DataGuard Configuration Physical Standby Database with Real Time Apply

Date: March 22, 2016

Contents

1 Overview

This document describes a manual and automatic step-by-step procedures of Oracle replication setup using DataGuard with a REAL TIME APPLY and Active Data Guard options.

Document gives a detailed description of the NDS operational scripts that automate DataGuard replication setup and administrative tasks.

2 Introduction

A physical standby database is physically identical to the primary database, with on disk database structures that are identical to the primary database on a block-for-block basis.

A Physical standby database has several main modes of operation:

- Recovery mode – In which the database base engine on the standby database server constantly applies redo information received from the primary server
- Open read only – In which the database is accessible to users (selects only). In this mode, no redo apply is performed but redo logs continue to be received.
- "Active Data Guard" mode – these options are available since Oracle 11g version and allows to open standby database for reads and to continue with apply (recovery) operation in the same time.

> **CAUTION!**
>
> "Active Data Guard option requires extra license fee on top of Enterprise Edition license

3 Pre-Installation Requirements

- The operating system and platform architecture on the primary and standby locations must be the same. (Oracle 10g)

> **Note** Since Oracle11g it is possible to have primary and standby databases with different CPU architectures, different Operating Systems, different Operating Systems binaries, more details can be found metalink note#413484.1

- The hardware (for example, the number of CPUs, memory size, storage configuration) can be different between the primary and standby systems
- The operating system running on the primary and standby locations must be the same, but the operating system release does not necessarily have to be the same.
- Oracle Data Guard is available only as a feature of Oracle Database Enterprise Edition.
- The primary database must run in a ARCHIVELOG mode
- In order to protect against direct writes that are not recorded in the redo log files in the primary database it is highly recommended to enforce logging at the database level by setting the `FORCE_LOGGING` parameter on the primary database before performing the initial backup of the datafiles for the standby database creation.

> **Note** Changes that have not been recorded in the redo log files cannot be propagated to the standby database, which could lead to a bad copy on the standby database.

4 Active Data Guard (since Oracle 11g)

- Active Data Guard is a new feature available with Oracle 11g EE enables you to open a physical standby database for read-only access for reporting while Redo Apply continues to apply changes from the production database to the standby database.
- With Active Data Guard, you can offload any operation that requires up-to-date, read-only access to the standby database
- Active Data Guard requires a separate license, and extends basic Data Guard functionality included with Oracle Database 11g Enterprise Edition.

Operations Allowable On a Read-Only Database

- Issue SELECT statements, including queries that require multiple sorts that leverage TEMP segments
- Use ALTER SESSION and ALTER SYSTEM statements
- Use SET ROLE
- Call stored procedures
- Use database links (dblinks) to write to remote databases
- Use stored procedures to call remote procedures via dblinks
- Use SET TRANSACTION READ ONLY for transaction level read consistency
- Issue complex queries (such as grouping SET queries and WITH CLAUSE queries)

Operations Disallowed On a Read-Only Database

- Any DMLs (excluding simple SELECT statements) or DDLs
- Query accessing local sequences
- DMLs to local temporary tables

5 Automatic DG setup and administration with NDS DG rpm

5.1 NDS DG rpm overview and rpm installation

Overview

NDS Data Guard rpm is a set of setup, administration and monitoring scripts for Linux/Unix environment that allows system engineer to install and manage Oracle DG replication without DBA assistance.

NDS DG rpm is built according to the following assumptions:

- All DG setup, administration and monitoring activities are performed with Oracle Operating user.
- Secure shell (ssh) between primary system and secondary system is working without prompting for a password in a context of oracle OS user.
- Oracle Enterprise Edition software is installed on primary and secondary systems according to the standard NDS procedures.
- In /etc/hosts files on primary and secondary systems the following cross reference should exist

Primary system /etc/hosts file system example

```
1.1.1.1             orcEMMGpkg
1.1.1.2             orcEMMGpkg_stby
```

Secondary system /etc/hosts file example

```
1.1.1.2             orcEMMGpkg
1.1.1.1             orcEMMGpkg_stby
```

> **Note** The common configuration assumes that primary and secondary systems are running with a Linux cluster where Oracle package has it's own Virtual IP name, like orcEMMGpkg in the examples above. On other hand there is no requirement that both systems will be in a Linux cluster.

- NDS DG rpm assumes that database name on primary and secondary systems is the same.
- NDS DG rpm was tested and supports the following:
  - OS – Linux RH4.4 / Oracle – 10.1.0.4
  - OS – Linux RH5.3 RH5.5 / Oracle – 11.1.0.7 with Active DG

- DG rpm supports backward compatibility

Installation

- The package is installed by a standard rpm utility. During the installation process DG scripts are only copied a system according to the directory structure described below

```
rpm –ivh  nds_ora_dg-<version>
```

- DG rpm scripts are installed under :
  - RH4 /home/software/nds/oracle_dg
  - RH5 /opt/nds/installed/nds_ora_dg
- All logs are generated in the following format :
  - RH4 : /var/log/nds/oracle/oracle_dg_*.log
  - RH5 : /opt/nds/installed/nds_ora_dg/log

5.2 DG configuration file

All scripts from DG rpm reference variables and setting that are defined in a configuration file.

By default the following configuration file is used

```
/opt/nds/installed/nds_ora_dg/etc/dg.cfg
```

There is an option to use a non-default configuration file. In order to that any script has to be activated with a flag –f <new config file path> , for example:

```
check_dg.sh –f /tmp/my_dg.cfg
```

Table 1 describes the DG configuration file parameters

Table 1 [DG Configuration file parameters]

	Parameter name	Default value	Description	Remarks
1.	source_sid	EMMG	Source database name	
2.	source_host	orcEMMGpkg	The source host name, usually DB package name	

	Parameter name	Default value	Description	Remarks
3.	target_sid	EMMG	Target database name, currently we support only the same name as a source.	
4.	target_host	orcEMMGpkg_stby	Target host name, should resolve to the DB package name on the target	
5.	tns_f	/opt/oracle/tnsnames.ora	Location of the tnsnames.ora file	On RH4 the default location is /home/oracle
6.	listener_f	/opt/oracle/listener.ora	Location of the listener.ora file	On RH4 the default location is /home/oracle
7.	arc_dir	/oradata/${source_sid}/arclogs	The default location of the archive files	
8.	clean_dir	/oradata/${source_sid}/arclogs	The default location where cleanup script has to delete files	Usually this is has to be the same location as arc_dir , except special cases when we move arc_dir temporary to an different location because of the space problems
9.	days	7	The default value that instructs a cleanup script number of days to keep applied archive files	
10.	threshold	80	This value indicates the threshold percentage of allowed occupied space in a file system.	If disk space goes beyond this threshold then space will be released on a primary server even if some required for DG archive files will be lost.

	Parameter name	Default value	Description	Remarks
11.	sql_file	${SQL_DIR}/gen_check.sql	SQL script that can be used to check if a standby database is in sync	This parameter is used by a check_data_sync.sh script
12.	switch_mode	rec	Mode to open a standby database	Possible values are: rec - Recovery mode, a normal mode for a standby database operations ro – READ ONLY mode, standby DB is opened for READs by synchronization is stopped but can be switched back to a recovery mode rw - READ WRITE mode, standby database is opened for reads and writes, synchronization is stopped and can not be switched back to a recovery mode. ra – READ ACTIVE DG , (applicable for Oracle 11g only with Active DG license), standby database is opened for reads and continues to apply changes

5.3 Initial DG Setup

In order to perform a first synchronization DG setup script has to be activated.

This script can be run several times till successful setup completion without causing any damage.

Script name : setup_dg.sh

Purpose : Automatic Oracle DG (physical standby) setup

Usage :

setup.sh

```
Example:
/opt/nds_installed/nds_ora_dg/utils/setup.sh
```

> **IMPORTANT!**
>
> When script is invoked without flag -f it uses default configuration file under ${ETC_DIR}/dg.cfg

Exit code : 0 – Success

1 - Failure

Detailed description:

The setup script is responsible for the following operations:

- Full set of pre-setup and pre-requisites checks that are described in the section 3 “Pre-Installation Requirements,” on page 6 and section A.1 “Preparing the Primary Database,” on page 19.
- First synchronization process between primary and secondary systems by coping the primary database datafiles and configuration files using ssh.
- Starting standby system with a real time apply using Standby redo logs
- Post setup verification checks

Script notes:

- Script has to be activated from a primary node only.
- Script can be rerun many times till successful completion
- Script may fail on pre-requisites with the following errors:

- ♦ ERROR – manual intervention of system engineer or DBA is required
- ♦ WARNING – setup script dynamically generates another script that can includes all commands that should fix the missing pre-requisites

- Script assumes that the directory of Oracle datafiles on the primary and secondary systems is exactly the same
- Script assumes that /etc/hosts files on primary and secondary systems are configured in a cross reference format

Primary system /etc/hosts file system example

```
1.1.1.1              orcEMMGpkg
1.1.1.2              orcEMMGpkg_stby
```

Secondary system /etc/hosts file example

```
1.1.1.2              orcEMMGpkg
1.1.1.1              orcEMMGpkg_stby
```

- Script creates Standby Redo Logs

5.4 DG synchronization monitoring

After successful setup check if DG is in sync by running check script.

Script name : check_dg.sh

Purpose : Monitor DG synchronization status between primary and secondary systems.

Usage :

check_dg.sh

Exit code : 0 – Success

1 - Failure

Detailed description:

- Script has to be activated from a primary node only.
- Script fails with error if secondary database is down
- Script prints the time synchronization between primary and secondary systems. The time synchronization is calculated according to the following:
  - ♦ Primary database time – is the time of first SCN of the current redo log
  - ♦ Secondary database time – is the time the last SCN of the last applied redo log

  Time synchronization check example:

```
Primary Database     Secondary Database
Last Recorded        Last Applied
Change Time          Change Time
-------------------- --------------------
25-DEC-07 04:25:17   25-DEC-07 04:25:17
```

- Script prints the current numbers of the redo logs on the primary and secondary systems

```
Primary Current   Log#:110
Primary Last Archived#:109
Standby Last applied log#:109
Standby Last working SRL#:110
```

According to the above output script automatically decides if DG is in sync or not.

5.5 Switchover

Script name : do_switchover.sh

Purpose : Switchover between primary and secondary databases

Usage :

```
do_switchover.sh
```

Exit code : 0 – Success

1 - Failure

Detailed description :

- Script has to be activated from a primary node only.
- Performs all necessary operations to complete a switchover process
- All open connections to the primary database are disconnected automatically during switchover process.
- Adds temporary files to the temp tablespace on a new primary site after switchover.
- DG synchronization is automatically checked upon switchover completion.

5.6 Keeping archive space under control

Script name : clean_ora_arch.sh, clean_dir.sh

Purpose : Clean Oracle archive directory from unnecessary archive files and keep the file system free space below the defined threshold on primary and secondary systems.

Usage :

clean_ora_arch.sh

Example:

```
clean_ora_arch.sh
```

Exit code : 0 – Success

1 – Failure

2 - Warning

Detailed description:

- Script has to be activated from a primary node only.
- Configuration file has to be updated with correct values for a disk threshold and number of days to keep archive files (default values are 80% and 7 days)
- Script fails with error if secondary database is down
- Script performs two types of checks and acts accordingly:
  - Delete unnecessary archive files on the primary and secondary systems according to the following conditions (AND):
    - Deletes archive files that are older then the last applied archive file on the target system
    - Deletes archive files that are older then a number of days that are defined by a “days” parameter inside configuration file
- After completion the previous step script checks if the used space is above the supplied the threshold on the of the file system where archive directory is located. If the used space is above the threshold script deletes archive files till the free space threshold condition is met.

WARNING!

The above operation may lead Oracle DG to get out of sync. Please check the DG synchronization if such event has been reported.

5.7 Operating with a standby database

Script name : open_stby.sh

Purpose : Allows to open the standby database in a READ-ONLY,RECOVERY,READ-WRITE or Active “Read Only” mode.

Usage :

open_stby.sh

```
Example:
open_stby.sh
```

Exit code : 0 – Success

1 - Failure

Detailed description :

- Script has to be activated from a primary node only.
- Script opens a standby database according to the defined parameter “mode” inside a configuration file.

> **Note**
>
> It is important to understand the exact meaning of each mode:
>
> RECOVERY - a normal state of the standby database. Standby database is closed for any activity and all the time performs a recovery operations.
>
> READ-ONLY - allows connecting to the standby database and running selects only. In this mode the recovery is stopped by changes continue to flow from primary to the secondary system. It is possible to switch back to the recovery mode.
>
> READ-WRITE - opens a standby database in for reads and writes. This operation takes standby database out of the DG replication and cannot be rolled back. After completion of this operation database cannot apply any changes that are received from the primary. The only option to synchronize the systems back is to run the setup procedure one more time.
>
> ACTIVE READ-ONLY - available since Oracle 11g only (**requires extra license fee on top of EE license).** This mode allows to run select statements directly on the standby database without stopping the recovery process.

5.8 Checking the data synchronization with a free SQL select

Script name : check_data_sync.sh

Purpose : Allows to check the DG data synchronization based on the free sql select that is supplied by a script

Usage :

check_data_sync.sh

```
Example:
check_data_sync.sh
```

Exit code : 0 – Success

1 - Failure

Detailed description:

- Script has to be activated from a primary node only.
- Script activates a user supplied SQL select statements that in a form of sql script on primary and secondary systems
- In order to run select statements the secondary database is opened in a READ-ONLY mode and after that opened back in the RECOVERY mode.

> **Note** The data synchronization script does not perform any validation checks on the supplied SQL scripts

5.9 Startup of Oracle with DG

- Oracle database that is running in a standby mode has to be started using special DG commands and not by using a common Oracle startup commands that are included in the NDS oracle package.
- In order to implement a correct Oracle startup procedure in a Data Guard environment do the following on both primary and secondary systems:

  1. Backup the current oracle_start.sh script

     ```
     cp  oracle_start.sh oracle_start.sh.original
     ```

  2. Copy the Oracle startup script that are specific for a Data Guard environment

     ```
     cp /opt/nds/installed/nds_ora_dg/utils/oracle_start.sh
     oracle_start.sh
     ```

- The new oracle_start.sh script activates a special Oracle DG startup script oracle_dg_start_check.sh that is responsible to identify whether the database in a standby mode and to open it accordingly.

5.10 Integration with NDS monitor

The DG health check is integrated with NDS monitor.

The wrapper script nds_mon_check_dg.sh is responsible to activate check_dg.sh script and redirect the check_dg.sh script output to the standard output in the format of NDS monitor.

Appendix A Manual DG (physical standby) setup procedure

A.1 Preparing the Primary Database

Make sure that force logging is enabled:

```
SQL>
select name,
       force_logging
FROM   v$database;
```

Sample Output (when force logging is enabled)

```
NAME      FORCE_LOGGING
--------- -------------
SSR       YES
```

In order to enable force logging, run the following command:

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

Make sure that you have a password file

```
   OS> ll $ORACLE_HOME/dbs/orapwSSR
```

In order to create the password file run the following command:

```
   OS> orapwd file=$ORACLE_HOME/dbs/orapwSSR password=<XXX>
```

Configure standby redo log files (SRL)

> **IMPORTANT!**
>
> Real-time apply allows log apply services to recover and apply redo data from standby redo log files at the same time these files are receiving new redo data from the primary database. This ensures the lag between the standby database and the primary database is as small as possible.

Ensure that the size of SRL files are identical to those of the redo log (RL) files by running the following command:

```
SQL>
SELECT group#,
     members,
     bytes/(1024*1024) "Size in MBS"
FROM    v$log;
```

```
Sample Output:
    GROUP#    MEMBERS Size in MBS
---------- ---------- -----------
```

```
         1          1         128
         2          1         128
         3          1         128
```

Use the following formula to determinate the appropriate number of SRL groups:

(Number of RL groups + 1) * number of threads

Create SRL groups (the number of groups was determined in the above formula):

```
SQL>
ALTER DATABASE ADD STANDBY LOGFILE GROUP <group#>
   ('<file name>') SIZE 128M;

Example:
    SQL>
ALTER DATABASE ADD STANDBY LOGFILE GROUP 7
      ('/oradata/SSR/stbyredo_SSR_04.log') SIZE 128M;
```

Verify that SRL files were created by running the following command:

```
    SQL>
SELECT group#,
       thread#,
       sequence#,
       archived,
       status
 FROM  v$standby_log;
```

```
Sample Output:
    GROUP#    THREAD#  SEQUENCE# ARC STATUS
---------- ---------- ---------- --- ----------
         4          0          0 YES UNASSIGNED
         5          0          0 YES UNASSIGNED
         6          0          0 YES UNASSIGNED
         7          0          0 YES UNASSIGNED
```

The following configuration parameters must be set in the Primary Database Oracle configuration file (SSR DB example):

```
    # DataGuard section
    log_archive_dest_1='LOCATION=/oradata/SSR/arclogs'
    log_archive_dest_state_1=ENABLE
    SERVICE_NAMES=SSR
    instance_name=SSR
    log_archive_dest_2='SERVICE=SSR_STBY LGWR ASYNC
                                VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)'
    log_archive_dest_state_2=enable
    log_archive_format='archive_SSR_%r_%t_%s.arc'
```

```
    # The following parameters also need to be set in the Primary
    # Database Oracle Configuration, but will only take effect at such
    # time when the Primary database assumes the standby role.
    fal_server=SSR_STBY
    fal_client=SSR
    LOG_FILE_NAME_CONVERT='/oradata/SSR/','/oradata/SSR/'
```

The following errors which might appear in alert file can be ignored:

```
<<
Errors in file /home/oracle/orahome/admin/SSR/bdump/ssr_arc1_1270.trc:
ORA-12154: TNS:could not resolve the connect identifier specified
PING[ARC1]: Heartbeat failed to connect to standby 'SSR_STBY'. Error is 12154.
>>
```

The reason for the above errors is because Standby database has not yet been started up.

Verify that the Primary database is running in archive log mode:

Run the following command as sysdba:

```
SQL>archive log list

Sample Output:
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            /oradata/SSR/arclogs
Oldest online log sequence     13
Next log sequence to archive   15
Current log sequence           15
```

If the database is not in archive mode do the following in order to place it in archive log mode:

```
SQL>
shutdown
startup mount
alter database archivelog
alter database open
```

A.2 Creating a Physical Standby Database

A.2.1 Take a full backup copy of the primary database (Hot Backup Option)

Check the current maximum archive log by running on the Primary database as follows:

```
SQL>
SELECT name,
       sequence#,
       first_change#,
       next_change#
FROM   v$archived_log
WHERE  sequence# >= (SELECT MAX(sequence#)
              FROM   v$archived_log);
```

```
Sample Output:
   NAME                                               SEQUENCE# FIRST_CHANGE#
   NEXT_CHANGE#
   -------------------------------------------------- --------- ------------- -
   -----------
   /oradata/SSR/arclogs/SSR_1_17_605987557.arc           17        304564
        417698
```

Connect as sysdba to the Primary database and execute the following command in order to place the database in a hot backup mode state:

```
SQL>   alter database begin backup;
```

Copy all datafiles and all **redo log files** to the standby system via standard O.S.commands.

Terminate the hot backup mode state by executing the following command:

```
SQL> alter database end backup;
```

Create a standby control file by executing the following:

```
SQL>
alter database create standby controlfile as
                    '/oradata/SSR_STBY/control_stby.ctl';
```

Archive all online redo log files by executing the following command:

```
SQL>
alter system switch logfile;
```

*Note:
The above command should be issued as many times as the number of archive log groups.

A.2.2 Creating an Oracle Password file on the Standby System

In order to create a password file run the following commands as oracle:

```
OS>
cd $ORACLE_HOME/dbs
orapwd file=orapwSSR_STBY password=< XXX >
```

Note: The password for user sys should be identical to the password on the primary system.

A.2.3 Modifying the configuration files listener.ora file and tnsnames.ora

Changing the listener name from SSR to SSR_STBY in the listener.ora file:

```
SSR_STBY=
 (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)(Host=orcssr_stby_pkg)(Port=1521)))

SID_LIST_SSR_STBY=
 (SID_LIST=
   (SID_DESC=
     (SID_NAME=SSR_STBY)
     (SDU=8192)
     (ORACLE_HOME=/home/oracle/orahome)
   )
 )
```

A.2.4 Starting the physical Standby Database

Execute the following commands in order to start the physical standby database's apply services (real time):

```
SQL>
STARTUP MOUNT;
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
      USING CURRENT LOGFILE DISCONNECT FROM SESSION;
```

```
At this time, on the Primary Database execute a log switch in order to
start transmit the redo log files to the Standby system:
```

```
SQL>
alter system switch logfile;
```

A.3 Verifying physical Standby Database

Checking archived log application on the standby database:

Actions to be taken on the physical standby database:
Identify what archive files were received and applied by executing the following:

```
SQL>
SELECT sequence#,
 first_time,
 next_time,
 registrar,
 applied,
 fal
     FROM   v$archived_log
     ORDER BY sequence#;

     Sample Output:
     SEQUENCE# FIRST_TIME NEXT_TIME REGISTR APPLIED FAL
   --------- ---------- --------- ------- ------- ---
         28 12-NOV-06  12-NOV-06 RFS     YES     NO
 29 12-NOV-06  12-NOV-06 RFS     YES     YES
 30 12-NOV-06  12-NOV-06 RFS     YES     YES
 31 12-NOV-06  14-NOV-06 RFS     YES     YES
 32 14-NOV-06  14-NOV-06 RFS     YES     YES
 33 14-NOV-06  14-NOV-06 RFS     YES     NO
 34 14-NOV-06  14-NOV-06 RFS     YES     NO
```

Actions to be taken on the primary database:
Identify what is the sequence of the current not archived redo log by running the following:

```
SQL>
SELECT sequence#,
     status,
     archived
         FROM   v$log;

         Sample Output:
      SEQUENCE# STATUS           ARCHIVED
      --------- ---------------- --------
         34 INACTIVE         YES
         35 CURRENT          NO
         33 INACTIVE         YES
```

Archiving the current log file by running:

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

Verifying that current log was archived by running once again the previous select statement:

```
SQL>
SELECT sequence#,
     status,
     archived
         FROM   v$log;

         Sample Output:
      SEQUENCE# STATUS           ARCHIVED
      --------- ---------------- --------
        34 INACTIVE         YES
        35 INACTIVE         YES
        36 CURRENT          NO
```

Additional actions to be taken on the physical standby database:
Checking to see that this log file sequence# has been applied:

```
SQL>
SELECT sequence#,
 first_time,
 next_time,
 registrar,
 applied,
 fal
          FROM  v$archived_log
          ORDER BY sequence#;

       Sample Output:
       SEQUENCE# FIRST_TIME NEXT_TIME REGISTR APPLIED FAL
       --------- ---------- --------- ------- ------- ---
                  31 12-NOV-06  14-NOV-06 RFS     YES     YES
                  32 14-NOV-06  14-NOV-06 RFS     YES     YES
                  33 14-NOV-06  14-NOV-06 RFS     YES     NO
                  34 14-NOV-06  14-NOV-06 RFS     YES     NO
                  35 14-NOV-06  15-NOV-06 RFS     YES     NO
```

Additional actions to be taken on the primary database:
Checking to see that the redo log has been written to the standby redo log:

```
SQL>
SELECT  sequence#,
                status
 FROM   v$log;

      Sample Output:
      SEQUENCE#  STATUS
      ---------- ----------------
            55 INACTIVE
            56 CURRENT
            54 INACTIVE
```

Additional actions to be taken on the physical standby database:

Checking the status of the standby redo logs:

```
SQL>
SELECT group#,
 thread#,
sequence#,
bytes/(1024*1024),
used/(1024*1024),
status
     FROM  v$standby_log
     WHERE sequence <= &current_sequence_from_primary

     Example:
     SQL>
     SELECT thread#,
        sequence#,
        bytes /(1024*1024) FILE_SIZE_MBS
        used /(1024*1024)  USED_MBS,
        status
     FROM   v$standby_log
     WHERE  sequence# <= 56;
```

Sample Output:

THREAD#	SEQUENCE#	FILE_SIZE_MBS	USED_MBS	STATUS
1	56	128	1.88330078	ACTIVE
1	0	128	.000488281	UNASSIGNED
0	0	128	.000488281	UNASSIGNED
0	0	128	.000488281	UNASSIGNED

Appendix B Manual operating with a physical Standby Database

B.1 Start/Stop redo apply

Starting redo apply in the physical stand by database:

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

Starting redo apply in the physical standby database (real time redo apply):

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

Verifying that the real time redo apply is working properly:

```
SQL>
SELECT dest_id,
   recovery_mode
FROM   v$archive_dest_status;
```

Note:
For real time apply, the recovery_mode is 'MANAGED REAL TIME APPLY'

Stopping redo apply in the physical standby database (redo real time redo apply):

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

B.2 Change between recovery , read-only mode and "Active DG " modes

From recovery mode to read-only mode:

Stop redo apply:

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

Open a physical standby database in a **read only** mode

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

Connect to the standby database and perform any type of select.

From read-only mode to recovery mode:

Kill all connected sessions
Open a physical standby database in recovery mode (real time apply)

```
SQL>

ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
DISCONNECT FROM SESSION;
```

From recovery to "Active DG" mode:

> **CAUTION!**
>
> This can be done in Oracle 11g only and requires special Active DG license on top of EE license.

Stop redo apply:

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

Open database in a read only mode

```
SQL> alter database open read only;
```

Continue redo apply

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

Check that standby DB is working in "Active DG" mode

```
SQL> SELECT 'Using Active Data Guard' ADG
 FROM V$MANAGED_STANDBY M, V$DATABASE D
WHERE M.PROCESS LIKE 'MRP%' AND D.OPEN_MODE='READ ONLY';
```

B.3 Switchover procedure

Definition: Switchover allows the primary database to switch roles with one of its standby databases. There is no data loss during a switchover. After a switchover, each database continues to participate in the Data Guard configuration with its new role.

A switchover is typically used to reduce primary database downtime during planned outages, such as operating system or hardware upgrades, or rolling upgrades of the Oracle database software and patch sets

Step1:

On the primary database check if it is possible to perform a switchover

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;
SWITCHOVER_STATUS
--------------------
TO STANDBY
```

Note: The TO STANDBY value in the SWITCHOVER_STATUS column indicates that it is possible to switch the primary database to the standby role

Step 2:

Start the switchover on primary by running

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY;
```

After this statement completes, the primary database is converted into a standby database and dismounted.

Step 3:

Perform restart of the former primary database

```
SQL> shutdown immediate
SQL> startup mount
```

At this stage both databases are considered as standby databases.

Step 4:

Switch the former standby database to the primary by

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
SQL> shutdown immediate
SQL> startup
```

Step 5:

On a new standby database start physical apply by

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

Step 5.1:

SQL> Check that you can login from a new primary site to the secondary site as a sysdba

```
sqlplus  “sys@<alias> as sysdba
```

Step 6: Check the role and status of both databases

```
SQL> select name,open_mode,DATABASE_ROLE,SWITCHOVER_STATUS from
v$database;
```

On new primary the output should be:

```
NAME      OPEN_MODE   DATABASE_ROLE    SWITCHOVER_STATUS
--------- ---------- ---------------- --------------------
SSR       READ WRITE PRIMARY          TO STANDBY
```

On new standby the output should be:

```
NAME      OPEN_MODE  DATABASE_ROLE    SWITCHOVER_STATUS
--------- ---------- ---------------- --------------------
SSR       MOUNTED    PHYSICAL STANDBY NOT ALLOWED
```

Step 7: Check if a new standby receives redo logs from the new primary

From a new primary:

```
SQL> select sequence# from v$log where status='CURRENT';
For example:
SEQUENCE#
----------
        85
SQL> alter system switch logfile;
```

On the new secondary:

Verify that the switched redo log was received and applied

```
SQL> SELECT sequence#,first_time, next_time,registrar,applied
  2  FROM   v$archived_log
  3  where sequence#=85;

 SEQUENCE# FIRST_TIM NEXT_TIME REGISTR APP
---------- --------- --------- ------- ---
        85 01-FEB-07 01-FEB-07 RFS     YES
```

Step 8: Check if a new standby working with standby redo log correctly

On the new primary:

```
 SQL> select sequence# from v$log where status='CURRENT';
```

For example:

```
SEQUENCE#
----------
        85
On the secondary:
SQL>
 select SEQUENCE#,bytes/1024/1024,used/1024/1024    from v$standby_log
  where sequence#=86;

 SEQUENCE# BYTES/1024/1024 USED/1024/1024
---------- --------------- --------------
        86             128     .986328125
```

B.4 Failover procedure

Definition: A failover is used when a primary site becomes completely unavailable and there is completely no possibility of restoring it to a service in a reasonable period of time.

> **Note**
>
> After a failover, the original primary database can no longer participate in a DG configuration.
>
> In order to add the original primary database to the DG configuration in a future initial DG setup has to be performed from a new primary system.

> **IMPORTANT!**
>
> All the steps listed below have to be run from a standby system.

1. Identify and resolve if possible any data gaps by doing the following:

```
SQL> select thread#,low_sequence#,high_sequence# from v$archive_gap;
```

If archive files that include data gap were found then register them in a standby database manually by running the following command:

```
SQL> ALTER DATABASE REGISTER PHYSICAL LOGFILE  '<full path to the
file>';
```

2. Start the failover operations:

```
SQL> alter database recover managed standby database cancel;
SQL> alter database recover managed standby database disconnect from
session;
SQL> shutdown immediate;
SQL> startup mount;
SQL> alter database recover managed standby database finish;
SQL>  alter database commit to switchover to primary;
SQL>  startup force;
```

3. Check if the database has been opened in a READ-WRITE mode

```
SQL> select open_mode from v$database;
```

4. Add the temporary files to the new primary database

```
SQL>
    set head off
    set pages 0
    set feed off
    set serveroutput on
    declare
    str varchar2(500);
    BEGIN
     for i in (select FILENAME from dba_hist_tempfile
           minus
           select name from v\\\$tempfile ) loop
      str:='alter tablespace temp add tempfile '''||i.filename||'''
reuse';
      execute immediate str;
     end loop;
   dbms_output.put_line('The following temp files were added');
   for i in (select name,bytes from v\\\$tempfile) loop
   dbms_output.put_line('Name: '||i.name||' Size:
'||i.bytes/1024/1024||' MB');
   end loop;
   end;
   /
```