



## Active Data Guard Hands On Lab

Larry M. Carpenter  
Distinguished Product manager

Vinay Srihari  
Software Development Director

Nitin Karkhanis  
Senior Manager, Software Development



# **Active Data Guard Hands On Lab Oracle Database 11g Release 2**

*Maximum  
Availability  
Architecture  
(MAA)*

*Oracle Best Practices For High Availability*



<b>INTRODUCTION</b> .....	<b>7</b>
<b>OVERVIEW OF THE EXERCISES</b> .....	<b>9</b>
<b>SETUP AND CONFIGURATION</b> .....	<b>11</b>
SOFTWARE AND DATABASES .....	11
ACCOUNTS AND PASSWORDS .....	11
<i>Required Software</i> .....	11
<i>Creating Your Amazon EC2 Account</i> .....	11
<i>Starting up your lab system</i> .....	11
<b>GETTING STARTED</b> .....	<b>13</b>
LOGIN INTO THE LAB SYSTEM .....	13
CONFIGURING YOUR VNC DESKTOP .....	15
DOCUMENTATION .....	15
ENVIRONMENT SCRIPTS AND COMMANDS .....	15
<b>ENABLING AND USING ACTIVE DATA GUARD</b> .....	<b>17</b>
ACTIVE DATA GUARD BENEFITS .....	18
<i>Active Data Guard Reader Farms</i> .....	18
ENABLING ACTIVE DATA GUARD .....	19
READING FROM AN ACTIVE DATA GUARD STANDBY DATABASE .....	20
MANAGING POTENTIAL APPLY LAGS.....	22
WRITING DATA WITH AN ACTIVE DATA GUARD STANDBY .....	26
<i>Schema Redirecting and Active Data Guard</i> .....	29
AVOIDING MEDIA CORRUPTION WITH ACTIVE DATA GUARD.....	34
<b>CONCLUSION</b> .....	<b>39</b>
<b>RESOURCES</b> .....	<b>41</b>
<b>APPENDIX A – COMMAND FILES</b> .....	<b>43</b>
<b>APPENDIX B – SETTING UP PUTTY TO CONNECT TO YOUR CLOUD INSTANCE</b> .....	<b>47</b>
RUNNING FROM BEHIND A CORPORATE FIREWALL.....	47
CONFIGURING ACCESS TO THE VNC SERVER.....	48
SAVING YOUR PUTTY CONFIGURATION .....	49



# Introduction

Oracle Data Guard ensures high availability, data protection, and disaster recovery for enterprise data. Data Guard provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases to enable production Oracle databases to survive disasters and data corruptions. These standby databases are maintained as transaction consistent copies of the production database.

If the production database becomes unavailable because of a planned or an unplanned outage, Data Guard can switch any standby database to the production role, minimizing the downtime associated with the outage. Data Guard can be used with traditional backup, restoration, and cluster techniques to provide a high level of data protection and data availability.

With Oracle Database 11g Data Guard, administrators can improve production database performance by offloading resource-intensive backup and query/reporting operations to Physical standby databases

A Data Guard configuration consists of one production database and one or more standby databases. The databases in a Data Guard configuration are connected by Oracle Net and may be dispersed geographically. There are no restrictions on where the databases are located, provided they can communicate with each other. For example, you can have a standby database on the same system as the production database, along with two standby databases on other systems at remote locations.

A standby database can be:

- Physical standby database
  - Provides a physically identical copy of the primary database, with on disk database structures that are identical to the primary database on a block-for-block basis. A physical standby database is kept synchronized with the primary database, though Redo Apply, which recovers the redo data, received from the primary database and applies the redo to the physical standby database.
- Logical standby database
  - Contains the same logical information as the production database, although the physical organization and structure of the data can be different. The logical standby database is kept synchronized with the primary database though SQL Apply, which transforms the data in the redo received from the primary database into SQL statements and then executes the SQL statements on the standby database. Some restrictions apply.
- Active Data Guard standby database
  - A Physical standby database that is open to read access with up-to-date data from the Primary database. Ancillary writes are also possible but the writes must be redirected to a read write database using database links.
- Snapshot Standby database
  - A fully read write standby database that is created by converting a physical standby database into a Read write snapshot standby database.
  - Receives and archives redo data from a primary database. A snapshot standby database does not apply the redo data that it receives. The redo data is not applied until the snapshot standby is converted back into a physical standby database, after first discarding any local updates made to the snapshot standby database.

You can manage primary and standby databases using the SQL command-line interface or the Data Guard Broker interfaces. The Data Guard Broker logically groups these primary and standby databases into a Broker configuration that allows the Broker to manage and monitor them together as an integrated unit.

You can manage a Broker configuration using either the Oracle Enterprise Manager graphical user interface or the Data Guard Broker command-line interface (DGMGRL).

This Active Data Guard handbook is your guide to exercising the capabilities of Active Data Guard in Oracle Database 11g Release 2.

# Overview of the Exercises

The exercises are set up to walk you through Oracle Database 11g Active Data Guard SQL\*Plus working with a Physical standby (Redo Apply) database using Active Data Guard to read your data as redo is being applied while maintaining data protection. The exercises are designed from the top down so that each step builds upon the previous exercise. **It is essential that you follow the exercises in order.**

The following is an outline of the exercises that you will perform in Oracle Database 11g.

- **Learn about Active data guard benefits**
- **Learn how to enable Active Data Guard**
- **Read data from an Active Data Guard standby database**
- **Manage potential apply lags**
- **Write data when using an Active Data Guard standby**
- **Use Schema redirection Active Data Guard**
- **Avoid Primary database media corruption with Active Data Guard**

As you go through these exercises look for the line beginning with “**TASK:**” as that will tell you what must be done at each step.

Each TASK refers to commands (SQL\*Plus, DGMGRL or Linux) that are contained in the file listed on the TASK line. These files are all in the subdirectory listed (**/home/oracle/hol/**) and are text files from which you can cut and paste the commands into the appropriate CLI. *Also note task files may contain a log in line. Please make sure to execute the log in line as well as you will be moving from SYS to HR and back periodically.*

**These are NOT scripts to be executed, they are merely there to facilitate cut and paste for each exercise.**

Any DGMGRL commands are to be executed in your DGMGRL terminal window, SQL commands in your SQL\*Plus terminal window. More on setting up the terminal windows is detailed at the end of this section.

**Please read the “Setup and Configuration” section carefully before attempting the exercises.**

**Please note that this Hand On Lab exercise book complete with instructions on setting up your own Amazon account to use the Amazon Machine Image created for this lab will be available on Oracle’s Technology Network called *Active Data Guard Hands-On Lab: 2010* at the following URL:**

**<http://tinyurl.com/DataGuard-HOL>**

**In addition there will be a 2<sup>nd</sup> Hands On Lab book called *Data Guard Complete Hands-On Lab: 2010* available at the same location that is a complete set of exercises for Data Guard, including these Active Data Guard exercises. With the complete guide you will create your standby using RMAN’s ‘from active database’ capability, configure the Data Guard Broker, perform switchovers, failovers (including automatic fail over with Fast-Start Failover), enable client failover using Oracle Restart, use Active Data Guard and convert your standby into a Snapshot Standby database for testing and use Real Application Testing on your standby.**



# Setup and Configuration

You will use the laptop system provided you have for the user interface to the Lab system. Your Lab system is one virtual Linux system on which all the necessary software and databases have been pre-installed. This will be the system where you will perform all of the exercises in this handbook. In a real life environment you would, of course, be using multiple systems.

## *Software and Databases*

The following has been pre-installed on your Lab system:

- Oracle Database 11g (Version 11.2.0.1)
  - A Primary database called ‘SFO’
  - A Physical Standby database called ‘NYC’
  - An OracleNet listener called ‘LISTENER’
  - A Data Guard Broker Configuration called OW2010

## *Accounts and Passwords*

The following accounts have been created on your Lab system.

- VNC password on your Lab system is ‘**oracle**’
- All Database ‘**SYS**’ and ‘**SYSTEM**’ accounts have the password ‘**oracle**’
- The Databases have the ‘**HR**’ account with the password ‘**oracle**’

**In case you missed it, all the passwords are ‘oracle’ with the exception of the normal scott account.**

## Required Software

You will need to install PuTTY and a VNC viewer on your computer to access the Hands On Lab system.

## Creating Your Amazon EC2 Account

To start up a lab system you will need to create an account on the Amazon Elastic Compute Cloud (EC2). To get started, read the “[Amazon EC2 Getting Started Guide](http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/)”<sup>1</sup> for instructions on setting up your “AWS” account and obtain your access key identifiers. You will then have to create your key pair and modify your security group to include SSH access to port 22 for PuTTY. After the getting started steps have been completed, go to the “[PuTTY appendix](#)”<sup>2</sup> and setup PuTTY to configure a secure tunnel to your Amazon Machine Image (AMI). **Note: See Appendix B for the setup required to configure the “Proxy” settings and to add a link to Port 5901 for the VNC server in the PuTTY configuration.** Once you have completed all these tasks, you can use the AWS Console to start up the Hands On Lab AMI.

## Starting up your lab system

Navigate to the [AWS Management Console](#)<sup>3</sup> and log in using the “Sign in to the AWS Console” button. Once the console starts up select the “AMI” link on the left. At the top you will see 4 buttons (3 grayed out and one active) and underneath that the word “Viewing” followed by two drop down menu and an empty box. The two

<sup>1</sup> <http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/>

<sup>2</sup> <http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/ConnectToInstanceLinux.html>

<sup>3</sup> <https://console.aws.amazon.com/>

Menus will say “All Images” and “All Platforms”. In the empty box to the right of these two menus enter the text “OW2010\_ADG\_HOL”. This will display the Active Data Guard Hands On Lab AMI. Select this AMI by checking the check box and then click the “Launch” button at the top (which will have gone from gray to active). You will see the following input form:

**Request Instances Wizard** Cancel

**CHOOSE AN AMI**    **INSTANCE DETAILS**    CREATE KEY PAIR    CONFIGURE FIREWALL    REVIEW

Provide the details for your instance(s). You may also decide whether you want to launch your instances as "on-demand" or "spot" instances.

**Number of Instances:**     **Availability Zone:**

**Instance Type:**

Type	CPU Units	CPU Cores	Memory
Micro (t1.micro)	Up to 2 ECUs	1 Core	613 MB
Large (m1.large)	4 ECUs	2 Cores	7.5 GB
Extra Large (m1.xlarge)	8 ECUs	4 Cores	15 GB
High-Memory Extra Large (m2.xlarge)	6.5 ECUs	2 Cores	17.1 GB
High-Memory Double Extra Large (m2.2xlarge)	13 ECUs	4 Cores	34.2 GB
High-Memory Quadruple Extra Large (m2.4xlarge)	26 ECUs	8 Cores	68.4 GB
High-CPU Extra Large (c1.xlarge)	20 ECUs	8 Cores	7 GB

**Launch Instances**  
 EC2 Instances let you p  
 commonly large fixed co

**Request Spot Ins**  
 **Launch Instances**

Enter the following information:

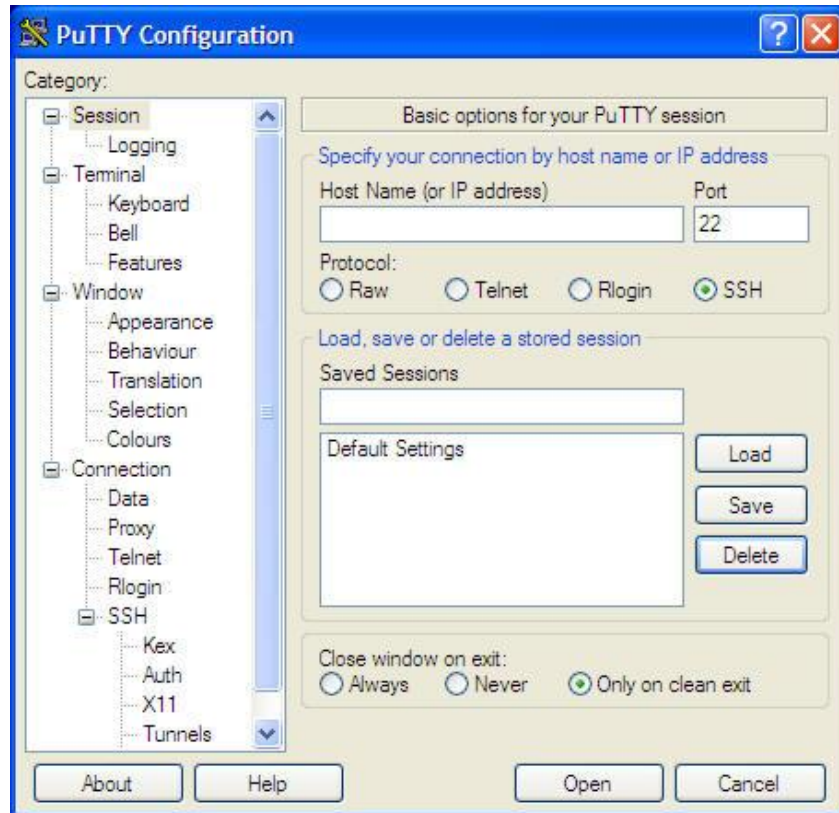
- The number of instances you want to start up
  - 1 for the standard Hands On Lab
- Select the image type you want (Large (m1.large) is what was used at Open World)
- Select your Key Pair

Click the “Launch” button. When you return to the console you will see your instance starting up. Once it is shown to be running you can continue.

# Getting Started

## *Login into the Lab System*

Once your AMI is started you can retrieve the public DNS name from the AWS console. Start PuTTY.



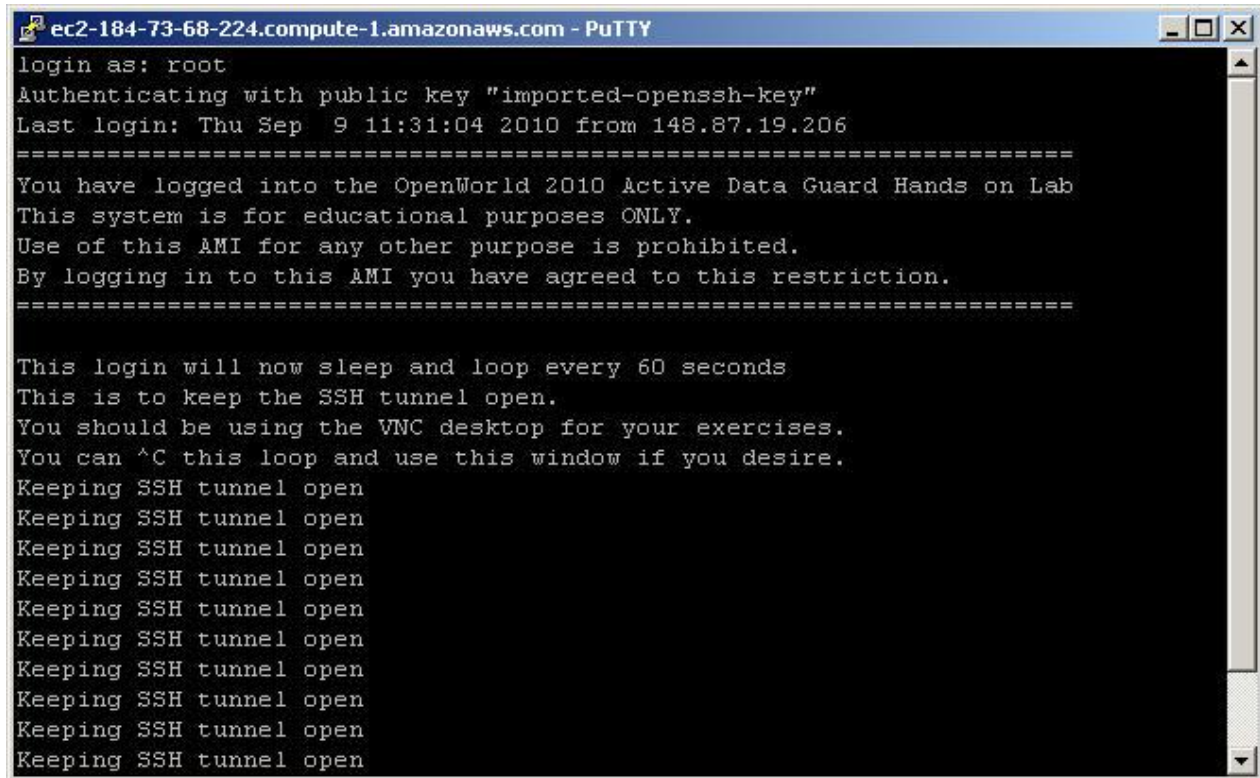
Enter your system name into the 'Host Name (or IP Address)' field and click the '**Open**' button at the bottom.

You will see the following PuTTY security alert.



Press the 'Yes' button to continue.

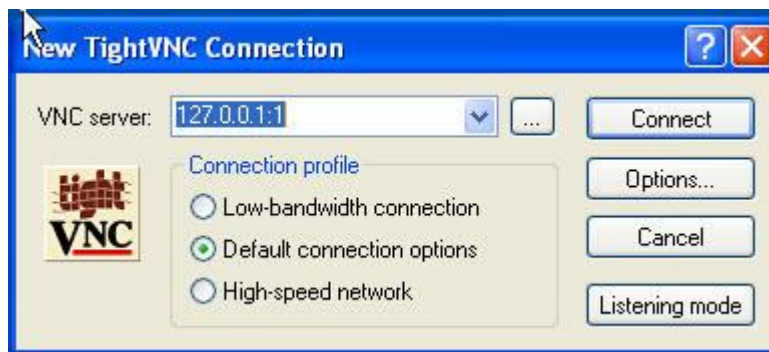
When the login prompt appears in the terminal window enter 'root' and press 'Enter'.



```
ec2-184-73-68-224.compute-1.amazonaws.com - PuTTY
login as: root
Authenticating with public key "imported-openssh-key"
Last login: Thu Sep  9 11:31:04 2010 from 148.87.19.206
=====
You have logged into the OpenWorld 2010 Active Data Guard Hands on Lab
This system is for educational purposes ONLY.
Use of this AMI for any other purpose is prohibited.
By logging in to this AMI you have agreed to this restriction.
=====

This login will now sleep and loop every 60 seconds
This is to keep the SSH tunnel open.
You should be using the VNC desktop for your exercises.
You can ^C this loop and use this window if you desire.
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
Keeping SSH tunnel open
```

When you are logged in you can then start the VNC Viewer by clicking on the "Tight VNC Viewer" icon.



Select (or enter) the system '127.0.0.1:1' and when prompted for the password enter, 'oracle'.



You will use this VNC Desktop for the exercises.

## *Configuring your VNC desktop*

Open 4 terminal windows (right mouse click and select “Open Terminal”).

In one window startup SQL\*Plus using “*sql*”. This will connect you to the current Primary SFO. To change back and forth you can use “*connect sys/oracle@SFO as sysdba*” or “*connect sys/oracle@NYC as sysdba*”. You will execute the SQL commands in this window.

In the 2<sup>nd</sup> window startup DGMGRL using “*dgmgrl sys/oracle@SFO*”. You will execute the DGMGRL commands in this window.

In the remaining two windows tail the alert logs of the two database using “*tasfo*” and “*tanyc*”.

## *Documentation*

The Oracle Database 11g Release 2 documentation can be accessed online at <http://www.oracle.com/pls/db112/homepage>.

## *Environment Scripts and Commands*

To aid in the setup of the various Oracle homes and databases several scripts and aliases have been provided to make your life easier. These are all kept in the ‘oracle’ account’s home directory under the subdirectory ‘setup’.

Opening a terminal window in your VNC desktop will setup all of the aliases and set your environment to the Oracle Database 11g Release 2 home and your Oracle SID to ‘SFO’. To change your environment use the oraenv command and specify SFO or NYC.

- **. oraenv**
  - **Note:** The leading period and space are necessary otherwise oraenv will not change the Oracle environment variables in the terminal session.
  - Enter either SFO (for the Primary) or NYC (for the Standby). The case is important, all uppercase.

The following commands are provided.

- **sql**
  - Starts SQL\*Plus and logs you into the current ORACLE\_SID database as the SYS user by executing *sqlplus "sys/oracle as sysdba"*. This is only provided for this training exercise and is not a recommended practice as the SYS password would be visible to anyone who could look at the aliases used on the system.
- **tanyc**
  - 'tail -f /u01/app/oracle/diag/rdbms/nyc/NYC/trace/alert\_NYC.log'
- **tasfo**
  - 'tail -f /u01/app/oracle/diag/rdbms/sfo/SFO/trace/alert\_SFO.log'

You will find all the files for the exercises in the “**/home/oracle/hol**” directory. These are just text files from which you can cut and paste the commands into DGMGRL or SQL\*Plus.



# Enabling and Using Active Data Guard

Oracle Active Data Guard – “The simplest solution to increase performance by offloading read-only workloads to a synchronized replica of the production database as well as enhance protection from media failures.”

Oracle Active Data Guard is an option for Oracle Database 11g Enterprise Edition. It enhances Quality of Service by offloading resource intensive workloads from your production database to one or more synchronized standby databases. This is accomplished by enabling read-only access to a physical standby database for queries, real-time reporting, web-based access, etc., *while continuously applying changes* received from the production database. Active Data Guard also eliminates the overhead of performing backups on production systems by enabling RMAN block-change tracking and fast incremental backups using a physical standby database and helps your applications avoid media failures by enabling the automatic correction of corrupted blocks on your Production and Active Data Guard standby databases.

An active standby can offload ad-hoc queries, reporting, and fast incremental backups from the primary database, improving performance and scalability while preventing data loss or downtime due to data corruptions, database and site failures, human error, or natural disaster. Active Data Guard is a Database Option for Oracle Enterprise Edition that is separately licensed from your normal Oracle Database 11g Enterprise Edition license. An Active Data Guard license is required when using :

- Real-time Query
- RMAN block-change tracking on a standby database

Active Data Guard is 100% compatible with new Data Guard functionality included with Oracle Database 11g Enterprise Edition.

The following diagram presents a high level overview of Oracle Active Data Guard.



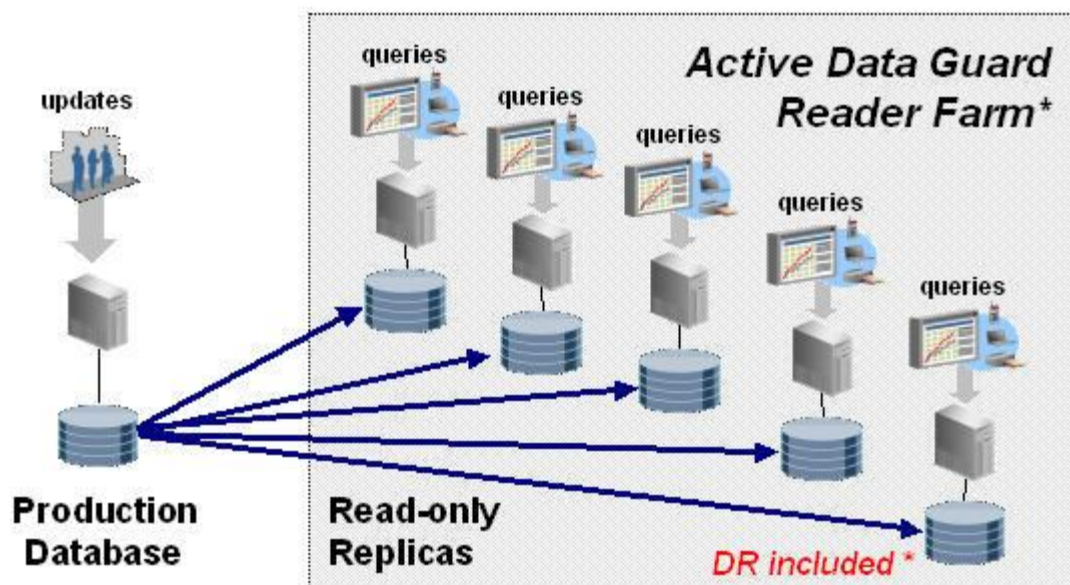
## Active Data Guard Benefits

- **Increase performance of production database:** Offload unpredictable workloads.
- **Simplify operations:** Eliminate traditional replication management complexity.
- **Eliminate compromise:** Reporting replica is up-to-date and online at all times.
- **Reduce cost:** Provides disaster protection and high availability and can serve as a QA system.
- **Reduce backup time:** Complete incremental backups on the standby up to 20x faster.
- **Automatic Block Media Recovery:** Repair corrupted blocks without returning an error to the application

## Active Data Guard Reader Farms

In the example of a Web-business, there is frequently a need to scale out performance to handle catalog queries, order lookup, and other read-only activities that can vary widely depending upon the time of year or other special circumstances that lead to sudden peaks in volume. Active Data Guard is uniquely suited for these situations, because standby databases can easily be provisioned to handle peak periods. A single production database can support direct connections to up to 30 standby databases (nine only prior to Oracle Database 11g Release 2), creating what is referred to as a Reader Farm.

The figure below shows an example of a simple Reader Farm. In addition, because Active Data Guard is compatible with Data Guard functionality - the Reader Farm pictured below has data protection and high availability already built-in. If the production database fails, any of the standby databases in the configuration can quickly transition to the production role - automatically keeping the remaining standby databases synchronized with the latest transactions.



## Enabling Active Data Guard

Enabling a Physical standby database for Active Data Guard is so simple that you may be disappointed that this exercise is so short and so easy. But that is the point, Active Data Guard is easy. To be able to read from your Physical standby database all you have to do is stop Redo Apply, open the Physical standby database Read Only and restart Redo Apply. If you were not using the Broker the steps would be as follows on your Physical standby database.

```
SQL> alter database recover managed standby database cancel;

SQL> alter database open read only;

SQL> alter database recover managed standby database using
      current logfile disconnect;
```

**DO NOT EXECUTE THESE COMMANDS**

Even if you are using the Data Guard Broker you could still do the above commands anyway and the Broker would recognize that the apply has been restarted. But best practices when using the Data Guard Broker is to perform Data Guard changes through the Broker, using DGMGRL or Grid Control (Active Data Guard is configurable from Grid Control 10.2.0.5 onwards). To enable Active Data Guard with the Broker:

```
DGMGRL> connect sys/oracle@NYC
DGMGRL> edit database NYC set state='apply-off';

SQL> connect sys/oracle@NYC
SQL> alter database open read only;

DGMGRL> connect sys/oracle@NYC
DGMGRL> edit database NYC set state='apply-on';
```

**DO NOT EXECUTE THESE COMMANDS**

After this your Physical standby is open and read for readers all the while new redo is being applied. However, as of Oracle Database 11g Release 2 the Broker will take care of the stopping and restarting of Redo Apply for you whenever you execute an open command on a Physical Standby that is under Broker control.

```
SQL> connect sys/oracle@nyc as sysdba;
Connected.
SQL> alter database open;

Database altered.

SQL>
```

### TASK:

- **Execute the SQL\*Plus commands from “/home/oracle/hol/EnableADG”**

If you monitor the alert log from the standby NYC you will see that when the open is executed, Redo Apply is canceled, the database opened in Read Only mode and Redo Apply restarted automatically. You can also verify that the standby is open using Active Data Guard by executing a DGMGRL “*show database nyc*” command.

```
DGMGRL> show database nyc;

Database - nyc

Role:                PHYSICAL STANDBY
Intended State:      APPLY-ON
Transport Lag:       0 seconds
Apply Lag:           0 seconds
Real Time Query:     ON
Instance(s):
  NYC

Database Status:
SUCCESS
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/VerifyADG”**

***Reading from an Active Data Guard Standby Database***

To show that you can read the data on the Physical Standby database as it is being applied, you can add some information to a table in the Primary database and read that data from the Physical Standby database. To begin, log into the Physical standby NYC as HR (password oracle) and read the HR.REGIONS table.

```
SQL> connect hr/oracle@nyc
Connected.
SQL> select * from hr.regions;

  REGION_ID REGION_NAME
-----
1 Europe
2 Americas
3 Asia
4 Middle East and Africa

SQL>
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/ReadHRADG”**

On the Primary database SFO, connect as HR and add a row to the HR.REGIONS table.

```
SQL> connect hr/oracle@sfo
Connected.
SQL> insert into HR.REGIONS values (30,'OpenWorld');

1 row created.

SQL> commit;

Commit complete.

SQL>
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/InsertHR”**

To verify that the data has been applied at the standby and is visible return to your physical standby NYC and rerun the select command.

```
SQL> connect hr/oracle@nyc
Connected.
SQL> select * from hr.regions;

  REGION_ID REGION_NAME
-----
           1 Europe
           2 Americas
           3 Asia
           4 Middle East and Africa
          30 OpenWorld

SQL>
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/ReadHRADG”**

You have now used Active Data Guard to read up-to-date data from your Physical standby database.

## Managing Potential Apply Lags

When you use real-time query to offload queries from a primary database to a physical standby database, you may want to monitor the apply lag to ensure that it is within acceptable limits. The current apply lag 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. This metric is computed to the nearest second. To obtain the apply lag, query the V\$DATAGUARD\_STATS view.

```
SQL> connect sys/oracle@NYC as sysdba
Connected.
SQL> SELECT name, value, datum_time, time_computed
       2 FROM V$DATAGUARD_STATS WHERE name like 'apply lag';
```

NAME	VALUE	DATUM_TIME	TIME_COMPUTED
apply lag	+00 00:00:00	08/05/2010 13:14:11	08/05/2010 13:14:11

### TASK:

- Execute the SQL\*Plus commands from “/home/oracle/hol/CheckADGLag”

The apply lag metric is computed using data that is periodically received from the primary database.

- DATUM\_TIME contains a timestamp of when this data was last received by the standby database.
- TIME\_COMPUTED contains a timestamp taken when the apply lag metric was calculated.

The difference between the values in these columns should be less than 30 seconds. If the difference is larger than this, the apply lag metric may not be accurate.

To obtain a histogram that shows the history of apply lag values since the standby instance was last started, query the V\$STANDBY\_EVENT\_HISTOGRAM view.

```
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	48612	08/05/2010 13:20:02
apply lag	1 seconds	102	08/05/2010 13:15:09
apply lag	2 seconds	16	08/05/2010 12:20:58
apply lag	3 seconds	4	08/05/2010 11:15:56

### TASK:

- Execute the SQL\*Plus commands from “/home/oracle/hol/ADGHistogram”

These two system views are useful for monitoring how up to date the data in an Active Data Guard standby is compared to the data in the Primary database. The problem is how do your applications know how recent is the data they are retrieving from an Active Data Guard standby when they may not have access to these views directly? Active Data Guard now has a new session parameter called *STANDBY\_MAX\_DATA\_DELAY* that can be used to specify a session-specific apply lag tolerance, measured in seconds, for queries issued by non-administrative users to a physical standby database that is in real-time query mode. This capability allows queries to be safely offloaded from the primary database to a physical standby database, because it is possible to detect if the standby database has become unacceptably stale.

You would use *ALTER SESSION* to set *STANDBY\_MAX\_DATA\_DELAY* to a value of your choosing based on the service level agreement (SLA) of the particular user. The rules for *STANDBY\_MAX\_DATA\_DELAY* are as follows.

- If set to the default value of NONE, queries issued to a physical standby database will be executed regardless of the apply lag on that database.
- If set to a non-zero value, a query issued to a physical standby database will be executed only if the apply lag is less than or equal to *STANDBY\_MAX\_DATA\_DELAY*.
  - Otherwise, an ORA-3172 error is returned to alert the client that the apply lag is too large.
- If set to 0, a query issued to a physical standby database is guaranteed to return the exact same result as if the query were issued on the primary database
  - If the standby database is lagging behind the primary database an ORA-3172 error is returned.
  - **Note:** Using zero has three requirements which if not met returns the ORA-3172 immediately:
    1. The configuration must be in Maximum Availability or Maximum Protection
    2. The Active Data Guard standby must be receiving the redo using SYNC.
    3. Redo Apply must be active.

To show this functionality set an SLA of 2 seconds, which means that if the data returned from the Active Data Guard standby is more than 2 seconds behind the Primary return an error rather than the stale data.

```
SQL> connect hr/oracle@nyc
Connected.
SQL> ALTER SESSION SET STANDBY_MAX_DATA_DELAY=2;

Session altered.

SQL> select * from regions;

  REGION_ID REGION_NAME
-----
          1 Europe
          2 Americas
          3 Asia
          4 Middle East and Africa
         30 OpenWorld

5 rows selected.
```

### TASK:

- **Execute the SQL\*Plus commands from “/home/oracle/hol/SetADGSLA”**

To cause an artificial delay you can stop redo apply on your Standby NYC. Return to your DGMGRL window and change the 'state' of NYC to 'APPLY-OFF' which will cancel Redo Apply.

```
DGMGRL> edit database NYC set state='apply-off';
Succeeded.
```

**TASK:**

- **Execute the DGMGRL command from “/home/oracle/hol/ApplyOff”**

Now return to your SQL\*Plus window on NYC and re-execute the previous select statement again.

```
SQL> select * from regions;
select * from regions
*
ERROR at line 1:
ORA-03172: STANDBY MAX DATA DELAY of 2 seconds exceeded
```

**TASK:**

- **Execute the SQL\*Plus command from “/home/oracle/hol/ReadHRADG”**

To remove the artificial delay restart redo apply on your Standby NYC. Return to your DGMGRL window and change the 'state' of NYC to 'APPLY-ON'.

```
DGMGRL> edit database NYC set state='apply-on';
Succeeded.
```

**TASK:**

- **Execute the DGMGRL command from “/home/oracle/hol/ApplyOn”**

Return to your SQL\*Plus window on NYC and re-execute the previous select statement for the 3<sup>rd</sup> time.

```
SQL> select * from regions;

REGION_ID REGION_NAME
-----
1 Europe
2 Americas
3 Asia
4 Middle East and Africa
30 OpenWorld

5 rows selected.
```

**TASK:**

- **Execute the SQL\*Plus command from “/home/oracle/hol/ReadHRADG”**

From these simple examples you can see how your applications can control the amount of lag between what they see in the Standby and where the Primary database actually is at every time they perform a query.

It is also possible for your application to be configured in a manner where it will not begin to allow queries to occur until the standby is in sync with the Primary. To do this the application can execute a new session statement “ALTER SESSION SYNC WITH PRIMARY”.

This statement will block until all redo data received by the standby database at the time that this command is issued has been applied to the physical standby database. Note that an ORA-3173 error is returned immediately, and synchronization will not occur, if the redo transport status at the standby database is not SYNCHRONIZED or if Redo Apply is not active. Using this command has the same three requirements as setting the session parameter *STANDBY\_MAX\_DATA\_DELAY* to zero.

You can ensure that Redo Apply synchronization occurs in specific cases by using a logon trigger which contains the `SYS_CONTEXT('USERENV','DATABASE_ROLE')` function to create a standby-only trigger. For example, you could create the following trigger on the Primary database (which would be sent to the standby databases and applied by Redo Apply) that would execute the *ALTER SESSION SYNC WITH PRIMARY* statement for a specific user connection at logon.

```
SQL> connect sys/oracle@sfo as sysdba
Connected.
SQL> CREATE OR REPLACE TRIGGER hr_logon_sync_trigger
  2 AFTER LOGON ON hr.schema
  3 BEGIN
  4   IF (SYS_CONTEXT('USERENV','DATABASE_ROLE')
  5       IN ('PHYSICAL STANDBY'))
  6   THEN
  7     execute immediate 'alter session sync with primary';
  8   END IF;
  9   END;
 10  /

Trigger created.
```

**DO NOT EXECUTE THESE COMMANDS**

Now when you log into NYC as the user HR the trigger would attempt to synchronize the session with the Primary, and in our case, return an error and disallow the logon since we do not meet the 3 requirements.

```
SQL> connect hr/oracle@nyc
ERROR:
ORA-00604: error occurred at recursive SQL level 1
ORA-03173: Standby may not be synced with primary
ORA-06512: at line 5

SQL> select * from regions;
SP2-0640: Not connected
```

**DO NOT EXECUTE THESE COMMANDS**

**Note: There is no TASK for this exercise as you would no longer be allowed to login as HR!**

## *Writing data with an Active Data Guard standby*

When connected to an Active Data Guard standby database, read-only applications can perform:

- Selects
- Alter session / system
- Set role
- Lock table
- Call stored procedures
- DBlinks to write to remote databases
- Stored procedures to call remote procedures via DBlinks
- SET TRANSACTION READ ONLY for transaction level read consistency
- Complex queries e.g. grouping set queries and with clause queries

If an application has the capability of directing read operations to the Active Data Guard service running on the Physical standby database and write operations to the Primary database then all you need to do is define the services to start on the correct database and the application will automatically offload the read workload to the Physical standby database.

However, as of Oracle Database 11g Release 1 with Active Data Guard, write operations can be performed on a Physical Standby that is opened read only, through the use of Database Links. The links are created on the Primary database and are applied to the Physical Standby through Redo Apply. They can point back to the Primary or to another database where the application has read write access. In this manner a primarily read application that has a need to make a few writes can make those writes on the standby and they will be automatically redirected to the read write database. If the applications require access to the data that is updated on the Active Data Guard standby then your database link must point to the Primary database so that the data is available to all users.

To begin with log into the Primary SFO as SYS ( AS SYSDBA) again and create a database link that uses 'sfo' as the connect string. The redo generated by this DDL will be sent to the physical standby where Redo Apply will apply the redo making the database link available to the application running on the physical standby.

```
SQL> connect sys/oracle@sfo as sysdba;
Connected.
SQL> create public database link write_ADG connect to HR
      2 identified by oracle using 'SFO';

Database link created.

SQL>
```

### **TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/CreateDBLink”**

Now, connecting back to the standby, select the rows in the HR.REGIONS table again where the data will be displayed as it appears in the standby.

```
SQL> connect hr/oracle@nyc
Connected.
SQL> select * from regions;

REGION_ID REGION_NAME
-----
          1 Europe
          2 Americas
          3 Asia
          4 Middle East and Africa
         30 OpenWorld
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/ReadHRADG”**

This time we will insert a new row into the HR.REGIONS table using the database link while connected to the Physical standby. The actual DML will be redirected to the Primary database and once the *COMMIT* is executed redo will be generated and sent to the Physical standby. Redo Apply will apply the redo making the new data available to the application.

```
SQL> connect hr/oracle@nyc
Connected.
SQL> insert into REGIONS@write_ADG values (99, 'Data Guard HOL');

1 row created.

SQL> commit;

Commit complete.
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/InsertHRADG”**

Finally, select the rows in the HR.REGIONS table again while connected to the Physical standby. The new data will be displayed as it now appears in the standby.

```
SQL> connect hr/oracle@nyc
Connected.
SQL> select * from regions;

REGION_ID REGION_NAME
-----
          1 Europe
          2 Americas
          3 Asia
          4 Middle East and Africa
         30 OpenWorld
         99 Data Guard HOL

6 rows selected.
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/ReadHRADG”**

How fast the data appears in the Physical standby depends on the workload on your Primary database, how fast the redo can be sent to the Physical standby and how well Redo Apply is performing. You can tune these factors by following the MAA papers for tuning Redo Transport and Apply. Refer to the Resources section at the end of this book for the links to the MAA papers.

## Schema Redirecting and Active Data Guard

You do have to be careful when setting up your database links and synonyms as it could be a performance problem if normal read or write access to a table on the Primary goes out over the network and back again, or a write on the standby goes over the link to the Primary but so do the reads, which defeats the purpose of having an Active Data Guard standby database. It is now possible to use schema redirection to make it easier to code applications so that reads are always local and writes only go out over the database link when you are connected to an Active Data Guard standby.

This method does require changing your application to use different names for reading and writing to the tables so that the reads can be done locally and the write redirected only when the application attaches to the Active Data Guard standby.

To do this you will have to setup new users/schemas which your application will be switched to with a login trigger whenever they connect to an Active Data Guard standby database. Then by using synonyms and database links reads can be local and writes are transparently redirected to the Primary. But, if the application connects to the Primary database you do not want it using any of the database links, for reads or writes. So your trigger will have to ensure, like the previous logon trigger we discussed, that the schema is only redirected when the application logs into the standby. **Read the following examples but do not execute the code until you get to the TASK and then make sure you execute the code as the correct USER and in the correct ORDER!**

```
SQL> connect sys/oracle@sfo as sysdba
Connected.

SQL> CREATE OR REPLACE TRIGGER hr_logon_switch_schema_trigger
 2 AFTER LOGON ON hr.schema
 3 BEGIN
 4   IF (SYS_CONTEXT('USERENV','DATABASE_ROLE')
 5       IN ('PHYSICAL STANDBY'))
 6   THEN
 7     execute immediate
 8       'alter session set current schema = hr syn';
 9   END IF;
10 END;
11 /
```

Now when your application logs into *HR* on the Primary you will be running in the *HR* schema but when it logs into *HR* on the standby your context will be switched to the *HR\_SYN* schema.

```
SQL> connect hr/oracle@sfo
SQL> select sys_context('USERENV','CURRENT_SCHEMA') from dual;

HR

SQL> connect hr/oracle@nyc
SQL> select sys_context('USERENV','CURRENT_SCHEMA') from dual;

HR SYN
```

That takes care of the schema redirect. But now you need to create a user, grant the necessary privileges, and create appropriate synonyms in the schemas that your application will use when reading or writing to the tables. In our case we will do this for the *HR\_SYN* and *HR* and only for the *HR.REGIONS* table.

```
SQL> connect sys/oracle@sfo as sysdba
Connected.
SQL> create user hr_syn identified by oracle account unlock;

User created.

SQL> grant connect, resource, create session, create synonym to hr_syn;

Grant succeeded.

SQL> create public database link hr_primary using 'sfo';

Database link created.
```

**TASK:**

- **Execute just the *SYS* user SQL\*Plus commands from “/home/oracle/hol/WriteRedirect”**

The next step is to allow our new user to create synonyms in the *HR* schema and create the same two synonyms that the application will be changed to use regardless of which database they are using.

```
SQL> connect hr/oracle@sfo
Connected.

SQL> grant all on regions to hr_syn;
Grant succeeded.

SQL> create synonym r_regions for hr.regions;
Synonym created.

SQL> create synonym w_regions for hr.regions;
Synonym created.
```

**TASK:**

- **Execute just the *HR* user SQL\*Plus commands from “/home/oracle/hol/WriteRedirect”**

If these two synonyms are not created for the normal *HR* user, your modified application will not be able to function if it has to be attached to the Primary database. You will see in a moment that when you read and write data with your ‘application’ you will be modifying the ‘application’ (our SQL\*Plus queries) to use these two synonyms. But, remember, the trigger you defined will point your application to the *HR\_SYN* schema when it logs into the standby, and the synonyms would be visible. But when your application logs into the Primary the schema ‘redirect’ would not take place and these synonyms would not be available unless you define them ahead

of time. In this case the reads and writes will take place on the *REGIONS* table directly since you would be attached to the Primary database.

Now when our application connects to the Primary database, all activity with the *HR.REGIONS* table will be local to the Primary. The last step is to do the same for the *HR\_SYN* schema. But in this case any reads that are directed to the *r\_regions* synonym will be read from the standby while any updates to the *w\_regions* synonym will automatically be sent to the Primary database using our database link.

```
SQL> connect hr_syn/oracle@sfo
Connected.

SQL> create synonym r_regions for hr.regions;
Synonym created.

SQL> create synonym w_regions for hr.regions@hr_primary;
Synonym created.
```

### TASK:

- **Execute just the *HR\_SYN* user SQL\*Plus commands from “/home/oracle/hol/WriteRedirect”**

Finally we are ready to point our application (which we have changed to use our new synonyms) at our Active Data Guard standby and begin reading and writing data. Reconnect to the *HR* user on your standby and try to read the *REGIONS* table again. You will see that it is no longer visible because your default schema is now *HR\_SYN* because of the logon trigger. But if you use the synonym you created, *r\_regions*, you will see the *REGIONS* data directly from the standby database. **Note that the next two code examples are both contained in the TASK file listed after the second example.**

```
SQL> connect hr/oracle@nyc;
Connected.

SQL> select * from regions;
select * from regions
          *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> select * from r_regions where region_name='OpenWorld';

  REGION_ID REGION_NAME
-----
         10 OpenWorld

SQL> update r_regions set region_id=88 where region_name='OpenWorld';
update r_regions set region_id=88 where region_name='OpenWorld'
          *
ERROR at line 1:
ORA-16000: database open for read-only access
```

Trying to update that data though ends up with an ORA-16000 error as you can see above.

When you retry the update command using the 'write' synonym *w\_regions*, you will be able to update the data because the DML is actually being transparently sent to the Primary database via the database link and will be visible in the standby once the redo has been transmitted and applied to the physical standby.

```
SQL> update w_regions set region_id=88 where region_name='OpenWorld';

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from r_regions where region_name='OpenWorld';

  REGION_ID REGION_NAME
-----
         88 OpenWorld
```

### TASK:

- **Execute the SQL\*Plus commands from “/home/oracle/hol/WriteADGRedirect”**

While this process requires some setup and configuration and modifications to your application, it does provide a method to make an application that requires some updating capability function normally even when attached to a Read Only Active Data Guard standby database.

But what happens if your 'application' connects directly to the Primary? Or perhaps you did a failover or switchover and what was your Active Data Guard standby is now the Primary database. How will your application function now? It will work as advertised and not use any of the database links you have defined as it will no longer be switched to the *HR\_SYN* schema. Logging back into the current primary you would see the following.

```
SQL> connect hr/oracle@sfo
Connected.

SQL> select sys_context('USERENV','CURRENT_SCHEMA') "Who Am I?"
       2 from dual;

Who Am I?
-----
HR
```

### TASK:

- **Execute the SQL\*Plus commands from “/home/oracle/hol/WriteNoRedirectCheck”**

So now when we rerun all of the 'application' commands you will see somewhat different results as shown below now that we are connected to the Primary database.

```
SQL> connect hr/oracle@sfo
Connected.

SQL> select * from regions where region_name='OpenWorld';

  REGION_ID REGION_NAME
-----
         88 OpenWorld

SQL> select * from r_regions where region_name='OpenWorld';

  REGION_ID REGION_NAME
-----
         88 OpenWorld

SQL> update r_regions set region_id=77 where region_name='OpenWorld';
1 row updated.

SQL> commit;

Commit complete.

SQL> select * from regions where region_name='OpenWorld';

  REGION_ID REGION_NAME
-----
         77 OpenWorld

SQL> update w_regions set region_id=66 where region_name='OpenWorld';
1 row updated.

SQL> commit;

Commit complete.

SQL> select * from regions where region_name='OpenWorld';

  REGION_ID REGION_NAME
-----
         66 OpenWorld
```

Not only does the table *REGIONS* exist again but you can read AND write to *R\_REGIONS* as well as *W\_REGIONS*. You can do this because you are connected to a Read Write database in the *HR* schema.

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/WriteNoRedirect”**

## *Avoiding Media Corruption with Active Data Guard*

Oracle Database 11g Release 2 now has the capability to automatically repair corrupt data blocks in your production database as soon as the corruption is detected by using your Active Data Guard standby database to retrieve good copies of the corrupted blocks. Automatic Block Media Recovery will also automatically repair corrupted blocks that are discovered in your physical standby databases. This feature reduces the amount of time that data is inaccessible due to block corruption and will avoid returning errors to your application.

This reduces block recovery time by using up-to-date good blocks in real-time, as opposed to retrieving blocks from disk or tape backups, or from Flashback logs. The process is as follows:

- If a corrupt data block is discovered on a *Primary* database.
  - If you have a physical standby database open read only with Active Data Guard then the standby can be used to repair corrupt data blocks in the primary database.
    - If possible, any corrupt data block encountered when a primary database is accessed is automatically replaced with an uncorrupted copy of that block from the Active Data Guard physical standby database.
    - An *ORA-1578* error is returned when automatic repair is not possible which would be the case if the corrupted block was the header record for example.
- If a corrupt data block is discovered on a *Physical standby* database.
  - The server attempts to automatically repair the corruption by obtaining a copy of the block from the primary database. This requires that the following database initialization parameters are configured on the standby database.
    - LOG\_ARCHIVE\_CONFIG parameter with a DG\_CONFIG list
    - LOG\_ARCHIVE\_DEST\_n parameter for the primary database
  - In any standby database configuration these parameters should always be defined. Since your configuration is being managed by the Broker these parameters are defined.

You can also manually repair a corrupted data block by using the *RMAN RECOVER BLOCK* command. This command searches several locations for an uncorrupted copy of the data block. By default, one of the locations is any available physical standby database that is open with Active Data Guard.

```
ERROR at line 1:
ORA-01578: ORACLE data block corrupted (file # 5, block # 139)
ORA-01110: data file 5:
'/hol/oracle/oradata/SFO/datafile/o1_mf_example_5bmrk40x_.dbf'
```

**DO NOT EXECUTE THESE COMMANDS**

You could then use RMAN to recover just that one block.

```
RMAN> recover datafile '.../o1_mf_example_5bmrk40x_.dbf' block 139;
Starting recover at 05-Aug-10
using target database control file instead of recovery catalog
finished standby search, restored 1 blocks
...
Finished recover at 05-Aug-10
```

**DO NOT EXECUTE THESE COMMANDS**

To show how this feature works we need to close our physical standby so that Automatic Block Media Recovery cannot happen at first so that your 'application' does get the error because it cannot automatically recover the block.

```
SQL> connect sys/oracle@nyc as sysdba
Connected.
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.
...
Database mounted.
```

**TASK:**

- **Execute the SQL\*Plus commands from “/home/oracle/hol/CloseADG”**

As we are a Broker controlled configuration, Redo Apply will automatically be restarted after the mount finishes. At this point you will corrupt a block in the *REGIONS* table of the Primary database. First shutdown the Primary SFO and then copy the first data block (skipping the header block) of the *REGIONS* table over the next block in the table. (We know where the *REGIONS* table is by looking at *DBA\_SEGMENTS*).

```
SQL> connect sys/oracle@sfo as sysdba
...
SQL> shutdown immediate
...
ORACLE instance shut down.

SQL>

bash-3.1$ dd if=/u01/app/oradata/SFO/example01.dbf
of=/u01/app/oradata/SFO/example01.dbf count=1 bs=8192 skip=141
seek=140 conv=notrunc
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 9.7e-05 seconds, 84.5 MB/s

SQL> connect sys/oracle@sfo as sysdba
...
SQL> startup
...
ORACLE instance started.
```

**TASK:**

- **Execute the SQL\*Plus and Linux commands from “/home/oracle/hol/CorruptRegions”**

That will have corrupted the 2<sup>nd</sup> block in the *REGIONS* table so when you go back to the Primary SFO and try to select the rows from the *REGIONS* table as the *HR* user, you will get an error since the Standby is not open in Active Data Guard at the moment.

```
SQL> connect hr/oracle@sfo
Connected.
SQL> select * from hr.regions;
select * from hr.regions
          *
ERROR at line 1:
ORA-01578: ORACLE data block corrupted (file # 5, block # 140)
ORA-01110: data file 5: '/u01/app/oradata/SFO/example01.dbf '
```

**TASK:**

- **Execute the SQL\*Plus command from “/home/oracle/hol/ReadHRPrimary”**

Leaving your Primary SQL\*Plus session at this point, open a new terminal window, start SQL\*Plus and connect to the standby database and reopen it in Active Data Guard.

```
SQL> connect sys/oracle@nyc as sysdba;
Connected.
SQL> alter database open;
Database altered.
```

**TASK:**

- **Execute the SQL\*Plus command from “/home/oracle/hol/EnableADG”**

At this point the corrupted block on the Primary database has not been repaired because it is not being accessed at present. But we can read the *REGIONS* table in the standby database without any errors.

```
SQL> connect hr/oracle@nyc
Connected.
SQL> select * from hr.regions;

  REGION_ID REGION_NAME
-----
           1 Europe
           2 Americas
           3 Asia
           4 Middle East and Africa
          66 OpenWorld
          99 Data Guard HOL
```

**TASK:**

- **Execute the SQL\*Plus command from “/home/oracle/hol/ReadHRADG”**

You can read the *REGIONS* table in the standby without any errors because Data Guard does NOT propagate these kinds of media failures to its standby databases.

Examine the alert log of your standby NYC and make sure that Redo Apply has restarted before continuing.

```
Completed: ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
THROUGH ALL SWITCHOVER DISCONNECT USING CURRENT LOGFILE
```

**DO NOT EXECUTE THESE COMMANDS**

Return to your Primary database terminal window where you originally executed the select command on *REGIONS* and got the error. Re-execute the command again and you will see that the read will succeed.

```
SQL> /

REGION_ID REGION_NAME
-----
          1 Europe
          2 Americas
          3 Asia
          4 Middle East and Africa
         66 OpenWorld
          99 Data Guard HOL

6 rows selected.
```

### TASK:

- **Re-Execute the SQL\*Plus SELECT command from “/home/oracle/hol/ReadHRPrimary”**

The error is not returned the 2<sup>nd</sup> time because Automatic BMR was initiated in the background to recover the corrupted block with a good copy from the Active Data Guard Standby. You can see this happening by examining the alert log of the Primary SFO.

```
Thu Aug 05 11:30:56 2010
ABMR started with pid=37, OS id=11994
Auto BMR service is active.
Requesting Auto BMR for (file# 5, block# 140)
Waiting Auto BMR response for (file# 5, block# 140)
Auto BMR successful
WARNING: AutoBMR fixed mismatched on-disk block 140008d with in-
mem rdba 140008c.
```

Transparent to your ‘application’ the corrupted block was fixed and the results of your query are returned. If the physical standby had already been opened in Active Data Guard when you first executed the select statement you would never have seen the *ORA-01578* error as the corrupted block would have been fixed immediately.



# Conclusion

This concludes the exercises for the Oracle Database 11g Active Data Guard Hands On Lab. During this session you have explored and exercised Active Data Guard's unique capabilities while protecting your data. You were able to read and write the Standby Database while changes are being applied from the Production database.

You have done the following exercises.

- Enabled Active Data Guard
  - Read live data from your Physical Standby database
  - Wrote data through your Active Data Guard standby
  - Used Schema redirection to write data through your Active Data Guard standby
  - Corrupted data blocks and watched Automatic Block Media Recovery in action

It is our sincere hope that these exercises have helped to guide you on your path to using Data Guard and understanding how to use the features that it brings to the Oracle Database world to provide you with Disaster Protection for your Oracle data.

Thank you for your participation.

Your Data Guard Staff.



# Resources

## The following documentation is available for Data Guard

- Oracle® Data Guard Concepts and Administration 11g Release 1 (11.1)
  - [http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28294/toc.htm](http://download.oracle.com/docs/cd/B28359_01/server.111/b28294/toc.htm)
- Oracle® Data Guard Broker 11g Release 1 (11.1)
  - [http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28295/toc.htm](http://download.oracle.com/docs/cd/B28359_01/server.111/b28295/toc.htm)
- Oracle® Data Guard Concepts and Administration 11g Release 2 (11.2)
  - [http://download.oracle.com/docs/cd/E11882\\_01/server.112/e10700/toc.htm](http://download.oracle.com/docs/cd/E11882_01/server.112/e10700/toc.htm)
- Oracle® Data Guard Broker 11g Release 2 (11.2)
  - [http://download.oracle.com/docs/cd/E11882\\_01/server.112/e10702/toc.htm](http://download.oracle.com/docs/cd/E11882_01/server.112/e10702/toc.htm)

## The Maximum Availability Architecture Best Practices papers are on Oracle OTN.

- [Best Practices for High Availability -- Maximum Availability Architecture \(MAA\)](http://www.oracle.com/technetwork/database/features/availability/maa-096107.html)
  - <http://www.oracle.com/technetwork/database/features/availability/maa-096107.html>



# Appendix A – Command Files

These are the files on the lab system that contain to commands you will be executing during the exercises.

## ADGHistogram

```
connect sys/oracle@NYC as sysdba
SELECT * FROM V$STANDBY_EVENT_HISTOGRAM WHERE NAME = 'apply lag'
AND COUNT > 0;
```

## ApplyOff

```
edit database NYC set state='apply-off';
```

## ApplyOn

```
edit database NYC set state='apply-on';
```

## CheckADGLag

```
connect sys/oracle@NYC as sysdba
SELECT name, value, datum_time, time_computed FROM V$DATAGUARD_STATS
WHERE name like 'apply lag';
```

## CloseADG

```
connect sys/oracle@nyc as sysdba
shutdown immediate
startup mount
```

## CorruptRegions

*# SQL\*Plus Command – Execute in your Sqlplus window*

```
connect sys/oracle@sfo as sysdba
shutdown immediate
```

*# Linux Command – Execute in a Linux window*

```
dd if=/u01/app/oradata/SFO/example01.dbf of=/u01/app/oradata/SFO/example01.dbf count=1 bs=8192
skip=141 seek=140 conv=notrunc
```

*# SQL\*Plus Command – Execute in your Sqlplus window*

```
connect sys/oracle@sfo as sysdba
startup
```

**CreateDBLink**

```
connect sys/oracle@sfo as sysdba
create public database link write_ADG connect to HR identified by oracle using 'SFO';
```

**EnableADG**

*# Do these commands in a new terminal window*

```
sqlplus sys/oracle@nyc as sysdba;
alter database open;
```

**InsertHR**

```
connect hr/oracle@sfo
insert into HR.REGIONS values (30,'OpenWorld');
commit;
```

**InsertHRADG**

```
connect hr/oracle@nyc
insert into REGIONS@write_ADG values (99, 'Data Guard HOL');
commit;
```

**ReadHRADG**

```
connect hr/oracle@nyc
select * from hr.regions;
```

**ReadHRPrimary**

```
connect hr/oracle@sfo
select * from hr.regions;
```

**SetADGSLA**

```
connect hr/oracle@nyc
ALTER SESSION SET STANDBY_MAX_DATA_DELAY=2;
select * from regions;
```

**VerifyADG**

```
show database nyc;
```

**WriteADGRedirect**

```
connect hr/oracle@nyc
select * from regions where region_name='OpenWorld';
select * from r_regions where region_name='OpenWorld';
update r_regions set region_id=88 where region_name='OpenWorld';
update w_regions set region_id=88 where region_name='OpenWorld';
commit;
select * from r_regions where region_name='OpenWorld';
```

**WriteNoRedirectCheck**

```
connect hr/oracle@sfo
select sys_context('USERENV','CURRENT_SCHEMA') "Who Am I?" from dual;
```

**WriteNoRedirect**

```
connect hr/oracle@sfo
select * from regions where region_name='OpenWorld';
select * from r_regions where region_name='OpenWorld';
update r_regions set region_id=77 where region_name='OpenWorld';
commit;
select * from regions where region_name='OpenWorld';
update w_regions set region_id=66 where region_name='OpenWorld';
commit;
select * from regions where region_name='OpenWorld';
```

**WriteRedirect***SYS user Commands*

```
connect sys/oracle@sfo as sysdba
```

```
CREATE OR REPLACE TRIGGER hr_logon_switch_schema_trigger
AFTER LOGON ON hr.schema
BEGIN
  IF (SYS_CONTEXT('USERENV','DATABASE_ROLE')
      IN ('PHYSICAL STANDBY'))
  THEN
    execute immediate
      'alter session set current_schema = hr_syn';
  END IF;
END;
/
```

```
create user hr_syn identified by oracle account unlock;
grant connect, resource, create session, create synonym to hr_syn;
create public database link hr_primary using 'sfo';
```

*HR user Commands*

```
connect hr/oracle@sfo
grant all on regions to hr_syn;
create synonym r_regions for hr.regions;
create synonym w_regions for hr.regions;
```

*HR\_SYN user Commands*

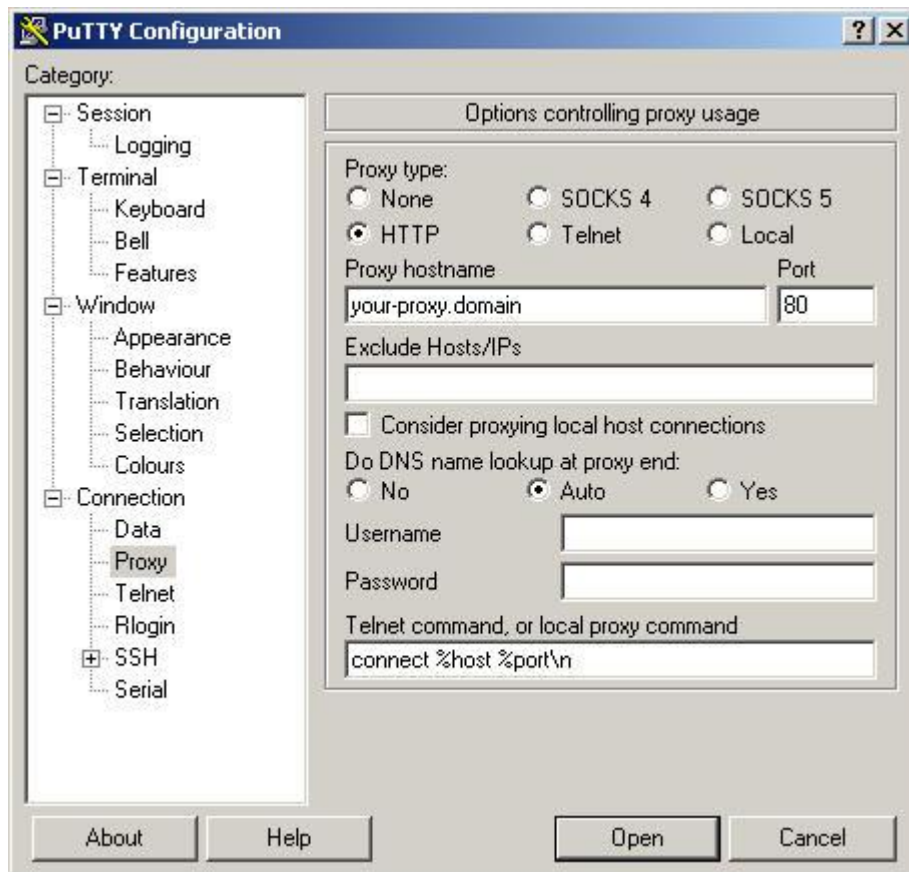
```
connect hr_syn/oracle@sfo
create synonym r_regions for hr.regions;
create synonym w_regions for hr.regions@hr_primary;
```

# Appendix B – Setting up PuTTY to connect to your Cloud Instance

In this appendix you will find the necessary steps to configure your PuTTY session so that your connection will work behind a Corporate Firewall and provide access to the VNC server running on your Amazon Image. Once you have completed the steps to install and generate your ‘key’ from the Amazon manual you can then add these extra modifications.

## *Running from Behind a Corporate Firewall*

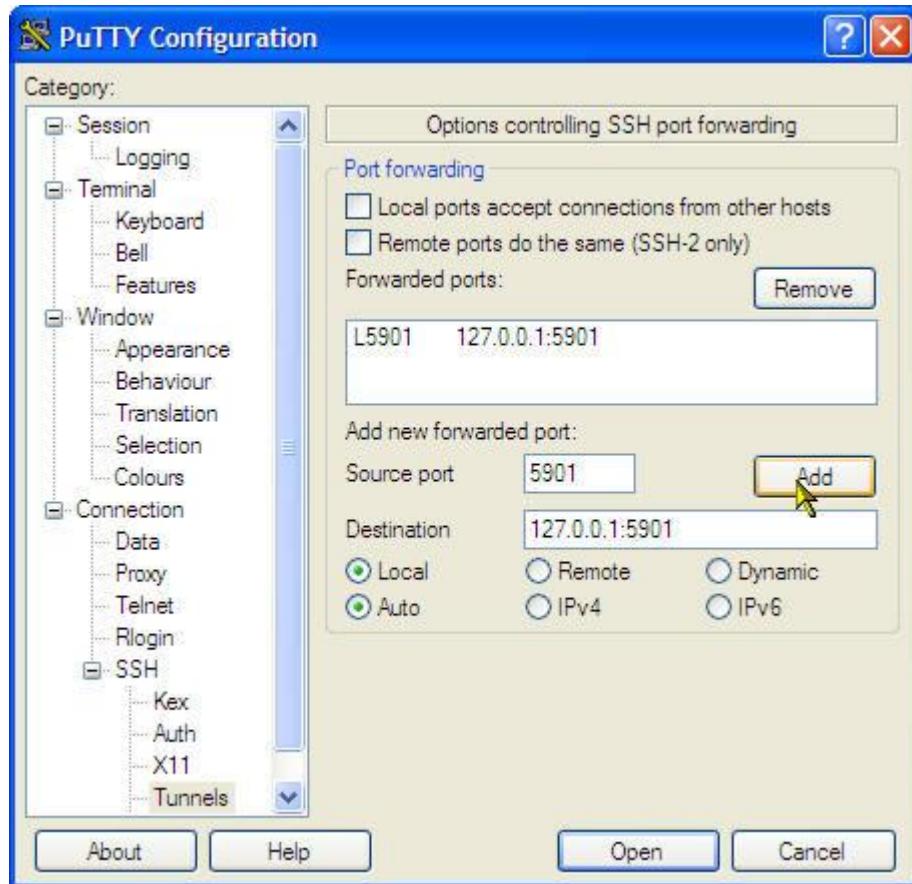
Click on ‘Proxy’ under ‘Connection’ on the left side and select the ‘HTTP’ radio button and put your proxy server in the Proxy Hostname field using your port number. In our example we have ‘your-proxy.domain’ on port 80.



## Configuring access to the VNC Server

On your instance there is a VNC server started that you can use to perform the exercises in this book. That server is running on port 5901 with a password of 'oracle'. To allow your PuTTY session to access that port you do the following.

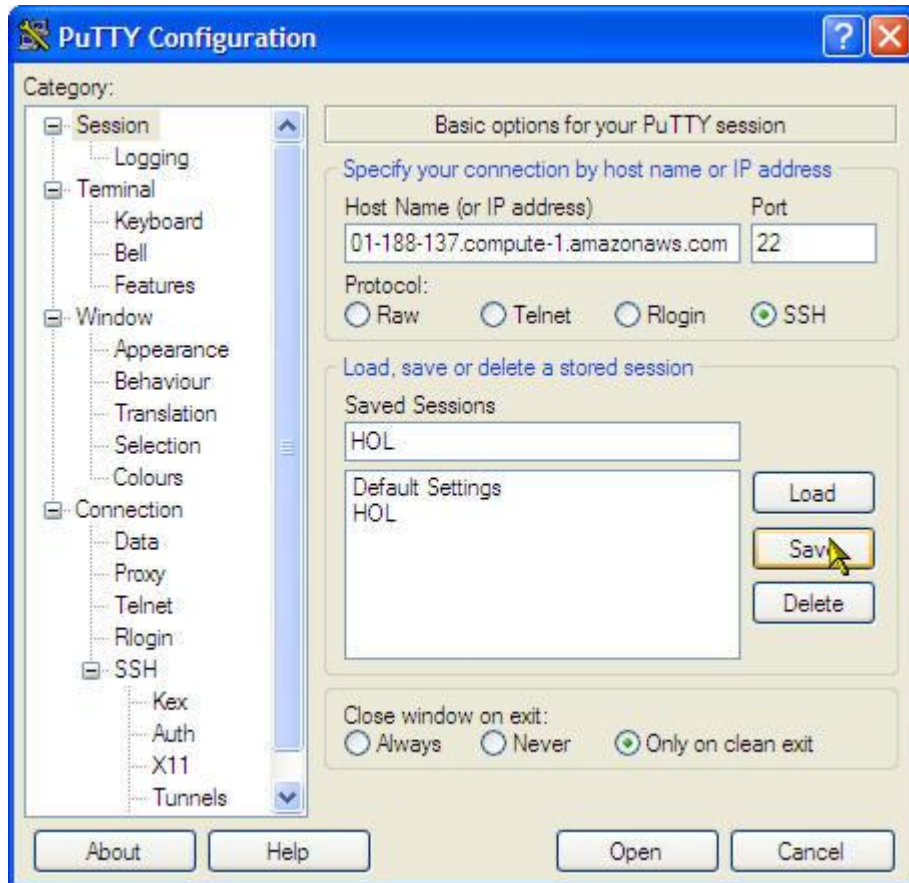
Select 'Tunnels' below 'Auth' (where you configured your key) and enter 5901 in the 'Source Port' field and '127.0.0.1:5901' into the 'Destination' field. Click the 'Add' button.



This allows you to enter '127.0.0.1:1' in the VNC Viewer connection field that you run on your client system.

## *Saving Your PuTTY Configuration*

You now have PuTTY configured. To save these changes, return to the main screen by clicking on ‘Session’ at the top left. Enter your Host Name, for example “ec2-75-101-178-131.compute-1.amazonaws.com” into the “Host Name” field and make sure SSH is checked. Type ‘HOL’ into the ‘Saved Sessions’ field and click the ‘Save’ button.



From now on when you want to attach to your system you would start PuTTY, select ‘HOL’ from the list and click the ‘Load’ button.

Click the ‘Open’ button and you are on your way!



Oracle OpenWorld Active Data Guard Hands on Lab Session ID: S318750

August 2010  
Authors: Larry M. Carpenter

Oracle USA  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
oracle.com

Copyright © 2010, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.