Rolling Database-Upgrade mit Data Guard

Claudia Hüffer
Oracle Deutschland B.V. & Co. KG
Hamburg

Schlüsselworte
Oracle Data Guard, Rolling Upgrade, DBMS_ROLLING, Transient Logical Standby, Active Data Guard, Extended Datatype Support

1. Einleitung

Data Guard Umgebungen können mit Hilfe der Logical Standby Datenbank auch eingesetzt werden, um bei geplanten Datenbank-Upgrades die dafür notwendige Downtime zu verringern. Seit der Oracle-Version 10.1.0.3 aufwärts kommt dafür die Logical Standby zum Einsatz. Mit Oracle 11gR1 wurde die sogenannte Transient Logical Standby eingeführt. Mit Oracle 12c wurde das Upgrade-Verfahren erneut erweitert. In Verbindung mit Active Data Guard steht für Datenbank-Upgrades jetzt das PL/SQL-Paket DBMS_ROLLING zur Verfügung. Der Vortrag beschreibt das Vorgehen eines Rolling Database Upgrades mit Data Guard mit minimaler Downtime. Das Vorgehen für Oracle 11g R2 als auch das mit Oracle 12c und Active Data Guard eingeführte PL/SQL-Paket DBMS_ROLLING werden im Rahmen des Vortrages vorgestellt.

2. Oracle Data Guard – ein Überblick

Oracle Data Guard ist DIE Disaster-Recovery-Lösung von Oracle. Eingeführt mit Oracle 7.3 ist die Oracle Standby Datenbank Technologie von Version zu Version stets weiterentwickelt worden und seit Oracle 9i unter dem Namen Data Guard bekannt. Oracle Data Guard bietet eine Software Infrastruktur zur Erstellung und Verwaltung einer oder mehrerer Standby Datenbanken als transaktionskonsistente Kopien der Produktions-Datenbank. Data Guard dient dem Schutz der unternehmenskritischen Daten vor Fehlern, Ausfällen, Desastern und Korruptionen.
Sollte die Produktions-Datenbank aus irgendwelchen planmäßigen oder außerplanmäßigen Gründen ausfallen, kann eine der angeschlossenen Standby Datenbanken direkt die Produktionsaufgaben übernehmen. Somit werden Ausfallzeiten minimiert und Datenverluste verhindert.
Data Guard ist Bestandteil der Oracle Enterprise Edition. Seit Oracle 11g Release 1 kann die Physical Standby Datenbank bei gleichzeitigem Apply lesend für Abfragen geöffnet werden. Dies erfordert die Lizenzierung der Active Data Guard Option. Der Funktionsumfang der Active Data Guard Option wurde mit 11gR2 und 12c deutlich erweitert. Die Active Data Guard Option umfasst mittlerweile Real-Time Query, Automatic Block Repair, Far Sync (Zero-Dataloss-Datenschutz auch über große Distanzen), RMAN Block Change Tracking (für beschleunigtes Backup auf der Standby-Seite), Global Database Services (Load Balancing und automatisiertes Service Management in replizierten Umgebungen), Application Continuity (verbesserter Ausfallschutz für DB-Applikationen) und das Active Data Guard Rolling Upgrade, das im Folgenden näher beschrieben wird.
Data Guard kann mit anderen Oracle Hochverfügbarkeits-Funktionalitäten wie zum Beispiel Real Application Clusters (RAC), Flashback Database und Recovery Manager (RMAN) kombiniert werden, um die Basis für höchstverfügbare Datenbank-Anwendungen zu bilden.
Eine Data Guard Konfiguration besteht immer aus einer Primary Datenbank und mindestens einer Standby Datenbank. Standby Datenbank–Konfigurationen können auch kaskadierend aufgebaut sein. Idealerweise sind beide Datenbanken räumlich getrennt voneinander installiert, um optimalen Ausfallschutz zu ermöglichen. Änderungen, die durch die Applikationen in der Primary Datanbank

durchgeführt werden, werden kontinuierlich über die RedoLog Informationen auf das Standby Datenbank-System übertragen und dort eingespielt (applied). Zur Administration einer Data Guard Umgebung kann SQL*Plus, das Command Line Interface `dgmgrl` des Data Guard Brokers oder idealerweise Enterprise Manager Cloud Control verwendet werden. Es gibt grundsätzlich zwei Arten von Standby Datenbanken – die Physical Standby Datenbank und die Logical Standby Datenbank. Bei der Physical Standby Datenbank kann zwischen verschiedenen Betriebsmodi gewählt werden; entweder befindet sich die Physical Standby Datenbank in einem gemounteten Zustand und fährt die Änderungen der Primary Redo Log Informationen nach. Oder sie ist im Snapshot Standby Mode für Applikationstest oder sie ist im Lesezugriff UND spielt die Änderungen ein. Letzteres ist der sogenannte Real Time Query Mode, der Bestandtteil der Active Data Guard Option ist.

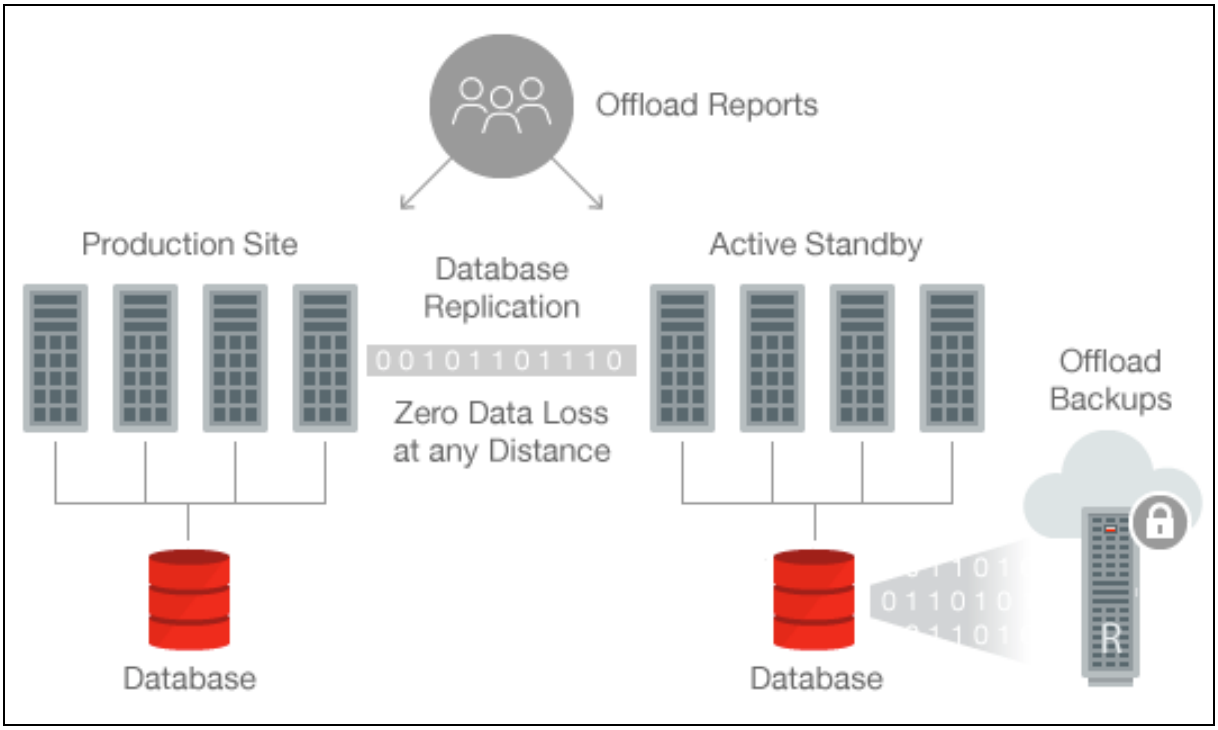


Abb. 1: Oracle Data Guard

a) Physical Standby

Bei der Physical Standby Datenbank ist der MRP (Managed Recovery Process) für das Apply verantwortlich. D.h., die Änderungen, die auf der Primär-Datenbank stattgefunden haben, werden durch ein Recovery in der Physical Standby Datenbank nachgezogen. Daher ist die Physical Standby Datenbank blockweise 1:1 identisch mit der Primär-Datenbank und erlaubt zum einen einen optimalen Desaster-Schutz, zum anderen kann die Physical Standby Datenbank für Backups herangezogen werden und somit das Primärsystem entlasten. Die Physical Standby Datenbank bietet somit optimalen Desaster-Schutz. Mit Hilfe der Snapshot Standby lässt sich die Physical Standby Datenbank auch temporär schreibend und lesend für Applikationstests einsetzen. Änderungen, die während der Tests stattfinden, werden nach Abschluss und Rückumwandlung in die Physical Standby Datenbank automatisch zurückgerollt. Eine Physical Standby Datenbank ist in der Regel immer im Apply-Mode und steht Benutzern wegen des Mount-Status nicht für Abfragen zur Verfügung.

b) Active Data Guard

Ein besonderer Betriebsmodus der Physical Standby Datenbank ist der Real Time Query Mode. Hier wird die Physical Standby Datenbank zunächst Readonly geöffnet und anschließend wird das Log-Apply gestartet. Somit können Benutzer und Anwendungen lesend auf die Daten zugreifen, die kontinuierlich mit den Änderungen der Primary aktualisiert werden. Dieser Betriebsmode erfordert die Lizenzierung der Active Data Guard Option.

c) Logical Standby

Der grundlegende Unterschied zwischen der Physical und der Logical Standby Datenbank liegt im Apply Mechanismus. Während bei der Physical Standby Datenbank das Apply in einer blockweise identischen Datenbank durch ein Recovery stattfindet, findet bei der Logical Standby Datenbank ein SQL Apply statt. Die Logical Standby Datenbank ist somit nicht mehr blockweise 1:1 identisch. Sie steht während des Apply den Anwendern lesend und unter bestimmten Voraussetzungen auch schreibend zur Verfügung. Für das Apply werden mit der LogMining-Technologie die Inhalte der Redo Logs analysiert und aus diesen Informationen SQL-Statements rekonstruiert. Diese SQL-Statements werden dann in der offenen Datenbank angewandt.
Dadurch kann die Logical Standby Datenbank während des Apply für Reportingaufgaben genutzt werden. Ferner erlaubt das PL/SQL-Paket DBMS_LOGSTDBY eine Manipulation des Apply-Mechanismus (z.B. Definition von Filtern).
Die Logical Standby Datenbank wurde erstmals mit Oracle 9i eingeführt und auch kontinuierlich hinsichtlich Apply-Performance und unterstützter Datentypen erweitert.
Die Physical Standby Datenbank kennt keine Einschränkungen hinsichtlich bestimmter Datentypen, da sie eine blockweise 1:1 Kopie der Produktions-Datenbank ist. Bei der Logical Standby Datenbank gibt es wegen des verwendeten LogMiners je nach Datenbank-Version mehr oder weniger Einschränkungen bei den unterstützten nicht relationalen Datentypen. Objekte, die derartige erweiterten Datentypen enthalten, werden dann bei einem Apply ignoriert, d.h. Änderungen an diesen Tabellen können auf der Logical Standby Datenbank nicht automatisch nachgefahren werden.
Da die Logical Standby Datenbank Grundlage für das Rolling Upgrade mit Data Guard ist, muss im Vorfeld eines solchen Upgrades die Datenbank auf das Vorhandensein nicht unterstützter Objekte untersucht werden.

3. Upgrades in Data Guard Konfiguration

Data Guard hat nur geringe Abhängigkeiten zu den zugrunde liegenden Strukturen wie Betriebssystemversion, Rechner-Architektur, Storage,... So dürfen sich bei Data Guard Umgebungen die Betriebssystemversionen von Primary und Standby Datenbank Rechner unterscheiden (z.B. OL 5 und OL 6), es kann ein RAC-Primary Datenbank-Cluster mit einer Single Instance Standby Datenbank abgesichert werden, es dürfen sich Verzeichnis-Strukturen oder Storagearten (Filesystem auf der einen, ASM auf der anderen Seite) unterscheiden. Bei bestimmten Plattformen ist sogar der Betrieb in gemischten Umgebungen (z.B. Windows – Linux) erlaubt. Dies ist in den Support-Notes „Data Guard Support for Heterogeneous Primary and Physical Standbys in Same Data Guard Configuration (Doc ID 413484.1)“ und „Data Guard Support for Heterogeneous Primary and Logical Standbys in Same Data Guard Configuration (Doc ID 1085687.1)“ beschrieben. Deutlich restriktiver sind die Anforderungen an die verwendete Oracle Datenbank-Software Version innerhalb einer Data Guard Konfiguration. Der Aufbau einer Data Guard Konfiguration erfolgt immer mit identischen Oracle RDBMS Versionen auf beiden Seiten. Stellt der Aufbau in der Regel kein Problem dar, sieht es beim Patchen einer Produktions-Datenbank, an die eine oder mehrere Standby Datenbanken angeschlossen sind, schon ganz anders aus. Bis Oracle 10gR1 wurden keinerlei Unterschiede in einer

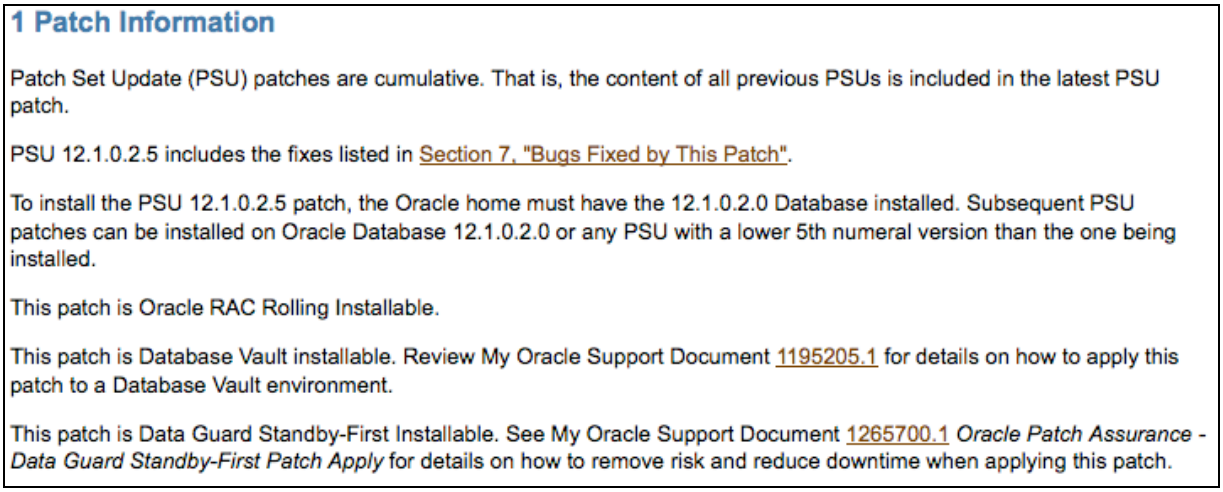
Data Guard Umgebung toleriert. War ein Upgrade oder ein Patchen erforderlich, musste die gesamte Konfiguration runtergefahren werden, die Binaries eingespielt werden und nach dem Neustart beider Seiten der in der Regel erforderliche SQL-Patch in die Produktions-Datenbank eingespielt werden. Beide Seiten waren also immer auf dem gleichen Patch-Level zu halten.

a) Upgrade mit Physical Standby „in place“

Möchte man also in Data Guard Umgebungen das System patchen, muss das Einspielen eines Patches immer auf beiden Seiten stattfinden, bevor das System wieder in Betrieb gehen darf. Bis Oracle 11.2.0.1 in Data Guard Umgebungen mit einer Physical Standby immer mit dem oben beschriebenen Offline-Patchen BEIDER Seiten. Ein Mix-Betrieb ist bis zu dieser Version nicht zulässig. Beginnend mit Oracle 11.2.0.2 sind bestimmte Patches als „Data Guard Standby First Patch“ deklariert. Diese dürfen in einer Data Guard Konfiguration mit einer Physical Standby Datenbank unter bestimmten Voraussetzungen zuerst auf der Standby Datenbank eingespielt werden. Das bedeutet, dass sich dann die Binaries der Datenbank-Software auf beiden Seiten unterscheiden, die Standby-Seite hat dann einen höheren Patch-Stand als die Primary. Während die Produktions-Seite ohne Downtime und ohne Veränderung mit der alten Version arbeitet, kann jetzt auf der Standby-Seite die Wirkungsweise/Auswirkungen des Patches getestet werden. Nach dem Einspielen des Patches auf der Physical Standby Seite kann zunächst beobachtet werden, ob es Probleme beim Einspielen der Redologs gibt. Danach kann die Physical Standby Datenbank entweder readonly geöffnet werden und durch readonly Abfragen getestet werden, oder man wandelt die gepatchte Standby Datenbank in eine Snapshot-Standby um und macht einen read/write-Applikationstest auf diesem System. Nach Abschluss der Tests kann nun auch die Primary-Datenbank gepatched werden. Dies erfordert dann eine Downtime der Primary. Nach dem Patchen der Binaries auf der Primary Seite wird dann ggf. noch das Einspielen des SQL-Patches durchgeführt. Danach ist das Gesamt-System komplett gepatched.
Alternativ kann auch ein Switchover auf die gepatchte Physical Standby gemacht werden, was aus Sicht der Datenbank eine kleine Downtime im Minutenbereich erfordert. Danach wird die neue Primary noch SQL-seitig gepatcht und die alte Primary, jetzige Standby, seitens der Binaries gepatcht.

Der Vorteil des „Data Guard Standby First Patch“ Verfahrens ist also die bessere Test-Möglichkeit von Patches und damit das geringere Risiko nach dem Patchen in unerwartete Fehler zu laufen.
Zu den Patches die mit „Data Guard Standby First Patch“ markiert sind, gehören:
- Einige Database Interim Patches
- Exadata Bundle Patches (monthly/quarterly)
- Datenbank PSUs

Ob ein Patch für das „Data Guard Standby First Patch“ Verfahren geeignet ist, steht im README des jeweiligen Patches.
Der Oktober 2015 Datenbank-PSU ist geeignet:



1 Patch Information

Patch Set Update (PSU) patches are cumulative. That is, the content of all previous PSUs is included in the latest PSU patch.

PSU 12.1.0.2.5 includes the fixes listed in Section 7, "Bugs Fixed by This Patch".

To install the PSU 12.1.0.2.5 patch, the Oracle home must have the 12.1.0.2.0 Database installed. Subsequent PSU patches can be installed on Oracle Database 12.1.0.2.0 or any PSU with a lower 5th numeral version than the one being installed.

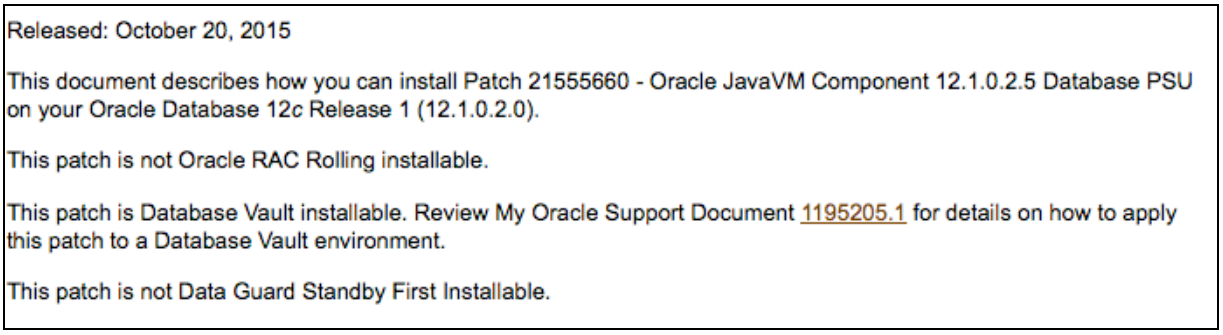
This patch is Oracle RAC Rolling Installable.

This patch is Database Vault installable. Review My Oracle Support Document 1195205.1 for details on how to apply this patch to a Database Vault environment.

This patch is Data Guard Standby-First Installable. See My Oracle Support Document 1265700.1 *Oracle Patch Assurance - Data Guard Standby-First Patch Apply* for details on how to remove risk and reduce downtime when applying this patch.

Abb. 2: Data Guard Standby-First installable PSU 12.1.0.2.5

Das im zugehörigen Combo-Patch enthaltene Java Patch dagegen NICHT:



Released: October 20, 2015

This document describes how you can install Patch 21555660 - Oracle JavaVM Component 12.1.0.2.5 Database PSU on your Oracle Database 12*c* Release 1 (12.1.0.2.0).

This patch is not Oracle RAC Rolling installable.

This patch is Database Vault installable. Review My Oracle Support Document 1195205.1 for details on how to apply this patch to a Database Vault environment.

This patch is not Data Guard Standby First Installable.

Abb. 3: NICHT Data Guard Standby-First installable Java Component Patch

Bei diesem Verfahren müssen folgende Voraussetzungen beachtet werden:
- das Patch Release-Date darf maximal ein Jahr von dem Release-Date der installierten Version abweichen
- Die Dauer des Mischbetriebes darf 31 Tage nicht überschreiten
- Die Mindest-Ausgangs-Version muss entweder Exadata 11.2.0.1 BP7 oder 11.2.0.2.4 sein

Details-Informationen zu „Data Guard Standby-First“ sind in Support-Note „Oracle Patch Assurance - Data Guard Standby-First Patch Apply (Doc ID 1265700.1)“ beschrieben.

Grundsätzlich NICHT geeignet sind Oracle Database Patchsets und Major Releases, das Verfahren kann also bei einem Wechsel von 11.2.0.2 nach 11.2.0.3 oder von 11.2 nach 12.1 nicht eingesetzt werden. Für diese Patch-Situationen kann mit einer Logical Standby Datenbank gearbeitet werden, um zum einen den Patch besser testen zu können und zum anderen die erforderliche Downtime zu minimieren.

b) Rolling Upgrade mit Logical Standby

Das Rolling Upgrade Verfahren mit einer Logical Standby Datenbank wurde erstmalig mit Oracle 10.1.0.3 eingeführt. Bei diesem Verfahren ist an die Primary Datenbank eine Logical Standby Datenbank angebunden. Zu Beginn des Upgrades laufen beide Systeme auf dem gleichen Release-

Stand x. Zunächst wird das eigentliche Upgrade auf der Logical Standby Datenbank durchgeführt. Dazu wird das Log-Shipping und Log Apply unterbrochen und die Standby Datenbank manuell gepatched, d.h. es werden die Binaries eingespielt und das erforderliche Dictionary-Upgrade gemäß Upgrade Guide durchgeführt. Nach Abschluss dieser Phase wird dann das Log-Shipping und Apply wieder aufgenommen und beobachtet. In dieser Phase findet ein Mischbetrieb statt, die Primary-Datenbank arbeitet noch mit Version x und die Standby-Datenbank bereits mit Version y. Bis zu diesem Zeitpunkt können die Anwender unterbrechungsfrei mit dem Primary-System arbeiten. Als letzter zwingender Schritt erfolgt nun das Switchover auf die Standby-Seite. Die Anwendungen arbeiten nun auf einer Primary mit der höheren Version y. Nun wird die alte Primary ebenfalls aktualisiert und optional kann ein erneutes Switchback erfolgen, um zur alten Rollenverteilung zurückzukehren. Um ein Rolling Upgrade durchführen zu können, müssen bei Oracle 11g folgende Voraussetzungen erfüllt sein:
Die Data Guard Konfiguration darf nicht Teil einer Broker-Konfiguration sein
- Der Protection Level darf nur „maximum availability“ oder „maximum performance“ sein
- LOG_ARCHIVE_DEST_n für die Logical standby muss optional sein
- COMPATIBLE muss während des Vorgangs der niedrigeren Version entsprechen

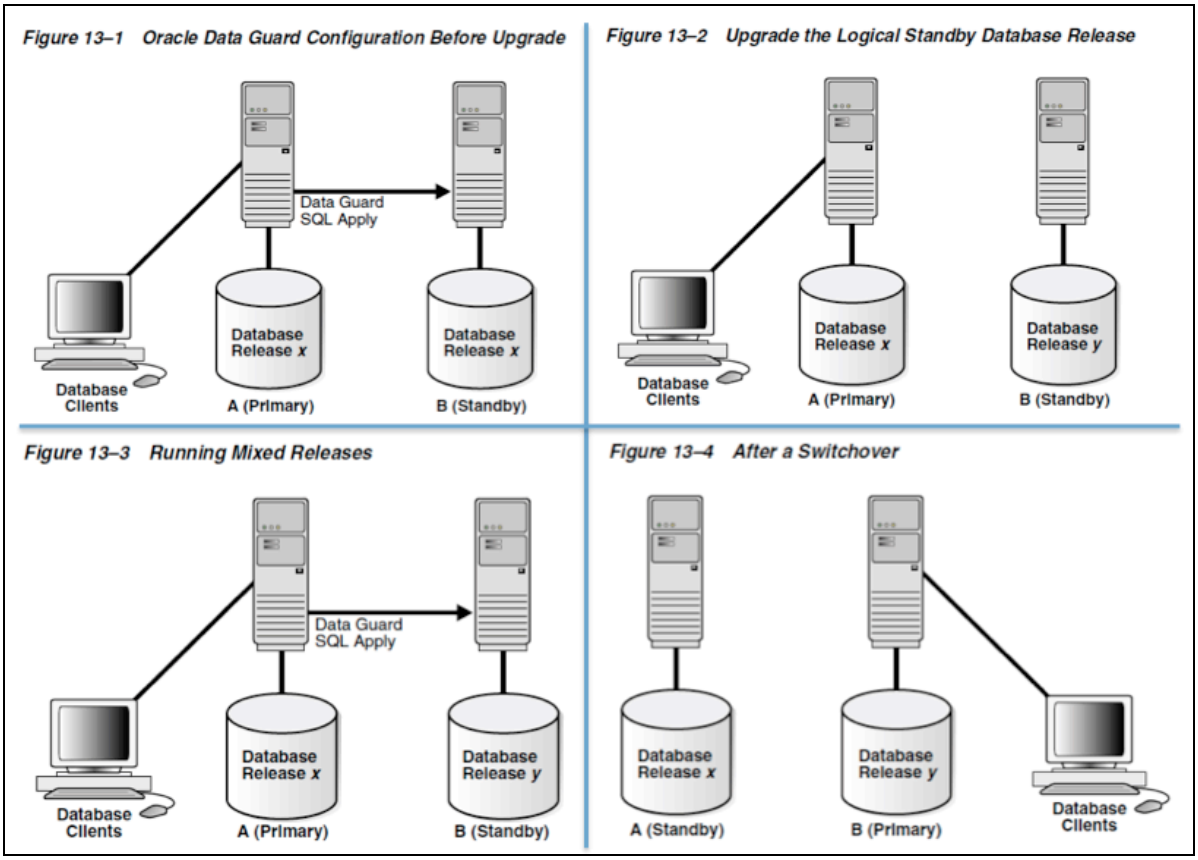


Abb. 4: Rolling Upgrade mit Logical Standby

Das Verfahren ist im Kapitel 13 des „Oracle 12c Data Guard Concepts and Administration Guide“ beschrieben. Die einzelnen Schritte – etwas detaillierter dargestellt – sind:

Step 1 Identify unsupported data types and storage attributes
Step 2 Create a logical standby database

Step 3 Perform a rolling upgrade:
Step 1 Prepare for rolling upgrade
Step 2 Upgrade the logical standby database
Step 3 Restart SQL Apply on the upgraded logical standby database
Step 4 Monitor events on the upgraded standby database
Step 5 Begin a switchover
Step 6 Import any tables that were modified during the upgrade
Step 7 Complete the switchover and activate user applications
Step 8 Upgrade the old primary database
Step 9 Start SQL Apply on the old primary database
Step 10 Optionally, raise the compatibility level on both databases
Step 11 Monitor events on the new logical standby database
Step 12 Optionally, perform another switchover

c) Rolling Upgrade mit Transient Logical Standby

Beim oben beschriebenen „Rolling Upgrade mit Logical Standby“ arbeitet man auschließlich mit einer Logical Standby Datenbank, d.h. man startet mit einer Primary-Logical-Standby Data Guard Umgebung und endet auch damit, unabhängig davon, ob man die Logical Standby Datenbank dauerhaft im Einsatz hat oder nur temporär für das Upgrade aufbaut.
Die meisten Kunden jedoch arbeiten im Data Guard Umfeld wegen des Desasterschutzes mit einer Physical Standby Datenbank. Bei dem Rolling Upgrade Verfahren mit **Transient** Logical Standby wird die Logical Standby nur vorübergehend verwendet. Man startet mit einer Primary – Physical Standby Umgebung und endet auch damit. Die zwischenzeitlich verwendete Logical Standby DB wird nur vorübergehend – transient – verwendet und entsteht aus der bereits vorhandenen Physical Standby Datenbank. Das Verfahren ist ab Version 11.1 dokumentiert.

Die einzelnen Schritte sind im Kapitel 13.6 des DG Concepts Guide beschrieben:

Step 1 Prepare the primary database for a rolling upgrade (perform these steps on Database A)
Step 2 Convert the physical standby database into a logical standby database (perform these steps on Database B)
Step 3 Upgrade the logical standby database and catch up with the primary database (perform these steps on Database B)
Step 4 Flashback Database A to the guaranteed restore point (perform these steps on Database A)
Step 5 Mount Database A using the new version of Oracle software
Step 6 Convert Database A to a physical standby
Step 7 Start managed recovery on Database A
Step 8 Perform a switchover to make Database A the primary database
Step 9 Clean up the guaranteed restore point created in Database A

d) Data Type Support

Bei beiden beschriebenen Verfahren wird für den Zeitraum des eigentlichen Upgrades eine Logical Standby eingesetzt. Ob dieses Verfahren beim eigenen Produktionssystem angewandt werden kann, hängt entscheidend von den verwendeten Daten-Typen ab. Da das Logical Standby Apply auf Basis des LogMiners arbeitet und dieser je nach Datenbankversion nicht alle Datentypen unterstützt, muss zunächst die Produktions-Datenbank dahingehend analysiert werden.
Dazu kann die View DBA_LOGSTDBY_UNSUPPORTED abgefragt werden:

```
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
```

Bei den so gefundenen Tabellen können nun die Spalten identifiziert werden, die nicht supported sind:

```
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
```

Tabellen, die Spalten mit nicht unterstützten Datentypen enthalten, werden vom Apply ausgenommen. D.h. Änderungen, die an diesen Tabellen stattfinden, werden zwar mit dem RedoLog Informationen auf die Standby Datenbank übertragen, die zugehörigen Anweisungen dann aber vom Apply automatisch übersprungen/geskippt. Daher können die Tabellen dann zunächst nicht automatisch aktualisiert werden.

Je nach Datentyp können jedoch mit dem sogenannten Extended Data Type Support (EDS) in der Datenbank Strukturen aufgebaut werden, die es ermöglichen, Änderungen trotzdem auf der Logical Standby Datenbank nachziehen zu lassen.

Die Anzahl der nicht unterstützten Datentypen nimmt mit jedem Release ab. Sie sind im Kapitel C des Data Guard Concepts and Administration Guide aufgelistet.

e) Extended Data Type Support (EDS)

Der Extended Data Type Support (EDS) wurde mit Version 10.2.0.4 erstmalig eingeführt.
EDS für 10.2 und 11.1 ist in der Support-Note „Extended Datatype Support (EDS) for SQL Apply (Doc ID 559353.1)“ näher erläutert.
Ziel des EDS ist es, eine Logical Standby Datenbank auch dann einsetzen zu können, wenn bestimmte nicht nativ unterstützte Datentypen zum Einsatz kommen. Dazu werden passend zu den Datentypen zusätzliche Objekte in der Datenbank angelegt. Bei den zusätzlichen Strukturen handelt es sich um eine Logging Tabelle (message table), einen Base Table Trigger (DML trigger on base table) und einen Logging Table Trigger (DML trigger on message table). Wenn eine Änderung auf der nicht unterstützten Basis-Tabelle stattfindet, zündet der Base Table Trigger und trägt die Änderung in die Logging-Tabelle ein. Dabei wird der nicht unterstützte Datentyp so dargestellt, dass er den supporteten Datentypen entspricht. Die Änderungen an der Basis-Tabelle und in der Logging-Tabelle werden auf die Standby-Datenbank übertragen. Das SQL Apply überspringt die nicht unterstützten Änderungen an der Basis-Tabelle, spielt aber die Änderungen an der Logging-Tabelle ein. Der Logging Table Trigger sorgt nun dafür, dass durch die Änderungen in der Logging Table die passenden Änderungen in der

Basis-Tabelle auf der Standby-Seite stattfinden, die genau den ursprünglichen auf der Basis-Tabelle auf der Primary entsprechen.

Bei 10g und 11gR1 können ab Version 10.2.0.4, bzw 11.1.0.7 zusätzliche Strukturen für den Support von Tabellen mit Objekt-Spalten mit einem User Defined Type, Tabellen mit VARRAYs und Tabellen mit Spatial Type SDO_GEOMETRY angelegt werden.
Dies ist in den Support-Notes
Note.565069.1 EDS for SQL Apply Example - Object Column with Simple User Defined Type
Note.565071.1 EDS for SQL Apply Example - Varray
Note.565074.1 EDS for SQL Apply Example - Oracle Spatial Type SDO_GEOMETRY
beschrieben.
Für 11gR2 wurde der Extended Data Type Support seitens SQL Apply erweitert. Hier können Tabellen mit SDO_GEOMETRY Spalten und Tabellen mit Binary XML Spalten repliziert werden. Siehe dazu auch Support Note: „SQL Apply Extended Datatype Support - 11.2 (Doc ID 949516.1)“

Für das Rolling-Upgrade Verfahren bedeutet dies, dass man auch für Applikations-Tabellen, die man zuvor in der View DBA_LOGSTDBY_UNSUPPORTED identifiziert hat, möglicherweise mit EDS eine Unterstützung für das SQL Apply implementieren kann und somit auch ein Rolling Upgrade mit Logical Standby durchführen kann, ohne Applikationen währenddessen abschalten zu müssen oder Tabelleninhalte hinterher mit Export/Import aktualisieren zu müssen. (Informationen dazu würde man in der View DBA_LOGSTDBY_EVENTS finden)

Zur Ermittlung der Tabellen, die für EDS in Frage kommen, kann ab Oracle 11.2 die View DBA_LOGSTDBY_EDS_SUPPORTED abgefragt werden.
Beginnend mit Oracle 11.2 ist EDS in Form von Paketen in die Datenbank integriert. Hat man Tabellen als EDS-Kandidaten identifiziert, kann man in wenigen Schritten die Replikations-Unterstützung implementieren:

Auf der Primary Datenbank werden mit

```
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE(table_owner =>'OE',
table_name =>'CUSTOMERS');
```

die erforderlichen Trigger und die Logging/Messaging/Shadow-Tabelle aufgebaut.
Auf der Logical Standby-Datenbank wird zunächst das Apply gestoppt:

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

und dann mit

```
SQL> EXECUTE DBMS_LOGSTDBY.EDS_ADD_TABLE(table_owner =>'OE',
table_name =>'CUSTOMERS',p_dblink => 'dblink_to_primary');
```

die Tabellendaten auf der Logical Standby bereitgestellt und die Replikation aktiviert.
Abschließend wird das Apply wieder gestartet.

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

EDS wird mit DBMS_LOGSTDBY.EDS_REMOVE_TABLE deaktiviert.

f) Rolling Upgrade mit DBMS_ROLLING

Das bisherige Rolling Upgrade Verfahren mit einer Logical Standby Datenbank – egal ob transient oder nicht – enthält zahlreiche manuell auszuführende Einzelschritte. Bis Oracle 11g gibt es keine Unterstützung durch den Enterprise Manager, weil die Konfiguration nicht Teil einer Broker-Konfiguration sein darf. Neben der Produkt-Dokumentation gibt es in der Reihe der MAA-Whitepaper das Paper „Database Rolling Upgrade Using Transient Logical Standby: Oracle Data Guard 11g“ das die Schritte detailliert beschreibt. Um das Verfahren zu vereinfachen, wurde ein Bourne-Shell Skript `physru` über den Support zur Verfügung gestellt, das das Verfahren etwas vereinfacht hat. Dieses vereinfachte Verfahren ist im MAA Whitepaper „ Database Rolling Upgrades Made Easy by Using a Data Guard Physical Standby Database“ beschrieben; das Skript kann über die Support-Note „Oracle11g Data Guard: Database Rolling Upgrade Shell Script (Doc ID 949322.1)“ heruntergeladen werden.
Beginnend mit Oracle 12c und in Verbindung mit der Active Data Guard Option kann nun zur Vereinfachung der vielen Einzelschritte das neue PL/SQL-Paket DBMS_ROLLING für ein Upgrade von 12.1 aufwärts eingesetzt werden.
Das Paket DBMS_ROLLING kommt dabei nicht nur für Rolling Upgrades in Frage, sondern kann auch in folgenden Situationen eingesetzt werden:

- Adding partitioning to non-partitioned tables
- Changing BasicFiles LOBs to SecureFiles LOBs
- Changing XMLType stored as CLOB to XMLtype stored as binary XML
- Altering tables to be OLTP-compressed

Im Zusammenhang mit einem Rolling Upgrade vereinfachen sich die einzelnen Schritte und die Fehlermöglichkeiten reduzieren sich. Das ganze Rahmen-Verfahren wurde auf drei Schritte reduziert, zwischen denen dann das eigentliche Upgrade durchgeführt wird.
Die beteiligten Datenbanken werden in eine Leading Group (LG) und eine Trailing Group (TG) eingeteilt. Sind mehr als zwei Datenbanken beteiligt, teilt man die Datenbanken in zwei Gruppen auf. In der Leading Group gibt es dann einen Leading Group Master (LGM) und in der Trailing Group einen Trailing Group Master (TGM). Sollen diese Datenbanken mit einer Standby abgesichert werden, so werden diese Standby Datenbanken dann Leading Group Standbys (LGS) bzw Trailing Group Standbys (TGS) genannt. Die Datenbank, die zuerst aktualisiert wird, wird LGM genannt, die Datenbank, die anschließend aktualisiert wird, also die alte Primary Datenbank, wird TGM genannt. Während die Datenbanken der Leading Group aktualisiert werden, können die Anwender unterbrechungsfrei mit den Datenbanken der Trailing Group arbeiten. Die Datenbanken der Trailing Group arbeiten so lange noch mit der alten Datenbank-Version bis alle Datenbanken der Leading Group aktualisiert worden sind und alle zwischenzeitlichen Änderung der Primary Datenbank eingespielt wurden. Erst dann kommt es zum Switchover (und damit zu einer kurzen Downtime für die Applikationen). Im abschließenden Schritt werden die Datenbanken der Trailing Group aktualisiert und als Standby-Datenbanken zugefügt, während die Benutzer bereits weiter arbeiten können.
Die Verwendung des DBMS_ROLLING Paketes steigert die Robustheit des Vorgangs und ermöglicht die Data Guard Absicherung der beteiligten Datenbanken um jederzeit Zero-Data Loss garantieren zu können.

Das Rolling-Upgrade Verfahren ist im Wesentlichen in die Phase „Planning a Rolling Upgrade“ und „Performing a Rolling Upgrade“, also eine reine Planungs- und eine reine Ausführungs-Phase, unterteilt. Diese beiden Phasen erfordern jeweils eine Reihe von einzelnen Schritten, die der reihe nach abgearbeitet werden. Die einzelnen Phasen des Rolling Upgrades werden anhand des Beispiels der Produkt-Dokumentation im Folgenden näher erläutert.

Beschreibung Demo-Beispiel

Im Kapitel 14 des „Oracle 12c Data Guard Concepts and Administration Guide“ ist das Rolling Upgrade-Verfahren mit dem neuen PL/SQL Paket DBMS_ROLLING detailliert anhand eines Beispiels beschrieben. Die Ausgangs-Situation in diesem Beispiel ist eine Produktions-Datenbank, an die drei Standby Datenbanken angeschlossen sind. Die DB_UNIQUE_NAMEs der beteiligten Datenbanken sind `seattle`, `boston`, `oakland` und `atlanta`. `seattle` ist die Primary Datenbank, die übrigen sind Physical Standby Datenbanken. Im Verlauf des Rolling Upgrades soll die Physical Standby `boston` die Rolle der Transient Logical Standby übernehmen. Damit zu jedem Zeitpunkt auf allen Seiten ein Desasterschutz gewährleistet ist, soll die Primary `seattle` durch die Physical Standby `oakland` und die zukünftige Primary `boston` durch die Standby Datenbank `atlanta` abgesichert werden.
Die folgende Abbildung zeigt die Start-Situation vor Anwendung der DBMS_ROLLING Prozeduren.

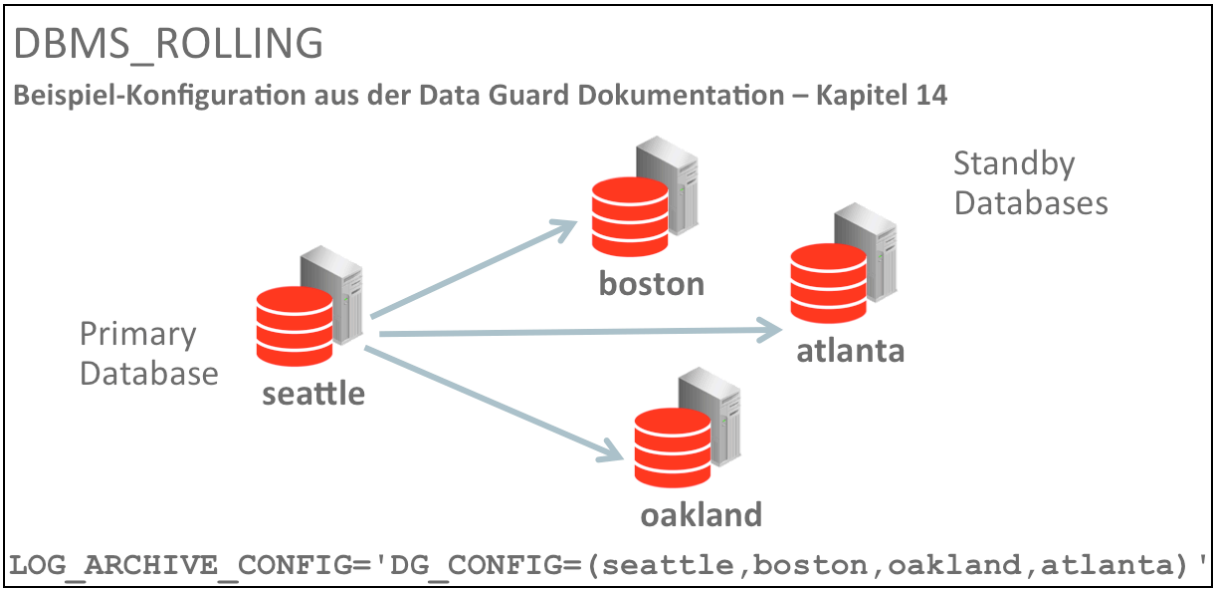


Abb. 5: Demo-Beispiel für DBMS_ROLLING

Planning a Rolling Upgrade-Phase
Im ersten Schritt der Planungs-Phase wird der eigentliche Upgrade-Plan initialisiert. Dies geschieht durch den Aufruf

```
SQL> EXECUTE DBMS_ROLLING.INIT_PLAN (future_primary => 'boston');
```

Hiermit wird festgelegt, dass die Physical Standby Datenbank `boston` im Verlauf des Upgrades in eine Transient Logical Standby Datenbank umgewandelt wird und somit gemäß obiger Notation zum LGM (Leading Group Master) wird.
Nach Abschluss dieses Schritts hat man in der LG (leading group) nur die Datenbank `boston` und die TG (trailing group) besteht aus den drei Datenbanken `seattle`, `oakland` und `atlanta`.

Dies wird auch in der View DBA_ROLLING_PARAMETERS entsprechend angezeigt:

SQL> select scope, name, curval from dba_rolling_parameters order by scope, name;

SCOPE	NAME	CURVAL
seattle	INVOLVEMENT	FULL
seattle	MEMBER	NONE
atlanta	INVOLVEMENT	FULL
atlanta	MEMBER	TRAILING
oakland	INVOLVEMENT	FULL
oakland	MEMBER	TRAILING
boston	INVOLVEMENT	FULL
boston	MEMBER	LEADING
	ACTIVE_SESSIONS_TIMEOUT	3600
	ACTIVE_SESSIONS_WAIT	0

Um einen vollständigen Desasterschutz über alle Phasen gewährleisten zu können, werden die Parameter des Plans angepasst. Die Physical Standby Datenbank `atlanta` wird der LG zugewiesen:

SQL> EXECUTE DBMS_ROLLING.SET_PARAMETER(scope=>'atlanta',name=>'member‘, value=>'leading');

Nach Abschluss dieser und eventueller weiterer Parameteranpassungen ergibt sich folgendes Bild der Umgebung:

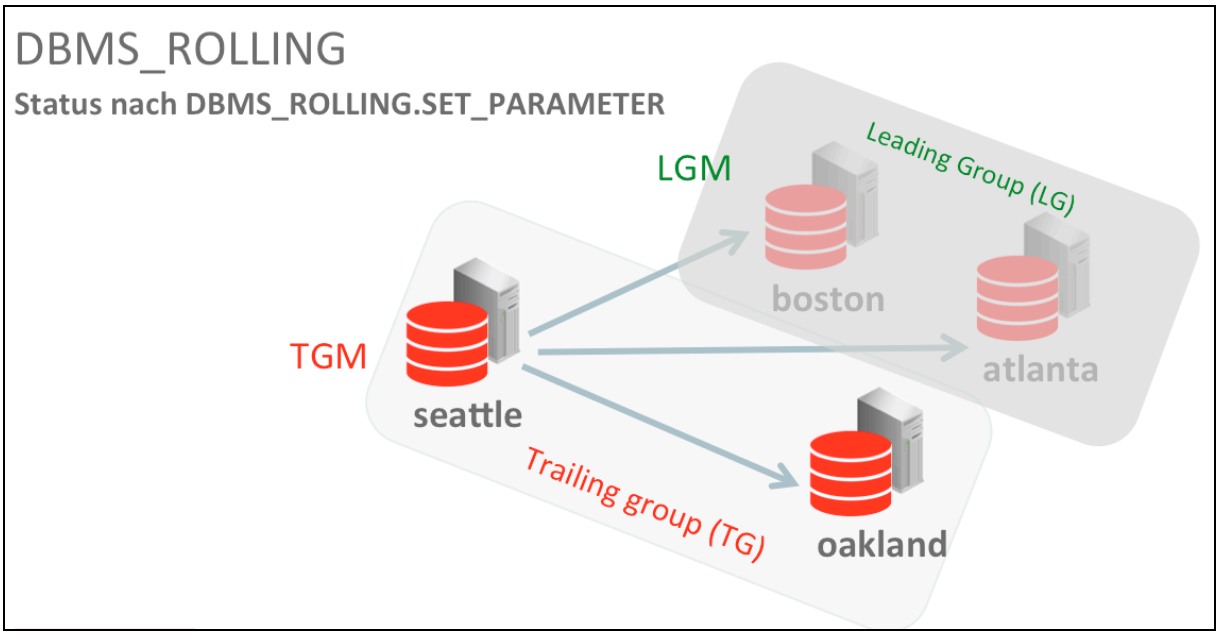


Abb. 6: Status nach DBMS_ROLLING.SET_PARAMETER

Bis zu diesem Zeitpunkt handelt es sich um eine reine Definition des Prozesses. Die Anwender arbeiten ohne Unterbrechung mit der Primary Datenbank und auch die Physical Standby Datenbanken werden nach wie vor von der Primary mit RedoLog Informationen versorgt.

Den Abschluss der Planungs-Phase bildet das Generieren des Ablauf-Plans mit

SQL> EXECUTE DBMS_ROLLING.BUILD_PLAN;

Dabei erfolgt eine Validierung der Einstellungen und bei Abweichungen von Best Practises Einstellungen werden in der View DBA_ROLLING_EVENTS entsprechenden Warnungen ausgewiesen.
Die Schritte, die dann beim Rolling Upgrade durchlaufen werden, können in der View DBA_ROLLING_PLAN abgefragt werden.

SQL> SELECT instid, target, phase, description FROM DBA_ROLLING_PLAN;

Damit ist die Planungs-Phase abgeschlossen.

Performing a Rolling Upgrade-Phase

In der Durchführungs-Phase werden nun mit Hilfe der Prozeduren des DBMS_ROLLING Paketes die einzelnen Schritte ausgeführt. In dieser Phase müssen auch die eigentlichen Software Upgrade Schritte durchgeführt werden. Dies geschieht manuell entweder durch den DBUA oder durch Aufruf der zugehörigen SQL-Skripte.
Der erste Schritt der Umsetzungs-Phase wird durch das Starten des Plans ausgelöst und ist der formelle Start des Rolling Upgrades, hier werden dann die notwendigen Änderungen an den beteiligten Datenbanken vorgenommen. Im Einzelnen werden folgende Schritte **automatisch** durchlaufen:

- Konfiguration von Transient Logical Standby und ggf. diese schützende Physical Standby Datenbanken
- Backup Control File to trace
- Anlegen von Guaranteed Restore Points
- Aufbau LogMiner Dictionary
- Erstellung Transient Logical Standby Database
- Konfiguration von LGS Datenbanken

Der zugehörige Befehl ist:

SQL> EXECUTE DBMS_ROLLING.START_PLAN

Nach Abschluss dieses ersten automatisierten Schrittes muss nun die Oracle Datenbank Software bei allen Datenbanken der TG aktualisiert werden und das Dictionary Upgrade der Transient Logical Standby Datenbank durchgeführt werden. Dies erfolgt manuell mit dem DBUA oder durch Aufruf der im Upgrade Guide beschriebenen SQL-Skripte.

Ist dieser zweite Schritt abgeschlossen, besteht die Gesamt-Konfiguration im vorliegenden Beispiel aus der Primary Datenbank `seattle`, die durch die Standby Datenbank `oakland` abgesichert ist. Beide Datenbanken laufen unverändert auf der „alten“ Datenbank Version x. Die Primary Datenbank schickt ihre Redolog Informationen außerdem an die Transient Logical Standby Datenbank `boston`, die ihrerseits durch die Physical Standby Datenbank `atlanta` abgesichert ist. `boston` und `atlanta` arbeiten bereits mit dem höheren Release-Stand x+n.

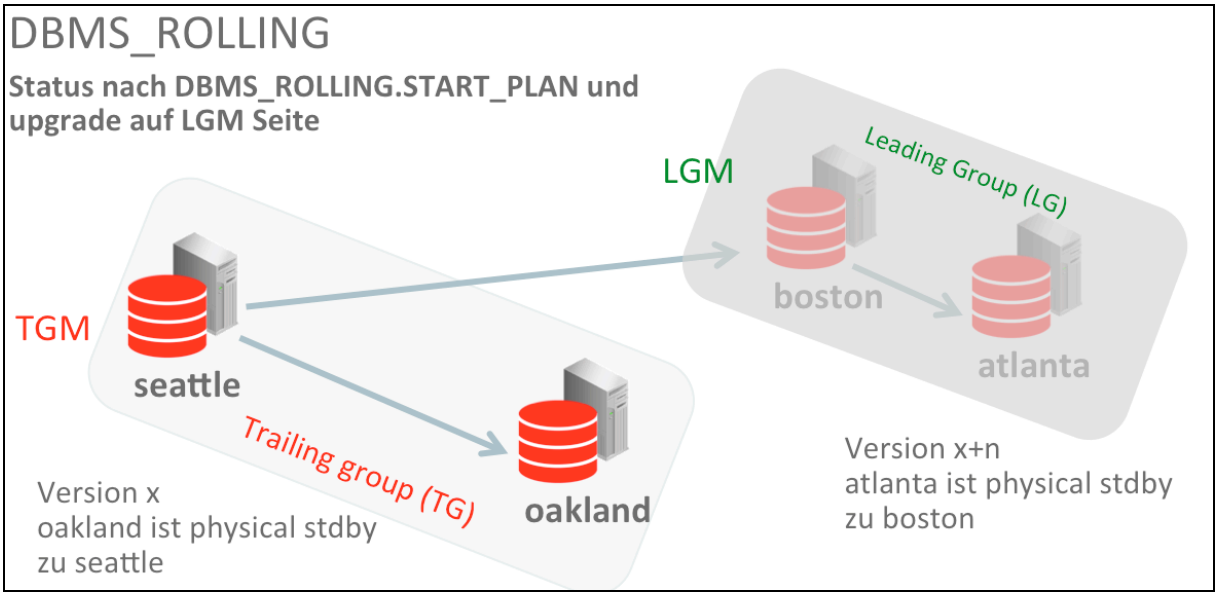


Abb. 7: Status nach DBMS_ROLLING.START_PLAN

Der dritte Schritt erfolgt nun wieder unterstützt durch das DBMS_ROLLING Paket. In diesem Schritt wird das eigentliche Switchover durchgeführt, bei dem es zu einer kurzen Auszeit für die Applikationen kommt. Aus Sicht der Datenbank ist das Switchover im Minutenbereich durchgeführt. Die bisherige Transient Logical Standby Datenbank `boston` wird zur neuen Primary Datenbank, die dank der vorherigen Konfiguration, auch bereits mit der Physical Standby Datenbank `atlanta` abgesichert ist.
Der Befehl für das automatisierte Switchover ist:

```
SQL> EXECUTE DBMS_ROLLING.SWITCHOVER;
```

Nach diesem Schritt müssen jetzt im vierten Schritt die verbleibenden Datenbanken der TG, also die alte Primary `seattle` und ihre Physical Standby Datenbank `oakland` mit der neuen Software manuell neu gestartet werden, nachdem diese installiert wurde.

Der abschließende fünfte Schritt beendet die Umsetzungs-Phase. In diesem Schritt werden folgende Aktionen automatisiert durchgeführt:

- Flashback von ehemaliger Primary und den TGS Standby Datenbanken
- Umwandlung alter Primary in Physical Standby
- Konfiguration TGS für Recovery von Upgrade Redo
- Einspielen des Upgrade Redos

Der Befehl für diesen letzten Schritt ist:

```
SQL> EXECUTE DBMS_ROLLING.FINISH_PLAN;
```

Damit ist das Rolling Upgrade abgeschlossen und man hat folgende Abschluss-Konfiguration:

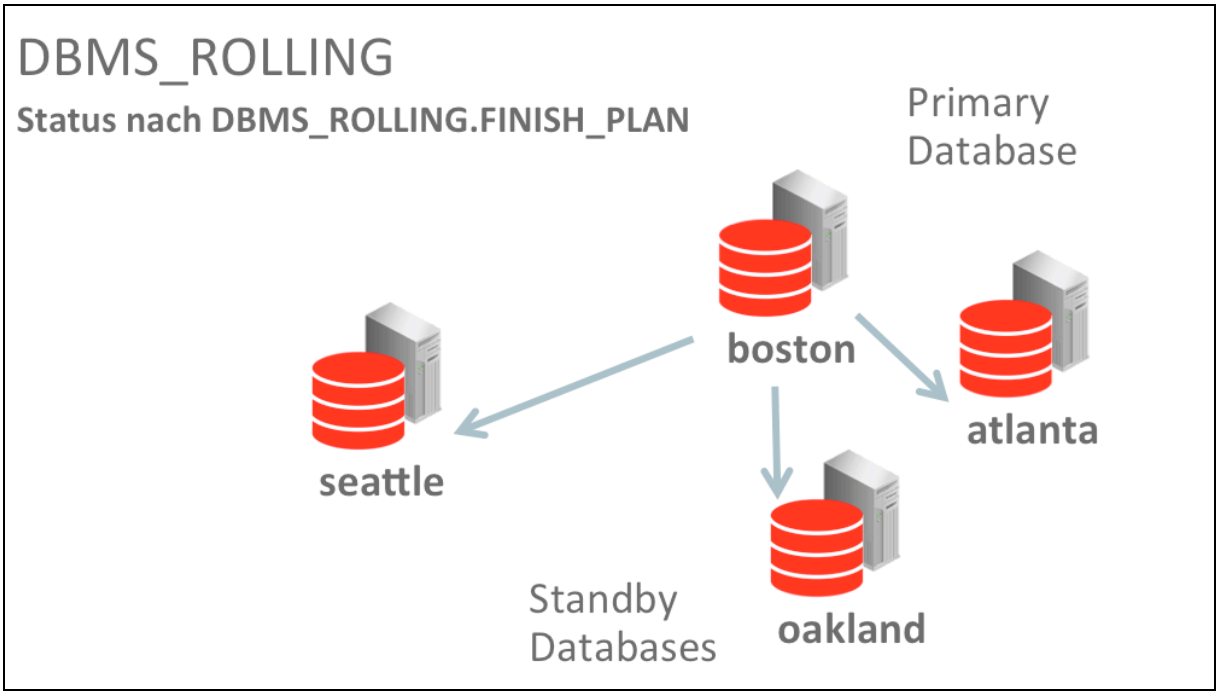


Abb. 8: Status nach DBMS_ROLLING.FINISH_PLAN

Monitoring des Rolling Upgrades

Für das Monitoring des Rolling Upgrades mit DBMS_ROLLING stehen folgende Views zur Verfügung:

DBA_ROLLING_STATUS: zeigt Information über den Overall Status des Upgrade
DBA_ROLLING_DATABASES: gibt Information über die Rolle, Protection Level und Recovery Status jeder am Rolling Upgrade involvierten Datenbank
DBA_ROLLING_STATISTICS: listet Statistiken wie z.B. Start- und Ende-Zeiten, Offline-Zeiten von Services,... auf

4. Fazit und Zusammenfassung

Betrachtet man die Komplexität eines manuell durchzuführenden Rolling Upgrades mit Transient Logical Standby, bei der eine Vielzahl von Einzelschritten durchgeführt werden muss, die ihrerseits sehr konzentriert durchgeführt werden müssen, dann stellt das Verfahren mit dem DBMS_ROLLING Paket eine große Erleichterung dar. Das Verfahren kann im Wesentlichen auf wenige Ausführungs-Schritte reduziert werden, egal wie komplex die Umgebung ist.

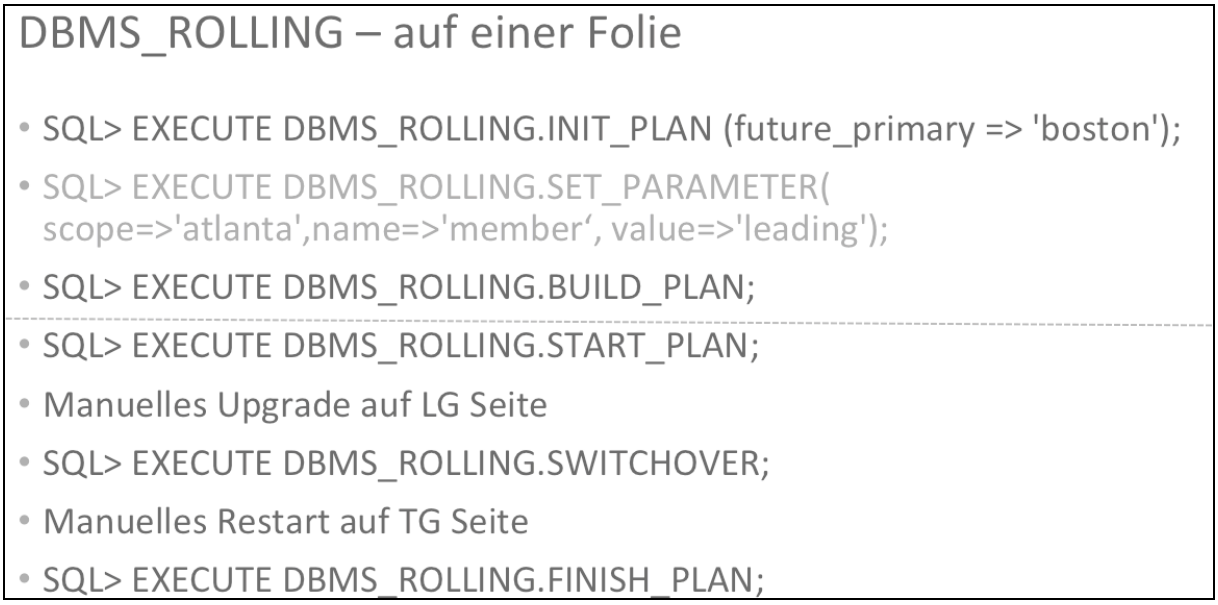


Abb. 9: DBMS_ROLLING alle Schritte auf einen Blick

Die Anpassung der Parameter mit DBMS_ROLLING.SET_PARAMETER ist nur bei Bedarf erforderlich, z.B. wenn beim Vorhandensein mehrerer Standby-Datenbanken eine Aufteilung auf die Gruppen LG und TG erfolgen soll. Defaultmäßig landen alle Standby Datenbanken in der TG.

Das Verfahren mit dem DBMS_ROLLING Paket steht jedoch erst ab Versionswechseln von 12.1 aufwärts zur Verfügung. Um von früheren Releases auf 12c zu wechseln, kann das traditionelle Rolling Upgrade mit Transient Logical Standby zum Einsatz kommen.

Kontaktadresse:
Claudia Hüffer
Oracle Deutschland B.V. & Co. KG
Kühnehöfe 5
D-22761 Hamburg

Telefon: +49 (0) 40-89091 135
Fax: +49 (0) 40-89091 250
E-Mail claudia.hueffer@oracle.com
Internet: www.oracle.com