Oracle Data Guard from A to Z
Zohar Elkayam, Service Director: Database Solutions
Agenda

- HA, DR and BCP: basic concepts
- Overview of the Oracle Data Guard solution
- Data Guard Physical Standby and a demo
- Data Guard Broker and a short demo
- Data Guard Logical Standby
- Data Protection modes
- Monitoring Data Guard and optimizing Data Guard
- Active Data Guard
- Snapshot Standby
מי אני?

זוהר אלקיים, ראש תחום בסיסי נתונים ויועץ אורקל בכיר ב-GlassHouse Technologies

• אוורקל בטכניות ואפליקטיביות منذ 1998 (משתתף ב-Applikity)

• העורך הטכני של הספר "יוכלות Oracle SQL" הערוך הטכני של הספר "יוכלות, מתcondeות שחפרו" המדריך לשולף המהיר

• בלוגר: zoharelkayam.wordpress.com

בולים: GlassHouse Technologies
About GlassHouse Technologies

• We provides IT Infrastructure services & solutions in most fields: storage, OS, virtualization, backup and databases
• Worldwide operations: US, UK, Swiss, Israel, Turkey and Australia
• Over 200 employees in Israel alone and ~1000 worldwide.
• Formally known as Integrity systems, in the databases business since 1995
Definitions: HA and DR

- **High Availability (HA):** The degree to which an application, service, or functionality is available upon user demand. Availability is measured by the perception of an application's end user.

- **Disaster Recovery (DR):** The ability of IT to recover (after a disaster) certain stated applications to defined Recovery Service Objectives and make these applications and their data available to the business units.
Recovery Service Objectives (RSO)

- **RTO/MTTR**: Recovery Time Objective defines the length of time it will take to recover all the applications in case of a disaster. Typically expressed in seconds, minutes, hours, days.

- **RPO**: Recovery Point Objective defines the point in time to which data can be recovered for all the applications in case of a disaster. Effectively defines tolerance for data loss. Typically expressed in minutes, hours or days of data loss.

- **SLA**: Agreement between two parties which specifies level of availability, performance or operation.
Business Continuance Procedure (BCP)

• The ability to continue (after a disaster) certain defined components of the business for a defined time at a defined level of competence.

• Sometimes known as COOP Continuance of Operations.

• Consists of two components:
  – People and manual processes – Typically BC.
  – The IT Environment – Typically DR.
Keeping the Business Running

Business Continuity
Resumption of full operations combining People, Processes and Platforms

Disaster Recovery
Site-level crisis, data and IT operations resumption

Backup and Restore
Presumes infrastructure is whole
97% is file/small unit related

High Availability
Presumes that the rest of the environment is active
Causes of Data/Service Loss

- Hardware & system errors: 49%
- Human errors: 36%
- Computer viruses: 7%
- Software corruption: 4%
- Natural disasters: 3%

Source: Disaster Recovery Journal
Real Application Clusters
Continuous Availability for all Applications

Data Guard
Guaranteed Zero Data Loss

Flashback
Guaranteed Zero Data Loss

ASM Mirroring
Storage Failure Protection

Dynamic Reconfiguration
Capacity on Demand without Interruption

Online Redefinition
Adapt to Change Online

Unplanned Downtime
- System Failures
- Site Failures
- Human Error
- Storage/Net Failures

Planned Downtime
- System Maintenance
- Database Maintenance

OracLe
Overview of the Oracle Data Guard solution
What Is Oracle Data Guard?

Primary database

Redo transport

Standby database

Oracle Net

Database

Database copy
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
Types of Standby Databases

• Physical standby database
  – Identical to the primary database on a block-for-block basis
  – 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
  – 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
  - Fully updatable standby database
  - Created by converting a physical standby database
  - Local updates are discarded when a 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
Oracle Data Guard: Architecture
(Overview)

Primary database transactions

LGWR
Online redo logs
ARCO
Archived redo logs

Redo buffer
LNSn
RFS
MRP or LSP
Standby database

Oracle net

Gap resolution

(Real-time apply)

Standby redo logs
ARCO
Archived redo logs

Backup
Reports
Primary Database Processes

Primary database transactions

Redo buffer

LGWR

Online redo logs

ARCO

Archived redo logs

LNSn

Oracle net

Gap resolution

RFS

MRP or LSP

Standby database

Standby redo logs

ARCO

(Real-time apply)

Archived redo logs

Backup

Reports

Oracle net

Infrastructure Optimized

18
Standby Database Processes

Primary database transactions

- LGWR
- Online redo logs
- ARC0

Redo buffer

LNSn

Oracle net

- LGWR
- Online redo logs
- ARC0

Redo buffer

Gap resolution

- ARCO
- Archived redo logs

RFS

(Real-time apply)

MRP or LSP

Standby database

Standby redo logs

Reports

Backup

Archived redo logs

GlassHouse

Infrastructures::Optimized
Physical Standby Database: Redo Apply Architecture
Logical Standby Database: SQL Apply Architecture

- Primary database
- Production database
- Logical standby database
- SQL Apply
- Transform redo information into SQL
- Redo transport
- Reports
Automatic Gap Detection and Resolution

Primary database transactions

Online redo logs

ARC0

Archived redo logs

Redo buffer

LGWR

RFS

(Redto resolve gap)

(Real-time apply)

Standby redo logs

ARC0

Backup

Standby database

Reports

MRP or LSP

Archived redo logs

Oracle net
Data Protection Modes

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

– Maximum protection
– Maximum availability
– Maximum performance (default)
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.
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)

• See MOS Note 413484.1 for latest capabilities and restrictions
Data Guard Operational Requirements: Network

• Depends on Redo generation
• Find peak redo in AWR report

<table>
<thead>
<tr>
<th>Load Profile</th>
<th>Per Second</th>
<th>Per Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redo size:</td>
<td>51,944.64</td>
<td>5,177.09</td>
</tr>
</tbody>
</table>

• Bandwidth in MBPS =
  \[(\text{redo bytes per sec} / 8) / 1,000,000\]
Role Transitions: Switchover and Failover

• Oracle Data Guard supports two role-transition operations:
  – Switchover
    • Planned role reversal
    • Used for OS or hardware maintenance
  – 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
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
• Causes users to disconnect from primary
Switchover: Before

Application

Read/write transactions

Primary database

San Francisco
Boston

Oracle Net

Standby database

Read-only reports

Application
Switchover: After

Read-only reports

Standby database

Primary database

Oracle Net

San Francisco

Boston

Application

Read/write transactions

Application

Application
Performing a Switchover by Using DGMGRL

DGMGRL> SWITCHOVER TO 'pc00sby1';
Performing switchover NOW, please wait...
New primary database "pc00sby1" is opening...
Operation requires shutdown of instance "pc00prmy" on 
database "pc00prmy"
Shutting down instance "pc00prmy"...
ORA-01109: database not open
Database dismounted.
ORACLE instance shut down.
Operation requires startup of instance "pc00prmy" on 
database "pc00prmy"
Starting instance "pc00prmy"...
ORACLE instance started.
Database mounted.
Switchover succeeded, new primary is "pc00sby1"
Situations That Prevent a Switchover

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

Primary database → Online Redo logs → Archived redo logs

Standby database becomes primary database.

Read/write transactions

San Francisco

Local archiving

Online redo logs

Archived redo logs

Boston
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

• Failover should be used only in an emergency.
• Old primary database is no longer part of the configuration.
• Data loss is possible.
• When choosing a standby database to fail over to, you should:
  – Choose a physical standby database when possible (in case there are logical standby in the configuration).
  – Choose the standby database that is most current.
Performing a Failover (Using DGMGRL)

1. Execute the `FAILOVER` command to initiate the failover operation:

   ```
   DGMGRL> FAILOVER TO 'pc00sby1' [IMMEDIATE];
   ```

2. Reset the protection mode (if necessary).
3. Reinstate the primary database to serve as a standby database in the configuration.
4. Reinstate or re-create other disabled standby databases in the configuration.
Reenabling Disabled Databases by Using DGMGRL

- Disabled databases must be reinstated or re-created to re-enable broker management.

- Reinstall a database using **REINSTATE DATABASE**:

  ```
  DGMGRL> REINSTATE DATABASE pc00prmy;
  ```

- If you cannot reinstall 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 pc00prmy;
  ```
Choosing an Interface for Administering a Data Guard Configuration

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

• Non–Data Guard broker configuration:
  – SQL commands
Oracle Data Guard Broker Framework

Repository

Agent

Primary database

Oracle Management Server

Enterprise Manager

Agent

Standby database

Data Guard broker

CLI management client
Creating a Physical Standby Database
Using SQL and RMAN Commands
Steps to Create a Physical Standby Database

1. Prepare the primary database.
2. Set parameters on the physical standby database.
4. Start the standby database instance.
5. Execute the **DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE** RMAN command or from backup.
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 is recommended to ensure data consistency.
- **FORCE LOGGING** forces redo to be generated even when **NOLOGGING** operations are executed.
- Temporary tablespaces and temporary segments are not logged.
- **FORCE LOGGING** is recommended for both physical and logical standby databases.
- Issue the following command on the primary database:

```
SQL> ALTER DATABASE FORCE LOGGING;
```
Configuring Standby Redo Logs

Online redo logs → Redo shipment → RFS → Standby redo logs

Primary database

Standby database
Creating Standby Redo Logs

Redo from primary database → RFS → Standby redo logs → ARCO → Archived redo logs

MRP/LSP → Standby database
Using SQL to Create Standby Redo Logs

```sql
SQL> ALTER DATABASE ADD STANDBY LOGFILE
  2  '/u01/app/oracle/oradata/orcl/srl01.log'
  3  SIZE 50M;
Database altered.
SQL> ALTER DATABASE ADD STANDBY LOGFILE
  2  '/u01/app/oracle/oradata/orcl/srl02.log'
  3  SIZE 50M;
Database altered.
```
Viewing Standby Redo Log Information

- View information about the standby redo logs:

```
SQL> SELECT group#, type, member FROM v$logfile
WHERE type = 'STANDBY';
GROUP# TYPE MEMBER
-------- ------ -------------------------------
  4 STANDBY /u01/app/oracle/oradata/pc00prmy/sr101.log
  5 STANDBY /u01/app/oracle/oradata/pc00prmy/sr102.log
  6 STANDBY /u01/app/oracle/oradata/pc00prmy/sr103.log
```

```
SQL> SELECT group#, dbid, thread#, sequence#, status FROM v$standby_log;
GROUP# DBID THREAD# SEQUENCE# STATUS
-------- -------- --------- ---------- ------------
  4 UNASSIGNED 0         0             UNASSIGNED
  5 UNASSIGNED 0         0             UNASSIGNED
  6 UNASSIGNED 0         0             UNASSIGNED
```
### Setting Initialization Parameters on the Primary Database to Control Redo Transport

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_ARCHIVE_CONFIG</td>
<td>Specifies the unique database name for each database in the configuration</td>
</tr>
<tr>
<td>LOG_ARCHIVE_DEST_n</td>
<td>Controls redo transport services</td>
</tr>
<tr>
<td>LOG_ARCHIVE_DEST_STATE_n</td>
<td>Specifies the destination state</td>
</tr>
<tr>
<td>ARCHIVE_LAG_TARGET</td>
<td>Forces a log switch after the specified number of seconds</td>
</tr>
<tr>
<td>LOG_ARCHIVE_TRACE</td>
<td>Controls output generated by the archiver process</td>
</tr>
</tbody>
</table>
Setting \texttt{LOG\_ARCHIVE\_CONFIG}

- Specify the \texttt{DG\_CONFIG} attribute to list the \texttt{DB\_UNIQUE\_NAME} for the primary database and each standby database in the Data Guard configuration.

\texttt{LOG\_ARCHIVE\_CONFIG='DG\_CONFIG=(pc00prmy,pc00sby1)'}
Setting \texttt{LOG\_ARCHIVE\_DEST\_n}

- Specify \texttt{LOG\_ARCHIVE\_DEST\_n} parameters for:
  - Local archiving (if not using the flash recovery area)
  - Standby database location
- Include (at a minimum) one of the following:
  - \texttt{LOCATION}: Specifies a valid path name
  - \texttt{SERVICE}: Specifies a valid Oracle Net Services name referencing a standby database
- Include a \texttt{LOG\_ARCHIVE\_DEST\_STATE\_n} parameter for each defined destination.

```
LOG_ARCHIVE_DEST_1=
    'SERVICE=pc00sby1
    VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
    DB_UNIQUE_NAME=pc00sby1'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
```
Specifying Role-Based Destinations

Primary database

log_archive_dest_1 = 'service=pc00sby1 async valid_for= (online_logfile, primary_role) db_unique_name=pc00sby1'

Not used

Standby database

log_archive_dest_1 = 'service=pc00prmy async valid_for= (online_logfile, primary_role) db_unique_name=pc00prmy'
### Combinations for **VALID_FOR**

<table>
<thead>
<tr>
<th>Combination</th>
<th>Primary</th>
<th>Physical</th>
<th>Logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLINE_LOGFILE, PRIMARY_ROLE</td>
<td>Valid</td>
<td>Ignored</td>
<td>Ignored</td>
</tr>
<tr>
<td>ONLINE_LOGFILE, STANDBY_ROLE</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Valid</td>
</tr>
<tr>
<td>ONLINE_LOGFILE, ALL_ROLES</td>
<td>Valid</td>
<td>Ignored</td>
<td>Valid</td>
</tr>
<tr>
<td>STANDBY_LOGFILE, STANDBY_ROLE</td>
<td>Ignored</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>STANDBY_LOGFILE, ALL_ROLES</td>
<td>Ignored</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>ALL_LOGFILES, PRIMARY_ROLE</td>
<td>Valid</td>
<td>Ignored</td>
<td>Ignored</td>
</tr>
<tr>
<td>ALL_LOGFILES, STANDBY_ROLE</td>
<td>Ignored</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>ALL_LOGFILES, ALL_ROLES</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
</tbody>
</table>
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 standby database.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB_FILE_NAME_CONVERT</td>
<td>Converts primary database file names</td>
</tr>
<tr>
<td>LOG_FILE_NAME_CONVERT</td>
<td>Converts primary database log file names</td>
</tr>
<tr>
<td>STANDBY_FILE_MANAGEMENT</td>
<td>Controls automatic standby file management</td>
</tr>
</tbody>
</table>
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 = ('/oracle1/dba/',
                         '/oral/stby_dba/',
                         '/oracle2/dba/',
                         '/ora2/stby_dba'/)
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 = ('/oracle1/logs/','/oral/stby_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**
    - Adds the data file automatically to the standby database.
    - Certain `ALTER` statements are no longer allowed on the standby database.

- **STANDBY_FILE_MANAGEMENT** applies to the primary database and physical standby database.

```
STANDBY_FILE_MANAGEMENT = auto
```
Example: Setting Initialization Parameters on the Primary Database

```sql
DB_NAME=pc00prmy
DB_UNIQUE_NAME=pc00prmy
LOG_ARCHIVE_CONFIG='DG_CONFIG=(pc00prmy,pc00sby1)'
CONTROL_FILES='/u01/app/oracle/oradata/pc00prmy/control1.ctl',
  '/u01/app/oracle/oradata/pc00prmy/control2.ctl'
LOG_ARCHIVE_DEST_1=
  'SERVICE=pc00sby1
   VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
   DB_UNIQUE_NAME=pc00sby1'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
LOG_ARCHIVE_FORMAT=%t_%s_%r.arc
```
Creating an Oracle Net Service Name for Your Physical Standby Database

- Update the `tnsnames.ora` file on the Primary server:

```sql
PC00SBY1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)
        (HOST = edt3r17p1.us.oracle.com)
        (PORT = 1521))
    )
  (CONNECT_DATA =
    (SERVICE_NAME = pc00sby1.us.oracle.com)
  )
)
```
Secure the transfer using SSH tunnels

- You may not wish your redo data being sent unencrypted across the internet to your standby. You can use ssh tunnels to avoid this.

```
SH> ssh -N -L 3333:standby:1521 oracle@standby
```

- Now the tnsnames entry points to the localhost.

```sql
PC00SBY1 =
  (DESCRIPTION =
  (SDU = 32767)
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)
        (HOST = edt3r17p1.us.oracle.com)
        (PORT = 3333))
    )
  (CONNECT_DATA =
    (SERVICE_NAME = pc00sby1.us.oracle.com)
  )
)```


Creating an Entry for Your Standby Database for the Listener

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

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = pc00sby1.us.oracle.com)
      (ORACLE_HOME = /u01/app/oracle/product/11.1.0/db_1)
      (SID_NAME = pc00sby1)
    )
  )
```
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 a single parameter:

```
DB_NAME=pc00sby1
```
Creating Directories for the Physical Standby Database

1. Create the audit trail directory in $ORACLE_BASE/admin:

[oracle@edt3r17p1-orcl ~]$ cd /u01/app/oracle/admin
[oracle@edt3r17p1-orcl admin]$ ls
orcl
[oracle@edt3r17p1-orcl admin]$ mkdir pc00sby1
[oracle@edt3r17p1-orcl admin]$ cd pc00sby1
[oracle@edt3r17p1-orcl orclsby1]$ mkdir adump

2. Create a directory for the data files in the $ORACLE_BASE/oradata directory:

[oracle@edt3r17p1-orcl oradata]$ mkdir pc00sby1
[oracle@edt3r17p1-orcl oradata]$ ls
orcl pc00sby1
Starting the Physical Standby Database

- Start the physical standby database in NOMOUNT mode:

```
SQL> startup nomount pfile=$HOME/dbs/pc00sby1.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
```
Setting **FAL_CLIENT** and **FAL_SERVER** Initialization Parameters

- **FAL_CLIENT**:
  - Specifies the FAL client name that is used by the FAL service

- **FAL_SERVER**:
  - Specifies the FAL server for a standby database

- **Fetch archive log (FAL)**:
  - Provides a client/server mechanism for resolving gaps detected in the range of archived redo logs that are generated at the primary database and received at the standby database.
  - Applicable for physical standby databases only.
  - Process is started only when needed, and shuts down as soon as it is finished.

**FAL_CLIENT = 'pc00sby1'**

**FAL_SERVER = 'pc00prmy'**
Creating an RMAN Script to Create the Physical Standby Database

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

```sql
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
}
```
spfile

parameter_value_convert 'pc00prmy','pc00sby1'
set db_unique_name='pc00sby1'
set db_file_name_convert='/pc00prmy/','/pc00sby1/
set log_file_name_convert='/pc00prmy/','/pc00sby1/
set control_files=
    '/u01/app/oracle/oradata/pc00sby1.ctl'
set log_archive_max_processes='5'
set fal_client='pc00sby1'
set fal_server='pc00prmy'
set standby_file_management='AUTO'
set log_archive_config='dg_config=(pc00prmy,pc00sby1)'
set log_archive_dest_1='service=pc00prmy ASYNC
    valid_for=(ONLINE_LOGFILE,PRIMARY_ROLE)
    db_unique_name=pc00prmy';
Creating the Physical Standby Database

1. Invoke RMAN and connect to the primary database and the physical standby database.
2. Execute the RMAN script to create the physical standby database.

RMAN> connect target sys/oracle
RMAN> connect auxiliary sys/oracle@pc00sby1
RMAN> @cr_phys_standby
Starting Redo Apply

• Execute the following command on the standby database to start Redo Apply:

```sql
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT FROM SESSION;
```
Enabling Real-Time Apply

Primary database

RFS

Standby redo log files

ARC0

Archived redo log files

MRP

Standby database
Real Time Apply Benefits

• Standby databases now more closely synchronized with the primary
  – More up-to-date, real-time reporting
  – Faster switchover and failover times
    • Reduces planned and unplanned downtime
    • Better Recovery Time Objective (RTO) for DR
DEMO
Special Note:
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.
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.
Oracle Data Guard Broker: Overview
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 Grid Control
  – DGMGRL (a command-line interface)
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 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

<table>
<thead>
<tr>
<th></th>
<th>With the Broker</th>
<th>Without the Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td>Manage databases as one</td>
<td>Manage databases separately</td>
</tr>
<tr>
<td><strong>Creation of the standby database</strong></td>
<td>Use Grid Control wizards</td>
<td>Manually create files</td>
</tr>
<tr>
<td><strong>Configuration and management</strong></td>
<td>Configure and manage from single interface</td>
<td>Set up services manually for each database</td>
</tr>
<tr>
<td><strong>Monitoring</strong></td>
<td>• Monitor continuously</td>
<td>Monitor each database individually through views and scripts</td>
</tr>
<tr>
<td></td>
<td>• Unified status and reports</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Integrate with EM events</td>
<td></td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>Invoke role transitions with a single command</td>
<td>Coordinate sequences of multiple commands across database sites for role transitions</td>
</tr>
</tbody>
</table>
Data Guard Broker: Components

- **Client-side:**
  - Oracle Enterprise Manager Grid 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 Configuration

Broker-controlled databases

Primary database

Instances

Standby database

Instances

Standby database
Data Guard Broker: Architecture

Graphical user interface or command-line interface

Data Guard Configuration

Primary site
- Primary database
- Online redo logs

Standby site
- Standby redo logs
- Configuration files

Configured by DMON

Standby site 9
- Standby site 2
- Standby site 1

Standby database
- Online redo logs
- Archived redo logs

Archived redo logs

Log transport services

Oracle Net

Log apply services

Oracle
Data Guard Monitor: **DMON Process**

- Server-side background process
- 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
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 Grid Control:
  – Provides wizards to simplify creating and managing standby databases
Using the Command-Line Interface of the Data Guard Broker

DGMGRL> connect sys/oracle
Connected.
DGMGRL> show configuration verbose

Configuration
Name: DGConfig1
Enabled: YES
Protection Mode: MaxAvailability
Databases:
  pc00prmy - Primary database
  pc00sbyl - Physical standby database

Fast-Start Failover: DISABLED

Current status for "DGConfig1": SUCCESS
Using Oracle Enterprise Manager 10g

Grid Control

Click “Setup and Manage” to access the Data Guard pages.
Data Guard Overview Page

**Database Instance:** pc00prmy.us.oracle.com

**Data Guard**

*Page Refreshed February 15, 2008 1:48:57 PM EST*

**Overview**

- **Data Guard Status:** Normal
- **Protection Mode:** Maximum Availability
- **Fast-Start Failover:** Disabled

**Primary Database**

- **Name:** pc00prmy.us.oracle.com
- **Host:** edt3r17p0.us.oracle.com
- **Data Guard Status:** Normal
- **Current Log Properties:** 38

**Standby Databases**

<table>
<thead>
<tr>
<th>Select Name</th>
<th>Host</th>
<th>Data Guard Status</th>
<th>Role</th>
<th>Last Received Log</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>pc00sby1.us.oracle.com</td>
<td>edt3r17p2.us.oracle.com</td>
<td>✔️ Normal</td>
<td>Physical Standby</td>
<td>37</td>
<td>36</td>
</tr>
</tbody>
</table>
Creating a Data Guard Broker Configuration
Data Guard Broker: Requirements

- Oracle Database Enterprise Edition
- Single-instance or multi-instance environment
- `COMPATIBLE` parameter: Set to 10.2.0.1.0 or later for primary and standby databases
- Oracle Net Services network files: Must be configured for the primary database and any existing standby databases. Enterprise Manager Grid 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_FILE**: Configured for any RAC databases
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 Grid 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 database 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. Invoke DGMGRL and connect to the primary database.
2. Define the configuration, including a profile for the primary database.
3. Add standby databases to the configuration.
4. Enable the configuration, including the databases.
Defining the Broker Configuration and the Primary Database Profile

DGMGRL> CREATE CONFIGURATION 'DGConfig1' AS
> PRIMARY DATABASE IS pc00prmy
> CONNECT IDENTIFIER IS pc00prmy;
Configuration "DGConfig1" created with primary database "pc00prmy"
DGMGRL>
Adding a Standby Database to the Configuration

DGMGRL> ADD DATABASE pc00sby1 AS
> CONNECT IDENTIFIER IS pc00sby1;
Database "pc00sby1" added
DGMGRL>
**Enabling the Configuration**

DGMGRL> ENABLE CONFIGURATION;
Enabled.
DGMGRL> SHOW CONFIGURATION

<table>
<thead>
<tr>
<th>Configuration</th>
<th>DGConfig1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled:</td>
<td>YES</td>
</tr>
<tr>
<td>Protection Mode:</td>
<td>MaxPerformance</td>
</tr>
<tr>
<td>Databases:</td>
<td></td>
</tr>
<tr>
<td>pc00prmy - Primary database</td>
<td>pc00sby1 - Physical standby database</td>
</tr>
<tr>
<td>Fast-Start Failover:</td>
<td>DISABLED</td>
</tr>
</tbody>
</table>

Current status for "DGConfig1": SUCCESS
Demo (cont.)
Changing Database Properties and States

- To alter a database property:

  ```
  DGMGRL> EDIT DATABASE pc00sby1
  > SET PROPERTY LogXptMode='SYNC';
  ```

- To alter the state of the standby database:

  ```
  DGMGRL> EDIT DATABASE pc00sby1 SET STATE='APPLY-OFF';
  ```

- To alter the state of the primary database:

  ```
  DGMGRL> EDIT DATABASE pc00prmy
  > SET STATE='TRANSPORT-OFF';
  ```
Managing Redo Transport Services by Using DGMGRL

• Specify database properties to manage redo transport services:
  – DGConnectIdentifier
  – LogXptMode
  – LogShipping
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
    • Required for maximum performance mode
  – SYNC
    • Sets the SYNC and AFFIRM attributes of LOG_ARCHIVE_DEST_n
    • Required for maximum protection and maximum availability modes
**Setting LogXptMode to ASYNC**

- **Primary database transactions**
  - Redo buffer
  - LGWR
  - Online redo logs
  - Sets the `ASYNC` and `NOAFFIRM` attributes of `LOG_ARCHIVEDEST_n`

- **Standby database**
  - MRP or LSP
  - Standby redo logs
  - ARCO
  - Archived redo logs

- **Oracle Net**
  - Standby ack
  - (Real-time apply)
Setting `LogXptMode` to `SYNC`

- Primary database transactions
- Redo buffer
- LGWR
- Online redo logs
- `LGWR` sets the `SYNC` and `AFFIRM` attributes of `LOG_ARCHIVE_DEST_n`
- `LNSn`
- `RFS`
- Standby ack
- MRP or LSP
- Standby database
- Archived redo logs
- Oracle
- `LGWR` sends online redo logs to `RFS`
- `RFS` sends the `standby ack` to `LNSn`
- `LNSn` sends the redo logs to the `standby redo logs`
- `ARCO` (Real-time apply)
Controlling the Shipping of Redo Data by Using the LogShipping Property

• **LogShipping** controls whether archived redo log files are sent to a specified standby database.

• **LogShipping** is applicable only when the primary database state is set to **TRANSPORT-ON**.
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 'pc00sby1';
```
Removing the Configuration or Standby Database

- Remove a standby database from the configuration:

```sql
DGMGRL> REMOVE DATABASE 'pc00sby1';
```

- Remove the configuration:

```sql
DGMGRL> REMOVE CONFIGURATION;
```
Creating a Logical Standby Database
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

- Production database
- Redo transport
- Primary database
- SQL Apply
- Transform redo information into SQL
- Logical standby database
- Reports
SQL Apply Process: Architecture

Redo data from primary database

Reader -> Preparer
Redo records
Log Mining

Apply processing
Applier -> Coordinator
Transactions to be applied

Shared pool
Logical change records not grouped into transactions

Builder
Transaction groups

Data files

Analyzer
Transactions sorted in dependency order

LCR
LCR
Preparing to Create a Logical Standby Database

• Perform the following steps on the primary database before creating a logical standby database:
  – Check for unsupported data types.
  – Be aware of unsupported DDL commands.
  – Ensure row uniqueness.
  – Verify that the primary database is configured for ARCHIVELOG mode.
Unsupported Objects

• 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 using table compression
  – Tables used to support materialized views
  – Global temporary tables
  – Tables with unsupported data types (see list on next page)
Unsupported Data Types

- Log apply services *automatically exclude* tables with unsupported data types when applying redo data to the logical standby database.
- Unsupported data types:
  - BFILE, ROWID, and UROWID
  - User-defined types
  - Multimedia data types (Spatial, Image, and Oracle Text)
  - Collections (VARRAYS and nested tables)
  - LOBs stored as SecureFiles
  - XMLType **stored as** OBJECT RELATIONAL
  - BINARY XML
Checking for Unsupported Tables

- Query `DBA_LOGSTDBY_UNSUPPORTED_TABLE` on the primary database for unsupported tables:

```sql
SQL> SELECT * FROM dba_logstdby_unsupported_table
    2  ORDER BY owner;
OWNER                              TABLE_NAME
------------------------------- --------------------------------------------
IX                                AQ$_STREAMS_QUEUE_TABLE_T
IX                                AQ$_STREAMS_QUEUE_TABLE_H
...                               
OE                                CUSTOMERS
OE                                WAREHOUSES
PM                                PRINT_MEDIA
PM                                ONLINE_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 unsupported data types:

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

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>COLUMN_NAME</th>
<th>ATTRIBUTES</th>
<th>DATA_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERS</td>
<td>CUST_ADDRESS</td>
<td></td>
<td>OBJECT</td>
</tr>
<tr>
<td>CUSTOMERS</td>
<td>PHONE_NUMBERS</td>
<td></td>
<td>VARRAY</td>
</tr>
<tr>
<td>CUSTOMERS</td>
<td>CUST_GEO_LOCATION</td>
<td></td>
<td>OBJECT</td>
</tr>
<tr>
<td>WAREHOUSES</td>
<td>WH_GEO_LOCATION</td>
<td></td>
<td>OBJECT</td>
</tr>
<tr>
<td>PURCHASEORDER</td>
<td>SYS_NC_ROWINFO$</td>
<td>Object Table</td>
<td>OPAQUE</td>
</tr>
<tr>
<td>CATEGORIES_TAB</td>
<td>CATEGORY_NAME</td>
<td>Object Table</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>CATEGORIES_TAB</td>
<td>CATEGORY_DESCRIPTION</td>
<td>Object Table</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>CATEGORIES_TAB</td>
<td>CATEGORY_ID</td>
<td>Object Table</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CATEGORIES_TAB</td>
<td>PARENT_CATEGORY_ID</td>
<td>Object Table</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

9 rows selected.
SQL Commands That Do Not Execute on the Standby Database

- ALTER DATABASE
- ALTER SESSION
- ALTER MATERIALIZED VIEW
- ALTER MATERIALIZED VIEW LOG
- ALTER SYSTEM
- CREATE CONTROL FILE
- CREATE DATABASE
- CREATE DATABASE LINK
- CREATE PFILE FROM SPFILE
- CREATE SCHEMA AUTHORIZATION
- CREATE MATERIALIZED VIEW
- CREATE MATERIALIZED VIEW LOG
- CREATE SPFILE FROM PFILE
- DROP DATABASE LINK
- DROP MATERIALIZED VIEW
- DROP MATERIALIZED VIEW LOG

- EXPLAIN
- LOCK TABLE
- SET CONSTRAINTS
- SET ROLE
- SET TRANSACTION
Ensuring Unique Row Identifiers

• Query `dba_logstdby_not_unique` on the primary database to find tables without a unique identifier:

```sql
SQL> SELECT * FROM dba_logstdby_not_unique;

OWNER   TABLE_NAME                  BAD_COLUMN
------   --------------------------    ----------
HR       EMP_HIST                    N
SCOTT    BONUS                       N
SCOTT    SALGRADE                    N
SH       SUPPLEMENTARY_DEMOGRAPHICS  N
SH       COSTS                       N
SH       SALES                      N
```

• 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.
5. Transition 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
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

`or`

```sql
DGMGRL> EDIT DATABASE 'pc00sbyl' set state='apply-off';
Succeeded.
```
Step 3: Prepare the Primary Database to Support a Logical Standby Database

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

```sql
LOG_ARCHIVE_DEST_2=
    'LOCATION=<directory>
    VALID_FOR=(STANDBY_LOGFILES, STANDBY_ROLE)
    DB_UNIQUE_NAME=pc00prmy'
LOG_ARCHIVE_DEST_STATE_2=enable
```

b. Set the value of `UNDO_RETENTION` to 3600.
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;
```

b. Shut down the logical standby database instance and restart it in MOUNT mode.

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

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

   ```sql
   SQL> ALTER DATABASE START LOGICAL STANDBY 2 APPLY IMMEDIATE;
   ```
Adding a Logical Standby Database to a Data Guard Broker Configuration

- Use DGMGRL to add a logical standby database to an existing Data Guard configuration:

```
DGMGRL> ADD DATABASE 'pc00sby2' AS > CONNECT IDENTIFIER IS pc00sby2;
Database "pc00sby2" added
DGMGRL>
```
Step 7: Verify That the Logical Standby Database Is Performing Properly

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

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

b. Begin sending redo data to the standby database:

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

c. Query the `DBA_LOGSTDDBY_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
SQL> SELECT name, value FROM v$logstdby_stats
  2   WHERE name = 'coordinator state';
```

e. Query the `V$LOGSTDBY_PROCESS` view to see current SQL Apply activity:

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

f. Check the overall progress of SQL Apply:

```sql
SQL> SELECT applied_scn, latest_scn
  2   FROM v$logstdby_progress;
```
Creating a Logical Standby Database by Using Enterprise Manager

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. If the list contains tables that you require to be maintained in the standby database, consider creating a physical script.

<table>
<thead>
<tr>
<th>Owner</th>
<th>Table Name</th>
<th>Column Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OE</td>
<td>CATEGORIES_TAB</td>
<td>CATEGORY_NAME</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>OE</td>
<td>PURCHASEORDER</td>
<td>SYS_NC_ROWINFO$</td>
<td>OPAQUE</td>
</tr>
<tr>
<td>PM</td>
<td>ONLINE_MEDIA</td>
<td>PRODUCT_TESTIMONIALS</td>
<td>OBJECT</td>
</tr>
<tr>
<td>PM</td>
<td>PRINT_MEDIA</td>
<td>AD_TEXTDOCS_NTAB</td>
<td>NESTED TABLE</td>
</tr>
<tr>
<td>PM</td>
<td>PRINT_MEDIA</td>
<td>AD_GRAPHIC</td>
<td>BFILE</td>
</tr>
<tr>
<td>PM</td>
<td>PRINT_MEDIA</td>
<td>AD_HEADER</td>
<td>OBJECT</td>
</tr>
<tr>
<td>PM</td>
<td>ONLINE_MEDIA</td>
<td>PRODUCT_PHOTO_SIGNATURE</td>
<td>OBJECT</td>
</tr>
<tr>
<td>PM</td>
<td>ONLINE_MEDIA</td>
<td>PRODUCT_AUDIO</td>
<td>OBJECT</td>
</tr>
<tr>
<td>PM</td>
<td>ONLINE_MEDIA</td>
<td>PRODUCT_VIDEO</td>
<td>OBJECT</td>
</tr>
<tr>
<td>PM</td>
<td>ONLINE_MEDIA</td>
<td>PRODUCT_THUMBNAIL</td>
<td>OBJECT</td>
</tr>
</tbody>
</table>
Using the Add Standby Database Wizard

**Add Standby Database: Configuration**

- **Primary Database**: pc00prmy.us.oracle.com
- **Primary Host**: ed3r17p0.us.oracle.com
- **Standby Host**: ed3r17p2.us.oracle.com

The following standby database parameters must be set. Default values are provided.

- **Database Name**: pc00sby3
- **Database Unique Name**: pc00sby3
- **Target Name**: pc00sby3
- **Standby Archive Location**: /u01/app/oracle/product/11.1.0/db_1

This name will be used for the standby database db_name parameter.

This name will be used for the standby database DB_UNIQUE_NAME parameter.

This name will be used by Enterprise Manager for the new standby database and recommends that this name be the same as the Database Unique Name.

Archived redo log files received from the primary database will be put in this location longer needed.
Using the Add Standby Database Wizard

Standby database `pc00sby3` will be created by job `DataGuardCreateStandby3` and added to

<table>
<thead>
<tr>
<th>Standby Database</th>
<th>pc00sby3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Name</td>
<td>pc00sby3</td>
</tr>
<tr>
<td>Database Name</td>
<td>pc00sby3</td>
</tr>
<tr>
<td>Instance Name</td>
<td>pc00sby3</td>
</tr>
<tr>
<td>Oracle Server Version</td>
<td>11.1.0.6.0</td>
</tr>
<tr>
<td>Oracle Home Host</td>
<td>/u01/app/oracle/product/11.1.0/db_1/edt3r17p2.us.oracle.com</td>
</tr>
<tr>
<td>Operating System</td>
<td>Enterprise Linux Enterprise Linux AS release 4 (October Update 5) 2.6.9</td>
</tr>
<tr>
<td>New backup</td>
<td>oracle</td>
</tr>
<tr>
<td>Directory Location</td>
<td>/u01/app/oracle/product/11.1.0/db_1/dbs</td>
</tr>
<tr>
<td>File Transfer Method</td>
<td>HTTP server</td>
</tr>
<tr>
<td>Database Unique Name</td>
<td>pc00sby3</td>
</tr>
<tr>
<td>Logical Standby</td>
<td>/u01/app/oracle/product/11.1.0/db_1/oradata(pc00sby3/arc</td>
</tr>
</tbody>
</table>
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

Redo logs from primary database → SQL Apply → Transform redo information into SQL → Logical standby database

Delete redo log files
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 flash recovery area is not being used to store remote archived logs
  - Has a default value of 1,440 minutes
Managing SQL Apply Filtering

- Use the following configurable database properties to define a SQL Apply filter:
  - `LsbyASkipCfgPr`: SQL Apply should ignore (skip) SQL statements as specified.
  - `LsbyASkipErrorCfgPr`: SQL Apply should ignore (skip) errors as specified.
  - `LsbyASkipTxFnCfgPr`: SQL Apply should ignore (skip) transactions as specified.
Managing SQL Apply Filtering

• Use the following configurable database properties to delete a previously defined SQL Apply filter:
  – LsbyDSkipCfgPr: Deletes an existing SQL Apply skip specification
  – LsbyDSkipErrorCfgPr: Deletes an existing SQL Apply skip error specification
  – LsbyDSkipTxnCfgPr: Reverses or removes the actions of the LsbyASkipTxnCfgPr property
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';

<table>
<thead>
<tr>
<th>ERROR</th>
<th>STATEMENT_OPT</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>DML</td>
<td>JOBS</td>
</tr>
</tbody>
</table>
```
Managing SQL Apply Filtering by Using Enterprise Manager

**Add Skip Table Entry**
Statements that fail cause the Logical Apply Engine to stop. Skip these statements and errors to avoid stopping the engine.

**TIP** Click on OK to add the changes to the table. Then click on Apply to permanently save the changes.

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>DML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>HR</td>
</tr>
<tr>
<td>Object Name</td>
<td>JOBS</td>
</tr>
<tr>
<td>Stored Procedure</td>
<td></td>
</tr>
<tr>
<td>Skip Statement Type</td>
<td>Always skip this statement type.</td>
</tr>
</tbody>
</table>

**Skip Table Entries**
Specifies what SQL statements that you do not want applied to the logical standby database.

**TIP** The entry has been added to the table. Click on apply to permanently save the changes.
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.
Configuring Data Protection Modes
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 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 the standby redo log on at least one synchronized standby database.
- The primary database does not shut down if it cannot write to at least one synchronized standby database.
- 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 \texttt{SYNC} and \texttt{AFFIRM} redo transport attributes.
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

<table>
<thead>
<tr>
<th>Mode</th>
<th>Risk of Data Loss</th>
<th>Transport</th>
<th>If no acknowledgment is received:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Protection</td>
<td>Zero data loss</td>
<td>SYNC</td>
<td>Stall primary until an acknowledgement is received</td>
</tr>
<tr>
<td></td>
<td>Double failure protection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Availability</td>
<td>Zero data loss</td>
<td>SYNC</td>
<td>Stall primary until threshold period expires, then resume processing</td>
</tr>
<tr>
<td>Maximum Performance</td>
<td>Potential for minimal data loss</td>
<td>ASYNC</td>
<td>Primary never waits for standby acknowledgement</td>
</tr>
</tbody>
</table>
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
   - Maximum performance: ASYNC
3. Set the data protection mode.

```sql
DGMGRL> EDIT DATABASE 'pc00sby1' SET PROPERTY 'LogXptMode'='SYNC';
Property "LogXptMode" updated
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MAXAVAILABILITY;
Succeeded.
```
Setting the Data Protection Mode

Click the Protection Mode link.
Setting the Data Protection 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 log 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 log 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 log transport mode. Can also be used with the ARCH log transport mode.
Monitoring Data Guard
Monitoring the Data Guard Configuration by Using Enterprise Manager Grid 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

Overview
- Data Guard Status: Normal
- Protection Mode: Maximum Availability
- Fast-Start Failover: Disabled

Primary Database
- Name: pc02prmy.us.oracle.com, edtr14p2.us.oracle.com
- Data Guard Status: Normal
- Current Log Properties: Edit

Standby Progress Summary
The transport lag is the time difference between the primary last update and the standby last apply.

Standby Databases
- Name: pc02stby1.us.oracle.com, edtr14p1.us.oracle.com
  - Data Guard Status: Normal
  - Role: Physical Standby
  - Last Resolved Log: 56
  - Last Applied Log: 56
Monitoring Data Guard Performance

**Performance Overview**
Page Refreshed November 5, 2007 4:55:24 PM EST

**Overview**
Primary Database Name: pc00prmy.us.oracle.com
Data Guard Status: Normal

**Test Application**
Run the test application to generate a load on the primary database.
Start | Stop

✔️ **TIP** Click on any of the charts for historical information.

**pc00sby1.us.oracle.com - Lag Times**

- Current Transport Lag (minutes): 15.4
- Current Apply Lag (minutes): 15.5

**pc00sby1.us.oracle.com - Apply Rate**

- Current Apply Rate (KB/sec): 2351

**pc00prmy.us.oracle.com - Redo Generation Rate**

- Current Redo Generation Rate (KB/sec): No data
Viewing 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. Related log transport and apply information is also displayed for diagnostic purposes.

**Primary Database Log File Details**

<table>
<thead>
<tr>
<th>Log Status</th>
<th>ResetLogs ID #</th>
<th>First Change # (SCN)</th>
<th>Last Change # (SCN)</th>
<th>Size (KB)</th>
<th>Time Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Applied</td>
<td>637377852</td>
<td>1600678</td>
<td>1600700</td>
<td>8 Nov 6, 2007 3:29:11 AM</td>
<td></td>
</tr>
</tbody>
</table>
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
  - InconsistentProperties
  - InconsistentLogXptProps

```
DGMGRL> SHOW DATABASE 'pc00prmy' 'StatusReport';
```
Using Monitorable Database Properties to Identify a Failure

1. Check the configuration status:
   
   ```
   DGMGRL> SHOW CONFIGURATION;
   ```

2. Check the database status:
   
   ```
   DGMGRL> SHOW DATABASE 'pc00prmy';
   ```

3. Check the \texttt{StatusReport} monitorable database property:
   
   ```
   DGMGRL> SHOW DATABASE 'pc00prmy' 'StatusReport';
   ```

4. Check additional monitorable database properties as appropriate.
Setting the `LOG_ARCHIVE_TRACE` Initialization Parameter

- `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.
  - On the standby database, processes write an audit trail of the archived logs received from the primary database into a trace file.
- Trace files are written to the Automatic Diagnostic Repository, the location of which is specified by the `DIAGNOSTIC_DEST` initialization parameter.
Monitoring Redo Apply by Querying

### V$MANAGED_STANDBY

```sql
SQL> SELECT process, status, group#, thread#, sequence#
    2   FROM v$managed_standby
    3   order by process, group#, thread#, sequence#;
```

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>STATUS</th>
<th>GROUP#</th>
<th>THREAD#</th>
<th>SEQUENCE#</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCH</td>
<td>CLOSING</td>
<td>4</td>
<td>1</td>
<td>142</td>
</tr>
<tr>
<td>ARCH</td>
<td>CLOSING</td>
<td>4</td>
<td>1</td>
<td>146</td>
</tr>
<tr>
<td>ARCH</td>
<td>CLOSING</td>
<td>4</td>
<td>1</td>
<td>148</td>
</tr>
<tr>
<td>ARCH</td>
<td>CLOSING</td>
<td>5</td>
<td>1</td>
<td>141</td>
</tr>
<tr>
<td>ARCH</td>
<td>CLOSING</td>
<td>5</td>
<td>1</td>
<td>147</td>
</tr>
<tr>
<td>MRP0</td>
<td>APPLYING_LOG</td>
<td>N/A</td>
<td>1</td>
<td>149</td>
</tr>
<tr>
<td>RFS</td>
<td>IDLE</td>
<td>2</td>
<td>1</td>
<td>149</td>
</tr>
<tr>
<td>RFS</td>
<td>IDLE</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RFS</td>
<td>IDLE</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Evaluating Redo Data by Querying `V$DATAGUARD_STATS`

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

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>TIME_COMPUTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>apply finish time</td>
<td>+00 00:00:00.0</td>
<td>13-FEB-2008 02:44:21</td>
</tr>
<tr>
<td>apply lag</td>
<td>+00 00:00:00</td>
<td>13-FEB-2008 02:44:21</td>
</tr>
<tr>
<td>estimated startup time</td>
<td>12</td>
<td>13-FEB-2008 02:44:21</td>
</tr>
<tr>
<td>standby has been open</td>
<td>N</td>
<td>13-FEB-2008 02:44:21</td>
</tr>
<tr>
<td>transport lag</td>
<td>+00 00:00:00</td>
<td>13-FEB-2008 02:44:21</td>
</tr>
</tbody>
</table>
### Viewing Data Guard Status Information by Querying `V$DATAGUARD_STATUS`

```sql
SQL> SELECT timestamp, facility, dest_id, message_num, error_code, message
FROM v$dataguard_status
ORDER by timestamp;
```

<table>
<thead>
<tr>
<th>TIMESTAMP</th>
<th>FACILITY</th>
<th>DEST_ID</th>
<th>MESSAGE_NUM</th>
<th>ERROR_CODE</th>
<th>MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-FEB-08</td>
<td>Log Apply Servi</td>
<td>0</td>
<td>562</td>
<td>0</td>
<td>Media Recovery Waiting for thread 1 sequence 151</td>
</tr>
<tr>
<td>13-FEB-08</td>
<td>Remote File Ser</td>
<td>0</td>
<td>563</td>
<td>0</td>
<td>Primary database is in MAXIMUM AVAILABILITY mode</td>
</tr>
<tr>
<td>13-FEB-08</td>
<td>Remote File Ser</td>
<td>0</td>
<td>564</td>
<td>0</td>
<td>Standby controlfile consistent with primary</td>
</tr>
<tr>
<td>13-FEB-08</td>
<td>Remote File Ser</td>
<td>0</td>
<td>565</td>
<td>0</td>
<td>RFS[25]: Successfully opened standby log 5: '/u01/app/oracle/oradata/pc00sby1/sr102.log'</td>
</tr>
</tbody>
</table>
Optimizing a Data Guard Configuration
Monitoring Configuration Performance by Using Enterprise Manager Grid Control

- Review detailed performance statistics on the Performance Overview page.

![Graph showing performance statistics](image-url)
Optimizing Redo Transport Services

• Optimize redo transport with the following techniques:
  – Optimizing asynchronous redo transmission by using multiple archiver processes
  – Compressing redo data
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.
- Setting is propagated to the **REOPEN** attribute of the **LOG_ARCHIVE_DEST_n** initialization parameter of the primary database.

```
DGMGRL> EDIT DATABASE 'pc00sby1'
> 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
- Setting is propagated to the `NET_TIMEOUT` attribute of the `LOG_ARCHIVE_DEST_n` initialization parameter of the primary database.

```
DGMGRL> EDIT DATABASE 'pc00sby1'
> SET PROPERTY 'NetTimeout'=20;
```
Optimizing Redo Transmission by Setting MaxConnections

Primary database transactions

Online redo logs

ARC0, ARC1, ARC2

Archived redo logs

LGWR, LNSn

RFS

MRP or LSP

Standby database

Oracle Net

Backup

Reports
Setting the MaxConnections Database Property

• This property specifies the number of archiver processes that are used to transmit redo data from a single archived redo log on the primary database to the archived redo log at the remote site for gap resolution.
• Broker default: 1
• Setting is propagated to the MAX_CONNECTIONS attribute of the LOG_ARCHIVE_DEST_n initialization parameter of the primary database.

DGMGRL> EDIT DATABASE 'pc00sby1'
> SET PROPERTY 'MaxConnections'=5;
Compressing Redo Data by Setting the **RedoCompression** Property

- This property enables compression of archived redo log files during gap transmission to the standby database.
- Setting is propagated to the **COMPRESSION** attribute of the `LOG_ARCHIVE_DEST_n` initialization parameter.
- Determine if redo compression is enabled by querying the **COMPRESSION** column of the `V$ARCHIVE_DEST` view.

```
DGMGRL> EDIT DATABASE 'pc00sby1'
> SET PROPERTY 'RedoCompression'='ENABLE';
```
Compressing Redo Data by Setting the RedoCompression Property

- This property enables compression of archived redo log files during gap transmission to the standby database.
- Setting is propagated to the COMPRESSION attribute of the LOG_ARCHIVE_DEST_n initialization parameter.
- Determine if redo compression is enabled by querying the COMPRESSION column of the V$ARCHIVE_DEST view.

DGMGRL> EDIT DATABASE 'pc00sbyl'
> SET PROPERTY 'RedoCompression'='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)
- Setting is propagated to the DELAY attribute of the LOG_ARCHIVE_DEST_n initialization parameter of the primary database.

```
DGMGRL> EDIT DATABASE 'pc00sby1'
> SET PROPERTY 'DelayMins'=5;
```
Using Enterprise Manager to Delay the Application of Redo

Specify the delay in minutes.
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: Number of APPLIER processes that are used to apply changes
  - MAX_SERVERS: Number of processes that SQL Apply uses to read and apply redo
  - PREPARE_SERVERS: 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.
  - Determine if APPLIER processes are busy:
    
    SQL> SELECT COUNT(*) AS IDLE_APPLIER
       FROM V$LOGSTDBY_PROCESS
       WHERE TYPE = 'APPLIER' and status_code = 16166;

  - Determine if there is enough work available for additional APPLIER processes:
    
    SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS
       WHERE NAME LIKE 'transactions%';

- 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.
  – Determine if all PREPARER processes are busy:
    
    SQL> SELECT COUNT(*) AS IDLE_PREPARER
    2 FROM V$LOGSTDBY_PROCESS
    3 WHERE TYPE = 'PREPARER' and status_code = 16166;
  – Determine if the number of transactions ready to be applied is less than the number of available APPLIER processes:
    
    SQL> SELECT NAME, VALUE FROM V$LOGSTDBY_STATS
    2 WHERE NAME LIKE 'transactions%';
    SQL> SELECT COUNT(*) AS APPLIER_COUNT
    2 FROM V$LOGSTDBY_PROCESS WHERE TYPE = 'APPLIER';
Adjusting the Number of PREPARER Processes

– Determine if there are idle APPLIER processes:

```
SQL> SELECT COUNT(*) AS IDLE_APPLIER
2   FROM V$LOGSTDBY_PROCESS
3   WHERE TYPE = 'APPLIER' and status_code = 16166;

181
```

• Adjust the `MAX_SERVERS` parameter if necessary.
• Adjust the `PREPARE_SERVERS` parameter to increase the number of PREPARER processes.
Oracle Active Data Guard
Oracle Active Data Guard

• Data Guard 11g
  – Recovery (redo apply) must be stopped to open a standby read-only
  – Redo Apply has exclusive access to data files – reads not allowed
  – Not possible to guarantee read consistency while redo apply is active

• Data Guard 11g with the Active Data Guard Option
  – Physical Standby is open read-only while redo apply is active
  – Read consistency is guaranteed
  – Redo apply is not adversely affected by read-only workload
Oracle Active Data Guard

• Is an option for Oracle Database 11g Enterprise Edition and has **extra licensing costs**!
• Enhances quality of service by offloading resource-intensive activities from a production database to a standby database
• Includes the following features:
  – Real-time query
  – RMAN block change tracking on a physical standby database
Using Real-Time Query

Using Real-Time Query

- Primary database
- Redo stream
- Redo transport
- Physical standby database
- Queries
Guarantee of Consistent Reads

- Maintained through Query SCN
  - Identifies most recent read point
  - Used by queries to insure consistent reads
  - Current value given by current_scn from v$database on standby

- Redo Apply advances the Query SCN
  - After all dependent changes have been fully applied
  - Propagated to all other instances in standby RAC
Enabling Active Data Guard

• Begin with a Data Guard 11g physical standby database
  – If redo apply is running, stop redo apply
  – Open the standby database read-only
  – Start redo apply
• If open read-only fails because standby instance was aborted or datafiles were restored then...
  – Bring to mount state and start redo apply
  – Stop redo apply and open read-only
  – Restart redo apply
Enabling Real-Time Query

1. Stop Redo Apply:

DGMGRL> EDIT DATABASE 'pc00sby1' set state='apply-off';
Succeeded.

2. Open the database for read-only access:

SQL> ALTER DATABASE OPEN;
Database altered.

3. Restart Redo Apply:

DGMGRL> EDIT DATABASE 'pc00sby1' SET STATE='apply-on';
Succeeded.
Disabling Real-Time Query

1. Shut down the standby database instance.
2. Restart the standby database instance in MOUNT mode.
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.
Creating Fast Incremental Backups

- Block change tracking optimizes incremental backups:
  - Tracks the blocks that have changed since the last backup
- Oracle Database has integrated change tracking:
  - A change tracking file is used.
  - Changed blocks are tracked as redo is generated.
  - Database backup automatically uses the changed-block list.
Enabling Block Change Tracking

```
ALTER DATABASE {ENABLE | DISABLE} BLOCK CHANGE TRACKING
[USING FILE '...']
```
Monitoring Block Change Tracking

```sql
SQL> SELECT filename, status, bytes
2  FROM v$block_change_tracking;
```

```sql
SQL> SELECT file#, avg(datafile_blocks),
2    avg(blocks_read),
3    avg(blocks_read/datafile_blocks) * 100 AS PCT_READ_FOR_BACKUP,
4    avg(blocks)
5 FROM v$backup_datafile
6 WHERE used_change_tracking = 'YES'
7 AND incremental_level > 0
8 GROUP BY file#;
```
Snapshot 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

Primary database transactions

LGWR

LNSn

RFS

Online redo logs

ARC0

Archived redo logs

Transactions

MRP

Snapshot standby database

Standby redo logs

ARC0

Archived redo logs
Converting a Physical Standby Database to a Snapshot Standby Database

• To convert a physical standby database to a snapshot standby database:

DGMGRL> CONVERT DATABASE pc00sby1
>TO SNAPSHOT STANDBY;

Converting database "pc00sby1" to a Snapshot Standby database, please wait...

Database "pc00sby1" converted successfully
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
```
Using DGMGRL to View Snapshot Standby Database Information

- View snapshot standby information by using the `SHOW CONFIGURATION` and `SHOW CONFIGURATION VERBOSE` commands:

```
DGMGRL> show configuration
Configuration
  Name:           DGConfig
  Enabled:        YES
  Protection Mode: MaxPerformance
  Databases:
    pc00prmy - Primary database
    pc00sby1 - Snapshot standby database

Fast-Start Failover: DISABLED

Current status for "DGConfig": SUCCESS
```
Using DGMGRL to View Snapshot Standby Database Information

• View snapshot standby information by using the `SHOW DATABASE` and `SHOW DATABASE VERBOSE` commands:

```
DGMGRL> show database pc00sby1

Database
 Name: pc00sby1
Role: SNAPSHOT STANDBY
 Enabled: YES
 Intended State: APPLY-OFF
Instance(s):
 pdb1

Current status for "pc00sby1": SUCCESS
```
Converting a Snapshot Standby Database to a Physical Standby Database

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

```sql
DGMGRL> CONVERT DATABASE pc00sby1
>TO PHYSICAL STANDBY;
```
Summary

• We talked about Oracle Data Guard
• We discussed the reasons to use each type of DG
• We saw how to setup physical data guard, logical data guard and a broker
• We talked about maintaining, monitoring and optimizing Data Guard solution
• More info will be available on my blog: ZoharElkayam.wordpress.com
Questions and Answers
Thank You!
Zohar Elkayam
054-5996413
zoharel@glasshouse.com
לתרוצות