Oracle 10G Data Guard Real Time Apply & Flashback Database Features – An Implementation Experience

TJ Mitra
Datacom Systems Limited
NZOUG 2007 Presentation

The Background

My recent project involves upgrading one of our Client's Banking databases from Oracle version 9i Release 9.2 to Oracle version 10G Release 10.2.

The Pre-Production environment has Primary and Standby Databases hosted on different Solaris Servers geographically located in two separate cities.

This paper tries to analyse & discuss various high availability aspects around Oracle 10G R2 Data Guard and usage of Flashback Database features.

Analysis of the Current environment

- Because of the Banking environment, the high availability & the integrity of the Data are of mandatory requirements
- Currently the Oracle version is Oracle 9.2.0.6.0
- Data Guard environment is Physical Standby
- Log Transport from Primary to Standby happens using Archiver(ARCH), not Log Writer(LGWR)
- Standby Mode is Maximum Performance
- Network Transmission Mode is ASYNC
- The Redo Log Sizes are 512KB
- Log shipping happens only when the Primary Redo Log fills up and switches
- This Data Guard version doesn't support Real-Time Apply but uses Redo Apply
- Redo Apply means Redo generated at Primary is applied to Standby only when there is a log switch at the Primary
- There is no Automatic Apply Delay set in the Standby meaning as soon as Archive Log arrives at the Standby host, it gets applied onto the Standby database
- Operations Menu has an option to disable the Log Apply to the Standby when desired, that will stop Standby following Primary
- Because of the smaller size of the Redo Logs, very frequent log switches have been observed during Batch Runs
- There is no flashback mechanism available in the current system
- Flashback is an Oracle RDBMS 10G Technology feature which allows database to be rewound back in the past based on Time Stamp or System Change Number(SCN) or a defined Restore point
- Using Oracle Data Guard Role Transition Services commands embedded in the Operations Menu, Switchover & Switchback happens between Primary & Standby Sites at regular month intervals
- Capability of Failover to the Standby Database after a Primary failure has been built into the Operations Menu

Following were the special considerations when the Current System was designed:

- Oracle Redo Log size was chosen as 512k to be compatible with the then available Network Bandwidth & Latency and also to minimize the risk of data loss
- Standby Mode was chosen as Maximum Performance
- Because redo transmission happened over the Wide Area Network(WAN), Maximum Protection mode was not chosen as it needed highly reliable network with low latency, high bandwidth, resilience and redundancy
- In Maximum Protection mode, redo is not committed on Primary unless it is also committed on Standby, so that could have negative performance impact on the Primary if redo transmission over slow WAN causes Redo Apply on Standby to become slow
- Also, in Maximum Protection mode if the Standby is shutdown that will cause Primary to shutdown

Following are the requirements for the Application

- Data/Transaction Loss should be minimum in case of a Failover to Standby following Primary failure
- There should be no performance impact on the Primary during Log Shipping
- Primary must not shutdown for any Redo Transport or Redo Apply problem
- There will not be any Redo Log Apply Delay on the Standby
- The Standby should try to follow the Primary as closely as possible
- But, at the same time, it should be possible to stop the Log Apply on Standby when and if desired or required
- If the Primary is rewound in time, then the Standby also should be rewound to the same point in time.
- E.g. if there is any problem in the Primary database e.g. failed Batch, then through Menu option, Operations Support should be able to rewind both the Primary & Standby databases back into the past to the point in time just before the problem has happened
- Capability of Alternate Site Switchover & Switchback using Oracle Data Guard Role Transition Services must continue to be available
- Capability of Failover to the Standby Database after a Primary failure must continue to be available
- As future enhancement, the possibility of using the DR database as Reporting Server is to be explored

Proposed Enhancements

To address the above mentioned requirements, the following Oracle 10G features were explored:

- Managed Recovery of the Physical Standby in Real Time Apply mode to minimise Data Loss on the Standby in the event of Primary failure

- Implementation of Flashback database feature in both Primary & Standby introducing the capability of rewinding both the databases to a defined state in the past without going through time-consuming Media Restore & Recovery Process

Oracle RDBMS Version

- Oracle RDBMS 10G will be used
- Initially Oracle 10G R1 was chosen, but because of few bugs (few mentioned below) it has been decided Oracle 10G R2 is to be used
- As Patch Release 10.2.0.3 has addressed fixes around flashback, this patch release is to be used
- Current Oracle 10G Patch version on the Servers is 10.2.0.2, so all Oracle Servers starting from Development to UAT to Pre-Production will be patched to Oracle 10.2.0.3
- Oracle Critical Patch released as of January 2007 will be applied on all Oracle Servers

Data Guard Environment

- The Data Guard Environment will continue to remain Physical Standby
- As the Primary/Standby Pair operates over a Wide Area Network and also as of other operational issues, Oracle 10G Data Guard Standby Mode will remain Maximum Performance, though possibility of using Maximum Protection mode could be explored
- Log Transport from Primary to Standby will happen using Log Writer not Archiver
- LGWR ASYNC LNS Buffer was set to 50MB as available in Oracle 10G R1 (10MB in Oracle 9i). In Oracle 10G R2, as further enhancement LNS process, instead of reading from in-memory buffer, reads from Primary online redo log, so is not constrained to in-memory buffer filling up situation. Also, Net_timeout attribute is not needed in Log_archive_dest_2 parameter as from Oracle 10.2, LGWR never waits for LNS.
- Send Data Unit (SDU) will be set to 32k for optimum performance of Sqlnet

WAN improvement

- WAN performance has been improved by five fold. This was possible by upgrading all the network gears and increased WAN bandwidth. Additional carriers were added for redundancy

Oracle 10G Data Guard Standby Real-Time Apply

- Oracle Data Guard 10G Standby Real-Time Apply mechanism will be used
- Real Time Apply will use LGWR on the Primary to write redo data to Standby Redo log on the Standby and Log Apply Services can apply the redo data in real-time without the need of the current standby redo log being archived

- In Real Time Apply, once a transaction is committed on the Primary, the committed changes will be available on the Standby in Real Time even without switching the log at the Primary
- Operations Menu will continue to have the option to disable the Log Apply to Standby when desired, that will stop Standby following Primary
- Because of the Real-Time Apply, the Database On-line Redo Log Sizes can be increased to 20MB each as suggested by Oracle 10G RDBMS Advisor during test Batch Runs. This will absorb frequent log switches during batch runs
- For good Real-Time Apply Performance, Oracle suggests Wide Area Network(WAN) Return Trip Time(RTT) to remain below 100ms
- Current WAN RTT, even during most active Batch Period, has been observed to be in the range of 15-20ms, well below the critical range
- A call was logged with Oracle to confirm that Oracle 10G Data Guard Real-Time Apply doesn't introduce any corruption of Data
- Oracle Corporation has confirmed, as of now, for usage of this feature no bug has been observed which could potentially lead to Data Corruption

Oracle 10G Flashback Database Feature

- Oracle 10G Flashback Database feature will be enabled on both Primary & Standby
- Flashbacking capability will be defined for 5 days
- Also, Named Restore Points, as available in Oracle 10G R2, will be used
- The Restore Points will be created with 'guarantee flashback database' option
- Operations Menu will have option to manually register Restore points for flashback
- Menu will have option to display the Restore points & the time each Restore point the database can flashback to
- Menu will have option to issue command to flashback the database to a desired Restore Point
- Menu will have option to drop old & obsolete Restore Points
- For Batch Application, there will be a named Restore Point called 'PreBatch' automatically dropped and re-created from within the Application at the beginning of the Batch every day.
- In case of Batch failure, instead of a DBA Callout, using Menu, Operations Support will Flashback the database to the Restore Point 'PreBatch'
- Currently to recover from that scenario, a DBA call out is necessary and would take about approximate 3-4 hours to recover the Database from the Backup
- Also, currently after a Failover, DBA needs to rebuild the original Primary as the current Standby from a most recent backup of the new Primary
- If Flashback feature is enabled, then this lengthy procedure can be avoided because a Flashback enabled Primary database can be easily rewound to a consistent point and then be made as new Standby and start receiving & applying logs from the new Primary
- Another call was logged with Oracle to confirm that Oracle 10G Flashback Database feature doesn't introduce any corruption of Data
- Oracle Corporation has confirmed, as of now, for the usage of this feature no bug has been observed which could potentially lead to Data Corruption
- Business has been made aware of that Flashback Database is applicable to address Logical Errors only, it is not a replacement for Database Media Recovery required after a Media failure.

Possibility of using Standby Database as Reporting Server

- This is very much possible, but when the Standby database is opened for Reporting it can not apply logs, so DR environment is compromised
- To address that situation, two Physical Standby Databases could be configured, One perpetually remaining in Managed Recovery mode providing DR, and the Other remaining in Managed Recovery mode but earmarked for Reporting, and the latter can be taken out of Managed Recovery mode when desired and can be opened in Read Only mode for Reporting
- The Change of State of a Physical Standby between Managed Recovery Mode & Read Only Mode is not automatic and has to be achieved using Operations Menu when desired or through the Report Job at the beginning of running report
- Possibility of using a Logical Standby for Reporting could be explored (Business has been informed that using Oracle Application Express as Reporting tool, the usage of Logical Standby as Reporting Server has been implemented in another Client Site)
- Oracle Data Guard Logical Standby, unlike Physical Standby, is an Open Database where Oracle applies Sqls instead of Redo Logs
- Oracle Logical Standby also supports Real Time Apply and Flashback features
- In previous versions of Oracle, a Physical Standby could have only two modes – Managed Recovery mode or Read Only mode
- Oracle 10G R2 has introduced a Read Write Mode of a Physical Standby, which gives the capability of opening Standby for not only doing Selects but also DMLs for running jobs, at the end of which it can be flashed back to a Restore Point and then can be converted back as Physical Standby and be put back in the Managed Recovery mode again

Overview of Oracle 10G Data Guard Technology

Oracle 10G Data Guard is vastly improved from its previous versions and addresses the High Availability in a very effective manner.

Oracle Data Guard offers two types of Standby features – Physical Standby & Logical Standby.

Physical Standby replicates data from the Primary by way of applying the Redo generated on the Primary.

Logical Standby replicates data from the Primary by applying sql statements mined by the Log miner from the Archive Redo logs. (Logical Standby implementation is described in detail in a previous presentation at NZOUG 2005 http://www.ausoug.org.au/pls/portal30/docs/FOLDER/NZOUG_CONF_07/MITRA+-+TAPAN.PDF)

Physical Standby can operate in 3 modes: Managed Recovery Mode, Read Only Mode & temporary Read-write Mode.

Oracle Data Guard has 3 protection modes – Maximum Protection, Maximum Availability & Maximum Performance.

Data Guard consists of 3 main services, namely:

- Log Transport Services, which control transfer of redo data from Primary to Standby in automated manner
- Log Apply Services, which apply redo data on the Standby

- Role Management Services, which handle the change of role of a database from Standby to Primary or from Primary to Standby using either Switchover/Switchback or Failover operation.

The difference between Redo Apply & Real-Time Apply

Normally, by default, Archiver processes will be responsible for Redo Transport from Primary to Standby.

Once a log switch happens on the Primary, the online redo log is archived in the Local Archive destination as pointed to by Log_archive_dest_1 by an Archiver process. Another Archiver process will then transmit the redo to the remote standby destination as indicated by Log_archive_dest_2. Data Guard Remote File Server (RFS) Process on the Standby then writes redo data from the Standby redo log file to archive redo log file. Log apply services then makes use of Managed Recovery Process (MRP) process to apply the redo to the standby database.

This method of propagating redo from the primary to standby is called Redo Apply and it happens only on log switch at the Primary.

When using Redo Apply mode, the status of MRP in v$managed_standby view will show as WAIT_FOR_LOG.

Real Time Apply, in contrast, uses either LGWR or Archiver on the Primary to write redo data to Standby Redo log on the Standby and Log Apply Services can apply the redo data in real-time without the need of the current standby redo log being archived. Once a transaction is committed on the Primary, the committed changes will be available on the Standby in Real Time even without switching the log.

When using Real Time Apply mode, the status of MRP in v$managed_standby view will show as APPLYING_LOG.

The Primary/Standby Pair Set up

The TNSNAME for the Primary is ‘proddb’.

The TNSNAME for the Standby is ‘standby_proddb’.

The Primary database must have to be in archivelog mode.

The Primary database will have the following Data Guard related init parameters:

```
######################################################################
## Data Guard related parameters

db_name='proddb'
db_unique_name=proddb

##COMMON TO BOTH PRIMARY AND STANDBY ROLES

LOG_ARCHIVE_CONFIG='DG_CONFIG=(proddb,standby_proddb)'
LOG_ARCHIVE_DEST_1='LOCATION=/d2/log_archive/proddb
```

```
                VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
                DB_UNIQUE_NAME=proddb'
LOG_ARCHIVE_DEST_2='SERVICE=standby_proddb lgwr async reopen=15
max_failure=10 optional net_timeout=30
                VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
                DB_UNIQUE_NAME=standby_proddb'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_MAX_PROCESSES=2
log_archive_format='%t_%s_%r.dbf'

#SPECIFIC TO STANDBY ROLE

STANDBY_FILE_MANAGEMENT=AUTO
STANDBY_ARCHIVE_DEST='/d2/log_archive/proddb'
FAL_SERVER=standby_proddb
FAL_CLIENT=proddb
##############################################################################
```

The Standby database will have the following init parameters

```
##############################################################################
# Data Guard related parameters

db_name='proddb'
db_unique_name=standby_proddb

##COMMON TO BOTH PRIMARY AND STANDBY ROLES

LOG_ARCHIVE_CONFIG='DG_CONFIG=(proddb,standby_proddb)'
LOG_ARCHIVE_DEST_1='LOCATION=/d2/log_archive/proddb
                VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
                DB_UNIQUE_NAME=standby_proddb'
LOG_ARCHIVE_DEST_2='SERVICE=proddb lgwr async reopen=15 max_failure=10
optional net_timeout=30
                VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
                DB_UNIQUE_NAME=proddb'
LOG_ARCHIVE_DEST_STATE_1=ENABLE
LOG_ARCHIVE_DEST_STATE_2=ENABLE
LOG_ARCHIVE_MAX_PROCESSES=2
log_archive_format='%t_%s_%r.dbf'

#SPECIFIC TO STANDBY ROLE

STANDBY_FILE_MANAGEMENT=AUTO
STANDBY_ARCHIVE_DEST='/d2/log_archive/proddb'
FAL_SERVER=proddb
FAL_CLIENT=standby_proddb

##############################################################################
```

TNSNAMES entries will be as follows :

```
proddb=
  ( DESCRIPTION =
    (SDU=32767)
    ( ADDRESS_LIST =
      ( ADDRESS =
        ( PROTOCOL = TCP )
        ( Host = host1)
        ( Port = 1700 )
      )
    )
    ( CONNECT_DATA =
      ( SID = proddb )
    )
  )


standby_proddb=
  ( DESCRIPTION =
    (SDU=32767)
    ( ADDRESS_LIST =
      ( ADDRESS =
        ( PROTOCOL = TCP )
        ( Host = host2)
        ( Port = 1700 )
      )
)
    ( CONNECT_DATA =
      ( SID = proddb )
    )
  )
```

In the respective Listener entries at the Primary & Standby hosts, SDU entry will be added as follows

```
SID_LIST_PRODDB =
  (SID_LIST =
    (SID_DESC =
      (SDU=32767)
      (SID_NAME = proddb)
      (ORACLE_HOME = /oracle/ora1020)
    )
  )
```

As part of the Real-Time Apply, the two requirements are as follows:

- Use Log Writer (LGWR) for Redo Transport from Primary to Standby
- Standby Redo Logs must be available on the Standby

Standby Redo Logs of size similar to the online redo logs were created at Primary

```
ALTER DATABASE ADD STANDBY LOGFILE GROUP 4
('/d1/proddb/proddb_standby_redolog4.dbf') size 20971520;

ALTER DATABASE ADD STANDBY LOGFILE GROUP 5
('/d1/proddb/proddb_standby_redolog5.dbf') size 20971520;

ALTER DATABASE ADD STANDBY LOGFILE GROUP 6
('/d1/proddb/proddb_standby_redolog6.dbf') size 20971520;
```

A cold backup of the Primary Database including datafiles, redo logfiles & standby redologfiles was transferred to the Standby Host.

A Standby Controlfile was created on the Primary and transferred to Standby.

On Primary the following command was used to create a standby controlfile

```
SQL> startup mount
SQL> alter database create standby controlfile as '/tmp/Proddb_standby.ctl';
SQL> alter database open;
```

On Standby

From this Standby controlfile, multiplexed controlfiles were created at their respective locations with proper names as mentioned in the init parameter file.

Then the Standby database is started in mounted mode:

```
SQL> startup mount
ORACLE instance started.
.
Database mounted.
SQL> select name, database_role,open_mode from v$database;

NAME      DATABASE_ROLE         OPEN_MODE
--------- ----------------------------  ----------
PRODDB   PHYSICAL STANDBY     MOUNTED
```

Following is the command to put the Standby in Redo Apply mode:

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

Database altered.

SQL> select PROCESS,STATUS from v$managed_standby;

PROCESS   STATUS
--------- ------------
ARCH      CONNECTED
ARCH      CONNECTED
MRP0      WAIT_FOR_LOG
```

Following is the command to put the Standby in Real Time Apply mode:

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

Database altered.

SQL>  select PROCESS,STATUS from v$managed_standby;

PROCESS   STATUS
---------   ------------
ARCH      CONNECTED
ARCH      CONNECTED
MRP0      APPLYING_LOG
```

In the Standby Alert Log it will show

```
MRP0: Background Managed Standby Recovery process started (proddb)
Managed Standby Recovery starting Real Time Apply
```

Cancelling Managed Recovery

At any time, to cancel Managed Recovery, either using Redo Apply or Real Time Apply, following command is used:

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

Database altered.
```

To put the Standby database in Read-Only mode, following command is used:

```
SQL> alter database open;
```

In pre Oracle 10G versions, to open a Standby database in Read-Only mode, the ‘read only’ clause was required:

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

This clause is not needed in Oracle 10G any more.

Oracle Flashback Database

Oracle Flashback Database is an Oracle 10G RDBMS technology feature which allows the database to be rewound back to the past to a particular SCN, Point in time timestamp or a defined Restore Point.

This is possible by defining a Flashback Area & a Flashback Retention Period and turning on the Flashback capability within the Database (Guaranteed Restore Points can be created with or without Flashback database feature ON). The database must be in Archivelog mode.

Following are the parameters defined for Flashback in Pre-Production:

```
### Parameters defined For Flashback
DB_FLASHBACK_RETENTION_TARGET=4320 (will be 7200 in Production)
DB_RECOVERY_FILE_DEST_SIZE=2G (will be 20G in Production)
DB_RECOVERY_FILE_DEST='/d2/flasharea'
```

As the transactions progress in the Database, Oracle 10G RDBMS RVWR process at regular intervals generates flashback logs in the flashback area. Flashback logs contain copies of the pre-images of all altered blocks in the datafiles.

An estimation of how much space needs to be added to the flash recovery area for flashback logs can be done by monitoring v$flashback_database_log view after running the database for a workload with a desired flashback retention target.

Flashback Recovery Area can be monitored using views v$flash_recovery_area_usage & v$recovery_file_dest.

```
SQL> select * from v$flash_recovery_area_usage ;

FILE_TYPE    PERCENT_SPACE_USED PERCENT_SPACE_RECLAIMABLE
NUMBER_OF_FILES
-------------------- ------------------ ------------------------ ---------------------------------------
CONTROLFILE                0                    0            0
ONLINELOG                  0                    0            0
ARCHIVELOG                 0                    0            0
BACKUPPIECE                0                    0            0
IMAGECOPY                  0                    0            0
FLASHBACKLOG             .38                    0            1

6 rows selected.

SQL> select * from v$recovery_file_dest;

NAME              SPACE_LIMIT  SPACE_USED SPACE_RECLAIMABLE NUMBER_OF_FILES
----------------  -------------------- ----------------   ------------------------------  --------------------------
/d2/flasharea       2147483648    8192000                  0                                1
```

Demonstration of Data Guard Real-Time Apply & Flashback Database features

First we do the Real-Time Apply Demo

On **PRIMARY**, count the number of rows of a table

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

NAME        DATABASE_ROLE
---------   ------------------------
PRODDB     PRIMARY

SQL> select count(*) from test;

  COUNT(*)
----------
      7
```

On **STANDBY**, check the number of rows and verify that it matches with that on Primary & then put the Standby in Real Time Apply mode

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

NAME        DATABASE_ROLE
---------  ------------------------------
PRODDB    PHYSICAL STANDBY

SQL>  alter database open;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------
7

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

Database altered.
```

On **PRIMARY**, increase the number of rows of the table from 7 to 56, commit & do not switch log

```
SQL> insert into test select * from test;

7 rows created.

SQL> /

14 rows created.

SQL> /

28 rows created.

SQL> commit;

Commit complete.

SQL> select count(*) from test;

  COUNT(*)
----------
        56
```

On **STANDBY**, check that it has applied redo from Primary in Real Time without a Log Switch on Primary

```
SQL> select process, status from v$managed_standby
SQL> /

PROCESS   STATUS
--------- ------------
ARCH      CLOSING
ARCH      CLOSING
RFS       IDLE
MRP0      APPLYING_LOG
RFS       IDLE

SQL> alter database recover managed standby database cancel;

database altered

SQL> alter database open;

database opened

SQL> select count(*) from test;

  COUNT(*)
----------
     56
```

Now we do the Flashback Database Demo

We turn on flashback in both Primary & Standby Databases now

On **PRIMARY**

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount
ORACLE instance started.
.
Database mounted.
SQL> select flashback_on from v$database;

FLASHBACK_ON
------------------
NO

SQL> alter database flashback on;

Database altered.

SQL> select flashback_on, database_role from v$database

FLASHBACK_ON       DATABASE_ROLE
-------------------------    ----------------------------
YES                          PRIMARY

SQL> alter database open
```

On **STANDBY**

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.
.
Database mounted.

SQL> select flashback_on from v$database;

FLASHBACK_ON
------------------
NO

SQL> alter database flashback on;

Database altered.
```

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

FLASHBACK_ON      DATABASE_ROLE
------------------------    ---------- ------------------
YES                         PHYSICAL STANDBY

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

Database altered.
```

On **PRIMARY,** find out the current scn and increase the number of rows from 56 to 224

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

  COUNT(*)
----------
        56

SQL> select current_scn from v$database;

CURRENT_SCN
-----------
     489802

SQL> insert into test select * from test;

56 rows created.

SQL> insert into test select * from test;

112 rows created.

SQL> commit;

Commit complete.

SQL> select count(*) from test;

  COUNT(*)
----------
       224
```

On **STANDBY**, check that number of rows has gone up to 224

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

Database altered.

SQL> alter database open;

Database altered.
```

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

  COUNT(*)
----------
       224
```

On **PRIMARY**, Flashback the database to the previously recorded SCN 489802 and check that the number of rows of the table has gone back to 56

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount;
ORACLE instance started.
.
Database mounted.

SQL> flashback database to scn 489802;

Flashback complete.

SQL> alter database open read only;

Database altered.

SQL>  select count(*) from test;

  COUNT(*)
----------
      56

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount
ORACLE instance started.
.
Database mounted.
SQL> alter database open resetlogs;

Database altered.

SQL>  select count(*) from test;

  COUNT(*)
----------
       56
```

On **PRIMARY**, obtain the value of (resetlogs_change# - 2) from the view v$database

```
SQL> select to_char(resetlogs_change# -2) from v$database;

TO_CHAR(RESETLOGS_CHANGE#-2)
----------------------------------------------------
489810
```

On **STANDBY**, obtain the current_scn from the standby database

```
SQL> select to_char(current_scn) from v$database;

TO_CHAR(CURRENT_SCN)
----------------------------------------
489914
```

If the current_scn on the Standby is more than (resetlogs_change# - 2) on the Primary, flashback the Standby database to (resetlogs_change# - 2) and open it to check that the number of rows has gone back successfully to 56

```
SQL> flashback standby database to scn 489810;

Flashback complete.

SQL> alter database open;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------
      56
```

Restore Points - Facts about Restore Points

From Oracle Version 10G R2, Oracle has introduced named Restore Points the database can flashback to.

There are two types of Restore Points – Normal Restore Point & Guaranteed Restore Point.

Creation of Restore Points makes entries in the controlfile.

Normal Restore Point is like a bookmark to specific time or scn.

Guaranteed Restore Point is different from Normal Restore Point, Guaranteed Restore Points can be created in the database with or without 'Flashback Database' feature on.

Guaranteed Restore Point, taken at a particular SCN ensures that the database can be flashed back to that scn, even if the flashback logging is not enabled in the database.

Guaranteed Restore Point with flashback logging enabled makes sure that flashback logs will be retained to be able to flashback the database to any point in time after the creation of earliest Guaranteed Restore Point.

Following is an example of how to define a Restore Point with a Guaranteed Clause, which will ensure that the database can be flashed back to the named restore point.

This example is without Flashback Database Feature on.

On **STANDBY**, create a Guaranteed Restore Point PRE_INSERT_STANDBY even without enabling Flashback database feature

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

FLASHBACK_ON
------------------------
NO

SQL> create restore point pre_insert_standby guarantee flashback database;

Restore point created.

SQL>  select * from v$restore_point;

 SCN DATABASE_INCARNATION# GUA STORAGE_SIZE       TIME                          NAME
---------------------------------------------------------------------------------------------------------------------------
612726            2                       YES     8192000     10-SEP-06 11.20.59.000000000 AM
PRE_INSERT_STANDBY
```

For normal restore point storage_size will be zero.

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

Database altered.

SQL> select flashback_on from v$database;

FLASHBACK_ON
---------------------------------
RESTORE POINT ONLY
```

On **PRIMARY**, create a Guaranteed Restore Point PRE_INSERT_PRIMARY. If the Flashback feature is not on, creation of first guaranteed restore point will need database to be in mount mode.

Then, Increase the number of rows in the table from 57344 to 114688

```
SQL> alter system switch logfile;

System altered.

SQL> create restore point pre_insert_primary guarantee flashback database;
create restore point pre_insert_primary guarantee flashback database
*
ERROR at line 1:
ORA-38784: Cannot create restore point 'PRE_INSERT_PRIMARY'.
ORA-38787: Creating the first guaranteed restore point requires mount mode when
flashback database is off.

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount;
ORACLE instance started.
.
.
Database mounted.
SQL>  create restore point pre_insert_primary guarantee flashback database;

Restore point created.

SQL> alter database open;

Database altered.

SQL> alter system switch logfile;

System altered.

SQL> select flashback_on from v$database;

FLASHBACK_ON
---------------------------------
RESTORE POINT ONLY
```

```
SQL>  select * from v$restore_point;

 SCN DATABASE_INCARNATION# GUA STORAGE_SIZE        TIME                          NAME
------------------------------------------------------------------------------------------------------------------------------
612935              2                          YES      8192000    10-SEP-06 11.24.55.000000000 AM
PRE_INSERT_PRIMARY

SQL> alter system switch logfile;

System altered.

SQL> select count(*) from test;

  COUNT(*)
----------------
     57344

SQL> insert into test select * from test;

57344 rows created.

SQL> select count(*) from test;

  COUNT(*)
----------------
    114688

SQL> commit;

Commit complete.
```

On **STANDBY**, cancel managed recovery and check that number of rows in the table ‘test’ has become the same as of Primary, then put it back into the Managed Recovery mode

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

Database altered.

SQL> alter database open;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------------
    114688

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

Database altered.
```

On **PRIMARY**, flashback database to the Guaranteed Restore Point PRE_INSERT_PRIMARY and check that the number of rows has gone back to 57344

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.
.
.
Database mounted.
SQL> flashback database to restore point pre_insert_primary;

Flashback complete.

SQL> alter database open read only;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------
     57344

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount;
ORACLE instance started.
.
.
Database mounted.
SQL> alter database open resetlogs;

Database altered.

SQL> alter system switch logfile;

System altered.

SQL>
```

On **STANDBY**, flashback standby database to the Guaranteed Restore Point PRE_INSERT_STANDBY and check that the row counts has gone back to 57344. Then put it back into Managed Recovery Mode

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

Database altered.
```

```
SQL> alter database open;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------
    114688

SQL> flashback standby database to restore point pre_insert_standby;

Flashback complete.

SQL> alter database open;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------
     57344

SQL> startup mount force;
ORACLE instance started.
.
.
Database mounted.


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

Database altered.
```

On **PRIMARY**, switch a log file for resynchronisation

```
SQL> alter system switch logfile;

System altered.
```

Demo of using Physical Standby in Temporary Read-Write mode

In Oracle 10G R2, flashback database beyond resetlogs capability enables usage of Physical Standby in Read Write mode for more effective application testing/processing instead of just query processing which is possible in Read Only mode.

On **STANDBY**, create a Restore Point

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

Database altered.

SQL> alter database open;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------
        56

SQL> create restore point before_read_write_standby guarantee flashback database;

Restore point created.
```

On **PRIMARY**, a logfile is switched and remote log shipping is disabled

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

  COUNT(*)
----------
        56

SQL> alter system switch logfile;

System altered.

SQL>  alter system set log_archive_dest_state_2=defer;

System altered.
```

On **STANDBY**, activate the Standby database and open for Read Write

```
SQL> alter database activate standby database;

Database altered.

SQL> startup mount force;
ORACLE instance started.
```

```
.
Database mounted.
SQL> alter database open;

Database altered.

SQL> select database_role, open_mode from v$database;

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

On this Activated database, do some DML

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

  COUNT(*)
----------
        56

SQL> insert into test select * from test;

112 rows created.

SQL> insert into test select * from test

224 rows created.

SQL> select count(*) from test;

  COUNT(*)
----------
       448
```

Now, we revert the temporarily activated database back to become Physical Standby database again by flashing it back to the defined Restore Point

```
SQL> startup mount force;
ORACLE instance started.
.
Database mounted.
SQL> flashback database to restore point before_read_write_standby;

Flashback complete.

SQL> alter database convert to physical standby;

Database altered.

SQL> startup mount force;
ORACLE instance started.
.
```

```
Database mounted.

SQL> select database_role, open_mode from v$database;

DATABASE_ROLE      OPEN_MODE
------------------------------  --------------------
PHYSICAL STANDBY   MOUNTED

SQL> alter database open;

Database altered.

SQL> select count(*) from test;

  COUNT(*)
----------
        56
```

On **PRIMARY**, enable log shipping to standby and switch logfile

```
SQL>  alter system set log_archive_dest_state_2=enable;

System altered.

SQL> alter system switch logfile;

System altered.
```

On **STANDBY**, Real Time Apply starts again from the same point

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

Database altered.
```

Few Observations

- So far, the best version to turn on the flashback feature in the database is Oracle Server Enterprise Edition - Version 10.2.0.3, as it has addressed few bug fixes around flashback
- Oracle Server versions 10.1.0.1 to 10.1.0.4 have got few bugs and issues with flashbacks e.g.

  Instance crashes when the Flashback Log is Inaccessible Ref: Metalink Note 311150.1

  Bug 4764435 ORA-600[2662] After Flashback, Recover and Flashback, bug fixed in Oracle version 10.2.0.2

  Bug 5106952 Flashback Log space not being reclaimed, bug fixed in Oracle version 10.2.0.3

  Bug 4874986 OERI[2142] while using flashback feature, bug fixed in Oracle version 10.2.0.3
- It has been observed that when a Batch Stream ran, it generated about 200MB worth of Archive Logs, it also generated similar amount of Flashback Log
- For a Batch Run of 15 minutes, It has taken about 6 minutes to Flashback the entire database to the PreBatch State
- By doing comparisons between database runs with & without Flashback, no major performance degradation has been observed in the Flashback enabled database for both OLTP & Batch operations. Of course, any performance impact for flashback logging in a very high volume transaction oriented system should be measured in a test/pre-production environment before implementation
- By doing comparisons between database runs with & without Real Time Apply, no major performance degradation has been observed for both OLTP & Batch processing in the Real Time Apply enabled Data Guard environment
- Guaranteed Restore Points with Database Flashback logging can cause significant space pressure in the flashback recovery area (to satisfy guarantee, flashback logs are not deleted even if there is space pressure) and hence close monitoring is required to check the space used in the flash recovery area

In conclusion, Oracle 10G Data Guard Real Time Apply & Flashback Database features are great advancements in Oracle RDBMS technology and by utilizing those features the High Availability requirements of the Databases and the Applications could very well be established.

Ref used: Oracle 10G R2 Documentations & Oracle Metalink Knowledge Base