフォワード](#)を参照してください。
- DBMS_ROLLING PL/SQLパッケージを使用して実行されるアップグレードに対するData Guard Brokerのサポートが拡張されました。[Data GuardブローカでのDBMS_ROLLINGアップグレードのサポート](#)を参照してください。
- 複数インスタンスのREDO Applyでブロック・チェンジ・トラッキングがサポートされるようになりました。[複数インスタンスREDO Applyの設定](#)を参照してください。

第I部 概要および管理

次の各項では、Oracle Data Guardの概要および管理について説明します。

- [Oracle Data Guardの概要](#)
- [Oracle Data Guardの開始](#)
- [フィジカル・スタンバイ・データベースの作成](#)
- [ロジカル・スタンバイ・データベースの作成](#)
- [遠隔同期インスタンスの使用](#)
- [Oracle Data Guardの保護モード](#)
- [REDO転送サービス](#)
- [適用サービス](#)
- [ロールの推移](#)
- [フィジカルおよびスナプショット・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](#)
- [RMANを使用したファイルのバックアップおよびリストア](#)
- [SQL Applyを使用したOracle Databaseのアップグレード](#)
- [DBMS_ROLLINGを使用したローリング・アップグレードの実行](#)
- [Oracle Data Guardの使用例](#)

1 Oracle Data Guardの概要

Oracle Data Guard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。

Oracle 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。Oracle Data Guardでは、これらのスタンバイ・データベースを本番データベースのコピーとしてメンテナンスします。したがって、本番データベースが計画的または計画外の停止によって使用不可能になった場合は、スタンバイ・データベースを本番ロールに切り替えて、停止時間を最小限にできます。Oracle 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。Oracle Data Guard転送サービスは、ソース・データベースから1つ以上のリモートの宛先への効率的で信頼性の高いREDO転送のため、Oracle StreamsやOracle GoldenGateなどの他のOracleの機能によっても使用されます。

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

Oracle Data Guardの概要を説明する次の項を参照してください。

- [Oracle Data Guardの構成](#)
- [Oracle Data Guardのサービス](#)
- [『Oracle Data Guard Broker』](#)
- [Oracle Data Guardの保護モード](#)
- [クライアント・フェイルオーバー](#)
- [Oracle Data Guardと補完テクノロジー](#)
- [Oracle Active Data GuardのOracle Shardingサポート](#)
- [Oracle Data Guardのメリットの要約](#)

1.1 Oracle Data Guardの構成

Oracle Data Guard構成には、1つのプライマリ・データベースと最大30の宛先が含まれます。

Oracle Data Guard構成内のメンバーはOracle Netで接続され、地理的に散在していてもかまいません。Oracle Data Guard構成内のメンバーの配置場所に制限はありませんが、相互に通信する必要があります。たとえば、1つのスタンバイ・データベースをプライマリ・データベースと同じデータ・センターに、2つのスタンバイ・データベースを別のデータ・センターに配置できます。

プライマリ・データベースとスタンバイ・データベースはSQLコマンドライン・インタフェースまたはOracle Data Guard Brokerインタフェースを使用して管理できます。ブローカでは、コマンドライン・インタフェース(DGMGRL)や、Oracle Enterprise Manager Cloud Controlに統合されたグラフィカル・ユーザー・インタフェースが提供されます。

1.1.1 プライマリ・データベース

Oracle Data Guardの構成には、プライマリ・データベースとも呼ばれる本番データベースが1つ含まれており、プライマリ・ロールで機能します。

プライマリ・データベースは、ほとんどのアプリケーションがアクセスするデータベースです。

プライマリ・データベースは、単一インスタンスのOracleデータベースまたはOracle Real Application Clusters(Oracle

RAC)データベースのいずれかです。

1.1.2 スタンバイ・データベース

スタンバイ・データベースは、トランザクション上の一貫性を持つ、プライマリ・データベースのコピーです。

プライマリ・データベースのバックアップ・コピーを使用すると、最大30個のスタンバイ・データベースを作成して、Oracle Data Guard構成に組み込むことができます。Oracle Data Guardでは、プライマリ・データベースからスタンバイ・データベースにREDOデータを転送し、スタンバイ・データベースにREDOを適用することによって、各スタンバイ・データベースを自動的にメンテナンスします

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

スタンバイ・データベースのタイプは、次のとおりです。

● フィジカル・スタンバイ・データベース

プライマリ・データベースと物理的に同一のコピーを提供します。これはプライマリ・データベースとブロック単位で同じディスク・データベース構造です。データベース・スキーマ(インデックスを含む)は同一です。フィジカル・スタンバイ・データベースは、REDO Applyによってプライマリ・データベースとの同期が保たれます。REDO Applyでは、プライマリ・データベースから受け取ったREDOデータがリカバリされ、このREDOがフィジカル・スタンバイ・データベースに適用されます。

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

さらにOracle Database 11gリリース2 (11.2.0.1)現在、フィジカル・スタンバイ・データベースを使用して、適格な個別パッチ、パッチ・セット更新(PSU)およびクリティカル・パッチ・アップデート(CPU)をローリングという方法でインストールできます。この機能の詳細は、<http://support.oracle.com>のMy Oracle Supportノート1265700.1を参照してください。

● ロジカル・スタンバイ・データベース

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

ロジカル・スタンバイ・データベースの柔軟性の高さにより、ローリング方式でほとんど停止時間なく、Oracle Database ソフトウェア(パッチ・セットおよび新しいOracle Databaseリリース)をアップグレードし、その他のデータベースのメンテナンスを実行することができます。Oracle Database 11g以降から、一時ロジカル・データベースのローリング・アップグレード・プロセスを、既存のフィジカル・スタンバイ・データベースとともに使用することもできます。

● スナップショット・スタンバイ・データベース

スナップショット・スタンバイ・データベースは完全に更新可能なスタンバイ・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・データベースと同様に、スナップショット・スタンバイ・データベースは、プライマリ・データベースからREDOデータを受信し、アーカイブします。フィジカルまたはロジカル・スタンバイ・データベースとは異なり、スナップショット・スタンバイ・データベースは、受信したREDOデータを適用しません。スナップショット・スタンバイ・データベースが受信したREDOデータは、スナップショット・スタンバイが元のフィジカル・スタンバイ・データベースに変換され、スナップショット・スタンバイ・データベースに対するすべてのローカルな更新が破棄された後に適用されます。

スナップショット・スタンバイ・データベースの最適な使用例は、更新可能なフィジカル・スタンバイ・データベースのスナップショットが一時的に必要な場合です。たとえば、Oracle Real Application Testingのオプションを使用してプライマ

リ上のデータベース・ワークロードを取得して、スナップショット・スタンバイでのテストの目的でそれに応答できます。スナップショット・スタンバイ・データベースで受信されたREDOデータは、フィジカル・スタンバイに変換されて戻るまで適用されないため、プライマリ・データベースの障害からリカバリするのに必要な時間は、適用する必要があるREDOデータの量に正比例します。

関連項目:

- Oracle Real Application Testingおよびその使用に必要なライセンスの詳細は、[Oracle Database Testing ガイド](#)を参照してください

1.1.3 遠隔同期インスタンス

Oracle Data Guard遠隔同期インスタンスは、プライマリ・データベースからREDOを受け取り、それをOracle Data Guard構成の他のメンバーに送信する、リモートのOracle Data Guardの宛先です。

遠隔同期インスタンスは、制御ファイルを管理し、REDOをスタンバイREDOログ(SRL)に受信し、それらのSRLをローカルのアーカイブREDOログにアーカイブしますが、スタンバイとの類似はそこまでです。遠隔同期インスタンスにはユーザー・データファイルは含まれず、アクセスのために開くことも、REDO Applyを実行することもできず、プライマリ・ロールで機能することも、スタンバイ・データベースの任意のタイプに変換することもできません。

遠隔同期インスタンスはOracle Active Data Guard遠隔同期機能の一部で、Oracle Active Data Guardのライセンスが必要です。

関連項目:

- [遠隔同期](#)

1.1.4 Zero Data Loss Recovery Appliance。

Zero Data Loss Recovery Appliance (リカバリ・アプライアンス)は、すべてのOracleデータベースのバックアップに対して単一のリポジトリを提供するエンタプライズレベルのバックアップ・ソリューションです。

リカバリ・アプライアンスは、ほとんどのOracleデータベースのバックアップおよびリストア処理を一元化されたバックアップ・システムにオフロードします。ストレージ使用率、パフォーマンスおよびバックアップ管理の容易性において著しい効率化を実現できます。

関連項目:

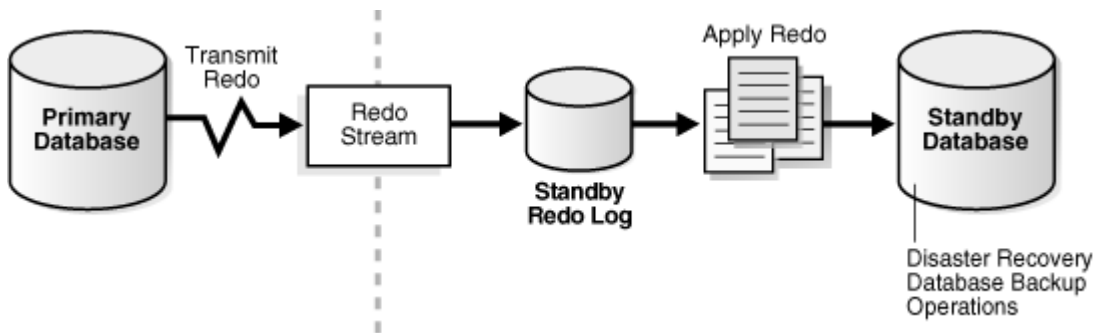
- [『Zero Data Loss Recovery Appliance管理者ガイド』](#)

1.1.5 構成の例

[図1-1](#)は、REDOデータをスタンバイ・データベースに転送するプライマリ・データベースが含まれる一般的なOracle Data Guard構成を示しています。スタンバイ・データベースは、障害時リカバリ操作とバックアップ操作に備えて、プライマリ・データベースから離れた位置に配置されます。スタンバイ・データベースはプライマリ・データベースと同じ位置にも構成できます。ただし、障

害時リカバリに使用の場合は、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勧めします。

図1-1 一般的なOracle Data Guard構成



1.2 Oracle Data Guardのサービス

Oracle Data Guardでは、REDO転送サービスおよび適用サービスを使用してREDOデータの転送、REDOデータの適用、およびデータベース・ロールの変更が管理されます。

- [REDO転送サービス](#)

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

- [適用サービス](#)

REDOデータは、リアルタイム適用を使用して書き込まれると同時に、スタンバイREDOログ・ファイルから直接適用されます。REDOログ・ファイルが構成されていない場合、REDOデータは適用される前に、最初にスタンバイ・データベースでアーカイブされる必要があります。

- [ロールの推移](#)

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

1.2.1 REDO転送サービス

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

REDO転送サービスでは、次のタスクを実行します。

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

1.2.2 適用サービス

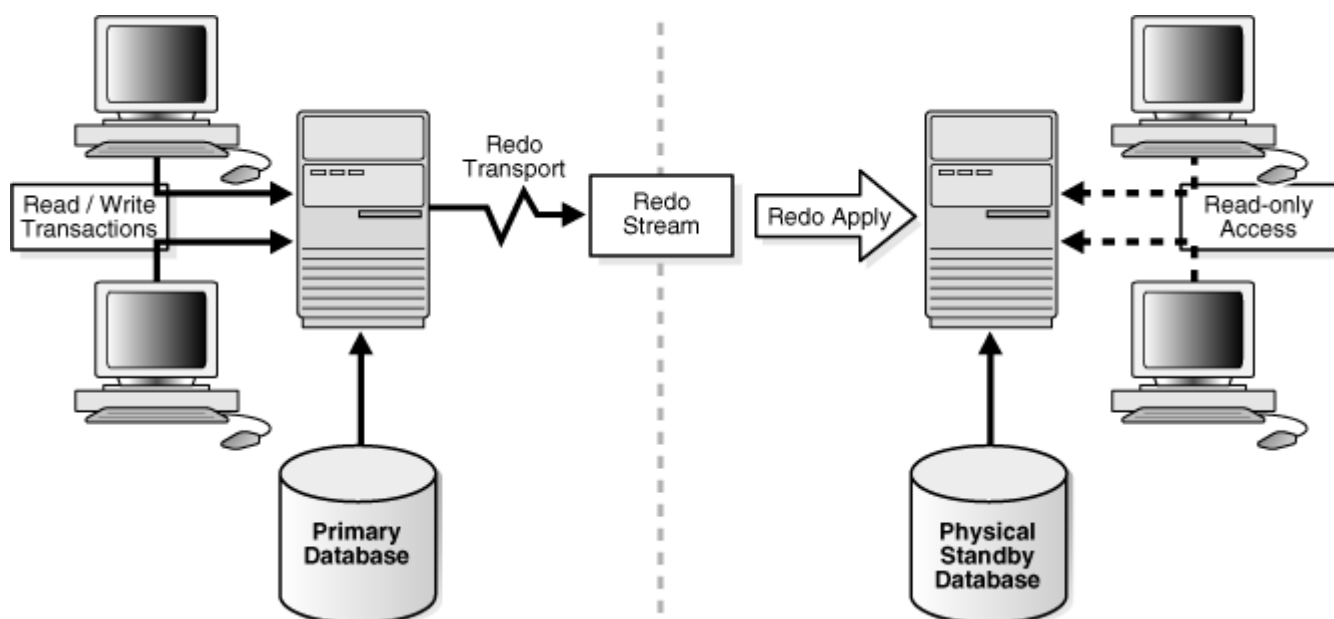
適用サービスによって、スタンバイ・データベースにREDOデータを自動的に適用し、プライマリ・データベースとの一貫性を維持します。

REDOデータは、プライマリ・データベースから送信され、スタンバイ・データベース上のスタンバイREDOログに書き込まれます。適用サービスでは、そのデータへの読取り専用アクセスも可能です。

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

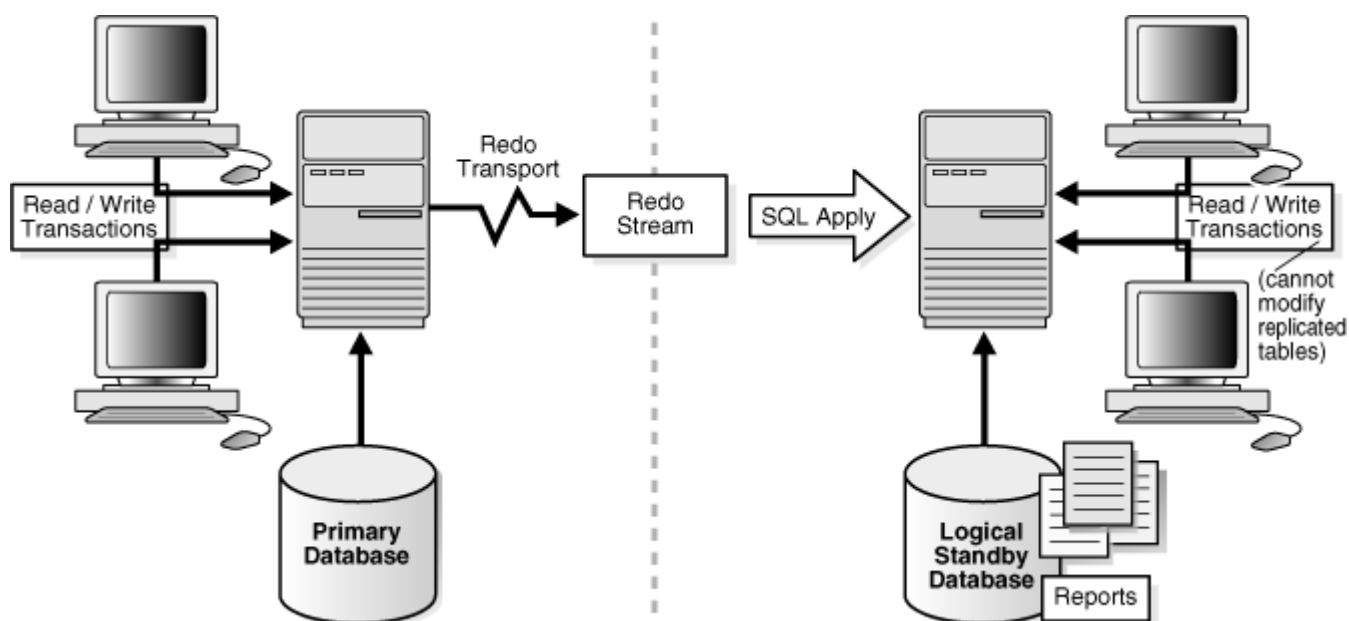
- フィジカル・スタンバイ・データベースの場合、Oracle Data Guardでは**REDO Apply**テクノロジーを使用します。このテクノロジーでは、[図1-2](#)に示すように、Oracleデータベースの標準リカバリ技法を使用してREDOデータがスタンバイ・データベースに適用されます。

図1-2 フィジカル・スタンバイ・データベースの自動更新



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

図1-3 ロジカル・スタンバイ・データベースの自動更新



1.2.3 ロールの推移

Oracle 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。

Oracleデータベースは、プライマリ・ロールまたはスタンバイ・ロールのいずれかで実行されます。

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

ライマリ・ロールに遷移します。

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

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

1.3 『Oracle Data Guard Broker』

Oracle Data Guard Brokerは、Oracle Data Guard構成の作成、メンテナンス、監視を自動化する分散管理フレームワークです。

Oracle Enterprise Manager Cloud Controlのグラフィカル・ユーザー・インタフェース(GUI)またはOracle Data Guard コマンドライン・インタフェース(DGMGRL)を使用すると、次の操作が自動化および単純化されます。

- REDO転送サービスや適用サービスの設定など、Oracle Data Guard構成の作成と有効化
- Oracle Data Guard構成全体を、構成内の任意のシステムから管理
- Oracle RACのプライマリ・データベースまたはスタンバイ・データベースが含まれるOracle Data Guard構成の管理と監視
- スイッチオーバーおよびフェイルオーバーを単純化して、Oracle Enterprise Manager Cloud Controlでキー・クリックを1回する、あるいはDGMGRLコマンドライン・インタフェースで単一のコマンドを使用すると起動できるようにします。
- プライマリ・データベースが使用できなくなった場合に、Oracle Data Guardのファスト・スタート・フェイルオーバーを有効化し、すぐにフェイルオーバーを自動的に行います。ファスト・スタート・フェイルオーバーが有効化されている場合、Oracle Data Guardブローカによってフェイルオーバーの必要性が判断され、指定されたターゲット・スタンバイ・データベースへのフェイルオーバーが自動的に開始されます。DBAによる操作は不要です。

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

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

関連項目:

詳細は、[『Oracle Data Guard Broker』](#)を参照してください。

1.3.1 Oracle Enterprise Manager Cloud Controlの使用

Oracle Enterprise Manager Cloud Controlは、Oracle Data Guard構成内のプライマリ・データベースおよびスタンバイ・データベースを表示、監視および管理するための、Webベースのインタフェースを提供します。

Oracle Enterprise Manager Cloud Controlは、単にCloud Controlと呼ばれることもあります。

Enterprise Managerの使いやすいインタフェースに、ブローカによる一元的な管理と、Oracle Data Guard構成の監視を組み合わせることで、高可用性、サイト保護、企業のデータ保護に必要なOracle Data Guardソリューションが強化されます。

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

1.3.2 Oracle Data Guardコマンドライン・インタフェースの使用

Oracle Data Guardコマンドライン・インタフェース(DGMGRL)では、Oracle Data Guard構成をDGMGRLプロンプトまたはスクリプト内から制御および監視できます。

DGMGRLを使用すると、構成でデータベースを管理および監視するために必要なアクティビティのほとんどを実行できます。

DGMGRLの詳細な参照情報と例は、『[Oracle Data Guard Broker](#)』を参照してください。

1.4 Oracle Data Guardの保護モード

Oracle Data Guardには、データ保護の3種類のモードがあります。

場合によっては、状況に関係なくデータの消失が許されないビジネスがあります。また、起こりそうにない多重障害の場合に起こりうるデータの消失よりデータベースの可用性の方が重要な場合もあります。アプリケーションの中には、常に最大のデータベース・パフォーマンスが必要であるため、コンポーネントに障害が発生した場合に少量のデータの消失ならば許容できるものがあります。これらの各状況で使用可能な保護モードの簡単な説明は、次のとおりです。

最大可用性

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

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

最大パフォーマンス

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

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

最大保護

この保護モードは、プライマリ・データベースに障害が発生した場合でも、データ消失がないことを保証します。このレベルの保護を提供するには、トランザクションがコミットされる前に、トランザクションのリカバリに必要なREDOデータを、オンラインREDOログおよび少なくとも1つの同期化されたスタンバイ・データベース上のスタンバイREDOログに書き込む必要があります。データの消

失が起り得ないようにするため、1つ以上の同期スタンバイ・データベースにREDOストリームを書き込めない場合、プライマリ・データベースは停止し、トランザクション処理を継続しません。

3つの保護モードはいずれも、特定のREDO転送オプションを使用して、REDOデータが少なくとも1つのスタンバイ・データベースに送信されるようにする必要があります。

関連項目:

- [「Oracle Data Guardの保護モード」](#)にはこのモードの詳細な説明と、プライマリ・データベースの保護モードの設定に関する情報が記載されています。

1.5 クライアント・フェイルオーバー

高可用性アーキテクチャには、データベースおよびデータベース・クライアントに対する高速フェイルオーバー機能が必要です。クライアント・フェイルオーバーには、障害の通知、失効した接続のクリーンアップおよび新しいプライマリ・データベースへの透過的な再接続が含まれます。

Oracle Databaseには、データベース・フェイルオーバーとフェイルオーバー・プロシージャを統合する機能があるため、クライアントは、データベース・フェイルオーバーから数秒以内に新しいプライマリ・データベースに自動的にリダイレクトされます。

関連項目:

- Oracle Data GuardのFast Application Notification(FAN)およびFast Connection Failover(FCF)に特有な構成要件、ならびにロール特有のデータベース・サービスの詳細は、[『Oracle Data Guard Broker』](#)を参照してください。
- 次の場所にアクセスし、Maximum Availability Architectureのクライアント・フェイルオーバーのベスト・プラクティスに関する技術概要を参照してください。

<http://www.oracle.com/goto/maa>

1.5.1 アプリケーション・コンティニューイティ

アプリケーション・コンティニューイティは、データベース・セッションを使用不可にするリカバリ可能なエラーの後に、データベースに対するリクエストを中断せず、迅速な方法でリプレイできるようにするOracle Databaseの機能です。

Oracle Data Guardのフィジカル・スタンバイ・データベースへのスイッチオーバーに対し、アプリケーション・コンティニューイティがサポートされます。また、最大可用性データ保護モードのフィジカル・スタンバイに対するファスト・スタート・フェイルオーバーもサポートされます。アプリケーション・コンティニューイティを使用するには、プライマリ・データベースおよびスタンバイ・データベースにOracle Real Application Clusters (Oracle RAC)またはOracle Active Data Guardのライセンスが必要です。

関連項目:

- アプリケーション・コンティニューイティの詳細は、[Oracle Real Application Clusters管理およびデプロイメント・ガイド](#)を参照してください。

1.6 Oracle Data Guardと補完テクノロジー

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

Oracleの高可用性テクノロジーには、次のものがあります。

- Oracle Real Application Clusters(Oracle RAC)

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

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

ローカル・サイトとリモート・サイトの使用、ロジカル・スタンバイ・データベースとフィジカル・スタンバイ・データベースの組合せとノードの使用によって、Oracle RACおよびOracle Data Guardを使用した様々なアーキテクチャを実現できます。Oracle RACおよびOracle Data Guardの統合については、[Oracle Data GuardとOracle Real Application Clusters](#)を参照してください。

- Oracle Real Application Clusters One Node(Oracle RAC One Node)

Oracle RAC One Nodeは、強化された高可用性を非クラスター・データベースに提供し、データベースを計画済停止時間および計画外停止時間の両方から保護します。Oracle RAC One Nodeでは次のことが提供されます。

- 常にオン状態の非クラスター・データベース・サービス
- データベース・サーバーの優れた統合
- 強化されたサーバーの仮想化
- 完全Oracle RACの開発およびテスト・プラットフォームのコストの低減

さらに、Oracle RAC One Nodeは、データベース記憶域の統合を容易にし、データベース環境を標準化します。必要に応じて、停止時間や中断を発生させることなく、完全な複数ノードOracle RACデータベースにアップグレードできます。

Oracle Database 11gリリース2(11.2.0.2)では、Oracle Data GuardとOracle Data Guard BrokerがOracle Real Application Clusters One Node (Oracle RAC One Node)と完全に統合されました。

- フラッシュバック・データベース

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

- バックアップをリストアして、エラーや破損が発生した時点までロールフォワード変更を行う必要がなくなります。かわりに、フラッシュバック・データベースでは、データファイルをリストアせずに、以前の時点までOracleデータベースをロールバックできます。
- REDOの適用を遅延してユーザー・エラーや論理的な破損から保護するための、代替手段を提供します。し

たがって、スタンバイ・データベースは、プライマリ・データベースとより密接に同期できるため、フェイルオーバーおよびスイッチオーバーの回数が少なくなります。

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

データベースのフラッシュバックの詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を、REDOデータの適用の詳細は、[「アーカイブREDOログ・ファイルの適用に対する遅延時間の指定」](#)を参照してください。

● Recovery Manager (RMAN)

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

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

[「Recovery Managerを使用したスタンバイ・データベースの作成」](#)を参照してください。

● Oracle Global Data Services (GDS)

Oracle Global Data Services (GDS) はOracle RACサービス・モデルをグローバル分散データベースのプールに適用して、共通サービスを提供するレプリケートされたデータベースのセットに対し、動的なロード・バランシング、フェイルオーバー、および集中サービス管理を提供します。データベースのセットには、Oracle RACと、Oracle Data Guard、Oracle GoldenGate、またはその他のレプリケーション技術で相互接続された単一インスタンスのOracleデータベースが含まれます。

GDSは、Oracle Data Guard Brokerに統合されています。これにより、ロールの推移がOracle Data Guard Broker構成内で発生すると、必要に応じてロール特有のグローバル・サービスを自動的に開始および停止できます。

GDSにより、グローバル・サービスに対するレプリケーションのラグ制限を指定できます。指定したレプリカでラグ制限が超過した場合、そのレプリカでのグローバル・サービスは一時的に停止し、新しいクライアント・リクエストが、ラグ制限を満たすレプリカにルーティングされます。グローバル・サービスは、レプリケーション・ラグがラグ制限を下回ると、元のレプリカで自動的に再開します。

1.7 Oracle Active Data GuardのOracle Shardingサポート

Oracle Shardingにより、複数の独立したOracle Database間でデータを水平に分割し、データベース接続リクエストを適切なデータを含むデータベースにルーティングできます。Oracle Data GuardとOracle Shardingは統合されたテクノロジーです。

シャーディングでは、物理リソースを共有しない複数の独立したデータベース(シャード)にデータを分割します。シャーディングは通常、データ・レプリケーション(Oracle Data Guardによって提供されたものなど)と組み合わせられています。Oracle Data Guardでは、Oracle Database全体の迅速な単一マスター・レプリケーションが可能です。Oracle Data Guard構成には、更新可能なプライマリ・データベースと、読取り専用アクセスで開くことのできる1つ以上のスタンバイ・データベースが含まれています。レプリケーションによって、シャードされたデータベースのパフォーマンスと拡張性が向上し、高可用性が実現し、障害回復が

可能になります。シャードされたデータベースのレプリケーション・トポロジは、GDSCTL create shardcatalogコマンドで-replオプションを使用して指定されます。デフォルトのレプリケーション・トポロジはOracle Data Guardです。

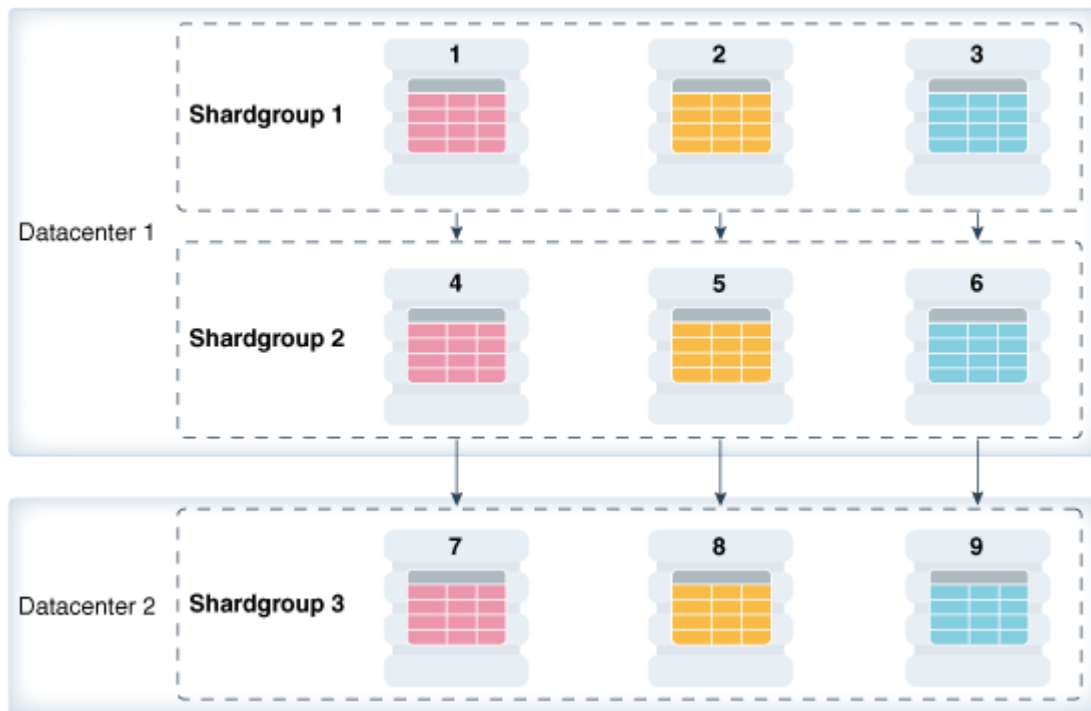
Oracle Shardingでは、システム管理および複合の2つのシャーディング方法をサポートしています。シャーディング方法は、GDSCTLコマンドCREATE SHARDCATALOGで指定されます。

シャードグループに属するシャードは通常、同じデータ・センターにあります。シャードグループ全体を、同じまたは異なるデータ・センターの1つ以上のシャードグループに完全にレプリケートできます。

Oracle Data Guardレプリケーションによるシステム管理シャーディング

システム管理シャーディングでは、レプリケーションの論理単位は、シャードグループと呼ばれるシャードのグループです。システム管理シャーディングでは、シャードグループに、シャードされたデータベースに格納されているすべてのデータが含まれます。データは、コンシステント・ハッシュによるパーティション化を使用してシャード間に自動的に分散されます(コンシステント・ハッシュは、スケラブルな分散システムで一般的に使用されるパーティション化方法です)。パーティション化アルゴリズムにより、データがシャード間に均一およびランダムに分散されます。次の図は、システム管理シャーディングとともにOracle Data Guardレプリケーションが使用される方法を示しています。プライマリ・シャードグループ(シャードグループ1)と2つのスタンバイ・シャードグループ(シャードグループ2とシャードグループ3)があります。

図1-4 システム管理シャーディングとOracle Data Guardレプリケーション



シャードグループ1はOracle Data Guardプライマリ・データベース(シャード1、2、3)で構成されます。シャードグループ2は、同じデータセンターにあり同期レプリケーションを行うように構成されたスタンバイ・データベース(シャード4、5、6)で構成されます。また、シャードグループ3は、別のデータセンターにあり非同期レプリケーションを行うように構成されたりモート・スタンバイ(シャード7、8、9)で構成されます。デフォルトのレプリケーション・トポロジはOracle Data Guardです。構成内の各スタンバイを読み取り専用モードでオープンするには、Oracle Active Data Guardを有効化する必要があります。これは、GDSCTL add shardgroupコマンドで-deploy_as ACTIVE_STANDBYオプションを使用して実行されます。

前の図に示されたシャードされたデータベースは、レプリケートされた3つのシャードのセット({1, 4, 7}、{2, 5, 8}および{3, 6, 9})で構成されます。レプリケートされた各シャードのセットは、ファスト・スタート・フェイルオーバーが有効なOracle Data Guard Broker構成として管理されます。

レプリケーションをデプロイするには、シャードグループのプロパティ(リージョン、ロールなど)を指定して、シャードを追加するだけで済みます。Oracle Shardingは自動的にOracle Data Guardを構成し、レプリケートされたシャードのセットごとにファスト・ス

タート・フェイルオーバー・オブザーバを起動します。また、読取り専用ワークロードのロード・バランシング、ロールベースのグローバル・サービスおよびレプリケーション・ラグとローカルティベースのルーティングを提供します。

前の図に示す構成をデプロイするには、次のGDSCTLコマンドを実行します。

```
CREATE SHARDCATALOG -database host00:1521:shardcat -region dc1, dc2

ADD GSM -gsm gsm1 -listener 1571 -catalog host00:1521:shardcat -region dc1
ADD GSM -gsm gsm2 -listener 1571 -catalog host00:1521:shardcat -region dc2

ADD SHARDGROUP -shardgroup shardgroup1 -region dc1 -deploy_as primary
ADD SHARDGROUP -shardgroup shardgroup2 -region dc1 -deploy_as standby
ADD SHARDGROUP -shardgroup shardgroup3 -region dc2 -deploy_as standby

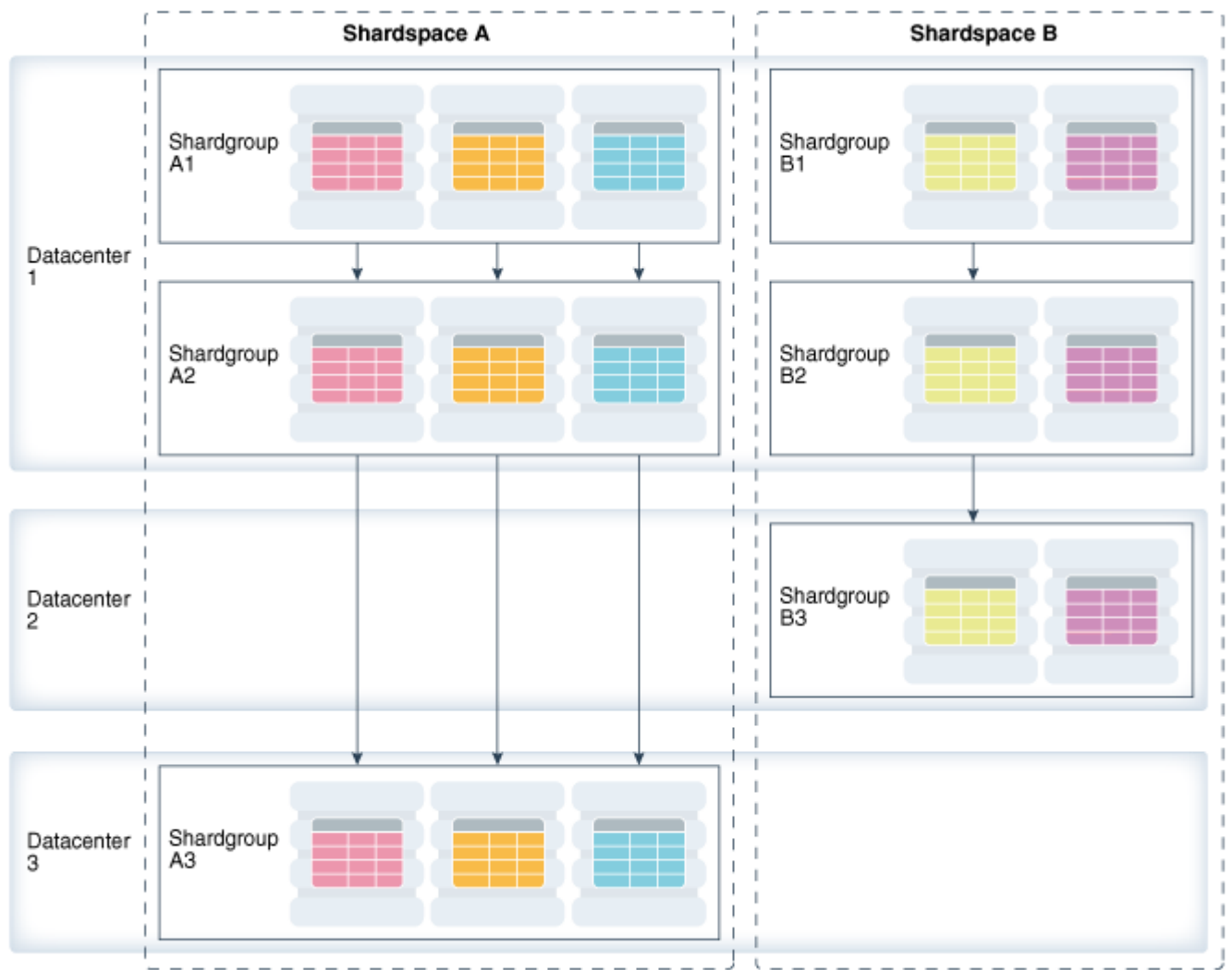
CREATE SHARD -shardgroup shardgroup1 -destination host01 -credential oracle_cred -netparamfile
/home/oracle/netca_dbhome.rsp
CREATE SHARD -shardgroup shardgroup1 -destination host02 -credential oracle_cred -netparamfile
/home/oracle/netca_dbhome.rsp
CREATE SHARD -shardgroup shardgroup1 -destination host03 -credential oracle_cred -netparamfile
/home/oracle/netca_dbhome.rsp
.....
CREATE SHARD -shardgroup shardgroup3 -destination host09 -credential oracle_cred -netparamfile
/home/oracle/netca_dbhome.rsp

DEPLOY
```

複合シャーディングとOracle Data Guardレプリケーション

複合シャーディングでは、システム管理シャーディングの機能とユーザー定義シャーディングの機能が組み合されています。複合シャーディングでは、レプリケーションの論理単位は、(システム管理シャーディングと同様に)シャードグループと呼ばれるシャードのグループです。また、複合シャーディングでは、シャードされたデータベースは複数のシャードスペースで構成されますが(システム管理シャーディングと同様に)、各シャードスペース(レプリケートされたシャードではなく)には次の図に示すようにレプリケートされたシャードグループが含まれます。

図1-5 複合シャーディングとOracle Data Guardレプリケーション



前の図に示す構成をデプロイするには、次のGDSCTLコマンドを実行します。

```
CREATE SHARDCATALOG -sharding composite -database host00:1521:cat -region dc1, dc2, dc3

ADD GSM -gsm gsm1 -listener 1571 -catalog host00:1521:cat -region dc1
ADD GSM -gsm gsm2 -listener 1571 -catalog host00:1521:cat -region dc2
ADD GSM -gsm gsm3 -listener 1571 -catalog host00:1521:cat -region dc3

ADD SHARDSPACE -shardspace shardspace_a
ADD SHARDSPACE -shardspace shardspace_b

ADD SHARDGROUP -shardgroup shardgroup_a1 -shardspace shardspace_a -region dc1
-deploy_as primary
ADD SHARDGROUP -shardgroup shardgroup_a2 -shardspace shardspace_a -region dc1 -deploy_as standby
ADD SHARDGROUP -shardgroup shardgroup_a3 -shardspace shardspace_a -region dc3 -deploy_as standby
ADD SHARDGROUP -shardgroup shardgroup_b1 -shardspace shardspace_a -region dc1
-deploy_as primary
ADD SHARDGROUP -shardgroup shardgroup_b2 -shardspace shardspace_a -region dc1 -deploy_as standby
ADD SHARDGROUP -shardgroup shardgroup_b3 -shardspace shardspace_a -region dc2 -deploy_as standby

CREATE SHARD -shardgroup shardgroup_a1 -destination host01 -credential orcl_cred -netparamfile
/home/oracle/netca_dbhome.rsp
.....
CREATE SHARD -shardgroup shardgroup_b3 -destination host09 -credential orcl_cred -netparamfile
/home/oracle/netca_dbhome.rsp

DEPLOY
```

関連トピック

- [Oracle Database管理者ガイド](#)
- [Oracle Database Global Data Services概要および管理ガイド](#)

1.8 Oracle Data Guardのメリットの要約

Oracle Data Guardは、効率のよい包括的な障害時リカバリおよび高可用性ソリューションを提供します。

Oracle Data Guardには次のようなメリットがあります。

- 高可用性

管理が容易なOracle Data Guard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

Oracle Data Guardでは、予期しない障害が発生した場合でもデータ消失がないことを保証します。スタンバイ・データベースにより、データ破損や管理エラーを含む、あらゆる種類の計画外停止を防ぎます。プライマリ・データベースから受信したREDOデータはスタンバイ・データベースで検証されるため、プライマリ・データベースで発生する可能性のある物理破損は、スタンバイ・データベースに伝播しません。スタンバイ・データベースで実行される追加の検証も、論理ブロック内破損や書き込みの欠落がスタンバイに伝播するのを防ぎます。同様に、記憶域管理者による偶然のファイル削除などの管理エラーも、スタンバイ・データベースに伝播しません。フィジカル・スタンバイ・データベースを使用して、Redo Applyの遅延、またはスタンバイを巻き戻してデータの正しいコピーを抽出するためのフラッシュバック・データベースの使用によるユーザー・エラーを防ぐこともできます。

- システム・リソースの効率的な使用

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

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

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

- 自動ギャップ検出および解決

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できません。接続が再度確立された後、Oracle Data Guardによって、欠落しているアーカイブREDOログ・ファイル(ギャップと呼ばれます)が自動的に検出され、それらのファイルがスタンバイ・データベースに自動的に転送されます。スタンバイ・データベースはプライマリ・データベースと同期化されるため、DBAによる手動操作は不要です。

- 集中化された単純な管理

Oracle Data Guard Brokerにはグラフィカル・ユーザー・インタフェースとコマンドライン・インタフェースがあり、Oracle Data Guard構成の複数のデータベースにまたがる管理タスクや運用タスクが自動化されます。また、ブローカは、単一のOracle Data Guard構成内のすべてのシステムを監視します。

- Oracle Databaseとの統合

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

- 自動的なロールの推移

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

2 Oracle Data Guardの開始

この項では、Oracle Data Guardの使用を開始する方法について説明します。

次のトピックを参照してください。

- [スタンバイ・データベースのタイプ](#)
- [Oracle Data Guard構成の管理のためのユーザー・インタフェース](#)
- [Oracle 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](#)
- [オンライン・データファイルの場所の移行](#)

2.1 スタンバイ・データベースのタイプ

スタンバイ・データベースは、Oracle本番データベースのトランザクション一貫性のあるコピーで、最初はプライマリ・データベースのバックアップ・コピーから作成されます。

スタンバイ・データベースが作成および構成されると、Oracle Data Guardは、プライマリ・データベースのREDOデータをスタンバイ・システムに転送し、スタンバイ・データベースに適用することによって、スタンバイ・データベースを自動的にメンテナンスします。

スタンバイ・データベースのタイプには、フィジカル・スタンバイ・データベース、ロジカル・スタンバイ・データベース、スナップショット・スタンバイ・データベースがあります。フィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースは、必要に応じてプライマリ・データベースのロールを担い、本番処理を引き継ぐことができます。Oracle Data Guard構成にはこのようなスタンバイ・データベースの種類の組合せ(フィジカル・スタンバイ・データベース、スナップショット・スタンバイ・データベースおよびロジカル・スタンバイ・データベース)を含めることができます。

2.1.1 フィジカル・スタンバイ・データベース

フィジカル・スタンバイ・データベースはプライマリ・データベースを正確に、ブロックごとにコピーしたものです。

フィジカル・スタンバイはREDO Applyと呼ばれるプロセスによって正確なコピーとして維持されます。このプロセスではデータベース・リカバリ・メカニズムを使用して、プライマリ・データベースから受信したREDOデータを継続的にフィジカル・スタンバイ・データベースに適用します。

フィジカル・スタンバイ・データベースは、読取り専用アクセス用にオープンし、問合せをプライマリ・データベースからオフロードするために使用できます。Oracle Active Data Guardオプションのライセンスを購入している場合は、フィジカル・スタンバイ・データベースを開いている間はRedo Applyをアクティブにすることができます。それにより、問合せによって、プライマリ・データベースからのものと同じ結果が得られます。この機能は、リアルタイム問合せ機能と呼ばれます。

関連項目:

- 「[フィジカル・スタンバイ・データベースのオープン](#)」
- Oracle Active Data Guardの詳細は、[『Oracle Databaseライセンス情報』](#)を参照してください。

フィジカル・スタンバイ・データベースの利点

フィジカル・スタンバイ・データベースには次のメリットがあります。

- 障害時リカバリと高可用性

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

- データ保護

フィジカル・スタンバイ・データベースは、予期しなかった障害時でもデータ損失を防止することができます。フィジカル・スタンバイ・データベースでは、すべてのデータ型、およびプライマリ・データベースがサポートしているすべてのDDLおよびDMLの操作がサポートされます。また、データの破損およびユーザー・エラーからも保護します。プライマリ・データベースに対するストレージ・レベルの物理的な破損は、スタンバイ・データベースに伝播しません。同様に、データ損失の原因となる論理的な破損ユーザー・エラーも簡単に解決できます。

- プライマリ・データベースのワークロードの低減

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

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

- パフォーマンス

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

2.1.2 ロジカル・スタンバイ・データベース

ロジカル・スタンバイ・データベースは、最初はプライマリ・データベースと同じ内容のコピーで作成されますが、後で異なる構造に変更できます。

ロジカル・スタンバイ・データベースは、SQL文を実行して更新されます。ロジカル・スタンバイ・データベースの柔軟性の高さにより、ローリング方式でほとんど停止時間なく、Oracle Databaseソフトウェア(パッチ・セットおよび新しいOracle Databaseリリース)をアップグレードし、その他のデータベースのメンテナンスを実行することができます。Oracle Database 11g以降から、一時ロジカル・データベースのローリング・アップグレード・プロセスを、既存のフィジカル・スタンバイ・データベースとともに使用することもできます。

Oracle Data Guardは、ログ・ファイルのデータをSQL文に変換し、ロジカル・スタンバイ・データベースでそのSQL文を実行することによって、アーカイブREDOログ・ファイルまたはスタンバイREDOログ・ファイルの情報をロジカル・スタンバイ・データベースに自動的に適用します。ロジカル・スタンバイ・データベースはSQL文を使用して更新されるため、オープン状態のままであることが必要です。ロジカル・スタンバイ・データベースは読取り/書込みモードでオープンされますが、再生成されたSQLに対するターゲット表は、読取り専用操作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。

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

ロジカル・スタンバイ・データベースの利点

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

- ソフトウェア・アップグレード時の停止時間の最短化

ロジカル・スタンバイ・データベースはOracle Data Guard構成をローリング方式でアップグレードするのに最適です。ロジカル・スタンバイを使用すると、パッチ・セットやソフトウェアの新しいリリースの適用に関連する停止時間を大幅に短縮できます。ロジカル・スタンバイは、新しいリリースへのアップグレード後に切り替えて、アクティブなプライマリにすることができます。これにより、古いプライマリをロジカル・スタンバイに変換してパッチ・セットを適用する間、完全な可用性を可能にします。ロジカル・スタンバイは、DBMS_ROLLING PL/SQLパッケージの基礎となるプラットフォームを提供し、このパッケージがローリング・アップグレードやその他の記憶域の再編成の状況で、Oracle Data Guard構成の可用性を高めることのできる機能を提供します。

- レポート生成要件および意思決定支援要件のためのサポート

ロジカル・スタンバイの主な利点は、レポート作成のワークロードを最適化する重要な補助構造(プライマリのトランザクション・レスポンス時間に対する抑制効果をもたらす構造)を作成できることです。ロジカル・スタンバイは、そのデータをパーティショニングの異なる各種ストレージに物理的に再構成し、多数の異なる索引を付けて、オンデマンド・リフレッシュ・マテリアライズド・ビューを作成および維持することができます。また、データ・キューブおよびその他のOLAPデータ・ビューの作成促進に使用することができます。しかし、ロジカル・スタンバイ・データベースではデータの変換はできません(列のサブセットのみのレプリケーション、ユーザー表の追加列の許可など)。これらのタイプのレポート・アクティビティには、Oracle GoldenGateが望ましいソリューションです。

2.1.3 スナップショット・スタンバイ・データベース

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

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

プライマリ・データベースのREDOデータは受信と同時に適用されないため、スナップショット・スタンバイ・データベースは、時間の経過とともにプライマリ・データベースとの差異が生じます。スナップショット・スタンバイ・データベースのローカルな更新によって差異が増えます。ただし、スナップショット・スタンバイはいつでもフィジカル・スタンバイ・データベースに戻すことができ、その後プライマリから受信したREDOデータを適用できるため、プライマリ・データベース内のデータは完全に保護されています。

スナップショット・スタンバイ・データベースのメリット

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

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

- 常にデータ保護を維持しながら、開発およびテスト用に本番データベースの正確なレプリカを提供します。Oracle Real Application Testingのオプションを使用してプライマリ・データベース・ワークロードを取得して、スナップショット・スタンバイでのテストの目的でそれに応答できます。
- フィジカル・スタンバイに変換して再同期化することで、現行の本番データが格納されるように簡単にリフレッシュできます。

スナップショット・スタンバイの作成、テスト、本番環境との再同期化に続いて、再びスナップショット・スタンバイの作成およびテス

トを実行できることは、必要に応じて何度も繰り返すことができる1サイクルです。同じプロセスを使用して、データへの読み取り/書き込みアクセスが必要なレポート生成用にスナップショット・スタンバイを簡単に作成し、定期的に更新できます。

関連項目:

- Oracle Real Application Testingおよびその使用に必要なライセンスの詳細は、[Oracle Database Testing ガイド](#)を参照してください

2.2 Oracle Data Guard構成の管理のためのユーザー・インタフェース

Oracle Data Guardには、Oracle Data Guard構成の構成、実装および管理に使用できる、いくつかのインタフェースが用意されています。

- Oracle Enterprise Manager Cloud Control。

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

- SQL*Plusコマンドライン・インタフェース

いくつかのSQL*Plus文では、STANDBYキーワードを使用して、スタンバイ・データベースに対する操作を指定します。他のSQL文にはスタンバイ固有の構文はありませんが、スタンバイ・データベースで操作を実行するときに役立ちます。関連する文のリストについては、[「Oracle Data Guardに関連するSQL文」](#)を参照してください。

- 初期化パラメータ

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

- Oracle Data Guardブローカ・コマンドライン・インタフェース(DGMGRL)

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

2.3 Oracle Data Guardの動作要件

Oracle Data Guardの使用には、特定のハードウェアおよびソフトウェアの前提条件が必要です。

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

2.3.1 ハードウェアおよびオペレーティング・システムの要件

同じリリースのOracle Database Enterprise Editionをプライマリ・データベースおよびすべてのスタンバイ・データベースにインストールする必要があります。ただし、ロジカル・スタンバイ・データベースまたは一時ロジカル・スタンバイ・データベースを使用したローリング・データベース・アップグレード時を除きます。

Oracle Database 11g以降、Oracle Data GuardではData Guard構成の柔軟性が高まり、プライマリ・システムとスタンバイ・システムで、異なるCPUアーキテクチャ、オペレーティング・システム(たとえばWindowsとLinux)、オペレーティング・シス

テムのバイナリ(32-bit/64-bit)、またはOracleデータベースのバイナリ(32-bit/64-bit)を使用できるようになりました。

プラットフォームの混在について向上した柔軟性は、<http://support.oracle.com>にあるMy Oracle Supportのノート413484.1および1085687.1で記述されている現行の制限事項の対象となります。

ノート413484.1では、フィジカル・スタンバイに対するプラットフォーム混在のサポートおよび制限事項について説明しています。

ノート1085687.1では、ロジカル・スタンバイに対するプラットフォーム混在のサポートおよび制限事項について説明しています。

関連項目:

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

2.3.2 Oracleソフトウェア要件

Oracle Data Guardを使用するには、特定のOracleソフトウェア要件を満たしている必要があります。

- Oracle Data Guardは、Oracle Database Enterprise Editionの専用機能として提供されています。Oracle Database Standard Editionには提供されていません。

ノート:



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

- 一般に、Oracle Data Guard Standby-First Patchの適用プロセスが進行中でないかぎり、フィジカル・スタンバイ・データベースのデータベース・ホームのバージョン(パッチ・セット例外(PSE)、クリティカル・パッチ・アップデート(CPU)、パッチ・セット更新(PSU)を含む)はプライマリ・データベースと同じである必要があります(<http://support.oracle.com>にあるMy Oracle Supportノート1265700.1を参照)。

さらにOracle Database 11gリリース2 (11.2.0.1)現在、フィジカル・スタンバイ・データベースを使用して、適格な個別パッチ、パッチ・セット更新(PSU)およびクリティカル・パッチ・アップデート(CPU)をローリングという方法でインストールできます。この機能の詳細は、<http://support.oracle.com>のMy Oracle Supportノート1265700.1を参照してください。

- Oracle Data Guard SQL Applyを使用して、Oracle Databaseソフトウェアをパッチ・セット・リリース *n*(10.1.0.3以上)から上位バージョンのパッチ・セットまたはメジャー・バージョン・リリースにローリング・アップグレードできます。ローリング・アップグレード時には、プライマリ・データベースおよびロジカル・スタンバイ・データベースを1つずつアップグレードする間、異なるリリースのOracle Databaseを実行できます。詳細は、[「SQL Applyを使用したOracle Databaseのアップグレード」](#)および該当するOracle Database 10gパッチ・セット・リリースのREADMEファイルを参照してください。

- Oracle Data Guard構成のすべてのデータベースで、COMPATIBLEデータベース初期化パラメータを同じ値に設定する必要がありますが、ロジカル・スタンバイ・データベースは例外で、プライマリ・データベースよりもCOMPATIBLEの設定値を大きくできます。
- 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。詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。
- プライマリ・データベースは、単一インスタンス・データベースまたはOracle Real Application Clusters(Oracle RAC)データベースのいずれかです。スタンバイ・データベースは、単一インスタンス・データベースまたはOracle RACデータベースで、フィジカル・タイプ、ロジカル・タイプおよびスナップショット・タイプを組み合わせることができます。
- プライマリ・データベースとスタンバイ・データベースには、それぞれ独自の制御ファイルが必要です。
- スタンバイ・データベースがプライマリ・データベースと同じシステムにある場合、スタンバイ・データベースのアーカイブ・ディレクトリは、プライマリ・データベースとは異なるディレクトリ構造を使用する必要があります。同じ構造を使用すると、スタンバイ・データベースがプライマリ・データベース・ファイルを上書きする可能性があります。
- プライマリ・データベースに対して、ログに記録されず、スタンバイ・データベースに伝播できないダイレクト書き込みが行われるのを防ぐには、プライマリ・データベースでFORCE LOGGINGをオンにした後で、スタンバイ作成用にデータファイルのバックアップを実行します。スタンバイ・データベースが必要な間は、データベースをFORCE LOGGINGモードに保持してください。
- プライマリおよびスタンバイ・データベース・インスタンスを管理するために使用するユーザー・アカウントは、SYSDBAまたはSYSDBA管理権限を持つ必要があります。
- 動作を簡略化するために、Oracle Automatic Storage Management(Oracle ASM)およびOracle Managed Files(OMF)をOracle Data Guard構成で設定する際は、プライマリおよびスタンバイ・データベースに対称的に設定することをお勧めします。移行やメンテナンス用に意図的に混合構成を実装する場合を除き、Oracle Data Guard構成のいずれかのデータベースでOracle ASM、OMF、またはその両方を使用する場合は、構成内のすべてのデータベースでもそれぞれOracle ASM、OMFまたはその両方を使用する必要があります。詳細は、[「OMFまたはOracle ASMを使用するスタンバイ・データベースの作成」](#)のシナリオを参照してください。

ノート:



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

2.4 スタンバイ・データベースのディレクトリ構造に関する考慮事項

各種スタンバイ・データベースのディレクトリ構造によってスタンバイ・データファイル、アーカイブREDOログ・ファイルおよびスタンバイREDOログ・ファイルのパス名が決まるため、ディレクトリ構造は重要です。

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

関連項目:

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

[図2-1](#)に、3つの基本構成オプションを示します。これには次のものがあります。

- 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にあるが、プライマリ・システムとは異なるディレクトリ構造を使用する。これは、[図2-1](#)のStandby1です。

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

- スタンバイ・データベースが別個のシステム上にあり、プライマリ・システムと同じディレクトリ構造を使用する。これは、[図2-1](#)のStandby2です。このオプションをお勧めします。
- スタンバイ・データベース別個のシステム上にあり、プライマリ・システムと異なるディレクトリ構造を使用する。これは、[図2-1](#)のStandby3です。

ノート:



動作を簡略化するために、Oracle Automatic Storage Management(Oracle ASM)および Oracle Managed Files(OMF)を Oracle Data Guard 構成で設定する際は、プライマリおよびスタンバイ・データベースに対称的に設定することをお勧めします。移行やメンテナンス用に意図的に混合構成を実装する場合を除き、Oracle Data Guard 構成のいずれかのデータベースで Oracle ASM、OMF、またはその両方を使用する場合は、構成内のすべてのデータベースでもそれぞれ Oracle ASM、OMF またはその両方を使用する必要があります。詳細は、[「OMF または Oracle ASM を使用するスタンバイ・データベースの作成」](#)のシナリオを参照してください。

図2-1 可能なスタンバイ構成

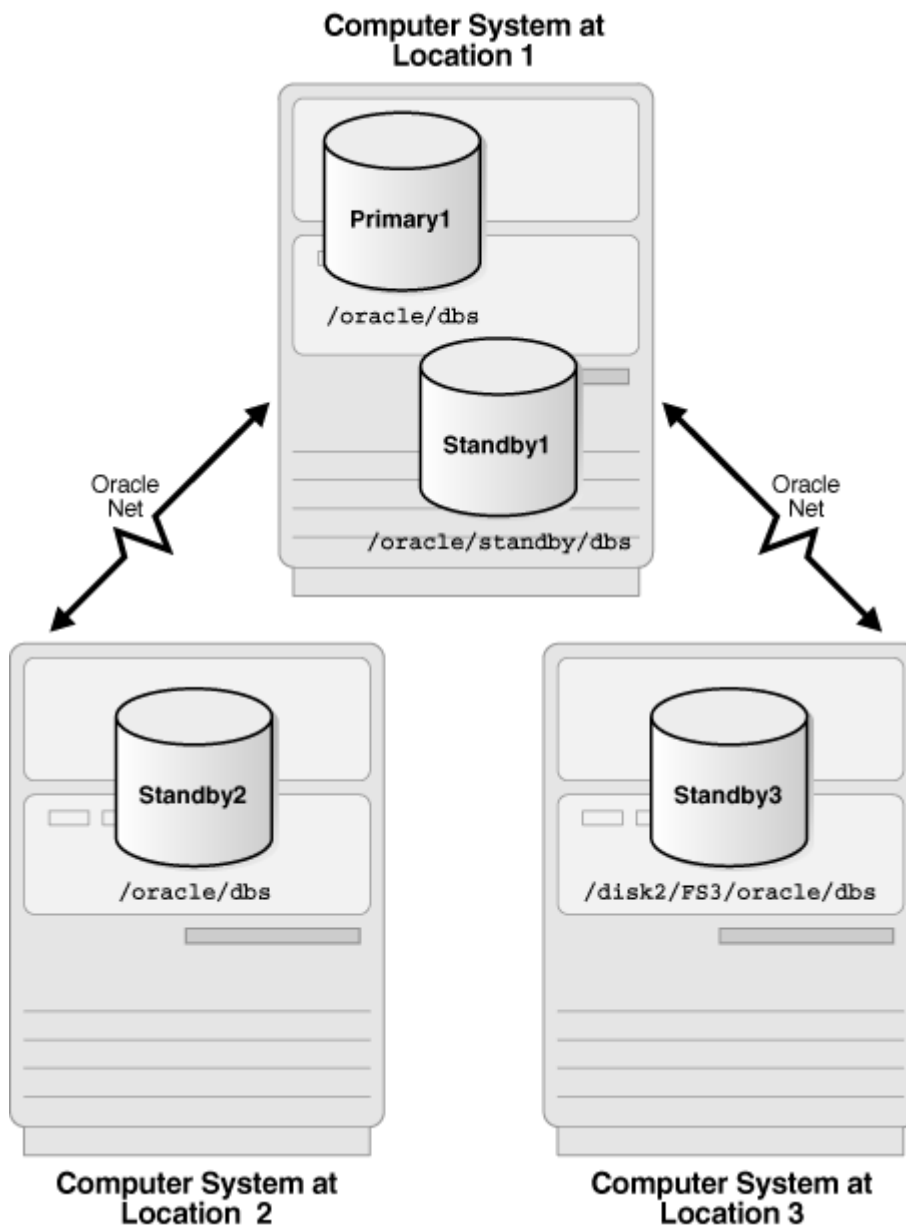


図2-1では、プライマリ・データベースとスタンバイ・データベースの可能な構成およびそれぞれの結果的な注意事項について説明します。

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		
システム	ディレクトリ構造	結果
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)	● ファイル名を手動で変更するか、スタンバイ・データベースで DB_FILE_NAME_CONVERT 初期化パラメータおよび LOG_FILE_NAME_CONVERT 初期化パラメータを設定すると、スタンバイ・データベース制御ファイル内のプライマリ・データベースのデータファイル、アーカイブ REDO ログ・ファイルおよびスタンバイ REDO ログ・ファイルのパス名を自動的に更新できます。(「プライマリ・データベースの初期化パラメータの設定」を参照してください。) ● スタンバイ・データベースは、プライマリ・データベースとスタンバイ・データベースが存在しているシステムを破壊するような障害に対する保護には役立ちませんが、計画的なメンテナンスに対するスイッチオーバー機能は提供します。
離れたシステム	プライマリ・システム	● スタンバイ・データベース制御ファイル内のプライマリ・データベースのファイル名、アーカイブ

ム	ムと同じ	イブ REDO ログ・ファイルおよびスタンバイ REDO ログ・ファイル名を変更する必要はありません。ただし、新しいネーミング計画が必要な場合(複数のディスクにファイルを分散する場合など)は、変更しても構いません。 ● スタンバイ・データベースを別の物理メディアに置くことにより、プライマリ・システムを破壊するような障害からプライマリ・データベースのデータを保護できます。
離れたシステム	プライマリ・システムと異なる	● ファイル名を手動で変更するか、またはスタンバイ・データベースで DB_FILE_NAME_CONVERT 初期化パラメータおよび LOG_FILE_NAME_CONVERT 初期化パラメータを設定すると、データファイル名を自動的に変更できます(「プライマリ・データベースの初期化パラメータの設定」を参照)。 ● スタンバイ・データベースを別の物理メディアに置くことにより、プライマリ・システムを破壊するような障害からプライマリ・データベースのデータを保護できます。

2.5 オンライン・データファイルの場所の移行

データベースがアクティブにファイルにアクセスしている間に、オンライン・データ・ファイルのある物理ファイルから別の物理ファイルへと移行することができます。

ファイルの場所を移行するには、SQL文ALTER DATABASE MOVE DATAFILEを使用します。

ALTER DATABASE MOVE DATAFILE文を使用して実行する操作は、データベースの可用性を向上させます。オンライン・データファイルの場所を移行するのにデータベースを停止する必要がないからです。

データファイルのオンライン移動操作はプライマリおよびスタンバイ(フィジカルまたはロジカル)のいずれもで独立して実行できます。スタンバイはデータファイルがプライマリで移動されても影響を受けず、その逆の場合も同様です。

フィジカル・スタンバイで、データベースをオープンするインスタンスが読み取り専用モードの場合、スタンバイのリカバリの実行中に、データファイルのオンライン移行操作を実行できます。この機能には、Oracle Active Data Guardのライセンスが必要です。

2.5.1 オンライン・データ・ファイルの場所の移行時の制限事項

オンライン・データファイルの場所の移行時には、制限事項があります。

- スタンバイがマウントされていてオープンされていない場合、ファスト・スタート・フェイルオーバー・ターゲットであるフィジカル・スタンバイのオンライン・データファイルの改名または再配置を、SQL ALTER DATABASE MOVE DATAFILEコマンドを使用して行うことはできません。
- マウントされているがオープンされていないインスタンスでのスタンバイのリカバリの実行中、フィジカル・スタンバイで、データファイルのオンライン移行操作は実行できません。
- スタンバイ・リカバリ・プロセスがデータファイルをオフラインで取得、ファイルを縮小、または表領域を削除すると、データファイルのオンライン移行操作は終了します。
- プライマリ・データベースで、プライマリ・データベースのすべてのインスタンスでクローズ済のプラガブル・データベース(PDB)に属するファイルで、データファイルのオンライン移行操作は実行できません。

関連項目:

- オンライン・データファイルの改名および再配置の詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。
- ALTER DATABASE MOVE DATAFILE文の詳細は、[『Oracle Database SQL言語リファレンス』](#)を参照してください。

3 フィジカル・スタンバイ・データベースの作成

非同期REDO転送、リアルタイム適用、デフォルトのOracle Data Guard構成を使用して、最大パフォーマンス・モードのフィジカル・スタンバイ・データベースを手動で作成できます。

次のメイン・トピック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](#)
- [フィジカル・スタンバイ・データベースの作成ステップ](#)
- [作成後のステップ](#)
- [DBCAを使用したData Guardスタンバイの作成](#)
- [CDBのフィジカル・スタンバイの作成](#)
- [プライマリ・データベースでのPDBの作成](#)

関連項目:

- サーバーのパラメータ・ファイルの作成と使用の詳細は、『[Oracle Database管理者ガイド](#)』を参照してください。
- Oracle Data Guard Brokerグラフィカル・ユーザー・インタフェース(GUI)を使用したフィジカル・スタンバイ・データベースの自動作成の詳細は、Enterprise Managerオンライン・ヘルプ・システムを参照してください。
- Oracle Recovery Manager (RMAN)およびバックアップ・ベースの重複またはネットワークを介したアクティブな重複のいずれかを使用して、プロセスの多くを自動化するフィジカル・スタンバイ・データベースを作成する別の方法については、『[Recovery Managerを使用したスタンバイ・データベースの作成](#)』を参照してください。
- Oracle Data Guard Brokerによって管理できるようにデータベースを構成する方法の詳細は、『[Oracle Data Guard Broker](#)』を参照してください。

ノート:



マルチテナント・コンテナ・データベース(CDB)環境で作業している場合、非 CDB 環境との動作の違いの詳細は、『[CDB のフィジカル・スタンバイの作成](#)』を参照してください。たとえば、CDB 環境では、DBA ビューの多くに代替として使用する必要のある類似の CDB ビューがあります。

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

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

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・データベースで次の準備作業を実行します。

- [適切なロギング・モードの有効化](#)
- [REDO転送の認証の構成](#)
- [REDOデータを受信するためのプライマリ・データベースの構成](#)
- [プライマリ・データベースの初期化パラメータの設定](#)

- [アーカイブの有効化](#)

ノート:



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

3.1.1 適切なロギング・モードの有効化

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の一部として、Data Guard構成の使用方法に適したロギング・モードを有効化する必要があります。

Data Guard構成に含まれないデータベースのデフォルトのロギング・モードでは、特定のデータ・ロード操作をログに記録されない方法で実行できます。このデフォルト・モードは、ロードされたデータがスタンバイから欠落することになり、修正には手動操作が必要になるため、スタンバイがあるデータベースには適していません。デフォルトのロギング・モードに加えて、プライマリ・データベースに適したモードが他に3つあります。

- FORCE LOGGINGモードでは、ログに記録されない方法でロード操作が実行されることを防止できます。この場合、ロードされたデータをREDOログにコピーする必要があるため、ロード・プロセスの速度が低下する可能性があります。FORCE LOGGINGモードを有効化するには、次のコマンドを使用します。

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

- STANDBY NOLOGGING FOR DATA AVAILABILITYモードでロード操作を実行すると、ロードされたデータがスタンバイ固有の接続を介して各スタンバイに送信されます。コミットは、すべてのスタンバイがActive Data Guard環境内の管理リカバリ実行の一部としてデータを適用するまで遅延されます。これは、次のコマンドを使用して有効化されます。

```
SQL> ALTER DATABASE SET STANDBY NOLOGGING FOR DATA AVAILABILITY;
```

- STANDBY NOLOGGING FOR LOAD PERFORMANCEは、前述のモードに似ていますが、プライマリにデータをロードする際のネットワーク速度を維持できない場合に、ロード・プロセスによるスタンバイへのデータ送信を停止できる点が異なります。このモードでは、スタンバイからデータが欠落する可能性があります。各スタンバイはActive Data Guard環境内の管理リカバリ実行の通常の一部として、プライマリ・データベースのデータを自動的にフェッチします。これは、次のコマンドを使用して有効化されます。

```
SQL> ALTER DATABASE SET STANDBY NOLOGGING FOR LOAD PERFORMANCE;
```

これらの文のいずれかを発行する場合は、プライマリ・データベースを少なくともマウントしておく(また、オープンもできるようにしておく)必要があります。この文は、完了までに非常に時間がかかる場合があります。これは、ログに記録されないダイレクト書込み I/Oがすべて完了するまで待機するためです。

ノート:



プライマリ・データベースで STANDBY NOLOGGING FOR DATA AVAILABILITY または STANDBY NOLOGGING FOR LOAD PERFORMANCE を有効にすると、複数インスタンスの REDO Apply 機能を使用しているすべてのスタンバイは REDO の適用を停止し、エラーORA-10892 が発生します。まず、REDO Apply を再開し、影響を受けたスタンバイが NOLOGGING 操作期間を超えて進行できるようにしてから、複数インスタンスの REDO Apply を有効化

する必要があります。

関連項目:

- FORCE LOGGINGモード指定の影響の詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。

3.1.2 REDO転送の認証の構成

Oracle Data Guardでは、Oracle Netセッションを使用してOracle Data Guard構成のメンバー間でREDOデータを転送し、メッセージを制御します。

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

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

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

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

ノート:



Oracle Database 12c リリース 2 (12.2.0.1)以降、プライマリ・データベースで行われたパスワード・ファイルの変更は、スタンバイ・データベースに自動的に伝搬されます。唯一の例外は、遠隔同期インスタンスです。遠隔同期インスタンスは REDO を受け取りますが、適用しないため、更新したパスワード・ファイルを引き続き手動で遠隔同期インスタンスにコピーする必要があります。遠隔同期インスタンスでパスワード・ファイルが更新されると、REDO は遠隔同期インスタンスから REDO ログを受け取るように設定されているすべてのスタンバイ・データベースに自動的に伝播されます。REDO が適用されると、スタンバイでパスワード・ファイルが更新されます。

関連項目:

- リモート・ログイン・パスワード・ファイルの詳細は、[『Oracle Database管理者ガイド』](#)および[『Oracle Database! フォレンジック』](#)を参照してください。
- SSLの詳細は、[『Oracle Databaseセキュリティ・ガイド』](#)を参照してください。
- Oracle Net Servicesの詳細は、[『Oracle Database Net Services管理者ガイド』](#)を参照してください。

3.1.3 REDOデータを受信するためのプライマリ・データベースの構成

スタンバイ・データベースが最初に構成に追加されるときに、REDOを受信するようにプライマリ・データベースを構成することがベスト・プラクティスです。

これで、プライマリ・データベースは、スタンバイ・ロールに速やかに推移してREDOデータの受信を開始できるようになります。

スタンバイREDOログを作成するには、SQL ALTER DATABASE ADD STANDBY LOGFILE文を使用します。次に例を示します。

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

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

各ログ・ファイルのサイズとログ・グループの数を決定する方法と、スタンバイREDOログの管理に関するその他の背景情報については、[「REDOデータを受信するためのOracleデータベースの構成」](#)を参照してください。

3.1.4 プライマリ・データベースの初期化パラメータの設定

プライマリ・データベースでは、データベースがプライマリ・ロールで動作している間のREDO転送サービスを制御する初期化パラメータを定義します。

また、プライマリ・データベースがスタンバイ・ロールに推移したときにREDOデータの受信と適用サービスを制御するパラメータを追加する必要があります。

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

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

```
DB_NAME=chicago
DB_UNIQUE_NAME=chicago
LOG_ARCHIVE_CONFIG='DG_CONFIG=(chicago,boston)'
```

```
CONTROL_FILES='/arch1/chicago/control1.ctl', '/arch2/chicago/control2.ctl'
```

```
LOG_ARCHIVE_DEST_1=
'LOCATION=USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
DB_UNIQUE_NAME=chicago'
```

```
LOG_ARCHIVE_DEST_2=
'SERVICE=boston ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=boston'
```

```
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
LOG_ARCHIVE_FORMAT=%t_%s_%r.arc
```

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

次の例は、プライマリ・データベース上のその他のスタンバイ・ロールの初期化パラメータを示しています。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。

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

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

次の表に、前の2つの例に示した各パラメータ設定に関する簡単な説明を示します。

パラメータ	お薦めする設定
DB_NAME	プライマリ・データベースで、データベースが作成されたときに使用された名前を指定します。フィジカル・スタンバイ・データベースで、プライマリ・データベースの DB_NAME を使用します。
DB_UNIQUE_NAME	各データベースの一意の名前を指定します。この名前はデータベースと固定され、プライマリ・データベースとスタンバイ・データベースのロールが逆になっても変更されません。
LOG_ARCHIVE_CONFIG	Oracle Data Guard の機能を完全に有効にするには、このパラメータの DG_CONFIG 属性を Oracle Data Guard 構成のデータベースごとに明示的に設定する必要があります。構成内の各データベース名の DB_UNIQUE_NAME を含み、各名前がカンマで区切られたテキスト文字列に DG_CONFIG を設定します。
CONTROL_FILES	プライマリ・データベースの制御ファイルのパス名を指定します。制御ファイルの別のコピーを用意しておき、問題のない制御ファイルを問題のある制御ファイルの場所にコピーした後、容易にインスタンスを再開できるようにしておくことをお薦めします。
LOG_ARCHIVE_DEST_n	REDO データがアーカイブされるプライマリ・システムおよびスタンバイ・システムの位置を指定します。 ● LOG_ARCHIVE_DEST_1 と指定すると、プライマリ・データベースにより生成された REDO データは、ローカルのオンライン REDO ログ・ファイルから /arch1/chicago/にあるローカルのアーカイブ REDO ログ・ファイルにアーカイブされます。 ● LOG_ARCHIVE_DEST_2 は、プライマリ・ロールにのみ有効です。この宛先を指定すると、REDO データはリモート・フィジカル・スタンバイ宛先 boston に転送されます。 ノート: 高速リカバリ領域を(DB_RECOVERY_FILE_DEST 初期化パラメータを使用して)構成しており、LOCATION 属性でローカル・アーカイブ先を明示的に構成していない場合は、デフォルトのローカル・アーカイブ先として LOG_ARCHIVE_DEST_1 初期化パラメータ

パラメータ	お薦めする設定
	(まだ設定されていない場合)が自動的に使用されます。LOG_ARCHIVE_DEST_ <i>n</i> の完全な情報については、 「LOG_ARCHIVE_DEST_<i>n</i> パラメータの属性」 も参照してください。
REMOTE_LOGIN_PASSWORDFILE	リモート・ログイン・パスワード・ファイルを使用して管理ユーザーまたは REDO 転送セッションを認証する場合は、このパラメータを EXCLUSIVE または SHARED に設定する必要があります。
LOG_ARCHIVE_FORMAT	スレッド(%t)、順序番号(%s)およびリセットログ ID(%r)を使用して、アーカイブ REDO ログ・ファイルのフォーマットを指定します。
FAL_SERVER	FAL サーバー(通常はプライマリ・ロールで実行中のデータベース)の Oracle Net サービス名を指定します。シカゴのデータベースがスタンバイ・ロールで実行されている場合、ボストンのデータベースが欠落しているログ・ファイルを自動的に送信できなければ、ボストンのデータベースが FAL サーバーとして使用され、そこから欠落しているアーカイブ REDO ログ・ファイルがフェッチ(要求)されます。
DB_FILE_NAME_CONVERT	スタンバイ・データベースのデータファイルのパス名とファイル名の位置を指定し、その後にプライマリの位置を指定します。このパラメータにより、プライマリ・データベースのデータファイルのパス名がスタンバイ・データベースのパス名に変換されます。このパラメータは、フィジカル・スタンバイ・データベースのパス名の変換にのみ使用されます。このパラメータにより、複数のペアのパスを指定できます。
LOG_FILE_NAME_CONVERT	スタンバイ・データベースのオンライン REDO ログ・ファイルの位置を指定し、その後にプライマリの位置を指定します。このパラメータにより、プライマリ・データベースのログ・ファイルのパス名がスタンバイ・データベースのパス名に変換されます。このパラメータにより、複数のペアのパスを指定できます。
STANDBY_FILE_MANAGEMENT	データファイルがプライマリ・データベースに追加または削除された場合に、それに対応してスタンバイ・データベースも自動的に変更されるように、AUTO に設定します。

ノート:



変更が必要なその他のパラメータがないか、初期化パラメータ・ファイルを確認してください。たとえばスタンバイ・データベース上のディレクトリ場所がプライマリ・データベースでの指定と異なる場合は、ダンプ先パラメータを変更する必要があります。

3.1.5 アーカイブの有効化

アーカイブが有効になっていない場合は、プライマリ・データベースをARCHIVELOGモードにし、自動アーカイブを有効にする必要が

あります。

次のSQL文を発行します。

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



ノート:

スタンバイ REDO ログ・アーカイブを実行するには、スタンバイ・データベースが ARCHIVELOG モードである必要があります。

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

3.2 フィジカル・スタンバイ・データベースの作成ステップ

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を次に示します。

次のトピックを十分に理解していることが求められます。

- データベース管理者の認証
- データベース初期化パラメータ
- REDOログ、データファイルおよび制御ファイルの管理
- アーカイブREDOログの管理
- 高速リカバリ領域
- Oracle Net構成

[表3-1](#)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。

表3-1 フィジカル・スタンバイ・データベースの作成

タスク	データベース
プライマリ・データベース・データファイルのバックアップ・コピーの作成	プライマリ
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
スタンバイ・データベース用のパラメータ・ファイルの作成	プライマリ
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
スタンバイ・データベースをサポートする環境の設定	スタンバイ

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

3.2.1 フィジカル・スタンバイの作成 タスク1: プライマリ・データベース・データファイルのバックアップ・コピーの作成

データベースを完全にリカバリするのに必要なアーカイブREDOログ・ファイルがあるかぎり、プライマリ・データベースのバックアップ・コピーを使用して、フィジカル・スタンバイ・データベースを作成できます。

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

データベース・バックアップ操作の実行の詳細は、[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)を参照してください。

3.2.2 フィジカル・スタンバイの作成 タスク2: スタンバイ・データベース用の制御ファイルの作成

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(プライマリ・データベースは開いている必要はありませんが、マウントされている必要があります)。

次に、スタンバイ・データベースの制御ファイルを作成する例を示します。

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

ALTER DATABASEコマンドはスタンバイ・ロールで動作するデータベースを指定します。この場合、bostonという名前のデータベースです。

プライマリ・データベースとスタンバイ・データベースの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。それぞれに独自のファイルが必要です。

ノート:



制御ファイル・バックアップがプライマリで作成され、スタンバイでリストアされる(またはその逆)場合、リストアされたシステム上のスナップショット制御ファイルの場所は、デフォルトになるように構成されます。(スナップショット制御ファイル名のデフォルト値はプラットフォーム固有であり、Oracle ホームに依存します。)RMAN の CONFIGURE SNAPSHOT CONTROLFILE コマンドを使用して、手動で正しい値に再構成します。

3.2.3 フィジカル・スタンバイの作成 タスク3: スタンバイ・データベース用のパラメータ・ファイルの作成

プライマリ・データベースで使用されるサーバー・パラメータ・ファイル(SPFIL)からパラメータ・ファイル(PFIL)を作成します。

次のステップを実行します。

1. 次のようなSQL文を発行します。

```
SQL> CREATE PFIL=' /tmp/initboston.ora' FROM SPFIL;
```

[スタンバイ・データベースをサポートする環境の設定](#)では、このパラメータ・ファイルをフィジカル・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が含まれるように変更した後で、このファイルからサーバー・パラメータ・ファイルを作成します。

2. 前のステップで作成したパラメータ・ファイルのパラメータ値を変更します。

パラメータ・ファイル内の初期化パラメータ設定の大部分はフィジカル・スタンバイ・データベースにも適していますが、一部変更が必要なものがあります。

[例3-1](#)では、プライマリで前に作成されたパラメータで、変更する必要のあるものが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=chicago
DB_UNIQUE_NAME=boston
LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, boston)'
CONTROL_FILES=' /arch1/boston/control1.ctl', ' /arch2/boston/control2.ctl'
DB_FILE_NAME_CONVERT=' /chicago/', ' /boston/'
LOG_FILE_NAME_CONVERT=' /chicago/', ' /boston/'
LOG_ARCHIVE_FORMAT=log%t_%s_%r.arc
LOG_ARCHIVE_DEST_1=
' LOCATION=USE_DB_RECOVERY_FILE_DEST
  VALID_FOR=(ALL_LOGFILES, ALL_ROLES)
  DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_2=
' SERVICE=chicago ASYNC
  VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE)
  DB_UNIQUE_NAME=chicago'
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
STANDBY_FILE_MANAGEMENT=AUTO
FAL_SERVER=chicago
.
.
.
```

プライマリ・データベースとスタンバイ・データベースでは、COMPATIBLE初期化パラメータを同じ値に設定する必要があります。値が異なる場合、REDO転送サービスがREDOデータをプライマリ・データベースからスタンバイ・データベースに転送できない可能性があります。

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

次の表に、[例3-1](#)に示したパラメータ設定のうち、プライマリ・データベースとは異なる設定に関する簡単な説明を示します。

パラメータ	お勧めする設定

パラメータ	お薦めする設定
DB_UNIQUE_NAME	このデータベースの一意の名前を指定します。この名前はデータベースと固定され、プライマリ・データベースとスタンバイ・データベースのロールが逆になっても変更されません。
CONTROL_FILES	スタンバイ・データベースの制御ファイルのパス名を指定します。 例 3-1 は 2 つの制御ファイルに対してこれを行う方法を示しています。制御ファイルの別のコピーを用意しておき、問題のない制御ファイルを問題のある制御ファイルの場所にコピーした後、容易にインスタンスを再開できるようにしておくことをお薦めします。
DB_FILE_NAME_CONVERT	プライマリ・データベースのデータファイルのパス名とファイル名の位置を指定し、その後スタンバイの位置を指定します。このパラメータにより、プライマリ・データベースのデータファイルのパス名がスタンバイ・データファイルのパス名に変換されます。
LOG_FILE_NAME_CONVERT	プライマリ・データベースのオンライン REDO ログ・ファイルの位置を指定し、その後スタンバイの位置を指定します。このパラメータにより、プライマリ・データベースのログ・ファイルのパス名がスタンバイ・データベースのパス名に変換されます。
LOG_ARCHIVE_DEST_n	REDO データのアーカイブ先を指定します。例 3-1 では、次のようになっています。 ● LOG_ARCHIVE_DEST_1 を指定すると、プライマリ・データから受信した REDO データは/arch1/boston/のアーカイブ REDO ログ・ファイルにアーカイブされます。 ● LOG_ARCHIVE_DEST_2 はプライマリ・ロールのみに有効であるため、この宛先は現在は無視されます。スイッチオーバーが発生して、このインスタンスがプライマリ・データベースになる場合は、REDO データがリモートのシカゴの宛先に転送されます。 ノート: 高速リカバリ領域を(DB_RECOVERY_FILE_DEST 初期化パラメータを使用して)構成しており、LOCATION 属性でローカル・アーカイブ先を明示的に構成していない場合は、デフォルトのローカル・アーカイブ先として LOG_ARCHIVE_DEST_1 初期化パラメータ(まだ設定されていない場合)が自動的に使用されます。 LOG_ARCHIVE_DEST_n の完全な情報については、LOG_ARCHIVE_DEST_n パラメータの属性も参照してください。
FAL_SERVER	FAL サーバー(通常はプライマリ・ロールで実行中のデータベース)の Oracle Net サービス名を指定します。ボストンのデータベースがスタンバイ・ロールで実行されている場合、シカゴのデータベースが欠落しているログ・ファイルを自動的に送信できなければ、シカゴのデータベースが FAL サーバーとして使用され、そこから欠落しているアーカイブ REDO ログ・ファイルがフェッチ(要求)されます。

ノート:



変更が必要なその他のパラメータがないか、初期化パラメータ・ファイルを確認してください。たとえばスタンバイ・データベース上のディレクトリ場所がプライマリ・データベースでの指定と異なる場合は、ダンプ先パラメータを変更する必要があります。

3.2.4 フィジカル・スタンバイの作成 タスク4: プライマリ・システムからスタンバイ・システムへのファイルのコピー

必要なディレクトリがすべて作成されていることを確認します。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、プライマリ・システムのバイナリ・ファイルをスタンバイ・システムの正しい場所にコピーします。

コピーするバイナリ・ファイルは次のとおりです。

- [「プライマリ・データベース・データファイルのバックアップ・コピーの作成」](#)で作成されたデータベース・バックアップ
- [「スタンバイ・データベース用の制御ファイルの作成」](#)で作成されたスタンバイ制御ファイル
- [「スタンバイ・データベース用のパラメータ・ファイルの作成」](#)で作成された初期化パラメータ・ファイル

3.2.5 フィジカル・スタンバイの作成 タスク5: スタンバイ・データベースをサポートする環境の設定

環境設定として、Windowsベース・サービス、パスワード・ファイルおよびSPFILEを作成し、Oracle Net環境を設定します。次のステップを実行します。

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

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

ORADIMユーティリティは、このサービスを作成する必要があるユーザー名を自動的に決定し、そのユーザー名のパスワードを求めます(そのユーザー名がパスワードを必要とする場合)。ORADIMユーティリティの使用の詳細は、[『Oracle Databaseプラットフォーム・ガイドfor Microsoft Windows』](#)を参照してください。

2. リモート・ログイン・パスワード・ファイルをプライマリ・データベース・システムからスタンバイ・データベース・システムにコピーします。

このステップは、オペレーティング・システムの認証を管理ユーザーに使用する場合およびSSLをREDO転送の認証に使用する場合はオプションです。いずれの場合でもない場合は、リモート・ログイン・パスワード・ファイルをプライマリ・データベースからフィジカル・スタンバイ・データベース・システムの適切なディレクトリにコピーします。

プライマリでのパスワード・ファイルへのその後の変更は、スタンバイに自動的に伝播されます。管理権限(SYSDG、SYSOPER、SYSDBAなど)が付与または解除されたとき、および管理権限を持つユーザーのパスワードが変更されたときのパスワード・ファイルの変更が含まれます。遠隔同期インスタンスはREDOを受け取りますが、適用しないため、更新したパスワード・ファイルを引き続き手動で遠隔同期インスタンスにコピーする必要があります。遠隔同期インスタンスでパスワード・ファイルが更新されると、プライマリでのパスワードの更新を含むREDOは、遠隔同期インスタンスからREDOログを受け取るように設定されているすべてのスタンバイ・データベースに自動的に伝播されます。REDOが適用されると、スタンバイでパスワード・ファイルが更新されます。

3. 構成されていない場合、スタンバイ・システムにリスナーを構成して、開始します。

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

4. Oracle Netサービス名を作成します。

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

Oracle Netネットワーク・サービス名は、プライマリ・データベースとスタンバイ・データベースに対するリスナーの構成時に指定したものと同一プロトコル、ホスト・アドレス、ポートおよびサービスを使用する接続記述子に解決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があります。

サービス名の詳細は、[『Oracle Database Net Services管理者ガイド』](#)を参照してください。

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

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

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

ノート:

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

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

3.2.6 フィジカル・スタンバイの作成 タスク6: フィジカル・スタンバイ・データベースの起動

フィジカル・スタンバイ・データベースおよびREDO Applyを起動するステップを次に示します。

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

```
SQL> STARTUP MOUNT;
```

2. [「プライマリ・データベース・データファイルのバックアップ・コピーの作成」](#)で取得し、[「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」](#)でスタンバイ・システムにコピーしたデータファイルのバックアップをリストアップします。
3. スタンバイ・データベースで、次のコマンドを発行してREDO Applyを開始します。

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

この文にはDISCONNECT FROM SESSIONオプションが指定されているため、REDO Applyはバックグラウンド・セッションで実行されます。

3.2.7 フィジカル・スタンバイの作成 タスク7: フィジカル・スタンバイ・データベースが正しく実行されているかどうかの確認

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

スタンバイ・データベースで、V\$DATAGUARD_PROCESSビューを問い合わせ、REDOがプライマリ・データベースから送信され、スタンバイ・データベースに適用されていることを確認します。

```
SQL> SELECT ROLE, THREAD#, SEQUENCE#, ACTION FROM V$DATAGUARD_PROCESS;
```

ROLE	THREAD#	SEQUENCE#	ACTION
RFS ping	1	9	IDLE
recovery apply slave	0	0	IDLE
recovery apply slave	0	0	IDLE
managed recovery	0	0	IDLE
recovery logmerger	1	9	APPLYING_LOG
RFS archive	0	0	IDLE
RFS async	1	9	IDLE

recovery logmergerロールは、REDOがスタンバイに適用されていることを示しています。

ノート:



V\$MANAGED_STANDBY ビューは Oracle Database 12c リリース 2 (12.2.0.1) から非推奨となっており、将来のリリースでサポートされなくなる可能性があるため、かわりに V\$DATAGUARD_PROCESS ビューを使用することをお勧めします。

3.3 フィジカル・スタンバイの作成: 作成後のステップ

フィジカル・スタンバイ・データベースの実行後、データ保護モードをアップグレードできます。また、フラッシュバック・データベースを有効化できます。

- データ保護モードのアップグレード

各種データ保護モードの構成の詳細は、[「Oracle Data Guardの保護モード」](#)を参照してください。

- フラッシュバック・データベースの有効化

フラッシュバック・データベースにより、フェイルオーバー後のプライマリ・データベースの再作成の必要がなくなります。フラッシュバック・データベースでは、データファイルをバックアップからリストアしたりREDOデータを広範囲に適用する必要がないため、従来のPoint-in-Timeリカバリよりもはるかに高速でデータベースを過去の最新時点の状態に戻すことができます。フラッシュバック・データベースは、プライマリ・データベースまたはスタンバイ・データベース、あるいはその両方で有効化できます。

関連項目:

- Oracle Data Guard環境でのフラッシュバック・データベースの使用例は、[「フラッシュバック・データベースを使用した」](#)

[障害が発生したプライマリ・スタンバイ・データベースへの変換](#)および[Open Resetlogs文の発行後のフラッシュバック・データベースの使用](#)を参照してください。

- データベースのフラッシュバックの詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

3.4 DBCAを使用したData Guardスタンバイの作成

Database Configuration Assistant (DBCA)も、簡単なコマンドラインによる方法としてOracle Data Guardのフィジカル・スタンバイ・データベースを作成するために使用できます。

フィジカル・スタンバイ・データベースの作成に使用されたDBCAコマンド・クオリファイアは、createDuplicateDBです。

基本のcreateDuplicateDBコマンドの構文は次のとおりです。

```
dbca -createDuplicateDB
      -gdbName global_database_name
      -primaryDBConnectionString easy_connect_string_to_primary
      -sid database_system_identifier
      [-createAsStandby
        [-dbUniqueName db_unique_name_for_standby]]
      [-customScripts scripts_list]
```

createDuplicateDBオプションの詳細は、カスタム・スクリプトの使用を含めて、[Oracle Database管理者ガイド](#)を参照してください。

次の2つの例では、プライマリ・データベースはchicagoでプライマリ・システムmyprimary.domain上にあります。それぞれの例では、コマンドが実行されるシステムにフィジカル・スタンバイbostonが作成されます。initParamsパラメータは、例の中では、他のDBCAパラメータをスタンバイ作成コマンドでどのように使用できるかを示すために使用されています。これらの例では、initParamsはスタンバイのINSTANCE_NAMEをDB_UNIQUE_NAMEと一致するように明示的にbostonに設定するために使用されています。

この最初の例では、後で実行されるカスタム・スクリプトなしでスタンバイ・スクリプトが作成されます。

```
dbca -silent -createDuplicateDB -primaryDBConnectionString myprimary.domain:1523/chicago.domain
      -gdbName chicago.domain -sid boston -initParams instance_name=boston -createAsStandby
```

```
Enter SYS user password:
Listener config step
33% complete
Auxiliary instance creation
66% complete
RMAN duplicate
100% complete
Look at the log file " /u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/dbca/chicago/chicago.log"
for further details.
```

次の例は、/tmp/test.sqlという名前のSQLスクリプトを実行する点を除いて、前の例とまったく同じです。このスクリプトは作成後の操作を実行するために使用できます。

```
dbca -silent -createDuplicateDB -primaryDBConnectionString myprimary.domain:1523/chicago.domain
      -gdbName chicago.domain -sid boston -initParams instance_name=boston -createAsStandby -customScripts
/tmp/test.sql
```

```
Enter SYS user password:
```

```
Listener config step
25% complete
Auxiliary instance creation
50% complete
RMAN duplicate
75% complete
Running Custom Scripts
100% complete
Look at the log file " /u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/dbca/chicago/chicago.log"
for further details.
```

ノート:



フィジカル・スタンバイ・システム上で稼働するリスナーが必要な場合でも、どちらのシステムにもこれらのコマンドを実行するためにデータベースの Oracle Net サービス名を構成する必要はありません。これらの例では、プライマリ・データベース Chicago への接続を作成するためと、スタンバイ Boston の作成を完了するために Easy Connect ネーミング・メソッドが使用されました。新しいスタンバイを Data Guard 構成に追加する前に、[フィジカル・スタンバイの作成](#)
[タスク 5: スタンバイ・データベースをサポートする環境の設定](#)のステップ 4 に説明されているように、最初に Oracle Net サービス名ディスクリプタを両方のシステムに構成します。

これらのコマンドがエラーなしで完了すると、フィジカル・スタンバイ Boston は Data Guard 構成に追加する準備ができています。追加の一部として、[フィジカル・スタンバイ タスク 3: スタンバイ・データベース用のパラメータ・ファイルの作成](#)に示されているように Data Guard パラメータを Chicago および Boston に定義する必要があります。オプションで、Oracle Data Guard Broker 構成がある場合は、ブローカの ADD DATABASE コマンドを使用して新しいスタンバイを構成に追加できます ([Oracle Data Guard Broker](#) を参照)。

3.5 CDB のフィジカル・スタンバイの作成

通常のプライマリ・データベースのフィジカル・スタンバイを作成できるように、マルチテナント・コンテナ・データベース (CDB) のフィジカル・スタンバイを作成できます。

CDB のフィジカル・スタンバイの作成および使用時に注意する動作の違いの一部を次に示します。

- 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する場合、CDB 全体でロールが変化します。
ENABLED_PDBS_ON_STANDBY 初期化パラメータを使用した場合、すべての PDB がプライマリおよびスタンバイ・データベースに存在するとは限らない可能性に留意してください。
- データベース・ロールは個別のコンテナ・レベルではなく、CDB レベルで定義されます。
- ロールの変更に関連する DDL は、ロールが CDB 全体に関連するため、ルート・コンテナで実行される必要があります。
個別のプラガブル・データベース (PDB) には固有のロールはありません。
- CDB のフィジカル・スタンバイで、SQL 文の構文は通常、コンテナでないデータベースと同じです。しかし、次に示すものを
含む、一部の文の影響は異なる場合があります。
 - ALTER DATABASE RECOVER MANAGED STANDBY はルート・コンテナでのみ機能し、PDB では使用できません。
 - 1 つのロールは CDB 全体に関連します。個別の PDB には固有のロールはありません。そのため、フィジカル・スタンバイに関連する次のロール変更 DDL は CDB 全体に影響します。

```
ALTER DATABASE SWITCHOVER TO target_db_name
```

```
ALTER DATABASE ACTIVATE PHYSICAL STANDBY
```

- ALTER PLUGGABLE DATABASE [OPEN|CLOSE] SQL文は、ルート・コンテナをすでにオープンしているならば、スタンバイでサポートされます。
- ALTER PLUGGABLE DATABASE RECOVER文はスタンバイでサポートされません。(スタンバイ・リカバリは常にCDBレベルです。)
- マルチテナント環境を管理するには、CDB_DBAロールが必要です。
- スタンバイ・データベースには固有のキーストアを含めることをお勧めします。
- マルチテナント環境では、REDOはスタンバイ・データベースのルート・コンテナに送信される必要があります。

REDOがルート・コンテナに送信されているかどうかを判断する方法の例を次に示します。プライマリ・データベースが次の設定であると想定します。

```
LOG_ARCHIVE_DEST_2=' SERVICE=boston ASYNC VALID_FOR=(ONLINE_LOGFILES,
PRIMARY_ROLE) DB_UNIQUE_NAME=boston'
```

REDOはbostonに送信されています。ルート・コンテナのコンテナID (CON_ID)は常に1のため、サービスbostonではCON_IDが1となるようにする必要があります。これを行うには、tnsnames.oraファイルのサービス名を確認します。次に例を示します。

```
boston = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=boston-server) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=boston.us.example.com))
```

bostonのサービス名は、boston.us.example.comです。

スタンバイ・データベースでCDB_SERVICESビューを問い合わせ、CON_IDを確認します。次に例を示します。

```
SQL> SELECT NAME, CON_ID FROM CDB_SERVICES;
```

NAME	CON_ID
boston.us.example.com	1

問い合わせの結果、bostonのCON_IDが1であると示されます。

関連項目:

- CDBの詳細は、[『Oracle Database概要』](#)を参照してください。
- キーストアの作成の詳細は、[『Oracle Database Advanced Securityガイド』](#)を参照してください。

3.6 プライマリ・データベースでのPDBの作成

Oracle Data Guard構成では、プライマリ・データベースのプラガブル・データベース(PDB)は、通常のデータベースのPDBを作成するのと同じ方法で作成します。

この項では、任意のタイプのPDBを作成したときにData GuardとOracle Multitenantオプションの間で行われるやり取りについての要点を示します。詳細な情報と例は、<http://support.oracle.com>にあるMy Oracle Supportノート 2049127.1を参照してください。

通常のデータベースでPDBを作成するステップは、[Oracle Multitenant管理者ガイド](#)を参照してください。これらのステップに従う前に、次の説明を読んでください。

- PDBのサブセットをマルチテナント・コンテナ・データベース(CDB)のフィジカル・スタンバイに複製するように指定できます。このためには、ENABLED_PDBS_ON_STANDBY初期化パラメータを使用してPDBのリストを指定するか、STANDBYS句をCREATE PLUGGABLE DATABASE文で使用するか、あるいはその両方を実行します。

ENABLED_PDBS_ON_STANDBYパラメータ([ENABLED_PDBS_ON_STANDBY](#)を参照)は、フィジカル・スタンバイでのみ有効であり、プライマリ・データベースでは無視されます。(プライマリ・データベースに設定して使用できるのは、そのデータベースがスタンバイ・データベースになった場合のみです。)これは、どのPDBをフィジカル・スタンバイ・データベースで有効化するかしないかを指定するために使用できます。このパラメータが指定されていない場合、STANDBYS句が使用されていないかぎりCDB内のすべてのPDBがスタンバイ上に作成されます。

STANDBYS句は、リカバリが後で有効になるか無効になるかに影響しません。SQL CREATE PLUGGABLE DATABASE文のSTANDBYS句の構文は次のとおりです。

```
CREATE PLUGGABLE DATABASE ... STANDBYS= ((' cdb_name', ' cdb_name', ...) | NONE | ALL [EXCEPT (' cdb_name', ' cdb_name', ...)])
```

- cdb_nameは、PDBを含めるフィジカル・スタンバイ用のDB_UNIQUE_NAMEです。
- NONEは、作成中のPDBをすべてのスタンバイCDBから除外します。PDBがすべてのスタンバイCDBから除外される場合、PDBのデータ・ファイルはオフラインになり、すべてのスタンバイCDBで名前なしとしてマークされます。PDBが作成された後にインスタンス化された新しいスタンバイCDBは、リカバリのために明示的にPDBを無効にし、スタンバイCDBから除外する必要があります。PDBをスタンバイCDBで除外した後に、そのスタンバイCDBで有効にすることができます。
- ALL(デフォルト)は、作成中のPDBをすべてのスタンバイCDBに含めます。
- EXCEPT cdb_nameは、作成中のPDBを、DB_UNIQUE_NAMEでこの句にリストされたCDBを除いてすべてのスタンバイCDBに含めます。

STANDBYS句にCDB名のリストを指定する場合は、リストをカッコで囲み、それぞれの名前を一重引用符で囲む必要があります。DB_UNIQUE_NAMEの値には最大30文字を指定できます。大/小文字は区別されません。データベース名で有効な文字は、英数字、アンダースコア(_)、シャープ記号(#)およびドル記号(\$)です。

PDBがシードから作成されるか、Active Data Guardを実行しているスタンバイのローカル・クローンとして作成されるか、またはSTANDBY_PDB_SOURCE_FILE_DBLINKパラメータを使用してリモート・クローンとして作成される場合を除き、リスト内のスタンバイには必要なファイルが事前移入されている必要があります。

- PDBを別のPDBまたは同じプライマリCDB内のシードPDBからローカル・クローンとして作成する場合は、次の点に注意してください。
 - シードから新しいPDBを作成すると、スタンバイはファイルを自動的にインスタンス化します。
 - Active Data Guard環境では、PDBが同じCDB内でクローニングされたときに、Data Guardがファイルを自動的にインスタンス化します。
 - 非Active Data Guard環境では、PDBが同じCDB内でクローニングされたときに、CREATE PLUGGABLE DATABASE文でSTANDBYS句を使用し、後でリカバリを有効化します。
- PDBのリモート・クローンまたはプラグインの実行時にスタンバイ・データベースを自動的に維持するには、次の初期化パラメータを使用します。
 - STANDBY_PDB_SOURCE_FILE_DBLINK - リモート・クローニングの場合に使用します。プライマリのPDBをクローニングするために使用するデータベース・リンクの名前を指定します。PDBのクローン操作に対するREDOがスタンバイに適用されると、スタンバイはそのデータベース・リンクを使用してソース・データベースに接続し、接続先のスタンバイ・データベースに対してクローニング・プロセスを繰り返します。このパラメータは、データベース・

ディクショナリにアクセスしてリンクの詳細情報を取得するActive Data Guardでのみ使用できます。

クローニング操作のソースPDBは、プライマリへのクローニングとスタンバイへのクローニングのどちらの期間中も読み取り専用モードである必要があります。

STANDBY_PDB_SOURCE_FILE_DBLINKパラメータを使用しないリモート・クローンでは、PDBの作成に使用するSQL CREATE PLUGGABLE DATABASE文でSTANDBYS=NONE句を使用する必要があります。

- STANDBY_PDB_SOURCE_FILE_DIRECTORY - プラグインの場合に使用します。繰り返し実行されるプラグインで使用されるPDBのファイルが格納されるディレクトリの場所を指定します。PDBプラグイン操作のREDOがスタンバイに適用されると、スタンバイはデータ・ファイルのディレクトリの場所を検索し、検出後、DB_CREATE_FILE_DESTおよびDB_FILE_NAME_CONVERTパラメータの設定に基づいて、ファイルをスタンバイ上の必要な場所にコピーします。



ノート:

ファイルは、ソース PDB データ・ファイルのイメージ・コピーである必要があります。このコピーは、ソース PDB のマニフェスト XML ファイルの作成後に作成する必要があります。

これらのパラメータは、使用して不要になった後にリセットする必要があります。これらのパラメータを設定およびリセットするためにインスタンスを再起動する必要はないため、この操作による実行時の影響はありません。

これらのパラメータの使用方法的詳細は、<http://support.oracle.com>にあるMy Oracle Supportノート 2274735.1を参照してください。

- PDBをXMLファイルから作成するには、XMLファイルで指定されたデータ・ファイルをスタンバイ・データベースにコピーします。

PDBをXMLファイルからプライマリ・データベースにプラグインするときは、構成内のすべてのスタンバイ・データベースにソースPDBのファイルへのアクセス権を付与する必要があります。(この唯一の例外は、CREATE PLUGGABLE DATABASE文のSTANDBYS句またはENABLED_PDBS_ON_STANDBY初期化パラメータのいずれかを使用してリカバリが延期されたスタンバイです。)

構成内のスタンバイにソースPDBのファイルへのアクセス権を設定するには、STANDBY_PDB_SOURCE_FILE_DIRECTORY初期化パラメータを使用するか、ソースPDBのファイル(通常はプライマリ・データベースにプラグインされる同じPDBファイル・セット)をスタンバイ・サイトにコピーします。管理スタンバイ・リカバリの中断を最小限に抑えるため、プライマリでプラグイン操作を実行する前にファイルをスタンバイにコピーする必要があります。次のようにして、管理スタンバイ・リカバリで検出できるスタンバイの適切な場所にファイルがコピーされていることを確認します。

- データ・ファイルが標準的なオペレーティング・システムのファイル・システムに存在する場合、スタンバイ・データベースでのファイルの場所は、DB_FILE_NAME_CONVERTパラメータの値に基づいています。プライマリ・データベース初期化パラメータの設定の詳細は、[「プライマリ・データベースの初期化パラメータの設定」](#)を参照してください
- データ・ファイルがASMに存在する場合、ASMCMDユーティリティを使用して、ファイルをスタンバイ・データベースの次の場所にコピーします。

```
db_create_file_dest/db_unique_name/GUID/datafile
```

GUIDパラメータは、PDBに割り当てられるグローバル一意識別子で、一度割り当てられると変わりません。GUIDパラメータの値を確認するには、元のソース・コンテナからPDBを切断する前に、V\$CONTAINERSビューを

問い合わせます。次の例は、ソース・コンテナのPDBコンテナIDが3であるPDBのGUIDパラメータ値の確認方法を示しています。

```
SELECT guid
  FROM V$CONTAINERS
 WHERE con_id=3;
```

GUID

D98C12257A951FC4E043B623F00A7AF5

この例では、DB_CREATE_FILE_DESTパラメータの値が+DATAFILEで、DB_UNIQUE_NAMEパラメータの値がBOSTONである場合、データ・ファイルが次の場所にコピーされます。

+DATAFILE/BOSTON/D98C12257A951FC4E043B623F00A7AF5/datafile

スタンバイ・データベースのデータ・ファイルのパス名は、プライマリでPDBを作成した結果のパス名と同じである必要があります。ただし、DB_FILE_NAME_CONVERTデータベース初期化パラメータがスタンバイで構成されている場合を除きます。この場合、スタンバイ・データベースのデータ・ファイルのパス名は、DB_FILE_NAME_CONVERTが適用されたプライマリでのパス名です。

関連項目:

- SQL文CREATE PLUGGABLE DATABASEの詳細は、[『Oracle Database SQL言語リファレンス』](#)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

ロジカル・スタンバイ・データベースを作成するときは、前提条件と作成後のタスクを含むいくつかの必要なステップがあります。

ロジカル・スタンバイ・データベースの作成の詳細は、次の各項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要件](#)
- [ロジカル・スタンバイ・データベースの作成ステップ](#)
- [作成後のステップ](#)
- [CDBのロジカル・スタンバイの作成](#)

関連項目:

- サーバーのパラメータ・ファイルの作成と使用の詳細は、『[Oracle Database管理者ガイド](#)』を参照してください。
- Oracle Data Guard Brokerグラフィカル・ユーザー・インタフェース(GUI)を使用したロジカル・スタンバイ・データベースの自動作成の詳細は、Oracle Enterprise Manager Cloud Controlオンライン・ヘルプ・システムを参照してください。

ノート:



マルチテナント・コンテナ・データベース(CDB)環境で作業している場合、非 CDB 環境との動作の違いの詳細は、[「CDB のロジカル・スタンバイの作成」](#)を参照してください。たとえば、CDB 環境では、DBA ビューの多くに代替として使用する必要のある類似の CDB ビューがあります。たとえば、ビューCDB_LOGSTDBY_NOT_UNIQUE には、DBA_LOGSTDBY_NOT_UNIQUE ビューに示されるデータと同じデータが含まれますが、PDB 名を含む追加の列があります。

4.1 ロジカル・スタンバイ・データベースの作成要件

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

ロジカル・スタンバイ・データベースを作成する前に、プライマリ・データベースで次の準備作業を実行します。

- [表のデータ型および記憶域属性のサポートの判別](#)
- [プライマリ・データベース内の表の行が一意に識別できることの確認](#)

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

ノート:



スタンバイ REDO ログ・アーカイブを実行するには、ロジカル・スタンバイ・データベースを ARCHIVELOG モードで実行する必要があります。

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

4.1.1 表のデータ型および記憶域属性のサポートの判別

ロジカル・スタンバイ・データベースを設定する前に、ロジカル・スタンバイ・データベースが、プライマリ・データベースのデータ型と表を保持できることを確認する必要があります。

サポートされていないデータ型またはストレージ属性があるかどうかを判断するのに使用できる特定のSQL問合せの詳細は、[\[サポートされない表\]](#)を参照してください。

4.1.2 プライマリ・データベース内の表の行が一意に識別できることの確認

プライマリ・データベースによって生成されたREDOレコードに含まれているROWIDは、ロジカル・スタンバイ・データベース内の対応する行を識別するためには使用できません。

これは、ロジカル・スタンバイ・データベースがプライマリ・データベースのバックアップ・コピーから作成されていても、ロジカル・スタンバイ・データベースの物理的な構成は、プライマリ・データベースの構成とは異なるためです。

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

- 表に主キーが定義されている場合、その主キーが、変更された行を識別するためにUPDATE文の一部として変更された列とともにログに記録されます。
- 主キーがない場合は、一番短く、NULL値が許容されていない一意制約/索引が、変更された行を識別するためにUPDATE文の一部として変更された列とともにログに記録されます。
- 主キーもNULL値が許容されていない一意制約/索引もない場合、4000バイトの最大長が宣言されたすべての列が、変更された行を識別するためにUPDATE文の一部としてログに記録されます。サポートされるデータ型に関する要件と制限がいくつかあります。詳細は、次の各項を参照してください。
 - [サポートされる表記憶域タイプ](#)
 - [サポートされない表記憶域タイプ](#)
- ファンクション索引は、一意として宣言されている場合でも、変更された行を一意に識別するためには使用できません。しかし、ロジカル・スタンバイ・データベースでは、変更された行を一意に識別できるかぎり、ファンクション索引が定義された表をレプリケーションできます。

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

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

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

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

この問合せの実行には数分間かかる可能性があります。

2. 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。

プライマリ・データベース表で無効化されたRELY制約を作成するには、ALTER TABLE文をRELY DISABLE句と一緒に使用します。次の例では、無効化されたRELY制約がmytabという表に作成されます。各行は、id列とname列を使用して一意に識別されます。

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

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

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

関連項目:

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

4.2 ロジカル・スタンバイ・データベースの作成ステップ

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を次に示します。

表4-1 ロジカル・スタンバイ・データベースの作成

タスク	データベース
フィジカル・スタンバイ・データベースの作成	プライマリ
フィジカル・スタンバイ・データベースでの REDO Apply の停止	スタンバイ

タスク	データベース
ロジカル・スタンバイ・データベースをサポートするためのプライマリ・データベースの準備	プライマリ
ロジカル・スタンバイ・データベースへの推移	スタンバイ
ロジカル・スタンバイ・データベースのオープン	スタンバイ
ロジカル・スタンバイ・データベースが正しく実行されているかどうかの確認	スタンバイ

4.2.1 ロジカル・スタンバイの作成 タスク1: フィジカル・スタンバイ・データベースの作成

ロジカル・スタンバイ・データベースを作成するには、次のように最初にフィジカル・スタンバイ・データベースを作成してから、ロジカル・スタンバイ・データベースへと推移させます。

[「フィジカル・スタンバイ・データベースの作成」](#)の指示に従ってフィジカル・スタンバイ・データベースを作成します。

4.2.2 ロジカル・スタンバイの作成 タスク2: フィジカル・スタンバイ・データベースでの REDO Applyの停止

ロジカル・スタンバイ・データベースに変換する前に、実行時間に関係なく、新規フィジカル・スタンバイ・データベースで REDO Applyを実行できます。

ただし、ロジカル・スタンバイ・データベースに変換する前に、フィジカル・スタンバイ・データベースで REDO Applyを停止する必要があります。REDO Applyの停止は、LogMinerディクショナリを含む REDO の後に変更が適用されることを避けるために必要です([「REDOデータでのディクショナリの構築」](#)を参照)。

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

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

4.2.3 ロジカル・スタンバイの作成 タスク3: ロジカル・スタンバイ・データベースをサポートするためのプライマリ・データベースの準備

ロジカル・スタンバイ・データベースの作成の一部として、ロジカル・スタンバイ・データベースをサポートするためのプライマリ・データベースを準備する必要があります。

次のトピックを参照してください。

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

4.2.3.1 ロールの推移のためのプライマリ・データベースの準備

プライマリ・データベースのロールを切り替える準備をする際に、特定のパラメータを変更する必要がある場合があります。

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

ノート:



このステップは、スイッチオーバーを実行する予定の場合のみ必要です。

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

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

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

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

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

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

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

4.2.3.2 REDOデータでのディクショナリの構築

LogMinerディクショナリは、SQL ApplyのLogMinerコンポーネントによってREDOでの変更が適切に解釈できるように、REDOデータに構築される必要があります。

LogMinerディクショナリの構築の一部として、主キーおよび一意制約/索引列を記録するサプリメンタル・ロギングが自動的に

設定されます。サブリメンタル・ロギング情報により、各更新に、文によって変更された各行を論理的に識別するための十分な情報が含まれていることが保証されます。

LogMinerディクショナリを構築するには、次の文を発行します。

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

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

ノート:

Oracle Database 11g リリース 2 (11.2)以上を使用して作成されたデータベースでは、サブリメンタル・ロギング情報は、既存のフィジカル・スタンバイ・データベースすべてに自動的に伝播されます。ただし、それより前のリリースのデータベースの場合や、それより前のリリースを使用してデータベースを作成してから 11.2 にアップグレードした場合、サブリメンタル・ロギングがプライマリ・データベースで有効化されているときは、それがフィジカル・スタンバイでも有効化されているかどうかを確認する必要があります。それがフィジカル・スタンバイで有効化されていない場合、スイッチオーバーまたはフェイルオーバーを実行する前に、すべての既存のフィジカル・スタンバイ・データベースでサブリメンタル・ロギングを有効化する必要があります。これを実行するには、各フィジカル・スタンバイで次の SQL 文を発行します。

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

これを行わないと、フィジカル・スタンバイ・データベースの 1 つに対してスイッチオーバーまたはフェイルオーバーが行われた場合に、同じ Oracle Data Guard 構成内のすべてのロジカル・スタンバイが使用できません。スイッチオーバーやフェイルオーバーがすでに発生しており、サブリメンタル・ロギングが有効化されていない場合、すべてのロジカル・スタンバイ・データベースを再作成する必要があります。

関連項目:

- [『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)のDBMS_LOGSTDBY.BUILD PL/SQLパッケージ
- [『Oracle Databaseリファレンス』](#)のUNDO_RETENTION初期化パラメータ

4.2.4 ロジカル・スタンバイの作成 タスク4: ロジカル・スタンバイ・データベースへの推移

ロジカル・スタンバイ・データベースに推移できるようにフィジカル・スタンバイ・データベースを準備するには、いくつかのタスクを実行する必要があります。

次の各項で、これらのタスクについて説明します。

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

4.2.4.1 ロジカル・スタンバイ・データベースへの変換

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

ノート:

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

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

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

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

db_name には、新しいロジカル・スタンバイ・データベースを識別するために、プライマリ・データベースとは異なるデータベース名を指定します。この文を発行するときにサーバー・パラメータ・ファイル (spfile) を使用する場合は、新しいロジカル・スタンバイ・データベースの情報でファイルを適切に更新します。spfile を使用していない場合は、データベースの停止後に `DB_NAME` パラメータの名前を設定するよう求めるメッセージが発行されます。

ノート:

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

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

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

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

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

注意:

Oracle Database 11g 以前のリリースでは、ロジカル・スタンバイ・データベースをオープンする前に新しいパスワード・ファイルを作成する必要がありました。現在は、その必要はありません。ロジカル・スタンバイ・データベースに新しい

パスワード・ファイルを作成すると、REDO 転送サービスは正常に機能しません。

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

ノート:



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

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

ロジカル・スタンバイ・データベース上でインスタンスをシャットダウンし、STARTUP MOUNT文を発行してデータベースの起動とマウントを行います。データベースは開かないでください。作成手順の後半まで閉じておき、ユーザー・アクセスは行わないでください。次に例を示します。

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

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

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

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

ノート:



データベースの互換性を 11.1 以降に設定した場合、高速リカバリ領域を使用してリモート・アーカイブ・ログを格納することができます。そのためには、次のパラメータを設定する必要があります(すでに DB_RECOVERY_FILE_DEST および DB_RECOVERY_FILE_DEST_SIZE パラメータが設定されているとします)。

```
LOG_ARCHIVE_DEST_1=  
'LOCATION=USE_DB_RECOVERY_FILE_DEST  
DB_UNIQUE_NAME=boston'
```

高速リカバリ領域を使用しているため、VALID_FOR パラメータを指定する必要はありません。デフォルト値は (ALL_LOGFILES, ALL_ROLES)で、この場合の適切な動作です。LOG_ARCHIVE_DEST_1 が、オンライン(プライマリ)

およびスタンバイの両方の、全ログ・ファイルに使用されます。

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

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

ノート:



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

例4-2 ロジカル・スタンバイ・データベース用の初期化パラメータの変更

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

4.2.5 ロジカル・スタンバイの作成 タスク5: ロジカル・スタンバイ・データベースのオープン

ALTER DATABASE SQL文を使用して、新しく作成したロジカル・スタンバイをオープンします。

たとえば、次の文を発行します(KEEP IDENTITYオプションを使用してロジカル・スタンバイを作成した場合は、RESETLOGSオプションを指定しない)。

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

ノート:



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

警告:

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



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

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

データベースがオープンされるのはこれが初回であるため、データベースのグローバル名は、新しいDB_NAME初期化パラメータと一致するように自動的に調整されます。(KEEP IDENTITYオプションを使用してロジカル・スタンバイを作成した場合、これは当てはまりません。)

ノート:



Oracle Database 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ロジカル・スタンバイ・データベースを作成する場合、SQL Apply ではサポートされないオブジェクトに対する更新に関心があるときは、DBMS_LOGSTDBY PL/SQL プロシージャを使用することをお勧めします。ロジカル・スタンバイ・データベースで、次のプロシージャを実行して情報を取得し、DBA_LOGSTDBY_EVENTS 表のイベントとして記録します。

```
EXEC DBMS_LOGSTDBY.APPLY_SET('MAX_EVENTS_RECORDED',  
DBMS_LOGSTDBY.MAX_EVENTS);
```

```
EXEC DBMS_LOGSTDBY.APPLY_SET('RECORD_UNSUPPORTED_OPERATIONS', 'TRUE');
```

このプロシージャは、ロジカル・スタンバイではサポートされない、プライマリで実行中のトランザクションに関する情報を取得します。アップグレードが完了し、本番を新しいバージョンに切り替える前に、この表を確認します。何も記録されていない場合は、すべてレプリケートされたことになります。何か記録されている場合は、修正処理を実行するか、アップグレードを破棄するかのいずれかを選択できます。

関連項目:

- DBA_LOGSTDBY_EVENTS ビューの詳細は、[「DBA_LOGSTDBY_EVENTS ビューでのイベントのロギングのカスタマイズ」](#)を参照してください
- DBMS_LOGSTDBY パッケージの詳細は、[『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』](#)を参照してください。

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

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

4.2.6 ロジカル・スタンバイの作成 タスク6: ロジカル・スタンバイ・データベースが正しく実行されているかどうかの確認

ロジカル・スタンバイ・データベースを作成した後は、そのデータベースが正しく実行されていることを確認することが重要です。

次のトピックを参照してください。

- [REDO転送サービス](#)
- [ロジカル・スタンバイ・データベースの管理](#)

4.3 ロジカル・スタンバイの作成: 作成後のステップ

ノート:

フィジカル・スタンバイ・データベースからロジカル・スタンバイ・データベースへの変換は、次のように 2 段階で行われます。

1. ALTER DATABASE RECOVER TO LOGICAL STANDBY 文(KEEP IDENTITY 句を指定した場合を除く)の処理の一部として、データベースの DBID が変更されます。
2. ALTER DATABASE START LOGICAL STANDBY APPLY 文が最初に正常に実行されたとき、処理の一部として制御ファイルが更新され、新たに作成されたロジカル・スタンバイ・データベースの制御ファイルと一致するようになります。

ALTER DATABASE START LOGICAL STANDBY APPLY 文の起動に成功した後、ロジカル・スタンバイ・データベースの全体バックアップを作成します。ロジカル・スタンバイ・データベースをリストアする際には、プライマリ・データベースから作成したバックアップを使用できないためです。

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

- データ保護モードのアップグレード

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

- フラッシュバック・データベースの有効化

フラッシュバック・データベースにより、フェイルオーバー後のプライマリ・データベースの再作成の必要がなくなります。フラッシュバック・データベースでは、データファイルをバックアップからリストアしたりREDOデータを広範囲に適用する必要がないため、従来のPoint-in-Timeリカバリよりもはるかに高速でデータベースを過去の最新時点の状態に戻すことができます。フラッシュバック・データベースは、プライマリ・データベースまたはスタンバイ・データベース、あるいはその両方で有効化できます。

関連項目:

- Oracle Data Guard環境でのフラッシュバック・データベースの使用例は、[「フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換」](#)および[「Open Resetlogs文の発行後のフラッシュバック・データベースの使用」](#)を参照してください。
- データベースのフラッシュバックの詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

4.4 CDBのロジカル・スタンバイの作成

通常のプライマリ・データベースのロジカル・スタンバイ・データベースを作成できるように、マルチテナント・コンテナ・データベース(CDB)のロジカル・スタンバイ・データベースを作成できます。

CDBのロジカル・スタンバイの作成および使用時に注意する動作の違いの一部を次に示します。

- データベース・ロールは、プラグブル・データベース(PDB)コンテナ・レベルではなく、CDBレベルで定義されます。
- 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する場合、CDB全体でロールが変化します。
- ロールの変更に関連するDDLは、CDBのルート・コンテナへの接続中に実行される必要があります。
- 通常のロジカル・スタンバイと同様に、CDBのロジカル・スタンバイは、REDOストリームを一度マイニングするプロセスの単一プールを操作しますが、すべてのPDBとCDBのルート・コンテナの更新の職責は共有されます。
- プライマリとスタンバイでPDBのセットが同じである必要はありません。しかし、プライマリとスタンバイの両方で同じコンテナに存在する表のみをレプリケートする必要があります。
- 通常、グローバル構成属性を変更するロジカル・スタンバイPL/SQLインタフェース(DBMS_LOGSTDBY.APPLY_SETなど)はルート・コンテナで実行されます。しかし、DBMS_LOGSTDBY.INSTANTIATE_TABLEは対象の表が存在するコンテナ内で呼び出される必要があり、DBMS_LOGSTDBY.SKIPプロシージャは対象のコンテナ内で呼び出される必要があります。
- ロジカル・スタンバイ・ビューは適切なコンテナ名を指定するように強化されました。多くのDBAビューには、名前がCDBで始まる類似のCDBビューがあります。たとえば、ビューCDB_LOGSTDBY_NOT_UNIQUEには、DBA_LOGSTDBY_NOT_UNIQUEビューに示されるデータと同じデータが含まれますが、PDB名を含む追加の列があります。CDB_LOGSTDBY_NOT_UNIQUEビューがルートで問い合わせられると、CDB内のすべてのデータベースのデータが示されます。
- CDBのロジカル・スタンバイで、SQL文の構文は通常、コンテナでないデータベースと同じです。しかし、次に示すものを

含む、一部の文の影響は異なる場合があります。

- ALTER DATABASE RECOVER TO LOGICAL STANDBYはCDBでのみ機能し、PDBでは使用できません。
- 1つのロールはCDB全体に関連します。個別のPDBには固有のロールはありません。そのため、ロジカル・スタンバイに関連する次のロール変更DDLはCDB全体に影響します。

```
ALTER DATABASE [PREPARE|COMMIT] TO SWITCHOVER
```

```
ALTER DATABASE ACTIVATE LOGICAL STANDBY
```

- ALTER DATABASE [START|STOP] LOGICAL STANDBY APPLYはルート・コンテナでのみ機能し、CDB全体に影響します。この文はPDBでは使用できません。
- ALTER DATABASE GUARDはルート・コンテナでのみ機能し、CDB全体に影響します。たとえば、ALTER DATABASE GUARD ALL文が発行されると、ルートおよびすべてのPDB内のユーザー・アクティビティは制限されます。
- マルチテナント環境を管理するには、CDB_DBAロールが必要です。

関連項目:

- CDBの詳細は、[『Oracle Database概要』](#)を参照してください。
- コンテナでのDBMS_LOGSTDBY. SKIPプロシージャの使用の詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)を参照してください。

5 遠隔同期インスタンスの使用

Oracle Data Guard遠隔同期インスタンスは、プライマリ・データベースからREDOを受け取り、それをOracle Data Guard構成の他のメンバーに送信する、リモートのOracle Data Guardの宛先です。

遠隔同期インスタンスは、制御ファイルを管理し、REDOをスタンバイREDOログ(SRL)に受信し、それらのSRLをローカルのアーカイブREDOログにアーカイブしますが、スタンバイとの類似はそこまでです。遠隔同期インスタンスにはユーザー・データファイルは含まれず、アクセスのために開くことも、REDO Applyを実行することもできず、プライマリ・ロールで機能することも、スタンバイ・データベースの任意のタイプに変換することもできません。

遠隔同期インスタンスはOracle Active Data Guard遠隔同期機能の一部で、Oracle Active Data Guardのライセンスが必要です。

遠隔同期インスタンスは、ディスク・リソースおよび処理リソースをほとんど消費しませんが、データ消失せずにターミナル宛先にフェイルオーバーする機能を提供します。また、プライマリ・データベースの他のタイプのオーバーヘッドをオフロードする機能(REDO転送など)も提供します。

一般的なスタンバイ宛先のサービス中にプライマリで使用可能なすべてのREDO転送オプションは、遠隔同期インスタンスのサービス中にも使用可能です。また、すべてのREDO転送オプションは、ターミナル宛先のサービス中に遠隔同期インスタンスで使用可能です(Oracle Advanced Compressionオプションのライセンスを保有している場合、REDO転送圧縮の実行など)。

多くの構成では、フェイルオーバー時に一部のデータが消失するリスクがあっても非同期転送を使用してスタンバイ・データベースにREDOを送信するプライマリ・データベースがあります。同期REDO転送を使用したデータ消失ゼロの実現は、2つのデータベース間のネットワーク待機時間によるプライマリでのコミット・レスポンス時間への影響のため、実行可能なオプションではない場合があります。

プライマリの近くに遠隔同期インスタンスを作成する利点は、データ保護の保証を強化できると同時に、(プライマリと遠隔同期インスタンス間のネットワーク待機時間が小さくなるため)コミット・レスポンス時間に対する影響を許容しきい値まで最小限に抑えることができることです。プライマリに障害が発生した場合、障害時に遠隔同期インスタンスが同期していたとすると、遠隔同期インスタンスおよびターミナル・スタンバイは、遠隔同期インスタンスからスタンバイへの最終REDO送信を調整してスタンバイでまだ使用できないREDOを送信してから、データ消失ゼロのフェイルオーバーを実行します。

次のトピックを参照してください。

- [遠隔同期インスタンスの作成](#)
- [代替アーカイブ先](#)
- [代替アーカイブ先の構成](#)
- [遠隔同期インスタンスでサポートされる保護モード](#)

5.1 遠隔同期インスタンスの作成

遠隔同期インスタンスの作成は、データファイルが遠隔同期インスタンスには存在しないことを除き、フィジカル・スタンバイの作成と似ています。

そのため、遠隔同期インスタンスでは、データファイルのコピーまたはバックアップからのデータファイルのリストアが不要です。遠隔同期インスタンスを作成したら、構成を変更して、プライマリ・データベースから遠隔同期インスタンスに同期的に最大可用性モードでREDOが送信された後、遠隔同期インスタンスによって非同期でリアルタイムに転送されるようにします。最後に、元の非同期スタンバイ(ターミナル・スタンバイと呼ばれます)を構成して、遠隔同期インスタンスとの通信が割り込まれたときには遠隔同期インスタンスに対する代替として機能するようにします。

ノート:



遠隔同期インスタンスを含む構成では、プライマリ・データベースとリモート・スタンバイ・データベース間にネットワークの直接接続が存在している必要があります。プライマリとリモート・スタンバイ間の直接接続は、ヘルス・チェックおよびスィッチオーバー処理タスクの実行に使用されます。この接続は、遠隔同期インスタンスが失敗し、保護レベルを維持するために構成された代替の遠隔同期が存在しない場合に備えて、スタンバイが代替の宛先として構成されていないかぎり、REDO 転送には使用されません。

5.1.1 遠隔同期インスタンスの作成および構成

遠隔同期インスタンスを作成するには、次のステップを実行します。

1. 次の例に示すように、遠隔同期インスタンス用の制御ファイルを作成します(プライマリ・データベースは開いている必要はありませんが、マウントはされている必要があります)。

```
SQL> ALTER DATABASE CREATE FAR SYNC INSTANCE CONTROLFILE AS -  
> ' /arch2/chicagoFS/control01.ctl' ;
```

作成された制御ファイルにより、chicagoFSはプライマリ・データベースchicagoからREDOを受信する遠隔同期インスタンスとして動作することができます。示されたパスとファイル名は1つの例であり、任意のパスやファイル名を使用できます。

2. プライマリ・データベースで使用されるサーバー・パラメータ・ファイル(SPFIL)からパラメータ・ファイル(PFIL)を作成します。パラメータ・ファイル内の初期化パラメータ設定の大部分は遠隔同期インスタンスにも適していますが、一部変更が必要なものがあります。たとえば、遠隔同期インスタンスでは、DB_FILE_NAME_CONVERTおよびLOG_FILE_NAME_CONVERTパラメータを設定し、遠隔同期インスタンスのDB_UNIQUE_NAMEおよび遠隔同期インスタンスの制御ファイルの場所を変更する必要があります。[例5-1](#)に、chicagoFSのDB_UNIQUE_NAMEを使用した、遠隔同期インスタンスのパラメータ・ファイル・コンテンツの例を示します。
3. パラメータ値の次の変更を容易にするには、編集されたパラメータ・ファイル(PFIL)からサーバー・パラメータ・ファイル(SPFIL)を作成します。SPFILを使用しないと、遠隔同期インスタンスがOracle Data Guard Broker構成に追加されたときに、SHOW CONFIGURATION出力で警告が返されます。
4. 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、ステップ1で作成した遠隔同期インスタンス制御ファイルと、ステップ3で作成したサーバー・パラメータ・ファイル(SPFIL)を、プライマリ・システムから遠隔同期インスタンスシステムの適切な場所にコピーします。
5. 通常のスタンバイ用に作成したのと同じ方法で、スタンバイREDOログを作成します。[「スタンバイREDOログの管理」](#)を参照してください。

LOG_FILE_NAME_CONVERTパラメータが遠隔同期インスタンスに指定されているため([例5-1](#)を参照)、[REDOデータを受信するためのプライマリ・データベースの構成](#)で説明しているようにREDOログがプライマリで作成された場合、REDO転送がプライマリから開始するときにスタンバイREDOログが自動的に作成されます。

ノート:



遠隔同期インスタンスで使用するスタンバイ REDO ログ・ファイルは他のデータベースと共有できません。そのため、スタンバイ REDO ログ・ファイルに関して[「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」](#)で検討したすべての関連する考慮事項は、遠隔同期インスタンスにも適用されます。

6. 遠隔同期インスタンスがWindowsシステムでホストされる場合、ORADIMユーティリティを使用してWindowsサービスを作成します。次に例を示します。

```
oradim -NEW -SID ChicagoFS -STARTMODE manual
```

遠隔同期インスタンスが作成されて実行されたら、STARTMODEをautoに変更し、遠隔同期インスタンスの自動起動を有効にすることができます。

ORADIMユーティリティは、このサービスを作成する必要があるユーザー名を自動的に決定し、そのユーザー名のパスワードを求めます(そのユーザー名がパスワードを必要とする場合)。ORADIMユーティリティの使用の詳細は、[『Oracle Databaseプラットフォーム・ガイドfor Microsoft Windows』](#)を参照してください。

7. このステップは、オペレーティング・システムの認証を管理ユーザーに使用する場合およびSSLをREDO転送の認証に使用する場合はオプションです。いずれの場合でもない場合は、プライマリ・データベースのリモート・ログイン・パスワード・ファイルを遠隔同期インスタンスの適切なディレクトリにコピーします。パスワード・ファイルは、管理権限(SYSDB、SYSOPER、SYSDBAなど)を付与または解除されるたびに、あるいは管理権限を持つユーザーのパスワード変更後、コピーし直す必要があります。

Oracle Database 12cリリース2 (12.2.0.1)以降、遠隔同期インスタンスでパスワード・ファイルが手動で更新されると、プライマリ・データベースからの同じパスワード変更内容を含むREDOが、遠隔同期インスタンスからREDOを受け取るように設定されているすべてのスタンバイ・データベースに自動的に伝播されます。REDOが適用されると、スタンバイでパスワード・ファイルが更新されます。

8. 遠隔同期インスタンス・サイトで、Oracle Net Managerを使用して遠隔同期インスタンスのリスナーを構成します。

リスナーの詳細は、[『Oracle Database Net Services管理者ガイド』](#)を参照してください。

9. プライマリ・システムで、Oracle Net Managerを使用して、REDO転送サービスで使用する遠隔同期インスタンス(chicagoFS)用のネットワーク・サービス名を作成します。

遠隔同期インスタンス・システムで、Oracle Net Managerを使用して、REDO転送サービスで使用するプライマリ(chicago)およびターミナル・スタンバイ(boston)用のネットワーク・サービス名を作成します。

Oracle Netネット・サービス名は、プライマリ・データベース、遠隔同期インスタンス、およびターミナル・スタンバイ・データベースに対するリスナーの構成時に指定したものと同一プロトコル、ホスト・アドレス、ポートおよびサービスを使用する接続記述子に解決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。

サービス名の詳細は、[『Oracle Database Net Services管理者ガイド』](#)を参照してください。

10. マウント・モードで遠隔同期インスタンスを開始します。
11. 遠隔同期インスタンスが適切に動作していることを確認します。

遠隔同期インスタンスの作成後の構成の検証の詳細は、[「構成の検証」](#)を参照してください。

12. 構成の保護モードを最大可用性に引き上げます。プライマリ・データベースで、次のコマンドを実行します。

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

関連項目:

- 遠隔同期および保護モードの詳細は、[「遠隔同期インスタンスでサポートされる保護モード」](#)を参照してください。

- 各種データ保護モードの構成の詳細は、[「Oracle Data Guardの保護モード」](#)を参照してください

例5-1 遠隔同期インスタンスに使用される初期化パラメータの一部

プライマリ・データベースchicago

```
DB_UNIQUE_NAME=chicago

CONTROL_FILES=' /arch1/chicago/control01.ctl'

DB_FILE_NAME_CONVERT=' /boston/' , ' /chicago/'

LOG_FILE_NAME_CONVERT=' /boston/' , ' /chicago/'

FAL_SERVER=boston

LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, boston)'

LOG_ARCHIVE_DEST_1=' LOCATION=USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(ALL_LOGFILES, ALL_ROLES) DB_UNIQUE_NAME=chicago'

LOG_ARCHIVE_DEST_2=' SERVICE=chicagoFS SYNC AFFIRM
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS'
```

遠隔同期インスタンスchicagoFS

```
DB_UNIQUE_NAME=chicagoFS

CONTROL_FILES=' /arch2/chicagoFS/control01.ctl'

DB_FILE_NAME_CONVERT=' /chicago/' , ' /chicagoFS/' , ' /boston/' , ' /chicagoFS/'

LOG_FILE_NAME_CONVERT=' /chicago/' , ' /chicagoFS/' , ' /boston/' , ' /chicagoFS/'

FAL_SERVER=chicago

LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, boston)'

LOG_ARCHIVE_DEST_1=' LOCATION= USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(ALL_LOGFILES, ALL_ROLES) DB_UNIQUE_NAME=chicagoFS'

LOG_ARCHIVE_DEST_2=' SERVICE=boston ASYNC
VALID_FOR=(STANDBY_LOGFILES, STANDBY_ROLE) DB_UNIQUE_NAME=boston'
```

フィジカル・スタンバイboston

```
DB_UNIQUE_NAME=boston

CONTROL_FILES=' /arch3/boston/control01.ctl'

DB_FILE_NAME_CONVERT=' /chicago/' , ' /boston/'

LOG_FILE_NAME_CONVERT=' /chicago/' , ' /boston/'

FAL_SERVER=' chicagoFS' , ' chicago'

LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, boston)'

LOG_ARCHIVE_DEST_1=' LOCATION= USE_DB_RECOVERY_FILE_DEST'
```

```
VALID_FOR=(ALL_LOGFILES, ALL_ROLES) DB_UNIQUE_NAME=boston'  
  
LOG_ARCHIVE_DEST_2=' SERVICE=chicago ASYNC  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicago'
```

5.2 代替アーカイブ先

[遠隔同期インスタンスの作成および構成](#)のステップを実行すると、その後、遠隔同期インスタンスには、WAN経由のリモート・サイトにあるターミナル・スタンバイにREDOを転送することで構成にデータ消失ゼロ機能が備わります。遠隔同期インスタンスがアクセスできない場合でも構成が保護されたままにするには、スタンバイ・データベースへの代替REDO転送パスを構成する必要があります。これは、LOG_ARCHIVE_DEST_nパラメータのGROUPおよびPRIORITY属性を使用して達成できます。(Oracle Database 12cリリース2 (12.2.0.1)以降、GROUPおよびPRIORITY属性で遠隔REDO宛先のALTERNATE属性が置き換えられています。)

可能な代替遠隔宛先の数は、ログ・アーカイブ先グループの概念により増加しました。ログ・アーカイブ先グループで複数のアーカイブ先を指定し、遠隔同期インスタンスから、またはカスケードにより、複数の宛先にREDOを配布するために使用できます。グループ内の宛先は、プライマリ・データベース上で一度に1つの宛先のみがアクティブになるように優先順位付けできます。その他の宛先は、アクティブな宛先が使用不可になると使用可能でアクティブになります。使用しているデータベースの可能なアーカイブ先の数を拡張するには、複数のグループを指定できます。

関連項目:

- 古いALTERNATE構文を使用した代替リモート宛先の構成については、[ALTERNATE属性を使用したリモート代替宛先の構成](#)を参照してください。

5.2.1 グループへのログ・アーカイブ先の割当て

LOG_ARCHIVE_DEST_n初期化パラメータのGROUP属性を使用して、ログ・アーカイブ先をグループに割り当てます。

ログ・アーカイブ先グループが使用されている場合、グループ内に少なくとも1つの宛先が使用できるかぎり、引き続き少なくとも1つの宛先が有効でアクティブになります。グループに割り当てられていないログ・アーカイブ先は、Oracle Database 12cリリース2 (12.2.0.1)より前のログ・アーカイブ先と同じように動作します。

1つのグループに最大30のログ・アーカイブ先が可能です。ログ・アーカイブ先グループは、グループの作成時に割り当てられるグループ番号で参照されます。グループは1から8まで番号付けされます。ログ・アーカイブ先グループには遠隔の(SERVICE=...)宛先のセットが含まれます。(ローカル・アーカイブ(LOCATION=...)の宛先はログ・アーカイブ先グループではサポートされず、代替ローカル・アーカイブ場所にはALTERNATE属性を使用する必要があります。[ALTERNATE](#)を参照してください。

グループ内の1つのログ・アーカイブ先が常にアクティブで、その他はアクティブなログ・アーカイブ先の障害の場合に使用可能です。障害があった宛先が再度使用可能になった場合、現在アクティブな宛先に障害が発生した場合の対象になりますが、その他のグループ・メンバーもすべて使用不可にならないかぎりすぐにアクティブにはなりません。たとえば、次の宣言を使用して3つの遠隔同期インスタンスを同じグループのメンバーで同じ優先順位を持つと指定できます(グループ内の優先順位は次の項で説明します)。これらはサンプル・パラメータ定義で、必要な属性のすべては含んでいません。Verbatimで使わないでください。この例では、最初の宛先のみがアクティブで、2番目の宛先は宛先1が使用不可になった場合の引継ぎに使用できます。

```
LOG_ARCHIVE_DEST_2=' SERVICE=chicagoFS SYNC  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1'  
  
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

```
LOG_ARCHIVE_DEST_3='SERVICE=chicagoFS1 SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

ノート:



ログ・アーカイブ先グループは LOG_ARCHIVE_DEST_n ALTERNATE 属性を置き換えるため、デフォルト・グループ (GROUP が 1 から 8 として指定) にはログ・アーカイブ先での ALTERNATE 属性の使用は許可されていません。

5.2.2 グループ内のログ・アーカイブ先に対する優先順位の割当て

LOG_ARCHIVE_DEST_n 初期化パラメータの PRIORITY 属性を使用してログ・アーカイブ先グループ内に宛先の優先順位を割り当てると、フェイルバック・メカニズムを、特に1つのグループ内に複数のメンバーがある場合に制御できます。また、PRIORITY 属性を使用して、優先宛先に障害があった場合に複数の宛先に送信するようにグループを構成することもできます。

前出の項では、2つの遠隔同期インスタンスが優先順位を持っていませんでした。これは、最初の宛先の障害後に代替宛先がアクティブ化されると、障害が発生するまではアクティブな宛先のままになることを意味します。優先順位は、データベースまたは遠隔同期インスタンスが起動したとき、または宛先に障害が発生したときにグループ内のどのログ・アーカイブ先をアクティブにするかを決定するために使用されます。ログ・アーカイブ先は、プライマリ・データベースが読取り/書込みモードでオープンされるか、遠隔同期インスタンスがマウントされるか、スタンバイ・データベースがマウントされるか読取り専用でオープンされる場合にアクティブになります。同じ優先順位値を2つ以上のログ・アーカイブ先に割り当てることができます。優先順位値は1から8の範囲の整数です。数値が低いほど優先度が高くなります。デフォルトの優先順位は1 (最も高い優先順位) です。

優先順位は、以前に障害が発生したアーカイブ先が再び使用可能になったときに機能します。同じグループに割り当てられたログ・アーカイブ先のセットは、デフォルトで同じ優先順位を持ちます。したがって、1つの宛先に障害があるともう一方のセット・メンバーに対してフェイルオーバーが発生します。障害が発生した宛先が再度使用可能になった場合、両方の宛先が同じ優先順位を持つためアクティブになりません。最初のアーカイブ先が再度使用可能になった後で2番目のアーカイブ先に障害が発生した場合、データベースは最初のアーカイブ先またはグループ内の同じ優先順位を持つ別のアーカイブ先にフェイルオーバーします。このサイクルは、アクティブな宛先に障害が発生する前に一方の宛先が常に使用可能であるという条件で無限に繰り返す可能性があります。

以前のサンプルを続行すると、優先順位がログ・アーカイブ先に追加され、宛先がいつアクティブになるかを制御できます。次の例では、3つ目の遠隔同期インスタンスが追加されますが、優先順位は低いです。

```
LOG_ARCHIVE_DEST_2='SERVICE=chicagoFS SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_2=ENABLE

LOG_ARCHIVE_DEST_3='SERVICE=chicagoFS1 SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE

LOG_ARCHIVE_DEST_4='SERVICE=chicagoFS2 ASYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=2'

LOG_ARCHIVE_DEST_STATE_4=ALTERNATE
```


この宣言結果の動作は、次のようになります。

- プライマリは2つの優先遠隔同期インスタンスの1つ目chicagoFSにREDOを送信します。
- chicagoFSが使用不可になると、プライマリがchicagoFS1に送信します。
- chicagoFSが再度使用可能になると、フェイルオーバーは発生しません。優先順位が同じためchicagoFS1に対する代替になります。
- chicagoFSとchicagoFS1の両方が使用不可になった場合、プライマリはchicagoFS2に送信します(この場合はASYNC REDO転送モード経由)。
- プライマリがchicagoFS2に対して送信中にchicagoFSまたはchicagoFS1のいずれかが使用可能になった場合、プライマリはその使用可能な優先ログ・アーカイブ先にフェイルバックします。

5.2.3 グループ内の複数のアクティブな宛先への送信

また、PRIORITY属性を使用して、優先宛先に障害が発生した場合にグループを複数の宛先に送信するように構成することもできます。

単一グループ内で複数のアクティブな宛先をサポートするメカニズムでは、そのグループ内で該当する優先順位を持つアクティブな宛先に対して最も低い優先順位(PRIORITY=8)が定義され、一般にターゲット・スタンバイ・データベースにREDOを直接に送信するために使用されます。次のログ・アーカイブ先宣言は、これをどのように構成できるかを示しています。この例では、2つのターミナル・スタンバイ・データベースにREDOを転送する遠隔同期インスタンスが1つあります。

```
LOG_ARCHIVE_DEST_2=' SERVICE=chicagoFS SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_2=ENABLE

LOG_ARCHIVE_DEST_3=' SERVICE=boston ASYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=8'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE

LOG_ARCHIVE_DEST_4=' SERVICE=newyork ASYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=8'

LOG_ARCHIVE_DEST_STATE_4=ALTERNATE
```

この宣言結果の動作は、次のようになります。

- プライマリは優先遠隔同期インスタンスchicagoFSにREDOを送信します。
- chicagoFSが使用不可の場合、プライマリはbostonとnewyorkの両方のターミナル・スタンバイにASYNCモードで直接送信します。
- bostonとnewyorkへの送信中にchicagoFSが使用可能になった場合、プライマリはbostonとnewyorkへの直接送信を停止して、かわりにchicagoFSへの送信を開始します。

5.2.4 複数のログ・アーカイブ先グループの使用

複数のログ・アーカイブ先グループは、サイト固有の高可用性の検討または大規模なカスケード(リーダー・ファーム)構成へのサービスの配布に使用できます。

次の宣言は、chicagoFSとchicagoFS1をグループ1、chicagoFS3とchicagoFS4をグループ2に含む複数のログ・アーカイブ先グループを設定します。


```
LOG_ARCHIVE_DEST_2=' SERVICE=chicagoFS SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_2=ENABLE

LOG_ARCHIVE_DEST_3=' SERVICE=chicagoFS1 SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=1 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE

LOG_ARCHIVE_DEST_4=' SERVICE=chicagoFS3 SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=2 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_4=ENABLE

LOG_ARCHIVE_DEST_5=' SERVICE=chicagoFS4 SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) GROUP=2 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_5=ALTERNATE
```

5.2.5 ログ・アーカイブ先の可用性ステータスの判断

Oracle Data Guardでは、ログ・アーカイブ先グループ内で使用可能であるが非アクティブな宛先の現在のステータスを、定期的に構成済宛先をポーリングすることにより追跡してその可用性を判断します。

可用性を判断するために使用される情報は、プライマリ・データベースが障害がある宛先を放棄する前に、REDO転送サービスが通信を再確立してその宛先へのREDOデータを転送する連続試行回数を指定するMAX_FAILURE属性から導出されます。MAX_FAILUREのデフォルト値は、GROUPおよびPRIORITY属性が使用されている場合は1です。

MAX_FAILURE属性の動作は、Oracle Database 12cリリース1 (12.1)とOracle Database 12cリリース2 (12.2)の間で異なります。この違いを理解しておくことが重要です。

関連項目:

- [MAX_FAILURE](#)

5.3 代替アーカイブ先の構成

遠隔同期インスタンス構成は、様々なレベルのデータ保護を提供するように設定できます。

次のトピックでは、前の項で示した例を拡張して、遠隔同期インスタンスを使用する場合にデータ保護を強化する追加の遠隔同期インスタンス構成2つの例を示します。

- [遠隔同期障害後の保護の低下](#)
- [遠隔同期インスタンスの高可用性](#)
- [ロール変更後の保護の維持](#)

5.3.1 遠隔同期障害後の保護の低下

すべての遠隔同期インスタンス構成で、REDOがターミナル・スタンバイへの送信を継続し、プライマリ・データベースを引き続き保護することが重要です。

最も単純な構成では、遠隔同期通信が1つ(chicagoFS)とターミナル・スタンバイ・データベースが1つ(boston)あります。

遠隔同期インスタンスに障害が発生した場合、REDOはプライマリ・データベースchicagoにログ・アーカイブ先を追加することでターミナル・スタンバイに直接送信する必要があります。これにより、REDO転送がSYNCモードのかわりにASYNCモードになるため、保護レベルが低下します。

例5-2 単一宛先フェイルオーバーに対する構成

プライマリ・データベースchicago

```
LOG_ARCHIVE_DEST_2='SERVICE=chicagoFS SYNC AFFIRM
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS GROUP=1 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_2=ENABLE

LOG_ARCHIVE_DEST_3='SERVICE=boston ASYNC NOAFFIRM
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=boston GROUP=1 PRIORITY=2'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

この宣言により、遠隔同期インスタンスchicagoFSに障害が発生した場合にプライマリ・データベースがターミナル・スタンバイに直接REDOを送信します。遠隔同期インスタンスが再度使用可能になった場合、アクティブな宛先となりREDO転送が遠隔同期インスタンスに送られます。

遠隔同期インスタンスに複数のターミナル・スタンバイがある場合、PRIORITY=8を使用して、遠隔同期インスタンスに障害が発生した場合にそれらの宛先のすべてがプライマリ・データベースから直接REDOを受け取るようにします。

例5-3 複数スタンバイ・データベースのREDO宛先フェイルオーバーに対する構成

プライマリ・データベースchicago

以前の例にあるように、遠隔同期インスタンスに対するプライマリ・データベースのログ・アーカイブ先を変更して優先順位1のグループに追加した後、遠隔同期インスタンスが優先順位8、ASYNCモードでサポートする各スタンバイに対する新しいログ・アーカイブ先を追加します。

```
LOG_ARCHIVE_DEST_2='SERVICE=chicagoFS SYNC AFFIRM
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS GROUP=1 PRIORITY=1'

LOG_ARCHIVE_DEST_STATE_2=ENABLE

LOG_ARCHIVE_DEST_3='SERVICE=boston ASYNC NOAFFIRM
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=boston GROUP=1 PRIORITY=8'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE

LOG_ARCHIVE_DEST_4='SERVICE=newyork ASYNC NOAFFIRM
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=newyork GROUP=1 PRIORITY=8'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

この宣言により、遠隔同期インスタンスchicagoFSに障害が発生した場合にプライマリ・データベースが両方のターミナル・スタンバイ・データベースに直接REDOを送信します。遠隔同期インスタンスが再度使用可能になった場合、アクティブな宛先となりREDO転送が遠隔同期インスタンスに送られます。

5.3.2 遠隔同期インスタンスの高可用性

代替遠隔同期インスタンスを構成することで、優先遠隔同期インスタンスがなんらかの理由で失敗した場合に、構成の保護レ

ベルが最大可用性の構成済保護レベルに維持されます。

先行するどちらの例でも、REDOがSYNCモードで送信されなくなるため保護レベルが「最大可用性」の範囲を外れる場合があります。システム障害またはネットワーク障害からの保護を強固にするため、アクティブな遠隔同期インスタンスが高い可用性を保持するように追加の遠隔同期インスタンスを構成できます。この構成では、一方は優先アクティブ遠隔同期インスタンスであり、他方は代替遠隔同期インスタンスです。

プライマリは遠隔同期インスタンスで障害が検知されると、自動的に代替遠隔同期インスタンスへの送信を開始します。このタイプの構成では、プライマリはいつでも1つの遠隔同期インスタンスのみを使用してREDOを再配信します。

最大可用性の保護レベルを維持するには、プライマリ・データベースに近い2つの遠隔同期インスタンスを構成してお互いを保護するように設定します。その結果、アクティブな遠隔同期インスタンスが使用できなくなった場合、プライマリ・データベースは代替遠隔同期インスタンスにREDOを同期モードで送信することを自動的に開始できるため、最大可用性の高保護レベルを維持します。ただし、この場合、2つの遠隔同期インスタンスが同じ優先順位であり、一方が他方を引き継ぐと障害が発生するまではアクティブな遠隔同期インスタンスのままになります。両方の遠隔同期インスタンスに障害が発生してもREDOが引き続きスタンバイ・データベースに送信されるようにするために、ターミナル・スタンバイ・データベースは以前のようにPRIORITY=2で構成されます。(2つ以上のターミナル・スタンバイ・データベースがある場合はPRIORITY=8を使用します)。

高可用性遠隔同期インスタンスは、[遠隔同期インスタンスの作成および構成](#)に示されているのと同じステップを使用して作成され、REDOをターミナル・スタンバイbostonに転送するように構成されます。

例5-4 高可用性遠隔同期インスタンスの設定に使用されるパラメータ

プライマリ・データベースchicago

```
LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, chicagoFS1, boston)'  
  
LOG_ARCHIVE_DEST_2=' SERVICE=chicagoFS SYNC AFFIRM  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS GROUP=1 PRIORITY=1'  
  
LOG_ARCHIVE_DEST_STATE_2=ENABLE  
  
LOG_ARCHIVE_DEST_3=' SERVICE=chicagoFS1 SYNC AFFIRM  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS1 GROUP=1 PRIORITY=1'  
  
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE  
  
LOG_ARCHIVE_DEST_4=' SERVICE=boston ASYNC NOAFFIRM  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=boston GROUP=1 PRIORITY=2'  
  
LOG_ARCHIVE_DEST_STATE_4=ALTERNATE
```

Oracle Data Guardは、遠隔同期インスタンスにREDOを同期的に送信し続けて、遠隔同期インスタンスに障害が発生しても必要なデータ消失ゼロの最大可用性保護モードを維持できるようになっています。両方の遠隔同期インスタンスに障害が発生した場合、REDOは、ASYNCモードで、低下した保護レベルでbostonに送信されます。前述のように、障害が発生した遠隔同期インスタンスのいずれかが再度使用できるようになると、Oracle Data Guardは自動的に再同期し、プライマリがアクティブな遠隔同期インスタンスにREDOを送信した後、その遠隔同期インスタンスがターミナル・スタンバイにそのREDOを転送する元の構成に戻ります。同期が完了すると、スタンバイ(前の例ではLOG_ARCHIVE_DEST_4)の代替宛先は再度代替として休止状態になります。

5.3.3 ロール変更後の保護の維持

これらの例では、ロールの変更後もデータ保護を維持する方法について説明します。

以前の項に記載された構成は良好に機能し、すべての遠隔同期インスタンスに障害が発生してREDOがスタンバイ・データ

ベースに直接送信されるまでは最大可用性で構成の稼働を維持します。しかし、bostonがプライマリ・データベースになりchicagoがターミナル・スタンバイになるというロールの推移の後は適切に動作しないことが考えられます。遠隔同期インスタンスchicagoFSとchicagoFS1は、2つのサイト間のネットワーク待機時間がコミット・レスポンス時間に影響を及ぼすほど大きいいため、bostonが同期宛先として使用するには遠すぎると考えられます。データ消失ゼロのために最大可用性の保護レベルを維持するには、将来のロールの推移イベントに備えて、2つ目の遠隔同期インスタンス構成をbostonの近くに確立する必要があります。

[遠隔同期インスタンスの作成および構成](#)で説明しているのと同じ手順を使用して、bostonFSおよびbostonFS1という2つの遠隔同期インスタンスをスタンバイ・データベースbostonの近くに作成し、アクティブなときにREDOをASYNCモードでchicagoに送信するようにその両方を構成します。その後、bostonがプライマリのときにREDOを遠隔同期インスタンスの1つにchicagoおよびその遠隔同期インスタンスに対して構成されたすべてのフェイルオーバー機能とともにSYNCモードで送信できるように、それらをbostonに追加します。新しいboston遠隔同期インスタンスをbostonとchicagoの両方のLOG_ARCHIVE_CONFIGに追加する必要があります。

例5-5 ロール変更後の保護設定に使用されるパラメータ

プライマリ・データベースboston

```
LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, chicagoFS1, boston, bostonFS, bostonFS1)'  
  
LOG_ARCHIVE_DEST_2=' SERVICE=bostonFS SYNC AFFIRM  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=bostonFS GROUP=1 PRIORITY=1'  
  
LOG_ARCHIVE_DEST_STATE_2=ENABLE  
  
LOG_ARCHIVE_DEST_3=' SERVICE=bostonFS1 SYNC AFFIRM  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=bostonFS1 GROUP=1 PRIORITY=1'  
  
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE  
  
LOG_ARCHIVE_DEST_4=' SERVICE=chicago ASYNC NOAFFIRM  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicago GROUP=1 PRIORITY=2'  
  
LOG_ARCHIVE_DEST_STATE_4=ALTERNATE
```

プライマリ・データベースchicago

```
LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, chicagoFS1, boston, bostonFS, bostonFS1)'
```

これらの宣言が指定されると、遠隔同期インスタンスbostonFSはbostonからREDOを受け取り、bostonがプライマリ・データベースである場合のみchicagoにそれを送信します。ただし、bostonがプライマリ・データベースでない場合でも、将来のロールの推移に備えて、遠隔同期インスタンスbostonFSとbostonFS1をマウントしたままにすることをお勧めします。

5.4 遠隔同期インスタンスでサポートされる保護モード

遠隔同期インスタンスは、最大パフォーマンス・モードまたは最大可用性モードのいずれかでサポートされます。

5.4.1 最大可用性モード構成での遠隔同期インスタンス

最大可用性モードでは、遠隔同期インスタンスはネットワーク待機時間を最小化するために比較的プライマリ・データベースに近く、プライマリはSYNC転送を使用して遠隔同期インスタンスにサービスを提供します。



ノート:

最大可用性モードでのプライマリと遠隔同期インスタンスの間の距離に、アーキテクチャ上の制限はありません。実際の距離の制限は、同期構成でのネットワーク待機時間の影響に対する、特定のアプリケーションの許容度範囲により異なります。また、新しい Oracle Data Guard FastSync 機能(SYNC/NOAFFIRM)を使用して、どのような距離に対してもパフォーマンスの影響を低減することが可能です。「[最大可用性モードでのパフォーマンス対保護](#)」を参照してください。

SYNC/AFFIRMセマンティクスとSYNC/NOAFFIRMセマンティクスはいずれも、遠隔同期インスタンス用にプライマリで確立される LOG_ARCHIVE_DEST_*n* でサポートされています。それぞれの使用のトレードオフについては、「[Oracle Data Guardの保護モード](#)」を参照してください。

プライマリがSYNC転送を使用した遠隔同期インスタンスをサービスするとき、すべてのコミットされたREDOは遠隔同期インスタンスのディスクに存在します。これにより、プライマリ・データベースが失われると、遠隔同期インスタンスはデータの消失のないフェイルオーバーのために、ターミナル・スタンバイ宛先の1つを使用できます。

遠隔同期インスタンスはASYNCTransferを使用して、より遠くにあるターミナル・スタンバイに受信REDOを送信します。これにより、トランザクション・スループットの低下が生じるために、プライマリ・データベースにとってSYNC転送を使用して直接サービスするには遠すぎる宛先へ、データの消失のない保護が拡大されます。これは、構成内に1つのスタンバイ宛先しか存在しない場合でも、遠隔同期インスタンスが役に立つ例です。

5.4.2 最大パフォーマンス・モード構成での遠隔同期インスタンス

最大パフォーマンス・モードでは、プライマリ・データベースは、ASYNCTransferを使用して遠隔同期インスタンスの宛先にサービスを提供します。

宛先がASYNCTransferでサービス提供されている場合は、ネットワーク待機時間が長くてもトランザクション・スループットに影響はないため、これには、プライマリと遠隔同期インスタンスとの間の物理的距離は関係ありません。

最大パフォーマンス・モードでは、遠隔同期インスタンスは複数のリモート宛先を管理するOracle Data Guard構成に役立ちます。各ASYNCTransfer宛先のプライマリ・データベースのパフォーマンスに与える影響はほとんどゼロですが、多くのリモート宛先が存在する場合は(リーダー・ファームを形成する複数のOracle Active Data Guardスタンバイなど)、その影響が少なからず発生することがあります。遠隔同期インスタンスが使用される場合、リモート宛先が構成に追加されることで影響が増加することはありません。さらに、REDO転送圧縮も遠隔同期インスタンスへオフロードすることができます。遠隔同期インスタンスを使用する場合、プライマリは遠隔同期インスタンスにサービスを提供するだけで、その遠隔同期インスタンスが残りの構成にサービスを提供します。宛先の数が増えるほど、パフォーマンスは大きく向上します。

6 Oracle Data Guardの保護モード

Oracle Data Guardでは、様々なデータ保護モードを設定できます。

これらの保護モードと、プライマリ・データベースでこれを設定する方法については、次の各項を参照してください。

- [Oracle Data Guardの保護モード](#)
- [プライマリ・データベースのデータ保護モードの設定](#)

6.1 Oracle Data Guardの保護モード

Oracle Data Guardには、最大可用性、最大パフォーマンスおよび最大保護という3つの保護モードが用意されています。

保護モードについての次の説明では、同期されたスタンバイ・データベースは、構成されたデータ保護モードの最小要件を満たしている、REDOギャップがないものである必要があります。REDOギャップについては、[「REDOギャップの検出および解決」](#)で説明します。

最大可用性

この保護モードは、プライマリ・データベースの可用性を低下させない範囲で可能な最高レベルのデータ保護を提供します。通常の操作では、トランザクションは、そのトランザクションのリカバリに必要なすべてのREDOデータがオンラインREDOログに書き込まれるまで、および(構成により)次のいずれかが当てはまるまでコミットされません。

- REDOがスタンバイで受信され、スタンバイREDOログへのI/Oが起動され、確認がプライマリに返された
- REDOが受信され、スタンバイでスタンバイREDOログへ書き込まれ、応答がプライマリに返された

1つ以上の同期スタンバイから確認を受信できない場合、プライマリは最大パフォーマンス・モードの場合と同様に動作して、同期スタンバイ・データベースへのREDOストリームの書き込みが再び可能になるまでプライマリ・データベースの可用性を維持します。

プライマリ・データベースに障害が発生した場合、このモードは、Oracle Data Guard構成で1つ以上の同期スタンバイが存在する場合に、データが消失しないことを保証します。最大可用性のサポートに必要なREDO転送設定および関連するトレードオフの詳細は、[「Pe最大可用性モードでのパフォーマンス対保護」](#)を参照してください。

Oracle Data GuardがREDOデータをスタンバイREDOログ・ファイルの永久記憶域に書き込むとすぐに、プライマリのトランザクションは保護されたものとみなされます。その後すぐにプライマリ・データベースに確認通知を戻し、次のトランザクションに進めるようにします。これによって、プライマリ・データベースでの同期転送スループットやレスポンス時間による影響を最小化できます。スタンバイ・データベースでの完全なOracle Data Guardの検証機能を最大限活用するには、必ずリアルタイムの適用モードで操作し、REDOの変更を受信と同時にスタンバイ・データベースに適用します。Oracle Data Guardは検出したすべての破損を通知するので、迅速な修正処理を実行できます。

最大可用性モードでのパフォーマンス対保護

最大可用性モードを使用するとき、ニーズに最も適した選択をすることができるよう、LOG_ARCHIVE_DEST_nの属性SYNC/AFFIRM対SYNC/NOAFFIRM (FastSync)を使用したことによってもたらされ得る結果を理解することが重要です。

SYNC/AFFIRMを使用して転送が実行されると、プライマリは書き込み操作を実行し、REDOがフィジカル・スタンバイに同期的に転送されてディスクに書き込まれたという応答を待ちます。SYNC/AFFIRMを使用した転送では、スタンバイREDOログへのI/Oの完了に必要な時間によるパフォーマンスへの影響がありますが、保護上のメリットが追加されます。

SYNC/NOAFFIRMを使用して転送が実行されると、プライマリは書き込み操作を実行し、データがスタンバイで受信されたという応答のみを待ち、ディスクに書き込まれたという応答は待ちません。SYNC/NOAFFIRMを使用した転送では、複数の障害が同時に発生する特別な場合でのデータ消失の危険性を伴いますが、パフォーマンス上のメリットがもたらされます。

これらの説明を考慮して、スタンバイ・サイトの電源喪失と同時にプライマリ・サイトで致命的な障害が発生することを想定してください。データが消失するかどうかは使用している転送モードにより決まります。SYNC/AFFIRMの場合、データがスタンバイのディスクに書き込まれたことがチェックされ、システムが修復されたときにデータはスタンバイで使用可能なため、データ消失はありません。SYNC/NOAFFIRMの場合、データがスタンバイのディスクに書き込まれたことは確認されず、データ消失が発生する可能性があります。

関連項目:

- SYNC、AFFIRMおよびNOAFFIRM属性の詳細は、[「LOG_ARCHIVE_DEST_nパラメータの属性」](#)を参照してください。
- FASTSYNCモードを使用した(可用性最大モードでSYNCおよびNOAFFIRMと一緒に使用)ブローカ構成でのREDO転送の詳細は、[『Oracle Data Guard Broker』](#)を参照してください。

最大パフォーマンス

この保護モードは、プライマリ・データベースのパフォーマンスに影響しない範囲で可能な最高レベルのデータ保護を提供します。これは、トランザクションによって生成されたすべてのREDOデータがオンライン・ログに書き込まれた直後に、そのトランザクションのコミットを可能にすることで実現されます。REDOデータは1つ以上のスタンバイ・データベースにも書き込まれますが、この書き込みはトランザクションのコミットとは非同期に実行されるため、REDOデータの送信に必要な時間およびスタンバイ・データベースからの応答の受信は、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ません。

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

これはデフォルトの保護モードです。

最大保護

最大保護は最大可用性と似ていますが、複数の失敗イベントの際により高いレベルでデータが保護されます。スタンバイ・データベースからの応答を受信できない場合でもプライマリが処理を続行できる最大可用性とは違い、最大保護では、プライマリ・データベースは停止し、保護されないトランザクションの処理は継続できません。

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

ノート:

非同期でコミットされたトランザクションは、そのトランザクションで生成された REDO が、少なくとも 1 つの同期スタンバイ・データベースのスタンバイ REDO ログに書き込まれるまで、データ消失に対して Oracle Data Guard の保護を受けません。



非同期コミット機能の詳細は、次のドキュメントを参照してください。

- [Oracle Database 概要](#)
- [『Oracle Database PL/SQL 言語リファレンス』](#)

6.2 プライマリ・データベースのデータ保護モードの設定

保護モードは、構成が保護モードの前提条件を満たしているかぎり、オープン・データベースで設定および変更できます(最大パフォーマンス・モードから最大可用性モードへの変更を含む)。

次のステップを実行して、プライマリ・データベースのデータ保護モードを設定します。

1. 可用性、パフォーマンスおよびデータ保護の要件を満たすデータ保護モードを選択します。データ保護モードの詳細は、[「Oracle Data Guardの保護モード」](#)を参照してください。
2. 少なくとも1つのスタンバイ・データベースが、必要なデータ保護モードの転送要件を満たしていることを検証します。

少なくとも1つのスタンバイ・データベースに対応するLOG_ARCHIVE_DEST_*n*データベース初期化パラメータには、次の表にリストされている、必要なデータ保護モードに対応するREDO転送属性が含まれている必要があります。

また、スタンバイ・データベースにはスタンバイREDOログが必要です。

表6-1 データ保護モードに対するREDO転送の必須属性

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

3. プライマリ・データベースおよび各スタンバイ・データベースでDB_UNIQUE_NAMEデータベース初期化パラメータが一意的の値に設定されていることを確認します。
4. LOG_ARCHIVE_CONFIGデータベース初期化パラメータがプライマリ・データベースおよび各スタンバイ・データベースに定義されていて、プライマリ・データベースおよび各スタンバイ・データベースのDB_UNIQUE_NAMEを含むDG_CONFIGリストが、その値に指定されていることを確認します。

次のサンプルSQL文はLOG_ARCHIVE_CONFIGパラメータを構成します。

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

5. プライマリ・データベースで次のSQL文を実行して、データ保護モードを設定します。

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

オープン状態のデータベースでデータ保護モードをMAXIMUM PROTECTIONに設定できるのは、現在のデータ保護モードがMAXIMUM AVAILABILITYで、同期されているスタンバイ・データベースが少なくとも1つある場合のみです。

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

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

7 REDO転送サービス

Oracle Data Guard構成では、Oracle REDO転送サービスを構成および監視する必要があります。

次のトピックを参照してください。

- [REDO転送サービスの概要](#)
- [REDO転送サービスの構成](#)
- [カスケードされたREDO転送先](#)
- [カスケード・スタンバイでのデータ保護の考慮点](#)
- [構成の検証](#)
- [REDO転送サービスの監視](#)
- [REDO転送のチューニング](#)

7.1 REDO転送サービスの概要

REDO転送サービスは、Oracle Data Guard構成のメンバーとその他のデータベースとの間でREDOデータの自動転送を実行します。

次のREDO転送先がサポートされています。

- Oracle Data Guardスタンバイ・データベース
このガイドでは、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・データベースおよびスナップショット・スタンバイ・データベースの作成および管理方法について説明します。
- アーカイブ・ログ・リポジトリ
この宛先タイプは、アーカイブREDOログ・ファイルの一時オフサイト記憶域に使用されます。アーカイブ・ログ・リポジトリは、Oracleデータベース・インスタンスとフィジカル・スタンバイ制御ファイルで構成されます。アーカイブ・ログ・リポジトリにはデータファイルが含まれないため、ロールの推移はサポートできません。
アーカイブ・ログ・リポジトリの作成に使用される手順は、データファイルのコピーを除いて、フィジカル・スタンバイ・データベースの作成に使用される手順と同じです。
- Oracle Streamsダウストリーム取得データベース
- 遠隔同期インスタンス
遠隔同期インスタンスの詳細は、[遠隔同期インスタンスの使用](#)を参照してください。
- Zero Data Loss Recovery Appliance (リカバリ・アプライアンス)

各REDO転送先は、次の2つのREDO転送モードのいずれかによりREDOデータを受信するように、個別に構成されます。

- 同期
同期REDO転送モードは、REDOデータをトランザクション・コミットに対して同期させて転送します。トランザクションによって生成されたすべてのREDOが、同期REDO転送モードを使用する使用可能なREDO転送先すべてに正常に送信されるまで、そのトランザクションはコミットできません。
プライマリ・データベースとSYNC REDO転送先との間の距離に制限はありませんが、プライマリ・データベースとSYNC REDO転送先との間のネットワーク待機時間が長くなるに従って、トランザクションのコミットの待機時間も長くなります。

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



ノート:

同期 REDO 転送は Zero Data Loss Recovery Appliance でサポートされていません。

- 非同期

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

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

7.2 REDO転送サービスの構成

REDOデータの送受信を行う前に、Oracleデータベースを構成する必要があります。構成プロセスの一環として、REDO転送のセキュリティを設定します。

次のトピックを参照してください。

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

これらのトピックでは、次のトピックを十分に理解していることが求められます。

- データベース管理者の認証
- データベース初期化パラメータ
- REDOログの管理
- アーカイブREDOログの管理
- 高速リカバリ領域
- Oracle Net構成

7.2.1 REDO転送のセキュリティ

REDO転送では、Oracle Netセッションを使用してREDOデータを転送します。

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

7.2.1.1 SSLを使用したREDO転送の認証

Secure Sockets Layer(SSL)は、ネットワーク接続を保護するための業界標準プロトコルです。

SSLでは、RSA公開キー暗号化および対称キー暗号化を使用して、認証、暗号化およびデータ整合性を提供します。SSLは、次の場合に2つのOracleデータベース間のREDO転送の認証に自動的に使用されます。

- データベースが同じOracle Internet Directory(OID)エンタープライズ・ドメインのメンバーであり、そのドメインが現行ユーザーのデータベース・リンクの使用を許可する場合
- データベースに対応するLOG_ARCHIVE_DEST_1およびFAL_SERVERの各データベース初期化パラメータで、SSL用に構成されたOracle Net接続記述子を使用する場合
- 各データベースに、データベースのOIDエントリの識別名(DN)と一致するDNが付けられたユーザー証明書を格納するOracleウォレットまたはサポートされるハードウェア・セキュリティ・モジュールがある場合

関連項目:

- SSLの詳細は、[『Oracle Databaseセキュリティ・ガイド』](#)を参照してください。
- エンタープライズ・ドメイン管理の詳細は、[『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』](#)を参照してください。
- Oracle Internet Directoryの管理方法の詳細は、[『Oracle Label Security管理者ガイド』](#)を参照してください。

7.2.1.2 パスワード・ファイルを使用したREDO転送の認証

SSL認証要件を満たしていない場合、各データベースはリモート・ログイン・パスワード・ファイルを使用する必要があります。

Oracle Data Guard構成では、すべてのフィジカルおよびスナップショット・スタンバイ・データベースはプライマリ・データベースからのパスワード・ファイルのコピーを使用する必要があります。ソース・データベースとターゲット・データベースの両方でデータベースの互換性が12.2以降に設定されている場合は、管理権限(SYSDG、SYSOPER、SYSDBAなど)が付与または解除されたとき、および管理権限を持つユーザーのパスワードが変更されたときに、パスワード・ファイルのコピーが自動的にリフレッシュされます。唯一の例外は、遠隔同期インスタンスです。遠隔同期インスタンスはREDOを受け取りますが、適用しないため、更新したパスワード・ファイルを引き続き手動で遠隔同期インスタンスにコピーする必要があります。遠隔同期インスタンスでパスワード・ファイルが更新されると、プライマリでのパスワードの更新を含むREDOは、遠隔同期インスタンスからREDOログを受け取るように設定されているすべてのスタンバイ・データベースに自動的に伝播されます。REDOが適用されると、スタンバイでパスワード・ファイルが更新されます。

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

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

関連項目:

リモート・ログイン・パスワード・ファイルの作成とメンテナンスの詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。

7.2.2 REDOデータを送信するためのOracleデータベースの構成

REDO転送先を指定するには、LOG_ARCHIVE_DEST_*n*データベース初期化パラメータを使用します(*n*は1から31の整数)。

各LOG_ARCHIVE_DEST_*n*パラメータに対応するLOG_ARCHIVE_DEST_STATE_*n*データベース初期化パラメータ(*n*は1から31の整数)があります。このパラメータを使用すると、対応するREDO転送先を有効または無効にできます。[表7-1](#)に、このパラメータに割り当てることができる有効な値を示します。

表7-1 LOG_ARCHIVE_DEST_STATE_*n*初期化パラメータの値

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

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

SERVICE属性は、REDO転送先の必須属性で、属性リストの最初に指定する必要があります。SERVICE属性は、REDO転送先への接続に使用されるOracle Netサービス名の指定に使用されます。サービス名は、Oracle Netネーミング・メソッドによって、REDO転送先のOracle Netリスナーと一致するOracle Net接続記述子に解決できる必要があります。この接続記述子は、REDO転送先のデフォルト接続タイプでなく、専用サーバー接続を使用するように指定する必要があります。

関連項目:

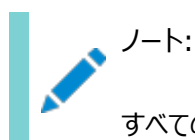
Oracle Netサービス名、接続記述子、リスナー、ネットワーク・セキュリティの詳細は、[『Oracle Database Net Services管理者ガイド』](#)を参照してください。

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

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

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

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



ノート:

すべての同期スタンバイ宛先に対してグローバルな、データベース初期化パラメータ DATA_GUARD_SYNC_LATENCY を

設定することもできます。少なくとも 1 つの同期スタンバイが REDO の受信を確認した後、後続の宛先を切断するまで、プライマリ・データベースが待機できる最大時間(秒単位)を定義します。

たとえば、3 つの同期スタンバイ宛先があり、DATA_GUARD_SYNC_LATENCY の値を 2 に設定するとします。最初のスタンバイが REDO の受信をただちに確認した場合、プライマリ・データベースは他の 2 つのスタンバイが応答するのを 2 秒間待ちます。1 つまたは両方が 2 秒以内に応答した場合、これらはアクティブな宛先として保持されます。時間内に応答しない宛先は失敗したものとしてマークされます。どちらの場合も、1 つの同期スタンバイが REDO の受信を確認したため、プライマリはゼロ・データ損失保護モードのままになります。失敗した同期スタンバイは、REOPEN 属性に指定された秒数が経過した後、通常どおりに再接続されます。

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

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

DB_UNIQUE_NAME属性を指定する場合、その値はDG_CONFIGリスト内のDB_UNIQUE_NAME値のいずれかと一致している必要があります。また、REDO転送先のDB_UNIQUE_NAMEデータベース初期化パラメータの値とも一致している必要があります。いずれかが一致していない場合、エラーがログに記録され、その宛先にREDO転送を行うことはできません。

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

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

COMPRESSION属性は、REDOデータを圧縮した形式でREDO転送先に転送するよう指定するために使用します。REDO転送を圧縮することにより、帯域幅が小さく待機時間が長いネットワーク・リンクにおいてREDO転送のパフォーマンスを大幅に向上できます。

REDO転送の圧縮は、Oracle Advanced Compressionオプションの機能です。REDO転送の圧縮機能を使用する前に、このオプションのライセンスを購入する必要があります。

次の例では、この項で説明したLOG_ARCHIVE_DEST_*n*の属性をすべて使用しています。DB_UNIQUE_NAMEは、圧縮の使用と同じく、両方の宛先に指定されています。REDO転送障害がいずれかの宛先で発生した場合、REDO転送はその宛先への再接続を試行しますが、60秒に1回より長い間隔で試行します。

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

ノート:



Zero Data Loss Recovery Appliance (リカバリ・アプライアンス)の構成は、スタンバイ・データベースの構成と同じになります。前述の例では、Chicago は ASYNC 転送先であるため、スタンバイ・データベースとリカバリ・アプライアンスのいずれの可能性もあります。(同期 REDO 転送はリカバリ・アプライアンスでサポートされていません)

7.2.2.1 V\$ARCHIVE_DESTビューを使用した属性の表示

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

7.2.3 REDOデータを受信するためのOracleデータベースの構成

REDO転送先は、REDOソース・データベースからREDOデータを受信およびアーカイブするように構成する必要があります。

次の各項を参照してください。

- [スタンバイREDOログの管理](#)
- [REDOがアーカイブREDOログ・ファイルに直接書き込まれる場合](#)

7.2.3.1 スタンバイREDOログの管理

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

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

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

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

スタンバイREDOログには、REDOソース・データベースのREDOスレッドごとに、REDOソース・データベースのREDOログより少なくとも1つ多いREDOログ・グループが必要です。REDOソース・データベースで、REDOソース・データベースのREDOログにいくつREDOログ・グループがあるかを確認するためにV\$LOGビューに問合せを行い、REDOソース・データベースにいくつREDOスレッドが存在するかを確認するためにV\$THREADビューに問合せを行います。

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

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

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

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

REDOソース・データベースがOracle Real Applications Cluster(Oracle RAC)またはOracle Real Application Clusters One Node (Oracle RAC One Node)データベースの場合、REDOソース・データベースのV\$LOGビューに問合せを行い、いくつREDOスレッドが存在するかを確認し、スタンバイREDOログにREDOログ・グループを追加する際に、対応するスレッド数を指定します。

2つのREDOスレッドを持つREDOソース・データベースからREDOを受信するためのスタンバイREDOログをデータベースに作成するには、次のサンプルSQL文が使用されます。

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 SIZE 500M;  
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 SIZE 500M;  
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 SIZE 500M;  
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 2 SIZE 500M;  
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 2 SIZE 500M;  
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 2 SIZE 500M;
```

ノート:



REDO ログ・グループをプライマリ・データベースに 1 つ追加するたびに、構成内の各スタンバイ・データベースのスタンバイ REDO ログにもログ・グループを 1 つ追加する必要があります。そうしないと、プライマリ・ログ・スイッチ後にスタンバイ・データベースが非同期化する可能性があり、その結果、一時的にデータ消失のないフェイルオーバーが機能しなくなったり、最大保護モードで稼働しているプライマリ・データベースが停止したりしかねません。

7.2.3.2 REDOがアーカイブREDOログ・ファイルに直接書き込まれる場合

スタンバイ・データベースにより受信されたREDOは、スタンバイREDOログ・グループが使用できない場合、またはREDOギャップを解決するためにREDOが送信された場合に、アーカイブREDOログ・ファイルに直接書き込まれます。これが行われた場合、別のデータベースから受信したアーカイブREDOについて有効なREDOはLOG_ARCHIVE_DEST_*n*パラメータのLOCATION属性で指定された場所へ書き込まれます。この目的で使用されるLOG_ARCHIVE_DEST_*n*パラメータは、スタンバイ・データベースのマウント時に決定され、LOG_ARCHIVE_DEST_*n*パラメータが修正されるたびに、再評価されます。

7.3 カスケードされたREDO転送先

カスケードされたREDO転送先(ターミナル宛先とも呼ばれる)では、プライマリ・データベースREDOを、プライマリ・データベースから直接ではなく、スタンバイ・データベースから間接的に受信します。

プライマリ・データベースREDOを1つ以上のカスケードされたターミナル宛先に、変更をそのローカル・データベース・ファイルに適用すると同時にカスケードするフィジカル・スタンバイ・データベースは、カスケード・スタンバイ・データベースとして知られています。

カスケードにより、プライマリ・データベースからカスケード・スタンバイ・データベースへのREDO転送の実行に関連したオーバーヘッドが軽減されます。

カスケード・スタンバイ・データベースは、プライマリ・データベースREDOを最大30のターミナル宛先にカスケードできます。

カスケード・スタンバイ・データベースは、リアルタイムまたは非リアルタイムでREDOをカスケードできます。リアルタイムではREDOがスタンバイREDOログ・ファイルに書き込まれ、非リアルタイムではすべてのスタンバイREDOログ・ファイルがカスケード・スタンバイでアーカイブされます。

カスケードには、次の制限があります。

- フィジカル・スタンバイ・データベースのみがREDOをカスケードできます。

- リアルタイム・カスケードにはOracle Active Data Guardオプションのライセンスが必要です。
- 非リアルタイム・カスケードは宛先1から10まででのみサポートされます。(リアルタイム・カスケードはすべての宛先でサポートされます。)

ノート:



Oracle Database アップグレード時のカスケードされた REDO 転送先の処理方法の詳細は、[Oracle Database ソフトウェアをパッチ適用またはアップグレードする前の注意事項](#)を参照してください。

また、次のトピックも参照してください:

- [ターミナル宛先の構成](#)
- [カスケードのシナリオ](#)

7.3.1 ターミナル宛先の構成

このステップでは、ターミナル宛先を構成する方法について説明します。

1. カスケード・スタンバイ・データベースとして構成するフィジカル・スタンバイ・データベースを選択します。
2. カスケード・スタンバイ・データベースで、FAL_SERVERデータベース初期化パラメータに、プライマリ・データベース、またはプライマリ・データベースからREDOを直接受信するスタンバイ・データベースのOracle Net別名を指定して構成します。
3. カスケード・スタンバイ・データベースで、1つまたは複数のターミナル宛先に対しLOG_ARCHIVE_DEST_*n*データベース初期化パラメータを構成します。この宛先のSERVICE属性にターミナル宛先のOracle Net別名を指定し、スタンバイ・ロールである間にスタンバイREDOログのアーカイブが有効になるようにVALID属性を構成します。

宛先1から10にASYNC転送モードを指定すると、REDOはリアルタイムで送信されます。転送モードを指定しない、または宛先1から10にSYNC転送モードを指定すると、REDOは非リアルタイムで送信されます。宛先11から31はASYNC(リアルタイム)転送モードでのみ動作します。

4. ターミナル宛先では、FAL_SERVERデータベース初期化パラメータに、カスケード・スタンバイ・データベース、またはプライマリ・データベースに直接接続されている別のスタンバイ・データベースのOracle Net別名を指定して構成します。プライマリ・データベースを指定することも可能ですが、そうするとプライマリ・データベースでのREDO転送オーバーヘッドを減らすというカスケードの目的が果せなくなります。
5. [例7-1](#)は、bostonというプライマリ・データベースがREDOをboston2というローカル・フィジカル・スタンバイ・データベースに送信し、そこからプライマリ・データベースREDOがdenverというリモート・フィジカル・スタンバイ・データベースにカスケードされるというOracle Data Guard構成のメンバーによって使用されるいくつかのデータベース初期化パラメータを示しています。

LOG_ARCHIVE_DEST_*n*データベース初期化パラメータは、データベースbostonがスタンバイ・ロールのときに、データベースdenverに対するスタンバイREDOログ・アーカイブとして有効なデータベースbostonでも構成できます。これにより、データベースbostonとデータベースboston2の間でスイッチオーバーが実行された場合でも、データベースdenverへのREDOカスケードが続行可能になります。

例7-1 REDOのカスケード時に使用される初期化パラメータの一部

プライマリ・データベース

```
DB_UNIQUE_NAME=boston
```

```
FAL_SERVER=boston2

LOG_ARCHIVE_CONFIG=' DG_CONFIG= (boston, boston2, denver) '

LOG_ARCHIVE_DEST_1=' LOCATION=USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(ALL_LOGFILES, ALL_ROLES) DB_UNIQUE_NAME=boston'

LOG_ARCHIVE_DEST_2=' SERVICE=boston2 SYNC
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=boston2'
```

カスケードしているフィジカル・スタンバイ・データベース

```
DB_UNIQUE_NAME=boston2

FAL_SERVER=boston

LOG_ARCHIVE_CONFIG=' DG_CONFIG= (boston, boston2, denver) '

LOG_ARCHIVE_DEST_1=' LOCATION= USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(ALL_LOGFILES, ALL_ROLES) DB_UNIQUE_NAME=boston2'

LOG_ARCHIVE_DEST_2=' SERVICE=denver
VALID_FOR=(STANDBY_LOGFILES, STANDBY_ROLE) DB_UNIQUE_NAME=denver'
```

カスケードされたフィジカル・スタンバイ・データベース

```
DB_UNIQUE_NAME=denver

FAL_SERVER=boston2

LOG_ARCHIVE_CONFIG=' DG_CONFIG= (boston, boston2, denver) '

LOG_ARCHIVE_DEST_1=' LOCATION= USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(ALL_LOGFILES, ALL_ROLES) DB_UNIQUE_NAME=denver'
```

カスケード環境の設定後の構成の検証の詳細は、[「構成の検証」](#)を参照してください。

7.3.2 カスケードのシナリオ

Data Guard構成は、1つまたは複数のフィジカル・スタンバイにカスケードするように設定できます。

次のトピックを参照してください。

- [フィジカル・スタンバイへのカスケード](#)
- [複数のフィジカル・スタンバイへのカスケード](#)

7.3.2.1 フィジカル・スタンバイへのカスケード

このステップでは、1つのフィジカル・スタンバイ・データベースにカスケードする例を示します。

このシナリオでは、きわめて重要なプライマリ・データベースがあります。このデータベースにはパフォーマンスやデータ保護に厳しい要件があるため、ローカルのフィジカル・スタンバイ・データベースをデプロイして、データ消失のない保護と、プライマリ・データベースとローカル・スタンバイ・データベースのサイトで地域災害に備えるための、カスケードされたりモート・フィジカル・スタンバイ・データベースを準備することに決定しました。

次のステップを実行することで、前述の目的を達成できます。

1. ローカル・サイトにフィジカル・スタンバイ・データベースを作成します。

2. プライマリ・データベースとローカル・スタンバイ・データベースのサイトでの地域災害に対する保護策として、十分に遠く離れたサイトでフィジカル・スタンバイ・データベースを作成します。
3. ローカル・スタンバイ・データベースを、プライマリ・データベースのSYNC REDO転送先として構成します。
4. リモート・フィジカル・スタンバイ・データベースを、ローカル・スタンバイ・データベースのターミナル宛先として構成します。

7.3.2.2 複数のフィジカル・スタンバイへのカスケード

このステップでは、複数のフィジカル・スタンバイ・データベースにカスケードする例を示します。

このシナリオでは、北アメリカにプライマリ・データベースがあり、読取り専用のレポート生成アプリケーションをサポートするために、ヨーロッパにこのデータベースのレプリカを3つデプロイしようと考えています。コストおよびパフォーマンス上の理由から、北アメリカからヨーロッパの各サイトへのネットワーク・リンクを維持しないつもりです。

次のステップを実行することで、前述の目的を達成できます。

1. 北アメリカのサイトとヨーロッパのサイトの1つとの間にネットワーク・リンクを作成します。
2. ヨーロッパの各サイトにフィジカル・スタンバイ・データベースを作成します。
3. [「フィジカル・スタンバイ・データベースのオープン」](#)で説明しているように、フィジカル・スタンバイ・データベースをリアルタイム問合せモードでオープンします。
4. 欧米間のネットワーク・リンクのヨーロッパ側の終点にあるフィジカル・スタンバイ・データベースを、他のヨーロッパのスタンバイ・データベースにREDOをカスケードするように構成します。
5. その他の2つのフィジカル・スタンバイ・データベースを、ステップ4で構成したカスケード・スタンバイ・データベースのターミナル宛先として構成します。

7.4 カスケード・スタンバイでのデータ保護の考慮点

構成にカスケード・スタンバイが含まれる場合、各宛先には、フェイルオーバー中に使用するそのソースを指すように定義されたLOG_ARCHIVE_DEST_*n*パラメータが含まれる必要があります。

リアルタイム・カスケードでは、カスケード・スタンバイ・データベースが、非同期REDO転送を使用してプライマリ・データベースからREDOを直接受信するスタンバイ・データベースとほぼ同じレベルのデータ保護を提供できます。しかし、REDOはリアルタイムで転送されますが、停止によりすべてのREDOがターミナル宛先に到達できなくなると、2つ目のネットワーク・ホップが存在する事実により、さらなるデータ消失の可能性が生じます。

7.5 構成の検証

Oracle Data Guard構成を作成後に検証するには、構成内の任意のデータベースからV\$DATAGUARD_CONFIGビューを問い合わせます。

ビューには、DB_UNIQUE_NAMEおよびLOG_ARCHIVE_CONFIG初期化パラメータで定義された一意のデータベース名が表示されます。

関連項目:

- V\$DATAGUARD_CONFIGビューの詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

7.6 REDO転送サービスの監視

REDO転送ステータスとともに、REDO転送のレスポンス時間を監視できます。

次のトピックを参照してください。

- [REDO転送ステータスの監視](#)
- [同期REDO転送のレスポンス時間の監視](#)
- [REDOギャップの検出および解決](#)
- [REDO転送サービスの待機イベント](#)

7.6.1 REDO転送ステータスの監視

REDOソース・データベースでREDO転送ステータスを監視するためにビューを問い合わせることができます。

次のステップを実行して、REDOソース・データベースでREDO転送ステータスを監視します。

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

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

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

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

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

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

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

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

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

THREAD#	SEQUENCE#
1	12
1	13
1	14

4. REDOソース・データベースおよび各REDO転送先でLOG_ARCHIVE_TRACEデータベース初期化パラメータを設定し、REDO転送の進捗状況をトレースします。詳細と例は、[「アーカイブ・トレースの設定」](#)を参照してください。

7.6.2 同期REDO転送のレスポンス時間の監視

V\$REDO_DEST_RESP_HISTOGRAMビューには、REDO転送先ごとのレスポンス時間データが格納されます。

レスポンス時間データは、同期REDO転送モードにより送信されたREDO転送メッセージに関して保持されます。

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

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

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

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

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

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

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

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

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

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

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

ノート:



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

7.6.3 REDOギャップの検出および解決

REDOギャップは、REDO転送が中断されるたびに発生します。

REDO転送が再開されると、REDO転送サービスはREDOギャップを自動的に検出し、欠落したREDOを宛先に送信して解決します。

REDOギャップの解決に必要な時間は、ギャップのサイズに正比例し、REDOソース・データベースとREDO転送先との間のネットワーク・リンクの実効スループットに反比例します。REDO転送圧縮を使用して、REDOギャップの解決時間を短縮します。LOG_ARCHIVE_DEST_nパラメータのCOMPRESSION属性を使用して、宛先に送信する前にREDOデータを圧縮するように指定します。

COMPRESSION属性の詳細は、[LOG_ARCHIVE_DEST_nパラメータの属性](#)を参照してください。

7.6.3.1 手動によるギャップの解決

場合によっては、ギャップの解決を自動的に実行できず、手動による実行が必要になります。

たとえば、プライマリ・データベースが使用できない場合には、ロジカル・スタンバイ・データベースでREDOギャップの解決を手動で実行する必要があります。

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

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

THREAD#	LOW_SEQUENCE#	HIGH_SEQUENCE#
1	7	10

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

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

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE THREAD#=1 AND -  
> DEST_ID=1 AND SEQUENCE# BETWEEN 7 AND 10;
```

NAME

```
-----  
/primary/thread1_dest/arcr_1_7.arc  
/primary/thread1_dest/arcr_1_8.arc  
/primary/thread1_dest/arcr_1_9.arc
```

ノート:



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

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

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

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

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

ノート:



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

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

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

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

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

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

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

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

ノート:



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

7.6.4 REDO 転送サービスの待機イベント

Oracle 待機イベントを使用して、REDO ソース・データベース上の REDO 転送待機時間を追跡できます。

[表 7-2](#) には、これらの Oracle 待機イベントをいくつかリストします。それらは、V\$SYSTEM_EVENT 動的パフォーマンス・ビュー内にあ

ります。

REDO転送で使用するOracle待機イベントの詳細リストは、次の場所からOracle Maximum Availability Architecture(MAA)のホームページにアクセスし、『Oracle Data Guard Redo Transport and Network Best Practices』技術概要を参照してください。

<http://www.oracle.com/goto/maa>

表7-2 REDO転送の待機イベント

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

7.7 REDO転送のチューニング

最高のパフォーマンスを得られるようにREDO転送を最適化できます。

Data Guardでは、パフォーマンスを最適化するために、REDO転送を自動的にチューニングします。ただし、チューニングをより詳細に制御する必要がある場合は、REDO転送で使用する次の領域をチューニングできます。

- ネットワーク構成
- 記憶域構成
- 高速リカバリ領域
- REDO転送構成

次のOracle Maximum Availability Architecture (MAA)のホームページで入手できる、Oracle Data Guard REDO転送とネットワーク構成のベスト・プラクティスに関する技術概要を参照してください。

<http://www.oracle.com/goto/maa>

8 適用サービス

次の概要では、REDOデータをスタンバイ・データベースに適用する方法について説明します。

- [適用サービスの概要](#)
- [適用サービスの構成オプション](#)
- [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](#)
- [プライマリでのPDBの削除または名前の変更時のスタンバイの考慮点](#)

8.1 適用サービスの概要

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

デフォルト設定では、適用サービスはスタンバイREDOログ・ファイルのアーカイブ保存が終了するまで待って、制約のあるREDOを適用します。ただし、リアルタイムでの適用を有効化しておくと、書込みと同時に、現行のスタンバイREDOログ・ファイルにREDOを適用することができます。リアルタイム適用については、[「リアルタイム適用によるREDOデータの即時適用」](#)で詳しく説明しています。

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

- REDO Apply(フィジカル・スタンバイ・データベースのみ)
メディア・リカバリを使用して、プライマリ・データベースとフィジカル・スタンバイ・データベースの同期を維持します。
- SQL Apply(ロジカル・スタンバイ・データベースのみ)
プライマリ・データベースから受信したREDOに基づいてSQL文を再構成し、そのSQL文をロジカル・スタンバイ・データベースに対して実行します。

8.2 適用サービスの構成オプション

REDOデータをすぐに適用することも、アーカイブREDOログ・ファイルを適用する遅延時間を指定することもできます。

次のトピックを参照してください。

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

8.2.1 リアルタイム適用によるREDOデータの即時適用

リアルタイム適用の機能を有効化すると、現行のスタンバイREDOログ・ファイルのアーカイブを待たずに、適用サービスによって受信したREDOデータを適用することができます。

この方法ではフェイルオーバーまたはスイッチオーバーが開始するまでに、スタンバイREDOログ・ファイルがスタンバイ・データベースに適用されているので、これによりスイッチオーバーとフェイルオーバーの速度が向上します。プライマリ・データベースとより厳密に同期が維持されることにより、Oracle Active Data Guardスタンバイでのリアルタイムのレポートも可能になります。

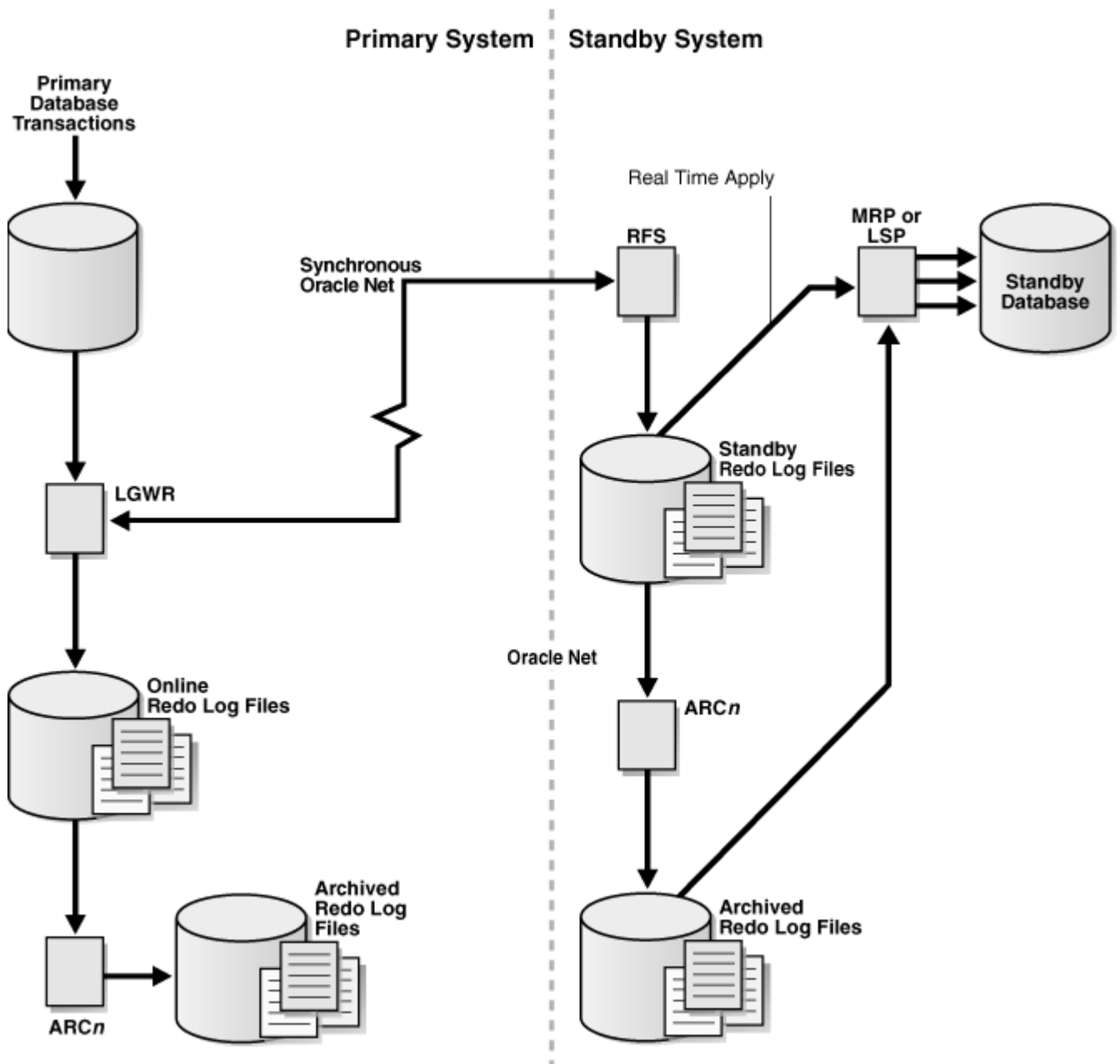
次のように、ALTER DATABASE文を使用して、リアルタイム適用機能を使用可能にします。

- フィジカル・スタンバイ・データベースの場合は、ALTER DATABASE RECOVER MANAGED STANDBY DATABASE文を発行します。(Oracle Database 12cリリース1 (12.1)現在、USING CURRENT LOGFILE句は非推奨で、リアルタイム適用を開始する必要はなくなりました)。
- ロジカル・スタンバイ・データベースの場合は、ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE文を発行します。

リアルタイムの適用には、スタンバイREDOログを使用して構成された、ARCHIVELOGモードのスタンバイ・データベースが必要です。

図8-1に、ローカル宛先とスタンバイ宛先があるOracle Data Guard構成を示します。リモート・ファイル・サーバー(RFS)プロセスは、スタンバイ・データベース上のスタンバイREDOログ・ファイルにREDOデータを書き込むため、適用サービスでは、スタンバイREDOログ・ファイルがいっぱいになったときにREDOをリカバリできます。

図8-1 リアルタイム適用を使用したスタンバイ宛先へのREDOデータの適用



8.2.2 アーカイブREDOログ・ファイルの適用に対する遅延時間の指定

プライマリ・サイトからREDOデータを受信した後、そのREDOデータをスタンバイ・データベースに適用するまでの間にタイム・ラグを作成することが必要な場合があります。

時間間隔(分単位)を指定すると、破損したデータや誤ったデータがスタンバイ・サイトに適用されるのを防止できます。DELAY間隔を設定すると、スタンバイ・データベースへのREDOデータの転送が遅延することはありません。かわりに、指定したタイム・ラグは、REDOデータがスタンバイ宛先で完全にアーカイブされたときに開始します。

ノート:



リアルタイム適用が使用可能になっている宛先に遅延を定義すると、その遅延は無視されます。次の段落に示すような遅延を定義する場合、[「Redo Apply の開始」](#)に示すように、USING ARCHIVED LOGFILE 句を使用して適用を開始する必要があります。

遅延時間の指定

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

遅延時間の取消し

次のように、指定した遅延間隔を取り消すことができます。

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

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

- ロジカル・スタンバイ・データベースで、次のSQLコマンドを指定します。

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

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

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

適用遅延を設定するかわりにフラッシュバック・データベースを使用して、スタンバイ・データベースへの破損データやエラー・データの適用からリカバリできます。

フラッシュバック・データベースを使用すると、スタンバイ・データベースを任意の時点まで、すばやく容易にフラッシュバックできます。

データベースのフラッシュバックでのOracle Data Guardの使用法を示すシナリオは[「Oracle Data Guardの使用例」](#)を参照してください。フラッシュバック・データベースの有効化と使用の詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』](#)を参照してください。

8.3 REDOデータのフィジカル・スタンバイ・データベースへの適用

REDO Applyを実行している場合、フィジカル・スタンバイ・データベースはリアルタイム機能を使用して、スタンバイREDOログ・ファイルがリモート・ファイル・サーバー(RFS)プロセスによって書き込まれているときに、スタンバイREDOログ・ファイルから直接REDOを適用できます。

この項では、次の項目について説明します。

- [REDO Applyの開始](#)
- [REDO Applyの停止](#)

- [フィジカル・スタンバイ・データベースでのREDO Applyの監視](#)

8.3.1 REDO Applyの開始

フィジカル・スタンバイ・データベースで適用サービスを開始するには、フィジカル・スタンバイ・データベースが開始していてマウントされていることを確認してから、REDO Applyを開始します。

次のようにフィジカル・スタンバイ・データベース上で適用サービスを開始します。

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

スタンバイ・データベースがスタンバイREDOログを使用して構成され、ARCHIVELOGモードであるならば、これによりリアルタイム適用も自動的に有効になります。

REDO Applyをフォアグラウンド・セッションまたはバックグラウンド・プロセスとして動作させることができます。REDO Applyをフォアグラウンドで開始するには、次のSQL文を発行します。

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

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

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

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

または

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING ARCHIVED LOGFILE 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。このコマンドによって、カレントSQLセッションが切断されることはありません。

8.3.2 REDO Applyの停止

Redo Applyを停止するには、ALTER DATABASE SQL文を使用します。

たとえば、次のSQL文を発行します。

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

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

フィジカル・スタンバイ・データベースで適用サービスのステータスを監視できます。

[「プライマリ、フィジカルおよびスナップショット・スタンバイ・データベースを監視するためのビューの使用」](#)を参照してください。スタンバイ・データベースは、Oracle Enterprise Manager Cloud Controlを使用して監視することもできます。

8.4 REDOデータのロジカル・スタンバイ・データベースへの適用

SQL Applyは、アーカイブREDOログまたはスタンバイREDOログのデータをSQL文に変換してから、そのSQL文をロジカル・スタンバイ・データベースで実行します。

ロジカル・スタンバイ・データベースはオープン状態のままであるため、メンテナンスされる表は、レポート生成、要約、問合せなどの他のタスクで同時に使用できます。

次のトピックを参照してください。

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

8.4.1 SQL Applyの開始

SQL Applyを開始するには、ALTER DATABASE SQL文を使用します。

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

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

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

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

8.4.2 ロジカル・スタンバイ・データベースでのSQL Applyの停止

ロジカル・スタンバイ・データベースでSQL Applyを停止するには、ALTER DATABASE SQL文を使用します。

たとえば、ロジカル・スタンバイ・データベースで、次の文を入力します。

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

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

8.4.3 ロジカル・スタンバイ・データベースでのSQL Applyの監視

ロジカル・スタンバイ・データベースでSQL Applyを管理および監視するために使用できる情報を提供するビューがあります。

[「ロジカル・スタンバイ・データベースの管理および監視関連のビュー」](#)を参照してください。スタンバイ・データベースは、Oracle Enterprise Manager Cloud Controlを使用して監視することもできます。[「Oracle Data Guardのトラブルシューティング」](#)を参照してください。

8.5 プライマリでのPDBの削除または名前の変更時のスタンバイの考慮点

プライマリがマルチテナント・コンテナ・データベース(CDB)である場合、プライマリでプラガブル・データベース(PDB)を削除または名前変更する際には、制限が適用されます。

- PDBでDDL UNPLUGおよびDROP操作を実行するには、PDBはすべてのスタンバイ・データベースと同様にプライマリで最初に閉じられる必要があります。
- PDBでDDL RENAME操作を実行するには、PDBはプライマリで最初にOPEN RESTRICTEDモードになり、すべてのスタンバイ・データベースで閉じられる必要があります。

プライマリ・データベースで削除または名前を変更する前に、スタンバイでPDBを閉じないと、スタンバイはすべてのPDBへのリカバリ・プロセスを停止します。スタンバイで削除されたPDBを閉じてから、次のSQL文を使用してリカバリを再開する必要があります。

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

9 ロールの推移

Oracle 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。

データベースの現行のロールを確認するには、V\$DATABASEビューのDATABASE_ROLE列を問い合わせます。

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

Oracle Data Guard構成でロールの推移を管理する方法の詳細は、次の各項を参照してください。

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

ノート:



これらの項では、SQL文を使用して、手動でロールの推移を実行する方法について説明します。ブローカにより管理される Oracle Data Guard 構成でロールの推移を実行する場合は、これらのマニュアルの手順は使用しないでください。[Oracle Data Guard Broker](#) で提供されているトランジション・プロシージャをかわりに使用します。

関連項目:

Oracle Data Guard Brokerの使用による以下の詳細は、『[Oracle Data Guard Broker](#)』を参照してください。

- Oracle Enterprise Manager Cloud Controlでキー・クリックを1回する、あるいはDGMGRLコマンドライン・インタフェースで単一のコマンドを使用すると起動できることにより、スイッチオーバーおよびフェイルオーバーを単純化します。
- **プライマリ・データベースが使用できなくなった場合に、ファスト・スタート・フェイルオーバーを有効化し、フェイルオーバーを自動的に実行します。**ファスト・スタート・フェイルオーバーが有効化されている場合、Oracle Data Guardブローカによってフェイルオーバーの必要性が判断され、指定されたターゲット・スタンバイ・データベースへのフェイルオーバーが自動的に開始されます。DBAによる操作は不要です。

9.1 ロールの推移の概要

データベースは、**プライマリ**または**スタンバイ**という相互に排他的なロールのいずれかで動作します。

Oracle Data Guardでは、SQL文を使用して、またはOracle Data Guard Brokerのいずれかのインタフェースを使用して、これらのロールを動的に変更できます。Oracle Data Guardは、次のロール・トランジションをサポートしています。

- **スイッチオーバー**

プライマリ・データベースが、スタンバイ・データベースの1つを使用してロールを切り替えられるようにします。スイッチオーバー中にデータが失われることはありません。スイッチオーバーの完了後、各データベースは、新しいロールとともに

Oracle Data Guard構成に継続して含まれます。

● フェイルオーバー

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

関連項目:

- [「ロールの推移の準備」](#)では、停止時間およびデータ消失のリスクを最小限にするために、どのロールの推移を選択するかについて説明します。
- スイッチオーバーの詳細は、[「スイッチオーバー」](#)を参照
- フェイルオーバーの詳細は、[「フェイルオーバー」](#)を参照してください
- ブローカの管理対象のフェイルオーバーが発生した場合にデータベース・クライアントが使用可能なイベント通知、およびデータベース接続フェイルオーバー・サポートの詳細は、[Oracle Data Guard Broker](#)を参照してください。

9.1.1 ロールの推移の準備

ロールの推移を開始する前に、各データベースが正しく構成されていること、およびスタンバイ・データベースにREDO転送エラーやREDOギャップがないことを確認する必要があります。

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

ノート:



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

- プライマリ・データベースでV\$ARCHIVE_DEST_STATUSビューを問い合せて、スタンバイ・データベースにREDO転送エラー、またはREDOギャップがないことを確認します。

たとえば、LOG_ARCHIVE_DEST_2に関連付けられているスタンバイ・データベースのステータスをチェックするには、次の問合せを使用します。

```
SQL> SELECT STATUS, GAP_STATUS FROM V$ARCHIVE_DEST_STATUS WHERE DEST_ID = 2;
```

```
STATUS GAP_STATUS
```

```
-----  
VALID NO GAP
```

スタンバイ・データベースに対応する行のSTATUS列の値がVALID、GAP_STATUS列の値がNOGAPになるまで、次に進まな

いでください。

- スタンバイ・データベースに存在する一時ファイルが、プライマリ・データベースの一時ファイルと一致することを確認します。
- 新たにプライマリ・データベースになるように設定されたスタンバイ・データベースで現在有効になっているREDOの適用遅延を解除します。遅延を解除しないと、スイッチオーバー時間が長くなり、スイッチオーバーが許可されなくなる場合があります。
- リアルタイム問合せモードにあるフィジカル・スタンバイ・データベースへのスイッチオーバーを実行する前に、できるだけ迅速に、クリーンにロール・トランジションを実施させるために、そのスタンバイ・データベースのすべてのインスタンスを、マウント済だがオープンされていない状態にすることを検討します。
- Oracle RACプライマリ・データベースからフィジカル・スタンバイ・データベースへのスイッチオーバーを実行する場合、1つを残してすべてのプライマリ・データベース・インスタンスを停止する必要はありません。

9.1.2 ロールの推移のターゲット・スタンバイ・データベースの選択

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

次のようなものがあります：

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

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

ノート：



スナップショット・スタンバイは、ロールの推移のターゲットにはできません。スナップショット・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使用するには、最初にこのデータベースをフィジカル・スタンバイ・データベースに変換し、プライマリ・データベースから受信したすべての REDO を適用できるようにします。[「フィジカル・スタンバイ・データベースへのスナップショット・スタンバイ・データベースの変換」](#)を参照してください。

Oracle Data GuardにはV\$DATAGUARD_STATSビューが用意されています。このビューを使用して、スタンバイ・データベースのデータの最新性の観点で各スタンバイ・データベースを評価し、使用可能なREDOデータがすべてスタンバイ・データベースに適用された場合のロールの推移の実行に必要な時間を評価できます。次に例を示します。

```
SQL> COLUMN NAME FORMAT A24
SQL> COLUMN VALUE FORMAT A16
SQL> COLUMN DATUM_TIME FORMAT A24
SQL> SELECT NAME, VALUE, DATUM_TIME FROM V$DATAGUARD_STATS;
```

NAME	VALUE	DATUM_TIME
transport lag	+00 00:00:00	06/18/2009 12:22:06
apply lag	+00 00:00:00	06/18/2009 12:22:06
apply finish time	+00 00:00:00.000	
estimated startup time	9	

この問合せの出力は、スタンバイ・データベースが、プライマリ・データベースにより生成されたすべてのREDOを受信し、適用したことを示しています。これらの統計は、2009年6月18日12:22.06にプライマリ・データベースから受信したデータを使用して計算されたものです。

apply lagおよびtransport lagメトリックは、プライマリ・データベースから受信されるデータに基づいて計算されます。これらのメトリックは、プライマリ・データベースとスタンバイ・データベースの間の通信が中断されると失効します。apply lagおよびtransport lagメトリックのDATUM_TIME列の値が変化していないことは、プライマリ・データベースとスタンバイ・データベース間の通信障害と考えられる原因で、これらのメトリックが更新されずに失効したことを示しています。

9.1.3 スイッチオーバー

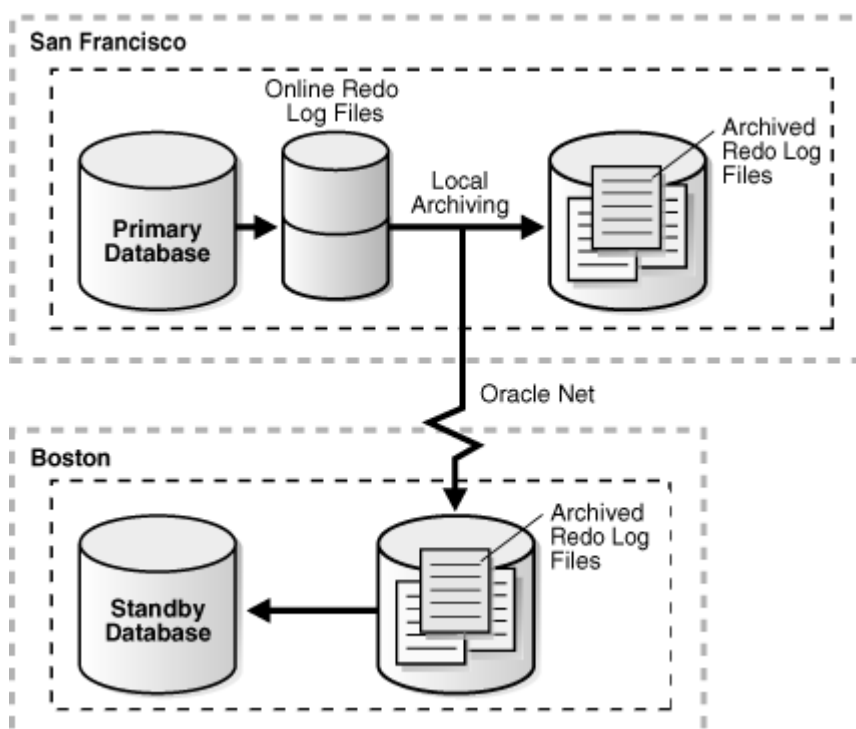
スイッチオーバーは、通常、計画停止時のプライマリ・データベースの停止時間を短縮するために使用されます。

計画済停止とは、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、またはOracleデータベース・ソフトウェアとパッチ・セットのローリング・アップグレードなどのイベントです。

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

[図9-1](#)は、データベースのロールを切り替える前の2つのサイトにあるOracle 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9-1 スイッチオーバー前のOracle Data Guard構成



[図9-2](#)に、元のプライマリ・データベースがスタンバイ・データベースにスイッチオーバーされた後、元のスタンバイ・データベースがまだ

新しいプライマリ・データベースになっていないOracle Data Guard環境を示します。このとき、Oracle 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9-2 新しいプライマリ・データベースへのスイッチオーバー前のスタンバイ・データベース

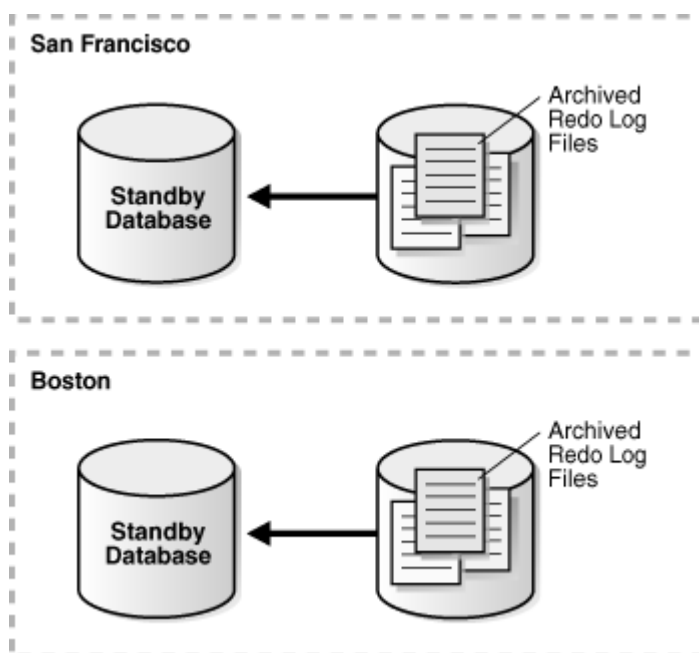
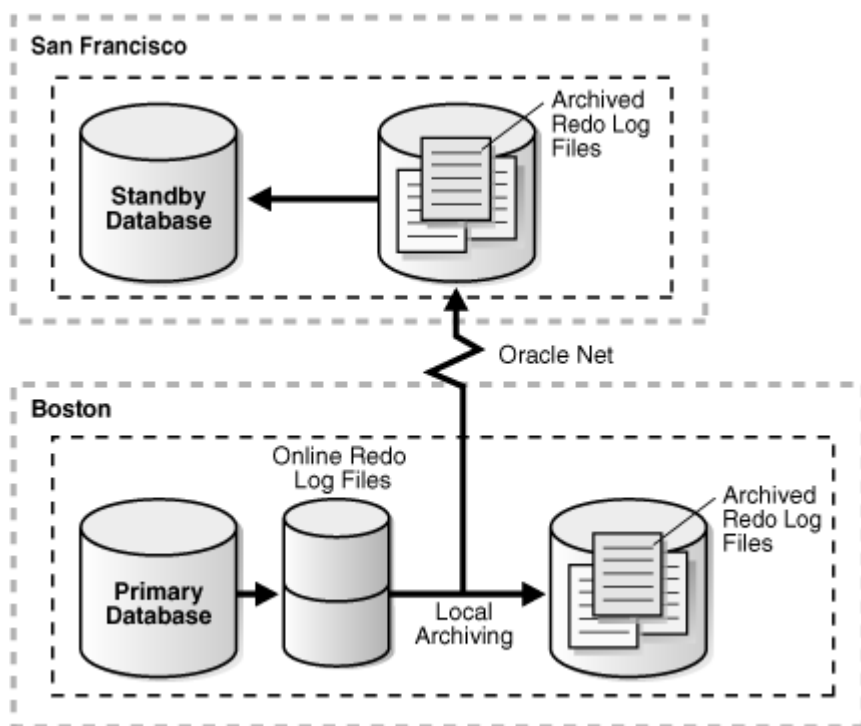


図9-3は、スイッチオーバー実行後のOracle Data Guard環境を示します。元のスタンバイ・データベースは新しいプライマリ・データベースになります。現在、プライマリ・データベースはボストンにあり、スタンバイ・データベースはサンフランシスコにあります。

図9-3 スwitchオーバー後のOracle Data Guard環境



スイッチオーバーの準備

「[ロールの推移の準備](#)」に示した前提条件が満たされていることを確認します。また、スイッチオーバーの場合は、次の前提条件が満たされている必要があります。

- フィジカル・スタンバイ・データベースが関与するスイッチオーバーの場合は、プライマリ・データベースがオープンし、REDO Applyがスタンバイ・データベースでアクティブであることを確認します。
- ロジカル・スタンバイ・データベースが関与するスイッチオーバーの場合は、プライマリおよびスタンバイ・データベース・インスタンスの両方がオープンし、SQL Applyがアクティブであることを確認します。

関連項目:

- [SQL Applyを使用したOracle Databaseのアップグレード](#)
- REDO Applyの詳細は、[「REDOデータのフィジカル・スタンバイ・データベースへの適用」](#)を参照してください
- SQL Applyの詳細は、[「REDOデータのロジカル・スタンバイ・データベースへの適用」](#)を参照してください

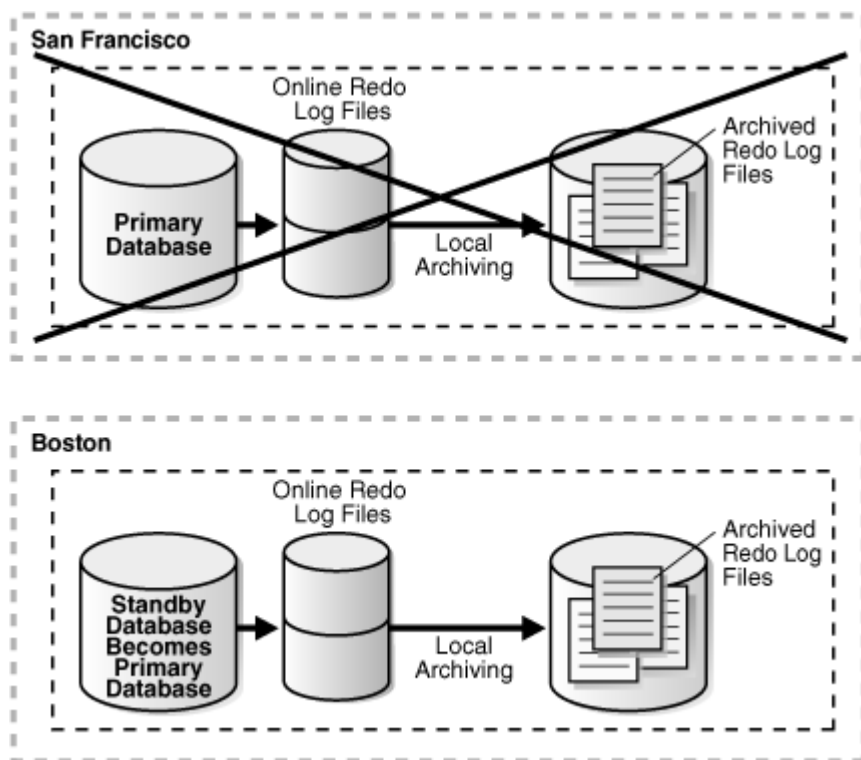
9.1.4 フェイルオーバー

通常、フェイルオーバーは、プライマリ・データベースが使用不可能になり、適正な時間内にプライマリ・データベースをリストアしてサービスを再開できない場合にのみ使用します。

フェイルオーバー時に実行するアクションは、フェイルオーバーに関与するスタンバイ・データベースがロジカルかフィジカルか、フェイルオーバー時のOracle Data Guard構成の状態、およびフェイルオーバーを開始するために使用する特定のSQL文によって異なります。

[図9-4](#)に、サンフランシスコにあるプライマリ・データベースからボストンにあるフィジカル・スタンバイ・データベースへのフェイルオーバーの結果を示します。

図9-4 スタンバイ・データベースへのフェイルオーバー



フェイルオーバーの準備

ノート:



フェイルオーバーのために選択した物理スタンバイ・データベースで管理されているスタンバイ・リカバリが、ORA-752 または ORA-600 [3020] のエラーで停止した場合は、このまま[「プライマリ・データベースでの書き込みの欠落エラーからのリカバリ」](#)に進んでください。

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

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

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

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

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

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

ノート:

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

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

9.1.5 ロールの推移のトリガー

ロール・トランジションが発生するたびに、DB_ROLE_CHANGEシステム・イベント信号が発行されます。

ロール・トランジション発生時にデータベースがオープン状態の場合は、このシステム・イベント信号が即座に発行されます。ロール・トランジション発生時にデータベースがクローズ状態の場合は、次回データベースがオープン状態になったときにシステム・イベント信号が発行されます。

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

9.2 フィジカル・スタンバイ・データベースが関与するロールの推移

Oracle Databaseリリース12cリリース1 (12.1)以降を実行している場合に、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およびフェイルオーバーを実行する手順が簡略化されました。

以前の手順も引き続きサポートされますが、次の項で説明している新しい手順を使用することをお勧めします。

- [フィジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)
- [フィジカル・スタンバイ・データベースへのフェイルオーバーの実行](#)

ノート:



Oracle Database リリース 18c では、ロールの推移中に Active Data Guard スタンバイ上のデータベース・バッファ・キャッシュに移入するためにディスクから読み取られた物理ブロックがアプリケーションのパフォーマンスに影響しないように、バッファ・キャッシュの状態が管理されます。これにより、ロールの推移後の新しいプライマリでアプリケーションのパフォーマンスが向上します。

ロールの推移中のフィジカル・スタンバイ・セッション接続の維持

Oracle Database 12cリリース2 (12.2.0.1)以降、フィジカル・スタンバイ・データベースがプライマリに変換されるときに、スイッチオーバー/フェイルオーバー中にすべてのセッションを中断せずフィジカル・スタンバイへの接続を維持するオプションがあります。

この機能を使用可能にするには、スタンバイ・インスタンスが開始する前にinit. oraファイルのSTANDBY_DB_PRESERVE_STATES初期化パラメータを設定します。このパラメータは、フィジカル・スタンバイ・データベースのみに適用されます。指定できる値は次のとおりです:

- NONE — スタンバイ上のユーザー・セッションまたは現在のバッファはスイッチオーバー/フェイルオーバー中に保持されません。これはデフォルト値です。
- ALL — ユーザー・セッションも現在のバッファもスイッチオーバー/フェイルオーバー中に保持されます。
- SESSION — ユーザー・セッションはスイッチオーバー/フェイルオーバー中に保持されます。
- BUFFER — 現在のバッファがスイッチオーバー/フェイルオーバー中に保持されます。

関連項目:

- フィジカル・スタンバイ・データベースへのロールの推移を実行するときに発生する可能性のある問題のトラブルシューティング方法については、[「Oracle Data Guardのトラブルシューティング」](#)を参照してください。
- 以前のリリースで使用された手順の詳細と、古い構文と新しい構文の比較については、[「古い構文を使用したロールの推移の実行」](#)を参照してください。
- STANDBY_DB_PRESERVE_STATES初期化パラメータの詳細は、[Oracle Databaseリファレンス](#)を参照してください。

9.2.1 フィジカル・スタンバイ・データベースへのスイッチオーバーの実行

このステップでは、フィジカル・スタンバイ・データベースへのスイッチオーバーを実行する方法について説明します。

ノート:



プライマリ・データベースおよびスタンバイ・データベースに接続する遠隔同期インスタンス(または優先および代替の遠隔同期インスタンスの組合せ)が存在する場合、スタンバイにスイッチオーバーする手順はこのトピックで説明しているものと同じです。遠隔同期インスタンスが使用可能か否かは、スイッチオーバーに影響しません。スイッチオーバー中、プライマリとスタンバイは直接相互に通信し、遠隔同期インスタンスを意識せずにスイッチオーバーのロールの推移ステップを実行できる必要があります。スイッチオーバー後、遠隔同期インスタンスが新しいロールの2つのデータベースにサービスを提供できるように構成を正しく設定する方法の例は、*Oracle Data Guard 概要および管理の遠隔同*

期インスタンスの使用方法を参照してください。

1. ターゲット・スタンバイ・データベースのスイッチオーバーの準備ができていることを確認します。

新しいスイッチオーバーの文にはVERIFYオプションがあり、スイッチオーバーに必要な多くの条件のチェックが実行されることとなります。チェックされる項目の一部は次のとおりです。Redo Applyがスイッチオーバー・ターゲットで実行しているか、スイッチオーバー・ターゲットのリリース・バージョンが12.1以降か、スイッチオーバー・ターゲットが同期しているか、および実行中のMRPがあるかどうか。

プライマリ・データベースのDB_UNIQUE_NAMEがBOSTONで、スイッチオーバー・ターゲット・スタンバイ・データベースのDB_UNIQUE_NAMEがCHICAGOだとします。プライマリ・データベースBOSTONで、次のSQL文を発行し、スイッチオーバー・ターゲットCHICAGOのスイッチオーバーの準備ができていることを確認します。

```
SQL> ALTER DATABASE SWITCHOVER TO CHICAGO VERIFY;  
ERROR at line 1:  
ORA-16470: Redo Apply is not running on switchover target
```

この操作が正常に実行されると、「データベースが変更されました。」というメッセージが返されますが、この例ではORA-16470エラーが返されます。このエラーは、スイッチオーバー・ターゲットCHICAGOのスイッチオーバーの準備ができていないことを意味します。スイッチオーバー操作の前にREDO Applyを開始する必要があります。

REDO Applyを開始した後で、次の文を再度発行します。

```
SQL> ALTER DATABASE SWITCHOVER TO CHICAGO VERIFY;  
ERROR at line 1:  
ORA-16475: succeeded with warnings, check alert log for more details
```

スイッチオーバー・ターゲットCHICAGOはスイッチオーバーの準備ができています。しかし、ORA-16475エラーで示された警告は、スイッチオーバーのパフォーマンスに影響を与える可能性があります。アラート・ログには次のようなメッセージが含まれます。

```
SWITCHOVER VERIFY WARNING: switchover target has dirty online redo logfiles that require  
clearing. It takes time to clear online redo logfiles. This may slow down switchover process.
```

問題を解決できる場合、またはスイッチオーバーのパフォーマンスが重要ではない場合は、これらの警告を無視できます。必要だと判断した解決を行った後で、次のSQL文を再度発行します。

```
SQL> ALTER DATABASE SWITCHOVER TO CHICAGO VERIFY;  
Database altered.
```

スイッチオーバー・ターゲットCHICAGOのスイッチオーバーの準備ができました。

2. 次のSQL文を発行して、プライマリ・データベースBOSTONでスイッチオーバーを開始します。

```
SQL> ALTER DATABASE SWITCHOVER TO CHICAGO;  
Database altered.
```

この文がエラーなしに終了した場合は、ステップ3に進みます。

エラーが発生したら、古いプライマリ・データベース(BOSTON)および古いスタンバイ・データベース(CHICAGO)をマウントします。両方のデータベースで、V\$DATABASEのDATABASE_ROLEを問い合わせます。BOSTONとCHICAGOのデータベース・ロールで考えられる組合せは3つあります。次の表で、その組合せを説明し、推定される原因と、それぞれの状況に対する高レベルの修正処置を示します。特定のエラー状況の詳細は、*Oracle Data Guard*概要および管理のOracle Data Guardのトラブルシューティングを参照してください。

V\$DATABASEの**DATABASE_ROLE列の値****原因と修正処置**

BOSTON データベースがプライマリ、
CHICAGO データベースがスタンバイ

原因: BOSTON データベースがスタンバイ・データベース・ロールへの変換に失敗しました。

処置: BOSTON のスタンバイ・ロールへの切替えを妨げているエラーの詳細についてアラート・ログを参照し、エラーを解決するために必要な処置をとり、必要であれば BOSTON のいずれかのノードを再度オープンし、スイッチオーバー・プロセスをステップ 1 から繰り返します。

BOSTON データベースがスタンバイ、
CHICAGO データベースもスタンバイ

原因: CHICAGO データベースがプライマリ・データベース・ロールへの変換に失敗しました。

処置: 次の SQL 文を発行して、BOSTON または CHICAGO のいずれかをプライマリ・データベースに変換します。

```
SQL> ALTER DATABASE SWITCHOVER TO target_db_name FORCE;
```

次に例を示します。

- CHICAGO データベースで、次の SQL 文を発行し、プライマリ・データベースに変換します。

```
ALTER DATABASE SWITCHOVER TO CHICAGO FORCE;
```

- BOSTON データベースで、次の SQL 文を発行し、プライマリ・データベースに変換します。

```
ALTER DATABASE SWITCHOVER TO BOSTON FORCE;
```

SQL 文が ORA-16473 エラーで失敗した場合、コマンドを再度発行する前に REDO Apply を開始する必要があります。

次のように REDO Apply を再開します。

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

次のように、スイッチオーバー・コマンドを再度発行します。

```
SQL> ALTER DATABASE SWITCHOVER TO BOSTON FORCE; Database  
altered.
```

BOSTON データベースがスタンバイ、
CHICAGO データベースがプライマリ

原因: BOSTON および CHICAGO データベースは新しいロールに正常に切り替わりましたが、最終的な成功のステータスの BOSTON への送信時にエラーが発生しました。

処置: ステップ 3 を続行してスイッチオーバー操作を終了します。

3. 次のSQL文を新しいプライマリ・データベースCHICAGOで発行して、開きます。

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

4. 次のSQL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BOSTONをマウントします:

```
SQL> STARTUP MOUNT;
```

または、BOSTONがOracle Active Data Guardのフィジカル・スタンバイ・データベースである場合、次のSQL文を発行して読み取り専用で開きます。

```
SQL> STARTUP;
```

5. 新しいフィジカル・スタンバイ・データベースでREDO Applyを開始します。次に例を示します。

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

関連トピック

- [Oracle Data Guardのトラブルシューティング](#)
- [遠隔同期インスタンスの使用](#)

9.2.2 フィジカル・スタンバイ・データベースへのフェイルオーバーの実行

このステップでは、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方法について説明します。

1. プライマリ・データベースをマウントできる場合、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、未送信のアーカイブされた現行のREDOをフラッシュします。この操作が正常に終了した場合、プライマリ・データベースがデータ消失ゼロのデータ保護モードで実行されていないとしても、データの消失がないようにフェイルオーバーすることができます。

まず、ターゲット・スタンバイ・データベースでREDO Applyがアクティブであることを確認します。次にプライマリ・データベースをマウントします。オープンはしないでください。プライマリ・データベースをマウントできない場合は、ステップ2に進みます。

まだ設定していない場合は、プライマリで構成されているリモートのLOG_ARCHIVE_DEST_*n*をターゲット宛先を指すように設定します。(ターゲット宛先に遠隔同期インスタンスによってサービスが提供されていない場合、またはターゲット宛先がカスケードされた構成のターミナル・スタンバイである場合は、リモートのLOG_ARCHIVE_DEST_*n*は構成されていない可能性があります。)また、NET_ALIAS_TARGET_DB_NAMEが有効であり、適切に確立されていることを確認して、プライマリがターゲット宛先に接続できることを確認してください。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_6='SERVICE=NET_ALIAS_TARGET_DB_NAME -  
> ASYNC VALID_FOR=(online_logfile, primary_role) -  
> DB_UNIQUE_NAME="target_db_unique_name"' SCOPE=memory;
```

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_6=ENABLE;
```

プライマリのLOG_ARCHIVE_CONFIG指定にターゲット宛先のDB_UNIQUE_NAMEが含まれている(およびターゲット宛先のLOG_ARCHIVE_CONFIGにプライマリのDB_UNIQUE_NAMEが含まれている)ことも前提となります。含まれていない場合は、必要に応じてプライマリおよびターゲット宛先でその情報をLOG_ARCHIVE_CONFIGに追加します。

次のSQL文をプライマリ・データベースで発行します。

```
SQL> ALTER SYSTEM FLUSH REDO TO target_db_name;
```

*target_db_name*には、プライマリ・データベースからフラッシュされたREDOを受信するスタンバイ・データベースのDB_UNIQUE_NAMEを指定します。

この文は、未送信のREDOをすべて、プライマリ・データベースからスタンバイ・データベースにフラッシュし、このREDOがスタンバイ・データベースに適用されるまで待機します。

この文がエラーなしで終了した場合は、ステップ5に進みます。この文でエラーが出た場合、または完了するまで待てないため、この文を停止しなければならない場合は、ステップ2に進みます。

2. ターゲット・スタンバイ・データベースでV\$ARCHIVED_LOGビューを問い合わせ、個々のREDOスレッドについて、最も大きなログ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#) -  
> OVER (PARTITION BY thread#) AS LAST from V$ARCHIVED_LOG;
```

THREAD	LAST
1	100

可能であれば、各プライマリ・データベースREDOスレッドに対する最新のアーカイブREDOログ・ファイルがスタンバイ・データベースになれば、これをコピーし、登録します。これは、REDO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';
```

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

次に例を示します。

```
SQL> SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE# FROM V$ARCHIVE_GAP;
```

THREAD#	LOW_SEQUENCE#	HIGH_SEQUENCE#
1	90	92

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

可能であれば、欠落しているすべてのアーカイブREDOログ・ファイルを、プライマリ・データベースからターゲット・スタンバイ・データベースにコピーして登録します。これは、REDO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';
```

4. ステップ3で実行する問合せでは、順序番号が最も大きいギャップに関する情報のみが表示されます。ギャップを解決した後、行が戻されなくなるまで問合せを繰り返します。

ステップ2からステップ4を実行した後、アーカイブREDOログ・ファイルのすべてのギャップを解決できない場合(障害の発生したプライマリ・データベースのホスト・システムへのアクセス権がない場合など)、フェイルオーバー時にデータが消失する可能性があります。

5. 次のSQL文をターゲット・スタンバイ・データベースで発行します。

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

6. 次のSQL文をターゲット・スタンバイ・データベースで発行します。

```
SQL> ALTER DATABASE FAILOVER TO target_db_name;
```

たとえば、ターゲット・スタンバイ・データベースの名前がCHICAGOだとします。

```
SQL> ALTER DATABASE FAILOVER TO CHICAGO;
```

この文がエラーなしに終了した場合は、ステップ10に進みます。

エラーがあればステップ7に進みます。

7. エラーが発生したら、エラーの原因の解決を試みてから、文を再発行します。

- 成功したらステップ10に進みます。
- エラーが引き続き発生し、遠隔同期インスタンスが関係する場合、ステップ8に進みます。
- エラーが引き続き発生し、遠隔同期インスタンスが関係しない場合、ステップ9に進みます。

8. このステップは、遠隔同期インスタンスのエラーの場合のみ使用します。エラーが遠隔同期インスタンスに関係し(使用できないなど)、問題の解決を試みて、成功しないまま文を再発行する場合、FORCEオプションを使用できます。次に例を示します。

```
SQL> ALTER DATABASE FAILOVER TO CHICAGO FORCE;
```

FORCEオプションは、フェイルオーバーに遠隔同期インスタンスとの対話時に発生したエラーを無視し、可能なかぎり、フェイルオーバーを続行するように指示します。(FORCEオプションはフェイルオーバー・ターゲットが遠隔同期インスタンスによりサービスされている場合にのみ意味を持ちます)。

FORCEオプションが成功したら、ステップ10に進みます。

FORCEオプションが成功しなかったら、ステップ9に進みます。

9. データ消失フェイルオーバーを実行します。

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

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

次の例では、フェイルオーバー操作がORA-16472エラーで失敗します。このエラーは、データベースが最大可用性または最大保護モードで構成されていますが、フェイルオーバー中にデータ消失が検出されたことを意味します。

```
SQL> ALTER DATABASE FAILOVER TO CHICAGO;  
ERROR at line 1:  
ORA-16472: failover failed due to data loss
```

次のSQL文を発行することで、データ消失フェイルオーバーを完了できます。

```
SQL> ALTER DATABASE ACTIVATE PHYSICAL STANDBY DATABASE;  
Database altered.
```

10. 新しいプライマリ・データベースをオープンします。

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

11. 新しいプライマリ・データベースの完全バックアップを実行することをお勧めします。

12. Data Guard構成のその他のフィジカル・スタンバイ・データベースのいずれかでREDO Applyが停止している場合は、これを再起動します。次に例を示します。

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

13. フェイルオーバーの完了後、元のプライマリ・データベースは[「フラッシュバック・データベースを使用した障害が発](#)

[生じたプライマリ・スタンバイ・データベースへの変換](#)または[RMANバックアップを使用した障害が発生したプライマリ・スタンバイ・データベースへの変換](#)で説明する方法に従って、新しいプライマリ・データベースのフィジカル・スタンバイ・データベースに変換できます。あるいは、[フィジカル・スタンバイ・データベースの作成ステップ](#)で説明した方法に従って、新しいプライマリ・データベースのバックアップからフィジカル・スタンバイ・データベースとして再作成できます。

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

9.3 ロジカル・スタンバイ・データベースが関与するロールの推移

ロールの推移ステップは、スイッチオーバーとフェイルオーバーのどちらを実行するかによって異なります。

ロジカル・スタンバイ・データベースに伴うスイッチオーバーおよびフェイルオーバーの実行方法の詳細は、次のトピックを参照してください。

- [ロジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)
- [ロジカル・スタンバイ・データベースへのフェイルオーバーの実行](#)

ノート:

ロジカル・スタンバイでは、データベース・サービスはレプリケートしません。ロジカル・スタンバイに対するフェイルオーバーまたはスイッチオーバーのイベントでは、プライマリでのサービスに接続している中間層が接続できなくなる(サービスの作成がレプリケートされないため)か、正しくないエディションに接続するようになります(サービス属性の変更がレプリケートされないため)。

Oracle Clusterware では、管理するサービスをロジカル・スタンバイにレプリケートしません。それらのプライマリとスタンバイ間での同期を手動で維持する必要があります。Oracle Clusterware の詳細は、『[Oracle Clusterware 管理およびデプロイメント・ガイド](#)』を参照してください。

9.3.1 ロジカル・スタンバイ・データベースへのスイッチオーバーの実行

プライマリ・データベースとロジカル・スタンバイ・データベースとの間でスイッチオーバーを実行する場合、スイッチオーバーは、常にプライマリ・データベースで開始し、ロジカル・スタンバイ・データベースで完了してください。

スイッチオーバーが成功するためには、次のステップを記載されているとおりの順序で実行する必要があります。

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

次に例を示します。

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

SWITCHOVER_STATUS
-----
TO STANDBY
1 row selected
```

SWITCHOVER_STATUS列の値TO STANDBYまたはSESSIONS ACTIVEは、プライマリ・データベースをロジカル・スタンバイ・ロールに切替え可能であることを示します。これらの値のいずれかが表示されない場合は、Oracle Data Guard構

成が正常に機能していることを確認してください(たとえば、LOG_ARCHIVE_DEST_nパラメータのすべての値が正しく指定されていることを確認します)。V\$DATABASEビューのSWITCHOVER_STATUS列に有効なその他の値の詳細は、『[Oracle Databaseリファレンス](#)』を参照してください。

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

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

この文は現行プライマリ・データベースに対して、間もなくロジカル・スタンバイ・ロールに切り替わり、新しいプライマリ・データベースからREDOデータを受信し始めることを通知します。LogMinerディクショナリを受信に備えてプライマリ・データベースでこのステップを実行し、ステップ3で説明したように、現行のロジカル・スタンバイ・データベースのREDOストリームに記録できるようにします。

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

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

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

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

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

4. プライマリ・データベースからロジカル・スタンバイ・ロールへのロール・トランジションを完了するには、プライマリ・データベースのV\$DATABASE固定ビューのSWITCHOVER_STATUS列を問い合わせることで、プライマリ・データベースがLogMinerディクショナリを受信したことを確認する必要があります。LogMinerディクショナリを受信しなければスイッチオーバーを続行できません。現行のプライマリ・データベースが以降のプライマリ・データベースから受信するREDOレコードを解釈できる必要があるためです。SWITCHOVER_STATUS列にスイッチオーバーの進捗が示されています。

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

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

```
SWITCHOVER_STATUS
-----
TO LOGICAL STANDBY
1 row selected
```

ノート:

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



- a. プライマリ・データベースでスイッチオーバーを取り消します。

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

- b. ロジカル・スタンバイ・データベースでスイッチオーバーを取り消します。

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

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

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

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

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

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

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

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

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

次に例を示します。

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

```
SWITCHOVER_STATUS
```

```
-----  
TO PRIMARY
```

```
1 row selected
```

V\$DATABASEビューのSWITCHOVER_STATUS列に有効なその他の値の詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

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

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

Oracle 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。[ロールの推移のターゲット・スタンバイ・データベースの選択](#)で説明したように、構成内の他のすべてのロジカル・スタンバイは新しいプライマリのスタンバイになりますが、フィジカル・スタンバイ・データベースは、元のプライマリ・データベースのスタンバイのままです。

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

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

9.3.2 ロジカル・スタンバイ・データベースへのフェイルオーバーの実行

ロジカル・スタンバイ・データベースが関与するフェイルオーバーによるロールの推移では、障害の発生したプライマリ・データベースとすべてのロジカル・スタンバイ・データベースで対処措置を実行する必要があります。

このトピックでは、ロジカル・スタンバイ・データベースに伴うフェイルオーバーの実行方法について説明します。障害の発生したプライマリ・データベースでフラッシュバック・データベースが有効化されていない場合は、現行のプライマリ・データベースから作成したバックアップを使用してデータベースを再作成する必要があります。それ以外の場合は、[「フラッシュバック・データベースを使用した障害の発生したプライマリのスタンバイ・データベースへの変換」](#)で説明する手順に従って、障害の発生したプライマリ・データベースを新しいプライマリ・データベースのロジカル・スタンバイ・データベースに変換できます。

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

1. プライマリ・データベースをマウントできる場合、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、未送信のアーカイブされた現行のREDOをフラッシュします。この操作が正常に終了した場合、プライマリ・データベースがデータ消失ゼロのデータ保護モードで実行されていない場合でも、データの消失がないようにフェイルオーバーすることができます。

まず、ターゲット・スタンバイ・データベースでREDO Applyがアクティブであることを確認します。次にプライマリ・データベースをマウントします。オープンはしないでください。

次のSQL文をプライマリ・データベースで発行します。

```
SQL> ALTER SYSTEM FLUSH REDO TO target_db_name;
```

target_db_nameには、プライマリ・データベースからフラッシュされたREDOを受信するスタンバイ・データベースのDB_UNIQUE_NAMEを指定します。

この文は、未送信のREDOをすべて、プライマリ・データベースからスタンバイ・データベースにフラッシュし、このREDOがスタンバイ・データベースに適用されるまで待機します。

2. 構成に含まれるコンポーネントの状態によっては、プライマリ・データベースのアーカイブREDOログ・ファイルにアクセスできる場合があります。その可能性がある場合は、次の手順に従います。
 - a. ロジカル・スタンバイ・データベースでアーカイブREDOログ・ファイルが欠落しているかどうかを判断します。
 - b. 欠落しているログ・ファイルを、プライマリ・データベースからロジカル・スタンバイ・データベースにコピーします。
 - c. コピーしたログ・ファイルを登録します。

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

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

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

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

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

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

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

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

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

FINISH APPLY句が指定されていない場合、現在のスタンバイREDOログ・ファイルからの適用されていないREDOログ・ファイルはスタンバイ・データベースがプライマリ・データベースになる前には適用されません。

5. [「フェイルオーバー後のロジカル・スタンバイ・データベースの構成」](#)で説明する方法に従って、既存のロジカル・スタンバイ・データベースが新しいプライマリ・データベースを引き続き保護できるようにします。
6. Oracle Data Guardデータベース・フェイルオーバー直後に、新しいプライマリ・データベースのバックアップを行います。データベースの完全なバックアップ・コピーがなければ、フェイルオーバー後の変更をリカバリすることができないので、バックアップをすぐに実行することは安全策として必要な作業です。
7. フェイルオーバーの完了後、元のプライマリ・データベースは[「フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換」](#)または[「RMANバックアップを使用した障害が発生したプライマリのスタンバイ・データベースへの変換」](#)で説明する方法に従って、新しいプライマリ・データベースのロジカル・スタンバイ・データベースに変換できます。あるいは、[「ロジカル・スタンバイ・データベースの作成」](#)で説明した方法に従って、新しいプライマリ・データベースのバックアップからロジカル・スタンバイ・データベースとして再作成できます。

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

9.4 ロールの推移後のフラッシュバック・データベースの使用

ロール・トランジションの後に、オプションでFLASHBACK DATABASEコマンドを使用してデータベースをロール・トランジション前のある時点、またはシステム変更番号(SCN)に戻すことができます。

プライマリ・データベースのフラッシュバックを行う場合、同じ(または以前の)SCNまたは時刻にすべてのスタンバイ・データベースをフラッシュバックする必要があります。この方法でプライマリまたはスタンバイ・データベースのフラッシュバックを行う場合は、過去のスイッチオーバーを気にする必要はありません。SCN/時刻が過去の任意の時点でのスイッチオーバーより前の場合、Oracleは自動的に過去のスイッチオーバーを越えてフラッシュバックを行うことができます。

ノート:



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

9.4.1 スイッチオーバー後のフラッシュバック・データベースの使用

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

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

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

9.4.2 フェイルオーバー後のフラッシュバック・データベースの使用

フラッシュバック・データベースを使用して、障害の発生したプライマリ・データベースをフェイルオーバー発生前の時点に変換してから、スタンバイ・データベースに変換できます。

詳細な手順は、[「フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換」](#)を参照してください。

10 フィジカルおよびスナップショット・スタンバイ・データベースの管理

この章では、フィジカルおよびスナップショット・スタンバイ・データベースの様々な管理方法について説明します。

次のトピックを参照してください。

- [フィジカル・スタンバイ・データベースの起動と停止](#)
- [フィジカル・スタンバイ・データベースのオープン](#)
- [フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更](#)
- [OPEN RESETLOGS文を使用したリカバリ](#)
- [プライマリ、フィジカル・スタンバイおよびスナップショット・スタンバイ・データベースの監視](#)
- [プライマリからスタンバイへのリストア・ポイントのレプリケート](#)
- [REDO Applyのチューニング](#)
- [SQLチューニング・アドバイザによるActive Data Guard環境でのデータベースのチューニング](#)
- [Oracle Diagnostic Packを使用したOracle Active Data Guardのスタンバイのチューニング](#)
- [スナップショット・スタンバイ・データベースの管理](#)

Oracle Data Guard Brokerによるフィジカルおよびスナップショット・スタンバイ・データベース管理の簡素化方法を学習するには、『[Oracle Data Guard Broker](#)』を参照してください。

10.1 フィジカル・スタンバイ・データベースの起動と停止

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

10.1.1 フィジカル・スタンバイ・データベースの起動

フィジカル・スタンバイ・データベースを起動するには、SQL*Plus STARTUPコマンドを使用します。

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

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

REDO Applyについては「[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](#)」および読取り専用モードでのフィジカル・スタンバイ・データベースのオープンについては、「[フィジカル・スタンバイ・データベースのオープン](#)」を参照してください。

ノート:



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

10.1.2 フィジカル・スタンバイ・データベースの停止

REDO Applyを停止してフィジカル・スタンバイ・データベースを停止するには、SQL*Plus SHUTDOWNコマンドを使用します。

停止処理が完了するまで、データベース停止を開始したセッションに制御が戻されません。

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

10.2 フィジカル・スタンバイ・データベースのオープン

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

ノート:



読取り専用モードで開かれたフィジカル・スタンバイ・データベースは、読取り専用モードで開かれたその他のOracle データベースと同じ制約を受けます。詳細は、[『Oracle Database 管理者ガイド』](#)を参照してください。

Oracle Active Data Guardオプションのライセンスを購入している場合は、フィジカル・スタンバイ・データベースを開いている間はRedo Applyをアクティブにすることができます。それにより、問合せによって、プライマリ・データベースからのものと同じ結果が得られます。この機能は、リアルタイム問合せ機能と呼ばれます。詳細は、[「リアルタイム問合せ」](#)を参照してください。

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

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

関連項目:

- Oracle Active Data Guardの詳細は、[『Oracle Databaseライセンス情報』](#)を参照してください。

10.2.1 リアルタイム問合せ

Oracle Active Data Guardオプションのリアルタイム問合せ機能を使用するためには、COMPATIBLEデータベース初期化パラメータを11.0以上に設定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・インスタンスは、そのデータベースのマウント済インスタンスでREDO Applyがアクティブである場合、オープンできません。次のSQL文を使用してREDO Applyを停止し、スタンバイ・インスタンスを読取り専用でオープンしてREDO Applyを再開します。

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

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

ノート:



REDO Apply がオープン・インスタンスでアクティブになっている場合は、REDO Apply を停止せずにその他のインスタンスをオープンすることができます。

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

例: スタンバイのオープン・モードを調べるためのV\$DATABASEの問合せ

この例は、フィジカル・スタンバイがリアルタイム問合せモードで開かれているときのV\$DATABASE. OPEN_MODE列の値の変化を示します。

1. フィジカル・スタンバイ・インスタンスを起動してオープンし、次のSQL問合せを実行して、データベースが読取り専用モードでオープンしていることを確認します。

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

```
OPEN_MODE
```

```
-----  
READ ONLY
```

2. 次のSQL文を発行してREDO Applyを開始します。

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

```
Database altered.
```

3. スタンバイがリアルタイム問合せモード(スタンバイが読取り専用モードでオープンし、REDO Applyがアクティブになっている)であるため、V\$DATABASE. OPEN_MODE列は次のように変化します。

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

```
OPEN_MODE
```

```
-----  
READ ONLY WITH APPLY
```

10.2.1.1 リアルタイム問合せ環境での適用ラグの監視

リアルタイム問合せを使用して、プライマリ・データベースからフィジカル・スタンバイ・データベースに問合せをオフロードしている場合、適用ラグを監視して許容範囲内かどうかを確認できます。

現行の適用ラグとは、最後に適用された変更がスタンバイで表示可能になってから、同じ変更がプライマリで最初に表示可能になるまでに経過した時間です。このメトリックは秒単位で計算されます。

適用ラグを取得するには、V\$DATAGUARD_STATSビューを問い合わせます。次に例を示します。

```
SQL> SELECT name, value, datum_time, time_computed FROM V$DATAGUARD_STATS -  
> WHERE name like 'apply lag';
```

NAME	VALUE	DATUM_TIME	TIME_COMPUTED
-----	-----	-----	-----
apply lag	+00 00:00:00	05/27/2009 08:54:16	05/27/2009 08:54:17

apply lagメトリックの計算には、プライマリ・データベースから定期的に受信されるデータが使用されます。DATUM_TIME列には、このデータがスタンバイ・データベースにより最後に受信されたときのタイムスタンプが格納されます。TIME_COMPUTED列には、apply lagメトリックが計算されたときに取得されたタイムスタンプが格納されます。これらの列の値の差は30秒未満である必要があります。差が30秒以上である場合、apply lagメトリックが正確ではない可能性があります。

スタンバイ・インスタンスが最後に起動されてからの適用ラグ値の履歴を示すヒストグラムを取得するには、V\$STANDBY_EVENT_HISTOGRAMビューを問い合わせます。次に例を示します。

```
SQL> SELECT * FROM V$STANDBY_EVENT_HISTOGRAM WHERE NAME = 'apply lag' -
> AND COUNT > 0;
```

NAME	TIME	UNIT	COUNT	LAST_TIME_UPDATED
apply lag	0	seconds	79681	06/18/2009 10:05:00
apply lag	1	seconds	1006	06/18/2009 10:03:56
apply lag	2	seconds	96	06/18/2009 09:51:06
apply lag	3	seconds	4	06/18/2009 04:12:32
apply lag	4	seconds	1	06/17/2009 11:43:51
apply lag	5	seconds	1	06/17/2009 11:43:52

6 rows selected

ある期間にわたってapply lagを評価するには、その期間の開始時のV\$STANDBY_EVENT_HISTOGRAMのスナップショットを取得し、その期間の終了時に取得したスナップショットと比較します。

10.2.1.2 リアルタイム問合せ環境での適用ラグ許容差の構成

リアルタイム問合せモードのフィジカル・スタンバイ・データベースへの非管理ユーザーからの問合せで、新しいSTANDBY_MAX_DATA_DELAYセッション・パラメータを使用して特定のセッションの適用ラグの許容値を指定できます。

この機能によって、スタンバイ・データベースが許容できないほど失効したかどうかを検出できるため、プライマリ・データベースからフィジカル・スタンバイ・データベースに問合せを安全に移動できます。

STANDBY_MAX_DATA_DELAYがデフォルト値NONEに設定されている場合、フィジカル・スタンバイ・データベースに発行された問合せは、そのデータベースの適用ラグに関係なく実行されます。

STANDBY_MAX_DATA_DELAYを0（ゼロ）以外の値に設定すると、フィジカル・スタンバイ・データベースに発行された問合せは、適用ラグがSTANDBY_MAX_DATA_DELAY以下の場合のみ実行されます。それ以外の場合は、ORA-3172エラーが返され、適用ラグが大きすぎることでクライアントに警告されます。

STANDBY_MAX_DATA_DELAYを0に設定すると、フィジカル・スタンバイ・データベースに発行される問合せは、スタンバイ・データベースがプライマリ・データベースから遅れていないかぎり、その問合せがプライマリ・データベースに発行された場合とまったく同じ結果を返すことが保証されます。遅れている場合、ORA-3172エラーが返されます。

ALTER SESSION SQL文を使用してSTANDBY_MAX_DATA_DELAYを設定します。次に例を示します。

```
SQL> ALTER SESSION SET STANDBY_MAX_DATA_DELAY=2
```

10.2.1.3 リアルタイム問合せ環境でのREDO Applyの強制同期

プライマリ・データベースから受信したすべてのREDOデータがフィジカル・スタンバイ・データベースに適用されたことを確認するには、SQL ALTER SESSION文を使用します。

次のSQL文を発行します。

```
SQL> ALTER SESSION SYNC WITH PRIMARY;
```

このコマンドが発行された時点でスタンバイ・データベースが受け取ったすべてのREDOデータが、フィジカル・スタンバイ・データベースに適用されるまで、この文によってその他の処理がブロックされます。スタンバイ・データベースのREDO転送ステータスがSYNCHRONIZEDでない場合、またはREDO Applyがアクティブでない場合は、ORA-3173エラーがすぐに返され、同期は行われません。

特定のケースでREDO Apply同期を確実に行うためには、SYS_CONTEXT('USERENV', 'DATABASE_ROLE') ファンクションを使用して、スタンバイ専用トリガー(プライマリで有効化されるが、スタンバイで実行される場合のみ特定の処理を実行するトリガー)を作成します。たとえば、特定のユーザー接続に対してログオン時にALTER SESSION SYNC WITH PRIMARY文を実行する次のトリガーを作成することができます。

```
CREATE TRIGGER adg_logon_sync_trigger
AFTER LOGON ON user.schema
begin
  if (SYS_CONTEXT('USERENV', 'DATABASE_ROLE') IN ('PHYSICAL STANDBY')) then
    execute immediate 'alter session sync with primary';
  end if;
end;
```

10.2.1.4 リアルタイム問合せの制限

このリストでは、リアルタイム問合せモードに関連する制限事項について説明しています。

- これまでに説明した、適用ラグの制御およびREDO Apply同期のメカニズムでは、クライアントが接続していることと、リアルタイム問合せモードのフィジカル・スタンバイ・データベースに問合せを発行することが必要です。
- STANDBY_MAX_DATA_DELAYが0に設定されている場合、またはALTER SESSION SYNC WITH PRIMARY SQL文が使用される場合には、この他に次の制限も適用されます。
 - スタンバイ・データベースがSYNC転送でREDOデータを受け取ります。
 - スタンバイ・データベースのREDO転送ステータスはSYNCHRONIZEDである必要があります。また、プライマリ・データベースは、最大保護モードまたは最大可用性モードのいずれかで実行されている必要があります。
 - リアルタイム適用を有効化しておく必要があります。
- Oracle Active Data Guardは、キャッシュ・フュージョンの使用を通じてOracle RAC環境でパフォーマンスの高いリアルタイム問合せを実現しています。これにより、Oracle Data Guardの適用インスタンスおよび問合せは、キャッシュで動作することが可能となり、ディスクI/O制限によって低速化することがなくなります。

この結果として、適用インスタンスの予期しない障害が発生すると、Oracle RACのすべてのオープン・インスタンス全体でバッファが一貫性のない状態となります。データベースが他のインスタンスでオープンしている場合、いずれかのオープン・インスタンスがActive Data Guardインスタンス・リカバリを実行してデータベースを一貫性のある状態にし、すべてのオープン・インスタンスがオープンしたままになります。

関連項目:

- ブローカによる適用インスタンス障害の処理方法の詳細は、『[Oracle Data Guard Broker](#)』を参照してください。
- Oracle Active Data GuardのOracle RACスタンバイにおける適用インスタンス障害の詳細は、<http://support.oracle.com>にあるMy Oracle Supportノート1357597.1を参照してください。

10.2.1.5 自動ブロック・メディア・リカバリ

データベースのアクセス時に破損データ・ブロックが検出されると、それらのブロックの破損していないコピーで自動的に置き換えることができます。

これには次の条件があります。

- フィジカル・スタンバイ・データベースがリアルタイム問合せモードで動作している必要があります(Oracle Active Data Guardライセンスが必要)。
- フィジカル・スタンバイ・データベースがリアルタイム適用を実行している必要があります。

自動ブロック・メディア・リカバリは、破損ブロックが検出されたのがプライマリかスタンバイかにより、2つの方向で動作します。

プライマリでの破損ブロック

破損データ・ブロックがプライマリ・データベースで検出されると、プライマリはそれらのブロックの破損していないコピーをスタンバイで自動的に検索し、見つければ、プライマリに送信します。

プライマリには、スタンバイのみ(フィジカル・スタンバイ、カスケード・フィジカル・スタンバイ、または遠隔同期インスタンス)へのLOG_ARCHIVE_DEST_#が必要です。プライマリには、ターミナル宛先へのLOG_ARCHIVE_DEST_#は必要ありません。それらのサービス名は自動的に確認できます。

スタンバイでの破損ブロック

破損データ・ブロックがスタンバイで検出されると、スタンバイは自動的にプライマリとの通信を起動し、それらのブロックの破損していないコピーをリクエストします。破損していないブロックをスタンバイに送信できるプライマリでは、次のデータベース初期化パラメータがスタンバイに構成されている必要があります。これは、プライマリが直接スタンバイにサービスを提供していない場合(カスケード構成など)でも適用されます。

- LOG_ARCHIVE_CONFIGパラメータはDG_CONFIGリストを使用して構成され、LOG_ARCHIVE_DEST_#パラメータはプライマリ・データベース用に構成されている。

または

- FAL_SERVERパラメータが構成され、その値にプライマリ・データベース用のOracle Netサービス名が含まれている。

自動ブロック・メディア修復の追加の考慮事項

- 自動修復は、Oracle Data Guardのどの保護モードでもサポートされます。ただし、スタンバイの破損していないバージョンのブロックを使用してプライマリの破損ブロックを修復する場合の効果は、プライマリによって生成されるREDOとスタンバイ適用がどの程度密接に同期化されているかによって変化します。
- 自動ブロック修復が実行されると、データベースのアラート・ログにメッセージが書き込まれます。
- 自動ブロック修復が可能でない場合は、ORA-1578エラーが返されます。

10.2.1.6 手動ブロック・メディア・リカバリ

破損データ・ブロックを手動で修復するには、RMAN RECOVER BLOCKコマンドを使用します。

このコマンドは、データ・ブロックの破損していないコピーをいくつかの場所で探します。デフォルトの場所の1つは、リアルタイム問合せモードで実行しているいずれかのフィジカル・スタンバイ・データベースです。RMAN RECOVER BLOCKコマンドのEXCLUDE STANDBYオプションを使用すると、置換ブロックのコピー元からフィジカル・スタンバイ・データベースを除外することもできます。

関連項目:

RMANのRECOVER BLOCKコマンドの詳細は、[『Oracle Databaseバックアップおよびリカバリ・リファレンス』](#)を参照してください。

10.2.1.7 フィジカル・スタンバイ・データベースでの問合せのチューニング

フィジカル・スタンバイ・データベースに対する問合せをチューニングしてパフォーマンスを最適化できます。

問合せの調整については、次の場所からOracle Maximum Availability Architecture (MAA)のホームページにある *Active Data Guard* のベスト・プラクティスに関する技術概要を参照してください。

<http://www.oracle.com/goto/maa>

強制全データベース・キャッシュ・モード

全データベース・キャッシュ・モードの強制の使用により、問合せがより高速に実行されるため、パフォーマンスが向上する可能性があります。

全データベース・キャッシュ・モードの強制の有効化および無効化はREDOに記録されないため、インメモリ・キャッシュのステータスはData Guard構成のすべてのメンバーで同じである必要はありません。

全データベース・インメモリ・キャッシュの強制機能の詳細は、『[Oracle Databaseパフォーマンス・チューニング・ガイド](#)』を参照してください。

10.2.1.8 フィジカル・スタンバイへの一時ファイルの追加

スタンバイを使用してプライマリ・データベースから問合せをオフロードする場合、少なくとも1つの一時データ・ファイルを持つ少なくとも1つの一時表領域でスタンバイを構成する必要があります。

ワークロードの性質上、スタンバイの初回作成時に自動的に作成される一時表領域よりも多くの領域が必要になる場合は、追加領域を手動で追加する必要があります。

一時ファイルをフィジカル・スタンバイ・データベースに追加するには、次のタスクを実行します。

1. 一時ファイルが格納される表領域を特定します。スタンバイ・データベースで次のコマンドを入力すると、特定できます。

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

TABLESPACE_NAME
-----
TEMP1
TEMP2
```

2. 前述の問合せで特定された表領域ごとに、新しい一時ファイルをスタンバイ・データベースに追加します。次の例では、TEMP1という新しい一時ファイルを追加します。このファイルは、プライマリ・データベースの一時ファイルと一致するサイズおよび再利用特性を持っています。

```
SQL> ALTER TABLESPACE TEMP1 ADD TEMPFILE
2> '/arch1/boston/temp01.dbf'
3> SIZE 40M REUSE;
```

10.2.2 Active Data GuardスタンバイでのSQLおよびPL/SQLの使用

Oracle Databaseリリース19c以降では、Active Data Guardスタンバイ・データベースでSQLおよびPL/SQL操作を実行できます。

関連トピック

- [Active Data Guardスタンバイ・データベースでのDML操作の実行](#)

- [Active Data Guardスタンバイ・データベースでの最上位のPL/SQL操作の実行](#)
- [変更されたPL/SQLオブジェクトの自動再コンパイル](#)

10.2.2.1 Active Data Guardスタンバイ・データベースでのDML操作の実行

Active Data Guardスタンバイ・データベースでDML操作を実行できます。これにより、スタンバイ・データベースで、DMLを実行する可能性のあるread-mostlyアプリケーションを実行できます。

スタンバイでのDML操作は、透過的にプライマリ・データベースにリダイレクトして実行できます。これには、PL/SQLブロックの一部であるDML文が含まれます。対応する変更がActive Data Guardスタンバイに送信され、適用されるまで、Active Data Guardセッションは待機します。DML操作中は読取り一貫性が維持され、DMLが実行されるスタンバイ・データベースでは、そのコミットされていない変更を表示できます。ただし、その他すべてのスタンバイ・データベース・インスタンスでは、トランザクションがコミットされた後のみ、これらの変更を表示できます。

ノート:



Active Data Guard スタンバイ・データベースで実行する DML 操作の数が多すぎないようにしてください。この操作は実際にはプライマリ上で実行されるため、DML が多すぎると、プライマリのパフォーマンスに影響を及ぼす可能性があります。

ノート:



Oracle XA トランザクションでの DML 操作は、Active Data Guard スタンバイ・データベースではサポートされていません。

プライマリへのDML操作の自動リダイレクトは、システム・レベルまたはセッション・レベルで構成できます。セッション・レベルの設定は、システム・レベルの設定をオーバーライドします。

Active Data Guard環境ですべてのスタンバイ・セッションに対するDML操作の自動リダイレクトを構成するには:

- ADG_REDIRECT_DML初期化パラメータをTRUEに設定します。

現在のセッションに対するDML操作の自動リダイレクトを構成するには、次のコマンドを使用します。

- ALTER SESSION ENABLE ADG_REDIRECT_DML;

例10-1 フィジカル・スタンバイ・データベースでのDML操作の実行

Active Data Guard設定のフィジカル・スタンバイ・データベースには、employeesという名前の表が含まれます。Active Data Guard環境のフィジカル・スタンバイ・データベースでDMLを実行して、この表に行を挿入できます。

スタンバイ・データベースで、現在のセッションに対してDMLリダイレクトを有効にします。

```
SQL> ALTER SESSION ENABLE ADG_REDIRECT_DML;
```

次のコマンドを使用して、employees表に行を追加します。

```
SQL> INSERT INTO employees VALUES (.....);
```

この時点で、変更されたデータはコマンドが実行されたスタンバイ・データベースによってのみ認識されます。挿入操作がプライマリ・データベースでコミットされると、変更は、すべてのスタンバイ・データベースに送信されて適用されます。

10.2.2.2 Active Data Guardスタンバイ・データベースでの最上位のPL/SQL操作の実行

Active Data Guardスタンバイ・データベースで実行する最上位のPL/SQLブロックは、バインド変数が含まれていない場合、プライマリ・データベースにリダイレクトして実行できます。

スタンバイで実行される最上位のPL/SQL操作をプライマリにリダイレクトするには、スタンバイ・データベースで、次のコマンドを使用して、自動リダイレクトを構成します。

- ALTER SESSION ENABLE ADG_REDIRECT_PLSQL;

最上位のPL/SQL操作の自動リダイレクトは、セッション・レベルでのみ構成できます。

10.2.2.3 変更されたPL/SQLオブジェクトの自動再コンパイル

スタンバイ・インスタンスで実行されるPL/SQLオブジェクトは、無効になっている場合、再コンパイルできます。

PL/SQLオブジェクトは、依存オブジェクトが変更または削除されると無効になります。Oracle Databaseリリース19c以降では、ADG_REDIRECT_DML初期化パラメータをTRUEに設定することで、スタンバイ・データベースで実行される、無効化されたPL/SQLオブジェクトを自動的に再コンパイルできます。これらのPL/SQLオブジェクトに対応するDDLは、プライマリ・データベースにリダイレクトされ、実行されます。スタンバイ・セッションは、操作が完了するまで待機します。

例10-2 変更されたPL/SQLオブジェクトの自動再コンパイル

Active Data Guard環境のプライマリ・データベースで、insert_emp1という名前のプロシージャが作成されます。このプロシージャは、employees表を更新するために使用されます。

```
CREATE OR REPLACE PROCEDURE update_sal (emp_id IN NUMBER, sal IN NUMBER)
AS BEGIN
    UPDATE employees SET salary=sal WHERE employee_id=emp_id);
END;
```

その後、employees表の構造は、ALTER TABLEコマンドを使用して変更されます。これにより、update_salプロシージャが無効化されます。

Active Data Guard環境のスタンバイ・データベースで、update_salプロシージャを使用し、従業員の給与を更新する必要があります。フィジカル・スタンバイで次のコマンドを実行します。

```
SQL> ALTER SESSION ENABLE ADG_REDIRECT_DML;
SQL> exec update_sal (105, 6000);
```

これらのコマンドを実行するユーザーには、update_salプロシージャを実行する権限が付与されています。

10.2.3 Active Data Guardインスタンスでの一時表の使用

Active Data Guardインスタンスでは、グローバル一時表とプライベート一時表の両方を作成できます。

一時表には、トランザクションまたはセッションの期間中にのみ存在するデータが保持されます。一時表内のデータはセッション専用です。各セッションで参照および変更できるのは、そのセッション自体のデータのみです。

関連項目:

- グローバル一時表、プライベート一時表およびそれらの相違点の詳細は、[Oracle Database概要](#)を参照してください。

10.2.3.1 Active Data Guardインスタンス上のグローバル一時表

Oracle Active Data Guardインスタンス上のグローバル一時表では、DMLおよびDDL操作が可能です。

DML操作

DML操作によってグローバル一時表が変更された場合、変更自体は一時表のみであるため、REDOは生成されません。しかし、変更に対して生成されたUNDOは次にREDOを生成します。読み取り専用データベース(Active Data Guardスタンバイなど)でのREDO生成は許可されません。しかし、一時UNDO機能によってグローバル一時表への変更に対するUNDOをUNDO表領域ではなく一時表領域に格納できるため、Oracle Active Data Guardスタンバイ上のグローバル一時表でのDML操作は可能です。また、一時表領域に格納されたUNDOはREDOを生成しません。

この機能は次のようにOracle Data Guardに利点をもたらします。

- 一時データの格納にグローバル一時表を使用するread-mostlyレポート・アプリケーションは、Oracle Active Data Guardインスタンスに負荷を軽減させることができます。
- 一時UNDOがプライマリ・データベースで有効化されていると、グローバル一時表への変更のUNDOはREDOに記録されないため、プライマリ・データベースが生成するREDOはより少なくなります。そのため、Oracle Data Guardがスタンバイに送信する必要があるREDOの量も減少するので、ネットワーク帯域幅と記憶域の使用量が減少します。

プライマリ・データベースで一時UNDOを有効化するには、TEMP_UNDO_ENABLED初期化パラメータを使用します。Oracle Active Data Guardスタンバイでは、一時UNDOはデフォルトで常に有効化されているため、TEMP_UNDO_ENABLEDパラメータは何も影響しません。

制限事項

- 一時UNDO機能では、データベース初期化パラメータCOMPATIBLEを12.0.0以降に設定する必要があります。
- Oracle Active Data Guardインスタンスでの一時UNDO機能は一時BL0Bまたは一時CL0Bをサポートしません。
- ローカル・オブジェクトに対する変更を伴っている場合、Oracle Active Data Guardインスタンス上の分散トランザクションは許可されません。たとえば、Oracle Active Data Guardインスタンスでグローバル一時表を変更し、かつデータベース・リンクを使用して他のデータベース上でリモート表を更新するようなトランザクションは、コミットできません。Active Data Guardインスタンス上のグローバル一時表に対する未処理のDML操作をコミットまたはロールバックしてから、リモートDML操作を発行する、あるいはその逆順を行う必要があります。EXPLAIN PLAN文などの操作によるグローバル一時表への暗黙的な書込みも、これに該当します。

関連項目:

- 一時表UNDOの詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。
- TEMP_UNDO_ENABLED初期化パラメータの詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

DDL操作

Active Data Guardスタンバイ・データベースでは、グローバル一時表を作成および削除できます。これらの操作のDDLは、プライマリ・データベースに透過的にリダイレクトされます。対応する変更がActive Data Guardスタンバイに送信され、適用されるまで、Active Data Guardセッションは待機します。グローバル一時表の作成例を次に示します。

```
SQL> CREATE GLOBAL TEMPORARY TABLE tab2(c1 number, c2 varchar(10)) ON COMMIT PRESERVE ROWS;
```

```
Table created.  
SQL>
```

ノート:



DDL リダイレクションが成功するには、Active Data Guard スタンバイ・データベースで、リアルタイム適用オプションを使用して管理スタンバイ・リカバリを開始し、かつ、Active Data Guard スタンバイ・データベースがプライマリ・データベースと同期している必要があります。

ノート:



グローバル一時表でのデータ定義言語(DDL)操作(CREATE や DROP など)は引き続きプライマリ・データベースから発行できます。DDL 変更はプライマリ・データベースでキャッチアップするとスタンバイで表示可能です。

10.2.3.2 Active Data Guardインスタンス上のプライベート一時表

Oracle Active Data Guardインスタンスでは、読取り専用であっても、プライベート一時表を作成できます。

読取り専用データベースでプライベート一時表を作成できる理由は、表のメタデータがディスク上ではなくメモリー上に格納されるためです。プライベート一時表の存続期間は表が作成されたセッション中のみであり、表はセッションの終了時に自動的に削除されます。この機能により、Active Data Guardスタンバイ・データベースに対してレポート・アプリケーションを実行できます。

ノート:



- [Oracle Database 管理者ガイド](#)(プライベート一時表の作成の詳細)

10.2.4 Active Data Guardスタンバイ環境でのIM列ストア

Oracle Database 12cリリース2 (12.2.0.1)以降、Active Data Guard (ADG)環境のスタンバイ・データベースで Oracle Databaseインメモリー列ストア(IM列ストア)がサポートされます。

Active Data Guardスタンバイ・データベースで実行するレポート作成のワークロードでは、IM列ストアを使用できます。IM列ストアを使用すると、圧縮列形式(インメモリー)のデータへのアクセスを最大限に利用できるため、ワークロードの実行パフォーマンスが向上します。さらに、プライマリ・データベースとスタンバイ・データベースでIM列ストアにまったく異なるデータ・セットを移入でき、アプリケーションで使用できるIM列ストアのサイズを事実上2倍にすることができます。複数インスタンスのREDO ApplyでIM列ストアのサポートを有効にするには、初期化パラメータenable_imc_with_miraをTRUEに設定します。

次の制約に注意してください:

- インメモリー式はプライマリ・データベースで実行された問合せのみに基づいて取得されます。
- アクセス条件に基づいたインメモリーの情報ライフサイクル管理(ILM)ポリシーは、プライマリ・データベースに記録されたアクセスのみに基づいてトリガーされます。

ノート:



- インメモリーの FastStart は、ADG 環境のスタンバイ・データベースでサポートされていません。
- インメモリーの Join-Groups は、ADG 環境のスタンバイ・データベースでサポートされていません。

関連トピック

- [『Oracle Database In-Memoryガイド』](#)
- [Oracle Databaseリファレンス](#)

10.2.5 Active Data Guard環境でのインメモリー外部表

Oracle Active Data Guardは、インメモリー外部表をサポートしています。

インメモリー外部表は、プライマリ・データベースとスタンバイ・データベースの両方で平行にロードされます。Active Data Guard環境で実行される問合せでは、平行問合せでインメモリー外部表セグメントを使用します。INMEMORYまたはNO INMEMORYを使用すると、アクティブなスタンバイ・データベースで、インメモリー外部表セグメントが解放されます。

ノート:



IM 列ストアに INMEMORY オブジェクトを移入するとき、INMEMORY... DISTRIBUTE 句の FOR SERVICE 副句を使用することは、プライマリ・データベース、スタンバイ・データベース、およびその両方についてサポートされていません。

関連項目:

[『Oracle Database In-Memoryガイド』](#)

10.2.6 Oracle Active Data Guardでの順序の使用

Oracle Active Data Guard環境では、デフォルトのCACHEオプションおよびNOORDERオプションでプライマリ・データベースによって作成された順序には、スタンバイ・データベースからもアクセスできます。

スタンバイ・データベースはそのような順序に最初にアクセスするとき、プライマリ・データベースに順序番号の範囲を割り当てることを要求します。範囲はキャッシュ・サイズと、順序が作成されたときに指定したその他の順序プロパティに基づきます。次に、データ・ディクショナリ内の対応する順序エントリを調整することで、プライマリ・データベースがそれらの順序番号を要求しているスタンバイ・データベースに割り当てます。スタンバイが範囲内のすべての番号を使用したら、別の番号の範囲を要求します。

プライマリ・データベースは、スタンバイ・データベースからの範囲の要求ごとに、プライマリ・データベースとスタンバイ・データベースの両方に以前に割り当てられたものと重複しないような順序番号の範囲が確実に取得されるようにします。これでOracle Data Guard構成全体にわたって一意の順序番号のストリームが生成されます。

順序の範囲に対するスタンバイの要求にはプライマリへのラウンドトリップが含まれるため、Oracle Active Data Guardスタンバイで使用される順序を作成するときは、CACHEキーワードに十分大きい値を指定するようにしてください。そうしないと、パフォーマンスが影響を受ける可能性があります。

また、ターミナル・スタンバイには、プライマリを指すように定義されたLOG_ARCHIVE_DEST_nパラメータが含まれることを確認してください。

例：複数スタンバイ構成での順序値の範囲の割当て

この例は、最初、または以前に割り当てられた順序値のすべてを使い果してしまった後で順序のNEXTVALを参照する場合に、データベースに割り当てることのできる順序値の範囲を示します。この例では、2つのスタンバイ・データベースがあります。

1. プライマリ・データベースで、次のSQL文を発行し、gttという名前の一時表とgという名前でキャッシュ・サイズが10の順序を作成します。

```
SQL> CREATE GLOBAL TEMPORARY TABLE gtt (a int);
```

```
Table created.
```

```
SQL> CREATE SEQUENCE g CACHE 10;
```

```
Sequence created.
```

2. 最初のスタンバイ・データベースで、次のSQL文を発行します。

```
SQL> INSERT INTO gtt VALUES (g.NEXTVAL);
```

```
1 row created.
```

```
SQL> INSERT INTO gtt VALUES (g.NEXTVAL);
```

```
1 row created.
```

```
SQL> SELECT * FROM gtt;
```

A
1
2

順序キャッシュ・サイズが10に設定され(ステップ1)、これが順序がアクセスされる最初なので、SELECT文の結果は、最初のスタンバイ・データベースが順序番号1から10に割り当てられることを示します。

3. プライマリ・データベースで、次のSQL文を発行します。

```
SQL> SELECT g.NEXTVAL FROM dual;
```

NEXTVAL
11

```
SQL> SELECT g.NEXTVAL FROM dual;
```

NEXTVAL
12

SELECT文の結果は、プライマリ・データベースが次の順序値の範囲、11から20に割り当てられることを示します。

4. 2番目のスタンバイ・データベースで、次のSQL文を発行します。

```
SQL> INSERT INTO gtt VALUES (g.NEXTVAL);
```

```
1 row created.
```

```
SQL> INSERT INTO gtt VALUES (g.NEXTVAL);
```



```
1 row created.
```

```
SQL> SELECT * FROM gtt;
```

A
21
22

SELECT文の結果は、2番目のスタンバイが次の順序値の範囲、21から30に割り当てられることを示します。



ノート:

ORDER または NOCACHE オプションを使用して作成した順序は、Oracle Active Data Guard スタンバイからアクセスできません。

10.2.6.1 セッション順序

セッション順序はセッションの可視性があるグローバル一時表で 사용되는ように特別に設計された特別なタイプの順序です。

既存の通常の順序(対比のために「グローバルな」順序と呼びます)とは異なり、セッションの順序は、複数のセッション全体ではなく、1つのセッションに限定された順序番号の一意の範囲を返します。また、セッションの順序は永続されない点も異なります。セッションが失われると、そのセッション中にアクセスされるセッションの順序も失われます。

セッション順序は、順序が定義されるときに指定する順序プロパティのほとんどをサポートします。しかし、CACHE/NOCACHEおよびORDER/NOORDERオプションはセッション順序には関係なく、無視されます。

セッション順序は読取り/書込みデータベースにより作成される必要がありますが、読取り/書込みまたは読取り専用データベース(通常のデータベースを読取り専用で一時的にオープンするか、またはスタンバイ・データベース)でアクセスできます。

セッション順序の作成および変更

セッション順序を作成するには、次のSQL文を発行します。

```
SQL> CREATE SEQUENCE ... SESSION;
```

既存のセッション順序を通常の(グローバル)順序に変更するには、次のSQL文を発行します。

```
SQL> ALTER SEQUENCE ... GLOBAL;
```

通常の順序をセッション順序に変更するには、次のSQL文を発行します。

```
SQL> ALTER SEQUENCE ... SESSION;
```

例: セッション順序の使用

この例では、セッション順序値がデータベース・セッションごとにどのように一意になるかを示しています。

1. プライマリ・データベースで、次のSQL文を発行し、gttという名前の一時表とsという名前のセッション順序を作成します。

```
SQL> CREATE GLOBAL TEMPORARY TABLE gtt (a int);
```

```
Table created.
```

```
SQL> CREATE SEQUENCE s SESSION;
```

```
Sequence created.
```

2. スタンバイ・データベースで、次のSQL文を発行します。

```
SQL> INSERT INTO gtt VALUES (s.NEXTVAL);

1 row created.

SQL> INSERT INTO gtt VALUES (s.NEXTVAL);

1 row created.

SQL> SELECT * FROM gtt;

      A
-----
      1
      2
```

3. 同じスタンバイ・データベースの別のセッションで、次のSQL文を発行します。

```
SQL> INSERT INTO gtt VALUES (s.NEXTVAL);

1 row created.

SQL> INSERT INTO gtt VALUES (s.NEXTVAL);

1 row created.

SQL> SELECT * FROM gtt;

      A
-----
      1
      2
```

SELECT文の結果は、割り当てされた順序値が前のステップの最初のセッションで割り当てられものと同じであることを示します。これは、順序値がデータベース・セッションごとに一意になる理由を示しています。

10.3 フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更

プライマリ・データベースに対して行われる構造変更のほとんどは、REDOデータによってフィジカル・スタンバイ・データベースに自動的に伝播されますが、手動操作が必要な変更がいくつかあります。

次の表に、フィジカル・スタンバイ・データベースで手動操作が必要な、プライマリ・データベースの構造および構成の変更を示します。

表10-1 フィジカル・スタンバイでの手動操作が必要なプライマリ・データベースの変更

プライマリ・データベースの変更	フィジカル・スタンバイ・データベースで必要なアクション
データファイルの追加または表領域の作成	STANDBY_FILE_MANAGEMENT データベース初期化パラメータが AUTO に設定されている場合、アクションは必要ない。このパラメータが MANUAL に設定されている場合は、新しいデータファイルをフィ

プライマリ・データベースの変更	フィジカル・スタンバイ・データベースで必要なアクション ジカル・スタンバイ・データベースにコピーする必要があります。
表領域の削除とデータファイルの削除	DROP または DELETE コマンドを含む REDO データがフィジカル・スタンバイに適用された後に、プライマリ・データベースとフィジカル・スタンバイ・データベースからデータファイルを削除します。
フィジカル・スタンバイ・データベースでのトランスポータブル表領域の使用	プライマリ・データベースとフィジカル・スタンバイ・データベースの間で表領域を移動する。
プライマリ・データベースのデータファイルの改名	フィジカル・スタンバイ・データベースでデータファイル名を変更します。
REDO ログ・ファイル・グループの追加または削除	フィジカル・スタンバイ・データベースで REDO ログおよびスタンバイ REDO ログの構成を評価し、必要に応じて調整する。
NOLOGGING またはリカバリ不能な操作	RMAN コマンド RECOVER ... NONLOGGED BLOCK を使用して、スタンバイの無効なブロックをプライマリの変更済ブロックに置き換えます。
パスワード・ファイルのリフレッシュ	Oracle Database 12c リリース 2 (12.2.0.1)以降、プライマリ・データベースで行われたパスワード・ファイルの変更は、スタンバイ・データベースに自動的に伝播されます。唯一の例外は、遠隔同期インスタンスです。遠隔同期インスタンスは REDO を受け取りますが、適用しないため、更新したパスワード・ファイルを引き続き手動で遠隔同期インスタンスにコピーする必要があります。遠隔同期インスタンスでパスワード・ファイルが更新された後、プライマリでのパスワードの更新を含む REDO は、遠隔同期インスタンスから REDO ログを受け取るように設定されているすべてのスタンバイ・データベースに自動的に伝播されます。REDO が適用されると、スタンバイでパスワード・ファイルが更新されます。
TDE マスター暗号化キーのリセット	フィジカル・スタンバイ・データベースのデータベース暗号化ウォレットをプライマリ・データベースのデータベース暗号化ウォレットの最新コピーで置き換える。
初期化パラメータ	フィジカル・スタンバイ・データベースで対応する変更を初期化パラメータに加える必要があるかどうかを評価する。

10.3.1 データファイルの追加または表領域の作成

STANDBY_FILE_MANAGEMENTデータベース初期化パラメータは、プライマリ・データベースへのデータ・ファイルの追加がフィジカル・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します。

- フィジカル・スタンバイ・データベースのSTANDBY_FILE_MANAGEMENTデータベース・パラメータがAUTOに設定されている場合、プライマリ・データベースに作成された新しいデータファイルはフィジカル・スタンバイ・データベースに自動的に作成されます。
- フィジカル・スタンバイ・データベースのSTANDBY_FILE_MANAGEMENTデータベース・パラメータがMANUALに設定されている場合、新しいデータファイルがプライマリ・データベースに追加されると、そのファイルは、プライマリ・データベースからフィジカル・スタンバイ・データベースに手動でコピーする必要があります。

ノート:

Oracle Active Data Guard オプションが有効になっているフィジカル・スタンバイでは、手動でのコピーの方法は使用できません。かわりに、スタンバイで次の SQL 文を実行して、空のデータファイルを作成する必要があります。



```
SQL> ALTER DATABASE CREATE DATAFILE [filename | filenumber] -
AS [NEW | new_filename];
```

名前を変更するもの(filename または filenumber)を指定する必要があります。

また、新しいファイル名または NEW のいずれかを指定します。NEW キーワードにより、Oracle Managed Files (OMF)が有効な場合、自動的に名前が選択されます。

別のデータベースの既存のデータ・ファイルがプライマリ・データベースにコピーされた場合、同様にそのファイルをスタンバイ・データベースにコピーする必要があり、STANDBY_FILE_MANAGEMENTパラメータの設定に関係なくスタンバイ制御ファイルを再作成する必要があります。

10.3.2 表領域の削除とデータファイルの削除

プライマリ・データベースから表領域またはデータファイルが削除されると、対応するデータファイルがフィジカル・スタンバイ・データベースから削除される必要があります。

次の例に、表領域を削除する方法を示します。

```
SQL> DROP TABLESPACE tbs_4;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

削除されたデータファイルがデータベースに属していないことを確認するには、V\$DATAFILEビューを問い合わせます。

過去の変更を含むREDOデータがスタンバイ・データベースに適用された後、スタンバイ・システムで対応するデータファイルを削除します。次に例を示します。

```
% rm /disk1/oracle/oradata/payroll/s2tbs_4.dbf
```

削除した表領域のREDO情報がスタンバイ・データベースで適用されたことを確認した後、プライマリ・データベースで表領域のデータファイルを削除できます。次に例を示します。

```
% rm /disk1/oracle/oradata/payroll/tbs_4.dbf
```

10.3.2.1 DROP TABLESPACE INCLUDING CONTENTS AND DATAFILESの使用

プライマリ・データベースでSQL DROP TABLESPACE INCLUDING CONTENTS AND DATAFILES文を発行すると、プライマリ・デー

データベースとスタンバイ・データベースの両方からデータファイルを削除できます。

この文を使用するには、STANDBY_FILE_MANAGEMENT初期化パラメータをAUTOに設定する必要があります。たとえば、プライマリ・サイトで表領域を削除するには、次のように入力します。

```
SQL> DROP TABLESPACE tbs_4 INCLUDING CONTENTS AND DATAFILES;  
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

10.3.3 フィジカル・スタンバイ・データベースでのトランスポータブル表領域の使用

Oracleトランスポータブル表領域機能を使用すると、Oracleデータベースのサブセットを移動して、別のOracleデータベースにプラグインできます。これにより、実際には表領域がデータベース間で移動します。

フィジカル・スタンバイの使用中に表領域セットをプライマリ・データベースに移動またはコピーするステップは、次のとおりです。

1. トランスポートする表領域セットのデータファイルとその表領域セットの構造情報を含むエクスポート・ファイルで構成される、トランスポータブル表領域セットを生成します。
2. 次の手順で表領域セットをトランスポートします。
 - a. データファイルをコピーし、プライマリ・データベースにエクスポートします。
 - b. データファイルをスタンバイ・データベースにコピーします。

DB_FILE_NAME_CONVERTデータベース初期化パラメータが構成されていないかぎり、データファイルのパス名はプライマリ・データベースとスタンバイ・データベースで同じものが使用される必要があります。DB_FILE_NAME_CONVERTが構成されてなく、プライマリ・データベースとスタンバイ・データベースでデータファイルのパス名が同じではない場合、ALTER DATABASE RENAME FILE文を発行して、データファイルの名前を変更します。この操作は、REDO Applyが、表領域をプライマリ・データベースにプラグすることで生成されたREDOの適用に失敗した後で行います。データ・ファイル名を変更する前に、STANDBY_FILE_MANAGEMENT初期化パラメータをMANUALに設定し、データ・ファイル名を変更した後で、前の値にリセットする必要があります。

3. 表領域をプラグインします。

データ・ポンプ・ユーティリティを起動して、表領域セットをプライマリ・データベースにプラグインします。スタンバイ・サイトでREDOデータが生成されて適用され、表領域がスタンバイ・データベースにプラグインされます。

トランスポータブル表領域の詳細は、『[Oracle Database管理者ガイド](#)』を参照してください。

10.3.4 プライマリ・データベースのデータファイルの改名

プライマリ・データベースで1つ以上のデータファイルを改名した場合、その変更はスタンバイ・データベースに伝播されません。それは手動で行う必要があります。

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。

次のステップでは、プライマリ・データベースでデータファイルを改名し、その変更をスタンバイ・データベースに手動で伝播する方法について説明します。

1. プライマリ・データベースでデータファイルを改名するには、表領域をオフラインにします。

```
SQL> ALTER TABLESPACE tbs_4 OFFLINE;
```

2. SQLプロンプトを終了し、UNIXのmvコマンドなどのオペレーティング・システム・コマンドを発行して、プライマリ・システム上のデータファイルを改名します。

```
% mv /disk1/oracle/oradata/payroll/tbs_4.dbf  
/disk1/oracle/oradata/payroll/tbs_x.dbf
```

3. プライマリ・データベース内のデータファイルを改名し、表領域をオンラインに戻します。

```
SQL> ALTER TABLESPACE tbs_4 RENAME DATAFILE -  
> '/disk1/oracle/oradata/payroll/tbs_4.dbf' -  
> TO '/disk1/oracle/oradata/payroll/tbs_x.dbf';  
  
SQL> ALTER TABLESPACE tbs_4 ONLINE;
```

ノート:



最初の 3 つのステップのかわりに、ALTER DATABASE MOVE DATAFILE コマンドを使用してデータファイルを改名する方法があります。このコマンドはデータファイルへの読み取り/書き込みアクセスを許可しながらデータファイルを改名できます。MOVE DATAFILE コマンドの実行中、データベースは元のデータファイルと改名したデータファイルの両方を 2 つの別個のファイルとして保持するので、データファイルの移動には、十分な記憶域が前提条件です。詳細は、[「オンライン・データ・ファイルの場所の移行」](#)を参照してください。

4. スタンバイ・データベースに接続し、REDO Applyを停止します。

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

5. スタンバイ・データベースを停止します。

```
SQL> SHUTDOWN;
```

6. UNIXのmvコマンドなどのオペレーティング・システム・コマンドを使用して、スタンバイ・サイトでデータファイルを改名します。

```
% mv /disk1/oracle/oradata/payroll/tbs_4.dbf /disk1/oracle/oradata/payroll/tbs_x.dbf
```

7. スタンバイ・データベースを起動し、マウントします。

```
SQL> STARTUP MOUNT;
```

8. スタンバイ制御ファイルのデータファイルを改名します。データ・ファイルを改名するには、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。これで、パラメータは、データ・ファイル名の変更後、前の値にリセットできます。

```
SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/tbs_4.dbf' -  
> TO '/disk1/oracle/oradata/payroll/tbs_x.dbf';
```

9. スタンバイ・データベースで、REDO Applyを再開します。

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

スタンバイ・システムで対応するデータ・ファイルを改名せず、スタンバイ・データベースの制御ファイルのリフレッシュを試みた場合、スタンバイ・データベースは改名したデータ・ファイルを使用しようとしていますが、見つかりません。次のようなメッセージがアラート・ログに書き込まれます。

```
ORA-00283: recovery session canceled due to errors  
ORA-01157: cannot identify/lock datafile 4 - see DBWR trace file  
ORA-01110: datafile 4: '/Disk1/oracle/oradata/payroll/tbs_x.dbf'
```


ノート:



ステップ 4-9 のかわりに、ALTER DATABASE MOVE DATAFILE コマンドを使用してスタンバイでデータファイルを改名する方法があります。詳細は、[「オンライン・データ・ファイルの場所の移行」](#)を参照してください。

10.3.5 REDOログ・ファイル・グループの追加または削除

プライマリ・データベースでログ・ファイル・グループを追加または削除した後に、フィジカル・スタンバイ・データベースのREDOログおよびスタンバイREDOログの構成を再評価し、必要に応じて調整する必要があります。

次のステップを実行して、フィジカル・スタンバイ・データベースでログ・ファイル・グループまたはスタンバイ・ログ・ファイル・グループを追加または削除します。

1. REDO Applyを停止します。
2. STANDBY_FILE_MANAGEMENT初期化パラメータをAUTOに設定している場合は、値をMANUALに変更します。
3. ログ・ファイル・グループを追加または削除します。

ノート:

オンライン・ログ・ファイル・グループは、フィジカル・スタンバイ・データベースから削除するには、常に手動でクリアする必要があります。次に例を示します。



```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

ステータスが CURRENT または CLEARING_CURRENT であるオンライン・ログ・ファイル・グループは、フィジカル・スタンバイ・データベースから削除できません。このステータスのオンライン・ログ・ファイル・グループは、ロール推移後に削除できます。

4. STANDBY_FILE_MANAGEMENT初期化パラメータおよびREDO Applyオプションを元の状態にリストアします。
5. REDO Applyを再開します。

Oracle RAC環境では、次の点に注意してください。

- プライマリ・データベースにオンラインREDOログ・グループが追加されるとき、スタンバイ・データベースにはオンラインREDOログ・グループを手動で追加する必要があります。自動的に実行されません。
- プライマリ・データベースに新しいREDOスレッドが追加されるとき、スタンバイには新しいREDOスレッドが自動的に追加されます。デフォルトでは、新しいスレッドは100 MBずつの2つのログ・グループで構成されます。これは変更もオーバーライドもできません。
- 既存のREDOスレッドに新しいログ・グループが追加されるとき、既存のスレッドに新しいログ・グループは自動的に追加されません。

10.3.6 NOLOGGING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、スタンバイのブロックが無効とマークされることがあります(ログに記録されないブロックとも呼ばれます)。

ログに記録されない方法で操作が実際に実行される場合の詳細は、[FORCE LOGGING設定の優先順位](#)を参照してください。

NOLOGGING句を使用した後のリカバリの詳細は、[「NOLOGGING句を指定した後のリカバリ」](#)を参照してください。

FORCE LOGGINGモードの指定の詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。

10.3.7 パスワード・ファイルのリフレッシュ

REMOTE_LOGIN_PASSWORDFILEデータベース初期化パラメータがSHAREDまたはEXCLUSIVEに設定されている場合は、フィジカル・スタンバイ・データベース上のパスワード・ファイルは、プライマリ・データベースからの最新コピーと自動的に置き換えられます。

ファイルは、管理権限が付与または解除されるか、管理権限があるユーザーのパスワードが変更されると、置き換えられます。唯一の例外は、遠隔同期インスタンスです。遠隔同期インスタンスはREDOを受け取りますが、適用しないため、更新したパスワード・ファイルを引き続き手動で遠隔同期インスタンスにコピーする必要があります。遠隔同期インスタンスでパスワード・ファイルが手動で更新されると、プライマリ・データベースからの同じパスワード変更内容を含むREDOが、遠隔同期インスタンスからREDOを受け取るように設定されているすべてのスタンバイ・データベースに自動的に伝播されます。REDOが適用されると、スタンバイでパスワード・ファイルが更新されます。

10.3.8 TDEマスター暗号化キーのリセット

フィジカル・スタンバイ・データベースのデータベース暗号化ウォレットは、プライマリ・データベースでTDEマスター暗号化キーがリセットされるたびに、プライマリ・データベースのデータベース暗号化ウォレットの最新コピーで置き換える必要があります。

フィジカル・スタンバイ・データベースのデータベース暗号化ウォレットをリフレッシュできないと、プライマリ・データベースでマスター暗号化キーがリセットされた後に変更された、フィジカル・スタンバイ・データベース上の暗号化された列にアクセスできません。

オンラインの表領域およびデータベースの場合、Oracle Database 12cリリース2 (12.2.0.1)以降、新規および既存の両方の表領域、およびOracle Data Guard環境内の既存のデータベースを暗号化、復号化およびキー更新できます。

オフラインの表領域およびデータベースの場合、Oracle Database 12cリリース2 (12.2.0.1)以降、新規および既存の両方の表領域、およびOracle Data Guard環境内の既存のデータベースを暗号化および復号化できます。

オンライン変換では、スタンバイ上の暗号化、復号化またはキー更新は、プライマリでの実行後に自動で行われます。オンラインの暗号化、復号化またはキー更新は、スタンバイ・データベースで直接に実行することはできません。

オフライン変換では、暗号化または復号化はプライマリとスタンバイ上の両方に手動で実行する必要があります。オフライン変換は、特定のプライマリまたはスタンバイ・データベースのみのデータ・ファイルに影響を及ぼします。プライマリおよびフィジカル・スタンバイの両方を同じ状態に維持する必要があります。最初にスタンバイ上の表領域を暗号化(または復号化)し、プライマリにスイッチしてからプライマリの表領域を暗号化(または復号化)することで、停止時間を最小化できます。

関連項目:

- [Oracle Database Advanced Securityガイド](#)

10.4 OPEN RESETLOGS文を使用したリカバリ

Oracle Data Guardでは、RESETLOGSオプションを使用してプライマリ・データベースがオープンされた後も、フィジカル・スタンバイ・データベースでリカバリを続行できます。

プライマリ・データベースでALTER DATABASE OPEN RESETLOGS文が発行されると、データベースのインカネーションが変更され、

REDOデータの新しいブランチが作成されます。

フィジカル・スタンバイ・データベースがREDOデータの新規ブランチを受信すると、REDO Applyではそれが自動的に使用されます。フィジカル・スタンバイ・データベースの場合、スタンバイ・データベースにより新規リセットログのSCNより後(REDOデータの新規ブランチの開始より後)のREDOデータが適用されていなければ、手動による介入は必要ありません。次の表に、スタンバイ・データベースとプライマリ・データベースのブランチを再同期化する方法を示します。

スタンバイ・データベースの状態 . .	操作 . .	実行するステップ . .
新規リセットログの SCN の後(REDO データの新規ブランチの開始より後)の REDO データが適用されておらず、OPEN RESETLOGS の新規 REDO ブランチがスタンバイに登録されている場合	REDO Apply は自動的に REDO の新規ブランチを使用します。	手動による介入は必要ありません。管理された REDO プロセス(MRP)は、自動的にスタンバイ・データベースを REDO データの新規ブランチと再同期化します。 ノート: 新規 REDO ブランチがスタンバイに登録されたかどうかを調べるには、プライマリおよびスタンバイで次の問合せを実行して、結果が一致することを確認します。 <pre>SELECT resetlogs_id, resetlogs_change# FROM V\$DATABASE_INCARNATION WHERE status=' CURRENT'</pre>
新規リセットログの SCN の後(REDO データの新規ブランチの開始より後)の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっている場合。	スタンバイ・データベースは、REDO データの将来の新規ブランチでリカバリされます。	1. 「特定時点へのフィジカル・スタンバイ・データベースのフラッシュバック」の手順に従って、フィジカル・スタンバイ・データベースをフラッシュバックします。 2. REDO Apply を再開し、新規リセットログ・ブランチへの REDO データの適用を続行します。 管理された REDO プロセス(MRP)は、自動的にスタンバイ・データベースを新規ブランチと再同期化します。
新規リセットログの SCN の後(REDO データの新規ブランチの開始より後)の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっていない場合。	指定のプライマリ・データベース・ブランチで、プライマリ・データベースとスタンバイの内容にずれが生じます。	「フィジカル・スタンバイ・データベースの作成」 の手順に従ってフィジカル・スタンバイ・データベースを再作成します。
REDO データの新規ブランチから介入するアーカイブ REDO ログ・ファイルが欠落している場合。	MRP は欠落しているログ・ファイルが取得されるまで続行できません。	各ブランチから欠落しているアーカイブ REDO ログ・ファイルを検索して登録します。

スタンバイ・データベースの状態 . .	操作 . .	実行するステップ . .
REDO データの前のブランチの終わりからアーカイブ REDO ログ・ファイルが欠落している場合。	MRP は欠落しているログ・ファイルが取得されるまで続行できません。	前のブランチから欠落しているアーカイブ REDO ログ・ファイルを検索して登録します。

データベース・インカネーション、OPEN RESETLOGSによるリカバリ、フラッシュバック・データベースの詳細は、[『Oracle Database バックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

10.5 プライマリRESETLOGS操作の後のマウント済スタンバイの自動フラッシュバック

マウント状態のスタンバイ・データベースは、プライマリでのRESETLOGS操作の後、プライマリ・データベースを自動的に継ぐことができます。これにより、プライマリでのRESETLOGS操作の後のスタンバイ管理が簡略化されます。

プライマリ・データベースまたはプライマリ・データベース内のPDBのいずれかでフラッシュバックまたはPoint-in-Timeリカバリが行されると、プライマリ・データベースまたはPDBは過去のある時点に戻され、その後、RESETLOGSオプションを使用してオープンされます。プライマリまたはプライマリ内のPDBの新しいインカネーションが作成されます。スタンバイが自動的にプライマリを継ぐようにするために、MRPによって次のアクションが実行されます。

- 新しいインカネーションを検出します
- スタンバイまたはスタンバイ上のPDBを、プライマリまたはプライマリ上のPDBと同じ時点までフラッシュバックします
- スタンバイ・リカバリを再開し、REDOの新規ブランチにスタンバイを移動します

フラッシュバック操作は、スタンバイ・データベースに十分なフラッシュバック・データがある場合にのみ成功します。

スタンバイが自動的にプライマリを継がないようにする場合は、スタンバイ・データベースをOPENモードのままにするか、スタンバイでMRPプロセスを停止します。

スタンバイ・データベースが読み取り専用モードでオープンされている場合、対応するエラー・メッセージはアラート・ログに記録されます。フィジカル・スタンバイをクローズした後にMRPを再起動すると、リカバリ・プロセスにより自動的にスタンバイ・データベースがフラッシュバックされ、REDOの新規ブランチの適用が続行されます。

関連トピック

- [プライマリでのPDB PITRまたはPDBフラッシュバック後にスタンバイで必要なアクション](#)
- [特定時点へのフィジカル・スタンバイ・データベースのフラッシュバック](#)

10.6 プライマリ、フィジカル・スタンバイおよびスナップショット・スタンバイ・データベースの監視

次の表に、一般的なプライマリ・データベースの管理アクションと、そのアクションに関する情報の参照先を示します。

表10-2 一般的なプライマリ・データベースの管理アクションに関する情報ソース

プライマリ・データベースのアクション	プライマリ・サイト情報	スタンバイ・サイト情報
REDO スレッドの有効化または無効化	● アラート・ログ	アラート・ログ

プライマリ・データベースのアクション	プライマリ・サイト情報	スタンバイ・サイト情報
	● V\$THREAD	
データベース・ロール、保護モード、保護レベル、スイッチオーバー・ステータス、ファスト・スタート・フェイルオーバー情報などの表示	V\$DATABASE	V\$DATABASE
REDO ログ・ファイル・グループの追加または削除	● アラート・ログ ● V\$LOG ● V\$LOGFILE の STATUS 列	アラート・ログ
CREATE CONTROLFILE	アラート・ログ	アラート・ログ
REDO Apply の監視	● アラート・ログ ● V\$ARCHIVE_DEST_STATUS	● アラート・ログ ● V\$ARCHIVED_LOG ● V\$LOG_HISTORY ● V\$MANAGED_STANDBY
表領域ステータスの変更	● V\$RECOVER_FILE ● DBA_TABLESPACES ● アラート・ログ	● V\$RECOVER_FILE ● DBA_TABLESPACES
データファイルまたは表領域の追加または削除	● DBA_DATA_FILES ● アラート・ログ	● V\$DATAFILE ● アラート・ログ
データファイルの名前を変更します。	● V\$DATAFILE ● アラート・ログ	● V\$DATAFILE ● アラート・ログ
ログに記録されていないまたはリカバリ不能な操作	● V\$DATAFILE ● V\$DATABASE	アラート・ログ
REDO 転送の監視	● V\$ARCHIVE_DEST_STATUS ● V\$ARCHIVED_LOG ● V\$ARCHIVE_DEST	● V\$ARCHIVED_LOG ● アラート・ログ

プライマリ・データベースのアクション	プライマリ・サイト情報	スタンバイ・サイト情報
● 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

10.6.1 プライマリ、フィジカルおよびスナップショット・スタンバイ・データベースを監視するためのビューの使用

プライマリ、フィジカル・スタンバイおよびスナップショット・スタンバイ・データベースを監視するには、動的パフォーマンス・ビューを使用します。

次の動的パフォーマンス・ビューについて説明します。

- [V\\$DATABASE](#)
- [V\\$MANAGED_STANDBY](#)
- [V\\$ARCHIVED_LOG](#)
- [V\\$LOG_HISTORY](#)
- [V\\$DATAGUARD_STATUS](#)
- [V\\$ARCHIVE_DEST](#)

関連項目:

ビューの詳細な参照情報は、『[Oracle Databaseリファレンス](#)』を参照してください。

10.6.1.1 V\$DATABASE

V\$DATABASEビューを使用すると、データ保護、スイッチオーバー・ステータス、ファスト・スタート・フェイルオーバーのステータスに関する情報を確認できます。

次の問合せは、プライマリ・データベース、フィジカル・スタンバイ・データベースまたはスナップショット・スタンバイ・データベースのデータ保護モード、データ保護レベル、データベース・ロールおよびスイッチオーバー・ステータスを表示します。

```
SQL> SELECT PROTECTION_MODE, PROTECTION_LEVEL, -
> DATABASE_ROLE ROLE, SWITCHOVER_STATUS -
> FROM V$DATABASE;
```

次の問合せは、ファスト・スタート・フェイルオーバーのステータスを表示します。

```
SQL> SELECT FS_FAILOVER_STATUS "FSFO STATUS", -
> FS_FAILOVER_CURRENT_TARGET TARGET, -
> FS_FAILOVER_THRESHOLD THRESHOLD, -
> FS_FAILOVER_OBSERVER_PRESENT "OBSERVER PRESENT" -
> FROM V$DATABASE;
```


10.6.1.2 V\$DATAGUARD_PROCESS

V\$DATAGUARD_PROCESSビューには、現在実行されているOracle Data Guardプロセスごとに1つの行が表示されます。

V\$DATAGUARD_PROCESSビューは、Oracle Database 12cリリース2 (12.2.0.1)から非推奨となっており、将来のリリースでサポートされなくなる可能性があるV\$MANAGED_STANDBYビューのかわりに使用します。

このビューの問合せの例を次に示します。

```
SQL> SELECT ROLE, THREAD#, SEQUENCE#, ACTION FROM V$DATAGUARD_PROCESS;
```

ROLE	THREAD#	SEQUENCE#	ACTION
RFS ping	1	9	IDLE
recovery apply slave	0	0	IDLE
recovery apply slave	0	0	IDLE
managed recovery	0	0	IDLE
recovery logmerger	1	9	APPLYING_LOG
RFS archive	0	0	IDLE
RFS async	1	9	IDLE

10.6.1.3 V\$MANAGED_STANDBY

V\$MANAGED_STANDBYビューを使用すると、フィジカル・スタンバイ・データベースでREDO ApplyおよびREDO転送ステータスを問い合わせることができます。

ノート:



このビューは、Oracle Database 12c リリース 2 (12.2.0.1)から非推奨となっており、将来のリリースで削除される可能性があります。かわりに、V\$DATAGUARD_PROCESS ビューを使用する必要があります。

V\$MANAGED_STANDBYビューの問合せの例を次に示します。

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

PROCESS	STATUS	THREAD#	SEQUENCE#	BLOCK#	BLOCKS
RFS	ATTACHED	1	947	72	72
MRPO	APPLYING_LOG	1	946	10	72

このサンプル出力は、リモート・ファイル・サーバー(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、REDO ApplyがアクティブにアーカイブREDOログ・ファイル順序番号946を適用していることを示しています。また、REDO ApplyがアーカイブREDOログ・ファイル順序番号946を適用していることも示しています。

10.6.1.4 V\$ARCHIVED_LOG

V\$ARCHIVED_LOGビューを使用すると、プライマリ・データベースからフィジカル・スタンバイ・データベースまたはスナップショット・スタンバイ・データベースで受信されたアーカイブREDOログ・ファイルに関する情報を問い合わせることができます。

たとえば、次の問合せを発行します。

```
SQL> SELECT THREAD#, SEQUENCE#, FIRST_CHANGE#, -  
> NEXT_CHANGE# FROM V$ARCHIVED_LOG;
```

THREAD#	SEQUENCE#	FIRST_CHANGE#	NEXT_CHANGE#
1	945	74651	74739
1	946	74739	74772
1	947	74772	74795

このサンプル出力は、プライマリ・データベースから受信した3つのアーカイブREDOログ・ファイルを示しています。

10.6.1.5 V\$LOG_HISTORY

V\$LOG_HISTORYビューを使用すると、アーカイブ・ログの履歴情報を問い合わせることができます。

たとえば、次の問合せを発行します。

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

10.6.1.6 V\$DATAGUARD_STATUS

V\$DATAGUARD_STATUSビューを使用して、アラート・ログまたはサーバー・プロセス・トレース・ファイルにメッセージが書き込まれる原因となったOracle Data Guardイベントによって生成されたメッセージを表示できます。

たとえば、次の問合せを発行します。

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

10.6.1.7 V\$ARCHIVE_DEST

V\$ARCHIVE_DESTビューを問い合わせ、各REDO転送先のステータスを表示します。スタンバイ・データベースであるREDO転送先については、そのスタンバイ・データベースで適用された最後のプライマリ・データベースREDOのSCNを表示します。

たとえば、次の問合せを発行します。

```
SQL> SELECT DEST_ID, STATUS, APPLIED_SCN FROM V$ARCHIVE_DEST WHERE TARGET=' STANDBY' ;
```

DEST_ID	STATUS	APPLIED_SCN
2	VALID	439054
3	VALID	439054

10.7 プライマリからスタンバイへのリストア・ポイントのレプリケート

プライマリ・データベースで作成されたリストア・ポイントは、自動的にスタンバイ・データベースにレプリケートされます。スタンバイ・データベースで作成されたリストア・ポイントは、レプリケートされたリストア・ポイントと呼ばれます。プライマリ・データベースのリストア・ポイントが保証付きリストア・ポイントか通常のリストア・ポイントかに関係なく、対応するレプリケートされたリストア・ポイントは、常に、通常のリストア・ポイントになります。

次の条件が満たされると、Oracle Databaseでは、リストア・ポイントがプライマリ・データベースからスタンバイ・データベースに自動的にレプリケートされます。

- プライマリ・データベースとスタンバイ・データベースの両方のCOMPATIBLE初期化パラメータが19.0.0以降に設定されている
- プライマリ・データベースがオープンされている

プライマリがマウント・モードの場合にプライマリ・データベース上で作成されたリストア・ポイントは、レプリケートされません。この制限は、リストア・ポイント情報がREDOを介してレプリケートされるためです。

レプリケートされたリストア・ポイントのネーミング規則では、プライマリ・データベース上のリストア・ポイントの名前にPRIMARYを接尾辞として付加したものが使用されます。スタンバイ・データベース上に同じ名前のレプリケートされたリストア・ポイントが存在する場合、レプリケートされたリストア・ポイントは作成されません。たとえば、プライマリ・データベースでPRE_MYTBSという名前のリストア・ポイントを作成すると、レプリケートされたリストア・ポイントはPRE_MYTBS_PRIMARYという名前になります。プライマリ上でリストア・ポイントを削除すると、スタンバイ上の対応するレプリケートされたリストア・ポイントも削除されます。

管理されたREDOプロセス(MRP)は、レプリケートされたリストア・ポイントの作成およびメンテナンスを管理します。MRPが実行されていないときにプライマリ・データベース上でリストア・ポイントが作成されると、これらのリストア・ポイントは、MRPの起動後にスタンバイ・データベースにレプリケートされます。

リストア・ポイントが自動的にレプリケートされるかどうかを判断するには、V\$RESTORE_POINTビューを問い合わせます。

10.8 REDO Applyのチューニング

REDO Applyおよびメディア・リカバリのパフォーマンスを最適化する方法について説明するOracleの技術概要が提供されています。

特に、*Active Data Guard*の11gのベスト・プラクティス(*REDO Apply*のベスト・プラクティスを含む)を参照してください。このホワイト・ペーパーは、次の場所からOracle Maximum Availability Architecture(MAA)のホームページにアクセスして入手することができます。

<http://www.oracle.com/goto/maa>

関連項目:

Standby Statspackのインストールと使用の詳細は、<http://support.oracle.com>のMy Oracle Supportノート454848.1を参照してください。Standby Statspackはフィジカル・スタンバイ・データベースからRedo Applyパフォーマンス・データを収集するために使用することもできます。

10.9 SQLチューニング・アドバイザによるActive Data Guard環境でのデータベースのチューニング

Active Data Guard環境では、SQLチューニング・アドバイザでプライマリ・データベース上のスタンバイ・ワークロードをチューニングできます。

データベース・リンクを使用すると、特定のデータベースでSQLチューニング・アドバイザ文を発行し、異なるデータベースで文を実行できます。

プライマリ・データベースのスタンバイ・データベース・ワークロードのチューニング

場合によっては、スタンバイ・データベースでは、データ保護ロールに加えてレポート・ロールを仮定できます。スタンバイ・データベースでは、問合せの固有のワークロードを含めることができ、一部でチューニングが必要になる場合があります。このシナリオで、スタンバイ・データベースで各チューニング文を発行してスタンバイ・データベース・ワークロードをチューニングしますが、SQLチューニング・アドバイザでは、スタンバイとプライマリのデータベース・リンクを使用してプライマリ・データベースで分析を実行します。

プライマリ・データベースのスタンバイ・データベース・ワークロードをチューニングするために実行する必要があるタスクを次に示します。タスクは、スタンバイ・データベースで、指定された順序で、DBMS_SQLTUNE PL/SQLパッケージを使用して実行する必要があります。

1. DBMS_SQLTUNE.CREATE_TUNING_TASK文を実行して、タスクの作成に必要なデータをプライマリ・データベースからフェッ

ちします。スタンバイは読取り専用データベースのため、タスクに関するデータはリモートでプライマリ・データベースに書き込まれます。プライマリに書き込むこのステップでは、データベース・リンク・パラメータが必要です。(データベース・リンク・パラメータは、このステップで作成されるタスクに関連付けられているため後続のステップではオプションです。)

2. DBMS_SQLTUNE.EXECUTE_TUNING_TASK文を実行します。最初は、タスクを実行するために必要なデータはリモートのプライマリ・データベースからフェッチされます。考えられるお薦め事項を見つけるためのチューニング分析プロセスが実行されます。スタンバイは読取り専用データベースのため、結果が使用可能な場合はリモートでプライマリに格納されます。
3. DBMS_SQLTUNE.REPORT_TUNING_TASK文を実行します。レポートを作成するために必要なデータは、リモートでプライマリ・データベースに格納されます。データはリモートでプライマリからフェッチされ、ローカルにスタンバイで作成されます。
4. DBMS_SQLTUNE.ACCEPT_SQL_PROFILE文を実行します。プロファイル・データは、スタンバイが読取り専用のためリモートのプライマリ・データベースに書き込まれます。
5. SQLプロファイルは、REDO Applyを使用してスタンバイで使用可能にされます。

関連項目:

- DBMS_SQLTUNEパッケージの詳細は、[Oracle Database PL/SQLパッケージおよびタイプ・リファレンス](#)を参照してください。
- ローカルおよびリモートのSQLチューニングの詳細は、[Oracle Database SQLチューニング・ガイド](#)を参照してください。

10.10 Oracle Diagnostic Packを使用したOracle Active Data Guardのスタンバイのチューニング

Oracle Diagnostic Packは、読取り専用モードでオープンされているOracle Active Data Guardスタンバイ・データベースとともに使用できます。

これにより、Oracle Active Data Guardスタンバイ用の自動ワークロード・リポジトリ(AWR)へのパフォーマンス・データを取得し、AWRデータに関する自動データベース診断モニター(ADDM)分析を実行できます。これらの操作の実行方法の詳細は、[Oracle Databaseパフォーマンス・チューニング・ガイド](#)を参照してください。

10.11 スナップショット・スタンバイ・データベースの管理

スナップショット・スタンバイ・データベースは完全に更新可能なスタンバイ・データベースです。これは、プライマリ・データベースからREDOデータを受信およびアーカイブしますが、適用はしません。

プライマリ・データベースから受信したREDOデータは、スナップショット・スタンバイ・データベースへのローカル更新がすべて破棄された後、スナップショット・スタンバイ・データベースが変換されてフィジカル・スタンバイ・データベースに戻ると適用されます。

プライマリ・データベースのREDOデータは受信と同時に適用されないため、スナップショット・スタンバイ・データベースは通常、時間の経過とともにプライマリ・データベースとの差異が生じます。スナップショット・スタンバイ・データベースのローカルな更新によって差異が増えます。ただし、スナップショット・スタンバイはいつでもフィジカル・スタンバイ・データベースに戻すことができ、その後プライマリから受信したREDOデータを適用できるため、プライマリ・データベース内のデータは完全に保護されています。

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

10.11.1 スナップショット・スタンバイ・データベースへのフィジカル・スタンバイ・データベースの変換

これらのステップでは、フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースに変換する方法について説明します。

1. REDO Applyがアクティブな場合は、停止します。
2. データベースがマウントされ、オープンしていないことを確認します。
3. 高速リカバリ領域が構成されていることを確認します。フラッシュバック・データベースを有効化する必要はありません。
4. 次のSQL文を発行して変換を実行します。

```
SQL> ALTER DATABASE CONVERT TO SNAPSHOT STANDBY;
```

5. 次のSQL文を発行して、読取り/書込みモードでスナップショット・スタンバイをオープンします。

```
SQL> ALTER DATABASE OPEN READ WRITE;
```

ノート:



Oracle Data Guard Broker で管理されるフィジカル・スタンバイ・データベースは、DGMGRL または Oracle Enterprise Manager Cloud Control のいずれかを使用してスナップショット・スタンバイ・データベースに変換できます。詳細は、『[Oracle Data Guard Broker](#)』を参照してください。

10.11.2 スナップショット・スタンバイ・データベースの使用

スナップショット・スタンバイ・データベースは読取り/書込みモードでオープンされ、全体を更新できます。

スナップショット・スタンバイ・データベースには、次の特性があります。

- スナップショット・スタンバイ・データベースは、スイッチオーバーまたはフェイルオーバーのターゲットにはできません。スナップショット・スタンバイ・データベースは、ロールの推移を実行する前にフィジカル・スタンバイ・データベースに変換して戻す必要があります。
- スナップショット・スタンバイ・データベースは、最大保護のOracle Data Guard構成では唯一のスタンバイ・データベースになりません。

ノート:



フラッシュバック・データベースを使用して、スナップショット・スタンバイ・データベースを変換してフィジカル・スタンバイ・データベースに戻すことができます。フラッシュバック・データベース・テクノロジーを使用して元に戻せない操作では、スナップショット・スタンバイを変換してフィジカル・スタンバイに戻せません。

フラッシュバック・データベースの詳細と制約は、『[Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。

10.11.3 フィジカル・スタンバイ・データベースへのスナップショット・スタンバイ・データベースの変換

これらのステップでは、スナップショット・スタンバイ・データベースをフィジカル・スタンバイ・データベースに変換する方法について説明します。

1. Oracle Real Application Clusters(Oracle RAC)データベースで、1つのインスタンス以外はすべて停止します。
2. データベースがマウントされ、オープンしていないことを確認します。
3. 次のSQL文を発行して変換を実行します。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

データベースがスナップショット・スタンバイ・データベースである間に受信したREDOデータは、REDO Applyが開始されると自動的に適用されます。

ノート:



スナップショット・スタンバイ・データベースは、少なくとも 1 回は読取り/書込みモードでオープンしてから、フィジカル・スタンバイ・データベースに変換する必要があります。

11 ロジカル・スタンバイ・データベースの管理

これらの概念を理解しておくと、ロジカル・スタンバイ・データベースを正常に管理するのに役立ちます。

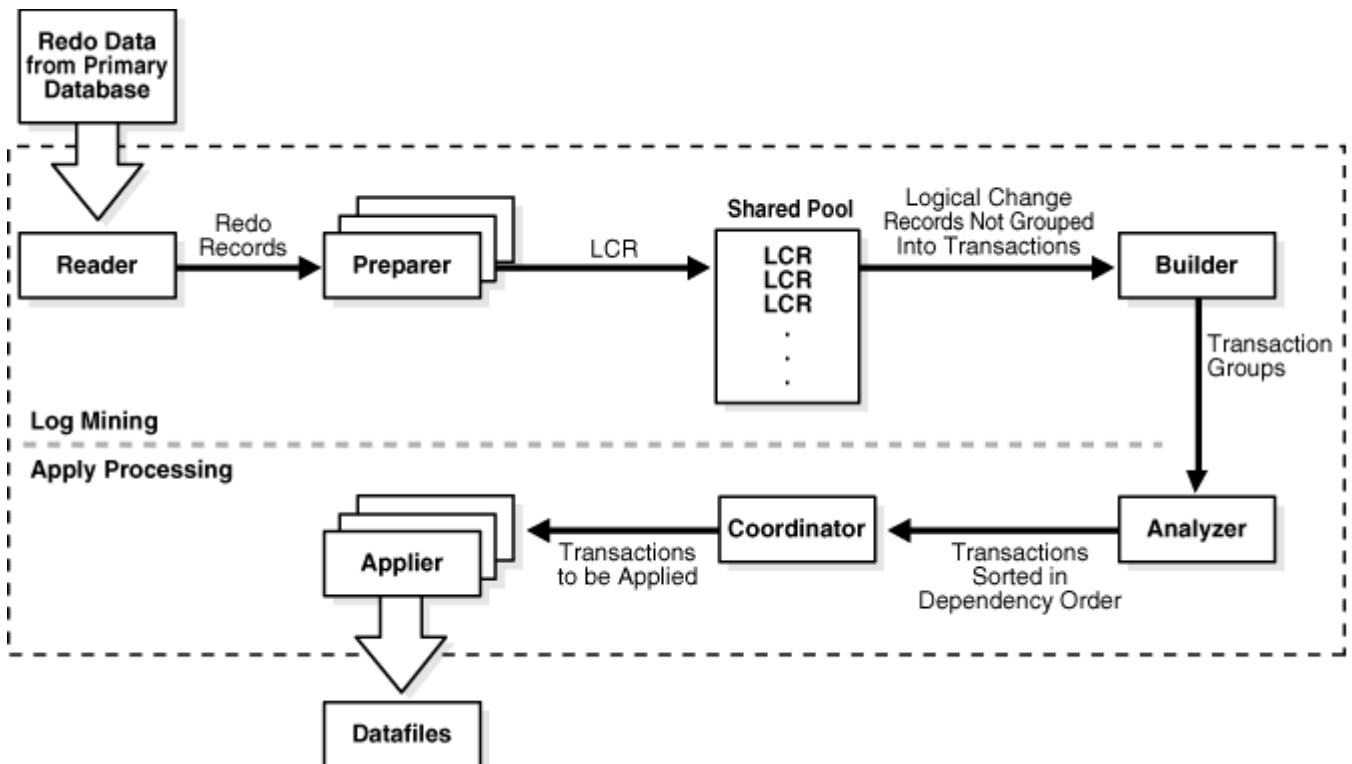
- [SQL Applyアーキテクチャの概要](#)
- [ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスの制御](#)
- [ロジカル・スタンバイ・データベースの管理および監視関連のビュー](#)
- [ロジカル・スタンバイ・データベースの監視](#)
- [ロジカル・スタンバイ・データベースのカスタマイズ](#)
- [ロジカル・スタンバイ・データベースの下での特定のワークロードの管理](#)
- [ロジカル・スタンバイ・データベースのチューニング](#)
- [ロジカル・スタンバイ・データベースの下でのバックアップおよびリカバリ](#)

11.1 SQL Applyアーキテクチャの概要

SQL Applyでは、バックグラウンド・プロセスの集合を使用して、プライマリ・データベースの変更がロジカル・スタンバイ・データベースに適用されます。

図11-1に、情報のフローと各プロセスで実行されるロールを示します。

図11-1 SQL Applyの処理



ログのマイニングおよび適用処理中は、次のように、様々なプロセスとその機能が関係します。

ログのマイニング中には、次のプロセスが実行されます。

- READERプロセスでは、REDOレコードがアーカイブREDOログ・ファイルまたはスタンバイREDOログ・ファイルから読み込まれます。
- PREPARERプロセスでは、REDOレコードに含まれるブロックの変更が論理変更レコード(LCR)に変換されます。特定の

REDOログ・ファイルに対して複数のPREPARERプロセスをアクティブにすることができます。LCRは、LCRキャッシュと呼ばれるシステム・グローバル領域(SGA)でステージングされます。

- BUILDERプロセスでは、LCRがトランザクション単位にグループ化されて、LCRキャッシュ内のメモリー管理、SQL Applyの再起動に関連するチェックポイント処理および重要でない変更のフィルタリングなど、その他のタスクが実行されます。

適用処理中には、次のプロセスが実行されます。

- ANALYZERプロセスでは、様々なトランザクション間の依存性が識別されます。
- COORDINATORプロセス(LSP)では、トランザクションを様々なAPPLIERに割り当て、トランザクション間の依存性が考慮されるように調整します。
- APPLIERプロセスでは、COORDINATORプロセスの管理下でロジカル・スタンバイ・データベースにトランザクションを適用します。

V\$LOGSTDBY_PROCESSビューで問い合せて、SQL Applyプロセスのアクティビティを確認することができます。現在のアクティビティの情報を提供する別なビューにV\$LOGSTDBY_STATSがあり、SQL Applyアクティビティ中のロジカル・スタンバイ・データベースのために統計、現在の状態、ステータス情報を表示します。このビューと関連するその他のビューの詳細は、[「ロジカル・スタンバイ・データベースの管理と監視に関連するビュー」](#)を参照してください。

ノート:



SQL Apply プロセス(COORDINATOR プロセス lsp0 を含む)はすべてバックグラウンド・プロセスです。リソース・マネージャによる調整は行われません。そのため、ロジカル・スタンバイ・データベースでリソース・グループを作成しても、SQL Apply プロセスには影響ありません。

11.1.1 SQL Applyに関する各種の考慮事項

トランザクション・サイズ、ページアウト、再起動、DML適用、およびパスワード検証について理解すると、ロジカル・スタンバイを管理して最適利用するうえで役立ちます。

次を参照してください。

- [トランザクション・サイズの考慮事項](#)
- [ページアウトの考慮事項](#)
- [再開の考慮事項](#)
- [DML適用の考慮事項](#)
- [DDL適用の考慮事項](#)
- [パスワード検証関数](#)

11.1.1.1 トランザクション・サイズの考慮事項

SQL Applyでは、トランザクションが大小2つのクラスに分類されます。

各クラスの定義は次のとおりです。

- 小さいトランザクション: SQL Applyは、REDOログ・ファイル内で小さいトランザクションのコミット・レコードを検出すると、そのトランザクションに属するLCRの適用を開始します。

- 大きいトランザクション: SQL Applyは、大きいトランザクションをトランザクション・チャンクと呼ばれる小さい単位に分割し、そのトランザクションのコミット・レコードをREDOログ・ファイル内で検出する前に、チャンクの適用を開始します。この処理が実行されるのは、LCRキャッシュのメモリー使用量を削減し、フェイルオーバー時間全体を短縮するためです。

たとえば、小さく分割しない場合、SQL*Loaderのロードが1000万行で、サイズがそれぞれ100バイトであれば、LCRキャッシュでは1GBを超えるメモリーが使用されます。LCRキャッシュに割り当てられているメモリーが1GB未満の場合は、LCRキャッシュからのページアウトが発生します。

メモリーの考慮事項とは別に、SQL ApplyがトランザクションのCOMMITレコードを検出するまでに1000万行のSQL*Loaderロードに関連する変更の適用を開始しない場合、ロールの推移が停止します。SQL Applyがロジカル・スタンバイ・データベース上でトランザクションを適用し終わるまで、そのトランザクションのコミット後に開始されたスイッチオーバーまたはフェイルオーバーは終了できません。

トランザクション・チャンクの使用にもかかわらず、800万行超を変更するトランザクションの処理時にSQL Applyのパフォーマンスが低下することがあります。800万行を超えるトランザクションでは、SQL Applyは一時セグメントを使用して、トランザクションの処理に必要な一部の内部メタデータを処理します。800万行を超えるトランザクションを正常に処理するには、SQL Applyで使用される一時セグメントに十分なスペースを割り当てます。

すべてのトランザクションは、最初は小さいトランザクションとして分類されます。SQL Applyでは、トランザクションが大きいトランザクションとして再分類されるタイミングは、LCRキャッシュに使用可能なメモリー量と、トランザクションに属するLCRのメモリー使用量に応じて決定します。

11.1.1.2 ページアウトの考慮事項

LCRキャッシュのメモリーがすべて使用され、SQL Applyを進行させるために領域の解放が必要になると、SQL Applyのコンテキスト内でページアウトが発生します。

たとえば、LCRキャッシュに100MBのメモリーが割り当てられている場合に、SQL Applyでサイズが300MBのLONG列を含む表に対するINSERTトランザクションが検出されるとします。この場合、ログ・マイニング・コンポーネントは、LONGデータの最初の部分をページアウトして列変更の後半部分を読み取ります。適切にチューニングされているロジカル・スタンバイ・データベースの場合、ページアウト・アクティビティはほとんど発生せず、システム全体のスループットには影響しません。

関連項目:

問題のあるページアウトを識別して修正処理を実行する方法の詳細は、[「ロジカル・スタンバイ・データベースのカスタマイズ」](#)を参照してください。

11.1.1.3 再開の考慮事項

トランザクションのコミット・レコードがREDOログ・ファイルからマイニングされてロジカル・スタンバイ・データベースに適用されるまで、ロジカル・スタンバイ・データベースに対する変更は永続的な変更になりません。

つまり、ユーザー指示の結果であるかシステム障害の結果であるかに関係なく、SQL Applyは停止されるたびに、コミットされていない最も古いトランザクションまで遡って再びマイニングする必要があります。

作業量は少ないが、長期間開いたままになるトランザクションの場合、SQL Applyを最初からやりなおすとSQL Applyはコミットされていない少量のトランザクションに対してREDOデータを再度読み込むために、アーカイブされた大量のREDOログ・ファイルを再度マイニングしなければならないので非常に効率が悪くなります。これを回避するために、SQL Applyはコミットされていない古いデータに対して、定期的にチェックポイントを行っています。チェックポイントを実施するSCNは、V\$LOGSTDBY_PROGRESSビューのRESTART_SCN列に反映されます。再開後、SQL ApplyはRESTART_SCN列に表示された値よりも大きいSCNで生成さ

れたREDOレコードのマイニングを開始します。SQL Applyは、アーカイブされたREDOログ・ファイルのうち再開が不要なものを自動的に削除します。

大量のDDLトランザクション、パラレルDML文(PDML)、および直接パス・ロードなどの特定のワークロードによっては、ワークロードが完了するまで、RESTART_SCNが中断します。

11.1.1.4 DML適用の考慮事項

SQL Applyがロジカル・スタンバイ・データベースのスループットと待機時間に影響するDMLトランザクションを適用する場合、次の特性があります。

- 1つの文で複数の行が変更されるバッチ更新または削除をプライマリ・データベースで実行した場合、ロジカル・スタンバイ・データベースでは行の個別変更として適用されます。そのため、メンテナンスされている各表に一意索引または主キーが必要になります。詳細は、[「プライマリ・データベース内の表の行が一意に識別できることの確認」](#)を参照してください。
- プライマリ・データベースで実行されたダイレクト・パス・インサートは、ロジカル・スタンバイ・データベースで従来型のINSERT文を使用して適用されます。
- パラレルDML(PDML)トランザクションは、ロジカル・スタンバイ・データベースではパラレルに実行されません。

11.1.1.5 DDL適用の考慮事項

SQL Applyがロジカル・スタンバイ・データベースのスループットと待機時間に影響するDDLトランザクションを適用する場合、次の特性があります。

- DDLトランザクションは、ロジカル・スタンバイ・データベースではシリアルに適用されます。そのため、プライマリ・データベースで同時に適用されたDDLトランザクションは、ロジカル・スタンバイ・データベースでは一度に1つずつ適用されます。
- ロジカル・スタンバイ・データベース上でDMLアクティビティ(CREATE TABLE AS SELECT(CTAS)文の一部)が抑制されるように、CTAS文が実行されます。CTAS文の一部として新規作成された表に挿入された行は、REDOログ・ファイルからマイニングされ、INSERT文を使用してロジカル・スタンバイ・データベースに適用されます。
- SQL Applyは、プライマリ・データベースで実行されたDDLを再発行し、DDL操作のターゲットである同じオブジェクトで、同じトランザクション内で発生するDMLがロジカル・スタンバイ・データベースでレプリケートされないようにします。そのため、次の2つの場合にプライマリ・サイトとスタンバイ・サイトは互いに異なります。
 - DDLに、プライマリ・データベースの状態から導出される非リテラル値が含まれる場合。このようなDDLの例を次に示します。

```
ALTER TABLE hr.employees ADD (start_date date default sysdate);
```

SQL Applyはロジカル・スタンバイで同じDDLを再発行するため、sysdate()ファンクションはロジカル・スタンバイで再評価されます。そのため、start_date列は、プライマリ・データベースでのデフォルト値とは異なるデフォルト値で作成されます。

- DDLで、ターゲット表で定義されているDMLトリガーを起動する場合。トリガーされるDMLはDDLと同じトランザクションで発生し、DDLのターゲット表で動作するため、これらのトリガーされるDMLは、ロジカル・スタンバイでレプリケートされません。

たとえば、次のような表を作成するとします。

```
create table HR.TEMP_EMPLOYEES (  
  emp_id      number primary key,  
  first_name  varchar2(64),  
  last_name   varchar2(64),
```

```
modify_date timestamp);
```

次に、表にトリガーを作成して、表が更新されるたびに、modify_dateを更新して変更時間が反映されるようにします。

```
CREATE OR REPLACE TRIGGER TRG_TEST_MOD_DT BEFORE UPDATE ON HR.TEST_EMPLOYEES
REFERENCING
NEW AS NEW_ROW FOR EACH ROW
BEGIN
:NEW_ROW.MODIFY_DATE:= SYSTIMESTAMP;
END;
/
```

通常のDML/DDLワークロードではこの表は正しく維持されます。ただし、この表にデフォルト値を含む列を追加すると、ADD COLUMN DDLがこの更新トリガーを起動し、表内のすべての行のMODIFY_DATE列を新しいタイムスタンプに変更します。MODIFY_DATE列へのこの種の変更は、ロジカル・スタンバイ・データベースにはレプリケートされません。REDOストリーム内に記録されたMODIFY_DATE列データがロジカル・スタンバイ・データベース内のデータと一致しないため、この表に対してこれ以降DMLを行うと、SQL Applyが停止します。

11.1.1.6 パスワード検証関数

パスワードの複雑性をチェックするパスワード検証関数は、SYSスキーマで作成する必要があります。

SQL ApplyはSYSスキーマで作成されたオブジェクトをレプリケートしないため、このような検証関数は、ロジカル・スタンバイ・データベースにレプリケートされません。ロジカル・スタンバイ・データベースでは、パスワード検証関数を手動で作成し、適切なプロファイルと関連付ける必要があります。

11.2 ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスの制御

SQLのALTER DATABASE GUARD文は、ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスを制御します。

デフォルトでは、ロジカル・スタンバイ・データベースについてはデータベース・ガードがALLに設定されています。

ALTER DATABASE GUARD文では、次のキーワードを使用できます。

- ALL

ALLを指定すると、SYS以外のすべてのユーザーが、ロジカル・スタンバイ・データベース内のデータを変更できなくなります。

- STANDBY

STANDBYを指定すると、SYS以外のすべてのユーザーが、SQL Applyでメンテナンスされる表または順序に対してDMLおよびDDLを変更できなくなります。

- なし

NONEを指定すると、データベース内の全データには通常のセキュリティが適用されます。

たとえば、SQL Applyによってメンテナンスされない表をユーザーが変更できるようにするには、次の文を使用します。

```
SQL> ALTER DATABASE GUARD STANDBY;
```

権限を持つユーザーは、ALTER SESSION DISABLE GUARDおよびALTER SESSION ENABLE GUARD文をそれぞれ使用して、現在のセッションでデータベース・ガードを一時的にオフまたはオンにできます。この文は、Oracle9iと同じ機能を実行するDBMS_LOGSTDBY.GUARD_BYPASS PL/SQLプロシージャを配置します。ALTER SESSION [ENABLE|DISABLE] GUARD文は、[「ロジカル・スタンバイ・データベースの変更」](#)に説明されているように、データベース・ガードを一時的に無効にしてデータベースを

変更するときに役立ちます。

ノート:



データベース・ガードが無効のときに、プライマリおよびロジカル・スタンバイ・データベースの内容に相違を生じさせないようにしてください。

11.3 ロジカル・スタンバイ・データベースの管理および監視関連のビュー

パフォーマンス・ビューを使用して、ロジカル・スタンバイ・データベースを保守するSQL Applyの動作を監視できます。

次の各トピックでは、ロジカル・スタンバイ・データベースの監視に使用できる主なビューについて説明します。

- [DBA_LOGSTDBY_EVENTSビュー](#)
- [DBA_LOGSTDBY_LOGビュー](#)
- [V\\$DATAGUARD_STATSビュー](#)
- [V\\$LOGSTDBY_PROCESSビュー](#)
- [V\\$LOGSTDBY_PROGRESSビュー](#)
- [V\\$LOGSTDBY_STATEビュー](#)
- [V\\$LOGSTDBY_STATSビュー](#)

関連項目:

ビューの詳細な参照情報は、『[Oracle Databaseリファレンス](#)』を参照してください。

11.3.1 DBA_LOGSTDBY_EVENTSビュー

DBA_LOGSTDBY_EVENTSビューには、SQL Apply操作中に発生した重要なイベントが記録されます。

デフォルトでは、最新の10,000のイベントが記録されます。ただし、記録されるイベントの数は、PL/SQLプロシージャDBMS_LOGSTDBY.APPLY_SET()をコールして変更できます。SQL Applyが突然停止した場合は、このビューにその原因も記録されます。

ノート:



SQL Apply を停止させたエラーはイベント・テーブル内に記録されます。このイベントは ALERT.LOG ファイルにも記録され、テキスト内に LOGSTDBY キーワードが含まれます。ビューの問合せの際に、EVENT_TIME_STAMP、COMMIT_SCN、および CURRENT_SCN によって列を順番に選択し、イベントの順序を正しく並べます。

このビューは、適用されたDDLトランザクションやスキップされたDDLトランザクションなどの他の情報も表示されるようにカスタマイズできます。次に例を示します。


```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YY HH24:MI:SS';
Session altered.
SQL> COLUMN STATUS FORMAT A60
SQL> SELECT EVENT_TIME, STATUS, EVENT FROM DBA_LOGSTDBY_EVENTS -
> ORDER BY EVENT_TIMESTAMP, COMMIT_SCN, CURRENT_SCN;

EVENT_TIME          STATUS
-----
EVENT
-----
23-JUL-02 18:20:12 ORA-16111: log mining and apply setting up
23-JUL-02 18:25:12 ORA-16128: User initiated shut down successfully completed
23-JUL-02 18:27:12 ORA-16112: log mining and apply stopping
23-JUL-02 18:55:12 ORA-16128: User initiated shut down successfully completed
23-JUL-02 18:57:09 ORA-16111: log mining and apply setting up
23-JUL-02 20:21:47 ORA-16204: DDL successfully applied
create table hr.test_emp (empno number, ename varchar2(64))
23-JUL-02 20:22:55 ORA-16205: DDL skipped due to skip setting
create database link link_to_boston connect to system identified by change_on_inst
7 rows selected.
```

この問合せによって、SQL Applyが数回開始および停止したことがわかります。またDDLが適用され、スキップされたこともわかります。

11.3.2 DBA_LOGSTDBY_LOGビュー

DBA_LOGSTDBY_LOGビューには、SQL Applyで処理中のアーカイブ・ログに関する動的な情報が表示されます。

次に例を示します。

```
SQL> COLUMN DICT_BEGIN FORMAT A10;
SQL> SET NUMF 99999999;
SQL> SELECT FILE_NAME, SEQUENCE# AS SEQ#, FIRST_CHANGE# AS F_SCN#, -
> NEXT_CHANGE# AS N_SCN#, TIMESTAMP, -
> DICT_BEGIN AS BEG, DICT_END AS END, -
> THREAD# AS THR#, APPLIED FROM DBA_LOGSTDBY_LOG -
> ORDER BY SEQUENCE#;

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

BEGおよびEND列のYESエントリは、LogMinerディクショナリのビルドがログ・ファイル順序番号5で開始したことを示しています。最新のアーカイブREDOログ・ファイルは順序番号13で、01:02:41にロジカル・スタンバイ・データベースが受信しました。

APPLIED列は、SQL ApplyがすべてのREDOをSCN 144057より前に適用したことを示しています。トランザクションが複数のアーカイブ・ログにまたがることもあるため、複数のアーカイブ・ログ・ファイルのAPPLIED列に、CURRENT値が現れることがあります。

11.3.3 V\$DATAGUARD_STATSビュー

V\$DATAGUARD_STATSビューには、ロジカル・スタンバイ・データベースのフェイルオーバー特性に関する情報が表示されます。

次の情報が含まれます。

- フェイルオーバー時間(apply finish time)

適用終了時間は、受信済で未適用のREDOをすべて適用するために必要な見積時間です。見積りは、REDOを最大適用率で適用できるという前提に基づいています。残りのすべてのREDOの適用にかかる実際の時間は、REDOのタイプおよびREDOの適用率によって異なります。

- ロジカル・スタンバイ・データベース内のコミット済データの最新性(apply lag)
- 障害時の潜在的なデータ損失(transport lag)

例11-1 ロジカル・スタンバイ・データベースのV\$DATAGUARD_STATSの内容

次の出力は、プライマリ・データベースから生成されたすべてのREDOを受信および適用したロジカル・スタンバイ・データベースからの出力です。

```
SQL> COL NAME FORMAT A20
SQL> COL VALUE FORMAT A12
SQL> COL UNIT FORMAT A30
SQL> SELECT NAME, VALUE, UNIT FROM V$DATAGUARD_STATS;
```

NAME	VALUE	UNIT
apply finish time	+00 00:00:00 day(2) to second(1) interval	
apply lag	+00 00:00:00 day(2) to second(0) interval	
transport lag	+00 00:00:00 day(2) to second(0) interval	

11.3.4 V\$LOGSTDBY_PROCESSビュー

このビューには、次のように、SQL Apply関連の各種プロセスの現在の状態に関する情報が表示されます。

- 識別情報(sid | serial# | spid)
- SQL Applyプロセス: COORDINATOR、READER、BUILDER、PREPARER、ANALYZERまたはAPPLIER(type)
- プロセスの現行のアクティビティのステータス(status_code | status)
- このプロセスで処理された最上位のREDOレコード(high_scn)

次に例を示します。

```
SQL> COLUMN SERIAL# FORMAT 9999
SQL> COLUMN SID FORMAT 9999
SQL> SELECT SID, SERIAL#, SPID, TYPE, HIGH_SCN FROM V$LOGSTDBY_PROCESS;
```

SID	SERIAL#	SPID	TYPE	HIGH_SCN
48	6	11074	COORDINATOR	7178242899
56	56	10858	READER	7178243497
46	1	10860	BUILDER	7178242901
45	1	10862	PREPARER	7178243295
37	1	10864	ANALYZER	7178242900
36	1	10866	APPLIER	7178239467
35	3	10868	APPLIER	7178239463
34	7	10870	APPLIER	7178239461
33	1	10872	APPLIER	7178239472

9 rows selected.

HIGH_SCN列は、READERプロセスが他のすべてのプロセスに先行することと、PREPARERおよびBUILDERプロセスが残りのプロセスに先行することを示しています。

```
SQL> COLUMN STATUS FORMAT A40
```

```
SQL> SELECT TYPE, STATUS_CODE, STATUS FROM V$LOGSTDBY_PROCESS;
```

TYPE	STATUS_CODE	STATUS
COORDINATOR	16117	ORA-16117: processing
READER	16127	ORA-16127: stalled waiting for additional transactions to be applied
BUILDER	16116	ORA-16116: no work available
PREPARER	16116	ORA-16117: processing
ANALYZER	16120	ORA-16120: dependencies being computed for transaction at SCN 0x0001.abdb440a
APPLIER	16124	ORA-16124: transaction 1 13 1427 is waiting on another transaction
APPLIER	16121	ORA-16121: applying transaction with commit SCN 0x0001.abdb4390
APPLIER	16123	ORA-16123: transaction 1 23 1231 is waiting for commit approval
APPLIER	16116	ORA-16116: no work available

出力は、実行中のSQL Applyのスナップショットを示しています。マイニング側では、READERプロセスはさらに読み取れるように追加のメモリーが使用可能になるまで待機中で、PREPARERプロセスはREDOLレコードの処理中、BUILDERプロセスに使用可能な作業はありません。適用側では、COORDINATORはAPPLIERプロセスにさらにトランザクションを割当中で、ANALYZERはSCN 7178241034で依存性を計算中、APPLIERの1つは使用可能な作業がなく、2つにはまだ満たされていない未処理の依存性があります。

関連項目:

出力例は[「SQL Applyの進捗の監視」](#)を参照してください。

11.3.5 V\$LOGSTDBY_PROGRESSビュー

このビューには、次のように、SQL Applyの進捗に関する詳細情報が表示されます。

- プライマリ・データベース上でコミットされた全トランザクションがロジカル・スタンバイ・データベースに適用されたSCNおよび時刻(applied_scn, applied_time)
- 再開時にSQL ApplyによるREDOLレコードの読取りが開始されるSCNおよび時刻(restart_scn, restart_time)
- ロジカル・スタンバイ・データベースで受信された最新REDOLレコードのSCNおよび時刻(latest_scn, latest_time)
- BUILDERプロセスにより処理された最新レコードのSCNおよび時刻(mining_scn, mining_time)

次に例を示します。

```
SQL> SELECT APPLIED_SCN, LATEST_SCN, MINING_SCN, RESTART_SCN -  
> FROM V$LOGSTDBY_PROGRESS;
```

APPLIED_SCN	LATEST_SCN	MINING_SCN	RESTART_SCN

この出力は次のことを示しています。

- SQL Applyにより、SCN 7178240496以前にコミットされた全トランザクションが適用されました。
- ロジカル・スタンバイ・データベースで受信された最新REDOレコードは、SCN 7178240507で生成されました。
- マイニング・コンポーネントにより、SCN 7178240507以前に生成された全REDOレコードが処理されました。
- SQL Applyがなんらかの理由で停止されて再開されると、SCN 7178219805以降に生成されたREDOレコードのマイニングが開始されます。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='yy-mm-dd hh24:mi:ss';
Session altered
```

```
SQL> SELECT APPLIED_TIME, LATEST_TIME, MINING_TIME, RESTART_TIME -
> FROM V$LOGSTDBY_PROGRESS;
```

APPLIED_TIME	LATEST_TIME	MINING_TIME	RESTART_TIME
05-05-12 10:38:21	05-05-12 10:41:53	05-05-12 10:41:21	05-05-12 10:09:30

この出力は次のことを示しています。

- SQL Applyにより、時刻05-05-12 10:38:21以前にコミットされた全トランザクションが適用されました (APPLIED_TIME)。
- 最後のREDOは、プライマリ・データベースで時刻05-05-12 10:41:53に生成されました(LATEST_TIME)。
- マイニング・エンジンにより、05-05-12 10:41:21以前に生成された全REDOレコードが処理されました (MINING_TIME)。
- SQL Applyの再開時には、05-05-12 10:09:30以後に生成されたREDOレコードのマイニングが開始されます。

関連項目:

出力例は[「SQL Applyの進捗の監視」](#)を参照してください。

11.3.6 V\$LOGSTDBY_STATEビュー

このビューには、次のように、SQL Applyの現在の状態のシノプシスが表示されます。

- プライマリ・データベースのDBID(primary_dbid)
- SQL Applyに割り当てられているLogMinerセッションID(session_id)
- SQL Applyがリアルタイムで適用中かどうか(realtime_apply)

次に例を示します。

```
SQL> COLUMN REALTIME_APPLY FORMAT a15
SQL> COLUMN STATE FORMAT a16
SQL> SELECT * FROM V$LOGSTDBY_STATE;
```

PRIMARY_DBID	SESSION_ID	REALTIME_APPLY	STATE
1562626987	1	Y	APPLYING

この出力は、SQL Applyがリアルタイム適用モードで実行中で、現在はプライマリ・データベースから受信したREDOデータを適用中であり、プライマリ・データベースのDBIDが1562626987で、SQL Applyセッションに関連付けられているLogMinerセッションIDが1であることを示しています。

関連項目:

出力例は[「SQL Applyの進捗の監視」](#)を参照してください。

11.3.7 V\$LOGSTDBY_STATSビュー

V\$LOGSTDBY_STATSビューには、SQL Applyに関連する統計、現行の状態およびステータス情報が表示されます。SQL Applyが実行されていない場合、このビューからは1行も戻されません。このビューは、ロジカル・スタンバイ・データベースの下でのみ意味があります。

次に例を示します。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT=' dd-mm-yyyy hh24:mi:ss' ;
Session altered
```

```
SQL> SELECT SUBSTR(name, 1, 40) AS NAME, SUBSTR(value,1,32) AS VALUE FROM V$LOGSTDBY_STATS;
```

NAME	VALUE
logminer session id	1
number of preparers	1
number of appliers	5
server processes in use	9
maximum SGA for LCR cache (MB)	30
maximum events recorded	10000
preserve commit order	TRUE
transaction consistency	FULL
record skipped errors	Y
record skipped DDLs	Y
record applied DDLs	N
record unsupported operations	N
realtime apply	Y
apply delay (minutes)	0
coordinator state	APPLYING
coordinator startup time	19-06-2007 09:55:47
coordinator uptime (seconds)	3593
txns received from logminer	56
txns assigned to apply	23
txns applied	22
txns discarded during restart	33
large txns waiting to be assigned	2
rolled back txns mined	4
DDL txns mined	40
CTAS txns mined	0
bytes of redo mined	60164040
bytes paged out	0
pageout time (seconds)	0
bytes checkpointed	4845
checkpoint time (seconds)	0
system idle time (seconds)	2921
standby redo logs mined	0

```

archived logs mined          5
gap fetched logs mined       0
standby redo log reuse detected 1
logfile open failures        0
current logfile wait (seconds) 0
total logfile wait (seconds) 2910
thread enable mined          0
thread disable mined         0
.
40 rows selected.

```

11.4 ロジカル・スタンバイ・データベースの監視

ロジカル・スタンバイ・データベースを使用するときは、SQL Applyの進捗を監視できます。また、ログ・ファイルを自動的に削除することもできます。

次のトピックを参照してください。

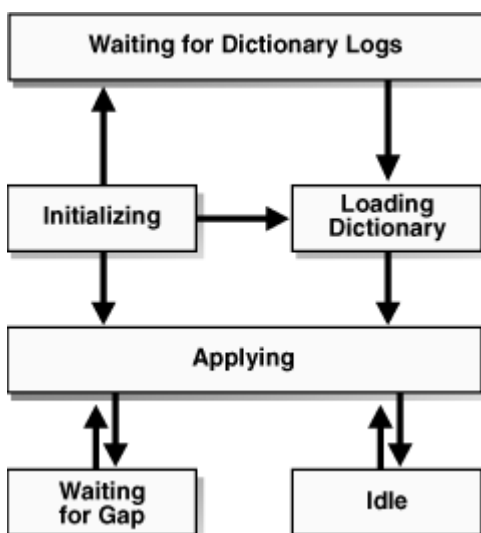
- [SQL Applyの進捗の監視](#)
- [ログ・ファイルの自動削除](#)

11.4.1 SQL Applyの進捗の監視

SQL Applyには、SQL Applyの初期化、ディクショナリ・ログの待機、LogMinerディクショナリのロード、(REDOデータの)適用、アーカイブのギャップ解消待ち、およびアイドルという6つの進捗状態があります。

[図11-2](#)はこれらの状態のフローを示しています。

図11-2 SQL Apply処理中の進捗状態



次の各項では、それぞれの状態を詳細に説明します。

初期化中状態

ALTER DATABASE START LOGICAL STANDBY APPLY文を発行してSQL Applyを開始すると、初期化中状態になります。

SQL Applyの現在の状態を判断するには、V\$LOGSTDBY_STATEビューを問い合わせます。次に例を示します。

```
SQL> SELECT SESSION_ID, STATE FROM V$LOGSTDBY_STATE;
```

SESSION_ID	STATE
1	INITIALIZING

SESSION_ID列は、SQL Applyにより作成された永続LogMinerセッションを示します。このセッションで、プライマリ・データベースで生成されたアーカイブREDOログ・ファイルがマイニングされます。

ディクショナリ・ログ待機中

SQL Applyは、初回の開始時に、REDOログ・ファイルで取得されたLogMinerディクショナリをロードする必要があります。SQL Applyは、LogMinerディクショナリのロードに必要なREDOデータをすべて受信するまで、WAITING FOR DICTIONARY LOGS状態にとどまります。

ディクショナリ・ロード中状態

ディクショナリ・ロード中状態は、しばらく継続する場合があります。大規模データベースでLogMinerディクショナリをロードするには、長時間かかることがあります。V\$LOGSTDBY_STATEビューを問い合わせると、ディクショナリのロード中は次の出力が戻されます。

```
SQL> SELECT SESSION_ID, STATE FROM V$LOGSTDBY_STATE;
```

SESSION_ID	STATE
1	LOADING DICTIONARY

LogMinerディクショナリが完全にロードされるまでに起動されるのは、COORDINATORプロセスとマイニング・プロセスのみです。したがって、この時点でV\$LOGSTDBY_PROCESSを問い合わせると、APPLIERプロセスは表示されません。次に例を示します。

```
SQL> SELECT SID, SERIAL#, SPID, TYPE FROM V$LOGSTDBY_PROCESS;
```

SID	SERIAL#	SPID	TYPE
47	3	11438	COORDINATOR
50	7	11334	READER
45	1	11336	BUILDER
44	2	11338	PREPARER
43	2	11340	PREPARER

V\$LOGMNR_DICTIONARY_LOADビューを問い合わせると、ディクショナリ・ロードの進捗に関する詳細情報を取得できます。ディクショナリ・ロードは、次の3つのフェーズで発生します。

1. LogMinerディクショナリのロードに関連するREDO変更を収集するために、関連アーカイブREDOログ・ファイルまたはスタンバイREDOログ・ファイルがマイニングされます。
2. 変更が処理され、データベース内のステー징表にロードされます。
3. 一連のDDL文が発行されて、LogMinerディクショナリ表がロードされます。

次に例を示します。

```
SQL> SELECT PERCENT_DONE, COMMAND -  
> FROM V$LOGMNR_DICTIONARY_LOAD -  
> WHERE SESSION_ID = (SELECT SESSION_ID FROM V$LOGSTDBY_STATE);
```

PERCENT_DONE	COMMAND
40	alter table SYSTEM.LOGMNR_CCOL\$ exchange partition P101 with table SYS.LOGMNRILT_101_CCOL\$ excluding indexes without validation

PERCENT_DONEまたはCOMMAND列が長時間変化しない場合は、V\$SESSION_LONGOPSビューを問い合わせ、問題のDDLトランザクションの進捗を監視してください。

適用中状態

この状態では、SQL ApplyはLogMinerディクショナリの初期スナップショットを正常にロードしており、現在はロジカル・スタンバイ・データベースにREDOデータを適用中です。

SQL Applyの進捗の詳細情報を確認するには、次のようにV\$LOGSTDBY_PROGRESSビューを問い合わせます。

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SQL> SELECT APPLIED_TIME, APPLIED_SCN, MINING_TIME, MINING_SCN -
> FROM V$LOGSTDBY_PROGRESS;
```

APPLIED_TIME	APPLIED_SCN	MINING_TIME	MINING_SCN
10-JAN-2005 12:00:05	346791023	10-JAN-2005 12:10:05	3468810134

プライマリ・データベース上でAPPLIED_SCN(またはAPPLIED_TIME)以前に表示されるコミット済トランザクションは、すべてロジカル・スタンバイ・データベースに適用されています。マイニング・エンジンは、プライマリ・データベースでMINING_SCN(およびMINING_TIME)以前に生成されたREDOレコードをすべて処理しています。安定状態では、MINING_SCN(およびMINING_TIME)は常にAPPLIED_SCN(およびAPPLIED_TIME)よりも前の値を示します。

ギャップ待機中状態

この状態が発生するのは、SQL Applyが使用可能な全REDOレコードのマイニングと適用を完了し、RFSプロセスによる新規ログ・ファイル(または欠落しているログ・ファイル)のアーカイブを待機している場合です。

```
SQL> SELECT STATUS FROM V$LOGSTDBY_PROCESS WHERE TYPE = 'READER';
```

STATUS

ORA-16240: waiting for log file (thread# 1, sequence# 99)

アイドル状態

SQL Applyは、プライマリ・データベースで生成されたREDOをすべて適用し終わると、この状態になります。

11.4.2 ログ・ファイルの自動削除

外部アーカイブ・ログには、プライマリ・データベースから送信されたREDOが含まれています。

外部アーカイブ・ログの格納には次の2つの方法があります。

- 高速リカバリ領域に格納
- 高速リカバリ領域外のディレクトリに格納

高速リカバリ領域内に格納される外部アーカイブ・ログは、必ずSQL Applyによって管理されます。ログに含まれるすべてのREDOレコードは、ロジカル・スタンバイ・データベースでの適用後、DB_FLASHBACK_RETENTION_TARGETパラメータで指定された期間(DB_FLASHBACK_RETENTION_TARGETが指定されていない場合は1440分間)保持されます。高速リカバリ領域内に格納されている外部アーカイブ・ログの自動管理は上書きできません。

高速リカバリ領域内に格納されない外部アーカイブ・ログは、デフォルトではSQL Applyによって管理されます。自動管理下では、高速リカバリ領域内に格納されない外部アーカイブ・ログは、ログに含まれるすべてのREDOレコードがロジカル・スタンバイ・データベースで適用された後、LOG_AUTO_DEL_RETENTION_TARGETパラメータで指定された期間保持されます。高速リカバリ領域内に格納されない外部アーカイブ・ログの自動管理は、次のPL/SQLプロシージャを実行すると上書きできます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE', 'FALSE');
```

ノート:



DBMS_LOGSTDBY.APPLY_SET プロシーダを使用してこのパラメータを設定します。

LOG_AUTO_DEL_RETENTION_TARGET を明示的に指定しなかった場合、DB_FLASHBACK_RETENTION_TARGET がロジカル・スタンバイ・データベースにデフォルト設定され、DB_FLASHBACK_RETENTION_TARGET が設定されていない場合は、デフォルトの 1440 分に設定されます。

デフォルトの自動ログ削除機能を上書きする場合は、定期的に次のステップを実行してSQL Applyで不要になったアーカイブ REDO ログ・ファイルを識別し、削除します。

1. 不要になったメタデータのロジカル・スタンバイ・セッションを消去するには、次のPL/SQL文を入力します。

```
SQL> EXECUTE DBMS_LOGSTDBY.PURGE_SESSION;
```

この文では、不要になったアーカイブ REDO ログ・ファイルを表示するDBA_LOGMNR_PURGED_LOGビューも更新されます。

2. DBA_LOGMNR_PURGED_LOGビューを問い合わせ、削除できるアーカイブ REDO ログ・ファイルのリストを表示します。

```
SQL> SELECT * FROM DBA_LOGMNR_PURGED_LOG;
```

FILE_NAME

```
-----  
/boston/arc_dest/arc_1_40_509538672.log  
/boston/arc_dest/arc_1_41_509538672.log  
/boston/arc_dest/arc_1_42_509538672.log  
/boston/arc_dest/arc_1_43_509538672.log  
/boston/arc_dest/arc_1_44_509538672.log  
/boston/arc_dest/arc_1_45_509538672.log  
/boston/arc_dest/arc_1_46_509538672.log  
/boston/arc_dest/arc_1_47_509538672.log
```

3. オペレーティング・システム固有のコマンドを使用して、問合せにより表示されたアーカイブ REDO ログ・ファイルを削除します。

11.5 ロジカル・スタンバイ・データベースのカスタマイズ

ロジカル・スタンバイ・データベースは、いくつかの方法でカスタマイズできます。これには、イベントのロギング、特定のスキーマ・オブジェクトに対する変更の防止、表の追加または再作成が含まれます。

次のトピックを参照してください。

- [DBA_LOGSTDBY_EVENTSビューでのイベントのロギングのカスタマイズ](#)
- [DBMS_LOGSTDBY.SKIPによる特定のスキーマ・オブジェクトに対する変更の防止](#)
- [DDL文のスキップ・ハンドラの設定](#)
- [ロジカル・スタンバイ・データベースの変更](#)
- [ロジカル・スタンバイ・データベースでの表の追加または再作成](#)

関連項目:

[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_LOGSTDBYパッケージ](#)

11.5.1 DBA_LOGSTDBY_EVENTSビューでのイベントのロギングのカスタマイズ

DBA_LOGSTDBY_EVENTSビューは、SQL Applyの下で発生する重要な最新イベントを含む循環型ログとみなすことができます。

デフォルトでは、最新の10,000イベントがイベント・ビューに記録されます。ログに記録されるイベント数は、DBMS_LOGSTDBY.APPLY_SETプロシージャを起動して変更できます。たとえば、最新の100,000イベントを確実に記録するには、次の文を発行できます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET (' MAX_EVENTS_RECORDED', ' 100000');
```

SQL Applyの停止原因となったエラーは、SYSTEM表領域に十分な領域があるかぎり、必ずDBA_LOGSTDBY_EVENTSビューに記録されます。これらのイベントは常にアラート・ファイルにも格納され、テキストにはLOGSTDBYというキーワードが含まれます。このビューを問い合わせるときは、EVENT_TIME、COMMIT_SCN、CURRENT_SCNの順に列を選択してください。この順序によって、停止障害がビューの最後に表示されます。

次の例では、ビューに記録されるイベントを指定するDBMS_LOGSTDBYサブプログラムを示します。

例1: DDL文が適用されたかどうかを判断する

たとえば、適用済DDLトランザクションをDBA_LOGSTDBY_EVENTSビューに記録するには、次の文を発行します。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET (' RECORD_APPLIED_DDL', ' TRUE');
```

例2: サポートされない操作がないかDBA_LOGSTDBY_EVENTSビューをチェックする

プライマリ・データベースで実行されていて、ロジカル・スタンバイ・データベースでサポートされないトランザクションに関する情報を取得するには、次の文を発行します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;SQL> EXEC  
DBMS_LOGSTDBY.APPLY_SET(' RECORD_UNSUPPORTED_OPERATIONS', ' TRUE');SQL> ALTER DATABASE START LOGICAL  
STANDBY APPLY IMMEDIATE;
```

次に、サポートされない操作がないかDBA_LOGSTDBY_EVENTSビューをチェックします。通常、サポートされない表に対する操作は、SQL Applyによって警告なしで無視されます。しかし、ローリング・アップグレード中(スタンバイ・データベースが上位バージョンであり、下位バージョンのプライマリ・データベースによって生成されたREDOをマイニングしているとき)に、サポートされない操作をプライマリ・データベースに対して実行した場合、ロジカル・スタンバイ・データベースにスイッチオーバーされない可能性があります。Oracle Data Guardは、表ごとに1つ以上のサポートされない操作をDBA_LOGSTDBY_EVENTSビューに記録します。ローリング・アップグレードに関する詳細は、[「SQL Applyを使用したOracle Databaseのアップグレード」](#)を参照してください。

11.5.2 DBMS_LOGSTDBY.SKIPによる特定のスキーマ・オブジェクトに対する変更の防止

デフォルトでは、プライマリ・データベース内でサポートされる表は、すべてロジカル・スタンバイ・データベースで複製されます。

このデフォルト動作は、特定の表に対する変更の適用をスキップするルールを指定することで変更できます。たとえば、HR.EMPLOYEES表に対する変更を省略するには、特定の表に対するDML変更とDDL変更の適用を防止するルールを指定できます。次に例を示します。

1. SQL Applyを停止します。

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

2. SKIPルールを登録します。

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP (stmt => 'DML', schema_name => 'HR', -
> object_name => 'EMPLOYEES');

SQL> EXECUTE DBMS_LOGSTDBY.SKIP (stmt => 'SCHEMA_DDL', schema_name => 'HR', -
> object_name => 'EMPLOYEES');
```

3. SQL Applyを開始します。

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

11.5.3 DDL文のスキップ・ハンドラの設定

特定のDDL文をインターセプトして元のDDL文を別のDDL文で置き換えるプロシージャを作成できます。

たとえば、ロジカル・スタンバイ・データベース内のファイル・システムの編成がプライマリ・データベースとは異なる場合、DBMS_LOGSTDBY.SKIPプロシージャを作成し、ファイルを指定してDDLトランザクションを透過的に処理できます。

次のプロシージャでは、ファイル指定文字列に特定のネーミング規則を使用するかぎり、プライマリ・データベースとスタンバイ・データベース間で異なるファイル・システム編成を処理できます。

1. 次のように、表領域のDDLトランザクションを処理するスキップ・プロシージャを作成します。

```
CREATE OR REPLACE PROCEDURE SYS.HANDLE_TBS_DDL (
  OLD_STMT  IN  VARCHAR2,
  STMT_TYP  IN  VARCHAR2,
  SCHEMA    IN  VARCHAR2,
  NAME      IN  VARCHAR2,
  XIDUSN    IN  NUMBER,
  XIDSLT    IN  NUMBER,
  XIDSQN    IN  NUMBER,
  ACTION    OUT NUMBER,
  NEW_STMT  OUT VARCHAR2
) AS
BEGIN

  -- All primary file specification that contains a directory
  -- /usr/orcl/primary/dbs
  -- should go to /usr/orcl/stdby directory specification

  NEW_STMT := REPLACE(OLD_STMT,
                      '/usr/orcl/primary/dbs',
                      '/usr/orcl/stdby');

  ACTION := DBMS_LOGSTDBY.SKIP_ACTION_REPLACE;

EXCEPTION
  WHEN OTHERS THEN
    ACTION := DBMS_LOGSTDBY.SKIP_ACTION_ERROR;
    NEW_STMT := NULL;
END HANDLE_TBS_DDL;
```

2. SQL Applyを停止します。

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

3. スキップ・プロシージャをSQL Applyに登録します。

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP (stmt => 'TABLESPACE', -
> proc_name => 'sys.handle_tbs_ddl');
```

4. SQL Applyを開始します。

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

11.5.4 ロジカル・スタンバイ・データベースの変更

ロジカル・スタンバイ・データベースは、SQL文が適用されている間でもレポート・アクティビティに使用できます。

データベース・ガードはロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスを制御し、ALTER SESSION DISABLE GUARD文はデータベース・ガードをバイパスしてロジカル・スタンバイ・データベース内の表に対する変更を可能にするために使用します。

ノート:

ロジカル・スタンバイ・データベースを使用して、独自に他の表を作成すると同時にプライマリ・データベースからレプリケートされるデータを処理する他のアプリケーションをホスト管理するには、データベース・ガードを STANDBY に設定する必要があります。このようなアプリケーションがシームレスに動作するには、PRESERVE_COMMIT_ORDER を TRUE (SQL Apply のデフォルト設定) に設定して実行する必要があります。(DBMS_LOGSTDBY PL/SQL パッケージの PRESERVE_COMMIT_ORDER パラメータの詳細は、[『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』](#)を参照してください。)

次の SQL 文を発行して、データベース・ガードを STANDBY に設定します。

```
SQL> ALTER DATABASE GUARD STANDBY;
```

このガード設定では、プライマリ・データベースからレプリケートされる表は、ユーザーによる変更から保護されますが、スタンバイ・データベースに作成された表は、ロジカル・スタンバイで実行されているアプリケーションで変更できます。

デフォルトでは、ロジカル・スタンバイ・データベースはデータベース・ガードが ALL (最も限定的な設定) に設定されている状態で動作し、ユーザーはデータベースに対して変更を実行できません。データベース・ガードを無視して、ロジカル・スタンバイ・データベースを変更可能にするには、ALTER SESSION DISABLE GUARD文を実行します。権限を付与されているユーザーは、この文を発行して、現行のセッションでデータベース・ガードをオフにできます。

次の各項で、例を示します。これらの例では、データベース・ガードが ALL または STANDBY に設定されていると仮定しています。

11.5.4.1 ロジカル・スタンバイ・データベースでのDDLの実行

SQL Applyを介してメンテナンスされている表に制約を追加できます。

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。SYSDBA、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ため、ロジカル・スタンバイ・データベースではDDL文を発行できません。

次の例は、SQL Applyを停止し、データベース・ガードをバイパスしてロジカル・スタンバイ・データベースでSQL文を実行した後、データベース・ガードを再び使用可能にする方法を示しています。この例では、部分一致の問合せを高速化するために、soundex索引がSCOTT. EMPのsurname列に追加されます。soundex索引をプライマリ・サーバーで保持すると、非常に高コストになる場合があります。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;  
Database altered.
```



```
SQL> ALTER SESSION DISABLE GUARD;
PL/SQL procedure successfully completed.

SQL> CREATE INDEX EMP_SOUNDINDEX ON SCOTT.EMP (SOUNDEX(ENAME));
Table altered.

SQL> ALTER SESSION ENABLE GUARD;
PL/SQL procedure successfully completed.

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
Database altered.

SQL> SELECT ENAME, MGR FROM SCOTT.EMP WHERE SOUNDEX(ENAME) = SOUNDEX(' CLARKE');
```

ENAME	MGR
CLARK	7839

オラクル社では、データベース・ガードのバイパスが使用可能になっている間は、SQL Applyにより保守される表に対してDML操作を実行しないことをお勧めします。そうすることで、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。

11.5.4.2 SQL Applyでメンテナンスされていない表の変更

場合によっては、レポート・アプリケーションを使用してサマリー結果を収集し、一時的に保存し、レポートの実行回数を追跡する必要があります。この種のアプリケーションの主な目的はレポート・アクティビティですが、アプリケーションはロジカル・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)操作を実行しなければならない場合があります。表の作成または削除が必要になることもあります。データがSQL Applyを介してメンテナンスされていない場合は、レポート生成操作でデータを変更できるように、データベース・ガードを設定できます。

次の設定を行う必要があります。

- DBMS_LOGSTDBY.SKIPプロシージャを実行して、アプリケーションによるデータの書き込みを可能にする、ロジカル・スタンバイ・データベース上の表のセットを指定します。スキップされた表は、SQL Applyでメンテナンスされません。
- スタンバイ表のみを保護するようにデータベース・ガードを設定します。

次の例では、レポートによって書き込みが行われる表がプライマリ・データベース上にあると仮定しています。

この例では、SQL Applyを停止し、表をスキップした後、SQL Applyを再開しています。この操作によって、レポート生成アプリケーションは、HRのTESTEMP%に書き込みます。表はSQL Applyでメンテナンスされなくなります。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;
Database altered.

SQL> EXECUTE DBMS_LOGSTDBY.SKIP(stmt => ' SCHEMA_DDL', -
    schema_name => ' HR', -
    object_name => ' TESTEMP%');
PL/SQL procedure successfully completed.

SQL> EXECUTE DBMS_LOGSTDBY.SKIP(' DML', ' HR', ' TESTEMP%');
PL/SQL procedure successfully completed.

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
Database altered.
```

SQL Applyでは、開始後に、新しく指定されてスキップ・ルールに追加された表について、スタンバイ・データベースのメタデータを

更新する必要があります。SQL Applyによりメタデータが更新される前に、新しくスキップされる表を変更しようとすると失敗します。追加したばかりのSKIPルールがSQL Applyで正常に考慮されたかどうかは、次の問合せを発行すると確認できます。

```
SQL> SELECT VALUE FROM SYSTEM.LOGSTDBY$PARAMETERS WHERE NAME = 'GUARD_STANDBY';
```

VALUE

Ready

VALUE列にReadyと表示される場合、SQL Applyではスキップされる表の関連メタデータがすべて正常に更新されており、その表を変更しても安全です。

関連項目:

[「ロジカル・スタンバイ・データベースがサポートしているDDL文」](#)、および『[Oracle Database PL/SQLパッケージおよびタイプ・リファレンス](#)』の[DBMS_LOGSTDBY](#)パッケージ

11.5.5 ロジカル・スタンバイ・データベースでの表の追加または再作成

リカバリ不可能な操作が発生した場合は、一般にDBMS_LOGSTDBY.INSTANTIATE_TABLEプロシージャを使用して表を作成しなおします。

また、このプロシージャを使用して、以前はスキップしていた表に対してSQL Applyを使用可能にすることもできます。

表を作成する前に、[「プライマリ・データベース内の表の行が一意に識別できることの確認」](#)に記載されている要件を満たす必要があります。その後、次のステップに従って表HR.EMPLOYEESを再作成し、SQL Applyを再開できます。この手順は、プライマリ・データベースにアクセスするデータベース・リンクBOSTONが定義済であることを前提としています。

次のリストは、表を再作成し、その表に対してSQL Applyを再開する方法を示しています。

1. SQL Applyを停止します。

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

2. DBA_LOGSTDBY_SKIPビューを問い合わせ、問題の表で操作がスキップされていないことを確認します。

```
SQL> SELECT * FROM DBA_LOGSTDBY_SKIP;
```

ERROR	STATEMENT_OPT	OWNER	NAME	PROC
N	SCHEMA_DDL	HR	EMPLOYEES	
N	DML	HR	EMPLOYEES	
N	SCHEMA_DDL	OE	TEST_ORDER	
N	DML	OE	TEST_ORDER	

ロジカル・スタンバイ・データベース上で再作成する表には、すでにスキップ・ルールが関連付けられているため、最初にそのルールを削除する必要があります。そのためには、DBMS_LOGSTDBY.UNSKIPプロシージャをコールします。次に例を示します。

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP(stmt => 'DML', -  
> schema_name => 'HR', -  
> object_name => 'EMPLOYEES');
```

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP(stmt => 'SCHEMA_DDL', -  
> schema_name => 'HR', -
```

```
> object_name => 'EMPLOYEES');
```

3. DBMS_LOGSTDBY. INSTANTIATE_TABLE プロシーダを使用し、ロジカル・スタンバイ・データベースの全データを使用して表 HR. EMPLOYEES を再作成します。次に例を示します。

```
SQL> EXECUTE DBMS_LOGSTDBY. INSTANTIATE_TABLE (schema_name => 'HR', -  
> table_name => 'EMPLOYEES', -  
> dblink => 'BOSTON');
```

4. SQL Apply を開始します。

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

関連項目:

DBMS_LOGSTDBY. UNSKIP および DBMS_LOGSTDBY. INSTANTIATE_TABLE プロシーダの詳細は、[『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』](#)を参照してください。

新規にインスタンス化された表とデータベースの残りの部分にまたがってビューの一貫性が得られるように、SQL Apply がプライマリ・データベースと一致するまで待機してから、この表を問い合わせます。これを行うには、以下のステップに従います。

1. プライマリ・データベースで、V\$DATABASE ビューを問い合わせて現在の SCN を判別します。

```
SQL> SELECT CURRENT_SCN FROM V$DATABASE@BOSTON;
```

```
CURRENT_SCN
```

```
-----  
345162788
```

2. 前の問合せで戻された CURRENT_SCN より前のコミット済のトランザクションがすべて、SQL Apply により適用されていることを確認します。

```
SQL> SELECT APPLIED_SCN FROM V$LOGSTDBY_PROGRESS;
```

```
APPLIED_SCN
```

```
-----  
345161345
```

この問合せで戻された APPLIED_SCN が最初の問合せで戻された CURRENT_SCN よりも大きい場合は、新規に再作成された表を問い合わせても安全です。

11.6 ロジカル・スタンバイ・データベースの下での特定のワークロードの管理

ロジカル・スタンバイの下で、特定のワークロードを管理できます。

次の各項を参照してください。

- [プライマリ・データベースへのトランスポータブル表領域のインポート](#)
- [マテリアライズド・ビューの使用](#)
- [ロジカル・スタンバイ・データベースでのトリガーと制約の処理方法](#)
- [サポートされていない表のレプリケートでのトリガーの使用](#)
- [プライマリでの Point-in-Time リカバリの実行によるリカバリ](#)

11.6.1 プライマリ・データベースへのトランスポータブル表領域のインポート

トランスポータブル表領域をプライマリ・データベースにインポートできます。

次のステップを実行します。

1. ロジカル・スタンバイ・データベースを変更できるように、ガード設定を無効にします。

```
SQL> ALTER DATABASE GUARD STANDBY;
```

2. ロジカル・スタンバイ・データベースで表領域をインポートします。
3. データベース・ガード設定を有効にします。

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

4. プライマリ・データベースで表領域をインポートします。

11.6.2 マテリアライズド・ビューの使用

ロジカル・スタンバイでは、マテリアライズド・ビューに関連した次のDDL文を自動的にスキップします。

たとえば、ロジカル・スタンバイでは、次の文がスキップされます。

- CREATE、ALTERまたはDROP MATERIALIZED VIEW
- CREATE、ALTERまたはDROP MATERIALIZED VIEW LOG

ロジカル・スタンバイ・データベースの作成後にプライマリ・データベースで作成、変更または削除された新規マテリアライズド・ビューは、ロジカル・スタンバイ・データベースには反映されません。ただし、ロジカル・スタンバイ・データベースの作成前にプライマリ・データベースで作成されたマテリアライズド・ビューは、ロジカル・スタンバイ・データベースに存在します。

ロジカル・スタンバイでは、他の種類の補助データ構造に加えて、ロジカル・スタンバイ・データベースでの新規マテリアライズド・ビューのローカルな作成およびメンテナンスをサポートしています。たとえば、オンライン・トランザクション処理(OLTP)システムでは、高度に正規化された表を更新パフォーマンスに頻繁に使用しますが、これらの表により、複雑な意思決定支援の問合せに対するレスポンス時間が長くなることがあります。ロジカル・スタンバイ・データベースでより効率的な問合せが作成できるように、次のようにして、レプリケートされたデータを非正規化するマテリアライズド・ビューを作成できます(これらの文を発行する前にユーザーSYSとして接続してください)。

```
SQL> ALTER SESSION DISABLE GUARD;
```

```
SQL> CREATE MATERIALIZED VIEW LOG ON SCOTT.EMP -  
> WITH ROWID (EMPNO, ENAME, MGR, DEPTNO) INCLUDING NEW VALUES;
```

```
SQL> CREATE MATERIALIZED VIEW LOG ON SCOTT.DEPT -  
> WITH ROWID (DEPTNO, DNAME) INCLUDING NEW VALUES;
```

```
SQL> CREATE MATERIALIZED VIEW SCOTT.MANAGED_BY -  
> REFRESH ON DEMAND -  
> ENABLE QUERY REWRITE -  
> AS SELECT E.ENAME, M.ENAME AS MANAGER -  
> FROM SCOTT.EMP E, SCOTT.EMP M WHERE E.MGR=M.EMPNO;
```

```
SQL> CREATE MATERIALIZED VIEW SCOTT.IN_DEPT -  
> REFRESH FAST ON COMMIT -  
> ENABLE QUERY REWRITE -  
> AS SELECT E.ROWID AS ERID, D.ROWID AS DRID, E.ENAME, D.DNAME -
```

```
> FROM SCOTT.EMP E, SCOTT.DEPT D WHERE E.DEPTNO=D.DEPTNO;
```

ロジカル・スタンバイ・データベースの場合

- トランザクションのコミットが発生すると、ロジカル・スタンバイ・データベースのON-COMMITマテリアライズド・ビューが自動的にリフレッシュされます。
- ON-DEMANDマテリアライズド・ビューは自動的にリフレッシュされません。リフレッシュするには、DBMS_MVIEW.REFRESH プロシージャを実行する必要があります。

たとえば、次のコマンドを発行すると、前の例で作成されたON-DEMANDマテリアライズド・ビューがリフレッシュされます。

```
SQL> ALTER SESSION DISABLE GUARD;
```

```
SQL> EXECUTE DBMS_MVIEW.REFRESH (LIST => 'SCOTT.MANAGED_BY', METHOD => 'C');
```

DBMS_SCHEDULERジョブを使用して定期的にオンデマンドでマテリアライズされたビューをリフレッシュする場合は、データベース・ガードをSTANDBYに設定しておく必要があります。(ALTER SESSION DISABLE GUARD文をPL/SQLブロック内で使用しても、効果がありません。)

11.6.3 ロジカル・スタンバイ・データベースでのトリガーと制約の処理方法

デフォルトでは、トリガーと制約はロジカル・スタンバイ・データベースで自動的に使用可能になり、処理されます。

SQL Applyでメンテナンスされる表にトリガーおよび制約がある場合：

- 制約：CHECK制約はプライマリ・データベースで評価されます。ロジカル・スタンバイ・データベースでの再評価は不要です。
- トリガー：プライマリ・データベースで実行されたトリガーの影響は、スタンバイ・データベースでログに記録され、適用されます。

SQL Applyでメンテナンスされない表にトリガーおよび制約がある場合：

- 制約は評価されます
- トリガーは発行されます

11.6.4 サポートされていない表のレプリケートでのトリガーの使用

表で作成されたDMLトリガーは、DBMS_DDL.SET_TRIGGER_FIRING_PROPERTYのfire_onceパラメータをTRUEにデフォルト設定します。

トリガーはユーザー・プロセスが変更された場合にのみ起動されます。SQL Applyプロセス内では自動的に無効化されるため、SQL Applyプロセスが表を変更しても起動されません。SQL Applyプロセスが保持している表に変更を加えた際に、トリガーを起動する方法が2つあります。

- トリガーのfire_onceパラメータをFALSEに設定します。ユーザー・プロセスとSQL Applyプロセスのどちらにも対応して起動できるようになります。
- apply_server_onlyパラメータをTRUEに設定します。こうすると、トリガーはSQL Applyプロセスに対応してのみ起動し、ユーザー・プロセスに対応しては起動しません。

fire_once	apply_server_only	説明
TRUE	FALSE	DML トリガーのデフォルトのプロパティ設定です。ユーザー・プロセスによって

fire_once	apply_server_only	説明
		実表が変更された場合のみトリガーが起動します。
FALSE	FALSE	トリガーは、実表を変更するユーザー・プロセスのコンテキストおよび SQL Apply プロセスのコンテキストで起動します。2 つのコンテキストは、DBMS_LOGSTDBY. IS_APPLY_SERVER ファンクションを使用して区別できます。
TRUE/FALSE	TRUE	SQL Apply プロセスによって実表が変更された場合のみトリガーが起動します。トリガーは、ユーザー・プロセスによって実表が変更される場合は起動しません。そのため、apply_server_only プロパティはトリガーの fire_once パラメータよりも優先されます。

単純なオブジェクト型の列のためにサポートされない表をレプリケートするには、SQL Applyプロセスに対応して起動するトリガーを作成します(トリガーのfire_onceパラメータをFALSEに設定するか、トリガーのapply_server_onlyパラメータをTRUEに設定します)。通常のDMLトリガーをプライマリ・データベースで使用して、オブジェクト型を、サポート可能な表にフラット化できます。ロジカル・スタンバイのSQL Applyプロセスに対応して起動するトリガーによって、オブジェクト型が再作成され、サポートされない表がトランザクション方式で更新されます。

関連項目:

- DBMS_DDL.SET_TRIGGER_FIRING_PROPERTYプロシージャとDBMS_LOGSTDBY.IS_APPLY_SERVER関数の詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)を参照してください。

次の例では、トリガーを使用して簡単なオブジェクト型の表をレプリケートする方法を示します。この例では挿入の処理方法を示しています。同じ原理で更新および削除も行えます。ネストされた表とVARRAYも、ネストされたデータの正規化を行うループを追加すれば、この方法でレプリケートすることができます。

```
-- simple object type
create or replace type Person as object
(
  FirstName    varchar2(50),
  LastName     varchar2(50),
  BirthDate    Date
)

-- unsupported object table
create table employees
(
  IdNumber     varchar2(10) ,
  Department   varchar2(50),
  Info         Person
)

-- supported table populated via trigger
create table employees_transfer
(
  t_IdNumber   varchar2(10),
  t_Department varchar2(50),
```



```

t_FirstName varchar2(50),
t_LastName  varchar2(50),
t_BirthDate Date
)
--
-- create this trigger to flatten object table on the primary
-- this trigger will not fire on the standby
--
create or replace trigger flatten_employees
after insert on employees for each row
declare
begin
insert into employees_transfer
(t_IdNumber, t_Department, t_FirstName, t_LastName, t_BirthDate)
values
(:new.IdNumber, :new.Department,
:new.Info.FirstName, :new.Info.LastName, :new.Info.BirthDate);
end

--
-- Option#1 (Better Option: Create a trigger and
-- set its apply-server-only property to TRUE)
-- create this trigger at the logical standby database
-- to populate object table on the standby
-- this trigger only fires when apply replicates rows
-- to the standby
--
create or replace trigger reconstruct_employees_aso
after insert on employees_transfer for each row
begin

insert into employees (IdNumber, Department, Info)
values (:new.t_IdNumber, :new.t_Department,
Person(:new.t_FirstName, :new.t_LastName, :new.t_BirthDate));

end

-- set this trigger to fire from the apply server
execute dbms_ddl.set_trigger_firing_property( -
trig_owner => 'scott', -
trig_name  => 'reconstruct_employees_aso',
property => dbms_ddl.apply_server_only,
setting => TRUE);

--
-- Option#2 (Create a trigger and set
-- its fire-once property to FALSE)
-- create this trigger at the logical standby database
-- to populate object table on the standby
-- this trigger will fire when apply replicates rows to -- the standby, but we will need to make sure
we are
-- are executing inside a SQL Apply process by invoking
-- dbms_logstdby.is_apply_server function
--
create or replace trigger reconstruct_employees_nfo
after insert on employees_transfer for each row
begin
if dbms_logstdby.is_apply_server() then
insert into employees (IdNumber, Department, Info)
values (:new.t_IdNumber, :new.t_Department,

```

```

Person(:new.t_FirstName, :new.t_LastName, :new.t_BirthDate));
    end if;
end

-- set this trigger to fire from the apply server
execute dbms_ddl.set_trigger_firing_property( -
trig_owner => 'scott', -
trig_name  => 'reconstruct_employees_nfo',
property => dbms_ddl.fire_once,
setting => FALSE);

```

11.6.5 プライマリでのPoint-in-Timeリカバリの実行によるリカバリ

ロジカル・スタンバイ・データベースがREDOデータの新規ブランチを受信すると、SQL Applyではそれが自動的に使用されます。

ロジカル・スタンバイ・データベースの場合、スタンバイ・データベースにより新規リセットログのSCNより後(REDOデータの新規ブランチの開始より後)のREDOデータが適用されていない場合、手動による介入は必要ありません。

次の表に、スタンバイ・データベースとプライマリ・データベースのブランチを再同期化する方法を示します。

スタンバイ・データベースの状態 . .	操作 . .	実行するステップ . .
新規リセットログの SCN の後(REDO データの新規ブランチの開始より後)の REDO データが適用されていない場合。	SQL Apply は、自動的に REDO データの新規ブランチを使用します。	手動による介入は必要ありません。SQL Apply は、自動的にスタンバイ・データベースを REDO データの新規ブランチと再同期化します。
新規リセットログの SCN の後(REDO データの新規ブランチの開始より後)の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっている場合。	スタンバイ・データベースは、REDO データの将来の新規ブランチでリカバリされます。	1. 「特定時点へのロジカル・スタンバイ・データベースのフラッシュバック」の手順に従って、ロジカル・スタンバイ・データベースをフラッシュバックします。 2. SQL Apply を再開し、新規リセットログ・ブランチへの REDO の適用を続行します。 SQL Apply は、自動的にスタンバイ・データベースを新規ブランチと再同期化します。
新規リセットログの SCN の後(REDO データの新規ブランチの開始より後)の REDO データが適用され、スタンバイ・データベースでフラッシュバック・データベースが使用可能になっていない場合。	指定のプライマリ・データベース・ブランチで、プライマリ・データベースとスタンバイの内容にずれが生じます。	「ロジカル・スタンバイ・データベースの作成」 の手順に従ってロジカル・スタンバイ・データベースを再作成します。
REDO データの前のブランチの終わりからアーカイブ REDO ログ・ファイルが欠	SQL Apply は欠落しているログ・ファイルが取得されるまで続	前のブランチから欠落しているアーカイブ REDO ログ・ファイルを検索して登録します。

落している場合。

行できません。

データベース・インカネーション、OPEN RESETLOGSによるリカバリ、フラッシュバック・データベースの詳細は、[『Oracle Database バックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

11.6.6 ロジカル・スタンバイ・データベースでのOracle Streams取得プロセスの実行

Oracle Streams取得プロセスをロジカル・スタンバイ・データベースで実行すると、ロジカル・スタンバイ・データベースに存在するすべての表(ローカル表またはプライマリ・データベースからレプリケートされているメンテナンス対象の表)の変更を取得できます。

メンテナンス対象の表の変更を取得するときは、Oracle Streams取得プロセスをプライマリ・データベースに対して実行する場合に比べると時間がかかります。時間が長くなるのは、ロジカル・スタンバイで実行しているとき、Oracle Streams取得プロセスは変更がプライマリからロジカル・スタンバイに転送され、SQL Applyによって適用されるのを待機する必要があるためです。ほとんどの場合、リアルタイム適用を実行していれば、数秒以上かかることはありません。

Oracle Streamsの取得プロセスは作成されたデータベースに関連付けられています。データベースのロールは関係ありません。たとえば、Bostonという名前のプライマリ・データベース、およびLondonという名前のロジカル・スタンバイがあると仮定します。ロール・トランジション時に、Oracle Streamsの取得プロセスを1つのデータベースから、別なデータベースに移動することはできません。たとえばLondonがロジカル・スタンバイのときに、Oracle Streamsの取得プロセスを作成した場合、スイッチオーバーやフェイルオーバーなどのロール・トランジションによって、Londonがプライマリになったとしても、London上に残ります。ロール・トランジション後もOracle Streamsの取得プロセスを継続するためには、次のようなロール・トランジション・トリガーを書く必要があります。

```
create or replace trigger streams_aq_job_role_change1
after DB_ROLE_CHANGE on database
declare
cursor capture_aq_jobs is
  select job_name, database_role
  from dba_scheduler_job_roles
  where job_name like 'AQ_JOB%';
u capture_aq_jobs%ROWTYPE;
my_db_role varchar2(16);
begin

  if (dbms_logstdby.db_is_logstdby() = 1) then my_db_role := 'LOGICAL STANDBY';
  else my_db_role := 'PRIMARY';
  end if;

  open capture_aq_jobs;
  loop
    fetch capture_aq_jobs into u;
    exit when capture_aq_jobs%NOTFOUND;

    if (u.database_role != my_db_role) then
      dbms_scheduler.set_attribute(u.job_name,
        'database_role',
        my_db_role);
    end if;
  end loop;
  close capture_aq_jobs;

exception
when others then
begin
```

```
raise;  
end;  
end;
```

11.7 ロジカル・スタンバイ・データベースのチューニング

次の各項では、ロジカル・スタンバイ・データベースをチューニングする様々な方法について説明します。

- [主キーRELY制約の作成](#)
- [コストベース・オブティマイザの統計の収集](#)
- [プロセス数の調整](#)
- [LCRキャッシュ用メモリの調整](#)
- [ロジカル・スタンバイ・データベースにおけるトランザクションの適用方法の調整](#)

11.7.1 主キーRELY制約の作成

プライマリ・データベースで、表に主キーまたは一意索引がなく、実際には一意である行がわかっている場合は、主キーのRELY制約を作成します。

ロジカル・スタンバイ・データベースでは、主キーを構成する列に索引を作成します。次の問合せは、ロジカル・スタンバイ・データベースで行を一意に識別するために適用できる索引情報がない表のリストを生成します。次の表に索引を作成することによって、パフォーマンスが大幅に向上する可能性があります。

```
SQL> SELECT OWNER, TABLE_NAME FROM DBA_TABLES -  
> WHERE OWNER NOT IN (SELECT OWNER FROM DBA_LOGSTDBY_SKIP -  
> WHERE STATEMENT_OPT = 'INTERNAL SCHEMA') -  
> MINUS -  
> SELECT DISTINCT TABLE_OWNER, TABLE_NAME FROM DBA_INDEXES -  
> WHERE INDEX_TYPE NOT LIKE ('FUNCTION-BASED%') -  
> MINUS -  
> SELECT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNsupported;
```

プライマリ・データベースの表に主キーのRELY制約を追加する手順は、次のとおりです。

1. プライマリ・データベースで主キーのRELY制約を追加します。

```
SQL> ALTER TABLE HR.TEST_EMPLOYEES ADD PRIMARY KEY (EMPNO) RELY DISABLE;
```

これにより、HR.TEST_EMPLOYEES表の各行を一意に識別するために使用可能なEMPNO列が、その表に対して実行される更新の一部としてサブリメンタル・ロギングで確実に記録されるようになります。

HR.TEST_EMPLOYEES表には、ロジカル・スタンバイ・データベースで指定された一意索引がまだないことに注意してください。このため、UPDATE文ではロジカル・スタンバイ・データベースに対して全表スキャンが実行される場合があります。ロジカル・スタンバイ・データベース上のEMPNO列に一意の索引を追加することで、これを修正できます。RELY制約の詳細は、[「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」](#)、および『[Oracle Database SQL 言語リファレンス](#)』を参照してください。

残りのステップをロジカル・スタンバイ・データベースで実行します。

2. SQL Applyを停止します。

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

3. ロジカル・スタンバイ・データベースでメンテナンスされる表を変更できるように、ガードを無効にします。

```
SQL> ALTER SESSION DISABLE GUARD;
```

4. EMPNO列に一意索引を追加します。

```
SQL> CREATE UNIQUE INDEX UI_TEST_EMP ON HR.TEST_EMPLOYEES (EMPNO);
```

5. ガードを有効にします。

```
SQL> ALTER SESSION ENABLE GUARD;
```

6. SQL Applyを開始します。

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

11.7.2 コストベース・オブティマイザの統計の収集

コストベース・オブティマイザ(CBO)で最適の問合せ実行パスを判別するために使用されるため、スタンバイ・データベースで統計を収集する必要があります。

以前の統計情報が正確でなくなるような変更をスキーマ・オブジェクトのデータまたは構造に対して実行した場合は、その後に新しい統計を収集してください。たとえば、多数の行を表に挿入または削除した後は、行数に関する新しい統計を収集します。

プライマリ・データベースでのDMLおよびDDL操作はワークロードの機能として実行されるため、統計はスタンバイ・データベースで収集してください。スタンバイ・データベースがプライマリ・データベースと論理的に等しい場合、SQL Applyではワークロードを異なる方法で実行する場合があります。そのため、ロジカル・スタンバイ・データベースでSTATSPACKパッケージおよびV\$SYSSTATビューを使用すると、リソースおよび表スキャンを最も多く使用している表を判断する際に役立ちます。

関連項目:

- [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](#)

11.7.3 プロセス数の調整

SQL Applyに割り当てられるプロセス数を制御するために変更できるパラメータには、MAX_SERVERS、APPLY_SERVERSおよびPREPARE_SERVERSの3つがあります。

常に、次の関係に該当する必要があります。

- $APPLY\_SERVERS + PREPARE\_SERVERS = MAX\_SERVERS - 3$

これは、SQL Applyが常にREADER、BUILDER、ANALYZERの各ロールに対して1つずつプロセスを割り当てるためです。

- デフォルトでは、MAX_SERVERSは9に、PREPARE_SERVERSは1に、APPLY_SERVERSは5に設定されます。
- DBMS_LOGSTDBY.APPLY_SETプロシージャを使用してMAX_SERVERSパラメータのみを変更し、SQL Applyでサーバー・プロセスを準備プロセスと適用プロセスの間に適当に分散できるようにすることをお勧めします。
- SQL Applyではプロセス割当てアルゴリズムを使用して、MAX_SERVERで指定されるように、SQL Applyに割り当てられる20サーバー・プロセスごとに1つのPREPARE_SERVERを割り当て、PREPARE_SERVERS の数を5に制限します。そのため、MAX_SERVERSを1から20の任意の値に設定すると、SQL ApplyはPREPARERとして機能するようにサーバー・プロセスを1つ割り当て、残りのプロセスをAPPLIERとして割り当てて前述の関係を満たします。同様に、MAX_SERVERSを21から40の値に設定すると、SQL ApplyはPREPARERSとして機能するようにサーバー・プロセスを2つ割り当て、残りをAPPLIERSとして割り当てて前述の関係を満たします。この内部プロセス割当てアルゴリズムは、前述の関係を満たし

ているならば、APPLY_SERVERSおよびPREPARE_SERVERSを直接設定することで上書きできます。

この項では、次の項目について説明します。

- [APPLIERプロセス数の調整](#)
- [PREPARERプロセス数の調整](#)

11.7.3.1 APPLIERプロセス数の調整

APPLIERプロセスの数を調整する前に、その調整でスループットを改善できるかどうか判断する必要があります。

これを決定するには、次のステップを実行します：

1. 次の問合せを発行して、APPLIERプロセスがビジーかどうかを確認します。

```
SQL> SELECT COUNT(*) AS IDLE_APPLIER -  
> FROM V$LOGSTDBY_PROCESS -  
> WHERE TYPE = 'APPLIER' and status_code = 16116;
```

```
IDLE_APPLIER  
-----  
0
```

2. アイドル状態のAPPLIERプロセスがないことを確認した後、次の問合せを発行し、APPLIERSの数を調整した場合に追加分のAPPLIERプロセスに十分な作業が使用可能かどうかを確認します。

```
SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME = 'txns applied' OR NAME = 'distinct txns  
in queue';
```

この2つの統計では、APPLIERプロセスによる適用準備が完了しているトランザクション数とすでに適用済のトランザクション数の累計が維持されます。

この数値(distinct txns in queue - txns applied)が使用可能なAPPLIERプロセス数の2倍を超えている場合は、APPLIERプロセス数を増やすことでスループットを改善できます。

ノート：



この数値は準備作業の概算です。ワークロードによっては、準備が完了したトランザクション間の依存性が、使用可能な追加のAPPLIERプロセスによる適用の妨げになる場合があります。たとえば、適用準備の完了しているトランザクションの大多数がDDLトランザクションの場合は、さらにAPPLIERプロセスを追加してもスループットは改善されません。

APPLIERプロセス数をデフォルト値の5から20に調整し、PREPARERプロセス数は1のままとします。これは、次の等式を満たす必要があるためです。

```
APPLY_SERVERS + PREPARE_SERVERS = MAX_SERVERS - 3
```

まずMAX_SERVERSを24に設定する必要があります。こうすると、APPLY_SERVERSの数を次のように20に設定できます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SERVERS', 24);  
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('APPLY_SERVERS', 20);
```


11.7.3.2 PREPARERプロセス数の調整

PREPARERプロセスの数の調整が必要になることは、まれです。その数を増やす前に、一定の条件が満たされていることを確認する必要があります。

trueになる必要がある条件は、次のとおりです。

- すべてのPREPARERプロセスがビジーである。
- 適用準備が完了しているトランザクションの数が、使用可能なAPPLIERプロセスの数よりも少ない。
- アイドル状態のAPPLIERプロセスがある。

前述の条件に該当しているかどうかを判断するステップは、次のとおりです。

1. すべてのPREPARERプロセスがビジーであることを確認します。

```
SQL> SELECT COUNT(*) AS IDLE_PREPARER -  
> FROM V$LOGSTDBY_PROCESS -  
> WHERE TYPE = 'PREPARER' and status_code = 16116;
```

```
IDLE_PREPARER  
-----  
0
```

2. 適用準備が完了しているトランザクションの数がAPPLIERプロセスの数よりも少ないことを確認します。

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME = 'txns applied' OR - > NAME =  
'distinct txns in queue';
```

NAME	VALUE
txns applied	27892
distinct txns in queue	12896

```
SQL> SELECT COUNT(*) AS APPLIER_COUNT -  
> FROM V$LOGSTDBY_PROCESS WHERE TYPE = 'APPLIER';
```

```
APPLIER_COUNT  
-----  
20
```

ノート：これが一時イベントでないことを確認するために、この問合せは数回発行してください。

3. アイドル状態のAPPLIERプロセスがあることを確認します。

```
SQL> SELECT COUNT(*) AS IDLE_APPLIER -  
> FROM V$LOGSTDBY_PROCESS -  
> WHERE TYPE = 'APPLIER' and status_code = 16116;
```

```
IDLE_APPLIER  
-----  
19
```

この例では、PREPARERプロセス数の増加に必要な3つの条件がすべて満たされています。APPLIERプロセス数の設定を20のままにし、PREPARERプロセス数を1から3に増やすとします。これは、常に次の等式を満たす必要があるためです。

```
APPLY_SERVERS + PREPARE_SERVERS = MAX_SERVERS - 3
```

まず、増加後のPREPARERプロセス数に対応するように、MAX_SERVERSの数を24から26に増やす必要があります。次に、PREPARERプロセス数を次のようにして増やすことができます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SERVERS', 26);
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('PREPARE_SERVERS', 3);
```

11.7.4 LCRキャッシュ用メモリの調整

ある程度のワークロードの場合、SQL Applyは多数のページアウト操作を使用することがあるため、システム全体のスループットが低下します。LCRキャッシュに割り当てられるメモリーを増やすと、改善する可能性があります。

LCRキャッシュに割当済のメモリーを増やした場合にスループットを改善できるかどうかを確認するステップは、次のとおりです。

1. 次の問合せを発行して、ページアウト・アクティビティのスナップショットを取得します。

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME LIKE '%page%' -
> OR NAME LIKE '%uptime%' OR NAME LIKE '%idle%';
```

NAME	VALUE
coordinator uptime (seconds)	894856
bytes paged out	20000
pageout time (seconds)	2
system idle time (seconds)	1000

2. この問合せを5分以内に再発行します。

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS WHERE NAME LIKE '%page%' -
> OR NAME LIKE '%uptime%' OR NAME LIKE '%idle%';
```

NAME	VALUE
coordinator uptime (seconds)	895156
bytes paged out	1020000
pageout time (seconds)	100
system idle time (seconds)	1000

3. 正規化されたページアウト・アクティビティを計算します。次に例を示します。

```
Change in coordinator uptime (C) = (895156 - 894856) = 300 secs
Amount of additional idle time (I) = (1000 - 1000) = 0
Change in time spent in pageout (P) = (100 - 2) = 98 secs
Pageout time in comparison to uptime = P / (C-I) = 98/300 ~ 32.67%
```

ページアウト・アクティビティによる消費量が稼働時間全体の5%以下であれば理想的です。間隔を長くして引き続きスナップショットを取得し、ページアウト・アクティビティが引き続き適用時間の大部分を占めていることがわかった場合は、メモリー・サイズを大きくするとスループットが改善される可能性があります。SQL Applyに割当済のメモリーは、LCRキャッシュに割り当てられるメモリーを設定する(この例ではSGAを1GBに設定)と増やすことができます。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SGA', 1024);
PL/SQL procedure successfully completed
```

11.7.5 ロジカル・スタンバイ・データベースにおけるトランザクションの適用方法の調整

デフォルトでは、トランザクションは、プライマリ・データベースでコミットされた順序に正確に従って、ロジカル・スタンバイ・データベースに適用されます。

厳密なデフォルト順序でのトランザクションのコミットにより、アプリケーションをロジカル・スタンバイ・データベースで透過的に実行できます。

ただし、多くのアプリケーションでは、すべてのトランザクションの間でそのような厳密な順序が必要なわけではありません。そのようなアプリケーションでは、オーバーラップしない一連の行を含むトランザクションを、プライマリ・データベースでコミットしたときと同じ順序でコミットする必要がありません。一般に、厳密でない順序により、ロジカル・スタンバイ・データベースでの適用率は高くなります。次のステップで、トランザクションをコミットするデフォルトの順序を変更できます。

1. SQL Applyを停止します。

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

2. 次の文を発行して、プライマリ・データベースでコミットされた順序と異なる順序でトランザクションを適用できるようにします。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET(' PRESERVE_COMMIT_ORDER', ' FALSE');  
PL/SQL procedure successfully completed
```

3. SQL Applyを開始します。

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

次のように適用モードを元に戻せます。

1. SQL Applyを停止します。

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

2. PRESERVE_COMMIT_ORDERパラメータのデフォルト値をリストアします。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_UNSET(' PRESERVE_COMMIT_ORDER');  
PL/SQL procedure successfully completed
```

3. SQL Applyを開始します。

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

一般的なオンライン・トランザクション処理(OLTP)のワークロードでは、デフォルト以外のモードにすると、デフォルトの適用モードに比べてスループットを50%以上改善できます。

11.8 ロジカル・スタンバイ・データベースの下でのバックアップおよびリカバリ

ロジカル・スタンバイ・データベースは、従来の方法を使用してバックアップでき、データベース・バックアップをリストアしてアーカイブ・ログに対してメディア・リカバリを実行すると、バックアップと同時にリカバリができます。

次の各項目は、ロジカル・スタンバイ・データベースの場合に該当します。

RMANのローカル・リカバリ・カタログの作成および使用時の考慮事項

RMANのリカバリ・カタログを作成する、またはカタログを変更するRMANのアクティビティを実行する場合、ロジカル・スタンバイ・データベースでは、GUARDをSTANDBYに設定して実行する必要があります。

ローカル・リカバリ・カタログがロジカル・スタンバイ制御ファイルにのみ保持されている場合は、GUARDをALLに設定したままにできます。

制御ファイルのバックアップに関する考慮事項

ロジカル・スタンバイ・データベースのインスタンス化の直後に、制御ファイルのバックアップを取得することをお勧めします。

Point-in-Timeリカバリに関する考慮事項

Point-in-Timeリカバリの後に初めてSQL Applyが開始されるとき、必要なアーカイブ・ログをローカル・システムで検出するか、プライマリ・データベースからフェッチできる必要があります。V\$LOGSTDBY_PROCESSビューを使用して、アーカイブ・ログをプライマリ・データベースでリストアする必要があるかどうかを判断します。

表領域のPoint-in-Timeリカバリに関する考慮事項

ロジカル・スタンバイ・データベースの表領域に対してPoint-in-Timeリカバリを実行する場合、次のいずれかを実行する必要があります。

- 表領域にSQL Applyプロセスでメンテナンスされている表またはパーティションが含まれないことを確認する。
- 表領域にSQL Applyプロセスでメンテナンスされている表またはパーティションが含まれる場合は、ロジカル・スタンバイ・データベースで、DBMS_LOGSTDBY.INSTANTIATE_TABLEプロシージャを使用して、リカバリされる表領域に含まれるメンテナンス対象の表をすべて再度インスタンス化するか、またはDBMS_LOGSTDBY.SKIPプロシージャを使用してリカバリされる表領域に含まれる表をすべて登録し、メンテナンス対象の表のリストからスキップされるようにする。

12 RMANを使用したファイルのバックアップおよびリストア

Oracle Data Guardおよびスタンバイ・データベースとともにOracle Recovery Manager(RMAN)を使用するバックアップ対策を作成できます。

RMANを使用すると、最小限の労力でプライマリ・データベースのバックアップを実行し、個々のデータファイルの消失またはデータベース全体をすばやくリカバリできます。RMANとOracle Data Guardをともに使用すると、Oracle Data Guard構成の管理がシンプルになります。

次のトピックを参照してください。

- [Oracle Data Guard構成でのRMANファイル管理について](#)
- [Oracle Data Guard環境でのRMAN構成について](#)
- [推奨されるRMAN構成およびOracle Database構成](#)
- [バックアップ手順](#)
- [Oracle Data Guard環境でのデータベースの登録および登録解除](#)
- [Oracle Data Guard環境におけるレポート生成](#)
- [Oracle Data Guard環境におけるバックアップ・メンテナンスの実行](#)
- [Oracle Data Guard環境におけるリカバリ例](#)
- [その他のバックアップ状況](#)
- [ネットワークを介したファイルのリストアおよびリカバリ](#)
- [Oracle Data Guard環境でのCDBに対するRMANサポート](#)

ノート:



ロジカル・スタンバイ・データベースはプライマリ・データベースのブロック単位のコピーではないため、ロジカル・スタンバイ・データベースを使用してプライマリ・データベースをバックアップすることはできません。

関連項目:

- RMANの概念と、Oracle Data Guard環境でのRMANの使用方法については、『[Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド](#)』を参照してください。
- RMANコマンドの詳細は、『[Oracle Databaseバックアップおよびリカバリ・リファレンス](#)』を参照してください。

12.1 Oracle Data Guard構成でのRMANファイル管理について

RMANでは、リカバリ・カタログを使用して、Oracle Data Guard環境のすべてのデータベース・ファイルについてファイル名を追跡管理します。

リカバリ・カタログは、1つ以上のOracleデータベースに関するメタデータを格納するためにRMANで使用されるデータベース・スキーマです。また、カタログには、オンラインREDOログ、スタンバイREDOログ、テンプレート、アーカイブREDOログ、バックアップ・

セットおよびイメージ・コピーが作成される場所も記録されます。

12.1.1 Oracle Data Guard環境でのバックアップの互換性

RMANコマンドは、リカバリ・カタログのメタデータを使用して、Oracle Data Guard環境の様々なフィジカル・データベースにわたって透過的に動作します。

たとえば、フィジカル・スタンバイ・データベースで表領域のバックアップを作成し、プライマリ・データベースでリストアおよびリカバリできます。同様に、プライマリ・データベースで表領域のバックアップを作成し、フィジカル・スタンバイ・データベースでリストアおよびリカバリできます。

ノート:



ロジカル・スタンバイ・データベースのバックアップは、プライマリ・データベースでは使用できません。

スタンバイ制御ファイルのバックアップと非スタンバイ制御ファイルのバックアップには互換性があります。たとえば、スタンバイ制御ファイルをプライマリ・データベースにリストアし、プライマリ制御ファイルをフィジカル・スタンバイ・データベースにリストアできます。この互換性は、制御ファイルのバックアップをOracle Data Guard環境の1つのデータベースにオフロードできることを意味します。RMANは、データベースでのリストアおよびリカバリ時に、データベース・ファイルのファイル名を自動的に更新します。

12.1.2 Oracle Data Guard環境でのバックアップの関連付け

リカバリ・カタログでは、すべてのデータベース・ファイル(バックアップ・ファイル)をDB_UNIQUE_NAMEに関連付けて、Oracle Data Guard環境のファイルを追跡管理します。

ファイルを作成するデータベースは、そのファイルに関連付けられます。たとえば、RMANでstandby1という一意の名前を持つデータベースをバックアップすると、standby1がこのバックアップに関連付けられます。バックアップは、CHANGE ... RESET DB_UNIQUE_NAMEを使用して別のデータベースに関連付けないかぎり、作成元のデータベースに関連付けられたままです。

12.1.3 Oracle Data Guard環境でのバックアップのアクセシビリティについて

Oracle Data Guard環境では、デフォルトで、関連付けられたデータベースのみがディスク・バックアップにアクセスできるとリカバリ・カタログによってみなされますが、1つのデータベースに作成されたテープ・バックアップにはすべてのデータベースがアクセスできます。

次のRMANコマンドを使用して、このデフォルトの動作を上書きできます。

```
SET BACKUP FILES FOR DEVICE TYPE device_specifier TO [NOTACCESSIBLE | ACCESSIBLE]
```

アクセシビリティ設定はセッション・レベルで行われます。つまり、RMANセッションが切断された後は、バックアップの関連付けがデフォルトの動作に戻ります。

バックアップ・ファイルがいずれのデータベースにも関連付けられていない場合、リカバリ・カタログ・ビュー内のバックアップ・ファイルに関する行のSITE_KEY列にnullが表示されます。デフォルト設定では、RMANはターゲット・データベースにSITE_KEYがnullのファイルに関連付けます。

BACKUP、RESTORE、CROSSCHECKなどのRMANコマンドは、アクセス可能なバックアップに対して作用します。たとえば、RECOVER COPY操作の場合、RMANはデータベースに関連付けられているイメージ・コピーのみをリカバリ対象とみなします。また、ディスクおよびテープ上の増分バックアップをイメージ・コピーのリカバリ対象とみなします。データベース・リカバリでは、RMANは、データベースに関連付けられたディスク・バックアップのみとテープ上のすべてのファイルをリストアの対象とみなします。

バックアップのアクセス可能性における差異を示すため、データベースprodおよびstandby1が異なるホスト上に存在すると仮定します。RMANは、PROD上のデータファイル1を本番ホスト上の/prmhost/disk1/df1.dbfおよびテープにバックアップします。RMANは、standby1上のデータファイル1をスタンバイ・ホスト上の/sbyhost/disk2/df1.dbfおよびテープにバックアップします。RMANがデータベースprodに接続されると、RMANコマンドを使用してスタンバイ・ホスト上にある/sbyhost/disk2/df1.dbfバックアップで操作を実行することはできません。しかし、RMANはstandby1で作成したテープ・バックアップをリストア対象とみなします。

ノート:



スタンバイ・ホストからプライマリ・ホスト(またはその逆)にバックアップをFTP転送し、対象のホスト上のデータベースにTARGETとして接続した後、バックアップに対して [CATALOG](#) を実行することができます。ファイルは、ターゲット・データベースによってカタログに追加された後に、ターゲット・データベースに関連付けられます。

12.2 Oracle Data Guard環境でのRMAN構成について

Oracle Data Guard構成では、制御ファイル、データファイルおよびアーカイブ・ログのバックアップ処理をスタンバイ・システムにオフロードできるため、バックアップが本番システムに与える影響は最小限に抑えられます。

これらのバックアップを使用すると、プライマリ・データベースまたはスタンバイ・データベースをリカバリできます。

RMANでは、DB_UNIQUE_NAME初期化パラメータを使用してデータベース・サイトを区別します。そのため、DB_UNIQUE_NAMEの一意性がOracle Data Guard構成で維持されることは不可欠なことです。

RMANのREGISTER DATABASEコマンドを使用して明示的に登録する必要があるのは、プライマリ・データベースのみです。この登録は、RMANをリカバリ・カタログに接続し、ターゲットとしてプライマリ・データベースに接続した後にを行います。

RMAN構成を設定するには、RMANのCONFIGUREコマンドを使用します。CONFIGUREコマンドがFOR DB_UNIQUE_NAMEオプションを指定して使用されると、指定したDB_UNIQUE_NAMEのデータベースに対してRMANサイト固有の構成が設定されます。

たとえば、リカバリ・カタログへの接続後、次のコマンドをRMANプロンプトで使用して、DBIDが1625818158のBOSTONデータベースに対してデフォルト・デバイス・タイプをSBTに設定できます。RMANのSET DBIDコマンドは、ターゲットとしてデータベースに接続していない場合にのみ必要です。

```
SET DBID 1625818158;  
CONFIGURE DEFAULT DEVICE TYPE TO SBT FOR DB_UNIQUE_NAME BOSTON;
```

12.3 推奨されるRMAN構成およびOracle Database構成

これらの構成によってバックアップおよびリカバリ操作を簡略化できます。

- [プライマリおよびスタンバイ・データベースでのOracle Database構成](#)
- [プライマリ・データベースでのRMAN構成](#)
- [バックアップが実行されるスタンバイ・データベースでのRMAN構成](#)
- [バックアップが実行されないスタンバイでのRMAN構成](#)

構成の前提

これらの構成は、次の前提に立っています。

- スタンバイ・データベースはフィジカル・スタンバイ・データベースで、バックアップはスタンバイ・データベースでのみ作成されます。プライマリ・データベースとスタンバイ・データベースの両方でバックアップを作成する場合の手順の変更は、[「スタンバイ・データベースが地理的に離れすぎているためにバックアップを共有できない場合」](#)を参照してください。
- 1つのデータベース・サーバーで作成されたバックアップを別なデータベース・サーバーにリストアできるようにするには、RMANリカバリ・カタログが必要です。プライマリ・データベースはスタンバイ・データベースでバックアップが作成されたことを知らないため、RMANリポジトリとして制御ファイルのみを使用するのでは不十分です。

RMANのリカバリ・カタログは、バックアップ履歴および他のリカバリ関連のメタデータを一元化された場所に編成します。リカバリ・カタログは、データベース内に構成され、バックアップ・メタデータが格納されます。リカバリ・カタログには、制御ファイルに関する領域の制限がないため、バックアップに関する履歴データをより多く格納できます。

カタログ・サーバーは、プライマリ・サイトおよびスタンバイ・サイトと物理的に切り離されていますが、Oracle Data Guard構成内に置くことをお勧めします。これは、いずれのサイトで障害が発生しても最新のバックアップをリカバリする機能に影響しないためです。

関連項目:

リカバリ・カタログの管理の詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

- 構成内のすべてのデータベースには、Oracle Database 11gリリース1 (11.1)以上を使用します。
- Oracle Secure Backupソフトウェアまたはサード・パーティのメディア管理ソフトウェアは、バックアップをテープに取るようにRMANで構成します。

12.3.1 プライマリおよびスタンバイ・データベースでのOracle Database構成

次のOracle Database構成は、Oracle Data Guard環境におけるすべてのプライマリ・データベースおよびスタンバイ・データベースで推奨されます。

- データベースごとに高速リカバリ領域を構成する(リカバリ領域はデータベースに対してローカル)。

高速リカバリ領域は、ファイル・システムまたはOracle Automatic Storage Management(Oracle ASM)ディスク・グループ上にある、リカバリに必要なすべてのファイルが存在する唯一の格納場所です。これらのファイルには、制御ファイル、アーカイブ・ログ、オンラインREDOログ、フラッシュバック・ログおよびRMANバックアップがあります。新しいバックアップおよびアーカイブ・ログが高速リカバリ領域に作成されると、古いファイル(保存期間を超過しているまたは三次記憶域にバックアップされている)は自動的に削除され、新しいファイル用の領域が作成されます。さらに、高速リカバリ領域の領域使用量が事前に定義された制限に近づいている場合に、DBAにアラートを送信するように通知を設定できます。DBAは、リカバリ領域の領域制限を上げる、ディスク・ハードウェアを追加する、あるいは保存期間を短縮するなどの処置を取ることができます。

次の初期化パラメータを設定して、高速リカバリ領域を構成します。

```
DB_RECOVERY_FILE_DEST = <file system location or Oracle ASM Disk Group>
DB_RECOVERY_FILE_DEST_SIZE = <disk space quota>
```

関連項目:

高速リカバリ領域の構成の詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

さい。

- サーバー・パラメータ・ファイル(SPFIL)を使用して、インスタンス・パラメータがバックアップに保存されるようにバックアップを作成できるようにする。
- プライマリ・データベースおよびスタンバイ・データベースでフラッシュバック・データベースを有効にする。

フラッシュバック・データベースを有効に設定すると、Oracle Databaseによってフラッシュバック・ログが高速リカバリ領域に保持されます。これらのログを使用すると、完全にリストアしなくても、前の時点までデータベースをロールバックできます。

関連項目:

フラッシュバック・データベースの有効化に関する詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。

12.3.2 プライマリ・データベースでのRMAN構成

RMANの継続的な使用を簡略化するために、多くの永続的な構成設定をOracle Data Guard環境のデータベースごとに設定できます。

これらの設定により、RMANの動作を多面的に制御します。たとえば、バックアップの保存ポリシー、デフォルトのバックアップ先(テープまたはディスク)、デフォルトのバックアップ・デバイス・タイプなどを構成できます。CONFIGUREコマンドを使用すると、RMAN構成を設定および変更できます。次のRMAN構成は、プライマリ・データベースにおいて推奨されます。

1. RMANをプライマリ・データベースおよびリカバリ・カタログに接続します。
2. データベースの保存ポリシーをn日間として構成します。

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF <n> DAYS;
```

この構成により、任意の時点までのデータベース・リカバリの実行に必要なバックアップを指定した日数の間保存できます。

DELETE OBSOLETEコマンドを使用して、特定日数の間はリカバリを実行する必要がないバックアップ(設定されている保存方針に基づき)を削除します。

3. アrchive・ログを削除できる時期をCONFIGURE ARCHIVELOG DELETION POLICYコマンドを使用して指定します。たとえば、すべての宛先への送信を確認した後にログを削除する場合は、次の構成を使用します。

```
CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO ALL STANDBY;
```

すべてのスタンバイ宛先への適用を確認した後にログを削除するには、次の構成を使用します。

```
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
```

4. RESYNC CATALOG FROM DB_UNIQUE_NAMEコマンドの使用時に、RMANがリモートで接続して、再同期化を行えるように、プライマリ・データベースおよびすべてのスタンバイ・データベースの接続文字列を構成します。ターゲット・インスタンスに接続する際、ネット・サービス名を提供する必要があります。再同期化元の他のデータベースがローカル・ホストにある場合でも、この要件は適用されます。ターゲットおよびリモート・インスタンスは同じSYSDBA(またはSYSBACKUP)パスワードを使用する必要があります。したがって、両方のインスタンスにパスワード・ファイルがなければなりません。単一のパスワードでパスワード・ファイルを作成し、そのパスワード・ファイルですべてのデータベース・インスタンスを起動することが

できます。たとえば、ボストンにあるスタンバイへの接続用TNS別名がboston_conn_strの場合、次のコマンドを使用してBOSTONデータベース・サイトの接続識別子を構成することができます。

```
CONFIGURE DB_UNIQUE_NAME BOSTON CONNECT IDENTIFIER 'boston_conn_str';
```

'boston_conn_str'にはユーザー名とパスワードが含まれていない点に注意してください。任意のデータベース・サイトからBOSTONデータベース・サイトに接続するために使用できる、Oracle Netサービス名だけが含まれています。

すべてのスタンバイ・データベースについて接続識別子が構成されたら、LIST DB_UNIQUE_NAME OF DATABASEコマンドを使用してスタンバイのリストを確認できます。

関連項目:

- RMAN構成の詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。
- RMAN CONFIGUREコマンドの詳細は、『[Oracle Databaseバックアップおよびリカバリ・リファレンス](#)』を参照してください。

12.3.3 バックアップが実行されるスタンバイ・データベースでのRMAN構成

次のRMAN構成は、バックアップが実行されるスタンバイ・データベースで推奨されます。

1. RMANをターゲットとして(バックアップが実行される)スタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. 制御ファイルおよびサーバー・パラメータ・ファイルの自動バックアップを有効にします。

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

3. 同じチェックポイントで有効なバックアップがすでに存在するデータファイルのバックアップをスキップします。

```
CONFIGURE BACKUP OPTIMIZATION ON;
```

4. メディア管理ソフトウェアによる要求に応じて、バックアップを作成するためのテープ・チャンネルを構成します。

```
CONFIGURE CHANNEL DEVICE TYPE SBT PARMS '<channel parameters>';
```

5. アーカイブ・ログがスタンバイ・データベースでバックアップされるため、ログ削除ポリシーにBACKED UPオプションを構成することをお勧めします。

```
CONFIGURE ARCHIVELOG DELETION POLICY BACKED UP n TIMES TO DEVICE TYPE SBT;
```

関連項目:

アーカイブREDOログの削除方針の有効化の詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。

12.3.4 バックアップが実行されないスタンバイでのRMAN構成

これらのRMAN構成は、バックアップが実行されないスタンバイ・データベースで推奨されます。

1. RMANをターゲットとしてスタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. アーカイブ・ログの自動削除を、スタンバイ・データベースで適用されたらすぐに有効にします(カスケードまたは遠隔同期インスタンス機能が使用されている場合に、これはすべてのターミナル・データベースにも適用可能です)。

12.4 バックアップ手順

RMANのスクリプトと手順を使用して、Oracle Data Guard構成内のOracle Databaseをバックアップできます。

次のトピックを参照してください。

- [テープ・バックアップ用キャッシュとしてのディスクの使用](#)
- [テープへの直接バックアップの実行](#)

ノート:



Oracle の Maximum Availability Architecture(MAA)ベスト・プラクティスでは、プライマリ・データベースとスタンバイ・データベースの両方が停止した場合や、スイッチオーバーおよびフェイルオーバーについてサイトのプラクティスを新たに導入しなくてもよいようにする場合に、プライマリ・データベースとスタンバイ・データベースの両方でバックアップを作成して MTTR を短縮することをお薦めしています。

サーバー・パラメータ・ファイルのバックアップ

SPFILEのバックアップを除いたすべてのバックアップ操作は、単一のスタンバイ・データベースにオフロードできます。SPFILEのバックアップは、バックアップ元のデータベースにのみリストアできます。

バックアップされていないデータベースの場合、少なくともSPFILEを既知のローカル・ディスクの場所にバックアップすることをお薦めします。SPFILEバックアップをさらにテープにバックアップする必要がある場合、テープへのバックアップが構成済のデータベース・サイトへコピーすることができます。さらに次のRMANコマンドを使用して、SPFILEバックアップをそのデータベースでカタログ化することができます。

```
CATALOG START WITH '<SPFILE backup directory>';
```

次に、SPFILEバックアップをテープにバックアップします。

```
BACKUP BACKUPSET ALL;
```

SPFILEが特定のデータベースにリストアされる必要があるときは、適切なSPFILEバックアップがディスクまたはテープからリストアされます。

12.4.1 テープ・バックアップ用キャッシュとしてのディスクの使用

スタンバイ・データベースの高速リカバリ領域は、テープ・バックアップ用のディスク・キャッシュとして機能できます。

ディスクはバックアップ用の一次記憶域として使用され、テープは長期間のアーカイブ記憶域となります。テープの増分バックアップは毎日、テープの全体バックアップは毎週作成されます。これらのバックアップの実行に使用されるコマンドは、次の各項を参照してください。

12.4.1.1 ディスクをキャッシュとして使用する日次テープ・バックアップ用のコマンド

バックアップ計画を決定する際に、日次増分バックアップを利用することをお薦めします。

データファイル・イメージ・コピーは、最新の増分バックアップを使用してロールフォワードできるため、常に最新のデータファイル・イメージ・コピーを提供できます。RMANでは、そのシステム変更番号(SCN)で取得された全体イメージ・コピーを使用するように、

結果のイメージ・コピーをメディア・リカバリに使用するため、データベースの全体イメージ・コピーを毎日実行するオーバーヘッドはありません。もう1つのメリットは、イメージ・コピーが最新のブロック変更で更新され、現在の状態にデータベースを戻すのに必要なREDOログが少なくなるため、リカバリ時間が短縮されることです。

日次増分バックアップを実施するには、1日目にデータベースの全体バックアップを作成し、2日目に増分バックアップを作成します。アーカイブREDOログを使用すると、いずれの日のどの時点についてもデータベースをリカバリできます。3日目以降は、前日の増分バックアップをデータファイル・コピーとマージし、当日の増分バックアップを作成して、最終日までのどの時点についても高速リカバリを可能にします。REDOログを使用すると、当日のどの時点についてもデータベースをリカバリできます。

日次バックアップを実行するスクリプトは次のとおりです(最終行DELETE ARCHIVELOG ALLは、高速リカバリ領域をログの格納に使用しない場合にのみ必要です)。

```
RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;  
RECOVER COPY OF DATABASE WITH TAG 'OSS';  
BACKUP DEVICE TYPE DISK INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG 'OSS' DATABASE;  
BACKUP DEVICE TYPE DISK DATAFILECOPY ALL NOT BACKED UP AT ANY SCN;  
BACKUP DEVICE TYPE SBT ARCHIVELOG ALL;  
BACKUP BACKUPSET ALL;  
DELETE ARCHIVELOG ALL;
```

スタンバイ制御ファイルは、制御ファイルの自動バックアップが有効に設定されているため、バックアップ操作の最後に自動的にバックアップされます。

次に、スクリプトの各コマンドの動作について説明します。

- RESYNC CATALOG FROM DB_UNIQUE_NAME ALL

Oracle Data Guard設定で、リカバリ・カタログが把握しているその他すべてのデータベース・サイト(プライマリおよびその他のスタンバイ・データベース)からの情報を再同期化します。RESYNC CATALOG FROM DB_UNIQUE_NAMEが機能するためには、Oracle Netサービス名を使用してRMANをターゲットに接続し、すべてのデータベースで同じパスワード・ファイルを使用する必要があります。

- RECOVER COPY OF DATABASE WITH TAG 'OSS'

前日に取得したレベル1の増分バックアップを適用して、データベースのレベル0のコピーをロールフォワードします。この例のスクリプトでは、前日のレベル1の増分にはOSSタグが付けられています。その増分は、BACKUP DEVICE TYPE DISK ... DATABASEコマンドによって生成されます。このコマンドが実行された1日目には、レベル1の増分がまだないため、ロールフォワードはありません。レベル0の増分が、BACKUP DEVICE TYPE DISK ... DATABASEコマンドによって生成されます。2日目にも、レベル0の増分しかないため、ロールフォワードはありません。OSSタグが付けられたレベル1の増分が、BACKUP DEVICE TYPE DISK ... DATABASEコマンドによって生成されます。3日目以降には、前日に作成されたOSSタグ付きのレベル1の増分を使用して、ロールフォワードが実行されます。

- BACKUP DEVICE TYPE DISK INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG 'OSS' DATABASE

新しいレベル1の増分バックアップを作成します。このコマンドの最初の実行日に、ベース・データファイルのコピーが作成されます。2日目以降は、レベル1の増分となります。

- BACKUP DEVICE TYPE DISK DATAFILECOPY ALL NOT BACKED UP AT ANY SCN

まだバックアップされていないデータ・ファイルのみをバックアップします。

- BACKUP DEVICE TYPE SBT ARCHIVELOG ALL

所定の削除ポリシーに従って、アーカイブ・ログをテープにバックアップします。

- BACKUP BACKUPSET ALL

増分バックアップの作成結果として作成されるバックアップ・セットをバックアップします。

- DELETE ARCHIVELOG ALL

CONFIGURE ARCHIVELOG DELETION POLICYコマンドで設定されたログ削除ポリシーに従って、アーカイブ・ログを削除します。アーカイブ・ログが高速リカバリ領域にある場合は、空きディスク領域がさらに必要になると自動的に削除されます。そのため、毎日ログを明示的に削除する場合にのみ、このコマンドを使用する必要があります。

12.4.1.2 ディスクをキャッシュとして使用する週次テープ・バックアップ用のコマンド

すべてのリカバリ関連ファイルをテープにバックアップするには、RMAN BACKUP RECOVERY FILESコマンドを週1回使用します。

これにより、ディスク上にある最新の増分バックアップ、イメージ・コピー・バックアップおよびアーカイブ・ログ・バックアップがすべてテープにバックアップされます。

12.4.2 テープへの直接バックアップの実行

OracleのMedia Management Layer(MML)APIにより、サード・パーティ・ベンダーは、メディア・マネージャ、RMANと連動して機能するソフトウェア、テープ・ドライブなどの順次メディア・デバイスへのバックアップを可能にするベンダーのハードウェアを構築できます。

メディア・マネージャは、テープなどの順次メディアのロード、アンロードおよびラベル付けを処理します。RMANを順次メディア・デバイスとともに使用するには、Oracle Secure Backupまたはサード・パーティのメディア管理ソフトウェアをインストールする必要があります。

デフォルトでは、次のステップを実行してテープへの直接バックアップを実行します。

1. RMANを(ターゲット・データベースとして)スタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. 次のように、CONFIGUREコマンドを実行します。

```
CONFIGURE DEFAULT DEVICE TYPE TO SBT;
```

この例では、スタンバイ・データベースで全体バックアップが毎週、増分バックアップが毎日作成されます。

関連項目:

メディア・マネージャ用のRMANの構成方法の詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』](#)を参照してください。

12.4.2.1 テープに直接実行する日次バックアップ用のコマンド

テープへの直接の日次バックアップ実行に使用されるRMANコマンドは、Oracle Data Guard環境にある他のデータベースすべてから情報を再同期化します。

次のステップを実行して、テープに日次バックアップを直接実行します。

1. RMANをスタンバイ・データベース(ターゲット・データベースとして)、およびリカバリ・マネージャに接続します。
2. 次のRMANコマンドを実行します。

```
RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;  
BACKUP AS BACKUPSET INCREMENTAL LEVEL 1 DATABASE PLUS ARCHIVELOG;  
DELETE ARCHIVELOG ALL;
```

またこれらのコマンドで、すべてのアーカイブ・ログを含むデータベースのレベル1の増分バックアップを作成します。このスクリプトが実行された1日目に、レベル0のバックアップが検出されなければ、レベル0のバックアップが作成されます。

DELETE ARCHIVELOG ALLコマンドは、すべてのアーカイブ・ログ・ファイルが高速リカバリ領域にない場合にのみ必要です。

12.4.2.2 テープに直接実行する週次バックアップ用のコマンド

週に1回、テープへの週次バックアップを直接実行します。

次のステップを実行します。

1. RMANを(ターゲット・データベースとして)スタンバイ・データベースに接続し、リカバリ・カタログに接続します。
2. 次のRMANコマンドを実行します。

```
RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;  
BACKUP AS BACKUPSET INCREMENTAL LEVEL 0 DATABASE PLUS ARCHIVELOG;  
DELETE ARCHIVELOG ALL;
```

これらのコマンドは、Oracle Data Guard環境内のその他すべてのデータベースの情報を再同期化し、すべてのアーカイブ・ログを含むレベル0のデータベース・バックアップを作成します。

DELETE ARCHIVELOG ALLコマンドは、すべてのアーカイブ・ログ・ファイルが高速リカバリ領域にない場合にのみ必要です。

12.5 Oracle Data Guard環境でのデータベースの登録および登録解除

REGISTER DATABASEコマンドを使用して明示的に登録する必要があるのは、プライマリ・データベースのみです。この登録は、RMANをリカバリ・カタログに接続し、TARGETとしてプライマリ・データベースに接続した後に行います。

新しいスタンバイは、スタンバイ・データベースに接続するか、またはCONFIGURE DB_UNIQUE_NAMEコマンドを使用して接続識別子を構成すると、リカバリ・カタログに自動的に登録されます。

特定のスタンバイ・データベースに関する情報を登録解除するには、UNREGISTER DB_UNIQUE_NAMEコマンドを使用できます。スタンバイ・データベースをOracle Data Guard環境から完全に削除すると、同じOracle Data Guard環境内の別のデータベースに接続した後に、リカバリ・カタログのデータベース情報も削除できます。登録解除されたデータベースと関連付けられたバックアップは、そのまま他のデータベースで使用できます。これらのバックアップは、CHANGE BACKUP RESET DB_UNIQUE_NAMEコマンドを使用して既存の他のデータベースに関連付けることができます。

INCLUDING BACKUPSオプションを指定してUNREGISTER DB_UNIQUE_NAMEコマンドを使用すると、登録解除中のデータベースに関連付けられたすべてのバックアップ・ファイルのメタデータもリカバリ・カタログから登録解除されます。

12.6 Oracle Data Guard環境におけるレポート生成

FOR DB_UNIQUE_NAME句を指定してRMANのLIST、REPORTおよびSHOWの各コマンドを使用すると、特定のデータベースに関する情報を表示できます。

たとえば、リカバリ・カタログに接続した後、次のコマンドを使用してDBIDが1625818158のデータベースに関する情報を表示し、Oracle Data Guard環境内のデータベースを出力できます。SET DBIDコマンドは、TARGETとしてデータベースに接続していない場合にのみ必要です。最後の3つのコマンドは、DB_UNIQUE_NAMEがBOSTONのデータベースについて、アーカイブ・ログ、データベース・ファイル名およびRMANの構成情報を出力します。

```
SET DBID 1625818158;  
LIST DB_UNIQUE_NAME OF DATABASE;  
LIST ARCHIVELOG ALL FOR DB_UNIQUE_NAME BOSTON;  
REPORT SCHEMA FOR DB_UNIQUE_NAME BOSTON;  
SHOW ALL FOR DB_UNIQUE_NAME BOSTON;
```

12.7 Oracle Data Guard環境におけるバックアップ・メンテナンスの実行

Oracle Data Guard環境のファイル(データファイル、アーカイブ・ログ、バックアップ・ピース、イメージ・コピー、プロキシ・コピー)は、DB_UNIQUE_NAMEパラメータを使用してデータベースに関連付けられます。

そのため、Oracle Data Guard環境のデータベースごとに、DB_UNIQUE_NAMEに指定された値は一意であることが重要です。この情報は、ファイル共有属性とともに、RMANの様々な操作の際にアクセスできるファイルを判別するのに使用されます。

ファイル共有属性は、ディスク上のファイルは関連付けられたデータベースでのみアクセス可能で、テープ上のすべてのファイルはすべてのデータベースでアクセス可能に設定されています。BACKUPおよびRESTOREなどのRMANコマンド、およびその他のメンテナンス・コマンドは、この想定で機能しています。たとえば、データベースでのイメージ・コピーのロールフォワード操作中は、そのデータベースに関連付けられたイメージ・コピーだけがロールフォワードされます。そのデータベースに関連付けられたディスクでの増分バックアップおよびテープでの増分バックアップは、イメージ・コピーのロール転送に使用されます。また、リカバリ操作中は、データベースに関連付けられたディスク・バックアップとテープ上のファイルのみがバックアップ元と見なされます。

関連項目:

RMANコマンドの詳細は、[『Oracle Databaseバックアップおよびリカバリ・リファレンス』](#)を参照してください。

12.7.1 リカバリ・カタログのメタデータの変更

様々なオペランドを指定してRMAN CHANGEコマンドを使用すると、リカバリ・カタログ内のメタデータを変更できます。

次に例を示します。

- スタンバイ・データベース間でのファイルの関連付けの変更

RESET DB_UNIQUE_NAMEオプションを指定してCHANGEコマンドを使用すると、Oracle Data Guard環境内のあるデータベースから別のデータベースにファイルの関連付けを変更できます。CHANGEコマンドは、ディスク・バックアップまたはアーカイブ・ログのあるデータベースから別のデータベースに転送し、転送先のデータベースでそれらのファイルを使用するときに役立ちます。また、CHANGEコマンドは、FOR DB_UNIQUE_NAMEオプションとRESET DB_UNIQUE_NAME TOオプションを使用すると、いずれのデータベースに直接接続しなくてもデータベース間でファイルの関連付けを変更できます。

- データベースのDB_UNIQUE_NAME初期化パラメータの変更

データベースのDB_UNIQUE_NAME初期化パラメータの値を変更すると、Oracle Data Guard環境でも同じ変更を行う必要があります。そのデータベース・インスタンスへの接続後、RMANのリカバリ・カタログはDB_UNIQUE_NAMEの新旧両方の値を認識します。リカバリ・カタログ・スキーマ内で新旧の値に関する情報をマージするには、RMANのCHANGE DB_UNIQUE_NAMEコマンドを使用する必要があります。データベースのDB_UNIQUE_NAME初期化パラメータの値を変更すると、新しいDB_UNIQUE_NAMEを認識させるために、RMANでも同じ変更を行う必要があります。たとえば、次のステップを実行し、DB_UNIQUE_NAMEをBOSTON_AからBOSTON_Bに変更することでデータベースを変更します。

1. 初期化パラメータ・ファイルまたはSQLで、DB_UNIQUE_NAME初期化パラメータをBOSTON_AからBOSTON_Bに変更します。
2. RMANで、Oracle Data Guard environment環境のデータベースにターゲット・データベースとして接続し、リカバリ・カタログに接続します。続いて、CHANGEコマンドを実行します。

```
CHANGE DB_UNIQUE_NAME FROM BOSTON_A TO BOSTON_B;
```

- バックアップの使用不可化またはメタデータの削除

AVAILABLE、UNAVAILABLE、KEEP、UNCATALOGなどのオプションを指定してCHANGEコマンドを使用すると、バックアップをリストアおよびリカバリの目的で使用可能または使用不可能にしたり、そのメタデータを保存または削除することができます。

関連項目:

RMANのCHANGEコマンドの詳細は、[『Oracle Databaseバックアップおよびリカバリ・リファレンス』](#)を参照してください。

12.7.2 アーカイブ・ログまたはバックアップの削除

RMAN DELETEコマンドを使用して、バックアップ・セット、イメージ・コピー、アーカイブ・ログ、プロキシ・コピーを削除します。

特定のデータベースに関連付けられているファイルのみを削除するには、FOR DB_UNIQUE_NAMEオプションをDELETEコマンドとともに使用する必要があります。

ファイルのメタデータは、現行のターゲット・データベースに関連付けられ、正常に削除されたすべてのファイル(またはどの既知のデータベースにも関連付けられていないファイル)について削除されます。ファイルが正常に削除できなかった場合は、FORCEオプションを使用してファイルのメタデータを削除します。

別のデータベースに関連付けられているファイルが正常に削除された場合、リカバリ・カタログのそのメタデータも削除されます。他のデータベースに関連付けられているファイルおよび正常に削除できなかったファイルは、DELETEコマンドの完了時に、ファイルが関連付けられているデータベースで同じ操作を実行するように求める指示とともに出力されます(ファイルはデータベースごとにグループ化されます)。FORCEオプションを使用して、この動作を上書きすることはできません。削除不可のファイルのメタデータを削除しても問題ないことが明確である場合は、CHANGE RESET DB_UNIQUE_NAMEコマンドを使用してデータベースとのファイルの関連付けに対するメタデータを変更し、FORCEオプションを指定してDELETEコマンドを使用し、ファイルのメタデータを削除します。

関連項目:

RMANのDELETEコマンドの詳細は、[『Oracle Databaseバックアップおよびリカバリ・リファレンス』](#)を参照してください。

12.7.3 リカバリ・カタログのメタデータの検証

CROSSCHECKコマンドを使用してリカバリ・カタログ・スキーマのファイル・ステータスを検証および更新します。

現行のターゲット・データベースに関連付けられたすべてのファイル(またはどのデータベースにも関連付けられていないあらゆるファイル)のメタデータは、CROSSCHECK操作の結果に応じてAVAILABLEまたはEXPIREDのマークが付けられます。

別のデータベースに関連付けられているファイルが正常に検査された場合、リカバリ・カタログのそのメタデータもAVAILABLEに変更されます。他のデータベースに関連付けられているファイルおよび正常に検査できなかったファイルは、CROSSCHECKコマンドの完了時に、ファイルが関連付けられているデータベースで同じ操作を実行するように求める指示とともに出力されます(ファイルはサイトごとにグループ化されます)。確認し、それでも使用不可のファイルのステータス・メタデータを変更する場合は、CHANGE RESET DB_UNIQUE_NAMEコマンドを使用してデータベースとのファイルの関連付けに対するメタデータを変更し、CROSSCHECKコマンドを実行してステータス・メタデータをEXPIREDに更新します。

関連項目:

RMANのCROSSCHECKコマンドの詳細は、[『Oracle Databaseバックアップおよびリカバリ・リファレンス』](#)を参照してください。

12.8 Oracle Data Guard環境におけるリカバリ例

Oracle Data Guard環境で発生する可能性のあるリカバリ・シナリオの一部を次に示します。

- [プライマリまたはスタンバイ・データベースでのファイルの損失からのリカバリ](#)
- [オンラインREDOログ・ファイルの損失からのリカバリ](#)
- [プライマリ・データベースの不完全リカバリ](#)
- [プライマリでのTSPITRまたは表領域プラグイン後のスタンバイで必要なアクション](#)

12.8.1 プライマリまたはスタンバイ・データベースでのファイルの損失からのリカバリ

必要なファイルが含まれるフィジカル・スタンバイ・データベースへ接続し、ネットワークを介してファイルをリストアおよびリカバリできます。

これは、プライマリ・データベースで失われたデータファイル、制御ファイルまたは表領域を、フィジカル・スタンバイ・データベース上の対応するファイルを使用してリストアする必要がある場合に有用です。同じプロセスを使用して、プライマリ・データベースを使用してフィジカル・スタンバイ・データベース上のファイルをリストアすることもできます。

ノート:



Oracle Database 12c より前のリリースでは、プライマリのファイルの損失からリカバリするには、RMAN リカバリ・カタログ、RMAN BACKUP、CATALOG DATAFILE および SWITCH DATAFILE コマンドを使用しました。スタンバイのファイルの損失からリカバリするには、RESTORE および RECOVER コマンドを使用しました。これらの方法は、Oracle Database 12c では不要になりました。これらの使用についての情報が必要な場合は、Oracle Database 11g のドキュメントを参照してください。

関連項目:

- ネットワークを介してファイルをリストアおよびリカバリする場合のRMANの使用の詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』](#)を参照してください。

12.8.2 オンラインREDOログ・ファイルの損失からのリカバリ

現行のACTIVEグループまたはアーカイブされていない非アクティブ・グループのオンライン・ログ・メンバーすべてが消失した場合、スタンバイ・データベースにフェイルオーバーする必要があります。

フェイルオーバー手順の詳細は[『ロールの推移』](#)を参照してください。

その他の状況でのオンラインREDOログ・ファイルの損失からのリカバリ方法の詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』](#)を参照してください。

12.8.3 プライマリ・データベースの不完全リカバリ

プライマリ・データベースの不完全リカバリを実行するのは、通常、データベースが論理的に(ユーザーまたはアプリケーションが原

因で)破損した場合や、表領域またはデータファイルがデータベースから意図せずに削除された場合などです。

スタンバイ・データベース・インスタンス上の現在のデータベース・チェックポイントSCNに応じて、次のいずれかの手順でプライマリ・データベースの不完全リカバリを実行できます。すべての手順は、最も所要時間が短いものから優先順位順になっています。

フラッシュバック・データベースの使用

ラッシュバック・データベースの使用は、プライマリ・データベースでフラッシュバック・データベース機能が使用可能で、データベース・ファイルが1つも消失しておらず、Point-in-Timeリカバリが最も古いフラッシュバックSCNより大きいか、最も早いフラッシュバック時間よりも後の場合の推奨手順です。フラッシュバック・データベースを使用してPoint-in-Timeリカバリを実行する手順の詳細は、[「Open Resetlogs文の発行後のフラッシュバック・データベースの使用」](#)を参照してください。

スタンバイ・データベース・インスタンスの使用

これは、スタンバイ・データベースが必要な不完全リカバリ時間より後のもので、プライマリまたはスタンバイ・データベースでフラッシュバック・データベースが使用可能でない場合の推奨手順です。

1. スタンバイ・データベースを必要な時点までリカバリします。次のコマンドを発行する前に、管理されているREDOプロセス(MRP)が停止していることを確認します。

```
RECOVER DATABASE UNTIL TIME ' time' ;
```

あるいは、SCNまたはログ順序番号を使用して不完全リカバリ時間を指定できます。

```
RECOVER DATABASE UNTIL SCN incomplete recovery SCN' ;  
RECOVER DATABASE UNTIL LOGSEQ incomplete recovery log sequence number THREAD thread number ;
```

2. スタンバイ・データベースを読取り専用モードでオープンし、データベースの状態を確認します。

必要な状態になっていない場合は、LogMinerユーティリティを使用してアーカイブREDOログ・ファイルを調べ、不完全リカバリに適切なターゲット時刻またはSCNを確認します。または、ターゲット時刻より前の判明している時点までスタンバイ・データベースをリカバリしてから、データベースを読取り専用モードでオープンし、データの状態を検査することもできます。データベースの状態が正常であると確認されるまで、このプロセスを繰り返します。データベースのリカバリ終了時点が遅すぎると(エラーが発生したSCNより後までリカバリすると)、それより前のSCNに戻せません。

3. SQL ALTER DATABASE ACTIVATE STANDBY DATABASE文を使用してスタンバイ・データベースをアクティブ化します。これにより、スタンバイ・データベースがプライマリ・データベースに変換されて、新しいリセットログ・ブランチが作成され、データベースがオープンします。スタンバイ・データベースが新しいリセット・ログ・ブランチにどのように対処するか学習するには、[「OPEN RESETLOGS文を使用したリカバリ」](#)を参照してください。

プライマリ・データベース・インスタンスの使用

すべてのスタンバイ・データベース・インスタンスがすでに必要な時点より後までリカバリされており、プライマリまたはスタンバイ・データベースでフラッシュバック・データベースが使用できない場合は、この方法で操作する必要があります。

次の手順に従ってプライマリ・データベースで不完全リカバリを実行します。

1. LogMinerまたは他の手段を使用して、データベースのすべてのデータが正常と判明している時点またはSCNを識別します。
2. その時点またはSCNを使用して次のRMANコマンドを発行し、不完全データベース・リカバリを実行し、RESETLOGSオプションを使用して(カタログ・データベースとMOUNT状態のプライマリ・インスタンスに接続した後で)データベースをオープンします。

```
RUN  
{  
SET UNTIL TIME ' time' ;  
RESTORE DATABASE ;
```



```
RECOVER DATABASE;  
}  
ALTER DATABASE OPEN RESETLOGS;
```

この処理の後に、Oracle Data Guard構成内ですべてのスタンバイ・データベース・インスタンスを再確立する必要があります。

12.8.4 プライマリでのTSPITRまたは表領域プラグイン後のスタンバイで必要なアクション

RMAN表領域Point-in-Timeリカバリ(TSPITR)がプライマリで実行された後、修復されたデータファイルは新しいシステム変更番号(SCN)を持つので、プライマリで新しいデータファイルのように扱われます。

これらのデータファイルはスタンバイで自動的に作成することはできません。

同様に、新しいプラグイン表領域がプライマリ・データベースに追加されると、データファイルはプライマリで新しいデータファイルのように扱われます。

Redo Applyプロセスがこれらの新しいファイルの作成を検出すると、スタンバイでの管理されたREDOプロセス(MRP)は停止します。必要な新しいデータファイルはコピーされ、スタンバイにリストアされる必要があります。バックアップまたはプライマリからの直接コピーのいずれかを使用してこれを行うことができます。たとえば、RMAN TSPITRが行われた表領域に属するすべてのファイルをコピーするには、次のRMANコマンドを使用できます。

```
RMAN> RESTORE TABLESPACE <tbs_name1, tbs_name2> FROM SERVICE <tnsalias-of-primary>
```

割り当てられるディスク・チャネルの数はRMAN構成ごとです。したがって、CONFIGURE DEVICE TYPE DISK PARALLELISM 4が実行されると、4つのディスク・チャネルを使用してプライマリ・データベースからファイルが取得されます。

新しいデータファイルがスタンバイで使用可能な場合、MRPを再起動してログの適用を継続します。

関連項目:

- RMAN TSPITRの詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

12.9 その他のバックアップ状況

スタンバイ・データベースとプライマリ・データベースでバックアップ・ファイルを共有できない構成、REDOログ・ファイルのリモート・アーカイブにスタンバイ・インスタンスのみが使用される構成、スタンバイ・データベースのファイル名がプライマリ・データベースとは異なる構成など、その他の構成に合わせてバックアップ手順を変更できます。

12.9.1 スタンバイ・データベースが地理的に離れすぎているためにバックアップを共有できない場合

スタンバイ・データベース間の距離が離れている場合、プライマリ・システムや他のスタンバイ・システムからは、これらの上で作成されたバックアップに簡単にアクセスできません。

すべてのシステム上でデータベースの完全バックアップを実行して、リカバリ操作を実行します。高速リカバリ領域は、プライマリおよびスタンバイ・システム上にローカルに置くことができます。プライマリ・データベースとスタンバイ・データベースで同一である必要はありません。

この使用例でも、[『Oracle Data Guard環境におけるリカバリ例』](#)で説明した一般的な対策を使用できますが、次の例外が

あります。

- RMANで作成されるバックアップ・ファイルには、タグとしてローカル・システム名を指定し、そのタグをRESTORE操作で使って、RMANによる同じホスト上で作成されたバックアップの選択を制限する必要があります。つまり、バックアップの作成時に、BACKUPコマンドではTAG system nameオプションを、RESTOREコマンドではFROM TAG system nameオプションを、RECOVERコマンドではFROM TAG system name ARCHIVELOG TAG system nameオプションをそれぞれ使用する必要があります。
- スタンバイ・サイトの障害時リカバリを実行する手順は、次のとおりです。

1. スタンバイの操作に使用されていたのと同じパラメータ・ファイルを使用して、スタンバイ・インスタンスをNOMOUNT状態で起動します。
2. プライマリ・インスタンスで、SQL ALTER DATABASE CREATE STANDBY CONTROLFILE AS filename文を使用してスタンバイ制御ファイルを作成し、作成された制御ファイルを使用してスタンバイ・インスタンスをマウントします。
3. 次のRMANコマンドを発行し、データベース・ファイルをリストアおよびリカバリします。

```
RESTORE DATABASE FROM TAG 'system name';  
RECOVER DATABASE FROM TAG 'system name' ARCHIVELOG TAG 'system name';
```

4. REDO Applyを再開します。

スタンバイ・インスタンスにより残りのアーカイブREDOログ・ファイルがフェッチされます。

12.9.2 FALサーバーとして使用されるスタンバイ・データベースにデータファイルが含まれていない場合

FALサーバーは、すべてのアーカイブREDOログ・ファイルのバックアップ・ソースとして使用できるため、アーカイブREDOログ・ファイルのバックアップをFALサーバーにオフロードします。

[「バックアップ手順」](#)で説明した手順を使用しますが、データベース・ファイルをバックアップするRMANコマンドはFALサーバーに対して実行できません。

12.9.3 スタンバイ・データベースのファイル名がプライマリ・データベースとは異なる場合

Oracle Database 11g現在、リカバリ・カタログは各スタンバイ・データベース・サイトのファイル名を再同期化できます。

ただし、まったく再同期化されていなかったプライマリ・データベースとスタンバイ・データベースでデータベース・ファイル名が異なる場合は、使用するRESTOREおよびRECOVERコマンドが若干異なります。スタンバイ・データベースで実際のデータファイル名を取得するには、V\$DATAFILEビューを問い合わせ、データベース内のすべてのデータファイルにSET NEWNAMEオプションを指定します。

```
RUN  
{  
SET NEWNAME FOR DATAFILE 1 TO 'existing file location for file#1 from V$DATAFILE';  
SET NEWNAME FOR DATAFILE 2 TO 'existing file location for file#2 from V$DATAFILE';  
...  
...  
SET NEWNAME FOR DATAFILE n TO 'existing file location for file#n from V$DATAFILE';  
RESTORE {DATAFILE <n,m,...> | TABLESPACE tbs_name_1, 2, ...} DATABASE;  
SWITCH DATAFILE ALL;  
RECOVER DATABASE {NOREDO};  
}
```

同様に、RMAN DUPLICATEコマンドのSET NEWNAMEオプションを使用してスタンバイ・データベースの作成時に新しいファイル名

を指定します。あるいは、LOG_FILE_NAME_CONVERTおよびDB_FILE_NAME_CONVERTパラメータを設定できます。

関連項目:

DB_FILE_NAME_CONVERTおよびDB_CREATE_FILE_DESTパラメータの両方がスタンバイに設定されている場合の優先ルールの詳細は、[「OMFまたはOracle ASMを使用するスタンバイ・データベースの作成」](#)を参照してください。

12.10 ネットワークを介したファイルのリストアおよびリカバリ

Oracle Database 12cでは、RMANを使用すると、必要なファイルが含まれるフィジカル・スタンバイ・データベースへネットワークを介して接続し、ファイルをリストアまたはリカバリできます。

データベース全体、データファイル、制御ファイル、spfileまたは表領域をリストアすることができます。プライマリ・データベースおよびスタンバイ・データベースを同期化する必要がある場合、ネットワークを介したファイルのリストアは、非常に有用です。

RMANは、RESTOREコマンドのFROM SERVICE句を使用し、フィジカル・スタンバイ・データベースからネットワークを介してデータベース・ファイルをリストアします。FROM SERVICE句では、ファイルをリストアする必要があるフィジカル・スタンバイ・データベースのサービス名を指定します。リストア操作では、リストアする必要があるファイルのバックアップ・セットがRMANによってフィジカル・スタンバイ・データベース上に作成され、これらのバックアップ・セットがネットワークを介してターゲット・データベースに転送されます。

ノート:



Oracle Database 12c より前のリリースでは、ネットワークを介してファイルをリストアまたはリカバリするには、RMAN の BACKUP INCREMENTAL FROM SCN コマンドを使用して、プライマリ・データベースでスタンバイの現行 SCN から始まるバックアップを作成し、それを使用してスタンバイ・データベースをロールフォワードしました。この手動の、複数のステップによる方法は、Oracle Database 12c では不要です。この方法の使用に関する情報が必要な場合は、Oracle Database 11g のドキュメントを参照してください。

関連項目:

- ネットワークを介してファイルをリストアおよびリカバリする場合のRMANの使用の詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。
- クロス・プラットフォームの増分バックアップを使用したトランスポータブル表領域の停止時間の短縮については、<http://support.oracle.com>にあるMy Oracle Supportノート2005729.1を参照してください。

12.11 1つコマンドを使用したスタンバイのロールフォワード

Oracle Database 18cでは、1つのRMANコマンドRECOVER STANDBY DATABASEを使用してネットワーク経由でスタンバイ・データベースをリフレッシュできます。

RECOVER STANDBY DATABASEコマンドは、スタンバイ・インスタンスを再起動し、プライマリ・データベースから制御ファイルをリフレッシュし、データファイル、一時ファイルおよびオンライン・ログの名前を自動的に変更します。プライマリ・データベースに追加された新しいデータ・ファイルをリストアし、スタンバイ・データベースを現時点までリカバリします。

RECOVER STANDBY DATABASEコマンドを使用してスタンバイ・データベースをリフレッシュするときは、FROM SERVICE句またはNOREDO句のいずれかを指定します。FROM SERVICE句は、プライマリ・サービスの名前を指定します。NOREDO句は、リフレッシュにバックアップを使用することを指定します。これにより、スタンバイを特定の時間またはSCNにロール・フォワードできます。

プライマリ・データベースとの同期化を試行する前に、スタンバイのMRPを手動で停止する必要があります。

RECOVER STANDBY DATABASEコマンドの使用例を次に示します。ここでは、オプションのPFILE句を使用してスタンバイ・データベースのパラメータ・ファイル(spfileが使用できない場合に使用)を指定しています。

```
RECOVER STANDBY DATABASE FROM SERVICE service_name PFILE=pfile_location;
```

関連項目:

- RECOVER STANDBY DATABASEコマンドを使用して、プライマリ・データベースで行われた変更を使用してフィジカル・スタンバイ・データベースをリフレッシュする方法の詳細は、[Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド](#)を参照してください。

12.12 Oracle Data Guard環境でのCDBに対するRMANサポート

RMANでは、スタンバイでのマルチテナント・コンテナ・データベース(CDB)のPoint-in-Timeリカバリ(PITR)がサポートされています。(個別のプラガブル・データベース(PDB)には個別のスタンバイはありません。)

これは、完全なデータベース・リカバリをサポートするためにRMANで提供され、スタンバイ・データベースでの完全なデータ・ファイル・リカバリを補完する追加機能です。

スタンバイでCDB PITRを実行するには、rootとしてCDBに接続し、必要に応じてRMAN BACKUP、RESTOREおよびRECOVERコマンドを発行します。

CDB PITRがスタンバイで実行される場合、無効状態だったプラガブル・データベース(PDB)がCDB PITRの前に有効になることに注意してください。PDBを無効状態に戻すには、PDBに接続し、閉じていることを確認し(V\$PDBSビューのOPEN_MODE列がMOUNTEDの値を示す)、SQL文ALTER PLUGGABLE DATABASE DISABLE RECOVERYを実行します。

ALTER PLUGGABLE DATABASE DISABLE RECOVERY文はPDBに属するすべてのデータ・ファイルをオフラインにし、PDBのリカバリを無効にします。PDBに属するデータ・ファイルはPDBが再度有効になるまで、リカバリ・セッションに含まれません。リカバリが無効な間に作成された新しいデータ・ファイルは、名前なしファイルとして作成され、オフラインにマークされます。

PDBに属するすべてのデータ・ファイルをオフラインに戻し、リカバリを有効にするには、PDBに接続し、閉じていることを確認し(V\$PDBSビューのOPEN_MODE列がMOUNTEDの値を示す)、SQL文ALTER PLUGGABLE DATABASE ENABLE RECOVERYを発行します。

PDBでリカバリが有効か無効かを確認するには、次のようにV\$PDBSビューを問い合わせます。

```
SQL> SELECT RECOVERY_STATUS FROM V$PDBS;
```

PDBのフラッシュバック

Oracle Database 12cリリース2 (12.2.0.1)より、FLASHBACK PLUGGABLE DATABASEコマンド(SQLまたはRecovery Managerを介して使用可能)を実行して特定のPDBでフラッシュバック操作を実行できます。CDBの他のPDBに影響を与えずに、過去特定のリストア・ポイント(システム変更番号(SCN)の別名)にフラッシュバックできます。(PDBでこのような操作を実行することは、非CDBでのFLASHBACK DATABASEに例えられます。)

スタンバイでPDBをフラッシュバックすることもできます。事実上、スタンバイでPDBをフラッシュバックすると、PDBのバックアップをリ

ストアしたかのように、PDBのデータ・ファイルが過去のある時点に巻き戻されます。スタンバイでPDBをフラッシュバックすることで、プライマリでPDB PITR/フラッシュバック操作を実行した後、スタンバイはすばやくプライマリのあとを継ぐことができます([「プライマリでのPDB PITR後にスタンバイに必要なアクション」](#)を参照)。

ノート:



ALTER PLUGGABLE DATABASE [ENABLE | DISABLE] 操作の結果として、オンラインまたはオフラインになったファイルは、その操作が実行される前のポイントにデータベースをフラッシュバックする場合でも、その状態のままです。

関連項目:

- プライマリでのPDB PITR後にスタンバイに必要なアクションの詳細は、[「プライマリでのPDB PITR後にスタンバイに必要なアクション」](#)を参照してください
- プライマリでのTSPITRまたは表領域プラグイン後にスタンバイに必要なアクションの詳細は、[「プライマリでのTSPITRまたは表領域プラグイン後のスタンバイに必要なアクション」](#)を参照してください

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

ロジカル・スタンバイ・データベースを使用して、Oracle Databaseソフトウェアのローリング・アップグレードを実行できます。

ローリング・アップグレード中は、プライマリ・データベースとロジカル・スタンバイ・データベースで異なるリリースのOracleデータベースを実行しながら、プライマリ・データベースの停止時間を最短に抑えて各リリースを一度に1つずつアップグレードできます。

Oracle Database 12cリリース1 (12.1)の最初のパッチ・セットで開始しているデータベースで、既存のフィジカル・スタンバイ・データベースを使用してローリング・アップグレードを実行する望ましい方法は、[DBMS_ROLLINGを使用したローリング・アップグレードの実行](#)で説明しているように、DBMS_ROLLING PL/SQLパッケージを使用する方法です。

次の各項では、Oracleデータベースのアップグレード中の停止時間を最短に抑える方法について説明します。

- [SQL Applyを使用するローリング・アップグレードのメリット](#)
- [SQL Applyを使用してローリング・アップグレードを実行するための要件](#)
- [アップグレード手順に使用する図と表記規則](#)
- [ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行](#)
- [既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行](#)
- [既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行](#)

ノート:



これらの項では、より停止時間が長い通常のアップグレード手順([「Oracle Data Guard 構成におけるデータベースのアップグレードおよびダウングレード」](#)を参照)に変わる方法について説明します。2つの手順のステップを組み合わせようとししないでください。

13.1 SQL Applyを使用するローリング・アップグレードのメリット

SQL Applyを使用してローリング・アップグレードを実行すると、次のようなメリットがあります。

- 本番データベースの停止時間が最短で済みます。停止時間全体がスイッチオーバーの実行所要時間と同様に短時間です。
- PL/SQLの再コンパイルによるアプリケーションの停止時間は発生しません。
- プライマリ・データベースに影響を与えずに、アップグレード後のデータベース・リリースを検証できます。
- ロジカル・スタンバイはアップグレードが実行されている間、アーカイブされたログを受け入れます。これにより障害保護のレベルが上がります。

ノート:



- Oracle Database 12c リリース 1 (12.1)現在、Oracle XML DB Repository では Oracle

Data Guard のローリング・アップグレードをサポートしています。このサポートに関して注意が必要な考慮事項および制限事項の詳細は、[『Oracle XML DB 開発者ガイド』](#)を参照してください。

- Oracle Database 12c リリース 1 (12.1)以降では、Oracle Database Vault を使用するデータベースを、Oracle Data Guard データベースのローリング・アップグレードを一時ロジカル・スタンバイおよび PL/SQL パッケージ DBMS_ROLLING とともに使用して、新しい Oracle データベースのリリースおよびパッチ・セットにアップグレードできます。
- Oracle Database 12c リリース 1 (12.1)以降では、Oracle Label Security (OLS)を使用するデータベースを、一時ロジカル・スタンバイ・データベースおよび PL/SQL パッケージ DBMS_ROLLING を使用する Oracle Data Guard データベースのローリング・アップグレードを使用して、新しい Oracle データベースのリリースおよびパッチ・セットにアップグレードできます。

13.2 SQL Applyを使用してローリング・アップグレードを実行するための要件

SQL Applyを使用してローリング・アップグレードを実行するための要件は、次のとおりです。

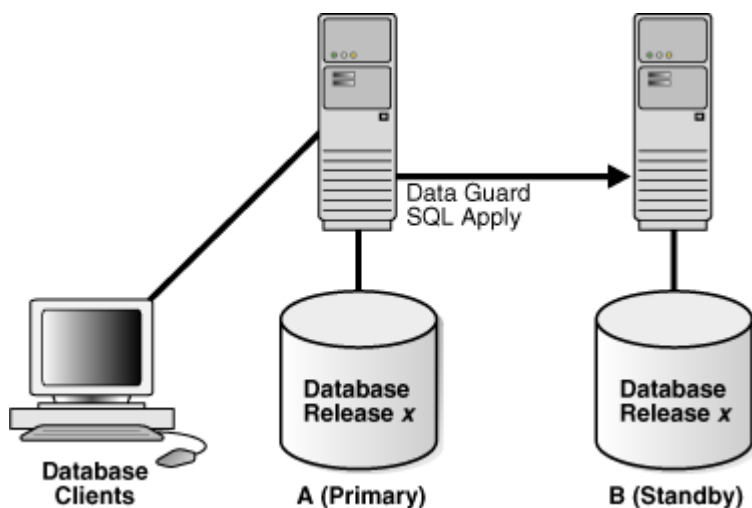
- データベースがOracle Data Guard Broker構成の一部の場合、ローリング・アップグレードの前にブローカ構成を無効にします。ブローカ構成の無効化の詳細は、[『Oracle Data Guard Broker』](#)を参照してください。
- Oracle Data Guardの保護モードが最大可用性モードまたは最大パフォーマンス・モードに設定されていること。現在の保護モード設定を確認するには、V\$DATABASEビューのPROTECTION_LEVEL列を問い合わせます。
- ロジカル・スタンバイ・データベースのアップグレード中もプライマリ・データベースで処理を進行できるようにするために、ロジカル・スタンバイ・データベースの宛先のLOG_ARCHIVE_DEST_*n*初期化パラメータがMANDATORYに設定されていないこと。
- プライマリ・データベースに設定されたCOMPATIBLE初期化パラメータが、アップグレード前のソフトウェア・リリースと一致していること。したがって、リリースxからリリースyへのローリング・アップグレードでは、プライマリ・データベースでCOMPATIBLE初期化パラメータをリリースxに設定する必要があります。ローリング・アップグレード・スタンバイ・データベースでは、COMPATIBLE初期化パラメータをx以上に設定する必要があります。

13.3 アップグレード手順に使用する図と表記規則

この背景情報は、アップグレードの実行手順を理解するのに役立ちます。

[図13-1](#)に、アップグレード開始前のOracle Data Guard構成を示します。この例では、プライマリ・データベースとロジカル・スタンバイ・データベースで同じリリースのOracle Databaseソフトウェアが実行されています。

図13-1 「アップグレード前のOracle Data Guard構成」



アップグレード処理中に、Oracle Data Guard構成は何度か混合データベース・リリースで動作します。リリース間にまたがるデータ保護は使用できません。これらのステップでは、データ保護を提供するためにOracle Data Guard構成に第2のスタンバイ・データベースがある場合を考えます。

アップグレード手順のステップと図では、データベースを「プライマリ・データベース」と「スタンバイ・データベース」ではなく「データベースA」および「データベースB」と呼びます。これは、アップグレード手順の実行中にデータベースのロールが切り替わるためです。最初は、[図13-1](#)に示すようにデータベースAがプライマリ・データベースでデータベースBがロジカル・スタンバイ・データベースです。

次の各項では、SQL Applyのローリング・アップグレード手順を使用できる例について説明します。

- [ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行](#)
- [既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行](#)
- [既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行](#)

13.4 ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行

この使用例では、既存のOracle Data Guard構成がなく、Oracle Databaseのローリング・アップグレードを実行するためにのみロジカル・スタンバイ・データベースを作成することを前提にしています。

[表13-1](#)に、プライマリ・データベースとスタンバイ・データベースでアップグレードを準備するステップを示します。

表13-1 ロジカル・スタンバイの新規作成によってローリング・アップグレードを実行するステップ

ステップ	説明
ステップ 1	サポートされないデータ型および記憶域属性を識別する
ステップ 2	ロジカル・スタンバイ・データベースを作成する
ステップ 3	ローリング・アップグレードを実行する

1. 次を実行して、プライマリ・データベースでサポートされないデータベース・オブジェクトを識別し、その処理方法を決定します。

- [「ロジカル・スタンバイ・データベースでサポートされるデータ型およびDDL」](#)を参照し、サポートされるデータ型および記憶域属性のリストを確認します。
- プライマリ・データベースでDBA_LOGSTDBY_UNSUPPORTEDおよびDBA_LOGSTDBY_SKIPビューを問い合わせます。

プライマリ・データベースで表示された表およびスキーマに対して行われた変更は、ロジカル・スタンバイ・データベースに適用されません。次の問合せを使用して、サポートされない表のリストを確認します。

```
SQL> SELECT DISTINCT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED;
```

次の問合せを使用して、サポートされない内部スキーマのリストを確認します。

```
SQL> SELECT OWNER FROM DBA_LOGSTDBY_SKIP -  
> WHERE STATEMENT_OPT = 'INTERNAL_SCHEMA';
```

2. [「ロジカル・スタンバイ・データベースの作成」](#)の手順に従ってロジカル・スタンバイ・データベースを作成します。

ノート:

SQL Apply を初めて実行する前に、プライマリ・データベースで実行中の、ロジカル・スタンバイ・データベースではサポートされないトランザクションに関する情報を取得します。次のプロシージャを実行して情報を取得し、DBA_LOGSTDBY_EVENTS ビューにイベントとして記録します。

```
EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_EVENTS_RECORDED',  
    DBMS_LOGSTDBY.MAX_EVENTS);  
  
EXECUTE DBMS_LOGSTDBY.APPLY_SET('RECORD_UNSUPPORTED_OPERATIONS',  
    'TRUE');
```

停止時間を最短に抑えるために、ロジカル・スタンバイ・データベースでスタンバイREDOログを構成することをお勧めします。

3. ロジカル・スタンバイ・データベースを作成したので、ローリング・アップグレードを実行します。[「既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行」](#)で説明している手順を実行します。この手順では、ロジカル・スタンバイで同じOracleソフトウェアを実行していることを前提としています。

13.5 既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行

このステップでは、ロジカル・スタンバイ・データベースおよびプライマリ・データベースでOracle Databaseのローリング・アップグレードを実行する方法について説明します。

[表13-2](#)にステップを示します。

表13-2 既存のロジカル・スタンバイを使用したローリング・アップグレードの実行ステップ

ステップ	説明
ステップ 1	ローリング・アップグレードの準備をする
ステップ 2	ロジカル・スタンバイ・データベースをアップグレードする
ステップ 3	アップグレード後のロジカル・スタンバイ・データベースで SQL Apply を再開する

ステップ	説明
ステップ 4	アップグレード後のスタンバイ・データベースでイベントを監視する
ステップ 5	スイッチオーバーを開始する
ステップ 6	アップグレード中に変更されたすべての表をインポートする
ステップ 7	スイッチオーバーを完了してユーザー・アプリケーションをアクティブ化する
ステップ 8	古いプライマリ・データベースをアップグレードする
ステップ 9	古いプライマリ・データベースで SQL Apply を開始する
ステップ 10	必要な場合は、両方のデータベースの互換性レベルを上げる
ステップ 11	新規ロジカル・スタンバイ・データベースでイベントを監視する
ステップ 12	必要な場合は、再度スイッチオーバーを実行する

1. 次のステップに従って、Oracleソフトウェアのローリング・アップグレードの実行準備をします。

- a. ロジカル・スタンバイ・データベース(データベースB)で次の文を発行し、SQL Applyを停止します。

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

- b. 必要に応じて、互換性を最高値に設定します。

COMPATIBLE初期化パラメータに、アップグレード前のプライマリ・データベースで実行されているOracle Databaseソフトウェアのリリース番号が指定されていることを確認します。

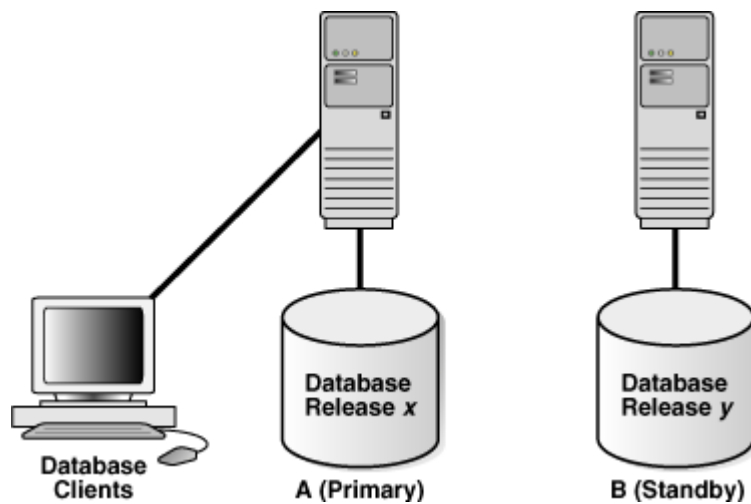
たとえば、プライマリ・データベースでリリース11.1が実行されている場合は、COMPATIBLE初期化パラメータを両方のデータベースで11.1に設定します。COMPATIBLE初期化パラメータは、必ずスタンバイ・データベースで設定してからプライマリ・データベースで設定してください。

2. ロジカル・スタンバイ・データベース(データベースB)のOracle Databaseソフトウェアをリリースyにアップグレードします。ロジカル・スタンバイ・データベースでは、アップグレード中、プライマリ・データベースからのREDOデータを受け入れません。

Oracle Databaseソフトウェアをアップグレードするには、該当するOracle Databaseリリースの『Oracle Databaseアップグレード・ガイド』を参照してください。

[図13-2](#)に、リリースxを実行中のデータベースAとリリースyを実行中のデータベースBを示します。アップグレード中は、REDOデータがプライマリ・システムに蓄積されます。

図13-2 ロジカル・スタンバイ・データベースのリリースのアップグレード

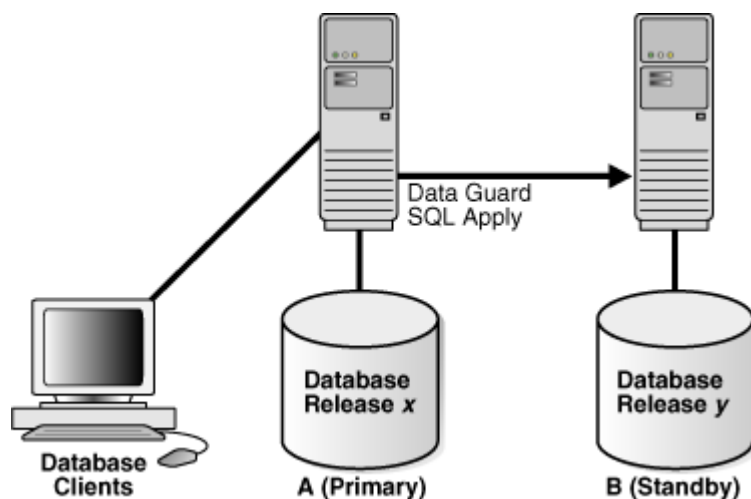


3. SQL Applyを再開し、データベースAではリリースx、データベースBではリリースyで動作させます。SQL Applyを開始するには、データベースBで次の文を発行します。

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

プライマリ・システムに蓄積されたREDOデータは、新しくアップグレードしたロジカル・スタンバイ・データベースに自動的に転送され、適用されます。Oracle Data Guard構成では、アップグレード後のOracle Databaseソフトウェア・リリースが本番環境で正常に実行されるかどうかを確認するために、任意の期間のみ図13-3に示す混合リリースを実行できます。

図13-3 混合リリースの実行



Database BがDatabase Aにどの程度速く追いつくかを監視するには、Database BのV\$LOGSTDBY_PROGRESSビューの問合せを行います。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 SYSDATE, APPLIED_TIME FROM V$LOGSTDBY_PROGRESS;
```

SYSDATE	APPLIED_TIME
27-JUN-05 17:07:06	27-JUN-05 17:06:50

4. 頻繁にDBA_LOGSTDBY_EVENTSビューを問い合わせ、データベースBに適用されなかったDDL文とDML文があるかどうかを確認することをお勧めします。

```
SQL> SET LONG 1000
SQL> SET PAGESIZE 180
SQL> SET LINESIZE 79
SQL> SELECT EVENT_TIMESTAMP, EVENT, STATUS FROM DBA_LOGSTDBY_EVENTS -
```

```
> ORDER BY EVENT_TIMESTAMP;
```

```
EVENT_TIMESTAMP
```

```
EVENT
```

```
STATUS
```

```
...
```

```
24-MAY-05 05.18.29.318912 PM
```

```
CREATE TABLE SYSTEM.TST (one number)
```

```
ORA-16226: DDL skipped due to lack of support
```

```
24-MAY-05 05.18.29.379990 PM
```

```
"SYSTEM". "TST"
```

```
ORA-16129: unsupported dml encountered
```

前述の例は、次のとおりです。

a. ORA-16226エラーは、サポートできないDDL文を示します。この場合は、内部スキーマに属しているためにサポートできません。

b. ORA-16129エラーは、適用されなかったDML文を示します。

この種のエラーは、データベースAで発生した変更の一部がデータベースBに適用されなかったことを示します。この時点で、アップグレード手順を続行するかどうかを判断する必要があります。ロジカル・スタンバイ・データベースとプライマリ・データベース間で、この差異が許容可能であることが確実な場合は、アップグレード手順を続行します。不確実な場合は、アップグレード手順を中断してデータベースBを再インスタンス化し、別の時点でアップグレード手順を実行します。

5. アップグレード後のデータベース・ソフトウェアが正常に動作していることを確認した後、データベースAで次の文を発行してスイッチオーバーを実行し、データベース・ロールを元に戻します。

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

この文は、既存のトランザクションが完了まで待機する必要があります。スイッチオーバーの完了所要時間を最短に抑えるには、データベースAにまだ接続されているユーザーがすぐにログオフし、データベースBに再接続する必要があります。

ノート:



[「ロジカル・スタンバイ・データベースへのスイッチオーバーの実行」](#)で説明した通常の2フェーズ準備済スイッチオーバーは、同じバージョンの Oracle ソフトウェアの実行にプライマリ・データベースとスタンバイ・データベースの両方が必要であり、この時点でプライマリ・データベースは下位バージョンの Oracle ソフトウェアを実行しているため、使用できません。かわりに、前述の1フェーズ準備なしスイッチオーバーの手順を使用します。準備なしスイッチオーバーは、ロジカル・スタンバイ・データベースを使用するローリング・アップグレードの場合にのみ使用してください。

ノート:



データベースAがプライマリ・データベースのときに、そこでサポートされない表またはパッケージに対するアクティビティを一時停止した場合、最終的にデータベースAに切り替える計画であれば、データベースBがプライマリ・データベースになっている間にデータベースBでも引き続き同じアクティビティを一時停止する必要

があります。

6. ステップ4では、変更中のサポートされない表をリストする方法について説明しました。プライマリ・データベースでサポートされないDML文が発行された場合は、Oracle Data Pumpなどのインポート・ユーティリティを使用して、各表の最新バージョンをインポートします。

たとえば、次のインポート・コマンドはscott. emp表を切り捨て、前のプライマリ・データベース(A)と一致するデータを移入します。

```
impdp SYSTEM NETWORK_LINK=databasea TABLES=scott. emp TABLE_EXISTS_ACTION=TRUNCATE
```

このコマンドは、実行前にimpdpパスワードの入力を求めてきます。

7. アップグレード後のデータベース・ソフトウェアが正常に動作していることを確認した後、スイッチオーバーを完了してデータベース・ロールを元に戻します。

- a. データベースBで、次のようにV\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせます。

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

```
SWITCHOVER_STATUS
```

```
-----  
TO PRIMARY
```

- b. SWITCHOVER_STATUS列にTO PRIMARYが表示される場合は、データベースBで次の文を発行してスイッチオーバーを完了します。

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

ノート:

[「ロジカル・スタンバイ・データベースへのスイッチオーバーの実行」](#)で説明した通常の2フェーズ準備済スイッチオーバーは、同じバージョンのOracleソフトウェアの実行にプライマリ・データベースとスタンバイ・データベースの両方が必要であり、この時点でプライマリ・データベースは下位バージョンのOracleソフトウェアを実行しているため、使用できません。かわりに、前述の1フェーズ準備なしスイッチオーバーの手順を使用します。準備なしスイッチオーバーは、ロジカル・スタンバイ・データベースを使用するローリング・アップグレードの場合にのみ使用してください。

- c. 現在はプライマリ・データベース・ロールで実行中のデータベースBで、ユーザー・アプリケーションとサービスをアクティブ化します。

スイッチオーバー後は、新規データベース・ソフトウェア・リリースを実行中の新規プライマリ・データベース(B)から旧ソフトウェア・リリースを実行中の新規スタンバイ・データベース(A)には、REDOデータを送信できません。これは次のことを意味します。

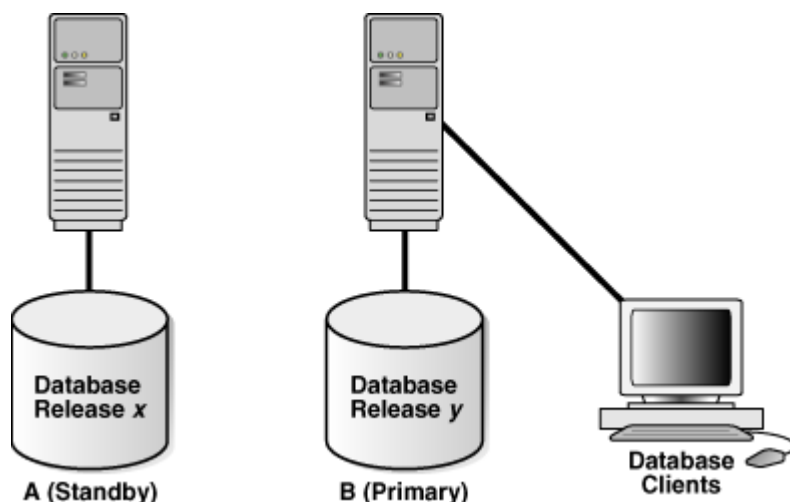
- d. REDOデータは、新規プライマリ・データベースに蓄積されます。
e. 新規プライマリ・データベースは、この時点では保護されていません。

[図13-4](#)に、前はスタンバイ・データベース(リリースyを実行)で現在はプライマリ・データベースになっているデータベースBと、前はプライマリ・データベース(リリースxを実行)で現在はスタンバイ・データベースになっているデータベースAを示しま

す。ユーザーはデータベースBに接続されます。

データベースBがプライマリ・データベースとして適切に機能でき、ビジネスにプライマリ・データベースをサポートするためのロジカル・スタンバイ・データベースが不要な場合、これでローリング・アップグレード処理は完了です。ユーザーにデータベースBへのログインとそこでの作業開始を許可し、問題のない時点でデータベースAを破棄します。それ以外の場合は、ステップ8に進みます。

図13-4 スイッチオーバー後

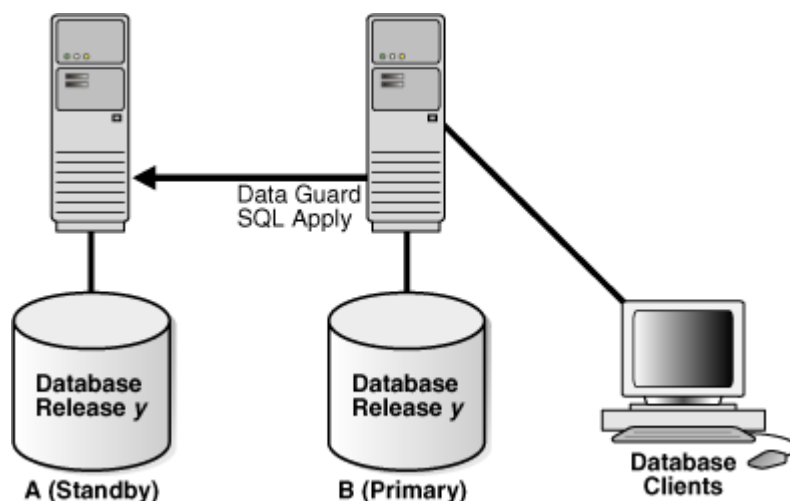


8. データベースAでは引き続きリリースxが実行されており、アップグレードしてSQL Applyを開始するまでは、データベースBからのREDOデータを適用できません。

Oracle Databaseソフトウェアのアップグレードの詳細は、『[Oracle Databaseアップグレード・ガイド](#)』を参照してください。

図13-5に、両方のデータベースがアップグレードされた後のシステムを示します。

図13-5 両方のデータベースがアップグレードされた後



9. データベースAで次の文を発行してSQL Applyを開始し、必要に応じてデータベースBへのデータベース・リンクを作成します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE NEW PRIMARY db_link_to_b;
```

ノート:



データベース・リンクを作成し(まだ設定されていない場合)、NEW PRIMARY 句を使用する必要があります。これは、ステップ4で1フェーズ準備なしスイッチオーバーを使用してデータベースAをスタンバイ・データ

ベースに切り替えたためです。

ユーザー-SYSTEMとして接続するか、同様のレベルの権限を持つアカウントで接続する必要があります。

データベースAでSQL Applyを開始すると、プライマリ・データベース(B)に蓄積されたREDOデータがロジカル・スタンバイ・データベース(B)に送信されます。すべてのREDOデータがスタンバイ・データベースで使用可能になると、プライマリ・データベースはデータ消失から保護されます。

10. COMPATIBLE初期化パラメータを設定して、両方のデータベースの互換性レベルを上げます。互換性レベルは、ロジカル・スタンバイ・データベースで上げてから、プライマリ・データベースで上げる必要があります。COMPATIBLEパラメータは、スタンバイ・データベースで設定してからプライマリ・データベースで設定します。COMPATIBLE初期化パラメータの詳細は、『[Oracle Databaseリファレンス](#)』を参照してください。

11. Database Bで行ったすべての変更がロジカル・スタンバイ・データベース(A)に正しく適用されていることを確認するには、ステップ4でDatabase Aに対して行ったように、DBA_LOGSTDBY_EVENTSビューの問合せを頻繁に行う必要があります。

変更が行われたためにデータベースAが既存のプライマリ・データベースのコピーとして無効になった場合は、データベースAを破棄し、かわりに新規のロジカル・スタンバイ・データベースを作成できます。詳細は、『[ロジカル・スタンバイ・データベースの作成](#)』を参照してください。

12. 必要な場合は、データベースAがプライマリ・データベース・ロールで再実行されるように([図13-1](#)を参照)、データベースのスイッチオーバーを再度実行します。

ノート:



この時点で、データベース A とデータベース B はいずれも同じバージョンの Oracle ソフトウェアを実行しているため、[ロジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)で説明した 2 フェーズ準備済スイッチオーバーを使用します。

13.6 既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行

このステップでは、Oracle Databaseソフトウェアのローリング・アップグレードを実行した後に元の構成に戻し、Aがプライマリ・データベース、Bがフィジカル・データベースで、いずれのデータベースもアップグレード後のOracleソフトウェアを実行している状態にする方法について説明します。

ノート:



これらのステップでは、プライマリ・データベース(A)とフィジカル・スタンバイ・データベース(B)が設定済で、Oracle Database リリース 11.1 以降を使用していることを前提にしています。

[表13-3](#)に關係するステップをまとめます。

表13-3 既存のフィジカル・スタンバイを使用したローリング・アップグレードの実行ステップ

ステップ	説明
ステップ 1	ローリング・アップグレードのためにプライマリ・データベースを準備する(データベース A でこのステップを実行する)
ステップ 2	フィジカル・スタンバイ・データベースをロジカル・スタンバイ・データベースに変換する(データベース B でこのステップを実行する)
ステップ 3	ロジカル・スタンバイ・データベースをアップグレードしてプライマリ・データベースと一致させる(データベース B でこのステップを実行する)
ステップ 4	データベース A を保証付きリストア・ポイントにフラッシュバックする(データベース A でこのステップを実行する)
ステップ 5	新しいバージョンの Oracle ソフトウェアを使用して、データベース B をマウントする
ステップ 6	データベース A をフィジカル・スタンバイに変換する
ステップ 7	データベース A で管理リカバリを開始する
ステップ 8	スイッチオーバーを実行してデータベース A をプライマリ・データベースにする
ステップ 9	データベース A に作成された保証付きリストア・ポイントをクリーンアップする

1. ローリング・アップグレードのためにプライマリ・データベースを準備する(データベースAでこのステップを実行する)

- a. まだ有効でない場合、フラッシュバック・データベースを有効にします。

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE FLASHBACK ON;
SQL> ALTER DATABASE OPEN;
```

- b. 保証付きリストア・ポイントを作成します。

```
SQL> CREATE RESTORE POINT pre_upgrade GUARANTEE FLASHBACK DATABASE;
```

2. フィジカル・スタンバイ・データベースをロジカル・スタンバイ・データベースに変換する(データベースBでこのステップを実行する)

- a. 次の点を除き、[「ロジカル・スタンバイ・データベースの作成」](#)に記載のステップに従います。[「ロジカル・スタンバイ・データベースへの変換」](#)では、別なコマンドを使用してロジカル・スタンバイ・データベースの変換を行う必要があります。ALTER DATABASE RECOVER TO LOGICAL STANDBY db_nameのかわりに、次のコマンドを実行します。

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

- b. 初めてSQL Applyを実行する場合は、その前に次のアクションを行う必要があります。

- i. 次のようにして、ロジカル・スタンバイで外部アーカイブ・ログの自動削除を無効にします。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET(' LOG_AUTO_DELETE', ' FALSE');
```

ノート:



ロジカル・スタンバイ・データベース(データベース B)で処理されたリモート・アーカイブ・ログは削除しないでください。これらのリモート・アーカイブ・ログは、後でローリング・アップグレード処理の際に必要になります。リカバリ領域を使用してリモート・アーカイブ・ログを格納している場合は、ロジカル・スタンバイ・データベースの通常動作を妨げることなくこれらのログを収容するのに十分な領域があることを確認してください。

- ii. プライマリ・データベースで実行されていて、ロジカル・スタンバイ・データベースでサポートされないトランザクションに関する情報が取得されていることを確認します。次のプロシージャを実行して情報を取得し、DBA_LOGSTDBY_EVENTS表にイベントとして記録します。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET(' MAX_EVENTS_RECORDED', -  
> DBMS_LOGSTDBY.MAX_EVENTS);
```

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET(' RECORD_UNSUPPORTED_OPERATIONS', ' TRUE');
```

- iii. 次のようにして、初めてのSQL Applyを開始します。

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

関連項目:

- iv. DBA_LOGSTDBY_EVENTSビューの詳細は、[「DBA_LOGSTDBY_EVENTSビューでのイベントのロギングのカスタマイズ」](#)を参照してください
- v. DBMS_LOGSTDBYパッケージの詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)を参照してください。

3. [「既存のロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行」](#)で説明されているとおり、ステップ1から7を実行できるようになりました。これらのステップが完了すると、データベースBはアップグレードされたバージョンのOracleソフトウェアを実行するプライマリ・データベースになり、データベースAはロジカル・スタンバイ・データベースになります。

次のステップに進んで、データベースAをデータベースBのフィジカル・スタンバイにします。

4. データベースAを保証付きリストア・ポイントにフラッシュバックする(データベースAでこのステップを実行する)

```
SQL> SHUTDOWN IMMEDIATE;  
SQL> STARTUP MOUNT;  
SQL> FLASHBACK DATABASE TO RESTORE POINT pre_upgrade;  
SQL> SHUTDOWN IMMEDIATE;
```

5. この時点で、上位バージョンのOracleソフトウェアを使用できるように、データベースAを切り替えます。データベースAはフィジカル・スタンバイに切り替えられており、データベースBによって生成されたREDOデータを適用して自動的にアップグレードされるため、アップグレード・スクリプトは実行しないでください。

次のようにして、データベースAをマウントします。

```
SQL> STARTUP MOUNT;
```

6. データベースAをフィジカル・スタンバイに変換する

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;  
SQL> SHUTDOWN IMMEDIATE;
```

7. データベースAで管理リカバリを開始する

データベースAは、データベースBにより生成されたREDOデータを適用して、自動的にアップグレードされます。管理リカバリは、プライマリからの新しいインカネーション・ブランチが登録されるまで待機した後、REDOの適用を開始します。

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

ノート:



REDO Apply が再開するとき、現行のプライマリ・データベース(データベース B)からの新しいインカネーションが登録されるのを待機します。

8. この時点では、Database Bがプライマリ・データベースでDatabase Aがフィジカル・スタンバイで、両方ともOracleソフトウェアの上位バージョンを実行しています。Database Aをプライマリ・データベースにするには、「[フィジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)」に記載のステップを実行します。

9. ディスク領域を保持するために、データベースAで作成された既存の保証付きリストア・ポイントを削除します。

```
SQL> DROP RESTORE POINT PRE_UPGRADE;
```

関連項目:

Oracle Maximum Availability Architecture (MAA)のホームページで入手できるベスト・プラクティスに関する技術概要『Database Rolling Upgrade Using Transient Logical Standby: Oracle Data Guard 11g』

<http://www.oracle.com/goto/maa>

14 DBMS_ROLLINGを使用したローリング・アップグレードの実行

Oracle Active Data Guardを使用したローリング・アップグレード機能は、ローリング・アップグレードを実行する効率的な方法を提供します。

これはDBMS_ROLLING PL/SQLパッケージを使用すると実装され、Oracle Data Guard構成内のデータベース・ソフトウェアをローリング方式でアップグレードできます。Oracle Active Data Guardを使用したローリング・アップグレード機能には、Oracle Active Data Guardオプションのライセンスが必要です。

この機能は、Oracle Database 12cの最初パッチセット以降、データベース・バージョン・アップグレードを実行するために使用できます。最初のOracle Database 12cパッチセットより前のバージョンからアップグレードするためには使用できません。つまり、Oracle Database 11gからOracle Database 12cへのアップグレード時またはOracle Database 12cの初期リリースからOracle Database 12cの最初のパッチセットへのアップグレード時には、手動の一時ロジカル・スタンバイ・アップグレード手順を引き続き使用する必要があります。

DBMS_ROLLINGパッケージは、Oracle Data Guardのスイッチオーバーを実行してプライマリ・データベース・サービスの停止時間を最小化します。DBMS_ROLLINGを使用する前に、スイッチオーバーを受け入れるようにOracle Data Guard環境を適切に構成する必要があります。設定要件は、Oracle Data Guard BrokerがDBMS_ROLLINGの実行中にアクティブかどうかによって異なります。

- ブローカがDBMS_ROLLING中にアクティブになる場合、ブローカにスイッチオーバーを設定する詳細について、[Oracle Data Guard Broker](#)を参照してください。
- ブローカがDBMS_ROLLING中にアクティブにならない場合、[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。Oracle Data Guard Brokerが存在しないということは、LOG_ARCHIVE_DEST_nパラメータターゲット・プライマリ・データベース上でREDO送信がスイッチオーバー後に再開するように適切に構成する必要があることを意味します。

また、この機能は、Oracle Database 12c リリース1 (12.1)以降、他のデータベース・メンテナンス・タスクにすぐに使用できます。メンテナンスが実行されるデータベースは、Oracle 12.1以上で動作する必要があります。このようなメンテナンス・タスクには、次のものが含まれます。

- パーティション化されていない表へのパーティションの追加
- BasicFiles LOBからSecureFiles LOBへの変更
- CLOBとして格納されるXMLTypeからバイナリXMLとして格納されるXML typeへの変更
- OLTP圧縮化対象への表の変更

次のトピックを参照してください。

- [ローリング・アップグレードの新しい概念](#)
- [DBMS_ROLLINGアップグレードとCDB](#)
- [DBMS_ROLLINGの使用の概要](#)
- [ローリング・アップグレードの計画](#)
- [ローリング・アップグレードの実行](#)
- [ローリング・アップグレードの監視](#)

- [ローリング・アップグレードのロールバック](#)
- [ローリング・アップグレード中に発生したロール変更の処理](#)
- [ローリング・アップグレードの例](#)

14.1 ローリング・アップグレードの新しい概念

Oracle Data Guard構成内のデータベース・ソフトウェアをローリング方式でアップグレードするには、最初にフィジカル・スタンバイを将来のプライマリ・データベースとして指定します。

概念的には、ローリング・アップグレード・プロセスはOracle Data Guard構成を2つのグループに分割します。先行グループ (LG)と後続グループ(TG)です。

先行グループのデータベースは先にアップグレードされるので先行グループという名前です。先行グループには、指定した将来のプライマリ・データベースと、それを保護するように構成できるフィジカル・スタンバイが含まれます。将来のプライマリは最初にロジカル・スタンバイ・データベースに変換されてから、新しいデータベース・ソフトウェアがそこにインストールされ、アップグレード・プロセスが実行します。先行グループのその他のスタンバイ・データベースも、この時点でそのソフトウェアをアップグレードする必要があります。

後続グループには、元のプライマリ・データベースと、ローリング・アップグレード・プロセス中に元のプライマリを保護するように構成できるフィジカル・スタンバイが含まれます。先行グループのデータベースのアップグレード・プロセス中は、ユーザー・アプリケーションは引き続き元のプライマリに接続し、変更できます。先行グループのすべてのデータベースがアップグレードされ、アップグレード・ウィンドウ中に元のプライマリ・データベースで生成された変更が適用されることにより、将来のプライマリが元のプライマリに追いつくまで、後続グループのデータベースは古いデータベース・ソフトウェアの実行を継続します。この時点で、スイッチオーバーはプライマリ・ロールの指定した将来のプライマリ・データベースへの転送を完了し、ユーザー・アプリケーションは新しいプライマリ・データベースにスイッチオーバーされています。次に新しいソフトウェアが後続グループの一部であるデータベースにインストールされ、新しいプライマリ・データベースへのスタンバイとして構成内に回復されます。

それぞれのグループのスタンバイは先行グループ・スタンバイ(LGS)および後続グループ・スタンバイ(TGS)と呼ばれます。指定した将来のプライマリ以外の、先行グループのその他のすべてのスタンバイは、フィジカル・スタンバイにのみなることができます。後続グループはフィジカルおよびロジカル・スタンバイの両方を含むことができます。そのスタンバイ・タイプを区別する必要がある場合は、後続グループ・フィジカル(TGP)と後続グループ・ロジカル(TGL)と呼びます。指定した将来のプライマリも、先行グループ・マスター(LGM)と呼ばれ、元のプライマリ・データベースは後続グループ・マスター(TGM)と呼ばれます。

DBMS_ROLLINGパッケージは次のように、ローリング・アップグレード・プロセスの堅牢性を向上させます。

- ローリング・アップグレード・プロセス中にエラーを処理できます。元のプライマリまたはTGMデータベースの失敗が許されます。後続グループ内のその他のフィジカル・スタンバイには通常のフェイルオーバー操作を開始でき、その後で新しいプライマリ・データベースをTGMとして指定します。
- ローリング・アップグレード・プロセス中にLGM (指定した将来のプライマリ)のデータ保護が可能です。LGMデータベースへのフィジカル・スタンバイを設定でき、アップグレード・プロセス中にそれを保護でき、アップグレード後にデータ消失ゼロも達成できます。LGMが正常にアップグレードされたら、フィジカル・スタンバイ・データベースへのフェイル・オーバーにより、LGMでの失敗は調整できます。次にファイルオーバー・ターゲット・データベースを指定して、LGMのロールを引き継ぐことができます。

[表14-1](#)では、スイッチオーバー操作の前と後の、TGPスタンバイとLGPスタンバイの特徴を比較します。

表14-1 後続グループ・フィジカル(TGP)と先行グループ・フィジカル(LGP)

スタンバイ・タイプ	スイッチオーバー前	スイッチオーバー後	ノート

スタンバイ・タイプ	スイッチオーバー前	スイッチオーバー後	ノート
後続グループ・フィジカル (TGP)	低い適用ラグ より低いデータ消失リスク	高い適用ラグ より高いデータ消失リスク	プライマリ・ロールへのフェイルオーバーが可能 元のプライマリのようにフラッシュバックが必要
先行グループ・フィジカル (LGP)	高い適用ラグ より高いデータ消失リスク	低い適用ラグ より低いデータ消失リスク	一時ロジカル・スタンバイ・ロールへのフェイルオーバーが可能 元のプライマリのようなフラッシュバックは不要

関連項目:

- DBMS_ROLLING PL/SQLパッケージの詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)
- DBMS_ROLLING PL/SQLパッケージを使用したアップグレードの実行中に、アップグレードに含まれるいずれかの表にサポートされないデータ型が含まれるかどうかを判別する方法の詳細は、[「ローリング・アップグレード中にサポートされない表」](#)を参照してください。
- DBMS_ROLLINGアップグレードの場合でのみサポートされるPL/SQLパッケージの詳細は、[「DBMS_ROLLINGアップグレードの場合でのみ使用可能な追加のPL/SQLパッケージのサポート」](#)を参照してください。

14.1.1 Data GuardブローカでのDBMS_ROLLINGアップグレードのサポート

Oracle Database 12cリリース2 (12.2.0.1)以降、DBMS_ROLLINGローリング・アップグレード時にData Guard Brokerは残ることができるようになり、無効化する必要はなくなりました。

Data Guard BrokerでのDBMS_ROLLINGアップグレードのサポートについては、次の点に注意してください。

- ブローカ・サポートは、ブローカが呼出し時に有効であればデフォルトでDBMS_ROLLING.BUILD_PLANプロシージャの実行時に有効です。ブローカ・サポートが有効になっている場合、ブローカは必要に応じて元のプライマリ・データベースとローリング・アップグレード・ターゲットからREDO転送先を設定します。
- DBMS_ROLLINGアップグレードを開始する前に、ファスト・スタート・フェイルオーバー機能を無効化する必要があります。
- ローリング・アップグレードの進行中にファスト・スタート・フェイルオーバーを有効化しようとすると、拒否されます。
- ローリング・アップグレードの進行中は、ロールの変更は現在のプライマリ・データベースを保護しているスタンバイ・データベースに対してのみ許可されます。ブローカは、SHOW CONFIGURATIONコマンドの実行時にローリング・アップグレード・ターゲットのロールをTransient Logical Standbyとしてレポートし、構成ステータスをROLLING DATABASE MAINTENANCE IS IN PROGRESSとしてレポートします。元のプライマリ・データベースとアップグレード・ターゲットの両方を保護しているスタンバイ・データベースがある場合、このトポロジは、SHOW CONFIGURATIONコマンドが現在のプライマリから発行されたときと同様にアップグレード・ターゲットから発行されたとき(プライマリ・データベースとして引継ぎが完了する前)に反映されます
- ブローカは、現在のプライマリを保護していないフィジカル・スタンバイへのロールの変更を回避します。これにより、スワ

チオーバー・フェーズの前は後続グループ・スタンバイへのロールの変更が許可され、スイッチオーバー・フェーズの後には先行グループ・スタンバイへのロールの変更のみが許可されます。

- アップグレード・プロセスの開始時に、アップグレード・ターゲットがOracle RACデータベースの場合、ブローカはターゲット・スタンバイを自動的に1つのインスタンスに削減して、アップグレードの続行を許可します。ブローカがない環境では、ターゲットで複数のインスタンスが実行されていることが検出された場合、アップグレードの開始は拒否されます。
- ブローカは、ローリング・アップグレードの過程で必要に応じてOracle ClusterwareおよびGlobal Data Servicesに通知します。
- ロールの推移は通常、ブローカを使用して実行されますが、ローリング・アップグレードのスイッチオーバー・ステップは引き続きDBMS_ROLLING SWITCHOVERプロシージャを使用して実行する必要があります。
- PL/SQLパッケージDBMS_ROLLINGを使用して実行されているローリング・アップグレードのステータスについての情報は、ブローカのコマンドSHOW CONFIGURATIONおよびSHOW DATABASEの出力に表示されます。

SHOW CONFIGURATIONコマンドは、アップグレード・ターゲットのロールとして一時ロジカル・スタンバイ・データベース、構成ステータスとしてROLLING DATABASE MAINTENANCE IN PROGRESSを示します。この出力の例は次のとおりです。

Configuration - DRSolution

```
Protection Mode: MaxPerformance
Members:
North_Sales - Primary database
South_Sales - Transient logical standby database
```

Fast-Start Failover: DISABLED

```
Configuration Status:
ROLLING DATABASE MAINTENANCE IN PROGRESS
```

SHOW DATABASEコマンドは、アップグレード・ターゲットおよび後続または先行スタンバイに対して、現在のローリング・アップグレードの進行に応じてWARNINGを該当するORAエラーとともに示します。この出力の例は次のとおりです。

Database - South_Sales

```
Role: Physical standby database
Intended State: APPLY-ON
Transport Lag: ***
Apply Lag: ***
Average Apply Rate: ***
Real Time Query: OFF
Instance(s):
South
Database Warning(s):
ORA-16866: database converted to transient logical standby database for rolling
database maintenance
```

```
Database Status:
WARNING
```

関連項目:

- [DBMS_ROLLING](#)
- [SHOW CONFIGURATION](#)
- [SHOW DATABASE](#)

14.2 DBMS_ROLLINGアップグレードとCDB

マルチテナント・コンテナ・データベース(CDB)でDBMS_ROLLINGを使用してローリング・アップグレードを実行するステップは、非CDB環境と変わりませんが、追加の要件と考慮事項があります。

CDBでのDBMS_ROLLINGアップグレードに固有の要件

DBMS_ROLLINGを使用してCDBでローリング・アップグレードを実行するための追加の要件は、次のとおりです。

- LOG_ARCHIVE_DEST_nパラメータで参照されるTNSサービスは、宛先データベースのルート・コンテナに解決されるサービスである必要があります。DBMS_ROLLINGを支援するプロセスでは、ルート・コンテナからのみ実行可能な数多くの操作が実行されます。
- DBMS_ROLLING.SWITCHOVERを呼び出す前に、一時ロジカル・スタンバイ上のすべてのコンテナ・データベースをプラグインして開く必要があります。これにより、特定のPDBに適用できないためにロジカル・スタンバイ適用エンジンが停止する可能性が排除されます。

DBMS_ROLLINGを使用したローリング・アップグレードの例は、[例14-5](#)を参照してください。

CDBでのDBMS_ROLLINGアップグレードの追加の考慮事項

DBMS_ROLLINGアップグレードの進行中は、アプリケーション・コンテナにインストールされたアプリケーションのインストール、アップグレード、パッチ適用はサポートされません。

- DBMS_ROLLINGアップグレードの進行中に、DDLを実行してアプリケーション・コンテナのインストール、アップグレード、パッチ適用を開始すると、エラーが返されます。(DBMS_ROLLINGアップグレードは続行されます。)
- アプリケーション・コンテナのアップグレードが進行中の場合、DBMS_ROLLINGアップグレードを開始しようとすると、エラーが発生します。(アプリケーション・コンテナのアップグレードは続行されます。)
- DBMS_ROLLINGアップグレードが実行され、データベースの互換性が12.2以上に設定されている場合、アプリケーション・コンテナのレプリケーションがサポートされます。DBMS_ROLLINGアップグレードを除き、ロジカル・スタンバイはアプリケーション・コンテナのサポートを提供しません。このようなコンテナはスキップされ、アプリケーション・コンテナがスキップされたことを示すメッセージがアラート・ログに書き込まれます。
- DBMS_ROLLINGを使用してアップグレードするCDBには、異なる文字セットのプラグブル・データベース(PDB)を含めることができます。

14.3 DBMS_ROLLINGの使用の概要

DBMS_ROLLING PL/SQLパッケージを使用したローリング・アップグレード・プロセスには、指定、コンパイル、実行という3つのステージがあります。

1. **指定:** 最初に、ローリング・アップグレード・プロセスの実装方法を指定します。将来のプライマリ・データベースを指定することは必須です。この動作は概念的には先行および後続グループの作成です。この時点では、先行グループのみにLGMが含まれます。オプションでLGMを保護する別のスタンバイを指定できます。

指定フェーズでは次のプロシージャを使用します。

- DBMS_ROLLING.INIT_PLAN
- DBMS_ROLLING.SET_PARAMETER

2. **コンパイル:** これはDBMS_ROLLING.BUILD_PLANプロシージャを呼び出すことで開始されます。BUILD_PLANプロシ-

ジャは、ローリング・アップグレードに関与するすべてのデータベースと通信し、ローリング・アップグレード計画を組み立てます。BUILD_PLANプロシージャも、既存のローリング・アップグレード計画を変更するために呼び出されます。変更は、DBMS_ROLLINGパラメータに対する変更の後、およびフェイルオーバーの結果としてのトポロジに対する変更の後が必要です。正常なローリング・アップグレードを確実にするために数多くの検証が必要なため、関与しているすべてのデータベースがBUILD_PLANプロシージャの実行中にアクセス可能である必要があります。

3. 実行: ローリング・アップグレードの実行には5つのステージがあります。

ステージ1: DBMS_ROLLING.START_PLANプロシージャはローリング・アップグレードの実行を開始します。これによりLGMデータベースがロジカル・スタンバイに変換され、LGMでSQL Applyプロセスが開始します。

ステージ2: 先行グループの一部のデータベースでデータベース・ソフトウェアをアップグレードします。LGMでアップグレード・スクリプトも実行します。この完了後、LGMデータベースでSQL Applyプロセスを再起動する必要があります。(アップグレード・スクリプトの詳細は、『[Oracle Databaseアップグレード・ガイド](#)』を参照してください。)より高いバージョンのバイナリを使用して再マウントすることにより、このステージ中に先行グループ・フィジカル・スタンバイも対応されます。これらのデータベースはLGMからREDOのリカバリによりアップグレードされます。

ステージ3: 適用ラグが指定したしきい値(デフォルトでは10分に設定されていますが、指定ステージで変更できます)に達すると、DBMS_ROLLING.SWITCHOVERプロシージャはスイッチオーバー処理を続行します。スイッチオーバーが完了すると、LGMはプライマリ・データベースになります。

ステージ4: LGMは新しいデータベース・ソフトウェアを実行するプライマリ・データベースとなり、先行グループ内のデータベースはそれを保護します。TGMはマウントされ、後続グループのデータベースはまだ古いバージョンのデータベース・ソフトウェアを実行しています。データベース・ソフトウェアのアップグレードと、より高いバージョンのバイナリへのデータベースの再マウントにより、アップグレード用にTGMおよびTGSデータベースを準備する必要があります。(アップグレード・スクリプトの詳細は、『[Oracle Databaseアップグレード・ガイド](#)』を参照してください。)

ステージ5: 現在のプライマリ・データベース(元のLGM)でDBMS_ROLLING.FINISH_PLANプロシージャを実行します。後続グループのすべてのデータベースを、現在のプライマリ・データベースのスタンバイになるように修復し、Applyプロセスを再起動します。FINISH_PLANプロシージャは、後続グループ内のすべてのデータベースが新しいリリースにアップグレードされるのを待機します(後続グループのデータベースのデータベース・ソフトウェアはステージ4で変更されましたが、後続グループのデータベースのデータ・ディクショナリは後続グループのロジカル・スタンバイを除き、LGMデータベースでアップグレード中に生成されたREDOのメディア・リカバリに基づいて更新されます)。

ローリング・アップグレードが正常に実行された後は、DBMS_ROLLING.DESTROY_PLANプロシージャを呼び出すことで、ローリング・アップグレードの指定を削除できます。

14.4 ローリング・アップグレードの計画

ローリング・アップグレードの計画は、正常にアップグレードするために不可欠です。計画フェーズでは、様々なアップグレード・パラメータを指定し、アップグレード計画を作成します。

パラメータとアップグレード計画は、環境に一意なすべての操作の詳細に焦点を当てます。アップグレード計画ではサイト特有の検証を実行し、ローリング・アップグレードを中断させる可能性のある構成およびリソースの問題を確認できます。

アップグレード・パラメータの定義およびアップグレード計画の作成に必要なタスクは、次のとおりです。

- アップグレード・パラメータを初期化します
- 現在のアップグレード・パラメータの値を表示します
- 必要に応じてアップグレード・パラメータの値を変更します
- アップグレード計画を作成します。

- 現在のアップグレード計画を表示します
- 必要に応じてアップグレード計画を修正します

これらの各ステップの詳細は、この項の残りの部分で説明します。これらは、表示されている順序で実行する必要があります。

1. 計画パラメータは、カスタマイズされる前に、システム生成のデフォルト値に初期化される必要があります。計画パラメータを初期化するには、DBMS_ROLLING.INIT_PLANプロシージャを呼び出します。このプロシージャは将来のプライマリ・データベースのDB_UNIQUE_NAME(先行グループのマスターまたはLGM)を識別します。LGMはSTART_PLANプロシージャ・コールの一部としてロジカル・スタンバイ・データベースに変換されます。次に、bostonが将来のプライマリ・データベースとして識別される、INIT_PLANプロシージャを呼び出すサンプルを示します。

```
DBMS_ROLLING.INIT_PLAN(future_primary=>'boston');
```

INIT_PLANプロシージャはシステム生成の計画パラメータの最初のセットを戻します。ここでは、ローリング・アップグレードの参加者としてDG_CONFIGのinit.oraパラメータに指定した各フィジカルおよびロジカル・スタンバイ・データベースを追加します。その他のデータベース(GoldenGateダウストリーム・デプロイメントをサービスするダウストリーム・データベースやスナプショット・スタンバイなど)は自動的に除外されます。

デフォルトでは、将来のプライマリ以外のスタンバイ・データベースはプライマリ・データベースを保護するように構成され、ローリング・アップグレードの必須参加者として構成されます。

データベース関連のパラメータを定義したら、INIT_PLANプロシージャでシステム提供のデフォルトを使用して操作パラメータを定義します。ほとんどの場合、計画パラメータは計画検証の準備ができていますが、必要性を満たしていることを確認するために、各パラメータを確認してください。

DESTROY_PLANプロシージャを呼び出してローリング・アップグレードに関連するすべての状態を削除するまで、計画パラメータはデータベース内に保持されます。

2. INIT_PLANプロシージャが完了したら、DBA_ROLLING_PARAMETERSビューを問い合わせる計画パラメータと現在の値を表示できます。計画パラメータはグローバルまたはローカルのいずれかの範囲です。グローバル・パラメータはローリング・アップグレード全体の属性で、データベースの参加者とは独立しています。グローバル・パラメータではSCOPE列にNULL値が含まれます。ローカル・パラメータではSCOPE列に関連する特定のデータベース名が含まれます。問合せの例を次に示します。

```
SQL> select scope, name, curval from dba_rolling_parameters order by scope, name;
```

SCOPE	NAME	CURVAL
seattle	INVOLVEMENT	FULL
seattle	MEMBER	NONE
boston	INVOLVEMENT	FULL
boston	MEMBER	TRAILING
oakland	INVOLVEMENT	FULL
oakland	MEMBER	TRAILING
atlanta	INVOLVEMENT	FULL
atlanta	MEMBER	LEADING
	ACTIVE_SESSIONS_TIMEOUT	3600
	ACTIVE_SESSIONS_WAIT	0
	BACKUP_CONTROLFILE	rolling_change_backup. f
	DGBROKER	0
	DICTIONARY_LOAD_TIMEOUT	3600
	DICTIONARY_LOAD_WAIT	0
	DICTIONARY_PLS_WAIT_INIT	300
	DICTIONARY_PLS_WAIT_TIMEOUT	3600
	EVENT_RECORDS	10000
	FAILOVER	0

GRP_PREFIX	DBMSRU_
IGNORE_BUILD_WARNINGS	0
IGNORE_LAST_ERROR	0
LAD_ENABLED_TIMEOUT	600
LOG_LEVEL	INFO
READY_LGM_LAG_TIME	600
READY_LGM_LAG_TIMEOUT	60
READY_LGM_LAG_WAIT	0
SWITCH_LGM_LAG_TIME	600
SWITCH_LGM_LAG_TIMEOUT	60
SWITCH_LGM_LAG_WAIT	1
SWITCH_LGS_LAG_TIME	60
SWITCH_LGS_LAG_TIMEOUT	60
SWITCH_LGS_LAG_WAIT	0
UPDATED_LGS_TIMEOUT	10800
UPDATED_LGS_WAIT	1
UPDATED_TGS_TIMEOUT	10800
UPDATED_TGS_WAIT	1

35 rows selected.

出力例で、データベースatlanta、boston、oaklandおよびseattleがすべてDG_CONFIGにより検出され、現在の計画のパラメータに割り当てられました。

関連項目:

- DBA_ROLLING_PARAMETERSビューの詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

3. 既存のローリング・アップグレード・パラメータを変更するには、DBMS_ROLLING.SET_PARAMETER PL/SQLプロシージャを使用します。次に、SET_PARAMETERプロシージャの使用例を示します。

```
DBMS_ROLLING.SET_PARAMETER (
  scope IN VARCHAR2,
  name IN VARCHAR2,
  value IN VARCHAR2);
```

SCOPEにはローカル・パラメータ用のDB_UNIQUE_NAME値、またはグローバル・パラメータ用のNULLのいずれかを指定します。データベース特有ではないパラメータにはNULLの範囲を指定する必要はありません。

NAMEは変更するパラメータの名前です。

VALUEにはそのパラメータの値を指定します。NULLの値はパラメータをシステム提供のデフォルト(存在する場合)に戻します。

関連項目:

- すべての使用可能なローリング・アップグレード・パラメータの完全なリストは、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)を参照してください。

次の例では、いくつかのローリング・アップグレード・パラメータの使用例を示しています。

4. すべての必要なパラメータが指定されたら、アップグレード計画を作成します。アップグレード計画は、Oracle Data Guard構成をローリング・アップグレードに沿ってガイドする、カスタム生成の手順のセットです。(生成される計画は、

Data Guardブローカを使用して構成を管理するかどうかによって異なります。)

アップグレード計画を作成するには、DBA_ROLLING_BUILD_PLAN PL/SQLプロシージャを使用します。このプロシージャには、計画パラメータで説明した通りの構成と、すべてのインスタンスが開始され、ネットワーク経由でアクセス可能であることが必要です。

プロシージャは次のように呼び出されます。

```
DBMS_ROLLING.BUILD_PLAN;
```

プロシージャはすべての入力をDBA_ROLLING_PARAMETERSビューから取得するため、指定する引数はありません。プロシージャは計画パラメータを検証し、ログ転送やフラッシュ・リカバリ領域の設定など、サイト特有のリソースの検証を実行します。通常、ベスト・プラクティスの値の条件を満たさない構成設定は警告として扱われ、DBA_ROLLING_EVENTSビューに記録されます。デフォルトで、IGNORE_BUILD_WARNINGSパラメータは1に設定され、警告によって、アップグレード計画が使用可能な状態に達するのを妨げられないことを意味します。計画の策定時に、より厳しいルールの実施が必要な場合には、このパラメータを0に設定できます。

ノート:



計画の生成時に実行される検証はローリング・アップグレードに特有です。これはアップグレードの準備状況を評価するアップグレード前情報ツールを実行する、推奨プラクティスのかわりではありません。

計画の生成後、次のステップに移動してそれを表示し、問題がないかどうか診断して、必要ならば修正します。

5. BUILD_PLANプロシージャが正常に戻った後、完全なアップグレード計画がDBA_ROLLING_PLANビューで表示可能です。ビューの各レコードは、実行がスケジュールされた特定の手順を識別します。

次の出力は、ローリング・アップグレード計画の表示例です。

```
SQL> SELECT instid, target, phase, description FROM DBA_ROLLING_PLAN;
```

INSTID	TARGET	PHASE	DESCRIPTION
1	seattle	START	Verify database is a primary
2	seattle	START	Verify MAXIMUM PROTECTION is disabled
3	boston	START	Verify database is a physical standby
4	boston	START	Verify physical standby is mounted
5	oakland	START	Verify database is a physical standby
6	oakland	START	Verify physical standby is mounted
7	atlanta	START	Verify database is a physical standby
8	atlanta	START	Verify physical standby is mounted
9	seattle	START	Verify server parameter file exists and is modifiable
10	boston	START	Verify server parameter file exists and is modifiable
11	oakland	START	Verify server parameter file exists and is modifiable
12	atlanta	START	Verify server parameter file exists and is modifiable
13	seattle	START	Verify Data Guard Broker configuration is disabled
14	boston	START	Verify Data Guard Broker configuration is disabled
15	oakland	START	Verify Data Guard Broker configuration is disabled
16	atlanta	START	Verify Data Guard Broker configuration is disabled
17	seattle	START	Verify flashback database is enabled
18	seattle	START	Verify available flashback restore points
19	boston	START	Verify flashback database is enabled
20	boston	START	Verify available flashback restore points
21	oakland	START	Verify flashback database is enabled

22	oakland	START	Verify available flashback restore points
23	atlanta	START	Verify flashback database is enabled
24	atlanta	START	Verify available flashback restore points
25	boston	START	Scan LADs for presence of atlanta destination
26	boston	START	Test if atlanta is reachable using configured TNS service
27	boston	START	Stop media recovery
28	oakland	START	Stop media recovery
29	atlanta	START	Stop media recovery
30	boston	START	Drop guaranteed restore point DBMSRU_INITIAL
31	boston	START	Create guaranteed restore point DBMSRU_INITIAL
32	oakland	START	Drop guaranteed restore point DBMSRU_INITIAL
33	oakland	START	Create guaranteed restore point DBMSRU_INITIAL
34	atlanta	START	Drop guaranteed restore point DBMSRU_INITIAL
35	atlanta	START	Create guaranteed restore point DBMSRU_INITIAL
36	seattle	START	Drop guaranteed restore point DBMSRU_INITIAL
37	seattle	START	Create guaranteed restore point DBMSRU_INITIAL
INSTID	TARGET	PHASE	DESCRIPTION
38	boston	START	Start media recovery
39	boston	START	Verify media recovery is running
40	oakland	START	Start media recovery
41	oakland	START	Verify media recovery is running
42	atlanta	START	Start media recovery
43	atlanta	START	Verify media recovery is running
44	seattle	START	Verify user_dump_dest has been specified
45	seattle	START	Backup control file to rolling_change_backup.f
46	boston	START	Verify user_dump_dest has been specified
47	boston	START	Backup control file to rolling_change_backup.f
48	oakland	START	Verify user_dump_dest has been specified
49	oakland	START	Backup control file to rolling_change_backup.f
50	atlanta	START	Verify user_dump_dest has been specified
51	atlanta	START	Backup control file to rolling_change_backup.f
52	seattle	START	Get current redo branch of the primary database
53	boston	START	Wait until recovery is active on the primary's redo branch
54	boston	START	Stop media recovery
55	seattle	START	Execute dbms_logstdby.build
56	boston	START	Convert into a transient logical standby
57	boston	START	Open database
58	boston	START	Configure logical standby parameters
59	boston	START	Start logical standby apply
60	boston	START	Get redo branch of transient logical standby
61	boston	START	Get reset scn of transient logical redo branch
62	atlanta	START	Stop media recovery
63	atlanta	START	Flashback database
64	seattle	START	Disable log file archival to atlanta
65	boston	START	Enable log file archival to atlanta
66	boston	START	Wait for log archive destination to atlanta to reach a valid state
67	atlanta	START	Wait until transient logical redo branch has been registered
68	atlanta	START	Start media recovery
69	atlanta	START	Wait until v\$logstandby_stats has been initialized
70	atlanta	START	Wait until recovery has started on the transient redo branch
71	seattle	START	Log pre-switchover instructions to events table
72	boston	START	Record start of user upgrade of boston
73	boston	SWITCH	Verify database is in OPENRW mode
74	boston	SWITCH	Record completion of user upgrade of boston
INSTID	TARGET	PHASE	DESCRIPTION
75	boston	SWITCH	Scan LADs for presence of seattle destination

76	boston	SWITCH	Scan LADs for presence of oakland destination
77	boston	SWITCH	Scan LADs for presence of atlanta destination
78	boston	SWITCH	Test if seattle is reachable using configured TNS service
79	boston	SWITCH	Test if oakland is reachable using configured TNS service
80	boston	SWITCH	Test if atlanta is reachable using configured TNS service
81	seattle	SWITCH	Enable log file archival to boston
82	boston	SWITCH	Enable log file archival to atlanta
83	boston	SWITCH	Start logical standby apply
84	atlanta	SWITCH	Start media recovery
85	atlanta	SWITCH	Wait until upgrade redo has been fully recovered
86	boston	SWITCH	Wait until apply lag has fallen below 600 seconds
87	seattle	SWITCH	Log post-switchover instructions to events table
88	seattle	SWITCH	Switch database to a logical standby
89	boston	SWITCH	Wait until end-of-redo has been applied
90	oakland	SWITCH	Wait until end-of-redo has been applied
91	seattle	SWITCH	Disable log file archival to oakland
92	boston	SWITCH	Switch database to a primary
93	oakland	SWITCH	Stop media recovery
94	seattle	SWITCH	Synchronize plan with new primary
95	seattle	FINISH	Verify only a single instance is active
96	seattle	FINISH	Verify database is mounted
97	seattle	FINISH	Flashback database
98	seattle	FINISH	Convert into a physical standby
99	oakland	FINISH	Verify database is mounted
100	oakland	FINISH	Flashback database
101	boston	FINISH	Verify database is open
102	boston	FINISH	Save the DBID of the new primary
103	boston	FINISH	Save the logminer session start scn
104	seattle	FINISH	Wait until transient logical redo branch has been registered
105	oakland	FINISH	Wait until transient logical redo branch has been registered
106	seattle	FINISH	Start media recovery
107	oakland	FINISH	Start media recovery
108	seattle	FINISH	Wait until apply/recovery has started on the transient branch
109	oakland	FINISH	Wait until apply/recovery has started on the transient branch
110	seattle	FINISH	Wait until upgrade redo has been fully recovered
INSTID	TARGET	PHASE	DESCRIPTION
111	oakland	FINISH	Wait until upgrade redo has been fully recovered
112	seattle	FINISH	Drop guaranteed restore point DBMSRU_INITIAL
113	boston	FINISH	Drop guaranteed restore point DBMSRU_INITIAL
114	oakland	FINISH	Drop guaranteed restore point DBMSRU_INITIAL
115	atlanta	FINISH	Drop guaranteed restore point DBMSRU_INITIAL

115 rows selected.

SQL>

このビューの列には次の情報が表示されます。

- INSTID - 手順IDで、実行する手順の順序です。手順は通常、グループ内で実行されます。
- PHASE - アップグレード計画の各手順は特定のフェーズと関連付けられます。フェーズとはDBMS_ROLLING PL/SQLパッケージ内のプロシージャにより実行される手順の論理グループです。DBMS_ROLLINGプロシージャが呼び出されると、そのフェーズのアップグレード計画内の関連するすべての手順が実行されます。可能なフェーズは次のとおりです。
 - START: リストア・ポイントの取得、一時ロジカル・スタンバイ・データベースのインスタンス化、LGSデータベースの構成などの、設定に関連するアクティビティから構成されます。このフェーズのアクティビティは

DBMS_ROLLING.SET_PLANプロシーヂャを呼び出すと開始します。「[ローリング・アップグレードの実行](#)」のステップ1を参照してください。

- SWITCH: 一時ロジカル・スタンバイの新しいプライマリ・データベースへのスイッチオーバーに関連するアクティビティから構成されます。このフェーズのアクティビティはDBMS_ROLLING.SWITCHOVERプロシーヂャを呼び出すと開始します。「[ローリング・アップグレードの実行](#)」のステップ3を参照してください。
- FINISH: アップグレードREDOのリカバリ用のスタンバイ・データベースの構成に関連するアクティビティから構成されます。このフェーズのアクティビティはDBMS_ROLLING.FINISH_PLANプロシーヂャを呼び出すと開始します。「[ローリング・アップグレードの実行](#)」のステップ5を参照してください。
- EXEC_STATUS - 手順の全体的なステータスです。
- PROGRESS - 手順の実行の進捗です。REQUESTINGの値は、実行のためにターゲット・データベースに転送されている手順を示します。EXECUTINGの値は、アクティブに実行中の手順を示します。REPLYINGの値は、完了情報が戻されていることを示します。
- DESCRIPTION - 実行をスケジュールされた特定の操作です。
- TARGET - 指定の手順が実行されるサイトです。
- EXEC_INFO - 手順に関連する追加のコンテキスト情報です。

関連項目:

- DBA_ROLLING_PLANビューの詳細は、『[Oracle Databaseリファレンス](#)』を参照してください。

6. ローリング・アップグレードまたはデータベース構成に変更があった後は、アップグレード計画は修正される必要があります。構成変更には次のものがあります。

- 任意のデータベースのローリング・アップグレードへの参加時のinit.oraパラメータ・ファイルの変更
- フェイルオーバー・イベントの結果としてのデータベース・ロールの変更
- ローリング・アップグレード・パラメータの変更

アクティブなアップグレード計画を修正するには、BUILD_PLANプロシーヂャを再度呼び出します。場合によっては、BUILD_PLANプロシーヂャで、特定の変更が受け入れられない場合に、エラーが発生する可能性があります。たとえば、ACTIVE_SESSIONS_WAITパラメータの設定は、スイッチオーバーがすでに発生していると、何も影響しません。

パラメータを個別に処理するのではなく、BUILD_PLANプロシーヂャを呼び出して、パラメータのグループの変更を処理することをお勧めします。

例14-1 適用ラグ要件を強制するスイッチオーバーの設定

次の例で、将来のプライマリにスイッチオーバーする前に、適用ラグが60秒を下回るのを待つように計画を構成する方法を示します。

```
DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_WAIT', '1');
DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_TIME', '60');
```

例14-2 ログのデフォルト値へのリセット

次の例では、LOG_LEVELグローバル・パラメータのデフォルト値へのリセットを示します。

```
DBMS_ROLLING.SET_PARAMETER (
  name=>'LOG_LEVEL',
```



```
value=>NULL);
```

例14-3 データベースのオプション参加者としての指定

次の例では、データベースで発生したエラーがローリング・アップグレード全体を妨げてはいけないことを示す、データベース atlanta の INVOLVEMENT ローカル・パラメータの設定を示します。

```
DBMS_ROLLING.SET_PARAMETER (
  scope=>'atlanta',
  name=>'involvement',
  value=>'optional');
```

例14-4 一時ロジカル・スタンバイを保護するデータベースの設定

次の例では、ローリング・アップグレード中にデータベースにより一時ロジカル・スタンバイを保護する必要があることを示す、データベース atlanta の MEMBER ローカル・パラメータの設定を示します。

```
DBMS_ROLLING.SET_PARAMETER (
  scope=>'atlanta',
  name=>'member',
  value=>'leading');
```

14.5 ローリング・アップグレードの実行

DBMS_ROLLING PL/SQL パッケージを使用してローリング・アップグレードを実行するには、次のステップを実行します。

[表14-2](#)に、このステップの要約を示します。これらのステップは、「[ローリング・アップグレードの計画](#)」での説明に従い、最初にアップグレード計画を正常に作成したことを前提としています。

表14-2 DBMS_ROLLINGを使用したローリング・アップグレードの実行ステップ

ステップ	説明	PHASE
ステップ 1	DBMS_ROLLING.START_PLAN プロシージャを呼び出し、将来のプライマリと、将来のプライマリを保護するように指定したフィジカル・スタンバイを構成します。	START
ステップ 2	将来のプライマリ・データベースとそれを保護するスタンバイで Oracle Database ソフトウェアを手動でアップグレードします。	SWITCH PENDING
ステップ 3	DBMS_ROLLING.SWITCHOVER プロシージャを呼び出し、現在と将来のプライマリ・データベース間でロールを切り替えます。	SWITCH
ステップ 4	以前のプライマリと残りのスタンバイ・データベースをより高いバージョンの Oracle Database で手動で再起動します。	FINISH PENDING
ステップ 5	DBMS_ROLLING.FINISH_PLAN プロシージャを呼び出し、以前のプライマリをフィジカル・スタンバイへ変換し、残りのスタンバイ・データベースをアップグレード REDO のリカバリ用に構成します。	FINISH

[表14-2](#)のPHASE列に示された、ローリング・アップグレードの特定のフェーズに属する各ステップの間に実行するアクティビティ。ローリング・アップグレード操作は任意の時点で1つのフェーズです。ローリング・アップグレードの現在のフェーズは、

DBA_ROLLING_STATUSビューのPHASE列にレポートされます。

各アップグレード・ステップの詳細は、この項の残りの部分で説明します。

1. DBMS_ROLLING.START_PLANプロシーチャを呼び出し、将来のプライマリと、将来のプライマリを保護するように指定したフィジカル・スタンバイを構成します。

DBMS_ROLLING.START_PLANプロシーチャはローリング・アップグレードの正式な開始です。START_PLANプロシーチャの目的は、一時ロジカル・スタンバイ・データベースとそれを保護するように指定されたフィジカル・スタンバイ・データベースを構成することです。呼び出されると、START_PLANプロシーチャはSTART_PLANのPHASE値を含むアップグレード計画内のすべての手順を実行します。実行される手順のタイプには次のものがあります。

- 各データベースの制御ファイルのトレース・ファイルへのバックアップ
- フラッシュバック・データベースの保証付きリストア・ポイントの作成
- プライマリ・データベースでのLogMinerディクショナリの作成
- 指定したフィジカル・スタンバイの一時ロジカル・スタンバイ・データベースへのリカバリ
- LogMinerディクショナリのロジカル・スタンバイ・データベースへのロード
- 一時ロジカル・スタンバイ・データベースを使用したLGSデータベースの構成

次のようにSTART_PLANプロシーチャを呼び出します(引数は必要ありません)。

```
SQL> EXECUTE DBMS_ROLLING.START_PLAN;
```

2. 将来のプライマリ・データベースとそれを保護するスタンバイでOracle Databaseソフトウェアを手動でアップグレードします。

START_PLANプロシーチャが完了した後、将来のプライマリ・データベースとそれを保護するスタンバイでOracle Databaseソフトウェアを手動でアップグレードする必要があります。これを行うには次のステップが必要になります。

- 一時ロジカル(LGM)および先行グループ・スタンバイ(LGS)のOracle Databaseソフトウェアをアップグレードします。
- LGSデータベースでメディア・リカバリを開始します。
- 手動またはOracle Database Upgrade Assistant(DBUA)を使用して、一時ロジカル・スタンバイ・データベースをアップグレードします。
- 一時ロジカル・スタンバイを読み取り/書き込みモードで再度開きます。

一時ロジカル・スタンバイおよびLGSデータベースは機能グループです。LGSデータベースは一時ロジカル・スタンバイがアップグレードされる前に、より高いバージョンのアクティブに実行中のメディア・リカバリで再起動する必要があります。LGSデータベースが最初に構成されない場合は、一時ロジカルのアップグレードは保護されません。このステップの最後に、一時ロジカルのアップグレードは完了し、メディア・リカバリがすべてのLGSデータベースで実行しています。

スイッチオーバーを実行する前に、すべてのLGSデータベースが完全にアップグレードされるまで待機することをお勧めします。DBA_ROLLING_DATABASESビューで関連レコードがUPDATED列にYESの値を報告している場合、LGSデータベースは完全にアップグレードされています。

3. DBMS_ROLLING.SWITCHOVERプロシーチャを呼び出し、現在と将来のプライマリ・データベース間でロールを切り替えます。

SWITCHOVERプロシーチャは現在と将来のプライマリ・データベース間でロールを切り替えます。このプロシーチャは、プライマリ・サービスの提示時間を最小化する、適用ラグが最小のときにスイッチオーバーが発生するように調整します。SWITCHOVERプロシーチャはSWITCHOVERのPHASE値を含むアップグレード計画内のすべての手順を実行します。実行さ

れる手順のタイプには次のものがあります。

- 現在の一時ロジカル・スタンバイである先行グループ・マスター(LGM)で適用ラグがしきい値を下回るまで待機する
- LGSデータベースで適用ラグがしきい値を下回るまで待機する
- プライマリからロジカル・スタンバイ・ロールへ切り替える
- 現在のロジカル・スタンバイである先行グループ・マスター(LGM)からプライマリ・ロールへ切り替える
- 新しいプライマリになった後で先行グループ・マスター(LGM)でログ・アーカイブ宛先を有効化する

次のようにSWITCHOVERプロシージャを呼び出します(引数は必要ありません)。

```
SQL> EXECUTE DBMS_ROLLING.SWITCHOVER;
```

プライマリからスタンバイ・ロールへのスイッチオーバー後にスイッチオーバー・エラーが発生したものの、一時ロジカルがプライマリ・ロールに正常に変換される前の場合、正常に完了するまで、引き続き以前のプライマリ・サイトでSWITCHOVERプロシージャを実行します。

4. この時点で、以前のプライマリと残りのスタンバイ・データベースをより高いバージョンのOracle Databaseで手動で再起動し、マウントする必要があります。ローリング・アップグレードを継続するために、DBMS_ROLLINGパッケージはスタンバイ・データベースと通信する必要があるため、スタンバイ・データベースのマウントは特に重要です。
5. FINISH_PLANプロシージャの全体的な目的は、アップグレードREDO全体をリカバリすることと、以前のプライマリおよびTGPスタンバイをフィジカル・スタンバイとして構成することです。呼び出されると、FINISH_PLANプロシージャはFINISHのPHASE値を含むアップグレード計画内のすべての手順を実行します。実行される手順のタイプには次のものがあります。
 - 以前のプライマリおよびTGPスタンバイのフラッシュバック
 - 以前のプライマリのフィジカル・スタンバイへの変換
 - 新しいREDOブランチでのメディア・リカバリの開始

次のようにFINISH_PLANプロシージャを呼び出します(引数は必要ありません)。

```
SQL> EXECUTE DBMS_ROLLING.FINISH_PLAN;
```

14.6 ローリング・アップグレードの監視

ローリング・アップグレードに関与するデータベースに関する情報を問い合わせることができるビューがいくつかあります。

- DBA_ROLLING_STATUS
アップグレードの全体的なステータスに関する情報を提供します。
- DBA_ROLLING_DATABASES
ローリング・アップグレードに関与する各データベースのロール、保護、およびリカバリ状態に関する情報を提供します。
- DBA_ROLLING_STATISTICS
開始および終了時間、サービスがオフラインだった時間などの統計情報を提供します。

関連項目:

- これらのビューの詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

14.7 ローリング・アップグレードのロールバック

ローリング・アップグレード手順をロールバックするには、DBMS_ROLLING. ROLLBACK_PLANプロシージャを呼び出すことができます。プロシージャは次のように呼び出されます。

```
DBMS_ROLLING. ROLLBACK_PLAN;
```

ROLLBACK_PLANプロシージャには次の要件があります。

- ROLLBACK_PLANプロシージャはDBMS_ROLLING. SWITCHOVERプロシージャがこれまで呼び出されたことがない場合にのみ呼び出すことができます。
- ROLLBACK_PLANプロシージャを使用する前に、フラッシュバック・データベースが差し迫っているので、一時ロジカル・スタンバイ・データベースをマウント状態に戻す必要があります。
- Oracle Databaseソフトウェアがすでにアップグレードされている場合、古いバージョンで結果のフィジカル・スタンバイを再起動し、メディア・リカバリを開始する必要があります。

14.8 ローリング・アップグレード中に発生したロール変更の処理

ローリング・アップグレードが進行中で、ロールオーバーが完了する前にOracle Data Guard構成でフェイルオーバーを実行する必要がある状況が発生した場合は、次の状況でのみフェイルオーバーを実行できます。

- DBMS_ROLLINGプロシージャの進行中にフェイルオーバーが実行されなかった。
- フェイルオーバーがプライマリ・データベースとフィジカル・スタンバイ・データベース間で発生し、これがデータ消失なしのフェイルオーバーだった。
- フェイルオーバーが一時ロジカル・スタンバイ・データベースと一時ロジカル・スタンバイ・データベースのフィジカル・スタンバイ間で発生した。

ロール変更は、様々な構成に応じたアップグレード計画内の手順を必ず無効化する重要なイベントです。ローリング・アップグレードを再開するには、新しい計画を作成する必要があります。FAILOVERパラメータを設定して、構成が変更されたことを示す必要があります。このパラメータはBUILD_PLANプロシージャの次の起動時に検出され、それに応じて既存の計画が修正されます。

修正された計画が作成されると、ローリング・アップグレードを再開できます。

14.9 ローリング・アップグレードの例

次の例は、様々なローリング・アップグレードのシナリオを示しています。

すべてのシナリオのいくつかの地点で、同じ基本的なローリング・アップグレードのステップが使用されています。これらのステップを[例14-5](#)に示します。残りの例は、同じステップを繰り返すのではなく、適切なこの例に戻って参照します。

この項のいくつかの例では、ローリング・アップグレードの再開が指示されます。つまり、停止したところから続行する必要があります。ローリング・アップグレードの再開には、ローリング・アップグレードの現在のフェーズの識別と、そのフェーズに関連するPL/SQLプロシージャまたはフェーズに関連するアクティビティのいずれかの再実行が含まれます。ローリング・アップグレードの現在のフェーズは、DBA_ROLLING_STATUSビューのPHASE列に示されます。



ノート:

この項であげたシナリオは、仮想上の例を示すためのものです。Oracle Active Data Guard を使用したローリン

グ・アップグレード機能を使用して、Oracle Database 12c に対する最初のパッチセット以降のデータベース・アップグレードを実行できます。

例14-5 基本ローリング・アップグレード・ステップ

1. ローリング・アップグレードを開始します。

```
SQL> EXECUTE DBMS_ROLLING.START_PLAN;
```

2. 一時ロジカル・スタンバイとその保護しているスタンバイをアップグレードします。

- a. Oracle Databaseソフトウェアの上位バージョンを使用してLGPスタンバイをマウントします。

- b. 先行グループ・フィジカル(LGP)でメディア・リカバリを開始します。

- c. Oracle Databaseソフトウェアの上位バージョンを使用してアップグレード・モードで、一時ロジカル・スタンバイである先行グループ・マスター(LGM)を開きます。

- d. 手動またはOracle Database Upgrade Assistant(DBUA)を使用して、一時ロジカル・スタンバイである先行グループ・マスター(LGM)をアップグレードします。

- e. 一時ロジカル・スタンバイである先行グループ・マスター(LGM)を読み取り/書き込みモードで再起動します。

3. 先行グループ・マスター(LGM)にスイッチオーバーします。

```
SQL> EXECUTE DBMS_ROLLING.SWITCHOVER;
```

4. 後続グループのデータベースを再起動します。これには、元のプライマリ・データベースと、後続グループ(TGP)内の保護しているすべてのスタンバイが含まれます。

- a. Oracle Databaseの上位バージョンを使用して前のプライマリをマウントします。

- b. Oracle Databaseの上位バージョンを使用して前のプライマリのフィジカル・スタンバイをマウントします。

5. ローリング・アップグレードを終了します。

```
SQL> EXECUTE DBMS_ROLLING.FINISH_PLAN;
```

例14-6 2つのデータベース間のローリング・アップグレード

次の例では、プライマリ・データベースおよびフィジカル・スタンバイ・データベースで構成されている2サイト構成でのローリング・アップグレードについて説明します。この例では、seattleが現在のプライマリで、bostonが将来のプライマリです。seattleデータベースが後続グループ・マスター(TGM)として自動的に選択され、操作に参加します。デフォルトで、seattleに対しては何も設定する必要はありません。

1. アップグレード計画を初期化します。

```
SQL> EXECUTE DBMS_ROLLING.INIT_PLAN(future_primary=>' boston');
```

2. アップグレード計画を作成します。

```
SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;
```

3. [例14-5](#)の説明のように、ローリング・アップグレードを実行します。

例14-7 3つのデータベース間のローリング・アップグレード

次の例では、プライマリ・データベースおよび2つのフィジカル・スタンバイ・データベースで構成されている3サイト構成でのローリング・アップグレードについて説明します。この例では、seattleが現在のプライマリ、bostonが将来のプライマリで、oaklandが

seattleのフィジカル・スタンバイです。

1. アップグレード計画を初期化します。

```
SQL> EXECUTE DBMS_ROLLING.INIT_PLAN (future_primary => 'boston');
```

2. アップグレード計画を作成します。

```
SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;
```

3. [例14-5](#)の説明のように、ローリング・アップグレードを実行します。

例14-8 4つのデータベース間のローリング・アップグレード

次の例では、プライマリ・データベースおよび3つのフィジカル・スタンバイ・データベースで構成されている4サイト構成でのローリング・アップグレードについて説明します。この例では、seattleがプライマリ・データベース、bostonが将来のプライマリ、oaklandがseattleのフィジカル・スタンバイで、atlantaがbostonのフィジカル・スタンバイです。

1. アップグレード計画を初期化します。

```
SQL> EXECUTE DBMS_ROLLING.INIT_PLAN (future_primary => 'boston');
```

2. atlantaを先行グループのスタンバイとして構成します。

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(scope=>'atlanta', name=>'member',  
value=>'leading');
```

3. アップグレード計画を作成します。

```
SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;
```

4. [例14-5](#)の説明のように、ローリング・アップグレードを実行します。

例14-9 リーダー・ファームでのローリング・アップグレード

次の例では、1つのプライマリ・データベースおよび9つのフィジカル・スタンバイ・データベースで構成されているリーダー・ファーム構成でのローリング・アップグレードについて説明します。この例では、スイッチオーバーの前後でフィジカル・スタンバイがOracle Active Data Guardスタンバイとして使用可能になるために、8つのフィジカル・スタンバイ・データベースが4つずつの2つのグループに分けられます。この例では、seattleがプライマリ、bostonが将来のプライマリ、データベースrf[a-d]がseattleのフィジカル・スタンバイで、データベースrf[e-h]がbostonのフィジカル・スタンバイです。将来のプライマリ・データベースのリーダー・ファーム・グループ間の適用ラグが60秒よりも短くなるまで、新しいプライマリへのスイッチオーバーが待機するように、ローリング・アップグレードが構成されます。

1. アップグレード計画を初期化します。

```
SQL> EXECUTE DBMS_ROLLING.INIT_PLAN ( future_primary => 'boston');
```

2. 将来のプライマリを保護するようにリーダー・ファーム・グループを構成します。

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(scope=>'rfe', name=>'member',  
value=>'leading');
```

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(scope=>'rff', name=>'member',  
value=>'leading');
```

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(scope=>'rfg', name=>'member',  
value=>'leading');
```

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(scope=>'rfh', name=>'member',
```



```
value=>' leading');
```

3. 将来のプライマリのリーダー・ファームに60秒の最大許容適用ラグを設定します。

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER (name=>' SWITCH_LGS_LAG_WAIT',  
value=>' 1');
```

4. アップグレード計画を作成します。

```
SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;
```

5. [例14-5](#)の説明のように、ローリング・アップグレードを実行します。

例14-10 Application Testingのローリング・アップグレード

次の例では、上位バージョンのデータベースのアプリケーションを検証するための、一時ロジカル・スタンバイおよび一時ロジカル・スタンバイのフィジカルを構成する、4サイト構成でのローリング・アップグレードの使用について説明します。プライマリ・データベースはseattleで、bostonは将来のプライマリ、oaklandはseattleのフィジカル・スタンバイで、atlantaはbostonのフィジカル・スタンバイです。そのため、この例では、seattleおよびoaklandが後続グループを構成し、bostonおよびatlantaが先行グループを構成します。テストの終わりに、seattleの保護を再開するために、bostonおよびatlantaが元のフィジカル・スタンバイにリストアされます。

1. アップグレード計画を初期化します。

```
SQL> EXECUTE DBMS_ROLLING.INIT_PLAN (future_primary => ' boston');
```

2. 将来のプライマリを保護するようにatlantaを構成します。

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER (scope=>' atlanta', name=>' member',  
value=>' leading');
```

3. アップグレード計画を作成します。

```
SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;
```

4. ローリング・アップグレードを開始します。

```
SQL> EXECUTE DBMS_ROLLING.START_PLAN;
```

5. bostonおよびatlantaをアップグレードします。

- a. 上位バージョンのデータベースを使用してatlantaをマウントします。
- b. atlantaでメディア・リカバリを開始します。
- c. 上位バージョンのデータベースを使用してアップグレード・モードでbostonを開きます。
- d. 手動またはOracle Database Upgrade Assistant(DBUA)を使用して、データベースbostonをアップグレードします。
- e. bostonを読取り/書き込みモードで再起動します。

6. 必要に応じてアプリケーションをテストします。

7. 構成をロールバックします。

- a. bostonをマウント・モードで再起動します。
- b. アップグレードをロールバックします。

```
SQL> EXECUTE DBMS_ROLLING.ROLLBACK_PLAN;
```

8. 下位バージョンのデータベースを使用してbostonおよびatlantaでメディア・リカバリを開始します。

a. 下位バージョンのデータベースを使用して bostonおよびatlantaをマウントします。

b. bostonおよびatlantaでメディア・リカバリを開始します。

例14-11 新しいプライマリへのフェイルオーバー後のローリング・アップグレードの再開

次の例では、フィジカル・スタンバイのプライマリ・ロールへのデータ消失なしのフェイルオーバーと、その後の3サイト構成でのローリング・アップグレード計画の再構成について説明します。この例では、seattleが現在のプライマリ、bostonが将来のプライマリで、oaklandがseattleのフィジカル・スタンバイです。データベースoaklandはフェイルオーバーされて新しいプライマリになります。(後続グループは(seattle, oakland)で先行グループはbostonです。)

1. oaklandの残りのREDOがリカバリされ、新しいプライマリ・ロールにフェイルオーバーされます。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;  
SQL> ALTER DATABASE RECOVER MANAGED STANDBY FINISH;  
SQL> ALTER DATABASE FAILOVER TO OAKLAND;  
SQL> ALTER DATABASE OPEN;
```

2. 必要に応じて、oaklandのログ・アーカイブ宛先を構成します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='service="boston" reopen=5  
2 LGWR ASYNC NET_TIMEOUT=180 valid_for=(ONLINE_LOGFILE, PRIMARY_ROLE)  
3 DB_UNIQUE_NAME="oakland" ;
```

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

3. フェイルオーバーが実行されたことを示すパラメータを設定します。:

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(name=>'failover', value=>'1');
```

4. アップグレード計画を修正します。

```
SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;
```

5. ローリング・アップグレードを再開します。

例14-12 新しい一時ロジカルへのフェイルオーバー後のローリング・アップグレードの再開

次の例では、フィジカル・スタンバイの一時ロジカル・ロールへのフェイルオーバーと、その後の5サイト構成でのローリング・アップグレード計画の再構成について説明します。この例では、seattleがプライマリ、bostonが将来のプライマリ、oaklandがseattleのフィジカル・スタンバイで、atlantaとmiamiがbostonのフィジカル・スタンバイです。データベースatlantaはフェイルオーバーされて新しい一時ロジカル・スタンバイになります。

1. atlantaの残りのREDOがリカバリされ、新しい一時ロジカル・ロールにフェイルオーバーされます。

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

2. 必要に応じて、atlantaのログ・アーカイブ宛先を構成します。

```
SQL> alter system set log_archive_dest_2='service="seattle" reopen=5  
2 LGWR ASYNC NET_TIMEOUT=180 valid_for=(ONLINE_LOGFILE, PRIMARY_ROLE)  
3 DB_UNIQUE_NAME="atlanta" ;
```

```
SQL> alter system set log_archive_dest_3='service="oakland" reopen=5
  2 LGWR ASYNC NET_TIMEOUT=180 valid_for=(ONLINE_LOGFILE, PRIMARY_ROLE)
  3 DB_UNIQUE_NAME="atlanta"' ;

SQL> alter system set log_archive_dest_4='service="miami" reopen=5
  2 LGWR ASYNC NET_TIMEOUT=180 valid_for=(ONLINE_LOGFILE, ALL_ROLES)
  3 DB_UNIQUE_NAME="atlanta"' ;
```

3. 新しい一時ロジカル・スタンバイ・データベースとしてatlantaを指定します。

```
SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(name=>'failover', value=>'1');
```

4. アップグレード計画を修正します。

```
SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;
```

5. ロールング・アップグレードを再開します。

15 Oracle Data Guardの使用例

これらのシナリオは、Oracle Data Guard構成の管理中に遭遇する可能性がある様々な状況を示します。それぞれ、ユーザー固有の環境に適応できます。

- [フェイルオーバー後のロジカル・スタンバイ・データベースの構成](#)
- [フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換](#)
- [Open Resetlogs文の発行後のフラッシュバック・データベースの使用](#)
- [NOLOGGING句を指定した後のリカバリ](#)
- [OMFまたはOracle ASMを使用するスタンバイ・データベースの作成](#)
- [プライマリ・データベースでの書き込みの欠落エラーからのリカバリ](#)
- [DBCMPプロシージャを使用した欠落書き込みおよびその他の不一致の検出](#)
- [RMANバックアップを使用した障害が発生したプライマリのスタンバイ・データベースへの変換](#)
- [フィジカル・スタンバイの再作成を伴わないプライマリの文字セットの変更](#)
- [プライマリでのPDB PITRまたはPDBフラッシュバック後にスタンバイで必要なアクション](#)

15.1 フェイルオーバー後のロジカル・スタンバイ・データベースの構成

プライマリ・データベースを別のスタンバイ・データベースにフェイルオーバーした後にロジカル・スタンバイ・データベースで実行する必要があるステップを次に示します。

フェイルオーバーの発生後、ロジカル・スタンバイ・データベースは、元のプライマリ・データベースからの最後のREDOが適用されるまで、新規プライマリ・データベースのスタンバイ・データベースとして機能できません。これは、フェイルオーバー中に新規のプライマリ・データベースで最後のREDOが適用される場合と同じです。実行する必要があるステップは、新規プライマリ・データベースがフェイルオーバー前にフィジカル・スタンバイ・データベースであったかロジカル・スタンバイ・データベースであったかに応じて異なります。

- [新規プライマリ・データベースがフィジカル・スタンバイ・データベースだった場合](#)
- [新規プライマリ・データベースがロジカル・スタンバイ・データベースだった場合](#)

15.1.1 新規プライマリ・データベースがフィジカル・スタンバイ・データベースだった場合

このステップでは、プライマリ・ロールを担う前はフィジカル・スタンバイ・データベースだった新規プライマリ・データベースをサポートするようにロジカル・スタンバイ・データベースを構成する方法について説明します。

この使用例では、SATはロジカル・スタンバイ・データベース、NYCはプライマリ・データベースです。

1. SATデータベースで次の文を発行して、ログ・ファイルの自動リカバリを有効化するためにFAL_SERVERパラメータを構成します。

```
SQL> ALTER SYSTEM SET FAL_SERVER='<tns_name_to_new_primary>';  
SQL> ALTER SYSTEM SET FAL_CLIENT='<tns_name_to_local_database>';
```

2. PREPARE_FOR_NEW_PRIMARYルーチンを呼び出して、ロジカル・スタンバイ・データベースが新規プライマリ・データベースのスタンバイ・データベースとして機能できることを確認します。このステップの間、データに相違を生じさせるおそれのあるログ・ファイルのローカル・コピーは、ローカル・データベースから削除されます。その後、これらのログ・ファイルは、

新しいプライマリ・データベースから直接、再アーカイブが要求されます。

SATデータベースで、次の文を発行します。

```
SQL> EXECUTE DBMS_LOGSTDBY.PREPARE_FOR_NEW_PRIMARY( -  
> former_standby_type => 'PHYSICAL' -  
> dblink => 'nyc_link');
```

ノート:

ORA-16109 メッセージが戻され、alert.log に「LOGSTDBY: prepare_for_new_primary failure -- applied too far, flashback required.」という警告が書き込まれる場合は、次のステップを実行します。



- a. データベースを警告に示された SCN までフラッシュバックします。
- b. 続行する前に、このステップを繰り返します。

ロジカル・スタンバイ・データベースを Apply SCN までフラッシュバックする方法の例は、[「特定の適用済 SCN へのロジカル・スタンバイ・データベースのフラッシュバック」](#)を参照してください。

3. SATデータベースで、次のコマンドを発行してSQL Applyを開始します。

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

15.1.2 新規プライマリ・データベースがロジカル・スタンバイ・データベースだった場合

このステップでは、プライマリ・ロールを担う前はロジカル・スタンバイ・データベースだった新規プライマリ・データベースをサポートするようにロジカル・スタンバイ・データベースを構成する方法について説明します。

この使用例では、SATはロジカル・スタンバイ・データベース、NYCはプライマリ・データベースです。

1. 新規プライマリ・データベースで、ロジカル・スタンバイ・データベースのサポート準備が完了していることを確認します。
NYCデータベースで、次の問合せにより値NONEが戻されることを確認します。それ以外の場合、新しいプライマリ・データベースは、ロジカル・スタンバイ・データベースに対するサポートを可能にするために必要な作業を完了していません。
次に例を示します。

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

古いプライマリ・データベースを修復しようとする前に、NONEの値が戻される必要があります。

2. SATデータベースで次の文を発行して、ログ・ファイルの自動リカバリを有効化するためにFAL_SERVERパラメータを構成します。

```
SQL> ALTER SYSTEM SET FAL_SERVER='<tns_name_to_new_primary>';  
SQL> ALTER SYSTEM SET FAL_CLIENT='<tns_name_to_local_database>';
```

3. PREPARE_FOR_NEW_PRIMARYルーチンを呼び出して、ロジカル・スタンバイ・データベースが新規プライマリのスタンバイとして機能できることを確認します。このステップの間、データに相違を生じさせるおそれのあるログ・ファイルのローカル・コピーは、ローカル・データベースから削除されます。その後、これらのログ・ファイルは、新しいプライマリ・データベースから直接、再アーカイブが要求されます。

SATデータベースで、次の文を発行します。

```
SQL> EXECUTE DBMS_LOGSTDBY.PREPARE_FOR_NEW_PRIMARY( -  
> former_standby_type => 'LOGICAL' -  
> dblink => 'nyc_link');
```

ノート:

ORA-16109 メッセージが戻され、alert.log ファイルに「LOGSTDBY: prepare_for_new_primary failure -- applied too far, flashback required」という警告が書き込まれる場合は、次のステップを実行します。



- a. データベースを警告に示された SCN までフラッシュバックします。
- b. 続行する前に、このステップを繰り返します。

ロジカル・スタンバイ・データベースを Apply SCN までフラッシュバックする方法の例は、[「特定の適用済 SCN へのロジカル・スタンバイ・データベースのフラッシュバック」](#)を参照してください。

4. SATデータベースで、次のコマンドを発行してSQL Applyを開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY nyc_link;
```

この文は、常にリアルタイム適用オプションを有効化せずに発行する必要があります。ロジカル・スタンバイ・データベースでリアルタイム適用を有効化する場合がある場合は、前述の文が正常終了するまで待機してから、次の文を発行します。

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

15.2 フラッシュバック・データベースを使用した障害が発生したプライマリのスタンバイ・データベースへの変換

フェイルオーバーの発生後、元のプライマリ・データベースは、修正され、新しい構成のスタンバイ・データベースとして確立されるまで、Oracle Data Guard構成に含まれません。

これを行うには、フラッシュバック・データベース機能を使用して、障害の発生したプライマリ・データベースを障害発生前の時点にフラッシュバックし、その後新しい構成内のフィジカルまたはロジカル・スタンバイ・データベースに変換します。この項では、次の項目について説明します。

- [障害が発生したプライマリ・データベースのフィジカル・スタンバイ・データベースへのフラッシュバック](#)
- [障害が発生したプライマリ・データベースのロジカル・スタンバイ・データベースへのフラッシュバック](#)

ノート:



フェイルオーバー前に、元のプライマリ・データベースでフラッシュバック・データベースが使用可能になっている必要があります。詳細は、[『Oracle Database バックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照して

ください。

- [特定の適用済SCNへのロジカル・スタンバイ・データベースのフラッシュバック](#)

関連項目:

失敗したプライマリ・データベースを新しいデータベースとして自動回復する(フラッシュバック・データベースを使用するかわりに)方法は、[『Oracle Data Guard Broker』](#)を参照してください。

15.2.1 障害が発生したプライマリ・データベースのフィジカル・スタンバイ・データベースへのフラッシュバックに関する項

このステップは、古いプライマリ・データベースをフィジカル・スタンバイ・データベースとしてOracle Data Guard構成に戻します。

次のステップでは、フィジカル・スタンバイ・データベースに対してフェイルオーバーが実行され、フェイルオーバーの際に古いプライマリ・データベースでフラッシュバック・データベースが使用可能にされていることを前提としています。

1. 新しいプライマリ・データベース上で次の問合せを発行し、古いスタンバイ・データベースが新しいプライマリ・データベースになったSCNを判別します。

```
SQL> SELECT TO_CHAR(STANDBY_BECAME_PRIMARY_SCN) FROM V$DATABASE;
```

2. 必要に応じて古いプライマリ・データベースを停止し、マウントして、前のステップで判別したSTANDBY_BECAME_PRIMARY_SCNの値にフラッシュバックします。

```
SQL> SHUTDOWN IMMEDIATE;  
SQL> STARTUP MOUNT;  
SQL> FLASHBACK DATABASE TO SCN standby_became_primary_scn;
```

3. データベースをフィジカル・スタンバイ・データベースに変換するには、古いプライマリ・データベースで次の文を発行します。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

4. 新しいフィジカル・スタンバイ・データベースへのREDO転送を開始するには、新しいプライマリ・データベースで次のステップを実行します。

- a. 次の問合せを発行し、アーカイブ宛先の現在の状態を調べます。

```
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, -  
> ERROR, SRL FROM V$ARCHIVE_DEST_STATUS;
```

- b. 必要に応じて、宛先を使用可能にします。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
```

- c. スタンバイ・データベースが新しいプライマリ・データベースからのREDOデータの受信を確実に開始するよう、ログ・スイッチを実行し、これが成功したことを確認します。次のSQL文を新しいプライマリ・データベースで発行します。

```
SQL> ALTER SYSTEM SWITCH LOGFILE;  
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, -  
> ERROR, SRL FROM V$ARCHIVE_DEST_STATUS;
```

新しいスタンバイ・データベースで、REDO転送サービスがREDOデータを別のデータベースに転送しないよう、LOG_ARCHIVE_DEST_#初期化パラメータも変更する必要があります。

5. 新しいフィジカル・スタンバイ・データベースでREDO Applyを開始します。

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

REDO Applyは、ロールの推移により生成されるREDOレコードが検出されるたびに自動的に停止するため、新しいプライマリ・データベースがプライマリ・データベースとなったSCNを超えて適用されるまで、1回以上再起動する必要があります。障害が発生したプライマリ・データベースがリストアされてスタンバイ・ロールで稼働し始めた後、オプションでスイッチオーバーを実行し、それぞれのデータベースを元の(障害発生前の)ロールに推移できます。詳細は、「[フィジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)」を参照してください。

15.2.2 障害が発生したプライマリ・データベースのロジカル・スタンバイ・データベースへのフラッシュバック

このステップは、新しいプライマリ・データベースから古いプライマリ・データベースを正式にインスタンス化せずに、古いプライマリ・データベースを新しいロジカル・スタンバイ・データベースとしてOracle Data Guard構成に戻します。

次のステップでは、Oracle Data Guard構成ですでにロジカル・スタンバイ・データベースを使用したフェイルオーバーが実行済で、古いプライマリ・データベースでフラッシュバック・データベースが使用可能にされていることを前提としています。

1. フラッシュバックSCNおよびリカバリSCNを判別します。フラッシュバックSCNは、障害が発生したプライマリ・データベースのフラッシュバック先となるSCNです。リカバリSCNは、障害が発生したプライマリ・データベースのリカバリ先となるSCNです。新しいプライマリで次の問合せを発行し、これらのSCNを特定します。

```
SQL> SELECT merge_change# AS FLASHBACK_SCN, processed_change# AS RECOVERY_SCN -  
> FROM DBA_LOGSTDBY_HISTORY -  
> WHERE stream_sequence# = (SELECT MAX(stream_sequence#)-1 -  
> FROM DBA_LOGSTDBY_HISTORY);
```

2. 障害が発生したプライマリ・データベースを前のステップで特定したフラッシュバックSCNまでフラッシュバックします。

```
SQL> FLASHBACK DATABASE TO SCN flashback_scn;
```

3. 障害が発生したプライマリをフィジカル・スタンバイに変換し、リカバリの準備としてスタンバイ・データベースを再度マウントします。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

4. ログ・ファイルの自動リカバリを有効化するために、FAL_SERVERパラメータを構成します。フィジカル・スタンバイ(障害の発生したプライマリ)で、次の文を発行します。

```
SQL> ALTER SYSTEM SET FAL_SERVER='<tns_name_to_new_primary>';
```

5. フェイルオーバー操作以後に作成されたアーカイブ・ログをすべて、障害が発生したプライマリ・データベースから削除します。障害が発生したプライマリ・データベースがスタンバイから分離されていた場合は、現行のプライマリ・データベースと一貫性がない不一致のアーカイブ・ログが存在する可能性があります。これらの不一致のアーカイブ・ログを適用しないようにするには、バックアップおよび高速リカバリ領域から削除する必要があります。次のRMANコマンドを使用すると、関連アーカイブ・ログを高速リカバリ領域から削除できます。

```
RMAN> DELETE FORCE ARCHIVELOG FROM SCN FLASHBACK_SCN;
```

削除されると、これらの不一致のログおよび後続のトランザクションはリカバリできなくなります。

6. ステップ1で特定したリカバリSCNまでリカバリします。

```
SQL> RECOVER MANAGED STANDBY DATABASE UNTIL CHANGE recovery_scn;
```

7. データベース・ガードを有効にします。

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

8. フィジカル・スタンバイをアクティブにしてプライマリ・データベースにします。

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

9. データベースをオープンします。

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

10. 新しいプライマリへのデータベース・リンクを作成してSQL Applyを開始します。

```
SQL> CREATE PUBLIC DATABASE LINK mylink -  
> CONNECT TO system IDENTIFIED BY password -  
> USING 'service_name_of_new_primary_database';  
  
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY mylink;
```

これで、ロールの可逆的な推移が完了しました。

15.2.3 特定の適用済SCNへのロジカル・スタンバイ・データベースのフラッシュバック

スタンバイ・データベースのメリットの1つは、プライマリ・データベース・サービスに影響を与えずに、スタンバイ・データベースでフラッシュバック・データベースを実行できることです。データベースを特定の時点までフラッシュバックするタスクは簡単ですが、ロジカル・スタンバイ・データベースでは、既知のトランザクションがコミットされる直前の時点までフラッシュバックする操作が必要になる場合があります。このような必要が生じるのは、フェイルオーバー後に新しいプライマリ・データベースを使用してロジカル・スタンバイ・データベースを構成する場合です。これらのステップでは、フラッシュバック・データベースとSQL Applyを使用して既知の適用済SCNまでリカバリする方法について説明します。

1. プライマリでの既知のSCN(APPLIED_SCN)の特定後に、フラッシュバック操作に使用するためにロジカル・スタンバイ・データベースで次の問合せを発行して対応するSCNを判別します。

```
SQL> SELECT DBMS_LOGSTDBY.MAP_PRIMARY_SCN (PRIMARY_SCN => APPLIED_SCN) -  
> AS TARGET_SCN FROM DUAL;
```

2. 次のSQL文を発行し、ロジカル・スタンバイ・データベースを指定したSCNまでフラッシュバックし、RESETLOGSオプションを指定してロジカル・スタンバイ・データベースをオープンします。

```
SQL> SHUTDOWN;  
SQL> STARTUP MOUNT EXCLUSIVE;  
SQL> FLASHBACK DATABASE TO SCN <TARGET_SCN>;  
SQL> ALTER DATABASE OPEN RESETLOGS;
```

3. 次の問合せを発行して、SQL ApplyがAPPLIED_SCNまで適用されたことを確認します。

```
SQL> SELECT APPLIED_SCN FROM V$LOGSTDBY_PROGRESS;
```

15.3 Open Resetlogs文の発行後のフラッシュバック・データベースの使用

Oracle Data Guard構成のプライマリ・データベースで、スタンバイ・データベースがリアルタイム適用を使用しているというエラーが発生した場合、同じエラーがスタンバイ・データベースにも適用されます。フラッシュバック・データベースが有効になっている場合

は、プライマリ・データベースとスタンバイ・データベースをエラー前の状態に戻すことができます。

これを行うには、プライマリ・データベースでFLASHBACK DATABASE文とOPEN RESETLOGS文を実行してから、適用サービスを再起動する前に、スタンバイ・データベースで同様のFLASHBACK STANDBY DATABASE文を発行します。(フラッシュバック・データベースが有効化されていない場合は、プライマリ・データベースのポイント・イン・タイム・リカバリを実行後、[「フィジカル・スタンバイ・データベースの作成」](#)と[「ロジカル・スタンバイ・データベースの作成」](#)で説明しているスタンバイ・データベースの再作成が必要です。)

15.3.1 特定時点へのフィジカル・スタンバイ・データベースのフラッシュバック

これらのステップでは、プライマリ・データベースでOPEN RESETLOGS文を発行した後、フィジカル・スタンバイ・データベースの再作成を回避する方法を説明します。

Oracle Databaseリリース19cでは、スタンバイがマウント・モードで、かつフラッシュバック・データベースが有効になっている場合、何も実行する必要はありません。一般的なケースでは、十分なフラッシュバック・データが使用可能な場合は、引き続きスタンバイがプライマリを継ぐように、スタンバイは自動的にフラッシュバックされます。

スタンバイがオープンし、フラッシュバック・データベースが有効になっている場合は、スタンバイをマウント・モードにしてからスタンバイ・リカバリを開始できます。プライマリをスタンバイに継がせるための試みとして、MRPにより、スタンバイの自動フラッシュバックがトリガーされます。この操作が成功した場合は、アラート・ログにレポートされるため、続いてActive Data Guardインスタンスを開く必要があります。

自動フラッシュバックがトリガーされない場合、または自動フラッシュバックによってプライマリをスタンバイが継がなかった場合は、この項で説明するステップを実行できます。

1. RESETLOGS操作が発生した前のSCNを判別します。たとえば、プライマリ・データベースで、次の問合せを使用してRESETLOGS操作がプライマリ・データベースで発生したよりも2つ前のシステム変更番号(SCN)の値を取得します。

```
SQL> SELECT TO_CHAR(RESETLOGS_CHANGE# - 2) FROM V$DATABASE;
```

2. スタンバイ・データベースで、次の問合せを使用して現行のSCNを取得します。

```
SQL> SELECT TO_CHAR(CURRENT_SCN) FROM V$DATABASE;
```

3. CURRENT_SCNの値がresetlogs_change# - 2の値より大きい場合、次の文を発行してスタンバイ・データベースをフラッシュバックします。

```
SQL> FLASHBACK STANDBY DATABASE TO SCN resetlogs_change# -2;
```

- CURRENT_SCNの値がresetlogs_change# - 2の値より小さい場合、ステップ4に進みます。
- スタンバイ・データベースのSCNがプライマリ・データベースのSCNよりも大幅に遅れており、OPEN RESETLOGS文のREDOの新規ブランチがスタンバイに登録されている場合、適用サービスは、OPEN RESETLOGS文中も停止せずに継続できます。この場合、適用サービスはREDOデータ内のOPEN RESETLOGS文に到達しても停止しないため、データベースのフラッシュバックは不要です。

4. フィジカル・スタンバイ・データベースでREDO Applyを開始するには、次の文を発行します。

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

スタンバイ・データベースはプライマリ・データベースからREDOを受信し、適用できます。

15.3.2 特定時点へのロジカル・スタンバイ・データベースのフラッシュバック

これらのステップでは、プライマリ・データベースをフラッシュバックし、OPEN RESETLOGS文を発行してオープンした後、ロジカル・スタ

ンバイ・データベースの再作成を回避する方法を説明します。

ノート:



SQL Apply は、プライマリ・データベースでリセットログ操作の発生を検出すると、ロジカル・スタンバイ・データベースをフラッシュバックしなくてもマイニングできる場合は、REDO の正しいブランチを自動的にマイニングします。それ以外の場合、SQL Apply は「ORA-1346: 指定されたりセット・ログ scn を超えて LogMiner が REDO を処理しました」エラーで停止します。この項では、SQL Apply はこのエラーですでに停止していると仮定しています。

1. プライマリ・データベースで、次の問合せを使用してRESETLOGS操作がプライマリ・データベースで発生したよりも2つ前のシステム変更番号(SCN)の値を取得します。

```
SQL> SELECT TO_CHAR(RESETLOGS_CHANGE# - 2) AS FLASHBACK_SCN FROM V$DATABASE;
```

2. ロジカル・スタンバイ・データベースでフラッシュバック操作のターゲットSCNを特定します。

このステップでは、PRIMARY_SCNのFLASHBACK_SCN値はステップ1から得られたものです。

```
SQL> SELECT DBMS_LOGSTDBY.MAP_PRIMARY_SCN (PRIMARY_SCN => FLASHBACK_SCN) -  
> AS TARGET_SCN FROM DUAL;
```

3. 次のSQL文を発行し、ロジカル・スタンバイ・データベースを指定したSCNまでフラッシュバックし、RESETLOGSオプションを指定してロジカル・スタンバイ・データベースをオープンします。

```
SQL> SHUTDOWN;  
SQL> STARTUP MOUNT EXCLUSIVE;  
SQL> FLASHBACK DATABASE TO SCN <TARGET_SCN>;  
SQL> ALTER DATABASE OPEN RESETLOGS;
```

4. SQL Applyが開始される前に、プライマリの新しいブランチからのログ・ファイルが登録されていることを確認します。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 resetlogs_id FROM V$DATABASE_INCARNATION WHERE status = 'CURRENT';
```

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 * FROM DBA_LOGSTDBY_LOG WHERE resetlogs_id = resetlogs_id_at_primary;
```

1以上の行が返された場合、これは、プライマリの新しいブランチからの登録されたログ・ファイルであることを確認します。

5. SQL Applyを開始します。

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

15.4 NOLOGGING句を指定した後のリカバリ

一部のSQL文では、NOLOGGING句を指定して、操作がオンラインREDOログ・ファイルに記録されないようにできます。

実際には、REDOレコードは引き続きオンラインREDOログ・ファイルに書き込まれますが、このレコードにはデータが関連付けられていません。これによりスタンバイ・サイトでのログ・アプリケーションまたはデータ・アクセス・エラーが発生し、このログ・ファイルを適用するには手動でのリカバリが必要になることがあります。ロジカル・スタンバイとフィジカル・スタンバイのどちらを使用するかに応じて次の操作を行うことで、これらのエラーを防止できます。

- ロジカル・スタンバイ- CREATE DATABASEまたはALTER DATABASE文でFORCE LOGGING句を指定します。
- フィジカル・スタンバイ - [適切なロギング・モードの有効化](#)の説明に従って、Data Guard構成の使用方法に適したロギング・モードを指定します。

現在のロギング・モードは、V\$DATABASE.FORCE_LOGGING列(CDBの場合)またはDBA_PDBS.FORCE_LOGGING列(PDBの場合)で確認できます。

15.4.1 ロジカル・スタンバイ・データベースのリカバリ・ステップ

ロジカル・スタンバイ・データベースでは、SQL ApplyでNOLOGGING句を使用してスキップされていない表で実行された操作のREDOレコードが検出されると、次のエラーで停止します。

```
ORA-16211 unsupported record found in the archived redo log
```

NOLOGGING句を指定した後のリカバリには、[「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。

ノート:



通常、NOLOGGING 句の使用はお薦めしません。このため、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、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。これは、DBMS_LOGSTDBY.SKIP プロシージャを使用して行うことができます。

15.4.2 フィジカル・スタンバイ・データベースのリカバリ・ステップ

REDO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リカバリ不能とマークされます。フィジカル・スタンバイ・データベースにフェイルオーバーするか、読取り専用アクセスでスタンバイ・データベースをオープンし、UNRECOVERABLEのマークが付けられたブロックの範囲を読み取ろうとすると、次のようなエラー・メッセージが表示されます。

```
ORA-01578: ORACLE data block corrupted (file # 1, block # 2521)
ORA-01110: data file 1: '/oracle/dbs/stdby/tbs_1.dbf'
ORA-26040: Data block was loaded using the NOLOGGING option
```

STANDBY NOLOGGING FOR LOAD PERFORMANCEモードを使用した場合も、フィジカル・スタンバイで実行された問合せがNOLOGGING操作による破損ブロックをレポートすることがあります。ただし、そのスタンバイでActive Data Guard構成の一部として管理リカバリが実行されている場合は、プライマリから置換ブロックが自動的にフェッチされます。(管理リカバリはALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT文を使用して開始されます。)管理リカバリは、そのようなブロックが最初に作成された直後にそのすべてをフェッチしようとし、それらの修正を成功するまで試行し続けます。管理リカバリが開始されるたびに、前のリカバリ・セッションで未処理だったブロックの修正が再開されます。この自動リカバリは、STANDBY NOLOGGING FOR LOAD PERFORMANCEモードを有効にして生成されたREDOが原因で発生した破損ブロックにのみ適用されます。従来のログに記録されない操作によって破損したブロックには、次の手順を使用する必要があります。このステップは、ログに記録されないすべてのブロックをリカバリする簡単な方法を示しています。(リカバリ不能な操作の後にバックアップが必要かどうかの決定 およびフィジカル・スタンバイ・データベースの一部分のリカバリなど、その他のアプローチについては後続の項を参照してください。)

1. スタンバイでリカバリを停止します。


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

2. RMANをスタンバイに接続し、次のコマンドを発行して、ログに記録されないブロックをリカバリします。

```
RMAN> RECOVER DATABASE NONLOGGED BLOCK;
```

ログに記録されないブロックがスイッチオーバー後にのみ検出される場合、これらの2つのステップを使用できますが、プライマリ・データベースを(開かずに)単にマウントして、RMANをプライマリに接続する必要があります。

RMANのRECOVERコマンドで、ログに記録されないブロックの一部をリカバリできないこともあります。これが起きる理由については、RMANコマンドが実行されたデータベースのアラート・ログに詳しく記載されています。最も一般的な理由は、プライマリで最近ブロックが変更されたが、まだ対応するデータファイルに書き込まれていないためです。つまり、スタンバイに送信されたブロックが古すぎて、スタンバイでリカバリ不能なブロックを置き換えることができないのです。これを解決するには、ブロックをデータファイルに書き込んだ後で、再びRECOVERコマンドを発行します。

次に、RECOVERコマンドを実行して一部のブロックがリカバリ不能になった場合の、アラート・ログ・エントリの例を示します。

```
Started Nonlogged Block Replacement recovery on file 7 (ospid 13005 rcvid 11319003446180375696)
Finished Nonlogged Block Replacement recovery on file 7. 5 blocks remain
Statistics for replacement block source database (service=dg3tns)
(Use of it stopped due to error 12942 received from it)
Blocks requested 5, blocks received 0.
```

Reason replacement blocks accepted or rejected	Blocks	Last block
Not received: Rejected by sender. Wrong state or SCN	5	21

この例では、スタンバイでコマンドが実行され、プライマリはブロックを送信せず、かわりに次のOracleエラーをレポートしました。

```
ORA-12942: database incarnation at source does not match
```

アラート・ログを調べたところ、プライマリでFLASHBACK DATABASEおよびOPEN RESETLOGSコマンドが実行されたが、スタンバイはフラッシュバックされていないことが判明しました。つまり、スタンバイは現在REDOの孤立したブランチにいるため、プライマリは正しいバージョンであることが確認されているデータ・ブロックを提供できなかったのです。

ノート:



Data Guard 構成に複数のスタンバイがある場合、スタンバイで実行される RMAN の RECOVER コマンドは、プライマリからのみブロックを取得しようとします。RECOVER コマンドをプライマリで実行した場合、適切なブロックを取得する可能性が最も高い 1 つのスタンバイ(通常、プライマリと最も密接に同期しているスタンバイ)からのみブロックを取得しようとします。

15.4.3 リカバリ不能処理後にバックアップが必要かどうかの判断

プライマリ・データベースでリカバリ不能な操作を実行した場合、新しいバックアップ操作が必要かどうかを判断する必要があります。

これを行うには、次のステップを実行します。

1. プライマリ・データベースでV\$DATAFILEビューの問合せを行い、Oracleデータベースが最新の無効なREDOデータを生成した**システム変更番号(SCN)**または時刻を特定します。
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。

```
SQL> SELECT UNRECOVERABLE_CHANGE#, -  
> TO_CHAR (UNRECOVERABLE_TIME, 'mm-dd-yyyy hh:mi:ss') -  
> FROM V$DATAFILE;
```

3. 前のステップの問合せで、データファイルが最後にバックアップされた時間よりも最近のデータファイルに対するリカバリ不能時間がレポートされる場合、問題のデータファイルの別のバックアップを作成します。

V\$DATAFILEビューの詳細は、『[Oracle Databaseリファレンス](#)』を参照してください。

15.4.4 フィジカル・スタンバイ・データベースの一部分のリカバリ・ステップ

RMANのRECOVER ... NONLOGGED BLOCKコマンドを使用して、データ・ファイルのセット、表領域のセット、または単に1つのプラガブル・データベース(PDB)およびマルチテナント・コンテナ・データベース(CDB)に属するブロックをリカバリできます。

この機能は、一部のデータ・ファイルに、ログに記録されないブロックを残すことが容認される場合に便利です(たとえば、現在削除されているオブジェクトに行をロードした結果として、ログに記録されないブロックが作成されたことがはっきりと分かっている場合など)。

V\$NONLOGGED_BLOCKビューには通常、各データ・ファイルの既知の無効なブロックの範囲がリストされ、エントリはメディア・リカバリの一環として保持されます。ただし、情報が完全でないことがあります。通常、これはOracle Database 12cリリース1 (12.1)より前のリリースからアップグレードした後、またはデータ・ファイルのオペレーティング・システム・バックアップをリストアした後などです。次にメディア・リカバリを実行すると、失効エントリは削除されて新たに無効化されたブロックが記録されますが、前の無効なブロックはV\$NONLOGGED_BLOCKにエントリがありません。データ・ファイルにV\$NONLOGGED_BLOCKエントリがない場合でも、V\$DATAFILEビューのFIRST_NONLOGGED_SCN列を使用して、データ・ファイルに少なくとも1つの無効なブロックが存在することを確認できます。

RMANコマンドVALIDATE ... NONLOGGED BLOCKを使用して、V\$NONLOGGED_BLOCKのエントリをデータ・ファイルと再び同期できます。それには、既存の範囲が完全かどうかを判断し、完全でない場合は、必要なデータ・ファイルをスキャンして無効なブロックを識別し、それらがV\$NONLOGGED_BLOCKのエントリによって取得されるようにします。VALIDATE ... NONLOGGED BLOCKコマンドには、単にデータ・ファイルのセット、表領域のセット、またはPDBやCDBを検証するための、RECOVER ... NONLOGGED BLOCKと同じオプションがあります。

ノート:



V\$NONLOGGED_BLOCK ビューのエントリは無効なブロックのスーパーセットであるため、無効なブロックに近い正常なブロックがいくつか含まれる可能性があります。たとえば、ブロック 100 で始まるファイル 7 のエントリに、50 のブロックが含まれる場合、これら 50 のブロックは一部または全部が無効であるか、いずれも無効ではありません。VALIDATE コマンドを実行すると、無効なブロックがない範囲はなくなり、範囲の最初と最後のブロックも無効なブロックになります。ただし、制御ファイルで保持できる範囲の合計数には制限があるため、同じファイルの範囲を結合しなければならないことがあり、それが通常のブロックが範囲に含まれる原因となります。

オフライン・データ・ファイルのみを検証またはリカバリする場合は、RMANコマンドの実行時に、それらが属するデータベースを開くことができます。

15.5 OMFまたはOracle ASMを使用するスタンバイ・データベースの作成

スタンバイ・データベースを作成するときは、プライマリ・データベースでOracle Managed Files (OMF)またはOracle Automatic Storage Management (Oracle ASM)を使用する場合に実行する必要がある追加ステップがあります。

この項の説明は、すでにフィジカル・スタンバイ・データベースの作成方法を理解しており、RMAN、OMFおよびOracle ASM機能に習熟していることを前提としています。

スタンバイ・データベースを作成する前に、次の準備作業を実行します。

1. プライマリ・データベースで強制ロギングを使用可能にします。
2. プライマリ・データベースでアーカイブを使用可能にします。
3. プライマリ・データベースで必要な初期化パラメータをすべて設定します。
4. スタンバイ・データベース用に初期化パラメータ・ファイルを作成します。
5. プライマリ・データベースがOMFを使用するよう構成されている場合、スタンバイ・データベースもOMFを使用するよう構成することをお勧めします。これを行うには、DB_CREATE_FILE_DESTおよびDB_CREATE_ONLINE_LOG_DEST_1の初期化パラメータに適切な値を設定します。プライマリおよびスタンバイ・データベースの両方で同じディスク・グループ名が使用されている場合、メンテナンスや後のロールの推移が簡易化されます。

ノート:



スタンバイに OMF パラメータが設定されている場合は、そのスタンバイの新しいファイルはプライマリでの作成方法とは関係なく、常に OMF として作成されます。したがって、DB_FILE_NAME_CONVERT および DB_CREATE_FILE_DEST パラメータの両方がスタンバイに設定されている場合は、DB_CREATE_FILE_DEST パラメータが優先されます。

6. STANDBY_FILE_MANAGEMENT初期化パラメータにAUTOを設定します。
7. スタンバイ・データベースに接続するために、必要に応じてOracle Netを構成します。
8. REDO転送の認証を構成します。詳細は、[「REDO転送の認証の構成」](#)を参照してください。
9. 制御ファイルをマウントせずにスタンバイ・データベース・インスタンスを起動します。

スタンバイ・データベースを作成するには、次の作業を実行します。

1. スタンバイ・データベースでOracle ASMを使用する場合、スタンバイ・データベース・システムにOracle ASMインスタンスが存在していなければ、インスタンスを作成します。
2. RMANのBACKUPコマンドを使用して、プライマリ・データベースのデータファイル、アーカイブ・ログ・ファイルおよびスタンバイ制御ファイルのコピーが含まれているバックアップ・セットを作成します。
3. RMANのDUPLICATE FOR STANDBYコマンドを使用して、バックアップ・セットのデータファイル、アーカイブREDOログ・ファイルおよびスタンバイ制御ファイルをスタンバイ・データベースの格納場所にコピーします。

DUPLICATE FOR STANDBYコマンドは、スタンバイ・インスタンスで実際のデータ移動を実行します。バックアップ・セットがテープ上にある場合、スタンバイ・インスタンスがバックアップ・セットを読み取れるように、メディア・マネージャを構成しておく必要があります。バックアップ・セットがディスク上にある場合は、ネットワーク・ファイル記憶域(NFS)でプライマリ・パス名を使用できるようにするか、スタンバイ・システムにコピーしてRMANのCATALOG BACKUPPIECEコマンドを使用し、バックアップ・ピースをカタログに追加してからリストアを行うことで、スタンバイ・インスタンスがバックアップ・ピースを読み取れるようにする必要があります。

これらのステップを完了した後、[「フィジカル・スタンバイ・データベースが正しく実行されているかどうかの確認」](#)のステップを実行してフィジカル・スタンバイ・データベースの構成を確認します。

ロジカル・スタンバイ・データベースを作成するには、[「ロジカル・スタンバイ・データベースの作成」](#)で説明されているスタンバイ・データベースの作成プロセスを実行しますが、次の点を変更します。

1. ロジカル・スタンバイ・データベースの場合、DB_CREATE_FILE_DESTパラメータを設定しても、OMFファイル名を強制的に作成しません。ただし、このパラメータをプライマリ・データベースで設定する場合、スタンバイ・データベースでも設定する必要があります。
2. プライマリ・システムでロジカル・スタンバイ制御ファイルを作成した後、このファイルをスタンバイ・システムにコピーするために、オペレーティング・システムのコマンドを使用しないでください。かわりにRMANのRESTORE CONTROLFILEコマンドを使用して、ロジカル・スタンバイ制御ファイルのコピーをスタンバイ・システムにリストアします。
3. プライマリ・データベースでOMFファイルを使用する場合、スタンバイ・データベースに作成された新しいOMFファイルをスタンバイ・データベース制御ファイルが使用するよう、RMANを使用してスタンバイ・データベース制御ファイルを更新します。この操作を実行するには、次の例に示すように、スタンバイ・データベースにのみ接続します。

```
> RMAN TARGET sys@lstdby

target database Password: password

RMAN> CATALOG START WITH '+stby_diskgroup';
RMAN> SWITCH DATABASE TO COPY;
```

これらのステップを完了した後、[「ロジカル・スタンバイ・データベースのオープン」](#)のステップを実行してロジカル・スタンバイ・データベースの起動、リカバリおよび検証を行います。

関連項目:

- [フィジカル・スタンバイ・データベースの作成](#)
- [ロジカル・スタンバイ・データベースの作成](#)
- [Recovery Managerを使用したスタンバイ・データベースの作成](#)
- OMFの詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。
- Oracle ASMの詳細は、[Oracle Automatic Storage Management管理者ガイド](#)を参照
- RMANの詳細は、[『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』](#)を参照してください。

15.6 プライマリ・データベースでの書込みの欠落エラーからのリカバリ

Oracle Data Guard構成でのメディア・リカバリの実行中に、フィジカル・スタンバイ・データベースを使用してプライマリ・データベースで書込みの欠落によるデータ破損エラーを検出できます。

これは、プライマリ・データベースのREDOログに格納されているブロックのSCNをフィジカル・スタンバイ・データベースのブロックのSCNと比較して行います。プライマリ・データベースのブロックのSCNがスタンバイ・データベースのSCNよりも小さい場合は、プライマリ・データベースで書込みの欠落が発生しています。

そのような状況で、書込み欠落の検出([DB_LOST_WRITE_PROTECT](#)初期化パラメータで設定)がプライマリとスタンバイ両方のレベルで有効な場合、スタンバイでリカバリを試行するとORA-752エラーになります。書込み欠落の検出が有効でない場合、リカバリを試行するとORA-600 [3020]エラーになります。ただし、ORA-600 [3020]エラーがすべて、プライマリにおける書込み欠落に起因するとはかぎりません。したがって、この項で説明するガイドラインに従う前にOracle Supportの担当者と協力し、ORA-600 [3020]エラーの根本原因が本当に、プライマリで発生した書込みの欠落なのかどうかを判断します。My Oracle

Supportのノート1265884.1 (<http://support.oracle.com>)の『Resolving ORA-752 or ORA-600 [3020] During Standby Recovery』も参照してください。

ノート:



書き込み欠落のエラーはプライマリによってブロックがキャッシュに読み取られた後、対応する REDO をスタンバイのブロックと比較した場合にのみ検出されるので、プライマリおよびスタンバイの両方で読取りや検証がまだ行われていない未検出ブロックが存在している可能性があります。古いブロックを読み取るまでに問合せや更新のためにスタンバイ上で使用したブロックのうち、現在適用している REDO の SCN までの全ブロックはスタンバイによる検証済なので、古いブロックが現行のデータベース操作に影響を及ぼすことはありません。

プライマリの書き込みの欠落エラーがスタンバイで検出されると、失効したブロックごとに次のようなブロック・エラー・メッセージが1つ以上スタンバイ・データベースのアラート・ファイルに出力されます。

```
Tue Dec 12 19:09:48 2006
STANDBY REDO APPLICATION HAS DETECTED THAT THE PRIMARY DATABASE
LOST A DISK WRITE OF BLOCK 26, FILE 7
NO REDO AT OR AFTER SCN 389667 CAN BE USED FOR RECOVERY.
```

次に、アラート・ファイルは、ORA-00752エラーがスタンバイ・データベースに発生し、管理リカバリが取り消されたことを示します。

```
Slave exiting with ORA-752 exception
Errors in file /oracle/log/diag/rdbms/dgstwite2/stwite2/trace/stwite2_pr00_23532.trc:
ORA-00752: recovery detected a lost write of a data block
ORA-10567: Redo is inconsistent with data block (file# 7, block# 26)
ORA-10564: tablespace TBS_2
ORA-01110: data file 7: '/oracle/dbs/btbs_21.f'
ORA-10561: block type 'TRANSACTION MANAGED DATA BLOCK', data object# 57503
```

次に、スタンバイ・データベースは、アラート・ファイルに出力されたSCN時点で、このエラーによるデータファイルの破損がない、一貫した状態にリカバリされます。

```
Recovery interrupted!
Recovered data files to a consistent state at change 389569
```

この最後のメッセージは、アラート・ファイルのかなり後の方に表示されることがあり、ブロック・エラー・メッセージより小さいSCNの場合もあります。また、プライマリ・データベースは、すでにデータファイルに破損があってもエラーが表示されずに動作している場合があります。

このようなエラーからリカバリするための推奨手順は、次のステップに示すように、フィジカル・スタンバイにフェイルオーバーします。

プライマリで書き込みの欠落が検出された後のフィジカル・スタンバイへのフェイルオーバーのステップ

1. プライマリ・データベースを停止します。ブロック・エラー・メッセージに出力されたSCN以降のデータがすべて消失します。
2. スタンバイ・データベースで次のSQL文を発行してプライマリに変換します。

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



```
Database altered.
```

```
Tue Dec 12 19:15:23 2006
```

```
alter database activate standby database
```

```
ALTER DATABASE ACTIVATE [PHYSICAL] STANDBY DATABASE (stwrite2)
```

```
RESETLOGS after incomplete recovery UNTIL CHANGE 389569
```

```
Resetting resetlogs activation ID 612657558 (0x24846996)
```

```
Online log /oracle/dbs/bt_log1.f: Thread 1 Group 1 was previously cleared
```

```
Online log /oracle/dbs/bt_log2.f: Thread 1 Group 2 was previously cleared
```

```
Standby became primary SCN: 389567
```

```
Tue Dec 12 19:15:23 2006
```

```
Setting recovery target incarnation to 3
```

```
Converting standby mount to primary mount.
```

```
ACTIVATE STANDBY: Complete - Database mounted as primary (stwrite2)
```

```
Completed: alter database activate standby database
```

3. 新しいプライマリのバックアップを行います。データベースの完全なバックアップ・コピーがなければ、フェイルオーバー後の変更をリカバリすることができないので、バックアップをすぐに実行することは安全策として必要な作業です。フェイルオーバーを行うと元のプライマリ・データベースはOracle Data Guard構成に含まれなくなり、その他のすべてのスタンバイ・データベースが新しいプライマリ・データベースからREDOデータの受信と適用を行うようになります。
4. 新規プライマリ・データベースをオープンします。
5. オプションのステップとして、失敗したプライマリをフィジカル・スタンバイとして再作成します。これはステップ3で新しいプライマリで作成したデータベース・バックアップを使用して実行できます。(この場合は、フラッシュバック・データベースまたはOracle Data Guardブローカを使用して、元のプライマリ・データベースを回復することはできません。)新しいプライマリから作成したバックアップを使用して作成したフィジカル・スタンバイは、古いスタンバイと同じデータ・ファイルを保持することに注意してください。したがって、新しいスタンバイでは同じブロックを比較するため、古いスタンバイがアクティブ化される前に存在していた未検出の書込みの欠落は、新しいスタンバイによって検出されません。プライマリまたはスタンバイで発生した新しい書込みの欠落はすべて検出されます。

関連項目:

書込み欠落の検出有効化の詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド](#)』を参照してください。

15.7 DBCOMPプロシージャを使用した欠落書込みおよびその他の不一致の検出

PL/SQLプロシージャDBMS_DBCOMP.DBCOMPを使用して、欠落書込みの検出およびプライマリ・データベースとフィジカル・スタンバイ・データベースの間の不一致の検出も可能です。

DBMS_DBCOMP.DBCOMPプロシージャでは、プライマリ・データベースとフィジカル・スタンバイ・データベースの同じデータ・ブロックが比較されます。このプロシージャはいつでも実行できます。(DB_LOST_WRITE_PROTECT初期化パラメータを有効化する必要はありません。)

V\$SESSION_LONGOPSビューを問い合わせると、継続中のブロック比較操作の進捗状況を監視できます。



ノート:

ロジカル・スタンバイ・データベース、遠隔同期インスタンスおよびカスケード・スタンバイは、DBMS_DBCOMP.DBCOMPプロシージャのターゲット・データベースにはなりません。

DBMS_DBCOMP.DBCOMPプロシージャでは、1つのプライマリ・データベースと1つ以上のフィジカル・スタンバイ・データベースがあることを前提としています。データベースはブロック比較の前に少なくともマウントされている必要があります。

次の例の場合、一意の名前のdgmainを持ったプライマリ・データベースがあり、フィジカル・スタンバイ・データベースの名前がdgmainb、dgmainc、dgmaindなどであると仮定します。

例15-1 プライマリおよびすべてのスタンバイがマウントまたはオープンされており、DBCOMPがプライマリから実行される場合

この場合、DBCOMPプロシージャをプライマリ・データベースから実行すると、指定されたデータ・ファイルはプライマリとすべてのフィジカル・スタンバイ・データベースの間でブロック単位で比較されます。たとえば、次の問合せを実行するとします。

```
SQL> exec sys.dbms_dbcomp.dbcomp( '1', 'BlockCompare', :retval);
```

生成される出力ファイルはBlockCompare_dgmainb_1およびBlockCompare_dgmainc_d_1です。

例15-2 プライマリおよびすべてのスタンバイがマウントまたはオープンされており、DBCOMPがスタンバイから実行される場合

この場合、DBCOMPプロシージャをスタンバイ・データベースの1つ(たとえばdgmainb)から実行すると、指定されたデータ・ファイルはプライマリと特定のスタンバイ・データベースの間でのみ比較されます。その他のスタンバイ・データベースは考慮されません。たとえば、次の問合せを実行するとします。

```
SQL> exec sys.dbms_dbcomp.dbcomp( '1', 'BlockCompare', :retval);
```

生成される出力ファイルはBlockCompare_dgmain_1です。

例15-3 プライマリがマウントまたはオープンされているが、スタンバイはすべてがマウントまたはオープンされているとはかぎらず、DBCOMPがプライマリから実行される場合

この場合、DBCOMPプロシージャをプライマリで実行すると、指定されたデータ・ファイルはプライマリ・データベースとマウントまたはオープンされたフィジカル・スタンバイ・データベースの間で比較されます。マウントもオープンもされていないスタンバイ・データベースの場合、処理はされません。

例15-4 プライマリがマウントまたはオープンされているが、スタンバイはすべてがマウントまたはオープンされているとはかぎらず、DBCOMPがスタンバイから実行される場合

この場合、指定されたデータ・ファイルはプライマリとDBCOMPプロシージャの実行元のスタンバイの間で比較されます。

例15-5 プライマリがマウントされていないが、複数のスタンバイがマウントまたはオープンされている場合

プライマリ・データベースはマウントもオープンもされていないため、DBCOMPプロシージャは、比較するプライマリとフィジカル・スタンバイ・データベースの適切なペアを見つけることができません。ORAエラー・メッセージは返されませんが、次のようなメッセージが対応する出力ファイルに出力されます: Remote database is not in the primary role.

例15-6 プライマリがマウントまたはオープンされているが、マウントまたはオープンされているスタンバイがない場合

プライマリとフィジカル・スタンバイ・データベースの適切なペアが見つからないため、対応する出力ファイルにメッセージが出力されますが、ORAエラーは返されません。

関連トピック

- [Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス](#)

15.8 RMANバックアップを使用した障害が発生したプライマリのスタンバイ・データベースへの変換

障害が発生したプライマリ・データベースを変換するには、プライマリでフラッシュバック・データベース機能を有効にし、必要に応じて次のいずれかの手順に従うことをお勧めします。

- [障害が発生したプライマリ・データベースのフィジカル・スタンバイ・データベースへのフラッシュバック](#)
- [障害が発生したプライマリ・データベースのロジカル・スタンバイ・データベースへのフラッシュバック](#)

これらの項の手順では、障害が発生したプライマリをフィジカル・スタンバイまたはロジカル・スタンバイに変換する最も速い方法を説明しています。しかし、障害が発生したプライマリでフラッシュバック・データベースが無効になっている場合でも、次のトピックで説明するように、障害が発生したプライマリのローカル・バックアップを使用して、障害が発生したプライマリをフィジカル・スタンバイまたはロジカル・スタンバイに変換できます。

- [RMANバックアップを使用した、障害が発生したプライマリのフィジカル・スタンバイへの変換](#)
- [RMANバックアップを使用した障害が発生したプライマリのロジカル・スタンバイへの変換](#)

15.8.1 RMANバックアップを使用した、障害が発生したプライマリのフィジカル・スタンバイへの変換

このステップでは、RMANバックアップを使用して障害が発生したプライマリをフィジカル・スタンバイに変換する方法について説明します。

この手順には、古いプライマリのCOMPATIBLE初期化パラメータが11.0.0以上に設定されている必要があります。

1. 新しいプライマリ・データベース上で次の問合せを発行し、古いスタンバイ・データベースが新しいプライマリ・データベースになったSCNを判別します。

```
SQL> SELECT TO_CHAR(STANDBY_BECAME_PRIMARY_SCN) FROM V$DATABASE;
```

2. スタンバイが新しいプライマリになったSCN(standby_became_primary_scn)に古いプライマリが到達する前に作成したバックアップを使用して、データベースをリストアします。次に、Point-in-Timeリカバリを実行して、古いプライマリをそれと同じ時点までリカバリします。

次のRMANコマンドを発行します。

```
RMAN> RUN
{
  SET UNTIL SCN <standby_became_primary_scn + 1>;
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

ユーザー管理リカバリでは、先にデータベースを手動でリストアできます。通常、フェイルオーバー前の、数時間を要したバックアップは古くなっています。そのため、障害が発生したプライマリは次のコマンドを使用してリカバリできます。

```
SQL> RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CHANGE -
> <standby_became_primary_scn + 1>;
```

フラッシュバック・データベースを使用する再インスタンス化とは異なり、この手順ではstandby_became_primary_scnに1を加算します。データファイルの場合、SCNまでフラッシュバックすることは、そのSCNに1を加算するまでリカバリすることと同義です。

3. 古いプライマリ・データベースで次のステップを実行します。

a. 古いプライマリ・データベースで次の文を発行します。

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

b. データベースを停止して、再開します。

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

4. 次のコマンドを発行します。

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

このステップの目的は、ディクショナリ・チェックを使用して制御ファイルをデータベースと同期させることです。このコマンドの後に、ディクショナリ・チェックで示されたアクションがないかアラート・ログをチェックします。通常、フェイルオーバー中に古いプライマリがデータファイルを追加または削除している途中でなければ、ユーザー・アクションは必要ありません。

5. Oracle Active Data Guardオプション用にライセンスを購入しているときに、フィジカル・スタンバイ・データベースをアクティブ問合せモードで使用する場合は、このステップをスキップします。それ以外の場合は、スタンバイ・データベースをMOUNT状態にします。

次に例を示します。

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

6. 新しいスタンバイ・データベースの作成前に、新しいプライマリ・データベースは、リモート宛先へのREDOの転送を停止しているはずです。REDO転送サービスを再開するには、新しいプライマリ・データベースで次のステップを実行します。

a. 次の問合せを発行し、アーカイブ宛先の現在の状態を調べます。

```
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, -  
> ERROR, SRL FROM V$ARCHIVE_DEST_STATUS;
```

b. 必要に応じて、宛先を使用可能にします。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
```

c. スタンバイ・データベースが新しいプライマリ・データベースからのREDOデータの受信を確実に開始するよう、ログ・スイッチを実行し、これが成功したことを確認します。

ノート:



これは、新しいプライマリに続いて、古いプライマリが新しいスタンバイになるために重要なステップです。このステップが実行されないと、古いプライマリは誤ったデータベース・ブランチにリカバリされます。問題を修正する唯一の方法は、古いプライマリを再度変換することです。

SQLプロンプトで次の文を入力します。

```
SQL> ALTER SYSTEM SWITCH LOGFILE;  
SQL> SELECT DEST_ID, DEST_NAME, STATUS, PROTECTION_MODE, DESTINATION, -  
> ERROR, SRL FROM V$ARCHIVE_DEST_STATUS;
```

新しいスタンバイ・データベースで、REDO転送サービスがREDOデータを別のデータベースに転送しないよう、LOG_ARCHIVE_DEST_1の初期化パラメータも変更する必要があります。1つのサーバー・パラメータ・ファイル (SPFILE)でプライマリおよびスタンバイ・データベース・ロールの両方がVALID_FOR属性で設定されている場合、このステップを実行する必要はありません。これにより、Oracle Data Guard構成は、ロールの推移後も正しく動作します。

7. 次のようにして、新しいフィジカル・スタンバイ・データベースでREDO Applyを開始します。

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

障害が発生したプライマリ・データベースがリストアされてスタンバイ・ロールで稼働し始めた後、オプションでスイッチオーバーを実行し、それぞれのデータベースを元の(障害発生前の)ロールに推移できます。詳細は、「[フィジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)」を参照してください。

15.8.2 RMANバックアップを使用した障害が発生したプライマリのロジカル・スタンバイへの変換

このステップでは、RMANバックアップを使用して障害が発生したプライマリをロジカル・スタンバイに変換する方法について説明します。

1. 新しいプライマリ・データベースで次の問合せを発行し、障害が発生したプライマリ・データベースをリカバリするSCNを判別します。

```
SQL> SELECT APPLIED_SCN RECOVERY_SCN FROM V$LOGSTDBY_PROGRESS;
```

また、新しいプライマリ・データベースで、次のようにして、アーカイブ・ログの処理に使用するSCNを判別します。

- a. すべてのスタンバイREDOログを必ずアーカイブしておきます。次の問合せを発行し、戻されるNONEの値を検索します。データベースのサイズおよびアーカイブする必要があるログの数によって、NONEのステータスが戻されるまでしばらく時間がかかることがあります。

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

- b. NONEのステータスが戻されたら、次の問合せを実行し、このリカバリの一環としてアーカイブ・ログの処理のためのSCNを取得します。

```
SQL> SELECT VALUE ARCHIVE_SCN FROM SYSTEM.LOGSTDBY$PARAMETERS -  
> WHERE NAME=' STANDBY_BECAME_PRIMARY_SCN' ;
```

2. フェイルオーバー操作以後に作成されたアーカイブ・ログをすべて、障害が発生したプライマリ・データベースから削除します。障害が発生したプライマリ・データベースがスタンバイから分離されていた場合は、現行のプライマリ・データベースと一貫性がない不一致のアーカイブ・ログが存在する可能性があります。これらの不一致のアーカイブ・ログを適用しないようにするには、バックアップおよび高速リカバリ領域から削除する必要があります。次のRMANコマンドを使用すると、関連アーカイブ・ログを高速リカバリ領域から削除できます。

```
RMAN> DELETE FORCE ARCHIVELOG FROM SCN ARCHIVE_SCN;
```

削除されると、これらの不一致のログおよび後続のトランザクションはリカバリできなくなります。

3. 新しいプライマリ・データベースで、次の問合せを発行し、バックアップからリカバリする前に、障害が発生したプライマリ・データベースにコピーする必要があるログ・ファイルの最小セットを判別します。

```
SQL> SELECT file_name FROM DBA_LOGSTDBY_LOG WHERE next_change# > ARCHIVE_SCN;
```

必要なスタンバイ・ログを取得し、バックアップ・セットを新しいスタンバイにコピーして新しい高速フラッシュ・リカバリ領域にリストアします。これらのログはスタンバイREDOログから得られるため、スタンバイの標準アーカイブの一部ではありません。RMANユーティリティでは、部分的なファイル名を使用して正しい場所からファイルを取得できます。

次に、RMANのBACKUPコマンドの使用例を示します。

```
RMAN> BACKUP AS COPY DEVICE TYPE DISK FORMAT '/tmp/test/%U'  
> ARCHIVELOG LIKE '<partial file names from above>%';
```

次に、RMANのRESTOREコマンドの使用例を示します。

```
RMAN> CATALOG START WITH '/tmp/test';  
RMAN> RESTORE ARCHIVELOG FROM SEQUENCE 33 UNTIL SEQUENCE 35;
```

4. 元のプライマリの全データファイルのバックアップをリストアし、RECOVERY_SCN + 1までリカバリします。現行の制御ファイルを利用することをお勧めします。

- a. データベースを制限モードで起動して、データベースのオープン後にGUARD ALLコマンドが発行されるまでログ・トランザクションから保護します。
- b. バックアップを使用して障害が発生したプライマリ・データベースのデータファイルをリストアします。
- c. フラッシュバック・データベースが有効な場合は、無効にします(USING BACKUP CONTROLFILE句に必要)。
- d. SQL*Plusで、RECOVERY_SCN +1までPoint-in-Timeリカバリを実行します。

現行の制御ファイルを使用してもバックアップ制御ファイルを使用しても、USING BACKUP CONTROLFILE句を指定してリストアされるアーカイブ・ログを指すことができるようにする必要があります。指定しないと、リカバリ・プロセスでは、ステップ3で取得したログではなく、オンラインREDOログへのアクセスが試行されることがあります。ステップ3で取得した順序の入力を求められた場合は、次のようにリストアされるアーカイブ・ログのコピーのファイル名を指定します。

```
SQL> RECOVER DATABASE UNTIL CHANGE RECOVERY_SCN + 1 USING BACKUP CONTROLFILE;
```

5. RESETLOGSオプションを指定してデータベースをオープンします。

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

6. Database Guardの有効化

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

7. 新しいプライマリ・データベースへのデータベース・リンクを作成してSQL Applyを開始します。

```
SQL> CREATE PUBLIC DATABASE LINK myLink -  
> CONNECT TO SYSTEM IDENTIFIED BY password -  
> USING 'service name of new primary database';  
  
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY myLink;
```

この時点で、制限セッション(ALTER SYSTEM DISABLE RESTRICTED SESSION)を無効にできます。つまり、データベースを再起動してステップ4cのフラッシュバックを再度有効にする必要がある場合は、RESTRICTED SESSIONをオフにして再起動します。

15.9 フィジカル・スタンバイの再作成を伴わないプライマリの文字セットの変更

Oracle Data Guardでは、構成内のフィジカル・スタンバイ・データベースを再作成することなく、プライマリ・データベースのデータベース文字セットと各国語文字セットの両方を変更できます。

プライマリ・データベースの文字セット変換の実行中に、中断を最小限に抑えて、フィジカル・スタンバイ・データベースを使用し続けることができます。

文字セット移行プロセスは、考えられる問題の調査や問題を解決する方法の特定など、準備ステップで構成されます。これらの準備ステップの実行中、Oracle Data Guard構成は変わらず動作でき、フィジカル・スタンバイを維持するために何ら追加のステップは必要ありません。準備ステップの完了後、実際の変換が実行されます。これにはシステム・データ(メタデータ)およびユーザー・データの両方への変更が含まれる可能性があります。Oracle Data Guardに特有のいくつかの手順を、変換の一部として実行する必要があります。これらを実行する手順には、Database Migration Assistant for Unicode (DMU) やその他の適切な文字セットの変換ツールにより実行されるステップが組み入れられています。

このプロセスに必要なステップの詳細は、<http://support.oracle.com>にあるMy Oracle Support ノート1124165.1を参照してください。

15.10 プライマリでのPDB PITRまたはPDBフラッシュバック後にスタンバイで必要なアクション

プライマリでPDB PITRまたはPDBフラッシュバックを実行した後、スタンバイ上のPDBまたはフラッシュバックをリストアしてプライマリをスタンバイに継がせることができます。

Oracle Databaseリリース19cでは、スタンバイがマウント・モードで、かつフラッシュバック・データベースが有効になっている場合、何も実行する必要はありません。一般的なケースでは、十分なフラッシュバック・データが存在する場合は、引き続きスタンバイがプライマリを継ぐように、スタンバイは自動的にフラッシュバックされます。

他のすべての場合、プライマリでPDB PITRまたはPDBフラッシュバックを実行し、操作の開始に関するREDOが初めて検出されると、スタンバイでのMRPはエラーORA-39874に続いて補助エラーORA-39873で終了します。次に、アラート・ログに出力されるメッセージの例を示します。

```
Recovery of pluggable database PDB1 aborted due to pluggable database open
resetlog marker.
To continue recovery, restore all data files for this PDB to
checkpoint SCN lower than 1437261, or timestamp before 11/15/2012 16:38:49,
and restart recovery
MRP0: Background Media Recovery terminated with error 39874

ORA-39874: Pluggable Database PDB1 recovery halted
ORA-39873: Restore all data files to a checkpoint SCN lower than 1437261.
```

Oracle Databaseリリース19cでは、スタンバイがオープンし、フラッシュバック・データベースが有効になっている場合は、スタンバイをマウント・モードにして、スタンバイ・リカバリを開始できます。プライマリをスタンバイに継がせるための試みとして、MRPにより、スタンバイの自動フラッシュバックがトリガーされます。これが成功すると、アラート・ログにレポートされ、後はActive Data Guardインスタンスを開くことのみが必要です。自動フラッシュバックがトリガーされない場合、または自動フラッシュバックによってプライマリをスタンバイが継がなかった場合は、この項で説明する手動によるステップを実行できます。

スタンバイでメディア・リカバリをこれ以上続ける前に、そのPDBのすべてのデータファイルをリストアする必要があります。それは、次の2つの方法で行うことができます。

- スタンバイでフラッシュバックが有効化されている場合は、スタンバイでPDBフラッシュバックを使用してから、スタンバイ管理リカバリを再開できます。下の「フラッシュバックが有効化されている場合のリカバリの実行」を参照してください。
- スタンバイでフラッシュバックが有効化されていない場合、プライマリでPDBがリカバリされた時点より前に作成したバックアップから、リカバリを実行できます。下の「フラッシュバックが有効化されていない場合のリカバリの実行」を参照してください。

どちらの方法でも、プライマリでPDB PITRまたはフラッシュバックを実施したPDBは、残りのスタンバイに追いつくまでスタンバイで開くことができません。

フラッシュバックが有効化されている場合のリカバリの実行

スタンバイでフラッシュバックが有効化されている場合は、スタンバイでPDBをフラッシュバックしてから、スタンバイ管理リカバリを再開できます。

1. 影響を受けるPDBおよびPITR SCNを特定します。

リカバリが停止したPDBの名前はORA-39874メッセージに、PITR SCNはORA-39873メッセージに表示されます。

2. スタンバイ・データベースがまだ開いている場合は、それを閉じます。

```
SQL> ALTER DATABASE CLOSE;
```

3. スタンバイでプラグブル・データベースをフラッシュバックします。

```
SQL> FLASHBACK PLUGGABLE DATABASE pdb1 TO SCN 1437260;
```

TO SCNとUNTIL SCNはセマンティックが異なるため、FLASHBACK PLUGGABLE DATABASEコマンドのSCNは、前の例のように1437261ではなく1437260です。

4. スタンバイでメディア・リカバリを再開します。

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

フラッシュバックが有効化されていない場合のリカバリの実行

1. 影響を受けるPDBおよびPITR SCNを特定します。

リカバリが停止したPDBの名前はORA-39874メッセージに、PITR SCNはORA-39873メッセージに表示されます。

2. スタンバイ・データベースがまだ開いている場合は、それを閉じます。

```
SQL> ALTER DATABASE CLOSE;
```

3. PDBデータファイルをリストアします。

```
RMAN> RESTORE PLUGGABLE DATABASE pdb1 UNTIL SCN 1437261;
```

UNTIL SCN構文により、RMANはリストア元となる適切なバックアップを自動的に選択できます。スタンバイでデータ・ファイルをリストアした後、MRPを再開してREDOログの適用を継続します。

4. スタンバイでメディア・リカバリを再開します。

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

関連項目:

- PDBのPoint-in-Timeリカバリの実行の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド』を参照してください。

第II部 リファレンス

次の各項では、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。さらに詳しい資料は、Oracle Database 10のドキュメント・セットを参照してください。

- [初期化パラメータ](#)
- [LOG_ARCHIVE_DEST_nパラメータの属性](#)
- [Oracle Data Guardに関連するSQL文](#)
- [Oracle Data Guardに関連するビュー](#)

16 初期化パラメータ

これは、Data Guard環境で設定される初期化パラメータをアルファベット順に並べたリストです。パラメータがプライマリ・ロールとスタンバイ・ロールのどちらに対して設定されるかを示し、各パラメータの設定に関するヒントも提供します。

『[Oracle Databaseリファレンス](#)』には、ALTER SYSTEM SET文の発行(たとえばALTER SYSTEM SET LOG_ARCHIVE_TRACEなど)、または初期化パラメータ・ファイルの編集による初期化パラメータの更新方法を含む、完全な初期化パラメータ情報が記載されています。初期化パラメータの設定方法の詳細は、オペレーティング・システム固有のOracleマニュアルを参照してください。

表16-1 Oracle Data Guard構成内のインスタンスの初期化パラメータ

パラメータ	適用可能対象	ノートおよび推奨事項
COMPATIBLE = release_number	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	スイッチオーバーを実行すると思われる場合は、同じ値をプライマリ・データベースおよびスタンバイ・データベースに指定します。値が異なる場合、REDO 転送サービスで REDO データをプライマリ・データベースからスタンバイ・データベースに転送できない場合があります。例については、スタンバイ・データベース用のパラメータ・ファイルの作成を参照してください。 ロジカル・スタンバイ・データベースは、スイッチオーバーがないと思われる場合、プライマリ・データベースより COMPATIBLE の設定が大きくなる可能性があります。 SQL Apply を使用したローリング・アップグレードの場合は、このパラメータを「ロジカル・スタンバイ・データベースの新規作成によるローリング・アップグレードの実行」で説明したガイドラインに従って設定します。
CONTROL_FILE_RECORD_KEE P_TIME = number_of_days	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。このパラメータを使用して、指定された日数(0 から 365)内に、制御ファイル内(アーカイブ REDO ログ・ファイルなどの必要な情報が含まれている)の再使用可能レコードが上書きされないようにします。
CONTROL_FILES = ' control_file_name', ' control_file_name' ''	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	必須。1 つ以上の制御ファイルのパス名とファイル名を指定します。制御ファイルはすでにデータベースに存在している必要があります。2 つの制御ファイルを使用することをお勧めします。現行の制御ファイルの別のコピーが存在していれば、問題のない制御ファイルを問題のある制御ファイルの場所にコピーした後、容易にインスタンスを再開できます。例については、スタンバイ・データベース用のパラメータ・フ

パラメータ	適用可能対象	ノートおよび推奨事項
	イ	イルの作成 を参照してください。
DB_FILE_NAME_CONVERT = 'location_of_primary_databases_datafile', 'location_of_standby_database_datafile'	フィジカル・スタンバイ スナップショット・スタンバイ	このパラメータには、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。
DB_UNIQUE_NAME = データベースの一意の名前	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	LOG_ARCHIVE_CONFIG パラメータの指定時には必須のもので、それ以外の場合も推奨されます。このデータベースの一意の名前を指定します。この名前はプライマリ・データベースとスタンバイ・データベースのロールが逆になっても変更されません。DB_UNIQUE_NAME パラメータは、DB_NAME パラメータ値にデフォルト設定されます。
FAL_CLIENT = Oracle_Net_service_name	フィジカル・スタンバイ スナップショット・スタンバイ	このパラメータは必要なくなりました。設定されていない場合、フェッチ・アーカイブ・ログ (FAL) サーバーは、クライアントの DB_UNIQUE_NAME に対応する LOG_ARCHIVE_DEST_n パラメータからクライアントのネットワーク・アドレスを取得します。
FAL_SERVER = Oracle_Net_service_name	フィジカル・スタンバイ スナップショット・スタンバイ	欠落しているアーカイブ REDO ログ・ファイルをこのスタンバイ・データベースがフェッチ (要求) できるフェッチ元データベースの Oracle Net サービス名を 1 つ以上指定します。
INSTANCE_NAME	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。このパラメータが定義されており、プライマリ・データベースとスタンバイ・データベースが同じホストに存在する場合は、スタンバイ・データベースに対してプライマリ・データベースとは異なる名前を指定します。例については、 スタンバイ・データベース用のパラメータ・ファイルの作成 を参照してください。
LOG_ARCHIVE_CONFIG ='DG_CONFIG=(db_unique_name, db_unique_name ...)'	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ	お薦めします。Oracle Data Guard の機能を完全に有効にするには、このパラメータの DG_CONFIG 属性を Oracle Data Guard 構成のデータベースごとに明示的に設定する必要があります。構成内の各データベース名の DB_UNIQUE_NAME を含み、各名前がカンマで区切られたテ

パラメータ	適用可能対象	ノートおよび推奨事項
	スナップショット・スタンバイ	キスト文字列に DG_CONFIG を設定します。
LOG_ARCHIVE_DEST_n = {LOCATION=path_name SERVICE=service_name, attribute, attribute }	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	必須。最大 30(ここで n = 1, 2, 3, ...31)の宛先を定義します。それぞれが LOCATION または SERVICE 属性を指定する必要があります。各 LOG_ARCHIVE_DEST_n パラメータに対し、それぞれ対応する LOG_ARCHIVE_DEST_STATE_n パラメータを指定します。
LOG_ARCHIVE_DEST_STATE_n = {ENABLE DEFER ALTERNATE}	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	必須。LOG_ARCHIVE_DEST_STATE_n パラメータを指定して、指定された(または代替の)宛先への REDO 転送サービスによる REDO データの転送を有効または無効に設定します。各 LOG_ARCHIVE_DEST_n パラメータに対し、LOG_ARCHIVE_DEST_STATE_n パラメータを定義します。 「LOG_ARCHIVE_DEST_n パラメータの属性」 も参照してください。
LOG_ARCHIVE_FORMAT =log %d_%t_%s_%r.arc	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	LOG_ARCHIVE_FORMAT および LOG_ARCHIVE_DEST_n パラメータが連結され、データベースに完全修飾されたアーカイブ REDO ログ・ファイル名が生成されます。
LOG_ARCHIVE_MAX_PROCESSES =integer	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。Oracle ソフトウェアが最初に起動するアーカイバ(ARCn)プロセスの数(1 から 30)を指定します。デフォルト値は 4 です。
LOG_ARCHIVE_MIN_SUCCEED_DEST	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ	オプション。このパラメータは、ログ・ファイル・グループを再使用する前にアーカイブする必要のあるローカルまたはリモートの必須の宛先、またはローカルのオプションの宛先の数指定します。

パラメータ	適用可能対象	ノートおよび推奨事項
	スナップショット・スタンバイ	
LOG_ARCHIVE_TRACE =integer	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。スタンバイ・サイトへの REDO データの転送をトレースするには、このパラメータを設定します。有効な整数値については、 「アーカイブ・トレースの設定」 に説明されています。
LOG_FILE_NAME_CONVERT = 'location_of_primary_database_redo_logs', 'location_of_standby_database_redo_logs'	ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	このパラメータは、プライマリ・データベースのオンライン REDO ログ・ファイルのパス名をスタンバイ・データベースのパス名に変換します。例については、 「スタンバイ・データベース用のパラメータ・ファイルの作成」 を参照してください。
REMOTE_LOGIN_PASSWORDFILE = {EXCLUSIVE SHARED}	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オペレーティング・システムの認証を管理ユーザーに使用する場合および SSL を REDO 転送の認証に使用する場合はオプション。それ以外の場合は、Oracle Data Guard 構成内の各データベースに対して、EXCLUSIVE または SHARED に設定する必要があります。
SHARED_POOL_SIZE = bytes	プライマリ ロジカル・スタンバイ フィジカル・スタンバイ スナップショット・スタンバイ	オプション。オンライン REDO ログ・ファイルから読み込んだ情報を処理する System Global Area(SGA)の指定に使用します。使用可能な SGA が大きいと、処理できる情報量も大きくなります。
STANDBY_FILE_MANAGEMENT = {AUTO MANUAL}	プライマリ フィジカル・スタンバイ スナップショット・スタンバイ	STANDBY_FILE_MANAGEMENT パラメータを AUTO に設定し、データファイルをプライマリ・データベースに追加、またはプライマリ・データベースから削除すると、手動操作なしで該当する変更内容がスタンバイ・データベースにも反映されるようにします。プライマリ・データベースとスタンバイ・データベースのディレクトリ構造が異なる場合は、 DB_FILE_NAME_CONVERT 初期化パラメータも設定して、

パラメータ	適用可能対象	ノートおよび推奨事項
		プライマリ・データベースのデータファイルの 1 つ以上のセットのファイル名を(フィジカル)スタンバイ・データベースのファイル名に変換する必要があります。

17 LOG_ARCHIVE_DEST_nパラメータの属性

これは、LOG_ARCHIVE_DEST_n初期化パラメータの属性のリストです(*n*は1から31の間の整数)。

- [AFFIRM](#)および[NOAFFIRM](#)
- [ALTERNATE](#)
- [COMPRESSION](#)
- [DB_UNIQUE_NAME](#)
- [DELAY](#)
- [ENCRYPTION](#)
- [GROUP](#)
- [LOCATION](#)および[SERVICE](#)(LOCATIONはLOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31ではサポートされない)
- [MANDATORY](#)(LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31ではサポートされない)
- [MAX_FAILURE](#)
- [NET_TIMEOUT](#)
- [NOREGISTER](#)
- [PRIORITY](#)
- [REOPEN](#)
- [SYNC](#)および[ASYNC](#)(SYNCはLOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31ではサポートされない)
- [TEMPLATE](#)
- [VALID_FOR](#)

使用上のノート

- Oracle Data Guard構成の各データベースは、通常、オンラインおよびスタンバイREDOログのアーカイブ用のLOCATION属性を持つ宛先を1つと、構成にあるその他のデータベース用のREMOTE属性を持つ宛先を1つ持ちます。
- 構成されている場合、LOG_ARCHIVE_DEST_1からLOG_ARCHIVE_DEST_10の各宛先には、ローカル・ディスクのディレクトリまたはリモートからアクセスするデータベースを指定する、LOCATIONまたはSERVICE属性が含まれている必要があります。LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31の各宛先にはSERVICE属性が含まれている必要があります。その他のすべての属性は省略可能です。
- LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31は、COMPATIBLE初期化パラメータが11.2.0.0以上に設定されている場合のみ使用できます。

ノート:



LOG_ARCHIVE_DEST_n 初期化パラメータの複数の属性が非推奨になりました。これらの属性は下位互換用にサポートされており、『[Oracle Database リファレンス](#)』で説明しています。

関連項目:

LOG_ARCHIVE_DEST_n宛先を定義してREDO転送サービスを設定する方法の詳細は、[REDO転送サービス](#)を参照してください

17.1 AFFIRMおよびNOAFFIRM

AFFIRMおよびNOAFFIRM属性は、REDO転送先が、受信済REDOデータをスタンバイREDOログに書き込む前または書き込んだ後に、受信済REDOデータを確認するかどうかを制御します。

各オプションの定義は次のとおりです。

- AFFIRM: REDO転送先で受信したREDOデータをスタンバイREDOログに書き込んだ後に確認するように指定します。
- NOAFFIRM: REDO転送先で受信したREDOデータをスタンバイREDOログに書き込む前に確認するように指定します。

カテゴリ	AFFIRM	NOAFFIRM
データ型	キーワード	キーワード
有効な値	該当なし	該当なし
デフォルト値	該当なし	該当なし
必須属性	SERVICE	SERVICE
属性との競合	NOAFFIRM	AFFIRM
対応先	V\$ARCHIVE_DEST ビューの AFFIRM 列	V\$ARCHIVE_DEST ビューの AFFIRM 列

使用上のノート

- AFFIRM属性およびNOAFFIRM属性が指定されていない場合、デフォルトは、SYNC属性が指定されているときはAFFIRM、ASYN属性が指定されているときはNOAFFIRMです。
- SYNC属性を使用しないAFFIRM属性の指定は非推奨であり、今後のリリースではサポートされません。

関連項目:



[SYNC および ASYN 属性](#)

例

次の例は、リモート宛先のAFFIRM属性を示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 SYNC AFFIRM'  
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

17.2 ALTERNATE

ALTERNATE属性は、元のアーカイブ先で障害が発生したときに使用する代替アーカイブ先を指定します。

ノート:



Oracle Database 12c リリース 2 (12.2.0.1)以降、代替アーカイブ先の数と機能を、リモート・ログ・アーカイブ先の ALTERNATE 属性のかわりに GROUP および PRIORITY 属性を使用することで拡張できます。この新しい方法は ALTERNATE 属性の方法とともに使用できません。詳細は、[代替宛先](#)を参照してください。

ALTERNATE 属性は、代替ローカル・アーカイブ先を構成するために予約されています。下位互換性のために、ALTERNATE をリモート・ログ・アーカイブに使用する例が [ALTERNATE 属性を使用したリモート代替宛先の構成](#)に提供されています。

カテゴリ	ALTERNATE=LOG_ARCHIVE_DEST_n
データ型	文字列
有効な値	LOG_ARCHIVE_DEST_n 宛先。n は 1 から 10 の値です。
デフォルト値	なし。代替宛先が指定されていない場合、REDO 転送サービスでは自動的に別のアーカイブ先に変更されません。
必須属性	なし 脚注 1
属性との競合	GROUP および PRIORITY 脚注 2
対応先	V\$ARCHIVE_DEST ビューの ALTERNATE および STATUS 列

脚注1

MAX_FAILUREがALTERNATEで使用されることは必須ではありませんが、代替宛先がアクティブになるには、ゼロ以外のMAX_FAILURE値が必要です。ALTERNATEをMAX_FAILUREのデフォルト値(ゼロ)とともに使用すると、動作は変わりません。

脚注2

REOPEN属性にゼロ以外の値を指定すると、試行間の最小時間がREOPENの値(秒)と等しくなり、エラーの回数がMAX_FAILUREの値と等しいか超えるまで代替宛先はアクティブになりません。

使用上のノート

- ALTERNATE属性はオプションです。代替宛先が指定されていない場合、元のアーカイブ先に障害が発生しても、アーカイブ・サービスでは自動的に別のアーカイブ先に変更されません。
- ローカルのLOG_ARCHIVE_DEST_nパラメータごとに指定できる代替宛先は、1つのみです(LOCATION=…)。
- 代替宛先には、次の例に示されているように同じローカル・プライマリ・データベース・システムまたはローカル・スタンバイ・データベース・システム上の異なるディスクの場所を指定する必要があります。

- 代替宛先を構成するには、LOG_ARCHIVE_DEST_STATE_nパラメータをALTERNATEに設定します。
- 代替宛先を手動で有効化するには、LOG_ARCHIVE_DEST_STATE_nパラメータをENABLEに設定します。
- 有効化された宛先が代替宛先を参照していない場合は、代替宛先を自動的に有効化する方法はないため、代替宛先は保留になることを意味しています。ただし、ALTER SYSTEM文を使用して実行時に代替宛先を有効化できます。
- 代替宛先に指定できる宛先には、次の制限事項があります。
 - ローカル宛先を少なくとも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はできない。
- 宛先に障害がある場合、次のアーカイブ操作で代替宛先が有効化されます。アーカイブ操作中に代替宛先を有効化するには処理済のブロックを再度読み取る必要があるため、これはサポートされていません。
- 代替宛先が指定されていない場合、またはMAX_FAILUREがゼロ(デフォルト)に設定されている場合、元の宛先に障害が発生しても、アーカイブ・サービスでは自動的に別の宛先に変更されません。

例

これらの例は、基本的な概念を示しており、示しているように使用することを意図しているわけではありません。ローカルのアーカイブ設定によって構成は異なります。

次の例に、サンプル初期化パラメータ・ファイルを示します。このサンプル・ファイルでは、書込み失敗や割当て失敗などのエラーが発生したり、デバイスがいっぱいになった場合、次のアーカイブ操作で、ローカル・アーカイブ先LOG_ARCHIVE_DEST_1は代替宛先LOG_ARCHIVE_DEST_2に自動的にフェイルオーバーします。

例17-1 代替ローカル宛先への自動フェイルオーバー

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY MAX_FAILURE=1
ALTERNATE=LOG_ARCHIVE_DEST_2'

LOG_ARCHIVE_DEST_STATE_1=ENABLE

LOG_ARCHIVE_DEST_STATE_2=ALTERNATE

LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'
```

元の宛先LOG_ARCHIVE_DEST_1へのアーカイブを再開するには、手動で再度有効にする必要があります。次に、LOG_ARCHIVE_DEST_2を前の代替状態にリセットしてアクティブなローカル・アーカイブ先が2つにならないようにする必要があります。そのためには、次のようにしてLOG_ARCHIVE_DEST_STATE_nパラメータを元の値に設定します。

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE

ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

このフォールバック・メカニズムは自動化できます。[例17-2](#)のサンプル初期化パラメータ・ファイルで示すように、優先宛先を指すALTERNATE属性を指定して、元の宛先と代替宛先のペアを作成します。

例17-2 自動ローカル代替フォールバック

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY MAX_FAILURE=1
ALTERNATE=LOG_ARCHIVE_DEST_2'

LOG_ARCHIVE_DEST_STATE_1=ENABLE

LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

```
LOG_ARCHIVE_DEST_2=' LOCATION=/disk2 MANDATORY  
ALTERNATE=LOG_ARCHIVE_DEST_1'
```

LOG_ARCHIVE_DEST_1が再度使用できるようになると、Oracle Data Guardはアクティブなローカル・アーカイブ宛先になるように自動的に設定し、LOG_ARCHIVE_DEST_2をその代替としてリセットします。

17.3 COMPRESSION

COMPRESSION属性は、REDOデータをREDO転送先に転送する前に圧縮するかどうかを指定するために使用します。

ノート:



REDO 転送の圧縮は、Oracle Advanced Compression オプションの機能です。REDO 転送の圧縮機能を使用する前に、このオプションのライセンスを購入する必要があります。

カテゴリ	COMPRESSION=[ENABLE DISABLE ZLIB LZO]
データ型	ブール
有効な値	ENABLE、DISABLE、ZLIB または LZO
デフォルト値	DISABLE
必須属性	なし
属性との競合	なし
対応先	V\$ARCHIVE_DEST ビューの COMPRESSION 列

使用上のノート

- ENABLEオプションを指定すると圧縮が可能になり、デフォルトでZLIB圧縮アルゴリズムが使用されます。
- COMPRESSION属性はオプションです。指定されていない場合、デフォルトの圧縮動作はDISABLEです。
- Oracle Data GuardのCOMPRESSION属性も使用するSYNC接続文字列の場合、sqlnet.oraファイルでSQLNET.COMPRESSION構成パラメータを無効に設定します(オフにします)。SQLNET.COMPRESSIONパラメータの詳細は、[『Oracle Database Net Servicesリファレンス』](#)を参照してください。

例

次の例は、COMPRESSION属性が指定されているLOG_ARCHIVE_DEST_nパラメータを示しています。ENABLEオプションが指定されているため、ZLIB圧縮アルゴリズムが使用されます。

```
LOG_ARCHIVE_DEST_3=' SERVICE=denvier SYNC COMPRESSION=ENABLE'  
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```


17.4 DB_UNIQUE_NAME

DB_UNIQUE_NAME属性は、この宛先にあるデータベースの一意の名前を指定します。

カテゴリ	DB_UNIQUE_NAME=name
データ型	String
有効な値	name は、DB_UNIQUE_NAME パラメータを使用してこのデータベースに定義された値と一致している必要があります。
デフォルト値	なし
必須属性	なし
属性との競合	なし
対応先	V\$ARCHIVE_DEST ビューの DB_UNIQUE_NAME 列

使用上のノート

- この属性は、次の場合にはオプションです。
 - LOG_ARCHIVE_CONFIG=DG_CONFIG初期化パラメータが指定されていない場合。
 - これがローカル宛先(LOCATION属性で指定)の場合。
- この属性が必須となるのは、LOG_ARCHIVE_CONFIG=DG_CONFIG初期化パラメータが指定されており、かつ、これがリモート宛先(SERVICE属性で指定)の場合です。
- プライマリ・データベースとスタンバイ・データベースの関係を明確に識別するには、DB_UNIQUE_NAME属性を使用します。この属性は、Oracle Data Guard構成に複数のスタンバイ・データベースが存在する場合に特に有用です。
- DB_UNIQUE_NAMEで指定する名前は、DG_CONFIGリストのDB_UNIQUE_NAME値の1つと一致している必要があります。REDO転送サービスにより、指定された宛先のデータベースのDB_UNIQUE_NAME属性がDB_UNIQUE_NAME属性と一致していること、またはその宛先への接続が拒否されていることが検証されます。
- DB_UNIQUE_NAME属性で指定する名前は、宛先で識別されるデータベースのDB_UNIQUE_NAME初期化パラメータで指定されている名前と一致している必要があります。

例

次の例では、DB_UNIQUE_NAMEパラメータではboston (DB_UNIQUE_NAME=boston)が指定されており、これはLOG_ARCHIVE_DEST_1パラメータのDB_UNIQUE_NAME属性にも指定されています。LOG_ARCHIVE_DEST_2パラメータのDB_UNIQUE_NAME属性は、宛先としてchicagoを指定しています。bostonとchicagoは、どちらもLOG_ARCHIVE_CONFIG=DG_CONFIGパラメータにリストされています。

```
DB_UNIQUE_NAME=boston
LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, boston, denver) '
LOG_ARCHIVE_DEST_1=' LOCATION=/arch1/
  VALID_FOR=(ALL_LOGFILES, ALL_ROLES)
  DB_UNIQUE_NAME=boston'
LOG_ARCHIVE_DEST_2=' SERVICE=Sales_DR
```

```
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE)
DB_UNIQUE_NAME=chicago'
```

17.5 DELAY

DELAY属性は、スタンバイ・サイトでプライマリ・サイトからのREDOデータがアーカイブされてから、スタンバイ・データベースまたはそこからカスケードされたスタンバイにアーカイブREDOログ・ファイルが適用されるまでの最小タイム・ラグを指定します。

カテゴリ	DELAY[=minutes]
データ型	数値
有効な値	0 分以上
デフォルト値	30 分
必須属性	SERVICE
属性との競合	LOCATION, VALID_FOR=(*,STANDBY_ROLE)
対応先	V\$ARCHIVE_DEST ビューの DELAY_MINS および DESTINATION 列

使用上のノート

- DELAY属性はオプションです。デフォルトでは、遅延は発生しません。
- DELAY属性は、スタンバイ宛先のアーカイブREDOログ・ファイルが、指定した時間が経過するまでリカバリに利用されないことを示します。時間間隔は分で表され、転送されたREDOデータがスタンバイ・サイトに正常に到達した時点から計測されます。
- DELAY属性を使用すると、破損したプライマリ・データまたは誤ったプライマリ・データからスタンバイ・データベースを保護できます。ただし、フェイルオーバー中は、破損が発生した時点までのすべてのREDOを適用するための所要時間が長くなるため、トレードオフを伴います。
- DELAY属性は、REDOデータのスタンバイ宛先への転送に影響しません。
- リアルタイム適用が使用可能になっている場合、設定した遅延は無視されます。
- DELAY属性の変更は、次にREDOデータをアーカイブすると(ログ・スイッチ後に)有効になります。進行中のアーカイブ操作には、影響しません。
- 指定した遅延間隔は、スタンバイ・サイトで次のように上書きできます。

- フィジカル・スタンバイ・データベース用

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

- ロジカル・スタンバイ・データベース用

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

- カスケードされたスタンバイが使用するDELAY値は、カスケードしているスタンバイにREDOを送信したプライマリのLOG_ARCHIVE_DEST_nパラメータに設定された値です。

関連項目:

これらのALTER DATABASE文の詳細は、『[Oracle Database SQL言語リファレンス](#)』を参照してください。

例

DELAY属性を使用すれば、プライマリ・データベースと様々なレベルで同期させる複数のスタンバイ・データベースを維持する構成を設定できます。ただし、REDO Applyで破損時点までのすべてのREDOを適用するまでにかかる時間が長くなるため、フェイルオーバー中は、この保護はなんらかのオーバーヘッドを伴います。

たとえば、プライマリ・データベースAにスタンバイ・データベースBおよびCがあると仮定します。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されているため、タイム・ラグはありません。スタンバイ・データベースCには2時間の遅延が設定されています。2時間あれば、スタンバイ・データベースに伝播する前に、ユーザー・エラーを検出できます。

次の例は、この構成用にDELAY属性を指定する方法を示しています。

```
LOG_ARCHIVE_DEST_1=' LOCATION=/arch/dest MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_2=' SERVICE=stbyB SYNC AFFIRM'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_DEST_3=' SERVICE=stbyC DELAY=120'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

ノート:



または、十分なフラッシュバック・ログ・データがあれば、フラッシュバック・データベースを使用して、データベースを障害発生時点または他のデータベース・インカネーションのSCNまで戻すこともできます。フラッシュバック・データベースの使用については、『[Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド](#)』で説明しています。

17.6 ENCRYPTION

ENCRYPTION属性は、Zero Data Loss Recovery Appliance (リカバリ・アプライアンス)に転送する前にREDOデータを暗号化するかどうかを指定するために使用されます。

ノート:



REDO 転送の暗号化はリカバリ・アプライアンスへの接続でのみ許可されます。リカバリ・アプライアンス以外のログ・アーカイブ先で暗号化を構成しようとするとエラーになります。

カテゴリ	ENCRYPTION=ENABLEまたはDISABLE
データ型	ブール
有効な値	ENABLE または DISABLE

カテゴリ	ENCRYPTION=ENABLEまたはDISABLE
デフォルト値	DISABLE
必須属性	SERVICE
属性との競合	COMPRESSION、SYNC および LOCATION
対応先	V\$ARCHIVE_DEST ビューの ENCRYPTION 列

使用上のノート

- ENCRYPTION属性はオプションです。指定しない場合、デフォルトの暗号化動作はDISABLEです。
- ENCRYPTION属性を使用するには、保護されたデータベースでCOMPATIBLE初期化パラメータを11.2.0.4以上に設定する必要があります。

例

次の例に、LOG_ARCHIVE_DEST_nパラメータに指定したENCRYPTION属性を示します。

```
LOG_ARCHIVE_DEST_3='SERVICE=denver ENCRYPTION=ENABLE'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

関連項目:

- [Zero Data Loss Recovery Appliance管理者ガイド](#)(LOG_ARCHIVE_DEST_nパラメータを使用したREDO暗号化の詳細)
- [Oracle Database Advanced Securityガイド](#)(透過的データ暗号化の詳細)

17.7 GROUP

GROUP属性を使用して、ログのアーカイブ先の特定のコレクションでメンバーシップを指定します。

グループは1から8まで番号付けされます。デフォルトのグループ(GROUP=0)は、割り当てることができないという点において特殊です。デフォルトのグループには、明示的にグループに割り当てられないすべての宛先が移入されます。

カテゴリ	GROUP=integer
データ型	整数
有効な値	1 から 8
デフォルト値	0
必須属性	SERVICE
属性との競合	ALTERNATE
対応先	該当なし

使用上のノート

- なし

例

次の例は、基本的な概念を説明するためのもので、示されたとおりに使用することを意図しているわけではありません。構成に応じて、DB_UNIQUE_NAMEなどの他の必須パラメータが存在する場合があります。

```
LOG_ARCHIVE_DEST_1='SERVICE=FS1 GROUP=1'  
LOG_ARCHIVE_DEST_2='SERVICE=FS2 GROUP=1'  
LOG_ARCHIVE_DEST_3='SERVICE=FS3 GROUP=2'  
LOG_ARCHIVE_DEST_4='SERVICE=FS4 GROUP=2'
```

関連項目:

- [グループへのログ・アーカイブ先の割当て](#)

17.8 LOCATIONおよびSERVICE

各宛先にはLOCATION属性またはSERVICE属性を指定して、REDO転送サービスがローカル・ディスクのディレクトリまたはリモート・データベースの宛先のどちらにREDOデータを転送できるかを明示する必要があります。

LOG_ARCHIVE_DEST_1からLOG_ARCHIVE_DEST_10の宛先は、LOCATION属性またはSERVICE属性を含むことができます。

LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31の宛先は、SERVICE属性しか含むことができません。

カテゴリ	LOCATION=local_disk_directoryまたは USE_DB_RECOVERY_FILE_DEST	SERVICE=net_service_name
データ型	文字列値	文字列値
有効な値	該当なし	該当なし
デフォルト値	なし	なし
必須属性	該当なし	該当なし
属性との競合	SERVICE、DELAY、NOREGISTER、SYNC、ASYNC、 NET_TIMEOUT、AFFIRM、NOAFFIRM、 COMPRESSION	LOCATION
対応先	V\$ARCHIVE_DEST ビューの DESTINATION および TARGET 列	V\$ARCHIVE_DEST ビューの DESTINATION お よび TARGET 列

使用上のノート

- LOCATIONまたはSERVICE属性を指定する必要があります。デフォルトはありません。
- LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31パラメータでは、LOCATION属性はサポートされません。

- 複数の属性を指定している場合は、属性のリストの最初にLOCATION属性またはSERVICE属性を指定します。
- LOCATION属性で少なくとも1つのローカル・ディスクのディレクトリを指定する必要があります。これにより、データベースのメディア・リカバリが必要になる場合、ローカルのアーカイブREDOログ・ファイルに確実にアクセスできるようになります。ローカルまたはリモートの追加宛先は最大30個まで指定できます。
- LOCATION属性には、次のいずれかを指定できます。
 - LOCATION= local_disk_directory
これは、データベースを置くシステム上のディスク・ディレクトリの一意のディレクトリ・パス名を指定します。これは、アーカイブREDOログ・ファイルのローカル宛先です。
 - LOCATION=USE_DB_RECOVERY_FILE_DEST
高速リカバリ領域を構成するには、DB_RECOVERY_FILE_DEST初期化パラメータを使用して、高速リカバリ領域として機能するディレクトリまたはOracle Storage Managerのディスク・グループを指定します。
- SERVICE属性を指定する場合は、次のようにします。
 - SERVICE属性と、REDOデータの送信先となるリモートOracleデータベース・インスタンスを識別する有効なOracle Netサービス名(SERVICE=net_service_name)を指定して、リモート宛先を識別します。

SERVICE属性で指定するOracle Netサービス名は、リモート・データベースへの接続に必要な情報を含む接続記述子に変換されます。

関連項目:

Oracle Netサービス名の設定の詳細は、[『Oracle Database Net Services管理者ガイド』](#)を参照してください。

- リモート宛先にREDOデータを転送するには、着信するアーカイブREDOデータを受け取るために、ネットワーク接続と、リモート宛先に対応付けられたOracleデータベース・インスタンスが必要です。
- LOCATIONおよびSERVICE属性の現行の設定を確認するには、V\$ARCHIVE_DEST固定ビューを問い合わせます。
 - TARGET列は、宛先がプライマリ・データベースにとってローカルかリモートかを示します。
 - DESTINATION列は、宛先に指定されている値を示します。たとえば、宛先パラメータ値は、アーカイブREDOログ・ファイルが配置されているリモートのOracleインスタンスを示すOracle Netサービス名を指定します。

例

次の例は、LOCATION属性の指定方法を示しています。

```
LOG_ARCHIVE_DEST_2='LOCATION=/disk1/oracle/oradata/payroll/arch/'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

次の例は、SERVICE属性の指定方法を示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

17.9 MANDATORY

MANDATORY属性は、書き込まれたオンライン・ログ・ファイルが再使用できるためには、その前に宛先に正常にアーカイブされる必

要があることを指定します。

カテゴリ	MANDATORY
データ型	キーワード
有効な値	該当なし
デフォルト値	該当なし
必須属性	該当なし
属性との競合	オプション
対応先	V\$ARCHIVE_DEST ビューの BINDING 列

使用上のノート

- LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31パラメータでは、MANDATORY属性はサポートされません。
- MANDATORYが指定されていない場合、デフォルトでは、宛先はオプションとみなされます。
すべての宛先がオプションである場合にも、最低1つの宛先は成功する必要があります。オプションの宛先へのアーカイブが失敗した場合、オンラインREDOログ・ファイルは再利用可能なままで、最終的に上書きされる可能性があります。しかし、必須の宛先へのアーカイブ操作が失敗した場合、オンラインREDOログ・ファイルは上書きされません。
- LOG_ARCHIVE_MIN_SUCCEED_DEST=nパラメータ(nは1から10の整数)は、オンラインREDOログ・ファイルを上書きできるようになる前に、正常にアーカイブしておく必要がある宛先の数を指定します。
LOG_ARCHIVE_MIN_SUCCEED_DEST=nの件数は、すべてのMANDATORYの宛先とオプションのローカル宛先によって満たされます。LOG_ARCHIVE_MIN_SUCCEED_DESTパラメータに設定した値が一致すると、オンラインREDOログ・ファイルは再利用可能になります。たとえば、次のようにパラメータを設定できます。

```
# Database must archive to at least two locations before
# overwriting the online redo log files.
LOG_ARCHIVE_MIN_SUCCEED_DEST = 2
```
- 1つ以上のローカル宛先が必要であり、それらにMANDATORYを宣言するか、またはオプションのままにしておくことができます。
LOG_ARCHIVE_MIN_SUCCEED_DESTパラメータの最小値は1のため、最低1つのローカル宛先が操作上必須です。
- 必須の宛先のいずれかに障害が発生すると、LOG_ARCHIVE_MIN_SUCCEED_DESTパラメータは不適切になります。
- LOG_ARCHIVE_MIN_SUCCEED_DESTパラメータ値を、必須の宛先数にオプションのローカル宛先数を加えた数よりも大きく設定することはできません。
- V\$ARCHIVE_DEST固定ビューのBINDING列は、アーカイブ操作に障害がどのように影響するのかを指定します。

例

次の例は、MANDATORY属性を示しています。

```
LOG_ARCHIVE_DEST_1='LOCATION=/arch/dest MANDATORY'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

17.10 MAX_FAILURE

MAX_FAILURE属性は、プライマリ・データベースが失敗した宛先を放棄する前に、REDO転送サービスが通信を再確立してその宛先へのREDOデータを転送するログ・スイッチ時の連続試行回数を制御します。

MAX_FAILURE属性は、Oracle Database 12cリリース1 (12.1)とOracle Database 12cリリース2 (12.2)の間で処理が異なります。この違いを理解しておくことが重要です。詳細は、次の「使用上のノート」を参照してください。

カテゴリ	MAX_FAILURE=count
データ型	数値
有効な値	>=0
デフォルト値	デフォルトのグループ宛先のデフォルト値は 0 です。デフォルト以外のログ・アーカイブ先のグループ宛先のデフォルト値は 1 です。
必須属性	REOPEN
属性との競合	なし
対応先	V\$ARCHIVE_DEST ビューの MAX_FAILURE、FAILURE_COUNT および REOPEN_SECS 列

Oracle Database 12cリリース2 (12.2)でのMAX_FAILUREの使用上のノート

- 新しいGROUP属性とPRIORITY属性を使用するREDO宛先では、エラー件数がMAX_FAILURE属性に指定された値に達した場合、宛先はERROR状態になり、アクセス可能になるまでその状態を継続します。これは、REOPEN属性に指定された値に応じて定期的にチェックされます。
- ログ・アーカイブ・グループのデフォルト宛先(新しいGROUP属性とPRIORITY属性を使用しないそのREDO宛先)では、MAX_FAILURE属性の動作はOracle Database 12cリリース1 (12.1.0.1)の場合と同じです

Oracle Database 12cリリース1 (12.1)でのMAX_FAILUREの使用上のノート

- MAX_FAILURE属性はオプションです。デフォルトでは、失敗した宛先に対するアーカイブの試行回数に制限はありません。
- この属性は、失敗後に回数制限付きでREDOデータの転送を再試行するように、宛先に対する障害の解決方法を指定するときに役立ちます。
- MAX_FAILURE属性を指定した場合は、REOPEN属性も設定する必要があります。指定したログ・スイッチ時の連続試行回数を超過すると、その宛先はREOPEN属性が指定されていないものとして処理されます。
- V\$ARCHIVE_DEST固定ビューのFAILURE_COUNT列で障害件数を確認できます。関連するREOPEN_SECS列は、REOPEN属性値を示します。

ノート:



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ための方法は、LOG_ARCHIVE_DEST_n パラメータを設定するのみです。この結果、障害件数がゼロ(0)にリセットされます。

- ALTER SYSTEM SET文によって宛先が変更されると、障害件数は0(ゼロ)にリセットされます。このため、現在の障害件数の値よりも小さな値にMAX_FAILURE属性を設定する問題が回避されます。
- 障害件数がMAX_FAILURE属性に設定された値以上になると、REOPEN属性値は暗黙的に0(ゼロ)に設定されます。これによって、REDO転送サービスは、次のアーカイブ操作ではREDOデータを代替宛先(ALTERNATE属性で指定)に転送するようになります。
- MAX_FAILURE属性を指定せずに(またはMAX_FAILURE=0を指定し)、REOPEN属性に0(ゼロ)ではない値を指定すると、REDO転送サービスは、失敗した宛先に対して無制限にアーカイブを試行します。宛先がMANDATORY属性を持つ場合、オンラインREDOログ・ファイルはこの宛先にアーカイブされるまで再利用できません。
- 優先代替として構成されていないログ・アーカイブ先では、エラー件数がMAX_FAILURE属性に指定された値に達した場合、宛先は無効化され、その宛先が手動で再度有効化されるまでそれ以上のアクセスはありません。
- 優先代替として構成されているログ・アーカイブ先では、エラー件数がMAX_FAILURE属性に指定された値に達した場合、代替宛先は有効化され、障害が発生した宛先はALTERNATE状態にスイッチされます。この宛先は優先代替のため、(REOPEN属性の値に応じて)定期的にチェックされます。
- 非優先代替として構成されているログ・アーカイブ先では、エラー件数がMAX_FAILURE属性に指定された値に達した場合、宛先は無効化され、その宛先が手動で再度有効化されるまでそれ以上のアクセスはありません。また、以前に優先されていた宛先(現在は使用不可でALTERNATE状態)はALTERNATE状態を継続し、手動で明示的に再度有効化されるまではサービスに戻りません。

例

次の例では、REDO転送サービスが障害の発生した宛先へのログ・スイッチ時に、各ログ・スイッチの間隔が5秒を超えているかぎり最大3回までの再接続を連続再試行できます。3回目の試行後アーカイブ操作に失敗すると、その宛先はREOPEN属性が指定されていないものとして扱われ、リセットされるまで永久障害としてマークされます。

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=5 MAX_FAILURE=3'  
LOG_ARCHIVE_DEST_STATE_1=ENABLE
```

17.11 NET_TIMEOUT

NET_TIMEOUT属性は、REDO転送先に送信されたREDOデータの確認待ちをLGWRバックグラウンド・プロセスがブロックする時間を秒数で指定します。

確認がNET_TIMEOUTの秒数内に受信されない場合は、エラーがログに記録され、その宛先へのREDO転送セッションは終了します。

カテゴリ	NET_TIMEOUT=seconds
データ型	数値

カテゴリ	NET_TIMEOUT=seconds
有効な値	1 脚注 3 から 1200
デフォルト値	30 秒
必須属性	SYNC
属性との競合	ASYNC(ASYNC 属性を指定しても、REDO 転送サービスでは無視され、エラーは戻されません。)
対応先	プライマリ・データベースの V\$ARCHIVE_DEST ビューの NET_TIMEOUT 列

脚注3

最小値は1秒に設定できますが、一時的なネットワーク・エラーによるスタンバイ・データベースからの切断を回避するために、最小値は8から10秒に設定することをお勧めします。

使用上のノート

- NET_TIMEOUT属性はオプションです。ただし、NET_TIMEOUT属性を指定しないと30秒に設定されますが、プライマリ・データベースが停止する可能性があります。この状況を回避するには、NET_TIMEOUT属性に0(ゼロ)以外の小さい値を指定することによって、ネットワーク・サーバーからのステータスを待機する際、ユーザーが指定したタイムアウト時間の期限が切れた後に、プライマリ・データベースが操作を続行できます。
- Oracle Database 12cリリース12.2 (12.2.0.1)より、すべての同期スタンバイ宛先に対してグローバルな、データベース初期化パラメータDATA_GUARD_SYNC_LATENCYがあります。少なくとも1つの同期スタンバイがREDOの受信を確認した後、後続の宛先を切断するまで、プライマリ・データベースが待機できる最大時間(秒単位)を定義します。
[『Oracle Databaseリファレンス』](#)を参照してください。

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10秒に指定する方法を示しています。

```
LOG_ARCHIVE_DEST_2='SERVICE=stby1 SYNC NET_TIMEOUT=10'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

17.12 NOREGISTER

NOREGISTER属性は、アーカイブREDOログ・ファイルの場所が対応する宛先に記録されないことを示します。

カテゴリ	NOREGISTER
データ型	キーワード
有効な値	該当なし
デフォルト値	該当なし

カテゴリ	NOREGISTER
必須属性	SERVICE
属性との競合	LOCATION
対応先	V\$ARCHIVE_DEST ビューの DESTINATION および TARGET 列

使用上のノート

- NOREGISTER属性は、スタンバイ・データベースの宛先がOracle Data Guard構成の一部である場合はオプションです。
- NOREGISTER属性は、宛先がOracle Data Guard構成の一部でない場合は必須です。
- これは、リモートの宛先のための属性です。各アーカイブREDOログ・ファイルの位置は、常にプライマリ・データベースの制御ファイルに記録されます。
- この属性は、ダウンストリームGoldenGateマイニング設定のないOracle Data Guard構成では使用しないでください。このシナリオでNOREGISTERを使用すると、スイッチオーバー操作中に問題が発生する可能性があります。

例

次の例は、NOREGISTER属性を示しています。

```
LOG_ARCHIVE_DEST_5=' NOREGISTER'
```

17.13 PRIORITY

PRIORITY属性は、ログ・アーカイブ先のコレクション内の優先順位を指定するために使用されます。

優先度は1から8まで番号付けされます。値が低いほど高い優先度を表します。最も低い優先順位(PRIORITY=8)は、その優先順位がアクティブな場合にその優先順位の宛先がすべてアクティブにされるという意味で特殊です。より高い優先順位の宛先がサービスに戻ると、その宛先がアクティブにされ、低い優先順位の宛先はすべて非アクティブにされます。

カテゴリ	Priority= <i>integer</i>
データ型	整数
有効な値	1 から 8
デフォルト値	1
必須属性	SERVICE
属性との競合	ALTERNATE
対応先	該当なし

使用上のノート

- PRIORITY属性は常にGROUP属性とともに使用されて、グループのメンバー(REDO宛先)の規則正しい有効化とフォールバックを提供します。

例

次の例は、基本的な概念を説明するためのもので、示されたとおりに使用することを意図していません。構成に応じて、DB_UNIQUE_NAMEなどの他の必須パラメータが存在する場合があります。優先順位を定義するサンプル・ログ・アーカイブ先の設定は、次のとおりです：

```
LOG_ARCHIVE_DEST_1='SERVICE=FS1 SYNC GROUP=1 PRIORITY=1'  
LOG_ARCHIVE_DEST_2='SERVICE=FS2 SYNC GROUP=1 PRIORITY=1'  
LOG_ARCHIVE_DEST_3='SERVICE=FS3 ASYNC GROUP=1 PRIORITY=2'  
LOG_ARCHIVE_DEST_4='SERVICE=TS ASYNC GROUP=1 PRIORITY=3'
```

この宣言結果の動作は、次のようになります。

- プライマリは、2つの優先遠隔同期インスタンスFS1またはFS2のいずれかに送信します。
- FS1とFS2の両方が使用不可になった場合、プライマリはFS3に送信します(この場合はASYNC経由)。
- プライマリがFS3に対して送信中にFS1またはFS2のいずれかが使用可能になった場合、プライマリはその使用可能な優先ログ・アーカイブ先にフェイルバックします。
- 優先順位が高い3つのログ・アーカイブ先のすべてに障害が発生した場合、プライマリはTS (ターミナル・スタンバイ)への送信を開始します。TSへの送信中にFS1、FS2またはFS3が使用可能になった場合、プライマリは新しく使用可能になった優先順位の高いほうの宛先にスイッチされます。

関連項目:

- [グループ内のログ・アーカイブ先に対する優先順位の割当て](#)

17.14 REOPEN

REOPEN属性は、REDO転送サービスが失敗した宛先の再オープンを試行するまでの最小秒数を指定します。

カテゴリ	REOPEN [=seconds]
データ型	数値
有効な値	0 秒以上
デフォルト値	300 秒
必須属性	なし
属性との競合	該当なし
対応先	V\$ARCHIVE_DEST ビューの REOPEN_SECS および MAX_FAILURE 列

使用上のノート

- REOPEN属性はオプションです。

- REDO転送サービスは、失敗した宛先の再オープンをログ・スイッチ時に試行します。
- REDO転送サービスは、最後のエラーの時刻にREOPEN間隔を加算した時刻が現在の時刻より小さいかどうかをチェックします。その場合、REDO転送サービスは宛先の再オープンを試行します。
- REOPENは、接続障害のみでなく、すべてのエラーに適用されます。エラーには、ネットワーク障害、ディスク・エラー、クォータ例外などがありますが、これらに制限されません。
- オプションの宛先にREOPENを指定すると、エラーが発生した場合にOracleデータベースでオンラインREDOログ・ファイルを上書きできます。MANDATORY宛先にREOPENを指定すると、REDO転送サービスは、REDOデータを正常に転送できない場合にプライマリ・データベースを停止します。この場合には、次のオプションを考慮します。
 - 宛先を遅延させる、宛先をOPTIONALとして指定する、またはSERVICE属性値を変更することで宛先を変更。
 - 代替宛先の指定。
 - 宛先の無効化。

例

次の例は、REOPEN属性を示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 MANDATORY REOPEN=60'
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

17.15 SYNCおよびASYNC

SYNCおよびASYNC属性は、使用するREDO転送モードが同期(SYNC)なのか、非同期(ASYNC)なのかを指定します。

カテゴリ	SYNC	ASYNC
データ型	キーワード	キーワード
有効な値	該当なし	該当なし
デフォルト値	該当なし	なし
必須属性	なし	なし
属性との競合	ASYNC、LOCATION	SYNC、LOCATION
対応先	V\$ARCHIVE_DEST ビューの TRANSMIT_MODE 列	V\$ARCHIVE_DEST ビューの TRANSMIT_MODE 列

使用上のノート

- LOG_ARCHIVE_DEST_11からLOG_ARCHIVE_DEST_31パラメータでは、SYNC属性はサポートされません。
- トランザクションをコミットするには、SYNC属性が指定されている使用可能な各宛先で、そのトランザクションによって生成されたREDOデータを受信しておく必要があります。
- プライマリ・データベースおよびロジカル・スタンバイの、宛先1から10のデフォルトはASYNC (リアルタイム・カスケード)に設

定されます。

フィジカル・スタンバイ、スナップショット・スタンバイ、および遠隔同期インスタンスの、宛先1から10のデフォルトはARCH転送モードに設定されます。(ARCH属性は非推奨です。この状況でARCHを使用することは、単にリアルタイムでないカスケードを示します。)

宛先11から31のデフォルトは常にASYNCに設定されます。

- 関連項目:
- LOG_ARCHIVE_DEST_ *n* 非推奨属性の詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

例

次の例は、SYNC属性が指定されているLOG_ARCHIVE_DEST_ *n* パラメータを示しています。

```
LOG_ARCHIVE_DEST_3='SERVICE=stby1 SYNC'  
LOG_ARCHIVE_DEST_STATE_3=ENABLE
```

17.16 TEMPLATE

TEMPLATE属性は、宛先でアーカイブされたREDOログの名前について、ディレクトリ仕様と書式テンプレートを定義します。

このテンプレートは、REDO宛先のLOG_ARCHIVE_FORMAT初期化パラメータにより定義されたデフォルトのファイル名書式とは異なるファイル名を生成するために使用されます。

カテゴリ	TEMPLATE=filename_template_%t_%s_%r
データ型	文字列値
有効な値	該当なし
デフォルト値	なし
必須属性...	SERVICE
属性との競合 ...	LOCATION
対応先 ...	V\$ARCHIVE_DEST ビューの REMOTE_TEMPLATE および REGISTER 列

使用上のノート

- TEMPLATE属性はオプションです。TEMPLATEが指定されていない場合、アーカイブされたREDOログには、LOG_ARCHIVE_FORMAT初期化パラメータの値を使用して名前が付けられます。
- TEMPLATE属性は、リモート・アーカイブ先でLOG_ARCHIVE_FORMAT初期化パラメータ設定をオーバーライドします。
- TEMPLATE属性は、リモート宛先(SERVICE属性で指定された宛先)でのみ有効です。
- [表17-1](#)で説明されているとおり、filename_templateに指定した値には、%s、%t、および%rディレクティブが必要

です。

表17-1 TEMPLATE属性のディレクティブ

ディレクティブ	説明
%t	インスタンス・スレッド番号を置換します。
%T	0(ゼロ)を埋め込んだインスタンス・スレッド番号を置換します。
%s	ログ・ファイル順序番号を置換します。
%S	0(ゼロ)を埋め込んだログ・ファイル順序番号を置換します。
%r	リセットログ ID を置換します。
%R	0(ゼロ)を埋め込んだリセットログ ID を置換します。

- filename_template値は宛先に送信され、そこでファイル名を作成する前に、変換および検証されます。

17.17 VALID_FOR

VALID_FOR属性には、REDOデータを宛先に書き込むかどうかを指定します。

次の要因について考えてみます。

- データベースがプライマリ・ロールとスタンバイ・ロールのどちらで現在実行されているか
- オンラインREDOログ・ファイルまたはスタンバイREDOログ・ファイル(あるいはその両方)が、現在この宛先のデータベースでアーカイブされているかどうか

カテゴリ	VALID_FOR=(redo_log_type, database_role)
データ型	文字列値
有効な値	該当なし
デフォルト値	VALID_FOR=(ALL_LOGFILES, ALL_ROLES)
必須属性	なし
属性との競合	なし
対応先	V\$ARCHIVE_DEST ビューの VALID_NOW、VALID_TYPE および VALID_ROLE 列

使用上のノート

- VALID_FOR属性はオプションです。ただし、Oracle Data Guard構成内の各データベースでREDO転送先ごとに

VALID_FOR属性を指定し、構成内のスタンバイ・データベースへのロール推移後もREDO転送が続行されるようにすることをお勧めします。

- LOG_ARCHIVE_DEST_*n*宛先ごとにこれらの要因を構成するには、キーワードのペア (VALID_FOR=(redo_log_type, database_role))を使用してこの属性を指定します。
 - redo_log_typeキーワードは、次のいずれかをアーカイブする場合に有効な宛先を識別します。
 - ONLINE_LOGFILE: この宛先は、オンラインREDOログ・ファイルをアーカイブする場合のみ有効です。
 - STANDBY_LOGFILE: この宛先は、スタンバイREDOログ・ファイルをアーカイブする場合のみ有効です。
 - ALL_LOGFILES: この宛先は、オンラインREDOログ・ファイルまたはスタンバイREDOログ・ファイルをアーカイブする場合に有効です。
 - database_roleキーワードは、この宛先がアーカイブに有効なロールを識別します。
 - PRIMARY_ROLE: この宛先は、データベースがプライマリ・ロールで実行されている場合のみ有効です。
 - STANDBY_ROLE: この宛先は、データベースがスタンバイ・ロールで実行されている場合のみ有効です。
 - ALL_ROLES: この宛先は、データベースがプライマリ・ロールまたはスタンバイ・ロールで実行されている場合に有効です。
- 宛先にVALID_FOR属性を指定しないと、データベースがプライマリ・ロールまたはスタンバイ・ロールで稼働しているかどうかに関係なく、デフォルトで、オンラインREDOログ・ファイルとスタンバイREDOログ・ファイルのアーカイブが宛先で有効になります。このデフォルトの動作は、VALID_FOR属性で (ALL_LOGFILES, ALL_ROLES) キーワード・ペアを設定した場合と同様です。
- VALID_FOR属性を使用すると、同じ初期化パラメータ・ファイルをプライマリ・ロールとスタンバイ・ロールの両方に使用できます。

例

次の例は、VALID_FORのデフォルトのキーワードのペアを示します。

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1/oracle/oradata VALID_FOR=(ALL_LOGFILES, ALL_ROLES)'
```

このデータベースがプライマリ・ロールまたはスタンバイ・ロールで稼働している場合、宛先1は、すべてのログ・ファイルをローカル・ディレクトリ/disk1/oracle/oradataにアーカイブします。

18 Oracle Data Guardに関連するSQL文

Oracle Data Guardに関連するSQL文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、数多くのSQLおよびSQL*Plus文があります。

次のトピックを参照してください。

- [ALTER DATABASE文](#)
- [ALTER SESSION文](#)
- [ALTER SYSTEM文](#)

すべてのSQL文の完全な構文と説明は、『[Oracle Database SQL言語リファレンス](#)』を参照してください。

ALTER SYSTEM SET文を使用して設定および動的な更新を行うことができる初期化パラメータのリストは、『[初期化パラメータ](#)』を参照してください。

18.1 ALTER DATABASE文

この表で、Oracle Data Guardに関連のあるALTER DATABASE文について説明します。

表18-1 Data Guard環境で使用されるALTER DATABASE文

ALTER DATABASE文	説明
ACTIVATE [PHYSICAL LOGICAL] STANDBY DATABASE FINISH APPLY	フェイルオーバーを実行します。スタンバイ・データベースは、マウント後に、この文を使用してアクティブ化する必要があります。 ノート: ALTER DATABASE ACTIVATE STANDBY DATABASE 文でフェイルオーバーを行うと、データが失われるので使用しないでください。かわりに次のベスト・プラクティスを使用します。 ● フィジカル・スタンバイ・データベースの場合、ALTER DATABASE RECOVER MANAGED STANDBY DATABASE 文で FINISH キーワードを使用します。これにより、ロールの推移がすばやく実行されるため、データ消失は防止されるかまたは最小限に抑えられ、他のスタンバイ・データベースが使用できなくなることはありません。 ● ロジカル・スタンバイ・データベースの場合、ALTER DATABASE PREPARE TO SWITCHOVER および ALTER DATABASE COMMIT TO SWITCHOVER 文を使用します。
ADD [STANDBY] LOGFILE [THREAD integer] [GROUP integer] <i>filespec</i>	指定したスレッドに 1 つ以上のオンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループを追加し、スレッドが割り当てられたインスタンスでログ・ファイルを使用できるようにします。 この文の例については、『REDO ログ・ファイル・グループの追加または』

ALTER DATABASE文	説明
	削除 を参照してください。
ADD [STANDBY] LOGFILE MEMBER 'filename' [REUSE] TO logfile-descriptor	既存のオンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループに新しいメンバーを追加します。
[ADD DROP] SUPPLEMENTAL LOG DATA {PRIMARY KEY UNIQUE INDEX} COLUMNS	この文は、ロジカル・スタンバイ・データベース専用です。 ロジカル・スタンバイ・データベースを作成する前に、サプリメンタル・ロギングを使用可能にするために使用します。サプリメンタル・ロギングがロジカル・スタンバイ・データベースに対する変更のソースであるため、このようにする必要があります。完全なサプリメンタル・ロギングを実装するには、この文で PRIMARY KEY COLUMNS または UNIQUE INDEX COLUMNS キーワードを指定する必要があります。
COMMIT TO SWITCHOVER	次のスイッチオーバーを実行します。 ● カレント・プライマリ・データベースをスタンバイ・データベース・ロールに変更する ● 1 つのスタンバイ・データベースをプライマリ・データベース・ロールに変更する フィジカル・スタンバイ・データベースへスイッチオーバーする場合、Oracle Database 12c リリース 1 (12.1) 時点では、COMMIT TO SWITCHOVER 文は SWITCHOVER TO 文で置き換わりました。COMMIT TO SWITCHOVER 文は引き続きサポートされますが、新しい SWITCHOVER TO 文を使用することをお勧めします。 ノート: ロジカル・スタンバイ・データベースでは、ALTER DATABASE COMMIT TO SWITCHOVER 文を発行する前に、ALTER DATABASE PREPARE TO SWITCHOVER 文を発行し、スイッチオーバーのためにデータベースを準備します。 この文の例については、「古い構文を使用したフィジカル・スタンバイ・データベースへのスイッチオーバーの実行」および「古い構文を使用したフィジカル・スタンバイ・データベースへのフェイルオーバーの実行」を参照してください。
CONVERT TO [[PHYSICAL SNAPSHOT] STANDBY] DATABASE	フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースに、またはその逆に変換します。
CREATE [PHYSICAL LOGICAL] STANDBY	フィジカルまたはロジカル・スタンバイ・データベースの維持に使用される制御ファイルを作成します。この文はプライマリ・データベースで発行

ALTER DATABASE文	説明
CONTROLFILE AS 'filename' [REUSE]	します。 この文の例については、「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。
DROP [STANDBY] LOGFILE logfile_descriptor	オンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループのすべてのメンバーを削除します。 この文の例については、「REDO ログ・ファイル・グループの追加または削除」を参照してください。
DROP [STANDBY] LOGFILE MEMBER 'filename'	オンライン REDO ログ・ファイル・グループまたはスタンバイ REDO ログ・ファイル・グループの 1 つ以上のメンバーを削除します。
FAILOVER TO <i>target_db_name</i>	この文は、フィジカル・スタンバイ・データベース専用です。 指定したホスト・データベースへのフェイルオーバーを開始します。
[NO]FORCE LOGGING	一時表領域および一時セグメントに対する変更を除くデータベースに対するすべての変更を、Oracle データベースが記録するかどうかを制御します。[NO]FORCE LOGGING 句は、スタンバイ・データベース間の一貫性の欠如を防止するために必要です。 この文を発行する場合、プライマリ・データベースを少なくともマウントしておく(また、オープンもできるようにしておく)必要があります。この文の例については、「強制ロギングの有効化」を参照してください。
GUARD	ロジカル・スタンバイ・データベース内の表に対するユーザー・アクセスを制御します。可能な値は、ALL、STANDBY および NONE です。詳細は、「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」を参照してください。
MOUNT [STANDBY DATABASE]	スタンバイ・データベースをマウントし、スタンバイ・インスタンスが REDO データをプライマリ・インスタンスから受信できるようにします。
OPEN	すでに起動され、マウントされたデータベースをオープンします。 ● フィジカル・スタンバイ・データベースは読取り専用モードでオープンされ、ユーザーを読取り専用トランザクションに制限し、REDO データの生成を防ぎます。 ● ロジカル・スタンバイ・データベースは読取り/書込みモードで

ALTER DATABASE文	説明
	オープンされます。
PREPARE TO SWITCHOVER	この文は、ロジカル・スタンバイ・データベース専用です。 スイッチオーバーが実行される前に LogMiner ディクショナリを構築することにより、プライマリ・データベースおよびロジカル・スタンバイ・データベースをスイッチオーバー用に準備します。ディクショナリ構築の完了後、ALTER DATABASE COMMIT TO SWITCHOVER 文を発行し、プライマリおよびロジカル・スタンバイ・データベースのロールを切り替えます。 この文の例については、「ロジカル・スタンバイ・データベースへのスイッチオーバーの実行」を参照してください。
RECOVER MANAGED STANDBY DATABASE [{ DISCONNECT [FROM SESSION] PARALLEL n] NODELAY UNTIL CHANGE integer }]	この文はフィジカル・スタンバイ・データベースに対する REDO Apply を開始および制御します。RECOVER MANAGED STANDBY DATABASE 句は、マウント、オープンまたはクローズされているフィジカル・スタンバイ・データベースで使用できます。例については、「フィジカル・スタンバイ・データベースの起動」のステップ 3 および「REDO データのフィジカル・スタンバイ・データベースへの適用」を参照してください。 ノート: 複数の句およびキーワードが非推奨となり、下位互換性のためにのみサポートされています。これらの非推奨の句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
RECOVER MANAGED STANDBY DATABASE CANCEL	CANCEL 句は、現行のアーカイブ REDO ログ・ファイルの適用後、フィジカル・スタンバイ・データベースへの Redo Apply を取り消します。 ノート: 複数の句およびキーワードが非推奨となり、下位互換性のためにのみサポートされています。これらの句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
RECOVER MANAGED STANDBY DATABASE FINISH	FINISH 句は、ターゲット・フィジカル・スタンバイ・データベースでフェイルオーバーを開始し、現行のスタンバイ REDO ログ・ファイルをリカバリします。FINISH 句を使用するのは、プライマリ・データベースに障害が発生した場合のみです。この句により、指定の遅延間隔がオーバーライドされます。 ノート: 複数の句およびキーワードが非推奨となり、下位互換性のためにのみサポートされています。これらの句の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。

ALTER DATABASE文	説明
REGISTER [OR REPLACE] [PHYSICAL LOGICAL] LOGFILE <i>filespec</i>	手動でコピーされたアーカイブ REDO ログ・ファイルを登録できます。 ノート: このコマンドは、対応するアーカイブ REDO ログ・ファイルを手動でスタンバイ・データベースにコピーした後にのみ発行します。ログ・ファイルのコピー中、あるいはログ・ファイルが存在しないときにこのコマンドを発行すると、後でスタンバイ・データベースにエラーが発生する可能性があります。
RECOVER TO LOGICAL STANDBY new_database_name	適用サービスに対して、データベースをロジカル・スタンバイ・データベースに変換するコマンドを発行するまでは、引き続き変更をフィジカル・スタンバイ・データベースに適用するように指示します。詳細は、「ロジカル・スタンバイ・データベースへの変換」を参照してください。
RESET DATABASE TO INCARNATION integer	データベースのターゲット・リカバリ・インカネーションを、現在のインカネーションから別のインカネーションにリセットします。
SET STANDBY DATABASE TO MAXIMIZE {PROTECTION AVAILABILITY PERFORMANCE}	この句を使用して、Oracle Data Guard 構成におけるデータの保護レベルを指定します。この句をプライマリ・データベースから指定します。
START LOGICAL STANDBY APPLY INITIAL [scn-value]] [NEW PRIMARY dblink]	この文は、ロジカル・スタンバイ・データベース専用です。 ロジカル・スタンバイ・データベースで SQL Apply を開始します。この文の例については、「SQL Apply の開始」を参照してください。
{STOP ABORT} LOGICAL STANDBY APPLY	この文は、ロジカル・スタンバイ・データベース専用です。 規則正しく、STOP 句を使用してロジカル・スタンバイ・データベースで SQL Apply を停止します。ABORT 句を使用すると SQL Apply が突然停止します。この文の例については、「ロジカル・スタンバイ・データベースへのフェイルオーバーの実行」を参照してください。
SWITCHOVER TO <i>target_db_name</i>	この文は、フィジカル・スタンバイ・データベース専用です。 プライマリ・データベースで、指定したフィジカル・スタンバイ・データベースへのスイッチオーバーを開始します。

18.2 ALTER SESSION文

この表で、Oracle Data Guardに関連のあるALTER SESSION文について説明します。

表18-2 Oracle Data Guard環境で使用されるALTER SESSION文

ALTER SESSION文	説明
ALTER SESSION [ENABLE DISABLE] GUARD	この文は、ロジカル・スタンバイ・データベース専用です。 権限を付与されているユーザーは、この文を使用して、現在のセッションでデータベース・ガードを選択または解除できます。 詳細は、「ロジカル・スタンバイ・データベースへの変更」を参照してください。
ALTER SESSION SYNC WITH PRIMARY	この文は、フィジカル・スタンバイ・データベース専用です。 この文は、起動時にフィジカル・スタンバイにより受信されたすべての REDO データが適用されるまでブロックすることにより、フィジカル・スタンバイ・データベースをプライマリ・データベースと同期します。 詳細は、「リアルタイム問合せ環境での REDO Apply の強制同期」を参照してください。

18.3 ALTER SYSTEM文

この表で、Oracle Data Guardに関連のあるALTER SYSTEM文について説明します。

表18-3 Oracle Data Guard環境で使用されるALTER SYSTEM文

ALTER SYSTEM文	説明
ALTER SYSTEM FLUSH REDO TO target_db_name [[NO] CONFIRM APPLY]	この文は、REDO データをすべて、プライマリ・データベースからスタンバイ・データベースにフラッシュし、必要に応じて、フラッシュされた REDO データがフィジカルまたはロジカル・スタンバイ・データベースに適用されるまで待機します。 この文は、マウントされているが、オープンされていないプライマリ・データベースで発行する必要があります。

19 Oracle Data Guardに関連するビュー

Oracle Data Guard環境を監視するときに特に便利なビューが多数あります。

表19-1では、各ビューについて説明し、そのビューがフィジカル・スタンバイ・データベース、ロジカル・スタンバイ・データベース、スナップショット・スタンバイ・データベース、プライマリ・データベースのいずれに適用されるのかを示します。ビューの詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

表19-1 Oracle Data Guard構成に関連のあるビュー

ビュー	データベース	説明
DBA_LOGSTDBY_EVENT S	ロジカルのみ	ロジカル・スタンバイ・データベースのアクティビティに関する情報を含んでいます。SQL Apply がロジカル・スタンバイ・データベースに REDO を適用中に発生した障害原因を特定するために使用します。
DBA_LOGSTDBY_HISTO RY	ロジカルのみ	Oracle Data Guard 構成におけるロジカル・スタンバイ・データベースのスイッチオーバーとフェイルオーバーの履歴を表示します。すべてのロール・トランジションについて、ローカル・システムで処理または作成されたすべての REDO ログ・ストリームを示すことでこれを行います。(ロール・トランジション後は、新しいログ・ストリームが開始し、新しいプライマリ・データベースによってログ・ストリーム順序番号が増分されます。)
DBA_LOGSTDBY_LOG	ロジカルのみ	ロジカル・スタンバイ・データベースにレジストリされているログ・ファイルを表示します。
DBA_LOGSTDBY_NOT_U NIQUE	ロジカルのみ	主キー制約または NOT NULL の一意制約がない表を識別します。
DBA_LOGSTDBY_PARAM ETERS	ロジカルのみ	SQL Apply で使用されるパラメータのリストが格納されます。
DBA_LOGSTDBY_SKIP	ロジカルのみ	SQL Apply によってスキップされる表をリスト表示します。
DBA_LOGSTDBY_SKIP_T RANSACTION	ロジカルのみ	選択されているスキップ設定をリスト表示します。
DBA_LOGSTDBY_UNSUP PORTED	ロジカルのみ	サポートされないデータ型が含まれているスキーマおよび表(およびその表の列)を識別します。このビューは、ロジカル・スタンバイ・データベースの作成を準備する際に使用します。
DBA_ROLLING_UNSUPP ORTED	ロジカルのみ	DBMS_ROLLING PL/SQL パッケージを使用するロジカル・スタンバイ・データベースのローリング・アップグレード操作でサポートされないデータ型を含むスキーマ、表および列を示します。DBMS_ROLLING を使用し

ビュー	データベース	説明
		たローリング・アップグレードの前にこのビューを使用して、何がサポートされていないか確認してください。
V\$ARCHIVE_DEST	プライマリ、フィジカル、スナップショットおよびロジカル	Oracle Data Guard 構成内のすべての宛先の現在の値、モードおよび状況を含めた説明を表示します。 ノート: 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_STAT US	プライマリ、フィジカル、スナップショットおよびロジカル	アーカイブ REDO ログ宛先の実行時および構成用の情報を表示します。 ノート: このビューに表示される情報は、インスタンスの停止後には保存されません。
V\$ARCHIVE_GAP	フィジカル、スナップショットおよびロジカル	アーカイブ REDO ログ・ファイルのギャップの識別に役立つ情報を表示します。
V\$ARCHIVED_LOG	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルのアーカイブ REDO ログ情報を、アーカイブ REDO ログ・ファイル名を含めて表示します。
V\$DATABASE	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルからのデータベース情報を表示します。ファスト・スタート・フェイルオーバー(Oracle Data Guard Broker でのみ使用可能)に関する情報が含まれます。
V\$DATABASE_INCARNATION	プライマリ、フィジカル、スナップショットおよびロジカル	すべてのデータベース・インカーネーションに関する情報を表示します。 Oracle Database では、RESETLOGS オプションを使用してデータベースがオープンされるたびに、新しいインカーネーションが作成されます。V\$DATABASE ビューには、現在のインカーネーションのみでなく、前のインカーネーションのレコードも表示されます。
V\$DATAFILE	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルからのデータファイル情報を表示します。
V\$DATAGUARD_CONFIG	プライマリ、フィジカル、スナップショットおよびロジカル	初期化パラメータ、DB_UNIQUE_NAME および LOG_ARCHIVE_CONFIG で定義された一意のデータベース名をリストにします。
V\$DATAGUARD_STATS	プライマリ、フィジカル、スナップショットおよびロジカル	適用ラグやトランスポート・ラグを含む各種 Oracle Data Guard 統計を表示します。このビューは、いかなるスタンバイ・データベースのイン

ビュー	データベース	説明
	よびロジカル	スタンスでも問合せできます。プライマリ・データベースを問い合わせた場合、1 行も戻されません。例および詳細は、 「ロールの推移のターゲット・スタンバイ・データベースの選択」 も参照してください。
V\$DATAGUARD_STATUS	プライマリ、フィジカル、スナップショットおよびロジカル	通常はアラート・ログまたはサーバー・プロセスのトレース・ファイルへのメッセージによってトリガーされるイベントが表示および記録されます。
V\$FS_FAILOVER_STATS	プライマリ	システムで発生したファスト・スタート・フェイルオーバーに関する統計を表示します。
V\$LOG	プライマリ、フィジカル、スナップショットおよびロジカル	オンライン REDO ログ・ファイルのログ・ファイル情報が格納されます。
V\$LOGFILE	プライマリ、フィジカル、スナップショットおよびロジカル	オンライン REDO ログ・ファイルとスタンバイ REDO ログ・ファイルに関する情報が格納されます。
V\$LOG_HISTORY	プライマリ、フィジカル、スナップショットおよびロジカル	制御ファイルからのログ履歴情報が格納されます。
V\$LOGSTDBY_PROCESS	ロジカルのみ	SQL Apply の動作を示す動的情報を提供します。このビューは、ロジカル・スタンバイ・データベースに SQL Apply を適用する際に、パフォーマンス問題を診断する場合に非常に役立ちます。
V\$LOGSTDBY_PROGRESS	ロジカルのみ	ロジカル・スタンバイ・データベースでの SQL Apply の進捗状況を表示します。
V\$LOGSTDBY_STATE	ロジカルのみ	SQL Apply とロジカル・スタンバイ・データベースの実行状態に関して、V\$LOGSTDBY_PROCESS および V\$LOGSTDBY_STATS ビューの情報を連結します。
V\$LOGSTDBY_STATS	ロジカルのみ	LogMiner 統計、現行のステータスおよび SQL Apply 時のロジカル・スタンバイ・データベースのステータス情報を表示します。SQL Apply が実行されていない場合、統計の値はクリアされます。
V\$LOGSTDBY_TRANSACTION	ロジカルのみ	ロジカル・スタンバイ・データベースで SQL Apply により処理中のすべてのアクティブ・トランザクションに関する情報を表示します。

ビュー	データベース	説明
V\$MANAGED_STANDBY	フィジカルおよびスナップショット	フィジカル・スタンバイ・データベースに関連する Oracle データベース・プロセスの現在の状態を表示します。 ノート: このビューに表示される情報は、インスタンスの停止後には保存されません。
V\$REDO_DEST_RESP_HISTOGRAM	プライマリ	SYNC 転送用に構成された宛先に関するレスポンス時間の情報が格納されます。 ノート: 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\$STANDBY_EVENT_HISTOGRAM	フィジカル	フィジカル・スタンバイの適用ラグ値のヒストグラムが格納されます。エントリは、対応する適用ラグのバケット(REDO Apply プロセス 1 秒単位)に作成されます。(このビューは、リアルタイム問合せモードで開かれたフィジカル・スタンバイ・データベースでのみ行を戻します。) ノート: 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\$STANDBY_LOG	フィジカル、スナップショットおよびロジカル	スタンバイ REDO ログ・ファイルのログ・ファイル情報が格納されます。

付録

この部は、次の付録で構成されています。

- [Oracle Data Guardのトラブルシューティング](#)
- [Oracle Data Guard構成におけるデータベースのアップグレードおよびダウングレード](#)
- [ロジカル・スタンバイ・データベースでサポートされるデータ型およびDDL](#)
- [Oracle Data GuardとOracle Real Application Clusters](#)
- [Recovery Managerを使用したスタンバイ・データベースの作成](#)
- [アーカイブ・トレースの設定](#)
- [古い構文を使用したロールの推移の実行](#)
- [ALTERNATE属性を使用したリモート代替宛先の構成](#)

A Oracle Data Guardのトラブルシューティング

スタンバイ・データベースで発生する可能性がある問題の一部と、それらに対応するためのトラブルシューティング手順を次に示します。

- [一般的な問題](#)
- [ログ・ファイル宛先の障害](#)
- [ロジカル・スタンバイ・データベース障害の処理](#)
- [フィジカル・スタンバイ・データベースへのスイッチオーバーの問題](#)
- [ロジカル・スタンバイ・データベースへのスイッチオーバーの問題](#)
- [SQL Applyが停止した場合の処置](#)
- [REDOデータ転送のネットワーク調整](#)
- [スタンバイ・データベースのディスクのパフォーマンスが遅い](#)
- [プライマリ・データベースの停止を回避するにはログ・ファイルを一致させる必要がある](#)
- [ロジカル・スタンバイ・データベースのトラブルシューティング](#)

A.1 一般的な問題

スタンバイ・データベースの使用時に発生する可能性がある一般的な問題の一部を次に示します。

- [ALTER DATABASE文を使用したデータファイルの改名](#)
- [スタンバイ・データベースがプライマリ・データベースからREDOデータを受信しない](#)
- [フィジカル・スタンバイ・データベースをマウントできない](#)

A.1.1 ALTER DATABASE文を使用したデータファイルの改名

STANDBY_FILE_MANAGEMENT初期化パラメータをAUTOに設定すると、スタンバイ・サイトのデータファイルを改名できません。

STANDBY_FILE_MANAGEMENT初期化パラメータをAUTOに設定した場合、次のSQL文は使用できなくなります。

- ALTER DATABASE RENAME
- ALTER DATABASE ADD/DROP LOGFILE
- ALTER DATABASE ADD/DROP STANDBY LOGFILE MEMBER
- ALTER DATABASE CREATE DATAFILE AS

スタンバイ・データベースでこれらの文のいずれかを使用しようとすると、エラーが表示されます。次に例を示します。

```
SQL> ALTER DATABASE RENAME FILE '/disk1/oracle/oradata/payroll/t_db2.log' to 'dummy';
```

```
alter database rename file '/disk1/oracle/oradata/payroll/t_db2.log' to 'dummy'
*
ERROR at line 1:
ORA-01511: error in renaming log/datafiles
ORA-01270: RENAME operation is not allowed if STANDBY_FILE_MANAGEMENT is auto
```

フィジカル・スタンバイ・データベースにデータファイルを追加する方法を学習するには、[「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。

A.1.2 スタンバイ・データベースがプライマリ・データベースからREDOデータを受信しない

スタンバイ・サイトがREDOデータを受信していない場合は、V\$ARCHIVE_DESTビューの問合せを行い、エラー・メッセージがないか確認します。

たとえば、次のような問合せを入力します。

```
SQL> SELECT DEST_ID "ID", -  
> STATUS "DB_status", -  
> DESTINATION "Archive_dest", -  
> ERROR "Error" -  
> FROM V$ARCHIVE_DEST WHERE DEST_ID <=5;
```

ID	DB_status	Archive_dest	Error
1	VALID	/vobs/oracle/work/arc_dest/arc	
2	ERROR	standby1	ORA-16012: Archivelog standby database identifier mismatch
3	INACTIVE		
4	INACTIVE		
5	INACTIVE		

5 rows selected.

問合せの出力によって解決しない場合は、次の問題リストを確認します。次のいずれかの条件に該当する場合、REDO転送サービスはスタンバイ・データベースにREDOデータを転送できません。

- プライマリ・データベースのtnsnames.oraファイル内でスタンバイ・インスタンスのサービス名が正しく構成されていない場合。
- プライマリ・データベースのLOG_ARCHIVE_DEST_nパラメータで指定されたOracle Netサービス名が不適切な場合。
- スタンバイ・データベースのLOG_ARCHIVE_DEST_STATE_nパラメータが値ENABLEに設定されていない場合。
- listener.oraファイルがスタンバイ・データベース用に正しく構成されていない場合。
- スタンバイ・サイトでリスナーが開始されていない場合。
- スタンバイ・インスタンスが起動されていない場合。
- スタンバイ・アーカイブ先をプライマリSPFILEまたはテキスト初期化パラメータ・ファイルに追加したが、その変更をまだ使用可能にしていない場合。
- REDO転送の認証が正しく構成されていない場合。REDO転送の認証の構成要件は、3.1.2項を参照してください。
- スタンバイ・データベースのベースとして無効なバックアップを使用した場合(たとえば、間違ったデータベースからのバックアップを使用、または正しい方法でスタンバイ制御ファイルを作成しなかったなど)。

A.1.3 フィジカル・スタンバイ・データベースをマウントできない

スタンバイ制御ファイルが、ALTER DATABASE CREATE [LOGICAL] STANDBY CONTROLFILE ... 文またはRMANコマンドを使用して作成されていない場合、スタンバイ・データベースをマウントすることはできません。

次のタイプの制御ファイル・バックアップは使用できません。

- オペレーティング・システムで作成されたバックアップ
- PHYSICAL STANDBYまたはLOGICAL STANDBYオプションのないALTER DATABASE文を使用して作成されたバックアップ

A.2 ログ・ファイル宛先の障害

MANDATORY宛先にREOPENを指定した場合、REDO転送サービスは、REDOデータを正常に転送できなかったとき、プライマリ・データベースを停止します。

MAX_FAILURE属性を使用する場合は、REOPEN属性は必須です。[例A-1](#)に、再試行時間を5秒に設定し、再試行回数を3回に制限する方法を示します。

例A-1 再試行時間と制限の設定

```
LOG_ARCHIVE_DEST_1='LOCATION=/arc_dest REOPEN=5 MAX_FAILURE=3'
```

LOG_ARCHIVE_DEST_1パラメータのALTERNATE属性を使用して、代替アーカイブ先を指定します。スタンバイ・データベースへのREDOデータの転送に失敗した場合は、代替アーカイブ先を使用できます。転送に失敗したときに、REOPEN属性が指定されていなかった場合、またはMAX_FAILURE属性のしきい値を超えた場合、REDO転送サービスは、次のアーカイブ操作時に代替先へのREDOデータの転送を試みます。

元のアーカイブ先で障害が発生したときに、元のアーカイブ先が自動的に代替アーカイブ先に変更されることを防ぐには、NOALTERNATE属性を使用します。

[例A-2](#)は、エラー発生時に、単一、必須のローカル宛先が異なる宛先へ自動的にフェイルオーバーするように初期化パラメータを設定する方法を示します。

例A-2 代替宛先の指定

```
LOG_ARCHIVE_DEST_1='LOCATION=/disk1 MANDATORY ALTERNATE=LOG_ARCHIVE_DEST_2'  
LOG_ARCHIVE_DEST_STATE_1=ENABLE  
LOG_ARCHIVE_DEST_2='LOCATION=/disk2 MANDATORY'  
LOG_ARCHIVE_DEST_STATE_2=ALTERNATE
```

LOG_ARCHIVE_DEST_1の宛先で障害が発生した場合、アーカイブ・プロセスは、プライマリ・データベースでの次のログ・ファイル・スイッチでLOG_ARCHIVE_DEST_2の宛先に自動的に切り替えます。

A.3 ロジカル・スタンバイ・データベース障害の処理

ロジカル・スタンバイ・データベース障害を処理するための重要なツールは、DBMS_LOGSTDBY.SKIP_ERRORプロシージャです。

表の重要度に基づいて、次のいずれかを実行できます。

- 表や特定のDDLに関する障害を無視する。
- ストアド・プロシージャをフィルタに関連付ける。これによって、文のスキップ、文の実行または代用文の実行について実行時に決定できます。

これらの処理のいずれかを実行すると、SQL Applyの停止を回避できます。存在している問題は、後でDBA_LOGSTDBY_EVENTSビューを問い合わせて検索し、訂正できます。PL/SQLコールアウト・プロシージャでのDBMS_LOGSTDBYパッケージ使用の詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)を参照してください。

A.4 フィジカル・スタンバイ・データベースへのスイッチオーバーの問題

スイッチオーバーの失敗の原因となる可能性のある問題の一部と、考えられる解決策を次に示します。

- [REDOデータが転送されていないためスイッチオーバーできない](#)
- [ORA-01102エラーによりスイッチオーバーできない](#)

- [スイッチオーバー後にREDOデータが適用されない](#)
- [失敗したスイッチオーバーをロールバックして最初からやりなおす](#)

A.4.1 REDOデータが転送されていないためスイッチオーバーできない

スイッチオーバーが正常に完了しない場合は、V\$ARCHIVED_LOGビューのSEQUENCE#列の問合せを行い、元のプライマリ・データベースから転送された最新のREDOデータが、スタンバイ・データベースに適用されていることを確認してください。

最後のREDOデータがスタンバイ・データベースに転送されていない場合は、そのREDOデータが含まれるアーカイブREDOログ・ファイルを元のプライマリ・データベースから古いスタンバイ・データベースに手でコピーし、SQL ALTER DATABASE REGISTER LOGFILE file_specification文を使用して登録します。次に適用サービスを開始すると、アーカイブREDOログ・ファイルが自動的に適用されます。V\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせます。SWITCHOVER_STATUS列からTO PRIMARYまたはSESSIONS ACTIVEが戻されると、プライマリ・ロールへのスイッチオーバーが可能になります。

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

```
SWITCHOVER_STATUS
```

```
-----
```

```
TO PRIMARY
```

```
1 row selected
```

スイッチオーバーを続行するには、[「フィジカル・スタンバイ・データベースへのスイッチオーバーの実行」](#)(フィジカル・スタンバイ・データベースの場合)または[「ロジカル・スタンバイ・データベースへのスイッチオーバーの実行」](#)(ロジカル・スタンバイ・データベースの場合)の説明に従って、ターゲットのスタンバイ・データベースをプライマリ・ロールに切り替える操作を再度実行します。

A.4.2 ORA-01102エラーによりスイッチオーバーできない

スイッチオーバー中にORA-01102エラーが表示された理由と、実行する必要があるアクションが説明されます。

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サイトに存在するとします。ALTER DATABASE SWITCHOVER TO *target_db_name*文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。

ノート:



フィジカル・スタンバイ・データベースが起動後に読み取り専用モードでオープンされていない場合、停止して再起動する必要はありません。

ただし、2番目のデータベースの起動は、ORA-01102エラー「データベースを排他モードでマウントすることができません。」で失敗します。

スタンバイ・データベース(元のプライマリ・データベース)で使用する初期化パラメータ・ファイルにDB_UNIQUE_NAMEパラメータを設定していないと、スイッチオーバー中にこの現象が発生することがあります。スタンバイ・データベースのDB_UNIQUE_NAMEパラメータが設定されていない場合、スタンバイ・データベースとプライマリ・データベースの両方が同じマウント・ロックを使用するため、2つ目のデータベースの起動時にORA-01102エラーが発生します。

処置: DB_UNIQUE_NAME=*unique_database_name*をスタンバイ・データベースが使用する初期化パラメータ・ファイルに追加して、スタンバイ・データベースとプライマリ・データベースを停止し、再起動します。

A.4.3 スイッチオーバー後にREDOデータが適用されない

スイッチオーバー後にアーカイブREDOログ・ファイルが新しいスタンバイ・データベースに適用されない場合は、スイッチオーバー後に一部の環境パラメータまたは初期化パラメータが正しく設定されていなかったことが原因と考えられます。

処置:

- 新しいプライマリ・サイトのtnsnames.oraファイルおよび新しいスタンバイ・サイトのlistener.oraファイルを調べます。スタンバイ・サイトにはリスナーのエントリ、プライマリ・サイトにはそれに対応するサービス名が必要です。
- リスナーがまだ起動されていない場合は、スタンバイ・サイトで起動します。
- プライマリ・サイトからスタンバイ・サイトにREDOデータを正しく転送するための、LOG_ARCHIVE_DEST_*n*初期化パラメータが設定されているかどうかを調べます。たとえば、プライマリ・サイトでV\$ARCHIVE_DEST固定ビューを次のように問い合わせます。

```
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・サイトでLOG_ARCHIVE_FORMAT初期化パラメータが正しく設定されていることを確認します。
- 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。

A.4.4 失敗したスイッチオーバーをロールバックして最初からやりなおす

フィジカル・スタンバイ・データベースでエラーが発生し、スイッチオーバーを続行できない場合は、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。

次のステップを実行します。(この機能は、Oracle Database 11gリリース2(11.2.0.2)以上で使用できます。)

1. 新しいスタンバイ・データベース(古いプライマリ)を停止し、マウントします。
2. 新しいスタンバイ・データベースでREDO Applyを開始します。
3. 新しいスタンバイ・データベースがいつでもプライマリ・ロールに戻せることを確認します。スタンバイ・データベースでV\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせます。次に例を示します。

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

```
SWITCHOVER_STATUS
```

```
-----  
TO_PRIMARY
```

```
1 row selected
```

値TO PRIMARYまたはSESSIONS ACTIVEは、スタンバイ・データベースでプライマリ・ロールに切り替える準備が完了していることを示します。値TO PRIMARYまたはSESSIONS ACTIVEが戻されるまで、この列の問合せを続行します。

4. 次の文を発行して、新しいスタンバイ・データベースを変換してプライマリ・ロールに戻します。

```
SQL> ALTER DATABASE SWITCHOVER TO target_db_name [FORCE];
```

この文が正常に実行されると、データベースはプライマリ・データベース・ロールで実行されるため、それ以降のステップを実行する必要はありません。

この文が正常に実行されなかった場合は、ステップ5に進んでください。

5. プライマリからフィジカル・スタンバイにロールを変更するスイッチオーバーを開始したときに、トレース・ファイルがログ・ディレクトリに書き込まれています。このトレース・ファイルには、元のプライマリ制御ファイルを再作成するために必要なSQL文が含まれています。トレース・ファイルの位置を特定し、SQL文を一時ファイルに抽出します。SQL*Plusから一時ファイルを実行します。これによって、新しいスタンバイ・データベースはプライマリ・ロールに戻されます。
6. 元のフィジカル・スタンバイ・データベースを停止します。
7. 新しいスタンバイ制御ファイルを作成します。このファイルは、プライマリ・データベースとフィジカル・スタンバイ・データベースの再同期化に必要です。フィジカル・スタンバイ制御ファイルを元のフィジカル・スタンバイ・システムにコピーします。フィジカル・スタンバイ制御ファイルの作成方法は、[「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。
8. 元のフィジカル・スタンバイ・インスタンスを再起動します。

この手順が正常終了し、アーカイブ・ギャップ管理が使用可能になると、FALプロセスが起動し、欠落したアーカイブ REDOログ・ファイルをフィジカル・スタンバイ・データベースに再アーカイブします。プライマリ・データベース上でログ・スイッチを強制実行し、プライマリ・データベースとフィジカル・スタンバイ・データベースの両方のアラート・ログを調べて、アーカイブ REDOログ・ファイルの順序番号が正しいことを確認します。

アーカイブ・ギャップ管理の詳細は[「手動によるギャップの解決」](#)を、トレース・ファイルの場所については[「アーカイブ・トレースの設定」](#)を参照してください。

9. スwitchオーバーを再度実行します。

この時点で、Oracle Data Guard構成は初期状態にロールバックされています。最初に失敗したスイッチオーバーの問題をすべて解決した後、スイッチオーバー操作を再度実行できます。

A.5 ロジカル・スタンバイ・データベースへのスイッチオーバーの問題

ロジカル・スタンバイ・データベースに関連するスイッチオーバー操作には、通常、準備とコミットの2つのフェーズがあります。これに対する例外は、ロジカル・スタンバイ・データベースを使用したOracleソフトウェアのローリング・アップグレードの場合、またはOracle Data Guard Brokerを使用している場合です。

ロジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行中、またはOracle Data Guard Brokerで開始されたスイッチオーバー操作の際に障害が発生した場合は、[スイッチオーバー操作のコミット・フェーズ中の障害](#)に直接進んでください。

ノート:



Oracle Data Guard 構成のすべてのデータベースで、フラッシュバック・データベースを有効にすることをお勧めします。この項のステップでは、使用する Oracle Data Guard 構成内のすべてのデータベースで、フラッシュバック・データベースが有効にされていると想定しています。

A.5.1 スwitchオーバー操作の準備フェーズ中の障害

スイッチオーバー操作の準備フェーズ中に障害が発生した場合、スイッチオーバーを取り消して、スイッチオーバー操作を再度初めから実行します。

A.5.1.1 プライマリ・データベースの準備中の障害

ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY文の実行中に障害が発生した場合は、スイッチオーバーの準備フェーズを取り消しできます。

そのためには、次のSQL文をプライマリ・データベースで発行します。

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

これで、スイッチオーバー操作を再度初めから行うことができます。

A.5.1.2 ロジカル・スタンバイ・データベースの準備中の障害

ALTER DATABASE PREPARE TO SWITCHOVER TO PRIMARY文の実行中に障害が発生した場合、プライマリ・データベースおよびターゲット・スタンバイ・データベースで準備操作を取り消す必要があります。

次のステップを実行します。

1. プライマリ・データベースで、スイッチオーバーの準備のために発行した文を取り消します。

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

2. スwitchオーバーのターゲットであるロジカル・スタンバイ・データベースで、スイッチオーバーの準備のために発行した文を取り消します。

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

これで、スイッチオーバー操作を再度初めから行うことができます。

A.5.2 スwitchオーバー操作のコミット・フェーズ中の障害

スイッチオーバーのコミットでは単一のSQL文のみを使用しますが、内部的には多数の操作が実行されます。

実行する必要がある修正処理は、エラー発生時のスイッチオーバー操作のコミットの状態によって異なります。

A.5.2.1 元のプライマリ・データベースの変換での障害

ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY文の実行中に障害が発生した場合に実行できるステップがあります。

1. 元のプライマリ・データベースのV\$DATABASE固定ビューのDATABASE_ROLE列をチェックします。

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

- 列にLOGICAL STANDBYの値が含まれる場合、スイッチオーバー操作は完了していますが、スイッチオーバー後のタスクで障害が発生しています。この場合、データベースを停止して再オープンすることをお勧めします。
- 列にPRIMARYの値が含まれる場合、ステップ2に進みます。

2. 元のプライマリで次の問合せを実行します。

```
SQL> SELECT COUNT(*) FROM SYSTEM.LOGSTDBY$PARAMETERS -  
> WHERE NAME = 'END_PRIMARY';
```

- 問合せで0が戻される場合、プライマリの状態は、スイッチオーバーのコミット・コマンドが発行される前と同一です。修正処理を実行する必要はありません。スイッチオーバー操作のコミットに進むか、[「ロジカル・スタンバイ・データベースの準備中の障害」](#)で説明するようにスイッチオーバー操作を取り消します。
 - 問合せで1が戻される場合、プライマリは一貫性がない状態のため、ステップ3に進む必要があります。
3. インスタンス化された既存のまたは新しいロジカル・スタンバイ・データベースにより保護されるための機能が維持されるよう、元のプライマリ・データベースで対処措置を実行します。

スイッチオーバー操作のコミット中に発生したエラーの基となる原因を修正してSQL文(ALTER DATABASE COMMIT TO

SWITCHOVER TO LOGICAL STANDBY)を再発行します。次のステップを実行することもできます。

- スイッチオーバー・コマンドのコミットを開始したインスタンスのアラート・ログから、元のプライマリへのフラッシュバックに必要なSCNを判別します。この情報は、SQL文ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBYの後に表示されます。

```
LOGSTDBY: Preparing the COMMIT TO SWITCHOVER TO LOGICAL STANDBY DDL at scn
[flashback_scn].
```

- プライマリ・データベースのすべてのインスタンスを停止します。

```
SQL> SHUTDOWN IMMEDIATE;
```

- プライマリ・データベースを排他モードでマウントします。

```
SQL> STARTUP MOUNT;
```

- データベースをアラート・ログから取得したSCNまでフラッシュバックします。

```
SQL> FLASHBACK DATABASE TO BEFORE SCN <flashback_scn>;
```

- プライマリ・データベースをオープンします。

```
SQL> STARTUP;
```

- 元のプライマリ・データベースのデータベース・ガードを低くします。

```
SQL> ALTER DATABASE GUARD NONE;
```

この時点で、プライマリの状態はスイッチオーバーのコミット・コマンドが発行される前と同一です。修正処理は必要ありません。スイッチオーバー操作のコミットに進むか、[「プライマリ・データベースの準備中の障害」](#)で説明するようにスイッチオーバー操作を取り消します。

A.5.2.2 ターゲット・ロジカル・スタンバイ・データベースの変換での障害

ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY文の実行中に障害が発生した場合に実行できるステップがあります。

1. ターゲット・スタンバイ・データベースのV\$DATABASE固定ビューのDATABASE_ROLE列をチェックします。

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

- 列にPRIMARYの値が含まれる場合、スイッチオーバー操作は完了していますが、スイッチオーバー後のタスクで障害が発生しています。この場合、次のステップを実行します。
 - a. データベースを停止して、再オープンします。
 - b. ALTER DATABASE GUARD NONEコマンドを発行して、データベースに対する書き込み制限を削除します。
- 列にLOGICAL STANDBYの値が含まれる場合、ステップ2に進みます。

2. ターゲット・ロジカル・スタンバイで次の問合せを実行します。

```
SQL> SELECT COUNT(*) FROM SYSTEM.LOGSTDBY$PARAMETERS -
> WHERE NAME = 'BEGIN_PRIMARY';
```

- 問合せで0が戻される場合、ロジカル・スタンバイの状態はスイッチオーバーのコミット・コマンドが発行される前と同一です。修正処理を実行する必要はありません。スイッチオーバー操作のコミットに進むか、[「ロジカル・スタンバイ・データベースの準備中の障害」](#)で説明するようにスイッチオーバー操作を取り消します。

- 問合せで1が返される場合、ロジカル・スタンバイは一貫性がない状態です。ステップ3に進んでください。

3. 新しいプライマリまたは別の新しいプライマリに対するバイスタンダになるための機能が維持されるよう、ロジカル・スタンバイで対処措置を実行します。

スイッチオーバー操作のコミット中に発生したエラーの基となる原因を修正してSQL文(ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY)を再発行します。次のステップを実行して、スイッチオーバーのコミットを試行する直前の一貫性の時点までロジカル・スタンバイ・データベースをフラッシュバックすることもできます。

- スイッチオーバーのコミット・コマンドを開始したインスタンスのアラート・ログから、ロジカル・スタンバイへのフラッシュバックに必要なSCNを判別します。この情報は、SQL文ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARYの後に表示されます。

```
LOGSTDBY: Preparing the COMMIT TO SWITCHOVER TO PRIMARY DDL at scn [flashback_scn].
```

- ターゲット・スタンバイ・データベースのすべてのインスタンスを停止します。

```
SQL> SHUTDOWN IMMEDIATE;
```

- ターゲット・ロジカル・スタンバイ・データベースをマウントします。

```
SQL> STARTUP MOUNT;
```

- 必要なSCNまでターゲット・ロジカル・スタンバイをフラッシュバックします。

```
SQL> FLASHBACK DATABASE TO BEFORE SCN <flashback_scn>;
```

- データベースをオープンします(Oracle RACの場合、すべてのインスタンスをオープンします)。

```
SQL> STARTUP OPEN;
```

この時点で、ターゲット・スタンバイの状態はスイッチオーバーのコミット・コマンドが発行される前と同一です。対処措置を実行する必要はありません。スイッチオーバー操作のコミットに進みます。

A.6 SQL Applyが停止した場合の処置

サポートされない文やパッケージが検出されると、SQL Applyは停止します。

適用サービスでは、サポートされないDML文、DDL文およびオラクル社が提供するパッケージを、SQL Applyが実行されているロジカル・スタンバイ・データベースに適用できません。

SQL Applyが、サポートされない文やパッケージのために停止した場合は、問題を修正し、[表A1](#)で説明されているアクションを実行して、ロジカル・スタンバイ・データベースでSQL Applyを再開できます。

表A-1 一般的なSQL Applyエラーの解決方法

エラー	解決方法
サポートされない文またはオラクル社が提供するパッケージが検出された可能性を示すエラー	DBA_LOGSTDBY_EVENTS ビューで、最後の文を検索してください。SQL Apply の失敗の原因となった文とエラーが表示されています。不適切な SQL 文が原因で SQL Apply に失敗した場合は、その文とエラーに関する情報とともにトランザクション情報を表示できます。このトランザクション情報は、問題の原因を正確に判断するために LogMiner ツールで使用できます。

エラー	解決方法
データベース管理を要求するエラー(特定の表領域内の領域不足など)	問題を解決した後、ALTER DATABASE START LOGICAL STANDBY APPLY 文を使用して SQL Apply を再開してください。
不適切な SQL 文の入力によるエラー(不適切なスタンバイ・データベースのファイル名が表領域文に入力された場合など)	適切な SQL 文を入力した後、DBMS_LOGSTDBY.SKIP_TRANSACTION プロシージャを使用して、次回 SQL Apply が実行されたときに不適切な文が無視されるようにしてください。その後、ALTER DATABASE START LOGICAL STANDBY APPLY 文を使用して SQL Apply を再開してください。
不適切な SKIP パラメータの設定によるエラー(特定の表で全 DML がスキップされるように指定したが、CREATE、ALTER および DROP TABLE の各文がスキップされるように指定していないなど)	DBMS_LOGSTDBY.SKIP('TABLE', 'schema_name', 'table_name', null) プロシージャを発行した後、SQL Apply を再開してください。
障害の原因を特定するためにDBA_LOGSTDBY_EVENTSビューを問い合わせる方法は、 「Oracle Data Guardに関連するビュー」 を参照してください。	

A.7 REDOデータ転送のネットワーク調整

最適なパフォーマンスを得るには、REDO転送サービスで使用する各Oracle Net接続記述子で、Oracle Net SDUパラメータを最大値の65535バイトに設定します。

次の例は、データベース初期化パラメータ・ファイル内のリモート宛先netservを定義する部分を示します。

```
LOG_ARCHIVE_DEST_3='SERVICE=netserv'
```

次の例は、tnsnames.oraファイル内の該当するサービス名の定義を示しています。

```
netserv=(DESCRIPTION=(SDU=32768)(ADDRESS=(PROTOCOL=tcp)(HOST=host)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=srvc)))
```

次の例は、listener.oraファイル内の定義を示します。

```
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
(HOST=host)(PORT=1521))))

SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(SDU=32768)(SID_NAME=sid)
(GLOBALDBNAME=srvc)(ORACLE_HOME=/oracle)))
```

待機時間が長いまたは帯域幅が大きいネットワーク・リンクを使用してリモート・サイトへのアーカイブを行う場合は、Oracle Netプロファイル・パラメータのSQLNET.SEND_BUF_SIZEおよびSQLNET.RECV_BUF_SIZEを使用して、ネットワークの送受信I/Oバッファのサイズを大きくすることでパフォーマンスを改善できます。

Oracle NET SDUパラメータのその他の変更方法の詳細は、『[Oracle Database Net Services管理者ガイド](#)』を参照してください。

A.8 スタンバイ・データベースのディスクのパフォーマンスが遅い

ファイル・システム上の非同期I/Oがパフォーマンス上の問題を示している場合は、ダイレクトI/Oオプションを使用してファイル・シ

ステムをマウントするか、またはFILESYSTEMIO_OPTIONS=SETALL初期化パラメータを設定します。

最大I/Oのサイズ設定は、1MBです。

A.9 プライマリ・データベースの停止を回避するにはログ・ファイルを一致させる必要がある

構成内の1つ以上のスタンバイ・データベースでスタンバイREDOログを構成した場合は、各スタンバイ・データベースのスタンバイREDOログ・ファイルのサイズが、プライマリ・データベースのオンラインREDOログ・ファイルのサイズと正確に一致していることを確認してください。

ログ・スイッチが発生したときに、プライマリ・データベースに新しい現行のオンラインREDOログ・ファイルのサイズと一致する使用可能なスタンバイREDOログ・ファイルがない場合、次のようになります。

- プライマリ・データベースが最大保護モードで作動しているときは停止します。

または

- スタンバイ・データベース上のRFSプロセスにより、スタンバイ・データベースにアーカイブREDOログ・ファイルが作成され、次のメッセージがアラート・ログに書き込まれます。

```
No standby log files of size <#> blocks available.
```

たとえば、プライマリ・データベースで、ログ・ファイルが100KのオンラインREDOログ・グループを2つを使用する場合、スタンバイ・データベースは、ログ・ファイル・サイズが100KのスタンバイREDOログ・グループを3つ持ちます。

また、REDOログ・グループをプライマリ・データベースに追加するたびに、対応するスタンバイREDOログ・グループをスタンバイ・データベースに追加する必要があります。これによって必要なサイズのスタンバイREDOログ・ファイルがログ・スイッチ時に使用できないことにより、プライマリ・データベースに悪影響が発生する可能性を軽減できます。

A.10 ロジカル・スタンバイ・データベースのトラブルシューティング

エラーのリカバリに役立つトラブルシューティングのヒントを次に示します。

- [エラーのリカバリ](#)
- [SQL*Loaderセッションのトラブルシューティング](#)
- [長時間実行トランザクションのトラブルシューティング](#)
- [フラッシュバック・トランザクションでのORA-1403エラーのトラブルシューティング](#)

A.10.1 エラーのリカバリ

ロジカル・スタンバイ・データベースでは、ユーザー表、順序およびジョブがメンテナンスされます。他のオブジェクトをメンテナンスするには、REDOデータ・ストリームにあるDDL文を再発行する必要があります。

SQL Applyに障害が発生した場合、エラーはDBA_LOGSTDBY_EVENTS表に記録されます。次の各項では、このようなエラーのリカバリ方法を説明します。

A.10.1.1 ファイル仕様が含まれているDDLトランザクション

DDL文は、プライマリ・データベースとロジカル・スタンバイ・データベースでは同じ方法で実行されます。

基礎となるファイル構造が両方のデータベースで同じ場合、DDLはスタンバイ・データベース上で予想どおりに実行されます。

エラーの原因が、ロジカル・スタンバイ・データベース環境に適合しないファイル仕様を含んだDDLトランザクションにある場合は、次のステップを実行して問題を解決してください。

1. ALTER SESSION DISABLE GUARD文を使用してデータベース・ガードをバイパスし、ロジカル・スタンバイ・データベースへの変更を可能にします。

```
SQL> ALTER SESSION DISABLE GUARD;
```

2. 正しいファイル仕様を使用してDDL文を実行し、データベース・ガードを再び使用可能にします。次に例を示します。

```
SQL> ALTER TABLESPACE t_table ADD DATAFILE '/dbs/t_db.f' SIZE 100M REUSE;  
SQL> ALTER SESSION ENABLE GUARD;
```

3. ロジカル・スタンバイ・データベースでSQL Applyを開始し、障害が発生したトランザクションをスキップします。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE -  
> SKIP FAILED TRANSACTION;
```

トランザクション障害の原因となった問題を修正し、トランザクションをスキップせずにSQL Applyを再開できる場合があります。たとえば、使用可能領域がすべて使用された場合などです。(DDLトランザクションをスキップするときに、プライマリおよびロジカル・スタンバイ・データベースの内容にずれを生じさせないようにしてください。可能な場合は、スキップされたトランザクションのかわりに補足用トランザクションを手動で実行します。)

次の例に、SQL Applyの停止、エラーの修正およびSQL Applyの再開を示します。

```
SQL> SET LONG 1000
```

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

Session altered.

```
SQL> SELECT EVENT_TIME, COMMIT_SCN, EVENT, STATUS FROM DBA_LOGSTDBY_EVENTS;
```

EVENT_TIME	COMMIT_SCN

EVENT

STATUS

22-OCT-03 15:47:58

ORA-16111: log mining and apply setting up

22-OCT-03 15:48:04	209627

insert into "SCOTT"."EMP"
values

```
"EMPNO" = 7900,  
"ENAME" = 'ADAMS',  
"JOB" = 'CLERK',  
"MGR" IS NULL,  
"HIREDATE" = TO_DATE('22-OCT-03', 'DD-MON-RR'),  
"SAL" = 950,  
"COMM" IS NULL,  
"DEPTNO" IS NULL
```

ORA-01653: unable to extend table SCOTT.EMP by %200 bytes in tablespace T_TABLE

この例で、ORA-01653メッセージは、表領域がいっぱいでも表を拡張できないことを示しています。問題を修正するには、新しいデータファイルを表領域に追加します。次に例を示します。

```
SQL> ALTER TABLESPACE t_table ADD DATAFILE '/dbs/t_db.f' SIZE 60M;
```

```
Tablespace altered.
```

次に、SQL Applyを再開します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;  
Database altered.
```

SQL Applyを再開すると、障害が発生したトランザクションが再実行され、ロジカル・スタンバイ・データベースに適用されます。

A.10.1.2 DML障害のリカバリ

イベント表に表示されるDMLのみでなく、トランザクションに関連するすべてのDMLがスキップされるので、DML障害のフィルタ処理にSKIP_TRANSACTIONプロシージャは使用しないでください。

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て。次のステップは、この問題に応答する1つの方法を示しています。

1. 表をスキップ・リストに追加することによって、トランザクションではなく、表をバイパスします。

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP('DML', 'SCOTT', 'EMP');  
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

以降は、SCOTT. EMP表のDMLアクティビティは適用されません。DBMS_LOGSTDBYパッケージ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・データベースにデータベース・リンクを設定している場合は、記憶域の問題を修正後、表を修正することができます。

2. プライマリ・データベースへのデータベース・リンクを使用して、ローカルなSCOTT. EMP表を削除し、表を再作成して、データをスタンバイ・データベースに転送します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;  
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE('SCOTT', 'EMP', 'PRIMARYDB');  
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

3. 新規にインスタンス化された表とデータベースの残りの部分にまたがってビューの一貫性が得られるように、SQL Applyがプライマリ・データベースと一致するまで待機してから、この表を問い合わせます。詳細な例は、[「ロジカル・スタンバイ・データベースでの表の追加または再作成」](#)を参照してください。

A.10.2 SQL*Loaderセッションのトラブルシューティング

Oracle SQL*Loaderには、様々なソースからOracle Databaseにデータをロードする方法が用意されています。

このトピックでは、SQL*Loaderユーティリティの一部のSQL Apply関連機能について説明します。

選択したデータ・ロード方法に関係なく、SQL*Loader制御ファイルには、APPENDおよびREPLACEキーワードを介して、新規データのロード先となるOracle表の現在の内容に対して実行する作業を示す指示が含まれています。次の例に、これらのキーワードをLOAD_STOKという表に使用する方法を示します。

- APPENDキーワードを使用すると、新規にロードされるデータはLOAD_STOK表の内容に追加されます。

```
LOAD DATA  
INTO TABLE LOAD_STOK APPEND
```

- REPLACEキーワードを使用すると、LOAD_STOK表の内容が削除されてから新規のデータがロードされます。SQL*Loaderでは、DELETE文を使用して表の内容が1つのトランザクションでページされます。

```
LOAD DATA  
INTO TABLE LOAD_STOK REPLACE
```

SQL*LoaderスクリプトにREPLACEキーワードを使用するかわりに、データのロード前に、プライマリ・データベースの表に対してSQL*Plus TRUNCATE TABLEコマンドを発行することをお勧めします。TRUNCATE TABLEコマンドがオンラインのREDOログ・ファイルに記録され、ロジカル・スタンバイ・データベース上のSQL Applyによって発行されるため、これによってプライマリとスタンバイの両方のデータベースの表コピーを、高速に、そして効率よくパージするのと同じ効果が得られます。

SQL*Loaderスクリプトに引き続きREPLACEキーワードを挿入できますが、プライマリ・データベースではオブジェクトからの0 (ゼロ)行のDELETEが試行されます。プライマリ・データベースから行が削除されていないため、REDOログ・ファイルにREDOは記録されません。したがって、ロジカル・スタンバイ・データベースに対してDELETE文は発行されません。

SQL文TRUNCATE TABLEを使用せずにREPLACEキーワードを発行すると、トランザクションをロジカル・スタンバイ・データベースに適用する必要がある場合に、SQL Applyに次のような問題が発生する可能性があります。

- 現在、表に多数の行が含まれている場合は、その行をスタンバイ・データベースから削除する必要があります。SQL Applyでは文の元の構文を判別できないため、プライマリ・データベースからパージする行ごとにDELETE文を発行する必要があります。

たとえば、プライマリ・データベースの表に最初は10,000行含まれていた場合、Oracle SQL*Loaderは1つのDELETE文を発行して10,000行をパージします。スタンバイ・データベースでは、SQL Applyはすべての行がパージされることを認識しないかわりに、10,000のDELETE文を個別に発行し、それぞれの文で1行ずつパージする必要があります。

- スタンバイ・データベースの表にSQL Applyで使用可能な索引が含まれていない場合、DELETE文では情報をパージするために全表スキャンが発行されます。

前述の例では、SQL Applyは10,000のDELETE文を個別に発行しているため、スタンバイ・データベースに対して10,000の全表スキャンが発行される可能性があります。

A.10.3 長時間実行トランザクションのトラブルシューティング

SQL Apply環境における長時間実行トランザクションの主な原因の1つは、全表スキャンです。

また、索引の作成時や再作成時のように、SQL文がスタンバイ・データベースに複製されるために、長時間実行トランザクションが発生することもあります。

長時間実行トランザクションの識別

SQL Applyで1つのSQL文が長時間実行されている場合は、SQL Applyインスタンスのアラート・ログに次のような警告メッセージがレポートされます。

```
Mon Feb 17 14:40:15 2003
WARNING: the following transaction makes no progress
WARNING: in the last 30 seconds for the given message!
WARNING: xid =
0x0016.007.000017b6 cscn = 1550349, message# = 28, slavid = 1
knacrb: no offending session found (not ITL pressure)
```

この警告メッセージの注意事項は次のとおりです。

- この警告は、Interested Transaction List(ITL)が不十分な場合に戻される警告メッセージに似ていますが、最終行がknacrbで始まります。この最終行は、次のことを示しています。
 - 全表スキャンが発生している可能性がある
 - この発行では、Interested Transaction List(ITL)が不十分な状態に対して何も実行されない
- この警告メッセージがレポートされるのは、1つの文の実行時間が30秒を超える場合のみです。

長時間実行文で実行されているSQL文を判別できない場合がありますが、次のSQL文を発行すると、SQL Applyが作業中のデータベース・オブジェクトを識別できます。

```
SQL> SELECT SAS.SERVER_ID -  
> , SS.OWNER -  
> , SS.OBJECT_NAME -  
> , SS.STATISTIC_NAME -  
> , SS.VALUE -  
> FROM V$SEGMENT_STATISTICS SS -  
> , V$LOCK L -  
> , V$STREAMS_APPLY_SERVER SAS -  
> WHERE SAS.SERVER_ID = &SLAVE_ID -  
> AND L.SID = SAS.SID -  
> AND L.TYPE = 'TM' -  
> AND SS.OBJ# = L.ID1;
```

また、次のSQL文を発行すると、実行ごとに多数のディスク読み取りを発生させたSQL文を識別できます。

```
SQL> SELECT SUBSTR(SQL_TEXT,1,40) -  
> , DISK_READS -  
> , EXECUTIONS -  
> , DISK_READS/EXECUTIONS -  
> , HASH_VALUE -  
> , ADDRESS -  
> FROM V$SQLAREA -  
> WHERE DISK_READS/GREATEST(EXECUTIONS,1) > 1 -  
> AND ROWNUM < 10 -  
> ORDER BY DISK_READS/GREATEST(EXECUTIONS,1) DESC;
```

すべての表にプライマリ・キーの制約を定義すると、列が自動的にNOT NULLとして定義されるので、この方法をお薦めしています。プライマリ・キーの制約を定義できない表については、該当する列で索引をNOT NULLと定義します。該当する列が表に存在しない場合は表を確認し、可能であればSQL Applyをスキップします。次のステップでは、SCOTTスキーマのFTS表に対して発行されたすべてのDML文をスキップする方法を説明します。

1. SQL Applyを停止します。

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

2. SCOTT.FTS表について、すべてのDMLトランザクションのスキップ・プロシーダを構成します。

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP(stmt => 'DML' , -  
> schema_name => 'SCOTT' , -  
> object_name => 'FTS');  
PL/SQL procedure successfully completed
```

3. SQL Applyを開始します。

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

ITLが不十分な状態のトラブルシューティング

Interested Transaction List(ITL)が不十分な状態は、SQL Applyインスタンスのアラート・ログでレポートされます。次に警告メッセージの例を示します。

```
Tue Apr 22 15:50:42 2003  
WARNING: the following transaction makes no progress  
WARNING: in the last 30 seconds for the given message!  
WARNING: xid =
```


リアルタイム適用の分析

上述の出力のメッセージは、SQL Applyプロセス(slavid)#17が過去30秒にわたって進捗がなかったことを示しています。適用プロセスにより発行されているSQL文を判別するには、次の問合せを発行します。

```
SQL> SELECT SA.SQL_TEXT -
> FROM V$SQLAREA SA -
> , V$SESSION S -
> , V$STREAMS_APPLY_SERVER SAS -
> WHERE SAS.SERVER_ID = &SLAVEID -
> AND S.SID = SAS.SID -
> AND SA.ADDRESS = S.SQL_ADDRESS
```

SQL_TEXT

```
insert into "APP"."LOAD_TAB_1" p("PK","TEXT") values (:1, :2)
```

ITLが不十分な状態を識別するには、次の例に示すようにV\$LOCKビューを問い合わせる方法もあります。TXロックのリクエスト値が4のセッションは、ITLが使用可能になるまで待機中です。

```
SQL> SELECT SID, TYPE, ID1, ID2, LMODE, REQUEST -
> FROM V$LOCK -
> WHERE TYPE = 'TX'
```

SID	TY	ID1	ID2	LMODE	REQUEST
8	TX	327688	48	6	0
10	TX	327688	48	0	4

この例では、SID 10はSID 8が保持しているTXロックを待機中です。

障害後のレビュー

セグメントのITLが不十分な状態が長時間続くことはまずありません。また、不十分な状態を継続する時間が30秒未満のITLは、スタンバイ・データベースのアラート・ログではレポートされません。したがって、ITLが不十分な状態になっているオブジェクトを判別するには、次の文を発行します。

```
SQL> SELECT OWNER, OBJECT_NAME, OBJECT_TYPE -
> FROM V$SEGMENT_STATISTICS -
> WHERE STATISTIC_NAME = 'ITL waits' -
> AND VALUE > 0 -
> ORDER BY VALUE;
```

この文では、インスタンスが前回起動された後にITLが不十分な状態になったことのあるデータベース・セグメントがすべてレポートされます。

ノート:



この SQL 文は、Oracle Data Guard 環境のロジカル・スタンバイ・データベースに限定されません。任意の Oracle データベースに適用されます。

ITLが不十分な状態の解消

特定のデータベース・オブジェクトのINITRANSの整数を増やすには、最初にSQL Applyを停止する必要があります。

関連項目:

データベース・オブジェクトに割り当てられた各データ・ブロック内に割り当てられた同時トランザクション・エントリの初期数である、INITRANS整数指定の詳細は、『[Oracle Database SQL言語リファレンス](#)』を参照してください。

次の例に、appスキーマ内の表load_tab_1のINITRANSを増やすためのステップを示します。

1. SQL Applyを停止します。

```
SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY;  
Database altered.
```

2. データベース・ガードを一時的にバイパスします。

```
SQL> ALTER SESSION DISABLE GUARD;  
Session altered.
```

3. スタンバイ・データベースでINITRANSを増やします。次に例を示します。

```
SQL> ALTER TABLE APP.LOAD_TAB_1 INITRANS 30;  
Table altered
```

4. データベース・ガードを再び使用可能にします。

```
SQL> ALTER SESSION ENABLE GUARD;  
Session altered
```

5. SQL Applyを開始します。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;  
Database altered.
```

また、スイッチオーバー時に新しいスタンバイ・データベースでエラーが発生しないように、プライマリ・データベースでデータベース・オブジェクトを変更することを検討してください。

A.10.4 フラッシュバック・トランザクションでのORA-1403エラーのトラブルシューティング

SQL Applyが「ORA-1403: データが見つかりません」エラーを戻した場合は、失われたデータを再構築するためにフラッシュバック・トランザクションを使用することができます。

この戦略は、スタンバイ・データベース・インスタンスで指定されたUNDO_RETENTION初期化パラメータに依存します。

通常は、ORA-1403エラーはロジカル・スタンバイ・データベース環境では発生しません。SQL Applyが管理している表内のデータがスタンバイ・データベースで直接変更され、同じデータがプライマリ・データベースで変更された場合にこのエラーが発生します。変更されたデータがプライマリ・データベースで更新され、その後ロジカル・スタンバイ・データベースで受信した場合、SQL Applyがレコードの更新前に、データの元のバージョンがスタンバイ・データベースに存在していることを検証します。この検証に失敗した場合は、「ORA-1403: データが見つかりません」エラーは戻されません。

初期エラー

SQL Applyの検証に失敗した場合、エラー・メッセージがロジカル・スタンバイ・データベースのアラート・ログにレポートされ、DBA_LOGSTDBY_EVENTSビューにレコードが挿入されます。アラート・ログ内の情報は切り捨てられますが、データベース・ビューではエラー全体がレポートされます。次に例を示します。

```
LOGSTDBY stmt: UPDATE "SCOTT"."MASTER"
SET
  "NAME" = 'john'
WHERE
  "PK" = 1 and
  "NAME" = 'andrew' and
  ROWID = 'AAAAAAAAEAAAAAPAAA'
LOGSTDBY status: ORA-01403: no data found
LOGSTDBY PID 1006, oracle@staco03 (P004)
LOGSTDBY XID 0x0006.00e.00000417, Thread 1, RBA 0x02dd.00002221.10
```

調査

最初のステップは、エラーの原因となった表の履歴データを分析することです。そのためには、SELECT文のVERSIONS句を使用します。たとえば、プライマリ・データベースで次の問合せを発行できます。

```
SELECT VERSIONS_XID
       , VERSIONS_STARTSCN
       , VERSIONS_ENDSCN
       , VERSIONS_OPERATION
       , PK
       , NAME
FROM SCOTT.MASTER
     VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE PK = 1
ORDER BY NVL (VERSIONS_STARTSCN, 0);
```

VERSIONS_XID	VERSIONS_STARTSCN	VERSIONS_ENDSCN	V	PK	NAME
03001900EE070000	3492279	3492290	I	1	andrew
02000D00E4070000	3492290		D	1	andrew

データベースで構成されているUNDO保存の期間(UNDO_RETENTION)および表に対するアクティビティによっては、返される情報が広範囲で、返される情報を制限するVERSIONS BETWEEN構文の変更が必要な場合があります。返された情報から、レコードが最初にSCN 3492279に挿入され、SCN 3492290でトランザクションID 02000D00E4070000の一部として削除されたことがわかります。トランザクションIDを使用して、データベースを問い合わせでトランザクションの範囲を確認します。これはFLASHBACK_TRANSACTION_QUERYビューの問合せによって行うことができます。

```
SELECT OPERATION
       , UNDO_SQL
FROM FLASHBACK_TRANSACTION_QUERY
WHERE XID = HEXTORAW('02000D00E4070000');
```

OPERATION	UNDO_SQL
DELETE	insert into "SCOTT"."MASTER" ("PK", "NAME") values ('1', 'andrew');
BEGIN	

常に、トランザクションの開始を表す1行が返されます。このトランザクションでは、マスター表から削除されたのは1行のみです。実行時のUNDO_SQL列では、元のデータが表にリストアされます。

```
SQL> INSERT INTO "SCOTT"."MASTER" ("PK", "NAME") VALUES ('1', 'ANDREW'); SQL> COMMIT;
```

SQL Applyを再開すると、トランザクションがスタンバイ・データベースに適用されます。

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

B Oracle Data Guard構成におけるデータベースのパッチ適用、アップグレードおよびダウングレード

次の各項では、Oracle Data Guard構成にフィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースが存在する場合に、Oracleデータベースをアップグレードおよびダウングレードする方法について説明します。

- [Oracle Databaseソフトウェアをアップグレードする前の注意事項](#)
- [Standby-First Patchの適用を使用したOracle Databaseへのパッチ適用](#)
- [フィジカル・スタンバイ・データベースが存在する場合のOracle Databaseのアップグレード](#)
- [ロジカル・スタンバイ・データベースが存在する場合のOracle Databaseのアップグレード](#)
- [アップグレード後のCOMPATIBLE初期化パラメータの変更](#)
- [ロジカル・スタンバイが存在しない場合のOracle Databaseのダウングレード](#)
- [ロジカル・スタンバイが存在する場合のOracle Databaseのダウングレード](#)

B.1 Oracle Databaseソフトウェアをパッチ適用またはアップグレードする前の注意事項

Oracle Databaseソフトウェアへのパッチ適用またはアップグレードを開始する前に、いくつかの考慮事項があります。

- Oracle Data Guardブローカを使用して構成を管理している場合は、ブローカ構成の削除や無効化の詳細は、[『Oracle Data Guard Broker』](#)マニュアルの指示を参照してください。
- これらの項で説明する手順は、[『Oracle Databaseアップグレード・ガイド』](#)に記載されている手順とともに使用します。
- NOLOGGING操作をチェックします。NOLOGGING操作が実行されていた場合は、スタンバイ・データベースを更新する必要があります。詳細は、[\[NOLOGGING句を指定した後のリカバリ\]](#)を参照してください。
- OFFLINE IMMEDIATEが原因でリカバリを必要とする表領域またはデータファイルをノートにとっておきます。アップグレード前に表領域またはデータファイルをリカバリし、オンラインまたはオフラインにする必要があります。
- Oracle Data Guard構成では、すべてのフィジカルおよびスナップショット・スタンバイ・データベースはプライマリ・データベースからのパスワード・ファイルのコピーを使用する必要があります。Oracle Database 12cリリース2 (12.2.0.1)以降、プライマリ・データベースで行われたパスワード・ファイルの変更は、スタンバイ・データベースに自動的に伝搬されます。(管理権限(SYSDBA、SYSOPER、SYSDBAなど)が付与または解除されたとき、または管理権限を持つユーザーのパスワードが変更されたときの、パスワード・ファイルの変更が含まれます。)

遠隔同期インスタンスは、自動更新機能の例外です。遠隔同期インスタンスはREDOを受け取りますが、適用しないため、更新したパスワード・ファイルを引き続き手動で遠隔同期インスタンスにコピーする必要があります。遠隔同期インスタンスでパスワード・ファイルが手動で更新されると、プライマリ・データベースからの同じパスワード変更内容を含むREDOが、遠隔同期インスタンスからREDOを受け取るように設定されているすべてのスタンバイ・データベースに自動的に伝播されます。REDOが適用されると、スタンバイでパスワード・ファイルが更新されます。

ノート:



カスケード・スタンバイが構成にある場合、これらのカスケード・スタンバイは他のスタンバイと同じルールに従う必要

がありますが、最後に停止し、最初に新しいホームで再起動します。

B.2 Standby-First Patchの適用を使用したOracle Databaseへのパッチ適用

Data Guard Standby-First Patchの適用を使用すると、プライマリ・データベースとそのフィジカル・スタンバイ・データベース間で異なるデータベース・ホーム・ソフトウェアがサポートされます。

このサポートは、プライマリ・データベースへのリスクを最小限に抑えるローリング方式で、Oracleのパッチおよびパッチ・バンドルを適用および検証する目的で提供されています。たとえば、Data Guard Standby-First Patchの適用を使用して、最初にデータベース・ホーム・パッチをフィジカル・スタンバイ・データベースに適用します。パッチをテストして評価するために、そのスタンバイを使用して読取り専用ワークロードまたは、スナップショット・スタンバイの場合は読取り/書き込みワークロードを実行します。評価を通ると、データベース・ホーム・パッチの有効性と安定性の保証が強くなるので、パッチをプライマリ・システムにインストールします。

Oracle Data Guard Standby-First Patchの適用は、Oracle Engineered Systems (Exadata、SuperCluster など)とEngineered Systems以外の両方で、Oracle Database 11.2.0.1以降の認定された個別パッチおよびパッチ・バンドル(Exadataに対するパッチ・セット更新やデータベース・パッチなど)に対してのみサポートされます。Data Guard Standby-Firstによって認定されたパッチおよびパッチ・バンドルは、パッチのREADMEで次のように提示しています。

Data Guard Standby-Firstでインストール可能

次のタイプのパッチは、Data Guard Standby-Firstによって認定される候補です。

- データベース・ホームの個別パッチ
- Exadataのバンドル・パッチ(Exadataの月および半期ごとのデータベース・パッチ)
- データベースのパッチ・セット更新

異なるデータベース・ホーム・ソフトウェアを実行しているプライマリ・システムとフィジカル・スタンバイ・システム間の相互運用性を破壊する可能性があるモジュールを更新するパッチおよびパッチ・バンドルは、"Data Guard Standby-Firstでインストール可能"と認定されず、パッチのREADMEにそのように提示していません。

Oracleパッチ・セットおよびメジャー・リリース・アップグレードは、Data Guard Standby-First Patchの適用の対象にはなりません。たとえば、11.2.0.2から11.2.0.3へのアップグレードや11.2から12.1へのアップグレードは対象外です。データベース・パッチ・セットおよびメジャー・リリースには、Data Guard一時ロジカル・スタンバイ・ローリング・アップグレード・プロセスを使用します。

さらにOracle Database 11gリリース2 (11.2.0.1)現在、フィジカル・スタンバイ・データベースを使用して、適格な個別パッチ、パッチ・セット更新(PSU)およびクリティカル・パッチ・アップデート(CPU)をローリングという方法でインストールできます。この機能の詳細は、<http://support.oracle.com>のMy Oracle Supportノート1265700.1を参照してください。

B.3 フィジカル・スタンバイ・データベースが存在する場合のOracle Databaseのアップグレード

これらのステップでは、構成にフィジカル・スタンバイ・データベースが存在する場合に、Oracle Database 12c リリース2 (12.2)にアップグレードする方法について説明します。

- 1.

ノート:



アップグレードするデータベースが Oracle Data Guard Broker 構成のメンバーである場合は、続行する前にファスト・スタート・フェイルオーバーを無効化し、ブローカを停止する必要があります。これを実行する方法の詳細は、[Oracle Data Guard Broker](#) を参照してください。『[Oracle Database アップグレード・ガイド](#)』のアップグレードの準備に関する項に記載されているステップを確認して実行します。

2. 『[Oracle Database アップグレード・ガイド](#)』に従って、Oracleソフトウェアの新しいリリースを、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・システムの新しいOracleホームにインストールします。
3. プライマリ・データベースを停止します。
4. フィジカル・スタンバイ・データベースを停止します。
5. アップグレード対象のOracleホームで実行しているすべてのリスナー、エージェントおよびその他のプロセスを停止します。このステップは、Oracle Real Application Cluster(Oracle RAC)環境のすべてのノードで実行します。
6. Oracle Automatic Storage Management(Oracle ASM)が使用中の場合は、Oracle ASMを使用するすべてのデータベースを停止してから、すべてのOracle ASMインスタンスを停止します。
7. 新しいOracleホームで、ステップ5で停止したすべてのリスナー、エージェントおよびその他のプロセスを再起動します。
8. フィジカル・スタンバイ・データベースを、新しいOracleホーム(アップグレード済バージョン)にマウントします。フィジカル・スタンバイ・データベースを起動する方法については、[「フィジカル・スタンバイ・データベースの起動」](#)を参照してください。

ノート:



プライマリ・データベースのアップグレードが完了するまで、スタンバイ・データベースをオープンしないでください。

9. フィジカル・スタンバイ・データベースでREDO Applyを開始します。Redo Applyを開始する方法については、[「フィジカル・スタンバイ・データベースの起動」](#)を参照してください。
10. 『[Oracle Database アップグレード・ガイド](#)』に従ってプライマリ・データベースをアップグレードします。フィジカル・スタンバイ・データベースは、プライマリ・データベースでアップグレード時に生成されたREDOを適用するとアップグレードされます。
11. アップグレード後のプライマリ・データベースをオープンします。
12. アップグレードの前にOracle Active Data Guardが使用されている場合、これをアップグレード後に再び有効にする方法は[「リアルタイム問合せ」](#)を参照してください。
13. オプションで、[「アップグレード後のCOMPATIBLE初期化パラメータの変更」](#)で説明されている手順に従い、COMPATIBLE初期化パラメータを変更します。

ノート:



Windows プラットフォームでは、ORADIM ユーティリティを使用して(旧データベース・バージョンの)データベース・サービスを削除し、新規データベース・バージョンの新規データベース・サービスを作成する必要があります。プライマリ・サーバーとスタンバイ・サーバーの両方で、OracleService<SID>を置換する必要があります。

B.4 ロジカル・スタンバイ・データベースが存在する場合のOracle Databaseのアップグレード

このステップでは、ロジカル・スタンバイ・データベースが存在する場合にOracle Databaseソフトウェアをアップグレードする従来の方法について説明します。

ノート:



「[SQL Apply を使用した Oracle Database のアップグレード](#)」では、ロジカル・スタンバイ・データベースが存在する場合にローリング・アップグレードを行って停止時間を最短に抑える別の方法を説明します。完全なアップグレードを実行するには、どちらか一方の方法のステップのみを使用してください。アップグレードの実行に両方の方法を使用したり、2 つの方法のステップを組み合わせようとしないでください。

構成にロジカル・スタンバイ・データベースが存在する場合に、Oracle Database 12c リリース2 (12.2)にアップグレードするステップは、次のとおりです。次のステップでは、プライマリ・データベースがMAXIMUM PERFORMANCEデータ保護モードで稼働していると仮定しています。

[Oracle Databaseアップグレード・ガイド](#)のアップグレードの準備に関する項に記載されているステップを確認して実行します

1. 必要に応じて、プライマリ・データベースでデータ保護モードをMAXIMUM PERFORMANCEに設定します。

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

2. プライマリ・データベースで、すべてのユーザー・アクティビティを停止し、ロジカル・スタンバイ・データベースに関連付けられているリモート・アーカイブ先を遅延させます(この手順では、LOG_ARCHIVE_DEST_2がロジカル・スタンバイ・データベースに関連付けられていると仮定しています)。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=DEFER SCOPE=BOTH;  
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

3. ロジカル・スタンバイ・データベースでSQL Applyを停止します。

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

4. プライマリ・データベースに、Oracleソフトウェアの新しいリリースをインストールします。
5. ロジカル・スタンバイ・データベースに、Oracleソフトウェアの新しいリリースをインストールします。

ノート:



ステップ 4 およびステップ 5 は、アップグレード手順中の停止時間を短縮するために、同時に実行できます(つまり、プライマリ・データベースとスタンバイ・データベースは同時にアップグレードできます)。

6. アップグレード後のロジカル・スタンバイ・データベースでSQL Applyを開始します。Oracle RACを使用している場合は、他のスタンバイ・データベース・インスタンスを起動します。

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

7. アップグレード後のプライマリ・データベースをオープンし、ユーザーが接続できるようにします。Oracle RACを使用してい

る場合は、他のプライマリ・データベース・インスタンスを起動します。

また、次のようにして、アップグレード後のロジカル・スタンバイ・データベースへのアーカイブを有効にします。

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

8. 必要に応じて、ステップ1で元のデータ保護モードを変更した場合は、そのモードをリセットします。

9. 必要に応じて、COMPATIBLE初期化パラメータを変更します。

関連トピック

- [Oracle Databaseアップグレード・ガイド](#)

B.5 アップグレード後のCOMPATIBLE初期化パラメータの変更

Oracle Databaseの新リリースにアップグレードする際、一部の新機能のためにデータベースが前のリリースと互換性がなくなる場合があります。

Oracle Databaseでは、COMPATIBLE初期化パラメータにより、データベースの互換性を制御できます。

アップグレードが完了したら、COMPATIBLE初期化パラメータの設定をOracle Databaseの新リリース用に最大レベルに増やすことができます。データベースを元のリリースにダウングレードする機能がもう必要ないことが確実な場合は、COMPATIBLE初期化パラメータを新しいデータベースで必要な互換性レベルに基づいて設定します。

Oracle Data Guard構成で、アップグレード後にCOMPATIBLE初期化パラメータの設定を増やすように決定した場合、次のステップを示した順序どおりに実行することが重要です(スタンバイ・データベースのCOMPATIBLE設定が、プライマリと同じか、あるいはそれ以上であることを確認します)。

1. まず、構成内のすべてのスタンバイ・データベースのCOMPATIBLE初期化パラメータの値を、次のように増やします。

a. 適用がスタンバイ・データベースで最新であることを確認します。

b. 各スタンバイ・データベースの1つのインスタンスで、次のSQL文を実行します。

```
ALTER SYSTEM SET COMPATIBLE=<value> SCOPE=SPFILE;
```

c. REDO ApplyまたはSQL Applyが実行中の場合は、停止します。

d. スタンバイ・データベースのすべてのインスタンスを再起動します。

e. 前にREDO ApplyまたはSQL Applyを停止した場合は、再起動します。

2. プライマリ・データベースでCOMPATIBLE初期化パラメータの値を、+次のように増やします。

a. プライマリ・データベースの1つのインスタンスで、次のSQL文を実行します。

```
ALTER SYSTEM SET COMPATIBLE=<value> SCOPE=SPFILE;
```

b. プライマリ・データベースのすべてのインスタンスを再起動します。

関連項目:

- 互換性の設定の詳細は、『[Oracle Databaseアップグレード・ガイド](#)』を参照してください。

B.6 ロジカル・スタンバイが存在しない場合のOracle Databaseのダウングレード

ロジカル・スタンバイ・データベースが含まれていないOracle Data Guard構成でOracle Databaseをダウングレードするには、次のステップを実行します。

1. すべてのフィジカル・スタンバイ・データベースがマウント済で、オープンされていないことを確認します。スタンバイ・データベースは、プライマリ・データベースのダウングレードによって生成されたすべてのREDOが適用されるまでオープンしないでください。
2. フィジカル・スタンバイ・データベースでREDO Applyをリアルタイム適用モードで開始します。
3. 『[Oracle Databaseアップグレード・ガイド](#)』で説明されている手順を使用し、次のことを念頭に置いて、プライマリ・データベースをダウングレードします。
 - ダウングレード手順においてスクリプトを実行する各ステップでは、プライマリ・データベースでのみスクリプトを実行します。プライマリ・データベースでのスクリプトの実行により生成されたすべてのREDOが、各フィジカル・スタンバイ・データベースに適用されるまで、次のダウングレード・ステップを実行しないでください。
 - ダウングレード手順においてスクリプトの実行以外のアクションが実行される各ステップでは、まずその手順をプライマリ・データベースで実行してから、各フィジカル・スタンバイ・データベースで実行します。そのアクションが各フィジカル・スタンバイ・データベースで実行されるまで、プライマリ・データベースで次のダウングレード・ステップを実行しないでください。
4. ダウングレード中にフェイルオーバーの実行が必要になった場合、フェイルオーバーを実行してから、新しいプライマリ・データベースでダウングレード手順を続行します。

B.7 ロジカル・スタンバイが存在する場合のOracle Databaseのダウングレード

ロジカル・スタンバイが存在するか、ロジカルとフィジカル・スタンバイ・データベースが混在するOracle Data Guard構成でOracle Databaseをダウングレードするには、このステップを実行します。

1. プライマリ・データベース(ここではデータベースP)で、ダウングレードする前に、次のコマンドを発行します。

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

データベースPは、プライマリ・データベース・ロールではなくなります。

2. 構成内のすべてのスタンバイ・データベースで、使用可能なすべてのREDOの適用が終了するまで待ちます。各スタンバイ・データベースで使用可能なすべてのREDOの適用が終了したかどうかを確認するために、各スタンバイ・データベースで次の問合せを実行します。

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

```
SWITCHOVER_STATUS
-----
TO PRIMARY
```

問合せにより構成内のすべてのスタンバイ・データベースについてTO PRIMARYの値が戻されるまで、ステップ3に進まないでください。

3. 『[Oracle Databaseアップグレード・ガイド](#)』で説明されている手順を使用し、次のことを念頭に置いて、ロジカル・スタンバイ・データベースをダウングレードします。

- ダウングレード手順においてスクリプトを実行する各ステップでは、ロジカル・スタンバイ・データベースでのみスクリプトを実行します。直前までプライマリ・ロール(データベースP)であったロジカル・スタンバイ・データベースでスクリプトを実行することにより生成されたすべてのREDOが、各フィジカル・スタンバイ・データベースに適用されるまで、次のダウングレード・ステップを実行しないでください。
 - ダウングレード手順においてスクリプトの実行以外のアクションが実行される各手ステップでは、まずその手順を直前までプライマリ・ロール(データベースP)であったロジカル・スタンバイ・データベースで実行してから、各フィジカル・スタンバイ・データベースで実行します。そのアクションが各フィジカル・スタンバイ・データベースで実行されるまで、直前までプライマリ・ロール(データベースP)であったロジカル・スタンバイ・データベースで次のダウングレード・ステップを実行しないでください。
4. 直前までプライマリ・ロール(データベースP)であったロジカル・スタンバイが正常にダウングレードされたら、それをオープンし、次のコマンドを発行します。

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

これでデータベースPは、プライマリ・ロールに戻りました。

5. 構成内の各ロジカル・スタンバイ・データベースで、次のコマンドを発行します(このコマンドでは、プライマリに戻るデータベース・リンクがすべてのロジカル・スタンバイ・データベースに存在する必要があります)。

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE NEW PRIMARY  
prim_db_link;
```

C ロジカル・スタンバイ・データベースでサポートされるデータ型およびDDL

ロジカル・スタンバイ・データベースの設定では、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維持していることを確認する必要があります。

次の各項では、ロジカル・スタンバイ・データベースのサポート対象および対象外の各種データベース・オブジェクト、ストレージ・タイプ、PL/SQL供給パッケージをリストにまとめています。

- [データ型に関する考慮事項](#)
- [ネイティブREDOサポートが欠如したデータ型のサポート](#)
- [透過的データ暗号化\(TDE\)のサポート](#)
- [表領域の暗号化のサポート](#)
- [行レベル・セキュリティおよびファイングレイン監査のサポート](#)
- [Oracle Label Security](#)
- [Oracle E-Business Suite](#)
- [サポートされる表記憶域タイプ](#)
- [サポートされない表記憶域タイプ](#)
- [PL/SQLパッケージに関する考慮事項](#)
- [サポートされない表](#)
- [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](#)
- [ロジカル・スタンバイ・データベースでサポートされるDDL文](#)
- [分散トランザクションおよびXAのサポート](#)
- [SecureFile LOBのサポート](#)
- [データベース・ファイル・システム\(DBFS\)のサポート](#)
- [文字セットに関する考慮事項](#)
- [DBMS_ROLLINGアップグレードの場合でのみ使用可能な追加のPL/SQLパッケージのサポート](#)

C.1 データ型に関する考慮事項

サポート対象および対象外のデータベース・オブジェクトの詳細は、次の各項を参照してください。

- [ロジカル・スタンバイ・データベースでサポートされるデータ型](#)
- [ロジカル・スタンバイ・データベースでサポートされないデータ型](#)

C.1.1 ロジカル・スタンバイ・データベースでサポートされるデータ型

ロジカル・スタンバイ・データベースでサポートされるデータ型を次に示します。

ロジカル・スタンバイ・データベースでは、次のデータ型がサポートされます。

- 抽象データ型(ADT)およびADT表
 - ADTには、最上位の列型としてサポートされないデータ型(ネストされた表、PKREF、BFILE、サポートされない不透明型など)を含めることはできません。
 - サポートされるADT列のある表では、スカラー最上位列でのみ構成されているプライマリ・キーが存在する必要があります(スカラーADT属性はこうした候補キーの一部にはなりません)。
- BINARY_DOUBLE
- BINARY_FLOAT
- BasicFileおよびSecureFileとして格納されるBLOB、CLOBおよびNCLOB。SecureFilesは、圧縮、暗号化または重複除外できます。SecureFileサポートには、11.2以上の互換性で動作するプライマリ・データベースが必要です。
[\[SecureFile LOBのサポート\]](#)を参照してください
- CHAR
- DATE
- INTERVALYEARTOMONTH
- INTERVALDAYTOSECOND
- LONG
- LONG RAW
- NCHAR
- NUMBER
- NVARCHAR2
- VARRAYとして格納されているオブジェクト(コレクションを除く)
- Oracle Text
- RAW
- マルチメディア([\[「ロジカル・スタンバイ・データベースでサポートされないデータ型」\]](#)にリストされている例外を参照)。
 - ORDAudio
 - ORDDataSource (内部)
 - ORDDicom
 - ORDDoc
 - ORDImage
 - ORDSource (内部)
 - ORDVideo
- 空間型([\[「ロジカル・スタンバイ・データベースでサポートされないデータ型」\]](#)にリストされている例外を参照)。
- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- TIMESTAMP WITH LOCAL TIMEZONE
- VARCHARおよびVARCHAR2
- 次のプライマリ・データベース互換性要件を仮定した場合は、すべてのストレージ・モデルのXMLTypeデータ。
 - CLOB形式で格納されているXMLTypeには、11.0以降の互換性設定で実行するプライマリ・データベースが必要です(CLOBとして格納されているXMLTypeはOracle Database 12cリリース1 (12.1)では非推奨です)。

- オブジェクト・リレーショナル形式またはバイナリXMLで格納されるXMLTypeには、11.2.0.3以上のREDO互換性が設定されたOracle Database 11gリリース2(11.2.0.3)以上でプライマリ・データベースを実行している必要があります。

ノート:



SQL Apply は、ADT、LOB または XMLType 列で DML を実行する関数コールを含む文をサポートしません。

ノート:



Oracle Database 12c リリース 1 (12.1)現在、COMPATIBLE 初期化パラメータを 12.0 以降に設定し、MAX_STRING_SIZE 初期化パラメータを EXTENDED に設定した場合、VARCHAR2、NVARCHAR2 および RAW データ型の最大サイズは 32 KB に増加しました。ロジカル・スタンバイ・データベースでは、ほとんどの場合、この増加したサイズがサポートされます。既知の制限については、「[プライマリ・データベース内の表の行が一意に識別できることの確認](#)」を参照してください。

C.1.1.1 互換性要件

これらの機能に対するSQL Applyサポートでは、プライマリ・データベースに対する互換性要件があります。

- マルチバイトのCLOBサポートには、10.1以上の互換性で動作するプライマリ・データベースが必要です。
- LOBおよびオーバーフローなしのIOTサポートには、10.1以上の互換性で動作するプライマリ・データベースが必要です。
- LOBおよびオーバーフローありのIOTサポートには、10.2以上の互換性で動作するプライマリ・データベースが必要です。
- TDEサポートには、11.1以上の互換性で動作するプライマリ・データベースが必要です。
- 基本圧縮および高度な行圧縮には、11.1以上の互換性で動作するプライマリ・データベースが必要です。
- ハイブリッド列圧縮サポートは、基礎となるストレージ・システムに依存します。

関連項目:

- ハイブリッド列圧縮の詳細は、[『Oracle Database概要』](#)を参照してください。

C.1.1.2 不透明型の制限

これらは、不透明型に関する制限事項です。

- インスタンスにユーザー定義の不透明データ型またはBFILEが格納されないかぎり、SYS. ANYDATAはサポートされます。
- SYS. ANYDATASET、SYS. ANYTYPEおよびユーザー定義の不透明型はサポートされていません。

C.1.2 ロジカル・スタンバイ・データベースでサポートされないデータ型

一部のデータ型はロジカル・スタンバイ・データベースでサポートされません。

表にサポートされていない次のデータ型が含まれている場合、SQL Applyを実行しても表全体が無視されます。(ネイティブの REDOベースのサポートが欠如したデータ型のサポートの詳細は、[「ネイティブREDOサポートが欠如したデータ型のサポート」](#)を参照してください。)

- ROWID、UROWID

論理UROWID列のみがサポートされます。非論理UROWIDは実行時に検出され、DMLがスキップされて、トレース・メッセージがレポート・ファイルに書き込まれます。

- ネストされた表
- ネストされた表のあるオブジェクト
- ID列

C.2 ネイティブREDOサポートが欠如したデータ型のサポート

拡張データ型サポート(EDS)機能では、ネイティブなREDOベースのサポートが欠如した特定のデータ型をサポートするためのロジカル・スタンバイ用のメカニズムが提供されます。

ノート:



Oracle Database 18c 以降では、拡張データ型サポート(EDS)は非推奨になりました。EDS でサポートされているすべての Oracle データ型がロジカル・スタンバイまたは Oracle GoldenGate でネイティブにサポートされるようになりました。

たとえば、SDO_GEOMETRY列を含む表はEDSを使用してレプリケートできます。(ソース表には主キーがある必要があります。)

DBA_LOGSTDBY_EDS_SUPPORTEDビューを問い合わせ、EDSの候補の表を探すことができます。

C.3 透過的データ暗号化(TDE)のサポート

Oracle Data Guard SQL Applyを使用すると、透過的データ暗号化(TDE)を有効に設定したプライマリ・データベースにデータ保護を提供できます。

ロジカル・スタンバイ・データベースを使用し、高度なセキュリティ要件を伴うアプリケーションにデータ保護を提供する場合は、次のことを考慮してください。

- サーバーが保持するキーを使用する透過的データ暗号化が指定された表は、プライマリ・データベースとスタンバイ・データベースの両方が11.1以上の互換性レベルで稼働している場合のロジカル・スタンバイ・データベースで複製されます。
- ハードウェア・セキュリティ・モジュールの下での透過的データ暗号化は、Oracle Database 11gリリース2 (11.2)以降のロジカル・スタンバイ・データベースではサポートされません。

ロジカル・スタンバイ・データベースの下では、暗号化された列が含まれる表をレプリケートするときに、次の制限事項を考慮する必要があります。

1. 暗号化されたREDOレコードを変換するには、透過的データ暗号化キーを含むオープン・ウォレットへのアクセス権がSQL Applyに必要です。そのため、キーを含むウォレットの作成後に、そのウォレットをプライマリ・データベースからスタンバイ・データベースにコピーする必要があります。
2. ウォレットは、マスター・キーが変更されるたびに、プライマリ・データベースからロジカル・スタンバイ・データベースにコピーする必要があります。

3. ロジカル・スタンバイ・データベースでは、プライマリ・データベースから暗号化された表をレプリケートしている間、マスター・キーをREKEY(キーの変更)しないことをお勧めします。これは、暗号化されたREDOレコードの検出時にSQL Applyが停止する原因になります。
4. ロジカル・スタンバイ・データベースでは、レプリケートされた表の暗号化キーをREKEYできます。このためには、REKEYコマンドを発行する前にガード設定をNONEにして低くする必要があります。
5. レプリケートされた暗号化された表では、プライマリ・データベースで使用された暗号化方式とは異なる方式を列に対して使用できます。たとえば、HR. EMPLOYEES表のSALARY列がプライマリ・データベースでAES102暗号化アルゴリズムを使用して暗号化される場合、ロジカル・スタンバイではAES256暗号化アルゴリズムを使用して暗号化することができます。あるいは、ロジカル・スタンバイ・データベースでは、SALARY列を暗号化しないままにしておくことも可能です。

C.4 表領域の暗号化のサポート

Oracle Data Guard SQL Applyを使用すると、表領域の暗号化を有効に設定したプライマリ・データベースにデータ保護を提供できます。

その場合、[透過的データ暗号化\(TDE\)のサポート](#)に示した制限事項1、2および3が適用されます。

プライマリ上の表領域の暗号化、キー更新または復号化でロジカル・スタンバイでの同じ操作の必要性が誘発されることはありません。ただし、ロジカル・スタンバイではキー更新の能力も必要です。

ノート:

暗号化された表領域の表に対する変更は、SQL Apply が REDO レコードをマイニングおよび適用した際、暗号化されていない形式のユーザー・データのレコードが長時間維持される場合があります。これを許容できない場合、次のコマンドを発行して、暗号化された表領域への SQL Apply のコンポーネントのマイニングに関係するすべてのメタデータ表を移動します。

```
SQL> DBMS_LOGMNR_D.SET_TABLESPACE (NEW_TABLESPACE => ' ENCRYPTED_LOGMNR_TS' );
```

C.5 行レベル・セキュリティおよびファイंगレイン監査のサポート

Oracle Database 11gでは、ロジカル・スタンバイによってDBMS_RLSおよびDBMS_FGA PL/SQLパッケージで提供されているセキュリティ環境を自動的にレプリケートすることができます。

このサポートによりセキュリティ環境が透過的に維持されるため、サーバーをスタンバイにフェイルオーバーする際に、セキュリティに配慮した管理の手間が軽減されます。また、プライマリ・データに適用されたアクセス・コントロール・ポリシーを自動的にスタンバイに必ず転送し、スタンバイ・データに同等な保護レベルを付与することができます。スタンバイ・サーバーを11gで新しく作成すると、このレプリケーションはデフォルト設定で有効化されますが、それ以外の場合はDBAが適宜有効化する必要があります。

これらのPL/SQLパッケージのレプリケーションをサポートするには、プライマリとスタンバイの両方が11.1以上の互換性設定で稼働している必要があります。

また、参照される表は、ロジカル・スタンバイによって保持されるオブジェクトである必要があります。たとえば、ROWID列を含む表のデータがロジカル・スタンバイによって保持されない場合、その表を参照するDBMS_RLSおよびDBMS_FGA呼出しは保持されません。

C.5.1 行レベルのセキュリティ

行レベルのセキュリティは、仮想プライベート・データベース(VPD)とも呼ばれる機能で、表、ビューまたはシノニムにアクセスするときに、セキュリティの粒度をFINEレベルにまで強化します。

VPDポリシーで保護された表、ビューまたはシノニムに直接または間接的にユーザーがアクセスすると、サーバーはそのユーザーのSQL文を動的に変更します。この変更では、セキュリティ・ポリシーを実装するファンクションによって戻されるWHERE条件(述語)が作成されます。文は、ファンクションで表される条件あるいはファンクションによって戻される条件を使用して、ユーザーに対して透過的に、動的に変更されます。VPDポリシーは、SELECT、INSERT、UPDATE、INDEXおよびDELETE文に適用されます。VPDは、DBMS_RLSパッケージを使用してセキュリティ・ポリシーを適用すると実装されます。

DBMS_RLSプロシージャをプライマリで実行すると、追加情報がREDOに取得されるため、スタンバイでプロシージャ・コールを論理的に再構築して実行することができます。ロジカル・スタンバイでは、VPDの補助オブジェクト(コンテキスト、データベース・ログオン、トリガー、サポート対象パッケージなど)をレプリケートできます。これらのオブジェクトが保持されるスキーマに存在し、レプリケーションを停止させるDDLスキップが構成されていないことを確認する必要があります。

C.5.2 ファイングレイン監査

きめ細かい監査により、select文を監査することができます。

DBMS_FGAパッケージでは、表にアクセスするすべてのselect文を、アクセスしたデータとともに取得することができます。FGAポリシーは特定の列、または指定された述語戻り値がTRUEの行を戻す、select文にも適用することができます。

DBMS_FGAプロシージャをプライマリで実行すると、追加情報がREDOに取得されるため、スタンバイでプロシージャ・コールを論理的に再構築して実行することができます。

C.5.3 PL/SQLレプリケーションのスキップおよび有効化

パッケージとプロシージャでのワイルドカードの使用の指定がサポートされていない点を除き、PL/SQLにはDDL文とまったく同様なskipとskip_errorルールを設定することができます。

たとえばVPDのすべての点をスキップするには、次を実行します。

```
DBMS_LOGSTDBY.Skip (
stmt => 'PL/SQL',
schema_name => 'SYS',
object_name => 'DBMS_RLS',
use_like => FALSE);
```

指定されたスキーマが、パッケージが定義されているスキーマです。パッケージの個々のプロシージャをスキップする場合の構文は、次のようになります。

```
DBMS_LOGSTDBY.Skip (
stmt => 'PL/SQL',
schema_name => 'SYS',
object_name => 'DBMS_RLS.ADD_POLICY',
use_like => FALSE);
```

特定のスキーマや表でVPDをスキップするには、スキップ・プロシージャを使用する必要があります。スキップ・プロシージャには、たとえば次のような実行予定の完全修飾PL/SQL文が渡されます。

```
DBMS_RLS.DROP_POLICY(
object_schema => 'SCOTT',
object_name => 'EMP',
policy_name => 'MYPOLICY');
```

すると、プロシージャは、文を解析してスキップするか、適用するか、適用を中止してDBAに補正アクションを実行させるかを決定します。

DDLとは異なり、PL/SQLに対するスキップ・プロシージャは代用文を戻しません。

C.6 Oracle Label Security

Oracle Database 12cリリース2 (12.2)以降では、Oracle Label Security (OLS)を使用するデータベースを、Oracle Data Guardデータベースのローリング・アップグレードを一時ロジカル・スタンバイ・データベースおよびPL/SQLパッケージDBMS_ROLLINGとともに使用して、新しいOracleデータベースのリリースおよびパッチ・セットにアップグレードできます。

C.7 Oracle Database Vault

Oracle Data Guardのローリング・アップグレードは、Oracle Database Vaultを使用するデータベースをサポートしています。

Oracle Database 12cリリース2 (12.2.0.1)以降では、Oracle Database Vaultを使用するデータベースを、Oracle Data Guardデータベースのローリング・アップグレードを一時ロジカル・スタンバイおよびPL/SQLパッケージDBMS_ROLLINGとともに使用して、新しいOracleデータベースのリリースおよびパッチ・セットにアップグレードできます。

C.8 Oracle E-Business Suite

ロジカル・スタンバイ・データベースは、サポートされていないデータ型を含む表があるため、Oracle E-Business Suite実装を完全にはサポートしていません。

しかし、SKIPルールを使用して、E-Business Suiteスキーマと表のサブセットを複製し、アプリケーションをロジカル・スタンバイにオフロードすることは可能です。

関連項目:

Oracle E-Business Suiteでのロジカル・スタンバイ使用の詳細は、<http://support.oracle.com>のMy Oracle Supportノート851603.1を参照してください。

C.9 サポートされる表記憶域タイプ

ロジカル・スタンバイ・データベースでは、いくつかの表記憶域タイプがサポートされます。

- クラス表(索引クラスおよびヒープ・クラスを含む)
- 索引構成表(オーバーフロー・セグメントを含むパーティション化および非パーティション化)。
- ヒープ構成表(パーティション化および非パーティション化)
- 高度な行圧縮および基本表圧縮。どちらのオプションでも、プライマリ・データベースの互換性設定が11.1.0以上に設定されている必要があります。
- SecureFilesとして格納されるLOB列を含む表(互換性が11.2以降に設定されている場合)。
- ハイブリッド列圧縮を使用している表(互換性が11.2.0.2以降に設定されている場合)。

関連項目:

- ハイブリッド列圧縮の詳細は、[『Oracle Database概要』](#)を参照してください。
- 仮想列を含む表(表にロジカル・スタンバイでサポートされていない他の列またはプロパティがない場合)
- 主キーもNULL値が許容されていない一意制約/索引もない場合、宣言最大長が4000バイトのすべての列が、変更された行を識別するためにUPDATE文の一部としてログに記録されます。行の識別の目的では、ロジカル・スタンバイには次のデータ型のいずれか1つ以上の表示可能な(仮想ではない)列を含む表が必要です。
 - CHAR
 - VARCHAR
 - VARCHAR2 (宣言済の列の長さ<= 4000バイト)
 - NVARCHAR
 - NVARCHAR2 (宣言済の列の長さ<= 4000バイト)
 - NUMBER
 - DATE
 - RAW
 - BINARY FLOAT
 - BINARY DOUBLE
 - TIMESTAMP
 - TIMESTAMPWITHTIMEZONE
 - TIMESTAMPWITHLOCALTIMEZONE
 - INTERVALYEARTOMONTH
 - INTERVALDAYTOSECOND

関連項目:

- [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](#)

C.10 サポートされない表記憶域タイプ

表にこれらのデータ型しか含まれない場合、ロジカル・スタンバイではサポートされません。

- LOB (CLOB、NCLOB、BLOB)
- LONG
- LONG RAW
- OBJECT TYPE
- COLLECTIONS
- XML
- VARCHAR2 (宣言済の列の長さ> 4000バイト)
- NVARCHAR2 (宣言済の列の長さ> 4000バイト)
- RAW (宣言済の列の長さ> 4000バイト)

関連項目:

- [SecureFile LOBのサポート](#)
- [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](#)

C.10.1 パーティション化の結果としてのサポートされない表

ロジカル・スタンバイでは、システム・パーティション化または参照パーティション化を使用する表はサポートされません。

可能な場合、DBA_LOGSTDBY_UNUNSUPPORTEDビューのATTRIBUTES列には、SQL Applyによって表がサポートされない理由が表示されます。表構造そのものがSQL Applyでサポートされていない場合(たとえば、表がシステム・パーティション化されている)、ATTRIBUTES列はNULLになることがあります。

C.11 PL/SQLパッケージに関する考慮事項

サポート対象および対象外のPL/SQLパッケージについては、次の考慮事項に注意してください。

- [サポートされるPL/SQLパッケージ](#)
- [サポートされないPL/SQLパッケージ](#)
- [ロジカル・スタンバイでのXMLおよびXDB PL/SQLパッケージの処理](#)

関連項目:

Oracle PL/SQL供給パッケージの詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)を参照してください。

C.11.1 サポートされるPL/SQLパッケージ

システムのメタデータやユーザー・データを変更しない、オラクル社が提供するPL/SQLパッケージは、アーカイブREDOログ・ファイルにフットプリントを残さないため、プライマリ・データベースで安全に使用できます。

この種のパッケージの例には、DBMS_OUTPUT、DBMS_RANDOM、DBMS_PIPE、DBMS_DESCRIBE、DBMS_TRACE、DBMS_METADATA、DBMS_CRYPTOがあります。

システムのメタデータは変更せず、ユーザー・データを変更することがある、オラクル社が提供するPL/SQLパッケージは、変更されるデータが[「ロジカル・スタンバイ・データベースでサポートされるデータ型」](#)に示すサポートされるデータ型に属するかぎり、SQL Applyでサポートされます。この種のパッケージの例には、DBMS_LOB、DBMS_SQL、DBMS_TRANSACTIONがあります。

Oracle Data Guardのロジカル・スタンバイでは、DBMS_DDL、DBMS_FGA、SDO_META、DBMS_REDACT、DBMS_REDEFINITION、DBMS_RLS、DBMS_SQL_TRANSLATOR、DBMS_XDS、DBMS_XMLINDEXおよびDBMS_XMLSCHEMAのパッケージを使用して実行されるアクションのレプリケーションがサポートされます。

ロジカル・スタンバイでサポートされているパッケージを特定するために、DBA_LOGSTDBY_PLSQL_SUPPORTビューを問い合わせることができます。たとえば、次の問合せを実行すると、汎用ロジカル・スタンバイでサポートされているパッケージを探することができます。

```
SQL> SELECT OWNER, PKG_NAME FROM DBA_LOGSTDBY_PLSQL_SUPPORT -  
> where support_level = 'ALWAYS';
```

DBMS_ROLLINGパッケージを使用して実行したローリング・アップグレードの場合にサポートされるパッケージを特定するために、次のようにしてDBA_LOGSTDBY_PLSQL_SUPPORTビューを問い合わせることができます。

```
SQL> SELECT OWNER, PKG_NAME FROM DBA_LOGSTDBY_PLSQL_SUPPORT -  
> where support_level = 'DBMS_ROLLING';
```

C.11.2 サポートされないPL/SQLパッケージ

システムのメタデータを変更する、オラクル社が提供するPL/SQLパッケージは、通常、SQL Applyではサポートされないため、その影響はロジカル・スタンバイ・データベースでは参照できません。

この種のパッケージの例には、DBMS_JAVA、DBMS_REGISTRY、DBMS_ALERT、DBMS_SPACE_ADMIN、DBMS_REFRESHおよびDBMS_AQがあります。

また、DBMS_RESOURCE_MANAGERパッケージはフィジカル・スタンバイのローリング・アップグレードにはサポートされません。

C.11.2.1 DBMS_JOBのサポート

DBMS_JOBに固有のサポートが用意されています。プライマリ・データベースで作成されたジョブはスタンバイ・データベースで複製されますが、このスタンバイがスタンバイ・ロールを維持しているかぎり、実行されません。

スイッチオーバーまたはフェイルオーバーが発生すると、元のプライマリ・データベースでスケジューラされていたジョブの実行は、新規のプライマリ・データベースで自動的に開始されます。

ロジカル・スタンバイでジョブを作成することもできます。これらのジョブは、ロジカル・スタンバイがスタンバイ・ロールを維持している間のみ実行されます。

C.11.2.2 DBMS_SCHEDULERのサポート

スタンバイ・データベースでジョブを実行できるように、DBMS_SCHEDULERに固有のサポートが用意されています。

database_roleと呼ばれるスケジューラ・ジョブの新しい属性がOracle Database 11gで作成されました。この内容はV\$DATABASEのdatabase_role属性と一致しています。スケジューラ・ジョブを作成すると、ローカル・ロールにデフォルト設定されます。したがって、スタンバイで作成されたジョブは、LOGICAL STANDBYのdatabase_roleにデフォルト設定されます。ジョブ・スケジューラは、現行のロールに固有のジョブだけを実行します。スイッチオーバーまたはフェイルオーバーの際は、スケジューラが新しいロールに固有のジョブ実行に切り替わります。

スケジューラ・ジョブはスタンバイにレプリケートされません。ただし、DBMS_ROLLING PL/SQLパッケージを使用して実行したローリング・アップグレードの場合を除きます。しかし、既存のジョブはDBMS_SCHEDULER.Set_Attributeプロシージャを使用して、新しいロールの下でアクティブ化できます。あるいは、両方のロールで実行する必要があるジョブをクローニングし、コピーを他のロールに固有のものにすることが可能です。DBA_SCHEDULER_JOB_ROLESビューには、どのジョブがどのロールに固有かが表示されます。

スケジューラのジョブをロジカル・スタンバイ・データベースで実行する際は、データベース・ガードに従います。したがって、維持されていない表の変更が必要なジョブを実行するには、データベース・ガードをSTANDBYに設定します。(ALTER SESSION DISABLE GUARD文をPL/SQLブロック内で使用しても、効果がありません。)

C.11.3 ロジカル・スタンバイでのXMLおよびXDB PL/SQLパッケージの処理

一部の互換性要件を伴い、ロジカル・スタンバイはすべてのストレージ・モデルのXMLTypeデータをサポートします。

要件は次のとおりです。

- CLOB形式で格納されているXMLTypeには、11.0以降の互換性設定で実行するプライマリ・データベースが必要です (CLOBとして格納されているXMLTypeはOracle Database 12cリリース1 (12.1)では非推奨です)。

- オブジェクト・リレーショナル形式またはバイナリXMLで格納されるXMLTypeには、11.2.0.3以上のREDO互換性が設定されたOracle Database 11gリリース2(11.2.0.3)以上でプライマリ・データベースを実行している必要があります。

完全にはサポートされないXMLとともに使用されるPL/SQLパッケージがいくつかあります。

ロジカル・スタンバイがサポートしているPL/SQLパッケージとプロシージャは、インメモリ構造だけで変更を行い、データベースに格納されたデータは変更しません。これらのパッケージはREDOを生成しないため、ロジカル・スタンバイにはレプリケートされません。

ロジカル・スタンバイでサポートされなくても、レプリケーション・アクティビティの続行にロジカル・スタンバイ・データベースで対応する起動が必要なXMLおよびXDBに関連する特定のPL/SQLパッケージおよびプロシージャは、プライマリ・データベースでこれらのプロシージャが起動することにより、プロシージャの起動を示す追加のREDOレコードが生成されるように装備されています。SQL ApplyでこのようなREDOレコードが検出されると、SQL Applyは停止し、プロシージャ名を示すエラー・メッセージがDBA_LOGSTDBY_EVENTS表に書き込まれます。これにより、DBAは対応するプロシージャを適切な時期にロジカル・スタンバイ・データベースで起動し、プライマリ・データベースで生成された後続のREDOレコードをロジカル・スタンバイ・データベースに正常に適用することができます。これらのサポートされないプロシージャの処理方法の詳細は、[「DBMS_XMLSCHEMAスキーマ」](#)から[「順序に依存するサポートされないPL/SQLの補正」](#)を参照してください。

次のパッケージには、サポートされないプロシージャが含まれます。

- DBMS_XMLSCHEMA(互換性が11.2以降に設定されている場合にサポートされます。)
- DBMS_XMLINDEX

これらのパッケージに加えて、ロジカル・スタンバイではXDBスキーマへの変更もサポートしません。XDBスキーマ内のオブジェクトは、システム・メタデータとみなされ、直接の変更はレプリケートされません。

Oracle XML DBリポジトリで管理される表は、階層対応表とも呼ばれますが、ロジカル・スタンバイではサポートされません。これらの表は、XMLデータの格納に使用され、通常のSQLアクセスに加えてFTPおよびHTTPプロトコルを使用してアクセスできます。これらの表の詳細は、[『Oracle XML DB開発者ガイド』](#)を参照してください。

C.11.3.1 DBMS_XMLSCHEMAスキーマ

DBMS_XMLSCHEMAパッケージ内の特定のプロシージャは、ロジカル・スタンバイではサポートされず、レプリケートできません。

ロジカル・スタンバイは、これらのプロシージャへのコールを検出すると、これらのコールに対してユーザーが補正アクションを実行できるように停止します。これらのサポートされないプロシージャの処理に使用できる代替方法の詳細は、[「サポートされないPL/SQLプロシージャの処理」](#)から[「順序に依存するサポートされないPL/SQLの補正」](#)を参照してください。

- COPYEVOLVE
- INPLACEEVOLVE
- COMPILESCHEMA

XDBスキーマは、Oracle管理スキーマです。このスキーマに対する変更は、ロジカル・スタンバイによって自動的にスキップされます。次のプロシージャは、XDBスキーマを変更しますが、レプリケートされません。

- GENERATEBEAN

C.11.3.2 DBMS_XMLINDEXパッケージ

DBMS_XMLINDEXパッケージ内のプロシージャはすべてサポートされています。ただし、次のものは除きます。

- DBMS_XMLINDEX. REGISTERPARAMETER
- DBMS_XMLINDEX. MODIFYPARAMETER
- DBMS_XMLINDEX. DROPPARAMETER

C.11.3.3 サポートされないPL/SQLプロシージャの処理

サポートされないPL/SQLプロシージャの処理には、2つのオプションがあります。

1つ目のオプションは、ロジカル・スタンバイの適用プロセスが停止してなんらかの補正アクションを手動で実行できるようにする方法です。2つ目のオプションは、事前対応アクションを実行し、さらにロジカル・スタンバイのスキップ・プロシージャを使用してサポートされないPL/SQLをスキップする方法です。次の各項で、これらのオプションについてそれぞれ説明します。

C.11.3.4 サポートされないPL/SQLの手動による補正

ロジカル・スタンバイでサポートされないものが検出されると、適用プロセスは停止し、エラーがDBA_LOGSTDBY_EVENTS表に記録されます。

この表を問い合わせると、スタンバイが停止した原因となるアクションと、該当する場合は補正のために必要なアクションを確認できます。

次の例に、問合せ内容のサンプルとその出力を示します。

```
select status, event from dba_logstdby_events
       where commit_scn >= (select applied_scn from dba_logstdby_progress) and
       status_code = 16265
       order by commit_scn desc;
```

STATUS

EVENT

ORA-16265: Unsupported PL/SQL procedure encountered

begin

"XDB"."DBMS_XMLSCHEMA"."REGISTERPARAMETER" (
 "NAME" => 'myIndexParam',
 "PARAMETER" => 'PATH TABLE

ORA-16265: Unsupported PL/SQL procedure encountered

begin

"XDB"."DBMS_XMLSCHEMA"."REGISTERPARAMETER" (
 "NAME" => 'myIndexParam',
 "PARAMETER" => 'PATH TABLE

2 rows selected.

ロジカル・スタンバイは失敗したトランザクションを自動的に再試行するので、同じ情報が含まれた2行が戻されます。結果は、xmlplsqsch2スキーマでDBMS_XMLSCHEMA.REGISTERSchemaの呼出しに遭遇したため、スタンバイが停止したことを示しています。この情報を使用してプライマリから必要なファイルを転送し、スタンバイにスキーマを登録することができます。

スキーマをスタンバイに正しく登録したら、ロジカル・スタンバイへの適用プロセスを再開できます。これはたとえば次のように、SKIP FAILED TRANSACTIONオプションを使用して実行する必要があります。

```
alter database start logical standby apply skip failed transaction'
```

ロジカル・スタンバイは、問題となるトランザクションをスキップし、プライマリからのREDOの適用を続行します。

サポートされないPL/SQLを手動でレプリケートするための一般的な手順は、これらのステップに従います：

1. サポートされないPL/SQLをプライマリ・データベースで実行します。
2. スタンバイ・データベースでサポートされないPL/SQLが検出され、適用が停止します。
3. DBA_LOGSTDBY_EVENTS表を調べ、適用が停止した原因を確認します。

4. スタンバイで、サポートされないPL/SQLについてなんらかの補正アクションを実行します。

5. スタンバイで適用を再開します。

C.11.3.5 順序に依存するサポートされないPL/SQLの補正

前述のアプローチは有効ですが、すべての場合に使用できるわけではありません。他のトランザクションとの相対的なPL/SQL実行時間が重要ではない場合にかぎり、安全に使用することができます。使用すべきではない場合の例として、DBMS_XMLSCHEMA.copyEvolveがあげられます。DBMS_XMLSCHEMA.copyEvolveプロシージャはスキーマを進化、または変化させ、列の追加または削除によって表を変更することがあり、XMLドキュメントの有効性/無効性も変更します。

ロジカル・スタンバイでのこのプロシージャの実行タイミングは非常に重要です。このプロシージャがプライマリ・データベースで実行されたことがわかった場合、ロジカル・スタンバイ上で適用を停止したタイミングが唯一安全です。

スキーマを発展させる前に、プライマリでそのスキーマを使用するトラフィックをすべてQUIESCE(静止)させることも重要です。そうしないと、2つのトランザクション間の依存性がロジカル・スタンバイには識別できないため、プライマリではevolveSchemaに間に合うようにクローズされるトランザクションが、ロジカル・スタンバイでは異なる順序で実行される可能性があります。そのため、順序に依存するPL/SQLが含まれる場合は、次のステップに従う必要があります。

1. プライマリで依存表への変更をQUIESCEさせます。
2. プライマリでCopyEvolveを実行します。
3. スタンバイでのCopyEvolve PL/SQLの停止を待機します。
4. スタンバイで補正CopyEvolveを適用します。
5. スタンバイで適用を再開します。

[例C-1](#)に、RegisterSchema呼出しの処理方法を決定するために使用されるプロシージャのサンプルを示します。

例C-1 RegisterSchema用のPL/SQLスキップ・プロシージャ

```
-- Procedures to determine how to handle registerSchema calls

-- This procedure extracts the schema URL, or name, from the statement
-- string that is passed into the skip procedure.

Create or replace procedure sec_mgr.parse_schema_str(
    statement          in varchar2,
    schema_name        out varchar2)
Is
    pos1 number;
    pos2 number;
    workingstr  varchar2(32767);
Begin

    -- Find the correct argument
    pos1 := instr(statement, '"SCHEMAURL" => ');
    workingstr := substr(statement, pos1 + 16);

    -- Find the end of the schema name
    pos1 := instr(workingstr, ' ');

    -- Get just the schema name
    workingstr := substr(workingstr, 1, pos1 - 1);

    schema_name := workingstr;

End parse_schema_str;
```

```

/
show errors

-- This procedure checks if a schema is already registered. If so,
-- it returns the value DBMS_LOGSTDBY.SKIP_ACTION_SKIP to indicate that
-- the PL/SQL should be skipped. Otherwise, the value
-- DBMS_LOGSTDBY.SKIP_ACTION_SKIP is returned and Logical Standby apply
-- will halt to allow the DBA to deal with the registerSchema call.

Create or replace procedure sec_mgr.skip_registerschema(
    statement          in varchar2,
    package_owner      in varchar2,
    package_name       in varchar2,
    procedure_name     in varchar2,
    current_user       in varchar2,
    xidusn             in number,
    xidslt             in number,
    xidsqn             in number,
    exit_status        in number,
    skip_action        out number)
Is
    schema_exists number;
    schemastr varchar2(2000);
Begin

    skip_action := DBMS_LOGSTDBY.SKIP_ACTION_SKIP;

    -- get the schame name from statement
    parse_schema_str(statement, schemastr);

    -- see if the schema is already registered
    select count(*) into schema_exists from sys.all_xml_schemas s
        where s.schema_url = schemastr and
              s.owner = current_user;

    IF schema_exists = 0 THEN
        -- if the schema is not registered, then we must stop apply
        skip_action := DBMS_LOGSTDBY.SKIP_ACTION_APPLY;
    ELSE
        -- if the schema is already registered, then we can skip this statement
        skip_action := DBMS_LOGSTDBY.SKIP_ACTION_SKIP;
    END IF;

End skip_registerschema;
/
show errors

-- Register the skip procedure to deal with the unsupported registerSchema
-- PL/SQL.
Begin
    sys.dbms_logstdby.skip(stmt => 'PL/SQL',
        schema_name => 'XDB',
        object_name => 'DBMS_XMLSCHEMA.REGISTERSHEMA',
        proc_name   => 'SEC_MGR.SKIP_REGISTERSHEMA',
        use_like    => FALSE );
    End;
/
show errors

```


C.12 サポートされない表

ロジカル・スタンバイ・データベースを作成する前に、プライマリ・データベースでサポートされないデータベース・オブジェクトを特定することが重要です。

プライマリ・データベースでサポートされないデータ型や表に対する変更は、ロジカル・スタンバイ・データベース上のSQL Applyで自動的にスキップされるためです。また、エラー・メッセージは返されません。

ノート:



Oracle Database リリース 12.2 以降では、新しいタイプまたは機能(長い識別子を含む)は、DBMS_ROLLING パッケージまたは Oracle Golden Gate を使用した論理レプリケーションでのみサポートされます。

ノート:



ソートされたハッシュ・クラスタ表は、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。

ロジカル・スタンバイのサポートの観点から、データベースには次の3種類のオブジェクトがあります。

- SQL Applyによって明示的にメンテナンスされるオブジェクト
- SQL Applyによって暗黙的にメンテナンスされるオブジェクト
- SQL Applyによってメンテナンスされないオブジェクト

Oracleデータベースに同梱される一部のスキーマ(SYSTEMなど)には、SQL Applyによって暗黙的にメンテナンスされるオブジェクトが含まれます。しかし、ユーザー定義の表をSYSTEMに入れると、サポートされるデータ型の列が含まれている場合でもその表はメンテナンスされません。SQL Applyによってメンテナンスされないオブジェクトを検出するには、次の2つの問合せを実行する必要があります。1つ目の問合せは次のとおりです。

```
SQL> SELECT OWNER FROM DBA_LOGSTDBY_SKIP WHERE STATEMENT_OPT = 'INTERNAL SCHEMA' ;
```

この問合せは、内部スキーマとみなされるスキーマをすべて返します。これらのスキーマにあるユーザー表は、ロジカル・スタンバイ・データベースでレプリケートされず、DBA_LOGSTDBY_UNSUPPORTEDビューにも表示されません。スキーマに実装された機能がロジカル・スタンバイの下でサポートされる場合、オラクル社で作成されたこれらのスキーマ内の表は、ロジカル・スタンバイで保持されます。

実行が必要な2つ目の問合せは次のとおりです。この問合せは、内部スキーマに属さず、サポートされないデータ型が含まれるためにSQL Applyによってメンテナンスされない表を返します。

```
SQL> SELECT DISTINCT OWNER, TABLE_NAME FROM DBA_LOGSTDBY_UNSUPPORTED -  
> 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 -  
> WHERE OWNER='OE' AND TABLE_NAME = 'CUSTOMERS';
```

COLUMN_NAME	DATA_TYPE
CUST_ADDRESS	CUST_ADDRESS_TYP
PHONE_NUMBERS	PHONE_LIST_TYP
CUST_GEO_LOCATION	SDO_GEOMETRY

プライマリ・データベースにサポートされない表が含まれている場合、REDOデータをロジカル・スタンバイ・データベースに適用すると、SQL Applyサービスによって、その表が自動的に除外されます。

ノート:



この項で示される問合せでは、マルチテナント・データベース(CDB)環境で作業している場合、DBA ビューの多くに代替として使用する必要のある類似の CDB ビューがあります。たとえば、DBA_LOGSTDBY_SKIP ビューのかわりに CDB_LOGSTDBY_SKIP ビューを問合せます。

C.12.1 ローリング・アップグレード中にサポートされない表

ローリング・アップグレードを実行する前に、含まれるいずれかの表にロジカル・スタンバイ・データベースでサポートされないデータ型が含まれるかどうかを判別してください。

これを行うには、実行中のローリング・アップグレードのタイプに応じて、DBA_LOGSTDBY_UNSUPPORTEDビューまたは DBA_ROLLING_UNSUPPORTEDビューのいずれかを問い合わせることができます。

[DBMS_ROLLINGを使用したローリング・アップグレードの実行](#)に説明されているように、DBMS_ROLLING PL/SQLパッケージを使用したローリング・アップグレードを実行している場合、DBA_ROLLING_UNSUPPORTEDビューを問い合わせてください。

DBMS_ROLLINGパッケージを使用せず、[SQL Applyを使用したOracle Databaseのアップグレード](#)に説明されている手動プロセスに従っている場合、DBA_LOGSTDBY_UNSUPPORTEDビューを問い合わせてください。

DBMS_ROLLINGを使用して実行されるローリング・アップグレードは、手動ローリング・アップグレード操作よりも多くのオブジェクト型をサポートします。たとえば、DBMS_ROLLINGを使用して実行されるアップグレードのみキュー表をサポートします。さらに、DBMS_ROLLINGを使用して実行されるローリング・アップグレードは、より多くのPL/SQLパッケージをサポートします。

関連項目:

- 手動ローリング・アップグレードの実行の詳細は、[「SQL Applyを使用したOracle Databaseのアップグレード」](#)を参照してください
- DBMS_ROLLING PL/SQLパッケージを使用したローリング・アップグレードの実行の詳細は、[「DBMS_ROLLINGを使用したローリング・アップグレードの実行」](#)を参照してください
- DBMS_ROLLINGアップグレードの場合でのみ使用可能なPL/SQLパッケージのサポートの詳細は、[「DBMS_ROLLINGアップグレードの場合でのみ使用可能な追加のPL/SQLパッケージのサポート」](#)を参照してください
- DBMS_ROLLING PL/SQLパッケージの詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)

- ビューの詳細情報は、『[Oracle Databaseリファレンス](#)』を参照してください。

C.12.2 PL/SQLファンクションで実行されたDMLの結果としてのサポートされない表

サポートされている表での挿入または更新DML操作中に、表外の列(LOB、XMLTypeまたはADT)がPL/SQLファンクションを介して変更され、次にそのファンクションがその実行の途中で別の表でDMLを実行する場合、生成されるREDOパターンはLogMinerでサポートされません。

結果としてこのようなワークロードのREDOは、LogMinerを使用して確実にマイニングすることはできません。

C.13 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 Applyによって特定のSQL文が自動的にスキップされます。

影響を受ける文は次のとおりです。

- ALTER DATABASE
- ALTER MATERIALIZED VIEW
- ALTERMATERIALIZEDVIEWLOG
- ALTER SESSION
- ALTER SYSTEM
- CREATE CONTROL FILE
- CREATE DATABASE
- CREATE DATABASE LINK
- CREATE PFILE FROM SPFILE
- CREATE MATERIALIZED VIEW
- CREATEMATERIALIZEDVIEWLOG
- CREATE SCHEMA AUTHORIZATION
- CREATE SPFILE FROM PFILE
- DROP DATABASE LINK
- DROP MATERIALIZED VIEW
- DROPMATERIALIZEDVIEWLOG
- EXPLAIN
- LOCK TABLE
- PURGE DBA_RECYCLEBIN
- PURGE INDEX
- SET CONSTRAINTS
- SET ROLE
- SET TRANSACTION
-

プライマリ・データベースで実行されるこの他のSQL文はすべて、ロジカル・スタンバイ・データベースに適用されます。

C.14 ロジカル・スタンバイ・データベースでサポートされるDDL文

DBMS_LOGSTDBY. SKIPプロシージャには、オプションのキーワードがいくつかあります。

[表C-1](#)に、DBMS_LOGSTDBY. SKIPプロシージャのstmtパラメータについて、サポートされる値を示します。表の左側の列には、キーワードの右側にあるSQL文セットの識別に使用される可能性のあるキーワードを示します。また、sys.audit_actions表に示されたSQL文(表1-13の右列を参照)はすべて、有効な値でもあります。キーワードは通常、データベース・オブジェクトによって定義されます。

関連項目:

[DBMS_LOGSTDBY](#)パッケージの詳細は、『[Oracle Database PL/SQLパッケージおよびタイプ・リファレンス](#)』および『[DDL文のスキップ・ハンドラの設定](#)』を参照してください

表C-1 DBMS_LOGSTDBY.SKIPプロシージャのstmtパラメータの値

キーワード	関連するSQL文
このグループの SQL 文に対するキーワードはない。	GRANT REVOKE ANALYZE TABLE ANALYZE INDEX ANALYZE CLUSTER
CLUSTER	AUDIT CLUSTER CREATE CLUSTER DROP CLUSTER TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT DROP CONTEXT
DATABASE LINK	CREATE DATABASE LINK CREATE PUBLIC DATABASE LINK DROP DATABASE LINK DROP PUBLIC DATABASE LINK
DIMENSION	ALTER DIMENSION CREATE DIMENSION DROP DIMENSION
DIRECTORY	CREATE DIRECTORY DROP DIRECTORY
DML	表内の DML 文を含む(例: INSERT、UPDATE、DELETE)
INDEX	ALTER INDEX CREATE INDEX DROP INDEX
NON_SCHEMA_DDL	特定のスキーマに関連付けられていないすべての <i>DDL</i>
	ノート: SCHEMA_NAME および OBJECT_NAME は NULL である必要があります。
PROCEDURE 脚注 1	ALTER FUNCTION ALTER PACKAGE ALTER PACKAGE BODY ALTER PROCEDURE CREATE FUNCTION CREATE LIBRARY CREATE PACKAGE CREATE PACKAGE BODY CREATE PROCEDURE DROP FUNCTION DROP LIBRARY DROP PACKAGE DROP PACKAGE BODY DROP PROCEDURE
PROFILE	ALTER PROFILE CREATE PROFILE DROP PROFILE
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM
ROLE	ALTER ROLE CREATE ROLE DROP ROLE SET ROLE
ROLLBACK SEGMENT	ALTER ROLLBACK SEGMENT CREATE ROLLBACK SEGMENT DROP ROLLBACK SEGMENT
SCHEMA_DDL	スキーマ・オブジェクト(例: 表、索引、列)の作成、変更または削除を行うすべての <i>DDL</i> 文

キーワード	関連するSQL文
	ノート: SCHEMA_NAME および OBJECT_NAME は NULL 以外である必要があります。
SEQUENCE	ALTER SEQUENCE CREATE SEQUENCE DROP SEQUENCE
SYNONYM	CREATE PUBLIC SYNONYM CREATE SYNONYM DROP PUBLIC SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT SQL_statements NOAUDIT SQL_statements
TABLE	CREATE TABLE ALTER TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE DROP TABLESPACE ALTER TABLESPACE
TRIGGER	ALTER TRIGGER CREATE TRIGGER DISABLE ALL TRIGGERS DISABLE TRIGGER DROP TRIGGER ENABLE ALL TRIGGERS ENABLE TRIGGER
TYPE	ALTER TYPE ALTER TYPE BODY CREATE TYPE CREATE TYPE BODY DROP TYPE DROP TYPE BODY
USER	ALTER USER CREATE USER DROP USER
VIEW	CREATE VIEW DROP VIEW

脚注1

Javaスキーマ・オブジェクト(ソース、クラス、リソース)は、SQL文をスキップ(無視)するためのプロシージャと同等とみなされます。

関連項目:

SKIPおよびUNSKIPオプションの使用例を示した次の各項

- [DBMS_LOGSTDBY.SKIPによる特定のスキーマ・オブジェクトに対する変更の防止](#)
- [DDL文のスキップ・ハンドラの設定](#)
- [ロジカル・スタンバイ・データベースの変更](#)
- [ロジカル・スタンバイ・データベースでの表の追加または再作成](#)

C.14.1 DBLINKSを使用するDDL文

SQL Applyは、データベース・リンクを参照するDDL文を正しく適用できない場合があります。

このような文の例を次に示します。

```
CREATE TABLE tablename AS SELECT * FROM bar@dblink
```

これは、ロジカル・スタンバイ・データベースで *dblink* がプライマリ・データベースと同じデータベースを指さないことがあるためです。このようなDDL文の実行中にSQL Applyに障害が発生した場合は、表の作成にDBMS_LOGSTDBY.INSTANTIATE_TABLEプロシージャを使用して、SQL APPLYの操作を再開します。

C.14.2 ロジカル・スタンバイ・プロセス上のAUD\$およびFGA_LOG\$のレプリケーション

ロジカル・スタンバイでは、監査およびファイングレイン監査がサポートされています。

プライマリ・データベースでAUD\$およびFGA_AUD\$表に対して行われた変更は、ロジカル・スタンバイにレプリケートされます。

AUD\$表およびFGA_AUD\$表の両方に、DBID列があります。DBID値がプライマリ・データベースの値の場合、その行は、プライマリでのアクティビティに基づいてロジカル・スタンバイにレプリケートされています。DBID値がロジカル・スタンバイ・データベースの値の場合、その行は、ロジカル・スタンバイのローカル・アクティビティの結果として挿入されています。

ロジカル・スタンバイ・データベースで、プライマリ・ロールをロールの推移(スイッチオーバーまたはフェイルオーバー)の結果とみなした後、新しいプライマリ(元のロジカル・スタンバイ)および新しいロジカル・スタンバイ(元のプライマリ)のAUD\$表およびFGA_AUD\$表は、必ず同期されるとはかぎりません。このため、新しいプライマリ・データベースのAUD\$表またはFGA_AUD\$表のすべての行が、新しいロジカル・スタンバイ・データベースに存在するとはかぎりません。ただし、データベースがプライマリ・ロールの際に挿入されたAUD\$およびFGA_LOG\$のすべての行はレプリケートされ、ロジカル・スタンバイ・データベースに存在します。

C.15 分散トランザクションおよびXAのサポート

複数の異なる方法を使用して分散トランザクションを実行できます。

- データベース・リンクを使用した調整方法により複数のデータベース内の表を変更します。
- 提供PL/SQLパッケージ内でDBMS_XAパッケージによって公開されたXAインタフェース、あるいはOCIまたはJDBCライブラリを介してXAインタフェースを使用します。XAインタフェースによって、X/Open分散トランザクション処理(DTP)アーキテクチャが実装されます。

これら2つの方法のいずれかを使用した分散トランザクション中に、プライマリ・データベースに対して変更が行われると、ロジカル・スタンバイ・データベースにレプリケートされます。

ただし、分散トランザクションの状態はレプリケートされません。ロジカル・スタンバイ・データベースは、その種のトランザクションの未決定、または準備状態は継承しません。またXAトランザクション用にプライマリ・データベースで使用されたものと同じグローバル・トランザクション識別子はレプリケートしません。その結果、分散トランザクションにコミットする前にロジカル・スタンバイ・データベースにフェイルオーバーすると、変更がロジカル・スタンバイにロールバックされます。プライマリ・データベース上の分散トランザクションが準備状態で、2段階から成るコミット・プロトコルのうち最初の段階を正常に終了した場合でも、ロールバックが発生します。スイッチオーバー操作はすべてのアクティブ分散トランザクションの完了を待機するので、この制約には影響されません。

XAトランザクションは次の2つの方法で実行できます。

- 密結合(異なるXAブランチがロックを共有する)
- 疎結合(異なるXAブランチがロックを共有しない)

疎結合のXAブランチによって行われる変更のレプリケーションは、COMPATIBLEパラメータの値に関係なくサポートされます。

Oracle RACプライマリ上で緊密に連結されたブランチ(11gリリース1で新たに提供)への変更のレプリケーションは、COMPATIBLE=11.2以上でのみサポートされています。

C.16 SecureFile LOBのサポート

SecureFile LOBは、データベースの互換性レベルが11.2以降に設定されている場合にサポートされます。

プライマリ・データベースでは、SecureFile LOB列で透過的なデータ暗号化およびデータ圧縮を使用できます。

SecureFile LOB列の重複除外およびSecureFile Database File System (DBFS)操作は完全にサポートされています。

SQL Applyがサポートされていない操作によって生成されたREDOを検出すると、「ORA-16211: サポートされていないレコードが、アーカイブREDOログで見つかりました。」エラーとともに停止します。続行するには、DBMS_LOGSTDBY.SKIPを使用した該当する表にスキップ・ルールを追加し、SQL Applyを再開します。

C.17 データベース・ファイル・システム(DBFS)のサポート

データベース・ファイル・システム(DBFS)では、データベース表に格納されたファイルおよびディレクトリの上に標準ファイル・システム・インタフェースが作成されるため、データベースに格納されたファイルへのアクセスおよび管理が簡単になります。

ロジカル・スタンバイでは、データベース・ファイル・システム(DBFS)がサポートされています。DBFSの詳細は、[『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』](#)を参照してください。

C.18 文字セットに関する考慮事項

文字セットについては、注意が必要な考慮事項があります。

- プライマリ・データベースとロジカル・スタンバイ・データベースでの使用文字セットが異なるData Guard構成を持つことはサポートされません。
- マルチテナント・コンテナ・データベース(CDB)が混合文字セットを持つ構成は、ローリング・アップグレードにDBMS_ROLLINGを使用するときのみサポートされます。混合文字セットは、CDB\$ROOTとCDBの1つ以上のプラグブル・データベース(PDB)が異なる文字セットを持つことを意味します。

C.19 DBMS_ROLLINGアップグレードの場合でのみ使用可能な追加のPL/SQLパッケージのサポート

特定のパッケージのレプリケーションは、DBMS_ROLLINGパッケージを使用して実行したローリング・アップグレードの場合でのみ使用可能です。

影響を受けるパッケージは次のとおりです。

- DBFS
 - DBMS_DBFS_CONTENT_ADMIN
 - DBMS_DBFS_SFS
 - DBMS_DBFS_SFS_ADMIN
- 軽量セキュリティ
 - XS_ACL
 - XS_DATA_SECURITY
 - XS_NAMESPACE
 - XS_PRINCIPAL
 - XS_ROLESET
 - XS_SECURITY_CLASS
- Oracle Streams Advanced Queuing(AQ)
 - DBMS_AQ
 - DBMS_AQJMS
 - DBMS_AQADM (次のプロシージャを除く。SCHEDULE_PROPAGATION、RECOVER_PROPAGATION、

UNSCHEDULE_PROPAGATION、ALTER_PROPAGATION_SCHEDULE、ENABLE_PROPAGATION_SCHEDULEおよび
DISABLE_PROPAGATION_SCHEDULE)

- Oracle Text
 - CTX_ADM
 - CTX_ANL
 - CTX_CLS
 - CTX_DDL
 - CTX_DOC
 - CTX_ENTITY
 - CTX_OUTPUT
 - CTX_QUERY
 - CTX_THES
 - CTX_TREE
- スケジューラ
 - DBMS_SCHEDULER
- XDB関連
 - DBMS_RESCONFIG
 - DBMS_XDB_CONFIG (特定のプロシージャはサポートされていません。(詳細は、[『Oracle XML DB開発者ガイド』](#)を参照してください。)
 - DBMS_XDB_REPOS
 - DBMS_XDBRESOURCE
 - DBMS_XDB_VERSION
 - DBMS_XDBZ (特定のプロシージャはサポートされていません。(詳細は、[『Oracle XML DB開発者ガイド』](#)を参照してください。)

関連項目:

- [DBMS_ROLLINGを使用したローリング・アップグレードの実行](#)
- 軽量セキュリティ・パッケージの詳細は、[『Oracle Database Real Application Security管理者および開発者ガイド』](#)を参照してください。
- DBMS_SCHEDULER、XDB関連およびDBFS関連パッケージの詳細は、[『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』](#)を参照してください。

D Oracle Data GuardとOracle Real Application Clusters

Oracle Data Guardは、シングル・インスタンスとOracle Real Application Clusters(Oracle RAC)の複数インスタンス・データベースを自由に組み合わせて構成することができます。

この付録ではOracle Data GuardをOracle RACデータベースと一緒に使用する場合の構成要件と考慮事項をまとめます。内容は次のとおりです。

- [Oracle RAC環境でのスタンバイ・データベースの構成](#)
- [Oracle RAC環境での構成に関する考慮事項](#)

D.1 Oracle RAC環境でのスタンバイ・データベースの構成

スタンバイ・データベースを構成すると、Oracle Real Application Clusters (Oracle RAC)を使用しているプライマリ・データベースを保護できます。

次の表では、プライマリ・データベースとスタンバイ・データベースのインスタンスの可能な組合せについて説明します。

インスタンスの組合せ	単一インスタンス・スタンバイ・データベース	複数インスタンス・スタンバイ・データベース
単一インスタンス・プライマリ・データベース	可	可
複数インスタンス・プライマリ・データベース	可	可

それぞれの組合せでは、プライマリ・データベースの各インスタンスが、REDOデータをスタンバイ・データベースのインスタンスへ転送します。Oracle Database 12cリリース2 (12.2.0.1)以降、複数インスタンスのREDO Applyも実行できます。

D.1.1 複数インスタンスのREDO Applyの設定

Oracle Database 12cリリース2 (12.2.0.1)以降、新しい `INSTANCES [ ALL | integer]`句がSQL `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE`コマンドで使用できます。

Active Data Guard環境において複数インスタンスのREDO Applyでインメモリー列ストアを使用するには、`enable_imc_with_mira`初期化パラメータをTRUEに設定します。

次の制限があります。

- この句は、Oracle Real Application Clusters (Oracle RAC)またはOracle RAC One Nodeデータベースのみに適用できます。
- プライマリ・データベースでSTANDBY NOLOGGING FOR DATA AVAILABILITYまたはSTANDBY NOLOGGING FOR LOAD PERFORMANCEを有効にした場合、複数インスタンスのREDO Applyは使用できません。

ALLオプションを指定すると、リカバリが開始した時点でオープン状態またはマウント状態になっているOracle RACスタンバイ・データベース内のすべてのインスタンスに対して、REDO Applyが実行されます。すべてのインスタンスは同じ状態(オープンまたはマウント)である必要があります。状態の混合は許可されません。

integerオプションは、REDO Applyが使用するインスタンスの数を、指定した数に制限します。integerには、1からスタンバ

イ・データベースのインスタンス数までの整数値を指定します。Redo Applyを実行するインスタンスはデータベースで選択され、ユーザーが特定のインスタンスを指定することはできません。

V\$RECOVERY_PROGRESSビューは、リカバリが開始されたインスタンス(MRP0プロセスが存在)にのみ移入されます。

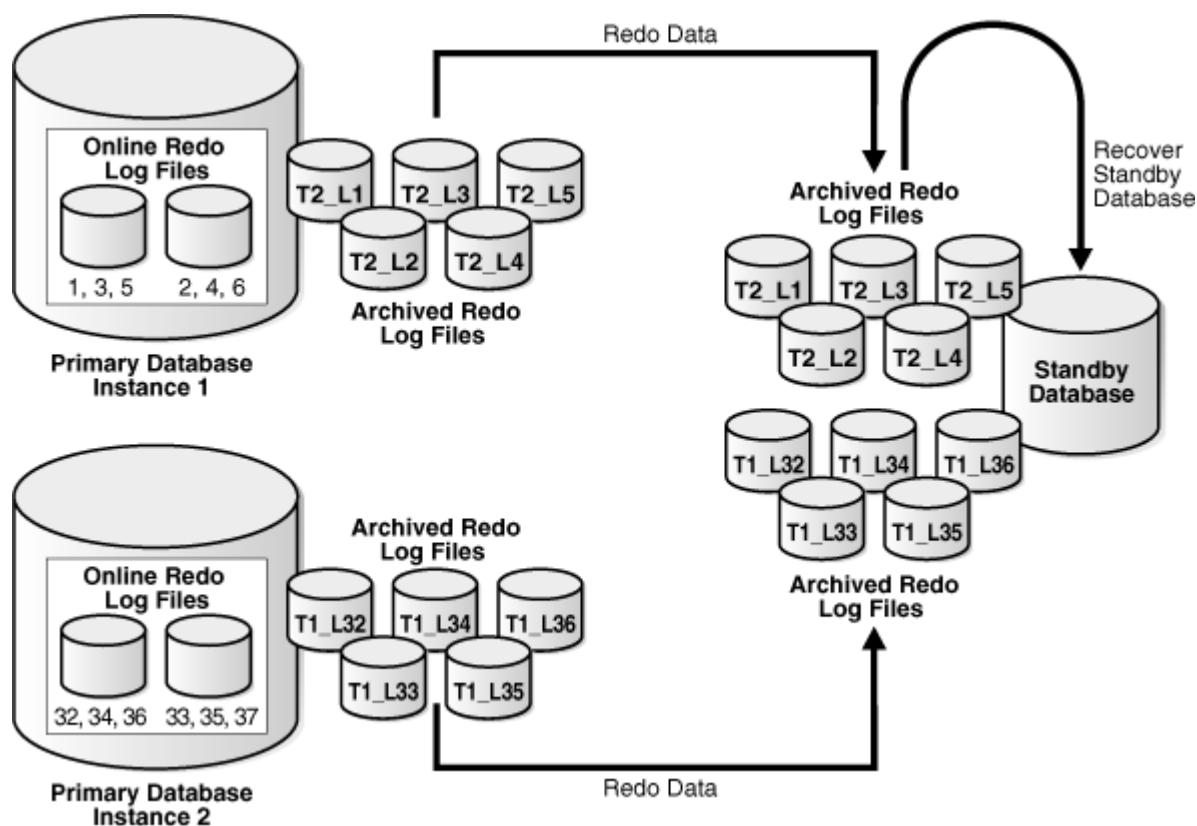
INSTANCES句を省略すると、リカバリはコマンドが発行されたインスタンスでのみ発生します。

リカバリ・プロセスではインスタンス内にREDOを送信するため、REDO Applyのパフォーマンスはネットワークの帯域幅および待ち時間と直接関係します。

D.1.2 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

この図では、プライマリ・データベース・インスタンスが2つあるOracle RACデータベース(複数インスタンス・プライマリ・データベース)が、単一インスタンス・スタンバイ・データベースへREDOデータを転送しています。

図D-1 複数インスタンス・プライマリ・データベースからのREDOデータの転送



この場合は、プライマリ・データベースのインスタンス1がローカル・アーカイブREDOログ・ファイル1、2、3、4、5にREDOデータをアーカイブし、スタンバイ・データベース宛先にREDOデータを転送するのに対し、インスタンス2がローカル・アーカイブREDOログ・ファイル32、33、34、35、36にREDOデータをアーカイブし、同じスタンバイ・データベース宛先にREDOデータを転送します。スタンバイ・データベースでは、アーカイブREDOログ・ファイルを適用する正しい順序が自動的に判断されます。

[図D-1](#)にはスタンバイREDOログは示されていませんが、プライマリの両方のインスタンスに対しスタンバイでスタンバイREDOログを構成することがベスト・プラクティスです。インスタンス1およびインスタンス2にあるプライマリ・オンラインREDOログ・ファイルからのREDOは、まずインスタンス1およびインスタンス2のスタンバイREDOログでそれぞれ受信され、次にアーカイブされます。

Oracle RAC環境でのプライマリ・データベースの構成方法

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。これを行うには、いくつかの準備ステップを実行する必要があり、それによりデータベースは、1つ以上のスタンバイ・データベースに対するプライマリ・データベースとして機能する準備が整います。

アーカイブREDOログ・ファイルおよびスタンバイREDOログ・ファイルの場所を指定するには、LOG_ARCHIVE_DEST_nおよびLOG_ARCHIVE_FORMATパラメータを定義します。

関連項目:

- スタンバイREDOログの詳細は、[「スタンバイREDOログの管理」](#)を参照してください
- フィジカル・スタンバイ・データベースを作成する際にプライマリ・データベースを構成する方法の詳細は、[「フィジカル・スタンバイ・データベースの作成」](#)を参照してください。
- ロジカル・スタンバイ・データベースを作成する際にプライマリ・データベースを構成する方法の詳細は、[「ロジカル・スタンバイ・データベースの作成」](#)を参照してください。

D.1.3 Oracle RACプライマリおよびスタンバイ・データベースの設定

Oracle RACプライマリ・データベースは、REDOデータをOracle RACスタンバイ・データベースに送信するように設定されている必要があります。Oracle RACスタンバイ・データベースは、REDOデータを受信するように設定する必要があります。

D.1.3.1 REDOデータを受信するためのOracle RACスタンバイ・データベースの構成

これらのステップでは、プライマリ・データベースからREDOデータを受信するようにOracle RACスタンバイ・データベースを構成する方法について説明します。

1. スタンバイ・データベースで、スタンバイREDOログを作成します。スタンバイREDOログのREDOログ・ファイルは、すべてのスタンバイ・データベース・インスタンスがアクセスできる場所(クラスタ・ファイル・システムやOracle ASMインスタンスなど)に存在する必要があります。スタンバイREDOログの作成の詳細は、[「スタンバイREDOログの管理」](#)を参照してください。
2. スタンバイREDOログ・アーカイブを、各スタンバイ・インスタンス上に構成します。スタンバイREDOログはすべてのスタンバイ・インスタンスがアクセス可能な場所にアーカイブする必要があります。また、各スタンバイ・データベース・インスタンスは、同じ場所にスタンバイREDOログをアーカイブするように構成しておく必要があります。

D.1.3.2 REDOデータを送信するためのOracle RACプライマリ・データベースの構成

REDOデータをOracle RACスタンバイ・データベースに送信するように、Oracle RACプライマリ・データベースの各インスタンスを構成します。これらは、REDOデータをOracle RACスタンバイ・データベースに送信するようにOracle RACプライマリ・データベースを構成する際に推奨されるベスト・プラクティスです。

1. 各プライマリ・データベース・インスタンスで同じLOG_ARCHIVE_DEST_nパラメータを使用して、REDOデータを特定のスタンバイ・データベースに送信する。
2. 特定のスタンバイ・データベースに対応する各LOG_ARCHIVE_DEST_nパラメータのSERVICE属性を同じネット・サービス名に設定する。
3. ネット・サービス名は、アドレス・リストを含むOracle Net接続記述子に解決する必要があり、そのアドレス・リストには、各スタンバイ・データベース・インスタンスの接続データが含まれる必要がある。

REDOデータを別のデータベースに送信するようにOracleデータベース・インスタンスを構成する方法は、[「REDOデータを送信するためのOracleデータベースの構成」](#)を参照してください。

D.2 Oracle RAC環境での構成に関する考慮事項

Oracle Real Application Clusters (Oracle RAC)では、Oracle Data Guardの構成時にアーカイブREDOログ・ファイル名の形式とデータ保護モードに関して一定の要件があります。

次のトピックを参照してください。

- [アーカイブREDOログ・ファイル名の形式](#)
- [データ保護モード](#)

D.2.1 アーカイブREDOログ・ファイル名の形式

アーカイブREDOログ・ファイル名の形式は、log_%parameterです。

%parameterには、[表D-1](#)のパラメータを1つ以上指定できます。

表D-1 LOG_ARCHIVE_FORMAT初期化パラメータのディレクティブ

ディレクティブ	説明
%a	データベース・アクティブ ID
%A	0(ゼロ)を埋め込んだデータベース・アクティブ ID
%d	データベース ID
%D	0(ゼロ)を埋め込んだデータベース ID
%t	インスタンス・スレッド番号
%T	0(ゼロ)を埋め込んだインスタンス・スレッド番号
%s	ログ・ファイル順序番号
%S	0(ゼロ)を埋め込んだログ・ファイル順序番号
%r	リセットログ ID
%R	0(ゼロ)を埋め込んだリセットログ ID

次に例を示します。

```
LOG_ARCHIVE_FORMAT = log%d_%t_%s_%r.arc
```

Oracle RACが、LOG_ARCHIVE_FORMATパラメータでアーカイブREDOログ・ファイルを一意に識別するためには、スレッド・パラメータ%tまたは%Tが必須です。

D.2.2 データ保護モード

Oracle RACプライマリ・データベースのいずれかのインスタンスでスタンバイ・データベースとの接続性が失われた場合、他のすべてのプライマリ・データベース・インスタンスでは、LOG_ARCHIVE_DEST_*n* REOPEN属性で指定された秒数の間、スタンバイ・データベースへのREDO送信を停止し、その後すべてのプライマリ・データベース・インスタンスでスタンバイ・データベースへの再接続が試みられます。

次に、Oracle RAC環境における保護モードの動作を説明します。

- 最大保護の構成

接続を失った宛先が最後のSYNC宛先の場合、インスタンスは接続を失い、停止します。Oracle RAC構成内でスタンバイ宛先への接続を維持しているインスタンスは、接続を失ったインスタンスをリカバリし、スタンバイ宛先へのREDO送信を継続します。Oracle RAC構成内のすべてのインスタンスが最後のスタンバイ宛先への接続を失った場合にのみ、プライマリ・データベースが停止します。

- 最大可用性構成と最大パフォーマンス構成

Oracle RAC構成内でスタンバイ宛先への接続を維持しているインスタンスは、接続を失ったインスタンスをリカバリし、スタンバイ宛先へのREDO送信を継続します。Oracle RAC構成内のすべてのインスタンスがスタンバイ宛先への接続を失うと、プライマリ・データベースは最大パフォーマンス・モードで操作を続行します。最大パフォーマンス・モードでは、スタンバイ全体の障害時を除き、データ消失が最小限に抑えられます。

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

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

このトピックでは、Oracle Recovery Managerを使用してスタンバイ・データベースを作成する方法を説明します。

- [前提条件](#)
- [RMANを使用したスタンバイ・データベースの作成の概要](#)
- [DUPLICATEコマンドを使用したスタンバイ・データベースの作成](#)

E.1 前提条件

RMANを使用してスタンバイ・データベースを作成する前に、データベース複製について理解している必要があります。

DUPLICATEコマンドを使用してRMANでスタンバイ・データベースを作成するので、DUPLICATEコマンド・エントリについても理解している必要があります。

また、RMANを使用してスタンバイ・データベースを作成する前に、スタンバイ・データベースの作成方法も理解しておいてください。

関連項目:

- バックアップを実行する方法の詳細は、[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)を参照してください
- RMANのDUPLICATEコマンドの詳細は、[Oracle Databaseバックアップおよびリカバリ・リファレンス](#)を参照してください。
- [フィジカル・スタンバイ・データベースの作成](#)
- [ロジカル・スタンバイ・データベースの作成](#)

E.2 RMANを使用したスタンバイ・データベースの作成の概要

RMANを使用してスタンバイ・データベースを作成する目的および基本概念については、この項を参照してください。

E.2.1 RMANを使用したスタンバイ・データベースの作成の目的

プライマリ・データベースのバックアップからスタンバイ・データベースを作成するには、手動で行うか、RMANのDUPLICATEコマンドを使用します。

プライマリ・データベースのバックアップからスタンバイ・データベースを作成するには、手動で行うか、RMANのDUPLICATEコマンドを使用します。RMANを使用してスタンバイ・データベースを作成すると、手動による作成と比べて次のメリットがあります。

- プライマリ・データベースで現在使用しているファイルをコピーしてスタンバイ・データベースを作成できます。バックアップは必要ありません。
- プライマリ・データベースのバックアップをスタンバイ・サイトにリストアしてスタンバイ・データベースを作成できます。したがって、スタンバイ・データベースの作成中にプライマリ・データベースが影響を受けることはありません。
- Oracle Managed Files(OMF)などのファイルやディレクトリ構造の名前の変更が自動化されます。
- アーカイブREDOログ・ファイルをバックアップからリストアされ、スタンバイ・データベースとプライマリ・データベースが同期化

されるようにメディア・リカバリが実行されます。

E.2.2 RMANを使用したスタンバイ・データベースの作成の基本概念

RMANを使用してスタンバイ・データベースを作成する手順は、複製データベースの作成とほぼ同じです。

スタンバイ・データベース固有の問題に対処するために、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』で説明されている複製手順を変更する必要があります。

Duplicateコマンドを使用してスタンバイ・データベースを作成するには、ターゲットとしてプライマリ・データベースに接続し、FOR STANDBYオプションを指定する必要があります。スタンバイ・データベースに接続してスタンバイ・データベースを追加作成することはできません。RMANは、制御ファイルをリストアおよびマウントして、スタンバイ・データベースを作成します。RMANではプライマリ・データベースの制御ファイルの既存のバックアップを使用できるので、スタンバイ・データベース専用制御ファイルのバックアップを作成する必要はありません。

FOR STANDBY OPTIONを指定せずにDUPLICATEで作成した複製データベースと異なり、スタンバイ・データベースには新しいDBIDが設定されません。したがって、スタンバイ・データベースをリカバリ・カタログに登録しないでください。

E.2.2.1 アクティブ・データベース複製とバックアップベース複製

スタンバイ・データベースの作成にRMANを使用する場合は、アクティブ複製とバックアップベース複製のどちらかを選択する必要があります。

FROM ACTIVE DATABASEを指定すると、RMANはデータファイルをプライマリ・データベースからスタンバイ・データベースに直接コピーします。プライマリ・データベースをマウントまたはオープンする必要があります。

FROM ACTIVE DATABASEを指定しないと、RMANはバックアップベース複製を実行します。プライマリ・データファイルのバックアップがスタンバイ・データベースにリストアされます。スタンバイ・データベースの作成およびリカバリに必要なバックアップおよびアーカイブREDOログ・ファイルはすべて、スタンバイ・ホストのサーバー・セッションでアクセスする必要があります。SET UNTILコマンドを実行する場合を除いて、RMANは最新のデータファイルをリストアします。

E.2.2.2 RMAN環境におけるDB_UNIQUE_NAMEの値

FOR STANDBYオプションを指定せずにDUPLICATEコマンドを使用して作成された複製データベースと異なり、スタンバイ・データベースには新しいDBIDが設定されません。

Oracle Data Guard環境でRMANを使用する場合、リカバリ・カタログに接続する必要があります。リカバリ・カタログには、環境内のすべてのプライマリ・データベースとスタンバイ・データベースに関するメタデータを格納できます。スタンバイ・データベースは、リカバリ・カタログに明示的に登録しないでください。

Oracle Data Guard環境内のデータベースは、初期化パラメータ・ファイルのDB_UNIQUE_NAMEパラメータによって一意に識別する必要があります。Oracle Data Guard環境でRMANを正常に動作させるには、同じDBIDを持つすべてのデータベース間でDB_UNIQUE_NAMEが一意である必要があります。

関連項目:

Oracle Data Guard環境でのRMAN操作の概要は、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。

E.2.2.3 スタンバイ・データベースのリカバリ

デフォルトで、RMANはスタンバイ・データベースの作成後、リカバリを実行しません。

RMANはスタンバイ・データベースをマウントしたままにしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。RMANは接続を切断し、スタンバイ・データベースのメディア・リカバリを実行しません。

RMANでスタンバイ・データベースの作成後にリカバリを行う場合、リカバリに対してスタンバイ制御ファイルが使用可能である必要があります。次の条件を満たしている必要があります。

- スタンバイ・データベースのリカバリ終了時刻は、スタンバイ制御ファイルのチェックポイントSCN以上である必要があります。
- スタンバイ制御ファイルのチェックポイントSCNが含まれているアーカイブREDOログ・ファイルは、リカバリ用にスタンバイ・サイトから使用可能である必要があります。

これらの条件を満たしていることを確認する方法として、プライマリ・データベースに制御ファイルをバックアップ後、ALTER SYSTEM ARCHIVE LOG CURRENT文を実行します。この文は、プライマリ・データベースのオンラインREDOログ・ファイルをアーカイブします。その後、最新のアーカイブREDO・ログ・ファイルをRMANでバックアップするか、またはアーカイブREDOログ・ファイルをスタンバイ・サイトに移動します。

DUPLICATEコマンドのDORECOVERオプションを使用して、RMANでスタンバイ・データベースをリカバリするように指定します。RMANは、スタンバイ・データベース・ファイルの作成後に次のステップを実行します。

1. RMANはメディア・リカバリを開始します。アーカイブREDOログ・ファイルを必要とするリカバリで、そのログ・ファイルがディスクにない場合、RMANはバックアップのリストアを試みます。
2. RMANは、指定された時間、システム変更番号(SCN)またはログ・ファイル順序番号にスタンバイ・データベースをリカバリします。そのいずれも指定されていない場合は、最新の生成済アーカイブREDOログ・ファイルにリカバリします。
3. RMANは、メディア・リカバリの完了後、スタンバイ・データベースをマウントしたままにしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。

E.2.2.3.1 スタンバイ・データベースREDOログ・ファイル

RMANは、スタンバイ・データベースでスタンバイREDOログ・ファイルを自動的に作成します。

ログ・ファイルは作成後、スタンバイ・データベースによってログ・ファイルの通常のルールに従って管理およびアーカイブされます。

バックアップベース複製を使用した場合、スタンバイ・データベースでスタンバイREDOログ・ファイルに名前を付けるときの唯一のオプションは、スタンバイ制御ファイルで指定されるログ・ファイルのファイル名です。スタンバイでのログ・ファイル名をプライマリのファイル名と異なる名前にする必要がある場合は、スタンバイ初期化パラメータ・ファイルでLOG_FILE_NAME_CONVERTを設定して、スタンバイREDOログのファイル名を指定する方法を選択できます。

スタンバイ・データベースでスタンバイREDOログ・ファイルのファイル名を指定する際、次の制限事項に注意してください。

- プライマリ・データベースおよびスタンバイ・データベースでログ・ファイルに異なるネーミング規則を使用する場合、スタンバイREDOログ・ファイルにはLOG_FILE_NAME_CONVERTパラメータを使用して名前を付ける必要があります。
- スタンバイREDOログ・ファイルの名前の変更には、SET NEWNAMEまたはCONFIGURE AUXNAMEコマンドを使用できません。
- スタンバイREDOログ・ファイルのファイル名の指定には、DUPLICATEコマンドのLOGFILE句を使用できません。
- スタンバイ・データベースでのスタンバイREDOログ・ファイル名をプライマリREDOログ・ファイル名と同じ名前にする場合、DUPLICATEコマンドのNOFILENAMECHECK句を指定する必要があります。指定しない場合、スタンバイ・データベースが異なるホスト上で作成されていても、RMANによりエラーが発生します。

E.2.2.4 スタンバイ・データベースのパスワード・ファイル

アクティブ・データベース複製を使用すると、スタンバイ・データベースのパスワード・ファイルはターゲット・データベースのパスワード・ファイルの完全コピーである必要があるため、RMANは常にパスワード・ファイルをスタンバイ・ホストにコピーします。

アクティブ・データベース複製を使用すると、スタンバイ・データベースのパスワード・ファイルはターゲット・データベースのパスワード・ファイルの完全コピーである必要があるため、RMANは常にパスワード・ファイルをスタンバイ・ホストにコピーします。その場合、PASSWORD FILE句は必要ありません。補助インスタンスの既存のパスワード・ファイルはすべて上書きされます。バックアップベース複製では、Oracle Data Guardによるログの送信のために、プライマリで使用されているパスワード・ファイルをスタンバイにコピーする必要があります。

E.3 DUPLICATEコマンドを使用したスタンバイ・データベースの作成

スタンバイ・データベースの作成手順は、基本的にRMANの複製手順と同じです。

RMAN複製手順の詳細は、[Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド](#)を参照してください。

E.3.1 アクティブ・データベース複製を使用したスタンバイ・データベースまたは遠隔同期インスタンスの作成

スタンバイ・データベースまたは遠隔同期インスタンスのいずれかを作成するには、アクティブ・データベース複製を使用できます。

アクティブ・データベース複製によるスタンバイ・データベースの作成

RMANのDUPLICATEを使用してプライマリ・データベースのアクティブなファイルからスタンバイ・データベースを作成するには、FOR STANDBYとFROM ACTIVE DATABASEの両方のオプションを指定します。DORECOVERなどの他のオプションも指定できます。次に例を示します。

```
DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE DORECOVER;
```

アクティブ・データベース複製による遠隔同期インスタンスの作成

Oracle Database 12cリリース12.2 (12.2.0.1)以降、RMANのDUPLICATEコマンドを使用して、プライマリ・データベースでアクティブなファイルから遠隔同期インスタンスを作成することもできます。このためには、コマンドラインでSTANDBYのかわりにFARSYNCを使用します(DORECOVERオプションは指定しないでください。遠隔同期インスタンスには許可されていません)。次に例を示します。

```
DUPLICATE TARGET DATABASE FOR FARSYNC FROM ACTIVE DATABASE;
```

アクティブ・データベース複製を使用してスタンバイ・データベースまたは遠隔同期インスタンスを作成するステップ

次のステップでアクティブ・データベース・ファイルからスタンバイ・データベースを作成しますが、これらのステップを使用してアクティブ・データベース・ファイルから遠隔同期インスタンスを作成することもできます。このステップでは、スタンバイ・ホスト(または遠隔同期インスタンス)およびプライマリ・データベース・ホストのディレクトリ構造が同じであることを前提にしています。

1. 補助データベース・インスタンスを準備します。詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド](#)』を参照してください。

アクティブ・データベース複製を使用するため、補助インスタンス用にパスワード・ファイルを作成し、Oracle Net接続を確立する必要があります。これは、複製操作中に上書きされる一時的なパスワード・ファイルです。

2. スタンバイ制御ファイル、データファイル、オンラインREDOログおよび一時ファイルの名前の指定方法を決定します。このステップの詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド](#)』を参照してください。

この使用例では、スタンバイ・データベース・ファイルには、プライマリ・データベース・ファイルと同じ名前を付けます。

3. RMANを起動して構成します。詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザズ・ガイド](#)』を参照してください。
4. DUPLICATEコマンドを実行します。

次の例に、アクティブ複製でのDUPLICATEの使用方法を示します。この例では、プライマリ・データベース・ファイルがスタンバイ・データベース・ファイルと同じ名前であるため、NOFILENAMECHECKオプションが必要です。SPFILEのSET句は、ログの送信が正常に行われるために必要です。db_unique_nameを設定して、カタログおよびOracle Data Guardでのデータベースがプライマリとは異なるものであると識別できるようにする必要があります。必要に応じて、DORECOVERオプションを指定し、スタンバイの作成後にデータベースをリカバリします(DORECOVERは遠隔同期インスタンスの有効なオプションではありません)。

```
DUPLICATE TARGET DATABASE
FOR STANDBY
FROM ACTIVE DATABASE
DORECOVER
SPFILE
SET "db_unique_name"="foou" COMMENT "Is a duplicate"
SET LOG_ARCHIVE_DEST_2="service=inst3 ASYNC REGISTER
VALID_FOR=(online_logfile,primary_role)"
SET FAL_SERVER="inst1" COMMENT "Is primary"
NOFILENAMECHECK;
```

RMANは、サーバー・パラメータ・ファイルをスタンバイ・ホストに自動的にコピーし、そのサーバー・パラメータ・ファイルを使用して補助インスタンスを起動してバックアップ制御ファイルをリストアし、必要なデータベース・ファイルおよびアーカイブ REDOLOGをすべてスタンバイ・ホストにネットワークを介してコピーします。RMANはスタンバイ・データベースをリカバリしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。

スタンバイではなく遠隔同期インスタンスを作成している場合、コマンドは次のようにSTANDBYがFARSYNCで置き換えられることを除いて同じになります。

```
DUPLICATE TARGET DATABASE
FOR FARSYNC
FROM ACTIVE DATABASE
SPFILE
SET "db_unique_name"="foou" COMMENT "Is a duplicate"
SET LOG_ARCHIVE_DEST_2="service=inst3 ASYNC REGISTER
VALID_FOR=(online_logfile,primary_role)"
SET FAL_SERVER="inst1" COMMENT "Is primary"
NOFILENAMECHECK;
```

ノート:



プライマリ・データベースが Oracle Real Application Clusters (Oracle RAC) データベースで、ターゲット・スタンバイ・データベースまたは遠隔同期インスタンスが単一インスタンスの Oracle データベースになる場合は、前述の RMAN の例にコマンド SET CLUSTER_DATABASE="FALSE"を追加します。

E.3.2 バックアップベース複製によるスタンバイ・データベースの作成

スタンバイ・データベースまたは遠隔同期インスタンスのいずれかを作成するには、バックアップベース複製を使用できます。

バックアップベース複製によるスタンバイ・データベースの作成

スタンバイ・データベースをバックアップから作成するには、FOR STANDBYを指定しますが、FROM ACTIVE DATABASEは指定しません。DORECOVERなどの他のオプションも指定できます。次に例を示します。

```
DUPLICATE TARGET DATABASE FOR STANDBY BACKUP LOCATION '+DATA/BACKUP' DORECOVER;
```


バックアップベース複製による遠隔同期インスタンスの作成

Oracle Database 12cリリース12.2 (12.2.0.1)以降、RMANのDUPLICATEコマンドを使用してData Guard遠隔同期インスタンスを作成するためにバックアップベース複製を使用することもできます。このためには、コマンドラインでSTANDBYのかわりにFARSYNCを使用します(DORECOVERオプションは指定しないでください。遠隔同期インスタンスには許可されていません)。次に例を示します。

```
DUPLICATE TARGET DATABASE FOR FARSYNC BACKUP LOCATION '+DATA/BACKUP' ;
```

バックアップからスタンバイ・データベースまたは遠隔同期インスタンスを作成するステップ:

次のステップでバックアップからスタンバイ・データベースを作成しますが、これらのステップを使用してバックアップから遠隔同期インスタンスを作成することもできます。このステップでは、スタンバイ・ホスト(または遠隔同期インスタンス)およびプライマリ・データベース・ホストのディレクトリ構造が同じであることを前提にしています。

1. 『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』に記載されているように、データベース・バックアップおよびアーカイブREDOログを複製ホスト上の補助インスタンス用に提供します。
2. 補助データベース・インスタンスを準備します。詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。
3. スタンバイ制御ファイル、データファイル、オンラインREDOログおよび一時ファイルの名前の指定方法を決定します。このステップの詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。
この使用例では、スタンバイ・データベース・ファイルには、プライマリ・データベース・ファイルと同じ名前を付けます。
4. RMANを起動して構成します。詳細は、『[Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド](#)』を参照してください。
5. DUPLICATEコマンドを実行します。

次の例に、バックアップベース複製でのDUPLICATEの使用法を示します。この例では、プライマリ・データベース・ファイルがスタンバイ・データベース・ファイルと同じ名前であるため、NOFILENAMECHECKオプションが必要です。必要に応じて、DORECOVERオプションを指定し、スタンバイの作成後にデータベースをリカバリします(DORECOVERは遠隔同期インスタンスの有効なオプションではありません)。

```
DUPLICATE TARGET DATABASE
FOR STANDBY
DORECOVER
SPFILE
SET "db_unique_name"="foou" COMMENT "Is a duplicate"
SET LOG_ARCHIVE_DEST_2="service=inst3 ASYNC REGISTER
VALID_FOR=(online_logfile,primary_role)"
SET FAL_SERVER="inst1" COMMENT "Is primary"
NOFILENAMECHECK;
```

RMANは、サーバー・パラメータ・ファイルをスタンバイ・ホストに自動的にコピーし、そのサーバー・パラメータ・ファイルを使用して補助インスタンスを起動し、必要なデータベース・ファイルおよびアーカイブREDOログをすべてスタンバイ・ホストにリストアします。RMANはスタンバイ・データベースをリカバリしますが、スタンバイ・データベースを手動または管理リカバリ・モードには設定しません。

F アーカイブ・トレースの設定

Oracleデータベースは、LOG_ARCHIVE_TRACE初期化パラメータを使用して、ログ・アーカイブおよびREDO転送アクティビティに関する包括的なトレース情報の生成を可能にし、制御します。

LOG_ARCHIVE_TRACEを非ゼロ値に設定したときに出力されるその他のトレースは、アーカイブ・プロセス、RFSプロセス、LGWRプロセス、SYNCプロセス、ASYNCRプロセス、フォアグラウンド・プロセス、MRPプロセス、リカバリ・プロセス、ログ適用プロセス、起動プロセス、停止プロセスおよびREDO転送サービスを使用するその他のプロセスのトレース・ファイルに表示される場合があります。トレース情報は、Automatic Diagnostic Repositoryに書き込まれます。

関連項目:

- LOG_ARCHIVE_TRACE初期化パラメータおよびその有効な値の詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。
- Automatic Diagnostic Repositoryの詳細は、[『Oracle Database管理者ガイド』](#)を参照してください。

F.1 LOG_ARCHIVE_TRACE初期化パラメータの設定

LOG_ARCHIVE_TRACEパラメータを有効化、無効化または変更するには、ALTER SYSTEM SQL文を発行します。

LOG_ARCHIVE_TRACEパラメータの書式は次のとおりで、このとき、trace_levelは整数です。

```
LOG_ARCHIVE_TRACE=trace_level
```

次に、LOG_ARCHIVE_TRACEパラメータを設定する方法の例を示します。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_TRACE=15;
```

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・レベルを組み合わせることができます。たとえば、この例では、LOG_ARCHIVE_TRACEパラメータを15に設定したため、トレース・レベルが1、2、4および8に設定されます。

有効なLOG_ARCHIVE_TRACE値の完全なリストおよび詳細は、[『Oracle Databaseリファレンス』](#)を参照してください。

次に、ログ・ファイル387を2つの異なる宛先(サービスstandby1およびローカル・ディレクトリ/oracle/dbs)にアーカイブすることによってプライマリ・サイトに生成されるARC0トレース・データの例を示します。

ノート:



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->KCRRESTART ARCH: invoking ARC0 ARC0: changing ARC0 KCRRESTART->KCRRACTIVE ARCH: Initializing ARC0 ARCH: ARC0 invoked ARCH: STARTING ARCH PROCESSES COMPLETE ARC0 started with pid=8 ARC0: Archival started

次に示すトレース・データは、スタンバイ・サイトで、ディレクトリ/stbyにアーカイブREDOログ・ファイル387を受信し、それをスタンバイ・データベースに適用するリモート・ファイル・サーバー(RFS)プロセスによって生成されたものです。

level	trace output (sample)
(4)	RFS: Startup received from ARCH pid 9272
(4)	RFS: Notifier
(4)	RFS: Attaching to standby instance
(1)	RFS: Begin archive log# 2 seq# 387 thrd# 1
(32)	Propagating archive 5 destination version 0 to version 2
(32)	Propagating archive 5 state version 0 to version 1
(8)	RFS: Creating archive destination file: /stby/parc1_387.log
(16)	RFS: Archiving block 1 count 11
(1)	RFS: Completed archive log# 2 seq# 387 thrd# 1
(8)	RFS: Closing archive destination file: /stby/parc1_387.log
(16)	RFS: ArchivedLog entry added: /stby/parc1_387.log
(1)	RFS: Archivelog seq# 387 thrd# 1 available 04/02/99 09:40:53
(4)	RFS: Detaching from standby instance
(4)	RFS: Shutdown received from ARCH pid 9272

G 古い構文を使用したロールの推移の実行

Oracle Database 12cリリース1 (12.1)より前は、フィジカル・スタンバイ・データベースへのスイッチオーバーおよびフェイルオーバーを実行するプロシージャは異なっていました。

これらのプロシージャは引き続きサポートされていますが、「[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](#)」で説明している新しいプロシージャを使用することをお勧めします。

Oracle Database 12cリリース1 (12.1)よりも前のリリースを使用している場合、古いプロシージャを使用する必要があります。

次の内容について説明します。

- [フィジカル・スタンバイが関与するロールの推移のSQL構文](#)
- [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](#)
- [フィジカル・スタンバイ・データベースへのスイッチオーバーのトラブルシューティング](#)

G.1 フィジカル・スタンバイが関与するロールの推移のSQL構文

Oracle Database 12cリリース1 (12.1)では、フィジカル・スタンバイ・データベースへのスイッチオーバーおよびフェイルオーバー操作の実行のために新しいSQL構文が導入されています。

Oracle Database 12cリリース1 (12.1)では、フィジカル・スタンバイ・データベースへのスイッチオーバーおよびフェイルオーバー操作の実行のために新しいSQL構文が導入されています。特別な指示がないかぎり、古いプロシージャ(この付録で説明)の構文と新しいプロシージャ([「ロールの推移」](#)で説明)の構文を混在させないでください。

12cより前のフィジカル・スタンバイ・データベースに対するロールの推移の構文	12cのフィジカル・スタンバイ・データベースに対するロールの推移の構文
フィジカル・スタンバイ・データベースにスイッチオーバーするには、プライマリ・データベースで次の手順を実行します。	フィジカル・スタンバイ・データベースにスイッチオーバーするには：
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY;	SQL> ALTER DATABASE SWITCHOVER TO <i>target_db_name</i> [FORCE] [VERIFY];
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。	
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;	
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には、「 古い構文を使用したフィジカル・スタンバイ・データベースへのフェイルオーバーの実行 」のステップ 6 およびステップ 8)：	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には、以前必要だった 2 つの文のかわりに次の文を使用します。
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;	SQL> ALTER DATABASE FAILOVER TO <i>target_db_name</i> ;

AND

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

関連項目:

- SQL構文の詳細は、[『Oracle Database SQL言語リファレンス』](#)を参照してください。

G.1.1 古い構文を使用する場合の新しい機能

Oracle Database 12cリリース1 (12.1)から、アクティブなSQLセッションを終了するときにWITH SESSION SHUTDOWN句は不要になりました。

次の文を発行してアクティブなSQLセッションを自動的に終了できます。

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

さらに、Oracle RACプライマリ・データベースからフィジカル・スタンバイ・データベースにスイッチオーバーを実行する場合、1つを残してすべてのプライマリ・データベース・インスタンスを停止する必要はなくなりました。スイッチオーバーの完了後、すべてのインスタンスは自動的に停止します。

G.2 フィジカル・スタンバイ・データベースが関与するロールの推移

Oracle Databaseリリース12cリリース1 (12.1)よりも前のリリースでは、フィジカル・スタンバイ・データベースへのスイッチオーバーおよびフェイルオーバーを実行するSQL構文が異なっていました。

次に示すのは、12.1より前のリリースを実行している場合に使用する必要のあるプロシージャです。

- [古い構文を使用したフィジカル・スタンバイ・データベースへのスイッチオーバーの実行](#)
- [古い構文を使用したフィジカル・スタンバイ・データベースへのフェイルオーバーの実行](#)

関連項目:

スイッチオーバーおよびフェイルオーバーの準備方法の詳細は[「ロールの推移」](#)を参照してください。

G.2.1 古い構文を使用したフィジカル・スタンバイ・データベースへのスイッチオーバーの実行

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

このトピックでは、フィジカル・スタンバイ・データベースへのスイッチオーバーを実行する方法について説明します。

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

プライマリ・データベースでV\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせます。次に例を示します。

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

```
SWITCHOVER_STATUS
```

```
-----  
TO STANDBY
```

```
1 row selected
```

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

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

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

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

3. 元のプライマリ・データベースをマウントします。例:

```
SQL> STARTUP MOUNT;
```

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

4. ターゲット・スタンバイ・データベースでV\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせ、スイッチオーバー・ターゲットをプライマリ・ロールに切り替える準備ができていることを確認します。次に例を示します。

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

```
SWITCHOVER_STATUS
```

```
-----  
TO PRIMARY
```

```
1 row selected
```

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

5. ターゲット・フィジカル・スタンバイ・データベースで次のSQL文を発行して、プライマリ・ロールに切り替えます。

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

ノート:



ステップ 4 で実行した問合せで TO PRIMARY の値が戻された場合、WITH SESSION SHUTDOWN 句をスイッチオーバー文から省略できます。

6. 新しいプライマリ・データベースをオープンします。

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


7. 新しいフィジカル・スタンバイ・データベースでREDO Applyを開始します。

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

8. Oracle Data Guard構成のその他のフィジカル・スタンバイ・データベースのいずれかでREDO Applyが停止している場合は、これを再開します。

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

G.2.2 古い構文を使用したフィジカル・スタンバイ・データベースへのフェイルオーバーの実行

このステップでは、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方法について説明します。

1. 可能な場合は、プライマリ・データベースをマウントして、未送信のアーカイブされた現行のREDOをプライマリ・データベースからスタンバイ・データベースにフラッシュします。この操作が正常に終了した場合、プライマリ・データベースがデータ消失ゼロのデータ保護モードで実行されていない場合、データの消失がないようにフェイルオーバーすることができます。

ターゲット・スタンバイ・データベースでREDO Applyがアクティブであることを確認します。

プライマリ・データベースをマウントします。オープンはしないでください。プライマリ・データベースをマウントできない場合は、ステップ2に進みます。

次のSQL文をプライマリ・データベースで発行します。

```
SQL> ALTER SYSTEM FLUSH REDO TO target_db_name;
```

*target_db_name*には、プライマリ・データベースからフラッシュされたREDOを受信するスタンバイ・データベースのDB_UNIQUE_NAMEを指定します。

この文は、未送信のREDOをすべて、プライマリ・データベースからスタンバイ・データベースにフラッシュし、このREDOがスタンバイ・データベースに適用されるまで待機します。

この文がエラーなしに終了した場合は、ステップ5に進みます。この文でエラーが出た場合、または完了するまで待てないため、この文を停止しなければならない場合は、ステップ2に進みます。

2. スタンバイ・データベースが、プライマリ・データベースREDOスレッドそれぞれについて、最新のアーカイブREDOログ・ファイルを持っていることを確認します。

それには、ターゲット・スタンバイ・データベースでV\$ARCHIVED_LOGビューを問い合わせ、個々のREDOスレッドについて、最も大きなログ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#) -  
> OVER (PARTITION BY thread#) AS LAST from V$ARCHIVED_LOG;
```

THREAD	LAST
1	100

可能であれば、各プライマリ・データベースREDOスレッドに対する最新のアーカイブREDOログ・ファイルがスタンバイ・データベースになければ、これをコピーし、登録します。これは、REDO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';
```

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

次に例を示します。

```
SQL> SELECT THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE# FROM V$ARCHIVE_GAP;
```

THREAD#	LOW_SEQUENCE#	HIGH_SEQUENCE#
1	90	92

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

可能であれば、欠落しているすべてのアーカイブREDOログ・ファイルを、プライマリ・データベースからターゲット・スタンバイ・データベースにコピーして登録します。これは、REDO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';
```

- すべてのギャップが解決されるまでステップ3を繰り返します。(ステップ3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るため、ギャップを解決した後、行が戻されなくなるまで問合せを繰り返す必要があります。)

ステップ2からステップ4を実行した後、アーカイブREDOログ・ファイルのすべてのギャップを解決できない場合(障害の発生したプライマリ・データベースのホスト・システムへのアクセス権がない場合など)、フェイルオーバー時にデータが消失する可能性があります。

- ターゲット・スタンバイ・データベースに次のSQL文を発行して、Redo Applyを停止します。

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

- 受信したすべてのREDOデータの適用を終了します。

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

この文がエラーなしに終了した場合は、ステップ7に進みます。

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

ステップ3およびステップ4で解決されなかったREDOギャップがある場合、REDOギャップがあることを知らせるエラーが表示されます。

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

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

ACTIVATE文が完了した後、ステップ9に進みます。

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

それには、ターゲット・スタンバイ・データベースでV\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせます。次に例を示します。

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

```
SWITCHOVER_STATUS
```

```
TO PRIMARY
```

```
1 row selected
```

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

8. ターゲット・フィジカル・スタンバイ・データベースで次のSQL文を発行して、フィジカル・スタンバイをプライマリ・ロールに切り
替えます。

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

ノート:



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

9. 新しいプライマリ・データベースをオープンします。

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

10. 新しいプライマリ・データベースをただちにバックアップすることをお勧めします。
11. Oracle Data Guard構成のその他のフィジカル・スタンバイ・データベースのいずれかでREDO Applyが停
止している場合は、これを再開します。

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

12. 必要に応じて、障害の発生したプライマリ・データベースをリストアします。

フェイルオーバーの完了後、元のプライマリ・データベースは[「フラッシュバック・データベースを使用した障害が発生したプ
ライマリのスタンバイ・データベースへの変換」](#)または[「RMANバックアップを使用した障害が発生したプライマリのスタンバ
イ・データベースへの変換」](#)で説明する方法に従って、新しいプライマリ・データベースのフィジカル・スタンバイ・データベ
ースに変換できます。あるいは、[「フィジカル・スタンバイ・データベースの作成ステップ」](#)で説明した方法に従って、新しいプ
ライマリ・データベースのバックアップからフィジカル・スタンバイ・データベースとして再作成できます。

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

G.3 フィジカル・スタンバイ・データベースへのスイッチオーバーのトラブルシュー ティング

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時に発生する可能性がある問題の一部を次に示します。

ノート:

次のトラブルシューティングの各項は、Oracle Database 12c リリース 1 (12.1)よりも前のリリースで使用可能なプロシージャを使用した、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およびフェイルオーバーを実行する場合にのみ該当します。

- [REDOデータが転送されていないためスイッチオーバーできない](#)
- [ORA-01102エラーによりスイッチオーバーできない](#)
- [スイッチオーバー後にREDOデータが適用されない](#)
- [失敗したスイッチオーバーをロールバックして最初からやりなおす](#)

G.3.1 REDOデータが転送されていないためスイッチオーバーできない

スイッチオーバーが正常に完了しない場合は、V\$ARCHIVED_LOGビューのSEQUENCE#列の問合せを行い、元のプライマリ・データベースから転送された最新のREDOデータが、スタンバイ・データベースに適用されていることを確認してください。

最後のREDOデータがスタンバイ・データベースに転送されていない場合は、そのREDOデータが含まれるアーカイブREDOログ・ファイルを元のプライマリ・データベースから古いスタンバイ・データベースに手でコピーし、SQL ALTER DATABASE REGISTER LOGFILE file_specification文を使用して登録します。次に適用サービスを開始すると、アーカイブREDOログ・ファイルが自動的に適用されます。V\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせます。SWITCHOVER_STATUS列からTO PRIMARYまたはSESSIONS ACTIVEが戻されると、プライマリ・ロールへのスイッチオーバーが可能になります。

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

```
SWITCHOVER_STATUS
-----
TO PRIMARY
1 row selected
```

V\$DATABASEビューのSWITCHOVER_STATUS列に対するその他の有効な値の詳細は、[「Oracle Data Guardに関連するビュー」](#)を参照してください。

スイッチオーバーを続行するには、[「古い構文を使用したフィジカル・スタンバイ・データベースへのスイッチオーバーの実行」](#)の指示に従い、ターゲット・スタンバイ・データベースをプライマリ・ロールへのスイッチを再試行します。

G.3.2 ORA-01102エラーによりスイッチオーバーできない

ORA-01102エラーが表示された場合に考えられる原因の一部と解決策を次に示します。

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。

ノート:



フィジカル・スタンバイ・データベースが起動後に読み取り専用モードでオープンされていない場合、停止して再起動する必要はありません。

ただし、2番目のデータベースの起動は、ORA-01102「データベースを排他モードでマウントすることができません」

ん。」エラーで失敗します。

スタンバイ・データベース(元のプライマリ・データベース)で使用する初期化パラメータ・ファイルにDB_UNIQUE_NAMEパラメータを設定していないと、スイッチオーバー中にこの現象が発生することがあります。スタンバイ・データベースのDB_UNIQUE_NAMEパラメータが設定されていない場合、スタンバイ・データベースとプライマリ・データベースの両方が同じマウント・ロックを使用するため、2つ目のデータベースの起動時にORA-01102エラーが発生します。

処置: DB_UNIQUE_NAME=*unique_database_name*をスタンバイ・データベースが使用する初期化パラメータ・ファイルに追加して、スタンバイ・データベースとプライマリ・データベースを停止し、再起動します。

G.3.3 スイッチオーバー後にREDOデータが適用されない

スイッチオーバー後にアーカイブREDOログ・ファイルが新しいスタンバイ・データベースに適用されない場合は、スイッチオーバー後に一部の環境パラメータまたは初期化パラメータが正しく設定されていなかったことが原因と考えられます。

処置:

- 新しいプライマリ・サイトのtnsnames.oraファイルおよび新しいスタンバイ・サイトのlistener.oraファイルを調べます。スタンバイ・サイトにはリスナーのエントリ、プライマリ・サイトにはそれに対応するサービス名が必要です。
- リスナーがまだ起動されていない場合は、スタンバイ・サイトで起動します。
- プライマリ・サイトからスタンバイ・サイトにREDOデータを正しく転送するための、LOG_ARCHIVE_DEST_*n*初期化パラメータが設定されているかどうかを調べます。たとえば、プライマリ・サイトでV\$ARCHIVE_DEST固定ビューを次のように問い合わせます。

```
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・サイトでLOG_ARCHIVE_FORMAT初期化パラメータが正しく設定されていることを確認します。
- 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。

G.3.4 失敗したスイッチオーバーをロールバックして最初からやりなおす

フィジカル・スタンバイ・データベースでエラーが発生し、スイッチオーバーを続行できない場合は、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。

次のステップを実行します。

1. 新しいスタンバイ・データベース(古いプライマリ)を停止し、マウントします。
2. 新しいスタンバイ・データベースでREDO Applyを開始します。
3. 新しいスタンバイ・データベースがいつでもプライマリ・ロールに戻せることを確認します。スタンバイ・データベースでV\$DATABASEビューのSWITCHOVER_STATUS列を問い合わせます。次に例を示します。

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

```
SWITCHOVER_STATUS
-----
TO_PRIMARY
1 row selected
```

値TO PRIMARYまたはSESSIONS ACTIVEは、スタンバイ・データベースでプライマリ・ロールに切り替える準備が完了していることを示します。値TO PRIMARYまたはSESSIONS ACTIVEが戻されるまで、この列の問合せを続行します。

4. 次の文を発行して、新しいスタンバイ・データベースを変換してプライマリ・ロールに戻します。

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

この文が正常に実行されると、データベースはプライマリ・データベース・ロールで実行されるため、それ以降のステップを実行する必要はありません。

この文が正常に実行されなかった場合は、ステップ5に進んでください。

5. プライマリからフィジカル・スタンバイにロールを変更するスイッチオーバーを開始したときに、トレース・ファイルがログ・ディレクトリに書き込まれています。このトレース・ファイルには、元のプライマリ制御ファイルを再作成するために必要なSQL文が含まれています。トレース・ファイルの位置を特定し、SQL文を一時ファイルに抽出します。SQL*Plusから一時ファイルを実行します。これによって、新しいスタンバイ・データベースはプライマリ・ロールに戻されます。
6. 元のフィジカル・スタンバイ・データベースを停止します。
7. 新しいスタンバイ制御ファイルを作成します。このファイルは、プライマリ・データベースとフィジカル・スタンバイ・データベースの再同期化に必要です。フィジカル・スタンバイ制御ファイルを元のフィジカル・スタンバイ・システムにコピーします。フィジカル・スタンバイ制御ファイルの作成方法は、[「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。
8. 元のフィジカル・スタンバイ・インスタンスを再起動します。

この手順が正常終了し、アーカイブ・ギャップ管理が使用可能になると、FALプロセスが起動し、欠落したアーカイブ REDOログ・ファイルをフィジカル・スタンバイ・データベースに再アーカイブします。プライマリ・データベース上でログ・スイッチを強制実行し、プライマリ・データベースとフィジカル・スタンバイ・データベースの両方のアラート・ログを調べて、アーカイブ REDOログ・ファイルの順序番号が正しいことを確認します。

アーカイブ・ギャップ管理の詳細は[「手動によるギャップの解決」](#)を、トレース・ファイルの場所については[「アーカイブ・トレースの設定」](#)を参照してください。

9. スwitchオーバーを再度実行します。

この時点で、Oracle Data Guard構成は初期状態にロールバックされています。最初に失敗したスイッチオーバーの問題をすべて解決した後、スイッチオーバー操作を再度実行できます。

H ALTERNATE属性を使用したリモート代替宛先の構成

Oracle Database 12cリリース2 (12.2)以降、アクティブな宛先に障害があった場合に引き継ぐリモート・スタンバイ・データベースおよび遠隔同期インスタンスの代替ログ・アーカイブ先を作成するための優先的な方法は、GROUPおよびPRIORITY属性を使用することです。

ローカル・アーカイブの場所(LOCATION=…)にはALTERNATE属性が引き続き使用されます。これは、アーカイブの場所へのアクセスを妨げるディスクまたはネットワークの問題が原因で元のアーカイブ・ディレクトリが使用できなくなった場合に高可用性を提供するためです。しかし、リモートのログ・アーカイブ先(SERVICE=…)に対するALTERNATE属性の使用は、下位互換性のためにのみ維持されています。この方法を使用するために次の項に示されている例は、遠隔同期インスタンスを作成するための続きですが、REDO宛先のカスケードにも適用されます。

[遠隔同期インスタンスの作成および構成](#)のステップを実行すると、この時点で、遠隔同期インスタンスにより、WAN経由のリモート・サイトにあるターミナル・スタンバイに対する構成にデータ消失ゼロ機能が備わります。遠隔同期インスタンスとの通信が失われた場合に、低い保護レベルでも構成が保護されたままにするために、オプションでターミナル・スタンバイを自動的に代替宛先になるよう構成できます。これにより、Oracle Data Guardは遠隔同期インスタンスを一時的にバイパスして、プライマリからターミナル・スタンバイに直接REDOを非同期で送信できるため、データ消失量が削減されます。

関連項目:

- [代替アーカイブ先](#)

H.1 代替宛先の構成

代替宛先を構成するには、プライマリ・データベースでこれらのパラメータを設定します。

プライマリ・データベースchicago

```
LOG_ARCHIVE_DEST_STATE_2=' ENABLE'

LOG_ARCHIVE_DEST_2=' SERVICE=chicagoFS SYNC AFFIRM MAX_FAILURE=1 ALTERNATE=LOG_ARCHIVE_DEST_3
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS'

LOG_ARCHIVE_DEST_STATE_3=' ALTERNATE'

LOG_ARCHIVE_DEST_3=' SERVICE=boston ASYNC ALTERNATE=LOG_ARCHIVE_DEST_2
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=boston'
```

この構成により、Oracle Data Guardは遠隔同期インスタンスchicagoFSに直接REDOを送信できなくなったときに、ターミナル・スタンバイbostonにREDOを非同期で送信し続けることができます。遠隔同期インスタンスが再度使用できるようになると、Oracle Data Guardは遠隔同期インスタンスchicagoFSと自動的に再同期し、プライマリが遠隔同期インスタンスにREDOを送信し、遠隔同期インスタンスがターミナル・スタンバイにそのREDOを転送する元の構成に戻ります。同期が完了すると、代替宛先(前の例ではLOG_ARCHIVE_DEST_3)は再度代替として休止状態になります。

上の例では、ALTERNATEリモート宛先が非同期REDO転送を使用して2つのデータベース間に直接設定されている場合、アクティブな遠隔同期インスタンスで障害が発生すると、構成の保護レベルは元の最大パフォーマンスに低下してフェイルオーバー時にデータ消失が発生します。システム障害またはネットワーク障害からの保護を強固にするため、アクティブな遠隔同期インスタンスが高い可用性を保持するように追加の遠隔同期インスタンスを構成できます。

次の構成では、一方は遠隔同期インスタンスであり、他方は代替遠隔同期インスタンスです。代替遠隔同期インスタンスを構成することで、優先遠隔同期インスタンスがなんらかの理由で失敗した場合に、構成に対する保護が継続されて最大の可用性で維持されます。プライマリは遠隔同期インスタンスで障害が検知されると、自動的に代替遠隔同期インスタンスへの送信を開始します。その後優先遠隔同期インスタンスが再確立すると、プライマリは優先遠隔同期インスタンスに切り替わり、代替遠隔同期インスタンスは代替状態に戻ります。

このタイプの構成では、プライマリはいつでも2つの遠隔同期インスタンスの一方のみを使用してREDOを再配信します。

2つ目の高可用性遠隔同期インスタンスは、[遠隔同期インスタンスの作成および構成](#)に示された同じステップを使用して作成されると、ターミナル・スタンバイではなく既存の遠隔同期インスタンスに対する代替になります。完了したときのchicagoのパラメータは、次のように構成されています(名前chicagoFS1は、新しい遠隔同期インスタンスの名前とします)。

プライマリ・データベースchicago

```
LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, chicagoFS1, boston)'  
  
LOG_ARCHIVE_DEST_STATE_2=' ENABLE'  
  
LOG_ARCHIVE_DEST_2=' SERVICE=chicagoFS SYNC AFFIRM MAX_FAILURE=1 ALTERNATE=LOG_ARCHIVE_DEST_3  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS'  
  
LOG_ARCHIVE_DEST_STATE_3=' ALTERNATE'  
  
LOG_ARCHIVE_DEST_3=' SERVICE=chicagoFS1 SYNC AFFIRM ALTERNATE=LOG_ARCHIVE_DEST_2  
VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE) DB_UNIQUE_NAME=chicagoFS1'
```

プライマリ・データベースboston

```
LOG_ARCHIVE_CONFIG=' DG_CONFIG=(chicago, chicagoFS, chicagoFS1, boston)'
```

この時点で、Oracle Data Guardは、遠隔同期インスタンスにREDOを同期的に送信し続けることができ、なんらかの理由で遠隔同期インスタンスに障害が発生しても、必要なデータ消失ゼロの最大可用性保護モードを維持します。前述のように、障害が発生した遠隔同期インスタンスが再度使用できるようになると、Oracle Data Guardは自動的に再同期し、プライマリが1つ目の遠隔同期インスタンスにREDOを送信し、遠隔同期インスタンスがターミナル・スタンバイにそのREDOを転送する元の構成に戻ります。同期が完了すると、代替宛先(前の例ではLOG_ARCHIVE_DEST_3)は再度代替として休止状態になります。しかし、両方の遠隔同期インスタンスに障害が発生した場合、3つ目の代替機能がないためREDOはターミナル・スタンバイbostonに送信されません。そのシナリオは、ALTERNATEのかわりにGROUPおよびPRIORITY属性を使用して実現できます。

関連項目:

- [ALTERNATE](#)
- [GROUP](#)
- [PRIORITY](#)

索引

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#)

A

- アクティブ化
 - ロジカル・スタンバイ・データベース [9.3.2](#)
 - フィジカル・スタンバイ・データベース [12.8.3](#)
- Active Data Guard
 - およびフィジカル・スタンバイ・データベース [2.1.1](#), [10.2](#)
 - およびリアルタイム問合せ機能 [10.2.1](#)
- 追加
 - データファイル [10.3.1](#), [A.10.1.1](#)
 - ロジカル・スタンバイ・データベースでの索引 [11.5.4.1](#)
 - 新規または既存のスタンバイ・データベース [1.3](#)
 - オンラインREDOログ・ファイル [10.3.5](#)
 - 表領域 [10.3.1](#)
- 調整
 - 初期化パラメータ・ファイル
 - ロジカル・スタンバイ・データベース用 [4.2.4.2](#)
- AFFIRM属性 [17.1](#)
- ALTER DATABASE文
 - ACTIVATE STANDBY DATABASE句 [9.3.2](#), [12.8.3](#)
 - ADD STANDBY LOGFILE句 [18.1](#)
 - ADD STANDBY LOGFILE MEMBER句 [18.1](#), [A.1.1](#)
 - ADD SUPPLEMENTAL LOG DATA句 [18.1](#)
 - COMMIT TO SWITCHOVER句 [9.3.1](#), [18.1](#)
 - トラブルシューティング [G.3.2](#)
 - CREATE STANDBY CONTROLFILE句 [3.2.2](#)
 - REUSE句 [18.1](#)
 - DROP STANDBY LOGFILE MEMBER句 [18.1](#)
 - GUARD句 [11.2](#)
 - MOUNT STANDBY DATABASE句 [18.1](#)
 - OPEN READ ONLY句 [18.1](#)
 - PREPARE TO SWITCHOVER句 [9.3.1](#), [18.1](#)
 - RECOVER MANAGED STANDBY DATABASE句 [3.2.6](#), [4.2.5](#), [18.1](#)
 - 取消し [8.3.2](#)
 - フェイルオーバー [18.1](#)
 - REGISTER LOGFILE句 [A.4.1](#), [G.3.1](#)
 - RENAME FILE句 [A.1.1](#)
 - SET STANDBY DATABASE句
 - TO MAXIMIZE AVAILABILITY句 [18.1](#)
 - PERFORMANCE TO MAXIMIZE句 [9.1.4](#)

- PROTECTION TO MAXIMIZE句 [18.1](#)
- START LOGICAL STANDBY APPLY句 [8.4.1](#)、[13.5](#)、[A.6](#)
 - IMMEDIATEキーワード [8.4.1](#)
 - SQL Applyの開始 [4.2.5](#)
- STOP LOGICAL STANDBY APPLY句 [8.4.2](#)、[9.3.2](#)
- 代替アーカイブ先
 - 初期化パラメータの設定 [A.2](#)
- ALTERNATE属性 [17.2](#)
 - LOG_ARCHIVE_DEST_n初期化パラメータ [A.2](#)
- ALTER SESSION DISABLE GUARD文
 - データベース・ガードのオーバーライド [11.5.4](#)
- ALTER SESSION文
 - ENABLE GUARD句 [18.2](#)
- ALTER TABLESPACE文 [10.3.4](#)、[A.10.1.1](#)
- アプリケーション・コンティニューイティ
 - Oracle Data Guardでのサポート [1.5.1](#)
- 適用
 - REDOデータの即時 [8.2.1](#)
 - スタンバイ・データベースのREDOデータ [1.2.2](#)、[8](#)
 - SQL文をロジカル・スタンバイ・データベースに [8.4](#)
- 適用中状態 [11.4.1](#)
- 適用ラグ
 - リアルタイム問合せ環境での監視 [10.2.1.1](#)
- 適用ラグ許容差
 - リアルタイム問合せ環境での構成 [10.2.1.2](#)
- 適用サービス
 - 定義 [1.2.2](#)、[8.1](#)
 - REDOデータの適用遅延 [8.2.2](#)、[17.5](#)
 - リアルタイム適用
 - 定義 [8.2.1](#)
 - REDO Apply
 - 定義 [8.1](#)、[8.3](#)
 - 監視 [8.3.3](#)
 - 開始 [8.3.1](#)
 - 停止 [8.3.2](#)
 - SQL Apply
 - 定義 [1.2.2](#)、[8.1](#)
 - 監視 [8.4.3](#)
 - 開始 [8.4.1](#)
 - 停止 [8.4.2](#)
- アーカイブ先
 - 代替 [A.2](#)
- アーカイブREDOログ・ファイル
 - 情報へのアクセス [10.6.1.4](#)
 - 適用

- REDO Applyテクノロジー [1.2.2](#)
 - SQL Applyテクノロジー [1.2.2](#)
- 適用の遅延 [17.5](#)
 - スタンバイ・データベース [8.2.2](#)
- 不要なファイルの削除 [11.4.2](#)
- 宛先
 - 無効化 [7.2.2](#)
 - 有効化 [7.2.2](#)
- ギャップの管理 [1.8](#)
 - 「ギャップ管理」を参照 [1.8](#)
- 転送されたREDOデータ [1.2.2](#), [8.1](#)
- 登録
 - フェイルオーバー時 [9.3.2](#)
- スタンバイ・データベース [8.3.3](#), [8.4.3](#), [10.6.1](#)
- スイッチオーバーの問題のトラブルシューティング [A.4.1](#), [G.3.1](#)
- ARCHIVELOGモード
 - ソフトウェア要件 [2.3.2](#)
- アーカイブ
 - リアルタイム適用 [8.2.1](#)
 - 指定
 - 障害解決ポリシー [17.14](#)
 - 失敗した宛先 [17.14](#)
- ASM
 - 「自動ストレージ管理(ASM)」を参照 [15.5](#)
- ASYNC属性 [17.15](#)
- 属性
 - LOG_ARCHIVE_DEST_n初期化パラメータについて非推奨 [17](#)
- AUD\$表
 - ロジカル・スタンバイでのレプリケーション [C.14.2](#)
- 自動ブロック修復 [10.2.1.5](#)
- 欠落しているログ・ファイルの自動検出 [1.2.1](#), [1.8](#)
- 自動フェイルオーバー [1.2.3](#)
- 自動ストレージ管理(ASM)
 - 使用するスタンバイ・データベースの作成 [15.5](#)
- 自動スイッチオーバー [1.2.3](#)
 - 「スイッチオーバー」を参照 [9.1](#)

B

- BACKUP INCREMENTAL FROM SCNコマンド
 - 使用例 [12.8.1](#), [12.10](#)
- バックアップ操作
 - リカバリ不能な操作の後 [15.4.3](#)
 - フィジカル・スタンバイ・データベースの構成 [1.1.5](#)

- スタンバイ・データベースでのオフロード [1.8](#)
 - プライマリ・データベース [1.1.2](#)
 - ブローカによる使用 [1.3](#)
 - RMANの使用 [12](#)
 - 読取り可能な基本スタンバイ・データベース「スタンバイ・データベース環境のシミュレーション」を参照 [2.3.2](#)
 - バッチ処理
 - ロジカル・スタンバイ・データベースで [11.1.1.4](#)
 - メリット
 - Data Guard [1.8](#)
 - ロジカル・スタンバイ・データベース [2.1.2](#)
 - ローリング・アップグレード [13.1](#)
 - フィジカル・スタンバイ・データベース [2.1.1](#)
 - ブロック修復, 自動 [10.2.1.5](#)
 - ブローカ
 - コマンドライン・インタフェース [1.8](#)
 - 定義 [1.3](#)
 - グラフィカル・ユーザー・インタフェース [1.8](#)
-

C

- カスケードされたREDO転送先 [7.3](#)
- REDOのカスケード
 - 非リアルタイム [7.3](#)
 - リアルタイム [7.3](#)
- REDOデータのカスケード [7.3](#)
 - 構成要件 [7.3](#)
 - 制限事項 [7.3](#)
- CDB
 - ロジカル・スタンバイの作成 [4.4](#)
 - フィジカル・スタンバイの作成 [3.5](#)
- 文字セット
 - プライマリ・データベースでの変更 [15.9](#)
 - 異なるものを使用して構成 [C.18](#)
- チェックリスト
 - フィジカル・スタンバイ・データベース作成に関するタスク [3.2](#)
 - スタンバイ・データベース作成に関するタスク [4.2](#)
- チェックポイント
 - V\$LOGSTDBY_PROGRESSビュー [11.1.1.3](#)
- チャンク
 - トランザクション [11.1.1.1](#)
- コマンドライン・インタフェース
 - ブローカ [1.8](#)
- COMMIT TO SWITCHOVER句
 - ALTER DATABASE [9.3.1](#), [18.1](#)

- トラブルシューティング [G.3.2](#)
- COMPATIBLE初期化パラメータ
 - Oracle Databaseソフトウェアのアップグレード後の設定 [B.5](#)
 - ローリング・アップグレードの設定 [13.5](#)
- 補完テクノロジー [1.6](#)
- COMPRESSION属性 [17.3](#)
- 構成オプション
 - Data Guard Brokerでの作成 [1.3](#)
 - 概要 [1.1](#)
 - フィジカル・スタンバイ・データベース
 - 位置とディレクトリ構造 [2.4](#)
 - スタンバイ・データベース
 - 遅延スタンバイ [8.2.2](#)
- 構成
 - スタンバイ・データベースでのバックアップ [1.1.5](#)
 - 障害時リカバリ [1.1.5](#)
 - 初期化パラメータ
 - 代替アーカイブ先 [A.2](#)
 - フィジカル・スタンバイ・データベースのリスナー [5.1.1](#)
 - データ消失はない [1.2.3](#)
 - フィジカル・スタンバイ・データベース [2.4](#)
 - ロジカル・スタンバイ・データベースでのレポート生成操作 [1.1.5](#)
 - リモート位置のスタンバイ・データベース [1.1.5](#)
- 制約
 - ロジカル・スタンバイ・データベースでの処理 [11.6.3](#)
- 制御ファイル
 - コピー [3.2.4](#)
 - スタンバイ・データベース用に作成 [3.2.2](#)
- 変換
 - ロジカル・スタンバイ・データベースからフィジカル・スタンバイ・データベースへ
 - 終了 [4.2.4.1](#)
 - フィジカル・スタンバイ・データベースからロジカル・スタンバイ・データベースへ, [4.2.4.1](#)
- ALTER DATABASE文のCONVERT TO SNAPSHOT STANDBY句 [18.1](#)
- COORDINATORプロセス
 - LSPバックグラウンド・プロセス [11.1](#)
- コピー
 - 制御ファイル [3.2.4](#)
- CREATE STANDBY CONTROLFILE句
 - ALTER DATABASE [3.2.2](#), [18.1](#)
- CREATE TABLE AS SELECT(CTAS)文
 - ロジカル・スタンバイ・データベースで適用 [11.1.1.5](#)
- 作成
 - ロジカル・スタンバイ・データベースでの索引 [11.5.4.1](#)

D

- データ可用性
 - システム・パフォーマンス要件とのバランス [1.8](#)
- データベース・キャッシュ・モード
 - 全データベース・キャッシュの強制
 - Data Guard構成のステータス [10.2.1.7](#)
- データベース・ガード [11.5.4](#)
 - オーバーライド [11.5.4](#)
- データベース・インカネーション
 - OPEN RESETLOGS [10.4](#)
- データベースのロール
 - プライマリ [9.1](#)
 - スタンバイ [1.1.2](#), [9.1](#)
 - 推移 [1.2.3](#)
- データベース
 - フェイルオーバー [9.1.4](#)
 - ロールの推移 [9.1](#)
 - 障害とデータ破損からの保護 [1](#)
 - ソフトウェア・バージョンのアップグレード [13.1](#)
- データベース・スキーマ
 - フィジカル・スタンバイ・データベース [1.1.2](#)
- データファイル
 - プライマリ・データベースへの追加 [10.3.1](#)
 - 監視 [10.6](#)
 - プライマリ・データベースでの改名 [10.3.4](#)
- Data Guard Broker
 - 定義 [1.3](#)
 - 分散管理フレームワーク [9](#)
 - フェイルオーバー [1.3](#)
 - ファスト・スタート [9](#)
 - 手動 [1.3](#), [9](#)
 - ファスト・スタート・フェイルオーバー [1.3](#)
 - スイッチオーバー [9](#)
- Data Guard構成
 - ログ・ライター・プロセスを使用したスタンバイ宛先へのアーカイブ [8.2.1](#)
 - 定義 [1.1](#)
 - 保護モード [1.4](#)
 - Oracle Databaseソフトウェアのアップグレード [B](#)
- データ消失
 - フェイルオーバーによる [1.2.3](#)
 - スイッチオーバー [9.1](#)
- データ保護
 - パフォーマンスとのバランス [1.8](#)
 - 柔軟性 [1.8](#)

- Data Guardによる実現 [1](#)
- データ保護モード
 - REDO転送サービスによる実行 [1.2.1](#)
 - 概要 [1.4](#)
- データ・ポンプ・ユーティリティ
 - フィジカル・スタンバイ・データベースでのトランスポータブル表領域の使用 [10.3.3](#)
- DB_FILE_NAME_CONVERT初期化パラメータ
 - スイッチオーバー後スタンバイサイトで設定 [A.4.3](#), [G.3.3](#)
 - フィジカル・スタンバイ・データベースで設定 [3.2.3](#)
 - 位置とディレクトリ構造の計画時 [2.4](#)
- DB_ROLE_CHANGEシステムイベント [9.1.5](#)
- DB_UNIQUE_NAME属性 [17.4](#)
- DB_UNIQUE_NAME初期化パラメータ [A.4.2](#), [G.3.2](#)
 - LOG_ARCHIVE_CONFIGパラメータとともに必須 [16](#)
 - データベース初期化パラメータの設定 [3.1.4](#)
- DBA_LOGMNR_PURGED_LOGビュー
 - 削除できるアーカイブREDOログ・ファイルのリスト [11.4.2](#)
- DBA_LOGSTDBY_EVENTSビュー [11.3.1](#)、[19](#)、[A.6](#)
- DBA_LOGSTDBY_HISTORYビュー [19](#)
- DBA_LOGSTDBY_LOGビュー [11.3.2](#)、[19](#)
- DBA_LOGSTDBY_NOT_UNIQUEビュー [19](#)
- DBA_LOGSTDBY_PARAMETERSビュー [19](#)
- DBA_LOGSTDBY_SKIP_TRANSACTIONビュー [19](#)
- DBA_LOGSTDBY_SKIP [19](#)
- DBA_LOGSTDBY_UNSUPPORTED [19](#)
- DBA_TABLESPACES ビュー [10.6](#)
- DBMS_LOGSTDBY.BUILDプロシージャ
 - REDOデータでのディクショナリの構築 [4.2.3.2](#)
- DBMS_LOGSTDBYパッケージ
 - INSTATIATE_TABLEプロシージャ [11.5.5](#)
 - SKIP_ERRORプロシージャ [A.3](#)
 - SKIP_TRANSACTIONプロシージャ [A.6](#)
 - SKIPプロシージャ [A.6](#)
- DBMS_ROLLINGパッケージ
 - ロールリング・アップグレードに使用 [14](#)
- DDL文
 - SQL Applyでサポート [C](#)
- DDL文
 - DBLINKSを使用 [C.14.1](#)
- DDLトランザクション
 - ロジカル・スタンバイ・データベースで適用 [11.1.1.5](#)
 - ロジカル・スタンバイ・データベースに適用 [11.1.1.5](#)
- DELAY属性 [17.5](#)
 - LOG_ARCHIVE_DEST_n初期化パラメータ [8.2.2](#)
- 遅延

- アーカイブREDOログ・ファイルの適用 [17.5](#)
 - REDOログ・ファイルの適用 [8.2.2](#)
- 削除
 - アーカイブREDOログ・ファイル
 - DBA_LOGMNR_PURGED_LOGビューに表示 [11.4.2](#)
 - SQL Applyで不要 [11.4.2](#)
- 非推奨になった属性
 - LOG_ARCHIVE_DEST_n初期化パラメータ [17](#)
- 宛先
 - V\$ARCHIVE_DESTビューで表示 [19](#)
 - ロールベースの定義 [17.17](#)
- 検出
 - 欠落しているアーカイブREDOログ・ファイル [1.2.1](#), [1.8](#)
- DGMGRLコマンドライン・インタフェース
 - フェイルオーバーの起動 [1.3](#), [9](#)
 - スイッチオーバーの単純化 [1.3](#), [9](#)
- ディクショナリ
 - LogMinerの構築 [4.2.3.2](#)
- ディレクトリ位置
 - Optimal Flexible Architecture(OFA) [2.4](#)
 - スタンバイ・データベースの構造 [2.4](#)
- ダイレクト・パス・インサート
 - SQL Apply DMLの考慮事項 [11.1.1.4](#)
- 無効化
 - アーカイブREDOログ・ファイルの宛先 [7.2.2](#)
- 障害時リカバリ
 - 構成 [1.1.5](#)
 - Oracle Data Guardによる実現 [1](#)
 - スタンバイ・データベースによる実現 [1.1.5](#)
- ディスクI/O
 - AFFIRMおよびNOAFFIRM属性で制御 [17.1](#)
- 分散トランザクション [C.15](#)
- DML
 - ロジカル・スタンバイ・データベースでのバッチ更新 [11.1.1.4](#)
- DML操作
 - 一時表 [10.2.3.1](#)
- DMLトランザクション
 - ロジカル・スタンバイ・データベースに適用 [11.1.1.4](#)
- ダウングレード
 - Oracle Databaseソフトウェア [B.7](#)
- 削除
 - オンラインREDOログ・ファイル [10.3.5](#)
- DROP STANDBY LOGFILE MEMBER句
 - ALTER DATABASE [18.1](#)

E

- ENABLE GUARD句
 - ALTER SESSION [18.2](#)
 - 有効化
 - ロジカル・スタンバイ・データベースにおけるデータベース・ガード [18.2](#)
 - アーカイブREDOログ・ファイルの宛先 [7.2.2](#)
 - リアルタイム適用
 - ロジカル・スタンバイ・データベース [8.4.1](#)
-

F

- フェイルオーバー [1.2.3](#)
 - Data Guard Broker [1.3](#), [9](#)
 - 定義 [1.2.3](#)
 - DBA_LOGSTDBY_HISTORYに履歴を表示 [19](#)
 - ファスト・スタート・フェイルオーバー [9](#)
 - 後のデータベースのフラッシュバック [9.4](#)
 - ロジカル・スタンバイ・データベース [9.3.2](#)
 - 手動と自動 [1.2.3](#)
 - フィジカル・スタンバイ・データベース [18.1](#)
 - 準備 [9.1.4](#)
 - Data Guard Brokerでの単純化 [9](#)
 - REDOデータの事前転送 [9.1.4](#)
 - ロジカル・スタンバイ・データベースの特性の表示 [11.3.3](#)
 - 最大パフォーマンス・モード [9.1.4](#)
 - 最大保護モード [9.1.4](#)
- 障害解決ポリシー
 - REDO転送サービスに関する指定 [17.14](#)
- 遠隔同期インスタンス [5](#)
 - 作成 [5.1](#)
 - サポートされる保護モード [5.4](#)
- ファスト・スタート・フェイルオーバー
 - 自動フェイルオーバー [1.3](#), [9](#)
 - 監視 [10.6](#)
- FastSyncモード [6.1](#)
- FGA_LOG\$表
 - ロジカル・スタンバイでのレプリケーション [C.14.2](#)
- ファイル指定
 - ロジカル・スタンバイ・データベースでの改名 [11.5.3](#)
- フラッシュバック・データベース
 - ロールの推移後 [9.4](#)
 - OPEN RESETLOGS後 [15.3](#)
 - ロールの推移後 [9.4](#)
 - Data Guardを補完する特性 [1.6](#)

- [フィジカル・スタンバイ・データベース 15.2.1](#)
 - 全データベース・キャッシュ・モードの強制
 - [Data Guard構成のステータス 10.2.1.7](#)
-

G

- ギャップ管理
 - [自動検出と自動解消 1.2.1, 1.8](#)
 - [欠落しているログ・ファイルの検出 1.8](#)
 - [アーカイブREDOログ・ファイルの登録](#)
 - [フェイルオーバー時 9.3.2](#)
 - [グローバル・データ・サービス 1.6](#)
 - [グローバル一時表](#)
 - [DML操作 10.2.3.1](#)
 - [Active Data Guardでの使用 10.2.3.1](#)
-

H

- 高可用性
 - [Real Application Clusters](#)
 - [Data Guardを補完する特性 1.6](#)
 - [Oracle Data Guardによる実現 1](#)
 - [Oracle RACおよびData Guardによる実現 1.6](#)
-

I

- [アイドル状態 11.4.1](#)
 - [データベースのインカネーション](#)
 - [変更 10.4](#)
 - [初期化パラメータ](#)
 - [DB_UNIQUE_NAME 3.1.4, A.4.2, G.3.2](#)
 - [LOG_ARCHIVE_MIN_SUCCEED_DEST 17.9](#)
 - [LOG_FILE_NAME_CONVERT E.2.2.3.1](#)
 - [プライマリ・ロールおよびスタンバイ・ロールの設定 17.17](#)
 - [INITIALIZING状態 11.4.1](#)
 - [INSTANTIATE_TABLEプロシージャ](#)
 - [DBMS_LOGSTDBY 11.5.5](#)
-

K

- [KEEP IDENTITY句 4.2.4.1](#)
-

L

- 待機時間
 - ロジカル・スタンバイ・データベース [11.1.1.4](#), [11.1.1.5](#)
- listener.oraファイル
 - REDO転送サービス調整 [A.7](#)
 - トラブルシューティング [A.1.2](#), [A.7](#)
- ディクショナリ・ロード中状態 [11.4.1](#)
- LOCATION属性[17.8](#)
 - 設定
 - LOG_ARCHIVE_DEST_n初期化パラメータ [A.2](#)
- LOG_ARCHIVE_CONFIG初期化パラメータ [3.1.4](#), [3.2.3](#)
 - 例 [17.4](#)
 - 定義済の一意のデータベース名のリスト表示 [19](#)
 - DB_UNIQUE_NAMEパラメータとの関係 [16](#)
- LOG_ARCHIVE_DEST_n初期化パラメータ
 - AFFIRM属性 [17.1](#)
 - ALTERNATE属性 [17.2](#), [A.2](#)
 - ASYNC属性 [17.15](#)
 - COMPRESSION属性 [17.3](#)
 - DB_UNIQUE_NAME属性 [17.4](#)
 - DELAY属性 [8.2.2](#), [17.5](#)
 - 非推奨になった属性 [17](#)
 - LOCATION属性 [17.8](#), [A.2](#)
 - MANDATORY属性 [17.9](#)
 - MAX_FAILURE属性 [17.10](#)
 - NET_TIMEOUT属性 [17.11](#)
 - NOAFFIRM属性 [17.1](#)
 - NOALTERNATE属性 [A.2](#)
 - NODELAY属性 [8.2.2](#)
 - NOREGISTER属性 [17.12](#)
 - REOPEN属性 [17.14](#)
 - SERVICE属性 [17.8](#)
 - SYNC属性 [17.15](#)
 - VALID_FOR属性 [17.17](#)
- LOG_ARCHIVE_MIN_SUCCEED_DEST初期化パラメータ [17.9](#)
- LOG_FILE_NAME_CONVERT初期化パラメータ
 - スイッチオーバー後スタンバイサイトで設定 [A.4.3](#), [G.3.3](#)
 - フィジカル・スタンバイ・データベースで設定 [3.2.3](#)
 - 位置とディレクトリ構造の計画時 [2.4](#)
- ログ適用サービス
 - REDO Apply
 - 監視 [10.6.1](#)
 - 開始 [10.1.1](#)
 - 停止 [10.1.2](#)

- REDO Applyに関するチューニング [10.8](#)
- 論理変更レコード(LCR)
 - 全体が使用済のキャッシュ・メモリー [11.1.1.2](#)
 - ステージング [11.1](#)
- ロジカル・スタンバイ・データベース [1.1.2](#)
 - 追加
 - データファイル [A.10.1.1](#)
 - 索引 [11.5.4.1](#)
 - 表 [11.5.5](#)
 - バックグラウンド・プロセス [11.1](#)
 - メリット [2.1.2](#)
 - 表に対するユーザー・アクセスの制御 [11.2](#)
 - 作成 [4](#)
 - フィジカル・スタンバイ・データベースからの変換 [4.2.4.1](#)
 - Data Guard Broker [1.3](#)
 - データベース・ガード
 - オーバーライド [11.5.4](#)
 - データ型
 - サポート対象 [C](#), [C.1.1](#)
 - サポート対象外 [C.1.2](#)
 - SQL文の実行 [1.1.2](#)
 - フェイルオーバー [9.3.2](#)
 - 履歴の表示 [19](#)
 - 障害の処理 [A.3](#)
 - V\$LOGSTDBY_STATSに特性を表示 [11.3.3](#)
 - ロジカル・スタンバイ・プロセス(LSP) [11.1](#)
 - マテリアライズド・ビュー
 - サポート [C.13](#)
 - 監視 [8.4.3](#), [19](#)
 - ファイル指定の改名, [11.5.3](#)
 - スキップ・ハンドラの設定 [11.5.3](#)
 - SQL Apply [1.2.2](#)
 - REDOのプライマリ・データベース・ブランチとの再同期化 [11.6.5](#)
 - DDL文のスキップ [C.13](#)
 - SQL文のスキップ [C.13](#)
 - リアルタイム適用の開始 [8.4.1](#)
 - 停止 [8.4.2](#)
 - テクノロジ [8.1](#)
 - トランザクション・サイズの考慮事項 [11.1.1.1](#)
 - 開始
 - リアルタイム適用 [8.4.1](#)
 - 状態
 - 適用 [11.4.1](#)
 - アイドル [11.4.1](#)
 - 初期化 [11.4.1](#)

- ディクショナリ・ロード中 [11.4.1](#)
 - 透過的データ暗号化を指定したプライマリ・データベースのサポート [C.3](#)
 - スイッチオーバー [9.3.1](#)
 - スループットと待機時間 [11.1.1.4](#), [11.1.1.5](#)
 - アップグレード [B.4](#)
 - ローリング・アップグレード [2.3.2](#)
 - ロジカル・スタンバイ・プロセス(LSP)
 - COORDINATORプロセス [11.1](#)
 - LogMinerディクショナリ
 - DBMS_LOGSTDBY.BUILDプロシージャを使用した構築 [4.2.3.2](#)
 - ロジカル・スタンバイ・データベースの作成時 [4.2.4.1](#)
 - ログ・ライター・プロセス(LGWR)
 - ASYNCネットワーク転送 [17.15](#)
 - NET_TIMEOUT属性 [17.11](#)
 - SYNCネットワーク転送 [17.15](#)
-

M

- 管理リカバリ操作
 - 「Redo Apply」を参照
- MANDATORY属性 [17.9](#)
- MAX_FAILURE属性 [17.10](#)
- 最大可用性モード
 - 概要 [1.4](#)
- 最大パフォーマンス・モード [9.1.4](#)
 - 概要 [1.4](#)
- 最大パフォーマンス保護モード [6.1](#)
- 最大保護モード
 - for Oracle Real Application Clusters [D.2.2](#)
 - 概要 [1.4](#)
 - スタンバイ・データベース [9.1.4](#)
- メモリー
 - 全体が使用済のLCRキャッシュ [11.1.1.2](#)
- 欠落しているログ順序番号
 - 検出 [1.8](#)
 - 「ギャップ管理」を参照 [1.8](#)
- 変更
 - ロジカル・スタンバイ・データベース [11.5.4](#)
- 監視
 - プライマリ・データベース・イベント [10.6](#)
 - 表領域の状態 [10.6](#)
- MOUNT STANDBY DATABASE句
 - ALTER DATABASE [18.1](#)
- マルチテナント・コンテナ・データベース

- 「CDB」を参照 [3.5](#), [4.4](#)
-

N

- NET_TIMEOUT属性 [17.11](#)
 - ネットワークI/O操作
 - ネットワーク・タイマー
 - NET_TIMEOUT属性 [17.11](#)
 - チューニング
 - REDO転送サービス [A.7](#)
 - ネットワーク・タイムアウト
 - 応答 [17.11](#)
 - NOAFFIRM属性 [17.1](#)
 - NOALTERNATE属性
 - LOG_ARCHIVE_DEST_n初期化パラメータ [A.2](#)
 - データ消失はない
 - データ保護モードの概要 [1.4](#)
 - 確認 [1.2.3](#)
 - 保証 [1.2.3](#)
 - 最大可用性モードによる実現 [1.4](#)
 - 最大保護モードによる実現 [1.4](#)
 - NODELAY属性
 - LOG_ARCHIVE_DEST_n初期化パラメータ [8.2.2](#)
 - NOREGISTER属性 [17.12](#)
-

O

- OMF
 - 「Oracle Managed Files(OMF)」を参照 [15.5](#)
- ディスク上のデータベース構造
 - フィジカル・スタンバイ・データベース [1.1.2](#)
- オンライン・データ・ファイル
 - 場所の移動 [2.5](#)
- オンラインREDOログ・ファイル
 - 追加 [10.3.5](#)
 - 削除 [10.3.5](#)
- OPEN READ ONLY句
 - ALTER DATABASE [18.1](#)
- OPEN RESETLOGS
 - 後のフラッシュバック [15.3](#)
- OPEN RESETLOGS句
 - データベース・インカーネーションの変更 [10.4](#)
 - リカバリ [10.4](#)
- 動作要件 [2.3](#), [2.3.2](#)

- Optimal Flexible Architecture(OFA)
 - ディレクトリ構造 [2.4](#)
 - ORA-01102メッセージ
 - スイッチオーバー障害の原因となる [A.4.2](#), [G.3.2](#)
 - Oracle Databaseソフトウェア
 - SQL Applyを使用したアップグレードの要件 [13.2](#)
 - アップグレード [2.3.2](#), [B.1](#)
 - SQL Applyを使用したアップグレード [13.1](#)
 - Oracle Enterprise Manager
 - フェイルオーバーの起動 [1.3](#), [9](#)
 - スイッチオーバーの起動 [1.3](#), [9](#)
 - Oracle Managed Files(OMF)
 - 使用するスタンバイ・データベースの作成 [15.5](#)
 - Oracle RAC One Node
 - Oracle Data Guardによりサポート [1.6](#)
 - Real Application Clusters
 - プライマリ・データベース [1.1.1](#), [D.1.2](#)
 - 設定
 - 最大データ保護 [D.2.2](#)
 - スタンバイ・データベース [1.1.2](#), [D.1](#)
 - Oracle Recovery Managerユーティリティ(RMAN)
 - フィジカル・スタンバイ・データベースのバックアップ・ファイルの作成 [12](#)
 - Oracle Sharding
 - Oracle Active Data Guardによりサポート [1.7](#)
 - Oracle Standard Edition
 - スタンバイ・データベース環境の再現 [2.3.2](#)
-

P

- ページアウトの考慮事項 [11.1.1.2](#)
- ページアウト
 - SQL Apply [11.1.1.2](#)
- パラレルDML(PDML)トランザクション
 - SQL Apply [11.1.1.3](#), [11.1.1.4](#)
- パッチ・セット・リリース
 - アップグレード [2.3.2](#)
- パフォーマンス
 - データ可用性とのバランス [1.8](#)
 - データ保護とのバランス [1.8](#)
- フィジカル・スタンバイ・データベース
 - およびOracle Active Data Guard [2.1.1](#)
 - REDOデータの適用 [8.1](#), [8.3](#)
 - REDO Applyテクノロジー [8.3](#)
 - REDOログ・ファイルの適用

- 開始 [8.3.1](#)
- メリット [2.1.1](#)
- 構成オプション [2.4](#)
- データファイルのパス名の変換 [3.2.3](#)
- ログ・ファイルのパス名の変換 [3.2.3](#)
- ロジカル・スタンバイ・データベースへの変換 [4.2.4.1](#)
- 作成
 - タスクのチェックリスト [3.2](#)
 - リスナーの構成 [5.1.1](#)
 - ディレクトリ構造 [2.4](#)
 - Data Guard Broker [1.3](#)
- 定義 [1.1.2](#)
- フェイルオーバー
 - 更新をチェック [9.1.4](#)
- フェイルオーバー後のフラッシュバック [15.2.1](#)
- 監視 [8.3.3](#), [10.6.1](#), [19](#)
- 読取り専用または読取り/書込みアクセス用にオープン [10.2](#)
- 読取り専用 [10.2](#)
- OPEN RESETLOGS操作によるリカバリ [10.4](#)
- REDO Apply [1.2.2](#)
- REDOのプライマリ・データベース・ブランチとの再同期化 [10.4](#)
- ロールの推移 [9.2](#), [G](#)
- BACKUP INCREMENTAL FROM SCNコマンドによるロール・フォワード [12.8.1](#), [12.10](#)
- 停止 [10.1.2](#)
- 開始
 - 適用サービス [8.3.1](#)
- ログ適用レートのチューニング [10.8](#)
- アップグレード [B.3](#)
- トランスポータブル表領域の使用 [10.3.3](#)
- PL/SQLパッケージ
 - サポート対象 [C.11.1](#)
 - サポート対象外 [C.11.2](#)
- PREPARERプロセス
 - SGA内でのLCRのステージング [11.1](#)
- PREPARE TO SWITCHOVER句
 - ALTER DATABASE [9.3.1](#), [18.1](#)
- プライマリ・データベース
 - 構成
 - Oracle Real Application Cluster [1.1.1](#)
 - 単一インスタンス [1.1.1](#)
 - データ・ファイル
 - 追加 [10.3.1](#)
 - 定義 [1.1.1](#)
 - ギャップの解決, [1.8](#)
 - イベントの監視 [10.6](#)

- ネットワーク接続
 - ネットワーク停止の回避 [17.11](#)
 - ネットワーク・タイムアウトの処理 [17.11](#)
- Oracle Real Application Clusters
 - 設定 [D.1.2](#)
- 準備
 - フィジカル・スタンバイ・データベースの作成 [3.1](#)
- 要件
 - ロジカル・スタンバイ・データベースの作成 [4.1](#)
- REDO転送サービス [1.2.1](#)
- ワークロードの低減 [1.8](#)
- リストア・ポイント
 - スタンバイへのレプリケート [10.7](#)
- スイッチオーバー [9.1.3](#)
- 表領域
 - 追加 [10.3.1](#)
- プライマリ・データベース
 - ARCHIVELOGモード [2.3.2](#)
 - ソフトウェア要件 [2.3.2](#)
- 主キー列
 - サプリメンタル・ロギングを使用したログ [4.2.3.2](#), [11.1.1.4](#)
- プライベート一時表
 - Active Data Guardでの使用 [10.2.3.2](#)
- プロセス
 - SQL Applyアーキテクチャ [11.1](#), [11.4.1](#)
- 保護モード
 - 最大可用性モード [1.4](#)
 - 最大パフォーマンス [6.1](#)
 - 最大パフォーマンス・モード [1.4](#)
 - 最大保護モード [1.4](#)
 - 監視 [10.6](#)
 - プライマリ・データベースでの設定 [6.2](#)

Q

- 問合せ
 - スタンバイ・データベースでのオフロード [1.8](#)

R

- 読取り専用操作 [1.2.2](#)
 - フィジカル・スタンバイ・データベース [10.2](#)
- リアルタイム適用
 - 定義 [8.2.1](#)

- 開始
 - ロジカル・スタンバイ [8.4.1](#)
- ロジカル・スタンバイ・データベースでの開始 [8.4.1](#)
- 停止
 - ロジカル・スタンバイ [8.4.2](#)
 - フィジカル・スタンバイ・データベース [10.1.2](#)
- リアルタイム・カスケード [7.3](#)
- リアルタイム問合せ機能 [10.2](#)
 - およびOracle Active Data Guard [10.2](#), [10.2.1](#)
 - 適用ラグ許容差の構成 [10.2.1.2](#)
 - REDO Applyの強制同期 [10.2.1.3](#)
 - 適用ラグの監視 [10.2.1.1](#)
 - 制限事項 [10.2.1.4](#)
 - 使用 [10.2.1](#)
- リカバリ
 - エラー [A.10.1](#)
 - ロジカル・スタンバイ・データベース [11.6.5](#)
 - フィジカル・スタンバイ・データベース
 - OPEN RESETLOGS後 [10.4](#)
 - リセットログの使用 [10.4](#), [11.6.5](#)
- RECOVER MANAGED STANDBY DATABASE CANCEL句
 - 終了 [4.2.4.1](#)
- RECOVER MANAGED STANDBY DATABASE句
 - ALTER DATABASE [3.2.6](#), [4.2.5](#), [8.3.1](#), [18.1](#)
- RECOVER TO LOGICAL STANDBY句
 - フィジカル・スタンバイ・データベースからロジカル・スタンバイ・データベースへの変換 [4.2.4.1](#)
- Recovery Manager
 - Data Guardを補完する特性 [1.6](#)
 - スタンバイ・データベース
 - 作成 [E.2.1](#)
 - LOG_FILE_NAME_CONVERT初期化パラメータ [E.2.2.3.1](#)
 - RMANの準備 [E.2.2](#)
- 再作成
 - ロジカル・スタンバイ・データベース上の表 [11.5.5](#)
- DML操作のリダイレクト
 - スタンバイ・データベース [10.2.2.1](#)
- PL/SQLのリダイレクト
 - スタンバイ・データベース [10.2.2.2](#)
- REDO Apply
 - 定義 [1.2.2](#), [8.1](#)
 - フェイルオーバー後のフラッシュバック [15.2.1](#)
 - 開始 [3.2.6](#)
 - 停止 [10.1.2](#)
 - テクノロジ [1.2.2](#)
 - ログ適用レートのチューニング [10.8](#)

- REDOデータ
 - 適用
 - REDO Applyテクノロジーの使用 [1.2.2](#)
 - SQL Applyテクノロジーの使用 [1.2.2](#)
 - スタンバイ・データベースへ [8.1](#)
 - スタンバイ・データベースへ [1.1.2](#)
 - フィジカル・スタンバイ・データベースからロジカル・スタンバイ・データベースへの変換中の適用 [4.2.4.1](#)
 - スタンバイ・システムでのアーカイブ [1.2.2](#), [8.1](#)
 - ディクショナリの構築 [4.2.3.2](#)
 - カスケード [7.3](#)
 - 転送 [1.1.2](#), [1.2.1](#)
- REDOギャップ [7.6.3](#)
 - 手動による解決 [7.6.3.1](#)
 - 解決時間の短縮 [7.6.3](#)
- REDOログ・ファイル
 - 適用の遅延 [8.2.2](#)
- REDOログ
 - フィジカル・スタンバイ・データベースでの自動適用 [8.3.1](#)
 - スタンバイ・データベース表の更新 [1.8](#)
- REDO転送サービス [7](#)
 - アーカイブ先
 - 代替 [A.2](#)
 - 失敗した宛先への再アーカイブ [17.14](#)
 - セッションの認証
 - SSLを使用 [7.2.1.1](#)
 - 構成 [7.2](#)
 - セキュリティの構成 [7.2.1](#)
 - 定義 [1.2.1](#)
 - ギャップの検出 [7.6.3](#)
 - アーカイブ障害の処理 [17.14](#)
 - ステータスの監視 [7.6.1](#)
 - ネットワーク
 - 調整 [A.7](#)
 - 保護モード
 - 最大可用性モード [1.4](#)
 - 最大パフォーマンス・モード [1.4](#)
 - 最大保護モード [1.4](#)
 - REDOデータの受信 [7.2.3](#)
 - REDOデータの送信 [7.2.2](#)
 - 同期および非同期ディスクI/O [17.1](#)
 - 待機イベント [7.6.4](#)
- 登録
 - アーカイブREDOログ・ファイル
 - フェイルオーバー時 [9.3.2](#)
- REGISTER LOGFILE句

- ALTER DATABASE [A.4.1](#), [G.3.1](#)
- REGISTER LOGICAL LOGFILE句
 - ALTER DATABASE [9.3.2](#)
- RELY制約
 - 作成 [4.1.2](#)
- リモート・ファイル・サーバー・プロセス(RFS)
 - ログ・ライター・プロセス [8.2.1](#)
- RENAME FILE句
 - ALTER DATABASE [A.1.1](#)
- 改名
 - データ・ファイル
 - プライマリ・データベース上 [10.3.4](#)
 - STANDBY_FILE_MANAGEMENTパラメータの設定 [10.3.4](#)
- REOPEN属性 [17.14](#)
- レポート生成操作
 - 構成 [1.1.5](#)
 - スタンバイ・データベースでのオフロード [1.8](#)
- 要件
 - ローリング・アップグレード [13.2](#)
- 再開の考慮事項
 - SQL Apply [11.1.1.3](#)
- リストア・ポイント
 - スタンバイへのレプリケート [10.7](#)
- 再同期化
 - ロジカル・スタンバイ・データベースとREDOの新規ブランチ [11.6.5](#)
 - フィジカル・スタンバイ・データベースとREDOの新規ブランチ [10.4](#)
- 取得
 - 欠落しているアーカイブREDOログ・ファイル [1.2.1](#), [1.8](#)
- RMAN BACKUP INCREMENTAL FROM SCNコマンド [12.8.1](#), [12.10](#)
- RMANバックアップ
 - Data Guard環境におけるアクセス可能性 [12.1.3](#)
 - Data Guard環境における関連付け [12.1.2](#)
 - Data Guard環境における互換性 [12.1.1](#)
- ロールベースの宛先
 - 設定 [17.17](#)
- ロール管理サービス
 - 定義 [9](#)
- ロールの推移 [1.2.3](#), [9.1](#)
 - タイプの選択 [9.1.1](#)
 - 定義 [1.2.3](#)
 - 後のデータベースのフラッシュバック [9.4](#)
 - ロジカル・スタンバイ・データベース [9.3](#)
 - 監視 [10.6](#)
 - フィジカル・スタンバイ・データベース [9.2](#), [G](#)
 - 可逆的な推移 [1.2.3](#), [9.1](#)

- ロールの推移のトリガー [9.1.5](#)
 - DB_ROLE_CHANGEシステムイベント [9.1.5](#)
 - ロールバック
 - スイッチオーバー障害後 [A.4.4](#), [G.3.4](#)
 - ロールリング・アップグレード
 - ソフトウェア要件 [2.3.2](#)
 - ロールリング・アップグレード
 - メリット [13.1](#)
 - DBMS_ROLLINGパッケージ [14](#)
 - パッチ・セット・リリース [2.3.2](#)
 - 要件 [13.2](#)
 - COMPATIBLE初期化パラメータの設定 [13.5](#)
 - KEEP IDENTITY句の使用 [4.2.4.1](#)
 - Active Data Guardの使用 [14](#)
-

S

- スキーマ
 - プライマリ・データベースと同一 [1.1.2](#)
- 順序
 - ロジカル・スタンバイ・データベースでサポートされない [C.12](#)
 - Oracle Active Data Guardでの使用 [10.2.6](#)
- SERVICE属性 [17.8](#)
- SET STANDBY DATABASE句
 - ALTER DATA [18.1](#)
 - ALTER DATABASE [9.1.4](#), [18.1](#)
- 停止
 - フィジカル・スタンバイ・データベース [10.1.2](#)
- 再現
 - スタンバイ・データベース環境 [2.3.2](#)
- SKIP_ERRORプロシージャ
 - DBMS_LOGSTDBYパッケージ [A.3](#)
- SKIP_TRANSACTIONプロシージャ
 - of DBMS_LOGSTDBY [A.6](#)
- スキップ・ハンドラ
 - ロジカル・スタンバイ・データベースでの設定 [11.5.3](#)
- SKIPプロシージャ
 - of DBMS_LOGSTDBY [A.6](#)
- スナップショット・スタンバイ・データベース [1.1.2](#)
 - 管理 [10.11](#)
- ソフトウェア要件 [2.3.2](#)
 - ロールリング・アップグレード [2.3.2](#)
- SQL Apply [8.4.2](#), [11.1.1.2](#)
 - OPEN RESETLOGS後 [11.6.5](#)

- CREATE TABLE AS SELECT(CTAS)文の適用 [11.1.1.5](#)
- DDLトランザクションの適用 [11.1.1.5](#)
- DMLトランザクションの適用 [11.1.1.4](#)
- アーキテクチャ [11.1](#), [11.4.1](#)
- 定義 [1.2.2](#), [8.1](#)
- アーカイブREDOログ・ファイルの削除 [11.4.2](#)
- パラレルDML(PDML)トランザクション [11.1.1.3](#), [11.1.1.4](#)
- ロールバック・アップグレードの実行 [13.1](#)
- ロールバック・アップグレードの要件 [13.2](#)
- 再開の考慮事項 [11.1.1.3](#)
- ロールバック・アップグレード [2.3.2](#)
- 開始
 - リアルタイム適用 [8.4.1](#)
- 停止
 - リアルタイム適用 [8.4.2](#)
- サポートされるデータ型 [C.1.1](#)
- DDL文のサポート [C](#)
- PL/SQLパッケージのサポート [C.11.1](#)
- トランザクション・サイズの考慮事項 [11.1.1.1](#)
- サポートされていないデータ型 [C.1.2](#)
- unsupported PL/SQL supplied packages [C.11.2](#)
- 停止した場合の処置 [A.6](#)
- SQL文
 - ロジカル・スタンバイ・データベースでの実行 [1.1.2](#), [1.2.2](#)
 - ロジカル・スタンバイ・データベースでのスキップ [C.13](#)
- STANDBY_FILE_MANAGEMENT初期化パラメータ
 - データファイルの改名時 [10.3.4](#)
- スタンバイ・データベース
 - ロジカル作成 [4](#)
- スタンバイ・データベース
 - RMANを使用した作成 [E.2.1](#)
 - REDOデータの適用 [8](#)
 - REDOログ・ファイルの適用 [1.2.2](#), [1.8](#)
 - 適用サービス [8.1](#)
 - 構成 [1.1](#)
 - 最大数 [2](#)
 - Oracle Real Application Clusters [1.1.2](#), [D.1](#)
 - リモート位置 [1.1.5](#)
 - 単一インスタンス [1.1.2](#)
 - 作成 [1.1.2](#)
 - タスクのチェックリスト [4.2](#)
 - ディレクトリ構造の考慮事項 [2.4](#)
 - プライマリがASMまたはOMFを使用している場合 [15.5](#)
 - リモート・ホスト上で同一のディレクトリ構造 [E.3](#)
 - タイム・ラグのある [8.2.2](#)

- 定義 [2.1](#)
- フェイルオーバー
 - 準備 [9.1.4](#)
- フェイルオーバー [9.1.4](#)
- LOG_FILE_NAME_CONVERT初期化パラメータ [E.2.2.3.1](#)
- 動作要件 [2.3](#), [2.3.2](#)
- RMANの準備 [E.2.2](#)
- OPEN RESETLOGS操作によるリカバリ [10.4](#)
- DML操作のリダイレクト [10.2.2.1](#)
- PL/SQLのリダイレクト [10.2.2.2](#)
- プライマリ・データベースとの再同期化 [1.8](#)
- ソフトウェア要件 [2.3.2](#)
- フィジカル・スタンバイ・データベースでの適用サービスの開始 [8.3.1](#)
- スタンバイREDOログ・ファイル
 - リアルタイム適用 [8.2.1](#)
- スタンバイREDOログ
 - 作成および管理 [7.2.3.1](#)
- スタンバイ・ロール [1.1.2](#)
- 開始
 - ロジカル・スタンバイ・データベース [4.2.5](#)
 - フィジカル・スタンバイ・データベース [3.2.6](#)
 - リアルタイム適用 [8.4.1](#)
 - ロジカル・スタンバイ・データベース [8.4.1](#)
 - REDO Apply [10.1.1](#)
 - SQL Apply [4.2.5](#), [8.4.1](#)
- START LOGICAL STANDBY APPLY句
 - IMMEDIATEキーワード [8.4.1](#)
 - ALTER DATABASE [4.2.5](#), [8.4.1](#), [13.5](#), [A.6](#)
- STOP LOGICAL STANDBY APPLY句
 - ALTER DATABASE [8.4.2](#), [9.3.2](#)
- 停止
 - リアルタイム適用
 - ロジカル・スタンバイ・データベース [8.4.2](#)
 - フィジカル・スタンバイ・データベースでのリアルタイム適用 [8.3.2](#)
 - REDO Apply [8.3.2](#)
 - SQL Apply [8.4.2](#)
- Streams取得
 - ロジカル・スタンバイでの実行 [11.6.6](#)
- サプリメンタル・ロギング
 - 主キー列と一意索引列を記録するための設定 [4.2.3.2](#), [11.1.1.4](#)
- サポートされるデータ型
 - ロジカル・スタンバイ・データベース [C](#), [C.14](#)
- サポートされるPL/SQLパッケージ [C.11.1](#)
- SWITCHOVER_STATUS列
 - V\$DATABASEビュー [A.4.1](#), [G.3.1](#)

- スイッチオーバー [1.2.3](#)
 - ターゲット・スタンバイ・データベースの選択 [9.1.2](#)
 - 定義 [1.2.3](#)
 - DBA_LOGSTDBY_HISTORYに履歴を表示 [19](#)
 - ORA-01102で失敗 [A.4.2](#), [G.3.2](#)
 - 後のデータベースのフラッシュバック [9.4](#)
 - ロジカル・スタンバイ・データベース [9.3.1](#)
 - 手動と自動 [1.2.3](#)
 - 監視 [10.6](#)
 - データ消失はない [9.1](#)
 - 準備 [9.1.3](#)
 - 最後のアーカイブREDOログ・ファイルが転送されたかどうかの確認 [A.4.1](#), [G.3.1](#)
 - 後のDB_FILE_NAME_CONVERTの設定 [A.4.3](#), [G.3.3](#)
 - 後のLOG_FILE_NAME_CONVERTの設定 [A.4.3](#), [G.3.3](#)
 - Data Guard Brokerでの単純化 [1.3](#), [9](#)
 - 最初からやりなおし [A.4.4](#), [G.3.4](#)
 - 代表的な使用例 [9.1.3](#)
 - SYNC属性 [17.15](#)
 - システム・イベント
 - ロールの推移 [9.1.5](#)
 - システム・グローバル領域(SGA)
 - 論理変更レコードのステージング [11.1](#)
 - システム・リソース
 - 効率的な使用 [1.8](#)
-

T

- 表
 - ロジカル・スタンバイ・データベース
 - 追加 [11.5.5](#)
 - 表の再作成 [11.5.5](#)
 - サポート対象外 [C.12](#)
- 表領域
 - 追加
 - 新規データファイル [A.10.1.1](#)
 - プライマリ・データベースへの [10.3.1](#)
 - 状態の変更の監視 [10.6](#)
 - データベース間の移動 [10.3.3](#)
- ターゲット・スタンバイ・データベース
 - スイッチオーバー [9.1.2](#)
- ターミナル宛先 [7.3](#)
 - 構成 [7.3.1](#)
- 終了
 - ネットワーク接続 [17.11](#)

- スループット
 - ロジカル・スタンバイ・データベース [11.1.1.4](#), [11.1.1.5](#)
- TIME_COMPUTED列 [9.1.2](#)
- V\$DATAGUARD_STATSビューのTIME_COMPUTED列 [9.1.2](#)
- タイム・ラグ
 - アーカイブREDOログ・ファイルの適用の遅延 [8.2.2](#), [17.5](#)
 - スタンバイ・データベース [8.2.2](#)
- tnsnames.oraファイル
 - REDO転送サービス調整 [A.7](#)
 - トラブルシューティング [A.1.2](#), [A.4.3](#), [A.7](#), [G3.3](#)
- トランザクション・サイズの考慮事項
 - SQL Apply [11.1.1.1](#)
- 透過的データ暗号化
 - SQL Applyでサポート [C.3](#)
- トランスポータブル表領域
 - フィジカル・スタンバイ・データベースでの使用 [10.3.3](#)
- トリガー
 - ロジカル・スタンバイ・データベースでの処理 [11.6.3](#)
 - ロールの推移 [9.1.5](#)
- トラブルシューティング
 - SQL Applyが停止した場合 [A.6](#)
 - 最後のREDOデータが転送されていない [A.4.1](#), [G.3.1](#)
 - listener.oraファイル [A.1.2](#), [A.7](#)
 - ロジカル・スタンバイ・データベース障害 [A.3](#)
 - SQL Apply [A.6](#)
 - スイッチオーバー [A.4](#)
 - ORA-01102メッセージ [A.4.2](#), [G.3.2](#)
 - ロールバックおよび最初からやりなおし [A.4.4](#), [G.3.4](#)
 - tnsnames.oraファイル [A.1.2](#), [A.4.3](#), [A.7](#), [G3.3](#)
- チューニング
 - REDO Applyのログ適用レート [10.8](#)

U

- 一意索引列
 - サプリメンタル・ロギングを使用したログ [4.2.3.2](#), [11.1.1.4](#)
 - リカバリ不能操作
 - 直後のバックアップ [15.4.3](#)
 - unsupported PL/SQL supplied packages [C.11.2](#)
 - アップグレード
 - Oracle Databaseソフトウェア [2.3.2](#), [13.1](#), [B](#), [B.1](#)
 - COMPATIBLE初期化パラメータの設定 [B.5](#)
-

- V\$ARCHIVE_DEST_STATUSビュー [19](#)
- V\$ARCHIVE_DESTビュー [19](#), [A.1.2](#)
 - 全ての宛先の情報の表示 [19](#)
- V\$ARCHIVE_GAPビュー [19](#)
- V\$ARCHIVED_LOGビュー [10.6.1.4](#), [19](#)
- V\$DATABASE_INCARNATIONビュー [19](#)
- V\$DATABASEビュー [19](#)
 - ファスト・スタート・フェイルオーバーの監視 [10.6](#)
 - SWITCHOVER_STATUS列 [A.4.1](#), [G.3.1](#)
- V\$DATAFILEビュー [15.4.3](#), [19](#)
- V\$DATAGUARD_CONFIGビュー [19](#)
 - LOG_ARCHIVE_CONFIGで定義済のデータベース名のリスト表示 [19](#)
- V\$DATAGUARD_STATSビュー [9.1.2](#), [19](#)
- V\$DATAGUARD_STATUSビュー [10.6.1.6](#)
- V\$FS_FAILOVER_STATSビュー [19](#)
- V\$LOG_HISTORYビュー [10.6.1.5](#), [19](#)
- V\$LOGFILEビュー [19](#)
- V\$LOGSTDBY_PROCESSビュー [11.3.4](#), [11.4.1](#), [11.7.3.1](#), [11.7.3.2](#), [19](#)
- V\$LOGSTDBY_PROGRESSビュー [11.3.5](#), [19](#)
 - RESTART_SCN列 [11.1.1.3](#)
- V\$LOGSTDBY_STATEビュー [9.1.2](#), [11.3.6](#), [11.4.1](#), [19](#)
- V\$LOGSTDBY_STATSビュー [11.3.7](#), [19](#)
 - フェイルオーバー特性 [11.3.3](#)
- V\$LOGSTDBY_TRANSACTIONビュー [19](#)
- V\$LOGビュー [19](#)
- V\$MANAGED_STANDBYビュー [10.6.1.3](#), [19](#)
- V\$REDO_DEST_RESP_HISTOGRAM
 - 同期REDO転送のレスポンス時間の監視に使用 [7.6.2](#)
- V\$REDO_DEST_RESP_HISTOGRAMビュー [19](#)
- V\$STANDBY_EVENT_HISTOGRAMビュー [19](#)
- V\$STANDBY_LOGビュー [19](#)
- VALID_FOR属性 [17.17](#)
- 確認
 - ロジカル・スタンバイ・データベース [4.2.6](#)
 - フィジカル・スタンバイ・データベース [3.2.7](#)
- バージョン
 - Oracle Databaseソフトウェアのアップグレード [13.1](#)
- ビュー
 - DBA_LOGSTDBY_EVENTS [11.3.1](#), [19](#), [A.6](#)
 - DBA_LOGSTDBY_HISTORY [19](#)
 - DBA_LOGSTDBY_LOG [11.3.2](#), [19](#)
 - DBA_LOGSTDBY_NOT_UNIQUE [19](#)
 - DBA_LOGSTDBY_PARAMETERS [19](#)

- DBA_LOGSTDBY_SKIP [19](#)
- DBA_LOGSTDBY_SKIP_TRANSACTION [19](#)
- DBA_LOGSTDBY_UNSUPPORTED [19](#)
- V\$ARCHIVE_DEST [19](#)、[A.1.2](#)
- V\$ARCHIVE_DEST_STATUS [19](#)
- V\$ARCHIVED_GAP [19](#)
- V\$ARCHIVED_LOG [10.6.1.4](#)、[19](#)
- V\$DATABASE [19](#)
- V\$DATABASE_INCARNATION [19](#)
- V\$DATAFILE [15.4.3](#)、[19](#)
- V\$DATAGUARD_CONFIG [19](#)
- V\$DATAGUARD_STATS [19](#)
- V\$DATAGUARD_STATUS [10.6.1.6](#)
- V\$FS_FAILOVER_STATS [19](#)
- V\$LOG [19](#)
- V\$LOG_HISTORY [10.6.1.5](#)、[19](#)
- V\$LOGFILE [19](#)
- V\$LOGSTDBY_PROCESS [11.3.4](#)、[19](#)
- V\$LOGSTDBY_PROGRESS [11.3.5](#)、[19](#)
- V\$LOGSTDBY_STATE [11.3.6](#)、[19](#)
- V\$LOGSTDBY_STATS [11.3.7](#)、[19](#)
- V\$LOGSTDBY_TRANSACTION [19](#)
- V\$MANAGED_STANDBY [10.6.1.3](#)、[19](#)
- V\$REDO_DEST_RESP_HISTOGRAM [19](#)
- V\$STANDBY_EVENT_HISTOGRAM [19](#)
- V\$STANDBY_LOG [19](#)

W

- 待機イベント
 - REDO転送サービス [7.6.4](#)
- WAITING FOR DICTIONARY LOGS状態 [11.4.1](#)