Tutorial on Backup and Recovery

When data disappears, your boss wants it back, and your job is on the line.

One of the innumerable tasks of the DBA is to ensure that all of the databases of the enterprise are always "available." Availability in this context means that the users must be able to access the data stored in the databases, and that the contents of the databases must be up-to-date, consistent, and correct. It must never appear to a user that the system has lost the data or that the data has become inconsistent. This would totally ruin the user's confidence in the database and the entire system.

Many factors threaten the availability of your databases. These include natural disasters (such as floods and earthquakes), hardware failures (for example, a power failure or disk crash), software failures (such as DBMS malfunctions -- read "bugs" -- and application program errors), and people failures (for example, operator errors, user misunderstandings, and keyboard trouble). To this list you can also add the threats I listed last month under security, such as malicious attempts to destroy or corrupt the contents of the database.

In a large enterprise, the DBA must ensure the availability of several databases, such as the development databases, the databases used for unit and acceptance testing, the operational online production databases (some of which may be replicated or distributed all over the world), the data warehouse databases, the data marts, and all of the other departmental databases. All of these databases usually have different requirements for availability. The online production databases typically must be available, up-to-date, and consistent for 24 hours a day, seven days a week, with minimal downtime. The warehouse databases must be available and up-to-date during business hours and even for a while after hours.

On the other hand, the test databases need to be available only for testing cycles, but during these periods the testing staff may have extensive requirements for the availability of their test databases. For example, the DBA may have to restore the test databases to a consistent state after each test. The developers often have even more ad hoc requirements for the availability of the development databases, specifically toward the end of a crucial deadline. The business hours of a multinational organization may also have an impact on availability. For example, a working day from 8 a.m. in central Europe to 6 p.m. in California implies that the database must be available for 20 hours a day. The DBA is left with little time to provide for availability, let alone perform other maintenance tasks.

Recovery is the corrective process to restore the database to a usable state from an erroneous state. The basic recovery process consists of the following steps:

1. Identify that the database is in an erroneous, damaged, or crashed state.
2. Suspend normal processing.
3. Determine the source and extent of the damage.
4. Take corrective action, that is:
    - Restore the system resources to a usable state.
    - Rectify the damage done, or remove invalid data.
    - Restart or continue the interrupted processes, including the re-execution of interrupted transactions.
5. Resume normal processing.

To cope with failures, additional components and algorithms are usually added to the system. Most techniques use recovery data (that is, redundant data), which makes recovery possible. When taking corrective action, the effects of some transactions must be removed, while other transactions must be re-executed; some transactions must even be undone and redone. The recovery data must make it possible to perform these steps.

The following techniques can be used for recovery from an erroneous state:

Dump and restart: The entire database must be backed up regularly to archival storage. In the event of a failure, a copy of the database in a previous correct state (such as from a checkpoint) is loaded back into the database. The system is then restarted so that new transactions can proceed. Old transactions can be re-executed if they are available. The following types of restart can be identified:

- A warm restart is the process of starting the system after a controlled system shutdown, in which all active transactions were terminated normally and successfully.
- An emergency restart is invoked by a restart command issued by the operator. It may include reloading the database contents from archive storage.
- A cold start is when the system is started from scratch, usually when a warm restart is not possible. This may also include reloading the database contents from archive storage. Usually used to recover from physical damage, a cold restart is also used when recovery data was lost.

Undo-redo processing (also called roll-back and re-execute): By using an audit trail of transactions, all of the effects of recent, partially completed transactions can be undone up to a known correct state. Undoing is achieved by reversing the updating process. By working backwards through the log, all of the records of the transaction in question can be traced, until the begin transaction operations of all of the relevant transactions have been reached. The undo operation must be "idempotent," meaning that failures during undo operations must still result in the correct single intended undo operation taking place. From the known correct state, all of the journaled transactions can then be re-executed to obtain the desired correct resultant database contents. The operations of the transactions that were already executed at a previous stage are obtained from the audit trail. The redo operation must also be idempotent, meaning that failures during redo operations must still result in the correct single intended redo operation taking place. This technique can be used when partially completed processes are aborted.

Roll-forward processing (also called reload and re-execute): All or part of a previous correct state (for example, from a checkpoint) is reloaded; the DBA can then instruct the DBMS to re-execute the recently recorded transactions from the transaction audit trail to obtain a correct state. It is typically used when (part of) the physical media has been damaged.

Restore and repeat: This is a variation of the previous method, where a previous correct state is restored. The difference is that the transactions are merely reposted from before and/or after images kept in the audit trail. The actual transactions are not re-executed: They are merely reapplied from the audit trail to the actual data table. In other words, the images of the updated rows (the effects of the transactions) are replaced in the data table from the audit trail, but the original transactions are not re-executed as in the previous case.

As a result, the DBA has an extensive set of requirements for the tools and facilities offered by the DBMS. These include facilities to back up an entire database offline, facilities to back up parts of the database selectively, features to take a snapshot of the database at a particular moment, and obviously journaling facilities to roll back or roll forward the transactions applied to the database to a particular identified time. Some of these facilities must be used online -- that is, while the users are busy accessing the database. For each backup mechanism, there must be a corresponding restore mechanism -- these mechanisms should be efficient, because you usually have to restore a lost, corrupt, or damaged database at some critical moment, while the users are waiting anxiously (sometimes highly irritated) and the managers are jumping up and down (often ineffectually)! The backup and restore facilities should be configurable -- you may want to stream the backup data to and from multiple devices in parallel, you may want to add compression and decompression (including using third-party compression tools), you may want to delete old backups automatically off the disk, or you may want to label the tapes according to your own standards. You should also be able to take the backup of a database from one platform and restore it on another -- this step is necessary to cater for non-database-related problems, such as machine and operating system failures. For each facility, you should be able to monitor its progress and receive an acknowledgment that each task has been completed successfully.

Some organizations use so-called "hot standby" techniques to increase the availability of their databases. In a typical hot standby scenario, the operations performed on the operational database are replicated to a standby database. If any problems are encountered on the operational database, the users are switched over and continue working on the standby database until the operational database is restored. However, database replication is an involved and extensive topic -- I will cover it in detail in a subsequent column.

In the remainder of this month's column I investigate the tools and facilities offered by IBM, Informix, Microsoft, Oracle, and Sybase for backup and recovery.

IBM DB2

IBM's DB2 release 2.1.1 provides two facilities to back up your databases, namely the BACKUP command and the Database Director. It provides three methods to recover your database: crash recovery, restore, and roll-forward.

Backups can be performed either online or offline. Online backups are only supported if roll-forward recovery is enabled for the specific database. To execute the BACKUP command, you need SYSADM, SYSCTRL, or SYSMAINT authority. A database or a tablespace can be backed up to a fixed disk or tape. A tablespace backup and a tablespace restore cannot be run at the same time, even if they are working on different tablespaces. The backup command provides concurrency control for multiple processes making backup copies of different databases at the same time.

The restore and roll-forward methods provide different types of recovery. The restore-only recovery method makes use of an offline, full backup copy of the database; therefore, the restored database is only as current as the last backup. The roll-forward recovery method makes use of database changes retained in logs -- therefore it entails performing a restore database (or tablespaces) using the BACKUP command, then applying the changes in the logs since the last backup. You can only do this when roll-forward recovery is enabled. With full database roll-forward recovery, you can specify a date and time in the processing history to which to recover.

Crash recovery protects the database from being left in an inconsistent state. When transactions against the database are unexpectedly interrupted, you must perform a rollback of the incomplete and in-doubt transactions, as well as the completed transactions that are still in memory. To do this, you use the RESTART DATABASE command. If you have specified the AUTORESTART parameter, a RESTART DATABASE is performed automatically after each failure. If a media error occurs during recovery, the recovery will continue, and the erroneous tablespace is taken offline and placed in a roll-forward pending state. The offline tablespace will need additional fixing up -- restore and/or roll-forward recovery, depending on the mode of the database (whether it is recoverable or non-recoverable).

Restore recovery, also known as version control, lets you restore a previous version of a database made using the BACKUP command. Consider the following two scenarios:

- A database restore will rebuild the entire database using a backup made earlier, thus restoring the database to the identical state when the backup was made.
- A tablespace restore is made from a backup image, which was created using the BACKUP command where only one or more tablespaces were specified to be backed up. Therefore this process only restores the selected tablespaces to the state they were in when the backup was taken; it leaves the unselected tablespaces in a different state. A tablespace restore can be done online (shared mode) or offline (exclusive mode).

Roll-forward recovery may be the next task after a restore, depending on your database's state. There are two scenarios to consider:

- Database roll-forward recovery is performed to restore the database by applying the database logs. The database logs record all of the changes made to the database. On completion of this recovery method, the database will return to its prefailure state. A backup image of the database and archives of the logs are needed to use this method.
- Tablespace roll-forward can be done in two ways: either by using the ROLLFORWARD command to apply the logs against the tablespaces in a roll-forward pending state, or by performing a tablespace restore and roll-forward recovery, followed by a ROLLFORWARD operation to apply the logs.

Informix

Informix for Windows NT release 7.12 has a Storage Manager Setup tool and a Backup and Restore tool. These tools let you perform complete or incremental backups of your data, back up logical log files (continuous and manual), restore data from a backup device, and specify the backup device.

Informix has a Backup and Restore wizard to help you with your backup and restore operations. This wizard is only available on the server machine. The Backup and Restore wizard provides three options: Backup, Logical Log Backup, and Restore.

The Backup and Restore tool provides two types of backups: complete and incremental. A complete backup backs up all of the data for the selected database server. A complete backup -- also known as a level-0 backup -- is required before you can do an incremental backup. An incremental backup -- also known as a level-1 backup -- backs up all changes that have occurred since the last complete backup, thereby requiring less time because only part of the data from the selected database server is backed up. You also get a level-2 backup, performed using the command-line utilities, that is used to back up all of the changes that have occurred since the last incremental backup. The Backup and Restore tool provides two types of logical log backups: continuous backup of the logical logs and manual backup of the logical logs. A Logical Log Backup backs up all full and used logical log files for a database server. The logical log files are used to store records of the online activity that occurs between complete backups.

The Informix Storage Manager (ISM) Setup tool lets you specify the storage device for storing the data used for complete, incremental, and logical log backups. The storage device can be a tape drive, a fixed hard drive, a removable hard drive, or none (for example, the null device). It is only available on the server machine. You can select one backup device for your general backups (complete or incremental) and a separate device for your logical log backups. You always have to move the backup file to another location or rename the file before starting your next backup. Before restoring your data, you must move the backup file to the directory specified in the ISM Setup and rename the backup file to the filename specified in ISM Setup.

If you specify None as your logical log storage device, the application marks the logical log files as backed up as soon as they become full, effectively discarding logical log information. Specify None only if you do not need to recover transactions from the logical log. When doing a backup, the server must be online or in administration mode. Once the backup has started, changing the mode will terminate the backup process. When backing up to your hard drive, the backup file will be created automatically.

The Restore option of the Backup and Restore wizard restores the data and logical log files from a backup source. You cannot restore the data if you have not made a complete backup. The server must be in offline mode during the restore operation. You can back up your active logical log files before doing the restore, and you can also specify which log files must be used. A level-1 (incremental) backup can be restored, but you will be prompted to proceed with a level-2 backup at the completion of the level-1 restore. Once the restore is completed, the database server can be brought back online, and processing can continue as usual. If you click on Cancel during a restore procedure, the resulting data may be corrupted.

Microsoft SQL Server

Microsoft SQL Server 6.5 provides more than one backup and recovery mechanism. For backups of the database, the user can either use the Bulk Copy Program (BCP) from the command line to create flat-file backups of individual tables or the built-in Transact-SQL DUMP and LOAD statements to back up or restore the entire database or specific tables within the database.

Although the necessary Transact-SQL statements are available from within the SQL environment, the Microsoft SQL Enterprise Manager provides a much more user-friendly interface for making backups and recovering them later on. The Enterprise Manager will prompt the DBA for information such as database name, backup device to use, whether to initialize the device, and whether the backup must be scheduled for later or done immediately. Alternatively, you can use the Database Maintenance wizard to automate the whole maintenance process, including the backup procedures. These tasks are automatically scheduled by the wizard on a daily or weekly basis. Both the BCP utility and the dump statement can be run online, which means that users do not have to be interrupted while backups are being made. This facility is particularly valuable in 24 X 7 operations.

A database can be restored up to the last committed transaction by also LOADing the transaction logs that were dumped since the previous database DUMP. Some of the LOAD options involve more management. For example, the database dump file and all subsequent transaction-log dump files must be kept until the last minute in case recovery is required. It is up to the particular site to determine a suitable backup and recovery policy, given the available options.

To protect against hardware failures, Microsoft SQL Server 6.5 has the built-in capability to define a standby server for automatic failover. This option requires sophisticated hardware but is good to consider for 24 X 7 operations. Once configured, it does not require any additional tasks on an ongoing basis. In addition, separate backups of the database are still required in case of data loss or multiple media failure.

Oracle

Oracle7 Release 7.3 uses full and partial database backups and a redo log for its database backup and recovery operations. The database backup is an operating system backup of the physical files that constitute the Oracle database. The redo log consists of two or more preallocated files, which are used to record all changes made to the database. You can also use the export and import utilities to create a backup of a database. Oracle offers a standby database scheme, with which it maintains a copy of a primary database on duplicate hardware, in a constant recoverable state, by applying the redo logs archived off the primary database.

A full backup is an operating system backup of all of the data files, parameter files, and the control file that constitute the database. A full database backup can be taken by using the operating system's commands or by using the host command of the Server Manager. A full database backup can be taken online when the database is open, but only an offline database backup (taken when the database server is shut down) will necessarily be consistent. An inconsistent database backup must be recovered with the online and archived redo log files before the database will become available. The best approach is to take a full database backup after the database has been shut down with normal or immediate priority.

A partial backup is any operating system backup of a part of the full backup, such as selected data files, the control file only, or the data files in a specified tablespace only. A partial backup is useful if the database is operated in ARCHIVELOG mode. A database operating in NOARCHIVE mode rarely has sufficient information to use a partial backup to restore the database to a consistent state. The archiving mode is usually set during database creation, but it can be reset at a later stage.

You can recover a database damaged by a media failure in one of three ways after you have restored backups of the damaged data files. These steps can be performed using the Server Manager's Apply Recovery Archives dialog box, using the Server Manager's RECOVER command, or using the SQL ALTER DATABASE command:

- You can recover an entire database using the RECOVER DATABASE command. This command performs media recovery on all of the data files that require redo processing.
- You can recover specified tablespaces using the RECOVER TABLESPACE command. This command performs media recovery on all of the data files in the listed tablespaces. Oracle requires the database to be open and mounted in order to determine the file names of the tables contained in the tablespace.
- You can list the individual files to be recovered using the RECOVER DATAFILE command. The database can be open or closed, provided that Oracle can take the required media recovery locks.

In certain situations, you can also recover a specific damaged data file, even if a backup file isn't available. This can only be done if all of the required log files are available and the control file contains the name of the damaged file. In addition, Oracle provides a variety of recovery options for different crash scenarios, including incomplete recovery, change-based, cancel-based, and time-based recovery, and recovery from user errors.

Sybase SQL Server

Sybase SQL Server 11 uses database dumps, transaction dumps, checkpoints, and a transaction log per database for database recovery. All backup and restore operations are performed by an Open Server program called Backup Server, which runs on the same physical machine as the Sybase SQL Server 11 process.

A database dump is a complete copy of the database, including the data files and the transaction log. This function is performed using the DUMP DATABASE operation, which can place the backup on tape or on disk. You can make dynamic dumps, which let the users continue using the database while the dump is being made. A transaction dump is a routine backup of the transaction log. The DUMP TRANSACTION operation also truncates the inactive portion of the transaction log file. You can use multiple devices in the DUMP DATABASE and DUMP TRANSACTION operations to stripe the dumps across multiple devices.

The transaction log is a write-ahead log, maintained in the system table called syslogs. You can use the DUMP TRANSACTION command to copy the information from the transaction log to a tape or disk. You can use the automatic checkpointing task or the CHECKPOINT command (issued manually) to synchronize a database with its transaction log. Doing so causes the database pages that are modified in memory to be flushed to the disk. Regular checkpoints can shorten the recovery time after a system crash.

Each time Sybase SQL Server restarts, it automatically checks each database for transactions requiring recovery by comparing the transaction log with the actual data pages on the disk. If the log records are more recent than the data page, it reapplies the changes from the transaction log.

An entire database can be restored from a database dump using the LOAD DATABASE command. Once you have restored the database to a usable state, you can use the LOAD TRANSACTION command to load all transaction log dumps, in the order in which they were created. This process reconstructs the database by re-executing the transactions recorded in the transaction log.

You can use the DUMP DATABASE and LOAD DATABASE operations to port a database from one Sybase installation to another, as long as they run on similar hardware and software platforms.

Prevention is Better than Cure. . .

Although each DBMS I reviewed has a range of backup and recovery facilities, it is always important to ensure that the facilities are used properly and adequately. By "adequately," I mean that backups must be taken regularly. All of the DBMSs I reviewed provided the facilities to repost or re-execute completed transactions from a log or journal file. However, reposting or re-executing a few weeks' worth of transactions may take an unbearably long time. In many situations, users require quick access to their databases, even in the presence of media failures. Remember that the end users are not concerned with physical technicalities, such as restoring a database after a system crash.

Even better than quick recovery is no recovery, which can be achieved in two ways. First, by performing adequate system monitoring and using proper procedures and good equipment, most system crashes can be avoided. It is better to provide users with a system that is up and available 90 percent of the time than to have to do sporadic fixes when problems occur. Second, by using redundant databases such as hot standby or replicated databases, users can be relieved of the recovery delays: Users can be switched to the hot backup database while the master database is being recovered.

A last but extremely important aspect of backup and recovery is testing. Test your backup and recovery procedures in a test environment before deploying them in the production environment. In addition, the backup and recovery procedures and facilities used in the production environment must also be tested regularly. A recovery scheme that worked perfectly well in a test environment is useless if it cannot be repeated in the production environment -- particularly in that crucial moment when the root disk fails during the month-end run!

Database Backup and Recovery from Oracle Point of view

Database Backups

No matter what backup and recovery scheme you devise for an Oracle database, backups of the database's datafiles and control files are absolutely necessary as part of the strategy to safeguard against potential media failures that can damage these files.

The following sections provide a conceptual overview of the different types of backups that can be made and their usefulness in different recovery schemes. The Oracle8 Server Backup and Recovery Guide provides more details, along with guidelines for performing database backups.

Whole Database Backups

A *whole database backup* is an operating system backup of all datafiles and the control file that constitute an Oracle database. A whole backup should also include the parameter file(s) associated with the database. You can take a whole database backup when the database is shut down or while the database is open. You should not normally take a whole backup after an instance failure or other unusual circumstances.

Consistent Whole Backups vs. Inconsistent Whole Backups

Following a clean shutdown, all of the files that constitute a database are closed and consistent with respect to the current point in time. Thus, a whole backup taken after a shutdown can be used to recover to the point in time of the last whole backup. A whole backup taken while the database is open is not consistent to a given point in time and must be recovered (with the online and archived redo log files) before the database can become available.

Backups and Archiving Mode

The datafiles obtained from a whole backup are useful in any type of media recovery scheme:

- If a database is operating in NOARCHIVELOG mode and a disk failure damages some or all of the files that constitute the database, the most recent consistent whole backup can be used to *restore* (not recover) the database.

Because an archived redo log is not available to bring the database up to the current point in time, all database work performed since the backup must be repeated. Under special circumstances, a disk failure in NOARCHIVELOG mode can be fully recovered, but you should not rely on this.

- If a database is operating in ARCHIVELOG mode and a disk failure damages some or all of the files that constitute the database, the datafiles collected by the most recent whole backup can be used as part of database *recovery*.

After restoring the necessary datafiles from the whole backup, database recovery can continue by applying archived and current online redo log files to bring the restored datafiles up to the current point in time.

In summary, if a database is operated in NOARCHIVELOG mode, a consistent whole database backup is the only method to partially protect the database against a disk failure; if a database is operating in ARCHIVELOG mode, either a consistent or an inconsistent whole database backup can be used to restore damaged files as part of database recovery from a disk failure.

Partial Database Backups

A *partial database backup* is any backup short of a whole backup, taken while the database is open or shut down. The following are all examples of partial database backups:

- a backup of all datafiles for an individual tablespace
- a backup of a single datafile
- a backup of a control file

Partial backups are only useful for a database operating in ARCHIVELOG mode. Because an archived redo log is present, the datafiles restored from a partial backup can be made consistent with the rest of the database during recovery procedures.

Datafile Backups

A partial backup includes only some of the datafiles of a database. Individual or collections of specific datafiles can be backed up independently of the other datafiles, online redo log files, and control files of a database. You can back up a datafile while it is offline or online.

Choosing whether to take online or offline datafile backups depends only on the availability requirements of the data - online datafile backups are the only choice if the data being backed up must always be available.

Control File Backups

Another form of a partial backup is a control file backup. Because a control file keeps track of the associated database's physical file structure, a backup of a database's control file should be made every time a structural change is made to the database.

Note: The Recovery Manager automatically backs up the control file in any backup that includes datafile 1, which contains the data dictionary.

Multiplexed control files safeguard against the loss of a single control file. However, if a disk failure damages the datafiles and incomplete recovery is desired, or a point-in-time recovery is desired, a backup of the control file that corresponds to the intended database structure should be used, not necessarily the current control file. Therefore, the use of multiplexed control files is not a substitute for control file backups taken every time the structure of a database is altered.

If you use Recovery Manager to restore the control file prior to incomplete or point-in-time recovery, Recovery Manager automatically restores the most suitable backup control file.

This section covers the structures and software mechanisms used by Oracle to provide:

- database recovery required by different types of failures
- flexible recovery operations to suit any situation
- availability of data during backup and recovery operations so that users of the system can continue to work

Why Is Recovery Important?

In every database system, the possibility of a system or hardware failure always exists. Should a failure occur and affect the database, the database must be recovered. The goals after a failure are to ensure that the effects of all committed transactions are reflected in the recovered database and to return to normal operation as quickly as possible while insulating users from problems caused by the failure.

Types of Failures

Several circumstances can halt the operation of an Oracle database. The most common types of failure are described below:

user error	User errors can require a database to be recovered to a point in time before the error occurred. For example, a user might accidentally drop a table. To allow recovery from user errors and accommodate other unique recovery requirements, Oracle provides for exact point-in-time recovery. For example, if a user accidentally drops a table, the database can be recovered to the instant in time before the table was dropped.
statement and process failure	Statement failure occurs when there is a logical failure in the handling of a statement in an Oracle program (for example, the statement is not a valid SQL construction). When statement failure occurs, the effects (if any) of the statement are automatically undone by Oracle and control is returned to the user. A *process failure* is a failure in a user process accessing Oracle, such as an abnormal disconnection or process termination. The failed user process cannot continue work, although Oracle and other user processes can. The Oracle background process PMON automatically detects the failed user process or is informed of it by SQL*Net. PMON resolves the problem by rolling back the uncommitted transaction of the user process and releasing any resources that the process was using. Common problems such as erroneous SQL statement constructions and aborted user processes should never halt the database system as a whole. Furthermore, Oracle automatically performs necessary recovery from uncommitted transaction changes and locked resources with minimal impact on the system or other users.
instance failure	Instance failure occurs when a problem arises that prevents an instance (system global area and background processes) from continuing work. Instance failure may result from a hardware problem such as a power outage, or a software problem such as an operating system crash. When an instance failure occurs, the data in the buffers of the system global area is

	not written to the datafiles. Instance failure requires *instance recovery*. Instance recovery is automatically performed by Oracle when the instance is restarted. The redo log is used to recover the committed data in the SGA's database buffers that was lost due to the instance failure.
media (disk) failure	An error can arise when trying to write or read a file that is required to operate the database. This is called disk failure because there is a physical problem reading or writing physical files on disk. A common example is a disk head crash, which causes the loss of all files on a disk drive. Different files may be affected by this type of disk failure, including the datafiles, the redo log files, and the control files. Also, because the database instance cannot continue to function properly, the data in the database buffers of the system global area cannot be permanently written to the datafiles. A disk failure requires *media recovery*. Media recovery restores a database's datafiles so that the information in them corresponds to the most recent time point before the disk failure, including the committed data in memory that was lost because of the failure. To complete a recovery from a disk failure, the following is required: backups of the database's datafiles, and all online and necessary archived redo log files.

Oracle provides for complete and quick recovery from all possible types of hardware failures including disk crashes. Options are provided so that a database can be completely recovered or partially recovered to a specific point in time.

If some datafiles are damaged in a disk failure but most of the database is intact and operational, the database can remain open while the required tablespaces are individually recovered. Therefore, undamaged portions of a database are available for normal use while damaged portions are being recovered.

Structures Used for Recovery

Oracle uses several structures to provide complete recovery from an instance or disk failure: the redo log, rollback segments, a control file, and necessary database backups.

The Redo Log

As described in "Redo Log Files" on page 1-11, the *redo log* is a set of files that protect altered database data in memory that has not been written to the datafiles. The redo log can consist of two parts: the online redo log and the archived redo log.

The Online Redo Log

The *online redo log* is a set of two or more *online redo log files* that record all committed changes made to the database. Whenever a transaction is committed, the corresponding redo entries temporarily stored in redo log buffers of the system global area are written to an online redo log file by the background process LGWR.

The online redo log files are used in a cyclical fashion; for example, if two files constitute the online redo log, the first file is filled, the second file is filled, the first file is reused and filled, the second file is reused and filled, and so on. Each time a file is filled, it is assigned a *log sequence number* to identify the set of redo entries.

To avoid losing the database due to a single point of failure, Oracle can maintain multiple sets of online redo log files. A *multiplexed online redo log* consists of copies of online redo log files physically located on separate disks; changes made to one member of the group are made to all members.

If a disk that contains an online redo log file fails, other copies are still intact and available to Oracle. System operation is not interrupted and the lost online redo log files can be easily recovered using an intact copy.

The Archived Redo Log

Optionally, filled online redo files can be archived before being reused, creating an *archived redo log*. *Archived (offline) redo log files* constitute the archived redo log.

The presence or absence of an archived redo log is determined by the mode that the redo log is using:

ARCHIVELOG	The filled online redo log files are archived before they are reused in the cycle.
NOARCHIVELOG	The filled online redo log files are not archived.

In ARCHIVELOG mode, the database can be completely recovered from both instance and disk failure. The database can also be backed up while it is open and available for use. However, additional administrative operations are required to maintain the archived redo log.

If the database's redo log is operated in NOARCHIVELOG mode, the database can be completely recovered from instance failure, but not from a disk failure. Additionally, the database can be backed up only while it is completely closed. Because no archived redo log is created, no extra work is required by the database administrator.

Control Files

The *control files* of a database keep, among other things, information about the file structure of the database and the current log sequence number being written by LGWR. During normal recovery procedures, the information in a control file is used to guide the automated progression of the recovery operation.

Multiplexed Control Files

This feature is similar to the multiplexed redo log feature: a number of identical control files may be maintained by Oracle, which updates all of them simultaneously.

Rollback Segments

As described in "Data Blocks, Extents, and Segments" on page 1-9, rollback segments record rollback information used by several functions of Oracle. During database recovery, after all changes recorded in the redo log have been applied, Oracle uses rollback segment information to undo any uncommitted transactions. Because rollback segments are stored in the database buffers, this important recovery information is automatically protected by the redo log.

Database Backups

Because one or more files can be physically damaged as the result of a disk failure, media recovery requires the restoration of the damaged files from the most recent operating system backup of a database. There are several ways to back up the files of a database.

Whole Database Backups

A whole database backup is an operating system backup of all datafiles, online redo log files, and the control file that constitutes an Oracle database. Full backups are performed when the database is closed and unavailable for use.

Partial Backups

A partial backup is an operating system backup of part of a database. The backup of an individual tablespace's datafiles or the backup of a control file are examples of partial backups. Partial backups are useful only when the database's redo log is operated in ARCHIVELOG mode.

A variety of partial backups can be taken to accommodate any backup strategy. For example, you can back up datafiles and control files when the database is open or closed, or when a specific tablespace is online or offline. Because the redo log is operated in ARCHIVELOG mode, additional backups of the redo log are not necessary; the archived redo log is a backup of filled online redo log files.

Basic Recovery Steps

Due to the way in which DBWR writes database buffers to datafiles, at any given point in time, a datafile may contain some data blocks tentatively modified by uncommitted transactions and may not contain some blocks modified by committed transactions. Therefore, two potential situations can result after a failure:

- Blocks containing committed modifications were not written to the datafiles, so the changes may only appear in the redo log. Therefore, the redo log contains committed data that must be applied to the datafiles.
- Since the redo log may have contained data that was not committed, uncommitted transaction changes applied by the redo log during recovery must be erased from the datafiles.

To solve this situation, two separate steps are always used by Oracle during recovery from an instance or media failure: rolling forward and rolling back.

Rolling Forward

The first step of recovery is to *roll forward*, that is, reapply to the datafiles all of the changes recorded in the redo log. Rolling forward proceeds through as many redo log files as necessary to bring the datafiles forward to the required time.

If all needed redo information is online, Oracle performs this recovery step automatically when the database starts. After roll forward, the datafiles contain all committed changes as well as any uncommitted changes that were recorded in the redo log.

Rolling Back

The roll forward is only half of recovery. After the roll forward, any changes that were not committed must be undone. After the redo log files have been applied, then the rollback segments are used to identify and undo transactions that were never committed, yet were recorded in the redo log. This process is called *rolling back*. Oracle completes this step automatically.

The Recovery Manager

The Recovery Manager is an Oracle utility that manages backup and recovery operations, creating backups of database files and restoring or recovering a database from backups.

Recovery Manager maintains a repository called the *recovery catalog*, which contains information about backup files and archived log files. Recovery Manager uses the recovery catalog to automate both restore operations and media recovery.

The recovery catalog contains:

- information about backups of datafiles and archivelogs
- information about datafile copies
- information about archived redo logs and copies of them
- information about the physical schema of the target database
- named sequences of commands called *stored scripts.*

Review Question

1. What are backups and why it is important?

Selected Bibliography

For more information about the Recovery Manager, see the Oracle8 Server Backup and Recovery Guide.

* IBM Corp., Armonk, NY; 800-425-3333 or 914-765-1900; www.ibm.com.
* Informix Software Inc., Menlo Park, CA; 800-331-1763, 415-926-6300, or fax 415-926-6593; www.informix.com.

* Microsoft Corp., Redmond, WA; 800-426-9400, 206-882-8080, or fax 206-936-7329; www.microsoft.com.
* Oracle Corp., Redwood Shores, CA; 800-672-2537, 415-506-7000, or fax 415-506-7200; www.oracle.com.
* Sybase Inc., Emeryville, CA; 800-879-2273, 510-922-3500, or fax 510-922-9441; www.sybase.com.