HANDS ON LAB ORACLE19C DATAGUARD

DATE: 26-10-2020

Change History

VERSION	DATE	DESCRTIPTION	AUTHOR
0.1	26-10-2020	Initial version	Rob Lasonder

Sources

SOURCE

Contents

1. Overview

In this lab I will install the Oracle 19c software in a Virtual Box environment in 2 Oracle Homes. These Homes will be configured as READ ONLY Oracle Homes. I will then create a physcial standby database with active dataguard and the dataguard broker, plus test some of the latest Dataguard Features.

Prerequisites:

- I am using Oracle Virtual Box 5.2.34 for this lab with 8 GB of available memory, 2 CPU and 80 GB of internal storage.
- I already have this Virtual Server setup and Linux 7 installed and configured
- I already have 2 Oracle Homes installed and configured (as READ ONLY Oracle_homes).
- I already have the source database prod created.

If you want to know how this can be done, please refer to one of my other hands on labs on this website:
"**HANDS ON LAB ORACLE VIRTUAL BOX 19C**"

The enviroment to be configured in this lab will be as follows:

Primary database prod:

- Instance_name **prod_db1**
- db_name prod, db_unique_name prod_db1
- host rob01db01
- db_unique_name prod_db1
- Oracle_Home = /u01/app/oracle/product/19.3.0/dbhome_1
- Db_create_file_dest = /u01/app/oracle/oradata/prod
- Db_recovery_file_dest = /media/sf_Shared/prod_db1_fra
- Listener:LISTENER1l on port 1521

Standby database prod

- Instance_name **prod_db2**
- db_name prod, db_unqiue_name prod_db2
- host rob01db01.
- Oracle_Home = /u01/app/oracle/product/19.3.0/dbhome_2
- Db_create_file_dest = /home/oracle/oradata/prod/prod_db2_fra
- Db_recovery_file_dest = /media/sf_Shared/prod_db1_fra
- Listener: LISTENER2 on port 1522

2. Create a physcial standby database

2.1. Prepare the Primary Database prod_db1

2.1.1. Change the logging modes of the database:

```
$ . oreanv => prod

$ sqlplus / as sysdba

SQL> select log_mode, force_logging, flashback_on from v$database;

LOG_MODE     FORCE_LOGGING                          FLASHBACK_ON
------------ -------------------------------------- ------------------
ARCHIVELOG   YES                                    NO

SQL> alter database flashback on;

SQL> alter database force logging;
```

Remarks:

- force logging is required to prevent NOLOGGING operations. We want all the data to be written to the online redo log files, otherwise we can face data loss and corruption.
- Flashback database is required to re-instate the Primary database after a failover and it is also needed on the standby database, for example when you create a snapshot standby database.

2.1.2. Change the db_unique_name

```
SQL> show parameter db_unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      prod

SQL> alter system set db_unique_name = 'prod_db1' scope = spfile;

System altered.

SQL> shutdown immediate

SQL> startup

SQL> show parameter db_unique_name

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      prod_db1
```

Remark: as a result, the database service prod_db1 is created and started. This service will be configured for database connectivity later on in this lab.

```
SQL> select name from v$active_services where name like 'prod_db1%';

NAME
----------------------------------------------------------------
prod_db1.robdomain
```

2.1.3. Change the ORACLE_SID

Because I am going to run both the Primary and the Standby database on the same server, I am going to rename the SID of each of these databases to the db_unique_name equivalent. If you are running the Standby Database server on a different server this step can be skipped.

```
SQL> alter system set instance_name='prod_db1' scope=spfile;

System altered.

SQL> shutdown immediate
```

We now also need to change the /etc/oratab file. The oratab file contains entries in the form of ORACLE_SID:ORACLE_HOME:Y, the last character being a Y or N indicating if the database should be started and stopped with the dbstart and dbstop commands, respectively.

Change prod to prod_db1:

```
[oracle@rob01db01 dbs]$ vi /etc/oratab
prod_db1:/u01/app/oracle/product/19.3.0/dbhome_1:N
```

We also need to rename the password file and the spfile with this new instance name:

```
[oracle@rob01db01 install]$ cd /u01/app/oracle/dbs
 [oracle@rob01db01 dbs]$ mv spfileprod.ora spfileprod_db1.ora
[oracle@rob01db01 dbs]$ mv orapwprod orapwprod_db1
```

Remark: notice that I am using a Read Only Oracle Home (Oracle18c new feature) with a different path to these files.

Now start the instance again:

```
[oracle@rob01db01 dbs]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Oct 30 16:20:45 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup
ORACLE instance started.

Total System Global Area 1258287344 bytes
Fixed Size                  9134320 bytes
Variable Size             318767104 bytes
Database Buffers          922746880 bytes
Redo Buffers                7639040 bytes
Database mounted.
Database opened.
SQL> !ps -ef|grep pmon
oracle   15106     1  0 16:20 ?        00:00:00 ora_pmon_prod_db1
```

Also, change the local listener entry in the spfile and in the tnsnames.ora

```
SQL> alter system set local_listener=LISTENER_PROD_DB1;

System altered.

$ vi /u01/app/oracle/homes/OraDB19Home1/network/admin/tnsnames.ora

LISTENER_PROD_DB1 =
  (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1521))
```

2.1.4. Change dataguard related parameters

```
SQL> alter system set fal_server='prod_db2';

SQL> alter system set standby_file_management=auto;

SQL> alter system set log_archive_config='DG_CONFIG=(prod_db1, prod_db2)' scope=both;

SQL> alter system set db_lost_write_protect='TYPICAL';

SQL> alter system set log_archive_dest_2='SERVICE=prod_db2 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
LGWR ASYNC DB_UNIQUE_NAME=prod_db2' scope = both;

SQL> alter system set log_archive_dest_state_2=ENABLE;

--just chaning this setting to create a seperate recovery environment for the Primary and the Standby
SQL> alter system set db_recovery_file_dest = '/media/sf_Shared/prod_db1'

SQL> !mkdir -p /media/sf_Shared/prod_db1
```

Remark: the archive dest parameters are just needed temporarily, as they will be removed once I create the broker configuration. The broker settings will then override these 2 parameters.

Remark: by default, all the PDBs will be replicated. I will show in one of the later chapters how to exclude or include certain PDBS from the replication.

```
SQL> show parameter enabled_PDBs

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
enabled_PDBs_on_standby              string      *
```

2.1.5. Change the rman retention settings

```
$ rman target /

RMAN> configure archivelog deletion policy to shipped to all standby;
```

2.1.6. Create standby redo log files:

```
SQL> select group#, members, bytes/(1024*1024), status from v$log;

    GROUP#    MEMBERS BYTES/(1024*1024) STATUS
---------- ---------- ----------------- ----------------
         1          2                50 INACTIVE
         2          2                50 CURRENT
         3          2                50 INACTIVE

SQL> alter database add standby logfile thread 1 group 11 size 50M;
SQL> alter database add standby logfile thread 1 group 12 size 50M;
SQL> alter database add standby logfile thread 1 group 13 size 50M;
SQL> alter database add standby logfile thread 1 group 14 size 50M;
```

```
SQL> select group#, thread#, status, bytes/(1024*1024) size_MB from v$standby_log;

    GROUP#    THREAD# STATUS        SIZE_MB
---------- ---------- ---------- ----------
        11          1 UNASSIGNED         50
        12          1 UNASSIGNED         50
        13          1 UNASSIGNED         50
        14          1 UNASSIGNED         50
```

2.1.7. Add tns and listener.ora entries on the Primary database

In the primary database environment: add the tnsnames.ora entries for the dataguard databases:

```
[oracle@rob01db01 admin]$ export ORACLE_HOME=/u01/app/oracle/product/19.3.0/dbhome_1
[oracle@rob01db01 ~]$ cd /u01/app/oracle/homes/OraDB19Home1/network/admin
[oracle@rob01db01 admin]$ vi tnsnames.ora

# entries for Dataguard Configuration

PROD_DB1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = prod_db1.robdomain)
      (UR=A)
    )
  )

PROD_DB2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1522))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = prod_db2.robdomain)
      (UR=A)
    )
  )
```

Remark: the UR=A clause allows for remote connections to connect to databases in NOMOUNT or RESTRICTED mode. This will be needed by the Dataguard Broker in case of switchover/failover operations.

In the primary database environment: add a static entry for the listener.ora

```
[oracle@rob01db01 admin]$ vi listener.ora

# static entries for dataguard configuration
SID_LIST_LISTENER1 =
  (SID_LIST =
     (SID_DESC =
      (SID_NAME = prod_db1)
      (GLOBAL_DBNAME=prod_db1_DGMGRL.robdomain)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
    )
    (SID_DESC =
      (SID_NAME = prod_db1)
      (GLOBAL_DBNAME=prod_db1.robdomain)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
    )
  )
```

The listener needs to be stopped and started for the changes to take effect. A listener reload will not do this.

```
[oracle@rob01db01 admin]$ lsnrctl stop listener1

[oracle@rob01db01 admin]$ lsnrctl start listener1
```

Remark about static entries, reference: Oracle Data Guard Broker and Static Service Registration (Doc ID 1387859.1):

- A static entry is needed for the Broker to be able to connect remotely to a database that has been shut down by the Broker during certain operations.
- To access a database that has been shutdown the Broker uses a default name for the static entry using the information from the LOCAL_LISTENER parameter of the instance and the keyword "_DGMGRL" and stores that information in the broker StaticConnectIdentifier property associated with each instance of the database.
- Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard Broker configurations that are managed by Oracle Restart, RAC On Node or RAC as the Broker will use the clusterware to restart an instance. (But because I am not using Oracle RAC or Restart in this lab, they are still required.)

Check the connectivity:

```
[oracle@rob01db01 admin]$ sqlplus sys/oracle@prod_db1 as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Oct 30 08:40:24 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

2.1.8. Copy the password file and the init.ora file from the Primary to the Standby

Because I am hosting the Primary and the Standby on the same server, this goes as follows

```
[oracle@rob01db01 dbs]$ cd $ORACLE_BASE/dbs

[oracle@rob01db01 dbs]$ cp orapwprod_db1 orapwprod_db2

SQL> create pfile = '/u01/app/oracle/dbs/initprod_db2.ora' from spfile;

File created.
```

2.2. Prepare the Standby Database prod_db2

2.2.1. Update the oratab

Add the following entry:

```
prod_db1:/u01/app/oracle/product/19.3.0/dbhome_1:N
prod_db2: /u01/app/oracle/product/19.3.0/dbhome_2:N
```

2.2.2. Edit the ini.ora file

```
[oracle@rob01db01 admin]$ cd $ORACLE_BASE/dbs
[oracle@rob01db01 admin]$ vi initprod_db2.ora
```

Perform the following changes:

- remove all the __parameters
- remove the control_file parameter
- remove the archive_log_dest_2 parameters

- change the following parameters to:
  - *.audit_file_dest='/u01/app/oracle/admin/prod_db2/adump'
  - *.db_create_file_dest='/home/oracle/oradata'
  - *.db_unique_name='prod_db2'
  - *.fal_server='prod_db1'
  - *.instance_name=prod_db2
  - *.local_listener='LISTENER_PROD_DB2'

Remark: db_create_file_dest needs to be changed only because I do not have sufficient disk space.

Here you can see the complete edited initprod_db2.ora

```
*.audit_file_dest='/u01/app/oracle/admin/prod_db2/adump'
*.audit_trail='db'
*.compatible='19.0.0'
*.db_block_size=8192
*.db_create_file_dest='/home/oracle/oradata'
*.db_domain='robdomain'
*.db_lost_write_protect='TYPICAL'
*.db_name='prod'
*.db_recovery_file_dest='/media/sf_Shared/prod_db2'
*.db_recovery_file_dest_size=20g
*.db_unique_name='prod_db2'
*.diagnostic_dest='/u01/app/oracle'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=prodXDB)'
*.enable_pluggable_database=true
*.fal_server='prod_db1'
*.instance_name='prod_db2'
*.local_listener='LISTENER_PROD_DB2'
```

```
*.log_archive_config='DG_CONFIG=(prod_db1, prod_db2)'
*.log_archive_format='%t_%s_%r.dbf'
*.nls_language='AMERICAN'
*.nls_territory='AMERICA'
*.open_cursors=300
*.pga_aggregate_target=248m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=1200m
*.standby_file_management='AUTO'
*.undo_tablespace='UNDOTBS1'
```

2.2.3. Create the audit directory and the recovery_file_dest directory

```
[oracle@rob01db01 dbs]$ $ mkdir -p u01/app/oracle/admin/prod_db2/adump

[oracle@rob01db01 dbs]$ mkdir -p /media/sf_Shared/prod_db2
```

2.2.4. Create an spfile and start the instance in nomount mode

```
[oracle@rob01db01 dbs]$ . oraenv
ORACLE_SID = [prod_db1] ? prod_db2
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@rob01db01 dbs]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Oct 30 16:51:01 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> create spfile from pfile;

SQL> startup nomount;
ORACLE instance started.

Total System Global Area 1258287344 bytes
Fixed Size                  9134320 bytes
Variable Size             318767104 bytes
Database Buffers          922746880 bytes
Redo Buffers                7639040 bytes

SQL> !ps -ef|grep pmon
oracle   15106     1  0 16:20 ?        00:00:00 ora_pmon_prod_db1
oracle   17411     1  0 16:51 ?        00:00:00 ora_pmon_prod_db2
```

2.2.5. Update the listener.ora and tnsnames.ora

Add the following entries to the tnsnames.ora:

```
[oracle@rob01db01 admin]$ vi /u01/app/oracle/homes/OraDB19Home2/network/admin/tnsnames.ora
# added entries for Dataguard Configuration

PROD_DB1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = prod_db1.robdomain)
      (UR=A)
    )
  )

PROD_DB2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1522))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = prod_db2.robdomain)
      (UR=A)
    )
  )

LISTENER_PROD_DB2 =
  (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1522))
```

Add the following entries to the listener.ora:

```
[oracle@rob01db01 admin]$ vi /u01/app/oracle/homes/OraDB19Home2/network/admin/listener.ora

#Oracle19c New Feature: Network Log File Segmentation
#because it is a test environment just keep 5 files of 2 MB each
LOG_FILE_NUM_LISTENER1=5
LOG_FILE_SIZE_LISTENER1=2

# static entries for dataguard configuration
SID_LIST_LISTENER2 =
  (SID_LIST =
     (SID_DESC =
      (SID_NAME = prod_db2)
      (GLOBAL_DBNAME=prod_db2_DGMGRL.robdomain)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_2)
    )
    (SID_DESC =
      (SID_NAME = prod_db2)
      (GLOBAL_DBNAME=prod_db2.robdomain)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_2)
    )
  )
```

Stop and start the listener2 for the changes to take effect:

```
[oracle@rob01db01 admin]$ lsnrctl stop listener2

[oracle@rob01db01 admin]$ lsnrctl start listener2
```

2.2.6. Overall Connectivity Check from prod_db1

Set the environment to prod_db1 and check if you can reach the primary and standby instance:

```
[oracle@rob01db01 admin]$ . oraenv
ORACLE_SID = [prod_db2] ? prod_db1
The Oracle base remains unchanged with value /u01/app/oracle

[oracle@rob01db01 admin]$ sqlplus sys/oracle@prod_db1 as sysdba

SQL> select instance_name, status from v$instance;

INSTANCE_NAME    STATUS
---------------- ------------
prod_db1         OPEN

[oracle@rob01db01 admin]$ sqlplus sys/oracle@prod_db2 as sysdba

SQL> select instance_name, status from v$instance;

INSTANCE_NAME    STATUS
---------------- ------------
prod_db2         STARTED
```

2.2.7. Overall Connectivity Check from prod_db2

Set the environment to prod_db2 and check if you can reach the primary and standby instance:

```
[oracle@rob01db01 admin]$ . oraenv
ORACLE_SID = [prod_db1] ? prod_db2
The Oracle base remains unchanged2with value /u01/app/oracle

[oracle@rob01db01 admin]$ sqlplus sys/oracle@prod_db2 as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Oct 30 19:06:32 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> exit
Disconnected from Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> select instance_name, status from v$instance;

INSTANCE_NAME    STATUS
---------------- ------------
prod_db2         STARTED

SQL> exit
Disconnected from Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

2.3. Create the Standby Database with RMAN duplicate from active database

Set the environment to the standby database (at the moment still the auxiliary instance) prod_db2, which is started in nomount mode:

```
[oracle@rob01db01 admin]$ . oraenv
ORACLE_SID = [prod_db2] ? prod_db2
The Oracle base remains unchanged with value /u01/app/oracle

[oracle@rob01db01 ~]$ export NLS_DATE_FORMAT='DD-MON-YYYY HH24:MI:SS'

[oracle@rob01db01 admin]$ rman TARGET sys/oracle@prod_db1 AUXILIARY sys/oracle@prod_db2

Recovery Manager: Release 19.0.0.0.0 - Production on Fri Oct 30 19:58:29 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: PROD (DBID=466914583)
connected to auxiliary database: PROD (not mounted)

RMAN> DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE DORECOVER NOFILENAMECHECK;
```

The complete output of this statement is provided in Supplement 1 of this document.

Perform a few post configuration actions:

```
SQL> alter database flashback on;

Database altered.

SQL> alter database enable block change tracking;

Database altered.

$ rman target /

RMAN> CONFIGURE ARCHIVELOG DELETION POLICY to applied on standby;

using target database control file instead of recovery catalog
new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON STANDBY;
new RMAN configuration parameters are successfully stored
```

Start the managed database recovery:

```
[oracle@rob01db01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Oct 30 20:57:58 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> alter database recover managed standby database disconnect from session;

Database altered.
```

Let's check the current configuration.

```
SQL> select message from v$dataguard_status;

MESSAGE
--------------------------------------------------------------------------------
Redo network throttle feature is disabled at mount time
STARTING ARCH PROCESSES
ARC0: Archival started
STARTING ARCH PROCESSES COMPLETE
Becoming a 'no FAL' ARCH
Gap Manager starting
STARTING ARCH PROCESSES
ARC1: Archival started
ARC2: Archival started
ARC3: Archival started
STARTING ARCH PROCESSES COMPLETE

MESSAGE
--------------------------------------------------------------------------------
Archiving previously deferred ORLs
Managed Standby Recovery not using Real Time Apply
Media Recovery Complete
Primary database is in MAXIMUM PERFORMANCE mode
Selected LNO:11 for T-1.S-30 dbid 466914583 branch 1055076250
Attempt to start background Managed Standby Recovery process
Background Managed Standby Recovery process started
Managed Standby Recovery starting Real Time Apply
Media Recovery Waiting for T-1.S-30 (in transit)

20 rows selected.
```

Start a second session, connect to the Primary database and issue a log switch:

```
[oracle@rob01db01 ~]$ . oraenv
ORACLE_SID = [prod_db2] ? prod_db1
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@rob01db01 ~]$ sqlplus / as sysdba

SQL> alter system switch logfile;

System altered.

SQL> select * from (select SEQUENCE#, ARCHIVED, APPLIED, to_char(NEXT_TIME,'DD-MON-YYYY HH24:MI:SS')
NEXT_TIME from v$archived_log order by sequence# desc) where rownum < 11;

 SEQUENCE# ARC APPLIED   NEXT_TIME
---------- --- --------- ----------------------------
        30 YES YES       30-OCT-2020 21:01:26
. . . .
. . . .
```

On the first session, which is connected to the Standby, issue the following statements to validate:

```
SQL> select message from v$dataguard_status;

. . . .
. . . .
Re-archiving LNO:11 T-1.S-30
Selected LNO:12 for T-1.S-31 dbid 466914583 branch 1055076250
Completed archiving T-1.S-30 (SCN:0x0000000000000000-SCN:0x0000000000000000)

26 rows selected.

SQL> select * from (select SEQUENCE#, ARCHIVED, APPLIED, to_char(NEXT_TIME,'DD-MON-YYYY HH24:MI:SS')
NEXT_TIME from v$archived_log order by sequence# desc) where rownum < 11;


 SEQUENCE# ARC APPLIED   NEXT_TIME
---------- --- --------- ----------------------------
        30 YES YES       30-OCT-2020 21:01:26
        29 YES YES       30-OCT-2020 20:07:54
        28 YES YES       30-OCT-2020 20:07:54
```

2.4. Configure Active Dataguard

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

Database altered.

SQL> shutdown immediate
ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.

SQL> startup mount
ORACLE instance started.

Total System Global Area 1258287344 bytes
Fixed Size                  9134320 bytes
Variable Size             318767104 bytes
Database Buffers          922746880 bytes
Redo Buffers                7639040 bytes
Database mounted.

SQL> alter database open read only;

Database altered.

SQL> alter database recover managed standby database disconnect from session;

Database altered.

SQL> SELECT database_role, open_mode FROM v$database;

DATABASE_ROLE    OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY READ ONLY WITH APPLY
```

We need to perform a few additional steps to also open the PDBs inside the container in read only mode:

```
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           MOUNTED
         4 PDB2                           MOUNTED

SQL> alter pluggable database all read only;

Pluggable database altered.

SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ ONLY  NO
         4 PDB2                           READ ONLY  NO

SQL> col name format a15
SQL> SELECT name, open_mode, recovery_status FROM v$pdbs ORDER BY 1;

NAME            OPEN_MODE  RECOVERY
--------------- ---------- --------
PDB$SEED        READ ONLY  ENABLED
PDB1            READ ONLY  ENABLED
PDB2            READ ONLY  ENABLED
```

3. Configure the Dataguard Broker Interface

3.1. Prepare the databases for the Broker Configuration

Perform this operation on both the Primary and the Standby Database:

```
SQL> show parameter dg_broker

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
dg_broker_config_file1               string      /u01/app/oracle/homes/OraDB19H
                                                 ome1/dbs/dr1prod_db1.dat
dg_broker_config_file2               string      /u01/app/oracle/homes/OraDB19H
                                                 ome1/dbs/dr2prod_db1.dat
dg_broker_start                      boolean     FALSE

SQL> alter system set dg_broker_start=true;

System altered.

SQL> alter system set log_archive_dest_2 = '';

System altered.
```

Remark: I need to reset the log_archive_dest_2 parameter because it will be configured by the broker. If I do not reset it first, I will receive this error when creating the broker configuration:
`Error: ORA-16698: member has a LOG_ARCHIVE_DEST_n parameter with SERVICE attribute set`

3.2. Create the Broker Configuration

```
[oracle@rob01db01 ~]$ . oreanv
ORACLE_SID = [prod_db1] ? prod_db1

[oracle@rob01db01 ~]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Sat Oct 31 08:09:18 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db1"
Connected as SYSDG.

DGMGRL> create configuration 'DG_PROD' as primary database is 'prod_db1' connect identifier is
prod_db1;
Configuration "DG_PROD" created with primary database "prod_db1"

DGMGRL> add database "prod_db2" as connect identifier is 'prod_db2' maintained as physical;
Database "prod_db2" added

DGMGRL> enable configuration;
Enabled.
```

Now wait a few moments for the configuration to synchronize and then check the status:

```
DGMGRL> show configuration

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 15 seconds ago)
```

```
DGMGRL> show database "prod_db1";

Database - prod_db1

  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
    PROD_DB1

Database Status:
SUCCESS
```

```
DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: 3.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

3.3. Perform a SWITCHOVER with the Broker Interface

3.3.1. Log on to the Broker Interface with a password

When you perform a switchover or failover operation it is important to log on with a password. Otherwise the Broker can not connect during the switchover operation. So don't log on as / !!

```
[oracle@rob01db01 admin]$ dgmgrl
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Sat Oct 31 09:53:36 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys/oracle
Connected to "prod_db2"
Connected as SYSDBA.
```

3.3.2. Validate the configuration before the switchover

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 44 seconds ago)

DGMGRL> show database "prod_db1";

Database - prod_db1

  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
    PROD_DB1

Database Status:
SUCCESS

DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: 1.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

```
DGMGRL> validate database "prod_db1";

  Database Role:    Primary database

  Ready for Switchover:  Yes

  Managed by Clusterware:
    prod_db1:  NO
    Validating static connect identifier for the primary database prod_db1...
    The static connect identifier allows for a connection to database "prod_db1".

DGMGRL> validate database "prod_db2";

  Database Role:     Physical standby database
  Primary Database:  prod_db1

  Ready for Switchover:  Yes
  Ready for Failover:    Yes (Primary Running)

  Managed by Clusterware:
    prod_db1:  NO
    prod_db2:  NO
    Validating static connect identifier for the primary database prod_db1...
    The static connect identifier allows for a connection to database "prod_db1".

  Log Files Cleared:
    prod_db1 Standby Redo Log Files:  Cleared
    prod_db2 Online Redo Log Files:   Not Cleared
    prod_db2 Standby Redo Log Files:  Available

DGMGRL> validate network configuration for all;

Connecting to instance "PROD_DB1" on database "prod_db1" ...
Connected to "prod_db1"
Checking connectivity from instance "PROD_DB1" on database "prod_db1 to instance "prod_db2" on database
"prod_db2"...
Succeeded.
Connecting to instance "prod_db2" on database "prod_db2" ...
Connected to "prod_db2"
Checking connectivity from instance "prod_db2" on database "prod_db2 to instance "PROD_DB1" on database
"prod_db1"...
Succeeded.

Oracle Clusterware is not configured on database "prod_db1".
Connecting to database "prod_db1" using static connect identifier
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=
prod_db1_DGMGRL.robdomain)(INSTANCE_NAME=prod_db1)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
The static connect identifier allows for a connection to database "prod_db1".

Oracle Clusterware is not configured on database "prod_db2".
Connecting to database "prod_db2" using static connect identifier
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1522))(CONNECT_DATA=(SERVICE_NAME=
prod_db2_DGMGRL.robdomain)(INSTANCE_NAME=prod_db2)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
The static connect identifier allows for a connection to database "prod_db2".
```

remark: with the “validate database **verbose** “prod_db2” more information is displayed.

So, all the checks are done and I am ready for the switchover

3.3.3. Perform the Switchover

Before I perform the switchover, I will first test a 12cR2 new feature that allows for active sessions that are connected to the Standby to remain connected, even during a switchover.

Start a seperate session to the Standby Database and log on. Take into account that local connections are not preserved, only connections that are made via SQL*Net !

```
[oracle@rob01db01 trace]$ echo $ORACLE_SID
prod_db2
[oracle@rob01db01 trace]$ sqlplus sys/oracle@prod_db2 as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Nov 2 19:24:42 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL>  select name, database_role, open_mode from v$database;

NAME      DATABASE_ROLE    OPEN_MODE
--------- ---------------- --------------------
PROD      PHYSICAL STANDBY READ ONLY WITH APPLY

SQL> show parameter standby_db_preserve_states

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
standby_db_preserve_states           string      ALL
```

This feature depends on database parameter standby_db_preserve_states, which needs to be set at the Standby database. This is a non dynamic parameter and the default is NONE, but I already set it to ALL.

Return to the other session, with the Broker Interface and perform the switchover.

```
DGMGRL> switchover to "prod_db2";
Performing switchover NOW, please wait...
Operation requires a connection to database "prod_db2"
Connecting ...
Connected to "prod_db2"
Connected as SYSDBA.
New primary database "prod_db2" is opening...
Operation requires start up of instance "PROD_DB1" on database "prod_db1"
Starting instance "PROD_DB1"...
Connected to an idle instance.
ORACLE instance started.
Connected to "prod_db1"
Database mounted.
Database opened.
Connected to "prod_db1"
Connected to "prod_db2"
Switchover succeeded, new primary is "prod_db2"
```

Check the new configuration. We need to exit the DGMGRL interface first, as this connection on the Primary was lost during the switchover. The parameter to maintain sessions ONLY counts for sessions connected to the Standby Database

```
DGMGRL> exit
[oracle@rob01db01 ~]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Mon Nov 2 19:39:52 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db1"
Connected as SYSDG.
DGMGRL> show configuration

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db2 - Primary database
    prod_db1 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 51 seconds ago)

DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db2 - Primary database
    prod_db1 - Physical standby database
      Error: ORA-16810: multiple errors or warnings detected for the member

Fast-Start Failover:  Disabled

Configuration Status:
ERROR   (status updated 36 seconds ago)
```

Now return to the session that was connected to the Standby and verify that you are still logged on:

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

NAME      DATABASE_ROLE    OPEN_MODE
--------- ---------------- --------------------
PROD      PRIMARY          READ WRITE
```

3.3.4. Perform a switchback

This is basically the same as the previous operation, just that we now switchover to prod_db1:

```
DGMGRL> switchover to "prod_db1";
Performing switchover NOW, please wait...
New primary database "prod_db1" is opening...
Operation requires start up of instance "prod_db2" on database "prod_db2"
Starting instance "prod_db2"...
Connected to "prod_db2"
ORACLE instance started.
Connected to "prod_db2"
Database mounted.
Database opened.
Connected to "prod_db2"
Connected to "prod_db1"
Switchover succeeded, new primary is "prod_db1"
```

3.4. Perform a FAILOVER with the Broker Interface

A failover usually occurs when there is no more communication with the Primary and you need to activate the Standby Database. After a failover the former Primary needs to be re-instated to assume the role as the new Standby. Assuming that flashback database is on (an Oracle DG best practise), all these steps can be performed with the Dataguard Broker:

You have to issue the failover command on the On the Standby Database server and Oracle_Home

```
[oracle@rob01db01 trace]$ echo $ORACLE_HOME
/u01/app/oracle/product/19.3.0/dbhome_2
[oracle@rob01db01 trace]$ echo $ORACLE_SID
prod_db2


[oracle@rob01db01 trace]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Tue Nov 3 19:38:23 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db2"
Connected as SYSDBA.

DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 0 seconds ago)
  Average Apply Rate: 1.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

```
DGMGRL> validate database "prod_db2";

  Database Role:     Physical standby database
  Primary Database:  prod_db1

  Ready for Switchover:  Yes
  Ready for Failover:    Yes (Primary Running)

  Managed by Clusterware:
    prod_db1:  NO
    prod_db2:  NO
    Validating static connect identifier for the primary database prod_db1...
    The static connect identifier allows for a connection to database "prod_db1".

  Log Files Cleared:
    prod_db1 Standby Redo Log Files:  Cleared
    prod_db2 Online Redo Log Files:   Not Cleared
    prod_db2 Standby Redo Log Files:  Available

DGMGRL> validate network configuration for "prod_db2";
Connecting to instance "prod_db2" on database "prod_db2" ...
Connected to "prod_db2"
Checking connectivity from instance "prod_db2" on database "prod_db2 to instance "PROD_DB1" on database
"prod_db1"...
Succeeded.
Connecting to instance "PROD_DB1" on database "prod_db1" ...
Connected to "prod_db1"
Checking connectivity from instance "PROD_DB1" on database "prod_db1 to instance "prod_db2" on database
"prod_db2"...
Succeeded.

Oracle Clusterware is not configured on database "prod_db2".
Connecting to database "prod_db2" using static connect identifier
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1522))(CONNECT_DATA=(SERVICE_NAME=
prod_db2_DGMGRL.robdomain)(INSTANCE_NAME=prod_db2)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
The static connect identifier allows for a connection to database "prod_db2".
```

```
DGMGRL> failover to "prod_db2";
Performing failover NOW, please wait...
Failover succeeded, new primary is "prod_db2"
```

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db2 - Primary database
    prod_db1 - Physical standby database (disabled)
      ORA-16661: the standby database needs to be reinstated

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 4 seconds ago)
```

The former Primary needs to be reinstated. This can be done by the Broker, provided that flashback database in ON. During the reinstate using dataguard broker, the failed primary site will be flashbacked and converted as physical standby database, then the media recovery process will be started automatically started. These can be view in the alert log of the database which is to be reinstated. If you do not have configured Flashback Database On, you need to recreate the database as a Standby manually. And because that is a lot of work as you can see in one of the previous labs, it is so much easier to use the reinstate command:

```
DGMGRL> reinstate database "prod_db1";
Reinstating database "prod_db1", please wait...
Operation requires shut down of instance "PROD_DB1" on database "prod_db1"
Shutting down instance "PROD_DB1"...
Connected to "prod_db1"
ORACLE instance shut down.
Operation requires start up of instance "PROD_DB1" on database "prod_db1"
Starting instance "PROD_DB1"...
Connected to an idle instance.
ORACLE instance started.
Connected to "prod_db1"
Database mounted.
Connected to "prod_db1"
Continuing to reinstate database "prod_db1" ...
Reinstatement of database "prod_db1" succeeded
```

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db2 - Primary database
    prod_db1 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 47 seconds ago)

DGMGRL> show database "prod_db1";

Database - prod_db1

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: 133.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    PROD_DB1

Database Status:
SUCCESS
```

In order to re-establish the original situation, I will now perform a switchover operation to make database prod_db1 the Primary Database again:

```
DGMGRL> switchove to "prod_db1";
```

3.5. Create a Snapshot Standby Database

The use of snapshot standby database is in the situation where if we want to clone a production database for testing something, we can convert the existing physical standby database to snapshot standby database which is as close as to the production database w.r.t data, do required testing on the snapshot standby database and convert it back to physical standby database. Redo data will continue to be received by the database while it is operating as a snapshot standby database, but it will not be applied until the snapshot standby is converted back into a physical standby database.

In this example I will use the broker interface to create a snapshot standby database and revert back to a physical database.

```
[oracle@rob01db01 ~]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Mon Nov 2 16:34:36 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db1"
Connected as SYSDBA.

DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 58 seconds ago)

DGMGRL> convert database 'prod_db2' to snapshot standby;
Converting database "prod_db2" to a Snapshot Standby database, please wait...
Database "prod_db2" converted successfully

DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Snapshot standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 45 seconds ago)
```

If I log on to the Standby Database I can see that the database now has a different role:

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

NAME      DATABASE_ROLE    OPEN_MODE
--------- ---------------- --------------------
PROD      SNAPSHOT STANDBY READ WRITE
```

The Standby Database is now ready for testing. DDL and DML can be changed. After the testing is done, the database can be reverted back to a Snapshot Database by the Broker. (what happens under the hood is that the broker creates a guaranteed restore point when the database is converted to Snapshot standby and then and flashes back the database to this restore point when the database is reverted back to Physical Standby:

```
DGMGRL> convert database 'prod_db2' to physical standby;
Converting database "prod_db2" to a Physical Standby database, please wait...
Operation requires shut down of instance "prod_db2" on database "prod_db2"
Shutting down instance "prod_db2"...
Connected to "prod_db2"
Database closed.
Database dismounted.
ORACLE instance shut down.
Operation requires start up of instance "prod_db2" on database "prod_db2"
Starting instance "prod_db2"...
Connected to an idle instance.
ORACLE instance started.
Connected to "prod_db2"
Database mounted.
Connected to "prod_db2"
Continuing to convert database "prod_db2" ...
Database "prod_db2" converted successfully
```

Wait a few minutes and then check the status:

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 11 seconds ago)

DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 0 seconds ago)
  Average Apply Rate: 7.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

4. Create Dataguard TAF services

4.1. Introduction to TAF

Transparent Application Failover (TAF) enables the application to automatically reconnect to a database, if the database instance to which the connection is made fails. In this case, the active transactions roll back. This feature can also be used in a Dataguard Environment.

There are 2 different methods to configure TAF for Dataguard:

- **Basic**: In this method, an application connects to secondary node in case of failure of primary node but it will take time to connect the secondary node till the Failover time.
- **Preconnect**: In this method, an application connects to both primary and secondary node at the same time which is faster than the basic method. Because as it already connects to the secondary node, it reduces the Failover time. But it will consume more resources due to extra connections.

There are 2 different types of TAF for Dataguard

- **Select**: In this type, if the connection is lost then Oracle establishes connection to the secondary node and re-executes the select statements by re-positioning the cursor without knowing the client's application.During the time, any uncommitted transactions will be rolled back.
- **Session**: In this type, if the connection is lost then Oracle establishes connection to the secondary node but does not re-execute the select statements and you will receive below error message.

In this example I will configure and test a Basic TAF service using the select type.

4.2. Configure TAF on the Primary and the Standby Database

4.2.1. Create the TAF Service PROD_PRIMARY

Log on to the Primary Database and create the service

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

DATABASE_ROLE
----------------
PRIMARY

SQL>
BEGIN
   DBMS_SERVICE.CREATE_SERVICE(service_name=>'PROD_PRIMARY',
   network_name=>'PROD_PRIMARY');
END;
/

PL/SQL procedure successfully completed.

SQL>
BEGIN
   dbms_service.modify_service('PROD_PRIMARY',
   failover_method =>'BASIC',
   failover_type =>'SELECT',
   failover_retries =>200,
   failover_delay =>1);
END;
/

PL/SQL procedure successfully completed.
```

Now start the service

```
SQL>
begin
   dbms_service.start_service('PROD_PRIMARY');
end;
/

PL/SQL procedure successfully completed.

SQL> select name from v$active_services order by 1;

NAME
--------------------
PROD_PRIMARY
SYS$BACKGROUND
SYS$USERS
prodXDB
prod_CFG
prod_db1.robdomain

6 rows selected.
```

Now I will create a trigger that will start the service when the database starts up, but only if the database has the role PRIMARY. There are now also role dependant services, which is an alternative mechanism to start the service, but I will use a trigger in this example:

```
SQL> create or replace trigger LOAD_SVC_PROD_PRIMARY after startup on database
declare
   v_role varchar(30);
begin
   select database_role into v_role from v$database;
   if v_role = 'PRIMARY' then
      DBMS_SERVICE.START_SERVICE('PROD_PRIMARY');
else
      DBMS_SERVICE.STOP_SERVICE('PROD_PRIMARY');
end if;
end;
/
commit;

Trigger created.
```

Navigate to the Oracle Home of the Primary database and add the following tns entry:

```
[oracle@rob01db01 ~]$ cd /u01/app/oracle/homes/OraDB19Home1/network/admin
[oracle@rob01db01 admin]$ vi tnsnames.ora

# TAF entry for the dataguard configuration
PROD_PRIMARY =
  (DESCRIPTION_LIST =
      (LOAD_BALANCE= OFF)
      (FAILOVER = ON)
  (DESCRIPTION =
  (SDU = 32767)
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1522))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = prod_primary.robdomain)
    )
  )
)
```

Test the entry:

```
[oracle@rob01db01 admin]$ tnsping prod_primary

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 04-NOV-2020 15:39:37

Copyright (c) 1997, 2019, Oracle.  All rights reserved.

Used parameter files:
/u01/app/oracle/homes/OraDB19Home1/network/admin/sqlnet.ora


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION_LIST = (LOAD_BALANCE= OFF) (FAILOVER = ON) (DESCRIPTION = (SDU =
32767) (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1521)) (ADDRESS = (PROTOCOL =
TCP)(HOST = rob01db01.robdomain)(PORT = 1522)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME =
prod_primary.robdomain))))
OK (0 msec)
[oracle@rob01db01 admin]$ sqlplus sys/oracle@prod_primary as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Nov 4 15:39:46 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT
```

So this connection works. Note that I am connecting to the root container. If you want to connect to a specific PDB, you need to configure the tns entry accordingly and you also need to create and start the service in that specific PDB. Refer to MOS note “HOW TO ADD GLOBAL SERVICE IN MULTITENANT 12C DATABASE AT PDB AND ROOT LEVEL OF CDB IN A GDS ENVIRONMENT (Doc ID 2015403.1)”

Now I also need to create the same TAF entry in the Oracle Home of the Standby Database:

```
[oracle@rob01db01 ~]$ cd /u01/app/oracle/homes/OraDB19Home2/network/admin
[oracle@rob01db01 admin]$ vi tnsnames.ora

# TAF entry for the dataguard configuration
PROD_PRIMARY =
  (DESCRIPTION_LIST =
      (LOAD_BALANCE= OFF)
      (FAILOVER = ON)
  (DESCRIPTION =
  (SDU = 32767)
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1522))
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = prod_primary.robdomain)
    )
  )
)
```

As can be seen in the example above, I switched the order of the port numbers so that in the Standby Home, the first attempt will be to connect to the Standby, rather than the Primary. This is so I can cut out any latency. This may not be relevant in this example, as I am using a single server for my lab, but could make some difference if the Standby Database is hosted in another Data Center.

Test the TNS entry:

```
[oracle@rob01db01 admin]$ . oraenv
ORACLE_SID = [prod_db1] ? prod_db2
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@rob01db01 admin]$ tnsping prod_primary

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 04-NOV-2020 15:48:45

Copyright (c) 1997, 2019, Oracle.  All rights reserved.

Used parameter files:


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION_LIST = (LOAD_BALANCE= OFF) (FAILOVER = ON) (DESCRIPTION = (SDU =
32767) (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1522)) (ADDRESS = (PROTOCOL =
TCP)(HOST = rob01db01.robdomain)(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME =
prod_primary.robdomain))))
OK (10 msec)
[oracle@rob01db01 admin]$ sqlplus sys/oracle@prod_primary as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Nov 4 15:48:59 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

4.3. Test the TAF configuration during a switchover operation

To test it properly I first create a little sleep function:

```
SQL> create or replace function f_sleep( in_time number ) return number is
begin
 dbms_lock.sleep(in_time);
 return 1;
end;
/

Function created.
```

To run a query that lasts for 5 minutes, I can run the following:

```
SQL> select f_sleep(300) from dual;

F_SLEEP(300)
------------
           1

Elapsed: 00:05:00.28
```

Now that I have the “long running” query in place, I can start the TAF test:

Start a first session and connect to the Broker and verify that everything is ready for a Switchover:

Session 1:

```
[oracle@rob01db01 admin]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Nov 4 20:05:36 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db2"
Connected as SYSDBA.

DGMGRL> validate database "prod_db2";

  Database Role:     Physical standby database
  Primary Database:  prod_db1

  Ready for Switchover:  Yes
  Ready for Failover:    Yes (Primary Running)

  Managed by Clusterware:
    prod_db1:  NO
    prod_db2:  NO
    Validating static connect identifier for the primary database prod_db1...
    The static connect identifier allows for a connection to database "prod_db1".

  Log Files Cleared:
    prod_db1 Standby Redo Log Files:  Cleared
    prod_db2 Online Redo Log Files:   Not Cleared
    prod_db2 Standby Redo Log Files:  Available
```

Start a second session via the TAF service and connect to the Primary:

Session 2:

```
[oracle@rob01db01 trace]$ sqlplus sys/oracle@prod_primary as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Nov 4 19:53:19 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY

SQL> show parameter db_unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      prod_db1
```

So, I can see that this session is connected to the Primary Database, which has db_unique_name **prod_db1**

Now I will start my long running query in this session

Session 2:

```
SQL> select f_sleep(300) from dual;
```

Now I will switch to session 1 and perform the switchover:

Session 1:

```
DGMGRL> switchover to "prod_db2";
Performing switchover NOW, please wait...
New primary database "prod_db2" is opening...
Operation requires start up of instance "PROD_DB1" on database "prod_db1"
Starting instance "PROD_DB1"...
Connected to "prod_db1"
ORACLE instance started.
Connected to "prod_db1"
Database mounted.
Database opened.
Connected to "prod_db1"
Connected to "prod_db2"
Switchover succeeded, new primary is "prod_db2"
```

The switchover was succesfull. Let's see how my query is doing in session 2

Session 2: wait 5 minutes and confirm that the query returns with the result:

```
SQL> select f_sleep(300) from dual; --this query is still running during the switchover

F_SLEEP(300)
------------
           1

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY

SQL> show parameter db_unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      prod_db2
```

So, I get my query result back fter 5 minutes and I can also so that **prod_db2** now is the Primary Database.

It works like magic (=:

Let's perform another switchover to go back to the original situation:

```
DGMGRL> switchover to "prod_db1";
Performing switchover NOW, please wait...
New primary database "prod_db1" is opening...
Operation requires start up of instance "PROD_DB2" on database "prod_db2"
Starting instance "PROD_DB2"...
Connected to "prod_d21"
ORACLE instance started.
Connected to "prod_db2"
Database mounted.
Database opened.
Connected to "prod_db2"
Connected to "prod_db1"
Switchover succeeded, new primary is "prod_db1"
```

5. Configure Fast Start Failover

5.1. Introduction to fast start failover

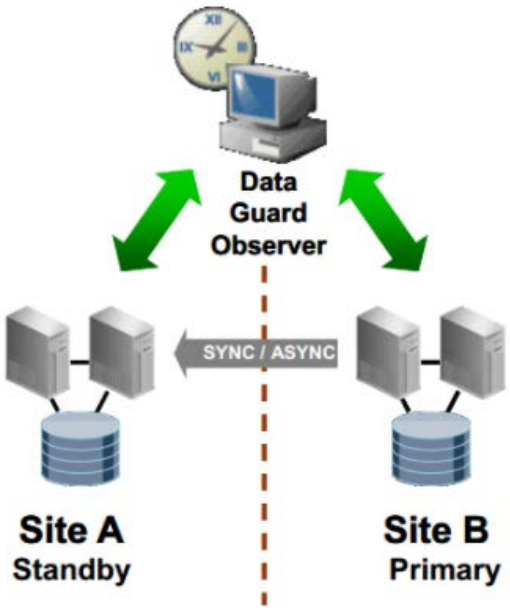
This feature increases the availability of the database by eliminating the need for DBA involvement as part of the failover process. This should only be implemented if any switchover or failover is completely automated including the corresponding applications and network services.

Fast-Start Failover enables the broker to perform these tasks automatically failover to standby database
The key to this feature is a monitoring process appropriately named the Observer. The Observer is a component of the DGMGRL interface that is configured on a system outside the systems actually running the Oracle Data Guard configuration, which monitors the availability of the primary database. it will issue a failover after waiting the number of seconds specified by the FastStartFailoverThreshold property.

You can configure the conditions for an automatic failover to occur. For example if certain Oracle errors occur in the alert file. By default, fast start failover occurs under the following conditions:

- Datafile Offline Failover is initiated if a datafile on the primary database experiences an I/O error resulting in a datafile being taken offline. This option is enabled by default.
- Corrupted Dictionary Failover is initiated if corruption of a critical database object is found. This option is enabled by default.
- Corrupted Controlfile Enabled by default, the detection of controlfile corruption will result in immediate failover.
- Inaccessible Log File This parameter, disabled by default, allows for failover to be initiated in the event that LGWR is unable to write to a member of a log group.
- Stuck Archiver Failover is initiated should the archiver on the primary database become hung. The default setting of this parameter is disabled.

The Observer shoud ideally be located in a remote 3rd location. See the architecture below:



In this Lab, since I have only 1 Virtual Server available, hosting both the Primary and the Standby database, I will configure the Observer on that same Virtual Server.

5.2. Configure Fast Start Failover

As stated before, you should ideally run the Observer in a third location, seperated from the Primary and the Standby Database, to avoid a split brain situation. But because I only have one virtual server available, I will run it on that server.

5.2.1. Enable Fast Start Failover and start the Observer

Set the environment to the Primary Database and start the Observer:

```
[oracle@rob01db01 trace]$ . oraenv
ORACLE_SID = [prod_db1] ? prod_db1
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@rob01db01 trace]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Sat Nov 7 11:36:44 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db1"
Connected as SYSDG.

DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 50 seconds ago)

DGMGRL> enable fast_start failover;
Enabled in Potential Data Loss Mode.
```

The warning refers to the fact that the Protection Mode is MaxPerformance. In order to have a 0 data loss mode, you need to have the configuration running in the Maximum Availability Mode. I will leave this for now. (In the lab about Far Syn Instance I will configure the Maximum Availability Mode).

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    Warning: ORA-16819: fast-start failover observer not started

    prod_db2 - (*) Physical standby database
      Warning: ORA-16819: fast-start failover observer not started

Fast-Start Failover: Enabled in Potential Data Loss Mode

Configuration Status:
WARNING   (status updated 46 seconds ago)
```

The next step will be to start the Observer.

```
DGMGRL> start observer logfile IS /home/oracle/observer.log
[W000 2020-11-07T14:03:00.062+00:00] FSFO target standby is prod_db2
Observer 'rob01db01.robdomain' started
[W000 2020-11-07T14:03:00.234+00:00] Observer trace level is set to USER
[W000 2020-11-07T14:03:00.234+00:00] Try to connect to the primary.
[W000 2020-11-07T14:03:00.234+00:00] Try to connect to the primary prod_db1.
[W000 2020-11-07T14:03:00.255+00:00] The standby prod_db2 is ready to be a FSFO target
[W000 2020-11-07T14:03:01.256+00:00] Connection to the primary restored!
[W000 2020-11-07T14:03:03.258+00:00] Disconnecting from database prod_db1.
```

Remark: it is imporant to log on with a password to start the Observer. Do not log on as /

I need to use CTRL+C to leave the session as it is stuck after starting the observer.

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - (*) Physical standby database

Fast-Start Failover: Enabled in Potential Data Loss Mode

Configuration Status:
SUCCESS   (status updated 59 seconds ago)

DGMGRL> show fast_start failover;

Fast-Start Failover: Enabled in Potential Data Loss Mode

  Protection Mode:    MaxPerformance
  Lag Limit:          30 seconds

  Threshold:          30 seconds
  Active Target:      prod_db2
  Potential Targets:  "prod_db2"
    prod_db2   valid
  Observer:           rob01db01.robdomain
  Shutdown Primary:   TRUE
  Auto-reinstate:     TRUE
  Observer Reconnect: (none)
  Observer Override:  FALSE

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

  Oracle Error Conditions:
    (none)
```

```
DGMGRL> show observer;

Configuration - DG_PROD

  Primary:            prod_db1
  Active Target:      prod_db2

Observer "rob01db01.robdomain" - Master

  Host Name:                    rob01db01.robdomain
  Last Ping to Primary:         3 seconds ago
  Last Ping to Target:          3 seconds ago
```

You can also check the status of the Observer in the database. These results should be returned for both the Primary as well as the Standby database.

```
SQL> select FS_FAILOVER_STATUS,FS_FAILOVER_CURRENT_TARGET, FS_FAILOVER_THRESHOLD,
FS_FAILOVER_OBSERVER_PRESENT  from v$database;

FS_FAILOVER_STATUS     FS_FAILOVER_CURRENT_TARGET     FS_FAILOVER_THRESHOLD FS_FAIL
---------------------- ------------------------------ --------------------- -------
TARGET UNDER LAG LIMIT prod_db2                                         30 YES
```

Finally, let's have a look at the observer log file. You can do a tail -f on that file to see what the Observer is doing. You would usually not find much in this file, as the Observer only writes to it when it starts to do work, as I will show in the next paragraph, when I force a failover by the Observer.

```
[oracle@rob01db01 ~]$ tail -f observer.log
Observer 'rob01db01.robdomain' started
[W000 2020-11-07T14:50:08.106+00:00] Observer trace level is set to USER
[W000 2020-11-07T14:50:08.106+00:00] Try to connect to the primary.
[W000 2020-11-07T14:50:08.106+00:00] Try to connect to the primary prod_db1.
[W000 2020-11-07T14:50:08.123+00:00] The standby prod_db2 is ready to be a FSFO target
[W000 2020-11-07T14:50:09.123+00:00] Connection to the primary restored!
[W000 2020-11-07T14:50:11.125+00:00] Disconnecting from database prod_db1.
```

Keep this tail -f session on the observer log file open as it will be handy to follow the forced fast start failover in the next paragraph.

Remark: you can stop the observer as follows: DGMGRL> stop observer;

5.3. Test Fast Start Failover

Check the current Observer status:

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - (*) Physical standby database

Fast-Start Failover: Enabled in Potential Data Loss Mode

Configuration Status:
SUCCESS   (status updated 33 seconds ago)

DGMGRL> show observer;

Configuration - DG_PROD

  Primary:            prod_db1
  Active Target:      prod_db2

Observer "rob01db01.robdomain" - Master

  Host Name:                    rob01db01.robdomain
  Last Ping to Primary:         26 seconds ago
  Last Ping to Target:          26 seconds ago

DGMGRL> show fast_start failover;

Fast-Start Failover: Enabled in Potential Data Loss Mode

  Protection Mode:    MaxPerformance
  Lag Limit:          30 seconds

  Threshold:          30 seconds
  Active Target:      prod_db2
  Potential Targets:  "prod_db2"
    prod_db2   valid
  Observer:           rob01db01.robdomain
  Shutdown Primary:   TRUE
  Auto-reinstate:     TRUE
  Observer Reconnect: (none)
  Observer Override:  FALSE

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

  Oracle Error Conditions:
    (none)
```

In the listing above you cans ee that the database will be failed over automatically as soon as there are Datafile Write errors (one of the conditions). It is also possible to add specific ORA-errors to trigger a a fast start failover, but in this case I will just force some datafile write errors to see what happens.

Kill the smon process of the Primary Database:

```
[oracle@rob01db01 ~]$ ps -ef|grep smon
oracle    7967  5601  0 16:25 pts/1    00:00:00 grep --color=auto smon
oracle   20462     1  0 02:35 ?        00:00:01 ora_smon_prod_db1
oracle   22795     1  0 11:50 ?        00:00:00 ora_smon_prod_db2

[oracle@rob01db01 ~]$ kill -9 20462
```

Let's have a look at the Standby Database to see what is going on:

```
[oracle@rob01db01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Sat Nov 7 16:30:32 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> select database_role, open_mode from v$database;

DATABASE_ROLE    OPEN_MODE
---------------- --------------------
PRIMARY          READ WRITE
```

The standby database has now become the Primary Database.

I can also see in the /home/oracle/observer.log file what took place:

```
. . .
Unable to connect to database using prod_db1
[W000 2020-11-07T16:26:40.550+00:00] Primary database cannot be reached.
[W000 2020-11-07T16:26:40.550+00:00] Fast-Start Failover threshold has not exceeded. Retry for the next
2 seconds
[W000 2020-11-07T16:26:41.562+00:00] Try to connect to the primary.
[W000 2020-11-07T16:26:42.693+00:00] Primary database cannot be reached.
[W000 2020-11-07T16:26:42.693+00:00] Fast-Start Failover threshold has expired.
[W000 2020-11-07T16:26:42.693+00:00] Try to connect to the standby.
[W000 2020-11-07T16:26:42.693+00:00] Making a last connection attempt to primary database before
proceeding with Fast-Start Failover.
[W000 2020-11-07T16:26:42.693+00:00] Check if the standby is ready for failover.
[S002 2020-11-07T16:26:42.725+00:00] Fast-Start Failover started...

2020-11-07T16:26:42.725+00:00
Initiating Fast-Start Failover to database "prod_db2"...
[S002 2020-11-07T16:26:42.727+00:00] Initiating Fast-start Failover.
Performing failover NOW, please wait...
Failover succeeded, new primary is "prod_db2"
2020-11-07T16:27:16.782+00:00
[S002 2020-11-07T16:27:16.782+00:00] Fast-Start Failover finished...
. . .
```

Let's now re-instate the original situation:

Startup the former primary prod_db1 in the mount mode:

```
[oracle@rob01db01 ~]$ . oraenv
ORACLE_SID = [prod_db2] ? prod_db1
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@rob01db01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Sat Nov 7 17:50:14 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup mount;
ORACLE instance started.

Total System Global Area 1258287344 bytes
Fixed Size                  9134320 bytes
Variable Size             520093696 bytes
Database Buffers          721420288 bytes
Redo Buffers                7639040 bytes
Database mounted.
```

```
[oracle@rob01db01 ~]$ . oraenv
ORACLE_SID = [prod_db2] ? prod_db2
The Oracle base remains unchanged with value /u01/app/oracle

[oracle@rob01db01 ~]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Sat Nov 7 17:52:14 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "PROD_DB2"
Connected as SYSDBA.

DGMGRL> stop observer;
Observer stopped.
D
GMGRL> disable fast_start failover;
Disabled.

DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db2 - Primary database
    prod_db1 - Physical standby database (disabled)
      ORA-16661: the standby database needs to be reinstated

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 50 seconds ago)

DGMGRL> reinstate database "prod_db1";
```

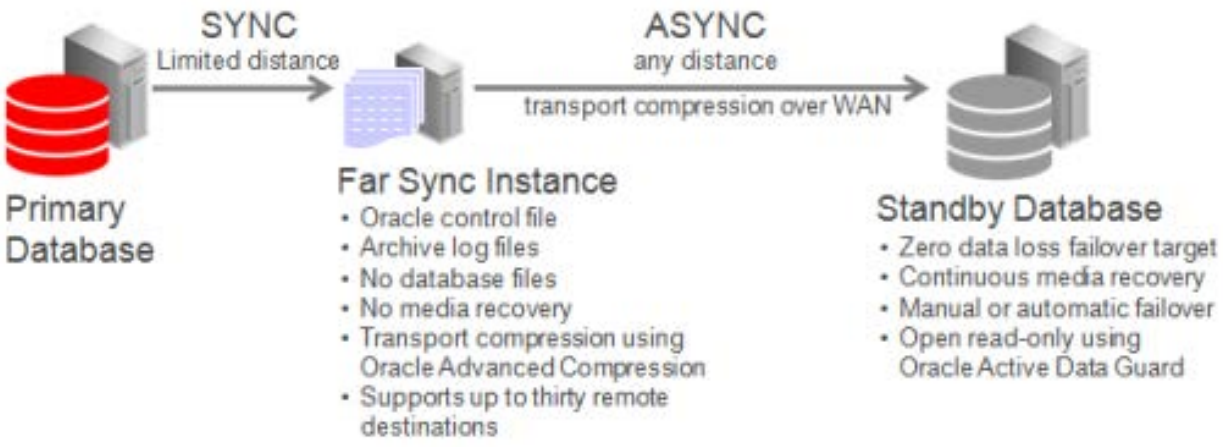
6. Configure a Far Sync Instance

6.1. Introduction

Oracle Dataguard Far Sync instance is a new feature introduced in Oracle Database 12c, which simply works like a archive/redo repeater. Far sync instance is something different than the Oracle Database instance, as its mission is accepting redo from the primary database and then shipping that redo to other members of the Oracle Data Guard configuration. Far Sync instance is an instance without a database..

In Maximum Availability mode, Sync transport method is used between primary and far sync instances.
In this configuration, far sync instance is placed close to the primary instance to reduce the network latency and to minimize the application latency caused by sync transport..

So, by the use of the sync transport mechanism, the redo information are transferred to the Far sync instance site with zero data loss. Once the redo is transferred to the Far Sync site, the Async method is used to transfer the redo to the Standby Site.



In this lab I will use Oracle 19c to create the Far Sync instance. However, starting with Oracke 20c, Oracle now can create a far sync instance for us and also add it in the broker configuration. For more information refer to https://blog.dbi-services.com/oracle-20c-create-a-far-sync-instance-is-now-easy/

Because of limited resources, I will be running the Far Sync instance out of the Primary Database Home, but normally, you would host the Far Sync instance on a remote server and possibly even a remote (but close by) location.

6.2. Create a Far Sync instance

Step 1: create a control file for the Far Sync instance. Log on to the primary and execute:

```
SQL> ALTER DATABASE CREATE FAR SYNC INSTANCE CONTROLFILE AS '/home/oracle/prod_fsi1.ctl';

Database altered.
```

Step 2: copy the far password file and the controlfile to the Far Sync instance home
(I am using the new Read Only Oracle Home structures)

```
[oracle@rob01db01 oracle]$ cd $ORACLE_BASE/dbs
[oracle@rob01db01 dbs]$ ls
hc_prod.dat      hc_prod_db2.dat   initprod.ora  lkPROD_DB1  orapwprod_db1  sample_config.cfg
snapcf_prod_db2.f  spfileprod_db1.ora  u01
hc_prod_db1.dat  initprod_db2.ora  lkPROD        lkPROD_DB2  orapwprod_db2  snapcf_prod_db1.f
snapcf_prod.f      spfileprod_db2.ora

[oracle@rob01db01 dbs]$ cp $HOME/prod_fsi1.ctl /u01/app/oracle/dbs/prod_fsi1.ctl
[oracle@rob01db01 dbs]$ cp orapwprod_db1 orapwprod_fsi1
```

Step 3: Create init.ora file initprod_fsi1.ora for the Far Sync instance. Use the following entires. Entries highlighted in yellow are changed in relation to the Primary instance parameters

```
[oracle@rob01db01 dbs]$ vi /u01/app/oracle/dbs/initprod_fsi1.ora
```

```
audit_file_dest='/u01/app/oracle/admin/prod_fsi1/adump'
audit_trail='db'
compatible='19.0.0'
control_files='/u01/app/oracle/oradata/prod_fsi/prod_fsi1.ctl'
db_block_size=8192
db_domain='robdomain'
db_name='prod'
dg_broker_start=TRUE
instance_name = 'prod_fsi1'
db_unique_name='prod_fsi1'
db_recovery_file_dest='/media/sf_Shared/prod_fsi1'
db_recovery_file_dest_size=4800m
diagnostic_dest='/u01/app/oracle'
dispatchers='(PROTOCOL=TCP) (SERVICE=prod_fsi1.fXDB)'
log_archive_dest_1='location=use_db_recovery_file_dest valid_for=(all_logfiles,all_roles)
db_unique_name=prod_fsi1'
log_archive_config='DG_CONFIG=(prod_db1,prod_db2,prod_fsi1)'
open_cursors=300
pga_aggregate_target=200m
processes=300
remote_login_passwordfile='EXCLUSIVE'
sga_target=1100m
undo_tablespace='UNDOTBS1'
fal_server='prod_db2'
db_file_name_convert='PROD_DB1','PROD_FSI1'
log_file_name_convert='PROD_DB1','PROD_FSI1'
```

Step 4: create the required directories and add the far sync instance to the oratab file

```
[oracle@rob01db01 dbs]$ mkdir -p /u01/app/oracle/admin/prod_fsi1/adump
[oracle@rob01db01 dbs]$ mkdir -p u01/app/oracle/oradata/prod_fsi1
[oracle@rob01db01 dbs]$ mkdir -p /media/sf_Shared/prod_fsi1

[oracle@rob01db01 dbs]$ vi /etc/oratab
prod_db1:/u01/app/oracle/product/19.3.0/dbhome_1:N
prod_db2:/u01/app/oracle/product/19.3.0/dbhome_2:N
prod_fsi1:/u01/app/oracle/product/19.3.0/dbhome_1:N
```

Step 5: Configure SQL*Net for the Far Sync Instance

Update tnsnames.ora in both the Primary and the Standby Home and add the following entry:

```
[oracle@rob01db01 admin]$ vi /u01/app/oracle/homes/OraDB19Home1/network/admin/tnsnames.ora

[oracle@rob01db01 admin]$ vi /u01/app/oracle/homes/OraDB19Home2/network/admin/tnsnames.ora
```

```
# FSI entry
PROD_FSI1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = prod_fsi1.robdomain)
      (UR=A)
    )
  )
```

```
[oracle@rob01db01 dbs]$ tnsping prod_fsi1

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 09-NOV-2020 14:37:36

Copyright (c) 1997, 2019, Oracle.  All rights reserved.

Used parameter files:
/u01/app/oracle/homes/OraDB19Home1/network/admin/sqlnet.ora


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = rob01db01.robdomain)(PORT =
1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = prod_fsi1.robdomain) (UR=A)))
OK (0 msec)
```

Add a static entry to the listener1 listener and stop and start the listener for the changes to take effect. (Remember that I am running the Far Sync Instance out of the Primary Database home in this lab. You would normally have it installed in a separate Oracle Home on a separate machine)

```
[oracle@rob01db01 admin]$ vi /u01/app/oracle/homes/OraDB19Home1/network/admin/listener.ora
# static entries for dataguard configuration
SID_LIST_LISTENER1 =
  (SID_LIST =
     (SID_DESC =
      (SID_NAME = prod_db1)
      (GLOBAL_DBNAME=prod_db1_DGMGRL.robdomain)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
    )
    (SID_DESC =
      (SID_NAME = prod_db1)
      (GLOBAL_DBNAME=prod_db1.robdomain)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
    )
    (SID_DESC =
      (SID_NAME = prod_fsi1)
      (GLOBAL_DBNAME=prod_fsi1.robdomain)
      (ORACLE_HOME = /u01/app/oracle/product/19.3.0/dbhome_1)
    )
  )

[oracle@rob01db01 dbs]$ lsnrctl stop listener1

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 09-NOV-2020 14:42:08

Copyright (c) 1991, 2019, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1521)))
The command completed successfully
[oracle@rob01db01 dbs]$ lsnrctl start listener1

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 09-NOV-2020 14:42:13

Copyright (c) 1991, 2019, Oracle.  All rights reserved.

Starting /u01/app/oracle/product/19.3.0/dbhome_1/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is /u01/app/oracle/homes/OraDB19Home1/network/admin/listener.ora
Log messages written to /u01/app/oracle/diag/tnslsnr/rob01db01/listener1/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rob01db01.robdomain)(PORT=1521)))
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     listener1
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
Start Date                09-NOV-2020 14:42:13
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/oracle/homes/OraDB19Home1/network/admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/rob01db01/listener1/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rob01db01.robdomain)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Services Summary...
Service "prod_db1.robdomain" has 1 instance(s).
  Instance "prod_db1", status UNKNOWN, has 1 handler(s) for this service...
Service "prod_db1_DGMGRL.robdomain" has 1 instance(s).
  Instance "prod_db1", status UNKNOWN, has 1 handler(s) for this service...
Service "prod_fsi1.robdomain" has 1 instance(s).
  Instance "prod_fsi1", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

Step 5: create an spfile and start the instance in no mount mode:

```
[oracle@rob01db01 oradata]$ . oraenv
ORACLE_SID = [prod_db2] ? prod_fsi1
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@rob01db01 oradata]$ echo $ORACLE_SID
prod_fsi1
[oracle@rob01db01 oradata]$ echo $ORACLE_HOME
/u01/app/oracle/product/19.3.0/dbhome_1
[oracle@rob01db01 oradata]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Nov 9 10:50:30 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> create spfile from pfile;

File created.

SQL> startup nomount;
ORACLE instance started.

Total System Global Area 1157627168 bytes
Fixed Size                  8895776 bytes
Variable Size             301989888 bytes
Database Buffers          838860800 bytes
Redo Buffers                7880704 bytes

SQL> alter database mount;

Database altered.

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
FAR SYNC
```

Now change the parameter below in both the Primary and the Standby database:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='DG_CONFIG=(prod_db1,prod_db2,prod_fsi1)';

System altered.
```

Check connectivity:

```
[oracle@rob01db01 oradata]$ sqlplus sys/oracle@prod_fsi1 as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Nov 9 15:12:52 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

6.3. Add the Far Syn Instance to the Broker Configuration

```
[oracle@rob01db01 oradata]$ . oraenv
ORACLE_SID = [prod_fsi1] ? prod_db1
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@rob01db01 oradata]$ dgmgrl sys/oracle
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Mon Nov 9 15:19:07 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db1"
Connected as SYSDBA.
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 45 seconds ago)

DGMGRL> ADD FAR_SYNC PROD_FSI1 AS CONNECT IDENTIFIER IS prod_fsi1;
far sync instance "prod_fsi1" added

DGMGRL> enable far_sync "prod_fsi1";
Enabled.

DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1  - Primary database
    prod_db2  - Physical standby database
    prod_fsi1 - Far sync instance

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 34 seconds ago)
```

I will now switch the protection mode from Maximum Performance to Maximum Availability:

```
DGMGRL> EDIT DATABASE 'prod_db1' SET PROPERTY 'RedoRoutes' = '(LOCAL : prod_fsi1 SYNC)';
Property "RedoRoutes" updated

DGMGRL> EDIT FAR_SYNC 'prod_fsi1' SET PROPERTY 'RedoRoutes' = '(prod_db1 : prod_db2 ASYNC)';
Property "RedoRoutes" updated

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

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxAvailability
  Members:
  prod_db1  - Primary database
     prod_fsi1 - Far sync instance
      prod_db2  - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS  (status updated 45 seconds ago)
```

Remark: To ensure that maximum availability protection mode can be maintained when prod_db2 is the primary database, after a switchover or a failover, add a second far sync instance to the configuration so that prod_db2 can send redo in synchronous mode which in turn will send redo to the new terminal database, prod_db1 after the role transition.

7. Automatic Block Corruption Detection and Repair

7.1. Introduction to lost writes

Reference for this chapter:
https://externaltable.blogspot.com/2013/03/testing-lost-writes-in-oracle-and-with.html

If a corrupt data block is encountered when a primary database is accessed, it is automatically replaced with an uncorrupted copy of that block from a physical standby database. This requires the following conditions:

- The physical standby database must be operating in real-time query mode, which requires an Active Data Guard license.
- The physical standby database must be running real-time apply.

Before I continue with an example, I will provide a little background information:
Lost writes: "A data block lost write occurs when an I/O subsystem acknowledges the completion of the block write, while in fact the write did not occur in the persistent storage" (from support note 1302539.1). That means, a write request to the disk was commited an the database is happy with that. But the write did not actually happen for whatever reason. So when the block will be read the next time, it is still in old state, any changed, deleted or added values are not included. The block itself is consistent, it is not corrupted. The DBA will not notice it since there is no error. An error will occur only when you restore the tablespace containing the block and then try to apply the redo stream.

Lost writes can be caused by faulty storage, but also by Oracle bugs, in general anything in between our data in RAM and storage can corrupt our data, including controllers and network. Lost writes can cause widespread database corruption if not propertly protected. Oracle Active Dataguard can automatically detect and repair lost writes on both the Primary database and the Standby database. In this chapter I will expain this functionality.

The DB_LOST_WRITE_DETECTION in Dataguard can be enabled with the parameter db_lost_write_protect. This parameter forces the database to record the SCN of all blocks that it reads from disk to the redo stream. The standby database can use this information to compare the recorded SCN from the redo stream to the actual SCN of the block at the standby site. If there is a difference, it can report a lost write. Oracle will also automatically repair the corrupt data block. If the corruption is on the Primary Database, Oracle will ship the correct block from the Standby to the Primary and vice versa. This all happens under the hood, but you can follow it in the database alert file. The parameter DB_LOST_WRITE_PROTECT is disabled (NONE) by default.

I will first force a Lost Write when there is no lost write protection and explain the danger of that situation. I will then enable lost write and repeat the exercise.

A useful technique to force a list write is the ability to read and write a single block from Oracle data files (in a test environment). For databases on filesystems (and also DBs on NFS) **dd** is the tool for this job

Examples:

```
read one 8KB block from filesystem (block 134 in this example):
dd if=testlostwrite.dbf bs=8192 count=1 skip=134 of=blk134.dmp
```

```
write one 8KB block to filesystem (block 134 in this example)::
dd of=testlostwrite.dbf bs=8192 count=1 seek=134 if=blk134.dmp conv=notrunc
```

Note when writing to an Oracle datafile we must use conv=notrunc or else we will end up with an unusable (truncated) output file. Note also the syntax for specifying the block offset, skip is used for input files and seek for output files (see dd manual).

7.2. Scenario 1 : Lost Write when DB_LOST_WRITE_PROTECT is disabled

Verify on both the Primary and the Standby that DB_:LOST_WRITE_DETECTION is disabled:

```
SQL> show parameter db_lost_write

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_lost_write_protect                string      NONE
```

Let's now create a test table insert some data

```
$ sqlplus pdbadmin/pbadmin@pdb1

SQL> create tablespace rob_lost_write datafile size 100M;

Tablespace created.

SQL> select name from v$datafile where name like '%rob%';

NAME
--------------------------------------------------------------------------------------------------------------------------------
/u01/app/oracle/oradata/PROD_DB1/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_rob_lost_ht30hw1g_.dbf

SQL> create table testlosttable (id number, payload varchar2(100)) tablespace rob_lost_write;

Table created.

SQL> create index i_testlosttable on testlosttable (id) tablespace rob_lost_write;

Index created.

SQL> insert into testlosttable values (10,'aaaaaaaaaaaaaaaaaaaaaaaa');

1 row created.

SQL> insert into testlosttable values (20,'bbbbbbbbbbbbbbbbbbbbbbbb');

1 row created.

SQL> commit;

Commit complete.

SQL> select rowid, dbms_rowid.ROWID_BLOCK_NUMBER(rowid), a.* from testlosttable a;

ROWID              DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)         ID PAYLOAD
------------------ ------------------------------------ ---------- --------------------------------
AAAR8NAAUAAAACGAAA                                  134         10 aaaaaaaaaaaaaaaaaaaaaaaa
AAAR8NAAUAAAACGAAB                                  134         20 bbbbbbbbbbbbbbbbbbbbbbbb
```

This query shows that the inserted data resides in block ID 134.

Let's save a copy of this block to disk:

```
$ SQL> !dd
if=/u01/app/oracle/oradata/PROD_DB1/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_rob_lost_ht30hw1g_.
dbf bs=8192 count=1 skip=134 of=/home/oracle/blk134_old.dmp
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 0.000231246 s, 35.4 MB/s
```

Now I am going to update this block as follows:

```
SQL> update pdbadmin.testlosttable set ID = 30 where id = 20;

1 row updated.

SQL> commit;

Commit complete.

SQL> select rowid, dbms_rowid.ROWID_BLOCK_NUMBER(rowid), a.* from pdbadmin.testlosttable a;

ROWID              DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)         ID PAYLOAD
------------------ ------------------------------------ ---------- ----------------------------
AAAR8NAAUAAAACGAAA                                  134         10 aaaaaaaaaaaaaaaaaaaaaaaa
AAAR8NAAUAAAACGAAB                                  134         30 bbbbbbbbbbbbbbbbbbbbbbbb

SQL> alter system checkpoint;

System altered.

SQL> alter system flush buffer_cache;

System altered.
```

Save this updated block on disk as well:

```
$ SQL> !dd
if=/u01/app/oracle/oradata/PROD_DB1/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_rob_lost_ht30hw1g_.
dbf bs=8192 count=1 skip=134 of=/home/oracle/blk134_new.dmp
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 0.000231246 s, 35.4 MB/s
```

I will now restore the OLD version of the block, that was saved to disk earlier on, in /home/oracle/blk134.dmp

```
SQL> !dd if=/home/oracle/blk134_old.dmp
of=/u01/app/oracle/oradata/PROD_DB1/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_rob_lost_ht30hw1g_.
dbf seek=134 count=1 bs=8192 conv=notrunc
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 0.000406965 s, 20.1 MB/s
```

Let's now rerun the query:

```
SQL> select rowid, dbms_rowid.ROWID_BLOCK_NUMBER(rowid), a.* from pdbadmin.testlosttable a;

ROWID              DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)         ID PAYLOAD
------------------ ------------------------------------ ---------- --------------------------
AAAR8NAAUAAAACGAAA                                  134         10 aaaaaaaaaaaaaaaaaaaaaaaa
AAAR8NAAUAAAACGAAB                                  134         20 bbbbbbbbbbbbbbbbbbbbbbbb
```

The old versionof the block is restored. As if the update from 20 to 30 never happend ! This can cause all kinds of logical corruption in the database and at this stage, the database is not aware of it. At some point, if the block is accessed, for example during a recovery, Oracle will notice that the SCN of the block is outdated. The block is not considered corrupt by RMAN. (Altough you could mark it as such and perform a block recovery). But first you have to detect it first !

Restore the situation by putting back the new block:

```
SQL> !dd if=/home/oracle/blk134_new.dmp
of=/u01/app/oracle/oradata/PROD_DB1/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_rob_lost_ht30hw1g_.
dbf seek=134 count=1 bs=8192 conv=notrunc
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 0.000406965 s, 20.1 MB/s
```

7.3. Scenario 2 : Lost Write when DB_LOST_WRITE_PROTECT is enabled

I will now repeat this exercise while DB_LOST_WRITE_PROTECTION is enabled.

First the parameter DB_LOST_WRITE_PROTECT must be set to typical in both the Primary and the Standby:

```
SQL> alter session set container=cdb$root;

Session altered.

SQL> alter system set db_lost_write_protect=TYPICAL;

System altered.
```

Check that Real Time Apply and Real Time Query are active on the Physcial Standby. This is a requirement for automatic block repair to work:

```
DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 0 seconds ago)
  Average Apply Rate: 0 Byte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

Now I will perform another update on the same block

```
SQL> show parameter db_unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      prod_db1

SQL> alter session set container=pdb1;

Session altered.

SQL> insert into testlosttable values (50,'ccccccccccccccccccccccc');

1 row created.

SQL> commit;

SQL> select rowid, dbms_rowid.ROWID_BLOCK_NUMBER(rowid), a.* from testlosttable a;

ROWID              DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)         ID PAYLOAD
------------------ ------------------------------------ ---------- ------------------------------
AAAR8NAAUAAAACFAAA                                  133         50 ccccccccccccccccccccccc
AAAR8NAAUAAAACGAAA                                  134         10 aaaaaaaaaaaaaaaaaaaaaaaa
AAAR8NAAUAAAACGAAB                                  134         20 bbbbbbbbbbbbbbbbbbbbbbbb
```

Now I am going to destroy the content of block 133:

```
SQL> !dd
of=/u01/app/oracle/oradata/PROD_DB1/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_rob_lost_ht30hw1g_.
dbf bs=8192 seek=133 conv=notrunc count=1 if=/dev/zero
1+0 records in
1+0 records out
8192 bytes (8.2 kB) copied, 0.000334223 s, 24.5 MB/s
```

Now flush the blocks to disk to force a disk read and read the data again:

```
SQL> alter system flush buffer_cache;

System altered.

SQL> select rowid, dbms_rowid.ROWID_BLOCK_NUMBER(rowid), a.* from testlosttable a;

ROWID              DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)         ID PAYLOAD
------------------ ------------------------------------ ---------- ------------------------------
AAAR8NAAUAAAACFAAA                                  133         50 ccccccccccccccccccccccc
AAAR8NAAUAAAACGAAA                                  134         10 aaaaaaaaaaaaaaaaaaaaaaaa
AAAR8NAAUAAAACGAAB                                  134         20 bbbbbbbbbbbbbbbbbbbbbbbb
```

The data comes back completely sound and the above corruption is repaired automatically ! I notice that the read takes noticably longer than before. This is because the Dataguard replaced the corrupted block on the Primary Database with the good block on the Standby Database. This is Dataguard Automatic Block Repair. You can see the evidence of this in the database alert file:

```
[oracle@rob01db01 trace]$ vi /u01/app/oracle/diag/rdbms/prod_db1/prod_db1/trace/alert_prod_db1.log

2020-11-06T10:49:48.878603+00:00
PDB1(3):Hex dump of (file 20, block 133) in trace file
/u01/app/oracle/diag/rdbms/prod_db1/prod_db1/trace/prod_db1_ora_2276.trc
PDB1(3):
PDB1(3):Corrupt block relative dba: 0x05000085 (file 20, block 133)
PDB1(3):Completely zero block found during multiblock buffer read
PDB1(3):
PDB1(3):Reading datafile
'/u01/app/oracle/oradata/PROD_DB1/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_rob_lost_ht30hw1g_.db
f' for corrupt data at rdba: 0x05000085 (file 20, block 133)
PDB1(3):Reread (file 20, block 133) found same corrupt data (no logical check)
PDB1(3):Starting background process ABMR
2020-11-06T10:49:48.902275+00:00
Corrupt Block Found
         TIME STAMP (GMT) = 11/06/2020 10:49:48
         CONT = 3, TSN = 7, TSNAME = ROB_LOST_WRITE
         RFN = 20, BLK = 133, RDBA = 83886213
         OBJN = 73485, OBJD = 73485, OBJECT = TESTLOSTTABLE, SUBOBJECT =
         SEGMENT OWNER = PDBADMIN, SEGMENT TYPE = Table Segment
2020-11-06T10:49:48.918417+00:00
ABMR started with pid=63, OS id=3278
2020-11-06T10:49:48.921836+00:00
Automatic block media recovery service is active.
2020-11-06T10:49:48.922331+00:00
PDB1(3):Automatic block media recovery requested for (file# 20, block# 133)
2020-11-06T10:49:51.190543+00:00
Automatic block media recovery successful for (file# 20, block# 133)
2020-11-06T10:49:51.191237+00:00
PDB1(3):Automatic block media recovery successful for (file# 20, block# 133)
```

So this is a great feature to have and I have acually see it being done in a real life production environment once.

Let;s clean up this test:

8. Dataguard New Feautures in 12cR2, 18c and 19c

8.1. 12cR2: ENABLED_PDBS_ON_STANDBY to exclude PDBs from Dataguard Replication

The ENABLED_PDBS_ON_STANDBY initialisation parameter was introduced in this form in 12.2 to control which pluggable databases are protected by a specific standby database. The parameter can be set on a primary or standby database, but it is only used by standby databases. Here are some examples of how the parameter might be used with wildcard and exclusions.

- "*" : All PDBs are protected.
- "PDB1", "PDB2" : Only pluggable databases called "PDB1" and "PDB2" are protected.
- "PDB*" : Only pluggable databases with a name beginning with "PDB" are protected.
- "*", "-PDB*" : All pluggable databases are protected, except those with a name beginning with "PDB".
- "*", "-PDB1" : All pluggable databases are protected, except if the name is "PDB1".

On the standby database I will issue the following command to prevent a pluggable database called PDB3 from being replicated to the standby database. All other PDBs will be protected as normal.

Just to have the same settings on the Primary, I will also set it on the Primary:

```
SQL> show parameter enabled_PDBs_on_standby

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
enabled_PDBs_on_standby              string      *

SQL> ALTER SYSTEM SET enabled_pdbs_on_standby="*", "-PDB3";

System altered.

SQL> show parameter enabled_pdbs_on_standby

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
enabled_PDBs_on_standby              string      *, -PDB3
```

On the Primary Database create PDB3:

```
SQL> create pluggable database pdb3 admin user pdbadmin identified by pdbadmin roles = (DBA);

Pluggable database created.

SQL> alter pluggable database pdb3 open;

Pluggable database altered.
```

Let's check the status on both the Primary and the Standby database:

```
--On the primary:
SQL> SELECT name, open_mode, recovery_status  FROM v$pdbs ORDER BY 1;

NAME                 OPEN_MODE  RECOVERY
-------------------- ---------- --------
PDB$SEED             READ ONLY  ENABLED
PDB1                 READ WRITE ENABLED
PDB2                 READ WRITE ENABLED
PDB3                 READ WRITE ENABLED
```

```
--On the standby:
SQL> SELECT name, open_mode, recovery_status  FROM v$pdbs ORDER BY 1;

NAME            OPEN_MODE  RECOVERY
--------------- ---------- --------
PDB$SEED        READ ONLY  ENABLED
PDB1            READ ONLY  ENABLED
PDB2            READ ONLY  ENABLED
PDB3            MOUNTED    DISABLED
```

Due to the setting the recovey of PDB3 is disabled and the database is in the MOUNTED state.

Let's clean up this little test and re-instate the original settings:

On the Primary:

```
SQL> alter pluggable database pdb3 close immediate;

SQL> drop pluggable database pdb3 including datafiles;
```

On both the Primary and the Standby:

```
SQL> alter system set enabled_pdbs_on_standby = '*';

System altered.
```

8.2. 12cR2: Detect lost writes via DBMS_DBCOMP.DBCOMP.

Oracle introduced very useful tool or rather PL/SQL package DBMS_DBCOMP which can be used to detect inconsistencies between primary and all its standby databases on block level. It can be run both on primary or standby which is mounted or opened and will do the job no matter of DB_LOST_WRITE_PROTECT setting.

You can run the package for all datafiles or for individual data files. First I use just 1 datafile:

```
SQL> select name from v$datafile where rownum < 2;

NAME
--------------------------------------------------------------------------------
/u01/app/oracle/oradata/PROD/datafile/o1_mf_system_hsog1w0s_.dbf
```

```
SQL> exec
sys.dbms_dbcomp.dbcomp('/u01/app/oracle/oradata/PROD/datafile/o1_mf_system_hsog1w0s_.dbf','/tmp/dbcomp_
test/dbcomp', true);

PL/SQL procedure successfully completed.

Elapsed: 00:00:06.93
```

After that I run it against all the datafiles:

```
SQL> exec sys.dbms_dbcomp.dbcomp('ALL','/home/oracle/rob_test',true);

PL/SQL procedure successfully completed.

Elapsed: 00:00:32.14
```

Navigate to the /home/oracle and examine the files:

```
[oracle@rob01db01 trace]$ cd /home/oracle
[oracle@rob01db01 ~]$ ls -altr rob_test*

-rw-r----- 1 oracle oinstall 2412 Nov  1 15:41 rob_test_prod_db2_1
-rw-r----- 1 oracle oinstall 2802 Nov  1 15:41 rob_test_prod_db2_3
-rw-r----- 1 oracle oinstall 1790 Nov  1 15:41 rob_test_prod_db2_4
-rw-r----- 1 oracle oinstall 2256 Nov  1 15:41 rob_test_prod_db2_5
-rw-r----- 1 oracle oinstall 2334 Nov  1 15:41 rob_test_prod_db2_6
-rw-r----- 1 oracle oinstall 2021 Nov  1 15:41 rob_test_prod_db2_7
-rw-r----- 1 oracle oinstall 1790 Nov  1 15:41 rob_test_prod_db2_8
-rw-r----- 1 oracle oinstall 2289 Nov  1 15:41 rob_test_prod_db2_9
-rw-r----- 1 oracle oinstall 2758 Nov  1 15:41 rob_test_prod_db2_10
-rw-r----- 1 oracle oinstall 1824 Nov  1 15:41 rob_test_prod_db2_11
-rw-r----- 1 oracle oinstall 1665 Nov  1 15:41 rob_test_prod_db2_12
-rw-r----- 1 oracle oinstall 2290 Nov  1 15:41 rob_test_prod_db2_13
-rw-r----- 1 oracle oinstall 2368 Nov  1 15:41 rob_test_prod_db2_14
-rw-r----- 1 oracle oinstall 1824 Nov  1 15:41 rob_test_prod_db2_15
-rw-r----- 1 oracle oinstall 1983 Nov  1 15:41 rob_test_prod_db2_19
-rw-r----- 1 oracle oinstall 4229 Nov  1 15:41 rob_test
```

If you check the output you can see that there is one file generated per primary/standby with datafile number suffix – apart from the file generated for the database where you run the DBCOMP.

Contents of file rob_test: Column 3 contains any corrupted blocks

```
[oracle@rob01db01 ~]$ cat rob_test
Client is connected to database: prod_db1. Role: primary database.
Remote database prod_db2.remote db role: physical standby

Slave Id  0
Summary:
*******************************************************************************
                       TOTAL: total no. of blocks found
                         |
   +--------+------------+-------+---------+---------+
   |        |                    |         |         |
   |      DIFFV:              LOST_WRITE    |       CORR: corrupted blocks
 SAMEV     diff ver              |       SKIPPED:
   |       block pairs         +--+--+      direct load, empty blocks,
+--+--+--+                     |     |      RMAN optimized blocks,
|  |  |  |                     |     |      flashback optimized blocks
|  |  | SAMEV&C:               |     |
|  |  |  same ver &            |   LWLOC: lost writes at local db
|  |  |  same checksum &     LWRMT: lost writes at remote db
|  |  |  same contents
|  |  |
|  | SAMEV_NO_CHKSUM: same ver & same contents but diff checksum
|  |                  (checksum can be diff but identical contents)
|  |
| DIFFPAIR: same ver but differrent contents (data inconsistency)
|
```

```
ENCERR: undecided block pairs due to encryption related issue
        (e.g. when Wallet is not open)


ID TOTAL   CORR SKIPPED DIFFV   SAMEV   SAMEV&C ENCERR  LWLOC  LWRMT DIFFPAIR
00 0115067 0000 0115067 0000000 0000000 0000000 0000000 000000 000000 0000000
02 0072290 0000 0048170 0000000 0024120 0011172 0000000 000000 000000 0000000
06 0287897 0000 0087336 0000001 0200560 0200168 0000000 000000 000000 0000000
14 0000004 0000 0000000 0000000 0000004 0000004 0000000 000000 000000 0000000
16 0004365 0000 0000000 0000000 0004365 0004365 0000000 000000 000000 0000000
23 0000301 0000 0000000 0000000 0000301 0000301 0000000 000000 000000 0000000
25 0000301 0000 0000000 0000000 0000301 0000301 0000000 000000 000000 0000000
26 0000034 0000 0000000 0000000 0000034 0000034 0000000 000000 000000 0000000
27 0005121 0000 0000000 0000000 0005121 0005121 0000000 000000 000000 0000000
29 0000015 0000 0000000 0000000 0000015 0000015 0000000 000000 000000 0000000
30 0001875 0000 0000000 0000000 0001875 0001875 0000000 000000 000000 0000000
32 0006334 0000 0000000 0000001 0006333 0006325 0000000 000000 000000 0000000
33 0005395 0000 0000000 0000000 0005395 0005395 0000000 000000 000000 0000000
35 0005395 0000 0000000 0000000 0005395 0005395 0000000 000000 000000 0000000
38 0000040 0000 0000000 0000001 0000039 0000039 0000000 000000 000000 0000000
58 0000353 0000 0000353 0000000 0000000 0000000 0000000 000000 000000 0000000
61 0012851 0000 0000000 0000000 0012851 0012851 0000000 000000 000000 0000000
62 0000580 0000 0000000 0000000 0000580 0000580 0000000 000000 000000 0000000
63 0001423 0000 0000000 0000000 0001423 0001423 0000000 000000 000000 0000000
64 0003548 0000 0000000 0000000 0003548 0003547 0000000 000000 000000 0000000
69 0001814 0000 0000000 0000000 0001814 0001814 0000000 000000 000000 0000000
70 0000023 0000 0000023 0000000 0000000 0000000 0000000 000000 000000 0000000
80 0000256 0000 0000000 0000000 0000256 0000256 0000000 000000 000000 0000000
81 0000002 0000 0000000 0000000 0000002 0000002 0000000 000000 000000 0000000
82 0000256 0000 0000000 0000000 0000256 0000256 0000000 000000 000000 0000000
83 0000256 0000 0000000 0000000 0000256 0000256 0000000 000000 000000 0000000
84 0000254 0000 0000000 0000000 0000254 0000254 0000000 000000 000000 0000000

Short description for each block type:
*******************************************************************************
02: KTU UNDO BLOCK
06: trans data
14: KTU UNDO HEADER W/UNLIMITED EXTENTS
16: DATA SEGMENT HEADER - UNLIMITED
23: BITMAPPED DATA SEGMENT HEADER
25: BITMAP INDEX BLOCK
26: BITMAP BLOCK
27: LOB BLOCK
29: KTFB Bitmapped File Space Header
30: KTFB Bitmapped File Space Bitmap
32: FIRST LEVEL BITMAP BLOCK
33: SECOND LEVEL BITMAP BLOCK
35: PAGETABLE SEGMENT HEADER
38: KTU SMU HEADER BLOCK
58: block type 58
61: NGLOB: Hash Bucket
62: NGLOB: Committed Free Space
63: NGLOB: Segment Header
64: NGLOB: Persistent Undo
69: NGLOB: Lob Extent Header
70: block type 70
80: KTFBN File Space Property Map
81: Stats Segment Header
82: Stats Map
83: Stats Map Summary
84: Stats bitmap
```

Column 3 – CORR: corrupted blocks
Column 5 – DIFFV: diff ver block pairs
Column 8 – ENCERR: undecided block pairs due to encryption related issue (e.g. when Wallet is not open)
Column 9 – LWLOC: lost writes at local db
Column 10 – LWRMT: lost writes at remote db
Column 11 – DIFFPAIR: same ver but different contents (data inconsistency

8.3. 12cR2: STANDBY_DB_PRESERVE_STATES to keep sessions during switchover

As of Oracle Database 12c Release 2 (12.2.0.1), when a physical standby database is converted into a primary you have the option to keep any sessions connected to the physical standby, without disruption, during the switchover/failover. When the database is reopened as the primary, the suspended sessions resume their operations as if nothing had happened. If the database (or an individual PDB) is not opened in the primary role, the sessions will be terminated.
To enable this feature, the STANDBY_DB_PRESERVE_STATES initialization parameter in the standby side is used. This parameter can have following values:

- NONE — No sessions on the standby are retained during a switchover/failover.
- SESSION - User sessions are retained during switchover/failover.
- BUFFER – All current buffers are retained
- ALL – Same as SESSION + BUFFER

Remark: session that use database links and long running queries are not retained, regardless of the setting.

Let's check out the current value:

```
SQL> show parameter standby_db_preserve_states

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
standby_db_preserve_states           string      NONE
```

The parameter needs to be changed on the Standby Database. I will also change it on the Primary, so it can take effect after a switchover or failover as well. The parameter can not be changed dynamically:

```
SQL> alter system set standby_db_preserve_states='ALL' scope=spfile;

System altered.
```

Now I need to restart both the Primary and the Standby database for the change to take effect.

Restart the instances for the parameter to take effect:

```
--Primary:
SQL> shutdown immediate

SQL> startup

--Standby:
SQL> shutdown immediate;

SQL> startup mount;

SQL> alter database open read only;
```

Now a sesssion on the Standby will be maintained during a switchover operation and failiover operation. This is shown in the chapter where I am performing a Dataguard switchover operation.

8.4. 18c: Private and Global Temporary tables in the Standby Database

8.4.1. Create a Private Temporary Table in the Standby Database

This is an Oracle18c New Feature which allows for the table definition itself as well as the data to be temporary to the transaction (the default) or the session (using the ONCOMMIT PRESERVE clause). In contrast, in the Global Temporary Table construction, the metadata table definition is fixed and only the data it holds is temporary.

It is not possible to create a private temporary table in the root container. But the user SYS is not allowed to create private temporary tables, leading to error “ORA-14451: unsupported feature with temporary table”.

So I will log in as system to the Standby Database

```
[oracle@rob01db01 trace]$ sqlplus system/oracle

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY


SQL> create private temporary table ORA$PTT_ROB3 (kol1 NUMBER);

Table created.

SQL> insert into ORA$PTT_ROB3 values (1);
insert into ORA$PTT_ROB3 values (1)
            *
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-02063: preceding line from ADGREDIRECT
```

I am getting an error message that the table or view does not exist. So, this feature is not compatible with the DML Redirection feature. Because of the DML redirection, the insert is shipped to the Primary database and there this private temporary table does not exist. So I need to disable DML redirection in order for this private temporary table feature to work on the Standby. I assume it is a bug that will be resolved at some stage.

```
SQL> show parameter dml

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
adg_redirect_dml                     boolean     TRUE
dml_locks                            integer     2076

SQL> alter system set adg_redirect_dml = false;

System altered.

SQL> insert into ORA$PTT_ROB3 values (1);

1 row created.

SQL> select * from ORA$PTT_ROB3;

      KOL1
----------
         1

SQL> commit;
```

```
Commit complete.

SQL> select * from ORA$PTT_ROB3;
select * from ORA$PTT_ROB3
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Now the private temporary table works as intended. The default is that the table as well as the data are automatically removed after a commit. The alternative is to keep the table for the duration of the session, adding the ON COMMIT PRESERVE clause to the create private temporary table definition.

8.4.2. Create a Global Temporary Table in the Standby Database

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

DATABASE_ROLE
----------------
PHYSICAL STANDBY

SQL> show parameter temp_undo

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
temp_undo_enabled                    boolean     FALSE

SQL> alter system set temp_undo_enabled=TRUE;

System altered.
```

Remark: the parameter temp_undo_enabled = TRUE allows for the UNDO generated on the Temporary Table to be stored in the TEMPORARY tablespace rather than the UNDO tablespace. This is a necessary requirement.

```
SQL> CREATE GLOBAL TEMPORARY TABLE GTT_ROB2 (col1 number, col2 varchar(10)) ON COMMIT PRESERVE ROWS;
CREATE GLOBAL TEMPORARY TABLE GTT_ROB2 (col1 number, col2 varchar(10)) ON COMMIT PRESERVE ROWS
*
ERROR at line 1:
ORA-16000: database or pluggable database open for read-only access
```

This error occurs if DML Redirection is active. If you disable DML Redirection as explained in the previous paragrah, the feature works on the Standby:

```
SQL> CREATE GLOBAL TEMPORARY TABLE GTT_ROB2 (col1 number, col2 varchar(10)) ON COMMIT PRESERVE ROWS;

Table created.
```

8.5. 18c: ADG_ACCOUNT_INFO_TRACKING

The ADG_ACCOUNT_INFO_TRACKING parameter extends the control of user account security information and reaction.

- 'LOCAL' (default value) continues to enforce the existing behavior: Maintains local copy of users account information in the Standby's in-memory view. Login failures are only tracked locally on a per database basis and login denied when the failure maximum is reached.
- 'GLOBAL' triggers the new secure behavior: Maintains a single global copy of users account info across all Data Guard databases Login failures across all databases in the Data Guard configuration count towards the maximum count and logins anywhere will be denied when the count is reached.

Perform the following steps on both the Primary Database and the Standby Database

```
SQL> show parameter adg_account_info_tracking

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
adg_account_info_tracking            string      LOCAL

SQL> alter system set adg_account_info_tracking = global scope = spfile;
```

Restart the instances for the parameter to take effect:

```
--Primary:
SQL> shutdown immediate

SQL> startup

--Standby:
SQL> shutdown immediate;

SQL> startup mount;

SQL> alter database open read only;
```

Now, on the Primary Database, create a test user to test out this parameter setting:

```
SQL> create profile c##prof_robtest limit failed_login_attempts 2;

Profile created.

SQL> create user c##test_user identified by test_user profile c##prof_robtest;

User created.

SQL> grant dba to c##test_user;

Grant succeeded.
```

Log on to the Standby Database and see if you can connect with this user:

```
[oracle@rob01db01 ~]$ sqlplus c##test_user/test_user

SQL*Plus: Release 19.0.0.0.0 - Production on Sun Nov 1 09:54:31 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY
```

Now log on the Standby Database 3 times with a wrong password:

```
[oracle@rob01db01 ~]$ sqlplus c##test_user/wrong_password

SQL*Plus: Release 19.0.0.0.0 - Production on Sun Nov 1 09:55:37 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

ERROR:
ORA-01017: invalid username/password; logon denied
```

The third time, the FAILED_LOGIN_ATTEMPTS profile setting is exceeded and the account is locked:

```
[oracle@rob01db01 ~]$ sqlplus c##test_user/wrong_password

SQL*Plus: Release 19.0.0.0.0 - Production on Sun Nov 1 09:57:43 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

ERROR:
ORA-28000: The account is locked.
```

So this all happened on the Standby Database. Now return to the Primary Database and try to log on as c##test_user with the correct password. Because ADG_ACCOUNT_INFO_TRACKING is global, the locked account is propagated to all the other instances that are part of the Dataguard Configuration.

```
[oracle@rob01db01 ~]$ echo $ORACLE_SID
prod_db1
[oracle@rob01db01 ~]$ sqlplus c##test_user/test_user

SQL*Plus: Release 19.0.0.0.0 - Production on Sun Nov 1 10:07:14 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

ERROR:
ORA-28000: The account is locked.
```

8.6. 18c: New Broker commands and new view v$DATAGUARD_PROCESS

8.6.1. New Dataguard Broker Commands

Check the current settings

```
[oracle@rob01db01 ~]$ dgmgrl /
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Sun Nov 1 10:50:51 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db1"
Connected as SYSDG.

DGMGRL> show all
trace_level         USER
echo                OFF
time                OFF
observerconfigfile = observer.ora
```

Compare the spfiles

```
--make sure to connect to the broker with a password (needed to logon to the Standby)
[oracle@rob01db01 ~]$ dgmgrl sys/oracle as sysdba

DGMGRL for Linux: Release 19.0.0.0.0 - Production on Sun Nov 1 10:59:13 2020
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db1"
Connected as SYSDBA.

DGMGRL> validate database "prod_db2" spfile;
Connecting to "prod_db1".
Connected to "prod_db1"

Connecting to "prod_db2".
Connected to "prod_db2"


Parameter settings with different values:

audit_file_dest:
prod_db1 (PRIMARY) : /u01/app/oracle/admin/prod/adump
prod_db2           : /u01/app/oracle/admin/prod_db2/adump
```

Verify the network configuration.

Very usefull prior to a switchover operation, in combination with validate database!

```
DGMGRL> validate network configuration for all;
Connecting to instance "PROD_DB1" on database "prod_db1" ...
Connected to "prod_db1"
Checking connectivity from instance "PROD_DB1" on database "prod_db1 to instance "prod_db2" on database
"prod_db2"...
Succeeded.
Connecting to instance "prod_db2" on database "prod_db2" ...
Connected to "prod_db2"
Checking connectivity from instance "prod_db2" on database "prod_db2 to instance "PROD_DB1" on database
"prod_db1"...
Succeeded.

Oracle Clusterware is not configured on database "prod_db1".
Connecting to database "prod_db1" using static connect identifier
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=
prod_db1_DGMGRL.robdomain)(INSTANCE_NAME=prod_db1)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
The static connect identifier allows for a connection to database "prod_db1".

Oracle Clusterware is not configured on database "prod_db2".
Connecting to database "prod_db2" using static connect identifier
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1522))(CONNECT_DATA=(SERVICE_NAME=
prod_db2_DGMGRL.robdomain)(INSTANCE_NAME=prod_db2)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
The static connect identifier allows for a connection to database "prod_db2".
```

Validate Static Connect Identifier

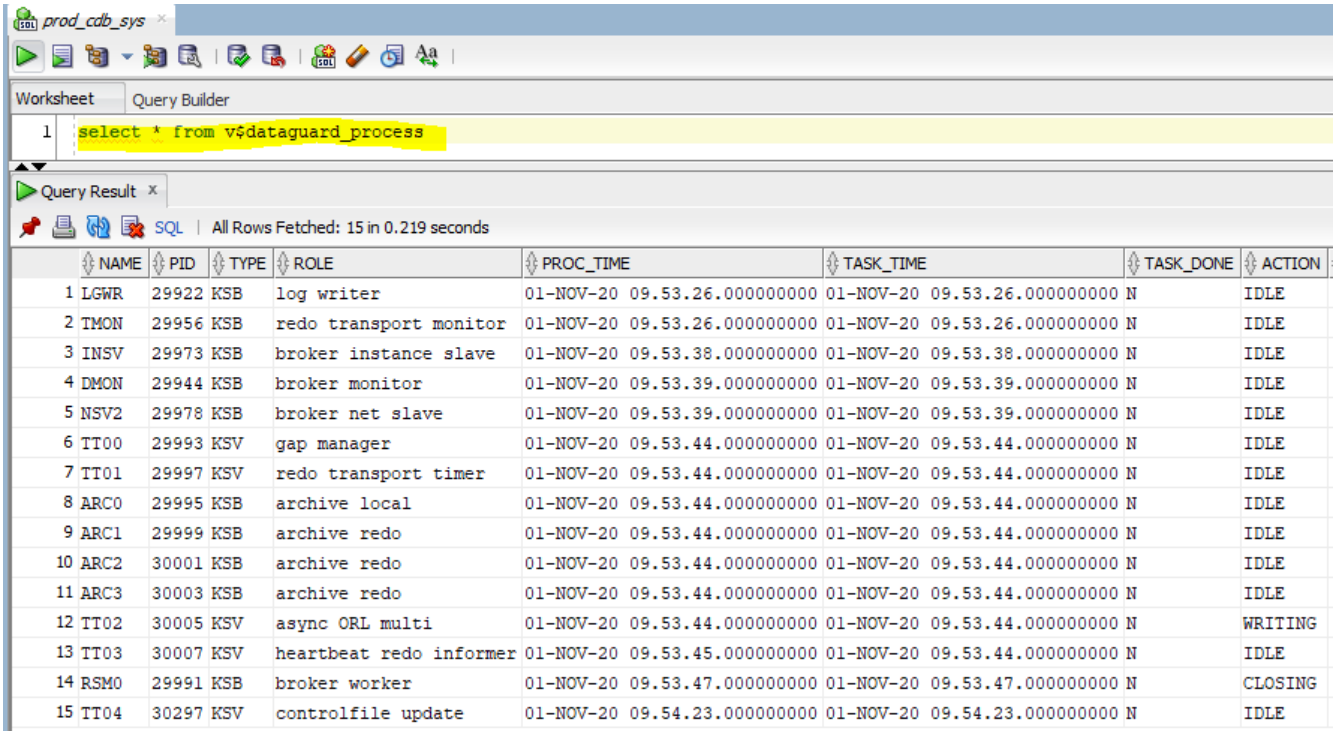
```
DGMGRL> validate static connect identifier for all;
Oracle Clusterware is not configured on database "prod_db1".
Connecting to database "prod_db1" using static connect identifier
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=
prod_db1_DGMGRL.robdomain)(INSTANCE_NAME=prod_db1)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
The static connect identifier allows for a connection to database "prod_db1".

Oracle Clusterware is not configured on database "prod_db2".
Connecting to database "prod_db2" using static connect identifier
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rob01db01.robdomain)(PORT=1522))(CONNECT_DATA=(SERVICE_NAME=
prod_db2_DGMGRL.robdomain)(INSTANCE_NAME=prod_db2)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
The static connect identifier allows for a connection to database "prod_db2".
```

Show configuration lag

```
DGMGRL> show configuration lag

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database
               Transport Lag:      0 seconds (computed 0 seconds ago)
               Apply Lag:          0 seconds (computed 0 seconds ago)

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 27 seconds ago)
```

8.6.2. New View v$dataguard_process

V$DATAGUARD_PROCESS displays one row for each Oracle Data Guard process that is currently running.



	NAME	PID	TYPE	ROLE	PROC_TIME	TASK_TIME	TASK_DONE	ACTION
1	LGWR	29922	KSB	log writer	01-NOV-20 09.53.26.000000000	01-NOV-20 09.53.26.000000000	N	IDLE
2	TMON	29956	KSB	redo transport monitor	01-NOV-20 09.53.26.000000000	01-NOV-20 09.53.26.000000000	N	IDLE
3	INSV	29973	KSB	broker instance slave	01-NOV-20 09.53.38.000000000	01-NOV-20 09.53.38.000000000	N	IDLE
4	DMON	29944	KSB	broker monitor	01-NOV-20 09.53.39.000000000	01-NOV-20 09.53.39.000000000	N	IDLE
5	NSV2	29978	KSB	broker net slave	01-NOV-20 09.53.39.000000000	01-NOV-20 09.53.39.000000000	N	IDLE
6	TT00	29993	KSV	gap manager	01-NOV-20 09.53.44.000000000	01-NOV-20 09.53.44.000000000	N	IDLE
7	TT01	29997	KSV	redo transport timer	01-NOV-20 09.53.44.000000000	01-NOV-20 09.53.44.000000000	N	IDLE
8	ARC0	29995	KSB	archive local	01-NOV-20 09.53.44.000000000	01-NOV-20 09.53.44.000000000	N	IDLE
9	ARC1	29999	KSB	archive redo	01-NOV-20 09.53.44.000000000	01-NOV-20 09.53.44.000000000	N	IDLE
10	ARC2	30001	KSB	archive redo	01-NOV-20 09.53.44.000000000	01-NOV-20 09.53.44.000000000	N	IDLE
11	ARC3	30003	KSB	archive redo	01-NOV-20 09.53.44.000000000	01-NOV-20 09.53.44.000000000	N	IDLE
12	TT02	30005	KSV	async ORL multi	01-NOV-20 09.53.44.000000000	01-NOV-20 09.53.44.000000000	N	WRITING
13	TT03	30007	KSV	heartbeat redo informer	01-NOV-20 09.53.45.000000000	01-NOV-20 09.53.44.000000000	N	IDLE
14	RSM0	29991	KSB	broker worker	01-NOV-20 09.53.47.000000000	01-NOV-20 09.53.47.000000000	N	CLOSING
15	TT04	30297	KSV	controlfile update	01-NOV-20 09.54.23.000000000	01-NOV-20 09.54.23.000000000	N	IDLE

8.7. 18c: Rolling Forward the Standby with one command

Reference: MOS Note “18c Roll Forward Physical Standby Using RMAN Incremental Backup in Single Command (Doc ID 2431311.1)”

When the Standby Database lags significantly behind the Primary Database, for example during a long outage of the Standby Database, it can be faster to refresh the Standby with Incremental Backups rather than applying all the redo that was generated during the outage. This technique has been around for quite some time, however, it has been made much easier as of 12cR2 and 18c.

Previsously, when rolling forward a Standby Database, the DBA had to initiate the following steps:

- Identify the start SCN on the Standby for performing an incremental backup on the Primary
- Perform incremental backup on the Primary with the “FROM SCN” clause
- Move the backup pieces to the standby
- Catalog the backup pieces on the standby
- Perform a recovery of the standby using recover database NOREDO
- Refresh the standby controlfile again from the primary

This has all been replaced on 18c with just one RMAN command: RECOVER STANDBY DATABASE.
There are 2 clauses:

- FROM SERVICE: this clause is used to spedify the name of the primary service
- NOREDO: this clause indicates that the refresh needs to be done using backups only, thereby allowing the standby to be rolled forward to a specific time or SCN.

Remark 1 : DBAs need to manually stop the MRP on the standby database before attempting to sync.

Remark 2: Differences between 12cR1 and 18c in relation to RECOVER DATABASE FROM SERVICE:

Starting from 12.1, we could use "RECOVER DATABASE FROM SERVICE"command which will automate a few steps like performing incremental backup on primary, transfer the backup-pieces to standby over network and perform recovery on standby. However, we still had to manually refresh the standby controlfile and manually restore newly-added datafiles. These steps required manual efforts and are error prone especially when standby files are physically located in a path different to that of primary.

Starting with 18.1, we can use a single command to refresh the standby with changes made on primary:

RMAN> RECOVER STANDBY DATABASE FROM SERVICE primary_connect_identifier;

This command will internally keep track of standby file locations, refresh standby controlfile from primary, update the new standby controlfile with standby file names, perform incremental backup on primary, transfer the backup-pieces over network to standby and perform recovery on standby

The recover standby database can be used to:

- Refresh the control file from the primary database
- Automatically rename temp file, data files and online logs
- Restore the new data files added to the primary database
- Restart the standby instance
- Recover the standby instance up to the current time

Example:

Step 1: Stop the Managed Recovery Process on the Standby Database:

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 50 seconds ago)

DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 0 seconds ago)
  Average Apply Rate: 1.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

```
DGMGRL> edit database "prod_db2" set state="APPLY-OFF";
Succeeded.
DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-OFF
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: (unknown)
  Real Time Query:    OFF
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

The following query on the Standby shows that there are no Managed Recovery Processes up and running:

```
SQL> select process,status,sequence#,thread# from gv$managed_standby where process like 'MRP%';

no rows selected
```

Step 2: process some information in the Primary Database

I will now process some information in the Primary Database in order to make sure that the Primary is now out of sync with the Standby Database:

```
[oracle@rob01db01 trace]$ sqlplus pdbadmin/pdbadmin@pdb1

SQL> show con_name

CON_NAME
------------------------------
PDB1

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY

SQL> create table rob_test as select * from dba_indexes;

Table created.

SQL> select count(*) from rob_test;

  COUNT(*)
----------
      3075
```

If I now log on to the Standby Database, I can see that this information is not availabe. This is, of course, because I disabled the Redo Apply via the Broker in the previous steps:

```
[oracle@rob01db01 trace]$ . oraenv
ORACLE_SID = [prod_db2] ? prod_db2

[oracle@rob01db01 trace]$ sqlplus / as sysdba

SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY

SQL> alter session set container=pdb1;

Session altered.

SQL> select count(*) from pdbadmin.rob_test;
select count(*) from pdbadmin.rob_test
                              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Step 3: Refresh the Standby Database from Service

```
[oracle@rob01db01 trace]$ echo $ORACLE_SID
prod_db2
[oracle@rob01db01 trace]$ rman target /

Recovery Manager: Release 19.0.0.0.0 - Production on Sat Nov 7 08:05:28 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: PROD (DBID=466914583)

RMAN> recover standby database from service prod_primary;

[last few lines of the output, for complete output see Supplement 2 in this document]
. . . . .
. . . . .
destination for restore of datafile 00019:
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_catalog_hsxlnjcc_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01

starting media recovery

media recovery complete, elapsed time: 00:00:00
Finished recover at 07-NOV-20
Reenabling controlfile options for auxiliary database
Executing: alter database enable block change tracking using file
'/home/oracle/oradata/PROD_DB2/changetracking/o1_mf_hss0l48g_.chg'
flashback needs to be reenabled on standby open
Executing: alter system set standby_file_management=auto
Finished recover at 07-NOV-20
```

The output of this command is substantial. The complete output is shown in Supplement 2 of this document. This command will internally keep track of standby file locations, refresh standby controlfile from primary, update the new standby controlfile with standby file names, perform incremental backup on primary, transfer the backup-pieces over network to standby and perform recovery on standby.

Start the managed recovery process again on the Standby Database

```
DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-OFF
  Transport Lag:      29 minutes 14 seconds (computed 3 seconds ago)
  Apply Lag:          (unknown)
  Average Apply Rate: (unknown)
  Real Time Query:    OFF
  Instance(s):
    prod_db2

  Database Warning(s):
    ORA-16855: transport lag has exceeded specified threshold

Database Status:
WARNING

DGMGRL> edit database "prod_db2" set state="APPLY-ON";
Succeeded.
```

Open the Database and Pluggable Databases in read only mode:

```
SQL> alter database open read only;

Database altered.

SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           MOUNTED
         4 PDB2                           MOUNTED

SQL> alter pluggable database all open read only;

Pluggable database altered.
```

Now check if the table I created earlier on in the Primary is also available on the Standby:

```
SQL> alter session set container=pdb1;

Session altered.

SQL> select count(*) from pdbadmin.rob_test;

  COUNT(*)
----------
      3075
```

Check !

Remark: I noticed after some time that I had some errors in the configuration after this operation:

```
DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      1 hour(s) 37 minutes 47 seconds (computed 3 seconds ago)
  Apply Lag:          1 hour(s) 37 minutes 47 seconds (computed 3 seconds ago)
  Average Apply Rate: 7.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

  Database Warning(s):
    ORA-16853: apply lag has exceeded specified threshold
    ORA-16855: transport lag has exceeded specified threshold

Database Status:
WARNING
```

In the Standby Database alert file:

```
Errors in file /u01/app/oracle/diag/rdbms/prod_db2/prod_db2/trace/prod_db2_rfs_10679.trc:
ORA-19527: physical standby redo log must be renamed
ORA-00312: online log 14 thread 1: '/u01/app/oracle/oradata/PROD_DB1/onlinelog/o1_mf_14_hstg7kkb_.log'
 rfs (PID:10679): No SRLs available for T-1

ORA-19527: physical standby redo log must be renamed
```

The reason I got this error is because I use the exact same db_recovery_file_dest destination for both the Primary and the Standby database. (This will most likely never be the case in any production like environment, but it is just because I host everything on the same server. However, I could have set it different anyway but did not anticipate this error)

A good workaround for this error is explained here: https://smarttechways.com/2018/10/08/6245/ and you can also find information in MOS note “ORA-19527: Physical Standby Redo Log Must Be Renamed...during switchover (Doc ID 2194825.1)”

So I got rid of this error as follows, on the Standby Database:

```
SQL> ALTER SYSTEM SET log_file_name_convert='dummy','dummy' scope=spfile;

SQL> shutdown immediate

SQL> startup mount;

SQL> alter database open read only;

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

After that the apply and transport lag were resolved:

```
DGMGRL> show configuration;

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 41 seconds ago)
```

8.8. Active Dataguard DML Redirection

DML redirection helps in load balancing between the primary and standby databases. When incidental DML is issued on an Active Data Guard standby database, the update is passed to the primary database where it is executed. The resulting redo of the transaction updates the standby database after which control is returned to the application.

On the Primary Database, log on to cdb$root as the sys user and create a sample table:

```
$ . oraenv => prod_db1

$ sqlplus / as sysdba

SQL> create table rob_tables as select * from dba_tables where rownum < 11;

Table created.

SQL> commit;

Commit complete.
```

Log on to the Standby Database and query the table rob_tables:

```
SQL> select name, database_role from v$database;

NAME      DATABASE_ROLE
--------- ----------------
PROD      PHYSICAL STANDBY

SQL> select table_name from rob_tables where rownum < 3;

TABLE_NAME
--------------------------------------------------------------------------------
TS$
ICOL$
```

Now, on the Standby Database, update this table as follows:

```
SQL> update rob_tables set table_name = 'TS_ROB' where table_name = 'T$';
update rob_tables set table_name = 'TS_ROB' where table_name = 'T$'
       *
ERROR at line 1:
ORA-16000: database or pluggable database open for read-only access
```

I need to set a database parameter on both the Primary and the Standby database to enable DML redirection:

```
SQL> show parameter adg_redirect_dml

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
adg_redirect_dml                     boolean     FALSE

SQL> alter system set adg_redirect_dml=true;

System altered.

SQL> show parameter adg_redirect_dml

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
adg_redirect_dml                     boolean     TRUE
```

After I set this parameter in both the Primary and the Standby, I will issue the DML again on the Standby:

```
SQL> select name, database_role from v$database;

NAME      DATABASE_ROLE
--------- ----------------
PROD      PHYSICAL STANDBY

SQL> update rob_tables set table_name = 'TS_ROB' where table_name = 'T$';
update rob_tables set table_name = 'TS_ROB' where table_name = 'T$'
       *
ERROR at line 1:
ORA-16397: statement redirection from Oracle Active Data Guard standby database
to primary database failed
```

The statement failed, because I logged on as /, without providing a password. I need to log on with a password and this password will be used by the redirect to process the DML on the Primary:

```
[oracle@rob01db01 ~]$ sqlplus sys/oracle@prod_db2 as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Sat Oct 31 17:43:15 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> update rob_tables set table_name = 'TS_ROB' where table_name = 'TS$';

1 row updated.

SQL> select count(*) from rob_tables where table_name = 'TS_ROB';

  COUNT(*)
----------
         1
```

Now issue the same statement in the Primary Database:

```
[oracle@rob01db01 ~]$ sqlplus sys/oracle@prod_db1 as sysdba


SQL> select count(*) from rob_tables where table_name = 'TS_ROB';

  COUNT(*)
----------
         0
```

So this change has not been committed on the Standby Database yet, and therefore the change is not yet visible for any other users. After the transaction is committed on the Standby Database the changes will become visible on the Primary Database as well.

8.9. 19c: Restore Point Replication from Primary to Standby

Normal restore points or guaranteed restore points can be defined at the primary site to enable fast point-in-time recovery in the event of any logical corruption issues. However, this restore point is stored in the control file and is not propagated to the standby database. In the event of a failover, the standby becomes the primary and the restore point information is lost. This feature ensures that the restore point is propagated from the primary to standby sites, so that the restore point is available even after a failover event.)

Log on to the Primary and check the settings:

```
SQL> select name, open_mode, controlfile_type ,database_role,flashback_on from v$database;

NAME      OPEN_MODE            CONTROL DATABASE_ROLE    FLASHBACK_ON
--------- -------------------- ------- ---------------- ------------------
PROD      READ WRITE           CURRENT PRIMARY          YES
```

```
SQL> create restore point ROB_TEST1 guarantee flashback database;

Restore point created.
```

```
SQL>
col TIME for a20
col GUARANTEE_FLASHBACK_DATABASE format a8
col name format a20
select scn, guarantee_flashback_database, to_char(time,'DD-MON-YYYY HH24:MI:SS') TIME, name, replicated
from v$restore_point;

       SCN GUARANTE TIME                 NAME       REPLICATED
---------- -------- -------------------- ---------- ---------------
   2702476 YES      02-NOV-2020 07:43:32 ROB_TEST1  NO
```

Now log on to the Standby Database and issue the same query:

```
SQL>
col TIME for a20
col GUARANTEE_FLASHBACK_DATABASE format a8
col name format a20
select scn, guarantee_flashback_database, to_char(time,'DD-MON-YYYY HH24:MI:SS') TIME, name, replicated
from v$restore_point;

       SCN GUARANTE TIME                 NAME                 REP
---------- -------- -------------------- -------------------- ---
   2702476 NO       02-NOV-2020 07:43:32 ROB_TEST1_PRIMARY    YES
```

So the default behaviour is that restore points are replicated to the Standby. This is controlled by the hidden parameter “_standby_auto_flashback”.

Let’s drop the restore point now on the Primary. If you check on the Standby you can see that the replicated restore point is also dropped.

```
SQL> drop restore point ROB_TEST;
```

8.10. 19c: Automatic Flashback of mounted Standby

Flashback Database moves the entire database to an older point in time and opens the database with RESETLOGS. In a Data Guard setup, if the primary database is flashed back, the standby site is no longer in sync with the primary.

In previous releases, getting the secondary to the same point in time as the primary requires a manual procedure to flash back standby databases. A new parameter(_standby_auto_flashback) is introduced which enables the standby database to be flashed back automatically when Flashback Database is performed on the primary database.

(On PRIMARY database, if you create restore point without explicitly enabling FLASHBACK ON, standby database failed to automatically flashback.
In my case I need to recreate standby database from scratch
You must enable “FLASHBACK ON” explicitly for standby database as well)

Test case:

Log on to the Primary Database and create a test table

```
SQL> select name, open_mode, controlfile_type ,database_role,flashback_on from v$database;

NAME      OPEN_MODE            CONTROL DATABASE_ROLE    FLASHBACK_ON
--------- -------------------- ------- ---------------- ------------------
PROD      READ WRITE           CURRENT PRIMARY          YES
```

```
SQL> create table rob_fb_test as select * from dba_objects;

Table created.

SQL> select count(*) from rob_fb_test;

  COUNT(*)
----------
     72509
```

On the Primary : Create a guaranteed restore point and check:

```
SQL> create restore point ROB_TEST2 guarantee flashback database;

Restore point created.

col TIME for a20
col GUARANTEE_FLASHBACK_DATABASE format a8
col name format a20
select scn, guarantee_flashback_database, to_char(time,'DD-MON-YYYY HH24:MI:SS') TIME, name, replicated
from v$restore_point;

       SCN GUARANTE TIME                 NAME                 REPLICATED
---------- -------- -------------------- -------------------- ---------------
   2718307 YES      02-NOV-2020 09:17:26 ROB_TEST2            NO
```

Check also on the Standby if the Restore Point is available:

```
SQL>
col TIME for a20
col GUARANTEE_FLASHBACK_DATABASE format a8
col name format a20
select scn, guarantee_flashback_database, to_char(time,'DD-MON-YYYY HH24:MI:SS') TIME, name, replicated
from v$restore_point;

       SCN GUARANTE TIME                 NAME                 REP
---------- -------- -------------------- -------------------- ---
   2718307 NO       02-NOV-2020 09:17:26 ROB_TEST2_PRIMARY    YES

--we can also see the test table on the Standby
SQL> select count(*) from rob_fb_test;

  COUNT(*)
----------
     72509
```

Now, let's truncate the test table on the Primary Database:

```
SQL> truncate table rob_fb_test;

Table truncated.

SQL> select count(*) from rob_fb_test;

  COUNT(*)
----------
         0
```

Now look on the Standby and verify that this table has now 0 records:

```
SQL> select count(*) from rob_fb_test;

  COUNT(*)
----------
         0
```

Flashback the Primary Database to the Restore Point:

```
SQL> show parameter db_unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_unique_name                       string      prod_db1

SQL> shutdown immediate
SQL> startup mount;
SQL> flashback database to restore point rob_test2;

Flashback complete.

SQL> alter database open resetlogs;

Database altered.

SQL> select count(*) from rob_fb_test;

  COUNT(*)
----------
     72509
```

Now check on the Standby database and see what happened. You can see some errors in the alert file:

```
Warning: Recovery target destination is in a sibling branch
of the controlfile checkpoint. Recovery will only recover
changes to datafiles.
Datafile 1 (ckpscn 2720187) is orphaned on incarnation#=2
PR00 (PID:10771): MRP0: Detected orphaned datafiles!
2020-11-02T09:30:14.605244+00:00
Errors in file /u01/app/oracle/diag/rdbms/prod_db2/prod_db2/trace/prod_db2_pr00_10771.trc:
ORA-19909: datafile 1 belongs to an orphan incarnation
ORA-01110: data file 1: '/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf'
PR00 (PID:10771): Managed Standby Recovery not using Real Time Apply
stopping change tracking
2020-11-02T09:30:14.864789+00:00
Recovery Slave PR00 previously exited with exception 19909
2020-11-02T09:30:14.865510+00:00
Errors in file /u01/app/oracle/diag/rdbms/prod_db2/prod_db2/trace/prod_db2_mrp0_4544.trc:
ORA-19909: datafile 1 belongs to an orphan incarnation
ORA-01110: data file 1: '/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf'
2020-11-02T09:30:17.137104+00:00
 rfs (PID:10784): Opened log for T-1.S-50 dbid 466914583 branch 1055076250
2020-11-02T09:30:18.228835+00:00
 rfs (PID:10784): Archived Log entry 24 added for B-1055076250.T-1.S-50 ID 0x1bd46717 LAD:2
2020-11-02T09:30:34.870816+00:00
MRP0 (PID:4544): Recovery coordinator encountered one or more errors during automatic flashback on
standby
2020-11-02T09:30:34.871042+00:00
Background Media Recovery process shutdown (prod_db2)
```

Now shutdown the Standby Instance and start again in mount mode:

```
SQL> shutdown immediate

SQL> startup mount
```

If I look in the Standby Database alert file I can see that the Media recovery started again:

```
Completed: ALTER DATABASE   MOUNT
2020-11-02T09:47:50.858812+00:00
Starting Data Guard Broker (DMON)
Starting background process INSV
2020-11-02T09:47:50.879423+00:00
INSV started with pid=46, OS id=12155
2020-11-02T09:47:54.976413+00:00
Starting background process NSV0
2020-11-02T09:47:55.013449+00:00
NSV0 started with pid=47, OS id=12160
2020-11-02T09:47:58.180990+00:00
Starting background process RSM0
2020-11-02T09:47:58.218854+00:00
RSM0 started with pid=48, OS id=12169
2020-11-02T09:47:58.297483+00:00
 rfs (PID:12171): Primary database is in MAXIMUM PERFORMANCE mode
2020-11-02T09:47:58.332870+00:00
 rfs (PID:12171): Selected LNO:14 for T-1.S-2 dbid 466914583 branch 1055410205
2020-11-02T09:48:02.682340+00:00
 rfs (PID:12177): Primary database is in MAXIMUM PERFORMANCE mode
 rfs (PID:12177): Re-archiving LNO:14 T-1.S-2
2020-11-02T09:48:02.743163+00:00
 rfs (PID:12177): Selected LNO:13 for T-1.S-3 dbid 466914583 branch 1055410205
2020-11-02T09:48:02.743660+00:00
ARC2 (PID:12141): Archived Log entry 25 added for T-1.S-2 ID 0x1bd9986e LAD:1
2020-11-02T09:48:04.094799+00:00
```

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT  NODELAY
2020-11-02T09:48:04.096535+00:00
Attempt to start background Managed Standby Recovery process (prod_db2)
Starting background process MRP0
2020-11-02T09:48:04.116336+00:00
MRP0 started with pid=52, OS id=12182
2020-11-02T09:48:04.118240+00:00
Background Managed Standby Recovery process started (prod_db2)
2020-11-02T09:48:09.145004+00:00
 Started logmerger process
2020-11-02T09:48:09.177454+00:00
PR00 (PID:12188): Managed Standby Recovery starting Real Time Apply
max_pdb is 5
Warning: Recovery target destination is in a sibling branch
of the controlfile checkpoint. Recovery will only recover
changes to datafiles.
Datafile 1 (ckpscn 2720187) is orphaned on incarnation#=2
PR00 (PID:12188): MRP0: Detected orphaned datafiles!
2020-11-02T09:48:09.194140+00:00
Errors in file /u01/app/oracle/diag/rdbms/prod_db2/prod_db2/trace/prod_db2_pr00_12188.trc:
ORA-19909: datafile 1 belongs to an orphan incarnation
ORA-01110: data file 1: '/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf'
PR00 (PID:12188): Managed Standby Recovery not using Real Time Apply
stopping change tracking
2020-11-02T09:48:09.393137+00:00
Recovery Slave PR00 previously exited with exception 19909
2020-11-02T09:48:09.475372+00:00
Errors in file /u01/app/oracle/diag/rdbms/prod_db2/prod_db2/trace/prod_db2_mrp0_12182.trc:
ORA-19909: datafile 1 belongs to an orphan incarnation
ORA-01110: data file 1: '/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf'
2020-11-02T09:48:29.483159+00:00
MRP0 (PID:12182): Recovery coordinator performing automatic flashback of database to
SCN:0x0000000000297a63 (2718307)
Flashback Restore Start
2020-11-02T09:48:30.141886+00:00
Completed: ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT  NODELAY
Flashback Restore Complete
Flashback Media Recovery Start
2020-11-02T09:48:30.229183+00:00
Setting recovery target incarnation to 2
2020-11-02T09:48:30.249925+00:00
 Started logmerger process
2020-11-02T09:48:30.300191+00:00
max_pdb is 5
2020-11-02T09:48:30.405120+00:00
Parallel Media Recovery started with 2 slaves
2020-11-02T09:48:31.210559+00:00
Block change tracking file is current.
starting change tracking :0
Starting background process CTWR
2020-11-02T09:48:31.235477+00:00
CTWR started with pid=56, OS id=12218
2020-11-02T09:48:31.255467+00:00
Block change tracking service is active.
2020-11-02T09:48:31.409731+00:00
Media Recovery Log /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_50_hsznf90n_.arc
Restore point ROB_TEST2_PRIMARY propagated from primary already exists
Resize operation completed for file# 1, old size 931840K, new size 942080K
2020-11-02T09:48:32.448815+00:00
Incomplete Recovery applied until change 2718307 time 11/02/2020 09:17:26
2020-11-02T09:48:32.459046+00:00
Flashback Media Recovery Complete
2020-11-02T09:48:32.661502+00:00
stopping change tracking
2020-11-02T09:48:32.661764+00:00
Block change tracking service stopping.
Stopping background process CTWR
2020-11-02T09:48:33.782821+00:00
Setting recovery target incarnation to 3
2020-11-02T09:48:33.820550+00:00
 Started logmerger process
2020-11-02T09:48:33.853944+00:00
PR00 (PID:12222): Managed Standby Recovery starting Real Time Apply
```

```
max_pdb is 5
2020-11-02T09:48:33.977195+00:00
Parallel Media Recovery started with 2 slaves
2020-11-02T09:48:34.084021+00:00
Media Recovery start incarnation depth : 1, target inc# : 3, irscn : 2718308
Block change tracking file is current.
starting change tracking :0
Starting background process CTWR
2020-11-02T09:48:34.816810+00:00
CTWR started with pid=56, OS id=12228
2020-11-02T09:48:34.836746+00:00
Block change tracking service is active.
```

Now open the database read only to have active dataguard:

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

OPEN_MODE
--------------------
MOUNTED

SQL> show pdbs;

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       MOUNTED
         3 PDB1                           MOUNTED
         4 PDB2                           MOUNTED

SQL> alter database open read only;

Database altered.

SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           MOUNTED
         4 PDB2                           MOUNTED

SQL> alter pluggable database all open read only;

Pluggable database altered.

SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ ONLY  NO
         4 PDB2                           READ ONLY  NO
```

Remark: I am not sure why the Pluggable Databases do not open read write automatically. Maybe I did not have the PDBs in the right state when I created the Standby Database. Or maybe this is something that always needs to be done on the Standby.

Let's check if test table on the Standby:

```
SQL> select count(*) from rob_fb_test;

  COUNT(*)
----------
     72509
```

Finally, let's check the Dataguard Broker configuration:

```
[oracle@rob01db01 trace]$ dgmgrl /
DGMGRL for Linux: Release 19.0.0.0.0 - Production on Mon Nov 2 10:14:38 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected to "prod_db2"
Connected as SYSDG.
DGMGRL> show configuration

Configuration - DG_PROD

  Protection Mode: MaxPerformance
  Members:
  prod_db1 - Primary database
    prod_db2 - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 14 seconds ago)

DGMGRL> show database "prod_db2";

Database - prod_db2

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: 1.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    prod_db2

Database Status:
SUCCESS
```

8.11. Export and Import Dataguard Broker Configuration

This can be quite usefull as you do not have to recreate a broker configuration at times. Now you can just export and import the Broker Configuration metadata:

Export the Broker Information:

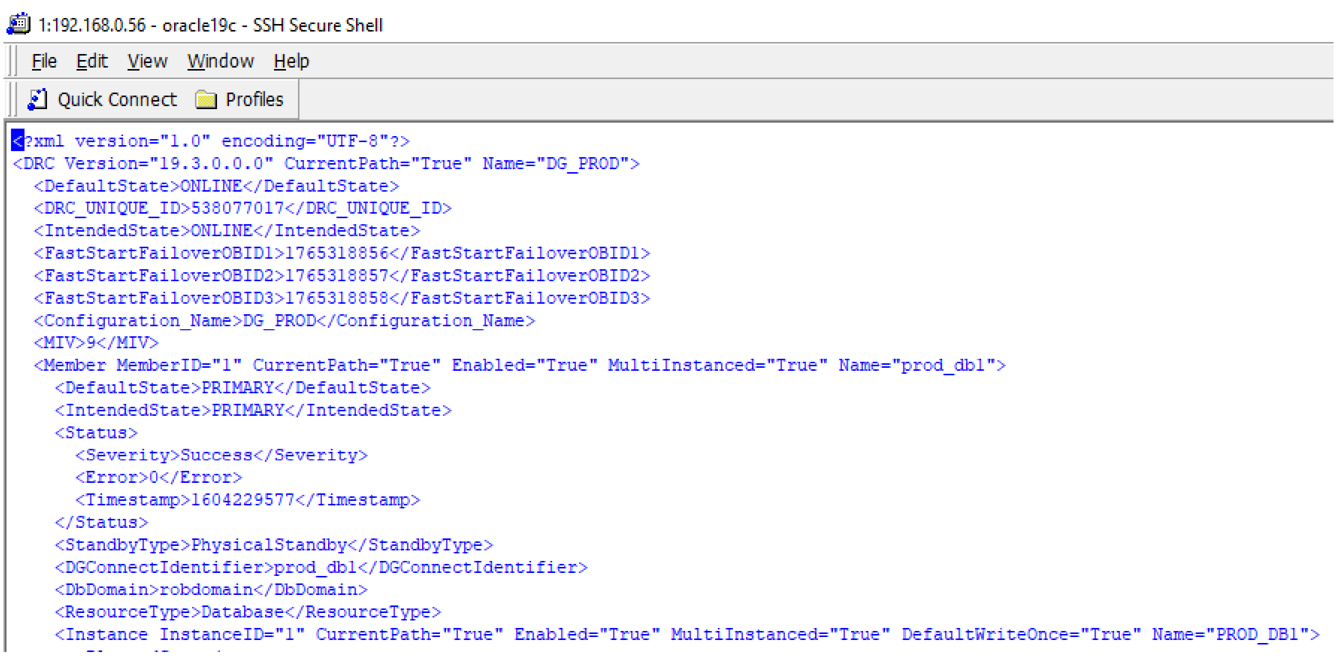
```
DGMGRL> EXPORT CONFIGURATION TO 'meta.xml' ;
Succeeded.
```

Import the Broker Information:

```
DGMGRL> IMPORT CONFIGURATION FROM 'meta.xml' ;
Succeeded.
```

Remark: I was not able to use subdirectories in the output file specification. The xml file is written to the database trace directory:

```
[oracle@rob01db01 u01]$ find . -name meta.xml
./app/oracle/diag/rdbms/prod_db1/prod_db1/trace/meta.xml
```



8.12. Miscellaneous New Features

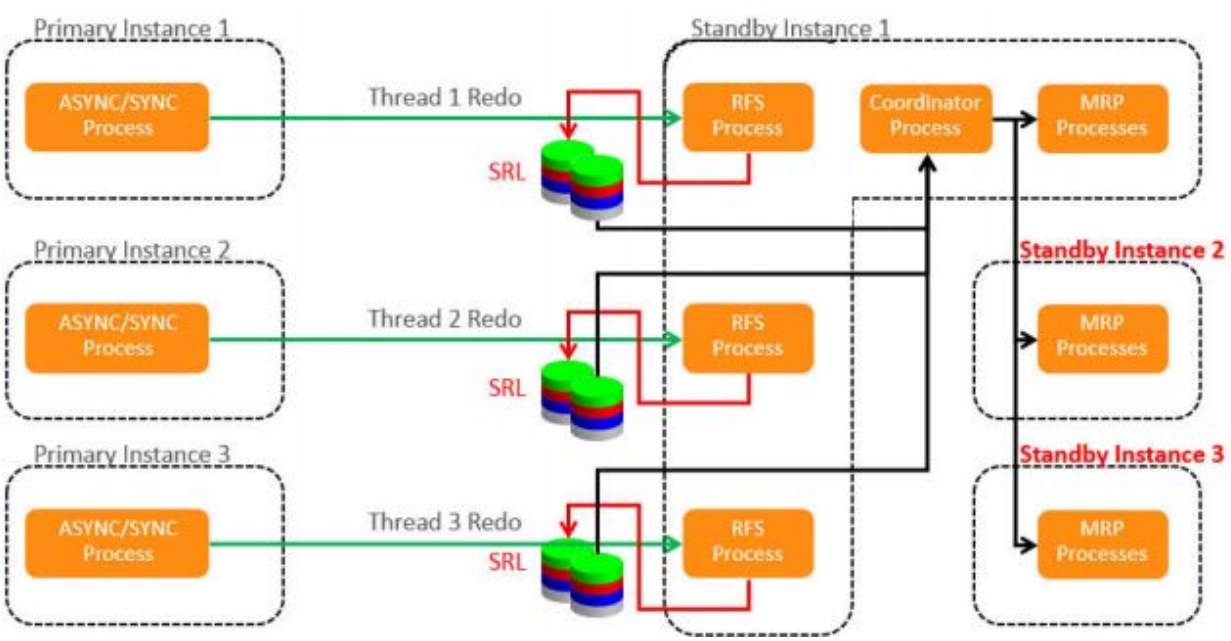
8.12.1. 12cR2 Miscealleaneous New Features

Oracle Database In-Memory column store (IM column store) is now supported on standby databases in Oracle Active Data Guard (ADG) environments.

You can use the new **FARSYNC** option on the RMAN DUPLICATE command to create an Oracle Data Guard far sync instance. You can do so using either active database duplication or backup-based duplication.

Password file changes done on the primary database are now automatically propagated to standby databases. The only exception to this is far sync instances. The password file is updated on the standby when the redo is applied

Parallel Redo Log Apply on Standby Instances



Remark: in 18c block change tracking was supported in combination with this feature and in 19c In Memory Columns Store (IMCS).

8.12.2. 18c Miscealleaneous New Features

Standby Nologging for Data Availability and Load Performance:
In versions before 18c, when configuring Data Guard, the only option we had for the logging property of the databases other than the default, was enabling FORCE LOGGING as:
ALTER DATABASE FORCE LOGGING;
Starting with 18c, we have the following new options:
ALTER DATABASE SET STANDBY NOLOGGING FOR DATA AVAILABILITY;
or
ALTER DATABASE SET STANDBY NOLOGGING FOR LOAD PERFORMANCE;

- STANDBY NOLOGGING FOR DATA AVAILABILITY mode causes the load operation to send the loaded data to each standby through its own connection to the standby. The commit is delayed until all the standbys have applied the data as part of running managed recovery in an Active Data Guard environment.

- STANDBY NOLOGGING FOR LOAD PERFORMANCE is similar to the previous mode except that the loading process can stop sending the data to the standbys if the network cannot keep up with the speed at which data is being loaded to the primary. In this mode it is possible that the standbys may have missing data, but each standby automatically fetches the data from the primary as a normal part of running managed recovery in an Active Data Guard environment.

One important thing to notice here: these options work in an Active Data Guard environment, which is and extra cost option and needs to be licensed on top of Enterprise Edition.

The **database buffer cache state** will be maintained on an ADG standby during a role change. Also, **sessions** will be preserved on the Standby Database during a Switchover.

8.12.3. 19c Miscealleaneous New Features

Automatically Deleting Flashback Logs: Fast recovery area management and database health are improved by automatically deleting flashback logs that are beyond the retention period.

In Oracle Database 19c, the DBA can tune the amount of wait time for this detection period by using two new parameters, **DATA_GUARD_MAX_IO_TIME** and **DATA_GUARD_MAX_LONGIO_TIME**.

These parameters allow the waits times to be tuned for a specific Data Guard configuration based on the user network and Disk I/O behavior. Users can now tune Oracle Data Guard automatic outage resolution to fit their specific needs.

9. Supplement 1 Output RMAN Duplicate statement

```
RMAN> DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE DORECOVER NOFILENAMECHECK;

Starting Duplicate Db at 30-OCT-2020 20:06:39
using target database control file instead of recovery catalog
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=252 device type=DISK
current log archived

contents of Memory Script:
{
   backup as copy reuse
   passwordfile auxiliary format  '/u01/app/oracle/dbs/orapwprod_db2'   ;
}
executing Memory Script

Starting backup at 30-OCT-2020 20:06:41
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=277 device type=DISK
Finished backup at 30-OCT-2020 20:06:44
duplicating Online logs to Oracle Managed File (OMF) location
duplicating Datafiles to Oracle Managed File (OMF) location

contents of Memory Script:
{
   sql clone "alter system set  control_files =
  ''/home/oracle/oradata/PROD_DB2/controlfile/o1_mf_hsrwln8t_.ctl'',
''/media/sf_Shared/PROD_DB2/controlfile/o1_mf_hsrwln91_.ctl'' comment=
 ''Set by RMAN'' scope=spfile";
   restore clone from service  'prod_db1' standby controlfile;
}
executing Memory Script

sql statement: alter system set  control_files =
''/home/oracle/oradata/PROD_DB2/controlfile/o1_mf_hsrwln8t_.ctl'',
''/media/sf_Shared/PROD_DB2/controlfile/o1_mf_hsrwln91_.ctl'' comment= ''Set by RMAN'' scope=spfile

Starting restore at 30-OCT-2020 20:06:44
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: restoring control file
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
output file name=/home/oracle/oradata/PROD_DB2/controlfile/o1_mf_hsrwlqdj_.ctl
output file name=/media/sf_Shared/PROD_DB2/controlfile/o1_mf_hsrwlql9_.ctl
Finished restore at 30-OCT-2020 20:06:48

contents of Memory Script:
{
   sql clone 'alter database mount standby database';
}
executing Memory Script

sql statement: alter database mount standby database

contents of Memory Script:
{
   set newname for clone tempfile  1 to new;
   set newname for clone tempfile  2 to new;
   set newname for clone tempfile  3 to new;
   set newname for clone tempfile  4 to new;
   switch clone tempfile all;
   set newname for clone datafile  1 to new;
   set newname for clone datafile  3 to new;
   set newname for clone datafile  4 to new;
   set newname for clone datafile  5 to new;
   set newname for clone datafile  6 to new;
   set newname for clone datafile  7 to new;
```

```
   set newname for clone datafile  8 to new;
   set newname for clone datafile  9 to new;
   set newname for clone datafile  10 to new;
   set newname for clone datafile  11 to new;
   set newname for clone datafile  12 to new;
   set newname for clone datafile  13 to new;
   set newname for clone datafile  14 to new;
   set newname for clone datafile  15 to new;
   restore
   from  nonsparse   from service
 'prod_db1'   clone database
   ;
   sql 'alter system archive log current';
}
executing Memory Script

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

renamed tempfile 1 to /home/oracle/oradata/PROD_DB2/datafile/o1_mf_temp_%u_.tmp in control file
renamed tempfile 2 to
/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_temp_%u_.tmp in control
file
renamed tempfile 3 to
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_temp_%u_.tmp in control
file
renamed tempfile 4 to
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_temp_%u_.tmp in control
file

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

Starting restore at 30-OCT-2020 20:06:53
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:15
channel ORA_AUX_DISK_1: starting datafile backup set restore
```

```
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00003 to
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_sysaux_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:08
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00004 to
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_undotbs1_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:07
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00005 to
/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_system_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00006 to
/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_sysaux_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:07
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00007 to
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_users_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00008 to
/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_undotbs1_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00009 to
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_system_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00010 to
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_sysaux_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00011 to
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_undotbs1_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00012 to
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_users_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00013 to
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_system_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00014 to
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_sysaux_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
```

```
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00015 to
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_undotbs1_%u_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
Finished restore at 30-OCT-2020 20:07:53

sql statement: alter system archive log current
current log archived

contents of Memory Script:
{
   restore clone force from service  'prod_db1'
          archivelog from scn  2365363;
   switch clone datafile all;
}
executing Memory Script

Starting restore at 30-OCT-2020 20:07:54
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting archived log restore to default destination
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: restoring archived log
archived log thread=1 sequence=28
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting archived log restore to default destination
channel ORA_AUX_DISK_1: using network backup set from service prod_db1
channel ORA_AUX_DISK_1: restoring archived log
archived log thread=1 sequence=29
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
Finished restore at 30-OCT-2020 20:07:57

datafile 1 switched to datafile copy
input datafile copy RECID=18 STAMP=1055189277 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf
datafile 3 switched to datafile copy
input datafile copy RECID=19 STAMP=1055189277 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_sysaux_hsrwmg0l_.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=20 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_undotbs1_hsrwmos4_.dbf
datafile 5 switched to datafile copy
input datafile copy RECID=21 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_system_hsrwmx7c_.dbf
datafile 6 switched to datafile copy
input datafile copy RECID=22 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_sysaux_hsrwn07p_.dbf
datafile 7 switched to datafile copy
input datafile copy RECID=23 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_users_hsrwn7fo_.dbf
datafile 8 switched to datafile copy
input datafile copy RECID=24 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_undotbs1_hsrwn8md_.d
bf
datafile 9 switched to datafile copy
input datafile copy RECID=25 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_system_hsrwn9v9_.dbf
datafile 10 switched to datafile copy
input datafile copy RECID=26 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_sysaux_hsrwnf2n_.dbf
datafile 11 switched to datafile copy
input datafile copy RECID=27 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_undotbs1_hsrwnj7z_.d
bf
datafile 12 switched to datafile copy
input datafile copy RECID=28 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_users_hsrwnkc6_.dbf
datafile 13 switched to datafile copy
input datafile copy RECID=29 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_system_hsrwnlm9_.dbf
datafile 14 switched to datafile copy
input datafile copy RECID=30 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_sysaux_hsrwnor5_.dbf
```

```
datafile 15 switched to datafile copy
input datafile copy RECID=31 STAMP=1055189278 file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_undotbs1_hsrwns0c_.d
bf

contents of Memory Script:
{
   set until scn  2365601;
   recover
   standby
   clone database
    delete archivelog
   ;
}
executing Memory Script

executing command: SET until clause

Starting recover at 30-OCT-2020 20:07:58
using channel ORA_AUX_DISK_1

starting media recovery

archived log for thread 1 with sequence 28 is already on disk as file
/media/sf_Shared/PROD_DB2/archivelog/2020_10_30/o1_mf_1_28_hsrwnvlc_.arc
archived log for thread 1 with sequence 29 is already on disk as file
/media/sf_Shared/PROD_DB2/archivelog/2020_10_30/o1_mf_1_29_hsrwnwr7_.arc
archived log file name=/media/sf_Shared/PROD_DB2/archivelog/2020_10_30/o1_mf_1_28_hsrwnvlc_.arc
thread=1 sequence=28
archived log file name=/media/sf_Shared/PROD_DB2/archivelog/2020_10_30/o1_mf_1_29_hsrwnwr7_.arc
thread=1 sequence=29
media recovery complete, elapsed time: 00:00:01
Finished recover at 30-OCT-2020 20:08:02

contents of Memory Script:
{
   delete clone force archivelog all;
}
executing Memory Script

released channel: ORA_DISK_1
released channel: ORA_AUX_DISK_1
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=277 device type=DISK
deleted archived log
archived log file name=/media/sf_Shared/PROD_DB2/archivelog/2020_10_30/o1_mf_1_28_hsrwnvlc_.arc RECID=1
STAMP=1055189275
deleted archived log
archived log file name=/media/sf_Shared/PROD_DB2/archivelog/2020_10_30/o1_mf_1_29_hsrwnwr7_.arc RECID=2
STAMP=1055189276
Deleted 2 objects

Finished Duplicate Db at 30-OCT-2020 20:08:09
```

10. Supplement 2: Output Recover Standby Database from service

```
RMAN> recover standby database from service prod_primary;

Starting recover at 07-NOV-20
using target database control file instead of recovery catalog
Executing: alter database flashback off
Executing: alter database disable block change tracking
Oracle instance started

Total System Global Area    1258287344 bytes

Fixed Size                     9134320 bytes
Variable Size                402653184 bytes
Database Buffers             838860800 bytes
Redo Buffers                   7639040 bytes

contents of Memory Script:
{
   restore standby controlfile from service  'prod_primary';
   alter database mount standby database;
}
executing Memory Script

Starting restore at 07-NOV-20
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=19 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
output file name=/home/oracle/oradata/PROD_DB2/controlfile/o1_mf_hsrwlqdj_.ctl
output file name=/media/sf_Shared/PROD_DB2/controlfile/o1_mf_hsrwlql9_.ctl
Finished restore at 07-NOV-20

released channel: ORA_DISK_1
Statement processed
Executing: alter system set standby_file_management=manual

contents of Memory Script:
{
set newname for tempfile  1 to
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_temp_hss0yrxo_.tmp";
set newname for tempfile  2 to
 "/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_temp_hss0yvj0_.tmp";
set newname for tempfile  3 to
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_temp_hsvc8m2h_.tmp";
set newname for tempfile  4 to
 "/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_temp_hsvc94cs_.tmp";
   switch tempfile all;
set newname for datafile  1 to
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf";
set newname for datafile  3 to
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_sysaux_hsrwmg0l_.dbf";
set newname for datafile  4 to
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_undotbs1_hsrwmos4_.dbf";
set newname for datafile  5 to
 "/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_system_hsrwmx7c_.dbf";
set newname for datafile  6 to
 "/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_sysaux_hsrwn07p_.dbf";
set newname for datafile  7 to
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_users_hsrwn7fo_.dbf";
set newname for datafile  8 to

"/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_undotbs1_hsrwn8md_.dbf";
set newname for datafile  9 to
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_system_hsrwn9v9_.dbf";
set newname for datafile  10 to
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_sysaux_hsrwnf2n_.dbf";
```

```
set newname for datafile  11 to

"/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_undotbs1_hsrwnj7z_.dbf";
set newname for datafile  12 to
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_users_hsrwnkc6_.dbf";
set newname for datafile  13 to
 "/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_system_hsrwnlm9_.dbf";
set newname for datafile  14 to
 "/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_sysaux_hsrwnor5_.dbf";
set newname for datafile  15 to

"/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_undotbs1_hsrwns0c_.dbf";
set newname for datafile  19 to
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_catalog_hsxlnjcc_.dbf";
   catalog datafilecopy  "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf",
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_sysaux_hsrwmg0l_.dbf",
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_undotbs1_hsrwmos4_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_system_hsrwmx7c_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_sysaux_hsrwn07p_.dbf",
 "/home/oracle/oradata/PROD_DB2/datafile/o1_mf_users_hsrwn7fo_.dbf",

"/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_undotbs1_hsrwn8md_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_system_hsrwn9v9_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_sysaux_hsrwnf2n_.dbf",

"/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_undotbs1_hsrwnj7z_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_users_hsrwnkc6_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_system_hsrwnlm9_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_sysaux_hsrwnor5_.dbf",

"/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_undotbs1_hsrwns0c_.dbf",
 "/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_catalog_hsxlnjcc_.dbf";
   switch datafile all;
}
executing Memory Script

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

Starting implicit crosscheck backup at 07-NOV-20
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=264 device type=DISK
Crosschecked 16 objects
Finished implicit crosscheck backup at 07-NOV-20

Starting implicit crosscheck copy at 07-NOV-20
using channel ORA_DISK_1
Finished implicit crosscheck copy at 07-NOV-20

searching for all files in the recovery area
cataloging files...
cataloging done

List of Cataloged Files
=======================
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_31_hst6ktdf_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_32_hst6pspj_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_33_hst6zmvv_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_34_hst6zqm8_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_35_hstd5qt3_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_36_hstw3zhd_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_37_hsvgd8sn_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_10_31/o1_mf_1_38_hsvk4nl5_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_39_hswy09r2_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_40_hsx1fb0x_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_41_hsx1ghc6_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_42_hsx22cby_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_43_hsxf06ch_.arc
```

```
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_44_hsxf0471_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_45_hsxlt3wm_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_46_hsxmmbdg_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_47_hsxpxdb9_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_01/o1_mf_1_48_hsxy2mk2_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_10_ht0rh2f4_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_11_ht0rh5hx_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_12_ht0rh5j4_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_1_hsznf4z4_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_2_hszoglk0_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_3_ht0dz0bh_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_49_hszfvhky_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_4_ht0fcc8d_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_50_hsznf90n_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_5_ht0ftr0y_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_6_ht0fxh5w_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_7_ht0qrsk0_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_8_ht0qsbsk_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_02/o1_mf_1_9_ht0qsw1w_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_13_ht2g38g2_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_14_ht335y3l_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_15_ht3dz2jg_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_1_ht3fd14o_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_2_ht3fd1jk_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_3_ht3fsls0_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_4_ht3fspgt_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_03/o1_mf_1_5_ht3fsosd_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_10_ht62nbnh_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_11_ht63sn2h_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_12_ht63sq3b_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_13_ht63sq33_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_6_ht59jdcb_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_7_ht5z6bk4_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_8_ht62lx67_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_04/o1_mf_1_9_ht62mk73_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_05/o1_mf_1_12_ht8lo0qf_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_05/o1_mf_1_13_ht8lo18s_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_05/o1_mf_1_14_ht7vxckf_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_05/o1_mf_1_14_ht8lo274_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_05/o1_mf_1_15_ht8lo2rf_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_05/o1_mf_1_1_ht8lnxl3_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_05/o1_mf_1_2_ht8lrmx9_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_06/o1_mf_1_3_htb3gd4c_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_07/o1_mf_1_4_htdl82z4_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_07/o1_mf_1_5_htdld17d_.arc
File Name: /media/sf_Shared/PROD_DB2/archivelog/2020_11_07/o1_mf_1_6_htdo1l27_.arc
File Name: /media/sf_Shared/PROD_DB2/autobackup/2020_10_31/o1_mf_s_1055193925_hst6fwfn_.bkp
File Name: /media/sf_Shared/PROD_DB2/autobackup/2020_10_31/o1_mf_s_1055240241_hstgjr7k_.bkp
File Name: /media/sf_Shared/PROD_DB2/autobackup/2020_11_02/o1_mf_s_1055436875_ht0ggx3z_.bkp
File Name: /media/sf_Shared/PROD_DB2/autobackup/2020_11_02/o1_mf_s_1055447713_ht0s1tvz_.bkp
File Name: /media/sf_Shared/PROD_DB2/autobackup/2020_11_04/o1_mf_s_1055535018_ht4omgjl_.bkp
File Name: /media/sf_Shared/PROD_DB2/autobackup/2020_11_05/o1_mf_s_1055662304_ht7bnymz_.bkp

renamed tempfile 1 to /home/oracle/oradata/PROD_DB2/datafile/o1_mf_temp_hss0yrxo_.tmp in control file
renamed tempfile 2 to
/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_temp_hss0yvj0_.tmp in
control file
renamed tempfile 3 to
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_temp_hsvc8m2h_.tmp in
control file
renamed tempfile 4 to
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_temp_hsvc94cs_.tmp in
control file

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME
```

```
executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

cataloged datafile copy
datafile copy file name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf RECID=4
STAMP=1055837760
cataloged datafile copy
datafile copy file name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_sysaux_hsrwmg0l_.dbf RECID=5
STAMP=1055837760
cataloged datafile copy
datafile copy file name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_undotbs1_hsrwmos4_.dbf RECID=6
STAMP=1055837760
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_system_hsrwmx7c_.dbf
RECID=7 STAMP=1055837760
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_sysaux_hsrwn07p_.dbf
RECID=8 STAMP=1055837760
cataloged datafile copy
datafile copy file name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_users_hsrwn7fo_.dbf RECID=9
STAMP=1055837760
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_undotbs1_hsrwn8md_.d
bf RECID=10 STAMP=1055837760
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_system_hsrwn9v9_.dbf
RECID=11 STAMP=1055837760
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_sysaux_hsrwnf2n_.dbf
RECID=12 STAMP=1055837760
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_undotbs1_hsrwnj7z_.d
bf RECID=13 STAMP=1055837760
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_users_hsrwnkc6_.dbf
RECID=14 STAMP=1055837761
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_system_hsrwnlm9_.dbf
RECID=15 STAMP=1055837761
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_sysaux_hsrwnor5_.dbf
RECID=16 STAMP=1055837761
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_undotbs1_hsrwns0c_.d
bf RECID=17 STAMP=1055837761
```

```
cataloged datafile copy
datafile copy file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_catalog_hsxlnjcc_.db
f RECID=18 STAMP=1055837761

datafile 1 switched to datafile copy
input datafile copy RECID=4 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf
datafile 3 switched to datafile copy
input datafile copy RECID=5 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_sysaux_hsrwmg0l_.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=6 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_undotbs1_hsrwmos4_.dbf
datafile 5 switched to datafile copy
input datafile copy RECID=7 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_system_hsrwmx7c_.dbf
datafile 6 switched to datafile copy
input datafile copy RECID=8 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_sysaux_hsrwn07p_.dbf
datafile 7 switched to datafile copy
input datafile copy RECID=9 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/datafile/o1_mf_users_hsrwn7fo_.dbf
datafile 8 switched to datafile copy
input datafile copy RECID=10 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/B2CFC44F4EFA7A28E0533800A8C067FE/datafile/o1_mf_undotbs1_hsrwn8md_.d
bf
datafile 9 switched to datafile copy
input datafile copy RECID=11 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_system_hsrwn9v9_.dbf
datafile 10 switched to datafile copy
input datafile copy RECID=12 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_sysaux_hsrwnf2n_.dbf
datafile 11 switched to datafile copy
input datafile copy RECID=13 STAMP=1055837760 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_undotbs1_hsrwnj7z_.d
bf
datafile 12 switched to datafile copy
input datafile copy RECID=14 STAMP=1055837761 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_users_hsrwnkc6_.dbf
datafile 13 switched to datafile copy
input datafile copy RECID=15 STAMP=1055837761 file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_system_hsrwnlm9_.dbf
datafile 14 switched to datafile copy
input datafile copy RECID=16 STAMP=1055837761 file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_sysaux_hsrwnor5_.dbf
datafile 15 switched to datafile copy
input datafile copy RECID=17 STAMP=1055837761 file
name=/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_undotbs1_hsrwns0c_.d
bf
datafile 19 switched to datafile copy
input datafile copy RECID=18 STAMP=1055837761 file
name=/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_catalog_hsxlnjcc_.db
f
Executing: alter database rename file '/media/sf_Shared/PROD_DB1/onlinelog/o1_mf_1_hsp9mdgp_.log' to
'/home/oracle/oradata/PROD_DB2/onlinelog/o1_mf_1_hsrwo3qz_.log'
Executing: alter database rename file
'/u01/app/oracle/oradata/PROD_DB1/onlinelog/o1_mf_1_hsp9mddd_.log' to
'/media/sf_Shared/PROD_DB2/onlinelog/o1_mf_1_hsrwo3vz_.log'
Executing: alter database rename file '/media/sf_Shared/PROD_DB1/onlinelog/o1_mf_2_hsp9j4od_.log' to
'/home/oracle/oradata/PROD_DB2/onlinelog/o1_mf_2_hsrwo4mo_.log'
Executing: alter database rename file
'/u01/app/oracle/oradata/PROD_DB1/onlinelog/o1_mf_2_hsp9j4lo_.log' to
'/media/sf_Shared/PROD_DB2/onlinelog/o1_mf_2_hsrwo4po_.log'
Executing: alter database rename file '/media/sf_Shared/PROD_DB1/onlinelog/o1_mf_3_hsp9jkf5_.log' to
'/home/oracle/oradata/PROD_DB2/onlinelog/o1_mf_3_hsrwo5hh_.log'
Executing: alter database rename file
'/u01/app/oracle/oradata/PROD_DB1/onlinelog/o1_mf_3_hsp9jkbm_.log' to
'/media/sf_Shared/PROD_DB2/onlinelog/o1_mf_3_hsrwo5m7_.log'

contents of Memory Script:
{
  recover database from service  'prod_primary';
```

```
}
executing Memory Script

Starting recover at 07-NOV-20
using channel ORA_DISK_1
skipping datafile 5; already restored to SCN 2143386
skipping datafile 6; already restored to SCN 2143386
skipping datafile 8; already restored to SCN 2143386
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00001:
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_system_hsrwlypz_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00003:
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_sysaux_hsrwmg0l_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00004:
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_undotbs1_hsrwmos4_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00007:
/home/oracle/oradata/PROD_DB2/datafile/o1_mf_users_hsrwn7fo_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:02
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00009:
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_system_hsrwn9v9_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00010:
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_sysaux_hsrwnf2n_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00011:
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_undotbs1_hsrwnj7z_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00012:
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_users_hsrwnkc6_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00013:
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_system_hsrwnlm9_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00014:
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_sysaux_hsrwnor5_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:02
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00015:
/home/oracle/oradata/PROD_DB2/B2D1135E74377732E0533800A8C08EB0/datafile/o1_mf_undotbs1_hsrwns0c_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting incremental datafile backup set restore
channel ORA_DISK_1: using network backup set from service prod_primary
destination for restore of datafile 00019:
/home/oracle/oradata/PROD_DB2/B2D0DE7BFA2C3108E0533800A8C0CE1B/datafile/o1_mf_catalog_hsxlnjcc_.dbf
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01

starting media recovery

media recovery complete, elapsed time: 00:00:00
Finished recover at 07-NOV-20
```

```
Reenabling controlfile options for auxiliary database
Executing: alter database enable block change tracking using file
'/home/oracle/oradata/PROD_DB2/changetracking/o1_mf_hss0l48g_.chg'
flashback needs to be reenabled on standby open
Executing: alter system set standby_file_management=auto
Finished recover at 07-NOV-20
```