Oracle 11g Data Guard in Action

Uwe Hesse
Principal Instructor
Oracle University Germany

In this edition of "OU Expert's Corner", we examine the 11g version of Oracle Data Guard. We begin with a brief explanation of Data Guard concepts, and continue with a step-by-step guided tour of Data Guard 11g configuration and deployment. In the process of doing so, we will highlight some of the brilliant new features of 11g Data Guard, especially *Active Data Guard* and *Snapshot Standby Databases*. It is assumed that readers are familiar with the concepts of Data Guard including Log Transport services, Role Management services, Managed Log Apply services and the Data Guard Broker management infrastructure.

Data Guard Concepts:

Data Guard is Oracle's disaster recovery solution, providing High Availability and data protection in case of site failure. Standby databases are deployed at one or more remote sites and are synchronised with the primary database, using redo shipped from the primary site for this purpose. Clients connected to the primary, may have their sessions failed over to the standby site should a disaster such as an earthquake, strike at the primary site, but client connectivity management is beyond the scope of this article.

The transmission of redo from primary site to standby sites is relatively inexpensive, allowing Data Guard to provide a reliable and popular solution for high availability and data protection without additional hardware costs other than the standby database server.

Before Oracle 9i, Redo was transmitted by the `ARCn` process on the primary database when a log switch occurred, but since Oracle 9i the `LGWR` process is used, allowing Redo to be shipped as soon as it is generated rather than waiting for a log switch. This provides greater data protection by reducing the possible data lost from all changes in a complete log, to no loss at all. Redo transmitted by the `LGWR` process on the Primary is written by the `RFS` process on the standby to Standby Redo Logs, and from 10g R2 may be applied immediately on the standby using *Real Time Apply*, thereby keeping the standby synchronised.

Protection Modes and Log Transport

Protection modes, determine the degree of possible data loss if the primary fails. *Maximum Performance* mode reduces the cost of the synchronisation by having the redo shipped asynchronously to the standby, but may involve loss of recently generated redo. *Maximum Availability* and *Maximum Protection* modes both require synchronous transmission via `LGWR` with confirmation from the standby site that the redo was received and written into standby. This guarantees a "Zero Data Loss" capability, where no committed transaction is lost during a complete primary site failure.

Maximum Protection **mode requires the confirmation from a standby site, that redo was received and written to standby log files before LGWR writes locally to the primary redo logs. Failure to receive this confirmation, using *Maximum Protection* will cause the primary database to terminate itself, thereby preventing divergence between the primary and standby. Best practice is to configure two standby databases when using this protection level to reduce the chance of the primary being unavailable.**

Maximum Availability **mode permits the primary database to continue normally when no standby databases are contactable. Redo continues to be generated on the primary and archived as normal. When contact is re-established with at least one standby, archived and online redo logs that have not yet been shipped, will be transmitted to the standby by one or more `ARCn` processes and applied on the standby, thereby resynchronising the standby with the primary. This enables *Zero Data Loss*, should only one component, such as the network connection to the standby, fail. But data loss is still possible due to unrecoverable divergence, if the primary fails permanently at some point, after losing contact with the standby, and transactions have been committed on the primary, during the interval between loss of contact and the primary database failure.**

Maximum Performance **mode permits redo transport using `ARCn` or `LGWR`. With ARCn whole archive logs are shipped at log switch on the primary but with LGWn, the redo is shipped asynchronously using network slave processes, thereby reducing the redo transport performance overhead on the primary. This reduces the degree to which the standby lags behind the primary in keeping up to date. From 10g use of ARCn for this protection mode is deprecated. If contact is lost with all standby sites, then a primary will continue normally as is the case with *Maximum Availability*.**

Standby Database Types

Regardless of the protection level, there are two types of standby database which differ in the way that redo shipped from the primary is consumed. In *Physical Standby* databases, *Redo Apply* uses a media recovery process to apply redo arriving from the primary. This is a pure physical technique that works regardless of the data types in the database and for all `DML` and `DDL` operations that occurred on the primary site. Prior to Oracle 11g, *Redo Apply* only worked with the standby database in the `MOUNT` state, preventing queries against the physical standby whilst media recovery was in progress. This has changed in Oracle 11g and will be discussed later in the article.

Logical Standby **databases were introduced in Oracle 9iR2 and use *SQL Apply* to maintain synchronisation with the primary. *SQL Apply* uses `LOGMINER` technology to reconstruct `DML` statements from the redo generated on the primary. These statements are replayed on the standby database whilst in the `OPEN` state. The advantage of logical standby databases is that they can be queried while applying `SQL`. The disadvantage is that not all data types or `DDL` commands on the primary are supported using *SQL Apply*.**

Step-by-step Implementation of 11g Data Guard

Now that we've discussed the basic concepts of Data Guard, let's go and implement a physical standby database and see some of the fascinating new features of the 11g version!

The following briefly describes the deployment of an 11g physical standby database on Oracle Enterprise Linux, which is quite simple and fast. It is assumed that you are an experienced DBA who is familiar with the administration of Oracle Net and Oracle databases familiar with such things as `SPFILEs`.

Preparing for Standby Creation

We will create a physical standby database on the same host where the primary resides as a proof of concept, but this should not be done in a live environment requiring a disaster recovery configuration. We will have a primary called "prima" and a physical standby database called "physt". The box below contains the listener.ora used for this configuration.

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = <name>)(PORT = 1521))
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = prima_DGMGRL)
      (ORACLE_HOME = /u01/app/oracle/product/11.1.0/db_1)
      (SID_NAME = prima)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = physt_DGMGRL)
      (ORACLE_HOME = /u01/app/oracle/product/11.1.0/db_1)
      (SID_NAME = physt)
    )
```

<u>Note</u>: The value of parameter `GLOBAL_DB_NAME`, which is required for the Data Guard Broker to connect to the database instances.

Our primary database "prima" is quite simple, having been created using as few initialisation parameters as possible. We have created a password file for it with the following command.

```
orapwd file=$ORACLE_HOME/dbs/orapwprima password=oracle force=y
ignorecase=y
```

Note: The `IGNORECASE` parameter: This may cause problems if omitted, because the Broker implicitly always connects using uppercase passwords.

Below is a copy of initprima.ora from which we created the spfile for the primary database.

```
compatible=11.1.0
db_block_size=8192
db_name='prima'
db_unique_name='prima'
instance_name='prima'
control_files='/home/oracle/prima/control01.ctl'
memory_target=300m
undo_management=auto
undo_tablespace=undotbs1
log_archive_config='dg_config=(prima,physt)'
db_unique_name=prima'
standby_file_management=auto
processes=100
db_recovery_file_dest='/home/oracle/flashback'
db_recovery_file_dest_ssize=5g
diagnostic_dest='/home/oracle/prima'
remote_login_passwordfile=exclusive
```

Note: The `LOG_ARCHIVE_CONFIG` parameter required by the Data Guard Broker.

The primary database must be in "archivelog mode" and must enable "force logging" before creating a standby database. We must then create a password file and an initialisation parameter file for the future standby database, prior to starting the standby instance in `NOMOUNT` state. The rest of the deployment may be done using `RMAN`. The standby password file can be created as follows:

```
orapwd file=$ORACLE_HOME/dbs/orapwphyst password=oracle force=y
ignorecase=y
```

The standby spfile was created from a pfile containing the following:

```
compatible=11.1.0
db_block_size=8192
db_name='prima'
db_unique_name='physt'
instance_name='physt'
control_files='/home/oracle/physt/control01.ctl'
memory_target=300m
undo_management=auto
undo_tablespace=undotbs1
db_file_name_convert='/home/oracle/prima','/home/oracle/physt'
log_file_name_convert='/home/oracle/prima','/home/oracle/physt'
log_archive_config='dg_config=(prima,physt)'
db_unique_name=physt'
standby_file_management=auto
processes=100
db_recovery_file_dest_size=5g
db_recovery_file_dest='/home/oracle/flashback'
diagnostic_dest='/home/oracle/physt'
remote_login_passwordfile=exclusive
```

Note: Parameters DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT are required here, because the standby database files may not reside in the same location as the primary database files do, when both databases are on the same host. In addition, the parameter DB_UNIQUE_NAME *must* differ from the value on the primary site.

Creating the Standby Database

Now we start the standby instance NOMOUNT, and let RMAN do the rest for us:

```
RMAN> connect target sys/oracle@prima
connected to target database: PRIMA (DBID=1904029631)
RMAN> connect auxiliary sys/oracle@physt
connected to auxiliary database: PRIMA (not mounted)
RMAN> duplicate target database for standby from active
database;
```

Note: The "FROM ACTIVE DATABASE" clause causes RMAN to create the standby database "on the fly", without the need of a previous backup.

Creating Standby Redo Logs

We are almost finished deploying the standby database, but we must create standby log files on both sites, so that after a possible switchover, the old primary may work properly as a standby for any protection mode. (If we would have created standby logfiles on the primary before the duplicate command, then RMAN would have created the standby logs on the standby for us automatically.) We can do this with SQL*Plus after RMAN is finished:

```
SYS@physt > alter database add standby logfile
'/home/oracle/physt/sblog_g1m1.rdo' size 100m;
SYS@physt > alter database add standby logfile
'/home/oracle/physt/sblog_g2m1.rdo' size 100m;
SYS@physt > alter database add standby logfile
'/home/oracle/physt/sblog_g3m1.rdo' size 100m;

SYS@prima > alter database add standby logfile
'/home/oracle/prima/sblog_g1m1.rdo' size 100m;
SYS@prima > alter database add standby logfile
'/home/oracle/prima/sblog_g1m1.rdo' size 100m;
SYS@prima > alter database add standby logfile
'/home/oracle/prima/sblog_g1m1.rdo' size 100m;
```

Note: Always create one additional standby log group than there are online log groups. Also, they must be of the same size than the online log groups.

Now we can start the Data Guard Monitor process (DMON) on both sites with the following commands:

```
SYS@prima > alter system set dg_broker_start=true;
SYS@physt > alter system set dg_broker_start=true;
```

Next, we are ready to use the Data Guard Broker, which can be administrated using Database Control in 11g, or using the `DGMGRL` utility:

```
DGMGRL> connect sys/oracle@prima
Connected.
create configuration mycon as primary database is prima connect
identifier is prima;
Configuration "mycon" created with primary database "prima"
DGMGRL> add database physt as connect identifier is physt
maintained as physical;
Database "physt" added
DGMGRL> enable configuration;
Enabled.
DGMGRL> show configuration;

Configuration
  Name:                mycon
  Enabled:             YES
  Protection Mode:     MaxPerformance
  Databases:
    prima - Primary database
    physt - Physical standby database

Fast-Start Failover: DISABLED

Current status for "mycon":
SUCCESS
```

We now have a fully functional physical standby database, with a protection mode of *Maximum Performance* for the configuration. If you examine the parameter `LOG_ARCHIVE_DEST_2` on the primary site, you will see that `DMON` has set it to the appropriate value, so that redo can be transmitted to the standby site. The use of `RMAN` made this all fairly easy to do.

Changing Protection Modes

To increase the protection level to *Maximum Availability* do the following:

```
DGMGRL> edit database physt set property logxptmode=sync;
Property "logxptmode" updated
DGMGRL>  edit database prima set property logxptmode=sync;
Property "logxptmode" updated
DGMGRL> edit configuration set protection mode as maxavailability;
Succeeded.
DGMGRL> show configuration;

Configuration
  Name:                mycon
  Enabled:             YES
  Protection Mode:     MaxAvailability
  Databases:
    prima - Primary database
    physt - Physical standby database

Fast-Start Failover: DISABLED

Current status for "mycon":
SUCCESS
```

<u>Note</u>: Our configuration is using *Real Time Apply*, which causes the standby recovery process to read directly from the standby logs, without waiting for them to be archived. This means that the redo is applied fairly soon after being received, reducing the "Apply Lag" to a very low value.

<u>11g Active Data Guard</u>

***Active Data Guard* is a brilliant new 11g feature, which allows a physical standby database to be opened "Read Only" as in past releases, but which then allows the media recovery process to be restarted. This allows `SQL` queries to be done whilst the standby database is applying redo, shipped from the primary! To implement this, do the following:**

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

SYS@physt > alter database open read only;
Database altered.

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

DML performed on your primary database is visible almost instantly on the physical standby due to the use of *Real Time Apply* discussed earlier. Consequently, you can offload query only workloads from your primary database to your physical standby, thereby reducing the load on your primary database, while maintaining your disaster recovery capability!

<u>11g Snapshot Standby Databases</u>

Another very useful feature of 11g is the ability to convert your standby database into a test database, while the redo from the primary site *is still received*. After your testing is done, you can convert your test database back to a standby database. Although data protection and a disaster recovery capability are maintained when using a "Snapshot Standby", a failover done while it exists, will most likely be slower, because it must be converted back to a "Physical Standby" and then all redo received from the primary whilst using the "Snapshot Standby", must be applied before the failover is completed. If you wish to use this feature, then perhaps you should deploy two standby databases, one for DR and one for use as a "Snapshot Standby" as required.

Here is how it is done:

```
DGMGRL> startup force mount

DGMGRL > convert database physt to snapshot standby;
```

```
DGMGRL> show configuration

Configuration
  Name:                mycon
  Enabled:             YES
  Protection Mode:     MaxAvailability
  Databases:
    prima - Primary database
    physt - Snapshot standby database

Fast-Start Failover: DISABLED

Current status for "mycon":
SUCCESS
```

You have now converted your standby database into a test database. This feature is extremely useful when combined with the *Real Application Testing* features provided in the 11g which are beyond the scope of this article.

Note: Under the covers, the convert command enables flashback database if not already enabled. When converting back to a physical standby, the snapshot standby is flashed back to the point in time when the physical standby was originally converted. When testing is completed, you convert the snapshot standby database back to a physcial standby database.

Converting back to a standby is done as follows:

```
DGMGRL> convert database physt to physical standby;
Converting database "physt" to a Physical Standby database,
please wait...
Operation requires shutdown of instance "physt" on database
"physt"
Shutting down instance "physt"...
[…]
Operation requires startup of instance "physt" on database
"physt"
[…]
Continuing to convert database "physt" ...
Operation requires shutdown of instance "physt" on database
"physt"
[…]
Operation requires startup of instance "physt" on database
"physt"
Starting instance "physt"...
ORACLE instance started.
Database mounted.
Database "physt" converted successfully
```

You may now decide to turn on *Active Data Guard* again as described above.

We hope that you enjoyed our little tour through the implementation of an 11g Data Guard Configuration, and that you liked the new features, the ease of creation, and the ease of operation as well! There is much more to Data Guard, but this should get you started on the most common configuration used.