

Integrating Streams Replication and Standby Databases.

*John Garmany, Senior Consulting
garmanyj@proxitec.com*

The use of Oracle Streams Replication has become more and more common, replacing the more cumbersome Multimaster Advanced Replication. Many shops use Stream Replication to off load reporting and high impact loads from production databases. More than half of the clients I support have the destination database in the same rack as the source database. With both databases located in close proximity, many shops are also implementing physical standbys of one or both of the replicated databases in a remote location. Oracle Streams and Dataguard Physical Standby database integrate easily, once you understand how the two technologies fit together.

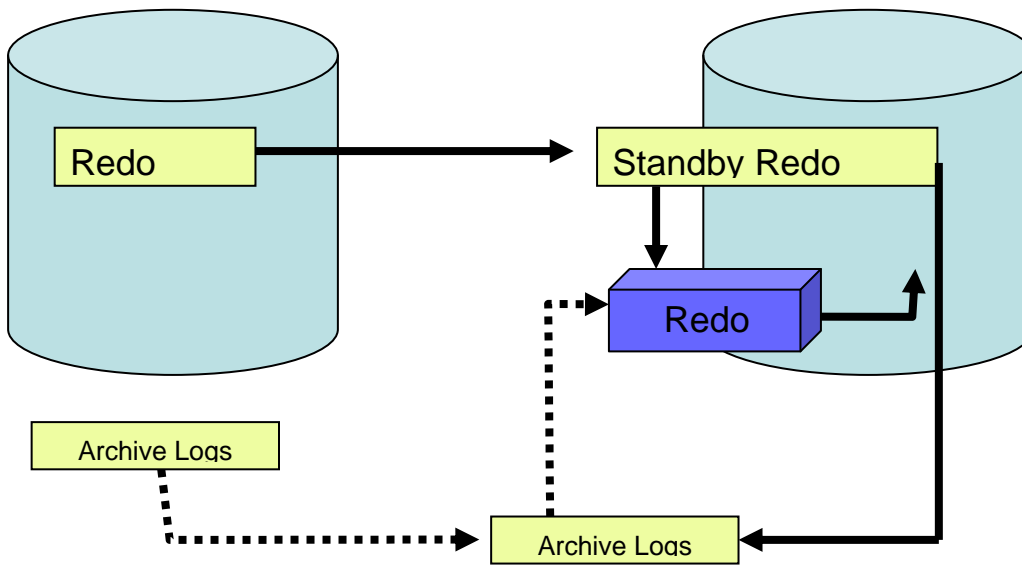
Why Have A Duplicate Database.

The Managed Physical Standby is the premier solution for fast failover to a local or remote database. The production and standby databases are exact matches to each other. Datafiles from the standby can be used to recover the production database. RMAN can be backup the standby and use the backup to recover the production database. This is a very powerful capability. A managed standby database is also the most efficient method of creating a standby from a network load point of view. The only data the production database sends to the standby are the changes. The one drawback is that the database is normally only usable for short periods of time and changes are not applied while the standby is open. Oracle 11g does allow the standby to be open and applying changes.

A Replicated database can be used to allow multiple locations to share one set of data. Each location has its own copy of the database and each database sends changes to all the other databases in the scheme. More often, the replicated database is used to offload heavy work from the primary database. These replication schemes are normal one direction with the source database being an OLTP database and the destination configured for some type of reporting. A replicated database can also be used for availability. If there are two databases implementing bidirectional replication and one becomes unavailable; the other can still be used. Likewise, if the network between the two databases does down, both systems will continue to function and changes will be propagated when the network is again functioning.

Functioning of the Oracle Managed Standby.

A detailed explanation of Oracle's Managed Standby Database is beyond the scope of this paper. This is just a brief explanation highlighting important details. A Managed Standby Database is a copy of the source database. It is maintained in a mounted mode performing automatic recovery of changes sent by the source/primary database. There is some delay in applying changes but in practice the properly configured standby will stay very close to the source database. It is called a managed standby because changes are automatically sent to and registered with the standby. The standby will insure changes arrive and are applied and if a change gets lost, it will request that change from the source database automatically. Changes on the source/primary database result in changes on the standby. New datafiles, users, tables, grants, database links all get recovered into the standby database. One item that does not get sent to the database is flashback database. If you flashback the source/primary database, you must flashback the standby.



Switchover

The real beauty of the Managed Standby Database is that it is always ready to take over the job from the source database. A switchover is where the source and standby databases switch roles. The source becomes a new standby and the standby becomes the new primary database. The switchover is a no data loss operation. The two databases coordinate so that the last change accepted by the source is applied to the standby during the role transition. Everything in the source is now in the activated standby.

Failover

A Failover is performed when the source database is lost. The standby applies all the changes it can get, and then transitions to the primary role. This operation can result in loss of data, but if the standby is properly configured it should be very little. The new primary (the old standby) assumes it is consistent and takes on the primary role.

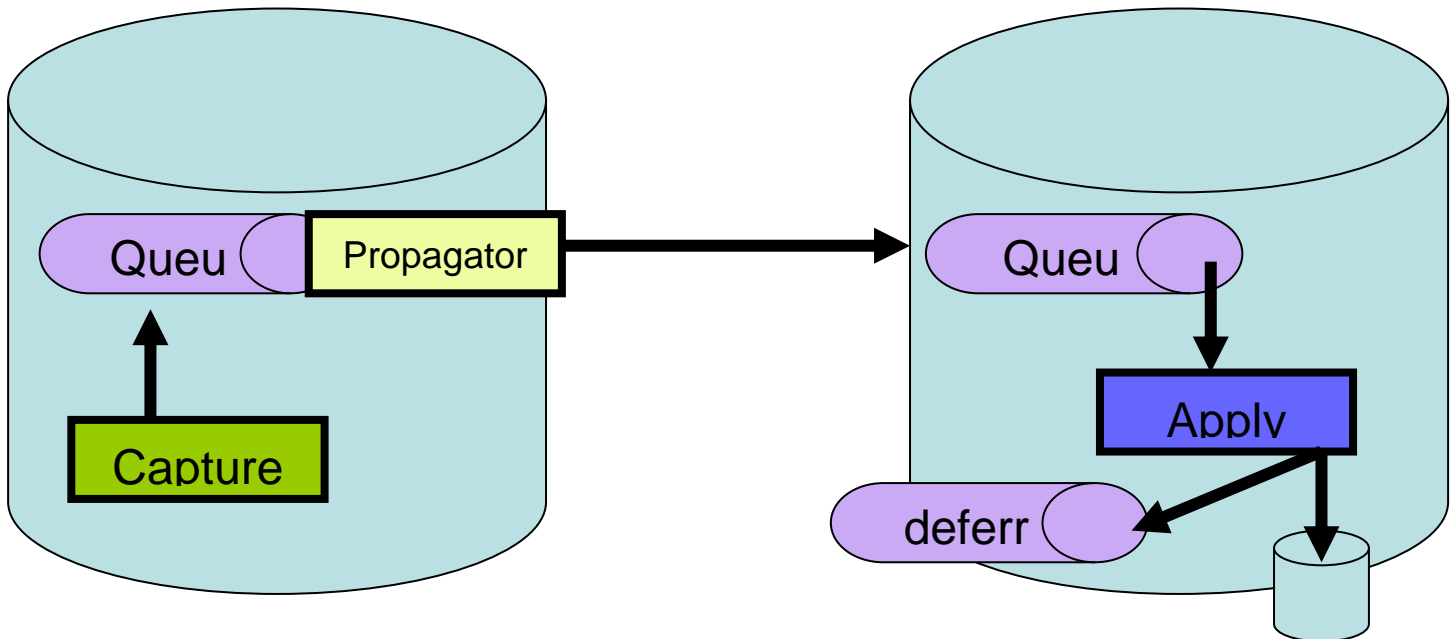
An important detail to note is that everything in the source/primary database is contained in the standby, to include all the replication components discussed in the next section.

Functioning of Oracle Streams Replication.

Oracle Streams Replication can be one direction, bi-directional or master to master. It can be mixed with other replication methods like snapshot or materialized view replication. Replication differs from a standby in that all databases are open and can be individually changes. The databases exchange changes by capturing them on the source and applying them on the destination. If the source database is under a heavy load, replication will fall behind but will catch backup once the load is reduced.

Stream replication has three major components, Capture, Propagator, Apply. Each of the components uses a set of defined Rules to determine if the change should be acted on. The Capture process runs on the source database and uses Log Miner to mine the redo, capturing changes according to its rules. Captured changes are placed in queue for forwarding to the destination database. The Propagator run on the source database and its function is to take the changes from the queue on the source database and if they match its Rules, place them in a queue on the destination database. The Apply process runs on the

destination database and will take changes from the queue and if they match its Rules, apply them to the destination database. The propagator uses a database link to connect to the destination database.

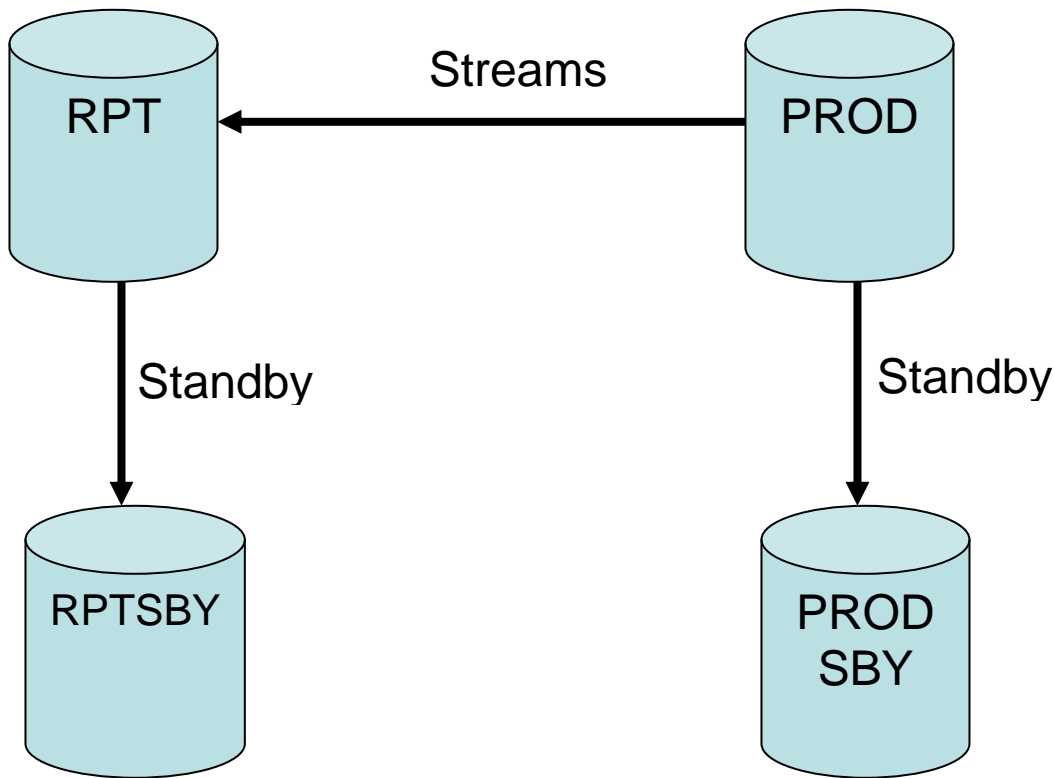


If the Apply process gets an error when it applies a change, it is placed in the deferror queue and waits for the DBA to fix the problem. The Rules are important as that is how Streams determines what to replicate. Whereas the Managed Standby is a complete copy of the source database, Streams may replicate all or part of the source database. It can replicate one or more tables, one or more schemas, or the entire database. Streams replication captures changes to tables and DDL. So if you are replicating a schema and create an index on a table in that schema, the create index replicates to the destination database. However, when you update a row in that table (and thus in that index), streams will only replicate the change in the table, causing the destination database to update the index. As with the Managed Standby, there is much more to Streams Replication but it is beyond the scope of this paper.

The Capture process has two important parameters, the `first_scn` and the `start_scn`. The `first_scn` is where the capture starts reading REDO. The `start_scn` is where the process starts capturing changes. The apply process has the `oldest_message_number` parameter, The `oldest_message_number` is the source scn for the last change applied.

The Setup

For discussion in the paper and the presentation we will use a common setup where one production database PROD streams a schema (or two) to a reporting database called RPT. Both PROD and RPT have managed standby database respectively called PRODSBY and RPTSBY.



This presentation will address three switchover scenarios.

1. PROD switchover to PRODSBY
2. RPT switchover to RPTSBY
3. Both switchover.

The switchover scenarios are actually easy. All these scenarios assume that databases are using the same tnsnames.ora file.

PROD switchover to PRODSBY

This scenario is trivial. Simply stop the Capture process, and then stop the Propagator. Perform the switchover. When PRODSBY becomes active as the primary, it already has all the replication components in the same state as PROD, including the database link that is pointing to RPT. Now restart the Propagator and Capture process on the new primary PRODSBY and changes are sent to RPT. In fact, RPT does not even know that its source database has changes.

RPT switchover to RPTSBY

This scenario starts with PROD replicating to RPT (the original setup). RPT has a problem and we are switching over to RPTSBY. Like the scenario above, stop the Capture and Propagation on PROD to stop the replication. Stop the Apply process on RPT. Now perform a normal switchover to RPTSBY. When RPTSBY becomes active, it contains the apply process from RPT. Now we need to edit the tnsnames.ora entry that is used by the database link on PROD so that it points to the RPTSBY server. Once that is completed, restart the APPLY on RPTSBY, restart the Capture and Propagator on PROD.

Switchover of Both Databases

This scenario is not harder than the two above. Stop the Capture and Propagator on PROD, then stop the Apply on RPT. Now perform a normal switchover of both databases. Edit the tnsnames.ora entry on PRODSBY to point to RPTSBY. Now restart the replication by starting the Capture and Propagator on PRODSBY and the Apply on RPTSBY.

Now that we breezed through the easy part, let's jump into the hard stuff, failover.

Failover Scenarios

Failover assumes some data loss. The primary database is lost. There are again three scenarios.

1. Failover of PROD leaves PRODSBY ahead of RPT.
2. Failover of PROD leaves PRODSBY behind RPT.
3. Failover of RPT leaves RPTSBY behind PROD.

So the difference in these scenarios is the lost data and its impact on the replication. If the dataloss was in the transition to the standby, the destination database could contain changes that were lost on the new source database. The first step is to determine if the source database (PROD, PRODSBY) is ahead or behind the destination (RPT, RPTSBY). To determine where the source database started as the primary you look in v\$database.

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

Now determine what has been applied on the destination database.

```
SQL> select hwm_message_number
       from v$streams_apply_coordinator
       where apply_name = 'APPLY_GLV';
```

```
SQL> select oldest_message_number from dba_apply_progress;
```

Which query you use is dependent on whether the Apply process is running or not.

If the Apply reports a higher SCN than the new primary, there are changes on the destination that are not on the source.

Failover of PROD leaves PRODSBY ahead of RPT.

In this scenario, the failover has left the new source ahead of the destination. All that is needed is to have the Capture process recapture the missing changes. Below are the steps.

- a. Execute the Failover.
- b. Determine if PRODSBY is ahead of RPT - It is ahead.
- c. Stop the Capture and drop the Propagator.

```

Begin
  Dbms_propagation_admin.drop_propagation ('PROP_PROD');
  Dbms_capture_admin.stop_capture('CAPTURE_PROD');
End;
/

```

d. Determine the last change applied on the destination

```
SQL> select oldest_message_number from dba_apply_progress;
```

e. Reset the Capture start_scn.

```

Begin
  Dbms_capture_admin.alter_capture(
    Capture_name=>'CAPTURE_PROD',
    Start_scn => <number for apply>);
End;
/

```

f. Recreate the Propagator.

```

Begin
  Dbms_propagation_admin.create_propagation(
    Propagation_name => 'PROP_PROD',
    Source_queue      => 'REP_CAPTURE_QUEUE',
    Destination_queue => 'REP_DEST_QUEUE',
    Destination_dblink => 'RPT.ORACLE.LOCAL');
End;
/

```

g. Restart the replication.

This scenario actually is the same as scenario 3. Both are based on the source database being ahead of the destination database. This leaves us with one final scenario, the hard one.

Failover of PROD leaves PRODSBY behind RPT.

This scenario is the most problematic to resolve and happily the least likely to occur. If the source database is behind the destination database, the destination database has applied changes that were lost on the source database. To resolve this dilemma, the DBA has three options. He can reenter the changes on the source database, he can write off the changes and recreate the replication so that the destination matches the source, or he can write off the changes and flashback the destination to the point where scenario 1 can be used to resync the replication.

It is rare that a DBA can determine what changes are missing and reapply them to the source database. In fact, I have never seen a case where changes lost during a failover could be identified.

The second option is for the DBA to write off the changes and recreate the replication. This will result in both being the source and destination being in synch but can be time and resource consuming. However this method guarantees that the two databases are in synch.

The third option is to write off the missing changes and use flashback database to remove the changes from the destination database. This option is the preferred option if it can be used. Using flashback is also a bit more complicated than it first appears.

- Remember that the source and destination databases are independent; there is no correlation between their respective scns except time.
- Flashback database must be on before the failover occurred.

The DBA must flashback the destination database in small steps until the Apply reports that it is behind the source database. Once the destination is behind the source, the previous method can be used to reset the Capture start_scn and resynch the replication.

Conclusion

Streams and Standby technologies work well together providing both data replication and fast recovery. Switchover operations are simple and easy to coordinate. Failover operations require additional preparation and planning. The last point I will make is that all of this becomes difficult under high stress. You need a test environment where you can practice each scenario.