CASE STUDY OF RMAN IMPLEMENTATION

Ketan Shah, Eaton Cutler-Hammer

SCOPE

This paper focuses on the real-world implementation of RMAN at Eaton Cutler-Hammer. It includes a brief explanation of various Oracle backup methods, requirements at Eaton Cutler-Hammer, the design of the RMAN backup solution and prototyping and tests that were done. Also included are the scripts that were used, and benefits of using RMAN and some future plans.

CHALLENGES AT EATON CUTLER-HAMMER

When we decided to build a backup solution for the Oracle database at Eaton Cutler-Hammer, we had the following challenges at hand:

1. We had HP, SUN and NT as our Oracle platforms. How can we architect a Multi-platform solution that is simple enough so that it can be deployed quickly?
2. The databases were growing by leaps and bounds. A few years ago, a 30G database was a very large database. In the coming years we would be looking databases that would be a few hundred gigabytes in size because of the growing business requirements of applications like e-commerce and datawarehouse. Keeping this in mind, the backup solution should be scalable, both in terms of number of databases and the sizes of the databases.
3. Traditionally, someone would go to the data center and load the tapes in each of the systems that the backups were supposed to run. What would happen if someone forgot to change the tapes? What about the special backups that we would need to do before performing an upgrade or a major maintenance. Manual tape changes in each system had to be eliminated.
4. Our databases were volatile and growing which caused us to continuously keep adding files to many databases. It only takes make one mistake for the backup to become become invalid. How do we overcome human error? We need as much automation as possible so that the human errors are minimized. We don't want to be in a situation when we find out that one someone forgot to add one file to the backup job that was added last week and now the database is crashed. Tough Luck! How do we eliminate problems with user errors?

We began with these challenges and came up with the following list of requirements for the backup solution.

1. Cost effective.
2. Standard backup solution.
3. Simple deployment.
4. Automated backup.
5. Easily deployable on all of the above systems.
6. Scalable.
7. Should be able to work with any media management vendor software with little or no modification.
8. Should be able to integrate easily with CH-OFA. CH-OFA is the Cutler-hammer adaptation of the Optimal Flexible Architecture.
9. Backup to disk and Tape library. We already had StorageTek in-house. We wanted to be able to leverage that.
10. No proliferation of passwords and independent of passwords, if possible.
11. Disaster recovery of the backup solution should accommodate business contingency plans.
12. All databases were archivelog mode. All backups should be online backups without degrading the normal system performance.

The following sections would list all backup methods that were considered and illustrate why we chose RMAN.

ORACLE BACKUP METHODS

Oracle is a complex database software that runs on a variety of platforms and operating systems. Oracle can be operated in a small single instance database to multi-instance OPS. It can be operated in *ARCHIVELOG* mode or in *NONARCHIVELOG* mode. It can run databases from a few hundred Megabytes to Terabytes. Oracle DBA's responsibility includes the backup and recovery of the database. Looking at the combination of all the operating environments, the backup and recovery of an Oracle database could be a big task and could take up a significant amount time, especially when we had several diverse platforms. We examined the following methods of performing the Oracle database backups:

- OS Level commands
- Third party backup solutions
- Oracle Tools

OS LEVEL COMMANDS

Each OS came with its own commands to backup the files from disk-to-disk and disk-to-sequential media, such as a tape. Depending upon the database mode, we could do an online backup or an offline or cold backup.

On Windows NT, we looked at COPY and BACKUP commands to copy the files of the database to a sequential media such as a tape. While COPY is a very crude command, BACKUP provides a little more flexibility as to which files needs to be backed up, depending upon the change time of the file or send the backup to a network or a floppy disk or a tape.

The problem with this kind of backup is that while it can be simple, there is no information about the backups that are performed. When restoring, one has to be absolutely certain about which files needs to be restored. This involved a many manual processes, which we did not want.

On UNIX systems, we examined the following commands. This is not a comprehensive list, but an outline of limitations of the Oracle backup methods have when deployed at the OS level.

- *fbackup*
  A slow filesystem level backup that allows backing up of files selectively. The selection of files is done by explicitly specifying the trees of the files to include or exclude from the backup session. One can perform a full backup or incremental backup. When performing the incremental backup, only files that have changed since the previous backup of the files are selected. *fbackup* also keeps a database for incremental backups which is a text file stored in a user-specified directory.
  A 36-track tape results in the highest performance, but requires manual changes of tapes if a backup spans multiple tapes. A DDS or DLT autochanger provides unattended backups. A magneto-optical autochanger can give unattended backup but with a very high cost.
  fbackup and *frecover* were designed for systems not having more than 1G file system storage. Hence, even though there is no limit as to how much you can backup or restore, complete backups and restores of substantially large systems can cause large amounts of system activity because of virtual memory used to store the indices.
  fbackup was designed to let backups run while the system is available. For absolute consistency, the system should be in single user mode. Because of this limitation; we could not use fbackup for an online backup because the datafiles continuously change when the database is running. We have seen instances, when the backup taken using fbackup was not good enough to restore the database.
  frecover is used to restore the files that were backed up using the fbackup.

- *vxdump*
  An incremental filesystem level backup for vxfs filesystems. *dump* can be used for non VxFS filesystems. It copies the files changed after a certain date to the magnetic tape. This backup should to be done with filesystem unmounted unless we buy VxFS Advanced package. This package allows the backup to be performed in a multiuser environment using a snapshot filesystem with the online backup facility. The way Oracle does the recovery allows the backup to be done online without unmounting the filesystem, because at the time of recovery, Oracle knows exactly what is the time this datafile is consistent up to.
  One issue with this type of backup was that if a filesystem has multiple files corresponding to multiple tablespaces, then we had to put all the tablespaces in backup mode and do the backup. Once the backup of the filesystem was done, we had to take the tablespaces out of backup mode. This was a bit tricky to program, considering the volatile nature of database structure because we were always playing games with the location of the data files. Every time when we took a backup, we had to dynamically find out all the tablespaces in the filesystem to be backed up and to make matters worse, consider the case where we had tablespace in 2 different filesystems.
  Another issue is that we needed to make sure that the backup would fit in one tape or otherwise an operator intervention was required.
  vxrestore is used to recover the files if required.
- *tar*
  tar saves and restores the archive of files on a magnetic media or disk files. With this command, you can not really have an incremental backup and also we can not have an online backup. Also, you can not create an archive for the files which are greater than 2G. Also any tape errors are handled ungracefully.

Although all the above commands allowed us to copy the database files on the sequential media, they were not enough to satisfy all of our requirements.

Database backups usually required a quite a bit of shell scripting in order to achieve a satisfactory backup. So it was complex solution to set up.

Only vxdump allowed us to do the online backup which again required a lot of complex programming.

Also, with tar or vxdump, we need to have a tape drive on each system and in some cases we need multiple tape drives. Which means that there is an additional hardware cost associated. And as the sizes of the database increases the speed of the backup reduces. So this solution was non-scalable and non-flexible.

Again, for different OS, we will need a separate set of scripts to do the backup.

THIRD PARTY BACKUP SOLUTIONS

There are many third party backup *media management* software solutions available. Each of which has a certain level of sophistication and reliability. On NT, we looked at Backup Exec from Veritas. We could label the tapes and keep a catalog of all the backups taken. At the time of restore, we can select any backup from the GUI that is provided. On UNIX, there are similar *media management* softwares like Legato Networker, Veritas, HP Omnibackup etc. We had Legato Networker at Cutler-Hammer. All these media management softwares store the information about the backup that was done and so it is easy to do the restore if necessary.

We found out that, if we use a media management software just by itself without a tape library, then we faced almost all of the problems that we faced with the OS level commands. So this method, even though it was better solution than the OS level commands was not complete enough for us.

NEW TECHNOLOGIES

With the advances in the enterprise backup technologies, we see the data centers of today equipped with what we call as *Tape Libraries* and one or more *Robots* control the tapes and the tape drives that comprise the Tape libraries. These are also called autochangers. These tape libraries are capable of storing backups from systems running different OS. So a data center can have a single tape library and all the systems will store their backup on it. This eliminates the need of buying a tape drive with each new system that an enterprise buys and the backup solution is ready once the system is in. StorageTek 9710 is the tape library that Cutler-Hammer owns.

The use of Autochanger along with the media management softwares was a big step forward, but it still required a lot of programming.

Vendors like Veritas, Legato, HP's *media management software* solutions use the underlying tape libraries and they also have modules that interface with various databases' backup methods including Oracle7's *EBU* and Oracle8's *RMAN*. These modules are usually sold at extra charge.

Then there are technologies like EMC's pathfinder, which is the mirroring technology and you take the backup by breaking the mirrors and taking the backup of the broken mirror and then after the backup is done, the broken mirror is again added to the system and pathfinder will resynchronize them. While this was an exciting technology, there was not enough cost justification for it.

ORACLE TOOLS

Oracle provides some tools to do the backup, but they generally have to be used with other technologies to get the backup to tapes.

- Export/Import - Export is an Oracle command used to take the logical backup. This is a logical backup of the Oracle database. One can take a partition level, table level, schema level or complete database export. Export will usually create a file, which is compatible with any platform. So export taken on one platform can easily be imported to another database on a different platform. The export file will then need to be copied to tape using any of the OS level command. Export is hardly a primary backup method for any shop, which has large databases. Also Export can not go back to a particular point in time. Exports are also bad for heavily used OLTP systems because it will create inconsistent backups. While it can be used to create consistent exports, it has other issues that come with it like rollback segments filling up. Import is used to do the restore.
- EBU - Oracle7 gives EBU as the backup/restore utility. If you are using the media management software, like Legato or Veritas, EBU gives you the flexibility to store the data on the tape library. EBU needs a catalog database and that catalog database stores the information about all the backups and the backup sets. This is the first flexible tool from Oracle where you can easily create the backups, online or offline, tablespace level or database level without worrying about the data files. It will find out the correct database file to backup. When doing an online backup, it puts the tablespaces in backup mode, and then copies the datafiles for that tablespace to the sequential media and then takes the tablespace out of the backup mode. When one needs to restore, EBU reads the catalog database and decides which backup set to restore. Once it decides which backup set to restore, it asks the media management layer to restore that backup set and the media manager will in turn restore the file for EBU. This tool was available only with Oracle7 and Oracle7 went out of support on Dec 31, 2000.
- RMAN - With Oracle8, Recovery Manager (RMAN) has replaced the EBU. This is a much more powerful tool than EBU in terms of the backups, recovery and the reports that you can generate to track the backups that are done.

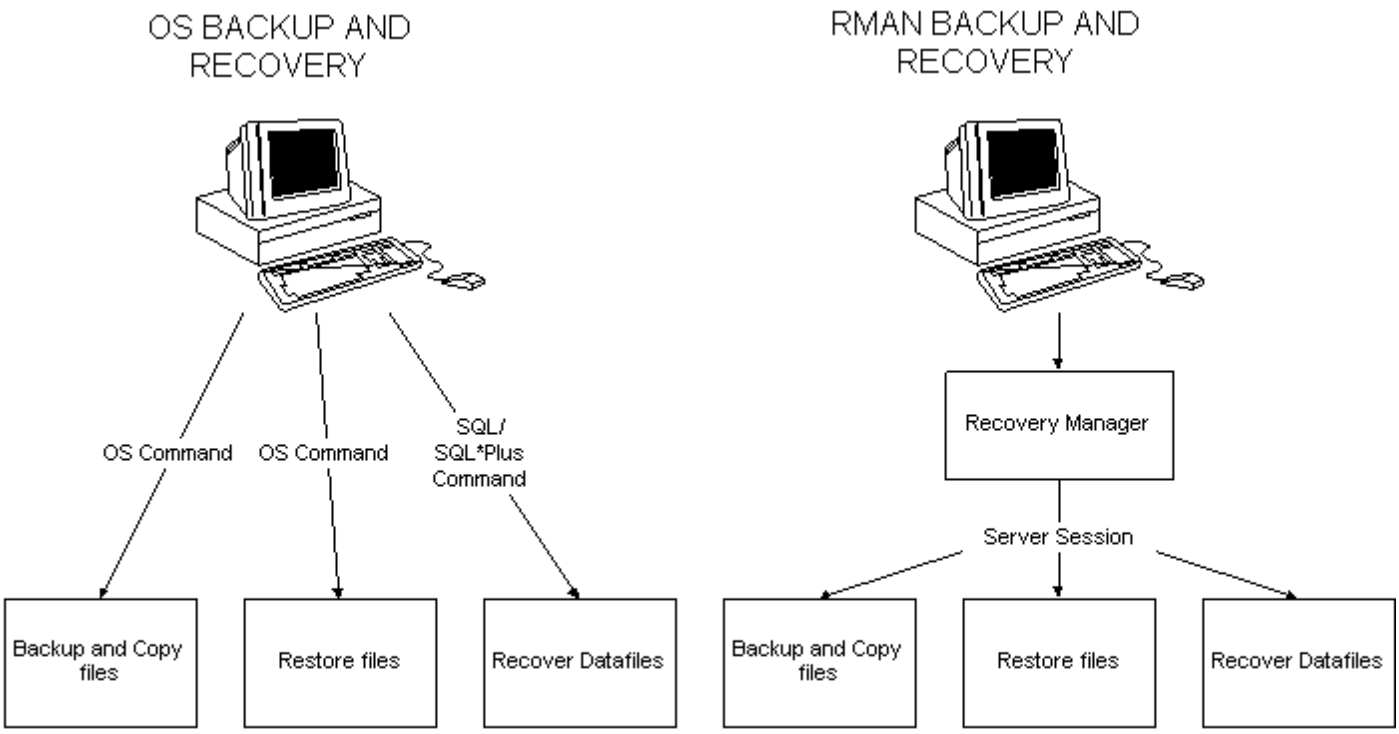

Figure 1

RMAN

The Recovery Manager (RMAN) is an Oracle utility that can be used to create backups, restore and recover database files and manage backups. It is included with the Oracle software, unlike EBU, which was an additional product. RMAN uses Oracle server sessions to automate the backup and recovery tasks. In addition to performing important backup and recovery procedures, the RMAN also greatly simplifies the job the DBA performs during these processes, which in turn significantly reduce the chance for human errors during backup and recovery operations.

RMAN provides three different interfaces for performing the backups.

1. Command Line Interface - This is most useful to the Database Administrators to automate the backup jobs.
2. Backup Manager - GUI tool integrated with OEM *(Oracle Enterprise Manager)*.
3. API - The application programmers use this one to create the custom backup tool.

How does RMAN work?

When we start RMAN, by default it opens two server sessions on the target database that is being backed up or recovered. If we are performing a tape or disk I/O, it requires that you allocate appropriate channel. Each channel corresponds to a server session. If you are using a catalog database, it opens a server session on the catalog database as well. RMAN uses PL/SQL interface to do the backup and restore operations.

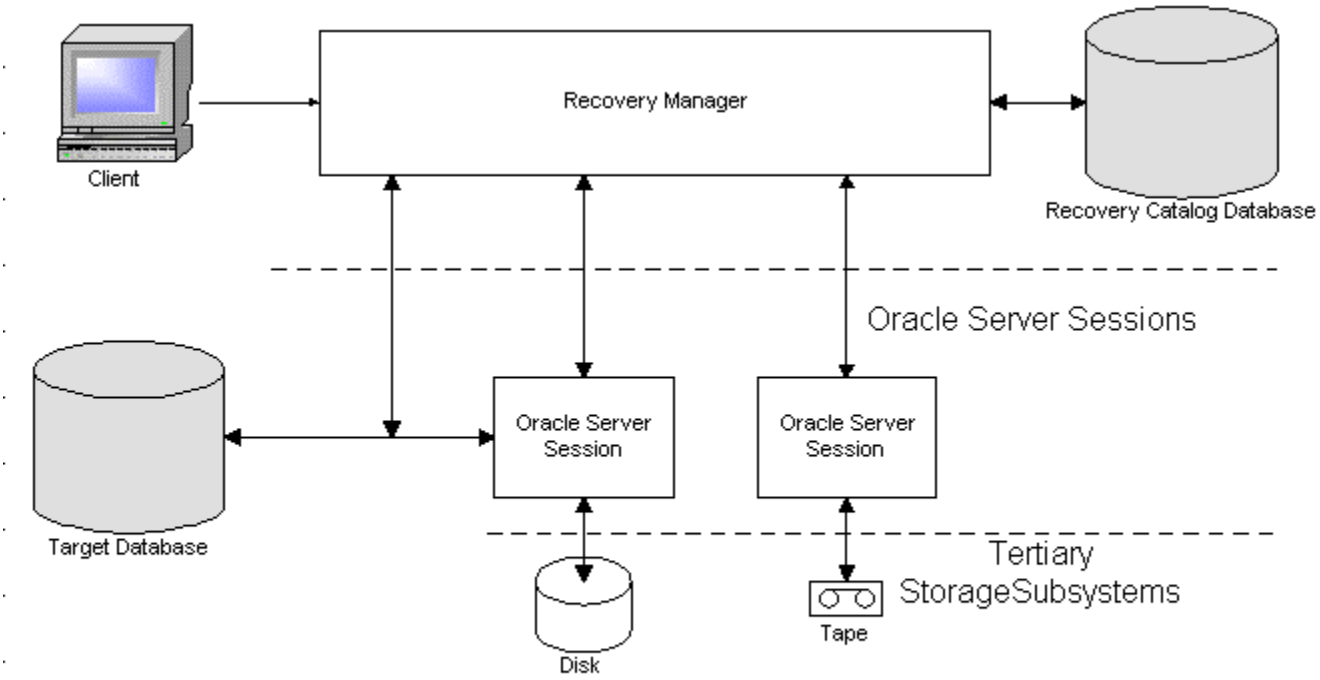

Fig 2

RMAN executes commands in 2 phases.

- Compilation
  During the compilation phase, RMAN actually finds out what object to operate upon and dynamically generates and compiles the PL/SQL programs that perform the required operations.
- Execution
  During the execution phase, RMAN executes the PL/SQL programs it generated during the compile phase.

RMAN is different than the normal operating system backups. It does backup or restore by the Oracle blocks. RMAN issues commands to Oracle server process. Oracle server process than reads or writes the datafile, control file or the archived redo log file that is being backed up or restored. When Oracle server process reads a datafile, it may encounter the split or fractured blocks. When it does, it will reread the block to get a consistent block.

What are split blocks? When performing open backups without using Recovery Manager, tablespaces must be put in hot backup mode in case the operating system reads a block for backup that is currently being written by DBWR, and is thus inconsistent. This phenomenon is known as "split" or "fractured blocks." When performing a backup using Recovery Manager, an Oracle server process reads the datafiles, not an operating system utility. The Oracle server process reads whole Oracle blocks, and checks to see whether the block is fractured by comparing control information stored in the header and footer of each block. If a fractured block is detected, the Oracle server process rereads the block. This is why, when using Recovery Manager to back up or copy database files, tablespaces do not need to be in hot backup mode. This has one advantage over the normal online backup methods: It does not generate the excessive redo when compared with the normal online tablespace backup method.

RMAN can spawn multiple server processes and hence can significantly improve the backup and restore operations if the system is capable of handling it. Usually, when the backup is being done, one can allocate multiple channels and each channel is a separate server process, running independently of the others, doing a backup of a different file.

Backups can either be written to disks or tape libraries. We can also write the backups to the disk and then copy that backup to the tape library. If we do that, we need to keep in mind that the restore will take a manual step to find the backup sets on the media manager catalog and restore them before rman can start its restore and recovery.

RMAN does not backup the online redo logs. The reason is that, in case of an accidental restore of the old online redo logs over the new ones can cause the loss of the valuable transaction current online logs and hence instead of a complete recovery, we may end up with incomplete recovery. Hence, if we are doing the backup and recovery of the database using the RMAN, it makes sure that you don't restore the online redo logs and hence eliminating the possibility of error when the crisis is in progress. *Always mirror your redo logs.*

RMAN keeps the information of all backups, backup sets and the databases in the recovery catalog. Recovery catalog is in the database's control file and optionally it can be stored in a separate database.

RMAN Features

- All RMAN operations are performed using the PL/SQL scripts executed by server sessions.
- RMAN uses a recovery catalog to store the metadata about the backup and recovery process. Recovery catalog is either stored in a separate Oracle database or it is stored in the control file of the database. That is why we see very large control files with Oracle8 and 8i.
- RMAN can store frequently used scripts in the catalog database if it is used.
- It compresses the backup by only including the blocks that are not empty. It knows about empty blocks because RMAN uses the server session and so it knows what an empty block is as opposed to an OS level utility.
- It provides a true incremental backup where it will backup only the blocks changed after the last backup. The other incremental OS level backups will copy the whole datafiles since the last backup.
- It easily integrates with the 3rd party media management softwares. So rman can easily create the backups directly to tape library as well as to the local disks. We had Legato Networker and Storagetek 9820 tape library. And rman was very easy to setup to use these technologies.
- It can easily parallelize the backup and recovery operations by allocating multiple channels.
- It performs crosschecks with the media manager to find out whether the backups exist or not in the media manager catalog.
- Checks for corrupted blocks during the backup.
- RMAN gives us a CLI which is uniform across all platforms just like SQL*Plus and so we can have one tool for backups on all our systems - HP, SUN and NT. And we can use the same scripts on each of the platforms to do the backup. Because we will be using the same script for all the systems we were able to create just a few scripts (Given later in the paper) and could be easily deployed.

Companion Technologies

RMAN by itself fell short of the requirements that we had. It could not backup database files to sequential media. So we needed some companion technologies like media manager and tape libraries to come up with an enterprise level solution. Oracle provides open media management API that is used by 3rd party vendors to build the modules that work seamlessly with RMAN. To do the backups to sequential media such as tape, we need to integrate the media management software with Oracle software. We had Legato Networker and Storagetek 9710 tape library (Autochanger). And with Legato Networker, we could get the Networker Module for Oracle(NMO) at extra charge. Once we install the NMO on database server, rman could directly backup the data directly to the tape library. Storagetek will usually have multiple tape drives, in our case 8 DLT drives.

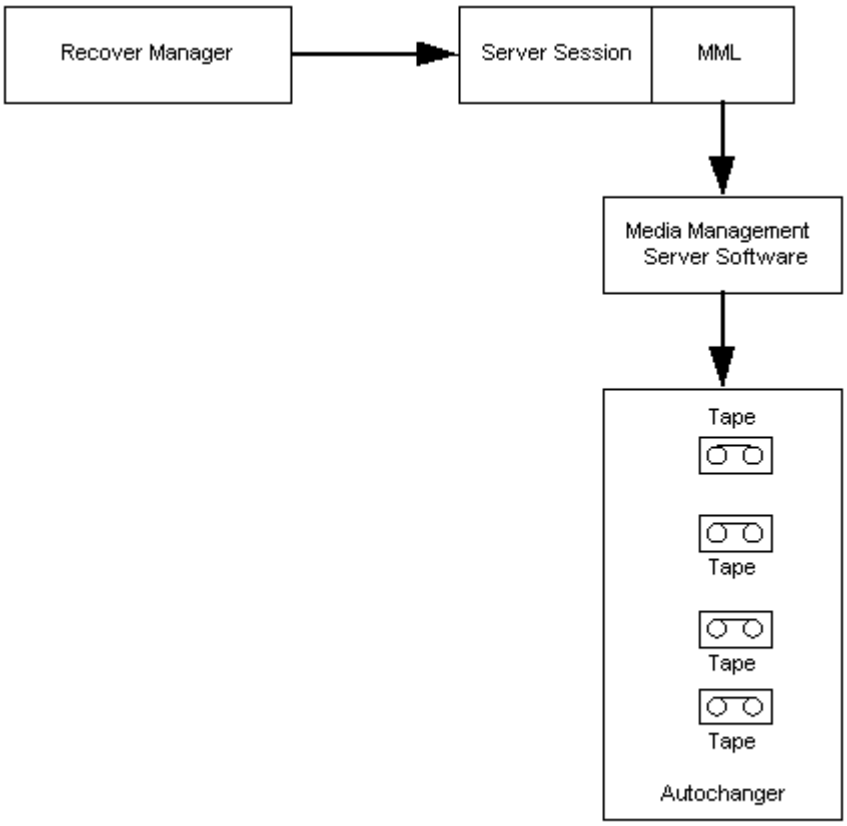

Fig 3

Some media management products can manage the I/O between the Oracle datafiles and the sequential media. Such products may use technologies such as high-speed connections between the storage subsystems and the tape libraries, thus removing the backup load from the primary database server.

To do the backups to disk, Oracle doesn't need any media management software.

RMAN BACKUPS ON TAPE

To do the backup to tape, Media manager software needs to be integrated with RMAN. The following rman script does the backup of a database to the tape controlled by the media management software.

```
run {
allocate channel t1 type 'sbt_tape';
backup database;
release channel t1;
};
```

When Oracle performs the above command script; it first asks media management layer identified by 'sbt_tape' to locate a tape that can store the backup and load it in tape drive. To get a list of all supported media devices can be found out from the V$BACKUP_DEVICE view. Media Manager labels and keeps track of each tape and names of files on each tape. With an autochanger with robotic arm, the Media Manager will automatically load and unload the tapes required by Oracle; if not, the Media Manager will request an operator to load a specified tape into the drive.

RMAN BACKUPS ON DISK

Oracle does not need to be integrated with media manager to do the backup to disk. Ideally, there should be separate disks, I/O controllers and channels to insulate database from the backup storage. This will also avoid the single point of failures of the database or your backup solution. The following script will do a backup to disk.

```
run {
allocate channel d1 type disk;
backup database format '/db03/obackup/sid/db/%u';
release channel d1;
};
```

Note that the script is not really different except that type is disk and the format is specified in the backup database command. This is done so that rman can write to /db03/obackup/*sid*/db directory. The %u parameter makes sure that the backup set that is created has a unique name than all the other backup sets.

RECOVERY USING RMAN

Here is an example of how RMAN can be used to do the complete database recovery.

```
run {
allocate channel t1 type 'sbt_tape';
restore database;
recover database;
release channel t1;
};
```

When RMAN issues the above script, Oracle finds out from the recovery catalog all the latest backup sets corresponding to the datafiles of the database. Media manager in turn requests its own catalog for those backup sets' information such as tape label of the tape containing the backup set. The media manager finds out that tape and loads it to the tape drive. It then tells Oracle about it and restores the database files. Once the restore is over, RMAN starts the recovery. The same process is repeated to find out all the archive logs required to recover the database and they are restored to directory specified LOG_ARCHIVE_DEST parameter in your *initSID.ora* file. It then automatically recovers the database to the current point in time.

RMAN AS EATON CUTLER-HAMMER SOLUTION

RMAN fulfilled most of the requirements that we had at Eaton Cutler-Hammer. It was cost effective because we didn't have to pay for it. It was simple to set up and it was scalable because of its parallelization options. It could be easily used to backup to the disk and tape library. We were able to leverage the existing resources like the Storagetek tape library and Legato Networker. It was not dependent on any one media manager software although Oracle ships Legato Netwoker with its software. It could be easily changed to use the Veritas Netbackup from Legato Networker. It was scalable because the size or number of databases being backed up didn't really matter as long as we had a tape drive to do the backup. And RMAN definitely reduced human error by automatically compiling the backup commands. Because of its consistent interface on all the systems, we decided that we would be able to use the same backup scripts on all the systems. Only thing that was different was command files(batch scripts) on NT to shell scripts on HP and SUN. For the requirements that were not fulfilled to begin with, we had enough level of confidence that we will be able to fulfill them during the implementation. One such requirement was that of security and passwords.

IMPLEMENTATION

We decided to approach the implementation through Designing, Prototyping, Testing the prototype, eliminating/solving any problems that come up during the tests and then finally putting the prototype in production. We decided build 2 prototypes, one on SUN Solaris and one on HP-UX.

DESIGN

One of the design goals was to make sure that all the tests that are listed in the next section are successful. That meant that we go ahead and set up RMAN on the HP and SUN systems that we selected, do all the tests and make sure that RMAN works the way it is advertised. Nothing is better then 'Trust But Verify' principle.

Next requirement was that the set up should be consistent on each of the systems. So we decided to create the rman scripts to do a particular type of backup, e.g. online backup to tape, online backup to disk, archivelog backup to tape and archivelog backup to disk etc. These scripts will be deployed on the systems in a specific directory on each system. We follow CH-OFA,

which is a CH implementation of Oracle's OFA architecture. And we have a directory that is pointed to by environment variables ORA_INSTANCE_PARM. Here is where all the rman script will reside. And when the RMAN backup is being run, the calling script will call the scripts in this directory.

Because we will be using the Tape Library with the Robot, we could directly write the backup to tape and with RMAN's parallelization features, we can keep the tapes streaming at the optimum speed. With 4 channels, we were able to keep the tapes back up 5-7 MB/s.

Security was another issue. We were using Legato to do the backup and it gives a special script, *nsrdmo.sh*, where we had to provide the connection information for the target and the catalog databases. When the Legato ran the rman backup, it printed all the connection information on log files including the passwords. This was unacceptable so we decided not to use nsrdmo.sh script.

RMAN mandates that you use the password files if you are running it remotely, i.e. if you are backing up a database that is not on the same system where you are running RMAN. So we decided to run RMAN on the same system running target database to be backed up. This meant that we created an OPS$ user on each of our databases with SYSDBA privileges. If we don't do this and run rman with the following command:

$ Rman target user/pwd catalog user/pwd …
then the ps command will display the userid and password of the target database. This was unacceptable. So we decided to run RMAN in using the following command:

$ Rman target / catalog user/pwd …

This meant that one can still see the userid and password of catalog database, but this is not compromising the security of the production databases. Now, Legato needed to run as root so connecting as '/' was not possible unless we change the group to dba. So we decided to run the rman backup in the pre step of Legato by doing an *'su'* to Oracle and running the rman script and actual backup step backed up the init.ora file for that database.

Another requirement was that the solution should be able to withstand the disaster. This meant that we could not lose the information of the backups that are done. So we decided to use the catalog database that is running on a system which is running the infrastructure softwares like the OEM and DNS etc. Now what happened if we lose the catalog database? We need something, which will allow for the disaster recovery of the catalog database itself. So we decided to create another catalog database, which will be set up off site. Every time we do a backup we resync both the primary and the secondary catalog databases.

To make the solution as cost effective as possible, we decided that for small databases, the backup should be done to the disks and from there the OS level backups will put the backup sets to the tape library. This meant that we have a pre step of Legato OS level backup that creates the backup sets using RMAN to a particular directory - /db*nn*/obackup/*sid*. The backup step will actually copy the backup sets created on the disks to the tape library. This allowed us to buy the NMO component only for the systems with large databases, where to set aside a lot of storage just to do the backups is not justifiable. The limit that we put was 5G for the backup to disk. Any database bigger than that meant that we buy the NMO.

All our databases are still small enough so that every night a complete database backup is still feasible. So we decided to take a complete database backups for all the databases.

PROTOTYPING

We selected 2 systems, one an HP-9000 running HP-UX 11.0 and running 8.0.6 and 8.1.6. We selected a SUN system running Solaris 2.6 and Oracle 8.1.6. We chose just Oracle 8.1.6 on Sun as we did not have any Oracle8 databases on Sun and we did not anticipate.

INSTALLATION OF SOFTWARE

We installed Oracle 8.0.6 and Oracle 8.1.6 on HP and Oracle 8.1.6 on Sun. We also had the Unix administrators install the Legato Networker and Networker Module for Oracle (NMO) on both the systems. Once the NMO is installed, we relinked the Oracle software to make sure that libobk.sl is proper. By default Oracle provides, libobk.sl with the disk interface, but we need to relink Oracle with the vendor supplied libobk.sl. Note that with Veritas Netbackup, this step is not required for the HP systems. All that is needed is that you change the symbolic link to libobk.sl to the Veritas supplied library.
Once we installed the software, we created a catalog database called "catp" and creates the catalogs for the 8.0.6 and 8.1.6

database using 'create catalog' command. We used a different user and schema for 8.0.6 and 8.1.6 databases in the catalog databases. The catalog database is running Oracle 8.1.6.

We set up a filesystem /db03/obackup for the backups to disks. We did not have the luxury of getting a separate controller for this set of disks.

TESTS

We performed the following tests to make sure that the backup and the recovery works as intended. It also helped us refine our scripts and thus helping us design a better solution. The trial and errors that were done during the testing phase also helped us learn various features of RMAN. This section just shows all the rman scripts and does not show all the shell scripting that went around it. That is described in a later section where all the scripts are given. All the scripts were run against the three databases on which the tests were done. Each of these scripts were run as follows:

$rman target / catalog rmanuser/catpwd@catp cmdfile rmanscript.rman log rmanscript.log

All the Oracle environment variables including NLS_LANG and NLS_DATE_FORMAT were set up before running the above command.

ONLINE BACKUP TO DISK

In this test we tested the rman backup to disk. The validity of the backup was verified using the complete restore. The script used to complete recovery is given later.

```
run {
allocate channel d1 type disk;
allocate channel d2 type disk;
setlimit channel d1 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel d1 kbytes 2097150 maxopenfies 32 readrate 200;
backup format '/db03/obackup/sid/db/df_%d_%s_%p_%t' database;
sql 'alter system archive log current';
backup filesperset 20 format '/db03/obackup/abc/arch/current/al_%d_%s_%p_%t' archivelog
all delete input;
backup format '/db03/obackup/sid/db/cf_%d_%s_%p_%t' current controlfile;
release channel d1;
release channel d2;
};
```

ONLINE BACKUP TO TAPE LIBRARY

In this test we tested the rman backup of the database to the tape library. The validity of the backup was tested by complete restore.

```
run {
allocate channel t1 type 'sbt_tape';
allocate channel t2 type 'sbt_tape';
allocate channel t3 type 'sbt_tape';
allocate channel t4 type 'sbt_tape';
setlimit channel t1 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel t2 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel t3 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel t4 kbytes 2097150 maxopenfies 32 readrate 200;
backup filesperset 4 format 'df_%d_%s_%p_%t' database;
sql 'alter system archive log current';
backup filesperset 20 format 'al_%d_%s_%p_%t' archivelog all delete input;
backup format 'cf_%d_%s_%p_%t' current controlfile;
release channel t1;
release channel t2;
release channel t3;
release channel t4;
};
```

PARTIAL BACKUP (TABLESPACE LEVEL BACKUP) ON TAPE

In this test we took a backup of a tablespace. The validity of the backup was verified by the restore and recovery of that tablespace.

```
run {
allocate channel t1 type 'sbt_tape';
allocate channel t2 type 'sbt_tape';
allocate channel t3 type 'sbt_tape';
allocate channel t4 type 'sbt_tape';
setlimit channel t1 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel t2 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel t3 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel t4 kbytes 2097150 maxopenfies 32 readrate 200;
backup filesperset 4 format 'ts_%d_%s_%p_%t' tablespace users;
sql 'alter system archive log current';
backup filesperset 20 format 'al_%d_%s_%p_%t' archivelog all delete input;
backup format 'cf_%d_%s_%p_%t' current controlfile;
release channel t1;
release channel t2;
release channel t3;
release channel t4;
};
```

PARTIAL BACKUP (TABLESPACE LEVEL BACKUP) ON DISK

In this test we took the backup of the tablespace to tape. The validity of the backup was verified by the restore and recovery of that tablespace.

```
run {
allocate channel d1 type disk;
allocate channel d2 type disk;
setlimit channel d1 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel d1 kbytes 2097150 maxopenfies 32 readrate 200;
backup filesperset 4 format '/db03/obackup/sid/db/ts_%d_%s_%p_%t' tablespace users;
sql 'alter system archive log current';
backup filesperset 20 format '/db03/obackup/abc/arch/current/al_%d_%s_%p_%t' archivelog
all delete input;
backup format '/db03/obackup/sid/db/cf_%d_%s_%p_%t' current controlfile;
release channel d1;
release channel d2;
};
```

RESTORE FROM DISK BACKUP

The restore from the disk backup was tested at multiple levels.

1. Loss of all the database files except the redo logs and the control files.
   We made some updates to the database and then did a complete database backup on disk. And we removed the database files while the database was running. We then did a 'shutdown abort'. And we restored using the following script. We made sure that the updates we made before the loss of the database was still there.

```
startup mount;
run
{
allocate channel d1 type disk;
restore database;
recover database;
alter database open;
};
```

2. Loss of all the database files except the redo log files.
   We made some changes to the database and then did a complete database backup on the disk. After that we removed all the database files except the redo logs. We then did a 'shutdown abort'. And we restored the backup using the following script. We made sure that the changes we made before the loss of the database were still there.

```
startup nomount;
run {
allocate channel d1 type disk;
restore controlfile;
alter database mount;
};
run {
allocate channel d1 type disk;
restore database;
recover database;
sql 'alter database open resetlogs';
};
reset database;
```

3. Loss of all the database files including the redo log and the control files.
   We made some updates to the database and did a complete backup of the database to disk. This included all the archive log backups also. We restored the control file first. Then we restored the database and recovered it. Because of the complete loss of all the database files, we had to do an incomplete recovery.

```
Startup nomount;
run {
allocate channel d1 type disk;
restore controlfile;
alter database mount;
};
run {
allocate channel d1 type disk;
restore database;
recover database;
};
```

   The recover database failed. So we did the following at the command line.

```
$ svrmgrl
SVRMGRL> connect internal;
SVRMGRL> recover database using backup controlfile until cancel;
SVRMGRL> cancel;
SVRMGRL> alter database open resetlogs;
SVRMGRL> exit;
```

Once the recovery is complete, the database was reset in the rman because it was opened with resetlogs.

RESTORE FROM THE TAPE BACKUP

The restore from the tape backup was tested the same way as the restore from the tapes except for the fact that the channel that was allocated was of type 'sbt_tape' instead of disk.

PARTIAL RESTORE FROM DISK

We made some changes on a table on the tablespace users. We did a tablespace backup of the tablespace users. Then we removed the datafile of the tablespace users. Once the datafile was removed, we did an 'ALTER SYSTEM CHECKPOINT;' in server manager to make Oracle give an error for the missing datafile. Then we used the following script to recover the users tablespace. The changes made to the table in the users tablespace was verified for the validity of the restore.

```
run {
allocate channel d1 type disk;
sql 'alter tablespace users offline immediate';
restore tablespce users;
recover tablespace users;
sql 'alter tablespace users online';
};
```

We also tested the following script to test the recoverability from the partial database backup.

```
run {
allocate channel d1 type disk;
sql 'alter database datafile ''/db02/oradata/abc/users01.dbf'' offline';
restore datafile '/db02/oradata/abc/users01.dbf';
recover datafile '/db02/oradata/abc/users01.dbf';
sql 'alter database datafile ''/db02/oradata/abc/users01.dbf'' online;
};
```

PARTIAL RESTORE FROM TAPE

The restore from the tape backup was tested the same way as the restore from the tapes except for the fact that the channel that was allocated was of type 'sbt_tape' instead of disk.

POINT IN TIME RECOVERY FROM DISK

We made some changes to the database and took a complete online database. We recorded the time of the changes (t1). Then,we made some more changes to the database and recorded the time again (t2). Then we removed all the datafiles of the database and did a shutdown abort. After that we ran the following script to recover the database to the time t1. After the database was recovered, we verified that the changes made before t1 was there in the database and changes made after the t1 were not there in the database.

```
Startup nomount;
Run {
allocate channel d1 type disk;
restore controlfile;
alter database mount;
};
run {
Set until time 't1';
allocate channel d1 type disk;
restore database;
```

```
Recover database;
alter database open resetlogs;
};
reset database;
```

POINT IN TIME RECOVERY FROM TAPE

The point in time recovery from the tape backup was tested the same way as the restore from the tapes except for the fact that the channel that was allocated was of type 'sbt_tape' instead of disk.

COLD BACKUP ON DISK

```
The cold backup using rman is also called a consistent backup. To do a consistent backup,
the database should be mounted but not open. We used the following script to do the
consistent backup.

shutdown immediate;
startup force dba;
shudown immediate;
startup mount;
run {
allocate channel d1 type disk;
backup filesperset 4 database format '/db03/obackup/abc/df_%d_%s_%p_%t';
};
alter database open;
```

COLD BACKUP ON TAPE

The cold database backup on tape is done in the same way as the backup to disk except for the fact that the channel that was allocated was of type 'sbt_tape' and format string did not have the directory name in it.

ARCHIVE LOG BACKUP ON DISK

Archived log backups were taken for the database on disk using the following script. It is the same script as for the database backup except that it does not have the backup database command. It just has the backup archivelog command. We have systems that will be doing a separate archivelog backup because of the amount of activity on the database.

```
run {
allocate channel d1 type disk;
allocate channel d2 type disk;
setlimit channel d1 kbytes 2097150 maxopenfies 32 readrate 200;
setlimit channel d1 kbytes 2097150 maxopenfies 32 readrate 200;
sql 'alter system archive log current';
backup filesperset 20 format '/db03/obackup/abc/arch/current/al_%d_%s_%p_%t'
archivelog all delete input;
backup format '/db03/obackup/sid/db/cf_%d_%s_%p_%t' current controlfile;
release channel d1;
release channel d2;
};
```

ARCHIVE LOG BACKUP ON TAPE

The archive log backup on disk has the same script, only change is in the channel type and the format string.

RESTORE TO ANOTHER HOST

This was one of the critical tests as the results of this test meant that we would be able to restore and recover a database of a failed host to another host and be in operation. We followed the following procedure :

1. Backup the database on host1 using any one of the online or offline backup script given above.
2. List all the datafile locations on host1.
   /db02/oradata/*sid*/system01.dbf
   /db02/oradata/*sid*/rbs01_1.dbf and so on.
3. Make backups available to host2.
   $ export ORACLE_SID=*sid*
   $ export BACKUP_DIR=*nsr_server*
   $ export NSR_SERVER=*nsr_server*
   $ export NSR_CLIENT=host1
   Here *nsr_server* is the Networker Server.
4. Restore init.ora on host2
   Restore the init*sid*.ora on host2 and make location specific changes e.g. user_dump_dest, log_archive_dest, control_files etc.
5. Setup connectivity to catalog database. (TNSNAMES information for catalog database)
6. Run the following RMAN restore/recover script in RMAN.

```
run {
#
# The following command is helpful when you need to stop a scheduled backup.
# You can query V$SESSION.CLIENT_INFO
set command id to 'rman';
#
# Step 1 : Allocate required number of channels.
  allocate channel t1 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  set command id to "RMAN";
  setlimit channel t1 kbytes 2097150 maxopenfiles 32 readrate 200;

restore controlfile;


# Step 6 : Release all channels.
  release channel t1;
}
alter database mount;
run {

#
# The following command is helpful when you need to stop a scheduled backup.
# You can query V$SESSION.CLIENT_INFO
set command id to 'rman';

# Step 1 : Allocate required number of channels.
  allocate channel t1 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  allocate channel t2 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  allocate channel t3 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  allocate channel t4 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  set command id to "RMAN";
  setlimit channel t1 kbytes 2097150 maxopenfiles 32 readrate 200;
  setlimit channel t2 kbytes 2097150 maxopenfiles 32 readrate 200;
  setlimit channel t3 kbytes 2097150 maxopenfiles 32 readrate 200;
  setlimit channel t4 kbytes 2097150 maxopenfiles 32 readrate 200;
```

```
set newname for datafile  1  to  '/db01/oradata/sid/system01.dbf';
set newname for datafile  2  to  '/db03/oradata/sid/rbs01_1.dbf';
set newname for datafile  3  to  '/db03/oradata/sid/rbs02_1.dbf';
set newname for datafile  4  to  '/db04/oradata/sid/temp01.dbf';
set newname for datafile  5  to  '/db01/oradata/sid/cms_table_space_sid.dat';
set newname for datafile  6  to  '/db01/oradata/sid/wf_table_space_ sid.dat';
set newname for datafile  7  to  '/db08/oradata/sid/cms_idxtable_space_ sid.dat';
set newname for datafile  8  to  '/db01/oradata/sid/ch_custom_01.dbf';
set newname for datafile  9  to  '/db04/oradata/sid/temp02.dbf';

restore database;

switch datafile all;

recover database;

# Step 6 : Release all channels.
  release channel t1;
  release channel t2;
  release channel t3;
  release channel t4;

}
sql 'alter database open resetlogs';
```

The restore script was not really different as we can see from the earlier script. Only thing that is different is the setup of the environment variables.

We did all the above tests, but what we needed was the simple few scripts that will do a full database backup and archive log backups. Also we needed a template for the recovery. So after verification of all the above tests, we decided to set up just a couple of rman scripts for the backup, one each for the database backup and the archive log backups. We also have a standard script template for the recovery. But recovery is always going to be manual process, as we will recover only the things that are necessary.

LESSONS LEARNED

During the testing phase, we learned a few of important lesson about the Legato Networker. We also learned a few things about rman as to how to set up and tune the backup processes.

LEGATO NETWORKER LESSIONS

To make the solution independent of the media management software, we decided not to rely on the Legato's set up of the backups. Legato has a special script called *nsrnmo.sh* where you put your backup script name and your target and catalog database passwords. One bad thing about this file is that that the log file that is generated during the backup operations will have all the passwords in it. This blows away the security of the database. Also, if we use the nsrnmo.sh, we would have been stuck with the Legato Networker. To go to some other media manager would mean that a lot of changes would have to be done. So we decided to go with the pre and post script of Legato. The Legato scheduler will fire a pre script before we take a Legato backup takes place and after the backup it will fire a post script. In the pre script, we actually call the rman backup script and during the Legato backup, we take the backup of init.ora file of the database. For the backup of the databases where we take the backup to disk and from there to tape, in the pre script, we do the rman backup of the database to disk and during the Legati backup, we take the backup of the /db*nn*/obackup/*SID* directory. So we will have the backup sets on disk untill the next backup runs. Pre step also eliminated the password issue because we could do "su - Oracle".
Any media manager scheduler will have a pre and post scripts implementation. So we have effectively insulated the set up against the Media manager changes.

RMAN LESSONS

- Set limit
  Set limit command helps Oracle throttle down the rman server processes back to that they don't eat up all the system I/O resources available to them. Backup is important, but the continuous system availability with adequate performance is more important. The following options are available and we use all of them in our script.

  - Kbytes - specifies the maximum size of the backup set. We use it to limit the size of the backup set to 2G.
  - Readrate - specifies the maximum no. of buffers(each of size DB_BLOCKSIZE * DB_FILE_DIRECT_IO_COUNT) per second read for backup or copy operations from each of the input datafiles. By default, this parameter is not set. Use this parameter to "throttle back" RMAN, that is, set an upper limit for block reads so that RMAN does not consume excessive disk bandwidth and thereby degrade online performance.
  - Maxopenfiles - controls the maximum number of input files that a backup command can have open at any given time. Use this parameter to prevent "Too many open files" operating system error messages when backing up a large number of files into a single backup set. If you do not specify maxopenfiles, then a maximum of 32 input files can be open concurrently.
- Change archivelog all crosscheck
  If you don't have any archive logs than the crosscheck command fails. So always create an archive log before running the crosscheck command.
- Stored scripts
  We decided not use the stored script because that meant that meant we had 2 copies, one in RMAN catalog and the other is a text file. We wanted less maintenance so we decided not to use the stored scripts.
- NSR variables
  To control which pool the backups go and what should be the retention period, we use the NSR_DATA_VOLUME_POOL and NSR_SAVESET_EXPIRATION in the allocate channel command of rman as follows:
  ```
  allocate channel t1 type 'SBT_TAPE' parms='ENV=(NSR_DATA_VOLUME_POOL=Main,
  NSR_SAVESET_EXPIRATION=2Week)';
  ```
  If we don't use these variables than in Legato the backup goes to the default pool and the backup never expires.
- Intermittent RMAN backup failures
  Due to a bug in RMAN in Oracle 8.1.6(give the bug number), the backup of a large database fails intermittently. One of the things that one can try is to use filesperset = 1. It has worked at our site, but there is no guarantee that it is a sure shot workaround.
- Asynchronous I/O
  In case of a problem, you might need to kill the backup. But to kill the backup you will need to kill the server sessions of rman processes. If you are using the asynchronous I/O and you kill the rman server process, it kills the instance most of the times on HP. We have not tested this on SUN or NT. *We did not use it because of that although Oracle recommends using asynchronous I/O to speed up the backup process.*

PUTTING IT ALL TOGETHER

We used a few scripts to set up all the backups. This section will give each script and the use of the script. We had the following scripts for the backup setup. Each script ends with a "# END OF SCRIPT #". The NT scripts are not included here but they will be same as the unix shell scripts in terms of the structure.

Files :

Directory	File
$ORA_INSTANCE/bin	*ORACLE_SID*_online_backup.sh
$ORA_INSTANCE/bin	*ORACLE_SID*_Archivelog_backup.sh
$ORA_INSTANCE/parm	Rman_online_backup.rman
$ORA_INSTANCE/parm	Archivelog_backup.rman
$ORACLE_PARM	Maintain_catalog.rman
$ORACLE_BIN	Notify.sh

ORACLE_SID_ONLINE_BACKUP.SH

This script is used to actually call the rman backup script - rman_online_backup.rman. There are 2 versions of this script, one each for the backup to disk and backup to tape.

BACKUP TO TAPE

```
#-------------------------------------------------------------------------------
#
#   oracle_sid_online_backup.sh
#   Cutler Hammer
#
#   Perform hot backup of sid database using RMAN and perform maintenance on
#   primary and secondary RMAN catalogs.
#
#-------------------------------------------------------------------------------

export job_name=oracle_sid_online_backup
export email_address=dba@mycompany.com
export NLS_DATE_FORMAT="DD-MON-YYYY HH24:MI:SS"

# The following 2 variables are used to determine if the CATP and CATP2
# Catalogs are available. RP for CATP and RP2 for CATP2.

RP=1
RP2=1

sid   # Point to correct database
if [ $? != 0 ]
then
   echo Error "pointing" to database
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   exit 1
fi

# Perform maintenance on primary RMAN catalog

echo "DOING CATP MAINTENANCE"
```

```
rman target / catalog rman/rmansys@catp cmdfile $ORACLE_PARM/maintain_catalog.rman
if [ $? != 0 ]
then
   echo Error doing maintenance on primary RMAN catalog
   echo "*** ${job_name}.sh Warning. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   RP=0
#   exit 1
fi

# Perform maintenance on secondary RMAN catalog

echo "DOING CATP2 MAINTENANCE"
rman target / catalog rman/rmansys@catp2 cmdfile $ORACLE_PARM/maintain_catalog.rman
if [ $? != 0 ]
then
   echo Error doing maintenance on secondary RMAN catalog
   echo "*** ${job_name}.sh Warning. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   RP2=0
#   exit 1
fi
# Perform backups
# Each instance's rman_online_backup.rman script will have to be changed to cater to
# that particular instance.

if [ $RP = 1 ]
then
   rman target / catalog rman/rmansys@catp cmdfile
$ORA_INSTANCE/parm/rman_online_backup.rman
   if [ $? != 0 ]
   then
      echo Error executing RMAN backups using CATP
      echo "*** ${job_name}.sh failed. ***"
      ksh $ORACLE_BIN/notify.sh FAILURE $email_address
      exit 1
   fi
elif [ $RP2 = 1 ]
then
   rman target / catalog rman/rmansys@catp2 cmdfile
$ORA_INSTANCE/parm/rman_online_backup.rman
   if [ $? != 0 ]
   then
      echo Error executing RMAN backups using CATP2
      echo "*** ${job_name}.sh failed. ***"
      ksh $ORACLE_BIN/notify.sh FAILURE $email_address
      exit 1
   fi
else
   echo No catalog database available for backup
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   exit 1
fi

# Resync secondary RMAN catalog

rman target / catalog rman/rmansys@catp2 <<EOF
resync catalog;
EOF
if [ $? != 0 ]
```

```
then
   echo Error resyncing catp2 database
   echo "*** ${job_name}.sh Warning. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
#   exit 1
fi

ksh $ORACLE_BIN/notify.sh SUCCESS dba@mycompany.com

echo "Normal end of script - ${job_name}.sh."
exit
```

END OF SCRIPT

BACKUP TO DISK

```
#-------------------------------------------------------------------------------
#
#   catp_online_backup.sh
#   Cutler Hammer
#
#   Perform hot backup of txp database using RMAN and perform maintenance on
#   primary and secondary RMAN catalogs.
#
#-------------------------------------------------------------------------------

export job_name=catp_online_backup
export email_address=dba@mycompany.com
export NLS_DATE_FORMAT="DD-MON-YYYY HH24:MI:SS"

# The following 2 variables are used to determine if the CATP and CATP2
# Catalogs are available. RP for CATP and RP2 for CATP2.

RP=1
RP2=1

catp   # Point to correct database
if [ $? != 0 ]
then
   echo Error "pointing" to database
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   exit 1
fi

echo '\n Deleting the following db backups from disk: \n'
find /db03/obackup/sid/db -type f -name "*CATP*" -print -exec rm {} \;
if [ $? != 0 ]
then
   echo Error deleting db backups from disk
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   exit 1
fi

echo '\n Deleting the following archivelog backups from disk: \n'
find /db03/obackup/sid/arch/history -type f -name "*CATP*" -print -exec rm {} \;
if [ $? != 0 ]
then
   echo Error deleting archivelog backups from disk
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   exit 1
```

```
fi

# Perform maintenance on primary RMAN catalog

echo "Doing maintenance on primary RMAN catalog."
rman target / catalog rman/rmansys@catp cmdfile $ORACLE_PARM/maintain_catalog.rman
if [ $? != 0 ]
then
   echo Error doing maintenance on primary RMAN catalog
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   RP=0
#   exit 1
fi

# Perform maintenance on secondary RMAN catalog
echo "Doing maintenance on secondary RMAN catalog."

rman target / catalog rman/rmansys@catp2 cmdfile $ORACLE_PARM/maintain_catalog.rman
if [ $? != 0 ]
then
   echo Error doing maintenance on secondary RMAN catalog
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   RP2=0
#   exit 1
fi

# Perform backups
# Each instance's oracle_sid_online_backup.rman script will have to be changed to cater to
# that particular instance.

if [ $RP = 1 ]
then
   rman target / catalog rman/rmansys@catp cmdfile
$ORA_INSTANCE/parm/rman_online_backup.rman
   if [ $? != 0 ]
   then
      echo Error executing RMAN backups using CATP
      echo "*** ${job_name}.sh failed. ***"
      ksh $ORACLE_BIN/notify.sh FAILURE $email_address
      exit 1
   fi
elif [ $RP2 = 1 ]
then
   rman target / catalog rman/rmansys@catp2 cmdfile
$ORA_INSTANCE/parm/rman_online_backup.rman
   if [ $? != 0 ]
   then
      echo Error executing RMAN backups using CATP2
      echo "*** ${job_name}.sh failed. ***"
      ksh $ORACLE_BIN/notify.sh FAILURE $email_address
      exit 1
   fi
else
   echo No catalog database available for backup
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   exit 1
fi
```

```
# Resync secondary RMAN catalog

rman target / catalog rman/rmansys@catp2 <<EOF
resync catalog;
EOF
if [ $? != 0 ]
then
   echo Error resyncing catp2 database
   echo "*** ${job_name}.sh failed. ***"
   ksh $ORACLE_BIN/notify.sh FAILURE $email_address
   exit 1
fi
sqlplus / << EOF
alter database backup controlfile to '/db03/obackup/sid/db/CATPcontrolfile.ctl';
quit;
EOF
ksh $ORACLE_BIN/notify.sh SUCCESS $email_address

echo "Normal end of script - ${job_name}.sh."
exit
# END OF SCRIPT #
```

ORACLE_SID_ARCHIVELOG_BACKUP.SH

This script is same as the online_backup script except that instead of calling the rman_online_backup.rman script it calls archivelog_backup.rman script.

RMAN_ONLINE_BACKUP.RMAN

This is the rman script and we have 2 versions of this script, one each for backup to disk and backup to tape.

BACKUP TO TAPE

```
#
# rman_online_backup.rman
#

run {

#
# The following command is helpful when you need to stop a scheduled backup.
# You can query V$SESSION.CLIENT_INFO
set command id to 'rman';

#
# Mark as "deleted" all archive logs which have been deleted from OS.
# Otherwise, you can't use "backup archivelog all" command.
change archivelog all crosscheck;

#
# Perform backups of database and archive log files

# Step 1 : Allocate required number of channels.
  allocate channel t1 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  allocate channel t2 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  allocate channel t3 type 'SBT_TAPE'
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  allocate channel t4 type 'SBT_TAPE'
```

```
   parms='ENV=(NSR_DATA_VOLUME_POOL=Main, NSR_SAVESET_EXPIRATION=2Week)';
  set command id to "RMAN";
  setlimit channel t1 kbytes 2097150 maxopenfiles 32 readrate 200;
  setlimit channel t2 kbytes 2097150 maxopenfiles 32 readrate 200;
  setlimit channel t3 kbytes 2097150 maxopenfiles 32 readrate 200;
  setlimit channel t4 kbytes 2097150 maxopenfiles 32 readrate 200;

# Step 2 : Backup the database.
  backup format 'df_%d_%s_%p_%t' database;

# Step 3 : Switch the archive log.
  sql 'alter system archive log current';

# Step 4 : Backup the archived redo logs.
  change archivelog all crosscheck;
  backup filesperset 20 format 'al_%d_%s_%p_%t' archivelog all delete input;

# Step 5 : Backup the control file.
# Even though "backup database" backs up the current controlfile, we need to back it
# up again.  The first backup doesn't contain information regarding the backups we just
# performed.
  backup format 'cf_%d_%s_%p_%t' current controlfile;

# Step 6 : Release all channels.
  release channel t1;
  release channel t2;
  release channel t3;
  release channel t4;

}
# END OF SCRIPT #
```

BACKUP TO DISK

```
#
# rman_online_backup.rman
#

run {

#
# The following command is helpful when you need to stop a scheduled backup.
# You can query V$SESSION.CLIENT_INFO
  set command id to "RMAN";

#
# Mark as "deleted" all archive logs which have been deleted from OS.
# Otherwise, you can't use "backup archivelog all" command.
change archivelog all crosscheck;

#
# Perform backups of database and archive log files

# Step 1 : Allocate required number of channels.
  allocate channel d1 type disk;
  allocate channel d2 type disk;
  setlimit channel d1 kbytes 2097150 maxopenfiles 32 readrate 200;
  setlimit channel d2 kbytes 2097150 maxopenfiles 32 readrate 200;

# Step 2 : Backup the database.
  backup format '/db03/obackup/sid/db/df_%d_%s_%p_%t' database;
```

```
# Step 3 : Switch the archive log.
  sql 'alter system archive log current';

# Step 4 : Backup the archived redo logs.
#  change archivelog all crosscheck;
  backup filesperset 20 format '/db03/obackup/sid/arch/current/al_%d_%s_%p_%t' archivelog
all delete input;

# Step 5 : Backup the control file.
# Even though "backup database" backs up the current controlfile, we need to back it
# up again.  The first backup doesn't contain information regarding the backups we just
# performed.
  backup format '/db03/obackup/sid/db/cf_%d_%s_%p_%t' current controlfile;

# Step 6 : Release all channels.
  release channel d1;
  release channel d2;

}
# END OF SCRIPT #
```

ARCHIVELOG_BACKUP.RMAN

The archivelog backup rman script is same as the online_backup.rman except that it does not have the backup database statements. We keep 2 versions of this file also - for disk and for tape.

MAINTAIN_CATALOG.RMAN

This is the script that we use to delete the old records from the rman catalogs. We have 2 versions of this script. The version for disk only deletes the information of backups that is 15 days old. This is because we have a 15 day retention policy on tape backups.

DISK VERSION

```
# This version is for backups to DISK.

# This script will delete records from the RMAN catalog which are more than
# two weeks old.

# Only records for the target database get deleted, not all databases.

allocate channel for maintenance type disk;

# Make sure there is at least one archivelog (otherwise you get an error).
sql 'alter system archive log current';

# Mark records as expired.  Because there is normally only one day's worth of
# backups on disk, we specify the "completed before" clause to make sure that
# information in the catalog does not get expired prematurely (Legato backups
# are kept on tape for 2 weeks).
crosscheck backup completed before "SYSDATE-15";

# Delete all "expired" records older than two weeks.
delete expired backup completed before "SYSDATE-15";
# END OF SCRIPT #
```

TAPE VERSION

```
# This version is for backups to TAPE.
```

```
# This script will delete records from the RMAN catalog for backups which
# no longer exist on tape.

# Only records for the target database get deleted, not all databases.

allocate channel for maintenance type 'sbt_tape';

# Make sure there is at least one archivelog (otherwise you get an error).
sql 'alter system archive log current';

# Mark records as expired for backups which no longer exist on tape.
crosscheck backup;

# Delete "expired" records.  We specify the "completed before" clause just
# to be safe.  No backups should ever expire sooner than two weeks.
delete expired backup completed before "SYSDATE-15";
# END OF SCRIPT #
```

NOTIFY.SH

This script notifies DBA's via email the success or failure of a backup.

```
#-----------------------------------------------------------------------
#
#   NOTIFY.SH
#   Cutler Hammer
#
#   Send email to Oracle DBA(s).
#
#   Parameters: $1 - SUCCESS or FAILURE
#               $2 - additional email addresses (optional)
#
#-----------------------------------------------------------------------
set +u   #  Don't want script to fail if parameters are not defined

echo "\nSending $1 Email Now: \c " ; date
mailx -s "${job_name}.sh $1" oracle DBA_MAILBOX@mycompany.com $2 < $logfile
# END OF SCRIPT #
```

FUTURE PLANS

Some of our future plans involves a move from Legato to Veritas Netbackup. This would involve a gradual phase out of the current backups and switch to the Veritas Netbackup. This will involve the changes in the rman backup scripts where we used the NSR variables. Nothing else changes. This is already under development. And so far we have not find a major problem.

Another future development is regarding the disaster recovery of the catalog database. What happens if both the catalog databases are unavailable? We can still do the backup using the control file and when the next backup runs, we can resync both the catalogs.

We are also looking at setting up asynchronous I/O for backup and resolve the issues related to it.

REFERENCES

Oracle8i Recover Manager User's Guide and Reference - Release 2 (8.1.6)

Oracle8i Backup and Recovery Guide - Release 2 (8.1.6)

Oracle8i Designing and Tuning for Performance - Release 2 (8.1.6)