Oracle Database Administration

Course Designer and Acquisition Editor

Centre for Information Technology and Engineering

Manonmaniam Sundaranar University

Tirunelveli

Oracle Database Administration

CONTENTS

Part 1 Basic Database Administration

Lecture 1 Oracle database administrator 1

- γ Types of Oracle Users
- γ Database Administrator Security and Privileges
- γ The DBA Role
- γ Database Administrator Utilities
- γ Initial Priorities of a Database Administrator
- γ Identifying Oracle Software Releases
- γ Release Number Format

Lecture 2 Database Administrator Authentication 11

- γ Database Administrator Authentication
- γ Using Operating System Authentication
- γ OSOPER and OSDBA
- γ Using an Authentication Password File
- γ Password File Administration

Lecture 3 The oracle server 21

- γ The Oracle Server
- γ Oracle Databases
- γ Memory Structure and Processes
- γ Memory Structures
- γ Process Architecture
- γ The Program Interface
- γ An Example of How Oracle Works

Lecture 4 Database structure and space management 31

- γ Database structure and space management
- γ Logical database structure
- γ Data blocks
- γ Extents
- γ Segments
- γ Physical database structure
- γ Datafiles
- γ Redo log files
- γ Control files

Lecture 5 Tablespaces and datafiles 43

- γ Introduction to tablespace and datafiles
- γ Tablespace
- γ Datafiles

Lecture 6 Creating oracle database 52

- γ Considerations Before Creating a Database
- γ Creating an Oracle Database

γ Troubleshooting Database
γ Dropping a Database
γ Parameters
γ Considerations After Creating a Database
γ Initial Tuning Guidelines

Lecture 7 Startup and shut down 64

γ Startup Procedures
γ Altering Database Availability
γ Shutdown Procedures
γ Using Parameter Files

Part 2 Oracle Server Configurations

Lecture 8 Managing oracle processes 75

γ Configuring Oracle for Dedicated Server Processes
γ Configuring Oracle for Multi-Threaded Server Processes
γ Modifying Server Processes
γ Terminating Sessions

Lecture 9 Managing online redo log 85

γ Planning the Online Redo Log
γ Multiplex the Online Redo Log
γ Creating Online Redo Log Groups and Members
γ Renaming and Relocating Online Redo Log Members
γ Dropping Online Redo Log Groups
γ Dropping Online Redo Log Members
γ Clearing an Online Redo Log File
γ Restrictions
γ Listing Information about the Online Redo Log

Lecture 10 Managing control files 95

γ Guidelines for Control Files
γ Multiplex control files
γ Creating Control Files
γ Relocating Control Files
γ Dropping Control Files

Lecture 11 Managing job queues 103

γ SNP Background Processes
γ Managing Job Queues
γ Terminating a Job
γ Viewing Job Queue Information

Part 3 Database Storage

Lecture 12 Managing tablespaces 113

γ Guidelines for managing tablespace
γ Creating tablespaces
γ Managing tablespace allocation
γ Making tablespace read-only

- γ Dropping tablespaces
- γ Viewing information about tablespaces

Lecture 13 Managing Datafiles 124

- γ Guidelines for Managing Datafiles
- γ Creating and Adding Datafiles to a Tablespace
- γ Changing a Datafile's Size
- γ Renaming and Relocating Datafiles
- γ Verifying Data Blocks in Datafiles
- γ Viewing Information About Datafiles

Lecture 14 Managing schema objects 134

- γ Managing Space in Data Blocks
- γ The PCTFREE Parameter
- γ The PCTUSED Parameter
- γ Setting Storage Parameters
- γ Deallocating Space

Lecture 15 Managing partitioned tables and indexes 144

- γ What Are Partitioned Tables and Indexes?
- γ Creating Partitions
- γ Maintaining Partitions.

Lecture 16 Managing rollback segments 154

- γ Guidelines for Managing Rollback Segments
- γ Creating Rollback Segments
- γ Specifying Storage Parameters
- γ Taking Rollback Segments Online and Offline
- γ Dropping Rollback Segments
- γ Monitoring Rollback Segment.

Part 4 Database Security

Lecture 17 Security policy 163

- γ System Security Policy
- γ Data Security Policy
- γ User Security Policy
- γ Password Management Policy
- γ Auditing Policy

Lecture 18 User and resources 175

- γ Session and User Licensing
- γ User Authentication
- γ Oracle Users
- γ Managing Resources with Profiles
- γ Listing Information About Database Users and Profiles

Lecture 19 Privileges and roles 191

- γ Identifying User Privileges
- γ Managing User Roles

γ Granting User Privileges and Roles
γ Revoking User Privileges and Roles

Lecture 20 Auditing 203

γ Introduction to auditing
γ Statement Auditing
γ Privilege Auditing
γ Schema Object Auditing
γ Focusing Statement, Privilege, and Schema Object Auditing

Lecture 21 Backup 213

γ Backups
γ Types of failure
γ Types of backups
γ Backup format

Lecture 22 Recovery 222

γ Introduction to database recovery
γ Errors and failure
γ Structures used for database recovery
γ Rolling forward and rolling back
γ Rollback segments and rolling back
γ Recovery concepts and strategies

Part 5 Oracle Performance Tuning

Lecture 23 What is performance tuning 233

γ Trade-offs Between Response Time and Throughput
γ Critical Resources
γ Effects of Excessive Demand
γ Adjustments to Relieve Problems
γ Steps of tuning method

Lecture 24 Sources of data for tuning 244

γ Sources of data for tuning
γ Oracle enterprise manager application
γ Introduction to Oracle Enterprise Manager
γ Oracle Performance Manager
γ Oracle Top Sessions
γ Oracle Trace
γ Oracle Tablespace Manager
γ Oracle Expert

Lecture 25 Optimization modes and hints **254**

γ Using Cost-Based Optimization
γ Choosing a Goal for the Cost-Based Approach
γ Parameters Which Affect Cost-Based Optimization Plans
γ Using Rule-Based Optimization
γ Introduction to Hints
γ How to Specify Hints

γ Hints for Optimization Approaches and Goals

Lecture 26 Archiving redo information 263

γ Choosing Between NOARCHIVELOG and ARCHIVELOG Mode
γ Turning Archiving On and Off
γ Tuning Archiving
γ Displaying Archiving Status Information
γ Specifying the Archived Redo Log Filename Format and Destination

Lecture 1

The Oracle Database Administrator

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about Types of Oracle Users
- Database Administrator Security and Privileges
- Database Administrator Utilities
- Initial Priorities of a Database Administrator

Coverage Plan

Lecture 1

1.1 Snap Shot

1.2 Types of users

1.3 Database administrator security and privileges

1.4 The DBA role

1.5 Database administrator utilities

1.6 Initial priority of database administrator

1.7 Identifying software releases

1.8 Release number format

1.9 Short Summary

1.10 Brain Storm

1.1 Snap Shot

This chapter describes the responsibilities of the person who administers the Oracle Server, the database administrator. The following topics are included:

- Types of Oracle Users
- Database Administrator Security and Privileges
- Database Administrator Utilities
- Initial Priorities of a Database Administrator
- Identifying Oracle Software Releases

1.2 Types of Oracle Users

At your site, the types of users and their responsibilities may vary. For example, at a large site the duties of a database administrator might be divided among several people.

This section includes the following topics:

- Database Administrators
- Security Officers
- Application Developers
- Application Administrators
- Database Users
- Network Administrators

Database Administrators

Because an Oracle database system can be quite large and have many users, someone or some group of people must manage this system. The *database administrator* (DBA) is this manager. Every database requires at least one person to perform administrative duties.

Security Officers

In some cases, a database might also have one or more security officers. A *security officer* is primarily concerned with enrolling users, controlling and monitoring user access to the database, and maintaining system security. You might not be responsible for these duties if your site has a separate security officer.

Application Developers

An *application developer* designs and implements database applications An application developer's responsibilities include the following tasks:

- Designing and developing the database application
- Designing the database structure for an application
- Estimating storage requirements for an application
- Specifying modifications of the database structure for an application
- Relaying the above information to a database administrator
- Tuning the application during development
- Establishing an application's security measures during development

Application Administrators

An Oracle site might also have one or more application administrators. An *application administrator* is responsible for the administration needs of a particular application.

Database Users

Database users interact with the database via applications or utilities. A typical user's responsibilities include the following tasks:

- entering, modifying, and deleting data, where permitted
- generating reports of data

Network Administrators

At some sites there may be one or more network administrators. Network administrators may be responsible for administering Oracle networking products, such as Net8.

1.3 Database Administrator Security and Privileges

To accomplish administrative tasks in Oracle, you need extra privileges both within the database and possibly in the operating system of the server on which the database runs. Access to a database administrator's account should be tightly controlled.

This section includes the following topics:

- The Database Administrator's Operating System Account
- Database Administrator Usernames
- The DBA Role

The Database Administrator's Operating System Account

To perform many of the administrative duties for a database, you must be able to execute operating system commands. Depending on the operating system that executes Oracle, you might need an operating system account or ID to gain access to the operating system. If so, your operating system account might require more operating system privileges or access rights than many database users require (for example, to perform Oracle software installation). Although you do not need the Oracle files to be stored in your account, you should have access to them.
In addition, Enterprise Manager requires that your operating system account or ID be distinguished in some way to allow you to use *operating system privileged* Enterprise Manager commands.

Database Administrator Usernames

Two user accounts are automatically created with the database and granted the DBA role. These two user accounts are:

- SYS (initial password: CHANGE_ON_INSTALL)
- SYSTEM (initial password: MANAGER)

These two usernames are described in the following sections.

Note:
To prevent inappropriate access to the data dictionary tables, you must change the passwords for the SYS and SYSTEM usernames immediately after creating an Oracle database.

You will probably want to create at least one additional administrator username to use when performing daily administrative tasks.

SYS
When any database is created, the user SYS, identified by the password CHANGE_ON_INSTALL, is automatically created and granted the DBA role.

All of the base tables and views for the database's data dictionary are stored in the schema SYS. These base tables and views are critical for the operation of Oracle. To maintain the integrity of the data dictionary, tables in the SYS schema are manipulated only by Oracle; they should never be modified by any user or database administrator, and no one should create any tables in the schema of the user SYS. (However, you can change the storage parameters of the data dictionary settings if necessary.)

Most database users should never be able to connect using the SYS account. You can connect to the database using this account but should do so only when instructed by Oracle personnel or documentation.

SYSTEM

When a database is created, the user SYSTEM, identified by the password MANAGER, is also automatically created and granted all system privileges for the database.

The SYSTEM username creates additional tables and views that display administrative information, and internal tables and views used by Oracle tools. Never create tables of interest to individual users in the SYSTEM schema.

1.4 The DBA Role

A predefined role, named "DBA", is automatically created with every Oracle database. This role contains all database system privileges. Therefore, it is very powerful and should only be granted to fully functional database administrators.

A database administrator's responsibilities can include the following tasks:

- installing and upgrading the Oracle Server and application tools
- allocating system storage and planning future storage requirements for the database system
- creating primary database storage structures (tablespaces) after application developers have designed an application
- creating primary objects (tables, views, indexes) once application developers have designed an application
- modifying the database structure, as necessary, from information given by application developers
- enrolling users and maintaining system security
- ensuring compliance with your Oracle license agreement
- controlling and monitoring user access to the database
- monitoring and optimizing the performance of the database
- planning for backup and recovery of database information
- maintaining archived data on tape

- backing up and restoring the database
- contacting Oracle Corporation for technical support

To accomplish administrative tasks in ORACLE, you need extra privileges both within the database and possibly in the operating system of the server on which the database runs. This section describes the privileges you need and database security facilities of which you should be aware.

1.5 Database Administrator Utilities

Several utilities are available to help you maintain and control the Oracle Server.

The following topics are included in this section:

- Enterprise Manager
- SQL*Loader
- Export and Import

Enterprise Manager

Enterprise Manager allows you to monitor and control an Oracle database. All administrative operations discussed in this book are executed using Enterprise Manager. Enterprise Manager has both GUI (Graphical User Interface) and line mode interfaces.

Enterprise Manager uses a superset of ANSI/ISO standard SQL commands. The most common administrative commands are available in the menus of Enterprise Manager/GUI. Commands used less frequently can be typed into a Enterprise Manager SQL Worksheet and executed.

SQL*Loader

SQL*Loader is used by both database administrators and users of Oracle. It loads data from standard operating system files (files in text or C data format) into Oracle database tables.

Export and Import

The Export and Import utilities allow you to move existing data in Oracle format to and from Oracle databases. For example, export files can archive database data, or move data among different Oracle databases that run on the same or different operating systems.

1.6 Initial Priorities of a Database Administrator

In general, you must perform a series of steps to get the database system up and running, and then maintain it.

The following steps are required to configure an Oracle Server and database on any type of computer system. The following sections include details about each step.

To Configure an Oracle Server

- Step 1: Install the Oracle Software
- Step 2: Evaluate the Database Server Hardware
- Step 3: Plan the Database
- Step 4: Create and Open the Database

- Step 5: Implement the Database Design
- Step 6: Back up the Database
- Step 7: Enroll System Users
- Step 8: Tune Database Performance

Note: If migrating to a new release, back up your existing production database before installation. For more information on preserving your existing production database, see Oracle8 Server Migration.

Step 1: Install the Oracle Software

As the database administrator, you must install the Oracle Server software and any front-end tools and database applications that access the database. In some distributed processing installations, the database is controlled by a central computer and the database tools and applications are executed on remote machines; in this case, you must also install the Oracle Net8 drivers necessary to connect the remote machines to the computer that executes Oracle.

Step 2: Evaluate the Database Server Hardware

After installation, evaluate how Oracle and its applications can best use the available computer resources. This evaluation should reveal the following information:

- how many disk drives are available to Oracle and its databases
- how many, if any, dedicated tape drives are available to Oracle and its databases
- how much memory is available to the instances of Oracle you will run (See your system's configuration documentation)

Step 3: Plan the Database

As the database administrator, you must plan:

- the database's logical storage structure
- the overall database design
- a backup strategy for the database

It is important to plan how the logical storage structure of the database will affect system performance and various database management operations. For example, before creating any tablespaces for your database, you should know how many data files will make up the tablespace, where the data files will be physically stored (on which disk drives), and what type of information will be stored in each tablespace. When planning the database's overall logical storage structure, take into account the effects that this structure will have when the database is actually created and running. Such considerations include how the database's logical storage structure will affect the following items:

- the performance of the computer executing Oracle
- the performance of the database during data access operations
- the efficiency of backup and recovery procedures for the database

Plan the relational design of the database's objects and the storage characteristics for each of these objects. By planning relationships between objects and the physical storage of each object before creating it, you can directly impact the performance of the database as a unit. Be sure to plan for the growth of the database.

In distributed database environments, this planning stage is extremely important. The physical location of highly accessed data can dramatically affect application performance.

During the above planning phases, also plan a backup strategy for the database. After developing this strategy, you might find that you want to alter the database's planned logical storage structure or database design to improve backup efficiency.

It is beyond the scope of this book to discuss relational and distributed database design; if you are not familiar with such design issues, refer to accepted industry-standard books that explain these studies.

Step 4: Create and Open the Database

Once you have finalized the database design, you can create the database and open it for normal use. Depending on your operating system, a database may already have been created during the installation procedure for Oracle. If so, all you need to do is start an instance and mount and open the initial database.

To determine if your operating system creates an initial database during the installation of Oracle, check your installation or user's guide.

Step 5: Implement the Database Design

Once you have created and started the database, you can create the database's planned logical structure by creating all necessary rollback segments and tablespaces. Once this is built, you can create the objects for your database.

Step 6: Back up the Database

After you have created the database structure, carry out the planned backup strategy for your database by creating any additional redo log files, taking the first full database backup (online or offline), and scheduling future database backups at regular intervals.

Step 7: Enroll System Users

Once you have backed up the database structure, you can begin to enroll the users of the database in accordance with your Oracle license agreement, create roles for these users, and grant appropriate roles to them.

Step 8: Tune Database Performance

Optimizing the database system's performance is one of your ongoing responsibilities.

1.7 Identifying Oracle Software Releases

Because Oracle products are always undergoing development and change, several releases of the products can be in use at any one time. To identify a software product fully, as many as five numbers may be required.
This section includes the following topics:

- Release Number Format
- Versions of Other Oracle Software
- Checking Your Current Release Number

1.8 Release Number Format

An Oracle Server distribution tape might be labeled "Release 8.0.4.1." The following sections translate this number.

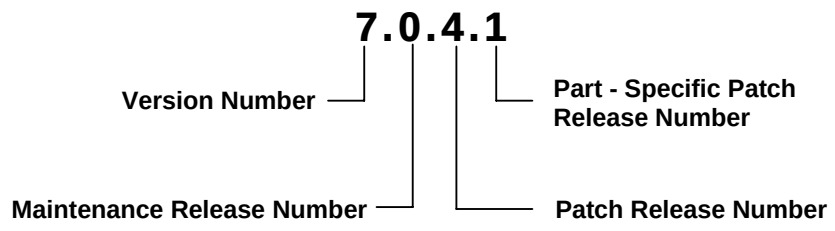


Figure: Example of an Oracle Release Number

Version Number

The version number, such as 8, is the most general identifier. A *version* is a major new edition of the software, which usually contains significant new functionality.

Maintenance Release Number

The maintenance release number signifies different releases of the general version, starting with 0, as in version 8.0. The maintenance release number increases when bug fixes or new features to existing programs become available.

Patch Release Number

The patch release number identifies a specific level of the object code, such as 8.0.4. A patch release contains fixes for serious bugs that cannot wait until the next maintenance release. The first distribution of a maintenance release always has a patch number of 0.

Port-Specific Patch Release Number

A fourth number (and sometimes a fifth number) can be used to identify a particular emergency patch release of a software product on that operating system, such as 8.0.4.1. or 8.0.4.1.3. An emergency patch is not usually intended for wide distribution; it usually fixes or works around a particular, critical problem.

Examples of Release Numbers

The following examples show possible release numbers for Oracle8:

8.0.0	the first distribution of Oracle8
8.1.0	the first maintenance release of Oracle8
8.2.0	the second maintenance release (the third release in all) of Oracle8
8.2.2	the second patch release after the second maintenance release

Versions of Other Oracle Software

As Oracle Corporation introduces new products and enhances existing ones, the version numbers of the individual products increment independently. Thus, you might have an Oracle Server Release 8.0.12.2 system working with Oracle Forms Version 4.0.3, SQL*Plus Version 3.1.9, and Pro*FORTRAN Version 1.5.2. (These numbers are used only for illustration.)

Checking Your Current Release Number

To see which release of Oracle and its components you are using, query the data dictionary view PRODUCT_COMPONENT_VERSION, as shown below (This information is useful if you need to call Oracle Support.):

SVRMGR> SELECT * FROM product_component_version;

PRODUCT	VERSION	STATUS
CORE	3.4.1.0.0	Production
NLSRTL	3.1.3.0.0	Production
Oracle8 Server	3.2.1.0.0	Beta Release
PL/SQL	2.2.1.0.0	Beta
TNS for SunOS:	2.1.4.0.0	Production

5 rows selected.

1.9 Short Summary

In this section we learn about types of users, they are database administrator, Security Officers, Application Developers, Application Administrators, Database Users, Network Administrators.

Two user accounts are automatically created with the database and granted the DBA role. These two user accounts are: SYS, SYSTEM.

1.10 Brain Storm

1. State the Roles and Responsibilities of DBA?
2. Differentiate between SGA and PGA?
3. Enunciate the difference between Shared SQL Area and Private SQL Area?
4. Enunciate the difference Multi – Threaded Server and Parallel Server?
5. What are the Oracle 8.0 specific features?
6. State the necessity of Background Process?

Lecture 2

Database Administrator Authentication

Objectives

After completing this lesson, you should be able to do the following:

- Selecting an Authentication method
- Using operating system authentication and password file
- Using ORAPWD & Connecting with administrator privileges
- Setting REMOTE_LOGIN_PASSWORDFILE
- Adding users to password file and maintaining password file

Coverage Plan

Lecture 2

2.1 Snap shot

2.2 Database administrator authentication

2.3 Using operating system authentication

2.4 OSOPER and OSDBA

2.5 Using an authentication password file

2.6 Password file administration

2.7 Short summary

2.8 Brain Storm

2.1 Snap Shot

In this section we learn about authentication and protection of database. Maintain the password file. This section includes the following topics:

- Selecting an Authentication Method
- Using Operating System Authentication
- OSOPER and OSDBA
- Using an Authentication Password File

2.2 Database Administrator Authentication

Database administrators must often perform special operations such as shutting down or starting up a database. Because normal database users should not perform these operations, the database administrator usernames need a more secure authentication scheme.

Selecting an Authentication Method

The following methods for authenticating database administrators replace the CONNECT INTERNAL syntax provided with earlier versions of Oracle (CONNECT INTERNAL continues to be supported for backwards compatibility only):

- Operating system authentication
- Password files

Depending on whether you wish to administer your database locally on the same machine where the database resides or to administer many different databases from a single remote client, you can choose between operating system authentication and password files to authenticate database administrators. Figure 2-1 illustrates the choices you have for database administrator authentication schemes.

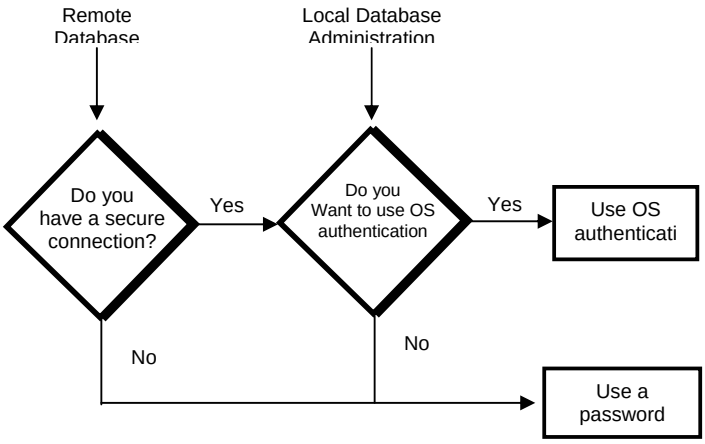


Figure 2-1: Database Administrator Authentication Methods

On most operating systems, OS authentication for database administrators involves placing the OS username of the database administrator in a special group (on UNIX systems, this is the DBA group) or giving that OS username a special process right.

The database uses password files to keep track of database usernames that have been granted administrator privileges.

2.3 Using Operating System Authentication

If you choose, you can have your operating system authenticate users performing database administration operations.

1. Set up the user to be authenticated by the operating system.

2 Make sure that the initialization parameter, REMOTE_LOGIN_PASSWORD, is set to NONE, which is the default value for this parameter.

3 Authenticated users should now be able to connect to a local database, or to connect to a remote database over a secure connection, by typing one of the following commands:
CONNECT / AS SYSOPER
CONNECT / AS SYSDBA

If you successfully connect as INTERNAL using an earlier release of Oracle, you should be able to continue to connect successfully using the new syntax shown in Step 3.

Note: Note that to connect as SYSOPER or SYSDBA using OS authentication you do not have to have been granted the SYSOPER or SYSDBA system privileges. Instead, the server verifies that you have been granted the appropriate OSDBA or OSOPER roles at the operating system level.

2.4 OSOPER and OSDBA

Two special operating system roles control database administrator logins when using operating system authentication: OSOPER and OSDBA.

OSOPER	Permits the user to perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, and RECOVER, and includes the RESTRICTED SESSION privilege.
OSDBA	Contains all system privileges with ADMIN OPTION, and the OSOPER role; permits CREATE DATABASE and time-based recovery.

OSOPER and OSDBA can have different names and functionality, depending on your operating system.
The OSOPER and OSDBA roles can only be granted to a user through the operating system. They cannot be granted through a GRANT statement, nor can they be revoked or dropped. When a user logs on with administrator privileges and REMOTE_LOGIN_PASSWORD FILE is set to NONE, Oracle communicates with the operating system and attempts to enable first OSDBA and then, if unsuccessful, OSOPER. If both attempts fail, the connection fails. How you grant these privileges through the operating system is operating system specific.

If you are performing remote database administration, you should consult your Net8 documentation to determine if you are using a secure connection. Most popular connection protocols, such as TCP/IP and DECnet, are not secure, regardless of which version of Net8 you are using.

2.5 Using an Authentication Password File

If you have determined that you need to use a password file to authenticate users performing database administration, you must complete the steps outlined below. Each of these steps is explained in more detail in the following sections of this chapter.

1. Create the password file using the ORAPWD utility.
   ORAPWD FILE=*filename* PASSWORD=*password* ENTRIES=*max_users*

2. Set the REMOTE_LOGIN_PASSWORDFILE initialization parameter to EXCLUSIVE.

3. Add users to the password file by using SQL to grant the appropriate privileges to each user who needs to perform database administration, as shown in the following examples.
   GRANT SYSDBA TO scott
   GRANT SYSOPER TO scott

The privilege SYSDBA permits the user to perform the same operations as OSDBA. Likewise, the privilege SYSOPER permits the user to perform the same operations as OSOPER.

Privileged users should now be able to connect to the database by using a command similar to the one shown below. CONNECT scott/tiger@acct.hq.com AS SYSDBA

2.6 Password File Administration

You can create a password file using the password file creation utility, ORAPWD or, for selected operating systems, you can create this file as part of your standard installation.
This section includes the following topics:

- Using ORAPWD
- Setting REMOTE_LOGIN_ PASSWORDFILE
- Adding Users to a Password File
- Connecting with Administrator Privileges
- Maintaining a Password File

Using ORAPWD

When you invoke the password file creation utility without supplying any parameters, you receive a message indicating the proper use of the command as shown in the following sample output:

```
> orapwd
Usage: orapwd file=<fname> password=<password> entries=<users>
    where
        file - name of password file (mand),
        password - password for SYS and INTERNAL (mand),
        entries - maximum number of distinct DBAs and OPERs (opt),
```

There are no spaces around the equal-to (=) character.

For example, the following command creates a password file named ACCT.PWD that allows up to 30 privileged users with different passwords. The file is initially created with the password SECRET for users connecting as INTERNAL or SYS:

ORAPWD FILE=acct.pwd PASSWORD=secret ENTRIES=30
Following are descriptions of the parameters in the ORAPWD utility.

FILE This parameter sets the name of the password file being created. You must specify the full pathname for the file. The contents of this file are encrypted, and the file is not user-readable. This parameter is mandatory.

The types of file names allowed for the password file are operating system specific. Some platforms require the password file to be a specific format (for example, orapw <SID>) and located in a specific directory. Other platforms allow the use of environment variables to specify the name and location of the password file. refer your operating system-specific Oracle documentation for the names and locations allowed on your platform.

If you are running multiple instances of Oracle using the Oracle Parallel Server, the environment variable for each instance should point to the same password file.

Warning: It is critically important to the security of your system that you protect your password file and environment variables that identify the location of the password file. Any user with access to these could potentially compromise the security of the connection.

PASSWORD This parameter sets the password for INTERNAL and SYS. If you issue the ALTER USER command to change the password after connecting to the database, both the password stored in the data dictionary and the password stored in the password file are updated. The INTERNAL user is supported for backwards compatibility only. This parameter is mandatory.

ENTRIES This parameter sets the maximum number of entries allowed in the password file. This corresponds to the maximum number of distinct users allowed to connect to the database as SYSDBA or SYSOPER. Entries can be reused as users are added to and removed from the password file. This parameter is required if you ever want this password file to be EXCLUSIVE.

Warning: If you ever need to exceed this limit, you must create a new password file. It is safest to select a number larger than you think you will ever need.

Setting REMOTE_LOGIN_ PASSWORDFILE

In addition to creating the password file, you must also set the initialization parameter REMOTE_LOGIN_PASSWORDFILE to the appropriate value. The values recognized are described below.

Note: To start up an instance or database, you must use Enterprise Manager. You must specify a database name and a parameter file to initialize the instance settings. You may specify a fully-qualified remote database name using Net8. However, the initialization parameter file and any associated files, such as a configuration file, must exist on the client machine. That is, the parameter file must be on the machine where you are running Enterprise Manager.

NONE Setting this parameter to NONE causes Oracle to behave as if the password file does not exist. That is, no privileged connections are allowed over non-secure connections. NONE is the default value for this parameter.

EXCLUSIVE An EXCLUSIVE password file can be used with only one database. Only an EXCLUSIVE file can contain the names of users other than SYS and INTERNAL. Using an EXCLUSIVE password file allows you to grant SYSDBA and SYSOPER system privileges to individual users and have them connect as themselves.

SHARED A SHARED password file can be used by multiple databases. However, the only users recognized by a SHARED password file are SYS and INTERNAL; you cannot add users to a SHARED password file. All users needing SYSDBA or SYSOPER system privileges must connect using the same name, SYS, and password. This option is useful if you have a single DBA administering multiple databases.

Suggestion: To achieve the greatest level of security, you should set the REMOTE_LOGIN_PASSWORDFILE file initialization parameter to EXCLUSIVE immediately after creating the password file.

Adding Users to a Password File

When you grant SYSDBA or SYSOPER privileges to a user, that user's name and privilege information is added to the password file. If the server does not have an EXCLUSIVE password file, that is, if the initialization parameter REMOTE_LOGIN_PASSWORDFILE is NONE or SHARED, you receive an error message if you attempt to grant these privileges.
A user's name only remains in the password file while that user has at least one of these two privileges. When you revoke the last of these privileges from a user, that user is removed from the password file.

To Create a Password File and Add New Users to It

1. Follow the instructions for creating a password file.
2. Set the REMOTE_LOGIN_PASSWORDFILE initialization parameter to EXCLUSIVE.
3. Connect with SYSDBA privileges as shown in the following example:
   CONNECT SYS/change_on_install AS SYSDBA
4. Start up the instance and create the database if necessary, or mount and open an existing database.
5. Create users as necessary. Grant SYSOPER or SYSDBA privileges to yourself and other users as appropriate.
6. These users are now added to the password file and can connect to the database as SYSOPER or SYSDBA with a username and password (instead of using SYS). The use of a password file does not prevent OS authenticated users from connecting if they meet the criteria for OS authentication.

Granting and Revoking SYSOPER and SYSDBA Privileges

If your server is using an EXCLUSIVE password file, use the GRANT command to grant the SYSDBA or SYSOPER system privilege to a user, as shown in the following example:

GRANT SYSDBA TO scott

Use the REVOKE command to revoke the SYSDBA or SYSOPER system privilege from a user, as shown in the following example:

REVOKE SYSDBA FROM scott

Because SYSDBA and SYSOPER are the most powerful database privileges, the ADMIN OPTION is not used. Only users currently connected as SYSDBA (or INTERNAL) can grant SYSDBA or SYSOPER system privileges to another user. This is also true of REVOKE. These privileges cannot be granted to roles, since roles are only available after database startup. Do not confuse the SYSDBA and SYSOPER database privileges with operating system roles, which are a completely independent feature.

Listing Password File Members

Use the V$PWFILE_USERS view to determine which users have been granted SYSDBA and SYSOPER system privileges for a database. The columns displayed by this view are as follows:

USERNAME
The name of the user that is recognized by the password file.
SYSDBA
If the value of this column is TRUE, the user can log on with SYSDBA system privileges.

SYSOPER
If the value of this column is TRUE, the user can log on with SYSOPER system privileges.

Connecting with Administrator Privileges

When you connect with SYSOPER or SYSDBA privileges using a username and password, you are connecting with a default schema of SYS, not the schema that is generally associated with your username.

Use the AS SYSDBA or AS SYSOPER clauses of the Enterprise Manager CONNECT command to connect with administrator privileges.

Connecting with Administrator Privileges: Example
For example, assume user SCOTT has issued the following commands:
CONNECT scott/tiger

```
CREATE TABLE scott_test(name VARCHAR2(20));
Later, when SCOTT issues these commands:
CONNECT scott/tiger AS SYSDBA

SELECT * FROM scott_test;
```

He receives an error that SCOTT_TEST does not exist. That is because SCOTT now references the SYS schema by default, whereas the table was created in the SCOTT schema.
Non-Secure Remote Connections

To connect to Oracle as a privileged user over a non-secure connection, you must meet the following conditions:

- The server to which you are connecting must have a password file.
- You must be granted the SYSOPER or SYSDBA system privilege.
- You must connect using a username and password.

Local and Secure Remote Connections

To connect to Oracle as a privileged user over a local or a secure remote connection, you must meet either of the following sets of conditions:

- You can connect using a password file, provided that you meet the criteria outlined for non-secure connections in the previous bulleted list.
- If the server is not using a password file, or you have not been granted SYSOPER or SYSDBA privileges and are therefore not in the password file, your operating system name must be authenticated for a privileged connection by the operating system. This form of authentication is operating system specific.

Consult your operating system-specific Oracle documentation for details on operating system authentication.

Maintaining a Password File

This section describes how to expand, relocate, and remove the password file, as well as how to avoid changing the state of the password file.

Expanding the Number of Password File Users

If you receive the file full error (ORA-1996) when you try to grant SYSDBA or SYSOPER system privileges to a user, you must create a larger password file and re-grant the privileges to the users.

To Replace a Password File

1. Note which users have SYSDBA or SYSOPER privileges by querying the V$PWFILE_USERS view.
2. Shut down the database.
3. Delete the existing password file.
4. Follow the instructions for creating a new password file using the ORAPWD utility in "Using ORAPWD" on page 1-10. Be sure to set the ENTRIES parameter to a sufficiently large number.
5. Follow the instructions in "Adding Users to a Password File" on page 1-12.

Relocating the Password File

After you have created the password file, you can relocate it as you choose. After relocating the password file, you must reset the appropriate environment variables to the new pathname. If your operating system uses a predefined pathname, you cannot change the password file location.

Removing a Password File

If you determine that you no longer need to use a password file to authenticate users, you can delete the password file and reset the REMOTE_LOGIN_PASSWORDFILE initialization parameter to NONE. After removing this file, only users who can be authenticated by the operating system can perform database administration operations.

Warning: Do not remove or modify the password file if you have a database or instance mounted using REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE (or SHARED). If you do, you will be unable to reconnect remotely using the password file. Even if you replace it, you cannot use the new password file, because the timestamp and checksums will be wrong.

Changing the Password File State

The password file state is stored in the password file. When you first create a password file, its default state is SHARED. You can change the state of the password file by setting the parameter REMOTE_LOGIN_PASSWORDFILE. When you STARTUP an instance, Oracle retrieves the value of this parameter from the initialization parameter file stored on your client machine. When you mount the database, Oracle compares the value of this parameter to the value stored in the password file. If these values do not match, the value stored in the file is overwritten.

Warning: You should use caution to ensure that an EXCLUSIVE password file is not accidentally changed to SHARED. If you plan to allow instance STARTUP from multiple clients, each of those clients must have an initialization parameter file, and the value of the parameter REMOTE_LOGIN_PASSWORDFILE must be the same in each of these files. Otherwise, the state of the password file could change depending upon where the instance was started.

2.7 Short summary

Permits the user to perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, and RECOVER, and includes the RESTRICTED SESSION privilege.

Contains all system privileges with ADMIN OPTION, and the OSOPER role; permits CREATE DATABASE and time-based recovery.

2.8 Brain storm

1. The SYSDBA privilege administers some of the following privileges also, they are

   a) The system privilege b) All system privileges granted with admin option
   c) The create Database privilege d) All of the above

2. Within the CATALOG.SQL script there are calls to other scripts, they are:

   a. CATPROC.SQL b. CATLDR.SQL c. CATEXP.SQL d. CATAUDIT.SQL
   e. Tick all that applies

3. The background process that coalesces small blocks of free space into larger blocks of free space is
   a. DBWR b. LGWR c. SMON d. PMON

4. The PROCESS parameter in INIT ORA files include Background Process and User Process
   a. True b. False

Lecture 3

The Oracle Server

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about databases and information management
- Discuss about what is oracle server and explain about oracle databases
- Discuss about memory structure and processes
- Example for how oracle works
- Explain about oracle instance

Coverage Plan

Lecture 3

3.1 Snap shot

3.2 The oracle server

3.3 Oracle databases

3.4 Memory structure and processes

3.5 Memory structures

3.6 Process architecture

3.7 The program interface

3.8 An example of how oracle works

3.9 Short summary

3.10 Brain storm

3.1 Snap Shot

A database server is the key to solving the problems of information management. In general, a server must reliably manage a large amount of data in a multi-user environment so that many users can concurrently access the same data. All this must be accomplished while delivering high performance. A database server must also prevent unauthorized access and provide efficient solutions for failure recovery.

3.2 The Oracle Server

The Oracle Server is an object-relational database management system that provides an open, comprehensive, and integrated approach to information management. An Oracle Server consists of an Oracle database and an Oracle Server instance. The following sections describe the relationship between the database and the instance.

Structured Query Language (SQL)

SQL (pronounced SEQUEL) is the programming language that defines and manipulates the database. SQL databases are relational databases; this means simply that data is stored in a set of simple relations. A database can have one or more tables. And each table has columns and rows. A table that has an employee database, for example, might have a column called employee number and each row in that column would be an employee's employee number.

You can define and manipulate data in a table with SQL commands. You use data definition language (DDL) commands to set up the data. DDL commands include commands to creating and altering databases and tables.

You can update, delete, or retrieve data in a table with data manipulation commands (DML). DML commands include commands to alter and fetch data. The most common SQL command is the SELECT command, which allows you to retrieve data from the database.

In addition to SQL commands, the Oracle Server has a procedural language called PL/SQL. PL/SQL enables the programmer to program SQL statements. It allows you to control the flow of a SQL program, to use variables, and to write error-handling procedures.

Database Structure

An Oracle database has both a physical and a logical structure. Because the physical and logical server structure is separate, the physical storage of data can be managed without affecting the access to logical storage structures.

Physical Database Structure

The operating system files that constitute the database determine an Oracle database's physical structure. Each Oracle database is made of three types of files: one or more datafiles, two or more redo log files, and one or more control files. The files of an Oracle database provide the actual physical storage for database information.

Logical Database Structure

An Oracle database's logical structure is determined by:

- one or more tablespaces. (A tablespace is a logical area of storage explained later in this chapter.)
- the database's schema objects. A *schema* is a collection of objects. *Schema objects* are the logical structures that directly refer to the database's data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links.

The logical storage structures, including tablespaces, segments, and extents, dictate how the physical space of a database is used. The schema objects and the relationships among them form the relational design of a database.

An Oracle Instance

Every time a database is started, a system global area (SGA) is allocated and Oracle background processes are started. The system global area is an area of memory used for database information shared by the database users. The combination of the background processes and memory buffers is called an Oracle *instance.*

An Oracle instance has two types of processes: user processes and Oracle processes.

- A *user process* executes the code of an application program (such as an Oracle Forms application) or an Oracle Tool (such as Enterprise Manager).

- *Oracle processes* are server processes that perform work for the user processes and background processes that perform maintenance work for the Oracle Server.

Communications Software and Net8

If the user and server processes are on different computers of a network or if the user processes connect to shared server processes through dispatcher processes, the user process and server process communicate using Net8. *Dispatchers* are optional background processes, present only when a multi-threaded server configuration is used. *Net8* is Oracle's interface to standard communications protocols that allows for the proper transmission of data between computers.

The Oracle Parallel Server: Multiple Instance Systems

Some hardware architectures (for example, shared disk systems) allow multiple computers to share access to data, software, or peripheral devices. Oracle with the Parallel Server option can take advantage of such architecture by running multiple instances that "share" a single physical database. In appropriate applications, the Oracle Parallel Server allows access to a single database by the users on multiple machines with increased performance.

Figure 3-1 illustrates a multiple-process Oracle instance.

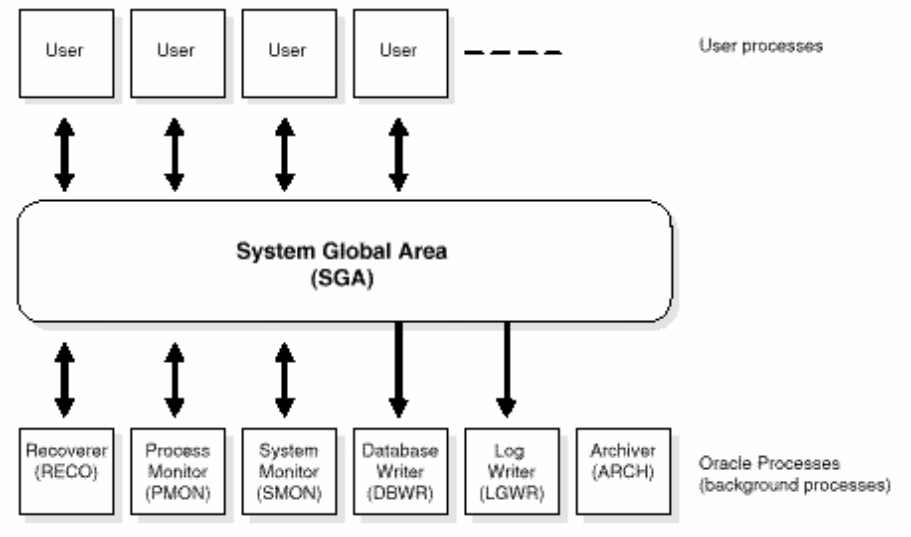


Figure 3-1: An Oracle Instance

3.3 Oracle Databases

An Oracle *database* is a collection of data that is treated as a unit. The general purpose of a database is to store and retrieve related information.

The database has *logical structures* and *physical structures*.

Open and Closed Databases

An Oracle database can be *open* (accessible) or *closed* (not accessible). In normal situations, the database is open and available for use. However, the database is sometimes closed for specific administrative functions that require the database's data to be unavailable to users.

3.4 Memory Structure and Processes

This section discusses the memory and process structures used by an Oracle Server to manage a database. Among other things, the architectural features discussed in this section provide an understanding of the capabilities of the Oracle Server to support:

- many users concurrently accessing a single database
- the high performance required by concurrent multi-user, multi-application database systems

An Oracle Server uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of the computers that constitute the database system. *Processes* are jobs or tasks that work in the memory of these computers.

3.5 Memory Structures

Oracle creates and uses memory structures to complete several jobs. For example, memory is used to store program code being executed and data that is shared among users. Several basic memory structures are associated with Oracle: the system global area (which includes the database buffers, redo log buffers, and the shared pool) and the program global areas. The following subsections explain each in detail.

System Global Area (SGA)

The *System Global Area (SGA)* is a shared memory region that contains data and control information for one Oracle instance. An SGA and the Oracle background processes constitute an Oracle instance. Oracle allocates the system global area when an instance starts and deallocates it when the instance shuts down. Each instance has its own system global area.

Users currently connected to an Oracle Server share the data in the system global area. For optimal performance, the entire system global area should be as large as possible (while still fitting in real memory) to store as much data in memory as possible and minimize disk I/O.

The information stored within the system global area is divided into several types of memory structures, including the database buffers, redo log buffer, and the shared pool. These areas have fixed sizes and are created during instance startup.

Database Buffer Cache

Database buffers of the system global area store the most recently used blocks of database data; the set of database buffers in an instance is the *database buffer cache*. The buffer cache contains modified as well as unmodified blocks. Because the most recently (and often the most frequently) used data is kept in memory, less disk I/O is necessary and performance is improved.

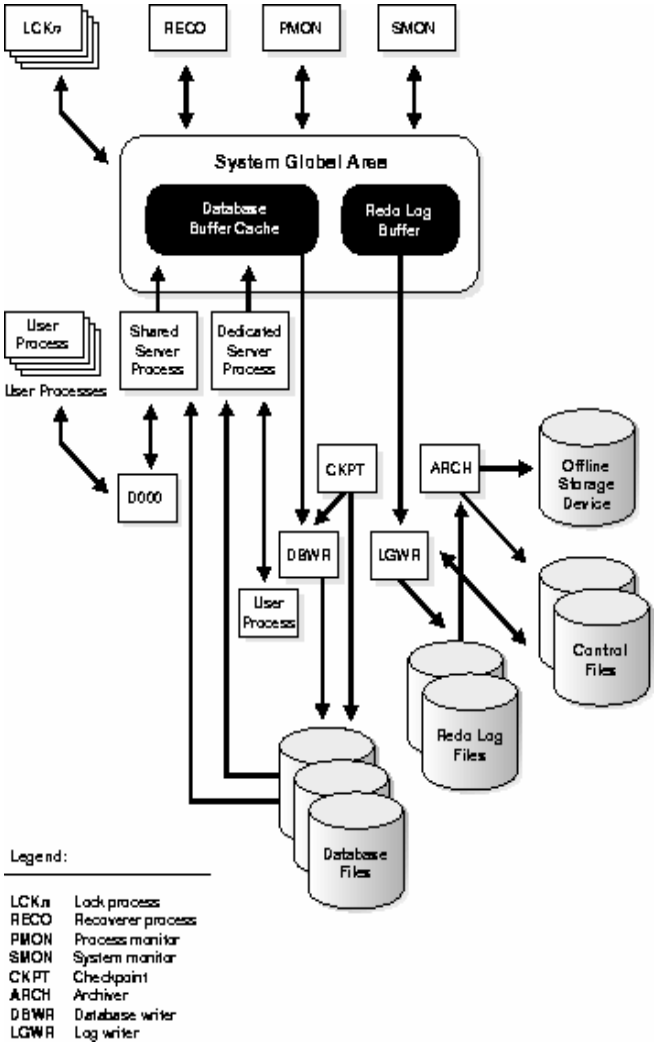


Figure 3.2: Memory Structures and Processes of Oracle

Redo Log Buffer

The *redo log buffer* of the system global area stores *redo entries* - a log of changes made to the database. The redo entries stored in the redo log buffers are written to an online redo log file, which is used if database recovery is necessary. Its size is static.

Shared Pool

The shared pool is a portion of the system global area that contains shared memory constructs such as shared SQL areas. A shared SQL area is required to process every unique SQL statement submitted to a database. A shared SQL area contains information such as the parse tree and execution plan for the corresponding statement. A single shared SQL area is used by multiple applications that issue the same statement, leaving more shared memory for other uses.

Statement Handles or Cursors

A *cursor* is a handle (a name or pointer) for the memory associated with a specific statement. (The Oracle Call Interface, OCI, refers to these as *statement handles*.) Although most Oracle users rely on the automatic cursor

handling of the Oracle utilities, the programmatic interfaces offer application designers more control over cursors.

For example, in precompiler application development, a cursor is a named resource available to a program and can be specifically used for the parsing of SQL statements embedded within the application. The application developer can code an application so that it controls the phases of SQL statement execution and thus improve application performance.

Program Global Area (PGA)

The *Program Global Area (PGA)* is a memory buffer that contains data and control information for a server process. A PGA is created by Oracle when a server process is started. The information in a PGA depends on the configuration of Oracle.

3.6 Process Architecture

A *process* is a "thread of control" or a mechanism in an operating system that can execute a series of steps. Some operating systems use the terms *job* or *task*. A process normally has its own private memory area in which it runs.

An Oracle Server has two general types of processes: user processes and Oracle processes.

User (Client) Processes

A *user process* is created and maintained to execute the software code of an application program (such as a Pro*C/C++ program) or an Oracle tool (such as Enterprise Manager). The user process also manages the communication with the server processes.

User processes communicate with the server processes through the program interface, which is described in a later section.

Oracle Process Architecture

Other processes to perform functions on behalf of the invoking process call oracle processes. The different types of Oracle processes and their specific functions are discussed in the following sections. They include server processes and background processes.

Server Processes

Oracle creates *server processes* to handle requests from connected user processes. A server process is in charge of communicating with the user process and interacting with Oracle to carry out requests of the associated user process. For example, if a user queries some data that is not already in the database buffers of the system global area, the associated server process reads the proper data blocks from the datafiles into the system global area.

Oracle can be configured to vary the number of user processes per server process. In a *dedicated server configuration*, a server process handles requests for a single user process. A *multi-threaded server configuration* allows many user processes to share a small number of server processes, minimizing the number of server processes and maximizing the utilization of available system resources.

On some systems, the user and server processes are separate, while on others they are combined into a single process. If a system uses the multi-threaded server or if the user and server processes run on different machines, the user and server processes must be separate. Client/server systems separate the user and server processes and execute them on different machines.

Background Processes

Oracle creates a set of *background processes* for each instance. They consolidate functions that would otherwise be handled by multiple Oracle programs running for each user process. The background processes asynchronously perform I/O and monitor other Oracle processes to provide increased parallelism for better performance and reliability.

An SGA and the set of Oracle background processes constitute an Oracle instance. Each Oracle instance may use several background processes. The names of these processes are DBWR, LGWR, CKPT, SMON, PMON, ARCH, RECO, D*nnn*, and LCK*n*.

Database Writer (DBWR)

The *Database Writer* writes modified blocks from the database buffer cache to the datafiles. Since Oracle uses write-ahead logging, DBWR does not need to write blocks when a transaction commits. Instead, DBWR is designed to perform batched writes with high efficiency. In the most common case, DBWR writes only when more data needs to be read into the system global area and too few database buffers are free. The least recently used data is written to the datafiles first. DBWR also performs writes for other functions such as checkpoint.

Log Writer (LGWR)

The *Log Writer* writes redo log entries to disk. Redo log data is generated in the redo log buffer of the system global area. As transactions commit and the log buffer fills, LGWR writes redo log entries into an online redo log file.

Checkpoint (CKPT)

At specific times, all modified database buffers in the system global area are written to the datafiles by DBWR; this event is called a checkpoint. The *Checkpoint* process is responsible for signalling DBWR at checkpoints and updating all the datafiles and control files of the database to indicate the most recent checkpoint.

System Monitor (SMON)

The *system monitor* performs instance recovery at instance startup. In a multiple instance system (one that uses the Parallel Server), SMON of one instance can also perform instance recovery for other instances that have failed. SMON also cleans up temporary segments that are no longer in use and recovers dead transactions skipped during crash and instance recovery because of file-read or offline errors. These transactions are eventually recovered by SMON when the tablespace or file is brought back online. SMON also coalesces free extents within the database to make free space contiguous and easier to allocate.

Process Monitor (PMON)

The *process monitor* performs process recovery when a user process fails. PMON is responsible for cleaning up the cache and freeing resources that the process was using. PMON also checks on dispatcher (see below) and server processes and restarts them if they have failed.

Archiver (ARCH)

The *archiver* copies the online redo log files to archival storage when they are full. ARCH is active only when a database's redo log is used in ARCHIVELOG mode.

Recoverer (RECO)

The *recoverer* is used to resolve distributed transactions that are pending due to a network or system failure in a distributed database. At timed intervals, the local RECO attempts to connect to remote databases and automatically complete the commit or rollback of the local portion of any pending distributed transactions.

Dispatcher (Dnnn)

Dispatchers are optional background processes, present only when a multi-threaded server configuration is used. At least one dispatcher process is created for every communication protocol in use (D000, . . ., D*nnn*). Each dispatcher process is responsible for routing requests from connected user processes to available shared server processes and returning the responses back to the appropriate user processes.

Lock (LCK*n*) : The *lock* processes (LCK0, . . ., LCK9) are used for inter-instance locking in the Oracle Parallel Server,for more information about this configuration.

3.7 The Program Interface

The *program interface* is the mechanism by which a user process communicates with a server process. It serves as a method of standard communication between any client tool or application (such as Oracle Forms) and Oracle software. Its functions are to:

- act as a communications mechanism, by formatting data requests, passing data, and trapping and returning errors
- perform conversions and translations of data, particularly between different types of computers or to external user program datatypes

3.8 An Example of How Oracle Works

The following example illustrates an Oracle configuration where the user and associated server process are on separate machines (connected via a network).

An instance is currently running on the computer that is executing Oracle (often called the *host* or *database server*).

1. A computer running an application (a *local machine* or *client workstation*) runs the application in a user process. The client application attempts to establish a connection to the server using the proper Net8 driver.

2. The server is running the proper Net8 driver. The server detects the connection request from the application and creates a (dedicated) server process on behalf of the user process.
3. The user executes a SQL statement and commits the transaction. For example, the user changes a name in a row of a table.

4. The server process receives the statement and checks the shared pool for any shared SQL area that contains an identical SQL statement. If a shared SQL area is found, the server process checks the user's access privileges to the requested data and the previously existing shared SQL area is used to process the statement; if not, a new shared SQL area is allocated for the statement so that it can be parsed and processed.

5. The server process retrieves any necessary data values from the actual datafile (table) or those stored in the system global area.

6. The server process modifies data in the system global area. The DBWR process writes modified blocks permanently to disk when doing so is efficient. Because the transaction committed, the LGWR process immediately records the transaction in the online redo log file.

7. If the transaction is successful, the server process sends a message across the network to the application. If it is not successful, an appropriate error message is transmitted.

8. Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention

between transactions that request the same data. These steps describe only the most basic level of operations that Oracle performs.

3.9 Short Summary

An *Oracle instance* has two types of processes: user processes and Oracle processes. A *user process* executes the code of an application program (such as an Oracle Forms application) or an Oracle Tool (such as Enterprise Manager). Oracle processes are *server processes* that perform work for the user processes and background processes that perform maintenance work for the Oracle Server.
The *program interface* is the mechanism by which a user process communicates with a server process.

The *System Global Area (SGA)* is a shared memory region that contains data and control information for one Oracle instance.

3.10 Brain storm

1. What is Trusted Oracle?

2. Nested tables is used to avoid join condition
   a) True b) False

3. Partition will occupy only one segment
   a) Trueb b) False

4. Differentiate between a Table and Synonym?

Lecture 4

Database Structure and Space Management

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about databases structure and space management
- Discuss about logical database structure
- Discuss about data blocks, extents and segments
- Describe about physical database structure
- Explain about datafiles, redo log files and control files

Coverage Plan

Lecture 4

4.1 Snap Shot

4.2 Database Structure and Space Management

4.3 Logical Database Structure

4.4 Data Blocks

4.5 Extents

4.6 Segments

4.7 Physical Database Structure

4.8 Datafiles

4.9 Redo Log Files

4.10 Control Files

4.11 Short Summary

4.12 Brain Storm

4.1 Snap Shot

This section describes the architecture of an Oracle database, including the physical and logical structures that make up a database. This discussion provides an understanding of Oracle's solutions to controlled data availability, the separation of logical and physical data structures, and fine-grained control of disk space management.

4.2 Database Structure and Space Management

An Oracle database has both a physical and a logical structure. Because the physical and logical server structure is separate, the physical storage of data can be managed without affecting the access to logical storage structures.

An Oracle *database* is a collection of data that is treated as a unit. The general purpose of a database is to store and retrieve related information. The database has *logical structures* and *physical structures*.

4.3 Logical Database Structures

The following sections explain logical database structures, including tablespaces, schema objects, data blocks, extents, and segments.

Tablespaces

A database is divided into logical storage units called *tablespaces*. A tablespace is used to group related logical structures together. For example, tablespaces commonly group all of an application's objects to simplify certain administrative operations.

Databases, Tablespaces, and Datafiles

The relationship among databases, tablespaces, and datafiles (datafiles are described in the next section) is illustrated in Figure 4.1.

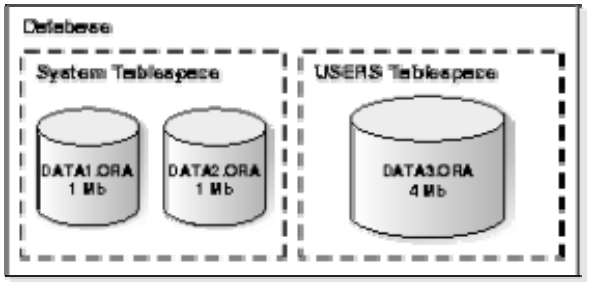


Figure 4-1: Databases, Tablespaces, and Datafiles
This figure illustrates the following:

- Each database is logically divided into one or more tablespaces.
- One or more datafiles are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace.
- The combined size of a tablespace's datafiles is the total storage capacity of the tablespace (SYSTEM tablespace has 2 MB storage capacity while USERS tablespace has 4 MB).

- The combined storage capacity of a database's tablespaces is the total storage capacity of the database (6 MB).

Online and Offline Tablespaces

A tablespace can be *online* (accessible) or *offline* (not accessible). A tablespace is normally online so that users can access the information within the tablespace. However, sometimes a tablespace may be taken offline to make a portion of the database unavailable while allowing normal access to the remainder of the database. This makes many administrative tasks easier to perform.

Schemas and Schema Objects

A *schema* is a collection of objects. *Schema objects* are the logical structures that directly refer to the database's data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. (There is no relationship between a tablespace and a schema; objects in the same schema can be in different tablespaces, and a tablespace can hold objects from different schemas.)

Data Blocks, Extents, and Segments

Oracle allows fine-grained control of disk space usage through the logical storage structures, including data blocks, extents, and segments.

Oracle Data Blocks

At the finest level of granularity, an Oracle database's data is stored in *data blocks*. One data block corresponds to a specific number of bytes of physical database space on disk. A data block size is specified for each Oracle database when the database is created. A database uses and allocates free database space in Oracle data blocks.

Extents

The next level of logical database space is called an extent. An *extent* is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

Segments

The level of logical database storage above an extent is called a segment. A *segment* is a set of extents allocated for a certain logical structure. For example, the different types of segments include the following:

Data Segment	Each non-clustered table has a data segment. All of the table's data is stored in the extents of its data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
Index Segment	Each index has an index segment that stores all of its data.
Rollback Segment	One or more rollback segments are created by the database administrator for a database to temporarily store "undo" information. This information is used:
Temporary Segment	Temporary segments are created by Oracle when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the system for future use.

Oracle dynamically allocates space when the existing extents of a segment become full. Therefore, when the existing extents of a segment are full, Oracle allocates another extent for that segment as needed. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

Data Blocks, Extents, and Segments

Oracle allocates logical database space for all data in a database. The units of database space allocation are data blocks, extents, and segments. The following illustration shows the relationships among these data structures:

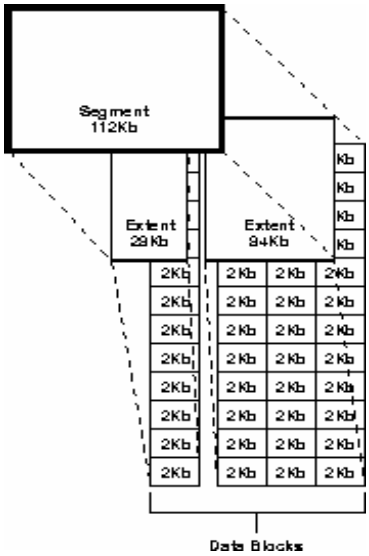


Figure 4.2: The Relationships Among Segments, Extents, and Data Blocks

At the finest level of granularity, Oracle stores data in *data blocks* (also called *logical blocks*, *Oracle blocks*, or *pages*). One data block corresponds to a specific number of bytes of physical database space on disk.

The next level of logical database space is called an *extent*. An extent is a specific number of contiguous data blocks allocated for storing a specific type of information.

The level of logical database storage above an extent is called a *segment*. A segment is a set of extents that have been allocated for a specific type of data structure. For example, each table's data is stored in its own *data segment*, while each index's data is stored in its own *index segment*. If the table or index is partitioned, each partition is stored in its own segment.

Oracle allocates space for segments in units of one extent. When the existing extents of a segment are full, Oracle allocates another extent for that segment. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

A segment (and all its extents) are stored in one tablespace. Within a tablespace, a segment can span datafiles (have extents with data from more than one file). However, each extent can contain data from only one datafile.

4.4 Data Blocks

Oracle manages the storage space in the datafiles of a database in units called *data blocks*. A data block is the smallest unit of I/O used by a database. In contrast, at the physical, operating system level, all data is stored in bytes. Each operating system has what is called a *block size*. Oracle requests data in multiples of Oracle data blocks, not operating system blocks.

You set the data block size for each Oracle database when you create the database. This data block size should be a multiple of the operating system's block size within the maximum (port-specific) limit to avoid unnecessary I/O. Oracle data blocks are the smallest units of storage that Oracle can use or allocate.

Data Block Format

The Oracle data block format is similar regardless of whether the data block contains table, index, or clustered data. Figure 4.3 illustrates the format of a data block.

Header (Common and Variable)

The header contains general block information, such as the block address and the type of segment (for example, data, index, or rollback).

Table Directory

This portion of the data block contains information about the tables having rows in this block.

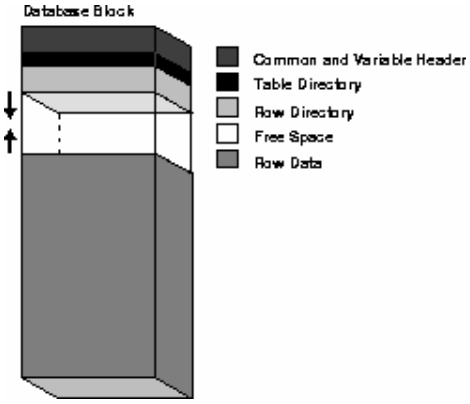


Figure 4.3: Data Block Format

Row Directory

This portion of the data block contains information about the actual rows in the block (including addresses for each row piece in the row data area).

Once the space has been allocated in the row directory of a data block's overhead, this space is not reclaimed when the row is deleted. Therefore, a block that is currently empty but had up to 50 rows at one time continues to have 100 bytes allocated in the header for the row directory. Oracle reuses this space only when new rows are inserted in the block.

Overhead

The data block header, table directory, and row directory are referred to collectively as *overhead*. Some block overhead is fixed in size; the total block overhead size is variable. On average, the fixed and variable portions of data block overhead total 84 to 107 bytes.

Row Data

This portion of the data block contains table or index data. Rows can span blocks.

Free Space

Free space is allocated for insertion of new rows and for updates to rows that require additional space (for example, when a trailing null is updated to a non-null value). Whether issued insertions actually occur in a

given data block is a function of current free space in that data block and the value of the space management parameter PCTFREE.

In data blocks allocated for the data segment of a table or cluster, or for the index segment of an index, free space can also hold transaction entries. A *transaction entry* is required in a block for each INSERT, UPDATE, DELETE, and SELECT...FOR UPDATE statement accessing one or more rows in the block. The space required for transaction entries is operating system dependent; however, transaction entries in most operating systems require approximately 23 bytes.

An Introduction to PCTFREE, PCTUSED, and Row Chaining

Two space management parameters, PCTFREE and PCTUSED, enable you to control the use of free space for inserts of and updates to the rows in all the data blocks of a particular segment. You specify these parameters when creating or altering a table or cluster (which has its own *data* segment). You can also specify the storage parameter PCTFREE when creating or altering an index (which has its own *index* segment).

Note: This discussion does not apply to the LOB datatypes (BLOB, CLOB, NCLOB, and BFILE) - they do not use the PCTFREE storage parameter or free lists.

The PCTFREE Parameter

The PCTFREE parameter sets the minimum percentage of a data block to be *reserved* as free space for possible updates to rows that already exist in that block. For example, assume that you specify the following parameter within a CREATE TABLE statement:

PCTFREE 20

This states that 20% of each data block in this table's data segment will be kept free and available for possible updates to the existing rows already within each block. New rows can be added to the row data area, and corresponding information can be added to the variable portions of the overhead area, until the row data and overhead total 80% of the total block size. Figure 4.4 illustrates PCTFREE.

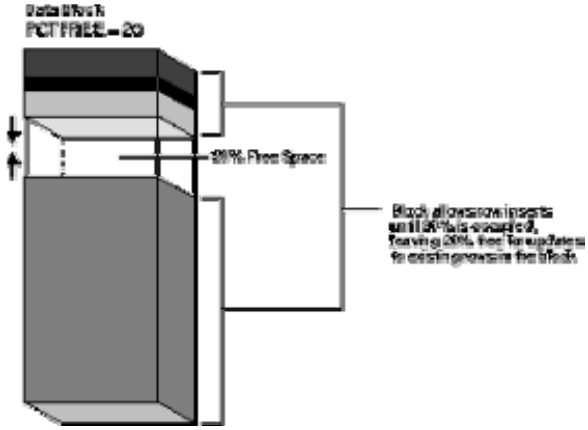


Figure 4.4: PCTFREE

The PCTUSED Parameter

The PCTUSED parameter sets the minimum percentage of a block that can be *used* for row data plus overhead before new rows will be added to the block. After a data block is filled to the limit determined by PCTFREE, Oracle considers the block unavailable for the insertion of new rows until the percentage of that block falls below the parameter PCTUSED. Until this value is achieved, Oracle uses the free space of the data block only for

updates to rows already contained in the data block. For example, assume that you specify the following parameter in a CREATE TABLE statement:

PCTUSED 40

In this case, a data block used for this table's data segment is considered unavailable for the insertion of any new rows until the amount of used space in the block falls to 39% or less (assuming that the block's used space has previously reached PCTFREE). Figure 4.5 illustrates this.

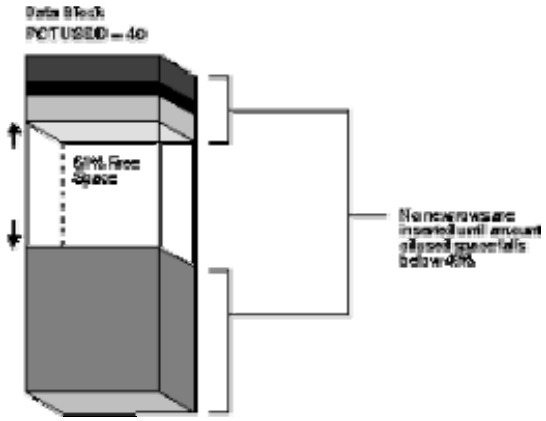


Figure 4.5: PCTUSED

Row Chaining and Migrating

In two circumstances, the data for a row in a table may be too large to fit into a single data block. In the first case, the row is too large to fit into one data block when it is first inserted. In this case, Oracle stores the data for the row in a *chain* of data blocks (one or more) reserved for that segment. Row chaining most often occurs with large rows, such as rows that contain a column of datatype LONG or LONG RAW. Row chaining in these cases is unavoidable.

However, in the second case, a row that originally fit into one data block is updated so that the overall row length increases, and the block's free space is already completely filled. In this case, Oracle *migrates* the data for the entire row to a new data block, assuming the entire row can fit in a new block. Oracle preserves the original row piece of a migrated row to point to the new block containing the migrated row; the ROWID of a migrated row does not change.
Note: For more information about ROWID.

When a row is chained or migrated, I/O performance associated with this row decreases because Oracle must scan more than one data block to retrieve the information for the row.

4.5 Extents

An extent is a logical unit of database storage space allocation made up of a number of contiguous data blocks. One or more extents in turn make up a segment. When the existing space in a segment is completely used, Oracle allocates a new extent for the segment.

4.6 Segments

A segment is a set of extents that contains all the data for a specific logical storage structure within a tablespace. For example, for each table, Oracle allocates one or more extents to form that table's data segment; for each index, Oracle allocates one or more extents to form its index segment.

Oracle databases use four types of segments:

- Data Segments
- Index Segments
- Temporary Segments
- Temporary Segments

The following sections discuss each type of segment.

Data Segments

Every nonclustered table or partition and every cluster in an Oracle database has a single data segment to hold all of its data. Oracle creates this data segment when you create the nonclustered table or cluster with the CREATE command.

The storage parameters for a nonclustered table or cluster determine how its data segment's extents are allocated. You can set these storage parameters directly with the appropriate CREATE or ALTER command. These storage parameters affect the efficiency of data retrieval and storage for the data segment associated with the object.

Note: Oracle creates segments for snapshots and snapshot logs in the same manner as for nonclustered and clustered tables. For more information on snapshots and snapshot logs,

Index Segments

Every index in an Oracle database has a single index segment to hold all of its data. Oracle creates the index segment for the index when you issue the CREATE INDEX command. In this command, you can specify storage parameters for the extents of the index segment and a tablespace in which to create the index segment. (The segments of a table and an index associated with it do not have to occupy the same tablespace.) Setting the storage parameters directly affects the efficiency of data retrieval and storage.

Temporary Segments

When processing queries, Oracle often requires temporary workspace for intermediate stages of SQL statement parsing and execution. Oracle automatically allocates this disk space called a *temporary segment*. Typically, Oracle requires a temporary segment as a work area for sorting. Oracle does not create a segment if the sorting operation can be done in memory or if Oracle finds some other way to perform the operation using indexes.

Operations Requiring Temporary Segments

The following commands may require the use of a temporary segment:

- CREATE INDEX
- SELECT ... ORDER BY

- SELECT DISTINCT ...
- SELECT ... GROUP BY
- SELECT ... UNION
- SELECT ... INTERSECT
- SELECT ... MINUS

Some unindexed joins and correlated subqueries may also require use of a temporary segment. For example, if a query contains a DISTINCT clause, a GROUP BY, and an ORDER BY, Oracle can require as many as two temporary segments. If applications often issue commands in the list above, the database administrator may want to improve performance by adjusting the initialization parameter SORT_AREA_SIZE.

Rollback Segments

Each database contains one or more rollback segments. A rollback segment records the old values of data that was changed by each transaction (whether or not committed). Rollback segments are used to provide read consistency, to roll back transactions, and to recover the database. For specific information about how rollback segments function in these situations, see the appropriate sections of this book:

Contents of a Rollback Segment

Information in a rollback segment consists of several *rollback entries*. Among other information, a rollback entry includes block information (the filenumber and block ID corresponding to the data that was changed) and the data as it existed before an operation in a transaction. Oracle links rollback entries for the same transaction, so the entries can be found easily if necessary for transaction rollback.

Neither database users nor administrators can access or read rollback segments; only Oracle can write to or read them. (They are owned by the user SYS, no matter which user creates them.)

4.7 Physical Database Structures

The following sections explain the physical database structures of an Oracle database, including datafiles, redo log files, and control files.

4.8 Datafiles

Every Oracle database has one or more physical *datafiles*. A database's datafiles contain all the database data. The data of logical database structures such as tables and indexes is physically stored in the datafiles allocated for a database. The following are characteristics of datafiles:

- A datafile can be associated with only one database.
- Database files can have certain characteristics set to allow them to automatically extend when the database runs out of space.
- One or more datafiles form a logical unit of database storage called a tablespace, as discussed earlier in this chapter.

The Use of Datafiles

The data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. For example, assume that a user wants to access some data in a table of a database. If the requested

information is not already in the memory cache for the database, it is read from the appropriate datafiles and stored in memory.

Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by the DBWR background process of Oracle.

4.9 Redo Log Files

Every Oracle database has a set of two or more *redo log files*. The set of redo log files for a database is collectively known as the database's *redo log*. The primary function of the redo log is to record all changes made to data. Should a failure prevent modified data from being permanently written to the datafiles, the changes can be obtained from the redo log and work is never lost.

Redo log files are critical in protecting a database against failures. To protect against a failure involving the redo log itself, Oracle allows a *multiplexed redo log* so that two or more copies of the redo log can be maintained on different disks.

The Use of Redo Log Files

The information in a redo log file is used only to recover the database from a system or media failure that prevents database data from being written to a database's datafiles.
For example, if an unexpected power outage abruptly terminates database operation, data in memory cannot be written to the datafiles and the data is lost. However, any lost data can be recovered when the database is opened, after power is restored. By applying the information in the most recent redo log files to the database's datafiles, Oracle restores the database to the time at which the power failure occurred.

The process of applying the redo log during a recovery operation is called *rolling forward*.

4.10 Control Files

Every Oracle database has a *control file*. A control file contains entries that specify the physical structure of the database. For example, it contains the following types of information:

- database name
- names and locations of a database's datafiles and redo log files
- time stamp of database creation

Like the redo log, Oracle allows the control file to be multiplexed for protection of the control file.

The Use of Control Files

Every time an instance of an Oracle database is started, its control file is used to identify the database and redo log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, a new datafile or redo log file is created), the database's control file is automatically modified by Oracle to reflect the change.

A database's control file is also used if database recovery is necessary.

4.11 Short summary

An Oracle *database* is a collection of data that is treated as a unit. The general purpose of a database is to store and retrieve related information. The database has *logical structures* and *physical structures*. Logical database structures, including tablespaces, schema objects, data blocks, extents, and segments.

4.12 Brain storm

1. Dictionary Cache is also referred as row cache
   a) True b) False

2. Every process that make a change to the Database must write an entry to the Redo log inorder to allow or act to recover the change
   a) True b) False

3. In the absence of LGWR, Checkpoint takes care of writing data to Log files
   a) True b) False

4. RECO and Dispatcher are optional Background process
   a) True b) False

5. One of the following is an invalid combination for Startup option.
   a) Startup Mount
   b) Startup Quiet
   c) Startup Restricted
   d) Startup Abort

6. What causes Row Migration?
   a) When PCTFREE is set extremely high
   b) When an update requires more space than is currently available on the block
   c) When a block is chained
   d) When PCTUSED is set extremely high

Lecture 5

Table Space and Data Files

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about Table spaces
- Describe about types of table spaces
- Discuss about data files, size and its content

Coverage Plan

Lecture 5

5.1 Snap Shot

5.2 Introduction Tablespace And Datafiles

5.3 Tablespace

5.4 Data Files

5.5 Short Summary

5.6 Brain Storm

5.1 Snap Shot

This chapter describes tablespaces, the primary logical database structures of any Oracle database, and the physical datafiles that correspond to each tablespace. The chapter includes:

- An Introduction to Tablespaces and Datafiles
- Tablespaces
- Datafiles

5.2 Tablespaces and Datafiles

Oracle stores data logically in *tablespaces* and physically in *datafiles* associated with the corresponding tablespace. Figure 5-1 illustrates this relationship.

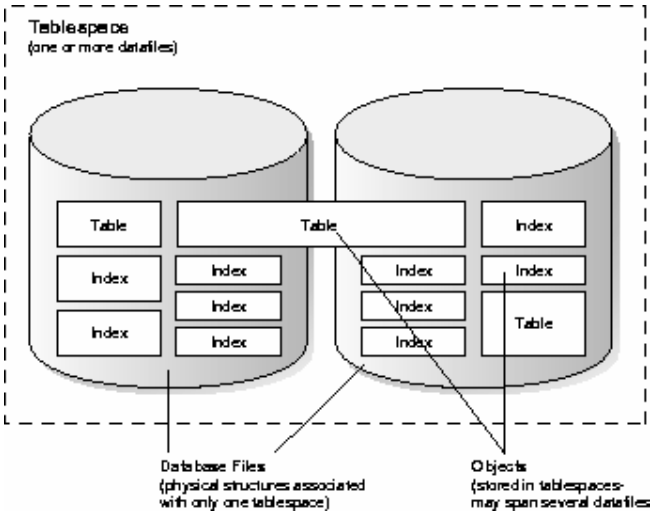


Figure 5-1: Datafiles and Tablespaces

Databases, tablespaces, and datafiles are closely related, but they have important differences:

Databases and tablespaces	An Oracle database consists of one or more logical storage units called tablespaces. The database's data is collectively stored in the database's tablespaces.
tablespaces and datafiles	Each tablespace in an Oracle database consists of one or more files called datafiles. These are physical structures that conform with the operating system in which Oracle is running.
Databases and datafiles	A database's data is collectively stored in the datafiles that constitute each tablespace of the database. For example, the simplest Oracle database would have one tablespace and one datafile. A more complicated database might have three tablespaces, each consisting of two datafiles (for a total of six datafiles).

The following sections further explain tablespaces and datafiles.

5.3 Tablespaces

A database is divided into one or more logical storage units called *tablespaces*. The database administrator (DBA) uses tablespaces to:

- control disk space allocation for database data
- assign specific space quotas for database users

- control availability of data by taking individual tablespaces online or offline
- perform partial database backup or recovery operations
- allocate data storage across devices to improve performance

A DBA can create new tablespaces, add datafiles to tablespaces, set and alter default segment storage settings for segments created in a tablespace, make a tablespace read-only or read-write, make a tablespace temporary or permanent, and drop tablespaces.

Tablespaces are divided into logical units of storage called *segments*.

The SYSTEM Tablespace

Every Oracle database contains a tablespace named SYSTEM, which Oracle creates automatically when the database is created. The SYSTEM tablespace always contains the data dictionary tables for the entire database.

A small database might need only the SYSTEM tablespace; however, Oracle Corporation recommends that you create at least one additional tablespace to store user data separate from data dictionary information. This gives you more flexibility in various database administration operations and reduces contention among dictionary objects and schema objects for the same datafiles.

Note: The SYSTEM tablespace is always online when the database is open.

Allocating More Space for a Database

You can enlarge a database in three ways:

- add a datafile to a tablespace
- add a new tablespace
- increase the size of a datafile

When you add another datafile to an existing tablespace, you increase the amount of disk space allocated for the corresponding tablespace. Figure 5-2 illustrates this kind of space increase.

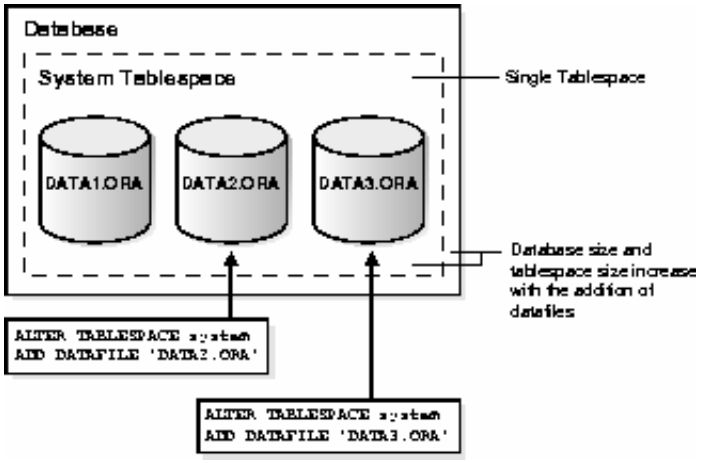


Figure 5-2: Enlarging a Database by Adding a Datafile to a Tablespace

Alternatively, you can create a new tablespace (which contains at lest one additional datafile) to increase the size of a database. Figure 5-3 illustrates this.

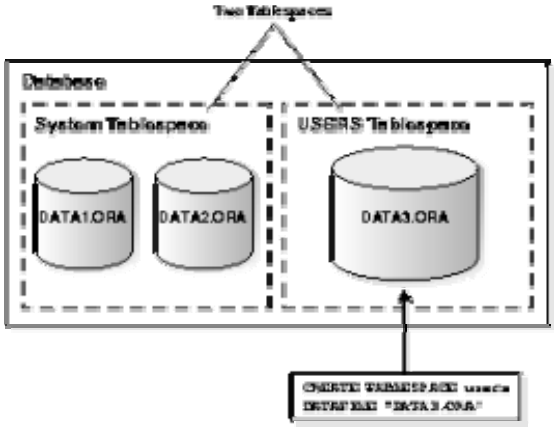


Figure 5-3: Enlarging a Database by Adding a New Tablespace

The size of a tablespace is the size of the datafile(s) that constitute the tablespace; the size of a database is the collective size of the tablespaces that constitute the database.

The third option is to change a datafile's size or allow datafiles in existing tablespaces to grow dynamically as more space is needed. You accomplish this by altering existing files or by adding files with dynamic extension properties. Figure 5-4 illustrates this.

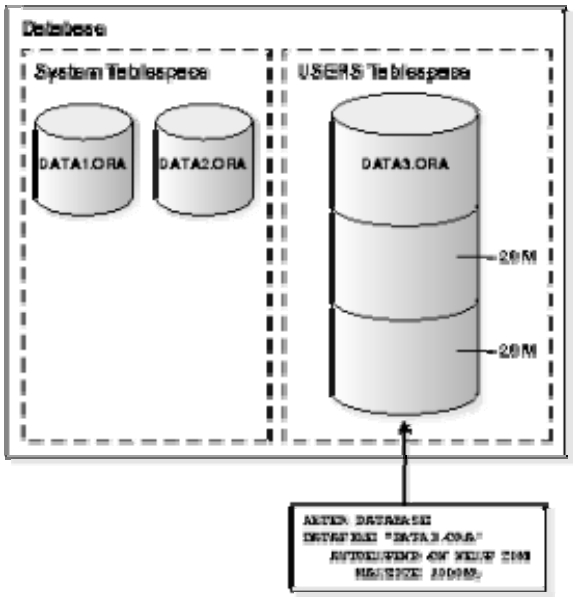


Figure 5-4: Enlarging a Database by Dynamically Sizing Datafiles

Bringing Tablespaces Online and Offline

A database administrator can bring any tablespace (except the SYSTEM tablespace) in an Oracle database *online* (accessible) or *offline* (not accessible) whenever the database is open.

Note: The SYSTEM tablespace is always online when the database is open because the data dictionary must always be available to Oracle.

A tablespace is normally online so that the data contained within it is available to database users. However, the database administrator might take a tablespace offline

- to make a portion of the database unavailable, while allowing normal access to the remainder of the database
- to perform an offline tablespace backup (although a tablespace can be backed up while online and in use)
- to make an application and its group of tables temporarily unavailable while updating or maintaining the application

You cannot take a tablespace offline if it contains any rollback segments that are in use.

When a Tablespace Goes Offline

When a tablespace goes offline, Oracle does not permit any subsequent SQL statements to reference objects contained in that tablespace. Active transactions with completed statements that refer to data in that tablespace are not affected at the transaction level. Oracle saves rollback data corresponding to those completed statements in a deferred rollback segment (in the SYSTEM tablespace). When the tablespace is brought back online, Oracle applies the rollback data to the tablespace, if needed.

When a tablespace goes offline or comes back online, this is recorded in the data dictionary in the SYSTEM tablespace. If a tablespace was offline when you shut down a database, the tablespace remains offline when the database is subsequently mounted and reopened.

You can bring a tablespace online only in the database in which it was created because the necessary data dictionary information is maintained in the SYSTEM tablespace of that database. An offline tablespace cannot be read or edited by any utility other than Oracle. Thus, tablespaces cannot be transferred from database to database.

Oracle automatically switches a tablespace from online to offline when certain errors are encountered (for example, when the database writer process, DBWR, fails in several attempts to write to a datafile of the tablespace). Users trying to access tables in the offline tablespace receive an error. If the problem that causes this disk I/O to fail is media failure, you must recover the tablespace after you correct the hardware problem.

Read-Only Tablespaces

The primary purpose of read-only tablespaces is to eliminate the need to perform backup and recovery of large, static portions of a database. Oracle never updates the files of a read-only tablespace, and therefore the files can reside on read-only media, such as CD ROMs or WORM drives.

Whenever you create a new tablespace, it is always created as read-write. You can change the tablespace to read-only with the READ ONLY option of the ALTER TABLESPACE command, making all of the tablespace's

associated datafiles read-only as well. You can use the READ WRITE option to make a read-only tablespace read-write again.

Making a tablespace read-only does not change its offline or online status. Offline datafiles cannot be accessed. Bringing a datafile in a read-only tablespace online makes the file only readable. The file cannot be written to unless its associated tablespace is returned to the read-write state. You can take the files of a read-only tablespace online or offline independently using the DATAFILE option of the ALTER DATABASE command.

Read-only tablespaces cannot be modified. To update a read-only tablespace, you must first make the tablespace read-write. After updating the tablespace, you can then reset it to be read-only.
You can drop items, such as tables and indexes, from a read-only tablespace, just as you can drop items from an offline tablespace. However, you cannot create or alter objects in a read-only tablespace.

You cannot add datafiles to a read-only tablespace, even if you take the tablespace offline. When you add a datafile, Oracle must update the file header, and this write operation is not allowed in a read-only tablespace.

Temporary Tablespaces

You can manage space for sort operations more efficiently by designating *temporary tablespaces* exclusively for sorts. Doing so effectively eliminates serialization of space management operations involved in the allocation and deallocation of sort space. All operations that use sorts-including joins, index builds, ordering (ORDER BY), the computation of aggregates (GROUP BY), and the ANALYZE command to collect optimizer statistics-benefit from temporary tablespaces. The performance gains are significant in Oracle Parallel Server environments.

A *temporary tablespace* can be used only for sort segments. (It is not the same as a tablespace that a user designates for temporary segments, which can be any tablespace available to the user.) No permanent objects can reside in a temporary tablespace. Sort segments are used when a segment is shared by multiple sort operations. One sort segment exists in every instance that performs a sort operation in a given tablespace.

Temporary tablespaces provide performance improvements when you have multiple sorts that are too large to fit into memory. The sort segment of a given temporary tablespace is created at the time of the first sort operation. The sort segment expands by allocating extents until the segment size is equal to or greater than the total storage demands of all of the active sorts running on that instance.

You create temporary tablespaces using the following SQL syntax:

CREATE TABLESPACE tablespace TEMPORARY| PERMANENT

You can also alter a tablespace from PERMANENT to TEMPORARY or vice versa using the following syntax:

ALTER TABLESPACE tablespace TEMPORARY

5.4 Datafiles

A tablespace in an Oracle database consists of one or more physical *datafiles*. A datafile can be associated with only one tablespace and only one database.

Oracle creates a datafile for a tablespace by allocating the specified amount of disk space plus the overhead required for the file header. When a datafile is created, the operating system in which Oracle is running is responsible for clearing old information and authorizations from a file before allocating it to Oracle. If the file is large, this process might take a significant amount of time.

The first tablespace in any database is always the SYSTEM tablespace, so Oracle automatically allocates the first datafiles of any database for the SYSTEM tablespace during database creation.

Datafile Contents

When a datafile is first created, the allocated disk space is formatted but does not contain any user data; however, Oracle reserves the space to hold the data for future segments of the associated tablespace - it is used exclusively by Oracle. As the data grows in a tablespace, Oracle uses the free space in the associated datafiles to allocate extents for the segment.

The data associated with schema objects in a tablespace is physically stored in one or more of the datafiles that constitute the tablespace. Note that a schema object does not correspond to a specific datafile; rather, a datafile is a repository for the data of any object within a specific tablespace. Oracle allocates space for the data associated with an object in one or more datafiles of a tablespace. Therefore, an object can "span" one or more datafiles. Unless table "striping" is used (where data is spread across more than one disk), the database administrator and end users cannot control which datafile stores an object.

Size of Datafiles

You can alter the size of a datafile after its creation or you can specify that a datafile should dynamically grow as schema objects in the tablespace grow. This functionality enables you to have fewer datafiles per tablespace and can simplify administration of datafiles.

Offline Datafiles

You can take tablespaces *offline* (make unavailable) or bring them *online* (make available) at any time except SYSTEM. All datafiles making up a tablespace are taken offline or brought online as a unit when you take the tablespace offline or bring it online, respectively. You can take individual datafiles offline; however, this is normally done only during some database recovery procedures.

5.5 Short Summary

A database is divided into one or more logical storage units called *tablespaces*. The database administrator (DBA) uses tablespaces to:

- control disk space allocation for database data
- assign specific space quotas for database users
- control availability of data by taking individual tablespaces online or offline
- perform partial database backup or recovery operations
- Allocate data storage across devices to improve performance

5.6 Brain Storm

1. Which components store the synchronization information needed for Database Recovery?
   a) Redo Log File b) Control File c) Parameter d) Trace File

2. What is the purpose of the index segment type?
   a) Stores Data entries
   b) Increases performance of retrieving Data
   c) Allows for the undo of transactions
   d) Allows for implicit sorts

3. To determine the storage allocation for temporary segments, the DBA can access which of the following views?

4. What is MINIMUM EXTENT?

5. Where can be current settings for the BACKGROUND_DUMP_DEST & USER_DUMP_DEST be viewed?

6. What will happen if a DEFAULT TABLESPACE is not specified when creating a user?

Lecture 6

Creating Oracle Database

Objectives

After completing this lesson, you should be able to do the following:

- Considerations before and after creating a database
- Creation of oracle database
- Discuss about Troubleshooting method of database
- Discuss about dropping method of database
- About parameters and initial tuning guidelines

Coverage Plan

Lecture 6

6.1 Snap Shot

6.2 Considerations Before Creating Database

6.3 Creation Of Database

6.4 Troubleshooting Database

6.5 Dropping Database

6.6 Parameters

6.7 Considerations After Creating Database

6.8 Initial Tuning Guidelines

6.9 Short Summary

6.10. Brain Storm

6.1 Snap Shot

This chapter lists the steps necessary to create an Oracle database, and includes the following topics:

- Considerations Before Creating a Database
- Creating an Oracle Database
- Parameters
- Considerations After Creating a Database
- Initial Tuning Guidelines

6.2 Considerations Before Creating a Database

This section includes the following topics:

- Creation Prerequisites
- Using an Initial Database
- Migrating an Older Version of the Database

Database creation prepares several operating system files so they can work together as an Oracle database. You need only create a database once, regardless of how many datafiles it has or how many instances access it. Creating a database can also erase information in an existing database and create a new database with the same name and physical structure.
Creating a database includes the following operations:

- creating new datafiles or erasing data that existed in previous datafiles
- information creating structures that Oracle requires to access and use the database (the data dictionary)
- creating and initializing the control files and redo log files for the database

Consider the following issues before you create a database:

- Plan your database tables and indexes, and estimate how much space they will require.
- Plan how to protect your new database, including the configuration of its online and archived redo log (and how much space it will require), and a backup strategy.
- Select the database character set. You must specify the database character set when you create the database. If you create the database with the wrong character set by mistake, you can update the SYS.PROPS$ table with the new character set; however, you cannot change characters already there. All character data, including data in the data dictionary, is stored in the database character set. If users access the database using a different character set, the database character set should be the same as, or a superset of, all character sets they use.

Also become familiar with the principles and options of starting up and shutting down an instance, mounting and opening a database, and using parameter files.

Creation Prerequisites

To create a new database, you must have the following:

- the operating system privileges associated with a fully operational database administrator

- sufficient memory to start the Oracle instance
- sufficient disk storage space for the planned database on the computer that executes Oracle

Using an Initial Database

Depending on your operating system, a database might have been created automatically as part of the installation procedure for Oracle. You can use this initial database and customize it to meet your information management requirements, or discard it and create one or more new databases to replace it.

Migrating an Older Version of the Database

If you are using a previous release of Oracle, database creation is required only if you want an entirely new database. Otherwise, you can migrate your existing Oracle databases managed by a previous version of Oracle and use them with the new version of the Oracle software.

6.3 Creating an Oracle Database

This section includes the following topics:

- Steps for Creating an Oracle Database
- Creating a Database: Example
- Troubleshooting Database Creation
- Dropping a Database

Steps for Creating an Oracle Database

These steps, which describe how to create an Oracle database, should be followed in the order presented.

To Create a New Database and Make It Available for System Use

1. Back up any existing databases.
2. Create parameter files.
3. Edit new parameter files.
4. Check the instance identifier for your system.
5. Start Enterprise Manager and connect to Oracle as an administrator.
6. Start an instance.
7. Create the database.
8. Back up the database.

Step 1: Back up any existing databases.

Oracle Corporation strongly recommends that you make complete backups of all existing databases before creating a new database, in case database creation accidentally affects some existing files. Backup should include parameter files, datafiles, redo log files, and control files.

Step 2: Create parameter files.

The instance (System Global Area and background processes) for any Oracle database is started using a parameter file.

Each database on your system should have at least one customized parameter file that corresponds only to that database. Do not use the same file for several databases.

To create a parameter file for the database you are about to make, use your operating system to make a copy of the parameter file that Oracle provided on the distribution media. Give this copy a new filename. You can then edit and customize this new file for the new database.

Note: In distributed processing environments, Enterprise Manager is often executed from a client machine of the network. If a client machine is being used to execute Enterprise Manager and create a new database, you need to copy the new parameter file (currently located on the computer executing Oracle) to your client workstation. This procedure is operating system dependent. For more information about copying files among the computers of your network, see your operating system-specific Oracle documentation.

Step 3: Edit new parameter files.

To create a new database, inspect and edit the following parameters of the new parameter file:

Table : Suggested Initialization Parameters to Edit

Parameter
DB_NAME
DB_DOMAIN
CONTROL_FILES
DB_BLOCK_SIZE
DB_BLOCK_BUFFERS
PROCESSES
ROLLBACK_SEGMENTS

You should also edit the appropriate license parameter(s):
Table: License Initialization Parameters

Parameter
LICENSE_MAX_SESSIONS
LICENSE_SESSION_WARNING
LICENSE_MAX_USERS

Step 4: Check the instance identifier for your system.

If you have other databases, check the Oracle instances identifier. The Oracle instance identifier should match the name of the database (the value of DB_NAME) to avoid confusion with other Oracle instances that are running concurrently on your system.

Step 5: Start Enterprise Manager and connect to Oracle as an administrator.

Once Enterprise Manager is running, connect to the database as an administrator.

Step 6: Start an instance.

To start an instance (System Global Area and background processes) to be used with the new database, use the Startup Database dialog box of Enterprise Manager. In the Startup Database dialog box, make sure that you have selected the Startup Nomount radio button.

After selecting the Startup Nomount, the instance starts. At this point, there is no database. Only an SGA and background processes are started in preparation for the creation of a new database.

Step 7: Create the database.

To create the new database, use the SQL command CREATE DATABASE, optionally setting parameters within the statement to name the database, establish maximum numbers of files, name the files and set their sizes, and so on.

When you execute a CREATE DATABASE statement, Oracle performs the following operations:

- creates the datafiles for the database
- creates the control files for the database
- creates the redo log files for the database
- creates the SYSTEM tablespace and the SYSTEM rollback segment
- creates the data dictionary
- creates the users SYS and SYSTEM
- specifies the character set that stores data in the database
- mounts and opens the database for use

Warning: Make sure that the datafiles and redo log files that you specify do not conflict with files of another database.

Step 8: Back up the database.

You should make a full backup of the database to ensure that you have a complete set of files from which to recover if a media failure occurs.

Creating a Database: Example

The following statement is an example of a CREATE DATABASE statement:

```
CREATE DATABASE test
DATAFILE 'test_system' SIZE 10M
LOGFILE GROUP 1 ('test_log1a', 'test_log1b') SIZE 500K,
GROUP 2 ('test_log2a', 'test_log2b') SIZE 500K;
```

The values of the MAXLOGFILES, MAXLOGMEMBERS, MAXDATAFILES, MAXLOGHISTORY, and MAXINSTANCES options in this example assume the default values, which are operating system-dependent.

The database is mounted in the default modes NOARCHIVELOG and EXCLUSIVE and then opened.

The items and information in the example statement above result in creating a database with the following characteristics:

- The new database is named TEST.
- The SYSTEM tablespace of the new database is comprised of one 10 MB datafile named TEST_SYSTEM.

- The new database has two online redo log groups, each containing two 500 KB members.
- The new database does not overwrite any existing control files specified in the parameter file.

N**ote:** You can set several limits during database creation. Some of these limits are also subject to superseding limits of the operating system and can affect each other. For example, if you set MAXDATAFILES, Oracle allocates enough space in the control file to store MAXDATAFILES filenames, even if the database has only one datafile initially; because the maximum control file size is limited and operating system-dependent, you might not be able to set all CREATE DATABASE parameters at their theoretical maximums.

6.4 Troubleshooting Database Creation

If for any reason database creation fails, shut down the instance and delete any files created by the CREATE DATABASE statement before you attempt to create it once again.

After correcting the error that caused the failure of the database creation, return to Step 6 of "Creating a Oracle Database."

6.5 Dropping a Database

To drop a database, remove its datafiles, redo log files, and all other associated files (control files, parameter files, archived log files).

To view the names of the database's datafiles and redo log files, query the data dictionary views V$DATAFILE and V$LOGFILE.

6.6 Parameters

As described in Step 3 of the section "Creating an Oracle Database", Oracle suggests you alter a minimum set of parameters. These parameters are described in the following sections:

- DB_NAME and DB_DOMAIN
- CONTROL_FILES
- DB_BLOCK_SIZE
- PROCESSES
- ROLLBACK_ SEGMENTS
- License Parameters
- DB_BLOCK_BUFFERS
- LICENSE_MAX_SESSIONS_and LICENSE_SESSIONS WARNING
- LICENSE_MAX_ USERS
- DB_NAME and DB_DOMAIN

A database's *global database name* (name and location within a network structure) is created by setting both the DB_NAME and DB_DOMAIN parameters before database creation. After creation, the database's name cannot be easily changed. The DB_NAME parameter determines the local name component of the database's name, while the DB_DOMAIN parameter indicates the domain (logical location) within a network structure. The combination of the settings for these two parameters should form a database name that is unique within a network. For example, to create a database with a global database name of TEST.US.ACME.COM, edit the parameters of the new parameter file as follows:

```
DB_NAME = TEST
DB_DOMAIN = US.ACME.COM
```

DB_NAME must be set to a text string of no more than eight characters. During database creation, the name provided for DB_NAME is recorded in the datafiles, redo log files, and control file of the database. If during database instance startup the value of the DB_NAME parameter (of the parameter file) and the database name in the control file are not the same, the database does not start.

DB_DOMAIN is a text string that specifies the network domain where the database is created; this is typically the name of the organization that owns the database. If the database you are about to create will ever be part of a distributed database system, pay special attention to this initialization parameter before database creation.

CONTROL_FILES

Include the CONTROL_FILES parameter in your new parameter file and set its value to a list of control filenames to use for the new database. If you want Oracle to create new operating system files when creating your database's control files, make sure that the filenames listed in the CONTROL_FILES parameter do not match any filenames that currently exist on your system. If you want Oracle to reuse or overwrite existing files when creating your database's control files, make sure that the filenames listed in the CONTROL_FILES parameter match the filenames that currently exist.

Warning: Use extreme caution when setting this option. If you inadvertently specify a file that you did not intend and execute the CREATE DATABASE statement, the previous contents of that file will be overwritten.

If no filenames are listed for the CONTROL_FILES parameter, Oracle uses a default filename.

Oracle Corporation strongly recommends you use at least two control files stored on separate physical disk drives for each database. Therefore, when specifying the CONTROL_FILES parameter of the new parameter file, follow these guidelines:

- List at least two filenames for the CONTROL_FILES parameter.
- Place each control file on a separate physical disk drives by fully specifying filenames that refer to different disk drives for each filename.

Note: The file specification for control files is operating system-dependent. Regardless of your operating system, *always* fully specify filenames for your control files.

When you execute the CREATE DATABASE statement (in Step 7), the control files listed in the CONTROL_FILES parameter of the parameter file will be created.

DB_BLOCK_SIZE

The default data block size for every Oracle Server is operating system-specific. The Oracle data block size is typically either 2K or 4K. Generally, the default data block size is adequate. In some cases, however, a larger data block size provides greater efficiency in disk and memory I/O (access and storage of data). Such cases include:

- Oracle is on a large computer system with a large amount of memory and fast disk drives. For example, databases controlled by mainframe computers with vast hardware resources typically use a data block size of 4K or greater.
- The operating system that runs Oracle uses a small operating system block size. For example, if the operating system block size is 1K and the data block size matches this, Oracle may be performing an

excessive amount of disk I/O during normal operation. For best performance in this case, a database block should consist of multiple operating system blocks.

Each database's block size is set during database creation by the initialization parameter DB_BLOCK_SIZE. The block size *cannot* be changed after database creation except by re-creating the database. If a database's block size is different from the operating system block size, make the data block size a multiple of the operating system's block size.

For example, if your operating system's block size is 2K (2048 bytes), the following setting for the DB_BLOCK_SIZE initialization parameter would be valid:

DB_BLOCK_SIZE=4096

DB_BLOCK_SIZE also determines the size of the database buffers in the buffer cache of the System Global Area (SGA).

DB_BLOCK_BUFFERS

This parameter determines the number of buffers in the buffer cache in the System Global Area (SGA). The number of buffers affects the performance of the cache. Larger cache sizes reduce the number of disk writes of modified data. However, a large cache may take up too much memory and induce memory paging or swapping.

Estimate the number of data blocks that your application accesses most frequently, including tables, indexes, and rollback segments. This estimate is a rough approximation of the minimum number of buffers the cache should have. Typically, 1000 to 2000 is a practical minimum for the number of buffers.

PROCESSES

This parameter determines the maximum number of operating system processes that can be connected to Oracle concurrently. The value of this parameter must include 5 for the background processes and 1 for each user process. For example, if you plan to have 50 concurrent users, set this parameter to at least 55.

ROLLBACK_SEGMENTS

This parameter is a list of the rollback segments an Oracle instance acquires at database startup. List your rollback segments as the value of this parameter.

A**ttention:** After installation, you must create at least one rollback segment in the SYSTEM tablespace in addition to the SYSTEM rollback segment before you can create any schema objects.

License Parameters

Oracle helps you ensure that your site complies with its Oracle license agreement. If your site is licensed by concurrent usage, you can track and limit the number of sessions concurrently connected to an instance. If your site is licensed by named users, you can limit the number of named users created in a database. To use this facility, you need to know which type of licensing agreement your site has and what the maximum number of sessions or named users is. Your site might use either type of licensing (session licensing or named user licensing), but not both.

LICENSE_MAX_SESSIONS_and LICENSE_SESSIONS WARNING

You can set a limit on the number of concurrent sessions that can connect to a database on the specified computer. To set the maximum number of concurrent sessions for an instance, set the parameter LICENSE_MAX_SESSIONS in the parameter file that starts the instance, as shown in the following example:

LICENSE_MAX_SESSIONS = 80

In addition to setting a maximum number of sessions, you can set a warning limit on the number of concurrent sessions. Once this limit is reached, additional users can continue to connect (up to the maximum limit), but Oracle sends a warning for each connecting user. To set the warning limit for an instance, set the parameter LICENSE_SESSIONS_WARNING. Set the warning limit to a value lower than LICENSE_MAX_SESSIONS.

For instances running with the Parallel Server, each instance can have its own concurrent usage limit and warning limit. However, the sum of the instances' limits must not exceed the site's session license.

LICENSE_MAX_USERS

You can set a limit on the number of users created in the database. Once this limit is reached, you cannot create more users.

Note: This mechanism assumes that each person accessing the database has a unique user name and that no people share a user name. Therefore, so that named user licensing can help you ensure compliance with your Oracle license agreement, do not allow multiple users to log in using the same user name.

To limit the number of users created in a database, set the LICENSE_MAX_USERS parameter in the database's parameter file, as shown in the following example:

LICENSE_MAX_USERS = 200

For instances running with the Parallel Server, all instances connected to the same database should have the same named user limit.

6.7 Considerations After Creating a Database

After you create a database, the instance is left running, and the database is open and available for normal database use. Use Enterprise Manager to subsequently start and stop the database. If more than one database exists in your database system, specify the parameter file to use with any subsequent database startup.

If you plan to install other Oracle products to work with this database, see the installation instructions for those products; some products require you to create additional data dictionary tables. See your operating system-specific Oracle documentation for the additional products. Usually, command files are provided to create and load these tables into the database's data dictionary.

The Oracle Server distribution media can include various SQL files that let you experiment with the system, learn SQL, or create additional tables, views, or synonyms.

A newly created database has only two users, SYS and SYSTEM. The passwords for these two usernames should be changed soon after the database is created.

6.8 Initial Tuning Guidelines

You can make a few significant tuning alterations to Oracle immediately following installation. By following these instructions, you can reduce the need to tune Oracle when it is running. This section gives recommendations for the following installation issues:

- Allocating Rollback Segments
- Choosing the Number of DB_BLOCK_LRU_LATCHES

- Distributing I/O

Allocating Rollback Segments

Proper allocation of rollback segments makes for optimal database performance. The size and number of rollback segments required for optimal performance depends on your application. The Oracle8 Server Tuning manual contains some general guidelines for choosing how many rollback segments to allocate based on the number of concurrent transactions on your Oracle Server. These guidelines are appropriate for most application mixes.

To create rollback segments, use the CREATE ROLLBACK SEGMENT command.

Choosing Sizes for Rollback Segments

The size of your rollback segment can also affect performance. Rollback segment size is determined by the storage parameters in the CREATE ROLLBACK SEGMENT statement. Your rollback segments must be large enough to hold the rollback entries for your transactions.

Choosing the Number of DB_BLOCK_LRU_LATCHES

Contention for the LRU latch can impede performance on symmetric multiprocessor (SMP) machines with a large number of CPUs. The LRU latch controls the replacement of buffers in the buffer cache. For SMP systems, Oracle automatically sets the number of LRU latches to be one half the number of CPUs on the system. For non-SMP systems, one LRU latch is sufficient.

You can specify the number of LRU latches on your system with the initialization parameter DB_BLOCK_LRU_LATCHES. This parameter sets the maximum value for the desired number of LRU latches. Each LRU latch will control a set of buffers and Oracle balances allocation of replacement buffers among the sets.

Distributing I/O

Proper distribution of I/O can improve database performance dramatically. I/O can be distributed during installation of Oracle. Distributing I/O during installation can reduce the need to distribute I/O later when Oracle is running.

There are several ways to distribute I/O when you install Oracle:

- redo log file placement
- datafile placement
- separation of tables and indexes
- density of data (rows per data block)

6.9 Short Summary

Creating a database includes the following operations:

- creating new datafiles or erasing data that existed in previous datafiles
- information creating structures that Oracle requires to access and use the database (the data dictionary)
- creating and initializing the control files and redo log files for the database

6.10 Brain Storm

1. Multiple Database can be mounted on the same instances
   a) True b) False

2. A Process is also reformed as
   a) Thread of control
   b) Mechanism in an Operating System
   c) Job
   d) Task

3. Single user Oracle is a Database system in which
   a) All Oracle code executed by one process
   b) Difference process are used to separate execution of the parts of Oracle and the Client application program
   c) All code of Oracle and the single users Database application is executed by a multiple process.
   d) None of the above

4. A Control file can be altered in
   a) No mount b) Mount c) Restricted d) Open

5. To maximize performance and accumulate many users a multiple process Oracle system uses some additional Oracle process called
   a) Server process b) DBWR c) Background Process d)None of the above

Lecture 7

Start Up and Shut Down

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about startup procedures
- Altering database availability
- Discuss about shutdown procedures
- Describe about parameter files

Coverage Plan

Lecture 7

7.1 Snap Shot

7.2 Startup Procedures

7.3 Altering Database Availability

7.4 Shutdown Procedures

7.5 Using Parameter Files

7.6 Short Summary

7.7 Brain Storm

7.1 Snap Shot

This chapter describes the procedures for starting and stopping an Oracle database, and includes the following topics:

- Startup Procedures
- Altering Database Availability
- Shutdown Procedures
- Using Parameter Files

7.2 Startup Procedures

This section includes the following topics:

- Preparing to Start an Instance
- Starting an Instance: Scenarios

To start up a database or instance, use either the Enterprise Manager Startup Database dialog box or the STARTUP command (after you connect to Oracle with administrator privileges). You can start an instance and database in a variety of ways:

- start the instance without mounting a database
- start the instance and mount the database, but leave it closed
- start the instance, and mount and open the database in:
  - unrestricted mode (accessible to all users)
  - restricted mode (accessible to DBAs only)

Attention: You cannot start a database instance if you are connected to the database via a multi-threaded server process.

In addition, you can force the instance to start, or start the instance and have complete media recovery begin immediately. If your operating system supports the Oracle Parallel Server, you may start an instance and mount the database in either exclusive or shared mode.

Preparing to Start an Instance

There are several tasks you need to perform before you attempt to start an instance.

1. Start Enterprise Manager and connect with administrator privileges.

To start up a database or instance, you must start Enterprise Manager. You must also be connected with administrator privileges.

2. Specify a database name.

When starting a database instance, specify the name of the database that will be mounted to the instance by either:

- **using the STARTUP command and specifying the database name**
- **specifying DB_NAME in the parameter file that starts the instance**

3. Specify the parameter filename.

When starting a database instance, choose a parameter file to initialize the instance's settings:

- using the Startup Database dialog box and entering a filename in the Parameter File text entry field
- using the STARTUP command with the PFILE option and a fully specified filename

Starting an Instance: Scenarios

The following scenarios describe the many ways in which you can start up an instance.

Note: You may encounter problems starting up an instance if control files, database files, or redo log files are not available. If one or more of the files specified by the CONTROL_FILES parameter do not exist or cannot be opened when you attempt to mount a database, Oracle returns a warning message and does not mount the database. If one or more of the datafiles or redo log files are not available or cannot be opened when attempting to open a database, Oracle returns a warning message and does not open the database.

Starting an Instance Without Mounting a Database

You might want to start an instance without mounting a database; this is usually the case only during database creation. To do this, use one of the following options of Enterprise Manager:

- the Startup Database dialog box, selecting the Startup Nomount radio button
- the STARTUP command with the NOMOUNT option

Starting an Instance and Mounting a Database

You might want to start an instance and mount a database, but not open the database because you want to perform specific maintenance operations. For example, the database must be mounted but not open during the following tasks:

- renaming datafiles
- adding, dropping, or renaming redo log files
- enabling and disabling redo log archiving options
- performing full database recovery

Start an instance and mount the database, but leave it closed using one of the following options of Enterprise Manager:

- the Startup database dialog box, selecting the Startup Mount radio button
- the STARTUP command with the MOUNT option

Starting an Instance, and Mounting and Opening a Database

Normal database operation means that an instance is started and the database is mounted and open; this allows any valid user to connect to the database and perform typical data access operations.

Start an instance, and mount and open the database, using one of the following options of Enterprise Manager:

- the Startup Database dialog box, selecting the Startup Open radio button
- the STARTUP command with the OPEN option

Restricting Access to a Database at Startup

You might want to start an instance and mount and open a database in restricted mode so that the database is available only to administrative personnel (not general database users). Use this mode of database startup when you need to accomplish one of the following tasks:

- perform structure maintenance, such as rebuilding indexes
- perform an export or import of database data
- perform a data load (with SQL*Loader)
- temporarily prevent typical users from using data

Typically, all users with the CREATE SESSION system privilege can connect to an open database. Opening a database in restricted mode allows database access only to users with both the CREATE SESSION and RESTRICTED SESSION system privilege; only database administrators should have the RESTRICTED SESSION system privilege.

Start an instance (and, optionally, mount and open the database) in restricted mode using one of the following options of Enterprise Manager:

- the Startup Database dialog box, selecting the Restrict button
- the STARTUP command with the RESTRICT option

Later, you can make the database accessible to users who do not have the RESTRICTED SESSION system privilege.

Forcing an Instance to Start

In unusual circumstances, you might experience problems when attempting to start a database instance. A database instance should not be forced to start unless you are faced with the following:

- The current instance cannot be successfully shut down using either the Normal or Immediate radio buttons of the Shutdown Database dialog box (or an equivalent SHUTDOWN statement).

- You experience problems when starting an instance.

If one of these situations arises, you can usually solve the problem by starting a new instance (and optionally mounting and opening the database) using either of the following options of Enterprise Manager:

- the Startup Database dialog box with the Force button selected
- the STARTUP command with the FORCE option

Starting an Instance, Mounting a Database, and Starting Complete Media Recovery

If you know that media recovery is required, you can start an instance, mount a database to the instance, and have the recovery process automatically start by using the STARTUP command with the RECOVER option.

Starting in Exclusive or Parallel Mode

If your Oracle Server allows multiple instances to access a single database concurrently, you must choose whether to mount the database exclusively or in parallel.

Starting Up an Instance and Database: Example

The following statement starts an instance using the parameter file INITSALE.ORA, mounts and opens the database named SALES in exclusive mode, and restricts access to administrative personnel. The DBA is already connected with administrator privileges.

STARTUP OPEN sales PFILE=INITSALE.ORA EXCLUSIVE RESTRICT;

Automatic Database Startup at Operating System Start

Many sites use procedures to enable automatic startup of one or more Oracle instances and databases immediately following a system start. The procedures for doing this are specific to each operating system.

Starting Remote Instances

If your local Oracle Server is part of a distributed database, you might need to start a remote instance and database. Procedures for starting and stopping remote instances vary widely depending on communication protocol and operating system.

7.3 Altering Database Availability

You can make a database partially available by opening a previously mounted but closed database so that users can connect to and use the database.

The following sections explain how to alter a database's availability:

- Mounting a Database to an Instance
- Opening a Closed Database
- Restricting Access to an Open Database

Mounting a Database to an Instance

When you need to perform specific administrative operations, the database must be started and mounted to an instance, but closed. Starting the instance and mounting the database can accomplish this.

When mounting the database, you can indicate whether to mount the database exclusively to this instance or concurrently to other instances.

To mount a database to a previously started instance, use either of the following options:

- the Mount menu item of Enterprise Manager
- the SQL command ALTER DATABASE with the MOUNT option

Use the following statement when you want to mount a database in exclusive mode:

ALTER DATABASE MOUNT;

Opening a Closed Database

You can make a mounted but closed database available for general use by opening the database. To open a mounted database, use either of the following options:

- the Open menu item of Enterprise Manager
- the SQL command ALTER DATABASE with the OPEN option

Use the following statement to open a mounted database:

ALTER DATABASE OPEN;

After executing this statement, any valid Oracle user with the CREATE SESSION system privilege can connect to the database.

Restricting Access to an Open Database

Under normal conditions, all users with the CREATE SESSION system privilege can connect to an instance. However, you can take an instance in and out of restricted mode. When an instance is in restricted mode, only users who have both the CREATE SESSION and RESTRICTED SESSION system privileges can connect to it. Typically, only administrators have the RESTRICTED SESSION system privilege.

Restricted mode is useful when you need to perform the following tasks:

- perform structure maintenance, such as rebuilding indexes
- perform an export or import of database data
- perform a data load (with SQL*Loader)
- temporarily prevent non-administrator users from using data

To place an instance in restricted mode, use the Restrict menu item of Enterprise Manager or the SQL command ALTER SYSTEM with the ENABLE RESTRICTED SESSION option. After placing an instance in restricted mode, you might want to kill all current user sessions before performing any administrative tasks.

To lift an instance from restricted mode, use the Allow All menu item of Enterprise Manager or the SQL command ALTER SYSTEM with the DISABLE RESTRICTED SESSION option.

7.4 Shutdown Procedures

The following sections describe shutdown procedures:

- Shutting Down a Database Under Normal Conditions
- Shutting Down a Database Immediately

Note: The SHUTDOWN IMMEDIATE statement disconnects all existing idle connections and shuts down the database. If, however, you've submitted processes (for example, inserts, selects or updates) that are awaiting results, the SHUTDOWN IMMEDIATE statement allows the process to complete before disconnecting you.

To initiate database shutdown, use either the Shutdown Database dialog box of Enterprise Manager or the SQL command SHUTDOWN. Control is not returned to the session that initiates a database shutdown until shutdown is complete. Users who attempt connections while a shutdown is in progress receive a message like the following:

ORA-01090: shutdown in progress - connection is not permitted

Attention: You cannot shut down a database if you are connected to the database via a multi-threaded server process.

To shut down a database and instance, you must first be connected with administrator privileges. This condition applies whether you are using Enterprise Manager/GUI or SQL commands.

Figure 7.1shows the sequence of events when the different SHUTDOWN commands are entered during a transfer of funds from one bank account to another.

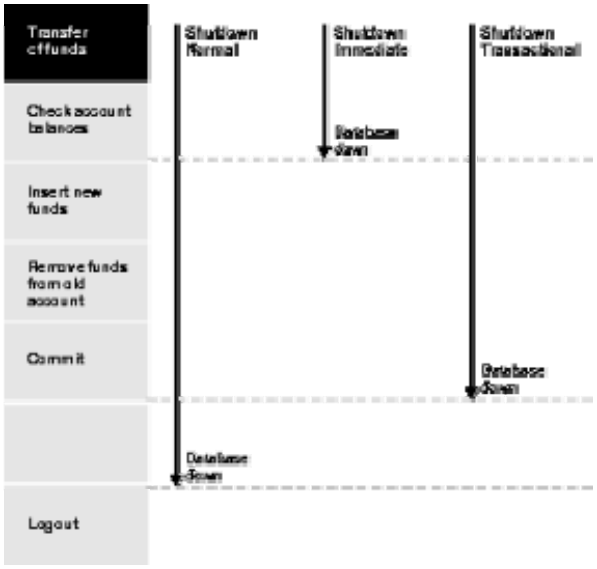


Figure 7.1 : Sequence of Events During Different Types of SHUTDOWN.

Shutting Down a Database under Normal Conditions

Normal database shutdown proceeds with the following conditions:

- No new connections are allowed after the statement is issued.
- Before the database is shut down, Oracle waits for all currently connected users to disconnect from the database.
- The next startup of the database will not require any instance recovery procedures.

To shut down a database in normal situations, use either of the following options of Enterprise Manager:

- the Normal radio button of the Shutdown Database dialog box
- the SHUTDOWN command with the NORMAL option (SHUTDOWN NORMAL;)

Shutting Down a Database Immediately

Use immediate database shutdown only in the following situations:

- A power shutdown is going to occur soon.
- The database or one of its applications is functioning irregularly.

Immediate database shutdown proceeds with the following conditions:

- Current client SQL statements being processed by Oracle are terminated immediately.

- Any uncommitted transactions are rolled back. (If long uncommitted transactions exist, this method of shutdown might not complete quickly, despite its name.)

- Oracle does not wait for users currently connected to the database to disconnect; Oracle implicitly rolls back active transactions and disconnects all connected users.

To shut down a database immediately, use either of the following options of Enterprise Manager:

- the Immediate radio button of the Shutdown database dialog box
- the SHUTDOWN command with the IMMEDIATE option

Note: The SHUTDOWN IMMEDIATE statement disconnects all existing idle connections and shuts down the database. If, however, you've submitted processes (for example, inserts, selects or updates) that are awaiting results, the SHUTDOWN IMMEDIATE statement allows the process to complete before disconnecting you.

Shutdown Transactional

When you wish to perform a planned shutdown of an instance while minimizing interruption to clients, you can use the SHUTDOWN command with the TRANSACTIONAL option:

SHUTDOWN TRANSACTIONAL;

After submitting this statement, no client can start a new transaction on this particular instance. If a client attempts to start a new transaction, they are disconnected. After all transactions have either committed or aborted, any client still connected to the instance is disconnected. At this point, the instance shuts down just as it would when a SHUTDOWN IMMEDIATE statement is submitted.

A transactional shutdown prevents clients from losing work, and at the same time, does not require all users to log off.

Aborting an Instance

You can shut down a database instantaneously by aborting the database's instance. If possible, perform this type of shutdown *only* when in the following situations:

- The database or one of its applications is functioning irregularly *and* neither of the other types of shutdown work.
- You need to shut down the database instantaneously (for example, if you know a power shutdown is going to occur in one minute).
- You experience problems when starting a database instance.

Aborting an instance shuts down a database and yields the following results:

- Current client SQL statements being processed by Oracle are immediately terminated.
- Uncommitted transactions are not rolled back.
- Oracle does not wait for users currently connected to the database to disconnect; Oracle implicitly disconnects all connected users.

If *both* the normal and immediate shutdown options do not work, abort the current database instance immediately by using either of the following options of Enterprise Manager:

- the Abort radio button of the Shutdown Database dialog box
- the SHUTDOWN command with the ABORT option

7.5 Using Parameter Files

The following sections include information about how to use parameter files:

- The Sample Parameter File
- The Number of Parameter Files

To start an instance, Oracle must read a *parameter file*, which is a text file containing a list of instance configuration parameters. Often, although not always, this file is named INIT.ORA or INIT*sid*.ORA, where *sid* is operating system specific.

You can edit parameter values in a parameter file with a basic text editor; however, editing methods are operating system-specific.

Oracle treats string literals defined for National Language Support (NLS) parameters in the file as if they are in the database character set.

The Sample Parameter File

A sample parameter file (INIT.ORA or INIT*sid*.ORA) is included in the Oracle distribution set. This sample file's parameters are adequate for initial installations of an Oracle database. After your system is operating and you have some experience with Oracle, you will probably want to change some parameter values.

The Number of Parameter Files

Each Oracle database has at least one parameter file that corresponds only to that database. This way, database-specific parameters (such as DB_NAME and CONTROL_FILES) in a given file always pertain to a particular database. It is also possible to have several different parameter files for a single database. For example, you can have several different parameter files for a single database so you can optimize the database's performance in different situations.

7.6 Short Summary

Start up process

- ♣ start the instance without mounting a database
- ♣ start the instance and mount the database, but leave it closed
- ♣ start the instance, and mount and open the database

Shut down process

- ♣ No new connections are allowed after the statement is issued.
- ♣ Before the database is shut down, Oracle waits for all currently connected users to disconnect from the database.
- ♣ The next startup of the database will not require any instance recovery procedures.

7.7 Brain Storm

1. SGA gets initialized
   a) When a user executes a query b) After a Database is opened
   c) When an instance startup d) All of the above

2. When a Checkpoint occurs the DBWR signals LGWR to write out buffer
   a) True b) False

3. DB_BLOCK_LRU_LATCHES by default is set to the number of CPU's on your System
   a) True b) False

4. DBWR writes the dirty buffers to the disk with
   a) Multiple single Block Write b) Single Multiple Block c) None of the above

5. Log writer writes (Check all that apply)
   a) When a user process commits a Transaction b) Every 3 seconds
   c) When the redo log buffer is full d) When a online redo log file is full

Lecture 8

Managing Oracle Process

Objectives

After completing this lesson, you should be able to do the following:

- Configuring oracle for dedicated server processes
- Configuring oracle for multi-threaded server processes
- Changing the Minimum Number of Shared Server Processes
- Adding and Removing Dispatcher Processes
- Terminating sessions

Coverage Plan

Lecture 8

8.1 Snap Shot

8.2 Configuring Oracle For Dedicated Server Processes

8.3 Configuring Oracle For Multi-Threaded Server Processes

8.4 Modifying Server Processes

8.5 Terminating Sessions

8.6 Short Summary

8.7 Brain Storm

8.1 Snap Shot

This chapter describes how to manage the processes of an Oracle instance, and includes the following topics:

- Configuring Oracle for Dedicated Server Processes
- Configuring Oracle for Multi-Threaded Server Processes
- Modifying Server Processes

8.2 Configuring Oracle for Dedicated Server Processes

When a user process executes the database application on one machine, and a server process executes the associated Oracle server on another machine, you have separate, distinct processes. The separate server process created on behalf of each user is a *dedicated server process* (see Figure 8.1). Oracle is automatically installed for this configuration. If your operating system can support Oracle in this configuration, it may also support multi-threaded server processes.

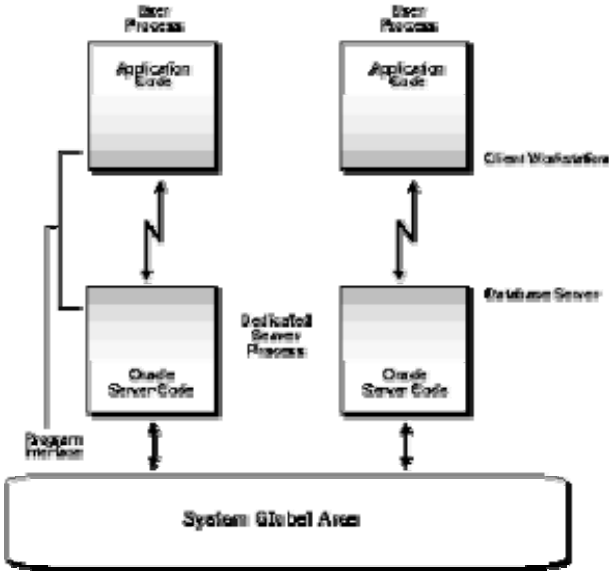


Figure 8.1: Oracle Dedicated Server Processes

To start an instance in a dedicated server configuration, set the following initialization parameters (in the parameter file) to "null", or omit them from the file altogether:

- MTS_SERVICE
- MTS_DISPATCHERS
- MTS_SERVERS
- MTS_LISTENER_ADDRESS

When to Connect to a Dedicated Server Process

If possible, users should connect to an instance via a dispatcher. This keeps the number of processes required for the running instance low. In the following situations, however, users and administrators should explicitly connect to an instance using a dedicated server process:

- to submit a batch job (for example, when a job can allow little or no idle time for the server process)
- to use Enterprise Manager to start up, shut down, or perform media recovery on a database

To request a dedicated server connection, users must include the SRVR=DEDICATED clause in their Net8 TNS connect string.

8.3 Configuring Oracle for Multi-Threaded Server Processes

Consider an order entry system with dedicated server processes. A customer places an order as a clerk enters the order into the database. For most of the transaction, the clerk is on the telephone talking to the customer and the server process dedicated to the clerk's user process remains idle. The server process is not needed during most of the transaction, and the system is slower for other clerks entering orders.

The *multi-threaded server* configuration eliminates the need for a dedicated server process for each connection (see Figure 8.2). A small number of shared server processes can perform the same amount of processing as many dedicated server processes. Also, the amount of memory required for each user is relatively small. Because less memory and process management are required, more users can be supported.

To set up your system in a multi-threaded server configuration, start a network listener process and set the following initialization parameters:

- SHARED_POOL_SIZE
- MTS_LISTENER_ADDRESS
- MTS_SERVICE
- MTS_DISPATCHERS
- MTS_MAX_DISPATCHERS
- MTS_SERVERS

MTS_MAX_SERVERS

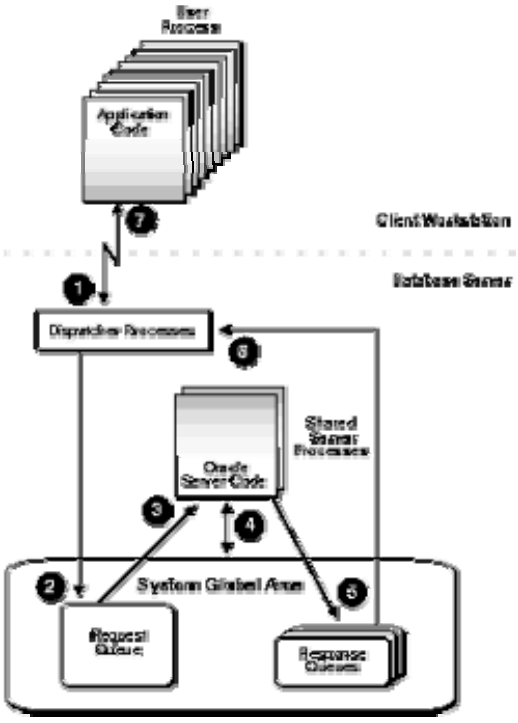


Figure 8.2: Oracle Multi-Threaded Sever Processes

After setting these initialization parameters, restart the instance, which at this point will use the multi-threaded server configuration. The multi-threaded server architecture requires Net8. User processes targeting the multi-threaded server must connect through Net8, even if they are on the same machine as the Oracle instance.

SHARED_POOL_ SIZE: Allocating Additional Space in the Shared Pool for Shared Server

When users connect through the multi-threaded server, Oracle needs to allocate additional space in the shared pool for storing information about the connections between the user processes, dispatchers, and servers. For each user who will connect using the multi-threaded server, add 1K to the setting of the parameter SHARED_POOL_SIZE.

MTS_LISTENER_ ADDRESS: Setting the Listener Process Address

Within the database's parameter file, set the initialization parameter MTS_LISTENER_ADDRESS for each port to which the database will connect. The parameter supports the following syntax:

MTS_LISTENER_ADDRESS = "(*addr*)"

In the syntax above, *addr* is an address at which the listener will listen for connection requests for a specific protocol. The parameter file may contain multiple addresses.

The following examples specify listener addresses:

```
MTS_LISTENER_ADDRESS = "(ADDRESS=(PROTOCOL=tcp)(PORT=5000)\
   (HOST=ZEUS)"
MTS_LISTENER_ADDRESS = "(ADDRESS=(PROTOCOL=decnet)\

   (OBJECT=OUTA)(NODE=ZEUS)"
```

Each address specified in the database's parameter file must also be specified in the corresponding listener's configuration file. You specify addresses differently for various network protocols.

MTS_SERVICE: Specifying Service Names for Dispatchers

Specify the name of the service associated with dispatchers using the parameter MTS_SERVICE. A user requests the multi-threaded server by specifying this service name in the connect string. A service name must be unique; if possible, use the instance's SID (system identifier).

If you do not set the MTS_SERVICE parameter, its value defaults to the DB_NAME parameter. (If DB_NAME is also not set, Oracle returns the error ORA-00114, "missing value for system parameter mts_service," when you start the database.)

If the dispatcher's service name is TEST_DB, the parameter would be set as follows:

```
MTS_SERVICE = "test_db"
```

A connect string for connecting to this dispatcher looks like the following:

```
SQLPLUS scott/tiger@\

 (DESCRIPTION=(ADDRESS=(PROTOCOL=decnet)(NODE=hq)\
    (OBJECT=mts7))(CONNECT_DATA=(SID=test_db)))
```

MTS_DISPATCHERS: Setting the Initial Number of Dispatchers

The number of dispatcher processes started at instance startup is controlled by the parameter MTS_DISPATCHERS. Estimate the number of dispatchers to start for each network protocol before instance startup.

When setting the MTS_DISPATCHERS parameter, you can include any valid protocol.
The appropriate number of dispatcher processes for each instance depends upon the performance you want from your database, the host operating system's limit on the number of connections per process, (which is operating system dependent) and the number of connections required per network protocol.

The instance must be able to provide as many connections as there are concurrent users on the database system; the more dispatchers you have, the better potential database performance users will see, since they will not have to wait as long for dispatcher service.

After instance startup, you can start more dispatcher processes if needed; however, you can only start dispatchers that use protocols mentioned in the database's parameter file. For example, if the parameter file starts dispatchers for protocol_A and protocol_B, you cannot later start dispatchers for protocol_C without changing the parameter file and restarting the instance.

Calculating the Initial Number of Dispatcher Processes

Once you know the number of possible connections per process for your operating system, calculate the initial number of dispatcher processes to create during instance startup, per network protocol, using the following formula Here, *connections per dispatcher* is operating system dependent:

$$\text{number of dispatchers} = \text{CEIL}\left(\frac{\text{maximum number of concurrent sessions}}{\text{connections per dispatcher}}\right)$$

For example, assume that your system typically has 80 users concurrently connected via TCP/IP and 40 users connected via DECNet. In this case, the MTS_DISPATCHERS parameter should be set as follows:

```
MTS_DISPATCHERS = "TCP, 3"
MTS_DISPATCHERS = "DECNET, 3"
```

MTS_MAX_ DISPATCHERS: Setting the Maximum Number of Dispatchers

The parameter MTS_MAX_DISPATCHERS sets the maximum number of dispatcher processes (of all network protocols combined) that can be started for the duration of an instance.

You can create as many dispatcher processes as you need, but the total number of processes, including dispatchers, cannot exceed the host operating system's limit on the number of running processes.

Estimating the Maximum Number of Dispatches

To estimate the maximum number of dispatcher processes an instance will require, use the following formula:

$$\text{MTS\_MAX\_DISPATCHERS} = \frac{\text{maximum number of concurrent sessions}}{\text{connections per dispatcher}}$$

`MTS_SERVERS: Setting the Initial Number of Shared Server Processes`

A number of shared server processes start at instance startup, as determined by the parameter MTS_SERVERS. The appropriate number of initial shared server processes for a database system depends on how many users typically connect to it, and how much processing each user requires. If each user makes relatively few requests over a period of time, then each associated user process is idle for a large percentage of time. In that case, one shared server process can serve 10 to 20 users. If each user requires a significant amount of processing, a higher ratio of server processes to user processes is needed to handle requests.

If you want Oracle to use shared servers, you must set MTS_SERVERS to at least 1. If you omit the parameter or set it to 0, Oracle does not start any shared servers at all. However, you can subsequently set MTS_SERVERS to a number greater than 0 while the instance is running.

It is best to estimate fewer initial shared server processes. Additional shared servers start automatically when needed and are deallocated automatically if they remain idle for too long. However, the initial servers always remain allocated, even if they are idle. If you set the initial number of servers high, your system might incur unnecessary overhead. Experiment with the number of initial shared server processes and monitor shared servers until you find the ideal system performance for typical database activity.

`MTS_MAX_SERVERS: Setting the Maximum Number of Shared Server Processes`

The maximum number of shared server processes that can be started for the duration of an instance is established during instance startup by the parameter MTS_MAX_SERVERS. In general, set this parameter to allow an appropriate number of shared server processes at times of highest activity. Experiment with this limit and monitor shared servers to determine an ideal setting for this parameter.

8.4 Modifying Server Processes

This section describes changes you can make after starting an instance, and includes the following topics:

- Changing the Minimum Number of Shared Server Processes
- Adding and Removing Dispatcher Processes

Changing the Minimum Number of Shared Server Processes

After starting an instance, you can change the minimum number of shared server processes by using the SQL command ALTER SYSTEM.
Oracle eventually terminates dispatchers and servers that are idle longer than the minimum limit you specify.

If you set MTS_SERVERS to 0, Oracle will terminate all current servers when they become idle and will not start any new servers until you increase MTS_SERVERS. Thus, setting MTS_SERVERS to 0 effectively disables the multi-threaded server temporarily.

To control the minimum number of shared server processes, you must have the ALTER SYSTEM privilege.

The following statement sets the number of shared server processes to two:

ALTER SYSTEM SET MTS_SERVERS = 2

Adding and Removing Dispatcher Processes

You can control the number of dispatcher processes in the instance. If the V$QUEUE and V$DISPATCHER views indicate that the load on the dispatcher processes is consistently high, start additional dispatcher processes to route user requests without waiting; you may start new dispatchers until the number of dispatchers equals MTS_MAX_DISPATCHER. In contrast, if the load on dispatchers is consistently low, reduce the number of dispatchers.

To change the number of dispatcher processes, use the SQL command ALTER SYSTEM. Changing the number of dispatchers for a specific protocol has no effect on dispatchers for other protocols.

You can start new dispatcher processes for protocols specified in the MTS_LISTENER_ADDRESS parameter and in the MTS_DISPATCHERS parameter. Therefore, you can add dispatchers only for protocols for which there are dispatchers; to start dispatchers for protocols for which there are currently no dispatchers, shutdown the database, change the parameter file, and restart the database.

If you reduce the number of dispatchers for a particular protocol, the dispatchers are not immediately removed. Rather, Oracle eventually terminates dispatchers that are idle for too long, down to the limit you specify in MTS_DISPATCHERS.

To control the number of dispatcher processes, you must have the ALTER SYSTEM privilege.

The following example adds a dispatcher process where the number of dispatchers was previously three:

```
ALTER SYSTEM
   SET MTS_DISPATCHERS = 'TCPIP,4';
```

8.5 Terminating Sessions

In some situations, you might want to terminate current user sessions. For example, you might want to perform an administrative operation and need to terminate all non-administrative sessions.

This section describes the various aspects of terminating sessions, and includes the following topics:

- Identifying Which Session to Terminate
- Terminating an Active Session
- Terminating an Inactive Session

When a session is terminated, the session's transaction is rolled back and resources (such as locks and memory areas) held by the session are immediately released and available to other sessions.

Terminate a current session using either the Disconnect Session menu item of Enterprise Manager, or the SQL command ALTER SYSTEM...KILL SESSION.

The following statement terminates the session whose SID is 7 and serial number is 15:

ALTER SYSTEM KILL SESSION '7,15';

Identifying Which Session to Terminate

To identify which session to terminate, specify the session's index number and serial number. To identify the index (SID) and serial numbers of a session, query the V$SESSION dynamic performance table.

The following query identifies all sessions for the user JWARD:

```
SELECT sid, serial#
  FROM v$session
  WHERE username = 'JWARD';
SID        SERIAL#   STATUS
---------  --------- --------
    7         15      ACTIVE
   12         63     INACTIVE
```

A session is ACTIVE when it is making an SQL call to Oracle. A session is INACTIVE if it is not making an SQL call to Oracle.

Terminating an Active Session

If a user session is making an SQL call to Oracle (is ACTIVE) when it is terminated, the transaction is rolled back and the user immediately receives the following message:
ORA-00028: your session has been killed

If, after receiving the ORA-00028 message, a user submits additional statements before reconnecting to the database, Oracle returns the following message:

ORA-01012: not logged on

If an active session cannot be interrupted (for example, it is performing network I/O or rolling back a transaction), the session cannot be terminated until the operation completes. In this case, the session holds all resources until it is terminated. Additionally, the session that issues the ALTER SYSTEM statement to terminate a session waits up to 60 seconds for the session to be terminated; if the operation that cannot be interrupted continues past one minute, the issuer of the ALTER SYSTEM statement receives a message indicating that the session has been "marked" to be terminated. A session marked to be terminated is indicated in V$SESSION with a status of "KILLED" and a server that is something other than "PSEUDO."

Terminating an Inactive Session

If the session is not making an SQL call to Oracle (is INACTIVE) when it is terminated, the ORA-00028 message is not returned immediately. The message is not returned until the user subsequently attempts to use the terminated session.

When an inactive session has been terminated, STATUS in the view V$SESSION is "KILLED." The row for the terminated session is removed from V$SESSION after the user attempts to use the session again and receives the ORA-00028 message.

In the following example, the DBA terminates an inactive session:

```
SVRMGR>   SELECT sid,serial#,status,server
     2> FROM v$session
     3> WHERE username = 'JWARD';

SID        SERIAL#  STATUS    SERVER
---------- -------- --------- ---------
      7      15      INACTIVE  DEDICATED

     12     63      INACTIVE  DEDICATED

2 rows selected.
```

```
SVRMGR> ALTER SYSTEM KILL SESSION '7,15';

Statement processed.

SVRMGR> SELECT sid, serial#, status, server
     2>    FROM v$session
     3>    WHERE username = 'JWARD';

SID       SERIAL#  STATUS    SERVER
--------- -------- --------- ---------
    7       15     KILLED     PSEUDO
   12       63    INACTIVE    DEDICATED

1 rows selected.
```

8.6 Short Summary

The *multi-threaded server* configuration eliminates the need for a dedicated server process for each connection. A small number of shared server processes can perform the same amount of processing as many dedicated server processes. Also, the amount of memory required for each user is relatively small. Because less memory and process management are required, more users can be supported.

8.7 Brain Storm

1. What causes Row Migration?

   a) When PCTFREE is set extremely high
   b) When an update requires more space than is currently available on the block
   c) When a block is chained
   d) When PCTUSED is set extremely high.

2. To determine the storage allocation for temporary segments, the DBA can access which of the following views?

3. ALTER TABLE TABLE DEALLOCATE UNUSED KEEP integer, what does the command do?

4. What is the privilege needed to take a tablespace offline?

Lecture 9

Managing Online Redo Log

Objectives

After completing this lesson, you should be able to do the following:

- Planning the online redo log
- Create, rename, relocate, retrieve and clearing online redo log groups and members
- Deletion of online redo log groups and members
- Restriction for clearing inline redo log file

Coverage Plan

Lecture 9

9.1 Snap Shot

9.2 Planning The Online Redo Log

9.3 Multiplex The Online Redo Log

9.4 Creating Online Redo Log Groups And Members

9.5 Renaming And Relocating Online Redo Log Members

9.6 Dropping Online Redo Log Groups

9.7 Dropping Online Redo Log Members

9.8 Clearing An Online Redo Log File

9.9 Restrictions

9.10 Listing Information About The Online Redo Log

9.11 Short Summary

9.12 Brain Storm

9.1 Snap Shot

This chapter explains how to manage the online redo log, and includes the following topics:

- Planning the Online Redo Log
- Creating Online Redo Log Groups and Members
- Renaming and Relocating Online Redo Log Members
- Dropping Online Redo Log Groups
- Dropping Online Redo Log Members
- Clearing an Online Redo Log File
- Listing Information about the Online Redo Log

9.2 Planning the Online Redo Log

Every instance of an Oracle database has an associated *online redo log*, which is a set of two or more online log files that record all committed changes made to the database. Online redo logs serve to protect the database in the event of an instance failure. 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.

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.

This section describes guidelines you should consider when configuring a database instance's online redo log, and includes the following topics:

- Multiplex the Online Redo Log
- Place Online Redo Log Members on Different Disks
- Set the Size of Online Redo Log Members
- Choose an Appropriate Number of Online Redo Log Files

9.3 Multiplex the Online Redo Log

The online redo log of a database instance should consist of multiplexed groups of online redo log files. Furthermore, members in the same group should be stored on separate disks so that no single disk failure can cause LGWR and the database instance to fail.

To avoid losing a database due to a single point of failure, Oracle can maintain multiple sets of on-line redo log files. A *multiplex 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.
Warning: Although the Oracle Server allows multiplexed groups to contain different numbers of members, this state should only be the temporary result of an abnormal situation such as a disk failure damaging a member of a group. If any group contains only one member, the failure of the disk containing that member could cause Oracle to halt.

While multiplexed groups require extra storage space, the cost of this space is usually insignificant compared to the potential cost of lost data (if a disk failure destroys a non-multiplexed online redo log).

Place Online Redo Log Members on Different Disks

When setting up a multiplex online redo log, place members of a group on different disks. This way, if a single disk fails, only one member of a group becomes unavailable to LGWR and other members remain accessible to LGWR, so the instance can continue to function.

If you archive the redo log, spread online redo log members across disks to eliminate contention between the LGWR and ARCH background processes. For example, if you have two groups of duplexed online redo log members, place each member on a different disk and set your archiving destination to a fifth disk. This way, there is never contention between LGWR (writing to the members) and ARCH (reading the members).

Datafiles and online redo log files should also be on different disks to reduce contention in writing data blocks and redo entries.

Set the Size of Online Redo Log Members

When setting the size of online redo log files, consider whether you will be archiving the redo log. Online redo log files should be sized so that a filled group can be archived to a single unit of offline storage media (such as a tape or disk), with the least amount of space on the medium left unused. For example, suppose only one filled online redo log group can fit on a tape and 49% of the tape's storage capacity remains unused. In this case, it would be better to decrease the size of the online redo log files slightly, so that two log groups could be archived per tape.

With multiplex groups of online redo logs, all members of the same group must be the same size. Members of different groups can have different sizes; however, there is no advantage in varying file size between groups. If checkpoints are not set to occur between log switches, make all groups the same size to guarantee that checkpoints occur at regular intervals.

Choose an Appropriate Number of Online Redo Log Files

The best way to determine the appropriate number of online redo log files for a database instance is to test different configurations. The optimum configuration has the fewest groups possible without hampering LGWR's writing redo log information.

In some cases, a database instance may require only two groups. In other situations, a database instance may require additional groups to guarantee that a recycled group is always available to LGWR. During testing, the easiest way to determine if the current online redo log configuration is satisfactory is to examine the contents of the LGWR trace file and the database's ALERT file. If messages indicate that LGWR frequently has to wait for a group because a checkpoint has not completed or a group has not been archived, add groups.

Consider the parameters that can limit the number of online redo log files before setting up or altering the configuration of an instance's online redo log. The following three parameters limit the number of online redo log files that you can add to a database:

- The MAXLOGFILES parameter used in the CREATE DATABASE statement determines the maximum number of groups of online redo log files per database; group values can range from 1 to MAXLOGFILES. The only way to override this upper limit is to re-create the database or its control file; thus, it is important to consider this limit *before* creating a database. If MAXLOGFILES is not specified for the CREATE DATABASE statement, Oracle uses an operating system default value.

- The LOG_FILES parameter (in the parameter file) can temporarily decrease the maximum number of groups of online redo log files for the duration of the current instance. However, LOG_FILES cannot override MAXLOGFILES to increase the limit. If LOG_FILES is not set in the database's parameter file, Oracle uses an operating system-specific default value.

- The MAXLOGMEMBERS parameter used in the CREATE DATABASE statement determines the maximum number of members per group. As with MAXLOGFILES, the only way to override this upper limit is to re-create the database or control file; thus, it is important to consider this limit *before* creating a database. If no MAXLOGMEMBERS parameter is specified for the CREATE DATABASE statement, Oracle uses an operating system default value.

9.4 Creating Online Redo Log Groups and Members

You can create groups and members of online redo log files during or after database creation. If you can, plan the online redo log of a database and create all required groups and members of online redo log files during database creation. To create new online redo log groups and members, you must have the ALTER DATABASE system privilege.

In some cases, you might need to create additional groups or members of online redo log files. For example, adding groups to an online redo log can correct redo log group availability problems. A database can have up to MAXLOGFILES groups.

Creating Online Redo Log Groups

To create a new group of online redo log files, use either the Add Logfile Group property sheet of Enterprise Manager, or the SQL command ALTER DATABASE with the ADD LOGFILE parameter.

The following statement adds a new group of redo logs to the database:

```
ALTER DATABASE
   ADD LOGFILE ('log1c', 'log2c') SIZE 500K;
```

Note: Fully specify filenames of new log members to indicate where the operating system file should be created; otherwise, the file is created in the default directory of the database server, which is operating system-dependent. If you want to reuse an existing operating system file, you do not have to indicate the file size.

Using the ALTER DATABASE statement with the ADD LOGFILE option, you can specify the number that identifies the group with the GROUP option:

```
ALTER DATABASE
   ADD LOGFILE GROUP 10 ('log1c', 'log2c') SIZE 500K;
```

Using group numbers can make administering redo log groups easier. However, the group number must be between 1 and MAXLOGFILES; do not skip redo log file group numbers (that is, do not number your groups 10, 20, 30, and so on), or you will consume unnecessary space in the control files of the database.

Creating Online Redo Log Members

In some cases, you might not need to create a complete group of online redo log files; the group may already exist, but not be complete because one or more members of the group were dropped (for example, because of a disk failure). In this case, you can add new members to an existing group.

To create new online redo log members for an existing group, use the Add Logfile Member property sheet of Enterprise Manager, or the SQL command ALTER DATABASE with the ADD LOG MEMBER parameter.

The following statement adds a new redo log member to redo log group number 2:

```
ALTER DATABASE
   ADD LOGFILE MEMBER 'log2b' TO GROUP 2;
```

Notice that filenames must be specified, but sizes need not be; the size of the new members is determined from the size of the existing members of the group.

When using the ALTER DATABASE command, you can alternatively identify the target group by specifying all of the other members of the group in the TO parameter, as shown in the following example:

```
ALTER DATABASE
   ADD LOGFILE MEMBER 'log2c' TO ('log2a', 'log2b');
```

Note: Fully specify the filenames of new log members to indicate where the operating system files should be created; otherwise, the files will be created in the default directory of the database server.

9.5 Renaming and Relocating Online Redo Log Members

You can rename online redo log members to change their locations. This procedure is necessary, for example, if the disk currently used for some online redo log files is going to be removed, or if datafiles and a number of online redo log files are stored on the same disk and should be separated to reduce contention.

To rename online redo log members, you must have the ALTER DATABASE system privilege. Additionally, you might also need operating system privileges to copy files to the desired location and privileges to open and back up the database.

Before renaming any online redo log members, ensure that the new online redo log files already exist.

Warning: The following steps only modify the internal file pointers in a database's control files; they do not physically rename or create any operating system files. Use your computer's operating system to copy the existing online redo log files to the new location.

Rename online redo log members with the Rename Logfile Member property sheet of Enterprise Manager, or the SQL command ALTER DATABASE with the RENAME FILE parameter.

1. Back up the database.

Before making any structural changes to a database, such as renaming or relocating online redo log members, completely back up the database (including the control file) in case you experience any problems while performing this operation.

2. Copy the online redo log files to the new location.

Operating system files, such as online redo log members, must be copied using the appropriate operating system commands. See your operating system manual for more information about copying files.

Suggestion: You can execute an operating system command to copy a file without exiting Enterprise Manager. Use the Enterprise Manager HOST command.

3. Rename the online redo log members.

Use the Rename Online Redo Log Member dialog box, or the ALTER DATABASE command with the RENAME FILE clause to rename the database's online redo log files.

4. Open the database for normal operation.

The online redo log alterations take effect the next time that the database is opened. Opening the database may require shutting down the current instance (if the database was previously opened by the current instance) or just opening the database using the current instance.

5. Back up the control file.

As a precaution, after renaming or relocating a set of online redo log files, immediately back up the database's control file.

The following example renames the online redo log members. However, first assume that:

- The database is currently mounted by, but closed to, the instance.
- The online redo log is duplexed: one group consists of the members LOG1A and LOG1B, and the second group consists of the members LOG2A and LOG2B. The files LOG1A and LOG2A are stored on Disk A, while LOG1B and LOG2B are stored on Disk B.
- The online redo log files located on Disk A must be relocated to Disk C. The new filenames will reflect the new location: LOG1C and LOG2C.

The files LOG1A and LOG2A on Disk A must be copied to the new files LOG1C and LOG2C on Disk C.

```
ALTER DATABASE
   RENAME FILE 'log1a', 'log2a'
      TO 'log1c', 'log2c';
```

9.6 Dropping Online Redo Log Groups

In some cases, you might want to drop an entire group of online redo log members. For example, you might want to reduce the number of groups in an instance's online redo log.

To drop an online redo log group, you must have the ALTER DATABASE system privilege.

Before dropping an online redo log group, consider the following restrictions and precautions:

- An instance requires at least two groups of online redo log files, regardless of the number of members in the groups. (A group is one or more members.)
- You can drop an online redo log group only if it is not the active group. If you need to drop the active group, first force a log switch to occur;
- Make sure an online redo log group is archived (if archiving is enabled) before dropping it. To see whether this has happened, use the Enterprise Manager ARCHIVE LOG command with the LIST parameter.

Drop an online redo log group with either the Drop Logfile Group menu item of Enterprise Manager, or the SQL command ALTER DATABASE with the DROP LOGFILE clause.

The following statement drops redo log group number 3:

ALTER DATABASE DROP LOGFILE GROUP 3;

When an online redo log group is dropped from the database, the operating system files are not deleted from disk. Rather, the control files of the associated database are updated to drop the members of the group from the database structure. After dropping an online redo log group, make sure that the drop completed successfully, and then use the appropriate operating system command to delete the dropped online redo log files.

9.7 Dropping Online Redo Log Members

In some cases, you might want to drop one or more specific online redo log members. For example, if a disk failure occurs, you might need to drop all the online redo log files on the failed disk so that Oracle does not try to write to the inaccessible files. In other situations, particular online redo log files become unnecessary; for example, a file might be stored in an inappropriate location.

To drop an online redo log member, you must have the ALTER DATABASE system privilege.

Consider the following restrictions and precautions before dropping individual online redo log members:

- It is all right to drop online redo log files so that a multiplexed online redo log becomes temporarily asymmetric. For example, if you use duplexed groups of online redo log files, you can drop one member of one group, even though all other groups have two members each. However, you should rectify this situation immediately so that all groups have at least two members, and thereby eliminate the single point of failure possible for the online redo log.

- An instance always requires at least two valid groups of online redo log files, regardless of the number of members in the groups. (A group is one or more members.) If the member you want to drop is the last valid member of the group, you cannot drop the member until the other members become valid; to see a redo log file's status, use the V$LOGFILE view. A redo log file becomes INVALID if Oracle cannot access it. It becomes STALE if Oracle suspects that it is not complete or correct; a stale log file becomes valid again the next time its group is made the active group.
- You can drop an online redo log member only if it is not part of an active group. If you want to drop a member of an active group, first force a log switch to occur.

- Make sure the group to which an online redo log member belongs is archived (if archiving is enabled) before dropping the member. To see whether this has happened, use the Enterprise Manager ARCHIVE LOG command with the LIST parameter.

To drop specific inactive online redo log members, use either the Drop Logfile Member menu item of Enterprise Manager, or the SQL command ALTER DATABASE command with the DROP LOGFILE MEMBER clause.

The following statement drops the redo log LOG3C:

```
ALTER DATABASE DROP LOGFILE MEMBER 'log3c';
```

When an online redo log member is dropped from the database, the operating system file is not deleted from disk. Rather, the control files of the associated database are updated to drop the member from the database

structure. After dropping an online redo log file, make sure that the drop completed successfully, and then use the appropriate operating system command to delete the dropped online redo log file.

9.8 Clearing an Online Redo Log File

If you have enabled redo log block checking, Oracle verifies each block before archiving it. If a particular redo log block is corrupted in all members of a group, archiving stops. Eventually all the redo logs become filled and database activity is halted, until archiving can resume.

In this situation, you can use the SQL command ALTER DATABASE... CLEAR LOGFILE to clear the corrupted redo logs and avoid archiving them. The cleared redo logs are available for use even though they were not archived.

The following statement clears the log files in redo log group number 3:

```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE
   GROUP 3;
```

9.9 Restrictions

You can clear a redo log file whether it is archived or not. However, when it is not archived, you must include the keyword UNARCHIVED.

If you clear a log file that is needed for recovery of a backup, then you can no longer recover from that backup. Oracle writes a message in the alert log describing the backups from which you cannot recover.
Attention: If you clear an unarchived redo log file, you should take another backup of the database.

If you want to clear an unarchived redo log that is needed to bring an offline tablespace online, you must use the clause UNRECOVERABLE DATAFILE in the ALTER DATABASE command.

If you clear a redo log needed to bring an offline tablespace online, you will not be able to bring the tablespace online again. You will have to drop the tablespace or perform an incomplete recovery.

9.10 Listing Information about the Online Redo Log

Use the VLOG, VLOGFILE, and V$THREAD views to see information about the online redo log of a database; the V$THREAD view is of particular interest for Parallel Server administrators.

The following query returns information about the online redo log of a database used without the Parallel Server:

```
SELECT group#, bytes, members
   FROM sys.v$log;

GROUP#        BYTES        MEMBERS
----------    ----------   ----------
         1      81920       2
         2      81920       2
```

To see the names of all of the member of a group, use a query similar to the following:

```
SELECT *
   FROM sys.v$logfile
```

```
   WHERE group# = 2;

GROUP#         BYTES       MEMBERS
----------    ----------  ---------
2               LOG2A
2               STALE          LOG2B
2               LOG2C
```

If STATUS is blank for a member, the file is in use.

9.11 Short Summary

Every instance of an Oracle database has an associated *online redo log*, which is a set of two or more online log files that record all committed changes made to the database.

A *multiplex 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.

9.12 Brain Storm

1. What should be the minimum number of online redo log files should an instance have? Can groups have varying number of members?

2. What privilege must you have to drop an online redo log member?

3. The datafiles of a particular kind of Tablespace cannot be renamed or resized? What is that tablespace?

4. Every Datafile has two associated file numbers. What are they? And what is the difference between them?

5. What is a configuration File?

Lecture 10

Managing Control Files

Objectives

After completing this lesson, you should be able to do the following:

- Guidelines for creating control files
- Creating initial and additional copies of control files
- Renaming and relocating control files
- Deletion of control files

Coverage Plan

Lecture 10

10.1 Snap Shot

10.2 Guidelines For Control Files

10.3 Multiplex Control Files

10.4 Creating Control Files

10.5 Relocating Control Files

10.6 Dropping Control Files

10.7 Short Summary

10.8 Brain Storm

10.1 Snap Shot

This chapter explains how to create and maintain the control files for your database, and includes the following topics:

- Guidelines for Control Files
- Creating Control Files
- Troubleshooting After Creating Control Files
- Dropping Control Files

10.2 Guidelines for Control Files

This section describes guidelines you can use to manage the control files for a database, and includes the following topics:

- Name Control Files
- Multiplex Control Files on Different Disks
- Place Control Files Appropriately
- Manage the Size of Control Files

Name Control Files

Assign control file names via the CONTROL_FILES initialization parameter in the database's parameter file. CONTROL_FILES indicates one or more names of control files separated by commas. The instance startup procedure recognizes and opens all the listed files. The instance maintains all listed control files during database operation.

During database operation, Oracle Server writes to all necessary files listed for the CONTROL_FILES parameter.

10.3 Multiplex Control Files

Every Oracle database should have at least two control files, each stored on a different disk. If a control file is damaged due to a disk failure, the associated instance must be shut down. Once the disk drive is repaired, the damaged control file can be restored using an intact copy of the control file and the instance can be restarted; no media recovery is required.

Behavior of Multiplexed Control Files

The following list describes the behavior of multiplexed control files:

- Two or more filenames are listed for the initialization parameter CONTROL_FILES in the database's parameter file.
- The first file listed in the CONTROL_FILES parameter is the only file read by the Oracle Server during database operation.

- If any of the control files become unavailable during database operation, the instance becomes inoperable and should be aborted.

The only disadvantage of having multiple control files is that all operations that update the control files (such as adding a datafile or checkpointing the database) can take slightly longer. However, this difference is usually

insignificant (especially for operating systems that can perform multiple, concurrent writes) and does not justify using only a single control file.

Attention: Oracle strongly recommends that your database has a minimum of two control files on different disks**.**

Place Control Files Appropriately

Each copy of a control file should be stored on a different disk drive. Furthermore, a control file copy should be stored on every disk drive that stores members of online redo log groups, if the online redo log is multiplexed. By storing control files in these locations, you minimize the risk that all control files and all groups of the online redo log will be lost in a single disk failure.

Manage the Size of Control Files

The main determinants of a control file's size are the values set for the MAXDATAFILES, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, and MAXINSTANCES parameters in the CREATE DATABASE statement that created the associated database. Increasing the values of these parameters increases the size of a control file of the associated database.

10.4 Creating Control Files

Every Oracle database has a *control file*. A control files records the physical structure of the database and contains:

- the database name
- names and locations of associated databases and online redo log files
- the timestamp of the database creation
- the current log sequence number
- checkpoint information

The control file of an Oracle database is created at the same time as the database. By default, at least one copy of the control file must be created during database creation. On some operating systems, Oracle creates multiple copies. You should create two or more copies of the control file during database creation. You might also need to create control files later, if you lose control files or want to change particular settings in the control files.
This section describes ways to create control files, and includes the following topics:

- Creating Initial Control Files
- Creating Additional Copies of the Control File, and Renaming and Relocating Control Files
- New Control Files
- Creating New Control Files

Creating Initial Control Files

You create the initial control files of an Oracle database by specifying one or more control filenames in the CONTROL_FILES parameter in the parameter file used during database creation. The filenames specified in CONTROL_FILES should be fully specified. Filename specification is operating system-specific.

If files with the specified names currently exist at the time of database creation, you must specify the CONTROLFILE REUSE parameter in the CREATE DATABASE command, or else an error occurs. Also, if the

size of the old control file differs from that of the new one, you cannot use the REUSE option. The size of the control file changes between some release of new version of Oracle, as well as when the number of files specified in the control file changes; configuration parameters such as MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES affect control file size.

If you do not specify files for CONTROL_FILES before database creation, Oracle uses a default filename. The default name is also operating system-specific.

You can subsequently change the value of the CONTROL_FILES parameter to add more control files or to change the names or locations of existing control files.

10.5 Relocating Control Files

You add a new control file by copying an existing file to a new location and adding the file's name to the list of control files.

Similarly, you rename an existing control file by copying the file to its new name or location, and changing the file's name in the control file list.

In both cases, to guarantee that control files do not change during the procedure, shut down the instance before copying the control file.

To Multiplex or Move Additional Copies of the Current Control Files

1. Shut down the database.
2. Exit Enterprise Manager.
3. Copy an existing control file to a different location, using operating system commands.
4. Edit the CONTROL_FILES parameter in the database's parameter file to add the new control file's name, or to change the existing control filename.
5. Restart Enterprise Manager.
6. Restart the database.

New Control Files

You can create a new control file for a database using the CREATE CONTROLFILE command. This is recommended in the following situations:

- All control files for the database have been permanently damaged and you do not have a control file backup.

- You want to change one of the permanent database settings originally specified in the CREATE DATABASE statement, including the database's name, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES.

For example, you might need to change a database's name if it conflicts with another database's name in a distributed environment. As another example, you might need to change one of the previously mentioned parameters if the original setting is too low.

The following statement creates a new control file for the PROD database (formerly a database that used a different database name):

```
CREATE CONTROLFILE
    SET DATABASE prod
    LOGFILE GROUP 1 ('logfile1A', 'logfile1B') SIZE 50K,
       GROUP 2 ('logfile2A', 'logfile2B') SIZE 50K
    NORESETLOGS
    DATAFILE 'datafile1' SIZE 3M, 'datafile2' SIZE 5M
    MAXLOGFILES 50
    MAXLOGMEMBERS 3
    MAXDATAFILES 200
    MAXINSTANCES 6
    ARCHIVELOG;
```

Warning: The CREATE CONTROLFILE command can potentially damage specified datafiles and online redo log files; omitting a filename can cause loss of the data in that file, or loss of access to the entire database. Employ caution when using this command and be sure to follow the steps in the next section.

Creating New Control Files

This section provides step-by-step instructions for creating new control files.
To Create New Control Files

1. Make a list of all datafiles and online redo log files of the database.

If you followed the recommendations for database backups, you should already have a list of datafiles and online redo log files that reflect the current structure of the database.

If you have no such lists and your control file has been damaged so that the database cannot be opened, try to locate all of the datafiles and online redo log files that constitute the database. Any files not specified in Step 5 are not recoverable once a new control file has been created. Moreover, if you omit any of the files that make up the SYSTEM tablespace, you might not be able to recover the database.

2. Shut down the database.

If the database is open, shut down the database with normal priority, if possible. Use the IMMEDIATE or ABORT options only as a last resort.

3. Back up all datafiles and online redo log files of the database.

4. Start up an new instance, but do not mount or open the database.

5. Create a new control file for the database using the CREATE CONTROLFILE command.

When creating the new control file, select the RESETLOGS option if you have lost any online redo log groups in addition to the control files. In this case, you will need to recover from the loss of the redo logs (Step 8). You must also specify the RESETLOGS option if you have renamed the database. Otherwise, select the NORESETLOGS option.

6. Store a backup of the new control file on an offline storage device.

7. Edit the parameter files of the database.

Edit the parameter files of the database to indicate all of the control files created in Step 5 and Step 6 (not including the backup control file) in the CONTROL_FILES parameter.

8. Recover the database if necessary.

If you are creating the control file as part of recovery, recover the database. If the new control file was created using the NORESETLOGS option (Step 5), you can recover the database with complete, closed database recovery.

If the new control file was created using the RESETLOGS option, you must specify USING BACKUP CONTROL FILE. If you have lost online or archived redo logs or datafiles, use the procedures for recovering those files.

9. Open the database.

Open the database using one of the following methods:

- If you did not perform recovery, open the database normally.
- If you performed complete, closed database recovery in Step 8, use the Startup Open radio button of the Startup Database dialog box of Enterprise Manager.
- If you specified RESETLOGS when creating the control file, use the ALTER DATABASE command, indicating RESETLOGS.

The database is now open and available for use.

10.6 Dropping Control Files

You can drop control files from the database. For example, you might want to do so if the location of a control file is inappropriate. Remember that the database must have at least two control files at all times.

1. Shut down the database.
2. Exit Enterprise Manager.
3. Edit the CONTROL_FILES parameter in the database's parameter file to delete the old control file's name.
4. Restart Enterprise Manager.
5. Restart the database.

Warning: This operation does not physically delete the unwanted control file from the disk. Use operating system commands to delete the unnecessary file after you have dropped the control file from the database.

10.7 Short Summary

Every Oracle database has a *control file*. A control files records the physical structure of the database and contains:

- the database name
- names and locations of associated databases and online redo log files
- the timestamp of the database creation
- the current log sequence number
- checkpoint information

10.8 Brain Storm

1. What should be the minimum number of online redo log files should an instance have? Can groups have varying number of members?

2. What privilege must you have to drop an online redo log member?

3. The datafiles of a particular kind of Tablespace cannot be renamed or resized? What is that tablespace?

4. Every Datafile has two associated file numbers. What are they? And what is the difference between them?

5. What is a configuration File?

6. If a log member becomes unavailable on the Database

   a) The instance will fail
   b) The instance will continue to run, but media recovery is needed
   c) The Database will continue to remain open, but instance recovery is needed
   d) The system will continue as normal

Lecture 11

Managing Job Queues

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about multiple SNP background processes and starting SNP processes
- How to managing job queues
- Discuss about the termination of job
- Viewing job queue information

Coverage Plan

Lecture 11

11.1 Snap Shot

11.2 SNP Background Processes

11.3 Managing Job Queues

11.4 Terminating a Job

11.5 Viewing Job Queue Information

11.6 Short Summary

11.7 Brain Storm

11.1 Snap Shot

This chapter describes how to use job queues to schedule periodic execution of PL/SQL code, and includes the following topics:

- SNP Background Processes
- Managing Job Queues
- Viewing Job Queue Information

11.2 SNP Background Processes

This section describes SNP background processes and their role in managing job queues, and includes the following topics:

- Multiple SNP processes
- Starting up SNP processes

You can schedule routines to be performed periodically using the job queue. A routine is any PL/SQL code. To schedule a job, you submit it to the job queue and specify the frequency at which the job is to be run. You can also alter, disable, or delete jobs you have submitted.

To maximize performance and accommodate many users, a multi-process Oracle system uses some additional processes called *background processes*. Background processes consolidate functions that would otherwise be handled by multiple Oracle programs running for each user process. Background processes asynchronously perform I/O and monitor other Oracle processes to provide increased parallelism for better performance and reliability.

SNP background processes execute job queues. SNP processes periodically wake up and execute any queued jobs that are due to be run. You must have at least one SNP process running to execute your queued jobs in the background.

SNP background processes differ from other Oracle background processes, in that the failure of an SNP process does not cause the instance to fail. If an SNP process fails, Oracle restarts it.

Multiple SNP processes

An instance can have up to thirty-six SNP processes, named SNP0 to SNP9, and SNPA to SNPZ. If an instance has multiple SNP processes, the task of executing queued jobs can be shared across these processes, thus improving performance. Note, however, that each job is run at any point in time by only one process. A single job cannot be shared simultaneously by multiple SNP processes.

Starting up SNP processes

Job queue initialization parameters enable you to control the operation of the SNP background processes. When you set these parameters in the initialization parameter file for an instance, they take effect the next time you start the instance.

11.3 Managing Job Queues

This section describes the various aspects of managing job queues, and includes the following topics:

- DBMS_JOB Package
- Submitting a Job to the Job Queue
- How Jobs Execute
- Removing a Job From the Job Queue
- Altering a Job
- Broken Jobs
- Forcing a Job to Execute
- Terminating a Job

DBMS_JOB Package

To schedule and manage jobs in the job queue, use the procedures in the DBMS_JOB package. There are no database privileges associated with using job queues. Any user who can execute the job queue procedures can use the job queue.

Procedure	Description
SUBMIT	Submits a job to the job queue.
REMOVE	Removes specified job from the job queue.
CHANGE	Alters a specified job. You can alter the job description, the time at which the job will be run, or the interval between executions of the job.
WHAT	Alters the job description for a specified job.
NEXT_DATE	Alters the next execution time for a specified job.
INTERVAL	Alters the interval between executions for a specified job.
BROKEN	Disables job execution. If a job is marked as broken, Oracle does not attempt to execute it.
RUN	Forces a specified job to run.

Table 1 : Procedures in the DBMS_JOB Package

Submitting a Job to the Job Queue

To submit a new job to the job queue, use the SUBMIT procedure in the DBMS_JOB package:

```
DBMS_JOB.SUBMIT(  job             OUT    BINARY_INTEGER,
                  what            IN     ARCHAR2,
                  next_date       IN     DATE DEFAULT SYSDATE,
                  interval        IN     VARCHAR2 DEFAULT 'null',
                  no_parse        IN      BOOLEAN DEFAULT FALSE)
```

The SUBMIT procedure returns the number of the job you submitted. Table 2 describes the procedure's parameters.

Parameter	Description
job	This is the identifier assigned to the job you created. You must use the job number whenever you want to alter or remove the job.
What	This is the PL/SQL code you want to have executed.

next_date	This is the next date when the job will be run. The default value is SYSDATE.
interval	This is the date function that calculates the next time to execute the job. The default value is NULL. INTERVAL must evaluate to a future point in time or NULL.
no_parse	This is a flag. The default value is FALSE.
	If NO_PARSE is set to FALSE (the default), Oracle parses the procedure associated with the job. If NO_PARSE is set to TRUE, Oracle parses the procedure associated with the job the first time that the job is executed. If, for example, you want to submit a job before you have created the tables associated with the job, set NO_PARSE to TRUE.

Table 2: Parameters for DBMS_JOB.SUBMIT

As an example, let's submit a new job to the job queue. The job calls the procedure DBMS_DDL.ANALYZE_OBJECT to generate optimizer statistics for the table DQUON.ACCOUNTS. The statistics are based on a sample of half the rows of the ACCOUNTS table. The job is run every 24 hours:

```
SVRMGR>   VARIABLE jobno number;
SVRMGR>   begin
          2>           DBMS_JOB.SUBMIT(:jobno,
          3>                 'dbms_ddl.analyze_object(''TABLE'',
          4>                   ''DQUON'', ''ACCOUNTS'',
          5>                   ''ESTIMATE'', NULL, 50);'
          6>                   SYSDATE, 'SYSDATE + 1');
          7>            commit;
          8>  end;
          9>  /
Statement processed.
SVRMGR> print jobno

JOBNO
----------
     14144
```

Job Environment

When you submit a job to the job queue or alter a job's definition, Oracle records the following environment characteristics:

- the current user
- the user submitting or altering a job
- the current schema
- MAC privileges (if appropriate)

Oracle also records the following NLS parameters:

- NLS_LANGUAGE
- NLS_TERRITORY
- NLS_CURRENCY
- NLS_ISO_CURRENCY
- NLS_NUMERIC_CHARACTERS
- NLS_DATE_FORMAT

- NLS_DATE_LANGUAGE
- NLS_SORT

Oracle restores these environment characteristics every time a job is executed. NLS_LANGUAGE and NLS_TERRITORY parameters are defaults for unspecified NLS parameters.

You can change a job's environment by using the DBMS_SQL package and the ALTER SESSION command.

Jobs and Import/Export

Jobs can be exported and imported. Thus, if you define a job in one database, you can transfer it to another database. When exporting and importing jobs, the job's number, environment, and definition remain unchanged.

Note: If the job number of a job you want to import matches the number of a job already existing in the database, you will not be allowed to import that job. Submit the job as a new job in the database.

Job Owners

When you submit a job to the job queue, Oracle identifies you as the owner of the job. Only a job's owner can alter the job, force the job to run, or remove the job from the queue.

Job Numbers

Its job number identifies a queued job. When you submit a job, its job number is automatically generated from the sequence SYS.JOBSEQ.

Once a job is assigned a job number, that number does not change. Even if the job is exported and imported, its job number remains the same.

Job Definitions

The *job definition* is the PL/SQL code specified in the WHAT parameter of the SUBMIT procedure.

Normally the job definition is a single call to a procedure. The procedure call can have any number of parameters.
Note: In the job definition, use two single quotation marks around strings. Always include a semicolon at the end of the job definition.

Parameter	Mode	Description
Job	IN	The number of the current job.
next_date	IN/OUT	The date of the next execution of the job. The default value is SYSDATE.
broken	IN/OUT	Status of job, broken or not broken. The IN value is FALSE.

Table 3: Special Parameter Values for Job Definitions

The following are examples of valid job definitions:

```
'myproc(''10-JAN-82'', next_date, broken);'
'scott.emppackage.give_raise(''JFEE'', 3000.00);'
'dbms_job.remove(job);'
```

Database Links and Jobs

If you submit a job that uses a database link, the link must include a username and password. Anonymous database links will not succeed.

How Jobs Execute

SNP background processes execute jobs. To execute a job, the process creates a session to run the job.
When an SNP process runs a job, the job is run in the same environment in which it was submitted and with the owner's default privileges.

When you force a job to run using the procedure DBMS_JOB.RUN, the job is run by your user process. When your user process runs a job, it is run with your default privileges only. Privileges granted to you through roles are unavailable.

Job Queue Locks

Oracle uses job queue locks to ensure that a job is executed one session at a time. When a job is being run, its session acquires a job queue (JQ) lock for that job.

The following query lists the session identifier, lock type, and lock identifiers for all sessions holding JQ locks:

```
SVRMGR> SELECT sid, type, id1, id2
     2> FROM v$lock
     3> WHERE type = 'JQ';
SID        TY ID1        ID2
---------- -- ---------- ----------
        12 JQ          0      14144
1 row selected.
```

In the query above, the identifier for the session holding the lock is 12. The ID1 lock identifier is always 0 for JQ locks. The ID2 lock identifier is the job number of the job the session is running.

Job Execution Errors

When a job fails, information about the failure is recorded in a trace file and the alert log. Oracle writes message number ORA-12012 and includes the job number of the failed job.

The following can prevent the successful execution of queued jobs:

- not having any SNP background processes to run the job
- a network or instance failure
- an exception when executing the job

Thus, if you can correct the problem that is preventing a job from running before the job has failed sixteen times, Oracle will eventually run that job again.

Removing a Job From the Job Queue

To remove a job from the job queue, use the REMOVE procedure in the DBMS_JOB package:

```
DBMS_JOB.REMOVE(job IN BINARY_INTEGER)
```

The following statement removes job number 14144 from the job queue:

```
DBMS_JOB.REMOVE(14144);
```

Altering a Job

To alter a job that has been submitted to the job queue, use the procedures CHANGE, WHAT, NEXT_DATE, or INTERVAL in the DBMS_JOB package.

Here's an example where the job identified as 14144 is now executed every three days:

```
DBMS_JOB.CHANGE(14144, null, null, 'SYSDATE + 3');
```

Syntax for CHANGE

You can alter any of the user-definable parameters associated with a job by calling the DBMS_JOB.CHANGE procedure. Table 3 describes the procedure's parameters.

```
DBMS_JOB.CHANGE( job                    IN  BINARY_INTEGER,
          what                          IN  VARCHAR2,
          next_date                     IN  DATE,
          interval                     IN  VARCHAR2)
```

If you specify NULL for WHAT, NEXT_DATE, or INTERVAL when you call the procedure CHANGE, the current value remains unchanged.

Note: When you change a job's definition using the WHAT parameter in the procedure CHANGE, Oracle records your current environment. This becomes the new environment for the job.

Syntax for WHAT

You can alter the definition of a job by calling the DBMS_JOB.WHAT procedure.

```
DBMS_JOB.WHAT( job                      IN  BINARY_INTEGER,
         what                          IN  VARCHAR2)
```

Note: When you execute procedure WHAT, Oracle records your current environment. This becomes the new environment for the job.

Syntax for NEXT_DATE

You can alter the next date that Oracle executes a job by calling the DBMS_JOB.NEXT_DATE procedure.

```
DBMS_JOB.NEXT_DATE( job                 IN  BINARY_INTEGER,
         next_date                     IN  DATE)
```

Syntax for INTERVAL

You can alter the execution interval of a job by calling the DBMS_JOB.INTERVAL procedure.

```
DBMS_JOB.INTERVAL( job                  IN BINARY_INTEGER,
         interval                      IN VARCHAR2)
```

Broken Jobs

A job is labeled as either broken or not broken. Oracle does not attempt to run broken jobs. However, you can force a broken job to run by calling the procedure DBMS_JOB.RUN. When you submit a job it is considered not broken.

There are two ways a job can break:

- Oracle has failed to successfully execute the job after sixteen attempts.
- You have marked the job as broken, using the procedure DBMS_JOB.BROKEN.

To mark a job as broken or not broken, use the procedure BROKEN in the DBMS_JOB package.

```
DBMS_JOB.BROKEN( job              IN BINARY_INTEGER,
         broken                   IN BOOLEAN,
         next_date                IN DATE DEFAULT SYSDATE)
```

The following example marks job 14144 as not broken and sets its next execution date to the following Monday:

```
DBMS_JOB.BROKEN(14144, FALSE, NEXT_DAY(SYSDATE, 'MONDAY'));
```

Once a job has been marked as broken, Oracle will not attempt to execute the job until you either mark the job as not broken, or force the job to be executed by calling the procedure DBMS_JOB.RUN.

Running Broken Jobs

If a problem has caused a job to fail sixteen times, Oracle marks the job as broken. Once you have fixed this problem, you can run the job by either:

- forcing the job to run by calling DBMS_JOB.RUN
- marking the job as not broken by calling DBMS_JOB.BROKEN and waiting for Oracle to execute the job

If you force the job to run by calling the procedure DBMS_JOB.RUN, Oracle runs the job immediately. If the job succeeds, then Oracle labels the job as not broken and resets its count of the number of failed executions for the job. Once you reset a job's broken flag (by calling either RUN or BROKEN), job execution resumes according to the scheduled execution intervals set for the job.

Forcing a Job to Execute

There may be times when you would like to manually execute a job. For example, if you have fixed a broken job, you may want to test the job immediately by forcing it to execute. To force a job to be executed immediately, use the procedure RUN in the DBMS_JOB package. Oracle attempts to run the job, even if the job is marked as broken:

```
DBMS_JOB.RUN( job          IN BINARY_INTEGER)
```

When you run a job using DBMS_JOB.RUN, Oracle recomputes the next execution date. For example, if you create a job on a Monday with a NEXT_DATE value of 'SYSDATE' and an INTERVAL value of 'SYSDATE + 7', the job is run every 7 days starting on Monday. However, if you execute RUN on Wednesday, the next execution date will be the next Wednesday.

11.4 Terminating a Job

You can terminate a running job by marking the job as broken, identifying the session running the job, and disconnecting that session. You should mark the job as broken, so that Oracle does not attempt to run the job again.

After you have identified the session running the job (via V$SESSION), you can disconnect the session using the Enterprise Manager Disconnect Session menu item, or the SQL command ALTER SYSTEM.

11.5 Viewing Job Queue Information

You can view information about jobs in the job queue via the data dictionary views in Table 4:

View	Description
DBA_JOBS	Lists all the jobs in the database.
USER_JOBS	Lists all jobs owned by the user.
DBA_JOBS_RUNNING	Lists all jobs in the database that are currently running. This view joins V$LOCK and JOB$.

Table 4: Views for Viewing Job Queue Information

For example, you can display information about a job's status and failed executions. The following sample query creates a listing of the job number, next execution time, failures, and broken status for each job you have submitted:

```
SVRMGR> SELECT job, next_date, next_sec, failures, broken
     2> FROM user_jobs;
JOB        NEXT_DATE NEXT_SEC FAILURES   B
---------- --------- -------- ---------- -
      9125 01-NOV-94 00:00:00          4 N

     14144 24-OCT-94 16:35:35          0 N

     41762 01-JAN-00 00:00:00         16 Y

3 rows selected.
```

You can also display information about jobs currently running. The following sample query lists the session identifier, job number, user who submitted the job, and the start times for all currently running jobs:

```
SVRMGR> SELECT sid, r.job, log_user, r.this_date, r.this_sec
     2> FROM dba_jobs_running r, dba_jobs j
     3> WHERE r.job = j.job;
SID        JOB        LOG_USER             THIS_DATE THIS_SEC
---------- ---------- -------------------- --------- --------
        12      14144 JFEE                 24-OCT-94 17:21:24

        25       8536 SCOTT                24-OCT-94 16:45:12

1 rows selected.
```

11.6 Short Summary

SNP background processes execute job queues. SNP processes periodically wake up and execute any queued jobs that are due to be run. You must have at least one SNP process running to execute your queued jobs in the background.

The *job definition* is the PL/SQL code specified in the WHAT parameter of the SUBMIT procedure.

11.7 Brain Storm

1. If you are not able to drop an online redo log group, then what is its status?
2. What are the main determinants of a Control Files size?

Lecture 12

Managing Table Spaces

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about guidelines for managing Tablespaces
- Discuss about the creation of Tablespaces
- Managing Tablespace Allocation
- Making a Tablespace read-only and viewing information about Tablespaces

Coverage Plan

Lecture 12

12.1 Snap Shot

12.2 Guidelines For Managing Tablespace

12.3 Creating Tablespaces

12.4 Managing Tablespace Allocation

12.5 Making Tablespace Read-Only

12.6 Dropping Tablespaces

12.7 Viewing Information About Tablespaces

12.8 Short Summary

12.9 Brain Storm

12.1 Snap Shot

This chapter describes the various aspects of tablespace management, and includes the following topics:

- Guidelines for Managing Tablespaces
- Creating Tablespaces
- Managing Tablespace Allocation
- Making a Tablespace Read-Only
- Dropping Tablespaces
- Viewing Information About Tablespaces

12.2 Guidelines for Managing Tablespaces

Before working with tablespaces of an Oracle database, consider the guidelines in the following sections:

- Using Multiple Tablespaces
- Specifying Tablespace Storage Parameters
- Assigning Tablespace Quotas to Users

Using Multiple Tablespaces

Using multiple tablespaces allows you more flexibility in performing database operations. For example, when a database has multiple tablespaces, you can perform the following tasks:

- Separate user data from data dictionary data.
- Separate one application's data from another's.
- Store different tablespaces' datafiles on separate disk drives to reduce I/O contention.
- Separate rollback segment data from user data, preventing a single disk failure from causing permanent loss of data.
- Take individual tablespaces offline while others remain online.
- Reserve a tablespace for a particular type of database use, such as high update activity, read-only activity, or temporary segment storage.
- Back up individual tablespaces.

Some operating systems set a limit on the number of files that can be simultaneously open; these limits can affect the number of tablespaces that can be simultaneously online. To avoid exceeding your operating system's limit, plan your tablespaces efficiently. Create only enough tablespaces to fill your needs, and create these tablespaces with as few files as possible. If you need to increase the size of a tablespace, add one or two large datafiles, or create datafiles with the autoextend option set on, rather than many small datafiles.
Review your data in light of these advantages and decide how many tablespaces you will need for your database design.

Specifying Tablespace Storage Parameters

When you create a new tablespace, you can specify default storage parameters for objects that will be created in the tablespace. Storage parameters specified when an object is created override the default storage parameters of the tablespace containing the object. However, if you do not specify storage parameters when creating an object, the object's segment automatically uses the default storage parameters for the tablespace.

Set the default storage parameters for a tablespace to account for the size of a typical object that the tablespace will contain (you estimate this size). You can specify different storage parameters for an unusual or exceptional object when creating that object.

Note: If you do not specify the default storage parameters for a new tablespace, the default storage parameters of Oracle become the tablespace's default storage parameters.

Assigning Tablespace Quotas to Users

Grant users who will be creating tables, clusters, snapshots, indexes, and other objects the privilege to create the object and a *quota* (space allowance or limit) in the tablespace intended to hold the object's segment. The security administrator is responsible for granting the required privileges to create objects to database users and for assigning tablespace quotas, as necessary, to database users.

12.3 Creating Tablespaces

The steps for creating tablespaces vary by operating system. On most operating systems you indicate the size and fully specified filenames when creating a new tablespace or altering a tablespace by adding datafiles. In each situation Oracle automatically allocates and formats the datafiles as specified. However, on some operating systems, you must create the datafiles before installation.

The first tablespace in any database is always the SYSTEM tablespace. Therefore, the first datafiles of any database are automatically allocated for the SYSTEM tablespace during database creation.

You might create a new tablespace for any of the following reasons:

- You want to allocate more disk storage space for the associated database, thereby enlarging the database.
- You need to create a logical storage structure in which to store a specific type of data separate from other database data.

To increase the total size of the database you can alternatively add a datafile to an existing tablespace, rather than adding a new tablespace.

Note: No data can be inserted into any tablespace until the current instance has acquired at least two rollback segments (including the SYSTEM rollback segment).

To create a new tablespace, use either the Create Tablespace property sheet of Enterprise Manager/GUI, or the SQL command CREATE TABLESPACE. You must have the CREATE TABLESPACE system privilege to create a tablespace.

As an example, let's create the tablespace RB_SEGS (to hold rollback segments for the database), with the following characteristics:

- The data of the new tablespace is contained in a single datafile, 50M in size.
- The default storage parameters for any segments created in this tablespace are explicitly set.
- After the tablespace is created, it is left offline.

The following statement creates the tablespace RB_SEGS:

```
CREATE TABLESPACE rb_segs
```

```
DATAFILE 'datafilers_1' SIZE 50M
DEFAULT STORAGE (
  INITIAL 50K
  NEXT 50K
  MINEXTENTS 2
  MAXEXTENTS 50
  PCTINCREASE 0)
OFFLINE;
```

If you do not fully specify filenames when creating tablespaces, the corresponding datafiles are created in the current directory of the database server.

Creating a Temporary Tablespace

If you wish to improve the concurrency of multiple sort operations, reduce their overhead, or avoid Oracle space management operations altogether, you can create *temporary tablespaces*.

Within a temporary tablespace, all sort operations for a given instance and tablespace share a single *sort segment*. Sort segments exist in every instance that performs sort operations within a given tablespace. You cannot store permanent objects in a temporary tablespace. You can view the allocation and deallocation of space in a temporary tablespace sort segment via the V$SORT_SEGMENTS table.

To identify a tablespace as temporary during tablespace creation, issue the following statement:

CREATE TABLESPACE tablespace TEMPORARY

To identify a tablespace as temporary in an existing tablespace, issue the following statement:

ALTER TABLESPACE tablespace TEMPORARY

Note: You can take temporary tablespaces offline. Returning temporary tablespaces online does not affect their temporary status.

12.4 Managing Tablespace Allocation

This section describes aspects of managing tablespace allocation, and includes the following topics:

- Altering Storage Settings for Tablespaces
- Coalescing Free Space

Altering Storage Settings for Tablespaces

You can change the default storage parameters of a tablespace to change the default specifications for *future* objects created in the tablespace. To change the default storage parameters for objects subsequently created in the tablespace, use either the Alter Tablespace property sheet of Enterprise Manager/GUI, or the SQL command ALTER TABLESPACE. Also, to alter the default storage parameters of a tablespace, you must have the ALTER TABLESPACE system privilege.

```
ALTER TABLESPACE users
   DEFAULT STORAGE (
     INITIAL 50K
     NEXT 50K
     MINEXTENTS 2
     MAXEXTENTS 20
```

```
PCTINCREASE 50);
```

New values for the default storage parameters of a tablespace affect only future extents allocated for the segments within the tablespace.

Coalescing Free Space

Space for tablespace segments is managed using extents, which are comprised of a specific number of contiguous data blocks. The free extent closest in size to the required extent is used when allocating new extents to a tablespace segment. Thus, a larger free extent can be fragmented, or smaller contiguous free extents can be coalesced into one larger free extent (see Figure 12.1). However, continuous allocation and deallocation of free space fragments your tablespace and makes allocation of larger extents more difficult. By default, SMON (system monitor) processes incrementally coalesce the free extents of tablespaces in the background. If desired, you can disable SMON coalescing.

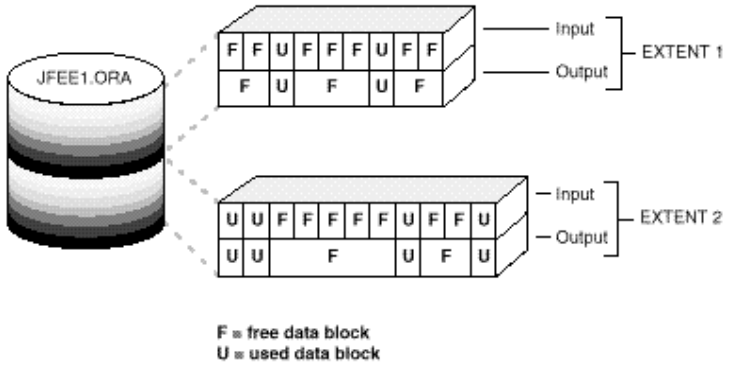


Figure 12.1: Coalescing Free Space

If you find that fragmentation of space is high (contiguous space on your disk appears as non-contiguous), you can coalesce your free space in a single space transaction. After every eight coalesces the space transaction commits and other transactions can allocate or deallocate space. You must have ALTER TABLESPACE privileges to coalesce tablespaces. You can coalesce all available free space extents in a tablespace into larger contiguous extents on a per tablespace basis by using the following command:

ALTER TABLESPACE tablespace COALESCE;

You can also use this command to supplement SMON and extent allocation coalescing, thereby improving space allocation performance in severely fragmented tablespaces. Issuing this command does not effect the performance of other users accessing the same tablespace. Like other options of the ALTER TABLESPACE command, the COALESCE option is exclusive; when specified, it should be the only option.

Viewing Information about Tablespaces

To display statistics about coalesceable extents for tablespaces, you can view the DBA_FREE_SPACE_COALESCED view. You can query this view to determine if you need to coalesce space in a particular tablespace.

12.5 Making a Tablespace Read-Only

This section describes issues related to making tablespaces read-only, and includes the following topics:

- Prerequisites

- Making a Read-Only Tablespace Writeable
- Creating a Read-Only Tablespace on a WORM Device

Making a tablespace read-only prevents further write operations on the datafiles in the tablespace. After making the tablespace read-only, you should back it up.

Use the SQL command ALTER TABLESPACE to change a tablespace to read-only. You must have the ALTER TABLESPACE system privilege to make a tablespace read-only. The following statement makes the FLIGHTS tablespace read-only:

ALTER TABLESPACE flights READ ONLY

After a tablespace is read-only, you can copy its files to read-only media. You must then rename the datafiles in the control file to point to the new location by using the SQL command ALTER DATABASE RENAME.

A read-only tablespace is neither online nor offline. Issuing the ALTER TABLESPACE command with the ONLINE or OFFLINE option does not change the read-only state of the tablespace; rather, it causes all of the datafiles in the tablespace to be brought online or offline.

Prerequisites

Before you can make a tablespace read-only, the following conditions must be met. It may be easiest to meet these restrictions by performing this function in restricted mode, so that only users with the RESTRICTED SESSION system privilege can be logged on.

- The tablespace must be online.
- There must not be any active transactions in the entire database.

This is necessary to ensure that there is no undo information that needs to be applied to the tablespace.

- The tablespace must not contain any active rollback segments.

For this reason, the SYSTEM tablespace can never be made read-only, since it contains the SYSTEM rollback segment. Additionally, because the rollback segments of a read-only tablespace are not accessible, it is recommended that you drop the rollback segments before you make a tablespace read-only.

- The tablespace must not currently be involved in an online backup, since the end of a backup updates the header file of all datafiles in the tablespace.
- The COMPATIBLE initialization parameter must be set to 7.1.0 or greater.

For better performance while accessing data in a read-only tablespace, you might want to issue a query that accesses all of the blocks of the tables in the tablespace just before making it read-only. A simple query, such as SELECT COUNT (*), executed against each table will ensure that the data blocks in the tablespace can be subsequently accessed most efficiently. This eliminates the need for Oracle to check the status of the transactions that most recently modified the blocks.
Warning: You cannot rename or resize datafiles belonging to a read-only tablespace.

Making a Read-Only Tablespace Writeable

Whenever you create a tablespace, it is both readable and writeable. To change a read-only tablespace back to a read-write tablespace, use the SQL command ALTER TABLESPACE. You must have the ALTER TABLESPACE system privilege to change a read-only tablespace to a read-write tablespace. The following command makes the FLIGHTS tablespace writeable:

ALTER TABLESPACE flights READ WRITE;

Making a read-only tablespace writeable updates the control file for the datafiles, so that you can use the read-only version of the datafiles as a starting point for recovery.

Prerequisites

To issue this command, all of the datafiles in the tablespace must be online. Use the DATAFILE ONLINE option of the ALTER DATABASE command to bring a datafile online. The V$DATAFILE view lists the current status of a datafile.

Creating a Read-Only Tablespace on a WORM Device

You may wish to create a read-only tablespace on a WORM (Write Once Read Many) device when you have read-only files that do not require updating.

1. Create a writeable tablespace on another device. Create the objects that belong in the tablespace and insert your data.
2. Issue the ALTER TABLESPACE command with the READ ONLY option to change the tablespace to read-only.
3. Copy the datafiles of the tablespace onto the WORM device. Use operating system commands to copy the files.
4. Take the tablespace offline.
5. Rename the datafiles to coincide with the names of the datafiles you copied onto your WORM device. Renaming the datafiles changes their names in the control file.
6. Bring the tablespace online.

12.6 Dropping Tablespaces

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. Any tablespace in an Oracle database, except the SYSTEM tablespace, can be dropped. You must have the DROP TABLESPACE system privilege to drop a tablespace.

When you drop a tablespace, only the file pointers in the control files of the associated database are dropped. The datafiles that constituted the dropped tablespace continue to exist. To free previously used disk space, delete the datafiles of the dropped tablespace using the appropriate commands of your operating system after completing this procedure.

You cannot drop a tablespace that contains any active segments. For example, if a table in the tablespace is currently being used or the tablespace contains an active rollback segment, you cannot drop the tablespace. For simplicity, take the tablespace offline before dropping it.

After a tablespace is dropped, the tablespace's entry remains in the data dictionary (see the DBA_TABLESPACES view), but the tablespace's status is changed to INVALID.

To drop a tablespace, use either the Drop tablespace menu item of Enterprise Manager/GUI, or the SQL command DROP TABLESPACE. The following statement drops the USERS tablespace, including the segments in the tablespace:

DROP TABLESPACE users INCLUDING CONTENTS;

If the tablespace is empty (does not contain any tables, views, or other structures), you do not need to check the Including Contained Objects checkbox. If the tablespace contains any tables with primary or unique keys referenced by foreign keys of tables in other tablespaces and you want to cascade the drop of the FOREIGN KEY constraints of the child tables, select the Cascade Drop of Integrity Constraints checkbox to drop the tablespace.

Use the CASCADE CONSTRAINTS option to cascade the drop of the FOREIGN KEY constraints in the child tables.

12.7 Viewing Information About Tablespaces

The following data dictionary views provide useful information about tablespaces of a database:

- USER_EXTENTS, DBA_EXTENTS
- USER_SEGMENTS, DBA_SEGMENTS
- USER_FREE_SPACE, DBA_FREE_SPACE
- DBA_USERS
- DBA_TS_QUOTAS
- USER_TABLESPACES, DBA_TABLESPACES
- DBA_DATA_FILES
- V$DATAFILE

The following examples illustrate how to use the views not already illustrated in other chapters of this manual. They assume you are using a database that contains two tablespaces, SYSTEM and USERS. USERS is made up of two files, FILE1 (100MB) and FILE2 (200MB); the tablespace has been taken offline normally.
Listing Tablespaces and Default Storage Parameters: Example

To list the names and default storage parameters of all tablespaces in a database, use the following query on the DBA_TABLESPACES view:

```
SELECT tablespace_name "TABLESPACE",
   initial_extent "INITIAL_EXT",
   next_extent "NEXT_EXT",
   min_extents "MIN_EXT",
   max_extents "MAX_EXT",
   pct_increase
   FROM sys.dba_tablespaces;

TABLESPACE  INITIAL_EXT  NEXT_EXT  MIN_EXT   MAX_EXT    PCT_INCREASE
----------  -----------  --------  -------   -------    ------------
SYSTEM      10240000     10240000        1        99            50
USERS       10240000     10240000        1        99            50
```

Listing the Datafiles and Associated Tablespaces of a Database: Example

To list the names, sizes, and associated tablespaces of a database, enter the following query on the DBA_DATA_FILES view:

```
SELECT  file_name, bytes, tablespace_name
   FROM sys.dba_data_files;
```

```
FILE_NAME    BYTES      TABLESPACE_NAME
------------ ---------- ------------------
filename1    10240000   SYSTEM
filename2    10240000   USERS
filename3    20480000   USERS
```

Listing the Free Space (Extents) of Each Tablespace: Example

To see the amount of space available in the free extents of each tablespace in the database, enter the following query:

```
SELECT tablespace_name, file_id,
   COUNT(*)    "PIECES",
   MAX(blocks) "MAXIMUM",
   MIN(blocks) "MINIMUM",
   AVG(blocks) "AVERAGE",
   SUM(blocks) "TOTAL"
   FROM sys.dba_free_space
WHERE tablespace_name = 'SYSTEM'
GROUP BY tablespace_name, file_id;


TABLESPACE    FILE_ID  PIECES   MAXIMUM    MINIMUM  AVERAGE     SUM
----------    -------  ------   -------    -------  -------   ------
SYSTEM              1       2      2928        115   1521.5     3043
```

SUM shows the amount of free space in each tablespace, PIECES shows the amount of fragmentation in the datafiles of the tablespace, and MAXIMUM shows the largest contiguous area of space. This query is useful when you are going to create a new object or you know that a segment is about to extend, and you want to make sure that there is enough space in the containing tablespace.

12.8 Short Summary

Create a new tablespace for any of the following reasons:

- You want to allocate more disk storage space for the associated database, thereby enlarging the database.
- You need to create a logical storage structure in which to store a specific type of data separate from other database data.

12.9 Brain Storm

1. When an Oracle user process ends abnormally, it is called as?
   a) Statement Failure b) Instance Failure c) Process Failure d)Media Failure
2. The Three types of complete media recovery are
   a) close database recovery
   b) open database, offline tablespace recovery
   c) open database, offline tablespace recovery
   d) None of the above
3. What is the DBA's most important responsibility?
   - Keeping the database organized
   - Keeping upto date backups
   - Keeping the database available to users
   - Keeping users from corrupting the database

4. What is the responsibility of DBA when media failure occurs
   a) Restoring the File system
   b) Restoring the operating System
   c) Restoring the database
   d) Recreating the Oracle account
5. Point-in-time recovery refers to
   a) Point-in-time the backup was taken
   b) Point-in-time the database failed
   c) Point-in-time when recovery is performed
   d) All of the above
6. Process failure is recoverd automatically
   a) RECO b) PMON c) SMON d) None of the Above

Lecture 13

Managing Data Files

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about guidelines for managing Datafiles
- Discuss about the creation and addition of Datafiles to a Tablespaces
- Discuss about renaming and relocating a Datafiles
- Viewing information about Datafiles and verifying data blocks in Datafiles

Coverage Plan

Lecture 13

13.1 Snap Shot

13.2 Guidelines For Managing Datafiles

13.3 Creating and Adding Datafiles to a Tablespace

13.4 Changing a Datafile's Size

13.5 Renaming and Relocating Datafiles

13.6 Verifying Data Blocks in Datafiles

13.7 Viewing Information about Datafiles

13.8 Short Summary

13.9 Brain Storm

13.1 Snap Shot

This chapter describes the various aspects of datafile management, and includes the following topics:

- Guidelines for Managing Datafiles
- Creating and Adding Datafiles to a Tablespace
- Changing a Datafile's Size
- Renaming and Relocating Datafiles
- Verifying Data Blocks in Datafiles
- Viewing Information About Datafiles

Datafiles can also be created as part of database recovery from a media failure.

13.2 Guidelines for Managing Datafiles

This section describes aspects of managing datafiles, and includes the following topics:

- Number of Datafiles
- Set the Size of Datafiles
- Place Datafiles Appropriately
- Store Datafiles Separately From Redo Log Files

Every datafile has two associated file numbers: an *absolute file number* and a *relative file number*.

An absolute file number uniquely identifies a datafile in the database. Prior to Oracle8, the absolute file number was referred to as simply the "file number."

A relative file number uniquely identifies a datafile within a tablespace. For small and medium size databases, relative file numbers usually have the same value as the absolute file number. However, when the number of datafiles in a database exceeds a threshold (typically 1023), the relative file number will differ from the absolute file number. You can locate relative file numbers in many of the data dictionary views.

Number of Datafiles

At least one datafile is required for the SYSTEM tablespace of a database; a small system might have a single datafile. In general, keeping a few large datafiles is preferable to many small datafiles, because you can keep fewer files open at the same time.

You can add datafiles to tablespaces, subject to the following operating system-specific datafile limits:

Operating system limit	Each operating system sets a limit on the maximum number of open files per process. Regardless of all other limits, more datafiles cannot be created when the operating system limit of open files is reached.
Oracle system limit	Oracle imposes a maximum limit on the number of datafiles for any Oracle database opened by any instance. This limit is port-specific.
Control file upper bound	When you issue CREATE DATABASE or CREATE CONTROLFILE statements, the MAXDATAFILES parameter specifies an initial size of the datafile portion of the control file. Later, if you add a file whose number exceeds MAXDATAFILES but is less than or equal to DB_FILES, the control file automatically expands to allow the datafile portion to accommodate more files.

instance or SGA upper bound	When starting an Oracle8 instance, the database's parameter file indicates the amount of SGA space to reserve for datafile information; the maximum number of datafiles is controlled by the DB_FILES parameter. This limit applies only for the life of the instance. **Note:** The default value of DB_FILES is operating system specific. With the Oracle Parallel Server, all instances must set the instance datafile upper bound to the same value.

When determining a value for DB_FILES, take the following into consideration:

- If the value of DB_FILES is too low, you will be unable to add datafiles beyond the DB_FILES limit without first shutting down the database.

- IF the value of DB_FILES is too high, memory is unnecessarily consumed.

Theoretically, an Oracle database can have an unlimited number of datafiles. Nevertheless, you should consider the following when determining the number of datafiles:

- Performance is better with a small number of datafiles rather than a large number of small datafiles. Large files also increase the granularity of a recoverable unit.

- Operating systems often impose a limit on the number of files a process can open simultaneously. Oracle's DBWR process can open all online datafiles. Oracle is also capable of treating open file descriptors as a cache, automatically closing files when the number of open file descriptors reaches the operating system-defined limit.

Oracle allows more datafiles in the database than the operating system-defined limit; this can have a negative performance impact. When possible, adjust the operating system limit on open file descriptors so that it is larger than the number of online datafiles in the database.

The operating system specific limit on the maximum number of datafiles allowed in a tablespace is typically 1023 files.

Set the Size of Datafiles

The first datafile (in the original SYSTEM tablespace) must be at least 7M to contain the initial data dictionary and rollback segment. If you install other Oracle products, they may require additional space in the SYSTEM tablespace (for online help, for example); see the installation instructions for these products.

Place Datafiles Appropriately

Tablespace location is determined by the physical location of the datafiles that constitute that tablespace. Use the hardware resources of your computer appropriately.

For example, if several disk drives are available to store the database, it might be helpful to store table data in a tablespace on one disk drive, and index data in a tablespace on another disk drive. This way, when users query table information, both disk drives can work simultaneously, retrieving table and index data at the same time.

Store Datafiles Separately From Redo Log Files

Datafiles should not be stored on the same disk drive that stores the database's redo log files. If the datafiles and redo log files are stored on the same disk drive and that disk drive fails, the files cannot be used in your database recovery procedures.

If you multiplex your redo log files, then the likelihood of your losing all of your redo log files is low, so you can store datafiles on the same drive as some redo log files.

13.3 Creating and Adding Datafiles to a Tablespace

You can create and add datafiles to a tablespace to increase the total amount of disk space allocated for the tablespace, and consequently the database.

Ideally, when creating a tablespace DBAs should estimate the potential size of the database objects and add sufficient files or devices. Doing so ensures that data is spread evenly across all devices.

To add datafiles to a tablespace, use either the Add Datafile dialog box of Enterprise Manager/GUI, or the SQL command ALTER TABLESPACE. You must have the ALTER TABLESPACE system privilege to add datafiles to a tablespace.

The following statement creates a new datafile for the RB_SEGS tablespace:

```
ALTER TABLESPACE rb_segs

   ADD DATAFILE 'filename1' SIZE 1M;
```

If you add new datafiles to a tablespace and do not fully specify the filenames, Oracle creates the datafiles in the default directory of the database server. Unless you want to reuse existing files, make sure the new filenames do not conflict with other files; the old files that have been previously dropped will be overwritten.

13.4 Changing a Datafile's Size

This section describes the various ways to alter the size of a datafile, and includes the following topics:

- Enabling and Disabling Automatic Extension for a Datafile
- Manually Resizing a Datafile

Enabling and Disabling Automatic Extension for a Datafile

You can create datafiles or alter existing datafiles so that they automatically increase in size when more space is needed in the database. The files increase in specified increments up to a specified maximum.

Setting your datafiles to extend automatically results in the following:

- reduces the need for immediate intervention when a tablespace runs out of space
- ensures applications will not halt because of failures to allocate extents

To find out if a datafile is auto-extensible, query the DBA_DATA_FILES view and examine the AUTOEXTENSIBLE column.

You can specify automatic file extension when you create datafiles via the following SQL commands:

- `CREATE DATABASE`
- `CREATE TABLESPACE`
- `ALTER TABLESPACE`

You can enable or disable automatic file extension for existing datafiles, or manually resize a datafile using the SQL command ALTER DATABASE.

The following example enables automatic extension for a datafile, FILENAME2, added to the USERS tablespace:

```
ALTER TABLESPACE users
  ADD DATAFILE 'filename2' SIZE 10M
 AUTOEXTEND ON
 NEXT 512K
MAXSIZE 250M
```

The value of NEXT is the minimum size of the increments added to the file when it extends. The value of MAXSIZE is the maximum size to which the file can automatically extend.

The next example disables automatic extension for the datafile FILENAME2:

```
ALTER DATABASE DATAFILE 'filename2'

    AUTOEXTEND OFF
```

Manually Resizing a Datafile

You can manually increase or decrease the size of a datafile using the ALTER DATABASE command.

Because you can change the sizes of datafiles, you can add more space to your database without adding more datafiles. This is beneficial if you are concerned about reaching the maximum number of datafiles allowed in your database.

Manually reducing the sizes of datafiles allows you to reclaim unused space in the database. This is useful for correcting errors in estimates of space requirements.

In this example, assume that the datafile FILENAME2 has extended up to 250M. However, because its tablespace now stores smaller objects, the datafile can be reduced in size.

The following command decreases the size of datafile FILENAME2:

```
ALTER DATABASE DATAFILE 'filename2'
   RESIZE 100M
```

Note: It is not always possible to decrease the size of a file to a specific value.

13.5 Renaming and Relocating Datafiles

This section describes the various aspects of renaming and relocating datafiles, and includes the following topics:

- Renaming and Relocating Datafiles for a Single Tablespace
- Renaming and Relocating Datafiles for Multiple Tablespaces

You can rename datafiles to change either their names or locations. Oracle provides options to make the following changes:

- Rename and relocate datafiles in a single offline tablespace (for example, FILENAME1 and FILENAME2 in TBSPACE1) while the rest of the database is open.

- Rename and relocate datafiles in several tablespaces simultaneously (for example, FILE1 in TBSP1 and FILE2 in TBSP2) while the database is mounted but closed.

Note: To rename or relocate datafiles of the SYSTEM tablespace, you must use the second option, because you cannot take the SYSTEM tablespace offline.

Renaming and relocating datafiles with these procedures only change the pointers to the datafiles, as recorded in the database's control file; it does not physically rename any operating system files, nor does it copy files at the operating system level. Therefore, renaming and relocating datafiles involve several steps. Read the steps and examples carefully before performing these procedures.

You must have the ALTER TABLESPACE system privilege to rename datafiles of a single tablespace.

Renaming and Relocating Datafiles for a Single Tablespace

The following steps describe how to rename or relocate datafiles from a single tablespace.

1. Take the non-SYSTEM tablespace that contains the datafiles offline.
2. Copy the datafiles to the new location or new names using the operating system.
3. Make sure that the new, fully specified filenames are different from the old filenames.
4. Use either the Rename Datafile dialog box of Enterprise Manager/GUI or the SQL command ALTER TABLESPACE with the RENAME DATAFILE option to change the filenames within the database.

For example, the following statement renames the datafiles FILENAME1 and FILENAME2 to FILENAME3 and FILENAME4, respectively:

```
ALTER TABLESPACE users
  RENAME DATAFILE 'filename1', 'filename2'
      TO 'filename3', 'filename4';
```

The new file must already exist; this command does not create a file. Also, always provide complete filenames (including their paths) to properly identify the old and new datafiles. In particular, specify the old filename exactly as it appears in the DBA_DATA_FILES view of the data dictionary.

Renaming and Relocating Datafiles for Multiple Tablespaces

You can rename and relocate datafiles of one or more tablespaces using the SQL command ALTER DATABASE with the RENAME FILE option. This option is the only choice if you want to rename or relocate datafiles of several tablespaces in one operation, or rename or relocate datafiles of the SYSTEM tablespace. If the database must remain open, consider instead the procedure outlined in the previous section.

To rename datafiles of several tablespaces in one operation or to rename datafiles of the SYSTEM tablespace, you must have the ALTER DATABASE system privilege.

1. Ensure that the database is mounted but closed.
2. Copy the datafiles to be renamed to their new locations and new names, using operating system commands.
3. Make sure the new copies of the datafiles have different fully specified filenames from the datafiles currently in use.
4. Use the SQL command ALTER DATABASE to rename the file pointers in the database's control file.

For example, the following statement renames the datafiles FILENAME 1 and FILENAME2 to FILENAME3 and FILENAME4, respectively:

```
ALTER DATABASE
  RENAME FILE 'filename1', 'filename2'
    TO 'filename3', 'filename4';
```

The new file must already exist; this command does not create a file. Also, always provide complete filenames (including their paths) to properly identify the old and new datafiles. In particular, specify the old filename exactly as it appears in the DBA_DATA_FILES view of the data dictionary.

Relocating Datafiles: Example

For this example, assume the following conditions:

- An open database has a tablespace named USERS that is comprised of datafiles located on the same disk of a computer.
- The datafiles of the USERS tablespace are to be relocated to a different disk drive.
- You are currently connected with administrator privileges to the open database while using Enterprise Manager.

1. Identify the Datafile names of interest.

The following query of the data dictionary view DBA_DATA_FILES lists the datafile names and respective sizes (in bytes) of the USERS tablespace:

```
SELECT file_name, bytes FROM sys.dba_data_files

   WHERE tablespace_name = 'USERS';
FILE_NAME         BYTES
---------------------------
FILENAME1         102400000
FILENAME2         102400000
```

Here, FILENAME1 and FILENAME2 are two fully specified filenames, each 1MB in size.

2. Back up the database.

Before making any structural changes to a database, such as renaming and relocating the datafiles of one or more tablespaces, always completely back up the database.

3. Take the tablespace containing the datafile offline, or shut down the database and restart and mount it, leaving it closed. Either option closes the datafiles of the tablespace.

4. Copy the datafiles to their new locations using operating system commands. For this example, the existing files FILENAME1 and FILENAME2 are copied to FILENAME3 and FILENAME4.

5. Rename the datafiles within Oracle.

The datafile pointers for the files that comprise the USERS tablespace, recorded in the control file of the associated database, must now be changed from FILENAME1 and FILENAME2 to FILENAME3 and FILENAME4, respectively.

If the tablespace is offline but the database is open, use the Enterprise Manager Rename Datafiles dialog box or ALTER TABLESPACE...RENAME DATAFILE command. If the database is mounted but closed, use the ALTER DATABASE...RENAME FILE command.

6. **Bring the tablespace online, or shut down and restart the database.**

If the USERS tablespace is offline and the database is open, bring the tablespace back online. If the database is mounted but closed, open the database.

7. **Back up the database. After making any structural changes to a database, always perform an immediate and complete backup.**

13.6 Verifying Data Blocks in Datafiles

If you want to configure Oracle to use checksums to verify data blocks, set the initialization parameter DB_BLOCK_CHECKSUM to TRUE. The default value of DB_BLOCK_CHECKSUM is FALSE.

When you enable block checking, Oracle computes a checksum for each block written to disk. Checksums are computed for all data blocks, including temporary blocks.

The DBWR process calculates the checksum for each block and stores it in the block's header. Checksums are also computed by the direct loader.

The next time Oracle reads a data block, it uses the checksum to detect corruption in the block. If a corruption is detected, Oracle returns message ORA-01578 and writes information about the corruption to a trace file.

13.7 Viewing Information About Datafiles

The following data dictionary views provide useful information about the datafiles of a database:

- `USER_EXTENTS, DBA_EXTENTS`
- `USER_SEGMENTS, DBA_SEGMENTS`
- `USER_FREE_SPACE, DBA_FREE_SPACE`
- `DBA_USERS`
- `DBA_TS_QUOTAS`
- `USER_TABLESPACES, DBA_TABLESPACES`
- `DBA_DATA_FILES`
- `V$DATAFILE`

Datafiles: Example Listing Status Information About

The following example illustrates how to use a view not already illustrated in other chapters of this manual. Assume you are using a database that contains two tablespaces, SYSTEM and USERS. USERS is made up of two files, FILE1 (100MB) and FILE2 (200MB); the tablespace has been taken offline normally. Here, you query V$DATAFILE to view status information about datafiles of a database:

```
SELECT name,

 file#,
    status,
```

```
    checkpoint_change# "CHECKPOINT"   FROM    $datafile;

NAME                                FILE#     STATUS    CHECKPOINT
------------------------------      -----     -------    ----------
filename1                             1       SYSTEM          3839
filename2                             2       OFFLINE         3782
filename3                             3       OFFLINE         3782
```

13.8 Short Summary

Setting your datafiles to extend automatically results in the following:

- Reduces the need for immediate intervention when a tablespace runs out of space
- Ensures applications will not halt because of failures to allocate extents

13.9 Brain Storm

1. Create Tablespace TS1 Datafile 'C:\..\..' size 1m;

   Create Rollback Segment R1 Tablespace TS1;
   Alter Rollback Segment R1 online;
   Create Table T1 (n int) tablespace TS1;
   Insert, Commit, Switch
   Drop Tablespace TS1 including contents
   What happens?
   a) Errors Occur
   b) Statement Processed

2. Which of the following is False

   a) The Grantee can grant/ revoke the system privilege to / from any user in the Database
   b) The grantee of a role can alter or drop the role
   c) The Grantee can further grant the System Privilege with the ADMIN OPTION
   d) The granter cannot drop his own System Privilege role.

3. Which Background Process helps to automatically resolve partly available status

   a) It cannot be reduced. It can be done only by DBA
   b) RECO
   c) PMON and SMON
   d) None of the above

4. When a mix of transactions (Small, Medium and Large) is not prevalent each Rollback Segment should be

   a) 50% of the size of the Database Largest Table
   b) Size of Databases Largest Table (100%)
   c) Only 10% of the size of the Database Largest Table
   d) Any size

Lecture 14

Managing Schema Objects

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about guidelines for managing space in data blocks
- Discuss about the setting storage parameters
- Discuss about PCTFREE and PCTUSED parameters
- Describe about deallocation of space

Coverage Plan

Lecture 14

14.1 Snap Shot

14.2 Managing Space In Data Blocks

14.3 The Pctfree Parameter

14.4 The Pctused Parameter

14.5 Setting Storage Parameters

14.6 Deallocating Space

14.7 Short Summary

14.8 Brain Storm

14.1 Snap Shot

This chapter describes guidelines for managing schema objects, and includes the following topics:

- Managing Space in Data Blocks
- Setting Storage Parameters
- Deallocating Space

14.2 Managing Space in Data Blocks

This section describes the various aspects of managing space in data blocks, and includes the following topics:

- The PCTFREE Parameter
- The PCTUSED Parameter
- Selecting Associated PCTUSED and PCTFREE Values

You can use the PCTFREE and PCTUSED parameters to make the following changes:

- increase the performance of writing and retrieving data
- decrease the amount of unused space in data blocks
- decrease the amount of row chaining between data blocks

14.3 The PCTFREE Parameter

The PCTFREE parameter is used to set the percentage of a block to be reserved for possible updates to rows that already are contained in that block. For example, assume that you specify the following parameter within a CREATE TABLE statement:

PCTFREE 20

This indicates that 20% of each data block used for this table's data segment will be kept free and available for possible updates to the existing rows already within each block.

Specifying PCTFREE

The default for PCTFREE is 10 percent. You can use any integer between 0 and 99, inclusive, as long as the sum of PCTFREE and PCTUSED does not exceed 100.

A smaller PCTFREE has the following effects:

- reserves less room for updates to expand existing table rows
- allows inserts to fill the block more completely
- may save space, because the total data for a table or index is stored in fewer blocks (more rows or entries per block)

A small PCTFREE might be suitable, for example, for a segment that is rarely changed.A larger PCTFREE has the following effects:

- reserves more room for future updates to existing table rows
- may require more blocks for the same amount of inserted data (inserting fewer rows per block)
- may improve update performance, because Oracle does not need to chain row pieces as frequently, if ever

A large PCTFREE is suitable, for example, for segments that are frequently updated.

PCTFREE for Non-Clustered Tables If the data in the rows of a non-clustered table is likely to increase in size over time, reserve some space for these updates. Otherwise, updated rows are likely to be chained among blocks.

PCTFREE for Clustered Tables The discussion for non-clustered tables also applies to clustered tables. However, if PCTFREE is reached, new rows from *any* table contained in the same cluster key go into a new data block that is chained to the existing cluster key.

PCTFREE for Indexes You can specify PCTFREE only when initially creating an index.

14.4 The PCTUSED Parameter

After a data block becomes full as determined by PCTFREE, Oracle does not consider the block for the insertion of new rows until the percentage of the block being used falls below the parameter PCTUSED. Before this value is achieved, Oracle uses the free space of the data block only for updates to rows already contained in the data block. For example, assume that you specify the following parameter within a CREATE TABLE statement:

PCTUSED 40

In this case, a data block used for this table's data segment is not considered for the insertion of any new rows until the amount of used space in the block falls to 39% or less (assuming that the block's used space has previously reached PCTFREE).

Specifying PCTUSED

The default value for PCTUSED is 40 percent. After the free space in a data block reaches PCTFREE, no new rows are inserted in that block until the percentage of space used falls below PCTUSED. The percent value is for the block space available for data after overhead is subtracted from total space.

You can specify any integer between 0 and 99 (inclusive) for PCTUSED, as long as the sum of PCTUSED and PCTFREE does not exceed 100.
A smaller PCTUSED has the following effects:

- reduces processing costs incurred during UPDATE and DELETE statements for moving a block to the free list when it has fallen below that percentage of usage
- increases the unused space in a database

A larger PCTUSED has the following effects:

- improves space efficiency
- increases processing cost during INSERTs and UPDATEs

Selecting Associated PCTUSED and PCTFREE Values

If you decide not to use the default values for PCTFREE or PCTUSED, keep the following guidelines in mind:

- The sum of PCTFREE and PCTUSED must be equal to or less than 100.
- If the sum equals 100, then Oracle attempts to keep no more than PCTFREE free space, and processing costs are highest.
- Block overhead is not included in the computation of PCTUSED or PCTFREE.

- The smaller the difference between 100 and the sum of PCTFREE and PCTUSED (as in PCTUSED of 75, PCTFREE of 20), the more efficient space usage is, at some performance cost.

14.5 Setting Storage Parameters

This section describes the storage parameters you can set for various data structures, and includes the following topics:

You can set storage parameters for the following types of logical storage structures:

- tablespaces (most defaults for any segment in the tablespace)
- tables, clusters, snapshots, and snapshot logs (data segments)
- indexes (index segments)
- rollback segments

Storage Parameters You Can Specify

Every database has default values for storage parameters. You can specify defaults for a tablespace, which override the system defaults to become the defaults for objects created in that tablespace only. Furthermore, you can specify storage settings for each individual object. The storage parameters you can set are:

- **INITIAL**

The size, in bytes, of the first extent allocated when a segment is created.

```
Default: 5 data blocks
Minimum: 2 data blocks (rounded up)
Maximum: operating system specific
```

- **NEXT**

The size, in bytes, of the next incremental extent to be allocated for a segment. The second extent is equal to the original setting for NEXT. From there forward, NEXT is set to the previous size of NEXT multiplied by (1 + PCTINCREASE/100).

```
Default: 5 data blocks
Minimum: 1 data block
Maximum: operating system specific
```

- **MAXEXTENTS**

The total number of extents, including the first, that can ever be allocated for the segment.

```
Default: dependent on the data block size and operating system
Minimum: 1 (extent)
Maximum: unlimited
```

- **MINEXTENTS**

The total number of extents to be allocated when the segment is created. This allows for a large allocation of space at creation time, even if contiguous space is not available.

```
Default: 1 (extent)
Minimum: 1 (extent)
Maximum: unlimited
```

- **PCTINCREASE**

The percent by which each incremental extent grows over the last incremental extent allocated for a segment. If PCTINCREASE is 0, then all incremental extents are the same size. If PCTINCREASE is greater than zero, then each time NEXT is calculated, it grows by PCTINCREASE. PCTINCREASE cannot be negative.

The new NEXT equals 1 + PCTINCREASE/100, multiplied by the size of the last incremental extent (the old NEXT) and rounded *up* to the next multiple of a block size.

```
Default: 50 (%)
Minimum: 0 (%)
Maximum: operating system specific
```

- **INITRANS**

Reserves a pre-allocated amount of space for an initial number of DML transaction entries to access rows in the data block concurrently. Space is reserved in the headers of all data blocks in the associated data or index segment.

The default value is 1 for tables and 2 for clusters and indexes.

- **MAXTRANS**

As multiple transactions concurrently access the rows of the same data block, space is allocated for each DML transaction's entry in the block. Once the space reserved by INITRANS is depleted, space for additional transaction entries is allocated out of the free space in a block, if available. Once allocated, this space effectively becomes a permanent part of the block header. The MAXTRANS parameter limits the number of transaction entries that can concurrently use data in a data block. Therefore, you can limit the amount of free space that can be allocated for transaction entries in a data block using MAXTRANS.

The default value is an operating system-specific function of block size, not exceeding 255.

Setting INITRANS and MAXTRANS

Transaction entry settings for the data blocks allocated for a table, cluster, or index should be set individually for each object based on the following criteria:

- the space you would like to reserve for transaction entries compared to the space you would reserve for database data
- the number of concurrent transactions that are likely to touch the same data blocks at any given time

For example, if a table is very large and only a small number of users simultaneously access the table, the chances of multiple concurrent transactions requiring access to the same data block is low. Therefore, INITRANS can be set low, especially if space is at a premium in the database.

Alternatively, assume that a table is usually accessed by many users at the same time. In this case, you might consider pre-allocating transaction entry space by using a high INITRANS (to eliminate the overhead of having to allocate transaction entry space, as required when the object is in use) and allowing a higher MAXTRANS so that no user has to wait to access any necessary data blocks.

Setting Default Storage Parameters for Segments in a Tablespace

You can set default storage parameters for each tablespace of a database. Any storage parameter that you do not explicitly set when creating or subsequently altering a segment in a tablespace automatically is set to the corresponding default storage parameter for the tablespace in which the segment resides.

With partitioned tables, you can set default storage parameters at the table level. When creating a new partition of the table, the default storage parameters are inherited from the partitioned table; if there is no partitioned table, then they are inherited from the tablespace. When specifying MINEXTENTS at the tablespace level, any extent allocated in the tablespace is rounded to a multiple of the number of minimum extents. Basically, the number of extents is a multiple of the number of blocks.

Setting Storage Parameters for Data Segments

You can set the storage parameters for the data segment of a non-clustered table, snapshot, or snapshot log using the STORAGE clause of the CREATE or ALTER statement for tables, snapshots, or snapshot logs.

In contrast, you set the storage parameters for the data segments of a cluster using the STORAGE clause of the CREATE CLUSTER or ALTER CLUSTER command, rather than the individual CREATE or ALTER commands that put tables and snapshots into the cluster. Storage parameters specified when creating or altering a *clustered* table or snapshot are ignored. The storage parameters set for the cluster override the table's storage parameters.

Setting Storage Parameters for Index Segments

Storage parameters for an index segment created for a table index can be set using the STORAGE clause of the CREATE INDEX or ALTER INDEX command. Storage parameters of an index segment created for the index used to enforce a primary key or unique key constraint can be set in the ENABLE clause of the CREATE TABLE or ALTER TABLE commands or the STORAGE clause of the ALTER INDEX command.

A PCTFREE setting for an index only has an effect when the index is created. You cannot specify PCTUSED for an index segment.

Setting Storage Parameters for LOB Segments

You can set storage parameters for LOB segments using the NOCACHE, NOLOGGING and PCTVERSION LOB storage parameters of the CREATE TABLE command.

Changing Values for Storage Parameters

You can alter default storage parameters for tablespaces and specific storage parameters for individual segments if the current settings are incorrect. All default storage parameters can be reset for a tablespace. However, changes affect only new objects created in the tablespace, or new extents allocated for a segment.

The INITIAL and MINEXTENTS storage parameters cannot be altered for an existing table, cluster, index, or rollback segment. If only NEXT is altered for a segment, the next incremental extent is the size of the new NEXT, and subsequent extents can grow by PCTINCREASE as usual.
If both NEXT and PCTINCREASE are altered for a segment, the next extent is the new value of NEXT, and from that point forward, NEXT is calculated using PCTINCREASE as usual.

Understanding Precedence in Storage Parameters

The storage parameters in effect at a given time are determined by the following types of SQL statements, listed in order of precedence:

1. ALTER TABLE/CLUSTER/SNAPSHOT/SNAPSHOT LOG/INDEX/ROLLBACK SEGMENT statement
2. CREATE TABLE/CLUSTER/SNAPSHOT/SNAPSHOT LOG/CREATE INDEX/ROLLBACK SEGMENT statement
3. ALTER TABLESPACE statement
4. CREATE TABLESPACE statement
5. Oracle default statement

Any storage parameter specified at the object level overrides the corresponding option set at the tablespace level. When storage parameters are not explicitly set at the object level, they default to those at the tablespace level. When storage parameters are not set at the tablespace level, Oracle system defaults apply. If storage parameters are altered, the new options apply only to the extents not yet allocated.

Storage Parameter Example

Assume the following statement has been executed:

```
CREATE TABLE test_storage
   ( . . . )
   STORAGE (INITIAL 100K   NEXT 100K
      MINEXTENTS 2   MAXEXTENTS 5
      PCTINCREASE 50);
```

Also assume that the initialization parameter DB_BLOCK_SIZE is set to 2K. The following table shows how extents are allocated for the TEST_STORAGE table.

14.6 Deallocating Space

This section describes aspects of deallocating unused space, and includes the following topics:

- Viewing the High Water Mark
- Issuing Space Deallocation Statements

It is not uncommon to allocate space to a segment, only to find out later that it is not being used. For example, you may set PCTINCREASE to a high value, which could create a large extent that is only partially used. Or you could explicitly overallocate space by issuing the ALTER TABLE ALLOCATE EXTENT statement. If you find that you have unused or overallocated space, you can release it so that the unused space can be used by other segments.

Viewing the High Water Mark

Prior to deallocation, you can use the DBMS_SPACE package, which contains a procedure (UNUSED_SPACE) that returns information about the position of the high water mark and the amount of unused space in a segment.

Within a segment, the high water mark indicates the amount of used space. You cannot release space below the high water mark (even if there is no data in the space you wish to deallocate). However, if the segment is completely empty, you can release space using the TRUNCATE DROP STORAGE statement.

Issuing Space Deallocation Statements

The following statements deallocate unused space in a segment (table, index or cluster). The KEEP clause is *optional*.

```
ALTER TABLE table DEALLOCATE UNUSED KEEP integer;
ALTER INDEX index DEALLOCATE UNUSED KEEP integer;
ALTER CLUSTER cluster DEALLOCATE UNUSED KEEP integer;
```

When you explicitly identify an amount of unused space to KEEP, this space is retained while the remaining unused space is deallocated. If the remaining number of extents becomes smaller than MINEXTENTS, the MINEXTENTS value changes to reflect the new number. If the initial extent becomes smaller, the INITIAL value changes to reflect the new size of the initial extent. If you do not specify the KEEP clause, all unused space (everything above the high water mark) is deallocated, as long as the size of the initial extent and MINEXTENTS are preserved. Thus, even if the high water mark occurs within the MINEXTENTS boundary, MINEXTENTS remains and the initial extent size is not reduced. You can verify that deallocated space is freed by looking at the DBA_FREE_SPACE view.

Deallocating Space: Examples

This section includes various space deallocation scenarios.

Example : Table DQUON consists of three extents (see figure Figure 3). The first extent is 10K, the second is 20K, and the third is 30K. The high water mark is in the middle of the second extent, and there is 40K of unused space. The following statement deallocates all unused space, leaving table DQUON with two remaining extents. The third extent disappears, and the second extent size is 10K.

ALTER TABLE dquon DEALLOCATE UNUSED;

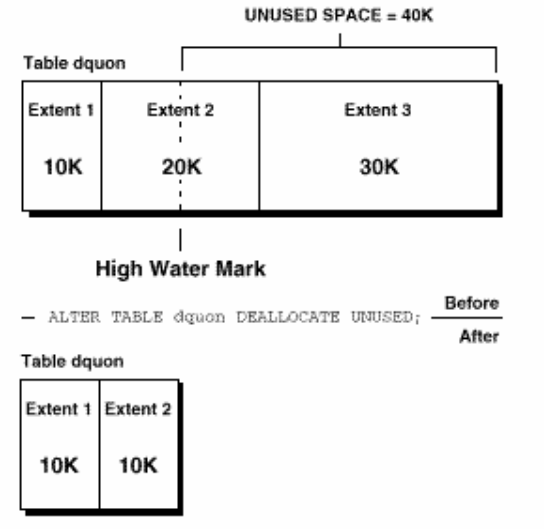


Figure 3: Deallocating All Unused Space

If you deallocate all unused space from DQUON and KEEP 10K (see Figure 4), the third extent is deallocated and the second extent remains in tact.

If you deallocate all unused space from DQUON and KEEP 20K, the third extent is cut to 10K, and the size of the second extent remains the same.

ALTER TABLE dquon DEALLOCATE UNUSED KEEP 20K;

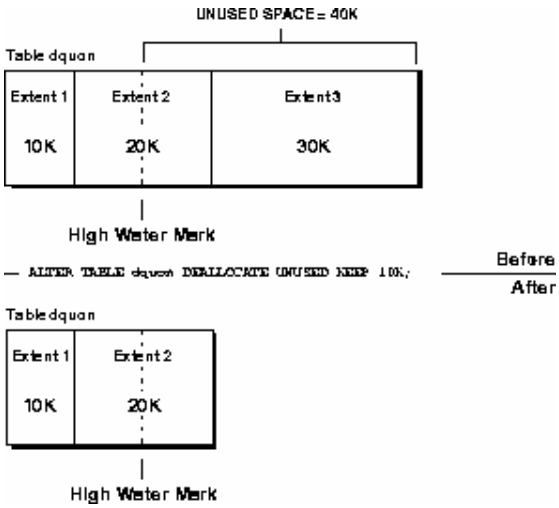


Figure 4: Deallocating Unused Space, KEEP 10K

14.7 Short Summary

The PCTFREE and PCTUSED parameters to make the following changes:

- ♣ increase the performance of writing and retrieving data
- ♣ decrease the amount of unused space in data blocks
- ♣ decrease the amount of row chaining between data blocks

14.8 Brain Storm

1. Hot backup of Control file is performed after

   a) Incomplete recovery b) Structural changes to Database
   c) Large insertion of data d) None of the above

2. A Checkpoint establishes a consistent point of the database

   a) across Redo log threads b)across the Datafiles
   c) across the objects in the cache d)all of the above

3. Which two factors determine the frequency with which backup should be performed?

   a) Size of the Database b)Frequency of structural changes made to the database
   c) Type of Operating System d)Nature of the Data

4. How would you backup an order processing database that is critical to the cash flow of your company?

   a) Backup the Database daily
   b) Backup the Database monthly
   c) Backup the Database during inventory
   d) Backup the Database weekly

Lecture 15

Managing Partitioned Tables and Indexes

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about Partition table and indexes
- Discuss about creation of partition table
- Describe about Moving, Adding, Dropping, Truncating, Splitting and Merging Partitions.

Coverage Plan

Lecture 15

15.1 Snap Shot

15.2 What Are Partitioned Tables And Indexes?

15.3 Creating Partitions

15.4 Maintaining Partitions

15.5 Short Summary

15.6 Brain Storm

15.1 Snap Shot

This chapter describes various aspects of managing partitioned tables and indexes, and includes the following sections:

- What Are Partitioned Tables and Indexes?
- Creating Partitions
- Maintaining Partitions

15.2 What Are Partitioned Tables and Indexes?

Note: Before attempting to create a partitioned table or index or perform maintenance operations on any partition, review the information about partitioning in Oracle8 Server Concepts.

Today's enterprises frequently run mission-critical databases containing upwards of several hundred gigabytes and, in many cases, several terabytes of data. These enterprises are challenged by the support and maintenance requirements of very large databases (VLDB), and must devise methods to meet those challenges.

One way to meet VLDB demands is to create and use *partitioned tables and indexes*. A partitioned table or index has been divided into a number of pieces, or *partitions,* which have the same logical attributes. For example, all partitions in a table share the same column and constraint definitions, and all partitions in an index share the same index options. Each partition is stored in a separate segment and can have different physical attributes (such as PCTFREE, PCTUSED, INITRANS, MAXTRANS, TABLESPACE, and STORAGE). Although you are not required to keep each table or index partition in a separate tablespace, it is to your advantage to do so. Storing partitions in separate tablespaces can:

- reduce the possibility of data corruption in multiple partitions
- make it possible to back up and recover each partition independently
- make it possible to control the mapping of partitions to disk drives (important for balancing I/O load)

15.3 Creating Partitions

This section describes how to create table and index partitions. Creating partitions is very similar to creating a table or index: you must use the CREATE TABLE statement with the PARTITION CLAUSE. Also, you must specify the tablespace name for each partition when you have partitions in different tablespaces.

The following example shows a CREATE TABLE statement that contains 4 partitions, one for each quarter's worth of sales. A row with SALE_YEAR=1994, SALE_MONTH=7, and SALE_DAY=18 has the partitioning key (1994, 7, 18), and is in the third partition, in the tablespace TSC. A row with SALE_YEAR=1994, SALE_MONTH=7, and SALE_DAY=1 has the partitioning key (1994, 7, 1), and also is in the third partition.
CREATE TABLE sales

```
Splitting PartitionsSplitting Partitions( invoice_no NUMBER,
     sale_year  INT NOT NULL,
     sale_month INT NOT NULL,
     sale_day   INT NOT NULL )
PARTITION BY RANGE ( sale_year, sale_month, sale_day)
    ( PARTITION sales_q1 VALUES LESS THAN ( 1994, 04, 01 )
      TABLESPACE tsa,
    PARTITION sales_q2 VALUES LESS THAN ( 1994, 07, 01 )
      TABLESPACE tsb,
    PARTITION sales_q3 VALUES LESS THAN ( 1994, 10, 01 )
```

```
  TABLESPACE tsc,
PARTITION sales q4 VALUES LESS THAN ( 1995, 01, 01 )
  TABLESPACE tsd);
```

15.4 Maintaining Partitions

This section describes how to accomplish specific partition maintenance operations, including:

- Moving Partitions
- Adding Partitions
- Dropping Partitions
- Truncating Partitions
- Splitting Partitions
- Merging Partitions

Moving Partitions

You can use the MOVE PARTITION clause of the ALTER TABLE statement to:

- re-cluster data and reduce fragmentation
- move a partition to another tablespace
- modify create-time attributes

Typically, you can change the physical storage attributes of a partition in a single step via a ALTER TABLE/INDEX MODIFY PARTITION statement. However, there are some physical attributes, such as TABLESPACE, that you cannot modify via MODIFY PARTITION. In these cases you can use the MOVE PARTITION clause.

Moving Table Partitions

You can use the MOVE PARTITION clause to move a partition. For example, a DBA wishes to move the most active partition to a tablespace that resides on its own disk (in order to balance I/O). The DBA can issue the following statement:

```
ALTER TABLE parts MOVE PARTITION depot2
  TABLESPACE ts094 NOLOGGING;
```

This statement always drops the partition's old segment and creates a new segment, even if you don't specify a new tablespace.

When the partition you are moving contains data, MOVE PARTITION marks the matching partition in each local index, and all global index partitions as unusable. You must rebuild these index partitions after issuing MOVE PARTITION.

Moving Index Partitions

Some operations, such as MOVE PARTITION and DROP TABLE PARTITION, mark all partitions of a global index unusable. You can rebuild the entire index by rebuilding each partition individually using the ALTER INDEX REBUILD PARTITION statement. You can perform these rebuilds concurrently.

You can also simply drop the index and re-create it.

Adding Partitions

This section describes how to add new partitions to a partitioned table and how partitions are added to local indexes.

Adding Table Partitions

You can use the ALTER TABLE ADD PARTITION statement to add a new partition to the "high" end (the point after the last existing partition). If you wish to add a partition at the beginning or in the middle of a table, or if the partition bound on the highest partition is MAXVALUE, you should instead use the SPLIT PARTITION statement.

When the partition bound on the highest partition is anything other than MAXVALUE, you can add a partition using the ALTER TABLE ADD PARTITION statement.

For example, a DBA has a table, SALES, which contains data for the current month in addition to the previous 12 months. On January 1, 1996, the DBA adds a partition for January:

```
ALTER TABLE sales
      ADD PARTITION jan96 VALUES LESS THAN ( '960201' )
      TABLESPACE tsx;
```

When there are local indexes defined on the table and you issue the ALTER TABLE ... ADD PARTITION statement, a matching partition is also added to each local index. Since Oracle assigns names and default physical storage attributes to the new index partitions, you may wish to rename or alter them after the ADD operation is complete.

Adding Index Partitions

You cannot explicitly add a partition to a local index. Instead, new partitions are added to local indexes only when you add a partition to the underlying table.

You cannot add a partition to a global index because the highest partition always has a partition bound of MAXVALUE. If you wish to add a new highest partition, use the ALTER INDEX SPLIT PARTITION statement.

Dropping Partitions

This section describes how to use the ALTER TABLE DROP PARTITION statement to drop table and index partitions and their data.

Dropping Table Partitions

You can use the ALTER TABLE DROP PARTITION statement to drop table partitions. If there are local indexes defined for the table, ALTER TABLE DROP PARTITION also drops the matching partition from each local index.

Dropping Table Partitions Containing Data and Global Indexes If, however, the partition contains data and global indexes, use either of the following methods to drop the table partition:

1. Leave the global indexes in place during the ALTER TABLE DROP PARTITION statement. In this situation DROP PARTITION marks all global index partitions unusable, so you must rebuild them afterwards.

Note: The ALTER TABLE DROP PARTITION statement not only marks all global index partitions as unusable, it also renders all non-partitioned indexes unusable. Because the entire partitioned index cannot be rebuilt using one statement, sal1 in the following statement is a non-partitioned index.

```
ALTER TABLE sales DROP PARTITION dec94;
ALTER INDEX sales_area_ix REBUILD sal1;
```

This method is most appropriate for large tables where the partition being dropped contains a significant percentage of the total data in the table.

2. Issue the DELETE command to delete all rows from the partition before you issue the ALTER TABLE DROP PARTITION statement. The DELETE command updates the global indexes, and also fires triggers and generates redo and undo log.

Note: You can substantially reduce the amount of logging by setting the NOLOGGING attribute (using ALTER TABLE...MODIFY PARTITION...NOLOGGING) for the partition before deleting all of its rows.
For example, a DBA wishes to drop the first partition, which has a partition bound of 10000. The DBA issues the following statements:

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales DROP PARTITION dec94;
```

This method is most appropriate for small tables, or for large tables when the partition being dropped contains a small percentage of the total data in the table.

Dropping Table Partitions Containing Data and Referential Integrity Constraints If a partition contains data and has referential integrity constraints, choose either of the following methods to drop the table partition:

1. Disable the integrity constraints, issue the ALTER TABLE DROP PARTITION statement, then enable the integrity constraints:

```
ALTER TABLE sales
DISABLE CONSTRAINT dname_sales1;
  ALTER TABLE sales DROP PARTITTION dec94;
  ALTER TABLE sales
   ENABLE CONSTRAINT dname_sales1;
```

This method is most appropriate for large tables where the partition being dropped contains a significant percentage of the total data in the table.

2. Issue the DELETE command to delete all rows from the partition before you issue the ALTER TABLE DROP PARTITION statement. The DELETE command enforces referential integrity constraints, and also fires triggers and generates redo and undo log.

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales DROP PARTITION dec94;
```

This method is most appropriate for small tables or for large tables when the partition being dropped contains a small percentage of the total data in the table.

Dropping Index Partitions

You cannot explicitly drop a partition from a local index. Instead, local index partitions are dropped only when you drop a partition from the underlying table. If a global index partition is empty, you can explicitly drop it by

issuing the ALTER INDEX DROP PARTITION statement. If a global index partition contains data, dropping the partition causes the next highest partition to be marked unusable. For example, a DBA wishes to drop the index partition P1 and P2 is the next highest partition. The DBA must issue the following statements:

```
ALTER INDEX npr DROP PARTITION P1;
ALTER INDEX npr REBUILD PARTITION P2;
```

Note: You cannot drop the highest partition in a global index.

Truncating Partitions

Use the ALTER TABLE TRUNCATE PARTITION statement when you wish to remove all rows from a table partition. You cannot truncate an index partition; however, the ALTER TABLE TRUNCATE PARTITION statement truncates the matching partition in each local index.

Truncating Partitioned Tables

You can use the ALTER TABLE TRUNCATE PARTITION statement to remove all rows from a table partition with or without reclaiming space. If there are local indexes defined for this table, ALTER TABLE TRUNCATE PARTITION also truncates the matching partition from each local index.

Truncating Table Partitions Containing Data and Global Indexes If, however, the partition contains data and global indexes, use either of the following methods to truncate the table partition:

1. Leave the global indexes in place during the ALTER TABLE TRUNCATE PARTITION statement. In this situation TRUNCATE PARTITION marks all global index partitions unusable, so you must

Note: The ALTER TABLE TRUNCATE PARTITION statement not only marks all global index partitions as unusable, it also renders all non-partitioned indexes unusable. Because the entire partitioned index cannot be rebuilt using one statement, sal1 in the following statement is a non-partitioned index.

```
ALTER TABLE sales TRUNCATE PARTITION dec94;
ALTER INDEX sales_area_ix REBUILD sal1;
```

This method is most appropriate for large tables where the partition being truncated contains a significant percentage of the total data in the table.

2. Issue the DELETE command to delete all rows from the partition before you issue the ALTER TABLE TRUNCATE PARTITION statement. The DELETE command updates the global indexes, and also fires triggers and generates redo and undo log.

This method is most appropriate for small tables, or for large tables when the partition being truncated contains a small percentage of the total data in the table.

Truncating Table Partitions Containing Data and Referential Integrity Constraints If a partition contains data and has referential integrity constraints, choose either of the following methods to truncate the table partition:

1. Disable the integrity constraints, issue the ALTER TABLE TRUNCATE PARTITION statement, then re-enable the integrity constraints:

```
ALTER TABLE sales
DISABLE CONSTRAINT dname_sales1;
ALTER TABLE sales TRUNCATE PARTITTION dec94;
```

```
ALTER TABLE sales
ENABLE CONSTRAINT dname_sales1;
```

This method is most appropriate for large tables where the partition being truncated contains a significant percentage of the total data in the table.

2. Issue the DELETE command to delete all rows from the partition before you issue the ALTER TABLE TRUNCATE PARTITION statement. The DELETE command enforces referential integrity constraints, and also fires triggers and generates redo and undo log.

Note: You can substantially reduce the amount of logging by setting the NOLOGGING attribute (using ALTER TABLE...MODIFY PARTITION...NOLOGGING) for the partition before deleting all of its rows.

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales TRUNCATE PARTITION dec94;
```

This method is most appropriate for small tables, or for large tables when the partition being truncated contains a small percentage of the total data in the table.

Splitting Partitions

This form of ALTER TABLE/INDEX divides a partition into two partitions. You can use the SPLIT PARTITION clause when a partition becomes too large and causes backup, recovery or maintenance operations to take a long time. You can also use the SPLIT PARTITION clause to redistribute the I/O load.

Splitting Table Partitions

You can split a table partition by issuing the ALTER TABLE SPLIT PARTITION statement. If there are local indexes defined on the table, this statement also splits the matching partition in each local index. Because Oracle assigns system-generated names and default storage attributes to the new index partitions, you may wish to rename or alter these index partitions after splitting them.

If the partition you are splitting contains data, the ALTER TABLE SPLIT PARTITION statement marks the matching partitions (there are two) in each local index, as well as all global index partitions, as unusable. You must rebuild these index partitions after issuing the ALTER TABLE SPLIT PARTITION statement.

Splitting a Table Partition: Scenario

In this scenario "fee_katy" is a partition in the table "VET_cats," which has a local index, JAF1. There is also a global index, VET on the table. VET contains two partitions, VET_parta, and VET_partb.

To split the partition "fee_katy", and rebuild the index partitions, the DBA issues the following statements:

```
ALTER TABLE vet_cats SPLIT PARTITION
      fee_katy at (100) INTO ( PARTITION
      fee_katy1 ..., PARTITION fee_katy2 ...);
ALTER INDEX JAF1 REBUILD PARTITION SYS_P00067;
ALTER INDEX JAF1 REBUILD PARTITION SYS_P00068;
ALTER INDEX VET REBUILD PARTITION VET_parta;
ALTER INDEX VET REBUILD PARTITION VET_partb;
```

Note: You must examine the data dictionary to locate the names assigned to the new local index partitions. In this particular scenario, they are SYS_P00067 and SYS_P00068. If you wish, you can rename them.

Also, unless JAF1 already contained partitions fee_katy1 and fee_katy2, names assigned to local index partitions produced by this split will match those of corresponding base table partitions.

Splitting Index Partitions

You cannot explicitly split a partition in a local index. A local index partition is split only when you split a partition in the underlying table.

You can issue the ALTER INDEX SPLIT PARTITION statement to split a partition in a global index if the partition is empty.

The following statement splits the index partition containing data, QUON1:

```
ALTER INDEX quon1 SPLIT
    PARTITION canada AT VALUES LESS THAN ( 100 ) INTO
    PARTITION canada1 ..., PARTITION canada2 ...);
ALTER INDEX quon1 REBUILD PARTITION canada1;
ALTER INDEX quon1 REBUILD PARTITION canada2;
```

Merging Partitions

While there is no explicit MERGE statement, you can merge a partition using either the DROP PARTITION or EXCHANGE PARTITION clauses.

Merging Table Partitions

You can use either of the following strategies to merge table partitions.
If you have data in partition OSU1 and no global indexes or referential integrity constraints on the table, OH, you can merge table partition OSU1 into the next highest partition, OSU2.

To merge partition OSU1 into partition OSU2:

1. Export the data from OSU1.

2. Issue the following statement:
   ```
   ALTER TABLE OH DROP PARTITION OSU1;
   ```

3. Import the data from Step 1 into partition OSU2.

Note: The corresponding local index partitions are also merged
Another way to merge partition OSU1 into partition OSU2:

1. Exchange partition OSU1 of table OH with "dummy" table COLS.

2. Issue the following statement:
   ```
   ALTER TABLE OH DROP PARTITION OSU1;
   ```

3. Insert as SELECT from the "dummy" table to move the data from OSU1 back into OSU2.

Merging Partitioned Indexes

The only way to merge partitions in a local index is to merge partitions in the underlying table.

If the index partition BUCKS is empty, you can merge global index partition BUCKS into the next highest partition, GOOSU, by issuing the following statement:

ALTER INDEX BUCKEYES DROP PARTITION BUCKS;

If the index partition BUCKS contains data, issue the following statements:

ALTER INDEX BUCKEYES DROP PARTITION BUCKS;

ALTER INDEX BUCKEYES REBUILD PARTITION GOOSU;
While the first statement marks partition GOOSU unusable, the second makes it valid again.

15.4 Short Summary

Storing partitions in separate tablespaces can:

- reduce the possibility of data corruption in multiple partitions
- make it possible to back up and recover each partition independently
- make it possible to control the mapping of partitions to disk drives

15.5 Brain Storm

1. Which type of backup is performed with online backup?
   a) Database level Backup
   b) Tablespace level Backup
   c) Table Level Backup

2. To ensure that all data is captured in an offline Backup the DBA should close the Database using the
   a) Shutdown Immediate
   b) Shutdown Normal
   c) Shutdown Transactional
   d) Check all that apply

3. Cumulative database EXPORT exports only tables that having modified or created
   a) since the most incremental export
   b) since the most recent cumulative export
   c) since the most complete export
   d) check all that apply

Lecture 16

Managing Rollback Segment

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about guidelines for managing rollback segments
- Describe the creation of rollback segments
- Discuss the Storage Parameters specification of rollback segments
- Discuss about taking online and offline for rollback segments
- Monitoring rollback segment and dropping method of rollback segment

Coverage Plan

Lecture 16

16.1 Snap Shot

16.2 Guidelines For Managing Rollback Segments

16.3 Creating Rollback Segments

16.4 Specifying Storage Parameters

16.5 Taking Rollback Segments Online And Offline

16.6 Dropping Rollback Segments

16.7 Monitoring Rollback Segment

16.8 Short Summary

16.9 Brain Storm

16.1 Snap Shot

This Lecture describes how to manage rollback segments, and includes the following topics:

- Guidelines for Managing Rollback Segments
- Creating Rollback Segments
- Specifying Storage Parameters for Rollback Segments
- Taking Rollback Segments Online and Offline
- Dropping Rollback Segments
- Monitoring Rollback Segment Information

16.2 Guidelines for Managing Rollback Segments

This section describes guidelines to consider before creating or managing the rollback segments of your databases, and includes the following topics:

- Use Multiple Rollback Segments
- Choose Between Public and Private Rollback Segments
- Specify Rollback Segments to Acquire Automatically

Every database contains one or more *rollback segments*, which are portions of the database that record the actions of transactions in the event that a transaction is rolled back. You use rollback segments to provide read consistency, rollback transactions, and recover the database.

Use Multiple Rollback Segments

Using multiple rollback segments distributes rollback segment contention across many segments and improves system performance. Multiple rollback segments are required in the following situations:

- When a database is created, a single rollback segment named SYSTEM is created in the SYSTEM tablespace. If a database is to have other tablespaces, it *must* have two or more rollback segments in the SYSTEM tablespace. You cannot create any objects in non-SYSTEM tablespaces (not even rollback segments) until you have created and brought online at least one additional rollback segment in the SYSTEM tablespace.

- When many transactions are concurrently proceeding, more rollback information is generated at the same time. You can indicate the number of concurrent transactions you expect for the instance with the parameter TRANSACTIONS, and the number of transactions you expect each rollback segment to have to handle with the parameter TRANSACTIONS_PER_- ROLLBACK_SEGMENT. Then, when an instance opens a database, it attempts to acquire at least TRANSACTIONS/ TRANSACTIONS_PER_ROLLBACK_ SEGMENT rollback segments to handle the maximum amount of transactions. Therefore, after setting the parameters, create TRANSACTIONS/TRANSACTIONS_PER_ ROLLBACK_ SEGMENT rollback segments.

Add a Rollback Segment to the SYSTEM Tablespace

An initial rollback segment called SYSTEM is created when a database is created. The SYSTEM rollback segment is created in the SYSTEM tablespace using the default storage parameters associated with that tablespace. You cannot drop this rollback segment.

An instance always acquires the SYSTEM rollback segment in addition to any other rollback segments it needs. However, if there are multiple rollback segments, Oracle tries to use the SYSTEM rollback segment only for special system transactions and distributes user transactions among other rollback segments; if there are too many transactions for the non-SYSTEM rollback segments, Oracle uses the SYSTEM segment. Therefore, after database creation, create at least one additional rollback segment in the SYSTEM tablespace.

Choose Between Public and Private Rollback Segments

A *private rollback segment* is acquired explicitly by an instance when the instance opens the database. *Public rollback segments* form a pool of rollback segments that any instance requiring a rollback segment can use.

If a database does not have the Parallel Server option, public and private rollback segments are identical. Therefore, you can create all public rollback segments. A database with the Parallel Server option can also have only public segments, as long as the number of segments is high enough that each instance opening the database can acquire at least one rollback segment in addition to its SYSTEM rollback segment. You may also use private rollback segments when using the Oracle Parallel Server.

Specify Rollback Segments to Acquire Automatically

When an instance starts, it acquires by default TRANSACTIONS/TRANSACTIONS_PER_ ROLLBACK_SEGMENT rollback segments. If you want to ensure that the instance acquires particular rollback segments that have particular sizes or particular tablespaces, specify the rollback segments by name in the ROLLBACK_SEGMENTS parameter in the instance's parameter file.

The instance acquires all the rollback segments listed in this parameter, even if more than TRANSACTIONS/TRANSACTIONS_PER_ROLL-BACK_SEGMENT segments are specified. The rollback segments can be either private or public.

16.3 Creating Rollback Segments

To create rollback segments, you must have the CREATE ROLLBACK SEGMENT system privilege. To create additional rollback segments for a database, use either the Create Rollback Segment property sheet of Enterprise Manager, or the SQL command CREATE ROLLBACK SEGMENT. The tablespace to contain the new rollback segment must be online.

The following statement creates a public rollback segment named USERS_RS in the USERS tablespace, using the default storage parameters of the USERS tablespace:

CREATE PUBLIC ROLLBACK SEGMENT users_rs TABLESPACE users;

Bringing New Rollback Segments Online

If you create a private rollback segment, you should add the name of this new rollback segment to the ROLLBACK_SEGMENTS parameter in the parameter file for the database. Doing so enables the private rollback segment to be captured by the instance at instance start up. For example, if two new private rollback segments are created and named RS1 and RS2, the ROLLBACK_SEGMENTS parameter of the parameter file should be similar to the following:

```
ROLLBACK SEGMENTS= (RS1, RS2)
```

16.4 Specifying Storage Parameters for Rollback Segments

This section describes aspects of specifying rollback segment storage parameters, and includes the following topics:

- Setting Storage Parameters When Creating a Rollback Segment
- Changing Rollback Segment Storage Parameters
- Altering Rollback Segment Format

- Shrinking a Rollback Segment Manually

Setting Storage Parameters when Creating a Rollback Segment

Suppose you wanted to create a public rollback segment DATA1_RS with storage parameters and optimal size set as follows:

- The rollback segment is allocated an initial extent of 50K.
- The rollback segment is allocated the second extent of 50K.
- The optimal size of the rollback segment is 750K.
- The minimum number of extents and the number of extents initially allocated when the segment is created is 15.
- The maximum number of extents that the rollback segment can allocate, including the initial extent, is 100.

The following statement creates a rollback segment with these characteristics:

```
CREATE PUBLIC ROLLBACK SEGMENT data1_rs
                TABLESPACE users
                STORAGE (
      INITIAL 50K
      NEXT 50K
      OPTIMAL 750K
      MINEXTENTS 15
      MAXEXTENTS 100);
```

You can also use the Create Rollback Segment property sheet of Enterprise Manager to set the rollback segment's storage parameters.

Changing Rollback Segment Storage Parameters

You can change a rollback segment's storage parameters after creating it. However, you cannot alter the size of any extent currently allocated to a rollback segment. You can only affect future extents.

Alter a rollback segment's storage parameters using either the Alter Rollback Segment property sheet of Enterprise Manager, or the SQL command ALTER ROLLBACK SEGMENT.

The following statement alters the maximum number of extents that the DATA1_RS rollback segment can allocate.

```
ALTER PUBLIC ROLLBACK SEGMENT data1_rs
         STORAGE (MAXEXTENTS 120);
```

You can alter the settings for the SYSTEM rollback segment, including the OPTIMAL parameter, just as you can alter those of any rollback segment.

Note: If you are altering a public rollback segment, you must include the keyword PUBLIC in the ALTER ROLLBACK SEGMENT command.

Altering Rollback Segment Format

To alter rollback segments, you must have the ALTER ROLLBACK SEGMENT system privilege.

You can define limited or unlimited format for rollback segments. When converting to limited or unlimited format, you *must* take the rollback segments offline. If you identify unlimited format for rollback segments, extents for that segment must have a minimum of 4 data blocks. Thus, a limited format rollback segment cannot be converted to unlimited format if it has less than 4 data blocks in any extent. If you want to convert from limited to unlimited format and have less than 4 data blocks in an extent, your only choice is to drop and re-create the rollback segment.

Shrinking a Rollback Segment Manually

To shrink a rollback segment using you must have the ALTER ROLLBACK SEGMENT system privilege.

You can manually decrease the size of a rollback segment using the SQL command ALTER ROLLBACK SEGMENT. The rollback segment you are trying shrink must be online.

The following statement shrinks rollback segment RBS1 to 100K:

```
ALTER ROLLBACK SEGMENT rbs1 SHRINK TO 100K;
```

16.5 Taking Rollback Segments Online and Offline

This section describes aspects of taking rollback segments online and offline, and includes the following topics:

- Bringing Rollback Segments Online
- Taking Rollback Segments Offline

A rollback segment is either *online* and available to transactions, or *offline* and unavailable to transactions. Generally, rollback segments are online and available for use by transactions.

You may wish to take online rollback segments offline in the following situations:

- When you want to take a tablespace offline, and the tablespace contains rollback segments. You cannot take a tablespace offline if it contains rollback segments that transactions are currently using. To prevent associated rollback segments from being used, you can take them offline before taking the tablespace offline.

- You want to drop a rollback segment, but cannot because transactions are currently using it. To prevent the rollback segment from being used, you can take it offline before dropping it.

Note: You cannot take the SYSTEM rollback segment offline.

You might later want to bring an offline rollback segment back online so that transactions can use it. When a rollback segment is created, it is initially offline, and you must explicitly bring a newly created rollback segment online before it can be used by an instance's transactions. You can bring an offline rollback segment online via any instance accessing the database that contains the rollback segment.

Bringing Rollback Segments Online

You can bring online only a rollback segment whose current status (as shown in the DBA_ROLLBACK_SEGS data dictionary view) is OFFLINE or PARTLY AVAILABLE. To bring an offline rollback segment online, use either the Place Online menu item of Enterprise Manager or the SQL command ALTER ROLLBACK SEGMENT with the ONLINE option.

Bringing a PARTLY AVAILABLE Rollback Segment Online

A rollback segment in the PARTLY AVAILABLE state contains data for an in-doubt or recovered distributed transaction, and yet to be recovered transactions. You can view its status in the data dictionary view DBA_ROLLBACK_SEGS as PARTLY AVAILABLE. The rollback segment usually remains in this state until the transaction is resolved either automatically by RECO, or manually by a DBA. However, you might find that all rollback segments are PARTLY AVAILABLE. In this case, you can bring a PARTLY AVAILABLE segment online, as described above.

Some resources used by the rollback segment for the in-doubt transaction remain inaccessible until the transaction is resolved. As a result, the rollback segment may have to grow if other transactions assigned to it need additional space.

As an alternative to bringing a PARTLY AVAILABLE segment online, you might find it easier to create a new rollback segment temporarily, until the in-doubt transaction is resolved.

Bringing Rollback Segment Online Automatically

If you would like a rollback segment to be automatically brought online whenever you start up the database, add the segment's name to the ROLLBACK_SEGMENTS parameter in the database's parameter file.

Bringing Rollback Segments Online: Example

The following statement brings the rollback segment USER_RS_2 online:

```
ALTER ROLLBACK SEGMENT user_rs_2 ONLINE;
```

After you bring a rollback segment online, its status in the data dictionary view DBA_ROLLBACK_SEGS is ONLINE.

Taking Rollback Segments Offline

To take an online rollback segment offline, use either the Take Offline menu item of Enterprise Manager, or the ALTER ROLLBACK SEGMENT command with the OFFLINE option. The rollback segment's status in the DBA_ROLLBACK_SEGS data dictionary view must be "ONLINE", and the rollback segment must be acquired by the current instance.

The following example takes the rollback segment USER_RS_2 offline:

```
ALTER ROLLBACK SEGMENT user_rs_2 OFFLINE;
```

If you try to take a rollback segment that does not contain active rollback entries offline, Oracle immediately takes the segment offline and changes its status to "OFFLINE".

In contrast, if you try to take a rollback segment that contains rollback data for active transactions (local, remote, or distributed) offline, Oracle makes the rollback segment unavailable to future transactions and takes it offline after all the active transactions using the rollback segment complete. Until the transactions complete, the rollback segment cannot be brought online by any instance other than the one that was trying to take it offline. During this period, the rollback segment's status in the view DBA_ROLLBACK_SEGS remains ONLINE; however, the rollback segment's status in the view V$ROLLSTAT is PENDING OFFLINE.

The instance that tried to take a rollback segment offline and caused it to change to PENDING OFFLINE can bring it back online at any time; if the rollback segment is brought back online, it will function normally.

Taking Public and Private Rollback Segments Offline

After you take a public or private rollback segment offline, it remains offline until you explicitly bring it back online *or* you restart the instance.

16.6 Dropping Rollback Segments

You can drop rollback segments when the extents of a segment become too fragmented on disk, or the segment needs to be relocated in a different tablespace.

Before dropping a rollback segment, make sure that status of the rollback segment is OFFLINE. If the rollback segment that you want to drop is currently ONLINE, PARTLY AVAILABLE, NEEDS RECOVERY, or INVALID, you cannot drop it. If the status is INVALID, the segment has already been dropped. Before you can drop it, you must take it offline.

To drop a rollback segment, you must have the DROP ROLLBACK SEGMENT system privilege.

If a rollback segment is offline, you can drop it using either the Drop menu item of Enterprise Manager, or the SQL command DROP ROLLBACK SEGMENT.

```
The following statement drops the DATA1_RS rollback segment:

DROP PUBLIC ROLLBACK SEGMENT data1_rs;
```

If you use the DROP ROLLBACK SEGMENT command, indicate the correct type of rollback segment to drop, public or private, by including or omitting the PUBLIC keyword.

Note: If a rollback segment specified in ROLLBACK_SEGMENTS is dropped, make sure to edit the parameter files of the database to remove the name of the dropped rollback segment from the list in the ROLLBACK_SEGMENTS parameter. If this step is not performed before the next instance startup, startup fails because it cannot acquire the dropped rollback segment.

After a rollback segment is dropped, its status changes to INVALID. The next time a rollback segment is created, it takes the row vacated by a dropped rollback segment, if one is available, and the dropped rollback segment's row no longer appears in the DBA_ROLLBACK_SEGS view.

16.7 Monitoring Rollback Segment Information

Use the MONITOR ROLLBACK feature of Enterprise Manager/GUI to monitor a rollback segment's size, number of extents, optimal number of extents, activity concerning dynamic deallocation of extents, and current usage by active transaction.

Displaying Rollback Segment Information

The DBA_ROLLBACK_SEGS data dictionary view stores information about the rollback segments of a database. For example, the following query lists the name, associated tablespace, and status of each rollback segment in a database:

```
SELECT segment_name, tablespace_name, status
        FROM sys.dba_rollback_segs;
```

```
SEGMENT_NAME  TABLESPACE_NAME   STATUS
------------- ----------------  ------
SYSTEM        SYSTEM            ONLINE
PUBLIC_RS     SYSTEM            ONLINE
USERS_RS      USERS             ONLINE
```

In addition, the following data dictionary views contain information about the segments of a database, including rollback segments:

- USER_SEGMENTS
- DBA_SEGMENTS

16.8 Short Summary

A *private rollback segment* is acquired explicitly by an instance when the instance opens the database. *Public rollback segments* form a pool of rollback segments that any instance requiring a rollback segment can use.

Transactions are explicitly assigned to rollback segments for the following reasons:

- The anticipated amount of rollback information generated by a transaction can fit in the current extents of the assigned rollback segment.

- Additional extents do not have to be dynamically allocated (and subsequently truncated) for rollback segments, which reduce overall system performance.

16.9 Brain Storm

1. What is Rollback Segments? What are the uses?
2. What are the Contents of Rollback Segment?
3. What is "Transaction Rollback" of Rollback Segments?
4. How do you allocate a particular transaction to specific Rollback Segment?
5. Explain "Transaction Table" of a Rollback Segment?
6. Explain "HEAD" & "TAIL" of the Rollback Segment?

Lecture 17

Security Policy

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about Database User Management, User Authentication and Operating System Security
- Describe about data and user security
- In user security discuss about general, end-user, administrator and application developers
- Discuss the password and auditing security

Coverage Plan

Lecture 17

17.1 Snap Shot

17.2 System Security Policy

17.3 Data Security Policy

17.4 User Security Policy

17.5 Password Management Policy

17.6 Auditing Policy

17.7 Short Summary

17.8 Brain Storm

17.1 Snap Shot

This chapter provides guidelines for developing security policies for database operation, and includes the following topics:

- System Security Policy
- Data Security Policy
- User Security Policy
- Password Management Policy
- Auditing Policy

17.2 System Security Policy

This section describes aspects of system security policy, and includes the following topics:

- Database User Management
- User Authentication
- Operating System Security

Each database has one or more administrators who are responsible for maintaining all aspects of the security policy: the security administrators. If the database system is small, the database administrator may have the responsibilities of the security administrator. However, if the database system is large, a special person or group of people may have responsibilities limited to those of a security administrator.

After deciding who will manage the security of the system, a security policy must be developed for every database. A database's security policy should include several sub-policies, as explained in the following sections.

Database User Management

Database users are the access paths to the information in an Oracle database. Therefore, tight security should be maintained for the management of database users. Depending on the size of a database system and the amount of work required to manage database users, the security administrator may be the only user with the privileges required to create, alter, or drop database users. On the other hand, there may be a number of administrators with privileges to manage database users. Regardless, only trusted individuals should have the powerful privileges to administer database users.

User Authentication

Database users can be *authenticated* (verified as the correct person) by Oracle using the host operating system, network services, or the database. Generally, user authentication via the host operating system is preferred for the following reasons:

- Users can connect to Oracle faster and more conveniently without specifying a username or password.
- Centralized control over user authorization in the operating system: Oracle need not store or manage user passwords and usernames if the operating system and database correspond.
- User entries in the database and operating system audit trails correspond.

User authentication by the database is normally used when the host operating system cannot support user authentication.

Operating System Security

If applicable, the following security issues must also be considered for the operating system environment executing Oracle and any database applications:

- Database administrators must have the operating system privileges to create and delete files.
- Typical database users should not have the operating system privileges to create or delete files related to the database.
- If the operating system identifies database roles for users, the security administrators must have the operating system privileges to modify the security domain of operating system accounts.

17.3 Data Security Policy

Data security includes the mechanisms that control the access and use of the database at the object level. Your data security policy determines which users have access to a specific schema object, and the specific types of actions allowed for each user on the object. For example, user SCOTT can issue SELECT and INSERT statements but not DELETE statements using the EMP table. Your data security policy should also define the actions, if any, that are audited for each schema object.

Primarily the level of security you wish to establish for the data in your database will determine your data security policy. For example, it may be acceptable to have little data security in a database when you wish to allow any user to create any schema object, or grant access privileges for their objects to any other user of the system. Alternatively, it might be necessary for data security to be very controlled when you wish to make a database or security administrator the only person with the privileges to create objects and grant access privileges for objects to roles and users. Overall data security should be based on the sensitivity of data. If information is not sensitive, then the data security policy can be more lax. However, if data is sensitive, a security policy should be developed to maintain tight control over access to objects.

17.4 User Security Policy

This section describes aspects of user security policy, and includes the following topics:

- General User Security
- End-User Security
- Administrator Security
- Application Developer Security
- Application Administrator Security

General User Security

For all types of database users, consider the following general user security issues:

- Password Security
- Privilege Management

Password Security

If user authentication is managed by the database, security administrators should develop a password security policy to maintain database access security. For example, database users should be required to change their passwords at regular intervals, and of course, when their passwords are revealed to others. By forcing a user to modify passwords in such situations, unauthorized database access can be reduced.

Secure Connections with Encrypted Passwords

To better protect the confidentiality of your password, Oracle can be configured to use encrypted passwords for client/server and server/server connections.

By setting the following values, you can require that the password used to verify a connection always be encrypted:

- Set the ORA_ENCRYPT_LOGIN environment variable to TRUE on the client machine.
- Set the DBLINK_ENCRYPT_LOGIN server initialization parameter to TRUE.

The DBLINK_ENCRYPT_LOGIN parameter is used for connections between two Oracle servers (for example, when performing distributed queries). If you are connecting from a client, Oracle checks the ORA_ENCRYPT_LOGIN environment variable. Oracle then checks the appropriate DBLINK_ENCRYPT_LOGIN or ORA_ENCRYPT_LOGIN value. If it set to FALSE, Oracle attempts the connection again using an unencrypted version of the password.

Privilege Management

Security administrators should consider issues related to privilege management for all types of users. For example, in a database with many usernames, it may be beneficial to use roles (which are named groups of related privileges that you grant to users or other roles) to manage the privileges available to users. Alternatively, in a database with a handful of usernames, it may be easier to grant privileges explicitly to users and avoid the use of roles.

Security administrators managing a database with many users, applications, or objects should take advantage of the benefits offered by roles. Roles greatly simplify the task of privilege management in complicated environments.

End-User Security

Security administrators must also define a policy for end-user security. If a database is large with many users, the security administrator can decide what groups of users can be categorized, create user roles for these user groups, grant the necessary privileges or application roles to each user role, and assign the user roles to the users. To account for exceptions, the security administrator must also decide what privileges must be explicitly granted to individual users.

The following actions, performed by the database or security administrator, address this simple security situation:

1. Create a role named ACCOUNTANT.
2. Grant the roles for the ACCTS_RECEIVABLE and ACCTS_PAYABLE database applications to the ACCOUNTANT role.

3. Grant each user of the accounting department the ACCOUNTANT role.

This security model is illustrated in Figure 17-1.

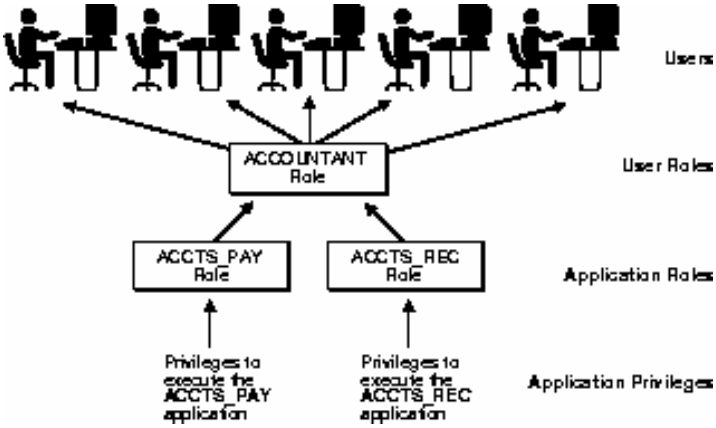


Figure 17-1: User Role

This plan addresses the following potential situations:

- If accountants subsequently need a role for a new database application, that application's role can be granted to the ACCOUNTANT role, and all users in the accounting department will automatically receive the privileges associated with the new database application. The application's role does not need to be granted to individual users requiring use of the application.

- Similarly, if the accounting department no longer requires the need for a specific application, the application's role can be dropped from the ACCOUNTANT role.

- If the privileges required by the ACCTS_RECEIVABLE or ACCTS_PAYABLE applications change, the new privileges can be granted to, or revoked from, the application's role. The security domain of the ACCOUNTANT role, and all users granted the ACCOUNTANT role automatically reflect the privilege modification.

- You have an index where a user requires only 1 role.

When possible, utilize roles in all possible situations to make end-user privilege management efficient and simple.

Administrator Security

Security administrators should have a policy addressing administrator security. For example, when the database is large and there are several types of database administrators, the security administrator may decide to group related administrative privileges into several administrative roles. The administrative roles can then be granted to appropriate administrator users. Alternatively, when the database is small and has only a few administrators, it may be more convenient to create one administrative role and grant it to all administrators.

Protection for Connections as SYS and SYSTEM

After database creation, *immediately* change the passwords for the administrative SYS and SYSTEM usernames to prevent unauthorized access to the database. Connecting as SYS and SYSTEM give a user the powerful privileges to modify a database in many ways. Therefore, privileges for these usernames are extremely sensitive, and should only be available to select database administrators.

Protection for Administrator Connections

Only database administrators should have the capability to connect to a database with administrator privileges. Connecting as SYSDBA gives a user unrestricted privileges to do anything to a database (such as startup, shutdown, and recover) or the objects within a database (such as create, drop, and delete from). Consider a scenario where the database administrator responsibilities at a large installation are shared among several database administrators, each responsible for the following specific database management jobs:

- an administrator responsible for object creation and maintenance
- an administrator responsible for database tuning and performance
- a security administrator responsible for creating new users, granting roles and privileges to database users
- a database administrator responsible for routine database operation (for example, startup, shutdown, backup)
- an administrator responsible for emergency situations, such as database recovery
- new, inexperienced database administrators needing limited capabilities to experiment with database management

In this scenario, the security administrator should structure the security for administrative personnel as follows:

1. Six roles should be defined to contain the distinct privileges required to accomplish each type of job (for example, DBA_OBJECTS, DBA_TUNE, DBA_SECURITY, DBA_MAINTAIN, DBA_RECOV, DBA_NEW).
2. Each role is granted the appropriate privileges.
3. Each type of database administrator can be granted the corresponding role.

This plan diminishes the likelihood of future problems in the following ways:

- If a database administrator's job description changes to include more responsibilities, that database administrator can be granted other administrative roles corresponding to the new responsibilities.
- If a database administrator's job description changes to include fewer responsibilities, that database administrator can have the appropriate administrative roles revoked.
- The data dictionary always stores information about each role and each user, so information is available to disclose the task of each administrator.

Application Developer Security

Security administrators must define a special security policy for the application developers using a database. A security administrator may grant the privileges to create necessary objects to application developers. Alternatively, the privileges to create objects may only be granted to a database administrator, who receives requests for object creation from developers. A security administrator's decision regarding this issue should be based on the following:

- the control desired over a database's space usage
- the control desired over the access paths to schema objects
- the database used to develop applications-if a test database is being used for application development, a more liberal development policy would be in order

Roles and Privileges for Application Developers

Security administrators can create roles to manage the privileges required by the typical application developer. For example, a typical role named APPLICATION_DEVELOPER might include the CREATE TABLE, CREATE VIEW, and CREATE PROCEDURE system privileges. Consider the following when defining roles for application developers:

- CREATE system privileges are usually granted to application developers so that they can create their own objects. However, CREATE ANY system privileges, which allow a user to create an object in any user's domain, are not usually granted to developers. This restricts the creation of new objects only to the developer's user account.

- Object privileges are rarely granted to roles used by application developers. This is often impractical because granting object privileges via roles often restricts their usability in the creation of other objects (primarily views and stored procedures). It is more practical to allow application developers to create their own objects for development purposes.

Application Administrator Security

In large database systems with many database applications (for example, precompiler and Forms applications), you might want to have application administrators. An application administrator is responsible for the following types of tasks:

- creating roles for an application and managing the privileges of each application role
- creating and managing the objects used by a database application
- maintaining and updating the application code and Oracle procedures and packages, as necessary

Often, an application administrator is also the application developer that designed the application. However, these jobs might not be the responsibility of the developer and can be assigned to another individual familiar with the database application.

17.5 Password Management Policy

Database security systems depend on passwords being kept secret at all times. Still, passwords are vulnerable to theft, forgery, and misuse.To allow for greater control over database security, Oracle's password management policy is controlled by DBAs.

Account Locking

When a particular user exceeds a designated number of failed login attempts, the server automatically locks that user's account. DBAs specify the permissible number of failed login attempts using the CREATE PROFILE statement. DBAs also specify the amount of time accounts remain locked. In the following example, the maximum number of failed login attempts for the user ASHWINI is 4, and the amount of time the account will remain locked is 30 days; the account will unlock automatically after the passage of 30 days.

```
CREATE PROFILE prof LIMIT
FAILED_LOGIN_ATTEMPTS 4
ACCOUNT_LOCK_TIME 1/24;
ALTER USER ashwini PROFILE prof;
```

If the DBA does not specify a time interval for unlocking the account, ACCOUNT_LOCK _TIME reverts to a default value. If the DBA specifies ACCOUNT_LOCK_TIME as UNLIMITED, then the system security officer must explicitly unlock the account. Thus, the amount of time an account remains locked depends upon how the

DBA configures the resource profile assigned to the user. After a user successfully logs into an account, that user's unsuccessful login attempt count, if there is one, is reset to 0. The security officer can also explicitly lock user accounts. When this occurs, the account cannot be unlocked automatically; only the security officer should unlock the account.

Password Aging and Expiration

DBAs use the CREATE PROFILE statement to specify a maximum lifetime for passwords. When the specified amount of time passes and the password expires, the user or DBA must change the password. The following statement indicates that ASHWINI can use the same password for 90 days before it expires:

```
CREATE PROFILE prof LIMIT
   FAILED_LOGIN_ATTEMPTS 4
   ACCOUNT_LOCK_TIME 30
   PASSWORD_LIFE_TIME 90;
ALTER USER ashwini PROFILE prof;
```

DBAs can also specify a grace period using the CREATE PROFILE statement. Users enter the grace period upon the first attempt to login to a database account after their password has expired. During the grace period, a warning message appears each time users try to log in to their accounts, and continues to appear until the grace period expires. Users must change the password within the grace period. If the password is not changed within the grace period, the account expires and no further log ins to that account are allowed until the password is changed. Figure 17-2 shows the chronology of the password lifetime and grace period.

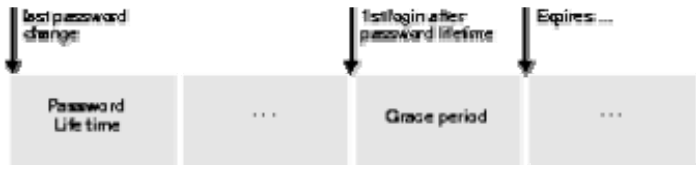


Figure 17-2: Chronology of Password Lifetime and Grace Period.

For example, the lifetime of a password is 60 days, and the grace period is 3 days. If the user tries to log in on *any* day after the 60th day (this could be the 70th day, 100th day, or another; the point here is that it is the first log in attempt after the password lifetime), that user receives a warning message indicating that the password is about to expire in 3 days. If the user does not change the password within three days from the first day of the grace period, the user's account expires. The following statement indicates that the user must change the password within 3 days of its expiration:

```
CREATE PROFILE prof LIMIT
   FAILED_LOGIN_ATTEMPTS 4
   ACCOUNT_LOCK_TIME 30
   PASSWORD_GRACE_TIME 3;
ALTER USER ashwini PROFILE prof;
```

The security officer can also explicitly expire the account. This is particularly useful for new accounts.

Password History

DBAs use the CREATE PROFILE statement to specify a time interval during which users cannot reuse a password. In the following statement, the DBA indicates that the user cannot reuse her password for 60 days.

```
CREATE PROFILE prof LIMIT
   PASSWORD_REUSE_TIME 60
   PASSWORD_REUSE_MAX UNLIMITED;
```

The next statement shows that the number of password changes the user must make before her current password can be used again is 3.

```
CREATE PROFILE prof LIMIT
PASSWORD_REUSE_MAX 3
PASSWORD_REUSE_TIME UNLIMITED;
```

Password Complexity Verification

Oracle's password complexity verification routine can be specified using a PL/SQL script (utlpwdmg.sql), which sets the default profile parameters. The password complexity verification routine performs the following checks:

- The password has a minimum length of 4.
- The password is not the same as the userid.
- The password has at least one alpha, one numeric and one punctuation mark.
- The password does not match simple words like welcome, account, database, or user.
- The password differs from the previous password by at least 3 letters.

Password Verification Routine Formatting Guidelines

DBAs can enhance the existing password verification complexity routine or create their own password verification routines using PL/SQL or third party tools. The DBA-authored PL/SQL call must adhere to the following format:

```
routine_name (
userid_parameter IN VARCHAR(30),
password_parameter IN VARCHAR(30),
old_password_parameter IN VARCHAR(30)
)
RETURN BOOLEAN
```

After a new routine is created, it must be assigned as the password verification routine using the user's profile or the system default profile.

```
CREATE/ALTER PROFILE profile_name LIMIT
PASSWORD_VERIFY_FUNCTION routine_name
The password verify routine must be owned by SYS.
Password Verification Routine: Sample Script
```

The following sample script sets default password resource limits and provides minimum checking of password complexity. You can use this sample script as a model when developing your own complexity checks for a new password.

This script sets the default password resource parameters, and must be run to enable the password features. However, you can change the default resource parameters if necessary. The default password complexity function performs the following minimum complexity checks:

- The password satisfies minimum length requirements.
- Ensures the password is not the username. You can modify this function based on your requirements.

17.6 Auditing Policy

Security administrators should define a policy for the auditing procedures of each database. You may, for example, decide to have database auditing disabled unless questionable activities are suspected. When auditing is required, the security administrator must decide what level of detail to audit the database; usually, general system auditing is followed by more specific types of auditing after the origins of suspicious activity are determined.

17.7 Short Summary

Each database has one or more administrators who are responsible for maintaining all aspects of the security policy: the security administrators.
Data security includes the mechanisms that control the access and use of the database at the object level.

The password complexity verification routine performs the following checks:

- The password has a minimum length of 4.
- The password is not the same as the userid.
- The password has at least one alpha, one numeric and one punctuation mark.
- The password does not match simple words like welcome, account, database, or user.
- The password differs from the previous password by at least 3 letters.

17.8 Brain Storm

1. Which parameter should be set when setting LICENSE_SESSION_WARNING
a) LICENSE_SESSION
b) LICENSE_MAX_SESSION
c) LICENSE_MAX_USERS
d) None of the above

2. To change the maximum concurrent usage limit or warning limit at runtime, which appropriate option you will use for that session.

   a) ALTER SESSION b) ALTER SYSTEM c) ALTER USER d) ALTER DATABASE

3. SYS discovers that the AUD$ Table cannot extent any more, he would like to have the auditing done into Operating System files instead of the Database, What is the INIT parameter files to include?

4. Which view is used to check the user account status
   a) DBA_USERS b) USER_USERS c)DBA_PROFILES d)ALL_USERS
   e)USER_PASSWORD_LIMITS

5. CREATE PROFILE P1
```
SESSION_PER_USER2
PASSWORD_REUSE_TIME 2
PASSWORD_LOCK_TIME 1
FAILED_LOGIN_ATTEMPTS 2
IDLE_TIME 30;
Find the errors.
```

6. To set the Resource Cost, Which system privilege is used

   a) ALTER PROFILE b) ALTER RESOURCE COST c) CREATE PROFILE
   b) No privilege is required

Lecture 18

User and Resources

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about session and user licensing
- Discuss about Concurrent Usage Licensing, Connecting Privileges, Setting the Maximum Number of Sessions, Setting the Session Warning Limit, Changing Concurrent Usage Limits While the Database is Running, Named User Limits, Viewing Licensing Limits and Current Values
- Describe about user authentication
- Managing users resources and profiles for oracle user.

Coverage Plan

Lecture 18

18.1 Snap Shot

18.2 Session And User Licensing

18.3 User Authentication

18.4 Oracle Users

18.5 Managing Resources With Profiles

18.6 Listing Information About Database Users And Profiles

18.7 Short Summary

18.8 Brain Storm

18.1 Snap Shot

This chapter describes how to control access to an Oracle database, and includes the following topics:

- Session and User Licensing
- User Authentication
- Oracle Users
- Managing Resources with Profiles
- Listing Information About Database Users and Profiles

Privileges and roles control the access a user has to a database and the schema objects within the database.

18.2 Session and User Licensing

This section describes aspects of session and user licensing, and includes the following topics:

- Concurrent Usage Licensing
- Connecting Privileges
- Setting the Maximum Number of Sessions
- Setting the Session Warning Limit
- Changing Concurrent Usage Limits While the Database is Running
- Named User Limits
- Viewing Licensing Limits and Current Values

Oracle helps you ensure that your site complies with its Oracle Server license agreement. If your site is licensed by concurrent usage, you can track and limit the number of sessions concurrently connected to a database. If named users license your site, you can limit the number of named users created in a database. In either case, you control the licensing facilities, and must enable the facilities and set the appropriate limits.

To use the licensing facility, you need to know which type of licensing agreement your site has, and what the maximum number of sessions or named users is. Your site may use either type of licensing (concurrent usage or named user), but not both.

Concurrent Usage Licensing

Concurrent usage licensing limits the number of sessions that can be connected simultaneously to the database on the specified computer. You can set a limit on the number of concurrent sessions before you start an instance. In fact, you should have set this limit as part of the initial installation procedure. You can also change the maximum number of concurrent sessions while the database is running.

Connecting Privileges

After your instance's session limit is reached, only users with RESTRICTED SESSION privilege (usually DBAs) can connect to the database. When a user with RESTRICTED SESSION privileges connects, Oracle sends the user a message indicating that the maximum limit has been reached, and writes a message to the ALERT file. When the maximum is reached, you should connect only to terminate unneeded processes. Do not raise the licensing limits unless you have upgraded your Oracle license agreement.

In addition to setting a maximum concurrent session limit, you can set a warning limit on the number of concurrent sessions. After this limit is reached, additional users can continue to connect (up to the maximum limit); however, Oracle writes an appropriate message to the ALERT file with each connection, and sends each

connecting user who has the RESTRICTED SESSION privilege a warning indicating that the maximum is about to be reached.

If a user is connecting with administrator privileges, the limits still apply; however, Oracle enforces the limit after the first statement the user executes. In addition to enforcing the concurrent usage limits, Oracle tracks the highest number of concurrent sessions for each instance. You can use this "high water mark."

Parallel Server Concurrent Usage Limits

For instances running with the Parallel Server, each instance can have its own concurrent usage limit and warning limit. However, the sum of the instances' limits must not exceed the site's concurrent usage license.

Setting the Maximum Number of Sessions

To set the maximum number of concurrent sessions for an instance, set the parameter LICENSE_MAX_SESSIONS as follows:

```
LICENSE_MAX_SESSIONS = 80
```

If you set this limit, you are not required to set a warning limit (LICENSE_SESSIONS_ WARNING). However, using the warning limit makes the maximum limit easier to manage, because it gives you advance notice that your site is nearing maximum use.

Setting the Session Warning Limit

To set the warning limit for an instance, set the parameter LICENSE_SESSIONS_WARNING in the parameter file used to start the instance.

Set the session warning to a value lower than the concurrent usage maximum limit (LICENSE_MAX_SESSIONS).

Changing Concurrent Usage Limits While the Database is Running

To change either the maximum concurrent usage limit or the warning limit while the database is running, use the ALTER SYSTEM command with the appropriate option. The following statement changes the maximum limit to 100 concurrent sessions:

```
ALTER SYSTEM SET LICENSE_MAX_SESSIONS = 100;
The following statement changes both the warning limit and the maximum limit:
ALTER SYSTEM
   SET LICENSE_MAX_SESSIONS = 64
   LICENSE_SESSIONS_WARNING = 54;
```

If you change either limit to a value lower than the current number of sessions, the current sessions remain; however, the new limit is enforced for all future connections until the instance is shut down. To change the limit permanently, change the value of the appropriate parameter in the parameter file. To change the concurrent usage limits while the database is running, you must have the ALTER SYSTEM privilege. Also, to connect to an instance after the instance's maximum limit has been reached, you must have the RESTRICTED SESSION privilege.

Named User Limits

Named user-licensing limits the number of individuals authorized to use Oracle on the specified computer. To enforce this license, you can set a limit on the number of users created in the database before you start an instance. You can also change the maximum number of users while the instance is running, or disable the limit altogether. You cannot create more users after reaching this limit. If you try to do so, Oracle returns an error indicating that the maximum number of users have been created, and writes a message to the ALERT file.

This mechanism operates on the assumption that each person accessing the database has a unique username, and that there are no shared usernames. Do not allow multiple users to connect using the same username.

Setting User Limits

To limit the number of users created in a database, set the LICENSE_MAX_USERS parameter in the database's parameter file. The following example sets the maximum number of users to 200:

```
LICENSE_MAX_USERS = 200
```

If the database contains more than LICENSE_MAX_USERS when you start it, Oracle returns a warning and writes an appropriate message in the ALERT file. You cannot create additional users until the number of users drops below the limit or until you delete users or upgrade your Oracle license.

Changing User Limits

To change the maximum named users limit, use the ALTER SYSTEM command with the LICENSE_MAX_USERS option. The following statement changes the maximum number of defined users to 300:

```
ALTER SYSTEM SET LICENSE_MAX_USERS = 300;
```

If you try to change the limit to a value lower than the current number of users, Oracle returns an error and continues to use the old limit. If you successfully change the limit, the new limit remains in effect until you shut down the instance; to change the limit permanently, change the value of LICENSE_MAX_USERS in the parameter file. To change the maximum named users limit, you must have the ALTER SYSTEM privilege.

Viewing Licensing Limits and Current Values

You can see the current limits of all of the license settings, the current number of sessions, and the maximum number of concurrent sessions for the instance by querying the V$LICENSE data dictionary view. You can use this information to determine if you need to upgrade your Oracle license to allow more concurrent sessions or named users:

```
SELECT sessions_max s_max,
   sessions_warning s_warning,
   sessions_current s_current,
   sessions_highwater s_high,
   users_max
   FROM v$license;

S_MAX    S_WARNING   S_CURRENT    S_HIGH    USERS_MAX
----------------------------------------------------------
100      80          65          82        50
```

In addition, Oracle writes the session high water mark to the database's ALERT file when the database shuts down, so you can check for it there. To see the current number of named users defined in the database, use the following query:

```
SELECT COUNT(*) FROM dba_users;

COUNT(*)
----------
       174
```

18.3 User Authentication

This section describes aspects of authenticating users, and includes the following topics:

- Database Authentication
- External Authentication
- Enterprise Authentication

Depending on how you want user identities to be authenticated, there are three ways to define users before they are allowed to create a database session:

1. You can configure Oracle so that it performs both identification and authentication of users. This is called *database authentication*.

2. You can configure Oracle so that it performs only the identification of users (leaving authentication up to the operating system or network service). This is called *external authentication*.

3. You can configure Oracle so that it performs only the identification of users (leaving authentication up to the Oracle Security Service). This is called *enterprise authentication*.

Database Authentication

If you choose database authentication for a user, administration of the user account, password, and authentication of that user is performed entirely by Oracle. To have Oracle authenticate a user, specify a password for the user when you create or alter the user. Users can change their password at any time. Passwords are stored in an encrypted format. Each password must be made up of single-byte characters, even if your database uses a multi-byte character set.

To enhance security when using database authentication, Oracle recommends the use of password management, including account locking, password aging and expiration, password history, and password complexity verification.

The following statement creates a user who is identified and authenticated by Oracle:

```
CREATE USER scott IDENTIFIED BY tiger;
```

Advantages of Database Authentication

Following are advantages of database authentication:

- User accounts and all authentication are controlled by the database.There is no reliance on anything outside of the database.

- Oracle provides strong password management features to enhance security when using database authentication.

- It is easier to administer small user communities.

External Authentication

When you choose external authentication for a user, the user account is maintained by Oracle, but password administration and user authentication is performed by an external service. This external service can be the operating system or a network service, such as the Oracle Advanced Networking Option (ANO).

With external authentication, your database relies on the underlying operating system or network authentication service to restrict access to database accounts. A database password is not used for this type of login. If your operating system or network service permits, you can have it authenticate users. If you do so, set the parameter OS_AUTHENT_PREFIX, and use this prefix in Oracle usernames. This parameter defines a prefix that Oracle adds to the beginning of every user's operating system account name. Oracle compares the prefixed username with the Oracle usernames in the database when a user attempts to connect. For example, assume that OS_AUTHENT_PREFIX is set as follows:

```
OS_AUTHENT_PREFIX=OPS$
```

If a user with an operating system account named "TSMITH" is to connect to an Oracle database and be authenticated by the operating system, Oracle checks that there is a corresponding database user "OPS$TSMITH" and, if so, allows the user to connect. All references to a user authenticated by the operating system must include the prefix, as seen in "OPS$TSMITH".

The default value of this parameter is "OPS$" for backward compatibility with previous versions of Oracle. However, you might prefer to set the prefix value to some other string or a null string (an empty set of double quotes: ""). Using a null string eliminates the addition of any prefix to operating system account names, so that Oracle usernames exactly match operating system usernames.

After you set OS_AUTHENT_PREFIX, it should remain the same for the life of a database. If you change the prefix, any database username that includes the old prefix cannot be used to establish a connection, unless you alter the username to have it use password authentication. The following command creates a user who is identified by Oracle and authenticated by the operating system or a network service:

```
CREATE USER scott IDENTIFIED EXTERNALLY;
```

Using CREATE USER IDENTIFIED EXTERNALLY, you can create database accounts that must be authenticated via the operating system or network service and cannot be authenticated using a password.

Operating System Authentication

By default, Oracle only allows operating system authenticated logins over secure connections. Therefore, if you want the operating system to authenticate a user, by default that user cannot connect to the database over Net8. This means the user cannot connect using a multi-threaded server, since this connection uses Net8. This default restriction prevents a remote user from impersonating another operating system user over a network connection.

If you are not concerned about remote users impersonating another operating system user over a network connection, and you want to use operating system user authentication with network clients, set the parameter REMOTE_OS_AUTHENT (default is FALSE) to TRUE in the database's parameter file. Setting the initialization parameter REMOTE_OS_AUTHENT to TRUE allows the RDBMS to accept the client operating system username received over a non-secure connection and use it for account access. The change will take effect the next time you start the instance and mount the database.

Network Authentication

Network authentication is performed via the Oracle Advanced Networking Option (ANO), which may be configured to use a third party service such as Kerberos. If you are using ANO as the only external authentication service, the setting of the parameter REMOTE_OS_AUTHENT is irrelevant, since ANO only allows secure connections.

Advantages of External Authentication

Following are advantages of external authentication:

- More choices of authentication mechanism are available, such as smart cards, fingerprints, Kerberos, or the operating system.
- If you are already using some external mechanism for authentication, such as one of those listed above, there may be less administrative overhead to use that mechanism with the database as well.

Enterprise Authentication

If you choose enterprise authentication for a user, the user account is maintained by Oracle, but password administration and user authentication is performed by the Oracle Security Service (OSS). This authentication service can be shared among multiple Oracle database servers and allows user's authentication and authorization information to be managed centrally. Use the following command to create a user (known as a *global user*) who is identified by Oracle and authenticated by the Oracle Security Service:

```
CREATE USER scott IDENTIFIED GLOBALLY as '<external name>';
```

Advantages of Enterprise Authentication

Following are advantages of enterprise authentication:

- It is easier to administer large user communities with many databases.
- You can use industry standard public key certificates, giving increased opportunity for interoperability.

18.4 Oracle Users

Each Oracle database has a list of valid database users. To access a database, a user must run a database application and connect to the database instance using a valid username defined in the database. This section explains how to manage users for a database, and includes the following topics:

- Creating Users
- Altering Users
- Dropping Users

Creating Users

To create a database user, you must have the CREATE USER system privilege. When creating a new user, tablespace quotas can be specified for any tablespace in the database, even if the creator does not have a quota on a specified tablespace. Due to such privileged power, a security administrator is normally the only type of user that has the CREATE USER system privilege.

You create a user with either the Create User property sheet of Enterprise Manager/GUI, or the SQL command CREATE USER. Using either option, you can also specify the new user's default and temporary segment tablespaces, tablespace quotas, and profile.

```
CREATE USER OPS$jward
    IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE data_ts
    TEMPORARY TABLESPACE temp_ts
    QUOTA 100M ON test_ts
    QUOTA 500K ON data_ts
    PROFILE clerk;
```

Specifying a Name

Within each database a username must be unique with respect to other usernames and roles; a user and role cannot have the same name. Furthermore, each user has an associated schema. Within a schema, each schema object must have unique names.

Setting a User's Authentication

In the previous CREATE USER statement, the new user is to be authenticated using the operating system. The username includes the default prefix "OPS$." If the OS_AUTHENT_PREFIX parameter is set differently (that is, if it specifies either no prefix or some other prefix), modify the username accordingly, by omitting the prefix or substituting the correct prefix.
Alternatively, you can create a user who is authenticated using the database and a password:

```
CREATE USER jward

   IDENTIFIED BY airplane
   . . . ;
```

In this case, the connecting user must supply the correct password to the database to connect successfully.

Assigning a Default Tablespace

Each user has a default tablespace. When a user creates a schema object and specifies no tablespace to contain it, Oracle stores the object in the user's default tablespace.

The default setting for every user's default tablespace is the SYSTEM tablespace. If a user does not create objects, this default setting is fine. However, if a user creates any type of object, consider specifically setting the user's default tablespace. You can set a user's default tablespace during user creation, and change it later. Changing the user's default tablespace affects only objects created after the setting is changed. Consider the following issues when deciding which tablespace to specify:

- Set a user's default tablespace only if the user has the privileges to create objects (such as tables, views, and clusters).
- Set a user's default tablespace to a tablespace for which the user has a quota.
- If possible, set a user's default tablespace to a tablespace other than the SYSTEM tablespace to reduce contention between data dictionary objects and user objects for the same datafiles.

In the previous CREATE USER statement, JWARD's default tablespace is DATA_TS.

Assigning a Temporary Tablespace

Each user also has a temporary tablespace. When a user executes a SQL statement that requires a temporary segment, Oracle stores the segment in the user's temporary tablespace. If a user's temporary tablespace is not explicitly set, the default is the SYSTEM tablespace. However, setting each user's temporary tablespace reduces file contention among temporary segments and other types of segments. You can set a user's temporary tablespace at user creation, and change it later. In the previous CREATE USER statement, JWARD's temporary tablespace is TEMP_TS, a tablespace created explicitly to only contain temporary segments.

Assigning Tablespace Quotas

You can assign each user a tablespace quota for any tablespace. Assigning a quota does two things:

- Users with privileges to create certain types of objects can create those objects in the specified tablespace.
- Oracle limits the amount of space that can be allocated for storage of a user's objects within the specified tablespace to the amount of the quota.

By default, a user has no quota on any tablespace in the database. If the user has the privilege to create a schema object, you must assign a quota to allow the user to create objects. Minimally, assign users a quota for the default tablespace, and additional quotas for other tablespaces in which they will create objects.

You can assign a user either individual quotas for a specific amount of disk space in each tablespace or an unlimited amount of disk space in all tablespaces. Specific quotas prevent a user's objects from consuming too much space in the database. You can assign a user's tablespace quotas when you create the user, or add or change quotas later. If a new quota is less than the old one, then the following conditions hold true:

- If a user has already exceeded a new tablespace quota, the user's objects in the tablespace cannot be allocated more space until the combined space of these objects falls below the new quota.
- If a user has not exceeded a new tablespace quota, or if the space used by the user's objects in the tablespace falls under a new tablespace quota, the user's objects can be allocated space up to the new quota.

Before granting the UNLIMITED TABLESPACE system privilege, consider the consequences of doing so:

Advantage

- You can grant a user unlimited access to all tablespaces of a database with one statement.

Disadvantages

- The privilege overrides all explicit tablespace quotas for the user.

- You cannot selectively revoke tablespace access from a user with the UNLIMITED TABLESPACE privilege. You can grant access selectively only after revoking the privilege.

Setting Default Roles

You cannot set a user's default roles in the CREATE USER statement. When you first create a user, the user's default role setting is ALL, which causes all roles subsequently granted to the user to be default roles. Use the ALTER USER command to change the user's default roles.

Altering Users

Users can change their own passwords. However, to change any other option of a user's security domain, you must have the ALTER USER system privilege. Security administrators are normally the only users that have this system privilege, as it allows a modification of *any* user's security domain. This privilege includes the ability to set tablespace quotas for a user on any tablespace in the database, even if the user performing the modification does not have a quota for a specified tablespace.

You can alter a user's security settings with either the Alter User property sheet of Enterprise Manager/GUI, or the SQL command ALTER USER. Changing a user's security settings affects the user's future sessions, not current sessions. The following statement alters the security settings for user AVYRROS:

```
ALTER USER avyrros
    IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE data_ts
    TEMPORARY TABLESPACE temp_ts
    QUOTA 100M ON data_ts
    QUOTA 0 ON test_ts
    PROFILE clerk;
```

The ALTER USER statement here changes AVYRROS's security settings as follows:

- Authentication is changed to use AVYRROS's operating system account.
- AVYRROS's default and temporary tablespaces are explicitly set.
- AVYRROS is given a 100M quota for the DATA_TS tablespace.
- AVYRROS's quota on the TEST_TS is revoked.
- AVYRROS is assigned the CLERK profile.

Changing a User's Authentication Mechanism

While most non-DBA users do not use Enterprise Manager, they can still change their own passwords with the ALTER USER command, as follows:

```
ALTER USER andy
   IDENTIFIED BY swordfish;
```

Users can change their own passwords this way, without any special privileges (other than those to connect to the database). Users should be encouraged to change their passwords frequently. Users must have the ALTER USER privilege to switch between Oracle database authentication, external authentication, and enterprise authentication; usually, only DBAs should have this privilege.

Changing a User's Default Roles

A default role is one that is automatically enabled for a user when the user creates a session. You can assign a user zero or more default roles.

Dropping Users

When a user is dropped, the user and associated schema is removed from the data dictionary and all schema objects contained in the user's schema, if any, are immediately dropped. A user that is currently connected to a database cannot be dropped. To drop a connected user, you must first terminate the user's sessions using either Enterprise Manager/GUI, or the SQL command ALTER SYSTEM with the KILL SESSION clause.

To drop a user and all the user's schema objects (if any), you must have the DROP USER system privilege. Because the DROP USER system privilege is so powerful, a security administrator is typically the only type of user that has this privilege. You can drop a user from a database using either the Drop menu item of Enterprise Manager/GUI, or the SQL command DROP USER.

If the user's schema contains any schema objects, use the CASCADE option to drop the user and all associated objects and foreign keys that depend on the tables of the user successfully. If you do not specify CASCADE and the user's schema contains objects, an error message is returned and the user is not dropped. Before dropping a user whose schema contains objects, thoroughly investigate which objects the user's schema contains and the implications of dropping them before the user is dropped. Pay attention to any unknown cascading effects. For example, if you intend to drop a user who owns a table, check whether any views or procedures depend on that particular table.

```
DROP USER jones CASCADE;
```

18.5 Managing Resources with Profiles

A profile is a named set of resource limits. If resource limits are turned on, Oracle limits database usage and instance resources to whatever is defined in the user's profile. You can assign a profile to each user, and a default profile to all users who do not have specific profiles. For profiles to take effect, resource limits must be turned on for the database as a whole. This section describes aspects of profile management, and includes the following topics:

- Creating Profiles
- Assigning Profiles
- Altering Profiles
- Using Composite Limits
- Dropping Profiles
- Enabling and Disabling Resource Limits

Creating Profiles

To create a profile, you must have the CREATE PROFILE system privilege. You can create profiles using either the Create Profile property sheet of Enterprise Manager/GUI, or the SQL command CREATE PROFILE. At the same time, you can explicitly set particular resource limits. The following statement creates the profile CLERK:

```
CREATE PROFILE clerk LIMIT
    SESSIONS_PER_USER 2
    CPU_PER_SESSION unlimited
```

```
    CPU_PER_CALL 6000
    LOGICAL_READS_PER_SESSION unlimited
    LOGICAL_READS_PER_CALL 100
    IDLE_TIME 30
    CONNECT_TIME 480;
```

All unspecified resource limits for a new profile take the limit set by the DEFAULT profile. You can also specify limits for the DEFAULT profile.

Using the DEFAULT Profile

Each database has a DEFAULT profile, and its limits are used in two cases:

- If a user is not explicitly assigned a profile, then the user conforms to *all* the limits of the DEFAULT profile.
- All unspecified limits of any profile use the corresponding limit of the DEFAULT profile.

Initially, all limits of the DEFAULT profile are set to UNLIMITED. However, to prevent unlimited resource consumption by users of the DEFAULT profile, the security administrator should change the default limits using the Alter Profile dialog box of Enterprise Manager, or a typical ALTER PROFILE statement:

```
ALTER PROFILE default LIMIT
   . . . ;
```

Any user with the ALTER PROFILE system privilege can adjust the limits in the DEFAULT profile. The DEFAULT profile cannot be dropped.

Assigning Profiles

After a profile has been created, you can assign it to database users. Each user can be assigned only one profile at any given time. If a profile is assigned to a user who already has a profile, the new profile assignment overrides the previously assigned profile. Profile assignments do not affect current sessions. Profiles can be assigned only to users and not to roles or other profiles. Profiles can be assigned to users using the Assign Profile dialog box of Enterprise Manager/GUI, or the SQL commands CREATE USER or ALTER USER.

Altering Profiles

You can alter the resource limit settings of any profile using either the Alter Profile property sheet of Enterprise Manager/GUI or the SQL command ALTER PROFILE. To alter a profile, you must have the ALTER PROFILE system privilege.

Any adjusted profile limit overrides the previous setting for that profile limit. By adjusting a limit with a value of DEFAULT, the resource limit reverts to the default limit set for the database. All profiles not adjusted when altering a profile retain the previous settings. Any changes to a profile do not affect current sessions. New profile settings are used only for sessions created after a profile is modified. The following statement alters the CLERK profile:

```
ALTER PROFILE clerk LIMIT
    CPU_PER_CALL default
    LOGICAL_READS_PER_SESSION 20000;
```

Using Composite Limits

You can limit the total resource cost for a session via composite limits. In addition to setting specific resource limits explicitly for a profile, you can set a single composite limit that accounts for all resource limits in a profile. You can set a profile's composite limit using the Composite Limit checkbox of the Create Profile and Alter Profile property sheets of Enterprise Manager/GUI, or the COMPOSITE_LIMIT parameter of the SQL commands CREATE PROFILE or ALTER PROFILE. A composite limit is set via a *service unit*, which is a weighted sum of all resources used.

The following CREATE PROFILE statement is defined using the COMPOSITE_LIMIT parameter:

```
CREATE PROFILE clerk LIMIT
    COMPOSITE_LIMIT 20000
    SESSIONS_PER_USER 2
    CPU_PER_CALL 1000;
```

Notice that both explicit resource limits and a composite limit can exist concurrently for a profile. The limit that is reached first stops the activity in a session. Composite limits allow additional flexibility when limiting the use of system resources.

Determining the Value of the Composite Limit

The correct service unit setting for a composite limit depends on the total amount of resource used by an average profile user. As with each specific resource limit, historical information should be gathered to determine the normal range of composite resource usage for a typical profile user.

Setting Resource Costs

Each system has its own characteristics; some system resources may be more valuable than others. Oracle enables you to give each system resource a *cost*. Costs weight each system resource at the database level. Costs are only applied to the composite limit of a profile; costs do not apply to set individual resource limits explicitly. To set resource costs, you must have the ALTER RESOURCE system privilege. Only certain resources can be given a cost, including CPU_PER_ SESSION, LOGICAL_READS_PER_SESSION, CONNECT_TIME, and PRIVATE_SGA. Set costs for a database using the SQL command ALTER RESOURCE COST:

```
ALTER RESOURCE COST
    CPU_PER_SESSION 1
    LOGICAL_READS_PER_SESSION 50;
```

A large cost means that the resource is very expensive, while a small cost means that the resource is not expensive. By default, each resource is initially given a cost of 0. A cost of 0 means that the resource should not be considered in the composite limit (that is, it does not cost anything to use this resource). No resource can be given a cost of NULL.

Dropping Profiles

To drop a profile, you must have the DROP PROFILE system privilege. You can drop a profile using either Enterprise Manager/GUI, or the SQL command DROP PROFILE. To successfully drop a profile currently assigned to a user, use the CASCADE option. The following statement drops the profile CLERK, even though it is assigned to a user:

```
DROP PROFILE clerk CASCADE;
```

Any user currently assigned to a profile that is dropped is automatically assigned to the DEFAULT profile. The DEFAULT profile cannot be dropped. Note that when a profile is dropped, the drop does not affect currently active sessions; only sessions created after a profile is dropped abide by any modified profile assignments.

Enabling and Disabling Resource Limits

A profile can be created, assigned to users, altered, and dropped at any time by any authorized database user, but the resource limits set for a profile are enforced only when you enable resource limitation for the associated database. Resource limitation enforcement can be enabled or disabled by two different methods, as described in the next two sections. To alter the enforcement of resource limitation while the database remains open, you must have the ALTER SYSTEM system privilege.

Enabling and Disabling Resource Limits Before Startup

If a database can be temporarily shut down, resource limitation can be enabled or disabled by the RESOURCE_LIMIT initialization parameter in the database's parameter file. Valid values for the parameter are TRUE (enables enforcement) and FALSE; by default, this parameter's value is set to FALSE. Once the parameter file has been edited, the database instance must be restarted to take effect. Every time an instance is started, the new parameter value enables or disables the enforcement of resource limitation.

Enabling and Disabling Resource Limits While the Database is Open

If a database cannot be temporarily shut down or the resource limitation feature must be altered temporarily, you can enable or disable the enforcement of resource limitation using the SQL command ALTER SYSTEM. After an instance is started, an ALTER SYSTEM statement overrides the value set by the RESOURCE_LIMIT parameter. For example, the following statement enables the enforcement of resource limitation for a database:

```
ALTER SYSTEM
   SET RESOURCE_LIMIT = TRUE;
```

An ALTER SYSTEM statement does not permanently determine the enforcement of resource limitation. If the database is shut down and restarted, the enforcement of resource limits is determined by the value set for the RESOURCE_LIMIT parameter.

18.6 Listing Information About Database Users and Profiles

The data dictionary stores information about every user and profile, including the following:

- all users in a database
- each user's default tablespace for tables, clusters, and indexes
- each user's tablespace for temporary segments
- each user's space quotas, if any
- each user's assigned profile and resource limits
- the cost assigned to each applicable system resource
- each current session's memory usage

The following data dictionary views may be of interest when you work with database users and profiles:

- ALL_USERS
- USER_USERS
- DBA_USERS
- USER_TS_QUOTAS
- DBA_TS_QUOTAS
- USER_PASSWORD_LIMITS

- USER_RESOURCE_LIMITS
- DBA_PROFILES
- RESOURCE_COST
- V$SESSION
- V$SESSTAT
- V$STATNAME

18.7 Short summary

- You can configure Oracle so that it performs both identification and authentication of users. This is called *database authentication*.

- You can configure Oracle so that it performs only the identification of users (leaving authentication up to the operating system or network service). This is called *external authentication*.

- You can configure Oracle so that it performs only the identification of users (leaving authentication up to the Oracle Security Service). This is called *enterprise authentication*.

18.8 Brain storm

1. Explain Undo. How long will it remain?

2. What is the status of a Rollback Segment when you startup the Database? How to make online when Database startup?

3. What are the types of Rollback Segments?

4. Who is the owner of Rollback Segment?

5. What are the status of Rollback Segment?

6. What is the view used to know the status of pending off line

   a) V$ROLLSTART b) DBA_ROLLBACK_SEGS c)UNDO$
   d) DBA_SEGMENTS

Lecture 19

Privileges and Roles

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about identification of user privileges
- Discuss about the management of user roles
- Describe about granting user privileges and roles
- revoking user privileges and roles information

Coverage Plan

Lecture 19

19.1 Snap Shot

19.2 Identifying User Privileges

19.3 Managing User Roles

19.4 Granting User Privileges And Roles

19.5 Revoking User Privileges And Roles

19.6 Short Summary

19.7 Brain Storm

19.1 Snap Shot

This chapter explains how to control the ability to execute system operations and access to schema objects using privileges and roles. The following topics are included:

- Identifying User Privileges
- Managing User Roles
- Granting User Privileges and Roles
- Revoking User Privileges and Roles

19.2 Identifying User Privileges

This section describes Oracle user privileges, and includes the following topics:

- System Privileges
- Object Privileges

A user *privilege* is a right to execute a particular type of SQL statement, or a right to access another user's object. Oracle also provides shortcuts for grouping privileges that are commonly granted or revoked together.

System Privileges

There are over 80 distinct system privileges. Each system privilege allows a user to perform a particular database operation or class of database operations. For security reasons, system privileges do not allow users to access the data dictionary. Hence, users with ANY privileges (such as UPDATE ANY TABLE, SELECT ANY TABLE or CREATE ANY INDEX) cannot access dictionary tables and views that have not been granted to PUBLIC.

System Privilege Restrictions

The dictionary protection mechanism prevents unauthorized users from accessing dictionary objects. Access to dictionary objects is restricted to the users SYSDBA and SYSOPER. System privileges providing access to objects in other schemas do *not* give you access to dictionary objects. For example, the SELECT ANY TABLE privilege allows you to access views and tables in other schemas, but does not enable you to select dictionary objects (base tables, views, packages, and synonyms). Also, attempting to connect with the SQL*Plus command connect SYS/password results in failure. However, the following two SQL*Plus commands are valid:

connect SYS/password as SYSDBA
connect SYS/password as SYSOPER

Accessing Frequently Used Dictionary Objects

Users with explicit object privileges and the SYSDBA can access dictionary objects. If, however, you need access to dictionary objects and do not have explicit object privileges, you can be granted the following roles:

- SELECT_CATALOG_ROLE :- Enables users to SELECT all exported catalog views and tables granted to this role. Grant this role to users who must access all exported views and tables in the data dictionary.
- EXECUTE_CATALOG_ROLE :- Provides EXECUTE privilege on exported packages in the dictionary.
- DELETE_CATALOG_ROLE :- Enables users to delete records from the AUD$ table.

These roles enable database administrators to access certain objects in the dictionary while maintaining dictionary security.

Object Privileges

Each type of object has different privileges associated with it. Not all types of schema objects. Many of the schema objects not listed here (such as clusters, indexes, triggers, and database links) are controlled exclusively using system privileges. For example, to alter a cluster, a user must own the cluster or have the ALTER ANY CLUSTER system privilege.

Object Privilege Shortcut

The ALL and ALL PRIVILEGES shortcuts grant or revoke all available object privileges for a object. This shortcut is not a privilege, rather, it is a way of granting or revoking all object privileges with one word in GRANT and REVOKE statements. Note that if all object privileges are granted using the ALL shortcut, individual privileges can still be revoked.

Likewise, all individually granted privileges can be revoked using the ALL shortcut. However, if you REVOKE ALL, and revoking causes integrity constraints to be deleted (because they depend on a REFERENCES privilege that you are revoking), you must include the CASCADE CONSTRAINTS option in the REVOKE statement.

19.3 Managing User Roles

This section describes aspects of managing roles, and includes the following topics:

- Creating a Role
- Predefined Roles

A *role* groups several privileges and roles, so that they can be granted and revoked simultaneously from users. Roles can be enabled and disabled per user.

Creating a Role

You can create a role using either the SQL command CREATE ROLE, or the Create Role property sheet of Enterprise Manager. You must have the CREATE ROLE system privilege to create a role. Typically, only security administrators have this system privilege. The following statement creates the CLERK role, which is authorized by the database using the password BICENTENNIAL:

```
CREATE ROLE clerk
         IDENTIFIED BY bicentennial;
Role Names
```

You must give each role you create a unique name among existing usernames and role names of the database. Roles are not contained in the schema of any user.

Role Names in Multi-Byte Character Sets

In a database that uses a multi-byte character set, Oracle Corporation recommends that each role name contain at least one single-byte character. If a role name contains only multi-byte characters, the encrypted role name/password combination is considerably less secure.

Predefined Roles

The roles are automatically defined for Oracle databases. These roles are provided for backward compatibility to earlier versions of Oracle. You can grant and revoke privileges and roles to these predefined roles, much the way you do with any role you define.

Table 19-1: Predefined Roles

Role Name	Privileges Granted To Role
CONNECT	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	CREATE CLUSTER, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER
DBA	All system privileges WITH ADMIN OPTION
EXP_FULL_DATABASE	SELECT ANY TABLE, BACKUP ANY TABLE, INSERT, DELETE, AND UPDATE ON THE TABLES SYS.INCVID, SYS.INCFIL, AND SYS.INCEXP
IMP_FULL_DATABASE	BECOME USER
DELETE_CATALOG_ROLE	DELETE privileges on all dictionary packages for this role.
EXECUTE_CATALOG_ROLE	EXECUTE privilege on all dictionary packages for this role.
SELECT_CATALOG_ROLE	SELECT privilege on all catalog tables and views for this role.

Role Authorization

A database role can optionally require authorization when a user attempts to enable the role. Role authorization can be maintained by the database (using passwords), by the operating system, or by a network service. To alter the authorization method for a role, you must have the ALTER ANY ROLE system privilege or have been granted the role with the ADMIN OPTION.

Role Authorization by the Database

The use of a role can be protected by an associated password. If you are granted a role protected by a password, you can enable or disable the role only by supplying the proper password for the role in a SET ROLE command.

Role Authorization by the Operating System

The following statement creates a role named ACCTS_REC and requires that the operating system authorize its use:

```
CREATE ROLE role IDENTIFIED EXTERNALLY;
```

Role authentication via the operating system is useful only when the operating system must be able to dynamically link operating system privileges with applications. When a user starts an application, the operating system grants an operating system privilege to the user. The granted operating system privilege corresponds to the role associated with the application. At this point, the application can enable the application role. When the application is terminated, the previously granted operating system privilege is revoked from the user's operating system account. If a role is authorized by the operating system, you must configure information for each user at the operating system level. This operation is operating system dependent. If roles are granted by the operating system, you do not need to have the operating system authorize them also; this is redundant.

Role Authorization and Network Clients

If users connect to the database over SQL*Net, by default their roles cannot be authenticated by the operating system. This includes connections through a multi-threaded server, as this connection requires SQL*Net. This

restriction is the default because a remote user could impersonate another operating system user over a network connection.

If you are not concerned with this security risk and want to use operating system role authentication for network clients, set the parameter REMOTE_OS_ROLES in the database's parameter file to TRUE. The change will take effect the next time you start the instance and mount the database. (The parameter is FALSE by default.)

Withholding Authorization

A role can also be created without authorization. If a role is created without any protection, the role can be enabled or disabled by any grantee.

Changing a Role's Authorization

You can set and change the authorization method for a role using either the Alter Role property sheet of Enterprise Manager/GUI or the SQL command ALTER ROLE. The following statement alters the CLERK role to be authorized externally:

```
ALTER ROLE clerk
         IDENTIFIED EXTERNALLY;
```

Changing a User's Default Roles

A user's list of default roles can be set and altered using either the Alter User dialog box of Enterprise Manager or the SQL command ALTER USER.

Dropping Roles

In some cases, it may be applicable to drop a role from the database. The security domains of all users and roles granted a dropped role are immediately changed to reflect the absence of the dropped role's privileges. All indirectly granted roles of the dropped role are also removed from affected security domains. Dropping a role automatically removes the role from all users' default role lists.

Because the creation of objects is not dependent on the privileges received via a role, tables and other objects are not dropped when a role is dropped. To drop a role, you must have the DROP ANY ROLE system privilege or have been granted the role with the ADMIN OPTION. You can drop a role using either the Drop menu item of Enterprise Manager or the SQL command DROP ROLE. The following statement drops the role CLERK:

```
DROP ROLE clerk;
```

19.4 Granting User Privileges and Roles

This section describes aspects of granting privileges and roles, and includes the following topics:

- Granting System Privileges and Roles
- Granting Object Privileges and Roles
- Granting Privileges on Columns

Granting System Privileges and Roles

You can grant system privileges and roles to other roles and users using either the Grant System Privileges/Roles dialog box of Enterprise Manager or the SQL command GRANT.

To grant a system privilege or role, you must have the ADMIN OPTION for all system privileges and roles being granted. Also, any user with the GRANT ANY ROLE system privilege can grant any role in a database. The following statement grants the system privilege and the ACCTS_PAY role to the user JWARD:

```
GRANT create session, accts_pay
        TO jward;
```

The ADMIN Option

When a user creates a role, the role is automatically granted to the creator with the ADMIN OPTION. A grantee with the ADMIN option has several expanded capabilities:

- The grantee can grant or revoke the system privilege or role to or from *any* user or other role in the database. (Users cannot revoke a role from themselves.)
- The grantee can further grant the system privilege or role with the ADMIN OPTION.
- The grantee of a role can alter or drop the role.

In the following statement, the security administrator grants the NEW_DBA role to MICHAEL:

```
GRANT new_dba TO michael WITH ADMIN OPTION;
```

The user MICHAEL cannot only use all of the privileges implicit in the NEW_DBA role, but can grant, revoke, or drop the NEW_DBA role as deemed necessary. Because of these powerful capabilities, exercise caution when granting system privileges or roles with the ADMIN OPTION. Such privileges are usually reserved for a security administrator and rarely granted to other administrators or users of the system.

Granting Object Privileges and Roles

You can grant object privileges to roles and users using the Add Privilege to Role/User dialog box of Enterprise Manager or the SQL command GRANT. To grant an object privilege, you must fulfill one of the following conditions:

- You own the object specified.
- You have been granted the object privileges being granted with the GRANT OPTION.

The following statement grants the SELECT, INSERT, and DELETE object privileges for all columns of the EMP table to the users JFEE and TSMITH:

```
GRANT select, insert, delete ON emp TO jfee, tsmith;
```

To grant the INSERT object privilege for only the ENAME and JOB columns of the EMP table to the users JFEE and TSMITH, issue the following statement:

```
GRANT insert(ename, job) ON emp TO jfee, tsmith;
```

To grant all object privileges on the SALARY view to the user JFEE, use the ALL shortcut, as shown in the following example:

```
GRANT ALL ON salary TO jfee;
```

The GRANT OPTION

The user whose schema contains an object is automatically granted all associated object privileges with the GRANT OPTION. This special privilege allows the grantee several expanded privileges:

- The grantee can grant the object privilege to any user or any role in the database.
- The grantee can also grant the object privilege to other users, with or without the GRANT OPTION.
- If the grantee receives object privileges for a table with the GRANT OPTION and the grantee has the CREATE VIEW or CREATE ANY VIEW system privilege, the grantee can create views on the table and grant the corresponding privileges on the view to any user or role in the database.

The GRANT OPTION is not valid when granting an object privilege to a role. Oracle prevents the propagation of object privileges via roles so that grantees of a role cannot propagate object privileges received by means of roles.

Granting Privileges on Columns

You can grant INSERT, UPDATE, or REFERENCES privileges on individual columns in a table. Grant INSERT privilege on the ACCT_NO column of the ACCOUNTS table to SCOTT:

```
GRANT INSERT (acct_no)
         ON accounts TO scott;
```

19.5 Revoking User Privileges and Roles

This section describes aspects of revoking user privileges and roles, and includes the following topics:

- Revoking System Privileges and Roles
- Revoking Object Privileges and Roles

Revoking System Privileges and Roles

You can revoke system privileges and/or roles using either the Revoke System Privileges/Roles dialog box of Enterprise Manager or the SQL command REVOKE.

Any user with the ADMIN OPTION for a system privilege or role can revoke the privilege or role from any other database user or role The grantor does not have to be the user that originally granted the privilege or role. Also, users with the GRANT ANY ROLE can revoke *any* role. The following statement revokes the CREATE TABLE system privilege and the ACCTS_REC role from TSMITH:

```
REVOKE create table, accts_rec FROM tsmith;
```

Revoking Object Privileges and Roles

You can revoke object privileges using Enterprise Manager or the SQL command REVOKE. To revoke an object privilege, the revoker must be the original grantor of the object privilege being revoked. For example, assuming you are the original grantor, to revoke the SELECT and INSERT privileges on the EMP table from the users JFEE and TSMITH, you would issue the following statement:

```
REVOKE select, insert ON emp
         FROM jfee, tsmith;
```

The following statement revokes all privileges (which were originally granted to the role HUMAN_RESOURCE) from the table DEPT:

```
REVOKE ALL ON dept FROM human_resources;
```

Revoking Column Selective Object Privileges

Although users can grant column selective INSERT, UPDATE, and REFERENCES privileges for tables and views, they cannot selectively revoke column specific privileges with a similar REVOKE statement. Instead, the grantor must first revoke the object privilege for all columns of a table or view, and then selectively re-grant the column specific privileges that should remain. For example, assume that role HUMAN_RESOURCES has been granted the UPDATE privilege on the DEPTNO and DNAME columns of the table DEPT. To revoke the UPDATE privilege on just the DEPTNO column, you would issue the following two statements:

```
REVOKE UPDATE ON dept FROM human_resources;
GRANT UPDATE (dname) ON dept TO human_resources;
```

The REVOKE statement revokes UPDATE privilege on all columns of the DEPT table from the role HUMAN_RESOURCES. The GRANT statement re-grants UPDATE privilege on the DNAME column to the role HUMAN_RESOURCES.

Revoking the REFERENCES Object Privilege

If the grantee of the REFERENCES object privilege has used the privilege to create a foreign key constraint (that currently exists), the grantor can only revoke the privilege by specifying the CASCADE CONSTRAINTS option in the REVOKE statement:

```
REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;
```

Any foreign key constraints currently defined that use the revoked REFERENCES privilege are dropped when the CASCADE CONSTRAINTS options is specified.

Effects of Revoking Privileges

Depending on the type of privilege, there may be cascading effects when a privilege is revoked.

System Privileges

There are no cascading effects when revoking a system privilege related to DDL operations, regardless of whether the privilege was granted with or without the ADMIN OPTION. For example, assume the following:

1. The security administrator grants the CREATE TABLE system privilege to JFEE with the ADMIN OPTION.
2. JFEE creates a table.
3. JFEE grants the CREATE TABLE system privilege to TSMITH.
4. TSMITH creates a table.
5. The security administrator revokes the CREATE TABLE system privilege from JFEE.

6. JFEE's table continues to exist. TSMITH still has the table and the CREATE TABLE system privilege.

Cascading effects can be observed when revoking a system privilege related to a DML operation. For example, if SELECT ANY TABLE is granted to a user, and that user has created any procedures, all procedures contained in the user's schema must be re-authorized before they can be used again.

Object Privileges

Revoking an object privilege may have cascading effects that should be investigated before issuing a REVOKE statement.

- Object definitions that depend on a DML object privilege can be affected if the DML object privilege is revoked. For example, assume the procedure body of the TEST procedure includes a SQL statement that queries data from the EMP table. If the SELECT privilege on the EMP table is revoked from the owner of the TEST procedure, the procedure can no longer be executed successfully.

- Object definitions that require the ALTER and INDEX DDL object privileges are not affected if the ALTER or INDEX object privilege is revoked. For example, if the INDEX privilege is revoked from a user that created an index on someone else's table, the index continues to exist after the privilege is revoked.

- When a REFERENCES privilege for a table is revoked from a user, any foreign key integrity constraints defined by the user that require the dropped REFERENCES privilege are automatically dropped. For example, assume that the user JWARD is granted the REFERENCES privilege for the DEPTNO column of the DEPT table and creates a foreign key on the DEPTNO column in the EMP table that references the DEPTNO column. If the REFERENCES privilege on the DEPTNO column of the DEPT table is revoked, the foreign key constraint on the DEPTNO column of the EMP table is dropped in the same operation.

- The object privilege grants propagated using the GRANT OPTION are revoked if a grantor's object privilege is revoked. For example, assume that USER1 is granted the SELECT object privilege with the GRANT OPTION, and grants the SELECT privilege on EMP to USER2. Subsequently, the SELECT privilege is revoked from USER1. This revoke is cascaded to USER2 as well. Any objects that depended on USER1's and USER2's revoked SELECT privilege can also be affected, as described in previous bullet items.

Granting to and Revoking from the User Group PUBLIC

Privileges and roles can also be granted to and revoked from the user group PUBLIC. Because PUBLIC is accessible to every database user, all privileges and roles granted to PUBLIC are accessible to every database user.

Security administrators and database users should only grant a privilege or role to PUBLIC if every database user requires the privilege or role. This recommendation reinforces the general rule that at any given time, each database user should only have the privileges required to accomplish the group's current tasks successfully.

Revoking a privilege from PUBLIC can cause significant cascading effects. If any privilege related to a DML operation is revoked from PUBLIC (for example, SELECT ANY TABLE, UPDATE ON emp), all procedures in the database, including functions and packages, must be *reauthorized* before they can be used again. Therefore, exercise caution when granting DML-related privileges to PUBLIC.

When Do Grants and Revokes Take Effect?

Depending on what is granted or revoked, a grant or revoke takes effect at different times:

- All grants/revokes of system and object privileges to anything (users, roles, and PUBLIC) are immediately observed.

- All grants/revokes of roles to anything (users, other roles, PUBLIC) are only observed when a current user session issues a SET ROLE statement to re-enable the role after the grant/revoke, or when a new user session is created after the grant/revoke.

19.6 Short Summary

A database role can optionally require authorization when a user attempts to enable the role. Role authorization can be maintained by the database (using passwords), by the operating system, or by a network service.

A user *privilege* is a right to execute a particular type of SQL statement, or a right to access another user's object. Oracle also provides shortcuts for grouping privileges that are commonly granted or revoked together.

19.7 Brain Storm

1. DEFAULT profile is set to _____.Can't they be prevented?

   a)None, Yes b)None,No c)Unlimited, No d) Unlimited, Yes

2. Which Background Process helps to automatically resolve partly available status

   a) It cannot be reduced. It can be done only by DBA b) RECO c) PMON and SMON
   d) None of the above

3. When a mix of transactions (Small, Medium and Large) is not prevalent each Rollback Segment should be

   a) 50% of the size of the Database Largest Table
   b) Size of Databases Largest Table (100%)
   c) Only 10% of the size of the Database Largest Table
   d) Any size

4. Create Tablespace TS1 Datafile 'C:\..\..' size 1m;

   Create Rollback Segment R1 Tablespace TS1;
   Alter Rollback Segment R1 online;
   Create Table T1 (n int) tablespace TS1;
   Insert, Commit, Switch
   Drop Tablespace TS1 including contents
   What happens?

   a) Errors Occur b) Statement Processed

5. Which of the following is False

a) The Grantee can grant/ revoke the system privilege to / from any user in the Database
b) The grantee of a role can alter or drop the role
c) The Grantee can further grant the System Privilege with the ADMIN OPTION
d) The granter cannot drop his own System Privilege role.

Lecture 20

Auditing

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about the concepts of auditing
- Discuss about features of auditing
- Describe about the types of auditing
- Focusing statement, privilege, and schema object auditing

Coverage Plan

Lecture 20

20.1 Snap Shot

20.2 Introduction To Auditing

20.3 Statement Auditing

20.4 Privilege Auditing

20.5 Schema Object Auditing

20.6 Focusing Statement, Privilege, And Schema Object Auditing

20.7 Short Summary

20.8 Brain Storm

20.1 Snap Shot

This chapter discusses the auditing feature of Oracle. It includes:

- Introduction to Auditing
- Statement Auditing
- Privilege Auditing
- Schema Object Auditing
- Focusing Statement, Privilege, and Schema Object Auditing

20.2 Introduction to Auditing

Auditing is the monitoring and recording of selected user database actions. Auditing is normally used to

- investigate suspicious activity. For example, if an unauthorized user is deleting data from tables, the security administrator might decide to audit all connections to the database and all successful and unsuccessful deletions of rows from all tables in the database.

- monitor and gather data about specific database activities. For example, the database administrator can gather statistics about which tables are being updated, how many logical I/Os are performed, or how many concurrent users connect at peak times.

Auditing Features

This section outlines the features of the Oracle auditing mechanism.
Types of Auditing - Oracle supports three general types of auditing:

statement auditing	The selective auditing of SQL statements with respect to only the type of statement, not the specific schema objects on which it operates. Statement auditing options are typically broad, auditing the use of several types of related actions per option. For example, AUDIT TABLE tracks several DDL statements regardless of the table on which they are issued. You can set statement auditing to audit selected users or every user in the database.
privilege auditing	The selective auditing of the use of powerful system privileges to perform corresponding actions, such as AUDIT CREATE TABLE. Privilege auditing is more focused than statement auditing because it audits only the use of the target privilege. You can set privilege auditing to audit a selected user or every user in the database.
schema object auditing	The selective auditing of specific statements on a particular schema object, such as AUDIT SELECT ON EMP. Schema object auditing is very focused, auditing only a specific statement on a specific schema object. Schema object auditing always applies to all users of the database.

Focus of Auditing

Oracle allows audit options to be focused or broad. You can

- audit successful statement executions, unsuccessful statement executions, or both
- audit statement executions once per user session or once every time the statement is executed
- audit activities of all users or of a specific user

Audit Records and the Audit Trail

Audit records include information such as the operation that was audited, the user performing the operation, and the date and time of the operation. Audit records can be stored in either a data dictionary table, called the database audit trail, or an operating system audit trail.

The database audit trail is a single table named AUD$ in the SYS schema of each Oracle database's data dictionary. Several predefined views are provided to help you use the information in this table. The audit trail records can contain different types of information, depending on the events audited and the auditing options set. The following information is always included in each audit trail record, provided that the information is meaningful to the particular audit action:

- the user name
- the session identifier
- the terminal identifier
- the name of the schema object accessed
- the operation performed or attempted
- the completion code of the operation
- the date and time stamp
- the system privileges used (including MAC privileges for Trusted Oracle)
- the label of the user session (for Trusted Oracle only)
- the label of the schema object accessed (for Trusted Oracle only)

The operating system audit trail is encoded and not readable, but it is decoded in data dictionary files and error messages as follows:

Action Code	This describes the operation performed or attempted. The AUDIT_ACTIONS data dictionary table contains a list of these codes and their descriptions.
Privileges Used	This describes any system privileges used to perform the operation. The SYSTEM_PRIVILEGE_MAP table lists all of these codes and their descriptions.
Completion Code	This describes the result of the attempted operation. Successful operations return a value of zero; unsuccessful operations return the Oracle error code describing why the operation was unsuccessful.

Auditing Mechanisms

This section explains the mechanisms used by the Oracle auditing features.

When Are Audit Records Generated?

The recording of audit information can be enabled or disabled. This functionality allows any authorized database user to set audit options at any time but reserves control of recording audit information for the security administrator.

When auditing is enabled in the database, an audit record is generated during the execute phase of statement execution.

SQL statements inside PL/SQL program units are individually audited, as necessary, when the program unit is executed.

The generation and insertion of an audit trail record is independent of a user's transaction; therefore, if a user's transaction is rolled back, the audit trail record remains committed.

Events Always Audited to the Operating System Audit Trail

Regardless of whether database auditing is enabled, the Oracle Server always records some database-related actions into the operating system audit trail:

Instance startup	An audit record is generated that details the OS user starting the instance, the user's terminal identifier, the date and time stamp, and whether database auditing was enabled or disabled. This information is recorded into the OS audit trail because the database audit trail is not available until after startup has successfully completed. Recording the state of database auditing at startup further prevents an administrator from restarting a database with database auditing disabled so that they are able to perform unaudited actions.
Instance shutdown	An audit record is generated that details the OS user shutting down the instance, the user's terminal identifier, the date and time stamp.
Connections to the database with administrator privileges	An audit record is generated that details the OS user connecting to Oracle with administrator privileges. This provides accountability of users connected with administrator privileges.

On operating systems that do not make an audit trail accessible to Oracle, these audit trail records are placed in an Oracle audit trail file in the same directory as background process trace files.
Additional Information: See your operating system-specific Oracle documentation for more information about the operating system audit trail.
When Do Audit Options Take Effect?

Statement and privilege audit options in effect at the time a database user connects to the database remain in effect for the duration of the session. A session does not see the effects of statement or privilege audit options being set or changed. The modified statement or privilege audit options take effect only when the current session is ended and a new session is created. In contrast, changes to schema object audit options become effective for current sessions immediately.

Auditing in a Distributed Database

Auditing is site autonomous; an instance audits only the statements issued by directly connected users. A local Oracle node cannot audit actions that take place in a remote database. Because remote connections are established through the user account of a database link, the remote Oracle node audits the statements issued through the database link's connection.

Auditing to the OS Audit Trail

Both Oracle and Trusted Oracle allow audit trail records to be directed to an operating system audit trail if the operating system makes such an audit trail available to Oracle. On some other operating systems, these audit records are written to a file outside the database, with a format similar to other Oracle trace files.

Additional Information: See your platform-specific Oracle documentation to see if this feature has been implemented on your operating system.

Both Oracle and Trusted Oracle allow certain actions that are *always* audited to continue, even when the operating system audit trail (or the operating system file containing audit records) is unable to record the audit record. The usual cause of this is that the operating system audit trail or the file system is full and unable to accept new records.

System administrators configuring OS auditing should ensure that the audit trail or the file system does not fill completely. Most operating systems provide administrators with sufficient information and warning to ensure this does not occur. Note, however, that configuring auditing to use the database audit trail removes this vulnerability, because the Oracle Server prevents audited events from occurring if the audit trail is unable to accept the database audit record for the statement.

20.3 Statement Auditing

Statement auditing is the selective auditing of related groups of statements that fall into two categories:

- DDL statements, regarding a particular *type* of database structure or schema object, but not a specifically named structure or schema object (for example, AUDIT TABLE audits all CREATE and DROP TABLE statements)

- DML statements, regarding a particular *type* of database structure or schema object, but not a specifically named structure or schema object (for example, AUDIT SELECT TABLE audits all SELECT . . . FROM TABLE/VIEW/SNAPSHOT statements, regardless of the table, view, or snapshot)

Statement auditing can be broad or focused, auditing the activities of all database users or the activities of only a select list of database users.

20.4 Privilege Auditing

Privilege auditing is the selective auditing of the statements allowed using a system privilege. For example, auditing of the SELECT ANY TABLE system privilege audits users' statements that are executed using the SELECT ANY TABLE system privilege. You can audit the use of any system privilege.

In all cases of privilege auditing, owner privileges and schema object privileges are checked before system privileges. If the owner and schema object privileges suffice to permit the action, the action is not audited.

If similar statement and privilege audit options are both set, only a single audit record is generated. For example, if the statement option TABLE and the system privilege CREATE TABLE are both audited, only a single audit record is generated each time a table is created.

Privilege auditing is more focused than statement auditing because each option audits only specific types of statements, not a related list of statements. For example, the statement auditing option TABLE audits CREATE TABLE, ALTER TABLE, and DROP TABLE statements, while the privilege auditing option CREATE TABLE audits only CREATE TABLE statements, since only the CREATE TABLE statement requires the CREATE TABLE privilege.

Like statement auditing, privilege auditing can audit the activities of all database users or the activities of only a select list of database users.

20.5 Schema Object Auditing

Schema bject auditing is the selective auditing of specific DML statements (including queries) and GRANT and REVOKE statements for specific schema objects. Schema object auditing audits the operations permitted by schema object privileges, such as SELECT or DELETE statements on a given table, as well as the GRANT and REVOKE statements that control those privileges.

You can audit statements that reference tables, views, sequences, *standalone* stored procedures and functions, and packages (procedures in packages cannot be audited individually).

Statements that reference clusters, database links, indexes, or synonyms are not audited directly. However, you can audit access to these schema objects indirectly by auditing the operations that affect the base table.
Schema object audit options are always set for all users of the database; these options cannot be set for a specific list of users. You can set default schema object audit options for all auditable schema objects. For more information, see "AUDIT (Schema Objects)" in Oracle8 Server SQL Reference.

Schema Object Audit Options for Views and Procedures

Views and procedures (including stored functions, packages, and triggers) reference underlying schema objects in their definition. Therefore, auditing with respect to views and procedures has several unique characteristics. Multiple audit records can be generated as the result of using a view or a procedure: The use of the view or procedure is subject to enabled audit options; and the SQL statements issued as a result of using the view or procedure are subject to the enabled audit options of the base schema objects (including default audit options).

Consider the following series of SQL statements:

```
AUDIT SELECT ON emp;
CREATE VIEW emp_dept AS
  SELECT empno, ename, dname
    FROM emp, dept
    WHERE emp.deptno = dept.deptno;
AUDIT SELECT ON emp_dept;
SELECT * FROM emp_dept;
```

As a result of the query on EMP_DEPT, two audit records are generated: one for the query on the EMP_DEPT view and one for the query on the base table EMP (indirectly through the EMP_DEPT view). The query on the base table DEPT does not generate an audit record because the SELECT audit option for this table is not enabled. All audit records pertain to the user that queried the EMP_DEPT view.

The audit options for a view or procedure are determined when the view or procedure is first used and placed in the shared pool. These audit options remain set until the view or procedure is flushed from, and subsequently replaced in, the shared pool. Auditing a schema object invalidates that schema object in the cache and causes it to be reloaded. Any changes to the audit options of base schema objects are not observed by views and procedures in the shared pool.

Continuing with the above example, if auditing of SELECT statements is turned off for the EMP table, use of the EMP_DEPT view would no longer generate an audit record for the EMP table.

20.6 Focusing Statement, Privilege, and Schema Object Auditing

Oracle allows you to focus statement, privilege, and schema object auditing in three areas:

- successful and unsuccessful executions of the audited SQL statement
- BY SESSION and BY ACCESS auditing

- for specific users or for all users in the database (statement and privilege auditing only)

Auditing Successful and Unsuccessful Statement Executions

For statement, privilege, and schema object auditing, Oracle allows the selective auditing of successful executions of statements, unsuccessful attempts to execute statements, or both. Therefore, you can monitor actions even if the audited statements do not complete successfully.

You can audit an unsuccessful statement execution only if a valid SQL statement is issued but fails because of lack of proper authorization or because it references a nonexistent schema object. Statements that failed to execute because they simply were not valid cannot be audited. For example, an enabled privilege auditing option set to audit unsuccessful statement executions audits statements that use the target system privilege but have failed for other reasons (for example, CREATE TABLE is set, but a CREATE TABLE statement fails due to lack of quota for the specified tablespace).

Using either form of the AUDIT command, you can include

- the WHENEVER SUCCESSFUL option, to audit only successful executions of the audited statement
- the WHENEVER NOT SUCCESSFUL option, to audit only unsuccessful executions of the audited statement
- neither of the previous options, to audit both successful and unsuccessful executions of the audited statement

Auditing BY SESSION versus BY ACCESS

Most auditing options can be set to indicate how audit records should be generated if the audited statement is issued multiple times in a single user session. This section describes the distinction between the BY SESSION and BY ACCESS options of the AUDIT command.

BY SESSION

For any type of audit (schema object, statement, or privilege), BY SESSION inserts only one audit record in the audit trail, per user and schema object, during the session that includes an audited action.

A *session* is the time between when a user connects to and disconnects from an Oracle database.

Example 1: Assume the following:

- The SELECT TABLE statement auditing option is set BY SESSION.
- JWARD connects to the database and issues five SELECT statements against the table named DEPT and then disconnects from the database.
- SWILLIAMS connects to the database and issues three SELECT statements against the table EMP and then disconnects from the database.

In this case, the audit trail will contain two audit records for the eight SELECT statements (one for each *session* that issued a SELECT statement).

Example 2: Alternatively, assume the following:

- The SELECT TABLE statement auditing option is set BY SESSION.
- JWARD connects to the database and issues five SELECT statements against the table named DEPT, and three SELECT statements against the table EMP, and then disconnects from the database.

In this case, the audit trail will contain two records (one for each schema object against which the user issued a SELECT statement in a session).

Note: If you use the BY SESSION option when directing audit records to the operating system audit trail, Oracle generates and stores an audit record each time an access is made. Therefore, in this auditing configuration, BY SESSION is equivalent to BY ACCESS.

BY ACCESS

Setting audit BY ACCESS inserts one audit record into the audit trail for each execution of an auditable operation within a cursor. Events that cause cursors to be reused include the following:

- an application, such as Oracle Forms, holding a cursor open for reuse
- subsequent execution of a cursor using new bind variables
- statements executed within PL/SQL loops where the PL/SQL engine optimizes the statements to reuse a single cursor

Note that auditing is NOT affected by whether a cursor is shared; each user creates her or his own audit trail records on first execution of the cursor.

Example: Assume that

- The SELECT TABLE statement auditing option is set BY ACCESS.
- JWARD connects to the database and issues five SELECT statements against the table named DEPT and then disconnects from the database.
- SWILLIAMS connects to the database and issues three SELECT statements against the table DEPT and then disconnects from the database.

The single audit trail contains eight records for the eight SELECT statements.

Defaults and Excluded Operations

The AUDIT command allows you to specify either BY SESSION or BY ACCESS. However, several audit options can be set only BY ACCESS, including

- all statement audit options that audit DDL statements
- all privilege audit options that audit DDL statements

For all other audit options, BY SESSION is used by default.

Auditing By User

Statement and privilege audit options can audit statements issued by any user or statements issued by a specific list of users. By focusing on specific users, you can minimize the number of audit records generated. For more information see Oracle8 Server SQL Reference.

Example: To audit statements by the users SCOTT and BLAKE that query or update a table or view, issue the following statements:

```
AUDIT SELECT TABLE, UPDATE TABLE
     BY scott, blake;
```

20.7 Short Summary

Auditing is the monitoring and recording of selected user database actions. Auditing is normally used to

- investigate suspicious activity. For example, if an unauthorized user is deleting data from tables, the security administrator might decide to audit all connections to the database and all successful and unsuccessful deletions of rows from all tables in the database.

- monitor and gather data about specific database activities. For example, the database administrator can gather statistics about which tables are being updated, how many logical I/Os are performed, or how many concurrent users connect at peak times.

20.8 Brain Storm

1. What is auditing?
2. Explain the types of auditing.
3. Explain the features of auditing.

Lecture 21

Backup

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about what is backups
- Discuss about the types of failure
- Discuss about types of backups
- Describe the format of backup

Coverage Plan

Lecture 21

21.1 Snap Shot

21.2 Backups

21.3 Types Of Failure

21.4 Types Of Backups

21.5 Backup Format

21.6 Short Summary

21.6 Brain Storm

21.1 Snap Shot

This chapter introduces database concepts that are fundamental to backup and recovery, and explains why taking backups is crucial for successful database operations. The following topics are included:

- ♣ What is backup?
- ♣ Types of failure
- ♣ Types of backups
- ♣ Backup format

21.2 What is Backups?

Simply speaking, a database backup is a representative copy of data. When the original data is lost, you can use the backup to reconstruct lost information (the physical files that constitute your Oracle database). This copy includes important parts of your database, such as the control file, archive logs and datafiles-structures described later in this chapter and throughout this book. In the event of a media failure, your database backup is the key to successfully recovering your data.

Why Are Backups Important?

Imagine the magnitude of lost revenue (not to mention the degree of customer dissatisfaction!) if the production database of a catalog company, express delivery service, bank or airline suddenly became unavailable, even for just 5 or 10 minutes; or, if you lose datafiles due to media failure and cannot restore or recover them because you do not have a backup. From your enterprise's perspective, the results may be quite grim. You must restore and recover your data quickly to resume operations. The key to your success in this situation is a well-defined backup and recovery strategy.

When to Take Backups

You should tailor your backup strategy to the needs of your business. For example, if it is acceptable to lose data in the event of a disk failure, you may not need to perform frequent backups. What if your database must be available twenty-four hours a day, seven days a week? In this case, your database would have to be frequently backed up. The frequency of your backups and types of backups performed is determined in large part by the needs of your business.

21.3 Types of Failures

An unfortunate aspect of every database system is the possibility of a system or hardware failure. The most common types of failure are described below.

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. h 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.
user or application 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 delete data from a table that is still required (for example, payroll taxes). 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 deletes data, the database can be recovered to the instant in time before the data was deleted.
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.

21.4 Types of Backups

This section defines and describes the following types of backups:

- Whole Database Backups
  - Consistent Whole Database Backups
  - Inconsistent Whole Database Backups
- Tablespace Backups
- Datafile Backups
- Controlfile Backups
- Archivelog Backups

Logical Backups, also known as Exports, are described in detail in the *Oracle8 Server Utilities Guide*.

Whole Database Backups

A whole database backup contains the control file, and all database files, which belong to that database. Whole database backups are the most common type of backups performed by database administrators. Before performing whole database backups (or any other type of backup), you should first be familiar with your enterprise's backup strategy. Specifically, you should be aware of the implications of backing up when in ARCHIVELOG mode and NOARCHIVELOG mode

During a whole database backup, all datafiles and control files belonging to that database are backed up (you should not need to back up the online redo logs).

Whole database backups do not require the database to be operated in a specific archiving mode; they can be taken whether a database is operating in ARCHIVELOG or NOARCHIVELOG mode.

If the database is in ARCHIVELOG mode, you can choose to back up the database while it is open or closed. A database running in NOARCHIVELOG mode should only be backed up when it is closed by a clean shutdown (for example, a shutdown using the immediate or normal options). *A backup of a database running in NOARCHIVELOG mode and not shutdown cleanly is useless.* In such cases, the backed up files are inconsistent with respect to a point-in-time, and because the database is in NOARCHIVELOG mode, there are no logs available to make the database consistent. Recovery Manager does not allow you to back up a database that has been running in NOARCHIVELOG mode and shutdown abnormally, because the backup is not usable for recovery.

You should always back up the parameter files associated with the database, as well as the instances' password file (if the instance uses password files) whenever they are modified.

Note: In ARCHIVELOG mode, a whole database backup can be constructed using backups of datafiles taken at different times. For example, if a database is composed of seven tablespaces, and each night a different tablespace is backed up (the control file is backed up every night), after seven nights all tablespaces in the database, and the control file will have been backed up - this can be considered a whole database backup.

There are two types of whole database backups: consistent and inconsistent.

Consistent Whole Database Backups

A *consistent* whole database backup is a backup where all files (datafiles and control files) are consistent to the same point in time (for example, checkpointed at the same SCN). The only tablespaces in a consistent backup that are allowed to have older timestamps (SCNs) are read-only and offline normal tablespaces, which are still consistent with the other datafiles in the backup.

You can open the set of files resulting from a consistent whole database backup without applying redo logs because the data is already consistent; no action is required to make the data in the restored datafiles correct.
A consistent whole database backup is the only valid backup option for databases running in NOARCHIVELOG mode. The only way to take a consistent whole database backup is to shut down the database cleanly and take the backup while the database is closed.

To make a consistent database backup current (or to take it to a later point in time) you will either need to apply redo to it, or, if you are using Recovery Manager to perform your backups, you may have the option of applying a combination of incremental backups and redo. The redo is located in the archived logs, and the online logs (if you are recovering to the current time).

If a database is not shut down cleanly (for example, an instance failure occurred, or a SHUTDOWN ABORT statement was issued), the database's datafiles will most likely be inconsistent.

Warning: A backup control file created during a consistent whole database backup should only be used if you are restoring the whole database backup and do not intend to perform any recovery. If you intend to perform recovery and have a current control file, you should not restore an older control file.

Inconsistent Whole Database Backups

If you are in a situation where your database must be up and running 24 hours a day, 7 days a week, you will need to perform *inconsistent* whole database backups. A backup of an open database is inconsistent because portions of the database (hence, datafiles in the database) are being modified and written to disk while the backup is progressing. The database must be in ARCHIVELOG mode to be able to perform open backups.

Deciding whether or not to perform an open backup depends only upon the availability requirements of your data-open backups are the only choice if the data being backed up must always be available.

Inconsistent backups are also created if a database is backed up after a system crash failure or shutdown abort. This type of backup is valid if the database is running in ARCHIVELOG mode, because all logs are available to make the backup consistent.

Note: It is not good practice to take inconsistent, closed database backups.

After open (i.e. hot) or even inconsistent closed backups, you should always archive the current online log file. You archive the current online log by issuing the sql command alter system archive log all. Once the current log file has been archived, you should backup all archivelogs produced since the backup began. This is necessary to ensure you can use the backup, as without all archived logs produced during the backup,the backup cannot be recovered, as you do not have all the logs necessary to make the backup consistent.

Unless you are taking a consistent whole database backup, you should back up your control file using the ALTER DATABASE command with the BACKUP CONTROLFILE option. A database must be consistent before it can be opened. Inconsistent whole database backups are made consistent by applying incremental backups and redo logs (archived and online).

Note: Inconsistent whole database backups of databases running in NOARCHIVELOG mode are usable only if the logs relating to that backup are available. For databases running in NOARCHIVELOG mode, you should aim to have a backup that is usable without performing any recovery. This aim is defeated if you need to apply redo from logs to recover a backup. If you need redo to make a database consistent, the database should really operate in ARCHIVELOG mode, because this is the way online logs are backed up.

Tablespace Backups

A tablespace backup is a backup of a subset of the database.

Tablespace backups are only valid if the database is operating in ARCHIVELOG mode. The only time a tablespace backup is valid for a database running in NOARCHIVELOG mode is when that tablespace (and all datafiles in that tablespace), is read-only or offline-normal.

Datafile Backups

A datafile backup is a backup of a single datafile. Database administrators usually take tablespace backups rather than datafile backups, because the tablespace is a logical unit of a database to back up.

Datafile backups are only valid if the database is operating in ARCHIVELOG mode. The only time a datafile backup is valid for a database running in NOARCHIVELOG mode is if that datafile is the only file in a tablespace. For example, the backup is a tablespace backup, but the tablespace only contains one file and is read-only or offline-normal.

Control File Backups

A control file backup is a backup of a database's control file. If the database is open you can issue the following statement:

```
ALTER DATABASE BACKUP CONTROLFILE to `location';
```

You can also use Recovery Manager to back up the control file.

If you are going to take an O/S backup of the control file, you must shut down the database first.

Archivelog Backups

If a database is operating in ARCHIVELOG mode, online logs are archived to the LOG_ARCHIVE_DEST, from where they are typically backed up to tertiary storage (such as magnetic tape).
Some sites require very fast recovery times, so they may make a copies of their archivelogs to another disk filesystem, as well as to tertiary storage. Setting the parameter LOG_ARCHIVE_DUPLEX_DEST will archive a second copy of each online log to this location. The on-disk archivelog copies would typically only be kept for a short time (for example, 48 hours) before being deleted.

The frequency of archivelog backups depends on:

- the number and size of archivelogs produced
- the size of the filesystem that the logs are archived to

Online Redo Log Backups Not Recommended

The best way to back up the contents of the current online log is to archive it, then back up the archived log.

Oracle recommends you do not copy a current online log, because when you do so, and then restore that copy, the copy will appear as the end of the redo thread. However, additional redo may have been generated in the thread. If you ever attempt to execute recovery supplying the redo log copy, recovery will erroneously detect the end of the redo thread and prematurely terminate, possibly corrupting the database.

21.5 Backup Formats

Each of the backup types described in the preceding sections can be made in the following formats:

- Backup Sets
  - Datafiles
  - Archived Redo Log Files
- Datafile Copies
- Operating System Backups

Backup Sets

Backup sets are created using the Recovery Manager backup command, and can contain either archive logs or datafiles, but not both.
An Oracle server process reads the datafiles being backed up and creates the backup set. This is why backup sets of open database files (made by Recovery Manager) do *not* need to be preceded by the ALTER TABLESPACE BEGIN BACKUP statement.

Backup sets are written out in an Oracle proprietary format. Files in a backup set cannot be used by an Oracle instance until they are restored to an instance-usable (usually datafile) format by Recovery Manager. For example, a tablespace backup in a backup set is a compressed version of each file in the tablespace and the Recovery Manager command restores the datafiles from the backup set to an instance-usable format.

You can back up any logical unit of an Oracle database in a backup set:

- If you direct Recovery Manager to perform a whole database backup, it will automatically back up every file in that database, as well as the control file.
- If you direct Recovery Manager to perform a tablespace backup, it will back up each datafile in that tablespace.
- If you direct Recovery Manager to perform a datafile backup, it will back up the datafiles you specify.
- If you direct Recovery Manager to perform an ARCHIVELOG backup, it will back up the archivelogs you specify (this could be `all' archivelogs generated).

You can also direct Recovery Manager to include a control file backup in any datafile backup set.

Note: Archivelogs and datafiles cannot be combined in the same backup set.

Datafile Copies

A datafile copy is created using the Recovery Manager copy command.

An Oracle server process reads the datafile and writes the copy out to disk, not an O/S routine. This is why datafile copies of open database files (made by Recovery Manager) do *not* need to be preceded by the ALTER TABLESPACE BEGIN BACKUP statement.

Datafile copies can be used immediately by an Oracle instance--they are already in an instance-usable (usual datafile) format.

Note: Datafile copies can only be made to disk.

Operating System Backups

Operating system (O/S) backups are created using an operating system command. O/S backups can be written to disk or tape in any format that your O/S utilities support.

Recovery Manager can catalog and use O/S backups that are image backups on disk.

Logical Backups

Logical backups store information about the schema objects created for a database. You can use the Export utility to write data from an Oracle database to operating system files that have an Oracle database format.

Because the Oracle Export utility can selectively export specific objects or portions of an object (for example, partitioned tables), you might consider exporting portions or all of a database for supplemental protection and flexibility in a database's backup strategy. Database exports are not a substitute for physical backups and cannot provide the same complete recovery advantages that the built-in functionality of Oracle offers.

21.6 Short Summary

When the original data is lost, you can use the backup to reconstruct lost information (the physical files that constitute your Oracle database).

Type of failures

- statement and process failure
- instance failure
- user or application error
- media (disk) failure

21.7 Brain storm

1. Hot backup of Control file is performed after

   a) Incomplete recovery b) Structural changes to Database
   c) Large insertion of data d) None of the above

2. A Checkpoint establishes a consistent point of the database

   a) across Redo log threads b) across the Datafiles
   c) across the objects in the cache d) all of the above

3. Which two factors determine the frequency with which backup should be performed?

   a) Size of the Database b) Frequency of structural changes made to the database
   c) Type of Operating System d) Nature of the Data

Lecture 21

Recovery

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about database recovery
- Describe about errors and failure
- Type errors user error, statement failure, process failure, network failure, database instance failure and disk failure
- Structures used for database recovery
- How to take backup the database
- Redo log and rolling formats and roll back segments

Coverage Plan

Lecture 22

22.1 Snap Shot

22.2 Introduction to database recovery

22.3 Errors and failure

22.4 Structure used for database recovery

22.5 Rolling forward and rolling back

22.6 Rollback segment and rolling back

22.7 Recovery concepts and strategies

22.8 Short summary

22.9 Brain storm

22.1 Snap Shot

In this session we will concentrate about recovery, and its problems. In this session we discuss about the recovery operations and error and failure types.

22.2 Introduction to database recovery

A major responsibility of the database administrator is to prepare for the possibility of hardware, software, network, process, or system failure. If such a failure affects the operation of a database system, you must usually recover the databases and return to normal operations as quickly as possible. Recovery should protect the databases and associated users from unnecessary problems and avoid or reduce the possibility of having to duplicate work manually.

Recovery processes vary depending on the type of failure that occurred, the structures affected, and the type of recovery that you perform. If no files are lost or damaged, recovery may amount to no more than restarting an instance.

22.3 Errors and Failures

Several problems can halt the normal operation of an Oracle database or affect database I/O to disk. The following sections describe the most common types. For some of these problems, recovery is automatic and requires little or no action on the part of the database user or database administrator.

User Error

A database administrator can do little to prevent user errors (for example, accidentally dropping a table). Usually, increased training on database and application principles can reduce user error. Furthermore, by planning an effective recovery scheme ahead of time, the administrator can ease the work necessary to recover from many types of user errors.

Statement Failure

Statement failure occurs when there is a logical failure in the handling of a statement in an Oracle program. For example, assume all extents of a table (in other words, the number of extents specified in the MAXEXTENTS parameter of the CREATE TABLE statement) are allocated, and are completely filled with data; the table is absolutely full. A valid INSERT statement cannot insert a row because there is no space available. Therefore, if issued, the statement fails.

If a statement failure occurs, the Oracle software or operating system returns an error code or message. A statement failure usually requires no action or recovery steps; Oracle automatically corrects for statement failure by rolling back the effects (if any) of the statement and returning control to the application. The user can simply re-execute the statement after correcting the problem conveyed by the error message.

Process Failure

A process failure is a failure in a user, server, or background process of a database instance (for example, an abnormal disconnect or process termination). When a process failure occurs, the failed subordinate process cannot continue work, although the other processes of the database instance can continue.

The Oracle background process PMON detects aborted Oracle processes. If the aborted process is a user or server process, PMON resolves the failure by rolling back the current transaction of the aborted process and releasing any resources that this process was using. Recovery of the failed user or server process is automatic. If

the aborted process is a background process, the instance cannot continue to function correctly (usually). Therefore, you must shut down and restart the instance.

Network Failure

When your system uses networks (for example, local area networks, phone lines, and so on) to connect client workstations to database servers, or to connect several database servers to form a distributed database system, network failures (such as aborted phone connections or network communication software failures) can interrupt the normal operation of a database system. For example:

- A network failure might interrupt normal execution of a client application and cause a process failure to occur. In this case, the Oracle background process PMON detects and resolves the aborted server process for the disconnected user process, as described in the previous section.

- A network failure might interrupt the two-phase commit of a distributed transaction. Once the network problem is corrected, the Oracle background process RECO of each involved database server automatically resolves any distributed transactions not yet resolved at all nodes of the distributed database system.

Database Instance Failure

Database instance failure occurs when a problem arises that prevents an Oracle database instance (SGA and background processes) from continuing to work. An instance failure can result from a hardware problem, such as a power outage, or a software problem, such as an operating system crash. Instance failure also results when you issue a SHUTDOWN ABORT or STARTUP FORCE statement.

Recovery from Instance Failure

Instance recovery restores a database to its transaction-consistent state just before instance failure. If you experience instance failure during online backup, media recovery might be required. In all other cases, recovery from instance failure is relatively automatic. For example, when using the Oracle Parallel Server, other instances perform instance recovery. In single-instance configurations, Oracle performs instance recovery for a database when the database is restarted (mounted and opened to a new instance). The transition from a mounted state to an open state automatically triggers instance recovery, if necessary. Instance recovery consists of the following steps:

1. Rolling forward to recover data that has not been recorded in the datafiles, yet has been recorded in the online redo log, including the contents of rollback segments.

2. Opening the database. Instead of waiting for all transactions to be rolled back before making the database available, Oracle enables the database to be opened as soon as cache recovery is complete. Any data that is not locked by unrecovered transactions is immediately available. This feature is called *fast warmstart.*

3. Marking all transactions system-wide that were active at the time of failure as DEAD and marking the rollback segments containing these transactions as PARTIALLY AVAILABLE.

4. Recovering dead transactions as part of SMON recovery.

5. Resolving any pending distributed transactions undergoing a two-phase commit at the time of the instance failure.

Read-Only Tablespaces and Instance Recovery

Recovery is not needed on read-only datafiles during instance recovery. Recovery during startup verifies that each online read-only file does not need any media recovery. That is, the file was not restored from a backup

taken before it was made read-only. If you restore a read-only tablespace from a backup taken before the tablespace was made read-only, you cannot access the tablespace until you complete media recovery.

Media (Disk) Failure

An error can arise when trying to write or read a file that is required to operate an Oracle database. This occurrence is called *media failure* because there is a physical problem reading or writing to files on the storage medium.
A common example of a media failure is a disk head crash, which causes the loss of all files on a disk drive. All files associated with a database are vulnerable to a disk crash, including datafiles, redo log files, and control files.

How Media Failures Affect Database Operation

Media failures can affect one or all types of files necessary for the operation of an Oracle database, including datafiles, online redo log files, and control files.

Database operation after a media failure of online redo log files or control files depends on whether the online redo log or control file is *multiplexed*, as recommended. A multiplexed online redo log or control file simply means that a second copy of the file is maintained. If a media failure damages a single disk, and you have a multiplexed online redo log, the database can usually continue to operate without significant interruption. Damage to a non-multiplexed online redo log causes database operation to halt and may cause permanent loss of data. Damage to any control file, whether it is multiplexed or non-multiplexed, halts database operation once Oracle attempts to read or write the damaged control file.

Media failures that affect datafiles can be divided into two categories: read errors and write errors. In a read error, Oracle discovers it cannot read a datafile and an operating system error is returned to the application, along with an Oracle error indicating that the file cannot be found, cannot be opened, or cannot be read. Oracle continues to run, but the error is returned each time an unsuccessful read occurs. At the next checkpoint, a write error will occur when Oracle attempts to write the file header as part of the standard checkpoint process.

If Oracle discovers that it cannot write to a datafile and Oracle archives filled online redo log files, Oracle returns an error in the DBWR trace file, and Oracle takes the datafile offline automatically. Only the datafile that cannot be written to is taken offline; the tablespace containing that file remains online.

If the datafile that cannot be written to is in the SYSTEM tablespace, the file is not taken offline. Instead, an error is returned and Oracle shuts down the database. The reason for this exception is that all files in the SYSTEM tablespace must be online in order for Oracle to operate properly. For the same reason, the datafiles of a tablespace containing active rollback segments must remain online.

If Oracle discovers that it cannot write to a datafile, and Oracle is not archiving filled online redo log files, DBWR fails and the current instance fails. If the problem is temporary (for example, the disk controller was powered off), instance recovery usually can be performed using the online redo log files, in which case the instance can be restarted. However, if a datafile is permanently damaged and archiving is not used, the entire database must be restored using the most recent backup.

22.4 Structures Used for Database Recovery

Several structures of an Oracle database safeguard data against possible failures. The following sections briefly introduce each of these structures and its role in database recovery.

Database Backups

A database backup consists of operating system backups of the physical files that constitute an Oracle database. To begin database recovery from a media failure, Oracle uses file backups to restore damaged datafiles or control files.

The Redo Log

The redo log, present for every Oracle database, records all changes made in an Oracle database. The redo log of a database consists of at least two redo log files that are separate from the datafiles (which actually store a database's data). As part of database recovery from an instance or media failure, Oracle applies the appropriate changes in the database's redo log to the datafiles, which updates database data to the instant that the failure occurred.
A database's redo log can consist of two parts: the online redo log and the archived redo log.

The Online Redo Log

Every Oracle database has an associated online redo log. The online redo log works with the Oracle background process LGWR to immediately record all changes made through the associated instance. The online redo log consists of two or more pre-allocated files that are reused in a circular fashion to record ongoing database changes.

The Archived (Offline) Redo Log

Optionally, you can configure an Oracle database to archive files of the online redo log once they fill. The online redo log files that are archived are uniquely identified and make up the archived redo log. By archiving filled online redo log files, older redo log information is preserved for more extensive database recovery operations, while the pre-allocated online redo log files continue to be reused to store the most current database changes.

Rollback Segments

Rollback segments are used for a number of functions in the operation of an Oracle database. In general, the rollback segments of a database store the old values of data changed by ongoing transactions (that is, uncommitted transactions). Among other things, the information in a rollback segment is used during database recovery to "undo" any "uncommitted" changes applied from the redo log to the datafiles. Therefore, if database recovery is necessary, the data is in a consistent state after the rollback segments are used to remove all uncommitted data from the datafiles.

Control Files

In general, the control file(s) of a database store the status of the physical structure of the database. Certain status information in the control file (for example, the current online redo log file, the names of the datafiles, and so on) guides Oracle during instance or media recovery.

22.5 Rolling Forward and Rolling Back

Database buffers in the SGA are written to disk only when necessary, using a least-recently-used algorithm. Because of the way that DBWR uses this algorithm to write database buffers to datafiles, datafiles might contain some data blocks modified by uncommitted transactions and some data blocks missing changes from committed transactions.
Two potential problems can result if an instance failure occurs:

- Data blocks modified by a transaction might not be written to the datafiles at commit time and might only appear in the redo log. Therefore, the redo log contains changes that must be reapplied to the database during recovery.

- Since the redo log might have also contained data that was not committed, the uncommitted transaction changes applied by the redo log (as well as any uncommitted changes applied by earlier redo logs) must be erased from the database.

To solve this dilemma, two separate steps are generally used by Oracle for a successful recovery of a system failure: rolling forward with the redo log and rolling back with the rollback segments.

The Redo Log and Rolling Forward

The redo log is a set of operating system files that record all changes made to any database buffer, including data, index, and rollback segments, *whether the changes are committed or uncommitted.* The redo log protects changes made to database buffers in memory that have not been written to the datafiles.

The first step of recovery from an instance or disk failure is to *roll forward*, or reapply all of the changes recorded in the redo log to the datafiles. Because rollback data is also recorded in the redo log, rolling forward also regenerates the corresponding rollback segments.

Rolling forward proceeds through as many redo log files as necessary to bring the database forward in time. Rolling forward usually includes online redo log files and may include archived redo log files.

After roll forward, the data blocks contain all committed changes as well as any uncommitted changes that were recorded in the redo log.

22.6 Rollback Segments and Rolling Back

Rollback segments record database actions that should be undone during certain database operations. In database recovery, rollback segments undo the effects of uncommitted transactions previously applied by the rolling forward phase.

After the roll forward, any changes that were not committed must be undone. After redo log files have reapplied all changes made to the database, then the corresponding rollback segments are used. Rollback segments are used to identify and undo transactions that were never committed, yet were recorded in the redo log and applied to the database during roll forward. This process is called *rolling back.*

Oracle can roll back multiple transactions simultaneously as needed. All transactions system-wide that were active at the time of failure are marked as DEAD. Instead of waiting for SMON to roll back dead transactions, new transactions can recover blocking transactions themselves to get the row locks they need. This feature is called fast transaction rollback.

Figure 22-1 illustrates rolling forward and rolling back, the two steps necessary to recover from any type of system failure.

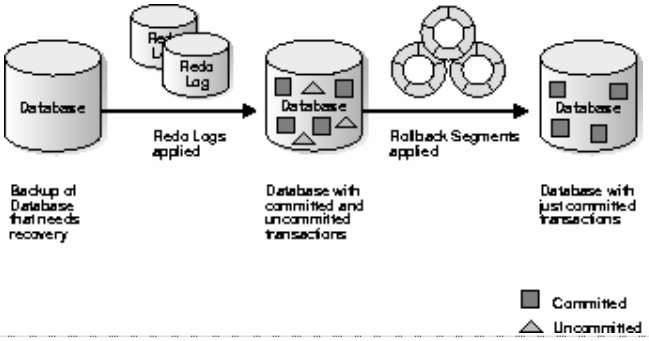


Figure 22-1: ***Basic Recovery Steps: Rolling Forward and Rolling Back***

22.7 Recovery Concepts and Strategies

Before recovering a database, familiarize yourself with the fundamental data structures, concepts and strategies of Oracle recovery. This section describes basic recovery issues, and includes the following topics:

- Important Recovery Data Structures
- Recovery Planning
- Recovery Operations

Of course, before learning about the data structures important to recovery operations, you should first understand what is meant by restoring and recovering data.
You *restore* a file by reconstructing or retrieving an original copy of it from a backup.

When you *recover* a file, you are making a restored file current, or you are making that file current to a specific point in time. The process of recovery is also known as "`rolling forward."

There are two types of recovery commands:

- SQL

The SQL RECOVER command rolls a restored file forward by applying redo (archived and online redo logs).

- Recovery Manager

A Recovery Manager **recover** command rolls a restored data file forward by applying incremental backups (if they exist) and then applying redo (archived and online redo logs).

Important Recovery Data Structures

Table 22-1 describes important data structures involved in recovery processes. Be familiar with these data structures before starting any recovery procedure.

Data Structure	Description
Control File	The control file contains records that describe and maintain information about the physical structure of a database. The control file is updated continuously during database use, and must be available for writing whenever the database is open. If the control file is not accessible, the database will not function properly.
System Change Number (SCN)	The system change number is a clock value for the Oracle database that describes a committed version of the database. The SCN functions as a sequence generator for a database, and controls concurrency and redo record ordering. Think of the SCN as a timestamp that helps ensure transaction consistency.
Rollback Segments	Information in a rollback segment is used during database recovery to undo any uncommitted changes applied from the redo log to the datafiles. After the rollback segments are used to remove all uncommitted data from the datafiles, data is in a consistent state.
Redo Records	A redo record is a group of change vectors describing a single, atomic change to the database. Redo records are constructed for all data block changes and saved on disk in the redo log. Redo records allow multiple database blocks to be changed so that either all changes occur or no changes occur, despite arbitrary failures.
Redo Logs	All changes to the Oracle database are recorded in redo logs, which consist of at least two redo log files that are separate from the datafiles. During database recovery from an instance or media failure, Oracle applies the appropriate changes in the database's redo

	log to the datafiles; this updates database data to the instant that the failure occurred.
Backup	A database backup is a copy of the physical files that constitute the Oracle database. When original data is lost, you can use the backup to reconstruct the lost information.
Incremental Backups (Recovery Manager only)	An incremental backup consists of one or more datafiles containing only those blocks that have been modified since a previous backup.
Recovery Catalog (Recovery Manager only)	A repository of information used and maintained by Recovery Manager. Contains information about the following: • datafile and archivelog backup sets and pieces • datafile copies • archived redo logs and copies of them • tablespaces and datafiles at the target database • user-created stored scripts
Checkpoint	A checkpoint is a data structure in the control file that defines a consistent point of the database across all threads of a redo log. Checkpoints are similar to SCNs, and also describe which threads exist at that SCN. Checkpoints are used by recovery to ensure that Oracle starts reading the log threads for the redo application at the correct point. For Parallel Server, each checkpoint has its own redo information.

Table 22-1: Recovery Data Structures

Recovery Planning

Before recovering a database, you should create a recovery plan or strategy. This section describes important issues to consider when defining your plan.

Factors Determining Your Recovery Strategy

Your recovery strategy will depend upon the answers to the following questions:

- Is the Database Running in ARCHIVELOG Mode?
- What Files Were Lost or Damaged?
- Has a User Accidentally Destroyed Data?
- Have You Tested Backup and Recovery Strategies?

After answering these questions, you can choose a recovery strategy that is appropriate for your particular situation.The following sections describe the types of recovery to perform in specific situations.

Is the Database Running in ARCHIVELOG Mode?

If your database is *not* running in ARCHIVELOG mode, you should restore the database from a consistent database backup. This is your only option when the database is not in ARCHIVELOG mode. The control file and all datafiles are restored from a consistent backup and the database is opened. All changes made subsequent to the backup are lost.

Note: The *only* time you can recover a database while operating in NOARCHIVELOG is when you have not already overwritten the online log files that were current at the time of the most recent backup.

What Files Were Lost or Damaged?

If one or more datafiles are lost, but the database stays open, you should perform tablespace (or datafile) recovery while the database is open. The tablespaces or datafiles are taken offline, restored from backups, recovered and placed online. No changes are lost and the database remains available during the recovery.

If an archive log or online log that is required for recovery is also lost, then you should perform point-in-time database or tablespace recovery.

If one or more datafiles and/or all control files are lost, and the database is not open, then you cannot use this procedure. All lost files are restored from backups, recovered and the database is opened. No changes are lost, but the database is unavailable during recovery. If an archive log or online log required for recovery is also lost, then you should perform point-in-time database recovery.

Has a User Accidentally Destroyed Data?

You should perform point-in-time database recovery when an archive log or online log required for recovery is lost, or to recover from user errors. All datafiles and the control file are restored from backups taken before the point-in-time, recovered to the point-in-time and the database is opened. All changes made subsequent to the point-in-time are lost. If the database remains open and no tablespaces that contain rollback segments are lost, you can also use point-in-time tablespace recovery.

Point-in-Time Tablespace Recovery

All changes made to the recovered tablespaces after the point-in-time are lost. The database, excluding the recovered tablespaces, is available during recovery.

Have You Tested Backup and Recovery Strategies?

You should test your backup and recovery strategies in a test environment before moving to a production system. You should continue to test your system regularly. That way, you can test the thoroughness of your strategies and later avoid real-life crises. Performing test recoveries regularly ensures that your archiving and backup procedures work. It also keeps you familiar with recovery procedures, so that you are less likely to make mistakes in a crisis.

Recovery Operations

Media recovery restores a database's datafiles to a point-in-time before disk failure, and includes the committed data in memory that was lost due to failure. Following is a list of media recovery operations:

- Complete Media Recovery

Complete media recovery includes the application of all necessary redo or incremental backups ever generated for the particular incarnation of the database being recovered. Complete media recovery can be performed on offline datafiles while the database is open. Complete media recovery may require an open resetlogs operation if a backup control file is included in the recovery, or a resetlogs operation creating a new control file has been performed. Types of complete media recovery are:

- Closed Database Recovery
- Open-Database, Offline-Tablespace Recovery
- Open-Database, Offline-Tablespace, Individual Datafile Recovery

- Incomplete Media Recovery

Incomplete media recovery produces a version of the database as it was at some time in the past. Incomplete media recovery must either continue on to become a complete media recovery, or be terminated by an open resetlogs operation that creates a new incarnation of the database. The database must be closed for incomplete media recovery operations. Types of incomplete media recovery are:

- Time-based Recovery

You specify the point in time to recover to

- Cancel-based Recovery

You decide to cancel during the recovery operation

- Change-based Recovery

For Recovery Manager, types of incomplete media recovery are:

- Time-based Recovery
- Change-based Recovery
- Log Sequence Recovery (replaces cancel-based recovery, above)

Recovers up to a specified log sequence number

22.8 Short Summary

A database's redo log can consist of two parts: the online redo log and the archived redo log. *Media recovery* restores a database's datafiles to a point-in-time before disk failure, and includes the committed data in memory that was lost due to failure.

22.9 Brain storm

1. What is recovery?
2. Explain the types of errors and failure.
3. How to backup database.

Lecture 23

What is Performance Tuning

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about trade-offs between response time and throughput and adjustments to relieve problems
- Discuss about critical resources and effects of excessive demand
- Priorities steps of tuning methods

Coverage Plan

Lecture 23

23.1 Snap Shot

23.2 Trade-Offs Between Response Time And Throughput

23.3 Critical Resources

23.4 Effects Of Excessive Demand

23.5 Adjustments To Relieve Problems

23.6 Steps Of Tuning Method

23.7 Short Summary

23.8 Brain Storm

23.1 Snap Shot

Performance must be built in! Performance tuning cannot be performed optimally after a system is put into production. To achieve performance targets of response time, throughput, and constraints you must tune application analysis, design, and implementation. This section introduces some fundamental concepts:

- Trade-offs Between Response Time and Throughput
- Critical Resources
- Effects of Excessive Demand
- Adjustments to Relieve Problems

23.2 Trade-offs Between Response Time and Throughput

Goals for tuning vary, depending on the needs of the application. Online transaction processing (OLTP) applications define performance in terms of throughput. These applications must process thousands or even millions of very small transactions per day. By contrast, decision support systems (DSS applications) define performance in terms of response time. Users of DSS applications make dramatically different kinds of demands on the database. One moment they may enter a query that fetches only a few records, and the next moment they may enter a massive parallel query that fetches and sorts hundreds of thousands of records from various different tables.

Response Time

Because response time equals service time plus wait time, you can increase performance in two ways:

- by reducing wait time
- by reducing service time

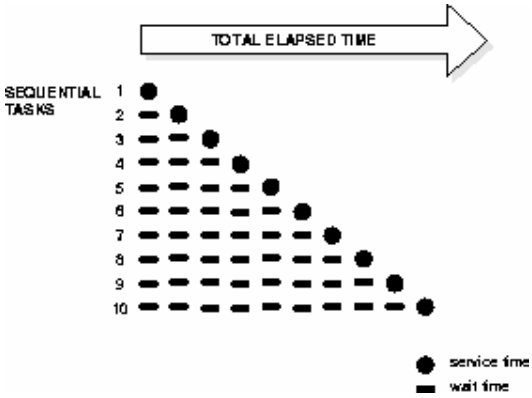


Figure 23-1: Sequential Processing of Multiple Independent Tasks

In this example only task 1 runs without having to wait. Task 2 must wait until task 1 has completed; task 3 must wait until tasks 1 and 2 have completed, and so on. (Although the figure shows the independent tasks as the same size, the size of the tasks will vary.)

System Throughput

System throughput equals the amount of work accomplished in a given amount of time. Two techniques of increasing throughput exist:

- Get more work done with the same resources (reduce service time).

- Get the work done quicker by reducing overall response time. To do this, look at the wait time. You may be able to duplicate the resource for which all the users are waiting. For example, if the system is CPU bound you can add more CPUs.

Wait Time

While the service time for a task may stay the same, wait time will go up as contention increases. If many users are waiting for a service that takes 1 second, the tenth user must wait 9 seconds for a service that takes 1 second.

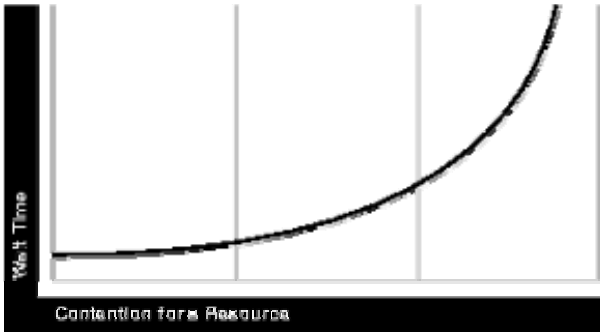


Figure 23-2: Wait Time Rising with Increased Contention for a Resource

23.3 Critical Resources

Resources such as CPUs, memory, I/O capacity, and network bandwidth are key to reducing service time. Added resources make possible higher throughput and swifter response time. Performance depends on the following:

- How many resources are available?
- How many clients need the resource?
- How long must they wait for the resource?
- How long do they hold the resource?

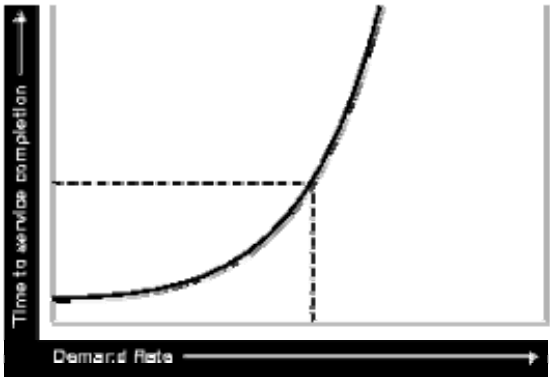


Figure 23-3: Time to Service Completion vs. Demand Rate

To manage this situation, you have two options:

- You can limit demand rate to maintain acceptable response times.
- Alternatively, you can add multiple resources: another CPU or disk.

23.4 Effects of Excessive Demand

Excessive demand gives rise to:

- greatly increased response time
- reduced throughput

If there is any possibility of demand rate exceeding achievable throughput, a demand limiter is essential.

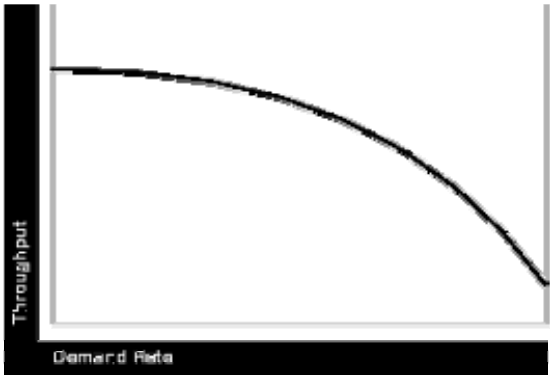


Figure 23-4: Increased Response Time/Reduced Throughput

23.5 Adjustments to Relieve Problems

Performance problems can be relieved by making the following adjustments:

adjusting unit consumption	Some problems can be relieved by using less resource per transaction or by reducing service time. Or you can take other approaches, such as reducing the number of I/Os per transaction.
Adjusting functional demand	Other problems can be abated by rescheduling or redistributing the work.
Adjusting capacity	Problems may also be relieved by increasing or reallocating resource. If you start using multiple CPUs, going from a single CPU to a symmetric multiprocessor, you will have multiple resources you can use.

For example, if your system's busiest times are from 9:00 AM until 10:30, and from 1:00 PM until 2:30, you can plan to run batch jobs in the background after 2:30, when more capacity is available. In this way you can spread out the demand more evenly. Alternatively, you can allow for delays at peak times.

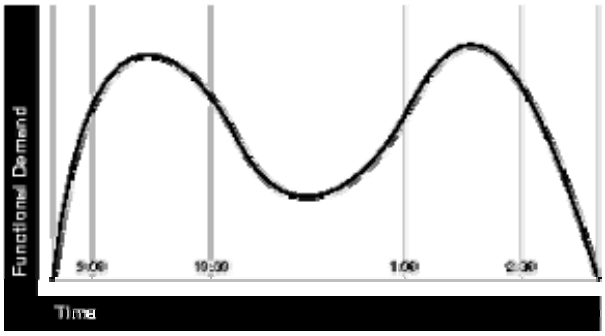


Figure 23-5: Adjusting Capacity and Functional Demand

23.6 Steps of the Tuning Method

The recommended method for tuning an Oracle database prioritizes steps in order of diminishing returns: steps, which have the greatest impact on performance, are listed first. For optimal results, therefore, tackle tuning issues in the order listed: from the design and development phases through instance tuning.

Step 1: Tune the Business Rules
Step 2: Tune the Data Design
Step 3: Tune the Application Design
Step 4: Tune the Logical Structure of the Database
Step 5: Tune the SQL
Step 6: Tune the Access Paths
Step 7: Tune Memory Allocation
Step 8: Tune I/O and Physical Structure
Step 9: Tune Resource Contention
Step 10: Tune the Underlying Platform(s)

After completing the steps of the tuning method, reassess your database performance and decide whether further tuning is necessary. Note that this is an iterative process. Performance gains made in later steps may pave the way for further improvements in earlier steps so additional passes through the tuning process may be useful.

Decisions you make in one step may influence subsequent steps. For example, in step 5 you may rewrite some of your SQL statements. These SQL statements may have significant bearing on parsing and caching issues addressed in step 7. Also, disk I/O, which is tuned in step 8, depends on the size of the buffer cache, which is tuned in step 7. Although the figure shows a loop back to step 1, you may need to loop back from any step to any previous step.

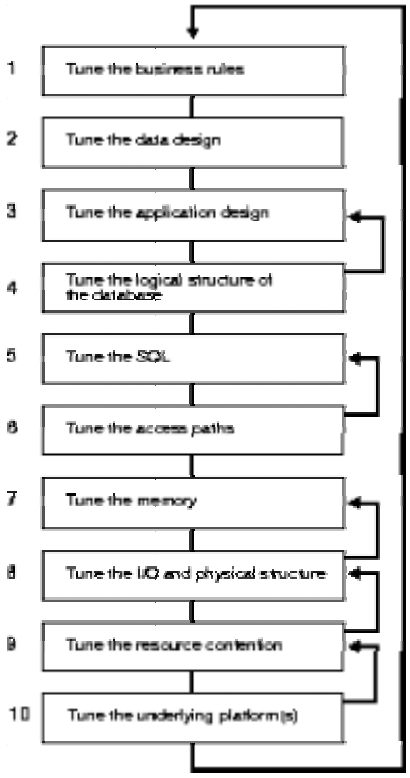


Figure 23-6: The Tuning Method

Step 1: Tune the Business Rules

For optimal performance, you may have to adapt business rules. These concern the high-level analysis and design of an entire system. Configuration issues are considered at this level, such as whether or not to use a multi-threaded server system-wide. In this way, the planners ensure that the performance requirements of the system correspond directly to concrete business needs. Performance problems encountered by the DBA may actually be caused by problems in design and implementation, or by inappropriate business rules. People commonly get in too deep when they write the business functions of an application. They document an implementation, rather than simply the function that must be performed. If business executives use care in abstracting the business function or requirement from the implementation, then designers have a wider field from which to choose the appropriate implementation.

Consider, for example, the business function of check printing. The actual requirement is to pay money to people; the requirement is not necessarily to print up pieces of paper. Whereas it would be very difficult to print up a million checks per day, it would be relatively easy to record that many direct deposit payments on a tape which could be sent to the bank for processing.

Business rules should be consistent with realistic expectations for the number of concurrent users, the transaction response time, and the number of records stored online that the system can support. For example, it would not make sense to run a highly interactive application over slow wide area network lines. Similarly, a company soliciting users for an Internet service might advertise 10 free hours per month for all new subscribers. If 50,000 users per day signed up for this service, the demand would far exceed the capacity for a client/server configuration. The company should instead consider using a multi-tier configuration. In addition, the signup process must be simple: it should require only one connection from the user to the database, or connection to

multiple databases without dedicated connections, making use of a multi-threaded server or transaction monitor approach.

Step 2: Tune the Data Design

In the data design phase, you must determine what data is needed by your applications. You need to consider what relations are important, and what their attributes are. Finally you need to structure the information to best meet performance goals.

The database design process generally undergoes a normalization stage, in which data is analyzed to ensure that no redundant data will be held anywhere. One fact should be stated in one and only one place in the database. Once the data is carefully normalized, however, you may need to denormalize it for performance reasons. You might, for example, decide that the database should hold frequently required summary values. Rather than forcing an application to recalculate the total price of all the lines in a given order each time it is accessed, you might decide to include the total value of each order in the database. You could set up primary key and foreign key indexes to access this information quickly. Another data design consideration is the avoidance of contention on data. Consider a database 1 terabyte in size, on which a thousand users access only 0.5% of the data. This "hot spot" in the data could cause performance problems.
Try also to localize access to the data--localize it to each process, to each instance, and to each partition. Contention begins when access becomes remote, and the amount of contention determines scalability. In Oracle Parallel Server, look for synchronization points--any point in time, or part of an application, which must run sequentially, one process at a time. The requirement of having sequential order numbers, for example, is a synchronization point which results from poor design.

Consider two Oracle8 enhancements that can help you to tune the data design to avoid contention:

- Consider whether or not to partition your data
- Consider whether to use local or global indexes.

Step 3: Tune the Application Design

Business executives and application designers need to translate business goals into an effective system design. Business processes concern a particular application within a system, or a particular part of an application.

An example of intelligent process design is strategically caching data. For example, in a retail application you can select the tax rate only once at the beginning of each day, and cache it within the application. In this way you avoid retrieving the same information over and over during the course of the day. At this level also, you can consider configuration of individual processes. For example, some PC users may be accessing the central system using mobile agents, whereas other users may be directly connected. Although they are running on the same system, the architecture is different. They may also require different mail servers and different versions of the application.

Step 4: Tune the Logical Structure of the Database

After the application and the system have been designed, you can plan the logical structure of the database. This primarily concerns fine-tuning the index design, to ensure that the data is neither over- or under-indexed. In the data design stage you determine the primary and foreign key indexes. In the logical structure design stage you may create additional indexes to support the application.

Performance problems due to contention often involve inserts into the same block, and using sequence numbers incorrectly. Use particular care in designing the use and location of indexes, the sequence generator, and clusters.

Step 5: Tune the SQL

System designers and application developers must understand Oracle's query processing mechanism to write effective SQL statements. Before tuning the Oracle Server itself, be certain that your application is taking full advantage of the SQL language and the Oracle features designed to speed application processing. Use features and techniques such as the following based on the needs of your application:

- array processing
- the Oracle optimizer
- the row-level lock manager
- PL/SQL

Step 6: Tune the Access Paths

Ensure that there is efficient access to data. Consider the use of clusters, hash clusters, B-tree indexes and bitmap indexes.

This may mean adding indexes, or adding indexes for a particular application and then dropping them again. Ensuring efficient access may mean revisiting your design after you have built the database. You may want to do more normalization or create alternative indexes at this point. Upon testing the application you may find that you're still not obtaining the required response time. Look for more ways to improve the design.

Step 7: Tune Memory Allocation

Appropriate allocation of memory resources to Oracle memory structures can have a large impact on performance. Oracle8 shared memory is allocated dynamically to the following structures, which are all part of the shared pool. Although you explicitly set the total amount of memory available in the shared pool, the system dynamically sets the size of each structure contained within it:

- the data dictionary cache
- the library cache
- context areas (if running multi-threaded server)

You *can* explicitly set memory allocation for the following structures:

- buffer cache
- log buffer
- sequence caches

Proper allocation of memory resources can improve cache performance, reduce parsing of SQL statements, and reduce paging and swapping.

Process local areas include:

- context areas (for systems not running multi-threaded server)
- sort areas
- hash areas

Be careful not to allocate to the SGA such a large percentage of the machine's physical memory that it causes paging or swapping.

Step 8: Tune I/O and Physical Structure

Disk I/O tends to reduce the performance of many software applications. Oracle Server, however, is designed so that its performance need not be unduly limited by I/O. Tuning I/O and physical structure involves these procedures:

- distributing data so that I/O is distributed, thus avoiding disk contention
- storing data in data blocks for best access: setting the right number of free lists, and proper values for PCTFREE and PCTUSED
- creating extents large enough for your data so as to avoid dynamic extension of tables, which would hurt high-volume OLTP applications
- evaluating the use of raw devices

Step 9: Tune Resource Contention

Concurrent processing by multiple Oracle users may create contention for Oracle resources. Contention may cause processes to wait until resources are available. Take care to reduce the following kinds of contention:

- block contention
- shared pool contention
- lock contention
- pinging (in a parallel server environment)
- latch contention

Step 10: Tune the Underlying Platform(s)

See your platform-specific Oracle documentation to investigate ways of tuning the underlying system. For example, on UNIX-based systems you might want to tune the following:

- size of the UNIX buffer cache
- logical volume managers
- memory and size for each process

23.7 Short Summary

Steps for tuning methods

Tune the Business Rules, Tune the Data Design, Tune the Application Design, Tune the Logical Structure of the Database, Tune the SQL, Tune the Access Paths, Tune Memory Allocation, Tune I/O and Physical Structure, Tune Resource Contention, Tune the Underlying Platform(s)

23.8 Brain Storm

1. To which value should you set the Oracle Blocks size when creating the database
   a) 2Ka b) 4K c) 8 K d) Maximum value allowed by the OS

2. Performance has degraded significantly on your system and you discover that paging and swaping is occuring. What is the probable cause of this problem?
   a) the SGA too small b) the PGA too small c) the SGA is too large
   d) the PGA is too small

3. A Plan_Table is created by runing
   a) UTLBSTAT.SQL b) UTLXPLAN.SQL c) UTLESTAT.SQL
   d) CATALAOG.SQL]

4. Truncate Table Plan is a valid SQL statement
   a) True b) False

5. Set Autotrace on
   a) within an application b) in a PL/SQL Proceedure c) within SQL Plus
   d) at every startup of Database

Lecture 24

Sources of Data for Tuning

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about data sources for tuning
- Discuss about oracle enterprise manager application
- Describe about oracle performance manager and oracle top sessions and trace
- Managing oracle Tablespace manager and experts

Coverage Plan

Lecture 24

24.1 Snap Shot

24.2 Sources Of Data For Tuning

24.3 Oracle Enterprise Manager Application

24.4 Introduction To Oracle Enterprise Manager

24.5 Oracle Performance Manager

24.6 Oracle Topsessions

24.7 Oracle Trace

24.8 Oracle Tablespace Manager

24.9 Oracle Expert

24.10 Short Summary

24.11 Brain Storm

24.1 Snap Shot

This section describes the various sources of data for tuning. Note that many of these sources may be transient. They include:

- Data Volumes, Online Data Dictionary, Operating System Tools, Dynamic Performance Tables, SQL Trace Facility, Alert Log, Application Program Output, Users, Initialization Parameter Files, Program Text, Design (Analysis) Dictionary, Comparative Data

24.2 sources of data for tuning

Data Volumes

The tuning data source most often overlooked is the data itself. The data may contain information that can tell you how many transactions were performed, at what time. The number of rows added to an audit table, for example, can be the best measure of the amount of useful work done (the throughput). Where such rows contain a time stamp, you can query the table and use a graphics package to plot the throughput against date and time. Such a date-time stamp need not be apparent to the rest of the application.

On the other hand, if your application does *not* contain an audit table, you might not want to add one: it would delay performance. Consider the trade-off between the value of obtaining the information and the performance cost of doing so.

Online Data Dictionary

The Oracle online data dictionary is a rich source of tuning data when used with the SQL statement ANALYZES object-type. This stores cluster, table, column and index statistics within the dictionary, primarily for use by the cost based optimizer. The dictionary also defines the indexes, which are available to help (or possibly hinder) performance.

Operating System Tools

Tools, which gather data at the operating system level, are primarily useful for determining scalability, but should also be consulted at an early stage in any tuning activity. In this way you can ensure that no part of the hardware platform is saturated (operating at or close to its maximum capacity). Network monitors are also required in distributed systems, primarily to check that no network resource is over committed. It is also worth using a simple mechanism such as the UNIX command ping to establish message turnaround time.

Dynamic Performance Tables

A number of V$ dynamic performance views are available to help you tune your system, and investigate performance problems. They allow users access to memory structures within the SGA.

SQL Trace Facility

SQL trace files record the SQL statements issued by a connected process and the resources used by these statements. In general, the virtual tables are used to tune the instance, and SQL trace file output is used to tune the applications.

Alert Log

Whenever something unexpected happens in an Oracle environment, it is worth checking the alert log to see if there is an entry at or around the time of the event.

Application Program Output

In some projects, all application processes (client-side) are instructed to record their own resource consumption to an audit trail. Where database calls are being made through a library, the response time of the client/server mechanism can be quite inexpensively recorded at the per-call level using an audit trail mechanism. Even without these levels of sophistication (which are not expensive in terms of either build or run), simply preserving the resource usages reported by a batch queue manager provides an excellent source of data for use in tuning.

Users

Users normally provide a stream of information as they encounter performance problems.

Initialization Parameter Files

It is vital to have accurate data on exactly what the system was instructed to do and how it was to go about doing it. Some of this data is available from the Oracle parameter file(s).

Program Text

Data on what the application was to do is also available from the code of the programs or procedures where both the program logic and the SQL statements reside. Server-side code (stored procedures, constraints and triggers) can be considered part of the same data population as client-side code, in this context. Tuners frequently become involved at sites where the program source code is not available, either as a result of a temporary problem or because the application is a package for which the source code is not released. In such cases it is still important for the tuner to acquire program-to-object cross reference information. For this reason executable code is a legitimate data source. Fortunately, SQL is held in text even in executable programs.

Design (Analysis) Dictionary

A design or analysis dictionary can also be used to track the intended action and resource usage of the application system. Only where the application has been entirely produced by code generators, however, can the design dictionary provide all of the data which would otherwise have to be extracted from the programs and procedures themselves.

Comparative Data

Comparative data is invaluable in most tuning situations. Tuning is often conducted from a cold start at each site; the tuners arrive with whatever expertise and experience they may have, plus a few tools for extracting the data. Experienced tuners may recognize similarities in particular situations, and try to apply the same solution as worked before. Normally, however, diagnoses such as these are purely subjective.

Tuning is much easier where a baseline exists, either from a capacity study performed for this application or (even better) data from this or another site running the same application with acceptable performance. The task is then to identify all differences between the two environments and attempt to bring them back into line. Where no directly relevant data can be found, you can check data from similar platforms and similar applications to see if they have the same performance profile. There is no point in trying to tune out a certain effect if it turns out to be ubiquitous!

Dynamic Performance Views

A primary tool for monitoring the performance of Oracle is the collection of dynamic performance views that Oracle provides to monitor your system. These views have names beginning with "V$", and this manual demonstrates their use in performance tuning. The database user SYS owns these views, and administrators can grant any database user access to them. Only some of these views are relevant to tuning your system.

Oracle and SNMP Support

The Simple Network Management Protocol (SNMP) enables users to write their own tools and applications. It is acknowledged as the standard, open protocol for heterogeneous management applications. Oracle SNMP support enables Oracle databases to be discovered on the network, identified, and monitored by any SNMP-based management application. Oracle supports several database Management Information Bases (MIBs): the standard MIB for any database management system (independent of vendor), and Oracle-specific MIBs which contains Oracle-specific information. Some statistics mentioned in this manual are supported by these MIBs, and others are not. If a statistic mentioned in this manual can be obtained through SNMP, this fact is noted.

EXPLAIN PLAN

EXPLAIN PLAN is a SQL statement that lists the access path determined by the query optimizer. Each plan has a row with ID = 0, which gives the statement type. EXPLAIN PLAN results should be interpreted with some discretion. Just because a plan does not seem efficient on the surface does not necessarily mean that the statement will run slowly. Choose statements for tuning based upon their actual resource consumption, not upon a subjective view of their execution plan.

The SQL Trace Facility and TKPROF

The SQL trace facility can be enabled for any session. It records in an operating system text file the resource consumption of every parse, execute, fetch, commit, or rollback request made to the server by the session. TKPROF summarizes the trace files produced by the SQL trace facility, optionally including the EXPLAIN PLAN output. Since the program reports each statement executed with the resources which it has consumed, the number of times it was called and the number of rows which it processed, it is quite easy to locate those statements which are using the greatest resource. It is also possible, with experience or with baselines available, to gauge whether the resources used are reasonable, given the work accomplished.

Supported Scripts

Many PL/SQL packages are supplied, thus a good number of SQL*Plus scripts that support instance tuning are supplied with the Oracle Server. Examples include UTLBSTAT.SQL and UTLESTAT; SQLUTLCHAIN.SQL, UTLDTREE.SQL, and UTLLOCKT.SQL. These statistical scripts support instance management, allowing a history to be built up over time. They can be used for the following purposes:

- to remove the need to issue DDL each time statistics are gathered
- to separate data gathering from reporting, and to allow a range of observations to be taken at intervals during a period of representative system operation, and then to allow the statistics to be reported from any start point to any end point
- to report a number of indicative ratios you can check to see if the instance is adequately tuned
- to present the LRU statistics from the buffer cache in a usable form

Application Registration

You can register with the database the name of an application and actions performed by that application. Registering the application allows system administrators and tuners to track performance by module. System administrators can also use this information to track resource usage by module. When an application registers with the database, its name and actions are recorded in the V$SESSION and V$SQLAREA views.

24.3 Oracle Enterprise Manager Applications

This section describes Oracle Enterprise Manager, and several of the most useful diagnostic and tuning tools which it provides. It covers:

- Introduction to Oracle Enterprise Manager
- Oracle Performance Manager
- Oracle TopSessions
- Oracle Trace
- Oracle Tablespace Manager
- Oracle Expert

24.4 Introduction to Oracle Enterprise Manager

Oracle Enterprise Manager is a major new infrastructure and tool set for managing Oracle environments. Oracle Enterprise Manager can be used to manage the wide range of Oracle implementations: departmental to enterprise, replication configurations, web-servers, media servers, and so forth. Oracle Enterprise Manager consists of:

- windows-based console for central administration and monitoring of Oracle databases
- common services for event management, service discovery, security, and job creation/execution
- server-side intelligent agent for monitoring events, running jobs, and communicating with the console
- applications for administering Oracle databases for security, storage, backup, recovery, import, and software distribution
- layered applications for managing Replication, Media, Web, Text, Mobile Agents, and other Oracle servers
- optional products for Oracle monitoring and tuning, known as the Oracle Performance Pack

The Oracle Enterprise Manager Performance Pack is a set of windows-based applications built on the new Oracle Enterprise Manager systems management technology. These applications address many Oracle performance management areas such as graphical monitoring, analysis, and automated tuning of Oracle databases.

24.5 Oracle Performance Manager

The Oracle Performance Manager is designed to capture, compute, and present performance data that will allow the user to monitor key metrics required for the effective use of memory, minimizing disk I/O and avoiding resource contention. It provides a graphical, real-time view of Oracle performance metrics, including the ability to drill down into a monitoring view for quick access to detailed data for performance problem solving. Oracle dynamic performance data is captured and displayed in real-time mode, and can be recorded for replay. The graphical monitor is customizable and extensible. Users can choose to display monitored information in a variety of two or three dimensional graphical views, such as tables, line, bar, cube and pie charts, and can customize the rate of monitoring. Users can also extend the system by defining charts for their own monitored sources, be they additional database performance data or application statistics. The Oracle Performance Manager tracks real-time memory performance in several ways, providing data that can be immediately put to use for memory performance management. For example, the Parse Ratio Chart gives the DBA a measure of the application's success at finding available parsed SQL in the database's library cache buffer; potentially indicating that Shared Pool memory allocation is insufficient. Monitor Charts can also be linked together, allowing the user to drill down in a logical progression of analysis. For example, if the DBA detects a performance problem with the Library Cache Hit Ratio, she can drill-down to the Library Cache Details Chart. Other memory monitoring charts include: Data Dictionary Cache Hit Ratio, Memory Allocated, and Sort Hit Ratio, to name a few.

User-defined charts can be created through the Oracle Performance Manager for virtually any data in your database, whether this data is database performance related or data from your business application tables that you want to chart. Oracle Monitor provides dialog boxes for entering the SQL to retrieve the data, for defining operations to be performed on the data, and for selecting the type of chart best suited to graphically display the data. The ability for the user to define his own charts can be combined with the power of Oracle Trace to create custom charts for monitoring application performance, application audit trails, or business transaction data. Using Oracle Trace in this way will be discussed in more detail later. Performance problems detected by using the Oracle Monitor can be corrected by using other Oracle Enterprise Manager applications. For example; memory management problems that might be corrected by modifying buffer sizes can be easily done using the Oracle Instance Manager application to reset buffer size parameters. Likewise, the DBA could address I/O or contention problems by using the Oracle Storage Manager application to reset storage parameters, or the Oracle Tablespace Manager application to further analyze the problem and defragment tables if necessary.

In addition, a DBA detecting performance problems through the Oracle Monitor can obtain a far greater degree of detail through two other Performance Pack applications: Oracle TopSessions and Oracle Trace. Ultimately, the DBA can elect to have a detailed tuning analysis conducted by the Oracle Expert automated performance tuning application. Oracle Expert produces recommendations and scripts for improving the performance of the database being monitored.

24.6 Oracle TopSessions

Often a DBA needs more information than provided by general database monitoring. For example, a DBA using the Oracle Performance Manager may detect a file I/O problem. In order to solve the problem quickly, it would be helpful to know which particular user sessions are causing the greatest I/O activity. The Oracle TopSessions application provides the DBA with a focused view of performance activity for the top *n* Oracle sessions at any given time. Oracle TopSessions extracts and analyzes sample Oracle dynamic performance data, automatically determining the top Oracle user sessions based on a specific selection criteria, such as file I/O activity. Using Oracle TopSessions, the DBA can quickly detect which user sessions are causing the greatest file I/O activity and require further investigation.

Oracle TopSessions provides two views of session data: an Overview of a select number of top sessions, and a Session Details view. The application starts with an overview of the top ten sessions connected to the database instance, with an initial default sort based on session PGA memory usage. The data displayed in the initial overview includes items such as: session ID, node, application, username, last session command executed, and the status of the session (idle, active, blocked or killed). The user can then customize the display by changing the number of sessions to be monitored, and can select the type of statistical filtering and sorting to be done for the Overview display of monitored sessions.

The Session Details display allows the user to drill down into a particular session, providing pages for detailed displays of general session information, session statistics, cursors, and locks. The Session Details General Page expands the information provided in Overview display, adding information such as identifiers for the schema, SQL, deadfalls, rows, and blocks as applicable. The Statistics Page displays detailed performance statistics for the session that are captured from the V$SESSTAT view. The Cursors page provides information on all shared cursors for the session, including SQL Statements and Explain Plans. The user has the option of displaying the session's currently executing SQL statements, or displaying all SQL statements that have and will be executed for the session. The Session Details Locks Page displays information about the database locks held or requested by session. A DBA monitoring multiple instances can open as many Oracle TopSessions displays as necessary. The information displayed in Oracle TopSessions is static until refreshed. Oracle TopSessions allows the user to determine if the refresh should be manual or automatic, and the frequency of automatic refresh.

24.7 Oracle Trace

Most data used in performance monitoring applications is collected based on sampling methodologies. For example, the Oracle Performance Manager and Oracle Top Sessions applications use this technique by periodically collecting data from the Oracle dynamic performance views.

The Oracle Trace product provides a new data collection methodology that goes a significant step further than sampling techniques. Oracle Trace collects performance data for each and every occurrence of key events in an application being monitored. It provides an entire census of performance data, rather than a sample of data, for a software application or database event. This allows performance problems detected through sampling techniques to be pinpointed to specific occurrences of a software product's execution. Oracle Trace will collect performance data for pre-defined events in products such as the Oracle Server, SQL*Net, and any other Oracle or third party application that has been programmed with Oracle Trace data collection API. An Oracle Trace "event" is an occurrence within the software product containing the Oracle Trace API calls. For example, specific events have been identified in the Oracle Server, such as a SQL parse, execute, and fetch. These events have been delimited with API calls, which are invoked when the event occurs during a scheduled Oracle Trace collection for the Oracle Server. Another example of an "event" to be monitored for performance data would be a transaction in an application, such as a "deposit" in a banking application. Any product can be programmed with Oracle Trace API calls for event-based data collection.

The type of performance data collected for events includes extensive resource utilization data, such as CPU time, memory usage and page faults, as well as performance data specific for the product being monitored. For example, user and form identification data would likely be collected for business application events, in addition to resource utilization data for those events. In addition, Oracle Trace provides the unique capability to correlate the performance data collected across any end-to-end client/server application containing Oracle Trace instrumented products. Performance can be tracked across multiple products involved in the enterprise transaction, allowing the application developer, DBA, or systems manager to easily identify the source of performance problems.

From a DBA's perspective, the value of Oracle Trace is embodied in the products that use Oracle Trace data for analysis and performance management. DBAs and other users do not have to instrument an application in order to use Oracle Trace, rather, the majority of users will employ Oracle Trace to collect data for a product that already contains the API calls, and will likely use the collected data in some other tool that performs monitoring or analysis. For example, Oracle Server and Net3 contain Oracle Trace API calls for event data collection. An Oracle Trace user will be able to schedule a collection of Oracle Trace data for either of these products, format the data and review it in reports. In addition, Oracle Trace data for Oracle Server can be imported into the Oracle Expert database tuning application, where it will be automatically analyzed for Oracle server tuning.

24.8 Oracle Tablespace Manager

A DBA who suspects database performance problems due to tablespace disorganization can use the Oracle Tablespace Manager to investigate and correct structure problems. The Oracle Tablespace Manager consists of two major features: a Tablespace Viewer and a Tablespace defragmentation function.

The Tablespace Viewer provides the administrator with a complete picture of the characteristics of all tablespaces associated with a particular Oracle instance, including: tablespace datafiles and segments, total data blocks, free data blocks and percentage of free blocks available in the tablespace's current storage allocation. The DBA has the option of displaying all segments for a tablespace or all segments for a datafile. The Tablespace Viewer also provides a map of the organization of a tablespace's segments. This map graphically displays the sequential allocation of space for segment extents within a selected tablespace or datafile. For example, a table

segment may consist of three extents, all of which are physically separated by other segment extents. The map will highlight the locations of the three extents within the tablespace or datafile. It will also show the amount of free space available for each segment. In this way, the Tablespace Viewer map provides the DBA with an easy birds-eye view of tablespace fragmentation.

When tablespace fragmentation is detected, the DBA can use the Oracle Tablespace Manager defragmentation feature to automatically correct the problem. The DBA can select a table for defragmentation from the list of table segments. The defragmentation process uses the Oracle export/import functions on the target table, and ensures that all rows, indexes, constraints and grants will remain intact. Before the table export occurs, the DBA is presented with a dialog box for modifying the storage parameters for the selected table if desired. The new parameters will then be used in the re-creation of the defragmented table. The DBA also has the option of compressing the table's extents into one large initial extent. In addition to managing fragmentation, a DBA must watch for opportunities to more effectively use available database resources. A database incurring lots of updates and deletes will develop empty data blocks; pockets of free space that are too small for new extents. The Tablespace Manager Viewer allows the DBA to visually identify free blocks. If these free blocks are adjacent, they can be automatically joined using the Oracle Tablespace Manager's coalesce feature. In this way they will become more useful space for future extents.

24.9 Oracle Expert

The Oracle Expert application provides automated performance tuning. Performance problems detected by the Oracle Performance Manager, Oracle TopSessions, and Oracle Trace can be analyzed and solved with Oracle Expert. Oracle Expert automates the process of collecting and analyzing data, and contains rules that provide database tuning recommendations, implementation scripts, and reports. Oracle Expert monitors several factors in the database environment and provides tuning recommendations in three major tuning categories:

- access method tuning
- instance parameter tuning
- database structure sizing and placement

Users can select a single tuning category for focused tuning or multiple categories for more comprehensive tuning. Tuning can also be focused on a specific portion of the database, such as a table or index. Users are able to graphically view and edit the data collected by Oracle Expert, including: database workload, systems environment, database schema, instance parameters and statistics, tablespace data, and so forth. Users can also modify Oracle Expert's rule values, for example, increasing or decreasing a rule's decision threshold. This powerful feature allows the user to play a part in the analysis and recommendations produced by Oracle Expert. Users can employ this capability to customize the collected data in order to test various tuning scenarios and to influence the final results.

After Oracle Expert has analyzed the data, the user can review the recommendations, including selectively viewing the detailed analysis. The user can choose to accept specific recommendations before generating the recommendation implementation files, which generally consist of new instance parameter files and implementation scripts. The user has full control over the implementation process. The implementation files can be invoked when the user is ready, and will automatically implement the changes the user has accepted. Oracle Expert also produces a series of reports to document the data and analysis behind the recommendations. These reports provide extensive documentation for the database and tuning process. For the less experienced DBA, they can be a valuable education in the factors that drive database performance.

24.10 Short Summary

In summary, Oracle Expert, along with the other Performance Pack applications, provides the Oracle DBA with a useful set of tools for monitoring and tuning Oracle databases.

24.11 Brain Storm

1. Which View can you query to determine if contention for the free list I high ?
   a) V$SESSION b) V$WAITSTAT c) V$RESOURCE
   d) V$LOADSTAT

2. What is increased when the Database contains migrated rows ?
   a) PCTUSED b) I/O c) SHARED_POOL_SIZE d) PCTFREE

3. You issue the command
   Analyze table Inventory.item COMPUTE STATISTICS;
   Which column in the DBA_TABLES view can you query to see the number of
   Migrated Rows in the Inventory.Item table
   a) Blocks b) Chain_cut c) AVG_ROW_LEN d) NUM_ROWS

4. Which statement type can you use reset the High water Mark on a table :
   a) Delete b) Truncate c) Update d) Insert

5. In which part of the Shared Pool are the Shared SQL and PL/SQL stored :
   a) Dictionary Cache b) Database Buffer Cache c) Library Cache
   d) User Global Area

Lecture 25

Optimization Modes and Hints

Objectives

After completing this lesson, you should be able to do the following:

- Discuss about using cost-based optimization and rule- based optimization
- Discuss about when to use the cost-based approach
- Describe about choosing a goal for the cost-based approach and parameters
- Discuss about hints and how to specify hints and its optimization approaches and goals

Coverage Plan

Lecture 25

25.1 Snap Shot

25.2 Using Cost-Based Optimization

25.3 Choosing A Goal For The Cost-Based Approach

25.4 Parameters Which Affect Cost-Based Optimization Plans

25.5 Using Rule-Based Optimization

25.6 Introduction To Hints

25.7 How To Specify Hints

25.8 Hints For Optimization Approaches And Goals

25.9 Short Summary

25.10 Brain Storm

25.1 Snap Shot

This chapter explains when to use the available optimization modes and how to use hints to enhance Oracle performance. Topics include:

- Using Cost-Based Optimization
- Using Rule-Based Optimization
- Introduction to Hints
- How to Specify Hints
- Hints for Optimization Approaches and Goals

25.2 Using Cost-Based Optimization

When to Use the Cost-Based Approach

Attention: In general, you should use the cost-based optimization approach. The rule-based approach is available for the benefit of existing applications, but all new optimizer functionality uses the cost-based approach.

The following features are *only* available with cost-based optimization; you *must* analyze your tables to get good plans:

- partitioned tables
- index-only tables
- reverse indexes
- parallel query
- star transformation
- star join

The cost-based approach generally chooses an execution plan that is as good as or better than the plan chosen by the rule-based approach, especially for large queries with multiple joins or multiple indexes. The cost-based approach also improves productivity by eliminating the need for you to tune your SQL statements yourself. Finally, many Oracle performance features are available only through the cost-based approach.

Cost based optimization must be used for efficient star query performance. Similarly, it must be used with hash joins and histograms. Cost-based optimization is always used with parallel query and with partitioned tables. You must perform ANALYZE in order to keep the statistics current.

25.3 Choosing a Goal for the Cost-Based Approach

The execution plan produced by the optimizer can vary depending upon the optimizer's goal. Optimizing for best throughput is more likely to result in a full table scan rather than an indexed scan, or a sort-merge join rather than a nested loops join. Optimizing for best response time is more likely to result in an index scan or a nested loops join.

For example, consider a join statement that can be executed with either a nested loops operation or a sort-merge operation. The sort-merge operation may return the entire query result faster, while the nested loops operation may return the first row faster. If the goal is best throughput, the optimizer is more likely to choose a sort-merge

join. If the goal is best response time, the optimizer is more likely to choose a nested loops join. Choose a goal for the optimizer based on the needs of your application:

- For applications performed in batch, such as Oracle Reports applications, optimize for best throughput. Throughput is usually more important in batch applications because the user initiating the application is only concerned with the time necessary for the application to complete. Response time is less important because the user does not examine the results of individual statements while the application is running.

- For interactive applications, such as Oracle Forms applications or SQL*Plus queries, optimize for best response time. Response time is usually important in interactive applications because the interactive user is waiting to see the first row accessed by the statement.

- For queries that use ROWNUM to limit the number of rows, optimize for best response time. Because of the semantics of ROWNUM queries, optimizing for response time provides the best results.

By default, the cost-based approach optimizes for best throughput. You can change the goal of the cost-based approach in these ways:

- To change the goal of the cost-based approach for all SQL statements in your session, issue an ALTER SESSION statement with the OPTIMIZER_GOAL option.

- To specify the goal of the cost-based approach for an individual SQL statement, use the ALL_ROWS or FIRST_ROWS hint.

Example: This statement changes the goal of the cost-based approach for your session to best response time:

```
ALTER SESSION SET OPTIMIZER_GOAL = FIRST_ROWS;
```

25.4 Parameters Which Affect Cost-Based Optimization Plans

The following initialization parameters affect cost-based optimization plans:

OPTIMIZER_GOAL	dynamically changeable parameter you can use to modify the goal of the cost-based optimization approach for your session. Used only in the session; not used in initialization file.
OPTIMIZER_MODE	sets the mode of the optimizer at instance startup: rule-based, cost based optimized for throughput or response time, or a choice based on presence of statistics. Used in initialization file only; use OPTIMIZER_MODE to change the value during a session.
OPTIMIZER_PERCENT_PARALLEL	defines the amount of parallelism that the optimizer uses in its cost functions
HASH_AREA_SIZE	larger value causes hash join costs to be cheaper, giving more hash joins
SORT_AREA_SIZE	large value causes sort costs to be cheaper, giving more sort merge joins
DB_FILE_MULTIBLOCK_READ_COUNT	large value gives cheaper table scan cost and favors table scans over indexes
The following parameters often need to be set in a data warehousing application:	
ALWAYS_ANTI_JOIN	sets the type of antijoin that Oracle uses: NESTED_LOOPS/MERGE/HASH

HASH_JOIN_ENABLED	enables or disables the hash join feature; should always be set to TRUE for data warehousing applications
SORT_DIRECT_WRITES	gives lower sort costs and more sort merge joins
PARTITION_VIEW_ENABLED	enables partition views
The following parameters rarely need to be changed:	
HASH_MULTIBLOCK_IO_COUNT	larger value causes hash join costs to be cheaper, giving more hash joins
SORT_WRITE_BUFFER_SIZE	large value causes sort costs to be cheaper, giving more sort merge joins
OPTIMIZER_SEARCH_LIMIT	the maximum number of tables in the FROM clause for which all possible join permutations will be considered
BITMAP_MERGE_AREA_SIZE	is the size of the area used to merge the different bitmaps that match a range predicate. Larger size will favor use of bitmap indexes for range predicates.

Note that the following sort parameters can now be modified using ALTER SESSION SET ... or ALTER SYSTEM SET ... DEFERRED:

```
SORT_AREA_SIZE
SORT_AREA_RETAINED_SIZE
SORT_DIRECT_WRITES
SORT_WRITE_BUFFERS
SORT_WRITE_BUFFER_SIZE
SORT_READ_FAC
```

Tips for Using the Cost-Based Approach

The cost-based optimization approach assumes that a query will be executed on a multiuser system with a fairly low buffer cache hit rate. Thus a plan selected by cost-based optimization may not be the best plan for a single user system with a large buffer cache. Timing a query plan on a single user system with a large cache may not be a good predictor of performance for the same query on a busy multiuser system. Analyzing a table uses more system resources than analyzing an index. It may be helpful to analyze the indexes for a table separately, with a higher sampling rate. Use of access path and join method hints causes the cost-based optimization to be invoked. Since cost-based optimization is dependent on statistics, it is important to analyze all tables referenced in a query, which has hints, even though rule-based optimization may have been selected as the system default.

25.5 Using Rule-Based Optimization

Rule-based optimization is supported in Oracle8, but users are advised to write any new applications using cost-based optimization. Cost-based optimization should be used for new applications and for data warehousing applications because it supports new and enhanced features. Much of the functionality in Oracle8 (such as hash joins, improved star query processing, and histograms) is available only through cost-based optimization. If you have developed existing OLTP applications using version 6 of Oracle and have carefully tuned your SQL statements based on the rules of the optimizer, you may want to continue using rule-based optimization when you upgrade these applications to Oracle8.

If you neither collect statistics nor add hints to your SQL statements, your statements will use rule-based optimization. However, you should eventually migrate your existing applications to use the cost-based approach, because the rule-based approach will not be available in future versions of Oracle. If you are using an application provided by a third-party vendor, check with the vendor to determine which type of optimization is best suited to that application.

You can enable cost-based optimization on a trial basis simply by collecting statistics. You can then return to rule-based optimization by deleting them or by setting either the value of the OPTIMIZER_MODE initialization parameter or the OPTIMIZER_GOAL parameter of the ALTER SESSION command to RULE. You can also use this value if you want to collect and examine statistics for your data without using the cost-based approach.

25.6 Introduction to Hints

As an application designer, you may know information about your data that the optimizer cannot. For example, you may know that a certain index is more selective for certain queries than the optimizer can determine. Based on this information, you may be able to choose a more efficient execution plan than the optimizer can. In such a case, you can use hints to force the optimizer to use your chosen execution plan.

Hints are suggestions that you give the optimizer for optimizing a SQL statement. Hints allow you to make decisions usually made by the optimizer. You can use hints to specify

- the optimization approach for a SQL statement
- the goal of the cost-based approach for a SQL statement
- the access path for a table accessed by the statement
- the join order for a join statement
- a join operation in a join statement

Note, however, that the use of hints involves extra code that must also be managed, checked, controlled.

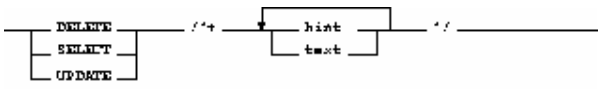
25.7 How to Specify Hints

Hints apply only to the optimization of the statement block in which they appear. A *statement block* is any one of the following statements or parts of statements:

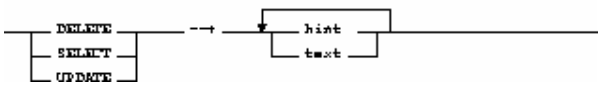
- a simple SELECT, UPDATE, or DELETE statement
- a parent statement or subquery of a complex statement
- a part of a compound query

For example, a compound query consisting of two component queries combined by the UNION operator has two statement blocks, one for each component query. For this reason, hints in this first component query apply only to its optimization, not to the optimization of the second component query. You can send hints for a SQL statement to the optimizer by enclosing them in a Comment within the statement.

A statement block can have only one Comment containing hints. This Comment can only follow the SELECT, UPDATE, or DELETE keyword. The syntax diagrams show the syntax for hints contained in both styles of Comments that Oracle supports within a statement block.



or



where:

DELETE *SELECT* *UPDATE*	Is a DELETE, SELECT, or UPDATE keyword that begins a statement block. Comments containing hints can only appear after these keywords.
+	Is a plus sign that causes Oracle to interpret the Comment as a list of hints. The plus sign must follow immediately after the Comment delimiter (no space is permitted).
hint	Is one of the hints discussed in this section. If the Comment contains multiple hints, each pair of hints must be separated by at least one space.
text	Is other Commenting text that can be interspersed with the hints.

If you specify hints incorrectly, Oracle ignores them but does not return an error:

- Oracle ignores hints if the Comment containing them does not follow a DELETE, SELECT, or UPDATE keyword.
- Oracle ignores hints containing syntax errors, but considers other correctly specified hints within the same Comment.
- Oracle ignores combinations of conflicting hints, but considers other hints within the same Comment.

Oracle also ignores hints in all SQL statements in environments which use PL/SQL Version 1, such as SQL*Forms Version 3 triggers, Oracle Forms 4.5, and Oracle Reports 2.5. The optimizer only recognizes hints when using the cost-based approach. If you include any hint (except the RULE hint) in a statement block, the optimizer automatically uses the cost-based approach. The following sections show the syntax of each hint.

25.8 Hints for Optimization Approaches and Goals

The hints described in this section allow you to choose between the cost-based and the rule-based optimization approaches and, with the cost-based approach, between the goals of best throughput and best response time.

- ALL_ROWS
- FIRST_ROWS
- CHOOSE
- RULE

If a SQL statement contains a hint that specifies an optimization approach and goal, the optimizer uses the specified approach regardless of the presence or absence of statistics, the value of the OPTIMIZER_MODE initialization parameter, and the OPTIMIZER_GOAL parameter of the ALTER SESSION command.

ALL_ROWS

The ALL_ROWS hint explicitly chooses the cost-based approach to optimize a statement block with a goal of best throughput (that is, minimum total resource consumption). For example, the optimizer uses the cost-based approach to optimize this statement for best throughput:

```
SELECT /*+ ALL_ROWS */ empno, ename, sal, job

   FROM emp

   WHERE empno = 7566;
```

FIRST_ROWS

The FIRST_ROWS hint explicitly chooses the cost-based approach to optimize a statement block with a goal of best response time (minimum resource usage to return first row). This hint causes the optimizer to make these choices:

- If an index scan is available, the optimizer may choose it over a full table scan.
- If an index scan is available, the optimizer may choose a nested loops join over a sort-merge join whenever the associated table is the potential inner table of the nested loops.
- If an index scan is made available by an ORDER BY clause, the optimizer may choose it to avoid a sort operation.

For example, the optimizer uses the cost-based approach to optimize this statement for best response time:

```
SELECT /*+ FIRST_ROWS */ empno, ename, sal, job

   FROM emp

   WHERE empno = 7566;
```

The optimizer ignores this hint in DELETE and UPDATE statement blocks and in SELECT statement blocks that contain any of the following syntax:

- set operators (UNION, INTERSECT, MINUS, UNION ALL)
- GROUP BY clause
- FOR UPDATE clause
- group functions
- DISTINCT operator

These statements cannot be optimized for best response time because Oracle must retrieve all rows accessed by the statement before returning the first row. If you specify this hint in any of these statements, the optimizer uses the cost-based approach and optimizes for best throughput.

If you specify either the ALL_ROWS or FIRST_ROWS hint in a SQL statement and the data dictionary contains no statistics about any of the tables accessed by the statement, the optimizer uses default statistical values (such as allocated storage for such tables) to estimate the missing statistics and subsequently to choose an execution plan. Since these estimates may not be as accurate as those generated by the ANALYZE command, you should use the ANALYZE command to generate statistics for all tables accessed by statements that use cost-based optimization. If you specify hints for access paths or join operations along with either the ALL_ROWS or FIRST_ROWS hint, the optimizer gives precedence to the access paths and join operations specified by the hints.

CHOOSE

The CHOOSE hint causes the optimizer to choose between the rule-based approach and the cost-based approach for a SQL statement based on the presence of statistics for the tables accessed by the statement. If the data dictionary contains statistics for at least one of these tables, the optimizer uses the cost-based approach and optimizes with the goal of best throughput. If the data dictionary contains no statistics for any of these tables, the optimizer uses the rule-based approach.

```
SELECT /*+ CHOOSE */
empno, ename, sal, job
        FROM emp
        WHERE empno = 7566;
```

RULE

The RULE hint explicitly chooses rule-based optimization for a statement block. This hint also causes the optimizer to ignore any other hints specified for the statement block. For example, the optimizer uses the rule-based approach for this statement:

```
SELECT   --+ RULE

empno, ename, sal, job

   FROM emp

   WHERE empno = 7566;
```

The RULE hint, along with the rule-based approach, may not be supported in future versions of Oracle.

25.9 Short Summary

Optimization is the process of choosing the most efficient way to execute a SQL statement. This is an important step in the processing of any data manipulation language (DML) statement: SELECT, INSERT, UPDATE, or DELETE.

Hints are suggestions that you give the optimizer for optimizing a SQL statement. Hints allow you to make decisions usually made by the optimizer.

25.10 Brain Storm

1. What does Oracle use to manage the SQL & PL/SQL in the Library cache
   a) Program Global Area b) User Global Area c) Clusters d) LRU Algorithm

2. Which occurrence place blocks in the DB Buffer Cache ?
   a) System Startup b) LGWR process reading blocks in the buffer
   c) DBWR process reading blocks into the buffer d) Server process reading blocks into the buffers for users

3. Which DML statement will generate the lease amount of Rollback on the database ?
   a) Delete b) Insert c) Update d) Alter

4. How many Redolog Groups must be created for a Database running in Archivelog mode ?
   a) Six b) three c) Two d) One

5. Which Oracle Mechanism maintains all locks in the Database
   a) Data Dictionary b) Library cache c) Enqueuses d) Dispatchers

Lecture 26

Archiving Redo Information

Objectives

After completing this lesson, you should be able to do the following:

- Choosing between noarchivelog and archivelog mode
- Discuss about Tuning archiving on and off
- Displaying archiving status information
- Specifying the archived redo log filename format and destination

Coverage Plan

Lecture 26

26.1 Snap Shot

26.2 Choosing Between Noarchivelog And Archivelog Mode

26.3 Turning Archiving On And Off

26.4 Tuning Archiving

26.5 Displaying Archiving Status Information

26.6 Specifying The Archived Redo Log Filename Format And Destination

26.7 Short Summary

26..8 Brain Storm

26.1 Snap Shot

This chapter describes how to create and maintain the archived redo log, and includes the following topics:

- Choosing Between NOARCHIVELOG and ARCHIVELOG Mode
- Turning Archiving On and Off
- Tuning Archiving
- Displaying Archiving Status Information
- Specifying the Archived Redo Log Filename Format and Destination

26.2 Choosing Between NOARCHIVELOG and ARCHIVELOG Mode

This section describes the issues you must consider when choosing to run your database in NOARCHIVELOG or ARCHIVELOG mode, and includes the following topics:

- Running a Database in NOARCHIVELOG Mode
- Running a Database in ARCHIVELOG Mode

Running a Database in NOARCHIVELOG Mode

When you run your database in NOARCHIVELOG mode, the archiving of the online redo log is disabled. Information in the database's control file indicates that filled groups are not required to be archived. Therefore, after a filled group becomes inactive and the checkpoint at the log switch completes, the group is available for reuse by LGWR.

NOARCHIVELOG mode protects a database only from instance failure, not from disk (media) failure. Only the most recent changes made to the database, stored in the groups of the online redo log, are available for instance recovery. In other words, if you are using NOARCHIVELOG mode, you can only *restore* (not recover) the database to the point of the most recent full database backup. You cannot recover subsequent transactions.

Also, in NOARCHIVELOG mode you cannot perform online tablespace backups. Furthermore, you cannot use online tablespace backups previously taken while the database operated in ARCHIVELOG mode. Only full backups taken while the database is closed can be used to restore a database operating in NOARCHIVELOG mode. Therefore, if you decide to operate a database in NOARCHIVELOG mode, take full database backups at regular, frequent intervals.

Running a Database in ARCHIVELOG Mode

When you run a database in ARCHIVELOG mode, the archiving of the online redo log is enabled. Information in a database control file indicates that a group of filled online redo log files cannot be used by LGWR until the group is archived. A filled group is immediately available to the process performing the archiving after a log switch occurs (when a group becomes inactive). The process performing the archiving does not have to wait for the checkpoint of a log switch to complete before it can access the inactive group for archiving.

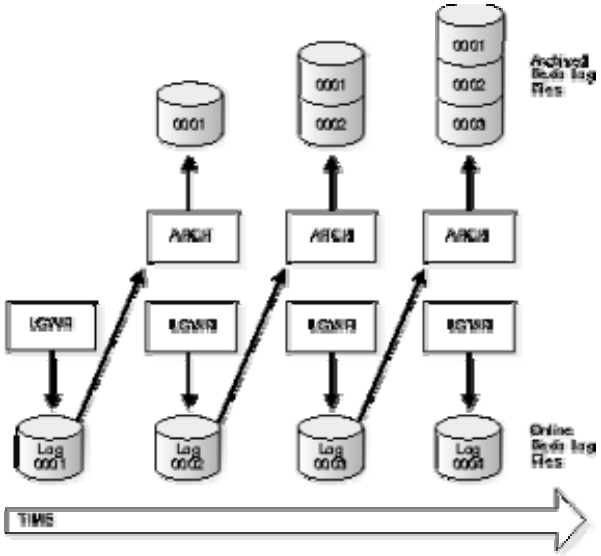


Figure 26-1: Online Redo Log File Use in ARCHIVELOG Mode

ARCHIVELOG mode enables complete recovery from disk failure as well as instance failure, because all changes made to the database are permanently saved in an archived redo log. If *all* databases in a distributed database operate in ARCHIVELOG mode, you can perform coordinated distributed database recovery. However, if *any* database in a distributed database uses NOARCHIVELOG mode, recovery of a global distributed database (to make all databases consistent) is limited by the last full backup of any database operating in NOARCHIVELOG mode.

Also, the entire database can be open and available for normal use while you back up or recover all or part of the database in ARCHIVELOG mode. Note that extra administrative operations are required to manage the files of the archived redo log and that you must have a dedicated tape drive or additional disk space to store the archived redo log files when the database operates in ARCHIVELOG mode. You must also decide how filled groups of the online redo log are to be archived. An instance can be configured to have Oracle automatically archive filled online redo log files, or you can manually archive filled groups.

26.3 Turning Archiving On and Off

This section describes aspect of archiving, and includes the following topics:

- Setting the Initial Database Archiving Mode
- Changing the Database Archiving Mode
- Enabling Automatic Archiving
- Disabling Automatic Archiving
- Performing Manual Archiving

You set a database's initial archiving mode as part of database creation. Usually, you can use the default of NOARCHIVELOG mode at database creation because there is no need to archive the redo information generated then. After creating the database, decide whether to change from the initial archiving mode. After a database has been created, you can switch the database's archiving mode on demand. However, you should generally not switch the database between archiving modes.

Setting the Initial Database Archiving Mode

When you create the database, you set the initial archiving mode of the redo log in the CREATE DATABASE statement. If you do not specify either ARCHIVELOG or NOARCHIVELOG, NOARCHIVELOG is the default.

Changing the Database Archiving Mode

To switch a database's archiving mode between NOARCHIVELOG and ARCHIVELOG mode, use the SQL command ALTER DATABASE with the ARCHIVELOG or NOARCHIVELOG option. The following statement switches the database's archiving mode from NOARCHIVELOG to ARCHIVELOG:

```
ALTER DATABASE ARCHIVELOG;
```

Before switching the database's archiving mode, perform the following operations:

1. Shut down the database instance.

An open database must be closed and dismounted and any associated instances shut down before the database's archiving mode can be switched. Archiving cannot be disabled if any datafiles need media recovery.

2. Back up the database.

Before making any major alteration to a database, always back up the database to protect against any problems that might occur.

3. Perform any operating system specific steps *(optional)*.

These steps may involve exiting Enterprise Manager to configure how Oracle will perform the archiving of the filled groups. Once this operation is complete, start Enterprise Manager again and continue to Step 4.

4. Start up a new instance and mount but do not open the database.
   To enable or disable archiving, the database must be mounted but not open.
5. Switch the database's archiving mode.

After using the ALTER DATABASE command to switch a database's archiving mode, open the database for normal operation. If you switched to ARCHIVELOG mode, you should also set the archiving options-decide whether to enable Oracle to archive groups of online redo log files automatically as they fill.

Enabling Automatic Archiving

If your operating system permits, you can enable automatic archiving of the online redo log. Under this option, no action is required to copy a group after it fills; Oracle automatically archives groups after they are filled. For this convenience alone, automatic archiving is the method of choice for archiving the filled groups of online redo log files.

To enable automatic archiving after instance startup, you must be connected to Oracle with administrator privileges.

Attention: Oracle does not automatically archive log files unless the database is also in ARCHIVELOG mode. Automatic archiving can be enabled before or after instance startup.

Enabling Automatic Archiving at Instance Startup

To enable automatic archiving of filled groups each time an instance is started, include the LOG_ARCHIVE_START parameter, set to TRUE, in the database's parameter file:

```
LOG_ARCHIVE_START=TRUE
```

The new value takes effect the next time you start the database.
Enabling Automatic Archiving After Instance Startup

To enable automatic archiving of filled online redo log groups without shutting down the current instance, use the SQL command ALTER SYSTEM with the ARCHIVE LOG START parameter; you can optionally include the archiving destination.

```
ALTER SYSTEM ARCHIVE LOG START;
```

Using either of the options above, you do not need to shut down the instance to enable automatic archiving. However, if an instance is shut down and restarted after automatic archiving is enabled, the instance is reinitialized using the settings of the parameter file, which may or may not enable automatic archiving.

Disabling Automatic Archiving

You can disable automatic archiving of the online redo log groups at any time. However, once automatic archiving is disabled, you must manually archive groups of online redo log files in a timely fashion. If a database is operated in ARCHIVELOG mode, automatic archiving is disabled, and all groups of online redo log files are filled but not archived, then LGWR cannot reuse any inactive groups of online redo log groups to continue writing redo log entries. Therefore, database operation is temporarily suspended until the necessary archiving is performed.

To disable automatic archiving after instance startup, you must be connected with administrator privilege and have the ALTER SYSTEM privilege. Automatic archiving can be disabled at or after instance startup.

Disabling Automatic Archiving at Instance Startup

To disable the automatic archiving of filled online redo log groups each time a database instance is started, set the LOG_ARCHIVE_START parameter of a database's parameter file to FALSE:

```
LOG_ARCHIVE_START=FALSE
```

The new value takes effect the next time the database is started.
Disabling Automatic Archiving after Instance Startup

To disable the automatic archiving of filled online redo log groups without shutting down the current instance, use the SQL command ALTER SYSTEM with the ARCHIVE LOG STOP parameter. The following statement stops archiving:

```
ALTER SYSTEM ARCHIVE LOG STOP;
```

If ARCH is archiving a redo log group when you attempt to disable automatic archiving, ARCH finishes archiving the current group, but does not begin archiving the next filled online redo log group. The instance does not have to be shut down to disable automatic archiving. However, if an instance is shut down and restarted after automatic archiving is disabled, the instance is reinitialized using the settings of the parameter file, which may or may not enable automatic archiving.

Performing Manual Archiving

If a database is operating in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived. You can manually archive groups of the online redo log whether or not automatic archiving is enabled:

- If automatic archiving is not enabled, you must manually archive groups of filled online redo log files in a timely fashion. If all online redo log groups are filled but not archived, LGWR cannot reuse any inactive groups of online redo log members to continue writing redo log entries. Therefore, database operation is temporarily suspended until the necessary archiving is performed.

- If automatic archiving is enabled, but you want to rearchive an inactive group of filled online redo log members to another location, you can use manual archiving. (However, the instance can decide to reuse the redo log group before you have finished manually archiving, and thereby overwrite the files; if this happens, Oracle will put an error message in the ALERT file.)

To manually archive a filled online redo log group, you must be connected with administrator privileges. Manually archive inactive groups of filled online redo log members using the SQL command ALTER SYSTEM with the ARCHIVE LOG clause. The following statement archives all unarchived log files:

```
ALTER SYSTEM ARCHIVE LOG ALL;
```

26.4 Tuning Archiving

This section describes aspects of tuning the archive process, and includes the following topics:

- Minimizing the Impact on System Performance
- Improving Archiving Speed

For most databases, the archive process has no effect on overall system performance. In some large database sites, however, archiving can have an impact on system performance. On one hand, if the archiver works very quickly, overall system performance can be reduced while the archiver runs, since CPU cycles are being consumed in archiving. On the other hand, if the archiver runs extremely slowly, it has little detrimental effect on system performance, but it takes longer to archive redo log files, and can be a bottleneck if all redo log groups are unavailable because they are waiting to be archived.

For these large database sites you can tune archiving to cause it to run either as slowly as possible without being a bottleneck or as quickly as possible without reducing system performance substantially. To do so, adjust the values of the initialization parameters LOG_ARCHIVE_BUFFERS (the number of buffers allocated to archiving) and LOG_ARCHIVE_BUFFER_SIZE (the size of each such buffer).

Note: When you change the value of LOG_ARCHIVE_BUFFERS or LOG_ARCHIVE_BUFFER_SIZE, the new value takes effect the next time you start the instance.

Minimizing the Impact on System Performance

To make the archiver work as slowly as possible without forcing the system to wait for redo logs, begin by setting the number of archive buffers (LOG_ARCHIVE_BUFFERS) to 1 and the size of each buffer (LOG_ARCHIVE_BUFFER_SIZE) to the maximum possible.
If the performance of the system drops significantly while the archiver is working, make the value of LOG_ARCHIVE_BUFFER_SIZE lower, until system performance is no longer reduced when the archiver runs.

Improving Archiving Speed

To improve archiving performance (for example, if you want to stream input to a tape drive), use multiple archive buffers, so that the archiver process can read the archive log at the same time that it writes the output log. You can set LOG_ARCHIVE_BUFFERS to 2, but for a very fast tape drive you might want to set it to 3 or more. Then, set the size of the archive buffers to a moderate number, and increase it until archiving is as fast as you want it to be without impairing system performance.

26.5 Displaying Archiving Status Information

To see the current archiving mode, query the V$DATABASE view:

```
SELECT log_mode FROM sys.v$database;
LOG_MODE
------------
NOARCHIVELOG
```

The V$ARCHIVE and V$LOG data dictionary views also contain archiving information of a database. For example, the following query lists all log groups for the database and indicates the ones that remain to be archived:

```
SELECT group#, archived
   FROM sys.v$log;

GROUP#       ARC
---------- ---
1            YES
2            NO
```

The command ARCHIVE LOG with the LIST parameter also shows archiving information for the connected instance:
ARCHIVE LOG LIST;

```
Database log mode                      ARCHIVELOG
Automatic archival                     ENABLED
Archive destination                    destination
Oldest online log sequence             30
Next log sequence to archive           32
Current log sequence number            33
```

This display tells you all the necessary information regarding the redo log settings for the current instance:

- The database is currently operating in ARCHIVELOG mode.
- Automatic archiving is enabled.
- The destination of the archived redo log (operating system specific) is *destination* (corresponds to LOG_ARCHIVE_DEST or an overriding destination).
- The oldest filled online redo log group has a sequence number of 30.
- The next filled online redo log group to archive has a sequence number of 32.
- The current online redo log file has a sequence number of 33.

You must archive all redo log groups with a sequence number equal to or greater than the *Next log sequence to archive*, yet less than the *Current log sequence number*. For example, the display above indicates that the online redo log group with sequence number 32 needs to be archived.

26.6 Specifying the Archived Redo Log Filename Format and Destination

When the database is used in ARCHIVELOG mode, Oracle must know the archived redo log filename format and destination so that automatic or manual archiving creates uniquely named archived redo log files in the proper location.

Archived redo log files are uniquely named as specified by the LOG_ARCHIVE_FORMAT parameter. Filename format is operating system specific; for most operating systems it consists of a text string, one or more parameters, and a filename extension. When a filled online redo log group is archived, the archiving process concatenates the supplied text string with the return values of the specified parameters to create uniquely identified archived redo log files. Each parameter has an upper bound, which is operating system dependent.

Table 26-1 lists the parameters that can be included in a filename format and corresponding examples to show how the parameter affects the filenames created by the archiving process.
Table 30-1: Archived Redo Log Filename Format Parameters

Parameter	Description	Example[a]
%T	thread number, left-zero-padded	arch0000000001
%t	thread number, not padded	arch1
%S	log sequence number, left–zero-padded	arch0000000251
%s	log sequence number, not padded	arch251

a. Assume LOG_ARCHIVE_FORMAT=arch%parameter, and the upper bound for all parameters is 10 characters.

The different options are provided so that you can customize the archived redo log filenames as you need. For example, you might want to take into account the operating system sorting algorithm used to list filenames.

The %T and %t are useful only when the Oracle Parallel Server is used. In a non-Parallel Server configuration, you must decide whether to use %S or %s to identify each archived redo log file uniquely. The following is a typical example of a common archived redo log filename format:

LOG_ARCHIVE_FORMAT = arch%S.arc

Here, *arch* is the filename, %S is the zero-padded log sequence parameter, and *.arc* is the file extension. Assuming the upper bound for the %S parameter is four, this filename format generates archived redo log filenames of the following format:
arch0001.arc
arch0002.arc
arch0003.arc
.
.
Take into account the maximum operating system filename length when specifying the archive filename format. If ARCH or a user process attempts to archive a file and the supplied filename format is too large, the process fails to archive the file.

The archived redo log destination is also operating system-specific. For most operating systems, the archive redo log destination points to a disk drive and a file directory. If permitted by your Oracle Server, this destination can also point to a tape drive dedicated to Oracle for archiving filled online redo log files.

The archived redo log destination is determined at instance startup by the LOG_ARCHIVE_DEST initialization parameter, but can be overridden while the instance is up:

- If a database's parameter file is edited to include a destination using the LOG_ARCHIVE_DEST parameter, the current instance must be shut down and restarted to read the new parameter file.

- If the current instance cannot be shut down, but the archived redo log destination must be specified or changed for automatic archiving, use the ALTER SYSTEM ARCHIVE LOG START 'destination' statement to override the automatic archiving destination.

- During manual archiving, a specified destination overrides the default archived redo log destination. However, automatic archiving continues to use the current automatic archive destination. If no destination is specified, Oracle automatically uses the destination specified by the LOG_ARCHIVE_DEST parameter of the parameter file used to start the instance. If no destination is supplied by the LOG_ARCHIVE_DEST parameter, Oracle uses a default destination that is operating system-dependent.

26.7 Short Summary

- ♣ A database can operate in two distinct modes: NOARCHIVELOG mode (media recovery disabled) or ARCHIVELOG mode (media recovery enabled).

- ♣ If a database is used in NOARCHIVELOG mode, the archiving of the online redo log is disabled. Information in the database's control file indicates that filled groups are not required to be archived.

- ♣ If an Oracle database is operated in ARCHIVELOG mode, the archiving of the online redo log is enabled. Information in a database control file indicates that a group of filled online redo log files cannot be reused by LGWR until the group is archived.

26.8 Brain Storm

1. How would you backup an order processing database that is critical to the cash flow of your company?

   a) Backup the Database daily
   b) Backup the Database monthly
   c) Backup the Database during inventory
   d) Backup the Database weekly

2. Which type of backup is performed with online backup?

   a) Database level Backup
   b) Tablespace level Backup
   c) Table Level Backup

3. To ensure that all data is captured in an offline Backup the DBA should close the Database using the

   a) Shutdown Immediate
   b) Shutdown Normal
   c) Shutdown Transactional
   d) Check all that apply

4. Cumulative database EXPORT exports only tables that having modified or created

   a) since the most incremental export
   b) since the most recent cumulative export
   c) since the most complete export
   d) check all that apply

ꕥBest of Luckꕥ

MS 3.3 Oracle Database Administration

Part 1 Basic Database Administration

Lecture 1 The Oracle Database Administration

Types of Oracle Users - Database Administrator Security and Privileges -The DBA Role -Database Administrator Utilities -Initial Priorities of a Database Administrator -Identifying Oracle Software Releases -Release Number Format

Lecture 2 Database Administrator Authentication

Database Administrator Authentication - Using Operating System Authentication -OSOPER and OSDBA - Using an Authentication Password File -Password File Administration

Lecture 3 The Oracle Server

The Oracle Server - Oracle Databases -Memory Structure and Processes - Memory Structures -Process Architecture -The Program Interface -An Example of How Oracle Works

Lecture 4 Database structure and space management

Database structure and space management - logical database structure – data blocks – Extents – segments – physical database structure – datafiles – redo log files – control files

Lecture 5 The Tablespace and Datafiles

Introduction to tablespace and datafiles – tablespace -datafiles

Lecture 6 Creating Oracle Database

Considerations Before Creating a Database -Creating an Oracle Database Troubleshooting Database Creation -Dropping a Database -Parameters -Considerations After Creating a Database -Initial Tuning Guidelines

Lecture 7 Start UP and Shut Down

Startup Procedures-Altering Database Availability -Shutdown Procedures -Using Parameter Files

Part 2 Oracle Server Configurations

Lecture 8 Managing Oracle Process

Configuring Oracle for Dedicated Server Processes -Configuring Oracle for Multi-Threaded Server Processes - Modifying Server Processes-Terminating Sessions

Lecture 9 Managing Onling Redo Log

Planning the Online Redo Log -Multiplex the Online Redo Log -Creating Online Redo Log Groups and Members -Renaming and Relocating Online Redo Log Members -Dropping Online Redo Log Groups - Dropping Online Redo Log Members -Clearing an Online Redo Log File -Restrictions -Listing Information about the Online Redo Log

Lecture 10 Managing Control Files

Guidelines for Control Files – multiplex control files - Creating Control Files - Relocating Control Files - Dropping Control Files

Lecture 11 Managing Job Queues

SNP Background Processes - Managing Job Queues -Terminating a Job -Viewing Job Queue Information

Part 3 Database Storage

Lecture 12 Managing Table Spaces

Guidelines for managing tablespace-Creating tablespaces-managing tablespace allocation-making tablespace read-only-dropping tablespaces-viewing information about tablespaces

Lecture 13 Managing Data Files

Guidelines for Managing Datafiles - Creating and Adding Datafiles to a Tablespace-Changing a Datafile's Size -Renaming and Relocating Datafiles -Verifying Data Blocks in Datafiles -Viewing Information About Datafiles

Lecture 14 Managing Schema Objects

Managing Space in Data Blocks -The PCTFREE Parameter -The PCTUSED Parameter -Setting Storage Parameters -Deallocating Space

Lecture 15 Managing Partioned Tables and Indexes

What Are Partitioned Tables and Indexes? -Creating Partitions -1

Lecture 16 Managing Rollback Segment

Guidelines for Managing Rollback Segments -Creating Rollback Segments -Specifying Storage Parameters -Taking Rollback Segments Online and Offline -Dropping Rollback Segments -Monitoring Rollback Segment.

Part 4 Database Security

Lecture 17 Security Policy

System Security Policy -Data Security Policy -User Security Policy -Password Management Policy -Auditing Policy

Lecture 18 User and Resources

Session and User Licensing -User Authentication -Oracle Users -Managing Resources with Profiles -Listing Information About Database Users and Profiles

Lecture 19 Privileges and Roles

Identifying User Privileges - Managing User Roles - Granting User Privileges and Roles -Revoking User Privileges and Roles

Lecture 20 Auditing

Introduction to auditing - Statement Auditing - Privilege Auditing - Schema Object Auditing - Focusing Statement, Privilege, and Schema Object Auditing

Lecture 21 Backup

Backups - Types of failure - Types of backups - Backup format

Lecture 22 Recovery

Recovery – Introduction to recovery – errors and failure – structure used for database recovery – rolling forward and rolling back – rollback segments and rolling back – recovery concepts and strategies

Part 5 Oracle Performance Tuning

Lecture 23 What is Performance Tuning

Trade-offs Between Response Time and Throughput - Critical Resources - Effects of Excessive Demand - Adjustments to Relieve Problems- Steps of tuning method

Lecture 24 Sources and Data for Tuning

Sources of data for tuning - Oracle enterprise manager application - Introduction to Oracle Enterprise Manager - Oracle Performance Manager - Oracle Top Sessions - Oracle Trace - Oracle Tablespace Manager - Oracle Expert

Lecture 25 Optimization Modes and Hints

Using Cost-Based Optimization - Choosing a Goal for the Cost-Based Approach - Parameters Which Affect Cost-Based Optimization Plans - Using Rule-Based Optimization - Introduction to Hints - How to Specify Hints - Hints for Optimization Approaches and Goals

Lecture 26 Archiving Redo Information

Choosing Between NOARCHIVELOG and ARCHIVELOG Mode - Turning Archiving On and Off - Tuning Archiving – Displaying Archiving Status Information - Specifying the Archived Redo Log Filename Format and Destination