Quick Guide to configuring

Oracle 11gR2 Data Guard

Physical Standby

Part II

Author: Jed Walker is the Manager of Database Operations for the Comcast Media Center in Centennial Colorado. He has been working with Oracle Database since 1997 and is an Oracle Certified Professional for 9i, 10g, and 11g Database.

Table of Contents

Introduction

Data Guard is an Oracle feature that primarily provides database redundancy. This is done by having a copy or standby database, preferably in another location and on separate disk. This standby database is maintained by applying the changes from the primary database to it. Standby databases can be maintained with either Redo (Physical standby) or SQL (Logical standby).

My intention with this paper is primarily to show that configuring Data Guard is not complex, and does not require any special skills or training to accomplish. Part I of this paper walked you through creating a basic Data Guard configuration. In this part I will walk you through switchover, failover (database and client), and other items. My hope is that if you're new to Data Guard, just considering it, or worried that it is too hard to setup, that this paper will help you through the process and get you up and running.

If you find mistakes, additional important notes or considerations, please let me know.

Previously

In Part I of this paper, we got a physical standby Data Guard configuration running. In this part I will cover switchovers, failovers, client failover, rebuilding using Flashback database, Active Data Guard, and a small discussion on backups.

Configuration for failover

Now that you have a Physical Standby database you might think you're ready to try doing a switchover, or even a failover, but first you want to make sure your clients will follow along. For that, we need to configure the databases and the clients to support this. To ensure your clients are able to find the correct database, you must configure a fail over supporting service on your database, and configure the client's TNS to know where the databases in the Data Guard cluster are and how to find the primary.

First, we'll create the Failover supporting service on the database. This involves creating the service, making sure it is started on the primary database, and making sure it is always running on only the primary database. The way I have done this is to create an additional service name for the clients to connect to. This allows you to connect to any specific database you want to, but also connect to the primary when that is what you want. This is how we create the service with SQL:

```
begin

    DBMS_SERVICE.CREATE_SERVICE (service_name => 'JED_RW',

                                 network_name => 'JED_RW',

                                 aq_ha_notifications => TRUE,

                                 failover_method => 'BASIC',

                                 failover_type => 'SELECT',

                                 failover_retries => 30,

                                 failover_delay => 5);
```

```
3
/
```

This service is setup to send fail over notifications to clients, and allow for SELECT queries to continue if a failover occurs. I use the naming convention of SID_RW to indicate that this is a Read/Write database (the primary). The reason I do this is for clarity of purpose when using Oracle Active Data Guard. I'll talk more about that later.

Next we need to make sure this service is always running on the Primary, but not on the Standby. To do that we create a procedure that starts the service if the database is primary and stops the service if it is standby.

```
create or replace procedure cmc_taf_service_proc

is

 v_role VARCHAR(30);

begin

  select DATABASE_ROLE into v_role from V$DATABASE;

  if v_role = 'PRIMARY' then

    DBMS_SERVICE.START_SERVICE('JED_RW');

  else

    DBMS_SERVICE.STOP_SERVICE('JED_RW');

  end if;

end;

/
```

Next, we create two triggers to run the procedure on database startup and on role change. Documentation I've found only mentions creating a trigger for role change in 11g, but if you bounce your database it won't restart the fail over service. This is why I create both.

```
create or replace TRIGGER cmc_taf_service_trg_startup

after startup on database

begin

  cmc_taf_service_proc;

end;

/

create or replace TRIGGER cmc_taf_manage_trg_rolechange

after db_role_change on database

begin
```

```
4
  cmc_taf_service_proc;

end;

/
```

Now that we have that in place, execute the procedure to make sure the service is running and archive the current log so the changes make their way to the standby.

```
SQL> exec cmc_taf_service_proc;

SQL> alter system archive log current;
```

We now have a service name of JED_RW that clients can use to connect to.

```
SQL> show parameter service_names

NAME             TYPE        VALUE

---------------- ----------- ------------

service_names    string      JED_RW
```

Having this service name available isn't enough though. You must also configure your client's TNS Names entry (or other connection method) to support this. The client TNS Names entry should appear as follows:

```
JED_RW =

   (DESCRIPTION =

    (ADDRESS_LIST=

     (ADDRESS = (PROTOCOL = TCP)(HOST = dev-db1.cmc.cable.comcast.com)(PORT = 1521))

     (ADDRESS = (PROTOCOL = TCP)(HOST = dev-db2.cmc.cable.comcast.com)(PORT = 1521))

    )

    (CONNECT_DATA = (SERVICE_NAME = JED_RW)

                    (FAILOVER_MODE=(TYPE=SELECT)(METHOD=BASIC)(RETRIES=30)(DELAY=5))

    )

   )
```

Note that it includes both of the host servers in the Data Guard configuration and uses the JED_RW service name to ensure that it connects to the primary database.

If you want to connect to a particular database, regardless of whether it is primary or standby, you can just use the standard entry:

```
JED =

  (DESCRIPTION_LIST=

   (DESCRIPTION =
```

```
    (ADDRESS = (PROTOCOL = TCP)(HOST = dev-db1.cmc.cable.comcast.com)(PORT = 1521))

    (CONNECT_DATA = (SERVICE_NAME = JED))

   )

  )
```

Once your client connects using this new net service name, they will be able to follow along when a switchover or failover occurs. If the client is running a query and has no DML in a transaction then their work will continue normally, with a delay, as long as the switchover or failover completes prior to the last retry (you should experiment with switchover and failover so your RETRIES and DELAY settings will be appropriate). If they have a transaction in progress they will be connected to the new primary, but they will get an error (e.g. ORA-25402: transaction must roll back), and will have to issue a rollback.

If your database is up and running, but you get "ORA-01033: ORACLE initialization or shutdown in progress" then it is likely you are trying to connect to the standby database. Check your TNS net service name entry carefully to make sure you have it configured for the "RW" service name.

Performing a Switchover

If you've made it to this point then things are looking good. Now you're ready to do a switchover, but there are a few checks you should perform before you start. First verify there are no redo gaps, by querying the primary:

```
SQL> select STATUS, GAP_STATUS from V$ARCHIVE_DEST_STATUS where DEST_ID = 2;
```

You should get VALID and NO GAP to proceed.

Verify that temporary files on the standby match those on the primary by querying v$datafile and checking at the file system.

Remove any settings you added to your LOG_ARCHIVE_DEST_N parameters to delay apply of redo logs on the standby – you want to have everything applied for a good switchover (no data loss). Then, verify that all available redo has been applied to the standby, by querying the standby:

```
SQL> select NAME, VALUE, DATUM_TIME from V$DATAGUARD_STATS;
```

There should be no "transport lag" or "apply lag", and "finish time" should be zero.

Having checked those pre-requisites, verify that the primary is ready to be transitioned to a standby, by querying the primary:

```
SQL> select SWITCHOVER_STATUS from V$DATABASE;
```

If you see TO STANDBY or SESSIONS ACTIVE then it is ready. So, switch the primary to standby role:

```
SQL> alter database commit to switchover to physical standby with session
shutdown;

SQL> shutdown immediate;
```

```
SQL> startup mount;
```

Next, you need to verify the standby can be switched to primary role, by querying the standby:

```
SQL> select SWITCHOVER_STATUS from V$DATABASE;
```

If you see TO PRIMARY or SESSIONS ACTIVE then you are ready. If you see SWITCHOVER LATENT or SWITCHOVER PENDING then check the alert log for any issues. Usually it just needs to apply some redo. If that is the case, run the following on the standby:

```
SQL> recover standby database using backup controlfile;
```

You should see SWITCHOVER PENDING from the previous query while it applies. When complete you should see TO PRIMARY or SESSIONS ACTIVE.

You can now switch the standby to primary role:

```
SQL> alter database commit to switchover to primary with session shutdown;

SQL> alter database open;
```

Now that you have completed the switchover, be sure to start Redo Apply on the standby:

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

If your clients are configured correctly they should failover to the new database. I'll cover this later.

Performing a Failover

Failover is used when the Primary database is either lost or not able to function correctly and service must be restored. A failover converts a standby database into a primary, but unlike a switchover does not change the role of the primary (it is not functioning). When a failover is done you must recreate the Primary, or using Flashback Database, roll it back to a point-in-time prior to the event, and then convert it to a standby and start redo apply.

I have classified failover into three categories: Graceful, Almost Graceful, and Standard. The category that is used is dependent on how bad the failure of the primary is, and no, a failover isn't graceful but how you do it can be. Start with Graceful and work your way to Standard to ensure the best failover (in regards to how much data you will lose, or not lose). Note that these are not Oracle terms for failover, I made them up to represent the three phases I see if performing a failover.

Fail Over - Graceful

If the standby is in maximum protection mode it must be switched to maximum performance mode in order for you to do a failover. To change this:

```
SQL> alter database set standby database to maximize performance;
```

Now, if you can mount the primary database, you can try flushing any unsent redo to the standby. If you can do this, it is possible you can failover without any data loss. Note that in this example we recently did a switchover to JED2, so we are now failing over to JED.

Try this:

```
SQL> startup mount

SQL> alter system flush redo to 'JED';
```

If this works and the redo is transported you will still need to apply it all. Now continue to “Fail Over – Almost Graceful” to make sure all redo is available at the standby.

Fail Over - Almost Graceful

To avoid losing as much data as possible you should try to get all archived redo logs applied. To do this you should copy over any archived redo logs from the primary to the standby. It is possible some of these may already be at the standby, but if you do this you can be sure you have as much redo as possible. Then you will need to resolve any gaps in the redo on the standby.

First, copy all archived redo log files to the standby and then register them with the database:

```
SQL> alter database register physical logfile '&logfile_path_name';
```

Then check for any redo gaps:

```
SQL> select THREAD#, LOW_SEQUENCE#, HIGH_SEQUENCE# from V$ARCHIVE_GAP;
```

If any gaps exist and the files are available on the failed primary then copy those logfiles across from the appropriate thread and register them as previously shown.

Fail Over - Standard

Stop managed redo apply by issuing this on the standby:

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

Finish applying any redo:

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

If you receive any errors on this you’ll need to examine the trace/alert information. Once you’ve run this command you cannot return this database to a standby, it must become a primary or be recreated.

Now verify the standby can be switched to primary, by running the following on the standby:

```
SQL> select SWITCHOVER_STATUS from V$DATABASE;
```

You should get TO PRIMARY or SESSIONS ACTIVE from this query. If so you can continue, if not, you probably have not finished applying all redo. Be sure you ran the RECOVER … FINISH command.

You can now failover your standby to primary:

```
SQL> alter database commit to switchover to primary with session shutdown;

SQL> alter database open;
```

If you have other standby locations you might need to restart Redo Apply at those locations. Your next step is to rebuild the old primary as a standby. If you have Flashback Database configured, then you can use that to make this task easier.

Rebuild using Flashback Database

Now that you've had to failover to your standby, you need to create a new standby from your old primary. To rebuild your old primary as a new standby using Flashback Database you first need to get the SCN at which the old standby became the new primary. To find this you can query the new primary:

```
SQL> SELECT to_char(STANDBY_BECAME_PRIMARY_SCN) from V$DATABASE;
```

With the SCN in hand, you can now use Flashback Database to return your old primary to the point-in-time the switchover occurred. On the old primary:

```
SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP MOUNT;

SQL> FLASHBACK DATABASE TO SCN &standby_became_primary_scn;
```

If you did not have Flashback Logging turned on then the command will fail with "ORA-38726: Flashback database logging is not on." and you will not be able to flashback. Instead you will need to recreate the database as a standby from the new primary.

Now that the old primary is at an SCN that can be used to recover from the new primary's redo, you can convert it to a physical standby database and start the Redo Apply process.

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;

SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP MOUNT;

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

Now perform a log switch and verify Redo Apply is working using the previous instructions.

Client failover

Something I really love is that the clients can automatically reconnect when you do a switchover or failover. Try the following example on your system and watch the results. For this example, I have JED as my primary database and JED2 as my standby database. The client has connected using the fail over capable JED_RW net service name.

First, connect a SQL*Plus client to the primary database using JED_RW as the SYSTEM user and see where you are connected:

```
CLIENT> connect system@jed_rw
```

```
CLIENT> select db_unique_name from v$database;
```

You should see JED. Now perform the switchover steps up to the point where you commit the standby to being a primary with the "`alter database commit to switchover to primary with session shutdown;`" command. At this point both your standby and primary are in the MOUNT state. Try your query again:

```
CLIENT> select db_unique_name from v$database;
```

The result should be what appears to be a hang. This is because your client is currently making attempts to find the primary database, but there currently isn't a primary database available. Now, go ahead and complete the switchover process.

When the process is complete your client should reconnect and re-issue the query. When it does your query should complete and return JED2 because the primary is now the JED2 database, not the JED database. Another cool way to watch this happen is to start a very long running query and start the switchover just after the results start scrolling in your session. You should notice the results pause and then restart when the switchover or failover is complete.

Active Data Guard

Warning! Active Data Guard is a separately licensed feature. So while it is easy to turn on you should not use this without licensing.

Active Data Guard is a new feature in Oracle 11g that allows you to have a physical standby open for read operations while at the same time actively applying redo. It is quickly apparent why this is such a great feature. The ability to have a physical (not logical) copy of your primary database that you can backup and read from while keeping it current is a very nice advantage. Oracle knows this too, and so it is separately licensed.

The great thing about Active Data Guard is how easy it is to activate. It is simply a matter of opening your standby database and then starting Redo Apply.

```
SQL> STARTUP MOUNT

SQL> ALTER DATABASE OPEN;

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

You can now login and run select queries from the database. You can also verify this with the following query:

```
SQL> SELECT name status, database_role, open_mode logins, log_mode FROM
v$instance, v$database;
```

In Active Data Guard mode you should see PHYSICAL STANDBY, READ ONLY WITH APPLY, and ARCHIVELOG.

Another change you'll want to make if you run Active Data Guard is allowing users to connect to the "correct" database for their needs, wherever it is. For this, I create a second service called SID_RO and run it only on the

Standby database. This indicates that you are connecting to the Read Only standby database, not the Read Write primary. This is done the same as the SID_RW service was:

```
begin
  DBMS_SERVICE.CREATE_SERVICE (service_name => 'JED_RO',
                               network_name => 'JED_RO',
                               aq_ha_notifications => TRUE,
                               failover_method => 'BASIC',
                               failover_type => 'SELECT',
                               failover_retries => 30,
                               failover_delay => 5);
end;
/
```

You then need to modify your procedure for starting/stopping the services, so that it starts the correct service:

```
create or replace procedure cmc_taf_service_proc
is
 v_role VARCHAR(30);
begin
  select DATABASE_ROLE into v_role from V$DATABASE;
  if v_role = 'PRIMARY' then
    begin
      DBMS_SERVICE.STOP_SERVICE('JED_RO');
    exception
     when others then null;
    end;
    DBMS_SERVICE.START_SERVICE('JED_RW');
  else
    begin
      DBMS_SERVICE.STOP_SERVICE('JED_RW');
    exception
```

```
11
     when others then null;

    end;

    DBMS_SERVICE.START_SERVICE('JED_RO');

  end if;

end;

/
```

You now have an Active Data Guard configuration, and your clients can connect to the appropriate instance with the ability to re-connect upon a switchover or failover.

Backup

Finally, a discussion of Data Guard wouldn't be complete without some discussion of backups. Data Guard is essentially a backup, but that does not mean you can go without your RMAN backups. You've taken the time to force all redo to be logged, so you might as well do some backups too!

With Data Guard and RMAN you can perform your backups on the primary or standby database, but since you have a physical standby you might as well take that load off of your primary. In general, the standard backup commands (and scripts) you use on the primary will work on the standby, but there are some catches that you should be aware of. These are documented in the Oracle documentation and I will just mention a few key things:

- You should use a Recovery Catalog. This is because the primary will need to be aware of what backup files are available on the standby. Also, you do not need to register the standby with the catalog, the catalog will recognize it as a standby.
- You cannot backup the standby control file, so don't turn off backups completely on the primary. At the least backup your control file and spfile.

With that said, backup and recovery could be a whole paper, so I'm just going to review how I setup my backups to get you started and then you can use that, or modify it, or come up with your own strategy. Please be sure you test that you can recover from whatever you implement. Before running your backups you should configure a few basic items.

Be sure you are doing backups of the control file and spfile:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

Set your retention policy as needed:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 3 DAYS;
```

Avoid backing up a file if it has been backed up already and has the same checkpoint SCN:

```
CONFIGURE BACKUP OPTIMIZATION ON;
```

Only remove archived logs from the Primary if they have been applied to the standby (or shipped if you like):

```
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
```

Allow RMAN to re-synchronize between Primary and Standby:

```
CONFIGURE DB_UNIQUE_NAME P10AC CONNECT IDENTIFIER ‘JED’;

CONFIGURE DB_UNIQUE_NAME P11AC CONNECT IDENTIFIER 'JED2';
```

On the primary I still backup the archived logs. First, by having archived logs on the primary and standby it gives me two locations where archived logs are stored (redundancy). Second, in the event I had to do some recovery (offline data file, etc) I have logs available on the primary already. The catch of course is that I need to remove them at some point to free up disk space. So, running an archivelog backup on the primary is required. The standard commands for removing archivelogs on a stand-alone don't work on a Data Guard setup, so you'll notice this is a bit different. Given that we have to use a recovery catalog, I create a global script for this:

```
dg_primary_arch

{

 backup archivelog all;

 delete noprompt archivelog all completed before 'sysdate-.5';

 delete noprompt backup of archivelog all completed before 'sysdate-2';

}
```

On the standby I run my standard backup of the database and archivelogs and remove the old obsolete backups. One catch with Data Guard is that if you include the archive logs in the backup command you'll often get “RMAN-08137: WARNING: archived log not deleted, needed for standby or upstream capture process”. To avoid this you can back them up in a separate command. This way you still have your database and archivelog backup without the pesky error. Again, I use a global script:

```
create global script dg_standby_full

{

 backup database plus archivelog;

 delete noprompt archivelog all completed before 'sysdate-1';

 delete noprompt obsolete;

}
```

Another useful technique is to use a shared filesystem for your backups if possible. That way you have visibility to the backups from both servers. In the event you need to do a recovery on your primary you do not have to copy over the backup files first. Note that in this case you will have two copies, but on the same file system, so a disk failure could cause the loss of both copies.

Conclusion

Oracle 11g Data Guard is a very nice feature that can be configured relatively easily and that provides the ability for the system to failover in the event of an outage on the primary database. It also offers the ability to offload backups to your standby thereby reducing the load on your primary database. In addition Oracle Data Guard Broker is a tool that claims to make all of this much simpler and easier to manage, but that is something for another paper.