The Oracle Magician

Spring, 2005 *OracleMagician.com* *Volume IV, number 2*

From the Editor

Welcome to the new issue of our online magazine, The *Oracle Magician*! This quarterly newsletter focuses on various "tricks of the trade" in the Oracle world--from DBAs, architects, developers, designers, and report writers.

Thank you to the many notes and comments from readers of my book, *The Art & Science of Oracle Performance Tuning*. Your notes are most appreciated!

In this issue, Brian Keating, of Database Specialists, presents the proper way to perform a graceful failover and recovery. Brian has spent a lot of time researching this issue, and in this article presents his findings. Note especially his points regarding failover when using locally managed tablespaces.

As always, we invite articles from readers that have interesting performance ideas. Send all ideas to

Edtor@OracleMagician.com

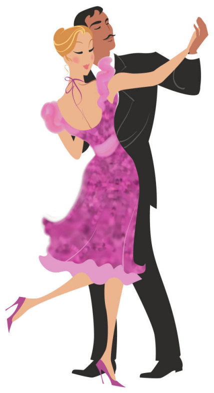
Chris Lawson

Editor

Graceful Failover & Failback Procedures in Non-Dataguard Environments

By Brian Keating

The document describes the procedures to run "graceful" failovers and failbacks of hot standby databases, in environments that are not using *Dataguard*.



What's a graceful failover?

A "graceful" failover is one that does *not* require databases to be opened with the "resetlogs" option – and as a result, graceful failbacks do *not* require the primary do not require the standby database's datafiles to be copied to the primary server.) Graceful failovers and failbacks are also known as "switchovers" and "switchbacks".

In order to avoid confusion, the *original* primary database will be referred to as database "abc"; and the *original* standby database will be referred to as database "xyz". In other words, before any of these procedures are run, the primary database is "abc", and the standby database is "xyz".

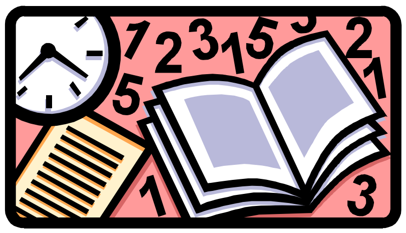
The procedures in this document are generally based on the information contained in *Metalink note ID 90817.1.*

Continued on page 2

Table of Contents

Continued from page 1

Part 1 – Failover procedures



1. Shut down database abc *and* database xyz, with ***shutdown normal*** or ***shutdown immediate***.

2. Copy the unsent archived redo logs, all of the online redo logs, and all of the control files from abc's host to xyz's host. Make sure that you copy all of those files into their pre-existing locations on xyz's host. For example, if a control file called "control01.ctl" exists in the /u01 directory on abc's host; but it exists in the /u04 directory on xyz's host, then copy that file from /u01 on abc's host to /u04 on xyz's host.

3. Start up database xyz, with "startup mount".

4. If necessary, rename the datafiles and online redo logs in database xyz, to reflect those files' locations on xyz's host, with "alter database rename file" commands.

5. In database xyz, execute the command "recover database;". If any archived redo logs are needed for recovery, then you will be prompted for the next log. In that case, type in AUTO at that prompt. When the instance finishes applying all of the changes contained in the archived redo logs *and* the online redo logs, the message "Media recovery complete." will be displayed.

6. In database xyz, type in the command "alter database open;".

The xyz database is now open in read-write mode; so at this point, that database is ready to begin being used as the *new* primary database.

7. In database xyz, type in the command "alter database create standby controlfile as *filename*;" (You must substitute a valid file name for the string *filename*.)

8. Copy the standby controlfile that you just created to abc's host. Then, on that host, use that standby controlfile to *overwrite* the existing, primary controlfiles on that server. In other words, copy the standby controlfile into each of the existing controlfile's directories; and then rename the standby controlfiles to match the name of the existing controlfiles.

9. Start up the abc database with "startup nomount" and "alter database mount standby database;" commands.

10. If necessary, rename the datafiles and online redo logs in database abc, to reflect those files' locations on abc's host, with "alter database rename file" commands.

At this point, the abc database is configured as the *new* standby database, and is ready to apply archive logs from the xyz database.

Part 2 – Failback procedures

(These procedures are basically identical to the failover procedures, above – the only difference is that you are reversing the "roles" of the abc and xyz databases.)

1. Shut down database abc *and* database xyz, with shutdown normal or shutdown immediate.

2. Copy the unsent archived redo logs, all of the online redo logs, and all of the control files from xyz's host to abc's host. Make sure that you copy all of those files into their pre-existing locations on abc's host. For example, if a control file called "control01.ctl" exists in

Continued on page 3

Continued from page 2

the /u01 directory on abc's host; but it exists in the /u04 directory on xyz's host, then copy that file from /u04 on xyz's host to /u01 on abc's host

3. Start up database abc, with "startup mount".

4. If necessary, rename the datafiles and online redo logs in database abc, to reflect those files' locations on abc's host, with "alter database rename file" commands.

5. In database abc, execute the command "recover database;". If any archived redo logs are needed for recovery, then you will be prompted for the next log. In that case, type in AUTO at that prompt. When the instance finishes applying all of the changes contained in the archived redo logs *and* the online redo logs, the message "Media recovery complete." will be displayed.

6. In database abc, type in the command "alter database open;".

The abc database is now open in read-write mode; so at this point, that database is ready to begin being used as the *new* primary database.

7. In database abc, type in the command "alter database create standby controlfile as *filename*;" (You must substitute a valid file name for the string *filename*.)

8. Copy the standby controlfile that you just created to xyz's host. Then, on that host, use that standby controlfile to *overwrite* the existing, primary controlfiles on that server. In other words, copy the standby controlfile into each of the existing controlfile's directories; and then rename the standby controlfiles to match the name of the existing controlfiles.

9. Start up the xyz database with "startup nomount" and "alter database mount standby database;" commands.

10. If necessary, rename the datafiles and online redo logs in database xyz, to reflect those files' locations on xyz's host, with "alter database rename file" commands.

At this point, the xyz database is configured as the *new* standby database, and is ready to apply archive logs from the abc database.

Part 3 – Requirements / Notes

This section describes the requirements which must be met, in order to use these procedures.

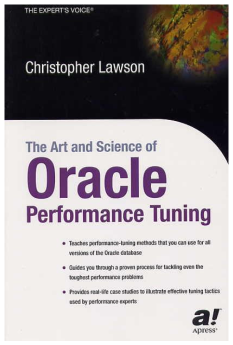
1. The primary database's host must be accessible. In order to use these procedures, the control files and redo logs from the primary database need to be copied from the primary database's host to the standby database's host. Therefore, these procedures cannot be used if the primary database's host is inaccessible – because that would prevent those files from being copied to the standby server. (It may be possible to configure a geographic disk mirroring utility, between the primary and standby hosts, in order to be able to eliminate this requirement; but I have not seen any successful implementation of geographic disk mirroring in Oracle environments.)

2. The primary and standby databases must be shut down with shutdown normal or shut-

Continued on page 4

Updates and Notes

If you find these tuning issues interesting, I discuss performance tuning in greater detail in my book, *The Art and Science of Oracle Performance Tuning*. It is available at most large bookstores or online at Amazon.com.



Continued from page 3

down immediate. If the databases (particularly the primary database) are shut down with shutdown abort, then it might not be possible to fully recover the standby database – and that would prevent these procedures from being usable.

3. These procedures include converting the original primary database into a standby database, after the original standby database is converted into a primary database. It is *not* required to convert the primary into a standby – it is still possible to fail back to the original primary, even if that database was not converted into a standby. However, if you do *not* convert the original primary database into a standby, then you must ensure that that database *never* gets opened, until you are performing the failback procedures. This is because the act of opening a primary database generates some redo in that database – and that redo will cause the primary database's datafiles to become permanently out of sync with the redo in the standby database. So, if you open up the original primary database without converting it to a standby, you will not be able to use these procedures – you will have to completely rebuild the primary database from a copy of the standby database's datafiles.

4. These procedures specify to copy the control files from the primary database to the standby database. An alternative to doing this is to *rebuild* the standby's control files, from the output of a "backup controlfile to trace". Note, however, that if you wish to rebuild the standby's control files, then you *must* ensure that the CREATE CONTROLFILE statement contains the NORESETLOGS option. If you change that option to specify RESETLOGS, then you will not be able to open the database with "alter database open;", you will be forced to use "alter database open resetlogs;" – and of course, the whole purpose of these procedures is to avoid opening the database with resetlogs.

5. There is an "idiosyncrasy" with these procedures, when they are used in conjunction with locally-managed temporary tablespaces. To be specific, *if* these procedures are used with a database that contains a locally-managed temporary tablespace, *and* if the tempfile(s) of that tablespace were *not* added to the hot standby database *before* the standby was converted into the new primary database, then that locally managed temporary tablespace will have to be completely dropped and re-created in the new primary database, in order to use the tempfiles in question.

The reason for this is that these procedures specify to copy the control files from the primary database to the standby database. Therefore, if you try to add the tempfile(s) to the standby database's tablespace, after converting it to the new primary, you will receive a "file is already part of database" error message – because the control files (which were copied from the original primary) will *already* contain the information about those tempfiles. Also, if the new primary instance tries to *use* any of the tempfiles (such as, for a sorting operation) then a "cannot identify/lock data file xxx" error message will be displayed – because the tempfile in question does *not* actually exist on the new primary's host. As mentioned above, one solution to this issue is to simply drop and re-create the locally managed temporary tablespace completely, after converting the standby database into a primary database. An alternate solution is to manually ensure that all tempfiles have been added to the standby, *before* converting the standby into the new primary.

Brian Keating is a Senior Consultant with Database Specialists, of San Francisco.

The Oracle Magician

Editor: Chris Lawson
OracleMagician.com