Oracle Maximum
Availability Architecture

Oracle Active Data GuardとDBMS_ROLLINGを使用した自動データベース・アップグレード

ベスト・プラクティス

Oracleホワイト・ペーパー | 2017年12月

ORACLE

目次

はじめに

新しい Oracle パッチ・セットまたはデータベース・リリースへのデータベース・ローリング・アップグレードや他の計画メンテナンスを実行する手順はこれまで手動で行う必要がありましたが、Oracle Active Data Guard に新たな PL/SQL パッケージと DDL コマンドが追加されたことで自動化できるようになりました。プロセスの開始時には現行バージョンだったプライマリ・データベースとフィジカル・スタンバイ・データベースが、プロセスの終了時には両方とも新しいバージョンになります。自動化される処理には、本番データベースを新しいバージョンへ移行するスイッチオーバーも含まれます。また、プロセスのステップごとにさまざまな検証も実行されます。問題が発生した場合は、エラーを修正してアップグレードを再開するのか、構成を元の状態にロールバックするのかを選択できます。実装には新しい DBMS_ROLLING PL/SQL パッケージを使用します。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るようにするのがこのパッケージです。Oracle Active Data Guard の機能を使用してローリング・アップグレードを行うには、Oracle Active Data Guard オプションのライセンスが必要です。

このパッケージを使用してデータベース・バージョンのローリング・アップグレードを実行できるのは、データベースの現行バージョンが Oracle Database 12c の最初のパッチセット（12.1.0.2）以降の場合です。つまり、現行のデータベース・バージョンが 11g（リリース 1 または 2）あるいは 12.1.0.1 の場合は、一時ロジカル・スタンバイ・アップグレード手順を使用する必要があります。ただし、次のようなローリング・メンテナンス・タスクには DBMS_ROLLING パッケージを使用できます。

- 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
- BasicFile LOB から SecureFile 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 XMLType への変更
- OLTP 圧縮表への変更

DBMS_ROLLING パッケージは何度実行しても同じ結果が得られるように設計されているため、失敗した場合は失敗したステップからやり直すことができます。

概要

ローリング・アップグレードは、データベースを 2 つのグループ（先行グループと後続グループ）に分割して行います。

先行グループはメンテナンスを実行する最初のデータベース・グループで、'将来のプライマリ'データベースと、複数スタンバイ構成内で将来のプライマリを保護するよう指定されたデータベースが含まれます。

後続グループには、元のプライマリ・データベースと、先行グループでのメンテナンス実行中に元のプライマリを保護するよう指定されたスタンバイ・データベースが含まれます。

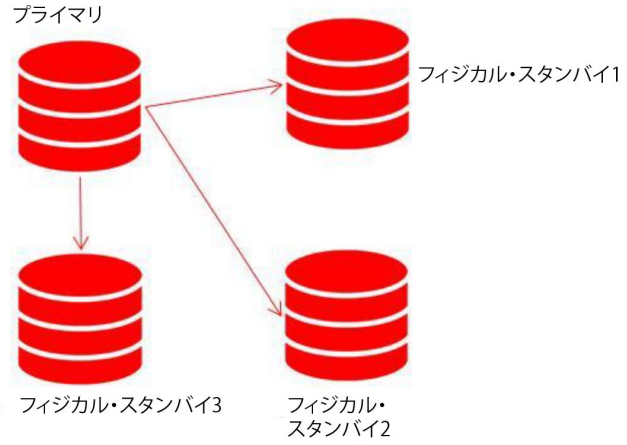

図 1：元の構成

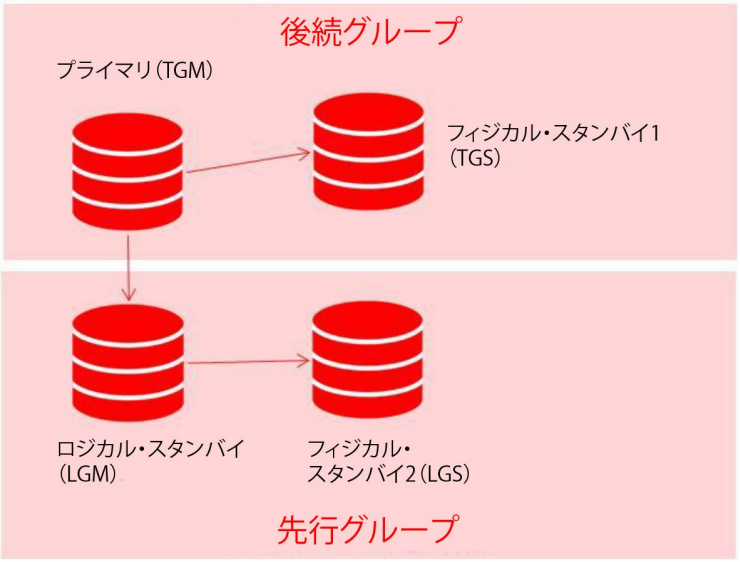

図 1：先行グループと後続グループを使用して START_PLAN を実行した後の構成

たとえば、アップグレード中は、先行グループは新しいデータベース・バージョンになっている一方で、後続グループは古いバージョンのデータベース・ソフトウェアで稼働しているということがあります。

このグループという概念があることで、次のような堅牢性が得られます。

» ローリング・アップグレード・プロセス中のエラーに対処できます。元のプライマリ、すなわち後続グループ・マスター（TGM）データベースでエラーが発生した場合は、後続グループに含まれる他のフィジカル・スタンバイに対して通常のフェイルオーバー操作を開始し、その後、新しいプライマリ・データベースを TGM として指定することができます。

» ローリング・アップグレード・プロセス中に、先行グループ・マスター（LGM：将来のプライマリに指定したデータベース）のデータを保護できます。LGM データベースにフィジカル・スタンバイを設定できるため、アップグレード・プロセス中に LGM データベースを保護することができ、アップグレード後のデータ消失ゼロも達成できます。LGM のアップグレードが正常に終了した後は、LGM でエラーが発生しても、先行グループに含まれる任意のフィジカル・スタンバイ・データベースにフェイルオーバーすれば対処できます。その後、フェイルオーバーしたターゲット・データベースに LGM のロールを引き継がせることができます。

もっとも単純なケース（フィジカル・スタンバイが 1 つの構成）では、TGM と LGM のみが存在します。

データベースのアップグレード手法の選択

データベースのアップグレードとは、データベースを新しいメジャー・リリース、メンテナンス・リリース、またはパッチ・セットに移行することです。データベースのアップグレードを実行する場合は次の Oracle 機能を使用できます。

» Database Upgrade Assistant（DBUA）を使用したアップグレード

» Data Guard SQL Apply 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アップグレード

» Oracle GoldenGate を使用したアップグレード

データベースをアップグレードするときに使用する手法は、次のことを考慮したうえで選択します。

» アップグレードを完了するまでに要する停止時間

» 停止時間の前に必要な設定時間および作業

» 一時的に必要となる追加リソース（ディスク領域、CPU など）

» アップグレードを完了するのに許容される手順の複雑さ

次の表に、データベースのアップグレードに使用できる手法と、その手法が推奨される具体的なケースをリストします。

データベースのアップグレード手法

アップグレード手法	この手法が適しているケース
Database Upgrade Assistant（DBUA）を使用したアップグレード	メンテナンス時間が十分にある場合、またはデータ型の制約によりこの表にある他の手法を使用できない場合に推奨される手法です。
Data Guard SQL Apply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アップグレード	DBUAではメンテナンス時間枠内に終了できず、データベースがOracle RACローリング・パッチ・アップグレードの対象ではない場合。フィジカル・スタンバイ・データベースが1つのみの構成の場合は、一時ロジカル・スタンバイを使用してください。
Oracle GoldenGateを使用したアップグレード	Oracle GoldenGateが完全なデータベース・レプリケーションにすでに使用されている場合、Oracle 10g（Oracle Data Guardによるデータベース・ローリング・アップグレードが可能な最小バージョン）より前のデータベース・バージョンを使用している場合、前のバージョンに戻すレプリケーションができる柔軟性が必要な場合（高速フォールバック・オプション）、またはマルチマスター・レプリケーションを使用した停止時間ゼロのアップグレードが必要な場合。

使用するアップグレード手法にかかわらず、『*Oracle Database アップグレード・ガイド*』のガイドラインと推奨事項、その関連ドキュメントである『Oracle 11gR2 Upgrade Companion』（My Oracle Support のノート 785351.1（https://support.oracle.com/rs?type=doc&id=785351.1）にあります）に沿ってアップグレードを行う必要があります。

DBMS_ROLLINGの使用

DBMS_ROLLING パッケージには 8 つのプロシージャが含まれています。この 8 つを、実行順に以下に示します。

INIT_PLAN

ターゲット・プライマリ（LGM）を設定し、構成に含まれるデータベースを特定し、計画パラメータをデフォルトに初期化します。初期化によってすべてのスタンバイ・データベースが後続グループに割り当てられますので、SET_PARAMETER を使用して手動で変更する必要があります。DBA_ROLLING_PARAMETERS を問い合わせると、INIT_PLAN で設定された値を表示できます。

例：exec dbms_rolling.init_plan('standby')

SET_PARAMETER

INIT_PLAN により生成されたパラメータを変更するのに使用します。これを使用するのは、スタンバイを先行グループに変更する場合や、プロセスの適切なタイムアウト値を設定する場合です。すべてのパラメータとその説明は、付録 A のリスト参照してください。

例：exec dbms_rolling.set_parameter('standby2','member','LEADING');

BUILD_PLAN

計画のパラメータを検証し、ローリング・プランを作成または変更します。DBA_ROLLING_PLAN を問い合わせて、BUILD_PLAN で生成されたすべてのステップを確認します。BUILD_PLAN は START_PLAN の前に実行する必要があり、SET_PARAMETER でパラメータを変更したら再実行する必要があります。

例：exec dbms_rolling.build_plan;

START_PLAN

ローリング操作を開始し、計画の START フェーズのすべてのステップを DBA_ROLLING_PLAN の記述に沿って実行します。これには、ROLLBACK または FINISH_PLAN の操作中に使用されるすべてのデータベースに保証付きリストア・ポイントを作成する手順も含まれます。

START_PLAN が正常に完了すると、INIT_PLAN に渡された将来のプライマリ・データベースは、完全に構成されたロジカル・スタンバイ兼先行グループ・マスター（LGM）になります。先行グループとして指定したスタンバイ・データベースは、LGM から REDO を受信する先行グループ・スタンバイ（LGS）になります。

例：exec dbms_rolling.start_plan;

SWITCHOVER

先行グループでのメンテナンスが完了した後に TGM で実行します。プライマリ TGM からロジカル・スタンバイ LGM へのスイッチオーバーはこのプロシージャで実行します。スイッチオーバーが完了すると LGM はプライマリ・データベースとなって読取り/書込みでオープンされ、TGM はロジカル・スタンバイ・データベースとしてマウントされます。

例：exec dbms_rolling.switchover;

FINISH_PLAN

FINISH_PLAN を実行すると、LGM がフィジカル・スタンバイ・データベースに変換され、後続グループに含まれるデータベースはすべて、START_PLAN で作成された保証付きリストア・ポイントにフラッシュされます。その後、メディア・リカバリが開始され、当該 SCN 以降のすべての REDO が LGM から適用され、データベースは同期されます。

例：exec dbms_rolling.finish_plan;

ROLLBACK_PLAN

構成全体をリストアし、START_PLAN を実行する前の状態に戻します。このプロシージャを呼び出せるのは、START_PLAN を呼び出した後に SWITCHOVER を実行していない場合のみです。

例：exec dbms_rolling.rollback_plan;

DESTROY_PLAN

このプロシージャは、存在するすべてのアップグレード計画とそのパラメータ、ローリング操作に関連するすべてのリソースを破棄します。例：exec dbms_rolling.destroy_plan;

プロセス・フロー

次のダイアグラムは DBMS_ROLLING のプロセス・フローを示しています。

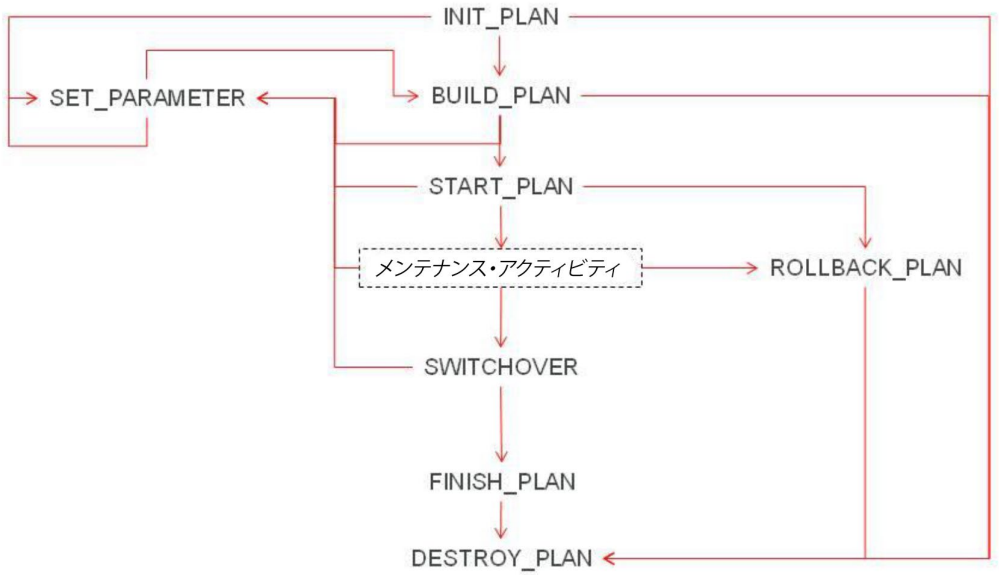

図 2.プロセス・フロー

DBMS_ROLLINGの各フェーズと停止時間/フォールバック・オプション

次の表は、DBMS_ROLLING プロセスのさまざまなフェーズと、そのフェーズのリカバリ時間目標（RTO）またはリカバリ・ポイント目標（RPO）への影響をまとめたものです。また、各フェーズのフォールバック・オプションもリストしています。

DBMS_ROLLING のフェーズ

DBMS_ROLLING のフェーズ	実行される操作	サービスの停止時間	RPO、RTOへの影響	フォールバック・オプション
前提条件フェーズ	11gまたは12.1.0.1からアップグレードする場合は、DBA_LOGSTDBY_UNSUPPORTED（プライマリがコンテナ・データベースの場合はCDB_LOGSTDBY_UNSUPPORTED）ビュー）を問い合わせます。 12.1.0.2からアップグレードする場合はDBA_ROLLING_UNSUPPORTEDビューを問い合わせます。 行が返されない場合は続行します。 行が返された場合は、アプリケーションの機能を制限するか、完全な停止時間を設けるかを判断します。	なし	なし	不要

dbms_rolling.init_plan	構成に含まれるデータベースを特定し、計画のパラメータを初期化します。	なし	RPOへの影響なしRTOへの影響なし	dbms_rolling.destroy_planを実行し、初期化済みの計画を削除します。
dbms_rolling.build_plan	計画のパラメータを検証し、ローリング計画を作成または変更します。	なし	RPOへの影響なしRTOへの影響なし	dbms_rolling.destroy_planを実行して計画のビルドを削除します。
dbms_rolling.start_plan	ローリング操作を開始し、STARTフェーズのすべての手順を実行します。START_PLANが正常に完了すると、将来のプライマリ・データベースは、完全構成されたロジカル・スタンバイになります。	なし	フィジカル・スタンバイをロジカル・スタンバイに変換するときのデータベース再起中はRPOがわずかに増加します。 RTO： ● フィジカル・スタンバイがロジカル・スタンバイに変換されるため、約2分に増加します。 ● ロジカル・スタンバイへのフェイルオーバーが許容される場合は、変換が完了するとRTOは前より短くなります。 ● ロジカル・スタンバイをフィジカル・スタンバイに戻す変換を行うdbms_rolling.rollback_planを実行する場合は、RTOが約3分増加します。 ● 後続グループにフィジカル・スタンバイを追加しても、RTOへの影響はありません。	start_planを実行する前の状態に構成全体をリストアするdbms_rolling.rollback_planを実行します。このプロシージャを呼び出せるのは、start_planを呼び出した後にスイッチオーバーを実行していない場合のみです。
データベースのアップグレード	データベースとそのインストール・オプションをDatabase Upgrade Assistant（推奨）または手動で12.1.0.2から12.2.0.1にアップグレードします。	なし	RPOへの影響なし RTO： ● ロジカル・スタンバイをアップグレードする前であればRTOへの影響はありません。 ● データベースをアップグレードした場合はフラッシュバックしてからでなければロジカル・スタンバイにフェイルオーバーできないため、アップグレード後またはアップグレード中はRTOが増加します。このフラッシュバックでRTO全体が1～2分長くなります。 ● フィジカル・スタンバイを追加すると、RTOへの影響はなくなります。	フォールバック・オプション： ● DBUAでアップグレードしている場合は、アップグレード前に作成したリストア・ポイントにデータベースをフラッシュバックします。テストによると、アップグレードしたデータベースのフラッシュバックは約30秒で完了します。 ● 手動でアップグレードする場合は、アップグレード前にリストア・ポイントを手動で作成しておく必要があります。 アップグレードしたデータベースのフラッシュバックは約30秒で完了します。 ● アップグレードの直前に取得したデータベース・バックアップをリストアします。

dbms_rolling.switchover	ロジカル・スタンバイ上でのアップグレードが完了した後にプライマリ上で実行します。ロジカル・スタンバイからプライマリへのスイッチオーバーはこのプロシージャで実行します。	15～20秒	RPO： ● スイッチオーバー前は影響なし ● スイッチオーバー後は、新しいプライマリ・データベースを更新するたびにRPOが増加します RTO： ● スイッチオーバー・プロセス中は約30秒増加します。 ● スイッチオーバーが完了した後も新しいロジカル・スタンバイを古いバージョンのままにしておくことにした場合は、RTOが増加し続けます。 ● 新しいプライマリ・データベース上でダウングレードを実行することにした場合も、RTOが増加します。	フォールバック・オプション： ● 12.2.0.1の新しいプライマリに対して何も更新が行われていない場合は、スイッチオーバーを実行して開始時の構成に戻します。 ● 12.2.0.1の新しいプライマリに対して更新が行われている場合は、ダウングレード・スクリプトを実行するのがフォールバック・オプションとなります。これはデータ損失ゼロのフォールバックですが、RTOが増加します。 ● 12.2.0.1の新しいプライマリに対する更新が始まった後は、スタンバイが関係するフォールバック・オプションを実行するとデータ損失が発生します。12.1.0.2のロジカル・スタンバイをアクティブにして、セッションをリダイレクトします。
12.2.0.1のホームでスタンバイ（元のTGM）を起動します。	ロジカル・スタンバイを停止し、12.2.0.1ソフトウェアでマウント状態に戻します。	なし	スタンバイの再起動中はRPOがわずかに（15秒）増加します。 ● スタンバイの再起動中はRTOがわずかに（15秒）増加します。	フォールバック・オプション： ● 12.1.0.2ソフトウェアでスタンバイを再起動します。
dbms_rolling.finish_plan	FINISH_PLANを実行すると、ロジカル・スタンバイ（以前のプライマリ）がフィジカル・スタンバイ・データベースに変換され、後続グループに含まれるデータベースはすべて、START_PLANで作成された保証付きリストア・ポイントにフラッシュされます。その後、メディア・リカバリが開始され、当該SCN以降のすべてのREDOが適用されます。	なし	スタンバイの再起動中はRPOがわずかに（15秒）増加します。 RTO： ● ロジカル・スタンバイがフィジカル・スタンバイにフラッシュバックされるため、RTOは増加します。 ● アップグレード・プロセスで発生したREDOがフィジカル・スタンバイに適用され始めると、RTOは減少し始めます。 アップグレード中のREDOがすべて適用され、フィジカル・スタンバイとプライマリが同期した状態になると、RTOは通常の値に戻ります。	フォールバック・オプション： ● ロジカル・スタンバイからフィジカル・スタンバイへの変換が失敗した場合は、プライマリ・データベースの新しいバックアップを使用してスタンバイを元の状態に戻すのがフォールバック・オプションとなります。

DBMS_ROLLINGプロセス

次の項では、DBMS_ROLLING を使用してアップグレードを実行する方法の例を詳しく説明します。

1. DBMS_Rolling プロセスの前提条件

ローリング・アップグレードを実行する前に、関係す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ていないかどうかを確認する必要があります。確認するには、DBA_ROLLING_UNSUPPORTED ビューを問い合わせます。

DBMS_ROLLING を使用してローリング・アップグレードを実行する場合のほうが、手動のローリング・アップグレード操作の場合より多くのオブジェクト・タイプがサポートされます。たとえば、キュー表がサポートされるのは DBMS_ROLLING を使用してアップグレードを実行する場合のみです。また、DBMS_ROLLING を使用してローリング・アップグレードを実行する場合のほうが、サポートされる PL/SQL パッケージも多くなっています。

DBMS_ROLLING 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場合のレプリケーションのサポートについては、付録 B を参照してください。

なお、ディクショナリの作成が実行されると、プライマリ（TGM）上で暗黙的にサプリメンタル・ロギングが有効化されます。サプリメンタル・ロギングがデータベースのパフォーマンスに影響する場合があるため、ローリング・アップグレードを実行する前に評価を行い、どのような影響があるかを確認する必要があります。また、アップグレード・プロシージャが完了したらサプリメンタル・ロギングを手動で無効化する必要があります。

Data Guard Broker 構成の観点から見ると、開始時の構成は次のようになっています。

```
Configuration - db1

Protection Mode:MaxPerformance

Members:

db1 - Primary database
db1stby1 - Physical standby database
db1stby2 - Physical standby database
db1stby3 - Physical standby database
```

ベスト・プラクティスとして、開始時点の既存のプライマリ・データベースはフィジカル・スタンバイで保護し、最終的に新しいプライマリ・データベースになるスタンバイは別のフィジカル・スタンバイで保護する必要があります。そのために、次のように REDO ルートの変更を行い、ブローカの構成を調整しました。

```
edit database db1 set property redoroutes='(LOCAL : db1stby1 SYNC, db1stby2 ASYNC)';
edit database db1stby1 set property redoroutes='(LOCAL : db1 SYNC)';
edit database db1stby2 set property redoroutes='(db1 : db1stby3 ASYNC)';
edit database db1stby3 set property redoroutes='(LOCAL : db1stby2 SYNC)';
```

この時点で、ブローカの構成は次のようになっています。

```
Configuration - db1

Protection Mode: MaxPerformance
Members:
db1 - Primary database
  db1stby1 - Physical standby database
  db1stby2 - Physical standby database
  db1stby3 - Physical standby database (receiving current redo)

Fast-Start Failover: DISABLED
```

このグループという概念があることで、次のような堅牢性が得られます。

» ローリング・アップグレード・プロセス中のエラーに対処できます。元のプライマリ（TGM）データベースでエラーが発生した場合は、後続グループに含まれる他のフィジカル・スタンバイに対して通常のフェイルオーバー操作を開始し、その後、新しいプライマリ・データベースをTGMとして指定することができます。

» ローリング・アップグレード・プロセス中に、LGM（将来のプライマリに指定したデータベース）のデータを保護できます。LGM データベースのフィジカル・スタンバイを設定できるため、アップグレード・プロセス中にLGMデータベースを保護することができ、アップグレード後のデータ消失ゼロも達成できます。

LGM のアップグレードが正常に終了した後は、LGM でエラーが発生しても、先行グループに含まれる任意のフィジカル・スタンバイ・データベースにフェイルオーバーすれば対処できます。その後、フェイルオーバーしたターゲット・データベースにLGMのロールを引き継がせることができます。

もっとも単純なケースでは、TGMとLGMのみが存在します。

2. DBMS_ROLLINGを使用したアップグレードの実行：ステップ・バイ・ステップ・プロセス

1. 計画を初期化します。ターゲット・プライマリ（LGM）を設定し、構成に含まれるデータベースを特定し、計画パラメータをデフォルトに初期化します。初期化によってすべてのスタンバイ・データベースが後続グループに割り当てられますので、SET_PARAMETER を使用して手動で変更する必要があります。DBA_ROLLING_PARAMETERS を問い合わせると、INIT_PLAN で設定された値を表示できます。

```
SQL> show parameter log_archive_config

NAME TYPE VALUE
------------------------------------ ----------- --------------------------
log_archive_config string dg_config=(db1,db1stby1,db1stb
y2,db1stby3)

SQL> exec dbms_rolling.init_plan('db1stby11');

PL/SQL procedure successfully completed.
```

2. 現行の計画のビルドのパラメータを表示します。

```
SQL> col name format a30
SQL> col scope format a15
SQL> col curval format a30
SQL> set pages 999|
SQL> select scope, name, curval from dba_rolling_parameters order by scope,
name;

SCOPE NAME CURVAL
--------------- ------------------------------ ----------------------------
db1 INVOLVEMENT FULL
db1 MEMBER NONE
db1stby1 INVOLVEMENT FULL
db1stby1 MEMBER TRAILING
db1stby2 INVOLVEMENT FULL
db1stby2 MEMBER TRAILING
db1stby3 INVOLVEMENT FULL
db1stby3 MEMBER TRAILING ACTIVE_SESSIONS_TIMEOUT 3600
ACTIVE_SESSIONS_WAIT 0
BACKUP_CONTROLFILE rolling_change_backup.f
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 1
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.
```

3. 計画をビルドします。計画のパラメータを検証し、ローリング・プランを作成または変更します。DBA_ROLLING_PLANを問い合わせて、BUILD_PLANで生成されたすべてのステップを確認します。BUILD_PLAN は START_PLAN の前に実行する必要があり、SET_PARAMETERでパラメータを変更したら再実行する必要があります。

```
SQL> exec dbms_rolling.build_plan;

PL/SQL procedure successfully completed.
```

4. 計画を表示します。アクティブなアップグレード計画を構成する手順をDBA_ROLLING_PLAN で表示できます。DBA_ROLLING_PLAN の各行を見ると、特定のデータベースでの実行がスケジュールされている具体的な手順がわかります。これらの手順は、DBMS_ROLLING.BUILD_PLAN プロシージャの呼出しが正常に完了すると作成されます。実行時には、手順のグループがリモート・データベースで実行されるようにバッチでスケジュールされます。手順のグループは、必ずBATCHIDの順に実行されることになっています。

```
SQL> col instid format 999
col target format a10
col phase format a10
col description format a65
set lines 99
set pages 999

SELECT instid, target, phase, description FROM DBA_ROLLING_PLAN;

INSTID TARGET PHASE DESCRIPTION
------ ---------- ---------- ---------------------------------------
--
1 db1 START Verify database is a primary
2 db1 START Verify MAXIMUM PROTECTION is disabled
3 db1stby1 START Verify database is a physical standby
4 db1stby1 START Verify physical standby is mounted
5 db1stby2 START Verify database is a physical standby
6 db1stby2 START Verify physical standby is mounted
7 db1stby3 START Verify database is a physical standby
8 db1stby3 START Verify physical standby is mounted
9 db1 START Verify server parameter file exists and is modifiable
10 db1stby1 START Verify server parameter file exists and is modifiable
11 db1stby2 START Verify server parameter file exists and is  modifiable
12 db1stby3 START Verify server parameter file exists and is  modifiable
13 db1 START Verify Data Guard Broker configuration is disabled
14 db1stby1 START Verify Data Guard Broker configuration is disabled
15 db1stby2 START Verify Data Guard Broker configuration is disabled
16 db1stby3 START Verify Data Guard Broker configuration is disabled
17 db1 START Verify flashback database is enabled
18 db1 START Verify available flashback restore points
19 db1stby1 START Verify flashback database is enabled
20 db1stby1 START Verify available flashback restore points
21 db1stby2 START Verify flashback database is enabled
```

```
22 db1stby2 START Verify available flashback restore points
23 db1stby3 START Verify flashback database is enabled
24 db1stby3 START Verify available flashback restore points
25 db1stby1 START Stop media recovery
26 db1stby2 START Stop media recovery
27 db1stby3 START Stop media recovery
28 db1stby1 START Drop guaranteed restore point DBMSRU_INITIAL
29 db1stby1 START Create guaranteed restore point DBMSRU_INITIAL
30 db1stby2 START Drop guaranteed restore point DBMSRU_INITIAL
31 db1stby2 START Create guaranteed restore point DBMSRU_INITIAL
32 db1stby3 START Drop guaranteed restore point DBMSRU_INITIAL
33 db1stby3 START Create guaranteed restore point DBMSRU_INITIAL
34 db1 START Drop guaranteed restore point DBMSRU_INITIAL
35 db1 START Create guaranteed restore point DBMSRU_INITIAL
36 db1stby1 START Start media recovery
37 db1stby1 START Verify media recovery is running
38 db1stby2 START Start media recovery
39 db1stby2 START Verify media recovery is running
40 db1stby3 START Start media recovery
41 db1stby3 START Verify media recovery is running
42 db1 START Verify user_dump_dest has been specified
43 db1 START Backup control file to rolling_change_backup.f
44 db1stby1 START Verify user_dump_dest has been specified
45 db1stby1 START Backup control file to rolling_change_backup.f
46 db1stby2 START Verify user_dump_dest has been specified
47 db1stby2 START Backup control file to rolling_change_backup.f
48 db1stby3 START Verify user_dump_dest has been specified
49 db1stby3 START Backup control file to rolling_change_backup.f
50 db1 START Get current redo branch of the primary database
51 db1stby1 START Wait until recovery is active on the primary's redo
   branch
52 db1stby1 START Stop media recovery
53 db1 START Execute dbms_logstdby.build
54 db1stby1 START Convert into a transient logical standby
55 db1stby1 START Open database
56 db1stby1 START Get redo branch of transient logical standby
57 db1stby1 START Get reset scn of transient logical redo branch
58 db1stby1 START Configure logical standby parameters
59 db1stby1 START Start logical standby apply
60 db1stby1 START Enable compatibility advance despite presence of GRPs
61 db1 START Log pre-switchover instructions to events table
62 db1stby1 START Record start of user upgrade of db1stby1
63 db1stby1 SWITCH Verify database is in OPENRW mode
64 db1stby1 SWITCH Record completion of user upgrade of db1stby1
65 db1stby1 SWITCH Scan LADs for presence of db1 destination
66 db1stby1 SWITCH Scan LADs for presence of db1stby2 destination
67 db1stby1 SWITCH Scan LADs for presence of db1stby3 destination
68 db1stby1 SWITCH Test if db1 is reachable using configured TNS service
69 db1stby1 SWITCH Test if db1stby2 is reachable using configured TNS
   service
70 db1stby1 SWITCH Test if db1stby3 is reachable using configured TNS
   service
```

```
71 db1 SWITCH Enable log file archival to db1stby1
72 db1stby1 SWITCH Start logical standby apply
73 db1stby1 SWITCH Archive all current online redo logs
74 db1stby1 SWITCH Wait until apply lag has fallen below 600 seconds
75 db1 SWITCH Log post-switchover instructions to events table
76 db1 SWITCH Switch database to a logical standby
77 db1stby1 SWITCH Wait until end-of-redo has been applied
78 db1stby2 SWITCH Wait until end-of-redo has been applied
79 db1stby3 SWITCH Wait until end-of-redo has been applied
80 db1 SWITCH Disable log file archival to db1stby2
81 db1 SWITCH Disable log file archival to db1stby3
82 db1 SWITCH Archive all current online redo logs
83 db1stby1 SWITCH Switch database to a primary
84 db1 SWITCH Enable compatibility advance despite presence of GRPs
85 db1stby2 SWITCH Enable compatibility advance despite presence of GRPs
86 db1stby3 SWITCH Enable compatibility advance despite presence of GRPs
87 db1stby2 SWITCH Stop media recovery
88 db1stby3 SWITCH Stop media recovery
89 db1 SWITCH Synchronize plan with new primary
90 db1 FINISH Verify only a single instance is active
91 db1 FINISH Verify database is mounted
92 db1 FINISH Flashback database
93 db1 FINISH Convert into a physical standby
94 db1stby2 FINISH Verify database is mounted
95 db1stby2 FINISH Flashback database
96 db1stby3 FINISH Verify database is mounted
97 db1stby3 FINISH Flashback database
98 db1stby1 FINISH Verify database is open
99 db1stby1 FINISH Save the DBID of the new primary
100 db1stby1 FINISH Save the logminer session start scn
101 db1 FINISH Wait until transient logical redo branch has been registered
102 db1stby2 FINISH Wait until transient logical redo branch has been
    registered
103 db1stby3 FINISH Wait until transient logical redo branch has been
    registered
104 db1 FINISH Start media recovery
105 db1stby2 FINISH Start media recovery
106 db1stby3 FINISH Start media recovery
107 db1 FINISH Wait until apply/recovery has started on the transient branch
108 db1stby2 FINISH Wait until apply/recovery has started on the transient
    branch
109 db1stby3 FINISH Wait until apply/recovery has started on the transient
    branch
110 db1 FINISH Wait until upgrade redo has been fully recovered
111 db1stby2 FINISH Wait until upgrade redo has been fully recovered
112 db1stby3 FINISH Wait until upgrade redo has been fully recovered
113 db1 FINISH Prevent compatibility advance if GRPs are present
114 db1stby1 FINISH Prevent compatibility advance if GRPs are present
115 db1stby2 FINISH Prevent compatibility advance if GRPs are present
116 db1stby3 FINISH Prevent compatibility advance if GRPs are present
117 db1 FINISH Drop guaranteed restore point DBMSRU_INITIAL
118 db1stby1 FINISH Drop guaranteed restore point DBMSRU_INITIAL
```

```
119 db1stby2 FINISH Drop guaranteed restore point DBMSRU_INITIAL
120 db1stby3 FINISH Drop guaranteed restore point DBMSRU_INITIAL

120 rows selected.
```

このビューの列で表示される情報は次のとおりです。

» INSTID - 手順 ID です。手順が実行される順序を表します。通常、手順はグループ単位で実行されます。

» PHASE - アップグレード計画に含まれる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 パッケージのプロシージャにより実行される手順を論理的にグループ化したものです。DBMS_ROLLING プロシージャが呼び出されると、アップグレード計画に含まれる手順のうち当該フェーズに関連するすべての手順が実行されます。考えられるフェーズは次のとおりです。

- START：設定に関連する操作として、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が行われます。このフェーズの操作は、DBMS_ROLLING.START_PLAN プロシージャを呼び出すと開始されます。
- SWITCH：一時ロジカル・スタンバイから新しいプライマリ・データベースにスイッチオーバーする処理に関連する操作が含まれます。このフェーズの操作は、DBMS_ROLLING.SWITCHOVER プロシージャを呼び出すと開始されます。
- FINISH：アップグレード REDO のリカバリ用スタンバイ・データベースを構成する処理に関連する操作が含まれます。このフェーズの操作は、DBMS_ROLLING.FINISH_PLAN プロシージャを呼び出すと開始されます。
  - EXEC_STATUS - 手順の全体的なステータスを表します。
  - PROGRESS - 手順の実行の進捗を表します。値 REQUESTING は、実行する手順をターゲット・データベースに送信中であることを示します。値 EXECUTING は、実際に手順を実行していることを示します。値 REPLYING は、完了情報が戻されている最中であることを示します。
  - DESCRIPTION - 実行されることになっている具体的な操作です。
  - TARGET - 所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。

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場合は、アップグレード計画を修正する必要があります。構成変更には次のようなものが考えられます。

- ローリング・アップグレードに関係しているデータベースのいずれかで init.ora パラメータ・ファイルを変更した
- フェイルオーバー・イベントの結果としてデータベース・ロールが変更された
- ローリング・アップグレード・パラメータを変更した

アクティブなアップグレード計画は、BUILD_PLAN プロシージャを再度呼び出すだけで修正できます。ただし、許容できない変更があった場合は、BUILD_PLAN プロシージャでエ

ラーが発生することもあります。たとえば、スイッチオーバーがすでに発生していた場合は、ACTIVE_SESSIONS_WAIT パラメータを設定しても何の効果もありません。

パラメータは個別に処理するのではなく、BUILD_PLAN プロシージャを呼び出して複数のパラメータ変更をまとめて処理することをお薦めします。

次に示すのは、適用ラグが60秒未満になってから将来のプライマリにスイッチオーバーするよう計画を構成する方法の例です。

```
DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_WAIT', '1');
DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_TIME', '60');
```

次に示すのは、LOG_LEVEL グローバル・パラメータをデフォルト値にリセットする例です。

```
DBMS_ROLLING.SET_PARAMETER (
    name=>'LOG_LEVEL',
    value=>NULL);
```

5. start_plan を実行します。ローリング操作を開始し、計画の START フェーズのすべての手順を DBA_ROLLING_PLAN の記述に沿って実行します。これには、ROLLBACK または FINISH_PLAN の操作中に使用されるすべてのデータベースに保証付きリストア・ポイントを作成する手順も含まれます。START_PLAN が正常に完了すると、INIT_PLAN に渡された将来のプライマリ・データベースは、完全に構成されたロジカル・スタンバイ兼先行グループ・マスター（LGM）になります。先行グループとして指定したスタンバイ・データベースは、LGM から REDO を受信する先行グループ・スタンバイ（LGS）になります。

```
SQL> exec dbms_rolling.start_plan

PL/SQL procedure successfully completed.
```

EBS ベンチマーク・キットによると、EBS R12.2.5 でのディクショナリの作成時間は、フル・インストールで約70秒かかりました。オブジェクトの数もさることながら、作成時間にもっとも影響する要因は、

"DBMS_LOGSTDBY.BUILD は、プロシージャ起動時にアクティブなすべてのトランザクション（分散トランザクションを含む）が完了するまで待機してから戻る"ということです。

そのため、長時間実行しているトランザクションに完了を妨げられる可能性があります。幸い、オンラインで実行した場合は、ディクショナリの作成によるアプリケーションへの影響は確認されていません。

また、ディクショナリの作成を実行すると、このコマンド（alter database add supplemental log data (primary key, unique index) columns）が暗黙的に実行されます。start_plan を実行する前の、データベース起動時のステータスは次のとおりです。

```
SQL> select SUPPLEMENTAL_LOG_DATA_MIN, SUPPLEMENTAL_LOG_DATA_PK,
SUPPLEMENTAL_LOG_DATA_UI from v$database;

SUPPLEME SUP SUP
-------- --- ---
NO       NO  NO
```

start_plan が終了すると、ステータスは次のように変わります。

```
SQL> select SUPPLEMENTAL_LOG_DATA_MIN, SUPPLEMENTAL_LOG_DATA_PK,
SUPPLEMENTAL_LOG_DATA_UI from v$database;

SUPPLEME SUP SUP
-------- --- ---
IMPLICIT YES YES
```

DBMS_ROLLING プロセスが完了してもサプリメンタル・ロギングは有効のままであるため、手動で無効にする必要があります。

6. Database Upgrade Assistant（DBUA）または好みの手法を使用して、スタンバイ（現在のロジカル・スタンバイ）でアップグレードを実行します。データベースのアップグレードはアップグレード・ガイドに従って行ってください。

```
SQL> exec dbms_rolling.switchover;

PL/SQL procedure successfully completed.
```

7. 現在は、プライマリだったものがロジカル・スタンバイになり、スタンバイだったものがプライマリになっています。

```
SQL> select database_role,open_mode from v$database;
DATABASE_ROLE OPEN_MODE
---------------- --------------------
LOGICAL STANDBY READ WRITE

SQL> select database_role,open_mode from v$database;
DATABASE_ROLE OPEN_MODE
---------------- --------------------
PRIMARY READ WRITE
```

8. 新しいホームで以前のプライマリ・インスタンスを起動します。

```
SQL> shutdown immediate
Database closed.
Database dismounted
ORACLE instance shut down.
SQL> exit
```

9. init.ora とパスワード・ファイルを新しい ORACLE_HOME/dbs にコピーし、該当する tnsnames.ora 記述子を新しいホームの下の適切な tnsnames.ora ファイルにコピーします。新しいホームが使用されるように環境を設定し、スタンバイ・インスタンスをマウントします。

```
SQL> startup mount
ORACLE instance started.
Total System Global Area 2.5770E+10 bytes
Fixed Size 6870952 bytes
Variable Size 3422554200 bytes
```

```
Database Buffers 2.2012E+10 bytes
Redo Buffers 328671232 bytes
Database mounted.
```

10. 新しいスタンバイをフィジカル・スタンバイに変換し、フィジカル・スタンバイをフラッシュバックしてロールフォワードするために、新しいプライマリで DBMS_ROLLING.FINISH_PLAN を実行します（finish_plan で実行されるすべての手順は、dba_rolling_plan で確認できます）。

```
SQL> exec dbms_rolling.finish_plan;

PL/SQL procedure successfully completed.
```

11. 適用を実行しながらプライマリがマウントされます。

```
SQL> select database_role,open_mode from v$database;

DATABASE_ROLE OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY MOUNTED

SQL> select status from v$managed_standby where process='MRP0';

STATUS
------------
APPLYING_LOG
```

3. DBMS_ROLLING アップグレード後の作業：

DBMS_ROLLING プロセスが完了してもサプリメンタル・ロギングは有効のままであるため、手動で無効にする必要があります。サプリメンタル・ロギングを無効にするために次のコマンドを実行します。

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;
ALTER DATABASE DROP SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
ALTER DATABASE DROP SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;
ALTER DATABASE DROP SUPPLEMENTAL LOG DATA;
```

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ために、次のように、DBMS_ROLLING.ROLLBACK_PLAN プロシージャを呼び出すことができます。

```
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 プロシージャを呼び出せるのは、DBMS_ROLLING.SWITCHOVER プロシージャをそれまで呼び出したことがない場合のみです。
- ROLLBACK_PLAN プロシージャを使用する前に、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。フラッシュバック・データベースが急迫しているためです。
- Oracle Database ソフトウェアがすでにアップグレードされていた場合は、結果としてフィジカル・スタンバイになったデータベースを古いバージョンで再起動し、メディア・リカバリを開始する必要があります。

付録A

DBMS_ROLLING.SET_PARAMETERプロシージャの有効な値

パラメータ名	グローバルか否か	説明	デフォルト
ACTIVE_SESSIONS_TIMEOUT	はい	ACTIVE_SESSIONS_WAITを実行してからローリング・アップグレードを停止するまでの最長時間（秒）。このパラメータは、ACTIVE_SESSIONS_WAITが1に設定されている場合のみ有効です。	3600
ACTIVE_SESSIONS_WAIT	はい	アクティブなセッションが終了するまでスイッチオーバー操作を待機させるかどうかを指定します。1に設定した場合は、アクティブなセッションが完了するまでSWITCHOVERプロシージャは起動されません。0に設定した場合はSWITCHOVERプロシージャが起動され、アクティブなセッションが強制終了されてスイッチオーバーが行われます。	0
BACKUP_CONTROLFILE	はい	ローリング・アップグレード中に作成されるバックアップ制御ファイルのファイル名。	rolling_change_backup.f
DGBROKER	はい	適用、リカバリおよびログのアーカイブ先を管理するのにData Guard Brokerを使用します。	ブローカが有効化されている場合は1、そうでない場合は0。
DICTIONARY_LOAD_TIMEOUT	はい	DICTIONARY_LOAD_WAITを実行してからローリング・アップグレードを停止するまでの最長時間（秒）。 このパラメータは、DICTIONARY_LOAD_WAITが1に設定されている場合のみ有効です。	3600
DICTIONARY_LOAD_WAIT	はい	一時ロジカル・スタンバイをインスタンス化するときに、データ・ディクショナリのスナップショットがREDOに完全にロードされるまで待機するかどうかを指定します。1に設定した場合は、ディクショナリが完全にロードされるまでSTART_PLANプロシージャは戻りません。0に設定した場合は、ディクショナリのロードが開始されたかどうかの確認のみがSTART_PLANプロシージャで行われます。	0
DICTIONARY_PLS_WAIT_INIT	はい	データ・ディクショナリをREDOに書き込むためにPL/SQL操作の停止を試行する間隔（秒）。	300
DICTIONARY_PLS_WAIT_TIMEOUT	はい	データ・ディクショナリをREDOに書き込むためにPL/SQL操作の停止を試行する最長時間（秒）。	3600
EVENT_RECORDS	はい	DBA_ROLLING_EVENTSで許可するレコードの最大数。	10000
FAILOVER	はい	フェイルオーバー・イベントが発生した場合は、アップグレード計画の調整が自動的に試行されます。このパラメータの値は、後続のBUILD_PLANへの呼出しが完了すると0にリセットされます。	0
GRP_PREFIX	適用外	DBMS_ROLLINGに含まれるプロシージャを実行すると、Data Guardを構成している各種データベースに多数の保証付きリストア・ポイント（GRP）が作成されます。そのようなGRPにはどれも同じ接頭辞を含む名前が付けられます。このパラメータを使用すると、デフォルトの接頭辞をオーバーライドできます。	DBMSRU
IGNORE_BUILD_WARNINGS	はい	無視しなかった場合はBUILD_PLANプロシージャの実行中に例外を発行することになる警告を無視します。	1
IGNORE_LAST_ERROR	はい	前回発生したエラーを無視し次のローリング操作を開始します。このパラメータの値は、ローリング・アップグレードを再開するプロシージャを呼び出すと0にリセットされます。	0
LAD_ENABLED_TIMEOUT	はい	最近有効化されたログ・アーカイブ先がVALID状態になるまで待機する最長時間（秒）。	600
LOG_LEVEL	はい	DBS_ROLLING PL/SQLパッケージのロギング・レベル。値INFOを指定すると、エラーと、関連する致命的でない警告のログが取得されます。値FULLを指定すると、すべてのイベントのログが取得されます。	INFO

MEMBER	いいえ	指定したデータベースをメンバーとして含むアップグレード・グループ。 値LEADINGは、スタンバイが先行アップグレード・グループのメンバーであることを示します。つまり、これは先行グループ・マスター（LGM）のスタンバイです。LGMは、一時ロジカル・スタンバイに変換され、スイッチオーバー後に新しいプライマリ・データベースになるデータベースです。 値TRAILINGは、スタンバイが後続アップグレード・グループのメンバーであることを示します。つまり、これは後続グループ・マスター（TGM）のスタンバイです。TGMは元のプライマリ・データベースです。	LEADING
READY_LGM_LAG_TIME	はい	READY_LGM_LAG_WAITパラメータに関連付けられている適用ラグ時間（秒）。	600
READY_LGM_LAG_TIMEOUT	はい	READY_LGM_LAG_WAITを実行してからローリング・アップグレードを停止するまでの最長時間（秒）。このパラメータは、READY_LGM_LAG_WAITが1に設定されている場合のみ有効です。	60
READY_LGM_LAG_WAIT	はい	START_PLANプロシージャでユーザーに制御権を戻すときに、先行グループ・マスターでの適用ラグがREADY_LGM_LAG_TIME秒未満になるまで待機するかどうかを指定します。1に設定すると待機が実行されます。0に設定すると待機は実行されません。	0
SWITCH_LGM_LAG_TIME	はい	SWITCH_LGM_LAG_WAITパラメータに関連付けられている適用ラグ時間（秒）。	600
SWITCH_LGM_LAG_TIMEOUT	はい	SWITCH_LGM_LAG_WAITを実行してからローリング・アップグレードを停止するまでの最長時間（秒）。このパラメータは、SWITCH_LGM_LAG_WAITが1に設定されている場合のみ有効です。	60
SWITCH_LGM_LAG_WAIT	はい	SWITCHOVERプロシージャでスイッチオーバーを開始するときに、先行グループ・マスターの適用ラグがSWITCH_LGM_LAG_TIME秒未満になるまで待機するかどうかを指定します。1に設定すると待機が実行されます。0に設定すると待機は実行されません。	1
SWITCH_LGS_LAG_TIME	はい	SWITCH_LGS_LAG_WAITパラメータに関連付けられている適用ラグ時間（秒）。	60
SWITCH_LGS_LAG_TIMEOUT	はい	SWITCH_LGS_LAG_WAITを実行してからローリング・アップグレードを停止するまでの最長時間（秒）。このパラメータは、SWITCH_LGS_LAG_WAITが1に設定されている場合のみ有効です。	60
SWITCH_LGS_LAG_WAIT	はい	SWITCHOVERプロシージャでスイッチオーバーを開始するときに、先行グループのスタンバイでの適用ラグがSWITCH_LGS_LAG_TIME秒未満になるまで待機するかどうかを指定します。1に設定すると待機が実行されます。0に設定すると待機は実行されません。	0
UPDATED_TGS_WAIT	はい	FINISH_PLANプロシージャでユーザーに制御権を戻すときに、後続グループのスタンバイですべてのアップグレードREDOのリカバリが完了するまで待機するかどうかを指定します。1に設定すると待機が実行されます。0に設定すると待機は実行されません。	1

付録B

次に示すのは、DBMS_ROLLING 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場合にサポートされないデータ型のリストです。

12.2.0.1 の時点でサポートされないデータ型：

以下のタイプの列が使用されている表は、ロジカル・スタンバイではサポートされません。

1. ネストした表
2. ID 列
3. 行 ID
4. BFILE
5. 時制有効性列
6. PKREF
7. PKOID
8. SDO_RDF_TRIPLE_S

サポートされないパーティション化/表の編成方法：

次の方法でパーティション化された表は、ロジカル・スタンバイではサポートされません。

1. 参照パーティション化
2. システム・パーティション化

DBA_LOGSTDBY_NOT_UNIQUE

DBA_LOGSTDBY_NOT_UNIQUE ビューで表示されるのは、一次索引も非 NULL の一意索引もないすべての表です。これらの表に LOB、XML などの表外列もある場合は、データの不一致が発生しやすいため、表のレプリケーションは行わないようにしてください。索引がない表をレプリケートする場合は、レプリケート中に表のスキャンやパフォーマンスの問題が発生しやすくなります。

スカラー列がない表：

データ型が LONG、LONG RAW、LOB（CLOB/BLOB）、XML、ADT、VARRAY、または BFILE の列のみの表（またはこれらの一部が混在している表）はサポートされません。レプリケーション時にこれらのデータ型を使用して行を特定することができないためです。

レプリケーションされないスキーマ：

次のスキーマはロジカル・スタンバイでは自動的にスキップされます。

ANONYMOUS	APPQOSSYS	AUDSYS
BI	CTXSYS	DBSFWUSER
DBSNMP	DIP	DMSYS
DVF	DVSYS	EXDSYS
EXFSYS	GGSYS	GSMADMIN_INTERNAL
GSMCATUSER	GSMUSER	LBACSYS

MDSYS	MGMT_VIEW	MTSSYS
ODM	ODM_MTR	OJVMSYS
OLAPSYS	ORACLE_OCM	ORDDATA
ORDPLUGINS	ORDSYS	OUTLN
REMOTE_SCHEDULER_AGENT	SI_INFORMTN_SCHEMA	SPATIAL_CSW_ADMIN
SPATIAL_CSW_ADMIN_USR	SPATIAL_WFS_ADMIN	SPATIAL_WFS_ADMIN_USR
SYS	SYS$UMF	SYSBACKUP
SYSDG	SYSKM	SYSMAN
SYSRAC	SYSTEM	TSMSYS
WKPROXY	WKSYS	WK_TEST
WMSYS	XDB	XS$NULL
XTISYS		

レプリケートされない特定の DDL

レプリケートされない DDL にはさまざまなものがあります。以下にリストします。

このリストには、次のオブジェクトに対する DDL（の一部）が含まれます。

DATABASE
PLUGGABLE DATABASE
CONTROL FILE
SPFILE/PFILE
DISK GROUP
SNAPSHOT
SUMMARY
DATABASE LINK
RECYCLE BIN
RESTORE POINT
ASSEMBLY
FLASHBACK ARCHIVE

システム生成名：

ロジカル・レプリケーションを実行すると名前が変更される場合があるため、システム生成名は問題となる可能性があります。そのため、システム生成名が付けられているオブジェクトに対して DDL を発行すると、通常、レプリケーションは失敗します（システム生成索引に対する DDL は一般的です）。

エディションベースの再定義

エディションベースの操作に対する基本的なサポートはありますが、エディションベースの再定義はオンライン DDL の実行に依存しているのが一般的です。RDBMS 11.x および 12.x で導入されたオンライン DDL 操作（ALTER TABLE ADD COLUMN など）は、ローリング・アップグレード時（またはサプリメンタル・ロギングが有効化されているとき）のオンライン操作としてはサポートされません。代わりに、そうした操作はオンライン実行ができないブロッキング・モデルに暗黙的にダウングレードされます。

アプリケーション・コンテナ

アプリケーション・コンテナの基本的なレプリケーションはサポートされますが、ローリング・アップグレード中にユーザーがアプリケーションのINSTALL、UPGRADE、PATCH操作を実行することはできません。

UNSUPPORTEDのプラグマが指定されているプロシージャ

プラグマがUNSUPPORTEDのPL/SQLの場合はプロシージャ呼出しの時点でSQLの適用が停止するため、手動操作を行うことができます。

スキーマ	パッケージ	プロシージャ	プラグマ
SYS	DBMS_REDEFINITION	ABORT_UPDATE	PRAGMA UNSUPPORTED
SYS	DBMS_REDEFINITION	EXECUTE_UPDATE	PRAGMA UNSUPPORTED
XDB	DBMS_XDBZ	ADD_APPLICATION_PRINCIPAL	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDBZ	CHANGE_APPLICATION_MEMBERSHIP	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDBZ	DELETE_APPLICATION_PRINCIPAL	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDBZ	SET_APPLICATION_PRINCIPAL	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_ADMIN	CREATENONCEKEY	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_ADMIN	INSTALLDEFAULTWALLET	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_ADMIN	MOVEXDB_TABLESPACE	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_ADMIN	REBUILDHIERARCHICALINDEX	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ADDAUTHENTICATIONMAPPING	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ADDAUTHENTICATIONMETHOD	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ADDTRUSTMAPPING	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ADDTRUSTSCHEME	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	CLEARHTTPDIGESTS	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	DELETEAUTHENTICATIONMAPPING	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	DELETEAUTHENTICATIONMETHOD	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	DELETETRUSTMAPPING	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	DELETETRUSTSCHEME	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ENABLECUSTOMAUTHENTICATION	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ENABLECUSTOMTRUST	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ENABLEDIGESTAUTHENTICATION	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	ISGLOBALPORTENABLED	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	SETDYNAMICGROUPSTORE	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	SETGLOBALPORTENABLED	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XDB_CONFIG	SETHTTPCONFIGREALM	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XMLINDEX	DROPPARAMETER	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XMLINDEX	MODIFYPARAMETER	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XMLINDEX	REGISTERPARAMETER	PRAGMA UNSUPPORTED w/ COMMIT
XDB	DBMS_XMLSCHEMA	COPYEVOLVE	PRAGMA UNSUPPORTED w/ COMMIT

付録C

DBMS_ROLLING を使用したアップグレードの処理前、処理中、処理後のステータスは、次のビューを使用して確認できます。

DBA_ROLLING_DATABASES

DBA_ROLLING_DATABASES では、ローリング操作による構成の対象となっているすべてのデータベースをリストできます。

DBA_ROLLING_EVENTS

DBA_ROLLING_EVENTS では、DBMS_ROLLING PL/SQL パッケージからレポートされたすべてのイベントをリストできます。

DBA_ROLLING_PARAMETERS

DBA_ROLLING_PARAMETERS では、DBMS_ROLLING PL/SQL パッケージで利用できるパラメータをリストできます。

DBA_ROLLING_PLAN

DBA_ROLLING_PLAN では、アクティブなアップグレード計画を構成する手順を表示できます。DBA_ROLLING_PLAN の各行を見ると、特定のデータベースでの実行がスケジュールされている具体的な手順がわかります。これらの手順は、DBMS_ROLLING.BUILD_PLAN プロシージャの呼出しが正常に完了すると作成されます。

実行時には、手順のグループがリモート・データベースで実行されるようにバッチでスケジュールされます。手順のグループは、必ず BATCHID の順に実行されることになっています。

DBA_ROLLING_STATISTICS

DBA_ROLLING_STATISTICS では、ローリング操作の統計の一覧を表示できます。

DBA_ROLLING_STATUS

DBA_ROLLING_STATUS では、ローリング操作の全体的なステータスを表示できます。

DBA_ROLLING_UNSUPPORTED

DBA_ROLLING_UNSUPPORTED では、DBMS_ROLLING PL/SQL パッケージを使用したロジカル・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、表および列を表示できます。DBMS_ROLLING を使用したローリング・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。

付録D

クラウド・データベース・アップグレードのユースケース：

» CDB に限定

» 15 分後にサービス停止を伴う短い計画メンテナンス時間枠（例：1 時間）。ワークロード量が非常に少なく、長時間実行のトランザクションとバッチが 1 つもない時間帯を計画メンテナンス時間枠として推奨

» 最初のスイッチオーバーによるサービス停止時間が 1 分未満（稼働時間に関する SLA に含まれます）

» スイッチオーバー後の評価フェーズが 30 分

» 元のプライマリへのスイッチバックが必要になった場合のサービス停止時間が 1 分未満（**稼働時間に関する SLA には含まれません**）

» 30 分間の検証とスイッチバック後の単純なロールバック。推定時間は 5 分（**アプリケーションのパフォーマンス・テストと事前検証はアップグレード前に顧客が実施することが前提条件であるため、稼働時間に関する SLA には含まれません**）

» データベース障害の結果としてのデータベース・フェイルオーバー・オプションに起因する停止時間は、引き続き稼働時間に関する SLA に含まれます

データベース・アップグレードのユースケースに Oracle GoldenGate よりも DBMS_ROLLING が適している理由

» 自動化とチェックが組み込まれているため、大幅に簡素化

» データ型の制約は基本的に GoldenGate と同じ

» 既存のスタンバイ・データベースを活用できるため、ストレージやシステム・リソースの追加要件を排除。スタンバイ・データベースは非常に一般的なマネージド・クラウド実装になります

» 追加したスタンバイ・データベースと統合可能

データベース・アップグレードのユースケースに固有の GoldenGate の利点

» データ損失なしで旧リリースにフォールバック可能

Oracle GoldenGate と DBMS_ROLLING の両方のソリューションに該当する考慮事項

» データ型と API に関わる制限。

» 特定のワークロードでのデータ・ラグが懸念される場合があります。特定のトランザクション（例：バッチ）のパフォーマンス・チューニングが困難な場合があります。

» Oracle GoldenGate と DBMS_ROLLING のいずれのソリューションも、スタンバイ・データベースを追加することでメリットが得られます。

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口
電話：**+1.650.506.7000**
ファクシミリ：**+1.650.506.7200**

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services

Oracle Active Data Guard DBMS Rolling MAA ベスト・プラクティス
2017 年 12 月
著者：Michael T Smith
共著者：