



# Oracle DataGuard 19c

# Table of Content



1. Introduction to Oracle Data Guard: 4
2. Oracle Net Services in a Data Guard Environment: 28
3. Creating a Physical Standby Database: 45
4. Managing Physical Standby Files: 96
5. Using Oracle Active Data Guard: 123
6. Using Oracle Active Data Guard: 155
7. Creating and Managing a Snapshot Standby Database: 172
8. Creating a Logical Standby Database: 185
9. Oracle Data Guard Broker: 226

# Table of Content



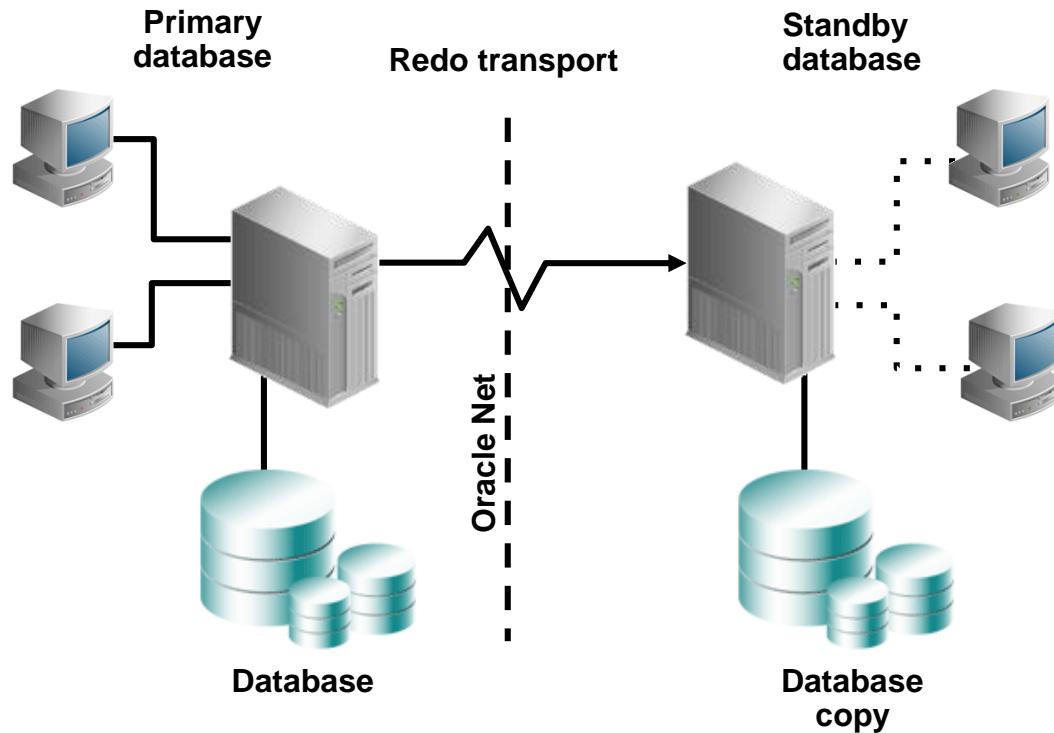
10. Creating a Data Guard Broker Configuration: 247
11. Monitoring a Data Guard Broker Configuration: 296
12. Configuring Data Protection Modes: 335
13. Optimizing and Tuning a Data Guard Configuration: 350
14. Performing Role Transitions: 375
15. Using Flashback Database in a Data Guard Configuration: 408
16. Enabling Fast-Start Failover: 428
17. Backup and Recovery Considerations: 490
18. Enhanced Client Connectivity: 520
19. Patching and Upgrading Databases: 553

# **1. Introduction to Oracle Data Guard**

# Objectives

- After completing this lesson, you should be able to:
  - Describe the basic components of Oracle Data Guard
  - Explain the differences between physical, snapshot, and logical standby databases
  - Explain the benefits of implementing Oracle Data Guard

# What Is Oracle Data Guard?



# 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
  - 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
  - Can be used concurrently for data protection, reporting, and database upgrades

# 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

# Types of Data Guard Services

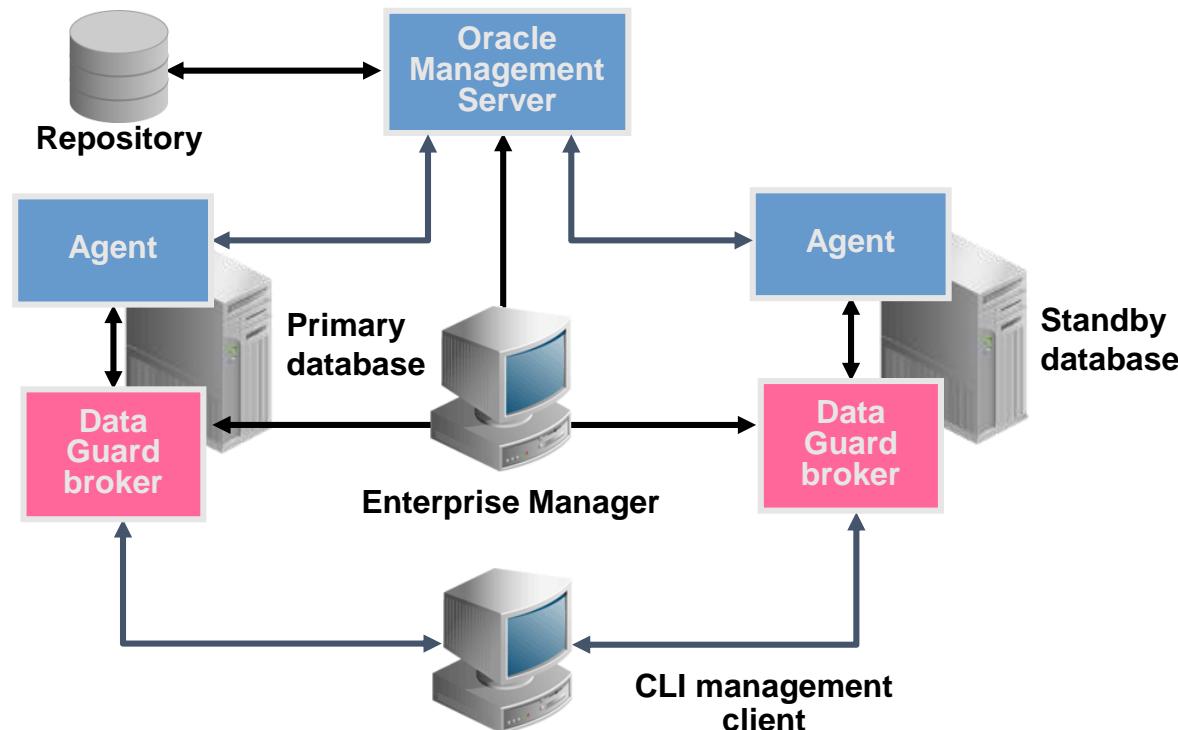
- Data Guard provides three types of services:
  - Redo transport services
  - Apply services
    - Redo Apply
    - SQL Apply
  - Role management services



# Role Transitions: Switchover and Failover

- Oracle Data Guard supports two role-transition operations:
  - Switchover
    - Planned role reversal
    - Used for OS or hardware maintenance/upgrade
  - Failover
    - Unplanned role reversal
    - Emergency use
    - Zero or minimal data loss (depending on choice of data-protection mode)
    - Can be initiated automatically when fast-start failover is enabled

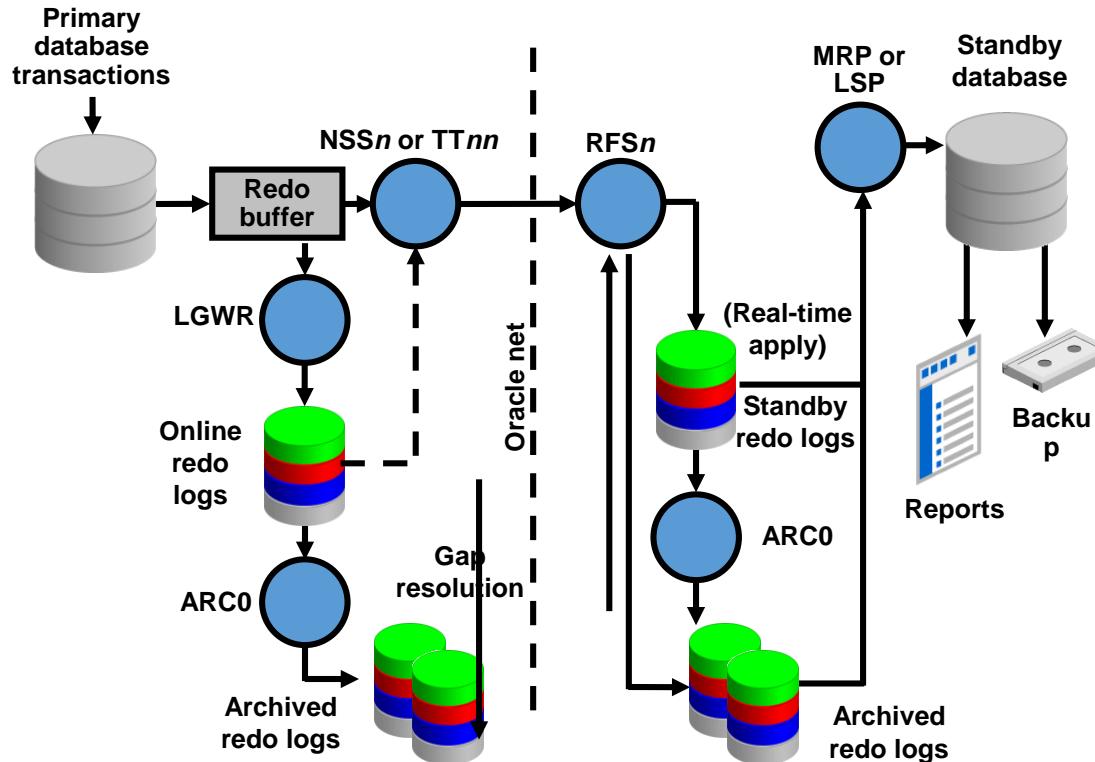
# Oracle Data Guard Broker Framework



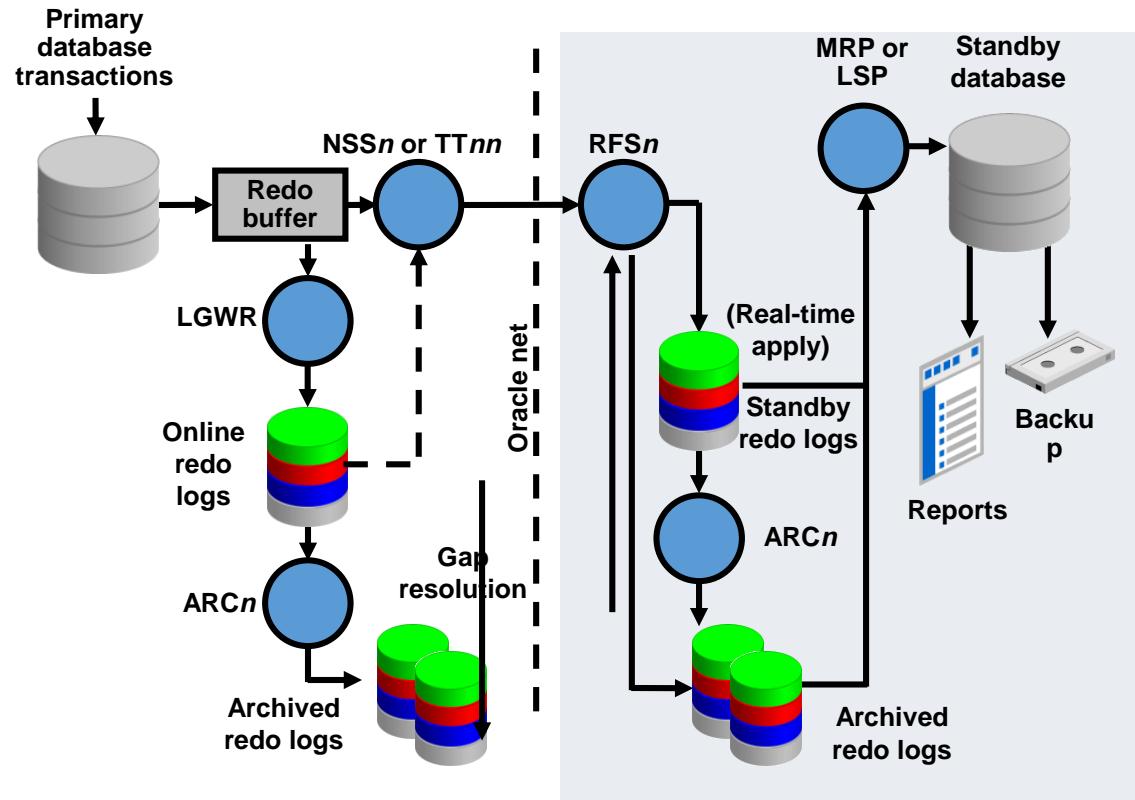
# Choosing an Interface for Administering a Data Guard Configuration

- Data Guard broker configuration:
  - DGMGRL command-line interface
  - Enterprise Manager Cloud Control
  - SQL commands to query data dictionary views
- Non–Data Guard broker configuration:
  - SQL commands

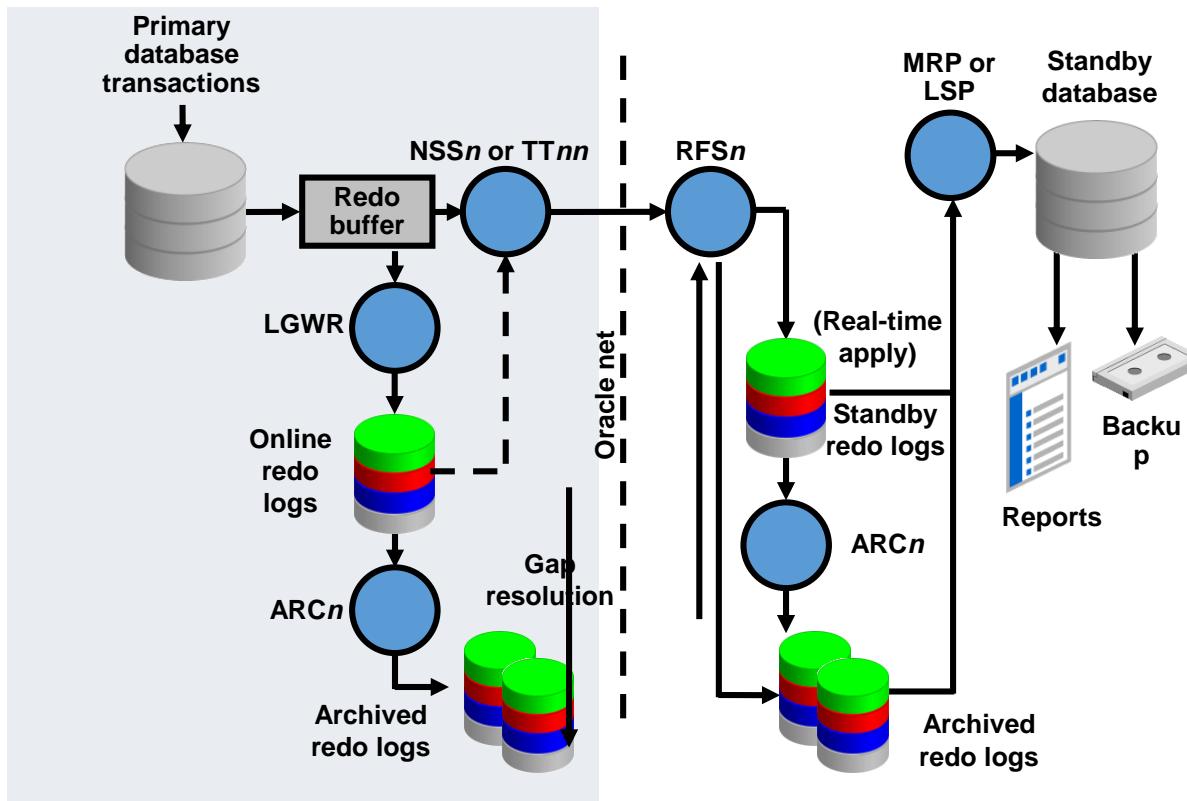
# Oracle Data Guard: Architecture (Overview)



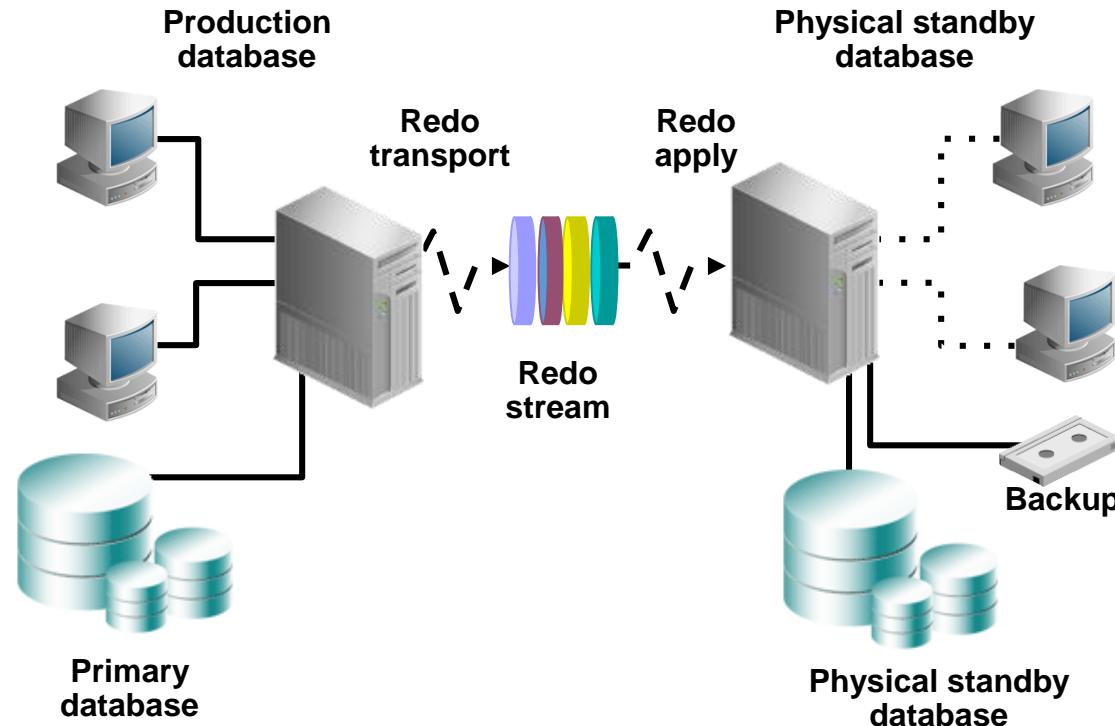
# Primary Database Processes



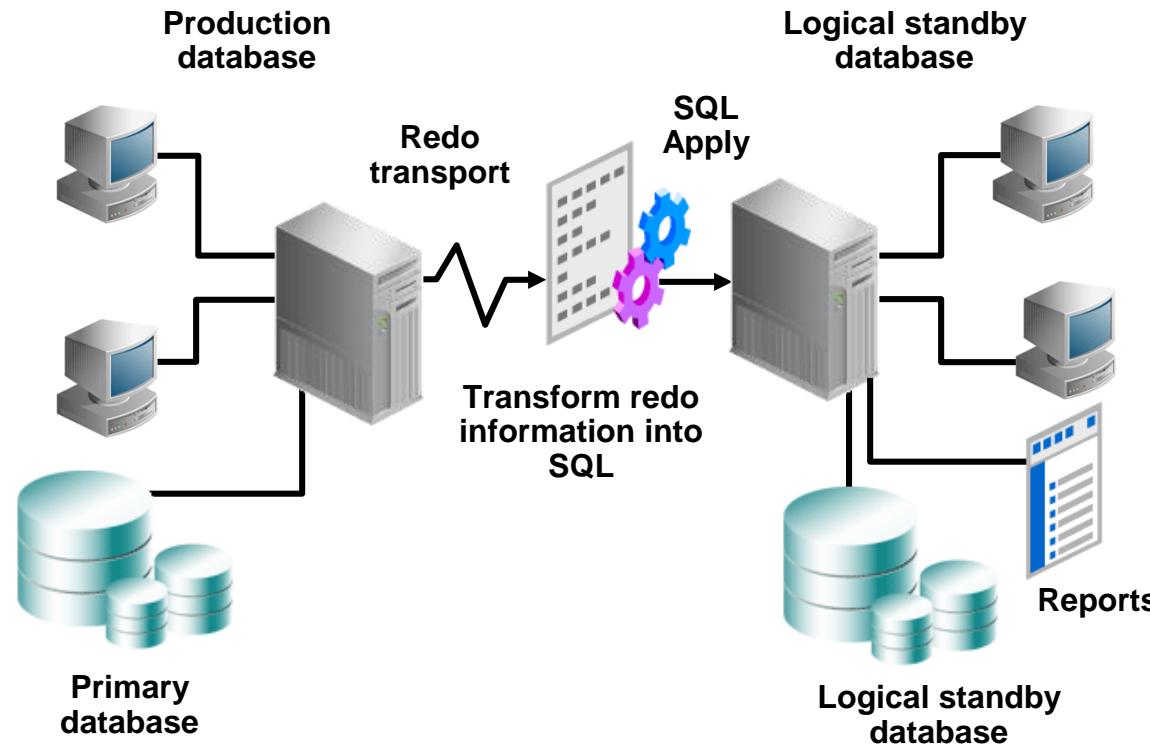
# Standby Database Processes



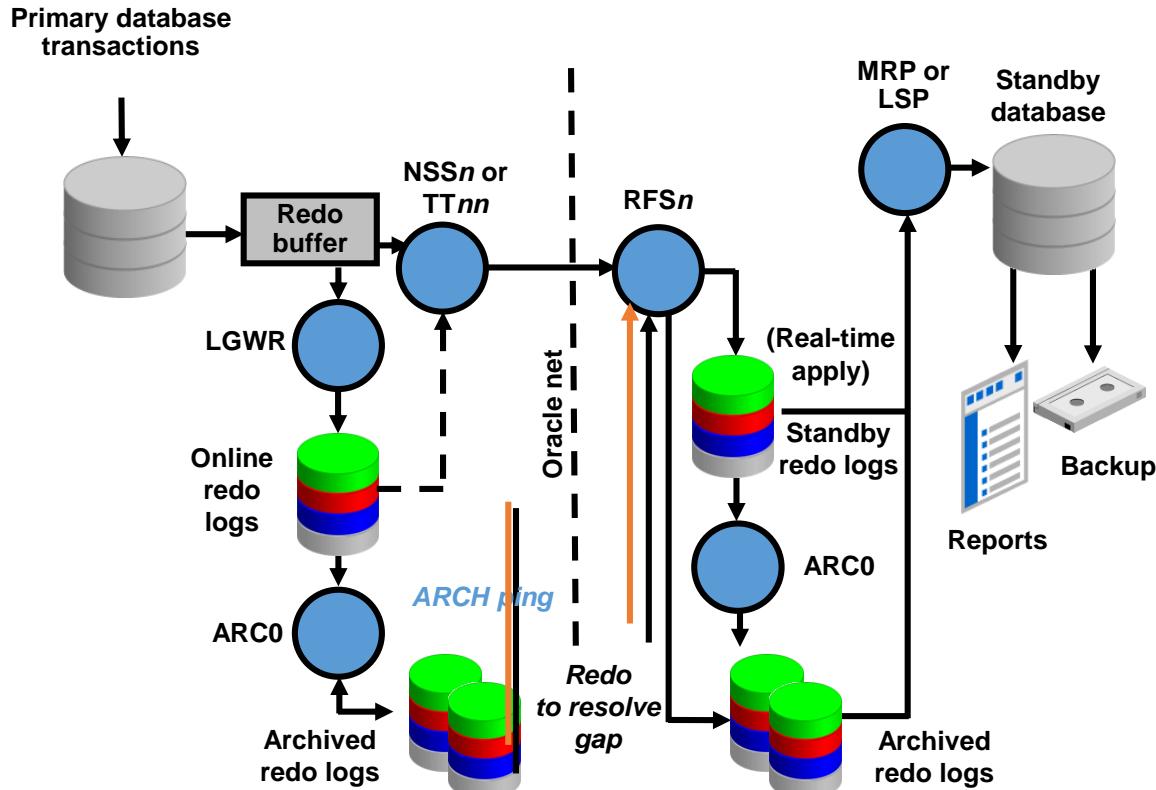
# Physical Standby Database: Redo Apply Architecture



# Logical Standby Database: SQL Apply Architecture



# Automatic Gap Detection and Resolution



# Data Protection Modes

- Select the mode to balance cost, availability, performance, and data protection:
  - Maximum protection
  - Maximum availability
  - Maximum performance



# Data Guard Operational Requirements: Hardware and Operating System

- Primary database systems and standby database systems may have different:
  - CPU architectures
  - Operating systems
  - Operating system binaries (32-bit or 64-bit)
  - Oracle Database binaries (32-bit or 64-bit)
- Many restrictions exist.

# Data Guard Operational Requirements: Oracle Database Software

- The same release of Oracle Database Enterprise Edition must be installed for all databases except when you perform a rolling database upgrade by using a logical standby database.
- If any database uses ASM or OMF, all databases should use the same combination.

# Benefits of Implementing Oracle Data Guard

- Oracle Data Guard provides the following benefits:
  - Continuous service during disasters or crippling data failures
  - Complete data protection against corruption and data loss
  - Elimination of idle standby systems
  - Flexible configuration of your system to meet requirements for business protection and recovery
  - Centralized management

# Quiz

- Which of the following are types of standby databases?
  - Physical
  - Primary
  - Logical
  - Snapshot

# Quiz

- What is the maximum number of standby databases supported by the Data Guard Broker?
  - 10
  - 20
  - 30
  - 40

# Summary

- In this lesson, you should have learned how to:
  - Describe the basic components of Oracle Data Guard
  - Explain the differences between physical and logical standby databases
  - Explain the benefits of creating a Data Guard environment

# Practice 1: Overview

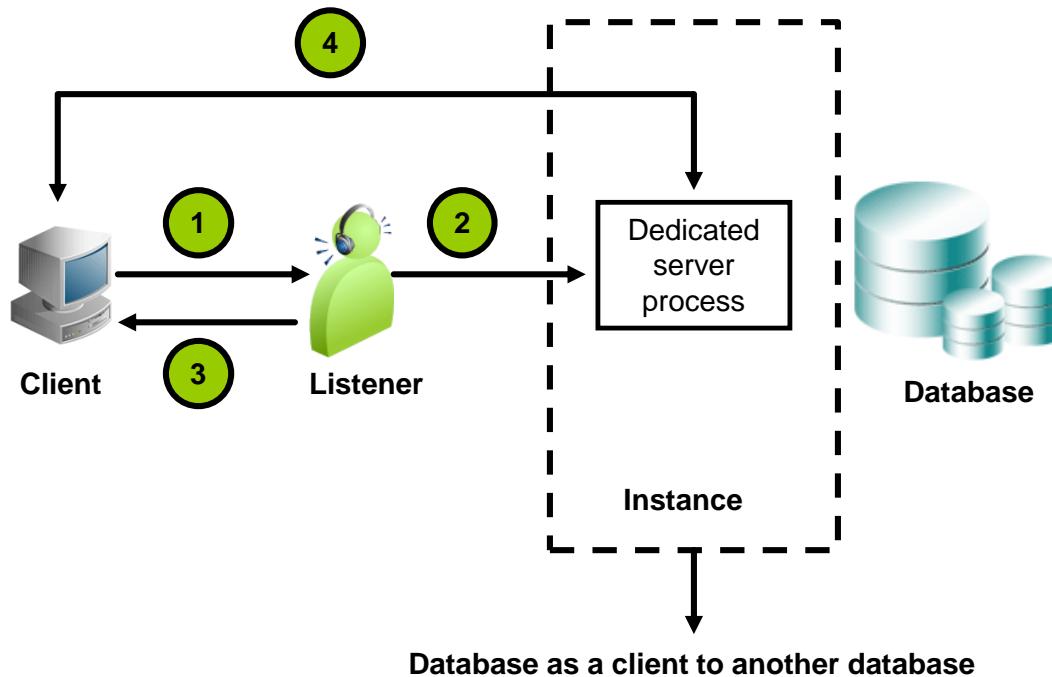
- This practice covers the following topics:
  - Discovering the Practice Environment

## **2. Oracle Net Services in a Data Guard Environment**

# Objectives

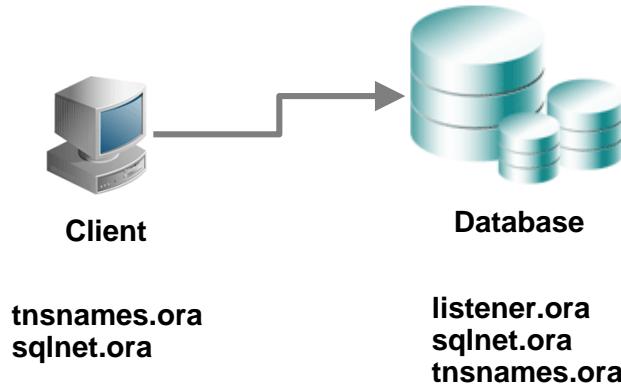
- After completing this lesson, you should be able to:
  - Understand the basics of Oracle Net Services
  - Configure client connectivity in a Data Guard configuration
  - Implement Data Guard best-practice solutions in networking setup

# Oracle Net Services Overview



# 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



# Understanding Name Resolution

- A naming method is a resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service. The following methods are available:
  - Local naming (uses static text-based configuration files)
  - Directory naming (uses a dynamic LDAP-compliant server)
  - Easy Connect naming (uses simple hard-coded strings)
  - External naming (uses a third-party naming service)

```
SQL> connect hr@london
```

Resolved into:

```
SQL> connect hr@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=host03.example.com) (PORT=1521)) (CONNECT_DATA=
(SERVICE_NAME=london.example.com)))
```

# Local Naming Configuration Files

- The location of the local naming method configuration files is specified by either the `TNS_ADMIN` or `ORACLE_HOME` operating system environment variables. For `ORACLE_HOME` locations consider:
  - Multiple products each with a different `ORACLE_HOME`
  - Multiple versions of the same product, each with a different `ORACLE_HOME`
  - Same product version installed multiple times for business isolation or upgrade path requirements, each with a different `ORACLE_HOME`



Client

`tnsnames.ora`  
`sqlnet.ora`



Database

`listener.ora`  
`sqlnet.ora`  
`tnsnames.ora`

# Local Naming: tnsnames.ora

- Database parameter file (spfileSID.ora/initSID.ora)

```
LOG_ARCHIVE_DEST_2='SERVICE=London ...'
```

- Oracle Net Configuration (on database host machine):  
\$ORACLE\_HOME/network/admin/tnsnames.ora

```
LONDON =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS=(PROTOCOL=TCP) (HOST=host03.example.com)
       (PORT=1521) (SEND_BUF_SIZE=10485760)
       (RECV_BUF_SIZE=10485760))
    )
    (SDU=65535)
    (CONNECT_DATA=(SERVICE_NAME=London.example.com) )
  )
```

# Connect-Time Failover: Planning for Role Reversal

- Connect-time failover is turned on by default for multiple address lists (ADDRESS\_LIST used) and does not have to be specified.

```
PRMY =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (FAILOVER=on)
      (ADDRESS=(PROTOCOL=TCP) (HOST=host01.example.com)
       (PORT=1521) (SEND_BUF_SIZE=10485760)
       (RECV_BUF_SIZE=10485760))
      (ADDRESS=(PROTOCOL=TCP) (HOST=host03.example.com)
       (PORT=1521) (SEND_BUF_SIZE=10485760)
       (RECV_BUF_SIZE=10485760)))
    )
    (SDU=65535)
    (CONNECT_DATA=(SERVICE_NAME=prmy.example.com))
  )
```

# Listener Configuration: listener.ora

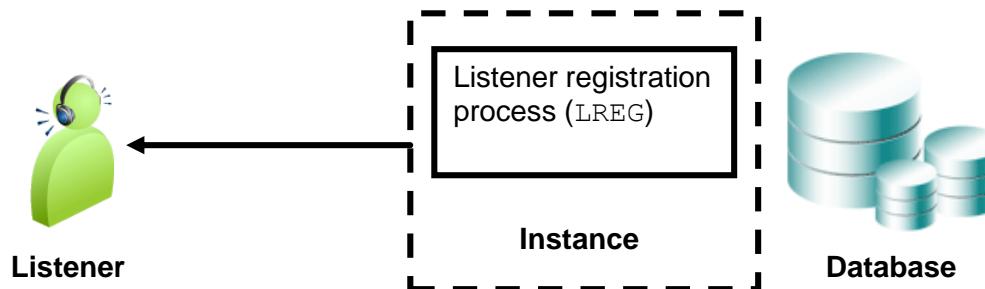
- Oracle Net Configuration (on database host machine):  
\$ORACLE\_HOME/network/admin/listener.ora
- The following entry defines two listening endpoints:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS=(PROTOCOL=TCP) (HOST=host03.example.com)
       (PORT=1521) (SEND_SDU=10485760)
       (RECV_SDU = 10485760)))
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1521)))
  )
ADR_BASE_LISTENER = /u01/app/oracle
LISTENER2 = ...
ADR_BASE_LISTENER2 = /u01/app/oracle
```

# Dynamic Service Registration

Dynamic service registration allows the 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



# Static Listener Entries: listener.ora

- Static listener entries are needed for Recovery Manager and Data Guard broker operations.

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = london.example.com)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
      (SID_NAME = london))
    (SID_DESC =
      (GLOBAL_DBNAME = london_DGMGRL.example.com)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
      (SID_NAME = london))
  )
```

# Optimizing Oracle Net for Data Guard

- Calculate the TCP Socket buffer size by multiplying the bandwidth delay product (BDP) by 3.

```
TCP Socket Buffer size=BDP*3=network bandwidth*latency*3
```

- Set the send and receive buffer sizes to the larger of BDP\*3 or 10 MB in both the `tnsnames.ora` and `listener.ora`.

```
(SEND_BUF_SIZE=10485760)  
(RECV_BUF_SIZE=10485760)
```

- Increase the session data unit (SDU) to 64 KB in both the `tnsnames.ora` and `listener.ora`.

```
(SDU=65535)
```

- Disable the TCP Nagle algorithm in the `sqlnet.ora` file.

```
TCP.NODELAY=YES
```

# Quiz

- Running the `oracle-database-preinstall-19c` package is sufficient to set up operating system kernel parameters for an Oracle Linux Data Guard environment using best-practice guidelines.
  - a. True
  - b. False

# Quiz

- In order to use the connect-time failover feature, the failover clause must be included in the `tnsnames.ora` file.
  - a. True
  - b. False

# Summary

- In this lesson, you should have learned how to:
  - Understand the basics of Oracle Net Services
  - Configure client connectivity in a Data Guard configuration
  - Implement Data Guard best-practice solutions in networking setup

# Practice 2: Overview

- This practice covers the following topics:
  - Configuring the tnsnames.ora Configuration File
  - Configuring the listener.ora Configuration File
  - Verifying the Oracle network configuration Files

# **3. Creating a Physical Standby Database by Using SQL and RMAN Commands**

# Objectives

- After completing this lesson, you should be able to:
  - Configure the primary database and Oracle Net Services to support the creation of the physical standby database and role transition
  - Describe the Database Nologging Enhancements
  - Create a physical standby database by using the `DUPPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE` RMAN command
  - Demonstrate the usage of PL/SQL procedure `DBMS_DBCOMP.DBCOMP`
  - Explain the creation of a standby database by using DBCA

# Steps to Create a Physical Standby Database

1. Prepare the primary database.
2. Set parameters on the physical standby database.
3. Configure Oracle Net Services.
4. Start the standby database instance.
5. Execute the **DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE RMAN** command.
6. Start the transport and application of redo.

# Preparing the Primary Database

- Enable FORCE LOGGING at the database level.
- Create a password file if required.
- Create standby redo logs.
- Set initialization parameters.
- Enable archiving.

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

# FORCE LOGGING Mode

- FORCE LOGGING mode ensures 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 for both physical and logical standby databases.
- Issue the following command on the primary database:

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

# Database Nologging Enhancements

- The STANDBY NOLOGGING FOR DATA AVAILABILITY mode causes the load operation to send the loaded data to each standby through its own standby connection.
- To enable STANDBY NOLOGGING FOR DATA AVAILABILITY mode:

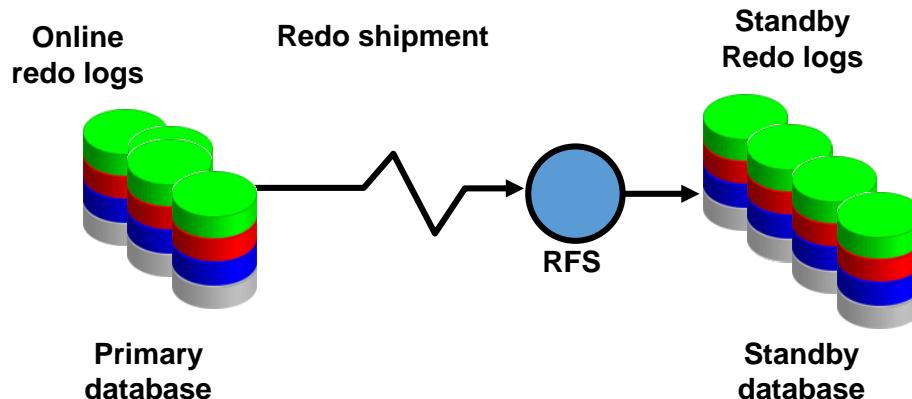
```
SQL> ALTER DATABASE SET STANDBY NOLOGGING FOR DATA AVAILABILITY;
```

- When STANDBY NOLOGGING FOR LOAD PERFORMANCE mode is enabled, the loading process can stop sending the data to the standbys if the network cannot keep up with the speed at which data is being loaded to the primary.
- To enable STANDBY NOLOGGING FOR LOAD PERFORMANCE mode:

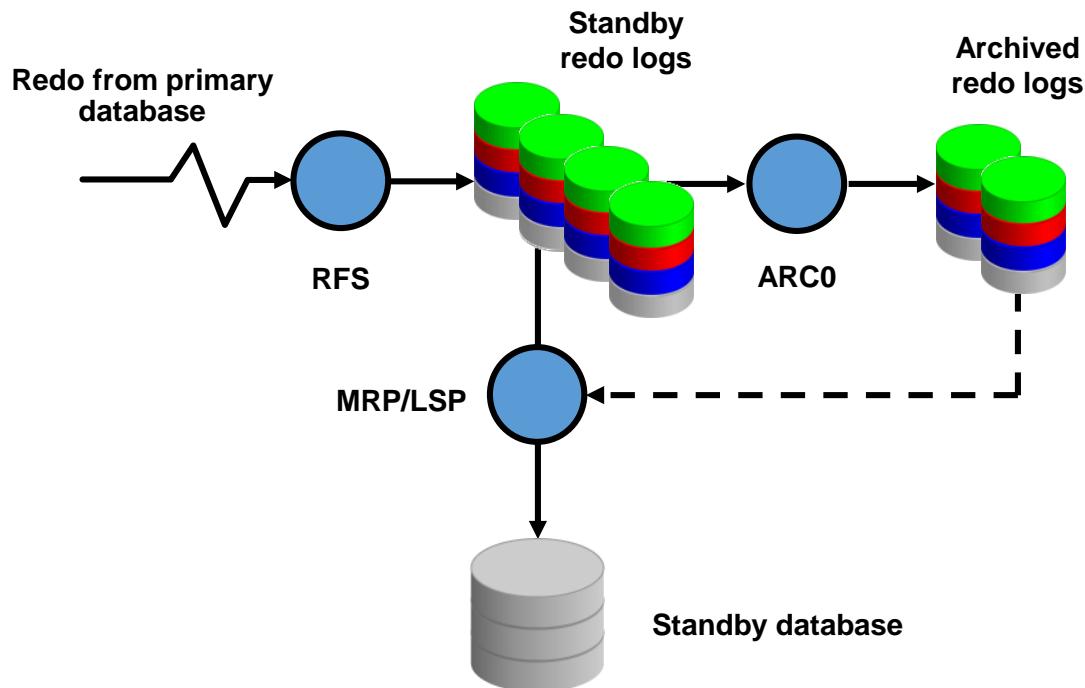
```
SQL> ALTER DATABASE SET STANDBY NOLOGGING FOR LOAD PERFORMANCE;
```

# 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.



# Standby Redo Log Usage



# Using SQL to Create Standby Redo Logs

- Create standby redo logs on the primary database to assist role changes:

```
SQL> alter database add standby logfile  
('/u01/app/oracle/oradata/boston/stdbyredo01.log') size 50M;  
Database altered.
```

or

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE  
  2  '+DATA' SIZE 52428800;  
Database altered.
```

# Viewing Standby Redo Log Information

- View information about the standby redo logs:

```
SQL> SELECT group#, type, member FROM v$logfile
  2 WHERE type = 'STANDBY';
GROUP# TYPE      MEMBER
-----
 4 STANDBY /u01/app/oracle/oradata/boston/stdbyredo01.log
 5 STANDBY /u01/app/oracle/oradata/boston/stdbyredo02.log
 6 STANDBY /u01/app/oracle/oradata/boston/stdbyredo03.log
 7 STANDBY /u01/app/oracle/oradata/boston/stdbyredo04.log

SQL> SELECT group#, dbid, thread#, sequence#, status
  2 FROM v$standby_log;
GROUP# DBID        THREAD#  SEQUENCE# STATUS
-----
 4 2581955083      1        44 ACTIVE
 5 UNASSIGNED       1          0 UNASSIGNED
 6 UNASSIGNED       1          0 UNASSIGNED
 7 UNASSIGNED       0          0 UNASSIGNED
```

# 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 and far sync instance in the configuration Enables or disables sending and receiving of redo
LOG_ARCHIVE_DEST_ <i>n</i>	Controls redo transport services
LOG_ARCHIVE_DEST_STATE_ <i>n</i>	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

# Setting LOG\_ARCHIVE\_CONFIG

- Specify the DG\_CONFIG attribute to list the DB\_UNIQUE\_NAME for the primary database and each standby database and far sync instance in the Data Guard configuration.

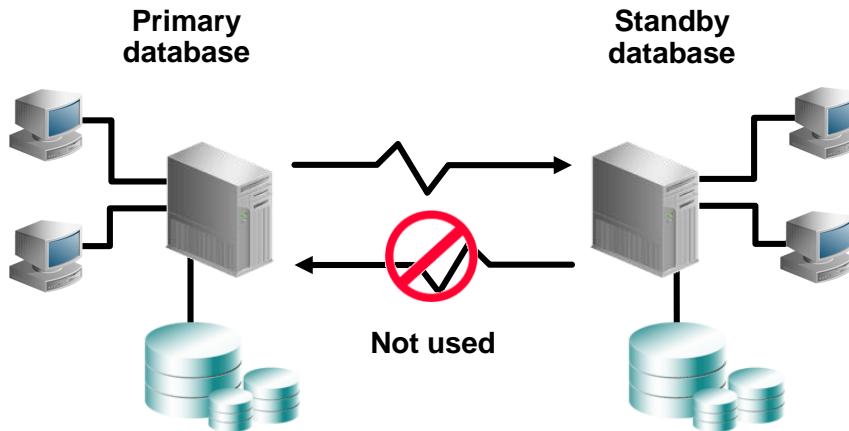
```
LOG_ARCHIVE_CONFIG='DG_CONFIG=(boston,bostonFS,linux,linux2,linuxFS)'
```

# Setting LOG\_ARCHIVE\_DEST\_n

- Specify LOG\_ARCHIVE\_DEST\_n parameters for:
  - Local archiving
  - Standby database location and far sync instance
- 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 or far sync instance
- Include a LOG\_ARCHIVE\_DEST\_STATE\_n parameter for each defined destination.

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

# Specifying Role-Based Destinations



```
log_archive_dest_2 =  
'service=london async  
valid_for=  
(online_logfile,  
primary_role)  
db_unique_name=london'
```

```
log_archive_dest_2 =  
'service=boston async  
valid_for=  
(online_logfile,  
primary_role)  
db_unique_name=boston'
```

# Combinations for VALID\_FOR

Combination	Primary	Physical	Logical
ONLINE_LOGFILE, PRIMARY_ROLE	Valid	Ignored	Ignored
ONLINE_LOGFILE, STANDBY_ROLE	Ignored	Ignored	Valid
ONLINE_LOGFILE, ALL_ROLES	Valid	Ignored	Valid
STANDBY_LOGFILE, STANDBY_ROLE	Ignored	Valid	Valid
STANDBY_LOGFILE, ALL_ROLES	Ignored	Valid	Valid
ALL_LOGFILES, PRIMARY_ROLE	Valid	Ignored	Ignored
ALL_LOGFILES, STANDBY_ROLE	Ignored	Valid	Valid
ALL_LOGFILES, ALL_ROLES	Valid	Valid	Valid

# Defining the Redo Transport Mode

- Use the attributes of `LOG_ARCHIVE_DEST_n`:
  - **SYNC** and **ASYNC**
    - Specify that network I/O operations are to be performed synchronously or asynchronously when using LGWR.
    - **ASYNC** is the default.
  - **AFFIRM** and **NOAFFIRM**
    - Ensure that redo was successfully written to disk on the standby destination.
    - **NOAFFIRM** is the default when **ASYNC** is specified; **AFFIRM** is the default when **SYNC** is specified.

# 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 physical 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

# 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.
- 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 = ('/u01/.../boston/' , '/u01/.../london/' ,  
'/u02/.../boston/' , '/u02/.../london/')
```

# 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.
- 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 = ('/u01/.../boston/logs/' , '/u01/.../london/logs/')
```

# 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
```

# 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 `boston` primary database:

```
FAL_SERVER = londonFS, london
```

- On the `london` standby database:

```
FAL_SERVER = bostonFS, boston
```

# Example: Setting Initialization Parameters on the Primary Database

- Primary database: boston; standby database: london

```
DB_NAME=boston
DB_UNIQUE_NAME=boston
LOG_ARCHIVE_CONFIG='DG_CONFIG=(boston, london, bostonFS, londonFS, london2)'
CONTROL_FILES='/u01/app/oracle/oradata/boston/control01.ctl',
 '/u01/app/oracle/oradata/boston/control02.ctl'
LOG_ARCHIVE_DEST_2=
 'SERVICE=london
  VALID_FOR=(ONLINE_LOGFILES, PRIMARY_ROLE)
  DB_UNIQUE_NAME=london'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
LOG_ARCHIVE_FORMAT=arch_%t_%s_%r.log
FAL_SERVER=london
STANDBY_FILE_MANAGEMENT=auto
DB_FILE_NAME_CONVERT='london', 'boston'
LOG_FILE_NAME_CONVERT='london', 'boston'
```

# Creating an Oracle Net Service Name on Primary for Your Physical Standby Database

- Use Oracle Net Manager to update the `tnsnames.ora` file:

```
LONDON =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS=(PROTOCOL=TCP) (HOST=host03.example.com)
       (PORT=1521) (SEND_BUF_SIZE=10485760)
       (RECV_BUF_SIZE=10485760))
    )
    (SDU=65535)
    (CONNECT_DATA=(SERVICE_NAME=london.example.com) )
  )
```

# Creating a Listener Entry for Your Standby Database

- Use Oracle Net Manager to configure an entry for your standby database in the `listener.ora` file:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = london.example.com)
      (ORACLE_HOME =
/u01/app/oracle/product/19.3.0/dbhome_1)
        (SID_NAME = london)
    )
  )
```

# Copying Your Primary Database Password File to the Physical Standby Database Host

1. Copy the primary database password file to the \$ORACLE\_HOME/dbs directory on the standby database host.
2. Rename the file for your standby database: orapw<SID>.

# Creating an Initialization Parameter File for the Physical Standby Database

- Create an initialization parameter file containing enough information to match the network listener services entries:
- File: \$ORACLE\_HOME/dbs/initlondon.ora

```
DB_NAME=london  
DB_DOMAIN=example.com
```

# Creating Directories for the Physical Standby Database

Create the baseline directory structures needed on the physical standby database host.

```
[oracle@host03]$ mkdir -p  
/u01/app/oracle/admin/london/adump  
[oracle@host03]$ mkdir -p  
/u01/app/oracle/oradata/london  
[oracle@host03]$ mkdir -p  
/u01/app/oracle/oradata/london/pdbseed  
[oracle@host03]$ mkdir -p  
/u01/app/oracle/oradata/london/dev1  
[oracle@host03]$ mkdir -p  
/u01/app/oracle/fast_recovery_area/london
```

# Starting the Physical Standby Database

- Start the physical standby database instance in NOMOUNT mode:

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

# Creating an RMAN Script to Create the Physical Standby Database

- 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.*

# Creating an RMAN Script to Create the Physical Standby Database

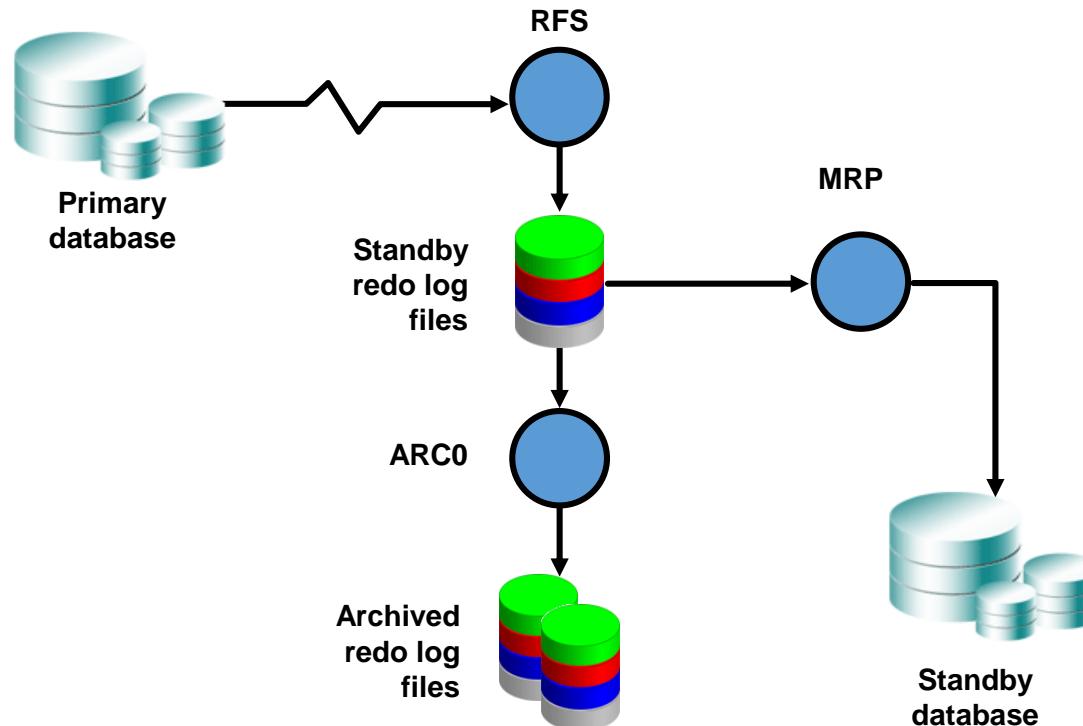
```
spfile
parameter_value_convert 'boston','london'
set db_unique_name='london'
set db_file_name_convert='boston','london'
set log_file_name_convert='boston','london'
set log_archive_max_processes='10'
set fal_server='boston'
set log_archive_config='dg_config=(boston,london)'
set log_archive_dest_2='service=boston ASYNC
    valid_for=(ONLINE_LOGFILE,PRIMARY_ROLE)
    db_unique_name=boston'
nofilenamecheck;
}
```

# Creating the Physical Standby Database

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

```
RMAN> connect target sys/oracle_4U@boston
RMAN> connect auxiliary sys/oracle_4U@london
RMAN> @cr_phys_standby
```

# Real-Time Apply (Default)



# Starting Redo Apply in Real Time

- Execute the following command on the standby database 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;
```

# Preventing Primary Database Data Corruption from Affecting the Standby Database

- Oracle Database processes can validate redo data before it is applied to the standby database.
- Corruption detection checks occur on the primary database during redo transport and on the standby database during Redo Apply.
- Implement lost-write detection by setting `DB_LOST_WRITE_PROTECT` to `TYPICAL` on the primary and standby databases.

# Detecting Lost Writes with DBMS\_DBCOMP . DBCOMP

- You can use DBMS\_DBCOMP . DBCOMP to detect lost writes and inconsistencies between the primary database and the physical standbys.
- DBMS\_DBCOMP . DBCOMP does not require that the parameter DB\_LOST\_WRITE\_PROTECT be enabled.
- You can monitor the progress of an ongoing block comparison operation by querying V\$SESSION\_LONGOPS.
- DBMS\_DBCOMP . DBCOMP assumes that there is one primary database and one or more physical standby databases.
- The databases should be at least mounted before block comparison.

# DBMS\_DBCOMP . DBCOMP Usage Cases

- 1.The primary (boston) and all standbys (london and chicago) are mounted or open and DBCOMP is executed from the primary.

```
SQL> exec sys.dbms_dbcomp.dbcomp('1','BlockCompare',:retval)
```

- The specified data files are compared block by block between the primary and every physical standby database.
- The generated output files are BlockCompare\_london\_1 and BlockCompare\_chicago\_1.

- 1.The primary and all standbys are mounted or open and DBCOMP is executed from a standby (london).

- The specified data files are compared between the primary and only that particular standby database.
- The generated output file is BlockCompare\_boston\_1.

# DBMS\_DBCOMP . DBCOMP Usage Cases

3.The primary database is mounted or open, but not all standbys are, and `DBCOMP` is executed from the primary.

- When `DBCOMP` is executed from the primary, the data files are compared between the primary and the mounted or open physical standbys.

4.The primary database is mounted or open, but not all standbys are, and `DBCOMP` is executed from a standby.

- The specified data files are compared between the primary and the standby from which `DBCOMP` is executed.

5.The primary database is not mounted, but multiple standbys are mounted or open.

- Because the primary is not mounted or open, `DBCOMP` cannot find the pair of primary and physical standbys to compare.

6.The primary is mounted or open, but no standbys are.

- Because no appropriate primary and physical standbys pair is found, a message is printed in the output file, but no ORA error is returned.

# Creating a Physical Standby Database by Using Enterprise Manager

The screenshot shows the Oracle Enterprise Manager Cloud Control interface. The top navigation bar includes 'Oracle Database', 'Performance', 'Availability' (which is currently selected), 'Security', 'Schema', and 'Administration'. Below the navigation is a summary card for 'Version 19.0.0.0.0' with 'MAA Advisor' and 'Backup & Recovery' sections. The 'Availability' section displays '0 days, 8 hrs Up Time' and 'n/a% Availability for Last 7 Days'. On the left, there are three main monitoring panels: 'Load and Capacity' (showing 0.00 Average Active Sessions and N/A Used Space (GB)), 'Incidents and Compliance' (with 0 errors, 1 warning, 0 incidents, and a note 'Compliance Not Configured'), and 'Recommendations'. The central area is titled 'Resources' and contains two charts: 'Memory (GB)' and 'Data Storage (GB)'. The 'Memory (GB)' chart shows a total of 1.00 GB used across various components: PGA (~0.50 GB), Large Pool (~0.25 GB), Cache (~0.15 GB), Shared Pool (~0.10 GB), and Other (~0.05 GB). The 'Data Storage (GB)' chart shows 'No data to display'.

# Using the Add Standby Database Wizard

**ORACLE® Enterprise Manager Cloud Control 13c**

## Add Standby Database

This wizard creates a Data Guard configuration containing a primary database and a standby database. Select how to add the standby database.

- Create a new physical standby database  
A physical standby database is maintained as an exact copy of the primary database.
- Create a new logical standby database  
A logical standby database duplicates the data from the primary database at the SQL level.
- Manage an existing standby database with Data Guard broker  
The existing standby database must be already configured to function with the primary database.
- Create a primary database backup only  
Creates a primary database backup that can be used for a future standby database creation.

Select “Create a new physical standby database.”

# Creating a Data Guard Standby by Using DBCA

- DBCA can be used as a simple command-line method to create an Oracle Data Guard physical standby database.
- DBCA can be used to create standby databases only for non-multitenant, single-instance primary databases.
- The DBCA command qualifier that is used to create the physical standby database is `createDuplicateDB`.

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

# Example: Physical Standby Creation

- **BOSTON**: Primary database
- **LONDON**: Standby database
- **example.com**: Domain

```
dbca -silent -createDuplicateDB -gdbName london.example.com  
-primaryDBConnectionString host01/boston.example.com  
-sid london -dbUniqueName london -createAsStandby
```

```
Enter SYS user password: *****  
Listener config step  
33% complete  
Auxiliary instance creation  
66% complete  
RMAN duplicate  
100% complete  
Look at the log file "/u01/app/oracle/product/19.3.0/dbhome_1  
/cfgtoollogs/dbca/london/london.log" for further details.
```

# Creating a Physical Standby by Using SQL

- Create a backup copy of the primary database data files.
- Create a control file for the standby database.

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS  
'/tmp/london01.ctl';
```

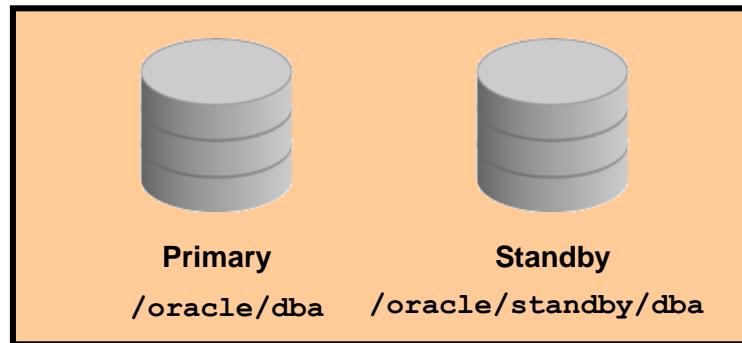
- Create a parameter file from the server parameter file used by the primary database and then adjust the parameters.

```
SQL> CREATE PFILE='/tmp/initlondon.ora' FROM SPFILE;
```

- Transfer the complete backup, standby control file, and parameter file to the standby system.
- Create a server parameter file and a startup instance.
- Restore the backup and start Redo Apply.

# Standby Database on the Same System

- Standby database data files must be at a different location.
- Each database instance must archive to different locations.
- Service names must be unique.
- This standby database does not protect against disaster.



# Quiz

- The FORCE LOGGING mode is required to support the unlogged operations in a Data Guard Environment.
  - a. True
  - b. False

# Quiz

- Which methods support the creation of a standby database in a multitenant environment?
  - a. Using EM
  - b. Using DBCA
  - c. Using RMAN DUPLICATE
  - d. All above

# Summary

- In this lesson, you should have learned how to:
  - Enable FORCE LOGGING
  - Create standby redo logs
  - Set initialization parameters on the primary database to support the creation of the physical standby database and role transition
  - Configure Oracle Net Services
  - Create a physical standby database by using the DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE **RMAN** command
  - Start the transport and application of redo

# Practice 3: Overview

- This practice covers the following topics:
  - Preparing the primary database prior to creating a physical standby database
  - Preparing hosts and creating the physical standby database
  - Verifying that the physical standby database is performing correctly

# **4. Managing Physical Standby Files After Structural Changes on the Primary Database**

# Objectives

- After completing this lesson, you should be able to describe:
  - The primary database changes that require manual intervention at a physical standby database.
  - The primary database changes that do not require manual intervention at a physical standby database.

# Scenario 1: Adding a Data File or Creating a Tablespace

- Assumptions:
  - STANDBY\_FILE\_MANAGEMENT = MANUAL
  - Active Data Guard option enabled
  - Action: Adding a Data File or Creating a Tablespace in the Primary Database



# Action Required on Physical Standby

1.On the primary database:

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

1.Checking the physical standby's Open Mode:

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

```
OPEN_MODE
```

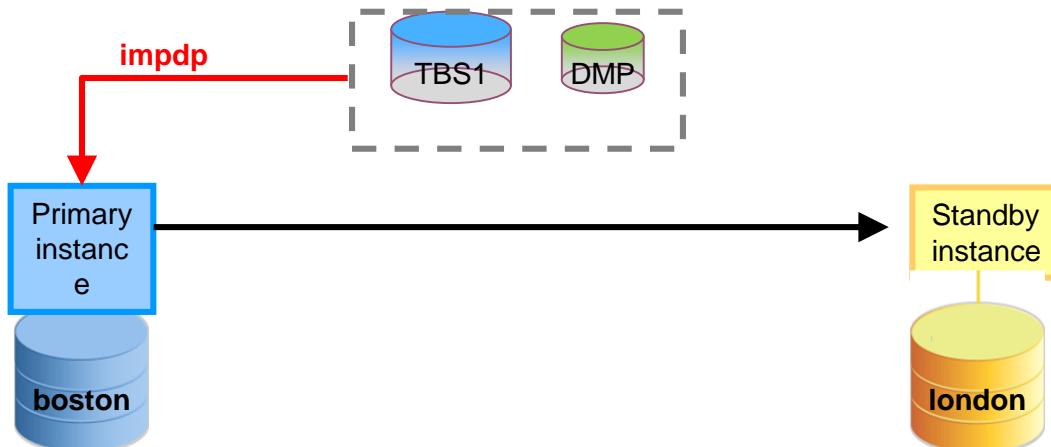
```
-----  
READ ONLY WITH APPLY
```

1.On the physical standby database:

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

# Scenario 2: Using Transportable Tablespaces with a Physical Standby Database

- Assumptions:
  - Path names are *NOT* the same on the primary and standby databases.
  - DB\_FILE\_NAME\_CONVERT is not configured.
  - Action: Performing Transportable Tablespaces in the Primary Database



# Action Required on Physical Standby

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

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

- 1.On the primary database, plug in the tablespace:

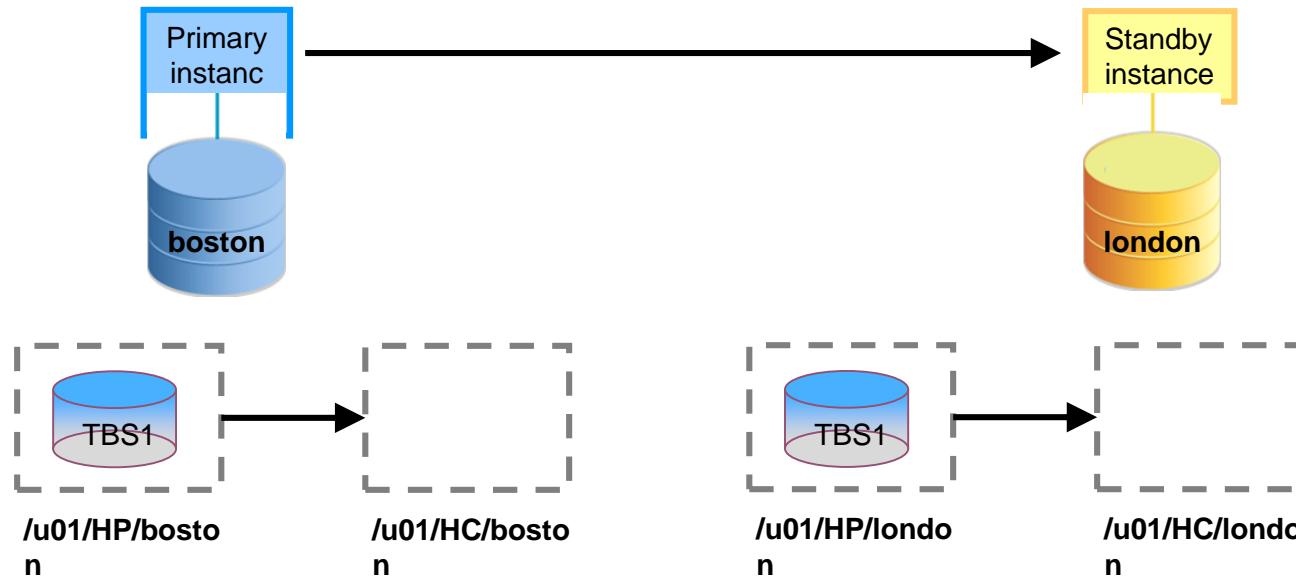
```
$ impdp system dumpfile=ttstest.dmp directory=data_pump_dir  
transport_datafiles='+DATA/boston/datafile/ttstest01.dbf'  
logfile=tts_import.log
```

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

# Scenario 3: Renaming a Data File in the Primary Database

- Assumptions:

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



# 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';
```

- On the standby database, note that the STANDBY\_FILE\_MANAGEMENT database initialization parameter must be set to MANUAL.
- Connect to the standby database and stop Redo Apply:

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

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

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

# Action Required on Physical Standby

5. Move the data file in the standby database

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

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

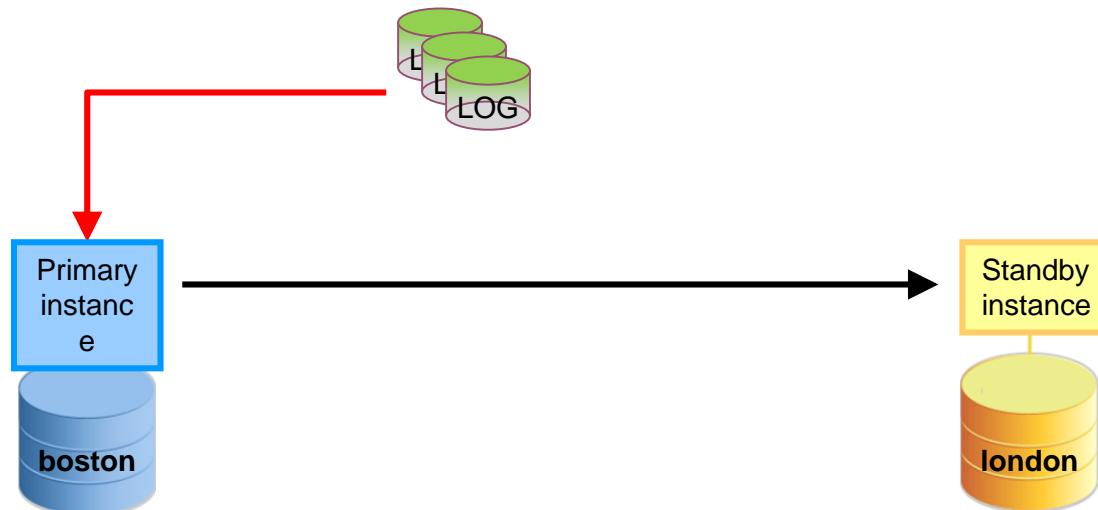
```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY  
DATABASE DISCONNECT;
```

5. The STANDBY\_FILE\_MANAGEMENT database initialization parameter can be reset after moving or renaming a data file.

# Scenario 4: Adding or Dropping a Redo File Group

- Assumption:

- STANDBY\_FILE\_MANAGEMENT = AUTO
- Action: Adding or Dropping a Redo File Group



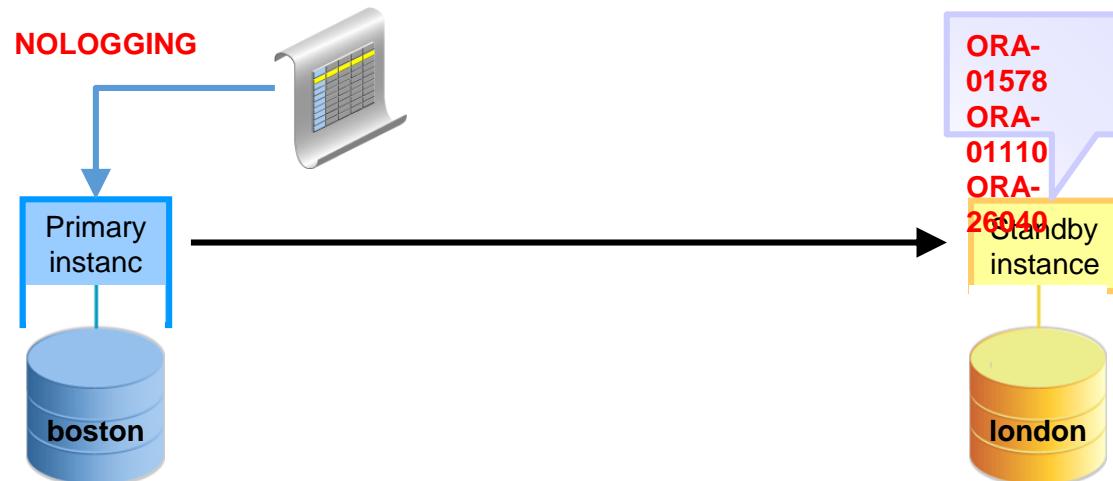
# Action Required on Physical Standby

1. Stop Redo Apply.
2. On the standby database, note that the STANDBY FILE MANAGEMENT database initialization parameter must be set to MANUAL.  
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.

# Scenario 5: NOLOGGING or Unrecoverable 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.



# 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
<hr/>	
/u01/boston/tbs_1.dbf	5216

NAME	UNRECOVERABLE
<hr/>	
/u01/london/tbs_1.dbf	5186

# 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;
```

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

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

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

# Scenario 6: Resetting the TDE Master Encryption Key

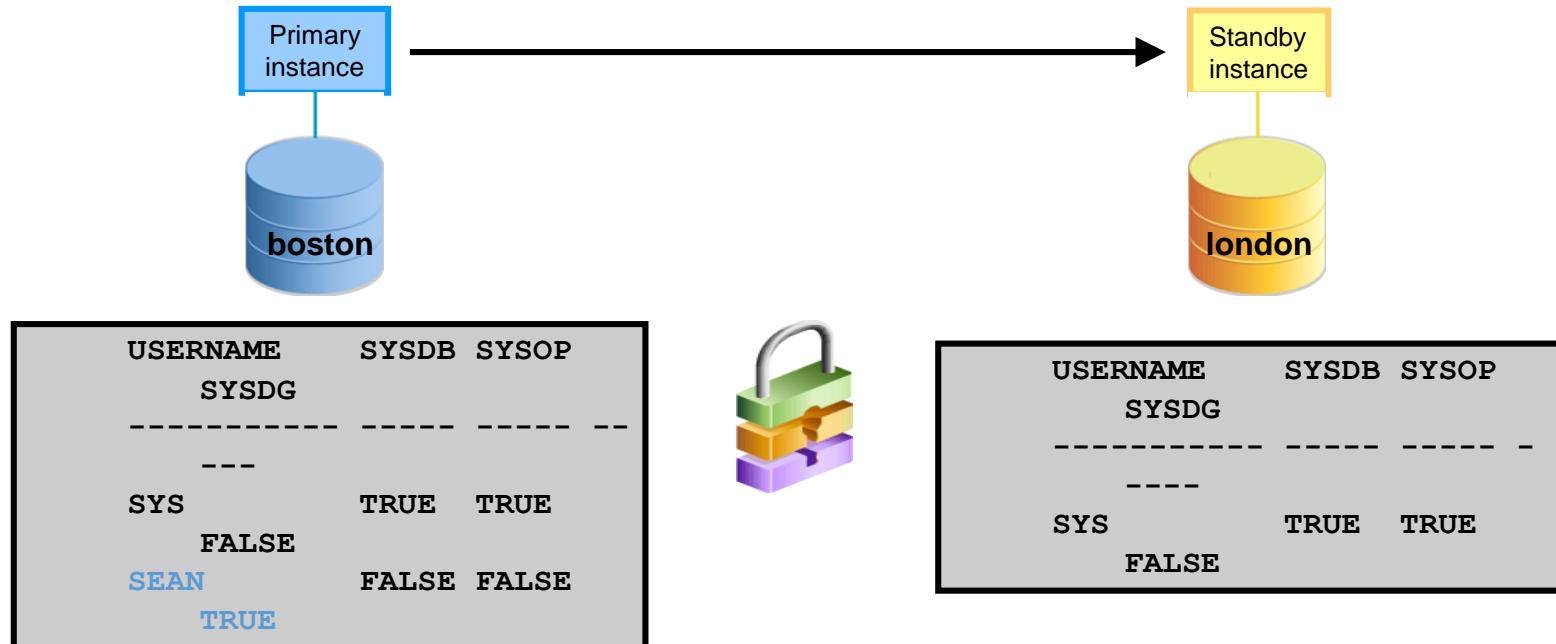
- Assumption:
  - The TDE master encryption key is reset on the primary database using the ADMINISTER KEY MANAGEMENT ALTER KEYSTORE PASSWORD command.



# Scenario 7: 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



# Scenario 8: Controlling PDB Replication

- The `ENABLED_PDBS_ON_STANDBY` parameter specifies which PDBs to replicate on a Data Guard standby database.
- The `ENABLED_PDBS_ON_STANDBY` parameter is valid only on a physical standby; it is ignored by primary databases.

```
ENABLED_PDBS_ON_STANDBY = "PDB1", "PDB2"
```

- If the parameter is not specified, all the PDBs in the CDB are created on the standby unless the `STANDBYS` clause is used.

```
CREATE PLUGGABLE DATABASE...
  STANDBYS='cdb_name1', 'cdb_name2'
```

- In an RAC environment, the same value must be used on all instances.

# Example: ENABLED\_PDBS\_ON\_STANDBY

- If STANDBY1 sets ENABLED\_PDBS\_ON\_STANDBY:
  1. "\*" on STANDBY1, all nine PDBs are created on STANDBY1
  2. "**PDB1\***" on STANDBY1, **PDB1A**, **PDB1B**, and **PDB1C** are created on STANDBY1
  3. "**PDB\*A**" on STANDBY1, **PDB1A**, **PDB2A**, and **PDB3A** are created on STANDBY1
  4. "**PDB1\***", "**-PDB\*A**" on STANDBY1, PDB1B and PDB1C are created on STANDBY1
  5. "**\***", "**-PDB\*A**", "**-PDB\*B**" on STANDBY1, **PDB1C**, **PDB2C** and **PDB3C** are created on STANDBY1
  6. "**\***", "**-PDB\*A**", "**PDB2A**" on STANDBY1, **PDB1A** and **PDB3A** are excluded, but all other PDBs, including **PDB2A**, will be created

# Scenario 9: Instantiating a PDB on a Standby

- Creating a PDB on a primary CDB:
  - From an XML file: Copy the data files specified in the XML file to the standby database.  
19c
  - Use the STANDBY\_PDB\_SOURCE\_FILE\_DIRECTORY parameter to specify a directory location on the standby where source data files for instantiating the PDB may be found □ Data files are automatically copied.  
18c
- As a clone from another PDB:
  - Copy the data files belonging to the source PDB to the standby database.  
19c
  - Use the STANDBY\_PDB\_SOURCE\_FILE\_DBLINK parameter to specify the name of a database link that is used to copy the data files from the source PDB to which the database link points.  
18c
    - The file copy is automatically done only if the database link points to the source PDB, and the source PDB is open in read-only mode.

# Quiz

- 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.
  - a. True
  - b. False

# Quiz

- The database role of a multitenant database is defined at the CDB level only. So, all the PDBs in the primary CDB must be created on the physical standby database.
  - a. True
  - b. False

# Quiz

- If you plan to create a PDB as a remote clone, then copying the data files belonging to the source PDB to the standby database is no longer required.
  - a. True
  - b. False

# Summary

- In this lesson, you should have learned how to describe:
  - The primary database changes that require manual intervention at a physical standby database
  - The primary database changes that do not require manual intervention at a physical standby database

# Practice 4: Overview

- This practice covers the following topics:
  - Refreshing the Password File in a Data Guard Configuration
  - Controlling PDB Replication in a Data Guard Configuration
  - Automating Instantiation of a PDB in a Data Guard Configuration

# 5. Using Oracle Active Data Guard:

## Supported Workloads in Read-Only Standby

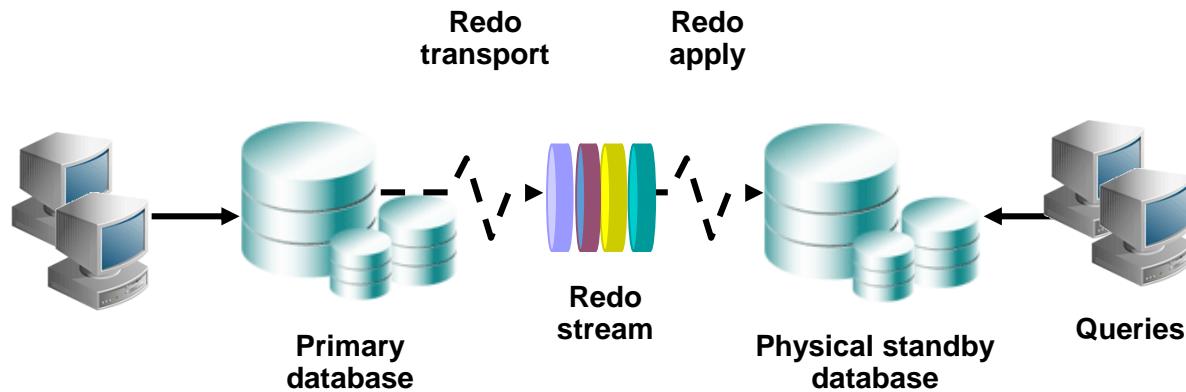
# Objectives

- After completing this lesson, you should be able to:
  - Describe the supported workload in Active Data Guard (Read-Only) instances
    - Perform DDL and DML on Global Temporary Tables
    - Perform DDL and DML on Private Temporary Tables
    - Configure Automatic Redirection of DML
    - Configure Automatic Redirection of PL/SQL
    - In-Memory Column Store Support

# Oracle Active Data Guard

- Is an option for Oracle Database Enterprise Edition
- Enhances quality of service by offloading resource-intensive activities from a production database to a standby database
- The lesson focuses on the following supported workloads:
  - Real-Time Query
  - DML on Temporary Tables
  - DDL on Temporary Tables
  - DML/DDL on Private Temporary Tables
  - Automatic Redirection of DML
  - Automatic Redirection of PL/SQL
  - In-Memory Column Store Support

# Using Real-Time Query



# Enabling Real-Time Query

1. Stop Redo Apply:

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

1. Open the database for read-only access:

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

1. Restart Redo Apply with the real-time default option:

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

# Disabling Real-Time Query

1. Shut down the standby database instance.

```
SQL> SHUTDOWN IMMEDIATE;
```

1. Restart the standby database instance in MOUNT mode.

```
SQL> STARTUP MOUNT;
```

1. Restart Redo Apply services (real-time apply).

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

# Checking the Standby's Open Mode

- A physical standby database opened in read-only mode:

```
SQL> SELECT open_mode FROM V$DATABASE;
OPEN_MODE
-----
READ ONLY
```

- A physical standby database opened in real-time query mode:

```
SQL> SELECT open_mode FROM V$DATABASE;
OPEN_MODE
-----
READ ONLY WITH APPLY
```

# Understanding Lag in an Active Data Guard Configuration

- A standby database configured with real-time query can lag behind the primary database as a result of:
  - Insufficient CPU capacity
  - High network latency
  - Limited bandwidth
- Queries on the standby database need to return current results and/or be within an established service level.
- Ways to “manage” the standby database lag and take necessary action:
  - Configure Data Guard with a maximum data lag that will trigger an error when it is exceeded.
  - Monitor the Redo Apply lag and take action when the lag is unacceptable.

# Monitoring Apply Lag: V\$DATAGUARD\_STATS

- *Apply lag*: This is the difference, in elapsed time, between when the last applied change became visible on the standby and when that same change was first visible on the primary.
- The apply lag row of the V\$DATAGUARD\_STATS view reflects statistics that are computed periodically and to the nearest second.

```
SQL> SELECT name, value, datum_time, time_computed
2> FROM v$dataguard_stats
3> WHERE name like 'apply lag';
```

NAME	VALUE	DATUM_TIME	TIME_COMPUTED
apply lag	+00 00:00:00	27-MAY-2019 08:54:16	27-MAY-2019 08:54:17

# Monitoring Apply Lag:

## V\$STANDBY\_EVENT\_HISTOGRAM

- View the histogram of apply lag on a physical standby database.
- Assess the value for STANDBY\_MAX\_DATA\_DELAY.
- Focus on periods of time when the apply lag exceeds desired levels so that the issue can be resolved.

```
SQL> SELECT * FROM V$STANDBY_EVENT_HISTOGRAM
2> WHERE NAME = 'apply lag' AND COUNT > 0;
```

NAME	TIME	UNIT	COUNT	LAST_TIME_UPDATED
apply lag	0	seconds	79681	06/18/2019 10:05:00
apply lag	1	seconds	1006	06/18/2019 10:03:56
apply lag	2	seconds	96	06/18/2019 09:51:06
apply lag	3	seconds	4	06/18/2019 04:12:32
apply lag	4	seconds	1	06/17/2019 11:43:51
apply lag	5	seconds	1	06/17/2019 11:43:52

```
6 rows selected
```

# Allowed Staleness of Standby Query Data

- The STANDBY\_MAX\_DATA\_DELAY session parameter specifies a session-specific limit for the amount of time (in seconds) allowed to elapse between when changes are committed on the primary and when those same changes can be queried on the standby database.

```
ALTER SESSION
```

```
SET STANDBY_MAX_DATA_DELAY = {INTEGER|NONE}
```

- If the limit is exceeded, an error message is returned.  
ORA-3172 STANDBY\_MAX\_DATA\_DELAY has been exceeded
- This setting is ignored for the SYS user.

# Configuring Zero Lag Between the Primary and Standby Databases

- Certain applications have zero tolerance for any lag.
- A query on the standby database must return the same result as if it were executed on the primary database.
- Enforce by setting `STANDBY_MAX_DATA_DELAY` to 0.
- The standby database must have advanced to a value equal to that of the current SCN on the primary database at the time the query was issued.
- Results are guaranteed to be the same as the primary database; otherwise, an `ORA-3172` error is returned.
- The primary database must operate in maximum availability or maximum protection mode.
- `SYNC` must be specified for redo transport.
- Real-time query must be enabled.

# Setting STANDBY\_MAX\_DATA\_DELAY by Using an AFTER LOGON Trigger

- Create an AFTER LOGON trigger that:
  - Is *database role-aware*
    - It uses DATABASE\_ROLE, a new attribute in the USERENV context.
    - SQL and PL/SQL clients can retrieve the database role programmatically using the SYS\_CONTEXT function.
    - It enables you to write role-specific triggers.
  - Sets STANDBY\_MAX\_DATA\_DELAY when the application logs on to a real-time query-enabled standby database
  - Allows for configuration of a maximum data delay without changing the application source code

## Example: Setting STANDBY\_MAX\_DATA\_DELAY by Using an AFTER LOGON Trigger

```
CREATE OR REPLACE TRIGGER sla_logon_trigger
  AFTER LOGON
  ON APP.SCHEMA
BEGIN
  IF (SYS_CONTEXT('USERENV', 'DATABASE_ROLE')
      IN ('PHYSICAL STANDBY'))
  THEN execute immediate
    'alter session set standby_max_data_delay=5';
  ENDIF;
END;
```

# Forcing Redo Apply Synchronization

- The `ALTER SESSION SYNC WITH PRIMARY` command:
  - Performs a blocking wait on the standby database upon execution
  - Blocks the application until the standby database is in sync with the primary database as of the time this command is executed
- When the `ALTER SESSION SYNC WITH PRIMARY` command returns control, the session can continue to process queries without having to wait for standby Redo Apply.
- An ORA-3173 Standby may not be synced with primary error is returned if Redo Apply is not active or is canceled before the standby database is in sync with the primary database as of the time this command is executed.

# Creating an AFTER LOGON Trigger for Synchronization

- Use an AFTER LOGON trigger to force a wait for synchronization between primary and standby databases.
- Use for dedicated connection only.
- This ensures that the reporting application starts with the current data without requiring a change to the application source code.

```
CREATE TRIGGER adg_logon_sync_trigger
AFTER LOGON ON user.schema
BEGIN
  IF (SYS_CONTEXT('USERENV', 'DATABASE_ROLE') IN
      ('PHYSICAL STANDBY'))
    THEN
      execute immediate 'alter session sync with
primary';
    END IF;
END;
```

# DDL on Global Temporary Tables

- The DDL operations on global temporary tables such as the CREATE or DROP commands can be run on an Active Data Guard standby database.
- The DDL for these operations is transparently redirected to the primary database.
- The Active Data Guard session then waits until the corresponding changes are shipped and applied to the Active Data Guard standby.

```
SQL> CREATE GLOBAL TEMPORARY TABLE tab2(c1 number,  
c2 varchar(10)) ON COMMIT PRESERVE ROWS;
```

- Enabled by setting parameter `enable_proxy_adg_redirect=TRUE` and appropriate connectivity parameters using `log_archive_dest_n`

# DML on Global Temporary Tables

- Read-mostly reporting applications that use global temporary tables for storing temporary data can be offloaded to an Oracle Active Data Guard instance.
  - When temporary undo is enabled on the primary, undo for changes to a global temporary table are not logged in the redo so the primary generates less redo.
  - The amount of redo that Oracle Data Guard must ship to the standby is also reduced, thereby reducing network bandwidth consumption and storage consumption.
- To enable temporary undo on the primary database, use the TEMP\_UNDO\_ENABLED initialization parameter.
- For a standby database the TEMP\_UNDO\_ENABLED initialization parameter is ignored because temporary undo is enabled by default on the standby database.

# DML/DDL on Private Temporary Tables

- You can create private temporary tables on Oracle Active Data Guard instances even though they are read-only.
- Private temporary tables can be created in read-only databases because the metadata is stored in memory, rather than on disk.
- The lifetime of a private temporary table is only during the session that created it.
- Two concurrent sessions may have a PTT with the same name but different shape.



- This allows reporting applications to run on Active Data Guard standby databases.

# Creating a Private Temporary Table

- Use the CREATE PRIVATE TEMPORARY TABLE statement to create a private temporary table.
- The ON COMMIT clause indicates if the data in the table is transaction-specific or session-specific.
  - This statement creates a private temporary table that is transaction-specific:

```
SQL> CREATE PRIVATE TEMPORARY TABLE ORA$PTT_sales_ptt_transaction  
      (time_id DATE, amount_sold NUMBER(10,2))  
      ON COMMIT DROP DEFINITION;
```

- This statement creates a private temporary table that is session-specific:

```
SQL> CREATE PRIVATE TEMPORARY TABLE ORA$PTT_sales_ptt_session  
      (time_id DATE, amount_sold NUMBER(10,2))  
      ON COMMIT PRESERVE DEFINITION;
```

# Support for Global Sequences

- Sequences created using the default CACHE and NOORDER options can be accessed from an Active Data Guard standby database.
- When first accessed by the standby, the primary allocates a unique range of sequence numbers.
- When all sequences within a range have been used, the standby requests another range of numbers.
- Because each range assigned to a standby is unique, there is a unique stream of sequences across the entire Data Guard configuration.
- Sequences created with the ORDER and NOCACHE options cannot be accessed on an Active Data Guard standby database.

# Support for Session Sequences

- Session sequences are specifically designed for use with global temporary tables that have session visibility.
  - They return a unique range of sequence numbers within a session.
  - Session sequences are not persistent. The state of the session sequences accessed during a session is lost when the session terminates.
- Session sequences are created by the primary database and are accessed on any read-write or read-only database.
- To create a session sequence:

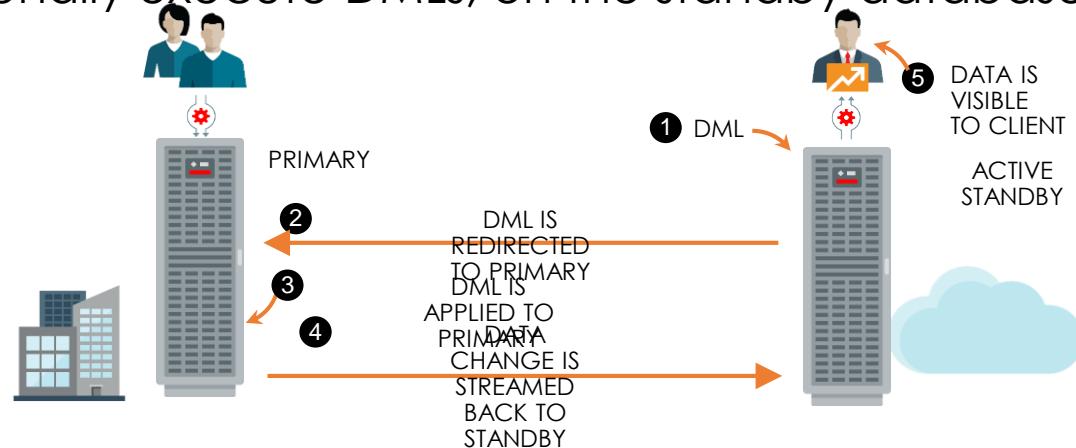
```
SQL> CREATE SEQUENCE ... SESSION;
```

# Benefits: Temporary Undo and Sequences

- Reporting and other applications that are generally read-only but require nonpersistent write access to the database can be run on an Active Data Guard standby by using temporary tables.
- Temporary undo reduces the redo volume if it is also enabled on the primary database. Temporary undo:
  - Is not logged in redo
  - Improves primary database performance
  - Reduces network bandwidth consumption
  - Reduces standby I/O
- Applications that are read-only except for the requirement to generate unique sequences can be offloaded to an Active Data Guard standby database.

# DML Operations on Active Data Guard Standby Databases

- DML operations can now be run on Active Data Guard standby databases.
- This enables you to run read-mostly applications, which occasionally execute DMLs, on the standby database.



# Configuring Automatic Redirection of DML operations

- Automatic redirection of DML operations to the primary can be configured at the system level or the session level.
- To configure automatic redirection of DML operations for all standby sessions, set the `ADG_REDIRECT_DML` initialization parameter to `TRUE`.
- To configure automatic redirection of DML operations for the current session:

```
SQL> ALTER SESSION ENABLE ADG_REDIRECT_DML;
```

- The session-level setting overrides the system-level setting.

# Example: Performing DML on a Physical Standby

- The physical standby database in an Active Data Guard setup contains a table named `employees`.
- To insert rows into this table by running DML on a physical standby database in the Active Data Guard environment:
  - On the standby database, enable DML redirection for the current

```
SQL> ALTER SESSION ENABLE ADG_REDIRECT_DML;
```

- Add a row to the `employees` table:

```
SQL> INSERT INTO employees VALUES (.....);
```
- The new data is visible only to the standby database on which the command was run.
- After the insert is committed on the primary database, the changes are shipped back and applied to all standby databases.

# PL/SQL Operations on Active Data Guard Standby Databases

- PL/SQL blocks that you run on Active Data Guard standby databases can be redirected to and run on the primary database.
  - The PL/SQL blocks should not contain bind variables.
- To redirect PL/SQL operations that are run on a standby to the primary, configure automatic redirection on the standby database:

```
SQL> ALTER SESSION ENABLE ADG_REDIRECT_PLSQL;
```

- You can configure automatic redirection for PL/SQL operations only at the session level.

# **IM Column Store in an Active Data Guard Standby Database**

- The IM column store is now supported on a standby database in an Active Data Guard environment.
- A reporting workload that is executing on an Active Data Guard standby database can use the IM column store.
- The IM column store improves the execution performance of the workload because it can take full advantage of accessing data in a compressed columnar format, in memory.
- It is possible to populate a completely different set of data in the IM column store on the primary and standby databases.

# Quiz

- The STANDBY\_MAX\_DATA\_DELAY parameter can be set at the system level.
  - a. True
  - b. False

# Quiz

- Which workloads can be automatically redirected to the primary database from an Active Data Guard (Read-Only) Database?
  - a. Analytic Queries
  - b. DML on Global Temporary Tables
  - c. Creation of Global Temporary Tables
  - d. Creation of Private Temporary Tables
  - e. DML on Regular Tables

# Summary

- In this lesson, you should have learned how to:
  - Describe the supported workload in Active Data Guard (Read-Only) instances
    - Perform DDL and DML on Global Temporary Tables
    - Perform DDL and DML on Private Temporary Tables
    - Configure Automatic Redirection of DML
    - Configure Automatic Redirection of PL/SQL
    - In-Memory Column Store Support

# Practice 5: Overview

- This practice covers the following topics:
  - Enabling Active Data Guard real-time query
  - Performing DDL/DML on Global Temporary Table in Active Data Guard Database
  - Performing DDL/DML on Private Temporary Table in Active Data Guard Database
  - Configuring Automatic Redirection of DML operations

# **6. Using Oracle Active Data Guard: Far Sync and Real-Time Cascading**

# Objectives

- After completing this lesson, you should be able to:
  - Use Far Sync to extend zero data loss protection for intercontinental configurations
  - Describe how to create a far sync instance by using RMAN
  - Describe how to create a far sync instance by using SQL
  - Describe the Real-Time Cascading

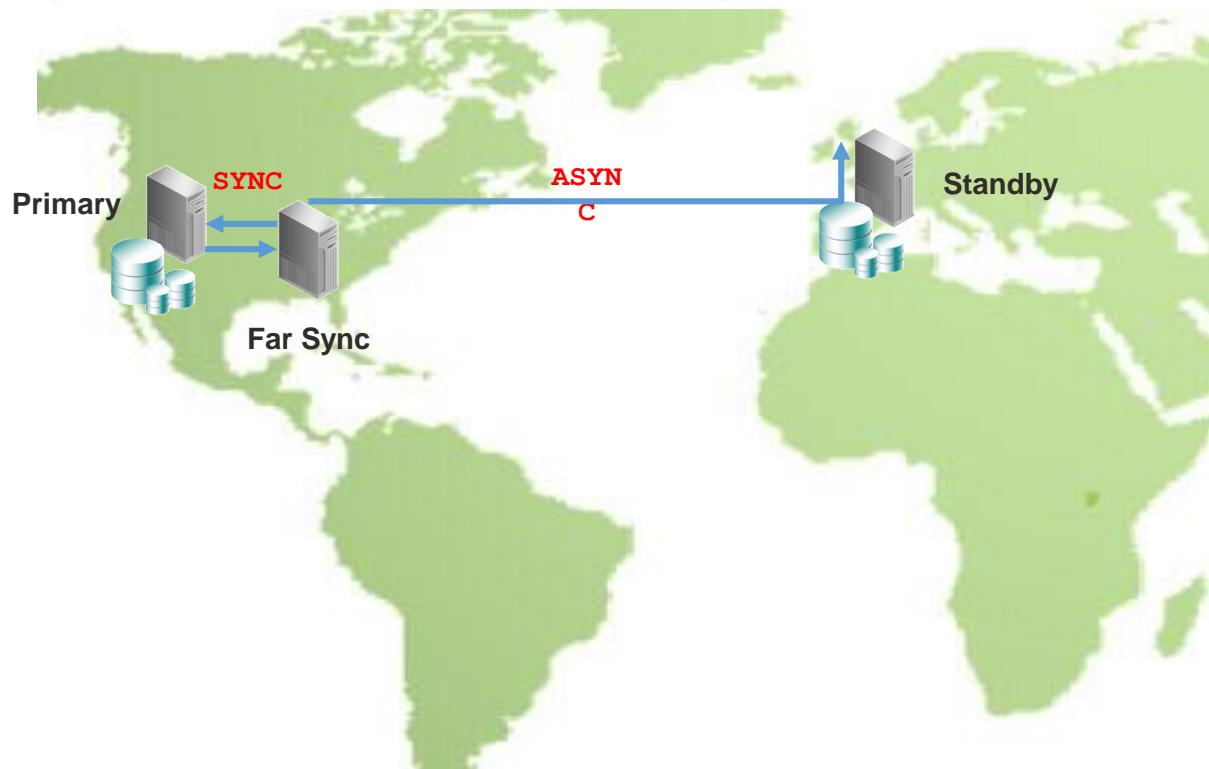
# Oracle Active Data Guard

- Is an option for Oracle Database Enterprise Edition
- Enhances quality of service by offloading resource-intensive activities from a production database to a standby database
- This lesson focuses on the following supported workloads:
  - Active Data Guard Far Sync
  - Real-Time Cascading

# Oracle Data Guard: Far Sync

- What is a Far Sync?
  - A lightweight Oracle database instance
    - Only spfile, far sync control file, password file, standby redo logs, and archive logs
    - No Oracle data files, no database to open for access, not running Redo Apply
  - Deployed within a distance such that the primary can tolerate the impact of network latency on synchronous transport
    - Looks like any other Data Guard destination to the primary database
    - With Data Guard 19c and higher, Fast Sync further extends the practical distance between the primary and a Far Sync.

# Far Sync: Redo Transport



# Far Sync: Redo Transport

- How does redo transport work with a Far Sync configuration?
  - The Far Sync instance receives redo synchronously from the primary database.
  - The Far Sync instance forwards redo asynchronously in real time to its final destination.
  - Redo transport supports one local and up to 30 additional local or remote destinations.
  - Oracle Recovery Manager (Oracle RMAN) deletion policies are used to automate archive log management.
  - A Far Sync instance can also compress redo transport.
  - An alternate Far Sync can be used for HA.

# Far Sync: Alternate Redo Transport Routes



```
LOG_ARCHIVE_DEST_2='SERVICE=bostonFS SYNC AFFIRM MAX_FAILURE=1
ALTERNATE=LOG_ARCHIVE_DEST_3 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=bostonFS'

LOG_ARCHIVE_DEST_STATE_2='ENABLE'

LOG_ARCHIVE_DEST_3='SERVICE=london ASYNC ALTERNATE=LOG_ARCHIVE_DEST_2
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=london'

LOG_ARCHIVE_DEST_STATE_3='ALTERNATE'
```

# Far Sync Instance Creation by Using RMAN

- 1.Prepare the auxiliary database instance as explained in the Oracle Database Backup and Recovery User's Guide.
- 2.Decide how to provide names for the standby control files, data files, online redo logs, and temp files.
- 3.Start RMAN and execute the DUPLICATE command.

```
DUPLICATE TARGET DATABASE
  FOR FARSYNC
FROM ACTIVE DATABASE
SPFILE
  SET "db_unique_name"="bostonFS" COMMENT "Is a duplicate"
  SET LOG_ARCHIVE_DEST_2="service=london ASYNC REGISTER
    VALID_FOR=(STANDBY_LOGFILE,STANDBY_ROLE)"
  SET FAL_SERVER="boston" COMMENT "Is primary"
NOFILENAMECHECK;
```

# Far Sync Instance Creation by Using SQL

- Create a control file using the mounted primary database:

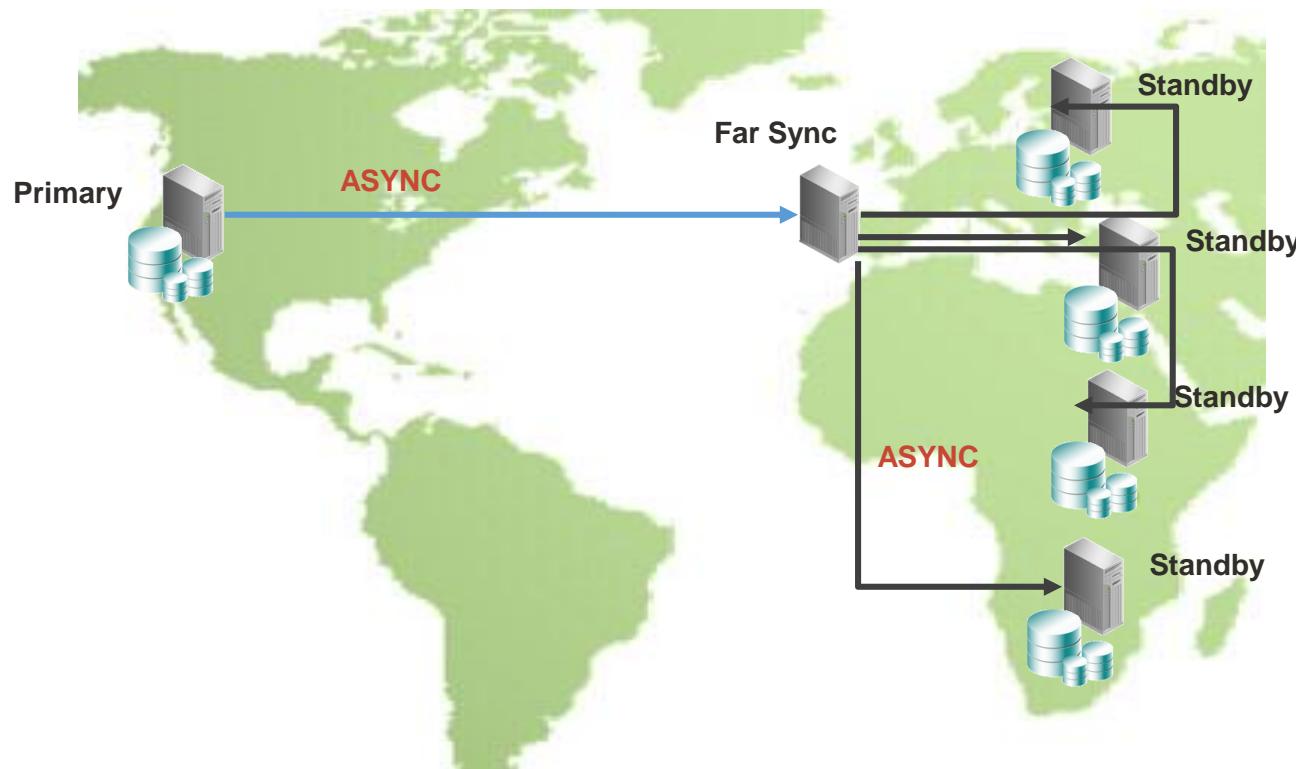
```
SQL> ALTER DATABASE CREATE FAR SYNC INSTANCE CONTROLFILE  
      AS '/tmp/boston1.ctl';
```

- Copy the parameter file (PFILE) and password file used by the primary database. (Several initialization parameters must also be modified for the Far Sync instance.)
- Copy the far sync control file to the Far Sync system.
- Create standby redo log files as you would for any standby.
- Start the Far Sync instance and mount the control file by using standard syntax.
- The **DATABASE\_ROLE** column in V\$DATABASE will contain the value FAR SYNC.

# Benefits: Far Sync

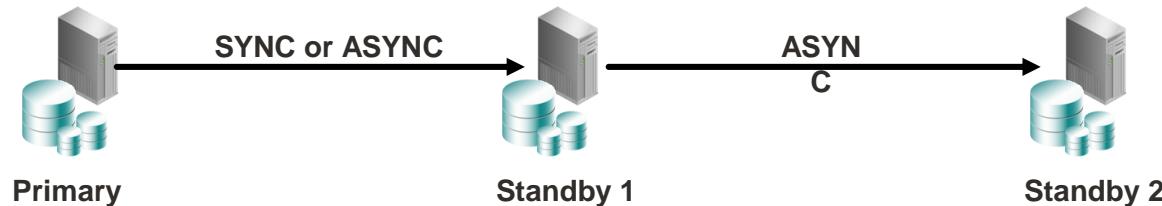
- Provides no-compromise data protection and primary offload
  - Ideal combination of attributes
    - Best data protection, least performance impact
    - Lower cost and complexity compared to previous multisite architecture
    - No change in applications required
    - Ideal for a combined near HA plus far DR model adopted by a growing number of users
  - Relevant to existing Data Guard `ASYNC` configurations
    - Enables the implementation of `SYNC` zero data loss where only `ASYNC` was previously possible
    - Offloads overhead for redo transport compression and for servicing multiple destinations

# Far Sync: Alternate Design



# Real-Time Cascade

- Traditional cascaded standby database
  - Primary redo is shipped to Standby 1 and then forwarded to Standby 2.
  - The redo is forwarded at the log switch, making Standby 2 always in the past of Standby 1.
- Real-time cascade, new for Data Guard 19c
  - Standby 1 forwards the redo to Standby 2 in real time, as it is received.
  - There is no propagation delay of Standby 2 waiting for a primary log switch.



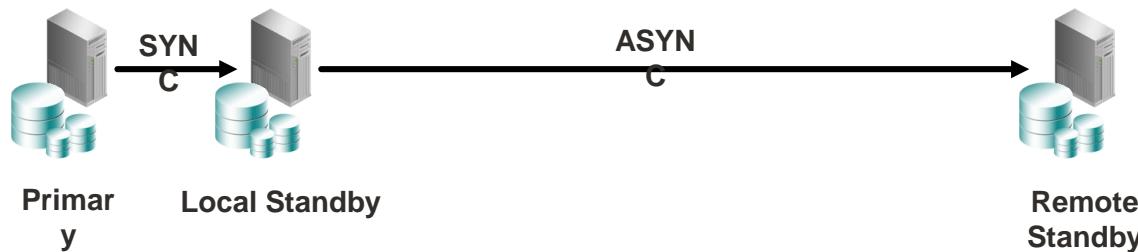
# Traditional Multi-Standby Database Architecture

- Multiple standby database configurations, the gold standard of WAN zero-data-loss architectures, consist of:
  - A local SYNC standby database for zero-data-loss protection
    - Fast, local HA failover with zero data loss
    - Offload RMAN backups, read-only workload, data extracts
    - Real Application Testing using Data Guard Snapshot Standby
    - Database rolling maintenance and standby-first patching
  - A remote ASYNC standby database for geo-protection



# Benefits: Real-Time Cascade

- Multiple standby database configurations can now use real-time cascade.
  - Less performance overhead
    - The primary database ships to a single destination.
  - Less network volume at the primary database



# Quiz

- Which statements are true about Far Sync instance?
  - a. The Far Sync includes a standby control file, password file, data files, standby redo logs, and archive logs
  - b. The Far Sync instance receives redo synchronously from the primary database.
  - c. The Far Sync instance applies redo received.
  - d. The Far Sync instance can be created using the RMAN DUPLICATE command

# Summary

- In this lesson, you should have learned how to:
  - Use Far Sync to extend zero data loss protection for intercontinental configurations
  - Describe how to create a far sync instance by using RMAN
  - Describe how to create a far sync instance by using SQL
  - Describe the Real-Time Cascading

# Practice 6: Overview

- This practice covers the following topics:
  - Add Far Sync to the Data Guard Environment
  - Add Second Far Sync to the Data Guard Environment

# **7. Creating and Managing a Snapshot Standby Database**

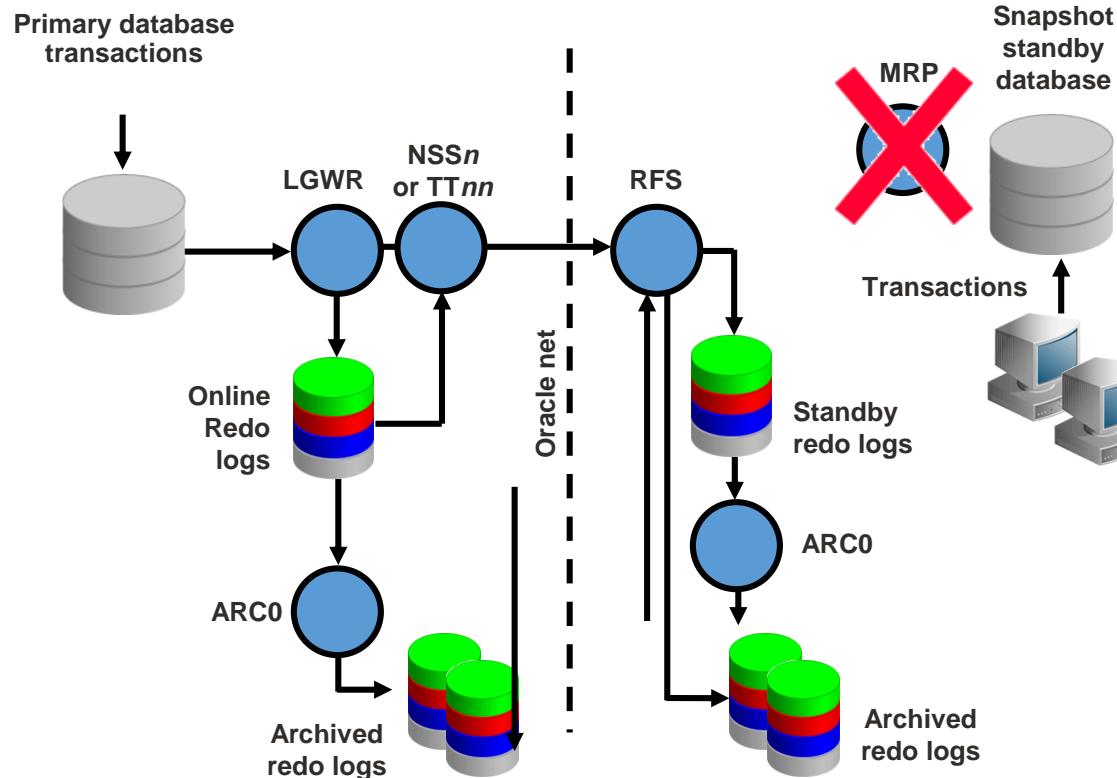
# Objectives

- 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
  - Convert a snapshot standby database back to a physical standby database

# Snapshot Standby Databases: Overview

- A snapshot standby database is a fully updatable standby database created by converting a physical standby database.
- 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.

# Snapshot Standby Database: Architecture



# Converting a Physical Standby Database to a Snapshot Standby Database

- To convert a physical standby to a snapshot standby:
  1. Stop Redo Apply if it is active.
  2. Ensure that the database is mounted, but not open.
  3. Ensure that the fast recovery area has been configured.
  4. Issue the following SQL statement to perform the

```
SQL> ALTER DATABASE CONVERT TO SNAPSHOT STANDBY;
```

1. Open the snapshot standby database in read-write mode

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

# Activating a Snapshot Standby Database: Issues and Cautions

- When activating a snapshot standby database, be aware of:
  - Potential data loss with a corrupted log file
  - Lengthy conversion of the snapshot standby database to a primary database in the event of a failure of the primary database

# Snapshot Standby Database: Target Restrictions

- A snapshot standby database cannot be:
  - The only standby database in a maximum protection configuration
  - The target of a switchover
  - A fast-start failover target

# Viewing Snapshot Standby Database Information

- View the database role by querying V\$DATABASE:

```
SQL> SELECT database_role FROM v$database;
```

```
DATABASE_ROLE
```

```
-----
```

```
SNAPSHOT STANDBY
```

# 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
<b>FLASHBACK LOG</b>	<b>2</b>	<b>.98</b>
FOREIGN ARCHIVED LOG	0	0
AUXILIARY DATAFILE COPY	0	0

# Activating a Snapshot Standby Database: Issues and Cautions

- To convert a physical standby to a snapshot standby:
  - 1.On an Oracle Real Applications Cluster (Oracle RAC) database, shut down all but one instance.
  - 2.Ensure that the database is mounted, but not open.
  - 3.Issue the following SQL statement to perform the conversion:

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

# Quiz

- A snapshot standby database can be created from a logical standby database.
  - a.True
  - b.False

# Summary

- In this lesson, you should have learned how to:
  - Create a snapshot standby database
  - Convert a snapshot standby database back to a physical standby database

# Practice 7: Overview

- This practice covers the following topics:
  - Converting a physical standby database to a snapshot standby database
  - Updating the primary database and the snapshot standby database
  - Converting the snapshot standby database back to a physical standby database

# 8. Creating a Logical Standby Database

# Objectives

- After completing this lesson, you should be able to:
  - Determine when to create a logical standby database
  - Create a logical standby database
  - Manage SQL Apply filtering

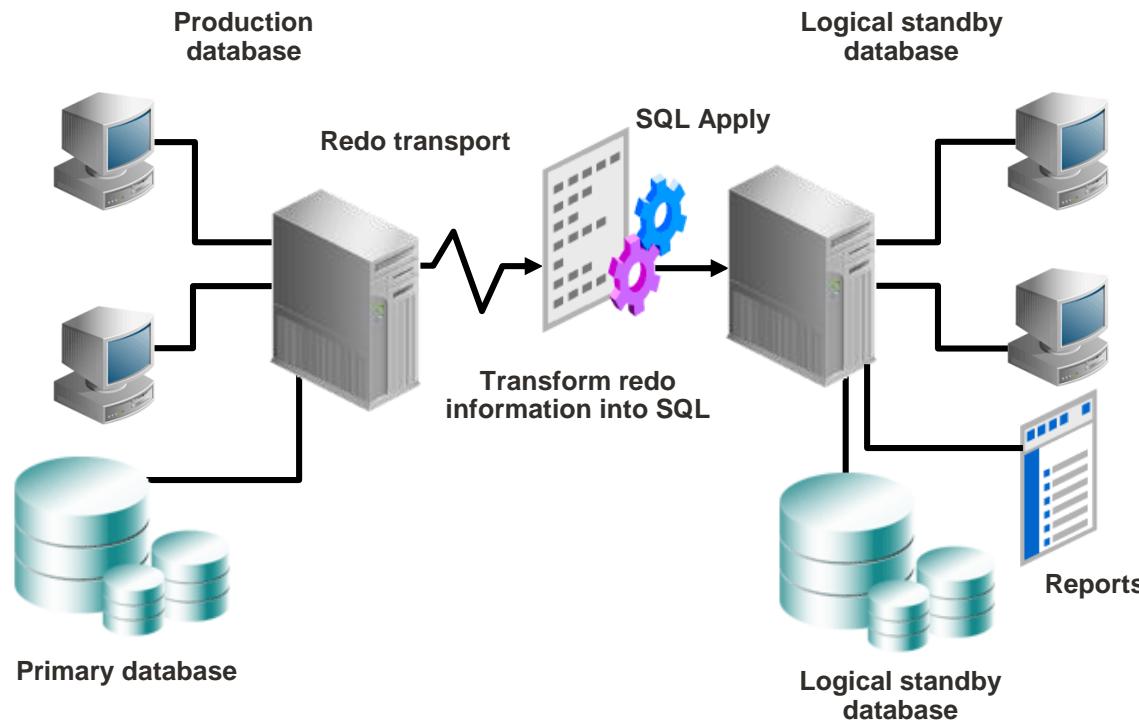
# Benefits of Implementing a Logical Standby Database

- Provides an efficient use of system resources:
  - Open, independent, and active production database
  - Additional indexes and materialized views can be created for improved query performance.
- Reduces workload on the primary database by offloading the following workloads to a logical standby database:
  - Reporting
  - Summations
  - Queries

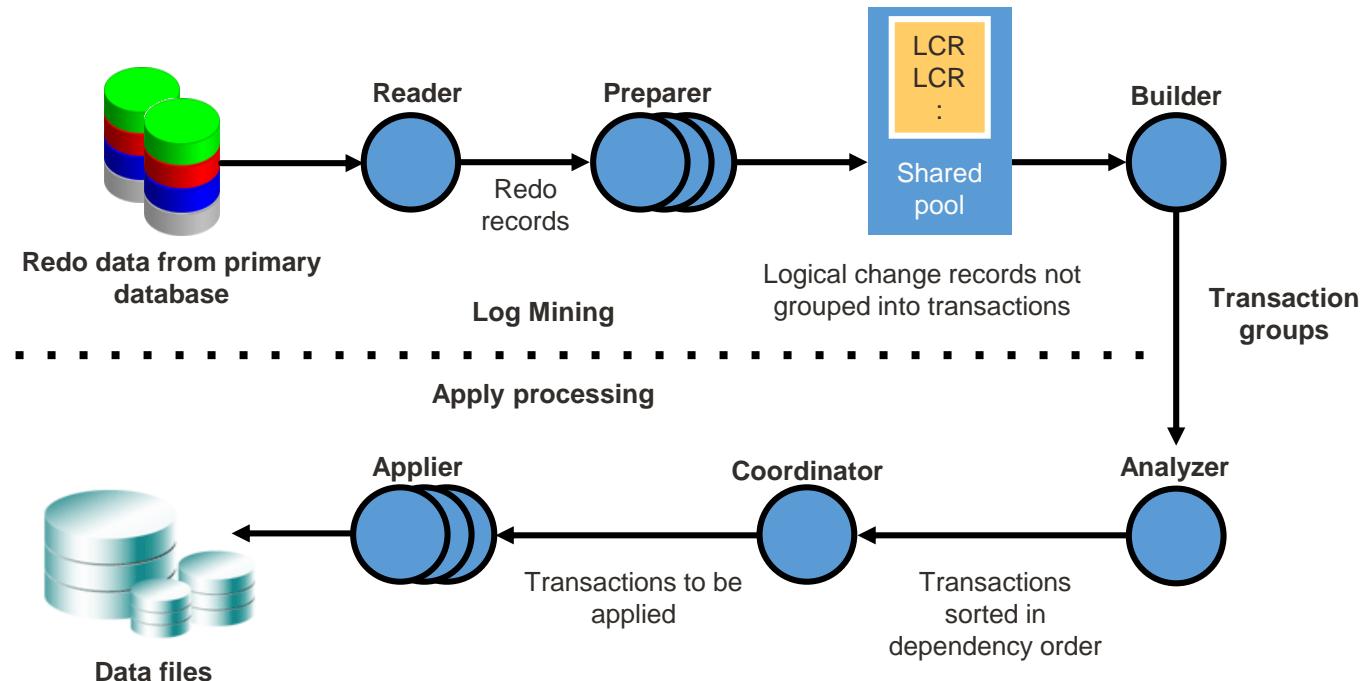
# Benefits of Implementing a Logical Standby Database

- Provides data protection:
  - Primary database corruptions not propagated
- Provides disaster recovery capabilities:
  - Switchover and failover
  - Minimizes down time for planned and unplanned outages
- Can be used to upgrade Oracle Database software and apply patch sets

# Logical Standby Database: SQL Apply Architecture



# SQL Apply Process: Architecture



# Preparing to Create a Logical Standby Database

- Perform the following steps on the primary database before creating a logical standby database:
  - Check for unsupported objects.
  - Be aware of unsupported DDL commands.
  - Ensure row uniqueness.
  - Verify that the primary database is configured for ARCHIVELOG mode.

# Unsupported Objects

- User tables placed in internal schemas will not be maintained.
- Log apply services *automatically exclude* unsupported objects when applying redo data to the logical standby database.
- Unsupported objects:
  - Tables and sequences in the SYS schema
  - Tables used to support materialized views
  - Global temporary tables
  - Tables with unsupported data types (listed in the next slide)

# Unsupported Data Types

- Log apply services *automatically exclude* entire tables with unsupported data types when applying redo data to the logical standby database.
- Unsupported data types:
  - BFILE
  - ROWID and UROWID
  - Collections (including VARRAYS and nested tables)
  - Objects with nested tables and REFS
  - The following Spatial types
    - MDSYS.SDO\_GEOASTER
    - MDSYS.SDO\_TOPO\_GEOMETRY

# Identifying Internal Schemas

- Query DBA\_LOGSTDBY\_SKIP on the primary database for internal schemas that will be skipped:

```
SQL> SELECT OWNER FROM DBA_LOGSTDBY_SKIP WHERE STATEMENT_OPT =
  'INTERNAL SCHEMA' ORDER BY OWNER;

Owner
-----
ANONYMOUS
APPQOSSYS
AUDSYS
BI
CTXSYS
...
SYSKM
SYSTEM
WMSYS
XDB
XS$NULL
30 rows selected.
```

# Checking for Unsupported Tables

- Query DBA\_LOGSTDBY\_UNSUPPORTED on the primary database for unsupported owners and tables:

```
SQL> SELECT DISTINCT OWNER, TABLE_NAME FROM  
DBA_LOGSTDBY_UNSUPPORTED ORDER BY OWNER, TABLE_NAME;
```

OWNER	TABLE_NAME
IX	AQ\$_STREAMS_QUEUE_TABLE_G
IX	AQ\$_STREAMS_QUEUE_TABLE_H
...	
OE	CATEGORIES_TAB
OE	CUSTOMERS
OE	PURCHASEORDER
PM	PRINT_MEDIA
SH	DIMENSION_EXCEPTIONS
20 rows selected.	

# Checking for Tables with Unsupported Data Types

- Query DBA\_LOGSTDBY\_UNSUPPORTED on the primary database for tables with specific unsupported data types:

```
SQL> SELECT table_name, column_name, data_type FROM  
dba_logstdby_unsupported WHERE owner = 'OE';
```

TABLE_NAME	COLUMN_NAME	DATA_TYPE
CUSTOMERS	PHONE_NUMBERS	VARRAY
PURCHASEORDER	SYS_NC_ROWINFO\$	OPAQUE
CATEGORIES_TAB	CATEGORY_NAME	VARCHAR2
CATEGORIES_TAB	CATEGORY_DESCRIPTION	VARCHAR2
CATEGORIES_TAB	CATEGORY_ID	NUMBER
CATEGORIES_TAB	PARENT_CATEGORY_ID	NUMBER

```
6 rows selected.
```

**Note:** Remove the WHERE clause for a full listing.

# SQL Commands That Do Not Execute on the Standby Database

- ALTER DATABASE
- ALTER MATERIALIZED VIEW
- ALTER MATERIALIZED VIEW LOG
- ALTER SESSION
- ALTER SYSTEM
- CREATE CONTROL FILE
- CREATE DATABASE
- CREATE DATABASE LINK
- CREATE PFILE FROM SPFILE
- CREATE MATERIALIZED VIEW
- CREATE MATERIALIZED VIEW LOG
- CREATE SCHEMA AUTHORIZATION
- CREATE SPFILE FROM PFILE
- DROP DATABASE LINK
- DROP MATERIALIZED VIEW
- DROP MATERIALIZED VIEW LOG
- EXPLAIN
- LOCK TABLE
- PURGE
- DBA RECYCLEBIN
- PURGE INDEX
- SET CONSTRAINTS
- SET ROLE
- SET TRANSACTION

# Unsupported PL/SQL-Supplied Packages

- Unsupported PL/SQL-supplied packages include:

- DBMS\_JAVA
- DBMS\_REGISTRY
- DBMS\_ALERT
- DBMS\_SPACE\_ADMIN
- DBMS\_REFRESH
- DBMS\_REDEFINITION
- DBMS\_AQ

- Packages supported only in the context of a rolling upgrade:

```
SQL> SELECT OWNER, PKG_NAME FROM DBA_LOGSTDBY_PLSQL_SUPPORT  
WHERE SUPPORT_LEVEL = 'DBMS_ROLLING';
```

# Ensuring Unique Row Identifiers

- Query DBA\_LOGSTDBY\_NOT\_UNIQUE on the primary database to find tables without a unique identifier:

```
SQL> SELECT * FROM dba_logstdby_not_unique;
      WHERE (OWNER, TABLE_NAME) NOT IN
            (SELECT DISTINCT OWNER, TABLE_NAME
              FROM DBA_LOGSTDBY_UNSUPPORTED)
      AND BAD_COLUMN = 'Y';
```

OWNER	TABLE_NAME	BAD_COLUMN
-----	-----	-----
APEX_040200	EMP_HIST	Y
...		

- BAD\_COLUMN indicates a table column is defined using an unbounded data type such as LONG or BLOB.
- Add a primary key or unique index to ensure that SQL Apply can efficiently apply data updates.

# Adding a Disabled Primary Key RELY Constraint

- You can add a disabled RELY constraint to uniquely identify rows:

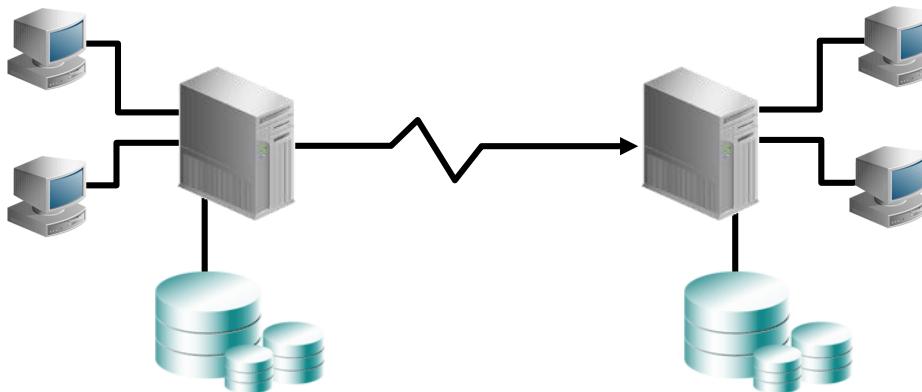
```
SQL> ALTER TABLE hr.employees
  2  ADD PRIMARY KEY (employee_id, last_name)
  3  RELY DISABLE;
```

# Creating a Logical Standby Database by Using SQL Commands

- To create a logical standby database by using SQL commands:
  1. Create a physical standby database.
  2. Stop Redo Apply on the physical standby database.
  3. Prepare the primary database to support a logical standby database.
  4. Build a LogMiner dictionary in the redo data on the primary database.
  5. Transition physical to a logical standby database.
  6. Open the logical standby database.
  7. Verify that the logical standby database is performing properly.

# Step 1: Create a Physical Standby Database

- a. Create a physical standby database.
- b. Ensure that the physical standby database is current with the primary database.



# Step 2: Stop Redo Apply on the Physical Standby Database

- Before converting the physical standby database to a logical standby database, stop Redo Apply.
- Stopping Redo Apply is required to avoid applying changes past the redo that contains the LogMiner dictionary.

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

## Step 3: Prepare the Primary Database to Support Role Transitions

Set the `LOG_ARCHIVE_DEST_n` initialization parameter for transitioning to a logical standby role.

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

**Note:** This step is necessary only if you plan to perform switchovers.

# Step 4: Build a LogMiner Dictionary in the Redo Data

- Build a LogMiner dictionary in the redo data so that SQL Apply can properly interpret changes in the redo.
- Supplemental logging is automatically enabled.
- Execute the procedure on the primary database:

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

# Step 5: Transition to a Logical Standby Database

- a. Execute the following command on the standby database to convert it to a logical standby database:

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

- a. Shut down the logical standby database instance and restart it in MOUNT mode.
- b. Adjust initialization parameters: `LOG_ARCHIVE_DEST_`*n*

# Step 6: Open the Logical Standby Database

- a. Open the new logical standby database with the RESETLOGS option:

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

- a. Start the application of redo data to the logical standby database:

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

## Step 7: Verify That the Logical Standby Database Is Performing Properly

- a. Verify that the archived redo log files were registered:

```
SQL> SELECT sequence#, first_time, next_time,
           dict_begin,      dict_end
      FROM dba_logstdby_log ORDER BY
sequence#;
```

- a. Begin sending redo data to the standby database:

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

- a. Query the DBA\_LOGSTDBY\_LOG view to verify that the archived redo log files were registered.

## Step 7: Verify That the Logical Standby Database Is Performing Properly

- d. Verify that redo data is being applied correctly:

```
SQL> SELECT name, value FROM v$logstdby_stats  
WHERE name = 'coordinator state';
```

- d. Query the V\$LOGSTDBY\_PROCESS view to see current SQL Apply activity:

```
SQL> SELECT sid, serial#, spid, type, high_scn  
FROM v$logstdby_process;
```

- d. Check the overall progress of SQL Apply:

```
SQL> SELECT applied_scn, latest_scn FROM  
v$logstdby_progress;
```

# Creating a Logical Standby Database by Using Enterprise Manager

**Add Standby Database**

This wizard creates a Data Guard configuration containing a primary database and a standby database. Select how to add the standby database.

- Create a new physical standby database  
A physical standby database is maintained as an exact copy of the primary database.
- Create a new logical standby database  
A logical standby database duplicates the data from the primary database at the SQL level.
- Manage an existing standby database with Data Guard broker  
The existing standby database must be already configured to function with the primary database.
- Create a primary database backup only  
Creates a primary database backup that can be used for a future standby database creation.

Select "Create a new logical standby database."

# Using the Add Standby Database Wizard

## SQL Apply Unsupported Tables

Some database tables are not maintained by SQL Apply due to unsupported data types in one or more columns of the table. Examine the list of unsupported tables below. If the list contains tables that you require to be maintained in the standby database, consider creating a physical standby database instead.

Previous 10 | 171-180 of 180 | Next

Show | Table Columns and Data Types | Go

Owner	Table Name	Column Name	Data Type
IX	AQ\$_ORDERS_QUEUETABLE_H	RETRY_COUNT	NUMBER
OE	PURCHASEORDER	SYS_NC_ROWINFO\$	OPAQUE
OE	CATEGORIES_TAB	CATEGORY_ID	NUMBER
OE	CATEGORIES_TAB	CATEGORY_DESCRIPTION	VARCHAR2
OE	CUSTOMERS	PHONE_NUMBERS	VARRAY
OE	CATEGORIES_TAB	PARENT_CATEGORY_ID	NUMBER
OE	CATEGORIES_TAB	CATEGORY_NAME	VARCHAR2
PM	PRINT_MEDIA	AD_GRAPHIC	BFILE
PM	PRINT_MEDIA	AD_TEXTDOCS_NTAB	NESTED TABLE
SH	DIMENSION_EXCEPTIONS	BAD_ROWID	ROWID

Previous 10 | 171-180 of 180 | Next

# Using the Add Standby Database Wizard

**ORACLE® Enterprise Manager Cloud Control 13c** SYSMAN ▾

Backup Type    Backup Options    Database Location    File Locations    Configuration    Review

**Add Standby Database: Configuration**

Primary Database **boston.example.com**    Primary Host **host01.example.com**    Standby Host **host03.example.com**

[Cancel](#) [Back](#) Step 5 of 6 [Next](#)

**Standby Database Parameters**

\* Database Name  This name will be used for the standby database db\_name parameter.

i \* Database Unique Name  Used to set the standby database DB\_UNIQUE\_NAME parameter, which must be unique within the enterprise.

\* Target Name  The display name used by Enterprise Manager for the standby database. Oracle recommends that it be the same as the Database Unique Name.

\* Standby Archive Location

Location for archived redo log files received from the primary database.

This location is a regular directory

Configure this location as a fast recovery area

[Fast Recovery Area Size \(MB\)](#)

Limit on the total space used by files created in the fast recovery area. The default value is twice the database size.

# Using the Add Standby Database Wizard

**ORACLE® Enterprise Manager Cloud Control 13c**      SYSMAN ▾      ...

Previous   Configuration   **Review**

### Add Standby Database: Review

The standby database creation process runs as an Enterprise Manager job. Standby database **london2.example.com** will be created by job **DataGuardCreateStandby2** and added to the Data Guard configuration.

**Cancel** **Back** Step 6 of 6 **Finish**

Primary Database		Standby Database	
Target Name	<b>boston.example.com</b>	Target Name	<b>london2.example.com</b>
Database Name	<b>boston</b>	Database Name	<b>london2</b>
Instance Name	<b>boston</b>	Instance Name	<b>london2</b>
Database Version	<b>19.0.0.0</b>	Oracle Server Version	<b>19.0.0.0.0</b>
Oracle Home	<b>/u01/app/oracle/product/19.3.0/dbhome_1</b>	Oracle Home	<b>/u01/app/oracle/product/19.3.0/dbh</b>
Host	<b>host01.example.com</b>	Host	<b>host03.example.com</b>
Operating System	<b>Oracle Linux Server release 7.7 4.14.35</b>	Operating System	<b>Oracle Linux Server release 7.7 4.14.3</b>
Host Username	<b>oracle</b>	Host Username	<b>oracle</b>
		Backup Type	<b>New backup</b>
		File Transfer Method	<b>RMAN duplicate</b>
		Database Unique Name	<b>london2</b>
		Standby Type	<b>Logical Standby</b>
		Fast Recovery Area	<b>/u01/app/oracle/oradata/london2/ar</b>
		Fast Recovery Area Size (MB)	<b>7340M</b>
		Automatically Delete Archived Redo Log Files	<b>No</b>

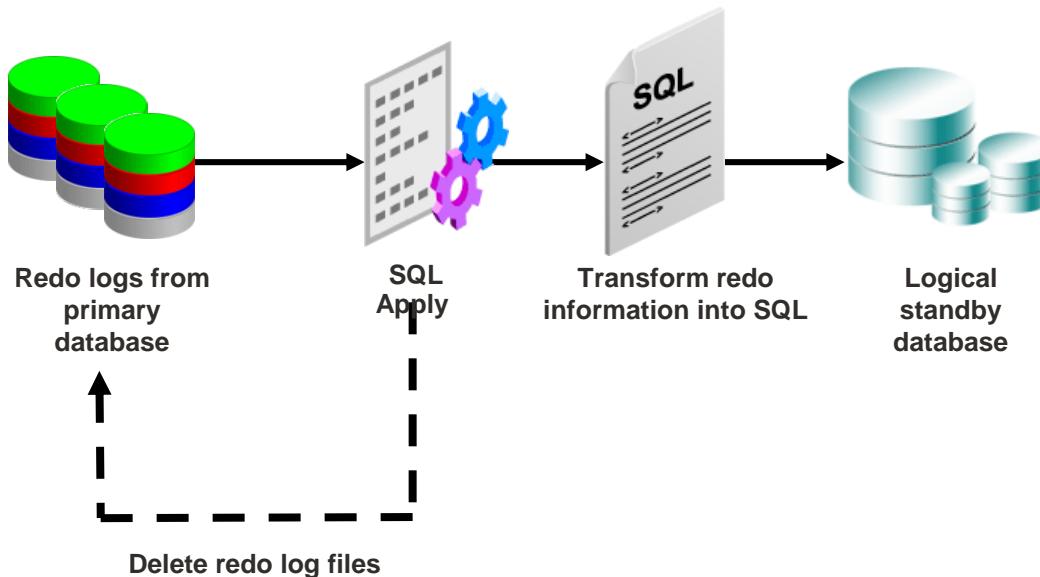
**Standby Database Storage**

**Cancel** **Back** Step 6 of 6 **Finish**

# Securing Your Logical Standby Database

- Configure the database guard to control user access to tables.
- ALTER DATABASE GUARD command keywords:
  - ALL: Prevents users from making changes to any data in the database
  - STANDBY: Prevents users from making changes to any data maintained by Data Guard SQL Apply
  - NONE: Normal security
- Query the GUARD\_STATUS column in V\$DATABASE.
- The database guard level is automatically set to ALL by the broker on the logical standby database.
- The database guard level applies to all users except SYS.

# Automatic Deletion of Redo Log Files by SQL Apply



# Managing Remote Archived Log File Retention

- The `LOG_AUTO_DEL_RETENTION_TARGET` parameter:
  - Is used to specify the number of minutes that SQL Apply keeps a remote archived log after completely applying its contents
  - Is applicable only if `LOG_AUTO_DELETE` is set to TRUE and the fast recovery area is not being used to store remote archived logs
  - Has a default value of 1,440 minutes

```
SQL> execute DBMS_LOGSTDBY.APPLY_SET
('LOG_AUTO_DEL_RETENTION_TARGET','2880') ;
```

# Creating SQL Apply Filtering Rules

- Procedures exist to create SQL Apply filtering rules.
  - To skip SQL statements:

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP(STMT => 'DML',  
schema_name => 'HR', object_name => '%') ;
```

- To skip errors and continue applying changes:

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP_ERROR('GRANT') ;
```

- To skip applying transactions:

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP_TRANSACTION  
(XIDUSN => 1, XIDSLT => 13, XIDSQN => 1726) ;
```

- To skip all SQL statements for a container:

```
SQL> EXECUTE DBMS_LOGSTDBY.SKIP(stmt =>  
'CONTAINER', object_name => 'DEV1') ;
```

# Deleting SQL Apply Filtering Rules

- Procedures exist to delete SQL Apply filtering rules.
  - To delete SQL statement rules:

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP(STMT => 'DML',  
schema_name => 'HR', object_name => '%') ;
```

- To delete error apply filter rules:

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP_ERROR('GRANT') ;
```

- To delete transaction filter rules:

```
SQL> EXECUTE DBMS_LOGSTDBY.UNSKIP_TRANSACTION  
(XIDUSN => 1, XIDSLT => 13, XIDSQN => 1726) ;
```

# Viewing SQL Apply Filtering Settings

- Query DBA\_LOGSTDBY\_SKIP to view SQL Apply filtering settings:

```
SQL> SELECT error, statement_opt, name
  2  FROM dba_logstdby_skip
  3  WHERE owner='HR';
```

ERROR	STATEMENT_OPT	NAME
N	DML	JOBS

# Using DBMS\_SCHEDULER to Create Jobs on a Logical Standby Database

- Scheduler jobs can be created on a standby database.
- When a Scheduler job is created, it defaults to the local role.
- Activate existing jobs by using the `DATABASE_ROLE` attribute of `DBMS_SCHEDULER.SET_ATTRIBUTE`, which has the following settings:
  - `PRIMARY`: The job runs only when the database is in the role of the primary database.
  - `LOGICAL STANDBY`: The job runs only when the database is in the role of a logical standby.

# Quiz

- By default, users are able to create additional indexes and materialized views in a logical standby database.
  - a.True
  - b.False

# Quiz

- Scheduler jobs created on the primary database are replicated to the logical standby database, but do not execute by default.
  - a.True
  - b.False

# Summary

- In this lesson, you should have learned how to:
  - Determine when to create a logical standby database
  - Create a logical standby database
  - Manage SQL Apply filtering

# Practice 8: Overview

- This practice covers the following topics to create a logical standby database:
  - Identify Unsupported Objects for Logical Standbys
  - Create a Logical Standby (Temporarily a Physical)
  - Start Redo Transport and Verify Operation
  - Convert Physical Standby to Logical Standby

# **9. Oracle Data Guard Broker: Overview**

# Objectives

- After completing this lesson, you should be able to describe:
  - The Data Guard broker architecture
  - Data Guard broker components
  - Benefits of the Data Guard broker
  - Data Guard broker configurations
  - How to use Enterprise Manager to manage your Data Guard configuration
  - How to invoke DGMGRL to manage your Data Guard configuration

# Oracle Data Guard Broker: Features

- The Oracle Data Guard broker is a distributed management framework.
- The broker automates and centralizes the creation, maintenance, and monitoring of Data Guard configurations.
- With the broker, you can perform all management operations locally or remotely with easy-to-use interfaces:
  - Oracle Enterprise Manager Cloud Control
  - DGMGRL (a command-line interface)

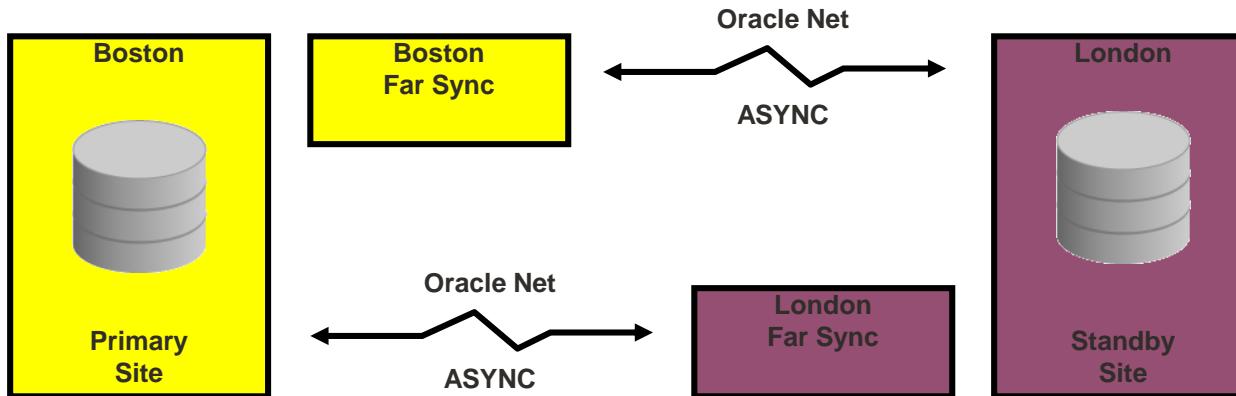
# Data Guard Broker: Components

- Client-side:
  - Oracle Enterprise Manager Cloud Control
  - DGMGRL (command-line interface)
- Server-side: Data Guard monitor
  - DMON process
  - Configuration files

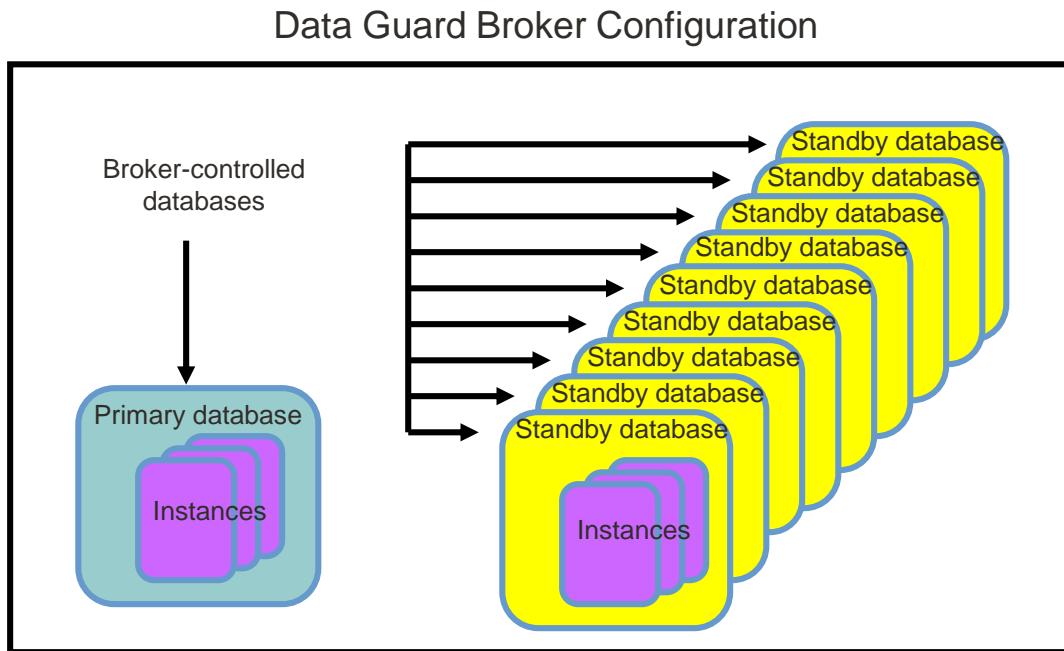


# Data Guard Broker: Configurations

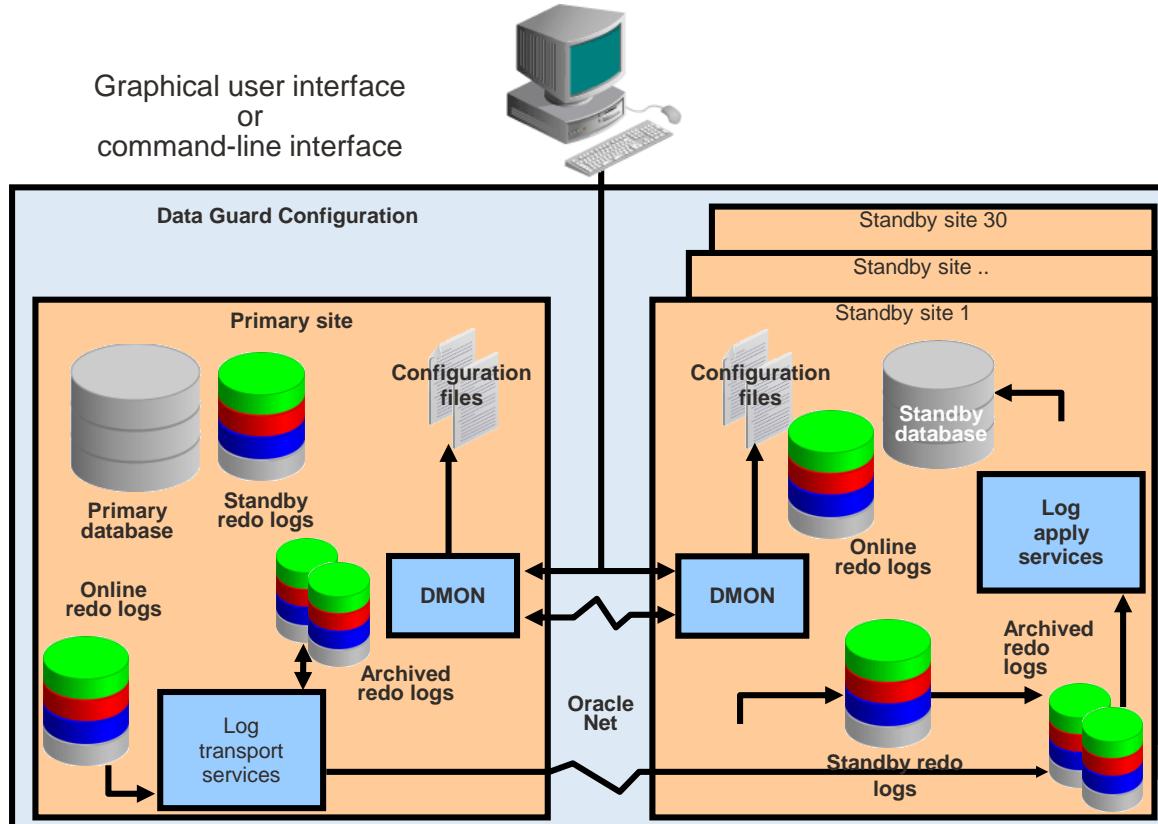
- The most common configuration is a primary database at one location and a standby database at another location.



# Data Guard Broker: Management Model



# Data Guard Broker: Architecture



# Data Guard Monitor: DMON Process

- A server-side background process
- A part of each database instance in the configuration
- Created when you start the broker
- Performs requested functions and monitors the resource
- Communicates with other DMON processes in the configuration
- Updates the configuration file
- Creates the `drc<SID>` trace file in the location set by the `DIAGNOSTIC_DEST` initialization parameter
- Modifies initialization parameters during role transitions as necessary

# Benefits of Using the Data Guard Broker

- Enhances the high-availability, data protection, and disaster protection capabilities inherent in Oracle Data Guard by automating both configuration and monitoring tasks
- Streamlines the process for any one of the standby databases to replace the primary database and to take over production processing
- Enables easy configuration of additional standby databases
- Provides simplified, centralized, and extended management
- Automatically communicates between the databases in a Data Guard configuration by using Oracle Net Services
- Provides built-in validation that monitors the health of all databases in the configuration

# Comparing Configuration Management With and Without the Data Guard Broker

	<b>With the Broker</b>	<b>Without the Broker</b>
General	Manage databases as one	Manage databases separately
Creation of the standby database	Use Enterprise Manager (EM) Cloud Control wizards	Manually create files
Configuration and management	Configure and manage from single interface	Set up services manually for each database
Monitoring	<ul style="list-style-type: none"><li>• Monitor continuously</li><li>• Unified status and reports</li><li>• Integrate with EM events</li></ul>	Monitor each database individually through views
Control	Invoke role transitions with a single command	Coordinate sequences of multiple commands across database sites for role transitions

# Data Guard Broker Interfaces

- Command-line interface (CLI):
  - Is started by entering `DGMGRl` at the command prompt where the Oracle server or an Oracle client is installed
  - Enables you to control and monitor a Data Guard configuration from the prompt or in scripts
- Oracle Enterprise Manager Cloud Control:
  - Provides wizards to simplify creating and managing standby databases

# Broker Controlled Database Initialization Parameters

- The broker controls the following parameters:

- LOG\_ARCHIVE\_DEST\_n
- LOG\_ARCHIVE\_DEST\_STATE\_n
- ARCHIVE\_LAG\_TARGET
- DB\_FILE\_NAME\_CONVERT
- LOG\_ARCHIVE\_FORMAT
- LOG\_ARCHIVE\_MAX\_PROCESSES
- LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST
- LOG\_ARCHIVE\_TRACE
- LOG\_FILE\_NAME\_CONVERT
- STANDBY\_FILE\_MANAGEMENT
- MAX\_EVENTS\_RECORDED
- MAX\_SERVERS
- MAX\_SGA
- PRESERVE\_COMMIT\_ORDER

# Using the Command-Line Interface of the Data Guard Broker

```
DGMGRL> connect sysdg/password
Connected.
DGMGRL> show configuration
Configuration - DRSolution
  Protection Mode: MaxPerformance
  Databases:
    boston      - Primary database
    bostonFS   - Far Sync
    london     - Physical standby database
    london2    - Logical standby database
    londonFS  - Far Sync (inactive)

  Fast-Start Failover: DISABLED

  Configuration Status:
    SUCCESS
```

# Using Oracle Enterprise Manager Cloud Control

The screenshot shows the Oracle Enterprise Manager Cloud Control dashboard. The top navigation bar includes links for Oracle Database, Performance, Availability (which is selected), Security, Schema, and Administration. A message at the top right indicates the page was refreshed on May 31, 2020, at 3:37:23 PM EDT.

The main content area displays several key metrics:

- Version:** 19.3.0.0.0
- Up Time:** 0 days, 2 hrs
- Availability for Last 7 Days:** 100%
- Last Backup:** N/A

A sidebar on the left provides quick access to Load and Capacity, Incidents and Compliance, and Recommendations. The Load and Capacity section shows 0.00 Average Active Sessions and N/A Used Space (GB). The Incidents and Compliance section shows 0 errors, 0 warnings, and 0 flags, with a note that Compliance Not Configured.

The central area features a chart titled "Active Sessions" over time, with a red line indicating CPU Cores. The chart shows spikes in activity around 3:17 PM, 3:27 PM, and 3:37 PM, primarily driven by Wait and User I/O.

A dropdown menu under the Availability tab lists several options: MAA Advisor, Backup & Recovery, Add Standby Database, Data Guard Administration (which is highlighted with a cursor), Data Guard Performance, and Verify Data Guard Configuration.

A callout box at the bottom of the slide instructs users to "Select ‘Data Guard Administration’ to access the Data Guard pages."

# Data Guard Overview Page

**boston.example.com (Container Database)** ⓘ

Logged in as sys ⓘ | host01.example.com

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

## Data Guard

Page Refreshed May 31, 2020 4:43:47 PM EDT

View Data | Real Time: 30 Second Refresh ⓘ

⟳

### Overview

Data Guard Status: ✓ Normal  
Protection Mode: Maximum Performance  
Fast-Start Failover: Disabled

### Primary Database

Name: boston  
Host: host01.example.com  
Data Guard Status: ✓ Normal  
Current Log: 30  
Properties: Edit

### Standby Database Progress Summary

Transport lag is the time difference between the last update on the primary database and the last received redo on the standby database.  
Apply lag is the time difference between the last update on the primary database and the last applied redo on the standby database.

Category	Value (minutes)
Transport Lag	2.1
Apply Lag	3

### Standby Databases

Add Far Sync | Add Standby Database

Edit | Remove | Switchover | Failover | Convert

Select	Name	Host	Data Guard Status	Role	Redo Source	Real-time Query	Last Received Log	Last Applied Log	Estimated Failover Time
<input checked="" type="radio"/>	london	host03.example.com	✓ Normal	Physical Standby	boston	✓ Enabled	29	29	< 1 second

# Benefits of Using Enterprise Manager

- Enables you to manage your configuration by using a familiar interface and event-management system
- Automates and simplifies the complex operations of creating and managing standby databases through the use of wizards
- Performs all Oracle Net Services configuration changes that are necessary to support redo transport services and log apply services
- Provides a verify operation to ensure that redo transport services and log apply services are configured and functioning properly
- Enables you to select a new primary database from a set of viable standby databases

# Quiz

- Which tools are used to interface with the Data Guard Broker?
  - a.DGMGRL
  - b.SRVCTL
  - c.Enterprise Manager
  - d.CRSCTL

# Quiz

- It is necessary to manage standby databases with the Data Guard Broker framework.
  - a.True
  - b.False

# Summary

- In this lesson, you should have learned how to:
  - Describe the Data Guard broker architecture
  - Describe Data Guard broker components
  - Describe benefits of the Data Guard broker
  - Describe Data Guard broker configurations
  - Use Enterprise Manager to manage your Data Guard configuration
  - Invoke DGMGRL to manage your Data Guard configuration

# 10. Creating a Data Guard Broker Configuration

# Objectives

- After completing this lesson, you should be able to use DGMGRL commands to:
  - Create a Data Guard broker configuration
  - Manage the Data Guard broker configuration
  - List the new Data Guard Broker commands

# Data Guard Broker: Requirements

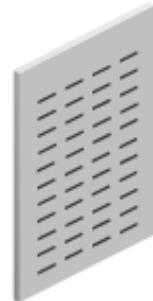
- Oracle Database Enterprise Edition
- Single-instance or multi-instance environment
- COMPATIBLE parameter: Set to 19.1 or later for primary and standby databases to take advantage of new 19.1 features (Optional).
- Oracle Net Services network files: Must be configured for the primary database and any 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

# 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 SERVICE attribute before creating a broker configuration

# Data Guard Broker and the SPFILE

- You must use a server parameter file (SPFILE) for initialization parameters.
- Using the SPFILE enables the Data Guard broker to keep its configuration file and the database SPFILE consistent.
- If you use the broker, use Enterprise Manager Cloud Control or DGMGRL to update database parameter values.



# Data Guard Monitor: Configuration File

- The broker configuration file is:
  - Automatically created and named using a default path name and file name when the broker is started
  - Managed automatically by the `DMON` process
- The configuration file and a copy are created at each managed site with default names:
  - `dr1<db_unique_name>.dat`
  - `dr2<db_unique_name>.dat`
- Configuration file default locations are operating system-specific:
  - Default location for UNIX and Linux: `ORACLE_HOME/dbs`
  - Default location for Windows: `ORACLE_HOME\database`
- Use `DG_BROKER_CONFIG_FILEn` to override the default path name and file name.

# Data Guard Broker: Log Files

- The broker log files contain information recorded by the DMON process.
- There is one file for each instance in the broker configuration.
- Broker log files are created in the same directory as the alert log and are named `drc<$ORACLE_SID>.log`.

# Creating a Broker Configuration

1. Clear existing `LOG_ARCHIVE_DEST_n` entries on every instance that references network locations.
2. Invoke DGMGRL and connect to the primary database.
3. Define the configuration, including a profile for the primary database.
4. Add standby databases to the configuration.
5. Add Far Sync instances to the configuration.
6. Enable the configuration, including the databases.
7. Define routing rules for the configuration.

# Clear Redo Transport Network Locations on Primary

- 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=bostonFS SYNC REOPEN=15
                     valid_for=(ONLINE_LOGFILES,
                     PRIMARY_ROLE) db_unique_name =
                     bostonFS
log_archive_dest_3   SERVICE=london2 SYNC REOPEN=15
                     valid_for=(ONLINE_LOGFILES,
                     PRIMARY_ROLE) db_unique_name =
                     london2
SQL> alter system log_archive_dest_2='' scope=both;
SQL> alter system log_archive_dest_3='' scope=both;
```

# 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@boston
```

- Adding a database user to the password file

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

# Defining the Broker Configuration and the Primary Database Profile

```
DGMGRL> CREATE CONFIGURATION 'DRSolution' AS PRIMARY DATABASE IS
      'boston' CONNECT IDENTIFIER IS boston;
Configuration "DRSolution" created with primary database "boston"

DGMGRL> show configuration
Configuration - DRSolution

  Protection Mode: MaxPerformance
  Databases:
    boston - Primary database

  Fast-Start Failover: DISABLED

  Configuration Status:
  DISABLED
```

# Adding a Standby Database to the Configuration

```
DGMGRL> add database 'london' as connect identifier is london;
Database "london" added

DGMGRL> show configuration
Configuration - DRSolution

  Protection Mode: MaxPerformance
  Databases:
    boston   - Primary database
    london   - Physical standby database

  Fast-Start Failover: DISABLED

  Configuration Status:
  DISABLED
```

# Adding a Far Sync to the Configuration

```
DGMGRL> add far_sync 'bostonFS' as connect identifier is bostonFS;
far sync instance "bostonFS" added

DGMGRL> show configuration
Configuration - DRSSolution

Protection Mode: MaxPerformance
Databases:
boston      - Primary database
  bostonFS - Far Sync (inactive)
  london    - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
DISABLED
```

# Enabling the Configuration

```
DGMGRL> ENABLE CONFIGURATION;
Enabled.

DGMGRL> SHOW CONFIGURATION
Configuration - DRSolution

Protection Mode: MaxPerformance
Databases:
  boston      - Primary database
  bostonFS   - Far Sync (inactive)
  london      - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

# Broker Support for Complex Redo Routing

- By default, a primary database sends its redo to every possible redo transport destination in a broker configuration.
  - The `RedoRoutes` broker property allows more complex routing to be defined.
    - Supports cascaded configurations
    - Supports Far Sync configurations
    - Supports real-time cascading and without real-time cascading
    - Supports rules dependent on which database is the current primary
    - Supports the new Fast Sync configuration

# Defining RedoRoutes by Using DGMGRL

- The RedoRoutes property is set to a character string that contains one or more redo routing rules.

```
DGMGRL> EDIT DATABASE <db_unique_name>
      SET PROPERTY 'RedoRoutes' = '(redo routing rule 1)(redo routing rule n)';
```

- Each rule contains one or more redo sources and one or more redo destinations, separated by a colon.
- The redo source field must contain the LOCAL keyword or a comma-separated list of DB\_UNIQUE\_NAME values.
- The redo destination field must contain the keyword ALL or a comma-separated list of database unique names, each of which can be followed by an optional redo transport attribute.

```
(redo source : redo destination) (LOCAL : redo destination ATTRIBUTE)
(redo source : redo destination, redo destination ATTRIBUTE)
```

# RedoRoutes Usage Guidelines

- The RedoRoutes property has a default value of NULL, which is treated as (LOCAL : ALL).
- A redo routing rule is active if its redo source field specifies the current primary database.
- If a rule is active, primary database redo is sent by the database at which the rule is defined to each destination specified in the redo destination field of that rule.
- The ASYNC redo transport attribute must be explicitly specified for a cascaded destination to enable real-time cascading to that destination.
- The RedoRoutes property cannot be set on a logical or snapshot standby database.

# How to Read Redo Routing Rules

If boston is the current primary database, the rule is active

```
DGMGRL> EDIT DATABASE bostonFS SET PROPERTY 'RedoRoutes' = '(boston:london  
ASYNC)';
```

Which causes bostonFS to ship redo to london

# Far Sync Example with RedoRoutes

- An example Far Sync configuration for a Maximum Availability broker configuration:
  - Primary database in Boston, MA
  - Far Sync in Cambridge, MA
  - Physical standby in Chicago, IL.
  - Far Sync in Shaumburg, IL. (for role reversal)

```
DGMGRL> EDIT DATABASE Boston SET PROPERTY 'RedoRoutes' = '(LOCAL : Cambridge SYNC)';  
DGMGRL> EDIT FAR_SYNC Cambridge SET PROPERTY 'RedoRoutes' = '(Boston : Chicago ASYNC)';  
DGMGRL> EDIT DATABASE Chicago SET PROPERTY 'RedoRoutes' = '(Local : Shaumburg SYNC)';  
DGMGRL> EDIT FAR_SYNC Shaumburg SET PROPERTY 'RedoRoutes' = '(Chicago : Boston ASYNC)';
```

# Changing Database Properties and States

- To alter the state of only the primary database:

```
DGMGRIL> EDIT DATABASE boston SET STATE='TRANSPORT-OFF' ;
```

- To alter the state of a physical or logical standby database:

```
DGMGRIL> EDIT DATABASE london SET STATE='APPLY-OFF' ;
```

- To alter a configurable database property:

```
DGMGRIL> EDIT DATABASE boston SET PROPERTY LogXptMode='SYNC' ;
```

- Note that if a database receives redo from a database or far sync instance where the RedoRoutes property has been configured with a redo transport mode, then the mode specified by that RedoRoutes property value overrides the value of the LogXptMode property.

# Managing Redo Transport Services by Using DGMGRL

- Specify database properties to manage redo transport services:

Property	Purpose
<b>ApplyLagThreshold</b>	Generates a warning status for a logical or physical standby when the member's apply lag exceeds the value specified by the property
<b>ArchiveLagTarget</b>	Limits the amount of data that can be lost and, effectively, increases the availability of the standby database or Far Sync instance by forcing a log switch after the amount of time you specify (in seconds) elapses
<b>Binding</b>	Specifies whether the redo destination is MANDATORY or OPTIONAL
<b>DGConnectIdentifier</b>	Specifies the connection identifier the broker uses when making connections to a configuration member
<b>LogArchiveFormat</b>	Specifies the format for filenames of archived redo log files using a database ID (%d), thread (%t), sequence number (%s), and resetlogs ID (%r)
<b>LogArchiveMaxProcesses</b>	Specifies the initial number of archiver processes (ARCn) that are invoked

# Managing Redo Transport Services by Using DGMGRL

- Specify database properties to manage redo transport services:

Property	Purpose
<b>LogArchiveMinSucceedDest</b>	Controls when online redo log files are available for reuse provided archiving has succeeded to a minimum number of destinations
<b>LogShipping</b>	Specifies the ENABLE and DEFER values for the LOG_ARCHIVE_DEST_STATE_n initialization parameter of the database or Far Sync instance that is sending redo data
<b>LogXptMode</b>	Specifies the redo transport services on each configuration member to SYNC, ASYNC, or FASTSYNC
<b>MaxFailure</b>	Specifies the maximum number of contiguous archiving failures before the redo transport services stop trying to transport archived redo log files to the standby database
<b>NetTimeout</b>	Specifies the number of seconds the LGWR waits for Oracle Net Services to respond to a LGWR request and is used to bypass the long connection timeout in TCP/IP

# Managing Redo Transport Services by Using DGMGRL

- Specify database properties to manage redo transport services:

Configurable Property	Purpose
<code>OnlineAlternateLocation</code>	Specifies an alternate online redo log archive location for primary, logical, and snapshot standby databases. A valid <code>LOG_ARCHIVE_DEST_n</code> parameter string with the <code>LOCATION</code> attribute specified, but no <code>VALID_FOR</code> , <code>ALTERNATE</code> , or <code>SERVICE</code> attributes
<code>OnlineArchiveoLocation</code>	Specifies the online redo log archive location for primary, logical, and snapshot standby databases. A valid <code>LOG_ARCHIVE_DEST_n</code> string with the <code>LOCATION</code> attribute specified
<code>RedoCompression</code>	Specifies whether redo data is transmitted to a standby database or Far Sync instance in compressed or uncompressed form
<code>ReopenSecs</code>	Specifies the minimum number of seconds before the archiver process (ARCn, foreground, or log writer process) should try again to access a previously failed destination

# Managing Redo Transport Services by Using DGMGRL

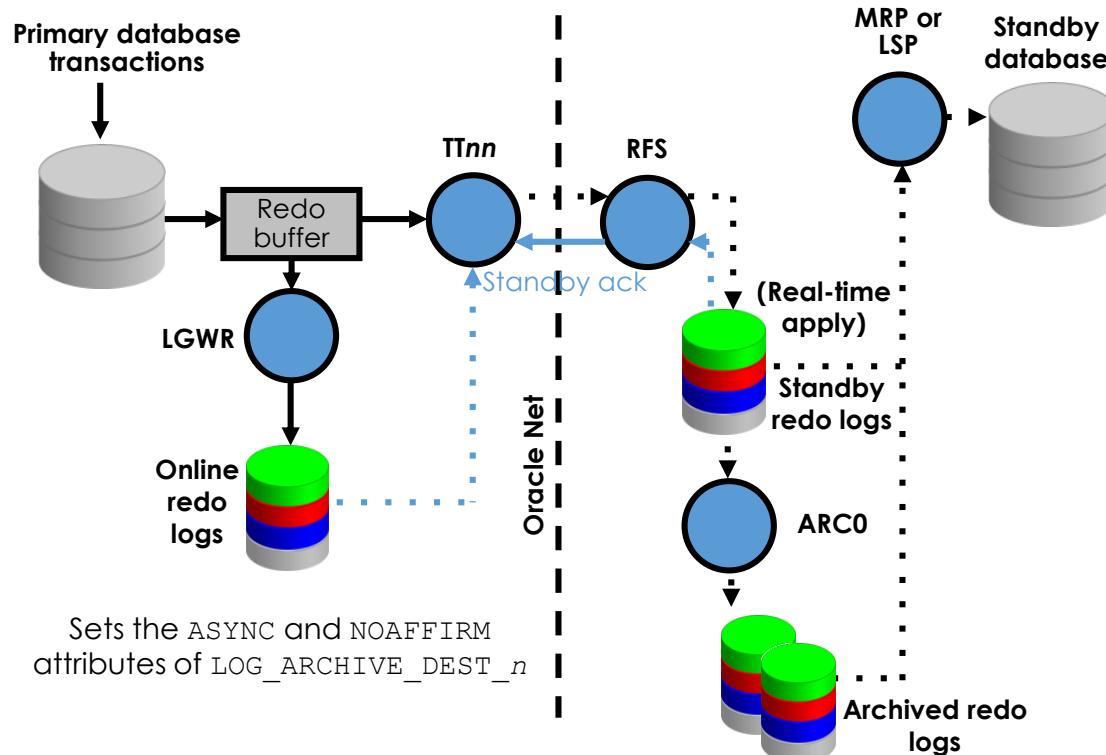
- Specify database properties to manage redo transport services:

Configurable Property	Purpose
<code>StandbyAlternateLocation</code>	Specifies an alternate disk location to store the archived redo log files in the standby when the location specified by the <code>StandbyArchiveLocation</code> configurable property fails
<code>StandbyArchiveLocation</code>	Specifies the location of archived redo log files arriving from a redo source
<code>TransportDisconnectedThreshold</code>	Used to generate a warning status for a logical, physical, or snapshot standby or a Far Sync instance when the last communication from the primary database exceeds the value specified in seconds
<code>TransportLagThreshold</code>	Used to generate a warning status for a logical, physical, or snapshot standby or a Far Sync instance when the member's transport lag exceeds the value specified in seconds

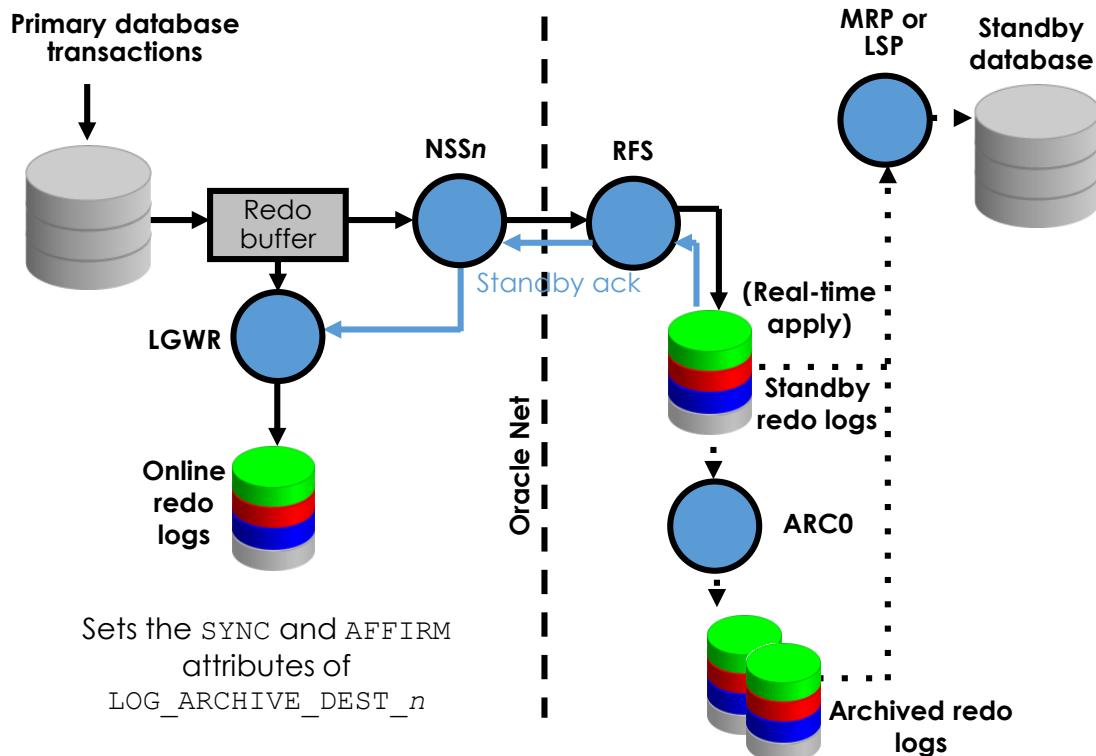
# Managing the Redo Transport Service by Using the LogXptMode Property

- The redo transport service must be set up for the chosen data protection mode.
- Use the `LogXptMode` property to set the redo transport services:
  - `ASYNC`: Sets the `ASYNC` and `NOAFFIRM` attributes of `LOG_ARCHIVE_DEST_n`
  - `SYNC`: Sets the `SYNC` and `AFFIRM` attributes of `LOG_ARCHIVE_DEST_n`
  - `FASTSYNC`: Sets the `SYNC` and `NOAFFIRM` attributes of `LOG_ARCHIVE_DEST_n`

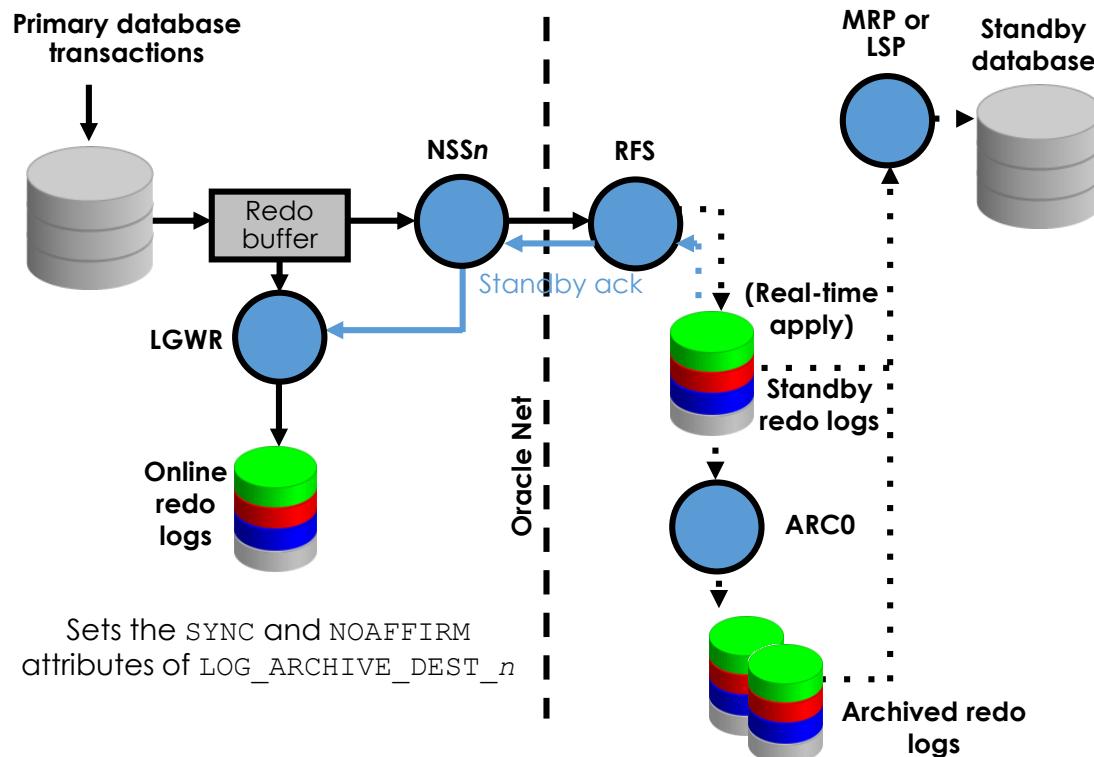
# Setting LogXptMode to ASYNC



# Setting LogXptMode to SYNC



# Setting LogXptMode to FASTSYNC



# New Commands to Set, Modify, and Display the Initialization Parameters:

- New commands are available to set, modify, and display the value of database initialization parameters:
  - EDIT DATABASE PARAMETER
  - EDIT DATABASE RESET PARAMETER
  - EDIT FAR\_SYNC RESET PARAMETER
  - EDIT RECOVERY\_APPLIANCE PARAMETER
  - EDIT RECOVERY\_APPLIANCE RESET PARAMETER
  - SET TRACE\_LEVEL

# The EDIT DATABASE SET PARAMETER Command

- The EDIT DATABASE SET PARAMETER command sets the specified initialization parameter for the named database.
- The following example edits the initialization parameter `log_archive_trace` for the database named `boston` and sets its value to 1.
  - The `SCOPE` setting specifies that the parameter must be changed in both the memory and in the database initialization parameter file.

```
DGMGRL> EDIT DATABASE 'boston'  
SET PARAMETER log_archive_trace = 1 'SCOPE = BOTH';
```

- The database must be available when this command is run.

# The EDIT DATABASE RESET PARAMETER Command

- The EDIT DATABASE RESET PARAMETER command resets the specified database initialization parameter for the named database to its default value.
- The following example shows how to reset the log\_archive\_trace parameter for the database named boston.

```
DGMGRL> EDIT DATABASE 'boston' RESET PARAMETER log_archive_trace;  
Succeeded.
```

# The EDIT DATABASE FAR\_SYNC RESET\_PARAMETER Command

- The EDIT FAR\_SYNC RESET\_PARAMETER command resets the specified database initialization parameter for the named far sync instance to its default value.
- The example shows how to reset the log\_filename\_convert initialization parameter to its default value for the far sync instance named bostonFS:

```
DGMGRL> EDIT FAR_SYNC 'bostonFS' RESET PARAMETER log_filename_convert;
```

# The EDIT RECOVERY\_APPLIANCE SET PARAMETER Command

- The EDIT RECOVERY\_APPLIANCE SET PARAMETER command sets the specified initialization parameter for the named Zero Data Loss Recovery Appliance.
- The following example edits the initialization parameter for the Recovery Appliance EnterpriseRecoveryAppliance:

```
DGMGRL> EDIT RECOVERY_APPLIANCE EnterpriseRecoveryAppliance  
      SET PARAMETER log_archive_trace = 1 SCOPE='spfile';
```

# The EDIT RECOVERY\_APPLIANCE RESET PARAMETER Command

- The EDIT RECOVERY\_APPLIANCE RESET PARAMETER command resets the specified database initialization parameter for the named Recovery Appliance to its default value.
- The example below shows how to reset the db\_filename\_convert parameter to its default value for the Recovery Appliance named EnterpriseRecoveryAppliance:

```
DGMGRL> EDIT RECOVERY APPLIANCE 'EnterpriseRecoveryAppliance'  
      RESET PARAMETER db_filename_convert;
```

# The SET TRACE\_LEVEL Command

- The SET TRACE\_LEVEL command sets the amount of tracing done by DGMGRL.
- This is a client-side setting and does not impact the tracing set for the broker within the Oracle Database.
- Set trace level to USER to limit the amount of tracing information stored.
  - This is the default setting and includes information about fast-start failover, status changes of the primary and target standby database, and error or warning messages.
- Set trace level to SUPPORT to increase the amount of tracing information to include lower-level information needed by Oracle Support Services.

# Example: SET TRACE\_LEVEL command

- Setting DGMGRL and Observer Tracing Levels
  - The SET TRACE LEVEL command is used to modify the trace level for DGMGRL to SUPPORT.

```
DGMGRL> SET TRACE_LEVEL SUPPORT;
```

- When the observer is started, use the TRACE\_LEVEL clause to set the observer's trace level to USER.

```
DGMGRL> START OBSERVER TRACE_LEVEL is USER;
```

- If you omit the TRACE\_LEVEL clause in the START OBSERVER command, the observer is started using the same trace level setting as DGMGRL, SUPPORT.

# Example: SET TRACE\_LEVEL command

- Setting the DGMGRL and Database Tracing Levels to Different Values
  - The example below sets the DGMGRL trace level to SUPPORT.

```
DGMGRL> SET TRACE_LEVEL support;
```

- The EDIT CONFIGURATION command below is used to set the trace level of the Oracle Database to USER.

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY TraceLevel = USER;
```

- The server-side trace level changes to USER while DGMGRL's trace level is still SUPPORT.

# The EDIT FAR\_SYNC SET PARAMETER Command

- The following example resets the initialization parameter of a far sync instance named london.

```
DGMGRL> DISABLE FAR_SYNC 'londonFS';
DGMGRL> EDIT FAR_SYNC 'londonFS' SET log_archive_trace=1;
DGMGRL> ENABLE FAR_SYNC 'londonFS';
```

# Data Guard Broker Deprecated and Desupported Properties

- These properties associated with initialization parameters are deprecated:
  - ArchiveLagTarget
  - DataGuardSyncLatency
  - LogArchiveMaxProcesses, LogArchiveMinSucceedDest, LogArchiveTrace, LogArchiveFormat, LogFileNameConvert
  - StandbyFileManagement
  - DbFileNameConvert
- Use the EDIT DATABASE...SET PARAMATER command to modify the database initialization parameters that correspond to the deprecated properties.

```
DGMGRL> EDIT DATABASE 'boston'  
SET PARAMETER log_archive_trace = 1 'SCOPE = BOTH';
```

# Data Guard Broker Deprecated and Desupported Properties

- The following properties related to logical standbys are deprecated:
  - LsbyMaxEventsRecorded, LsbyMaxServers, LsbyMaxSga, LsbyPreserveCommitOrder, , LsbyRecordAppliedDdl
  - LsbyRecordSkipDdl, LsbyRecordSkipErrors, and LsbyParameter.
- Use the DBMS\_LOGSTDBY.APPLY\_SET procedure to configure properties related to logical standbys.

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE', FALSE);
```

- The MaxConnections configuration property is desupported.

# Export Broker Configuration

- The EXPORT CONFIGURATION command enables you to save the metadata contained in the broker configuration file to a text file.
- This command to maintain an up-to-date copy of the broker configuration metadata.
- The command format is as follows: EXPORT CONFIGURATION [TO *file-name*] ;
- The broker stores the exported configuration in the trace directory.
- The following example exports the metadata in the broker configuration file to a file named `dg_config.txt` in the trace directory.

```
DGMGRl> EXPORT CONFIGURATION TO 'dg_config.txt' ;
```

# Import Broker Configuration

- The IMPORT CONFIGURATION command allows you to import the broker configuration metadata that was previously exported using the EXPORT CONFIGURATION command.
- The command format is as follows: IMPORT CONFIGURATION [FROM *file-name*] ;
- The imported metadata is stored in the in-memory metadata and either of the metadata files specified by DG\_BROKER\_CONFIG\_FILE1 or DG\_BROKER\_CONFIG\_FILE2.
- The following command imports configuration metadata stored in the file named dg\_config.txt in the trace directory into the memory and to the broker metadata file.

```
DGMGRL> IMPORT CONFIGURATION FROM 'dg_config.txt';
```

# Disabling Broker Management of the Configuration or Standby Database

- Disable broker management of the configuration:

```
DGMGRL> DISABLE CONFIGURATION;
```

- Disable broker management of a standby database:

```
DGMGRL> DISABLE DATABASE 'london' ;
```

- Disable broker management of a Far Sync:

```
DGMGRL> DISABLE FAR_SYNC 'londonFS' ;
```

# Removing the Configuration or Standby Database

- Remove a standby database from the configuration:

```
DGMGRL> REMOVE DATABASE 'london2' [PRESERVE DESTINATIONS];
```

- Remove a Far Sync from the configuration

```
DGMGRL> REMOVE FAR_SYNC 'londonFS';
```

- Remove the configuration:

```
DGMGRL> REMOVE CONFIGURATION [PRESERVE DESTINATIONS];
```

# Quiz

- How many Data Guard broker configuration files are created for each database unique name?
  - a.One
  - b.Two
  - c.Three
  - d.Four

# Quiz

- The EDIT DATABASE ... SET PARAMETER command is to set the database property for the named database.
  - a.True
  - b.False

# Summary

- In this lesson, you should have learned how to use DGMGRL commands to:
  - Create a Data Guard broker configuration
  - Manage the Data Guard broker configuration
  - List the new Data Guard Broker commands

# Practice 10: Overview

- This practice covers the following topics:
  - Establishing Local and Remote Connections with DGMGRL
  - Creating and Enabling a Data Guard Broker Configuration
  - Verifying and Examining the Data Guard Environment Using Enterprise Manager

# 11. Monitoring a Data Guard Broker Configuration

# Objectives

- After completing this lesson, you should be able to:
  - Use Enterprise Manager Cloud Control to monitor the configuration
  - Use DGMGRL to monitor the configuration
  - List the new Data Guard Broker VALIDATE command
  - Use SQL to monitor the configuration

# Monitoring the Data Guard Configuration by Using Enterprise Manager Cloud Control

- On the Data Guard Overview page, you can:
  - View the Standby Progress Summary graph that shows the transport lag and the apply lag
  - Access additional performance and configuration information
    - Performance Overview page: Information about data archived and applied, standby database progress, and log services
    - Log File Details page: Information about the primary database online redo log file
  - Perform a Verify operation

# Viewing the Data Guard Configuration Status

**boston.example.com (Container Database)** ⓘ

Logged in as sys ⓘ | host01.example.com

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

## Data Guard

Page Refreshed May 31, 2020 4:43:47 PM EDT

View Data Real Time: 30 Second Refresh ⏪

### Overview

Data Guard Status: ✓ Normal  
Protection Mode: Maximum Performance  
Fast-Start Failover: Disabled

### Primary Database

Name: boston  
Host: host01.example.com  
Data Guard Status: ✓ Normal  
Current Log: 30  
Properties: Edit

### Standby Database Progress Summary

Transport lag is the time difference between the last update on the primary database and the last received redo on the standby database.  
Apply lag is the time difference between the last update on the primary database and the last applied redo on the standby database.

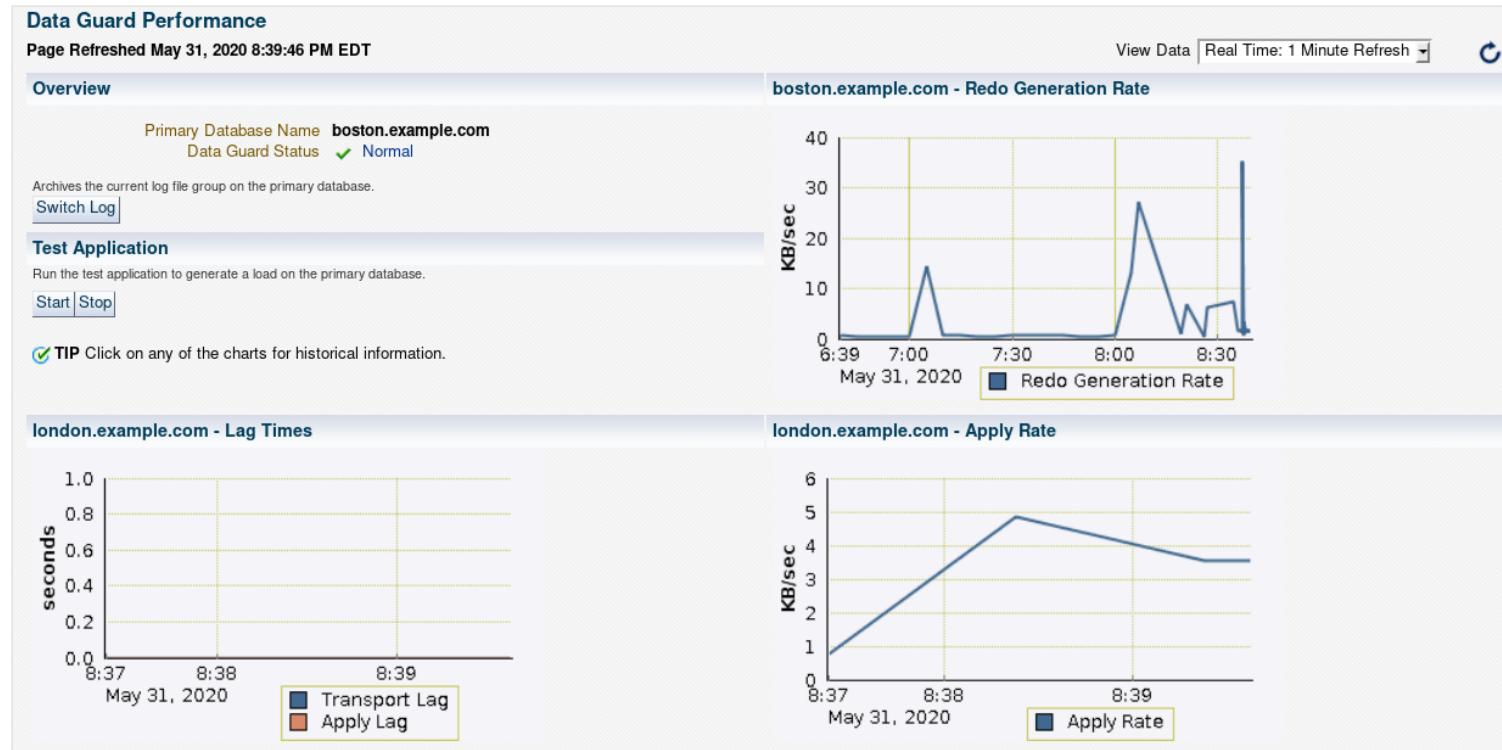
lag type	value (minutes)
Transport Lag	2.1
Apply Lag	3

### Standby Databases

Add Far Sync | Add Standby Database

Edit	Remove	Switchover	Failover	Convert					
Select	Name	Host	Data Guard Status	Role	Redo Source	Real-time Query	Last Received Log	Last Applied Log	Estimated Failover Time
<input checked="" type="radio"/>	london	host03.example.com	✓ Normal	Physical Standby	boston	✓ Enabled	29	29	< 1 second

# Monitoring Data Guard Performance



# Viewing Log File Details

**ORACLE® Enterprise Manager Cloud Control 13c**

boston.example.com (Container Database)

Logged in as sys host01.example.com

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

Data Guard > Log File Details

View Data | Real Time: Manual Refresh

Page Refreshed May 31, 2020 8:21:21 PM EDT

### Log File Details

A table is shown below for each standby database in the configuration, listing redo log files that have not been received by the standby database and redo log files that have been received but not applied to the standby database. Related redo transport and apply information is also displayed for diagnostic purposes.

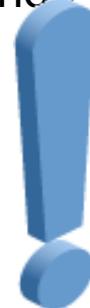
Primary Current Log 34

**london.example.com**

Redo Transport Services		On	Redo Apply Services		On		
Redo Transport Status		Normal			Status Normal		
				Apply Delay (minutes)	0		
Log	Status	ResetLogs ID #	First Change # (SCN)	Last Change # (SCN)	Size (KB)	Time Generated	Time Completed
33	Partially Applied	1041500351	2620726	2623093	1158	May 31, 2020 8:04:31 PM	May 31, 2020 8:19:04 PM
33	Not Received	1041500351	2620726	2623093	1158	May 31, 2020 8:04:31 PM	May 31, 2020 8:19:04 PM

# Enterprise Manager Metrics and Alerts

- **Metrics:** Units of measurement that are used to assess the health of your system
- **Thresholds:** Boundary values against which monitored metric values are compared
- **Alert:** An indication that is generated when a threshold is reached



# Data Guard Metrics

- Enterprise Manager provides Data Guard metrics:

The screenshot shows the Oracle Enterprise Manager Cloud Control 13c interface for managing a Container Database named 'boston.example.com'. The top navigation bar includes links for Home, Applications, Monitoring, Security, Schema, Administration, and SYSMAN, along with user information and a search bar.

The main content area displays a list of monitoring tasks under the 'Oracle Database' category. These tasks include:

- Alert Log Error Status: Disabled, Every 15 Minutes
- Archive Area: Disabled
- Database Monitoring User Privileges Check: Disabled
- Database Vault Configuration Issues - Command Rules: Disabled
- Database Vault Configuration Issues - Realms: Disabled
- Database Vault Policy Changes: Disabled
- Data Failure: Every 5 Minutes
- Data Guard Fast-Start Failover Observer**: Every 1 Minute

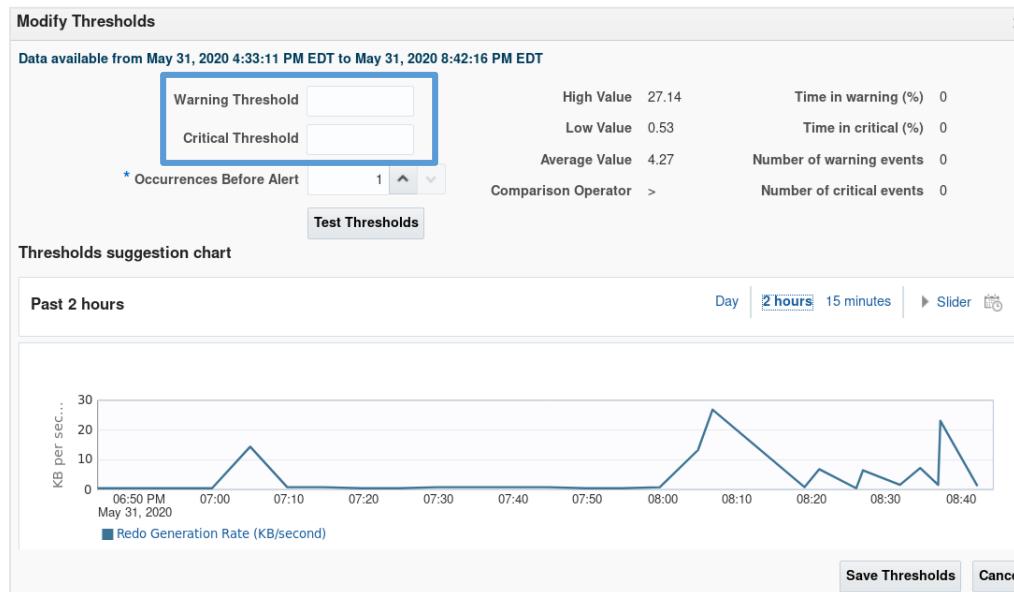
A specific section for 'Data Guard Status' is highlighted with a blue border. It contains two rows:

Observer Status	Contains	Error	None	
Data Guard Status	Contains	Warning	Error	None

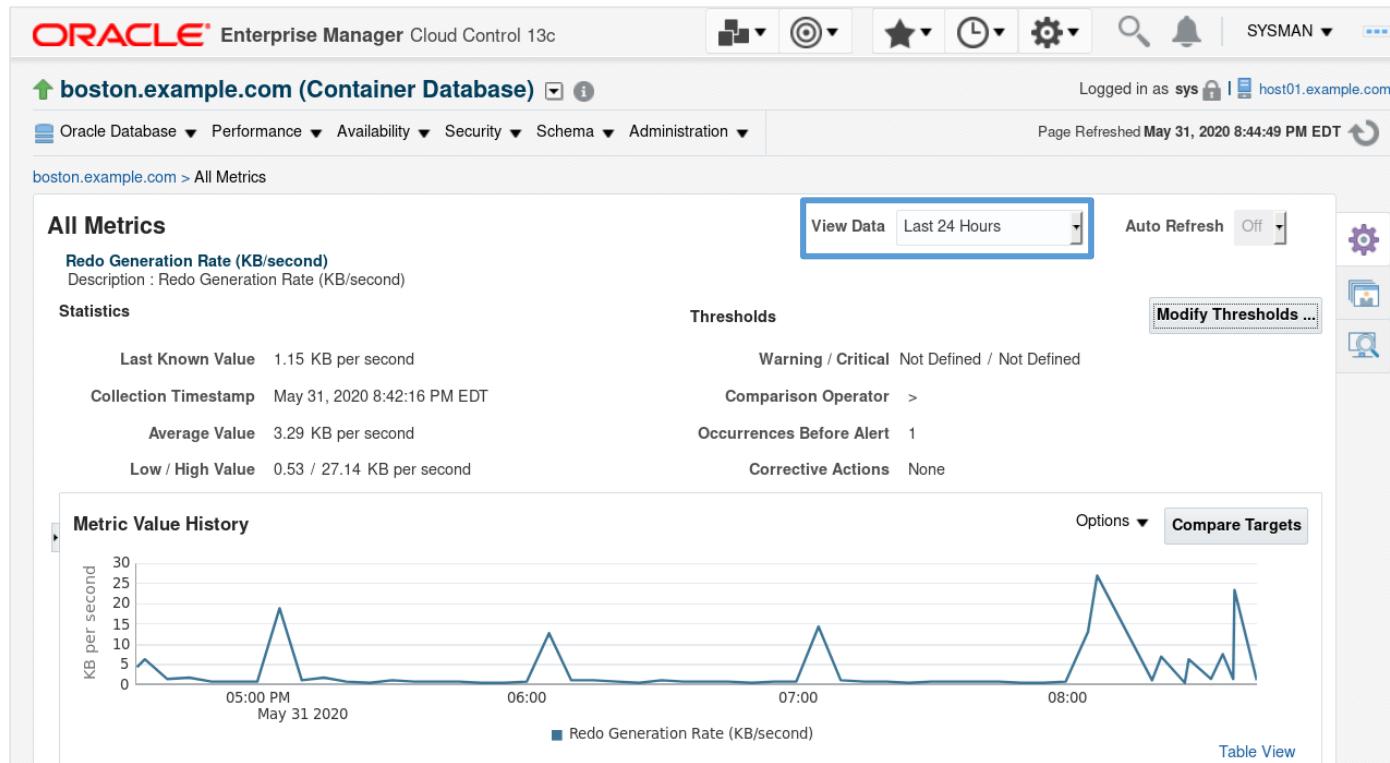
Icons for edit and delete are visible next to the status rows.

# Managing Data Guard Metrics

1. Configure notification methods.
2. View the All Metrics page.
3. Set or change Data Guard metric thresholds.



# Viewing Metric Value History



# Viewing Data Guard Diagnostic Information

- The Data Guard broker records information in:
  - Oracle database alert log files
  - Data Guard broker log files
- Database status information is available by issuing the SHOW DATABASE command.
- Use the following database properties to obtain additional information:
  - StatusReport
  - LogXptStatus
  - InconsistentLogXptProps
  - LsbyFailedTxnInfo

```
DGMGR> SHOW DATABASE london 'RecvQEntries' ;
```

- [Recv/Send] QEntries
- TopWaitEvents

# Using Monitorable Database Properties to Identify a Failure

1. Check the configuration status:

```
DGMGRL> SHOW CONFIGURATION;
```

1. Check the database status:

```
DGMGRL> SHOW DATABASE london;
```

1. Check a Far Sync status:

```
DGMGRL> SHOW FAR_SYNC 'bostonFS'
```

# Using the SHOW CONFIGURATION DGMGRL Command to Monitor the Configuration

- Use the SHOW CONFIGURATION command to display a brief description of the configuration status:

```
DGMGRL> show configuration
Configuration - DRSolution
  Protection Mode: MaxPerformance
  Databases:
    boston      - Primary database
      WARNING: ORA-16809: multiple warnings detected for
this database
    bostonFS   - Far Sync
    london     - Physical standby database
    london2    - Logical standby database
    londonFS  - Far Sync (inactive)

  Fast-Start Failover: DISABLED
  Configuration Status:
  WARNING
```

# Using the SHOW DATABASE VERBOSE DGMGRL Command to Monitor the Configuration

- Use the SHOW DATABASE VERBOSE command to display all property values for a database:

```
DGMGRL> show database verbose london
Database - london

Role: PHYSICAL STANDBY
Intended State: APPLY-ON
Transport Lag: 0 seconds (computed 1 second ago)
Apply Lag: 0 seconds (computed 1 second ago)
Apply Rate: 1.52 MByte/s
Real Time Query: OFF
Instance(s):
    london

Properties:
    DGConnectIdentifier          = 'london'
...
```

# The VALIDATE DATABASE DATAFILE Command

- The VALIDATE DATABASE DATAFILE command validates data files for both primary and standby databases.
- Data file validation detects lost writes in either database.
- You can specify a data file to be compared by data file name or data file number.

```
DGMGRL> VALIDATE DATABASE boston DATAFILE ALL OUTPUT=dbcomp.out;
Operation requires a connection to database "boston"
Connecting ...
Output files are created in /.../boston/trace on host "boston"
```

# New Data Guard Broker Commands:

## VALIDATE DATABASE SPFILE

- `VALIDATE DATABASE SPFILE` performs a comparison of `SPFILE` entries between the primary database and a specified standby database.

```
DGMGRL> VALIDATE DATABASE london SPFILE;
```

- Parameter file validation detects parameter value discrepancies between the primary and the specified standby database so that they can be rectified before a role change.
- Using this command frees you from having to restart a database to correct improperly set parameters.

# New Data Guard Broker Commands:

## VALIDATE NETWORK CONFIGURATION

- VALIDATE NETWORK CONFIGURATION performs network connectivity checks between members of a broker configuration.

```
DGMGRl> VALIDATE NETWORK CONFIGURATION FOR ALL;
```

- The connect identifier for each connectivity check is generated based on the DGConnectIdentifier property of the associated database.

# New Data Guard Broker Commands:

## VALIDATE STATIC CONNECT IDENTIFIER

- VALIDATE STATIC CONNECT IDENTIFIER validates the static connect identifier of a database.

```
DGMGRL> VALIDATE STATIC CONNECT IDENTIFIER FOR boston;
```

- To perform this validation, the broker makes a new connection to the database using a static connect identifier based on the StaticConnectIdentifier property.
- A new attribute, STATIC\_SERVICE=TRUE, is added to the connect identifier to ensure that a connection to the database is established using only a static service.

# Miscellaneous New Data Guard Broker Commands: SET ECHO and SHOW ALL

- SET ECHO controls whether or not to echo commands that are issued either at the command-line prompt or from a DGMGRL script.

```
SET ECHO [ON | OFF];
```

- SHOW ALL shows the values of DGMGRL CLI properties.

```
DGMGRL> SHOW ALL;  
debug ON  
echo OFF  
time OFF  
observerconfigfile = observer.ora
```

# Viewing Standby Redo Log Information in V\$LOGFILE

```
SQL> SELECT group#, member
  2  FROM v$logfile
  3  WHERE type = 'STANDBY';

GROUP# STATUS MEMBER
----- -----
 4          /u01/app/oracle/oradata/boston/onlinelog/group_4.278.711989145
 5          /u01/app/oracle/oradata/boston/onlinelog/group_5.279.711989151
 6          /u01/app/oracle/oradata/boston/onlinelog/group_6.280.711989159
 7          /u01/app/oracle/oradata/boston/onlinelog/group_7.281.711989165
```

# Viewing Standby Redo Log Information in V\$STANDBY\_LOG

```
SQL> SELECT group#, dbid, archived, status
  2  FROM v$standby_log;
```

GROUP#	DBID	ARC	STATUS
4	UNASSIGNED	NO	UNASSIGNED
5	3303427449	YES	ACTIVE
6	UNASSIGNED	YES	UNASSIGNED
7	UNASSIGNED	YES	UNASSIGNED

# Viewing Standby Redo Log Information in V\$LOGFILE

```
SQL> SELECT group#, member
  2  FROM v$logfile
  3  WHERE type = 'STANDBY';

GROUP# STATUS MEMBER
----- -----
 4          /u01/app/oracle/oradata/boston/onlinelog/group_4.278.711989145
 5          /u01/app/oracle/oradata/boston/onlinelog/group_5.279.711989151
 6          /u01/app/oracle/oradata/boston/onlinelog/group_6.280.711989159
 7          /u01/app/oracle/oradata/boston/onlinelog/group_7.281.711989165
```

# Viewing Standby Redo Log Information in V\$STANDBY\_LOG

```
SQL> SELECT group#, dbid, archived, status
  2  FROM v$standby_log;
```

GROUP#	DBID	ARC	STATUS
4	UNASSIGNED	NO	UNASSIGNED
5	3303427449	YES	ACTIVE
6	UNASSIGNED	YES	UNASSIGNED
7	UNASSIGNED	YES	UNASSIGNED

# Identifying Destination Settings

```
SQL> SELECT dest_id,valid_type,valid_role,valid_now
  2  FROM v$archive_dest;

DEST_ID VALID_TYPE      VALID_ROLE    VALID_NOW
-----  -----
  1 ALL_LOGFILES     ALL_ROLES      YES
  2 STANDBY_LOGFILE STANDBY_ROLE  WRONG_VALID_TYPE
  3 ONLINE_LOGFILE  STANDBY_ROLE  WRONG_VALID_ROLE
  4 ALL_LOGFILES    ALL_ROLES      UNKNOWN
  5 ALL_LOGFILES    ALL_ROLES      UNKNOWN
  6 ALL_LOGFILES    ALL_ROLES      UNKNOWN
  7 ALL_LOGFILES    ALL_ROLES      UNKNOWN
...
  9 ALL_LOGFILES    ALL_ROLES      UNKNOWN
 30 ALL_LOGFILES   ALL_ROLES      UNKNOWN
 31 ALL_LOGFILES   ALL_ROLES      UNKNOWN

31 rows selected.
```

# Viewing Standby Redo Log Information in V\$STANDBY\_LOG

- `LOG_ARCHIVE_TRACE` is optional and is used for diagnostic purposes.
- Set this parameter to an integer value to see the progression of redo log archiving to the standby system.
  - On the primary database, processes write an audit trail of the archived logs sent to the standby system into a trace file and information into the alert log.
  - On the standby database, processes write an audit trail of the archived logs received from the primary database into a trace file and information into the alert log.
- Trace files are written to the Automatic Diagnostic Repository, the location of which is specified by the `DIAGNOSTIC_DEST` initialization parameter.

# Viewing Redo Transport Errors by Querying V\$ARCHIVE\_DEST

```
SQL> SELECT DEST_ID "ID", STATUS "DB_status", DESTINATION
  "Archive_dest", ERROR "Error" FROM V$ARCHIVE_DEST
 WHERE DEST_ID <=5;

ID DB_status Archive_dest          Error
-- -----
 1 VALID      /u01/app/oracle/
                  fast_recovery_area/
                  boston
 2 ERROR      london           ORA-16012: Archivelog
                           standby database
                           identifier mismatch
 3 INACTIVE
 4 INACTIVE
 5 INACTIVE

5 rows selected.
```

# Evaluating Redo Data by Querying V\$DATAGUARD\_STATS

```
SQL> SELECT name, value, time_computed FROM v$dataguard_stats;
```

NAME	VALUE	TIME_COMPUTED
transport lag	+00 00:00:00	02/08/2020 20:07:05
apply lag	+00 00:00:00	02/08/2020 20:07:05
apply finish time	+00 00:00:00	02/08/2020 20:07:05
estimated startup time	10	02/08/2020 20:07:05

# Viewing Data Guard Status Information by Querying V\$DATAGUARD\_STATUS

```
SQL> SELECT timestamp, facility, message FROM v$dataguard_status
ORDER by timestamp;

TIMESTAMP FACILITY
-----
MESSAGE
-----
...
ARC9: Beginning to archive thread 1 sequence 122 (5336651-5404240)

08-FEB-20 Log Apply Services
Media Recovery Waiting for thread 1 sequence 123 (in transit)

08-FEB-20 Log Transport Services
ARC9: Completed archiving thread 1 sequence 122 (0-0)
```

# Monitoring Redo Apply by Querying V\$DATAGUARD\_PROCESS

- To verify database modifications are being successfully transmitted from the primary database to the standby database:

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

- The recovery logmerger role shows that redo is being applied at the standby.

**Note:** The V\$DATAGUARD\_PROCESS view now shows broker processes.

# Monitoring SQL Apply by Querying V\$LOGSTDBY\_TRANSACTION

- Query V\$LOGSTDBY\_TRANSACTION to view information about all transactions that are actively being processed by SQL Apply:

```
SQL> SELECT primary_xid, type, mining_status, apply_status
  2  FROM v$logstdby_transaction;
```

# Quiz

- You can click the Data Guard performance charts that show real-time data in Oracle Enterprise Manager to view historical data.
  - a. True
  - b. False

# Quiz

- The LOG\_ARCHIVE\_TRACE initialization parameter is valid only on the primary database to display diagnostic information regarding sending of redo data.
  - a. True
  - b. False

# Quiz

- What can be performed using the Data Guard Broker VALIDATE command?
  - a. Performs a comparison of SPFILE entries between the primary database and a specified standby database
  - b. Performs network connectivity checks between members of a broker configuration
  - c. Validates the static connect identifier of a database
  - d. Validates data files for both primary and standby databases
  - e. All of above

# Summary

- In this lesson, you should have learned how to:
  - Use Enterprise Manager Grid Control to monitor the configuration
  - Use DGMGRL to monitor the configuration
  - List the new Data Guard Broker VALIDATE command
  - Use SQL to monitor the configuration

# Practice 11: Overview

- This practice covers the following topics:
  - Monitoring the Physical Standby Database
  - Examining Data Guard Log and Trace Files
  - Using the Data Guard Broker `VALIDATE` commands
  - Performing a Test Workload

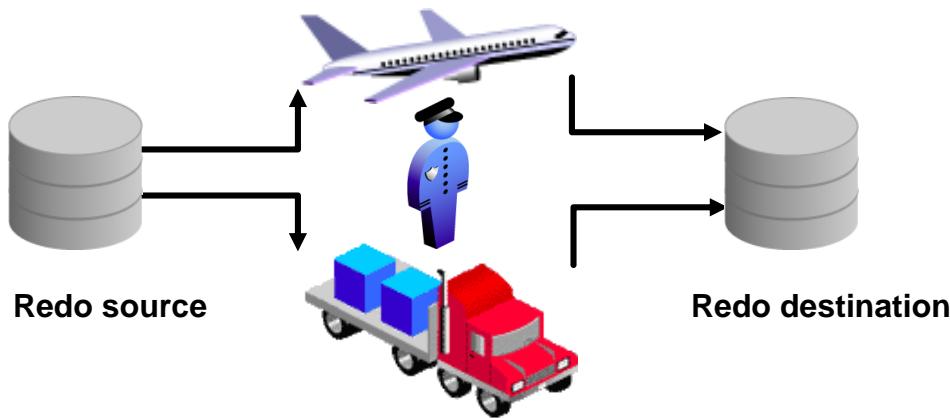
# 12. Configuring Data Protection Modes

# Objectives

- After completing this lesson, you should be able to:
  - Describe the data protection modes
  - Change the data protection mode of your configuration
    - Using Enterprise Manager Cloud Control
    - Using DGMGRL
    - Using SQL

# Data Protection Modes and Redo Transport Modes

- A data protection mode requires a specific redo transport mode.
- A redo transport mode alone does not define a data protection mode.



# Data Protection Modes

- Three data protection modes:
  - Maximum protection
  - Maximum availability
  - Maximum performance
- Help to balance data availability and system performance



# Maximum Protection Mode

- Maximum protection mode ensures zero data loss in the event of a failure of the primary database, the network, or all standby databases.
- The primary database shuts down if a fault prevents it from writing its redo stream to at least one synchronized standby database.
- Redo data must be written to both the local online redo log and the remote standby redo log on at least one synchronized standby database.
- Configuration requirements: At least one standby database must have a standby redo log, and that standby database destination must be configured with the `SYNC` and `AFFIRM` redo transport attributes.

# Maximum Availability Mode

- Maximum availability mode ensures zero data loss without compromising the availability of the primary database.
- Redo data must be written to both the local online redo log and received by the remote RFS process of a standby database.
- The primary database does not shut down if it cannot receive acknowledgement from at least one standby.
- If no synchronized standby databases are available, the primary database operates in an unsynchronized mode until at least one standby database is synchronized.
- Configuration requirements: At least one standby database must have a standby redo log, and that standby database destination must be configured with the **SYNC** and **NOAFFIRM** redo transport attributes as a minimum.

# Maximum Performance Mode

- Maximum performance mode is the default level of data protection.
- This mode provides the highest possible level of data protection without affecting the performance of the primary database.
- Transactions can commit as soon as the redo data is written to the local online redo log.
- Redo data is shipped to the standby database asynchronously with respect to the commitment of the transactions that create the redo data.
- Configuration requirements:
  - Standby redo log on at least one standby database
  - At least one standby database that is configured with the `ASYNC` and `NOAFFIRM` redo transport attributes

# Comparing Data Protection Modes

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

# Setting the Data Protection Mode by Using Enterprise Manager

**ORACLE® Enterprise Manager Cloud Control 13c**

**boston.example.com (Container Database)** i

Logged in as **sys** host01...

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

**Data Guard**

Page Refreshed May 31, 2020 10:10:18 PM EDT

View Data | Real Time: Manual Refresh

**Overview**

Data Guard Status Normal  
Protection Mode Maximum Performance  
Fast-Start Failover Disabled

**Standby Database Progress Summary**

Click the Protection Mode link.

The time difference between the last update on the primary database and the last received redo on the standby database is the time difference between the last update on the primary database and the last applied redo on the standby database.

seconds

1.0

0.5

0.0

0 0

london.example.com

Transport Lag

Apply Lag

**Primary Database**

Name boston.example.com  
Host host01.example.com  
Data Guard Status Normal  
Current Log 35  
Properties Edit

# Setting the Data Protection Mode by Using Enterprise Manager

The screenshot shows the Oracle Enterprise Manager Cloud Control 13c interface. The top navigation bar includes the Oracle logo, the title "Enterprise Manager Cloud Control 13c", and various management icons. The main header displays the URL "boston.example.com (Container Database)" and the user "Logged in as sys". The menu bar below the header lists "Oracle Database", "Performance", "Availability", "Security", "Schema", and "Administration". The breadcrumb navigation shows "Data Guard > Change Protection Mode: Select Mode". The main content area is titled "Change Protection Mode: Select Mode" and contains a descriptive text about Data Guard protection modes. It lists three options: "Maximum Protection", "Maximum Availability" (which is selected), and "Maximum Performance". Each option has a detailed description. At the bottom right of the content area are "Cancel" and "Continue" buttons.

Data Guard > Change Protection Mode: Select Mode

**Change Protection Mode: Select Mode**

Data Guard provides multiple protection modes. Higher protection modes reduce data loss but may affect performance of the primary database. When changing to maximum protection or maximum availability, a SYSDBA connection is required to the primary database and all standby databases to determine if standby redo log files are needed.

**Maximum Protection**  
Provides the highest level of data protection. No data will be lost. Possible primary database downtime if connectivity to the standby database is lost. Requires the SYNC redo transport mode to be set on at least one standby database.

**Maximum Availability**  
Provides very high data protection. No primary database downtime if connectivity to the standby database is lost but data may diverge. Requires the SYNC redo transport mode to be set on at least one standby database.

**Maximum Performance**  
No performance impact on the primary database. Provides high data protection with the ASYNC redo transport mode.

**Cancel** **Continue**

# Setting the Data Protection Mode by Using DGMGRL

1. Configure standby redo logs.
2. Set the `LogXptMode` property (if necessary).
  - Maximum protection: `SYNC`
  - Maximum availability: `SYNC` or `FASTSYNC`
  - Maximum performance: `ASYNC`
3. Set the data protection mode.

```
DGMGRL> EDIT DATABASE 'london' SET PROPERTY 'LogXptMode'='FASTSYNC';
Property "LogXptMode" updated
```

```
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MAXAVAILABILITY;
Succeeded.
```

# Setting the Data Protection Mode by Using SQL

1. Set the data protection mode on the primary database:

```
SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE AVAILABILITY;  
Database Altered.
```

1. Confirm that the primary database is operating in the new protection mode:

```
SQL> SELECT protection_mode FROM v$database;
```

# Quiz

- Which of the following are allowed Data Guard protection modes?
  - a.Maximum Protection
  - b.Maximum Availability
  - c.Maximum Throughput
  - d.Maximum Performance

# Summary

- In this lesson, you should have learned how to:
  - Describe the data protection modes
  - Change the data protection mode of your configuration
    - Using Enterprise Manager Cloud Control
    - Using DGMGRL
    - Using SQL

# Practice 12: Overview

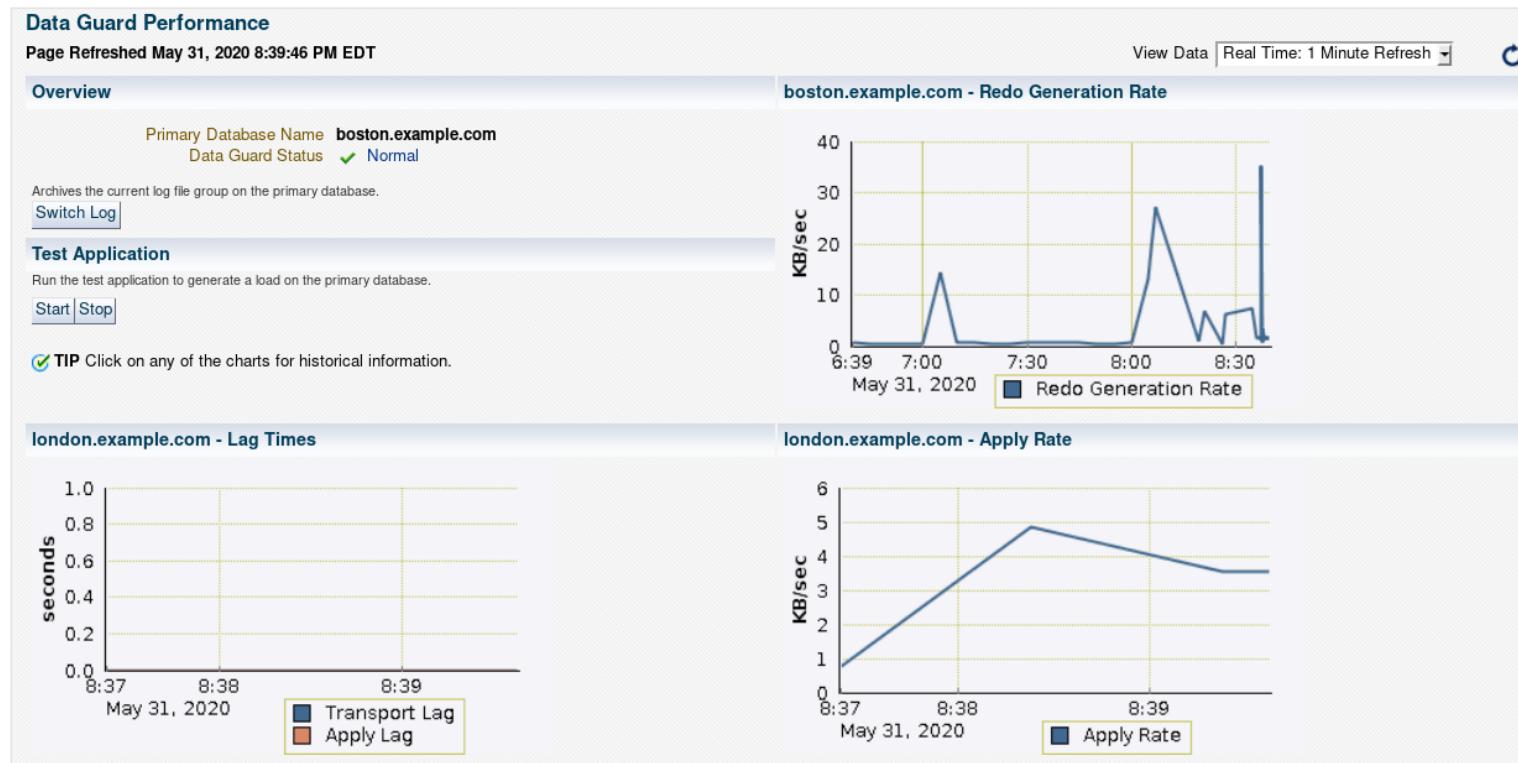
- This practice covers the following topics:
  - Examining the Maximum Availability Protection Mode
  - Examining the Maximum Protection Mode

# 13. Optimizing and Tuning a Data Guard Configuration

# Objectives

- After completing this lesson, you should be able to:
  - Monitor configuration performance
  - Optimize redo transport for best performance
  - Optimize SQL Apply
  - Describe Tunable Automatic Outage Resolution
  - List Diagnostic Tools in Active Data Guard (Read-Only) environment

# Monitoring Configuration Performance by Using Enterprise Manager Cloud Control



# Optimizing Redo Transport Services

- Optimize redo transport with the following techniques:
  - Setting the NetTimeout Database Property
  - Setting the DataGuardSyncLatency Database Property
  - Using Redo Transport Compression

# Setting the ReopenSecs Database Property

- This property specifies the minimum number of seconds before the archiver process tries to access a previously failed destination.
- Broker default: 300
- Set for the standby database or Far Sync instance.
- The setting is propagated to the REOPEN attribute of the LOG\_ARCHIVE\_DEST\_n initialization parameter of the sending instance.

```
DGMGRl> EDIT DATABASE 'london' SET PROPERTY 'ReopenSecs'=600;
```

# Setting the NetTimeout Database Property

- This property specifies the number of seconds that the log writer process (LGWR) waits for Oracle Net Services to respond to a request.
- Broker default: 30
- The setting is propagated to the `NET_TIMEOUT` attribute of the `LOG_ARCHIVE_DEST_n` initialization parameter of the sending instance.

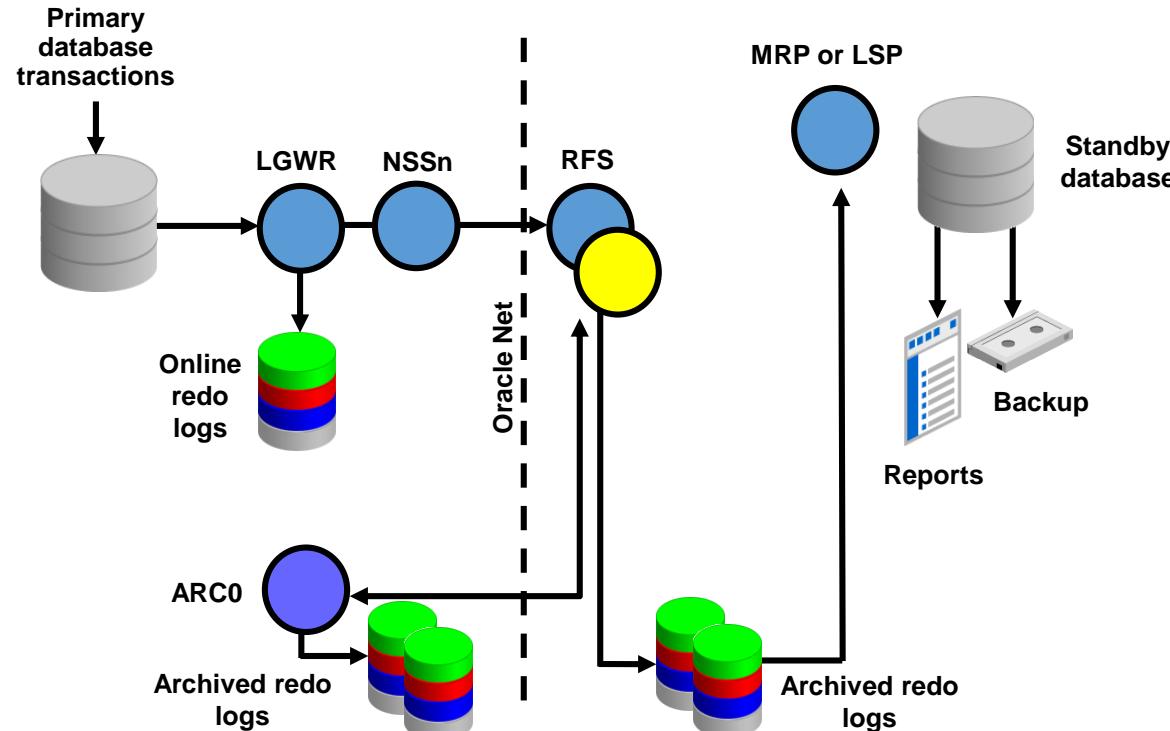
```
DGMGRL> EDIT DATABASE 'london' SET PROPERTY 'NetTimeout'=20;
```

# Setting the DataGuardSyncLatency Property

- The DataGuardSyncLatency property defines the maximum amount of time that the primary must wait before disconnecting other destinations after at least one SYNC standby has acknowledged receipt of the redo.
- It supports the DATA\_GUARD\_SYNC\_LATENCY initialization parameter.
- Broker default: 0 (meaning that the LGWR will wait up to the number of seconds specified by the NetTimeout database property for each SYNC standby destination)
- Setting the property:

```
DGMGRL> EDIT DATABASE 'london' SET PARAMETER  
DATA_GUARD_SYNC_LATENCY = 2 'SCOPE = BOTH';
```

# Optimizing Redo Transmission by Using Redo Transport Compression



# Compressing Redo Data by Setting the RedoCompression Property

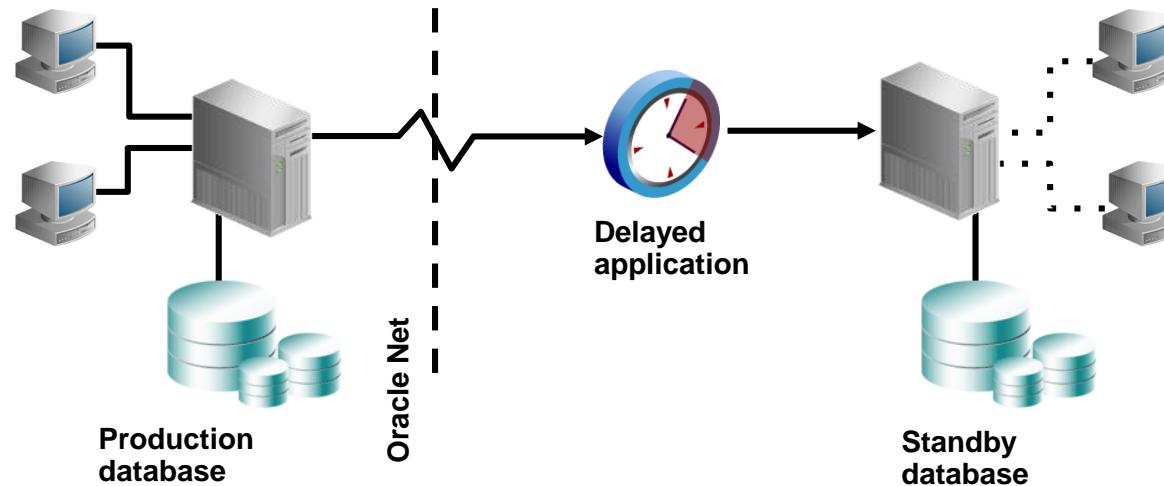
- This can be enabled for all redo transport methods (ASYNC and SYNC).
- The setting is propagated to the COMPRESSION attribute of the LOG\_ARCHIVE\_DEST\_n initialization parameter.
- Determine whether redo compression is enabled by querying the COMPRESSION column of the V\$ARCHIVE\_DEST view.
- Enable by using DGMGRL:

```
DGMGRL> EDIT DATABASE 'london' SET PROPERTY  
'RedoCompression'='ENABLE';  
Enable by using SQL.
```

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_3 = 'SERVICE=london SYNC  
COMPRESSION=ENABLE';
```

# Delaying the Application of Redo

- Delaying the application of redo helps safeguard against:
  - Data corruption
  - User errors



# Setting the DelayMins Database Property to Delay the Application of Redo

- This property specifies the number of minutes that log apply services must wait before applying redo data to the standby database.
- Broker default: 0 (meaning that apply services applies redo data as soon as possible)
- The setting is propagated to the `DELAY` attribute of the `LOG_ARCHIVE_DEST_n` initialization parameter of the primary database or Far Sync.

```
DGMGRL> EDIT DATABASE 'london' SET PROPERTY 'DelayMins'=5;
```

# Using Enterprise Manager to Delay the Application of Redo

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

Data Guard > Edit Standby Database Properties: london.example.com  
**Edit Standby Database Properties: london.example.com**

Revert Apply

General Standby Role Properties Common Properties

**TIP** The database is currently in the standby role. Any modifications take effect immediately.

<b>Redo Transport Mode</b>	ASYNC
Method used to transport redo to this standby database.	
<b>Redo Compression</b>	DISABLE
Specifies whether redo data is transmitted in compressed form.	
<b>Net Timeout</b>	30
Amount of time the primary database will wait for acknowledgement from this destination before terminating the network connection.	
<b>Apply Delay (minutes)</b>	0
Number of minutes to delay applying archived redo log files. If zero, redo is applied as soon as possible.	
<b>Online Archive Location</b>	
Location on the database host for online redo log files.	
<b>Standby Archive Location</b>	
Location on the standby host for archived redo log files.	

Specify the delay in minutes.

Show Advanced Properties

Revert Apply

# Optimizing SQL Apply

- Adjust the number of processes allocated to SQL Apply.
  - APPLIER processes
  - PREPARER processes
- Modify SQL Apply parameters to control the number of processes allocated to SQL Apply.
  - APPLY\_SERVERS: The number of APPLIER processes that are used to apply changes
  - MAX\_SERVERS: The number of processes that SQL Apply uses to read and apply redo
  - PREPARE\_SERVERS: The number of PREPARER processes that are used to prepare changes
- Set SQL Apply parameters by using the DBMS\_LOGSTDBY package.

# Adjusting the Number of APPLIER Processes

- Determine whether adjusting the number of APPLIER processes achieves greater throughput.
  - Check whether APPLIER processes are busy:

```
SQL> SELECT COUNT(*) AS IDLE_APPLIER
  2  FROM V$LOGSTDBY_PROCESS
  3 WHERE TYPE = 'APPLIER' and status_code = 16116;
```

- Ensure there is enough work available for additional APPLIER processes:

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS
  2 WHERE NAME = 'txns applied'
  3 OR NAME = 'distinct txns in queue';
```

- Adjust the MAX\_SERVERS parameter (if necessary).
- Adjust the APPLY\_SERVERS parameter to increase the number of APPLIER processes.

# Adjusting the Number of PREPARER Processes

- Determine whether adjusting the number of PREPARER processes is required.
  - Ensure all PREPARER processes are busy:

```
SQL> SELECT COUNT(*) AS IDLE_PREPARER
  2  FROM V$LOGSTDBY_PROCESS
  3  WHERE TYPE = 'PREPARER' and status_code = 16116;
```

- Ensure the number of transactions ready to be applied is less than the number of available APPLIER processes:

```
SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS
  2  WHERE NAME = 'txns applied'
  3  OR NAME = 'distinct txns in queue'
SQL> SELECT COUNT(*) AS APPLIER_COUNT
  2  FROM V$LOGSTDBY_PROCESS WHERE TYPE = 'APPLIER';
```

# Adjusting the Number of PREPARER Processes

- Determine whether there are idle APPLIER processes:

```
SQL> SELECT COUNT(*) AS IDLE_APPLIER
  2  FROM V$LOGSTDBY_PROCESS
  3  WHERE TYPE = 'APPLIER' and status_code = 16116;
```

- Adjust the MAX\_SERVERS parameter if necessary.
- Adjust the PREPARE\_SERVERS parameter to increase the number of PREPARER processes.

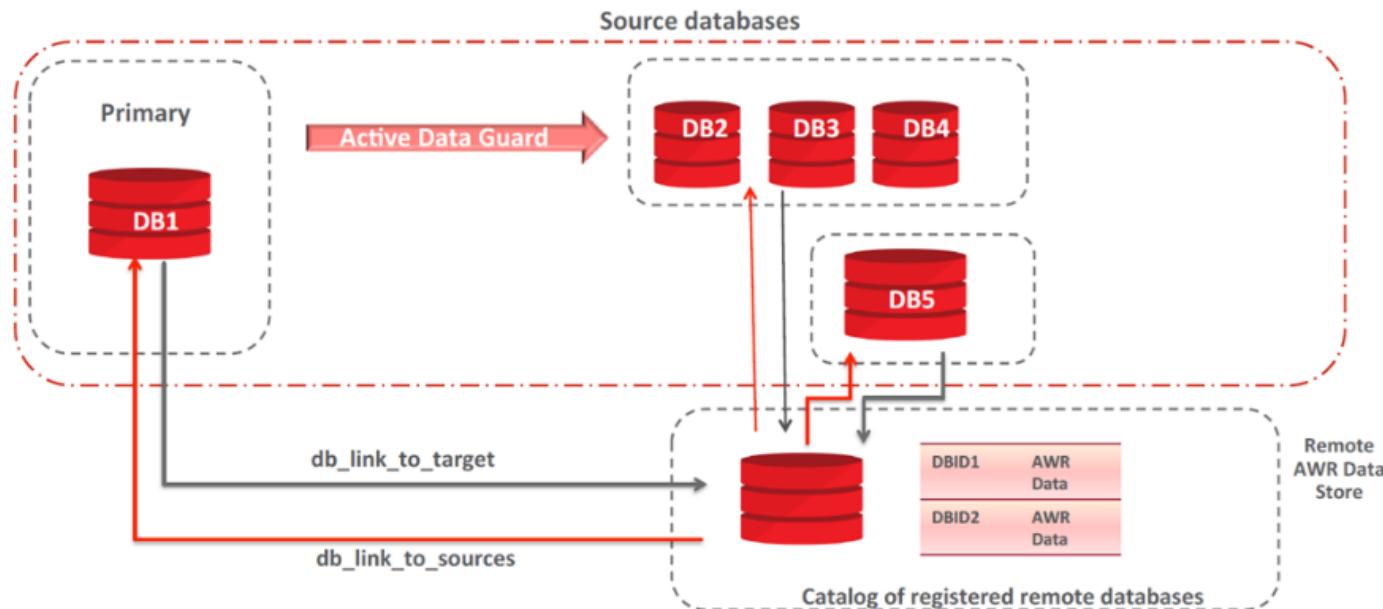
# Tuning Automatic Outage Resolution

- Data Guard maintains internal mechanisms that detect and correct issues with its redo transport and gap resolution processes.
  - In case of network or disk I/O problems, these mechanisms prevent those processes from hanging and causing unnecessarily long gaps.
- Use the following parameters to influence the outage resolution:
  - DATA\_GUARD\_MAX\_IO\_TIME: Sets the maximum number of seconds that can elapse before a process is considered hung while performing reads, writes, and status operations.
  - DATA\_GUARD\_MAX\_LONGIO\_TIME: Sets the maximum number of seconds as above, but for operations such as open and close.

# Diagnostic Tools in Active Data Guard (Read-Only) Standby Databases

- Standby Statspack
- In-Memory Active Session History (ASH)
- Real-Time SQL Monitoring
- 10046 & 10053 trace
- SQLT XTRSBY Method
- SQL Health Check Report
- Systemstate Dump and Hanganalyze Trace
- Automatic Workload Repository (AWR)
- Automatic Diagnostic Database Monitor (ADDM)

# AWR Unified Management Framework Topology



# Remote Snapshot Configuration for ADG Database



# Quiz

- Enabling Data Guard network redo compression requires the Oracle database Advanced Compression option.
  - a. True
  - b. False

# Quiz

- The Data Guard MAX\_SERVERS, APPLY\_SERVERS, and PREPARE\_SERVERS parameters can be adjusted with the database initialization parameter file.
  - a. True
  - b. False

# Quiz

- Which diagnostic tools are available for the Active Data Guard instances?
  - In-Memory Active Session History (ASH)
  - a. AWR
  - b. ADDM
  - c. Real-Time SQL Monitoring
  - d. All of above

# Summary

- In this lesson, you should have learned how to:
  - Monitor configuration performance
  - Optimize redo transport for best performance
  - Optimize SQL Apply
  - Describe Tunable Automatic Outage Resolution
  - List Diagnostic Tools in Active Data Guard (Read-Only) environment

# Practice 13: Overview

- This practice covers the following topics:
  - Configuring network compression of redo data.
  - Generating AWR report for an Active Data Guard Instance
  - Using SQL Tuning Advisor for an Active Data Guard Instance
  - Using ADDM for an Active Data Guard Instance

# 14. Performing Role Transitions

# Objectives

- After completing this lesson, you should be able to:
  - Explain the database roles
  - Perform a switchover
  - Perform a failover
  - Explain how to keep physical standby sessions during role transit

# Role Management Services

- In a Data Guard configuration, a database operates in one of two mutually exclusive roles:
  - Primary role
  - Standby role (Physical, Logical, Snapshot subtypes)
- With role management services, you can change these roles dynamically.



# Role Transitions: Switchover and Failover

- Switchover
  - Planned role transition
  - Used for operating-system or hardware maintenance
  - Used for database rolling upgrade
  - Manually invoked on primary database
- Failover
  - Unplanned role transition
  - Used in an emergency
  - Minimal or no data loss (depending on the data-protection mode)
  - Fast-start failover can be enabled for automatic failover
  - Initiated at standby database

# Buffer Cache Preservation During Role Change

- The database buffer cache state is now maintained on an Active Data Guard (ADG) standby during a role change.
- Previously when an open ADG Standby transitioned to Primary, the buffer cache that was in use was recycled and had to be populated again from disk as blocks were read.
- With 18.1, the buffer cache remains intact during the role change so that performance is not affected by physical blocks read from disk to populate the buffer cache.
- Because of this, application performance is improved on the new primary after a role transition.

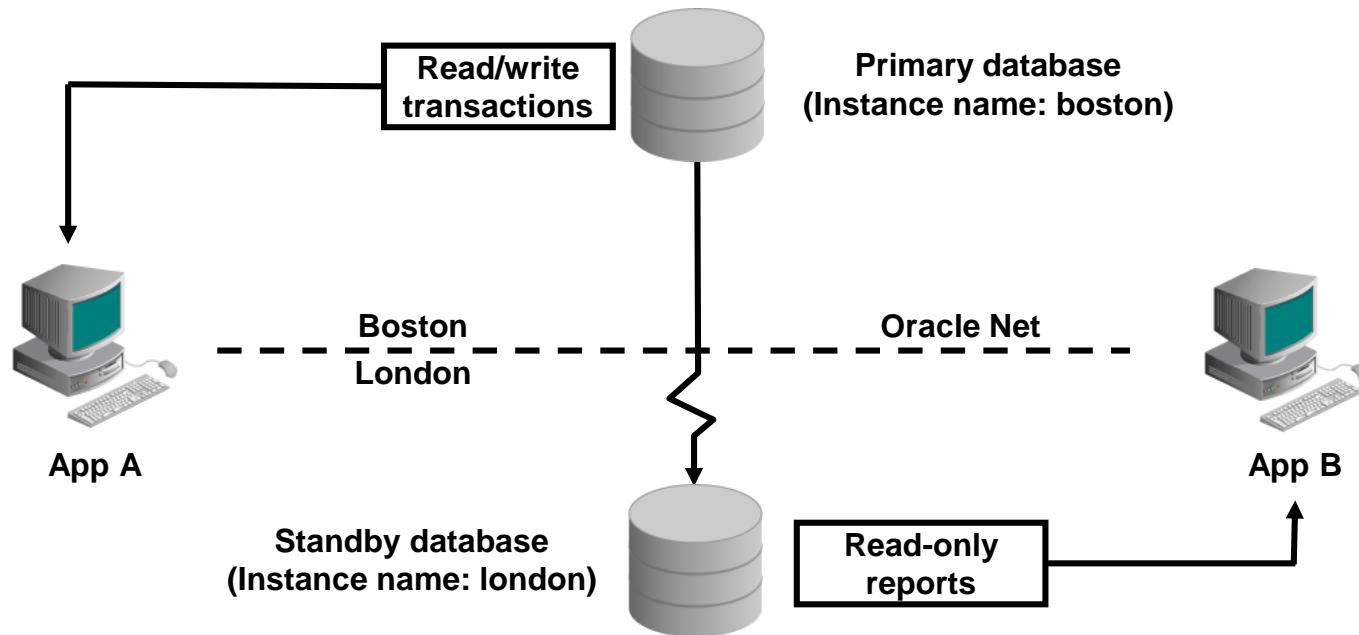
# Keeping Physical Standby Sessions Connected During Role Transition

- **STANDBY\_DB\_PRESERVE\_STATES** allows you to keep user sessions when a physical standby is converted to a primary.
- The values that are allowed include:
  - **NONE**: No sessions on the standby are retained during a switchover or failover. This is the default value.
  - **SESSION**: User sessions are retained during a switchover or failover.
  - **BUFFER**: Current buffers are retained during switchover or failover.
  - **ALL**: This value is equivalent to setting both the SESSION and BUFFER values.
- This parameter is meaningful on a physical standby database that is open in real-time query mode.

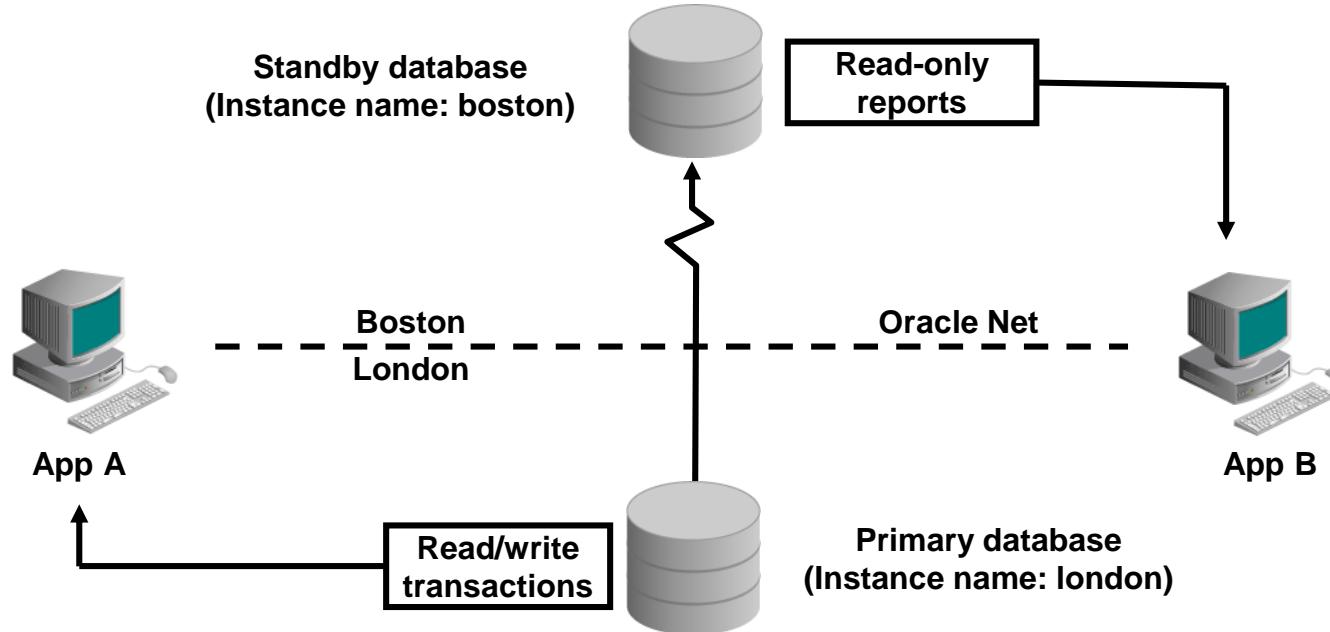
# Switchover

- Transitions the roles of the primary and standby databases
- Requires no resetting of the online redo logs of the new primary database
- Incurs no data loss

# Switchover: Before



# Switchover: After



# Performing a Switchover by Using Enterprise Manager

Logged in as sys host01.example.com

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

## Data Guard

Page Refreshed May 31, 2020 10:35:43 PM EDT

View Data | Real Time: Manual Refresh

### Overview

Data Guard Status	Normal
Protection Mode	Maximum Performance
Fast-Start Failover	Disabled

### Primary Database

Name	boston.example.com
Host	host01.example.com
Data Guard Status	Normal
Current Log	41

Select the database and click Switchover.

### Standby Database Progress Summary

Transport lag is the time difference between the last update on the primary database and the last received redo on the standby database. Apply lag is the time difference between the last update on the primary database and the last applied redo on the standby database.

seconds

0.0 0.5 1.0

0.0 0.5 1.0

0 0

london.example.com

Transport Lag

Apply Lag

### Standby Databases

Add Far Sync | Add Standby Database

Edit	Remove	Switchover	Failover	Convert					
Select	Name	Host	Data Guard Status	Role	Redo Source	Real-time Query	Last Received Log	Last Applied Log	Estimated Failover Time
<input checked="" type="radio"/>	london.example.com	host03.example.com	Normal	Physical Standby	boston	Disabled	40	40	< 1 second

# Performing a Switchover by Using Enterprise Manager

The screenshot shows the Oracle Enterprise Manager Cloud Control 13c interface. The top navigation bar includes the ORACLE logo, the product name, and various system icons. The main header displays the URL "boston.example.com (Container Database)" and the user "Logged in as sys". The menu bar below the header lists "Oracle Database", "Performance", "Availability", "Security", "Schema", and "Administration". A confirmation dialog box is open, asking if the user wants to switch over to "london.example.com". It provides information about the operation and a link to "Browse Primary Database Sessions". Below the dialog, there is a "Monitoring Settings" section with a checkbox for swapping monitoring settings. At the bottom right of the dialog, there are "No" and "Yes" buttons.

ORACLE Enterprise Manager Cloud Control 13c

boston.example.com (Container Database) i

Logged in as sys host01.example.com

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

✓ Confirmation: Switchover to london.example.com

Are you sure you want to switchover to london.example.com?

A switchover will cause the primary and standby databases to switch roles. The switchover operation cannot be cancelled.

Any active sessions connected to the primary database will be closed automatically during the switchover operation.

Browse Primary Database Sessions

**Monitoring Settings**

Monitoring settings can optionally be swapped between the primary and standby databases as a part of the role change operation.

Swap Monitoring Settings

The current Enterprise Manager monitoring settings (including metric thresholds) for the primary and standby databases will be swapped after the role change, overriding all settings for each database with the values from the other database. If more granular monitoring standard swapping is desired, de-select this option and use the Monitoring Standards interface to create monitoring templates prior to the role change and apply them afterwards.

No Yes

Click Yes to confirm.

# Performing a Switchover by Using Enterprise Manager

The screenshot shows the Oracle Enterprise Manager interface for a Container Database at [boston.example.com](http://boston.example.com). The user is logged in as sys with host01.example.com. The navigation bar includes links for Oracle Database, Performance, Availability, Security, Schema, and Administration. A processing message indicates "Switching over to london.example.com". It states that the process takes time and returns to the Data Guard overview page upon completion. A tip message notes that the process cannot be cancelled if the browser window is closed. The status bar shows a progress bar and three tasks: "Performing role change" (checkmark), "Restarting new standby database" (blue arrow), and "Waiting for switchover to complete".

Logged in as sys host01.example.com

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

**Processing: Switchover**  
Switching over to london.example.com

This process takes some time. The page automatically returns to the Data Guard overview page upon completion.  
Click on the alert log link to view progress details in a new browser window. View alert log: [boston.example.com](#) [london.example.com](#)

**Performing role change**  
→ Restarting new standby database  
Waiting for switchover to complete

**TIP** This process cannot be cancelled. It will continue even if the browser window is closed.

# Validating Databases for Switchover by Using DGMGRL

- The `VALIDATE DATABASE` command performs a comprehensive set of database checks prior to a role change.

```
DGMGRL> validate database london
Database Role:      Physical standby database
Primary Database:   boston
Ready for Switchover: Yes
Ready for Failover:  Yes (Primary Running)
Flashback Database Status:
    boston: Off
    london: Off
Current Log File Groups Configuration:
    Thread#  Online Redo Log Groups  Standby Redo Log Groups
                  (boston)           (london)
                    1                 3                   2
Future Log File Groups Configuration:
    Thread #  Online Redo Log Groups  Standby Redo Log Groups
                  (london)           (boston)
                    1                 3                   0
```

# Performing a Switchover by Using DGMGRL

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

```
DGMGRL> SWITCHOVER TO 'london';
Performing switchover NOW, please wait...
Operation requires a connection to instance "london" on
database "london"
Connecting to instance "london"...
Connected as SYSDG.
New primary database "london" is opening...
Operation requires startup of instance "boston" on
database "boston"
Starting instance "boston"...
ORACLE instance started.
Database mounted.
Switchover succeeded, new primary is "london"
```

# Preparing for a Switchover Using SQL

- To prepare for the SQL switchover operation, verify that:
  - Each database is properly configured for the role it is about to assume
  - There are no redo transport errors or gaps at the standby database by querying `V$ARCHIVE_DEST_STATUS`
  - Temporary files exist on the standby database to match the temporary files on the primary database
  - Delays in applying redo on the standby database that will become the new primary are removed
  - Active Data Guard is stopped for the fastest possible role transition
  - Any user sessions connected to the physical standby are cleanly terminated prior to switchover

# Performing a Switchover by Using SQL

1. Verify that the target standby database is ready:

```
SQL> ALTER DATABASE SWITCHOVER TO 'london' VERIFY;  
Database altered.
```

2. Initiate the switchover on the primary database:

```
SQL> ALTER DATABASE SWITCHOVER TO 'london';  
Database altered.
```

3. Open the new primary database.
4. Mount the new physical standby database.
5. Start Redo Apply on the new physical standby database.

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT;  
Database altered.
```

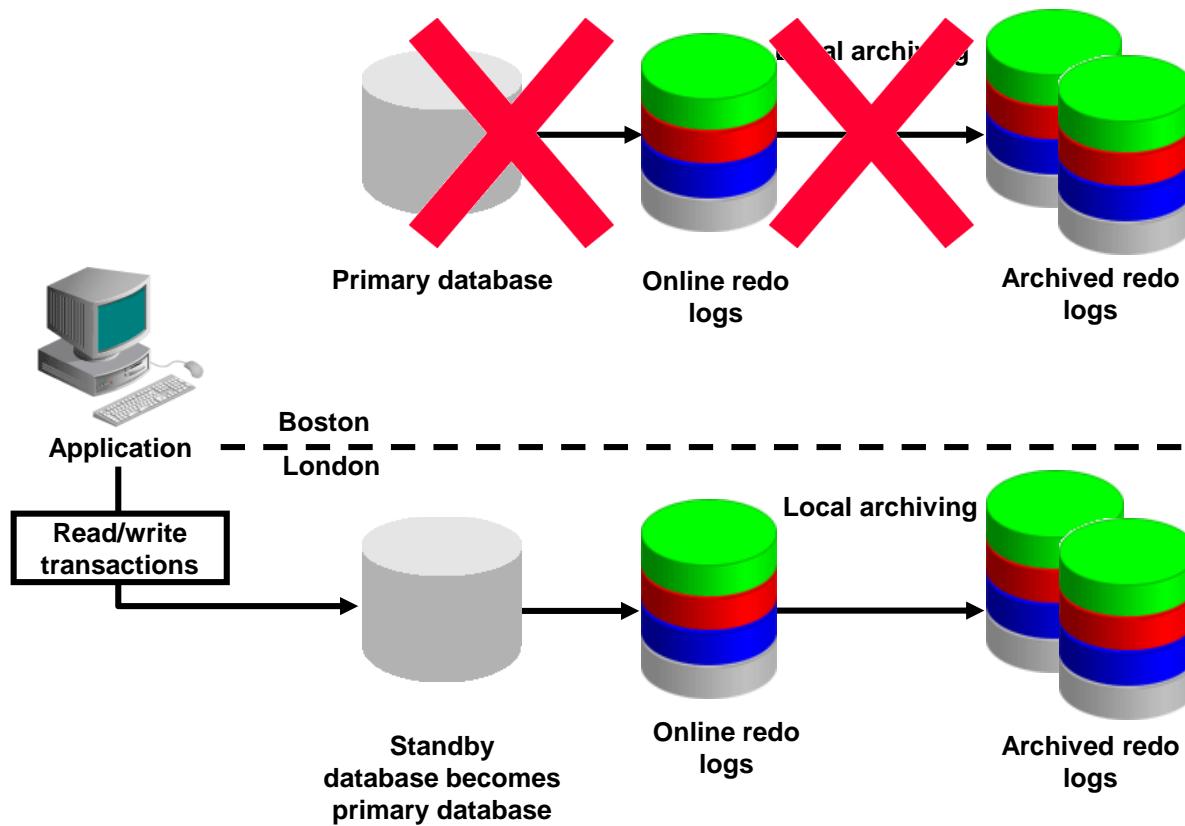
# Considerations When Performing a Switchover to a Logical Standby Database

- The switchover operation does not cause a shutdown of the primary database instance.
- Although there is no need to terminate user sessions, termination is recommended.
- The logical standby database may not have all data.
- Switchover to a logical standby database invalidates and disables all physical and snapshot standby databases in the broker-managed Data Guard configuration.
- Other logical standbys in the configuration will become standbys of the new primary.

# Situations That Prevent a Switchover

- You cannot perform a switchover if:
  - Archived redo log files are unavailable
  - Point-in-time recovery is required
  - The primary database is not open and cannot be opened

# Failover



# Types of Failovers

- Manual failover: Invoked by the DBA
  - Complete: Attempts to minimize data loss by applying all available redo on the standby database
  - Immediate: No additional data is applied on the standby database
- Fast-start failover: Invoked automatically by the Data Guard broker

# Failover Considerations

- The old primary database is disabled from the Data Guard configuration.
- Data loss is possible.
- Failover should be used only in an emergency.
- When choosing a standby database to fail over to, you should:
  - Choose a physical standby database when possible
  - Choose the standby database that is most current

# Performing a Failover by Using Enterprise Manager

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

## Data Guard

Page Refreshed May 31, 2020 10:45:49 PM EDT

View Data | Real Time: 30 Second Refresh |

### Overview

Data Guard Status: ✓ Normal  
Protection Mode: Maximum Performance  
Fast-Start Failover: Disabled

### Primary Database

Name: london.example.com  
Host: host03.example.com  
Data Guard Status: ✓ Normal  
Current Log: 44  
Properties: Edit

Standby Database Progress Summary

Transport lag is the time difference between the last update on the primary database and the last received redo on the standby database. Apply lag is the time difference between the last update on the primary database and the last applied redo on the standby database.

No data is currently available.

Select the database and click Failover.

### Standby Databases

Add Far Sync | Add Standby Database

	Edit	Remove	Switchover	<b>Failover</b>	Convert				
Select	Name	Host	Data Guard Status	Role	Redo Source	Real-time Query	Last Received Log	Last Applied Log	Estimated Failover Time
<input checked="" type="radio"/>	boston.example.com	host01.example.com	✓ Normal	Physical Standby	london	Disabled	43	43	Not available

# Performing a Failover by Using Enterprise Manager

 **Confirmation: Failover to boston.example.com**

Are you sure you want to failover to boston.example.com?

A failover will cause the standby database to become the primary database. The failover operation cannot be cancelled.

**Select Failover Option**

**Complete**  
All available redo data will be applied on the standby database, thereby minimizing data loss. Oracle recommends this type of failover.

**Immediate**  
No additional redo data will be applied on the standby database; data may be lost. This is the fastest type of failover.

**Monitoring Settings**

Monitoring settings can optionally be swapped between the primary and standby databases as a part of the role change operation.

**Swap Monitoring Settings**  
The current Enterprise Manager monitoring settings (including metric thresholds) for the primary and standby databases will be swapped after the role change, overriding all settings for each database with the values from the other database. If more granular monitoring standard swapping is desired, de-select this option and use the Monitoring Standards interface to create monitoring templates prior to the role change and apply them afterwards.

**No** **Yes**

# Performing a Failover by Using Enterprise Manager

The screenshot shows a web interface for Oracle Database Enterprise Manager. At the top, there is a navigation bar with tabs: Oracle Database ▾, Performance ▾, Availability ▾, Security ▾, Schema ▾, and Administration ▾. Below the navigation bar, the main content area has a title "Processing: Failover" with a gear icon. Underneath it, the text "Failing over to boston.example.com" is displayed. A message states: "This process takes some time. The page automatically returns to the Data Guard overview page upon completion." Below this, a link says "Click on the alert log link to view progress details in a new browser window. View alert log: london.example.com boston.example.com". In the center, there is a large blue progress bar with a white loading icon. To the right of the progress bar, the text "Performing failover" and "Waiting for failover to complete" is shown with a blue arrow pointing right. At the bottom left, there is a green checkmark icon followed by the text "TIP This process cannot be cancelled. It will continue even if the browser window is closed."

# Performing a Failover to a Physical Standby Database

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

**Information**  
Failover completed successfully.

**Data Guard**  
Page Refreshed May 31, 2020 11:19:46 PM EDT

View Data Real Time: 30 Second Refresh

**Overview**

Data Guard Status	✓ Normal
Protection Mode	Maximum Performance
Fast-Start Failover	Disabled

**Primary Database**

Name	boston.example.com
Host	host01.example.com
Data Guard Status	✓ Normal
Current Log	1
Properties	Edit

**Standby Database Progress Summary**

Transport lag is the time difference between the last update on the primary database and the last received redo on the standby database. Apply lag is the time difference between the last update on the primary database and the last applied redo on the standby database.

No data is currently available.

The standby databases must be reinstated.

**Standby Databases**

Add Far Sync | Add Standby Database

Select	Name	Host	Data Guard Status	Role	Redo Source	Real-time Query	Last Received Log	Last Applied Log	Estimated Failover Time
<input checked="" type="radio"/>	london.example.com	host03.example.com	Database must be reinstated	Physical Standby	Unknown	Disabled	Not available	Not available	Not available

# Performing a Manual Failover by Using DGMGRL

1. Execute the FAILOVER command to initiate the failover operation to the standby database:

```
DGMGRL> FAILOVER TO 'london' [IMMEDIATE] ;
```

1. Reset the protection mode (if necessary).
2. Reinstate or re-create the former primary database to serve as a standby database in the configuration.
3. Reinstate or re-create other disabled standby databases in the configuration.

# Re-enabling Disabled Databases by Using DGMGRL

- Disabled databases must be reinstated or re-created to re-enable broker management.
- Reinstate a database using REINSTATE DATABASE:

```
DGMGRL> REINSTATE DATABASE boston;
```

- If you cannot reinstate a database, re-create it from a copy of the primary database and then re-enable the database by using ENABLE DATABASE:

```
DGMGRL> ENABLE DATABASE boston;
```

# Quiz

- The database buffer cache state is now maintained on an Active Data Guard (ADG) standby during a role change.
  - a.True
  - b.False

# Quiz

- After a failover operation, the old primary database is removed from the Data Guard configuration and must be re-created from a backup.
  - a.True
  - b.False

# Summary

- In this lesson, you should have learned how to: use
  - DGMGRL to perform switchover and failover operations
  - Enterprise Manager to perform switchover and failover operations

# Practice 14: Overview

- This practice covers the following topics:
  - Performing a Switchover Using DGMGRL
  - Keeping Physical Standby Session Connected During Role Transition

# 15. Using Flashback Database in a Data Guard Configuration

# Objectives

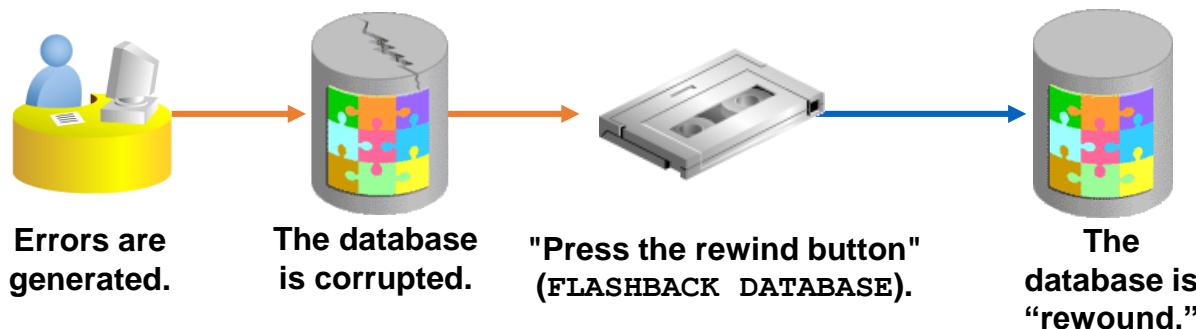
- After completing this lesson, you should be able to:
  - Explain the advantages of using Flashback Database in a Data Guard configuration
  - Configure Flashback Database
  - Explain the functionality of replicated restore points
  - Explain the functionality of automatic flashback

# Using Flashback Database in a Data Guard Configuration

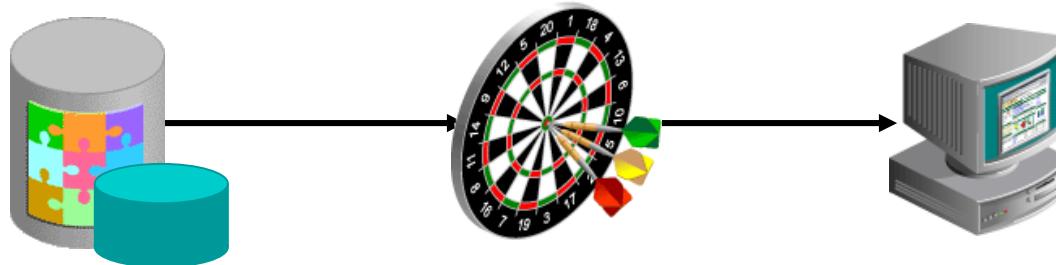
- Flashback Database provides the following in a Data Guard configuration:
  - An alternative to restoring and recovering the primary database
  - A way to reinstate the primary database that was disabled as part of a failover to any standby database operation
  - An alternative to delaying the application 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

# Overview of Flashback Database

- The Flashback Database operation:
  - Works like a “rewind” button for the database
  - Can be used when users make logical data corruptions



# Configuring Flashback Database



1. Configure the fast recovery area.

2. Set the retention target.

3. Enable Flashback Database.

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

# Configuring Flashback Database by Using Enterprise Manager

- Verify that the database is in ARCHIVELOG mode:

**Media Recovery**

The database is currently in ARCHIVELOG mode. In ARCHIVELOG mode, hot backups and recovery to the latest time are possible, but you must provide space for archived redo log files. If you change the database to ARCHIVELOG mode, you should perform a backup immediately. In NOARCHIVELOG mode, only cold backups are possible and data may be lost in the event of database corruption.

ARCHIVELOG Mode\*

Log Archive Filename Format\*

Number	Archived Redo Log Destination	Status	Type
1	USE_DB_RECOVERY_FILE_DEST	VALID	Local
2	(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host03.example.com)(PC	VALID	Remote

[Add Another Row](#)

 **TIP** It is recommended that archived redo log files be written to multiple locations spread across the different disks.  
 **TIP** You can specify up to 10 archived redo log destinations.

# Configuring Flashback Database by Using Enterprise Manager

- Set the fast recovery area and enable Flashback Database:

**Fast Recovery**

This database is using a fast recovery area. The chart shows space used by each file type that is not reclaimable by Oracle. Performing backups to tertiary storage is one way to make space reclaimable. Usable Fast Recovery Area includes free and reclaimable space.

Fast Recovery Area Location /u03/app/oracle/fast\_recovery\_area

Fast Recovery Area Size 15000 MB

Non-reclaimable Fast Recovery Area (MB) 645.64

Reclaimable Fast Recovery Area (MB) 200

Free Fast Recovery Area (GB) 13.82

Enable Flashback Database

Flashback database can be used for fast database point-in-time recovery, as it returns the database to a prior point-in-time without restoring files. Flashback is the preferred point-in-time recovery method in the recovery wizard when appropriate. The fast recovery area must be set to enable flashback database.

Flashback Retention Time 24 Hours

Current size of the flashback logs(MB) 400

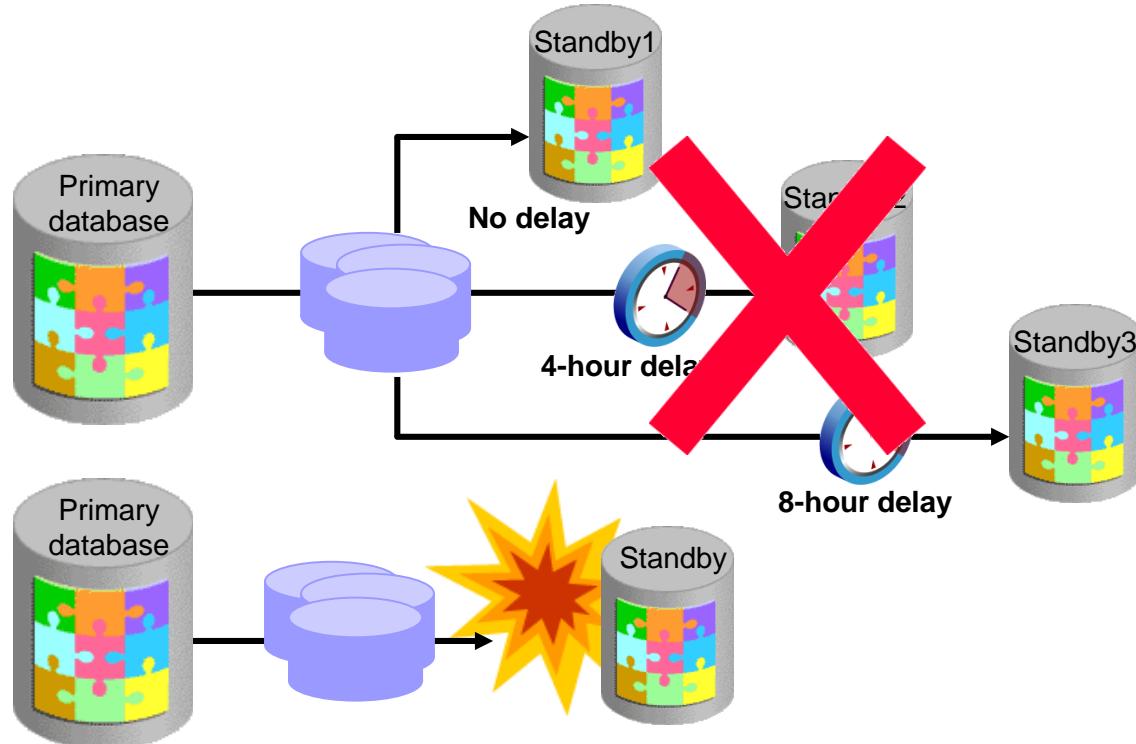
Lowest SCN in the flashback data 2750323

Flashback Time May 31, 2020 10:44:32 PM

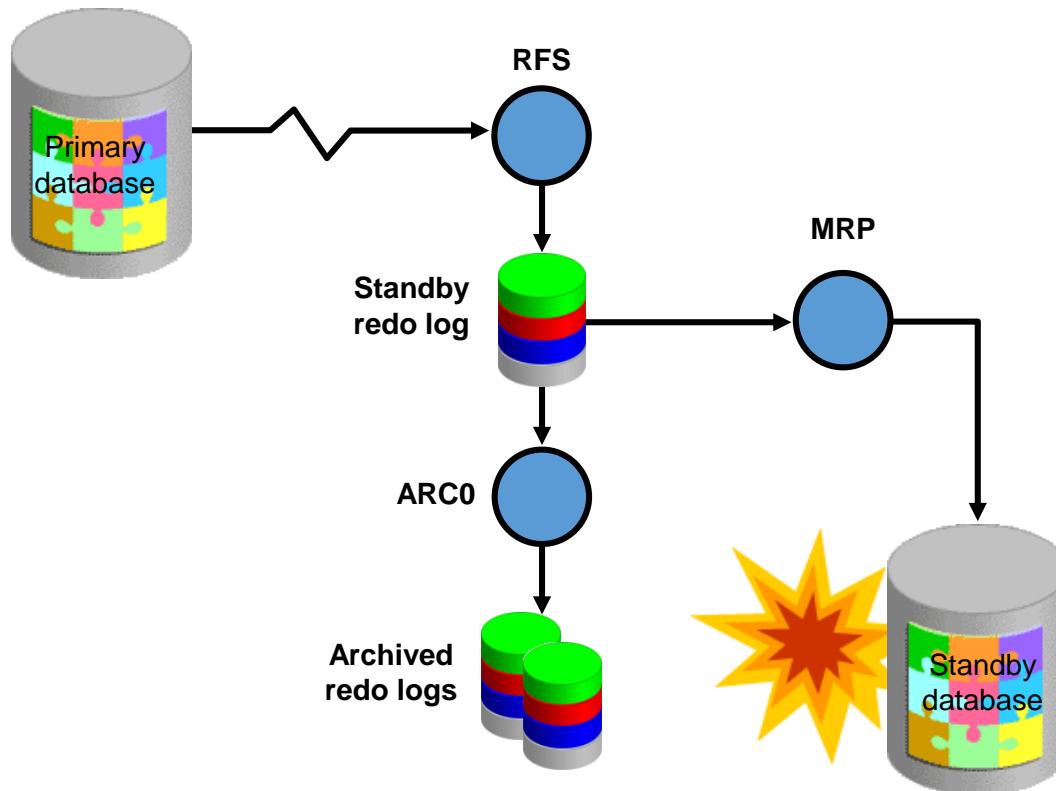
**Fast Recovery Area Usage**

File Type	Size (GB)	Percentage
Archived Redo Log	0.42	2.8%
Flashback Log	0.2	1.3%
Backup Piece	0.02	0.1%
Control File	0	0%
Redo Log	0	0%
Image Copy	0	0%
Auxiliary Datafile Copy	0	0%
Usable	14.02	95.7%

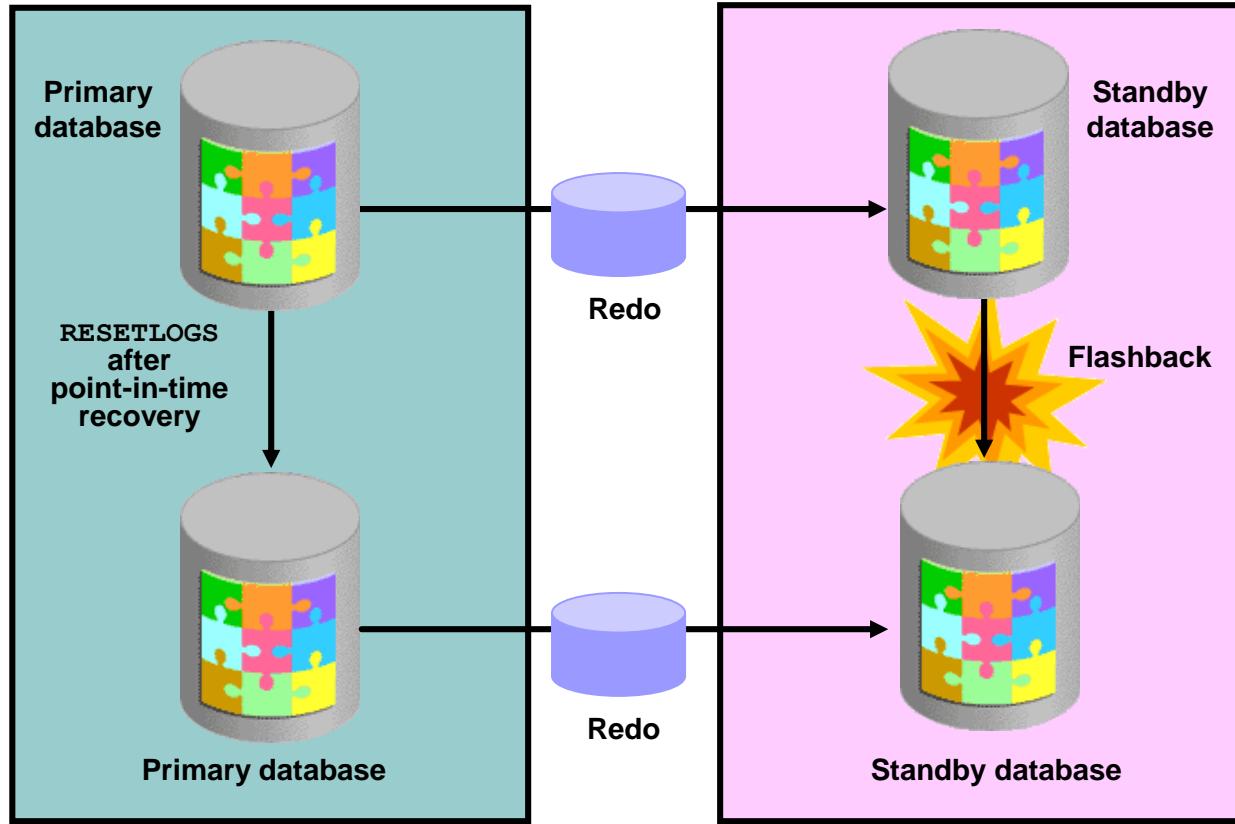
# Using Flashback Database Instead of Apply Delay



# Using Flashback Database and Real-Time Apply



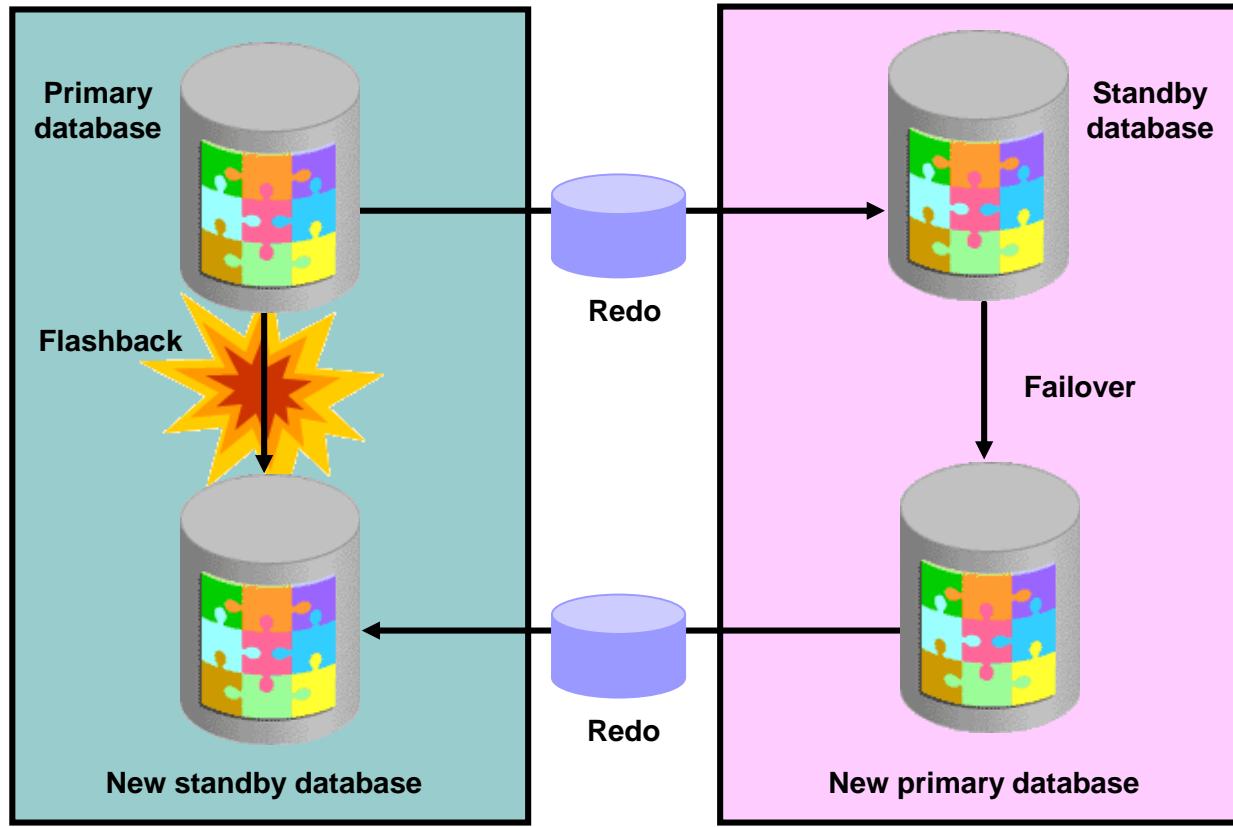
# Using Flashback Database After RESETLOGS



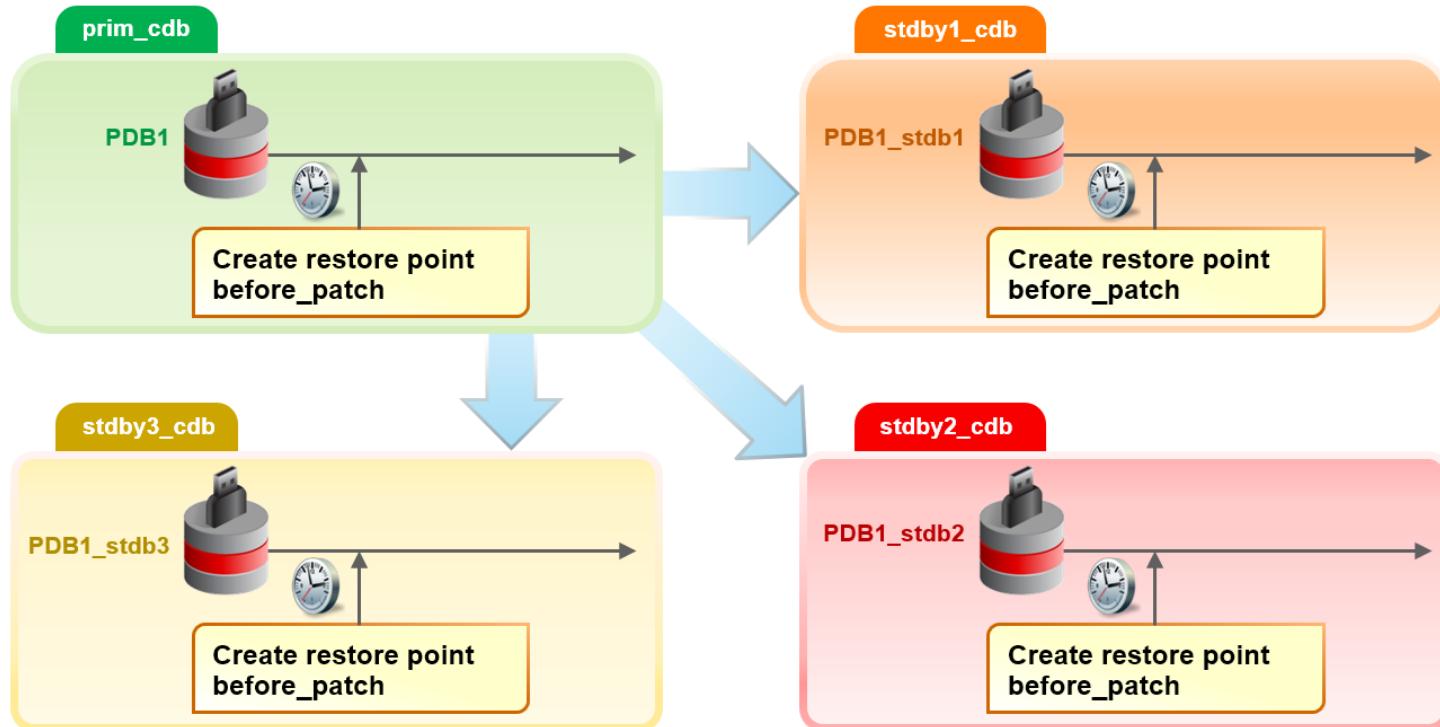
# Flashback Through Standby Database Role Transitions

- Use Flashback Database to flash back a database to a point in time before a switchover or failover.
- Primary and standby databases retain their current roles when you flash back through physical standby switchovers or failovers.
- Database roles are flashed back when you flash back through logical standby switchovers or failovers.
- Flashback Database can be used to undo a physical database activation.

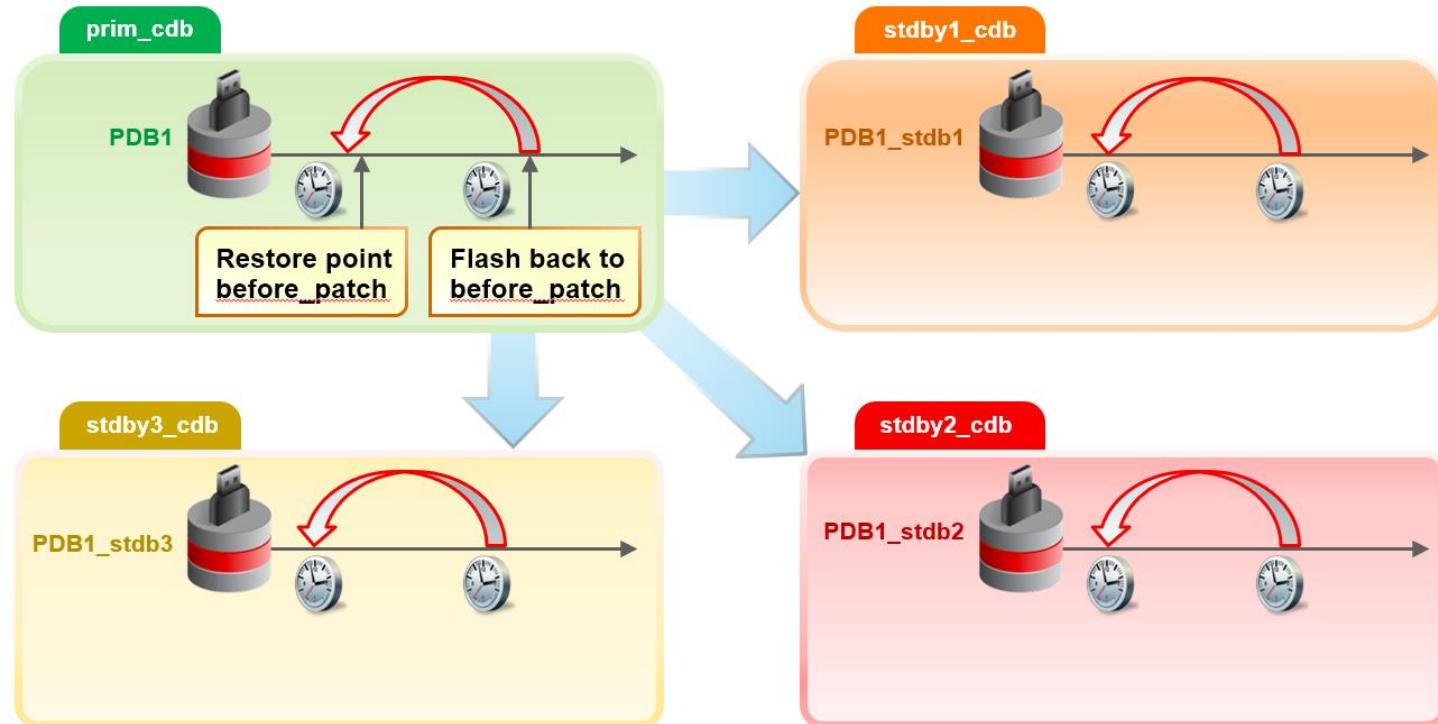
# Using Flashback Database After Failover



# Automatic Propagation of Restore Points to All Standby Databases



# Automatic Flashback on Physical Standby Databases



# Quiz

- Flashback Database can be performed only on databases that are in the primary role.
  - True
  - False

# Quiz

- When a user issues a flashback on a primary database, DBA is responsible for manually issuing a flashback or restoring the data files prior to the RESETLOGS.
  - True
  - False

# Summary

- In this lesson, you should have learned how to:
  - Enable Flashback Database
  - Use Flashback Database effectively in a Data Guard configuration
  - Explain the functionality of replicated restore points
  - Explain the functionality of automatic flashback

# Practice 15: Overview

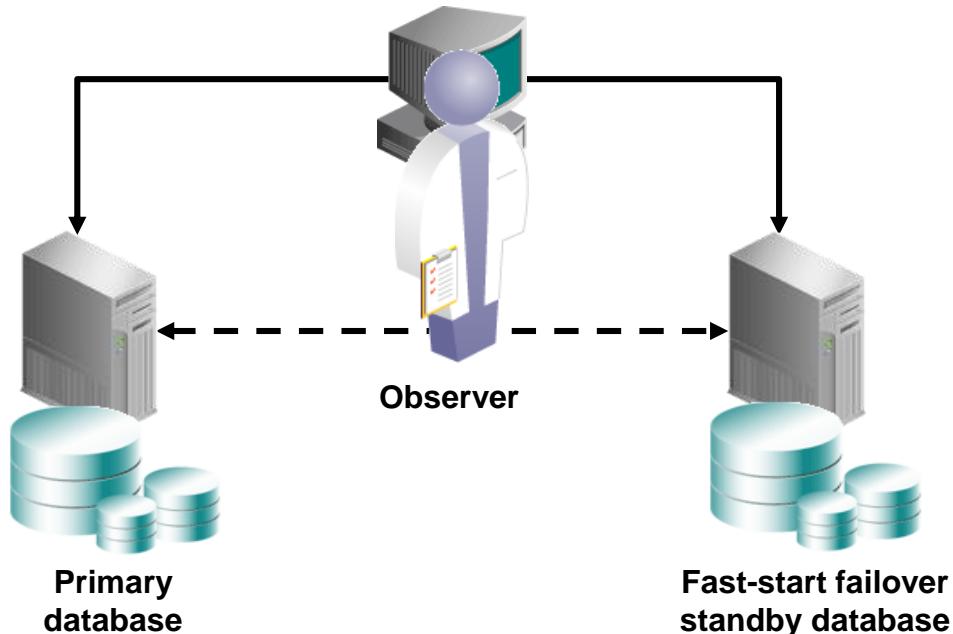
- This practice covers the following topics:
  - Configuring Flashback Database on the Primary Database
  - Configuring Flashback Database on the Physical Standby Database
  - Configuring Flashback Database on the Logical Standby Database
  - Testing Automatic Flashback of Standby Database

# 16. Enabling Fast-Start Failover

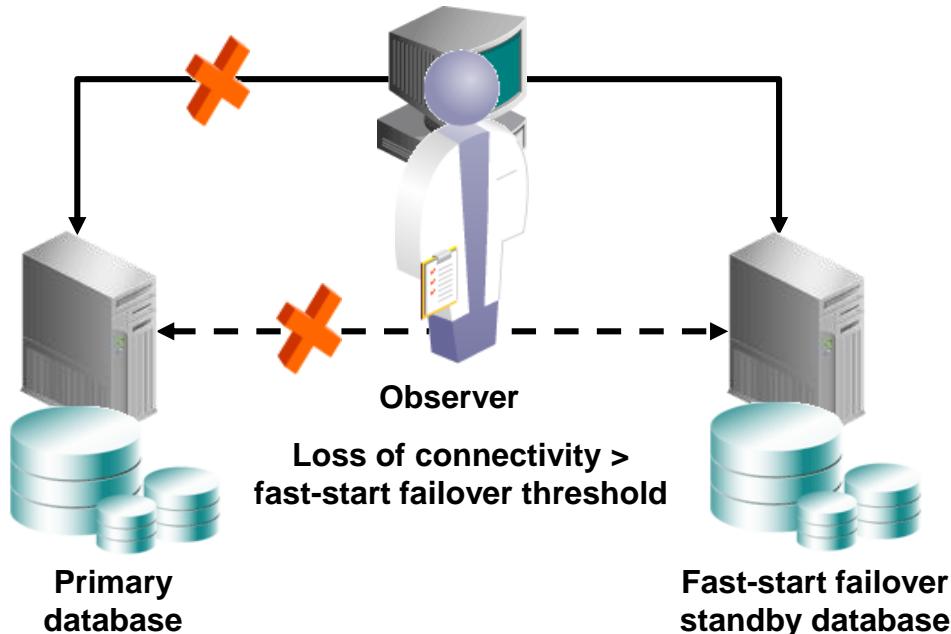
# Objectives

- After completing this lesson, you should be able to:
  - Configure fast-start failover
  - View information about the fast-start failover configuration
  - Manage the observer
  - Perform role changes in a fast-start failover configuration
  - Manually reinstate the primary database

# Fast-Start Failover: Overview



# When Does Fast-Start Failover Occur?



# Installing the Observer Software

- The observer is a separate OCI client-side component that monitors the availability of the primary database.
- Install observer software on a different computer from the primary and standby databases.
- Manage the observer by using Oracle Enterprise Manager or DGMGRL commands.



Observer

# Fast-Start Failover Prerequisites

- The following prerequisites must be met to enable fast-start failover:
  - Configure the protection mode.
  - LogXptMode property of target must be set as follows:
    - SYNC in Maximum protection mode
    - SYNC or FASTSYNC in Maximum availability mode
    - ASYNC in Maximum performance mode
  - Flashback Database must be enabled on the primary database and the pre-selected target standby database(s).
  - Configure tnsnames.ora entries for the observer.
  - Create a static service name so that the observer can automatically restart databases.

# 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.

# Step 1: Specify the Target Standby Databases

- Fast-start failover allows the broker to fail over to a previously chosen standby in the event of loss of the primary database.
- If the primary has multiple standbys, you can specify multiple fast-start failover targets by using `FastStartFailoverTarget`.
- The standby targets are referred to as **candidate** targets.
- The broker selects a target based on the order in which the targets are specified in the `FaststartFailoverTarget` property.

```
DGMGRL> EDIT DATABASE boston SET PROPERTY  
FastStartFailoverTarget='london3, london4' ;
```

# Dynamically Change Fast-start Failover Target

- SET FAST\_START FAILOVER TARGET enables you to set the fast-start failover target to the named standby database without:
  - Disabling fast-start failover
  - Modifying the fast start failover list
- The syntax for the SET FAST\_START FAILOVER TARGET is as follows:

```
SET FAST_START FAILOVER TARGET TO database-name [NOWAIT];
```

- The following example shows how to set the fast-start failover target to the standby database named Boston:

```
DGMGRL> SET FAST_START FAILOVER TARGET TO Boston;
Changing fast-start failover target to 'Boston'...
Succeeded.
```

# Setting a Fast Start Failover Target Using NOWAIT Mode

- The NOWAIT clause specifies that the fast-start failover target be updated immediately.

```
DGMGRL> SHOW FAST_START FAILOVER;
...
Active Target: Nashua
Potential Targets: "Nashua, Boston"
...
DGMGRL> SET FAST_START FAILOVER TARGET TO Boston NOWAIT;
Fast-start failover target switch to "Boston" requested.

DGMGRL> SHOW FAST_START FAILOVER;
...
Active Target: Boston
Potential Targets: "Nashua, Boston"
...
```

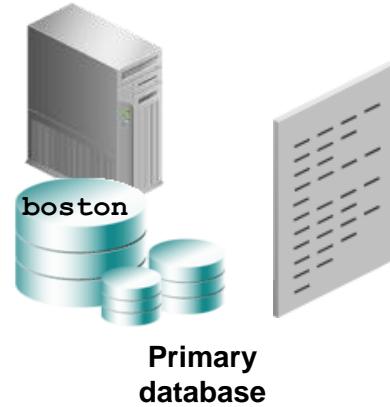
# Step 2: Set the Protection Mode

```
DGMGRL> EDIT DATABASE boston SET PROPERTY LogXptMode=SYNC;  
DGMGRL> EDIT DATABASE london SET PROPERTY LogXptMode=SYNC;  
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxProtection;
```

```
DGMGRL> EDIT DATABASE boston SET PROPERTY LogXptMode=SYNC;  
DGMGRL> EDIT DATABASE london SET PROPERTY LogXptMode=SYNC;  
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
```

```
DGMGRL> EDIT DATABASE boston SET PROPERTY LogXptMode=ASYNC;  
DGMGRL> EDIT DATABASE london SET PROPERTY LogXptMode=ASYNC;  
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxPerformance;  
DGMGRL> EDIT CONFIGURATION SET PROPERTY  
FastStartFailoverLagLimit=45;
```

# Step 3: Set the Fast-Start Failover Threshold



```
DGMGRL> EDIT CONFIGURATION SET PROPERTY  
FastStartFailoverThreshold = 15;
```

# Step 4: (Optional) Set Additional Fast-Start Failover Properties

- Additional fast-start failover properties can be set if desired. The optional properties include the following:
  - `FastStartFailoverLagLimit`
  - `FastStartFailoverPmyShutdown`
  - `FastStartFailoverAutoReinstate`
  - `ObserverConnectIdentifier`
  - `ObserverOverride`
  - `ObserverReconnect`
- These parameters are covered in the following slides.

# Setting the Lag-Time Limit

- Specifies a data loss threshold representing the maximum amount of data that can be lost with fast-start failover still being possible.
- Set the `FastStartFailoverLagLimit` property to configure a lag-time limit for a configuration in maximum performance mode:

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverLagLimit = 30;
```

failover target standby databases.

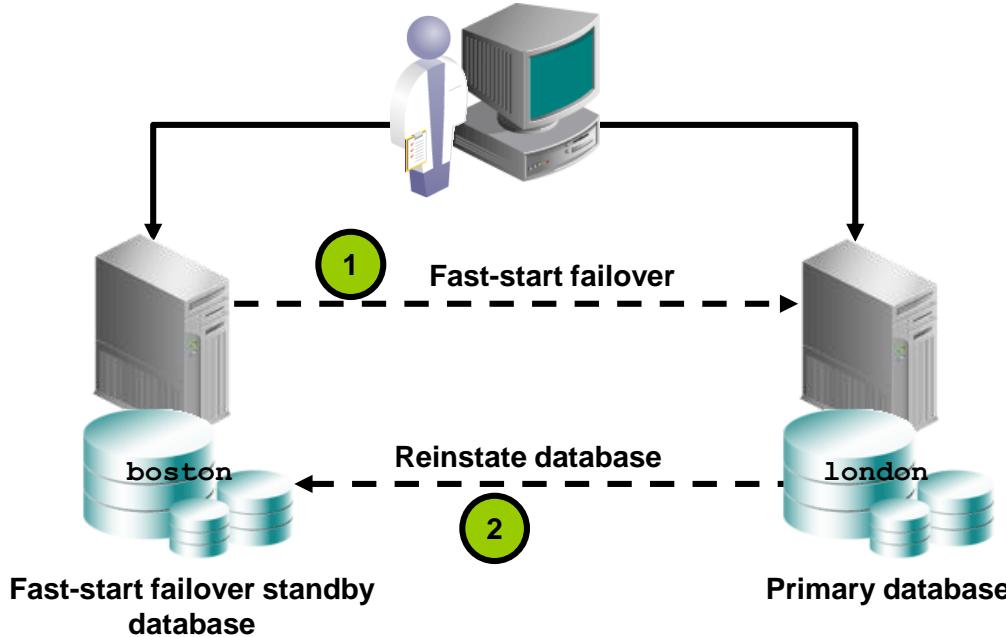
- Real-time apply must be enabled on the standby database.
- Fast-start failover cannot be enabled in maximum performance mode with a Far Sync.

# Configuring the Primary Database to Shut Down Automatically

- Use `FastStartFailoverPmyShutdown` to control whether the primary database shuts down if redo generation stalls and the primary database loses connectivity with the observer and target standby database for a longer time than the value of `FastStartFailoverThreshold`.
- Default value: TRUE

```
DGMGRIL> EDIT CONFIGURATION SET PROPERTY  
FastStartFailoverPmyShutdown = FALSE;
```

# Automatic Reinstatement After Fast-Start Failover



# Configuring Automatic Reinstatement of the Primary Database

- Use the `FastStartFailoverAutoReinstate` property to determine whether the old primary database should be automatically reinstated if a fast-start failover was initiated because of a loss of connectivity.
- Default value: `TRUE`

```
DGMGRl> EDIT CONFIGURATION SET PROPERTY  
FastStartFailoverAutoReinstate = FALSE;
```

# Setting a Connect Identifier for the Observer

- Set the `ObserverConnectIdentifier` database property to specify how the observer should connect to and monitor the primary and standby databases:

```
DGMGRL> EDIT DATABASE boston SET PROPERTY  
ObserverConnectIdentifier = 'boston.example.com' ;
```

- The default connect identifier is the value of the `DGConnectIdentifier` property.

# Setting an Observer Override

- Set the `ObserverOverride` property to allow an automatic failover to occur when the observer has lost connectivity to the primary, even if the standby can connect to the primary:

```
DGMGRl> EDIT CONFIGURATION SET PROPERTY  
ObserverOverride = TRUE;
```

- The default value of the `ObserverOverride` property is false.

# Setting Observer Reconnection Frequency

- Set the `ObserverReconnect` property to specify how often the observer establishes a new connection to the primary database:

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY ObserverReconnect = 15;
```

- The default value of the `ObserverReconnect` property is zero, preventing periodic new connection attempts.
- New connection attempts are necessary to discover if the primary database is not reachable.
- The unit of measurement is seconds.

# Step 5: Configure Additional Fast-Start Failover Conditions

- Use ENABLE/DISABLE FAST\_START FAILOVER commands to specify conditions for fast-start failover:

```
ENABLE FAST_START FAILOVER CONDITION "Stuck Archiver";
```

Observer initiates a fast start failover without waiting for

FastStartFailoverThreshold to expire.

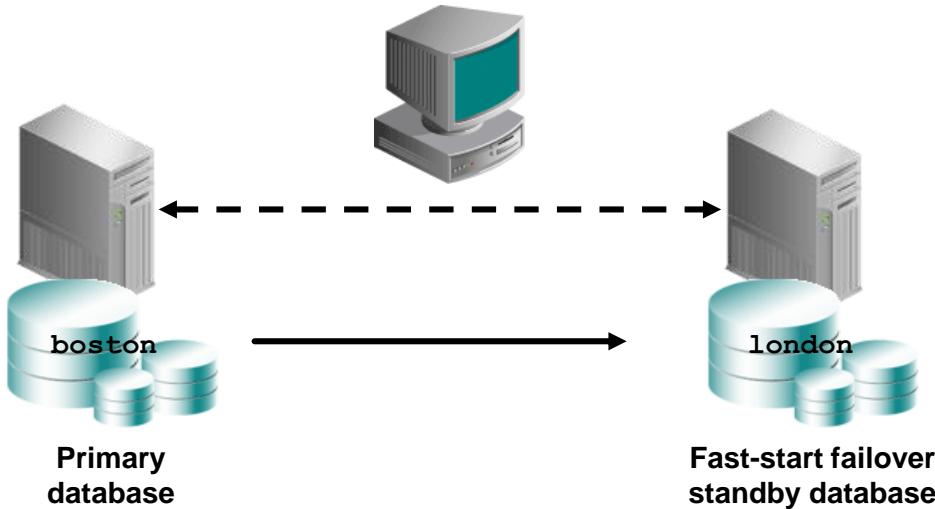
- Configurable conditions:
  - Conditions detectable through database health check
  - Errors raised by the Oracle server
- Use the SHOW FAST\_START FAILOVER command to obtain a list of valid conditions.

# Configuring Fast-Start Failover Conditions

- Specify additional conditions for a fast-start failover:

Health Condition	Default Value	Description
Datafile Offline	ENABLED	Data file offline due to a write error
Corrupted Controlfile	ENABLED	Corrupted control file
Corrupted Dictionary	ENABLED	Dictionary corruption of a critical database object
Inaccessible Logfile	DISABLED	LGWR unable to write to any member of a log group due to an I/O error
Stuck Archiver	DISABLED	Archiver unable to archive a redo log because the device is full or unavailable

# Step 6: Enable Fast-Start Failover



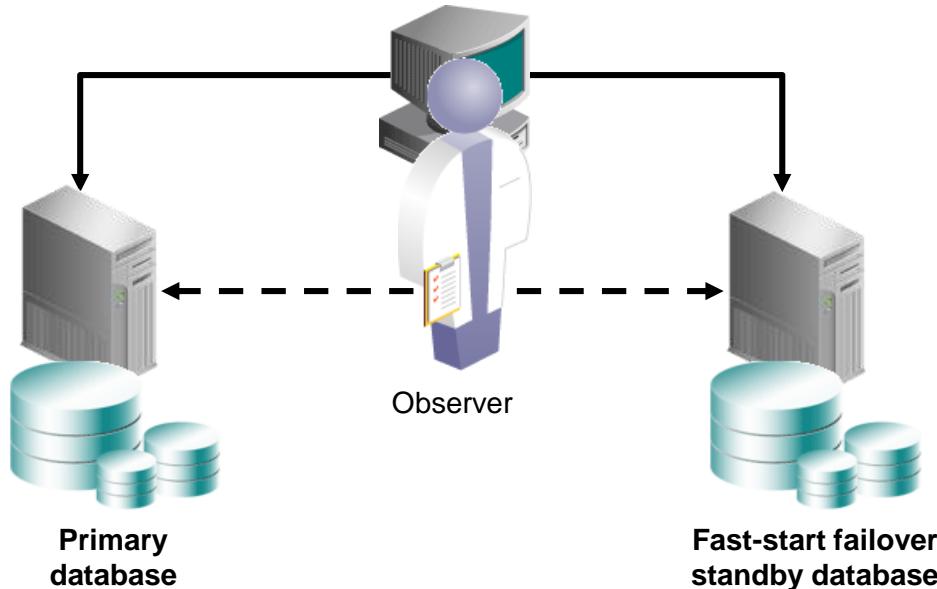
```
DGMGRL> ENABLE FAST_START FAILOVER;
```

# Configuring Fast-Start Failover in Observe-only Mode

- The observe-only mode for fast-start failover enables you to test how fast-start failover will work in your environment.
  - Observe-only mode has no impact on your current configuration or on applications.
  - In this mode, no actual changes are made to your Broker configuration.
- When the conditions for fast-start failover are met, the Broker adds messages to the observer log and broker log, indicating that fast-start failover would have been initiated.
- To configure fast-start failover in observe-only mode:

```
DGMGRl> ENABLE FAST_START FAILOVER OBSERVE ONLY;
```

# Step 7: Start the Observer



```
DGMGRL> START OBSERVER;
```

# Data Guard Broker Property: ConfigurationWideServiceName

- This property provides a single, automatically started service to connect to any primary or standby database in a broker configuration.
- The name defaults to the DB\_UNIQUE\_NAME initialization parameter of the primary database with \_CFG appended.
- ConfigurationWideServiceName is required to implement Simplified Observer Management.
- To set ConfigurationWideServiceName:

```
DGMGRl> edit configuration set property  
ConfigurationWideServiceName = 'BOSTON_CFG' ;
```

- To show the value of ConfigurationWideServiceName:

```
DGMGRl> show configuration verbose ConfigurationWideServiceName ;  
ConfigurationWideServiceName = 'BOSTON_CFG'
```

# Multiple Observers Using a Single Configuration

- You can now start observers on multiple hosts to manage a single Data Guard broker configuration.
- If an observer fails, a backup observer will take its place so that the configuration is never in an unobserved state.
- You can register up to three observers to monitor a single Data Guard broker configuration.
- Each observer is identified by a name that you supply when you issue the START OBSERVER command.
- The SHOW FAST\_START FAILOVER command shows a list of registered observers and indicates which one is the master.
- The SHOW OBSERVER command shows detailed information about the registered observers.

# Master Observer and Backup Observers

- When fast-start failover is enabled, the primary and standby randomly choose one of the observers to be the master.
- If there are no registered observers when fast-start failover is enabled, the first observer started is designated as the master.
- Only the master observer can coordinate fast-start failover with the Data Guard broker.
- All other registered observers are considered to be backup observers.
- You can use the SET MASTEROB SERVER command to change the observer that is the master.

```
DGMGRL> SET MASTEROB SERVER TO boston-observer;  
Succeeded.
```

# Starting Observers as Background Processes

- To run an observer as a background process, use the DGMGRL command START OBSERVER IN BACKGROUND.
- Provide a connect identifier through which one or more databases in a specific broker configuration can be reached.

```
DGMGRL> START OBSERVER observer1 IN BACKGROUND
FILE IS /net/sales/dat/oracle/broker/fsfo.dat
LOGFILE IS /net/sales/dat/oracle/broker/observer.log
CONNECT IDENTIFIER IS sales_p;
Submitted command "START OBSERVER" using connect identifier "sales_p"
```

- This command uses Oracle wallet to obtain the credentials to log in to the database server and register observers.

# Step 8: Verify the Configuration

```
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

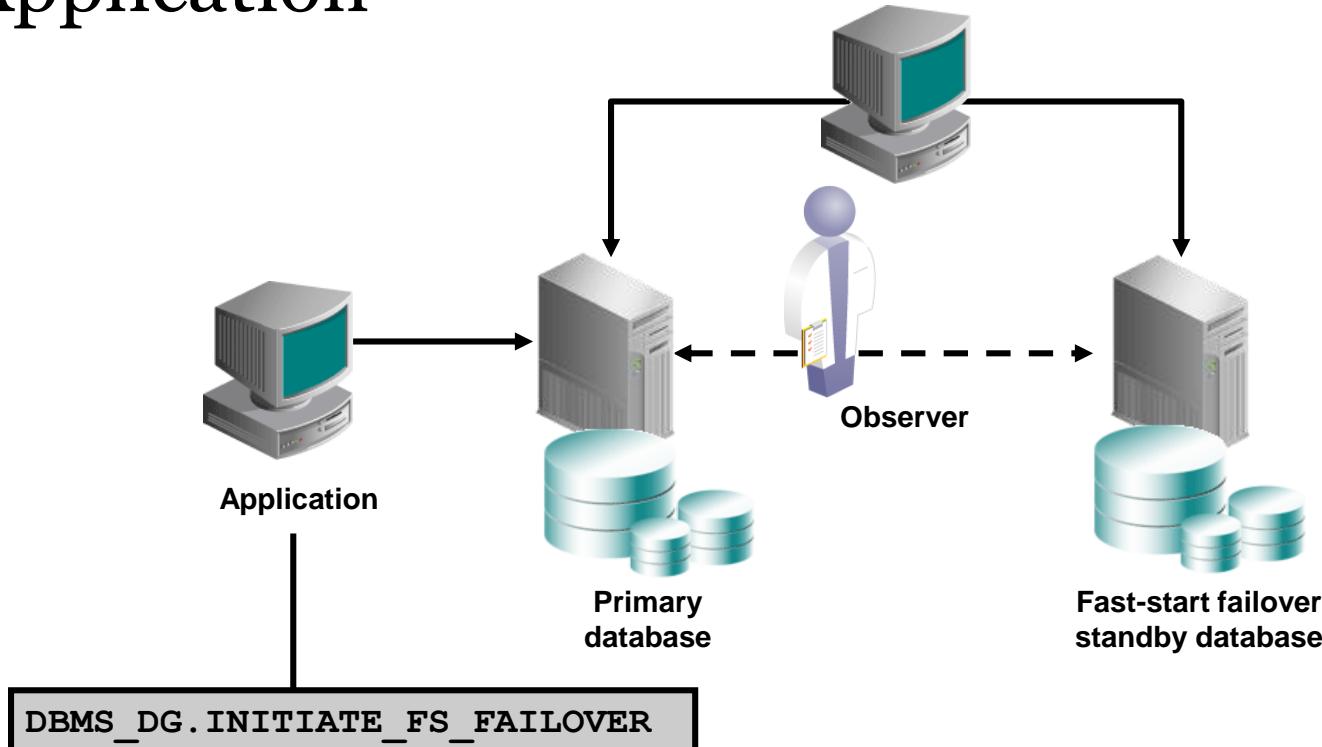
### Health Conditions:

Corrupted Controlfile	YES
Corrupted Dictionary	YES
Inaccessible Logfile	NO
Stuck Archiver	NO
Datafile Offline	YES

### Oracle Error Conditions:

(none)

# Initiating Fast-Start Failover from an Application



# Initiating Fast-Start Failover from an Application

- DBMS\_DG package contains the INITIATE\_FS\_FAILOVER function that is used to initiate a fast-start failover from an application:

```
FUNCTION dbms_dg.initiate_fs_failover  
(condstr IN VARCHAR2) RETURN BINARY_INTEGER;
```

- The DBMS\_DG package is defined as an invoker's rights package to address privilege concerns.

# Viewing Fast-Start Failover Information

```
SELECT fs_failover_status as STATUS,
       fs_failover_current_target as CURR_TGET,
       fs_failover_threshold as THRESHOLD,
       fs_failover_observer_present as OBS_PRES,
       fs_failover_observer_host as OBS_HOST
  FROM v$database;
```

STATUS	CURR_TGET	THRESHOLD	OBS_PRES	OBS_HOST
TARGET UNDER LAG LIMIT london		30	YES	host04.example.com

# Determining the Reason for a Fast-Start Failover

- Determine the reason for fast-start failover by querying the V\$FS\_FAILOVER\_STATS view:

```
SQL> SELECT last_failover_time, last_failover_reason  
      FROM v$fs_failover_stats;
```

LAST_FAILOVER_TIME	LAST_FAILOVER_REASON
--------------------	----------------------

-----	-----
-------	-------

10/17/2013 22:30:12	Primary Disconnected
---------------------	----------------------

# Prohibited Operations After Enabling Fast-Start Failover

- Many configuration changes are not allowed with fast-start failover enabled.



Configuration protection mode changes



Disable or delete the Far Sync

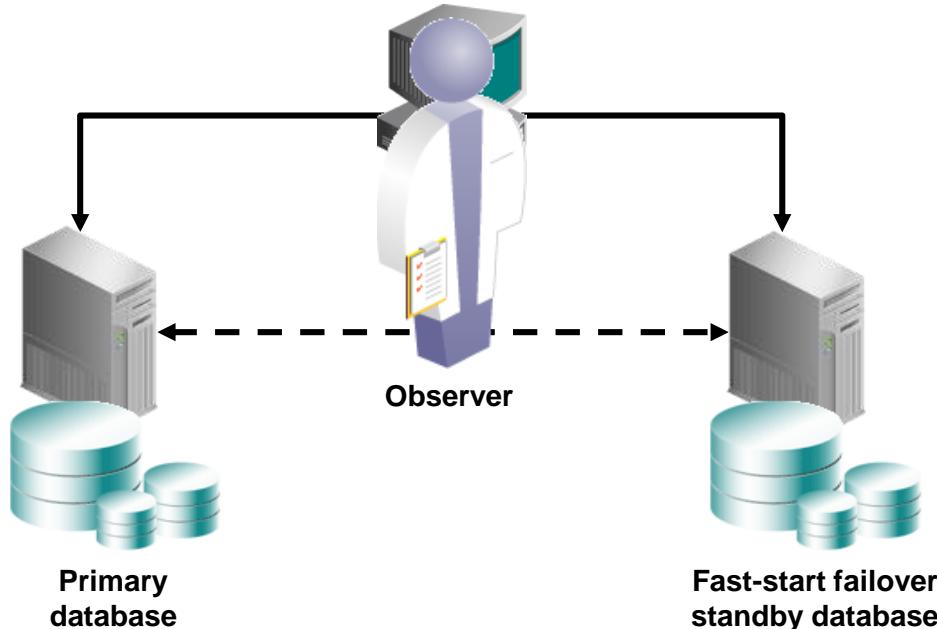


Disable or delete the fast-start failover standby database



```
DGMGRL> EDIT DATABASE SET PROPERTY LogXptMode;  
DGMGRL> EDIT DATABASE SET PROPERTY FastStartFailoverTarget;
```

# Disabling Fast-Start Failover



```
DGMGRL> DISABLE FAST_START FAILOVER [FORCE] ;
```

# Disabling Fast-Start Failover Conditions

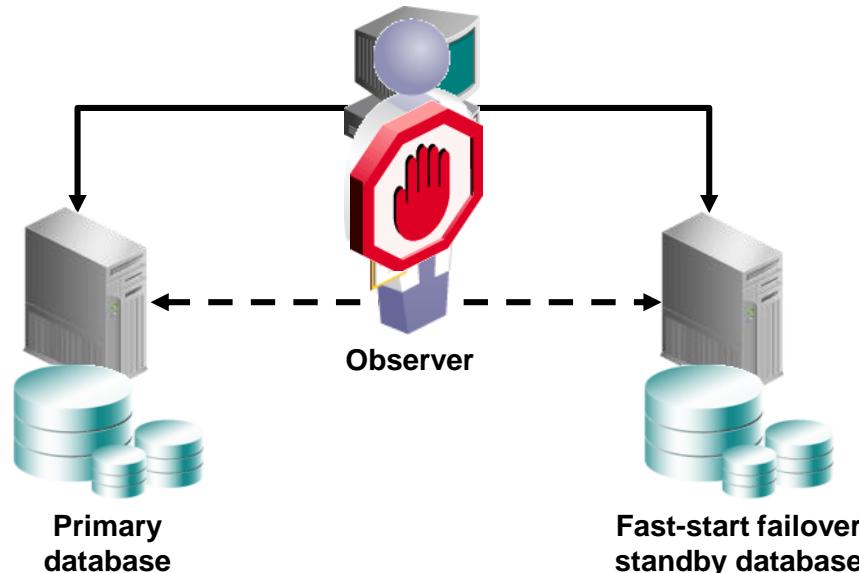
- Remove specific conditions for which a fast-start failover should be performed:

```
DGMGRl> DISABLE FAST_START FAILOVER CONDITION "Datafile Offline";
```

# Using the FORCE Option

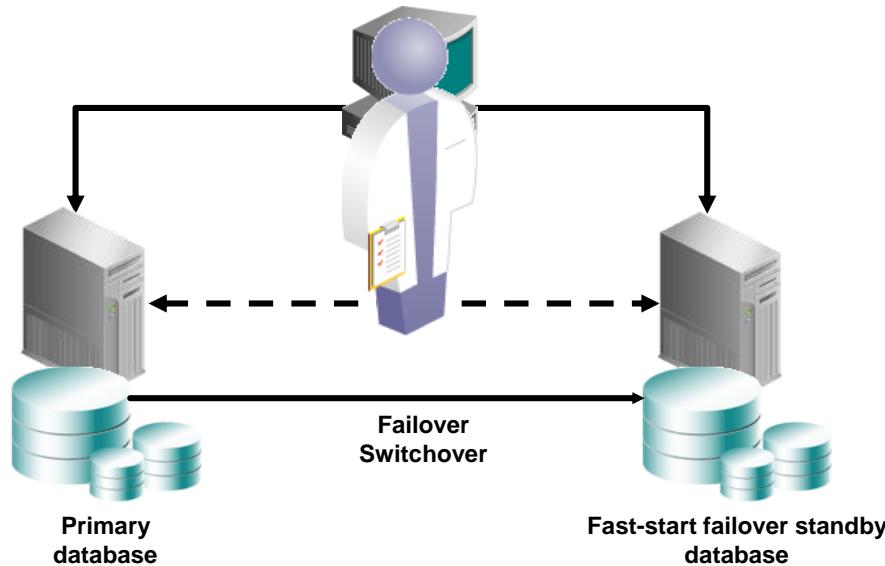
- Use the FORCE option in the following situations:
  - When the fast-start failover environment is synchronized and the primary has lost connectivity to the observer and the target standby database
  - To prevent a fast-start failover from occurring on the target standby database
  - To conduct a manual failover when the fast-start failover environment is unsynchronized

# Stopping the Observer

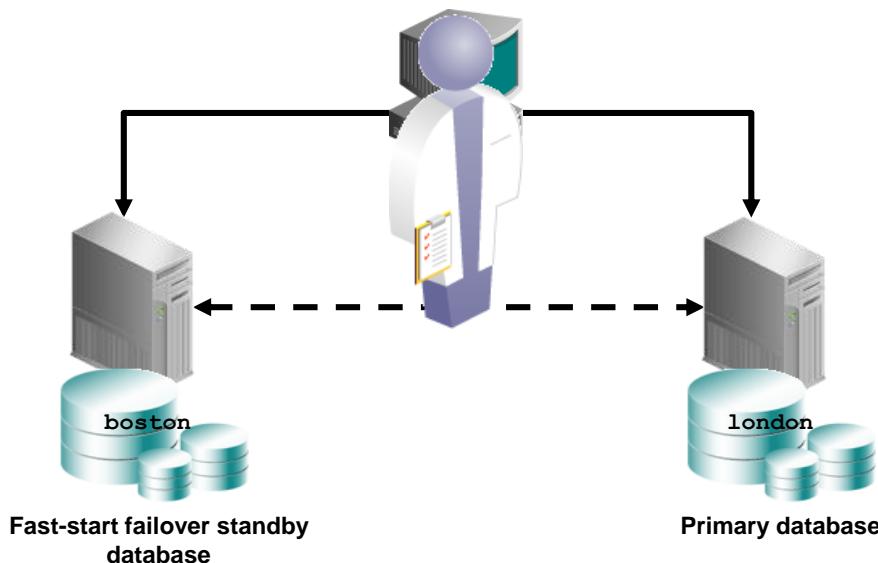


```
DGMGRL> STOP OBSERVER;
```

# Performing Manual Role Changes



# Manually Reinstating the Database



```
DGMGRL> REINSTATE DATABASE boston;
```

# Using Enterprise Manager to Enable Fast-Start Failover

**boston.example.com (Container Database)**

Logged in as sys host01.example.com

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

## Data Guard

Page Refreshed May 31, 2020 11:33:46 PM EDT

View Data Real Time: 30 Second Refresh

### Overview

Data Guard Status	✓ Normal
Protection Mode	Maximum Performance
Fast-Start Failover	Disabled

**Standby Database Progress Summary**

Transport lag is the time difference between the last update on the primary database and the last received redo on the standby database. Apply lag is the time difference between the last update on the primary database and the last applied redo on the standby database.

seconds

0.0 0.5 1.0

0 0

london.example.com

Legend: Transport Lag (blue), Apply Lag (orange)

**Primary Database**

Name	boston.example.com
Host	host01.example.com
Data Guard Status	✓ Normal
Current Log	3
Properties	Edit

**Standby Databases**

Add Far Sync | Add Standby Database

Edit | Remove | Swithchover | Failover | Convert

Select	Name	Host	Data Guard Status	Role	Redo Source	Real-time Query	Last Received Log	Last Applied Log	Estimated Failover Time
<input checked="" type="radio"/>	london.example.com	host03.example.com	✓ Normal	Physical Standby	boston	Disabled	2	2	Not available

# Using Enterprise Manager to Enable Fast-Start Failover

The screenshot shows the Oracle Enterprise Manager interface for configuring Fast-Start Failover. The top navigation bar includes links for Oracle Database, Performance, Availability, Security, Schema, and Administration. The current page is 'Data Guard > Fast-Start Failover: Configure'. A yellow box labeled 'Information' states: 'There is currently no observer for this configuration, nor has one been specified.' Below this, the title 'Fast-Start Failover: Configure' is displayed, along with 'Cancel' and 'Continue' buttons. The main section, 'Target Database Selection', asks to select a standby database for failover. It lists a single entry: 'Select Name: london.example.com, Role: Physical Standby, Redo Transport Mode: ASYNC'. A blue arrow points from the text 'Select the database.' to the 'Name' column of the table. The 'Observer' section below explains the requirement for an observer and provides a 'Configure Observer' link. A blue arrow points from the text 'Click Configure Observer.' to the 'Configure Observer' link.

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

Data Guard > Fast-Start Failover: Configure

**Information**  
There is currently no observer for this configuration, nor has one been specified.

**Fast-Start Failover: Configure**

**Target Database Selection**

Select a standby database to be the fast-start failover target. The redo transport mode for the selected database will be set to ASYNC (if not currently set to ASYNC).

Select	Name	Role	Redo Transport Mode
<input checked="" type="radio"/>	london.example.com	Physical Standby	ASYNC

**Observer**

Fast-start failover requires a Data Guard observer process. For highest availability, Oracle recommends that the observer be on a separate host from the primary and standby databases.

i Observer Location Not Set [Configure Observer](#)

Select the database.

Click Configure Observer.

# Using Enterprise Manager to Enable Fast-Start Failover

**Fast-Start Failover: Configure Observer**

**Observer Location**

There is currently no observer for this configuration. Select the discovered host and Oracle Home where Enterprise Manager will start the observer.

Observer Host

Observer Oracle Home

**Cancel** **OK**

**Failover Properties**

**Failover Threshold**   Amount of time the primary database must be out of contact with the observer and the standby database before a fast-start failover is initiated.

**Lag Limit**   Amount of time the standby database is allowed to fall behind the primary database, beyond which a fast-start failover will not be allowed.

User Configurable Failover Conditions [Edit](#)

**Primary Database Properties**

Automatically Reinstate Primary  Yes Controls whether the observer will automatically reinstate the former primary database once contact is re-established after the former primary database is restarted. Does not control reinstate behavior for failovers caused by an error condition.

Automatically Shutdown Primary  Yes Controls whether the primary database will shut itself down if it independently discerns that a fast-start failover may have occurred, but cannot verify it due to network isolation from the observer and the standby database. Does not control shutdown behavior for failovers caused by an error condition.

**Cancel** **Continue**

# Using Enterprise Manager to Enable Fast-Start Failover

The screenshot shows the Oracle Enterprise Manager interface for a container database named 'boston.example.com'. The top navigation bar includes the database name, a help icon, and a log-in status message: 'Logged in as sys' with a lock icon, and 'host01.example.com'. Below the navigation, there is a horizontal menu with tabs: 'Oracle Database' (selected), 'Performance', 'Availability', 'Security', 'Schema', and 'Administration'. The main content area displays a confirmation dialog titled 'Confirmation: Enable Fast-Start Failover'. The dialog contains the following text:  
Are you sure you want to enable fast-start failover to database london.example.com?  
The observer will be started on host host01.example.com.  
Automatic observer restart will not be enabled on observer host host01.example.com.

# Using Enterprise Manager to Enable Fast-Start Failover

The screenshot shows a navigation bar at the top with links: Oracle Database ▾, Performance ▾, Availability ▾, Security ▾, Schema ▾, and Administration ▾. Below the navigation bar, a message box displays the following information:

**Processing: Enable Fast-Start Failover**  
Enabling fast-start failover to target database london.example.com.

This process takes some time. The page automatically returns to the Data Guard overview page upon completion.  
Click on the alert log link to view progress details in a new browser window. View alert log: [boston.example.com](#)

A progress bar is shown, with the first two items checked off:

- ✓ Preparing databases
- ⇒ Starting observer
- Enabling fast-start failover
- Waiting for process to complete

**TIP** This process cannot be cancelled. It will continue even if the browser window is closed.

# Changing the Protection Mode and Disabling Fast-Start Failover

The screenshot shows the Oracle Enterprise Manager Cloud Control 13c interface. The top navigation bar includes the ORACLE logo, the product name, and various system icons like a gear, magnifying glass, and bell. The main header displays the host name "boston.example.com (Container Database)" and the user "sys". The menu bar below the header lists categories: Oracle Database, Performance, Availability, Security, Schema, and Administration.

A central modal dialog box is open, titled "Confirmation: Change Protection Mode". It contains a message: "If you change the protection mode from Maximum Performance, fast-start failover will be disabled and re-enabled. Are you sure you want to change the protection mode to Maximum Availability?" At the bottom right of the dialog are two buttons: "No" and "Yes".

# Using Enterprise Manager to Disable Fast-Start Failover

**Data Guard**  
Page Refreshed May 31, 2020 11:57:13 PM EDT

**Overview**

Data Guard Status	✓ Normal
Protection Mode	Maximum Performance
Fast-Start Failover	Enabled to london.example.com
Observer Location	host01.example.com

**Primary Database**

Name	boston.example.com
Host	host01.example.com
Data Guard Status	✓ Normal
Current Log	3
Properties	Edit

Click the Enabled link to access the Change Mode page.

**Data Guard > Fast-Start Failover: Change Mode**  
**Fast-Start Failover: Change Mode**

Fast-start failover is currently enabled. Choose whether to edit fast-start failover properties or disable fast-start failover, then click Continue.

Edit properties  
Change the fast-start failover target database and threshold, and start or restart the observer process.

Disable  
Disable fast-start failover. Does not stop the observer unless the Stop Observer option is selected.

Stop All Observers  
To temporarily suspend fast-start failover, select Disable but do not select the Stop All Observers option.

**Cancel** **Continue**

**Cancel** **Continue**

# Using Enterprise Manager to Suspend Fast-Start Failover

The screenshot shows a web-based interface for managing Oracle Database Fast-Start Failover. At the top, there's a navigation bar with tabs: Oracle Database, Performance, Availability, Security, Schema, and Administration. Below the navigation bar, the path is Data Guard > Fast-Start Failover: Change Mode. The main title is "Fast-Start Failover: Change Mode". On the right, there are "Cancel" and "Continue" buttons. The central content area contains a message: "Fast-start failover is currently enabled. Choose whether to edit fast-start failover properties or disable fast-start failover, then click Continue." Below this, there are two radio button options: "Edit properties" and "Disable". The "Disable" option is selected. Under "Disable", there is a checkbox labeled "Stop All Observers" which is unchecked. A note next to it says: "To temporarily suspend fast-start failover, select Disable but do not select the Stop All Observers option." A blue arrow points from the text "Do not stop observer for suspension only." at the bottom left to the "Stop All Observers" checkbox. The entire "Do not stop observer for suspension only." text is highlighted with a yellow box.

Oracle Database

Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

Data Guard > Fast-Start Failover: Change Mode

**Fast-Start Failover: Change Mode**

Fast-start failover is currently enabled. Choose whether to edit fast-start failover properties or disable fast-start failover, then click Continue.

Edit properties  
Change the fast-start failover target database and threshold, and start or restart the observer process.

Disable  
Disable fast-start failover. Does not stop the observer unless the Stop Observer option is selected.

Stop All Observers  
To temporarily suspend fast-start failover, select Disable but do not select the Stop All Observers option.

Cancel Continue

Do not stop observer for suspension only.

# Moving the Observer to a New Host

- To move the fast-start failover observer to a new host:
  1. Execute the `STOP OBSERVER` command to sever the link between the original observer and the broker configuration.
  2. Execute the `SHOW CONFIGURATION VERBOSE` and `SHOW DATABASE` commands to verify that the observer is stopped.
  3. Start the observer on your new host.

# Quiz

- Which protection mode can be used to enable fast-start failover?
  - a.Maximum Protection
  - b.Maximum Availability
  - c.Maximum Performance
  - d.All of the above

# Quiz

- You can configure multiple active observers in a single Data Guard broker configuration to coordinate fast-start failover with the Data Guard broker.
  - a.True
  - b.False

# Quiz

- When you configure fast-start failover in Observe-only mode, the Data Broker adds messages to the observer log and broker log, indicating that fast-start failover would have been initiated.
  - a.True
  - b.False

# Summary

- In this lesson, you should have learned how to:
  - Configure fast-start failover
  - View information about the fast-start failover configuration
  - Manage the observer
  - Perform role changes in a fast-start failover configuration
  - Manually reinstate the primary database

# Practice 16: Overview

- This practice covers the following topics:
  - Enabling Fast-Start Failover in Observer-only mode
  - Enabling Fast-Start Failover
  - Testing Fast-Start Failover
  - Switchover to Reinstated Database

# **17. Backup and Recovery Considerations in an Oracle Data Guard Configuration**

# Objectives

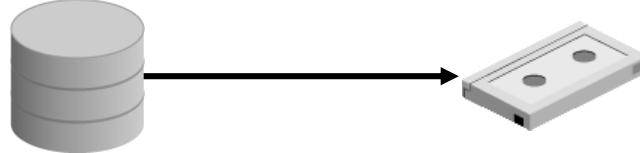
- After completing this lesson, you should be able to:
  - Use Recovery Manager (RMAN) to back up and restore files in a Data Guard configuration
  - Offload backups to a physical standby database
  - Recover your primary database by using a file from the physical standby database
  - Recover your primary database over the network
  - Synchronize Standby Database from Primary Database with one command

# Using RMAN to Back Up and Restore Files in a Data Guard Configuration

- RMAN can be used to back up and recover databases in a Data Guard configuration.
- Physical Standby database backups are usable on the primary database.
- Backups of logical standby databases are not usable at the primary database.
- The recovery catalog is required when using RMAN in a Data Guard configuration for physical standby databases.
- RMAN and Data Guard best practices:
  - Perform backups at the primary and standby databases to reduce mean time to recover (MTTR).
  - Take daily incremental backups.

# Offloading Backups to a Physical Standby

- Backups of data files and archived redo logs are fully interchangeable.
- Backups of standby control files and nonstandby control files are interchangeable.
- SPFILE backups are not interchangeable.
- Primary and standby databases must use the same recovery catalog.
- It is not necessary to register the standby database.



# Restrictions and Usage Notes

- You must back up a physical standby database.
- You must be connected to the recovery catalog when backing up.
- You must connect to the physical standby database with the TARGET keyword.



# Association and Accessibility of RMAN Backups

- Every database file or backup file is associated with a `DB_UNIQUE_NAME`.
- The database that creates a file is associated with the file.
- Disk backups are accessible only to the database with which it is associated.
- Tape backups created on one database are accessible to all databases.
- Association of an existing backup file can be changed with the `CHANGE ... RESET DB_UNIQUE_NAME` command.
- A copied backup file from one host to another can be associated with another database with the `CATALOG` command.

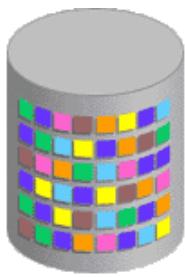
# Backup and Recovery of a Logical Standby Database

- Backup considerations:
  - Use the same backup method that you use for your primary database.
  - Files are not interchangeable with the primary database.
- Recovery considerations:
  - If you lose individual files, perform recovery as you would for any other database.
  - Re-create the logical standby database if you lose the entire database.
  - Use a binary copy of the control file for control-file recovery.

# Using the RMAN Recovery Catalog in a Data Guard Configuration

- Use of the RMAN recovery catalog permits backups taken on one database server to be restored to another database server.
- RMAN uses the recovery catalog metadata to behave transparently across different physical databases in the Data Guard configuration.
- Configure a separate server for the recovery catalog so that a disaster at any site in the Data Guard configuration does not affect the ability to recover from the latest backups.

# Creating the Recovery Catalog



**1. Configure the recovery catalog database.**



**2. Create the recovery catalog owner.**



**3. Create the recovery catalog.**

# Registering a Database in the Recovery Catalog

- Only the primary database is explicitly registered:

```
$ rman TARGET / CATALOG username/password@net_service_name
RMAN> REGISTER DATABASE;
```

- A standby database is automatically registered when you connect to it or when the CONFIGURE DB\_UNIQUE\_NAME command is executed.
- RMAN performs the following actions:
  - Creates rows in the recovery catalog tables for the target database
  - Copies data from the target database control file to the recovery catalog tables

# Setting Persistent Configuration Settings

- Use the CONFIGURE command with the FOR DB UNIQUE NAME clause to create a persistent configuration for a database in a Data Guard configuration.
- You do not have to connect to the standby database or primary database as TARGET.
- RMAN must be connected to a recovery catalog when you create or alter a configuration for a database in the Data Guard configuration.

```
RMAN> configure controlfile autobackup on for db_unique_name boston;
new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored
```

# Setting RMAN Persistent Configuration Parameters on the Primary Database

- Configure the backup retention policy:

```
RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF <n> DAYS;
```

- Specify deletion of archived redo logs:

```
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO ALL STANDBY;  
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
```

- Configure the connect identifier for the primary database and the standby databases:

```
RMAN> CONFIGURE DB_UNIQUE_NAME boston CONNECT IDENTIFIER 'boston';  
RMAN> CONFIGURE DB_UNIQUE_NAME london CONNECT IDENTIFIER 'london'
```

# Setting RMAN Persistent Configuration Parameters on the Physical Standby Database

- Set the following parameters on the physical standby database where you will perform backups:
  - Enable automatic backup of the control file and the server parameter file:

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

Specifying the same controlfile and spfile is a valid backup with the same checkpoint:

```
RMAN> CONFIGURE BACKUP OPTIMIZATION ON;
```

Specify when the archived logs can be deleted.

```
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED  
UP 1 TIMES TO DEVICE TYPE DISK;
```

# Setting RMAN Persistent Configuration Parameters on the Other Standby Databases

- On the standby databases where you will *not* be performing backups, specify that the archived logs can be deleted after they are applied at the standby database:

```
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
```

# Enabling Block Change Tracking on a Physical Standby Database

- Enable block change tracking on a physical standby database for fast incremental backups.
- Data file blocks that are affected by each database update are tracked in a block change tracking file.
- The block change tracking file is a binary file used by RMAN to record changed blocks to improve incremental backup performance.
- To enable Block Change Tracking:

```
ALTER DATABASE{ENABLE|DISABLE} BLOCK CHANGE TRACKING[USING FILE '...']
```

# Configuring Daily Incremental Backups

- Configure a daily backup strategy:
  - Day 1: Create an incremental level 0 backup.
  - Day 2: Create an incremental level 1 backup.
  - Day 3: Create an incremental level 1 backup.  
The previous day's incremental backup is merged with the data file copy, and a current incremental backup is taken.
- A single RMAN script can be created to implement this strategy:

```
RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;
RECOVER COPY OF DATABASE WITH TAG 'dgbkup';
BACKUP DEVICE TYPE DISK INCREMENTAL LEVEL 1 FOR RECOVER
  OF COPY WITH TAG 'dgbkup' DATABASE;
BACKUP DEVICE TYPE SBT ARCHIVELOG ALL;
BACKUP BACKUPSET ALL;
DELETE ARCHIVELOG ALL;
```

# Recovering from the Loss of a Data File on the Primary Database

- Using backups
- Restoring and recovering files over the network

# Using a Backup to Recover a Data File on the Primary Database

- Using backups to restore and recover data files:

```
RMAN> CONNECT TARGET sys@boston  
RMAN> RESTORE DATAFILE 5,6;  
RMAN> RECOVER DATAFILE 5,6;
```

- Using backups to restore and recover tablespaces:

```
RMAN> CONNECT TARGET sys@boston  
RMAN> RESTORE TABLESPACE tbs_name1,tbs_name2;  
RMAN> RECOVER TABLESPACE tbs_name1,tbs_name2;
```

# Restoring and Recovering a Data File on the Primary Database Over the Network

1. Connect to the primary database as the target database:

```
RMAN> CONNECT TARGET sys/oracle_4U@boston
```

1. Restore the lost data file from the physical standby database over the network:

```
RMAN> RESTORE TABLESPACE USERS FROM SERVICE 'london';
```

1. Recover the restored tablespace:

```
RMAN> RECOVER TABLESPACE USERS;
```

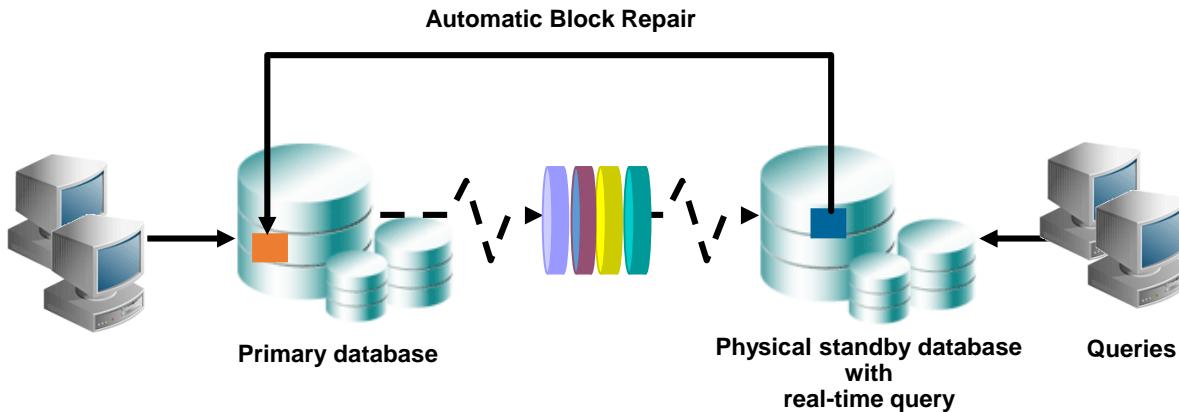
The same procedure is used to restore and recover the physical standby data files from the primary database.

# Automated Standby Synchronization from Primary Database

- A standby database might lag behind the primary for various reasons like:
  - Unavailability or insufficient network bandwidth between primary and standby database
  - Unavailability of standby database
  - Corruption/accidental deletion of archive redo data on primary
- Manually restore the primary controlfile on standby after use of the RECOVER FROM SERVICE command.
- The RECOVER FROM SERVICE command automatically rolls a standby forward:
  1. Remember all data file names on the standby.
  2. Restart standby in nomount.
  3. Restore controlfile from primary.
  4. Mount standby database.
  5. Rename data files from stored standby names.
  6. Restore new data files to new names.
  7. Recover standby.

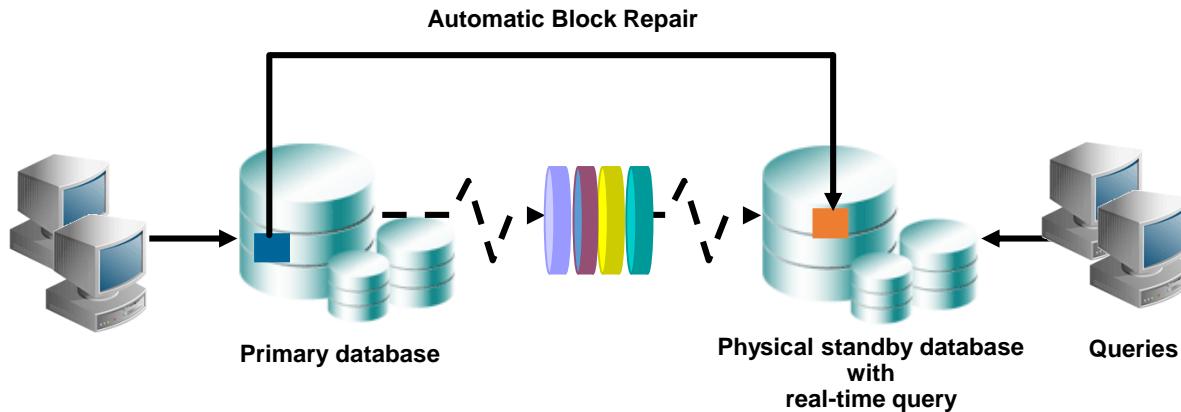
# Enhancements to Block Media Recovery

- Corrupted blocks in the primary database are automatically repaired by using blocks from a physical standby database.
- Real-time query and Active Data Guard must be enabled on the physical standby database.



# Enhancements to Block Media Recovery

- Corrupted blocks in the physical standby database are automatically repaired by using blocks from the primary database
- Real-time query and Active Data Guard must be enabled on the physical standby database



# Executing the RECOVER BLOCK Command

- Block media recovery can also be performed manually by using the RECOVER BLOCK command.

```
RECOVER DATAFILE 6 BLOCK 3;          Recover a single block

RECOVER
blocks
DATAFILE 2 BLOCK 43,79              Recover multiple
                                     in multiple data files
DATAFILE 6 BLOCK 183;
```

1. Physical standby database
2. Flashback logs
3. Blocks in full or level 0 incremental backup

# Excluding the Standby Database

- By default, block media recovery searches the physical standby database first for blocks to use.
- Exclude the standby database from the search by including the EXCLUDE STANDBY clause.

```
RECOVER BLOCK ... EXCLUDE STANDBY
```

# Quiz

- You must register the standby database to the RMAN recovery catalog in order to exchange backup files with the primary database.
  - a. True
  - b. False

# Quiz

- If a logical standby database has applied all data from the primary database, the data files can be exchanged with that of the primary database.
  - a. True
  - b. False

# Summary

- In this lesson, you should have learned how to:
  - Use RMAN to back up and restore files in a Data Guard configuration
  - Offload backups to a physical standby database
  - Recover your primary database by using a file from the physical standby database
  - Recover your primary database over the network
  - Synchronize Standby Database from Primary Database with one command

# Practice 17: Overview

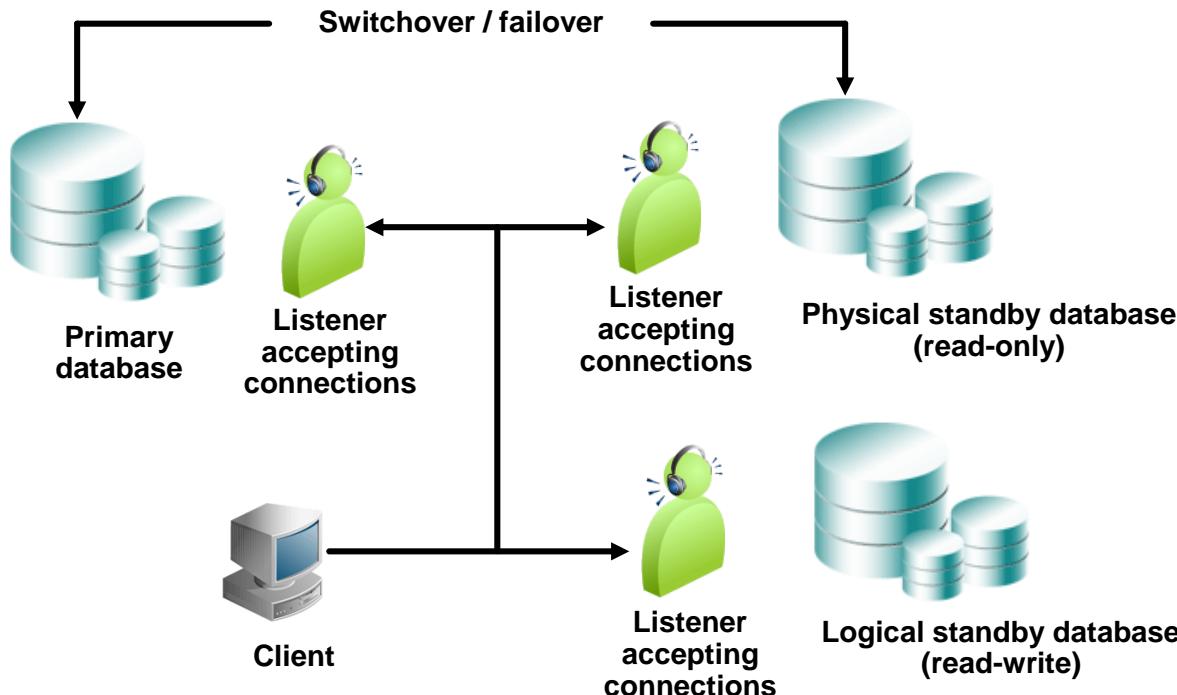
- This practice covers the following topics:
  - Enabling Change Tracking on the Physical Standby Database
  - Creating a Recovery Manager Catalog
  - Registering Your Database in the Recovery Catalog
  - Configuring RMAN Parameters
  - Recovering a Data File on Your Primary Database Over the Network
  - Rolling Forward a Standby Database with One Command

# **18. Enhanced Client Connectivity in a Data Guard Environment**

# Objectives

- After completing this lesson, you should be able to:
  - Configure client connectivity in a Data Guard configuration
  - Explain Transaction Guard and Application Continuity
  - Implement failover procedures to automatically redirect clients to a new primary database

# Connecting to the Appropriate Environment



# Understanding Client Connectivity in a Data Guard Configuration

- Be aware of the following issues when you manage client connectivity in a Data Guard configuration:
  - Databases reside on different hosts in a Data Guard configuration.
  - Clients must connect to the correct database that provides a specific business role such as primary, logical standby, snapshot standby, or physical standby.
  - Primary and standby databases can alternate hosts with switchover and failover operations.
  - If clients send connection requests to the wrong host, they may be connected to the wrong database or receive an error.
  - Clients must automatically reconnect to the correct database in the event of a failover.

# Preventing Clients from Connecting to the Wrong Database

- Use database services to prevent clients from connecting to the wrong database in the Data Guard configuration.
- Database services act as an abstraction layer between the client and database instances.
- Database services register with listeners.
- 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.

# Managing Services

- Database services can be managed by using the DBMS\_SERVICE package when Oracle Clusterware or Oracle Restart is not used.
- Database services attributes:
  - Service Name: Name of the service for administration
  - Network Name: Network name of the service as used in SQL\*Net connect descriptors for client connections
  - Parameter Array: Associative array with name-value pairs used to define:
    - Transparent Application Failover (TAF) attributes
    - Workload management goal directives for the service
    - Fast Application Notification (FAN)
    - Distributed Transaction Processing (DTP or XA)

# Creating Services for the Data Guard Configuration Databases

```
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');
```

# Connecting Clients to the Correct Database

- Use a database event trigger to ensure that clients connect to a database in the Data Guard configuration that is in the correct state and role.
- If no database is in the correct state and role, the trigger ensures that clients do not connect to a database.
- Use the trigger to start database services that have already been created.
  - DG\_PROD: Primary database
  - DG\_RTQ: Physical standby database opened in READ ONLY mode (real-time query)
  - DG\_SNAP: Physical standby database converted to a snapshot standby database
  - DG\_LSBY: Logical standby database

# Creating the AFTER\_DB\_ROLE\_CHANGE Trigger for Role-Based Services

```
CREATE TRIGGER MANAGE_SERVICES AFTER DB_ROLE_CHANGE 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;
/
```

# Configuring Service Names in the tnsnames.ora File

```
PROD = (DESCRIPTION = (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL = TCP) (HOST = host01) (PORT = 1521))
    (ADDRESS=(PROTOCOL = TCP) (HOST = host03) (PORT = 1521)))
    (CONNECT_DATA = (SERVICE_NAME = DG_PROD)))

RTQ = (DESCRIPTION = (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL = TCP) (HOST = host01) (PORT = 1521))
    (ADDRESS=(PROTOCOL = TCP) (HOST = host03) (PORT = 1521)))
    (CONNECT_DATA = (SERVICE_NAME = DG_RTQ)))

SNAP = (DESCRIPTION = (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL = TCP) (HOST = host01) (PORT = 1521))
    (ADDRESS=(PROTOCOL = TCP) (HOST = host03) (PORT = 1521)))
    (CONNECT_DATA = (SERVICE_NAME = DG_SNAP)))

LSBY = (DESCRIPTION = (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL = TCP) (HOST = host01) (PORT = 1521))
    (ADDRESS=(PROTOCOL = TCP) (HOST = host03) (PORT = 1521)))
    (CONNECT_DATA = (SERVICE_NAME = DG_LSBY)))
```

# Configuring Role-Based Services Using Oracle Clusterware

- Use SRVCTL to configure Oracle Clusterware–managed services on each database in the Data Guard configuration.
- Role changes managed by the Data Guard broker 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.
- Services can be started manually.

```
srvctl add service -db <db_unique_name> -service <service_name> -role  
"[PRIMARY] [,PHYSICAL_STANDBY] [,LOGICAL_STANDBY] [,SNAPSHOT_STANDBY]"  
[-policy {AUTOMATIC | MANUAL}]
```

# Adding Standby Databases to Oracle Restart Configuration

- Standby databases created with Enterprise Manager are not automatically added to the Oracle Restart configuration.
- Use SRVCTL to add the standby databases.

```
srvctl add database -d <db_unique_name> -o <oracle_home>
[-m <domain_name>] [-p <spfile>] [-r {PRIMARY | PHYSICAL_STANDBY |
LOGICAL_STANDBY | SNAPSHOT_STANDBY }] [-s {start_options}] [-t {stop_options}]
[-n <db_name>] [-y {AUTOMATIC | MANUAL}] [-a "<diskgroup_list>"]
```

- Example:

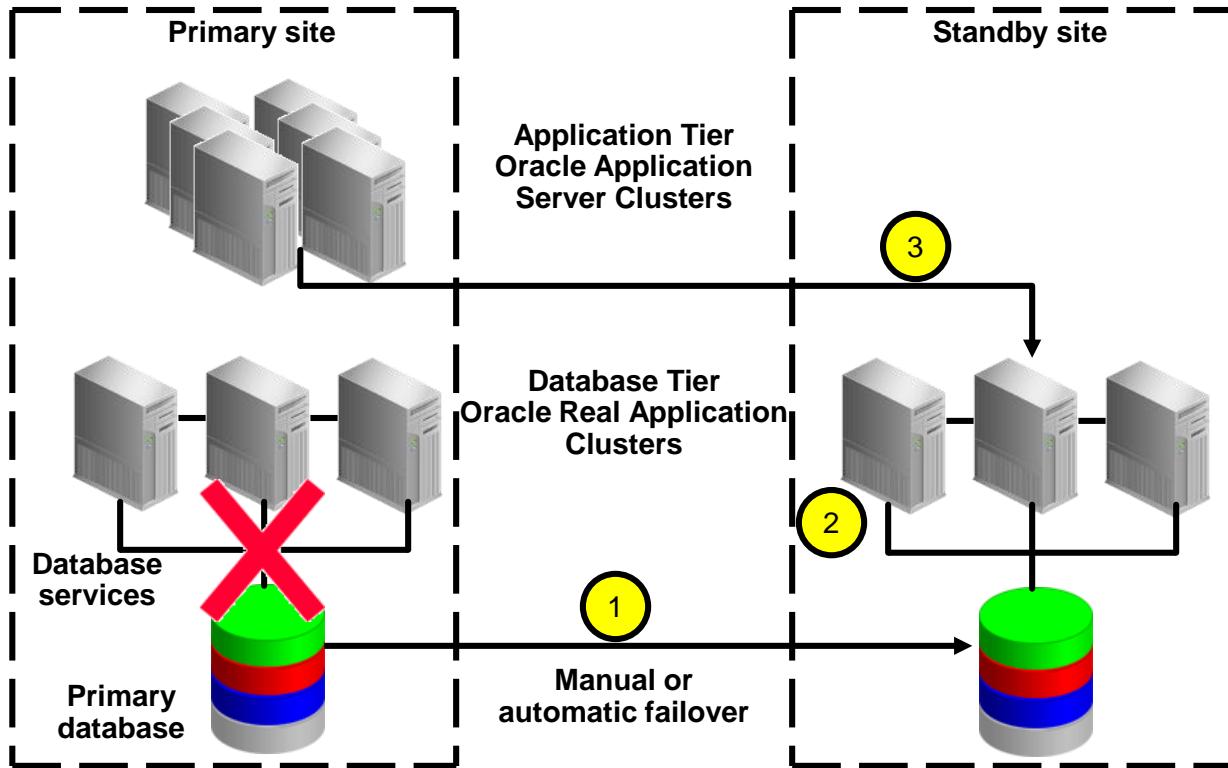
```
srvctl add database -d london -o
/u01/app/oracle/product/19.3.0/dbhome_1 -m example.com -p
/u01/app/oracle/product/19.3.0/dbhome_1/dbs/spfilelondon.ora
-r PHYSICAL_STANDBY -n boston -a "SBDAT,SBFRA"
```

# Example: Configuring Role-Based Services

- PAYROLL: Read-write service that always runs on the database with the primary role
- ORDERSTATUS: Read-only service that always runs on an Active Data Guard standby database

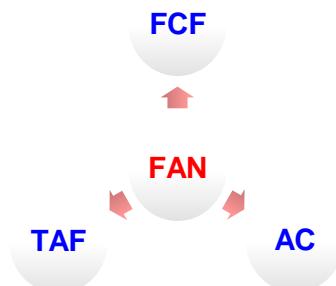
```
srvctl add service -d boston -s payroll -l PRIMARY  
                  -m BASIC -e SELECT -w 1 -z 180  
  
srvctl add service -d london -s payroll -l PRIMARY  
                  -m BASIC -e SELECT -w 1 -z 180  
  
srvctl add service -d boston -s orderstatus  
                  -l PHYSICAL_STANDBY  
  
srvctl add service -d london -s orderstatus  
                  -l PHYSICAL_STANDBY
```

# Automatic Failover of Applications to a New Primary Database



# Client Failover: Components

- The following features are used to implement client failover and minimize the impact of planned and unplanned outages:
  - Connect time failover
  - DB\_ROLE\_CHANGE system event
  - Fast Application Notification (FAN)
  - Transparent Application Failover (TAF)
  - Fast Connection Failover (FCF)
  - Transaction Guard
  - Application Continuity



# Data Guard Broker and Fast Application Notification (FAN)

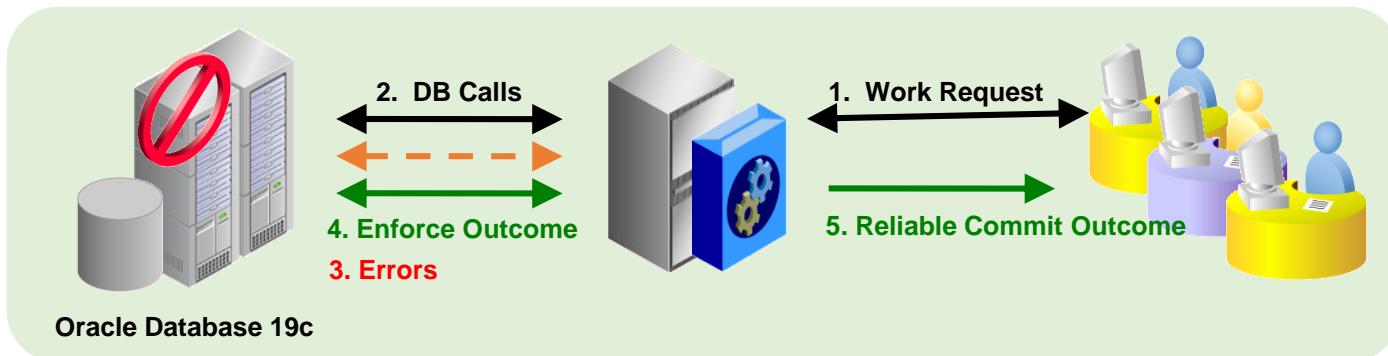
- The Data Guard broker publishes FAN events at failover time.
- Applications respond to FAN events without programmatic changes if using Oracle-integrated database clients:
  - Oracle Database JDBC
  - Oracle Database Oracle Call Interface (OCI)
  - Oracle Database ODP.NET
- Clients that receive FAN events can be configured for Fast Connection Failover (FCF) 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.

# Automating Client Failover by Using Transaction Guard and Application Continuity

- Oracle Database 19c introduced 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

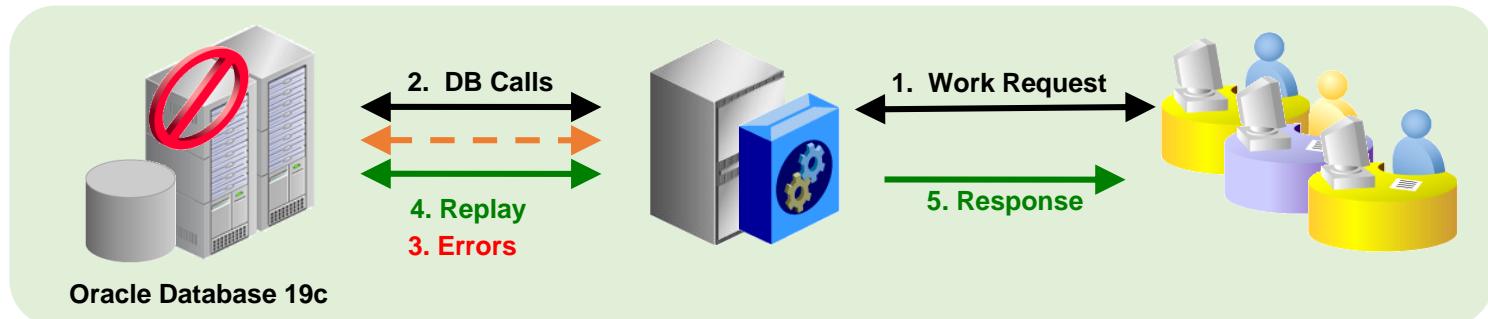
# What Is Transaction Guard?

- Transaction Guard:
  - Is a tool that provides a reliable commit outcome for the last transaction after errors
  - Is an API available for JDBC Thin, C/C++ (OCI/OCCI), and ODP.NET
  - Is used by Application Continuity for at-most-once execution
  - Can be used independently of Application Continuity
- Without Transaction Guard, retry can cause logical corruption.



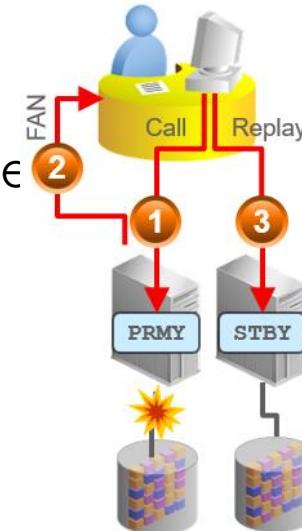
# What Is Application Continuity?

- Replays in-flight work on recoverable errors
- Masks many hardware, software, network, storage errors, and outages, when successful
- Improves the end-user experience



# 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
- Supported clients:
  - JDBC Thin driver, OCI drivers, Universal Connection Pool, WebLogic Server, ODP.NET, OCI Session Pool, and Tuxedo



# 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 driver, OCI drivers, Universal Connection Pool, WebLogic Server, ODP.NET, OCI Session Pool, and more

# Configure Services on Single Instance Databases to Use Application Continuity

- To configure services in a single-instance database, use the DBMS\_SERVICE package:

```
DECLARE
  params dbms_service.svc_parameter_array;
BEGIN
  params('FAILOVER_TYPE') := 'TRANSACTION';
  params('REPLAY_INITIATION_TIMEOUT') := 1800;
  params('RETENTION_TIMEOUT') := 86400;
  params('FAILOVER_DELAY') := 10;
  params('FAILOVER_RETRIES') := 30;
  params('FAILOVER_RESTORE') := 'LEVEL1';
  params('commit_outcome') := 'true';
  params('aq_ha_notifications') := 'true';
  dbms_service.modify_service('prmy',params);
END;
/
```

**Mandatory Settings for Application Continuity**

**Mandatory Setting for Transaction Guard**

# Configuring Client: JDBC-Thin Driver

1. Ensure all recommended patches are applied at the client. Refer to the MOS note Client Validation Matrix for Application Continuity (Doc ID 2511448.1)
2. Configure the Oracle JDBC Replay Data Source in the property file or on console:
3. Use JDBC Statement Cache for Coverage and Performance
4. Tune the Garbage Collector
5. JDBC Concrete Classes
6. Configure Fast Connection Failover (FCF)

# Configuring Client: OCI Driver

1. Ensure all recommended patches are applied at the client. Refer to the MOS Note Client Validation Matrix for Application Continuity (Doc ID 251148.1).
2. Replace `OCISTmtPrepare` with `OCISTmtPrepare2`.  
`OCISTmtPrepare()` has been deprecated since 12.2. All applications should use `OCISTmtPrepare2()`. TAC and AC allow `OCISTmtPrepare` but do not replay this statement.
3. To use FAN for OCI-based applications, do the following:
  - Use `aq_ha_notifications` on the services
  - Use the recommended Connection String for auto-ons
  - Set `auto_config`, `events`, and `wallet_location` (optional) in `oraaccess.xml`
  - Link the application with the O/S client thread library
  - Open port 6200 for ONS (6200 is the default port, a different port may have been chosen)

# Configuring Client: ODP.NET Unmanaged Provider Driver

1. Ensure all recommended patches are applied at the client. Refer to the MOS Note Client Validation Matrix for Application Continuity (Doc ID 251148.1).
2. To use FAN for OCI-based applications, do the following:
  - Use `aq_ha_notifications` on the services
  - Use Recommended Connection String for auto-ons
  - Set `onsConfig` and `wallet_location` (optional) in `oraaccess.xml`
  - Open port 6200 for ONS (6200 is the default port, a different port may have been chosen)
  - Set FAN, in the connection string
  - (optional) Set Runtime Load Balancing, also in the connection string

# Quiz

- In a Real Application Cluster (RAC) environment, which method should you generally use to manage role-based services?
  - a.DBMS\_SERVICE
  - b.SRVCTL

# Quiz

- FAN events are not enabled or started during the installation of Grid Infrastructure for Standalone Servers for Data Guard.
  - a.True
  - b.False

# Quiz

- When does Application Continuity work in a Data Guard environment?
  - a.Switchover
  - b.Manual Failover
  - c.Fast-Start Failover with Maximum Protection
  - d.Fast-Start Failover with Maximum Availability
  - e.Fast-Start Failover with Maximum Performance

# Summary

- In this lesson, you should have learned how to:
  - Configure client connectivity in a Data Guard configuration
  - Explain Transaction Guard and Application Continuity
  - Implement failover procedures to automatically redirect clients to a new primary database

# Practice 18: Overview

- This practice covers the following topics:
  - Creating and Testing Primary Database Services
  - Modifying the Primary Database Service to Use Application Continuity

# 19. Patching and Upgrading Databases in a Data Guard Configuration

# Objectives

- After completing this lesson, you should be able to patch and upgrade databases in your Data Guard configuration:
  - By using Standby-First Patch Apply technique
  - By using traditional upgrade methods
  - By performing rolling upgrades
  - By using the DBMS\_ROLLING package with Active Data Guard

# Data Guard Standby-First Patch Apply

- Provides support for different software releases between a primary database and physical standby database for the purpose of 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 one year 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

# Data Guard Standby-First Patch Apply

- Steps to apply the patch to the standby database first:
  1. Shut down all standby instances on the standby database (if patch is not RAC Rolling).
  2. On the standby site, apply the patch as described in the patch README.
  3. Restart the standby instances.
  4. Restart media recovery on the physical standby database.
  5. Evaluate the patch on the standby by monitoring alert and log files, using snapshot standby, or Active Data Guard.
  6. Apply the patch to the primary database site.
    - Standby can be read-only if the patch is RAC Rolling capable.
    - Standby must be mounted if the patch is not RAC Rolling capable.
    - Alternatively, switch over to standby and apply the patch to the former primary.

# Upgrading an Oracle Data Guard Broker Configuration

- If you are using the Data Guard broker, you must perform the following steps before upgrading your databases:
1. Disable broker management of the configuration.

```
DGMGRL> DISABLE CONFIGURATION;
```

1. Stop the Data Guard broker.

```
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE;
```

- After completing the upgrade:
1. Start the Data Guard broker.

```
SQL> ALTER SYSTEM SET DG_BROKER_START=TRUE;
```

1. Enable broker management of the configuration by connecting to the primary database.

```
DGMGRL> ENABLE CONFIGURATION;
```

# Upgrading Oracle Database in a Data Guard Configuration with a Physical Standby Database

- To perform a traditional upgrade:
  1. Perform the preparation steps for an upgrade.
  2. Install the Oracle Database software on physical standby database systems and the primary database system.
  3. Shut down the primary database.
  4. Shut down the physical standby database(s).
  5. Stop all listeners, agents, and other processes running in the Oracle homes that are to be upgraded.
  6. If Oracle Automatic Storage Management (ASM) is in use, it should be upgraded before the databases.

# Upgrading Oracle Database in a Data Guard Configuration with a Physical Standby Database

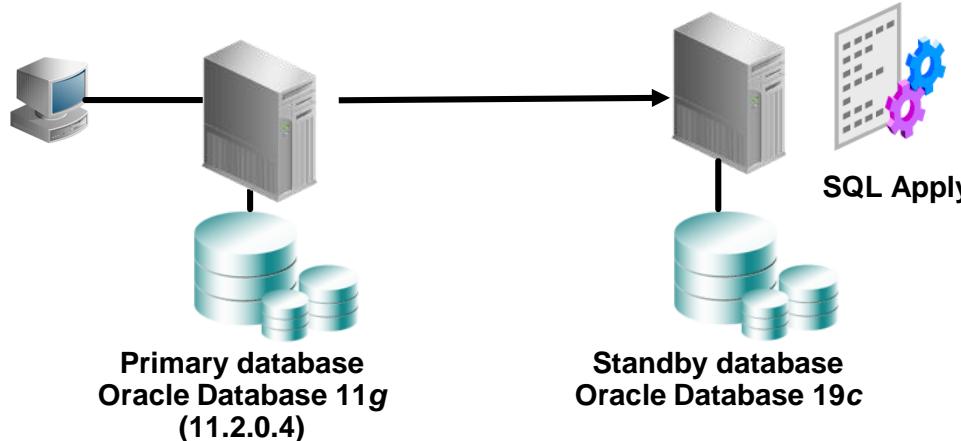
7. In the new Oracle home, restart all listeners, agents, and other processes stopped in step 5.
8. Mount the physical standby database(s) on the new Oracle home.
9. Start Redo Apply on the physical standby database(s).
10. Upgrade the primary database as described in the *Oracle Database Upgrade Guide*. Redo Transport will propagate this to the standby database(s).
11. Open the upgraded primary database.
12. If Active Data Guard was being used prior to the upgrade, it will need to be re-enabled.
13. Optionally modify the COMPATIBLE initialization parameter.

# Upgrading Oracle Database in a Data Guard Configuration with a Logical Standby Database

- To perform a traditional upgrade:
  1. Perform the preparation steps for an upgrade.
  2. Set the data protection mode to MAXIMUM PERFORMANCE.
  3. Stop all user activity on the primary database and defer the logical standby database remote archival destination.
  4. Stop SQL Apply on the standby database.
  5. Install Oracle Database software on the primary database system and upgrade the primary database.
  6. Install Oracle Database software on the logical standby database system and upgrade the standby database.
  7. Restart SQL Apply on the standby database.
  8. Open the upgraded primary database.
  9. Reset the data protection mode, if necessary.

# Using SQL Apply to Upgrade the Oracle Database

- Use a logical standby database to perform a rolling upgrade of Oracle Database 11g software from patch set release *n* to any higher-versioned patch set or major version release.
- Incur minimal down time by using different releases of Oracle Database on the primary database and logical standby database while upgrading.



# Requirements for Using SQL Apply to Perform a Rolling Upgrade

- The broker configuration should be disabled before a rolling upgrade.
- Data Guard protection mode must be maximum availability or maximum performance.
- The `LOG_ARCHIVE_DEST_n` initialization parameter for the logical standby destination must *not* be set to `MANDATORY`.
- The `COMPATIBLE` initialization parameter value must match the software release prior to the upgrade.

# Performing a Rolling Upgrade by Using SQL Apply

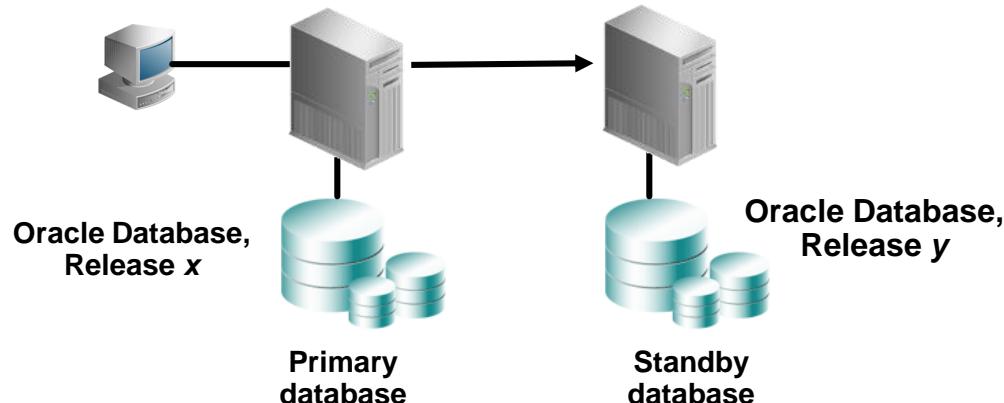
- **Case 1:** Performing a Rolling Upgrade With an Existing Logical Standby Database.
- **Case 2:** Performing a Rolling Upgrade By Creating a New Logical Standby Database.
- **Case 3:** Performing a Rolling Upgrade With an Existing Physical Standby Database

# Identifying Unsupported Data Types

- Identify data types and storage attributes on the primary database that are not supported in a logical standby database:
  - If you are performing a rolling upgrade using the `DBMS_ROLLING` PL/SQL package, then query the `DBA_ROLLING_UNSUPPORTED` view.
  - If you are not using the `DBMS_ROLLING` package, but are instead following the manual process, then query the `DBA_LOGSTDBY_UNSUPPORTED` view.
  - Determine how to handle unsupported objects during the rolling upgrade.

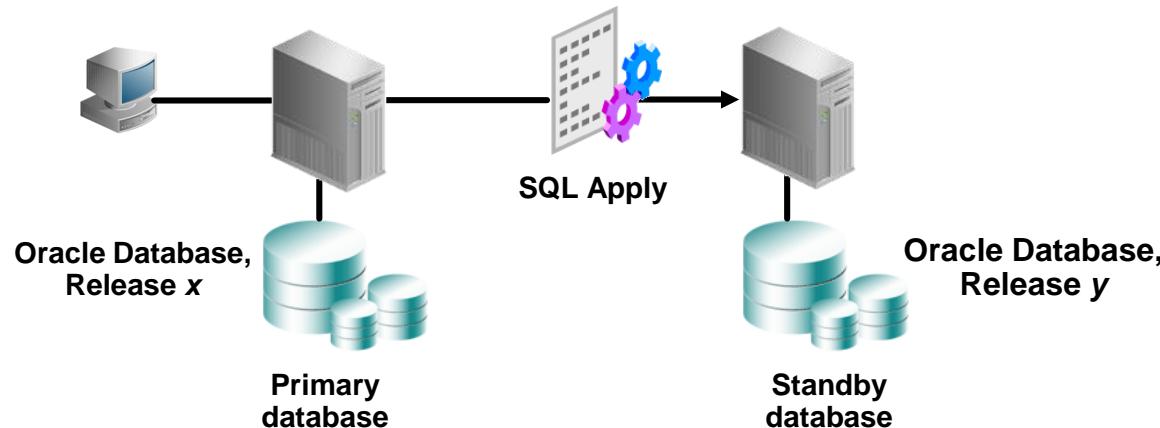
# Case 1: Performing a Rolling Upgrade by Using an Existing Logical Standby Database

1. Prepare for the rolling upgrade by stopping SQL Apply and setting the COMPATIBLE initialization parameter.
2. Upgrade the Oracle Database software on the logical standby database and upgrade the logical standby database.



# Performing a Rolling Upgrade by Using an Existing Logical Standby Database

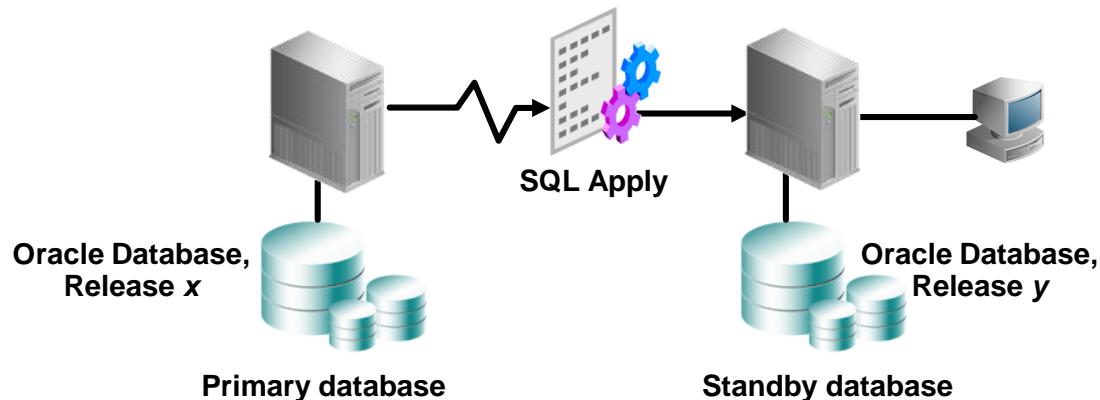
3. Restart SQL Apply on the upgraded logical standby database.



# Performing a Rolling Upgrade by Using an Existing Logical Standby Database

4. Query DBA\_LOGSTDBY\_EVENTS to monitor the upgraded logical standby database.
5. Begin a switchover to the upgraded logical standby database:

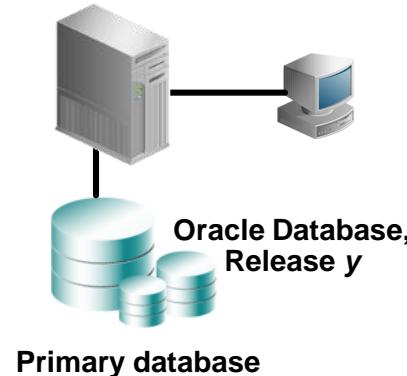
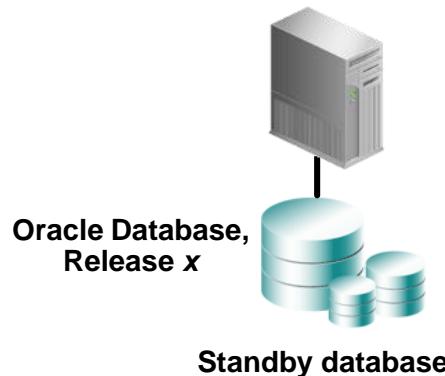
```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;
```



# Performing a Rolling Upgrade by Using an Existing Logical Standby Database

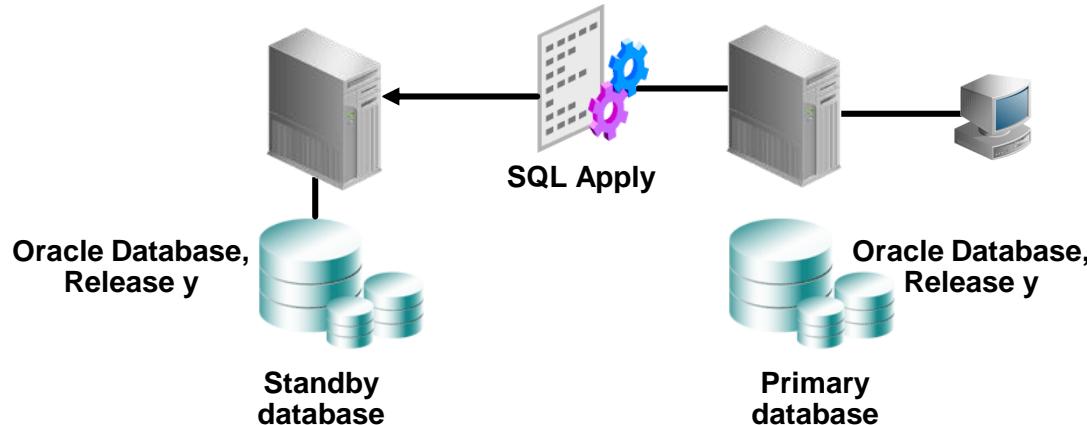
6. Import any tables that were unsupported and modified.
7. Complete the switchover and activate user applications:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL PRIMARY;
```



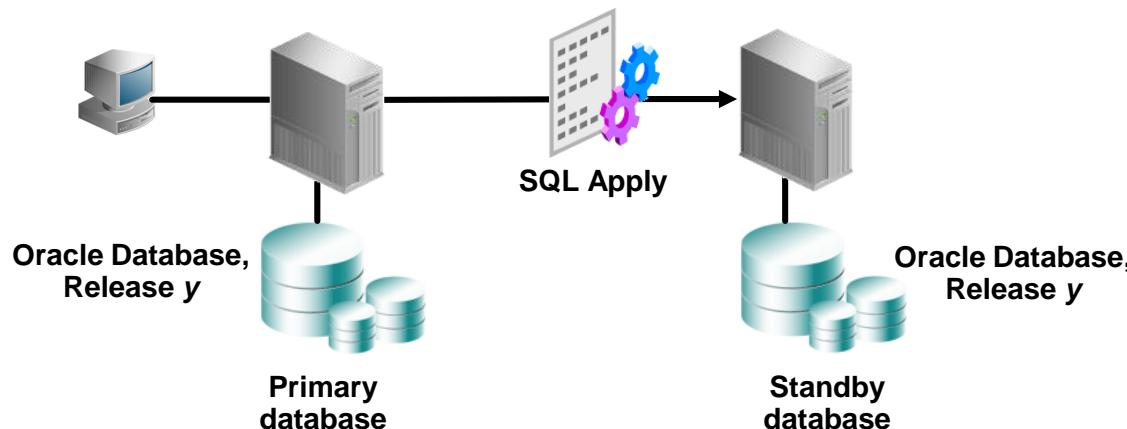
# Performing a Rolling Upgrade by Using an Existing Logical Standby Database

8. Upgrade the original primary database.
9. Start SQL Apply on the original primary database.



# Performing a Rolling Upgrade by Using an Existing Logical Standby Database

10. Monitor events on the new logical standby database.
11. Optional: Perform a switchover to return to the original configuration.
12. Optional: Raise the compatibility level on both databases.

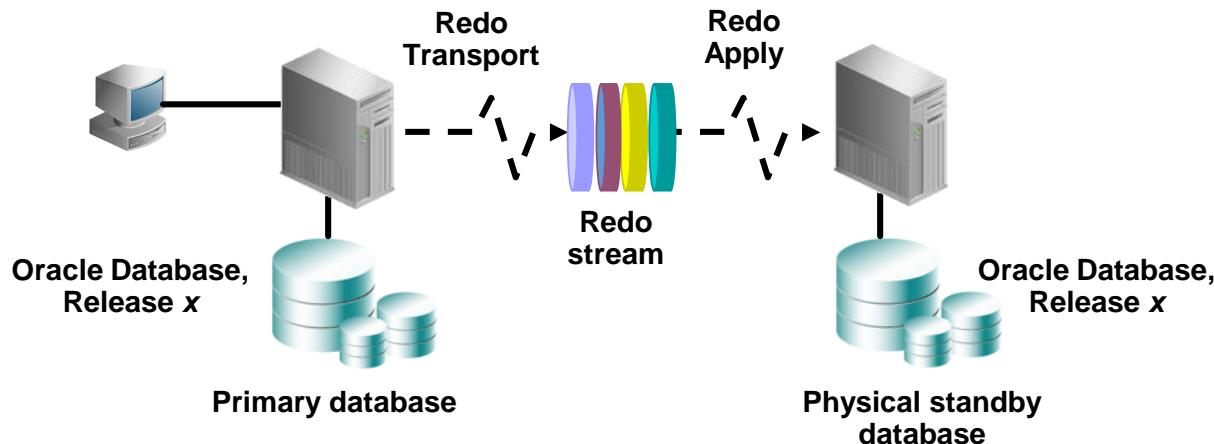


# Case 2: Performing a Rolling Upgrade by Creating a New Logical Standby Database

1. Identify data types and storage attributes on the primary database that are not supported in a logical standby database.
2. Create a logical standby database.
3. Perform a rolling upgrade.

# Case 3: Performing a Rolling Upgrade by Using a Physical Standby Database

1. Prepare the primary database for a rolling upgrade.
  - a. Enable Flashback Database.
  - b. Create a guaranteed restore point.

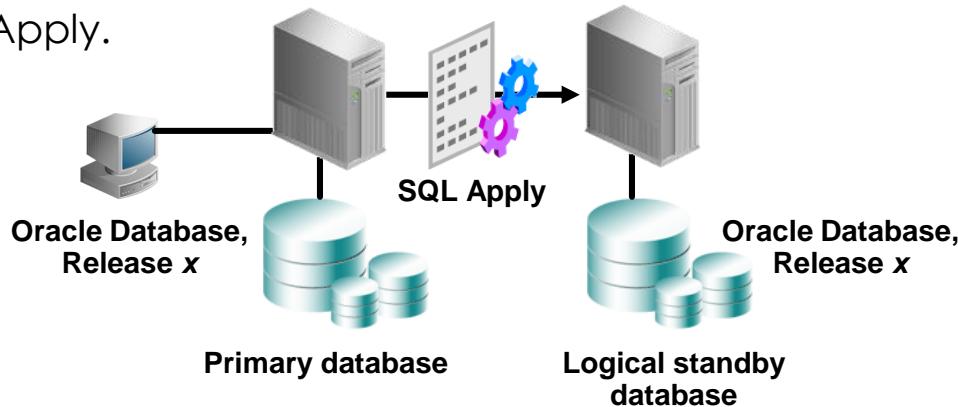


# Performing a Rolling Upgrade by Using a Physical Standby Database

2. Convert the physical standby database to a logical standby database.
  - a. Create a logical standby database and execute:

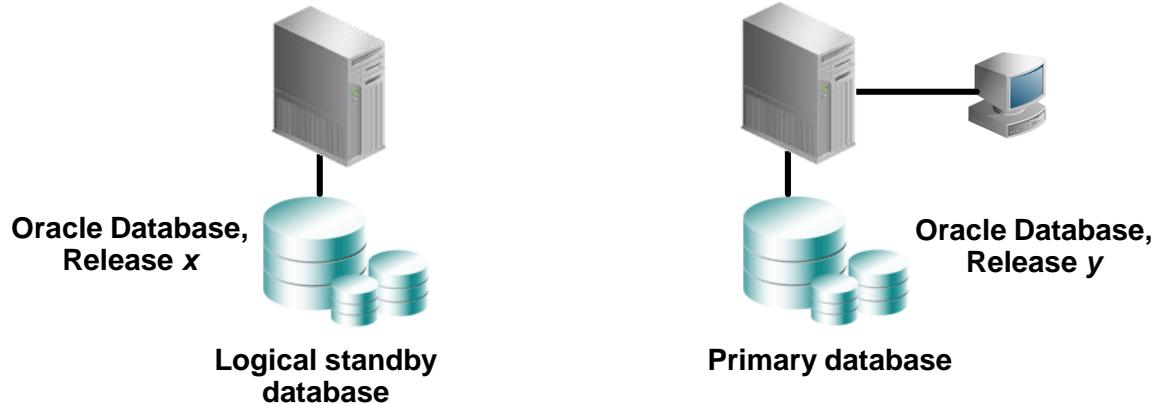
```
SQL> ALTER DATABASE RECOVER TO LOGICAL STANDBY KEEP IDENTITY;
```

- a. Disable automatic deletion of foreign archived logs at the logical standby database.
  - b. Start SQL Apply.



# Performing a Rolling Upgrade by Using a Physical Standby Database

3. Upgrade the logical standby database (steps 1–7 of “Performing a Rolling Upgrade by Using a Logical Standby Database”).



# Performing a Rolling Upgrade by Using a Physical Standby Database

4. 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;  
SQL> SHUTDOWN IMMEDIATE
```

4. Mount the original primary database using the new version of the software:

```
SQL> startup mount
```

# Performing a Rolling Upgrade by Using a Physical Standby Database

6. Convert the original primary database to a physical standby:

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;  
SQL> SHUTDOWN IMMEDIATE;
```

6. Start managed recovery on the original primary database.

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

# Performing a Rolling Upgrade by Using a Physical Standby Database

8. Perform a switchover to return your original primary database to the primary database role. (Optional)
9. Clean up the guaranteed restore point created in step 1.

# Rolling Upgrades Using DBMS\_ ROLLING and Active Data Guard

- Database software upgrades using the DBMS\_ ROLLING PL/SQL package will be usable starting with the first patchset of Oracle Database 19c to the second patchset.
- Database upgrades from Oracle Database 11g to Oracle Database 19c will need to use the classic Transient Logical Standby upgrade procedure.
- The DBMS\_ ROLLING PL/SQL package can be used for other database maintenance tasks with the first release of Oracle Database 19c.
- As of Oracle Database 19c Release 2 (12.2.0.1), Data Guard broker can remain on during a DBMS\_ ROLLING rolling upgrade; there is no longer any need to disable it.

# DBMS\_ROLLING: Concepts

- Rolling changes can be applied on the whole Data Guard configuration.
  - Three stages: Specification, Compilation, and Execution
  - Execution has three phases: Start, Switchover, and Finish
- Two key groups:
  - Leading group (LG)
    - These databases are upgraded first (before switchover).
    - The LG has a master database (the future primary database).
  - Trailing group (TG)
    - These databases are upgraded last (after switchover).
    - The TG has a master database (the original primary database).

# DBMS\_ROLLING: Concepts

- Leading group
  - The Leading Group Master database (LGM) must be identified during Specification.
  - The LGM starts as a physical standby, converted into a logical standby (START), and then becomes the primary database (SWITCHOVER).
  - Other databases in the leading group protect the LGM.
  - LGM responsibility is transferrable on failure.
- Trailing group
  - The TG contains the original primary database (Trailing Group Master [TGM]).
  - Other databases in the Trailing Group protect the TGM.
  - TGM responsibility is transferrable on failure.

# DBMS\_ROLLING: Overview

- **Specification:** You first specify how you want to implement the rolling upgrade process. You use the following procedures during the specification phase:
  - DBMS\_ROLLING.INIT\_PLAN
  - DBMS\_ROLLING.SET\_PARAMETER
- **Compilation:** This stage is to assemble a rolling upgrade plan.
  - DBMS\_ROLLING.BUILD\_PLAN
- **Execution:** Execution of the rolling upgrade has five stages.
  - Stage 1: DBMS\_ROLLING.START\_PLAN
  - Stage 2: Upgrade Leading Group
  - Stage 3: DBMS\_ROLLING.SWITCHOVER
  - Stage 4: Upgrade Training Group
  - Stage 5: DBMS\_ROLLING.FINISH\_PLAN

# Database Rolling Upgrade: Specification and Compilation Stages

- Generate an *upgrade plan*:
  - Specify parameters of the rolling upgrade, such as target software versions, participating databases, apply lag requirements, and logging levels.
    - DBMS\_ROLLING.INIT\_PLAN procedure
    - DBMS\_ROLLING.SET\_PARAMETER procedure
    - DBA\_ROLLING\_PARAMETERS view
  - Use the DBMS\_ROLLING.BUILD\_PLAN procedure to generate an upgrade plan and perform validations.
  - Use the DBA\_ROLLING\_PARAMETERS, DBA\_ROLLING\_PLAN, and DBA\_ROLLING\_EVENTS views to display the current plan and diagnose any problems with the plan.

# Example: Specification Stage

- Initialize the upgrade parameters:

```
SQL> exec DBMS_ROLLING.INIT_PLAN(future_primary=>'london');
```

- View the current upgrade parameter values:

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

SCOPE	NAME	CURVAL
boston	INVOLVEMENT	FULL
	SWITCH_LGM_LAG_WAIT	60
...		

- Configuring the plan to wait for the apply lag to fall below 60 seconds before switching over to the future primary:

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

# Example: Compilation Stage

- Build the upgrade plan:

```
SQL> exec DBMS_ROLLING.BUILD_PLAN;
```

- View the current upgrade plan:

```
SQL> select instid, target, phase, description from dba_rolling_plan;
```

INSTID	TARGET	PHASE	DESCRIPTION
-----			
1	boston	START	Verify database is a primary
2	boston	START	Verify MAXIMUM PROTECTION is disabled
3	london	START	Verify database is a physical standby
4	london	START	Verify physical standby is mounted
5	boston	START	Verify server parameter file exists and is modifiable
...			

# Database Rolling Upgrade: Execution Stage

1. Call DBMS\_ROLLING.START\_PLAN to configure the primary and standby databases participating in the upgrade.
  2. Upgrade the RDBMS software for leading group databases.
  3. Call DBMS\_ROLLING.SWITCHOVER to swap roles between the current primary database and the new primary database. Switchover is the only required down time.
  4. Restart the former primary and any bystander standby databases by using new binaries.
  5. Call DBMS\_ROLLING.FINISH\_PLAN to complete the upgrade of the former primary and any bystanders and resynchronize with the new primary.
- No arguments are needed on these three procedures.

# Rolling Upgrade Support for Multitenant Databases

- The steps to perform a rolling upgrade by using DBMS\_ROLLING on a CDB are no different from non-CDB environments.
- There are, however, a few additional requirements that must be met:
  - The TNS services in the LOG\_ARCHIVE\_DEST\_n parameters must resolve to the root container of the destination database.
  - All CDBs on the transient logical standby must be plugged in and opened before calling DBMS\_ROLLING.SWITCHOVER.
- Installing, upgrading, or patching of applications is not supported while a DBMS\_ROLLING upgrade is in progress.

# Quiz

- You can perform a rolling upgrade using a logical standby SQL apply technique with zero down time.
  - a.True
  - b.False

# Quiz

- During a DBMS\_ROLLING upgrade, the Data Guard broker can be enabled and running throughout the upgrade procedures.
  - a.True
  - b.False

# Summary

- In this lesson, you should have learned how to patch and upgrade databases in your Data Guard configuration:
  - By using Standby-First Patch Apply methods
  - By using traditional upgrade methods
  - By performing rolling upgrades with `DBMS_ROLLING`

# **Thank you**