;login:

THE MAGAZINE OF USENIX & SAGE
August 2003 • volume 28 • number 4

inside:

SYSADMIN
Hoskins: Oracle Hot Backup

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

Oracle hot backup

by Matthew E. Hoskins

Matthew Hoskins is a UNIX system administrator for the New Jersey Institute of Technology, where he maintains many of the corporate administrative systems. He enjoys trying to get wildly different systems and software working together, usually with a thin layer of Perl (aka "MattGlue").

matt@njit.edu

Introduction

I am not really a DBA. However, like many UNIX sysadmins, I am charged with the care and feeding of a number of Oracle database servers. In this short article I will explain the concepts behind various methods of backing up Oracle database servers, a critically important subject for anyone in my position. This is not an introductory article to Oracle, and I do assume at least a basic understanding of SQL and Oracle command line utilities.

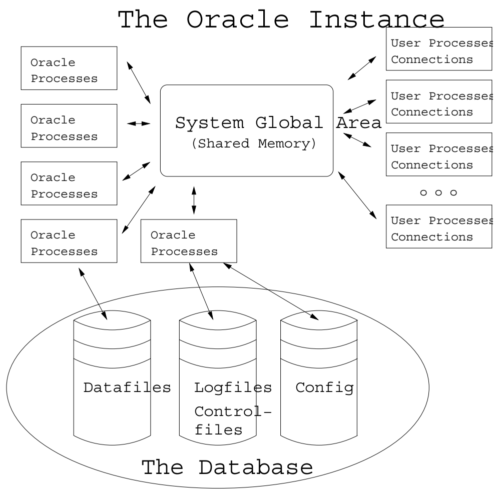
Enterprise database servers can seem complex and daunting, but Oracle has a long history and is based on proven technology and concepts that most UNIX system administrators are comfortable with, like textual configuration files, scripting capability, and command line interfaces. Even if your DBAs or programmers manage their own Oracle backups, it can be of value to understand the methods and possibilities.

Oracle Concepts

Oracle database servers are composed of "instances" which are identified by an SID (system identifier). An instance contains memory structures, processes, tablespaces, and various data and configuration files. In most circles, the word "database" is synonymous with "instance," at least in Oracle, but in actuality the database is a component of the instance.

Tablespaces comprise a group of datafiles which can be dynamically allocated. During normal operation, Oracle is constantly updating various files inside the instance; these files can be (and usually should be) spread between physical partitions/devices on the system. If Oracle files are spread across physical partitions in a logical manner, the loss of any one partition should not result in any data loss. Oracle calls this layout "OFA" (Oracle Flexible Architecture). Using the concepts of OFA is key in creating a recoverable and manageable Oracle installation. After all, the first step in implementing a successful backup system is having a properly configured database server.

In most cases, Oracle gives you the option to maintain multiple copies of critical files (such as control files(s) and online redo logs) automatically; Oracle calls this "multiplexing." This can give you an extra layer of protection against physical media failure. Every update applied to a datafile is handled as a transaction and can be "rolled back." This is handled internally by maintaining sets of online redo logs. For additional data recoverability, the online redo logs will be automatically archived if an instance is placed in ARCHIVELOG mode. Archive logs enable one to play back every update applied to datafiles/tables literally. This is very useful for recovery from disaster or human error. Using archived redo logs, a DBA can "time travel" and restore any table to any point for which uninterrupted archive logs are available, a truly powerful feature.



Types of Oracle Backup

Three major methods exist for backing up Oracle databases: offline, RMAN, and "online hot." Offline backup is virtually unacceptable in 24/7 production environments, but it is the safest and also easiest from which to restore. During offline backup, the Oracle instance is completely shut down and all files can be safely backed up using normal backup tools (since none of them is changing). Offline backups are still very useful; it is quite common to perform an offline backup both before and after applying Oracle software patches/upgrades.

RMAN (Recovery Manager) is a relatively new method of online backup and is mainly designed to provide an API for integrating automated enterprise backup software with Oracle. One major feature of RMAN is its ability to make incremental backups, which is sometimes necessary for huge tablespaces. However, RMAN has a large overhead of complexity that can be avoided unless you need its advanced features.

Finally, "hot backups" put individual tablespaces in a special mode that enables datafiles to be copied or archived using normal backup tools. The instance is required to be in ARCHIVELOG mode for online hot backups to work. Hot backups generally make the most sense to the average sysadmin, because they may be scripted using tools with which we have great familiarity.

A fourth method of backup, mentioned only for completeness, is the "logical backup" in which one does a partial or full export of the schema and data in an instance. An export creates a single file (which might be broken up) that contains the data and commands to recreate whatever was exported. While this method might seem elegant for backups, in order to do a restore one has to have an already working Oracle installation, which would require running Oracle Universal Installer (OUI) and applying all patches to the same point as the export was created from. This can be rather time-consuming in Oracle and is not recommended as a backup method. Logical imports/exports are better used for point-in-time archives and moving data from one server to another. Import/export can also be useful for copying data from a production instance to a test or development instance. Since import/export recreates database components using data manipulation operations, it can be quite time-consuming on large databases, because the data must be re-indexed as it is loaded.

Hot Backup Concepts

The remainder of this article focuses on hot backups. In my opinion, they are the most flexible method from the standpoint of a sysadmin for small to medium-sized tablespaces. When datafiles are put into hot backup mode, Oracle is still writing to them; this makes your average system administrator raise an eyebrow. Oracle slightly changes the way it performs updates to datafiles by copying whole datafile blocks, rather than just what has changed, into the online redo logs. Hot backups, by their very nature, take a corrupt copy of the datafiles. But relax, it's OK. Since the instance is in ARCHIVELOG mode, all changes to the datafiles are available upon restore to correct any discrepancies created as the archiving process passes over any given datafile. This is why Oracle records redo information in whole blocks during hot backup: to help recover any inconsistent blocks in the datafiles upon restore. The basic steps in online hot backups are as follows (SQL statements are in parenthesis):

1. Get listing of tablespaces and datafiles (`SELECT tablespace_name, file_name FROM sys.dba_data_files`).
2. For each tablespace
    a. Put tablespace in hot backup mode (`ALTER TABLESPACE $TABLESPACENAME BEGIN BACKUP`).
    b. Copy, archive, or snapshot each datafile in this tablespace.
    c. End hot backup mode for this tablespace (`ALTER TABLESPACE $TABLESPACENAME END BACKUP`).
3. Back up the Oracle "controlfile" (`ALTER DATABASE BACKUP CONTROLFILE TO TRACE` and/or `ALTER DATABASE BACKUP CONTROLFILE TO $FILE`).

4. Confirm all tablespaces returned to normal mode (SELECT FILE#,STATUS,CHANGE#,TIME FROM v$backup WHERE STATUS != 'NOT ACTIVE').
5. Perform an archive log switch (ALTER SYSTEM ARCHIVE LOG CURRENT).
6. Backup archived redo logs.
7. Backup everything else (binaries, config files, etc . . .).

Most of the steps are performed by executing SQL statements in SQL*Plus or some other method; the archiving of datafiles and software can be done with any normal methods (e.g., tar, cp, cpio). As you can see, this process spans two distinct operating environments. Steps must be performed inside Oracle, then further operations must be executed at the OS and file-system level. This sounds like a job for a script! More on that later.

Proper Treatment of Archive Logs

Archive logs are rather important and should always be hosted on a separate physical disk or array from the datafiles. If you lose the partition that houses datafiles, the archive logs can be used to *completely* recover the datafiles up to the moment they were lost. Depending on the application and its importance, it might be beneficial to copy archive logs off-site a number of times per day.

The “rsync” (*http://rsync.samba.org/*) utility works well for this task, because it can keep a remote copy of a file system up to date by only transferring incremental changes. Since every restore done from hot backups requires datafile media recovery using archived redo logs, these logs should always be archived after all tablespaces are taken out of hot backup mode and kept with the archived datafiles. Archive logs must be treated with respect; a single missing or corrupt archive log file will cause all the logs created after it to be useless! Generally archive logs compress well – many DBAs use gzip or bzip2 to compress the archive logs nightly with a cron job – but make sure you don’t try to compress the one that Oracle is writing to.

The Restore

Restoring from hot backups is a multi-step process that is mostly handled by Oracle itself, provided you supply all the files necessary. When one or more datafiles have been lost, the following high-level steps are performed:

1. Take the affected tablespace offline (if Oracle is still running, ALTER TABLESPACE $TABLESPACENAME OFFLINE TEMPORARY, where $TABLESPACE is the tablespace that lost the datafile).
2. Restore lost datafile(s) from last available hot backup.
3. Confirm that archived redo logs are available in the location Oracle expects them to be, and that they are uncompressed. (Oracle will prompt interactively if the ones it’s looking for are not found.)
4. Tell Oracle to recover the datafile(s) automatically (ALTER DATABASE RECOVER AUTOMATIC TABLESPACE $TABLESPACE). You can monitor the progress by tailing the alert.log files.
5. Bring tablespace online (ALTER TABLESPACE $TABLESPACE ONLINE).

And you’re open for business again.

For a complete listing of scenarios and proper restore procedures, see the Oracle Backup and Restore manual referenced at the end of this document. It is important to make a regular habit of testing your backups. In fact, I would go so far as to say it would be insane to move forward with a backup solution without testing various restore scenarios. I would suggest testing at least loss of a single datafile and loss of the entire ORACLE_HOME and all datafiles. For testing, it is quite easy to set up a secondary Oracle instance which can be used to test backups. If this system is in a different geographic location, it can be used for disaster recovery also.

In any event, it is a good idea to completely script the process of restoring all datafiles and recovering the instance. This can make disaster recovery much less painful. If you are the person people come looking for when things are broken, this is definitely a good thing. It is, of course, important to keep this script up-to-date as circumstances change around it. Also, while this standby system is not being used for disaster recovery, it can be used as a perfect playground for development.

Script Design

There is no shortage of good Oracle hot backup scripts freely available; simply do a Google search for “Oracle hot backup script”. However, none of the ones I found exactly scratched my itch. The solution I designed uses disk-to-disk backups for our various databases (MySQL, LDAP, Oracle, etc.). Disks are quite cheap these days; large IDE RAID systems and either direct attached or NAS (Network Attached Storage) devices are surprisingly affordable. These high-capacity/low-cost disk arrays can be used to widen your backup window by first staging data to disk before tape. Also note, bandwidth permitting, these disks might be off-site, thus automating off-site backups and eliminating the expense of an off-site backup service.

For certain high importance databases, we may keep several backup images on disk to save time if we need a restore. Using this methodology, our database dump process does not have to be synchronized with our enterprise network backup system which finally puts the data on tape. Another method we use (snapshots) exploits features available in some modern file systems that enable you to create a temporary read-only view of a file system from a particular point-in-time. Since these snapshots are normally instantaneous, it is common to place *all* tablespaces in hot backup mode during snapshot, then immediately disable hot backup after snapshot is completed. Since I use both of these methods of backing up databases, the script I constructed is able to handle both procedures. Some considerations when selecting any hot backup solution are:

1. Graceful error handling. If an error occurs during a hot backup, it is vitally important that the tablespace be taken out of hot backup mode before the script bombs out. During hot backup mode, Oracle takes a performance hit because it is recording redo data in full blocks, and it is important that the database not be allowed to remain in this mode for long periods.
2. Sanity checking. It is very important that the script check that the proper environment variables are set and are sane values. Also, some checking should be done to confirm the database is in a good state to be backed up. For instance, no datafiles should be listed as “Needs recovery.” The solution should also have sufficient locking that overlapping instances of the backup process will not clobber each other.
3. Event reporting. Errors (and success) should be reported. Syslog works nicely for this; in my organization we use a home-grown solution for real-time monitoring logs from all our systems for various events and use this as a basis for sending alerts.

4. Ease of restoring. Backups created should be in a universal and open format that preserves owner, mode, and original location of the datafiles (like tar with full path).
5. Should be able to clean up its own messes. If a backup is interrupted in the middle of its execution, it can leave tablespaces in hot backup mode. This condition should be detected and there should be a method for correcting the situation without needing to enter SQL*Plus manually.

The script I cooked up is written in Perl, has been tested on Oracle 8i and 9i on Linux and Solaris, and is released under GPL, so it is free for use in your organization. Get it at *http://www.njit.edu/~matt/oracle-hot-backup/*.

References

Oracle, *Oracle9i User-Managed Backup and Recovery Guide*, June 2001, Part No. A90134-01.

Michael Wessler, *Oracle DBA on Unix and Linux* (Sams, 2002).

USENIX and SAGE Need You

People often ask how they can contribute. Here is a list of tasks for which we hope to find volunteers.

The SAGEwire and SAGEweb staff are seeking:

- Interview candidates
- Short article contributors (see *http://sagewire.sage.org*)
- White paper contributors for topics like these:

Back-ups	Emerging technology	Privacy
Career development	User education/training	Product round-ups
Certification	Ethics	SAGEwire
Consulting	Great new products	Scaling
Culture	Group tools	Scripting
Databases	Networking	Security implementation
Displays	New challenges	Standards
Email	Performance analysis	Storage
Education	Politics and the sysadmin	Tools, system

- Local user groups: If you have a local user group affiliated (or wishing to affiliate) with SAGE, please email the particulars to *kolstad@sage.org* so they can be posted on the Web site.

;login: always needs conference summarizers for USENIX conferences. Contact Alain Hénon, *ah@usenix.org,* if you'd like to help.