Oracle8i Database Backup and Recovery: Planning Metrics and Rules of Thumb

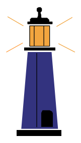
Using Compaq Hardware and Microsoft Windows NT

version 1.2
Last Modified September 19, 2000

by Thomas B. Cox

tbcox@att.net
http://www.geocities.com/tbcox23/

Acknowledgments: This paper was made possible by the generosity of Compaq Computer Corporation, who offered me the use of their Database Technology and Solutions Center (DTSC) Lab in San Bruno, California, and in particular by the sponsorship, patience, assistance, and tireless encouragement of the Lab's manager, Gary Stevens. Additional inspiration came from Ed Whalen, a good friend and a leading authority on database tuning.



May this paper be as a lighthouse, helping you steer your database safely away from the merciless rocks of unscheduled downtime.

Version Information

1.0 Initial version
1.1 Added Appendix G, walkthrough of creating a standby database using RMAN
1.2 Updated home page link to geocities; added Gnu Public License.

Copyright Notice

Database Backup Overview

What a backup is and why it is important

One of the guarantees of life, together with death and taxes, is that computer hardware and software will eventually fail. If you use a computer for anything important, you will probably want your important task to continue even if you should experience a hardware or software failure. The only protection you have from these failures is regular backups.

When you back up your data, you make a copy of that data that is logically identical to, but physically distinct from, the original – i.e. it gets stored somewhere else, on a different disk or tape, or on a different computer, or even in another city.

What a backup can and cannot protect you from

Database backups can protect you from most sorts of computer failures: loss of power, hard disk crash, disk controller failure, and 'soft' crashes caused by operating system and software bugs.

Backups can sometimes protect you from user mistakes, i.e. erroneously deleting the product list.

Backups cannot protect you from bugs in your application code that may record the wrong data. (In such an event, the backup will faithfully copy the wrong data that your application created.) And backups may not be able to protect you from disasters that you have not anticipated – having a complete backup computer identical to the original doesn't help if both are in the same room and the building burns down.

Definition of terms

Backup and recovery is a technical effort, and like any technical effort it uses a number of specialized terms. Some of these are:

Archive	To copy a file (such as a redo log) to a safe location for long term storage and possible later use
Archivelog	A mode in which the database requires filled redo logs to be archived as soon as they are filled; in this mode the database will not reuse a redo log until it has been successfully archived
Back up	To make a copy of (a database or file) as a potential replacement if the original becomes lost or damaged
Backup	A copy made by a back up process
Cold	Synonym for Offline
Full backup	A backup containing the entire database; a backup from which the entire database can be reconstructed
Hot	Synonym for Online
Incremental backup	A backup of changes since an earlier backup; can be used (together with an earlier full backup) to restore a database to a more recent state
Incremental level	Among incremental backups, defines what sort of changes are to be backed up, and which will be needed for recovery. A full backup is considered a 'level 0 (zero) incremental backup'; a level 1 backup contains all database changes since the most recent level 0 backup; a level 2 backup contains all database changes since the most recent level 1 backup; etc.
Log	see Redo Log
Offline	An object (i.e. database or file) not currently available to be looked at or changed
Online	An object (i.e. database or file) that currently available to be looked at or changed
Recover	To make a restored file current, as by applying incremental backups, archived logs, or online logs; also, to roll forward logged transactions to make a database current
Redo Log	A file in which all changes to the database are recorded as they are being made; a redo log file can be applied to an earlier backup copy of the database to bring it up to date
Restore	To replace a lost or damaged original file using a backup copy

Other terms of interest

In addition to the terms defined above, the Oracle DBA culture has adopted some semi-formal language that is useful in the context of backup and recovery, but that is not exactly technical, and that can differ from one Oracle shop to another. The following terms are not necessarily used rigorously or technically in this paper:

Verbs

One can 'make', 'take', or 'create' a backup. One can 'perform' a backup. Or one can 'back up' the database. One can 'run', 'start', or 'fire off' the backup procedures.

Nouns

The database as it exists before a particular change is applied is sometimes called the 'before image' and its state after the change is sometimes called the 'after image'. A particular change may be called a 'delta', or an incremental backup may be called a 'delta' between one state of the database and another. The 'backup device' is usually but not always a tape drive, and could actually be more than one device.

Planning your Backup Strategy

Create a Service Level Agreement (SLA) for database uptime and performance. This is the single most important step in both planning and managing your backup strategy. The SLA will state how long the database must be up and available each day, how much down time is acceptable, and how much data can be lost in a crash. Importantly, the SLA also states how long the database will take to come back up from a crash, or rather, from different sorts of crashes. Once you have the SLA in place, you can work out the backup and recovery strategy you will need to meet the agreement.

One useful way to look at the SLA is as a sort of interface specification between the DBA and the users of the database. The SLA defines, at a high level of abstraction, what the users should expect out of the database. The DBA is then free to work out various approaches and particular tasks to manage the database in a way that meets expectations.

These tasks should be part of a DBA Checklist[1], which is a document describing all of the recurring tasks (Operational Activities) that the DBA needs to perform in order to meet the SLA.

[1] Cox, Thomas B. (1999). DBA Checklist (white paper, available at http://www.geocities.com/tbcox23/)

Sample SLA #1: Light Usage

Here is a reasonable SLA for a departmental database of moderate importance:

Item	Service Element	Agreed Level
01.	System up time	8 hours a day, weekdays
02.	System hours available	8 a.m. to 4 p.m. local time
03.	Max data loss per crash	No more than one week's data
04.	Time to come up from a soft crash (power loss, etc.)	Time of power restoration plus 15 minutes to boot OS and start database
05.	Time to come up from a hard crash (hardware failure, etc.) damaging no database files	Unknown time to switch hardware and power up, plus up to 1 hour to restore any lost non-database files
06.	Time to come up from a hard disk crash that damages a drive containing database data	Unknown time to switch hardware and power up, plus 15 minutes to boot OS and start database, plus up to 3 hours to restore cold backup
07.	Database response times for representative queries/transactions	Query01: 90% under 0.5 sec Query02: 90% under 3.5 sec Update01: 90% under 0.7 sec
08.	Scheduled down time:	Weekdays 4 p.m. to 8 a.m. plus weekends
09.	Projected hours of unscheduled down time per year	Up to 3 days per year

Partial DBA Task List to Support SLA #1

Item	Operational DBA Tasks	Required Frequency	SLA Ref #
01.	Perform cold backup of database	Weekly	3
02.	Test Performance of Sample Queries	Daily	7
03.	Test backup and recovery	Monthly	3-6
04.	Examine logs of automatic startup, shutdown, and backup	Daily	1, 2, 9

As you can see, the uptime requirements of the database in SLA #1 are fairly forgiving. The DBA tasks to support this SLA are not difficult.

Sample SLA #2: Moderate Usage

Here is a reasonable SLA for a departmental database of high importance. It is largely the same as the previous one, except for items in ***Bold Italics***:

Item	Service Element	Agreed Level
01.	System up time	***12 hours a day, Monday-Saturday***
02.	System hours available	***6 a.m. to 6 p.m. local time***
03.	Max data loss per crash	No more than ***one hour*** of data
04.	Time to come up from a soft crash (power loss, etc.)	Time of power restoration plus 15 minutes to boot OS and start database
05.	Time to come up from a hard crash (hardware failure, etc.) damaging no database files	***Maximum 4 hours*** to switch hardware and power up, plus up to 1 hour to restore any lost non-database files
06.	Time to come up from a hard disk crash that damages a drive containing database data	***Maximum 4 hours*** to switch hardware and power up, plus 15 minutes to boot OS and start database, plus up to 3 hours to restore from backup
07.	Database response times for representative queries/transactions	Query01: 90% under 0.5 sec Query02: 90% under 3.5 sec Update01: 90% under 0.7 sec
08.	Scheduled down time:	***Weekdays 6 p.m. to 6 a.m., and all day Sunday***
09.	Projected hours of unscheduled down time per year	Up to 3 days per year

Partial DBA Task List to Support SLA #2

Item	Operational DBA Tasks	Required Frequency	SLA Ref #
01.	***Perform hot backup of database***	***Daily***	3
02.	Perform cold backup of database	Weekly	3
03.	Test Performance of Sample Queries	Daily	7
04.	Test backup and recovery	Monthly	3-6
05.	Examine logs of automatic startup, shutdown, and backup	Daily	1, 2, 9
06.	***Obtain a support contract that guarantees replacement hardware within 2 hours of placing a call***	***Annually***	5, 6

As you can see, the uptime requirements of the database in SLA #2 are much less forgiving than those in SLA #1. The DBA will find that the tasks to support SLA #2 are a bit more demanding.

Sample SLA #3: Heavy Usage

Here is a reasonable SLA for a corporate or Internet database of high importance. It is largely the same as the previous one, except for items in ***Bold Italics***:

Item	Service Element	Agreed Level
01.	System up time	***24 hours a day, every day***
02.	System hours available	***24 hours a day local time***
03.	Max data loss per crash	No more than ***one half hour*** of data
04.	Time to come up from a soft crash (power loss, etc.)	Time of power restoration plus 15 minutes to boot OS and start database
05.	Time to come up from a hard crash (hardware failure, etc.) damaging no database files	***Maximum 1 hour*** to switch hardware and power up, plus up to 1 hour to restore any lost non-database files
06.	Time to come up from a hard disk crash that damages a drive containing database data	***Maximum 1 hour*** to switch hardware and power up, plus 15 minutes to boot OS and start database, plus up to 1 hour to restore from backup
07.	Database response times for representative queries/transactions	Query01: 90% under 0.5 sec Query02: 90% under 3.5 sec Update01: 90% under 0.7 sec
08.	Scheduled down time:	***30 minutes once a month***
09.	Projected hours of unscheduled down time per year	Up to 1 day per year

Partial DBA Task List to Support SLA #3

Item	Operational DBA Tasks	Required Frequency	SLA Ref #
01.	***Perform hot backup of database***	***Daily***	3
02.	Perform cold backup of database	***Monthly***	3
03.	Test Performance of Sample Queries	Daily	7
04.	Test backup and recovery	***Weekly***	3-6
05.	Examine logs of automatic startup, shutdown, and backup	Daily	1, 2, 9
06.	***Obtain a support contract that guarantees replacement hardware will be stored on-site for immediate use***	***Annually***	5, 6
07.	***Ensure that staff are on duty at all times who are competent to replace failed hardware and perform all recovery tasks***	***Ongoing***	5,6
08.	***Optional: Maintain a 'hot standby' database server***	***Ongoing***	all

As you can see, the uptime requirements of the database in SLA #3 are even less forgiving than those in SLA #2. The DBA will find that the tasks to support SLA #3 are a lot more demanding.

Important tips for backing up

1. Keep two backups or two 'generations' of backup
2. Mix backup methods when possible; e.g. perform weekly cold disk backups and daily hot tape backups, or perform OS backups and EXPort backups
3. Keep detailed and methodical notes on every backup. If the backup is automated with a script of any kind, make sure the script is documented and that it writes a detailed log of its actions.
4. Create a written recovery plan for each type of database failure
5. Test every recovery plan; an untested recovery plan is dangerous
6. Practice database recovery at least once a month
7. Create and maintain a Service Level Agreement for each database, and use it to decide when and how to back up.
8. Network with other DBAs and stay current on best practices.
9. Look for news stories or DBA e-mail reporting new or unusual failure scenarios, and verify that such scenarios are anticipated in the existing recovery plan. If not, update the plan.

Types of backup

There are two basic types of backup available to users of an Oracle database: offline and online. Also, within online backups there is a further division between full and incremental backups. We now examine each of these options, with their advantages and drawbacks.

Offline Backup

An offline (or 'cold') backup is one that occurs when the database is offline, i.e. it has been shut down and is not in use during the backup. It is the most reliable form of backup, as well as the easiest to create and to use.

Create an Offline Backup

To create an offline backup, follow these steps:

1. shut down the database in NORMAL or IMMEDIATE mode (do not use SHUTDOWN ABORT – cold backups taken after a Shutdown Abort are useless and cannot be used for recovery)
2. back up the following files:
   - 2.1. each data file
   - 2.2. at least one copy of the control file
   - 2.3. the INIT.ORA file and any pfiles it may refer to
3. start up the database and resume normal use

Recover Using an Offline Backup

To recover the database using a cold backup, follow these steps:

1. Shut down the database (if it's up) in NORMAL or IMMEDIATE mode. (On Windows NT you can't use ABORT alone because it does not clear the operating system file locks and you can't overwrite the datafiles with your backup copies. If you must use Abort, you should then shut down and restart the operating system or use a file system utility to clear the file locks.)
2. Restore from backup the following files:
   - 2.1. each data file
   - 2.2. one copy of the control file (use it to replace all old copies)
   - 2.3. the INIT.ORA file and any pfiles it may refer to
3. Start up the database and resume normal use

Benefits of Offline Backup

An offline backup is the easiest to make, and is the easiest to recover from. Provided you follow the basic steps precisely, the risk of messing up the backup or recovery is low, as is the risk of a mistake leaving the database in an unrecoverable state.

Drawbacks of Offline Backup

Though easy to perform, an offline backup is inflexible. It can only be performed when the database is shut down, and thus is only an option for databases with fairly forgiving uptime requirements. It can only restore the database to the condition it was in when you shut it down for the backup – so any changes since then are always lost. And restoring is always a lengthy operation: should you suffer damage to just a single datafile, with an offline backup you have to restore every datafile from backup, not just the damaged one.

Online Backup

An online (or 'hot') backup is one that occurs when the database is online, i.e. it is running and may be in active use during the backup.

Create an Online Backup

To create an online backup, follow these steps:

1. verify that the database is in ARCHIVELOG mode (hot backups taken when the database is not in ARCHIVELOG mode are useless and cannot be used for recovery)
2. for each tablespace in the database, perform these steps:
    2.1. place the tablespace in Backup mode with the 'Alter Tablespace [ts_name] Begin Backup' command
    2.2. back up each data file for the tablespace
    2.3. take the tablespace out of Backup mode with the 'Alter Tablespace [ts_name] End Backup' command
3. back up the following files:
    3.1. one copy of the control file
    3.2. the INIT.ORA file and any pfiles it may refer to, if different from current

(Note that, if you use RMAN for online backup, you will need to follow a different procedure.)

Recover Using an Online Backup

To recover the database using a hot backup, follow these steps:

1. Shut down the database in NORMAL or IMMEDIATE or ABORT mode. (On Windows NT you can't use ABORT alone because it does not clear the operating system file locks and you can't overwrite the datafiles with your backup copies. If you must use Abort, you should then shut down and restart the operating system or use a file system utility to clear the file locks.)
2. Restore from backup the following files:
    2.1. each damaged file
    2.2. any archived redo logs needed to make restored files current
    2.3. one copy of the control file, only if damaged
    2.4. the INIT.ORA file and any pfiles it may refer to, if different from current
3. Start up the database
4. Recover the database and resume normal use

Benefits of Online Backup

An online backup is the most flexible sort of backup. It can be used to recover the database up to the very moment of failure, in most cases. Backups can be performed when the database is available, though they should not be performed during periods of peak use. If only a single datafile is damaged, usually only that file needs to be restored and recovered, making the restore from hot backup much quicker than restore from cold backup.

Drawbacks of Online Backup

Online backups can be very tricky to perform correctly, and even harder to use for recovery. Worse, a single mistake during the restore and recovery process can cause more damage than the original problem, and require a longer and more difficult fix. A single error can be fatal. In some cases certain mistakes can leave the database in an essentially unrecoverable state, requiring a full restore. And because the hot backup will generate various backup files at various points in time, you must be extremely well organized and methodical to stay on top of things.

Otherwise, you may have all the pieces you need to restore, but not know which pieces to use, in what order, or in what manner.

And since hot backups are usually reserved for the most important databases, you will face this tricky recovery process at a time when the pressure on you is high to get the database up quickly – which raises the risk you'll make a mistake.

Full Backup

A full backup is a backup that includes the entire database, in such a fashion that the whole database can be recovered or re-created in a consistent state. (This is different from a flawed backup that might include all of the database's datafiles, but which might be unusable for purposes of recovery – such a backup, while distressingly common, would be unworthy of the name 'full backup'.)

Note that all cold backups, when properly performed, are full backups. Many hot backups are also full backups.

The only alternative to a full backup is an incremental backup, and all incremental backups are hot.

Benefits of Full Backup

Full backups are vital. An incremental backup is only useful when taken together with its preceding full backup, whereas a full backup is useful on its own. Full backups are easier to use during recovery, compared to incremental backups. And full backups are comparatively easy to create.

Drawbacks of Full Backup

Full backups take longer to create than incremental backups, and generally occupy as much space as the database itself. (However, a full backup with RMAN can be smaller than the database, as the RMAN backup does not copy empty blocks. See the section on RMAN for more.) For a large database that undergoes little change, a full backup will keep copying and re-copying data that is no different from one full backup to the next.

Incremental Backup

An incremental backup is a backup that consists only of the differences (i.e. the 'net changes') between the database as it exists currently, and the database as it existed at the time of the last full backup. Incremental backups require the use of RMAN.

For example, consider a database containing an address book of 10,000 entries. Suppose a full backup of the database is taken on Sunday, and that one new address is added on Monday. On Tuesday, if something were to go wrong, we could restore the backup from Sunday, but we would lose the one new address. If that is not an acceptable risk, then we could perform a full backup Monday after the new address is added, which means backing up 10,001 entries just to save the one new one.

It would be nice to be able to create a backup containing just the new stuff – in this case, a backup of the one new address. This backup, together with the backup from Sunday, could be used to re-create the database as it existed on Monday.

Experienced DBAs will notice that archived redo logs are in a sense a kind of incremental backup, since they can be used to bring an older backup copy of the database up to date. This is true, and that's why redo logs get archived. However, it can take a long time to apply archived redo logs and make a database current. It is generally faster to use an incremental backup for this purpose. In most cases of this sort of recovery, first a full backup will be used to restore the database, next one or more incremental backups will be used to make the database as current as possible, and lastly the archived redo logs (dating from after the most recent incremental backup) will be used to bring the database up to the present moment.

Benefits of Incremental Backup

Incremental backups are usually faster to create than full backups. Because of this, busy DBAs are more likely to make incremental backups rather than just wait for the next full backup; this provides an extra level of protection for the database.

Incremental backups perform much the same purpose as archived redo logs – they allow you to bring an old full backup more up to date. An incremental backup is usually one file, or a few files, rather than the dozens or perhaps hundreds of archived redo logs one might otherwise use. Consequently, the incremental backup is both faster to create than a full backup, it's faster to apply than are archived redo logs, and easier to manage than archived redo logs. Furthermore, it is possible that a particular data element might be changed multiple times over the course of a week, and to restore the database all we really need to know is what the most recent value is. An incremental backup will store this most recent value (and will apply it, once, during recovery). Archived redo logs, in contrast, will slavishly change and re-change the data element through all its intermediate values, wasting time and slowing down recovery.

It is probably possible to use incremental backups without archiving redo logs, but this is probably rare.

Drawbacks of Incremental Backup

Incremental backups are potentially difficult to perform, especially for less experienced DBAs, and the backup files thus created are only useful in combination with other previous backups. The DBA faces the task of coordinating these files, storing them, and pulling up the right ones at the right time. RMAN will automate most of these housekeeping tasks (storing and finding files) but the DBA must learn RMAN, itself a complicated tool.

Tools available for backup

Cold and hot backups can be performed using a couple of different tools. On Windows NT these include:

- Operating System COPY commands
- Oracle OCOPY utility
- Recovery Manager (RMAN) with the Bundled Legato LSM, and the recovery catalog
- Third Party Backup solutions

Using COPY

The Windows NT command COPY can be used to create a cold backup of a database. It cannot be used to make a hot backup. Attempting to perform a hot backup with COPY will usually result in an error message being generated as the COPY command fails – during a hot backup the database is running and thus the database files are locked by the Oracle database process, and COPY cannot work on a file that is so locked. COPY cannot write directly to most tape devices under Windows NT.

Using OCOPY

OCOPY is a special, Oracle-supplied version of COPY that works on locked files. Thus it can be used to make hot backups. (OCOPY can also be used to make cold backups, but COPY fills that need just as well.) When performing a hot backup, be sure to follow carefully all directions. It's very easy to make what you think is a good backup, but isn't. OCOPY cannot write directly to most tape devices under Windows NT.

COPY-like commands on other operating systems

If you work on an operating system other than Windows NT, such as Unix, you will probably be able to use a single command (such as *dd*, *cp*, *tar*, or *cpio*) to perform both hot and cold backups. This can be convenient, but makes it doubly important that you know what you are doing. You could create what looks like a valid cold backup, but if the database was running, you actually created a useless backup from which you cannot recover.

Using RMAN and Legato

The newest backup solution from Oracle is the Server Managed Backup and Recovery Manager, usually abbreviated Recovery Manager or RMAN. RMAN uses a Recovery Catalog to help it automate most of the steps in an online backup and in recovery. RMAN also supports incremental backups, and a single Recovery Catalog can support the backup and recovery of dozens of different Oracle databases all around a network.

Using third party backup solutions

One of the most common third-party backup solutions for Oracle on Windows NT is ARCserve. Such products generally act like the COPY command (i.e. can perform a cold backup) and are able to write directly to a tape device. ARCserve can also act like OCOPY (i.e. can perform a hot backup) and is able to write directly to a tape device.

Backup and recovery speed

When you set up your backup and recovery processes and procedures, you will of course want to be confident that you can finish your regular backups quickly, and recover the database whenever disaster strikes. But you probably won't know for sure. Most DBAs grope only slowly towards an acceptable solution, and usually have to suffer through some disasters before they learn what works, what doesn't, and why.

The key to avoiding the suffer-through-disasters stage is to start with a Service Level Agreement for database uptime, backup windows, and acceptable data loss. Next, compare these needs with some basic metrics from your system:

- Size of data to be backed up
- Amount of time available for backup
- Maximum amount of acceptable downtime in the event of a restore
- Maximum amount of potential data loss in the event of a system failure
- How quickly archived redo logs can be applied
- How many redo logs are filled between backups
- Throughput (theoretical and real) of backup devices
- Latency of backup devices (especially tape devices)

We'll consider each of these in turn.

Backup speed

Backup speed is pretty easy to estimate, since you can control just exactly what you're backing up.

Speed of full backups

You can estimate your full backup time with two variables, and estimate the acceptability of your full backup plan with three variables.

Your backup time should be a function of the real throughput of your backup device, together with the size of your database. If your device can back up 4 MB per minute and your database is 40 MB, then backup time should be 10 minutes. (Note that the size of the other things you will back up, i.e. the control files and INIT.ORA file, are small enough to be ignored, though you may want to add them in for completeness.)

If your window for a full backup is 10 minutes or more, than you should be okay. If you are allowed 10 minutes of down time for backups, you can make a cold backup; if not, you'll have to make a hot backup.

That's the easy part.

Speed of incremental backups

Your backup time may be harder to estimate if you are performing incremental backups. For example, a very large database with a very small number of net changes would generate a very small incremental backup, but it could take the backup process a long time to look through the database to find the changes. In such a case, the limiting factor on backup performance is not the throughput of the backup device, but the speed with which the incremental changes can be found and placed into the backup set.

A rough estimate of the maximum backup time in such a case could be made as follows: find out how long it takes the backup process to scan through the database looking for changes (this will generally be a function of database size). This is time T1. Then find out how large the set of net changes is, and estimate how long it would take your backup device to accept that much data. This is time T2. Your incremental backup should take longer than (T1) but no more than (T1 + T2).

You can make T1 smaller by using multiple parallel backup processes to scan for changes. You'll have to also have multiple backup devices that can be written to in parallel, and the resulting files (tapes) will all have to be available during recovery.

You can reduce backup time by not backing up the entire database – either by doing incremental backups, or by excluding TEMPORARY tablespaces.

Recovery Speed

Overall recovery speed is a lot harder to estimate, largely because it's by nature unpredictable – you won't know in advance what is going to fail, how badly it will fail, how long it will take you to figure out what failed, and how long it will take you to replace broken hardware and start recovery.

Fortunately, many of these unknowns can be guessed, estimated, or controlled for. The important thing is to identify them all in advance and start exerting some control over them.

The overall equation for recovery time is: Time to identify problem (T*id*) + time to replace or fix hardware (T*hw*) + time to bring operating system back on line (T*os*) + time to restore files (T*restore*) + time to recover the database (T*recover*).

Controlling unknowns

Each unknown factor in overall recovery time is considered in turn.

T*id*

Time to identify the problem – an often overlooked factor, and one a panicky DBA or system administrator is tempted to rush through. Do not seize upon the first explanation for your problem that presents itself. If you save five minutes by identifying the wrong problem, you can easily spend hours trying to fix it, and in the process you may damage the database further. This is a surprisingly common problem, and Oracle Support has estimated that something like 80% of total elapsed database down time (of the unplanned and unwanted sort) is caused by a DBA who screws up a normal database recovery, does the wrong thing, and makes the matter worse.

The only reliable way to minimize T*id* is to practice various recovery scenarios with a test database on hardware similar to the production hardware, and for the DBA to be in close communication with the server machine's system administrator. It is very rare for the Oracle ORA-xxxxx messages to provide enough information for correct diagnosis of the underlying problem; the system administrator will certainly have to perform or confirm the diagnosis, and will have to be the one to fix the hardware (T*hw*) and bring the machine back up (T*os*).

T*hw*

One of the biggest unknowns during recovery involves replacing broken hardware. Many smaller shops have no plan in place for this inevitable occurrence. A good solution for important production machines is to have a support contract with the hardware manufacturer with a guaranteed response time for replacing failed hardware. With such a contract, you will have a maximum contracted time for T*hw*, and after the first incident you will have some idea of how much you can trust that contracted time.

In any event, it's much easier psychologically on the DBA, the system administrator, and management if there is someone specific to call for help, as opposed to having one of the staff running around to three or four computer shops looking for a replacement PCI Fast-Wide SCSI controller of a certain make and model, and trying to make an almost-identical replacement work with slightly different device drivers, all under deadline.

T*os*

A separate step involves bringing the operating system back up. Included in this are such tasks as formatting and mounting new or replacement disk drives, applying software patches (if the crash was caused by a software bug, including a database bug), and otherwise making the machine ready for the next step. By the end of T*os*, the machine is ready to support the restoration of database backup files.

To minimize T*os*, practice various recovery scenarios with a test database on hardware similar to the production hardware.

T*restore*

Once the server hardware and operating system are back on line, we can identify which database files are damaged and need to be restored from backup. The speed of restore is a function of how many datafiles are damaged, how large they are, how quickly they can be found on the backup media (latency), and the throughput of the backup device.

To minimize T*restore*, use multiple high-throughput backup devices in parallel. Also, don't back up TEMPORARY tablespaces if it would be faster to re-create them rather than restore them.

T*recover*

After damaged database files have been replaced with undamaged but out-of-date copies, these copies must be brought up-to-date. This step is called recovery. The speed of recovery is a function of how many datafiles are out of date, how many changes need to be applied to those datafiles before they are up to date, and how quickly the system is able to apply those changes.

To minimize T*recover*, perform frequent full and incremental backups.

Speed of recovery from cold backup

When recovering from a cold backup (and assuming the database is in NOARCHIVELOG mode), even if only a single datafile is damaged, all datafiles from the cold backup must be restored. So the overall time to recover from a lost datafile using a cold backup is a function of the size of the database and the throughput of the backup device.

Furthermore, the database cannot be made any more up to date than the time of the cold backup. Any data entered since that time is lost.

Speed of recovery from hot backup

When recovering from a hot backup, only damaged datafiles need to be replaced, and only they need to be brought up to date.

The time required to recover from a hot backup is a function of the number of damaged datafiles, their size, the number of changes that need to be applied to make the restored datafiles current, and how quickly those changes can be applied.

To speed recovery, perform frequent backups. The more up to date the restored file is, the fewer changes need to be applied to it to make it current, and the quicker it can be brought back on line.

Backup and restore scenarios

Following are some performance numbers obtained using a Compaq Proliant 5500 server, configured as follows:

System:

Proliant 5500 Server

* 4 400Mhz/512K processors
* 4GB Memory
* System BIOS P12 9/18/1998
* NT 4.0 Enterprise, SP3
* Oracle 8.1.4 , Legato Networker local backup client
* 2 Smart 3200 Array Controllers w 64MB cache
* 1 Smart 2/DH with 16MB cache
* 2 Ultra-Wide SCSI controllers (Compaq)
* 4 DLT 7000 tape drives (2 per controller)
* 5 F1 Storage cabinets
* 37 4.3 GB Pluggable FW drives
* 2 9.1 GB Pluggable drives

Disks were Configured as 9 logical drives

0	OS	RAID1	4 GB	2 spindles	smart 2
1	ARCH logs	RAID1	9 GB	2 spindles	
2	Bckup	RAID0	28 GB	7 spindles	smart 2
3	RBS	RAID1	8 GB	4 spindles	
4	TMP	RAID1	8 GB	4 spindles	
5	INDX	RAID1	12 GB	6 spindles	
6	DATA	RAID1	16 GB	8 spindles	
7	SYS	RAID1	4 GB	2 spindles	
8	LOGS	RAID1	8 GB	4 spindles	

We show how the raw performance of the components can be used to predict the real world behavior of the database backup and the expected time to recover.

About the Test Database

We used a 17 GB (total size) database built using the Benchmark Factory OTLP benchmark generation tool. The 'scale' of the database was 100; its datafiles totaled 17 GB in size, consisting of 10 GB of stored data (including indexes) and 7 GB in a combination of temporary space and pre-allocated growth space[2].

How We Measured Backup and Restore Time

Most of our backup and restore commands were controlled with batch files or command scripts, and we embedded a call to the 'TIME' command at the beginning and end of each file. Comparing these start and stop times gave us our elapsed time for the operation performed. For an example, see the sample RMAN command scripts in an appendix of this paper.

For timing the cold backup to tape using the Legato Networker client, we watched the diagnostic window of the Legato Networker server, which displayed the beginning and ending times of our backup commands.

[2] The space was pre-allocated mostly to the tablespaces, and to a much lesser extent to the individual tables and indexes. This amount of free space is not unusual, especially in data warehouses or other databases performing large scale queries, or in OLTP systems where space has been pre-allocated to minimize DBA labor.

For tracking and eliminating the tape latency times (especially during recovery), we watched the diagnostic window of the Legato Networker server, which displayed the time when a backup set was requested by the client and the time when the tape actually began to provide the backup set. The difference between these times was the tape latency for that request.

Offline, NOARCHIVELOG mode, 17 GB Database

Backup to Disk

We backed up the datafiles using the NT OS command 'copy', as shown by the sample NT command file in Appendix B. In our primary test, we used the NT 'start' command to have all the copy commands run simultaneously and in parallel, in order to maximize throughput and finish the backup as quickly as the hardware would allow. (In another test of cold backup using the 'copy' command, we let the copy commands execute serially.) For examples of how to use the 'copy' and 'start' commands for cold backup, see the sample NT backup script in an appendix of this paper.

Given the size of the database (17 GB or 17,408 MB) and expected real throughput of our disks and controllers (40% of the maximum throughput of 40 MB/sec, or about 16 MB/sec), we predict a backup time of roughly 1088 seconds or 18:08.

The actual backup time was 18:30, and the actual throughput was about 15.7 MB/sec. That is within 2% of our projected time. We based the projection on an expectation of throughput coming in at about 40% of the theoretical maximum, and based the 40% number on our experience with this hardware and with similar setups.

The key to projecting accurate backup times is to know how your hardware will perform, and to run the backup in a way that uses the hardware as efficiently as possible. We believe the NT 'copy' command is quite efficient.

Recovery from Disk Backup

We restored up the datafiles using the OS command 'copy'. We used the 'start' command to have all the copy commands run simultaneously and in parallel, in order to maximize throughput and finish the restore as quickly as the hardware would allow.

Given the sizes and throughputs above, we initially assumed a restore time equal to the backup time of roughly 1,088 seconds or 18:08.

The actual restore time was 28:04. This is 55% longer than initially predicted. The actual throughput was about 10.3 MB/sec. As it turns out, our backup disk device and controller was different from our database disk device and controller, their RAID levels were different, and the speed of copying files between them was asymmetric – i.e. copying in one direction was faster than copying back the other way. This does illustrate the importance of testing actual hardware to obtain real world results; if system users had been promised a 20-minute restore time using the above configuration, they would have been disappointed.

The differences in the disk RAID levels (RAID 0 vs. RAID 5) accounted for about 30% of this (RAID 5 imposes a roughly 30% write penalty compared to RAID 0), and differences in the hardware accounted for the other 25%.

Observations on Tape Drives and Latency

There is one thing that makes backup to tape different from backup to disk: disks can usually be written to immediately; tapes sometimes have to seek the unused part of the tape before they can be written to. In our tests this delay was never more than 2:44, i.e. less than three minutes. This was true of both backup and restore. Another source of latency can be seen when a person must manually find and mount a needed tape; unless computer operators are alert and responsive, this can introduce lengthy or even disastrous delays.

Cold Backup to Tape

Windows NT does not allow one to COPY directly to tape[3]. Therefore, we used the Legato Networker client software to perform a cold backup to tape. This consisted of opening the Networker GUI client, clicking on the datafiles to be backed up, and creating a backup job. We then saved the resulting backup job and ran it.

For this cold backup to tape, we predicted the backup time – based on the expected throughput of a single DLT tape drive of about 4 MB/sec (half the 8 MB/sec theoretical throughput) and the size of the database – would be about 4,352 seconds or about 72:32 (i.e. 1:12:32).

Actual backup time was 72:03.

(For the sake of completeness, we should mention another option for cold backups to tape. RMAN can be used to perform cold backups to tape, but they are awkward to set up and odd [the database must be started but not mounted or opened], and the recovery steps are quite arcane; RMAN cold backups to tape are not recommended for inexperienced RMAN users. Because of the general awkwardness of using RMAN in this fashion, we do not suggest you use it, and we did not test its performance.)

Online, ARCHIVELOG mode, 17 GB Database

The most common *hot* backup method for Oracle on Windows NT is to use the OCOPY command to back up datafiles. We tested this two ways, one with the database idle and the other with the database moderately busy.

OCOPY serial hot backup to disk, database idle

We backed up the datafiles using the NT OS command 'ocopy'. The 'ocopy' command is provided by Oracle and is much like 'copy' but allows us to read files that are active. We used the NT 'start' command to have all the copy commands run simultaneously and in parallel, in order to maximize throughput and finish the backup as quickly as the hardware would allow.

Given the size of the database (17 GB or 17,408 MB) and expected real throughput of our disks and controllers (40% of the maximum throughput of 40 MB/sec, or about 16 MB/sec), we predicted a backup time of roughly 1,088 seconds or 18:08. The additional overhead of the ALTER TABLESPACE commands should not add much to the total time. However, we didn't run the 'ocopy' commands in parallel (as we did for cold backup) so we may not be able to fully saturate the disk subsystems; therefore a longer backup time was considered likely.

The actual backup time was 33:06, and the actual throughput was just under 8.8 MB/sec. This tells us that the real throughput of a series of 'ocopy' commands was 22% of the theoretical throughput, or 55% of the maximum real throughput. The database was idle during the backup, and thus did not compete with the backup for disk resources.

OCOPY serial hot backup to disk, database active

In the next scenario, we backed up exactly as before, but with the database being lightly used.

To simulate the load of users during an off-peak period, we ran a TPC-C workload at about 1,020 transactions per minute, which was 20% of the maximum workload the database had shown it could handle.

Given the size of the database (17 GB or 17,408 MB) and the performance of serial 'ocopy' from the idle database backup (22% of the maximum throughput of 40 MB/sec, or about 8.8 MB/sec), we predict a backup time of at least 1,980 seconds or 33:00. However, the load of the database work should cause some significant slowdown in the backup process. Since the database activity is competing with the backup activity for access to the same disks, the backup might take as much as 66:00.

The actual backup time was 55:02, and the actual throughput was about 5.3 MB/sec.

Recovery from Disk Backup

We restored the datafiles using the OS command 'copy'. We used the 'start' command to have all the copy commands run simultaneously and in parallel, in order to maximize throughput and finish the restore as quickly as the hardware would allow.

[3] This is a standard limitation of NT. Some products may exist that bypass this limit, but they are exceptions to this general rule.

Given the sizes and throughputs above, and using our knowledge of the asymmetric behavior of the COPY command between our disk systems (discovered during cold backup to disk, above), we predicted a restore time of roughly 28:00.

The actual restore time was 28:33. The actual throughput was about 10.3 MB/sec. (This does not include the roll-forward recovery time to bring the database back up, which was a function of the total amount of work that the database had done since the beginning of the hot backup.)

RMAN and Recovery Catalog, 17 GB Database

Hot backups include full database backups, incremental database backups, and backups of archived logs. We will consider each of these in turn.

Hot Backup to Tape, idle database

To predict overall performance using RMAN was difficult because RMAN scans through the database datafiles and only backs up data blocks that have been used. This means RMAN will spend some small amount of time looking at blank (pre-allocated but empty) areas of the database but will exclude these data blocks from the backup set. We expected this to reduce backup time by effectively reducing the backup set from 17 GB of raw datafiles to 10 GB (10,240 MB) of 'interesting' data. (As explained in the section 'About the Test Database', our 17 GB total database space breaks down into 10 GB of data and 7 GB of empty space used for temporary segments and table growth. Since RMAN only backs up actual data, it was reasonable to suppose RMAN would back up about 10 GB rather than 17 GB of data.)

It was very easy with RMAN to use our four DLT tape drives in parallel. RMAN allows multiple tape 'channels' to be allocated at the beginning of a backup or restore session, which RMAN will use as it chooses. Our cold backup procedure did not allow us to address multiple tape devices at once. Using four tape drives in parallel increased the expected effective throughput of the tape devices to 16 MB/sec. This gave us a predicted backup time of at least 10:40 (if all operations were perfectly parallel) and probably no more than twice that, or 21:20 (if RMAN used the tapes somewhat asymmetrically). We carefully tracked tape latency and eliminated it from our measurements.)

Actual backup time was 15:35. Clearly RMAN was not feeding data to the tapes quite as quickly as the tapes could take it. Some of that extra time probably came from scanning through the datafiles; the rest probably came from RMAN's own internal bookkeeping processes. Additionally, recall that the basic unit of RMAN backup is the backup set, and the smallest a backup set can be is one datafile per backup set. (When we used a single tape for serial backup, we created a single backup set containing all datafiles.) Since we had a relatively small number (9) of datafiles which were not of uniform size, RMAN wasn't able to be perfectly parallel in its use of the available tape drives. Users who wish to increase their RMAN parallelism should use more datafiles of a smaller, uniform size. Typical datafile sizes are ½ GB, 1 GB, and 2 GB. We used a mix of these sizes.

Hot Backup to Tape, active database

For our hot backup, we used Oracle's RMAN together with Legato Networker.

We predicted the backup time would be at least 10:40 and probably no more than twice that, or 21:20. (We carefully tracked tape latency and eliminated it from our measurements.)

Actual backup time was 15:35. Clearly RMAN was not feeding data to the tapes quite as quickly as the tapes could take it. Some of that extra time probably came from scanning through the datafiles; the rest probably came from RMAN's own internal bookkeeping processes.

RMAN allows cold backups to tape, but they are awkward to set up and odd (the database must be started but not mounted or opened), making use of the control file to find datafiles, and the recovery steps are quite arcane; RMAN cold backups to tape are not recommended for inexperienced RMAN users.

Incremental Backup to Tape

We used three increment levels: 0, 1, and 2. To create an incremental backup, use a line like the following in your RMAN control file:

```
      backup incremental level 1
```

where Level 0 is the same as a full backup.

We performed a Level 0 backup in 0:15:35. We then ran the database for a while, until 100 MB of redo logs had been generated. Then we ran a Level 1 backup in 0:09:08. Then we ran the database some more, until 8 MB of redo logs had been generated. Then we ran a Level 2 backup in 0:08:35.

RMAN was able to scan through our entire 17 GB database, looking for changed blocks, in about eight minutes, for a phenomenally fast scan rate of 36 MB/sec. Additional time was needed for the backup, depending on how many changes the incremental backup found, with each additional 92 MB of changes needing roughly 33 additional seconds, or about 2.8 MB/sec.

Backup of Archived Logs to Tape

Using the appropriate RMAN command, back up all archived logs from the archive destination to a single tape, and delete them from the archive destination. We backed up 493 MB of logs in 0:01:50, or about 4.5 MB/sec. This is actually a bit faster than some of our other single-tape operations, and was quite efficient.

Database Point in Time Recovery (DBPITR) from Tape Backup

An advanced recovery technique is to recover the database to a point in time other than the most recent possible moment.

For example, suppose you have a midnight cold backup and an 8 a.m. hot backup. Furthermore, suppose a user error caused vital data to be deleted from a table at 10:04 a.m., and you found out at 10:30. You could shut down and perform a normal restore from the 8 a.m. hot backup, whereby the normal roll-forward recover brings the database back to… 10:30. Or you could shut down and restore from cold backup, bringing the database back to midnight.

Your real goal is to restore the database from the 8 a.m. hot backup and do part of the normal roll-forward recovery, but stop that recovery just before the 10:04 a.m. mistake. The way to do this is with 'Database Point in Time Recovery' (DBPITR).

This is done much like a normal restore, except that the RMAN command

```
recover database;
```

is changed to be

```
recover database until time '1999-03-05:11:33:00';
```

See the Appendix E for an example RMAN script that performs DBPITR.

Recovery from RMAN Tape Backup

We made no prediction of the total recovery time because RMAN performs both the Restore and the Recover steps, and because we didn't know if RMAN would ask for the data from the tapes in the same order that they'd been stored there. (The tape latency during this activity was beyond our power to measure or eliminate.)

Actual recovery time using a single RMAN tape channel (controlling all four tape drives) was 44:43. Actual recover time using four RMAN tape channels across the four drives was 28:04. Using the tape drives in parallel took 37% less time than serially.

By comparison, using RMAN to back up to disk, parallel operations generally took 5-50% less time than the same command performed serially.

The recovery steps were simple:

1. Decide on what failure to create – in this case, that a datafile would become damaged. We simulated this by deleting the file.
2. Create an RMAN recovery script that will fix the problem to be created.
3. Back up the database using cold backup to disk. This gives us a fallback point if recovery should fail.
4. Back up the database using the RMAN hot backup to tape backup script.
5. Cause the failure.
6. Run the RMAN recovery script created in step #2.
7. If recovery from hot backup fails, recover from cold backup.

For more on setting up RMAN, consult the Oracle documentation.

Here is a sample RMAN recovery script:

```
#file: restore01.rcv
#
```

```
run {
host 'time < null ' ;  #show current date and time
# allocate channels for parallel tape access
 allocate channel t1 type 'sbt_tape' ;
 allocate channel t2 type 'sbt_tape' ;
 allocate channel t3 type 'sbt_tape' ;
 allocate channel t4 type 'sbt_tape' ;
# allocate channels for serial disk access
 allocate channel d2 type disk ;

 restore database ;

 release channel t1 ;
 release channel t2 ;
 release channel t3 ;
 release channel t4 ;
 release channel d2 ;

host 'time < null ' ;
}
```

Description of Hardware and Software used in testing

Hardware

- Compaq Proliant 5500, 4way 400XEON, 3GB memory
  - -Smart 2/P array controllers
  - -Smart 2/DH array controllers
  - -M1 Disk cabinets
  - -4.3GB fastwide and ultrawide plug drives (with breakdown of how they are cut up in ACU)
  - -35/70 DLT tapes
  - -Compaq UW SCSI cards

- Compaq Deskpro 2000 P180, for client load

Software

- Windows NT 4.0 w/ SP3
- Oracle 8.1.4
- Legato LSM 2.1.1
- Benchmark factory 97

Test Types

The tests were devised to show how varying different dimensions of the database would affect the backup time. The tests varied these dimensions:

- Dataset size: 4.1 GB database vs. 17 GB database, and partial vs. whole database backups
- Hot vs. Cold backup
- Database idle vs. moderately active
- Parallel vs. Serial movement of files to/from backup destination

Test Scenarios

The details in the scenarios include

- Dataset size
- Backup time
- Restore time
- Scripts used in the scenario (to be found in Appendix)
- Analysis of how the testing went

Test ID	**Backup Size**	**Time**	**Script**	**Analysis**
BU01	4.1 GB	0:06:58	mk_cold_bkup.sql	'copy' in parallel +/- 30 seconds
BU02	3.1 GB	0:07:33	mk_hot_bkup.sql	'ocopy' in serial +/- 150 seconds, idle database; excludes the TEMPORARY tablespace which can only contain TEMPORARY objects; expect to re-create it rather than recover it in this scenario.
BU03	3.1 GB	0:08:58	mk_hot_bkup.sql	Busy database: tps dropped from ~78 to 58 when the USERDATA tablespace was placed in backup mode.
BU04	435 MB	0:02:52	tc10.rcv	RMAN serial to disk, idle database
BU05	456 MB	0:02:56	tc10.rcv	RMAN serial to disk, busy database
BU06	466 MB	0:02:14	tc08.rcv	RMAN parallel to disk, idle database
BU07	470 MB	0:02:16	tc08.rcv	RMAN parallel to disk, busy database
BU08	478 MB	0:05:25	tc11.rcv	RMAN serial to tape, idle database
BU09	515 MB	0:06:22	tc11.rcv	RMAN serial to tape, busy database
BU10	530 MB	0:04:12	tc09.rcv	RMAN parallel to tape, idle database
BU11	545 MB	0:04:00	tc09.rcv	RMAN parallel to tape, busy database
Using larger database from this point onward				
BU12	17 GB	0:18:30	mk_cold_bkup.sql	cold parallel backup to disk
BU13.0	~10 GB	0:15:35	tc09_0.rcv	hot parallel backup to tape, idle db, increment=0
BU13.1	~10 GB	0:09:08	tc09_1.rcv	hot parallel backup to tape, idle db, increment=1 after 100 MB of logged changes
BU13.2	~10 GB	0:08:35	tc09_2.rcv	hot parallel backup to tape, idle db, increment=2 after 8 MB of logged changes
BU14.0i	~10 GB	0:16:16	tc09_0.rcv	hot parallel backup to tape, idle db, increment=0
BU14.0b	~10 GB	0:17:20	tc09_0.rcv	hot parallel backup to tape, busy db, increment=0
BU14.1b	~10 GB	0:09:54	tc09_1.rcv	hot parallel backup to tape, busy db, increment=1
BU14.2i	~10 GB	0:08:04	tc09_2.rcv	hot parallel backup to tape, idle db, increment=2
BU14.arch	~10 GB	0:02:46	tc09_arch.rcv	backed up 102 logs, 493 MB total size
Restore01	~2 GB	0:08:46	restore01.rcv	delete one datafile; restore automatically using RMAN and recovery catalog; parallel from tape
Restore04	~10 GB	0:44:43	restore04.rcv	delete all datafiles; restore automatically using RMAN and recovery catalog; serial from tape
Restore06	~10 GB	0:28:04	restore06.rcv	delete all datafiles; restore automatically using RMAN and recovery catalog; parallel from tape
Restore08	~10 GB	0:27:39	restore08.rcv	delete all datafiles; restore to point in time (excludes recovery), parallel from tape
Restore08a	~423 MB	0:12:30	restore08a.rcv	roll-forward portion of database point-in-time recovery, from disk archive destination, applying 86 archive logs (423 MB)

A final note on completeness and accuracy

This paper is by no means complete, exhaustive, or all-encompassing. It is not intended to be. Knowledgeable readers will note some statements that, upon close examination, are not entirely true – for example, where I say that an incomplete cold backup is useless. While a sufficiently experienced DBA could use a partial cold backup, together with archived redo logs, to perform certain kinds of recovery in certain circumstances, it is also true that an incomplete cold backup is useless in the hands of a DBA of more average experience, and the circumstances under which one would want to intentionally create an incomplete cold backup are rare indeed. By keeping the focus on what one might reasonably expect of an average DBA and an average database, I hope I have not offended the more expert among my readers.

Comments and corrections are welcome. The author may be reached at tbcox@att.net. An up-to-date copy of this paper can usually be found at http://www.geocities.com/tbcox23.

Appendix A:
Create cold backup and restore on Windows NT

```
REM  Filename: mk_cold_bkup.sql
REM  Purpose:  creates a cold backup script and a cold recovery script
REM  Author:   Thomas B. Cox <tbcox@att.net>, TrueNorth Consulting
REM  Version:  1.1
REM  Last Mod: 06-Apr-1999
REM
REM  Added in 1.1:  comments and header
REM
REM  Known bugs:
REM             does not back up INIT.ORA file
REM

set scan off
set serveroutput on  size 200000
set echo off
set verify off
set feedback off

spool _cold_bkup.sql

DECLARE
   ---------------------------------------------
   --  OSD constants
   ---------------------------------------------
   --
   -- the OS command to copy a cold datafile to the destination
   vc_host_copy_cmd CONSTANT VARCHAR2(2000) := 'start cmd /c copy' ;
   --
   -- backup file destination; NO TRAILING DELIMETER!
   vc_bkup_dest CONSTANT VARCHAR2(2000) := 'D:\oradata\fred\backup' ;
   --

   ---------------------------------------------
   --  Cursors
   ---------------------------------------------
   --
   -- gets each tablespace except those flagged as TEMPORARY
   CURSOR ts_cur
      IS
      SELECT ts.tablespace_name AS ts_nm
        FROM dba_tablespaces ts ;     --  WHERE ts.contents != 'TEMPORARY' ;

   -- gets each datafile for a given tablespace
   CURSOR ts_df_cur ( ts_nm_in IN VARCHAR2 )
      IS
      SELECT df.file_name as ts_df_nm
        FROM dba_data_files df
       WHERE df.tablespace_name = ts_nm_in ;
   -- gets all of the control files
   CURSOR cf_cur
      IS
      SELECT cf.name as cf_nm
        FROM v$controlfile cf ;
BEGIN
   dbms_output.put_line( 'connect internal/manager ;' ) ;
   dbms_output.put_line( 'shutdown immediate ;' ) ;

   -- get control files
   FOR cf_rec IN cf_cur LOOP
      dbms_output.put_line( 'host ' || vc_host_copy_cmd
         || ' ' || cf_rec.cf_nm || ' ' || vc_bkup_dest );
   END LOOP ;

   -- get tablespaces and datafiles
   FOR ts_rec IN ts_cur LOOP
      dbms_output.put_line( 'REM alter tablespace ' || ts_rec.ts_nm || ' begin backup ; ' );
      FOR df_rec IN ts_df_cur ( ts_rec.ts_nm ) LOOP
         dbms_output.put_line( 'host ' || vc_host_copy_cmd
            || ' ' || df_rec.ts_df_nm || ' ' || vc_bkup_dest );
      END LOOP ;

      dbms_output.put_line( 'REM alter tablespace ' || ts_rec.ts_nm || ' end backup ; ' );
   END LOOP ;
   -- dbms_output.put_line( 'startup ' ) ;
   -- dbms_output.put_line( 'exit ' ) ;
END ;
/
spool off

prompt
prompt File '_cold_bkup.sql' created.
prompt

/**************
*
*  END OF FIRST SCRIPT
*
**************/



/**************
*
*  BEGINNING OF SECOND SCRIPT
*
**************/

spool _cold_restore.bat

DECLARE
   ---------------------------------------------
   --  Variables
   ---------------------------------------------
   -- bkup file name
   -- v_bkup_df_nm

   ---------------------------------------------
   --  OSD constants
   ---------------------------------------------
   --
   -- the 'host' keyword plus the OS command to copy a cold datafile to the destination
   vc_host_copy_cmd CONSTANT VARCHAR2(2000) := 'start cmd /c copy' ;
   --
   -- backup file destination; NO TRAILING DELIMETER!
   vc_bkup_dest CONSTANT VARCHAR2(2000) := 'D:\oradata\fred\backup' ;
   --
   -- Character that separates file name(s) from directory path
   vc_dir_sep CONSTANT VARCHAR2(1) := '\' ;
   --

   ---------------------------------------------
   --  Cursors
   ---------------------------------------------
   --
   -- gets each tablespace except those flagged as TEMPORARY
   -- MUST be the same cursor as that used above!
   CURSOR ts_cur
      IS
      SELECT ts.tablespace_name AS ts_nm
        FROM dba_tablespaces ts ;      -- WHERE ts.contents != 'TEMPORARY' ;

   -- gets each datafile for a given tablespace
   CURSOR ts_df_cur ( ts_nm_in IN VARCHAR2 , sep_in IN VARCHAR2 )
      IS
      SELECT df.file_name as ts_df_nm
           , substr ( file_name
                    , instr ( file_name , sep_in, -1, 1 ) + 1
                    , ( length ( file_name )
                      + - instr ( file_name , sep_in, -1, 1 ) )
                    ) as just_df_nm
        FROM dba_data_files df
       WHERE df.tablespace_name = ts_nm_in ;
   -- gets all of the control files
   CURSOR cf_cur ( sep_in IN VARCHAR2 )
      IS
      SELECT cf.name as cf_nm
           , substr ( name
                    , instr ( name , sep_in, -1, 1 ) + 1
                    , ( length ( name )
                      + - instr ( name , sep_in, -1, 1 ) )
                    ) as just_cf_nm
        FROM v$controlfile cf ;
BEGIN

   -- get control files
   FOR cf_rec IN cf_cur ( vc_dir_sep ) LOOP
      dbms_output.put_line ( vc_host_copy_cmd || ' '  || vc_bkup_dest
         || vc_dir_sep || cf_rec.just_cf_nm || ' ' || cf_rec.cf_nm );
   END LOOP ;

   -- get tablespaces and datafiles
   FOR ts_rec IN ts_cur LOOP

      dbms_output.put_line( 'REM  Start restoring tablespace ' || ts_rec.ts_nm || '.' );

      FOR df_rec IN ts_df_cur ( ts_rec.ts_nm , vc_dir_sep ) LOOP

         dbms_output.put_line ( vc_host_copy_cmd || ' '  || vc_bkup_dest
            || vc_dir_sep || df_rec.just_df_nm || ' ' || df_rec.ts_df_nm );

      END LOOP ;

      dbms_output.put_line( 'REM  Finished restoring tablespace ' || ts_rec.ts_nm || '.' );

   END LOOP ;

   dbms_output.put_line( 'REM  End of restore.  You may now start your database. ' ) ;

END ;
/
spool off

prompt
prompt File '_cold_restore.bat' created.
prompt
```

Appendix B: Sample Output of Cold Backup Creation Script

File _cold_bkup.sql

```
connect internal/manager ;
shutdown immediate ;
host start cmd /c copy C:\ORADATA\FRED\CTRL01.CTL D:\oradata\fred\backup
host start cmd /c copy E:\ORADATA\FRED\CTRL02.CTL D:\oradata\fred\backup
host start cmd /c copy F:\ORADATA\FRED\CTRL02.CTL D:\oradata\fred\backup
REM alter tablespace SYSTEM begin backup ;
host start cmd /c copy C:\ORADATA\FRED\SYST01.DBF D:\oradata\fred\backup
REM alter tablespace SYSTEM end backup ;
REM alter tablespace RBS begin backup ;
host start cmd /c copy E:\ORADATA\FRED\RBSG01.DBF D:\oradata\fred\backup
REM alter tablespace RBS end backup ;
REM alter tablespace USR begin backup ;
host start cmd /c copy C:\ORADATA\FRED\USER01.DBF D:\oradata\fred\backup
REM alter tablespace USR end backup ;
REM alter tablespace TEMPORARY begin backup ;
host start cmd /c copy C:\ORADATA\FRED\TEMP01.DBF D:\oradata\fred\backup
REM alter tablespace TEMPORARY end backup ;
REM alter tablespace INDX begin backup ;
host start cmd /c copy F:\ORADATA\FRED\INDX01.DBF D:\oradata\fred\backup
REM alter tablespace INDX end backup ;
REM alter tablespace DES2K begin backup ;
host start cmd /c copy E:\ORADATA\FRED\DS2K01.DBF D:\oradata\fred\backup
REM alter tablespace DES2K end backup ;
REM alter tablespace D2IDX begin backup ;
host start cmd /c copy F:\ORADATA\FRED\DIDX01.DBF D:\oradata\fred\backup
REM alter tablespace D2IDX end backup ;
```

File _cold_restore.bat

```
start cmd /c copy D:\oradata\fred\backup\CTRL01.CTL C:\ORADATA\FRED\CTRL01.CTL
start cmd /c copy D:\oradata\fred\backup\CTRL02.CTL E:\ORADATA\FRED\CTRL02.CTL
start cmd /c copy D:\oradata\fred\backup\CTRL02.CTL F:\ORADATA\FRED\CTRL02.CTL
REM  Start restoring tablespace SYSTEM.
start cmd /c copy D:\oradata\fred\backup\SYST01.DBF C:\ORADATA\FRED\SYST01.DBF
REM  Finished restoring tablespace SYSTEM.
REM  Start restoring tablespace RBS.
start cmd /c copy D:\oradata\fred\backup\RBSG01.DBF E:\ORADATA\FRED\RBSG01.DBF
REM  Finished restoring tablespace RBS.
REM  Start restoring tablespace USR.
start cmd /c copy D:\oradata\fred\backup\USER01.DBF C:\ORADATA\FRED\USER01.DBF
REM  Finished restoring tablespace USR.
REM  Start restoring tablespace TEMPORARY.
start cmd /c copy D:\oradata\fred\backup\TEMP01.DBF C:\ORADATA\FRED\TEMP01.DBF
REM  Finished restoring tablespace TEMPORARY.
REM  Start restoring tablespace INDX.
start cmd /c copy D:\oradata\fred\backup\INDX01.DBF F:\ORADATA\FRED\INDX01.DBF
REM  Finished restoring tablespace INDX.
REM  Start restoring tablespace DES2K.
start cmd /c copy D:\oradata\fred\backup\DS2K01.DBF E:\ORADATA\FRED\DS2K01.DBF
REM  Finished restoring tablespace DES2K.
REM  Start restoring tablespace D2IDX.
start cmd /c copy D:\oradata\fred\backup\DIDX01.DBF F:\ORADATA\FRED\DIDX01.DBF
REM  Finished restoring tablespace D2IDX.
REM  End of restore.  You may now start your database.
```

Appendix C:
Create hot backup and restore on Windows NT

```
REM  Filename: mk_hot_bkup.sql
REM  Purpose:  creates a hot backup script and a hot recovery script
REM  Author:   Thomas B. Cox <tbcox@att.net>, TrueNorth Consulting
REM  Version:  1.1
REM  Last Mod: 06-Apr-1999
REM
REM  Added in 1.1:  check for ARCHIVELOG mode
REM
REM  Known bugs:
REM             does not back up INIT.ORA file
REM

set serveroutput on size 200000
set echo off
set verify off
set feedback off

spool _hot_bkup.sql

DECLARE
   ---------------------------------------------
   --  OSD constants
   ---------------------------------------------
   --
   -- the OS command to copy a hot datafile to the destination
   vc_host_copy_cmd CONSTANT VARCHAR2(2000) := 'ocopy' ;
   --
   -- backup file destination; NO TRAILING DELIMETER!
   vc_bkup_dest CONSTANT VARCHAR2(2000) := 'U:\oradata\pd01\backup' ;
   --
   -- Character that separates file name(s) from directory path
   vc_dir_sep CONSTANT VARCHAR2(1) := '\' ;

   ---------------------------------------------
   --  Cursors
   ---------------------------------------------
   --
   -- gets each tablespace except those flagged as TEMPORARY
   CURSOR ts_cur
      IS
      SELECT ts.tablespace_name AS ts_nm
        FROM dba_tablespaces ts
       WHERE ts.contents != 'TEMPORARY' ;

   -- gets each datafile for a given tablespace
   CURSOR ts_df_cur ( ts_nm_in IN VARCHAR2 )
      IS
      SELECT df.file_name as ts_df_nm
        FROM dba_data_files df
       WHERE df.tablespace_name = ts_nm_in ;
   -- gets all of the control files
   CURSOR cf_cur
      IS
      SELECT cf.name as cf_nm
        FROM v$controlfile cf ;

   ---------------------------------------------
   --  Variables
   ---------------------------------------------
   --
   v_Log_Mode v$database.log_mode%TYPE ;


BEGIN

   -- if the database is not in ARCHIVELOG mode, halt w/ error
   select log_mode into v_Log_Mode from v$database ; -- always a 1-row table
   if v_Log_Mode != 'ARCHIVELOG' then RAISE ; end if ;

   -- begin creating the output file
   dbms_output.put_line ( 'set echo on' ) ;

   -- get control files
   FOR cf_rec IN cf_cur LOOP
      dbms_output.put_line( 'host ' || vc_host_copy_cmd
         || ' ' || cf_rec.cf_nm || ' ' || vc_bkup_dest );
   END LOOP ;

FOR ts_rec IN ts_cur LOOP
dbms_output.put_line( 'alter tablespace ' || ts_rec.ts_nm || ' begin backup ; ' );

   FOR df_rec IN ts_df_cur ( ts_rec.ts_nm ) LOOP
      dbms_output.put_line( 'host ' || vc_host_copy_cmd || ' ' || df_rec.ts_df_nm || ' ' ||
vc_bkup_dest );
   END LOOP ;

dbms_output.put_line( 'alter tablespace ' || ts_rec.ts_nm || ' end backup ; ' );
END LOOP ;
END ;
/
spool off

prompt
prompt File '_hot_bkup.sql' created.
prompt

/**************
*
*  END OF FIRST SCRIPT
*
**************/



/**************
*
*  BEGINNING OF SECOND SCRIPT
*
**************/

spool _hot_restore.bat

prompt
prompt    Do not use this file until you have tested it!
prompt

DECLARE
   --------------------------------------------
   --  Variables
   --------------------------------------------
   -- bkup file name
   -- v_bkup_df_nm

   --------------------------------------------
   --  OSD constants
   --------------------------------------------
   --
   -- the OS command to copy a cold datafile to the destination
   vc_host_copy_cmd CONSTANT VARCHAR2(2000) := 'copy' ;
   --
   -- backup file destination; NO TRAILING DELIMETER!
   vc_bkup_dest CONSTANT VARCHAR2(2000) := 'U:\oradata\pd01\backup' ;
   --
   -- Character that separates file name(s) from directory path
   vc_dir_sep CONSTANT VARCHAR2(1) := '\' ;

   --------------------------------------------
   --  Cursors
   ---------------------------------------------
   --
   -- gets each tablespace except those flagged as TEMPORARY
   -- MUST be the same cursor as that used above!
   CURSOR ts_cur
      IS
      SELECT ts.tablespace_name AS ts_nm
        FROM dba_tablespaces ts
       WHERE ts.contents != 'TEMPORARY' ;

   -- gets each datafile for a given tablespace
   CURSOR ts_df_cur ( ts_nm_in IN VARCHAR2 , sep_in IN VARCHAR2 )
      IS
      SELECT df.file_name as ts_df_nm
           , substr ( file_name
                    , instr ( file_name , sep_in, -1, 1 ) + 1
                    , ( length ( file_name )
                      + - instr ( file_name , sep_in, -1, 1 ) )
                    ) as just_df_nm
        FROM dba_data_files df
       WHERE df.tablespace_name = ts_nm_in ;
   -- gets all of the control files
   CURSOR cf_cur ( sep_in IN VARCHAR2 )
      IS
      SELECT cf.name as cf_nm
           , substr ( name
                    , instr ( name , sep_in, -1, 1 ) + 1
                    , ( length ( name )
                      + - instr ( name , sep_in, -1, 1 ) )
                    ) as just_cf_nm
        FROM v$controlfile cf ;
BEGIN

   -- get control files
   FOR cf_rec IN cf_cur ( vc_dir_sep ) LOOP
      dbms_output.put_line ( vc_host_copy_cmd || ' '  || vc_bkup_dest
         || vc_dir_sep || cf_rec.just_cf_nm || ' ' || cf_rec.cf_nm );
   END LOOP ;

   -- get tablespaces and datafiles
   FOR ts_rec IN ts_cur LOOP

      dbms_output.put_line( 'REM  Start restoring tablespace ' || ts_rec.ts_nm || '.' );

      FOR df_rec IN ts_df_cur ( ts_rec.ts_nm , vc_dir_sep ) LOOP

         dbms_output.put_line ( vc_host_copy_cmd || ' '  || vc_bkup_dest
            || vc_dir_sep || df_rec.just_df_nm || ' ' || df_rec.ts_df_nm );

      END LOOP ;

      dbms_output.put_line( 'REM  Finished restoring tablespace ' || ts_rec.ts_nm || '.' );

   END LOOP ;

   dbms_output.put_line( 'REM  End of restore.  You may now start your database and begin
recovery. ' ) ;

END ;
/
spool off

prompt
prompt File '_hot_restore.bat' created.
prompt
```

Appendix D: Sample Output of Hot Backup Creation Script

File _hot_bkup.sql

```
set echo on
host ocopy C:\ORADATA\FRED\CTRL01.CTL U:\oradata\pd01\backup
host ocopy E:\ORADATA\FRED\CTRL02.CTL U:\oradata\pd01\backup
host ocopy F:\ORADATA\FRED\CTRL02.CTL U:\oradata\pd01\backup
alter tablespace SYSTEM begin backup ;
host ocopy C:\ORADATA\FRED\SYST01.DBF U:\oradata\pd01\backup
alter tablespace SYSTEM end backup ;
alter tablespace RBS begin backup ;
host ocopy E:\ORADATA\FRED\RBSG01.DBF U:\oradata\pd01\backup
alter tablespace RBS end backup ;
alter tablespace USR begin backup ;
host ocopy C:\ORADATA\FRED\USER01.DBF U:\oradata\pd01\backup
alter tablespace USR end backup ;
alter tablespace INDX begin backup ;
host ocopy F:\ORADATA\FRED\INDX01.DBF U:\oradata\pd01\backup
alter tablespace INDX end backup ;
alter tablespace DES2K begin backup ;
host ocopy E:\ORADATA\FRED\DS2K01.DBF U:\oradata\pd01\backup
alter tablespace DES2K end backup ;
alter tablespace D2IDX begin backup ;
host ocopy F:\ORADATA\FRED\DIDX01.DBF U:\oradata\pd01\backup
alter tablespace D2IDX end backup ;
alter tablespace T_FOO1 begin backup ;
host ocopy T_FOO1 U:\oradata\pd01\backup
alter tablespace T_FOO1 end backup ;
```

File _hot_restore.bat

```
Do not use this file until you have tested it!

copy U:\oradata\pd01\backup\CTRL01.CTL C:\ORADATA\FRED\CTRL01.CTL
copy U:\oradata\pd01\backup\CTRL02.CTL E:\ORADATA\FRED\CTRL02.CTL
copy U:\oradata\pd01\backup\CTRL02.CTL F:\ORADATA\FRED\CTRL02.CTL
REM  Start restoring tablespace SYSTEM.
copy U:\oradata\pd01\backup\SYST01.DBF C:\ORADATA\FRED\SYST01.DBF
REM  Finished restoring tablespace SYSTEM.
REM  Start restoring tablespace RBS.
copy U:\oradata\pd01\backup\RBSG01.DBF E:\ORADATA\FRED\RBSG01.DBF
REM  Finished restoring tablespace RBS.
REM  Start restoring tablespace USR.
copy U:\oradata\pd01\backup\USER01.DBF C:\ORADATA\FRED\USER01.DBF
REM  Finished restoring tablespace USR.
REM  Start restoring tablespace INDX.
copy U:\oradata\pd01\backup\INDX01.DBF F:\ORADATA\FRED\INDX01.DBF
REM  Finished restoring tablespace INDX.
REM  Start restoring tablespace DES2K.
copy U:\oradata\pd01\backup\DS2K01.DBF E:\ORADATA\FRED\DS2K01.DBF
REM  Finished restoring tablespace DES2K.
REM  Start restoring tablespace D2IDX.
copy U:\oradata\pd01\backup\DIDX01.DBF F:\ORADATA\FRED\DIDX01.DBF
REM  Finished restoring tablespace D2IDX.
REM  Start restoring tablespace T_FOO1.
copy U:\oradata\pd01\backup\T_FOO1 T_FOO1
REM  Finished restoring tablespace T_FOO1.
REM  End of restore.  You may now start your database and begin recovery.
```

Appendix E: Selected RMAN scripts

File tc08.rcv

```
# database hot backup parallel to disk
#
run {
host 'time < null ' ;
 allocate channel d1 type disk
    format 'U:\oradata\pd01\backup\%d_%u' ;
allocate channel d2 type disk
    format 'U:\oradata\pd01\backup\%d_%u' ;
allocate channel d3 type disk
    format 'U:\oradata\pd01\backup\%d_%u' ;
allocate channel d4 type disk
    format 'U:\oradata\pd01\backup\%d_%u' ;
 backup full tag = full_19990218
  filesperset = 1 ( database ) ;
host ' time < null ' ;
}
```

File tc09.rcv

```
# database hot backup parallel to tape
#
run {
host 'time < null ' ;
 allocate channel t1 type 'sbt_tape' ;
 allocate channel t2 type 'sbt_tape' ;
 allocate channel t3 type 'sbt_tape' ;
 allocate channel t4 type 'sbt_tape' ;
 backup full tag="19990216_full"
 filesperset 1 format '%u' ( database ) ;
host 'time < null ' ;
}
```

File tc10.rcv

```
# database hot backup serially to disk
#
run {
host 'time < null ' ;
 allocate channel d1 type disk
    format 'U:\oradata\pd01\backup\%d_%u' ;
 backup full tag = full_19990218
  filesperset = 9 ( database ) ;
host ' time < null ' ;
}
```

File tc11.rcv

```
# database hot backup serially to tape
#
run {
host 'time < null ' ;
 allocate channel t1 type 'sbt_tape' ;
 backup full tag = full_19990218
  filesperset = 99 ( database ) ;
host ' time < null ' ;
}
```

File recover01.rcv

```
# recover database
```

```
#
run {
host 'time < null ' ;  # show current date and time
# allocate channels for parallel tape access
 allocate channel t1 type 'sbt_tape' ;
 allocate channel t2 type 'sbt_tape' ;
 allocate channel t3 type 'sbt_tape' ;
 allocate channel t4 type 'sbt_tape' ;
# allocate channels for serial disk access
 allocate channel d2 type disk ;

 restore database ;

 recover database ;

 release channel t1 ;
 release channel t2 ;
 release channel t3 ;
 release channel t4 ;
 release channel d2 ;

host 'time < null ' ;
}
```

File recover08.rcv

```
# recover from loss of all datafiles
# and restore to a point in time

run {
host 'time < null ' ;
 allocate channel t1 type 'sbt_tape' ;
 allocate channel t2 type 'sbt_tape' ;
 allocate channel t3 type 'sbt_tape' ;
 allocate channel t4 type 'sbt_tape' ;
 allocate channel d2 type disk ;

 sql "alter session set nls_language=american " ;
 sql 'alter session set nls_date_format="YYYY-MM-DD:HH24:MI:SS" ';

 restore database ;

 recover database until time '1999-03-05:11:33:00';

 release channel t1 ;
 release channel t2 ;
 release channel t3 ;
 release channel t4 ;
 release channel d2 ;

host 'time < null ' ;
}
```

File recover08.sql

```
# delete database and restore to a point in time
# invoke this file from server manager
connect internal/manager

set echo on

shutdown immediate ;

host del I:\ORADATA\PD01\SYST01.DBF

host del E:\ORADATA\PD01\RBSG01.DBF

host del H:\ORADATA\PD01\DATA01.DBF
host del H:\ORADATA\PD01\DATA02.DBF
host del H:\ORADATA\PD01\DATA03.DBF
host del H:\ORADATA\PD01\DATA04.DBF
host del H:\ORADATA\PD01\DATA05.DBF
host del H:\ORADATA\PD01\DATA06.DBF
host del H:\ORADATA\PD01\DATA07.DBF
host del H:\ORADATA\PD01\DATA08.DBF
host del H:\ORADATA\PD01\DATA09.DBF
host del H:\ORADATA\PD01\DATA10.DBF

host del G:\ORADATA\PD01\INDX01.DBF
host del G:\ORADATA\PD01\INDX02.DBF
host del G:\ORADATA\PD01\INDX03.DBF
host del G:\ORADATA\PD01\INDX04.DBF

host del F:\ORADATA\PD01\TEMP01.DBF
host del F:\ORADATA\PD01\TEMP02.DBF

startup nomount ;

host rman target internal/manager rcvcat rman/rman@pd02.world cmdfile restore08.rcv

shutdown immediate ;

startup ;
```

Appendix F:
Typical backup/recovery panic

Below is a posting from the internet newsgroup comp.databases.oracle.server; the author's name has been removed. It's a good example of the kinds of mistakes it's very easy to make during both backup and recovery, and how these mistakes can pile on top of each other, turning a potentially easy-to-fix situation into a disaster. (The author of thisposting, who gave permission for it to be included here, is to be commended for keeping clear notes of the problem and for having the courage to ask for help.)

> Let me describe my dilemma. An Oracle 7.1.16 DB began getting ORA-00376 (file can't be read at this time [/u05/oradata/dataJIT_03]) errors two days ago. When I found out about it, I looked at Oracle and in the v$datafiles table it showed the file status as AVAILABLE. I tried to alter the associated tablespace OFFLINE NORMAL to bring the file off-line and then on-line. I got an ORA-01191 (file already off-line - can't do a normal off-line). Then we altered the tablespace OFFLINE IMMEDIATE. The tablespace wouldn't come back on-line, and we got an ORA-01113 error (file needs media recovery [/u04/oradata/dataJIT_01.dbf]).
>
> Instead of applying the archivelog files, I shut down the db using SHUTDOWN NORMAL, and did an image backup of the DB. Just for laughs, we tried to start the DB again, and we got another ORA-00376 (file can't be read at this time [/u05/oradata/dataJIT_02.dbf]).
>
> Here's where we screwed up. We didn't shut down the DB while we were restoring from Monday night's image BU. That totally stepped on the DB files - including the control files. Someone realized the DB was up (duh), and tried to shut it down. It wouldn't shutdown until I executed a shutdown abort. When, we finished restoring the Monday night's BU, we started having control file version differences (ORA-00376 (how did that happen?)). When we finally got that ironed out by copying increasingly newer versions of one control file on top of the others, we got an ORA-01113 (system tablespace needs media recovery [/u01/oradata/sysJIT_01.dbf]).
>
> So, in the last 2 days, we've gotten messages that 3 different devices need media recovery. Did Oracle "blow its cookies," or do we have an obscure hardware (controller?) problem.
>
> Also, should we just restore the Wednesday morning BU, and apply the archive logs after having HP run diagnostics on the machine?

Appendix G:
Step-by-step RMAN standby database creation

This walks us step-by-step through the creation of a Standby database, using RMAN on two Solaris machines protected with 'ssh' (Secure Shell, a Unix security scheme using public key encryption).

1. Create the standby database
   - 1.1. make a full RMAN backup of the source or target database
     On 'mama' this can be done with the command 'do_rman_bkup.bat' located in the ~oracle/admin/aprd/backup directory.
   - 1.2. copy the resulting RMAN backup files to the standby machine
   - 1.3. run the 'make_duplicate.bat' script, which calls the 'tc_dup.rcv' file
   - 1.4. ignore the final error at the end of the RMAN session
     On 'phobos' the RMAN session will end with an error, saying that the "create controlfile" command failed because the database name "aaux" is different from the database name "aprd" in the init.ora file. This is normal – we don't want to create a control file anyway.
   - 1.5. make a standby control file and copy it to the standby machine
     On 'mama' connect with svrmgrl and issue "alter database create standby controlfile as 'file';" an then copy 'file' to the standby machine
   - 1.6. start the standby database using the standby controlfile from the previous step
     On 'phobos' connect with svrmgrl. Make sure database is started but not mounted, and issue 'select * from v$controlfile' – this will tell you where the database thinks its controlfile is or should be. Next, copy 'file' from the previous step to the filename you got from v$controlfile. Next, issue "alter database mount standby database".
2. Bring the standby database up to date (do this daily)
   - 2.1. copy all the archived redo logs to the standby machine
   - 2.2. from svrmgrl, issue 'recover standby database'
   - 2.3. accept prompts until you run out of logs, then enter 'cancel'
3. Maintain the standby database
   - 3.1. daily, copy all the new archived redo logs to the standby machine
   - 3.2. from svrmgrl, issue 'recover standby database'
   - 3.3. accept prompts until you run out of logs, then enter 'cancel'
4. Activate the standby database
   - 4.1. copy as many archived logs as you can get from the down database machine
   - 4.2. apply these logs
   - 4.3. copy the most recent "active" redo log from the production to the standby machine
   - 4.4. apply this log, supplying its filename explicitly
   - 4.5. from svrmgrl, issue 'alter database activate standby database'
5. Prepare the old production database as the new standby database
   - 5.1. drop the old RMAN catalog user (if no other RMAN backups are tracked therein)
   - 5.2. create a new RMAN catalog user and 'create catalog'
   - 5.3. create a full RMAN backup of the new production database
   - 5.4. repeat the previous steps