



Dedication | Experience | Results

White Paper:

Cloning Oracle Databases

Author: Randy Cunningham



Introduction

Cloning is a term applied to any of several methods used to duplicate a database to a different location on the same or a different host computing system. Ordinarily, this is accomplished by making a physical copy of the database datafiles. We presume a tacit requirement that the original database be unaffected – and often, uninterrupted – by any cloning procedure.

Application and Scope

Database cloning has a number of useful applications for both the Database Administrator as well as the organization served.

- Production support – Cloning a production database for purposes of providing ongoing application support is valuable, especially when recent data are required to investigate issues and to perform destructive testing.
- Application test – A production clone may be required in order to accomplish destructive testing of an application without adversely impacting the availability of the primary production environment.
- Upgrade/patch testing – Upgrading or patching a clone database enables validation of the process before applying it to the primary database. A cloned database can be thoroughly examined for unintended consequences of the change.
- Audit or information freeze requirements – A clone database as of the end of a quarter or year can prove helpful for satisfying audit requirements or other situations where production data must not change. Such a clone database might be made available as a read-only database.
- Reporting – An organization might require a reporting replica of production, either for reasons of performance management or in order to maintain data stability over a time interval.
- Rehosting – Cloning is an essential tool when a database is relocated from one host computing system to another.

Comparison with Other Replication Methods

Database cloning also underlies the set up of Oracle standby databases. The specific details of standby databases are beyond the scope of this paper; however, the principles of cloning a database are prerequisite to understanding and implementing a standby database as part of an organization's disaster recovery implementation.

Comparison with Other Replication Methods

Database cloning is but one of several tools available for replicating data from one place to another. The most significant differentiator between cloning and other methods is that cloning has minimal prerequisites at the destination. Ordinarily, the requirements are simply identical hardware architecture, an unmounted instance and adequate storage to receive the copied database. Other replication methods usually require that a destination database must exist and require that it must be open.

Datapump/export/import – By default, these utilities place the database content into a flat file, unless UNIX pipes are set up and configured for a coordinated transfer. Objects are created from DDL, resulting in alterations to object numbers and timestamps in the data dictionary. Even with a full export, there is no guarantee that all objects will successfully import and be in a valid state in the destination database. During import, indexes are rebuilt from scratch and statistics typically are gathered anew, resulting in a huge performance penalty compared to cloning. One virtue of Datapump, export and import is their granularity. Individual schemas and tables can be selected for replication. When cloning, the granularity is usually an entire database. Also, these utilities permit movement of data across differing architectures, something that is not supported with most techniques for physical datafile movement.

Advanced replication – Advanced replication handles replication of a variety of objects from a master database to one or more "replication sites," where an open database must already exist. While it is quite comprehensive in the

objects that it can replicate, not all objects or data types are handled. Advanced replication can also have a palpable performance impact on the source database environment.

Oracle Streams – Among the several capabilities of Oracle Streams is the capability to capture, propagate and apply data and object changes from one database to another. Oracle Streams requires that affected destination objects be instantiated on an existing, open database before changes are applied. One method of instantiation is cloning, so there is a possibility that cloning and Oracle Streams could be employed together to create and maintain a continuously up-to-date copy of a database.

Transportable tablespaces – Mechanically, transportable tablespaces most closely approach cloning. Physical datafiles, along with accompanying metadata produced by the export utility, are transported from one database to another and plugged into an existing, open database within a matter of seconds.

However, the tablespaces must be read only in order to be transported, quite possibly an unacceptable situation if an active, production database is the source. In addition, the system tablespace (and, hence, the data dictionary) cannot be transported.

There are complex replication scenarios where cloning and transportable tablespaces could be used together to consolidate data from multiple sources into a single, clone database.

Summary – There are two distinctive differences that separate cloning from all other replication methods:

- 1)Cloning is generally quicker than the other methods for duplicating a specific quantity of data; and
- 2)An existing destination database is not required – cloning creates the destination database as part of its process.

Planning to Clone

Cloning a database must not be approached as an ad hoc activity. For best results, advance planning and preparation should be performed. Here are some of the most important considerations:

Same or Different Host Computer? Cloning a database to the same host as the source requires specific steps to change the name of the database, as well as the pathnames of all files cloned, while cloning to a different host requires you to consider which method will be used to migrate files to that host. In general, the source and destination computing systems must be architecturally identical and must be running essentially identical operating systems.

Same or Different Database Name? If the database is duplicated to a different host system, you have a choice of whether or not to rename the database and whether file names will be identical to or different from those in the source database.

Hot or Cold Clone? If it is a requirement that the source database be cloned while it is up (hot cloning), then it is absolutely essential that the source database be operating in archivelog mode. If the copy is being made from a hot backup, all applicable archived logs must be accounted for and must be available on the destination host system.

Sufficient storage? Very simply, the clone database requires exactly the identical quantity of storage as its source. The logical storage contiguity footprint must be adequate to accommodate each datafile object to be copied.

Specific point-in-time or simply current time? If it is a requirement that the database copy be “as of” a specific point in time, consideration must be given to the availability of a sufficiently old backup and the availability of all archived logs needed to perform point-in-time recovery.

Source media? Duplication can be undertaken from the database itself (either while running or while down), or it can be performed from full database backups stored on disk or tape.

Method or tool to be used? Surveyed below are three different techniques for cloning an Oracle database. Each of them provides the same end result; not all of them are available or practical in every installation, due to differences among operating systems, SAN hardware, backup methods, operating policies, as well as site-specific roles and responsibilities. Each of them can be customized according to specific installation requirements.

Survey of Methods

Conventional OS and SQL commands (manual or scripted)

The best way to learn the mechanics of database duplication is to perform all of the steps using operating system commands and SQL commands available to DBAs. Initially, you might want to try this with a non-critical database on a development system. Scripting your work minimizes typographical errors and promotes functional repeatability.

Advantages

- Best way to learn the mechanics of database duplication
- Ability to implement customized functions, such as partial cloning and merge cloning
- Ability to implement your choice of host-to-host copying function into scripts

Disadvantages

- Requires ongoing maintenance
- Might prove to be less reliable than other methods, especially when used unattended

RMAN Duplicate Command

The Recovery Manager's DUPLICATE command provides a convenient, easy-to-use mechanism for database cloning. It is especially advantageous when the clone is to be made from a backup as of a specific point-in-time ("as of" clone). RMAN quietly takes care of many details of cloning, including thorough error checking and changing the database ID (DBID) so that the cloned database can be registered and managed using an RMAN repository.

Advantages

- Procedure is documented and supported by Oracle Corporation
- Lowest maintenance solution; quickest to implement
- Extensive error checking

Disadvantages

- No built-in capabilities for movement of data from one host system to another.
- Requires availability of backup media in order to perform the clone operation.

Split Mirror Cloning

Most contemporary SAN systems have software solutions available that enable "splitting a mirror," or making a copy of a logical volume, which copy can be independently mounted or copied to a location distinct from the original logical volume, either in the same SAN storage unit or possibly a different or even a remote unit.

Because synchronization is accomplished entirely in the storage array itself, I/O loads and queuing on the database host computer systems are eliminated. In addition, some systems such as EMC's TimeFinder®, have a built-in capability to track and copy only areas of a volume that have changed since a prior synchronization operation. This capability can result in very rapid synchronization rates on databases where a large portion of the database content is historical or seldom changes.

The granularity for this type of operation is a logical volume. Consequently, the clone database's storage footprint includes the same number and sizes of logical volumes as the original database. If routine cloning is anticipated, careful planning of the storage architecture should be performed for both the original and clone databases. Sharing of logical volumes by multiple databases should be minimized or avoided altogether. This method of cloning is especially well suited for databases that are stored on raw devices.

Oracle Corporation currently maintains a list of vendors on its web site who have validated their spit mirror solutions [5]. This list includes solutions such as IBM's FlashCopy, HP's StorageWorks Business Copy, EMC's TimeFinder® and others.

Advantages

- Potentially quickest method for cloning a database
- Minimizes the impact of I/O loading and performance impact on the database host computer systems

Disadvantages

- Requires use of system administrative level commands (UNIX root) to mount, umount and use LVM commands.
- Scripting can be quite complex, and typically incorporates vendor proprietary commands.

How to Clone: Step by Step

Following is a step by step overview of the cloning process. These steps could be used to create your own cloning procedure or scripts. When RMAN DUPLICATE command is used, steps 7) through 10) and step 14) are performed by the RMAN utility. Some of these steps only need to be performed for the initial cloning operation. For example, it is not necessary to create directories if the cloning operation is a "refresh" of an existing database. Prepare the Destination (first time cloning only)

- 1. Create directories. Create infrastructure directories in \$ORACLE_BASE/admin, plus any data directories required.**
- 2. Build a pfile (init.ora). Create an initialization parameter file adequate for starting an instance nomount. In some cases, simply copying the pfile from the source database and changing all occurrences of the source name to the destination name with a text editor will suffice.**
- 3. Create a password file. Use the orapwd utility to create a password file.**
- 4. Enable network connectivity. Using the Net Configuration Assistant, or other tool of your choice, ensure that a connection to the destination database can be established over the network.**
- 5. Start up the instance, NOMOUNT, to confirm that it can start. Then, perform a SHUTDOWN ABORT.**
- 6. Create an spfile [optional]. At this time you can prepare an spfile from the pfile, for example:
CREATE SPFILE='?/dbs/spfilezoom.ora' FROM PFILE;**

Build out the destination controlfile

- 7. You can copy the source controlfile to the destination controlfile in specific, limited circumstances:**
 - a) The source and destination databases are on different hosts; and
 - b) The source and destination databases will have the same name (at least initially).
- 8. Otherwise, connected to the source database as DBA or SYSDBA, execute the following two commands:**

```
ALTER SESSION SET TRACEFILE_IDENTIFIER='CCF';  
ALTER DATABASE BACKUP CONTROLFILE TO TRACE RESETLOGS;
```

9. Copy the controlfile creation script. Navigate to the source database UDUMP directory and search for a file containing the string CCF in its name. Copy the most recent such file, created in step 8, to the destination with the following path and name: \$ORACLE_BASE/admin/<instance>/create/CCF.sql

10. Edit the controlfile. Using a text editor, open the file just copied to the destination database directory. Do the following:

- a) Remove the prologue text from the beginning of the file;
- b) Change any remaining comments in the file from the "#" character to "--" (Oracle9i only).
- c) Locate a command beginning:
CREATE CONTROLFILE REUSE DATABASE ...
Change as follows:
CREATE CONTROLFILE REUSE SET DATABASE ...
- d) On this same command, confirm and change as required the ARCHIVELOG and RESETLOGS keywords.
- e) Change all occurrences of the source instance name to the destination instance name and change all occurrences of the source database name to the destination database name, as appropriate.
- f) Change the list of datafiles to be the names that the copied datafiles will have in the destination
- g) Save your work.

Copy the data files to the destination

11. There are many methods that can be used to copy the database data files from the source to the destination. Among the issues that should be considered include these:

Does copying occur while the database is up? If so, to avoid "fuzzy" copies, it is necessary to condition ORACLE. Do this by specifying ALTER TABLESPACE ... BEGIN BACKUP prior to copying and ALTER TABLESPACE ... END BACKUP immediately after copying each tablespace. In Oracle10g, you can specify ALTER DATABASE BEGIN BACKUP prior to copying all files and ALTER DATABASE END BACKUP after copying is complete. An even easier way is to copy with RMAN, using the BACKUP AS COPY command. Before hot-copying issue:
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;

Is the destination on a different host than the source? If so, an appropriate means must be used to transport the files to the destination. This might include such commands/tools as FTP, UNIX rcp, Windows drive mappings or NFS mounted file systems, to name a few. Files might be written to DVD and physically transported to the destination. Or, once again, RMAN could be used to restore the files from backup tapes.

Are raw devices being copied? To copy raw devices on a UNIX system, use the dd utility; on Windows use the OCOPY utility. To copy raw devices to a different destination host, use RMAN backups or vendor-supplied utilities. Are either the source or destination files managed by ASM? If so, be alert to requirements and restrictions described in [2], Chapter 11.

Are there adequate resources at the destination? Ensure that the destination has adequate space for copying and that required file system features, such as largefile support, are enabled.

NOTE: Do not copy temporary tablespace files (tempfiles) or online redo logs. Tempfiles will be created on the destination database, and recovery will be performed from [copies of] archived logs.

Example 1: Cold copy the ORCL starter database from a single-drive Windows system to a file share of C: on another Windows system, mapped as drive X: (this also could be done as a copy-paste operation from Windows Explorer):

```
C:\> COPY C:\ORACLE\ORADATA\ORCL\*.DBF X:\ORACLE\ORADATA\ORCL
```

Example 2: Hot copy the TEST database to the TRNG database on the same UNIX host (run from SQL*Plus):

```
SQL> ALTER TABLESPACE SYSTEM BEGIN BACKUP;
```

```

SQL> HOST cp /db01/oradata/test/system01.dbf /db13/oradata/trng
SQL> ALTER TABLESPACE SYSTEM END BACKUP;
SQL> ALTER TABLESPACE UNDOTBS BEGIN BACKUP;
SQL> HOST cp /db02/oradata/test/undotbs01.dbf /db15/oradata/trng
SQL> ALTER TABLESPACE UNDOTBS END BACKUP;
SQL> ALTER TABLESPACE TOOLS BEGIN BACKUP;
SQL> HOST cp /db03/oradata/test/tools01.dbf /db17/oradata/trng
SQL> ALTER TABLESPACE TOOLS END BACKUP;
SQL> ALTER TABLESPACE USERS BEGIN BACKUP;
SQL> HOST cp /db04/oradata/test/users01.dbf /db19/oradata/trng
SQL> ALTER TABLESPACE USERS END BACKUP;

```

Example 3: Cold copy the TRNG database to database VRFY on another UNIX host, empire (run from ksh):

```

oracle@origin:/home/oracle> rcp /db13/oradata/trng/system01.dbf empire:/db01/oradata/vrfy
oracle@origin:/home/oracle> rcp /db15/oradata/trng/undotbs01.dbf empire:/db04/oradata/vrfy
oracle@origin:/home/oracle> rcp /db17/oradata/trng/tools01.dbf empire:/db11/oradata/vrfy
oracle@origin:/home/oracle> rcp /db19/oradata/trng/users01.dbf empire:/db12/oradata/vrfy

```

Example 4: Hot copy the TEST database files to the TRNG database on the same UNIX host using RMAN (10g):

```

RMAN> BACKUP COPY OF DATAFILE '/db01/oradata/test/system01.dbf'
format '/db13/oradata/trng/system01.dbf';
RMAN> BACKUP COPY OF DATAFILE '/db02/oradata/test/undotbs01.dbf'
format '/db15/oradata/trng/undotbs01.dbf';
RMAN> BACKUP COPY OF DATAFILE '/db03/oradata/test/tools01.dbf'
format '/db17/oradata/trng/tools01.dbf';
RMAN> BACKUP COPY OF DATAFILE '/db04/oradata/test/users01.dbf'
format '/db19/oradata/trng/users01.dbf';

```

Switch Logs -- Copy archived logs to destination host (if different)

12. Once all hot copying of datafiles has been completed, you should archive the current on-line redo log in order to have a clean recovery position. From SQL*Plus, connected as SYSDBA to the source database, issue the following:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

13. You can consult the V\$ARCHIVED_LOG view on the source database to ascertain the log timestamps and log sequence numbers. This will provide you with an inventory of the archived logs that need to be copied to the destination host system for recovery on the destination database. If practical, copy the archived logs to the log_archive_dest directory for the destination database, using the destination database's log_archive_format. Doing so potentially avoids a significant amount of typing during recovery, in the next step.

Create the controlfile and open the new database

14. From SQL*Plus, connected as SYSDBA to the destination database, run the CCF.sql file which you edited in step 10. In theory, this step manages the remainder of the cloning process and opens the database for unrestricted user access. In practice, there are a number of possibilities for error, including these:

- a) The word SET was not inserted into the CREATE CONTROLFILE command.
- b) One or more datafiles or archived logs are missing or have been misspelled, either in the copy process in CCF.sql
- c) A copied datafile is fuzzy, or is not sufficiently old to perform recovery.

- d) All files are present, but have permissions or ownership issues that make them inaccessible to ORACLE.
- e) Resource issues, such as insufficient storage for the redo logs or tempfile occur.

15. Add tempfiles to all temporary tablespaces. The tempfiles need not be copied from the source database; they can be created freshly on the destination database with DDL:

```
SQL> ALTER TABLESPACE TEMP ADD TEMPFILE 'C:\ORADATA\SPCL\TEMP01.DBF';
```

Verify and finalize the new database

- 16. Confirm that the new database is complete by examining DBA_DATA_FILES, DBA_TEMP_FILES, V\$LOG. Compare the results with what is expected or with the equivalent views on the source database. Query a representative sample of application tables to confirm that no I/O errors or other unseemly symptoms are observed.**
- 17. If the new database is to be backed up using an RMAN repository, it will be necessary to change the database ID, to distinguish the clone from the original database. Use the DBNEWID utility (a.k.a. nid) to change the database ID.**
- 18. Verify the ARCHIVELOG mode of the database and ensure that archiving (if enabled) is occurring as expected.**
- 19. Confirm that a user (not SYSDBA) connection to the database can be established.**
- 20. Add the new database to /etc/oratab, to the intelligent agent and to other software or scripts, in a fashion similar to what you would use to add any new database to a host computing system.**

Cloning Using RMAN Duplicate

Cloning a database with Oracle's Recovery Manager utility is possibly the simplest and most convenient way to accomplish the task, even if RMAN is not your installation's choice for a backup/restore/recovery solution. Complete instructions for duplicating a database using RMAN are found in Oracle's reference materials [1], [2]. Here are a few caveats about using RMAN DUPLICATE:

- The terminology is not immediately intuitive. What is commonly called a "source" is called a "target" in RMAN; what is often called a "target" or "destination" is called an "auxiliary" when using RMAN.
- RMAN only duplicates from a database backup (which could be an image copy); there is no capability presently for direct database to database duplication.
- Recovery from a failure in an RMAN DUPLICATE operation can be daunting. Refer to [6c].
- Tablespaces that were transported into the source database using the transportable tablespace facility will not duplicate, unless they have been opened for read/write in that source database.
- Sample RMAN DUPLICATE script

The following example is the standard output of a successful RMAN DUPLICATE operation. Note that the original script itself, 16 lines long, is echoed toward the beginning of the output:

```
RMAN> connect target 'sys/*****@dbtx.dbalacrity.com'
2> connect auxiliary 'sys/*****@newdbtx.dbalacrity.com'
3> run {
4> # set until logseq 32;
5> allocate auxiliary channel ch00 type DISK;
6> allocate auxiliary channel ch01 type DISK;
7> allocate channel ch09 type DISK;
8> DUPLICATE TARGET DATABASE TO DBTX
9> LOGFILE
10> GROUP 1 ('/u06/oradata/dbtx/redo01.log') SIZE 200M,
11> GROUP 2 ('/u07/oradata/dbtx/redo02.log') SIZE 200M,
12> GROUP 3 ('/u06/oradata/dbtx/redo03.log') SIZE 200M,
13> GROUP 4 ('/u07/oradata/dbtx/redo04.log') SIZE 200M
```

```
14> NOFILENAMECHECK;  
15> release channel ch09;  
16> }
```

connected to target database: DBTX (DBID=3670308537)

connected to auxiliary database: DBTX (not mounted)

using target database controlfile instead of recovery catalog
allocated channel: ch00
channel ch00: sid=10 devtype=DISK

allocated channel: ch01
channel ch01: sid=11 devtype=DISK

allocated channel: ch09
channel ch09: sid=17 devtype=DISK

Starting Duplicate Db at 10-Dec-2004 21:08:26

printing stored script: Memory Script

```
{  
  set until scn 674431;  
  set newname for datafile 1 to "/dbtx/oradata/system/system01.dbf";  
  set newname for datafile 2 to "/dbtx/oradata/undo/undotbs01.dbf";  
  set newname for datafile 3 to "/dbtx/oradata/users/users01.dbf";  
  set newname for datafile 4 to "/dbtx/oradata/tools/tools01.dbf";  
  set newname for datafile 5 to "/dbtx/oradata/data/tax_data01.dbf";  
  set newname for datafile 6 to "/dbtx/oradata/index/tax_indx01.dbf";  
  restore  
  check readonly  
  clone database  
  ;  
}
```

executing script: Memory Script

```
executing command: SET until clause  
executing command: SET NEWNAME  
executing command: SET NEWNAME  
executing command: SET NEWNAME  
executing command: SET NEWNAME  
executing command: SET NEWNAME  
executing command: SET NEWNAME  
executing command: SET NEWNAME
```

Starting restore at 10-Dec-2004 21:08:26

```
channel ch01: starting datafile backupset restore  
channel ch01: specifying datafile(s) to restore from backup set  
restoring datafile 00001 to /dbtx/oradata/system/system01.dbf  
restoring datafile 00006 to /dbtx/oradata/index/tax_indx01.dbf  
channel ch00: starting datafile backupset restore  
channel ch00: specifying datafile(s) to restore from backup set  
restoring datafile 00002 to /dbtx/oradata/undo/undotbs01.dbf
```

```
restoring datafile 00003 to /dbtx/oradata/users/users01.dbf
restoring datafile 00004 to /dbtx/oradata/tools/tools01.dbf
restoring datafile 00005 to /dbtx/oradata/data/tax_data01.dbf
channel ch01: restored backup piece 1
piece handle=/dbtx/oradata/export/bk_2_1_544564711.BU tag=DUP_DEMO params=NULL
channel ch01: restore complete
channel ch00: restored backup piece 1
piece handle=/dbtx/oradata/export/bk_3_1_544564711.BU tag=DUP_DEMO params=NULL
channel ch00: restore complete
Finished restore at 10-Dec-2004 21:09:06
sql statement: CREATE CONTROLFILE REUSE SET DATABASE "DBTX" RESETLOGS ARCHIVELOG
MAXLOGFILES      5
MAXLOGMEMBERS    3
MAXDATAFILES     1000
MAXINSTANCES     1
MAXLOGHISTORY    226
LOGFILE
GROUP 1 ( '/u06/oradata/dbtx/redo01.log' ) SIZE 209715200 ,
GROUP 2 ( '/u07/oradata/dbtx/redo02.log' ) SIZE 209715200 ,
GROUP 3 ( '/u06/oradata/dbtx/redo03.log' ) SIZE 209715200 ,
GROUP 4 ( '/u07/oradata/dbtx/redo04.log' ) SIZE 209715200
DATAFILE
'/dbtx/oradata/system/system01.dbf'
CHARACTER SET WE8ISO8859P1
```

printing stored script: Memory Script

```
{
  switch clone datafile all;
}
```

executing script: Memory Script

```
datafile 2 switched to datafile copy
input datafilecopy recid=1 stamp=544568878 filename=/dbtx/oradata/undo/undotbs01.dbf
datafile 3 switched to datafile copy
input datafilecopy recid=2 stamp=544568878 filename=/dbtx/oradata/users/users01.dbf
datafile 4 switched to datafile copy
input datafilecopy recid=3 stamp=544568878 filename=/dbtx/oradata/tools/tools01.dbf
datafile 5 switched to datafile copy
input datafilecopy recid=4 stamp=544568878 filename=/dbtx/oradata/data/tax_data01.dbf
datafile 6 switched to datafile copy
input datafilecopy recid=5 stamp=544568878 filename=/dbtx/oradata/index/tax_indx01.dbf
```

printing stored script: Memory Script

```
{
  set until scn 674431;
  recover
  clone database
  delete archivelog
  ;
}
```

executing script: Memory Script

executing command: SET until clause

Starting recover at 10-Dec-2004 21:09:07

starting media recovery

channel ch00: starting archive log restore to default destination

channel ch00: restoring archive log

archive log thread=1 sequence=31

channel ch00: restored backup piece 1

piece handle=/dbtx/oradata/export/bk_4_1_544564729.BU tag=DUP_DEMO params=NULL

channel ch00: restore complete

archive log filename=/app/oracle/product/9.2.0/dbs/arch1_31.dbf thread=1 sequence=31

channel clone_default: deleting archive log(s)

archive log filename=/app/oracle/product/9.2.0/dbs/arch1_31.dbf recid=1 stamp=544568878

media recovery complete

Finished recover at 10-Dec-2004 21:09:09

printing stored script: Memory Script

```
{
  shutdown clone;
  startup clone nomount ;
}
```

executing script: Memory Script

database dismounted

Oracle instance shut down

connected to auxiliary database (not started)

Oracle instance started

Total System Global Area 789579952 bytes

Fixed Size 743600 bytes

Variable Size 570425344 bytes

Database Buffers 218103808 bytes

Redo Buffers 307200 bytes

printing stored script: Memory Script

```
{
  Alter clone database open resetlogs;
}
```

executing script: Memory Script

database opened

Finished Duplicate Db at 10-Dec-2004 21:09:18

released channel: ch09

Recovery Manager complete.

Common Pitfalls in Database Duplication

- Destination database won't recover because one or more files are "fuzzy". Be sure to use ALTER TABLESPACE BEGIN/END BACKUP or use RMAN to create datafiles from a source database that is open during the copy. (Symptoms: ORA-01194, ORA-01195, ORA-01208)
- Archived logs unavailable during recovery. Check V\$ARCHIVED_LOG and ensure that all required archives have been copied to intended directory for the destination database. If a suspiciously old log sequence number is requested, be sure that all of the datafiles have been copied to the destination.
- Problems with transportable or other read-only tablespaces. In general, data files for read only tablespaces should be brought on-line following a recovery using a backup controlfile. For transportable tablespaces, simply repeating the transport import operation might be easiest, after recovering and then dropping the tablespaces, including contents, on the destination database. (Symptoms: ORA-01159, ORA-01177, ORA-01189, ORA-01190, ORA-0120x)
- Oracle reports file existence errors (typically, ORA-01116 error in opening database file 'xxx') during CREATE CONTROLFILE or when attempting to recover or open the database. Be sure that all files have been copied under the intended names and that the CREATE CONTROLFILE, ALTER DATABASE RENAME FILE, or RMAN SET NEWNAME match the names that were copied. (Symptoms: ORA-01111, ORA-01116, ORA-01141, ORA-01179)
- Storage space errors are reported. Ensure sufficient space for the destination database files prior to copying.
- Database comes up and appears to be operational, but some users report receiving "ORA-25153 Temporary Tablespace is Empty." Be sure to ALTER TABLESPACE ... ADD TEMPFILE for all temporary tablespaces in the destination database.
- Script that worked previously, now reports errors. If structural changes are made to the source database, it is essential that scripts (especially scripts which issue CREATE CONTROLFILE) be revised to include the change.
- RMAN repository errors on the clone. Be sure to run the nid utility and register the new database in the repository.

References

- [1] Ashdown, Lance, et al. Oracle9i Recovery Manager User's Guide, Release 2 (9.2). Chapter 12. March 2002. Oracle Corporation Part No. A96566-01.
- [2] Romero, Antonio, et al. Oracle Database Backup and Recovery Advanced User's Guide, 10g Release 1 (10.1). Chapters 11, 12 & 16. December 2003. Oracle Corporation Part No. B10734-01.
- [3] Kodomudi, Shankar P (Oracle) & Manning, Paul (EMC). Split Mirror Backup and Database Replication Using EMC TimeFinder. http://www.emc.com/partnersalliances/pdfs/smbudr_tf.pdf
- [4] Kao, Poulin and Aschoff, John. User Guide for IBM Enterprise Storage Server (ESS) – Copy Services Functions with Oracle8i Databases. 23 July 2001. <http://www-1.ibm.com/servers/storage/disk/ess/pdf/ess-oracle2.pdf>
- [5] OSCP Snapshot Vendors. [List of Oracle Partners who have been validated through the Oracle Storage Compatibility Program (OCSP)]
http://www.oracle.com/technology/deploy/availability/htdocs/vendors_snapshot.html
- [6] Oracle Support Site (subscription required). <http://metalink.oracle.com>. Resources:
- [6a] Note: 18070.1 How to Make a Copy of a Database on the Same Unix Machine
- [6b] Note: 73301.1 How to make a copy of a database on the same Windows NT machine
- [6c] Note: 174625.1 RMAN: Incomplete Duplication and its Consequences
- [6d] Note: 228257.1 RMAN Duplicate Database in Oracle9i
- [6e] Note: 259767.1 Oracle10G RMAN Database Duplication

About the Author

Randy Cunningham is Solutions Architect with SageLogix, Inc., based in Englewood, Colorado, USA. He has worked with Oracle since version 4, and has been consulting exclusively to clients using Oracle products for the past nine years. His primary professional focus is on the architecture, implementation and performance optimization of ERP and data warehousing applications, with a secondary emphasis on database replication techniques.