



Hardware and Software
Engineered to Work Together



Oracle Database 12c: RAC and Data Guard Integration Workshop

Student Guide

D94430GC10

Edition 1.0 | May 2016 | D95960

Learn more from Oracle University at oracle.com/education/

Author

Sean Kim

**Technical Contributors
and Reviewers**

Jerry Lee
Jim Womack
Peter Fusek

Editors

Chandrika Kennedy
Aju Kumar
Aishwarya Menon

Graphic Designer

Kavya Bellur

Publishers

Asief Baig
Veena Narasimhan

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Introduction to High Availability Best Practices

- Objectives 1-2
- Causes of Unplanned Down Time 1-3
- Causes of Planned Down Time 1-4
- Oracle's Solution to Down Time 1-5
- Review of Oracle Clusterware 1-7
- Clusterware Services 1-8
- Deployment Option: Standard Cluster 1-9
- Deployment Option: Flex Cluster 1-10
- Review of Oracle ASM 1-11
- ASM Features and Benefits 1-12
- Deployment Option: Standard ASM 1-13
- Deployment Option: Flex ASM 1-14
- Flex ASM and Flex Clusters 1-15
- Review of Oracle RAC 1-16
- Typical Oracle RAC Architecture 1-17
- Review of Oracle Data Guard 1-18
- Types of Standby Databases 1-19
- RAC and Data Guard Complementarity 1-21
- Deployment Option: Maximum Availability Architecture 1-22
- Summary 1-23
- Practice 1: Overview 1-24

2 Converting a Single Instance to an Oracle RAC Database

- Objectives 2-2
- Single Instance-to-RAC Conversion 2-3
- Considerations for Converting Single-Instance Databases to Oracle RAC 2-4
- Scenario 1: Using DBCA 2-5
- Step 1: Create an Image of the Single-Instance Database 2-6
- Example: Result of Step 1 2-7
- Step 2: Create an Oracle Cluster for RAC 2-8
- Example: Result of Step 2 2-9
- Step 3: Copy the Preconfigured Database Image 2-10
- Example: Database Structure File (*.dbc) 2-11
- Example: Result of Step 3 2-12

Step 4: Create an Oracle RAC Database	2-13
Scenario 2: Using rconfig	2-14
Step 1: Check the Database Type	2-15
Step 2: Modify the XML File for the rconfig Utility	2-16
Example: ConvertToRAC_AdminManaged.xml	2-17
Step 3: Perform Prerequisite Checks	2-18
Step 4: Convert to an Oracle RAC Database	2-19
Step 5: Verify the Conversion	2-21
Example: Result of Using rconfig	2-22
Quiz	2-23
Summary	2-24
Practice 2: Overview	2-25

3 Configuring Oracle Net Services in a Data Guard Environment with RAC

Objectives	3-2
Review of Oracle Net Services	3-3
Oracle Net Configuration Files	3-4
Single Client Access Name in RAC	3-5
Example: Client Database Connections in RAC	3-7
Configuring Oracle Net Services in a Data Guard Environment	3-8
Primary and Standby Databases with RAC	3-9
Example: tnsnames.ora on Primary Hosts	3-10
Example: tnsnames.ora on Standby Hosts	3-11
Review of Dynamic Service Registration	3-12
Example: listener.ora on Primary Hosts	3-13
Example: listener.ora on Standby Hosts	3-14
Starting and Stopping the Node Listener	3-16
Quiz	3-17
Summary	3-18
Practice 3: Overview	3-19

4 Deploying a Physical Standby Database in an Oracle RAC Environment

Objectives	4-2
Creating a Physical Standby Database in RAC	4-3
Task 1: ARCHIVELOG Mode in RAC	4-4
Task 1: FORCE LOGGING Mode	4-5
Task 1: Creating a Password File	4-6
Task 1: Configuring Standby Redo Logs	4-7
Example: Configuring Standby Redo Logs for RAC	4-8
Task 1: Setting Initialization Parameters	4-9

Setting Initialization Parameters on the Primary Database to Control Redo	
Transport	4-10
Setting LOG_ARCHIVE_CONFIG	4-11
Setting LOG_ARCHIVE_DEST_n	4-12
Example: LOG_ARCHIVE_DEST_n	4-13
Setting Initialization Parameters on the Primary Database	4-14
Specifying Values for DB_FILE_NAME_CONVERT	4-15
Specifying Values for LOG_FILE_NAME_CONVERT	4-16
Specifying a Value for STANDBY_FILE_MANAGEMENT	4-17
Specifying a Value for FAL_SERVER	4-18
Example: FAL_SERVER for RAC	4-19
Example: Setting Initialization Parameters on the Primary Database	4-20
Creating a Physical Standby Database in RAC	4-21
Task 2: Configuring Oracle Net Services	4-22
Task 2: Configuring a Temporary Oracle Net Service for Your Standby	
Database	4-23
Creating a Physical Standby Database in RAC	4-24
Task 3: Copying the Password File to the Standby Hosts	4-25
Task 3: Creating Directories for the Physical Standby Database	4-26
Creating a Physical Standby Database in RAC	4-27
Task 4: Creating a Temporary Initialization Parameter File	4-28
Task 4: Starting the Physical Standby Database	4-29
Creating a Physical Standby Database in RAC	4-30
Step 5: Creating an RMAN Script	4-31
Example: RMAN Script 1	4-32
Example: RMAN Script 2	4-33
Example: RMAN Script 3	4-34
Step 5: Creating the Physical Standby Database	4-35
Creating a Physical Standby Database in RAC	4-36
Step 6: Creating an Spfile on the Standby Database	4-37
Step 6: Registering the RAC Standby Database with CRS	4-38
Step 6: Verify the RAC Standby Database	4-39
Creating a Physical Standby Database in RAC	4-40
Step 7: Starting Redo Apply in Real Time in RAC	4-41
Quiz	4-42
Summary	4-43
Practice 4: Overview	4-44

5 Configuring Oracle Data Guard in an Oracle RAC Environment

Objectives	5-2
Configuration Considerations in RAC	5-3

Data Guard Broker: Management Model	5-4
Data Guard Broker: Requirements	5-5
Clearing Redo Transport Network Locations on Primary Database	5-7
Separation of DBA Duties: SYSDG	5-8
Connecting to the Primary Database with DGMGRL	5-9
Shared Password File in a Disk Group	5-10
Example: Creating a Shared Password File	5-11
Data Guard Broker Configuration Files for RAC	5-12
Creating a Broker Configuration	5-13
Verifying the Configuration	5-14
Configuration Considerations in RAC	5-15
Redo Transport Service in RAC	5-16
Example: Redo Transport Service in RAC	5-17
Redo Apply Service in RAC	5-18
Example: Redo Apply Service in RAC	5-19
Switching Apply Instance in RAC	5-20
Example: Switching Apply Instance in RAC	5-21
Automatic Apply Instance Failover	5-22
Configuration Considerations in RAC	5-24
Data Guard Deployment Options	5-25
Behavior of the Data Protection Mode in RAC	5-26
Behavior of the Maximum Protection Configuration in RAC	5-27
Configuration Considerations in RAC	5-29
Performing a Switchover in RAC	5-30
Example: Validating Databases for Switchover	5-31
Example: Switchover by Using DGMGRL	5-32
Performing a Failover in RAC	5-33
Configuration Considerations in RAC	5-34
Using Flashback Database	5-35
Configuring Flashback Database	5-36
Configuration Considerations in RAC	5-37
Fast-Start Failover in RAC	5-38
When Does Fast-Start Failover Occur?	5-39
Fast-Start Failover Prerequisites	5-40
Configuring Fast-Start Failover	5-41
Avoiding a False Failover in RAC	5-42
Example: Fast-Start Failover	5-43
Summary	5-44
Practice 5: Overview	5-45

6 Managing Physical Standby Files After Structural Changes on the Primary Database

Objectives	6-2
Scenario 1: Creating a Tablespace	6-3
Action Required on Physical Standby	6-4
Scenario 2: Using a Transportable Tablespace	6-5
Action Required on Physical Standby	6-6
Review: Online Move Data File in 12c	6-8
Scenario 3: Moving an Online Data File	6-9
Action Required on Physical Standby	6-10
Scenario 4: Adding or Dropping a Redo File Group	6-12
Action Required on Physical Standby	6-13
Scenario 5: NOLOGGING Operations	6-15
Action Required on Physical Standby	6-16
Scenario 6: Refreshing the Password File	6-20
Action Required on Physical Standby	6-21
Review: Transparent Data Encryption	6-22
Scenario 7: Resetting the TDE Master Encryption Key	6-23
Summary	6-24
Practice 6: Overview	6-25

7 Effective Client Failover Using Application Continuity

Objectives	7-2
Understanding Client Connectivity in a Data Guard Configuration	7-3
Effective Client Failover: Overview	7-4
Client Failover Considerations	7-5
Connecting to the Appropriate Environment	7-6
Managing Database Services	7-7
Example: Creating Services for Data Guard	7-8
Example: Using AFTER STARTUP Trigger	7-9
Example: Configuring Services for JDBC Clients	7-10
Example: Configuring JDBC URL	7-11
Client Failover Considerations	7-12
Fast Application Notification (FAN)	7-13
Client Failover Considerations	7-14
Implementing Client Failover with FAN	7-15
Example: Fast Connection Failover (UCP JDBC)	7-16
Example: Transparent Application Failover	7-17
Example: TAF Basic Configuration with FAN	7-18
Example: tnsnames.ora for OCI Applications	7-20
Client Failover Considerations	7-21

Introducing Transaction Guard	7-22
How Transaction Guard Works	7-23
Using Transaction Guard	7-24
Example: Creating Services for Transaction Guard	7-25
Benefits of Transaction Guard	7-26
Client Failover Considerations	7-27
Introducing Application Continuity	7-28
How Does Application Continuity Work?	7-29
Using Application Continuity	7-30
Configuring the JDBC Replay Data Source	7-31
Creating Services for Application Continuity	7-32
Benefits of Application Continuity	7-33
RAC and Application Continuity	7-34
Data Guard and Application Continuity	7-35
Quiz	7-36
Summary	7-37
Practice 7 Overview: Using Application Continuity	7-38

8 Effective Service Failover and Workload Management Using Global Data Services

Objectives	8-2
Lesson Agenda	8-3
Review of Effective Client Failover	8-4
Review of Workload Management in RAC	8-5
Connection Time Load Balancing in RAC: Example	8-6
Runtime Connection Load Balancing (UCP JDBC/ODP.NET) in RAC: Example	8-7
Lesson Agenda	8-8
Global Data Services	8-9
Global Data Services Capabilities	8-10
Global Data Services Architecture	8-11
Lesson Agenda	8-12
Global Service: Overview	8-13
Global Services in a RAC Database	8-14
Global Services in a Data Guard Broker Configuration	8-15
Lesson Agenda	8-17
Global Service Attributes	8-18
preferred, available, and preferred_all	8-19
Role-Based Services	8-20
Replication Lag	8-21
Lesson Agenda	8-22
Global Connection Load Balancing: Overview	8-23

Client-Side Load Balancing	8-24
Global Connection Time Load Balancing	8-25
Global Connection Time Load Balancing: Example	8-26
Global Run Time Load Balancing	8-27
Global Runtime Connection Load Balancing: Example	8-28
Locality-Based Routing	8-29
Lesson Agenda	8-30
Client Connectivity in GDS	8-31
Application Continuity and GDS with Active Data Guard	8-32
Configuring Global Service for Application Continuity: Example	8-33
GDS Deployment: Overview	8-34
Quiz	8-35
Summary	8-36
Practice 8: Overview	8-37

9 Performing Database Recovery in an Oracle Data Guard Environment

Objectives	9-2
Lesson Agenda	9-3
Performing Database Recovery	9-4
Loss of a Temp File	9-5
Loss of All Control Files	9-6
Loss of a Redo Log Group	9-7
Loss of Data Files	9-9
Tablespace Point-in-Time Recovery	9-10
Flashback Database	9-12
Flashback Table	9-13
Recovering Tables from Backups in 12c	9-14
Lesson Agenda	9-15
Recovery Considerations in Oracle Data Guard	9-16
Scenario 1: Complete Recovery	9-17
Scenario 2: Incomplete Recovery	9-18
Scenario 3: Standby Control Files Recovery	9-20
Scenario 4: Tablespace Point-In-Time Recovery	9-21
Recovery Considerations in Oracle Data Guard	9-22
Incomplete Recovery of the Primary Database: Overview	9-23
Scenario 5: Using Flashback Database	9-24
Scenario 6: Using Standby Database	9-25
Scenario 7: Using Primary Database	9-27
Summary	9-28
Practice 9: Overview	9-29

10 Performing Data Guard Standby-First Patch Apply

Objectives	10-2
Background: Data Guard Support for Heterogeneous Configuration	10-3
Data Guard Standby-First Patch: Overview	10-5
Data Guard Standby-First Patch Installable	10-6
README File: Example	10-7
Data Guard Standby-First Patch Apply: Phases	10-8
Phase 1: Patch 1st Standby Host	10-9
Phase 1: Patch 2nd Standby Host	10-10
Phase 2: Evaluate Patch on Standby Database	10-11
Phase 3: Complete Patch Installation or Rollback	10-12
Phase 3: Data Guard Switchover and Apply Patch to New Physical Standby (Option 2)	10-13
Benefits	10-14
Summary	10-15
Practice 10: Overview	10-16

11 Disassociating a Snapshot Standby Database from a Data Guard Configuration

Objectives	11-2
Snapshot Standby Databases: Overview	11-3
Converting a Physical Standby Database into a Snapshot Standby Database	11-4
View Snapshot Standby Database Information	11-5
Snapshot Standby Space Requirements	11-6
Considerations for Lengthy Testing	11-7
Disassociating a Snapshot Standby from a Data Guard Configuration	11-8
Example: Initial Environment	11-9
Steps for Disassociating a Snapshot Standby from a Data Guard Configuration	11-10
Step 1: View Snapshot Standby Information	11-11
Step 2: Drop a Guaranteed Restore Point	11-12
Example: Result of Step 2	11-13
Step 3: Remove Snapshot Standby	11-14
Step 4: Use the DBNEWID Utility	11-15
Example: Output of the DBNEWID Utility	11-16
Example: Result of Step 4	11-17
Step 5: Modify the Initialization Parameter File	11-18
Step 6: Create the Password File and Change the Global Database Name	11-19
Step 7: Open the Database with RESETLOGS	11-20
Example: Result of Step 7	11-21
Summary	11-22
Practice 11: Overview	11-23

12 Rolling Database Upgrade Using Transient Logical Standby

Objectives 12-2

Rolling Database Upgrade with Logical Standby Database 12-3

Phases for Rolling Database Upgrade with Transient Logical Standby

Database 12-4

Example: Initial Environment 12-5

Phases for Rolling Database Upgrade with Transient Logical Standby

Database 12-6

Phase 1: Completing Prerequisites 12-7

Example: Result of Phase 1 12-8

Phase 2: Preparing for Upgrade 12-9

Example: Result of Phase 2 12-10

Phase 3: Performing Pre-Upgrade Tasks 12-11

Example: Result of Phase 3 12-13

Phases for Rolling Database Upgrade with Transient Logical Standby

Database 12-14

Phase 4: Upgrading the Transient Logical Standby 12-15

Example: Result of Phase 4 12-17

Phases for Rolling Database Upgrade with Transient Logical Standby

Database 12-18

Phase 5: Performing Post-Upgrade Tasks 12-19

Example: Result of Phase 5 12-21

Phase 6: Preparing the Original Primary Database for Upgrade 12-22

Example: Result of Phase 6 12-23

Phase 7: Performing the Final Upgrade Tasks 12-24

Example: Result of Phase 7 (Before Switchover) 12-25

Example: Result of Phase 7 (After Switchover) 12-26

Benefits and Challenges 12-27

Simplified Method 1 12-29

Simplified Method 2 12-30

Summary 12-31

Practice 12: Overview 12-32

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

1

Introduction to High Availability Best Practices

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

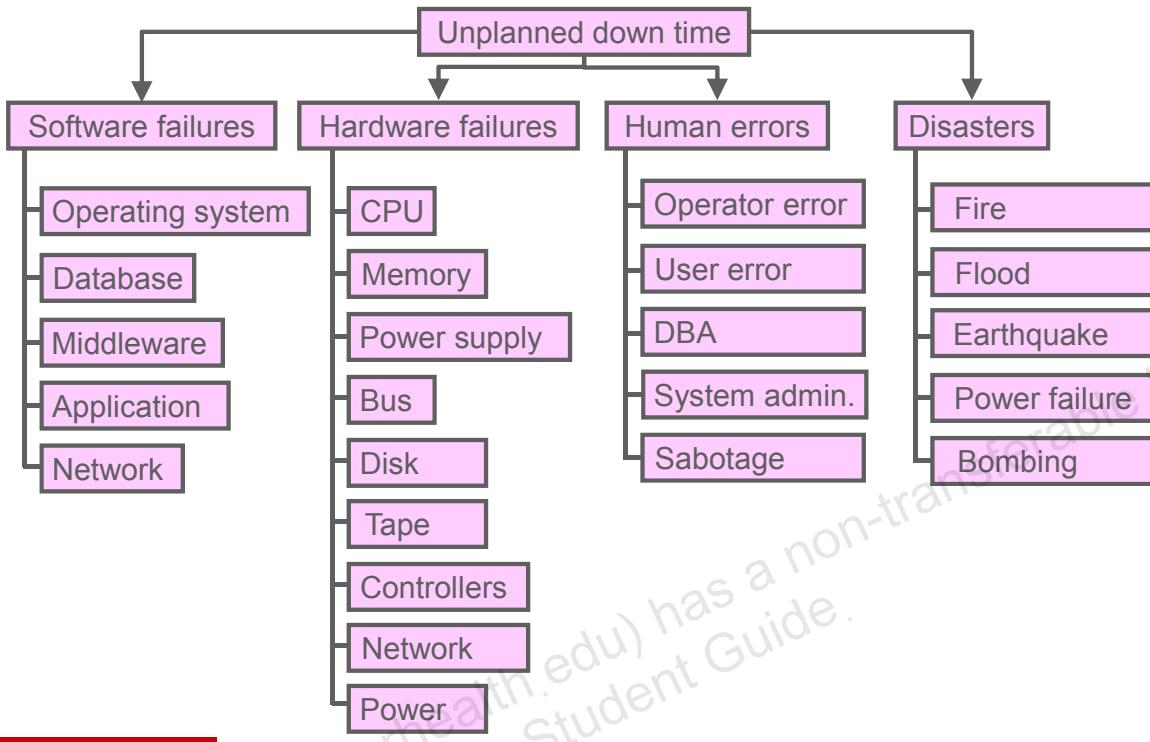
- Describe Oracle Clusterware
- Explain Oracle Automatic Storage Management (ASM)
- Describe Oracle Real Application Clusters (RAC)
- Explain Oracle Data Guard
- Design a Maximum Availability Architecture in your environment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Causes of Unplanned Down Time



ORACLE®

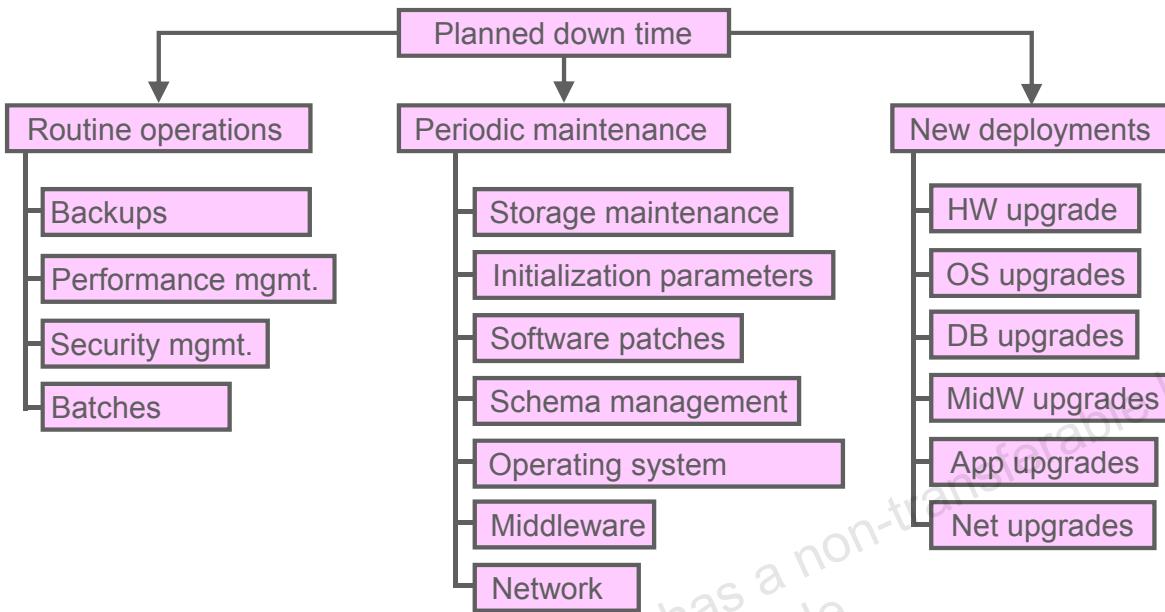
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

One of the true challenges in designing a highly available solution is examining and addressing all the possible causes of down time. It is important to consider causes of both unplanned and planned down time. The diagram shown in the slide, which is a taxonomy of unplanned failures, classifies failures as software failures, hardware failures, human error, and disasters.

Under each category heading is a list of possible causes of failures related to that category:

- Software failures include operating system, database, middleware, application, and network failures. A failure of any one of these components can cause a system fault.
- Hardware failures include system, peripheral, network, and power failures.
- Human error, which is a leading cause of failures, includes errors by an operator, user, database administrator, or system administrator. Another type of human error that can cause unplanned down time is sabotage.
- The final category is disasters. Although infrequent, these can have extreme impacts on enterprises, because of their prolonged effect on operations. Possible causes of disasters include fires, floods, earthquakes, power failures, and bombings. A well-designed high-availability solution accounts for all these factors in preventing unplanned down time.

Causes of Planned Down Time



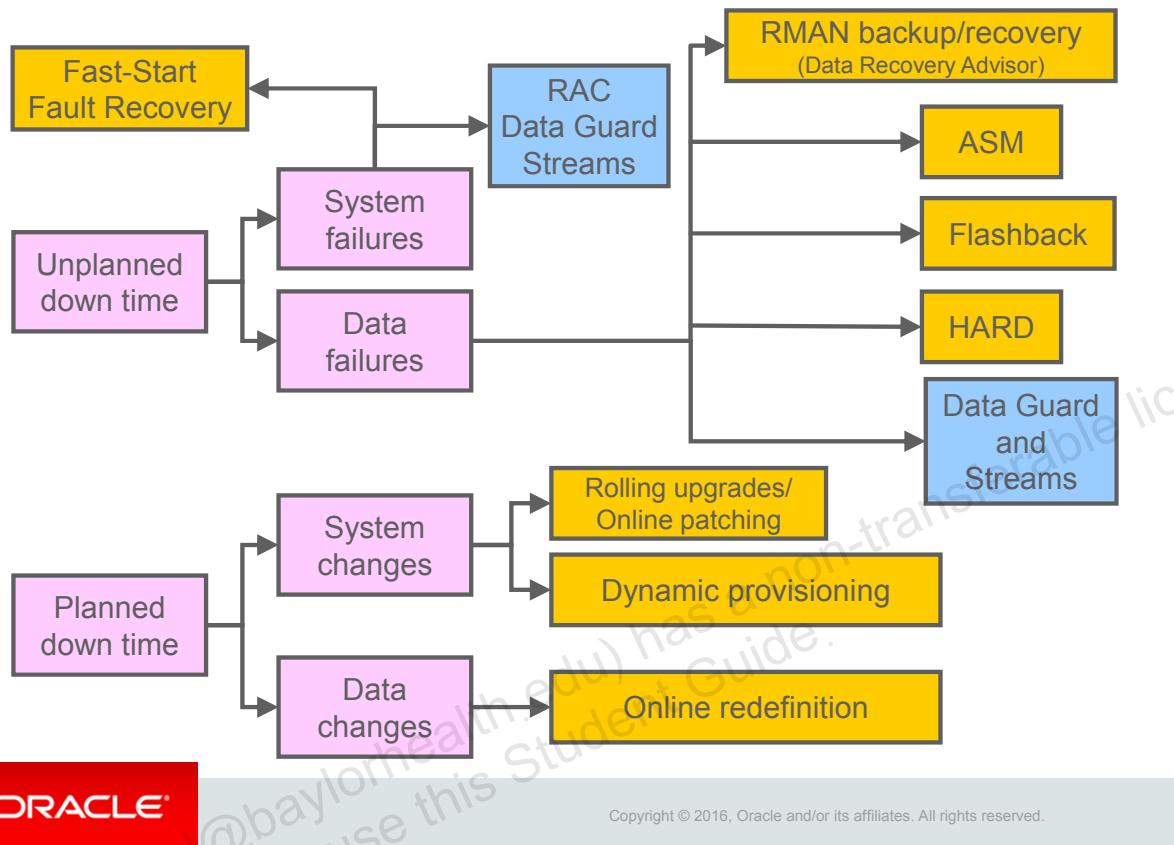
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Planned down time can be just as disruptive to operations, especially in global enterprises that support users in multiple time zones, up to 24 hours per day. In these cases, it is important to design a system to minimize planned interruptions. As shown in the diagram in the slide, causes of planned down time include routine operations, periodic maintenance, and new deployments. Routine operations are frequent maintenance tasks that include backups, performance management, user and security management, and batch operations.

Periodic maintenance, such as installing a patch or reconfiguring the system, is occasionally necessary to update the database, application, operating system, middleware, or network.

New deployments describe major upgrades to the hardware, operating system, database, application, middleware, or network. It is important to consider not only the time to perform the upgrade, but also the effect the changes may have on the overall application.

Oracle's Solution to Down Time



Unplanned down time is primarily the result of computer failures or data failures. Planned down time is primarily due to data changes or system changes:

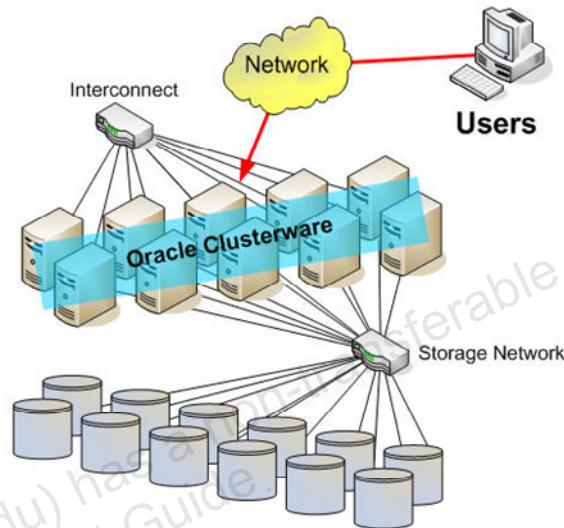
- RAC provides optimal performance, scalability, and availability gains.
- Fast-Start Fault Recovery enables you to bound the crash/recovery time. The database self-tunes checkpoint processing to safeguard the desired recovery time objective.
- ASM provides a higher level of availability by using online provisioning of storage.
- Flashback provides a quick resolution to human errors.
- Oracle Hardware Assisted Resilient Data (HARD) is a comprehensive program designed to prevent data corruptions before they happen.
- Recovery Manager (RMAN) automates database backup and recovery. Data Recovery Advisor (not supported for RAC) diagnoses data failures and presents repair options.
- Data Guard must be the foundation of any Oracle database disaster-recovery plan.
- The increased flexibility and capability of Streams over Data Guard with SQL Apply requires more expense and expertise to maintain an integrated high availability solution.

- With online redefinition, the Oracle database supports many maintenance operations without disrupting database operations, or preventing users from updating or accessing data.
- Oracle Database continues to broaden support for dynamic reconfiguration, enabling it to adapt to changes in demand and hardware with no disruption of service.
- Oracle Database supports the application of patches to the nodes of a RAC system, as well as database software upgrades, in a rolling fashion.

Review of Oracle Clusterware

Oracle Clusterware is:

- A key part of Oracle Grid Infrastructure
- Integrated with ASM
- The basis for Oracle Cloud File System
- A foundation for RAC
- A generalized cluster infrastructure for all kinds of applications



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

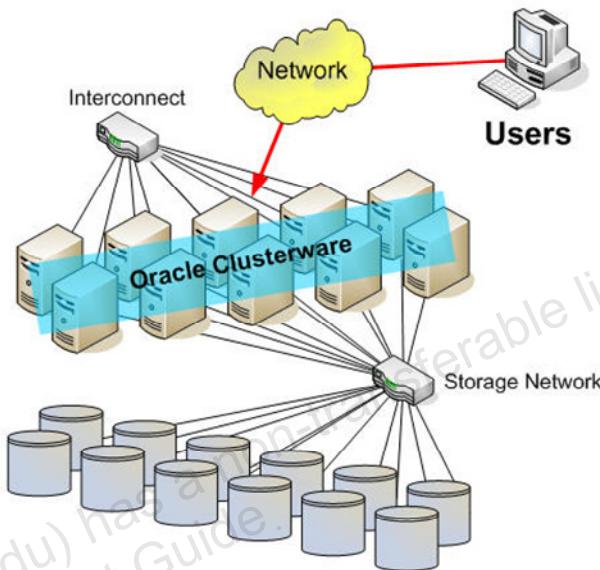
Oracle Clusterware is a key part of Oracle Grid Infrastructure, which also includes ASM and the Oracle Cloud File System. Oracle Clusterware can use ASM for all the shared files required by the cluster. Oracle Clusterware is also a foundation for the Oracle ASM Cluster File System (ACFS), a generalized cluster file system that can be used for most file-based data such as documents, spreadsheets, and reports.

The combination of Oracle Clusterware, ASM, and ACFS provides administrators with a unified cluster solution that is not only the foundation for the RAC database, but can also be applied to all kinds of other applications. Oracle Clusterware also manages resources, such as virtual IP (VIP) addresses, databases, listeners, services, and so on.

Using Oracle Clusterware eliminates the need for proprietary vendor clusterware and provides the benefit of using only Oracle software. Oracle provides an entire software solution, including everything from disk management with Oracle ASM to data management with Oracle Database and Oracle RAC. In addition, Oracle Database features such as Oracle Services provide advanced functionality when used with the underlying Oracle Clusterware high availability framework.

Clusterware Services

- Shared disk cluster architecture supporting application load balancing and failover
- Services include:
 - Cluster management
 - Node monitoring
 - Event services
 - Time synchronization
 - Network management
 - High availability
 - Cluster Interconnect Link Aggregation (HAIP)



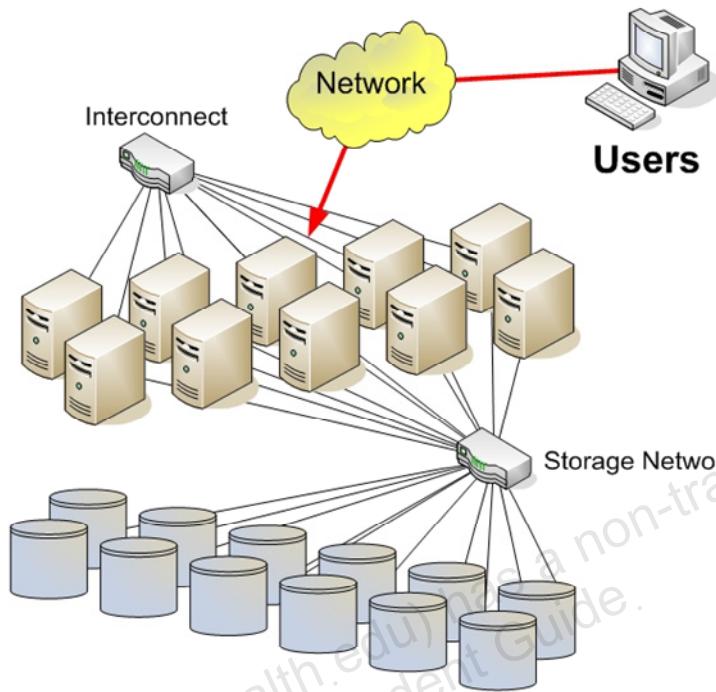
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Clusterware provides a complete set of cluster services to support the shared disk, load balancing cluster architecture of the RAC database. Oracle Clusterware can also be used to provide failover clustering services for single-instance Oracle databases and other applications. The services provided by Oracle Clusterware include:

- Cluster management, which allows cluster services and application resources to be monitored and managed from any node in the cluster
- Node monitoring, which provides real-time information regarding which nodes are currently available and the resources they support. Cluster integrity is also protected by evicting or fencing unresponsive nodes.
- Event services, which publishes cluster events so that applications are aware of changes in the cluster
- Time synchronization, which synchronizes the time on all nodes of the cluster
- Network management, which provisions and manages VIP addresses that are associated with cluster nodes or application resources to provide a consistent network identity regardless of which nodes are available. In addition, Grid Naming Service (GNS) manages network naming within the cluster.
- High availability, which services, monitors, and restarts all other resources as required
- Cluster Interconnect Link Aggregation (HAIP)

Deployment Option: Standard Cluster



ORACLE®

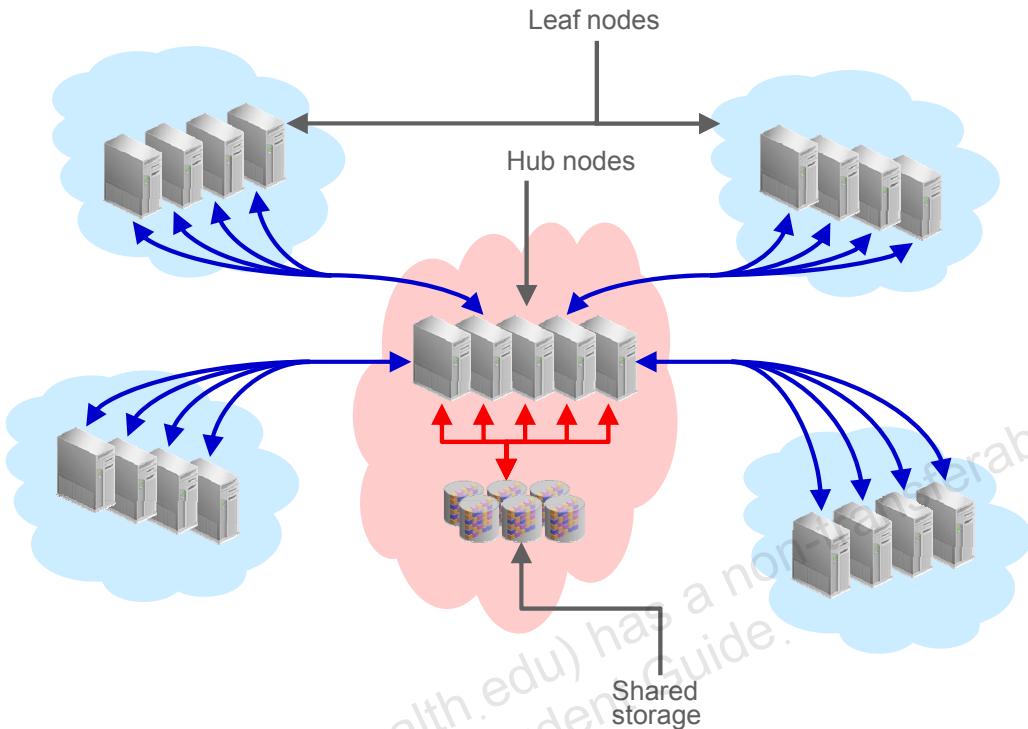
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Clusterware can be installed with one of two deployment options: Standard Cluster and Flex Cluster. The diagram in the slide shows an example of the standard cluster architecture.

The standard cluster consists of a group of independent but interconnected computers whose combined resources can be applied to a processing task. The architecture uses a dedicated network (cluster interconnect) for communication and coordination between cluster nodes and shared disk storage.

With the introduction of Oracle 12c Flex Clusters, pure shared disk clusters are not the only type of clustered hardware supported. The architecture has become hybrid with the introduction of hub and leaf nodes.

Deployment Option: Flex Cluster



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Previous releases of Oracle Clusterware have been used to build large production clusters containing between 32 and 64 nodes. A few clusters larger than 100 nodes have been successfully deployed. With Oracle Clusterware 12c, a new set of features enables Flex Clusters. In this release, Flex Clusters are designed to scale well up to 2000 nodes.

In release 12.1, you can use Flex Clusters to:

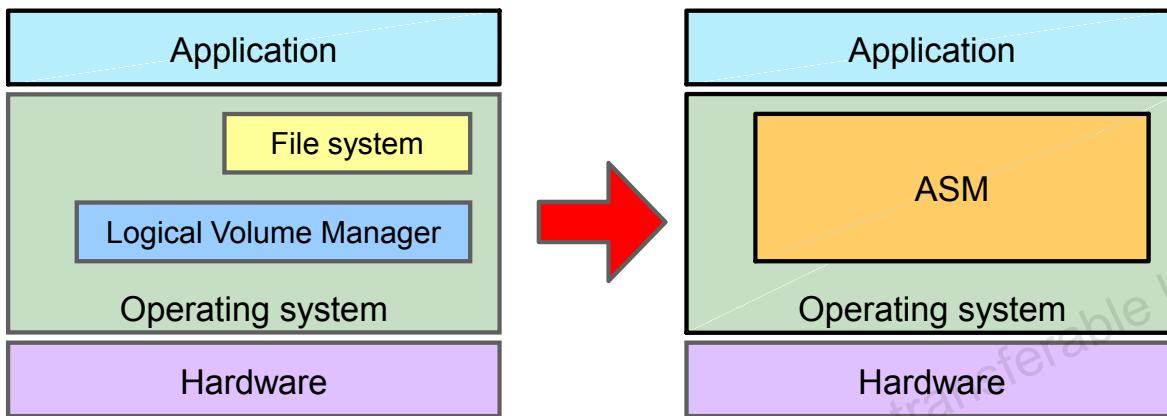
- Manage large pools of application resources with high-availability and failover protection
- Efficiently support multiple highly available databases and applications running in a single cluster

Flex Clusters use a hub-and-spoke topology, as illustrated in the slide.

The core of a Flex Cluster is a group of Hub Nodes. The group is essentially the same as a release 11.2 cluster, and can scale up to the size of an existing release 11.2 cluster. There must be one, and only one, group of Hub Nodes in a Flex Cluster deployment, and like a release 11.2 cluster, each Hub Node must be connected to storage that is shared across the group of Hub Nodes.

Zero or more Leaf Nodes may be connected to a Flex Cluster. Each Leaf Node is connected to the cluster through a Hub Node. Leaf Nodes do not require direct access to the shared storage connected to the Hub Nodes.

Review of Oracle ASM



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle ASM is a volume manager and a file system for Oracle Database files that supports single-instance Oracle Database and Oracle RAC configurations. Oracle ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

Combining volume management functions with a file system allows a level of integration and efficiency that would not otherwise be possible. For example, ASM is able to avoid the overhead associated with a conventional file system and achieve native raw disk performance for Oracle data files and other file types supported by ASM.

ASM is engineered to operate efficiently in both clustered and non-clustered environments.

Oracle ASM files can coexist with other storage management options such as raw disks and third-party file systems. This capability simplifies the integration of Oracle ASM into pre-existing environments.

ASM Features and Benefits

- Stripes files rather than logical volumes
- Provides redundancy on a file basis
- Enables online disk reconfiguration and dynamic rebalancing
- Reduces significantly the time taken to resynchronize a transient failure by tracking changes while the disk is offline
- Provides adjustable rebalancing speed
- Is cluster-aware
- Supports reading from mirrored copy instead of primary copy for extended clusters
- Is automatically installed as part of the Grid Infrastructure



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

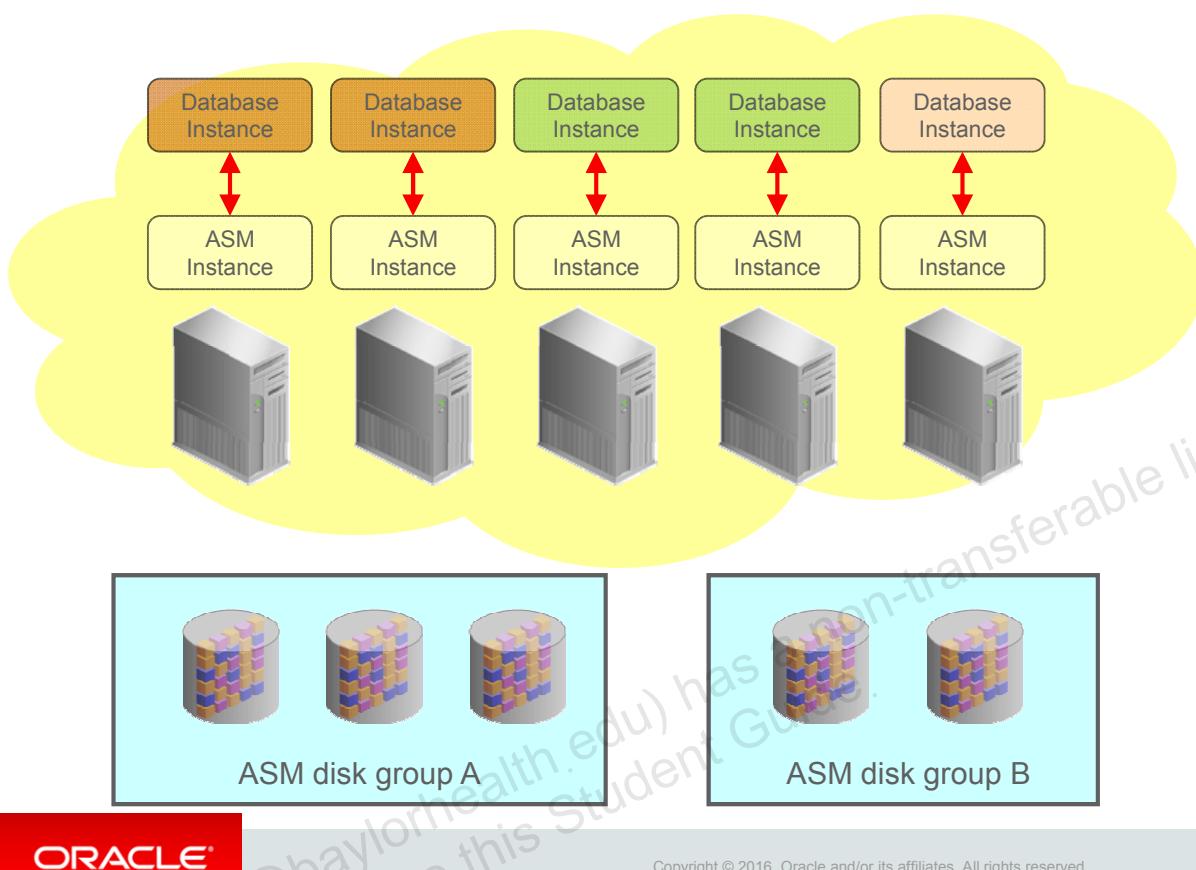
ASM provides striping and mirroring without the need to purchase a third-party Logical Volume Manager. ASM divides a file into pieces and spreads them evenly across all the disks. It uses an index technique to track the placement of each piece. Traditional striping techniques use mathematical functions to stripe complete logical volumes. ASM is unique in that it applies mirroring on a file basis, rather than on a volume basis. Therefore, the same disk group can contain a combination of files protected by mirroring or not protected at all.

When your storage capacity changes, ASM does not restripe all the data. However, in an online operation, ASM moves data proportional to the amount of storage added or removed to evenly redistribute the files and maintain a balanced I/O load across the disks. You can adjust the speed of rebalance operations to increase or decrease the speed and adjust the impact on the I/O subsystem. This capability also enables fast resynchronization of disks that may suffer a transient failure.

ASM supports all Oracle database file types. ASM supports RAC and eliminates the need for a cluster Logical Volume Manager or a cluster file system. In extended clusters, you can set a preferred read copy.

ASM is included in the Grid Infrastructure installation. It is available for both the Enterprise Edition and the Standard Edition installations.

Deployment Option: Standard ASM



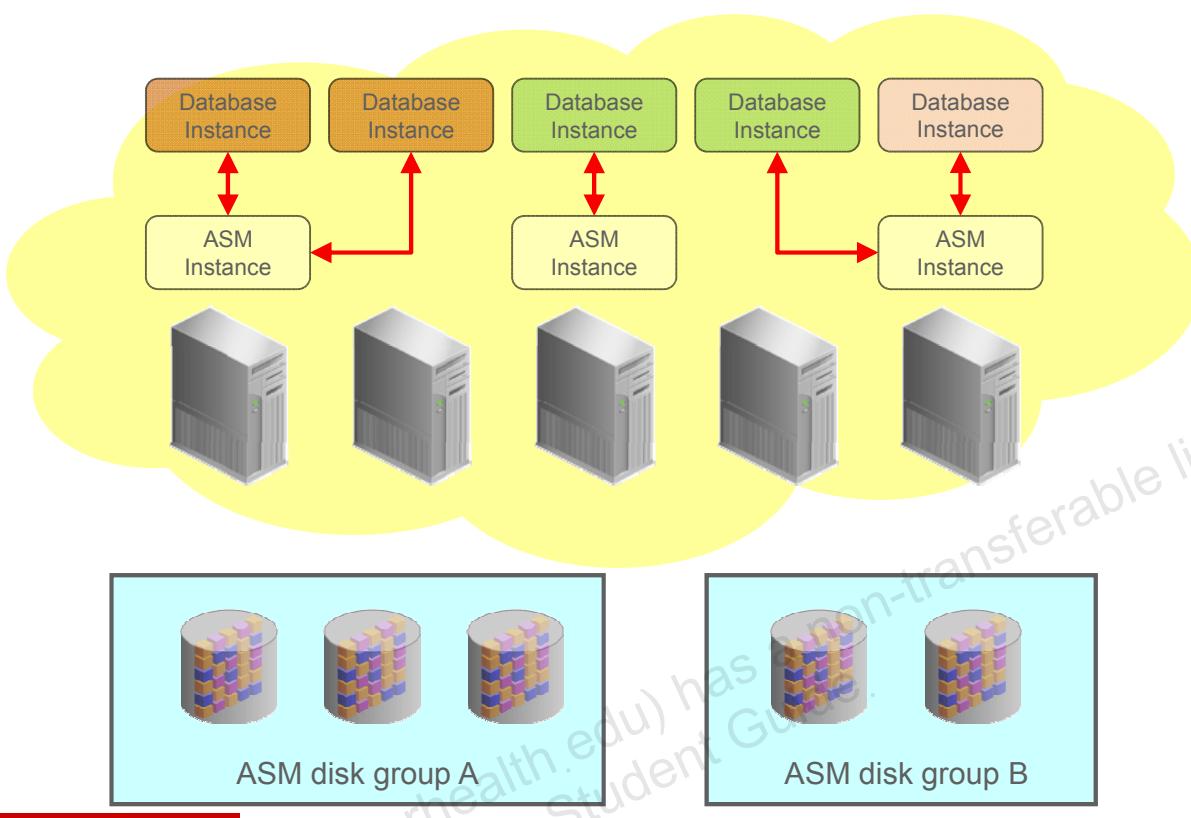
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide illustrates an example of the standard ASM configuration. In this ASM configuration, there is one ASM instance for each node that can provide space management for multiple Oracle RAC or single-instance databases in the cluster.

The ASM instances coordinate with each other when provisioning disk space from the multiple disk groups. In this design, ASM provides storage consolidation, and RAC possibly provides database consolidation.

Deployment Option: Flex ASM



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide illustrates another possible ASM instance design, which is Flex ASM. Prior to Oracle Database 12c, an ASM client (database instance or ACFS) can only connect to an ASM instance running on the same host. This requires every database server to dedicate system resources to ASM, which increases the overall system resource requirement to run Oracle Database in conjunction with ASM. This tightly coupled model also has availability concerns because, if an ASM instance fails, all ASM clients on that host must also fail.

Oracle Database 12c introduces Flex ASM. Flex ASM allows ASM clients to connect to ASM over a network. By relaxing the hard dependency between ASM and its clients, the previous architectural limitations are overcome. With Flex ASM, a smaller pool of ASM instances can be used to serve a large pool of database servers. If an ASM instance fails, its clients can reconnect to another ASM instance.

Flex ASM and Flex Clusters

- Flex Clusters require Flex ASM.
 - Standard ASM is not supported on a Flex Cluster.
- Flex ASM does not require a Flex Cluster.
 - Flex ASM can run on a standard cluster servicing clients across the cluster.
 - Flex ASM can run on the Hub Nodes of a Flex Cluster servicing clients across the Hub Nodes of the Flex Cluster.
- The benefits of Flex ASM apply regardless of cluster type:
 - Smaller ASM resource footprint
 - Protection from ASM failure



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Standard ASM is not supported on a Flex Cluster. Therefore, Flex Clusters require Flex ASM. However, Flex ASM does not require a Flex Cluster. Flex ASM can be configured on either a standard cluster or a Flex Cluster.

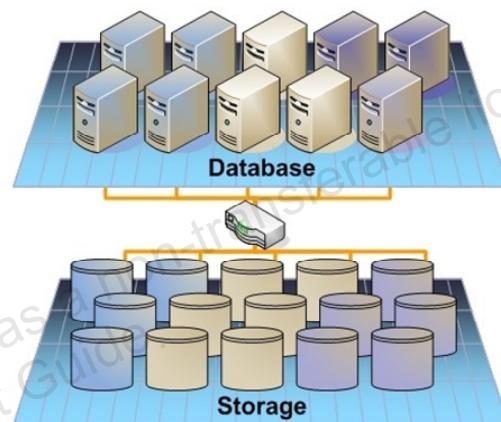
When Flex ASM runs on a standard cluster, ASM services can run on a subset of cluster nodes servicing clients across the cluster. When Flex ASM runs on a Flex Cluster, ASM services can run on a subset of Hub Nodes servicing clients across all of the Hub Nodes in the Flex Cluster.

The fundamental benefits of Flex ASM apply regardless of the type of cluster being used. These benefits are as follows:

- The overall resource footprint is smaller because a smaller pool of ASM instances can be used to serve a larger pool of database servers.
- Higher availability can be achieved because if an ASM instance fails, its clients can reconnect to another ASM instance.

Review of Oracle RAC

- RAC is a feature of the Oracle Database that enables multiple clustered instances of Oracle to simultaneously transact against a single instance database.
- Key benefits:
 - Flexible workload management
 - Incremental scalability
 - High availability
 - Failure protection
 - Rolling updates
 - Cost efficiency
 - High performance
 - Application compatibility



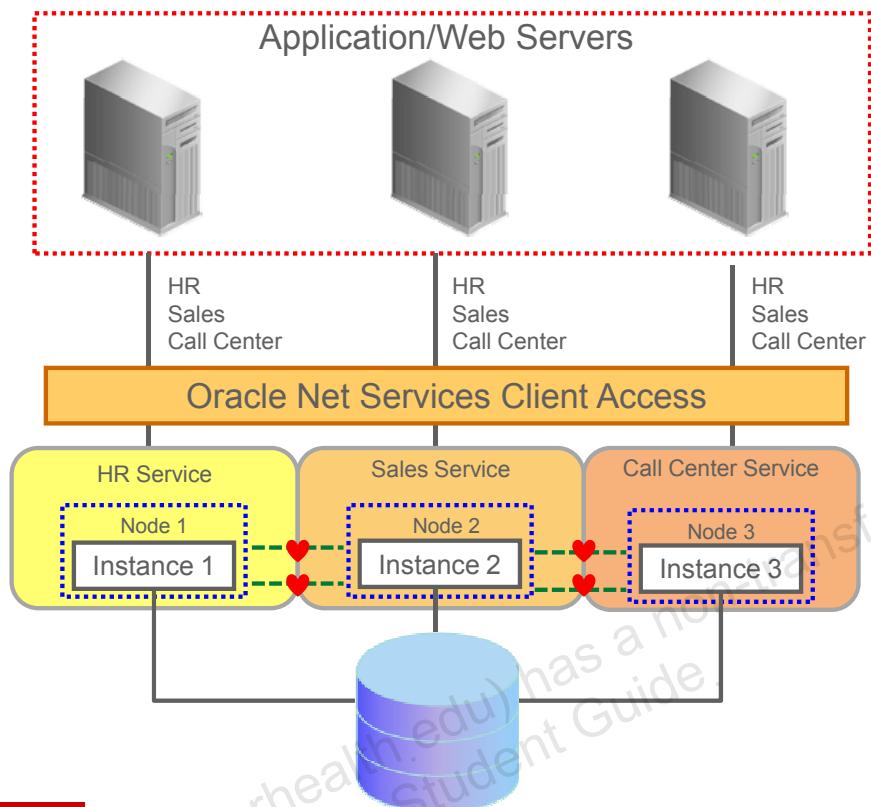
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

RAC is a feature of the Oracle Database that enables multiple clustered instances of Oracle to simultaneously transact against a single instance database. RAC delivers the following key benefits:

- **Flexible resource management:** You can isolate workloads running on the cluster to provide guaranteed performance for each workload. You can dynamically reconfigure resources to optimize your environment for different workload mixtures.
- **Incremental scalability:** You can expand a RAC database by easily adding cluster nodes as workloads increase.
- **High availability:** Overall database availability is maintained if a cluster node experiences a failure or if servers need to be patched or updated.
- **Cost efficiency:** You can assemble powerful clusters by using commonly available servers that are relatively inexpensive when compared with large-scale offerings.
- **High performance:** The combined power of a cluster delivers extremely high performance. This has been demonstrated in many large-scale customer deployments along with industry-standard benchmarks.
- **Application compatibility:** Most prepackaged and custom applications can leverage the benefits of RAC without any application code changes. Sophisticated enterprise business applications such as SAP, PeopleSoft, Siebel, and Oracle E-Business Suite have been certified against RAC.

Typical Oracle RAC Architecture



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

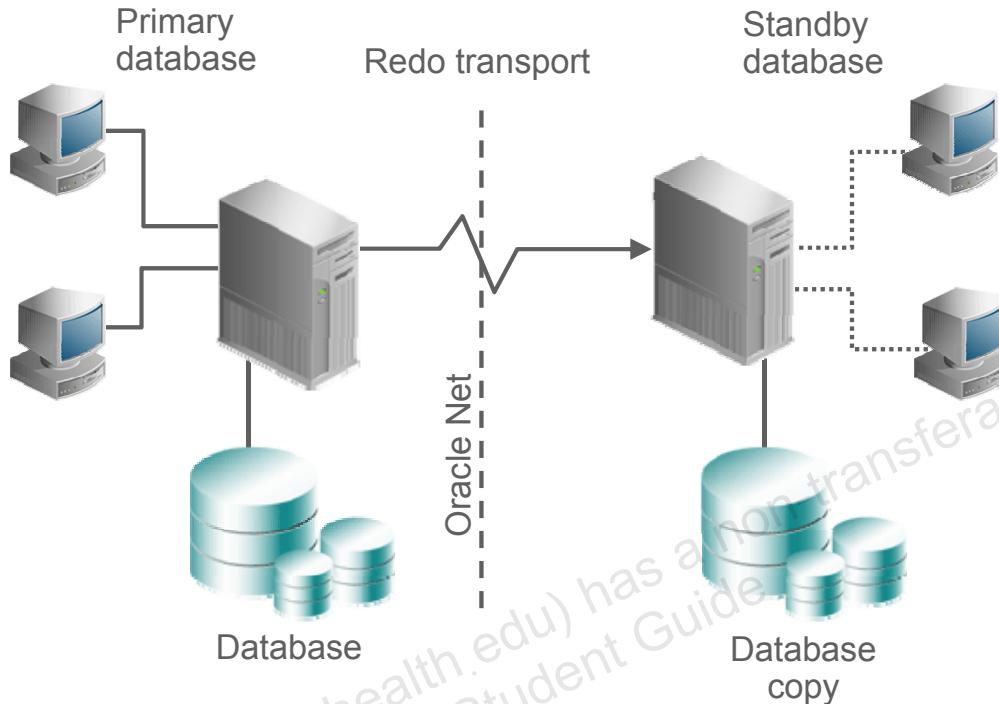
The graphic in the slide shows how Oracle RAC is the Oracle Database option that provides a single system image for multiple servers to access one Oracle database. In Oracle RAC, each Oracle instance must run on a separate server.

Traditionally, an Oracle RAC environment is located in one data center. However, you can configure Oracle RAC on an extended distance cluster, which is an architecture that provides extremely fast recovery from a site failure and allows for all nodes, at all sites, to actively process transactions as part of a single database cluster. In an extended cluster, the nodes in the cluster are typically dispersed, geographically, such as between two fire cells, between two rooms or buildings, or between two different data centers or cities. For availability reasons, the data must be located at both sites, thus requiring the implementation of disk mirroring technology for storage.

If you choose to implement this architecture, you must assess whether this architecture is a good solution for your business, especially considering distance, latency, and the degree of protection it provides. Oracle RAC on extended clusters provides higher availability than is possible with local Oracle RAC configurations, but an extended cluster may not fulfill all of the disaster-recovery requirements of your organization. A feasible separation provides great protection for some disasters (for example, local power outage or server room flooding) but it cannot provide protection against all types of outages.

For comprehensive protection against disasters, including protection against corruptions and regional disasters, Oracle recommends the use of Oracle Data Guard with Oracle RAC.

Review of Oracle Data Guard



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Data Guard is a central component of an integrated Oracle Database High Availability (HA) solution set that helps organizations ensure business continuity by minimizing the various kinds of planned and unplanned down time that can affect their businesses.

Oracle Data Guard is a management, monitoring, and automation software infrastructure that works with a production database and one or more standby databases to protect your data against failures, errors, and corruptions that might otherwise destroy your database. It protects critical data by providing facilities to automate the creation, management, and monitoring of the databases and other components in a Data Guard configuration. It automates the process of maintaining a copy of an Oracle production database (called a *standby database*) that can be used if the production database is taken offline for routine maintenance or becomes damaged.

In a Data Guard configuration, a production database is referred to as the *primary database*. A *standby database* is a synchronized copy of the primary database. By using a backup copy of the primary database, you can create from 1 to 30 standby databases. The standby databases, together with the primary database, make up a Data Guard configuration.

All Data Guard standby databases can enable up-to-date read access to the standby database while redo being received from the primary database is applied. This makes all standby databases excellent candidates for relieving the primary database of the overhead of supporting read-only queries and reporting.

Types of Standby Databases

- Physical standby database:
 - Is identical to the primary database on a block-for-block basis
 - Is synchronized with the primary database through application of redo data received from the primary database
 - Can be used concurrently for data protection and reporting
- Logical standby database:
 - Shares the same schema definition as the production database
 - Is kept synchronized with the primary database by transforming the redo data received from the primary database into SQL statements and then executing the SQL statements
 - Can be used concurrently for data protection, reporting, and database upgrades



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Physical Standby Database

A physical standby database is physically identical to the primary database, with on-disk database structures that are identical to the primary database on a block-for-block basis. The physical standby database is updated by performing recovery using redo data that is received from the primary database. Oracle Database12c enables a physical standby database to receive and apply redo while it is open in read-only mode.

Logical Standby Database

A logical standby database contains the same logical information (unless configured to skip certain objects) as the production database, although the physical organization and structure of the data can be different. The logical standby database is kept synchronized with the primary database by transforming the data in the redo received from the primary database into SQL statements and then executing the SQL statements on the standby database. This is done with the use of LogMiner technology on the redo data received from the primary database. The tables in a logical standby database can be used simultaneously for recovery and for other tasks such as reporting, summations, and queries.

Note: For more information about LogMiner, see *Using LogMiner to Analyze Redo Log Files* in *Oracle Database Utilities 12c Release 1 (12.1)* documentation.

Types of Standby Databases

- Snapshot standby database:
 - Is a fully updatable standby database
 - Is created by converting a physical standby database
 - Can be used for updates, but those updates are discarded before the snapshot standby database is converted back into a physical standby database
 - Can be used for testing

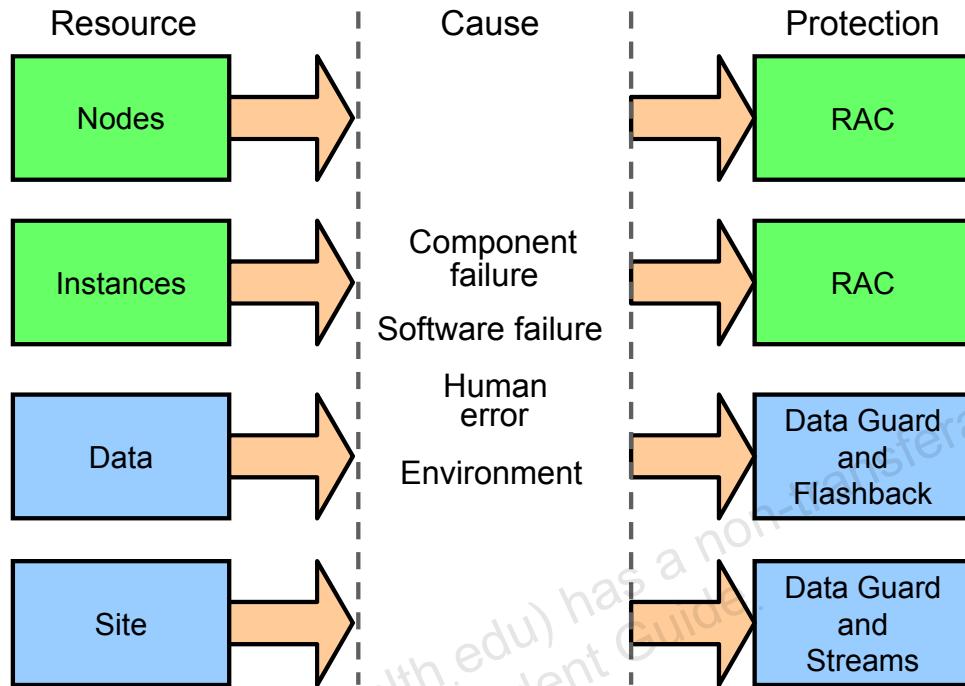


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Snapshot Standby Database

A snapshot standby database is a database that is created by converting a physical standby database into a snapshot standby database. The snapshot standby database receives redo from the primary database, but does not apply the redo data until it is converted back into a physical standby database. The snapshot standby database can be used for updates, but those updates are discarded before the snapshot standby database is converted back into a physical standby database. The snapshot standby database is appropriate when you require a temporary, updatable version of a physical standby database.

RAC and Data Guard Complementarity



ORACLE®

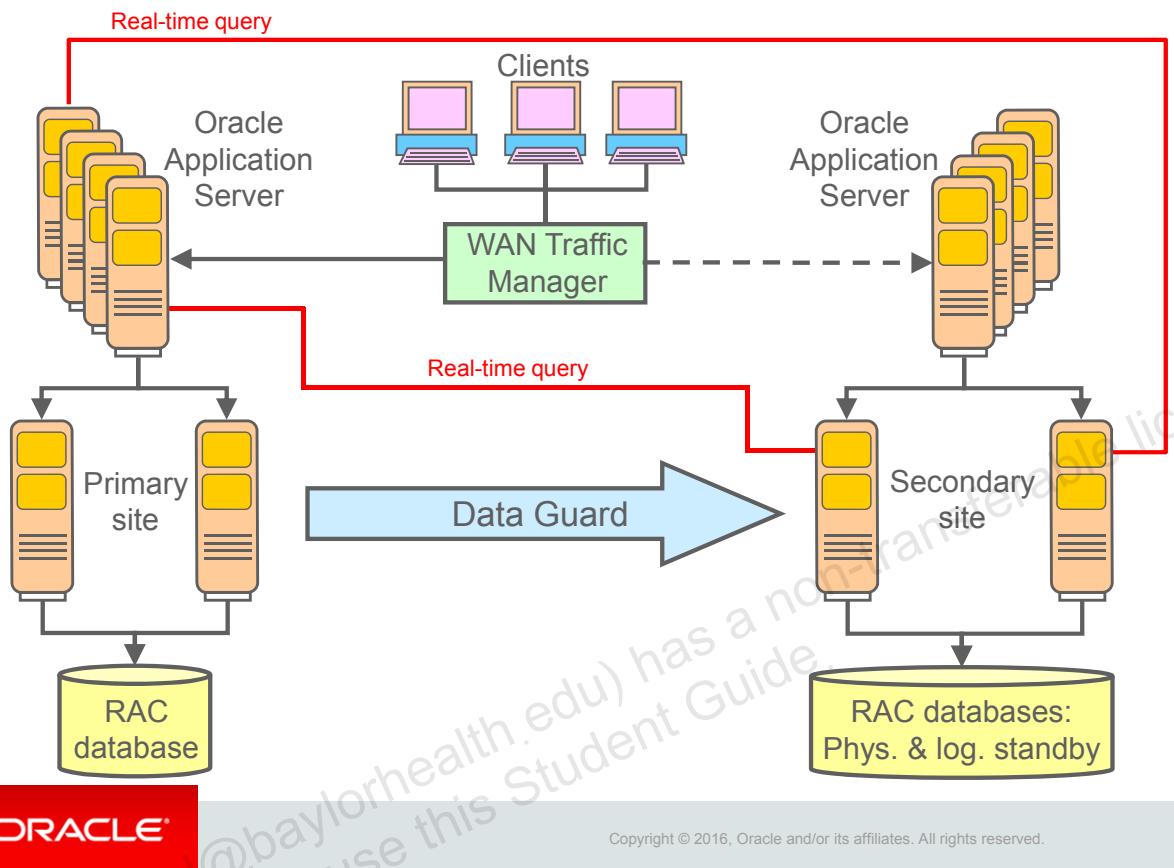
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

RAC and Data Guard together provide the benefits of system-level, site-level, and data-level protection, resulting in high levels of availability and disaster recovery without loss of data.

- RAC addresses system failures by providing rapid and automatic recovery from failures, such as node failures and instance crashes.
- Data Guard addresses site failures and data protection through transactionally consistent primary and standby databases that do not share disks, enabling recovery from site disasters and data corruption.

Note: Unlike Data Guard using SQL Apply, Oracle Streams enables updates on the replica and provides support for heterogeneous platforms with different database releases. Therefore, Oracle Streams may provide the fastest approach for database upgrades and platform migration.

Deployment Option: Maximum Availability Architecture



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Maximum Availability Architecture (MAA)

RAC and Data Guard provide the basis of the database MAA solution. MAA provides the most comprehensive architecture for reducing down time for scheduled outages and preventing, detecting, and recovering from unscheduled outages. The recommended MAA has two identical sites. The primary site contains the RAC database, and the secondary site contains both a physical standby database and a logical standby database on RAC. Identical site configuration is recommended to ensure that performance is not sacrificed after a failover or switchover. Symmetric sites also enable processes and procedures to be kept the same between sites, making operational tasks easier to maintain and execute.

The graphic in the slide illustrates identically configured sites. Each site consists of redundant components and redundant routing mechanisms, so that requests are always serviceable even in the event of a failure. Most outages are resolved locally. Client requests are always routed to the site playing the production role.

After a failover or switchover operation occurs due to a serious outage, client requests are routed to another site that assumes the production role. Each site contains a set of application servers or mid-tier servers. The site playing the production role contains a production database using RAC to protect from host and instance failures. The site playing the standby role contains one standby database, and one logical standby database managed by Data Guard. Data Guard switchover and failover functions allow the roles to be traded between sites.

Note: For more information, see the following website:
<http://otn.oracle.com/deploy/availability/htdocs/maa.htm>

Summary

In this lesson, you should have learned how to:

- Describe Oracle Clusterware
- Explain Oracle ASM
- Describe Oracle RAC
- Explain Oracle Data Guard
- Design a Maximum Availability Architecture in your environment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 1: Overview

This practice introduces the laboratory environment.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

2

Converting a Single Instance to an Oracle RAC Database

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Convert a single-instance database in a non-clustered environment to a RAC database
- Convert a single-instance database in a clustered environment to a RAC database

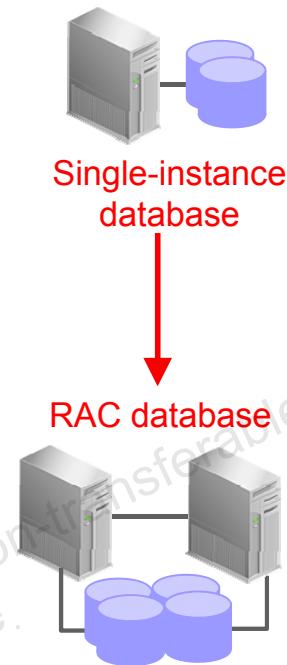


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Single Instance–to-RAC Conversion

- Single-instance databases can be converted to RAC using:
 - DBCA
 - Enterprise Manager
 - RCONFIG utility
- Before conversion, ensure that:
 - Your hardware and operating system are supported
 - Your cluster nodes have access to shared storage



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A single-instance Oracle database can be converted using several methods, which include Database Configuration Assistant (DBCA), Enterprise Manager, and the RCONFIG utility. You can use one of these tools based on your needs.

Before converting a single-instance database to a RAC database, ensure that your system meets the following conditions:

- It has a supported hardware and operating system configuration.
- It has shared storage. A supported Cluster File System, network file system (NFS) mount, or ASM is available and accessible from all nodes.
- Your applications have no design characteristics that preclude their use with cluster database processing.

Considerations for Converting Single-Instance Databases to Oracle RAC

- Backup procedures should be available before conversion takes place.
- Archiving in Oracle RAC environments requires a thread number in the archive file format.
- The archived logs from all instances of an Oracle RAC database are required for media recovery.
- By default, all database files are migrated to Oracle Managed Files (OMF).



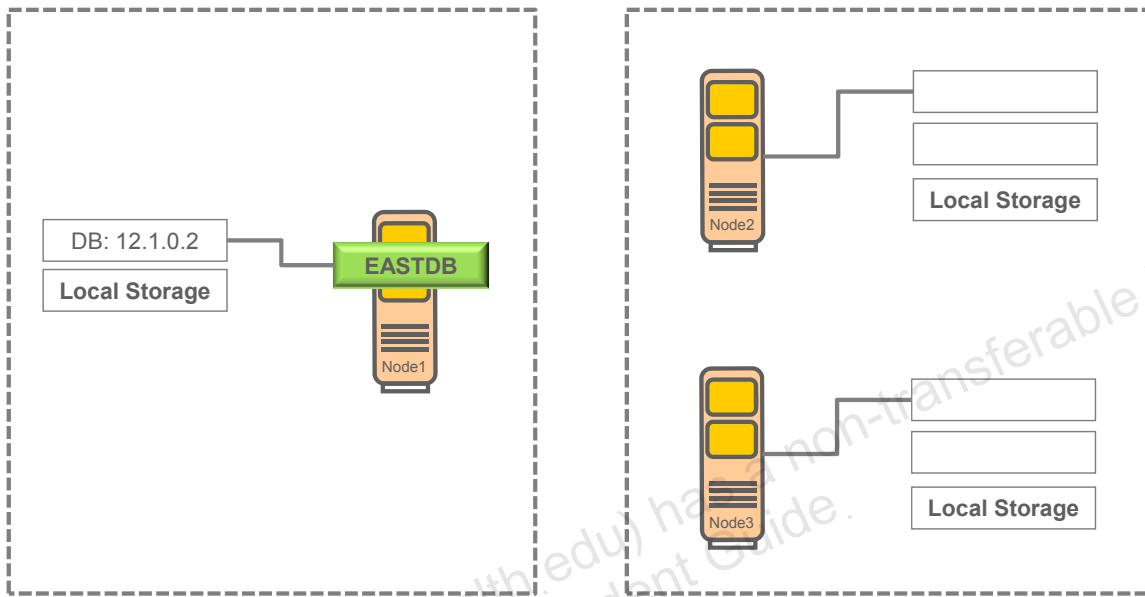
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Whatever method you choose to use for the conversion, note the following administrative considerations before converting single-instance databases to Oracle RAC:

- Backup procedures should be available before converting from a single-instance Oracle database to Oracle RAC. You should take a backup of your database before starting any major change.
- For archiving with Oracle RAC environments, the archive file format requires a thread number.
- The archived logs from all instances of an Oracle RAC database are required for media recovery. Because of this, if you archive to a file and you do not use a cluster file system, or some other means to provide shared file systems, then you require a method of accessing the archive logs from all nodes on which the cluster database has instances.
- By default, all database files are migrated to Oracle Managed Files (OMF). This feature simplifies tablespace creation, ensures data file location consistency and compliance with Optimal Flexible Architecture (OFA) rules, and reduces human error with data file management.

Scenario 1: Using DBCA

Conversion steps for a single-instance database **in a *non-clustered* environment:**



ORACLE®

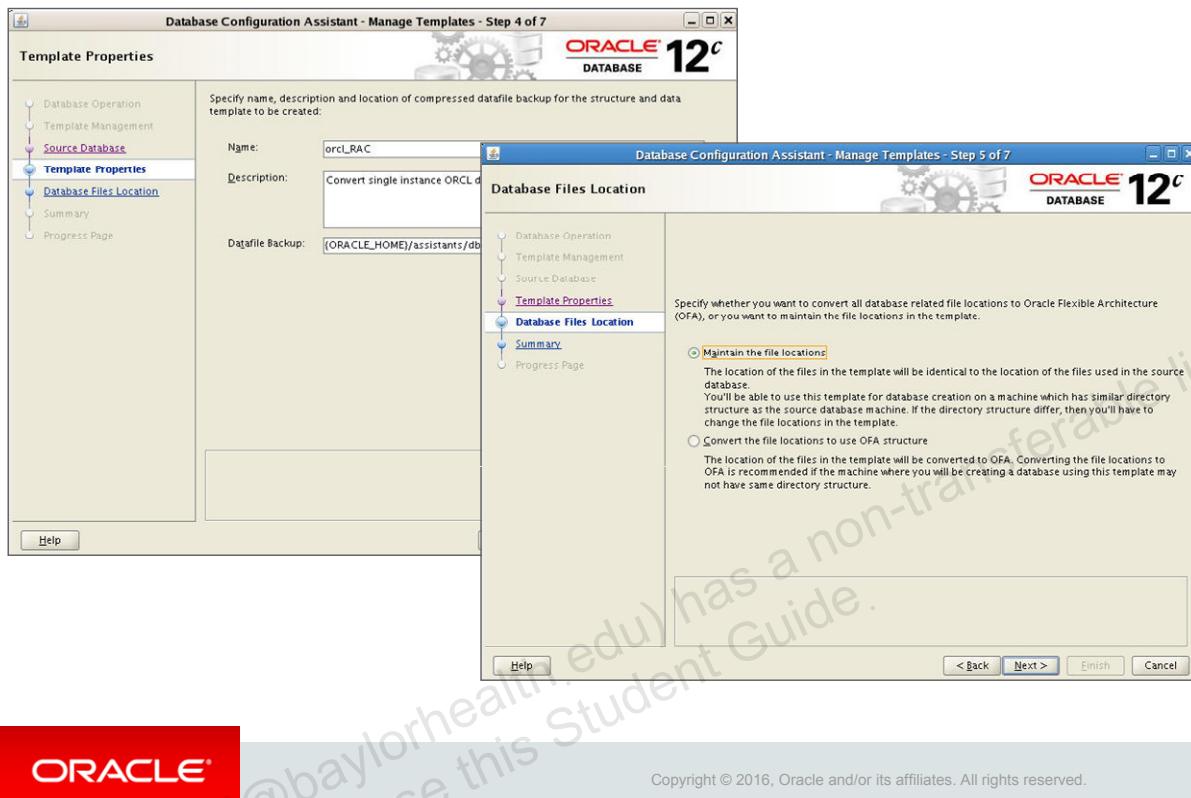
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can use DBCA to convert from single-instance Oracle databases to Oracle RAC or Oracle RAC One Node databases. DBCA automates the configuration of the control file attributes, creates the undo tablespaces and the redo logs, and creates the initialization parameter file entries for cluster-enabled environments. DBCA also configures Oracle Net Services, Oracle Clusterware resources, and the configuration for Oracle RAC database management using Oracle Enterprise Manager or the Server Control utility (SRVCTL).

To convert from a single-instance Oracle database that is on a non-cluster computer to a RAC database, perform the following steps:

1. Create an image of the single-instance database using DBCA.
2. Create an Oracle Cluster for RAC.
3. Copy the preconfigured database image files.
4. Create an Oracle RAC database using DBCA.

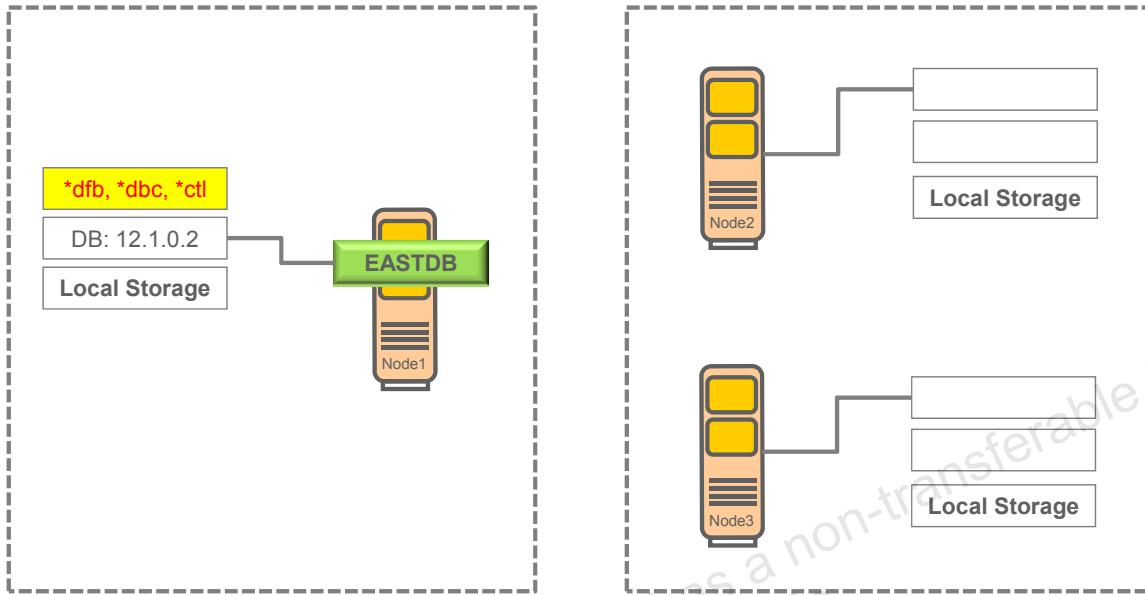
Step 1: Create an Image of the Single-Instance Database



Use the DBCA to create a preconfigured image of your single-instance database by using the following procedure:

1. Navigate to the `bin` directory in `$ORACLE_HOME`, and start the DBCA.
2. In the Welcome window, click Next.
3. In the Operations window, select Manage Templates, and click Next.
4. In the Template Management window, select “Create a database” template and “From an existing database (structure as well as data),” and click Next.
5. In the Source Database window, enter the database name in the Database instance field, and click Next.
6. In the Template Properties window, enter a template name in the Name field. By default, the template files are generated in `$ORACLE_HOME/assistants/dbca/templates`. Enter a description of the file in the Description field, and change the template file location in the Template data file field if you want. When you have finished, click Next.
7. In the Location of Database Related Files window, select “Maintain the file locations” so that you can restore the database to the current directory structure, and click Finish. The DBCA generates two files: a database structure file (`*.dbc`) and a database preconfigured image file (`*.dfb`).

Example: Result of Step 1



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- The DBCA generated three files: a database structure file (*.dbc), a control file (*.ctl), and a database preconfigured image file (*.dfb).
- The default location of these files is \$ORACLE_HOME/assistants/dbca/templates.

Step 2: Create an Oracle Cluster for RAC

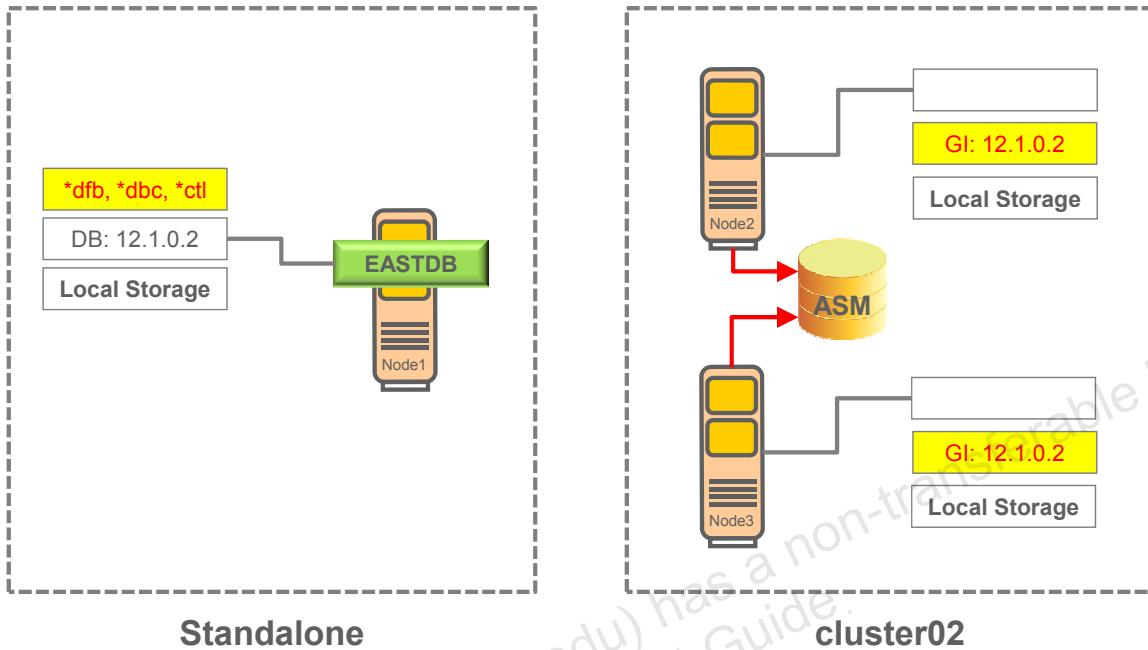
- Perform the pre-installation steps.
 - Tasks include kernel parameter configuration, hardware setup, network configuration, and shared storage setup.
- Set up and validate the cluster.
 - Create a cluster with the required number of nodes according to your hardware vendor's documentation.
 - Validate cluster components before installation.
 - Install Oracle Clusterware.
 - Validate the completed cluster installation by using `cluvfy`.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Perform the Pre-installation Steps:
Complete the Oracle Clusterware installation, as described in the *Oracle Grid Infrastructure Installation Guide* for your platform.
- Set Up and Validate the Cluster:
Form a cluster with the required number of nodes according to your business needs. When you have configured all the nodes in your cluster, validate cluster components by using the Cluster Verification Utility, and then install Oracle Clusterware. When the clusterware is installed, validate the completed cluster installation and configuration by using the Cluster Verification Utility.

Example: Result of Step 2



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Created a 2-node Oracle Cluster called **cluster02** for RAC by installing the Grid Infrastructure software on **node 2** and **node 3**

Step 3: Copy the Preconfigured Database Image

- Copy the preconfigured database image to a temporary location in one of the cluster nodes.
 - The database structure * .dbc file
 - The preconfigured database image * .dfb file
 - The control file image * .ctl file
- Move the database structure * .dbc file to the \$ORACLE_HOME/assistants/dbca/templates directory.
- Modify the * .dbc file to point to the preconfigured database image file in the temporary location.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- This includes copying the database structure * .dbc file and the database preconfigured image * .dfb file that the DBCA created in step one (Back Up the Original Single-Instance Database) to a temporary location on the node in the cluster from which you plan to run the DBCA.
- Move the database structure * .dbc file to the \$ORACLE_HOME/assistants/dbca/templates directory.
- Modify the * .dbc file to point to the preconfigured database image file in the temporary location.

Example: Database Structure File (*.dbc)

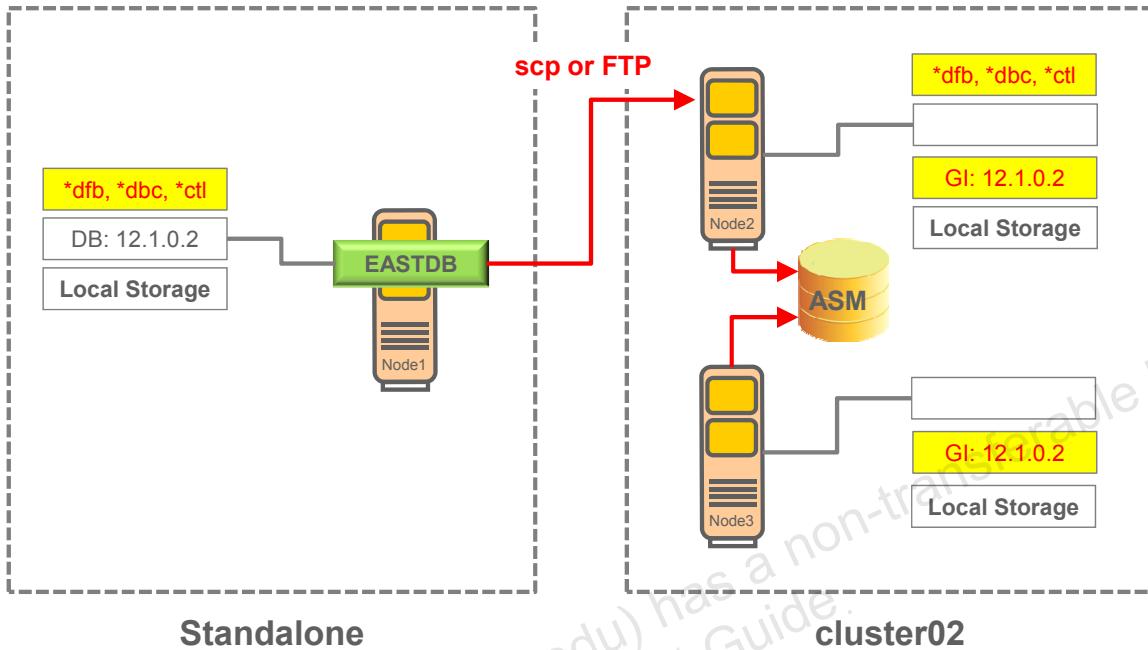
```
$ cd $ORACLE_HOME/assistants/dbca/templates  
$ vi template_name.dbc  
  
----- The Output Truncated-----  
  
<DataFiles>  
  <Location>  
    {ORACLE_HOME}/assistants/dbca/templates/template_name.dfb  
  </Location>  
  <SourceDBName cdb="false">eastdb</SourceDBName>  
  <Name id="1" Tablespace="SYSTEM" ... system01.dbf</Name>  
  <Name id="3" Tablespace="SYSAUX" ... sysaux01.dbf</Name>  
  <Name id="4" Tablespace="UNDOTBS1" ... undotbs01.dbf</Name>  
  <Name id="6" Tablespace="USERS" ... users01.dbf</Name>  
</DataFiles>  
<TempFiles>  
  <Name id="1" Tablespace="TEMP" ... temp01.dbf</Name>  
</TempFiles>  
  
----- The Output Truncated-----
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide shows the content of the database structure file (*.dbc). You need to modify this file if the location of the preconfigured database image (*.dfb) is different from the path specified in the file.

Example: Result of Step 3



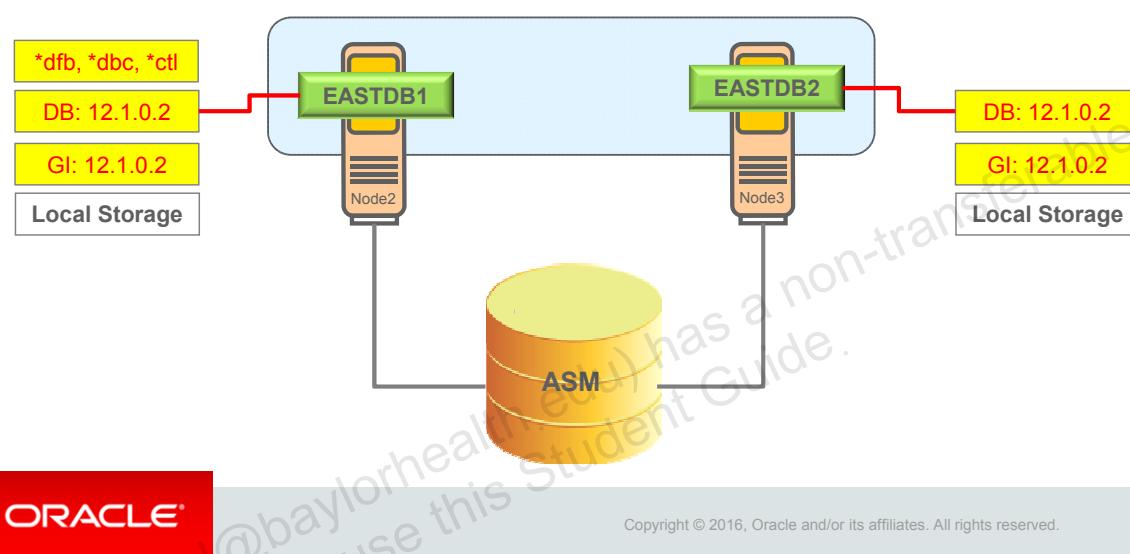
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Copied the preconfigured image files to the local storage of node 2
- Moved the template file (*.dbc) to \$ORACLE_HOME/assistants/dbca/templates directory
- Modified the template file (*.dbc) to point to the location where the preconfigured database image (*.dfb) was copied

Step 4: Create an Oracle RAC Database

- Install the Oracle Database 12c software.
- Create an Oracle RAC database using the preconfigured database images.

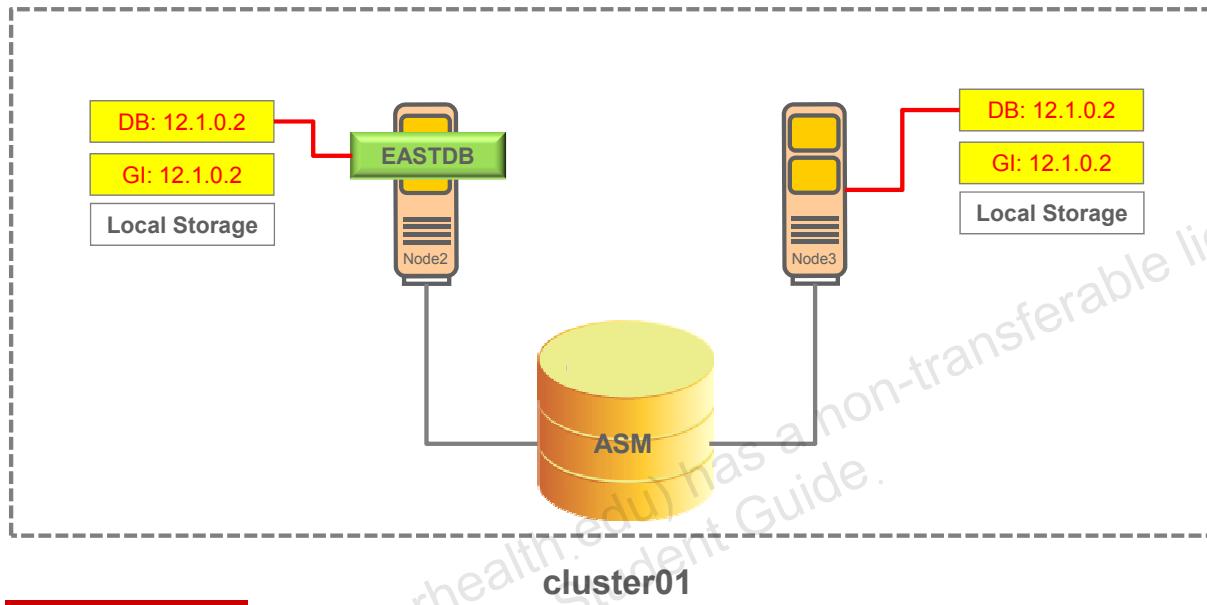


Install the Oracle Database 12c Software with RAC:

1. Run the Oracle Universal Installer (OUI) to perform an Oracle database installation with RAC. Select Cluster Installation Mode and select the nodes to include in your RAC database.
2. On the Oracle Universal Installer Database Configuration Types page, select the Advanced installation type. After installing the software, the OUI runs post-installation tools such as NETCA, DBCA, and so on.
3. In the DBCA Template Selection window, use the template that you copied to a temporary location in the “Copy the Preconfigured Database Image” step. Use the Browse option to select the template location.
4. After creating the RAC database, the DBCA displays the Password Management page in which you must change the passwords for database privileged users. When the DBCA exits, the conversion is complete.

Scenario 2: Using rconfig

- Conversion steps for a single-instance database **in a clustered environment:**



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can use the `rconfig` utility to convert from single-instance Oracle databases to Oracle RAC or Oracle RAC One Node databases. The `rconfig` utility automates many conversion steps.

To convert from a single-instance Oracle database that is on a cluster computer to a RAC database, perform the following steps:

1. Check the database type.
2. Modify the XML file for the `rconfig` utility.
3. Perform prerequisite checks.
4. Convert to an Oracle RAC database.
5. Verify the conversion.

Step 1: Check the Database Type

```
$ srvctl config database -db eastdb
Database unique name: eastdb
Database name: eastdb
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/eastdb/spfileeastdb.ora
Password file: +DATA/eastdb/orapweastdb
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: eastdb
Database instances: eastdb
Disk Groups: DATA, FRA
Mount point paths:
Services:
Type: SINGLE
Database is administrator managed
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The database type is **SINGLE**, which indicates a single-instance database.

Step 2: Modify the XML File for the rconfig Utility

- Locate the appropriate .xml file located in the \$ORACLE_HOME/assistants/rconfig/sampleXMLs directory.
- Modify the ConvertToRAC_AdminManaged.xml or ConvertToRAC_PolicyManaged.xml file as required for your system.
- Save the file under a different name.

```
$ cd $ORACLE_HOME/assistants/rconfig/sampleXMLs  
$ cp ConvertToRAC_AdminManaged.xml ConvertToRAC_AdminManaged.xml.bkp  
$ vi ConvertToRAC_AdminManaged.xml
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Before running the rconfig utility, modify the XML file as required for your system.

- Go to the \$ORACLE_HOME/assistants/rconfig/sampleXMLs directory as the oracle user and open the ConvertToRAC_*.xml file using a text editor, such as vi.
- Review the XML file, and modify the parameters as required for your system. The XML sample file contains comment lines that provide instructions about how to configure the file.
- When you have finished making changes, save the file with the syntax filename.xml. Make a note of the name you select.

Example: ConvertToRAC_AdminManaged.xml

- Modify the XML file as required for your system:

```
$ vi ConvertToRAC_AdminManaged.xml

<n:Convert verify="ONLY">
<n:SourceDBHome>/u01/app/oracle/product/12.1.0/dbhome_1</n:SourceDBHome>
<n:TargetDBHome>/u01/app/oracle/product/12.1.0/dbhome_1</n:TargetDBHome>
<n:SourceDBInfo SID="eastdb">
<n:Node name="enode01"/>
<n:Node name="enode02"/>
<n:InstancePrefix>eastdb</n:InstancePrefix>
<n:Password>oracle_4U</n:Password>
<n:TargetDatabaseArea></n:TargetDatabaseArea>
<n:TargetFlashRecoveryArea></n:TargetFlashRecoveryArea>
```

- Change the password for sys user using SQL*Plus to match the Password parameter in the XML file.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide shows an example of the input parameters for the rconfig XML file.

- SourceDBHome and TargetDBHome are same if the single-instance database is already running out of the RAC-enabled Oracle Home.
- Leave the TargetDatabaseArea and TargetFlashRecoveryArea parameters as empty if the database is already in the desired shared storage.
- Change the password of sys user to match <n>Password>oracle_4U</n>Password> in the XML file.

Note: The Convert verify option in the .xml file has three options:

- Convert verify="YES": rconfig performs checks to ensure that the prerequisites for single-instance to RAC conversion have been met before it starts conversion.
- Convert verify="NO": rconfig does not perform prerequisite checks, and starts conversion.
- Convert verify="ONLY": rconfig performs only prerequisite checks; it does not start conversion after completing the prerequisite checks.

Step 3: Perform Prerequisite Checks

- Run rconfig with Convert verify="ONLY":

```
$ rconfig ConvertToRAC_AdminManaged.xml

<?xml version="1.0" ?>
<RConfig version="1.1" >
<ConvertToRAC>
  <Convert>
    <Response>
      <Result code="0" >
        Operation Succeeded
      </Result>
    </Response>
    <ReturnValue type="object">
There is no return value for this step </ReturnValue>
  </Convert>
</ConvertToRAC></RConfig>
```

- The default listener must be configured in Grid Infrastructure Home.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Run rconfig with Convert verify="ONLY" to perform a test conversion to ensure that a conversion can be completed successfully.
- rconfig with Convert verify="ONLY" does not check if the default listener is configured in Grid Infrastructure Home or not. However, it is checked when running rconfig with Convert verify="YES". If the default listener is configured in Oracle Database Home, you will receive the following messages:

```
[oracle@enode01 sampleXMLs]$ rconfig ConvertToRAC_AdminManaged.xml
...
oracle.sysman.assistants.rconfig.engine.InvalidConfigurationException:
oracle.sysman.assistants.rconfig.engine.InvalidConfigurationException:
Default Listener is not configured in Grid Infrastructure Home.
Operation Failed. Refer logs at
/u01/app/oracle/cfgtoollogs/rconfig/rconfig_09_18_15_17_28_33.log for
more details.
...
```

Step 4: Convert to an Oracle RAC Database

- Run rconfig with Convert verify= "YES":

```
$ rconfig ConvertToRAC_AdminManaged.xml

Converting Database "eastdb" to Cluster Database. Target Oracle Home:
/u01/app/oracle/product/12.1.0/dbhome_1. Database Role: PRIMARY.
Setting Data Files and Control Files
Adding Database Instances
Adding Redo Logs
Enabling threads for all Database Instances
Setting TEMP tablespace
Adding UNDO tablespaces
Adding Trace files
Setting Fast Recovery Area
Updating Oratab
Creating Password file(s)
Configuring Listeners
Configuring related CRS resources
Starting Cluster Database

----- The Output Continued next page -----
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Step 4: Convert a Single Instance to an Oracle RAC Database

- Run rconfig with Convert verify= "YES" to start a conversion.

During the conversion, the rconfig utility performs the following operations:

- Setting data files and control files
- Adding database instances
- Adding redo logs
- Enabling threads for all database instances
- Setting TEMP tablespace
- Adding UNDO tablespaces
- Adding Trace files
- Setting Fast Recovery Area
- Updating Oratab
- Creating password file(s)
- Configuring listeners
- Configuring related CRS resources
- Starting cluster database

Step 4: Convert to an Oracle RAC Database

```
<?xml version="1.0" ?>
<RConfig version="1.1" >
<ConvertToRAC>
  <Convert>
    <Response>
      <Result code="0" >
        Operation Succeeded
      </Result>
    </Response>
    <ReturnValue type="object">
      <Oracle_Home>
        /u01/app/oracle/product/12.1.0/dbhome_1
      </Oracle_Home>
      <Database type="ADMIN_MANAGED" >
        <InstanceList>
          <Instance SID="eastdb1" Node="enode01" >
          </Instance>
          <Instance SID="eastdb2" Node="enode02" >
          </Instance>
        </InstanceList>
      </Database> </ReturnValue>
    </Convert>
  </ConvertToRAC></RConfig>
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Step 4: Convert a Single Instance to an Oracle RAC Database

- The output shows that the conversion has been completed successfully.

Step 5: Verify the Conversion

```
$ srvctl config database -db eastdb
Database unique name: eastdb
Database name: eastdb
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/eastdb/spfileeastdb.ora
Password file: +DATA/eastdb/orapweastdb
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: eastdb
Database instances: eastdb1,eastdb2
Disk Groups: DATA,FRA
Mount point paths:
Services:
Type: RAC
Database is administrator managed
```

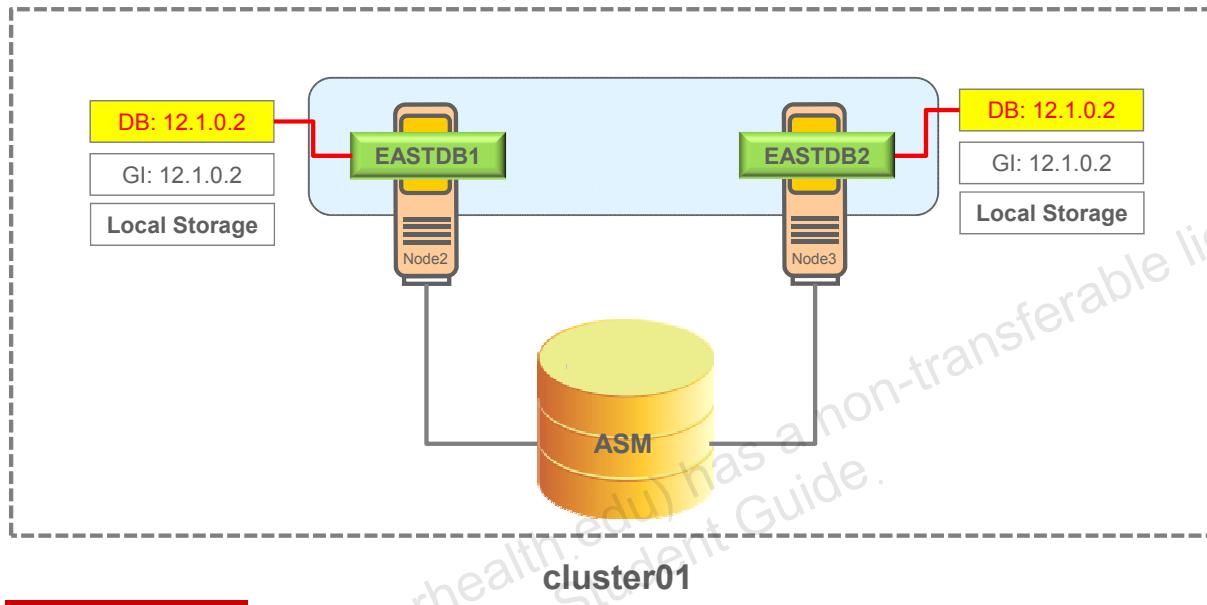


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The database type is now RAC, which indicates a RAC database.

Example: Result of Using rconfig

A single-instance database **in a *clustered* environment** has been successfully converted using the `rconfig` utility.



- A single-instance database in the clustered environment has been successfully converted to an Oracle RAC database without having to move the database files.

Quiz



Which of the following can be used to convert a single-instance database to a RAC database? (Choose the correct options.)

- a. rconfig
- b. netca
- c. dbca



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Choices a and c are correct.

Summary

In this lesson, you should have learned how to:

- Convert a single-instance database in a non-clustered environment to a RAC database
- Convert a single-instance database in a clustered environment to a RAC database



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 2: Overview

This practice covers the following topics:

- Converting a Single Instance on cluster01 to an Oracle RAC Database on cluster02 Using DBCA
- Dropping the Existing RAC Database on cluster02
- Converting a Single Instance on cluster01 to an Oracle RAC Database in the Same Cluster Using the Rconfig Utility



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

3

Configuring Oracle Net Services in a Data Guard Environment with RAC

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

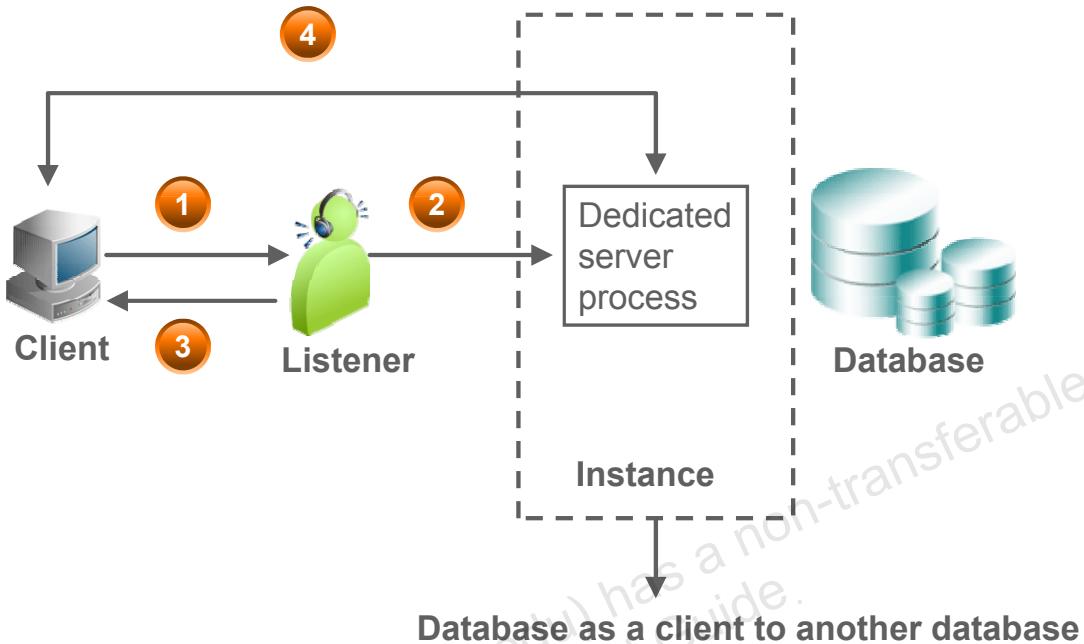
- Describe the basics of Oracle Net Services
- Configure the primary database and Oracle Net Services to support the creation of the physical standby database and role transition in a Data Guard environment with RAC



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Review of Oracle Net Services



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Net Services provides enterprise-wide connectivity solutions in distributed, heterogeneous computing environments. It enables a network session from a client application to an Oracle Database server.

A dedicated server process is a type of service handler that the listener starts when it receives a client request. The slide depicts the process of establishing a connection from a client to an Oracle database (non-shared server architecture) by using the following steps:

1. The listener receives a client connection request.
2. The listener starts a dedicated server process.
3. The listener provides the location of the dedicated server process in a redirect message.
4. The client connects directly to the dedicated server.

Note: Depending on the operating system and transport protocol, step three may be eliminated. In this case, the dedicated server process inherits the connection request from the listener. The result is the same—a network session is established between the client and the dedicated server process.

If the client and the database exist on the same computer, then the application initiating the session can spawn a dedicated server process without going through the listener. This is known as the bequeath protocol and is often used when starting a database instance.

Oracle Net Configuration Files

- sqlnet.ora
 - Contains information on how both Oracle server and Oracle client have to use Oracle Net capabilities for networked database access
 - On both Client and Server
- listener.ora
 - Is used to configure Oracle Net Listeners to accept remote connection requests
 - On Server only
- tnsnames.ora
 - Contains Connect Name to Descriptor mappings
 - On both Client and Server



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Net Services uses text-based configuration files stored on both the client and the database server to resolve network service names (connect identifiers) into detailed connect descriptors. Operating system environment variables determine the location of these files. The TNS_ADMIN variable is checked first. It allows the configuration files to be centrally located (for example, on a cluster file system and shared among many machines).

If the TNS_ADMIN variable is not defined, then the ORACLE_HOME variable is used to locate the configuration files. Several different ORACLE_HOME locations can be present on the same host machine for different reasons. Each may contain network configuration files if desired. For example, the tnsnames.ora and sqlnet.ora files could be found in the ORACLE_HOME location for each of the Grid Infrastructure software, database software, middleware software, and Enterprise Manager software products. The value of the ORACLE_HOME environment variable will point to a single ORACLE_HOME location at a time. The listener.ora configuration file is used only for database software ORACLE_HOME locations.

Note: In a RAC environment, the Oracle Database listener usually runs from the ORACLE_HOME of the Grid Infrastructure software and not the ORACLE_HOME of the database software. The ORACLE_HOME of the Grid Infrastructure software will need networking configuration if ASM is being deployed. If separation of duties exists between the cluster software administrator and the database administrator, the database listener could run from the ORACLE_HOME of the database software.

Single Client Access Name in RAC

- Single client access name (SCAN) is the address used by clients connecting to the cluster.
- The SCAN provides a stable, highly available name for clients to use, independent of the nodes that make up the cluster.

```
$ nslookup cluster01-scan.cluster01.example.com
Server:      192.0.2.1
Address:     192.0.2.1#53

Non-authoritative answer:
Name:   cluster01-scan.cluster01.example.com
Address: 192.0.2.243
Name:   cluster01-scan.cluster01.example.com
Address: 192.0.2.244
Name:   cluster01-scan.cluster01.example.com
Address: 192.0.2.245
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The single client access name (SCAN) is the address used by clients connecting to the cluster. The SCAN is a fully qualified host name (host name + domain) registered to three IP addresses. If you use Grid Naming Services (GNS) and you have DHCP support, then the GNS will assign addresses dynamically to the SCAN.

If you do not use GNS, the SCAN should be defined in the DNS to resolve to the three addresses assigned to that name. This should be done before you install Oracle Grid Infrastructure. The SCAN and its associated IP addresses provide a stable name for clients to use for connections, independent of the nodes that make up the cluster.

SCANS function like a cluster alias. However, SCANS are resolved on any node in the cluster, so unlike a virtual IP (VIP) address for a node, clients connecting to the SCAN no longer require updated VIP addresses as nodes are added to or removed from the cluster. Because the SCAN addresses resolve to the cluster, rather than to a node address in the cluster, nodes can be added to or removed from the cluster without affecting the SCAN address configuration.

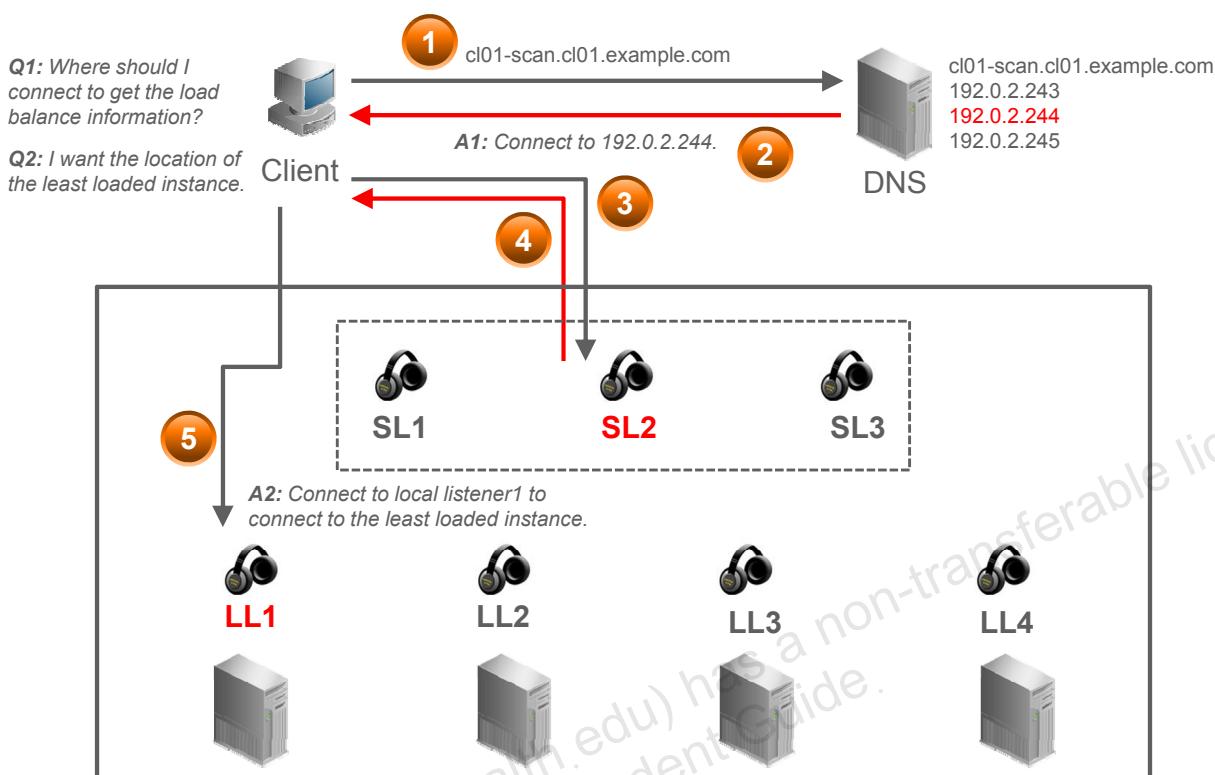
During installation, listeners are created on each node for the SCAN IP addresses. Oracle Clusterware routes application requests to the cluster SCAN to the least loaded instance providing the service.

SCAN listeners can run on any node in the cluster. SCANS provide location independence for databases so that the client configuration does not have to depend on which nodes run a particular database.

Instances register with SCAN listeners only as remote listeners. Upgraded databases register with SCAN listeners as remote listeners, and also continue to register with all other listeners.

If you specify a GNS domain during installation, the SCAN defaults to *clustername-scan.GNS_domain*. If a GNS domain is not specified at installation, the SCAN defaults to *clustername-scan.current_domain*.

Example: Client Database Connections in RAC



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a client requests a connection in a non-Grid Plug and Play (GPnP) environment:

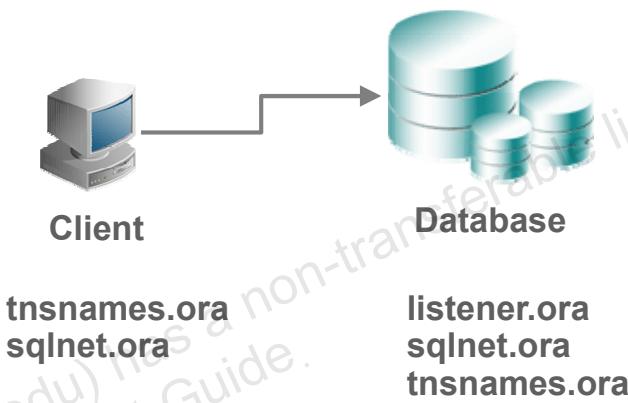
- The connection request with SCAN (`cl01-scan.cl01.example.com`) is forwarded.
- The DNS resolves the SCAN, returning the IP address (`192.0.2.244`) matching the name given; this address is then used by the client to contact the SCAN listener.
- The SCAN listener (`SL2`) uses its connection load balancing system to pick an appropriate local listener, whose name it returns to the client in an Oracle Net Redirect message.
- The client reconnects to the selected local listener (`LL1`).

The SCAN listeners must be known to all the database listener nodes and clients. The database instance nodes cross-register only with known SCAN listeners, also sending them per-service connection metrics.

Configuring Oracle Net Services in a Data Guard Environment

Configuration of Oracle Net Services is required for:

- RMAN duplication for standby
- Redo Transport Service
- Role Transition Service
- FAL_SERVER
- Data Guard broker
- Better performance
- More

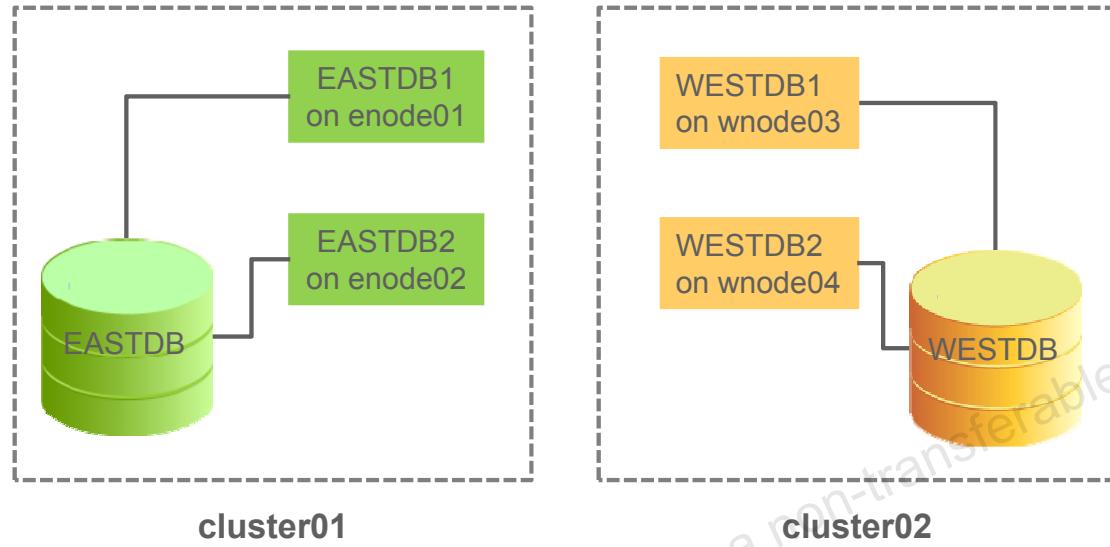


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You must configure Oracle Net Services in a Data Guard environment to support:

- RMAN duplication for standby
- Redo Transport Service
- Role Transition Service
- FAL_SERVER
- Data Guard broker
- Better performance
- More

Primary and Standby Databases with RAC



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram shown in the slide is an example of an Oracle Data Guard environment with RAC. The following slides will be based on this environment.

Example: tnsnames.ora on Primary Hosts

- The alias references the scan listener (cluster01-scan and cluster02-scan) and not the node-vip:

```
$ cat $ORACLE_HOME/network/admin/tnsnames.ora

eastdb =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = cluster01-scan)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = eastdb.example.com)))

westdb =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = cluster02-scan)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = westdb.example.com)))
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide shows an example of using the local naming method on the primary hosts.

- The Oracle net aliases called eastdb and westdb can be used to support the redo transport service, role transition service, fal_server, and Data Guard broker.

Example: tnsnames.ora on Standby Hosts

- The alias references the scan listener (cluster01-scan and cluster02-scan) and not the node-vip:

```
$ cat $ORACLE_HOME/network/admin/tnsnames.ora

eastdb =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = cluster01-scan)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = eastdb.example.com)))

westdb =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = cluster02-scan)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = westdb.example.com)))
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

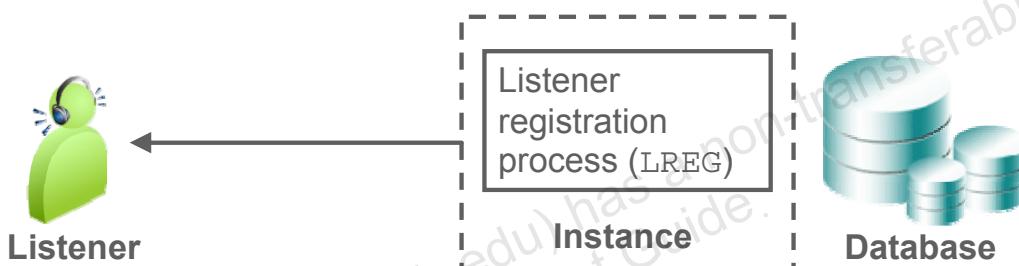
The slide shows an example of using the local naming method on the standby hosts.

- The Oracle net alias called `eastdb` and `westdb` can be used to support the redo transport service, role transition service, `fal_server`, and Data Guard broker.

Review of Dynamic Service Registration

Dynamic service registration allows a database instance to identify its available services to the listener. Available service names are determined by the following parameters:

- DB_UNIQUE_NAME, DB_NAME, and DB_DOMAIN
- SERVICE_NAMES
- INSTANCE_NAME
- LOCAL_LISTENER and REMOTE_LISTENER



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Dynamic service registration is configured in the database initialization file. It does not require any configuration in the `listener.ora` file. If not specified, the value for the `SERVICE_NAMES` parameter defaults to the global database name, a name comprised of the `DB_UNIQUE_NAME` and `DB_DOMAIN` parameters in the initialization parameter file. If not explicitly defined, the `DB_UNIQUE_NAME` parameter defaults to the value `DB_NAME`. The value for the `INSTANCE_NAME` parameter defaults to the Oracle system identifier (SID). All of these names can be explicitly defined to non-default values.

The `SERVICE_NAMES` parameter specifies one or more names by which clients can connect to the instance. The instance registers its service names with the listener. You can specify multiple service names to distinguish among different uses of the same database. For example:

```
SERVICE_NAMES=PROD,DG_PRMY,DG_RW,MAIN_REPORTING
```

By default, the listener registration (LREG) process registers service information with its local listener on the default local address of TCP/IP port 1521. To have the LREG process register with a local listener that does not use TCP/IP port 1521, configure the `LOCAL_LISTENER` parameter in the initialization parameter file to locate the local listener.

A remote listener is a listener residing on one computer that redirects connections to a database instance on another computer. You can configure registration to remote listeners using the `REMOTE_LISTENER` parameter.

Example: listener.ora on Primary Hosts

Static listener entries are needed for **Data Guard broker** operations:

```
[enode01]$ cat $ORACLE_HOME/network/admin/listener.ora
```

```
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(GLOBAL_DBNAME = eastdb_DGMGRL.example.com)
(ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
(SID_NAME = eastdb1)))
```

```
[enode02]$ cat $ORACLE_HOME/network/admin/listener.ora
```

```
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(GLOBAL_DBNAME = eastdb_DGMGRL.example.com)
(ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
(SID_NAME = eastdb2)))
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Dynamic service registration allows the LREG process of a database instance to identify its available services to the listener without entries in the `listener.ora` configuration file. The listener then acts as a port mapper for those services.

However, when the database instance is stopped, the listener discards all information for the dynamic services related to that database. Any attempt to establish a network session to the unknown service will usually receive the error message “ORA-12514: Listener does not currently know of service requested in connect descriptor.” Static registration allows the listener to know of a service, even if the database instance is not running. This is often important with tools and utilities that try to remotely start and stop a database instance.

To enable DGMGRL to restart instances during the course of broker operations, a static service must be registered with the local listener and assume a static service name of `db_unique_name_DGMGRL.db_domain`.

Note: Static “_DGMGRL” entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard broker configurations that are managed by Oracle Restart, RAC On Node or RAC as the broker will use the clusterware to restart an instance.

Example: listener.ora on Standby Hosts

Static listener entries are needed for **Data Guard broker** operations:

```
[wnode03]$ cat $ORACLE_HOME/network/admin/listener.ora
```

```
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(GLOBAL_DBNAME = westdb_DGMGRL.example.com)
(ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
(SID_NAME = westdb1)))
```

```
[wnode04]$ cat $ORACLE_HOME/network/admin/listener.ora
```

```
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(GLOBAL_DBNAME = westdb_DGMGRL.example.com)
(ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
(SID_NAME = westdb2)))
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To enable DGMGRL to restart instances during the course of broker operations, a static service must be registered with the local listener and assume a static service name of db_unique_name_DGMGRL.db_domain.

Note: Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard broker configurations that are managed by Oracle Restart, RAC On Node or RAC as the broker will use the clusterware to restart an instance.

Example: listener.ora on Standby Hosts

Static listener entries are needed for **Recovery Manager** operations:

```
[wnode03]$ cat $ORACLE_HOME/network/admin/listener.ora

SID_LIST_LISTENER_CLONE =
  (SID_LIST =
    (SID_DESC =
      (ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = westdb1)))

LISTENER_CLONE =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)
        (HOST = wnode03) (PORT = 1531) (IP = FIRST))))
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can configure a new listener (if necessary) or optionally update the default listener with a static service entry for your physical standby database. This entry is needed because the instance is shut down and restarted during standby database creation using RMAN.

Starting and Stopping the Node Listener

- Using the lsnrctl utility:

```
$ lsnrctl start listener
LSNRCTL for Linux: Version 12.1.0.1.0-Production on 31-JUL-2013 16:45:35
Copyright (c) 1991, 2013, Oracle. All rights reserved.
Starting /u01/app/12.1.0/grid/bin/tnslsnr: please wait.....
Intermediate output removed ...
The command completed successfully
$
```

- Using the srvctl utility (preferred):

```
$ srvctl start listener -n host01
$
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A standard clustered ASM installation configures an Oracle network listener under the Grid home directory. This listener can be manually started and stopped using the lsnrctl utility installed as part of the ASM installation:

```
$ lsnrctl start listener
$ lsnrctl stop listener
```

You can alternatively use the Server Control utility (srvctl) to start and stop the ASM listener as follows:

```
$ srvctl start listener -n <node>
$ srvctl stop listener -n <node>
```

The lsnrctl and srvctl utilities exist in both the *Grid_home* and RDBMS home directories; which one you use depends on where the listener configuration files reside. The Grid Infrastructure installation will start a listener with the configuration files in the *Grid_home*. By default, the database installation will use that listener. In this case, set the ORACLE_HOME and PATH environment variables to use *Grid_home* and then run the utilities.

If you create a new listener with configuration files in the RDBMS home, set ORACLE_HOME and PATH environment variables to use the RDBMS home and then run the utilities.

Quiz



Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard broker configurations that are managed by Oracle Restart, RAC On Node or RAC.

- a. True
- b. False



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe the basics of Oracle Net Services
- Configure the primary database and Oracle Net Services to support the creation of the physical standby database and role transition in a Data Guard environment with RAC



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 3: Overview

This practice covers the following topics:

- Modifying the tnsnames.ora Configuration File on the Primary and Standby Hosts
- Understanding Listener Management in an Oracle RAC Environment
- Modifying the listener.ora Configuration File on the Primary and Standby Hosts



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

4

Deploying a Physical Standby Database in an Oracle RAC Environment

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Prepare the primary database
- Configure Oracle Net Services
- Prepare the standby hosts
- Start the standby database instance
- Execute the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN command
- Complete the RAC configuration
- Start the transport and application of redo



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Creating a Physical Standby Database in RAC

- Task 1: Preparing the Primary Database
- Task 2: Configuring Oracle Net Services
- Task 3: Preparing the Standby Hosts
- Task 4: Starting the Standby Database Instance
- Task 5: Executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN Command
- Task 6: Completing the RAC Configuration
- Task 7: Starting the Transport and Application of Redo



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You perform the steps listed in the slide when using SQL and RMAN commands to create a physical standby database. Detailed information about each step is provided in the remaining slides of the lesson.

Note: See *Oracle Data Guard Concepts and Administration* and Oracle Support Note titled "*Creating a Standby using RMAN duplicate (RAC or Non-RAC, Doc ID 1617946.1)*" for detailed information about creating a physical standby database by using SQL and RMAN commands.

Task 1: ARCHIVELOG Mode in RAC

- Shut down all primary instances and mount one instance:

```
$ srvctl stop database -db eastdb  
$ srvctl start instance -db eastdb -instance eastdb1  
-startoption mount
```

- Configure the primary database in archive log mode:

```
SQL> ALTER DATABASE ARCHIVELOG;
```

- Restart all primary instances:

```
$ srvctl stop instance -db eastdb -instance eastdb1  
$ srvctl start database -db eastdb
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If archiving is not enabled, issue the `ALTER DATABASE ARCHIVELOG` command to put the primary database in ARCHIVELOG mode and enable automatic archiving. See the *Oracle Database Administrator's Guide* for additional information about archiving.

Task 1: FORCE LOGGING Mode

- FORCE LOGGING mode is recommended to ensure data consistency.
- FORCE LOGGING forces redo to be generated even when NOLOGGING operations are executed.
- Temporary tablespaces and temporary segments are not logged.
- FORCE LOGGING is recommended for both physical and logical standby databases.
- Issue the following command on the primary database:

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



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

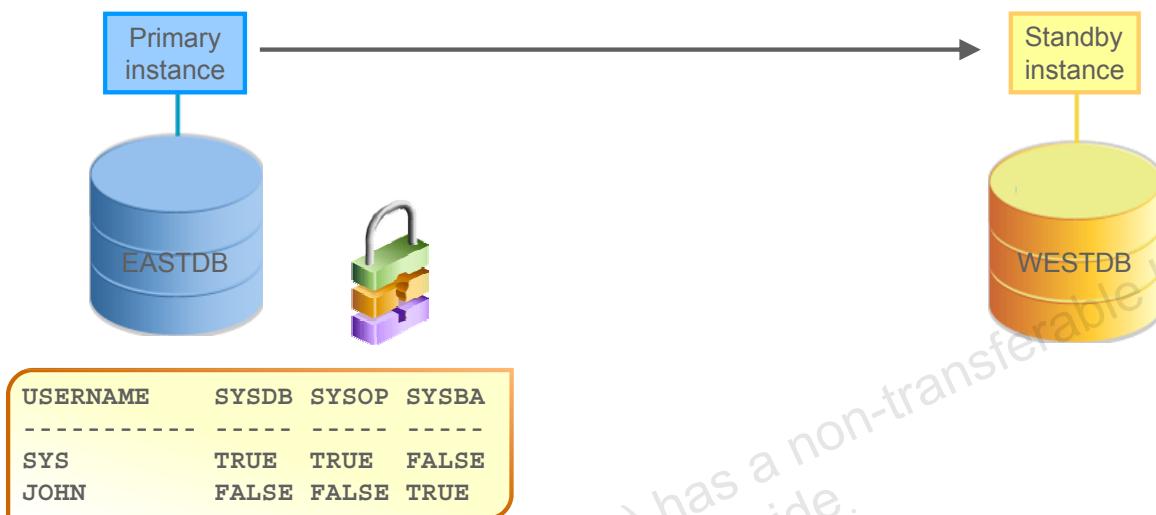
FORCE LOGGING mode determines whether the Oracle database server logs all changes in the database (except for changes to temporary tablespaces and temporary segments). The [NO] FORCE LOGGING clause of the ALTER DATABASE command contains the following settings:

- **FORCE LOGGING:** This setting takes precedence over (and is independent of) any NOLOGGING or FORCE LOGGING settings that you specify for individual tablespaces and any NOLOGGING setting that you specify for individual database objects. All ongoing, unlogged operations must finish before forced logging can begin.
- **NOFORCE LOGGING:** Places the database in NOFORCE LOGGING mode. This is the default.

The FORCE_LOGGING column in V\$DATABASE contains a value of YES if the database is in FORCE LOGGING mode.

Task 1: Creating a Password File

- Create a password file if required.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unless you have configured Oracle Advanced Security and public key infrastructure (PKI) certificates, every database in a Data Guard configuration must use a password file, and the password for the `SYS` user must be identical on every system for redo data transmission to succeed. For details about creating a password file, see the *Oracle Database Administrator's Guide*.

Task 1: Configuring Standby Redo Logs

- Create standby redo logs on the primary database initially (recommended).
- Create standby redo logs using the same file size as the primary database online redo logs.
- Create one additional group more than the number of online redo log groups.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A standby redo log is used only when the database is in the standby role to store redo data received from the primary database. Standby redo logs form a separate pool of log file groups. Configuring standby redo log files is highly recommended for all databases in a Data Guard configuration to aid in role reversal.

You should create at least one more standby redo log group than you have online redo log groups in the primary database. In addition, each standby redo log file must be at least as large as the largest redo log file in the redo log of the redo source database.

A standby redo log is required to implement the following:

- Synchronous transport mode
- Real-time apply
- Cascaded redo log destinations

Note: By configuring the standby redo logs on the primary database, the standby redo logs are created automatically on the standby database when you execute the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN command.

Example: Configuring Standby Redo Logs for RAC

- Oracle recommends having the same number of standby redo logs for each thread plus one additional standby redo log:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1
GROUP 5 '+DATA' SIZE 50M
GROUP 6 '+DATA' SIZE 50M
GROUP 7 '+DATA' SIZE 50M;

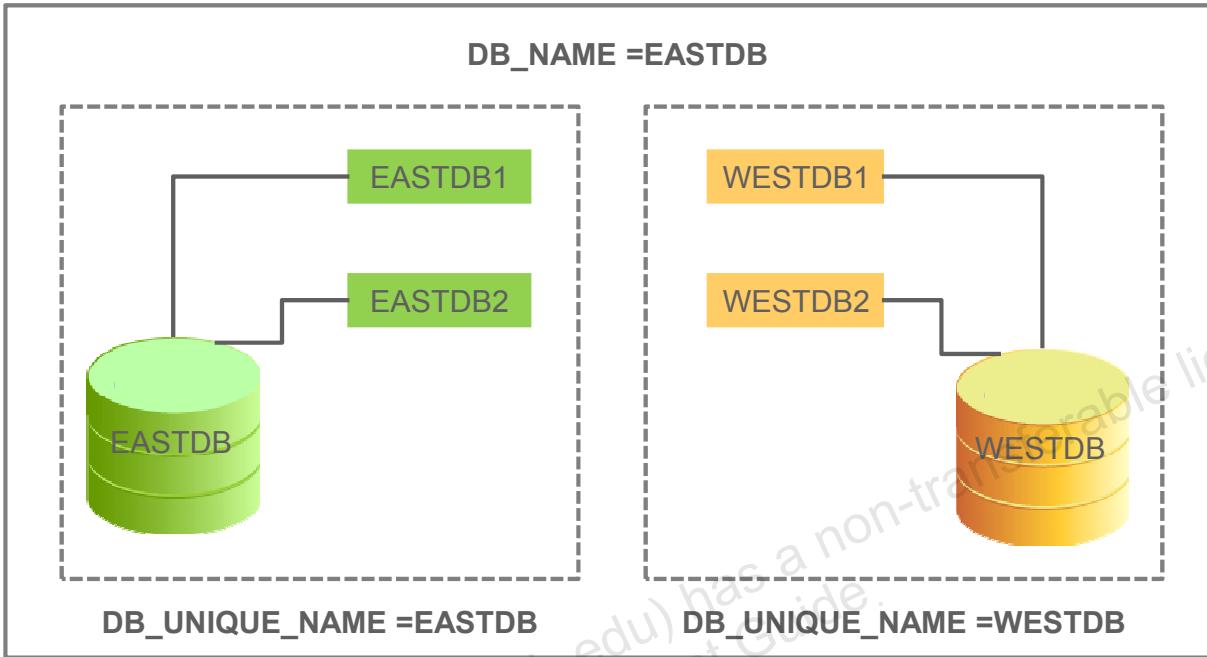
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 2
GROUP 8 '+DATA' SIZE 50M
GROUP 9 '+DATA' SIZE 50M
GROUP 10 '+DATA' SIZE 50M;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows how to configure standby redo logs in a Data Guard environment with RAC. Oracle recommends having the same number of standby redo logs for each thread plus one additional standby redo log.

Task 1: Setting Initialization Parameters



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

On the primary database, you define initialization parameters that control redo transport services while the database is in the primary role. There are additional parameters you need to add that control the receipt of the redo data and apply services where the primary database is transitioned to the standby role.

The diagram shown in the slide is an example of the RAC-RAC Data Guard configuration, which will be used for the rest of this lesson.

Setting Initialization Parameters on the Primary Database to Control Redo Transport

Parameter Name	Description
LOG_ARCHIVE_CONFIG	Specifies the unique database name for each database in the configuration Enables or disables sending and receiving of redo
LOG_ARCHIVE_DEST_n	Controls redo transport services
LOG_ARCHIVE_DEST_STATE_n	Specifies the destination state
ARCHIVE_LAG_TARGET	Forces a log switch after the specified number of seconds
LOG_ARCHIVE_TRACE	Controls output generated by the archiver process



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

On the primary database, you define initialization parameters that control redo transport services while the database is in the primary role. These parameters are described in more detail in the following slides.

Setting LOG_ARCHIVE_CONFIG

Specify the DG_CONFIG attribute to list the DB_UNIQUE_NAME for the primary database and each standby database in the Data Guard configuration:

```
LOG_ARCHIVE_CONFIG='DG_CONFIG=(eastdb,westdb)'
```



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Specify the DG_CONFIG attribute of the LOG_ARCHIVE_CONFIG parameter to list the DB_UNIQUE_NAME of the primary and standby databases in the Data Guard configuration. By default, the LOG_ARCHIVE_CONFIG parameter enables the database to send and receive redo. The LOG_ARCHIVE_CONFIG parameter can be used to disable the sending of redo logs to remote destinations or disable the receipt of remote redo logs. The complete syntax for the LOG_ARCHIVE_CONFIG parameter is as follows:

```
LOG_ARCHIVE_CONFIG = {
  [ SEND | NOSEND ] [ RECEIVE | NORCEIVE ]
  [ DG_CONFIG=(remote_db_unique_name1
  [, ... remote_db_unique_name9) | NODG_CONFIG ] }
```

Use the V\$DATAGUARD_CONFIG view to see the unique database names defined with the DB_UNIQUE_NAME and LOG_ARCHIVE_CONFIG initialization parameters; you can thus view the Data Guard environment from any database in the configuration. The first row of the view lists the unique database name of the current database that was specified with the DB_UNIQUE_NAME initialization parameter. Additional rows reflect the unique database names of the other databases in the configuration that were specified with the DG_CONFIG keyword of the LOG_ARCHIVE_CONFIG initialization parameter.

Setting LOG_ARCHIVE_DEST_n

- Specify LOG_ARCHIVE_DEST_n parameters for:
 - Local archiving
 - Standby database location
- Include (at a minimum) one of the following:
 - LOCATION: Specifies a valid path name
 - SERVICE: Specifies a valid Oracle Net Services name referencing a standby database
- Include a LOG_ARCHIVE_DEST_STATE_n parameter for each defined destination.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

By using the various LOG_ARCHIVE_DEST_n attributes, you define most of the settings for the Data Guard configuration. The Redo Transport Service is directly controlled by these settings. Several different attributes can be set for each LOG_ARCHIVE_DEST_n parameter. Most have defaults that are adequate for most configurations. See *Oracle Data Guard Concepts and Administration* for a complete list and a description of each attribute.

You should specify a LOG_ARCHIVE_DEST_n parameter (where n is an integer from 1 to 31) for the local archiving destination and one for the standby location. In previous versions of Oracle Database, LOG_ARCHIVE_DEST_10 was set to USE_DB_RECOVERY_FILE_DEST when the fast recovery area was being used. Beginning with Oracle Database 11g Release 2, you must manually set a location to use the fast recovery area. Query the V\$ARCHIVE_DEST view to see current settings of the LOG_ARCHIVE_DEST_n initialization parameter.

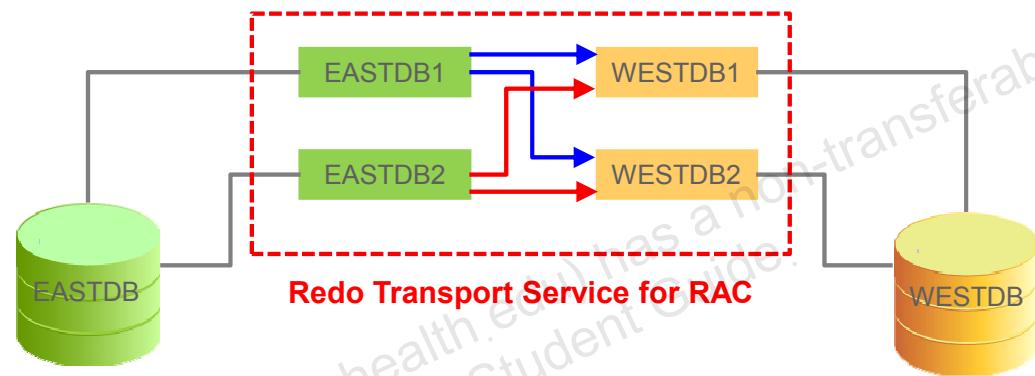
All defined LOG_ARCHIVE_DEST_n parameters must contain, at a minimum, either a LOCATION attribute or a SERVICE attribute.

In addition, you must have a LOG_ARCHIVE_DEST_STATE_n parameter for each defined destination. LOG_ARCHIVE_DEST_STATE_n defaults to ENABLE.

Example: LOG_ARCHIVE_DEST_n

- Example: Configure the Redo Transport Service for RAC

```
LOG_ARCHIVE_DEST_2=
  'SERVICE=westdb
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=westdb'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Setting Initialization Parameters on the Primary Database

- Specify parameters when standby databases have disk or directory structures that differ from the primary database.
- Use parameters when the primary database is transitioned to a standby database.

Parameter Name	Description
DB_FILE_NAME_CONVERT	Converts primary database file names
LOG_FILE_NAME_CONVERT	Converts primary database log file names
STANDBY_FILE_MANAGEMENT	Controls automatic standby file management
FAL_SERVER	Specifies the fetch archive log server for a standby database



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The parameters listed in the slide are required if the disk configuration is not the same for the primary and standby databases. The parameters are also applicable when the primary database is transitioned to a standby database.

Specifying Values for DB_FILE_NAME_CONVERT

- DB_FILE_NAME_CONVERT must be defined on standby databases that have different disk or directory structures from the primary database.
- Multiple pairs of file names can be listed in the DB_FILE_NAME_CONVERT parameter.
- DB_FILE_NAME_CONVERT applies only to a physical standby database.
- DB_FILE_NAME_CONVERT can be set in the DUPLICATE RMAN script.

```
DB_FILE_NAME_CONVERT = ('+DATA', '+SBDATA')
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When files are added to the standby database, the DB_FILE_NAME_CONVERT parameter is used to convert the data file name on the primary database to a data file name on the standby database. The file must exist and be writable on the physical standby database; if it is not, the recovery process halts with an error.

You specify the path name and file name location of the primary database data files followed by the standby location by setting the value of this parameter to two strings. The first string is the pattern found in the data file names on the primary database. The second string is the pattern found in the data file names on the physical standby database. You can use as many pairs of primary and standby replacement strings as required. You can use single or double quotation marks. Parentheses are optional.

In the example in the slide, +DATA is used to match file names coming from the primary database. +SBDATA is the corresponding strings for the physical standby database.

Multiple pairs can be specified such as ('a', 'b', '1', '2').

Specifying Values for LOG_FILE_NAME_CONVERT

- LOG_FILE_NAME_CONVERT is similar to DB_FILE_NAME_CONVERT.
- LOG_FILE_NAME_CONVERT must be defined on standby databases that have different disk or directory structures from the primary database.
- LOG_FILE_NAME_CONVERT applies only to a physical standby database.
- LOG_FILE_NAME_CONVERT can be set in the DUPLICATE RMAN script.

```
LOG_FILE_NAME_CONVERT = ('+DATA', '+SBDATA')
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The LOG_FILE_NAME_CONVERT parameter is used to convert the name of a redo log file on the primary database to the name of a redo log file on the standby database. Adding a redo log file to the primary database requires adding a corresponding file to the standby database. When the standby database is updated, this parameter is used to convert the log file name from the primary database to the log file name on the standby database. This parameter is required if the standby database is on the same system as the primary database or on a separate system that uses different path names.

Specify the location of the primary database online redo log files followed by the standby location. The use of parentheses is optional.

Both DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT parameters perform simple string substitutions. For example, ('a', 'b') will transform the following:

```
/disk1/primary/mya/a.dbf into  
/disk1/primbry/myb/b.dbf
```

Multiple pairs can be specified such as ('a', 'b', '1', '2').

Specifying a Value for STANDBY_FILE_MANAGEMENT

- STANDBY_FILE_MANAGEMENT is used to maintain consistency when you add or delete a data file on the primary database.
 - MANUAL (default)
 - Data files must be manually added to the standby database.
 - AUTO
 - Data files are automatically added to the standby database.
 - Certain ALTER statements are no longer allowed on the standby database.
- STANDBY_FILE_MANAGEMENT applies to physical standby databases only, but can be set on a primary database for role changes.

STANDBY_FILE_MANAGEMENT = AUTO



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When STANDBY_FILE_MANAGEMENT is set to AUTO, you cannot execute the following commands on the standby database:

- ALTER DATABASE RENAME
- ALTER DATABASE ADD/DROP LOGFILE [MEMBER]
- ALTER DATABASE ADD/DROP STANDBY LOGFILE MEMBER
- ALTER DATABASE CREATE DATAFILE AS . . .

When you add a log file to the primary database and want to add it to the physical standby database as well (or when you drop a log file from the primary and want to drop it from the physical), you must do the following:

1. Set STANDBY_FILE_MANAGEMENT to MANUAL on the physical standby database.
2. Add the redo log files to (or drop them from) the primary database.
3. Add them to (or drop them from) the standby database.
4. Reset to AUTO afterward on the standby database.

Specifying a Value for FAL_SERVER

- FAL_SERVER is:
 - Used to specify the name of the fetch archive log server, usually the primary database
 - Used only by databases in the standby role
 - Created on both primary and standby databases for role reversal
- On the eastdb primary database:

```
FAL_SERVER = westdb
```

- On the westdb standby database:

```
FAL_SERVER = eastdb
```



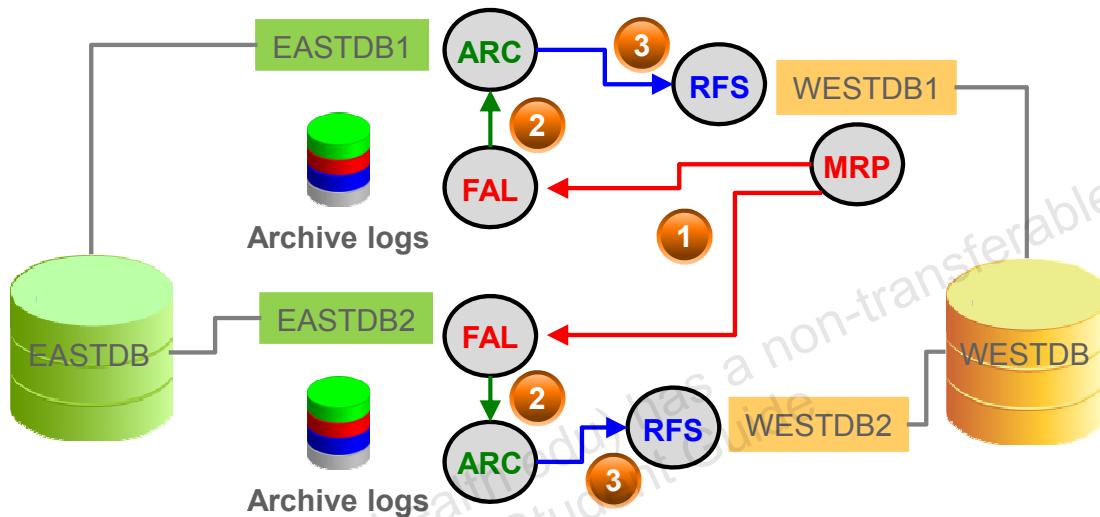
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The FAL_SERVER parameter is used to specify the Oracle Net service name of the fetch archive log (FAL) server (typically this is the database running in the primary role). When the westdb database is running in the standby role, it uses the eastdb database as the FAL server from which to fetch (request) missing archived redo log files if eastdb is unable to automatically send the missing log files. It is assumed that the Oracle Net service name is configured properly on the standby database system to point to the desired FAL server.

Example: FAL_SERVER for RAC

- Example: FAL_SERVER for the RAC primary database

FAL_SERVER = eastdb



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates how the FAL_SERVER parameter is used in a RAC environment. When there is a communication loss between the primary and standby databases, log shipping is affected and there may be gaps.

- If the current primary database instances (EASTDB1 and EASTDB2) are unable to automatically send the missing log files after the communication issue is resolved, the managed recovery process (MRP) (FAL Client) running in the apply instance (WESTDB1) uses the FAL servers in primary instances (EASTDB1 and EASTDB2) to request the missing archived redo log files.
- The Archiver (ARC) processes running in the primary instances (EASTDB1 and EASTDB2) transmit the requested log files, which are the missing log files to resolve the redo gap issue.
- The Remote File Server (RFS) processes receive the missing log from the primary instances and directly write into archive redo logs.

Example: Setting Initialization Parameters on the Primary Database

```
DB_NAME=eastdb
DB_UNIQUE_NAME=eastdb
LOG_ARCHIVE_CONFIG='DG_CONFIG=(eastdb,westdb)'
CONTROL_FILES='+DATA','+FRA'
LOG_ARCHIVE_DEST_2=
'SERVICE=westdb
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=westdb'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
LOG_ARCHIVE_FORMAT=arch_%t_%s_%r.log
FAL_SERVER=westdb
STANDBY_FILE_MANAGEMENT=auto
DB_FILE_NAME_CONVERT='+SBDATA','+DATA'
LOG_FILE_NAME_CONVERT='+SBDATA','+DATA'
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, assume that the primary database is named `eastdb` and the standby is named `westdb`. For each database, an Oracle Net Services name is defined.

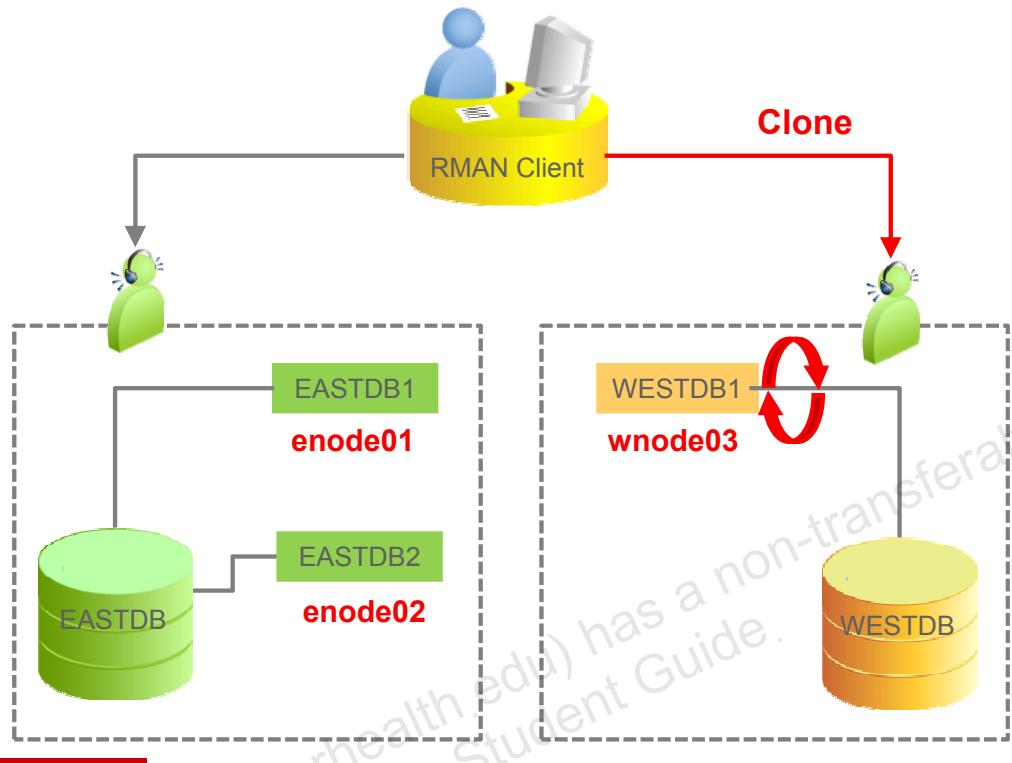
Creating a Physical Standby Database in RAC

- Task 1: Preparing the Primary Database
- Task 2: Configuring Oracle Net Services
- Task 3: Preparing the Standby Hosts
- Task 4: Starting the Standby Database Instance
- Task 5: Executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN Command
- Task 6: Completing the RAC Configuration
- Task 7: Starting the Transport and Application of Redo



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Task 2: Configuring Oracle Net Services



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The following slides show the entries in the `tnsnames.ora` and `listener.ora` files to be used when invoking RMAN and executing the `DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE` command.

Task 2: Configuring a Temporary Oracle Net Service for Your Standby Database

- Update the `tnsnames.ora` file for the initial standby database creation:

```
clone =
  (DESCRIPTION =
    (ADDRESS =
      (PROTOCOL = TCP) (HOST = wnode03) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED) (SID = westdb1)))
```

- Configure an entry for your standby database in the `listener.ora` file:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = clone)
      (ORACLE_HOME = /u01/app/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = westdb1)))
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This TNS entry in the slide is used to connect to the standby database when invoking RMAN and executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE command. This entry can point to the new temporary listener or default listener depending on your listener configuration.

Note: Make sure that the alias references the local listener on wnode03 where the physical standby instance will be running initially. According to Oracle Support Note (1144273.1), even when executing the RMAN DUPLICATE command in the standby host, you will still need the same alias in the primary database.

The second example is to configure a new listener (if necessary) or optionally to update the default listener with a static service entry for your physical standby database. This entry is needed because the instance is shut down and restarted during the standby database creation using RMAN.

Creating a Physical Standby Database in RAC

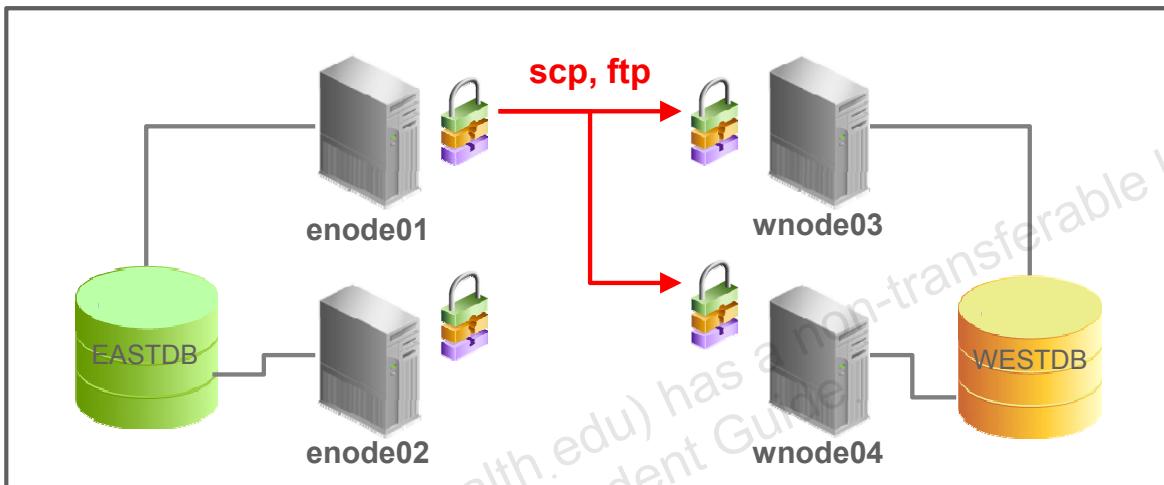
- Task 1: Preparing the Primary Database
- Task 2: Configuring Oracle Net Services
- **Task 3: Preparing the Standby Hosts**
- Task 4: Starting the Standby Database Instance
- Task 5: Executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN Command
- Task 6: Completing the RAC Configuration
- Task 7: Starting the Transport and Application of Redo



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Task 3: Copying the Password File to the Standby Hosts

- Copy the primary database password file to the \$ORACLE_HOME/dbs directory on the standby database hosts and rename the file for your standby database: orapw<SID>.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can create a password file for your physical standby database by copying the primary database password file to the physical standby database hosts and renaming it.

Note: You can use the `orapwd` utility to create a password file, but that technique should always be avoided with Data Guard. The Recovery Manager `DUPLICATE DATABASE` command is being used in this lesson and RMAN will automatically copy the password file from the primary database and replace the one that was created.

Task 3: Creating Directories for the Physical Standby Database

- On all standby hosts, create the audit directory for the `westdb` database:

```
[wnode03]$ mkdir -p /u01/app/oracle/admin/westdb/adump  
[wnode04]$ mkdir -p /u01/app/oracle/admin/westdb/adump
```

- The directories to be created depend on whether the primary database is using file locations for data files or Automatic Storage Manager.
- If multi-tenant architecture is being used, additional directories are needed.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Create an initial directory structure for the physical standby database such as the audit directory. The directories to be created depend on whether the primary database is using file locations for data files or Automatic Storage Manager. If multi-tenant architecture is being used, additional directories are needed.

Creating a Physical Standby Database in RAC

- Task 1: Preparing the Primary Database
- Task 2: Configuring Oracle Net Services
- Task 3: Preparing the Standby Hosts
- **Task 4: Starting the Standby Database Instance**
- Task 5: Executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN Command
- Task 6: Completing the RAC Configuration
- Task 7: Starting the Transport and Application of Redo



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Task 4: Creating a Temporary Initialization Parameter File

- Create an initialization parameter file (PFILE) containing enough information to match the network listener service entries:

```
[wnode03]$ vi $ORACLE_HOME/dbs/initwestdb.ora
```

```
DB_NAME=eastdb  
DB_UNIQUE_NAME=westdb  
DB_DOMAIN=example.com
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Create a text initialization parameter file containing only the DB_NAME, DB_UNIQUE_NAME, and DB_DOMAIN initialization parameters.

This initialization parameter file is used to start the physical standby database in NOMOUNT mode, before the execution of the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN command. When you execute this command, RMAN creates a server parameter file for the standby database.

Task 4: Starting the Physical Standby Database

Start the physical standby database instance in NOMOUNT mode:

```
SQL> startup nomount  
pfile=$ORACLE_HOME/dbs/initwestdb.ora
```

```
ORACLE instance started.
```

Total System Global Area	150667264 bytes
Fixed Size	1298472 bytes
Variable Size	92278744 bytes
Database Buffers	50331648 bytes
Redo Buffers	6758400 bytes



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Set the ORACLE_SID environment variable to your physical standby database. Start the physical standby database instance in NOMOUNT mode by using the text initialization parameter file. With ASM installed, there will be multiple software home locations on each machine. This will require that the ORACLE_HOME and PATH location change accordingly. Oracle recommends the oraenv utility to change environment variables, provided entries exist in the /etc/oratab file. The oraenv utility will adjust ORACLE_SID, ORACLE_BASE, ORACLE_HOME, PATH, and LD_LIBRARY_PATH environment variables. The ORACLE_HOME variable should point to the Grid Infrastructure software directories when starting the listener by using the LSNRCTL utility. However, the ORACLE_HOME variable should point to the database software directories when starting the database.

Note: Because the initialization parameter file contains only entries for DB_NAME and DB_DOMAIN, memory sizes for the System Global Area will use default values. Later the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN command will copy the initialization parameter values for memory sizing from the primary database configuration.

Creating a Physical Standby Database in RAC

- Task 1: Preparing the Primary Database
- Task 2: Configuring Oracle Net Services
- Task 3: Preparing the Standby Hosts
- Task 4: Starting the Standby Database Instance
- **Task 5: Executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN Command**
- Task 6: Completing the RAC Configuration
- Task 7: Starting the Transport and Application of Redo



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Step 5: Creating an RMAN Script

- Create an RMAN script to create the physical standby database:

```
run {
    allocate channel prmy1 type disk;
    allocate channel prmy2 type disk;
    allocate channel prmy3 type disk;
    allocate channel prmy4 type disk;
    allocate auxiliary channel stby type disk;

    duplicate target database for standby
        from active database
```

Note: The script continues in the next slide.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Create an RMAN script containing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE command.

There are several advantages to using RMAN to create the standby database. They include:

- RMAN can create a standby database by copying the files currently in use by the primary database. No backups are required.
- RMAN can create a standby database by restoring backups of the primary database to the standby site. Thus, the primary database is not affected during the creation of the standby database.
- RMAN automates renaming of files, including Oracle Managed Files (OMF) and directory structures.
- RMAN restores archived redo log files from backups and performs media recovery so that the standby and primary databases are synchronized.

Note: You can use the CONFIGURE ... PARALLELISM *integer* command to configure automatic channels for the specified device type. For additional information, see the *Oracle Database Backup and Recovery Reference*.

Example: RMAN Script 1

- Duplicate between two systems where the diskgroup names are the same:

```
spfile
set cluster_database='false'
set db_unique_name='westdb'
set remote_listener='cluster02-scan:1521'
set control_files='+DATA', '+FRA'
set fal_server='eastdb'
set audit_file_dest='/u01/app/oracle/admin/westdb/adump'
nofilenamecheck;
allocate auxiliary channel stby type disk;
sql channel stby "alter database recover managed
standby database disconnect";
}
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In the RMAN script, specify the settings for the physical standby initialization parameters. The example in the slide illustrates how to duplicate between two systems where the diskgroup names are the same.

- CLUSTER_DATABASE must be initially disabled because the RMAN duplication works with only one node.
- REMOTE_LISTENER is required to register the database with the SCAN listeners.

Example: RMAN Script 2

- Duplicate between two systems where the diskgroup names are different:

```
spfile
parameter_value_convert '+DATA','+SBDATA','+FRA','+SBFRA'
set cluster_database='false'
set db_unique_name='westdb'
set remote_listener='cluster02-scan:1521'
set control_files='+SBDATA','+SBFRA'
set fal_server='eastdb'
set db_file_name_convert='+DATA','+SBDATA',
set log_file_name_convert='+DATA','+SBDATA','+FRA','+SBFRA'
nofilenamecheck;
allocate auxiliary channel stby type disk;
sql channel stby "alter database recover managed standby
database disconnect";
}
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In the RMAN script, specify the settings for the physical standby initialization parameters. The example in the slide illustrates how to duplicate between two systems where the diskgroup names are different.

- PARAMETER_VALUE_CONVERT specifies conversion strings for initialization parameters whose values specify path names, with the exception except the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT parameters. The primary purpose of PARAMETER_VALUE_CONVERT is so that you can set a collection of initialization parameters and avoid explicitly setting them one by one.
Note: PARAMETER_VALUE_CONVERT can update all string values, not just those containing path names. The values are case-sensitive.
- CLUSTER_DATABASE must be initially disabled because the RMAN duplication works with only one node.
- REMOTE_LISTENER is required to register the database with the SCAN listeners.
- DB_FILE_NAME_CONVERT specifies a rule for creating the filenames for duplicate datafiles and tempfiles. Note that DB_FILE_NAME_CONVERT specified on the DUPLICATE command overrides the initialization parameter DB_FILE_NAME_CONVERT if it is set in the initialization parameter file.
- LOG_FILE_NAME_CONVERT specifies a rule for the online redo logs.

Example: RMAN Script 3

- Duplicate between two systems where the source is on a file system and the target is using a diskgroup:

```
spfile
parameter_value_convert '/u01/data','+DATA','/u01/fra','+FRA'
set remote_listener='cluster02-scan:1521'
set cluster_database='false'
set db_unique_name='westdb'
set control_files='+DATA','+FRA'
set fal_server='eastdb'
set db_file_name_convert='/u01/app','+DATA',
set log_file_name_convert='/u01/data','+DATA','/u01/fra','+FRA'
nofilenamecheck;
allocate auxiliary channel stby type disk;
sql channel stby "alter database recover managed standby
database disconnect";
}
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In the RMAN script, specify the settings for the physical standby initialization parameters. The example in the slide illustrates how to duplicate between two systems where the source is on a file system and the target is using a diskgroup.

Step 5: Creating the Physical Standby Database

- Invoke RMAN and connect to the primary database and the physical standby database.
- Execute the RMAN script (previous two slides) to create the physical standby database.

```
RMAN> connect target sys/oracle_4U@eastdb
RMAN> connect auxiliary sys/oracle_4U@clone
RMAN> @cr_phys_standby
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Connect to the primary database instance (target) and the physical standby database instance (auxiliary). Execute the script that you created. The script can be run using the RMAN utility on either the primary database or the standby database.

Creating a Physical Standby Database in RAC

- Task 1: Preparing the Primary Database
- Task 2: Configuring Oracle Net Services
- Task 3: Preparing the Standby Hosts
- Task 4: Starting the Standby Database Instance
- Task 5: Executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN Command
- Task 6: Completing the RAC Configuration
- Task 7: Starting the Transport and Application of Redo



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Step 6: Creating an Spfile on the Standby Database

- Create a temporary pfile from spfile on the standby database:

```
SQL> create pfile='/tmp/init.ora' from spfile;
```

- Modify the temporary pfile to match the standby instance names.
- Create an spfile in a shared storage for the standby database:

```
SQL> create spfile='+DATA/westdb/spfilewestdb.ora' from  
pfile='/tmp/init.ora';
```

- On all standby hosts, create the `initwestdb1.ora` and `initwestdb2.ora` files that point to the spfile created in the previous step.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If the primary database was RAC, then a few configuration items need to be completed to finish the RAC configuration on the standby database.

These remaining configuration steps are:

- Create an spfile on the standby database.
- Register the RAC standby database with Cluster Ready Services (CRS).

Step 6: Registering the RAC Standby Database with CRS

- Register the RAC standby database with CRS:

```
$ srvctl add database -db westdb  
-oraclehome /u01/app/oracle/product/12.1.0/dbhome_1  
-dbtype RAC -spfile '+DATA/westdb/spfilewestdb.ora'  
-role physical_standby -diskgroup "DATA,FRA"  
-dbname eastdb -domain example.com  
$ srvctl add instance -db westdb -instance westdb1 -node wnode03  
$ srvctl add instance -db westdb -instance westdb2 -node wnode04
```

- Restart the standby instances using the newly created spfile:

```
$ srvctl start database -db westdb -startoption mount
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows how to register the RAC standby database with CRS. Then, restart the standby database using the newly created spfile.

Step 6: Verify the RAC Standby Database

```
$ srvctl config database -db westdb
Database unique name: westdb
Database name: eastdb
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/westdb/spfilwestdb.ora
Password file:
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PHYSICAL STANDBY
Management policy: AUTOMATIC
Server pools:
Database instances:
Disk Groups:
Mount point paths:
Services:
Type: RAC
...
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows the configuration of the 2-node RAC standby database.

Creating a Physical Standby Database in RAC

- Task 1: Preparing the Primary Database
- Task 2: Configuring Oracle Net Services
- Task 3: Preparing the Standby Hosts
- Task 4: Starting the Standby Database Instance
- Task 5: Executing the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN Command
- Task 6: Completing the RAC Configuration
- Task 7: Starting the Transport and Application of Redo



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Step 7: Starting Redo Apply in Real Time in RAC

- Execute the following command on one of the standby database instances to start Redo Apply in real time:

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

- To disable real-time Redo Apply:

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

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY  
DATABASE USING ARCHIVED LOGFILE DISCONNECT;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

On the standby database, issue the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE SQL command to start Redo Apply. This statement automatically mounts the database. In addition, include the DISCONNECT FROM SESSION option so that Redo Apply runs in a background session. The FROM SESSION portion of the syntax is no longer needed, but is acceptable.

The transmission of redo data to the remote standby location does not occur until after a log switch. Issue the following command on the primary database to force a log switch:

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

Note: The syntax option USING CURRENT LOGFILE of the ALTER DATABASE RECOVER MANAGED STANDBY DATABASE statement has been deprecated for Oracle Database 12c Release 1.

Quiz



The DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN command cannot be used for systems where the primary database is on a diskgroup and the standby database is on a file system.

- a. True
- b. False



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Prepare the primary database
- Configure Oracle Net Services
- Prepare the standby hosts
- Start the standby database instance
- Execute the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN command
- Complete the RAC configuration
- Start the transport and application of redo



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 4: Overview

This practice covers the following topics:

- Preparing the Primary Database to Create a Physical Standby Database in RAC
- Preparing the Standby Hosts and Creating a Physical Standby Database Using RMAN
- Completing a Physical Standby Database for RAC
- Verifying Redo Transport and Redo Apply Operations



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

5

Configuring Oracle Data Guard in an Oracle RAC Environment

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to describe the following considerations of the Data Guard configuration in RAC:

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

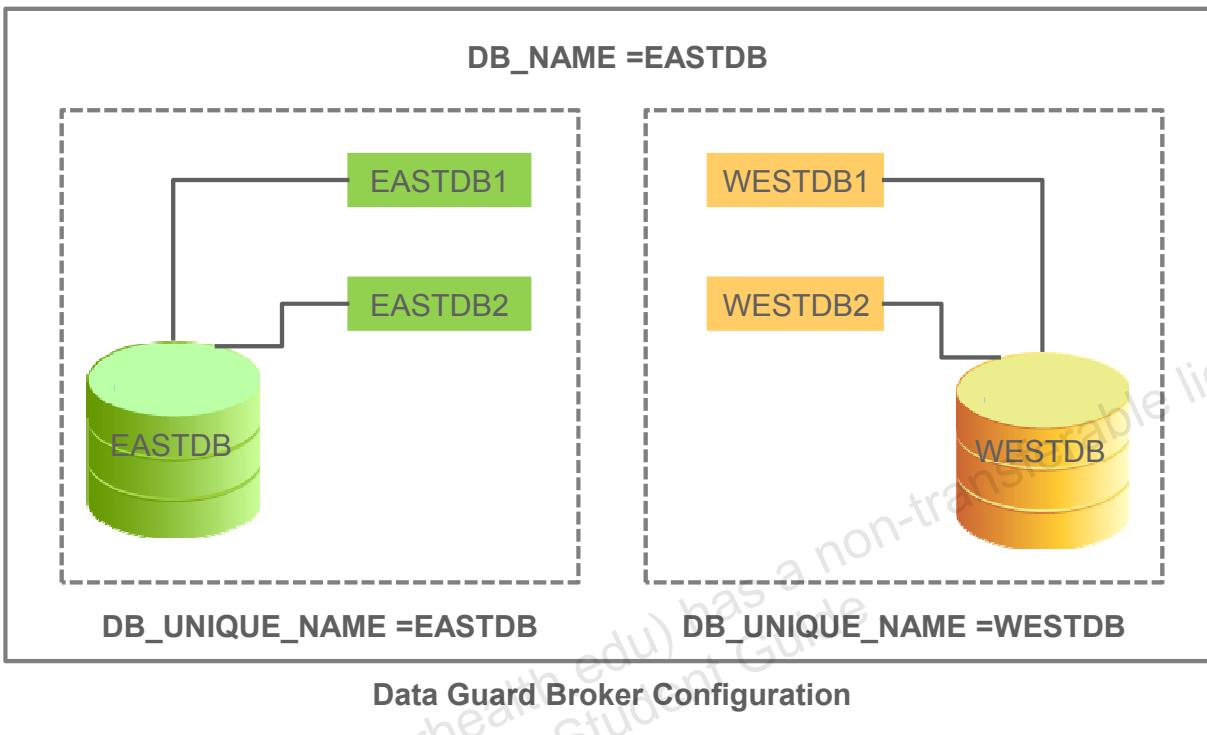
Configuration Considerations in RAC

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Data Guard Broker: Management Model



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The Data Guard broker helps you create, control, and monitor a Data Guard configuration. This configuration consists of a primary database that is protected by one or more standby databases. After the broker has created the Data Guard configuration, the broker monitors the activity, health, and availability of all systems in that configuration.

A broker configuration consists of the following objects:

- **Configuration object:** A named collection of database profiles. A database profile is a description of a database object, including its current state, current status, and properties.
- **Database objects:** Objects corresponding to primary (EASTDB) or standby databases (WESTDB)
- **Instance objects:** A database object may comprise one or more instance objects if it is a RAC database (EASTDB1, EASTDB2, WESTDB1, and WESTDB2).

The broker supports one or more Data Guard configurations, each of which includes a profile for one primary database as well as profiles for up to 30 physical, logical, RAC, or non-RAC standby databases. Far Sync destinations are also counted toward the limit of 30 locations.

Data Guard Broker: Requirements

- Oracle Database Enterprise Edition
- Single-instance or multi-instance environment
- COMPATIBLE parameter: Set to 12.1 or later for primary and standby databases to take advantage of new 12.1 features (optional).
- Oracle Net Services network files: Must be configured for the primary database and any existing standby databases or Far Sync instances. Enterprise Manager Cloud Control configures files for new standby databases.
- GLOBAL_DBNAME attribute: Set to a concatenation of db_unique_name_DGMGRL.db_domain.

Note: Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To use the Data Guard broker, you must comply with the following requirements:

- Use the Enterprise Edition of Oracle Database.
- Use a single-instance or multi-instance environment.
- Set the COMPATIBLE initialization parameter to 12.1 to take advantage of new Oracle Database 12c Release 1 (12.1) features.
- Enterprise Manager automatically configures the Oracle Net Services network files when it creates a standby database. If you configure an existing standby database in the broker configuration, you must configure the network files. You must use TCP/IP.
- To enable the Data Guard broker to restart instances during the course of broker operations, a service with a specific name must be statically registered with the local listener of each instance. The value of the GLOBAL_DBNAME attribute must be set to a concatenation of db_unique_name_DGMGRL.db_domain.

Note: Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard Broker configurations that are managed by Oracle Restart, RAC On Node or RAC as the broker will use the clusterware to restart an instance.

Data Guard Broker: Requirements

- DG_BROKER_START initialization parameter: Set to TRUE
- Primary database: ARCHIVELOG mode
- All databases: MOUNT or OPEN mode
- DG_BROKER_CONFIG_FILEn: Configured for shared access for any RAC databases
- LOG_ARCHIVE_DEST_n parameters: Must be cleared on any instance when using the SERVICE attribute before creating a broker configuration
- You must use a server parameter file (SPFILE) for initialization parameters.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Set the DG_BROKER_START initialization parameter to TRUE. This starts the DMON process. **Note:** When you use Enterprise Manager to create your configuration, this parameter is automatically set to TRUE.
- The primary database must be in ARCHIVELOG mode.
- Any database that is managed by the broker (including, for a RAC database, all instances of the database) must be mounted or open. The broker cannot start an instance.
- If any database in your configuration is a RAC database, you must configure the DG_BROKER_CONFIG_FILEn initialization parameters for that database so that they point to the same shared files for all instances of that database. You cannot use the default values for these parameters. **Note:** The shared files could be files on a cluster file system or on raw devices, or the files could be stored using Automatic Storage Management (ASM).
- As of Oracle Database 12c Release 1 (12.1), for all databases to be added to a broker configuration, any LOG_ARCHIVE_DEST_n parameters that have the SERVICE attribute set, but not the NOREGISTER attribute, must be cleared.
- To ensure that the broker can update the values of parameters in both the database instance and the configuration file, you must use the persistent server parameter file (SPFILE) to control static and dynamic initialization parameters.

Clearing Redo Transport Network Locations on Primary Database

- Clear existing LOG_ARCHIVE_DEST_n entries on each instance that references network locations:

```
SQL> select name,value from v$parameter where value
like '%SERVICE%';

NAME                      VALUE
-----
log_archive_dest_2        SERVICE=westdb SYNC REOPEN=15
                           valid_for=(ONLINE_LOGFILES,
                           PRIMARY_ROLE) db_unique_name =
                           westdb

SQL> alter system log_archive_dest_2='' scope=both;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

As of Oracle Database 12c Release 1 (12.1), no database can be added to the Data Guard broker configuration if the database instance has a LOG_ARCHIVE_DEST_n parameter that uses the SERVICE attribute. For an existing Data Guard environment created with SQL, this includes the primary database, physical standby databases, logical standby databases, and Far Sync instances.

You must clear any remote redo transport destinations on the primary database that do not have the NOREGISTER attribute, before a configuration can be created. Otherwise, the following error message is returned when you attempt to create the configuration:

ORA-16698: LOG_ARCHIVE_DEST_n parameter set for object to be added Failed.

Separation of DBA Duties: SYSDG

Introducing task-specific and least-privileged administrative privilege SYSDG

- Used to perform Data Guard operations with Data Guard Broker or DGMGRL
- Does not include data access privileges such as SELECT ANY TABLE
- Is granted to the SYSDG user that is created during database installation
- Can be explicitly used in Data Guard Broker connection by a SYSDG privileged user

```
DGMGRL> connect sysdg/password
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c provides support for separation of database administration (DBA) duties for the Oracle database by introducing task-specific and least-privileged administrative privileges that do not require the SYSDBA administrative privilege. The new privilege to connect and execute commands with Data Guard Broker is the SYSDG privilege.

Similar to SQL*Plus, DGMGRL can connect using operating system authentication (not shown in the slide), local user authentication, and remote user authentication using the password file. The generic syntax is as follows:

```
DGMGRL> CONNECT username/password[@connect-identifier]
```

The username and password must be valid for the configuration member to which you are trying to connect. The username you specify must have the SYSDG or SYSDBA privilege. Do not include AS SYSDG or AS SYSDBA on the CONNECT command. DGMGRL first attempts an AS SYSDG connection. If that fails, it will then attempt an AS SYSDBA connection. Additional database account users can be added to the password file by granting them the SYSDG privilege.

Connecting to the Primary Database with DGMGRL

- Connecting to the primary database on the local system:

```
DGMGRL> connect sysdg/password
```

- Connecting to the primary database remotely using the password file:

```
DGMGRL> connect sysdg/password@eastdb
```

- Adding a database user to the password file:

```
SQL> grant sysdg to dguser;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The following query can show who has the SYSKG privilege:

```
SQL> select * from v$pwfile_users;
USERNAME          SYSDB  SYSOP  SYSAS  SYSBA  SYSKG  SYSKM
-----
SYS              TRUE   TRUE   FALSE  FALSE  FALSE
SYSKG             FALSE  FALSE  FALSE  FALSE  TRUE   FALSE
SYSBACKUP        FALSE  FALSE  FALSE  TRUE   FALSE  FALSE
SYSKM             FALSE  FALSE  FALSE  FALSE  FALSE  TRUE
```

Shared Password File in a Disk Group

- A password file for Oracle Database or Oracle ASM can reside on a designated Oracle ASM disk group.
- Having the password files reside on a single location accessible across the cluster reduces:
 - Maintenance costs
 - Situations where passwords become out of sync
- The COMPATIBLE.ASM disk group attribute must be at least 12.1 for the disk group where the password is located.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

An individual password file for Oracle Database or Oracle ASM can reside on a designated Oracle ASM disk group. Having the password files reside on a single location accessible across the cluster reduces maintenance costs and situations where passwords become out of sync.

You can use a password file located on a disk group for authentication only if the Oracle ASM instance is running and the designated disk group is mounted. Otherwise, operating system authentication must be used to bootstrap the startup of the Oracle ASM instance and stack.

The COMPATIBLE.ASM disk group attribute must be set to at least 12.1 for the disk group where the password is to be located. The SYSASM privilege is required to manage the Oracle ASM password file. The SYSDBA privilege on Oracle ASM is required to manage the database password file.

The shared password file in a disk group is managed by `asmcmd` commands, the `orapwd` tool, and `srvctl` commands. `orapwd` supports the creation of password files on an Oracle ASM disk group. All other password file manipulation is performed with `asmcmd` or `srvctl` commands.

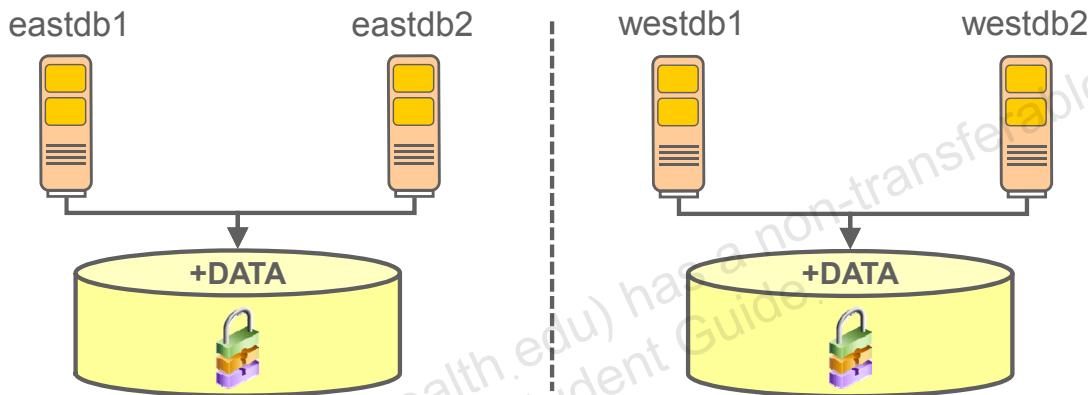
You must specify the disk group location and database unique name when using `orapwd` to create a database password file on a disk group.

Example: Creating a Shared Password File

- Specify the disk group location and database unique name when using `orapwd` to create a database password file:

```
$ orapwd file='+data/EASTDB/orapwdb' dbuniqueName='eastdb'
```

```
$ orapwd file='+data/WESTDB/orapwdb' dbuniqueName='westdb'
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

For example:

```
$ orapwd file='+data/ORCL/orapwdb' dbuniqueName='orcl'
```

The `asm` switch specifies that `orapwd` should create an Oracle ASM password file rather than a database password file. For example:

```
$ orapwd file='+data/ASM/orapwasm' asm=y
```

You can create a new password file in a disk group using a password file from a previous release. For example:

```
$ orapwd input_file='/oraclegrid/dbs/orapwasm'
file='+data/ASM/orapwasm' asm=y
```

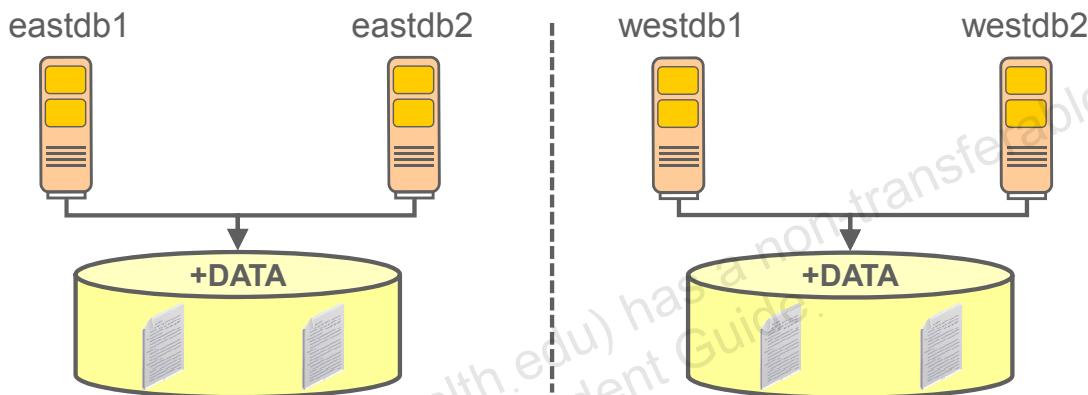
`srvctl` commands include updates to manage a password file in a disk group, such as the following for updating and displaying the location of the password file:

```
$ srvctl modify asm -pwfile location
$ srvctl modify database -db dbname -pwfile location
$ srvctl config asm
ASM home: /u01/app/12.1.0/grid
Password file: +DATA/orapwASM
ASM listener: LISTENER
ASM instance count: 3
Cluster ASM listener: ASMNET1LSNR_ASM
```

Data Guard Broker Configuration Files for RAC

- Configured for shared access for any RAC primary and standby databases:

```
* .DG_BROKER_CONFIG_FILE1=+DATA/eastdb/dr1config.dat  
* .DG_BROKER_CONFIG_FILE2=+DATA/eastdb/dr2config.dat
```



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Two copies of the Data Guard Broker (DGB) configuration files are maintained for each database so as to always have a record of the last-known valid state of the configuration. When the broker is started for the first time, the configuration files are automatically created and named using a default path name and file name that is operating system-specific.

When using a RAC environment, the DGB configuration files must be shared by all instances of the same database. You can override the default path name and file name by setting the following initialization parameters for that database: DG_BROKER_CONFIG_FILE1 and DG_BROKER_CONFIG_FILE2.

You have two possible options to share those files:

- Cluster file system
- ASM

Creating a Broker Configuration

- Define the broker creation and create a profile for the primary database:

```
DGMGRL> CREATE CONFIGURATION 'DG_CONFIG'  
AS PRIMARY DATABASE IS 'eastdb'  
CONNECT IDENTIFIER IS eastdb;
```

- Add a standby database to the configuration:

```
DGMGRL> ADD DATABASE 'westdb'  
as connect identifier is westdb;
```

- Enable the configuration:

```
DGMGRL> ENABLE CONFIGURATION;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Execute the CREATE CONFIGURATION command to define the broker creation and create a profile for the primary database. The following parameters must be specified:
 - **configuration-name**: User-specified name for the configuration
 - **database-name**: Used by the broker to reference the primary database. You must use the value of the DB_UNIQUE_NAME initialization parameter for the database name.
 - **connect-identifier**: The value you specify for the connect identifier is a fully specified connect descriptor or a name to be resolved by an Oracle Net Services naming method. The broker uses this value to communicate with the other databases defined in the configuration. The DGConnectIdentifier database property is set to the connect identifier value.
2. Use the ADD DATABASE DGMGRL command to define the standby database and create a broker configuration profile. The database name specified must be the same as the value of the DB_UNIQUE_NAME initialization parameter. The connect identifier is used by Oracle Net Services to access the database from all other databases in the configuration.
3. After defining the databases in the configuration, you enable the configuration and its databases by executing the ENABLE CONFIGURATION DGMGRL command.

Verifying the Configuration

- Verify the Data Guard broker configuration, including the primary and standby databases:

```
DGMGRL> SHOW CONFIGURATION
Configuration - DG_CONFIG

Protection Mode: MaxPerformance
Databases:
eastdb    - Primary database
westdb    - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

After defining the databases in the configuration, you verify the configuration by executing the SHOW CONFIGURATION DGMGRL command.

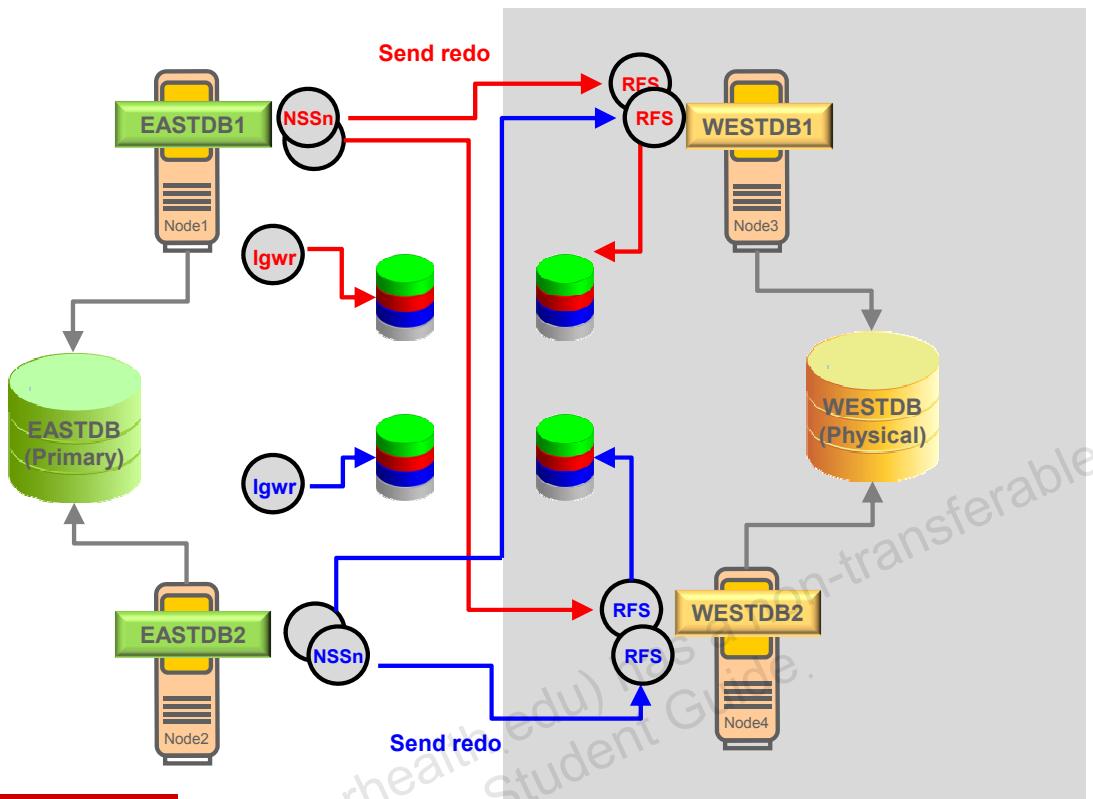
Configuration Considerations in RAC

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Redo Transport Service in RAC



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A RAC-to-RAC DG configuration can be set up in different ways. The slide shows you one possibility with a symmetric configuration where each primary instance sends its redo stream to a corresponding standby instance using standby redo log files. It is also possible for each primary instance to send its redo stream to only one standby instance that can also apply this stream to the standby database. However, you can get performance benefits by using the configuration shown in the slide. For example, assume that the redo generation rate on the primary is too great for a single receiving instance on the standby side to handle. Suppose further that the primary database is using the SYNC redo transport mode. If a single receiving instance on the standby cannot keep up with the primary, then the primary's progress is going to be throttled by the standby. If the load is spread across multiple receiving instances on the standby, then this is less likely to occur.

Example: Redo Transport Service in RAC

```
LOG_ARCHIVE_DEST_2=
  'SERVICE=westdb
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=westdb'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

```
DGMGRL> SHOW DATABASE eastdb
DATABASE - eastdb

Role:          PRIMARY
Intended State: TRANSPORT-ON
Instance(s):
  eastdb1
  eastdb2

Database Status:
SUCCESS
```



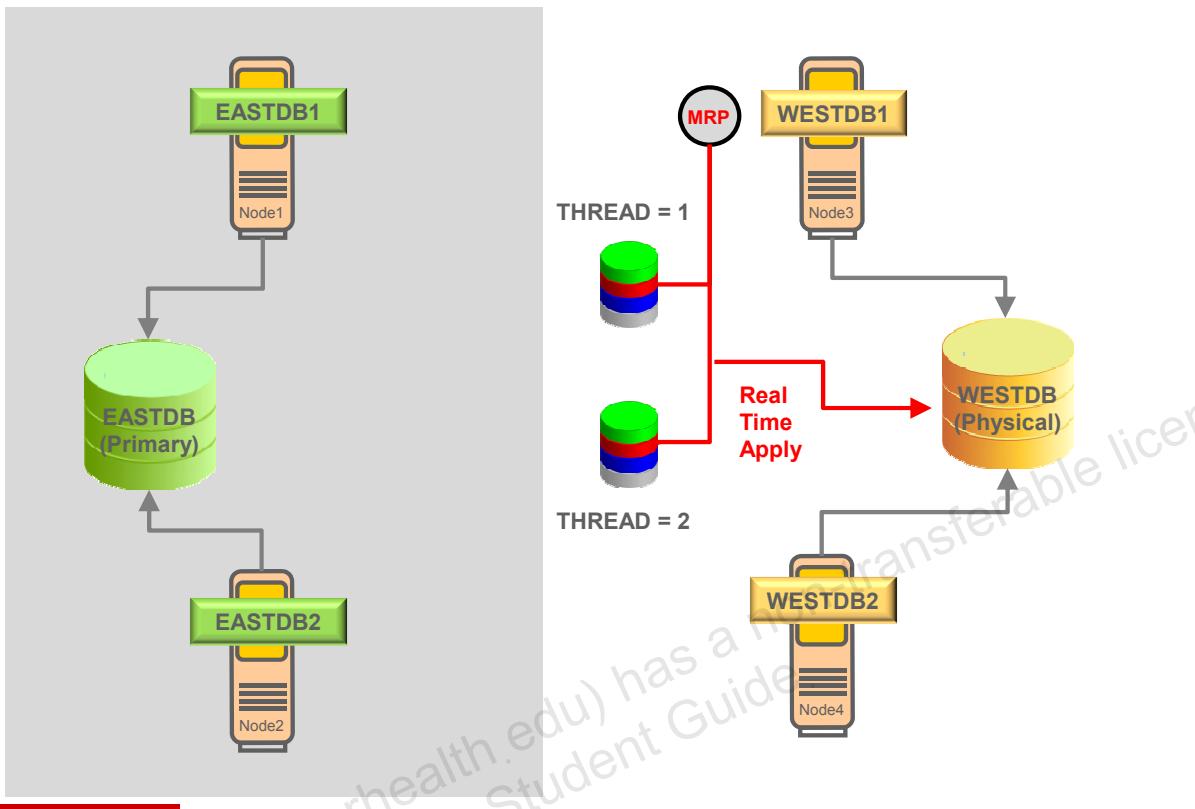
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide shows the primary instances that can send the redo stream to one or all standby instances.

Oracle recommends the following best practices when configuring an Oracle RAC primary database to send redo data to an Oracle RAC standby database:

- Use the same `LOG_ARCHIVE_DEST_n` parameter on each primary database instance to send redo data to a given standby database.
- Set the `SERVICE` attribute of each `LOG_ARCHIVE_DEST_n` parameter that corresponds to a given standby database to the same net service name.
- The net service name should resolve to an Oracle Net connect descriptor that contains an address list, and that address list should contain connection data for each standby database instance.

Redo Apply Service in RAC



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If a standby database is an Oracle RAC database, only one instance of the RAC database can have log apply services running at any time. This instance is called the **apply instance**. If the apply instance fails, the broker automatically moves log apply services to a different instance; this is called **apply instance failover**.

Example: Redo Apply Service in RAC

- Only one standby instance can apply the redo stream generated by the primary instances:

```
DGMGRL> SHOW DATABASE westdb
DATABASE - westdb

Role: PHYSICAL STANDBY
Intended State: APPLY-ON
Transport Lag: 0 seconds (computed 0 seconds ago)
Apply Lag: 0 seconds (computed 0 seconds ago)
Real Time Query: ON
Instance(s):
  westdb1 (apply instance)
  westdb2

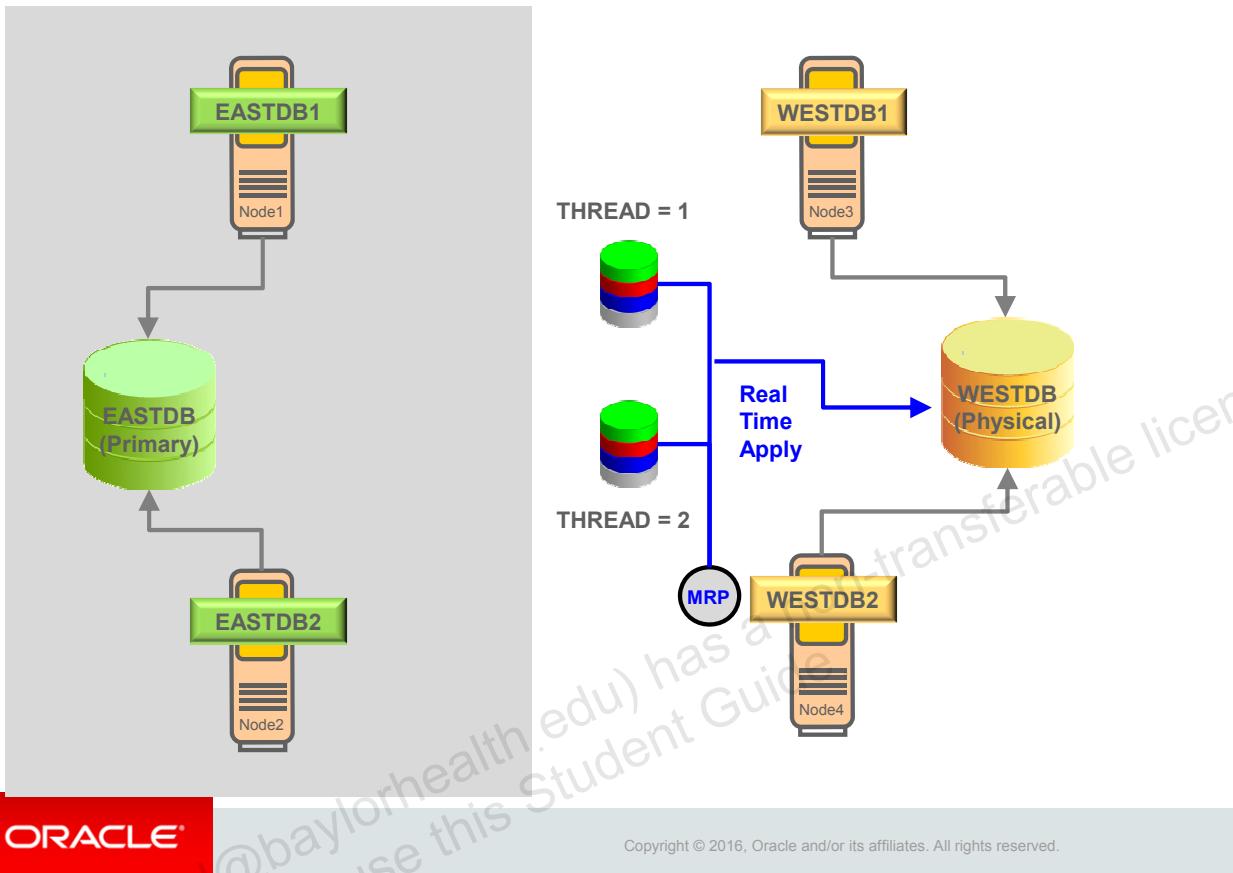
Database Status:
SUCCESS
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide shows that only one standby instance can apply the redo stream generated by the primary instances. If you have no preference about which instance should be the apply instance in an Oracle RAC standby database, the broker randomly picks an apply instance.

Switching Apply Instance in RAC



In a RAC-to-RAC DG configuration mode, if you want to select a particular instance as the apply instance, there are two methods to do this.

The first method is to pick an apply instance before there is an apply instance running in the RAC standby database. To do so, set the value of the `PreferredApplyInstance` configurable database property to the name of the instance (see the `SidName` property) that should be the apply instance. The broker starts log apply services on the instance specified by the `PreferredApplyInstance` property when no apply instance is yet selected in the RAC standby database. This could be the case before you enable the standby database for the first time, or if the apply instance just failed and the broker is about to do an apply instance failover, or if the RAC database is currently the primary and you want to specify its apply instance in preparation for a switchover. When the apply instance is selected and, as long as the apply instance is still running, the broker disregards the value of the `PreferredApplyInstance` property even if you change it.

The second method is to change the apply instance when the apply instance is already selected and is running. To change the apply instance, issue the `DGMGRl SET STATE` command to set the standby database state to `APPLY-ON`, with a specific apply instance argument. The `SET STATE` command will update the `PreferredApplyInstance` property to the new apply instance value, and then move log apply services to the new instance. The next slide shows the second method.

Example: Switching Apply Instance in RAC

```
DGMGRL> EDIT DATABASE westdb SET STATE= 'APPLY-ON'  
WITH APPLY INSTANCE = westdb2  
  
DGMGRL> SHOW DATABASE westdb 'PreferredApplyInstance';  
PreferredApplyInstance = 'westdb2'  
  
DGMGRL> SHOW DATABASE westdb  
DATABASE - westdb  
  
Role: PHYSICAL STANDBY  
Intended State: APPLY-ON  
Transport Lag: 0 seconds (computed 0 seconds ago)  
Apply Lag: 0 seconds (computed 0 seconds ago)  
Real Time Query: ON  
Instance(s):  
    westdb1  
    westdb2 (apply instance)  
  
Database Status:  
SUCCESS
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can issue the `EDIT DATABASE` command to relocate the apply service to another RAC instance.

Here is the syntax:

```
EDIT DATABASE database-name
```

```
SET STATE = state [WITH APPLY INSTANCE = instance-name];
```

- If the target state is `APPLY-ON` and this database is currently a physical or logical standby database, the optional `WITH APPLY INSTANCE` clause specifies which instance will become the apply instance.
- If the target state is not `APPLY-ON` or if the database is currently in the primary role, the `WITH APPLY INSTANCE` clause is ignored even if it is specified.
- You cannot change the state of a snapshot standby database.
- All instances of an Oracle RAC database are affected by this database state change.

Automatic Apply Instance Failover

- Set the `ApplyInstanceTimeout` configurable database property:

```
DGMGRL> EDIT DATABASE westdb SET PROPERTY  
'ApplyInstanceTimeout' = '10';  
DGMGRL> SHOW DATABASE westdb 'ApplyInstanceTimeout';  
ApplyInstanceTimeout = '10'
```

- After the broker initiates an apply instance failover, the broker selects a new apply instance according to the following rule:
 - If the `PreferredApplyInstance` property indicates an instance that is currently running, select it as the new apply instance.
 - Otherwise, pick a random instance that is currently running to be the new apply instance.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To tolerate a failure of the apply instance, the broker leverages the availability of the RAC standby database by automatically failing over log apply services to a different standby instance. The apply instance failover capability provided by the broker enhances data protection.

To set up apply instance failover, set the `ApplyInstanceTimeout` configurable database property to specify the time period that the broker will wait after detecting an apply instance failure and before initiating an apply instance failover. To select an appropriate timeout value, you need to consider the following:

- If there is another mechanism in the cluster that will try to recover the failed apply instance
- How long your business can tolerate not applying redo data on the standby database
- The overhead associated with moving the log apply services to a different instance. The overhead may include retransmitting, from the primary database, all log files accumulated on the failed apply instance that have not been applied if those log files are not saved in a shared file system that can be accessed from other standby instances.

The broker default value of the `ApplyInstanceTimeout` property is zero seconds, indicating that apply instance failover should occur immediately upon detection of the failure of the current apply instance.

After the broker initiates an apply instance failover, the broker selects a new apply instance according to the following rule: If the PreferredApplyInstance property indicates an instance that is currently running, select it as the new apply instance; otherwise, pick a random instance that is currently running to be the new apply instance.

Configuration Considerations in RAC

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Data Guard Deployment Options

Requirements	Deployment Options	Transport	If no acknowledgment is received:
Zero data loss Double failure protection	Maximum Protection	SYNC	Stall primary until an acknowledgement is received.
Zero data loss with single site failure	Maximum Availability	SYNC (AFFIRM)	Stall primary until threshold period expires; then resume processing.
Potential for data loss when multiple sites fail	Maximum Availability	SYNC (NOAFFIRM)	Stall primary until threshold period expires; then resume processing.
Potential for minimal data loss	Maximum Performance	ASYNC	Primary never waits for standby acknowledgement.



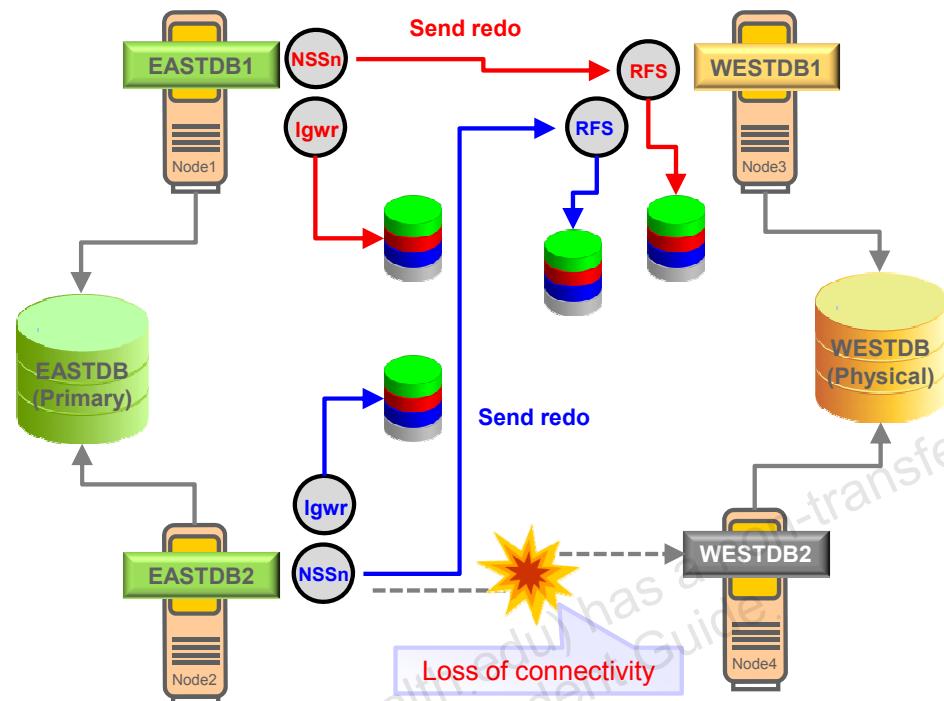
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Data Guard offers maximum protection, maximum availability, and maximum performance modes to help enterprises balance data availability against system performance requirements.

In some situations, a business cannot afford to lose data. In other situations, the availability of the database may be more important than the loss of data. Some applications require maximum database performance and can tolerate the potential loss of data.

Consider the characteristics of each protection mode. You must balance cost, availability, performance, and transaction protection when choosing the protection mode.

Behavior of the Data Protection Mode in RAC



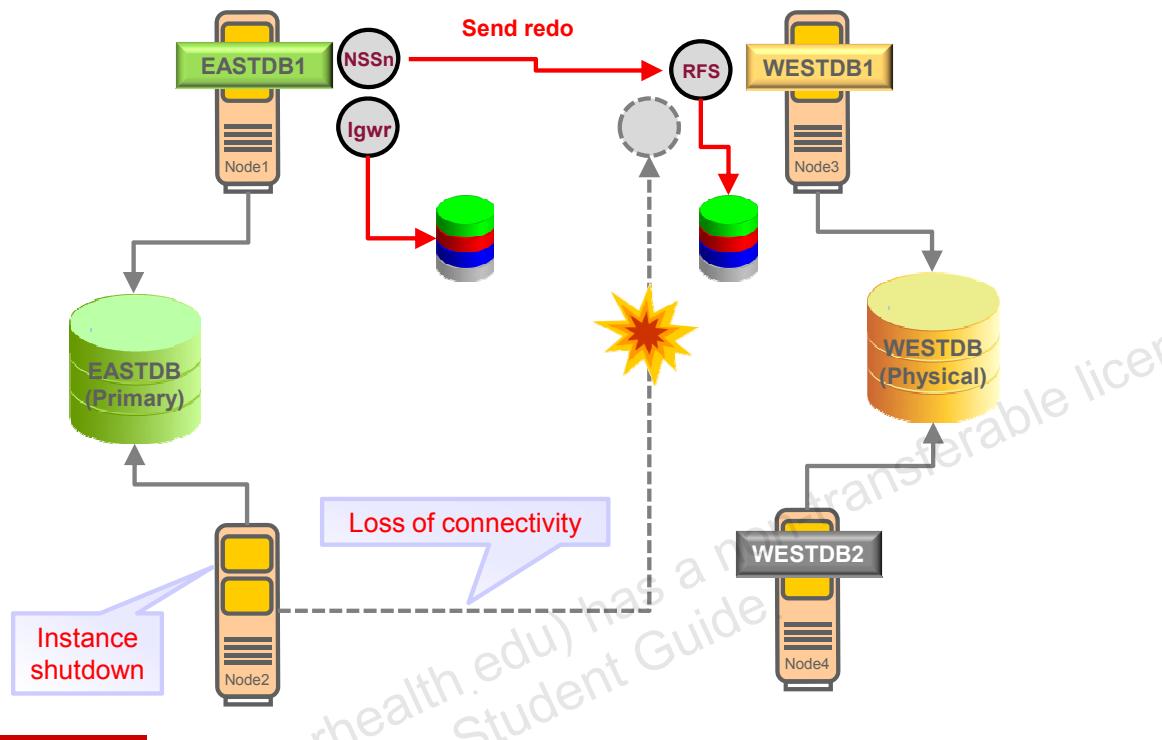
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If any instance of an Oracle RAC primary database(eastdb2) loses connectivity with a standby database(westdb), you will see the following behavior in all three data protection modes in Oracle RAC environments:

1. All other primary database instances(eastdb1) stop sending redo to the standby database(westdb) for the number of seconds specified on the `LOG_ARCHIVE_DEST_n REOPEN` attribute.
2. Then, all primary database instances attempt to reconnect to the standby database. The diagram in the slide shows the primary instance(eastdb2) reconnected to another standby instance(westdb1).

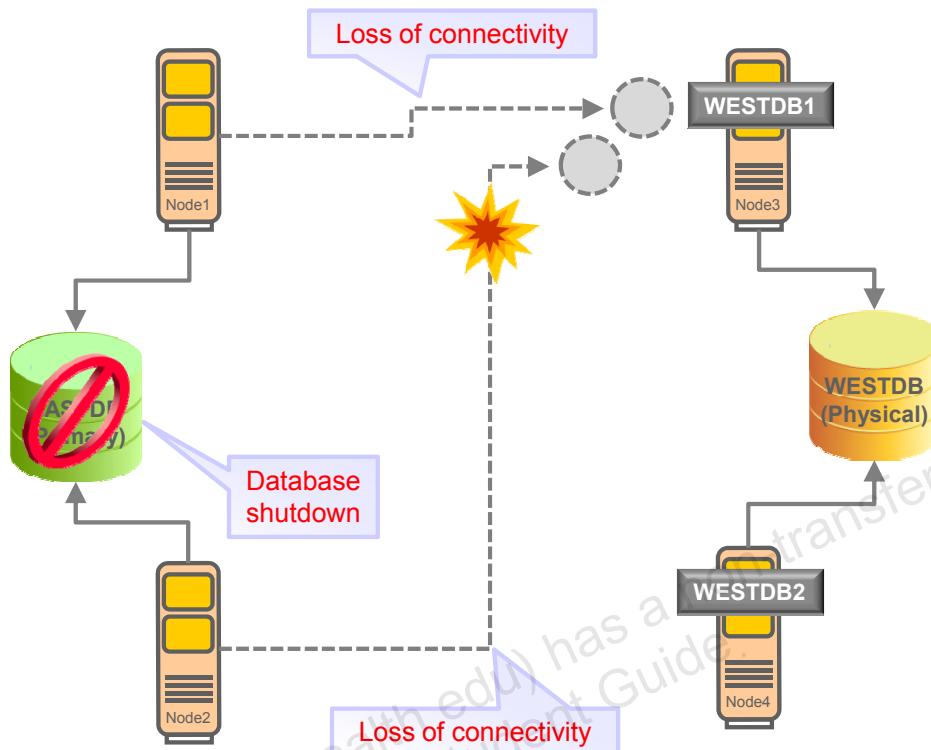
Behavior of the Maximum Protection Configuration in RAC



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example illustrates the connectivity issue from one of the primary instances(eastdb2) to the standby instance(westdb1). In the maximum protection configuration, if a primary instance(eastdb2) has lost connection to the *last* participating SYNC destination, the instance(eastdb2) loses connectivity and will be shut down. Other instances(eastdb1) in an Oracle RAC configuration that still have connectivity to the standby destinations will recover the lost instance and continue sending redo to their standby destinations.

Behavior of the Maximum Protection Configuration in RAC



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In the maximum protection configuration, when every instance in an Oracle RAC configuration loses connectivity to the last standby destination, the primary database will be shut down.

In the maximum availability and maximum performance configurations, when every instance in an Oracle RAC configuration loses connectivity to the standby destination, the primary database will continue operation in maximum performance mode.

Configuration Considerations in RAC

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Performing a Switchover in RAC

- Switchover
 - Planned role transition
 - Used for operating system or hardware maintenance
 - Manually invoked on primary database
- Switchover in RAC
 - As of Oracle Database 12c, when you perform a switchover from an Oracle RAC primary database to a physical standby database, it is no longer necessary to shut down all but one primary database instance.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Switchover

You can use the switchover feature to switch the role of the primary database to one of the available standby databases. The chosen standby database becomes the primary database, and the original primary database then becomes a standby database. There is no need to re-create any of the databases involved in the switchover operation. There is no data divergence between the original and new primary databases after successful completion of the switchover.

Note that as of Oracle Database 12c Release 1 (12.1), when you perform a switchover from an Oracle RAC primary database to a physical standby database, *it is no longer necessary to shut down all but one primary database instance*.

Example: Validating Databases for Switchover

- The VALIDATE DATABASE command performs a comprehensive set of database checks before a role change:

```
DGMGRL> VALIDATE DATABASE eastdb;
Database Role: Primary database
Ready for Switchover: Yes

DGMGRL> VALIDATE DATABASE westdb;
Database Role: Physical standby database
Primary Database: eastdb
Ready for Switchover: Yes
Ready for Failover: Yes (Primary Running)
Transport-Related Property Settings:
  Property      eastdb Value          westdb Value
  RedoRoutes    (eastdb:eastdb ASYNC) (westdb:westdb ASYNC)

DGMGRL>
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The VALIDATE DATABASE command performs a comprehensive set of database checks before a role change. The checks use information available in various Oracle Data Guard views, as well as the Automatic Diagnostic Repository. The VALIDATE DATABASE command shows a brief summary of the database, and reports any errors or warnings that were detected. VALIDATE DATABASE VERBOSE shows everything in the brief summary, plus all items that were validated.

Example: Switchover by Using DGMGRL

- After verifying the conditions required for a switchover, execute the SWITCHOVER command:

```
DGMGRL> SWITCHOVER TO westdb;

Performing switchover NOW, please wait...
Operation requires a connection to instance "eastdb2"
on database "eastdb"
Connecting to instance "eastdb2"...
Connected as SYSDBA.
New primary database "eastdb" is opening...
Oracle Clusterware is restarting database "westdb" ...
Switchover succeeded, new primary is "eastdb"

DGMGRL>
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

After verifying the conditions required for executing a switchover, execute the SWITCHOVER command to perform the switchover operation. The Data Guard broker performs the following operations:

- Verifies that the primary database and the target standby database are in the correct states
- Shuts down any instances as necessary
- Switches roles between the primary and standby databases
- Updates the broker configuration file with the new role information
- Restarts the new standby database (former primary) with Redo Apply running
- Opens the new primary database in read/write mode and starts redo transport services

Note: For detailed information about each step, see *Oracle Data Guard Broker*.

Performing a Failover in RAC

- Failover
 - Unplanned role transition
 - Used in an emergency
 - Minimal or no data loss (depending on the data-protection mode)
 - Initiated at standby database
- Failover in RAC
 - Before performing a failover to an Oracle RAC standby database, first shut down all but one standby instance.
 - After the failover completes, restart the instances that were shut down.
 - In a broker environment, the broker directs Oracle Clusterware to restart all instances that may have been shut down before the failover.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Failover

You invoke a failover operation when a failure occurs on the primary database and there is no possibility of recovering the primary database in a timely manner. During a failover operation, the standby database assumes the primary database role. You invoke the failover operation on the standby database that should fail over to the primary role.

In a broker environment, if the failover target database is an Oracle RAC physical or snapshot standby database, the broker directs Oracle Clusterware to restart all instances that may have been shut down before the failover.

Here is an example of the manual failover with Data Guard broker.

```
DGMGRL> failover to westdb
Performing failover NOW, please wait...
Failover succeeded, new primary is "westdb"
DGMGRL>
```

Configuration Considerations in RAC

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Flashback Database

- Flashback Database provides the following in a Data Guard configuration:
 - An alternative to restoring and recovering the primary database
 - A way to reinstate the primary database that was disabled as part of a failover to any standby database operation
 - An alternative to delaying the application of redo to protect against user errors or logical corruptions
- Flashback Database is used by the following features in a Data Guard configuration:
 - Fast-start failover
 - Snapshot standby



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Flashback Database provides the following advantages in a Data Guard configuration:

- Provides an alternative to delaying the application of redo to protect against user errors or logical corruptions. By using Flashback Database in this context, standby databases are more closely synchronized with the primary database, thereby reducing failover and switchover times.
- Eliminates the need to completely re-create the original primary database after a failover. The failed primary database can be flashed back to a point in time before the failover and converted to be a standby database for the new primary database.

Flashback Database is used in a Data Guard configuration for the following features:

- **Fast-start failover:** You must enable Flashback Database and set up a fast recovery area on the primary database and the target standby database before enabling fast-start failover. (See the lesson titled “Enabling Fast-Start Failover” for additional information.)
- **Snapshot standby:** To convert a physical standby database to a snapshot standby database, you must configure the fast recovery area and size. If Flashback Database is not enabled, it will be enabled when the physical standby database is converted to a snapshot standby database. (See the lesson titled “Creating and Managing a Snapshot Standby Database” for additional information.)

Configuring Flashback Database

- On primary and standby databases, configure the fast recovery area:

```
SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='+FRA';
```

- Set the retention target:

```
SQL> ALTER SYSTEM SET DB_FLASHBACK_RETENTION_TARGET=2880  
SCOPE=BOTH;
```

- Enable Flashback Database:

```
SQL> ALTER DATABASE FLASHBACK ON;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The database can be open when you enable Flashback. To enable Flashback Database:

1. Configure the fast recovery area.
2. Set the retention target with the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter.

You can specify an upper limit, in minutes, on how far back you want to be able to flashback the database. The example uses 2,880 minutes, which is equivalent to two days.

This parameter is only a target and does not provide any guarantee. Your flashback time interval depends on how much flashback data was kept in the fast recovery area.

3. Enable Flashback Database with the following command:

```
ALTER DATABASE FLASHBACK ON;
```

Before you can issue the command to enable Flashback Database, the database must be configured for archiving.

Determine whether Flashback Database is enabled with the following query:

```
SELECT flashback_on FROM v$database;
```

Disable Flashback Database with the `ALTER DATABASE FLASHBACK OFF` command. As a result, all existing Flashback Database logs are deleted automatically.

Configuration Considerations in RAC

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Fast-Start Failover in RAC

- Fast-Start Failover implements automatic failover to a standby database:
 - Triggered by failure of site, hosts, storage, data file offline immediate, or network
 - Works with and supplements RAC server failover
- Failover occurs in seconds (< 20 seconds).
 - Comparable to cluster failover
- The original production site automatically rejoins the configuration after recovery.
- Automatically monitored by an Observer process:
 - Located on a distinct server in a distinct data center
 - Can be restarted on failure by Enterprise Manager
 - Installed through Oracle Client Administrator



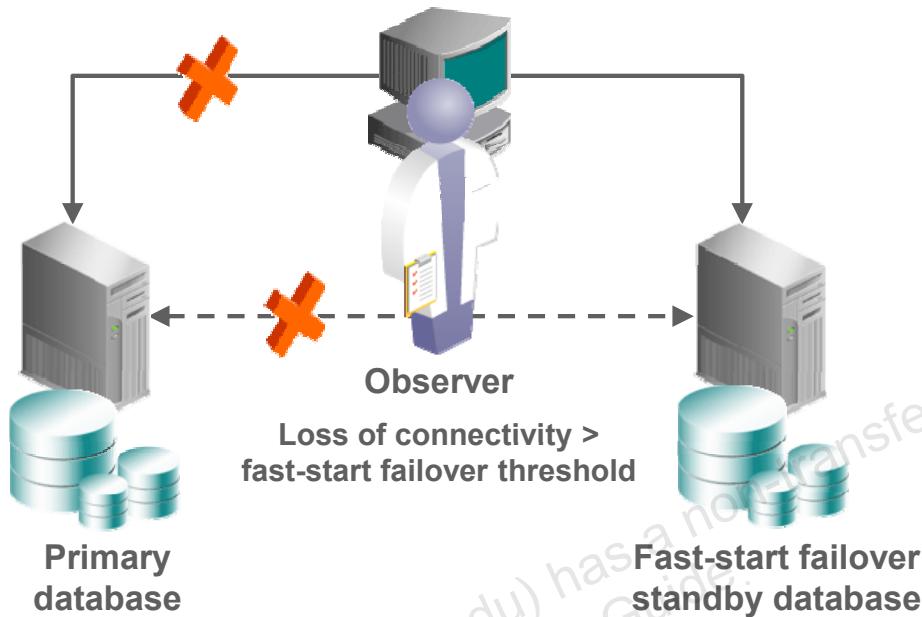
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Fast-Start Failover is a feature that automatically, quickly, and reliably fails over to a designated, synchronized standby database in the event of loss of the primary database, without requiring manual intervention to execute the failover. In addition, following a fast-start failover, the original primary database is automatically reconfigured as a new standby database upon reconnection to the configuration. This enables Data Guard to restore disaster protection in the configuration as soon as possible.

Fast-Start Failover is used in a Data Guard configuration under the control of the Data Guard Broker, and may be managed using either `dgmgrl` or Oracle Enterprise Manager Grid Control. There are three essential participants in a Fast-Start Failover configuration:

- The primary database, which can be a RAC database
- A target standby database, which becomes the new primary database following a fast-start failover
- The Fast-Start Failover Observer, which is a separate process incorporated into the `dgmgrl` client that continuously monitors the primary database and the target standby database for possible failure conditions. The underlying rule is that out of these three participants, whichever two can communicate with each other will determine the outcome of the fast-start failover. In addition, a fast-start failover can occur only if there is a guarantee that no data will be lost.

When Does Fast-Start Failover Occur?



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Fast-start failover occurs when any of the following conditions are met:

- Loss of connectivity between both the primary database and the observer—and between the primary database and the fast-start failover target standby database—exceeds the fast-start failover threshold.
- The database health-check mechanism determines any of the following (as optionally configured):
 - A data file is offline because of a write error.
 - Dictionary corruption of a critical database object occurs.
 - Control file is permanently damaged because of a disk failure.
 - Log Writer (LGWR) is unable to write to any member of the log group because of an I/O error.
 - Archiver is unable to archive a redo log because the device is full or unavailable.
- An instance crash occurs for a single-instance database.
- All instances of a RAC primary database crash.
- Shutdown abort of the primary database occurs.
- An application initiates a fast-start failover by calling the DBMS_DG.INITIATE_FS_FAILOVER function.

Fast-Start Failover Prerequisites

The following prerequisites must be met to enable fast-start failover:

- Configuration must be in maximum availability or maximum performance mode.
- LogXptMode property of target must be set as follows:
 - SYNC or FASTSYNC in maximum availability mode
 - ASYNC in maximum performance mode
- Flashback Database must be enabled on the primary database and the preselected target standby database.
- tnsnames.ora entries should be configured for the observer.
- A static service name should be created so that the observer can automatically restart databases.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- The broker configuration must be operating in maximum availability or maximum performance mode.
- The primary database must be configured with standby redo log files.
- The standby database that is the target of fast-start failover must have its LogXptMode property set to SYNC or FASTSYNC to enable fast-start failover in maximum availability mode or to ASYNC to enable fast-start failover in maximum performance mode. To use a Far Sync instance with fast-start failover, the primary database must use SYNC or FASTSYNC to the Far Sync instance. The Far Sync instance must use ASYNC to the target standby database. The RedoRoutes property must be set if Far Sync is shipping the redo information to the fast-start failover target.
- Flashback Database must be enabled on the primary database and the target standby database.
- The primary database and the target standby database must have connectivity.
- The tnsnames.ora file should be configured on the observer system so that the observer is able to connect to the primary database and the preselected target standby database.
- A static service name should be created so that the observer can automatically restart a database as part of reinstatement.

Configuring Fast-Start Failover

1. Specify the target standby database.
2. Set the protection mode.
3. Set the `FastStartFailoverThreshold` property.
4. Set additional database properties.
5. Set additional fast-start failover conditions.
6. Enable fast-start failover.
7. Start the observer.
8. Verify the configuration.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You will practice how to configure fast-start failover with the manual steps in detail.

Avoiding a False Failover in RAC

- For RAC, consider specifying a higher value to minimize the possibility of a false failover in the event of an instance failure.
- FastStartFailoverThreshold property: 30 seconds
 - 30 seconds (by default)
 - 6 seconds (lowest possible value)
- Recommended settings for the FastStartFailoverThreshold property:
 - RAC primary =
(CSS miss count + reconfiguration time) + (24–40 seconds)

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY  
FastStartFailoverThreshold = <value>;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The fast-start failover threshold specifies how long the observer and the target standby database should simultaneously wait to hear from the primary database before initiating a fast-start failover. The threshold value is specified as a positive, nonzero number of seconds. The default value for the `FastStartFailoverThreshold` property is 30 seconds. When choosing a value for this property, you must balance the increased risk of an unnecessary failover (for example, if the network connection was temporarily broken for a few seconds) against the benefits of faster failover and less down time during a critical outage.

Recommended settings for the `FastStartFailoverThreshold` property:

- Single-instance primary, low-latency reliable network = 10–15 seconds
- Single-instance primary, high-latency network over WAN = 30–45 seconds
- RAC primary = (Cluster Synchronization Service (CSS) miss count + reconfiguration time) + (24–40 seconds)

Execute the `EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold = threshold-val` command when connected to the primary database or to any standby database in the Data Guard broker configuration with connectivity to the primary database.

Note: You can modify this property whether fast-start failover is enabled or disabled.

Example: Fast-Start Failover

```
DGMGRL> show fast_start failover;
Fast-Start Failover: ENABLED

Threshold:          30 seconds
Target:             london
Observer:           host04
Lag Limit:          30 seconds
Shutdown Primary:  TRUE
Auto-reinstate:    TRUE
Observer Reconnect: (none)
Observer Override: FALSE

Configurable Failover Conditions
Health Conditions:
  Corrupted Controlfile      YES
  Corrupted Dictionary       YES
  Inaccessible Logfile       NO
  Stuck Archiver             NO
  Datafile Offline            YES

Oracle Error Conditions:
  (none)
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can also use the SHOW FAST_START FAILOVER command to display all fast-start failover information.

Summary

In this lesson, you should have learned how to describe the following considerations of the Data Guard configuration in RAC:

- Data Guard Broker Configuration
- Redo Transport and Apply Services
- Data Guard Deployment Options
- Role Transition Services
- Flashback Database
- Fast-Start Failover



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 5: Overview

This practice covers the following topics:

- Using a Shared Password File in an ASM Disk Group
- Creating a Data Guard Broker Configuration for RAC
- Monitoring Redo Transport Service and Redo Apply Service in RAC
- Configuring Flashback Database
- Enabling Fast-Start Failover
- Testing Fast-Start Failover in RAC
- Performing Switchover to Reinstated Database in RAC and Disabling Fast-Start Failover



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

6

Managing Physical Standby Files After Structural Changes on the Primary Database

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to describe the primary database changes that require manual intervention at a physical standby database.

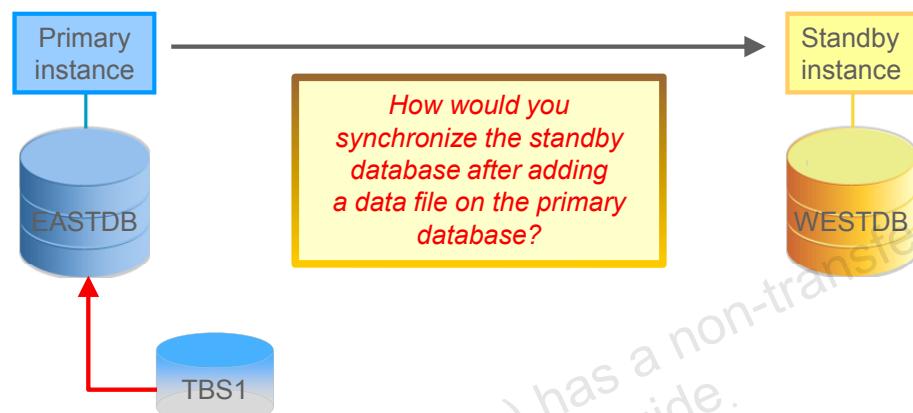


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Scenario 1: Creating a Tablespace

- Assumptions:
 - STANDBY_FILE_MANAGEMENT = MANUAL
 - Active Data Guard option enabled



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The STANDBY_FILE_MANAGEMENT database initialization parameter controls whether the addition of a data file to the primary database is automatically propagated to physical standby databases. If the STANDBY_FILE_MANAGEMENT database parameter on the physical standby database is set to AUTO, any new data files created on the primary database are automatically created on the physical standby database.

However, if you create a tablespace in the situations where the STANDBY_FILE_MANAGEMENT database parameter on the physical standby database is set to MANUAL (for example, you created a tablespace while synchronizing the physical standby database manually), then a new data file must be manually copied from the primary database to the physical standby databases or an empty data file must be created in the physical standby database by using the ALTER DATABASE CREATE DATAFILE command.

Action Required on Physical Standby

- On the primary database:

```
SQL> CREATE TABLESPACE <TBS_NAME> ..
```

- Checking the physical standby's Open Mode:

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

```
OPEN_MODE
```

```
-----  
READ ONLY WITH APPLY
```

- On the physical standby database:

```
SQL> ALTER DATABASE CREATE DATAFILE <filename> ..
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

On a physical standby database for which the Oracle Active Data Guard option has been enabled, you cannot use the manual copy method. Instead, you must execute the following SQL statement on the standby to create an empty data file:

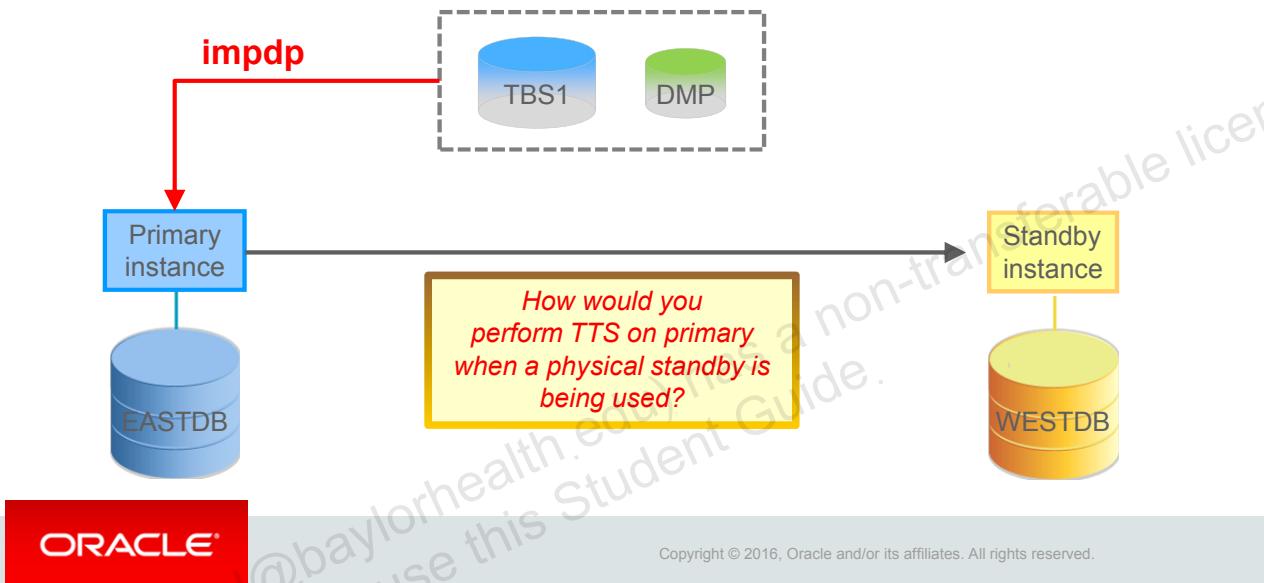
```
SQL> ALTER DATABASE CREATE DATAFILE [filename | filenumber]  
      AS [NEW | new_filename];
```

You must specify which one to rename: the filename or the filenumber.

You should also specify either the new filename or NEW. The NEW keyword lets Oracle automatically select a name, if Oracle Managed Files (OMF) is enabled.

Scenario 2: Using a Transportable Tablespace

- Assumptions:
 - Path names are *NOT* the same on the primary and standby databases.
 - DB_FILE_NAME_CONVERT is not configured.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can use the Oracle transportable tablespaces (TTS) feature to move a subset of an Oracle database and plug it into another Oracle database, essentially moving tablespaces between the databases. To move or copy a set of tablespaces into a primary database when a physical standby is being used, you need to perform additional tasks. The next slide shows these steps.

Note that if an existing data file from another database is copied to a primary database, it must also be copied to the standby database and the standby control file must be re-created, regardless of the setting of the STANDBY_FILE_MANAGEMENT parameter.

You will practice this task in practice 6-2 titled “Using Transportable Tablespaces with a Physical Standby Database.”

Action Required on Physical Standby

- Copy the data files and the export file to the primary database and copy the data files to the standby database.
- On the physical standby:

```
SQL> ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT = MANUAL;
SQL> ALTER DATABASE RENAME FILE ...
```

- On the primary database, plug in the tablespace:

```
impdp system dumpfile=ttstest.dmp directory=data_pump_dir
transport_datafiles='+DATA/eastdb/datafile/ttstest01.dbf'
logfile=tts_import.log
```

- Redo data will be generated and applied at the standby site to plug the tablespace into the standby database.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To move or copy a set of tablespaces into a primary database when a physical standby is being used, perform the following steps:

1. Generate a transportable tablespace set that consists of data files for the set of tablespaces being transported and an export file containing structural information for the set of tablespaces.
2. Transport the tablespace set:
 - Copy the data files and the export file to the primary database.
 - Copy the data files to the standby database.

The data files must have the same path name on the primary and standby databases unless the DB_NAME_CONVERT database initialization parameter has been configured.

If DB_NAME_CONVERT has not been configured and the path names of the data files are not the same on the primary and standby databases, issue the ALTER DATABASE RENAME FILE statement to rename the data files. Do this after Redo Apply has failed to apply the redo generated by plugging the tablespace into the primary database. The STANDBY_FILE_MANAGEMENT initialization parameter must be set to MANUAL before renaming the data files, and should be reset to the previous value after renaming the data files.

3. Plug in the tablespace.

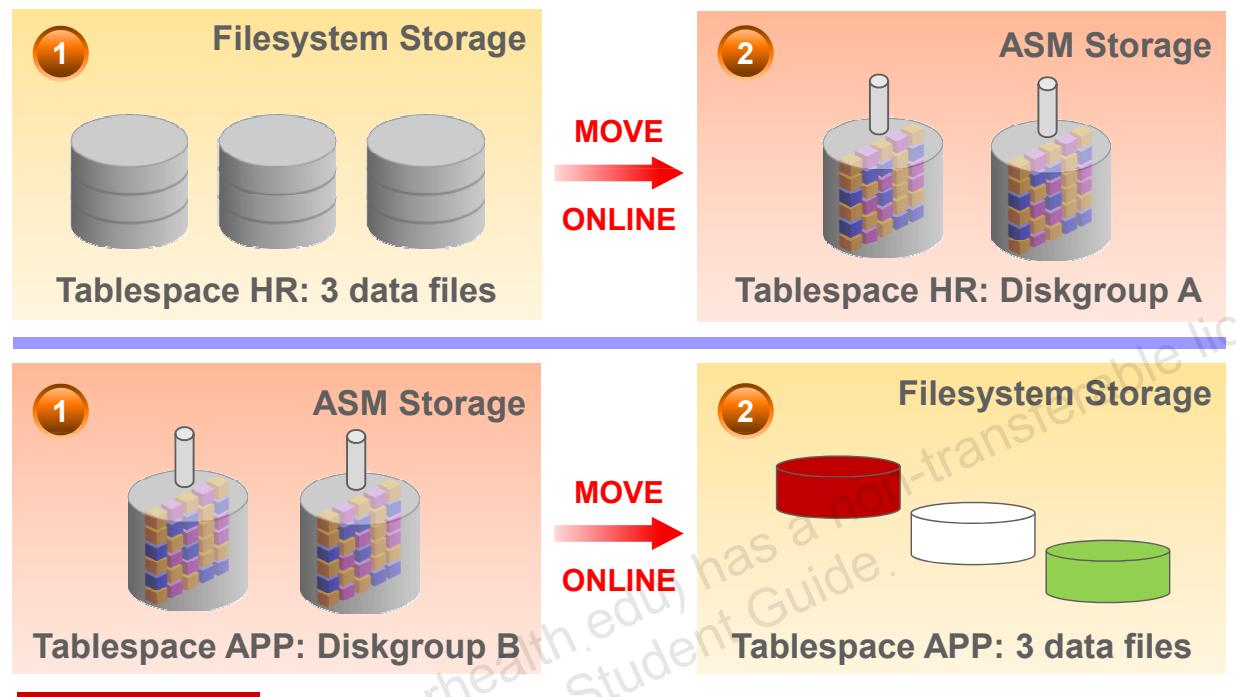
Invoke the Data Pump utility to plug the set of tablespaces into the primary database.

Redo data will be generated and applied at the standby site to plug the tablespace into the standby database.

You will practice this task in practice 6-2 titled “Using Transportable Tablespaces with a Physical Standby Database.”

Review: Online Move Data File in 12c

Move a **data file** to another kind of storage system **ONLINE**.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The Oracle Database 12c Online Move data file feature provides the capability to move an online data file from one kind of storage system to another while the database is open and accessing the file.

This implies that queries and DML/DDL operations can be performed while the data file is being moved:

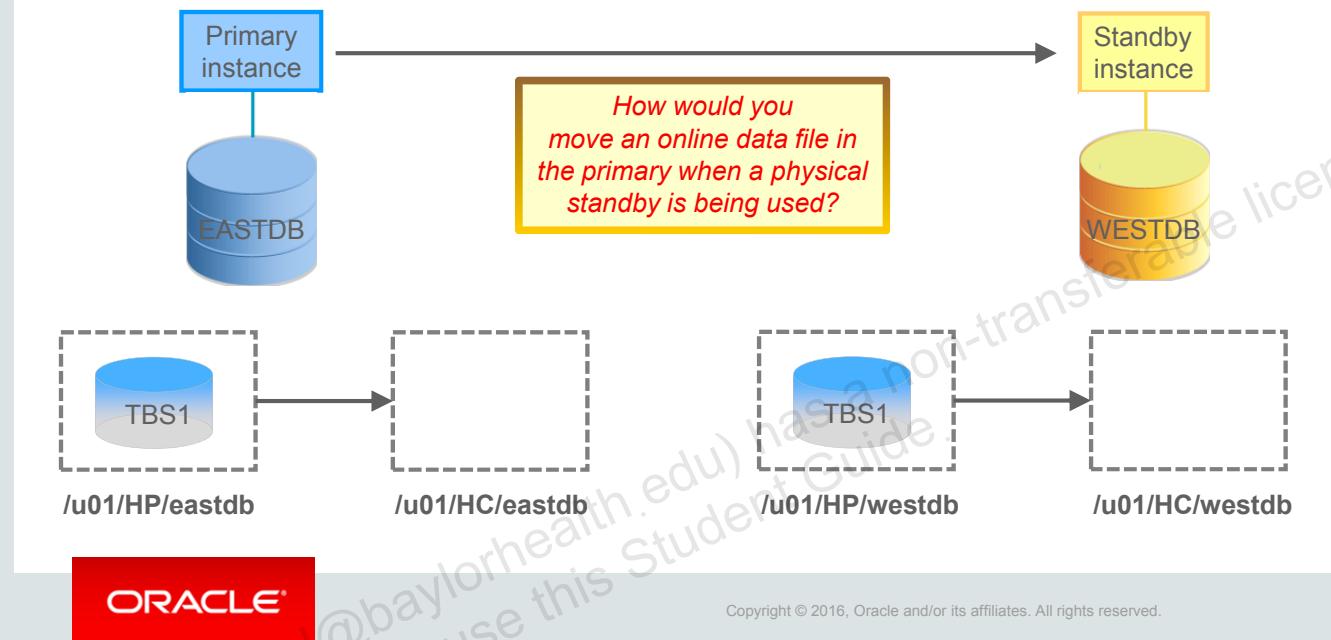
- You can create tables and indexes.
- You can rebuild indexes online.
- You can select tables and partitions data.
- If objects are compressed while the data file is moved, the compression remains the same.

This means that many maintenance operations can be used to move:

- Data files from one kind of storage system to another
- A database into ASM without going down

Scenario 3: Moving an Online Data File

- Assumptions:
 - Rename one or more data files in the primary database.
 - Or move a data file online in the primary database.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When you move or rename a data file in the primary database, the change is not propagated to the standby database. Therefore, if you want to rename the same data file on the standby database, you must manually make the equivalent modifications on the standby database because the modifications are not performed automatically, even if the STANDBY_FILE_MANAGEMENT initialization parameter is set to AUTO.

Action Required on Physical Standby

1. Move the data file in the primary database online:

```
SQL> ALTER DATABASE MOVE datafile 5 TO  
'/disk1/oracle/oradata/payroll/tbs_x.dbf';
```

2. On the standby database, note that the STANDBY_FILE_MANAGEMENT database initialization parameter must be set to MANUAL.
3. Connect to the standby database and stop Redo Apply:

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

4. Open the standby database in read-only mode or start Redo Apply with an Oracle Active Guard license:

```
SQL> ALTER DATABASE OPEN READ ONLY;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The following steps describe how to move or rename a data file in the primary database and manually propagate the changes to the standby database:

1. Use the ALTER DATABASE MOVE DATAFILE command to rename a data file.

Note: This command lets you rename a data file while allowing read/write access to the data file. Adequate storage area is a prerequisite for moving a data file because during the execution of the MOVE DATAFILE command, the database maintains both the original and the renamed data files as two separate files.

```
SQL> ALTER DATABASE MOVE datafile 5 TO  
'/disk1/oracle/oradata/payroll/tbs_x.dbf';
```

2. Set the STANDBY_FILE_MANAGEMENT database initialization parameter to MANUAL.
3. Connect to the standby database and stop Redo Apply.
4. Open the standby database in read-only mode. Note that the online move data file operation cannot be executed on physical standby while standby recovery is running in a mounted but not open instance.

Note: On a physical standby, an online move data file operation can be executed while standby recovery is running if the instance that opens the database is in read-only mode. This functionality requires an Oracle Active Data Guard license.

Action Required on Physical Standby

5. Move the data file in the standby database online:

```
SQL> ALTER DATABASE MOVE datafile 5 TO  
'/disk1/oracle/oradata/payroll/tbs_x.dbf';
```

6. Shut down the standby database.
7. Start and mount the standby database.
8. Restart Redo Apply:

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

9. The STANDBY_FILE_MANAGEMENT database initialization parameter can be reset after moving or renaming a data file.



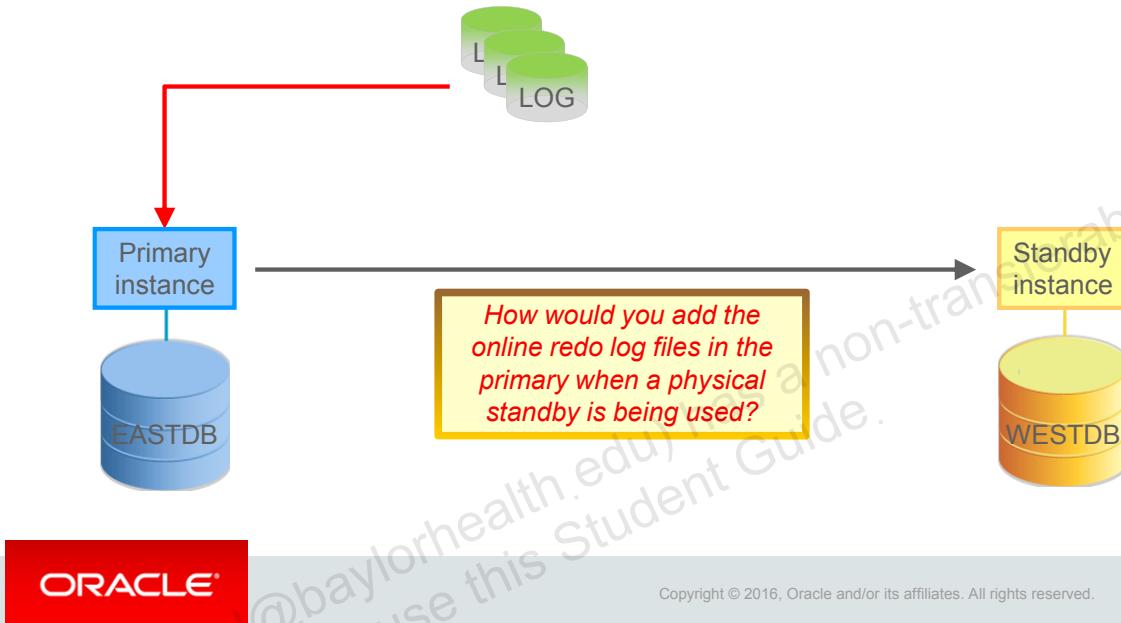
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

5. Use the ALTER DATABASE MOVE DATAFILE command to rename a data file in the standby.
6. Shut down the standby database.
7. Start and mount the standby database.
8. Restart Redo Apply.
9. The STANDBY_FILE_MANAGEMENT parameter can be reset after renaming or moving the data file.

You will practice this task in practice 6-3 titled “Moving the Location of a Data File Online.”

Scenario 4: Adding or Dropping a Redo File Group

- Assumption:
 - STNADBY_FILE_MANAGEMENT = AUTO



The configuration of the redo log and the standby redo log on a physical standby database should be reevaluated and adjusted as necessary after adding or dropping a log file group on the primary database.

Action Required on Physical Standby

1. Stop Redo Apply.
2. On the standby database, note that the STANDBY_FILE_MANAGEMENT database initialization parameter must be set to MANUAL.
3. Add or drop a log file group in the primary database.
4. Add or drop a log file group in the standby database.
5. Restore the STANDBY_FILE_MANAGEMENT database initialization parameter to its original state.
6. Restart Redo Apply.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When you add a log file to the primary database and want to add it to the physical standby database as well (or when you drop a log file from the primary and want to drop it from the physical), you must do the following:

1. Stop Redo Apply.
2. Set STANDBY_FILE_MANAGEMENT to MANUAL on the physical standby database.
3. Add the redo log files to (or drop them from) the primary database.
4. Add the redo log files to (or drop them from) the standby database.

Note: An online logfile group must always be manually cleared before it can be dropped from a physical standby database. For example:

```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

An online logfile group that has a status of CURRENT or CLEARING_CURRENT cannot be dropped from a physical standby database. An online logfile group that has this status can be dropped after a role transition.

5. Reset to AUTO afterward on the standby database.
6. Restart Redo Apply.

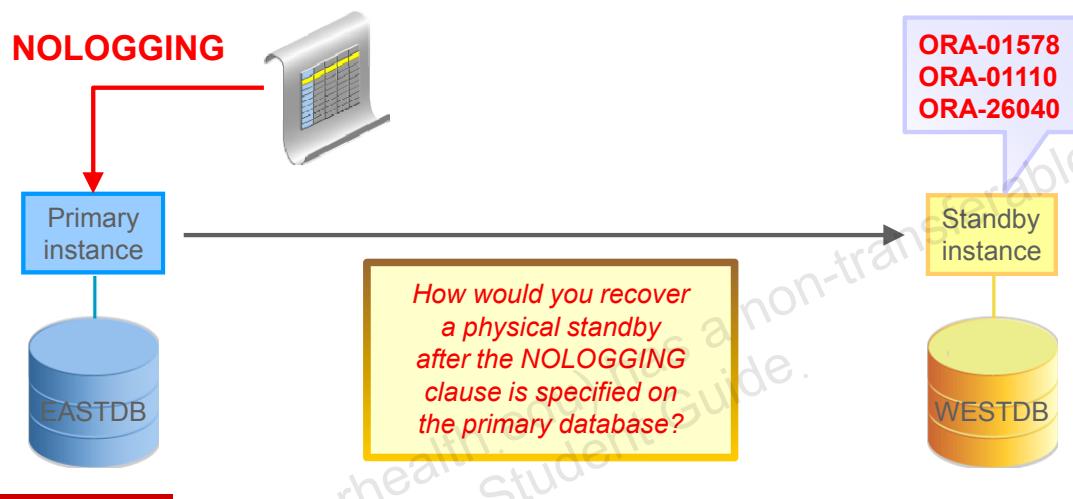
In Oracle RAC environments, keep the following in mind:

- When an online redo log group is added to a primary database, you must manually add an online redo log group to the standby database. It is not done automatically.
- When a new redo thread is added to a primary database, a new redo thread is automatically added to the standby. By default, the new thread is configured with two log groups of 100 MB each. This cannot be changed or overridden.
- When a new log group is added to an existing redo thread, a new log group is not automatically added to its existing thread.

You will practice this task in practice 6-4 titled “Adding Online and Standby Log Groups.”

Scenario 5: NOLOGGING Operations

- Assumptions:
 - Initially FORCE LOGGING is not enabled for any reasons.
 - A DML operation is performed using the NOLOGGING clause.
 - Then, FORCE LOGGING mode is configured.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When you perform a DML or DDL operation using the NOLOGGING or UNRECOVERABLE clause, the standby database is invalidated and may require substantial DBA administrative activities to repair. To avoid these problems, Oracle recommends that you always specify the FORCE LOGGING clause in the CREATE DATABASE or ALTER DATABASE statements. See the *Oracle Database Administrator's Guide*.

However, this statement will not repair an already invalidated database.

When the archived redo log file is copied to the standby site and applied to the physical standby database, a portion of the data file is unusable and is marked as being unrecoverable. When you either fail over to the physical standby database, or open the standby database for read-only access, and attempt to read the range of blocks that are marked as UNRECOVERABLE, you will see error messages similar to the following:

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

To recover after the NOLOGGING clause is specified, you need to copy the data file that contains the missing redo data from the primary site to the physical standby site. The next slide shows the recovery steps for the physical standby database.

Action Required on Physical Standby

1. Determine which data file should be copied.

- Query the primary database:

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
```

- Query the standby database:

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
```

- Compare the query results of the primary and standby databases:

NAME	UNRECOVERABLE
/oracle/dbs/tbs_1.dbf	5216

NAME	UNRECOVERABLE
/oracle/dbs/tbs_1.dbf	5186



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Determine which data files should be copied.

- Query the primary database:

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
```

```
NAME          UNRECOVERABLE
```

```
-----
```

```
/oracle/dbs/tbs_1.dbf 5216
```

```
/oracle/dbs/tbs_2.dbf 0
```

```
2 rows selected.
```

- Query the standby database:

```
SQL> SELECT NAME, UNRECOVERABLE_CHANGE# FROM V$DATAFILE;
```

```
NAME          UNRECOVERABLE
```

```
-----
```

```
/oracle/dbs/tbs_1.dbf 5186
```

```
/oracle/dbs/tbs_2.dbf 0
```

```
2 rows selected.
```

- Compare the query results of the primary and standby databases.

Compare the value of the UNRECOVERABLE_CHANGE# column in both query results. If the value of the UNRECOVERABLE_CHANGE# column in the primary database is greater than the same column in the standby database, then the data file must be copied from the primary site to the standby site. In this example, the value of the UNRECOVERABLE_CHANGE# column in the primary database for the tbs_1.dbf data file is greater, so you need to copy the tbs_1.dbf data file to the standby site.

Action Required on Physical Standby

2. On the primary site, back up the data file you need to copy to the standby site:

```
SQL> ALTER TABLESPACE TBS1 BEGIN BACKUP;
--- Copy the needed data file to a local directory ---
SQL> ALTER TABLESPACE TBS2 END BACKUP;
```

3. Copy the data file to the standby database.
4. On the standby database, restart Redo Apply:

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

5. If you receive a series of errors due to the archive gaps, manually resolve the gaps and repeat Step 4.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

2. On the primary site, back up the data file you need to copy to the standby site. Issue the following SQL statements:

```
SQL> ALTER TABLESPACE system BEGIN BACKUP;
SQL> EXIT;
```

Copy the needed data file to a local directory.

```
SQL> ALTER TABLESPACE system END BACKUP;
```

3. Copy the data file to the standby database.

Copy the data file that contains the missing redo data from the primary site to a location on the physical standby site where files related to recovery are stored.

4. On the standby database, restart Redo Apply. Issue the following SQL statement:

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

You might get the following error messages (possibly in the alert log) when you try to restart Redo Apply:

```
ORA-00308: cannot open archived log 'standby1'  
ORA-27037: unable to obtain file status  
SVR4 Error: 2: No such file or directory  
Additional information: 3  
ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get  
error below  
ORA-01152: file 1 was not restored from a sufficiently old backup  
ORA-01110: data file 1: '/oracle/dbs/stdby/tbs_1.dbf'
```

If you get the ORA-00308 error and Redo Apply does not terminate automatically, you can cancel recovery by issuing the following SQL statement from another terminal window:

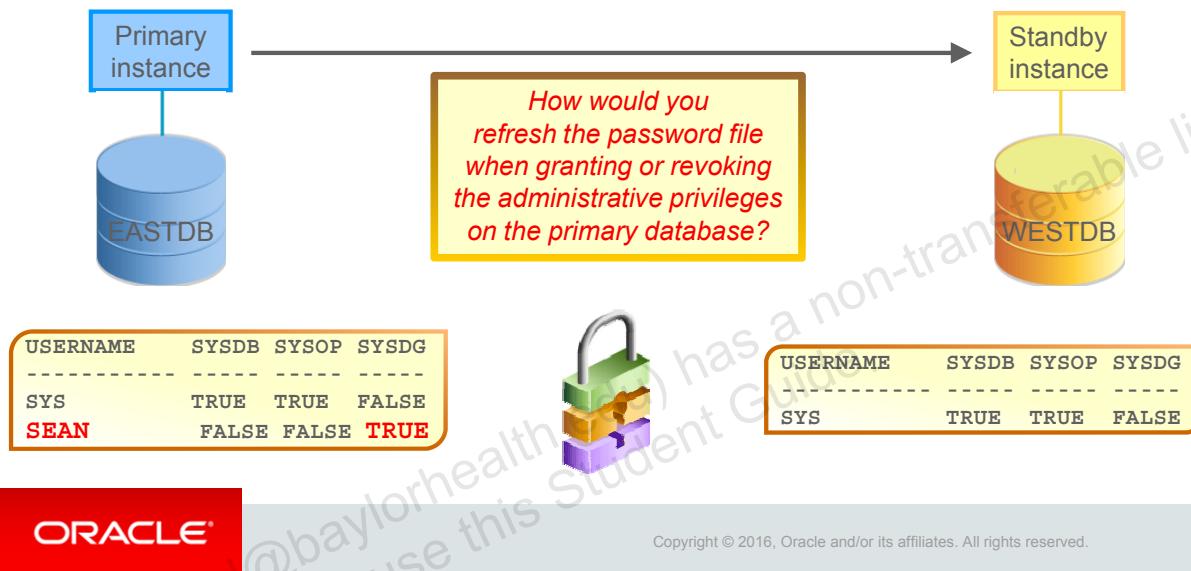
```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

These error messages are returned when one or more log files in the archive gap have not been successfully applied. If you receive these errors, manually resolve the gaps, and repeat Step 4.

See Section 7.6.3.1 Manual Gap Resolution in *Data Guard Concepts and Administration Guide* for information about manually resolving an archive gap.

Scenario 6: Refreshing the Password File

- Assumptions:
 - REMOTE_LOGIN_PASSWORDFILE is set to EXCLUSIVE.
 - The password file is located in an ASM disk group.
 - GRANT SYSDG TO SEAN on primary database



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If the REMOTE_LOGIN_PASSWORDFILE database initialization parameter is set to SHARED or EXCLUSIVE, the password file on a physical standby database must be replaced with a fresh copy from the primary database after granting or revoking administrative privileges or changing the password of a user with administrative privileges.

Failure to refresh the password file on the physical standby database may cause authentication of redo transport sessions or connections as SYSDG, SYSDBA, or SYSOPER to the physical standby database to fail.

Action Required on Physical Standby

- On the primary, grant the SYSDG privilege to the user:

```
SQL> GRANT SYSDG to SEAN;
```

```
SQL> SELECT USERNAME, SYSDBA, SYSOPER, SYSDG FROM V$PFILE_USERS;
```

USERNAME	SYSDB	SYSOP	SYSDG
SYS	TRUE	TRUE	FALSE
SEAN	FALSE	FALSE	TRUE

- The password file on a physical standby database must be replaced with a fresh copy.
- If you have stored the password file in an Oracle ASM disk group in the standby database, then you must copy the updated password file from the primary database to the Oracle ASM location at the standby database.



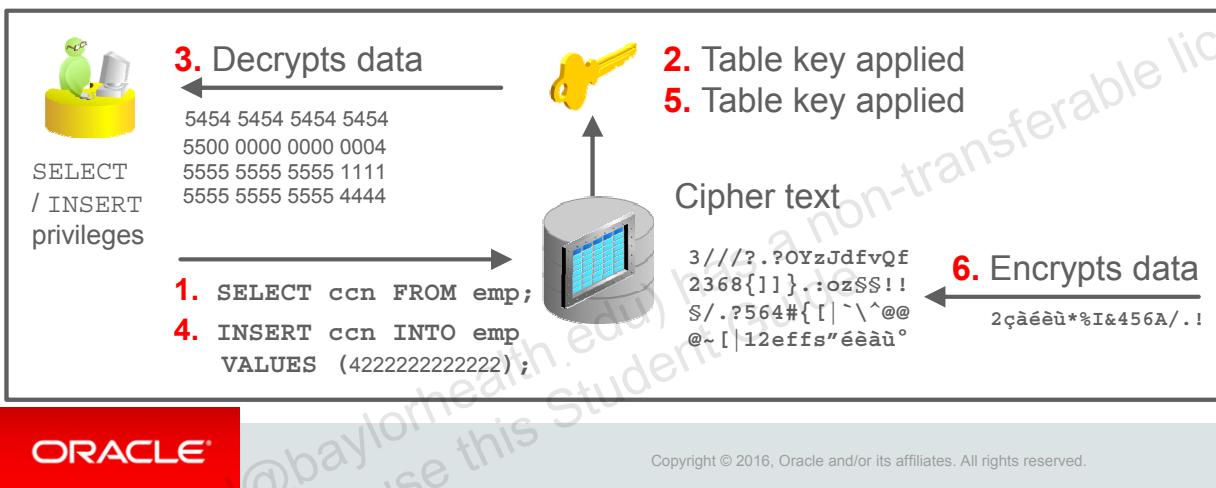
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If you have stored the password file in an Oracle ASM disk group in the standby database, then you must copy the updated password file from the primary database to the Oracle ASM location at the standby database.

You practiced this task in practice 5-1 titled “Using a Shared Password File in an ASM Disk Group.”

Review: Transparent Data Encryption

- Encrypts data in:
 - Data files (tablespaces, columns, indexes)
 - Redo log and archive log files
 - Memory (only for column encryption)
 - File backups
- Manages keys automatically



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Transparent Data Encryption (TDE) is available with Oracle Advanced Security and provides easy-to-use protection for your data without requiring changes to your applications. TDE allows you to encrypt sensitive data in individual columns or entire tablespaces without having to manage encryption keys. TDE does not affect access controls, which are configured using database roles, secure application roles, system and object privileges, views, Virtual Private Database (VPD), Oracle Database Vault, and Oracle Label Security. Any application or user that previously had access to a table will still have access to an identical encrypted table.

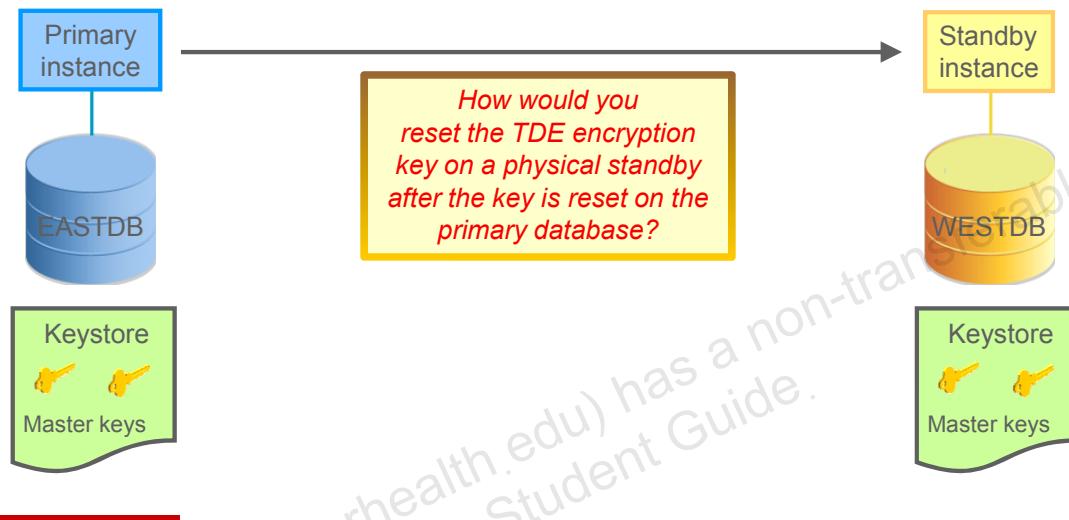
TDE is designed to protect data in storage, but does not replace proper access control.

TDE is transparent to existing applications. Encryption and decryption occurs at different levels depending on whether it is tablespace or column level, but in either case, encrypted values are not displayed and are not handled by the application. For example, with TDE, applications designed to display a 16-digit credit card number do not have to be recoded to handle an encrypted string that may have many more characters.

TDE eliminates the ability of anyone who has direct access to the data files to gain access to the data by circumventing the database access control mechanisms. Even users with access to the data file at the operating system level cannot see the data unencrypted. TDE stores the master key outside the database in an external security module, thereby minimizing the possibility of both personally identifiable information (PII) and encryption keys being compromised. TDE decrypts the data only after database access mechanisms have been satisfied.

Scenario 7: Resetting the TDE Master Encryption Key

- Assumption:
 - The TDE master encryption key is reset on the primary database using the ADMINISTER KEY MANAGEMENT ALTER KEYSTORE PASSWORD command.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The database encryption wallet on a physical standby database must be replaced with a fresh copy of the database encryption wallet from the primary database whenever the TDE master encryption key is reset on the primary database.

Failure to refresh the database encryption wallet on the physical standby database will prevent access to encrypted columns on the physical standby database that are modified after the master encryption key is reset on the primary database.

Summary

In this lesson, you should have learned how to describe the primary database changes that require manual intervention at a physical standby database.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 6: Overview

This practice covers the following topics:

- Creating an Archival Backup
- Using Transportable Tablespaces with a Physical Standby Database
- Moving the Location of a Data File Online
- Adding Online and Standby Log Groups



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

7

Effective Client Failover Using Application Continuity

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

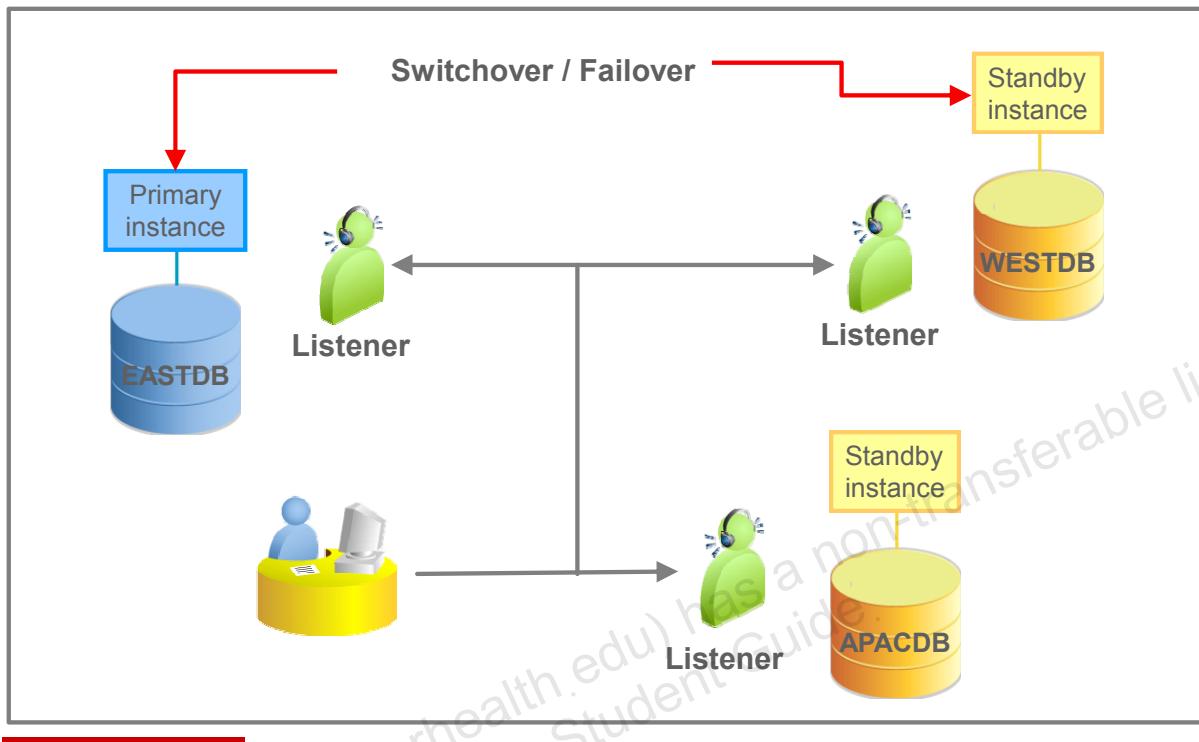
After completing this lesson, you should be able to:

- Understand effective client failover
- Describe client failover considerations
- Describe the purpose and architecture of Application Continuity
- Describe how applications can leverage Application Continuity
- Configure an Oracle RAC database to enable the use of Application Continuity to provide transparent failover in the case of node or instance failure
- Configure Oracle Data Guard to enable the use of Application Continuity to provide effective client failover in case of site failure



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Understanding Client Connectivity in a Data Guard Configuration

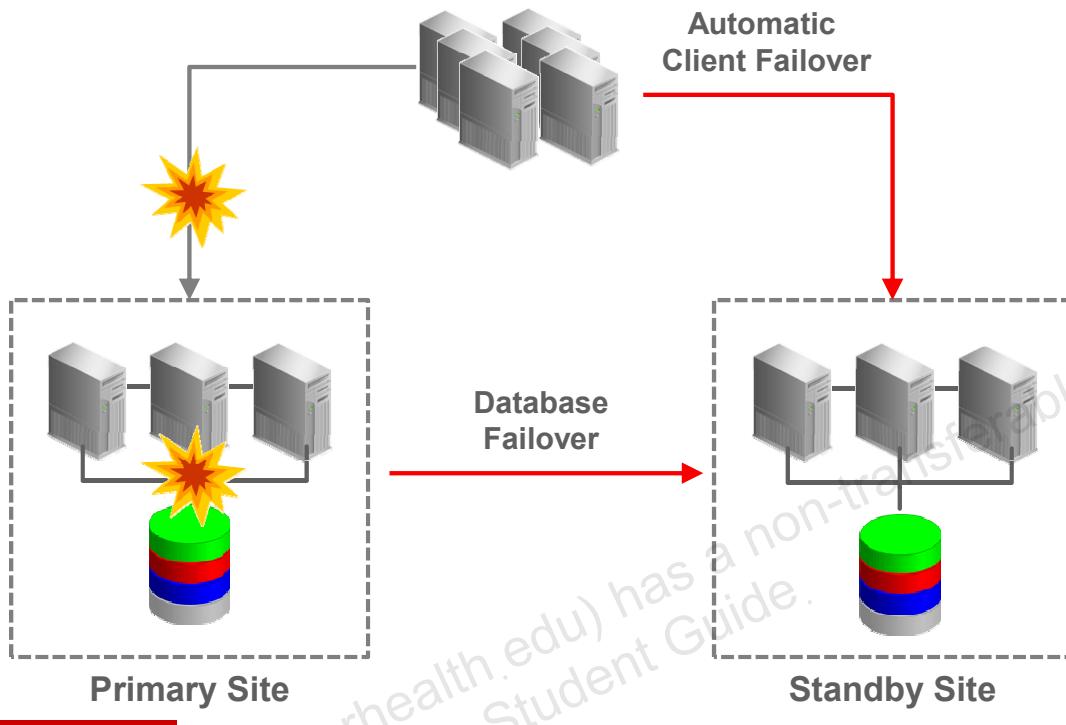


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A physical standby database not using Active Data Guard maintains the database in a mounted mode and would not normally allow listener connections for normal users. However, with Active Data Guard, the database is open read-only to allow reporting against the physical standby. Also, the addition of logical standby databases that are open read/write would allow a client to successfully connect using standard database authentication schemes such as username and password. It is important that the client connect to the appropriate environment.

Effective Client Failover: Overview



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The ability to quickly perform a database failover is only the first requirement for high availability. Applications must also be able to quickly drop their connections to a failed primary database and quickly reconnect to the new primary database.

With the Oracle Data Guard broker, user-written database triggers are required to implement automatic failover as follows:

- A startup trigger is used to start database services on the new primary database.
- A role-change trigger is used to publish a Fast Application Notification (FAN) Oracle Notification Server (ONS) event to break JDBC clients still connected to the original primary database out of a TCP timeout.

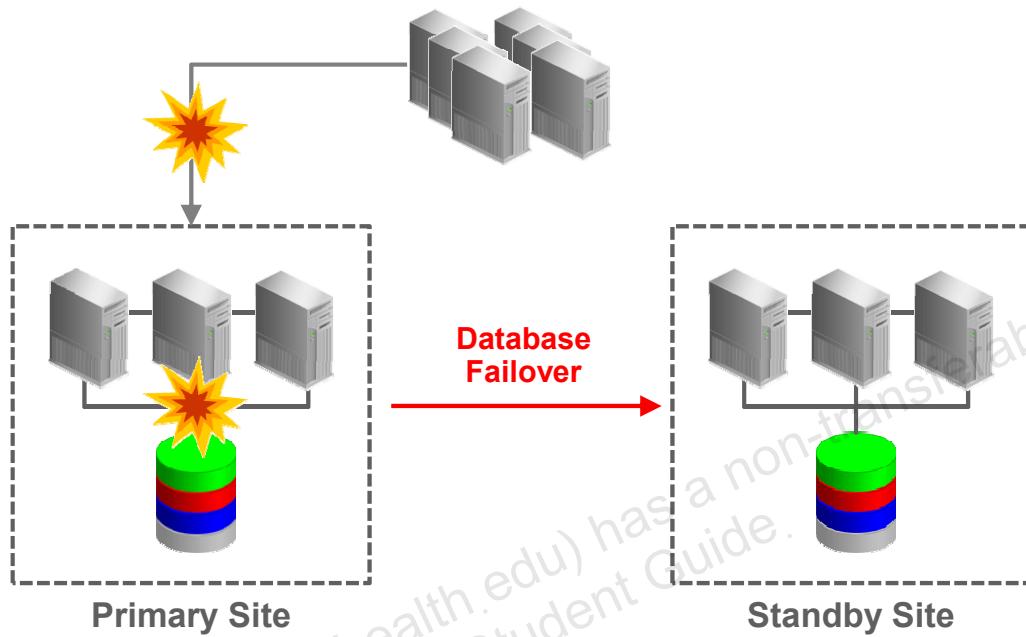
You can automate fast failover of applications to a new primary database without the need for user-written triggers. You must use the Data Guard broker to use this feature.

Automatic fast failover of application clients to a new primary database following failover requires the following:

- Fast database failover
- Restarting database services on the new primary database
- Notifying clients that a failure has occurred, to break them out of TCP timeout and redirect them to the new primary database
- Handling in-flight transactions
- Replying incomplete requests

Client Failover Considerations

1. Connecting to the Appropriate Environment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide shows an example of the role transition service, which can be switchover, manual failover, or fast-start failover.

A series of effective client failover considerations will be discussed throughout this lesson. The first topic is how to connect to the appropriate environment using the database services in case of a role transition such as switchover or failover.

Connecting to the Appropriate Environment

- Clients who send connection requests to the wrong host might receive an error message such as the following:

ORA-01033: ORACLE initialization or shutdown in progress

- Use Database Services to prevent clients from connecting to the wrong database in the Data Guard configuration.
 - Clients connect to database services instead of database instances.
 - Listeners use registration details to determine which instances support a particular service at a particular moment in time.
 - Listeners then direct connection requests to the correct instances; otherwise, the appropriate error is returned.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Clients who send connection requests to the wrong host might be connected to the wrong database instance, or they might receive an error message such as the following:

ORA-01033: ORACLE initialization or shutdown in progress

You can prevent clients from connecting to the wrong database by using database services.

Managing Database Services

- In a standalone environment:
 - Use the DBMS_SERVICE package.
 - Role changes can be managed by the AFTER STARTUP trigger to start services appropriate to the database role.
- In a clustered environment:
 - Use SRVCTL to configure the role-based services on each database in the Data Guard configuration.
 - Role changes can be managed by the Data Guard broker to automatically start services appropriate to the database role. The service is started when ROLE matches the current role of the database and MANAGEMENT POLICY is set to AUTOMATIC.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Database services are implemented with the DBMS_SERVICE package in standalone environments. This package provides for the creation, deletion, starting, and stopping of services for a single database instance.

In the Oracle RAC environments, the SRVCTL utility should be used to manage role-based services, instead of DBMS_SERVICE.

Example: Creating Services for Data Guard

- Create the Database Services in a Data Guard environment:

```
DBMS_SERVICE.CREATE_SERVICE( -  
    SERVICE_NAME => 'DG_PROD', -  
    NETWORK_NAME => 'DG_PROD', -  
    FAILOVER_METHOD => 'BASIC', -  
    FAILOVER_TYPE => 'SELECT', -  
    FAILOVER_RETRIES => 180, -  
    FAILOVER_DELAY => 1);
```

```
DBMS_SERVICE.CREATE_SERVICE( -  
    SERVICE_NAME => 'DG_RTQ', -  
    NETWORK_NAME => 'DG_RTQ');
```

```
DBMS_SERVICE.CREATE_SERVICE('DG_LSBY', 'DG_LSBY');
```

```
DBMS_SERVICE.CREATE_SERVICE('DG_SNAP', 'DG_SNAP');
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

By using the DBMS_SERVICE.CREATE_SERVICE procedure, you define a service to represent each role or state in which the databases in your Data Guard configuration can operate. A service created with DBMS_SERVICE.CREATE_SERVICE is not aware of the actual database role. This functionality is only available with the SRVCTL interface for both Real Application Clusters and Oracle Restart. In the examples in the slide, the service name is being used to imply the database role, which could be inaccurate after role transitions. The CREATE_SERVICE procedure creates a service name in the data dictionary.

You should create services for the physical standby database when it is opened in read-only mode (using real-time query) and when it is converted into a snapshot standby database. The DBMS_SERVICE.CREATE_SERVICE procedure will fail to execute on a physical standby database even if it is open read-only. The service must be created on the primary and allowed to propagate to the physical standby. In addition, create a service for logical standby databases in your configuration.

Note: Database service names in the slide are examples.

Example: Using AFTER STARTUP Trigger

```
CREATE TRIGGER MANAGE_SERVICES AFTER STARTUP ON DATABASE
DECLARE
    ROLE VARCHAR2(30);
    OMODE VARCHAR2(30);
BEGIN
    SELECT DATABASE_ROLE INTO ROLE FROM V$DATABASE;
    SELECT OPEN_MODE INTO OMODE FROM V$DATABASE;
    IF ROLE = 'PRIMARY' THEN
        DBMS_SERVICE.START_SERVICE ('DG_PROD');
    ELSIF ROLE = 'PHYSICAL STANDBY' THEN
        IF OMODE LIKE 'READ ONLY%' THEN
            DBMS_SERVICE.START_SERVICE ('DG_RTQ');
        END IF;
    ELSIF ROLE = 'LOGICAL STANDBY' THEN
        DBMS_SERVICE.START_SERVICE ('DG_LSBY');
    ELSIF ROLE = 'SNAPSHOT STANDBY' THEN
        DBMS_SERVICE.START_SERVICE ('DG_SNAP');
    END IF;
END;
/
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The AFTER STARTUP trigger is invoked when the database is opened. The trigger checks the database role and the open mode of the database and, based on the values, invokes the DBMS_SERVICE.START_SERVICE procedure to start the appropriate service. The CREATE TRIGGER SQL command must be run only in the primary database. It will be replicated to all standby databases.

Starting with Oracle Database 12c Release 1 (12.1), there is a new role value for the DATABASE_ROLE column of the V\$DATABASE view. The value FAR SYNC has been added. No additional services should be started for this role type.

Note: The trigger shown in the slide is an example using the service names defined earlier in the lesson. Additional trigger functionality may be necessary based on your circumstances. It is not necessary to employ an AFTER STARTUP trigger if the services are managed by Oracle Clusterware.

Example: Configuring Services for JDBC Clients

- OrderEntry: Read/write service that always runs on the database with the primary role

```
$ srvctl add service -d EASTDB -s OrderEntry -l PRIMARY  
-r EASTDB1,EASTDB2 -q FALSE -e NONE -m BASIC -w 0 -z 0  
$ srvctl add service -d WESTDB -s OrderEntry -l PRIMARY  
-r WESTDB1,WESTDB2 -q FALSE -e NONE -m BASIC -w 0 -z 0
```

- OrderReport: Read-only service that always runs on an Active Data Guard standby

```
$ srvctl add service -d EASTDB -s OrderReport  
-l PHYSICAL_STANDBY -r EASTDB1,EASTDB2  
-q FALSE -e NONE -m BASIC -w 0 -z 0  
$ srvctl add service -d WESTDB -s OrderReport  
-l PHYSICAL_STANDBY -r WESTDB1,WESTDB2  
-q FALSE -e NONE -m BASIC -w 0 -z 0
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example illustrates creating the database services for JDBC applications. Do not enable HA notifications (-q) or TAF (-e) on services for JDBC database services that are FAN enabled:

- OrderEntry is a read/write service that always runs on the database with the primary role.
- OrderReport is a read-only service that always runs on Active Data Guard standby databases. A role transition can stop this service.

The following attributes are used for Transparent Application Failover:

- -e: The type of failover. The possible values are NONE, SESSION, or SELECT.
- -m: Setting for fast failover from the primary site to the standby site. The possible values are NONE or BASIC.
- -w: The amount of time in seconds to wait between connect attempts
- -z: The number of times to attempt to connect after a failover
- -q: Send Oracle Advanced Queuing (AQ) HA notifications. For standalone servers, this is applicable in Oracle Data Guard environment only.

For OCI applications, see the Oracle White Paper titled “*Client Failover Best Practices for Highly Available Oracle Databases*.”

Example: Configuring JDBC URL

- Configure JDBC clients to use a connect descriptor that includes an address list with the SCAN address for each site:

```
"jdbc:oracle:thin:@\" +  
"(DESCRIPTION_LIST=\" +  
"(LOAD_BALANCE=off)\" +  
"(FAILOVER=on)\" +  
"(DESCRIPTION=\" +  
"(ADDRESS_LIST=\" +  
"(LOAD_BALANCE=on)\" +  
"(ADDRESS=(PROTOCOL=TCP)(HOST=cluster01-scan)(PORT=1521))\" +  
"(CONNECT_DATA=(SERVICE_NAME=OrderEntry)))\" +  
"(DESCRIPTION=\" +  
"(ADDRESS_LIST=\" +  
"(LOAD_BALANCE=on)\" +  
"(ADDRESS=(PROTOCOL=TCP)(HOST=cluster02-scan)(PORT=1521))\" +  
"(CONNECT_DATA=(SERVICE_NAME=OrderEntry))))\";
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

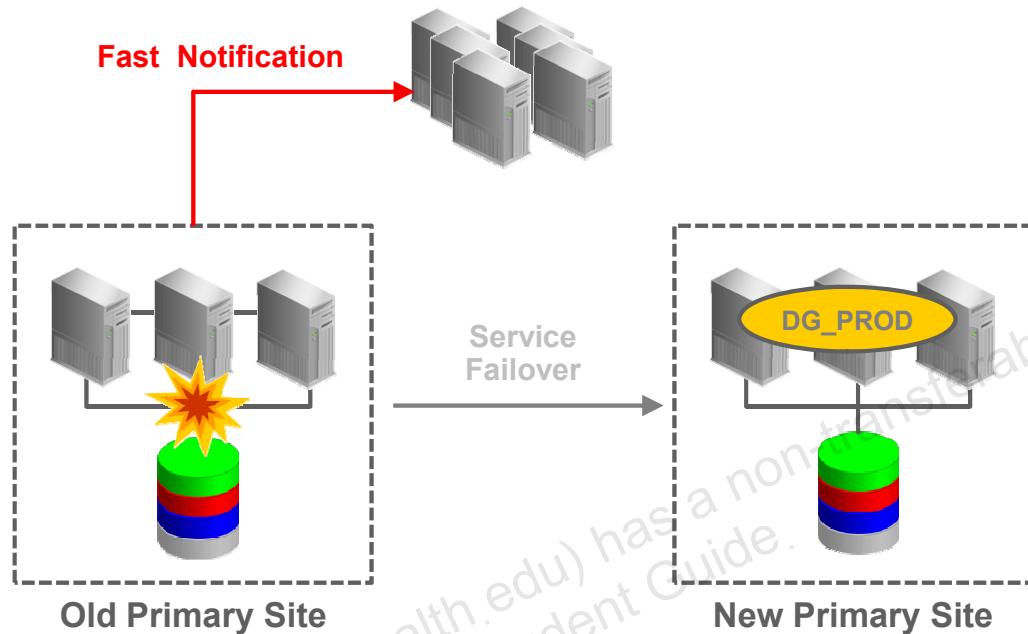
After a connection is made using the JDBC URL:

- Oracle Net contacts DNS and resolves primary SCAN to a total of three IP addresses.
- It randomly picks up one of the three IP addresses and attempts to make a connection.
- If the connection to primary site is unsuccessful, it then contacts DNS and resolves standby SCAN to three addresses.
- It then randomly picks up one of the IP addresses and tries to connect.

Note: For best performance while creating new connections, the Oracle Net alias should have LOAD_BALANCE=OFF for the DESCRIPTION_LIST so that DESCRIPTIONS are tried in an ordered list, top to bottom. With this configuration, the second DESCRIPTION is only attempted if all connection attempts to the first DESCRIPTION have failed. If role transitions occur frequently, consider randomizing connections between DESCRIPTION and enable LOAD_BALANCE at the DESCRIPTION_LIST level. For OCI applications, see the Oracle White Paper titled “Client Failover Best Practices for Highly Available Oracle Databases.”

Client Failover Considerations

2. Fast Notification of Clients



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

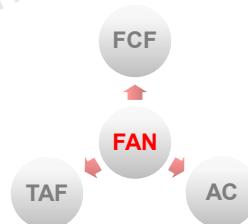
The diagram shows the following scenarios:

- The database role was transitioned.
- The DG_PROD service was relocated to the new primary database as a result of database failover.

The next effective client failover consideration is how to notify clients about the status changes of the database and service. You will discuss a notification mechanism called FAN.

Fast Application Notification (FAN)

- FAN is used by RAC to notify other processes about configuration and service level information.
 - This includes service status changes like UP or DOWN events.
 - UP and DOWN events can apply to instances, services, and nodes.
- The Data Guard Broker also publishes FAN events.
 - FAN notification is sent after failover for databases configured with Cluster Ready Services (CRS) to notify applications that the new primary database is available.
 - Upon receipt of that event, FAN client subscribers can automatically reconnect to the service started on the new primary database.
- Using FAN events eliminates:
 - Applications waiting on TCP timeouts
 - Time wasted in processing the last result at the client after a failure has occurred



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

FAN is a high-availability notification mechanism that Oracle RAC uses to notify other processes about configuration and service level information that includes service status changes, such as UP or DOWN events.

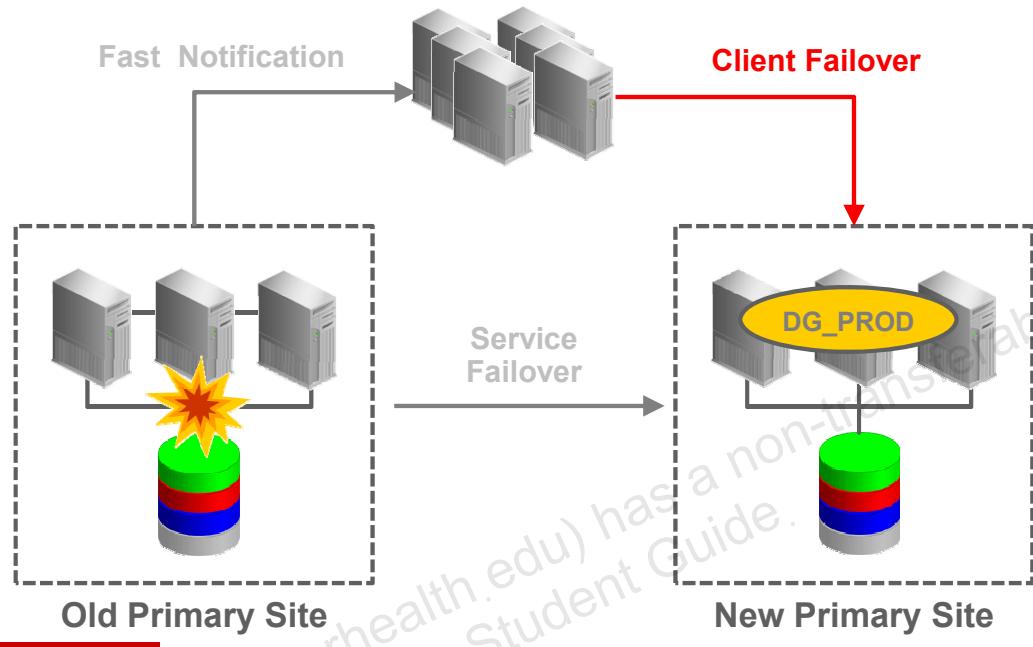
The Data Guard broker also publishes FAN events for databases configured with Oracle Clusterware. After failover is complete, FAN notification is sent to notify applications that the new primary database is available.

Using FAN events eliminates applications waiting on TCP timeouts, time wasted processing the last result at the client after a failure has occurred, and time wasted executing work on slow, hung, or dead nodes. For cluster configuration changes, the Oracle RAC high-availability framework publishes a FAN event immediately when a state change occurs in the cluster. Instead of waiting for the application to time out against the database and detect a problem, applications can receive FAN events and react immediately. With FAN, in-flight transactions are immediately terminated and the client notified when the instance fails.

For applications that do not support FAN, see the Oracle White Paper titled “*Client Failover Best Practices for Highly Available Oracle Databases*.”

Client Failover Considerations

3. Automatic Client Failover



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram shows the following scenarios:

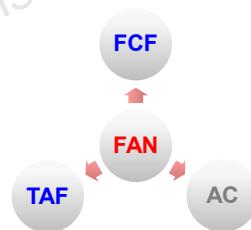
- The database role was transitioned.
- The DG_PROD service was relocated as a result of database failover.
- The Data Guard Broker and Clusterware notified the application about the status changes of the database and service.

The next effective client failover consideration is how to implement automatic client failover by leveraging FAN events.

Implementing Client Failover with FAN

You can take advantage of FAN events in the following ways:

- Applications respond to FAN events without programmatic changes if using Oracle-integrated database clients:
 - Oracle Database JDBC
 - Oracle Database Oracle Call Interface (OCI)
 - Oracle Database ODP.NET
- Clients that receive FAN events can be configured for Fast Connection Failover (FCF) or Transparent Application Failover (TAF) to automatically connect to a new primary database.
- Clients connect to the new primary database using an Oracle Net connect descriptor configured for connect-time failover.

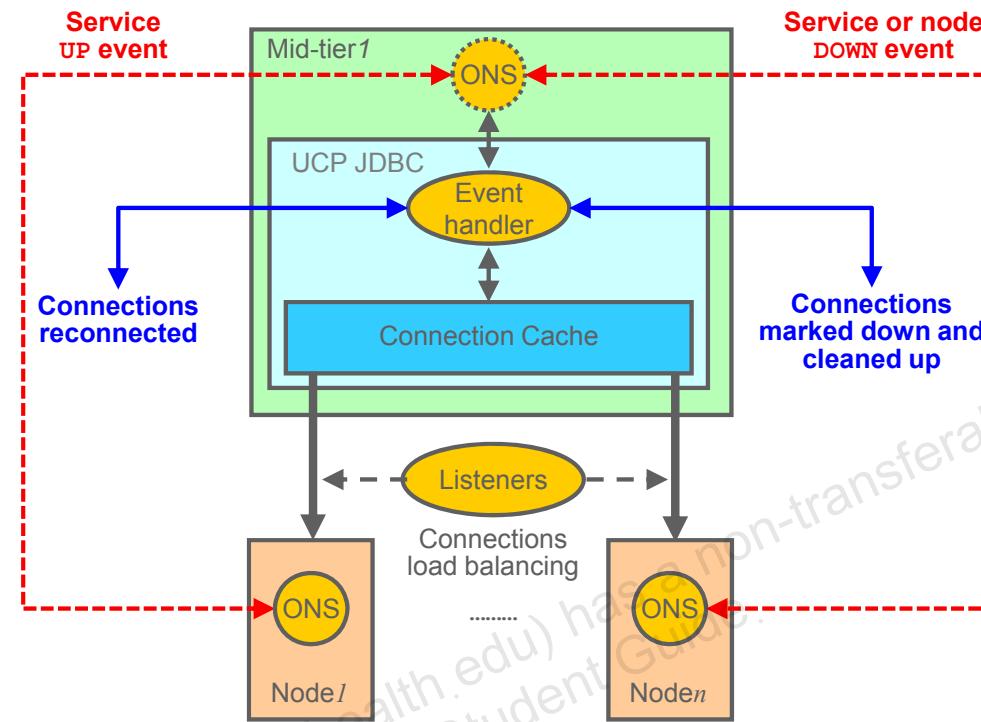


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can take advantage of FAN events in the following ways:

- Your application can use FAN without programmatic changes if you use an integrated Oracle client. The integrated clients for FAN events include Oracle JDBC Universal Connection Pool, ODP.NET connection pool, OCI session pool, Oracle WebLogic Server Active Gridlink for Oracle RAC, and OCI and ODP.NET clients.
- This includes applications that use Fast Connection Failover, Transparent Application Failover, Application Continuity, or Transaction Guard. The integrated Oracle clients must be Oracle Database 10g release 2 or later to take advantage of the FAN high-availability events.
- Applications can use FAN programmatically by using the JDBC and Oracle RAC FAN application programming interface (API) or by using callbacks with OCI and ODP.NET to subscribe to FAN events and to execute event handling actions upon the receipt of an event.
- Clients can connect to the new instance or primary database using an Oracle Net connect descriptor configured for connect-time failover.

Example: Fast Connection Failover (UCP JDBC)



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Universal Connection Pool (UCP) provides a tight integration with Oracle RAC database features like FCF. Basically, FCF is a FAN client implemented through the connection pool. FCF quickly and automatically recovers lost or damaged connections. This automatic connection management results from FAN events received by the local ONS daemon, or by a remote ONS if a local one is not used, and handled by a special event handler thread. Both JDBC Thin and JDBC OCI drivers are supported.

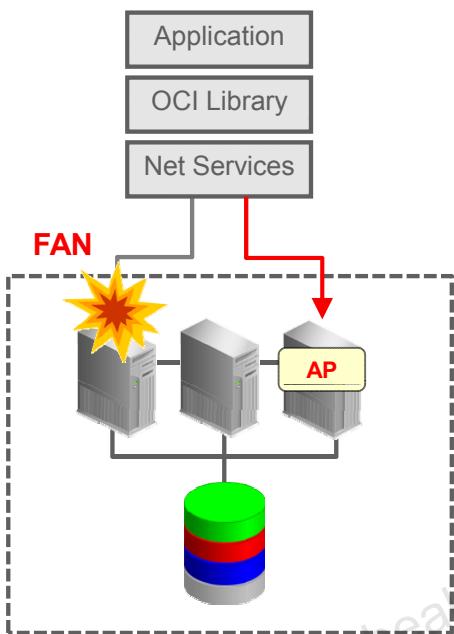
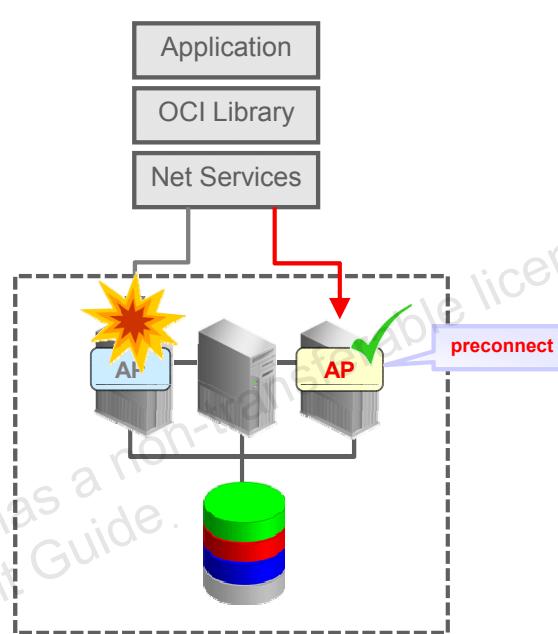
Therefore, if UCP and FCF are enabled, your Java program automatically becomes an ONS subscriber without having to manage FAN events directly.

Whenever a service or node down event is received by the mid-tier ONS, the event handler automatically marks the corresponding connections as down and cleans them up. This prevents applications that request connections from the cache from receiving invalid or bad connections.

Whenever a service up event is received by the mid-tier ONS, the event handler recycles some unused connections and reconnects them using the event service name. The number of recycled connections is automatically determined by the connection cache. Because the listeners perform connection load balancing, this automatically rebalances connections across the preferred instances of the service without waiting for connection requests or retries.

For more information see *Oracle Universal Connection Pool for JDBC Developer's Guide*.

Example: Transparent Application Failover

TAF Basic**TAF Preconnect**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

TAF is a runtime feature of the OCI driver. It enables your application to automatically reconnect to the service if the initial connection fails. During the reconnection, although your active transactions are rolled back, TAF can optionally resume the execution of a `SELECT` statement that was in progress. TAF supports two failover methods:

- With the `BASIC` method, the reconnection is established at failover time. After the service has been started on the nodes, the initial connection is made. The listener establishes the connection, and your application accesses the database until the connection fails for any reason. Your application then receives an error the next time it tries to access the database. Then, the OCI driver reconnects to the same service, and the next time your application tries to access the database, it transparently uses the newly created connection. TAF can be enabled to receive FAN events for faster down events detection and failover.
- The `PRECONNECT` method is similar to the `BASIC` method except that it is during the initial connection that a shadow connection is also created to anticipate the failover. TAF guarantees that the shadow connection is always created on the available instances of your service by using an automatically created and maintained shadow service.

Note: Optionally, you can register TAF callbacks with the OCI layer. These callback functions are automatically invoked at failover detection and allow you to have some control of the failover process. For more information, refer to the *Oracle Call Interface Programmer's Guide*.

Example: TAF Basic Configuration with FAN

- OrderEntry: Read/write service that always runs on the database with the primary role

```
$ srvctl add service -d EASTDB -s OrderEntry -l PRIMARY  
-r EASTDB1,EASTDB2 -q TRUE -e SESSION -m BASIC -w 10 -z 150  
$ srvctl add service -d WESTDB -s OrderEntry -l PRIMARY  
-r WESTDB1,WESTDB2 -q TRUE -e SESSION -m BASIC -w 10 -z 150
```

- OrderReport: Read-only service that always runs on an Active Data Guard standby

```
$ srvctl add service -d EASTDB -s OrderReport  
-l PHYSICAL_STANDBY -r EASTDB1,EASTDB2  
-q TRUE -e SESSION -m BASIC -w 10 -z 150  
$ srvctl add service -d WESTDB -s OrderReport  
-l PHYSICAL_STANDBY -r WESTDB1,WESTDB2  
-q TRUE -e SESSION -m BASIC -w 10 -z 150
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows how to create database services for OCI clients with HA notifications (-q) and server-side TAF (-e) enabled. The example refers to an Oracle RAC primary and standby database:

- OrderEntry is a read/write service that always runs on the database with the primary role.
- OrderReport is a read-only service that always runs on Active Data Guard standby databases. A role transition can stop this service.

The following attributes are used for TAF:

- -e: The type of failover. The possible values are NONE, SESSION, or SELECT.
- -m: Setting for fast failover from the primary site to the standby site. The possible values are NONE or BASIC.
- -w: The amount of time in seconds to wait between connect attempts
- -z: The number of times to attempt to connect after a failover
- -q: Send Oracle Advanced Queuing (AQ) HA notifications. For stand-alone servers, it is applicable in Oracle Data Guard environment only.

Example: TAF Basic Configuration with FAN

- Run the following SQL at the primary database:

```
EXECUTE DBMS_SERVICE.CREATE_SERVICE('OrderReport','OrderReport', -  
NULL, NULL, TRUE, 'BASIC', 'SESSION', 150, 10, NULL);
```

- Start the TAF enabled primary role service:

```
$ srvctl start service -db eastdb -s OrderEntry
```

- Start the TAF enabled standby role service:

```
$ srvctl start service -db west -s OrderReport
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In addition to creating the database service "OrderReport" on both clusters, the SQL statement in the slide must also be run on the primary database so that the service definition is transmitted via the redo stream and applied to the physical standby database. The examples in the slide illustrate how to create role-based services with server-side TAF enabled. Any OCI client that connects to a service that has the TAF attributes set implicitly inherits those attributes. There is no need to configure TAF at the client side in the tnsnames.ora file.

Example: tnsnames.ora for OCI Applications

- Oracle Net alias should specify both the primary and standby SCAN hostnames:

```
SALES=
  (DESCRIPTION_LIST=
    (LOAD_BALANCE=off) (FAILOVER=on)
    (DESCRIPTION=
      (CONNECT_TIMEOUT=5) (TRANSPORT_CONNECT_TIMEOUT=3) (RETRY_COUNT=3)
      (ADDRESS_LIST=
        (LOAD_BALANCE=on)
        (ADDRESS= (PROTOCOL=TCP) (HOST=cluster01-scan) (PORT=1521)))
      (CONNECT_DATA= (SERVICE_NAME=OrderEntry)))
    (DESCRIPTION=
      (CONNECT_TIMEOUT=5) (TRANSPORT_CONNECT_TIMEOUT=3) (RETRY_COUNT=3)
      (ADDRESS_LIST=
        (LOAD_BALANCE=on)
        (ADDRESS= (PROTOCOL=TCP) (HOST=cluster02-scan) (PORT=1521)))
      (CONNECT_DATA= (SERVICE_NAME=OrderEntry))))
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a new connection is made using the Oracle Net alias in the slide, the following logic is used:

- Oracle Net contacts DNS and resolves cluster01-scan to a total of three IP addresses.
- Oracle Net randomly picks one of the three IP addresses and attempts to make a connection. If the connection attempt to the IP address does not respond in 3 seconds (TRANSPORT_CONNECT_TIMEOUT), the next IP address is attempted. All three IP addresses will be tried a total of four times (initial attempt plus RETRY_COUNT in the example).
- If the connection to primary site is unsuccessful, it then contacts DNS and resolves cluster02-scan to three addresses.
- The same sequence is performed for the standby cluster02-scan as it was for the cluster01-scan.

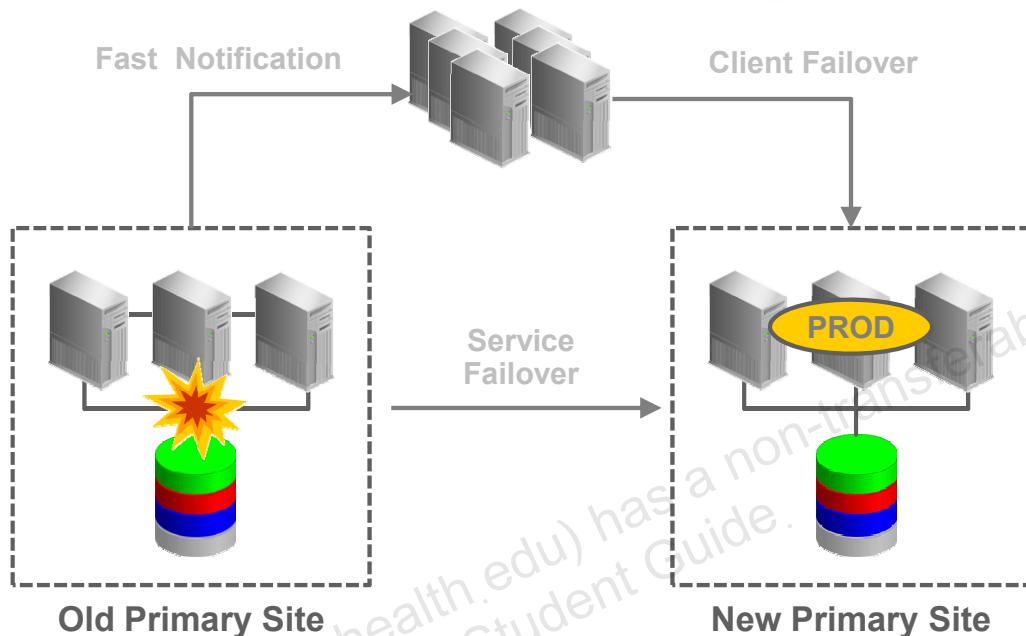
The following parameters are available in Oracle Database 11g Release 2 or later:

- CONNECT_TIMEOUT controls the overall time to connect to the service.
- TRANSPORT_CONNECT_TIMEOUT is the amount of time for the TCP connection to complete. CONNECT_TIMEOUT is set to a value slightly greater than TRANSPORT_CONNECT_TIMEOUT.
- RETRY_COUNT parameter specifies the number of times an address list is traversed before the connection attempt is terminated.

Client Failover Considerations

4. Handling In-Flight Transactions

Did the last transaction commit?



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram shows the following scenarios:

- The database role was transitioned.
- The DG_PROD service was relocated as a result of database failover.
- The Data Guard Broker and Clusterware notified the applications about the status changes of the database and service.
- Applications took actions to perform automatic client failover using FCF or TAF.

The next effective client failover consideration is how to handle the in-flight transactions after failover is complete. First discuss a way to check the commit outcome of the last in-flight transactions.

Introducing Transaction Guard

Oracle Database 12c introduces two fundamental capabilities for ensuring continuity of applications after database outages:

- **Transaction Guard:** A new reliable protocol and API that returns the outcome of the last transaction after a recoverable error has occurred
- Application Continuity: A feature that attempts to mask database session outages by recovering the in-flight work for requests submitted to the database



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c introduces two fundamental capabilities for ensuring continuity of applications after database outages:

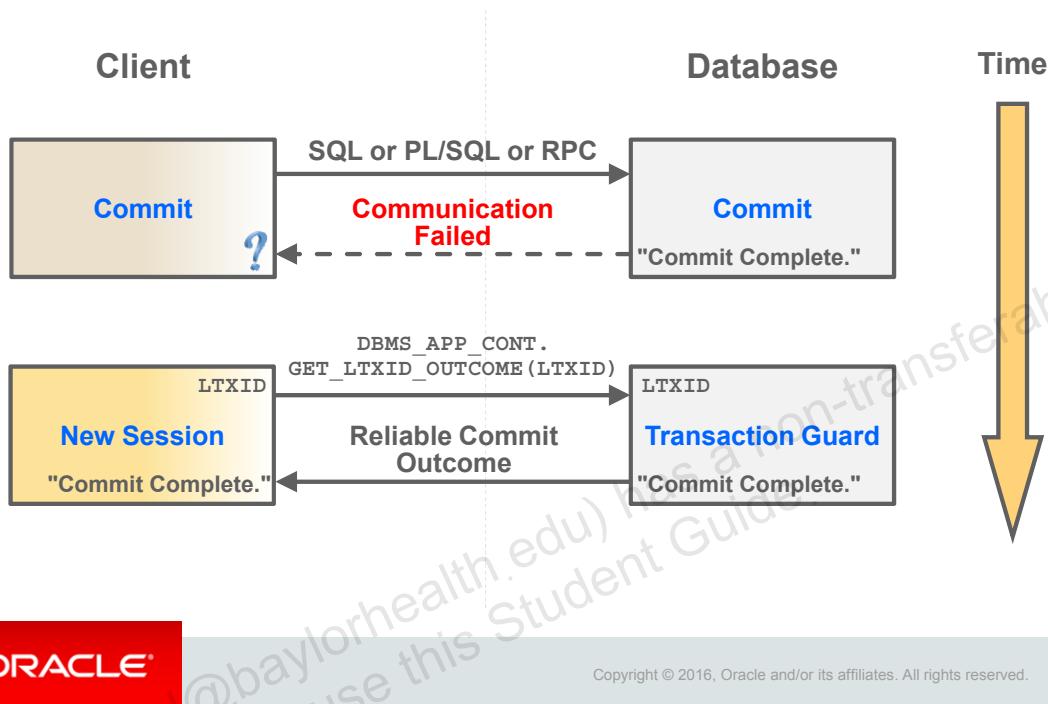
1. A foolproof way for applications to know the outcome of transactions
2. The ability to mask outages by reconnecting to the database and replaying the workload

These capabilities are provided by two new features. The first feature is Transaction Guard:

- Transaction Guard is an API that applications use in error handling. It is a new and reliable way to return the outcome of the last transaction after a recoverable error has occurred. By itself, Transaction Guard can dramatically improve the end-user experience by erasing the doubt over whether the last transaction was committed or not.

How Transaction Guard Works

Transaction Guard is a reliable protocol and API that enables applications to know the outcome of the last transaction.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In the standard commit case, the database commits a transaction and returns a success message to the client. In the illustration shown in the slide, the client submits a commit statement and receives a message stating that communication failed. This type of failure can occur due to several reasons, including a database instance failure or network outage. In this scenario, without Transaction Guard, the client does not know the outcome of the transaction.

Oracle Database solves the problem by using a globally unique identifier called a logical transaction ID (LTXID). When the application is running, both the database and the client hold the logical transaction ID. The database supplies the client with a logical transaction ID at authentication and at each round trip from the client driver that executes one or more commit operations.

When a recoverable outage occurs, the logical transaction ID uniquely identifies the last database transaction submitted on the session that failed. A new PL/SQL interface (**DBMS_APP_CONT.GET_LTXID_OUTCOME**) returns the reliable commit outcome. Further detail on using the Transaction Guard API is provided later in the lesson.

Using Transaction Guard

- Supported transaction types:
 - Local commit
 - Auto-commit and Commit on Success
 - Commit embedded in PL/SQL
 - DDL, DCL, and Parallel DDL
 - Remote, Distributed commit
- Not supported in release 12.1:
 - XA
 - Read/write database links from Active Data Guard
- Server configuration:
 - Set the COMMIT_OUTCOME=TRUE service attribute.
 - Optionally, set the RETENTION_TIMEOUT service attribute.
- Supported clients:
 - JDBC Thin, OCI, OCCI, and ODP.NET



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Transaction Guard supports all the transaction types listed in the slide. The primary exclusions in Oracle Database 12c release 12.1 are:

- Transaction Guard is not supported for applications developed by using Oracle XA.
- Transaction Guard is not supported if you are using Active Data Guard with read/write database links to another database.

To enable Transaction Guard, set the service attribute COMMIT_OUTCOME=TRUE. Optionally, change the RETENTION_TIMEOUT service attribute to specify the amount of time that the commit outcome is retained. The retention timeout value is specified in seconds; the default is 86400 (24 hours), and the maximum is 2592000 (30 days).

Example: Creating Services for Transaction Guard

- To create a service using Transaction Guard but not Application Continuity:

```
$ srvctl add service -db racdb -service app3  
  -serverpool Srvpool1  
    -commit_outcome TRUE          Mandatory Settings for Transaction Guard  
    -retention 86400 -failoverretry 30 -failoverdelay 10  
    -notification TRUE -rlbgoal SERVICE_TIME -clbgoal SHORT
```

- To modify an existing service to enable Transaction Guard:

```
$ srvctl modify service -db racdb -service app4  
  -commit_outcome TRUE -retention 86400  
  -notification TRUE
```

- To use Transaction Guard, a DBA must grant permission:

```
SQL> GRANT execute ON DBMS_APP_CONT;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To enable Transaction Guard but not Application Continuity, create the service using SRVCTL and set only `-commit_outcome` to TRUE.

You can use SRVCTL to modify an existing service to enable Transaction Guard, similar to the following command, where racdb is the name of your Oracle RAC database, and app2 is the name of the service you are modifying:

```
$ srvctl modify service -db racdb -service app2 -commit_outcome TRUE  
  -retention 86400 -notification TRUE
```

In the preceding example, the `-retention` parameter specifies how long, in seconds, to maintain the history. Additionally, the `-notification` parameter is set to TRUE, enabling FAN events. To use Transaction Guard, a DBA must grant permission, as follows:

```
GRANT EXECUTE ON DBMS_APP_CONT;
```

Benefits of Transaction Guard

- After outages, users know what happened to their in-flight transactions, such as fund transfers, flight bookings, and bill payments.
- Transaction Guard provides better performance and reliability than home-built code for idempotence.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Transaction Guard is a powerful feature. Some of the benefits are:

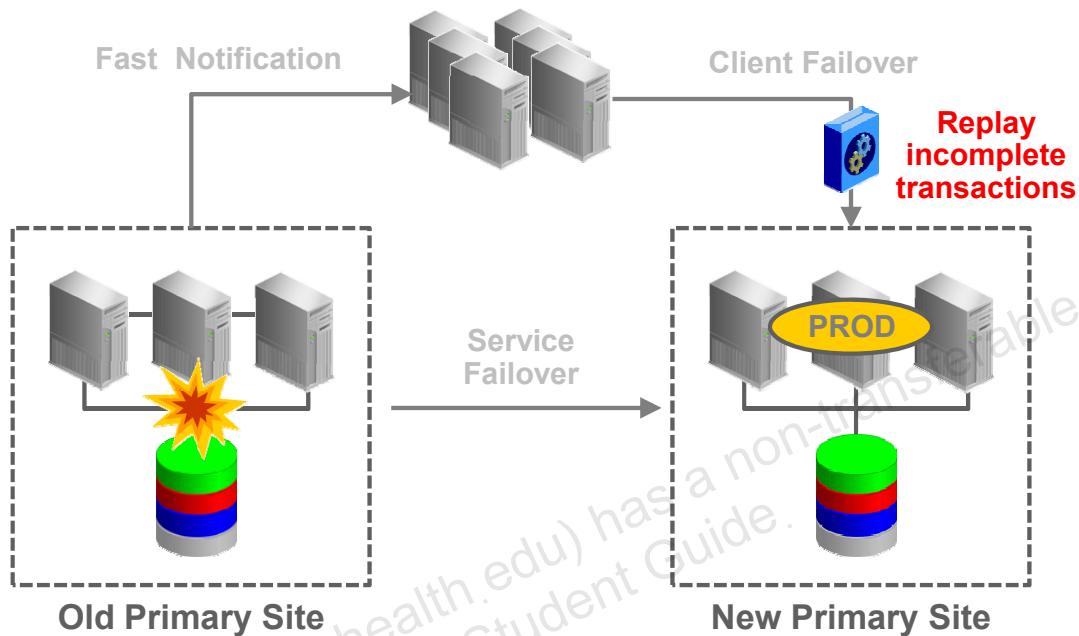
- For applications that integrate Transaction Guard, users can know what happened to their last submission and proceed with confidence and certainty. Without Transaction Guard, doubt following a failure can lead to resubmitting a database request, which can cause logical corruption.
- Because it is integrated into the database kernel, Transaction Guard provides better performance and reliability than home-built code for idempotence.

Client Failover Considerations

5. Replying Incomplete Requests



Did the last transaction commit?



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram shows the following scenarios:

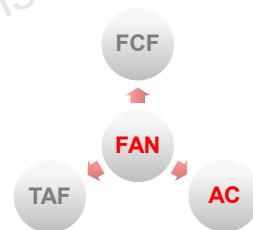
- The database role was transitioned.
- The DG_PROD service was relocated as a result of database failover.
- The Data Guard Broker and Clusterware notified the applications about the status changes of the database and service.
- Applications leveraged Transaction Guard to check the commit outcome of the last in-flight transaction.

The next effective client failover consideration is how to resubmit the incomplete requests if the last commit operation of the in-flight transactions was not successful.

Introducing Application Continuity

Oracle Database 12c introduces two fundamental capabilities for ensuring continuity of applications after database outages :

- Transaction Guard: A new reliable protocol and API that returns the outcome of the last transaction after a recoverable error has occurred
- **Application Continuity:** A feature that attempts to mask database session outages by recovering the in-flight work for requests submitted to the database



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

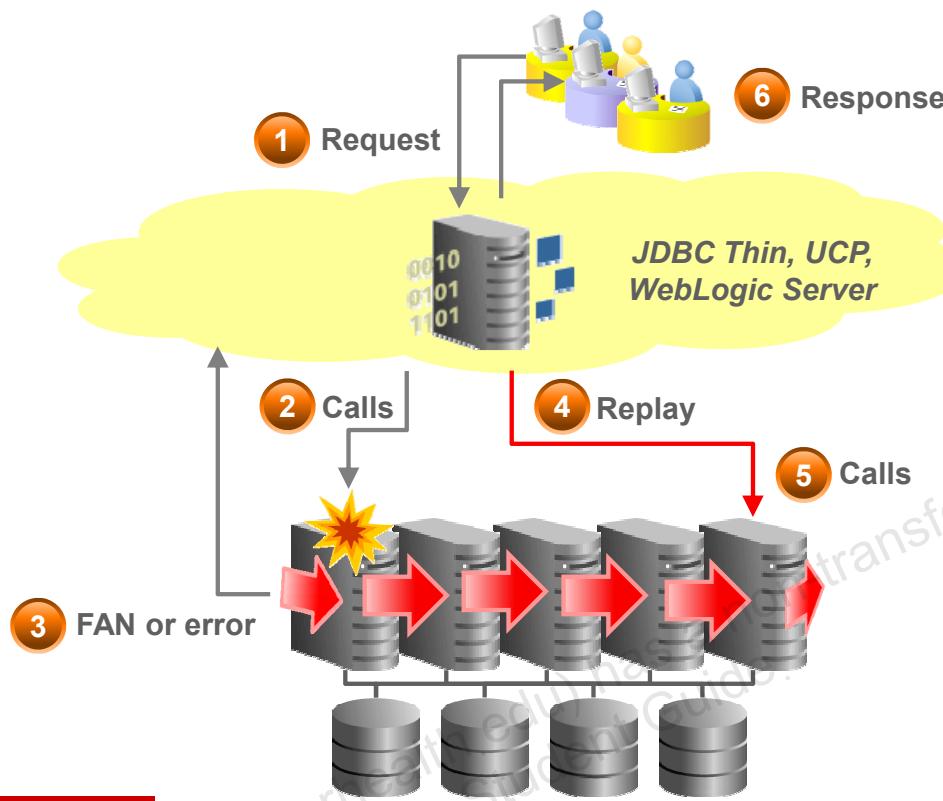
Oracle Database 12c introduces two fundamental capabilities for ensuring continuity of applications after database outages:

1. A foolproof way for applications to know the outcome of transactions
2. The ability to mask outages by reconnecting to the database and replaying the workload

These capabilities are provided by two new features. You learned about Transaction Guard in the previous slides. The second feature is Application Continuity.

- Application Continuity is a feature that masks recoverable outages from end users and applications. Application Continuity attempts to replay the transactional and nontransactional work that constitutes a database request. When replay is successful, the outage appears to the end user as if the execution was slightly delayed. With Application Continuity, the end-user experience is improved because users may never sense that an outage has occurred. Furthermore, Application Continuity can simplify application development by removing the burden of dealing with recoverable outages.

How Does Application Continuity Work?



The graphic in the slide illustrates how Application Continuity works. Following is a description of a typical workflow involving Application Continuity:

1. The client sends a work request to the application.
2. The application sends the calls that make up the request to the database using the JDBC replay driver.
3. The JDBC replay driver receives a FAN notification or a recoverable error.
4. The replay driver performs the following actions:
 - It checks that the request has replay enabled and that the replay initiation timeout has not expired. If all is in order, the driver obtains a new database session. If a callback is registered, the callback is executed to initialize the session.
 - It checks with the database to determine whether the last transaction completed.
 - If replay is required, the JDBC replay driver resubmits calls, receiving directions for each call from the database. Each call must result in the same client-visible state.
5. When the last call is replayed, the replay driver ends the replay and returns to normal runtime mode.
6. If the replay succeeds, the application responds normally to the user.

Using Application Continuity

- Supported database operations:
 - SQL, PL/SQL, and JDBC RPC: SELECT, ALTER SESSION, DML, DDL, COMMIT, ROLLBACK, SAVEPOINT, and JDBC RPCs
 - Transaction models: Local, Parallel, Remote, Distributed, and Embedded PL/SQL
 - Mutable functions
 - Transaction Guard
- Works in conjunction with:
 - Oracle RAC and RAC One
 - Oracle Active Data Guard
- Hardware acceleration on current Intel and SPARC chips
- Supported clients:
 - JDBC Thin, Universal Connection Pool, and WebLogic Server



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide lists key points relating to the use of Application Continuity. The following notes elaborate further:

- Application Continuity recovers the database request, including any in-flight transaction and the database session states. The requests may include most SQL and PL/SQL, RPCs, and local JDBC calls. Note that for remote and distributed transactions, all databases involved must be release 12.1 or later.
- Application Continuity offers the ability to keep the original values for some Oracle functions, such as `SEQUENCE.NEXTVAL`, that change their values each time that they are called. This improves the likelihood that the replay will succeed.
- Application Continuity uses Transaction Guard. Transaction Guard tags each database session with a logical transaction ID (LTXID), so that the database recognizes whether a request committed the transaction before the outage.
- Application Continuity works in conjunction with Oracle RAC, RAC One, and Oracle Active Data Guard.
- On the database server, the validation performed by Application Continuity is accelerated using processor extensions built into current SPARC and Intel chips.
- Application Continuity provides client support for JDBC Thin, Universal Connection Pool, and WebLogic Server.

Configuring the JDBC Replay Data Source

Use the `oracle.jdbc.replay.OracleDataSourceImpl` data source to obtain JDBC connections.

- Configure the data source in the property file for UCP or WebLogic Server.
- Reference the data source in JDBC Thin applications:

```
datasource=oracle.jdbc.replay.OracleDataSourceImpl
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To use Application Continuity for Java, you must use the `oracle.jdbc.replay.OracleDataSourceImpl` data source to obtain JDBC connections. This data source supports all the properties and configuration parameters of all the Oracle JDBC data sources (for example, `oracle.jdbc.pool.OracleDataSource`).

You can configure the replay data source by changing the data source property for Universal Connection Pool or by setting it using the WebLogic Console. You can also set the replay data source for JDBC Thin applications in the property file or directly in the application.

Creating Services for Application Continuity

- To create a service for Application Continuity for a policy-managed RAC database:

```
$ srvctl add service -db racdb -service app2
  -serverpool Srvpool1
  -failovertype TRANSACTION
  -commit outcome TRUE
  -replay_init_time 1800 -failoverretry 30 -failoverdelay 10
  -retention 86400
  -notification TRUE -rlbgoal SERVICE_TIME -clbgoal SHORT
```

Mandatory Settings for Application Continuity

Optional Settings for Application Continuity

- To modify an existing service for Application Continuity:

```
$ srvctl modify service -db racdb -service app1 -clbgoal SHORT
  -rlbgoal SERVICE_TIME -failoverretry 30 -failoverdelay 10
  -failovertype TRANSACTION -commit_outcome TRUE
  -replay_init_time 1800 -retention 86400 -notification TRUE
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To use Application Continuity, set the following mandatory service attributes:

- FAILOVERTYPE=TRANSACTION to enable Application Continuity
- COMMIT_OUTCOME=TRUE to enable Transaction Guard

Additionally, you can set values for the following service parameters for Application Continuity and load balancing:

- REPLAY_INIT_TIME: This setting specifies the number of seconds within which replay must start. If replay does not start within the specified time, then it is abandoned. Oracle recommends that you select a value based on how long you will allow replay to be initiated. The default value is 300 seconds.
- RETENTION: This setting specifies the number of seconds that the commit outcome is stored in the database. The default is 86400 (24 hours), and the maximum is 2592000 (30 days).

After a COMMIT has executed, if the session state was changed in that transaction, then it is not possible to replay the transaction to reestablish that state if the session is lost. When configuring Application Continuity, the applications are categorized depending on whether the session state after the initial setup is dynamic or static, and then whether it is correct to continue past a COMMIT operation within a request.

Benefits of Application Continuity

- Uninterrupted user service, when replay is successful
- Can help relocate database sessions to remaining servers for planned outages
- Improves developer productivity by masking outages that can be masked
- Few or no application changes
- Simple configuration



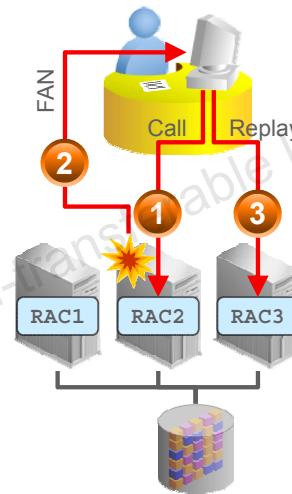
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Here are some benefits of Application Continuity:

- User service is uninterrupted when the request replay is successful.
- Application Continuity can be used to migrate the database sessions to the remaining servers without the users perceiving an outage.
- Masking outages that can be masked improves developer productivity. Error handling code will be invoked less often and can potentially be simplified.
- Replay often requires few or no application changes.
- Application Continuity is simple to configure.

RAC and Application Continuity

- Application Continuity transparently replays database requests after a failed session.
 - Users are shielded from many types of problems.
- Using Application Continuity with RAC provides:
 - Protection against a wider variety of failure scenarios
 - Faster reconnect and replay
 - Request replay on another RAC instance



ORACLE®

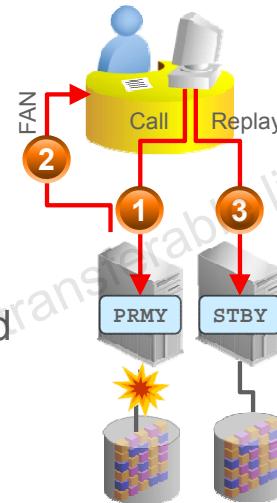
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Application Continuity is a feature that rapidly and transparently replays a request against the database after a recoverable error that makes the database session unavailable. The request can contain transactional and nontransactional work. With Application Continuity, the end-user experience is improved by masking many system, communication, hardware, and storage problems from the end user.

When Application Continuity is used in conjunction with Oracle RAC, failed sessions can be quickly restarted and replayed on another RAC database instance. Using Application Continuity in conjunction with Oracle RAC provides protection against a wider variety of possible failures compared with using Application Continuity against a single-instance database. Also, using Application Continuity in conjunction with Oracle RAC enables quicker replay compared with using Application Continuity in conjunction with Data Guard because reconnecting to another already running database instance can be completed in a few seconds while a Data Guard failover operation may take a few minutes.

Data Guard and Application Continuity

- Using Application Continuity with Data Guard provides:
 - Protection against a wider variety of failure scenarios
 - Faster reconnect and replay
 - Request replay on a new primary database
- Application Continuity works with the following events:
 - Switchover
 - Manual Failover
 - Fast-Start Failover with Maximum Availability
- Use of Application Continuity for a Data Guard failover requires that both source and target databases be at Oracle 12.1.0.2 or later.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Application Continuity is supported for switchover or failover to physical standby databases. Application Continuity can also mask unplanned outages of a Data Guard primary database configured in Maximum Availability (zero data loss failover) with Data Guard Fast-Start Failover (automatic database failover). Use of Application Continuity for a Data Guard failover requires that both source and target databases be at Oracle 12.1.0.2 or later.

Quiz



Application Continuity attempts to mask recoverable database outages from applications and users by restoring database sessions and replaying database calls.

- a. True
- b. False



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe effective client failover
- Describe client failover considerations
- Describe the purpose and architecture of Application Continuity
- Describe how applications can leverage Application Continuity
- Configure an Oracle RAC database to enable the use of Application Continuity to provide transparent failover in the case of node or instance failure
- Configure Oracle Data Guard to enable the use of Application Continuity to provide effective client failover in case of site failure



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 7 Overview: Using Application Continuity

This practice covers the following topics:

- Preparing the Data Guard Environment for Application Continuity
- Using Application Continuity in RAC
- Using Application Continuity in Data Guard



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Effective Service Failover and Workload Management Using Global Data Services

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the effective client failover in a Data Guard environment with RAC
- Describe the workload management in RAC
- Explain the benefits provided by Global Data Services
- List the components of the Global Services Framework
- Explain how Global Service connections are load balanced
- Describe the process of Global Services failover
- Configure client connectivity in a Global Data Services configuration
- Describe Global Data Services Deployment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Concept Review
 - Effective Client Failover in a Data Guard Environment with RAC
 - Workload Management in RAC
- Global Data Services Overview
- Global Services Overview
 - Global Services in RAC
 - Global Services in Data Guard
- Global Services Attributes
 - Global Service Placement
 - Global Connection Load Balancing
- Client Connectivity in GDS
- GDS Deployment Overview

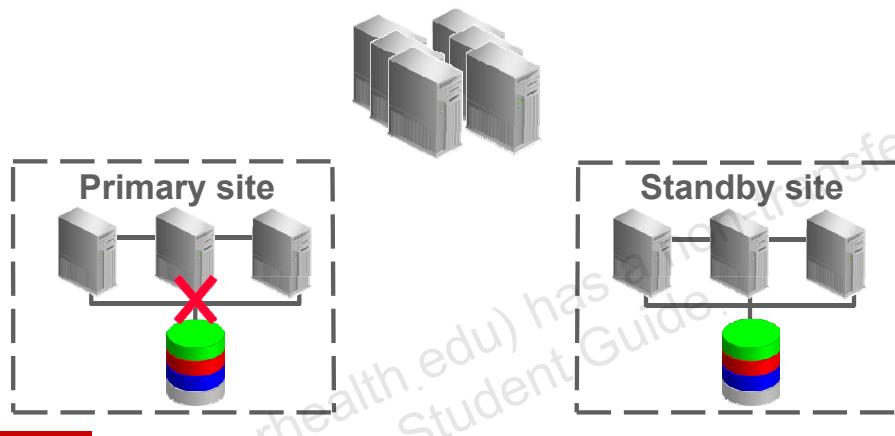


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Review of Effective Client Failover

- Connecting to the appropriate environment
- Fast Notification of Clients
- Automatic Client Failover
- Handling In-Flight Transactions
- Replaying Incomplete Requests



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You learned about automatic fast failover of application clients in a Data Guard configuration, which requires:

- Fast database failover
- Restarting database services on the new primary database
- Notifying clients that a failure has occurred to break them out of TCP timeout and redirect them to the new primary database
- Handling In-Flight Transactions
- Replaying Incomplete Requests

In this lesson, you will learn about Global Data Services, which can provide effective service failover in replicated environments, which include the Oracle Data Guard environments.

Review of Workload Management in RAC

- Connection load balancing is rendered possible by configuring multiple listeners on multiple nodes:
 - Client-side connect-time load balancing
 - Client-side connect-time failover
 - Server-side connect-time load balancing
- Runtime connection load balancing is rendered possible by using connection pools:
 - Work requests automatically balanced across the pool of connections
 - Native feature of Oracle Universal Connection Pool (UCP) for Java and ODP.NET connection pool



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

With RAC, multiple listeners on multiple nodes can be configured to handle client connection requests for the same database service.

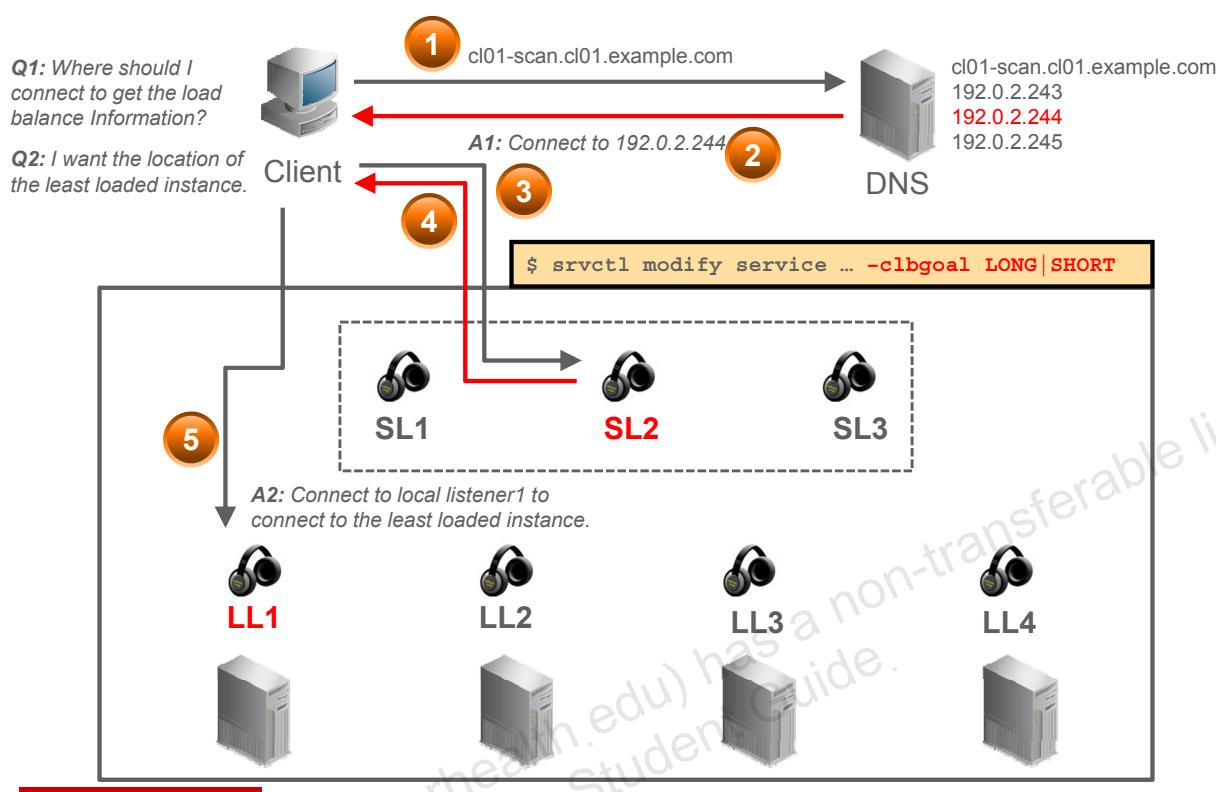
A multiple-listener configuration enables you to leverage the following failover and load-balancing features:

- Client-side connect-time load balancing
- Client-side connect-time failover
- Server-side connect-time load balancing

These features can be implemented either one by one, or in combination with each other.

In this lesson, you will learn about the effective workload management feature in the replicated environments, which is very similar to the workload management in the Oracle RAC environments.

Connection Time Load Balancing in RAC: Example

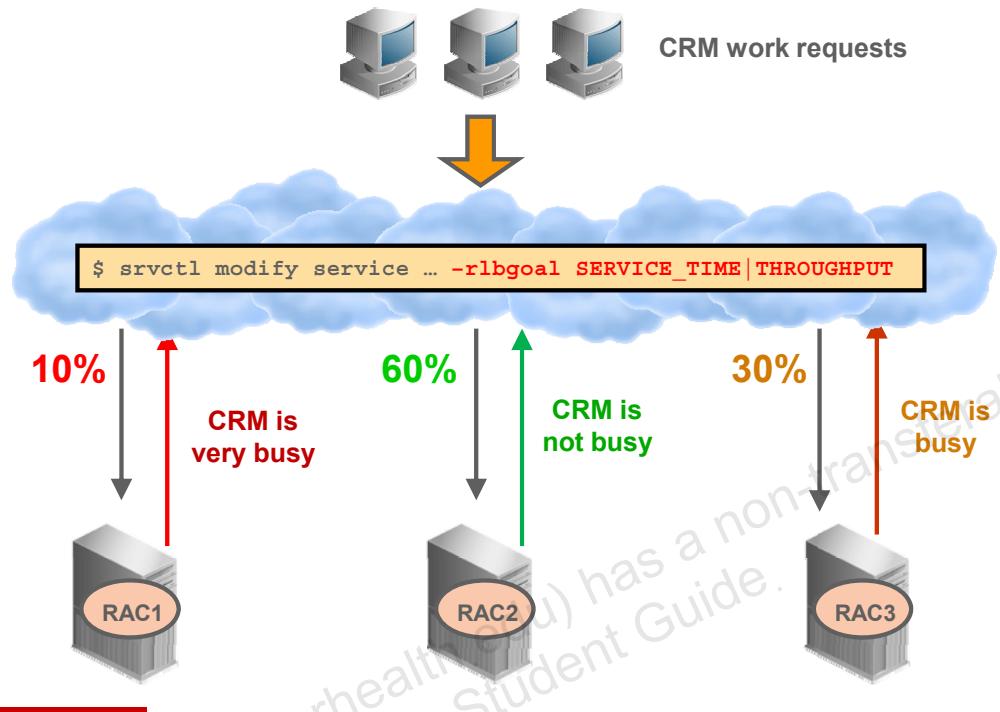


The diagram in the slide illustrates an example of connection load balancing in a RAC environment:

- The connection request with SCAN (`cl01-scan.cl01.example.com`) is forwarded.
- The DNS resolves SCAN returning the IP address (`192.0.2.244`) matching the name given; this address is then used by the client to contact the SCAN listener.
- The SCAN listener (`SL2`) uses its connection load balancing system to pick an appropriate local listener, whose name it returns to the client in an Oracle Net Redirect message.
- The client reconnects to the selected local listener (`LL1`).

The SCAN listeners must be known to all the database listener nodes and clients. The database instance nodes cross-register only with known SCAN listeners, also sending them per-service connection metrics.

Runtime Connection Load Balancing (UCP JDBC/ODP.NET) in RAC: Example



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates an example of runtime load balancing in a RAC environment.

The Runtime Connection Load Balancing feature provides an assignment of connections based on the Load Balancing Advisory information from the instances in the RAC cluster. The Connection Cache assigns connections to clients on the basis of a relative number indicating what percentage of work requests each instance should handle.

In the diagram in the slide, the feedback indicates that the CRM service on INST1 is so busy that it should service only 10% of the CRM work requests; INST2 is so lightly loaded that it should service 60%; and INST3 is somewhere in the middle, servicing 30% of requests. Note that these percentages apply to, and the decision is made on, a per-service basis. In this example, CRM is the service in question.

Note: Runtime Connection Load Balancing is a feature of Oracle connection pools.

Lesson Agenda

- Concept Review
 - Effective Client Failover in a Data Guard Environment with RAC
 - Workload Management in RAC
- Global Data Services Overview
- Global Services Overview
 - Global Services in RAC
 - Global Services in Data Guard
- Global Services Attributes
 - Global Service Placement
 - Global Connection Load Balancing
- Client Connectivity in GDS
- GDS Deployment Overview



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Global Data Services

- Many companies maintain replicas of their databases locally and in geographically disparate data centers.
- Reasons for having both local and global replicas include:
 - Business continuity and disaster recovery
 - Performance optimization for local clients
 - Content localization and caching
 - Compliance with local laws
 - Integration of data centers spurred by mergers/acquisitions
- Some enterprises using Oracle databases implement their own distributed database workload management solutions.
- These generic solutions cannot provide critical functionality, such as runtime load balancing and reliable failover.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

There are multiple business reasons for having both local and global replicas, including:

- Business continuity and disaster recovery
- Performance optimization for local clients
- Content localization and caching
- Compliance with local laws
- Integration of data centers obtained through mergers and acquisitions

In a system containing multiple replicated databases, a particular database server may cause a slower response time because of an increased demand for a database service, whereas replica servers capable of offering the same service may be underutilized.

Many large enterprises using Oracle databases implement their own solutions for workload management in distributed database systems. These solutions, however, cannot provide critical functionality, such as runtime load balancing and reliable failover, because they are not integrated with the Oracle software stack.

Global Data Services Capabilities

- Global Data Services (GDS) for database clouds applies the RAC service model to sets of globally distributed databases.
- The capabilities of GDS include:
 - Region-based Workload Routing
 - Lag-based Workload Routing for Active Data Guard
 - Connection Time Load Balancing
 - Run-Time Load Balancing for Oracle integrated clients
 - Global Service Failover
 - Role-based Global Services for Active Data Guard
 - Centralized Service Management Framework



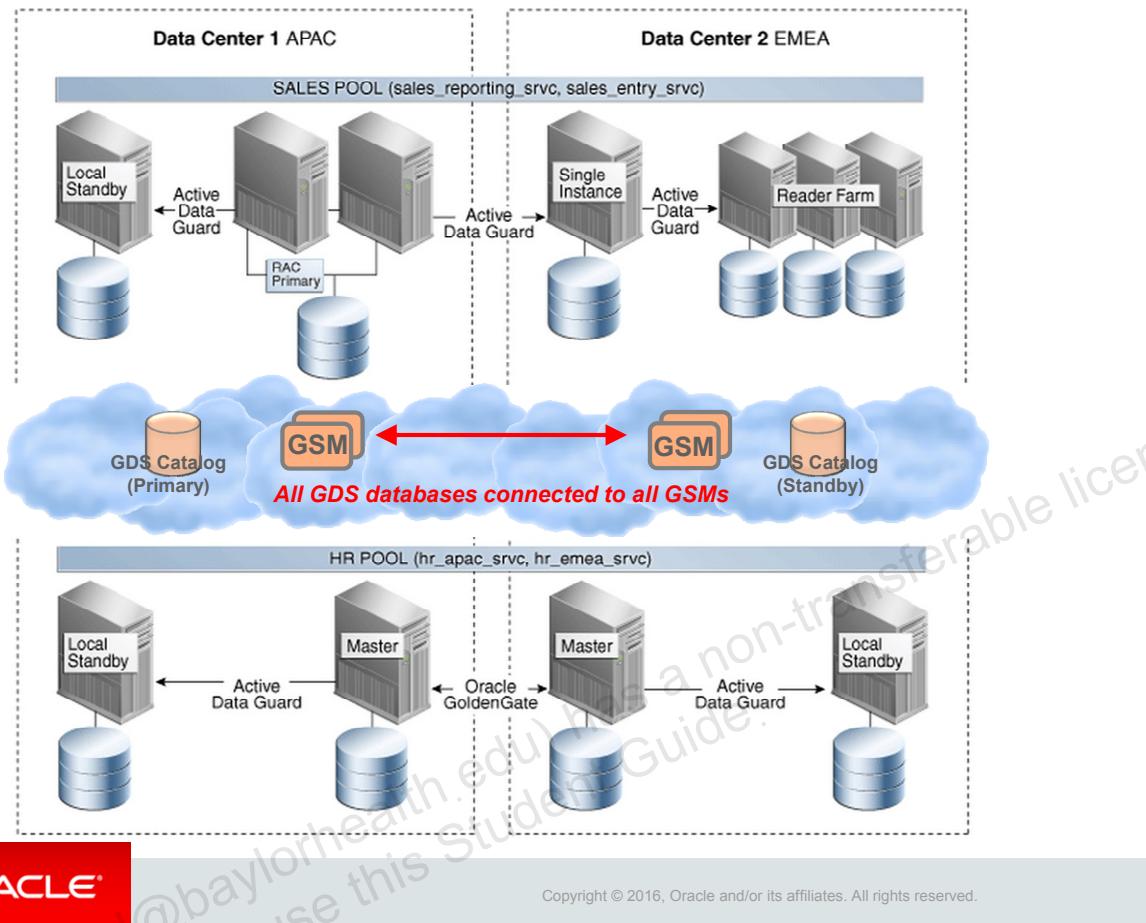
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Global Data Services for database clouds applies the Oracle RAC service model to sets of globally distributed, heterogeneous databases, providing dynamic load balancing, failover, and centralized service management for a set of replicated databases that offer common services. The set of databases can include Oracle RAC and non-cluster Oracle databases interconnected through Oracle Data Guard, Oracle GoldenGate, or any other replication technology.

Features of Global Data Services enable you to integrate your locally and globally distributed, loosely coupled databases running on heterogeneous platforms into a scalable and highly available private database cloud that can be shared by clients around the globe.

Distributed database systems, in most cases, do not maintain absolute data consistency across replicas. Therefore, not all services that can currently run on multiple instances of an Oracle RAC database can be scaled to run in a multi-database environment. Global Data Services is primarily intended for applications that are replication aware, use read-only services, or both. Applications that cannot be modified to work with replicated data can still benefit from improved high availability and disaster-recovery capabilities of a distributed database system.

Global Data Services Architecture



The framework of Global Data Services (GDS) includes logical and physical components.

The logical components include:

- **GDS Configuration:** A set of databases that provide global services
- **GDS Pool:** Databases that offer a common set of global services
- **GDS Region:** Group of databases and clients in close network proximity
- **GDS Service:** Database Service provided by multiple databases with replicated data

The physical components include:

- **Global Service Manager (GSM):** It is a regional listener to the incoming database connections; performs connection-time load balancing and publishes runtime load balancing advisory and FAN events via ONS.
- **GDS Catalog:** It stores GDS configuration metadata.
- **GDSCTL utility:** It is the command-line interface to administer GDS configuration. The graphical interface of GDS is available in EMCC DB Plug-in 12.1.0.5.
- **Databases:** A global service is provided by a set of databases residing in the same Global Data Services pool.
- **ONS:** An Oracle Notification Server (ONS) runs with each GSM delivering FAN events and runtime load balancing metrics to clients.

Lesson Agenda

- Concept Review
 - Effective Client Failover in a Data Guard Environment with RAC
 - Workload Management in RAC
- Global Data Services Overview
- Global Services Overview
 - Global Services in RAC
 - Global Services in Data Guard
- Global Services Attributes
 - Global Service Placement
 - Global Connection Load Balancing
- Client Connectivity in GDS
- GDS Deployment Overview



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Global Service: Overview

- For database clients, a Global Data Services configuration is represented by a set of global services.
- A GSM serving a Global Data Services configuration is aware of all global services provided by the configuration.
 - It acts as a mediator between database clients and databases in the GDS configuration.
- A client program connects to a regional global service manager and requests a connection to a global service.
- The GSM forwards the client's request to the optimal instance in the GDS configuration that offers the global service.
- The configuration and runtime status of global services are stored in the Global Data Services catalog.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

For database clients, a Global Data Services configuration is represented by a set of global services. A global service manager serving a Global Data Services configuration is aware of all global services that the GDS configuration provides and acts as a mediator between database clients and databases in the GDS configuration. A client program connects to a regional global service manager and requests a connection to a global service. The client does not need to specify which database or instance it requires. The global service manager forwards the client's request to the optimal instance in the GDS configuration that offers the global service. Database clients that share a global service must have the same service-level requirements.

The functionality of local services is not changed by global services. Oracle Database 12c can provide local and global services simultaneously. A client application can also work with global and local services simultaneously.

The configuration and runtime status of global services are stored in the Global Data Services catalog. Each database that offers global services also maintains information about those services in a local repository (such as a system dictionary or Oracle Cluster Registry), along with data on local services. Global services that are stored in a local repository have a special flag to distinguish them from traditional local services.

Note: Databases before Oracle Database 12c can provide local services, but only Oracle Database 12c, and later, can provide global services.

Global Services in a RAC Database

- Some properties of a global service are applicable only to RAC databases and are unique for each RAC database.
- These properties are related to placement of global services with instances within a RAC database, including:
 - Server pools and service cardinality
 - Instance assignment (for Admin-managed databases)
 - Distributed transaction processing
- You can specify attributes for these properties by using the `srvctl` utility.
- Global Data Services supports policy-managed and administer-managed RAC databases.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Some properties of a global service are applicable only to RAC databases and are unique for each RAC database included in a GDS configuration. These properties are related to placement of global services with instances within an Oracle RAC database, including:

- Server pools and service cardinality for policy-managed databases
- Distributed transaction processing

You can specify attributes for these properties using `srvctl`; however, you must manage these properties in an Oracle RAC database. This means that all current and future service placement functionality on Oracle RAC databases will be supported for global services. Local management of database-specific service properties also provides better performance and availability. All other existing global service attributes, such as load balancing, role, transparent application failover parameters, and database edition, must be the same for all databases offering a global service. You must specify these attributes at the global service level.

By default, in an Oracle RAC environment, a SQL statement executed in parallel can run across all of the nodes in the cluster. The cross-node parallel execution is not intended to be used with GDS load balancing. For an Oracle RAC database in a GDS configuration, it is recommended that you restrict the scope of the parallel execution to an Oracle RAC node by setting the `PARALLEL_FORCE_LOCAL` initialization parameter to `TRUE`.

Global Services in a Data Guard Broker Configuration

- When you include a broker configuration in a GDS configuration, broker configurations are managed as a single unit.
 - Only an entire broker configuration can be added to or deleted from a Global Data Services pool.
 - A broker configuration cannot span multiple pools.
- A database is added to the GDS pool by adding it to the broker configuration by using the Data Guard `dgmgrl` utility.
- After adding a database to the broker configuration, run the following command to synchronize GDS and Data Guard:

```
$ gdsctl sync brokerconfig
```
- Global services can be configured with a role attribute to be active in a specific role, such as primary or physical standby.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Data Guard enables one primary database to be connected to up to 30 standby databases. The Oracle Data Guard broker logically groups these primary and standby databases into a broker configuration that enables the broker to manage and monitor the databases together as an integrated unit. When you include a broker configuration in a Global Data Services configuration, you manage the broker configuration as a single unit. Only an entire broker configuration can be added to or deleted from a Global Data Services pool, and a broker configuration cannot span multiple pools.

If you attempt to add or remove a database that belongs to a broker configuration to or from a Global Data Services pool, an error occurs. You can add a database to the Global Data Services pool only by adding it to the broker configuration using the Data Guard `dgmgrl` utility. When you add a database to the broker configuration, you must run the `gdsctl sync brokerconfig` command to synchronize Global Data Services and Data Guard.

Conversely, when you remove a database from a broker configuration, it is removed from the Global Data Services pool to which this broker configuration belongs. This is the only way to remove a database from a pool that contains a broker configuration.

You can configure global services with a role attribute to be active in a specific database role, such as primary or physical standby database. If you enable fast-start failover, the Oracle Data Guard broker automatically fails over to a standby database if the primary database fails.

The global service managers configured to work with the Oracle Data Guard broker ensure that the appropriate database services are active and that the appropriate Fast Application Notification (FAN) events are published after a role change. The Global Data Services framework supports the following Oracle Data Guard broker configurations:

- The set of databases in a Global Data Services pool can be either the set of databases that belong to a single broker configuration or a set of databases that do not belong to a broker configuration. You can add a broker configuration only to an empty Global Data Services pool and, if a pool already contains a broker configuration, then to add a database to the pool, you must add the database to the broker configuration contained in the pool.
- Role-based global services are supported only for database pools that contain a broker configuration.

Lesson Agenda

- Concept Review
 - Effective Client Failover in a Data Guard Environment with RAC
 - Workload Management in RAC
- Global Data Services Overview
- Global Services Overview
 - Global Services in RAC
 - Global Services in Data Guard
- Global Services Attributes
 - Global Service Placement
 - Global Connection Load Balancing
- Client Connectivity in GDS
- GDS Deployment Overview



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Global Service Attributes

- Global services attributes control:
 - Global service startup
 - Load-balancing connections to the global services
 - Failing over those connections
- Local service attributes, including those specific to RAC and Data Guard, are also applicable to global services.
- Attributes unique to global services include:
 - Preferred or available databases
 - Replication lag
 - Region affinity
- You can enable and disable, move, and change the properties of a global service just like a local service.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Global services have a set of attributes associated with them that control starting the global services, load-balancing connections to the global services, failing over those connections, and more. Attributes applicable to local services, including those specific to Oracle RAC and Oracle Data Guard broker environments, are also applicable to global services.

The following attributes are unique to global services:

- Preferred or available databases
- Replication lag
- Region affinity

You can modify global services as you can modify local services. You can enable and disable a global service, move the global service to a different database, and change the properties of the global service.

Note: You cannot upgrade a local service to a global service.

preferred, available, and preferred_all

- You can specify which databases will support a service.
 - These databases are referred to as *preferred databases*.
- Options for Service Placement:
 - **preferred**: Databases designated to provide Global Service
 - **available**: Databases that provide Global Service if not enough *preferred* databases are running. If one of the preferred databases fails, then GSM maintains the cardinality of the Global service by starting the service on an *available* database.
 - **preferred_all**: All databases in a GDS Pool preferred for Global Service
- Example

```
GDSCTL> add service -service sales_reporting_srvc  
-gdspool sales -preferred eastdb -available westdb
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can specify which databases will support a service. These databases are referred to as *preferred databases*. The GSM ensures that a global service runs on all preferred databases for which it has been specified. The number of preferred databases is referred to as the database cardinality of a global service. You must specify at least one preferred database for a global service.

When you add or modify a global service, you can specify a list of available databases for this global service. If one of the preferred databases fails to provide a global service, the global service manager relocates the service to an available database to maintain the specified database cardinality for that service.

In a Global Data Services pool that contains an Oracle Data Guard broker configuration, a role-based global service can be started on a database only if the database is listed as preferred or available for the service and the role attribute of the database corresponds to the role attribute specified for the service. For example, a global service that can run on any database in a broker configuration (as long as the role attribute of the database is set to primary), must have primary specified for its role attribute and have all other databases in the broker configuration with role attributes set to preferred.

Note: If you set `preferred_all` for which databases will support a service, you do not have to explicitly specify `preferred` or `available` databases. The `preferred_all` setting implies that all databases in the pool are preferred.

Role-Based Services

- In a GDS pool with a Data Guard broker configuration, the GDS framework supports role-based global services.
- Valid roles are:
 - PRIMARY
 - PHYSICAL_STANDBY
 - LOGICAL_STANDBY
 - SNAPSHOT_STANDBY
- A global service is started *only* when the database role matches the role specified for the service.
- Example

```
GDSCTL> add service -service sales_reporting_srvc  
-gdspool sales -preferred_all -role PRIMARY |  
PHYSICAL_STANDBY | LOCAL_STANDBY | SNAPSHOT_STANDBY
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In a Global Data Services pool that contains an Oracle Data Guard broker configuration, the Global Data Services framework supports role-based global services. Valid roles are PRIMARY, PHYSICAL_STANDBY, LOGICAL_STANDBY, and SNAPSHOT_STANDBY. The Global Data Services framework automatically starts a global service only when the database role matches the role specified for the service.

If a database switches roles or fails, the Oracle Data Guard broker notifies the Global Data Services framework about the role change, and the global service manager ensures that services start according to the new database roles. A global service cannot fail over from a database in one Global Data Services region to a database in another region if the locality parameter is set to LOCAL_ONLY, and inter-region failover is not enabled.

When a global service fails over, fast connection failover, if enabled on Oracle clients, provides rapid failover of the connections to that global service. The Global Data Services framework, similar to Oracle RAC, uses Fast Application Notification (FAN) to notify applications about service outages. Instead of waiting for the application to poll the database and detect a problem, clients receive FAN events and react immediately. Sessions to the failed instance or node will be terminated, and new connections will be directed to available instances providing the global service.

Replication Lag

- For performance reasons, distributed environments often use asynchronous replication of data between databases.
- This increases the probability of a delay between the time an update is made on a primary database and when it appears on a replica database.
 - This is known as **replication lag**.
- GDS enables applications to differentiate global services providing real-time data from those returning out-of-date data.
- Example:

```
GDSCTL> add service -service sales_reader_lag15_srvc  
-gdspool sales -preferred_all -role PHYSICAL_STANDBY  
-lag 15 -failover_primary
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

For performance reasons, distributed database systems often use asynchronous replication of data between databases, which means that there is the possibility of a delay between the time an update is made to data on a primary database and the time this update appears on a replica database. When this happens, the replica database lags behind its primary database.

Global Data Services enables applications to differentiate between global services that provide real-time data from services that can return out-of-date data because of replication lag. For applications that can tolerate a certain degree of lag, you can configure a maximum acceptable lag value. For applications that cannot tolerate any replication lag, you can set the lag time for global services to zero. Requests for this global service are forwarded only to a primary database, or to a replica database that is synchronized with the primary database.

For many applications, it is acceptable to read out-of-date data as long as it is consistent. Such applications can use global services running on any database replica irrespective of the length of the replication lag time. If you configure the lag time to a value other than zero, then a client request can be forwarded only to a replica database that is not lagging behind the primary database by longer than the configured lag time for the service. Specification of the maximum replication lag is supported only for Active Data Guard configurations.

Lesson Agenda

- Concept Review
 - Effective Client Failover in a Data Guard Environment with RAC
 - Workload Management in RAC
- Global Data Services Overview
- Global Services Overview
 - Global Services in RAC
 - Global Services in Data Guard
- Global Services Attributes
 - Global Service Placement
 - Global Connection Load Balancing
- Client Connectivity in GDS
- GDS Deployment Overview



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Global Connection Load Balancing: Overview

- A client connecting to a RAC database using a service can take advantage of Oracle Net connection load balancing.
- Clients connecting to a global service are load balanced as necessary, across different databases and regions.
- The global connection load-balancing functionality includes:
 - Client-side load balancing
 - Server-side load balancing
 - Region affinity for global services



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When a client connects to an Oracle RAC database using a service, the client can take advantage of the Oracle Net connection load-balancing feature to spread user connections across all the instances that support that service. Similarly, in a Global Data Services configuration, clients connecting to a global service are load balanced, as necessary, across different databases and regions.

Client-Side Load Balancing

- Client-side load balancing and failover in a Global Data Services configuration is similar to that for a RAC database.
- In a GDS configuration, a client in a GDS region first tries to connect to any of the GSMS in its local region.
- If a GSM from the local region does not respond, the client tries a GSM in another region.
- To enable client-side load balancing and failover across multiple regions, clients must:
 - Use a connect descriptor containing a list of addresses of local and buddy GSMS for load balancing and intra-region failover



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Client-side load balancing balances connection requests across listeners and includes connection failover. With connection failover, if an error is returned from the chosen listener address, Oracle Net Services tries the next address in the address list until either a successful connection is made or it has exhausted all the addresses in the list.

Client-side load balancing and failover in a Global Data Services configuration is similar to that for an Oracle RAC database. In a Global Data Services configuration, however, a client in a Global Data Services region first tries to connect to any of the global service managers in its local region. If a global service manager from the local region does not respond, the client tries a global service manager in another region.

To enable client-side load balancing and failover across multiple regions in a GDS configuration, clients must use an Oracle Net connect descriptor that contains a list of addresses of local and buddy GSMS for load balancing and intra-region failover. If a region is not specified, it defaults to the region name of the global service manager to which the client is connected. You can also configure timeout and retry attempts for each list to enable multiple connection attempts to the current global service manager before moving to another global service manager in the list.

Global Connection Time Load Balancing

- GDS supports CLB for all clients.
- The listener directs connection requests to the best database instance in Global Data Services Pool.
- Consider Load Statistics from all GDS pool databases.
- Consider inter-region network latency, locality, and CLB goal.
- Example:

```
GDSCTL> add service -service sales_reporting_srvc  
-gdspool sales -preferred_all -clbgoal LONG | SHORT
```

- **LONG**: For applications having long-lived connections. This is typical for connection pools and SQL*Forms sessions.
- **SHORT**: For applications that have short-lived connections (Default)



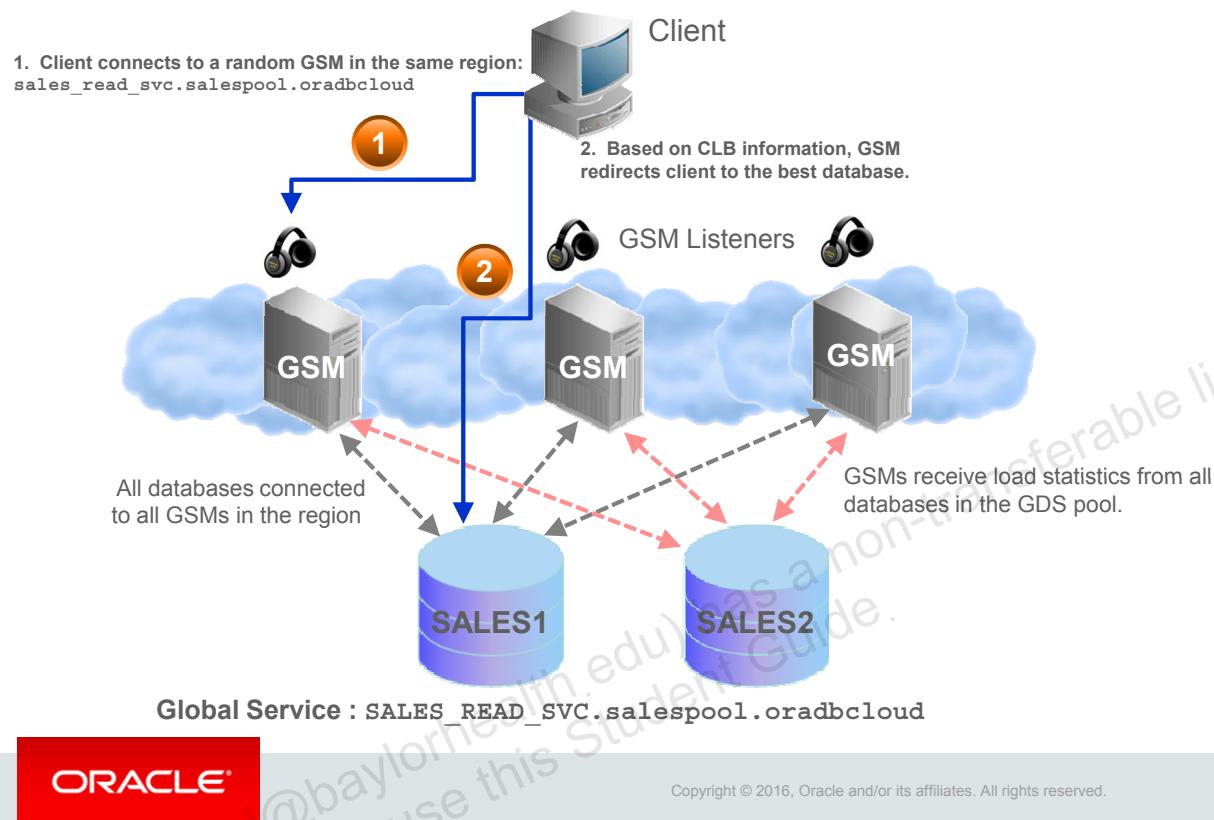
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Server-side connection load balancing for an Oracle RAC database has the listener directing connection requests to the best Oracle RAC database instance. Some applications have long-lived connections, whereas other applications have short-lived connections.

For global services, server-side connection load balancing works similarly, except that, instead of being limited to a single database, workloads are balanced across multiple databases in the Global Data Services configuration. In most cases, a global service manager directs a client request for a global service to a database server in the same region, unless all local servers are overloaded and a remote server can provide significantly better quality of service.

In some cases, to take advantage of data caching on a local server, you might want to direct requests to the local region. Global Data Services enables you to specify a desired level of client/server affinity for a global service.

Global Connection Time Load Balancing: Example



The diagram in the slide illustrates how the global connection time load balancing works with Global Data Services.

1. Client connects to a random GSM in the same region.
2. Based on the connection time load balancing information, GSM redirects client to the best databases.

Global Run Time Load Balancing

- GDS supports the RLB feature of connection pool for OCI, JDBC/UCP, ODP.NET, and WLS.
- Publish RLB Advisory to Client.
- Based on advisory, clients distribute workload requests across persistent connections spanning Global Data Services Pool database instances.
- Consider per-service performance data from pool databases.
- Consider inter-region network latency, locality, and RLB goal.
- Example:

```
GDSCTL> add service -service sales_reporting_srvc  
-gdspool sales -preferred_all  
-rlbgoal SERVICE_TIME | THROUGHTPUT
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

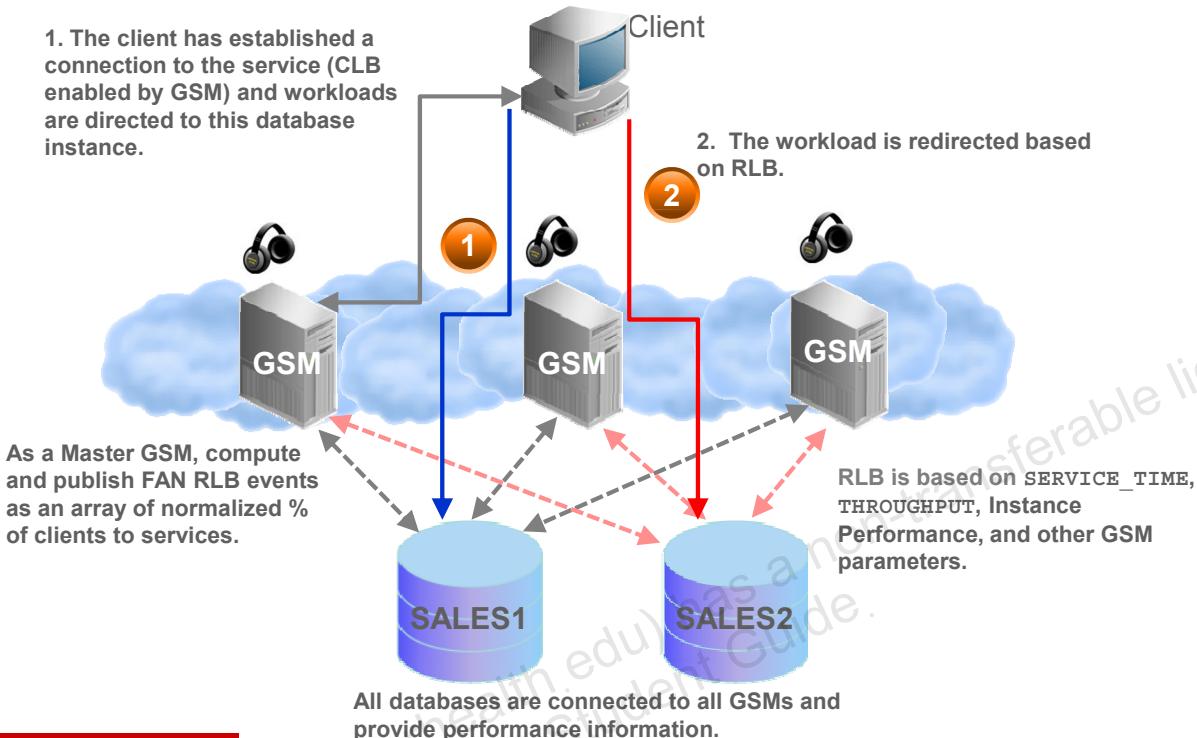
Runtime connection load balancing distributes client work requests across persistent connections that span the instances of an Oracle RAC database, based on load-balancing information from the database. The database uses the runtime connection load-balancing goal for a service and the relative performance of database instances to generate a recommendation about where to direct service requests. There are two types of service-level goals for runtime connection load balancing:

- **SERVICE_TIME**: Attempts to direct work requests to instances according to response time. Load-balancing data is based on elapsed time for work done in the service plus available bandwidth to the service.
- **THROUGHTPUT**: Attempts to direct work requests according to throughput. The load-balancing data is based on the rate at which work is completed in the service plus available bandwidth to the service.

The Global Data Services framework also supports balancing of work requests at run time to a global service. In the Global Data Services framework, the requests are spread across connections to instances in multiple databases. Work is routed to provide the best service times globally and routing responds gracefully to changing system conditions.

To provide global runtime connection load balancing, a global service manager receives performance data for each service from all database instances in the Global Data Services configuration.

Global Runtime Connection Load Balancing: Example



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates how the global runtime load balancing works with Global Data Services.

1. The client has established a connection to the service (CLB enabled by GSM) and workloads are directed to this database instance.
2. The workload is redirected based on runtime load balancing.

Locality-Based Routing

- *Region affinity* is the ability to configure global services within specific regions or in any region in the GDS configuration.
- Options for Locality-Based Routing include:
 - Locality **ANYWHERE**
 - Locality **LOCAL_ONLY**
 - Locality **LOCAL_ONLY –region_failover**
- Example:

```
GDSCTL> add service -service sales_reporting_srvc  
-gdspool sales -preferred_all -locality ANYWHERE |  
LOCAL_ONLY | LOCAL_ONLY -region failover
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can configure global services to operate within specific regions or in any region in the Global Data Services configuration. This is called region affinity. The Global Data Services framework supports three types of region affinity:

- **Any-region affinity:** Client connections and work requests are routed to any region for load balancing or failover.
- **Affinity to a local region:** Regardless of load, GDS will not route to databases in other regions.
- **Affinity to a local region with inter-region failover:** Client connects and work requests are routed to another region when all databases in a region have failed.

Lesson Agenda

- Concept Review
 - Effective Client Failover in a Data Guard Environment with RAC
 - Workload Management in RAC
- Global Data Services Overview
- Global Services Overview
 - Global Services in RAC
 - Global Services in Data Guard
- Global Services Attributes
 - Global Service Placement
 - Global Connection Load Balancing
- Client Connectivity in GDS
- GDS Deployment Overview



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Client Connectivity in GDS

- Example:

```
sales_reporting_srvc =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP) (HOST = gsm-enode01)
     (PORT = 1571)))
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP) (HOST = gsm-wnode03)
     (PORT = 1571)))
  (CONNECT_DATA =
    (SERVICE_NAME = sales_reporting_srvc.sales.oraclecloud)
    (REGION = WEST)
  )
)
```

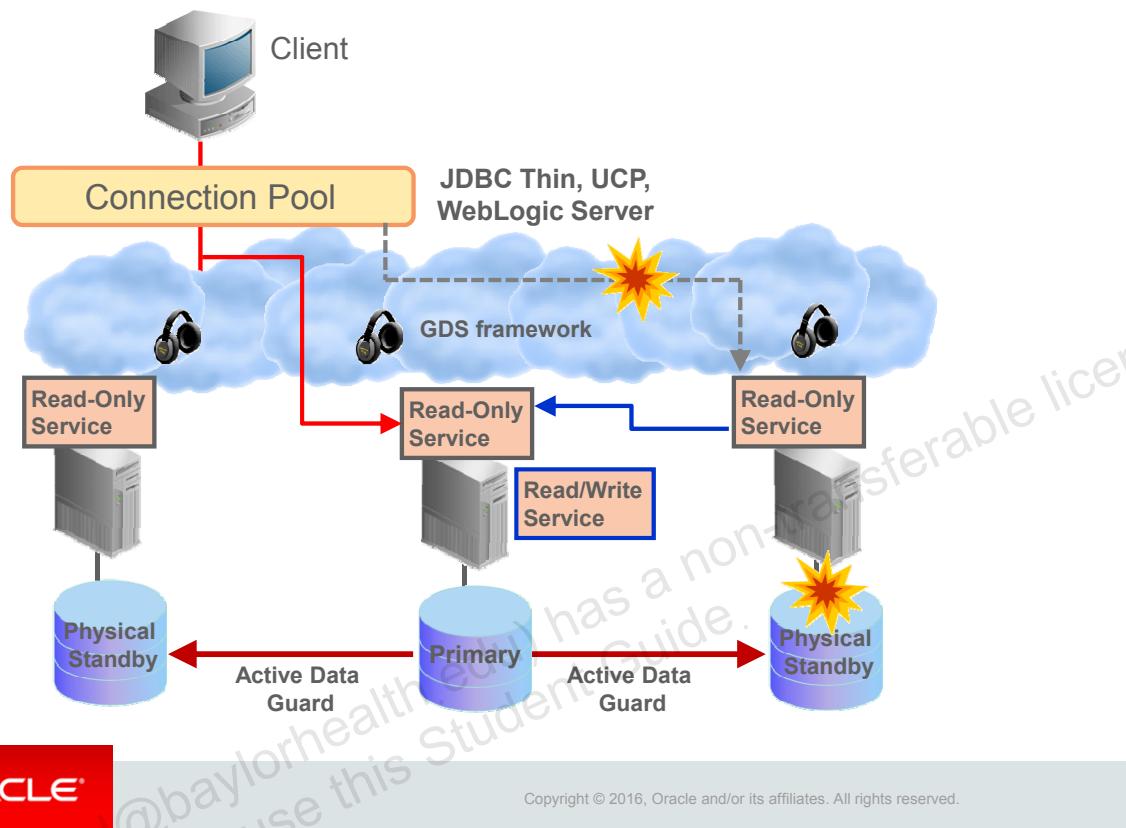
- The connect description in the example uses the GSM listener end points and not the RAC SCAN listeners or local listeners.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Clients connect to GSM listener instead of database listeners.
 - GDS forwards the connection to the local listener bypassing the SCAN listeners.
2. TNS-entries must contain two lists of addresses:
 - One list of local GSMS for load balancing and intra-region failover
 - Another list of addresses for remote GSMS for inter-region failover
3. Clients specify global service name and which region they are in.

Application Continuity and GDS with Active Data Guard



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In a GDS configuration, application continuity can be used to mask many recoverable database outages (when replay is successful) from applications and users by restoring the database session. You learned about application continuity in an earlier lesson.

Configuring Global Service for Application Continuity: Example

To create a service for application continuity:

```
GDSCTL> add service -service sales_reporting_srvc  
-gdspool sales -preferred_all
```

```
-failovertype TRANSACTION  
-commit_outcome TRUE
```

Mandatory Settings
for Application Continuity

```
-replay_init_time 1800 -failoverretry 30  
-failoverdelay 10 -retention 86400
```

Optional Settings
for Application Continuity

```
-notification TRUE -rlbgoal SERVICE_TIME  
-clbgoal SHORT
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows how to configure a global service to use application continuity.

GDS Deployment: Overview

- Install GSM software on GSM servers:
 - Minimum of one GSM per region
 - Recommended three GSMS per region
- Set up GDS Administrator accounts and privileges.
- Configure GDS:
 - Create GDS Catalog.
 - Add GSMS.
 - Add GDS Regions.
 - Add GDS Pools.
 - Add GDS Databases.
 - Add Global Services.
- Set up Client Connectivity.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide shows an overview of Global Data Services deployment.

Quiz



Which statements regarding Global Data Services are true?

- a. Global Data Services applies the RAC service model to sets of globally distributed, heterogeneous databases.
- b. Global Data Services employs a centralized, vertical framework.
- c. Global Data Services provides dynamic load balancing, failover, and centralized service management for a set of replicated databases offering common services.
- d. The set of databases can include RAC and noncluster Oracle databases interconnected through Data Guard, GoldenGate, or any other replication technology.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Answer: a, c, and d

Summary

In this lesson, you should have learned how to:

- Describe the effective client failover in a Data Guard environment with RAC
- Describe the workload management in RAC
- Explain the benefits provided by Global Data Services
- List the components of the Global Services Framework
- Explain how Global Service connections are load balanced
- Describe the process of Global Services failover
- Configure client connectivity in a Global Data Services configuration
- Describe Global Data Services Deployment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 8: Overview

This practice covers the following topics:

- Reconfiguring the Environment for GDS
- Installing and Configuring Global Data Services
- Performing Global Service Failover
- Configuring Role-Based Global Services
- Configuring Replication Lag-Based Routing



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Performing Database Recovery in an Oracle Data Guard Environment

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Explain the basic concept of Oracle Database Recovery
- Perform the appropriate type of restore and recovery operation based on the nature of your database failure in an Oracle Data Guard environment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Concept Review
 - Complete media recovery (Media Failure)
 - Incomplete media recovery (Media or Logical Failure)
 - Flashback Technology (Logical Failure)
- Recovery Considerations in Oracle Data Guard
 - Recovery from Media Failure
 - Recovery from Logical Failure



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Performing Database Recovery

Recover your database at different levels:

- Complete media recovery (Media Failure):
 - Loss of a Temp File
 - Loss of All Control files
 - Loss of a Redo Log Group
 - Loss of a Critical Data File
 - Loss of a Noncritical Data File
- Incomplete media recovery (Media or Logical Failure)
 - Loss of a Data File and Required Archived Log
 - Logical Corruptions
- Flashback database (Logical Failure)



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The following is a brief summary of the data repair techniques:

- Data file media recovery is a form of media recovery that enables you to restore data file backups and apply archived redo logs or incremental backups to recover lost changes. You can either recover a whole database or a subset of the database. Data file media recovery is the most general-purpose form of recovery and can protect against both physical and logical failures.
- Block media recovery is a form of media recovery that enables you to recover individual blocks within a data file rather than the whole data file.
- Logical flashback features enable you to view or rewind individual database objects or transactions to a past time. These features do not require the use of RMAN.
- Oracle Flashback Database is a block-level recovery mechanism that is similar to media recovery, but is generally faster and does not require a backup to be restored. You can return your whole database to a previous state without restoring old copies of your data files from backup, if you have enabled flashback logging in advance. You must have a fast recovery area configured for logging for flashback database or guaranteed restore points.

Loss of a Temp File

SQL statements that require temporary space to execute may fail if one of the temp files is missing.

```
SQL> select * from big_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13;
select * from big_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13
*
ERROR at line 1:
ORA-01565: error in identifying file
'/u01/app/oracle/oradata/orcl/temp01.dbf'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
```

Good news:

- Automatic re-creation of temporary files at startup
- Manual re-creation also possible



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If a temporary file (tempfile) that belongs to the temporary tablespace is lost or damaged, then the extents in that file will not be available. This problem may manifest itself as an error during the execution of SQL statements that require temporary space for sorting.

The SQL statement shown in the slide has a long list of columns to order by, which results in the need for temporary space. The missing file error is encountered when this statement requiring a sort is executed.

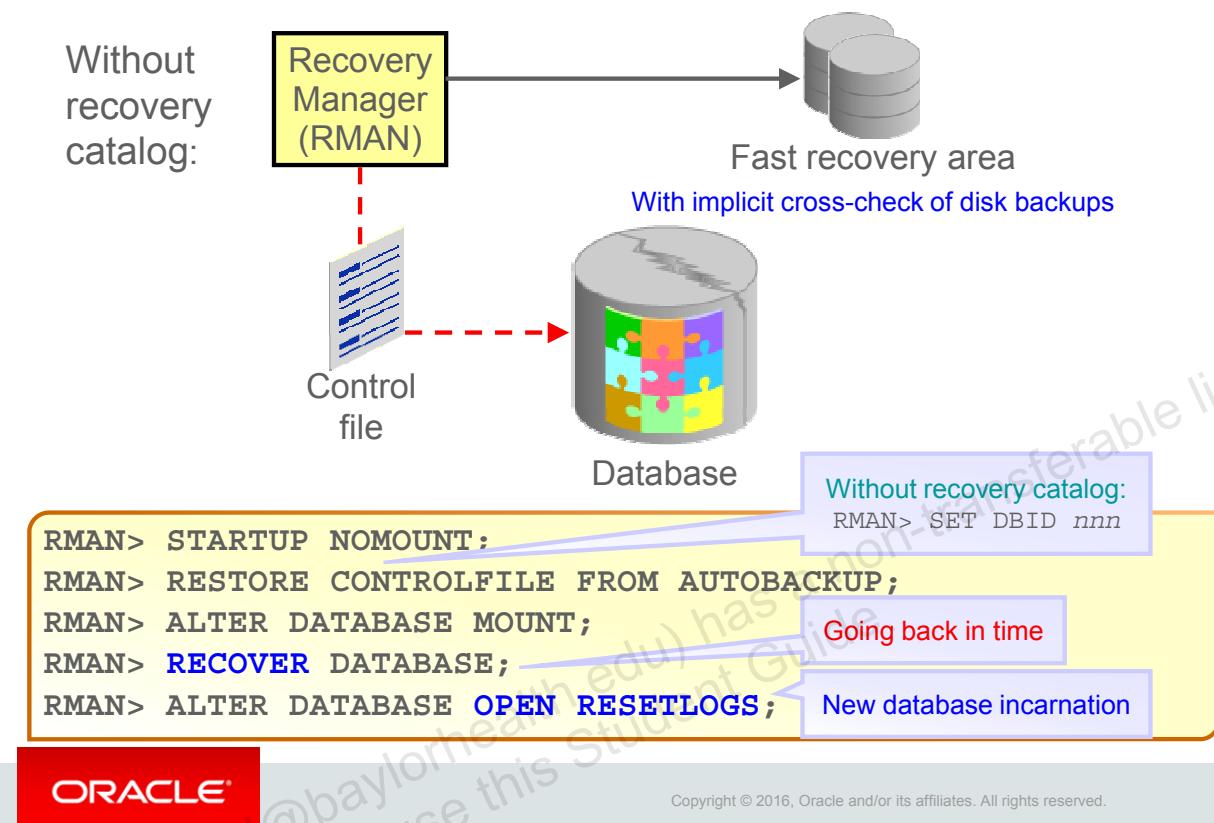
The Oracle database instance can start up with a missing temporary file. If any of the temporary files do not exist when the database instance is started, they are created automatically and the database opens normally. When this happens, a message similar to the following appears in the alert log during startup:

Re-creating tempfile /u01/app/oracle/oradata/orcl/temp01.dbf

In the unlikely case that you decide a manual re-creation serves you better, use the following commands:

```
SQL> ALTER TABLESPACE temp ADD TEMPFILE
'/u01/app/oracle/oradata/orcl/temp02.dbf' SIZE 20M;
SQL> ALTER TABLESPACE temp DROP TEMPFILE
'/u01/app/oracle/oradata/orcl/temp01.dbf';
```

Loss of All Control Files



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

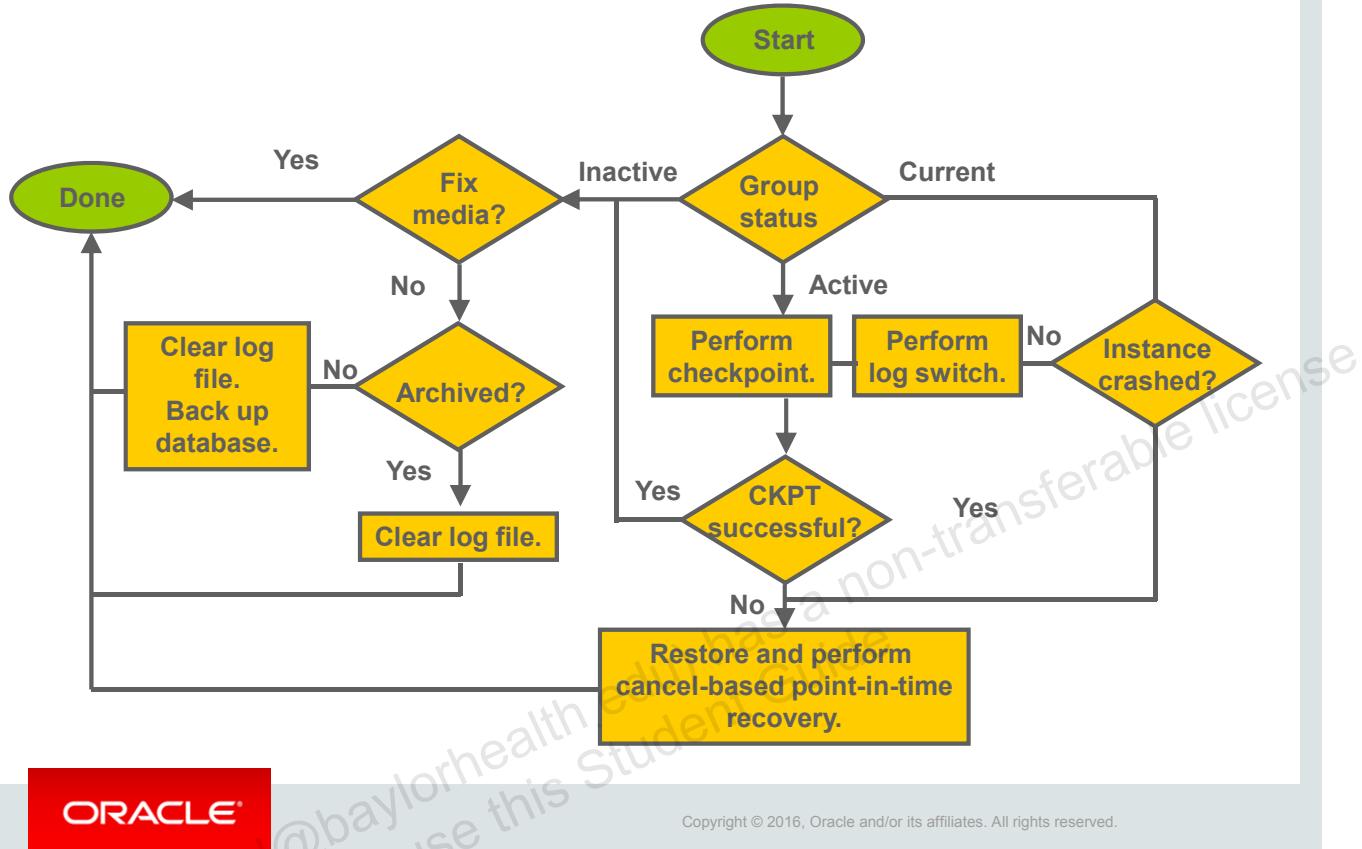
Oracle Corporation recommends that you should have AUTOBACKUP of the control file configured, so that you are able to quickly restore the control file if needed. The commands used for restoring your control file are the same, whether or not you are using a fast recovery area. However, if you are using a fast recovery area, RMAN implicitly cross-checks backups and image copies listed in the control file, and catalogs any files in the fast recovery area that are not recorded in the restored control file; thereby improving the usefulness of the restored control file in the restoration of the rest of your database.

Use the commands shown in the slide to recover from lost control files:

1. Start the instance in NOMOUNT mode. It cannot be mounted because there is no control file.
2. Restore the control file from backup.
3. Now that there is a control file, you can mount the database.
4. You must recover the database, because you now have a backup control file that contains information about an older version of the database.
5. After recovering the database, you can open it. You must specify RESETLOGS because the new control file represents a different instantiation of the database.

Note: Tape backups are not automatically cross-checked after the restoration of a control file. After restoring the control file and mounting the database, you must cross-check the backups on tape.

Loss of a Redo Log Group



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If you have lost an entire redo log group, all copies of the log files for that group are unusable or gone.

The simplest case is where the redo log group is in the `INACTIVE` state. That means it is not currently being written to, and it is no longer needed for instance recovery. If the problem is temporary, or you are able to fix the media, then the database continues to run normally, and the group is reused when enough log switch events occur. Otherwise, if the media cannot be fixed, you can clear the log file. When you clear a log file, you are indicating that it can be reused.

If the redo log group in question is `ACTIVE`, even though it is not currently being written to, it is still needed for instance recovery. If you are able to perform a checkpoint, the log file group is no longer needed for instance recovery, and you can proceed as if the group were in the `inactive` state.

If the log group is in the `CURRENT` state, then it is, or was, being actively written to at the time of the loss. You may even see the LGWR process fail in this case. If this happens, the instance crashes. Your only option at this point is to restore from backup, perform cancel-based point-in-time recovery, and then open the database with the `RESETLOGS` option.

Clear a log file using the following command:

```
ALTER DATABASE CLEAR [UNARCHIVED] LOGFILE GROUP <n>
[UNRECOVERABLE DATAFILE]
```

When you clear a log file, you are indicating that it can be reused. If the log file has already been archived, the simplest form of the command can be used. Use the following query to determine which log groups have been archived:

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

For example, the following command clears redo log group 3, which has already been archived:

```
SQL> ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

If the redo log group has not been archived, you must specify the UNARCHIVED keyword. This forces you to acknowledge that it is possible that there are backups that rely on that redo log for recovery, and you have decided to forgo that recovery opportunity. This may be satisfactory for you, especially if you take another backup right after you correct the redo log group problem; you then no longer need that redo log file.

It is possible that the redo log is required to recover a data file that is currently offline.

Loss of Data Files

- Example: Loss of SYSTEM or UNDO data files

```
RMAN> STARTUP MOUNT;
RMAN> RESTORE TABLESPACE und01;
RMAN> RECOVER TABLESPACE und01;
RMAN> ALTER DATABASE OPEN;
```

- Example: Loss of noncritical data files

```
RMAN> ALTER TABLESPACE tbs2 OFFLINE IMMEDIATE;
RMAN> RESTORE TABLESPACE tbs2;
RMAN> RECOVER TABLESPACE tbs2;
RMAN> ALTER TABLESPACE tbs2 ONLINE;
```



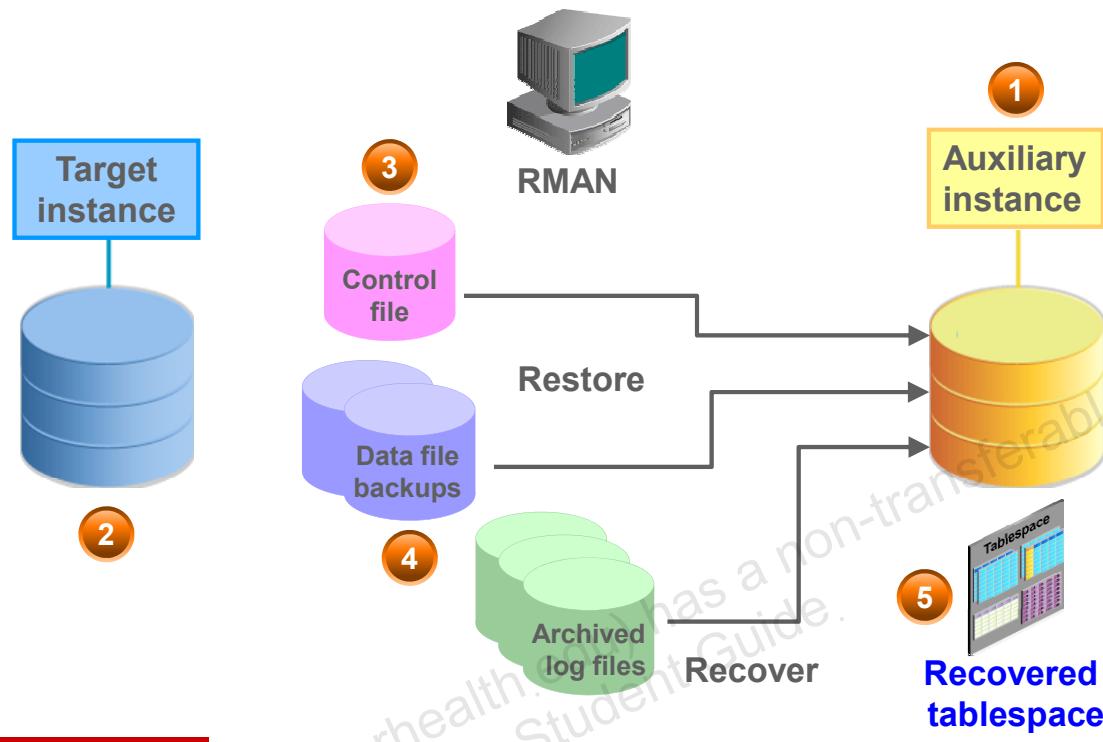
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If the missing or corrupted data file belongs to the SYSTEM or UNDO tablespace, the instance requires a shutdown and a media recovery. In a RAC environment, you would shut all instances down. The database must be mounted before restoring and recovering the missing data file. After the data file is recovered, open the database.

If the data file that is missing or corrupted is a data file of the tablespaces other than SYSTEM or UNDO, you offline the tablespace, and then perform a tablespace media recovery. The database remains open while restoring and recovering the missing data file.

After the data file is recovered, bring the tablespace online.

Tablespace Point-in-Time Recovery



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

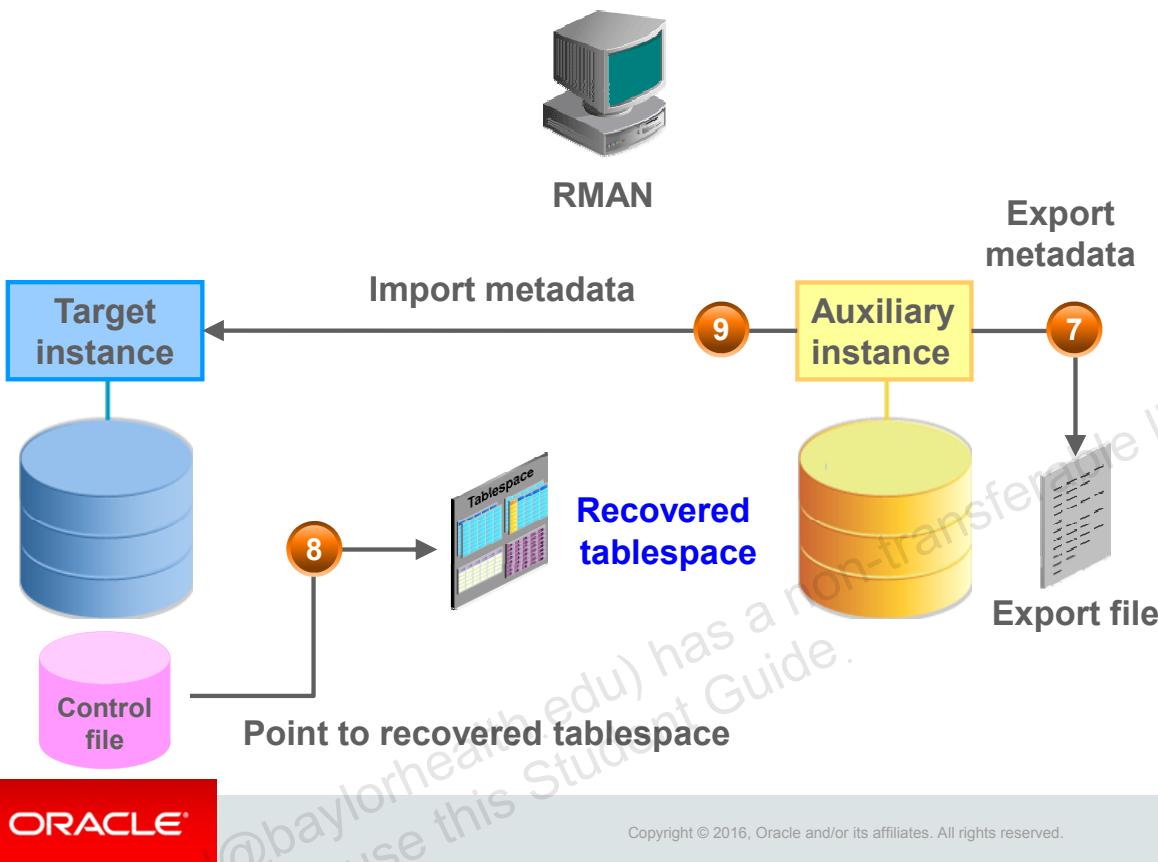
In the diagram in the slide, the following TSPITR entities are shown:

- **Target database**: Contains the tablespace to be recovered
- **Control file**: Provides backup information to RMAN
- **Backup sets**: Come from the target database and are the source of the reconstructed tablespace
- **Archived redo logs**: Come from the target database and are the source of the reconstructed tablespace
- **Auxiliary instance**: Is the Oracle database instance used during the recovery process to perform the recovery

RMAN performs the following steps:

1. Creates the auxiliary instance, starts it, and connects to it
2. Takes the tablespaces that will be recovered offline
3. Restores a backup control file from a point in time before the target time to the auxiliary instance
4. Restores the data files from the recovery set and the auxiliary set to the auxiliary instance
5. Recovers the restored data files to the specified time
6. Opens the auxiliary database with the RESETLOGS option

Tablespace Point-in-Time Recovery



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

RMAN processing steps (continued):

7. Exports the dictionary metadata about objects in the recovered tablespaces to the target database
8. Shuts down the auxiliary instance (Issues `SWITCH` commands on the target database so that the target database control file points to the data files in the recovery set that were recovered on the auxiliary instance)
9. Imports the dictionary metadata from the auxiliary instance to the target instance
10. Deletes all auxiliary set files

Flashback Database

- Flashback Database: Database mounted in exclusive mode.

```
SQL> STARTUP MOUNT  
SQL> FLASHBACK DATABASE TO SCN 53943;
```

- To review changes: Open database in READ ONLY.

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

- To finalize: Flash back again if necessary and open database with RESETLOGS.

```
RMAN> SHUTDOWN IMMEDIATE  
RMAN> STARTUP MOUNT  
RMAN> FLASHBACK DATABASE TO SCN 10;  
RMAN> ALTER DATABASE OPEN RESETLOGS;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can use the RMAN FLASHBACK DATABASE command to execute the Flashback Database operation. You can use SEQUENCE and THREAD to specify a redo log sequence number and thread as a lower limit.

Alternatively, you can use the SQL FLASHBACK DATABASE command to return the database to a past time or SCN. If you use the TO SCN clause, you must provide a number. If you specify TO TIMESTAMP, you must provide a time stamp value. You can also specify a restore point name.

Notes

- The database must be mounted in exclusive mode to issue the FLASHBACK DATABASE command and opened read-only to review changes.
- The database must be opened read/write with the RESETLOGS option when finished.

Flashback Table

- Recovering a table or tables with associated objects to a specific point in time without restoring a backup
- Using data from the undo tablespace
- Flashback Table operations:
 - Cannot be performed on system tables
 - Cannot span DDL operations
 - Generate undo and redo data
- Example:

```
FLASHBACK TABLE hr.departments TO TIMESTAMP  
TO_TIMESTAMP('2013-01-25 21:00:00',  
'YYYY-MM-DD HH24:MI:SS');
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

With Flashback Table, you can recover a table or tables to a specific point in time without restoring a backup. When you use this feature, the data in tables and their associated objects (indexes, constraints, triggers, and so on) is restored. The data used to satisfy a Flashback Table request is retrieved from the undo tablespace. You can use Flashback Version Query and Flashback Transaction Query to determine the appropriate flashback time.

Flashback Table provides a way for users to easily and quickly recover from accidental modifications without a database administrator's involvement. You must grant the FLASHBACK TABLE or FLASHBACK ANY TABLE system privilege to any user that uses the Flashback Table feature. In addition, you must grant the SELECT, INSERT, DELETE, and ALTER object privileges to the user.

You can use Enterprise Manager or SQL*Plus to flash back a table. The wizard guides you through the process.

You must enable row movement on a table to be able to flash back the table. When you enable row movement, the Oracle server can move a row in the table. You can enable row movement by using Enterprise Manager or by using the ALTER TABLE command.

Recovering Tables from Backups in 12c

When to recover tables and table partitions from RMAN backups:

- Small number of tables (no tablespace point-in-time recovery [TSPITR])
- Not in a self-contained tablespace (no TSPITR)
- Purged tables (no Flashback drop)
- Beyond the available undo (no Flashback Table)
- After a structural DDL change (no Flashback Table)
- Example:

```
RECOVER TABLE HR.EMP UNTIL TIME 'SYSDATE-4'  
AUXILIARY DESTINATION '/tmp/backups'  
REMAP TABLE 'HR'.'EMP':'EMP_RECVR';
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c and later, RMAN enables you to recover one or more tables or table partitions to a specified point in time without affecting the remaining database objects. Recovering tables and table partitions is useful in the following situations:

- You need to recover a very small number of tables to a particular point in time. In this situation, TSPITR is not the most effective solution because it moves all the objects in the tablespace to a specified point in time.
- You need to recover a tablespace that is not self-contained to a particular point in time. TSPITR can be used only if the tablespace is self-contained.
- You need to recover tables that have either been corrupted or deleted by using the PURGE option, so you cannot use the Flashback Drop functionality.
- You enabled logging for a Flashback Table, but the flashback target time or SCN is beyond the available undo.
- You want to recover data that is lost after a data definition language (DDL) operation has changed the structure of tables. You cannot use Flashback Table to rewind a table to before the point of a structural change, such as a truncate table operation.

Lesson Agenda

- Concept Review
 - Complete media recovery (Media Failure)
 - Incomplete media recovery (Media or Logical Failure)
 - Flashback Technology (Logical Failure)
- Recovery Considerations in Oracle Data Guard
 - Recovery from Media Failure
 - Recovery from Logical Failure



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Recovery Considerations in Oracle Data Guard

- Recovery from Media Failure
 - Complete Recovery of Primary or Standby Database
 - Incomplete Recovery of Primary Database
 - Recovery of Standby Control Files
 - Actions After TSPITR on the Primary Database
 - Recovery from Loss of Online Redo Log Files
- **Assumption:** The standby database shown in the following scenarios is the physical standby database.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Scenario 1: Complete Recovery

- **Option 1:** Restore the Backup of the Primary Database
- **Option 2:** Restore the missing file over the network (12.1)



- Example for Option 2

```
RMAN> RESTORE TABLESPACE users, example FROM SERVICE 'westdb';
```

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

As of Oracle Database 12c Release 1 (12.1), you can restore and recover files over the network by connecting to a physical standby database that contains the required files. This can be useful when you want to restore lost data files, control files, or tablespaces on a primary database using the corresponding files on the physical standby database. You can also use the same process to restore files on a physical standby database by using the primary database.

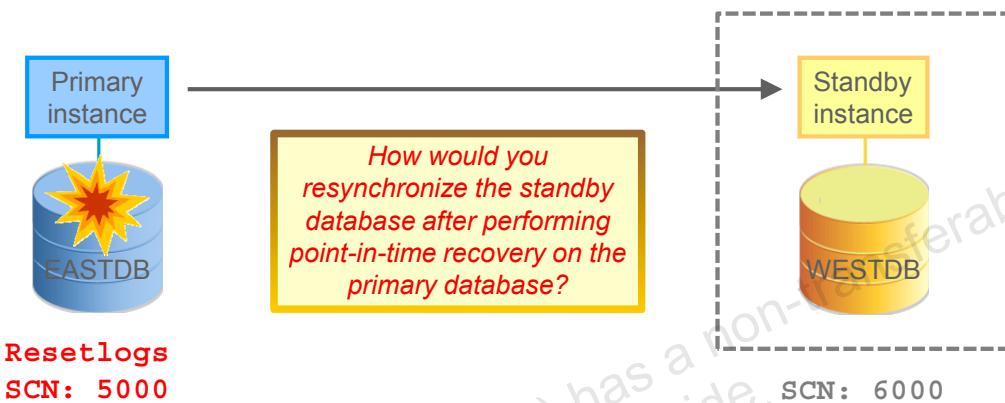
For an example of how to restore and recover files by connecting over the network, see *Oracle Database Backup and Recovery User's Guide*.

Note: In releases before Oracle Database 12c, to recover from loss of files on the primary, you used the RMAN recovery catalog, and the RMAN BACKUP, CATALOG DATAFILE, and SWITCH DATAFILE commands. To recover from loss of files on the standby, you used the RESTORE and RECOVER commands. Those methods are no longer necessary in Oracle Database 12c. If you need information about using them, refer to Oracle Database 11g documentation.

Scenario 2: Incomplete Recovery

Assumptions:

- Point-In-Time Recovery on **EASTDB** until SCN = 5000
- **WESTDB**: Current SCN = 600



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Physical Standby Configuration

For a physical standby database, it is possible that the Redo Apply service might not halt when it encounters the `OPEN RESETLOGS` command in the redo information. If the physical standby database's system change number (SCN) is far less than the primary database's SCN, the Redo Apply service can interpret the `OPEN RESETLOGS` command without stopping. This feature is known as “recovery through `RESETLOGS`.”

Use the following procedure to avoid re-creating a standby database after you issue an `OPEN RESETLOGS` command on the primary database and the managed recovery process is halted on the physical standby database:

1. On the primary database, determine an SCN that is at least two SCNs before the SCN when the `OPEN RESETLOGS` command was issued. This is necessary to enable the standby to recover properly through `OPEN RESETLOGS`. Use the following query to find the “before `RESETLOGS`” SCN:

```
SELECT TO_CHAR(resetlogs_change# - 2) FROM v$database;
```

2. On the standby database, obtain the current SCN by using the following query:
SELECT TO_CHAR(current_scn) FROM v\$database;
3. Flash back the standby database to the “before RESETLOGS” SCN that you queried in step 1:
FLASHBACK STANDBY DATABASE TO SCN <before RESETLOGS SCN>;
4. Restart managed recovery on the standby database. The standby database will be ready to receive and apply logs from the primary database.
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;

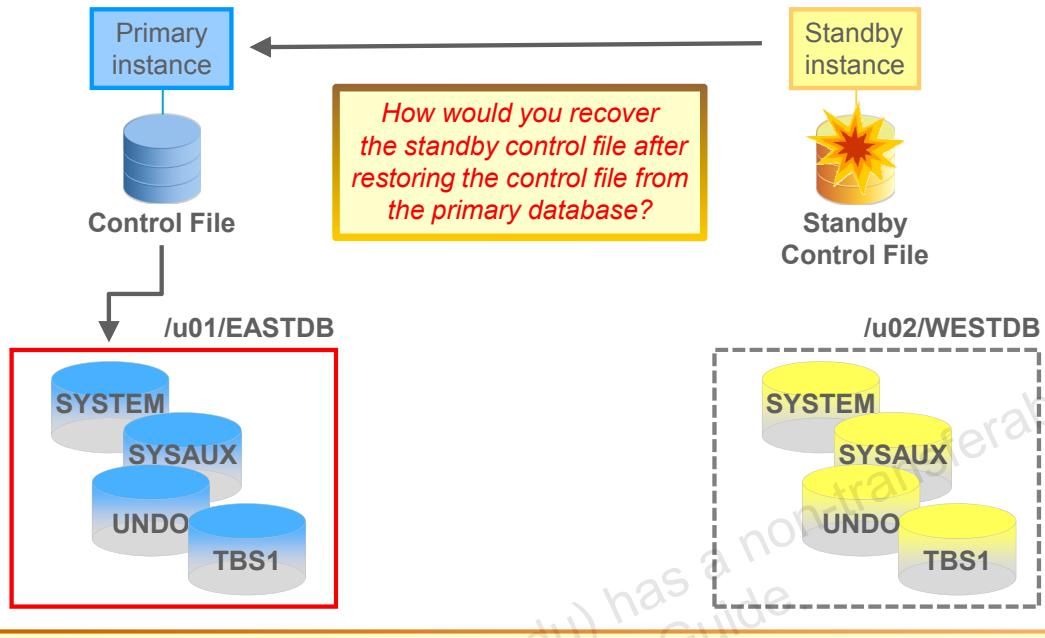
Logical Standby Configuration

For a logical standby database, it is possible that the SQL Apply service might not halt when it encounters the OPEN RESETLOGS command in the redo information. If the logical standby database’s SCN is far less than the primary database’s SCN, the SQL Apply service will be able to interpret the OPEN RESETLOGS command without stopping.

Use the following procedure to avoid re-creating a standby database after you perform an OPEN RESETLOGS on the primary database and the SQL Apply process has halted on the logical standby database:

1. On the primary database, determine an SCN that is at least two SCNs before the SCN when the OPEN RESETLOGS command was issued. This is necessary to enable the standby to recover properly through OPEN RESETLOGS. Use the following query to find the “before RESETLOGS” SCN:
SELECT TO_CHAR(resetlogs_change# - 2) FROM v\$database;
2. On the standby database, obtain the current SCN by using the following query:
SELECT TO_CHAR(current_scn) FROM v\$database;
3. Flash back the standby database to the “before RESETLOGS” SCN that you queried in step 1:
FLASHBACK STANDBY DATABASE TO SCN <before RESETLOGS SCN>;
4. Restart SQL Apply on the standby database. The standby database will be ready to receive and apply logs from the primary database.
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;

Scenario 3: Standby Control Files Recovery



```
RMAN> RESTORE STANDBY CONTROLFILE FROM SERVICE 'eastdb';
```

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

As of Oracle Database 12c Release 1 (12.1), you can restore and recover files over the network by connecting to a physical standby database that contains the required files. This can be useful when you want to restore lost data files, control files, or tablespaces on a primary database by using the corresponding files on the physical standby database. You can also use the same process to restore files on a physical standby database by using the primary database.

The following is an outline of a possible solution to recover the standby control files:

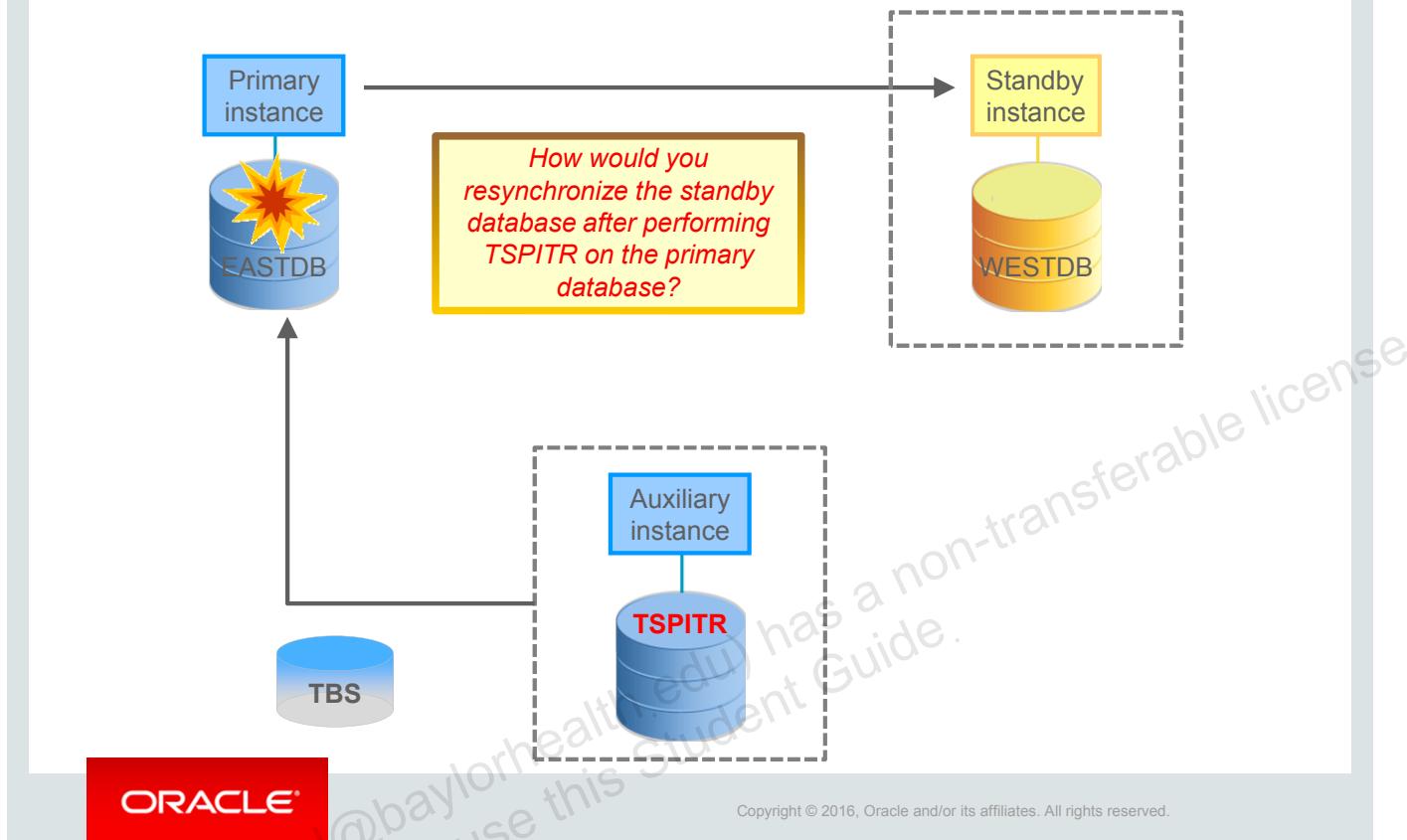
- Identify the issue on the standby database.
- Restore the standby control file from the primary database over the network:

```
RMAN> run {
  Shutdown immediate;
  Startup nomount;
  Restore standby controlfile from service eastdb; }
```

- Recover the restored standby control file in the standby database.

```
RMAN> catalog start with '+DATA/WESTDB/DATAFILE/';
RMAN> switch database to copy;
SQL> alter database recover managed standby database disconnect
from session;
```

Scenario 4: Tablespace Point-In-Time Recovery



After an RMAN tablespace point-in-time recovery (TSPITR) is performed at the primary, the recovered data files have a new SCN, and are therefore treated like new data files at the primary. These data files cannot be automatically created at the standby.

Likewise, when a new plugged-in tablespace is added to the primary database, the data files are treated like new data files at the primary.

The managed recovery process (MRP) at the standby stops when the Redo Apply process encounters creation of these new files. The required new data files must be copied and restored to the standby. You can do this using either backups or a direct copy from the primary. For example, to copy all files that belong to a tablespace that has undergone an RMAN TSPITR, you can use the following RMAN command:

```
RMAN> RESTORE TABLESPACE <tbs_name1, tbs_name2> FROM SERVICE  
<tnsalias-of-primary>
```

Note that the number of disk channels allocated will be as per RMAN configurations. So, if `CONFIGURE DEVICE TYPE DISK PARALLELISM 4` is executed, then four disk channels will be used to pull the files from the primary database.

When the new data files are available at the standby, MRP should be restarted to continue applying the logs.

Recovery Considerations in Oracle Data Guard

- Recovery from Logical Failure:
 - Using Flashback Database
 - Using Standby Database
 - Using Primary Database
- Assumption: The standby database shown in the scenarios in the following slides is the physical standby database.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Incomplete Recovery of the Primary Database: Overview

- The following scenarios are all to perform incomplete recovery of the primary database in case of logical failures.
- All the scenarios are in order of preference, starting with the one that is the least time-consuming.
 - **Scenario 5:** Using Flashback Database
 - **Scenario 6:** Using Standby Database
 - **Scenario 7:** Using Primary Database



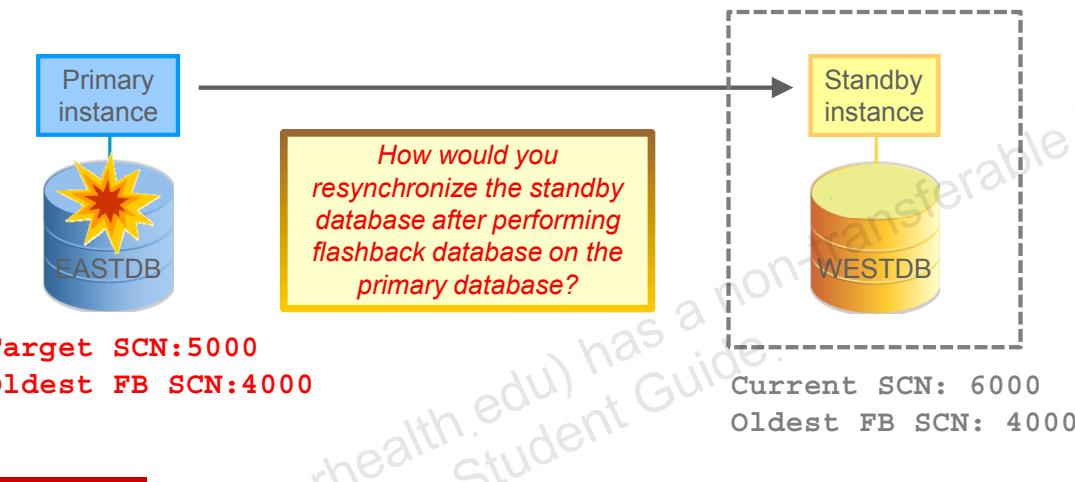
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Incomplete recovery of the primary database is normally done in cases such as when the database is logically corrupted (by a user or an application) or when a tablespace or data file was accidentally dropped from database.

Depending on the current database checkpoint SCN on the standby database instances, you can use one of the procedures in the following slides to perform incomplete recovery of the primary database.

Scenario 5: Using Flashback Database

- Assumptions:
 - Flashback Database is Enabled on the Primary Database.
 - Flashback Database is Enabled on the Standby Database.
 - None of the Database Files are lost.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Flashback Database is the recommended procedure when the Flashback Database feature is enabled on the primary database, none of the database files are lost, and the point-in-time recovery is greater than the oldest flashback SCN or the oldest flashback time.

The following is the procedure to use Flashback Database to do point-in-time recovery:

- On the primary database, determine an SCN that is at least two SCNs before the SCN when the OPEN RESETLOGS command was issued. This is necessary to enable the standby to recover properly through OPEN RESETLOGS. Use the following query to find the “before RESETLOGS” SCN:

```
SELECT TO_CHAR(resetlogs_change# - 2) FROM v$database;
```

- On the standby database, obtain the current SCN by using the following query:

```
SELECT TO_CHAR(current_scn) FROM v$database;
```

- Flash back the standby database to the “before RESETLOGS” SCN that you queried in step 1:

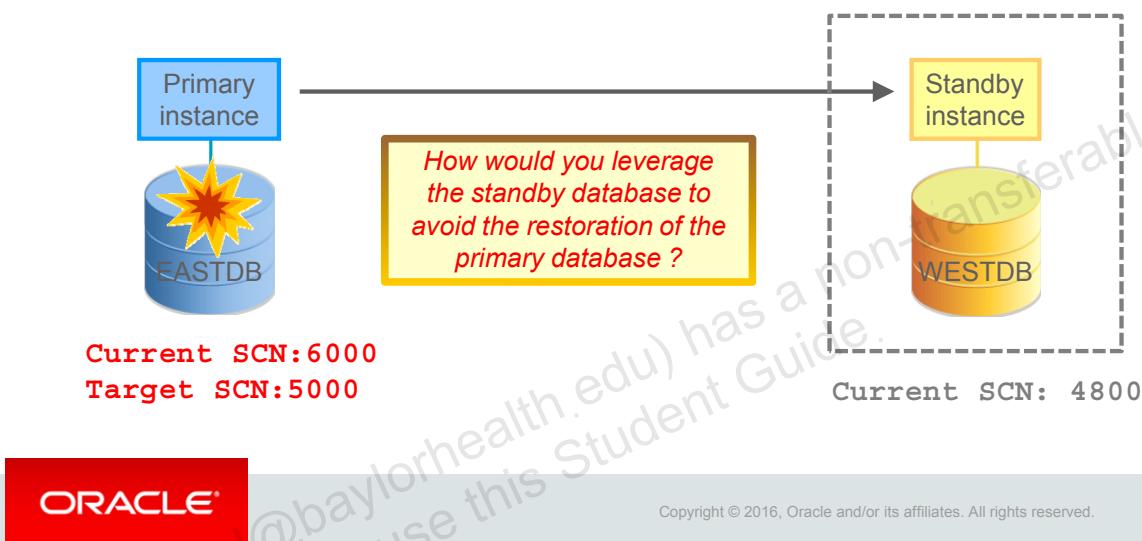
```
FLASHBACK STANDBY DATABASE TO SCN <before RESETLOGS SCN>;
```

- Restart managed recovery on the standby database. The standby database will be ready to receive and apply logs from the primary database.

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

Scenario 6: Using Standby Database

- Assumptions:
 - Flashback Database Not Enabled on the Primary Database
 - Flashback Database Not Enabled on the Standby Database
 - Recovery Target SCN > Standby SCN



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This is the recommended procedure when the standby database is behind the desired incomplete recovery time, and Flashback Database is not enabled on the primary or standby databases:

1. Recover the standby database to the desired point in time. Be sure to stop the managed recovery process (MRP) before issuing the following command:

```
RECOVER DATABASE UNTIL TIME 'time';
```

Alternatively, incomplete recovery time can be specified using the SCN or log sequence number:

```
RECOVER DATABASE UNTIL SCN incomplete recovery SCN;
```

```
RECOVER DATABASE UNTIL LOGSEQ incomplete recovery log seq number
THREAD thread number;
```

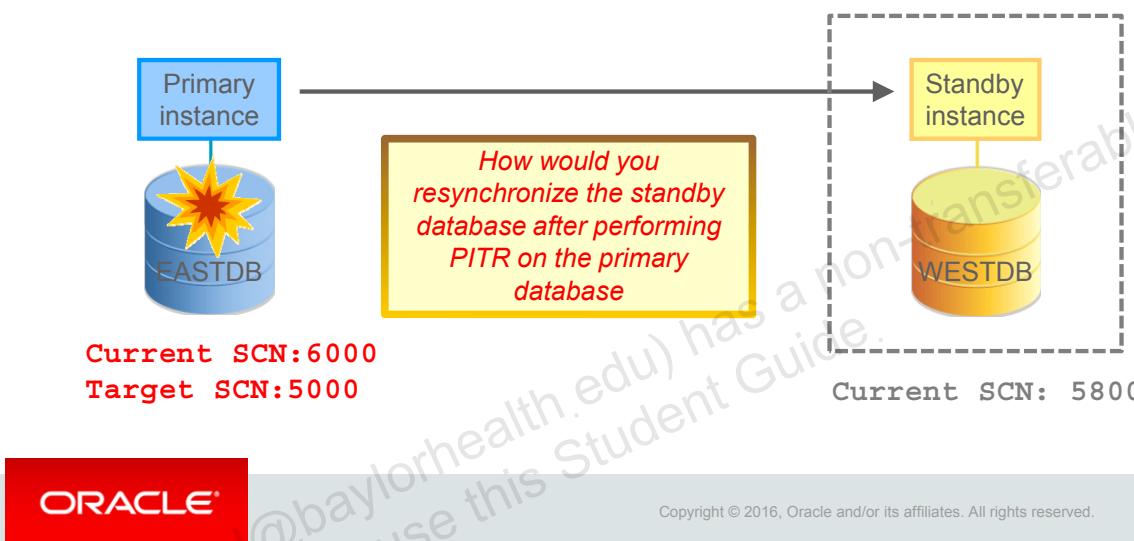
2. Open the standby database in read-only mode to verify the state of the database.

If the state is not what is desired, use the LogMiner utility to look at the archived redo log files to find the right target time or SCN for incomplete recovery. Alternatively, you can start by recovering the standby database to a point that you know is before the target time, and then open the database in read-only mode to examine the state of the data. Repeat this process until the state of the database is verified to be correct. Note that if you recover the database too far (that is, past the SCN where the error occurred), you cannot return it to an earlier SCN.

Activate the standby database by using the SQL ALTER DATABASE ACTIVATE STANDBY DATABASE statement. This converts the standby database to a primary database, creates a new resetlogs branch, and opens the database.

Scenario 7: Using Primary Database

- Assumptions:
 - Flashback Database Not Enabled on the Primary Database
 - Flashback Database Not Enabled on the Standby Database
 - Recovery Target SCN < Standby SCN



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

If all of the standby database instances have already been recovered past the desired point in time and Flashback Database is not enabled on the primary or standby database, then this is your only option.

Use the following procedure to perform incomplete recovery on the primary database:

- Use LogMiner or another means to identify the time or SCN at which all the data in the database is known to be good.
- Using the time or SCN, issue the following RMAN commands to do incomplete database recovery and open the database with the RESETLOGS option (after connecting to the catalog database and the primary instance that is in the MOUNT state):

```
RUN {
  SET UNTIL TIME 'time';
  RESTORE DATABASE;
  RECOVER DATABASE;
}
ALTER DATABASE OPEN RESETLOGS;
```

After this process, all standby database instances must be re-established in the Oracle Data Guard configuration.

Summary

In this lesson, you should have learned how to:

- Explain the basic concept of Oracle Database Recovery
- Perform the appropriate type of restore and recovery operation based on the nature of your database failure in an Oracle Data Guard environment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 9: Overview

This practice covers the following topics:

- Reconfiguring the Environment
- Performing Complete Recovery on the Primary Database Over the Network in a Data Guard Environment
- Recovering Standby Control Files (Media Failure)
- Performing the Steps for Rolling Forward a Physical Standby Database Using RMAN Incremental Backup
- Performing Incomplete Recovery Using Flashback Database (Logical Failure)
- Performing Actions After TSPITR on the Primary Database



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Performing Data Guard Standby-First Patch Apply

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to patch in your Data Guard configuration by using the Standby-First Patch Apply method.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Background: Data Guard Support for Heterogeneous Configuration

- Data Guard has long supported running different configuration between primary and standby systems.
- This included support for the following:
 - Differences in hardware
 - Differences in operating system
 - Differences in database storage
 - Differences in Oracle Clusterware version and patch level



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Data Guard has long supported running different configuration between primary and standby systems. This has included support for the following:

- Differences in hardware (for example, X3 Exadata Database Machine with X4 Exadata Database Machine)
- Differences in operating system (for example, Oracle Linux 5.7 with Oracle Linux 5.8)
- Differences in database storage (for example, Oracle ASM-based storage with NFS-based storage, or Exadata 11.2 with Exadata 12.1)
- Differences in Oracle Clusterware version and patch level (for example, 11.2.0.3 GIPSU4 with 11.2.0.3 GIPSU5)

Refer to My Oracle Support notes 413484.1, 395982.1, and 414043.1 for additional information.

Background: Data Guard Support for Heterogeneous Configuration

- However, differences in **database home software** were limited to rolling upgrade scenarios supported only by logical standby databases.
Example: Patching with limited heterogeneous support
- To apply a later database home patch (for example, Exadata bundle patch or database PSU) to a Data Guard environment with physical standby, you had to perform one of the following actions:
 - Shut down both the primary and standby databases and apply the update to both systems before restarting.
 - Convert the physical standby database to a logical standby database and apply the update using the rolling upgrade process. Then convert the standby database back to a physical standby (a feature known as transient logical standby).



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Data Guard Standby-First Patch: Overview

Provides support for different software releases between a primary database and its physical standby databases for applying and validating patches in a rolling fashion

- The COMPATIBLE RDBMS parameter must remain the same on all systems.
- The database patch release dates must be no more than 365 days apart.
- The patches must be within six versions of each other.
- Data Guard role transitions are allowed between different releases on the primary and physical standby databases for Standby-First Patch.
- The patch must be certified as being a “Standby-First” patch.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

With Data Guard Standby-First Patch Apply, Oracle supports different database home software between a primary database and its physical standby databases, in addition to the differences listed in the slide.

Oracle Data Guard Standby-First Patch Apply provides support for different database home software between a primary database and its physical standby databases for applying and validating Oracle patches and patch bundles in rolling fashion with minimal risk to the primary database. For example, with Data Guard Standby-First Patch Apply, you apply a database home patch first to a physical standby database. The standby is used to run read-only workload, or read/write workload if it is a snapshot standby, for testing and evaluation of the patch. After passing evaluation, the patch is then installed on the primary system with greater assurance of the effectiveness and stability of the database home patch.

Data Guard Standby-First Patch Installable

The following types of patches are supported:

- Oracle Exadata Database Machine bundled patch or Quarter Database Patch for Exadata
- Oracle Exadata Storage Server Software Update
- Oracle Exadata Database Machine hardware or network changes
- Oracle Grid Infrastructure Patches or Software Updates
- Patch Set Update (PSU)
- Critical Patch Update (CPU)
- Patch Set Exception (PSE)
- Operating System software changes that do not have any dependencies on the Oracle database software



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Data Guard Standby-First Patch Apply is supported only for certified interim patches and patch bundles (for example, Patch Set Update, or Database Patch for Exadata) for Oracle Database 11.2.0.1 and later, on both Oracle Engineered Systems (for example, Exadata, SuperCluster) and non-Engineered Systems.

Oracle patch sets and major release upgrades do not apply. For example, upgrades from 11.2.0.2 to 11.2.0.3 or 11.2 to 12.1 do not qualify for standby-first patch apply. Use the Data Guard transient logical standby method for patch sets and major releases.

Refer to the README file for the patch to determine whether a target patch is certified as being a “Standby-First” or “DG Rolling Installable” patch.

README File: Example

```
[oracle@enode01 20108098]$ cat README.txt

Oracle Database 12c Release 12.1.0.2.0
ORACLE DATABASE Patch for Bug# 20108098 for Linux-x86-64
Platforms

This patch is RAC Rolling Installable - Please read My
Oracle Support Document 244241.1
https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=244241.1 Rolling Patch - OPatch
Support for RAC.

This patch is Data Guard Standby-First Installable
Please read My Oracle Support Note 1265700.1
https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=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.
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A patch and patch bundle that is Data Guard Standby-First certified will state the following in the patch README: **Data Guard Standby-First Installable**.

The following types of patches are candidates to be Data Guard Standby-First certified:

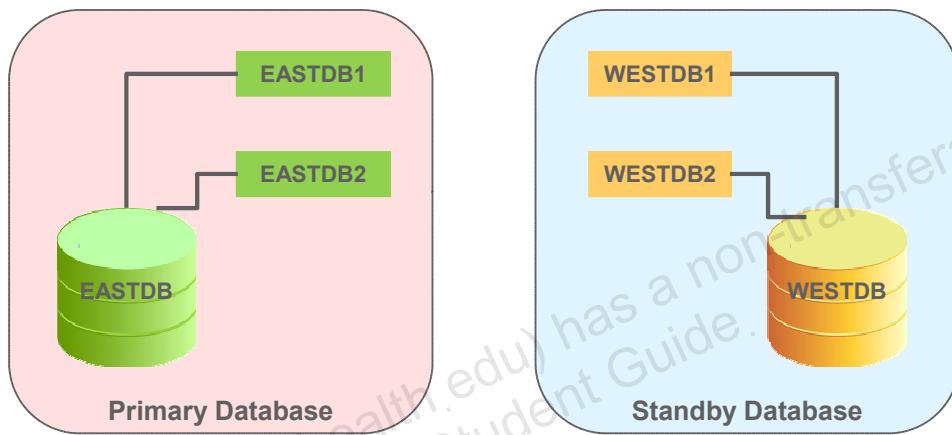
- Database home interim patches
- Exadata bundle patches (for example, monthly and quarterly database patches for Exadata)
- Database patch set updates

Note: Patches and patch bundles that update modules that may potentially disrupt the interoperability between primary and physical standby systems running different database home software will not be certified "Data Guard Standby-First Installable" and will not state so in the patch README.

Data Guard Standby-First Patch Apply: Phases

To accomplish Data Guard Standby-First Patch Apply, do the following:

- **Phase 1:** Perform Patch Binary Installation on Standby Only
- **Phase 2:** Evaluate Patch on Standby Database
- **Phase 3:** Complete Patch Installation or Rollback



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

To accomplish Data Guard Standby-First Patch Apply, do the following:

- **Phase 1:** Perform Patch Binary Installation on Standby Only
- **Phase 2:** Evaluate Patch on Standby Database
- **Phase 3:** Complete Patch Installation or Rollback

Option 1: Apply Patch to Primary Database

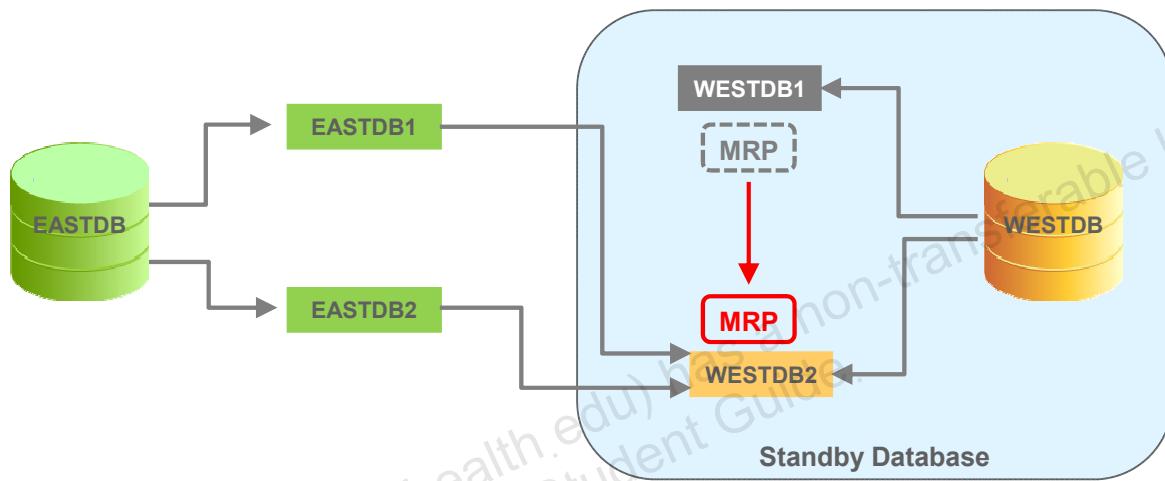
Option 2: Data Guard Switchover, Apply Patch to New Standby

Option 3: Rollback Patch on Standby System

Phase 1: Patch 1st Standby Host

Assumptions:

- The patch is RAC Rolling Installable.
- The patch is Data Guard Standby-First Installable.
- MRP is currently running on WESTDB1.



ORACLE®

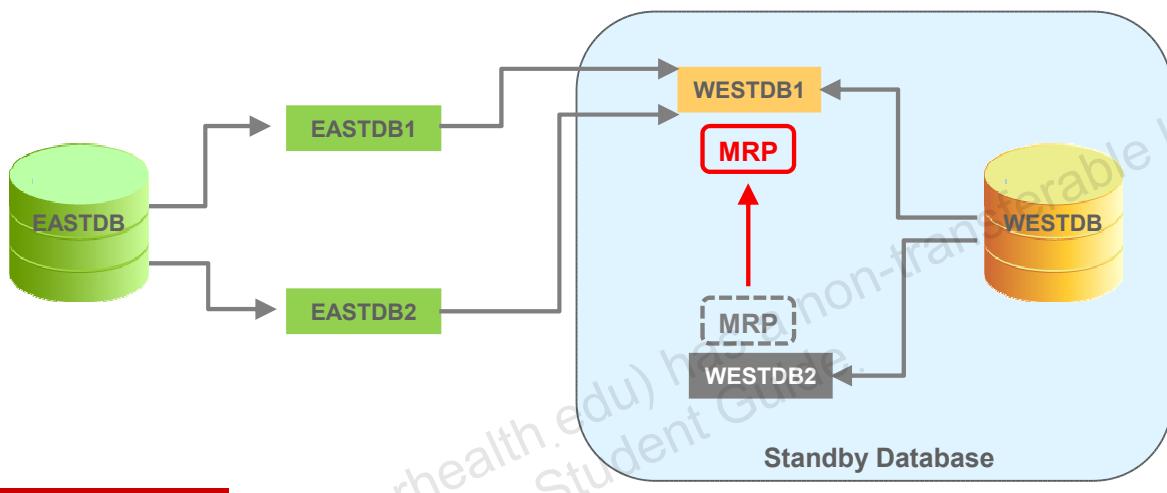
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Start the managed recovery process (MRP) in the 2nd standby instance (WESTDB2) if the current apply instance is the 1st standby instance (WESTDB1).
2. Shut down the 1st standby instance on the standby database if patch is RAC Rolling.
3. On the 1st standby host, apply the patch as described in the patch README.
Note: Do not perform SQL installation (for example, do not run `catbundle.sql` or `datapatch`) for the patch at this time. SQL installation is performed after the primary database and all standby databases have their database home binaries patched to the same level in Phase 3.
4. Restart the 1st standby instance.
5. Relocate the managed recovery process (MRP) back to the 1st standby instance.

Phase 1: Patch 2nd Standby Host

Assumptions:

- The patch is RAC Rolling Installable.
- The patch is Data Guard Standby-First Installable.
- MRP is currently running on WESTDB2.



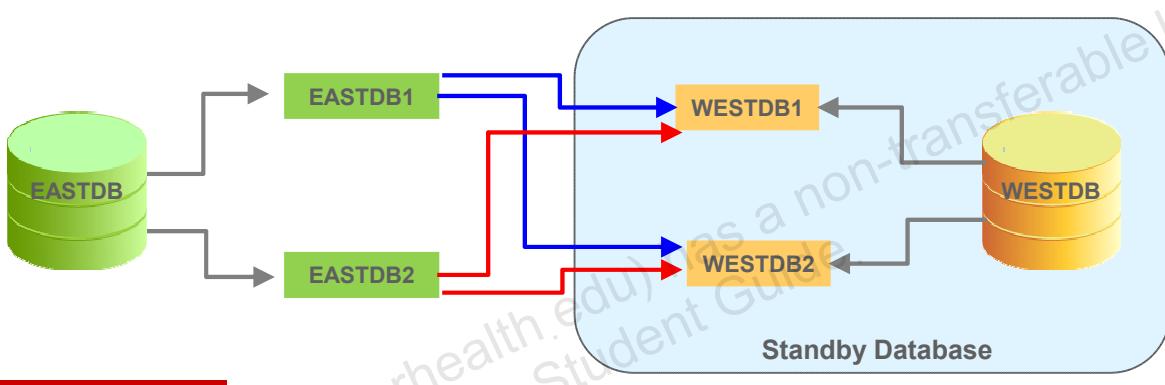
ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Verify the current apply instance. If the apply instance is the 2nd standby instance (WESTDB2), start the MRP process in the 1st standby instance (WESTDB1).
2. Shut down the 2nd standby instance on the standby database.
3. On the 2nd standby host, apply the patch as described in the patch README.
Note: Do not perform SQL installation (for example, do not run catbundle.sql or datapatch) for the patch at this time. SQL installation is performed after the primary database and all standby databases have their database home binaries patched to the same level in Phase 3.
4. Restart the 2nd standby instance.
5. Relocate the MRP process back to the 2nd instance if it is desired.

Phase 2: Evaluate Patch on Standby Database

- The Oracle-recommended best practice and the most comprehensive evaluation method is to use Snapshot Standby and Oracle Real Application Testing.
- The maximum supported duration to have different database home software between primary and standby is 31 days.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The Oracle-recommended best practice and most comprehensive evaluation method is to use Snapshot Standby and Oracle Real Application Testing in the following manner:

1. Convert the standby database into a snapshot standby. (See the *Oracle Data Guard Concepts and Administration Guide*.)
2. Perform any required SQL installation steps for the patch on the snapshot standby.
3. Use Oracle Real Application Testing to evaluate stability and performance of the new software using real application workload.
4. After testing is complete, convert the snapshot standby back to a physical standby. Conversion back to a physical standby will roll back changes made by Oracle Real Application Testing workload replay, and made by SQL installation steps for the patch.

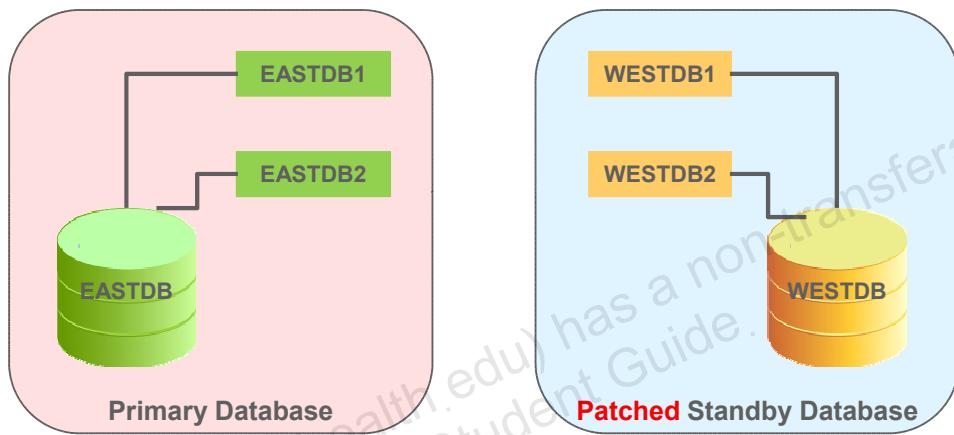
Less comprehensive evaluation can be performed by the following:

- If using the Active Data Guard option, open the standby database in read-only mode and stress the standby database by running your read-only workload.
- Leave the standby database in managed recovery mode at the mount state, and monitor for any issues in the standby alert log and trace files.

Phase 3: Complete Patch Installation or Rollback

There are three options for phase 3.

- Option 1: Apply patch to the primary database.
- Option 2: Execute Switchover and apply patch to the new standby database.
- Option 3: Roll back patch from the standby database.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

At this point, the patch has been applied only to the standby system binaries; therefore, it is only partially installed. The environment may remain in mixed version state for a maximum of 31 days. To complete patch installation, binary installation must be completed on the primary system, and SQL installation (if necessary for the patch) must be performed.

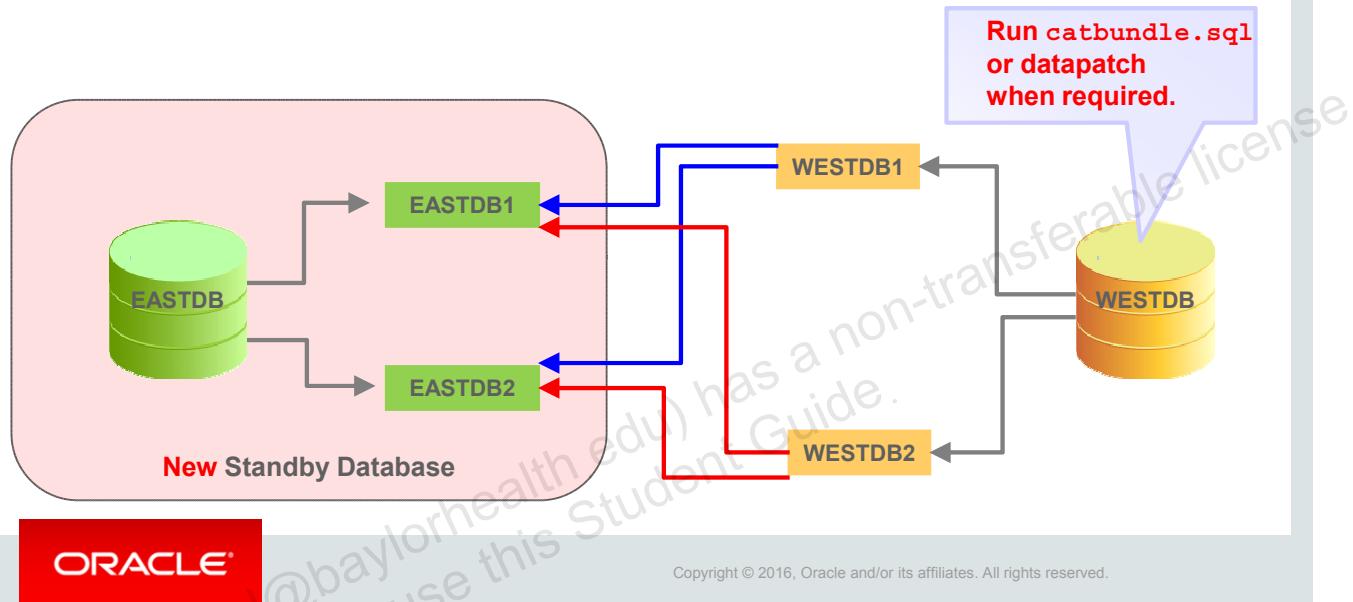
There are three options for Phase 3.

- Option 1 – Apply patch to primary.
- Option 2 – Execute Switchover and apply the patch to standby.
- Option 3 – Roll back the patch from standby.

Phase 3: Data Guard Switchover and Apply Patch to New Physical Standby (Option 2)

Assumptions:

- The Data Guard Switchover was performed.
- The patch installation on New Standby hosts has been completed.



The main steps in Option 2 are to perform Data Guard switchover, perform binary installation in the new standby database home, and perform SQL installation against the new primary database. Option 2 has brief impact to the primary database during Data Guard switchover, but there is no impact to the primary while completing patch installation.

The main steps in Option 2 are the following:

- Perform a Data Guard switchover so that the new primary (that is, old standby) is now running on the patched binaries.
- Perform binary installation on the new standby (that is, old primary).
- Perform SQL installation on the new primary (that is, old standby). Changes made to the new primary database during SQL installation will propagate to the new standby via redo.
- Optionally, perform a switchover to get back to the original configuration.

Benefits

Data Guard Standby-First Patch Apply has the following advantages:

- Ability to apply software changes to the physical standby database for recovery, backup or query validation before role transition, or before application on the primary database
- Ability to switch over to the targeted database after completing validation with reduced risk and minimum down time
- Ability to switch back, also known as fall back, if there are stability or performance regressions



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Data Guard Standby-First Patch Apply has the following advantages:

- Ability to apply software changes to the physical standby database for recovery, backup or query validation before role transition, or before application on the primary database. This mitigates risk and potential down time on the primary database.
- Ability to switch over to the targeted database after completing validation with reduced risk and minimum down time
- Ability to switch back, also known as fallback, if there are stability or performance regressions

Summary

In this lesson, you should have learned how to patch in your Data Guard configuration by using the Standby-First Patch Apply method.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 10: Overview

This practice covers the following topics:

- Patching the RAC Database Home in a Data Guard Configuration by Using the Standby-First Rolling Patch Apply Method
- Performing Switchover
- Patching the RAC Database Homes in the New Standby Hosts
- Performing Switchover to the Original Configuration



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Disassociating a Snapshot Standby Database from a Data Guard Configuration

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Create a snapshot standby database to meet the requirement for a temporary, updatable snapshot of a physical standby database
- Describe a method to disassociate a physical standby database from a Data Guard environment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Snapshot Standby Databases: Overview

- A snapshot standby database is a fully updatable standby database created by converting a physical standby database.
- **It is most useful when you are already using Data Guard and you want to perform application testing, development, and reporting.**
- Snapshot standby databases receive and archive—but do not apply—redo data from a primary database.
- When the physical standby database is converted, an implicit guaranteed restore point is created and Flashback Database is enabled.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A snapshot standby database is a fully updatable standby database that is created by converting a physical standby database into a snapshot standby database. A snapshot standby database receives and archives—but does not apply—redo data from a primary database. Redo data received from the primary database is applied when a snapshot standby database is converted back into a physical standby database, after discarding all local updates to the snapshot standby database.

You can create a snapshot standby database by using DGMGRL commands or SQL commands.

When the standby database is converted into a snapshot standby database, an implicit guaranteed restore point is created and Flashback Database is enabled. After performing operations on the snapshot standby database, you can convert it back into a physical standby database.

Data Guard implicitly flashes the database back to the guaranteed restore point and automatically applies the primary database redo that was archived by the snapshot standby database since it was created. The guaranteed restore point is dropped after this process is completed.

Converting a Physical Standby Database into a Snapshot Standby Database

Convert a physical standby database into a snapshot standby database:

```
DGMGRL> CONVERT DATABASE westdb TO SNAPSHOT STANDBY;  
Converting database "westdb" to a Snapshot Standby  
database, please wait...  
Database "westdb" converted successfully
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

You can use the `CONVERT DATABASE` command in DGMGRL to convert a physical standby database into a snapshot standby database as shown in the slide.

Alternatively, you can use the following procedure to perform the conversion without the aid of the Data Guard broker:

1. Stop Redo Apply if it is active.
2. On an Oracle Real Applications Cluster (RAC) standby database, shut down all but one instance.
3. Ensure that the standby database is mounted, but not open.
4. Ensure that a fast recovery area has been configured.
5. Issue the following SQL statement to perform the conversion:

```
ALTER DATABASE CONVERT TO SNAPSHOT STANDBY;
```

Note: A snapshot standby database must be opened at least once in read/write mode before it can be converted back into a physical standby database.

View Snapshot Standby Database Information

View snapshot standby information by using the SHOW CONFIGURATION and SHOW CONFIGURATION VERBOSE commands:

```
DGMGRL> show configuration
Configuration
Name: DGConfig
Enabled: YES
Protection Mode: MaxPerformance
Databases:
  eastdb - Primary database
  westdb - Snapshot standby database

Fast-Start Failover: DISABLED

Current status for "DGConfig":
SUCCESS
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using DGMGRL to View Snapshot Standby Database Information

Use the SHOW CONFIGURATION and SHOW CONFIGURATION VERBOSE commands in DGMGRL to display information about the snapshot standby database.

Snapshot Standby Space Requirements

Monitor the space requirements for enabling snapshot standby:

```
SQL> select file_type, number_of_files, percent_space_used
  from v$recovery_area_usage;
```

FILE_TYPE	NUMBER_OF_FILES	PERCENT_SPACE_USED
CONTROL FILE	0	0
REDO LOG	0	0
ARCHIVED LOG	106	41.81
BACKUP PIECE	1	.17
IMAGE COPY	0	0
FLASHBACK LOG	2	.98
FOREIGN ARCHIVED LOG	0	0
AUXILIARY DATAFILE COPY	0	0



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Flashback Database does not have to be enabled to issue the command to convert a physical standby database into a snapshot database; however, the flash recovery area must be configured. The command will automatically enable Flashback Database and create a guaranteed restore point. This will result in flashback logs being created in the flash recovery area. If there is not enough space in the flash recovery area to create the flashback logs, then an error will occur and an error message will be written to the alert log. The longer the period of time that the standby database is in snapshot mode, the larger the flashback logs will become. It will be necessary to monitor this space usage with the SQL statement presented in the slide. The query in the slide monitors overall space usage in the flash recovery area. To determine the specific space requirements for the guaranteed restore point that was created, the following query can be used:

```
SQL> SELECT NAME, STORAGE_SIZE FROM V$RESTORE_POINT;
NAME                                     STORAGE_SIZE
-----
SNAPSHOT_STANDBY_REQUIRED_08/16/2013 16:41:11      52428800
```

Considerations for Lengthy Testing

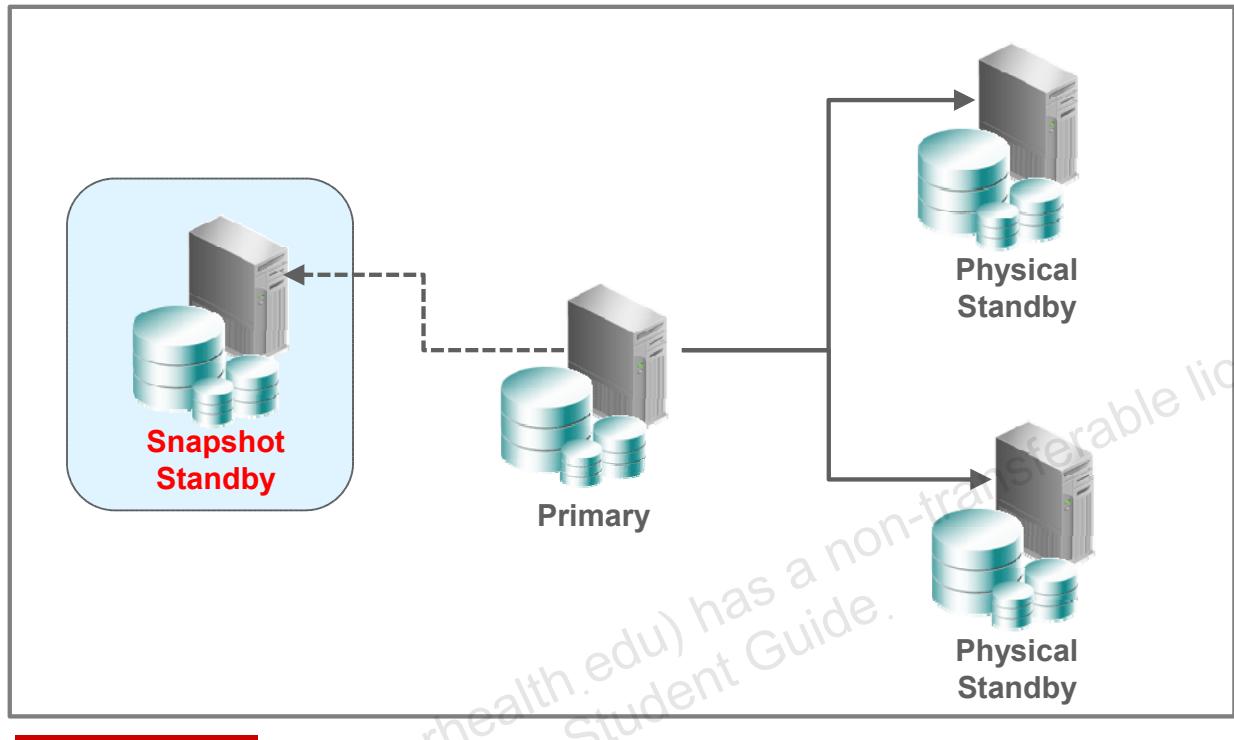
When activating a snapshot standby database, keep the following in mind:

- To support a lengthy application development testing, ensure that there is adequate space to hold archived logs and flashback logs.
- When space limit is reached, you would need to convert the snapshot standby back into the physical standby. **All updates that have been made will be wiped out even if you want to keep them for further development or testing.**



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Disassociating a Snapshot Standby from a Data Guard Configuration



ORACLE®

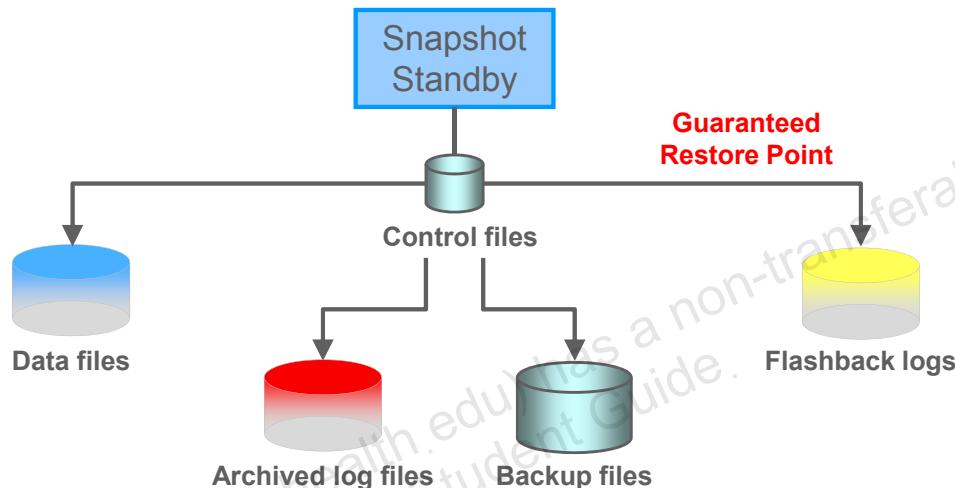
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Under some circumstances, you would need to disconnect a snapshot standby database from a Data Guard configuration as a new independent database while the other primary and standby databases are functioning properly. For example, if there is space limitation due to a lengthy testing, but further testing or development in the same environment is required. The problem could be simply resolved by adding additional space. Alternatively, you could disassociate the snapshot standby database from the Data Guard environment. The remaining part of this lesson introduces the latter.

Example: Initial Environment

Assumptions:

- DB_NAME: EASTDB
- DB_UNIQUE_NAME: WESTDB
- Database Role: Snapshot Standby



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates the initial environment to be used throughout this lesson.

Steps for Disassociating a Snapshot Standby from a Data Guard Configuration

- Step 1: View the snapshot standby database information.
- Step 2: Drop the guaranteed restore point.
- Step 3: Remove the snapshot standby database from the Data Guard Broker configuration.
- Step 4: Use the DBNEWID (nid) utility.
- Step 5: Modify the initialization parameter file.
- Step 6: Create the password file and change the global database name.
- Step 7: Open the database with RESETLOGS.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Step 1: View Snapshot Standby Information

- View the snapshot standby database information by querying V\$DATABASE:

```
SQL> SELECT database_role, controlfile_type FROM v$database;

DATABASE_ROLE      CONTROLFILE_TYPE
-----  -----
SNAPSHOT STANDBY    CURRENT
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The DATABASE_ROLE and CONTROLFILE_TYPE columns of the V\$DATABASE view indicate that the database is a snapshot standby database and disconnected from Data Guard temporarily.

Step 2: Drop a Guaranteed Restore Point

- Obtain the guaranteed restore point:

```
SQL> SELECT database_role, controlfile_type FROM v$database;  
  
NAME                           GUA  
-----  
SNAPSHOT_STANDBY_REQUIRED_07/29/2015 17:13:31    YES
```

- Drop the guaranteed restore point:

```
SQL> DROP RESTORE POINT  
"SNAPSHOT_STANDBY_REQUIRED_07/29/2015 17:13:31";  
*  
ERROR at line 1:  
ORA-38799: Cannot drop guaranteed restore point internally  
created for snapshot standby
```

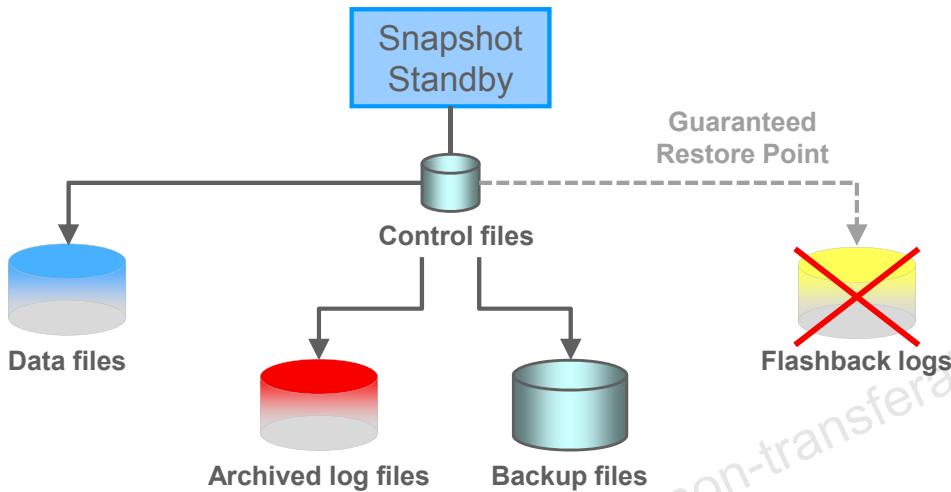


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When the physical standby database is converted, an implicit guaranteed restore point is created and Flashback Database is enabled. This guaranteed restore point must be dropped to disassociate the snapshot standby database from the Data Guard configuration later on.

Note: Depending on your database release, dropping the guaranteed restore point can be done implicitly or explicitly. In your practice environment, this step will be done implicitly.

Example: Result of Step 2



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Dropped the Guaranteed Restore Point from the control file

Step 3: Remove Snapshot Standby

- Remove the snapshot standby database from the Data Guard broker configuration:

```
DGMGRL> REMOVE DATABASE WESTDB;
```

- Verify the Data Guard broker configuration:

```
DGMGRL> SHOW CONFIGURATION;
Configuration Name: DRSolution
Enabled: YES
Protection Mode: MaxPerformance
Databases: EASTDB - Primary database

Fast-Start Failover: DISABLED

Current status for "DRSolution":
SUCCESS
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Use the REMOVE DATABASE command to delete the snapshot database from the broker configuration.

Step 4: Use the DBNEWID Utility

- In the case of an Oracle Real Application Clusters database, the database must be mounted in NOPARALLEL mode:

```
SQL> alter system set cluster_database=false scope=spfile;
```

- Ensure that the snapshot standby database is mounted:

```
$ srvctl stop database -d westdb  
$ srvctl start instance -d westdb -i westdb1 -o mount
```

- Change the DBID and DBNAME using the DBNEWID utility:

```
$ nid target=sys dbname=westdb
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

DBNEWID is a database utility that can change the internal database identifier (DBID) and the database name (DBNAME) for an operational database instead of re-creating a control file. Ensure that the snapshot standby database was shut down consistently and mounting. At this time, you can use the DBNEWID (nid) utility to change the DBID and DBNAME.

Example: Output of the DBNEWID Utility

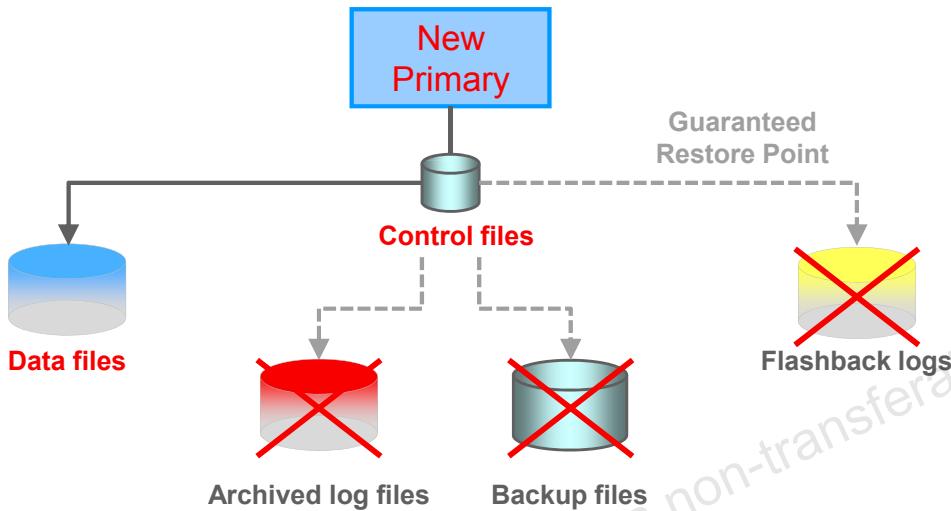
```
$ nid target=sys dbname=westdb
Connected to database EASTDB (DBID=86997811)
...
Changing database ID from 86997811 to 1250654267
Changing database name from EASTDB to WESTDB
Database name changed to WESTDB
Modify parameter file and generate a new password file before restarting.
Database ID for database WESTDB changed to 1250654267.
All previous backups and archived redo logs for this database are
unusable.
Database has been shutdown, open database with RESETLOGS option.
Successfully changed database name and ID.
DBNEWID - Completed successfully.
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The DBNEWID utility performs validations in the headers of the data files and control files before attempting I/O to the files. If validation is successful, DBNEWID prompts you to confirm the operation (unless you specify a log file, in which case, it does not prompt), changes the DBID (and the DBNAME, if specified, as in this example) for each data file, including offline normal and read-only data files, shuts down the database, and then exits. The slide shows an example of what the output for this would look like.

Example: Result of Step 4



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Changed the DBID and DBNAME
- All previous backups and archived redo logs for this database are unusable.

Step 5: Modify the Initialization Parameter File

- Create pfile from spfile:

```
SQL> create pfile from spfile;
```

- Modify the initialization parameter file:

```
$ vi initwestdb.ora
```

- Create spfile from pfile:

```
SQL> create spfile from pfile;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Because DBID and DBNAME have been changed, the initialization parameters must be modified to support the new database.

Step 6: Create the Password File and Change the Global Database Name

- Create the password file on the new database hosts:

```
$ orapwd file=$ORACLE_HOME/dbs/orapwwestdb password=oracle
```

- Change the global database name:

```
SQL> alter database rename global_name to westdb.example.com;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The DBNEWID utility does not create the password file for the new database. If you are dealing with a database in a distributed database system, then each database should have a unique global database name. The DBNEWID utility does not change global database names.

Step 7: Open the Database with RESETLOGS

- Open the database with the RESETLOGS option:

```
SQL> startup mount  
SQL> alter database open resetlogs;
```

- Verify the database role:

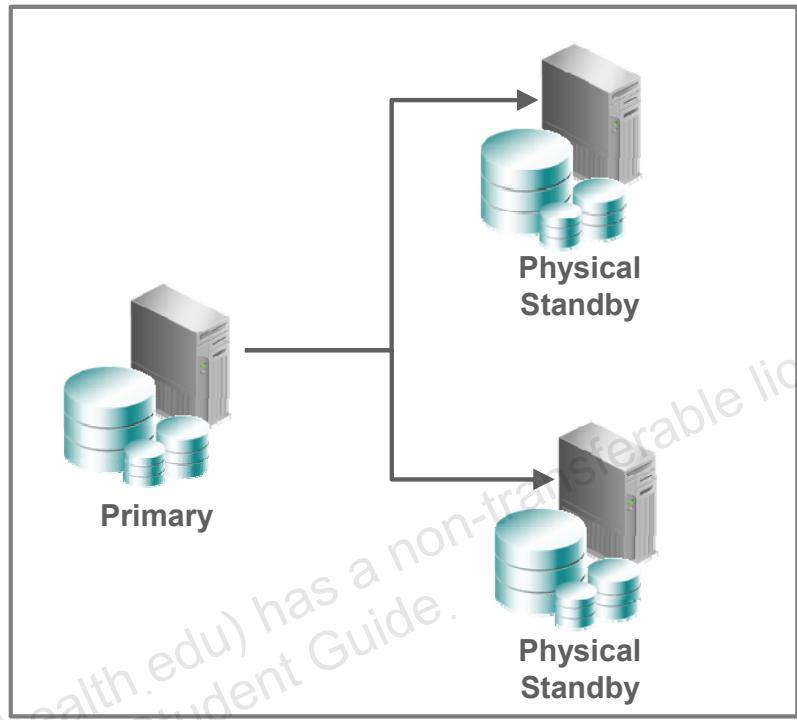
```
SQL> SELECT database_role, controlfile_type FROM v$database;  
  
DATABASE_ROLE      CONTROLFILE_TYPE  
-----  
-----  
PRIMARY           CURRENT
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Finally, open the new database with the RESETLOGS option and verify the database role.

Example: Result of Step 7



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The snapshot standby database was converted to the new independent primary database.

Summary

In this lesson, you should have learned how to:

- Create a snapshot standby database to meet the requirement for a temporary, updatable snapshot of a physical standby database
- Describe a method to disassociate a physical standby database from a Data Guard environment



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 11: Overview

This practice covers the following topic:

- Converting a Snapshot Standby Database into a New Independent Database by Using the DBNEWID (NID) Utility (for testing only)
- Reconfiguring the Environment for Rolling Upgrade



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Rolling Database Upgrade Using Transient Logical Standby

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

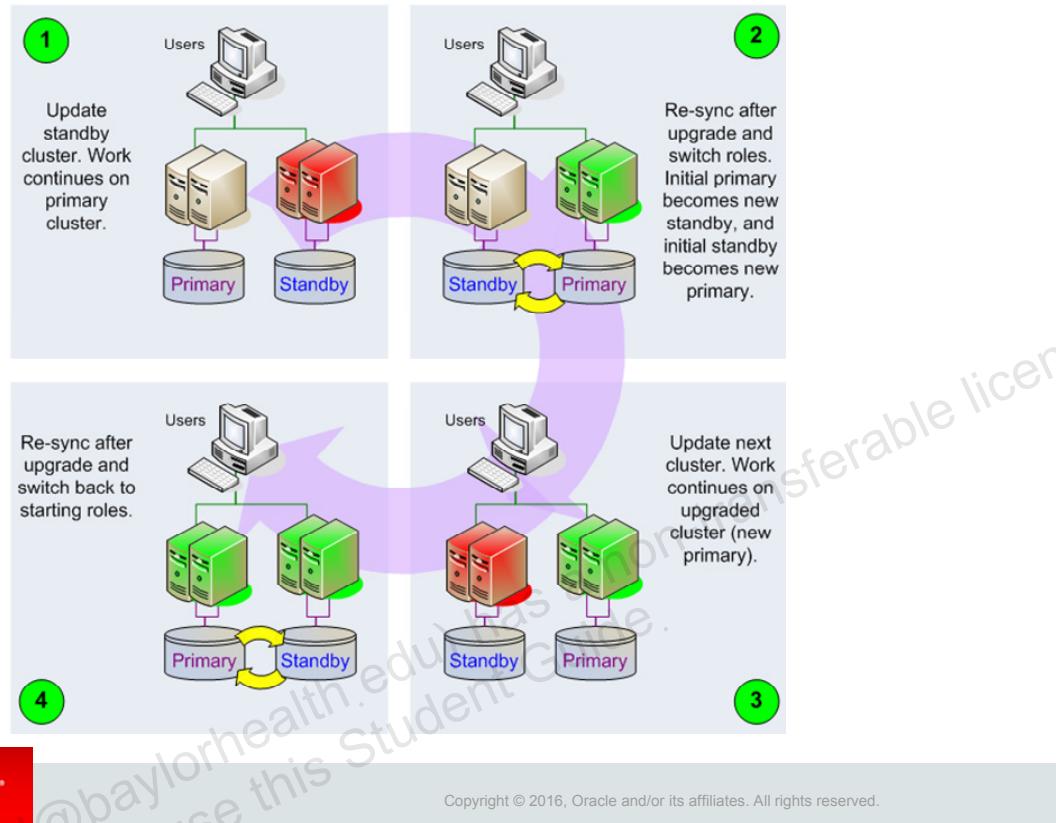
- Describe the procedure for database rolling upgrade using transient logical standby
- Perform a database rolling upgrade using transient logical standby
- Describe the simplified methods for database rolling upgrade using transient logical standby



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Rolling Database Upgrade with Logical Standby Database



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates the concept of rolling upgrade using an Oracle Data Guard logical standby database with SQL Apply. The diagram shows an example with two 2-node clusters; however, the concept applies to single-instance databases and Oracle RAC databases with any number of instances.

With this configuration, the rolling database upgrade procedure is as follows:

1. Work continues on the primary database while the standby database is upgraded using the normal (nonrolling) upgrade procedure.
2. After the standby database is upgraded, it is started and synchronized with the primary database. Then the primary and standby databases switch roles and the workload moves over to the upgraded database, which is now the new primary database.
3. Next, the new standby (original primary) database is upgraded using the normal upgrade procedure.
4. Finally, the newly updated standby database is started and resynchronized with the primary database. After synchronization, the databases can optionally be switched back to their original roles.

Throughout this process, the primary database is available at all times except for the brief periods when a role switch is happening.

Optionally, you can use an archive log repository or a bystander database so that redo is still saved away during each database upgrade. This provides additional protection in cases where the primary database suffers a failure while the standby database is being upgraded.

Phases for Rolling Database Upgrade with Transient Logical Standby Database

- Phase 1: Completing Prerequisites
- Phase 2: Preparing for Upgrade
- Phase 3: Performing Pre-Upgrade Tasks
- Phase 4: Upgrading the Transient Logical Standby
- Phase 5: Performing Post-Upgrade Tasks
- Phase 6: Preparing the Original Primary Database for Upgrade
- Phase 7: Performing the Final Upgrade Tasks



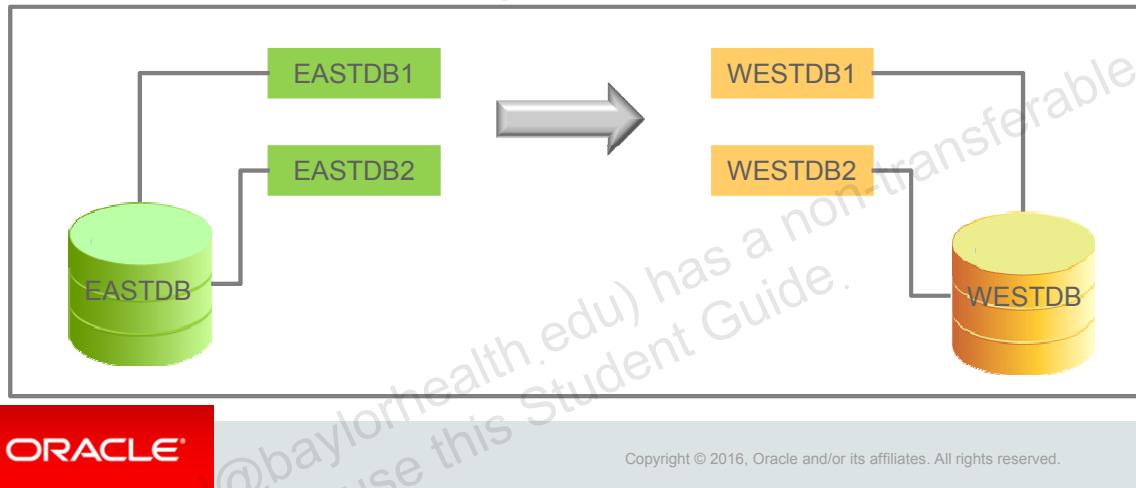
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This slide shows the phases for rolling database upgrade with a transient logical standby database. To maximize database availability during the upgrade process, many operations and role changes are required in a precise order. These operations can be simplified and automated using a script-based utility, which is available for download from My Oracle Support.

Example: Initial Environment

The following environment is assumed:

- Primary Database (EASTDB): 2-node RAC
- Physical Standby Database (WESTDB): 2-node RAC
- Grid Infrastructure SW: 11.2.0.4
- Database SW: 11.2.0.4
- Data Guard Broker Configuration Created



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The environment shown in the slide is assumed to discuss a rolling database upgrade using a transient logical standby database in this lesson.

Phases for Rolling Database Upgrade with Transient Logical Standby Database

- Phase 1: Completing Prerequisites
- Phase 2: Preparing for Upgrade
- Phase 3: Performing Pre-Upgrade Tasks
- Phase 4: Upgrading the Transient Logical Standby
- Phase 5: Performing Post-Upgrade Tasks
- Phase 6: Preparing the Original Primary Database for Upgrade
- Phase 7: Performing the Final Upgrade Tasks



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Phase 1: Completing Prerequisites

- Upgrade Grid Infrastructure first for primary/standby sites.
- Enable Flashback Database for each database.
- Disable Data Guard Broker:

```
DGMGRL> DISABLE CONFIGURATION;  
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

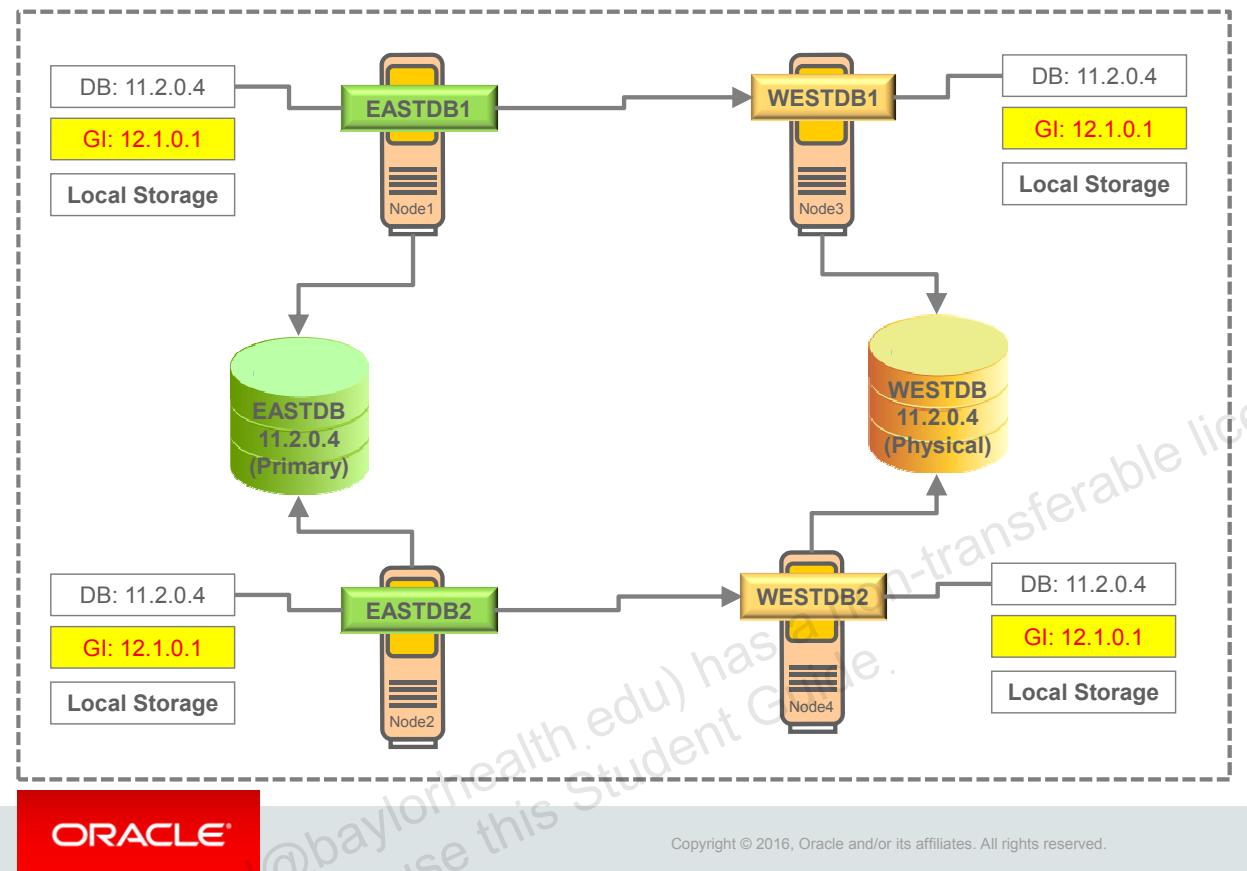
1. In any installation, the Oracle Database software version cannot be newer than the Oracle Grid Infrastructure software that supports it. Because of this, the Oracle Grid Infrastructure software must be upgraded before the database upgrade can be performed.
2. Ensure that the Flashback Database feature is enabled in the primary and standby databases.
3. Disable broker management of the databases in the Data Guard configuration by executing the following DGMGRL command:

```
DGMGRL> DISABLE CONFIGURATION;
```

Execute the following SQL*Plus statement to stop the broker:

```
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE;
```

Example: Result of Phase 1

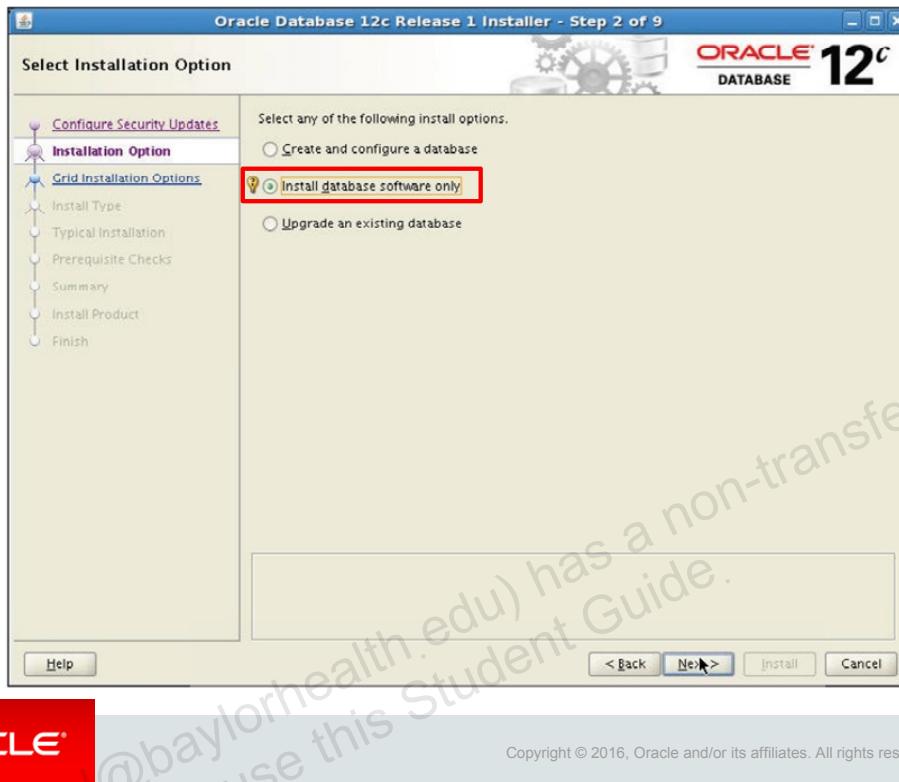


The diagram in the slide illustrates the result of tasks in phase 1.

- Upgraded Grid Infrastructure software version (12.1.0.1) in the primary and standby sites
- Enabled the Flashback Database feature in the primary and standby databases
- Disabled the Data Guard broker configuration

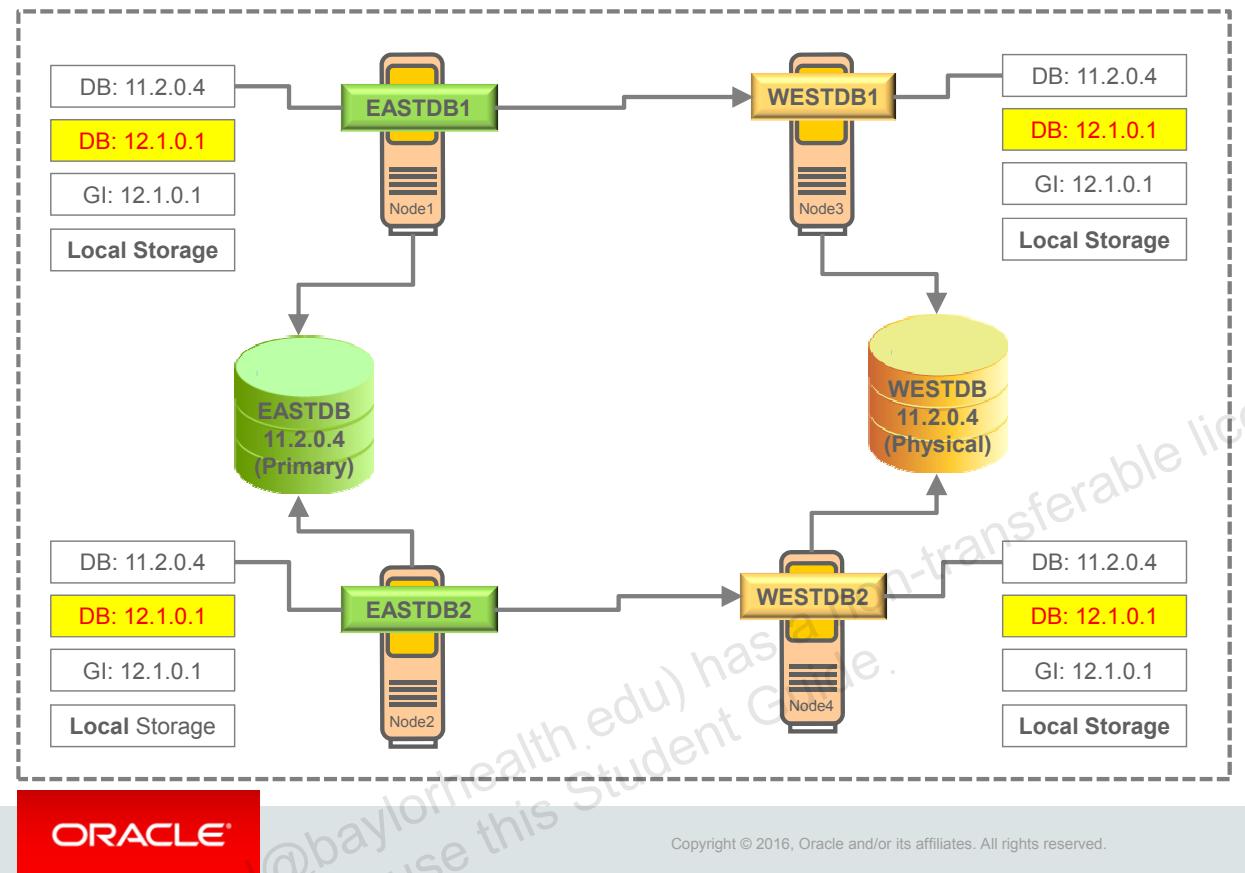
Phase 2: Preparing for Upgrade

Out-of-place software upgrade on the primary and standby hosts



As an option, you can install the software for the new Oracle Database release (12.1.0.1) before you perform the upgrade of Oracle Database. The installation procedure installs the Oracle software into a new Oracle home. This is referred to as an out-of-place upgrade. To use the pre-installed software in conjunction with DBUA later on, you must first run the root configuration script (`root . sh`) located in the Oracle Database 12c home directory.

Example: Result of Phase 2



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates the result of tasks in phase 2.

- Pre-installed the database software version (12.1.0.1) in the primary and standby sites

Phase 3: Performing Pre-Upgrade Tasks

- Create a Guaranteed Restore Point:

```
SQL> create restore point pre_upgrade_guarantee flashback  
database;
```

- For RAC, reduce the RAC to a single instance:

```
$ srvctl stop instance -d westdb -i westdb2 -o abort
```

- Disable Oracle RAC:

```
SQL> alter system set cluster_database=false scope=spfile
```

- Restart a single instance (westdb1) of the physical standby database in mount mode:

```
SQL> shutdown abort  
SQL> startup mount
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Create a guaranteed restore point.

Before converting the physical standby database into a transient logical standby database, perform the following tasks:

2. For RAC, reduce the RAC to a single instance.
3. Disable the Oracle RAC.
4. Restart a single instance of the physical standby database in mount mode.

Phase 3: Performing Pre-Upgrade Tasks

- Convert the physical standby database into a transient logical standby database:

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

- Disable automatic deletion of foreign archived logs at the logical standby database:

```
SQL> exec DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE','FALSE');
```

- Start SQL Apply:

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



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Convert the physical standby database into a logical standby database. This is done temporarily only for the duration of the rolling upgrade.

- Create a logical standby database and execute the following command:

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

Note: A logical standby database created with the `KEEP IDENTITY` clause retains the same `DB_NAME` and `DBID` as those of its primary database.

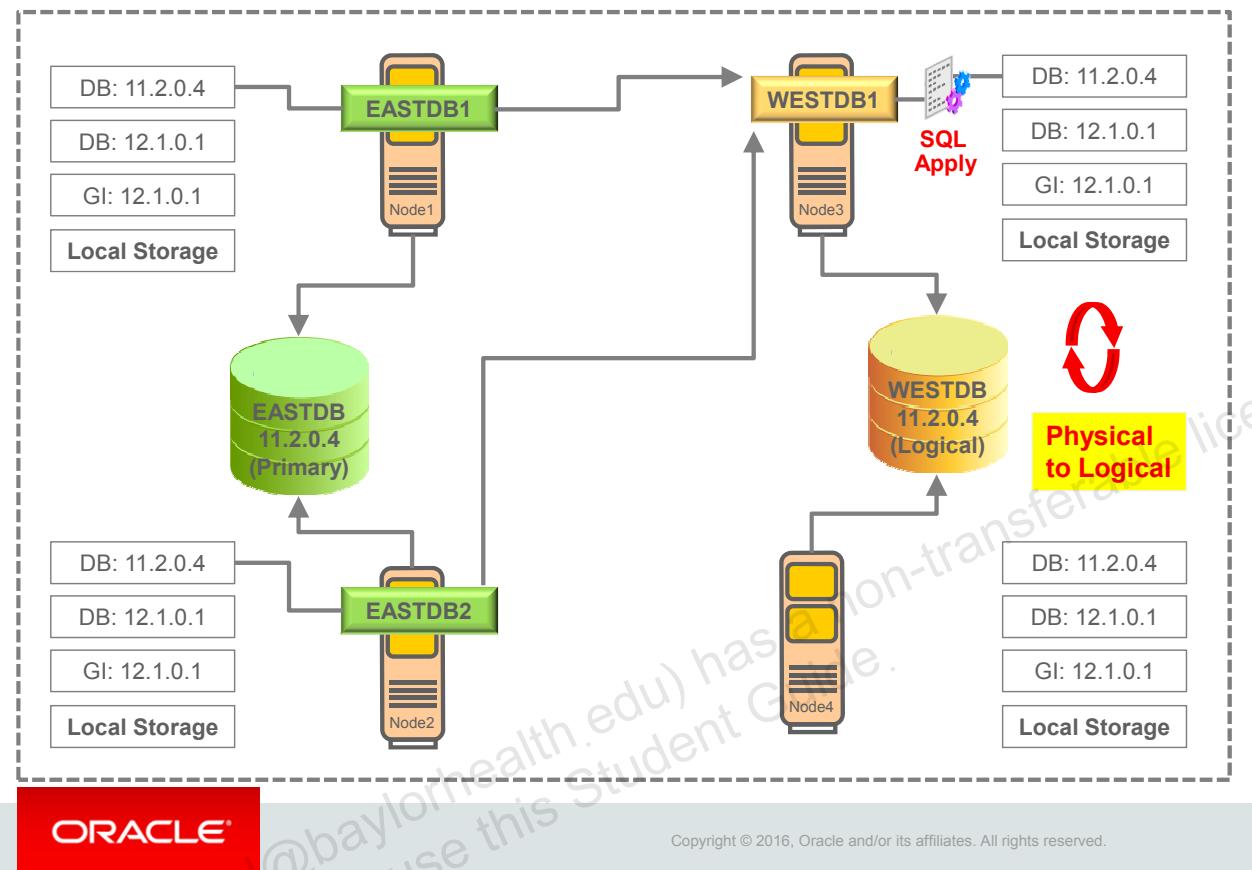
- Disable automatic deletion of foreign archived logs at the logical standby database:

```
execute DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE', 'FALSE');
```

- Start SQL Apply:

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

Example: Result of Phase 3



The diagram in the slide illustrates the result of tasks in phase 3.

- Converted the 2-node RAC physical standby database into the transient logical standby database

Phases for Rolling Database Upgrade with Transient Logical Standby Database

- Phase 1: Completing Prerequisites
- Phase 2: Preparing for Upgrade
- Phase 3: Performing Pre-Upgrade Tasks
- Phase 4: Upgrading the Transient Logical Standby
- Phase 5: Performing Post-Upgrade Tasks
- Phase 6: Preparing the Original Primary Database for Upgrade
- Phase 7: Performing the Final Upgrade Tasks



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Phase 4: Upgrading the Transient Logical Standby

- Stop SQL Apply:

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

- Set the COMPATIBLE initialization parameter.
- Re-enable Oracle RAC:

```
SQL> alter system set cluster_database=true scope=spfile;
```

- Restart all the database instances:

```
SQL> shutdown abort  
SQL> exit  
  
$ srvctl start database -d westdb
```

- Run the root configuration script in the pre-installed Oracle Database 12c home directory.

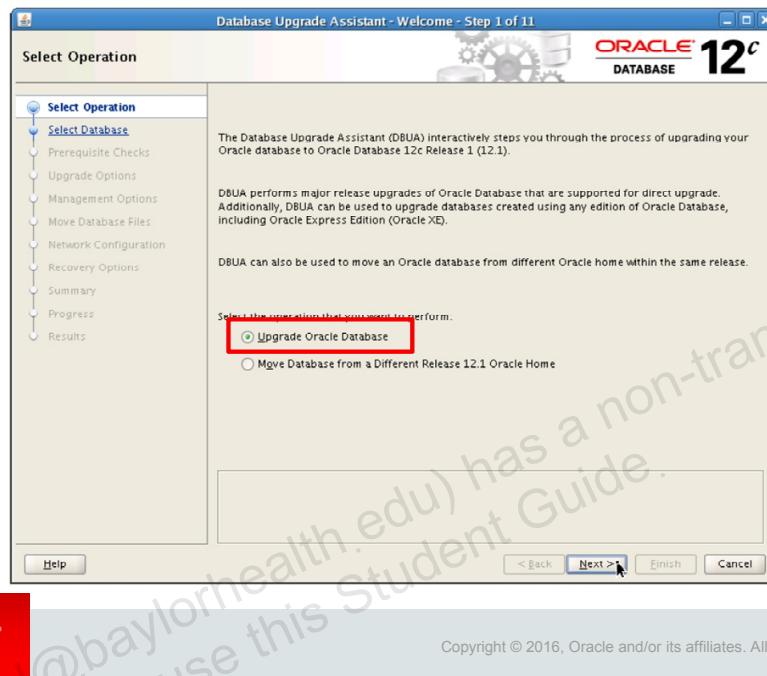


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Prepare for the rolling upgrade as follows:
 - a. Stop SQL Apply by issuing the statement given in the slide on the logical standby database.
 - b. Set the COMPATIBLE initialization parameter to the highest value. Ensure that the COMPATIBLE initialization parameter specifies the release number for the Oracle Database software running on the primary database before the upgrade.
2. When DBUA is used on an Oracle RAC database, it requires all the database instances to be open at the beginning of the DBUA session. Re-enable Oracle RAC and restart all the instances.
3. To use the pre-installed binaries in conjunction with DBUA, you must first run the root configuration script (`root.sh`) located in the Oracle Database 12c home directory.

Phase 4: Upgrading the Transient Logical Standby

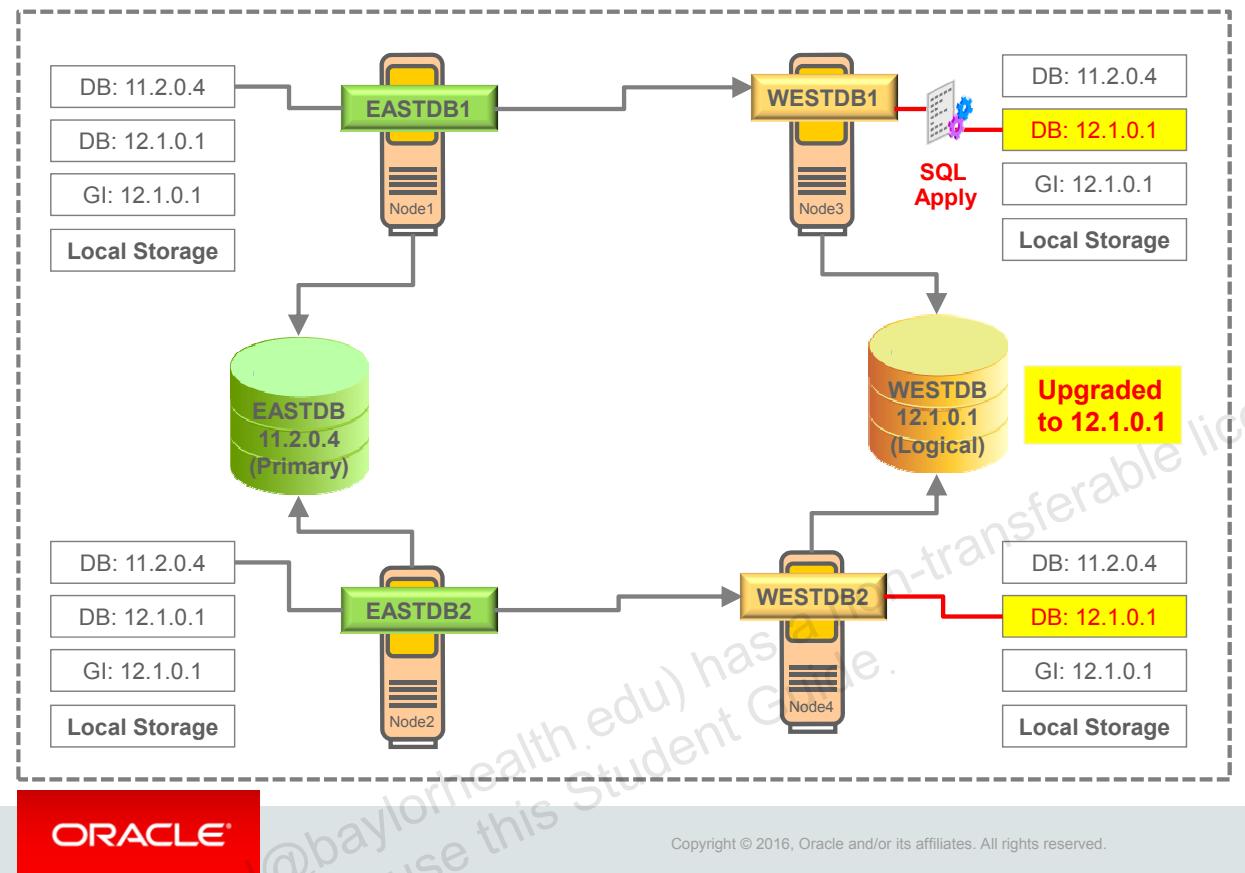
- Upgrade the transient logical standby database manually or by using DBUA.
- Restart SQL Apply on the upgraded logical standby database.



4. Upgrade the logical standby database to 12.1.0.1. While the logical standby database is being upgraded, it does not accept redo data from the primary database. See the *Oracle Database Upgrade Guide* for detailed information.
5. Restart SQL Apply on the upgraded logical standby database:

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

Example: Result of Phase 4



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates the result of tasks in phase 4.

- Upgraded the standby database (WESTDB) to 12.1.0.1

Phases for Rolling Database Upgrade with Transient Logical Standby Database

- Phase 1: Completing Prerequisites
- Phase 2: Preparing for Upgrade
- Phase 3: Performing Pre-Upgrade Tasks
- Phase 4: Upgrading the Transient Logical Standby
- Phase 5: Performing Post-Upgrade Tasks
- Phase 6: Preparing the Original Primary Database for Upgrade
- Phase 7: Performing the Final Upgrade Tasks



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Phase 5: Performing Post-Upgrade Tasks

- Query DBA_LOGSTDBY_EVENTS to monitor the upgraded logical standby database.
- Begin a switchover to the upgraded logical standby database:

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

- Import any tables that were unsupported and modified.
- Complete the switchover and activate user applications:

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



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Query DBA_LOGSTDBY_EVENTS to determine whether there are any DDL and DML statements that were not applied on the logical standby database.
2. Begin a switchover to the upgraded logical standby database by executing the following statement on the primary database:

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

Note: This statement waits for existing transactions to complete. To minimize the time it takes to complete the switchover, users connected to the primary database should log off immediately and reconnect to the standby database.

3. Import any tables that were modified during the upgrade from the primary database that were unsupported in the logical standby database.
4. Complete the switchover by executing the following statement on the logical standby database:

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

Note: After the switchover, you cannot send redo data from the new primary database (using the new Oracle Database software release) to the new standby database (using the older Oracle Database software release).

Phase 5: Performing Post-Upgrade Tasks

- Reduce Oracle RAC to a single instance:

```
$ srvctl stop instance -d eastdb -i eastdb2 -o abort
```

- Flash back the original primary database to the guaranteed restore point:

```
SQL> SHUTDOWN IMMEDIATE  
SQL> STARTUP MOUNT  
SQL> FLASHBACK DATABASE TO RESTORE POINT pre_upgrade;
```

- Convert the original primary database into a physical standby database:

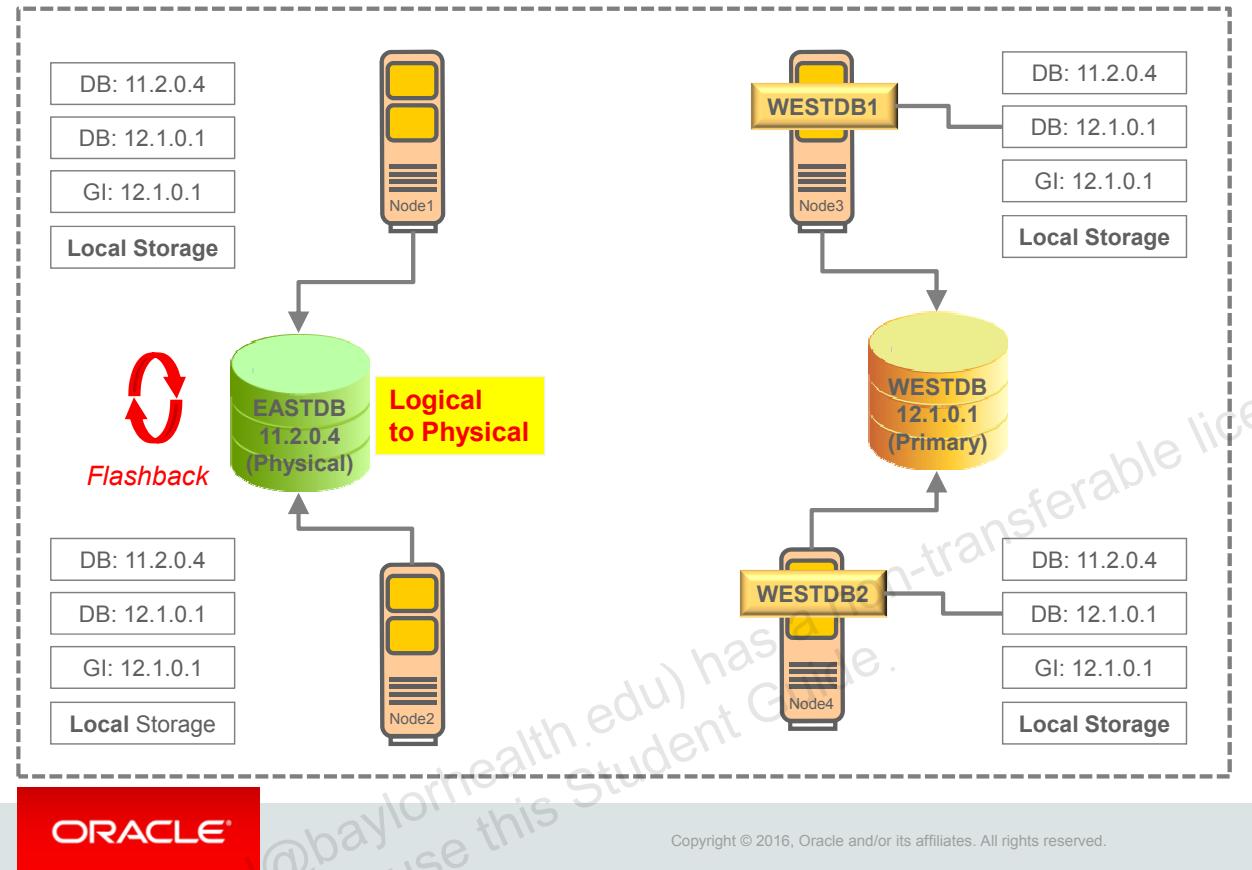
```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;  
SQL> SHUTDOWN IMMEDIATE;
```



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

5. For RAC, reduce Oracle RAC to a single instance before performing flashback on the original primary database.
6. Flashback the original primary database to the guaranteed restore point that you created in step 1.
7. Mount the original primary database using the new version of the software. You will not run the upgrade scripts, because this database will be turned into a physical standby, and will be upgraded automatically as it applies redo data generated from the new primary database.

Example: Result of Phase 5



The diagram in the slide illustrates the result of tasks in phase 5.

- Performed a switchover operation
- Flashed back the original primary database to the restore point
- Converted the original primary database to the physical standby database

Phase 6: Preparing the Original Primary Database for Upgrade

- Remove the clusterware service entry for the original primary database:

```
$ srvctl remove database -d eastdb
```

- Re-create the clusterware service entry:

```
$ srvctl add database -d eastdb -o <NEW_OH>
$ srvctl add instance -d eastdb -i eastdb1 -n enode01
$ srvctl add instance -d eastdb -i eastdb2 -n enode02
```

- Mount the original primary database using **the new version of the software (12.1.0.1)**:

```
$ srvctl start database -d eastdb -o mount
```

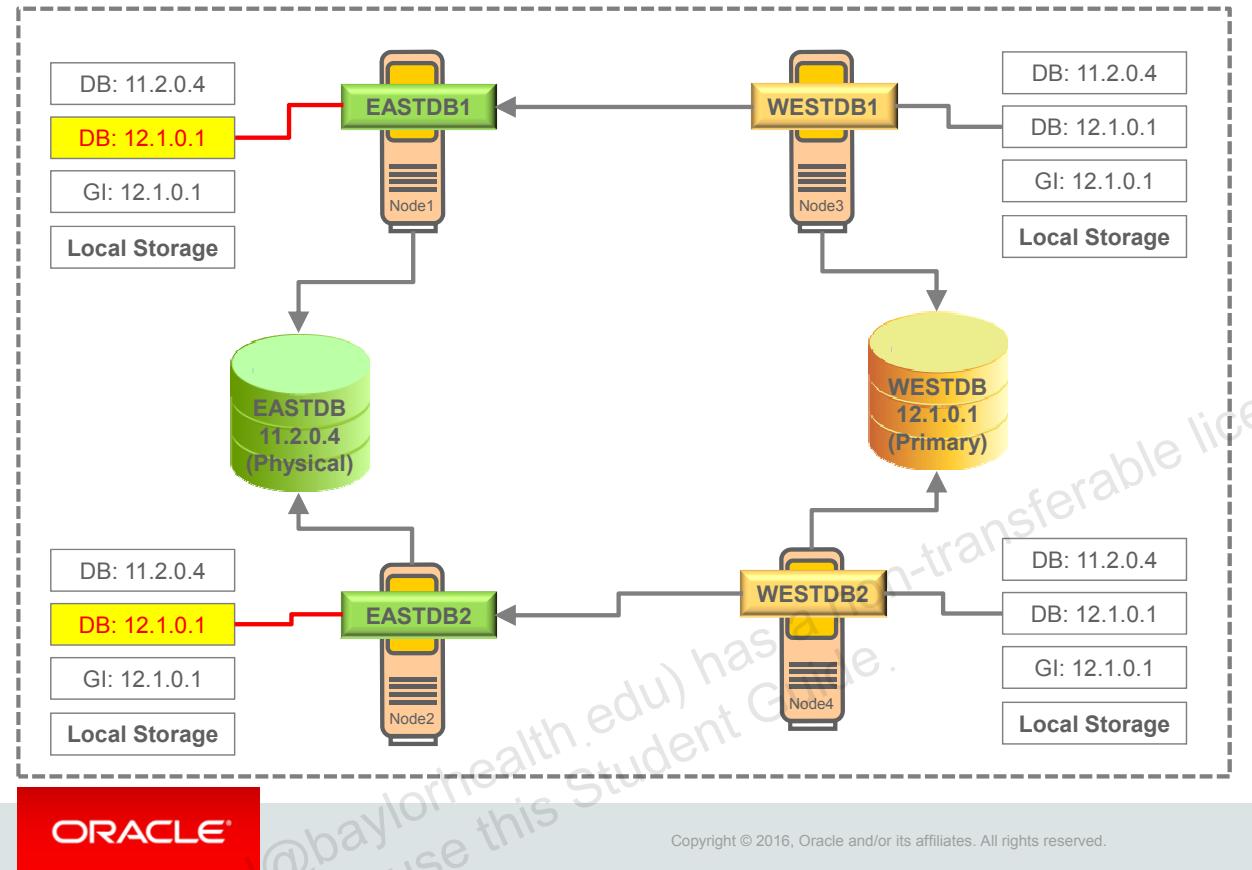


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

1. Remove the clusterware service entry for the EASTDB database.
2. Re-create the clusterware service entry for the EASTDB database and its instances.
3. Mount the original primary database using the new version of the software.

Note: You will not run the upgrade scripts, because this database will be turned into a physical standby, and will be upgraded automatically as it applies redo data generated from the new primary database.

Example: Result of Phase 6



The diagram in the slide illustrates the result of tasks in phase 6.

- Re-registered the clusterware service entry for the EASTDB database based on the new Oracle Database Home
- Mounted the EASTDB instances by using the upgraded 12c software

Phase 7: Performing the Final Upgrade Tasks

- Start managed recovery on the original primary database.

Note: *The original primary database will be upgraded automatically as it applies redo data generated from the new primary database:*

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

- Perform a switchover to return your original primary database to the primary database role. (Optional)
- Clean up the guaranteed restore point:

```
SQL> DROP RESTORE POINT PRE_UPGRADE;
```

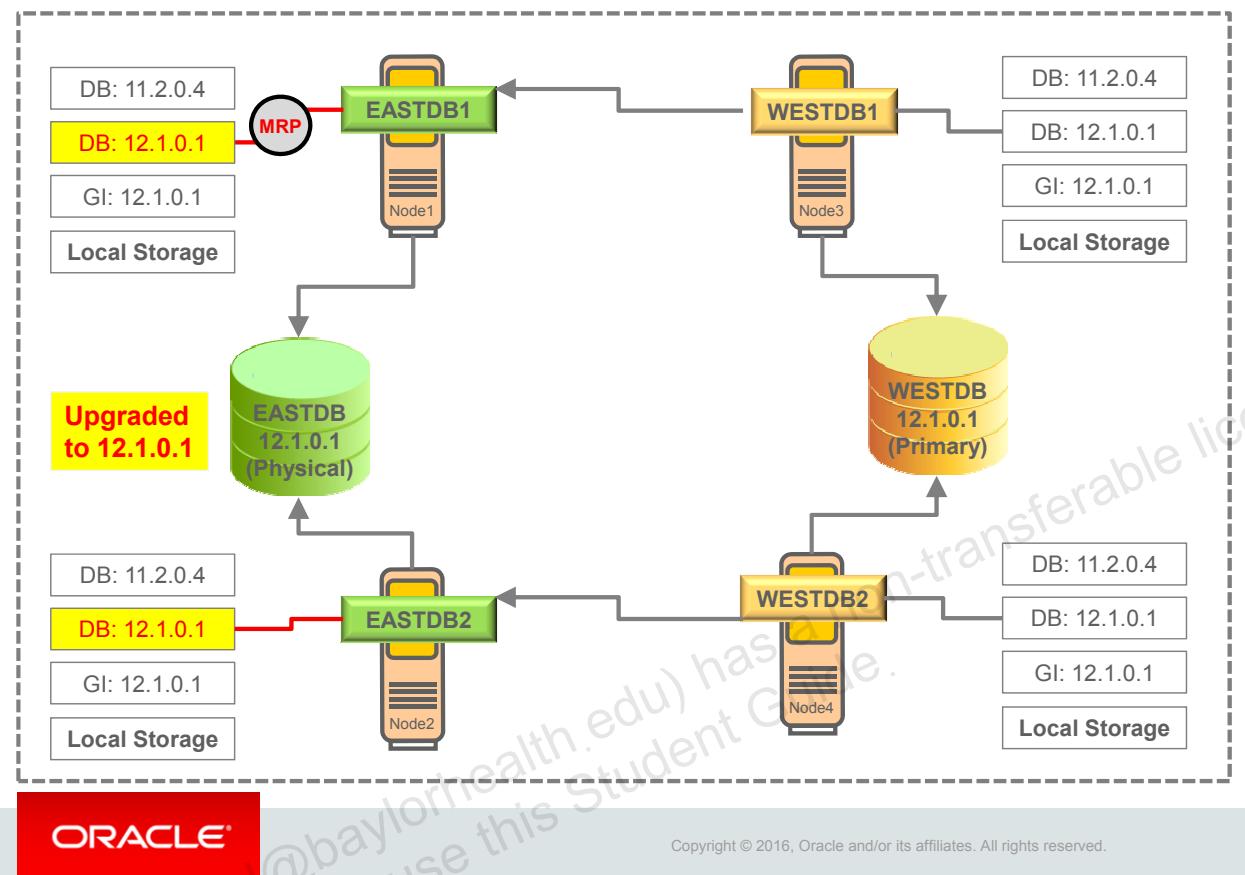
- Start all the Oracle RAC database instances.
- Re-enable the Oracle Data Guard Broker.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Start managed recovery on the original primary database. The original primary database will be upgraded automatically as it applies redo data generated from the new primary database.
- Optionally, perform a switchover to return to your original configuration.
- Clean up the guaranteed restore point.
- Start all the Oracle RAC database instances.
- Re-enable the Oracle Data Guard Broker.

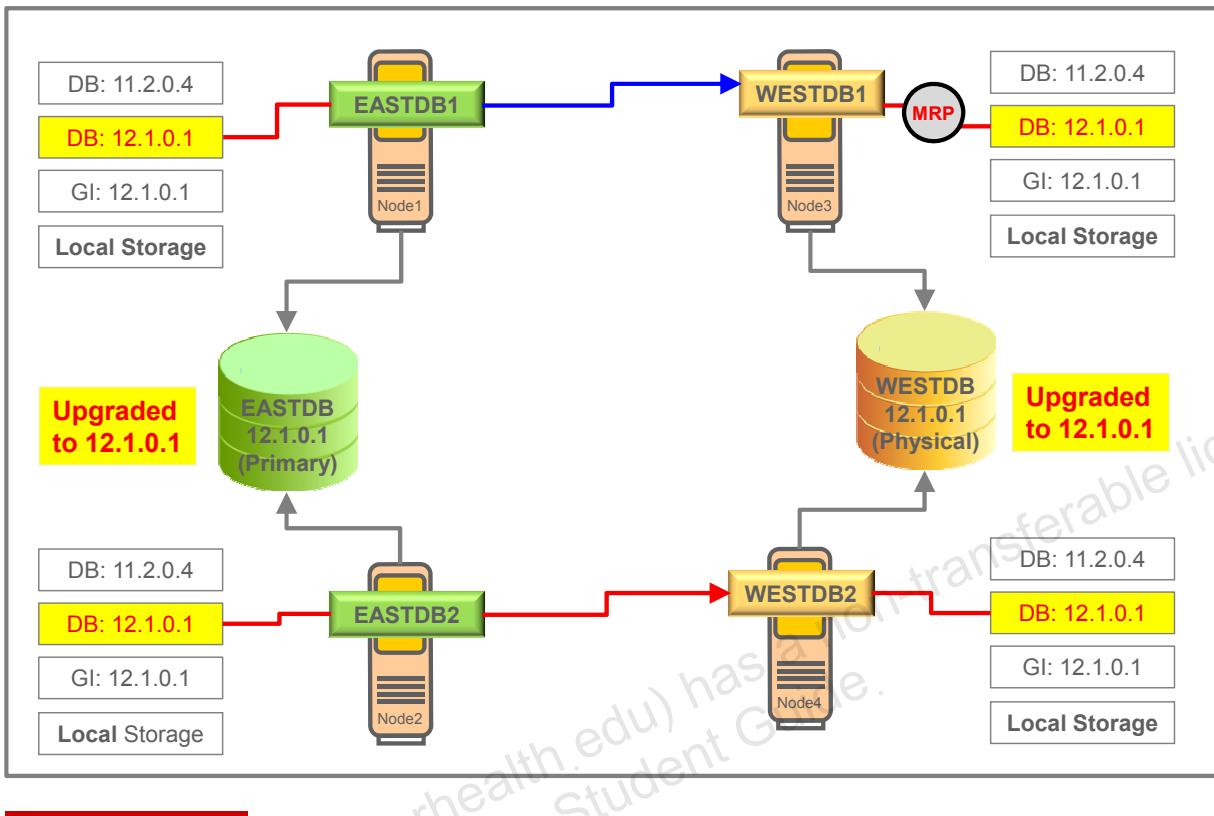
Example: Result of Phase 7 (Before Switchover)



The diagram in the slide illustrates the result of tasks in phase 7.

- Started managed recovery on the original primary database and automatically upgraded to 12.1.0.1

Example: Result of Phase 7 (After Switchover)



The diagram in the slide illustrates the result of tasks in phase 7.

- Performed a switchover to the original Data Guard configuration

Benefits and Challenges

- Benefits:
 - Minimal business disruption
 - In-place physical database upgrade
 - Opportunity to test the upgraded standby before upgrading the primary
- Challenges:
 - Process requires an Oracle Data Guard standby database.
 - Process is long and complex.
 - Unsupported data types



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The slide lists the various benefits and challenges associated with rolling database upgrade by using transient logical standby database.

Benefits

- **Minimal business disruption:** Rolling database upgrade with transient logical standby database requires much less system down time compared to the normal (nonrolling) database upgrade procedure. Depending on the environment, down time can range from seconds to a few minutes. Because each database is upgraded while it is in the standby role, the full processing power of the primary system remains available for business processing. Also, the process can be executed over an extended time period, which might allow you to fit the required down time into existing maintenance periods.
- **In-place physical database upgrade:** Because the upgrade is in-place, there is no need to unload and reload your data, which can be particularly useful for large databases. Also, because the data remains inside the database, your existing security controls are maintained, which is particularly useful for sensitive databases.
- **Opportunity to test the upgraded standby before upgrading the primary:** After the standby database has been upgraded, it can be used to evaluate performance and stability of the new release before there is any impact to production.

Challenges

- **Process requires an Oracle Data Guard standby database:** This might not be an issue for existing customers of Oracle Data Guard. However, customers that do not already use Oracle Data Guard might not be able to justify the investment in a standby environment solely for performing an upgrade.
- **Process is quite long and complex:** The process requires numerous tasks to be performed correctly and in a precise order. This opens up the possibility for administrator error, which could lead to unforeseen costs and delays. These risks can be mitigated by thorough planning and testing, and also by using `physru.sh` to simplify and automate the process.
- **Unsupported data types:** When setting up a logical standby database, you must ensure that the logical standby database can maintain the data types and tables in your primary database. Limits and restrictions associated with Oracle Data Guard logical standby database may result in some data not being replicated on the standby database. You should review Appendix C titled “Data Type and DDL Support on a Logical Standby Database” in *Oracle Data Guard Concepts and Administration* for complete details.

Even if unsupported data types are identified, there are cases when a transient logical standby procedure can still be used. The determination has to be made if there is a satisfactory way to handle the unsupported data types. Options for using rolling upgrade when unsupported data types exist include the following:

- Temporarily suspend or prohibit changes to the unsupported tables for the period of time it takes to perform the upgrade procedure.
- If you cannot prevent changes to unsupported tables during the upgrade, any unsupported transactions that occur are recorded in the `DBA_LOGSTBY_EVENTS` table on the logical standby database. After the upgrade is completed, you could use Oracle Data Pump or the Export/Import utility to copy the changed tables.
- Use Extended Datatype Support (EDS), which enables SQL Apply to replicate changes to tables that contain some data types that are not natively supported. See the chapter titled “Managing a Logical Standby Database” in *Oracle Data Guard Concepts and Administration* for complete details.

Simplified Method 1

- Using the physru.sh script:

```
$ physru.sh <username> <primary_tns> <standby_tns> <primary_name>  
<standby_name> <upgrade_version>
```

- The physru.sh script:
 - Simplifies Phase 3: Performing Pre-Upgrade Tasks
 - Simplifies Phase 5: Performing Post-Upgrade Tasks
 - Simplifies Phase 7: Performing the Final Upgrade Tasks
 - Contains many automatic checks to minimize errors
 - Provides guidance for additional manual tasks
 - Is available through My Oracle Support bulletin 949322.1



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The process for rolling database upgrade with transient logical standby database requires many operations and role changes to be performed in a precise order. To simplify the process and reduce the likelihood of administrator error, the physru.sh script is available through the My Oracle Support bulletin 949322.1. This script automates much of the process and provides guidance regarding the required manual steps. The script also performs many checks along the way to maximize the probability of success.

Simplified Method 2

- Using DBMS_ROLLING with Active Data Guard:
 - Database software upgrades using the DBMS_ROLLING PL/SQL package will be usable starting with the first patchset of Oracle Database 12c to the second patchset.
- This method has the following benefits:
 - Data Guard configuration is aware of the upgrade
 - Early problem detection
 - Centralized, simplified, and uniform execution
 - Dedicated interface: DBMS_ROLLING PL/SQL package
 - Built-in fault tolerance
 - Configuration rollback
 - Centralized monitoring: DBA_ROLLING views



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The Rolling Upgrade Using Active Data Guard feature, new as of Oracle Database 12c Release 1 (12.1), provides a streamlined method of performing rolling upgrades. It is implemented using the new DBMS_ROLLING PL/SQL package, which allows you to upgrade the database software in a Data Guard configuration in a rolling fashion. The Rolling Upgrade Using Active Data Guard feature requires the Oracle Active Data Guard option.

You can use this feature for database version upgrades starting with the first patchset of Oracle Database 12c. See the chapter titled “Using DBMS_ROLLING to Perform a Rolling Upgrade” in *Oracle Data Guard Concepts and Administration* for complete details.

Summary

In this lesson, you should have learned how to:

- Describe the procedure for database rolling upgrade using transient logical standby
- Perform a database rolling upgrade using transient logical standby
- Describe the simplified methods for database rolling upgrade using transient logical standby



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 12: Overview

In this practice, you will perform a Database Rolling Upgrade Using Transient Logical Standby. The initial Oracle Database software version on both clusters is 11.2.0.4, and the target software version for the upgraded environment is 12.1.0.1.

This practice covers the following topics:

- Introduction to the lab environment
- Grid Infrastructure Upgrade
- Database Preparation for Rolling Upgrade
- Database Rolling Upgrade
- Post-upgrade Steps



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.