Oracle9i for UNIX: Backup and Recovery Using a NetApp Filer in a SAN Environment

Richard Jooss | Brian Casper | 04/2004 | TR 3210

Abstract

This document covers the techniques for backing up and restoring an Oracle9*i*™ for UNIX® database using SnapDrive for UNIX® when a NetApp storage system is used for database storage.

Table of Contents

1. Purpose and Scope

This document covers the techniques for backing up and restoring an Oracle9*i*™ for UNIX® database when a NetApp storage system is used for database storage. Specifically, this report covers the following issues:

Backing up a database while the database is shut down ("cold" backup)

Backing up a database while the database is running ("hot" backup)

Recovering a database that failed while running in noarchivelog mode

Recovering a database that failed while running archivelog mode

2. Key Advantages

The performance and reliability of the backup and recovery operations are critical to effective database operation. NetApp provides unique functionality that affords the database administrator significant reliability and performance benefits in both backup and recovery. The key feature of NetApp's approach to database backup is the use of snapshots™. For more information on how snapshots work, see: *File System Design for an NFS File Server Appliance* by Dave Hitz, James Lau, and Michael Malcolm (http://www.netapp.com/tech_library/3002.html). Those with NetApp On the Web, or "NOW", access can also check the following NetApp®ONTAP 6.5 documentation, which covers snapshots in detail. (NetApp On the Web is NetApp's online support site. This site is open to all current NetApp customers.)

Snapshots are critical because they allow the database administrator to quickly and easily create an image of the entire Netapp Filer volume including the LUNs containing database contents whether the database is running on a filesystem or raw devices. Once the snapshot is taken, the database can be returned to normal operation. The snapshot can then be used as the backing store for a new read-write able LUN, and mounted as a new separate file system on the host server or another server. A backup can then be written to tape from the snapshot file system while the database is being used from the active file system. A server-free backup scheme can also be implemented by mounting the new LUN directly on secondary backup host. Actual backup-to-tape performance is of secondary importance, since the impact of this operation on the database's performance is negligible. For the backup operation, the paramount issue is that the database must either be:

Shutdown and restarted in a minimal amount of time ("cold" backup); or

Remain in "hot backup" mode for a minimal amount of time ("hot" backup)

During a "cold" backup, the database is shutdown and unavailable. Obviously, reducing the period of time during which this is the case provides a key advantage.

During a "hot" backup, the database runs in "hot backup" mode. This results in significant performance overhead, particularly in write operations. Further, being in "hot backup" mode is risky. Thus, getting the database out of "hot backup" mode more quickly also provides a key advantage.

Since the time required for the backup operation is reduced so much, many DBAs may find offline or "cold" backups become feasible when running a database with storage on a filer. Many databases cannot be shut down for the hours that may be required to perform an offline backup; however, shutting down the database for only a few seconds may be acceptable. The functionality of snapshots affords

the DBA the ability to shut down the database for only a few seconds, during which a snapshot is taken, and then bring the database back up again. "Cold" backups are preferable to "hot" backups for many reasons. For more information on this issue see: 4.1. "Cold" Backup.

Using hot backups, you can also take snapshots several times per day, and back up only one of them to tape. This offers the DBA additional flexibility. Effectively, you can take an online backup of your entire database every few hours. (Note: The amount of storage overhead associated with a snapshot is based upon the number of blocks that are different between the snapshot and the active file system. Using LUN storage requires space reservation to be enabled within the volume to ensure that the host will never see a volume full-induced write error. As a result, the initial snapshot requires one times the amount of disk space in use by the LUNs in the volume. Subsequent snapshots require incremental block allocation to cover the data blocks that have changed. As this increases, the size of the snapshot's storage requirements will increase as well. Thus, recent snapshots are generally less costly to store than older ones.) For more information on this issue see: 4.2. "Hot" Backup.

Snapshots also provide key advantages for the restore operation. If the data that you need is in a snapshot, you can restore the entire database, regardless of size, in a few minutes. This is accomplished with the Data ONTAP "snap restore" command. If you save several days' worth of snapshots, the chances are that you will never need to restore from tape at all, barring a failure on the filer itself. Imagine the following scenario. A company is running a 200 GB database on a UNIX platform, with storage on local disk. An errant application causes corrupt data to be written to the datafiles. As a result, the entire database must be restored from tape. At a rate of about 50 GB per hour, it will take approximately four hours to restore the data.

Now consider the same scenario with storage on a filer. Since the DBA saved a snapshot from a period of time prior to the failure, revert to a previous snapshot is all that is required to restore the datafiles. Using the SnapRestore™ ("snap restore") command, that can be accomplished in only a few minutes. This provides vastly improved mean time to recovery (MTTR) over that provided by the local disk solution.

For more information on restoring a database using the "snap restore" command see: 5.1. Restoring a Database Which Was Run in noarchivelog Mode and 5.2. Restoring a Database Which Was Run in archivelog Mode.

3. Requirements and Assumptions

This technical report covers backup and recovery of an Oracle9*i* database running under UNIX with database storage on a NetApp storage system. We assume that you are familiar with Oracle9*i* and the operation of NetApp filers. We also assume that you are familiar with the operation of your version of UNIX. All examples in this technical document are tested on Oracle9i Enterprise Edition under Sun Solaris Version 8. The scripts contained in this paper may require significant modifications to run under your specific environment.

The Oracle9i datafiles in our example are stored in a file system that is created on top of a LUN designated from space within a filer volume. For more information about creating LUNs on a filer for use with Oracle, see: "TR 307: Oracle9i™ for UNIX®: Integrating with a NetApp® Filer in a SAN Environment."

You must have your UNIX host configured to perform remote shell operations on the filer. This technical report assumes that the Oracle server machine has this capability. For instructions on how to set up remote shell access to a filer see: http://now.netapp.com/NOW/knowledge/docs/ontap/rel651r1/html/ontap/mgmtsag/admin6.htm. (Note: this link requires NOW access. For more information, please see: 2. Key Advantages.) The examples in this technical report use the following syntax:

```
rsh -l root data SomeCommand
```

On our filer, the following entry is placed in the hosts.equiv file:

```
sarak oracle
```

"sarak" is the name of our Oracle server machine, and "oracle" is the name of our Oracle user account. This entry in the hosts.equiv file on the filer allows the oracle user on sarak to become root on the filer for the purpose of running rsh commands. As long as this is properly administered, it does not create a security hole. For more security a ssh connection can be configured in place of the rsh connections shown in this paper. This also requires the IP address for the Oracle server machine to be specified in the "/etc/hosts" file on the filer, or that a properly administered DNS server be utilized to provide correct name resolution for the Oracle server.

The report also assumes that NetApp SnapDrive for UNIX is installed on the host and configured to access the NetApp Filer. SnapDrive performs the tasks of syncing and flushing the host filesystems to a stable state and creating a SnapShot on the Filer in one autonomous step. It also allows management of previously created SnapShots on the filers. More information on SnapDrive for UNIX can be obtained from: http://now.netapp.com/NOW/knowledge/docs/snapdrive/relunix10/html/index.shtml (Note: this link requires NOW access).

The space_reservations option must be enabled on any volume that contains a LUN. This feature guarantees that there will always be enough free space to perform a write operation to the LUN in case a snapshot has been taken and 100% of the blocks have changed. With NFS, a file system full error is generally tolerable at the host level since the filer and WAFL owns and manages the file system. With FCP and block devices, it is up to the host to manage the file system and therefore the host will always assume that 100% of disk is available to be overwritten. In the event that the filer volume is full, the host attempts to write to a block, and WAFL is unable to provide a free block, the filer will return a SCSI

write error to the host. This can be fatal for the host depending on what data is stored in the file system and what type of file system was created on the host. Enabling space_reservations will prevent this scenario.

The sample scripts in this technical report assume the following:

The name of the filer is "data".

The Oracle Home directory is “/export/home/oracle”.

The name of the Oracle instance is "fcpdb".

The NetApp volume “data:/vol/sundb” contains a LUN for the database datafiles and is mounted at “/u01”.

The NetApp volume “data:/vol/sunarchive” contains a LUN for the database datafiles and is mounted at “/u02”.

The following entries exist in the /etc/vfstab:

```
# sundb
/dev/vx/dsk/oradb/db  /dev/vx/rdsk/oradb/db  /u01      vxfs    2
yes     forcedirectio,logging,onerror=lock
# sunarch
/dev/vsxdsk/oralog/log  /dev/vx/rdsk/oralog/log  /u02      vxfs    2
yes     forcedirectio,logging,onerror=lock
```

4. Backup

The following sections contain SQL and operating system scripts that show a technique for backing up an Oracle database with datafiles stored on a NetApp filer.

4.1 "Cold" Backup

If the database is run in noarchivelog mode, the only option is to take an offline, or "cold" backup. A cold backup is often periodically taken for a database that is run in archivelog mode as well. A cold backup requires that the DBA shut down the database, back up all critical files using operating system commands, and then restart the database.

Taking a cold backup using a NetApp filer is a simple and fast operation. It requires three scripts, which have the following functions:

Script Name	Function
shutdown.sql	Shuts down the Oracle instance.
startup.sql	Starts up the Oracle instance.
docoldbackup.sh	A wrapper shell script that calls shutdown.sql, calls NetApp SnapDrive for UNIX to flush and suspend I/O to the file system, take a snapshot, resumes I/O to the file system, calls startup.sql, and performs the backup-to-tape from the snapshot. This script also does some housecleaning to keep four snapshots online at a time. If the script is run every night as a cron job, that means that four days' worth of snapshots would be available at any given time. NOTE: This script must be run as the root user to allows SnapDrive for UNIX operation. For more information, see: 2. Key Advantages.

The following is the text of the shutdown.sql script:

```
CONNECT / as sysdba
SHUTDOWN IMMEDIATE
EXIT
```

You may want to wrap your "SHUTDOWN IMMEDIATE" command inside a shell script that provides some warning and error processing. For more information about error handling within Oracle, refer to the documentation that came with your version of Oracle.

The following is the text of the startup.sql script:

```
CONNECT / as sysdba
STARTUP
EXIT
```

The following is text of the docoldbackup.sh shell script:

```
#!/bin/sh
# This script must be run as root to allow file system locking
set `/usr/bin/id`
if [ $1 != "uid=0(root)" ]; then
        echo "$0: must be run as root to allow snapdrive execution"
        exit 1
fi

# Shutdown the Oracle instance
su - oracle -c "sqlplus <shutdown.sql"

# Rename and delete old snapshots
snapdrive snap delete data:/vol/sundb:old3
snapdrive snap rename data:/vol/sundb:old2 old3
snapdrive snap rename data:/vol/sundb:old1 old2
snapdrive snap rename data:/vol/sundb:new old1


# Take a new snapshot (snapdrive flushes and syncs FS and vols)
snapdrive snap create -fs /u01 -snapname new


# Startup the Oracle instance again
su - oracle -c "sqlplus <startup.sql"

# Perform your backup using dump
# or other operating system commands here.
# For example:
# rsh -l root data dump 0ufbln rst0a 63 /vol/sundb/.snapshot/new/
```


The options for dumping data from a filer to tape or Nearstore are beyond the scope of this technical report. They are thoroughly documented in the *NetApp Data Protection Guides*, which can be found online at: http://now.netapp.com/NOW/knowledge/docs/ontap/rel651r1/html/index.shtml. (Note: this link requires NOW access)

You will need to set the privileges on the docoldbackup.sh shell script to allow it to be executed. The following commands will do this:

```
sarak# chmod 500 docoldbackup.sh
sarak# ls -l docoldbackup.sh
-r-x------   1 root     other        865 Jul 25 11:45 docoldbackup.sh
```

At that point, calling the docoldbackup shell script will do the following:

```
sarak# ./docoldbackup.sh
Sun Microsystems Inc.  SunOS 5.8       Generic Patch   October 2001

SQL*Plus: Release 9.0.1.3.0 - Production on Thu Jul 25 14:19:17 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.
```

```
SQL> Connected.
SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
snap delete: deleted snapshot data:/vol/sundb:old3
snap rename renamed snapshot data:/vol/sundb:old2 to new name old3
snap rename renamed snapshot data:/vol/sundb:old1 to new name old2
snap rename renamed snapshot data:/vol/sundb:old to new name old1
snap create: snapshot new contains:
        disk group oradb containing host volumes
                df (filesystem: /u01)
snap create: created snapshot data:/vol/sundb:new

Sun Microsystems Inc.  SunOS 5.8       Generic Patch   October 2001

SQL*Plus: Release 9.0.1.3.0 - Production on Thu Jul 25 14:19:33 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.

SQL> Connected to an idle instance.
SQL> ORACLE instance started.

Total System Global Area  235701300 bytes
Fixed Size                   279604 bytes
Variable Size             167772160 bytes
Database Buffers           67108864 bytes
Redo Buffers                 540672 bytes
Database mounted.
Database opened.
SQL> Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```


On our system the entire operation took approximately one minute. Note: the size of the database will not appreciably affect this interval, since the time required to take a snapshot is not dependent upon the size of the file system, or the files within the file system. Rather, a snapshot operation triggers a consistency point within the filer, and then simply copies the master inode of the WAFL file system to a new location. Thus, only about 4 KB of data is actually copied. For more information see: *File System Design for an NFS File Server Appliance* by Dave Hitz, James Lau, and Michael Malcolm (http://www.netapp.com/tech_library/3002.html).

4.2 "Hot" Backup

If the database is run in archivelog mode, you have the option to take online or "hot" backups. The database is open for user access while a hot backup is running. However, there is a significant performance hit, and an increased level of redo is generated. For this reason, it is critical that the database remain in hot backup mode for the absolute minimum period of time possible. NetApp

affords the DBA the opportunity to reduce the interval during which the database is in hot backup mode to only a few minutes.

Since a database in hot backup mode can be restored up to the point of failure in most instances, you should plan for recovery by organizing your files differently on the filer. The configuration recommended by NetApp for Oracle databases running in archivelog mode is to place the datafiles in a file system on a LUN on one volume, and the control file and log files (including the archived log files) in a different LUN on a second volume. This enables you to recover the datafiles using the "snap restore" command, while maintaining the control and log files intact. Once you apply the log files, your database is current up to the moment of failure. For more information on this, see: 5.2. Restoring a Database Which Was Run in archivelog Mode.

There is less margin for error when taking a hot backup. Since a cold backup using a NetApp filer takes about one minute, a cold backup may be a feasible solution for many databases. For more information see: 4.1. "Cold" Backup.

Taking a hot backup using a NetApp filer is a simple and fast operation. It requires five scripts, two of which are created during the operation. These scripts have the following functions:

Script Name	Function
dobegin.sql	Writes out the begin.sql script, and then calls it.
begin.sql	Places all of the tablespaces that need to be backed up into hot backup mode.
doend.sql	Writes out the end.sql script, and then calls it.
end.sql	Takes all of the tablespaces that need to be backed up out of hot backup mode.
dohotbackup.sh	A wrapper shell script which calls dobegin.sql, calls NetApp SnapDrive for UNIX to flush and suspend I/O to the file system, take a snapshot, resumes I/O to the file system, calls doend.sql, and performs the backup-to-tape from the snapshot. This script also does some housecleaning to keep four snapshots online. If the script is run every night as a cron job, that means that four days' worth of snapshots would be available at any given time. Alternatively, you may want to take a hot backup more frequently than once a day. NOTE: This script must be run as the root user to enable SnapDrive for UNIX operation. For more information, see: 2. Key Advantages.

The following is the text of the dobegin.sql script:

```
CONNECT system/manager
SET FEEDBACK off
SET PAGESIZE 0
SPOOL begin.sql
SELECT
    'ALTER TABLESPACE ' ||
    tablespace_name ||
    ' BEGIN BACKUP;'
FROM
    dba_tablespaces
WHERE
    CONTENTS <> 'TEMPORARY';
SPOOL off
@begin
EXIT
```

On our system (admittedly a small test database), the following begin.sql script is generated by dobegin.sql:

```
ALTER TABLESPACE SYSTEM BEGIN BACKUP;
ALTER TABLESPACE UNDOTBS BEGIN BACKUP;
ALTER TABLESPACE CWMLITE BEGIN BACKUP;
ALTER TABLESPACE DRSYS BEGIN BACKUP;
ALTER TABLESPACE EXAMPLE BEGIN BACKUP;
ALTER TABLESPACE INDX BEGIN BACKUP;
ALTER TABLESPACE TOOLS BEGIN BACKUP;
ALTER TABLESPACE USERS BEGIN BACKUP;
```

The following is the text of the doend.sql script:

```
CONNECT system/manager
SET FEEDBACK off
SET PAGESIZE 0
SPOOL end.sql
SELECT
    'ALTER TABLESPACE ' ||
    tablespace_name ||
    ' END BACKUP;'
FROM
    dba_tablespaces
WHERE
    CONTENTS <> 'TEMPORARY';
SPOOL off
@end
EXIT
```

On our system the following end.sql script is generated by doend.sql:

```
ALTER TABLESPACE SYSTEM END BACKUP;
ALTER TABLESPACE UNDOTBS END BACKUP;
ALTER TABLESPACE CWMLITE END BACKUP;
ALTER TABLESPACE DRSYS END BACKUP;
ALTER TABLESPACE EXAMPLE END BACKUP;
ALTER TABLESPACE INDX END BACKUP;
```

```
ALTER TABLESPACE TOOLS END BACKUP;
ALTER TABLESPACE USERS END BACKUP;
```

The following is text of the dohotbackup.sh shell script:

```
#!/bin/sh
# This script must be run as root to allow snapdrive execution
set `/usr/bin/id`
if [ $1 != "uid=0(root)" ]; then
        echo "$0: must be run as root"
        exit 1
fi

# Place all of the critical tablespaces in
# hot backup mode.
su - oracle -c "sqlplus system/manager @dobegin.sql"

# Rename and delete old snapshots
snapdrive snap delete data:/vol/sundb:old3
snapdrive snap rename data:/vol/sundb:old2 old3
snapdrive snap rename data:/vol/sundb:old1 old2
snapdrive snap rename data:/vol/sundb:new old1


# Take a new snapshot (snapdrive flushes and syncs FS and vols)
snapdrive snap create -fs /u01 -snapname new


# Remove all affected tablespaces from hot
# backup mode.
su - oracle -c "sqlplus system/manager @doend.sql"

# Perform your backup using dump
# or other operating system commands here.
# For example:
# rsh -l root data dump 0ufbln rst0a 63 /vol/sundb/.snapshot/new/
```


The options for dumping data from a filer to tape or Nearstore are beyond the scope of this technical report. They are thoroughly documented in the *NetApp Data Protection Guides*, which can be found online at: http://now.netapp.com/NOW/knowledge/docs/ontap/rel651r1/html/index.shtml. (Note: this link requires NOW access).

You will need to set the privileges on the dohotbackup.sh shell script to allow it to be executed. The following commands will do this:

```
sarak# chmod 500 dohotbackup.sh
sarak# ls -l dohotbackup.sh
-r-x------   1 root     other        946 Jul 25 15:23 dohotbackup.sh
```

At that point, calling the dohotbackup shell script will do the following:

```
sarak# ./dohotbackup.sh
Sun Microsystems Inc.  SunOS 5.8       Generic Patch   October 2001
```

```
SQL*Plus: Release 9.0.1.3.0 - Production on Thu Jul 25 17:01:49 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.0.1.3.0 - Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production

Connected.
ALTER TABLESPACE SYSTEM BEGIN BACKUP;
ALTER TABLESPACE UNDOTBS BEGIN BACKUP;
ALTER TABLESPACE CWMLITE BEGIN BACKUP;
ALTER TABLESPACE DRSYS BEGIN BACKUP;
ALTER TABLESPACE EXAMPLE BEGIN BACKUP;
ALTER TABLESPACE INDX BEGIN BACKUP;
ALTER TABLESPACE TOOLS BEGIN BACKUP;
ALTER TABLESPACE USERS BEGIN BACKUP;
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
snap delete: deleted snapshot data:/vol/sundb:old3
snap rename renamed snapshot data:/vol/sundb:old2 to new name old3
snap rename renamed snapshot data:/vol/sundb:old1 to new name old2
snap rename renamed snapshot data:/vol/sundb:old to new name old1
snap create: snapshot new contains:
        disk group oradb containing host volumes
                df (filesystem: /u01)
snap create: created snapshot data:/vol/sundb:new

Sun Microsystems Inc.  SunOS 5.8       Generic Patch   October 2001

SQL*Plus: Release 9.0.1.3.0 - Production on Thu Jul 25 17:01:53 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.0.1.3.0 - Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production

Connected.
ALTER TABLESPACE SYSTEM END BACKUP;
ALTER TABLESPACE UNDOTBS END BACKUP;
ALTER TABLESPACE CWMLITE END BACKUP;
ALTER TABLESPACE DRSYS END BACKUP;
ALTER TABLESPACE EXAMPLE END BACKUP;
ALTER TABLESPACE INDX END BACKUP;
ALTER TABLESPACE TOOLS END BACKUP;
ALTER TABLESPACE USERS END BACKUP;
```

```
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```

On our system the entire operation took approximately one minute. Note: the size of the database will not appreciably affect this interval, since the time required to take a snapshot is not dependent upon the size of the file system, or the files within the file system. Rather, a snapshot operation triggers a consistency point within the filer, and then simply copies the master inode of the WAFL file system to a new location. Thus, only about 4 KB of data is actually copied. For more information see: *File System Design for an NFS File Server Appliance* by Dave Hitz, James Lau, and Michael Malcolm (http://www.netapp.com/tech_library/3002.html).

4.3 NearStore Issues

After securing a snapshot within the filer, there are many optional ways to archive the data. Traditionally, a dump to tape process may be employed to create a long-term archival or off-line disaster recovery copy. Another alternative is to synchronize a copy to a second filer using SnapMirror. A new lower-cost option is to SnapVault the data to a NetApp NearStore.

There are many advantages to using a NetApp Nearstore device. Once the backup has been secured via SnapMirror or Snapvault, a backup to tape can be made independent of the host and without additional load on the primary filer. The SnapMirror/SnapVault process normally completes in a reduced timeframe over direct backup to tape. Since the NearStore is available most of the time for backup, the window for tape backups is much longer while maintaining filer performance and availability for the application. Also, the NearStore can be used to consolidate backup storage and centralize all backups from many filers. Perhaps the most compelling advantage is that rapid data recovery is possible since the datafiles can be brought back online directly without having to first restore from tape.

For a discussion of filer and Nearstore integration, see: *SnapMirror® and SnapVault™ Capacity Planning for the NearStore™* (http://www.netapp.com/tech_library/3151.html). For more information about Nearstore architecture, see: *NearStore Blueprint and Architecture Overview* (http://www.netapp.com/tech_library/3149.html).

5. Restore

The examples concerning restore in this technical report assume that the data required is still stored in a snapshot. If that is not the case (either because the snapshot has been deleted, or because the failure was on the filer itself) then you will first need to perform a restore from Nearstore or a restore from tape operation.

5.1 Restoring a Database Which Was Run in noarchivelog Mode

If your database was running in noarchivelog mode at the time of the failure, point-in-time recovery is the only option. You simply recover the database up to the time when the last cold backup was made.

Recovery is a manual process. The DBA should be intimately involved in each step of this process. For this reason, we do not supply a set of canned scripts to do a recovery. Instead, we illustrate the steps involved.

The steps for point-in-time recovery are as follows:

Shut down the database with the "shutdown abort" command (if the database is still running).

Restore the file system containing the datafiles (including the control file) from the last snapshot taken using the snap restore command.

Restart the database.

The following commands should be issued from within Server Manager to shut down the database prior to recovery:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Thu Jul 25 17:13:07 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.0.1.3.0 - Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production

SQL> shutdown abort
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```

Now the snapshot can be restored with the following command:

```
sarak# snapdrive snap restore -fs /u01 -snapname data:/vol/sundb:new
Starting to restore LUNs in disk group vgr
  WARNING:  This can take several minutes.
            DO NOT CONTROL-C!
```

```
            If snap restore is interrupted, the disk group
            being restored may have inconsistent or corrupted
            data.
  For detailed progress information, see the log file
/var/snapdrive/sd-recovery.log
        Importing oradb
snap restore: snapshot new contains:
        disk group oradb containing host volumes
                db (filesystem: /u01)
snap restore: restored snapshot data:/vol/sundb:new
```

Finally, start the database back up again:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Thu Jul 25 17:38:42 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.

Connected to an idle instance.

SQL> startup
ORACLE instance started.

Total System Global Area  235701300 bytes
Fixed Size                   279604 bytes
Variable Size             167772160 bytes
Database Buffers           67108864 bytes
Redo Buffers                 540672 bytes
Database mounted.
Database opened.
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```


The time required to do a recovery is minimal, only two or three minutes. This is true regardless of the size of the database, since the act of restoring a snapshot is merely the copying of a single 4 KB block. This allows the DBA to recover a database that can be many gigabytes in size very quickly.

5.2 Restoring a Database Which Was Run in archivelog Mode

A greater range of options is available when restoring a database that was running in archivelog mode at the time of a failure. Depending on the needs of the moment, the DBA can do a point-of-failure recovery, a cancel-based recovery, or a point-in-time recovery. You can also restore individual tablespaces or files. If the affected tablespaces or files are taken offline, you can even do the recovery while the database is open. The full range of techniques covering all of these options is far beyond the scope of this technical report. The example shown below assumes that the DBA wishes to perform a point-of-failure recovery of all tablespaces while the database is closed. (That is, the recovery will take the database forward to the point where the failure occurred.) The current control file and all archived

and online redo log files are assumed to be available. The most current backup of the datafiles is assumed to be in the snapshot called "new".

As stated in 4.2. "Hot" Backup, the database must be stored on two volumes in order to support point-of-failure recovery by using snap restore. One volume should have a LUN containing the datafiles, and the other volume should have a LUN containing the control file and the log files (including the archived log files). This allows you to run the snap restore command against the datafile volume only, and thus preserve your control and log files intact.

Recovery is a manual process. The DBA should be intimately involved in each step of this process. For this reason, we do not supply a set of canned scripts to do a recovery. Instead, we illustrate the steps involved.

The steps for point-of-failure recovery are as follows:

1) Shut down the database with the "shutdown abort" command (if the database is still running).
2) Restore the datafiles from the last snapshot taken using the "snap restore" command. (A current copy of the control file should be stored on the volume where the log files are located.)
3) Mount the database using the current control file referred to above.
4) Recover the database from the archived and online redo log files.
5) Open the database for user access.

The following commands should be issued from Server Manager to shut down the database:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Fri Jul 26 10:22:05 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.0.1.3.0 - Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production

SQL> shutdown abort
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```

To restore the LUN containing datafiles, run the following:

```
sarak# snapdrive snap restore -fs /u01 -snapname data:/vol/sundb:new
Starting to restore LUNs in disk group vgr
  WARNING:  This can take several minutes.
```

```
            DO NOT CONTROL-C!
            If snap restore is interrupted, the disk group
            being restored may have inconsistent or corrupted
            data.
  For detailed progress information, see the log file
/var/snapdrive/sd-recovery.log
        Importing oradb
snap restore: snapshot new contains:
        disk group oradb containing host volumes
                db (filesystem: /u01)
snap restore: restored snapshot data:/vol/sundb:new
```

Again, we assume that the control and log files are stored on a separate volume. Thus, this operation will not affect the control and log files. If any non-current copies of the control files have been restored during the previous operation, they should be replaced with a current copy in order for the recovery to continue. At that point, you should issue the following commands from within Server Manager to complete the recovery:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Fri Jul 26 10:49:37 2002

(c) Copyright 2001 Oracle Corporation. All rights reserved.

Connected to an idle instance.

SQL> startup mount
ORACLE instance started.

Total System Global Area  235701300 bytes
Fixed Size                   279604 bytes
Variable Size             167772160 bytes
Database Buffers           67108864 bytes
Redo Buffers                 540672 bytes
Database mounted.
SQL> recover automatic database ;
Media recovery complete.
SQL> alter database open ;

Database altered.

SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```

The database should now be running in normal mode, with all data intact up to the point of failure. 3317 titled “Connecting to SAN Snapshots on SUN”.

5.3 Obtaining file level access within a Snapshot

It is also possible to create a new LUN from a snapshot by utilizing the snapshot as backing store. If the new LUN is then mounted on a host system, access to full copy of the data is possible. The possible uses for this technology are limitless, and further discussion is beyond the scope of this technical report. However, as a primer, consider the following examples of where this technique may be useful:

Restoration of single database files

Clientless backups performed from a second UNIX host

Creation of a second database for data mining

Creating a backup of a production database for lab/development access

Disaster Recovery scenarios

Database reorganization testing

Host platform migration

Validating a backup before migrating to tape

The steps needed for creating a LUN with a snapshot as backing store are described in the TR 3317 titled "Connecting to SAN Snapshots on SUN".

5.4 Single Database file restore

Even given the ease and speed of restoring a database using SnapShots there are circumstances where it is not wanted or desired to restore the entire database. In these cases it is possible to get access to the individual database files using the techniques described in the technical report mentioned above and using the following methods to bring the database online with the replaced database file.

For the purposes of this example, the following steps will be taken:

extend the EXAMPLE tablespace onto a new datafile

take a snapshot of the volume

initiate a log switch to force updates to the active datafiles

introduce an error in the datafile

bounce the database to confirm the error

restore the single file from the snapshot

restart the database and perform media recovery with the archive logs

First we will create our new datafile by extending the tablespace EXAMPLE:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Mon Aug 5 17:06:12 2002

(c) Copyright 2001 Oracle Corporation.  All rights reserved.

Connected to an idle instance.

SQL> alter tablespace EXAMPLE
```

```
  2  add datafile '/u01/oradata/fcpdb/example02.dbf' size 1m ;

Tablespace altered.

SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```

Now we will take a snapshot of the volume utilizing the dohotbackup script from 4.2 "Hot" Backup.

```
sarak# ./dohotbackup.sh

[ … output abbreviated … ]
```

With the snapshot secured, user activity will eventually result in the active file system data becoming out of sync with the snap data A direct way to ensure that the active file system is more current from the database point in time view is to force a log switch. This will generate tablespace modifications that will be recorded in the archive logs that can be used for media recovery operations in the future.

The commands to force a log switch are as follows:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Mon Aug 5 17:31:24 2002

(c) Copyright 2001 Oracle Corporation.  All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.0.1.3.0 - Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production

SQL> alter system switch logfile ;

System altered.

SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```


To simulate a failure of the data in the LUN, we will delete the new datafile. For simplicity, the database will be shut down while this task is completed Upon restart of the database, the error indicates the inconsistent state of the database with the missing datafile.

```
sarak$ sqlplus " / as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Mon Aug 5 17:37:18 2002
```

```
(c) Copyright 2001 Oracle Corporation.  All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.0.1.3.0 - Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
sarak$ rm /u01/oradata/fcpdb/example02.dbf
```

Now that the datafile has been removed, the database is in an inconsistent state. Restart the database to confirm the error:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.3.0 - Production on Mon Aug 5 17:40:58 2002

(c) Copyright 2001 Oracle Corporation.  All rights reserved.

Connected to an idle instance.

SQL> startup
ORACLE instance started.

Total System Global Area  235701300 bytes
Fixed Size                   279604 bytes
Variable Size             167772160 bytes
Database Buffers           67108864 bytes
Redo Buffers                 540672 bytes
Database mounted.
ORA-01157: cannot identify/lock data file 9 - see DBWR trace file
ORA-01110: data file 9: '/u01/oradata/fcpdb/example02.dbf'


SQL> shutdown immediate
ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.3.0 -
Production
With the Partitioning option
JServer Release 9.0.1.3.0 - Production
```

Note that any number of errors or data corruptions could result in the necessity to restore a datafile. For the purposes of this example, we will assume that access to the data file is gained using the process described in the technical report 3317 titled "Connecting to SAN Snapshots on SUN" and copied to the original location.

Finally, restart the database and perform media recovery from the archive logs:

```
sarak$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.0.1.0.0 - Production on Mon Aug 5 18:37:13 2002

(c) Copyright 2001 Oracle Corporation.  All rights reserved.

Connected to an idle instance.

SQL> startup mount
ORACLE instance started.

Total System Global Area  235701300 bytes
Fixed Size                   279604 bytes
Variable Size             167772160 bytes
Database Buffers           67108864 bytes
Redo Buffers                 540672 bytes
Database mounted.
SQL> recover automatic database ;
Media recovery complete.
SQL> alter database open ;

Database altered.

SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.0.1.0.0 -
Production
With the Partitioning option
JServer Release 9.0.1.0.0 - Production
```

Single file restore is a valuable tool that can assist in database recovery where an entire volume does not need to be completely restored.

6. Conclusions

A NetApp storage system offers the Oracle DBA compelling advantages in terms of backup and recovery. Use of snapshots, combined with conventional backup-to-tape techniques, can dramatically optimize the Oracle database backup operation. Retaining a number of online snapshots allows the DBA to restore the database without the necessity to restore from tape in many circumstances. Backup and recovery performance is dramatically improved over conventional local disk configurations.

7. Caveats

For more information regarding use of NetApp storage systems with Oracle, see: TR3023 *Using ORACLE with a Multiprotocol Filer* by Bruce Clarke. You can also find support on Oracle's web site for NetApp filers. However, NetApp has not tested this configuration with any version of UNIX other than Sun Solaris 8, and has certainly not tested with all of the combinations of hardware and software options available on Solaris. There may be significant differences in your configuration that will alter the procedures necessary to accomplish the objectives outlined in this paper. If you find that any of these procedures do not work in your environment, please contact the author immediately.

7.1 Revision History

Date	Name	Description
6/20/2004	Richard Jooss	Update
7/23/2000	Brian Casper	Creation