Oracle Database 10*g*: Administration Workshop I

Student Guide

D17090GC10
Edition 1.0
March 2004
D39126

ORACLE®

Authors

Ric Van Dyke
Russ Lowenthal

Technical Contributors and Reviewers

Donna Keesling
S. Matt Taylor
Jean-Francois Verrier
Craig Hollister
Bob Bungenstock
Tony Woodell
Joel Goodman
John Watson
Dairy Chan
Martin Jensen
Janet Stern
Dr. Sabine Teuber
Kyle Hailey
Christopher Lawless
Dominique Laugraud
Isabelle Marchand
Yi Lu

Editor

Elizabeth Treacy

Publisher

Jobi Varghese

Contents

1 Introduction

ORACLE

Objectives

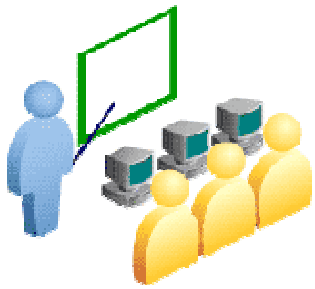
After completing this lesson, you should be able to do the following:

- **Explain the course objectives**
- **Identify the Oracle product line**
- **Describe the basic concepts of a relational database**
- **List the core database administrator tasks**

Course Objectives

In this course, you will gain hands-on experience with:

- **Installing, creating, and administering an Oracle Database 10*g* Enterprise Edition database**
- **Configuring the database for an application**
- **Implementing a backup and recovery strategy**
- **Employing basic monitoring procedures**



ORACLE

Course Objectives

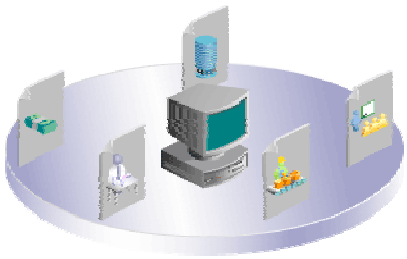
In this course you will install the Oracle Database 10*g* Enterprise Edition software, create a new database, and learn how to administer the database.

You will also configure the database to support an application, performing tasks such as creating users, defining storage structures, and setting up security. This course uses a fictional application. However you will be performing all the core tasks necessary for a real application.

Database administration doesn't end after the database is configured, so you will also learn how to protect your database by designing a backup and recovery strategy, and how to monitor the database to ensure it operates smoothly.

Oracle Products

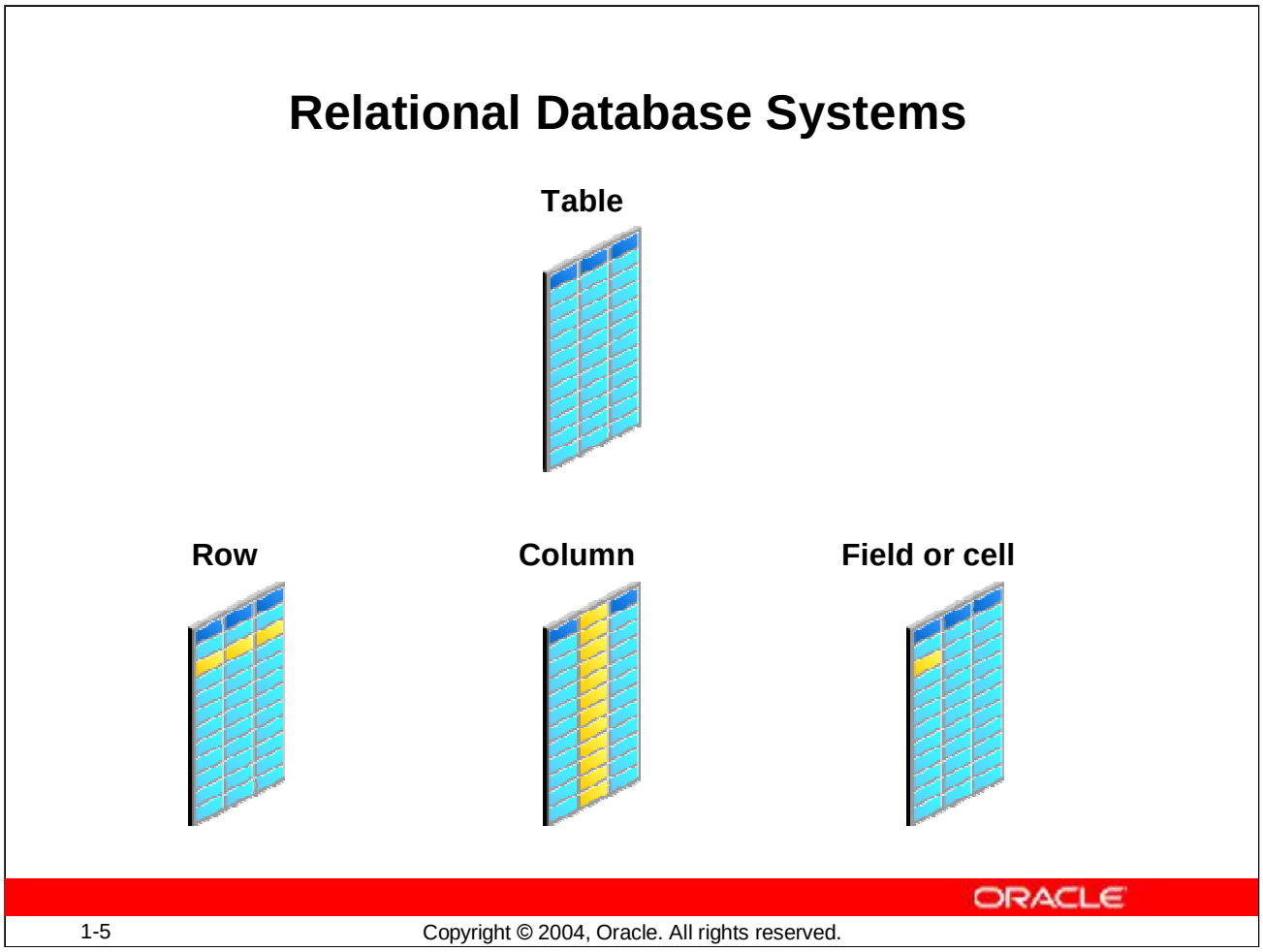
- Oracle Database
- Oracle Application Server
- Oracle Applications
- Oracle Collaboration Suite
- Oracle Development Suite
- Oracle Services



ORACLE

Oracle Products

- Oracle Database: Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost-effective way to manage information and applications. Oracle Database is available in three editions — Enterprise, Standard, and Personal.
- Oracle Application Server: Oracle's Java 2 Enterprise Edition (J2EE) certified server, integrating everything needed to develop and deploy Web-based applications. The application server deploys e-business portals, Web services, and transactional applications including PL/SQL, Oracle Forms, and J2EE-based applications.
- Oracle Applications: Oracle E-Business Suite is a complete set of business applications for managing and automating processes across your organization.
- Oracle Collaboration Suite: Oracle Collaboration Suite is a single, integrated system for all your organization's communications data: voice, e-mail, fax, wireless, calendar information, and files.
- Oracle Development Suite: Oracle Developer Suite is a complete, integrated environment that combines application development and business intelligence tools.
- Oracle Services: Services like Oracle Consulting and Oracle University provide the expertise you need to keep your Oracle project on track and running smoothly.



Relational Database Systems

Dr. E. F. Codd published , “A Relational Model of Data for Large Shared Data Banks,” in June, 1970, in the Association of Computer Machinery (ACM) journal, *Communications of the ACM*. Codd's model is now accepted as the definitive model for relational database management systems (RDBMS). The language, called structured English query language (SEQUEL), was developed by IBM Corporation, Inc., to use Codd's model. SEQUEL later became SQL (still pronounced “sequel”). In 1979, Relational Software, Inc. (now Oracle Corporation) introduced the first commercially available implementation of SQL. Today, SQL is accepted as the standard RDBMS language.

The basic element of a relational database system is a two-dimensional table. A table consists of zero or more rows of data. Each row has one or more columns. A single column of a single row is referred to as a field or cell of data.

Each row of data is a collection of data items relating to a given subject. For example, you might have a table to hold information about employees in your company. Each row represents one employee in the company, and the columns may be things such as, first and last name, e-mail, phone number, hire date, and so on. Each column has a name and a data type associated with it. The data type defines what kind of data is allowed in the column, for example numbers or characters.

How the Data Is Organized

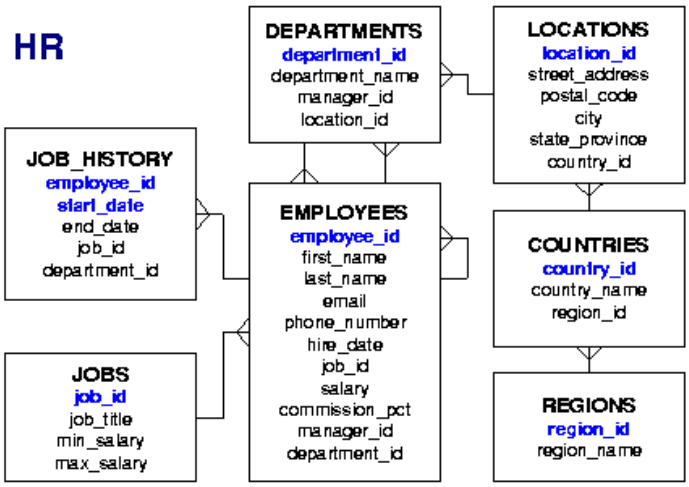
ORACLE

How the Data Is Organized

A relational database has many tables, which can be used independently or joined together by using a common column (or columns) between the tables. With Oracle Database 10*g* you may require that the column used to establish the relationship with another table contain values that already exist in that table. This mandatory relationship is known as referential integrity. Using the example of employees, each employee in your company is assigned to a particular department. Because a department may have many employees, it would be redundant to have the department information stored with each row in the employees table. Instead, each row in the employees table has a column which is an identifier pointing to the departments table. This allows you to join the employees table with the department table and see which department an employee is assigned to.

This identifier is a key field. In the employees table the departments field is called a foreign key, because it points to a row in a different (foreign) table. In the department table the field referenced by a foreign key in the employees table is typically the primary key, or sometimes an alternate key. This primary key field must have certain properties in the department table. The field must be unique and must always have a value. This way each row in the employees table will point to one and only one row in the department table.

Integrity Constraints



ORACLE

Integrity Constraints

You can use integrity constraints to define business rules in your relational database. The diagram shows an entity relationship diagram (ERD) of a Human Resources (HR) application. The ERD shows the tables and relations between them. The ERD is used to define constraints that are used to ensure valid data is stored in your database tables.

Integrity constraints are defined as part of a table's definition in the data dictionary, so that all applications accessing the table must adhere to the rules defined by the constraints. When you need to change a rule, you need to change it only once at the database level and not in each application that accesses the table.

Oracle Database 10*g* supports the following types of constraints:

- `NOT NULL`: Disallows nulls (empty entries) in a table's column
- `UNIQUE`: Disallows duplicate values in a column or set of columns
- `PRIMARY KEY`: Disallows duplicate values and nulls in a column or set of columns
- `FOREIGN KEY`: Requires each value in a column or set of columns to match a value in a related table's `UNIQUE` or `PRIMARY KEY`. `FOREIGN KEY` integrity constraints also define referential integrity actions that dictate what Oracle should do with dependent data if the data it references is altered.
- `CHECK`: Disallows values that do not satisfy the logical expression of the constraint.

Structured Query Language

- **Structured query language (SQL): A standard interactive and programming language for getting information from and updating information in a database**
- **SQL provides statements for a variety of tasks, including:**
  - **Querying data**
  - **Inserting, updating, and deleting rows in a table**
  - **Creating, replacing, altering, and dropping objects**
  - **Controlling access to the database and its objects**
  - **Guaranteeing database consistency and integrity**

ORACLE

Structured Query Language

Structured Query Language (SQL) is the set of statements with which all programs and users access data in an Oracle database. Application programs and Oracle tools often allow users access to the database without using SQL directly, but these applications in turn must use SQL when executing the user's request.

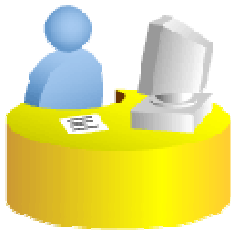
SQL enables you to work with data at the logical level. For example, to retrieve a set of rows from a table, you define a condition that is used to filter the rows. All rows satisfying the condition are retrieved in a single step and can be passed as a unit to an application or to another SQL statement. You do not need to know how the rows are physically stored or retrieved. All SQL statements use the Oracle Database optimizer to determine the most efficient means of accessing the specified data.

SQL will be covered in more detail in Lesson 4, "Database Interfaces." Also refer to the *Oracle Database SQL Reference* for additional information.

Tasks of an Oracle Database Administrator

Prioritized approach for designing, implementing, and maintaining an Oracle database:

1. **Evaluate the database server hardware.**
2. **Install the Oracle software.**
3. **Plan the database.**
4. **Create and open the database.**
5. **Back up the database.**
6. **Enroll system users.**
7. **Implement the database design.**
8. **Recover from database failure.**
9. **Monitor database performance.**



ORACLE

Tasks of an Oracle Database Administrator

An Oracle database administrator (DBA) is typically responsible for installing the Oracle software and creating the database. You may be responsible for creating the database storage structures such as tablespaces. In addition, you may create the schema or set of objects to hold application data.

You need to ensure that the database is available for users. You can accomplish this by starting up the database, backing up the database on a regular basis, and monitoring the performance of the database.

As you proceed through the lessons in this course, you will learn how to perform each of these tasks. You can also refer to the *Oracle Database Administrator's Guide* for additional information about each of the tasks outlined in the slide.

Summary

In this lesson, you should have learned how to:

- **Explain the course objectives**
- **Describe the Oracle product line**
- **Identify the basic concepts of a relational database**
- **List the core Database Administrator tasks**

ORACLE

2

Installing Oracle Database 10*g* Software

Objectives

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

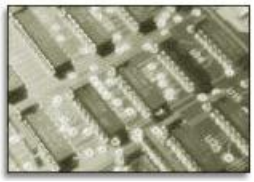
- **Identify system requirements**
- **Use Optimal Flexible Architecture**
- **Install software with Oracle Universal Installer**

System Requirements

Hardware:

- **512 MB of physical random access memory (RAM)**
- **1 GB of swap space (or twice the size of RAM)**
- **400 MB of disk space in the temporary directory (`/tmp` or `\Temp`)**
- **1.5 GB of disk space for the Oracle software**
- **1.5 GB of disk space for the preconfigured database**

Operating system: See documentation



ORACLE

System Requirements

The hardware requirements listed in the slide are minimal requirements across all platforms. Your installation may require more, especially disk space. Also, as a general rule, the more RAM you have available the better.

Note: On systems with 2 GB or more of RAM, the swap space can be between one and two times the size of RAM.

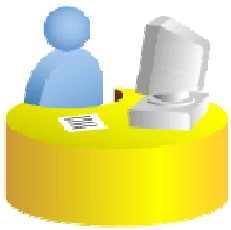
Because each operating system has different requirements, you must refer to the operating system–specific Oracle documentation for information about operating system requirements. These will list the releases of the operating system that are supported, along with how to set up any operating system groups or users, and other steps that are needed for the installation.

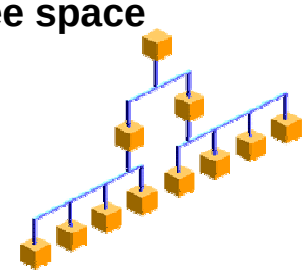
For this course all the necessary operating system requirement steps have been completed.

Optimal Flexible Architecture (OFA)

OFA is designed to:

- **Organize large amounts of software**
- **Facilitate routine administrative tasks**
- **Facilitate switching between multiple Oracle databases**
- **Adequately manage and administer database growth**
- **Help eliminate fragmentation of free space**





ORACLE

Optimal Flexible Architecture (OFA)

OFA is a method for configuring the Oracle database and other databases. OFA takes advantage of the OS and disk subsystems capabilities to create an easy-to-administer configuration that allows maximum flexibility for growing and high-performance databases. The methods described here are the basics of OFA.

OFA is designed to:

- Organize large amounts of complicated software and data on disk, to avoid device bottlenecks and poor performance
- Facilitate routine administrative tasks such as software and data backup, which are often vulnerable to data corruption
- Facilitate switching between multiple Oracle databases
- Adequately manage and administer database growth
- Help eliminate fragmentation of free space in the data dictionary, isolate other fragmentation, and minimize resource contention

For details on the goals and implementation of OFA, refer to *Oracle Installation Guide for UNIX Systems,* Appendix D.

Using Optimal Flexible Architecture

Naming mount points:

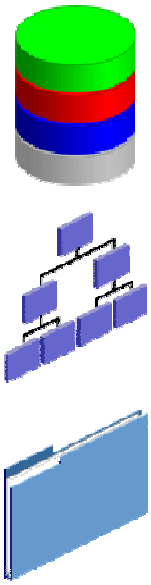
- **`/u01`**
- **`/disk01`**

Naming directories:

- **`/u01/app/oracle`**
- **`/u01/app/applmgr`**

Naming files:

- **Control files: `control`*`n`*`.ctl`**
- **Redo log files: `redo`*`n`*`.log`**
- **Data files: *`tn`*`.dbf`**



Using Optimal Flexible Architecture

At the core of OFA is a naming scheme, which gives you a standard to apply to your mount points (which are often the physical disks), directories and subdirectories on those mount points, and finally the files themselves.

Mount point syntax: Name all mount points using the syntax */*`pm`, where *p* is a string constant and *m* is a unique fixed-length key (typically a two-digit number) used to distinguish each mount point. Example mount points are `/u01`, and `/u02`.

Home directories syntax: Name all home directories using the syntax */*`pm`*/h/u*. Where *pm* is a mount point name, *h* is a standard directory name and *u* is the name of the owner of the directory. Examples of OFA-compliant home directories are:

```
/u01/app/oracle
/u01/home/oracle
```

Software directories syntax: To help fulfill the OFA feature of simultaneously executing multiple versions of application software, store each version of the Oracle software in a directory matching the pattern */*`pm`*/h/u/*`product`*/v*. Here `product` is a literal and the variable *v* is used for the version number. So an OFA-compliant installation of the Oracle Database 10*g* version 10.1.0 would look like:

```
/u01/app/oracle/product/10.1.0
```

Using Optimal Flexible Architecture (continued)

Naming subdirectories syntax: To facilitate the organization of administrative data, you should store database-specific administration files in subdirectories matching the pattern */h/*admin*/d/a/*, where h is the Oracle software owner's home directory, admin is a literal, *d* is the database name, and *a* is a subdirectory for each of the database administration files. Following is a list of these administration file subdirectories:

- adhoc: Ad hoc SQL scripts for a particular database
- arch: Archived redo log files
- adump : Audit files (Set the AUDIT_FILE_DEST initialization parameter to the adump directory. Clean out this subdirectory periodically.)
- Bdump: Background process trace files
- Cdump: Core dump files
- Create: Programs used to create the database
- Exp: Database export files
- Logbook: Files recording the status and history of the database
- Pfile: Instance parameter files
- udump: User SQL trace files

File-naming syntax: The following naming convention for database files ensures that they are easily identifiable:

- Control files: */pm/q/d/*control.ctl
- Redo log files: */pm/q/d/*redo*n*.log
- Data files: */pm/q/d/tn*.dbf

The variables used in these file names are:

- *Pm*: A mount point name as described previously
- *q:* A string distinguishing Oracle data from all other files (commonly named ORACLE or oradata)
- *d:* The value of the initialization parameter DB_NAME (the database name)
- *t:* An Oracle tablespace name
- *n:* A two-digit string

Note: Do not store files other than control files, redo log files, or data files associated with database *d* in the path */pm/q/d/*.

Setting Environment Variables

- **ORACLE_BASE: The base of the Oracle directory structure for OFA**
- **ORACLE_HOME: The directory containing the Oracle software**
- **ORACLE_SID: The initial instance name (ORCL by default)**
- **NLS_LANG: The language, territory and client character set settings**

ORACLE

Setting Environment Variables

There are many Oracle environment variables, and the ones mentioned here are very important to a successful installation and use of an Oracle database. None of these are required to be set, but by setting them prior to the installation you may avoid future problems.

- `ORACLE_BASE`: Specifies the base of the Oracle directory structure for OFA. Use of this is optional, but if used this can facilitate future installations and upgrades. It is a directory path for example:
  ```
  /u01/app/oracle
  ```
- `ORACLE_HOME`: Specifies the directory containing the Oracle software. It is a directory path, for example:
  ```
  $ORACLE_BASE/product/10.1.0
  ```
- `ORACLE_SID`: The initial instance name (`ORCL` by default). It is a string of numbers and letters that must begin with a letter. Oracle Corporation suggests that a maximum of eight characters be used for system identifiers.

Setting Environment Variables (continued)

- NLS_LANG: Specifies the initial NLS settings for a session on the form *language_territory.character set*. For example, a setting of:

```
AMERICAN_DENMARK.WE8MSWIN1252
```

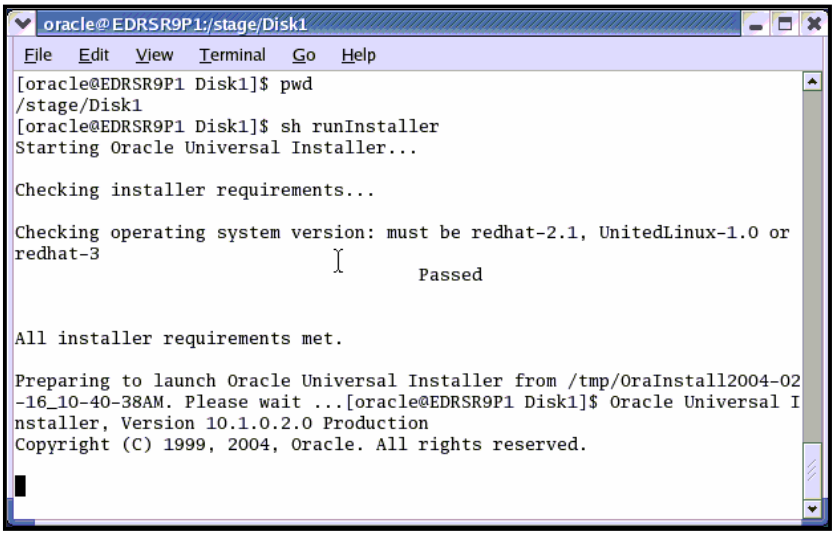
This sets the session to use the language AMERICAN for Oracle messages, sorting, day names, and month names. The territory is DENMARK, which sets the time format, date format, and numeric and monetary conventions. The character set of WE8MSWIN1252 instructs Oracle Net to convert character information to this character set. This is an environment variable in UNIX and a registry setting in Windows. You can query the actual NLS settings of your current session by:

```
select * from nls_session_parameters;
```

Refer to the *Globalization Support Guide* for valid languages, territories, character sets, and more information about language support.

Note: A Windows installation defaults the NLS_LANG values in the registry where the *language* part originates from the keyboard language. This has the effect that default installation on Windows with non-American keyboards will get the non-American value in the NLS_LANG setting. This in turn will default the NLS_SORT session variable to be different from "binary," which makes it difficult for the optimizer to use character-based indexes for sessions from this node.

Preinstallation Checks



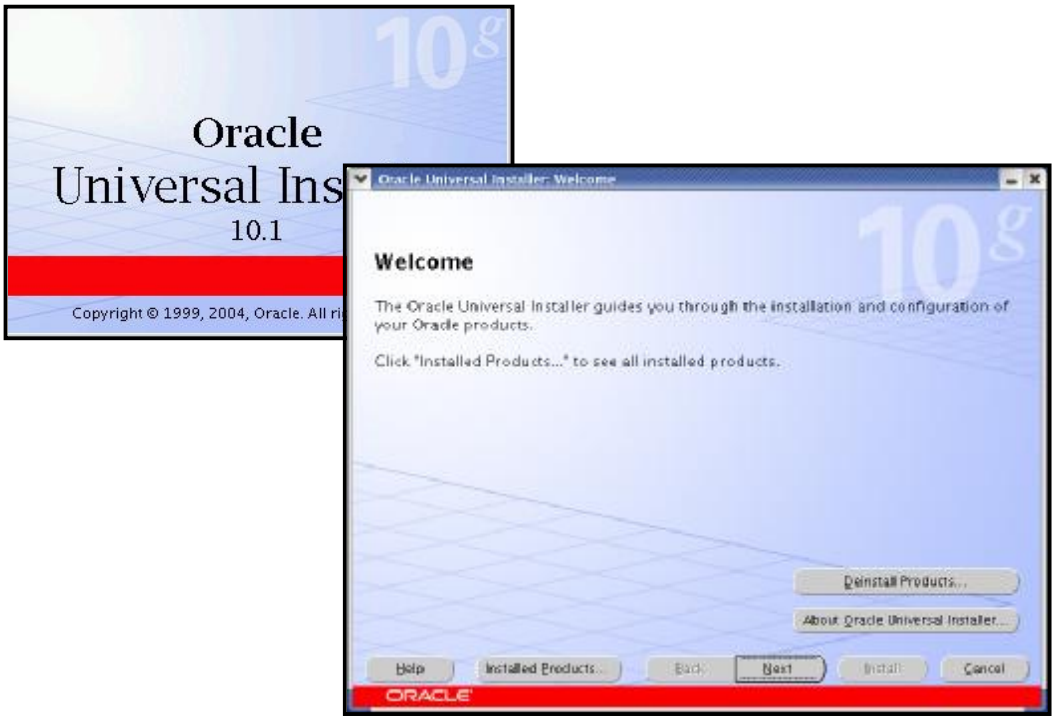
Preinstallation Checks

Before the Oracle Universal Installer GUI interface is launched, there are a number of checks that are performed to verify that minimal requirements are met. In the event that your system doesn't meet one or more of the requirements, you can choose to continue the installation by using the `-ignoreSysPrereqs` flag when running the installer. However, the likelihood of a successful installation is greatly reduced if the requirements are not met. For example:

```
$ sh runInstaller -ignoreSysPrereqs
```

Note: You can use the `-help` flag to see more options for the Oracle Universal Installer.

Oracle Universal Installer



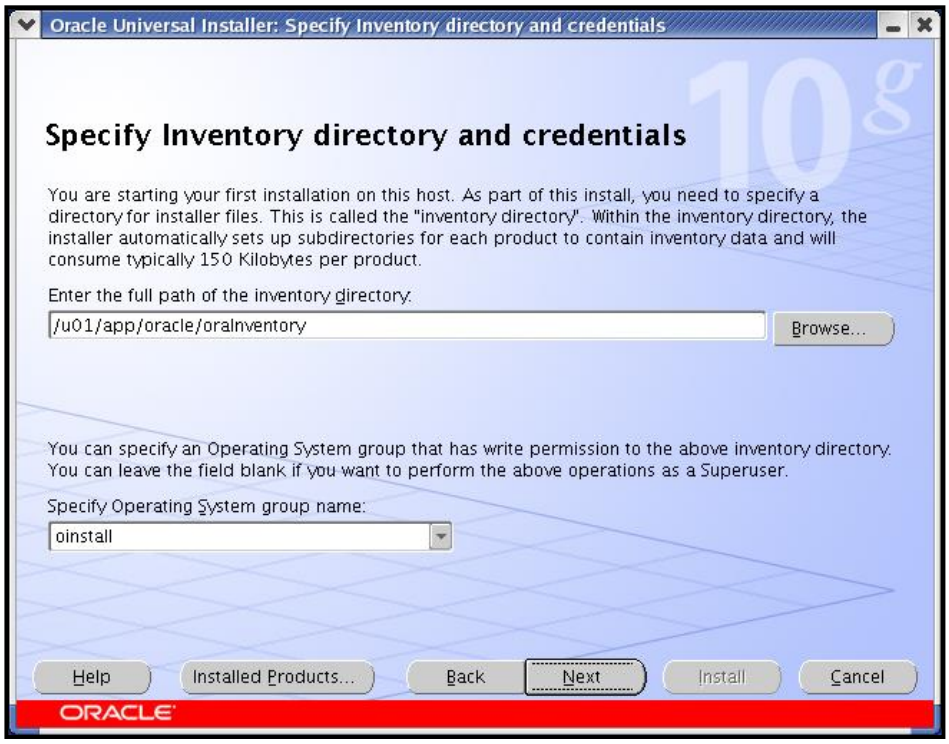
ORACLE

Oracle Universal Installer

Oracle Universal Installer is a Java application that performs component-based installations and enables different levels of integrated bundle, suite, and Web-based installations, as well as complex logic in a single package. The installation engine is easily portable across all Java-enabled platforms, and platform-specific issues can be encapsulated from the overall installation process. The Oracle Universal Installer provides the following capabilities for addressing software management and distribution:

- Automatic dependency resolution and complex logic handling
- Installation from the Web
- Component and suite installations
- Implicit deinstallation
- Support for multiple Oracle homes
- National language support (NLS)/globalization support
- Support for distributed installations
- Unattended "silent" installations using response files

Inventory and UNIX Group Name



Inventory and UNIX Group Name

The inventory directory is an area that is used during the installation of the software. The files in this directory should not be deleted. The Oracle Universal Installer inventory is the location for the Oracle Universal Installer's bookkeeping. The inventory stores information about:

- All Oracle software products installed in all Oracle homes on a machine
- Other non-Oracle products included with Oracle software, such as the Java Runtime Environment (JRE)

You should expect these files to take up about 4 MB of space.

If you are installing a product on a UNIX system, the Installer will also prompt you to provide the name of the group that should own the base directory. You must choose a UNIX group name that will have permissions to update, install, and deinstall Oracle software. Members of this group must have write permissions to the base directory that is chosen. Only users who belong to this group are able to install or deinstall software on this machine.

orainstRoot.sh


```
# sh /u01/app/oracle/oraInventory/orainstRoot.sh
Creating the Oracle inventory pointer file (/etc/oraInst.loc)
Changing groupname of /u01/app/oracle/oraInventory to
oinstall.
```

ORACLE

orainstRoot.sh

When installing on a UNIX platform, you must run the `orainstRoot.sh` script as root. This script creates the inventory pointer file (on Linux this file is `/etc/oraInst.loc`). The inventory pointer file is used by the Oracle Universal Installer at start up to find the inventory location. An example of the file is:

```
inventory_loc=/oracle/oraInventory
inst_group=oinstall
```

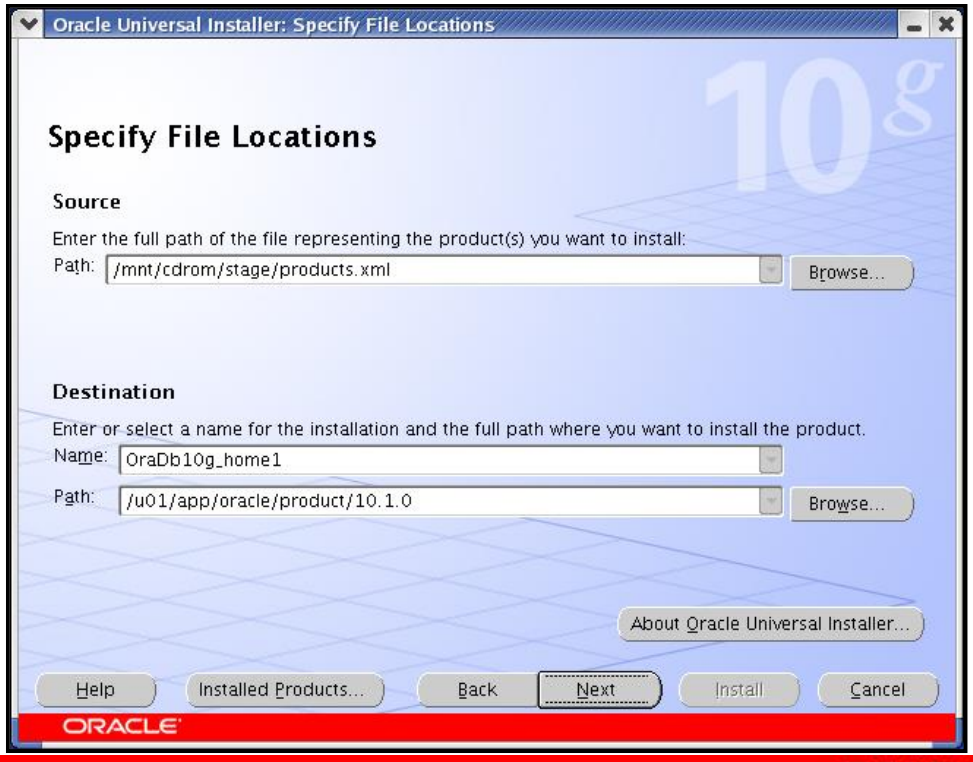
With a Microsoft Windows installation the inventory location is contained within the registry.

orainstRoot.sh (continued)

The oraintRoot.sh script looks like the following:

```
#!/bin/sh
INVPTR=/etc/oraInst.loc
INVLOC=/oracle/oraInventory
GRP=oinstall
PTRDIR="`dirname $INVPTR`";
# Create the software inventory location pointer file
if [ ! -d "$PTRDIR" ]; then
 mkdir -p $PTRDIR;
fi
echo "Creating the Oracle inventory pointer file
($INVPTR)";
echo    inventory_loc=$INVLOC > $INVPTR
echo    inst_group=$GRP >> $INVPTR
chmod 644 $INVPTR
# Create the inventory directory if it doesn't exist
if [ ! -d "$INVLOC" ];then
 echo "Creating the Oracle inventory directory ($INVLOC)";
 mkdir -p $INVLOC;
 chmod 775 $INVLOC;
fi
echo "Changing groupname of $INVLOC to oinstall.";
chgrp oinstall $INVLOC;
if [ $? != 0 ]; then
 echo "WARNING: chgrp of $INVLOC to oinstall failed!";
fi
```

File Locations



File Locations

The Universal Installer provides a default value on the Specify File Locations page that points to the location of the product's installation kit or stage.

Oracle homes are identified by name. The Oracle home name identifies the program group associated with a particular Oracle home and the Oracle services installed on the associated home. The Oracle home name must be 1 to 16 characters long and can include only alphanumeric characters and underscores, and must not include spaces.

Oracle Universal Installer maintains the following Oracle homes on Windows platforms:

- All Oracle homes that are already created using the Oracle Universal Installer
- All homes created using the previous Oracle Universal Installer (ORCA-based)
- The home saved in the `ORACLE_HOME` registry entry.

Oracle Universal Installer maintains the following Oracle homes on UNIX:

- All Oracle homes that are already created using the Oracle Universal Installer
- All homes as defined in the `/var/opt/oratab` file (In Linux, `/etc/oratab`)
- The home saved in the `ORACLE_HOME` environment variable

If none of these homes exists, a default home is calculated based on the disk volume with the largest amount of free space.

Install Type

Install Type

Oracle Database Enterprise Edition offers industry-leading scalability and reliability in both clustered and single system configurations. It provides the most comprehensive features for online transaction processing and business intelligence.

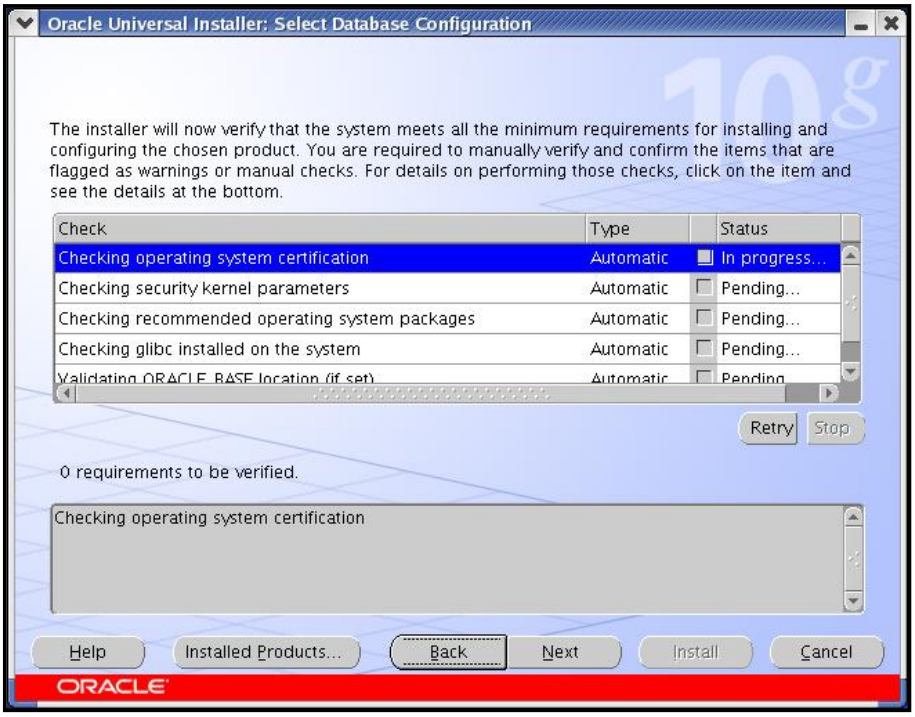
Oracle Standard Edition is for small and medium business, or departmental applications that demand all the power, reliability, and security of Oracle without all the Enterprise options.

The following enterprise-level features are some of the options not available with the Oracle Standard Edition:

- Oracle Data Guard: A comprehensive set of services that create, manage, and monitor one or more standby databases.
- Oracle Advanced Security, Oracle Label Security, Enterprise User Security, Virtual Private Database, N-tier authentication, and Fine Grained Auditing: These options extend the security features of the Oracle Database 10*g*.
- Oracle Partitioning, Oracle OLAP, Export Transportable Tablespace, and other features that support Enterprise Data Warehouses.

The custom install allows you to pick individual components to install.

Prerequisite Checks

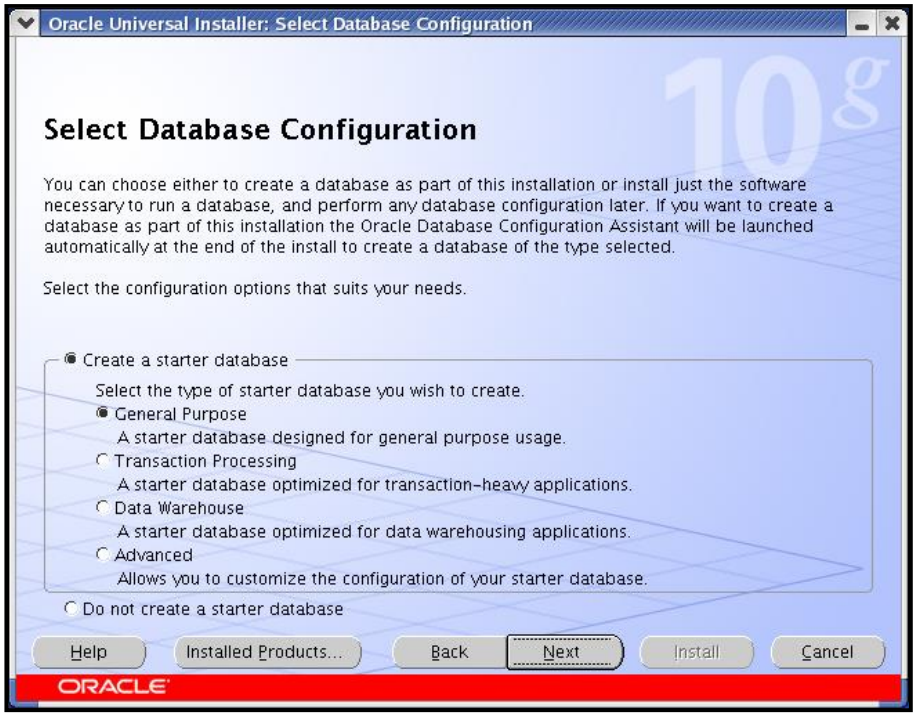


Prerequisite Checks

Before installation, the Oracle Universal Installer checks the environment for requirements necessary for a successful installation. This page show the name, type and status for all prerequisite checks. Automatic checks are run first, if there are optional checks you can run them once the automatic checks are compete.

Once all checks are complete, there will be a summary of the checks done on the lower part of the page. A check may require verification if a warning was produced during the execution or a manual check was not confirmed. To verify a warning or manual check, click the check box for the check that produced a warning or is a manual check.

Starter Database

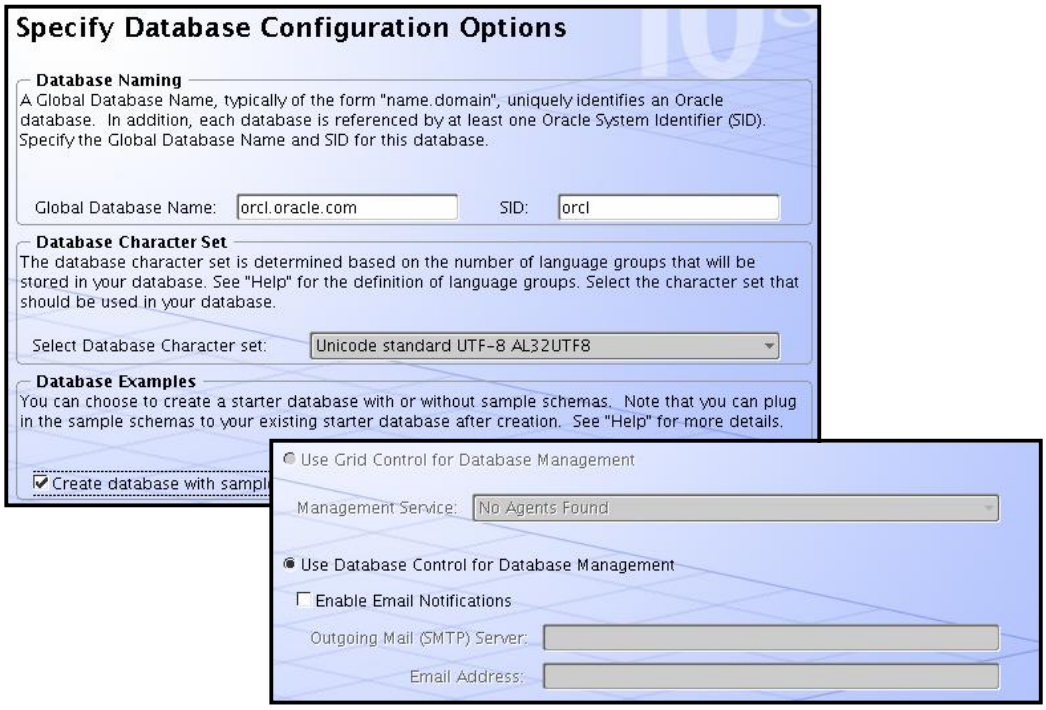


Starter Database

There are three template database from which you can choose, or you can set up a custom starter database. The template databases use pre configured files and create a database quickly, although with these you can customize the database. The advanced option allows you total flexibility in the creation of the started database, but does take longer to build.

Note: If you choose not to create a starter database, the summary page appears next.

Configuration and Management

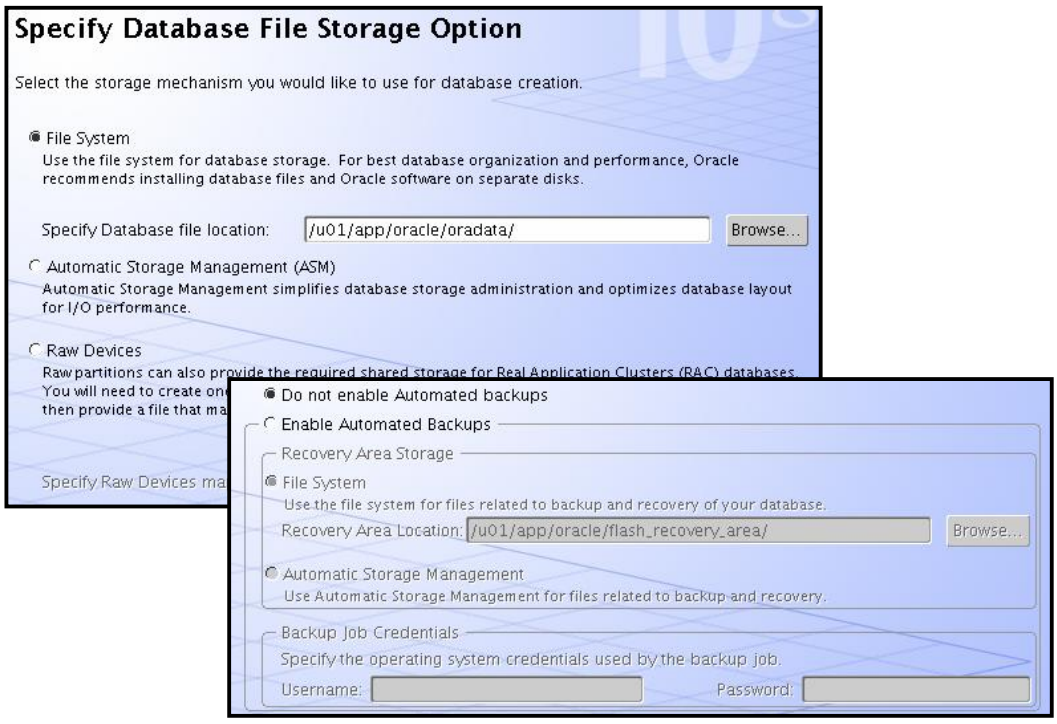


Configuration and Management

If you select to create a starter database as part of your install, you will go through a series of pages asking you for information to configure the starter database.

- Database Naming: Here you name the database; the default is `orcl`.
- Database Character Set: This is the character set used in the database to store your data. You should take some time to determine what is the base character set for your data. Although it is possible to change a character set later, it can be a very time-consuming task, and will require some down time for your database. If unsure about which character set to use and you know you will be using multiple languages, the Unicode Standard UFT-8 AL32UTF8 is a good choice. The WE8ISO8859P1 character set doesn't have support for the euro symbol (€) , but WE8ISO8859P15 does. For more details on selecting the appropriate character set, see the *Oracle Database Globalization Support Guide*.
- Sample Schemas: The sample schemas are a set of schemas used for demonstrations and training.
- Database Management: Grid control is used to manage many databases either on one machine or several machines. Database control is used to manage a single database.

File Storage and Backup Recovery



File Storage and Backup Recovery

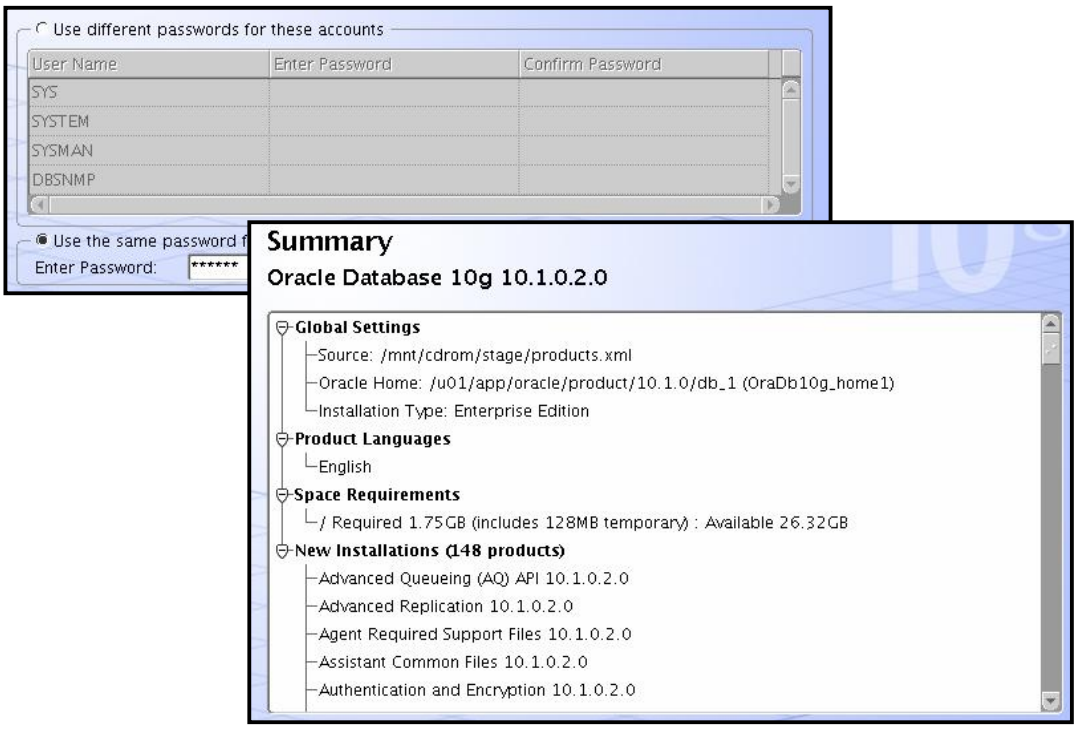
File storage options:

- File System: This specifies where to store files in your OS configured file system.
- Automatic Storage Management (ASM): Automatic Storage Management files are created and managed automatically, and you get the additional benefits of features such as mirroring and striping. For details on how to set up ASM, see the *Oracle Database Administrator's Guide*.
- Raw Devices (Partitions): These are disk partitions without a file system on them. Generally you should use these only if you are very familiar with the use of raw partitions already. Check your OS documentation for details on setting up and maintaining raw partitions.

Backup and Recovery Options:

- Do not enable Automated backups: Eventually you will need to set up a backup plan. If you select not to set it up now you can do so later.
- Enable Automated Backups: When you enable Automated Backups, you need to specify where the backups will be stored, and information about the backup job used for the backups.

Passwords and Summary

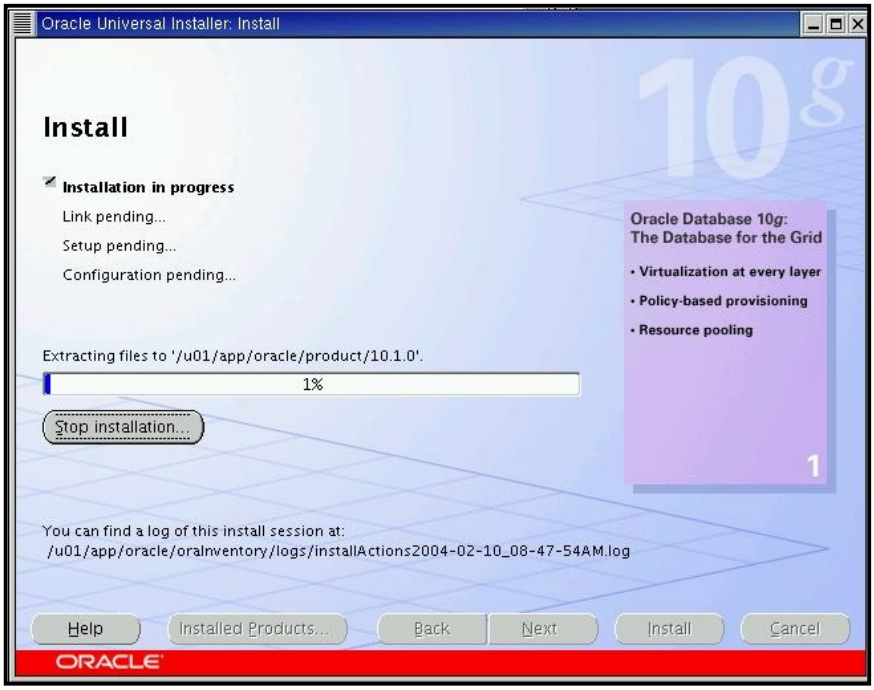


ORACLE

Passwords and Summary

- Database Schema Passwords: Provide passwords for the administrative users SYS, SYSTEM, SYSMAN, and DBSNMP. You can provide a password for each one separately or provide one password for all.
- Summary: Here you see all the products and settings you selected for the installation. The next step is to initiate the installation. Check the summary then click Install to start the actual installation.

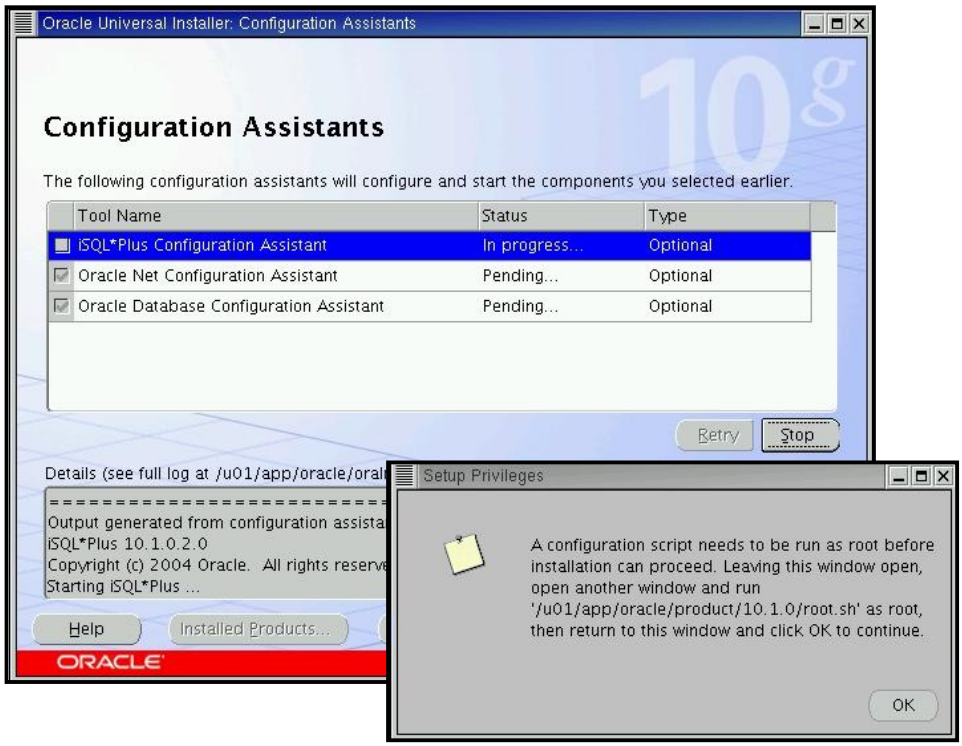
Installation



Installation

During the installation you can watch the progression of the installation.

Configuration Assistants



Configuration Assistants

If you have selected to create a starter database with the install, the configuration assistants page will be displayed at the completion of the software install.

- *i*SQLPlus Configuration Assistant: This configures the OC4J (Oracle Containers for Java) instance which is used by *i*SQLPlus and other tools to connect to the Oracle database.
- Oracle Net Configuration Assistant: This configures basic network components during installation, including (these components are covered in detail in later lessons):
  - Listener names and protocol addresses
  - Naming methods the client will use to resolve connect identifiers to connect descriptors
  - Net service names in a tnsnames.ora file
  - Directory server usage
- Oracle Database Configuration Assistant: This creates the starter database you selected. It uses the selection you made to create the database. In the next lesson you will learn more about Oracle Database Configuration Assistant. When this configuration assistant finishes you will have the opportunity to unlock accounts and change passwords.

Configuration Assistants (continued)

When installing on a UNIX (or Linux) OS, or you did not chose to create a started database, you will be prompted to run one more script as root. The script is named `root.sh`. This script updates or creates a file that contains information about `ORACLE_HOME` locations and databases. In a Linux installation the file is named `/etc/oratab`.

With an installation on Microsoft Windows OS, there is no `root.sh` script because the information about `ORACLE_HOME` and databases is stored in the registry. The `oratab` file is the file where the user places references to all databases to be discovered by the Agent and controlled by Oracle Enterprise Manager.

Summary

In this lesson, you should have learned how to:

- **Identify system requirements**
- **Use Optimal Flexible Architecture**
- **Install software with Oracle Universal Installer**

Practice 2:
Installing the Oracle Software

This practice exercise covers installing the Oracle software.

Practice 2 – Install the Oracle Software

During the practices of this course you will assume the role of a new DBA who will be supporting a database which is used for a Human Resources application. You work for an IT manager; your coworker is an Oracle Certified Professional.

The OS accounts on your machine are:

- The `oracle` user with a password of `oracle`.
- The `root` user with a password of `oracle`.

The system administrator has set up the OS so that it is ready for the install and the install media is staged at `/stage/Disk1`. Install the software as the user `oracle`.

- Set the global name to `orcl.oracle.com` and the SID to `orcl`
- Use `AL32UTF8` for the database character set
- Create the database with sample schemas
- Make sure `oinstall` is the selected group name on the Unix Group Name screen
- Accept the defaults on the Specify File Locations page
- Create a database using the General Purpose template
- Use `oracle` as the password for all the administration accounts (`sys`, `system`, `sysman` and `dbsnmp`)
- Unlock the HR schema

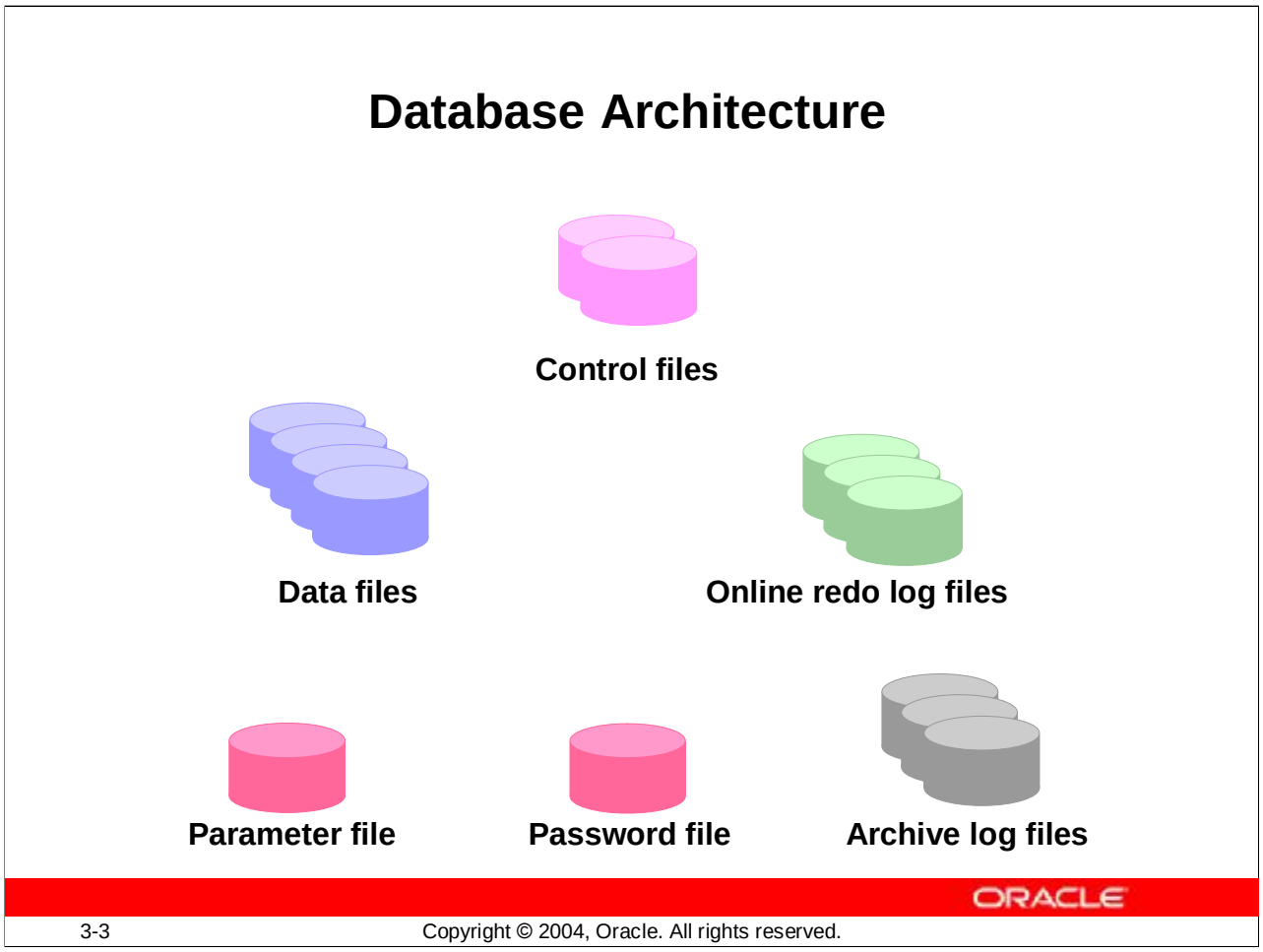
Creating an Oracle Database

Objectives

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

- **Describe the Oracle database architecture**
- **Understand the instance architecture**
- **Use the management framework**
- **Use DBCA to**
  - **Create a database**
  - **Configure a database**
  - **Drop a database**
  - **Manage templates**

ORACLE



Database Architecture

The files that constitute an Oracle database are organized into the following:

- Control files: Contain data about the database itself, called the metadata. These files are critical to the database. Without them you cannot open the data files to access the data within the database.
- Data files: Contain the data of the database
- Online redo log files: Allow for instance recovery of the database. If the database crashes and does not lose any data files, the instance can recover the database with the information in these files.

There are other files which are not officially part of the database but are important to the successful running of the database. These are:

- Parameter file: Used to define how the instance will be configured when it starts up
- Password file: Allows users to connect remotely to the database and perform administrative tasks
- Archive log files: Contain an ongoing history of the redo generated by the instance. These files allow for database recovery; using these files and a backup of the database, you can recover a lost data file.

Exploring the Storage Structure

ORACLE

Exploring the Storage Structure

Logical data structures are stored in the physical files of the database. You can easily view the logical structures of your database through Enterprise Manager. Detailed information on each structure can be obtained by clicking the links in the Storage region of the Administration page.

Control Files

- **Contain physical database structure information**
- **Multiplexed to protect against loss**
- **Required to start the instance**

Control files

Control Files

When you start the instance and mount the database, the control file is read. The entries in the control file specify the physical files that the database comprises.

When you add additional files to your database, the control file is automatically updated.

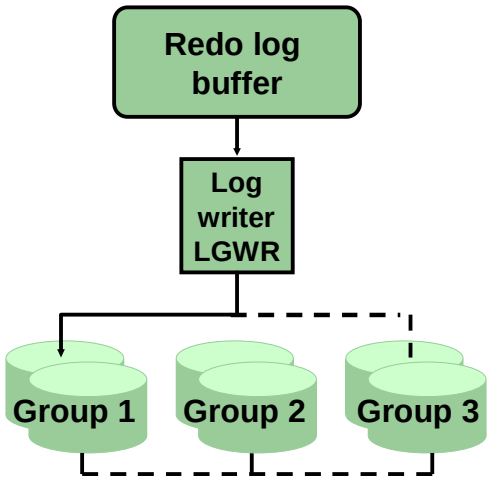
The location of the control files is specified in an initialization parameter.

To protect against failure of the database due to the loss of the control file, you should multiplex the control file on at least three different physical devices. By specifying multiple files through the initialization parameter you enable the Oracle database server to maintain multiple copies of the control file.

You can access information about the control files in your database by clicking the Controlfiles link in the Storage region of the Administration page in Enterprise Manager. The Controlfiles General page displays the name and location of the control files in your database. The Advanced page provides information about the creation of the control file and database identification. The Record Section page displays information about the entries in the control file.

Redo Log Files

- Record changes to the database
- Multiplex to protect against loss



Redo Log Files

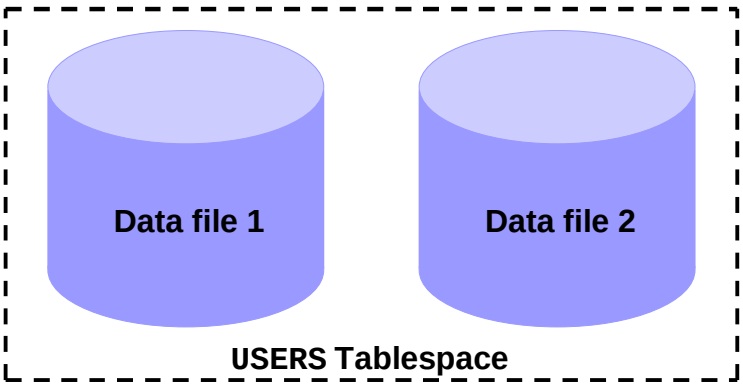
You use redo log files to record changes to the database as a result of transactions and internal Oracle database server actions. They protect the database from loss of integrity due to system failures caused by power outages, disk failures, and so on. Redo log files should be multiplexed to ensure that the information stored in them is not lost in the event of a disk failure.

The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group and each group is identified by a number. The log writer (LGWR) process writes redo records from the redo log buffer to a redo log group until the file is filled or a log switch operation is requested. Then it switches and writes to the files in the next group. The redo log groups are used in a circular fashion.

You can access information about the redo log files in your database by clicking the Redo Log Groups link in the Storage region of the Administration page. You can view detailed information, including the names of the redo log files, by selecting a group and clicking View.

Tablespaces and Datafiles

- Tablespaces consist of one or more data files
- Data files belong to only one tablespace



ORACLE

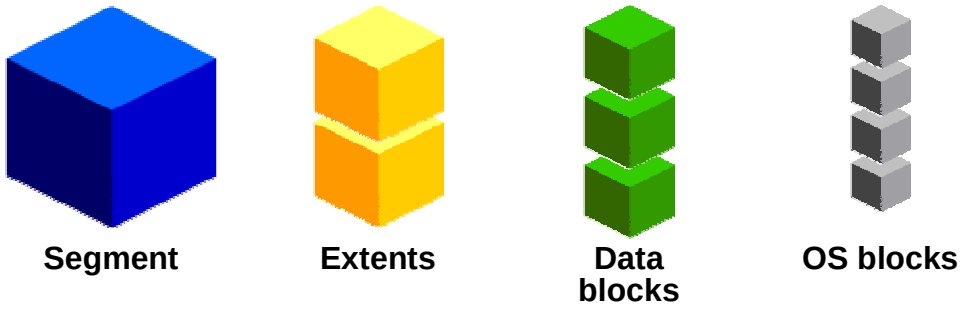
Tablespaces and Datafiles

A database is divided into logical storage units called tablespaces, which can be used to group related logical structures together. Each database is logically divided into one or more tablespaces. One or more data files are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace.

Note: You can also create `bigfile` tablespaces, which are tablespaces with a single, but very large (up to 4G blocks) data file. Traditional `smallfile` tablespaces (which are the default), can contain multiple data files, but the files cannot be as large. For more information about `bigfile` tablespaces see the *Database Administrator's Guide*.

Segments, Extents, and Blocks

- Segments exist within a tablespace.
- Segments are made of a collection of extents.
- Extents are a collection of data blocks.
- Data blocks are mapped to OS blocks.



ORACLE

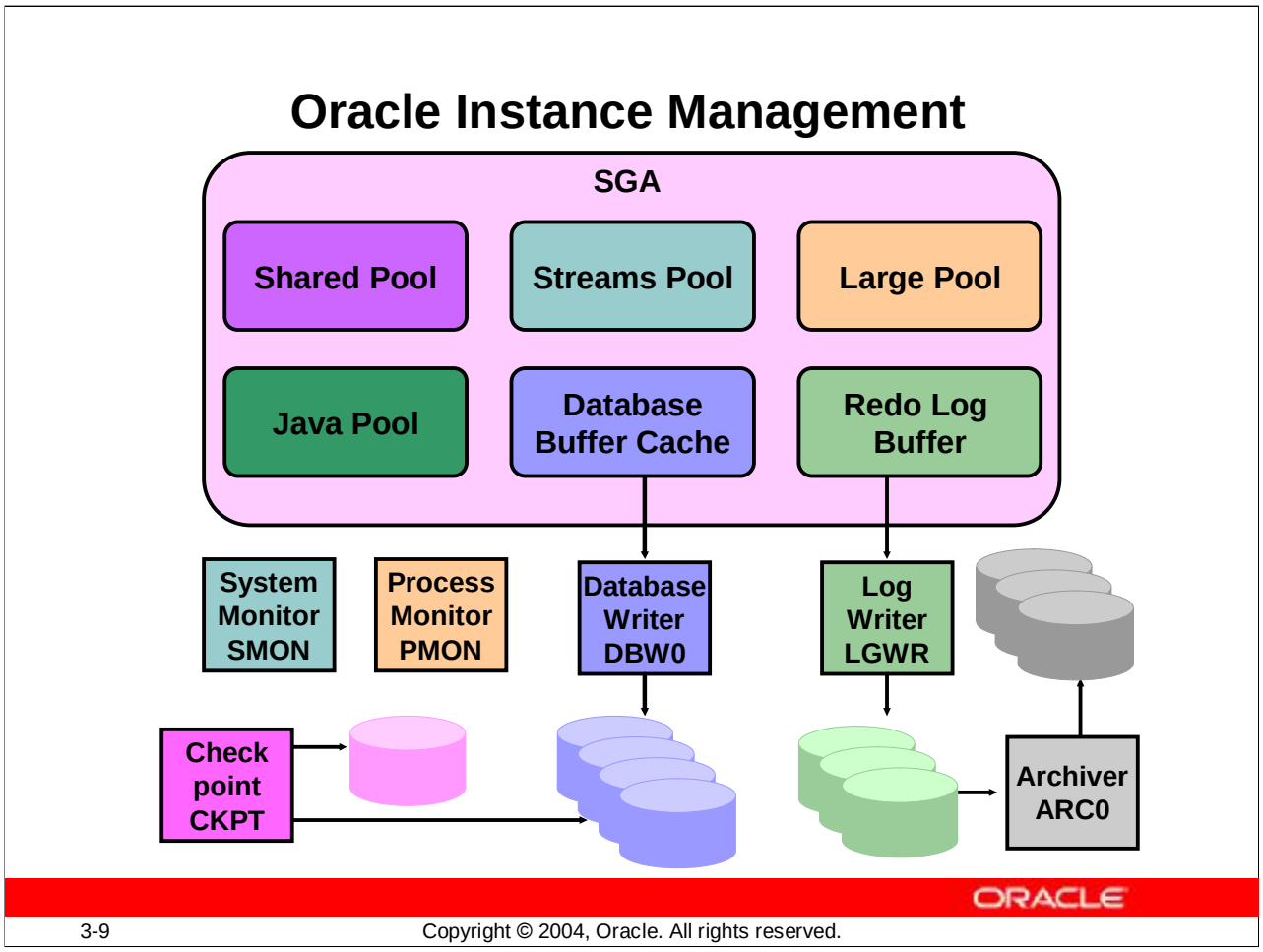
Segments, Extents, and Blocks

Database objects such as tables and indexes are stored in tablespaces as segments. Each segment contains one or more extents. An extent consists of contiguous data blocks, which means that each extent can exist only in one data file. Data blocks are the smallest unit of I/O in the database.

When the database requests a set of data blocks from the OS, the OS maps this to an actual OS block on the storage device. Because of this, you need not be aware of the physical address of any of the data in your database. This also means that a data file can be striped and or mirrored on several disks.

The size of the data block can be set at database creation time. The default size of 8 K is adequate for most databases. If your database supports a data warehouse application that has large tables and indexes, then a larger block may be beneficial. If your database supports a transactional application where reads and write are very random, then specifying a smaller block size may be beneficial. The maximum block size depends on your OS. The minimum block size is 2 K and should rarely (if ever) be used.

You can have tablespaces with different block sizes. Generally this should be used only for support of transportable tablespaces. For details see the *Database Administrator's Guide*.

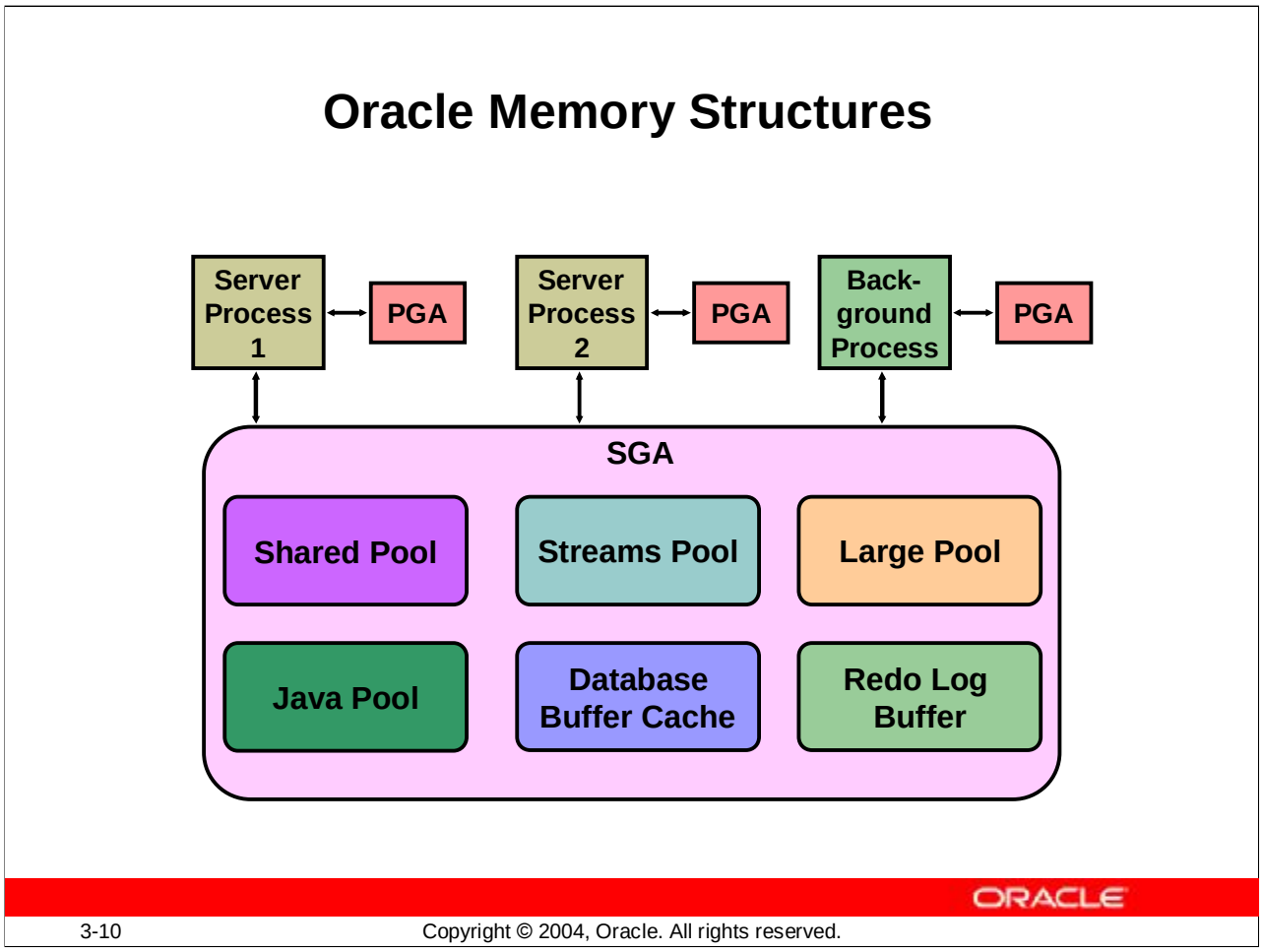


Oracle Instance Management

An Oracle database server consists of an Oracle database and an Oracle instance. An Oracle instance is made up of memory buffers known as the System Global Area (SGA) and background processes that handle much of the behind-the-scenes work involved in running an instance.

The instance is idle (nonexistent) until it is started. When the instance is started, an initialization parameter file is read and the instance is configured according to instructions contained within the parameter file.

After the instance is started and the database is opened, users can access the database.



Oracle Memory Structures

The basic memory structures associated with an Oracle instance include:

- System Global Area (SGA): Shared by all server and background processes
- Program Global Area (PGA): Private to each server and background process; there is one PGA for each process.

The System Global Area (SGA) is a shared memory area that contains data and control information for the instance.

The SGA includes the following data structures:

- Database buffer cache: Caches blocks of data retrieved from the database
- Redo log buffer: Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on disk
- Shared pool: Caches various constructs that can be shared among users
- Large pool: Optional area used for buffering large I/O requests
- Java pool: Used for all session-specific Java code and data within the Java Virtual Machine (JVM)
- Streams pool: Used by Oracle Streams

When you start the instance using Enterprise Manager or SQL*Plus, the memory allocated for the SGA is displayed.

Oracle Memory Structures (continued)

With the dynamic SGA infrastructure, the size of the database buffer cache, the shared pool, the large pool, the Java pool, and the Streams pool changes without shutting down the instance.

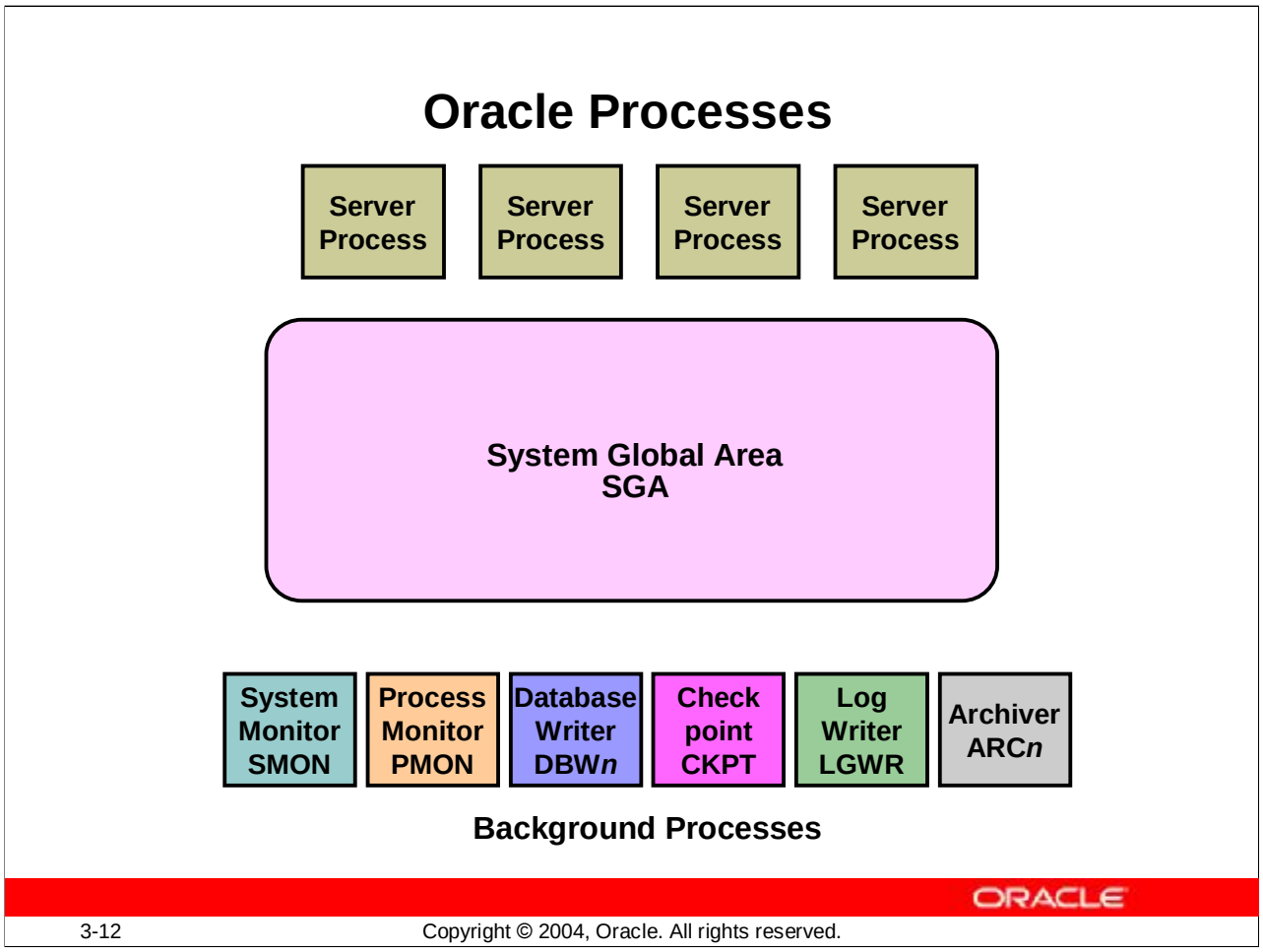
The preconfigured database has been pretuned with adequate settings for the memory parameters. However, as your database usage expands you may find it necessary to alter the settings of the memory parameters.

Oracle provides alerts and advisors to identify memory sizing problems and to help you determine appropriate values for memory parameters.

A Program Global Area (PGA) is a memory region which contains data and control information for each server process. A server process is a process that services a client's requests. Each server process has its own private PGA area that is created when the server process is started. Access to it is exclusive to that server process, and is read and written only by Oracle code acting on behalf of it.

The amount of PGA memory used and its content depends on whether the instance is configured in shared server mode (shared server will be discussed in lesson 13). Generally, the PGA contains the following:

- Private SQL area: Contains data such as bind information and run-time memory structures. Each session that issues a SQL statement has a private SQL area.
- Session memory: Memory allocated to hold session variables and other information related to the session.



Oracle Processes

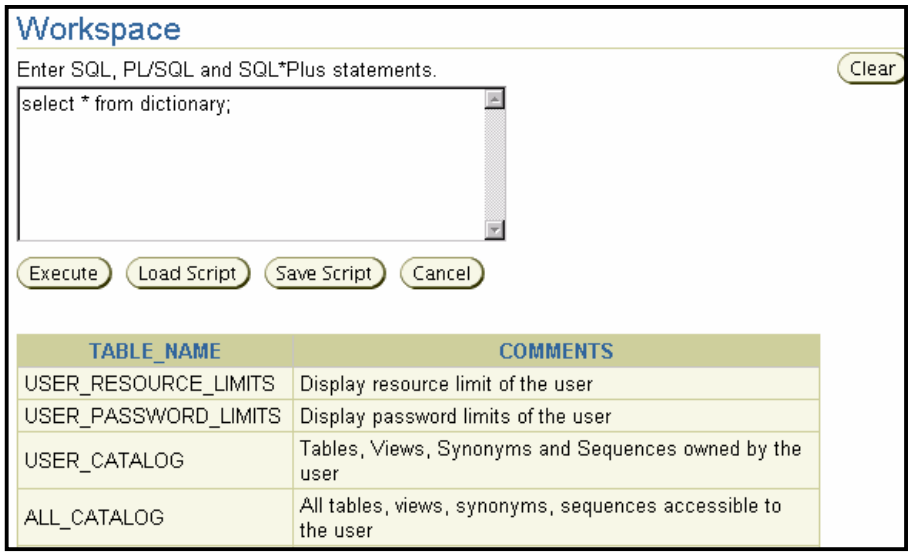
When you invoke an application program or an Oracle tool such as Enterprise Manager, the Oracle server creates a server process to execute commands issued by the application.

Oracle also creates a set of background processes for an instance that interact with each other and with the operating system to manage the memory structures, asynchronously perform I/O to write data to disk, and do general housekeeping.

Which background processes are present depends upon the features that are being used in the database. The most common background processes are the following:

- System monitor (SMON): Performs crash recovery when the instance is started following a failure
- Process monitor (PMON): Performs process cleanup when a user process fails.
- Database writer (DBW*n*): Writes modified blocks from the database buffer cache to the files on disk
- Checkpoint (CKPT): Signals DBW*n* at checkpoints and updates all of the data files and control files of the database to indicate the most recent checkpoint
- Log writer (LGWR): Writes redo log entries to disk
- Archiver (ARC*n*): Copies the redo log files to archival storage when the log files are full or a log switch occurs

Data Dictionary



ORACLE

Data Dictionary

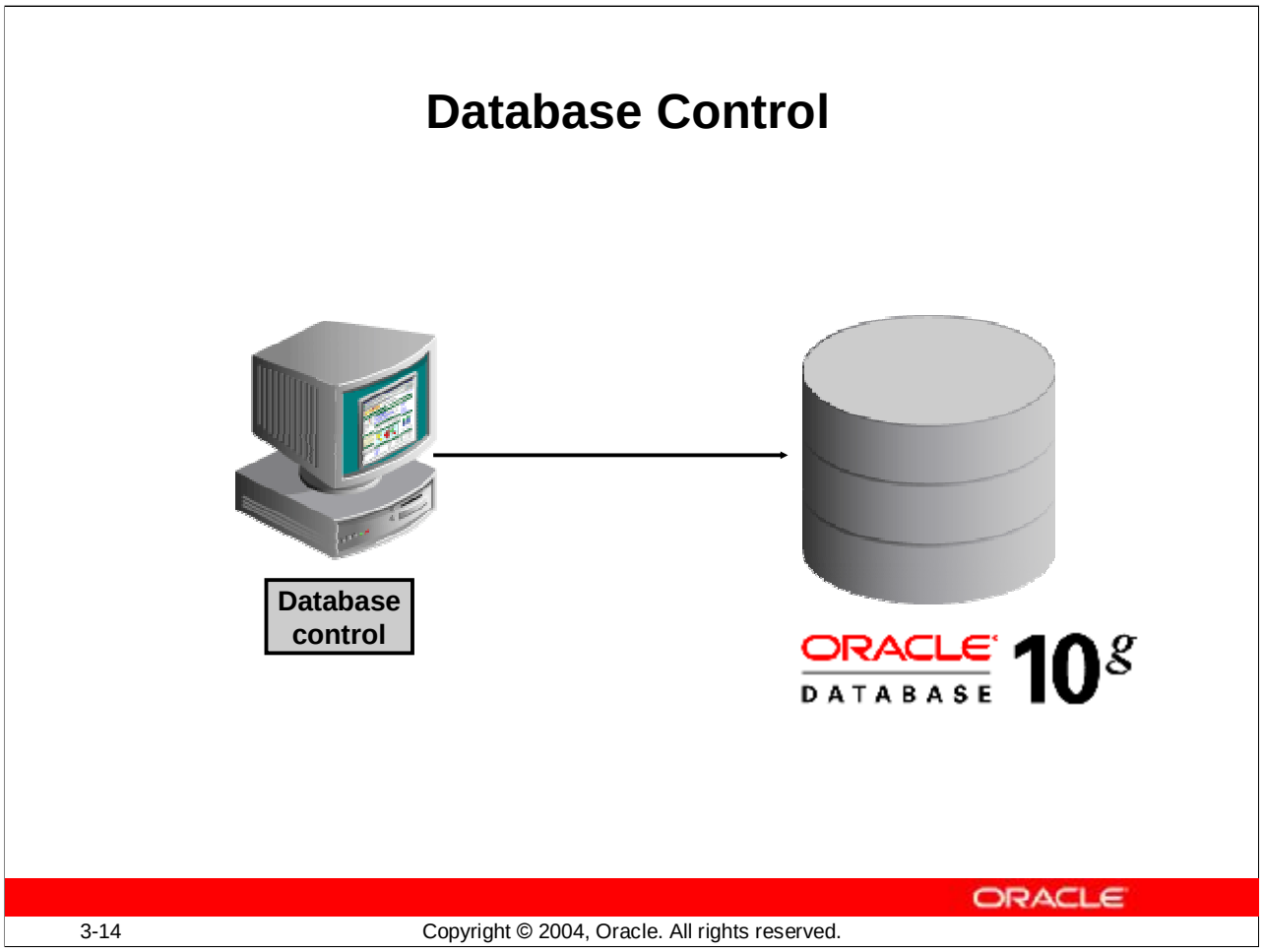
The data dictionary is the central set of tables and views that are used as a read-only reference about a particular database. A data dictionary stores information such as:

- The logical and physical structure of the database
- Valid users of the database
- Information about integrity constraints
- How much space is allocated for a schema object and how much of it is in use

A data dictionary is created when a database is created and is automatically updated when the structure of the database is updated.

The data dictionary is where Enterprise Manager retrieves information about objects in the database. You can also select information from the data dictionary tables. Enterprise Manager does this for you and presents the information in a very easy-to-use format. The `DICTIONARY` view contains descriptions of data dictionary tables and views. These tables and views generally have one of three prefixes:

- `USER`: Information pertaining to the objects owned by the current user.
- `ALL`: Information pertaining to the objects accessible to the current user.
- `DBA`: Information pertaining to every object in the database.



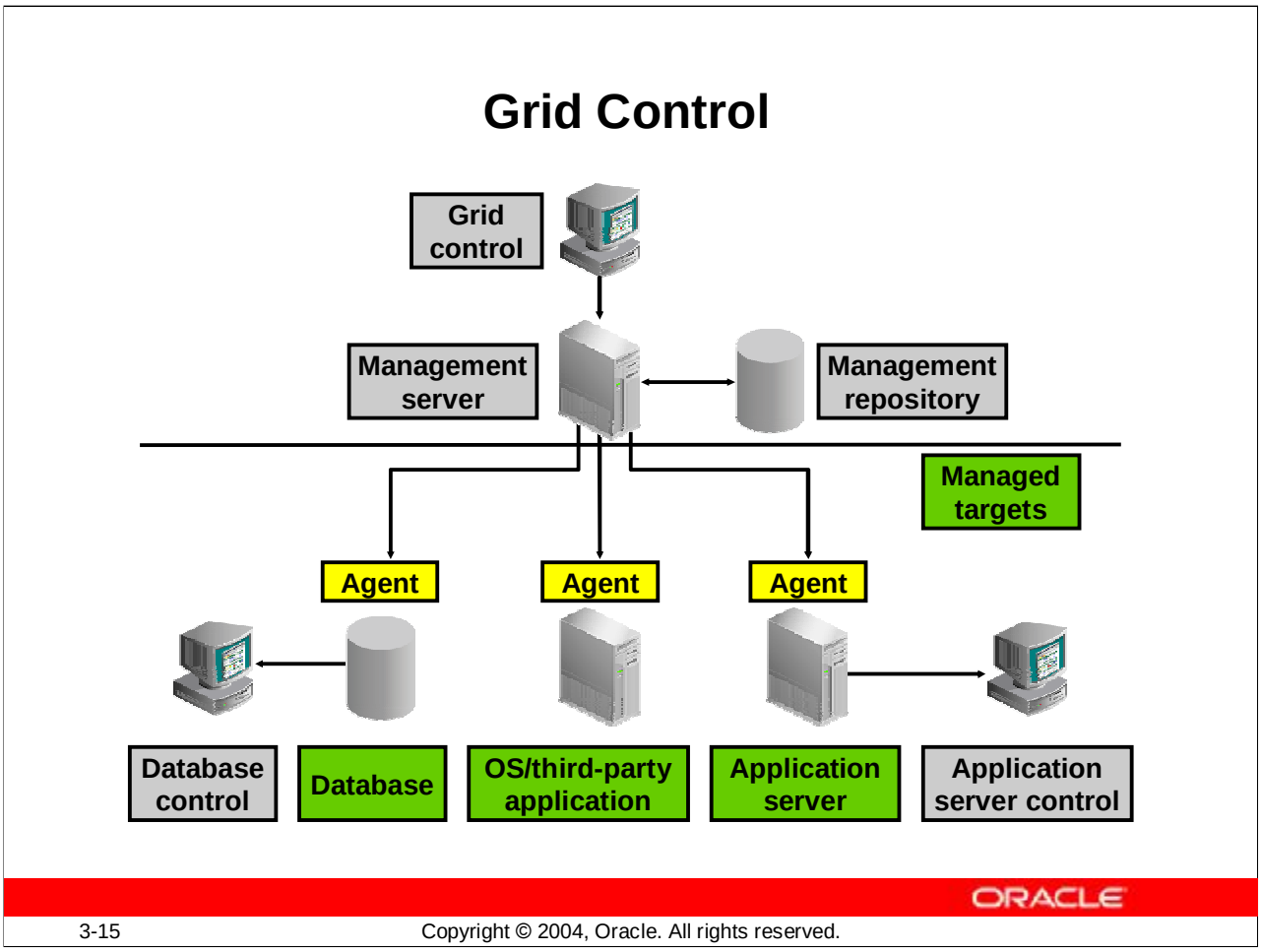
Database Control

Oracle Database 10*g* ships with Oracle Enterprise Manager's Database Control. Database Control is a web-enabled control console that the database administrator can use for:

- Performance monitoring
- Managing proactive alerts
- Controlling maintenance wizards and advisors
- User and database object administration
- Database backup and recovery
- Storage management

and much more.

Each Oracle Database 10*g* you create will have its own Database Control. You will be using Enterprise Manager Database Control in this course to manage the database on your classroom PC.



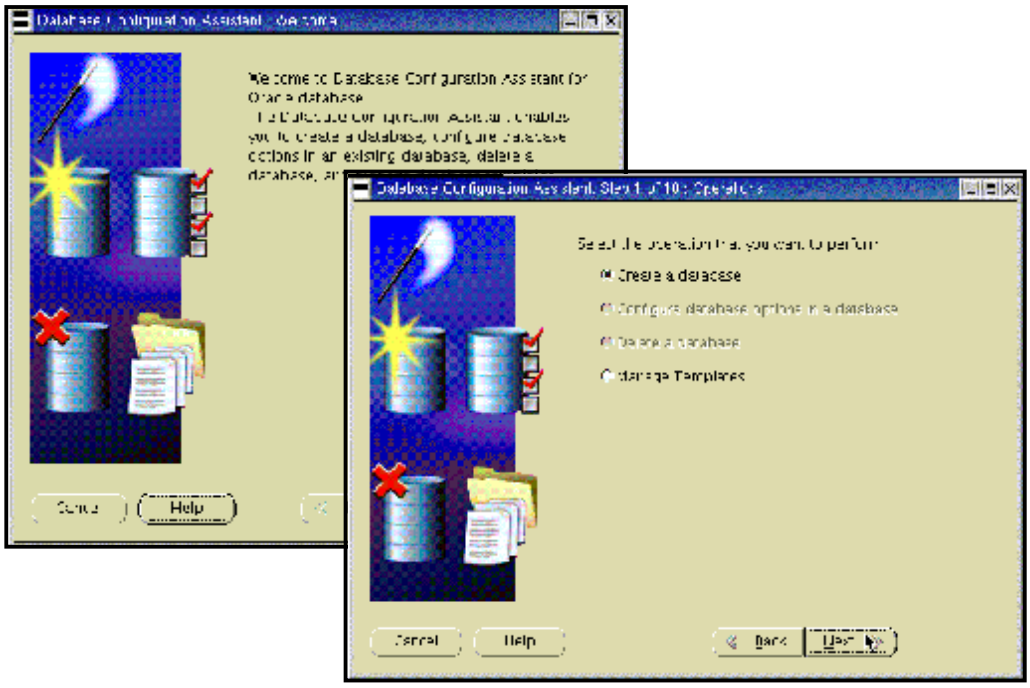
Grid Control

Database Control's capabilities can be extended and integrated with the rest of your systems using Oracle Enterprise Manager's Grid Control. The architecture of the Grid Control framework provides a high level of flexibility and functionality. You can easily customize Enterprise Manager to suit the monitoring and administrative needs of your environment.

The typical Enterprise Manager framework configuration consists of the following functional areas:

- Managed targets
- Management services
- Web-based grid control
- Database control
- Application server control

Database Configuration Assistant (DBCA) Overview



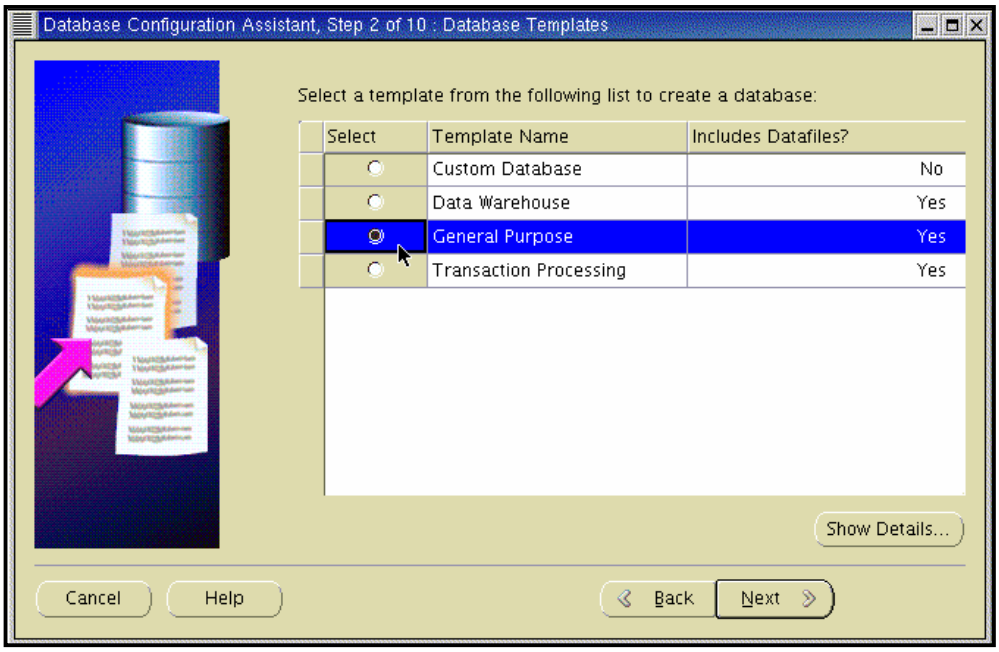
ORACLE

Database Configuration Assistant Overview (DBCA)

Database Configuration Assistant (DBCA) enables you to create, change the configuration of, or delete a database. You can also create a database from a list of predefined templates or use an existing database as a sample to create a new database or template. A template is a predefined database that you use as a starting point for a new database.

- Create a database: If you select this option, you can create a new database or template.
- Configure options in a database: If you select the "Change database configuration" option, you can configure installed options that have not previously been configured for use with your database. You can also enable or disable shared server support.
  Note: The "Change database configuration" option is not available for Oracle Real Application Clusters.
- Delete a database: If you select this option, you can delete all the database files.
- Manage Templates: If you select this option, you have three ways to create a template:
  - From an existing template
  - From an existing database (structure only)
  - From an existing database (structure as well as data)

Creating a Database



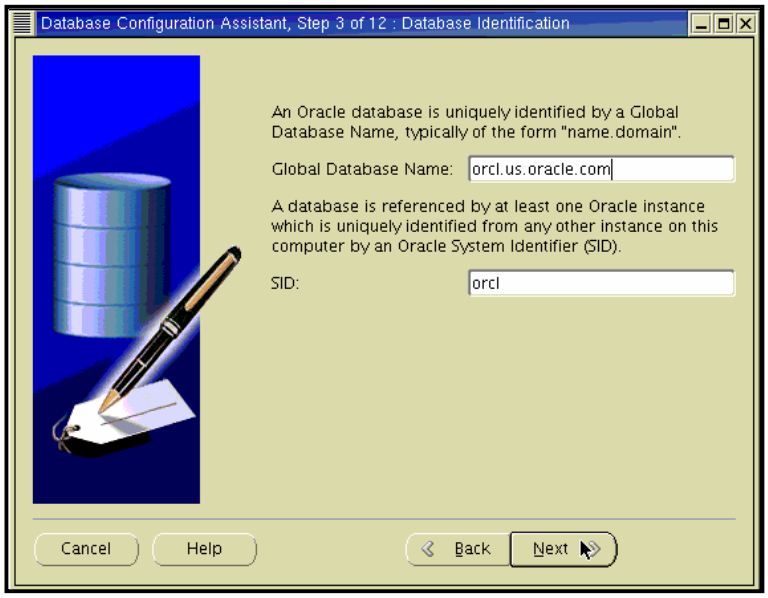
ORACLE

Creating a Database

When creating a database with DBCA, you can select one of three predefined databases, or create a custom database. Oracle Corporation ships predefined templates. There are templates for data warehouse, general purpose, and transaction processing databases. The templates contain settings optimized for workload. Click Show Details to see the configuration for each type of database. Choose the template suited to the type of workload your database will support. If you are not sure, select the default General Purpose template.

For more complex environments, you may want to select the Custom Database option. This results in a more extensive interview and takes longer to create your database, because a database creation script must be run.

Database Identification

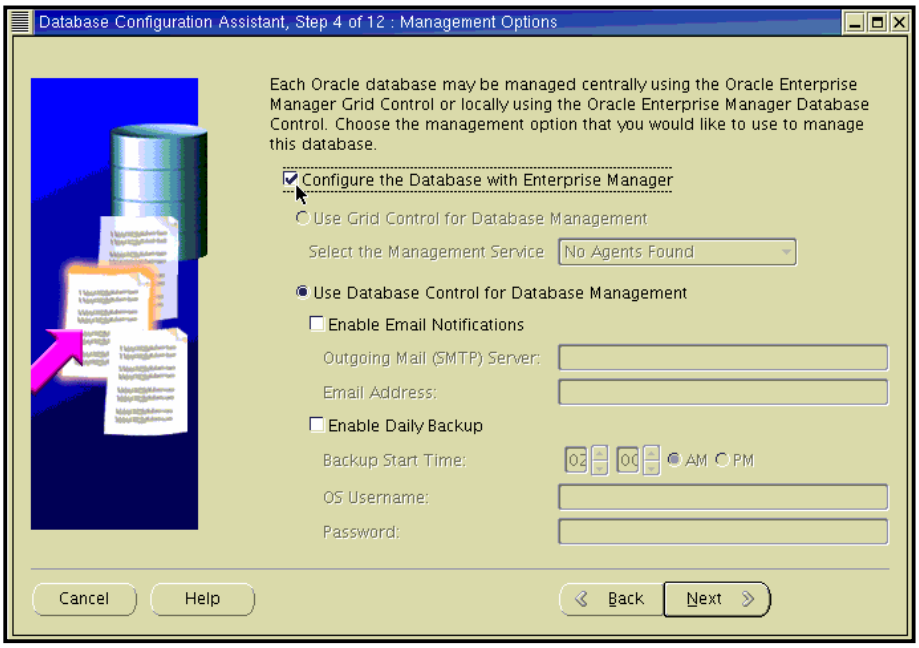


ORACLE

Database Identification

Enter the Global Database Name, in the form `database_name.domain_name` and SID (Oracle system identifier). The SID defaults to the database name and uniquely identifies the instance that runs the database. It is important to understand that the SID is the name of the instance that will connect to a database and not necessarily the name of the database. An instance and the database the instance connects to need not have the same name, although it is convenient. With Real Application Clusters, multiple instances open the same database and the SIDs for each instance will be different.

Management Options

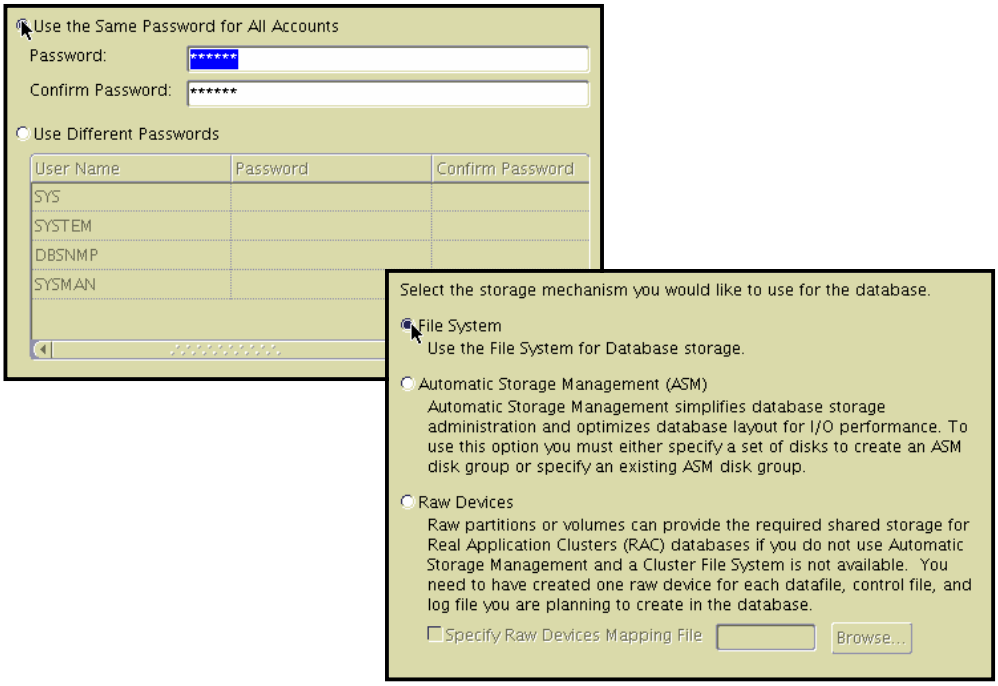


Management Options

Use this page to set up your database so it can be managed with Oracle Enterprise Manager, which provides web-based management tools for individual databases, as well as central management tools for managing your entire Oracle environment. To use Enterprise Manager, select Configure the Database with Enterprise Manager.

If the Oracle Management Agent has been installed on your host computer, then you are given the option of selecting central management by selecting Use Grid Control for Database Management. If you select this type of management, you must also indicate which management service to use in the drop-down menu. Otherwise, select Use Database Control for Database Management to manage your database individually. If you choose this option, you can additionally enable Email Notifications and Enable Daily Backup. Click Help for more information about these options.

Passwords and Storage



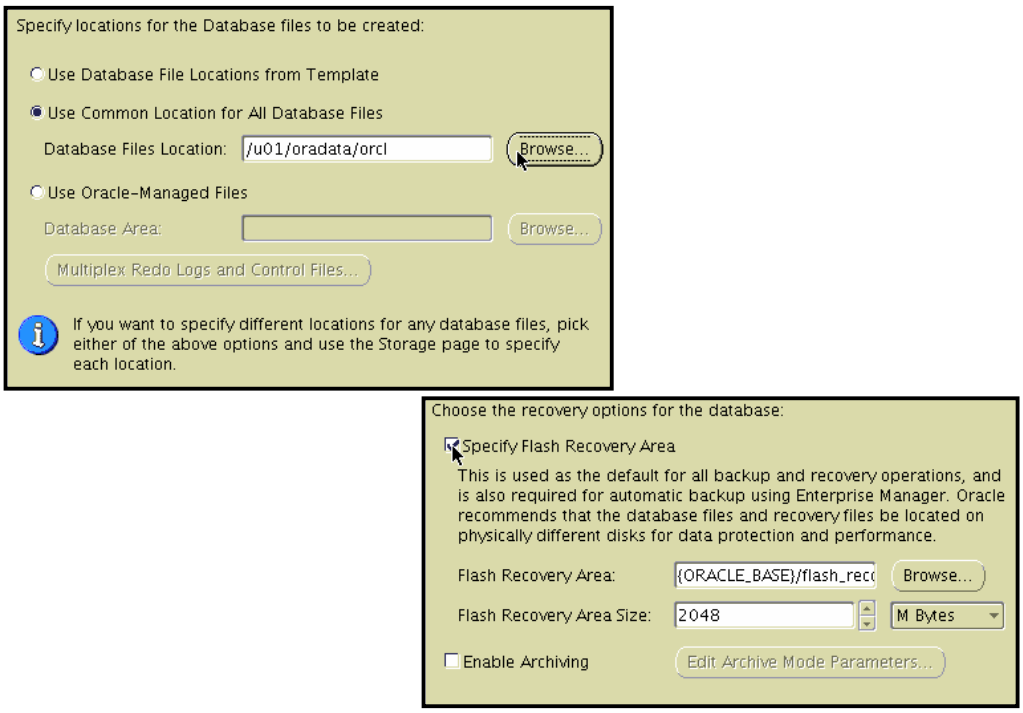
Passwords and Storage

Database schema passwords: Provide passwords for the administrative users, SYS, SYSTEM, SYSMAN and DBSNMP. You can provide a password for each one separately or provide one password for all.

File Storage Options

- File System: This stores files in your OS-configured file system.
- Automatic Storage Management (ASM): Automatic Storage Management files are created and managed automatically, and you get the additional benefits of features such as mirroring and striping. For details on how to set up ASM, see the *Oracle Database Administrator's Guide*.
- Raw Devices (partitions): These are disk partitions without a file system on them. Generally you should use these only if you are very familiar with the use of raw partitions already. Check your OS documentation for details on setting up and maintaining raw partitions.

File Locations and Backup Recovery



File Locations and Backup Recovery

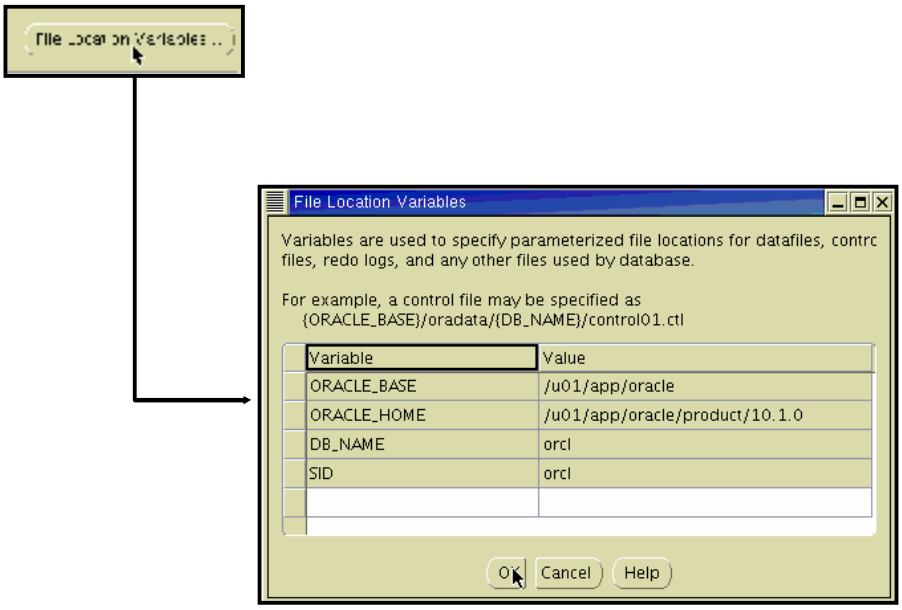
File Locations

- Use Database File Locations from Template: Selecting this option instructs the DBCA to use the directory information as specified in the template. You will have an opportunity later to make modifications to database file names and locations.
- Use Common Location for All Database Files: This option requires you to specify a new common area for all your database files. You will have an opportunity later to make modifications to database file names and locations on the Storage page.
- Use Oracle-Managed Files: Select this option to eliminate the need for you, the DBA, to directly manage operating system files that an Oracle database comprises. You must provide the path to the database area. For more details on Oracle Managed Files see the *Database Administrator's Guide*.

Backup and Recovery Options

- A flash recovery area is a location in which Oracle can store and manage files related to backup and recovery. For details on setting up and sizing the flash recovery area, see the *Oracle Database Backup and Recovery Basics* guide.
- Enabling archiving puts the database in archive log mode at creation time. Archiving will be covered more detail in the lesson titled "Backup and Recovery Concepts."

File Location Variables

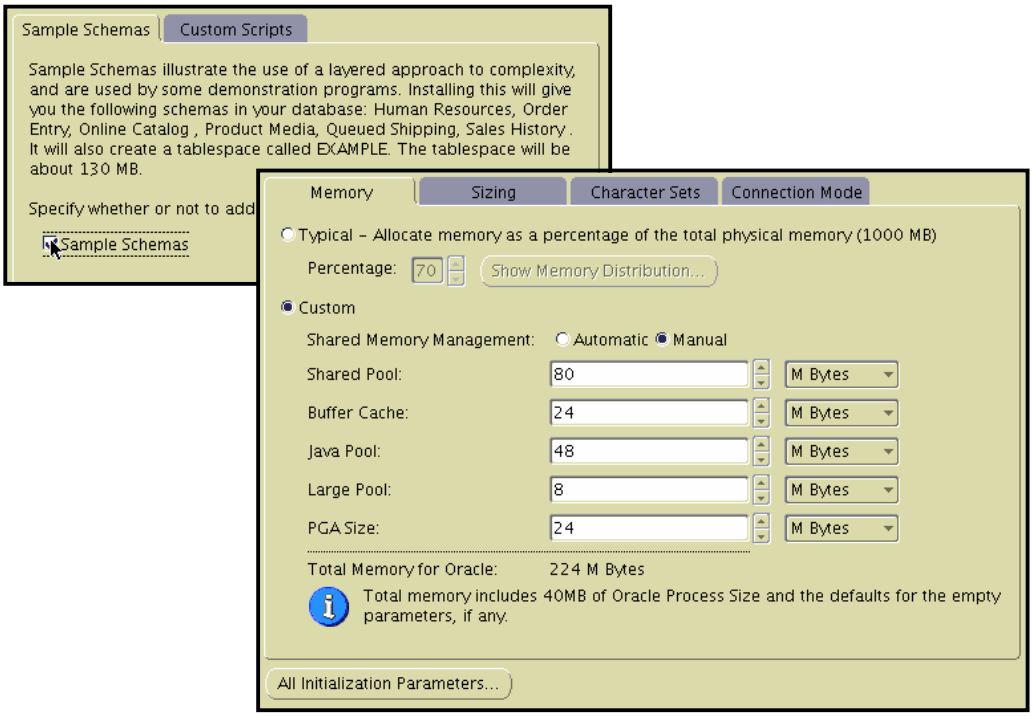


ORACLE

File Location Variables

On several pages you can click File Location Variables to open a page that shows you the definition of defined variables. These variables are used in the path definition for files of the database. You cannot change the values while in DBCA. If you need these values modified, you must exit DBCA, change them in the OS environment, and then restart DBCA.

Content and Initialization Parameters



ORACLE

Content and Initialization Parameters

Sample schemas: The sample schemas are a set of schemas used for demonstrations and training.

Custom scripts: Here you can specify any scripts you want run at creation time.

Initialization parameters: The four tabs can be used to set the most common parameters, and by clicking All Initialization Parameters you can view and set all the parameters.

- Memory: This allocates the memory used by the SGA and each PGA of the user processes.
- Sizing: Here you can set the block size, but if using a template the block size cannot be changed. You can also set the maximum number of OS processes that are allowed to connect to the instance.
- Character sets: Here you set the default character set for the database and the national character set. The default character set is used for most data types in the database. The `NCHAR`, `NVARCHAR2`, and `NCLOB` data types support Unicode data only, which is the national character set option. You can use either the `AL32UTF8` or the `AL16UTF16` character set. For more information on choosing a character set refer to *Globalization Support Guide*.

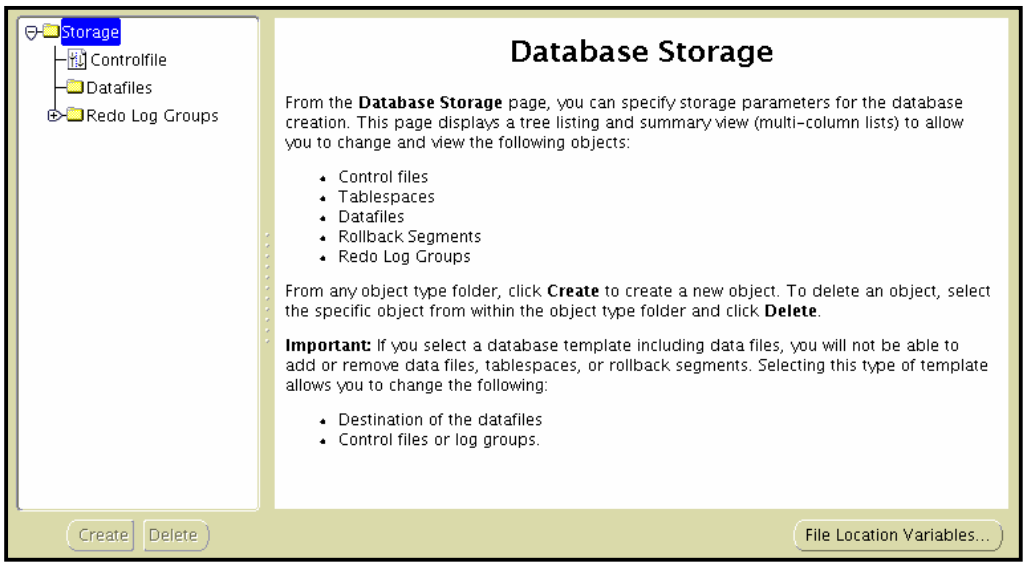
Content and Initialization Parameters (continued)

Connection mode: Oracle Database creates server processes to handle the requests of user processes connected to an instance. A server process can be either of the following:

- A dedicated server process, where one server process services only one user process
- A shared server process, where a server process can service multiple user processes

Your database is always enabled to allow dedicated server processes, but you must specifically configure and enable shared server by setting one or more initialization parameters. Using Oracle Shared Servers will be discussed in a later lesson. You can also refer to the *Database Administrator's Guide*.

Database Storage



ORACLE

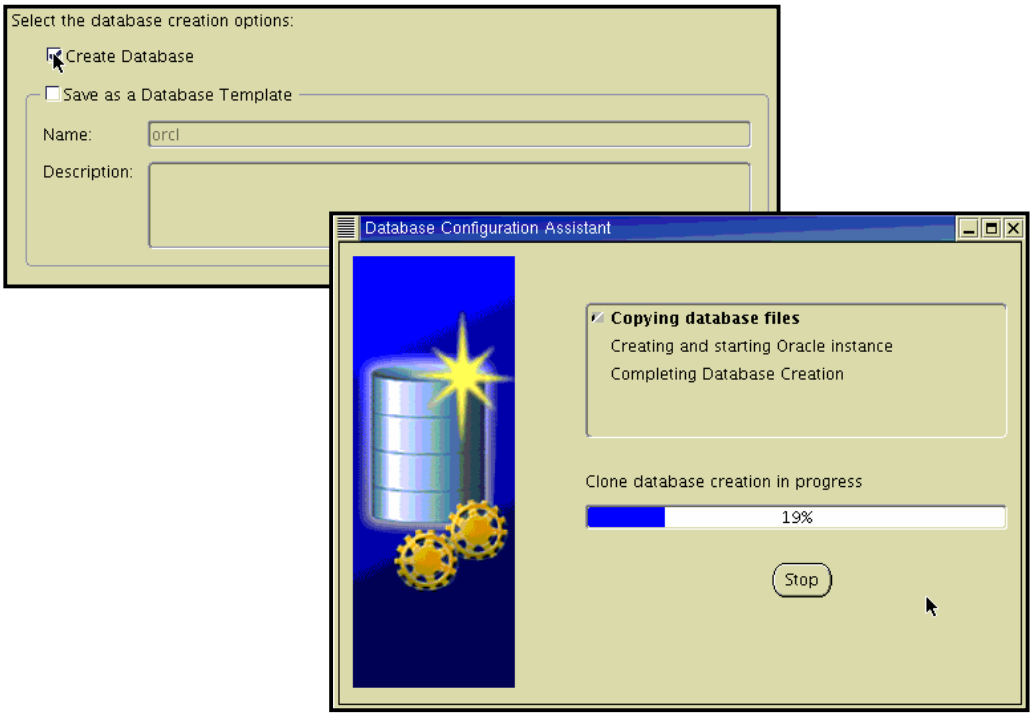
Database Storage

On this page you can see the storage settings for the control files, data files, and online redo log files. When using a template you cannot add any data files to the database, but you can add more control files and online redo log files.

Multiple control files are all maintained such that each is an exact copy of the others. DBCA automatically multiplexes the control file across three files. You can add more if you want.

Redo logs work in groups and should also be multiplexed. DBCA doesn't automatically multiplex the online redo log files. Each file in a log group is an exact copy of the other members in the group. You can add in more members per group now or add them at anytime after creation of the database.

Creation Options and Create



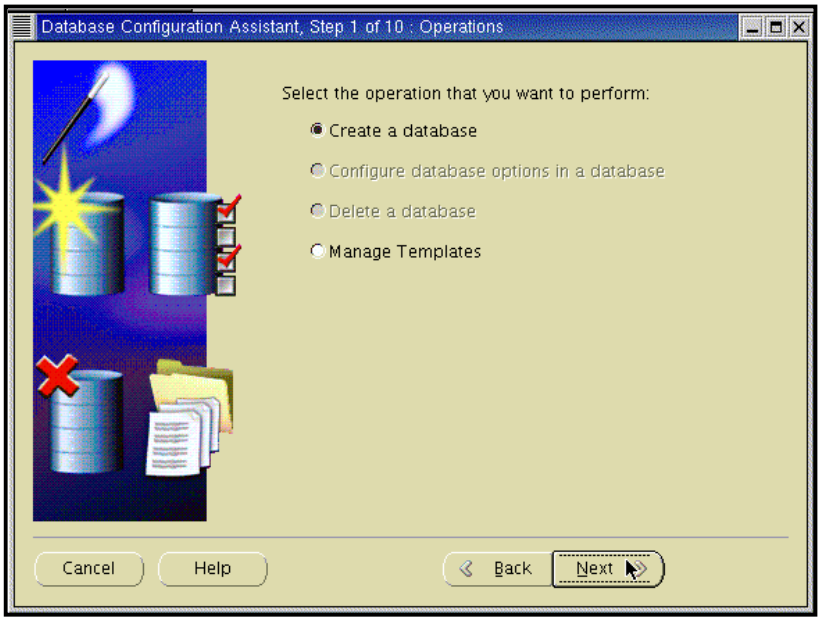
Creation Options and Create

You have the option of saving the database that you have defined as a template. This template can be used later to create databases with all the options you have defined.

After you click Next on the Creation Options page, a Confirmation page appears where you can review all the options taken. This is the last chance to make a change before the creation process starts. You can also save the Confirmation page as an HTML file to review later. After you click OK on the Confirmation page, the database creation starts.

At the end of the installation you will see a page presenting you the opportunity to unlock accounts created and change passwords if you desire. Click Password Management to unlock accounts and change password. Click Ok when don't managing the accounts, then click Exit to DBCA.

Other Actions with DBCA



ORACLE

Other Actions with DBCA

Configure database options in a database: This allows you to add in options to an existing database.

Delete a database: This permanently removes a database from your system.

Manage Templates: This allows you to perform the following with templates.

- Create a Template:
  - From an existing Template: Uses and existing template as a starting point to build a new template. Many of the pages are the same as those in the create database process.
  - From an existing database (structure only): This requires logging in to an existing database and uses its structure as a starting point.
  - From an existing database (structure as well as data):This requires logging into an existing database and uses it's structure as a base line; however it captures the data files as well.
- Delete a database template: This permanently removes a database template from your system

Summary

In this lesson, you should have learned how to:

- **Describe the Oracle database architecture**
- **Understand the instance architecture**
- **Use the management framework**
- **Use DBCA to**
  - **Create a database**
  - **Configure a database**
  - **Drop a database**
  - **Manage templates**

ORACLE

Practice 3:
Creating an Oracle Database

This practice covers creating an Oracle database by using DBCA.

Practice 3: Create an Oracle Database

Your IT manager returns from a meeting with a few of the users that will be using the new system you are going to support. They want a second database for storage of historical data.

Using DBCA you will create a database using the General Purpose template with the following information:

- Set the global database name `hist.oracle.com` and the SID to `hist`.
- Set the passwords to oracle.
- For the storage options use File System.
- Use Flash Recovery area, accept the default size and location, disable the backups.

After you create the database the users decided that they don't need to track the historical data. Drop the `hist` database

4

Database Interfaces

Objectives

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

- **Use SQL*Plus and *i*SQL*Plus to access the Oracle Database 10*g***
- **Describe the logical structure of tables**
- **Use SQL to query, manipulate, and define data**
- **Identify common database interfaces**

ORACLE

Objectives

SQL is the language that is used to work with the database. This lesson provides an overview of this language. In later lessons, individual commands will be introduced and discussed in more detail.

PL/SQL is an extension of SQL. Lesson 10 will cover PL/SQL in more detail.

Java and OCI are mentioned only to familiarize you with the ability to use other languages to access an Oracle database.

What Is SQL?

SQL provides statements for a variety of tasks, including:

- **Querying data**
- **Inserting, updating, and deleting rows in a table**
- **Creating, replacing, altering, and dropping objects**
- **Controlling access to the database and its objects**

SQL unifies all of the preceding tasks in one consistent language.

ORACLE

What Is SQL?

Structured query language (SQL) is the set of statements with which all programs and users access data in an Oracle database. SQL statements are divided into the following categories:

- Data definition language (DDL) statements
- Data manipulation language (DML) statements
- Transaction control statements
- Session control statements
- System control statements
- Embedded SQL statements

With the advent of graphical user interface (GUI) tools, you don't have to write these statements yourself. For many tasks, the tools writes the statements for you. However, if you need to run the SQL within a script, then you must write the SQL statements.

Note: This section provides you an overview of the basic SQL statements available. These statements have many options which are not discussed here, and there are many statements not covered. For full syntax and other commands, see the *Oracle Database SQL Reference* documentation.

Using SQL

There are several tools for interfacing with the database using SQL:

- **Oracle SQL*Plus and *i*SQL*Plus**
- **Oracle Forms, Reports, and Discoverer**
- **Oracle Enterprise Manager**
- **Third-party tools**

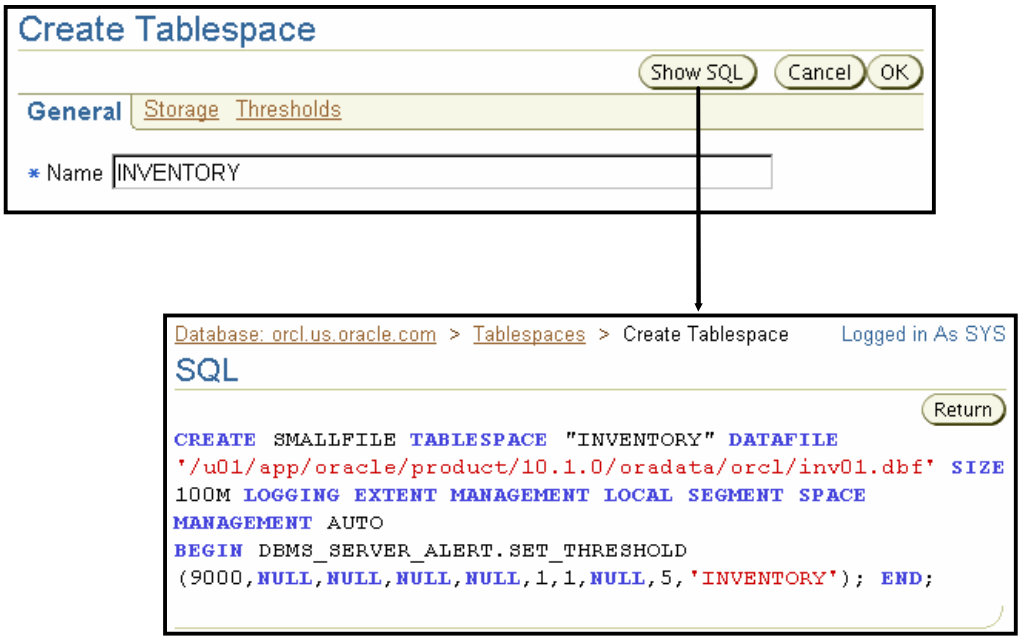
ORACLE

Using SQL

SQL is a rich language whose syntax allows statements as simple as a few words, and statements so complex they fill pages with their syntax. There are several ways to send SQL commands to the database.

- Oracle SQL*Plus and Oracle *i*SQL*Plus: Both of these tools provide a simple, command-line interface with the database.
- Oracle Forms: Masks the SQL from users by providing a graphical-based environment from which they can manipulate data.
- Oracle Reports and Oracle Discoverer: These reporting tools allow complex database queries to be executed with a single mouse click. Oracle Reports is generally used for standard, fixed reports. Oracle Discoverer is a data mining tool that allows users to browse data without knowing the underlying structure and relationship of that data.
- Oracle Enterprise Manager: A graphical tool that manipulates database objects and structures. By using Enterprise Manager, the administrator can concentrate on accomplishing the task at hand rather than remembering syntax.
- Third-party tools. SQL is a standards-based language used by most major database vendors. There are hundreds of tools for interfacing with databases using SQL.

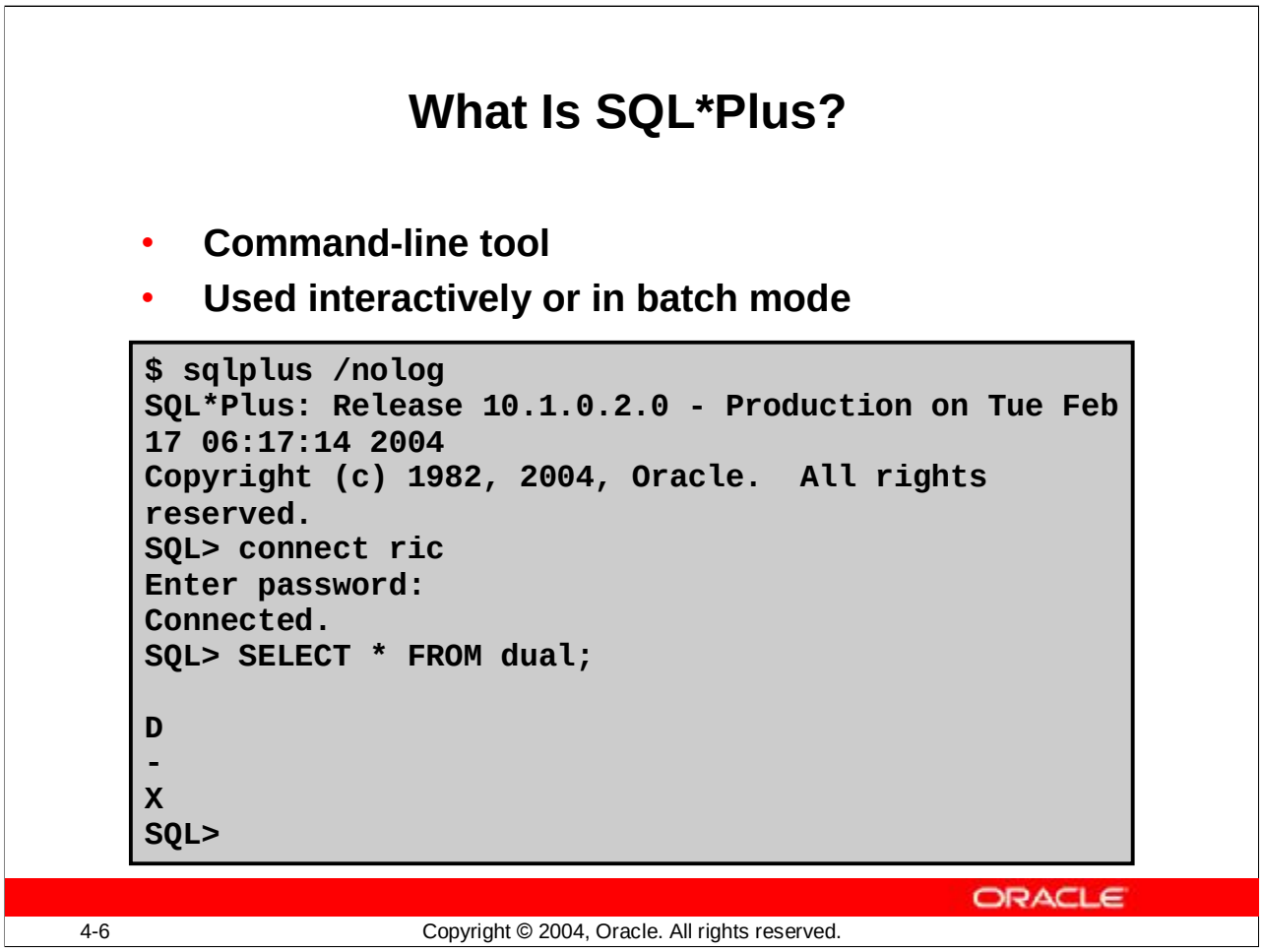
Enterprise Manager: Seeing the SQL



ORACLE

Enterprise Manager: Seeing the SQL

Using the Enterprise Manager Database Control you do not need to write the SQL statements used to administer your database. Enterprise Manager creates the SQL for you. If you want to see the SQL that is being generated, click Show SQL at anytime to see it displayed. You can copy and paste the SQL from here to a text file if you want to create a script file that will be run in batch mode with a script, or if you just want to keep it for historical purposes.



What Is SQL*Plus?

You can use the command-line interface to SQL*Plus to write SQL*Plus, SQL and PL/SQL commands to:

- Enter, edit, run store, retrieve, and save SQL commands and PL/SQL blocks
- Format, calculate, store, and print query results
- List column definitions for any table
- Send messages to and accept responses from an end user
- Perform database administration

Perform the following steps to start SQL*Plus:

1. Open a terminal window.
2. At the command-line prompt, enter the SQL*Plus command in the form:
   ```
   $ sqlplus /nolog
   ```
3. Enter `connect` followed by the user you want to connect as.
4. When prompted, enter the user's password.
5. SQL*Plus starts and connects to the default database.

What Is *i*SQL*Plus?

What Is *i*SQL*Plus?

*i*SQL*Plus is a browser-based interface to an Oracle database. It is a component of the SQL*Plus product. *i*SQL*Plus has a server-side listener process that must be started before you can connect with a browser. To start this server process use:

```
isqlplusctl start
```

After the server process is started, perform the following steps to connect to *i*SQL*Plus:

1. Connect to the Internet or your intranet, and start your Web browser.
2. Enter your the *i*SQL*Plus URL. The *i*SQL*Plus URL looks like the following:

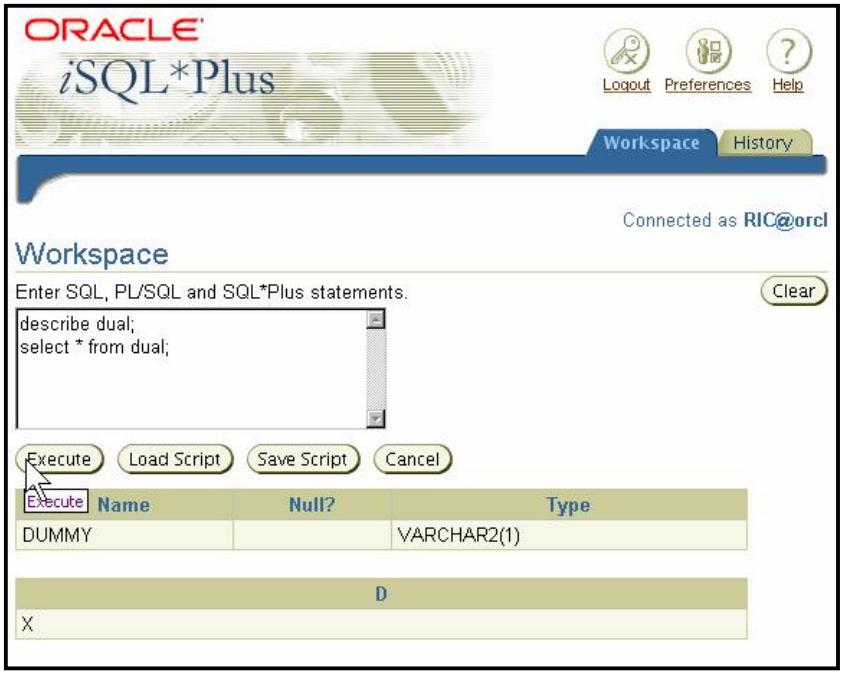
```
http://hostname:port/isqlplus.
```

The port number used by *i*SQL*Plus is usually 5560 unless the Oracle Universal Installer (OUI) detects something already using that port. Check `$ORACLE_HOME/install/portlist.ini` to find the port used by *i*SQL*Plus.

What Is *i*SQL*Plus? (continued)

3. Press Enter to go to the URL. The *i*SQL*Plus Login page is displayed in your Web browser.
4. Enter your Oracle username and password in the Username and Password fields. If you do not know your Oracle username and password, ask your database administrator.
5. Leave the Connection Identifier field blank to connect to the default database. Enter the Oracle Net database alias to connect to a database other than the default.
6. Click Login to connect to the database. The *i*SQL*Plus Workspace is displayed in your Web browser.

Using *i*SQL*Plus



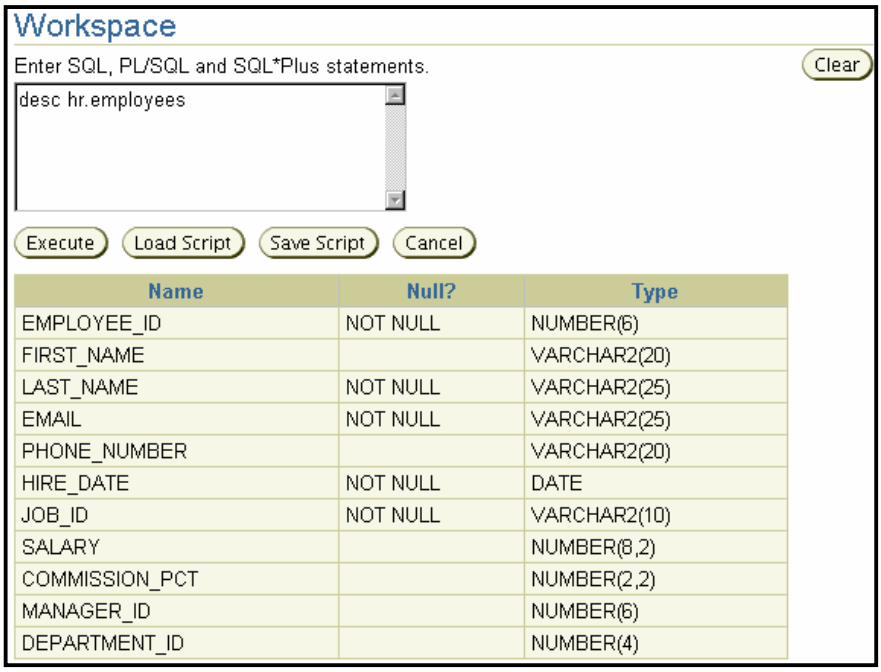
Using *i*SQL*Plus

In the Workspace tabbed page you can enter one or more SQL and SQL*Plus commands. When you click Execute all the commands are executed in order top to bottom. The output is displayed below the command panel. If you have more than one command, SQL commands must be terminated by a semicolon (;) or slash (/).

The History tabbed page shows the prior commands used within the session. You can load or delete prior commands. By default the last 10 commands are saved.

You can click Preferences to change both the look of the *i*SQL*Plus window as well as its functionality. For example you can set the number of commands to be saved (Range is 0 to 100), how the output is formatted, and execution options.

Describing Data



Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

ORACLE

Describing Data

The `DESCRIBE` command is used to see the description of a database object. This command is not part of the SQL standard, but is one of several SQL*Plus command unique to Oracle tools. Unlike SQL commands, the SQL*Plus commands can be abbreviated to its first four letters.

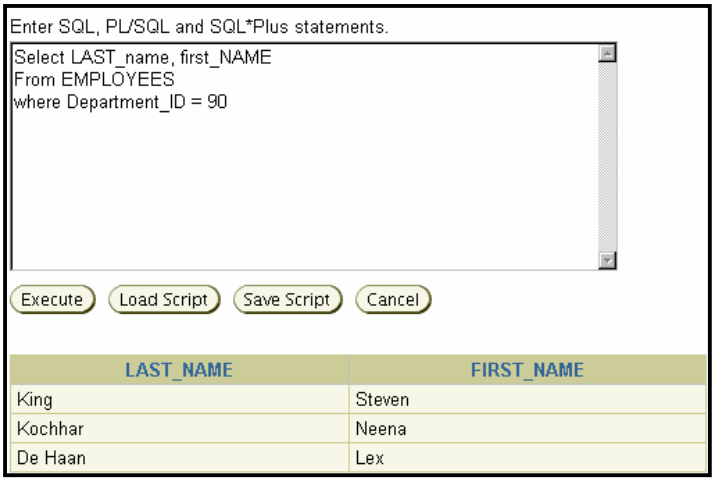
The example shows the structure of the `employees` table. From the output you can see the name of each column, whether the column is mandatory, and the data type associated with that column. For example, the column with the name of `LAST_NAME` is mandatory because `NOT NULL` means that rows must contain a value in this column. Also the description shows this column to be of type `VARCHAR2(25)`, which means it can contain character data and have a maximum length of 25 characters.

For more information about SQL*Plus and it's commands see the *SQL*Plus Reference Guide*.

Querying Data

The SELECT has three basic parts:

- **The SELECT List**
- **The FROM clause**
- **The WHERE condition (optional)**



ORACLE

Querying Data

The most basic SELECT statement has three parts:

- The SELECT list is the list of columns you want to retrieve from the table or tables. This is mandatory.
- The FROM clause is where you specify the table or tables from which you are retrieving data. This is mandatory.
- The WHERE condition limits the rows retrieved from the tables in the FROM clause. This part is optional; if not included you will retrieve all the rows.

SQL is also not a case-sensitive language. In the example, the case is mixed rather unusually only as an example to demonstrate this. Many organizations have case standards that facilitate readability. By default, the only time case is important is in a quoted string. For example, if you want to retrieve only the row that has the last name of De Haan, then the WHERE condition looks like the following:

```
WHERE last_name = 'De Haan'
```

Sorting the Data

```
SQL> SELECT last_name, department_id, phone_number
  2  FROM employees
  3  ORDER BY last_name;

LAST_NAME       DEPARTMENT_ID PHONE_NUMBER
--------------- ------------- --------------------
Abel                       80 011.44.1644.429267
Ande                       80 011.44.1346.629268
Atkinson                   50 650.124.6234
Austin                     60 590.423.4569
Baer                       70 515.123.8888
Baida                      30 515.127.4563
Banda                      80 011.44.1346.729268
```


Sorting the Data

Use the `ORDER BY` clause on a select statement to sort the output. The default order is ascending, to get the data in descending order use `DESC` after the column name you are sorting on. For example:

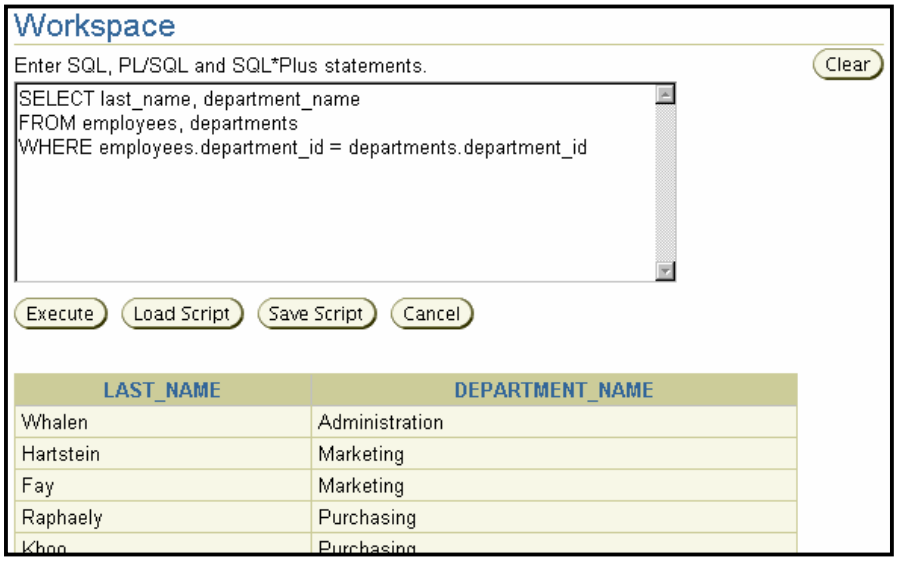
```
ORDER BY last_name desc
```

You can sort on more then one column. For example, this will sort descending by the last name and then ascending by department id:

```
ORDER BY last_name desc, department_ID
```

Joining Tables

Getting data from more than one table



Joining Tables

Sometimes you need to use data from more than one table. In the slide example, the report displays data from two separate tables. To produce the report, you need to link the `employees` and `departments` tables and access data from both of them.

When data from more than one table in the database is required, a join condition is used. Rows in one table can be joined to rows in another table according to common values that exist in corresponding columns; these are usually primary and foreign key columns.

To display data from two or more related tables, write a join condition in the `WHERE` clause. The example in the slide is:

```
SELECT last_name, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id
```

Note that the column used to join the tables (`department_id`) need not be in the `SELECT` list of the statement. Also, because the `department_id` column has the same name in each table, the column must be qualified with the table name in front of it in the `WHERE` clause .

Joining Tables (continued)

An equijoin has a join condition that contains an equality operator. An equijoin combines rows that have equivalent values for the specified columns. The example in the slide is an equijoin.

A self join is a join of a table to itself. This table name appears twice in the FROM clause and is followed by table aliases that qualify column names in the join condition. For example:

```
SELECT e1.last_name||' works for '||e2.last_name
"Employees and Their Managers"
FROM employees e1, employees e2
WHERE e1.manager_id = e2.employee_id
AND e1.last_name LIKE 'R%';

Employees and Their Managers
------------------------------
Rajs works for Mourgos
Raphaely works for King
Rogers works for Kaufling
Russell works for King
```

An inner join (sometimes called a simple join) is a join of two or more tables that returns only those rows that satisfy the join condition. The example in the slide is an inner join.

An outer join extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition. For example, the following query returns all department numbers even if there are no employees assigned to the department:

```
SELECT d.department_id, e.last_name
FROM departments d LEFT OUTER JOIN employees e
ON d.department_id = e.department_id
ORDER BY d.department_id;
```

A Cartesian product is generated if a join condition is omitted. This means you will retrieve all rows from one table joined with all the rows of any other table or tables in the SELECT list.

Oracle also support ANSI join syntax. For example the slide example can be rewritten as:

```
SELECT first_name, department_name
FROM employees JOIN departments USING (department_id);
```

For more information on join types see the *SQL Reference* documentation.

Manipulating Data

```
SQL> INSERT INTO employees
  2  (EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE_NUMBER,
  3  HIRE_DATE,JOB_ID,SALARY,COMMISSION_PCT,
  4  MANAGER_ID,DEPARTMENT_ID)
  5  VALUES
  6  (9999,'Bob','Builder','bob@abc.com',NULL,sysdate,
  7  'IT_PROG',NULL,NULL,100,90);

1 row created.

SQL> UPDATE employees SET SALARY=6000
  2  WHERE EMPLOYEE_ID = 9999;

1 row updated.

SQL> DELETE from employees
  2  WHERE EMPLOYEE_ID = 9999;

1 row deleted.
```

ORACLE

Manipulating Data

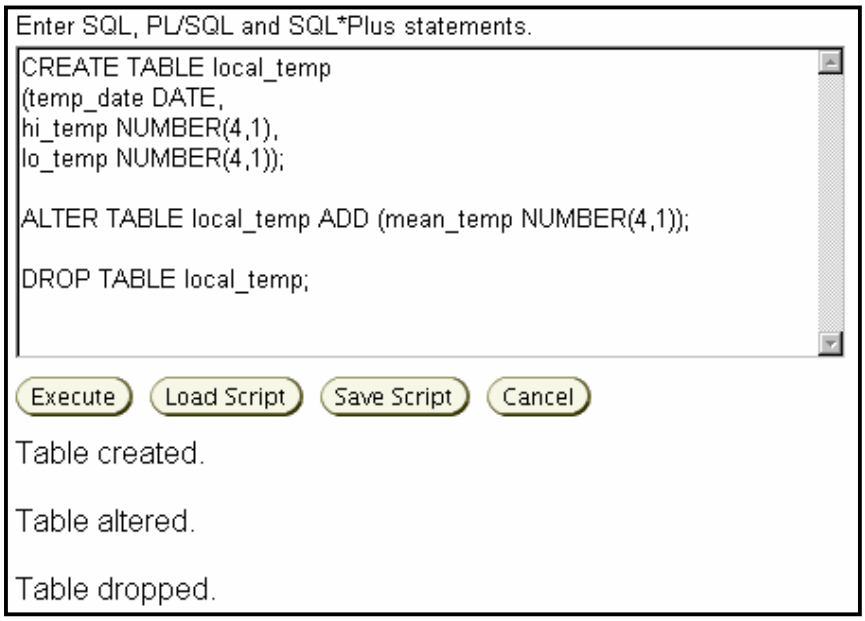
The statements that make up the data manipulation language (DML) are used to change data in the database.

- INSERT puts new rows of data into a table in the database. Used in the form shown in the slide, one row of data is created in the table at a time.
- UPDATE modifies an existing row (or set of rows) in a table. The number of rows modified is controlled by the WHERE condition. If you don't have a WHERE condition on the UPDATE, then all rows in the table are modified.
- DELETE removes rows from a table. Like UPDATE the number of rows removed is controlled by a WHERE condition. Without a WHERE condition, all rows are removed from the table.

On most systems it is not common to use these statements regularly. Instead of issuing DML statement such as in the slide, you will most likely use an application. The application will then generate the needed DML statements to change the data as you need.

Note: The NULL keyword inserts no value for the column. This is not a blank nor a zero; it is rather no value at all. The SYSDATE keyword inserts the current date and time into a DATE data type column.

Defining Data



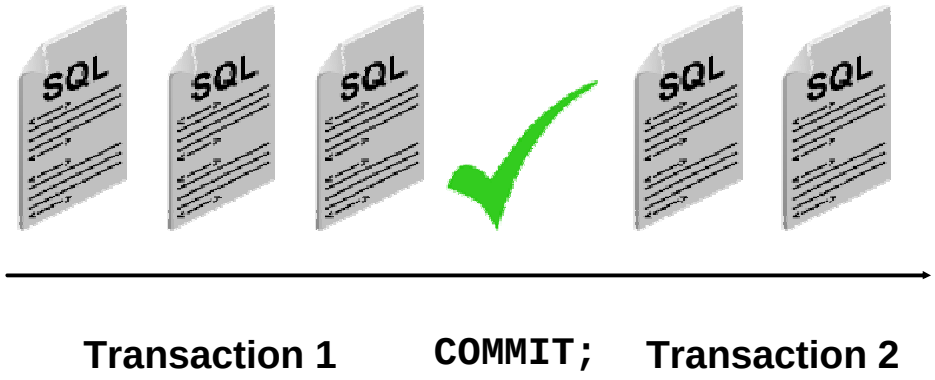
ORACLE

Defining Data

The statements that make up the data definition language (DDL) are used to work with database objects.

- The CREATE statement defines new objects in the database. You can create many different objects in the database, including tables, indexes, sequences, views, and others.
- The ALTER statement modifies existing objects in the database. After an object is defined, you may need to modify that object. Depending on the object, there are different things that can be modified. For example, you can add a column to a table, but you cannot add a column to an index.
- The DROP command removes objects in the database. After an object is dropped it can no longer be referenced. It is possible to reinstate a dropped a table.

Overview of Transactions



ORACLE

Overview of Transactions

When an executable SQL statement is executed a transaction begins. Oracle assigns the transaction to an available undo segment in an undo tablespace to record the undo entries for the new transaction. The entries in the undo tablespace are used when a rollback command is issued to get the data back to its last committed state prior to the current transaction. A transaction ends when any of the following occurs:

- A user issues a `COMMIT` or `ROLLBACK` statement without a `SAVEPOINT` clause.
- A user runs a DDL statement such as `CREATE`, `DROP`, `RENAME`, or `GRANT`. If the current transaction contains any DML statements, Oracle first commits the transaction, and then runs and commits the DDL statement as a new, single statement transaction.
- A user disconnects from Oracle. The current transaction is committed.
- A user process terminates abnormally. The current transaction is rolled back.

After one transaction ends, the next executable SQL statement automatically starts the next transaction.

Note: A set of `SELECT` commands that accesses data in the same database the user is connected to does not constitute a transaction.

Transaction Control Statements

```
SQL> SELECT * FROM local_temp;
no rows selected

SQL> INSERT INTO local_temp VALUES
  2  (SYSDATE, 76, 58);
1 row created.

SQL> SELECT * from local_temp;
TEMP_DATE    HI_TEMP    LO_TEMP
--------- ---------- ----------
27-OCT-03         76         58

SQL> ROLLBACK;
Rollback complete.

SQL> SELECT * FROM local_temp;
no rows selected
```

ORACLE

Transaction Control Statements

There are three transaction control statements, COMMIT, ROLLBACK, and SAVEPOINT.

- Committing a transaction means making permanent the changes performed by the SQL statements within the transaction.
- Rolling back means undoing any changes to data that have been performed by SQL statements within an uncommitted transaction. Oracle uses undo tablespaces or rollback segments to store old values. The redo log contains a record of changes.
- You can declare intermediate markers, called savepoints, within the context of a transaction. Savepoints divide a long transaction into smaller parts. When you set up a savepoint, you can then later roll back to that particular savepoint. Each savepoint must have a name, which you can use to roll back to that named savepoint. For example:

```
SQL> SAVEPOINT before_insert;
Savepoint created.
SQL> INSERT INTO local_temp VALUES (SYSDATE, 76, 58);
1 row created.
SQL>  ROLLBACK TO before_insert;
Rollback complete.
```

Locking Data

Oracle Database 10*g* automatically locks data so that only one user can make changes at a time.

Locking Data

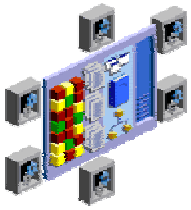
Oracle provides data concurrency and integrity between transactions using its locking mechanisms. This mechanism prevents multiple users from trying to update the same data at the same time.

Oracle Database 10*g* uses a fine-grained locking mechanism to ensure that users don't unnecessarily block other users. Someone reading data (using select statements) does not block someone from updating data. Someone updating data does not block another user unless that user tries to update the same row of data.

Oracle's locking mechanism will be discussed in more detail in a later lesson.

Other Statement Categories

- **Session control statements: Manage the properties of a user session**
- **System control statement: Manages the properties of an Oracle instance**
- **Embedded SQL statements: SQL statements within a procedural language program**


ORACLE

Other Statement Categories

- Session control statements: Session control statements dynamically manage the properties of a user session. These statements do not implicitly commit the current transaction. PL/SQL does not support session control statements. The session control statements are:
  - `ALTER SESSION`
  - `SET ROLE`
- System control statement: The single system control statement, `ALTER SYSTEM`, dynamically manages the properties of an Oracle instance. This statement does not implicitly commit the current transaction and is not supported in PL/SQL.
- Embedded SQL statements: Embedded SQL statements place DDL, DML, and transaction control statements within a procedural language program. Embedded SQL is supported by the Oracle precompilers and is documented in the following:
  - *Pro*COBOL Programmer's Guide*
  - *Pro*C/C++ Programmer's Guide*
  - *SQL*Module for Ada Programmer's Guide*

What Is PL/SQL?

PL/SQL is a block-structured language, which extends SQL with:

- **Declarations:**
  - **Variables**
  - **Constants**
  - **Cursors**
- **Control structures:**
  - **Conditional control**
  - **Iterative control**
  - **Sequential control**
- **Error handling**

ORACLE

What Is PL/SQL?

PL/SQL is a block-structured language. That is, the basic units (procedures, functions, and anonymous blocks) that make up a PL/SQL program are logical blocks, which can contain any number of nested subblocks. Typically, you create each logical block to solve a problem or subproblem. One way to look at it is that PL/SQL adds structured programming techniques to the SQL language.

A block of PL/SQL code has these parts:

- `DECLARE`: This optional part is where you define constants and variables, then use them in SQL and procedural statements anywhere an expression can be used. You must declare a constant or variable before referencing it in other statements, including other declarative statements.
- `BEGIN`: This is the beginning of the code of the block. The code will be made up of both SQL and PL/SQL commands. This is mandatory.
- `EXCEPTION`: This optional part is where you define any local exceptions. Generally exceptions are raised because of an error condition.
- `END`: this ends the block. This is mandatory.

Example PL/SQL Block

```
DECLARE
  qty_on_hand NUMBER(5);
BEGIN
    SELECT quantity INTO qty_on_hand FROM
      inventory
    WHERE product = 'TENNIS RACKET'
    FOR UPDATE OF quantity;
    IF qty_on_hand > 0 THEN -- check quantity
       UPDATE inventory SET quantity = quantity - 1
       WHERE product = 'TENNIS RACKET';
       INSERT INTO purchase_record
       VALUES ('Tennis racket purchased', SYSDATE);
    ELSE
       INSERT INTO purchase_record
       VALUES ('Out of tennis rackets', SYSDATE);
    END IF;
  COMMIT;
END;
```

Example PL/SQL Block

The anonymous block in the slide starts off with defining a local variable to hold a number. The code of the block starts with the `BEGIN` statement. The first action is to select the amount of available inventory for tennis rackets into the local variable. Next follows some conditional control in the from of an `IF THEN ELSE` statement.

If there are more than zero tennis rackets, then the quantity is reduced by one and a message written to the `purchase_record` table. If there are zero tennis rackets, then only a message is written to the `purchase_record` table.

Last in the code of the block is a `COMMIT` which makes any changes done in the block permanent. The block is closed with an `END` statement.

For more details on PL/SQL, refer to *PL/SQL User's Guide and Reference* documentation.

Uses of PL/SQL

Blocks of PL/SQL are used in:

- **Anonymous blocks**
- **Functions**
- **Procedures**
- **Packages**
- **Triggers**
- **Object types**

ORACLE

Uses of PL/SQL

You use PL/SQL in the following structures:

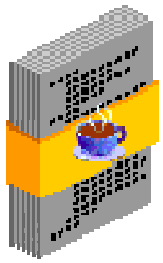
- Anonymous blocks: These are unnamed blocks of PL/SQL. An anonymous block is generally found inside a named block (a function or procedure for example), it is the nesting of these anonymous blocks that provides greater control over the execution of the code.
- Functions: A function is a PL/SQL block that returns a single value by using the RETURN PL/SQL command
- Procedures: A procedure is is a PL/SQL block that performs a specific action
- Packages: A package is a collection of procedures, functions, and anonymous blocks that are logically related.
- Triggers: A trigger is a PL/SQL block that is executed when a particular event happens in the database. These events can be based on a table, such as when a row is inserted into the table. They can also be database events, such as when a user logs in to the database.
- Object types: An object type is a kind of data type. For details see the *Application Developer's Guide - Object-Relational Features* documentation.

These structures (except object types) will be discussed in more detail in a later lesson.

What Is Java?

Java is an industry-standard, object-oriented programming language. It includes the following concepts:

- **A Java Virtual Machine (JVM), which provides platform independence**
- **Automated storage management techniques**
- **Language syntax that borrows from C and enforces strong typing**



ORACLE

What Is Java?

Java, an object-oriented programming language released by Sun Microsystems in 1995, has had a profound effect on online applications. Java programs can be run on any computer that has a Java interpreter (called a Java Virtual Machine), which is included in most Web browsers. Programmers use Java to provide dynamic content, such as graphics, motion, and sound. Java-enhanced Web pages enable interactive applications such as simulations and computer-based training. Java is complied and stored in the database as bytecode, a code form between machine code and source code. Java bytecode is platform independent, which make it easy to transfer between operating systems. The JVM converts bytecode into machine-readable instructions.

The Oracle instance has a built-in JVM.

Oracle and Java

A PL/SQL function:

```
FUNCTION balance (acct_id NUMBER) RETURN NUMBER IS
acct_bal NUMBER;
BEGIN
 SELECT bal INTO acct_bal FROM accts
   WHERE acct_no = acct_id;
 RETURN acct_bal;
END;
```

Calling the function with Java:

```
CallableStatement
 cstmt = conn.prepareCall("{? = CALL balance(?)}");
 cstmt.registerOutParameter(1, Types.FLOAT);
 cstmt.setInt(2, acctNo);
 cstmt.executeUpdate();
float acctBal = cstmt.getFloat(1);
```

ORACLE

Oracle and Java

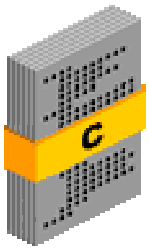
You can call Java stored procedures from any PL/SQL block, subprogram, or package. Similarly, JDBC (Java Database Connectivity) allows you to call PL/SQL stored functions and procedures.

To learn more about Java's usages in an Oracle Database, see the *Java Developer's Guide* documentation. To learn more about JDBC, see the *Oracle Database JDBC Developer's Guide and Reference* documentation.

What Is OCI?

OCI provides for:

- **The Oracle Call Interface (OCI) is how all database features are made accessible to application developers.**
- **OCI makes scalable and high-performance applications possible.**
- **Higher-level APIs and tools use OCI indirectly for database access.**



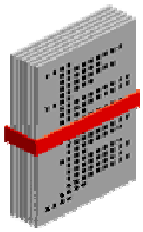
ORACLE

What Is OCI?

The Oracle Call Interface (OCI) is an application programming interface (API) that allows applications written in C to interact with one or more Oracle Database servers. OCI is a complex and powerful low-level API. It provides improved performance and scalability through the efficient use of system memory and network connections. OCI allows dynamic session and transaction management in a two-tier client/server or multitier environment and provides for N-tier authentication. New database capabilities and features are first exposed to application developers through OCI. Other APIs are layered over OCI.

Other APIs

- Java Database Connectivity (JDBC)
- Pro*C/C++
- Pro*COBOL
- Oracle C++ Interface (OCCI)
- Open Database Connectivity (ODBC)
- Oracle Data Provider for .NET (ODP.NET)
- Oracle Objects for OLE (OO4O)



ORACLE

Other APIs

This course will not cover these APIs in any detail. The intent here is to inform you of the options that are available.

Java Database Connectivity (JDBC): JDBC is the standard API for accessing relational databases from programs written in Java. Stand-alone Java applications, applets running in a browser, and Java 2 Platform, Enterprise Edition (J2EE) middle-tier components, such as servlets and Enterprise JavaBeans, can all use JDBC to access an Oracle Database. In addition, Java classes can be loaded into an Oracle Database and published as stored procedures, where they also use JDBC for database access. Java programs can use JDBC to call both Java and PL/SQL stored procedures. Oracle provides two different kinds of JDBC drivers. Thin drivers are a 100% Java implementation of the JDBC specification. OCI drivers interface with the same native OCI client libraries as all other APIs. For more information see the *JDBC Developer's Guide and Reference* guide.

Other APIs (continued)

Pro*C/C++ and Pro*COBOL: From the earliest versions of the Oracle Database programmers have embedded SQL statements inside programs written in third-generation programming languages such as C and COBOL. Pro*C/C++ and Pro*COBOL are precompilers that translate this embedded SQL into Oracle run-time library calls for database access. For more information see the *Pro*C/C++ Precompiler Programmer's Guide* and the *Pro*COBOL Precompiler Programmer's Guide*.

Oracle C++ Interface (OCCI): OCCI provides object-oriented access to an Oracle database through an interface that, for C++ programmers, is simpler than OCI and more natural than Pro*C/C++. At the same time it exposes most of the advanced features of the more low-level OCI.

Open Database Connectivity (ODBC): ODBC provides a standard interface that allows one application to access many different data sources. A database driver links the application to a specific data source. The Oracle ODBC driver enables Windows applications to access Oracle databases through the ODBC interface using native OCI client libraries.

Oracle Data Provider for .NET (ODP.NET): ODP.NET allows fast and reliable access to database features and data from any .NET application. It also uses and inherits from classes and interfaces available in the Microsoft .NET Framework Class Library. For more information see the *Oracle Data Provider for .NET Developer's Guide*.

Oracle Objects for OLE (OO4O): Oracle Objects for OLE (OO4O) allows easy access to data stored in an Oracle Database from any programming or scripting language that supports the Microsoft COM Automation and ActiveX technology. This includes Visual Basic, Visual C++, Visual Basic For Applications (VBA), IIS Active Server Pages (VBScript and JavaScript), and others.

Other Access Protocols

When the Oracle XML DB is enabled, standard Internet protocols such as FTP, HTTP, and Web Distributed Authoring and Versioning (WebDAV) can be used to access data in an Oracle database. For more information see the *XML DB Developer's Guide*.

Summary

In this lesson, you should have learned how to:

- **Use SQL*Plus and *i*SQL*Plus to access Oracle Database 10*g***
- **Describe the logical structure of tables**
- **Use SQL to query, manipulate, and define data**
- **Identify common database interfaces**

ORACLE

Practice 4: Using SQL

This practice covers using *i*SQL*Plus to:

- **Describe tables**
- **Select from tables**
- **Update a table**
- **Delete from a table**
- **Undo changes**

ORACLE

Practice 4: Using SQL

Background: In this practice session you are acting as a clerk in the Human Resources department. You will need to connect to the database as the HR user. Once connected you will view and manipulate data.

Tasks:

- Start *i*SQL*Plus
- Connect to the database using *i*SQL*Plus
- Connect to the database using SQL*Plus
- Describe database objects
- Query data stored within the database.
- Update existing data stored within the database.
- Rollback an update to restore data to its original value.
- To familiarize your self with the data in the database you decide to use *i*SQL*Plus to see the table data in the HR schema.

1. Connect to the database as user HR using *i*SQL*Plus.
2. Describe the EMPLOYEES, DEPARTMENTS and LOCATIONS tables.
3. Write a query that show you the last name, phone number and department id of everyone in the EMPLOYEES table.
4. Rewrite the query so that the department name is shown instead of the department_id.
5. Rewrite the query to show the last name, department name, and city of everyone in the EMPLOYEES table.
6. Notice that the results are not returned in any particular order. Rewrite the query so that results are returned in alphabetical order sorted first by city, then by department name, then by last name.
7. Write and execute an SQL statement that gives everyone in department 80 a 10% raise.
8. The 10% raise you gave in step (7) has been rescinded by upper management. Reverse the transaction.
9. Connect to the database as user HR using SQL*Plus.
10. Write a query that show you the last name, department name and state of everyone in department 30.

5

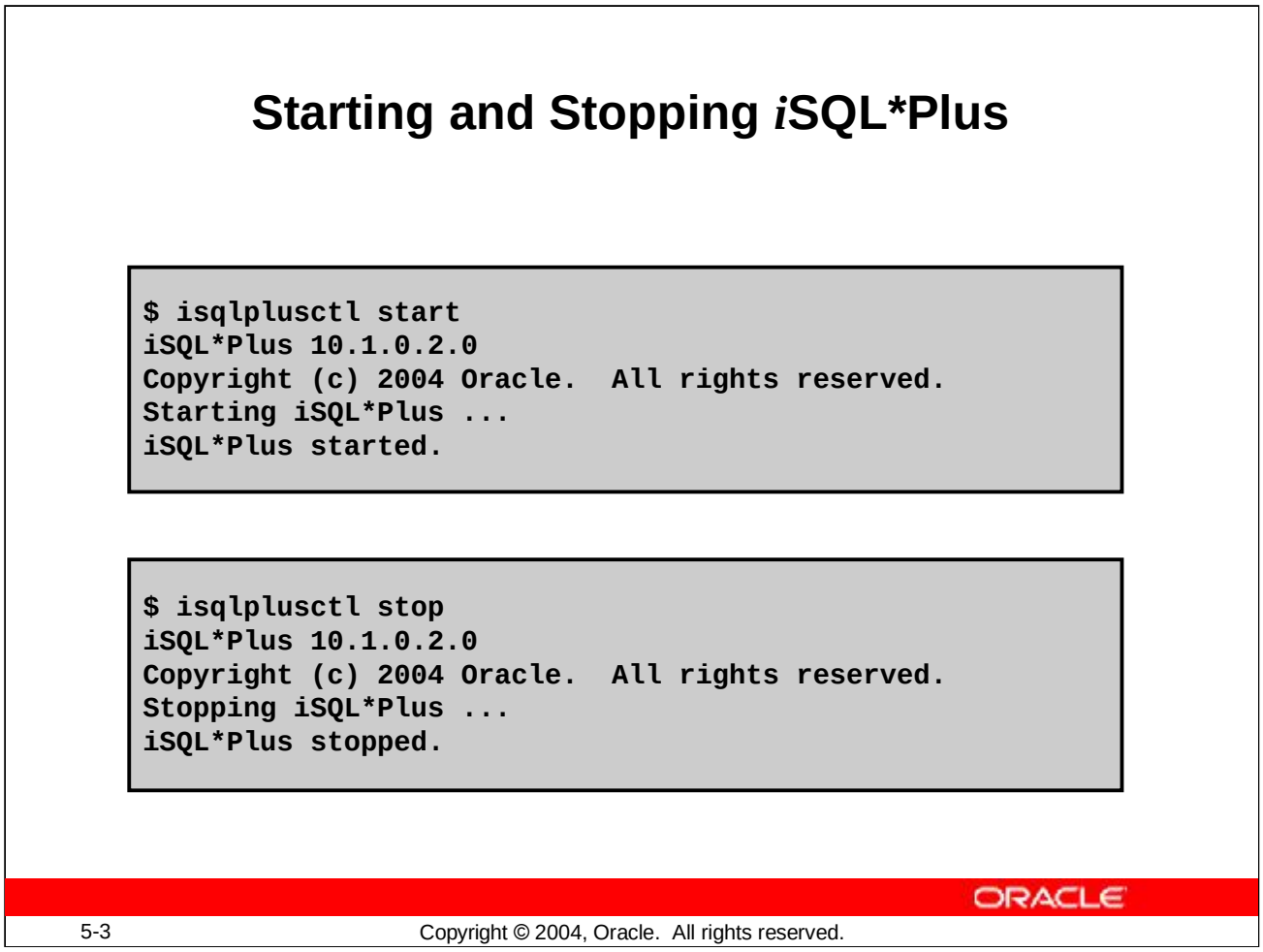
Controlling the Database

Objectives

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

- **Start and stop *i*SQL*Plus**
- **Start and stop Enterprise Manager Database Control**
- **Start and stop the Oracle Listener**
- **Start up and shut down Oracle Database 10*g***

ORACLE



Starting and Stopping *i*SQL*Plus

The *i*SQL*Plus Application Server must be running before you can start an *i*SQL*Plus session. A command-line utility and a Windows Service are supplied to start and stop *i*SQL*Plus on Windows. The *i*SQL*Plus Application Server is started by default during Oracle Database installation.

To start or stop the *i*SQL*Plus Application Server on Unix (Linux).

- Start a terminal session.
- Enter the following to start: `isqlplusctl start`
- Enter the following to stop: `isqlplusctl stop`

Note: You may need to navigate into your `$ORACLE_HOME/bin` directory if this directory is not in your OS path.

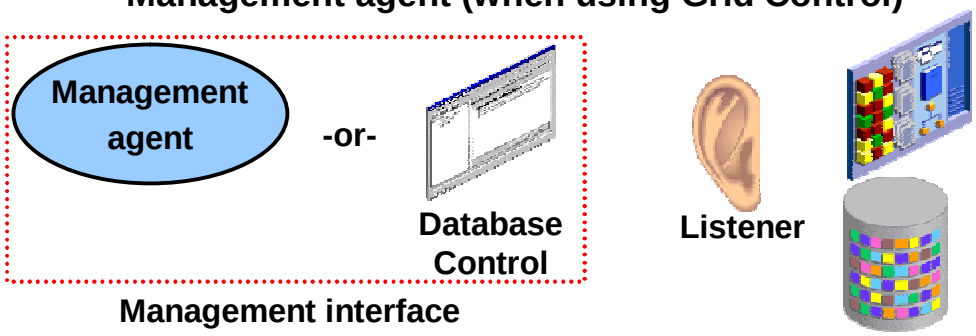
To start or the *i*SQL*Plus Application Server on Windows

- Select Services from the Start > Programs > Administrative Tools menu.
- Locate the *i*SQL*Plus Windows Service, OracleOracleHomeName*i*SQL*Plus.
- Start the Windows Service to start the *i*SQL*Plus Application Server, or stop the service to stop the *i*SQL*Plus Application Server.
- Alternatively, you can start *i*SQL*Plus from a command prompt. Use the same command as in Unix (Linux).

Management Framework

The three components of the Oracle Database 10*g* management framework are:

- **Database instance**
- **Listener**
- **Management interface**
  - **Database Control**
  - **Management agent (when using Grid Control)**



ORACLE

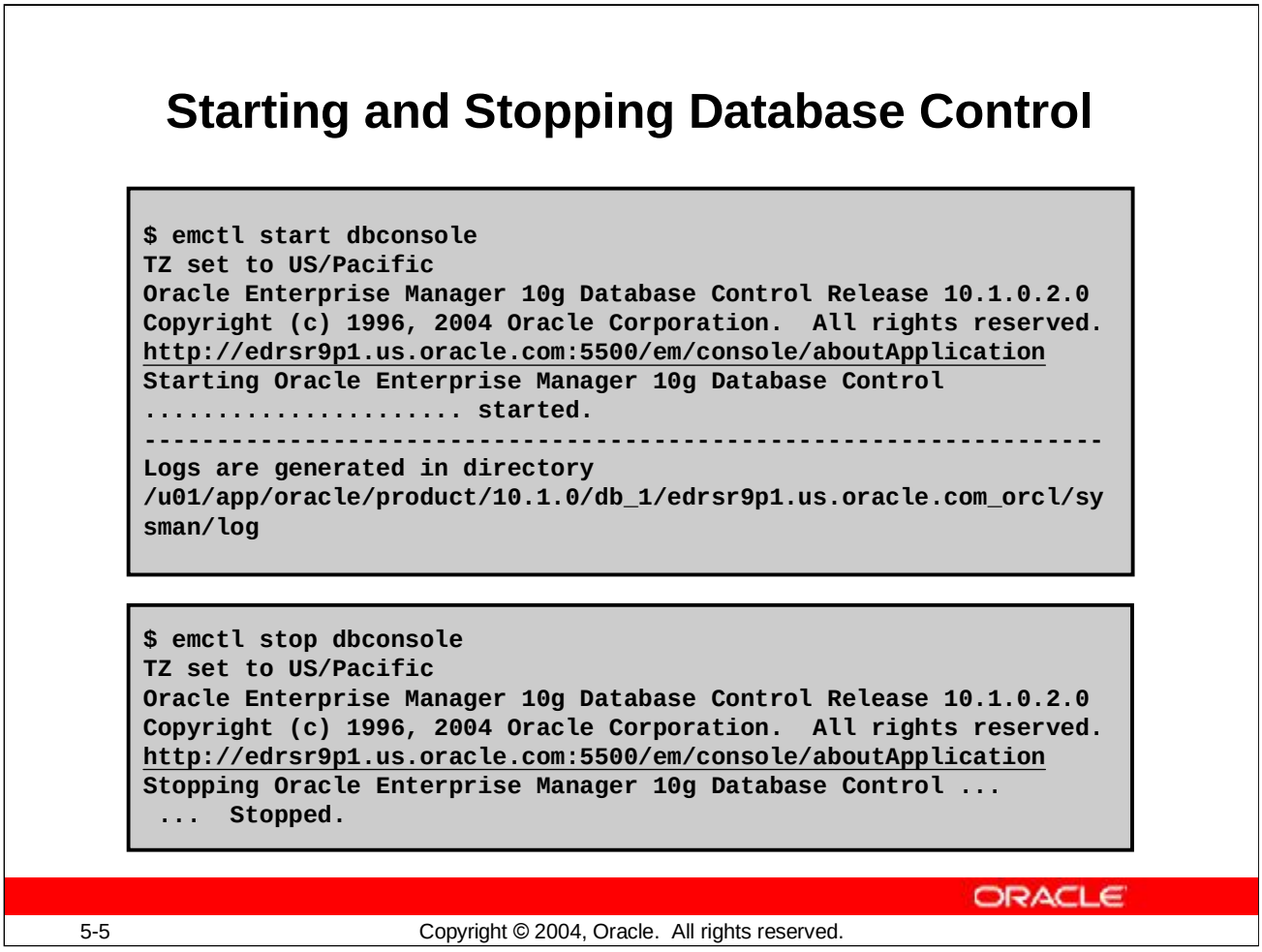
Management Framework

There are three major components of the Oracle Database 10*g* management framework:

- The database and instance being managed
- A listener that allows connections to the database
- The management interface. This may be either a management agent connecting this server to Oracle Enterprise Manager Grid Control, or the stand-alone Oracle Enterprise Manager Database Control.

Each of these components must be explicitly started before you can use the services of the component and should be shut down cleanly when shutting down the server hosting Oracle Database 10*g*.

The first component to be started is the management interface. After this is up and running, the management interface can be used to start the other components.



Starting and Stopping Database Control

For databases not connected to the Grid Control framework, Oracle provides a stand-alone management console called Database Control. Each database managed with Database Control has a separate Database Control installation, and from any one Database Control you can manage only one database. Database Control requires that a `dbconsole` process be started prior to use. To start the dbconsole process:

```
emctl start dbconsole
```

To stop the `dbconsole` process:

```
emctl stop dbconsole
```

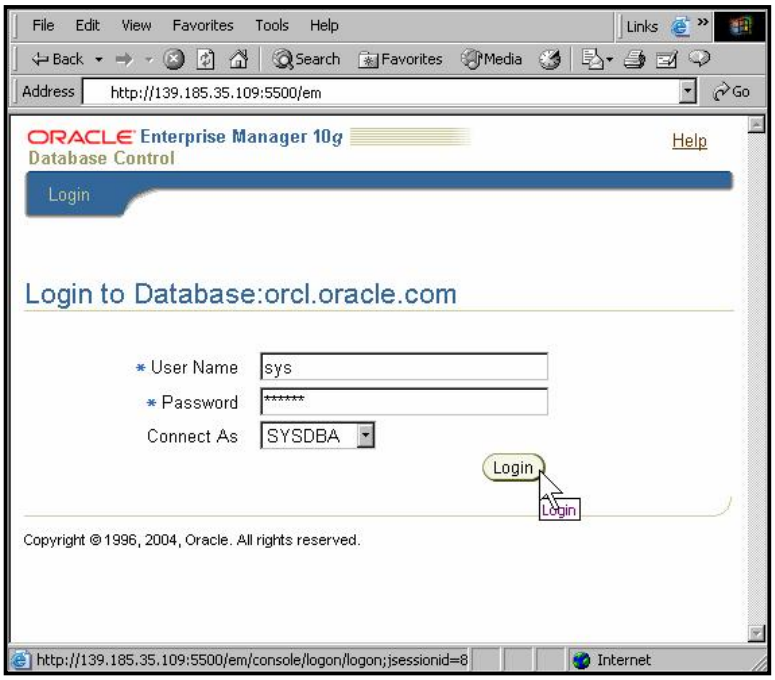
To view the status of the `dbconsole` process:

```
emctl status dbconsole
```

Note: You may need to navigate into your `$ORACLE_HOME/bin` directory if this directory is not in your OS path.

Database Control uses a server-side agent process. This agent process automatically starts and stops when the `dbconsole` process is started or stopped.

Accessing Database Control



ORACLE

Accessing Database Control

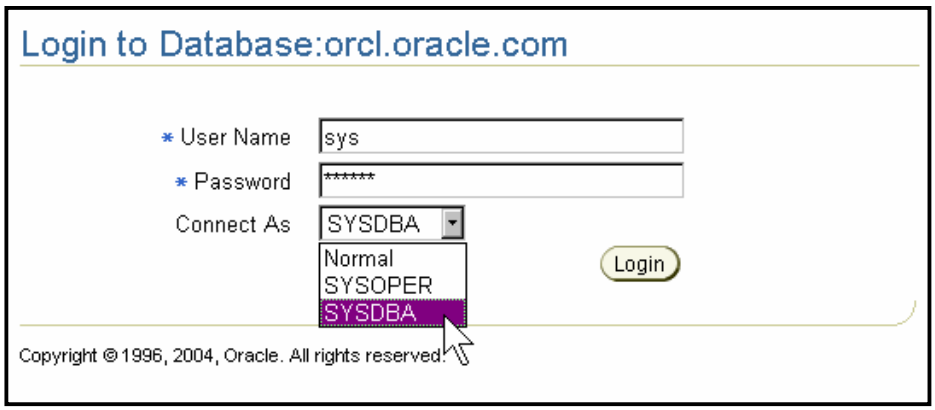
Open your Web browser and enter the following URL (the default port is 5500):

`http://hostname:portnumber/em`

If the database is up, Enterprise Manager displays the Database Control Login page. Log in to the database using a username that is authorized to access Database Control. This initially will be SYS, SYSMAN or SYSTEM. Use the password you specified for the account during the database installation.

If the database is down and needs to be started, Enterprise Manager displays the Startup/Shutdown and Perform Recovery page. If this is the case, click the Startup/Shutdown button. You are then prompted for the host and target database login usernames and passwords, which you must enter. For the database user and password, use SYS and the password you specified during installation. Click OK to start the database. In the Confirmation page, click YES to start the database in open mode.

SYSOPER and SYSDBA



ORACLE

SYSOPER and SYSDBA

SYSOPER: Is a special database administration role that permits a database administrator to perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, and RECOVER, and includes RESTRICTED SESSION privileges. When you connect with SYSDBA privileges, you are in the schema owned by SYS.

SYSDBA: Is a special database administration role that contains every system privilege with ADMIN OPTION and SYSOPER system privileges. SYSDBA also permits CREATE DATABASE actions and incomplete recovery. When you connect as SYSOPER, you are in the public schema.

More details on user management will be covered in the lesson titled "Administering Users."

Database Home Page

ORACLE

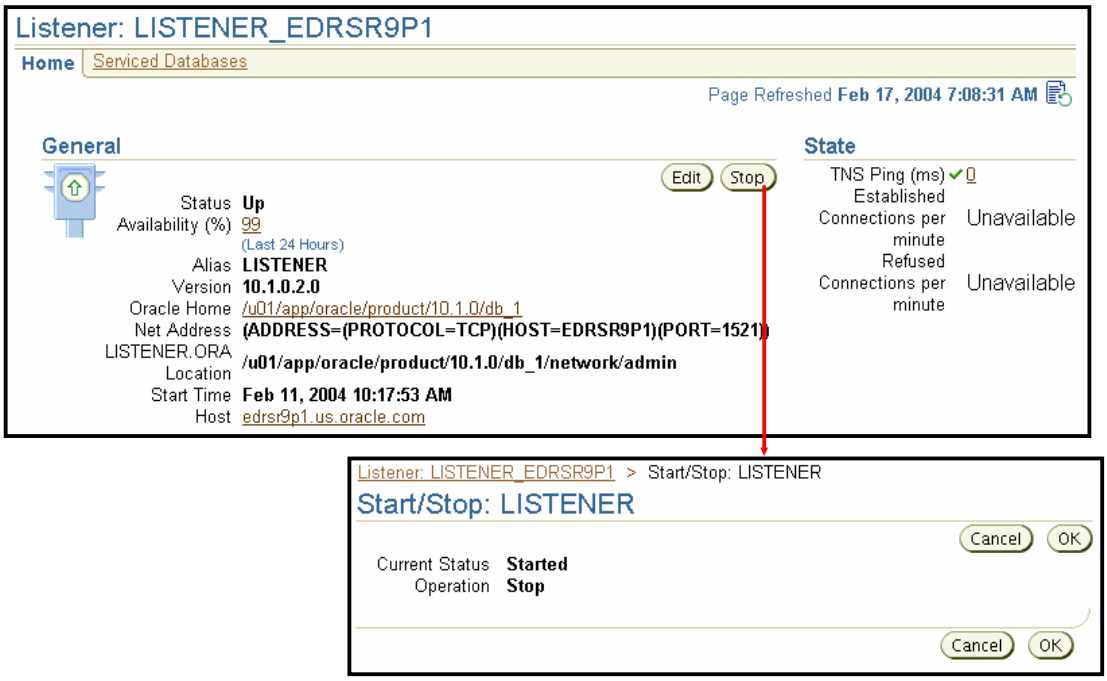
Database Home Page

The property pages across the top of the Database home page enable you to access performance, administration, and maintenance pages for managing your database. The various sections of the Database home page, and related links, provide a wealth of information about the database's environment and health.

To grant management access to other database users, use the following procedure:

1. Start your Web browser and log in to Database Control as the SYS or SYSTEM database user.
2. Click Setup at the top of the Database home page.
3. Click Administrators in the left navigation bar.
4. Click Create to create a new Enterprise Manager user by assigning the management privileges to an existing database user.
5. Click the flashlight icon next to the Name field and select an existing database user from the pop-up window.
6. Enter the password for the selected user and click Finish.

Changing the Listener Status



ORACLE

Changing the Listener Status

From the Database home page click the listener name to open the Listener home page. Click Stop to stop the listener if it is running or Start to start the listener if it is not running. You must log on to the host as an OS user if you have not done so already. This will be the OS user that starts or stops the listener.

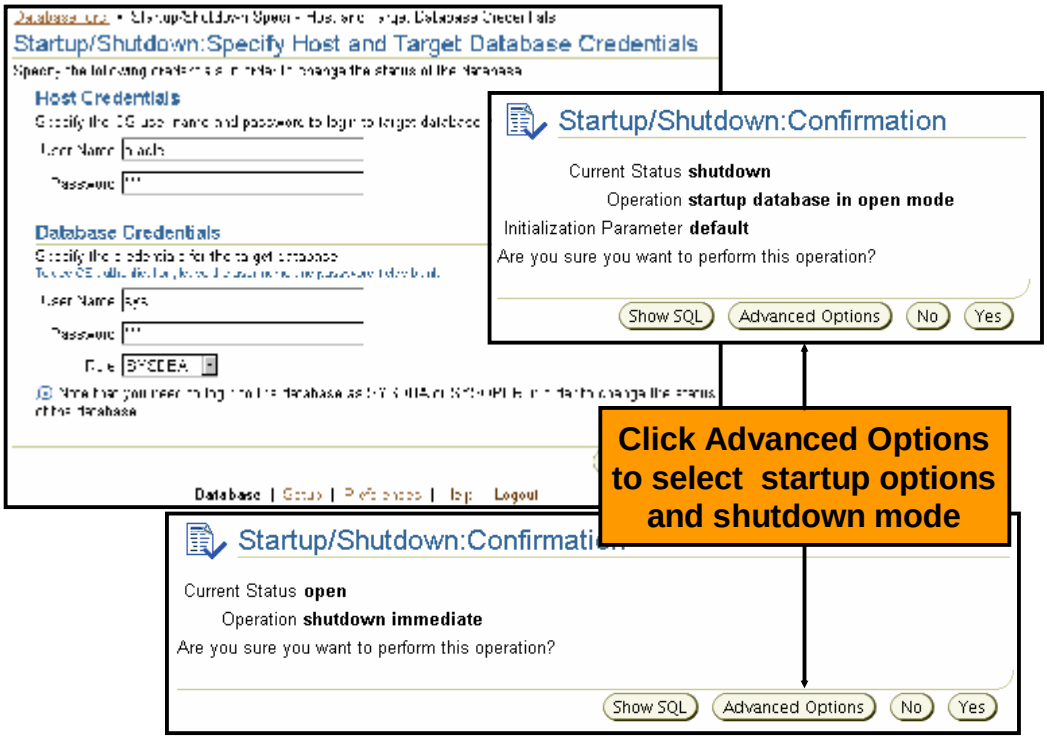
The command line method for starting, stopping, and seeing the status of the listener is:

```
lsnrctl START [listener_name]
lsnrctl STOP [listener_name]
lsnrctl STATUS [listener_name]
```

Where *listener_name* is the name of the listener defined in the `listener.ora` file. It is not necessary to identify the listener if you are using the default listener, named `LISTENER`.

The `STATUS` command provides basic status information about a listener, including a summary of listener configuration settings, the listening protocol addresses, and a summary of services registered with the listener.

Startup and Shutdown

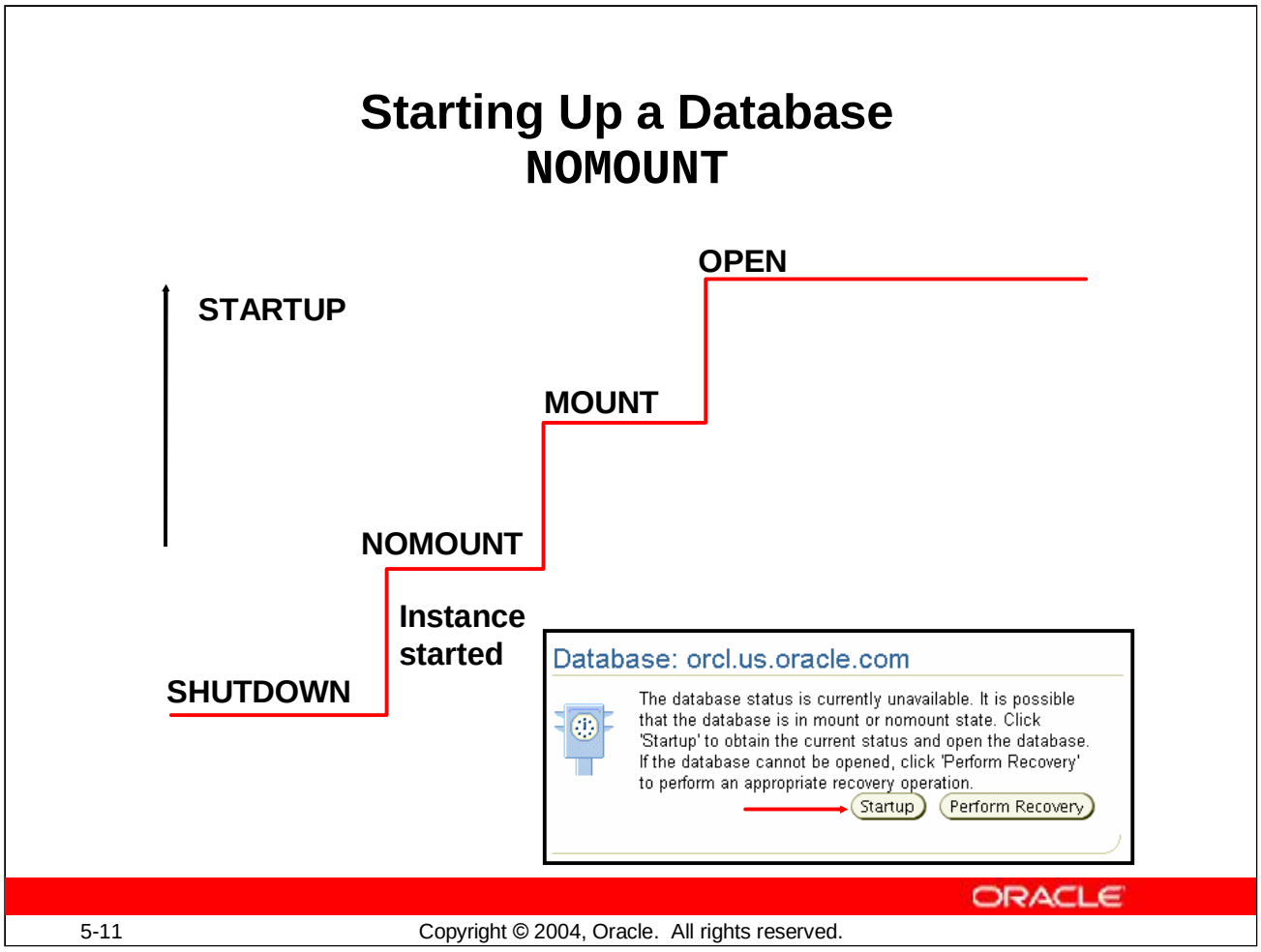


ORACLE

Startup and Shutdown

When you click either startup or shutdown you may be prompted for credentials that will be used for logging into both the host (the computer the database resides on) and the database itself.

You can then click Advanced Options to change any startup options or the shutdown mode as needed. Also you can click Show SQL to see the SQL statement that will be used for the startup or shutdown.



Starting Up a Database

When starting the database, you select the state in which it starts. The following scenarios describe different stages of starting up an instance.

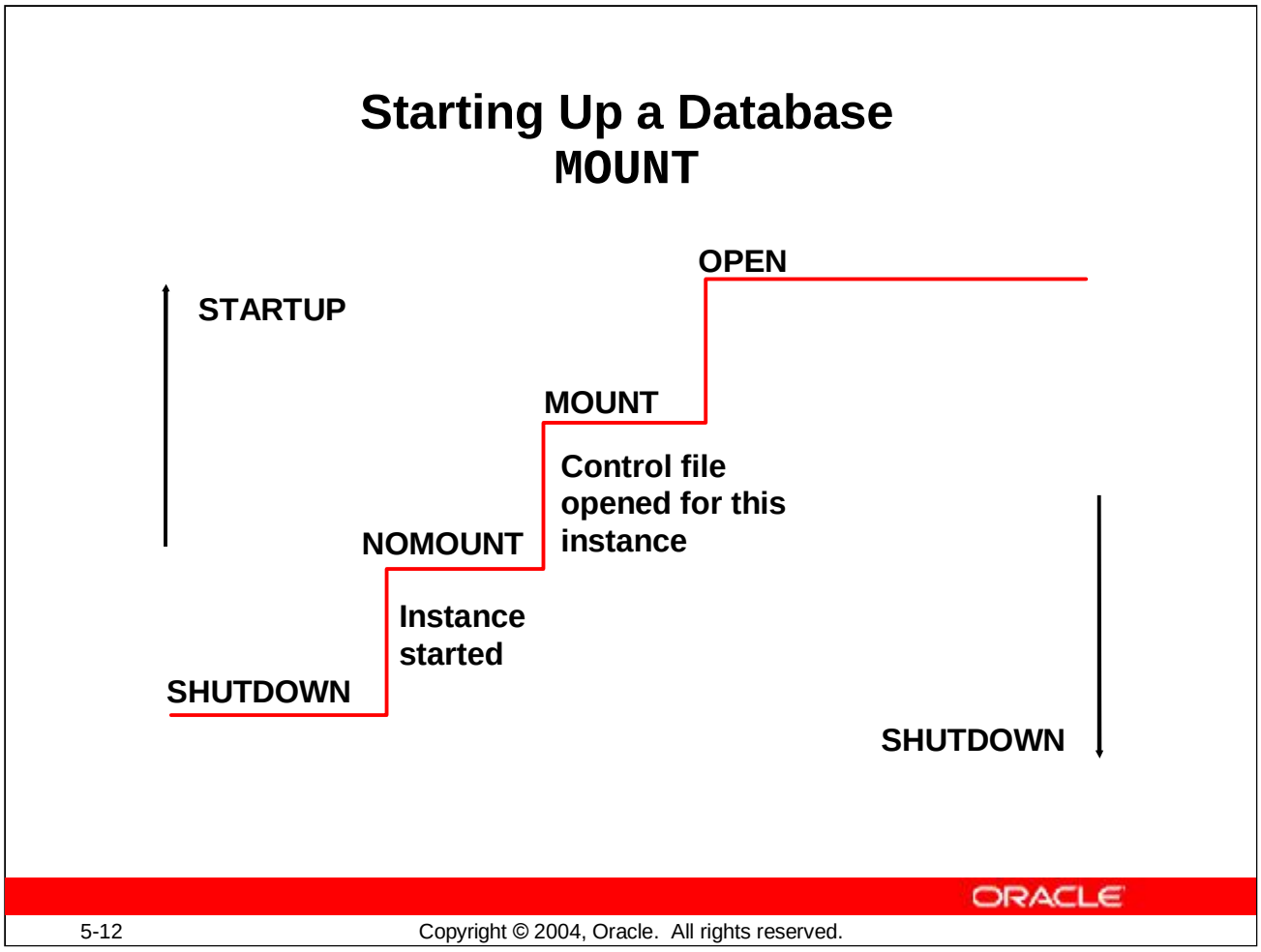
Starting the Instance (NOMOUNT)

An instance is started in the `NOMOUNT` stage only during database creation or the re-creation of control files.

Starting an instance includes the following tasks:

- Reading the initialization file from `$ORACLE_HOME/dbs` in the following order:
  - First `spfileSID.ora`
  - If not found then, `spfile.ora`
  - If not found then, `initSID.ora`

  Specifying the `PFILE` parameter with `STARTUP` overrides the default behavior.
- Allocating the SGA
- Starting the background processes
- Opening the `alertSID.log` file and the trace files



Starting Up a Database (continued)

Mounting the Database (MOUNT)

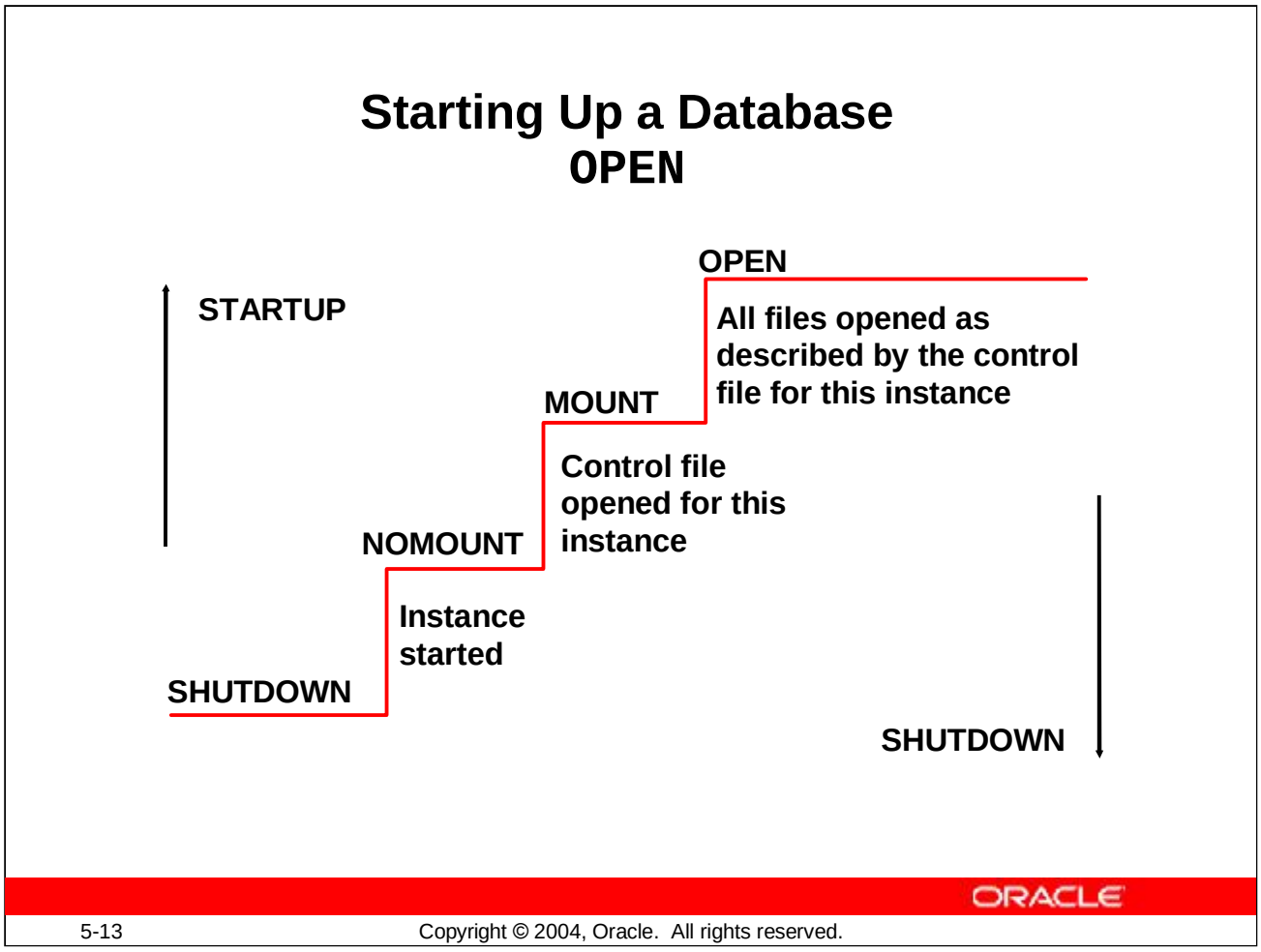
To perform specific maintenance operations, you start an instance and mount a database but do not open the database.

For example, the database must be mounted but not open during the following tasks:

- Renaming data files (data files for an offline tablespace can be renamed when the database is open)
- Enabling and disabling online redo log file archiving options
- Performing full database recovery

Mounting a database includes the following tasks:

- Associating a database with a previously started instance
- Locating and opening the control files specified in the parameter file
- Reading the control files to obtain the names and status of the data files and online redo log files. However, no checks are performed to verify the existence of the data files and online redo log files at this time.



Starting Up a Database (continued)

Opening the Database (OPEN)

Normal database operation means that an instance is started and the database is mounted and open. With normal database operation, any valid user can connect to the database and perform typical data access operations.

Opening the database includes the following tasks:

- Opening the online data files
- Opening the online redo log files

If any of the data files or online redo log files are not present when you attempt to open the database, the Oracle server returns an error.

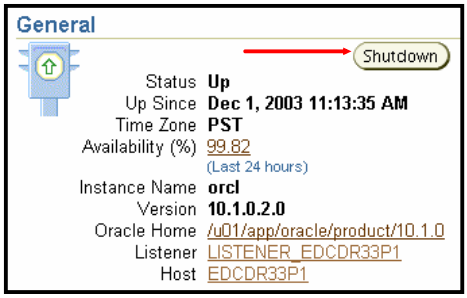
During this final stage, the Oracle server verifies that all the data files and online redo log files can be opened and checks the consistency of the database. If necessary, the SMON background process initiates instance recovery.

Shutting Down the Database

Shutdown Mode	A	I	T	N
Allow new connections	No	No	No	No
Wait until current sessions end	No	No	No	Yes
Wait until current transactions end	No	No	Yes	Yes
Force a checkpoint and close files	No	Yes	Yes	Yes

Shutdown mode:

- **A = ABORT**
- **I = IMMEDIATE**
- **T = TRANSACTIONAL**
- **N = NORMAL**



ORACLE

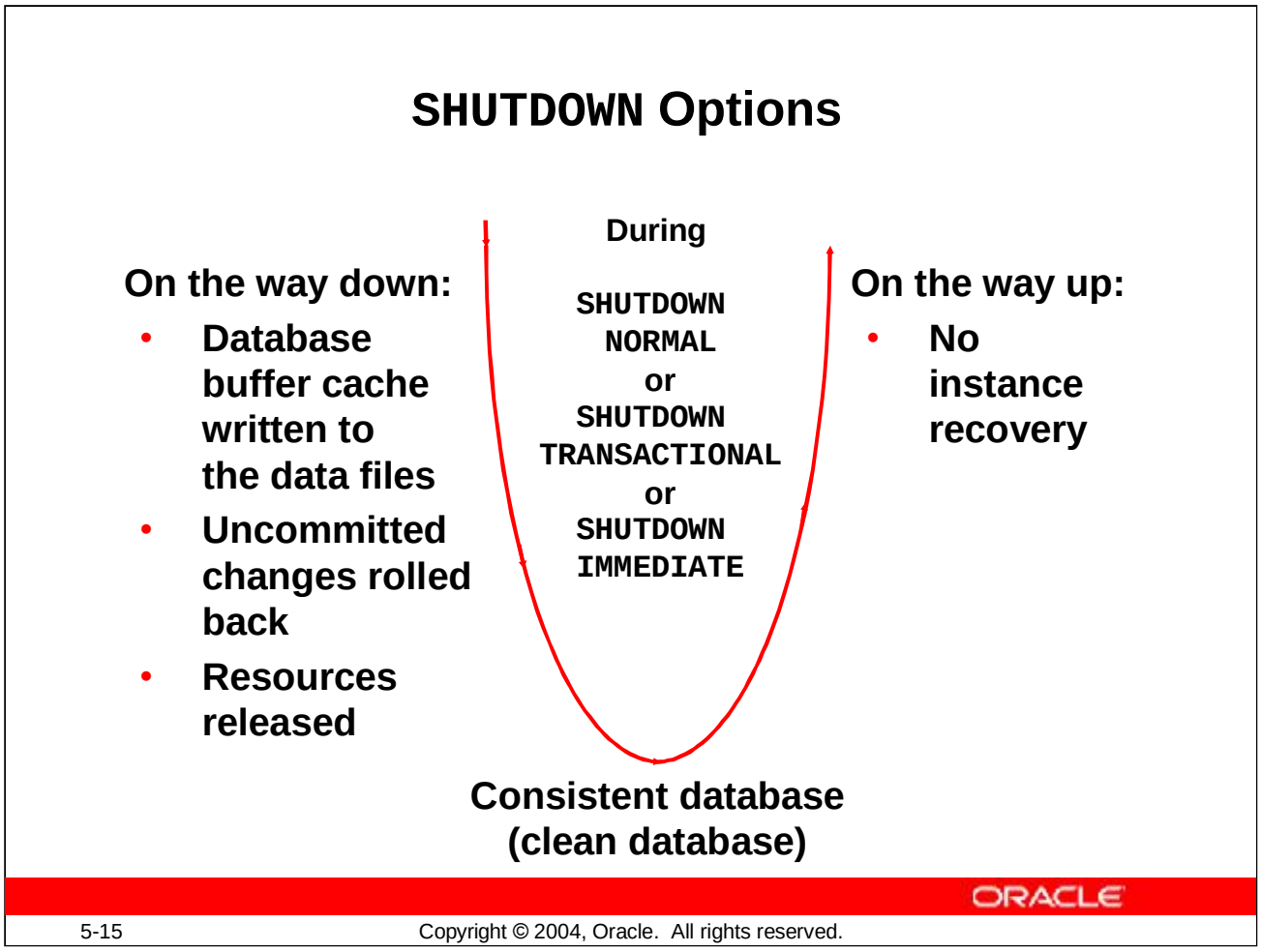
Shutting Down the Database

Shut down the database to make operating system offline backups of all physical structures and to have modified static initialization parameters take effect when restarted.

To shut down an instance you must connect as SYSOPER or SYSDBA and use the following command:

```
SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT ]
```

This is the command that gets generated when you click Shutdown and confirm your intention on the subsequent page.



SHUTDOWN Options

SHUTDOWN NORMAL

Normal is the default shutdown mode. Normal database shutdown proceeds with the following conditions:

- No new connections can be made.
- The Oracle server waits for all users to disconnect before completing the shutdown.
- Database and redo buffers are written to disk.
- Background processes are terminated, and the SGA is removed from memory.
- Oracle closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.

SHUTDOWN TRANSACTIONAL

A transactional shutdown prevents clients from losing work. A transactional database shutdown proceeds with the following conditions:

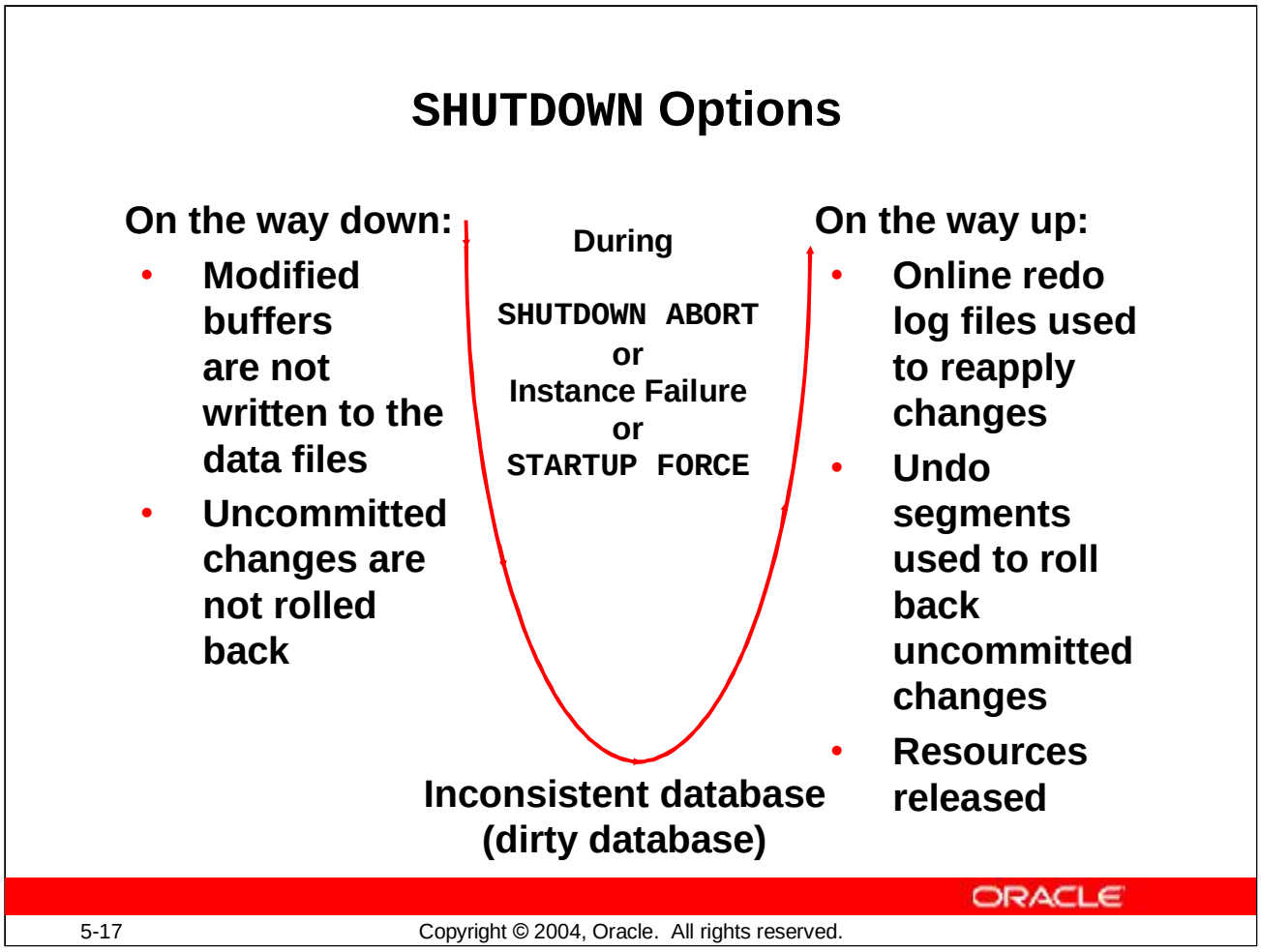
- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have finished, a shutdown occurs immediately.
- The next startup does not require an instance recovery.

SHUTDOWN Options (continued)

SHUTDOWN IMMEDIATE

Immediate database shutdown proceeds with the following conditions:

- Current SQL statements being processed by Oracle are not completed.
- The Oracle server does not wait for the users, who are currently connected to the database, to disconnect.
- Oracle rolls back active transactions and disconnects all connected users.
- Oracle closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.



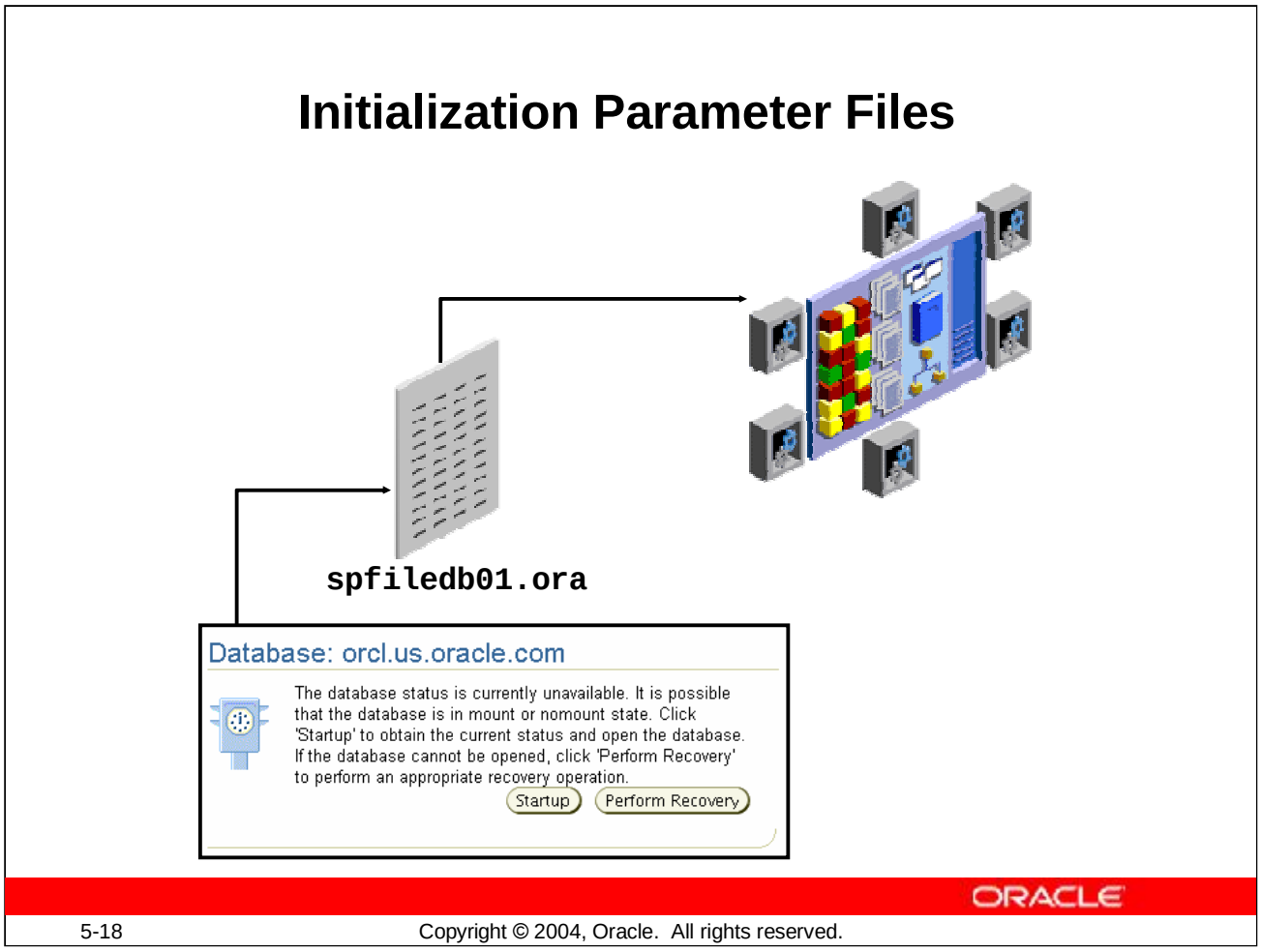
SHUTDOWN Options (continued)

SHUTDOWN ABORT

If the NORMAL and IMMEDIATE shutdown options do not work, you can abort the current database instance. Aborting an instance proceeds with the following conditions:

- Current SQL statements being processed by the Oracle server are immediately terminated.
- Oracle does not wait for users currently connected to the database to disconnect.
- Database and redo buffers are not written to disk.
- Uncommitted transactions are not rolled back.
- The instance is terminated without closing the files.
- The database is not closed or dismounted.
- The next startup requires instance recovery, which occurs automatically.

Note: It is not advisable to back up a database that is in an inconsistent state.



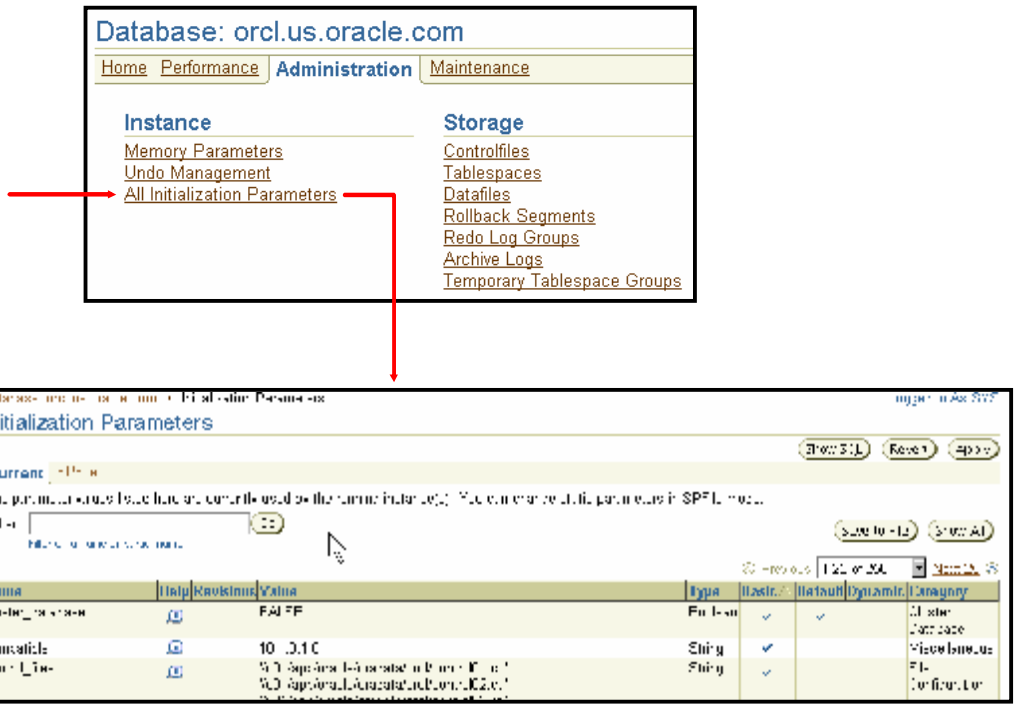
Initialization Parameter Files

To start an instance, Oracle must read either an initialization parameter file or a server parameter file. These files contain a list of configuration parameters for that instance and database. Oracle traditionally stores initialization parameters in a text initialization parameter file. You can also choose to maintain initialization parameters in a binary server parameter file (SPFILE).

Initialization parameters stored in a server parameter file are persistent, in that any changes made to the parameters while an instance is running can persist across instance shutdown and startup. Initialization parameters are divided into two groups: basic and advanced. In the majority of cases, it is necessary to set and tune only the basic parameters to get reasonable performance. In rare situations, modification to the advanced parameters may be needed for optimal performance.

Oracle suggests that you create a server parameter file (SPFILE) as a dynamic means of maintaining initialization parameters. By using a server parameter file you can store and manage your initialization parameters persistently in a server-side disk file.

Viewing Initialization Parameters



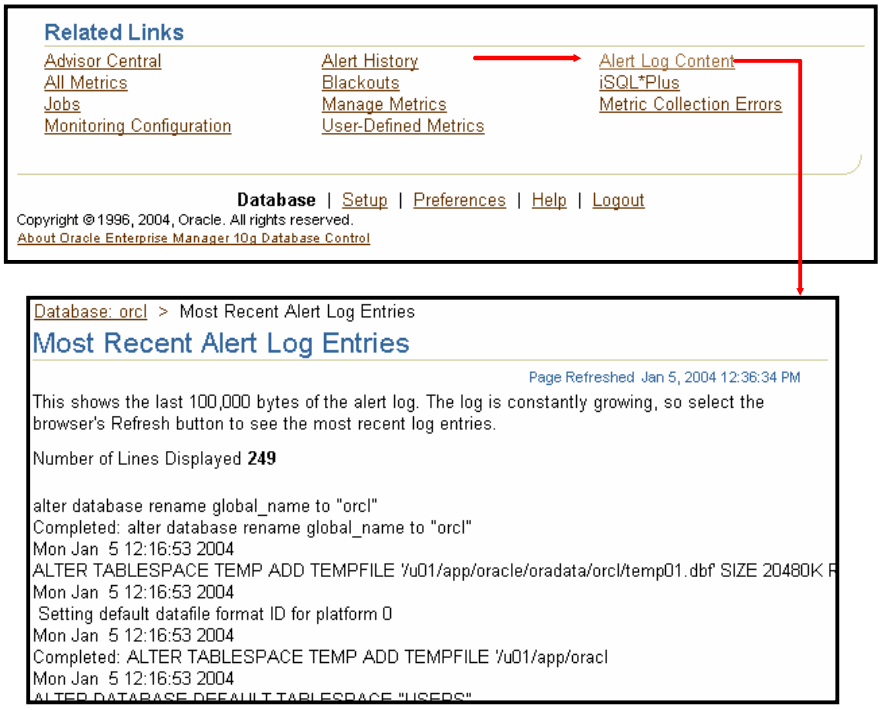
ORACLE

Viewing Initialization Parameters

The Oracle Database provides a number of initialization parameters to optimize its operation in diverse environments. Only a few of these parameters must be explicitly set because the default values are adequate in the majority of cases. There are 28 basic parameters.

The advanced parameters are preserved to allow expert DBAs to adapt the behavior of the Oracle Database to meet unique requirements without overwhelming those who have no such requirements.

Viewing the Alert Log



Viewing the Alert Log

Each database also has an `alert_sid.log`. The file is on the server with the database and is stored in the directory specified with the initialization parameter `background_dump_dest`. The alert file of a database is a chronological log of messages and errors, including the following:

- All internal errors (`ORA-600`), block corruption errors (`ORA-1578`), and deadlock errors (`ORA-60`) that occur
- Administrative operations, such as the SQL statements `CREATE`, `ALTER`, `DROP DATABASE`, `TABLESPACE`, `ROLLBACK SEGMENT` and the Enterprise Manager or SQL*Plus statements `STARTUP`, `SHUTDOWN`, `ARCHIVE LOG`, and `RECOVER`
- Several messages and errors relating to the functions of shared server and dispatcher processes
- Errors during the automatic refresh of a materialized view

EM monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and informative messages. Also the file can grow to an unmanageable size if left alone. You should make a back up of the alert file occasionally and delete the current alert file. When the database attempts to write to the alert file again, the database will then re-create a new alert file.

Summary

In this lesson, you should have learned how to:

- **Start and stop *i*SQL*Plus**
- **Start and stop Enterprise Manager Database Console**
- **Start and stop the listener**
- **Start up and shut down the database**

ORACLE

Practice 5:
Controlling the Database

This practice covers the following topics:

- **Connecting to the database using SQL*Plus and *i*SQL*Plus**
- **Stopping and starting Enterprise Manager Database Control**
- **Stopping and starting the database**
- **Stopping and starting *i*SQL*Plus**

ORACLE

Practice 5: Controlling the Database

Background: Your system administrator asks that you stop all Oracle services in preparation for system maintenance. After maintenance is completed, restart all Oracle services..

Tasks:

- Start and stop the Oracle Listener
- Start and stop Oracle Database 10*g* instance
- Start and stop *i*SQL*Plus
- Start and stop the Oracle Enterprise Manager Database Control

1. Connect to Database Control.
2. Stop the Oracle Listener using Database Contol.
3. Shut down the database instance using Database Control.
4. Stop *i*SQL*Plus.
5. Stop Database Control.
6. Start the Oracle Listener using command-line tools.
7. Start Database Control.
8. Start *i*SQL*Plus.
9. Start the Oracle Database 10*g* instance.
10. View information in the instance's alert log.
11. View initialization parameters.

6

Storage Structures

Objectives

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

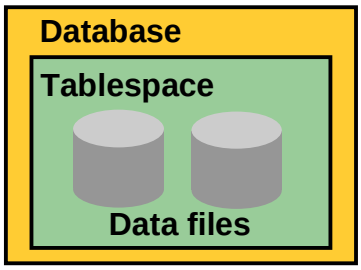
- **Define the purpose of tablespaces and data files**
- **Create tablespaces**
- **Manage tablespaces**
- **Obtain tablespace information**

ORACLE

Tablespaces and Data Files

Oracle stores data logically in tablespaces and physically in data files.

- **Tablespaces:**
  - **Can belong to only one database at a time**
  - **Consist of one or more data files**
  - **Are further divided into logical units of storage**
- **Data files:**
  - **Can belong to only one tablespace and one database**
  - **Are a repository for schema object data**



ORACLE

Tablespaces and Data Files

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

- An Oracle database consists of one or more logical storage units called tablespaces, which collectively store all of the database's data.
- Each tablespace in an Oracle database consists of one or more files called data files, which are physical structures that conform with the operating system on which Oracle is running.
- A database's data is collectively stored in the data files that constitute each tablespace of the database. For example, the simplest Oracle database would have one tablespace and one data file. Another database can have three tablespaces, each consisting of two data files (for a total of six data files). A single database could potentially have as many as 65,535 data files.

Space Management in Tablespaces

- **Locally managed tablespace:**
  - **Free extents are managed in the tablespace.**
  - **Bitmap is used to record free extents.**
  - **Each bit corresponds to a block or group of blocks.**
  - **Bit value indicates free or used.**
- **Dictionary-managed tablespace:**
  - **Free extents are managed by the data dictionary.**
  - **Appropriate tables are updated when extents are allocated or deallocated.**

ORACLE

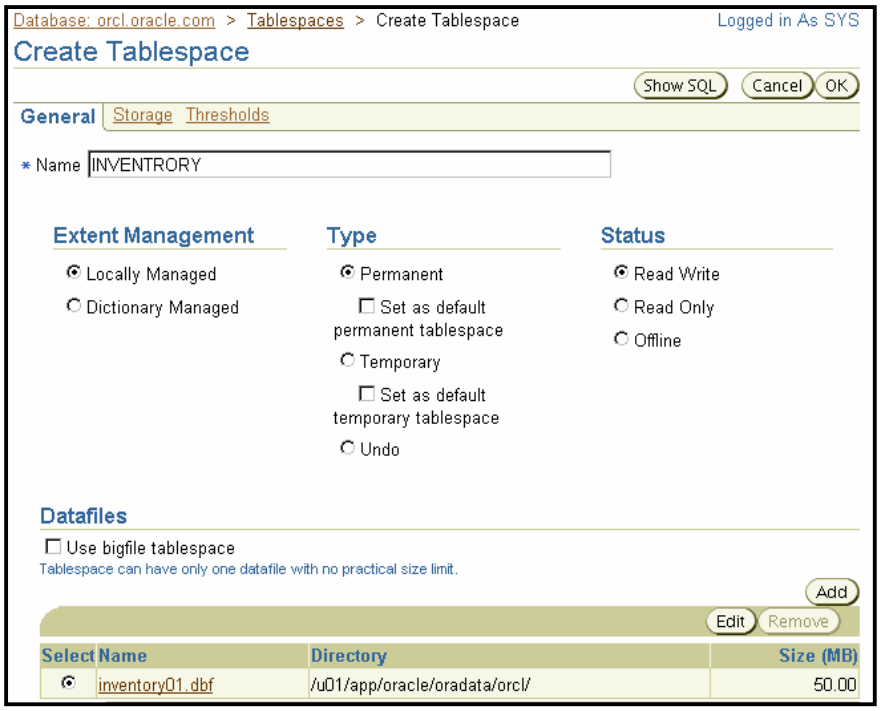
Space Management in Tablespaces

Tablespaces allocate space in extents. Tablespaces can be created to use one of the following two different methods of keeping track of free and used space:

Locally managed tablespaces: The extents are managed within the tablespace via bitmaps. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, the Oracle server changes the bitmap values to show the new status of the blocks.

Dictionary-managed tablespaces: The extents are managed by the data dictionary. The Oracle server updates the appropriate tables in the data dictionary whenever an extent is allocated or deallocated. This is for backward compatibly; you should use locally managed for all tablespaces.

Creating a New Tablespace



ORACLE

Creating a New Tablespace

To create a tablespace, perform the following steps:

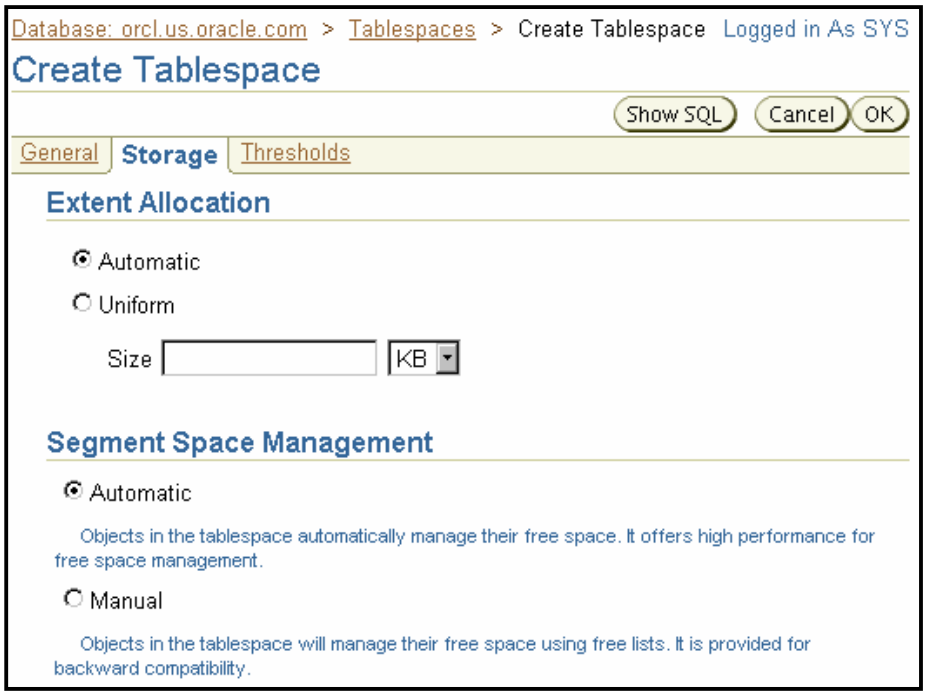
1. Navigate to the Tablespaces page. Go to the Administration tab, then click Tablespaces under the Storage heading.
2. Click the Create button.
   Note: If you want to create a tablespace that is like an existing tablespace, select an existing tablespace and select Create Like from the Actions menu. Click Go.
   The Create Tablespace General page appears.
3. Enter a name for the tablespace.
4. Under the Extent Management heading, select Locally Managed. The extents of a locally managed tablespace are managed efficiently within the tablespace by the Oracle database server. For a dictionary managed tablespace you must more actively manage extents and data dictionary access is required for tracking them. Dictionary managed tablespaces are being deprecated. Oracle does not suggest their use.
5. Under the Type heading, select Permanent. Permanent tablespaces store permanent database objects created by the system or users.
6. Under the Status heading, select Read Write. Read/write status means users can read and write to the tablespace after it is created. This is the default.

Creating a New Tablespace (continued)

7. In the Datafiles region of the page click Add to add datafiles to the tablespace, a tablespace must have at least one file. Bigfile tablespaces are used with ultra large databases where Oracle's Automatic Storage Management or other logical volume managers support striping or RAID, and dynamically extensible logical volumes.
8. In the Add Datafiles page, enter a file name. Accept the defaults for the File Directory and File Size.
9. Under the Storage region, select "Automatically extend datafile when full (AUTOEXTEND)" and specify an amount in the Increment field by which you want to extend the data file each time it fills. Leave the Maximum File Size set to Unlimited. Click OK. You are returned to the Create Tablespace General page.
10. Click the Storage tab. The Create Tablespace Storage page appears.
11. Accept all of the defaults on the Storage page.
12. Click the Thresholds tab to open the Thresholds page. This page enables you to set monitored thresholds for space usage. You receive advice and an option for action when the threshold is reached.
13. After specifying thresholds click OK to add the tablespace. You are returned to the Tablespaces page where you receive a confirmation of the creation of the tablespace. You can view your new tablespace in the Results section.

Note: These steps are intended to show you how to quickly create a tablespace for most situations. You may need to change some options depend on your particular system requirements.

Storage for Locally Managed Tablespaces



ORACLE

Storage for Locally Managed Tablespaces

Extents within a locally managed tablespace can be allocated in one of two ways:

- Automatic: Also called autoallocate, specifies that the size of the extents within the tablespace are system managed. You cannot specify an extent size. You cannot specify automatic for a temporary tablespace.
- Uniform: Specifies that the tablespace is managed with uniform extents of a size you specify. The default size is 1 megabyte. All extents of temporary tablespaces are of uniform size, so this is optional for a temporary tablespace. You cannot specify uniform for an undo tablespace.

Segment space management within a locally managed tablespace:

- Automatic: Oracle uses bitmaps to manage the free space within segments. A bitmap, in this case, is a map that describes the status of each data block within a segment with respect to the amount of space in the block available for inserting rows. As more or less space becomes available in a data block, its new state is reflected in the bitmap. Bitmaps allow Oracle to manage free space more automatically, and thus, this form of space management is called automatic segment-space management.

Storage for Locally Managed Tablespaces (continued)

- Manual: This tells Oracle that you want to use free lists for managing free space within segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space within segments is called manual segment-space management because of the need to specify and tune the `PCTUSED`, `FREELISTS`, and `FREELIST GROUPS` storage parameters for schema objects created in the tablespace. This is supported for backward compatibility.

Advantages of Locally Managed Tablespaces

Locally managed tablespaces have the following advantages over dictionary-managed tablespaces:

- Local management avoids recursive space management operations. This can occur in dictionary-managed tablespaces if consuming or releasing space in an extent results in another operation that consumes or releases space in an undo segment or data dictionary table.
- Because locally managed tablespaces do not record free space in data dictionary tables, they reduce contention on these tables.
- Local management of extents automatically tracks adjacent free space, eliminating the need to coalesce free extents.
- The sizes of extents that are managed locally can be determined automatically by the system.
- Changes to the extent bitmaps do not generate undo information because they do not update tables in the data dictionary (except for special cases such as tablespace quota information).

Note: If you are managing a database that has dictionary managed tablespaces and you want to convert them to locally managed, use the `DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL` procedure to do this. For details on the use of this procedure see the *PL/SQL Packages and Types Reference* and the *Database Administrator's Guide*.

Logging

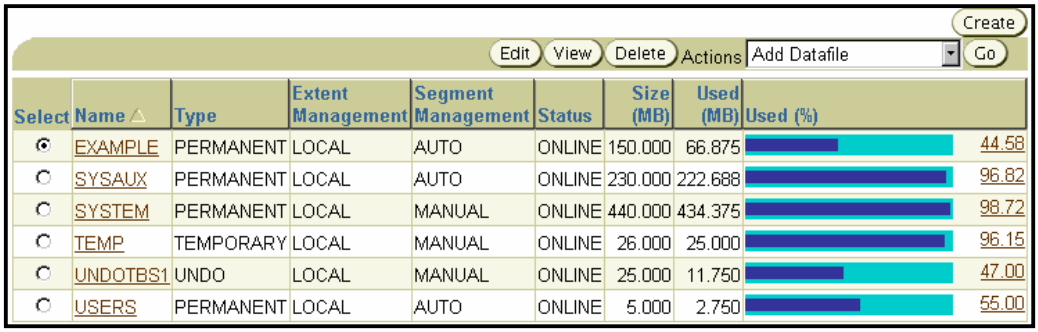
When changes are made to objects in the tablespace, the change is logged in to what is called the redo stream. This redo stream starts in memory, the is written to the online redo log files and may be written to archive log files. You can turn this processes off. If you do turn off logging, the objects in this tablespace will be unrecoverable in the event of any kind of failure.

Thresholds

Use the Tablespace Thresholds tab to set the space used thresholds for the current database tablespace. You can choose to use the default usage thresholds for tablespaces for the database or you can specify the threshold for the percentage of space used for the current tablespace by entering the values for the Warning and Critical fields. You can also disable space used thresholds entirely for this tablespace. Thresholds are covered in more detail in lesson 15.

Tablespaces in the Preconfigured Database

- **SYSTEM**
- **SYSAUX**
- **TEMP**
- **UNDOTBS1**
- **USERS**
- **EXAMPLE**



Select	Name	Type	Extent Management	Segment Management	Status	Size (MB)	Used (MB)	Used (%)
◉	EXAMPLE	PERMANENT	LOCAL	AUTO	ONLINE	150.000	66.875	44.58
○	SYSAUX	PERMANENT	LOCAL	AUTO	ONLINE	230.000	222.688	96.82
○	SYSTEM	PERMANENT	LOCAL	MANUAL	ONLINE	440.000	434.375	98.72
○	TEMP	TEMPORARY	LOCAL	MANUAL	ONLINE	26.000	25.000	96.15
○	UNDOTBS1	UNDO	LOCAL	MANUAL	ONLINE	25.000	11.750	47.00
○	USERS	PERMANENT	LOCAL	AUTO	ONLINE	5.000	2.750	55.00

ORACLE

Tablespaces in the Preconfigured Database

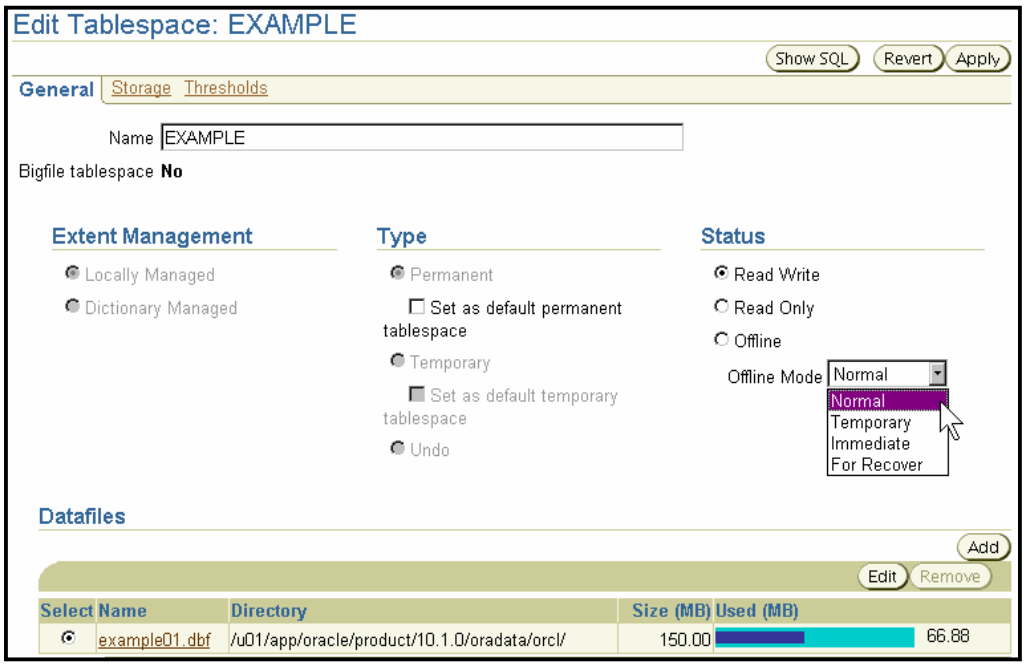
The following tablespaces are created for you in the preconfigured database:

- SYSTEM: The SYSTEM tablespace is used by the Oracle database server to manage the database. It contains the data dictionary and tables that contain administrative information about the database. These are all contained in the SYS schema, and can be accessed only by the user SYS, or other administrative users with the required privilege.
- SYSAUX: This is an auxiliary tablespace to the SYSTEM tablespace. Some components and products that used the SYSTEM tablespace or their own tablespaces in prior releases of Oracle, now use the SYSAUX tablespace. Every Oracle Database10*g* or higher-level database must have a SYSAUX tablespace.
- TEMP: This tablespace is used to store temporary tables and indexes when processing SQL statements. It would, for example, be used for sort work space. Every database should have a temporary tablespace that is assigned to users as their temporary tablespace. In the preconfigured database, the TEMP tablespace is specified as the default temporary tablespace. This means that if no temporary tablespace is specified when the user account is created, then Oracle assigns this tablespace to the user.

Tablespaces in the Preconfigured Database (continued)

- UNDOTBS1: This is the undo tablespace used by the database server to store undo information. Every database must have an undo tablespace that is created during database creation.
- USERS: This tablespace is used to store permanent user objects and data. In the preconfigured database, the USERS tablespace is the default tablespace for all objects created by nonsystem users. For the SYS and SYSTEM users (the system users), the default permanent tablespace remains SYSTEM.
- EXAMPLE: This tablespace contains the sample schemas that can be installed when you create the database. The sample schemas provide a common platform for examples. Oracle documentation and courseware contain examples based upon the sample schemas.

Altering a Tablespace



Altering a Tablespace

After you create a tablespace, you can later alter it in several ways as the needs of your system change.

Renaming: Simply enter a new name for the tablespace and click Apply.

Changing the Status: A tablespace can be in one of three different statuses or states. Depending on the type of tablespace, not all states may be available:

- Read Write: The tablespace is online and can be read from and written to.
- Read Only: Specify read-only to place the tablespace in transition read-only mode. In this state, existing transactions can complete (commit or roll back), but no further DML operations are allowed to the tablespace except for rollback of existing transactions that previously modified blocks in the tablespace. The tablespace is online while in the read-only state.You cannot make the SYSTEM or SYSAUX tablespace read-only.

Altering a Tablespace (continued)

- Offline: You can take an online tablespace offline so that this portion of the database is temporarily unavailable for general use. The rest of the database is open and available for users to access data. When you take it offline you have choices of how to do this:
  - Normal: A tablespace can be taken offline normally if no error conditions exist for any of the data files of the tablespace. Oracle takes a checkpoint for all data files of the tablespace as it takes them offline.
  - Temporary: A tablespace can be taken offline temporarily, even if there are error conditions for one or more files of the tablespace. Oracle takes offline the data files that are not already offline, checkpointing them as it does so. If no files are offline, but you use the temporary clause, media recovery is not required to bring the tablespace back online. However, if one or more files of the tablespace is offline because of write errors, and you take the tablespace offline temporarily, the tablespace requires recovery before you can bring it back online.
  - Immediate: A tablespace can be taken offline immediately, without Oracle taking a checkpoint on any of the data files. When you specify Immediate, media recovery for the tablespace is required before the tablespace can be brought online. You cannot take a tablespace offline immediately if the database is running in `NOARCHIVELOG` mode.
  - For Recover: The `FOR RECOVER` setting has been deprecated. The syntax is supported for backward compatibility.

Change the Size: You can add space to an existing tablespace by adding data files to the tablespace or you can change the size of an existing data file.

- To add a new data file to the tablespace click Add, and fill in the information about the data file on the Add Data File page. Note that the tablespace name is fixed.
- To change the size of an existing data file, select the data file in the Datafiles region of the Edit Tablespace page by clicking the name of the data file, or select the data file and click Edit. Then on the Edit Datafile page you can change the size of the data file. You can make the tablespace either larger or smaller. However you cannot make a data file smaller than the used space in the file; if you try you get the following error:

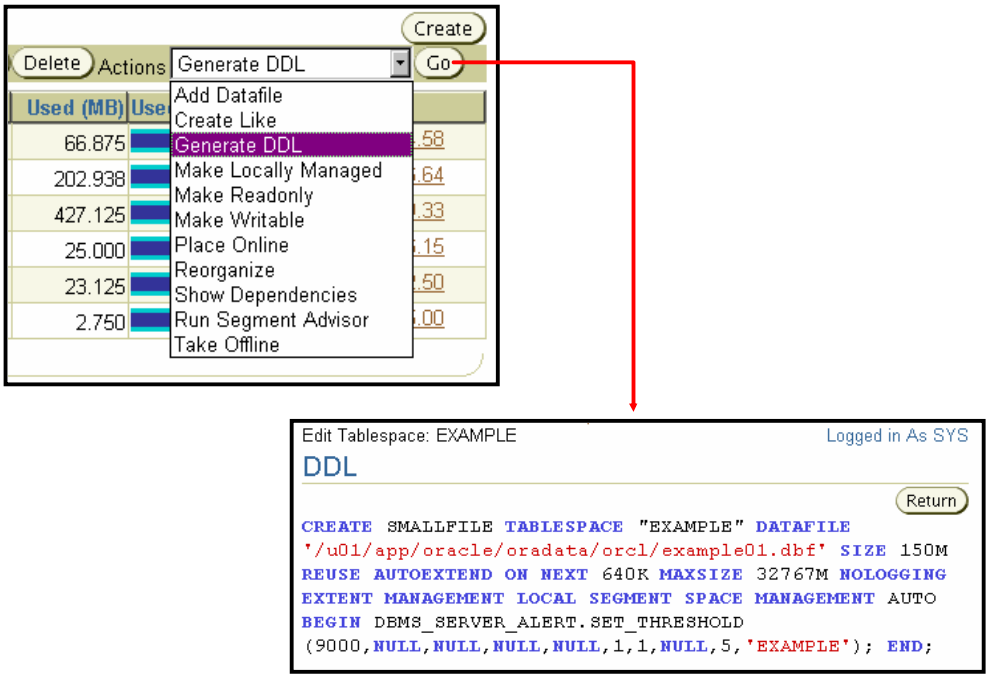
```
ORA-03297: file contains used data beyond requested RESIZE
value
```

Storage Options: Click Storage to change change the logging behavior of the tablespace.

Thresholds: Click Thresholds to can change the warning and critical used space alters for the tablespace. You have three options:

- Use Default Thresholds: This uses preset defaults, and you have the option of setting these defaults.
- Specify Thresholds: This allows you to set thresholds for this particular tablespace.
- Disable Thresholds: This turns off space usage alerts for this tablespace.

Actions with Tablespaces



ORACLE

Actions with Tablespaces

With the Actions Menu you can perform a verity of tasks with your tablespaces. Select a tablespace and then the action you want to perform:

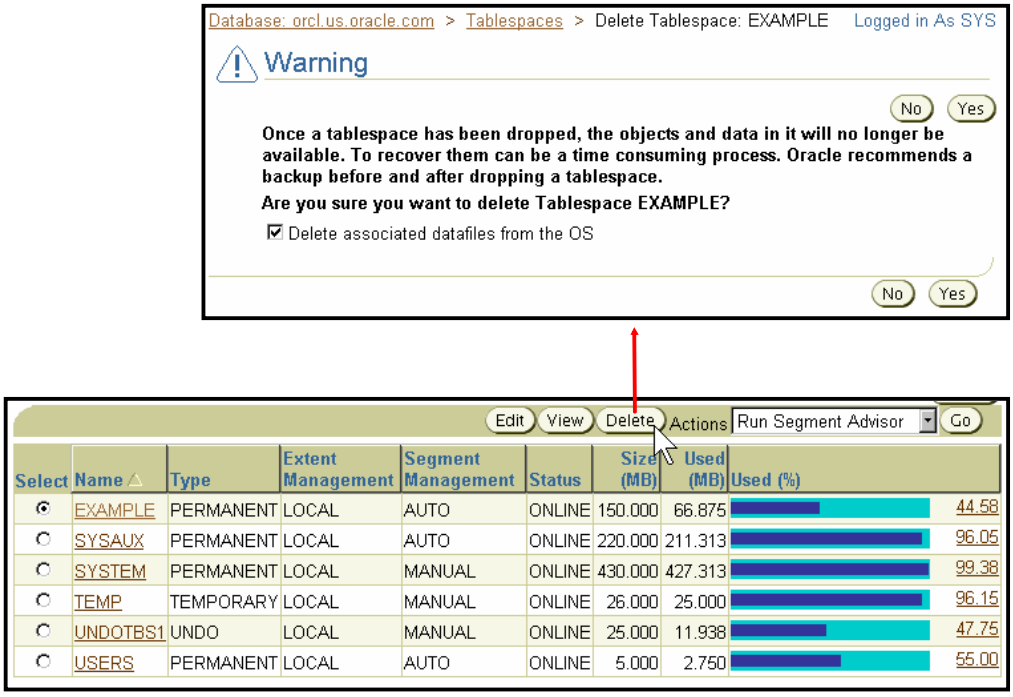
- Add Datafile: Adds a data file to the tablespace, which makes the tablespace larger.
- Create Like: Create another tablespace using the tablespace as a template.
- Generate DDL: Generate the DDL statement that creates the tablespace. This can then be copied and pasted into a text file for use as a script or for documentation purposes.
- Make Locally Managed: If the tablespace is currently dictionary managed, this will convert the tablespace to locally managed.
- Make Readonly: Stops all writes to the tablespace. Current transactions are allowed to compete but no new DML or other write activities are allowed to start on the tablespace.
- Make Writable: Allows DML and other write activities to be initiated on objects in the tablespace.

Actions with Tablespaces (continued)

- Place Online: If the tablespace is currently offline, this brings it back online.
- Reorganize: This starts the Reorganization Wizard, which you can use to move objects around within the tablespace to reclaim space that otherwise might not be used. This is a task that should be done during nonpeak usage of the objects in the tablespace.
- Show Dependencies: This shows objects that this tablespace depends on or objects that depend on this tablespace.
- Run Segment Advisor: The Segment Advisor helps you determine whether an object has space available for reclamation based on the level of space fragmentation within the object. At the tablespace level, advice is generated for every segment in the tablespace.
- Take Offline: If the tablespace is currently online this will make the tablespace unavailable. The tablespace is not deleted or drop, just unavailable.

Note: Not all actions are available for each tablespace. Depending on the type of tablespace selected, some actions cannot be performed. For example, you cannot take the `SYSTEM` tablespace offline, nor can you make an undo tablespace read-only.

Dropping Tablespaces



ORACLE

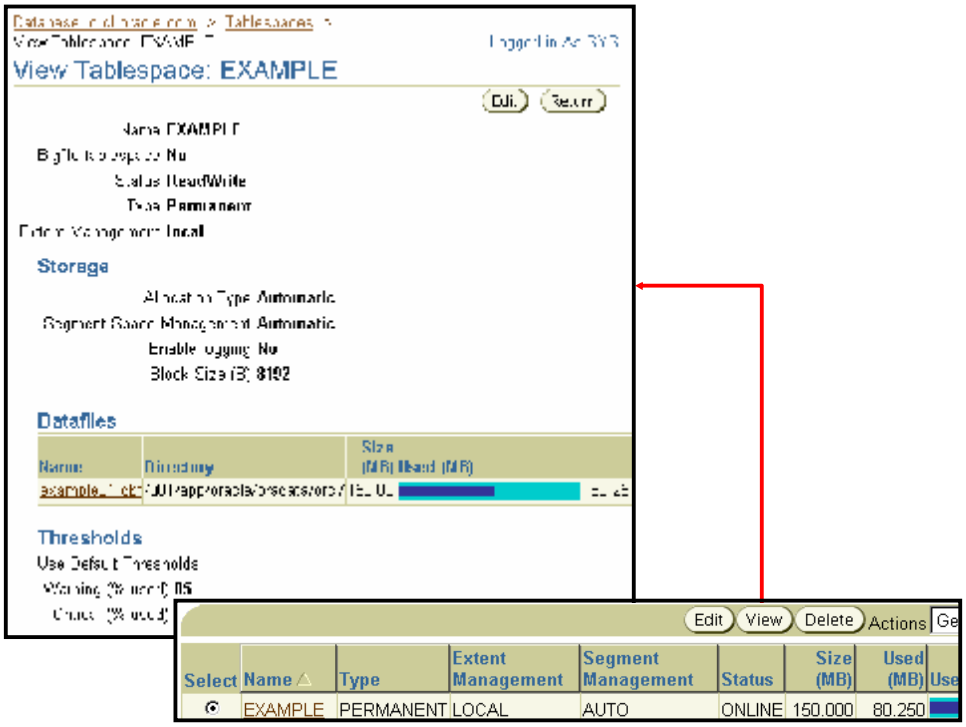
Dropping Tablespaces

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. You must have the DROP TABLESPACE system privilege to drop a tablespace.

When you drop a tablespace, the file pointers in the control file of the associated database are removed. You can optionally direct Oracle to delete the operating system files (data files) that constituted the dropped tablespace. If you do not direct Oracle to delete the data files at the same time that it deletes the tablespace, you must later use the appropriate commands of your operating system to delete them.

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 undo data that is needed to roll back uncommitted transactions, you cannot drop the tablespace. The tablespace can be online or offline, but it is best to take the tablespace offline before dropping it.

Viewing Tablespace Information



ORACLE

Viewing Tablespace Information

Click View to see information about the selected tablespace. In the View Tablespace page, you can also click edit to alter the tablespace.

Obtaining tablespace and data file information can also be obtained by querying the following:

- Tablespace information:
  - DBA_TABLESPACES
  - V$TABLESPACE
- Data file information:
  - DBA_DATA_FILES
  - V$DATAFILE
- Temp file information:
  - DBA_TEMP_FILES
  - V$TEMPFILE

Summary

In this lesson, you should have learned how to:

- **Use tablespaces to separate data**
- **Create various types of tablespaces**
- **Manage tablespaces**
- **Obtain tablespace information**

ORACLE

Practice 6:
Working with Tablespaces

This practice covers the following topics:

- **Creating tablespaces**
- **Gathering information about tablespaces**

ORACLE

Practice 6: Working with Tablespaces

Background: You will be supporting a new inventory application with your database. The installation instructions for the application instruct you to create a tablespace to hold data for the new application.

Tasks:

- Examine tablespaces and datafiles
- Create a tablespace
- Add space to an existing tablespace.

1. Use Database Control to view all tablespaces in your database. For each tablespace record the tablespace name, type, size, and percent used.
2. View all datafiles in your database. For each datafile record the file name, tablespace name, current size, autoextend status, and maximum file size (if autoextend is enabled).

Tablespace Name	Type	Size (MB)	Used %

File Name	Tablespace Name	Current Size (MB)	Auto extend?	Maximum Size (MB)

Practice 6: Working with Tablespaces (continued)

3. Create a new tablespace to hold information for the inventory application. Characteristics for the new tablespace are:

Tablespace Name: INVENTORY	File size: 50MB
Extent management: Local	AUTOEXTEND: disabled
Type: Permanent	Extent Allocation: Automatic
Status: Read Write	Segment space management: Auto
File name: inventory01.dbf	Enable Logging: Yes
File directory: default	Use default thresholds

7 Administering Users

Objectives

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

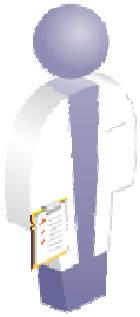
- **Create and manage database user accounts**
- **Create and manage roles**
- **Grant and revoke privileges**
- **Control resource usage by users**

ORACLE

Database User Accounts

Each database user account has a:

- **Unique username**
- **Authentication method**
- **Default tablespace**
- **Temporary tablespace**
- **User profile**



ORACLE

Database User Accounts

To access the database a user must specify a valid database user account and successfully authenticate as required by that user account.

In some systems, each database user has his or her own database account. In others, many users share a common database account. Regardless of which model your database follows, each user account will have:

- A unique username. Usernames cannot exceed 30 characters, cannot contain special characters, and must start with a letter.
- An authentication method. The most common authentication method is a password, but Oracle Database 10*g* supports several other authentication methods including biometric, certificate, and token authentication.
- A default tablespace. A place where the user will create objects if the user does not specify some other tablespace. Note that having a default tablespace does not imply the user has the *privilege* of creating objects in that tablespace nor a *quota* of space within that tablespace in which to create objects. Both these are granted separately.
- A temporary tablespace. A place where the user can create temporary objects such as sorts and temporary tables.
- A user profile. A set of resource and password restrictions assigned to the user.

Creating a User

Select Users from the Administration properties page. Click the Create button.

ORACLE

Creating a User

- In Oracle Enterprise Manager you can manage the list of database users that are allowed to access the current database through the Users property sheet. You can use the property sheet to create, delete, and modify the settings of a user.
- The Users property sheet consists of several pages. You can create or edit the security settings for a database user by moving to the other pages associated with the Users property sheet. Click Help for more information about a specific page.

To create a database user:

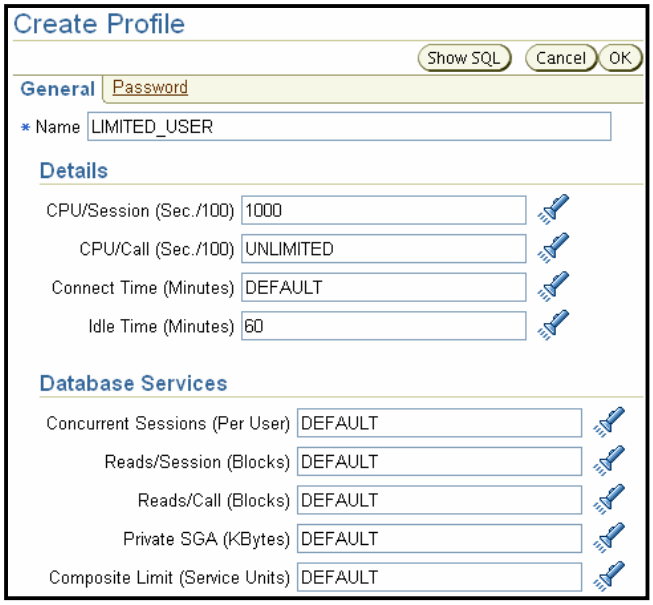
1. Navigate to the Database home page.
2. On the Administration properties page under Security, click Users. The Users Search page appears containing a results table that lists all current users for the database.
3. Click Create to create a new user.

Profiles and Users

Users are assigned only one profile at any given time.

Profiles:

- **Control resource consumption**
- **Manage passwords**



ORACLE

Profiles and Users

Profiles impose a named set of resource limits on database usage and instance resources. Profiles also place limitations on users' passwords (length, expiration time, and so on.) Every user is assigned to a profile and may belong to only one profile at any given time.

The default profile serves as the basis for all other profiles. As you can see in the slide, limitations for a profile can be either implicitly specified (as in CPU/Session), unlimited (as in CPU/Call) or reference whatever setting is in the default profile (as in Connect Time).

Profiles cannot impose resource limitations on users unless the `RESOURCE_LIMIT` initialization parameter is set to `TRUE`. With `RESOURCE_LIMIT` at its default value of `FALSE`, profile limitations are ignored.

Profiles allow the administrator to control the following system resources:

- CPU. CPU resources may be limited on a per-session or per-call basis. A CPU/session limitation of 1000 means that if any individual session using this profile consumes more than 10 seconds of CPU time (CPU time limitations are in 100ths of a second), then that session receives an error and is logged off:

```
ORA-02392: exceeded session limit on CPU usage, you are being
logged off
```

Profiles and Users (continued)

A per-call limitation does the same thing, but instead of limiting the user's overall session, it prevents any single command from consuming too much CPU. If CPU/Call is limited and the user exceeds the limitation, the command aborts and the user gets an error message such as the following:

```
ORA-02393: exceeded call limit on CPU usage
```

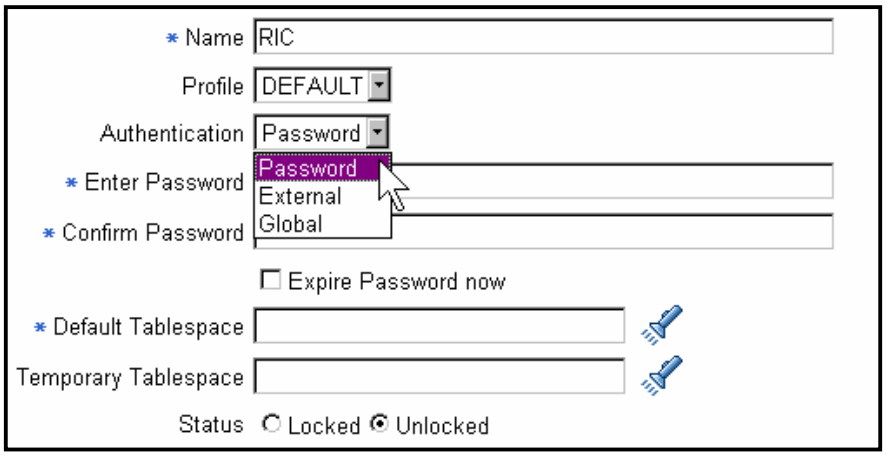
- Network/Memory: Each database session consumes system memory resources and (if the session is from a user not local to the server) network resources.
  - Connect Time: How many minutes a users can be connected before being automatically logged off.
  - Idle Time: How many minutes a user's session can remain idle before being automatically logged off. Idle time is calculated for the server process only. It does not take into account application activity. The `IDLE_TIME` limit is not affected by long-running queries and other operations.
  - Concurrent Sessions: How many concurrent sessions can be created using a database user account.
  - Private SGA: Limits the amount of space consumed within the SGA for sorting, merging bitmaps, and so on. This restriction takes effect only if the session uses a shared server (shared servers are discussed in a later lesson)
- Disk I/O: Limits the amount of data a user can read either at the per-session or per-call level. Reads/Session and Reads/Call place a limitation on the total number of reads from both memory and disk. This can be done to ensure that no I/O intensive statements hoard memory and tie up the disk.

Profiles also allow a composite limit. Composite limits are based on a weighted combination of CPU/session, reads/session, connect time, and private SGA. Composite limits are discussed in more detail in the *Oracle Database Security Guide*.

Using profiles to manage user passwords will be discussed in a later lesson.

Authenticating Users

- **Password**
- **External**
- **Global**



ORACLE

Authenticating Users

Authentication means verifying the identity of someone (a user, device, or other entity) who wants to use data, resources, or applications. Validating that identity establishes a trust relationship for further interactions. Authentication also enables accountability by making it possible to link access and actions to specific identities. After authentication, authorization processes can allow or limit the levels of access and action permitted to that entity.
When you create a user you must decide on the authentication technique to use, which can be modified later.

Password: Also referred to as authentication by the Oracle database, create each user with an associated password that must be supplied when the user attempts to establish a connection. When setting up a password you can expire the password immediately, which forces the user to change the password after first logging in. If you plan on expiring user passwords, make sure the users have the ability to change the password. Some applications do not have this functionality.

Authenticating Users (continued)

External: Also referred to as authentication by the operating system, users can connect to Oracle more conveniently, without specifying a username or password. 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 `OS_AUTHENT_PREFIX` initialization parameter and use this prefix in Oracle usernames. The `OS_AUTHENT_PREFIX` parameter defines a prefix that Oracle adds to the beginning of every user's operating system account name. The default value of this parameter is `OPS$` for backward compatibility with previous versions of Oracle. 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$
```

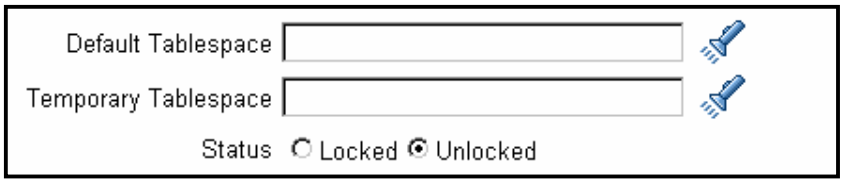
If a user with an operating system account named `tsmith` is to connect to an Oracle database and be authenticated by the operating system, Oracle checks that there is a corresponding database user `OPS$tsmith` and, if so, allows the user to connect. All references to a user authenticated by the operating system must include the prefix, as seen in `OPS$tsmith`.

Note: The text of the `OS_AUTHENT_PREFIX` initialization parameter is case sensitive on some operating systems. See your operating system–specific Oracle documentation for more information about this initialization parameter.

Global: Strong authentication through the Oracle Advanced Security option, global authentication allows users to be identified through the use of biometrics, x509 certificates, token devices, and the Oracle Internet Directory. For more information on advanced authentication methods refer to the *Oracle Enterprise Identity Management* course.

Default and Temporary Tablespaces and Locking

- **Default: Default location of database objects**
- **Temporary: Used for sorting**



ORACLE

Default and Temporary Tablespaces and Locking

A default tablespace is the tablespace where objects are created if a tablespace is not specified in the creation of the object. If you do not choose a default tablespace, then the system-defined default permanent tablespace is used.

The temporary tablespace is the tablespace where any sorting action happens. There are several ways that a sort can take place, for example when creating an index or using an `ORDER BY` clause on a `SELECT` statement. If you do not specify one, then system-defined temporary tablespace is used.

When a user is created, that user can have a status of locked or unlocked. If a user is locked, that means that no one can log in as this user. New users' accounts are unlocked by default.

Note: It is possible to define a default and temporary tablespace at the database level. If you define them at the database level, then any user not assigned a specific tablespace for default and temporary when created, is assigned the database default and temporary tablespaces.

Database Users and Schemas

- **The collection of objects owned by a user is the *schema*.**
- **A user can be associated with only one schema.**
- **Username and schema are often used interchangeably.**

Schema Objects
- Tables
- Triggers
- Indexes
- Views
- Sequences
- Stored program units
- Synonyms
- User-defined data types
- Database links

ORACLE

Database Users and Schemas

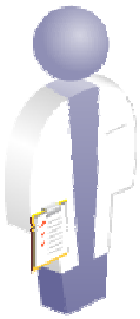
A schema is the collection of database objects owned by a database user. A schema has the same name as the user that owns it. Schema objects include structures such as tables, views, indexes, and Java and PL/SQL stored code. There is no relationship between a tablespace and a schema. Objects in the same schema can be in different tablespaces, and a tablespace can hold objects from different schemas. When a database user is created, a corresponding schema with the same name is created for that user. A user can be associated only with a schema of the same name, and therefore *username* and *schema* are often used interchangeably.

Schema objects that occupy space within the database (such as tables and indexes) are created in the user's default tablespace unless specifically placed somewhere else. Schema owners have full control over their own schema objects and can grant permission to other users to use those objects.

Some user accounts are created specifically to provide a schema and aren't intended to allow users to log in to the database. These accounts should be locked, which means that no one can log in to the account. To lock an account, select Locked in the status section of the Create or Edit User page.

Checklist for Creating Users

- **Select a profile.**
- **Select an authentication technique.**
- **Assign a default tablespace and temporary tablespace.**
- **Grant privileges and roles to the user.**
- **Decide on quotas for each tablespace.**



ORACLE

Checklist for Creating Users

To create a user, the administrator must assign a profile, choose an authentication technique, and assign tablespaces. By default a when a user is created they are not given any privileges, which means they will not be able to do any actions in the database. Database Control automatically grants a few privileges which allows the user to connect to the database and create some objects in the database. However if the user doesn't have quota in any tablespace, the user will still be unable to create objects.

Privileges

There are two types of user privileges:

- **System: Enables users to perform particular actions in the database**
- **Object: Enables users to access and manipulate a specific object**

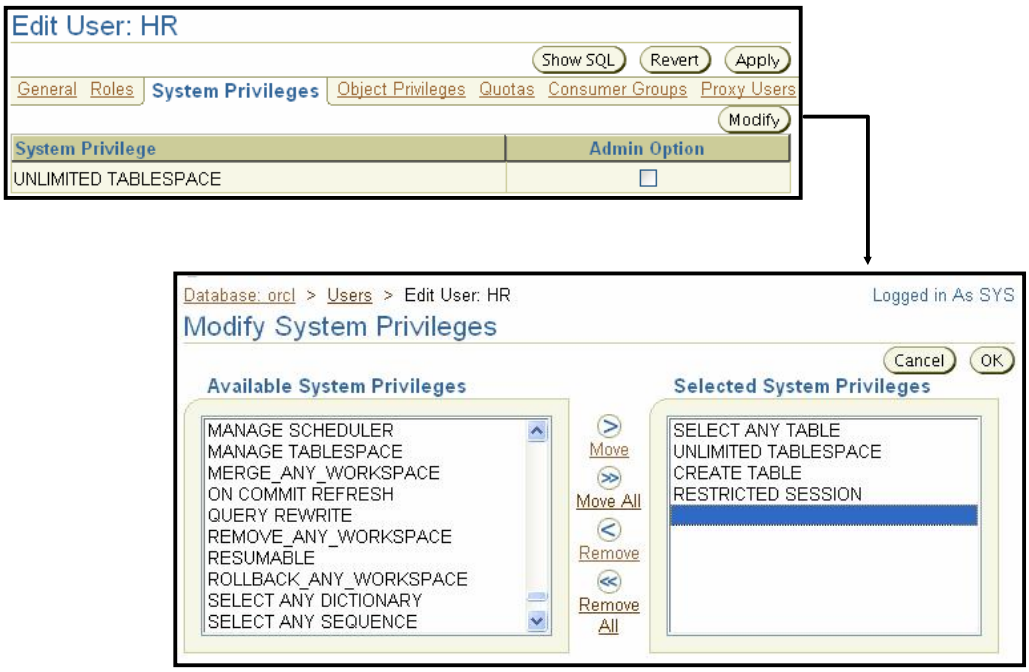
ORACLE

Privileges

A privilege is a right to execute a particular type of SQL statement or to access another user's object. Oracle allows very fine-grained control over what users can or cannot do within the database. Privileges are divided into two categories:

- System privileges. Each system privilege allows a user to perform a particular database operation or class of database operations; for example, the privilege to create tablespaces is a system privilege. System privileges can be granted by the administrator or by someone who explicitly gives permission to administer the privilege. There are over 100 distinct system privileges.
- Object privileges. Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package. Without specific permission, users can access only their own objects. Object privileges can be granted by the owner of an object, by the administrator, or by someone who was explicitly given permission to grant privileges on the object.

System Privileges



ORACLE

System Privileges

To grant system privileges, click the Systems Privileges link, select appropriate privileges from the list of available privileges, and move them to the Selected System Privileges list box by clicking the Move arrow.

Granting a privilege with the ANY clause means the privilege crosses schema lines. For example, the CREATE TABLE privilege allows the user to create a table, but only within their own schema. The SELECT ANY TABLE privilege allows a user to select from tables owned by other users.

Selecting the Admin Option check box allows this user to administer the privilege and grant the system privilege to other users.

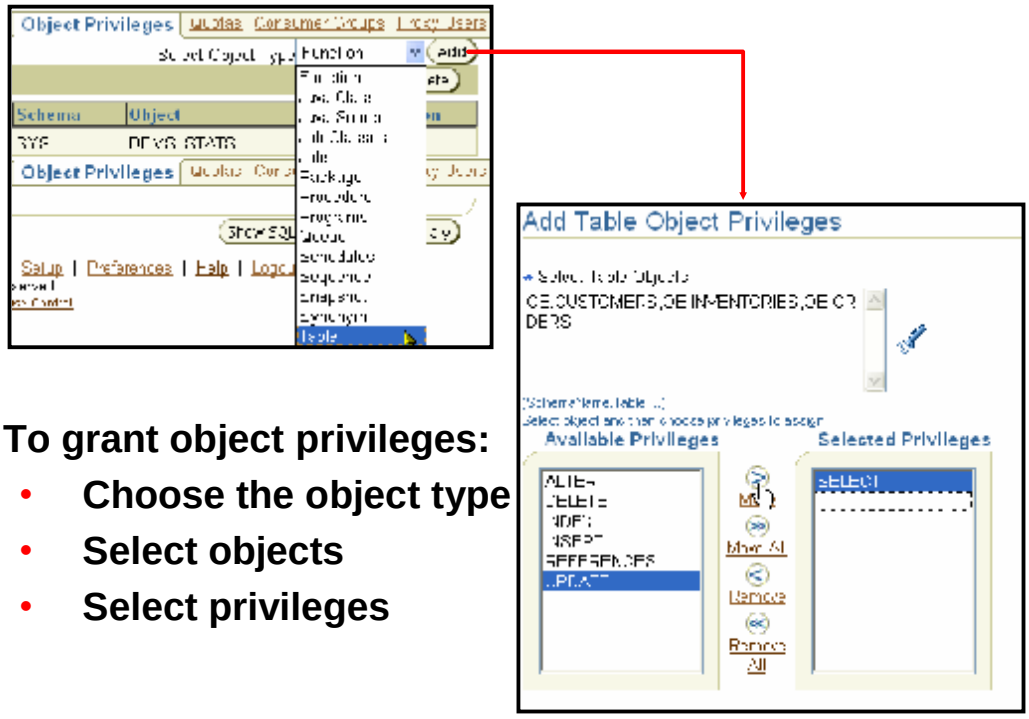
Carefully consider security requirements before granting system permissions. Some system privileges are usually granted only to administrators:

- RESTRICTED SESSION: This privilege allows the user to log in even if the database has been opened in restricted mode.

System Privileges (continued)

- SYSDBA and SYSOPER: These privileges allow the user to shut down, start up, perform recovery, and other administrative tasks in the database.
- DROP ANY *object*: The DROP ANY privilege allows users to delete objects they do not own.
- CREATE, MANAGE, DROP, ALTER TABLESPACE: Nonadministrators should not usually be able to control tablespaces.
- CREATE ANY DIRECTORY: Oracle allows developers to call external code (for example, a C library) from within PL/SQL. As a security measure, the operating system directory where the code resides must be linked to a virtual Oracle directory object. With the CREATE ANY DIRECTORY privilege, a user can potentially call insecure code objects.
- EXEMPT ACCESS POLICY: This privilege allows a user to bypass security functions placed on tables or views.
- GRANT ANY OBJECT PRIVILEGE: This privilege allows users to grant object permissions on objects they do not own.
- ALTER DATABASE and ALTER SYSTEM: Nonadministrators should not usually be allowed to change the database or instance.

Object Privileges



To grant object privileges:

- **Choose the object type**
- **Select objects**
- **Select privileges**

ORACLE

Object Privileges

To grant object privileges, click the Object Privileges link, select the type of object you want to grant privileges on, and click the Add button. Choose the objects you want to grant privileges on by either entering in the *username.object name* or by selecting them from the list.

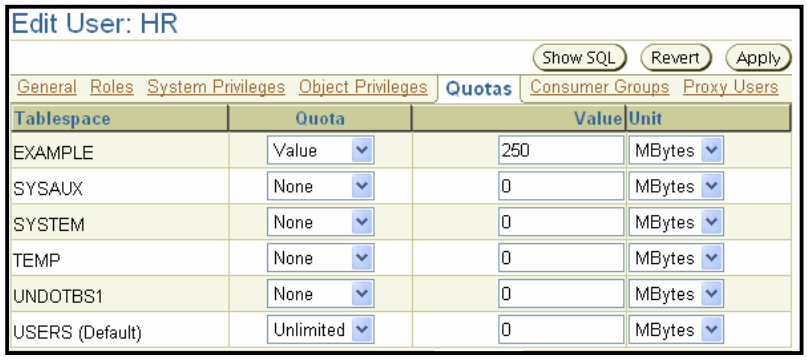
Next, select the appropriate privileges from the Available Privileges list box and click the Move button.

Select the Grant check box from the list of object privileges to allow this user to grant other users the same access.

Assigning Quota to Users

Users who do not have the unlimited tablespace system privilege must be given a quota before they can create objects in a tablespace. Quotas can be:

- **Unlimited**
- **A specific value in megabytes or kilobytes**



ORACLE

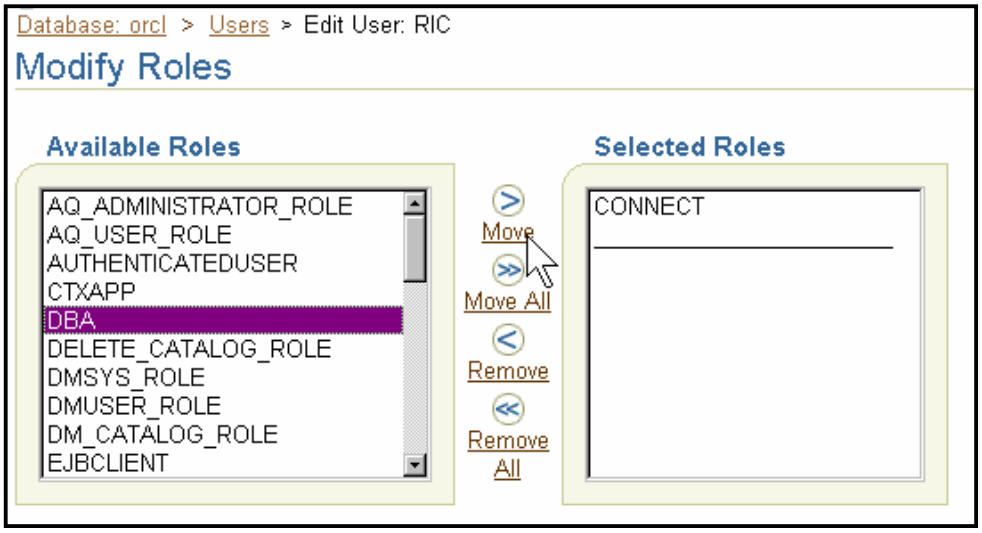
Assigning Quota to Users

Quota is a space allowance in a given tablespace. By default a user has no quota on any of the tablespaces. You have three options for providing a user quota on a tablespace.

- Unlimited: This allows the user to use as much space as is available in the tablespace.
- Value: This is a number of kilobytes or megabytes that the user can use. This doesn't guarantee that the space is set aside for the user. This value can be larger or smaller than the current space available in the tablespace.
- `UNLIMITED TABLESPACE` system privilege: This system privilege overrides all individual tablespace quotas and gives the user unlimited quota on all tablespaces, including `SYSTEM` and `SYSAUX`. This privilege should be granted with caution.

You should not give quota to users on the `SYSTEM` or `SYSAUX` tablespace. Typically, only the users `SYS` and `SYSTEM` should be able to create objects in the `SYSTEM` or `SYSAUX` tablespace. Users do not need quota on their assigned temporary tablespaces or any undo tablespaces.

Assigning Roles to Users



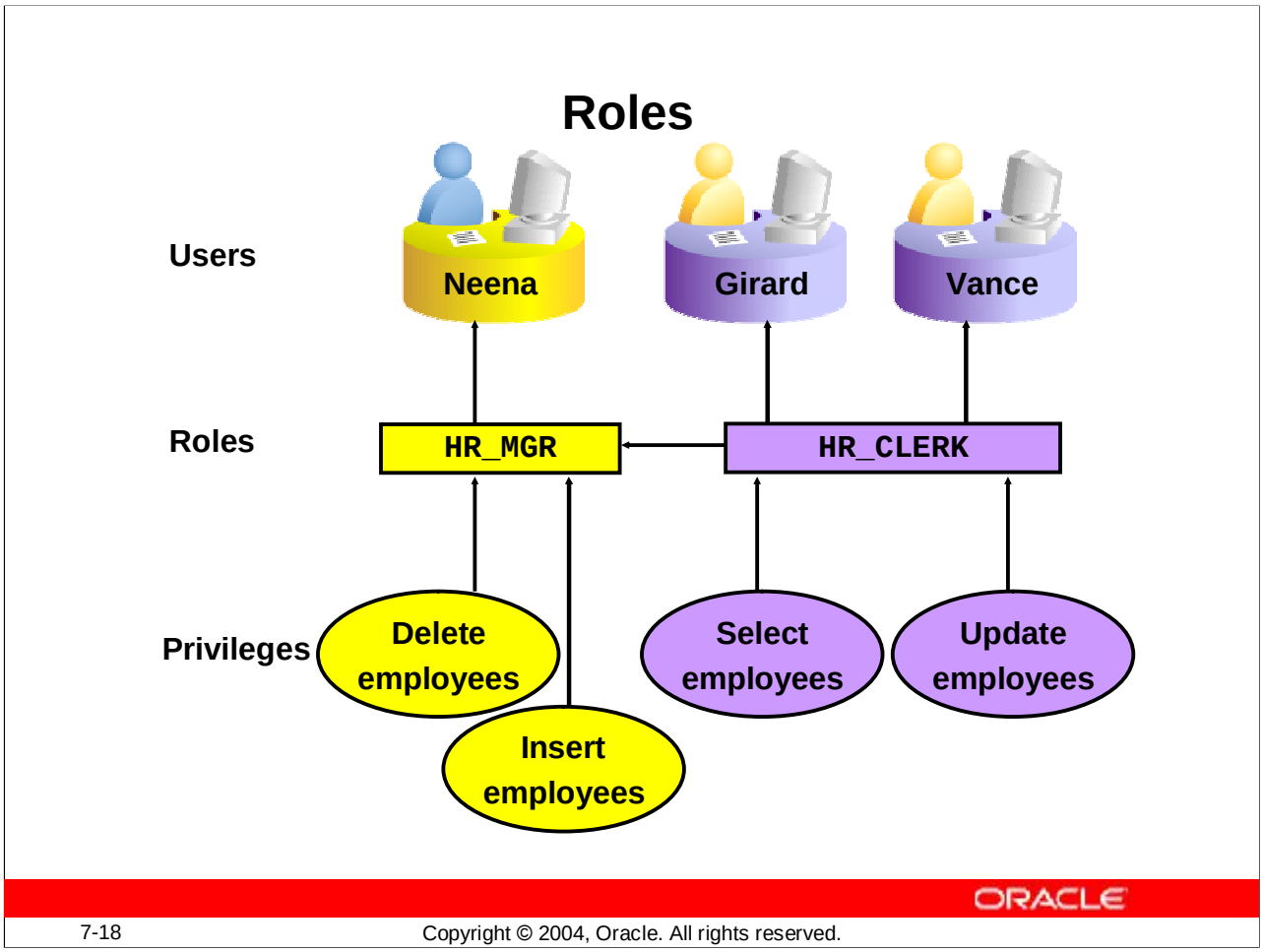
ORACLE

7-17

Assigning Roles to Users

A role is a set of privileges that can be granted to users or to other roles. You can use roles to administer database privileges. You can add privileges to a role and then grant the role to a user. The user can then enable the role and exercise the privileges granted by the role. A role contains all privileges granted to that role and all privileges of other roles granted to it.

By default, Enterprise Manager automatically grants the connect role to new users. This allows users to connect to the database and create database objects in their own schema.



Roles

In most systems it is too time-consuming to grant necessary privileges to each user individually, and there is too great a chance of error. Oracle provides for easy and controlled privilege management through roles. Roles are named groups of related privileges that are granted to users or to other roles. They are designed to ease the administration of privileges in the database, and hence improve security.

Role Characteristics

- Privileges are granted to and revoked from roles as if the role were a user.
- Roles can be granted to and revoked from users or other roles as if they were system privileges.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Roles are not owned by anyone; and they are not in any schema.

In the example above, the `HR_CLERK` role is granted `SELECT` and `UPDATE` privileges on the `employees` table. The `HR_MGR` role is granted the `DELETE` and INSERT privileges on the `employees` table *and* the `HR_CLERK` role. The manager is granted the `HR_MGR` role and can now select, delete, insert, and update the `employees` table.

Benefits of Roles

- Easier privilege management
- Dynamic privilege management
- Selective availability of privileges
- Can be granted through the operating system

ORACLE

Benefits of Roles

Easier Privilege Management

Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role, and then grant that role to each user.

Dynamic Privilege Management

If the privileges associated with a role are modified, all the users who are granted the role acquire the modified privileges automatically and immediately.

Selective Availability of Privileges

Roles can be enabled and disabled to turn privileges on and off temporarily. Enabling a role can also be used to verify that a user has been granted that role.

Granting Through the Operating System

Operating system commands or utilities can be used to assign roles to users in the database.

Predefined Roles

CONNECT	CREATE SESSION, CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE DATABASE LINK, CREATE CLUSTER, ALTER SESSION
RESOURCE	CREATE TABLE, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TRIGGER, CREATE TYPE, CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR
SCHEDULER_ ADMIN	CREATE ANY JOB, CREATE JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER
DBA	Most system privileges, several other roles. Do not grant to nonadministrators.
SELECT_ CATALOG_ ROLE	No system privileges but over 1600 object privileges on the data dictionary

ORACLE

Predefined Roles

There are several roles defined automatically for Oracle databases when you run database creation scripts. CONNECT is granted automatically to any user created with Enterprise Manager. SELECT_CATALOG_ROLE is provided for accessing data dictionary views and packages (this role is deprecated in favor of the SELECT_ANY_DICTIONARY system privilege). The DBA role includes nearly all privileges and should not be granted to nonadministrators.

Functional Roles

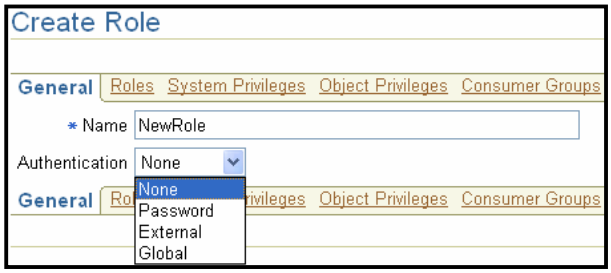
Other roles that authorize you to administer special functions are created when that functionality is installed. For example, XDBADMIN contains the privileges required to administer the XML database if that feature is installed. The AQ_ADMINISTRATOR_ROLE provides privileges to administer advanced queuing. The HS_ADMIN_ROLE includes the privileges needed to administer heterogeneous services. You should not alter the privileges granted to these functional roles without the assistance of Oracle Support because you may inadvertently disable needed functionality.

Secure Roles

Roles may be nondefault.

```
SET ROLE vacationdba;
```

Roles may be protected through authentication.



Roles may also be secured programmatically.

```
CREATE ROLE secure_application_role
IDENTIFIED USING <security_procedure_name>;
```

ORACLE

Secure Roles

Roles are usually enabled by default, which means that if a role is granted to a user, that user can exercise the privileges given to that role. It is possible to:

- Make a role nondefault. When the role is granted to a user, clear the DEFAULT check box. Now the user must explicitly enable the role before the role's privileges can be exercised.
- Have a role require additional authentication. The default authentication for a role is NONE but it is possible to have the role require additional authentication before it can be set.
- Create secure application roles that can be enabled only by executing a PL/SQL procedure successfully. The PL/SQL procedure can check things such as the user's network address, which program the user is running, time of day, or anything else that is needed to properly secure a group of permissions.

Summary

In this lesson you should have learned how to:

- **Create and manage user accounts**
- **Create and manage roles**
- **Grant and revoke privileges**
- **Control resource usage by users**

ORACLE

Practice 7:
Administering Users

This practice covers the following:

- **Creating a profile to limit resource consumption**
- **Creating two roles:**
  a. **`HRCLERK`**
  b. **`HRMANAGER`**
- **Creating three new users, one manager, and two clerks**

ORACLE

Practice 7: Administering Users

Background: You need to create a user account for Jenny Goodman, the new human resources department manager. There are also two new clerks in the human resources department, David Hamby and Rachel Pandya. All three of them should be able to log in to the database, select from the `hr.employees` table, and update records in that table. The manager also needs to be able to insert and delete new employee records. Ensure that if the new users forget to log out at the end of the day, they will automatically be logged off after fifteen minutes.

Tasks:

- Create a profile with resource limitations
- Create roles with appropriate permissions
- Create users and assign them the correct roles

1. Create a profile named `HRPROFILE` that limits idle time to 15 minutes. Leave all other fields set to Default.
2. Create roles with appropriate permissions.
   a) Create a role name `HRCLERK` with `SELECT` and `UPDATE` permissions on the `hr.employees` table.
   b) Create a role name `HRMANAGER` with `INSERT` and `DELETE` permissions on the `hr.employees` table. Grant the `HRCLERK` role to the `HRMANAGER` role.
3. Create three new users.
   a) Create an account for David Hamby, a new HR clerk.
   b) Create an account for Rachel Pandya, another new HR clerk.
   c) Create an account for Jenny Goodman, the new HR manager.
4. Test the new users.
   a) Connect to the database as user `DHAMBY`. Attempt to select from the `hr.employees` table.
   b) Now attempt to delete a record from the `hr.employees` table.
   c) Connect to the database as `JGOODMAN` and attempt to select and then delete from the `hr.employees` table.
   d) Roll back the delete operation because this was only a test.
5. When you created the new users you did not select a default or temporary tablespace. What determines which tablespaces the new users will use?
6. You did not grant the `CREATE SESSION` system privilege to either of the new users, but they can all connect to the database. Why?
7. Create a new user account to own database objects for a new inventory application. The username should be inventory with a password of `verysecure`. Make the user's default tablespace the inventory tablespace. Grant the user the connect and resource roles.
8. Leave one of the new users connected to the database during the next lesson or at the end of the day. Verify that the user is automatically logged out after fifteen minutes.

8

Managing Schema Objects

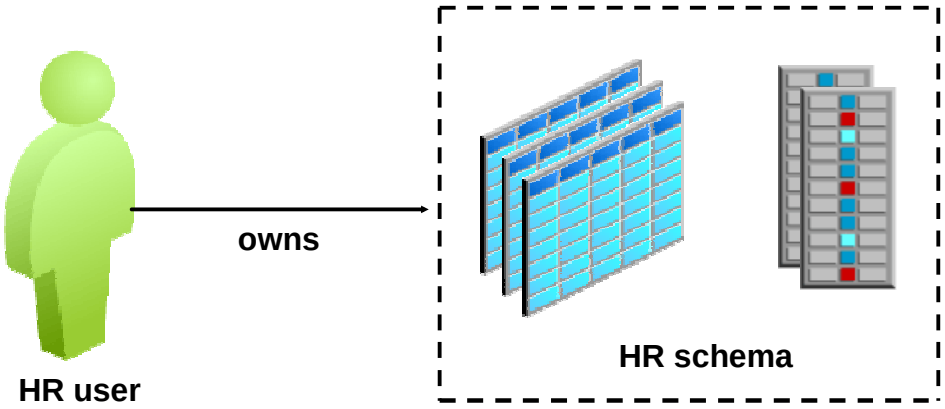
Objectives

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

- **Create and modify tables**
- **Define constraints**
- **View the attributes of a table**
- **View the contents of a table**
- **Create indexes and views**

ORACLE

What Is a Schema?



What Is a Schema?

A schema is a collection of database objects owned by a particular user. A schema has the same name as the user that owns the schema. Schema objects are the logical structures that directly refer to the database's data. Schema objects include structures such as tables, views, and indexes.

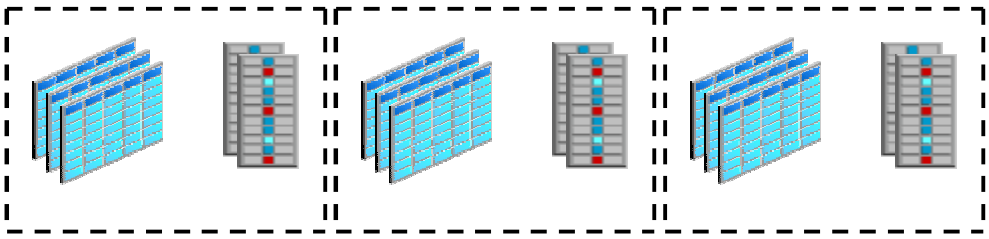
Note: There is no relationship between a tablespace and a schema. Objects in the same schema can be in different tablespaces, and a tablespace can hold objects from different schemas.

You can create and manipulate schema objects using SQL or Enterprise Manager. When you use Enterprise Manager, the underlying SQL is generated for you.

Schemas

Schemas created as part of the database creation process:

- **SYS**
- **SYSTEM**
- **Sample schemas**



Schemas

When you create the database, a number of schemas are created for you. Two of particular importance are:

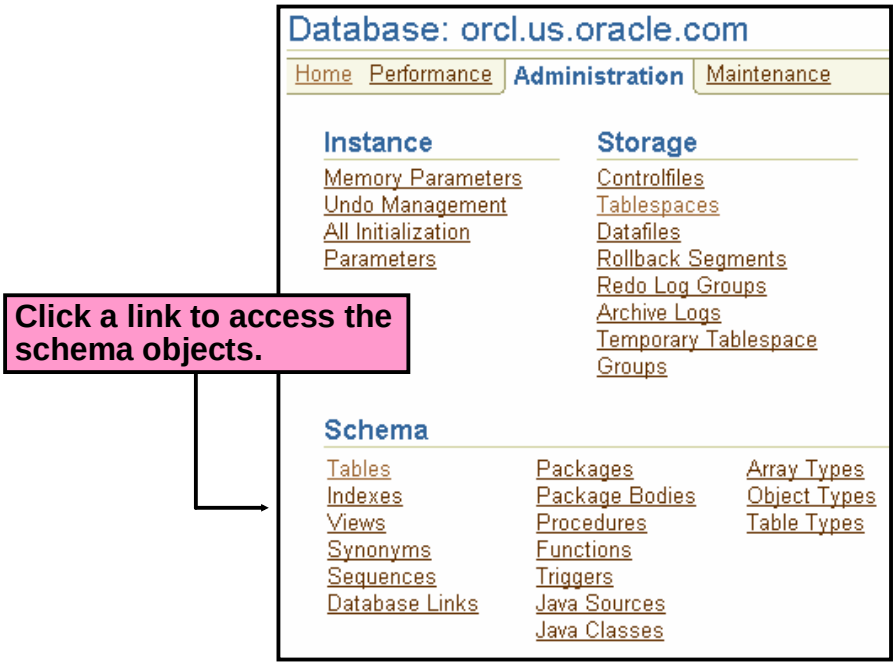
- SYS schema: All of the base tables and views that constitute the database's data dictionary are created in the SYS schema. The data dictionary is a collection of tables that describe the Oracle database. The data dictionary is created in the SYSTEM tablespace when the database is created and is updated by the Oracle database server when a data definition language (DDL) statement is executed. The data dictionary contains information about users, schema objects, and storage structures.
  You can use the data dictionary as a read-only reference for information about the database. When you use Enterprise Manager you access the data dictionary tables through views. Objects in the SYS schema should never be modified by any user or database administrator, and no one should create any tables in the schema of user SYS.
- SYSTEM schema: Contains additional tables and views that store administrative information, and internal tables and views used by various Oracle options and tools. You should not create any additional objects in the SYSTEM schema.

Schemas (continued)

During a complete installation of an Oracle database, the sample schemas are installed automatically with the seed database. The sample schemas serve the purpose of providing a common platform for examples in Oracle documentation and curriculum. They are a set of interlinked schemas aimed at providing a layered approach to complexity and include the following:

- HR: The Human Resources schema is a simple schema for introducing basic topics. An extension to this schema supports Oracle Internet Directory demonstrations.
- OE: The Order Entry schema is for dealing with matters of intermediate complexity. A multitude of data types is available in the OE schema. The OC (Online Catalog) subschema is a collection of object-relational database objects built inside the OE schema.
- PM: The Product Media schema is dedicated to multimedia data types.
- QS: The Queued Shipping schema contains a set of schemas that are used to demonstrate Oracle Advanced Queuing capabilities.
- SH: The Sales History schema is designed to allow demonstrations with larger amounts of data. An extension to this schema provides support for advanced analytic processing.

Accessing Schema Objects



ORACLE

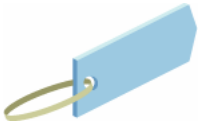
Accessing Schema Objects

You can quickly access many types of schema objects from the Schema region of the Database Administration page.

After clicking one of the links, the Results page is displayed. In the Search region of the page you can enter a schema name and object name to search for a specific object. In addition, you can search for other types of objects from the Search region by selecting the object type from the drop down menu. The drop-down menu includes additional object types that are not shown as links on the Database Administration page.

Naming Database Objects

- **Names must be from 1 to 30 bytes long with these exceptions:**
  - **Names of databases are limited to 8 bytes**
  - **Names of database links can be as long as 128 bytes**
- **Nonquoted names cannot be Oracle reserved words.**
- **Nonquoted names must begin with an alphabetic character from your database character set.**



ORACLE

Naming Database Objects

When you name an object in the database you have the option of enclosing names in double quotation marks ("). If you do this then you can break several of the naming rules mentioned in the slide. However this is not recommended, because if you name an object this way you must always refer to it with the quotes around the name. For example, if you name a table "Local Temp" you must do the following:

```
SQL>  select * from "Local Temp";
TEMP_DATE    LO_TEMP    HI_TEMP
--------- ---------- ----------
01-DEC-03         30         41
```

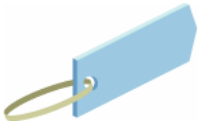
If you mistype the case you will get:

```
SQL> select * from "local temp";
select * from "local temp"
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Nonquoted names are stored in uppercase and are not case sensitive. When a SQL statement is processed, nonquoted names are converted to all uppercase.

Naming Database Objects

- **Nonquoted names can contain only**
  - **Alphanumeric characters from your database character set**
  - **The underscore (_)**
  - **Dollar sign ($)**
  - **Pound sign (#)**
- **No two objects can have the same name within the same namespace**



ORACLE

Naming Database Objects (continued)

Nonquoted identifiers can contain only alphanumeric characters from your database character set and the underscore (_), the dollar sign ($), and the pound sign (#). Database links can also contain periods (.) and the "at" sign (@). You are strongly discouraged from using $ and # in nonquoted identifiers.

Quoted identifiers can contain any characters and punctuations marks as well as spaces. However, neither quoted nor nonquoted identifiers can contain double quotation marks.

Schema Object Namespaces

The following are in the same namespace:

- Tables
- Views
- Sequences
- Private synonyms
- Stand-alone procedures
- Stand-alone stored functions
- Packages
- Materialized views
- User-defined types

The following have their own namespace:

- Indexes
- Constraints
- Clusters
- Database triggers
- Private database links
- Dimensions

ORACLE

Schema Object Namespaces

The Oracle database uses namespaces to resolve schema object references. When you refer to an object in a SQL statement, Oracle considers the context of the SQL statement and locates the object in the appropriate namespace. After locating the object, Oracle performs the operation specified by the statement on the object. If the named object cannot be found in the appropriate namespace, then Oracle returns an error.

Because tables and views are in the same namespace, a table and a view in the same schema cannot have the same name. However, tables and indexes are in different namespaces. Therefore, a table and an index in the same schema can have the same name.
Each schema in the database has its own namespaces for the objects it contains. This means, for example, that two tables in different schemas are in different namespaces and can have the same name.

Specifying Data Types in Tables

Common data types:

- **CHAR(*size*): Fixed-length character data of length size bytes**
- **VARCHAR2(*size*): Variable-length character string having maximum length size bytes**
- **DATE: Valid date range from January 1, 4712 BC to December 31, 9999 AD**
- **NUMBER(*p*,*s*): Number having precision *p* and scale *s***


ORACLE

Specifying Data Types in Tables

When you create a table, you must specify a data type for each of its columns. When you create a procedure or function, you must specify a data type for each of its arguments. These data types define the domain of values that each column can contain or each argument can have.

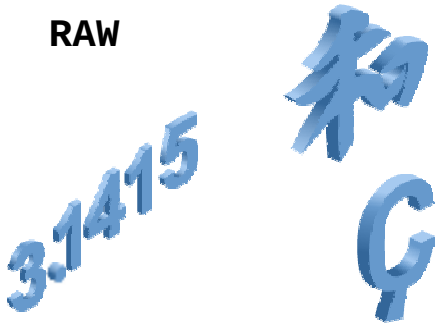
Oracle Database built-in data types include the following:

- CHAR: Fixed-length character data of length size bytes. Maximum size is 2000 bytes. Default and minimum size is 1 byte.
- VARCHAR2: Variable-length character string having maximum length size bytes or characters. Maximum size is 4000 bytes. You must specify size for VARCHAR2.
- DATE: Valid date range from January 1, 4712 BC to December 31, 9999 AD; also stores the time down to the second.
- NUMBER: Number having precision *p* and scale *s*. The precision can range from 1 to 38. The scale can range from -84 to 127.

Refer to the *Oracle Database SQL Reference* for a complete list of built-in data types and user-defined types.

Other Data Types

- FLOAT
- INTEGER
- NCHAR
- NVARCHAR2
- LONG
- LONG RAW
- RAW
- ROWID
- UROWID
- BLOB
- CLOB
- NCLOB
- BFILE
- TIMESTAMP



ORACLE

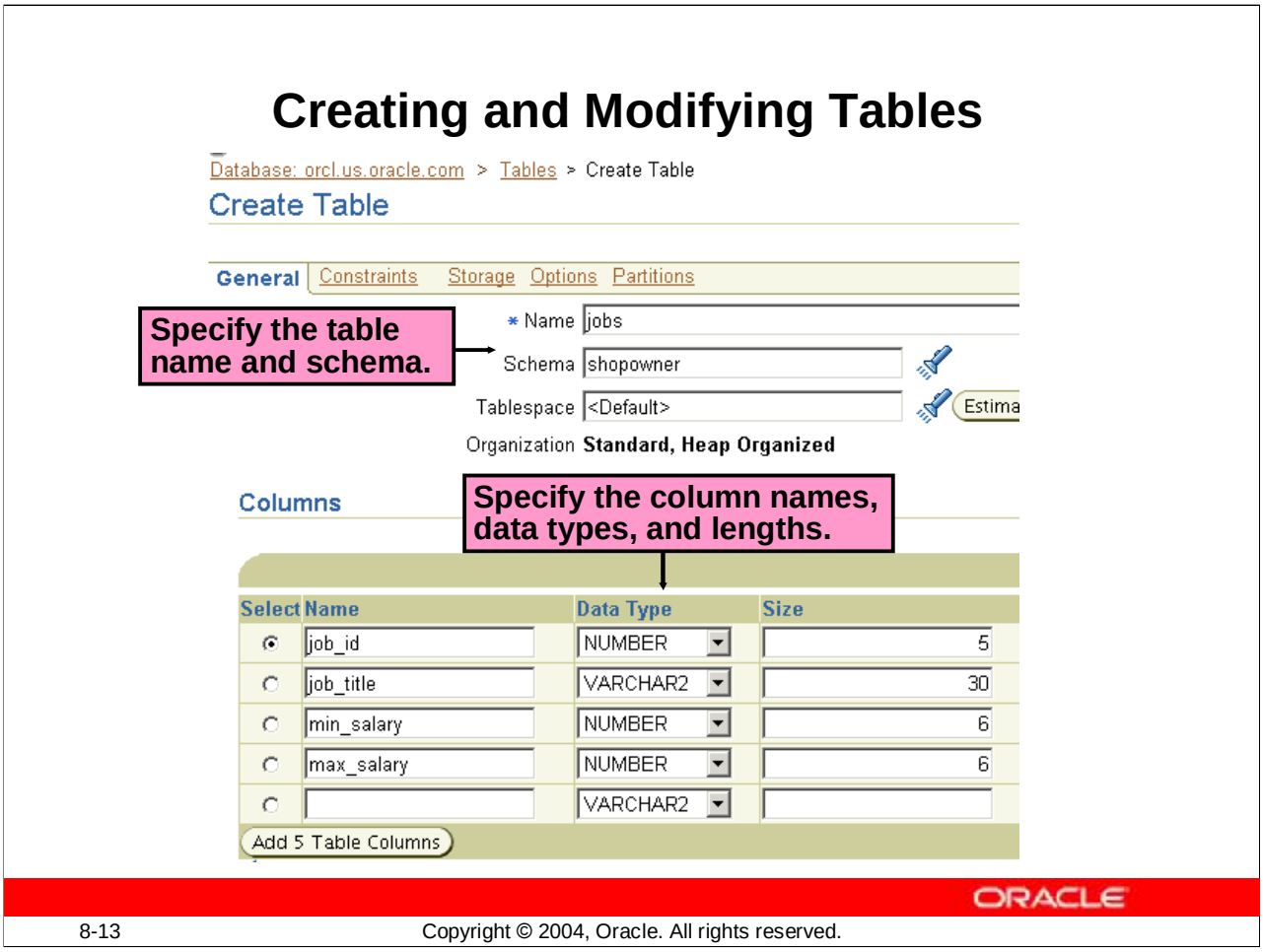
Other Data Types

- `FLOAT(p)`: This is an ANSI data type. The `FLOAT` data type is a floating-point number with a binary precision *p*. The default precision for this data type is 126 binary, or 38 decimal.
- `INTEGER`: This is equivalent to `NUMBER(p,0)`.
- `NCHAR(length)`: The `NCHAR` data type is a Unicode-only data type. When you create a table with an `NCHAR` column, you define the column length in characters. You define the national character set when you create your database. The maximum length of a column is determined by the national character set definition. Width specifications of character data type `NCHAR` refer to the number of characters. The maximum column size allowed is 2000 bytes. If you insert a value that is shorter than the column length, then Oracle blank-pads the value to column length. You cannot insert a `CHAR` value into an `NCHAR` column, nor can you insert an `NCHAR` value into a `CHAR` column.
- `NVARCHAR2(length)`: The `NVARCHAR2` data type is a Unicode-only data type. It is like `NCHAR` expect that its maximum length is 4000 bytes and it is not blank-padded to the length specified.

Other Data Types (continued)

- LONG: Character data of variable length up to 2 gigabytes, or 2^{31} -1 bytes.
- LONG RAW: Raw binary data of variable length up to 2 gigabytes.
- RAW(*size*): Raw binary data of length size bytes. Maximum size is 2000 bytes. You must specify size for a RAW value.
- ROWID: Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn.
- UROWID: Base 64 string representing the logical address of a row of an index-organized table. The optional size is the size of a column of type UROWID. The maximum size and default is 4000 bytes.
- BLOB: A binary large object.
- CLOB: A character large object containing single-byte or multibyte characters. Both fixed-width and variable-width character sets are supported, and both use the CHAR database character set.
- NCLOB: A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, and both use the NCHAR database character set. Stores national character set data.

  Note: The maximum size for all LOB datatypes (BLOB, CLOB and NCLOB) is: (4 gigabytes - 1) * (the value of CHUNK). A CHUNK is an optional attribute which you can set when defining a LOB. The CHUNK specifies the number of bytes to be allocated for LOB manipulation. If size is not a multiple of the database block size, then the database rounds up in bytes to the next multiple. For example, if the database block size is 2048 and CHUNK size is 2050, then the database allocates 4096 bytes (2 blocks). The maximum value is 32768 (32K), which is the largest Oracle Database block size allowed. The default CHUNK size is one Oracle Database block.
- BFILE: Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes.
- TIMESTAMP(*fractional_seconds_precision*): Year, month, and day values of date, as well as hour, minute, and second values of time, where *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND date time field (as in fractions of a second). Accepted values of *fractional_seconds_precision* are 0 to 9. The default is 6.



Creating and Modifying Tables

Tables are the basic unit of data storage in an Oracle database. They hold all user-accessible data. Each table has columns and rows.

Creating a Table

You can create a table using Enterprise Manager as follows:

1. Click Tables in the Schema region of the Administration page. The Tables page appears.
2. If you know the schema name, enter all or part of it in the Schema field in the Search region. If you do not know the schema name, click the flashlight icon next to the Schema field. The Search and Select: Schema window appears. You can page through the schema names and select the one you are looking for.
3. Click Create. The Create Table: Table Organization page appears.
4. Accept the default of Standard, Heap Organized by clicking Continue. The Create Table page appears.
5. Enter the table name in the Name field.
6. Enter the schema name in the Schema field or click the flashlight icon to invoke the search function.

Creating and Modifying Tables (continued)

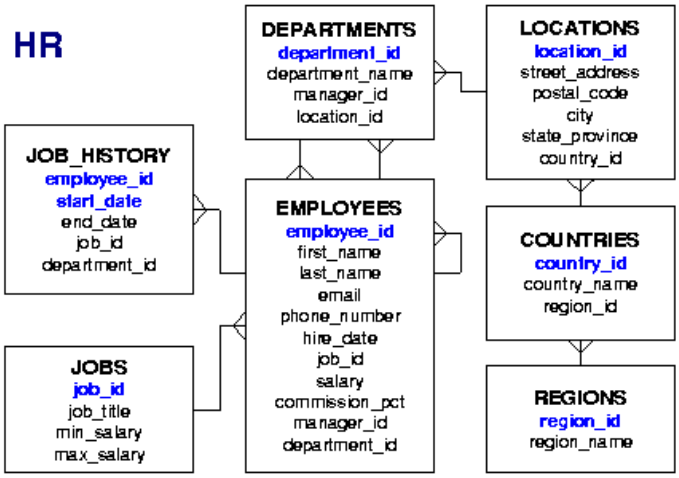
7. Enter the tablespace name in the Tablespace field or click the flashlight icon to invoke the search function.
8. In the Columns section, enter the column name and data types.
9. Click OK. An update message appears indicating the table has been successfully created.

Modifying a Table

You can modify a table using Enterprise Manager as described in the following steps. In this example, an additional column is added to a table.

1. In the Tables page, select the table in the results list and click Edit.
2. In the Edit Table page, click the Add 5 Table Columns button. An editable columns list appears.
3. Enter the new column name, data type, and size. Click Apply.
4. An update message appears indicating that the table has been modified successfully.

Understanding Data Integrity



ORACLE

Understanding Data Integrity

You can use the following integrity constraints to impose restrictions on the input of column values:

- `NOT NULL`: By default, all columns in a table allow null values. Null means the absence of a value. A `NOT NULL` constraint requires that a column of a table contain no null values. For example, you can define a `NOT NULL` constraint to require that a value be input in the `last_name` column for every row of the `employees` table.
- `UNIQUE` Key: A `UNIQUE` key integrity constraint requires that every value in a column or set of columns (key) be unique—that is, no two rows of a table have duplicate values in a specified column or set of columns. For example, a `UNIQUE` key constraint is defined on the `department_name` column of the `departments` table to disallow rows with duplicate department names.
- `PRIMARY KEY`: Each table in the database can have at most one `PRIMARY KEY` constraint. The values in the group of one or more columns subject to this constraint constitute the unique identifier of the row. In effect, each row is named by its primary key values.

Understanding Data Integrity (continued)

The Oracle implementation of the PRIMARY KEY integrity constraint guarantees that both of the following are true:

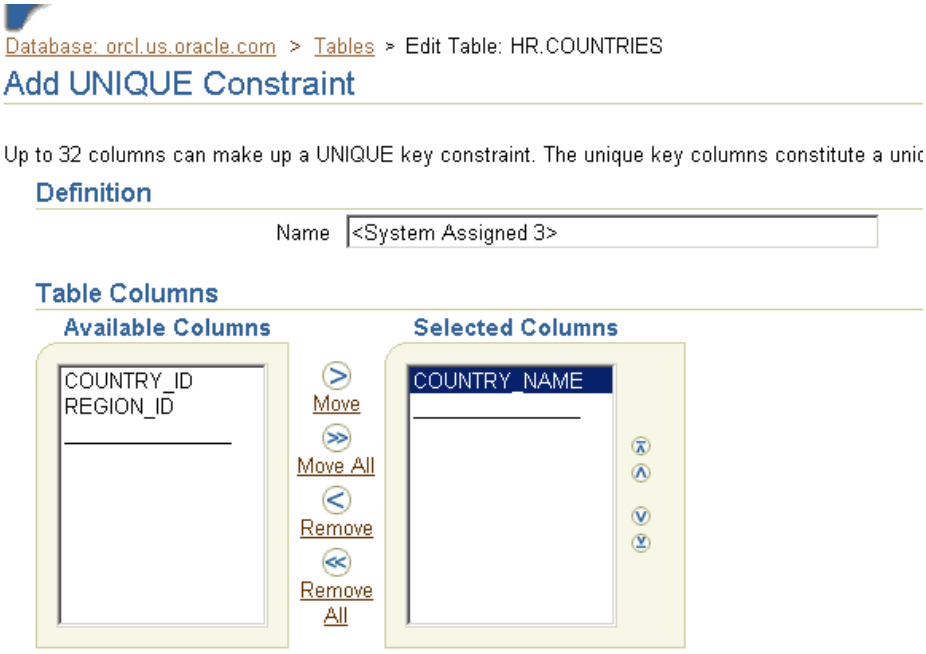
- No two rows of a table have duplicate values in the specified column or set of columns.
- The primary key columns do not allow nulls. That is, a value must exist for the primary key columns in each row.

Oracle enforces all PRIMARY KEY constraints using indexes. The primary key constraint created for the department_id column in the departments table is enforced by the implicit creation of:

- A unique index on that column
- A NOT NULL constraint for that column

- Referential integrity constraints: Different tables in a relational database can be related by common columns, and the rules that govern the relationship of the columns must be maintained. Referential integrity rules guarantee that these relationships are preserved.
  A referential integrity constraint requires that for each row of a table, the value in the foreign key matches a value in a parent key.
  As an example, a foreign key defined is defined on the department_id column of the employees table. It guarantees that every value in this column must match a value in the primary key of the departments table (also the department_id column). Therefore, no erroneous department numbers can exist in the department_id column of the departments table.
  Another type of referential integrity constraint is called a self-referential integrity constraint. This type of foreign key references a parent key in the same table.
- Check constraints: A CHECK integrity constraint on a column or set of columns requires that a specified condition be true or unknown for every row of the table. If a DML statement results in the condition of the CHECK constraint evaluating to false, then the statement is rolled back.

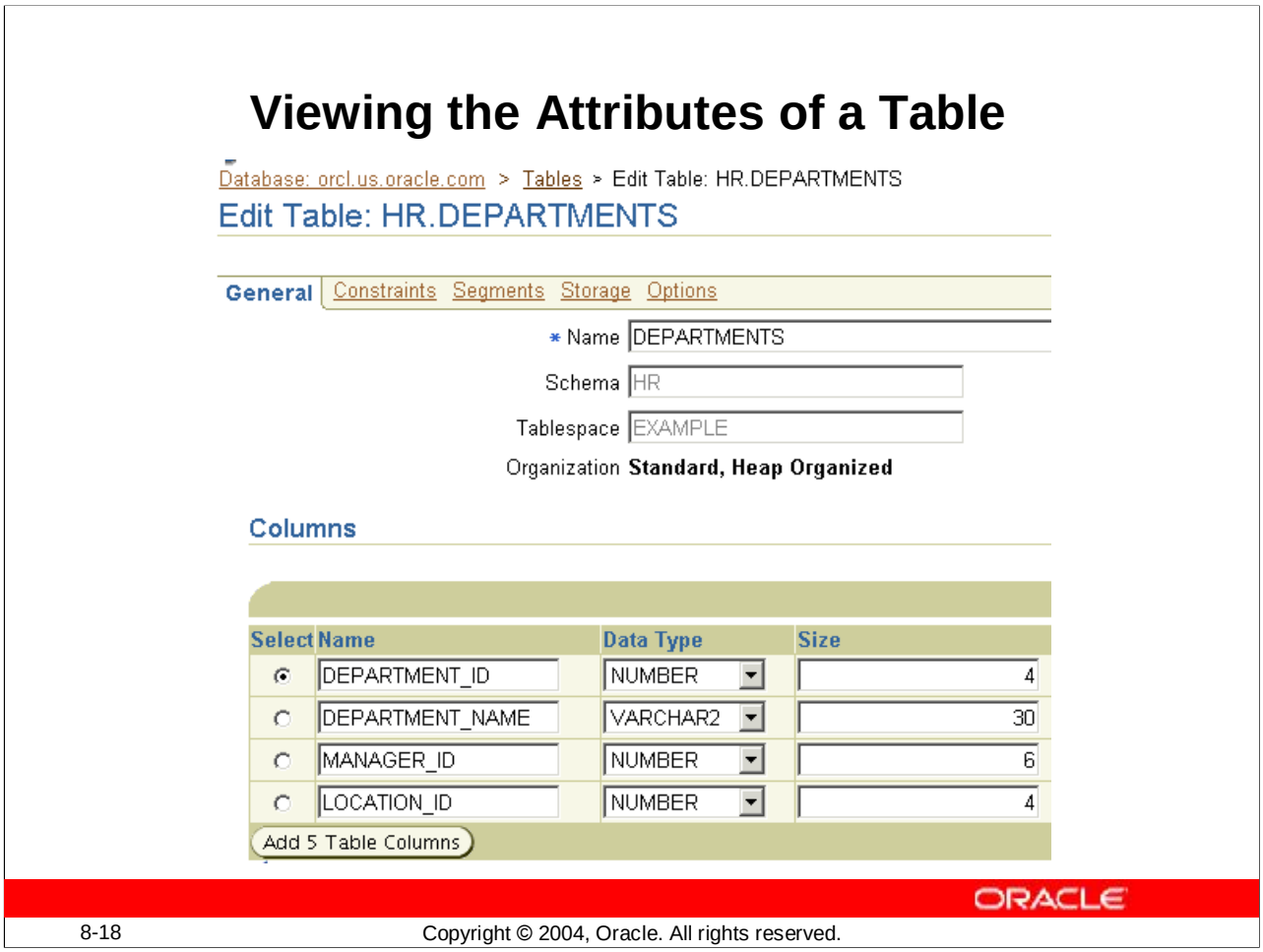
Defining Constraints



Defining Constraints

You can add a constraint to a table as follows:

1. Select the table on the Tables page and click Edit.
2. Click Constraints. The Constraints page is displayed showing all constraints that have been defined on the table.
3. Select the type of constraint you want to add from the drop-down menu and click Add.
4. Enter the appropriate information for the type of constraint you are defining. Click OK.

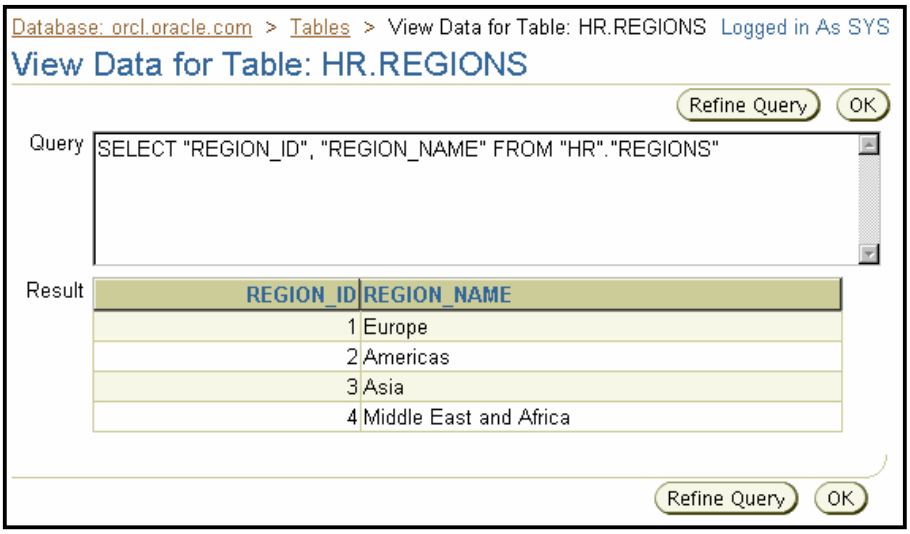


Viewing the Attributes of a Table

You can use Enterprise Manager to view the attributes of a table as follows:

1. Click the Tables link in the Schema section of the Database Administration page.
2. Select a table from the Results list and click the View button to see the attributes of the table.

Viewing the Contents of a Table



ORACLE

Viewing the Contents of a Table

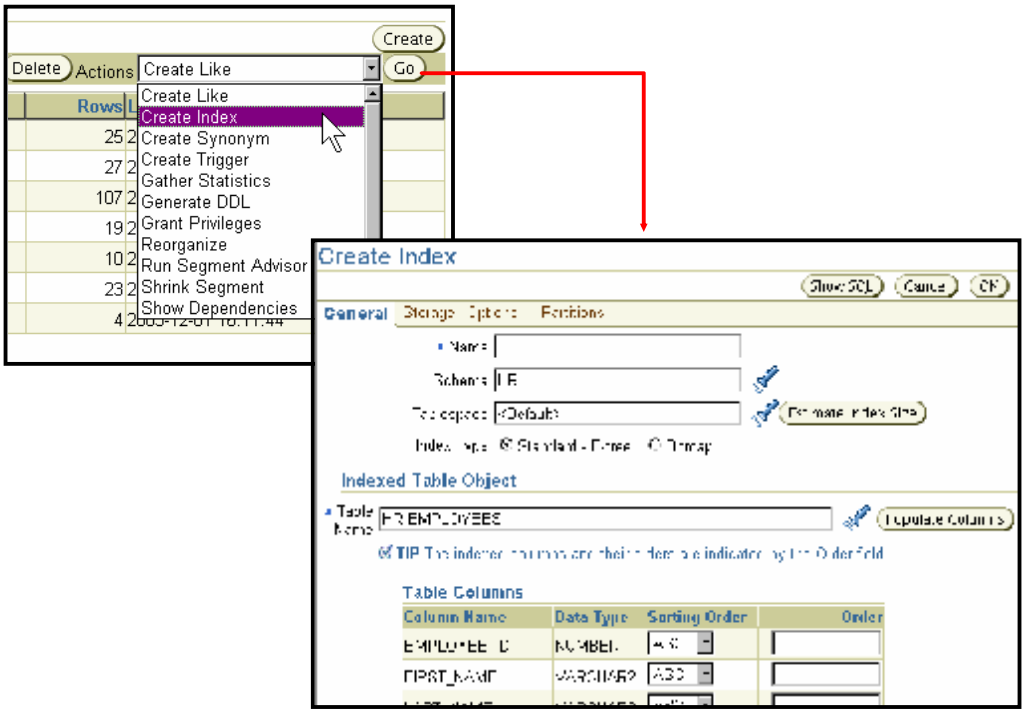
You can easily view the rows in a table using Enterprise Manager as follows:

1. Select the table on the Tables page.
2. Select View Data from the Actions menu and click Go.

The View Data For Table page appears. The row data for the table is shown in the Results section. The Query box displays the SQL query that was executed to produce the results. On this page you can click any column name and sort the data in the column in either ascending or descending order. If you want to change the query, click the Refine Query button. On the Refine Query for Table page, you can select the columns that you want to display and specify a `WHERE` clause for the SQL statement to limit the results.

Refer to the *Oracle Database SQL Reference* for more information on `WHERE` clauses in SQL statements.

Actions with Tables



ORACLE

Actions with Tables

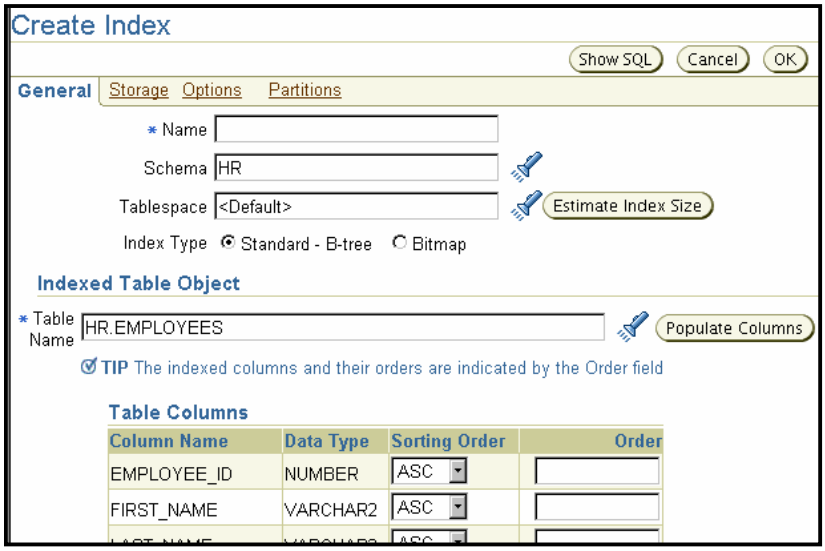
You can select a table and then perform actions on that table.

- Create Like: With this you can create a table that has the same structure as the selected table. You must change the table name and the constraint names. You can also add or delete columns and make other changes to the table structure. When created, the table will be empty.
- Create Index: Use this option to create indexes on a table. Indexes should be created only on columns that are used in `WHERE` conditions of select and DML statements.
- Create Synonym: A synonym is a name used in place of the full table name. A synonym can be private or public.
- Create Trigger: A table-level trigger is a PL/SQL block that is executed because of an event happening on the table. For example you may want to keep a copy of data in another table as it gets deleted from the `employees` table. You would then create a "before delete" trigger that would insert the data about to be deleted from the `employees` table into another table. For more information about triggers refer to the *Application Developer's Guide – Fundamentals* and the *PL/SQL Users Guide and Reference*.

Actions with Tables (continued)

- Gather Statistics: The Gather Statistics Wizard guides you through the process of generating and modifying optimizer statistics. The statistics are stored in the data dictionary and used by the cost-based optimizer. Up-to-date statistics can greatly improve the performance of SQL queries against your objects. Oracle suggests that you use automated tasks to generate statistics regularly within maintenance windows.
- Generate DDL: This generates the DDL command to create the table selected. This can then be copy pasted into a text file for use as a script or for documentation purposes.
- Grant Privileges: By default when a table is created only the owner can do anything with it. The owner must grant out privileges to other users to be able to perform DML and some DDL on the table.
- Reorganize: You can use the Reorg Wizard to rebuild fragmented indexes or tables, move objects to a different tablespace, or optimize the storage attributes of specified objects.
- Run Segment Advisor: The segment advisor determines whether objects have unused space that can be released, taking estimated future space requirements into consideration. The estimated future space calculation is based on historical trends.
- Shrink Segment: The shrink operation compacts fragmented space and, optionally, frees the space.
- Show Dependencies: This show objects that this table depends on or objects that depend on this table.
- View Data: Runs a select statement on the table, you refine the select statement. You cannot change the data.
- Flashback Table: The Flashback Tables facility allows you to recover a table to a previous point in time. It provides a method of recovering a table that has been accidentally modified or deleted by a user or application. You can restore table data along with all its associated attributes, such as indexes, triggers, and so on. This is done while the database is online by rolling back only the changes to the given tables. You can revert a table and its contents to a certain time or user-specified system change number (SCN). Use flashback table with flashback query and row versions to find a time to which the table should be restored. For more information about using the Flashback Tables feature, see the *Oracle Database Administrators Guide*.
- Flashback by Row Versions: Flashback Versions Query allows you to query metadata and historical data within a time interval. You can perform queries on the table as of a certain wall clock time. Select the filter conditions that allow you to retrieve the different versions of rows in a table that existed in a specific time interval.

Creating Indexes



ORACLE

Creating Indexes

Indexes are optional structures associated with tables. They can be created to increase the performance of data retrieval. An Oracle index provides a direct access path to table data.

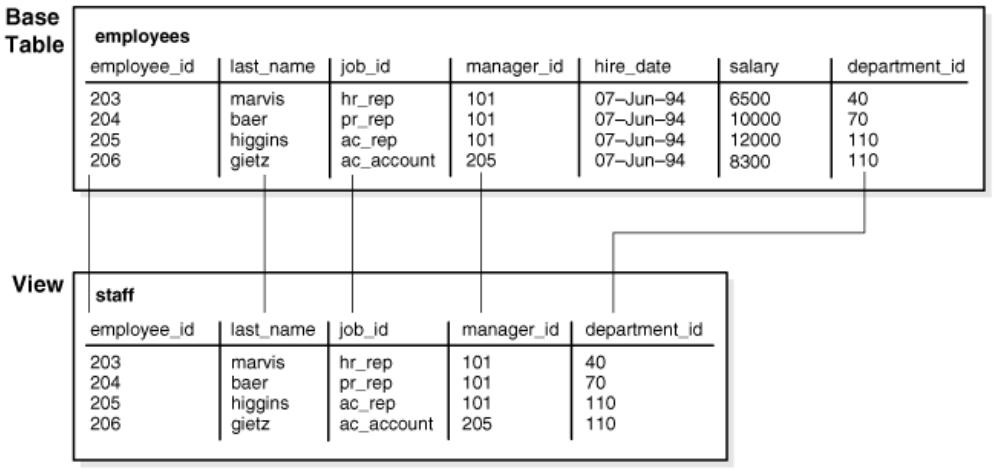
Indexes can be created on one or more columns of a table. After an index is created, it is automatically maintained and used by the Oracle server. Updates to a table's data, such as adding new rows, updating rows, or deleting rows, are automatically propagated to all relevant indexes with complete transparency to the user making the change.

You can click the Indexes link under the Schema heading of the Administration page to view the Indexes page. You can view index attributes, or use the Actions menu to View Dependencies for an index.

Indexes can be created explicitly, or implicitly through constraints that are placed on a table.

What Is a View?

- **Tailored representation of data in a table or view**
- **Views do not contain data**

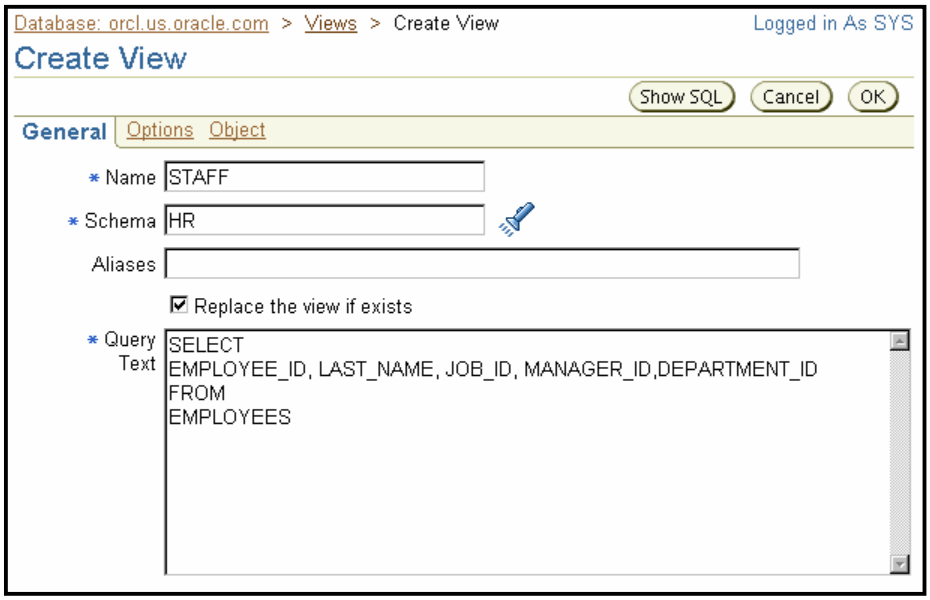


ORACLE

What Is a View?

Views are customized representations of data in one or more tables or other views. They can be thought of as stored queries. Views do not actually contain data, but instead they derive their data from the tables upon which they are based. These tables are referred to as the base tables of the view.

Creating Views



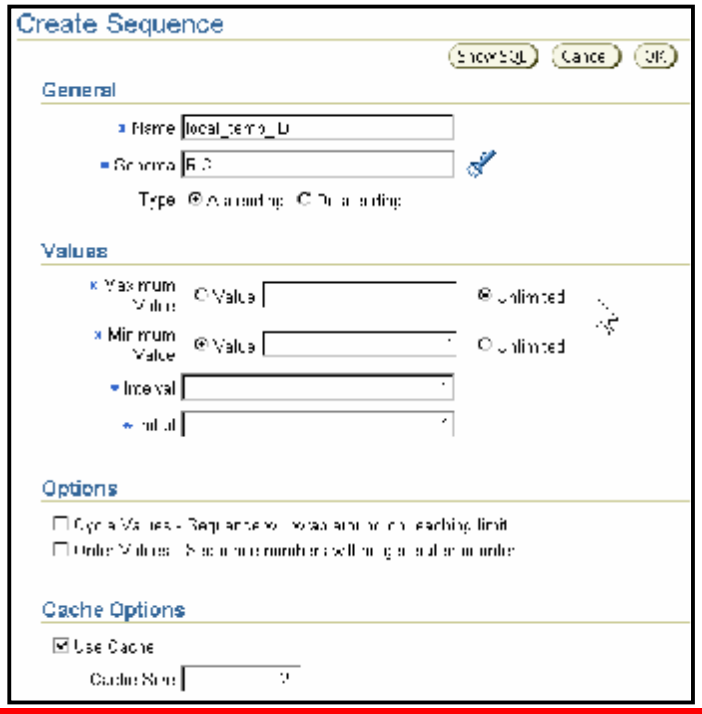
ORACLE

Creating Views

Like tables, views can be queried, updated, inserted into, and deleted from, with some restrictions. All operations performed on a view actually affect the base tables of the view. Views provide an additional level of security by restricting access to a predetermined set of rows and columns of a table. They also hide data complexity and store complex queries.

You can click the Views link under the Schema heading on the Administration page to see the views defined in the database.

What Is a Sequence?



What Is a Sequence?

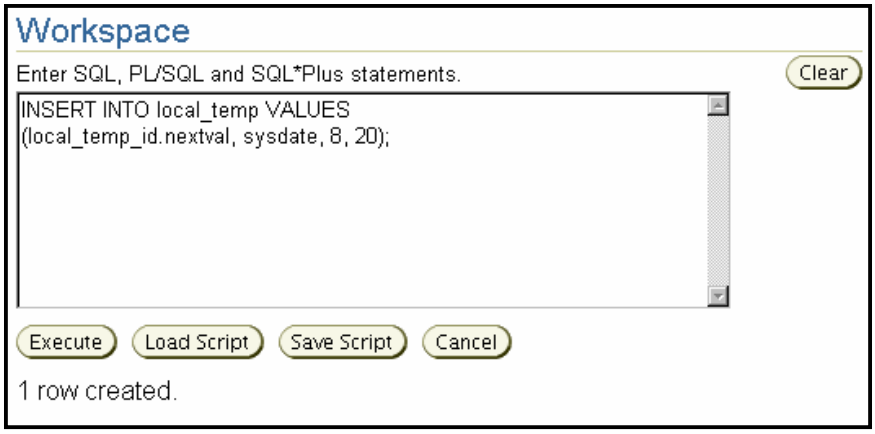
A sequence is a database object from which multiple users can generate unique integers. You generally use sequences to generate primary key values.

- Name: Use the naming rules covered earlier to name a sequence.
- Schema: This is the owner of the sequence.
- Type: A sequence can either be ascending or descending
- Maximum Value: Specify the maximum value the sequence can generate. This integer value can have 28 or fewer digits. It must be greater than Minimum Value and Initial. Using Unlimited indicate a maximum value of 10^{27} for an ascending sequence or -1 for a descending sequence. The default is Unlimited.
- Minimum Value: Specify the minimum value of the sequence. This integer value can have 28 or fewer digits. It must be less then or equal to Initial and less then Maximum Value. Using Unlimited indicates a minimum value of 1 for an ascending sequence or -10^{26} for a descending sequence. The default is Unlimited.

What Is a Sequence? (continued)

- Interval: Specify the interval between sequence numbers. This integer value can be any positive or negative integer, but it cannot be zero. This value can have 28 or fewer digits. The default is one.
- Initial: Specify the first sequence number to be generated. Use this clause to start an ascending sequence at a value greater than its minimum or to start a descending sequence at a value less than its maximum.
- Cycle Values: After an ascending sequence reaches its maximum value, it generates its minimum value. After a descending sequence reaches its minimum, it generates its maximum value. If you do not choose this option, an error will be returned if you attempt to retrieve a value after the sequence has been exhausted.
- Order Values: This guarantees that sequence numbers are generated in order of request. This clause is useful if you are using the sequence numbers as timestamps. Guaranteeing order is usually not important for sequences used to generate primary keys. This option is necessary only to guarantee ordered generation if you are using Oracle with Real Application Clusters.
- Cache Options: Specify how many values of the sequence Oracle preallocates and keeps in memory for faster access. This integer value can have 28 or fewer digits. The minimum value for this parameter is 2. For sequences that cycle, this value must be less than the number of values in the cycle. You cannot cache more values than will fit in a given cycle of sequence numbers.

Using a Sequence



ORACLE

Using a Sequence

You refer to sequence values in SQL statements with the following pseudocolumns:

- CURRVAL: Returns the current value of a sequence
- NEXTVAL: Increments the sequence and returns the next value

You must qualify CURRVAL and NEXTVAL with the name of the sequence:

sequence.CURRVAL
sequence.NEXTVAL

The first reference to NEXTVAL returns the initial value of the sequence. Subsequent references to NEXTVAL increment the sequence value by the defined increment and return the new value. Any reference to CURRVAL always returns the current value of the sequence, which is the value returned by the last reference to NEXTVAL.

Summary

In this lesson, you should have learned how to:

- **Create and modify tables**
- **Define constraints**
- **View the attributes of a table**
- **View the contents of a table**
- **Create indexes and views**

Practice 8:
Working with Tables

This practice covers the following:

- **Creating tables and indexes**
- **Modifying tables**
- **Dropping a table**
- **Creating a view**

ORACLE

Practice 8: Managing Schema Objects

Background: You need to create a schema objects for the new inventory application.

Tasks:

- Create tables to store data
- Ensure referential integrity through the use of constraints
- Create indexes to improve access to data
- Modify existing tables
- Create views that simplify user access to data

1. In the INVENTORY tablespace, create the PRODUCT_MASTER table in the INVENTORY schema. The specifications of the table are:
   `PRODUCT_ID number(5)`. This is the primary key field.
   `PRODUCT_NAME varchar2(50)` with a Not Null constraint.
   `CODE varchar2(20)` with a Not Null constraint.
   `REORDER_THRESHOLD number(5)` with a check constraint ensuring that the number is always greater than zero.
   `COST number(5,2)`
   `PRICE number(5,2)`
2. In the INVENTORY tablespace create the PRODUCT_ON_HAND table in the INVENTORY schema. The specifications of the table are:
   `ON_HAND_ID number(5)`. This is the primary key field.
   `PRODUCT_ID number(5)`. This field should have a foreign key constraint linking it to the product_id field in the product_master table.
   `QUANTITY number(5)`
   `WAREHOUSE_CITY varchar2(30)`
3. In the INVENTORY tablespace create the OBSOLETE_PRODUCTS table in the INVENTORY schema. The specifications of the table are:
   `PRODUCT_ID number(5)`. This is the primary key field.
   `PRODUCT_NAME varchar2(50)` with a Not Null constraint.
   `CODE varchar2(20)` with a Not Null constraint.
   `COST number(5,2)`
   `PRICE number(5,2)`
4. In the INVENTORY tablespace, create an index on the PRODUCT_NAME column of the OBSOLETE_PRODUCTS table in the INVENTORY schema.
5. In the INVENTORY tablespace, create an index on the PRODUCT_NAME and CODE columns of the PRODUCT_MASTER table in the INVENTORY schema.
6. In the INVENTORY tablespace, create an index the PRODUCT_ID and QUANTITY column of the PRODUCT_ON_HAND table in the INVENTORY schema.
7. You receive an update for the inventory application that requires you to add two columns to the PRODUCT_MASTER table. Add a column named PRIMARY_SOURCE of datatype varchar2 with size 50. Add another column named SECONDARY_SOURCE of datatype varchar2 with size 50.

Practice 8: Managing Schema Objects (continued)

8. The update for the inventory application also requires you to add a column to the PRODUCT_ON_HAND table. Add a column named LAST_UPDATE of datatype date.
9. The update for the inventory application also requires you to add a column to the OBSOLETE_PRODUCTS table. Add a column named OBSOLETED of datatype date.
10. You receive another update for the inventory application. This update instructs you to drop the OBSOLETE_PRODUCTS table and add a column OBSOLETED to the PRODUCT_MASTER table with datatype date.
11. The second update to the inventory application also instructs you to create a view named WAREHOUSE_VW in the INVENTORY schema that the shows (in order):
    - The name of the product
    - The amount of the product on hand
    - The warehouse city name.

Managing Data

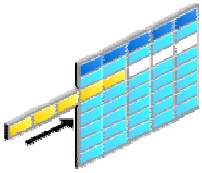
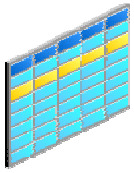
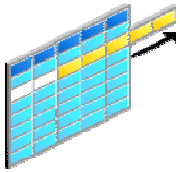
Objectives

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

- **Manipulate data through SQL**
- **Use Data Pump to export data**
- **Use Data Pump to import data**
- **Load data with SQL Loader**

Manipulating Data Through SQL

```
SQL> INSERT INTO employees VALUES
  2  (9999,'Bob','Builder','bob@abc.net',NULL,SYSDATE,
  3  'IT_PROG',NULL,NULL,100,90);

1 row created.

SQL> UPDATE employees SET SALARY=6000
  2  WHERE EMPLOYEE_ID = 9999;

1 row updated.

SQL> DELETE from employees
  2  WHERE EMPLOYEE_ID = 9999;

1 row deleted.
```

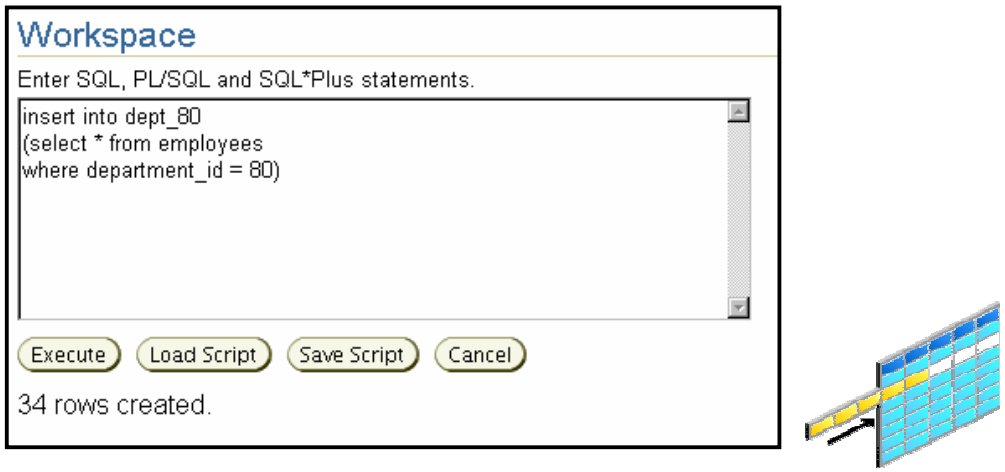
ORACLE

Manipulating Data Through SQL

The basic DML statements are the way data is manipulated in the database. Although these statements were briefly mentioned in the lesson titled “Database Interfaces,” in this lesson they will be covered in more detail.

The INSERT Command

- **Create one row at a time.**
- **Insert many rows from another table.**



The INSERT Command

The basic INSERT statement creates one row at a time. Using what is called a subselect, you can cause the insert to copy rows from one table to another. This method is also referred to as an INSERT select statement. The example on the slide is the following INSERT command:

```
insert into dept_80 (select * from employees
where department_id = 80);
```

In this case the dept_80 table has the exact same structure as the employees table. If this is not the case you can name the columns in each table. The column values will match in the order as named in the INSERT and SELECT statements. All that is required is that the data types match. For example:

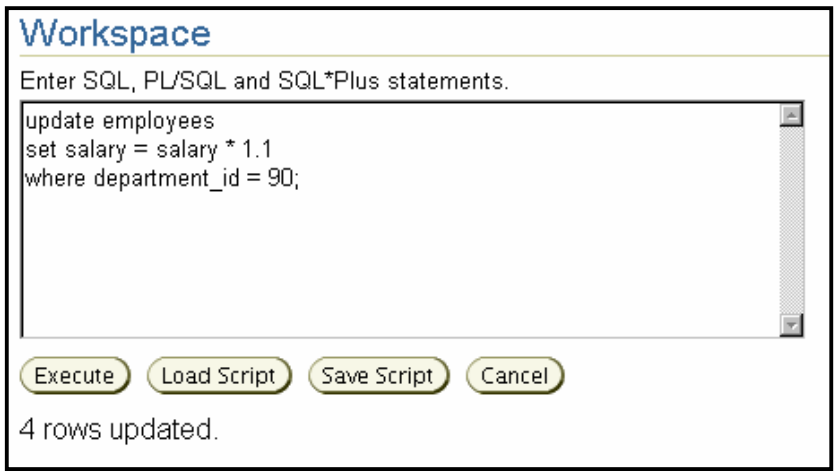
```
insert into just_names (first, last)
(select first_name, last_name from employees);
```

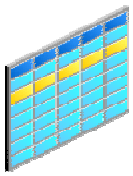
Here the just_names table has only two columns that have the same data type as the first_name and last_name columns in the employees table.

Using the insert select method is a way to bulk load data from one or more tables into another table.

The UPDATE Command

Use to change zero or more rows of a table.





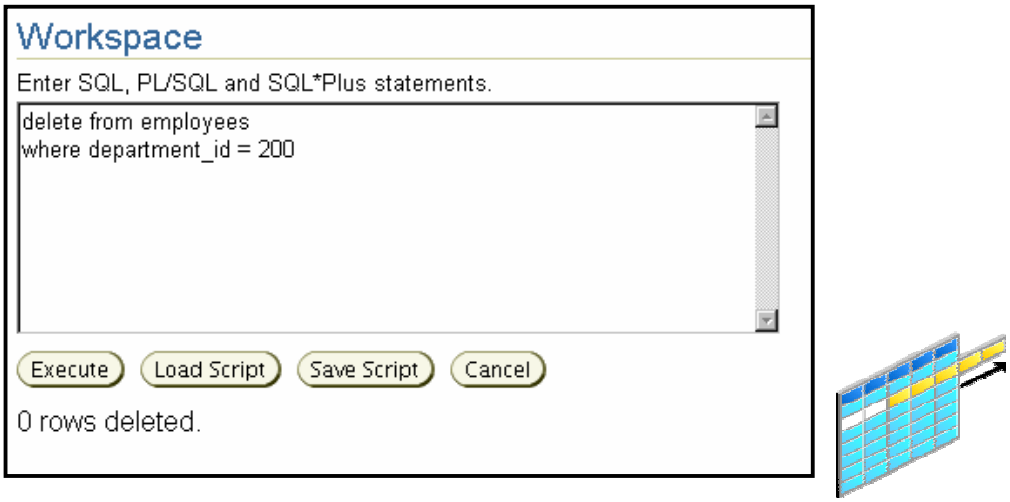
ORACLE

The UPDATE Command

The UPDATE command is used to modify existing rows in a table. The number or rows modified by the update will depend on the WHERE condition. If the WHERE clause is omitted, then all rows will be changed. If no rows satisfy the WHERE condition, then no rows will be modified.

The DELETE Command

Use to remove zero or more rows from a table.



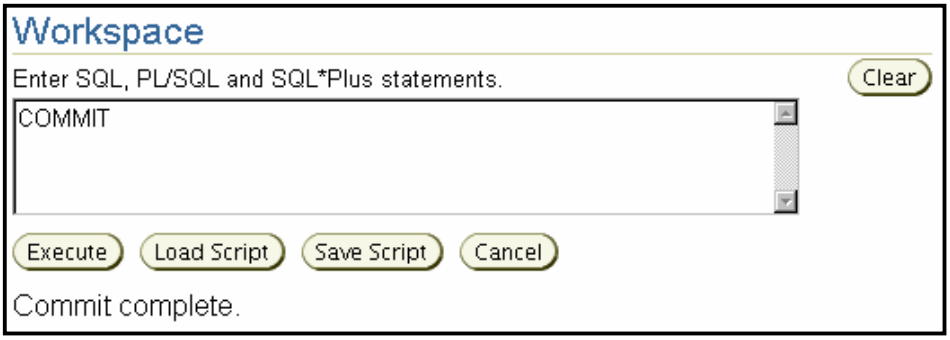
The DELETE Command

The DELETE command is used to remove existing rows from a table. The number or rows modified by the delete depends on the WHERE condition. If the WHERE clause is omitted, then all rows will be removed. If no rows satisfy the WHERE condition then no rows will be removed. Note in the example, when no rows are deleted it is not an error; the message returned only states that zero rows have been removed from the table.

The COMMIT and ROLLBACK Commands

Used to finish a transaction.

- **Commit: Makes the change permanent**
- **Rollback: Undoes the change**



ORACLE

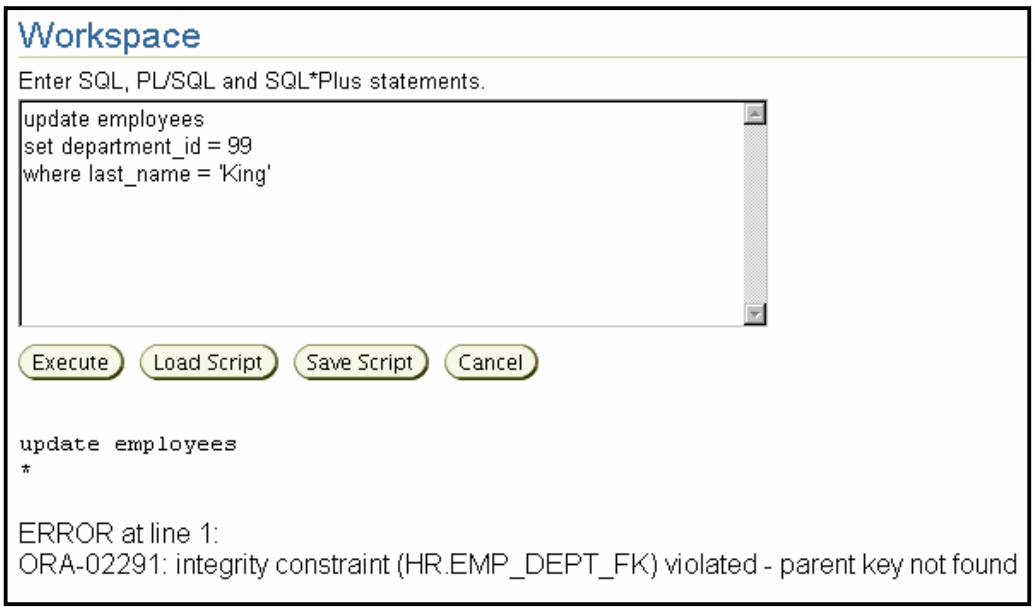
The COMMIT and ROLLBACK Commands

By default each DML command entered is not committed. Several tools (including *i*SQL*Plus) have options that can be set to commit on each command or a group of commands.

Before a `COMMIT` or `ROLLBACK` is issued, the changes are in a pending state. Only the user who made the change is allowed to see the changed data. Other users can select the same data, but will see the data as it was before any change is made. Other users cannot issue DML on the same data that another user has changed.

By default a user trying to make a change on the same row as another user will wait until the first user either commits or rolls back the change. This is controlled automatically by Oracle's locking mechanism. Because the locking mechanism is built into the row itself, there is no way the database will run out of locks.

Integrity Constraints and DML



ORACLE

Integrity Constraints and DML

All integrity constraints are enforced whenever DML statements are executed.

`FOREIGN KEY` columns
- `INSERT` and `UPDATE`: The value must exist in the parent table.
- `DELETE`: A row in the parent table that has at least one row still referring to it cannot be deleted.

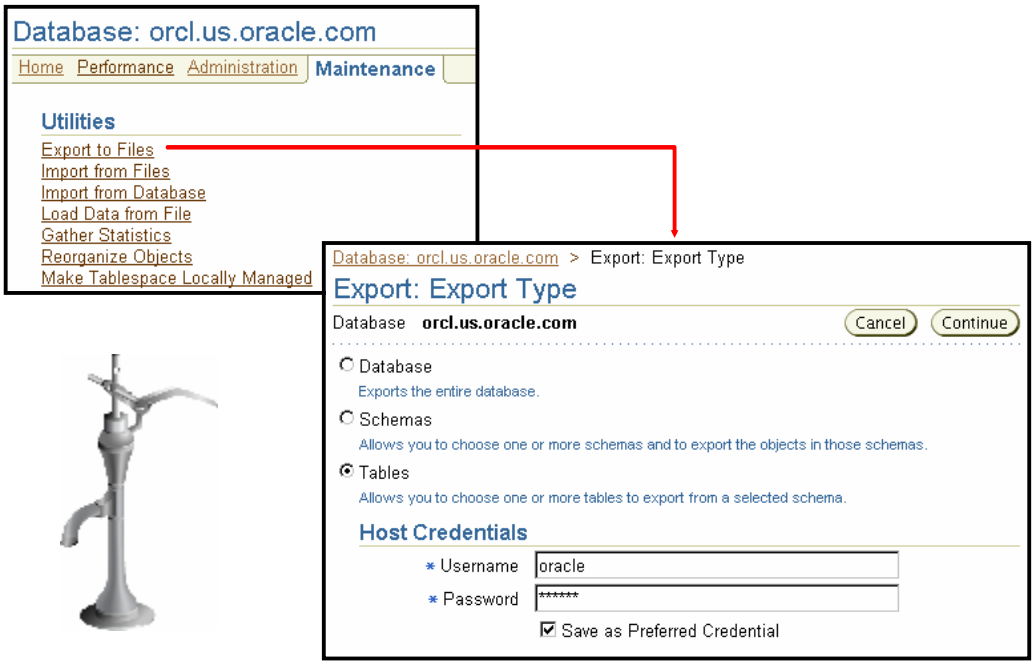
`NOT NULL` columns
- `INSERT`: Cannot insert without a value
- `UPDATE`: Cannot modify the value to `NULL`

`UNIQUE` key columns:
- `INSERT` and `UPDATE`: Cannot have the same value as any other column in the table, except for null. Each null is considered unique, so all rows could have a null value in a column with a `UNIQUE` key constraint.

`PRIMARY KEY` columns:
- Apply the rules for `UNIQUE` key and `NOT NULL` columns.

`CHECK` columns:
- `INSERT` and `UPDATE`: Values must meet the condition of the constraint.

Data Pump Export



ORACLE

Data Pump Export

Oracle's Data Pump utility enables high-speed transfer of data from one database to another. For example, you may want to export a table and its associated indexes from one database and import it into another database. For more information about Data Pump see the *Oracle Database Utilities Guide*.

On the maintenance page click Export to Files to use Data Pump to write data out to a file.

1. On the first page, you select what you want to export, and enter in the host login credentials in the Username and Password fields:
   - Database: This exports the entire database, and is also called a full export. The following system schemas are not exported as part of a full export because the metadata they contain is exported as part of other objects in the dump file set: `SYS`, `ORDSYS`, `ORDPLUGINS`, `CTXSYS`, `MDSYS`, `LBACSYS`, and `XDB`. This mode requires that you have the `EXP_FULL_DATABASE` role.

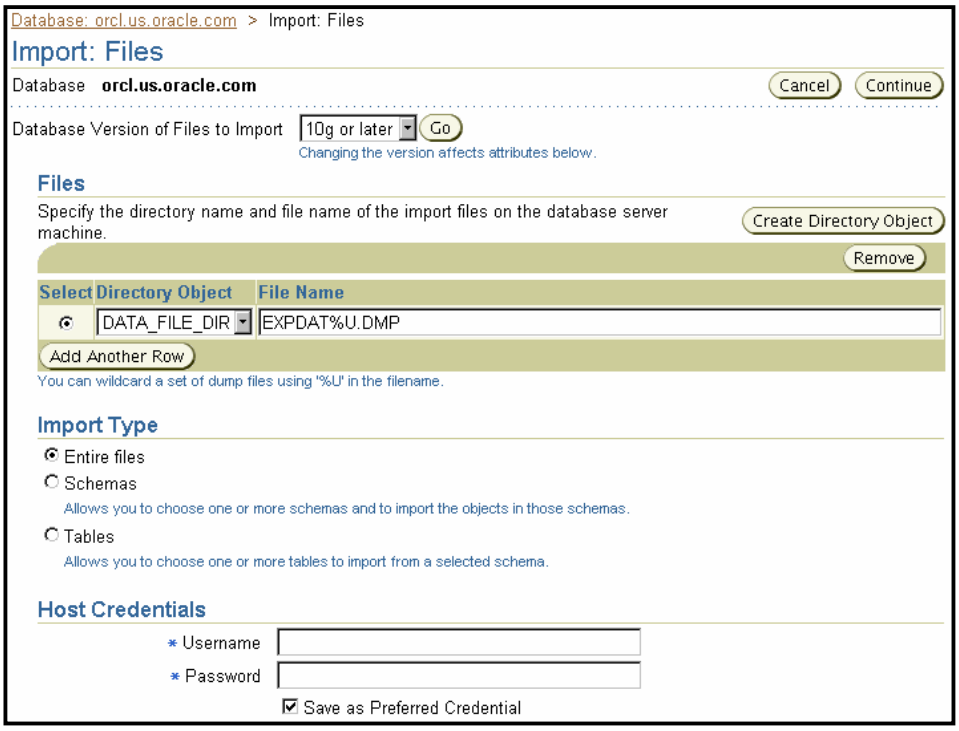
Data Pump Export (continued)

- Schemas: This exports all objects of one or more schemas. If you do not have the EXP_FULL_DATABASE role, then you can specify only your own schema. Cross-schema references are not exported unless the referenced schema is also specified in the list of schemas to be exported.
- Tables: In table mode, only a specified set of tables, partitions, and their dependent objects are unloaded. You must have the EXP_FULL_DATABASE role to specify tables that are not in your own schema, and only one schema can be specified. Note that type definitions for columns are not exported in table mode. It is expected that the type definitions already exist in the target instance at import time. Also, as in schema exports, cross-schema references are not exported.

2. If you select Schemas or Tables for the export type, the next page will allow you to select the set of Schemas or Tables that you want to export. Click Add to select the Schemas or Tables you want to export.
3. The Options page is the next step, if you selected a Database export you will go directly to this page after selecting the type of export. The Maximum Number of Threads in Export Job is the degree of parallelism to be used. Here you can see and estimate of how big the export will be, select blocks for a quick estimate, and statistics for a more precise estimate. Click Estimate Disk Space Now to see the results. You also can specify information about the optional log file here. Click Show Advanced Options for the following:
   - Content: You can export just the metadata or the data, or both. You can also set up a condition to filter out by object name.
   - Flashback: Get a consistent export to a particular time.
   - Query: Get only data that matches a WHERE clause that you provide.
4. The next page is Files. You use the Files page to specify the directory name and file name, and the maximum size for the export files. The %U is a substitution variable that begins with 01 for the first file and then goes to 02, 03, and so on. Enough dump files are created to allow all processes specified by the current setting of the PARALLEL parameter to be active. For example, if expa%U, expb%U, and expc%U were all specified for a job having a parallelism of 6, the initial dump files created would be expa01.dmp, expb01.dmp, expc01.dmp, expa02.dmp, expb02.dmp, and expc02.dmp.
5. The next page is the Schedule. You use the Schedule page to schedule the export job.
6. The last page is Review. You use the Review page to display the PL/SQL code for the export command generated by the selections you made in the previous pages of the wizard. You cannot change the code generated. You can copy out for creation of a script file if needed. Click Submit Job to begin the export process.

After the job is submitted, click View Job to monitor the job progress.

Data Pump Import



ORACLE

Data Pump Import

Use Data Pump Import to load data extracted by Data Pump Export. On the Maintenance page click Import from Files to use Data Pump Import to read files written by Data Pump Export back into a database.

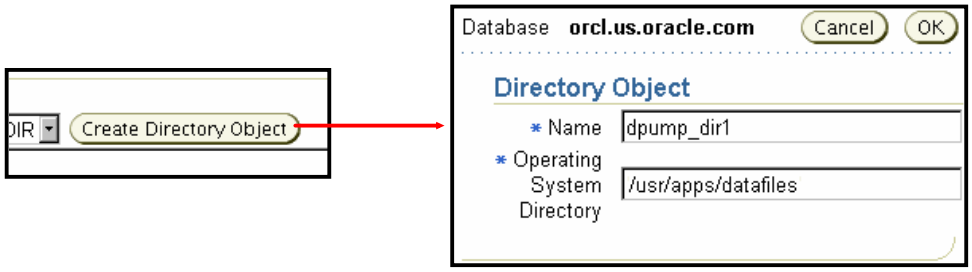
1. Files: Use the Files page to designate the path and name of the import files on the database server. Then you must select one of three possible import choices—whether to import entire files, specific schemas, or specific tables. You can also choose to generate a log file and then enter credentials in the host Username and Host Password fields.
2. Select Objects: This page is displayed only when you choose the option on the Files page to select specific objects to import. Use this page to select the users you want to import from the Available Users list and move them to the Selected Users list.

Data Pump Import (continued)

3. Re-Mapping: This page is displayed only if you previously selected objects in the Select Objects page. Use this page to designate whether to import each user's data into the same user's schema, or to import into different user's schema as designated in the Source User and Destination User fields.
4. Options: Use the Options page to set options for data file reuse, object creation error, and unusable indexes.
5. Schedule: Use the Schedule page to schedule the import procedure as a job in the Job system.
6. Review: Use the Review page to display the PL/SQL code for the import command generated by the selections you made in the previous pages of the wizard. You can edit the PL/SQL code manually or click Import to begin the import process.

After the job is submitted, click View Job to monitor the job progress.

DIRECTORY Objects




```
SQL> CREATE DIRECTORY dpump_dir1
  2  AS '/usr/apps/datafiles';

Directory created.

SQL> SELECT * FROM DBA_DIRECTORIES
  2  WHERE DIRECTORY_NAME = 'DPUMP_DIR1';

OWNER   DIRECTORY_NAME  DIRECTORY_PATH
------- --------------- --------------------
SYS     DPUMP_DIR1      /usr/apps/datafiles
```

ORACLE

DIRECTORY Objects

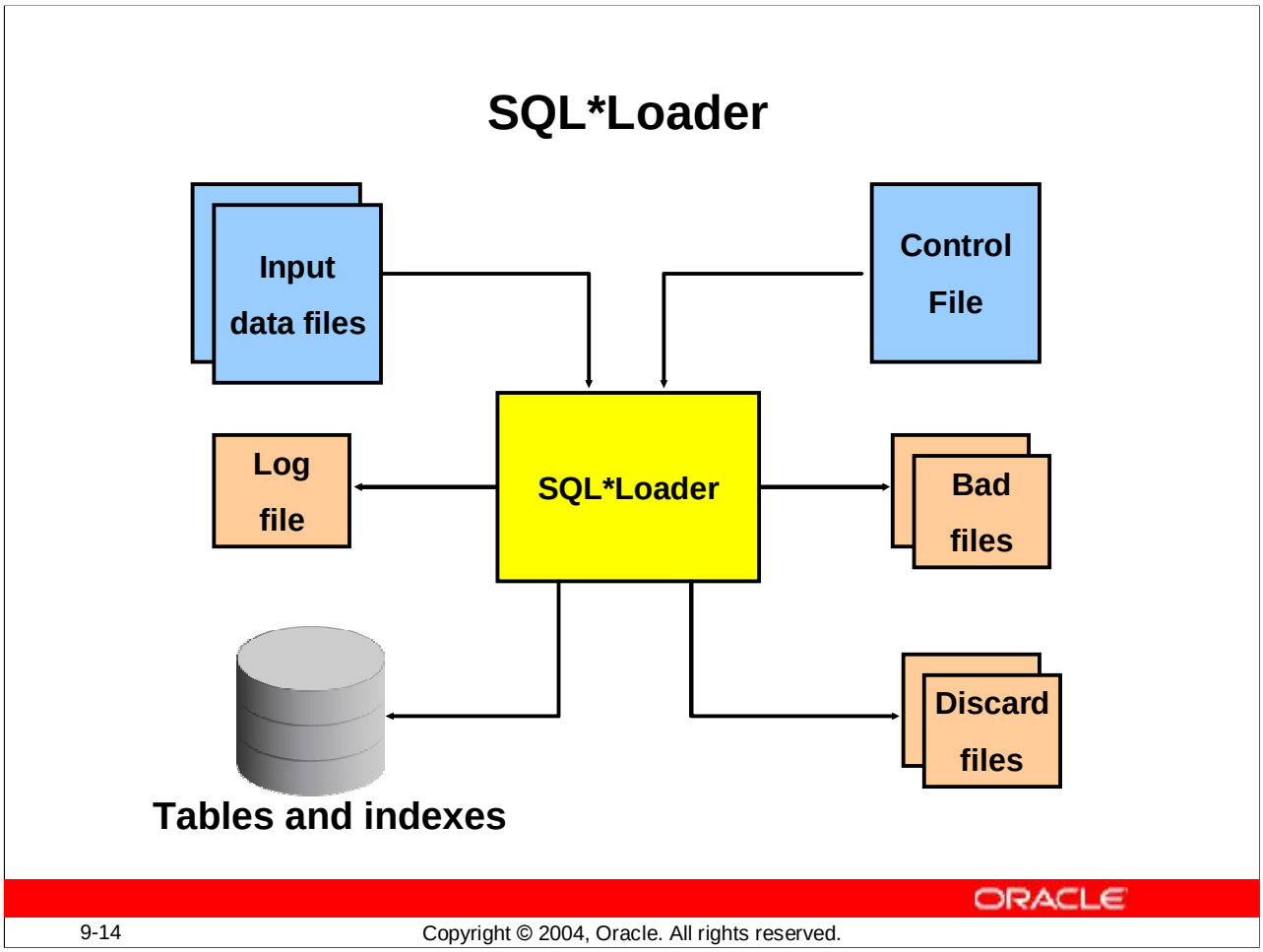
Because Data Pump is server based (rather than client based), dump files, log files, and SQL files are accessed relative to server-based directory paths. Data Pump requires that you specify directory paths as directory objects. A directory object maps a name to a directory path on the file system.

Use the `CREATE DIRECTORY` statement to create a directory object, or click Create Directory Object. A directory object specifies an alias for a directory on the server's file system. You must have `CREATE ANY DIRECTORY` system privileges to create directories.

When you create a directory, you are automatically granted the read and write object privileges on the directory, and you can grant these privileges to other users and roles.

To view the definition of directory object use the `DBA_DIRECTORIES` view.

Note: Oracle does not verify that the directory you specify actually exists. Therefore, take care that you specify a valid directory in your operating system. In addition, if your operating system uses case-sensitive path names, be sure you specify the directory in the correct format. (However, you need not include a trailing slash at the end of the path name). Remember that the operating system will control security for files in the directory.



SQL*Loader

SQL*Loader loads data from external files into tables of an Oracle database. It has a powerful data parsing engine that puts little limitation on the format of the data in the data file. The files used by SQL*Loader are as follows:

Input Data Files: SQL*Loader reads data from one or more files (or operating system equivalents of files) specified in the control file. From SQL*Loader's perspective, the data in the data file is organized as records. A particular data file can be in fixed record format, variable record format, or stream record format. The record format can be specified in the control file with the `INFILE` parameter. If no record format is specified, the default is stream record format.

Control File: The control file is a text file written in a language that SQL*Loader understands. The control file tells SQL*Loader where to find the data, how to parse and interpret the data, where to insert the data, and more. Although not precisely defined, a control file can be said to have three sections.

- The first section contains session wide information, for example:
  - Global options such as bind size, rows, records to skip, and so on
  - `INFILE` clauses to specify where the input data is located
  - Data to be loaded

SQL*Loader(continued)

- The second section consists of one or more `INTO TABLE` blocks. Each of these blocks contains information about the table into which the data is to be loaded, such as the table name and the columns of the table.
- The third section is optional and, if present, contains input data.

Log File: When SQL*Loader begins execution, it creates a log file. If it cannot create a log file, execution terminates. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load.

Bad File: The bad file contains records that were rejected, either by SQL*Loader or by the Oracle database. Data file records are rejected by SQL*Loader when the input format is invalid. After a data file record is accepted for processing by SQL*Loader, it is sent to the Oracle database for insertion into a table as a row. If the Oracle database determines that the row is valid, then the row is inserted into the table. If the row is determined to be invalid, then the record is rejected and SQL*Loader puts it in the bad file.

Discard File: This file is created only when it is needed, and only if you have specified that a discard file should be enabled. The discard file contains records that were filtered out of the load because they did not match any record-selection criteria specified in the control file.

For more information on SQL*Loader refer to the *Oracle Database Utilities* documentation.

The SQL*Loader Control File

The loader control file tells SQL*Loader:

- **Where to find the load data**
- **The data format**
- **Configuration details:**
  - **Memory management**
  - **Record rejection**
  - **Interrupted load handling details**
- **How to manipulate the data**

ORACLE

The SQL*Loader Control File

The SQL*Loader control file is a text file that contains data definition language (DDL) instructions. DDL is used to control the following aspects of a SQL*Loader session:

- Where SQL*Loader finds the data to load
- How SQL*Loader expects that data to be formatted
- How SQL*Loader configures (memory management, rejecting records, interrupted load handling, and so on) as it loads the data
- How SQL*Loader manipulates the data being loaded

The SQL*Loader Control File (continued)

The example below illustrates a typical SQL*Loader control file.

```
1   -- This is a sample control file
2  LOAD DATA
3  INFILE 'SAMPLE.DAT'
4  BADFILE 'sample.bad'
5  DISCARDFILE 'sample.dsc'
6  APPEND
7  INTO TABLE emp
8  WHEN (57) = '.'
9  TRAILING NULLCOLS
10 (hiredate SYSDATE,
       deptno POSITION(1:2) INTEGER EXTERNAL(3)
        NULLIF deptno=BLANKS,
       job POSITION(7:14) CHAR TERMINATED BY WHITESPACE
       NULLIF job=BLANKS "UPPER(:job)",
       mgr POSITION(28:31) INTEGER EXTERNAL
       TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
        ename POSITION(34:41) CHAR
       TERMINATED BY WHITESPACE "UPPER(:ename)",
       empno POSITION(45) INTEGER EXTERNAL
       TERMINATED BY WHITESPACE,
       sal POSITION(51) CHAR TERMINATED BY WHITESPACE
       "TO_NUMBER(:sal,'$99,999.99')",
       comm INTEGER EXTERNAL ENCLOSED BY '(' AND '%'
       ":comm * 100"
    )
```

Sample control file explanation, by line numbers:

1. Comments can appear anywhere in the command section of the file, but they should not appear within the data. Precede any comment with two hyphens. All text to the right of the double hyphen is ignored, until the end of the line.
2. The `LOAD DATA` statement tells SQL*Loader that this is the beginning of a new data load. If you are continuing a load that was been interrupted in progress, then use the `CONTINUE LOAD DATA` statement.
3. The `INFILE` keyword specifies the name of a data file containing data that you want to load.

The SQL*Loader Control File (continued)

4. The BADFILE keyword specifies the name of a file into which rejected records are placed.
5. The DISCARDFILE keyword specifies the name of a file into which discarded records are placed.
6. The APPEND keyword is one of the options you can use when loading data into a table that is not empty. To load data into a table that is empty, you use the INSERT keyword.
7. The INTO TABLE keyword enables you to identify tables, fields, and data types. It defines the relationship between records in the data file and tables in the database.
8. The WHEN clause specifies one or more field conditions that each record must match before SQL*Loader will load the data. In this example SQL*Loader will only load the record if the 57th character is a decimal point. That decimal point delimits dollars and cents in the field and causes records to be rejected if SAL has no value.
9. The TRAILING NULLCOLS clause tells SQL*Loader to treat any relatively positioned columns that are not present in the record as null columns.
10. The remainder of the control file contains the field list, which provides information about column formats in the table that is being loaded.

Control File Syntax Considerations

- **The syntax is free-format.**
- **Syntax is not case sensitive.**
- **Comments extend from the two hyphens (--) that mark the beginning of the comment to the end of the line.**
- **The `CONSTANT` keyword is reserved.**

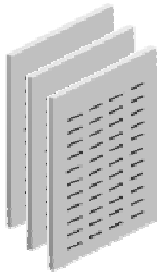
ORACLE

Control File Syntax Considerations

- The syntax is free-format. (Statements can extend over multiple lines.)
- It is case insensitive; however, strings enclosed in single or double quotation marks are taken literally, including case.
- In control file syntax, comments extend from the two hyphens (--) that mark the beginning of the comment to the end of the line. The optional third section of the control file is interpreted as data rather than as control file syntax; consequently, comments in this section are not supported.
- The `CONSTANT` keyword has special meaning to SQL*Loader and is therefore reserved. To avoid potential conflicts, do not use the word `CONSTANT` as a name for any tables or columns.

Input Data and Data Files

- SQL*Loader reads data from one or more files specified in the control file.
- From SQL*Loader's perspective, the data in the data file is organized as records.
- A data file can be in one of three formats:
  - Fixed-record format
  - Variable-record format
  - Stream-record format



ORACLE

Input Data and Data Files

Fixed-Record Format

A file is in fixed-record format when all records in a data file are the same byte length. Although this format is the least flexible, it results in better performance than variable or stream format. Fixed-record format is also simple to specify. For example:

```
INFILE <datafile_name> "fix n"
```

This example specifies that SQL*Loader should interpret the particular data file as being in fixed-record format where every record is *n* bytes long.

The following example shows a control file that specifies a fixed-record format data file. The data file contains four physical records. The first record is `[0001, abcd]`, which is exactly nine bytes long (using a single-byte character set) and the carriage return is the tenth byte.

```
load data
infile 'example.dat' "fix 10"
into table example
fields terminated by ','
(col1, col2)
example.dat:
```

Input Data and Data Files (continued)

```
0001,abcd
0002,fghi
0003,klmn
```

Variable-Record Format

A file is in variable-record format when the length of each record in a character field is included at the beginning of each record in the data file. This format provides some added flexibility over the fixed-record format and a performance advantage over the stream-record format. For example, you can specify a data file that is to be interpreted as being in variable-record format as follows:

```
INFILE "datafile_name" "var n"
```

In this example, ***n*** specifies the number of bytes in the record length field. If ***n*** is not specified, SQL*Loader assumes a length of 5. Specifying ***n*** larger than 40 results in an error. The following example shows a control file specification that tells SQL*Loader to look for data in the `example.dat` data file and to expect variable-record format where the record length fields are 3 bytes long. The `example.dat` data file consists of three physical records. The first is specified to be 009 (that is, nine) bytes long, the second is 010 bytes long (including a one-character newline), and the third is 012 bytes long. This example also assumes a single-byte character set for the data file.

```
load data
infile 'example.dat' "var 3"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1 char(5),col2 char(7))
example.dat:
009hello,cd,
010world,im,
012my,name is,
```

Stream-Record Format

A file is in stream-record format when the records are not specified by size; instead SQL*Loader forms records by scanning for the ***record terminator***. Stream-record format is the most flexible format, but there can be a negative effect on performance. The specification of a data file that is to be interpreted as being in stream-record format looks similar to the following:

```
INFILE <datafile_name> ["str terminator_string"]
```

The `terminator_string` is specified as either '`char_string`' or X'`hex_string`'

where:

- '`char_string`' is a string of characters enclosed in single or double quotation marks
- X'`hex_string`' is a byte string in hexadecimal format

Input Data and Data Files (continued)

When the terminator_string contains special (nonprintable) characters, it should be specified as a X'hex_string'. However, some nonprintable characters can be specified as ('char_string') by using a backslash. For example:

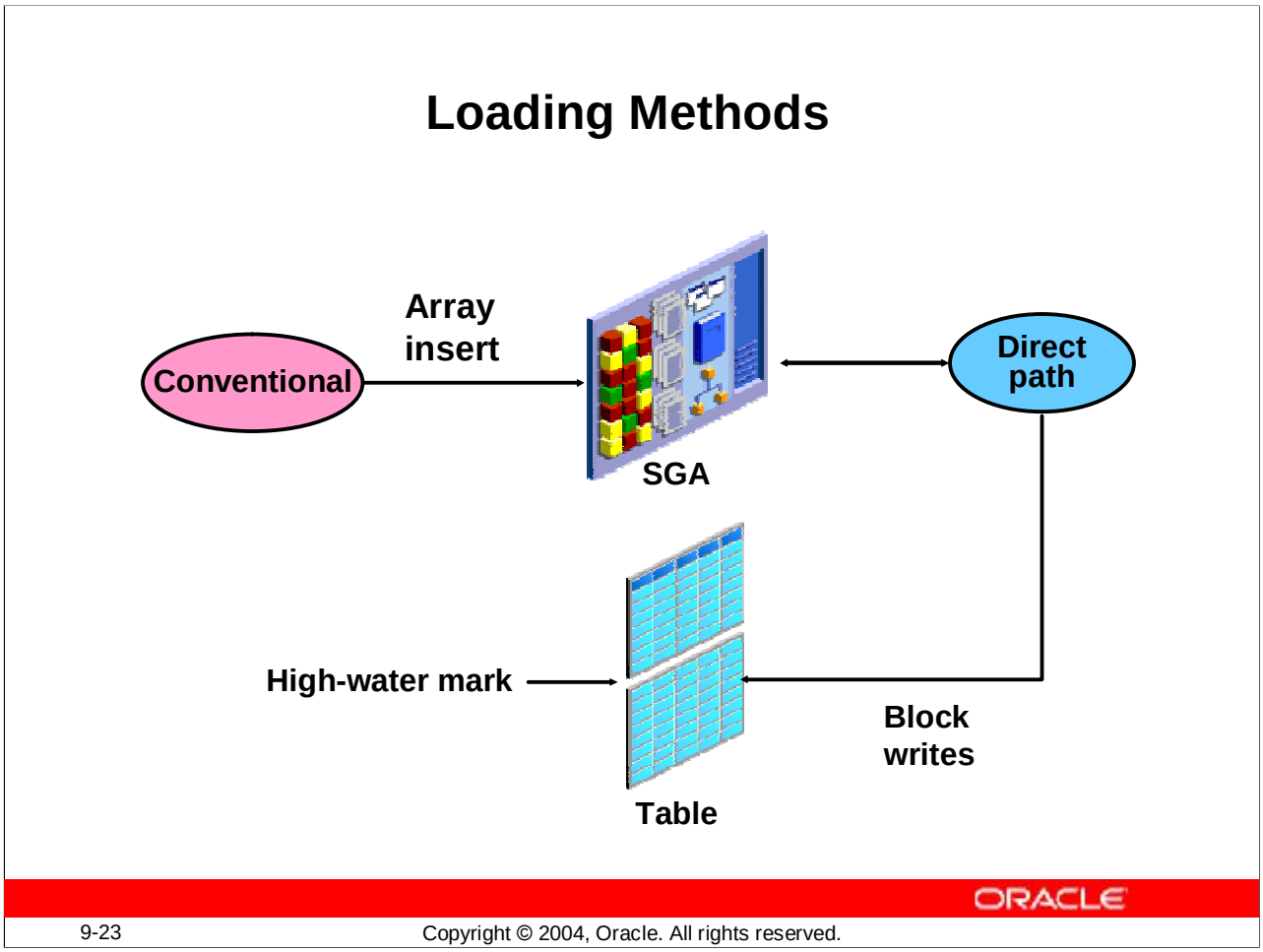
- \n linefeed (newline)
- \t horizontal tab
- \f formfeed
- \v vertical tab
- \r carriage return

If the character set specified with the NLS_LANG parameter for your session is different from the character set of the data file, character strings are converted to the character set of the data file.

Hexadecimal strings are assumed to be in the character set of the data file, so no conversion is performed. If no terminator_string is specified, it defaults to the new line (end-of-line) character (line feed in UNIX-based platforms, carriage return followed by a line feed on Microsoft platforms, and so on). The newline character is connected to the character set of the data file.

The following example illustrates loading data in stream-record format where the terminator string is specified using a character string, '|\n'. The use of the backslash character allows the character string to specify the nonprintable line feed character.

```
load data
infile 'example.dat' "str '|\n'"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1 char(5),
col2 char(7))
example.dat:
hello,world,|
james,bond,|
```



Loading Methods

SQL*Loader provides two methods for loading data:

- Conventional path
- Direct path

Conventional Path Load

Conventional path load builds an array of rows to be inserted and uses the SQL `INSERT` statement to load the data. During conventional path loads, input records are parsed based on field specifications, and an array of records is built and inserted into the table specified in the control file. Records that do not conform to the field specifications are rejected and those records that do not satisfy the selection criteria are discarded.

Conventional path loads can be used to load data into both the clustered and unclustered tables. Redo generation is controlled by the logging attribute for the table that is being loaded.

Loading Methods (continued)

Direct Path Load

A direct path load builds blocks of data in memory and saves these blocks directly into the extents allocated for the table being loaded. Online redo log file entries are not generated unless the database is in `ARCHIVELOG` mode. Direct path loads use the field specifications to build whole Oracle blocks of data, and write the blocks directly to Oracle data files above the high water mark. The high-water mark is the highest point to which data has been written into the table so far. Direct path load bypasses the database buffer cache and accesses the SGA only for extent management and adjustment of the high-water mark.

Comparing Direct and Conventional Path Loads

Conventional Load	Direct Path Load
Uses COMMIT to make changes permanent	Uses data saves
Redo entries always generated	Generates redo only under specific conditions
Enforces all constraints	Enforces only PRIMARY KEY, UNIQUE, and NOT NULL
INSERT triggers fire	INSERT triggers do not fire
Can load into clustered tables	Cannot load into clustered tables
Other users can make changes to tables	Other users cannot make changes to tables

ORACLE

Comparing Direct and Conventional Path Loads

Method of Saving Data

Conventional path loads use SQL processing and a database `COMMIT` for saving data. The insertion of an array of records is followed by a commit operation. Each data load might involve several transactions.

Direct path loads use data saves to write blocks of data to Oracle data files. The following features differentiate a data save from a `COMMIT`:

- During a data save, only full database blocks are written to the database.
- The blocks are written after the high-water mark of the table.
- After a data save, the high-water mark is moved.
- Internal resources are not released after a data save.
- A data save does not end the transaction.
- Indexes are not updated at each data save.

Comparing Direct and Conventional Path Loads (continued)

Logging Changes

Conventional path loading generates redo entries similar to any DML statement. When using a direct path load, redo entries are not generated if:

- The database is in NOARCHIVELOG mode
- The database is in ARCHIVELOG mode, but logging is disabled. Logging can be disabled by setting the NOLOGGING attribute for the table or by using the UNRECOVERABLE clause in the control file.

Enforcing Constraints

During a conventional path load, all enabled constraints are enforced in the same way that they are during any DML operation.

During direct path loads, the constraints are handled as follows:

- NOT NULL constraints are checked when arrays are built.
- FOREIGN KEY and CHECK constraints are disabled, and can be enabled at the end of the run by using the appropriate commands in the control file. FOREIGN KEY constraints are disabled because they reference other rows or tables, and CHECK constraints are disabled because they may use SQL functions. If only a small number of rows are to be inserted into a large table, use conventional loads.
- PRIMARY KEY and UNIQUE constraints are checked during and at the end of the run, and may be disabled if they are violated.

Firing INSERT Triggers

WHILE INSERT triggers are fired during conventional loads; they are disabled before a direct path load and reenabled at the end of the run. They may remain disabled if a referenced object is not accessible at the end of the run. Consider using conventional path loads to load data into tables with INSERT triggers.

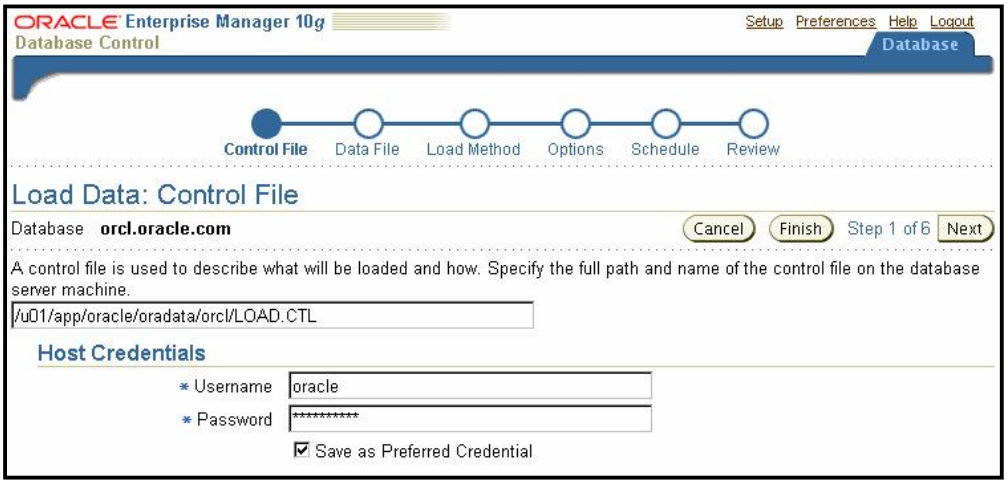
Loading into Clustered Tables

Direct loads cannot be used to load rows into clustered tables. Clustered tables can be loaded using conventional path loads only.

Locking

While a direct path load is in progress, other transactions cannot make changes to the tables that are being loaded. The only exception to this rule is when several parallel direct load sessions are used concurrently.

Loading Data with SQL*Loader



Loading Data with SQL*Loader

Use the Load Data from File Wizard to load data from a non-Oracle database into an Oracle database. To display the wizard, select Load Data from File on the Utilities section of the Database home page Maintenance subtab.

You can load data as follows:

1. Log in to Enterprise Manager as SYS or another user with administrator privileges.
2. Click Load Data from File under Utilities on the Maintenance page to invoke a six step loading wizard.
3. Enter the full path of your control file on the database server machine on the Load Data: Control File page. Also enter the username and password for the host machine. Click Next.
4. Select “Provide the full path and name on the database server machine” on the Load Data: Data File page and enter the path. Click Next.
5. The default is Conventional Path for the loading method on the Load Data: Load Method page. You may choose Direct Path if needed.

Loading Data with SQL*Loader (continued)

6. Select "Generate log file where logging information is to be stored" under Optional Files on the Load Data: Options page. You can accept the default file name and path or enter a different one. Note that this page gives you the option of limiting the number of rows loaded. Click Next
7. Enter a Job Name and Description on the Load Data: Schedule page. Select Immediately to run the job now. Click Next.
8. The Load Data: Review page allows you to review your file names and loading methods. If you want to change something, you can click Back. Otherwise, click Submit Job to start the loading.
9. The Status page appears with the Load Data Submit Successful message. Click View Job to view the job summary. The Summary page should indicate that the job has succeeded. You can view the log file by clicking your job under the Logs heading or by viewing the log file directly.
10. You can confirm the loaded data by navigating to the Tables page , selecting the table, and choosing View Data as the action.

Summary

In this lesson, you should have learned how to:

- **Manipulate data through SQL**
- **Use Data Pump to export data**
- **Use Data Pump to import data**
- **Load data with SQL Loader**

ORACLE

Practice 9:
Using Data Pump Export and Import

This practice covers the following:

- **Creating a directory object**
- **Extracting the HR schema objects**
- **Using Data Pump import to load tables into a different schema**

ORACLE

Practice 9: Managing Data

Background: You need to create additional tables for the new inventory application and load data into your existing tables.

Tasks:

- Use Data Pump to export data from the database.
- Use Data Pump to import data into the database.
- Use Data Pump to move objects and data between schemas.
- Use SQL Loader to load data from a text file into the database.

1. Export the HR schema to a file named `hrexport.dmp`. All the files for the export need to go into the home directory of the operating system user "`oracle`".
2. Import EMPLOYEES, DEPARTMENTS and LOCATIONS tables into INVENTORY schema and the INVENTORY tablespace using the `hrexport.dmp` you created in step 1.
3. Use Database Control to verify that you have successfully copied data from the HR schema to the INVENTORY schema using export/import.
4. Use SQL Loader to load data from text files into the `product_master` and `product_on_hand` tables.
   a) Use the control file `/home/oracle/labs/lab09_04_a.ctl` to load data from the text file `/home/oracle/labs/lab09_04_a.dat` into the `product_master` table.
   b) Use the control file `/home/oracle/labs/lab09_04_f.ctl` to load data from the text file `/home/oracle/labs/lab09_04_f.dat` into the `product_on_hand` table.

10

PL/SQL

Objectives

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

- **Identify PL/SQL objects**
- **Describe triggers and triggering events**
- **Identify configuration options that affect PL/SQL performance**

ORACLE

PL/SQL

Procedural Language/Structured Query Language (PL/SQL) is a fourth generation (4GL) programming language. PL/SQL provides:

- **Procedural extensions to SQL**
- **Portability across platforms and products**
- **Support for object-oriented programming**

ORACLE

PL/SQL

PL/SQL is an Oracle proprietary fourth generation programming language that provides procedural extensions to the SQL language. PL/SQL provides a common programming environment for Oracle databases and applications regardless of the operating system or hardware platform.

With PL/SQL, you can manipulate data with SQL statements and control program flow with procedural constructs such as IF-THEN, CASE, and LOOP. You can also declare constants and variables, define procedures and functions, use collections and object types, and trap run-time errors. PL/SQL program can also call programs written in other languages such as C, C++, and Java.

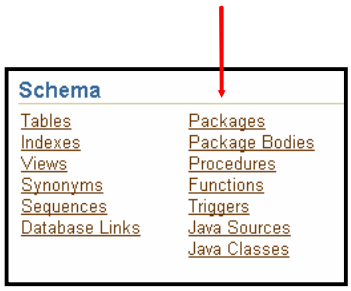
Because it runs inside the database, PL/SQL code is very efficient for data-intensive operations, and minimizes network traffic in applications.

For more details on procedural constructs and uses of PL/SQL, refer to *PL/SQL User's Guide and Reference* documentation.

Administering PL/SQL Objects

Database administrators should be able to:

- **Identify problem PL/SQL objects**
- **Recommend appropriate use of PL/SQL**
- **Load PL/SQL objects into the database**
- **Assist PL/SQL developers in troubleshooting**



ORACLE

Administering PL/SQL Objects

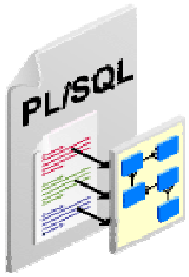
As the DBA, you are not usually responsible for loading PL/SQL code into the database and for assisting developers in troubleshooting. You are also not usually expected to write applications using PL/SQL, but you should be familiar with the different PL/SQL objects sufficiently to make recommendations to application developers and identify problem objects.

In Database Control you can access PL/SQL objects in the Administration tab, under Schema. When you click the object type you can view, modify, and create the type of PL/SQL object selected.

PL/SQL Objects

There are many types of PL/SQL database objects:

- **Package**
- **Package body**
- **Type body**
- **Procedure**
- **Function**
- **Trigger**

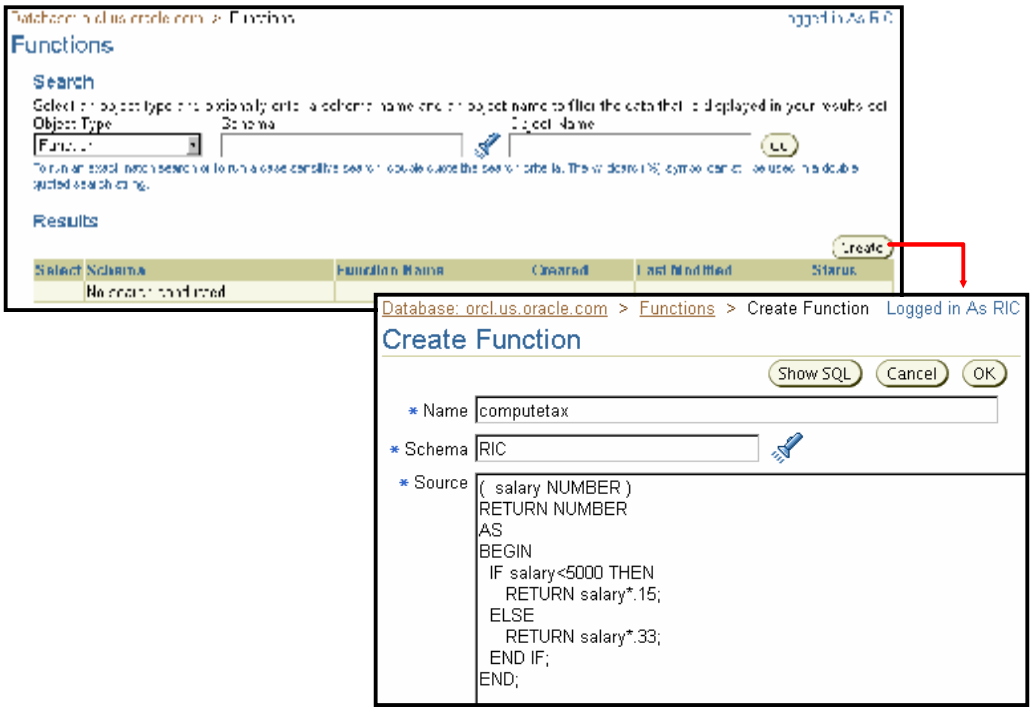


ORACLE

PL/SQL Objects

- Packages: A package is a collection of procedures, and functions that are logically related. This part of a package is also called the specification (or *spec*), and is the interface to your applications; it declares the types, variables, constants, exceptions, cursors, and subprograms available for use.
- Package body: The body fully defines cursors and subprograms, and so implements the spec. The body holds implementation details and private declarations, which are hidden from your application.
- Type body: A collection of methods (procedures and functions) associated with user-defined data types. For more information on user-defined data types, refer to *Oracle Database Application Developer's Guide – Object Relational Features*
- Procedures: A procedure is a PL/SQL block that performs a specific action.
- Functions: A function is a PL/SQL block that returns a single value by using the `RETURN` PL/SQL command.
- Triggers: A trigger is a PL/SQL block that is executed when a particular event happens in the database. These events can be based on a table, such as when a row is inserted into the table. They can also be database events, such as when a user logs in to the database.

Functions



ORACLE

Functions

PL/SQL functions are typically used to compute a value. There are many built-in functions such as SYSDATE, SUM, AVG, and TO_DATE. Developers also create their own functions when writing applications. The code for a PL/SQL function *must* contain a RETURN statement. PL/SQL functions are invoked by selecting from them as show above.

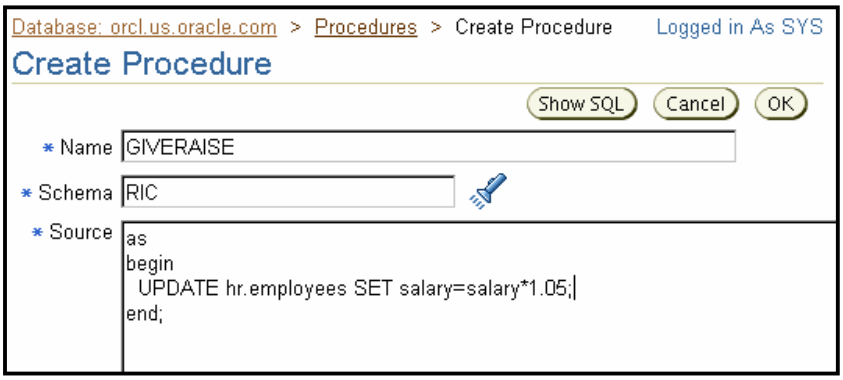
The `computetax` function shown in the slide is created with the SQL command:

```
CREATE OR REPLACE FUNCTION computetax (salary NUMBER)
RETURN NUMBER
AS
BEGIN
  IF salary<5000 THEN
     RETURN salary*.15;
  ELSE
     RETURN salary*.33;
  END IF;
END;
/
```

Procedures

Procedures are used to perform a specific action.
Procedures:

- **Transfer values in and out through an argument list**
- **Are called with the CALL command**



ORACLE

Procedures

PL/SQL procedures perform a specific action. The procedure shown in the slide produces the following results:

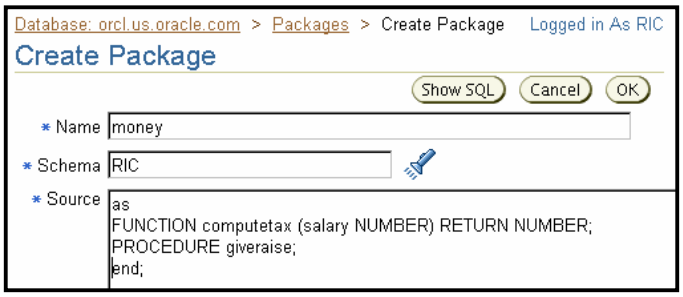
```
SQL> SELECT sum(salary) FROM hr.employees;
SUM(SALARY)
-----------
     691400

SQL> call giveraise();
Call completed.

SQL> SELECT sum(salary) FROM hr.employees;
SUM(SALARY)
-----------
     725970
```

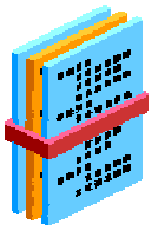
Like functions, procedures can accept input values and perform conditional statements such as IF-THEN, CASE, and LOOP.

Packages

Packages are collections of functions and procedures. Each package should consist of two objects:

- **Package specification**
- **Package body**





ORACLE

Packages

Packages are groupings of functions and procedures. There are performance and maintainability advantages in grouping functions and procedures into a single package. Each package should be made up of two separately compiled database objects:

- Package specification, sometimes known as the package header. This object has an object type of PACKAGE and only contains the definition of the procedures functions, and variables for the package.
- Package body. This object has an object type of `PACKAGE BODY` and contains the actual code for the subprograms defined in the package specification.

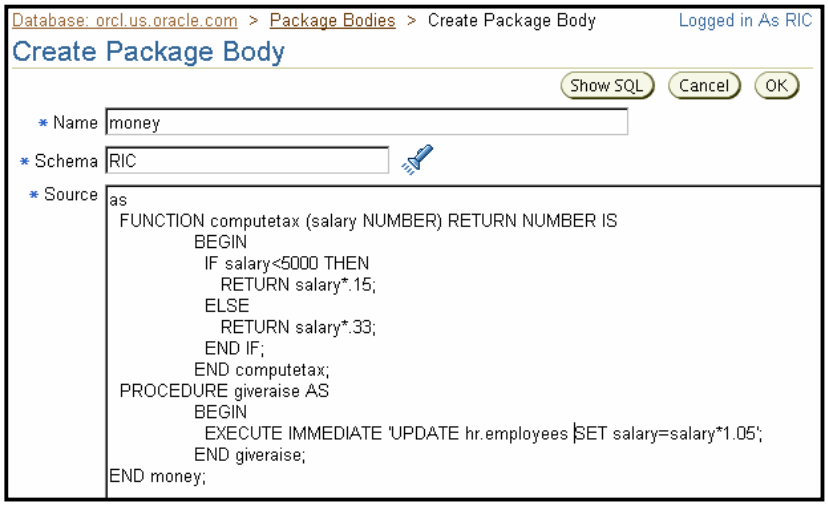
Procedures and functions called from within a package are called using dot notation:

```
package_name.procedure or function name
```

In the package shown above the subprograms are executed as follows:

```
SQL> SELECT money.computetax(salary) FROM hr.employees
     WHERE employee_id=107;
SQL> EXECUTE money.giveraise;
```

Package Body



ORACLE

Package Body

Package bodies:

- Are separate from the package specifications. Because of this, the code of the body can be changed and recompiled, and other objects that are dependant on the specification are not marked invalid.
- Contain the code for subprograms defined in the package specification. This is where the work is done. The specification shows how to call subprograms within the package; the body is the code section.
- Cannot be compiled unless the package specification has already been compiled. You can create a specification without a body, but you cannot create a body without a specification.
- May be wrapped to hide details of the code. Wrap is a stand-alone program that obfuscates PL/SQL source code so that you can deliver PL/SQL applications without exposing your source code. For more information on the use of wrap see the *PL/SQL User's Guide and Reference*.

Package Body (continued)

The code for the money package body is as follows:

```
CREATE OR REPLACE PACKAGE BODY money AS
FUNCTION computetax (salary NUMBER) RETURN NUMBER IS
      BEGIN
        IF salary<5000 THEN
           RETURN salary*.15;
        ELSE
           RETURN salary*.33;
        END IF;
      END computetax;
PROCEDURE giveraise AS
      BEGIN
        EXECUTE IMMEDIATE 'UPDATE hr.employees -
                       SET salary=salary*1.05';
      END giveraise;
END money;
/
```

Built-In Packages

Oracle Database 10*g* comes with over 350 built-in PL/SQL packages providing:

- **Administration and maintenance utilities**
- **Extended functionality**

Use the DESCRIBE command to view subprograms

```
SQL> DESCRIBE dbms_stats
PROCEDURE ALTER_DATABASE_TAB_MONITORING
 Argument Name    Type     In/Out    Default?
 ---------------- -------  ------    --------
 MONITORING       BOOLEAN      IN    DEFAULT
 SYSOBJS          BOOLEAN      IN    DEFAULT
...
```


Built-In Packages

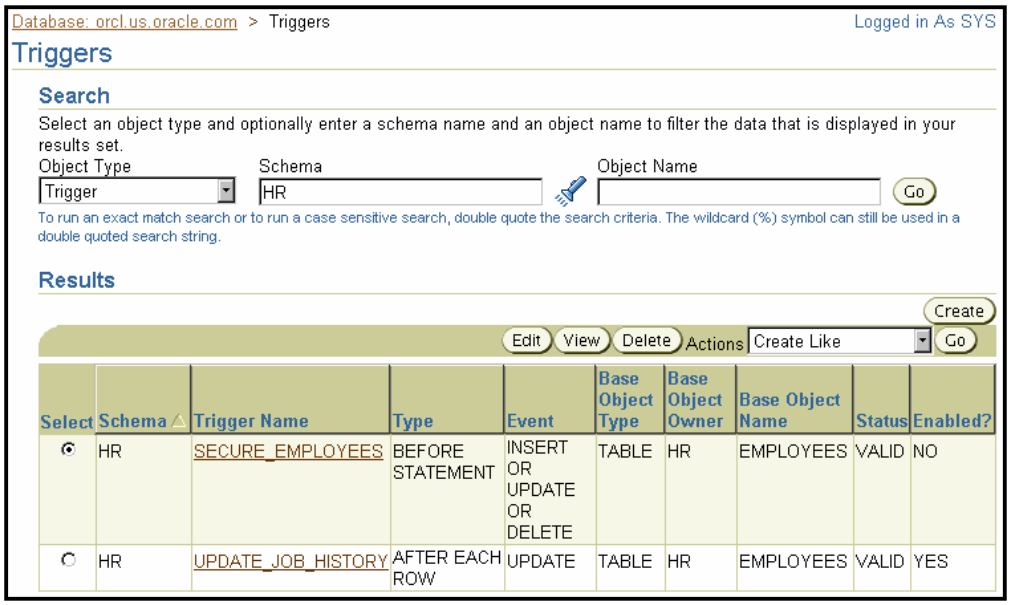
The built-in PL/SQL packages supplied with the Oracle Database 10*g* provide access to extended database functionality such as advanced queuing, encryption, and heterogeneous services. They also include many administration and maintenance utilities.

Which packages an administrator uses depends on the type of applications the database serves. A few of the more common administration and maintenance packages are:

- `DBMS_STATS`: Gather, view, and modify optimizer statistics
- `DBMS_TTS`: Validate transportable tablespaces
- `DBMS_WORKLOAD_REPOSITORY`: Manage the ADDM repository
- `DBMS_SESSION`: PL/SQL access to `ALTER SESSION` and `SET ROLE` statements
- `DBMS_RLS`: Administer the Virtual Private Database
- `DBMS_RESOURCE_MANAGER`: Maintain resource manager objects
- `DBMS_OBFUSCATION_TOOLKIT`: Encrypt, decrypt, and compute checksums
- `DBMS_SCHEDULER`: Scheduling functions and procedures that are callable from any PL/SQL program

For details on these and other built in packages, see the *PL/SQL Packages and Types Reference* manual.

Triggers



Triggers

Triggers are PL/SQL code objects stored in the database that automatically run or "fire" when something happens. Oracle Database 10*g* allows many actions to server as triggering events, including an insert into a table, a user logging in to the database, someone trying to drop a table or change audit settings.

Triggers may call other procedures or functions (an experienced developer usually keeps the trigger's code very short and places anything that requires lengthy code in a separate package).

DBAs use triggers to assist in value-based auditing (discussed in a later lesson), to enforce complex constraints, and to automate many tasks. For example, the `SECURE_EMPLOYEES` trigger that is shown above, logs all DDL statements to a holding table.

There are many different events that can be used to fire a trigger. Most can fire the trigger either before the event is allowed to occur or after (exceptions are noted in the following table). For the DML triggers (`INSERT`, `UPDATE`, `DELETE`), the trigger can be designed to fire once for the statement, or with each row that is modified.

Triggers (continued)

The following events can be used to fire triggers:

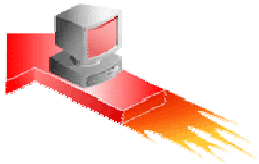
INSERT	UPDATE	DELETE
STARTUP (AFTER)	SHUTDOWN (BEFORE)	SERVERERROR (AFTER)
ALTER	DROP	ANALYZE
ASSOCIATE STATISTICS	AUDIT	NOAUDIT
COMMENT	CREATE	DDL
DISASSOCIATE STATISTICS	GRANT	LOGOFF (BEFORE)
LOGON (AFTER)	RENAME	REVOKE
SUSPEND (AFTER)	TRUNCATE	

PL/SQL Configuration Options

There are several PL/SQL compiler settings that control PL/SQL performance.

For fastest performance set:

- **PLSQL_CODE_TYPE=NATIVE**
- **PLSQL_DEBUG=FALSE**
- **PLSQL_OPTIMIZE_MODE=2**
- **PLSQL_WARNING=DISABLE:ALL**



ORACLE

PL/SQL Configuration Options

A few of the more important initialization parameters controlling PL/SQL performance are:

- PLSQL_CODE_TYPE: Compiled PL/SQL can be stored in two different ways—as interpreted bytecode or as native machine code. The default, interpreted bytecode provides better debugging features for development. Native machine code provides the best run-time performance, up to twice as fast as interpreted bytecode.
- PLSQL_DEBUG: A setting of TRUE enables additional debugging features that are helpful in a development environment and forces code to be stored as interpreted, regardless of the PLSQL_CODE_TYPE setting. The default setting of FALSE turns off extended debugging features.
- PLSQL_OPTIMIZE_LEVEL: The default setting of 1 provides optimum compiler performance. In cases where code changes are infrequent and applications make heavy use of stored PL/SQL, a setting of 2 provides better run-time performance but slightly degrades compiler performance.
- PLSQL_WARNINGS: Accepts two arguments. The first enables or disables warning messages from the PL/SQL compiler. The second applies the first to ALL warning messages, or limits it to SEVERE, INFORMATIONAL, or PERFORMANCE messages. A production setting of DISABLE:ALL provides the best performance.

For details on these parameters, see the *Oracle Database Reference* manual.

Summary

In this lesson you should have learned how to:

- **Identify PL/SQL objects**
- **Describe triggers and triggering events**
- **Identify configuration options that affect PL/SQL performance**

ORACLE

Practice Overview

There is no practice exercise for this lesson.

You will be managing and creating PL/SQL objects several times during the rest of this course.

ORACLE

11

Oracle Database Security

Objectives

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

- **Apply the principal of least privilege**
- **Manage default user accounts**
- **Implement standard password security features**
- **Audit database activity**

ORACLE

Database Security

A secure system ensures the confidentiality of the data it contains. There are several aspects of security:

- **Restricting access to data and services**
- **Authenticating users**
- **Monitoring for suspicious activity**

ORACLE

Database Security

Oracle Database 10*g* provides the industry's best framework for a secure system, but for that framework to be effective the database administrator should follow best practices and continually monitor database activity.

Restricting Access to Data and Services

All users should not have access to all data. Depending on what is stored in your database, restricted access can be mandated by business requirements, customer expectations, and increasingly by legal restrictions. Credit card information, health care data, identity information, and more must be protected from unauthorized access. Oracle provides extremely fined-grained authorization controls to limit database access. Restricting access should include applying the principal of least privilege.

Database Security (continued)

Authenticating User

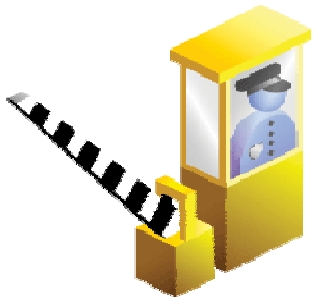
to enforce access controls on sensitive data, the system must first know who is trying to access the data. Compromised authentication can render all other security precautions useless. The most basic form of user authentication is by challenging the user to provide something they know such as a password. Ensuring that passwords follow simple rules can greatly increase the security of your system. Stronger authentication methods include requiring the user to provide something they have, such as a token or Public Key Infrastructure (PKI) certificate. An even stronger form of authentication is to identify the user through a unique biometric characteristic such as a fingerprint, iris scan, bone structure patterns, and so on. Oracle supports advanced authentication techniques such as token-, biometric-, and certificate-based identification through the Advanced Security Option. User accounts that are not in use should be locked to prevent attempts to compromise authentication.

Monitoring for Suspicious Activity

Even authorized, authenticated users can sometimes compromise your system. Identifying unusual database activity such as an employee who suddenly begins querying large amounts of credit card information, research results, or other sensitive information, can be the first step to detecting information theft. Oracle provides a rich set of auditing tools to track user activity and identify suspicious trends.

Apply the Principle of Least Privilege

- Protect the data dictionary
- Revoke unnecessary privileges from `PUBLIC`
- Restrict the directories accessible by users
- Limit users with administrative privileges
- Restrict remote database authentication



ORACLE

Apply the Principle of Least Privilege

Oracle Database 10*g* database server leads the industry in security. However, to fully maximize the security features offered by Oracle Database 10*g* in any business environment, it is imperative that Oracle Database 10*g* itself be well protected and properly configured.

Proper use of embedded security features and adherence to basic security practices will help protect against database-related threats and attacks and provide a much more secure operating environment for the Oracle Database 10*g*.

Practice the Principle of Least Privilege

The principle of least privilege means that a user should be given only those privileges that are required to efficiently complete a task. This reduces the chances that users, either accidentally or maliciously, can modify or view data that they should not have the privilege to modify or view.

Protect the Data Dictionary

- **Protect the data dictionary by ensuring the following initialization parameter is set to FALSE:**

```
O7_DICTIONARY_ACCESSIBILITY = FALSE
```

- **This configuration prevents users with ANY TABLE system privileges from accessing data dictionary base tables.**
- **A FALSE setting also prevents user SYS from logging in as anything other than SYSDBA**
- **The default value of this parameter is FALSE. If you find it set to TRUE, ensure there is a good business reason.**

ORACLE

Protect the Data Dictionary

Nonadministrators do not need access to the data dictionary, but may gain access if you grant one of the * ANY TABLE system privileges such as SELECT ANY TABLE or UPDATE ANY TABLE. The data dictionary contains information that a malicious user can leverage to penetrate or damage your system. To exempt data dictionary tables from the * ANY TABLE privileges, set the O7_DICTIONARY_ACCESSIBILITY initialization parameter to FALSE.

If there are nonadministrator users who *do* need access to the data dictionary, you can grant that access by:

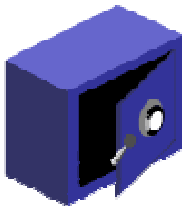
- Using standard GRANT commands to allow the user to see the specific data dictionary objects required
- Granting the SELECT ANY DICTIONARY system privilege to give access to the entire data dictionary

In Oracle Database 10*g* and Oracle9*i* Database, the default value for O7_DICTIONARY_ACCESSIBILITY is FALSE; however, in Oracle8*i* and earlier, it defaults to TRUE; therefore with older version you should manually set it to FALSE to enable data dictionary protection.

Caution: If this parameter is set to TRUE, any user with the DROP ANY TABLE system privilege will be able to maliciously or accidentally drop parts of the data dictionary.

Revoke Unnecessary Privileges from PUBLIC

- Revoke all unnecessary privileges and roles from the database server user group PUBLIC.
- Many built-in packages grant EXECUTE to PUBLIC.
- Execute on the following packages should usually be revoked from PUBLIC:
  - UTL_SMTP
  - UTL_TCP
  - UTL_HTTP
  - UTL_FILE
  - DBMS_OBFUSCATION_TOOLKIT



- Example:

```
SQL> REVOKE execute ON utl_file FROM PUBLIC;
```

ORACLE

Revoke Unnecessary Privileges from PUBLIC

Because any database user can exercise privileges that are granted to PUBLIC, revoke unnecessary privileges and roles from the database server user group PUBLIC.

Such privileges include EXECUTE on several PL/SQL packages that may permit a minimally privileged user to access and execute packages that you may not want them to access. Many of the built-in DBMS_* and UTL_* packages are installed with the EXECUTE privilege granted to PUBLIC. Following the principle of least privilege you should usually revoke these permissions for a few of the more sensitive packages, and then grant individual execute permissions to users who need them. Restricting access to public privileges affects *all* users. You should retest relevant system components. If restricting access is not practical, you may also choose to audit access instead.

The more powerful packages that may potentially be misused include:

- UTL_SMTP: Permits arbitrary mail messages to be sent using the database as an SMTP mail server. Granting this package to PUBLIC may permit unauthorized exchange of mail messages.
- UTL_TCP: Permits outgoing network connections to be established by the database server to any receiving or waiting network service. Thus, arbitrary data can be sent between the database server and any waiting network service.

Revoke Unnecessary Privileges from PUBLIC (continued)

- UTL_HTTP: Allows the database server to request and retrieve data via HTTP. Granting this package to PUBLIC may permit data to be sent via HTML forms to a malicious Web site.
- UTL_FILE: If configured improperly, allows text-level access to any file on the host operating system. Even when properly configured, this package does not distinguish between its calling applications, with the result that one application with access to UTL_FILE may write arbitrary data into the same location that is written to by another application.
- DBMS_OBFUSCATION_TOOLKIT: Encrypts data. Generally, most users should not have the privilege to encrypt data because encrypted data is nonrecoverable if the encryption keys are not securely stored and managed.

These packages are extremely useful to applications that need them, but require proper configuration to be used securely. Thus, unless absolutely necessary, revoke them from PUBLIC and grant them only to individual users or roles when required.

Listing Objects Executable by Public

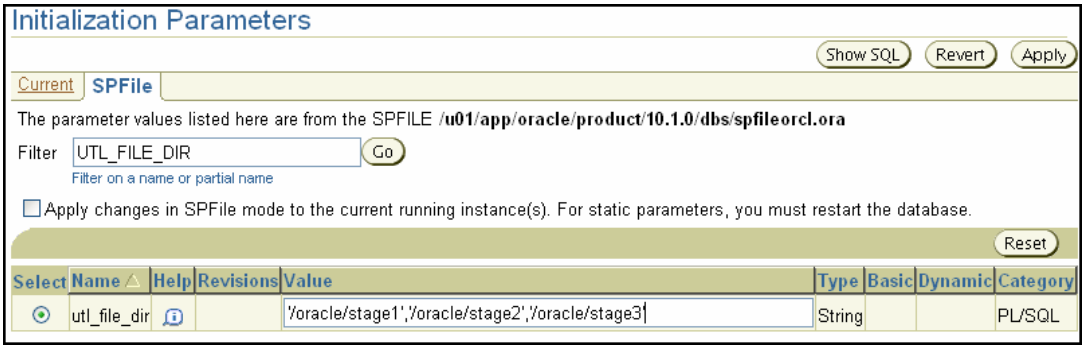
Use the following query to list the objects owned by SYS that have the EXECUTE privilege granted to PUBLIC:

```
SQL> SELECT table_name
  2    FROM dba_tab_privs
  3    WHERE owner='SYS'
  4    AND privilege = 'EXECUTE'
  5    AND grantee='PUBLIC'
  6  /

TABLE_NAME
--------------------
AGGXMLIMP
AGGXMLINPUTTYPE
...
XMLTYPEEXTRA
XMLTYPEPI

437 rows selected.

SQL>
```

Restrict the Operating System Directories Accessible by the User

The `UTL_FILE_DIR` configuration parameter:

- **Designates which directories are available for PL/SQL file I/O**
- **Enables database users to read or write from the listed directories on the database server**



ORACLE

Restrict the Operating System Directories Accessible by the User

The `UTL_FILE_DIR` configuration parameter designates which operating system directories PL/SQL can read from and write to. By default, no directories can be accessed.

Operating system privileges still apply. Directories that the user who started the database could not access are still not accessible regardless of the `UTL_FILE_DIR` setting.

To specify multiple directories, list all required directories in single quotes as a comma-separated list (as shown above). This is not a dynamic parameter so the instance must be restarted for changes to take affect. Remember not to use environment variables in the directory path. The instance does not check to see whether the directories exist, so you can change the parameter and create the directories later.

Because all PL/SQL users can read or write to all files specified by this parameter, all PL/SQL users must be trusted with the information in the directories specified by this parameter.

The directories listed must also be accessible by the Oracle instance.

Caution: Never set `UTL_FILE_DIR = *`, because this enables access to all directories accessible by the Oracle instance, including the data file and redo log directories.

Limit Users with Administrative Privileges

- **Restrict the following types of privileges:**
  - **Grants of system and object privileges**
  - **SYS-privileged connections: SYSDBA and SYSOPER**
  - **DBA-type privileges, such as DROP ANY TABLE**
  - **Run-time permissions**
- **Example: List all users with the DBA role:**

```
SQL> SELECT grantee FROM dba_role_privs
  2   WHERE granted_role = 'DBA';
GRANTEE
------------------------------
SYS
SYSTEM
```


Limit Users with Administrative Privileges

Do not provide database users more privileges than necessary. To implement least privilege, restrict the following types of privileges:

- Grants of system and object privileges
- SYS-privileged connections to the database, such as SYSDBA and SYSOPER
- Other DBA-type privileges, such as DROP ANY TABLE

Use the query shown on the slide to list all users that have the DBA role granted to them. Non-administrators should not be granted the DBA role. Instead, determine the actual privileges they need and grant only those privileges.

To see users who have been granted SYSDBA or SYSOPER privileges (effectively granting them the DBA role), use the following query:

```
SQL> SELECT * FROM V$PWFILE_USERS;
USERNAME                       SYSDBA SYSOPER
------------------------------ ------ -------
SYS                            TRUE   TRUE
```

There is seldom reason for any user other than SYS to have SYSDBA privileges.

Disable Remote Operating System Authentication

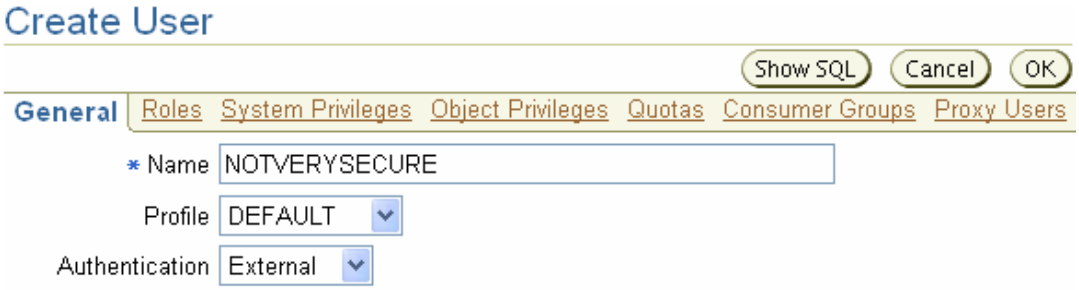
- Remote authentications should be used only when you trust all clients to appropriately authenticate users.
- Remote authentication process:
  - The database user is authenticated externally.
  - The remote system authenticates the user.
  - The user logs on to the database without further authentication.
- To disable, ensure that the following instance initialization parameter is at its default setting:

```
REMOTE_OS_AUTHENT = FALSE
```

ORACLE

Disable Remote Operating System Authentication

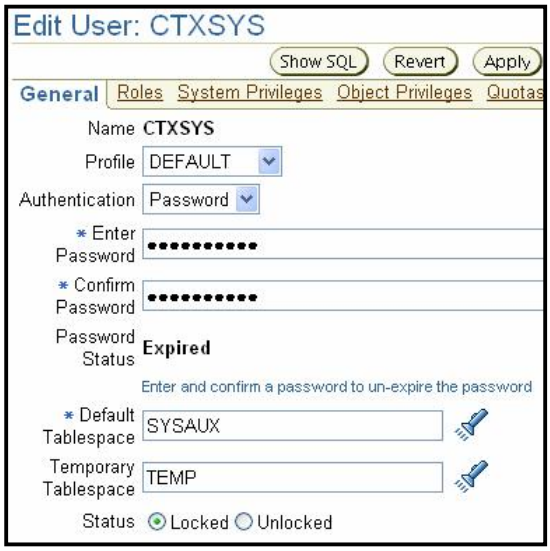
This feature is disabled by default. When enabled (TRUE), external authentication of database users is delegated to the remote system. This means that the instance implicitly trusts that users have authenticated properly on their client PC and does not check authentication credentials again. Users can connect to the instance without providing a password as long as their operating system username matches the database username and the database user is allowed to be authenticated externally.



Most remote operating systems, especially for users connecting from PCs, should not be trusted to perform operating system authentication properly. It is very poor security practice to turn on this feature.

Manage Default User Accounts

- **DBCA expires and locks all accounts, except:**
  - **SYS**
  - **SYSTEM**
  - **SYSMAN**
  - **DBSNMP**
- **For a manually created database, lock and expire any unused accounts.**



ORACLE

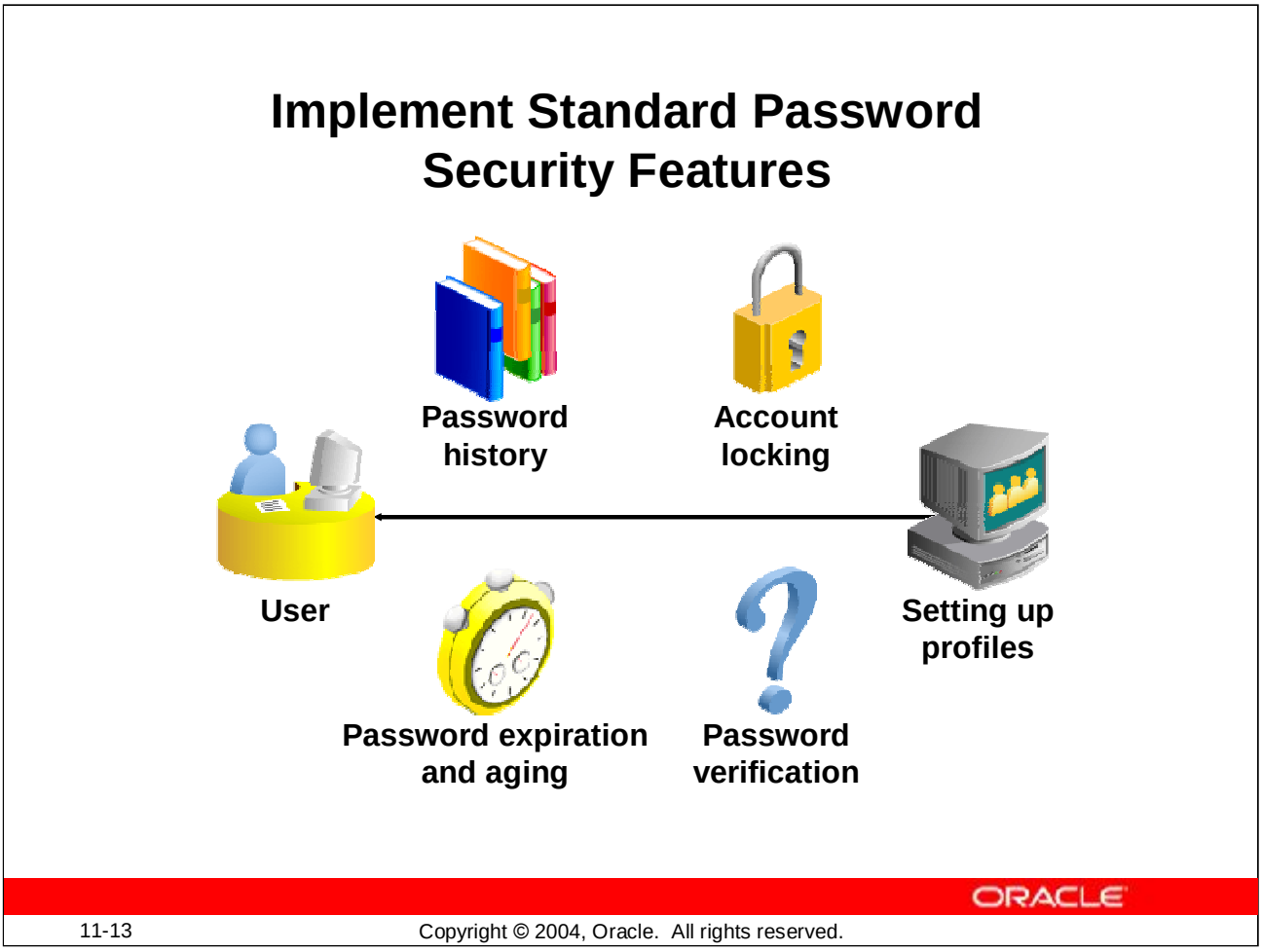
Manage Default User Accounts

Oracle Database 10*g* installs with a number of default, preset database server user accounts. These accounts are intended to store data and own PL/SQL or Java code objects, not to permit database connections. When the Database Creation Assistant (DBCA) is used to create a database it automatically locks and expires all default database user accounts except for the following:

- SYS
- SYSTEM
- DBSNMP
- SYSMAN

Oracle supports scripted creation of custom databases. Many applications expect their database to be configured in a certain way and automate that configuration through the use of scripted database creation. Databases created in this way may not lock the default accounts. Also, many applications create user accounts that should be locked.

Validate that unlocked accounts are really being used for database connections, and not simply to store data.



Implement Standard Password Security Features

Oracle password management is implemented with user profiles.

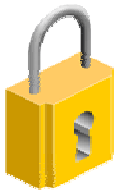
Profiles can provide many standard security features including:

- Account locking: Enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts
- Password aging and expiration: Enables user passwords to have a lifetime, after which they expire and must be changed
- Password history: Checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes
- Password complexity verification: Makes a complexity check on the password to verify that it meets certain rules. The check should ensure the password is complex enough to provide protection against intruders who might try to break into the system by guessing the password.

Remember that when users are created they are assigned the DEFAULT profile unless another profile is implicitly specified.

Password Account Locking

Parameter	Description
FAILED_LOGIN_ATTEMPTS	Number of failed login attempts before lockout of the account
PASSWORD_LOCK_TIME	Number of days the account is locked after the specified number of failed login attempts



ORACLE

Password Account Locking

The Oracle server automatically locks an account after the user fails to log in before the FAILED_LOGIN_ATTEMPTS value is reached. The account is either automatically unlocked after a specified time determined by PASSWORD_LOCK_TIME or it must be unlocked by the database administrator using the ALTER USER command.

The database account can be explicitly locked with the ALTER USER command or by using Enterprise Manager. When this happens, the account is not automatically unlocked after PASSWORD_LOCK_TIME has expired, but must be manually unlocked by the DBA.

```
SQL> ALTER USER hr ACCOUNT LOCK;
User altered.
SQL> CONNECT hr/hr
ERROR: ORA-28000: the account is locked
Warning: You are no longer connected to ORACLE.
SQL> CONNECT / as sysdba
Connected.
SQL> ALTER USER hr ACCOUNT UNLOCK;
User altered.
```

Password Expiration and Aging

Parameter	Description
PASSWORD_LIFE_TIME	Lifetime of the password in days after which the password expires
PASSWORD_GRACE_TIME	Grace period in days for changing the password after the first successful login after the password has expired

ORACLE

Password Expiration and Aging

The database administrator can specify a grace period PASSWORD_GRACE_TIME, which begins after the first attempt to log in to the database after password expiration. A warning message is generated every time the user tries to log in until the grace period is over. If a user does not change their password within the grace period, their account is locked.

Note: If the account is an application account (not accessed through SQL*Plus), verify that the application is able to handle the change password error before enabling password expiration. Many DBAs assign a separate profile to application user accounts and monitoring tools.

A user's account can be manually expired by setting the password to expired.

```
SQL> ALTER USER hr PASSWORD EXPIRE;
User altered.
SQL> CONNECT hr/hr
ERROR: ORA-28001: the password has expired
Changing password for hr
New password: ********
Retype new password: ********
Password changed
```

Password History

Parameter	Description
PASSWORD_REUSE_TIME	Number of days before a password can be reused
PASSWORD_REUSE_MAX	Number of password changes required before the current password can be reused

ORACLE

Password History

Password history checks ensure that a user cannot reuse a password for a specified time interval. These checks can be implemented using one of the following:

- PASSWORD_REUSE_TIME: Specifies that a user cannot reuse a password for a given number of days
- PASSWORD_REUSE_MAX: Specifies the number of password changes required before the current password can be reused.

These two parameters are mutually exclusive, so when one parameter is set to a value other than UNLIMITED (or DEFAULT if the DEFAULT profile has the value set to UNLIMITED), the other parameter must be set to UNLIMITED.

Password Verification

Parameter	Description
PASSWORD_VERIFY_FUNCTION	A PL/SQL function that makes a password complexity check before a password is assigned

Password verification functions must:

- **Be owned by the SYS user**
- **Return a Boolean value (true or false)**

ORACLE

Password Verification

Before assigning a new password to a user, a PL/SQL function can be invoked to verify the validity of the password.

The Oracle server provides a default verification routine that can be loaded by running the SQL script located at `$ORACLE_HOME/rdbms/admin/utlpwdmg.sql` or the database administrator can write a custom PL/SQL function that meets their own security requirements.

In addition to the restrictions listed in the slide, custom password verification functions must use the following specification to declare input variables:

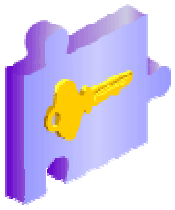
```
function_name(userid_parameter IN VARCHAR2,
              password_parameter IN VARCHAR2,
              old_password_parameter IN VARCHAR2)
RETURN BOOLEAN
```

If the password function raises an exception or becomes invalid, an error message is returned and the `ALTER USER` or `CREATE USER` command is terminated.

Supplied Password Verification Function: VERIFY_FUNCTION

The supplied password verification function enforces password restrictions where the:

- **Minimum length is four characters**
- **Password cannot be equal to username**
- **Password must have at least one alphabetic, one numeric, and one special character**
- **Password must differ from the previous password by at least three letters**



ORACLE

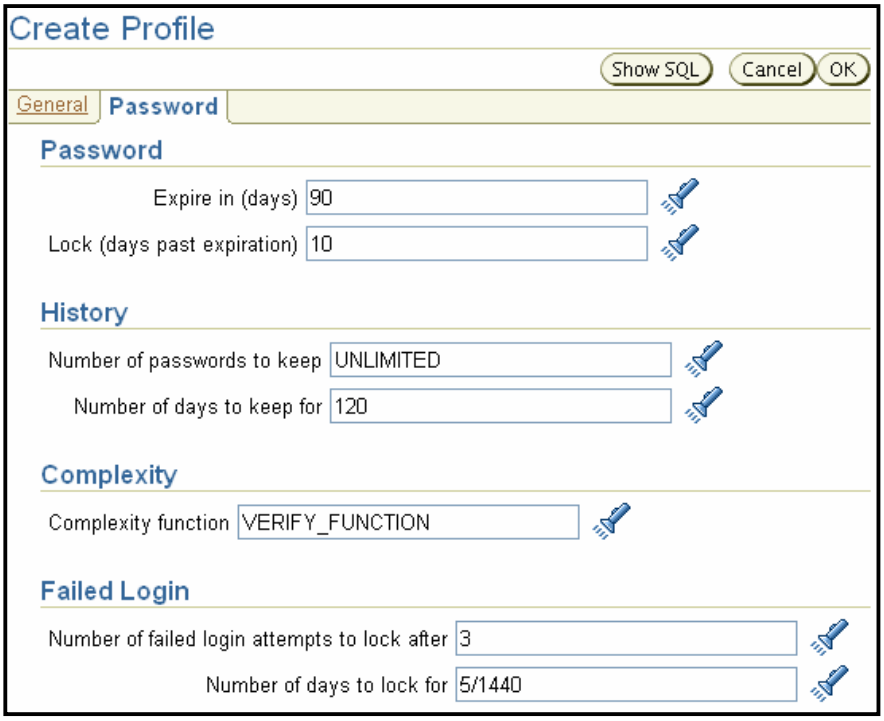
Supplied Password Verification Function: VERIFY_FUNCTION

The Oracle server provides a complexity verification function named VERIFY_FUNCTION. This function is created with the script $ORACLE_HOME/rdbms/admin/utlpwdmg.sql. The password verification function must be created in the SYS schema.

In addition to creating the VERIFY_FUNCTION, the utlpwdmg script also changes the DEFAULT profile with the following ALTER PROFILE command:

```
ALTER PROFILE default LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIFY_FUNCTION verify_function;
```

Creating a Password Profile



Creating a Password Profile

To create a password profile, open Enterprise Manager and navigate to the Administration page. Select Profiles and click the Create button.

Common values for each of the settings can be chosen from a list of values (click the flashlight icon to browse) or you can enter a custom value.

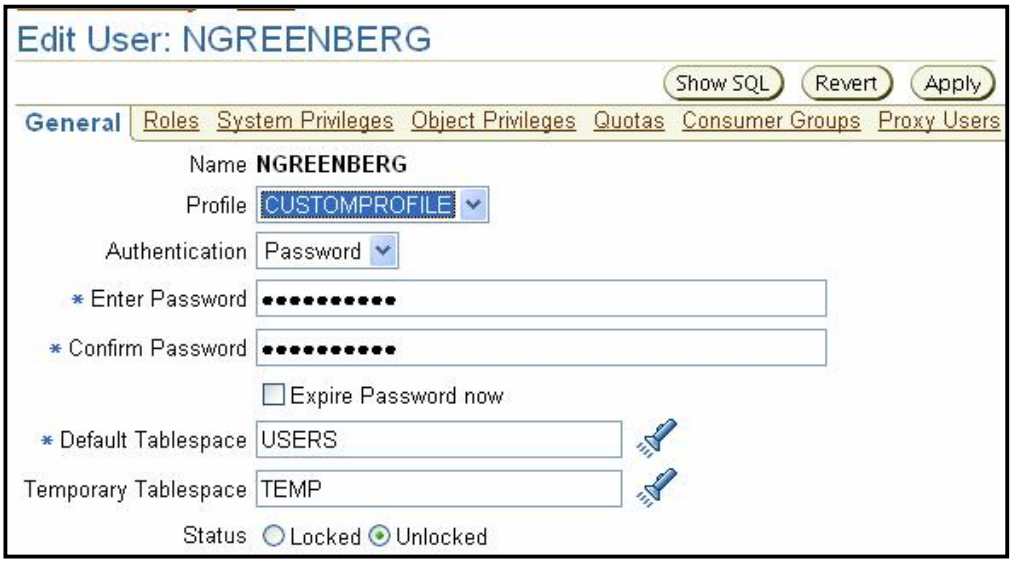
All time periods are expressed in days, but can be expressed as fractions. There are 1440 minutes in a day, so 5/1440 is five minutes.

Enterprise Manager can also be used to edit existing password profiles.

Dropping a Password Profile

If you drop a profile, all users assigned that profile will automatically be assigned to the default profile.

Assigning Users to a Password Profile



ORACLE

Assigning Users to a Password Profile

To assign a user to a password profile:

1. Open Enterprise Manager and navigate to the Administration page.
2. Select Users. Select the user you wish to assign to a profile and click the Edit button.
3. From the Profile drop-down list, select the profile that you want to apply to the user, and click the Apply button.

Remember that a user can have only one profile in effect at any given time. If users have already logged in when you change their profile, the change does not take effect until their next log in.

Note that user accounts can be also be locked or expired from the Edit User page.

Monitoring for Suspicious Activity

Monitoring or auditing should be an integral part of your security procedures.

Oracle's built-in audit tools include:

- **Database auditing**
- **Value-based auditing**
- **Fine-grained auditing (FGA)**

Monitoring for Suspicious Activity

Oracle Database 10*g* provides three different types of auditing. The administrator can audit all actions that take place within the database. It is important to remember that capturing and storing information about what is happening in the system increases the amount of work the system must do. Auditing should be focused so that only events that are of interest are captured. Properly focused auditing has minimal impact on system performance. Improperly focused auditing can significantly affect performance.

Standard database auditing captures several pieces of information about an audited event including that the event occurred, when it occurred, the user who caused the audited event, and which client machine the user was on when the event happened.

Use value-based auditing to audit changes to data (inserts, updates, deletes). Value-based auditing extends standard database auditing, capturing not only that the audited event occurred, but the actual values that were inserted, updated, or deleted. Value-based auditing is implemented through database triggers.

Use fine-grained auditing (FGA) to audit SQL statements. FGA extends standard database auditing, capturing the actual SQL statement that was issued rather than only that the action occurred.

Audit Tool Comparisons

Type of Audit	What Is Audited?	What Is in the Audit Trail?
Standard database auditing	Privilege use including object access	Fixed set of data
Value-based auditing	Data changed by DML statements	Administrator defined
Fine-grained auditing (FGA)	SQL statements (insert, update, delete, and select) based on content	Fixed set of data including the SQL statement

Standard Database Auditing

Enabled through the `AUDIT_TRAIL` parameter

- **`NONE`: Disables collection of audit records**
- **`DB`: Enables auditing with records stored in the database**
- **`OS`: Enables auditing with records stored in the operating system audit trail**

Can audit:

- **Login events**
- **Exercise of system privileges**
- **Exercise of object privileges**
- **Use of SQL statements**

ORACLE

Standard Database Auditing

To use database auditing you must first set the non-dynamic parameter `AUDIT_TRAIL` to point to a storage location for audit records. The normal setting for this parameter is `DB`, which causes audit records to be stored in the `DBA_AUDIT_TRAIL` table.

Database auditing can capture information about login events, the exercise of system privileges, and the exercise of object privileges. Audit information can be focused by the user generating the audit event or by the status of the event (successful or not). The following audit command that generates information is probably not required because it is not well focused. This option captures any operation that affects any table:

```
SQL> AUDIT TABLE;
Audit succeeded.
```

A better example of an audit command (because it is more narrowly focused) is:

```
SQL> AUDIT DELETE ON hr.employees WHENEVER SUCCESSFUL;
Audit succeeded.
```

Specifying Audit Options

- **SQL statement auditing**

```
AUDIT table;
```

- **System privilege auditing (nonfocused and focused)**

```
AUDIT select any table, create any trigger;
AUDIT select any table BY hr BY SESSION;
```

- **Object privilege auditing (nonfocused and focused)**

```
AUDIT ALL on hr.employees;
AUDIT UPDATE,DELETE on hr.employees BY ACCESS;
```

```
AUDIT session whenever not successful;
```

ORACLE

Specifying Audited Options

SQL statement auditing: The statement shown in the slide will audit any DDL statement that affects a table including `CREATE TABLE`, `DROP TABLE`, `TRUNCATE TABLE`, and so on. SQL Statement auditing can be focused by username or by success/failure.

```
SQL> AUDIT TABLE BY hr WHENEVER NOT SUCCESSFUL;
```

System privilege auditing can be used to audit the exercise of any system privilege (such as `DROP ANY TABLE`). It can be focused by username or success/failure. By default, each time an audited system privilege is exercised an audit record is generated. You can choose to group those records so that only one record is generated per session (that way if a user updates 100,000 records in a table belonging to another user, you only gather one audit record). If the `BY SESSION` clause is not specified, the default is `BY ACCESS`. Consider using the `BY SESSION` clause to limit the performance and storage impact of system privilege auditing.

Object privilege auditing can be used to audit actions on tables, views, procedures, sequences, directories and user-defined data types. This type of auditing can be focused by success/failure and grouped by session or access. Unlike system privilege auditing, the default grouping is by session so you must implicitly specify `BY ACCESS` if you want a separate audit trail record generated for each action.

Specifying Audit Options (continued)

The `AUDIT SESSION` option audits the creation of user sessions. It can be focused by username or by success/failure. This option is unique because it generates a single audit record for each session created by connections to an instance. An audit record is inserted into the audit trail at connect time and updated at disconnect time. Cumulative information about a session such as connection time, disconnection time, logical and physical I/Os processed, and more is stored in a single audit record that corresponds to the session. In many databases it is common to use the `AUDIT SESSION` (nonfocused) command. In almost all databases you should `AUDIT SESSION WHENEVER NOT SUCCESSFUL` because this allows you to detect attempts to break into your database

Note: Often your audit options start as nonfocused because you are not sure what type of activity you are looking for. The `AUDIT ALL` option is a convenient shortcut to audit a broad range of activity. If used with object privilege auditing (as shown in the slide), it detects:

Alter	Audit	Comment	Delete
Grant	Index	Insert	Lock
Read	Rename	Select	Update

If the `AUDIT ALL` option is used with a username:

```
SQL> AUDIT ALL BY hr;
```

The user will have all DDL statements audited for the following objects:

Alter System	Cluster	Context	Create Session
Database Link	Dimension	Directory	Index
Materialized View	Not Exists	Procedure	Profile
Public Database Link	Public Synonym	Role	Rollback Segment
Sequence	Synonym	System Audit	System Grant
Table	Tablespace	Trigger	Type
User	View		

Viewing Auditing Options

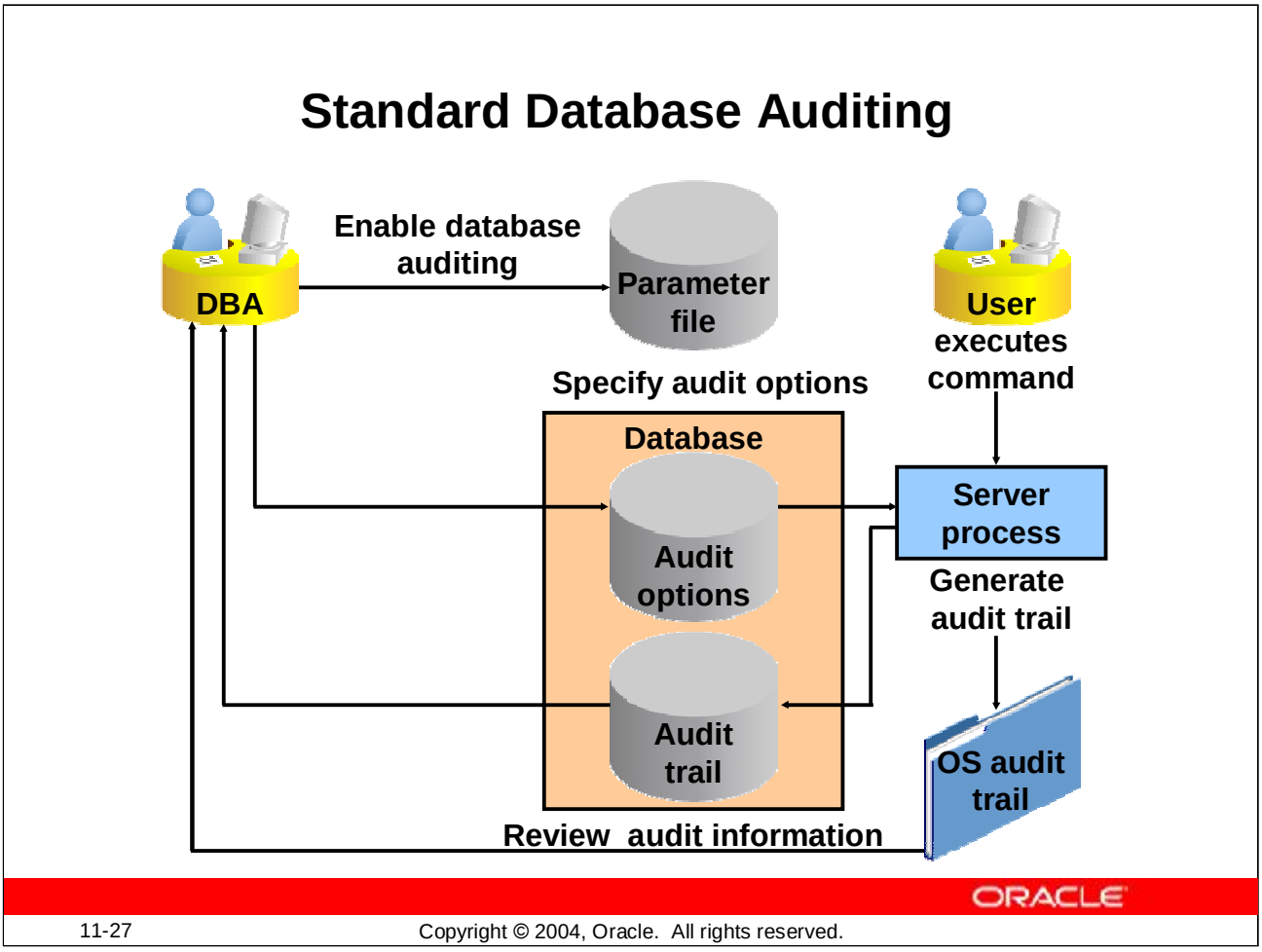
Data Dictionary View	Description
ALL_DEF_AUDIT_OPTS	Default audit options
DBA_STMT_AUDIT_OPTS	Statement auditing options
DBA_PRIV_AUDIT_OPTS	Privilege auditing options
DBA_OBJ_AUDIT_OPTS	Schema object auditing options

Viewing Auditing Options

To see which audit options have been selected, use the views listed above. DBA_STMT_AUDIT_OPTS and DBA_PRIV_AUDIT_OPTS contain only records of statement or privilege audit options that have been specified. DBA_OBJ_AUDIT_OPTS contains one record per auditable object regardless of what object audit options have been specified. The view shows a column for each auditable option. For example, INSERT audit options are shown in the INS column. Audit options are displayed as SUCCESSFUL/NOT SUCCESSFUL with three possible values for each status:

- \- Not audited
- S Collect audit records by session
- A Collect audit records by access

```
SQL> SELECT object_name, object_type, ins, upd
FROM dba_obj_audit_opts WHERE object_name = 'EMPLOYEES'
OBJECT_NAME  OBJECT_TY INS UPD
------------ --------- --- ---
EMPLOYEES    TABLE     A/S -/-
```



Standard Database Auditing

After the administrator has enabled database auditing (with the `AUDIT_TRAIL` parameter) and specified auditing options (with SQL statements as show on the previous few pages), the database begins collecting audit information.

If `AUDIT_TRAIL` is set to `OS`, the audit records will be stored in the operating system's audit system. In a Windows environment, this is the event log. In a UNIX or Linux environment, audit records are stored in a file. The location of that file is specified with `AUDIT_FILE_DEST`. Assuming that `AUDIT_TRAIL` is set to `DB`, audit records are stored in a table that is part of the `SYS` schema.

Maintaining the audit trail is an important administrative task. Depending on the focus of the audit options, the audit trail can grow very large very quickly. If not properly maintained, the audit trail can consume so much space that it affects the performance of the system.

Viewing Auditing Results

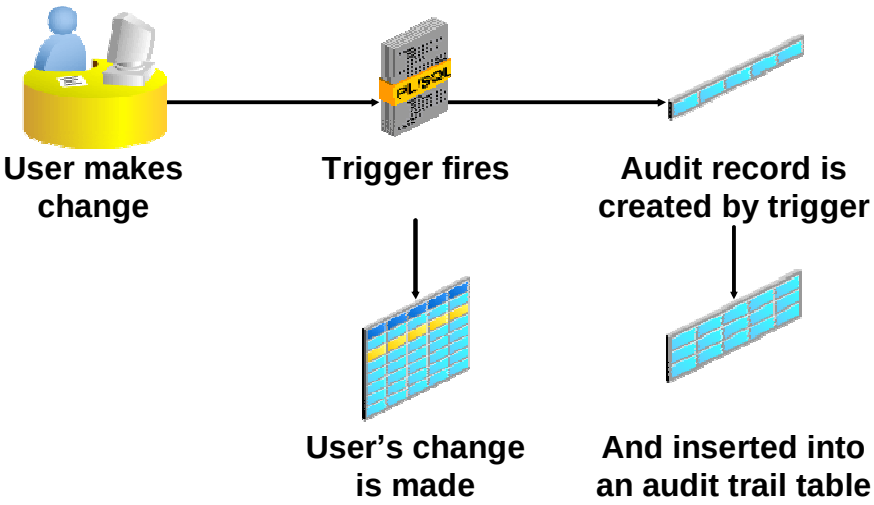
Audit Trail View	Description
DBA_AUDIT_TRAIL	All audit trail entries
DBA_AUDIT_EXISTS	Records for AUDIT EXISTS/NOT EXISTS
DBA_AUDIT_OBJECT	Records concerning schema objects
DBA_AUDIT_SESSION	All connect and disconnect entries
DBA_AUDIT_STATEMENT	Statement auditing records

ORACLE

Viewing Auditing Results

Access to audit records should be tightly controlled because they may contain sensitive information. Usually the task of managing the audit trail is handled by the administrator but if it needs to be delegated the DELETE_CATALOG_ROLE grants permission to delete from the audit trail.

Value-Based Auditing



Value-Based Auditing

Database auditing records that inserts, updates, and deletes have occurred on audited objects, but does not capture the actual values that were changed. Value-based auditing extends database auditing, capturing the actual values that have been changed. Value-based auditing leverages database triggers (event-driven PL/SQL constructs).

When a user inserts, updates, or deletes data from a table with the appropriate trigger attached, the trigger works in the background to copy audit information to a table designed to contain the audit information. Value-based auditing tends to degrade performance more than standard database auditing because the audit trigger code must be executed each time the insert, update, or delete operation occurs. The degree of degradation depends on the efficiency of the trigger code. Value-based auditing should be used only in situations where the information captured by standard database auditing is insufficient.

Value-Based Auditing (continued)

The key to value-based auditing is the audit trigger. A typical audit trigger is show below.

```
CREATE OR REPLACE TRIGGER system.hrsalary_audit
      AFTER UPDATE OF salary
      ON hr.employees
      REFERENCING NEW AS NEW OLD AS OLD
      FOR EACH ROW
  BEGIN
   IF :old.salary != :new.salary THEN
       INSERT INTO system.audit_employees
       VALUES (sys_context('userenv','os_user'), sysdate,
       sys_context('userenv','ip_address'),
       :new.employee_id ||' salary changed from
  '||:old.salary||
       ' to '||:new.salary);
  END IF;
   END;
/
```

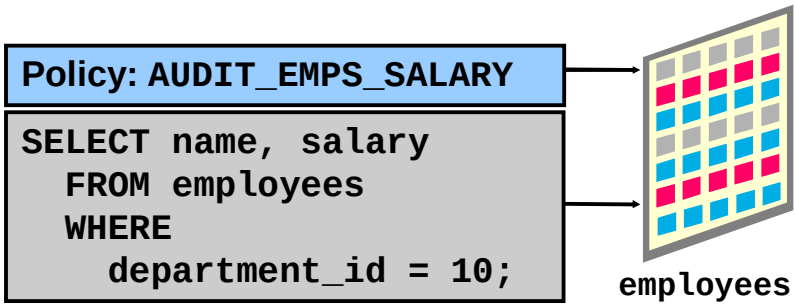
This trigger focuses auditing to capture changes to the salary column of the `hr.employees` table. When a row is updated the trigger checks the salary column. If the old salary is not equal to the new salary, then the trigger inserts an audit record into the `audit_employees` table (created via a separate operation in the `SYSTEM` schema). The audit record will include the username, IP address from which the change was made, primary key identifying which record was changed, and the actual salary values that were changed.

Database triggers can also be used to capture information about user connections in cases where standard database auditing does not gather sufficient data. With logon triggers, the administrator can capture:

- IP address of the person logging in
- The first 48 characters of the program name used to connect to the instance
- Terminal name used to connect to the instance

Fine-Grained Auditing (FGA)

- Monitors data access based on content
- Audits SELECT or INSERT, UPDATE, DELETE
- Can be linked to a table or view
- May fire a procedure
- Is administered with the DBMS_FGA package



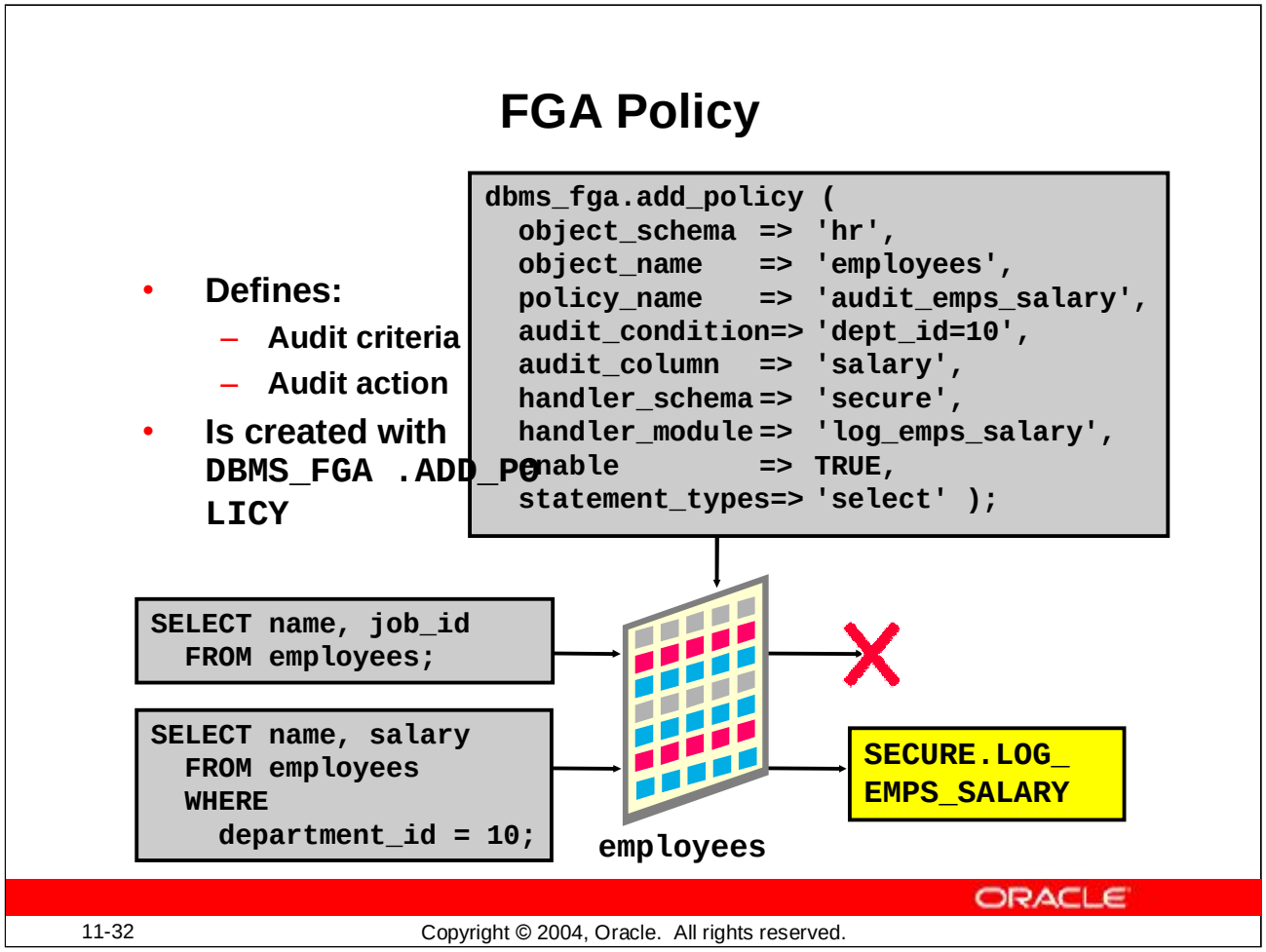
Fine-Grained Auditing (FGA)

Database auditing records that an operation has occurred, but does not capture information about the statement that caused the operation. Fine grained auditing (FGA) extends that capability to allows the capture of actual SQL statements that query or manipulate data. FGA also allows auditing to be more narrowly focused than standard or value-based database auditing.

FGA audit options can be focused by individual columns within a table or view, and can even be conditional so that audits are captured only if certain administrator-defined specifications are met.

Unlike value-based auditing, FGA does not require the use of database triggers and has a performance impact similar to standard database auditing.

The administrator uses the DBMS_FGA PL/SQL package to create an audit policy on the target table or view. If any of the rows returned from a query block matches the audited column and the audit condition, then an audit event causes an audit record to be created and stored in the audit trail. Optionally, the audit event can also execute a procedure. FGA automatically focuses auditing at the statement level so a SELECT statement that returns thousands of rows generates only one audit record.



FGA Policy

The example in the slide shows an FGA policy being created with the `DBMS_FGA.ADD_POLICY` procedure. The procedure accepts the following arguments:

Policy Name

You assign each FGA policy a name when you create it. The example in the slide names the policy `AUDIT_EMPS_SALARY`, using the following argument:

```
policy_name => 'audit_emps_salary'
```

Audit Condition

The audit condition is a SQL predicate that defines when the audit event should fire. In the example on the slide, all rows in the `sales` department are audited, using the following condition argument:

```
audit_condition => 'department_id = 10'
```

Statement Type

Which type of statements should be audited? The choices are `SELECT` and (all one string) `INSERT,UPDATE,DELETE.`

FGA Policy (continued)

Audit Column

The audit column defines the data that is being audited. An audit event occurs only if this column is included in the SELECT statement. The example in the slide audits the SALARY column, using the following argument:

```
audit_column => 'salary'
```

This argument is optional. If it is not specified, then only the AUDIT_CONDITION argument determines whether an audit event occurs.

Object

The object is the table or view that is being audited. It is passed as two arguments:

- The schema that contains the object
- The name of the object

The example on the slide audits the hr.employees table using the following arguments:

```
object_schema => 'hr'
object_name => 'employees'
```

Handler

An optional event handler is a PL/SQL procedure that defines any additional actions that should be taken during auditing. For example, the event handler could send an alert page to the administrator. If it is not defined, then an audit event entry is inserted into the audit trail. If an audit event handler is defined, then the audit entry is inserted into the audit trail and the audit event handler is executed.

The audit event entry includes the FGA policy that caused the event, the user executing the SQL statement, and the SQL statement and its bind variables.

The event handler is passed as two arguments:

- The schema that contains the PL/SQL program unit
- The name of the PL/SQL program unit

The example on the slide executes the SECURE.LOG_EMPS_SALARY procedure using the following arguments:

```
handler_schema => 'secure'
handler_module => 'log_emps_salary'
```

Status

The status indicates whether the FGA policy is enabled. In the example on the slide, the following argument enables the policy:

```
enable => TRUE
```

DBMS_FGA Package

Subprogram	Description
ADD_POLICY	Creates an audit policy using the supplied predicate as the audit condition
DROP_POLICY	Drops an audit policy
ENABLE_POLICY	Enables an audit policy
DISABLE_POLICY	Disables an audit policy

DBMS_FGA Package

The DBMS_FGA package is the administration tool for fine-grained audit functions. Execute privileges on DBMS_FGA are needed to administer fine-grained audit policies. Because the fine-grained audit trail can contain sensitive information, execute privileges on this package should be reserved for administrators only.

Enabling and Disabling an FGA Policy

- **Enable a policy:**

```
dbms_fga.enable_policy (
  object_schema => 'hr',
  object_name   => 'employees',
  policy_name   => 'audit_emps_salary' );
```

- **Disable a policy:**

```
dbms_fga.disable_policy (
  object_schema => 'hr',
  object_name   => 'employees',
  policy_name   => 'audit_emps_salary' );
```

ORACLE

Enabling and Disabling an FGA Policy

Disabling an FGA policy means that the policy does not generate audit events. You can cause the policy to start generating log events again by enabling it. You can cause the policy to start generating log events again by enabling it. By default, the policy is enabled when it is created. The example on the slide shows how to enable and disable a policy. For both procedures, all arguments are required.

The example on the slide show how to enable and disable a policy.

Dropping an FGA Policy

```
SQL> EXEC dbms_fga.drop_policy ( -
>  object_schema => 'hr', -
>  object_name   => 'employees', -
>  policy_name   => 'audit_emps_salary');

PL/SQL procedure successfully completed.

SQL>
```


Dropping an FGA Policy

If you no longer require an FGA policy, you can remove it with `DBMS_FGA.DROP_POLICY`. All arguments are required.

Triggering Audit Events

- **The following SQL statements cause an audit:**

```
SELECT  count(*)
  FROM hr.employees
  WHERE department_id = 10
    AND salary > v_salary;
```

```
SELECT salary
  FROM hr.employees;
```

- **The following statement does *not* cause an audit:**

```
SELECT last_name
  FROM hr.employees
  WHERE department_id = 10;
```

ORACLE

Triggering Audit Events

In general, fine-grained auditing policy is based on columns audited and simple user-defined SQL predicates. During parsing whenever policy conditions are met for a statement, the statement is audited, and if there is an event handler, it is also fired.

The audit function is executed as an autonomous transaction. Each audit policy is applied individually. That is, as long as the rows being returned or changed fit into any of the audit condition defined on the table, an audit record will be generated, and there will be one record generated for each policy per SQL statement.

Examples

The first two examples on the slide cause an audit event because they access the `salary` column and rows with the `department_id = 10`. Oracle server is aware that the second example accesses the rows in department 10, even though the `department_id` is not referenced in the SQL statement `WHERE` clause.

The last example does not cause an audit event because the `salary` column is not accessed. If the `salary` column had not been specified as the `AUDIT_COLUMN` when the policy was created, then this `SELECT` statement would cause an audit event.

Data Dictionary Views

View Name	Description
DBA_FGA_AUDIT_TRAIL	All FGA events
ALL_AUDIT_POLICIES	All FGA policies for objects the current user can access
DBA_AUDIT_POLICIES	All FGA policies in the database
USER_AUDIT_POLICIES	All FGA policies for objects in the current user schema

ORACLE

Data Dictionary Views

FGA audit entries are logged into a separate table from object and privilege audits. The entries are recorded in the DBA_FGA_AUDIT_TRAIL view.

The other three views listed on the slide contain policy definitions.

DBA_FGA_AUDIT_TRAIL

```
SQL> SELECT to_char(timestamp, 'YYMMDDHH24MI')
  2            AS timestamp,
  3     db_user,
  4     policy_name,
  5     sql_bind,
  6     sql_text
  7   FROM dba_fga_audit_trail;

TIMESTAMP  DB_USER POLICY_NAME       SQL_BIND
---------- ------- ---------------- ----------
SQL_TEXT
------------------------------------------------
0201221740 SYSTEM  AUDIT_EMPS_SALARY #1(4):1000
SELECT  count(*)
    FROM hr.employees
    WHERE department_id = 10
      AND salary > :b1
```


DBA_FGA_AUDIT_TRAIL

Name	Description
TIMESTAMP	The date and time of execution
DB_USER	The database name of the user executing
OS_USER	The operating system name of the user executing
OBJECT_SCHEMA	The owner of the audited object
OBJECT_NAME	The name of the audited object
POLICY_NAME	The policy name the caused the audit event
SCN	The SCN of the transaction
SQL_TEXT	The SQL statement causing the audit event
SQL_BIND	The bind variable of the audit event, formatted as: #n(s):v: where *n* is the number of the bind variable in the statement *s* is the bind variable length, *v* is the bind variable value
COMMENT$TEXT	A comment

Note: This is a partial list of columns available in `dba_fga_audit_trail`. For more information refer to the *Oracle Database Reference* guide.

DBA_FGA_AUDIT_TRAIL (continued)

Selecting from the FGA Audit Trail

The following example displays the two audit rows created by the valid examples from the previous page. The `sql_bind` column in the second row has a value of `#1(4):1000`, which includes the following components:

`#1` indicates that this is the first bind variable in the statement.
`(4)` Indicates that the bind variable has a length of 4.
`1000` indicates that the bind variable has a value of 1000.

Example

This example is similar to the one on the slide, except it also includes a row for a policy without an audit event handler.

```
SQL> COL timestamp  FORMAT A10
   SQL> COL db_user   FORMAT A7
   SQL> COL policy_name FORMAT A20
   SQL> COL sql_bind  FORMAT A20
   SQL> COL sql_text  FORMAT A60
   SQL>
   SQL> SELECT to_char(timestamp, 'YYMMDDHH24MI')
     2            AS timestamp,
     3     db_user,
     4     policy_name,
     5     sql_bind,
     6     sql_text
     7   FROM dba_fga_audit_trail;

   TIMESTAMP  DB_USER POLICY_NAME          SQL_BIND
   ---------- ------- -------------------- ------------------
   SQL_TEXT
   ------------------------------------------------------------
   0201221740 SYSTEM  AUDIT_EMPS_SALARY     #1(4):1000
   SELECT  count(*)
       FROM  hr.employees
       WHERE department_id = 10
         AND salary > :b1
   0201221741 SYSTEM  AUDIT_EMPS_SALARY
   SELECT  salary
     FROM  hr.employees

   SQL>
```

FGA Guidelines

- To audit all statements, use a `null` condition.
- If you try to add a policy that already exists, error ORA-28101 is raised.
- The audited table or view must already exist when you create the policy.
- If the audit condition syntax is invalid, an ORA-28112 is raised when the audited object is accessed.
- If the audit column does not exist in the table, no rows are audited.
- If the event handler does not exist, no error is returned and the audit records is still created.

ORACLE

FGA Guidelines

Audit Condition

When creating a new FGA policy, the audit condition defaults to `null`, which means all statements will be audited.

Policy Name Error

Policy names must be unique within the database. They have no owner. If a duplicate name is used, you receive the following error when creating the policy:

```
ORA-28101: policy already exists
```

Audited Object Errors

The audited table or view must exist when you create the policy. If it does not you receive the following error:

```
ORA-00942: table or view does not exist
```

FGA Guidelines (continued)

Audit Condition Errors

If the audit condition has invalid syntax, the policy is created without an error, but the following error is raised when the audited object is accessed:

```
ORA-28112: failed to execute policy function
```

If the audit condition has valid syntax, but is incorrect, then the wrong rows are audited.

Audit Column Errors

If the audit column does not exist in the table, then the policy is created; however, no rows are audited, because the audit column is never accessed.

If the audit column name is valid, but incorrect, then the wrong rows are audited.

Event Handler Error

When the policy references a nonexistent or invalid event handler, the policy is created; however, no rows are returned when an audit event occurs.

Auditing SYSDBA and SYSOPER Users

User's with SYSDBA or SYSOPER privileges can be connecting with the database closed.

- **Audit trail must be stored outside of the database.**
- **Connect as SYSDBA or SYSOPER is always audited.**
- **Enable additional auditing of SYSDBA or SYSOPER actions with `audit_sys_operations`.**
- **Control audit trail with `audit_file_dest`. Default is:**
  - **`$ORACLE_HOME/rdbms/audit` (UNIX/Linux)**
  - **Windows Event Log (Windows)**

ORACLE

Auditing SYSDBA and SYSOPER Users

SYSDBA and SYSOPER users have privileges to start and shut down the database. Because they may make changes while the database is closed, the audit trail for these privileges must be stored outside the database. Oracle captures login events by SYSDBA and SYSOPER users automatically, but does not capture anything other than the login unless auditing is specifically enabled.

Enable auditing of SYSDBA and SYSOPER users by setting the initialization parameter:

`audit_sys_operations=TRUE` (the default is FALSE)

If SYS operations are audited, the `audit_file_dest` initialization parameter controls where the audit records will be stored. On a Windows platform, the audit trail defaults to the Windows Event Log. On UNIX or Linux platforms, audit records are stored in `$ORACLE_HOME/rdbms/audit`.

Security Updates

- **Oracle posts security alerts on the Oracle Technology Network Web site at:**
  http://otn.oracle.com/deploy/security/alerts.htm
- **Oracle database administrators and developers can also subscribe to be notified about critical security alerts via e-mail by clicking the "Subscribe to Security Alerts Here" link.**

Security Updates

Oracle security alerts contain a brief description of the vulnerability, an assessment of the risk and degree of exposure associated with the vulnerability, and applicable workarounds or patches. Oracle also includes an acknowledgement of the individual or organization that notified us of the vulnerability.

Security alerts are posted on the Oracle Technology Network Web site and on Oracle*Meta*Link (*Meta*Link). Although security alerts are publicly acknowledged for anyone interested in them, only customers with a current Customer Support Identification (CSI) number can download patches.

Oracle appreciates your cooperation in keeping its products secure through prompt, complete, and confidential notification of potential security vulnerabilities. If you discover a security vulnerability with any Oracle product, please notify us by submitting a service request through *Meta*Link or by e-mailing secalert_us@oracle.com.

Summary

In this lesson you should have learned how to:

- **Apply the principal of least privilege**
- **Manage default user accounts**
- **Implement standard password security features**
- **Audit database activity**

ORACLE

Practice 11-1 Overview: Database Security (Part 1)

Tasks:

- **Prevent the use of simple passwords**
- **Force accounts to lock for 10 minutes after four failed login attempts**
- **Exempt the application server login from forced password changes**
- **Audit unsuccessful attempts to connect to the database**

ORACLE

Practice 11-1 Overview: Database Security (Part 1)

Background

While conducting user training you discover that some users are using very simple passwords, and suspect that someone may be trying to guess the passwords of privileged users so they can gain access to sensitive data. Configure your database to prevent the use of simple passwords and to detect attempts to guess passwords

Tasks

- Prevent the use of simple passwords
- Force accounts to lock for 10 minutes after four failed login attempts
- Exempt the application server login from forced password changes
- Audit unsuccessful attempts to connect to the database

1. Prevent the use of simple passwords
   a) What profiles exist within the database?
   b) Use Enterprise Manager to see what password restrictions are enforced by the default profile?
   c) Using SQL*Plus, connect to the database as `sysdba` and run the `utlpwdmg.sql` script located in `$ORACLE_HOME/rdbms/admin`

```
SQL> connect / as sysdba
SQL> @?/rdbms/admin/utlpwdmg.sql
Function created.
Profile altered.
```

   d) Using Enterprise Manager, view the changes made to the default profile by the `utlpwdmg.sql` script. Note that:
      - Passwords now expire every 60 days.
      - If a user doesn't change his or her password within 10 days of expiration, the account will be locked.
      - Passwords may not be reused within 1800 days.
      - After a user fails to provide the correct password within three consecutive login attempts the account will automatically lock for one minute.
2. Edit the default profile so that users who fail to log in correctly four times in a row will have their accounts locked for 10 minutes.

3. Exempt the HR user from forced password changes.
    a) Create a new profile called HRPROFILE using the default profile as a template.
    b) Edit the new profile to make password expiration unlimited.
    c) Assign user HR to the new profile.
    d) If you were to drop the HRPROFILE what would happen to the HR user?
        1. Nothing would happen to the HR user. The drop statement would fail because the HRPROFILE cannot be dropped while a user is assigned to it.
        2. The HR user would also be dropped.
        3. The HRPROFILE would be dropped and the HR user would be unable to log in until the administrator assigned a different profile.
        4. The HR user would be automatically assigned the DEFAULT profile.
4. Audit unsuccessful attempts to connect to the database
    a) Enable collection of audit information. Store the audit information in the database.
    b) Begin collecting audit records for users who unsuccessfully attempt to log in.
    c) Verify that unsuccessful attempts to connect to the database are captured.
    d) Why did you have to restart the instance after changing the `AUDIT_TRAIL` initialization parameter?
    e) What would have happened if you had left `AUDIT_TRAIL` at its default setting of `NONE`?

Practice 11-2 Overview: Database Security (Part 2)

Tasks:

- **Audit select on the SALARY column of the EMPLOYEES table**
- **Audit changes to the SALARY column of the EMPLOYEES table, capture:**
  - **Old value**
  - **New value**
  - **User who made the change**
  - **What location the change was made from**

ORACLE

Practice 11-2: Database Security (Part 2)

Background

You suspect that someone has been viewing and possibly changing employee salary data without proper permission. Configure your database to detect unauthorized access to salary data and capture any changes to salary information.

Tasks

- Audit select on the SALARY column of the EMPLOYEES table
- Audit changes to the SALARY column of the EMPLOYEES table. Capture
  - old value - new value
  - which user made the change - what location the change was made from.

1. Audit select on the `salary` column of the `employees` table. Because we only want to capture audit information if someone selects the salary column we must use fine grained auditing rather than standard database auditing.
   Note: "-" is used to continue a statement on a new line in PL/SQL
   a) Use the DBMS_FGA package to add a fine grained audit policy to HR's `employees` table. Only capture audit information if someone reads the `salary` column.
   b) Verify that only `SELECT` statements that include the `salary` column generate an audit trail.
2. Audit changes to the `salary` column of the `employees` table. Because you want to capture the old and new values rather than just the fact that a change happened, you must use value-based auditing rather than standard database auditing. Remember that value-based auditing is implemented through the use of database triggers.
   a) Create a table called `AUDIT_EMPLOYEES` in the `SYSTEM` schema to hold information captured through database auditing. Make the table a standard, heap-organized table with four columns:

```
who        varchar2(10)
event_date date
ipaddress  varchar2(16)
what       varchar2(2000)
```

   b) Create a trigger to capture changes to the `salary` column.

```
Connect as user system
Run the script $HOME/LABS/hrsalarytrig.sql
SQL> connect system/manager@dba10g
Connected.
SQL> @$HOME/LABS/hrsalarytrig.sql
Trigger Created.
```

   c) Verify that audit information about changes to the `salary` column are now captured.

12

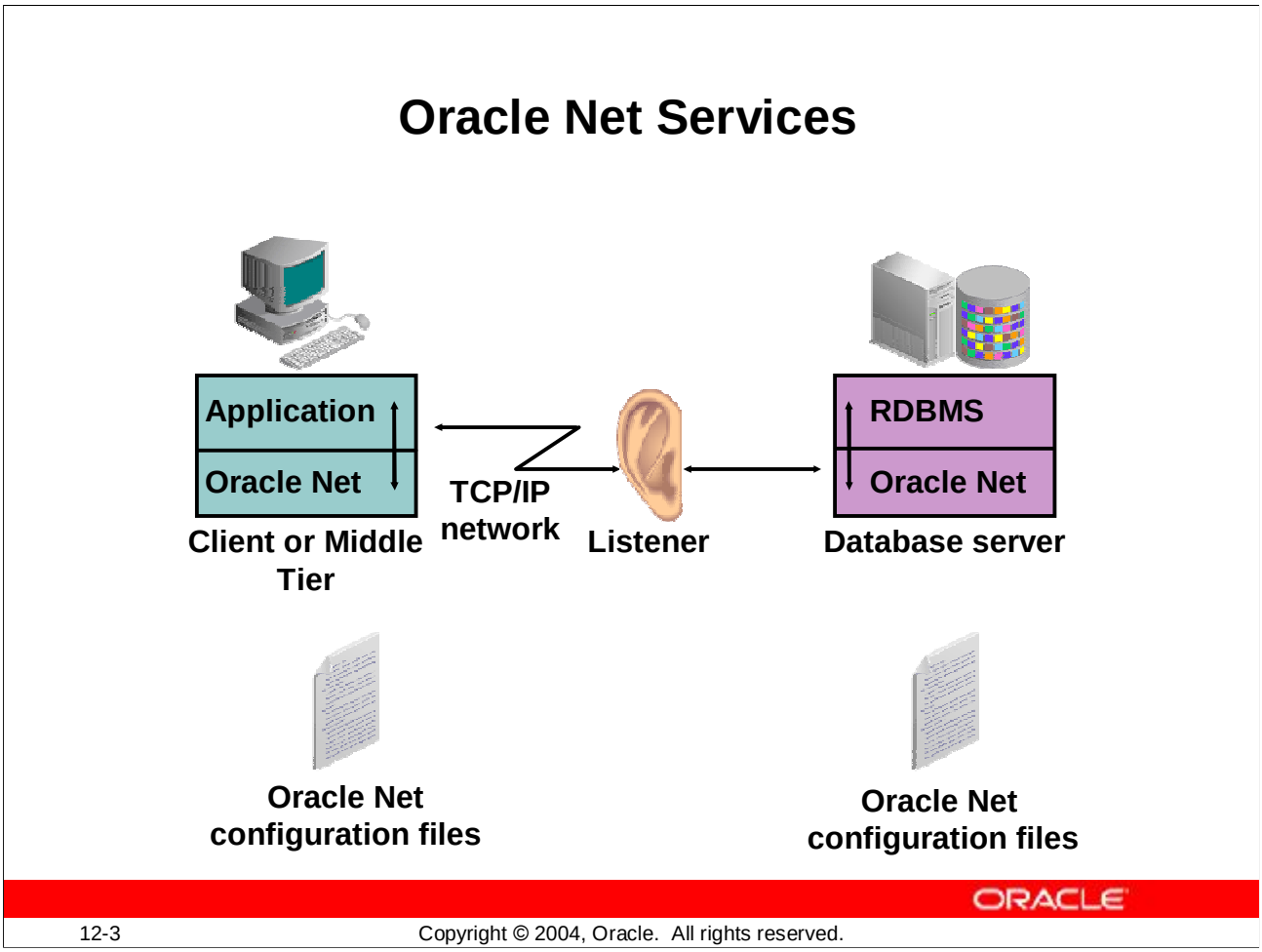
Oracle Net Services

Objectives

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

- Use Database Control to
  - Create additional listeners
  - Create Oracle Net service aliases
  - Configure connect time failover
  - Control the Oracle Net listener
- Use the Oracle Net Manager to configure client and middle-tier connections.
- Use TNSPING to test Oracle Net connectivity

ORACLE



Oracle Net Services

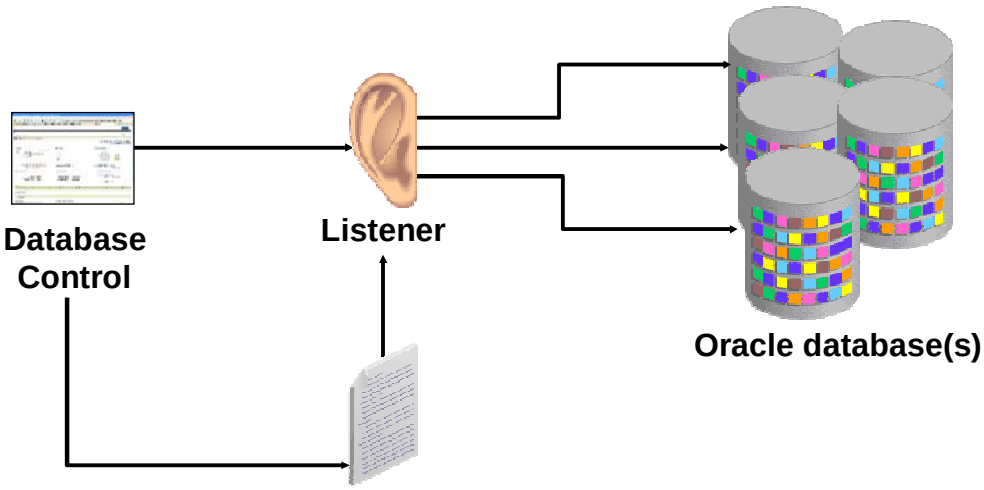
Oracle Net services enable network connections from a client or middle-tier application to the Oracle Database Server. After a network session is established, Oracle Net acts as the data courier for both the client application and the database server. It is responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. Oracle Net, or something that simulates Oracle Net such as Java Database Connectivity (JDBC), is located on each computer that needs to talk to the database server.

On the client machine, Oracle Net is a background component accessed by whatever application needs to connect to the database (SQL*Plus, Oracle Forms, Oracle Discoverer and many others). The user never sees Oracle Net, just the application that is using it.

On the database server, Oracle Net includes an active process called the listener. The Oracle Net listener is responsible for coordinating connections between the database and external applications. Without the listener, external connections to the database are not possible.

While the most common use of Oracle Net is to allow incoming database connections, services also can be configured to allow access to external code libraries and procedures (EXTPROC) and to connect the Oracle instance to non-Oracle data sources such as Sybase, Informix, DB2, and SQL Server through Oracle Heterogeneous services.

Oracle Net Listener



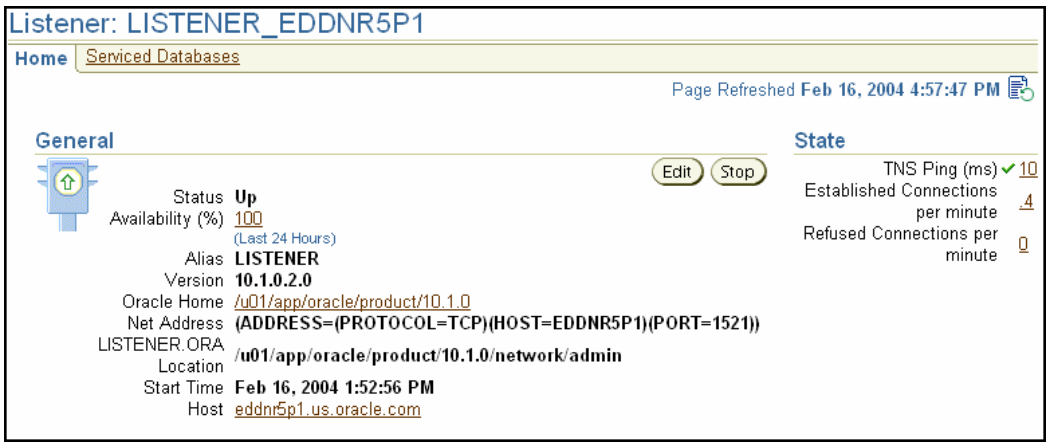
Oracle Net Listener

The Oracle Net listener is the gateway to the Oracle instance for all nonlocal user connections. A single listener can service multiple database instances and thousands of client connections.

The Oracle Net listener is controlled by Database Control or by the `lsnrctl` command-line utility. Listeners are configured using Database Control. For the server, Database Control controls the configuration of the actual listener as well as general parameters such as password protection and log file locations.

Advanced administrators can also configure Oracle Net services by manually editing the configuration files with a text editor such as Vi, gedit, or Notepad if necessary.

Monitoring the Listener



ORACLE

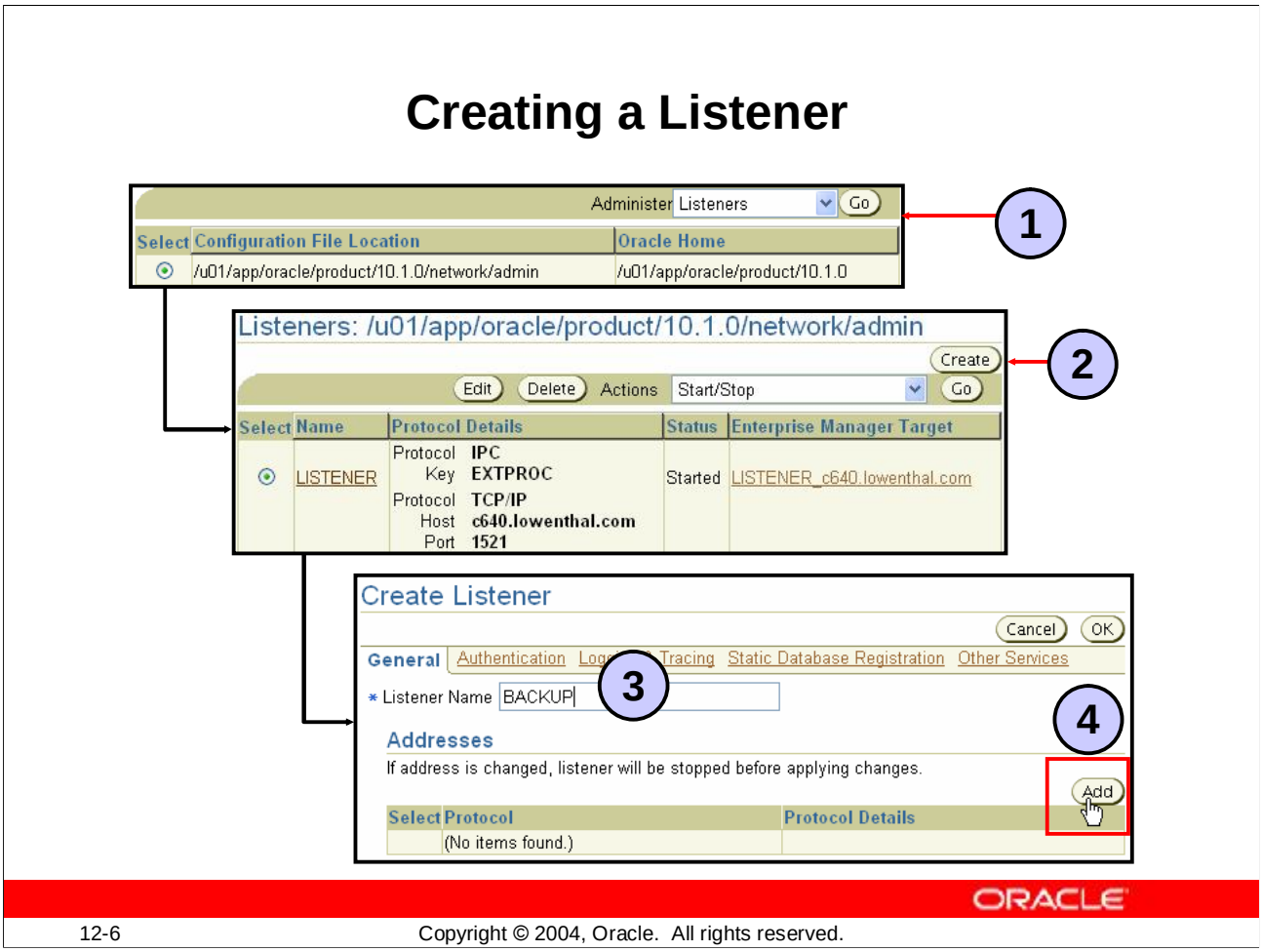
Monitoring the Listener

Selecting the Listener link from the Enterprise Manager home page takes you to the listener status page.

From this page you can see:

- The listener version and ORACLE_HOME
- The first listening address for the listener
- The location of the configuration files used to start the listener
- The availability of the listener and when it was started
- The listener response time: TNS Ping (ms)
- Average number of connections refused/established per minute

Clicking the links for availability, response time, or connections per minute takes you to detailed information about those metrics.



Creating a Listener

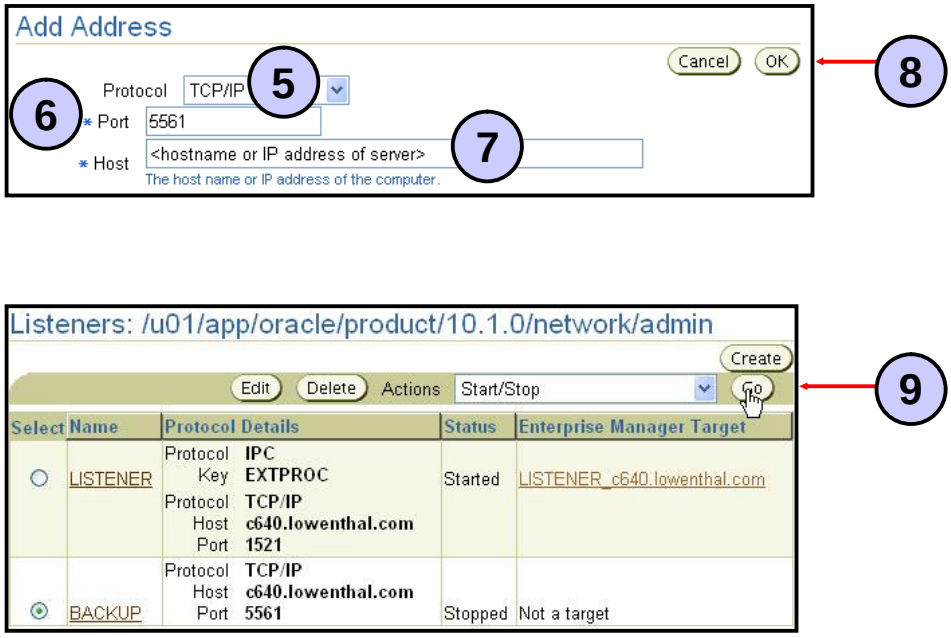
Each listener must have at least one listening location. In the detail pane click the Add Address button to create a listening location. Select the network protocol. TCP/IP is most commonly used and is the default. Other choices are Internal Process Communication (IPC)—normally used for connecting to local applications (resident on the database server)—or external code libraries (EXTPROC), Named Pipes (NMP) and TCP/IP with SSL. Enter the name or IP address of the server the listener will be running on, and the port you want the listener to monitor. Oracle Net's default port is 1521. If you choose to use a port other than 1521, additional configuration is required for the listener or for the instance.

All other configuration steps are optional for the listener. The only required configuration is the listening location. Select File then Save Network Configuration from the menu.

To create an Oracle Net listener, select Net Services Administration from the Related Links section of the listener properties page.

1. Choose Listeners from the Administer drop-down box and click go.
2. Click Create
3. Enter a listener name. The name must be unique for this server.
4. Add a listening address.

Listening Addresses

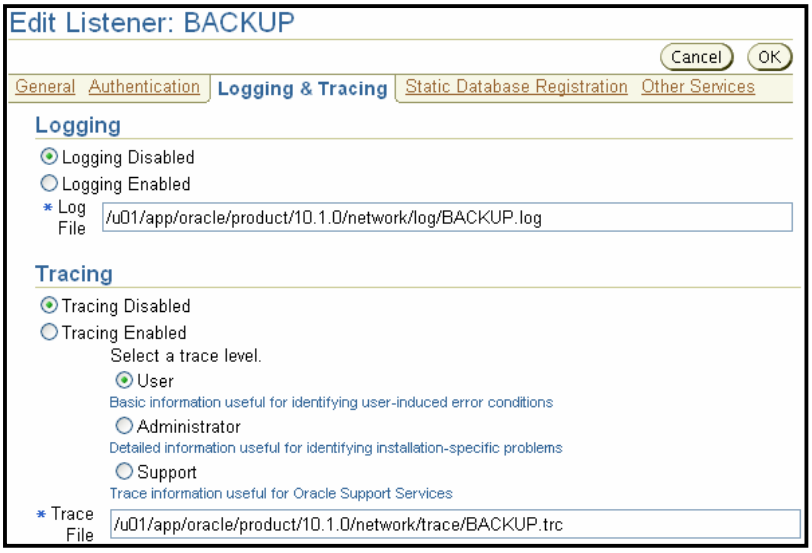


Listening Addresses

Each listener must have at least one listening address.

5. Select the network protocol. TCP/IP is most commonly used and is the default.
   Note: The NMP and EXTPROC protocols are configured using the Other Services tab.
6. Enter the name or IP address of the server the listener will be running on, and the port you wish the listener to monitor. Oracle Net's default port is 1521. If you choose to use a port other than 1521, additional configuration is required for the listener or for the instance.
7. Enter the name or IP address of the server the listener will run on.
8. All other configuration steps are optional for the listener. Click OK to save the address. The only required configuration is the listening address and name. Click OK to save your changes.
9. To start the new listener, select Start/Stop from the Actions drop-down list and click Go.

Configuring Optional Parameters



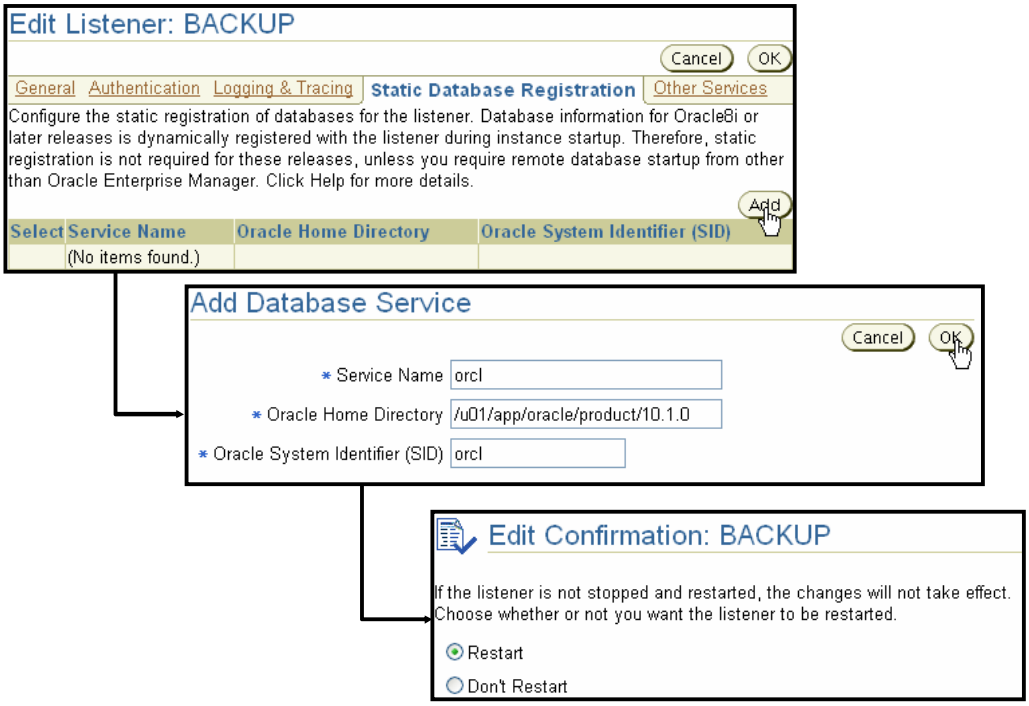
ORACLE

Configuring Optional Parameters

Although only an address is required for a listener, you can also configure optional parameters:

- Logging: Do you want to keep a log of Oracle Net connections and if so, where do you want the log file stored?
- Tracing: Do you want to keep detailed information about each Oracle Net connection? If so, how much information do you want to collect (Trace Level) and where do you want the trace files stored?

Static Database Registration



Static Database Registration

For a listener to forward client connections to an instance, the listener must know the name of the instance and where the instance's ORACLE_HOME is located. The listener can find this information in two ways:

- Dynamic service registration: Oracle8*i*, Oracle9*i*, and Oracle Database 10*g* instances automatically register with the default listener on database startup. No additional listener configuration is required for the default listener.
- Static service registration: Older releases of the Oracle Database do not automatically register with the listener and therefore require that the listener configuration file contain a list of all database services the listener will serve. You may still choose to use static service registration with newer releases if:
  - Your listener is not on the default port of 1521 and you do not wish to configure your instance to register with a nondefault port.
  - Your application requires static service registration.

To add a static database service, select Static Database Registration from the Edit Listener page and click the Add Database button. Enter the service name (same as the global database name *<DB_NAME>.<DB_DOMAIN>*), ORACLE_HOME path, and SID (same as the instance name). Click OK. You will have to restart your listener for the changes to take effect.

Listener Control Utility

Oracle Net listeners can also be controlled with the command-line `lsnrctl` utility.

```
#lsnrctl
LSNRCTL for Linux: Version 10.1.0.1.0 on 05-NOV-2003 13:27:51
Copyright (c) 1991, 2003, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> help
The following operations are available
An asterisk (*) denotes a modifier or extended command:

start               stop                status
services            version             reload
save_config         trace               spawn
change_password     quit                exit
set*                show*
```


Controlling the Listener

The listener control utility enables you to control the listener. With `lsnrctl` you can:

- Start the listener
- Stop the listener
- Check the status of the listener
- Check the status of listener services
- Reinitialize the listener from the configuration file parameters
- Dynamically configure many listener
- Change the listener password

The basic command syntax for this utility is as follows:

```
LSNRCTL> command [listener_name]
```

When a `lsnrctl` command is issued, the command acts on the default listener "listener" unless a different listener name is specified or the `SET LISTENER` command is executed. If the listener name is LISTENER, the *`listener_name`* argument can be omitted.

Valid commands for `lsnrctl` are shown in the slide above.

Listener Control Utility Syntax

Commands from the listener control utility can be issued from the command-line or from the LSNRCTL prompt.

- **UNIX or Linux command-line syntax:**

```
# lsnrctl <command name>
# lsnrctl start
# lsnrctl status
```

- **Prompt syntax:**

```
LSNRCTL> <command name>
LSNRCTL> start
LSNRCTL> status
```

ORACLE

Listener Control Utility Syntax

`lsnrctl` commands can be issued from within the utility (prompt syntax) or the command-line. The following two commands have the identical effect. First, using the command-line syntax:

```
# lsnrctl start
```

Then using the prompt syntax:

```
# lsnrctl
LSNRCTL for Linux: Version 10.1.0.1.0 on 05-NOV-2003
Copyright (c) 1991, 2003, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> start
```

Usually, command-line syntax is used to execute one-off or scripted commands. If you plan to execute several consecutive `lsnrctl` commands, the prompt syntax is most efficient. Notice that the `listener_name` argument was omitted, so the stop command would affect the listener named `LISTENER`. Prompt syntax must be used if your listener is password protected.

Listener Control Utility Syntax (continued)

Remember that if your listener is named something other than LISTENER, you must either include the listener name with the command or use the SET CURRENT_LISTENER command. Suppose your listener was named BACKUP. Using prompt syntax:

```
LSNRCTL> stop backup
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel)(PORT=5521)
))
The command completed successfully
```

The above produces the same results as follows:

```
LSNRCTL> set cur backup
Current Listener is backup
LSNRCTL> stop
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel)(PORT=5521)
))
The command completed successfully
```

Note: In the above syntax, the current_listener can be abbreviated as cur.

You can also achieve the same results with command-line syntax:

```
/home/oracle> lsnrctl stop backup
LSNRCTL for Linux:Version 10.1.0.1.0 on 05-NOV-2003
15:19:33
Copyright (c) 1991, 2003, Oracle. All rights reserved.
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel)(PORT=5521)
))
The command completed successfully
```

Which syntax you use is purely a matter of choice and convenience.

Monitoring with Listener Control

Listener control provides two monitoring options, `services` and `status`.

```
LSNRCTL> SERVICES
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
Services Summary...
Service "dba10g" has 1 instance(s).
  Instance "dba10g", status READY, has 1 handler(s)for
this service...
    Handler(s):
      "DEDICATED" established:12 refused:0 state:ready
…
The command completed successfully
```

ORACLE

Monitoring with Listener Control

The `lsnrctl` services command provides the status of every Oracle Net Service the listener is handling, both static and dynamic. Output includes the:

- Name of the service
- Status of the service (static services will show status of `UNKNOWN`)
- Number of connections established or refused for dedicated and shared connections (shared server connections will be discussed in a later lesson)

Monitoring with Listener Control (continued)

The listener control status command returns detailed information about the listener and summary information about all services the listener is handling.

```
LSNRCTL> SET PASSWORD
Password:
The command completed successfully
LSNRCTL> STATUS
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 10.1.0.1.0 -
   Beta
Start Date                05-NOV-2003 15:48:08
Uptime                    0 days 16 hr. 40 min. 2 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
   /oracle/product/ora10g/network/admin/listener.ora
Listener Log File
   /oracle/product/ora10g/network/log/listener.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=rhel)(PORT=1521)))
Services Summary...
Service "dba10g" has 2 instance(s).
  Instance "dba10g", status READY, has 1 handler(s) for this
   service...
  Instance "dba10g", status UNKNOWN, has 1 handler(s) for this
   service...
Service "rhel" has 1 instance(s).
  Instance "dba10g", status READY, has 1 handler(s) for this
   service...
The command completed successfully
```

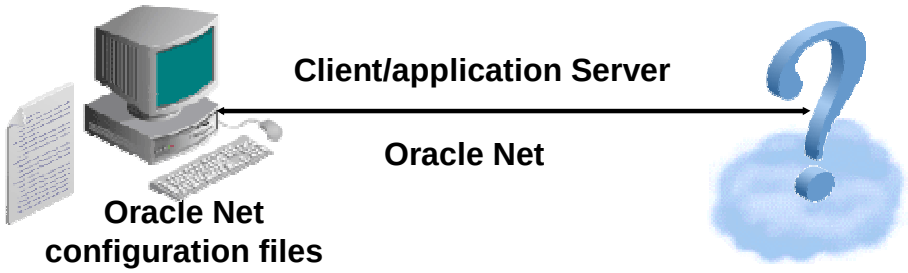
Status information includes:

- Listener name and version
- When the listener was started and how long it has been up
- Location of configuration and log files
- Trace level and security status
- Listener address information including host, port, and protocols serviced
- Summary information about each service the listener is handling. Status information for statically registered services will display as `UNKNOWN`. As you can see above, an instance can have multiple service names (instance `dba10g` is also answering to `rhel`)

Oracle Net Connections

To make a client or middle-tier connection, Oracle Net requires the client to know the:

- **Host where the listener is running**
- **Port the listener is monitoring**
- **Protocol the listener is using**
- **Name of the service the listener is handling**



ORACLE

Oracle Net Connections

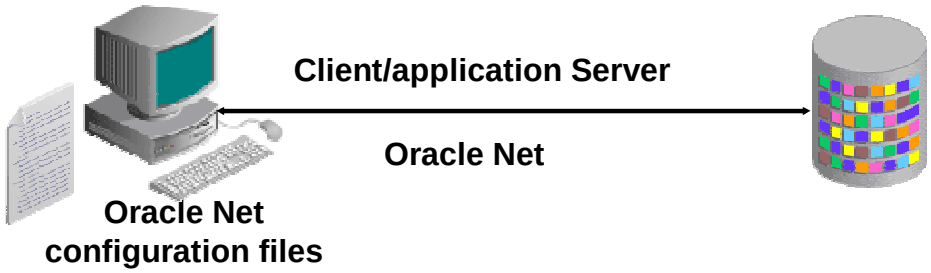
In order for an application to connect to a service through an Oracle Net listener, the application must know information about that service, including the address or host where the listener resides, the protocol the listener will accept, and which port the listener is monitoring. After the listener is located, the final piece of information the application needs is the name of the service it wants to connect to.

The process of determining this connection information is called Names Resolution.

Names Resolution

Oracle Net supports several methods of resolving connection information:

- **Easy Connect**
- **Local naming**
- **Directory naming**
- **External naming**



ORACLE

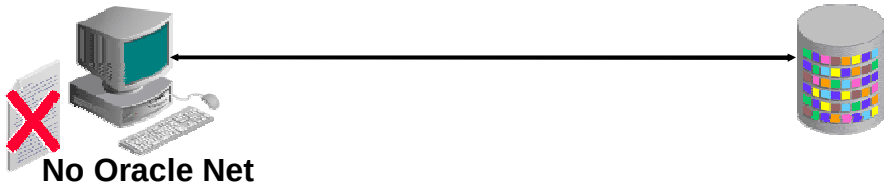
Names Resolution

Easy Connect, local naming, and directory naming are discussed in the following pages. External naming uses third-party naming services such as Network Information Services (NIS), Distributed Computing Environment (DCE), or Cell Directory Services (CDS). Conceptually, external naming is very similar to directory naming.

Easy Connect

- Enabled by default
- Requires no client-side configuration
- Supports only TCP/IP protocol (no SSL)
- No support for advanced connection options like:
  - Connect-time failover
  - Source routing
  - Load balancing

```
SQL> CONNECT hr/hr@db.us.oracle.com:1521/dba10g
```



No Oracle Net configuration files

ORACLE

Easy Connect

With Easy Connect the user supplies all information required for the Oracle Net connection as part of the connect string. Easy Connect connection strings take the form of:

```
<username>/<password>@<hostname>:<listener port>/<service
   name>
```

The listener port and service name are optional. If the listener port is not provided, then Oracle Net assumes the default port of 1521 is being used. If the service name is not provided Oracle Net assumes the database service name and host name provided in the connect string are identical.

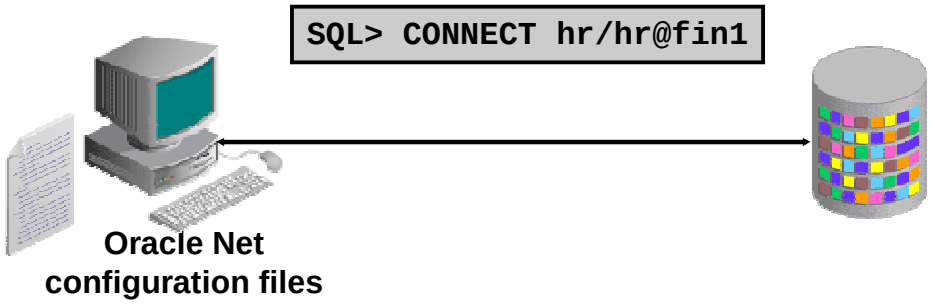
Assuming the listener was using the TCP protocol to listen on port 1521 and instance parameters `SERVICE_NAMES=db`, `DB_DOMAIN=us.oracle.com`, the connect string shown in the slide above could be shortened to:

```
SQL> connect hr/hr@db.us.oracle.com
```

Note: The `SERVICE_NAMES` initialization parameter can accept multiple comma-separated values. Only one of those values needs to be `db` for this scenario to work.

Local Naming

- Requires a client-side names resolution file
- Supports all Oracle Net protocols
- Supports advanced connection options like:
  - Connect-time failover
  - Source routing
  - Load balancing



ORACLE

Local Naming

With local naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against a local list of known services and if it finds a match, converts the alias into host, protocol, port, and service name.

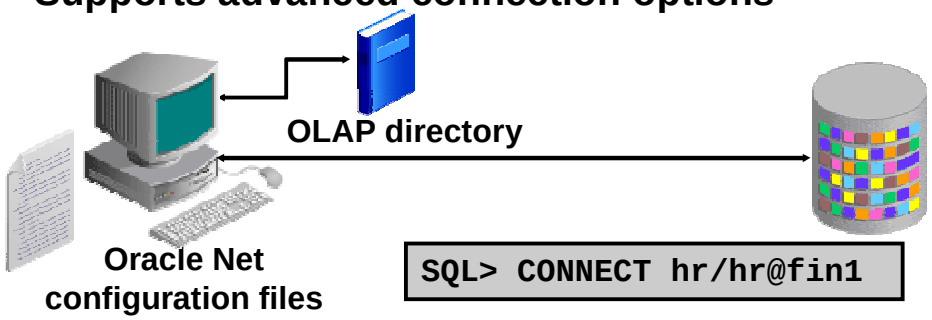
One advantage of local naming is that the database users only need to remember a short alias rather than the long connect string required by Easy Connect.

The local list of known services is stored in in the text configuration file `$ORACLE_HOME/network/admin/tnsnames.ora`.

Local naming is appropriate for organizations where Oracle Net service configurations do not change often.

Directory Naming

- Requires a Lightweight Directory Access Protocol (LDAP) with Oracle Net names resolution information loaded
  - Oracle Internet Directory
  - Microsoft Active Directory Services
- Supports all Oracle Net protocols
- Supports advanced connection options



ORACLE

Directory Naming

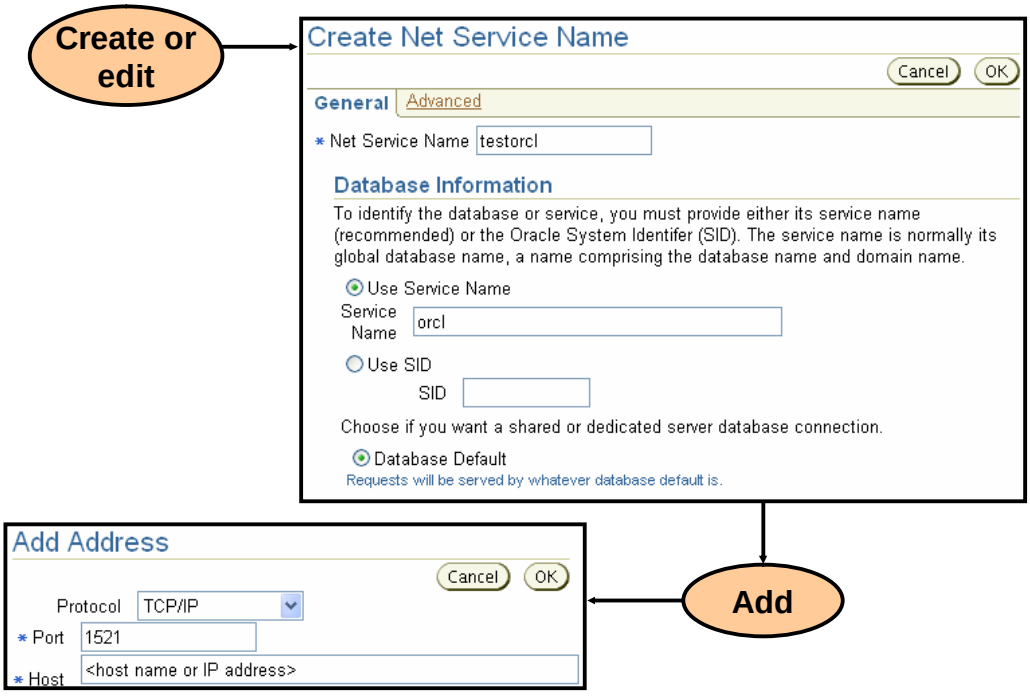
With directory naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against an external list of known services and if it finds a match, converts the alias into host, protocol, port, and service name.

Like local naming, database users only need to remember a short alias rather than the long connect string required by Easy Connect.

One advantage of directory naming is that as soon as a new service name is added to the LDAP directory, the service name is available for users to connect with. With local naming the DBA must first distribute updated `tnsnames.ora` files containing the changed service name information before users can connect to new or modified services.

Directory naming is appropriate for organizations where Oracle Net service configurations change frequently.

Configuring Service Aliases



Configuring Service Aliases

To create a local Oracle Net service alias, select Local Naming from the Administer drop-down list and click Go, and then click Create.

To configure service aliases for directory naming, select Directory Naming instead of Local Naming.

Note: If directory naming has not already been configured, then you cannot select the Directory Naming option. Directory Naming is discussed in the *Oracle Enterprise Identity Management* course as well as the *Oracle Advanced Security Administration* manual.

In the Create Net Service Name page, enter a unique name in the Net Service Name field (this is the name users will enter when they want to use this alias). Enter the service name or SID of the database you want to connect to, and click the Add button to enter the address for the service name.

For the address, enter the protocol, port, and host used by the listener for the service you want to connect to.

Advanced Connection Options

Oracle Net supports advanced connection options with local and directory naming

- **Connect-time failover**
- **Load balancing**
- **Source routing**

ORACLE

Advanced Connection Options

With advanced connection options, Oracle Net can take advantage of listener failover and load balancing, as well as Oracle Connection Manager source routing.

With connect-time failover enabled, the alias has two or more listener addresses listed. If the first address is not available, the second is then tried. Oracle Net will keep trying addresses in order until it reaches a listener that is functioning or until all addresses have been tried and failed.

With load balancing enabled, Oracle Net picks an address at random from the list of addresses. In a Real Application Cluster this balances the workload across multiple instances of the same database.

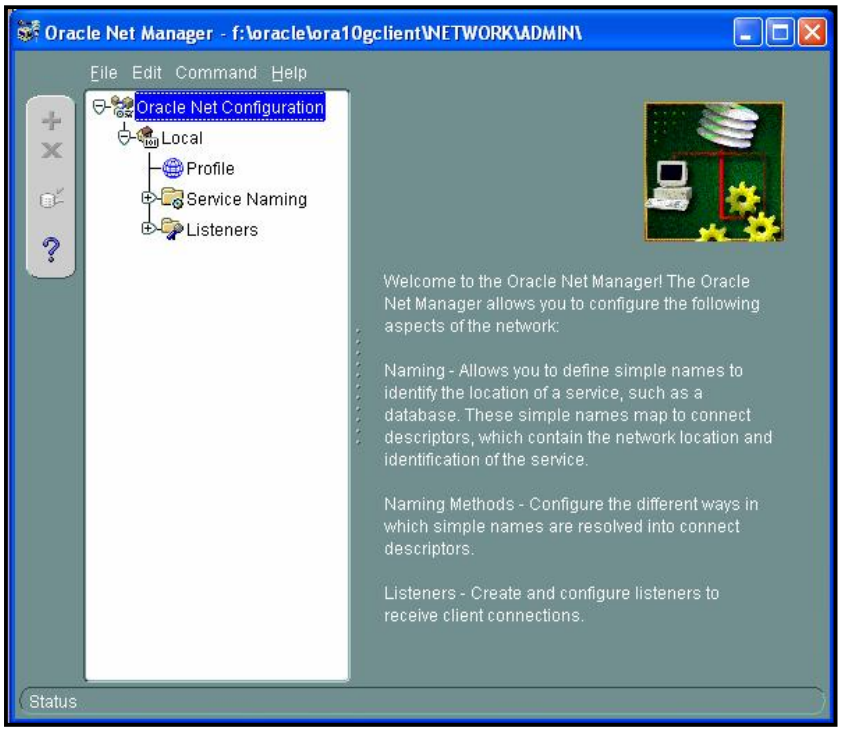
Source routing is used with Oracle Connection Manager. Oracle Connection Manager serves as a proxy server for Oracle Net traffic, enabling Oracle Net traffic to be routed securely through a firewall. Oracle Net treats the addresses as a list of relays, connecting to the first address and then requesting to be passed from the first to the second until the destination is reached. It differs from failover or load balancing in that all addresses are used each time a connection is made. For more information on Oracle Connection Manager please refer to Chapter 11 of the *Oracle Net Services Administrator* guide.

Advanced Connection Options (continued)

Note that there are five choices for connect-time failover and load balancing. The five options translate to:

Option	Advanced Functionality
Try each address, in order, until one succeeds	Failover
Try each address, randomly, until one succeeds	Failover Load balancing
Try one address, selected at random	Load balancing
Use each address in order until destination reached	Source routing
Use only the first address	None

Oracle Net Manager



ORACLE

Oracle Net Manager

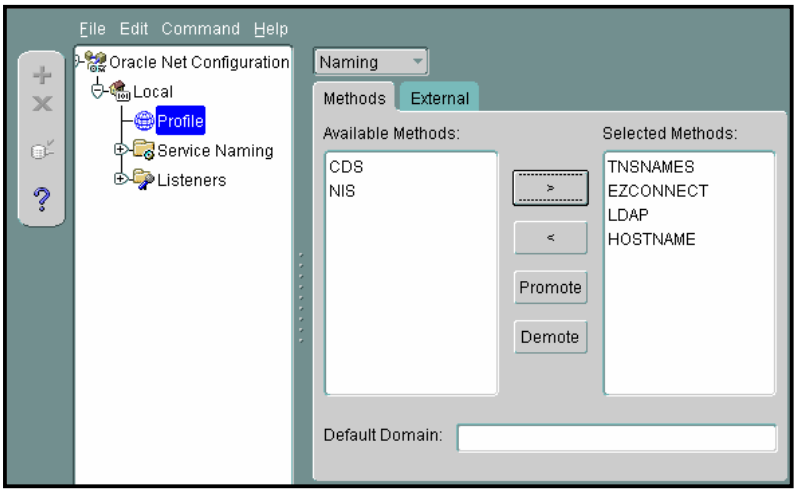
Enterprise Manager Database Control configures the server side of Oracle Net, but does not help with client or middle-tier configuration. The Oracle Net Manager is a graphical tool primarily used to configure client-side Oracle Net connections including:

- Oracle Net service names
- Oracle Net profiles
  - Security
  - Troubleshooting (logging and tracing)
  - Names resolution methods

The Oracle Net Manager can also be used to create and configure Oracle Net listeners, and to configure Lightweight Directory Access Protocol (LDAP) service name resolution.

To start Oracle Net Manager, run `netmgr` from a command prompt. When you open Oracle Net Manager, you will see a navigation tree on the left, and a detail window on the right. You can drill down into each node in the tree structure using the expand/contract symbol (+/-) next to the node you want to configure.

Choosing Naming Methods



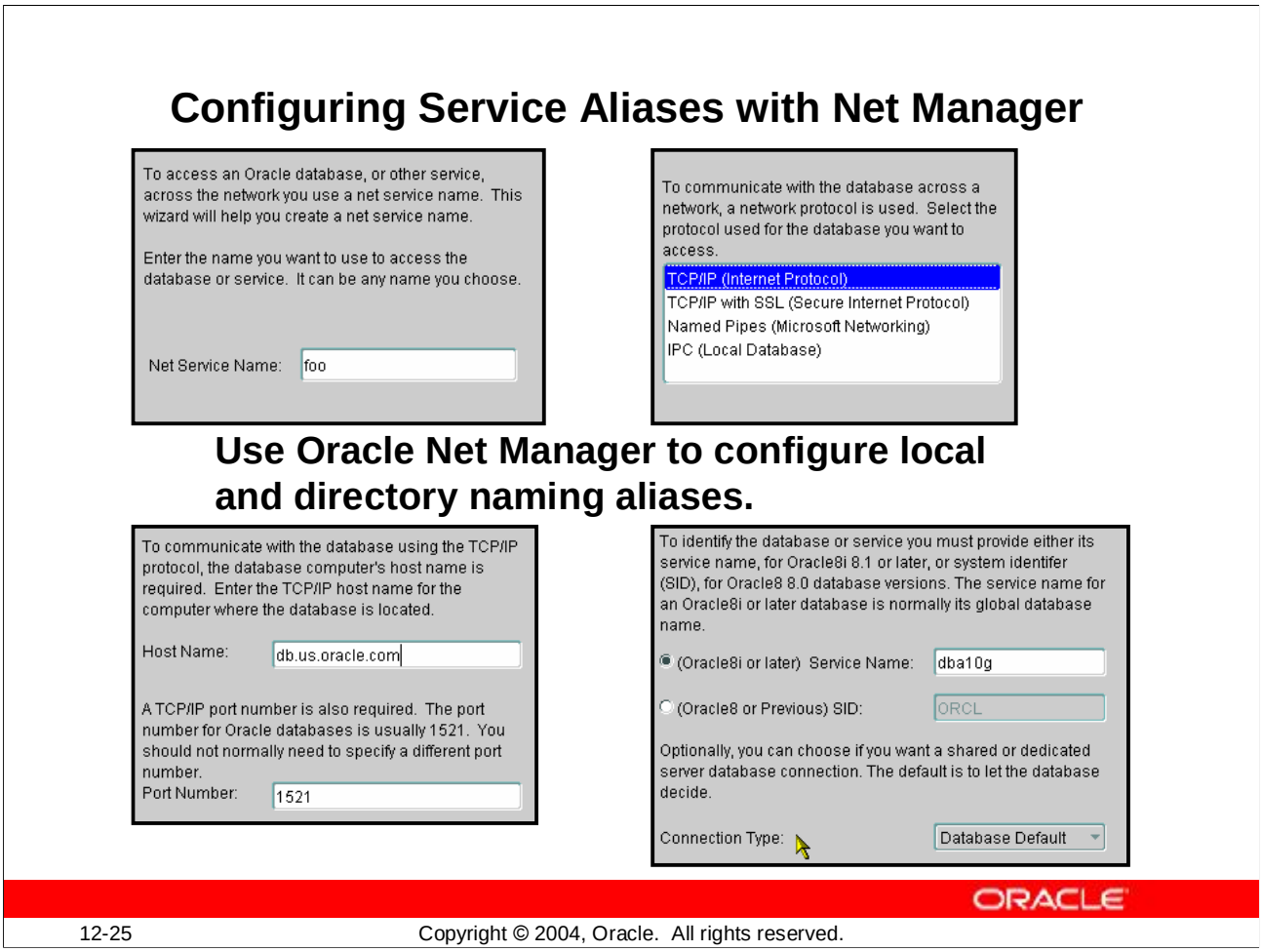
Oracle Net Manager configures the names resolution methods a client or middle-tier will use and the order in which they are checked.

ORACLE

Choosing Naming Methods

The default names resolution methods are local naming (TNSNAMES), Easy Connect (EZCONNECT) and host naming (a deprecated predecessor to Easy Connect).

With the configuration shown above, Oracle Net first examines the local names resolution file, attempts to use Easy Connect, checks the directory server,and finally attempts to use host naming until it either finds a match for the user-supplied connect string or fails. If you want directory naming to be checked before Easy Connect, then select LDAP and click the Promote button.



Configuring Service Aliases with Net Manager

You can also use Oracle Net Manager to add Oracle Net Service aliases to the local naming or directory naming repository. For local naming, expand the Local node and select Service Names from the tree structure. Then click the plus (+) icon (upper left) or use the menu by selecting Edit and then Create. This opens a configuration wizard where you first choose the alias name, then the protocol, then the address and port, and finally the actual service name that the listener is handling.

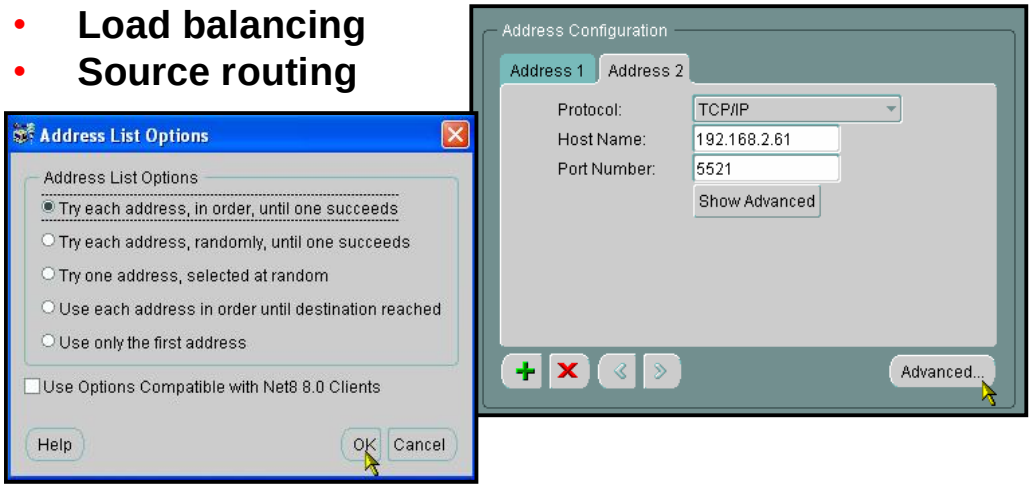
To configure service aliases for directory naming, expand the Directory node rather than the Local node.

After your alias has been created you can test the connection. Unlike Database Control, Oracle Net Manager attempts to connect to the service with the username `SCOTT` using a password of `TIGER`. You must click the Change Login button and supply the correct username and password.

Advanced Connection Options Using Oracle Net Manager

Oracle Net supports advanced connection options with local and directory naming

- **Connect-time failover**
- **Load balancing**
- **Source routing**



ORACLE

Advanced Connection Options Using Oracle Net Manager

Like Database Control, Oracle Net Manager allows you to select advanced connection options such as listener failover and load balancing, as well as Oracle Connection Manager source routing.

Option	Advanced Functionality
Try each address, in order, until one succeeds	Failover
Try each address, randomly, until one succeeds	Failover Load balancing
Try one address, selected at random	Load balancing
Use each address in order until destination reached	Source routing
Use only the first address	None

Testing Oracle Net Connectivity

The `tnsping` utility tests Oracle Net service aliases.

- **Ensures connectivity between client and the Oracle Net listener**
- **Does not verify that the requested service is available**
- **Supports Easy Connect names resolution**

```
# tnsping db.us.oracle.com:1521/dba10g
```

- **Supports local and directory naming**

```
# tnsping foo
```

ORACLE

Testing Oracle Net Connectivity

`tnsping` is the Oracle Net equivalent of the TCP/IP ping utility. It offers a quick test to verify that the network path (including port and protocol) to a destination is good.

```
# tnsping db.us.oracle.com:1521/dba10g
TNS Ping Utility for Linux: Version 10.1.0.1.0 - Production on 06-NOV-20
Copyright (c) 1997, 2003, Oracle. All rights reserved.
Used parameter files:
/oracle/prodouct/ora10g/network/admin/sqlnet.ora
Used EZCONNECT adapter to resolve the alias
Attempting to contact (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=dba10g))
                (ADDRESS=(PROTOCOL=TCP)(HOST=192.168.2.61)(PORT=1521)))
OK (380 msec)
```

The utility validates that the host name, port, and protocol reach a listener. It does not actually check that the listener handles the service name. One other useful thing that `tnsping` reveals is the location of the configuration files. In a system with multiple `ORACLE_HOME` locations, this can be helpful.

Summary

In this lesson you should have learned how to:

- Use Database Control to:
  - Create additional listeners
  - Password-protect the listener
  - Create Oracle Net service aliases
- Control the Oracle Net listener
- Use the Oracle Net Manager to configure client or middle-tier connections
- Use `TNSPING` to test Oracle Net connectivity

ORACLE

Practice 12 Overview: Oracle Net Services

This practice covers the following topics:

- **Creating a listener**
- **Configuring local names resolution**
- **Configuring connect time failover**

ORACLE

Practice 12: Oracle Net Services

Background: Users need to connect to your database. Work with them to allow connections using several different methods. Ensure that users can use connect-time failover to take advantage of a backup listener.

Tasks:

- Create a backup listener to support connect-time failover
- Modify your local name resolution file so you can connect to another database
- Modify your local name resolution file so that users fail over to the backup listener if the primary is unavailable

1. Create a backup listener to support connect-time failover.
   - Name the new listener BACKUP.
   - Use TCP/IP protocol on port 1561.
   - Use static registration to register your ORCL database with the new listener.
   - Start the new listener.
2. Modify your local name resolution file so that you can connect to another database. Test your changes to the network configuration using SQL*Plus or *i*SQL*Plus.
3. Modify your local name resolution file so that you fail over to the backup listener if the primary listener is unavailable. Test your changes by shutting down the default listener and verifying that you can now connect through the backup listener.

13

Oracle Shared Servers

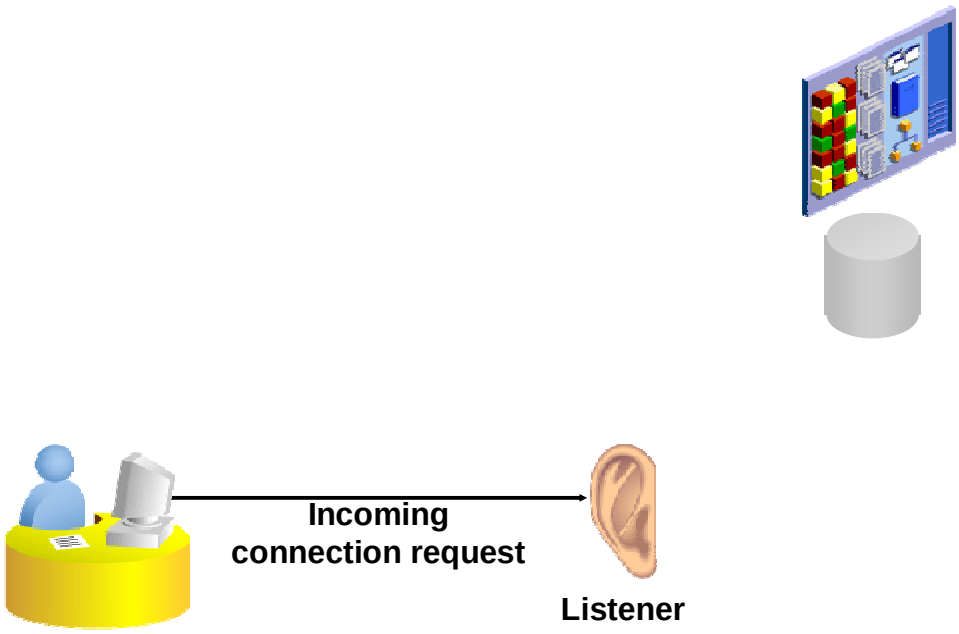
Objectives

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

- **Identify when to use Oracle Shared Servers**
- **Configure Oracle Shared Servers**
- **Monitor Shared Servers**

ORACLE

Establishing a Connection

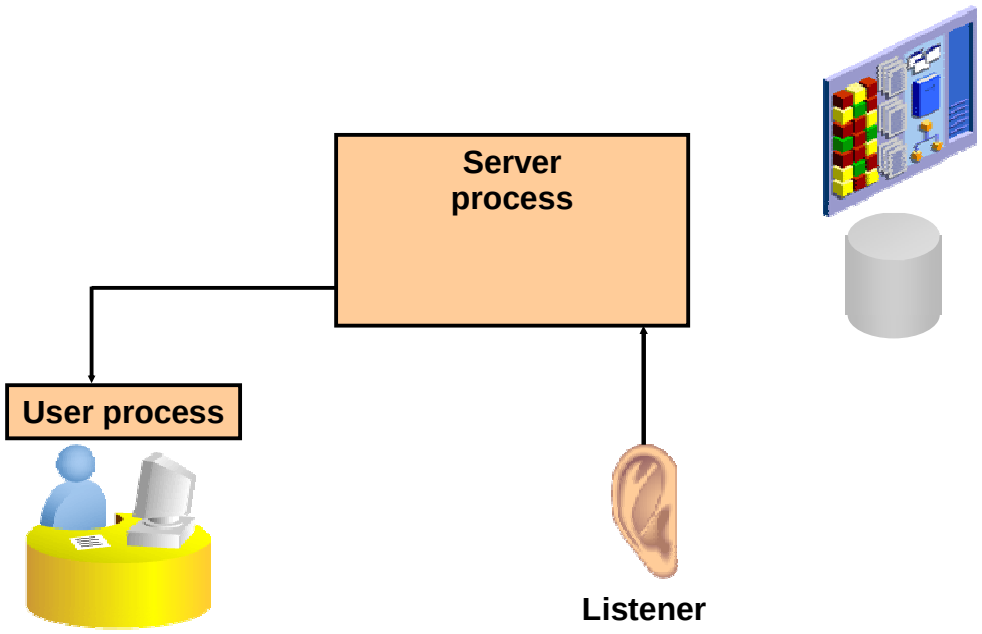


Establishing a Connection

After Oracle Net names resolution is complete, a connection request is passed from the user or middle-tier application (referred to as the user process from here on) to the Oracle Net Listener. The listener receives a `CONNECT` packet and checks to see whether that `CONNECT` packet is requesting a valid Oracle Net service name.

If the service name is not requested (as in the case of a `TNSPING` request), the listener acknowledges the connect request and does nothing else. If an invalid service name is requested, the listener transmits an error code to the user process.

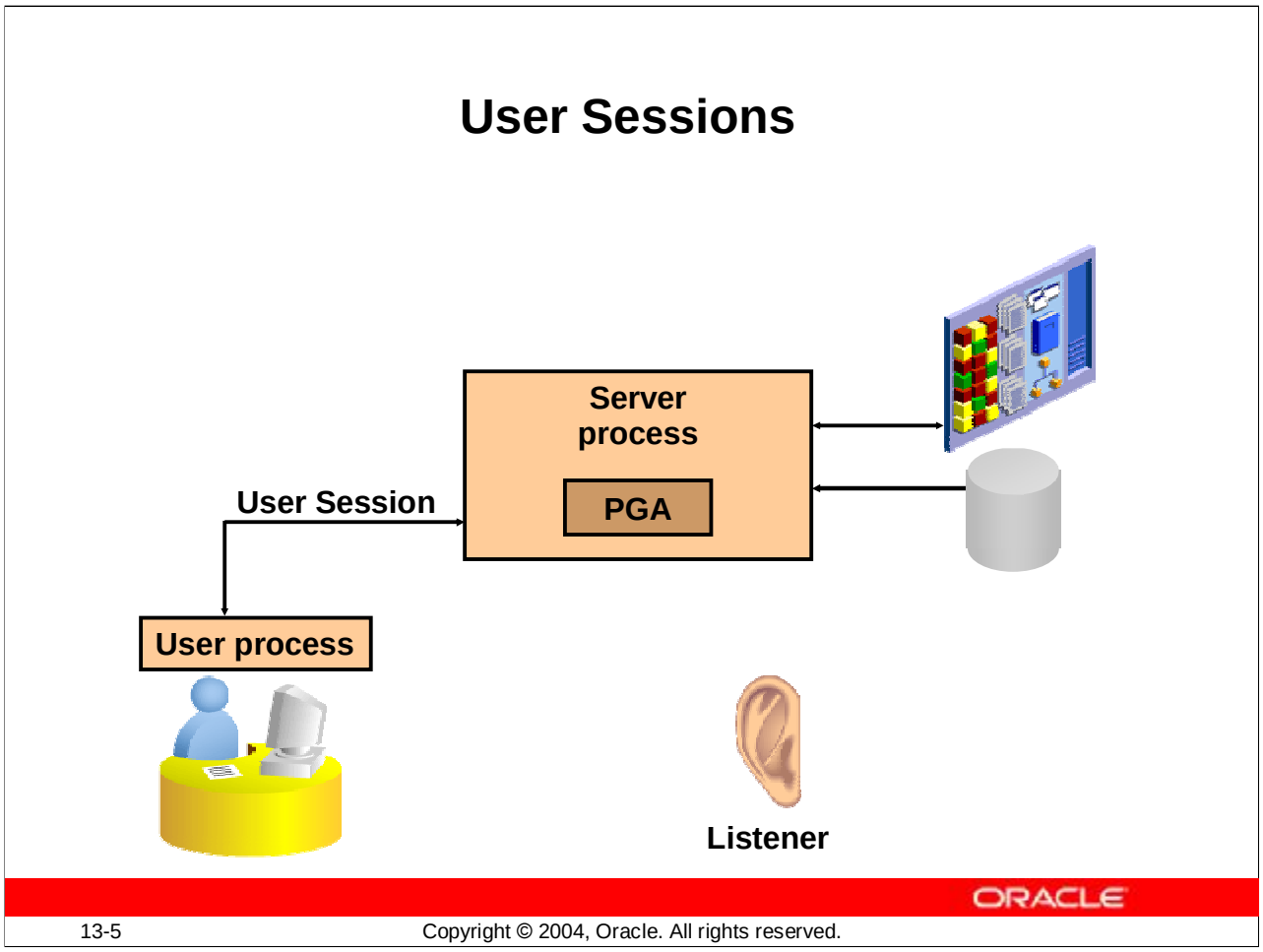
Dedicated Server Process



Dedicated Server Process

If the CONNECT packet requests a valid service name, the listener spawns a new process to deal with the connection. This new process is known as the "server process" and is sometimes also referred to as the "shadow process." After the process has been spawned, the listener connects to the process and passes it initialization information, including the address information for the user process. At this point the listener no longer deals with the connection and all work is handed off to the server process.

The server process now transmits a RESEND packet back to the user process.



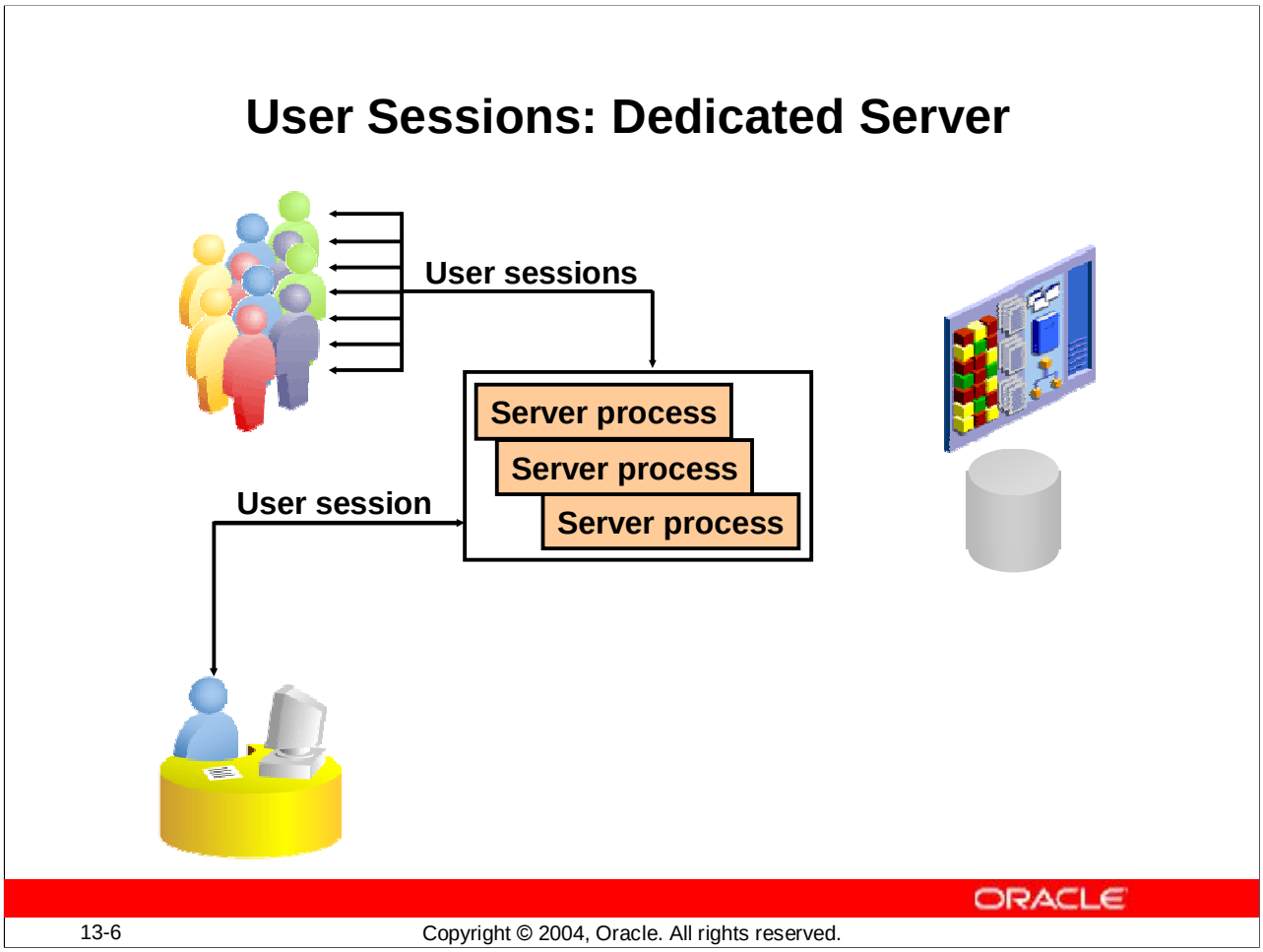
User Sessions

After the user session receives the `RESEND` packet, it retransmits the `CONNECT` packet. The server process checks the user's authentication credentials (usually a password) and if the credentials are valid, a user session is created.

Dedicated server process: With the session established, the server process now acts as the user's agent on the server. The server process is responsible for:

- Parsing and running any SQL statements issued through the application
- Checking the database buffer cache for data blocks required to perform SQL statements
- Reading necessary data blocks from datafiles on disk into the database buffer cache portion of the SGA, if the blocks are not already present in the SGA
- Managing all sort activity. A portion of the server process called the Program Global Area (PGA) contains a memory area known as the Sort Area that is used to work with sorting.
- Returning results to the user process in such a way that the application can process the information

Server processes also reserve memory for specialized work such as bitmap and hash joins. The amount of memory consumed by the dedicated server process depends on several initialization parameter settings. It can be automatically controlled through the use of `PGA_AGGREGATE_TARGET` and `WORKAREA_SIZE_POLICY` or fined-tuned if needed for advanced use.

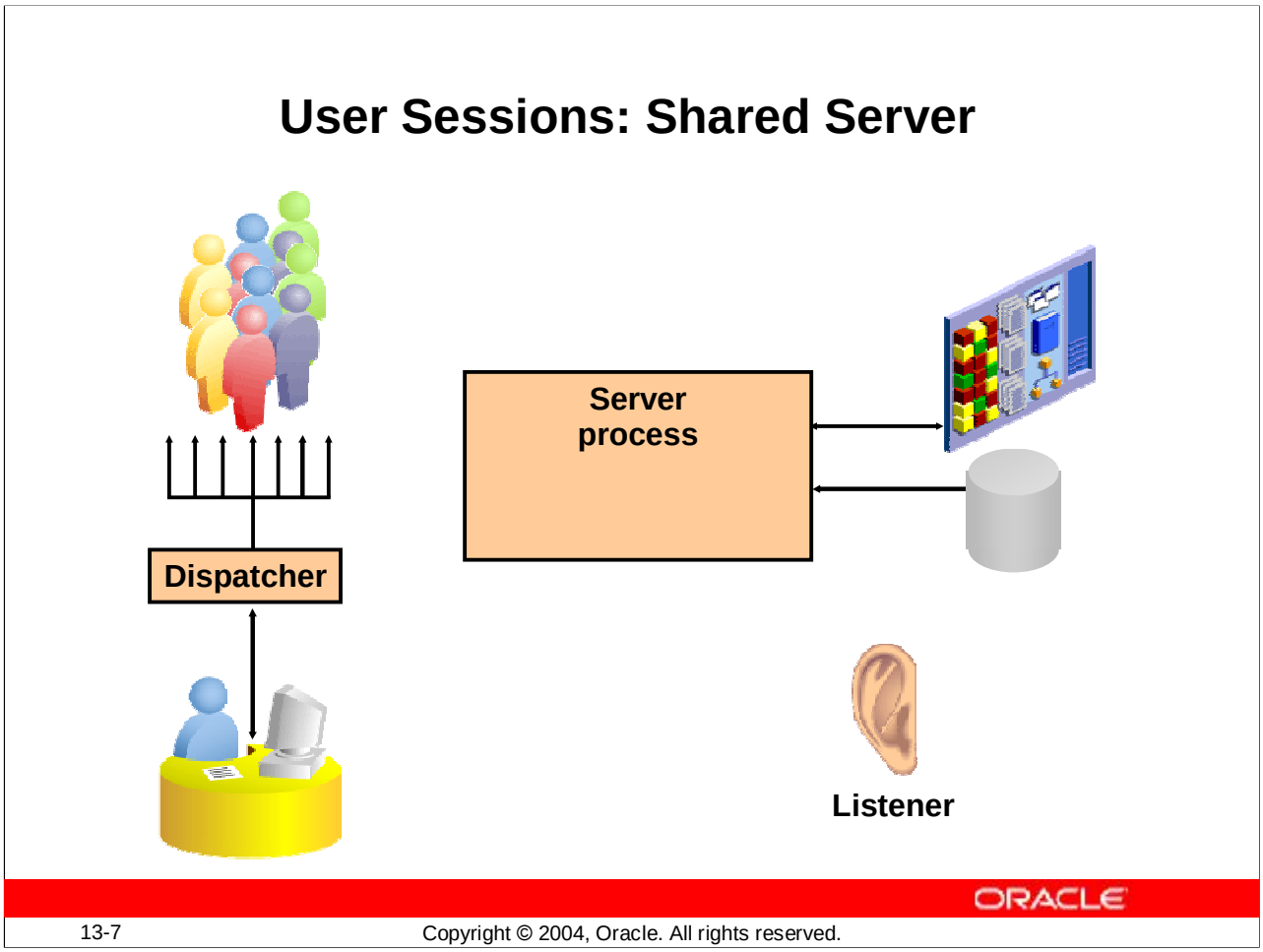


User Sessions: Dedicated Server

With dedicated server processes there is a one-to-one ratio between server processes and user processes. Each server process consumes system resources including CPU cycles and memory.

In a heavily loaded system, the memory and CPU resources consumed by dedicated server processes can be prohibitive and negatively affect the system's scalability. If your system is being negatively impacted by the resource demands of the dedicated server architecture you have two options:

- Increase system resources by adding more memory and additional CPU capability.
- Use the Oracle Shared Server architecture.



User Sessions: Shared Servers

Each service that participates in the Shared Server architecture has at least one (and usually more) dispatcher process. When a connection request arrives, the listener does not spawn a dedicated server process. Instead, the listener maintains a list of dispatchers available for each service name, along with the connection load (number of concurrent connections) for each dispatcher.

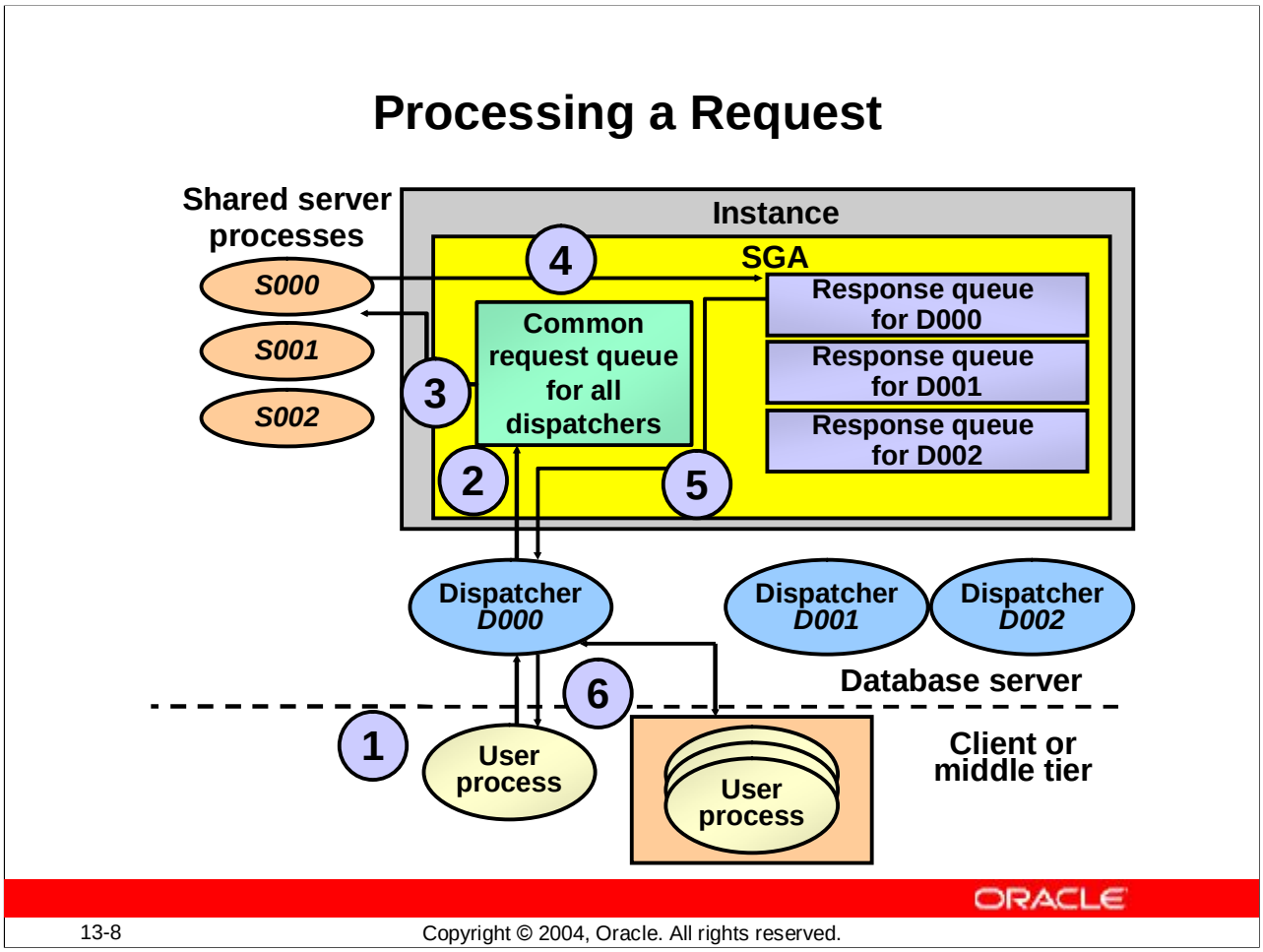
Connection requests are routed to the lightest loaded dispatcher that is servicing a given service name. Users remain connected to the same dispatcher for the duration of a session.

Unlike dedicated server processes, a single dispatcher can manage hundreds of user sessions.

Dispatcher's do not actually handle the work of user requests. Instead they pass user requests to a common queue located in the shared pool portion of the SGA.

Shared Server processes take over most of the work of dedicated server processes, pulling requests from the queue and processing them until complete.

Because a single user session may have requests processed by multiple shared server processes, most of the memory structures usually stored in the Program Global Area (PGA) must be in a shared memory location. In a Shared Server architecture, most of these memory areas are stored in the large pool portion of the SGA.



Processing a Request

When a user connects through the Shared Server architecture submits a database request:

1. The user process forwards the request to its dispatcher.
2. The dispatcher places the request into the common request queue in the SGA.
3. The next available shared server picks up the request from the request queue and processes the request.
4. The shared server places the response on the calling dispatcher's response queue. Each dispatcher has its own response queue.
5. The dispatcher retrieves the response from its response queue.
6. The dispatcher returns the response to the user.

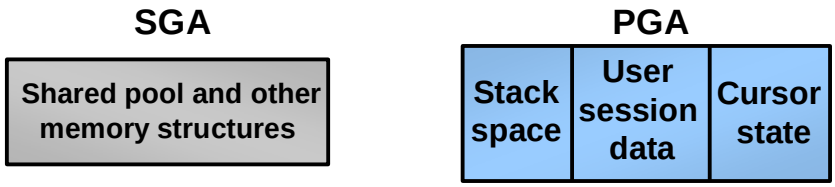
After the user call has been completed, the shared server process is released and is available to service another user call in the request queue.

Request Queue

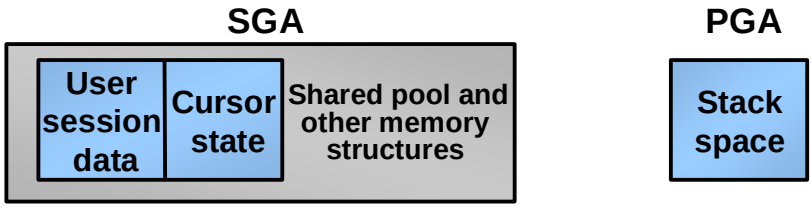
- One request queue is shared by all dispatchers.
- Shared servers monitor the request queue for new requests.
- Requests are processed on a first-in, first-out (FIFO) basis. There is no priority setting.

The SGA and PGA

Dedicated server: User session data is kept in the PGA.



Oracle Shared Server: User session data is held in the SGA.



ORACLE

The SGA and PGA

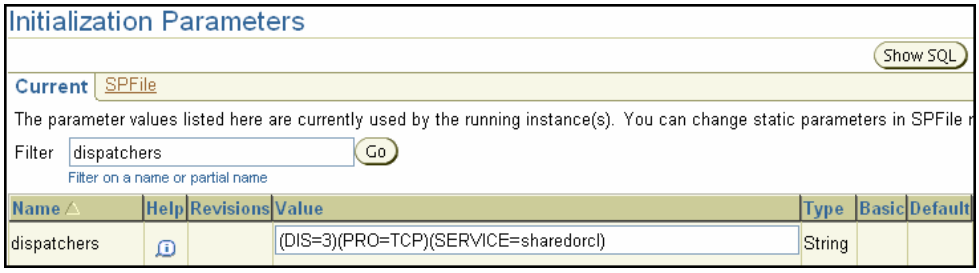
The contents of the System Global Area (SGA) and the Program Global Area (PGA) differ when dedicated servers or shared servers are used.

- Text and parsed forms of all SQL statements are stored in the SGA.
- The cursor state contains run-time memory values for the SQL statement, such as rows retrieved.
- User-session data includes security and resource usage information.
- The stack space contains local variables for the process.

Technical Note

The change in the SGA and PGA is transparent to the user; however, if supporting multiple users, you need to increase the `LARGE_POOL_SIZE`. Each shared server process needs to access the data spaces of all sessions so that any server can handle requests from any session. Space is allocated in the SGA for each session's data space. You can limit the amount of space that a session can allocate by setting the `PRIVATE_SGA` resource limit in the Database Services region of the General page of the user's profile.

Configuring Oracle Shared Server



Required parameter

- **DISPATCHERS**

Optional parameters

- **SHARED_SERVERS**
- **MAX_SHARED_SERVERS**
- **CIRCUITS**
- **SHARED_SERVER_SESSIONS**

ORACLE

Configuring Oracle Shared Server

To configure Oracle Shared Server, you must edit the initialization parameters for your instance. Most of the optional parameters have sensible defaults. On many systems, the only parameter that must be configured is `DISPATCHERS`.

Depending on the options selected when you created your database, the `DISPATCHERS` parameter may already be configured to start one dispatcher to service the XML database. The `DISPATCHERS` parameter accepts multiple sets of values in the format:

```
'<parameters for first set>','<parameters for second set>'
```

DISPATCHERS

Specifies the number of dispatchers that are initially started for a given protocol.

```
DISPATCHERS = "(PROTOCOL=TCP)(DISPATCHERS=2)\
              (PROTOCOL=IPC)(DISPATCHERS=1)"
```

ORACLE

DISPATCHERS

The DISPATCHERS parameter enables various attributes for each dispatcher.

Oracle Database10*g* supports a name-value syntax (similar to the syntax used by Oracle Net Services) to enable the specification of existing and additional attributes in a position-independent, noncase-sensitive manner.

For example: `DISPATCHERS = '(PROTOCOL=TCP)(DISPATCHERS=3)'`

Parameter Type	String (Specify as a quoted string)
Parameter class:	Dynamic
Default value:	Null (no dispatchers will be started)

Although the number of connections a dispatcher can handle varies greatly depending on the type of workload, a good measure to use is to allow one dispatcher per every fifty concurrent database connections that use the shared server architecture.

DISPATCHERS (continued)

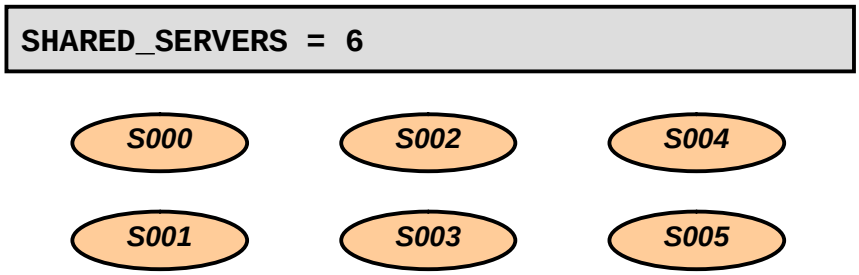
The only required dispatcher attribute is PROTCOL. All others are optional. A few of the possible arguments for the DISPATCHERS parameter are described below. Notice that arguments use a three-letter abbreviation instead of the full argument name.

Attribute	Description
PROTOCOL (PRO or PROT)	Specifies the network protocol for which the dispatcher generates a listening endpoint (usually TCP)
DISPATCHERS (DIS or DISP)	The initial number of dispatchers to start (default is 1)
SERVICE (SER or SERV)	The Oracle Net Service name the dispatcher registers with the listener. If not given, the dispatcher registers the values in SERVICE_NAMES.
LISTENER (LIS or LIST)	Specifies an alias name for the listeners with which the PMON process registers dispatcher information. Set the alias to a name which is resolved through a naming method. This attribute need only be specified if the listener is a local listener that uses a nondefault port (not 1521) and is not specified with the LOCAL_LISTENER parameter *or* the listener is on another node.
SESSIONS (SES or SESS)	The maximum number of network sessions for each dispatcher. The default is operating system–specific. Most operating systems have a default of 16 K.
CONNECTIONS (CON or CONN)	Specifies the maximum number of network connections to allow for each dispatcher. The default is operating system–specific. For example, 1024 is the default for Sun Solaris and Windows.

Note: There are many more possible attributes for the DISPATCHERS parameter. Further details on the DISPATCHERS parameter can be found in the "Initialization Parameters" section in the *Oracle Database Reference Manual*.

SHARED_SERVERS

Specifies the number of shared server processes created when an instance is started up, and retained during instance operation.



ORACLE

SHARED_SERVERS

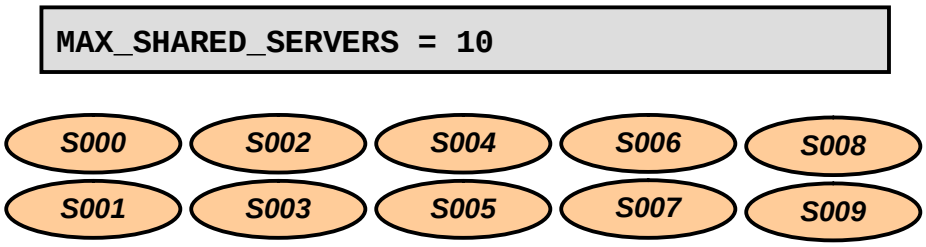
SHARED_SERVERS specifies the minimum number of server processes that that will be retained. Setting this parameter is not usually critical because the instance monitors the Common Request Queue and will start additional shared servers as needed to serve the queue, and then dispose of them when no longer needed.

Parameter type	Integer
Parameter class	Dynamic
Default value	0 if DISPATCHERS is NULL, 1 if DISPATCHERS is set
Range of values	Operating system dependent

A good measure to use is to retain one shared server for every twenty-five concurrent database connections using the shared server architecture.

MAX_SHARED_SERVERS

- **Specifies the maximum number of shared servers that can be started**
- **Allows shared servers to be allocated dynamically based on the length of the request queue**



ORACLE

MAX_SHARED_SERVERS

MAX_SHARED_SERVERS specifies the maximum number of shared server processes that will be allowed to run simultaneously. The setting is important because the instance automatically creates additional shared server processes as needed to service the common request queue.

Parameter type	Integer
Parameter class	Dynamic
Default value	None (unlimited)
Range of values	Operating system dependent

Estimating the Maximum Number of Shared Servers

In general, set this parameter for 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. To find the maximum numbers of servers started (high-water mark), query the V$SHARED_SERVER_MONITOR data dictionary view.

CIRCUITS

- **Specifies the total number of virtual circuits that are available for inbound and outbound network sessions**
- **Contributes to total SGA size**

```
CIRCUITS = 100
```

ORACLE

CIRCUITS

Virtual circuits are user connections to the database through dispatchers and servers.The `CIRCUITS` parameter specifies the total number of virtual circuits that are available for inbound and outbound network sessions.

Parameter type	Integer
Parameter class	Dynamic
Default value	If Oracle Shared Server is configured, then the value of `CIRCUITS` will match that of `SESSIONS`. Otherwise, the value is 0

Set this parameter only if you want to limit the total number of connections users can make via the shared server architecture. This parameter is of interest because it is one of several parameters that contribute to the total SGA requirements of an instance.

SHARED_SERVER_SESSIONS

- Specifies the total number of Oracle Shared Server user sessions to allow
- Enables you to reserve user sessions for dedicated servers

```
SHARED_SERVER_SESSIONS = 100
```

ORACLE

SHARED_SERVER_SESSIONS

This parameter controls the total number of shared server sessions open concurrently at any point in time. The use of this parameter allows resources for dedicated user sessions to be reserved.

Parameter type	Integer
Parameter class	Dynamic
Default value	None (unlimited)

Related Parameters

Other initialization parameters affected by Oracle Shared Server that may require adjustment:

- **LARGE_POOL_SIZE**
- **SESSIONS**
- **PROCESSES**
- **LOCAL_LISTENER**

ORACLE

Related Parameters

Other parameters affected by Oracle Shared Server that may require adjustment:

- LARGE_POOL_SIZE specifies the size in bytes of the large pool. Oracle Shared Server uses the large pool to store session information that usually resides in the PGA in a dedicate server session.
- SESSIONS specifies the maximum number of sessions that can be created in the system. This may need to be adjusted for Oracle Shared Server because your system can now service more sessions.
- PROCESSES controls the number of server-side processes.
- LOCAL_LISTENER defines the port and protocol used by the listeners. If your listener is not using TCP/IP on port 1521, or if you have multiple listeners, you must configure LOCAL_LISTENER so the dispatchers can register with the listeners.

If you do not set a value for LARGE_POOL_SIZE, then Oracle uses the shared pool for Oracle Shared Server user session memory. This can negatively impact the performance of PL/SQL, SQL, and other services that rely on the shared pool.

Oracle allocates some fixed amount of memory (about 10 K) per configured session from the shared pool, even if you have configured the large pool.

Verifying Shared Server Setup

- **Verify that the dispatcher has registered with the listener when the database was started by issuing:**

```
$ lsnrctl SERVICES
```

- **Verify that you are connected using shared servers by making a connection and then query V$CIRCUIT view to show one entry per shared server connection.**

ORACLE

Verifying Shared Server Setup

When using Oracle Shared Server, you should first start the listener, then the database, so that the dispatchers can immediately register with the listener. If you later restart the listener, allow one minute for services to reregister. To verify that registration has taken place issue the following command:

```
$ lsnrctl services
Service "TEST" has 1 instance(s).
Instance "TEST", status READY, has 3 handler(s) for this service.
 Handler(s):
"DISPATCHER" established:1 refused:0 curr:0 max:1022 state:ready
   D001 <machine: db.us.oracle.com, pid: 8705>
   (ADDRESS=(PROTOCOL=tcp)(HOST=db.us.oracle.com)(PORT=35230))
"DISPATCHER" established:1 refused:0 curr:0 max:1022 state:ready
   D000 <machine: db.us.oracle.com, pid: 8703>
   (ADDRESS=(PROTOCOL=tcp)(HOST=db.us.oracle.com)(PORT=35229))
"DEDICATED" established:0 refused:0
```

Verifying Shared Server Setup (continued)

Verify that your connections are using shared servers by making connections, and then query the V$CIRCUIT view to show one entry per shared server connection. This also verifies that the listener is performing load balancing for incoming connections.

```
SQL>select dispatcher, circuit, server, status from v$circuit;

DISPATCH CIRCUIT  SERVER   STATUS
-------- -------- -------- ----------------
82890064 8257BA64 8288F6A4 NORMAL
8288F9E4 8257BBB0 00       NORMAL
8288FD24 8257BCFC 00       NORMAL
```

Data Dictionary Views

- **V$CIRCUIT**
- **V$SHARED_SERVER**
- **V$DISPATCHER**
- **V$SHARED_SERVER_MONITOR**
- **V$QUEUE**
- **V$SESSION**

ORACLE

Data Dictionary Views

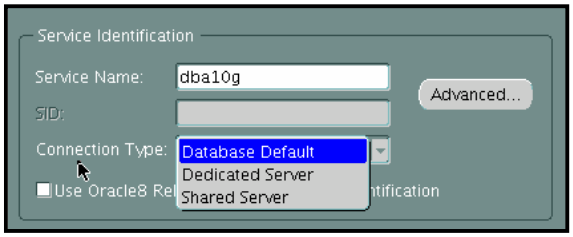
V$CIRCUIT	This view contains information about virtual circuits, which are user connections to the database through dispatchers and servers. Any shared server connection creates an entry in V$CIRCUIT.
V$SHARED_SERVER	This view contains information on the shared server processes.
V$DISPATCHER	This view provides information on the dispatcher processes.
V$SHARED_SERVER_ MONITOR	This view contains information for tuning the shared server processes.
V$QUEUE	This view contains information on request and response queues.
V$SESSION	This view lists session information for each current session.

Choosing a Connection Type

Unless otherwise configured, Oracle Net connections will use:

- **Shared server if one is available**
- **Dedicated server if a shared server connection is not available**

Using local or directory naming, the connection type can be specified as part of the Net service alias.



ORACLE

Choosing a Connection Type

Oracle Net's default connection type is the shared server connection. If the instance has been configured for a shared server and a connect request does not specifically ask for a dedicated server, the connection type will be shared.

The Oracle Net Manager allows the connection type to be specified for both local naming and directory naming. Select the desired `Connection Type` from drop-down list.

When Not to Use Shared Server

Certain types of database work should not be performed using shared servers:

- **Database administration**
- **Backup and recovery operations**
- **Batch processing and bulk load operations**
- **Data warehouse operations**

Dispatcher

Server process

ORACLE

When Not to Use Shared Server

The Oracle Shared Server architecture is an efficient process and memory use model, but is not appropriate for all connections. Because of the common request queue, and the fact that many users may share a dispatcher response queue, shared servers do not perform well with operations that must deal with large sets of data such as warehouse queries or batch processing.

Backup and recovery sessions using Oracle Recovery Manager (discussed in later chapters) also deal with very large data sets and should make use of dedicated connections.

Many administration tasks cannot or should not be performed using shared server connections. These include startup and shutdown of the instance, creating tablespaces or datafiles, index and table maintenance, analyzing statistics, and many other tasks commonly performed by the DBA. All DBA sessions should choose dedicated servers.

Summary

In this lesson you should have learned how to:

- **Identify when to use Oracle Shared Servers**
- **Configure Oracle Shared Servers**
- **Monitor Shared Servers**

ORACLE

Practice 13 Overview: Oracle Shared Servers

This practice covers the following topics:

- **Investigating the impact of dedicated server connections on your system**
- **Configuring your system to use shared servers**
- **Investigating the impact of shared server connections on your system**

ORACLE

Practice 13: Oracle Shared Server

Background: You've noticed that during peak load times, your system is performing poorly. After investigating, you find that user sessions are consuming so much memory that your system is swapping excessively. Configure your system to reduce the amount of memory consumed by user sessions.

Tasks:

- Investigate the impact of dedicated server connections on your system
- Configure your system to use shared servers
- Investigate the impact of shared server connections on your system

1. Investigate the impact of dedicated server connections on your system.
   a) Open four terminal sessions on your server.
   b) Check to see how many Oracle processes are running.
   c) In your remaining three terminal sessions start SQL*Plus sessions. Verify that none of your three SQL*Plus sessions is using shared servers.
   d) Your default service name is configured to use dedicated server processes. Note that with each SQL*Plus session started, two new processes are created. Remember that each of your SQL*Plus sessions starts *two* resource-consuming processes. One is the SQL*Plus session itself (which will normally consume resources on the client or middle tier), the other is the dedicated server process (which consumes resources on the server).
2. Configure your system to use shared servers. Use the TCP protocol and your system's service name.
3. Investigate the impact of shared server connections on your system.
   a) Exit your three SQL*Plus sessions (leave the four terminal sessions open; you will need them later).
   b) Create a new local naming alias that uses the default database connection.
   c) Open three SQL*Plus sessions using the new service name.
   d) Verify that you are connected using a shared server (you will probably see an extra connection in `V$CIRCUIT` because the Enterprise Manager monitoring process usually logs in to the new shared server system before you can).
   e) Check to see how many Oracle processes are running now. Note that your three SQL*Plus sessions started only three new processes (these are the three SQL*Plus processes). All three of these are usually located on the client machine or application server, which means that these three new sessions have not added *any* new processes on the server.

14

Performance Monitoring

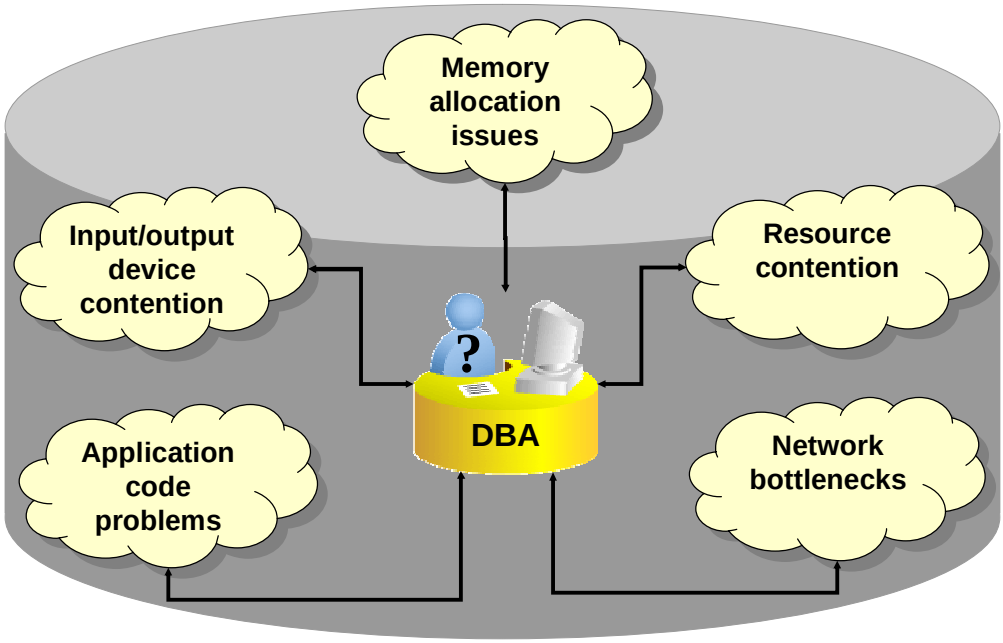
Objectives

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

- **Troubleshoot invalid and unusable objects**
- **Gather optimizer statistics**
- **View performance metrics**
- **React to performance issues**

ORACLE

Performance Monitoring



ORACLE

Performance Monitoring

To administer Oracle Database 10*g* and keep it running smoothly, the database administrator must regularly monitor performance to locate bottlenecks and correct problem areas.

There are hundreds of performance measurements the DBA can look at, covering everything from network performance to disk input/output speed to the time spent working on individual application operations. These performance measurements are commonly referred to as database metrics.

Monitoring Methodologies

- **Reactive**
- **Proactive**
  - **Server-generated alerts**
  - **Automated Database Diagnostic Monitor (ADDM)**

ORACLE

Monitoring Methodologies

The Oracle Database 10*g* supports both proactive and reactive monitoring. Reactive monitoring is a response to a known or reported problem. You may begin monitoring performance metrics in reaction to user complaints about response time, instance failures, or errors found in the alert log.

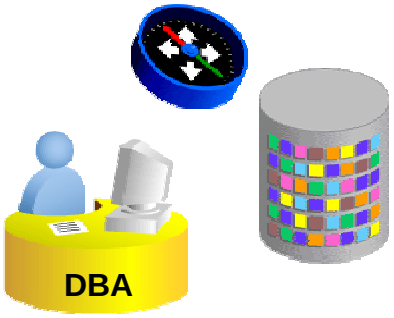
Reactive monitoring is inevitably necessary sometimes, but your goal should be to detect and repair problems before they affect system operation. Fixing problems before they occur, or at least before they are widely noticed, is a proactive approach to system maintenance.

Oracle Database 10*g* includes several tools that aid in proactive monitoring. Two of these are server-generated alerts and the Automated Database Diagnostic Monitor (ADDM). Proactive monitoring will be covered in the next lesson.

Database and Instance Metrics

Several hundred different performance statistics are available through:

- **Data dictionary**
- **Dynamic performance views**
- **Optimizer statistics**



ORACLE

Database and Instance Metrics

There are hundreds of different measurement points available to monitor your system.

The data dictionary provides information about space consumption and object status.

Real-time performance metrics covering memory usage, wait events, input/output device throughput, and instance activity are available through dynamic performance views, also known as v$ views.

Oracle Databases can be extremely large (terabyte databases are becoming common and exabyte databases are possible). With such large datasets it is critical that the instance locate the particular piece of data a user is looking for in the most efficient way possible. The portion of the instance that decides how to look for data is known as the *optimizer*.

Data distribution metrics are used to help the Oracle Database 10*g* optimizer choose the most efficient way to retrieve data, and are therefore commonly referred to as “optimizer statistics.”

Database and Instance Metrics (continued)

Optimizer statistics showing data distribution for each table include:

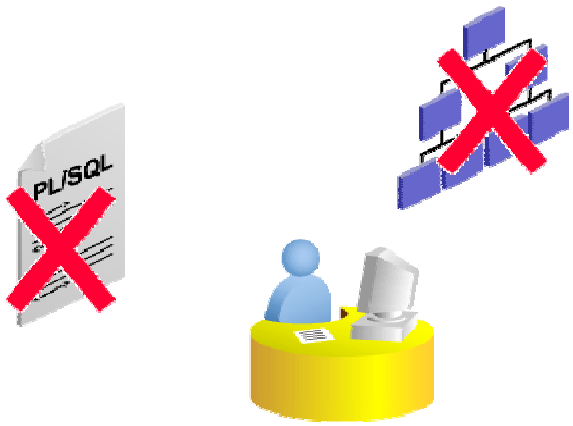
- Number of rows
- Average row length
- Amount of empty space that is allocated to the table
- Number of rows that are “chained” (existing in two or more data blocks due to long row length or inefficient data storage settings)

One example of an optimizer choice is between a full table scan (reading the entire table to find a particular row) and an index scan (looking for the row in an index first, then going directly to the data block in the table that holds the row). For small tables, a full table scan is usually more efficient than looking through an index for data and later jumping to the correct block. For a large table (remember that large can easily be hundreds of millions of rows) an index scan probably makes much more sense.

Data Dictionary Metrics

Object status:

- **PL/SQL code objects**
- **Indexes**



ORACLE

Data Dictionary Metrics

One of the most important data dictionary metrics deals with object status. Database indexes and stored procedures both need to be `VALID` to be used.

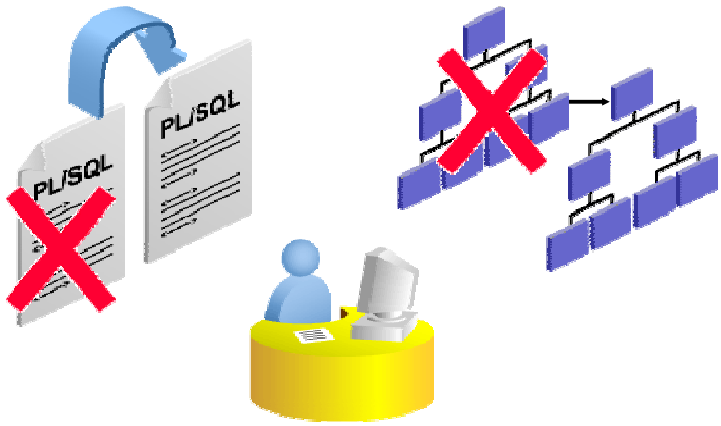
Indexes can become `UNUSABLE` due to normal maintenance operations on tables. Indexes contain pointers to the physical location of individual rows of data. If the DBA causes those rows to change location through maintenance operations such as the `MOVE` command, the pointers are no longer valid and the index will not be usable until it is rebuilt.

PL/SQL code objects have an `INVALID` status for several reasons. If the code contains a programming error, the PL/SQL object will not compile correctly and will be marked invalid. A valid procedure can *become* invalid if an object the procedure references changes. Suppose a procedure references the `salary` column of the `employees` table. If the column precision is changed, or if the column is dropped, there is no guarantee that the procedure that worked correctly on the *old* table definition will still run on the *new* table definition.

Unusable indexes always require DBA intervention to restore them to a valid state. Invalid PL/SQL objects will normally recompile automatically the first time they are called, but sometimes require DBA intervention in cases where the automatic recompilation fails.

Invalid and Unusable Objects

- **PL/SQL code objects are recompiled.**
- **Indexes are rebuilt.**



ORACLE

Invalid and Unusable Objects

If you find PL/SQL objects with a status of `INVALID`, the first question you need to answer is "has this object *ever* been `VALID`?" Often, an application developer neglects to clean up code that does not work. If the PL/SQL object is invalid because of a code error, then there is little that can be done until that error is fixed. If the procedure was valid at some time in the past and has recently become invalid, you have two options to fix the problem:

1. Do nothing. Most PL/SQL objects will automatically recompile if needed when they are called. Users will experience a slight delay while the objects recompile (in most cases this delay is not even noticeable).
2. Manually recompile the invalid object.

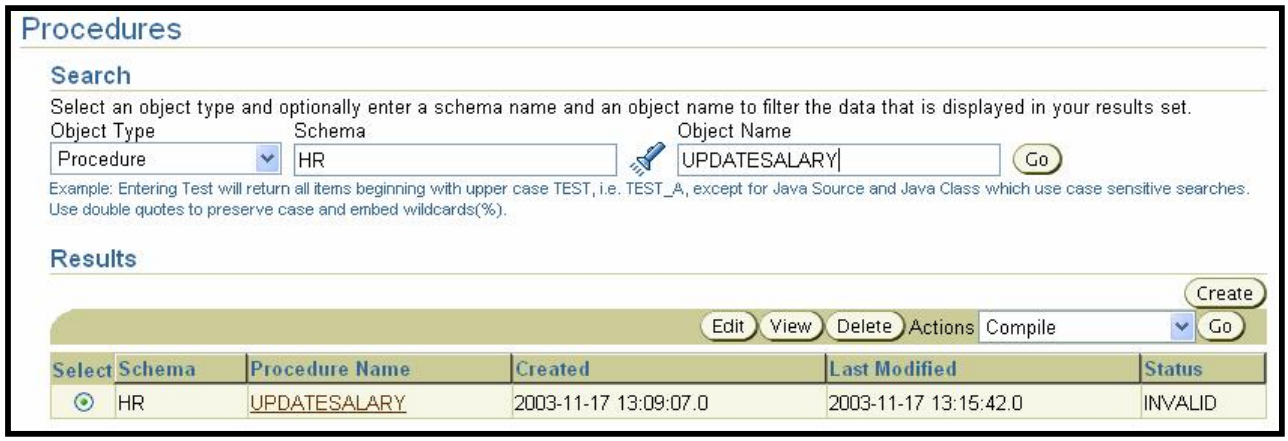
Invalid PL/SQL objects can be manually recompiled using Enterprise Manager or through SQL commands:

```
ALTER PROCEDURE HR.updatesalary COMPILE;
```

Manually recompiling PL/SQL packages requires two steps:

```
ALTER PACKATE HR.maintainemp COMPILE;
ALTER PACKATE HR.maintainemp COMPILE BODY;
```

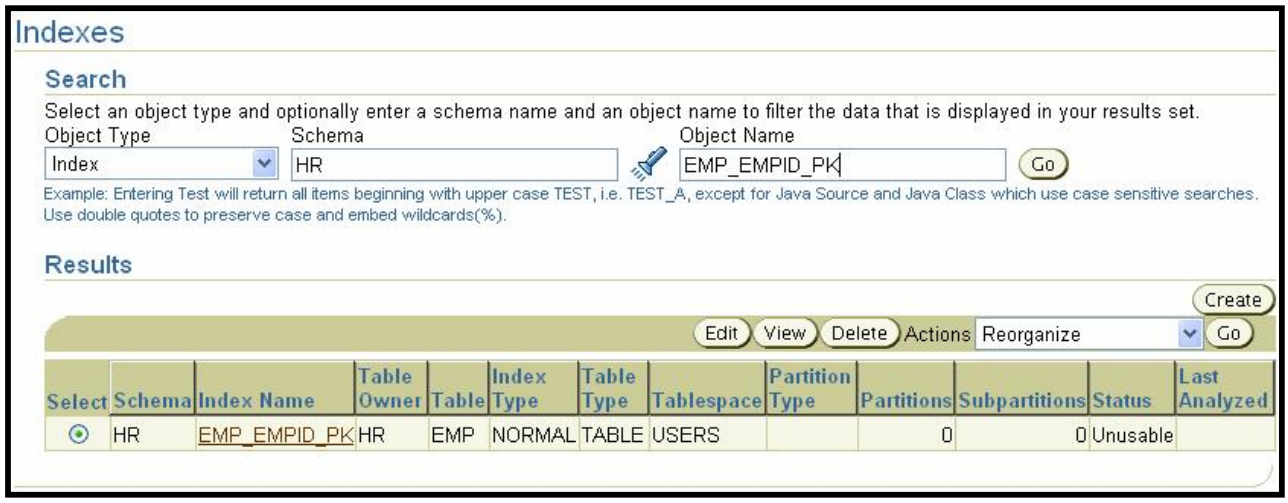
Invalid and Unusable Objects (continued)



Unusable indexes are made valid by rebuilding them to recalculate the pointers. Rebuilding an unusable index re-creates the index in a new location, then drops the unusable index. This can either be done using Enterprise Manager or through SQL commands.

```
ALTER INDEX HR.emp_empid_pk REBUILD;
ALTER INDEX HR.emp_empid_pk REBUILD ONLINE;
ALTER INDEX HR.email REBUILD TABLESPACE USERS;
```

If the TABLESPACE clause is left out, the index is rebuilt in the same tablespace where it already exists. The REBUILD ONLINE clause allows users to continue updating the index's table while the rebuild takes place (without the ONLINE keyword users must wait for the rebuild to finish before performing DML on the affected table).



Enterprise Manager uses the reorganize action to repair an UNUSABLE index.

Note: Rebuilding an index requires that free space be available for the rebuild. Verify that there is sufficient space before attempting the rebuild. Enterprise Manager checks space requirements automatically.

Optimizer Statistics

Optimizer statistics are:

- **Not real-time**
- **Persistent across instance restarts**

```
SQL> SELECT COUNT(*) FROM hr.employees;
  COUNT(*)
----------
       214
SQL> SELECT num_rows FROM dba_tables
  2  WHERE owner='HR' AND table_name = 'EMPLOYEES';
  NUM_ROWS
----------
       107
```

ORACLE

Optimizer Statistics

Optimizer statistics for tables and indexes are stored in the data dictionary. These statistics are not intended to provide real-time data. They provide the optimizer a *statistically* correct snapshot of data storage and distribution which the optimizer uses to make decisions on how to access data.

Metrics collected include:

- Size of the table or index in database blocks
- Number of rows
- Average row size and chain count (tables only)
- Height and number of deleted leaf rows (indexes only)

As data is inserted, deleted, and modified these values change. The performance impact of maintaining real-time data distribution statistics would be prohibitive, so these statistics are updated by periodically gathering statistics on tables and indexes.

If your database was created with the Database Creation Assistant (DBCA), optimizer statistics are automatically collected once a day between 10:00 PM and 6:00 AM. If your database was not created with the DBCA, or if your database contains tables and indexes that are very volatile, you must manually collect optimizer statistics.

Optimizer Statistics (continued)

A large table that experiences 10 percent growth (or reduction) within a 24-hour period is usually considered too volatile for statistics collection once per day to be sufficient. For tables that experience this level of change, Oracle recommends collecting statistics more frequently, preferably often enough that the table never changes by more than about 10 percent between collection periods.

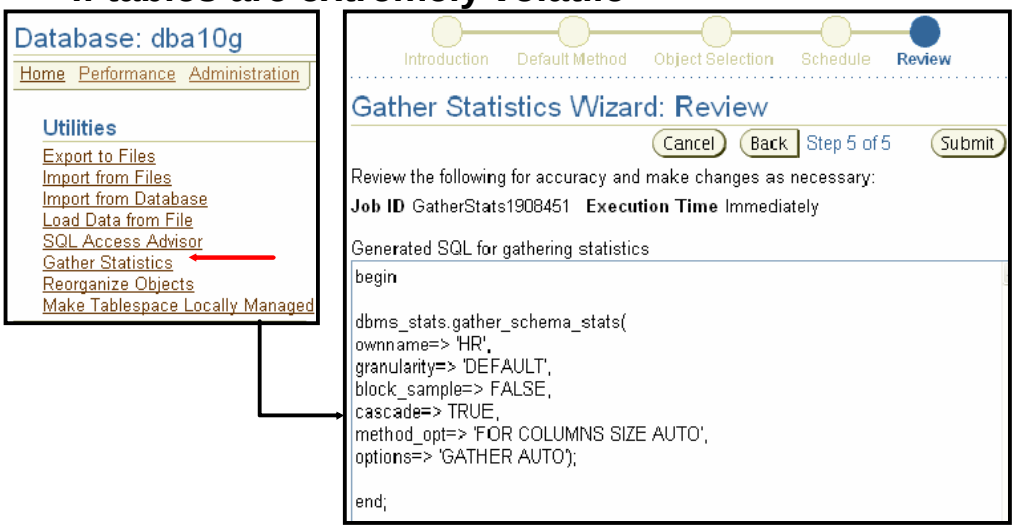
Statistics can be manually collected using Enterprise Manager, or through the use of the DBMS_STATS package as shown below:

```
SQL> EXEC dbms_stats.gather_table_stats('HR','EMP');
SQL> SELECT num_rows FROM dba_tables
  2  WHERE owner='HR' AND table_name = 'EMP';
  NUM_ROWS
----------
       214
```

Note that the number of rows now correctly reflects what was in the table as of the time statistics were gathered. DBMS_STATS also allows manual collection of statistics for an entire schema, or even the whole database.

Manually Gather Optimizer Statistics

- If database was not created with the DBCA
- If tables are extremely volatile



ORACLE

Gathering Optimizer Statistics

Enterprise Manager's Gather Statistics Wizard (available from the Maintenance page) is the recommended way to manually collect or update optimizer statistics. The wizard prompts for information and then creates a job to collect statistics.:

1. Default Method: Offers the choice to delete, estimate, or compute statistics. Computing statistics examines every row in a table; estimating takes a statistical sample. For indexes and small tables the compute method should be used. For large tables an estimate is usually best. In most cases the default estimate percentage should be used for estimates.
   Note: An estimate percentage of 50% or higher triggers a compute. When manually choosing an estimate size, 5–15% is usually sufficient.
2. Object selection: Choose the objects to gather statistics for. You can select individual tables or indexes or broaden the scope to collect for entire schemas or even the complete database.
3. Schedule: Provides host credentials and determines when the statistics will be collected. Choices include immediately, a scheduled time, or a named maintenance window.
4. Review: Shows the code generated by the wizard and allows you to submit the job.

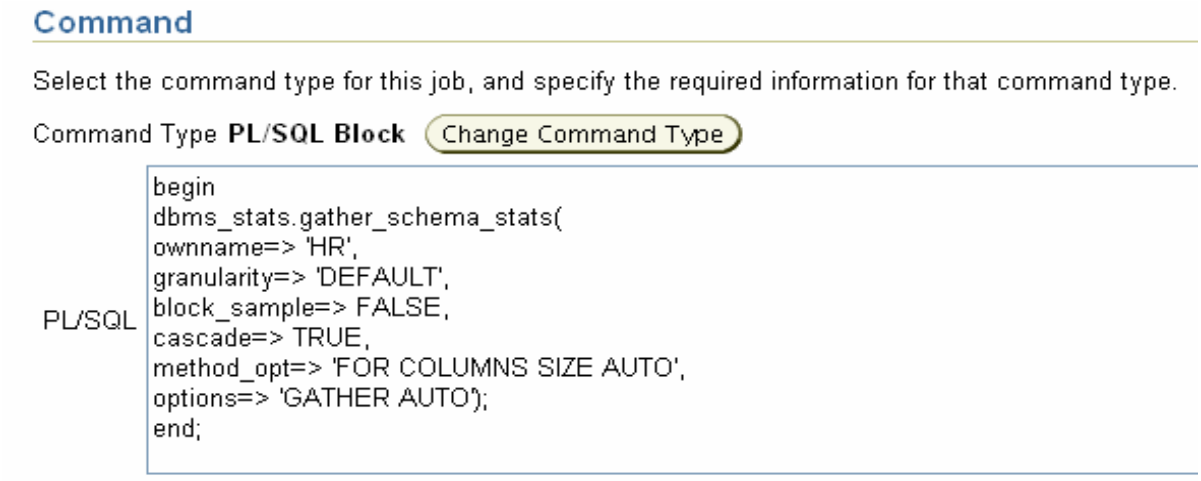
Automate Optimizer Statistics Collection

Use the Oracle Scheduler to automate customized statistics collection.

ORACLE

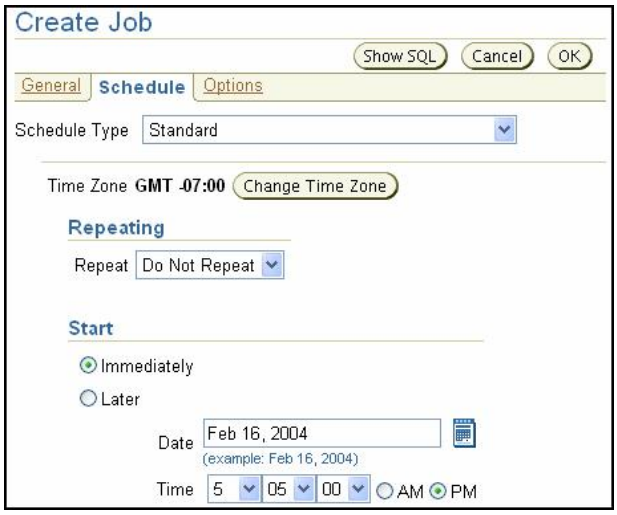
Automate Optimizer Statistics Collection

The Oracle Scheduler automates repetitive tasks such as statistics collection. From the Administration page, click Jobs and create a new job. If you are unfamiliar with PL/SQL, you can copy the code generated by manually collecting statistics and paste it in the command section.



Schedule Optimizer Statistics Collection

Statistics should be gathered as needed to ensure the optimizer can make appropriate decisions.



ORACLE

Schedule Optimizer Statistics Collection

Schedule collection of optimizer statistics frequently enough so that the optimizer can make good decisions on the best methods to access data.

Remember that the act of collecting statistics causes the instance to perform work. Avoid scheduling statistics collection during peak operating times to minimize the impact on normal operations.

Dynamic Performance Views

Dynamic Performance views are:

- **Real-time**
- **Non-persistent across instance restarts**

```
SQL> SELECT name, value FROM v$sysstat
  2  WHERE name='sorts (memory)' ORDER BY name;
NAME                 VALUE
-------------- ----------
sorts (memory)     1979183
SQL> /
NAME                 VALUE
-------------- ----------
sorts (memory)     1979184
```

ORACLE

Dynamic Performance Views

There are over 300 dynamic performance views in the Oracle Database 10*g* that provide detailed information on session and system statistics, file I/O, and instance activity.

Dynamic performance views provide a snapshot of real-time data. As you can see from the example above, each access of a dynamic performance view gives up-to-date results. For the statistic shown above, the act of selecting the information with the `ORDER BY` clause caused `sorts (memory)` to increment.

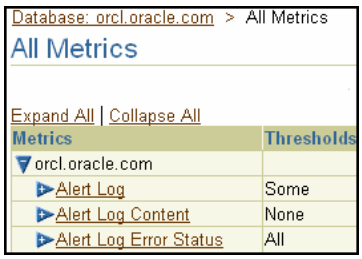
Dynamic performance data is also nonpersistent. After the instance is shut down, the views reset and begin counting up again. That means that performance data obtained from dynamic performance views should not be used to make configuration decisions until the instance has been operational for a while.

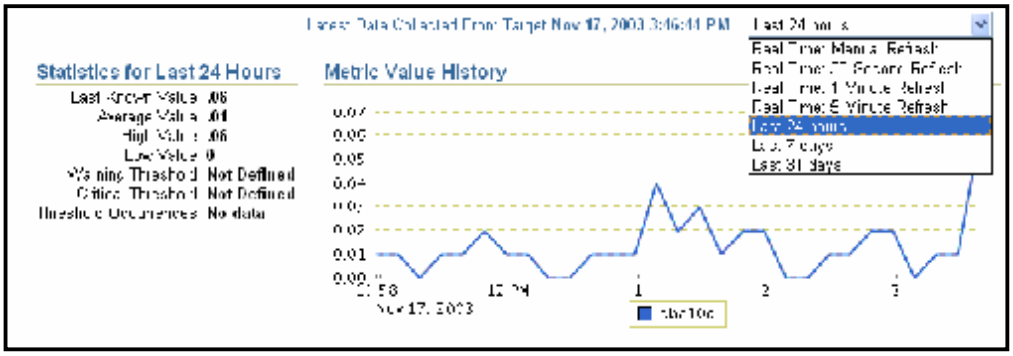
Dynamic performance metrics are usually not meaningful as a single measurement. They should be considered in the context of performance trends (increasing, decreasing) or against baseline data.

For more information on dynamic performance views, refer to *Oracle Database: Reference*.

Viewing Metric Information

- **Use the All Metrics link in the Related Links region.**
- **Drill-down for in-depth analysis.**





Viewing Performance Information

Enterprise Manager's All Metrics page groups metrics by functional area.

1. Expand the area of interest.
2. Click the link for the metric you want to view.

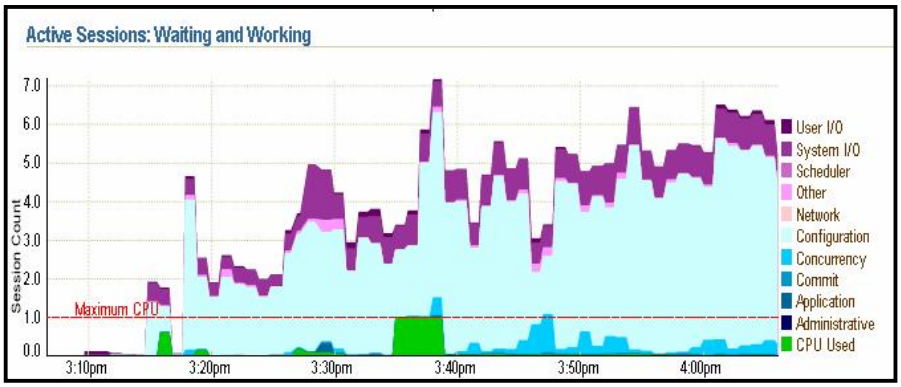
The metric detail page displays:

- Last known value
- High- and low- water mark values
- Average readings
- Graphical representation of the metric plotted over a user-defined time period

Reacting to Performance Issues

Use Enterprise Manager to:

- **Find key performance issues**
- **Drill down to the root cause**



ORACLE

Reacting to Performance Issues

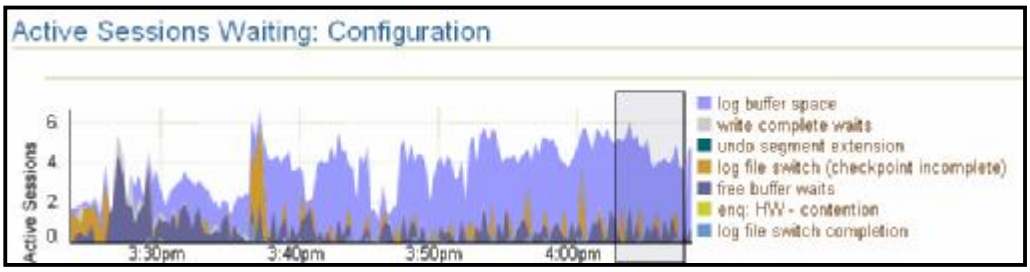
Enterprise Manager's Performance page is the best place to start if you notice a degradation in system performance. The Performance page is divided into three main regions:

- Host: Server metrics showing the number of waiting processes and the amount of memory paging
- Active Sessions: Waiting and Working: An overview of instance performance showing grouped metrics from multiple categories. If one category is consuming a significant portion of the wait time for the instance, that category is where troubleshooting should start.
- Instance Throughput: Information about current sessions, redo generation, and data file read activity.

Each of these sections can be drilled into by clicking the metric of interest. In the example shown in the slide, users are spending a significant portion of their time waiting on a configuration issue. Click the Configuration measurement to drill down and see where the problem is.

Reacting to Performance Issues

Drill down into performance measurements to identify bottlenecks

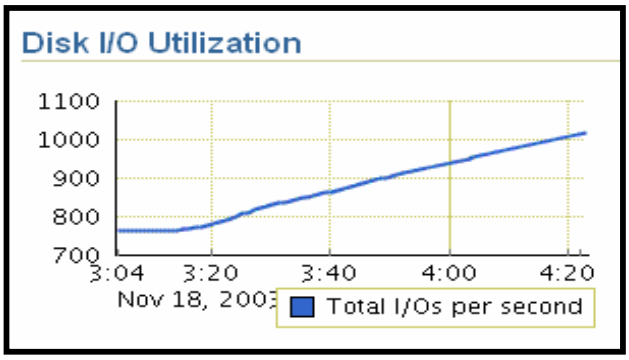


- **Key bottleneck: log buffer space**

Reacting to Performance Issues (continued)

Drilling down into the configuration wait time reveals that the majority of the wait time is for space in the redo log buffer. In this case, the DBA must determine whether the problem is with the log buffer itself (needing more memory allocated) or with the redo log files (file write time may be too high for `LGWR` to keep up with demand).

Drilling into the Run Queue graph in the Host region of the Performance page reveals that disk I/O is high and has been steadily climbing. After further investigation using operating system tools you might find that the redo log groups are on the same disks as the data files.



Disk I/O contention is often a performance bottleneck. Oracle suggests:

- Putting redo log files on separate disks
- Separating index and data tablespace files

Summary

In this lesson you should have learned how to:

- **Troubleshoot invalid and unusable objects**
- **Gather optimizer statistics**
- **View performance metrics**
- **React to performance issues**

ORACLE

Practice 14:
Monitoring Performance

This practice covers the following:

- **Viewing performance metrics**
- **Repairing unusable indexes**
- **Manually collecting optimizer statistics**
- **Automating statistics collection**

ORACLE

Practice 14: Monitoring Performance

Background: User are complaining about slower than normal performance for operations involving the human resources and order entry applications. Upon questioning other members of the DBA staff you find that maintenance was recently performed on some of the tables belonging to the HR schema. Troubleshoot and make changes as appropriate to resolve the performance problems.

Tasks:

- View overall instance performance.
- Detect and repair unusable indexes.
- Manually collect statistics on the HR and OE schemas.
- Automate statistics collection using the scheduler.

1. Simulate a working load on your instance by running the script $HOME/labs/lab14_01.sql. Expect this script to take about 30 minutes to complete so run it in a separate terminal window and continue on with the lab while it runs.
   Note: Because this script generates a fairly heavy load in terms of CPU and disk I/O you will notice response time for Database Control is slower.

```
SQL> connect hr/hr@orcl
Connected.
SQL> @$HOME/labs/lab14_01.sql
```

2. Detect and repair unusable indexes.
   a) Connect to the database as user HR and run the script $HOME/labs/lab14_02_a.sql.

```
SQL> connect hr/hr@dba10g
Connected.
SQL> @$HOME/labs/lab14_02_a.sql
Table Altered.
```

   b) As user HR , attempt to add a new employee by running the script $HOME/labs/lab14_02_b.sql.

```
SQL> connect hr/hr@dba10g
Connected.
SQL> @$HOME/labs/lab14_02_b.sql
INSERT INTO hr.employees VALUES (301, 'John','JONES', ERROR
at line 1:
ORA-01502: index 'HR.EMP_EMP_ID_PK' or partition of such
index is in unusable state
```

   c) Using Enterprise Manager, check to see whether any other indexes in the HR schema are unusable.
   d) Reorganize any unusable indexes.
   e) Repeat step d for the remaining unusable indexes. After all indexes have been reorganized, attempt to add a new employee again by running the $HOME/labs/lab14_02_b.sql script.

```
SQL> @$HOME/labs/lab14_02_b.sql
1 row created.
SQL> commit;
Commit complete.
```

Practice 14: Monitor Performance (continued)

f) Why did the original attempt to insert a row in step 2.b fail?
g) What impact would an unusable index have on `SELECT` statements?
h) Were any objects other than indexes made not valid when the `employees` table was moved? If so, what do you need to do as an administrator to correct them?

3. Collect statistics on the `HR` and `OE` schemas.
4. Automate statistics collection so that statistics are collected for the `OE` and `HR` schemas each evening. Remember to check your job tomorrow morning to verify that it succeeded.
5. View overall instance performance.
   a) In Enterprise Manager navigate to the Performance page and investigate system performance.
   b) Click the Run Queue Length metric in the Host region. Notice that the three major indicators of system performance (CPU, Memory, and Disk I/O) are displayed graphically. Click the Disk I/O Utilization graph. For how long does the instance retain metric data on Disk I/O utilization?
   c) Return to the Database Performance page by clicking the Database tab at the top right of the page and selecting Performance. Drill-down into User I/O by clicking the User I/O label to the right of the Active Sessions: Waiting and Working region. Notice the bar graph labeled Top Files by Wait Count %. What happens if you move your cursor over one of the bars?
   d) Select the Top Objects page. Look at the pie chart labeled Top Objects By Wait Count %. Notice that as you move the shaded box on the timeline (top portion of the page) the objects that caused wait events during the selected time period change. Click one of the object numbers in the legend box. What happens?

15

Proactive Maintenance

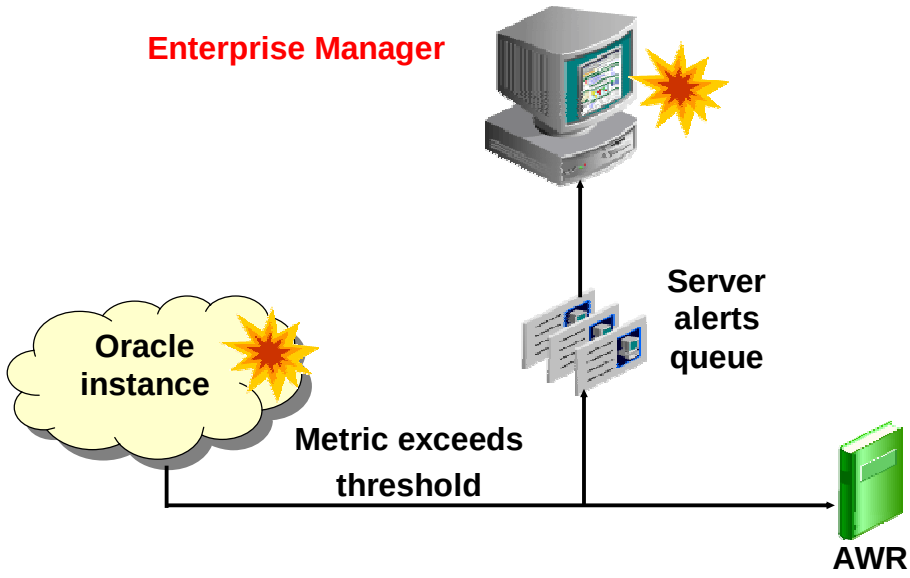
Objectives

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

- **Set warning and critical alert thresholds**
- **Collect and use baseline metrics**
- **Use tuning and diagnostic advisors**
- **Use the Automatic Database Diagnostic Monitor (ADDM)**
- **Manage the Automatic Workload Repository**

ORACLE

Server Generated Alerts



Server-Generated Alerts

Oracle Database 10*g* provides alerts about problems to the administrator via the Enterprise Manager Database Console. Enterprise Manager can be optionally configured to e-mail the administrator about problem conditions as well as display alert information on the console.

You can also set thresholds on many of the pertinent metrics for your system. Oracle Database 10*g* will proactively notify you if the database deviates from normal readings enough to reach those thresholds. Early notification of potential problems allows you to respond quickly, and often resolve issues before users even notice them.

A few key metrics that can provide early problem notification are:

- Average File Read Time (centi-seconds)
- Dump Area Used (%)
- Response Time (per transaction)
- SQL Response Time (%)
- Tablespace Used (%)
- Wait Time (%)

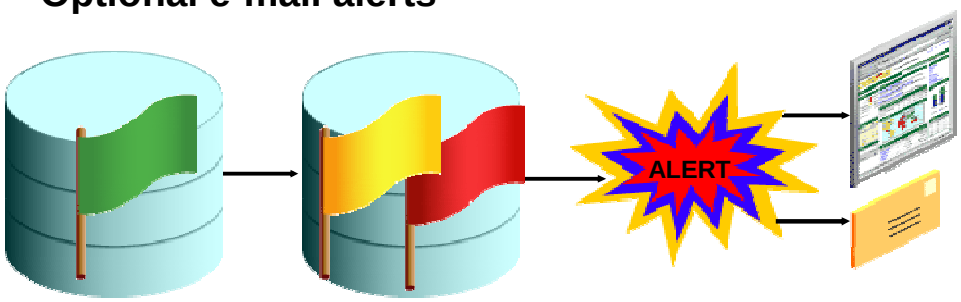
Thresholds

Each metric can be assigned two thresholds:

- **Warning**
- **Critical**

When thresholds are reached, alerts are triggered by:

- **Notifications that appear in the Alerts region of the Database Control home page**
- **Optional e-mail alerts**



ORACLE

Thresholds

Thresholds are boundary values against which monitored metric values are compared. You can specify warning and critical thresholds such that, when a monitored metric value crosses those thresholds, an alert is generated. This alert in turn notifies you of impending problems, which you can address in a timely manner.

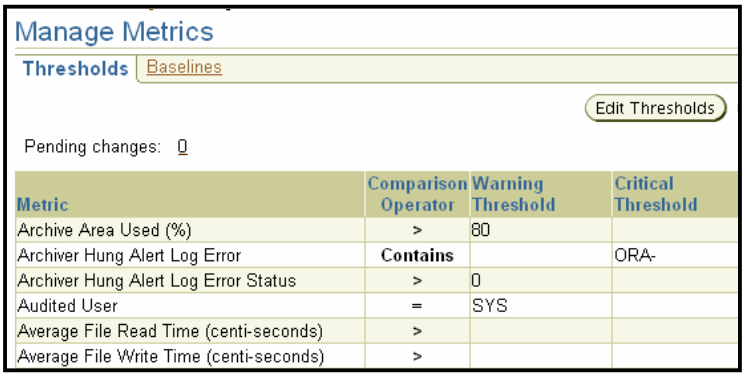
Warning thresholds are generally used to provide advance notification when performance trends start to deviate from normal activity. No action is usually required for performance warnings, they are meant to alert the DBA to begin monitoring the situation more carefully.

Critical alerts should be used to indicate problems that require immediate attention. Automated response actions can be established to activate when critical thresholds are reached.

The administrator's goal should be to solve problems before the critical thresholds are reached.

Setting Thresholds

Enterprise Manager's Manage Metrics property page provides access to threshold settings.



Setting Thresholds

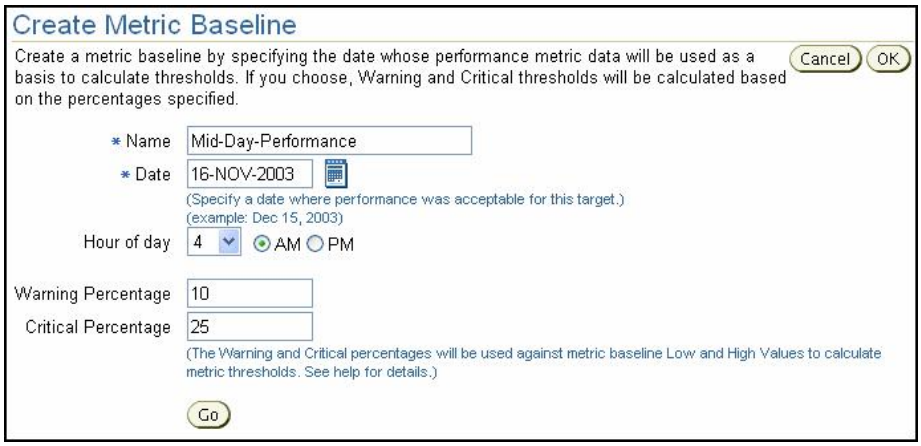
Click the Edit Threshold button from the Manage Metrics page to set warning or critical thresholds.

For some metrics you may choose to set more detailed targets. For example, you may want to set the metric Tablespace Space Used (%) differently for different tablespaces. Perhaps your transaction tablespaces should alert at 70% full but your read-only tablespaces are not a problem even when 95% full.

Detailed thresholds can be set by selecting the appropriate metric and clicking Manage Metric Indexes.

Baseline Measurements

Baseline measurements provide threshold recommendations based on actual performance data.



ORACLE

Baseline Measurements

You may want thresholds for performance metrics to be based on deviations from real target performance instead of absolute numbers. For example, if the database was running under a normal workload with acceptable performance on a given hour in given day, you may want to define thresholds such that you are warned if database performance becomes 10% worse than the performance on that given day, and receive a critical alert if performance degrades by 25%.

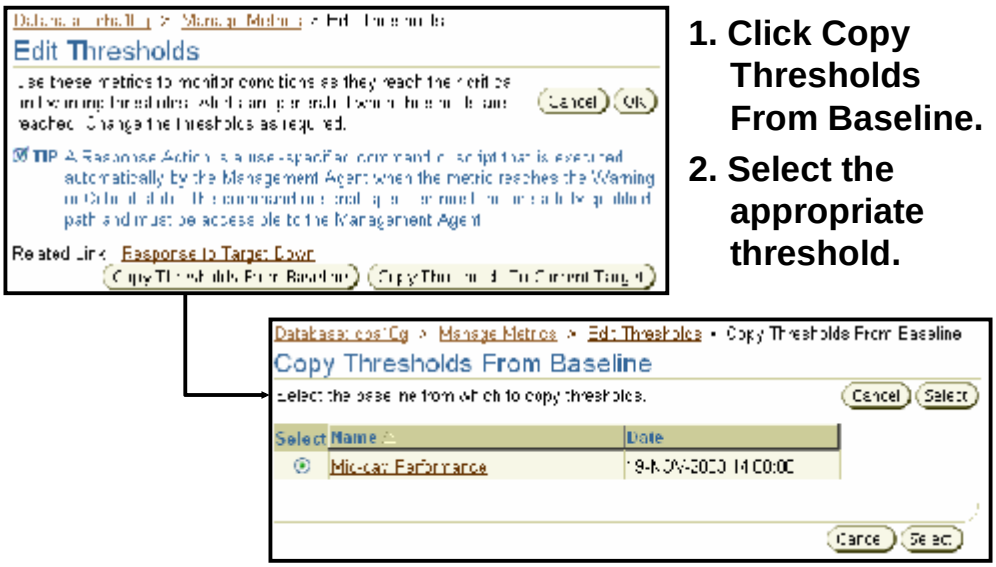
The Create Baseline Wizard allows you to capture average performance statistics for any given hour or, if you do not select an hour, for the entire 24-hour period indicated by the Date field. It will calculate warning and critical deviations from average performance.

To capture a baseline:

1. Invoke the Create Baseline Wizard from the Manage Metrics page.
2. Specify a date and optional hour when performance and workload were normal.
3. Provide default warning and critical percentage deviations and click the Go button.
4. Choose the metric or metrics you want to capture (the default is to capture a baseline measurement for all metrics). Threshold values will populate automatically using the deviation percentages entered in step 3. You can customize these if needed.
5. After any desired changes have been made to the calculated thresholds, click the OK button.

Using Baselines

To activate a stored baseline:



1. **Click Copy Thresholds From Baseline.**
2. **Select the appropriate threshold.**

ORACLE

Using Baselines

After you have created a baseline, you apply that baseline by clicking the Copy Thresholds From Baseline button in the Edit Thresholds page, then select the baseline you want to use.

Threshold values calculated from the baseline are applied for all metrics contained within the baseline. Of course, you can edit the threshold values for individual metrics if needed.

Tuning and Diagnostic Advisors

Oracle Database 10*g* provides several tuning and diagnostic advisors:

- **Automatic Database Diagnostic Monitor (ADDM)**
- **SQL Tuning and Access Advisors**
- **Memory Advisor**
- **Mean-Time-To-Recover (MTTR) Advisor**
- **Segment Advisor**
- **Undo Management Advisor**

ORACLE

Tuning and Diagnostic Advisors

Automatic Database Diagnostic Monitor: A server-based expert that reviews database performance every 60 minutes. ADDM's goal is to detect possible system bottlenecks early and recommend fixes before system performance degrades noticeably.

SQL Tuning Advisor: This advisor analyzes an individual SQL statement and makes recommendations for improving its performance. Recommendations might include actions such as rewriting the statement, changing the instance configuration, or adding indexes. The SQL Tuning Advisor is not invoked directly. Instead it is called from within other tools, such as Top SQL or Top Sessions, to help optimize high-impact SQL statements.

SQL Access Advisor: This advisor analyzes all SQL statements issued within a given period and suggests the creation of additional indexes or materialized views that will improve performance.

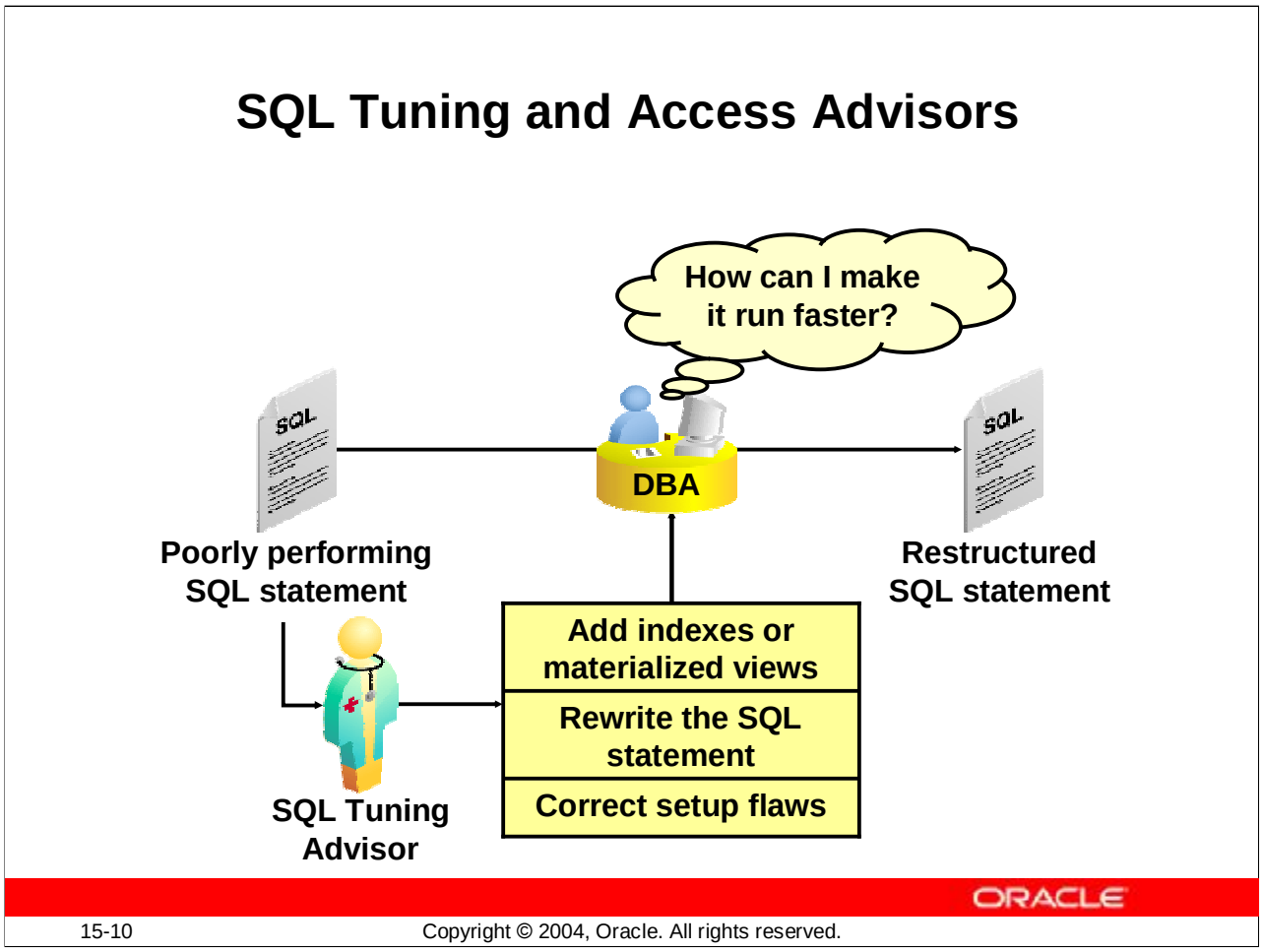
Memory Advisor: The memory advisor is actually a collection of several advisory functions that help determine the best settings for the shared pool, database buffer cache, and Program Global Area. In addition to the advisory functions, this page provides a central point of control for the large pool and the Java pool.

Tuning and Diagnostic Advisors (continued)

Mean-Time-To-Recover (MTTR) Advisor: The MTTR advisor assists you in setting the length of time required for the database to recover after an instance crash. MTTR and the use of this advisor will be covered more in later lessons.

Segment Advisor: This advisor looks for tables and indexes that are consuming more space than they require. The advisor checks for inefficient space consumption at the tablespace or schema level and produces scripts to reduce space consumption where possible.

Undo Management Advisor: The Undo Management Advisor helps you determine the undo tablespace size that is required to support a given retention period. Undo management and the use of the advisor will be covered in a later lesson.



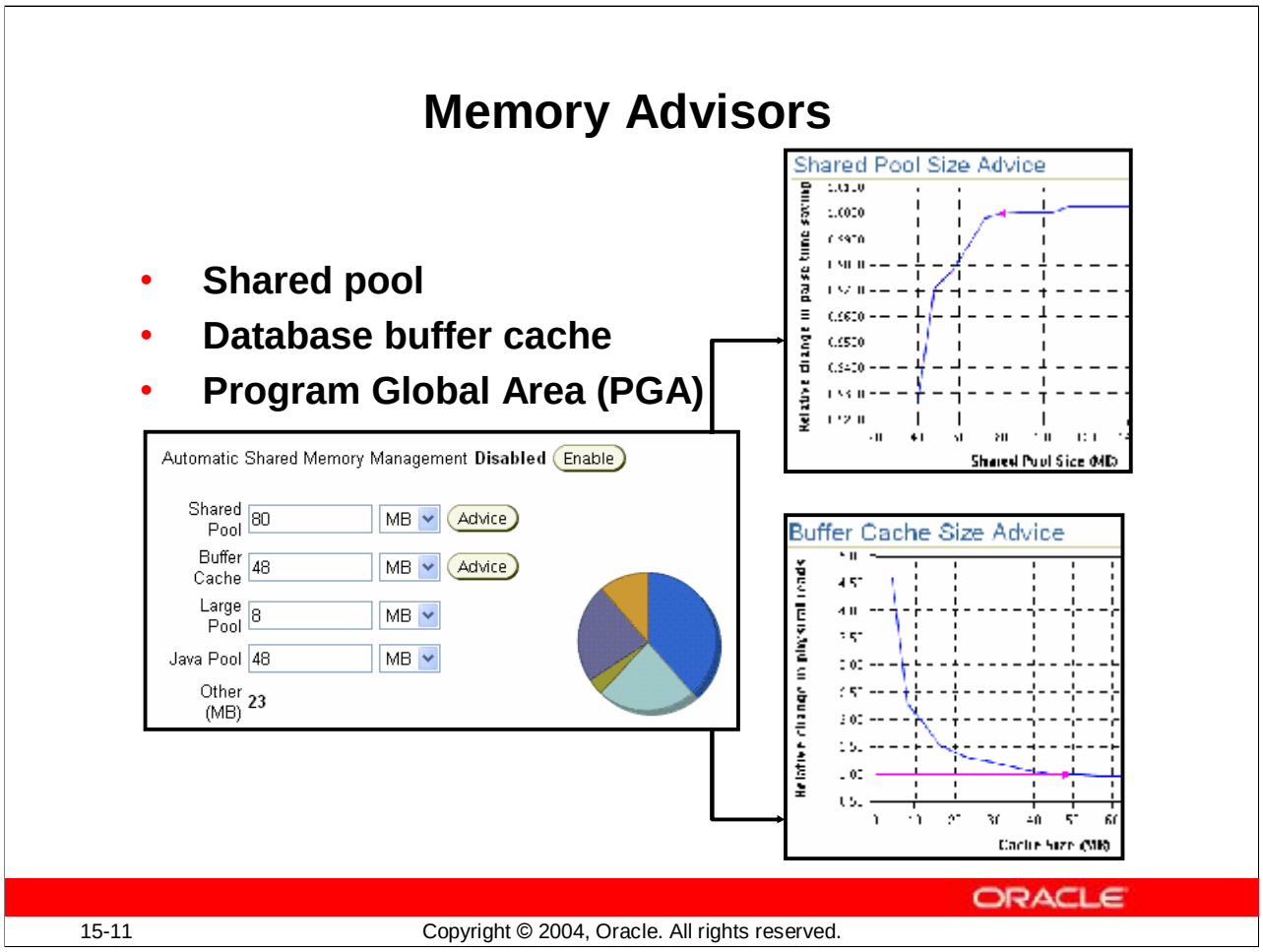
SQL Tuning and Access Advisors

The goal of SQL analysis is to help you identify high-impact SQL statements and make suggestions on how to improve their performance.

SQL statement performance can be improved by adding or changing indexing, by providing staged aggregate data as materialized views, by rewriting the SQL to be more efficient, and by correcting setup flaws such as improper initialization parameters.

Manually analyzing and tuning SQL statements is a time-consuming process that requires substantial expertise. The SQL tuning and access wizards help by:

- Making it easier to find high-impact SQL statements
- Recognizing common errors in SQL statement construction
- Recommending places where indexes or materialized views could help and informing the DBA of the cost of adding those objects



Memory Advisors

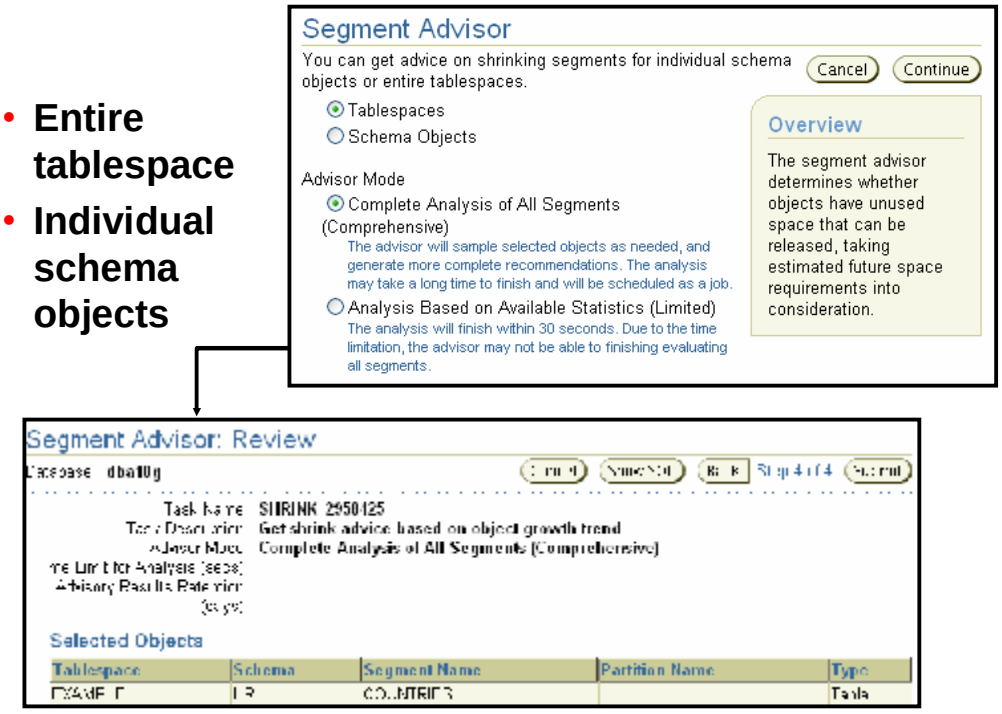
Sizing the SGA too small will severely degrade performance, but it is often difficult to know how much memory is enough. Conversely, most experts agree that there is little value in allocating additional memory to any of the individual SGA areas if that memory does nothing to improve performance. In fact, by consuming valuable memory resources needlessly, sizing the SGA too large can degrade system performance just as much as sizing it too small.

The memory advisors allow you to easily find and set your SGA components to the most efficient sizes. Each of the advisors presents the administrator with a graph showing the benefit of increased memory contrasted with the amount of memory required to achieve that benefit. The administrator should try to size the different cache areas somewhere near the “knee” of the curve (the point at which performance gains either level out or provide minimal performance improvement for the cost of the memory allocation).

Remember to balance the SGA and PGA needs with the overall system memory requirements. It does little good to increase the amount of memory allocated to the instance if in doing so you drive the server into swapping (using disk to simulate real memory).

Segment Advisor

- **Entire tablespace**
- **Individual schema objects**



ORACLE

Segment Advisor

When rows are deleted from a table, or when data within a row is updated to a value that consumes less space than the original, the space allocated to that row is returned to the table for future use.

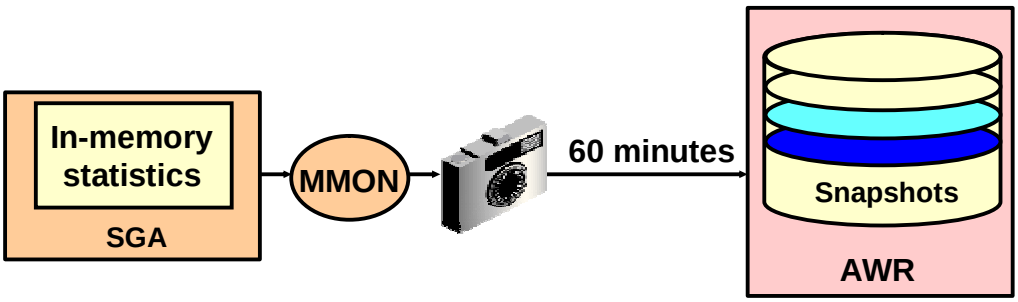
Over time this results in tables with more space allocated than needed. The segment advisor looks for tables or indexes with excess free space, and suggests ways to return that space to the rest of the database.

The segment advisor can be run using existing optimizer statistics (limited mode) or can actually scan the tables and indexes (comprehensive mode) to make the best recommendations. The analysis scope can be narrowed to individual tables and indexes, or can cover an entire tablespace.

Segment Advisor recommendations usually require that table properties be changed to enable row movement. This can be done through Enterprise Manager by opening the Administration properties page, clicking Tables, and editing the table you want to modify. Enable Row Movement appears on the Options tab of the Edit Table page. Alternatively, the following SQL command can be used:

```
SQL> ALTER TABLE owner.table_name ENABLE ROW MOVEMENT;
```

Automatic Workload Repository (AWR)



- Built-in repository of performance information
- Snapshots of database metrics are taken every 60 minutes and retained for 7 days
- Foundation for all self-management functions

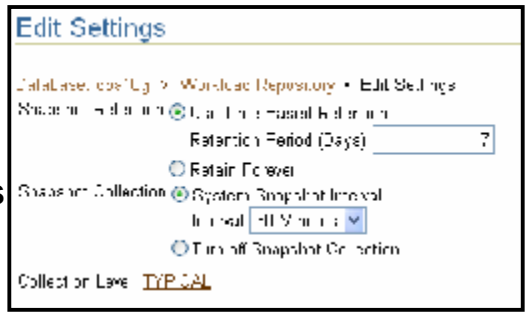
Automatic Workload Repository

By default, every 30 minutes, the database automatically captures statistical information from the SGA, and stores it inside the Automatic Workload Repository (AWR) in the form of snapshots. These snapshots are stored on disk and retained for seven days (both the snapshot interval and retention intervals can be modified by the administrator if desired).

The AWR contains hundreds of tables, all belonging to the `SYSMAN` schema and stored in the `SYSAUX` tablespace. Oracle does not support direct SQL access to the repository. Instead, use Enterprise Manager or the `DBMS_WORKLOAD_REPOSITORY` package to work with the AWR.

Managing the AWR

- **Retention period**
  - **Default 7 days**
  - **Consider storage needs**
- **Collection interval**
  - **Default 30 minutes**
  - **Consider storage needs, performance impact**
- **Collection level**
  - **Basic (disables most of ADDM functionality)**
  - **Typical (recommended)**
  - **All (adds additional SQL tuning information to snapshots)**



ORACLE

Managing the AWR

Open the Administration properties page and click the Workload Repository link to access the AWR settings page.

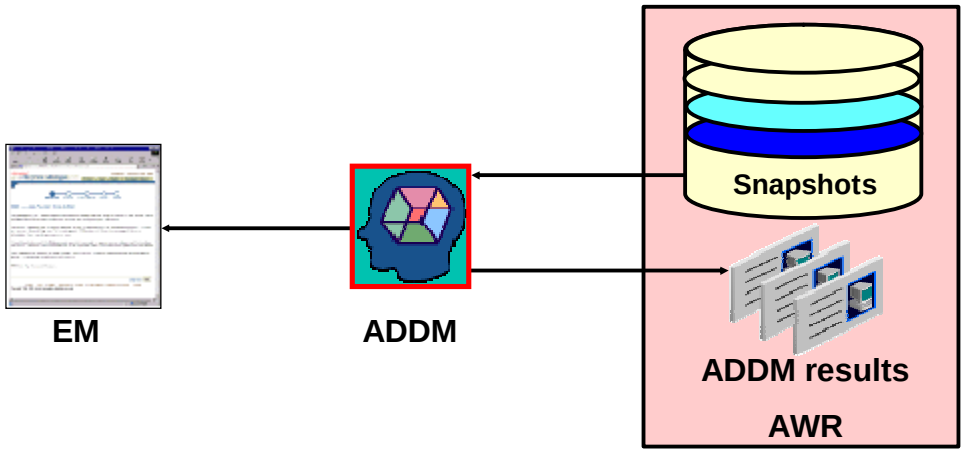
AWR settings include retention period, collection interval, and collection level. Remember that decreasing any of these settings impacts the functionality of components that depend on the AWR, including the advisors.

Increasing the settings can provide improved advisor recommendations, but does so at the cost of the space required to store the snapshots and the performance expended collecting the snapshot information.

Consider setting collection level to `ALL` when tuning a new application. The `ALL` setting collects SQL execution plans and timing statistics that enhance the recommendations of the SQL advisors. When tuning is complete, this setting should be returned to the `TYPICAL` setting.

Automatic Database Diagnostic Monitor (ADDM)

- Runs after each AWR snapshot
- Monitors the instance and detects bottlenecks
- Results stored within the AWR



Automatic Database Diagnostic Monitor (ADDM)

Unlike the other advisors, the ADDM runs automatically after each AWR snapshot. Each time a snapshot is taken, the ADDM does an analysis of the period corresponding to the last two snapshots. ADDM proactively monitors the instance, and detects most bottlenecks before they become a significant problem.

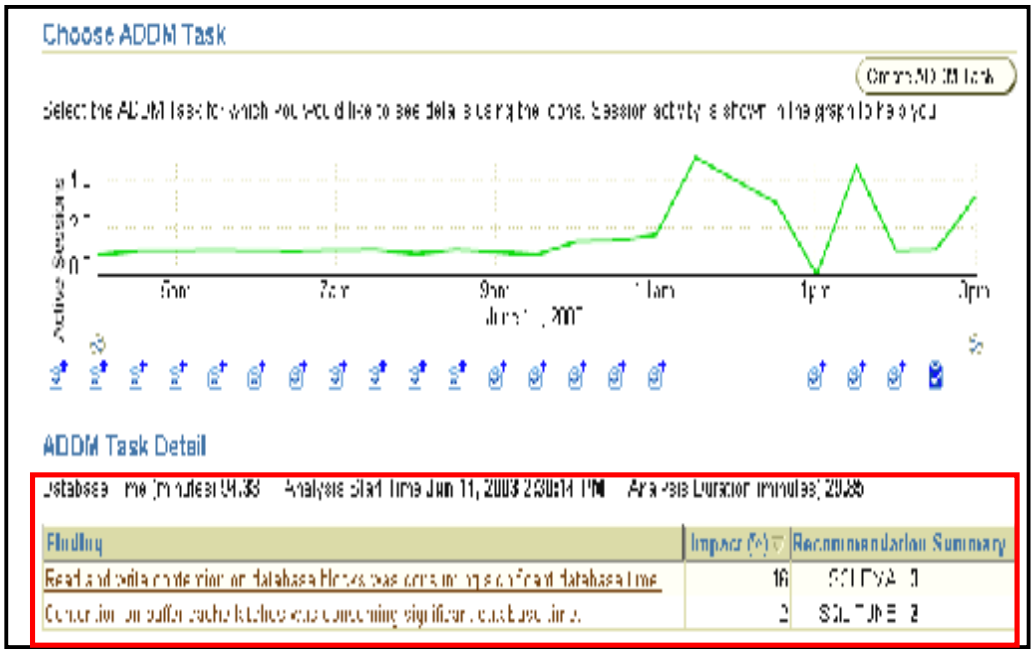
In many cases, ADDM recommends solutions for detected problems and even quantifies the benefits for the recommendations.

Some common problems detected by ADDM include:

- CPU bottlenecks
- Poor Oracle Net connection management
- Lock contention (locks will be discussed in a later lesson)
- I/O capacity
- Undersizing of Oracle memory structures
- High load SQL statements
- High PL/SQL and Java time
- High checkpoint load and cause; for example, small log files

The results of each ADDM analysis are stored inside the Automatic Workload Repository, and are also accessible through the Enterprise Manager console.

ADDM Findings

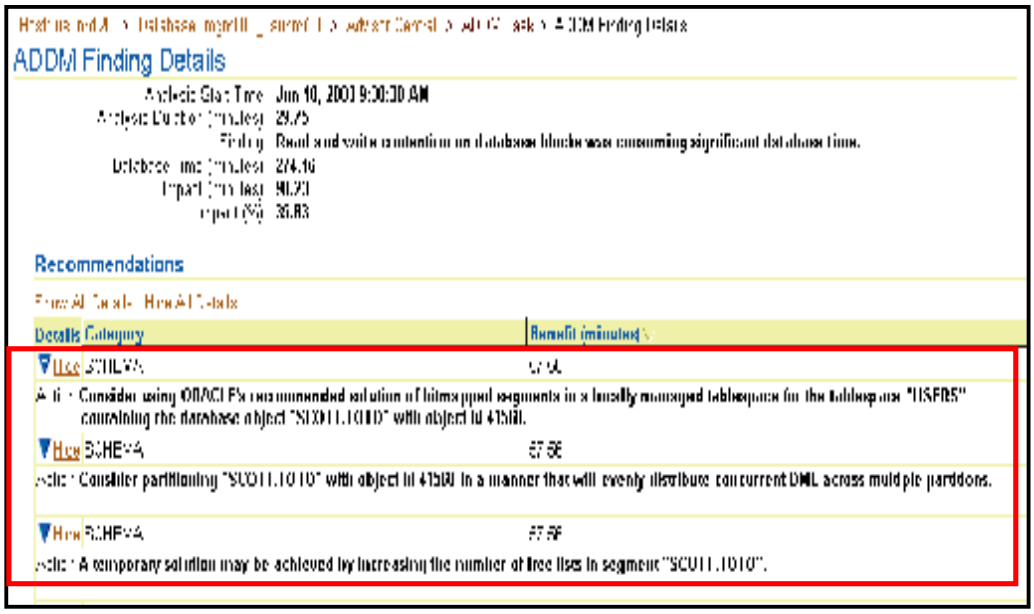


ADDM Findings

On the ADDM Findings page you can see the detailed findings for the latest ADDM run. Database Time represents the sum of the non-idle time spent by sessions in the database for the analysis period. A specific impact percentage is given for each finding. The impact represents the time consumed by the corresponding issue compared to the database time for the analysis period.

By clicking a particular finding, you are taken to the ADDM Finding Details page, which contains more information about the finding as well as any recommendations the ADDM may have generated.

ADDM Recommendations



ADDM Recommendations

On the ADDM Finding Details page, you are given some recommendations to solve the corresponding issue. Recommendations are categorized by SCHEMA, SQL Tuning, DB configuration, and many others. The Benefit column gives you the maximum reduction in database elapse time if the recommendation is implemented.

Summary

In this lesson you should have learned how to:

- Set warning and critical alert thresholds
- Collect and use baseline metrics
- Use tuning and diagnostic advisors
- Use the Automatic Database Diagnostic Monitor (ADDM)
- Manage the Automatic Workload Repository

Practice 15:
Proactive Maintenance

This practice covers configuring your database for proactive maintenance.

ORACLE

Practice 15: Proactive Maintenance

Background: You want to proactively monitor your database so that common problems can be fixed before they affect users.

Tasks:

- Configure your system to alert you if the application tablespace falls below 25% free space.
- Configure your system to alert you if free memory in the shared, large, or java pools falls below 10%.
- Use the Segment Advisor to detect and correct storage problems.
- Establish a baseline measurement for the Wait Time % metric.

1. Configure your system to alert you if the USERS tablespace falls below 25% free space
2. Configure your system to alert you when shared pool, large pool, or java pool free memory falls below 10%.
3. Use the segment advisor to detect and correct storage problems
4. Monitor the wait time percentage to establish a baseline.

16

Undo Management

Objectives

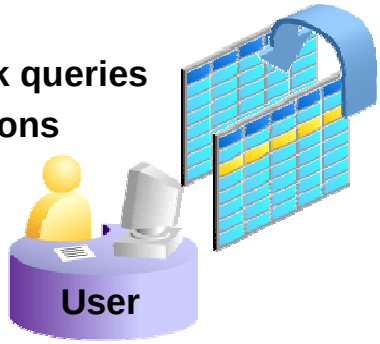
After completing this lesson you should be able to do the following:

- **Monitor and administer undo**
- **Configure undo retention**
- **Guarantee undo retention**
- **Use the Undo Advisor**

Undo Data

Undo data is:

- **A copy of original, premodification, data**
- **Captured for *every* transaction that changes data**
- **Retained at least until the transaction is ended**
- **Used to support:**
  - **Rollback operations**
  - **Read-consistent and flashback queries**
  - **Recovery from failed transactions**



ORACLE

Undo Data

Oracle saves the old value (undo data) when a process changes data in a database. It stores the data as it existed before being modified. Capturing undo data permits users to change their minds (roll back). Undo also supports read-consistent and flashback queries.

Read-consistent queries are queries, started before a data change, that finish sometime after a data change. Oracle provides results consistent with the data as of the time a query started. For a read-consistent query to succeed, the original information must still exist as undo information. As long as the undo information is retained, Oracle can reconstruct data to satisfy read-consistent queries.

Flashback queries are queries that purposefully ask for a picture of the data as it existed at some time in the past. As long as undo information for that past time still exists, the flashback queries can complete successfully.

Undo data is also used to recover from failed transactions. A failed transaction occurs when a user session ends abnormally (possibly due to network errors or a failure on the client machine) before the user decides to commit or roll back the transaction. Failed transactions may also occur when the instance crashes. In case of a failed transaction,

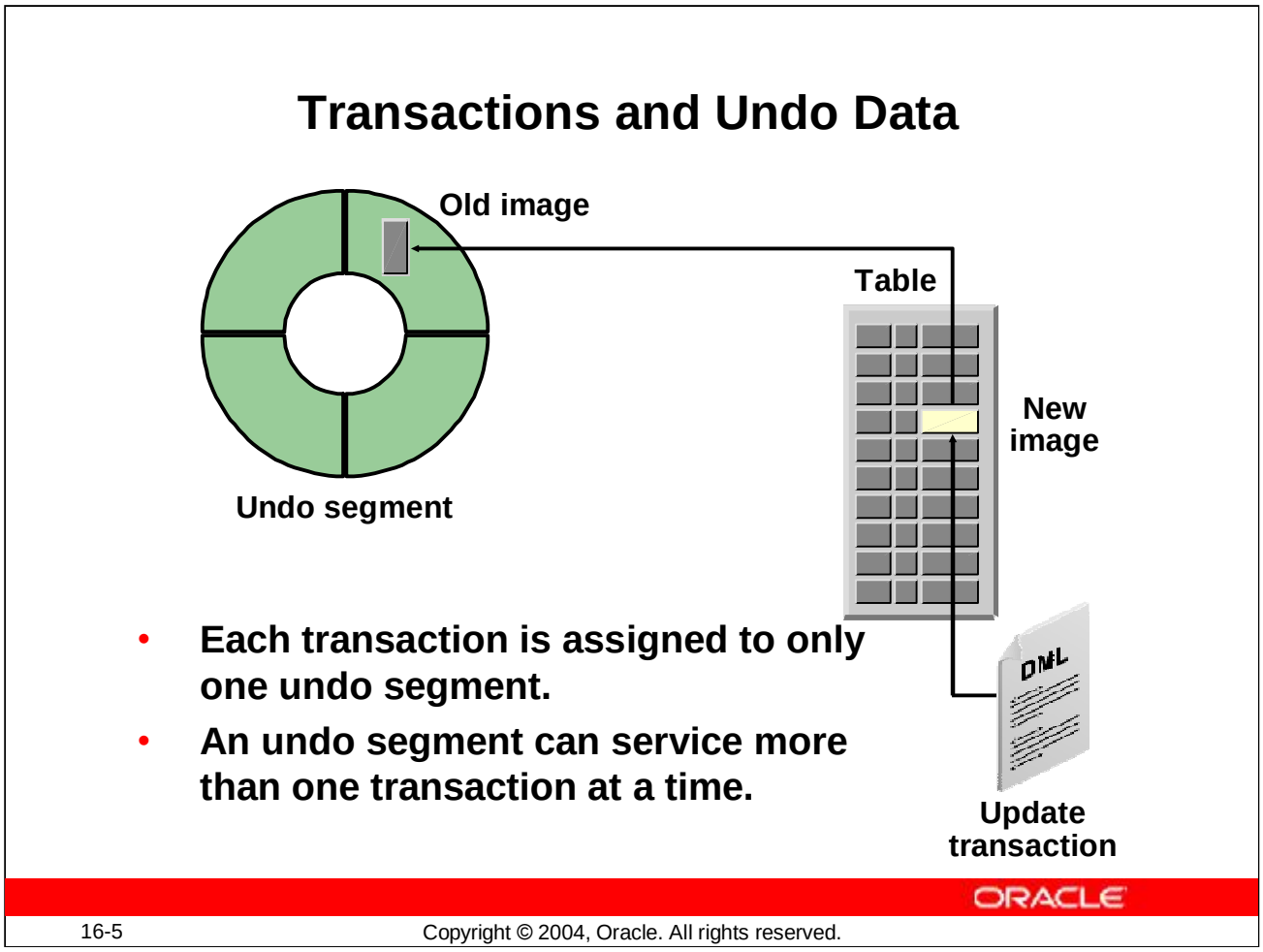
Undo Data (continued)

the safest behavior is chose and Oracle reverses all changes made by a user, restoring the original data.

Undo information is retained for all transactions at least until the transaction ends by:

- Users changing their mind (rolls back)
- Users ending a transaction (commits)
- User-session abnormally terminating (rolls back)
- User-session normally terminating with an exit (commits)

Undo information may be retained longer depending on the amount of database activity and the database configuration.



Transactions and Undo Data

When a transaction starts, that transaction is assigned to an undo segment. Throughout the life of the transaction, any changes made during the transaction will copy original values for the changed data to the undo segment. You can tell which transactions are assigned to which undo segments by checking the dynamic performance view `v$transaction`.

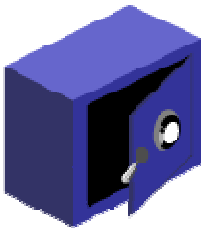
Undo segments are specialized segments that are automatically created by the instance as needed to support transactions. Like all segments, undo segments are made up of extents which in turn consist of data blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions.

Transactions fill extents in their undo segment extents until a transaction completes, or all space is consumed. If an extent fills up and more space is needed, the transaction will acquire that space from the next extent in the segment. Once all extents have been consumed, the transaction will either wrap around back into the first extent (overwriting undo data that is no longer needed) or request that a new extent be allocated to the undo segment (if unable to wrap around because the undo information in the first extent is still required).

Storing Undo Information

Undo information is stored in undo segments, which are in turn stored in an undo tablespace. Undo tablespaces:

- **Are only used for undo segments**
- **Have special recovery considerations**
- **May only be associated with a single instance, and an instance can only have one active undo tablespace at a time**



ORACLE

Storing Undo Information

Undo segments can exist only in a specialized form of tablespace called an undo tablespace. Whereas a database may have many undo tablespaces, each instance can open only one undo tablespace.

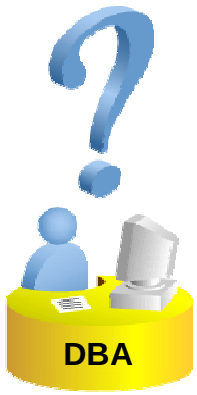
Undo segments have a segment type of "`TYPE 2 UNDO`" and are always owned by `SYS`. Because the segments act as a circular buffer, each segment will have a minimum of two extents. The default maximum number of extents depends on the database block size but is very high (32,765 for an 8 K block size).

Undo tablespaces are permanent, locally managed tablespaces with automatic extent allocation. They are managed like any other tablespace with the exception of recovery. Because undo data is required to recover from failed transactions (such as might occur when an instance crashes), undo tablespaces can be recovered only while the instance is in the `MOUNT` state. Recovery considerations for undo tablespaces will be covered in a later lesson.

Monitoring Undo

Undo usually requires little management. Areas to monitor include:

- **Undo tablespace free space**
- **"Snapshot too old" errors**



ORACLE

Monitoring Undo

Most of the time undo is managed automatically by the instance with little need for DBA intervention. A few things that may require administrator involvement include:

- Insufficient space for undo
- Users receiving `ORA-01555 snapshot too old` error messages

Undo information is always retained until a transaction ends. That means that if extremely large amounts of data are deleted or updated (insert operations consume very little undo space because the original image of inserted data is a null value) without periodically committing the changes, the undo tablespace must be large enough to contain the original data. Imagine a case where a 50 GB table had all rows deleted with the command:

```
SQL> DELETE FROM reallybigtable;
```

The undo tablespace would be required to make room for 50 GB of original information just in case the user who issued the above statement changed his or her mind and wanted to roll back the change. When the undo tablespace runs out of room for undo data, users receive an error message such as the following:

```
ORA-01650: unable to extend rollback segment
```

Note: Proactive monitoring, discussed in an earlier lesson, detects undo tablespace space problems before they affect users.

Monitoring Undo (continued)

Another possible problem the administrator may encounter with undo information occurs when a query needs to access undo information that has already been overwritten. This might happen in a long running or flashback query. When a query needs a "snapshot" of data as of some time in the past, and reconstructing that snapshot requires undo data that no longer exists, the query returns the following error:

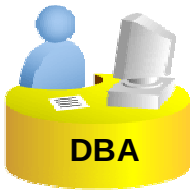
```
ORA-01555: snapshot too old
```

Administering Undo

Administration of undo should include preventing:

- **Undo tablespace space errors**
  - **Size the undo tablespace properly**
  - **Ensure large transactions commit periodically**
- **"Snapshot too old" errors**
  - **Configure an appropriate undo retention interval**
  - **Size the undo tablespace properly**
  - **Consider guaranteeing undo retention**

```
UNDO_MANAGEMENT=AUTO
UNDO_TABLESPACE=UNDOTBS1
```



ORACLE

Administering Undo

Oracle Database 10*g* recommends the use of automatic undo management, configured by setting the `UNDO_MANAGEMENT` initialization parameter to `AUTO`. Manual undo management is supported for backwards compatibility with Oracle 8*i* and earlier versions, but requires a great deal more DBA interaction.

With automatic undo management the DBA manages undo at the tablespace level, controlling which undo tablespace an instance uses with the `UNDO_TABLESPACE` initialization parameter. After selecting the undo tablespace the administrator need only worry about providing sufficient space and configuring an undo retention interval.

With manual management the DBA must also consider:

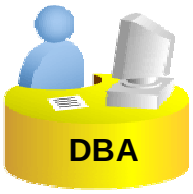
- Segment sizing including maximum extents and extent sizing
- Identifying and eliminating blocking transactions
- Creating enough rollback segments (in manual mode undo segments are known as rollback segments) to handle transactions
- Choosing a tablespace to contain the rollback segments (undo tablespaces are only used with automatic undo management)

Configuring Undo Retention

Undo retention specifies (in seconds) the amount of already committed undo information to retain.

- **Default value is 0 (automatic).**
- **Maximum value is 2^{32} seconds (more than 187 years).**
- **A setting of 0 indicates automatic undo retention mode.**

```
UNDO_RETENTION=0
```



ORACLE

Configuring Undo Retention

Undo retention is configured with the `UNDO_RETENTION` initialization parameter. This parameter sets the age at which undo information expires and may be overwritten as needed.

Automatically managed undo will attempt to retain undo information until it expires, but if an active transaction needs space in the undo tablespace, committed (unexpired) information will be overwritten regardless of the undo retention settings rather than allow a transaction to fail.

An undo retention setting of 0 turns on automatic undo retention tuning. In this mode the instance retains as much undo information as needed to satisfy the longest running query. If the undo tablespace is too small to satisfy the undo retention required by the current longest running query, automatic undo retention retains as much undo as possible using the current available space (without extending the undo data files) unless the amount of information retained drops below 900 seconds (15 minutes). Automatic undo retention keeps at least 15 minutes of undo information if possible without giving a space error.

Configuring Undo Retention (continued)

Undo information is divided into three categories:

- Uncommitted undo information: Supports a currently running transaction and is required if a user wants to roll back or if the transaction failed. Uncommitted undo information is never be overwritten.
- Committed undo information: No longer needed to support a running transaction, but still needed to satisfy the undo retention interval. Also known as “unexpired” undo information. Committed undo information is retained when possible without causing an active transaction to fail due to lack of space.
- Expired undo information: No longer needed to support a running transaction. Overwritten when space is required by an active transaction.

Guaranteeing Undo Retention

Committed undo information will be overwritten rather than cause transactions to fail for lack of undo space *unless* undo retention is "guaranteed."

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
Tablespace altered.

SQL> SELECT contents, retention
  2  FROM dba_tablespaces
  3  WHERE TABLESPACE_NAME='UNDOTBS1';
CONTENTS  RETENTION
--------- -----------
UNDO      GUARANTEE
```


Guaranteeing Undo Retention

The default undo behavior is to overwrite committed transactions that have not yet expired rather than allowing an active transaction to fail due to lack of undo space.

This behavior can be changed by guaranteeing retention. With guaranteed retention, undo retention settings are enforced even if they cause transactions to fail.

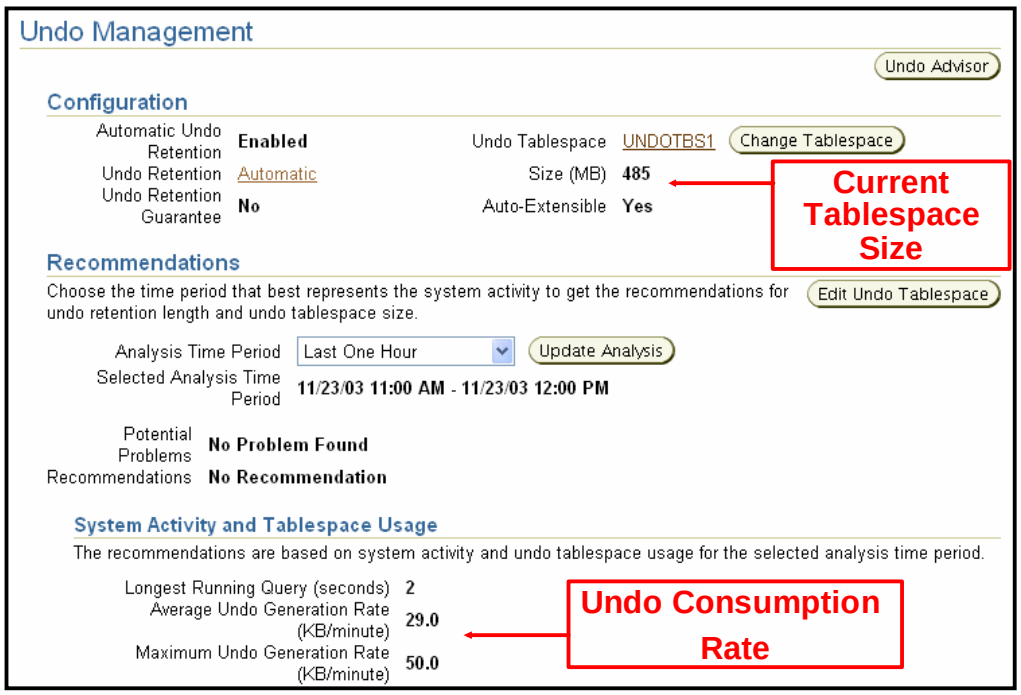
`RETENTION GUARANTEE` is a tablespace attribute rather than an initialization parameter. This attribute can be changed only with SQL command-line statements. The syntax to change an undo tablespace to guarantee retention is shown in the slide. To return a guaranteed undo tablespace to its normal setting, use the following command:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;
```

The retention guarantee applies only to undo tablespaces. Attempts to set it on a non-undo tablespace result in:

```
SQL> ALTER TABLESPACE example RETENTION GUARANTEE;
ERROR at line 1:
ORA-30044: 'Retention' can only specified for undo
tablespace
```

Sizing the Undo Tablespace

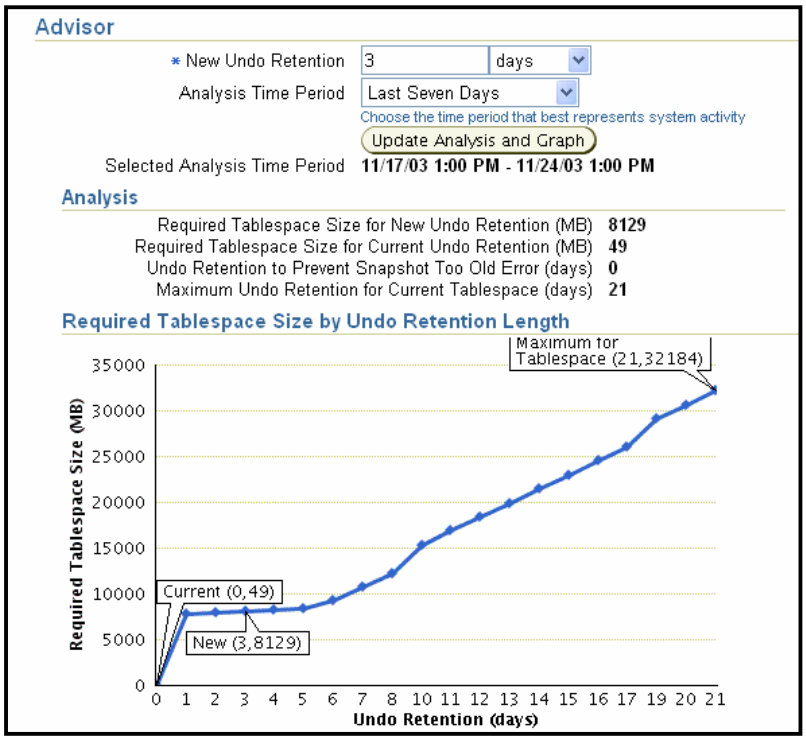


Sizing the Undo Tablespace

Undo tablespaces should be sized so that they can contain the original information for all transactions. Clicking the Undo Management link on Enterprise Manager's Administration properties page reveals an overview of system undo including current settings, undo consumption per minute, and the length of the longest running query observed during a given time period.

Data files belonging to an undo tablespace may be can to automatically extend when they run out of free space. Unlike other tablespaces, Oracle recommends that data files associated with undo tablespaces not have automatic extension enabled. When first determining undo space requirements, you may want to enable automatic extension of the data files, but after you have properly sized the tablespace you should disable it. Disabling automatic extension in an undo tablespace's data files prevents a single user from inadvertently consuming large amounts of disk space by neglecting to commit transactions.

Using the Undo Advisor



ORACLE

Using the Undo Advisor

The Undo Advisor is accessed through the Undo Management properties page. It provides an estimate of the undo tablespace size required to satisfy a given undo retention.

Enter the desired retention period and click the Update Analysis and Graph button. The analysis section of the advisor displays the tablespace size required to support the retention period.

You can also click a point on the graph to see the tablespace size required to support the selected period.

After you have selected an undo retention period, click OK to implement the new retention period.

Summary

In this lesson you should have learned how to:

- **Monitor and administer undo**
- **Configure undo retention**
- **Guarantee undo retention**
- **Use the Undo Advisor**

ORACLE

Practice 16:
Managing Undo

This practice covers performing typical undo management tasks including:

- **Calculating undo tablespace sizing to support a 48-hour retention interval**
- **Modifying undo tablespace to support a 48-hour retention interval**

ORACLE

Practice 16: Managing Undo

Background: A new version of your application includes several reports based on very long-running queries. Configure your system to support these reports

Tasks:

- Calculate the amount of undo space required to support a report that takes two days to run.
- Resize the undo tablespace to support the retention period required by the new reports.

1. Use the Undo Advisor to calculate the amount of undo space required to support a report that takes two days to run. What does the analysis recommend as the `Required Tablespace Size for New Undo Retention`?

   _____

2. Resize the undo tablespace to support the retention period required by the new reports (or 1GB, whichever is smaller).

17

Monitoring and Resolving Lock Conflicts

Objectives

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

- **Detect and resolve lock conflicts**
- **Manage deadlocks**

Locks

- **Prevent multiple sessions from changing the same data at the same time**
- **Automatically obtained at the lowest possible level for a given statement**

Transaction 1

```
SQL> UPDATE hr.employees
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

Transaction 2

```
SQL> UPDATE hr.employees
  2  SET salary=salary*1.1
  3 WHERE employee_id=100;
```

ORACLE

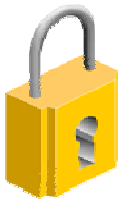
Locks

Before the database allows a session to modify data, the session must first lock the data being modified. A lock gives the session exclusive control over the data so that no other transaction can modify the locked data until the lock is released.

Transactions can lock individual rows of data, multiple rows, or even entire tables. Oracle Database 10*g* supports both manual and automatic locking. Automatically acquired locks always choose the lowest level of locking possible to minimize potential conflicts with other transactions.

Locking Mechanism

- High level of data concurrency
  - Row-level locks for inserts, updates, and deletes
  - No locks required for queries
- Automatic queue management
- Locks held until transaction ends (with commit or rollback operation)



Transaction 1

```
SQL> UPDATE hr.employees
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

Transaction 2

```
SQL> UPDATE hr.employees
  2  SET salary=salary*1.1
  3  WHERE employee_id=101;
```

ORACLE

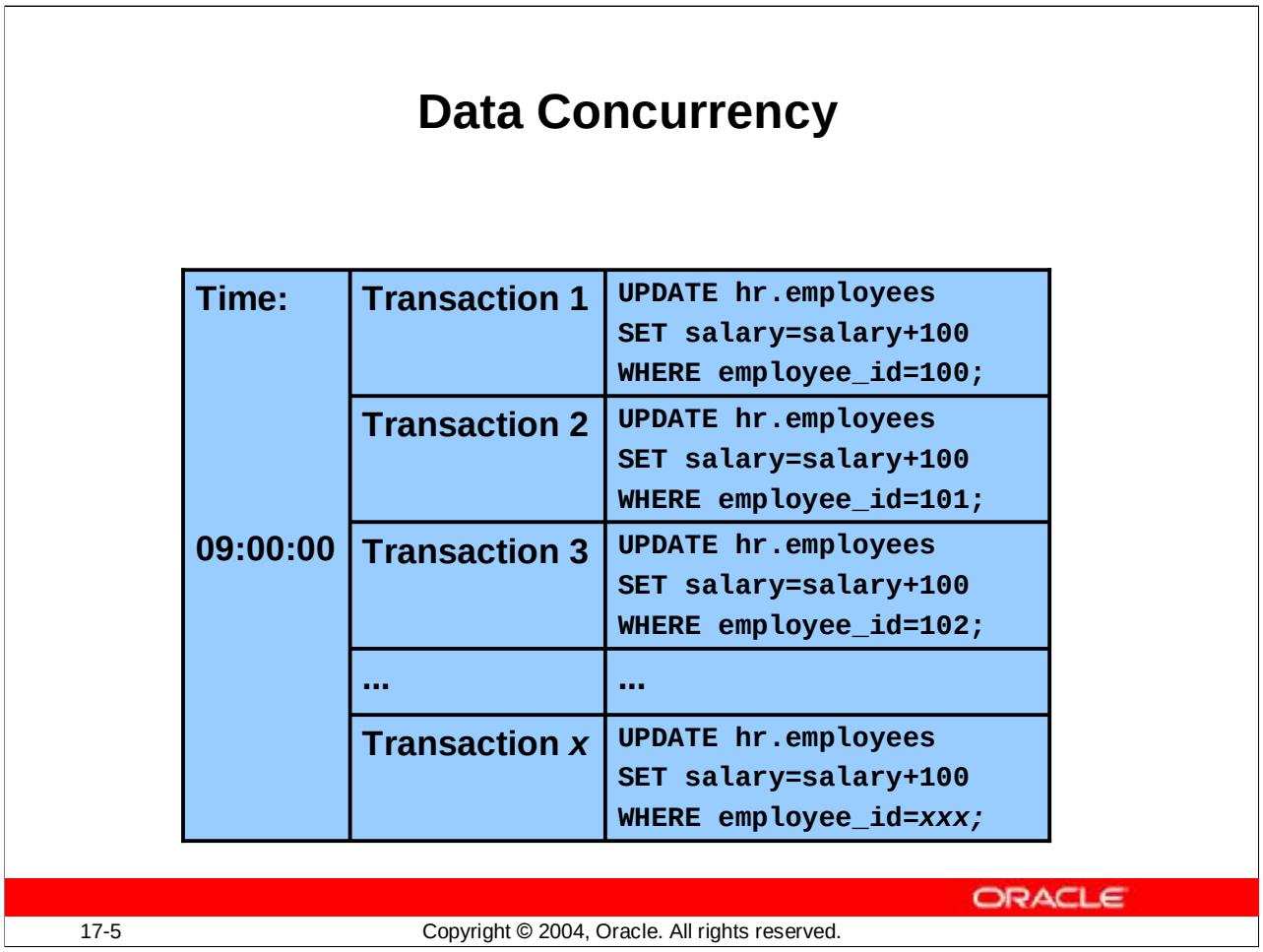
Locking Mechanism

The locking mechanism is designed to provide the maximum possible degree of data concurrency within the database. Transactions that modify data acquire row-level locks rather than page or table-level locks. Modifications to objects (such as table moves) obtain object level locks rather than whole database or schema locks.

Data queries do not require a lock, and a query succeeds even if someone has locked the data (always showing the original, prelock value reconstructed from undo information).

When multiple transactions need to lock the same resource, the first transaction to request the lock obtains it. Other transactions wait enqueue until the first transaction completes. The queue mechanism is automatic and requires no administrator interaction.

All locks are released at the end of a transaction. Transactions are completed when a commit or rollback is issued. In the case of a failed transaction, the same background process that automatically rolls back any changes from the failed transaction releases all locks held by the transaction.



Data Concurrency

The lock mechanism defaults to a fine-grained, row-level locking mode. Different transactions can be updating different rows within the same table without interfering with one other.

While the default model is to lock at the row level, Oracle Database 10*g* supports manual locking at higher levels if needed:

```
SQL> LOCK TABLE hr.employees IN EXCLUSIVE MODE;
Table(s) Locked.
```

With the above statement, any other transaction that tries to update a row in the locked table must wait enqueue until the transaction that issued the lock request completes. EXCLUSIVE is the strictest lock mode. Other lock modes are:

- ROW SHARE: Permits concurrent access to the locked table, but prohibits sessions from locking the entire table for exclusive access
- ROW EXCLUSIVE: The same as ROW SHARE, but also prohibits locking in SHARE mode. ROW EXCLUSIVE locks are automatically obtained when updating, inserting, or deleting data.
- SHARE: Permits concurrent queries but prohibits updates to the locked table. A SHARE lock is required (and automatically requested) to create an index on a table.

Data Concurrency (continued)

- SHARE ROW EXCLUSIVE: Used to query a whole table and to allow others to query rows in the table, but prohibits others from locking the table in SHARE mode or updating rows
- EXCLUSIVE: Permits queries on the locked table but prohibits any other activity on it. An exclusive lock is required to drop a table.

Like any request for a lock, manual lock statements wait enqueue until all sessions that either already have locks, or have previously requested locks, release their locks. The LOCK command accepts a special argument that controls the waiting behavior, NOWAIT.

NOWAIT returns control to you immediately if the specified table is already locked by another session.

```
SQL> LOCK TABLE hr.employees IN SHARE MODE NOWAIT;
LOCK TABLE hr.employees IN SHARE MODE NOWAIT
                  *
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT specified
```

DML Locks

Transaction 1

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 107;
1 row updated.
```

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 106;
1 row updated.
```

Each DML transaction must acquire *two* locks:

- **Row-exclusive lock for the row or rows being updated**
- **Shared table-level lock for the table containing the rows**

DML Locks

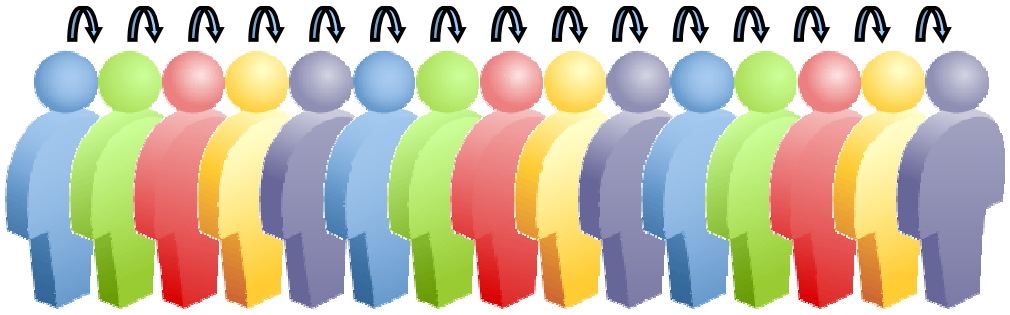
Each DML transaction obtains two locks:

- A row-exclusive lock on the row or rows being updated. There will be one row-exclusive lock regardless of the number of rows changed.
- A shared table-level lock on the table being updated. This is to prevent another session from locking the whole table (possibly to drop or truncate it) while the change is being made.

Enqueue Mechanism

The enqueue mechanism keeps track of:

- **Sessions waiting for locks**
- **The requested lock mode**
- **The order in which sessions requested the lock**



Enqueue Mechanism

Requests for locks are automatically queued. As soon as the transaction holding a lock completes, the next session in line receives the lock.

The enqueue mechanism tracks the order in which locks were requested and the requested lock mode.

Sessions that already hold a lock can request to *convert* that lock without having to go to the end of the queue. For example, suppose a session holds a shared lock on a table. The session can request to convert the shared lock to an exclusive lock. As long as no one else already has an exclusive or shared lock on the table, the session holding the shared lock will be granted an exclusive lock without having to wait in the queue again.

Lock Conflicts

Transaction 1	Time	Transaction 2
`UPDATE hr.employees SET salary=salary+100 WHERE employee_id=100;` `1 row updated.`	9:00:00	`UPDATE hr.employees SET salary=salary+100 WHERE employee_id=101;` `1 row updated.`

`UPDATE hr.employees SET COMMISION_PCT=2 WHERE employee_id=101;`
Session waits enqueue due to lock conflict.
	9:00:05 	`SELECT sum(salary) FROM hr.employees;`
		`SUM(SALARY)`
		`-----------`
		`692634`

Session still waiting!	16:30:00	Many selects, inserts, updates, and deletes during the last 7.5 hours, but no commits or rollbacks!
`1 row updated.` Session continues.	16:30:01	`commit;`

ORACLE

Lock Conflicts

Lock conflicts occur often, but are usually resolved through time and the enqueue mechanism. In certain rare cases a lock conflict may require administrator intervention. In the case above, transaction 2 obtains a lock on a single row at 9:00:00 and neglects to commit, leaving the lock in place. Transaction 1 attempts to update the entire table, requiring a lock on all rows, at 9:00:05. Transaction 1 is blocked by transaction 2 until transaction 2 commits at 16:30:01.

The user attempting to perform transaction 1 would almost certainly contact the administrator for help in this case, and the DBA would have to detect and resolve the conflict.

Possible Causes of Lock Conflicts

- **Uncommitted changes**
- **Long-running transactions**
- **Unnecessarily high locking levels**

ORACLE

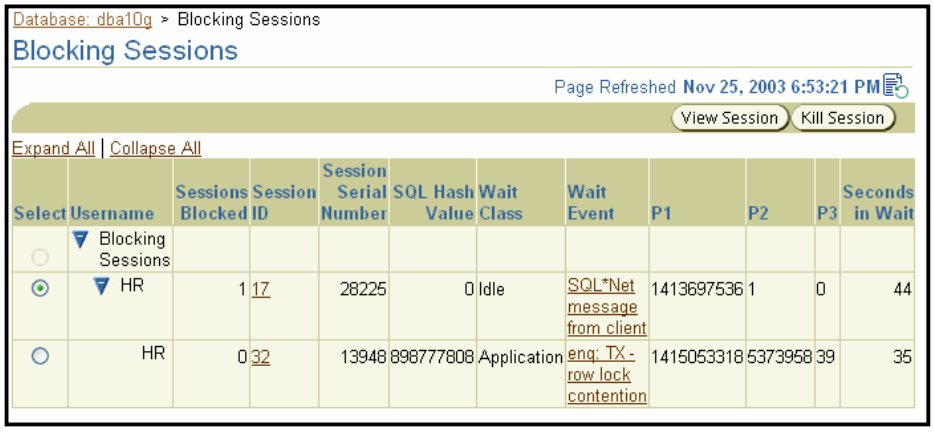
Possible Causes of Lock Conflicts

The most common cause of lock conflict is an uncommitted change, but there are a few other possible causes:

- Long-running transactions. Many applications use batch processing to perform bulk updates. These batch jobs are usually scheduled for times of low or no user activity, but in some cases may not have finished or may take too long to run during the low activity period. Lock conflicts are common when transaction and batch processing are being performed simultaneously.
- Unnecessarily high locking levels. Not all databases support row-level locking (Oracle added support for row-level locks in 1988 with release 6). Some databases still lock at the page or table level. Developers writing applications intended to run on many different databases often write their application with artificially high locking levels so that the Oracle database behaves similar to these less capable databases. Developers new to Oracle also sometimes unnecessarily code in higher locking levels than required by Oracle Database 10*g*.

Detecting Lock Conflicts

Select Blocking Sessions from the Performance page.



Click the Session ID link to view information about the locking session, including the actual SQL statement.

Detecting Lock Conflicts

Use the Blocking Sessions page in Enterprise Manager to locate lock conflicts. Conflicting lock requests are shown in hierarchical layout with the session holding the lock at the top, and sessions enqueue for the lock show in order below.

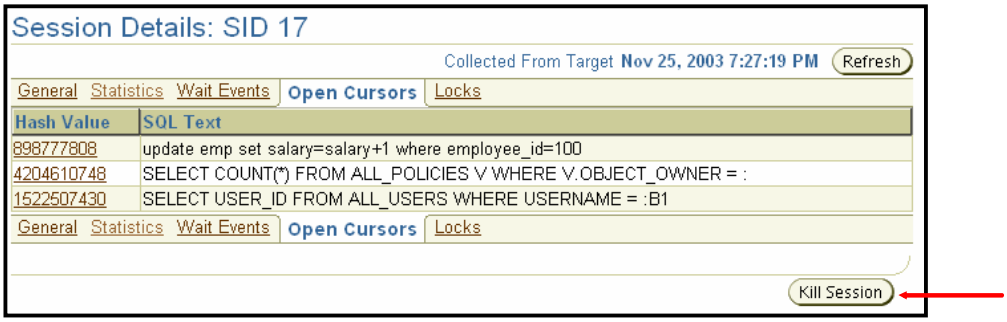
For each session involved in the conflict you are given the username, session ID, and the seconds the session has been waiting. Drill down into the session ID to see actual SQL statements executed or being requested by the session.

ADDM will also automatically detect lock conflicts and can advise you of inefficient locking trends.

Resolving Lock Conflicts

To resolve a lock conflict:

- **Have the session holding the lock commit or roll back.**
- **Terminate the session holding the lock as a last resort.**



Resolving Lock Conflicts

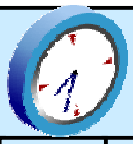
To resolve a lock conflict, the session holding the lock must release it. The best way to have the session release the lock is to contact the user and ask that the transaction be completed.

In an emergency, it is possible for the administrator to terminate the session holding the lock by clicking the Kill Session button. Keep in mind that when a session is killed all work within the current transaction is lost (rolled back). Users whose session are killed must log in again and redo all work since their commit.

Users whose sessions have been killed will receive the following error the next time they try to issue an SQL statement:

```
ORA-00028: your session has been killed
```

Deadlocks

Transaction 1		Transaction 2

`UPDATE employees`
`SET salary = salary x 1.1`
`WHERE employee_id = 1000;`
	9:00	`UPDATE employees`
		`SET manager = 1342`
		`WHERE employee_id = 2000;`

`UPDATE employees`
`SET salary = salary x 1.1`
`WHERE employee_id = 2000;`
	9:15	`UPDATE employees`
		`SET manager = 1342`
		`WHERE employee_id = 1000;`

`ORA-00060:`
`Deadlock detected while`
`waiting for resource`
	9:16	

ORACLE

Deadlocks

A deadlock is a special example of a lock conflict. Deadlocks arise when two or more sessions wait for data locked by each other. Because each is waiting on the other, neither can complete their transaction to resolve the conflict.

The Oracle server automatically detects and resolves deadlocks by rolling back the statement that detected the deadlock.

In the example above, transaction 1 must resubmit the update from 9:15, but the update issued at 9:00 does not need to be resubmitted.

Summary

In this lesson you should have learned how to:

- **Detect and resolve lock conflicts**
- **Manage deadlocks**

Practice 17:
Locks in the Oracle Database

This practice covers common administrative tasks relating to locks in Oracle Database 10*g*, including:

- **Detecting which session is causing the locking conflict**
- **Resolving locking conflicts**

ORACLE

Practice 17: Locks in the Oracle Database

Background: The Help desk just received a call from Susan Mavris, an HR representative complaining that the database is "frozen." Upon questioning the user, you find that she was trying to update John Chen's personnel record with his new phone number, but when she entered the new data her session froze and she couldn't do anything else.

Tasks:

- Detect which session is causing the locking conflict.
- Resolve the conflict in favor of the session that complained.

1. Set up a lock conflict by running the SQL script `$HOME/labs/lab17_01.sql`. Do not worry if the session seems to "hang"—this is the condition we are trying to create.

```
SQL> @$HOME/labs/lab17_01.sql
```

2. Detect which session is causing the locking conflict.
3. Resolve the conflict in favor of the session that complained.

18

Backup and Recovery Concepts

Objectives

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

- Describe the basics of database backup, restore and recovery.
- List the types of failure that may occur in an Oracle Database.
- Describe ways to tune instance recovery.
- Identify the importance of checkpoints, redo log files, and archived log files.
- Configure `ARCHIVELOG` mode.

ORACLE

Backup and Recovery Issues

The administrator's duty is to:

- **Protect the database from failure wherever possible.**
- **Increase the Mean-Time-Between-Failures (MTBF).**
- **Decrease the Mean-Time-To-Recover (MTTR).**
- **Minimize the loss of data.**

ORACLE

Backup and Recovery Issues

The DBA's goal is to ensure the database is open and available when users need it. In support of that goal, the DBA, usually working with the system administrator:

- Anticipates and works against common causes of failure
- Works to increase the mean-time-between-failure, ensuring hardware is as reliable as possible, that critical components are protected by redundancy, and that operating system maintenance is performed in a timely manner. Oracle provides advanced configuration options to increase MTBF including:
  - Real Application Clusters (discussed in another course)
  - Streams (discussed in another course)
- Decreases the mean-time-to-recover, practicing recovery procedures in advance and configuring backups so they are readily available when needed
- Minimizes the loss of data. DBAs who follow accepted best practices can configure their databases so that no committed transaction is ever lost. Tools to assist in guaranteeing this include:
  - Archived redo logs (discussed later in this lesson)
  - Standby databases and Oracle Data Guard (discussed in another course)

Categories of Failures

Failures can generally be divided into the following categories:

- **Statement failure**
- **User process failure**
- **Network failure**
- **User error**
- **Instance failure**
- **Media failure**

ORACLE

Categories of Failures

Failures can be divided into a few broad categories:

- Statement failure: A single database operation (select, insert, update, delete) fails.
- User process failure: A single database session fails.
- Network failure: Connectivity to the database is lost.
- User error: A user successfully completes an operation, but the operation was incorrect (dropping a table, entering incorrect data).
- Instance failure: The database instance shuts down unexpectedly.
- Media failure: One or more of the database files are lost (deleted, failed disk).

Statement Failures

Typical Problems	Possible Solutions
Attempts to enter invalid data into a table	Work with users to validate and correct data.
Attempts to perform operations with insufficient privileges	Provide appropriate object or system privileges.
Attempts to allocate space that fail	Enable resumable space allocation. Increase user quota. Add space to tablespace.
Logic errors in applications	Work with developers to correct program errors.

Statement Failures

When a single database operation fails, DBA involvement may be needed to correct errors with user privileges or database space allocation.

User Process Failure

Typical Problems	Possible Solutions
User performed an abnormal disconnect.	DBA action is not usually needed to resolve user process failures. Instance background processes roll back uncommitted changes and release locks. Watch for trends.
User's session was abnormally terminated.	
User experienced a program error which terminated the session.	

User Process Failure

Users who are abnormally disconnected from the instance may have uncommitted work in progress that needs to be cleaned up. The PMON background process periodically polls server processes to ensure that their sessions are still connected. If PMON finds a server process whose user is no longer connected, PMON recovers from any ongoing transactions, including rolling back uncommitted changes and releasing any locks held by the failed session.

DBA intervention should not be required to recover from user process failure but the administrator should watch for trends. One or two users disconnecting abnormally is not a cause for concern. A small percentage of user process failures is normal. Consistent and systemic failures indicate other problems. A large percentage of abnormal disconnects may indicate a need for user training (teach them to log out rather than just terminating their programs). It may also be indicative of network or application problems.

Network Failure

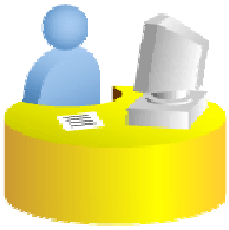
Typical Problems	Possible Solutions
Listener fails	Configure a backup listener and connect-time failover.
Network Interface Card (NIC) fails	Configure multiple network cards.
Network connection fails	Configure a backup network connection.

Network Failure

The best solution to network failures is to provide redundant paths for network connections. Backup listeners, network connection, and network interface cards reduce the chance of network failures affecting system availability.

User Errors

Typical Causes	Possible Solutions
User inadvertently deletes or modifies data.	Roll back or use flashback query to recover.
User drops a table.	Recover table from recycle bin.



User Errors

Users may inadvertently delete or modify data. When that happens, the DBA may need to assist the user in recovering from the error. If the user has not yet committed or exited their program, they can simply roll back their operation. If the user has already committed the changes, flashback queries can be used to determine what the previous values were (and then the data can be updated to restore the original information.)

```
SQL> SELECT salary FROM employees WHERE employee_id=100;
SALARY
------
25
SQL> SELECT salary FROM employees
2    AS OF TIMESTAMP(SYSTIMESTAMP-INTERVAL'10' minute)
3    WHERE employee_id=100;
SALARY
------
24000
```

In cases where flashback queries are not possible because the undo retention period has been exceeded, the DBA may still be able to recover the original information through the use of Oracle LogMiner.

User Errors (continued)

Oracle LogMiner allows you to query your online redo logs and archived redo logs through an SQL interface. Transaction data may persist in online redo logs longer than it does in undo, and if you have configured archiving of redo information redo persists until you delete the archived files.

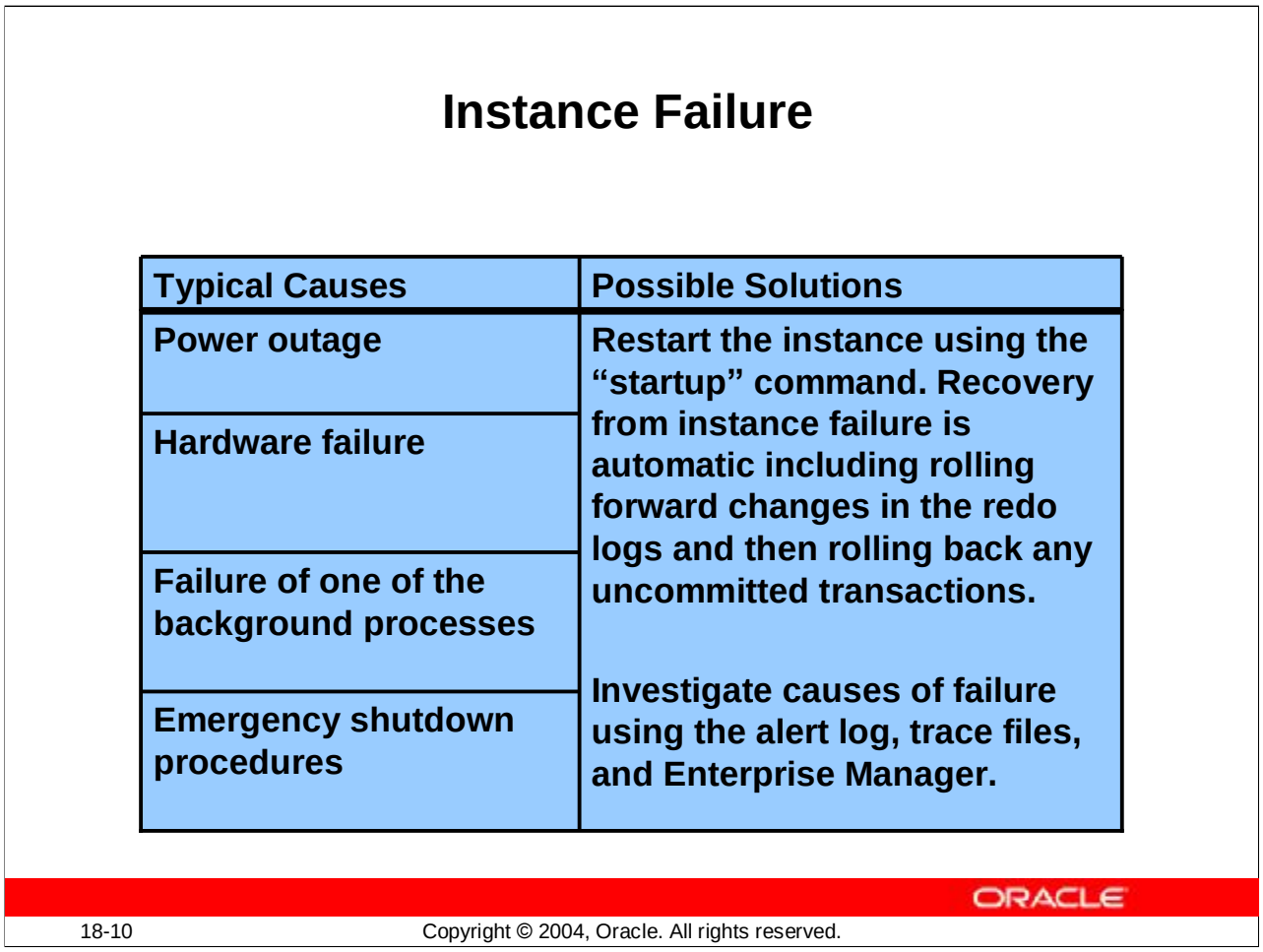
Oracle LogMiner is discussed in the *Oracle Database 10g: Administration Workshop II* course and in the *Oracle Database: Utilities* reference manual.

Users who drop tables can recover those tables from the recycle bin by flashing the table back to before the drop.

```
SQL> DROP TABLE hr.job_history;
Table dropped.

SQL> SELECT COUNT(*) FROM hr.job_history;
SELECT COUNT(*) FROM hr.job_history
              *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> FLASHBACK TABLE hr.job_history TO BEFORE DROP;
Flashback complete.

SQL> SELECT COUNT(*) FROM hr.job_history;
  COUNT(*)
----------
        10
```

If the recycle bin has already been purged, or if the user dropped the table with the PURGE option, the dropped table can *still* be recovered by using point-in-time recovery (PITR) if the database has been properly configured.

PITR is discussed in the *Oracle Database 10g: Administration Workshop II* course and in the *Oracle Database: Backup and Recovery Advanced User's Guide.*



Instance Failure

Instance failure occurs when the database instance is shut down before synchronizing all database files. An instance failure can occur due to hardware or software failure, or through the use of the emergency shutdown commands `SHUTDOWN ABORT` and `STARTUP FORCE`.

Administrator involvement in recovering from instance failure is usually limited to restarting the instance and working to prevent future occurrences.

Instance Recovery

Instance or crash recovery:

- **Is caused by attempts to open a database whose files were not synchronized on shutdown**
- **Is automatic**
- **Uses information stored in redo log groups to synchronize files**
- **Involves two distinct operations**
  - **Rolling forward: Data files are restored to their state before the instance failed.**
  - **Rolling back: Changes made but not committed are returned to their original state.**

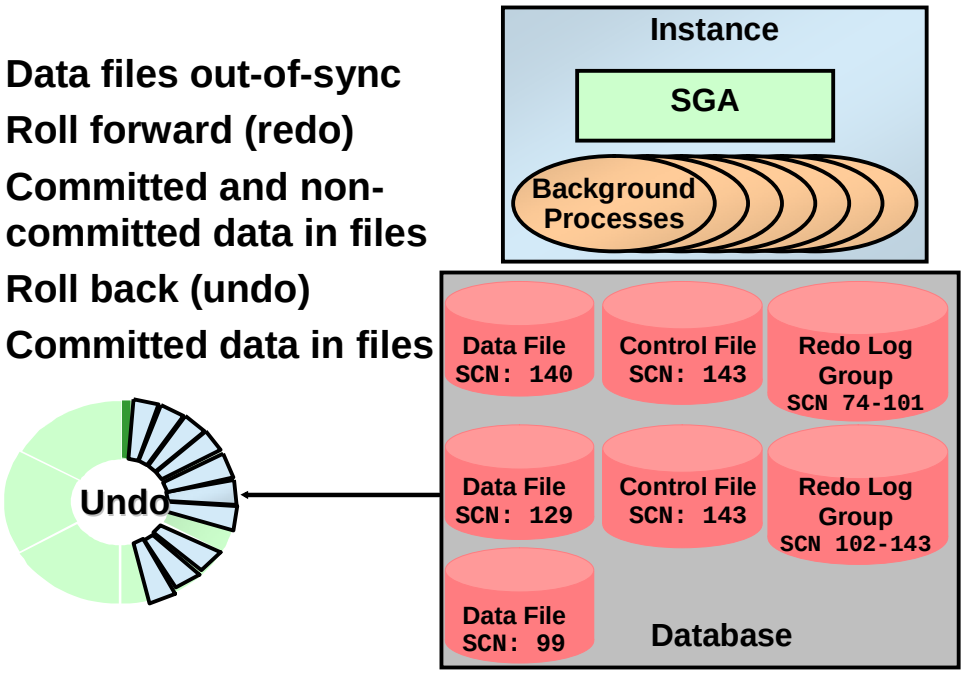
ORACLE

Instance Recovery

The Oracle Database 10*g* automatically recovers from instance failure. All the DBA needs to do is start the instance normally. The instance will mount the control files and then attempt to open the data files. When it discovers the data files have not been synchronized during shutdown, the instance uses information contained in the redo log groups to roll the data files forward to the time of shutdown and then (because the undo tablespace was also rolled forward) roll back any uncommitted transactions.

Phases of Instance Recovery

1. Data files out-of-sync
2. Roll forward (redo)
3. Committed and non-committed data in files
4. Roll back (undo)
5. Committed data in files



Phases of Instance Recovery

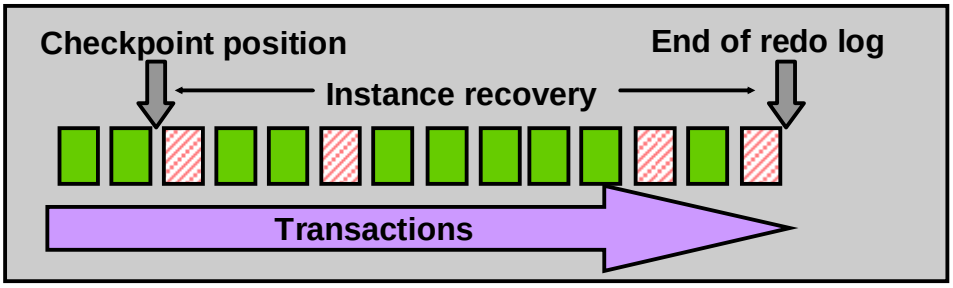
In order for an instance to open a data file, the system change number (SCN) contained within the data file's header must match the current SCN stored in the database's control files.

If the numbers do not match, the instance applies redo from the online redo logs, sequentially "redoing" transactions until the data files are up-to-date. After all data files have been synchronized with the control files, the database is opened and users may now log in.

When redo was applied, *all* transactions were applied to bring the database up to the state as of the time of failure. This usually includes transactions that were in progress but had not yet been committed. After the database has been opened, those uncommitted transactions are rolled back. At the end of the rollback phase of instance recovery, the data files will contain only committed data.

Tuning Instance Recovery

- During instance recovery the transactions between the checkpoint position and end of redo log must be applied to the data files.
- Tune instance recovery by controlling the difference between the checkpoint position and end of redo log.



Tuning Instance Recovery

Transaction information is always recorded in the redo log groups before the instance returns `commit complete` for a transaction. The information in the redo log groups guarantees that the transaction can be recovered in case of a failure. That same transaction information also needs to be written to the data file. The data file write usually happens sometime after the information is recorded in the redo log groups because the data file write process is much slower than the redo writes (random writes for data files are slower than serial writes for redo log files).

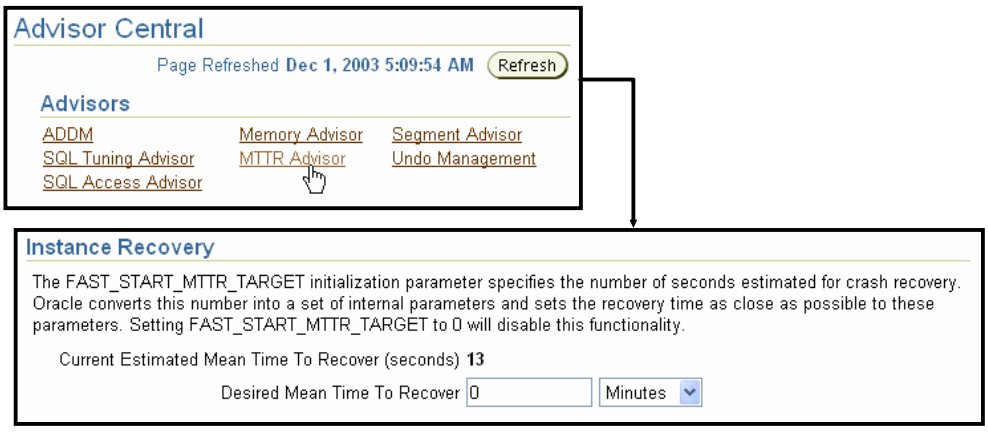
To keep track of what has already been written to the data files, the database uses checkpoints. A checkpoint guarantees that as of the time the checkpoint occurs, all data up to a certain SCN is recorded in the data file. Transactions after the checkpoint position may or may not have yet been written to the appropriate data file. In the graphic above, the striped blocks have not yet been written to disk.

The time required for instance recovery is the time required to bring the data files from their last checkpoint to the latest SCN recorded in the control file. The administrator controls that time by setting a MTTR target (in seconds) and through the size of the redo log groups.

The distance between the checkpoint position and the end of the redo log group can never be more than 90% of the smallest redo log group.

Using the MTTR Advisor

- **Specify the desired time in seconds or minutes.**
- **Default value is 0 (disabled).**
- **Maximum value is 3600 seconds (one hour).**



MTTR Advisor

Click the MTTR Advisor from Enterprise Manager's Advisor Center for assistance in setting the MTTR target. The advisor allows you to specify a desired mean-time-to-recover and translates that into settings for the FAST_START_MTTR_TARGET initialization parameter.

The default setting of 0 disables the MTTR target, reducing the likelihood that writes to the log groups will wait on writes to the data files. This should be set to a value that supports the service level agreement for your system.

Setting the MTTR target too small means that writes to the log groups wait for writes to the data files (impacting performance). Setting the MTTR target too large means that the instance takes longer to recover after a crash.

Media Failure

Typical Causes	Possible Solutions
Failure of disk drive	1. Restore the affected file from backup. 2. If necessary, inform the database of a new file location. 3. If necessary, recover the file by applying redo information.
Failure of disk controller	
Deletion or corruption of database file	

Media Failure

Oracle defines media failure as any failure which results in the loss or corruption of one or more database files (data, control, or redo log file).

Recovering from media failure requires that you restore and recover the missing files. To ensure your database can be recovered from media failure, follow best practices as outlined in the next few pages. Recovery from media failure will be discussed in more detail in a future lesson.

Configuring for Recoverability

To configure your database for maximum recoverability:

- **Schedule regular backups**
- **Multiplex control files**
- **Multiplex redo log groups**
- **Retain archived copies of redo logs**

Configuring for Recoverability

To provide the best protection for your data you should:

- Schedule regular backups. Most media failures require that you restore the lost or damaged file from backup.
- Multiplex control files. All control files associated with a database are identical. Recovering from the loss of single control file is not difficult. Recovering from the loss of *all* control files is much more challenging. Guard against losing all control files by having multiple copies (at least three).
- Multiplex redo log groups. To recover from instance or media failure, redo log information is used to roll data files forward to the last committed transaction. If your redo log groups rely on a single redo log file, then the loss of that file means data is likely to be lost. Ensure there are at least two copies of each redo log group.
- Retain archived copies of redo logs. If a file is lost and restored from backup, the instance must apply redo information to bring that file up to the latest SCN contained in the control file. The default setting is to overwrite redo information once it has been written to the data files. Your database can be configured to retain redo information in archived copies of the redo logs. This is known as placing the database in `ARCHIVELOG` mode.

Control Files

Protect against database failure by multiplexing control files.

- **At least two copies (Oracle suggests three)**
- **Each copy on a separate disk**
- **At least one copy on a separate disk controller**

Control Files

ORACLE

Control Files

The control file is a small binary file that describes the structure of the database. It must be available for writing by the Oracle server whenever the database is mounted or open. Without this file, the database cannot be mounted and recovery or re-creation of the control file will be required. Your database should have a minimum of two control files (three is preferred) on different disks to minimize the impact of a loss of one control file.

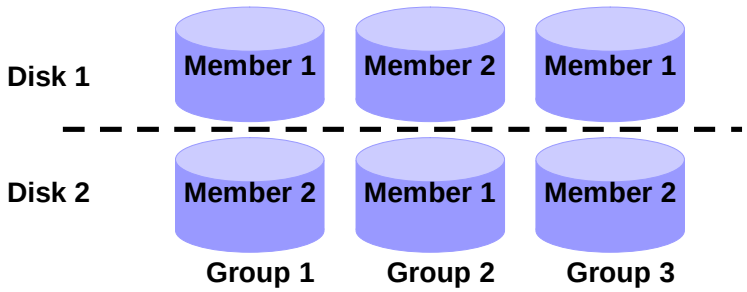
If your database was created with the DBCA, you should already have three control files.

Loss of a single control file will cause the instance to fail because all control files must be available at all times, but recovery is a simple matter of copying one of the other control files. Loss of all control files is slightly more difficult to recover from, but not usually catastrophic. Recovery from loss of control files will be discussed in a later lesson.

Redo Log Files

Multiplexing redo log groups to protect against media failure and loss of data.

- **At least two members (files) per group**
- **Each member on a separate disk drive**
- **Each member on a separate disk controller**
- **Redo logs heavily influence performance**



ORACLE

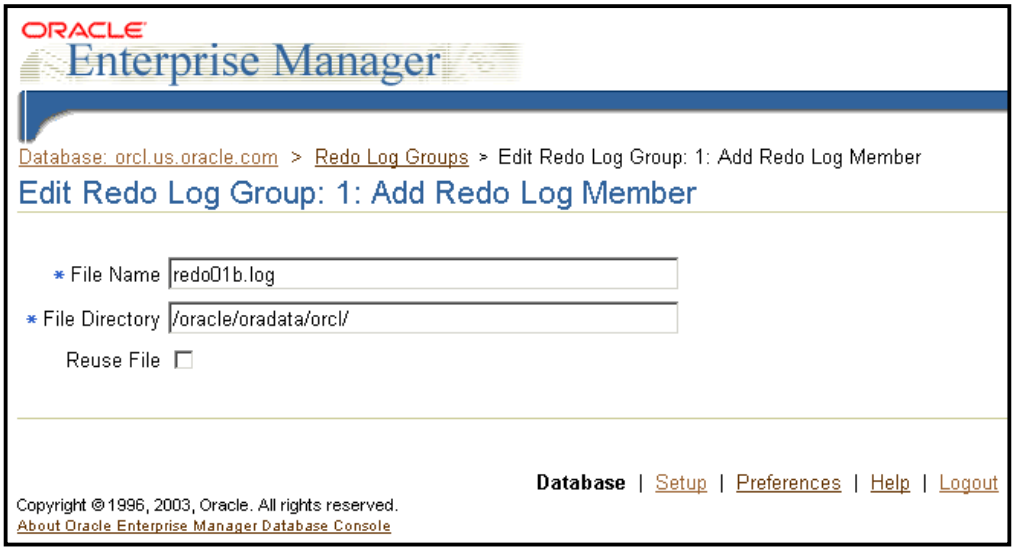
Redo Log Files

Redo log groups are made up one or more redo log files. Each log file within a group is a duplicate of the others. Oracle recommends that redo log groups have at least two files per group, with the files distributed on separate disks/controllers so that no single equipment failure will destroy an entire log group.

Loss of an entire log group is one of the most serious possible media failures because it can result in loss of data. Loss of a single member within a multiple-member log group is trivial, and will not affect database operation other than causing an alert to be published in the alert log. Recovery from loss of a single log file will be discussed in a later lesson. Recovery from loss of an entire log group requires advanced recovery techniques and is discussed in Oracle Database 10*g*: Workshop II.

Remember that redo logs heavily influence database performance because a commit cannot complete until the transaction information has been written to the logs. You should place your redo log files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files. Because only one group is written to at a given time, there is no harm in having members from several groups on the same disk.

Multiplexing the Redo Log



Multiplexing the Redo Log

You can multiplex your redo log by adding a member to an existing log group. Perform the following steps to add a member to a redo log group, this can be done when the database is open with no impact on user performance:

1. Navigate to the Redo Log Groups page.
2. Select a group and click the Edit button, or click the group number link.The Edit Redo Log Group page appears.
3. In the Redo Log Members section, click Add. The Add Redo Log Member page appears.
4. Enter the file name and the file directory. Click OK.
   Note: It is recommended that you store members on separate drives to protect against total loss of the redo log entries in the event of a disk failure.

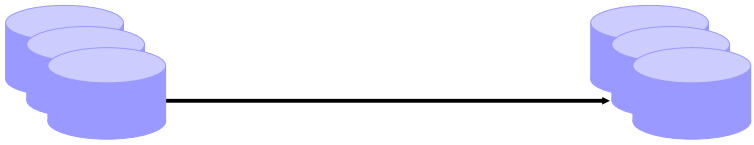
Repeat these steps for every existing group.

When you add the redo log member to a group, the group's status is marked `INVALID`. This is the expected state because a member of the group has not yet been written to. When a log switch occurs and the invalid group becomes the current group, the status changes to `CURRENT`.

Archived Log Files

To preserve redo information, create archived copies of redo log files.

- **Specify archived log file naming convention.**
- **Specify one or more locations to archive logs to.**
- **Switch the database to `ARCHIVELOG` mode.**



Online Redo Log Files **Archived log files**

ORACLE

Archived Log Files

The instance treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next. After all groups have been written to, the instance begins overwriting information in the first log group.

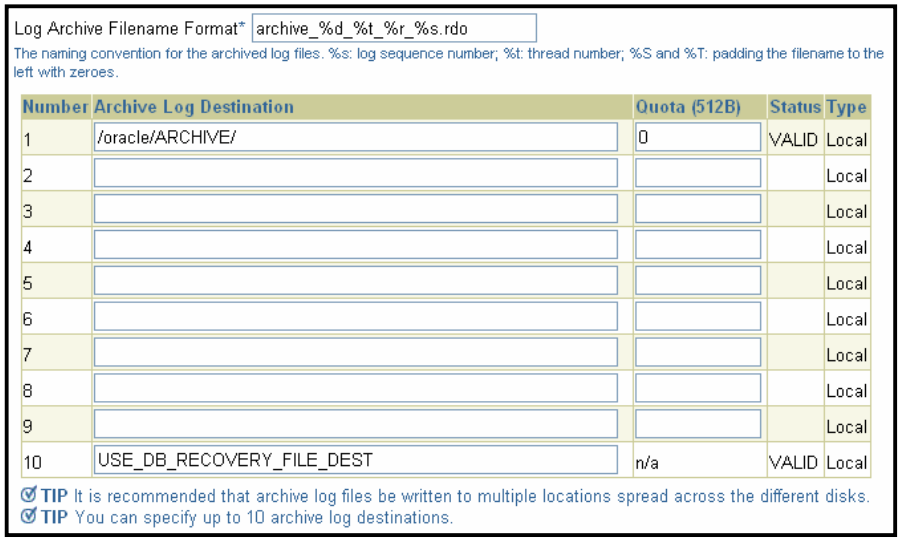
To configure your database for maximum recoverability, you should instruct the database to make a copy of the online redo log group before allowing it to be overwritten. These copies are known as archived logs. To facilitate the creation of archive logs you should:

1. Specify a naming convention for your archived logs.
2. Specify a destination or destinations for storing your archived logs.
3. Place the database in `ARCHIVELOG` mode.

Note: The destination must exist prior to placing the database in `ARCHIVELOG` mode. When a directory is specified as a destination, there should be a trailing slash at the end of the directory name.

Archive Log File Naming and Destinations

Specify archived log file name and destinations.



Archived Log File Naming and Destinations

Configure log file naming an destinations by clicking Configure Recovery Settings from the Maintenance page.

Each archive log file must have a unique name to avoid overwriting older log files. You specify the naming format as show above. To help create unique file names, Oracle Database 10*g* allows several wildcard characters in the name format:

- %s: Includes the log sequence number as part of the file name
- %t: Includes the thread number as part of the file name
- %r: Resetlogs ID. Ensures that the archive log file name remains unique even after certain advanced recovery techniques that reset log sequence numbers
- %d: Includes the database ID as part of the file name

The format *must* include %s, %t, and %r. The use of %d is optional but it should be included if multiple databases share the same archive log destination.

Archived log files can be written to as many as ten different destinations. Destinations may be local (a directory) or remote (an Oracle Net alias for a standby database). Local destinations should end in a slash (/), or a backslash (\) if using Windows.

Archive Log File Naming and Destinations (Continued)

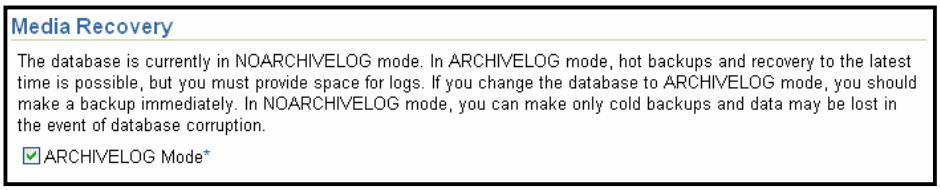
The default destination (number 10) sends archived log files to a location determined by the DB_RECOVERY_FILE_DEST initialization parameter. DB_RECOVERY_FILE_DEST is also known as the flash recovery area. This destination is visible at the bottom of the Configure Recovery Settings properties page as the Flash Recovery Area Location. If you do not want archives sent to this location, simply delete USE_DB_RECOVERY_FILE_DEST.

In order to change recovery settings you must be connected as SYSDBA or SYSOPER.

ARCHIVELOG Mode

Place the database in ARCHIVELOG mode.

- **Click the ARCHIVELOG Mode checkbox**
- **Click Apply. The database can only be set to ARCHIVELOG mode from the MOUNT state. Click Yes when asked if you want to restart the database.**



ORACLE

ARCHIVELOG Mode

Placing the database in ARCHIVELOG mode prevents redo logs from being overwritten until they have been archived, and is the last step in configuring the database for archiving redo information.

The SQL command to place the database in ARCHIVELOG mode is:

```
SQL> ALTER DATABASE ARCHIVELOG;
```

This command can be issued only while the database is in the MOUNT state, so the instance must be restarted to complete this last step. You will be asked for operating system and database credentials during the restart of the database. The database credentials *must* be for a user with SYSDBA privileges.

After the instance is restarted, the changes you made to the archive processes, log format, and log destinations will take effect.

With the database in NOARCHIVELOG mode (the default), recovery is possible only up until the time of the last backup. All transactions made after that backup are lost.

In ARCHIVELOG mode, recovery is possible up until the time of the last commit. Most production databases are run in ARCHIVELOG mode.

Summary

In this lesson you should have learned how to:

- Describe the basics of database backup, restore and recovery
- List the types of failure that may occur in an Oracle Database
- Identify the importance of checkpoints, redo log files, and archived log files
- Configure `ARCHIVELOG` mode
- Describe ways to tune instance recovery

ORACLE

Practice 18:
Backup and Recovery Concepts

This practice covers the following:

- **Multiplexing control files**
- **Multiplexing redo log groups**
- **Placing your database in `ARCHIVELOG` mode**
- **Ensuring that redundant archive logs are created**

ORACLE

Practice 18: Backup and Recovery Concepts

Background: Your database is ready to move from test and development into production. Configure your database to reduce the chances of failure or data loss.

Tasks:

- Verify that you have three control files to ensure dual redundancy.
- Verify that all redo log groups have two members to ensure redundancy.
- Place your database in `ARCHIVELOG` mode.
- Ensure that redundant archive log files are created.

1. Verify that you have three control files to ensure dual redundancy.
2. Add additional members to each log group to ensure redundancy.
3. Place your database in `ARCHIVELOG` mode. Configure redundant archive log destinations, one to the flash recovery area, the other to `/u01/app/oracle/archive`.

19

Database Backups

Objectives

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

- **Create consistent database backups**
- **Back up your database without shutting it down**
- **Create incremental backups**
- **Automate database backups**
- **Monitor the flash recovery area**

ORACLE

Terminology

- **Backup strategy may include:**
  - **The entire database (whole)**
  - **A portion of the database (partial)**
- **Backup type may be:**
  - **All information from all data files (full)**
  - **Only information that has changed since some previous backup (incremental)**
- **Backups mode may be:**
  - **Offline (consistent, cold)**
  - **Online (inconsistent, hot)**

Terminology

A whole database backup includes all data files and at least one control file (remember that all control files within a database are identical).

Partial database backups may include zero or more tablespaces, zero or more data files, and may or may not include a control file.

Full backups make a copy of every data block within the files being backed up that contains data.

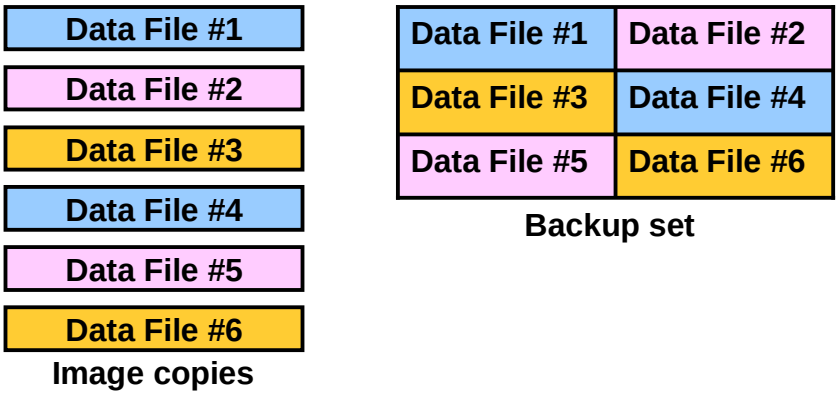
Incremental backups make a copy of all data blocks that have changed since some previous backup. Oracle Database 10*g* supports two levels of incremental backup (0-1). A level 0 or baseline backup is equivalent to a full backup and contains all data blocks. A level 1 incremental backs up all database blocks changed since the level 0 backup. To restore using incremental backups, the baseline backup must first be restored, and then the incremental.

Offline backups (also known as consistent backups) are taken while the database is not open. They are consistent because at the time of the backup, the SCN data file headers matches the SCN in the control files.

Online backups (also known as hot or inconsistent backups) are taken while the database is open. The backups are inconsistent because with the database open there is no guarantee that the data files are synchronized with the control files. Inconsistent backups require recovery in order to be used.

Terminology

- **Backups may be stored as:**
  - **Image copies**
  - **Backup sets**



Terminology (continued)

Image copies are duplicates of data or archived log files (similar to simply copying the files using operating system commands).

Backup sets are copies of one or more data or archived log files. With backup sets, empty data blocks are not stored, thereby causing backup sets to use less space on disk or tape. Backup sets can be compressed to further reduce the space requirements of the backup.

Image copies must be backed up to disk. Backup sets can be sent to disk or directly to tape.

The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy only the file or files need to be retrieved from tape. With backup sets the entire backup set must be retrieved from tape before you extract the file or files that are needed.

The advantage of creating backups as backup sets is better space usage. Most databases contain 20% or more empty blocks. Image copies back every single data block up, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems the advantages of backup sets outweigh the advantages of image copies.

Databases in `NOARCHIVELOG` mode must perform offline, full, whole database backups. Databases in `ARCHIVELOG` mode have access to the full range of backup options.

Recovery Manager (RMAN)

Enterprise Manager uses Recovery Manager (RMAN) to perform backup and recovery operations.

- **Command-line client for advanced functions**
- **Powerful control and scripting language**
- **Published API that allows interface with most popular backup software**
- **Backs up data, control, archived log, and server parameter files**
- **Backs up files to disk or tape**

ORACLE

Recovery Manager (RMAN)

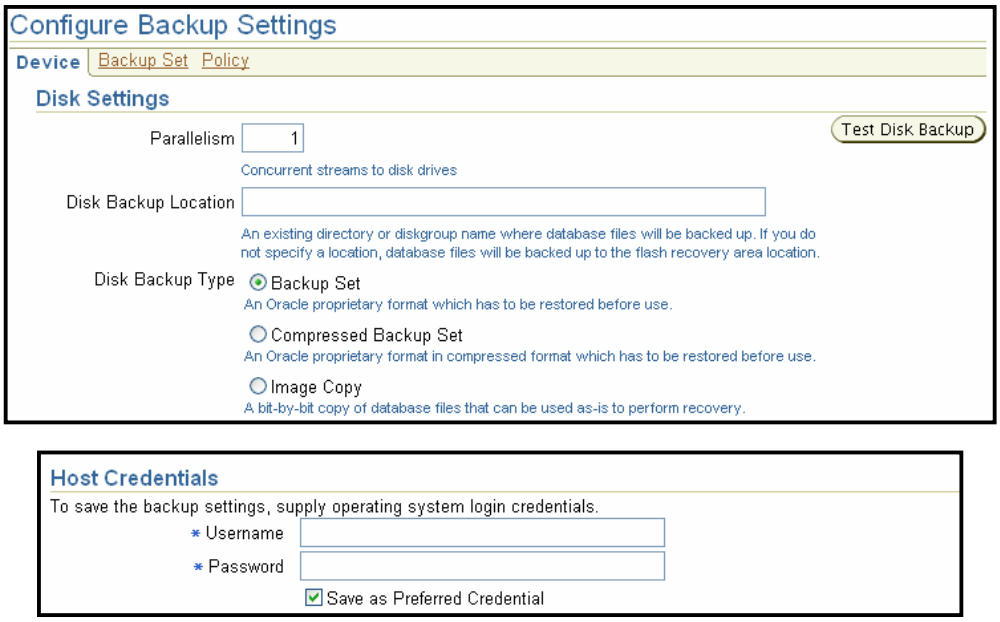
Recovery Manager is the component of Oracle Database 10*g* that is used to perform backup and recovery operations. RMAN can take consistent and inconsistent backups, perform incremental or full backups, and back up either the whole database or an portion of it.

RMAN uses its own powerful job control and scripting language, as well as a published API that interfaces RMAN with many popular backup software solutions.

RMAN can store backups on disk for quick recovery, or place them on tape for long-term storage. In order for RMAN to store backups on tape, an interface to the tape device known as a media management layer (MML) must be configured.

Enterprise Manager supplies a graphical interface to the most commonly used RMAN functionality. Advanced backup and recovery operations are accessible through RMAN's command-line client. For more information on advanced RMAN capabilities, refer to the *Oracle Database 10g: Administration Workshop II* or *Oracle Backup and Recovery Advanced User's Guide*.

Configuring Backup Settings



ORACLE

Configuring Backup Settings

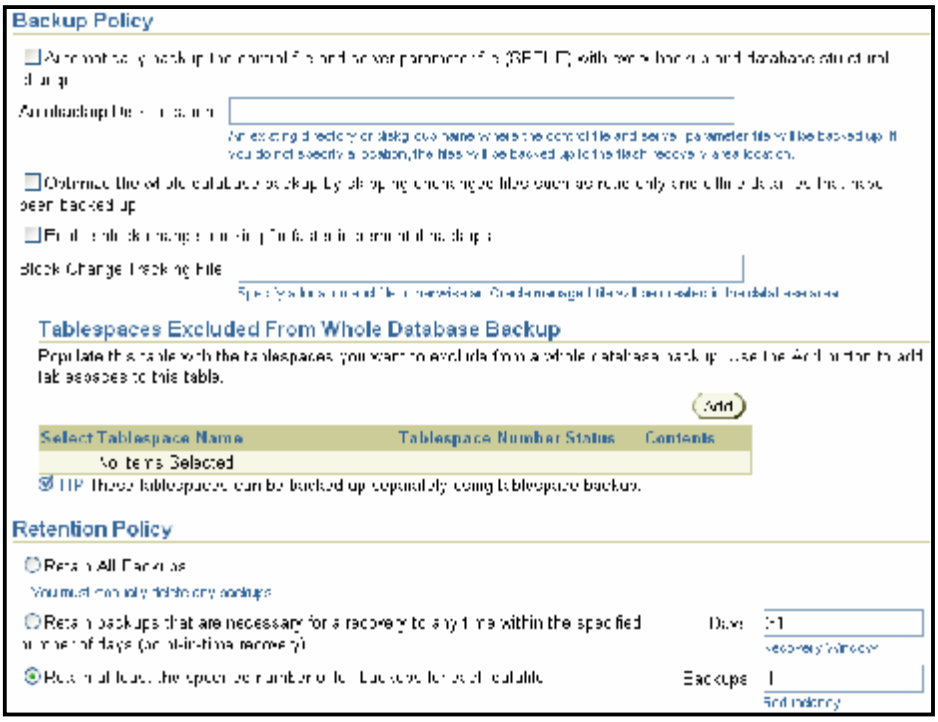
Navigate to the Maintenance page and click Configure Backup Settings. From this property page you manage the persistent backup settings that will be used for creating backups. There are separate settings for disk and tape. Tape settings depend on the media management layer capabilities. Disk settings include:

- Parallelism: How many separate streams of backup information do you want to create. The best setting for parallelism depends on your hardware. A single CPU, single disk controller, or single disk server would not benefit from conducting parallel backups. As hardware resources increase, the appropriate degree of parallelism also increases.
- Disk backup location: Where should backups be stored? The default is the Flash Recovery Area. If you change this, click “Test Disk Backup” to verify that RMAN can write to the new location.
- Disk backup type: Select image copy, backup set, or compressed backup set.

Click the Backup Set tab to set the maximum size of backup set files (backup sets can be divided further if needed for easier archiving).

Host credentials are required for Enterprise Manager to save any changes to the backup settings.

Configuring Backup Settings



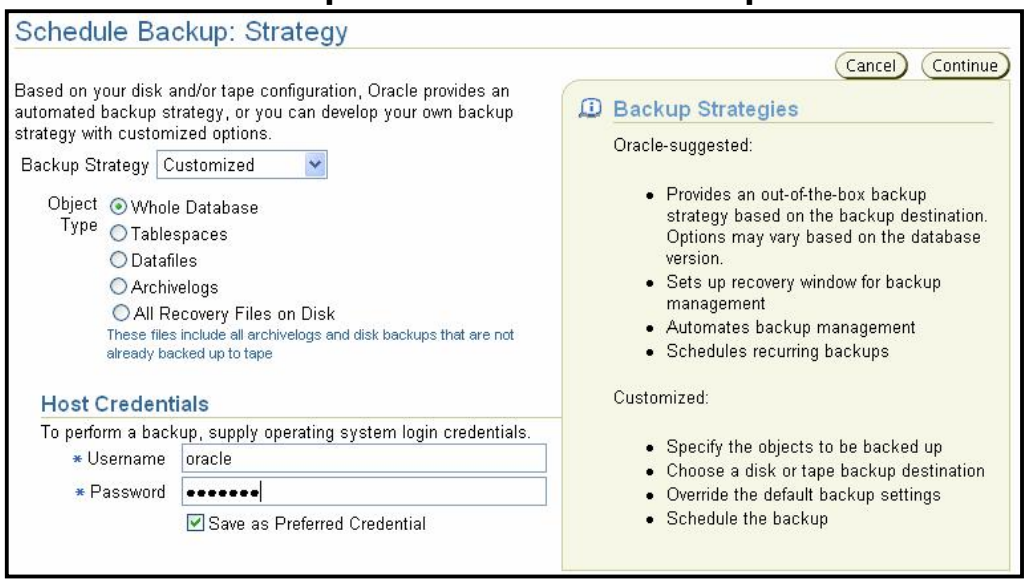
Configuring Backup Settings (continued)

Click the Policy tab to:

- Automatically back up the control and SPFILE with each backup. You are also given the opportunity to specify a location for these backups if you don't want them to go to the flash recovery area.
- Optimize backups by not backing up files that exactly match a file already part of the retained backups. This setting allows you to skip read-only and offline data files.
- Enable block change tracking and specify a location for the tracking file. If you intend to create incremental backups this setting can decrease the time required to choose which blocks to include in the incremental backup.
- Exclude tablespace from a whole database backup. Some administrators choose not to back up tablespaces containing data or objects that can easily be re-created (such as indexes or data that is batch-loaded frequently).
- Retention policy: How long should RMAN keep your backups? If you are using the flash recovery area to store backups, RMAN automatically deletes old backups to make room for new ones (if the retention policy allows it). By default only the last backup is retained. The retention policy can be specified as a number of backups or a number of days.

Scheduling Backups: Strategy

Choose whole or partial database backup.

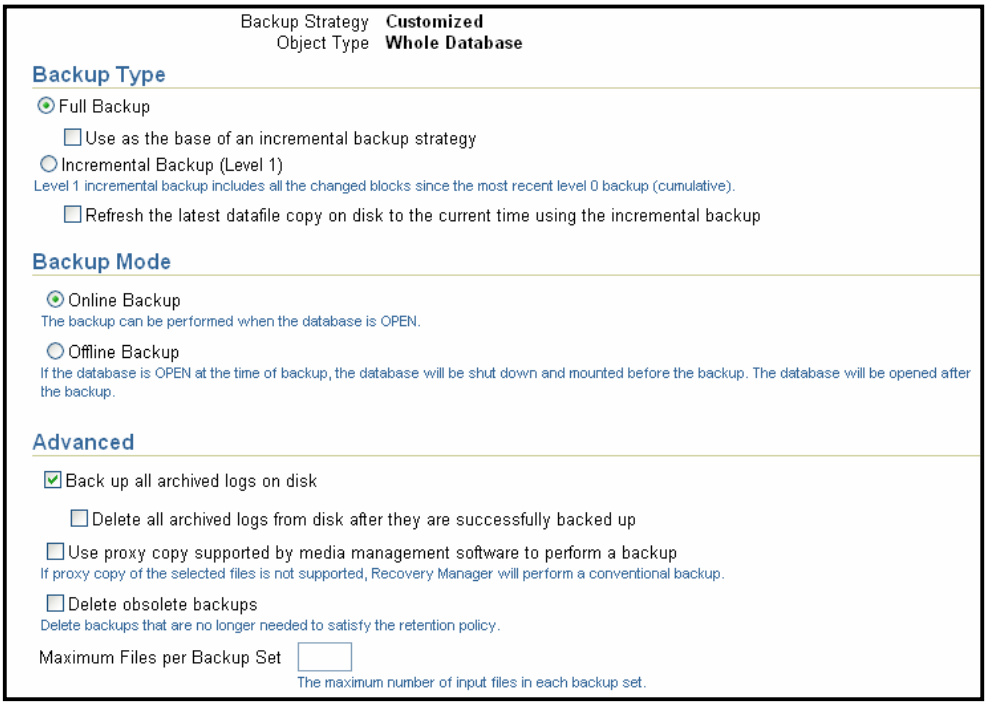


Scheduling Backups: Strategy

Click Schedule Backup from the Backup/Recovery section of the Maintenance properties page. Select either the Oracle-suggested backup strategy or your own customized strategy. The Oracle-suggested backup strategy makes a one-time whole-database, baseline incremental level 0, online backup, and then schedules incremental level 1 backups for each following day.

By selecting Customized, you gain access to a wider range of configuration options. Select which objects you want to back up—the whole database (the default) or individual tablespaces, data files, archivelogs, or any Oracle backups currently residing on disk (to move them to tape).

Scheduling Backups: Options



Scheduling Backups: Options

Choose full or incremental level backups. If performing a full database backup, you can select "Use as the base of an incremental backup strategy" to make the full database backup an incremental level 0. If using image copies, selecting the "Refresh the latest datafile copy on disk to the current time using the incremental backup" check box updates the existing backup rather than creating a new image copy.

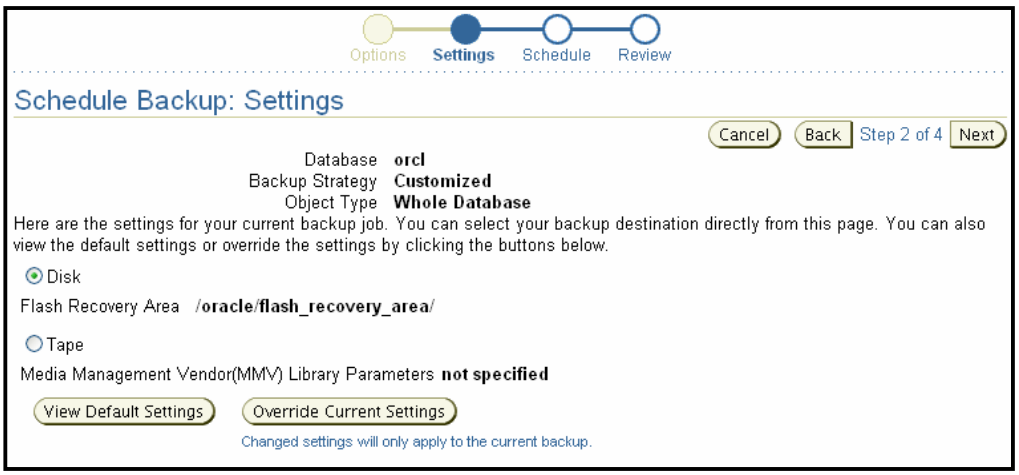
Select online or offline backup modes. Remember that `NOARCHIVELOG` mode databases must perform offline backups.

You may also choose to back up archived log files and delete them from disk once they are successfully backed up.

Click the delete obsolete backups to remove any backups that fall outside the retention policy you configured earlier. RMAN will automatically remove obsolete backups if you are backing up to the flash recovery area.

Select the maximum number of files you want to place in each backup set. Limiting the number of files per set may speed the restore process because the size of the backup set that must be returned from tape will be smaller than if a single backup set containing all the data files for the database were created.

Scheduling Backups: Settings

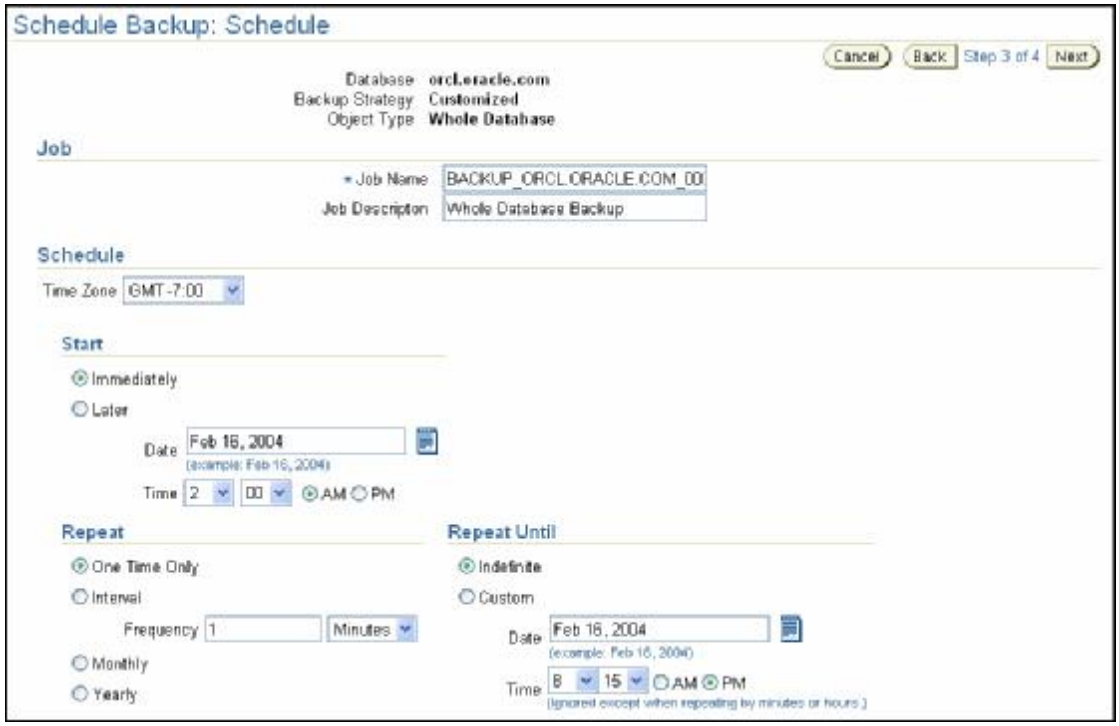


Persistent backup configuration settings can be overridden for this backup by clicking Override Current Settings.

Scheduling Backups: Settings

From the Settings page you can override many of the persistent configuration settings you created earlier.

Scheduling Backups: Schedule



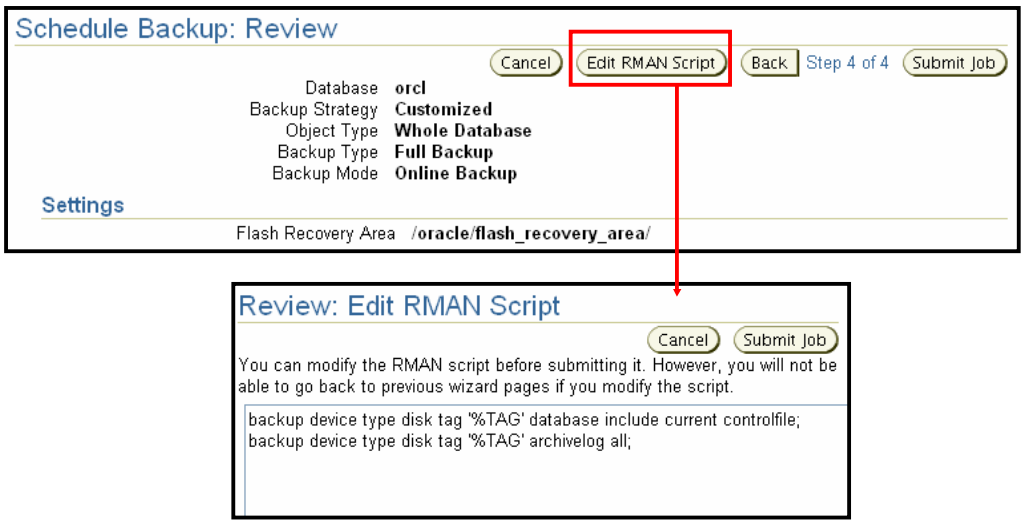
ORACLE

Scheduling Backups: Schedule

Choose how you want the backup to be scheduled—either as a one time job or as an automated, reoccurring process.

To configure a database for maximum recoverability, Oracle suggests regularly scheduled backups. Automating backups can simplify the administrator's workload.

Scheduling Backups: Review



Click Edit RMAN Script to review RMAN commands.

ORACLE

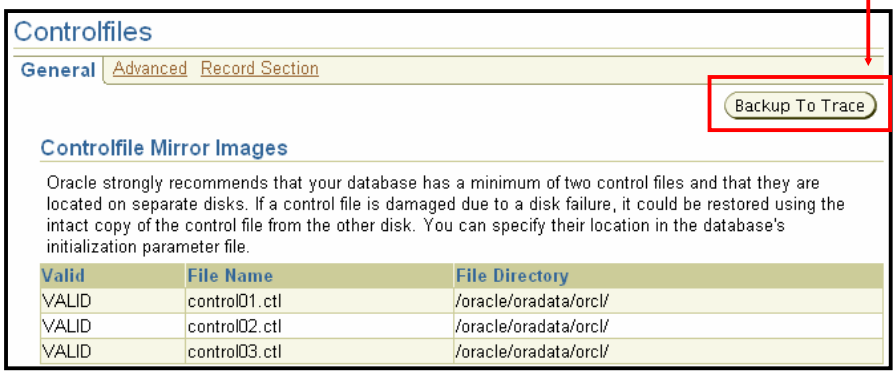
Scheduling Backups: Review

RMAN uses its own command syntax and scripting language. The Edit RMAN Script button gives you a look at the commands the backup scheduler has generated based on your inputs.

From this page you can customize the RMAN scripts if needed, or copy them for recording purposes.

Backup Control File to Trace

Control files have an additional backup option.



Control file trace backups may be used to recover from loss of all control files.

ORACLE

Backup Control File to Trace

Click Controlfiles in the Storage section of the Administration properties page to manage your database's control files. Control files have an additional backup option; they may be backed up to trace. A control file trace backup contains the SQL statement required to re-create the control files in the event that all control files are lost.

Although it is very unlikely that a properly configured database, with multiple copies of the control file placed on separate disks and separate controllers, would lose all control files at the same time, it is possible and therefore the administrator should back the control file up to trace after each change to the physical structure of the database (adding tablespaces or data files, adding additional redo log groups).

Trace copies of the control file can be created using Enterprise Manager (as show above) by clicking Controlfiles from the Administration properties page, or with the SQL command:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

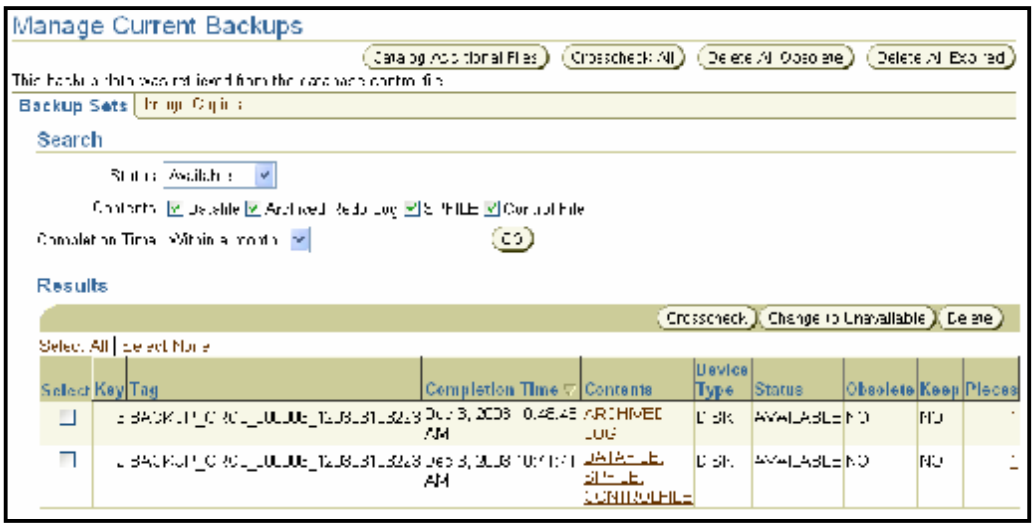
The trace backup is created in the location specified by the `USER_DUMP_DEST` initialization parameter with a file name such as *`sid_ora_pid.trc`*.

Backup Control File to Trace (continued)

The trace file contains information about archive log destinations followed by commands that create replacement control files and then recover the database:

```
CREATE CONTROLFILE REUSE DATABASE ORCL NORESETLOGS ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
    MAXINSTANCES 8
    MAXLOGHISTORY 226
LOGFILE
  GROUP 1 '/oracle/oradata/orcl/redo01.log'  SIZE 10M,
  GROUP 2 '/oracle/oradata/orcl/redo02.log'  SIZE 10M,
  GROUP 3 '/oracle/oradata/orcl/redo03.log'  SIZE 10M
DATAFILE
  '/oracle/oradata/orcl/system01.dbf',
  '/oracle/oradata/orcl/undotbs01.dbf',
  '/oracle/oradata/orcl/sysaux01.dbf',
  '/oracle/oradata/orcl/users01.dbf',
  '/oracle/oradata/orcl/example01.dbf'
CHARACTER SET WE8ISO8859P1;
-- Commands to re-create incarnation table
-- Below log names MUST be changed to existing filenames on
-- disk. Any one log file from each branch can be used to
-- re-create incarnation records.
-- ALTER DATABASE REGISTER LOGFILE
  '/oracle/flash_recovery_area/ORCL/archivelog/2003_12_05/o1_mf_1_
  1_%u_.arc';
-- ALTER DATABASE REGISTER LOGFILE
  '/oracle/flash_recovery_area/ORCL/archivelog/2003_12_05/o1_mf_1_
  1_%u_.arc';
-- Recovery is required if any of the datafiles are restored
  backups,
-- or if the last shutdown was not normal or immediate.
RECOVER DATABASE
-- All logs need archiving and a log switch is needed.
ALTER SYSTEM ARCHIVE LOG ALL;
-- Database can now be opened normally.
ALTER DATABASE OPEN;
-- Commands to add tempfiles to temporary tablespaces.
-- Online tempfiles have complete space information.
-- Other tempfiles may require adjustment.
ALTER TABLESPACE TEMP ADD TEMPFILE
  '/oracle/oradata/orcl/temp01.dbf'
     SIZE 20971520  REUSE AUTOEXTEND ON NEXT 655360  MAXSIZE
  32767M;
```

Manage Backups



Click Edit RMAN Script to review RMAN commands.

ORACLE

Manage Backups

Click Manage Current Backups from the Maintenance properties page to manage your existing backups. From this page you can see when a backup was completed, where it was created (disk or tape), and if it is still available.

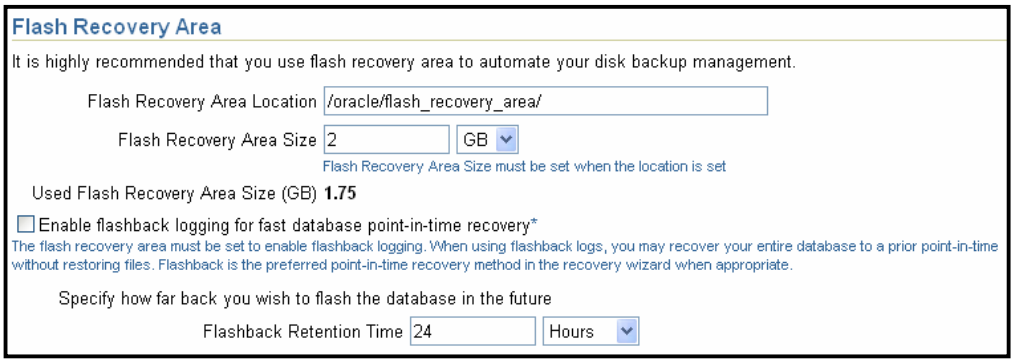
At the top of the Manage Current Backups page you can see four buttons that allow you to work with existing backups.

- Catalog Additional Files: Although RMAN (working through Enterprise Manager) is the recommended way to create backups, it is possible to create image copies in other ways (for example, by shutting down the database and copying the files). RMAN and Enterprise Manager will not know about those backups unless you add them to the catalog.
- Crosscheck All: RMAN can automatically delete obsolete backups, but you can also delete them by using operating system commands. If you delete a backup outside of RMAN, the catalog will not know it is missing until you perform a crosscheck between the catalog and what is really there.
- Delete All Obsolete: Deletes backups older than the retention policy
- Delete All Expired: Deletes the catalog listing for any backups that were not found when the crosscheck was performed

Flash Recovery Area

Monitor the Flash Recovery Area

- Configure flashback logging
- Size the recovery area
- Monitor current space consumption



Flash Recovery Area

The flash recovery area is a space set aside on disk to contain archived logs, backups, and flashback logs.

If you have configured your archived logs to be written to this location (with the USE_DB_RECOVERY_AREA flag in one of the locations), it is important to monitor this space to ensure it doesn't reach capacity. If the instance is unable to create an archived log due to lack of space, it will pause until the administrator corrects the situation.

Clicking Configure Recovery Settings from the Maintenance property page takes you to the Flash Recovery Area settings. From this page you can:

- Specify the location of the flash recovery area
- Specify the size of the flash recovery area (Oracle suggests this be at least twice the size of the database so it can hold a backup and several archived logs)
- Verify how much of the flash recovery area has been consumed
- Configure flashback database (flashback database is discussed in *Oracle Database 10g: Administration Workshop II* and in the *Oracle Database Administration* guide.

Summary

In this lesson you should have learned how to:

- Create consistent database backups
- Back up your database without shutting it down
- Create incremental backups
- Automate database backups
- Monitor the flash recovery area

Practice 19:
Database Backups

This practice covers the following:

- **Configuring your database for backups**
- **Backing up your database while the database is open for user activity**
- **Scheduling automatic nightly incremental backups for your database**

ORACLE

Practice 19: Database Backups

Background: Your database is ready to move from test and development into production. Ensure your database is configured so that recovery is possible without loss of data.

Tasks:

- Configure your database for backups.
- Back your database up while the database is open for user activity.
- Schedule nightly incremental backups for your database.

1. Configure your database for backups.
   - Set Parallelism to 1.
   - Set Disk Backup Location to /u01/app/oracle/backup/.
     Note: The directory must exist before a backup operation can be performed.
   - Choose Backup Set as your Disk Backup Type.
2. Back up your database while the database is open for user activity.
3. Schedule nightly incremental backups for your database.
   - Set the Job Name to Nightly_Backup and accept the default Job Description
   - Accept today's date and use the drop-down lists and option buttons to select 11:00 PM for the Time.
   - Select Interval from the Repeat region, and set the Frequency to 1 day.
   - Select Indefinite in the Repeat Until region.

20

Database Recovery

Objectives

After completing this lesson you should be able to recover from loss of a:

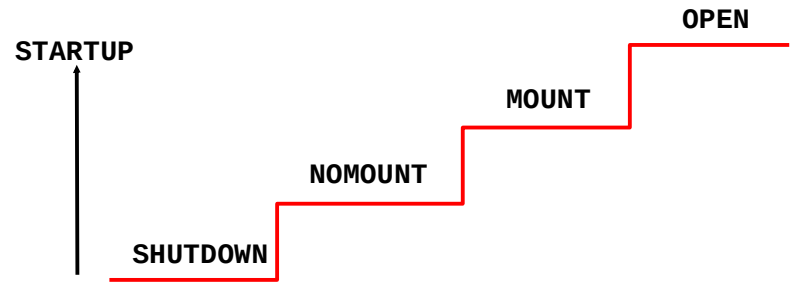
- **Control file**
- **Redo log file**
- **Data file**

ORACLE

Opening a Database

To open a database:

- **All control files must be present and synchronized**
- **All online data files must be present and synchronized**
- **At least one member of each redo log group must be present**



ORACLE

Opening a Database

As a database moves from the shutdown stage to being fully open, it performs internal consistency checks at each stage.

- NOMOUNT: In order for an instance to reach NOMOUNT (also known as STARTED) status, the instance must read the initialization parameter file. No database files are checked in reaching NOMOUNT.
- MOUNT: As the instance moves to MOUNT status, it checks that all control files listed in the initialization parameter file are present and synchronized. If even one control file is missing or corrupt, the instance will return an error to the administrator noting the missing control file and will remain at the NOMOUNT state.
- OPEN: When the instance moves from the MOUNT to the OPEN state it:
  - Checks that all redo log groups known to the control file have at least one member present. Any missing members are noted in the alert log.

Opening a Database (continued)

- Verifies that all data files known to the control file are present unless they have been taken offline. Offline files are not checked until the administrator tries to bring them online. The administrator may take a data file offline and open the instance as long as the data file does not belong to the SYSTEM or UNDO tablespaces. If any files are missing an error noting the first missing file is returned to the administrator and the instance remains at the MOUNT state. When the instance finds files that are missing, only the first file causing a problem appears in the error message. To find all files needing recovery, the administrator can check the v$recover_file dynamic performance view to get a complete list of files needing attention:

```
SQL> startup
ORACLE instance started.
Total System Global Area  171966464 bytes
Fixed Size                   775608 bytes
Variable Size             145762888 bytes
Database Buffers           25165824 bytes
Redo Buffers                 262144 bytes
Database mounted.
ORA-01157: cannot identify/lock data file 4 - see DBWR trace
file
ORA-01110: data file 4: '/oracle/oradata/orcl/users01.dbf'
SQL> SELECT name, error
  2  FROM v$datafile
  3  JOIN v$recover_file
  4  USING (file#);
NAME                                 ERROR
------------------------------------ ------------------
/oracle/oradata/orcl/users01.dbf     FILE NOT FOUND
/oracle/oradata/orcl/example01.dbf   FILE NOT FOUND
```

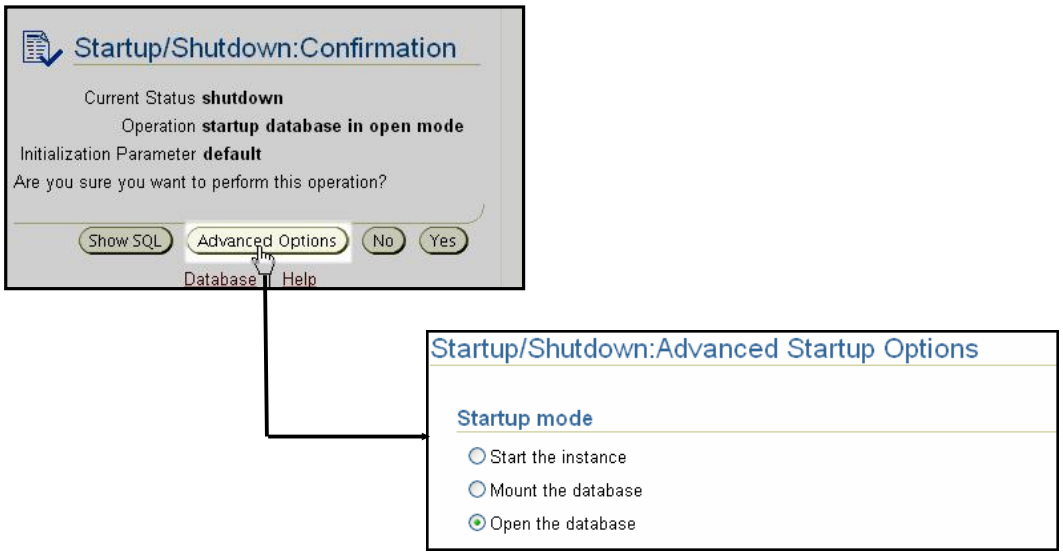
- Verifies that all data files that are not offline or read-only are synchronized with the control file. If necessary, instance recovery is automatically performed. However if a file is out of synchronization more than can be recovered using the online redo log groups, then the administrator must perform media recovery. If any files require media recovery, an error message noting the first file requiring recovery is returned to the administrator and the instance remains at the MOUNT state.

```
ORA-01113: file 4 needs media recovery
ORA-01110: data file 4: '/oracle/oradata/orcl/users01.dbf'
```

Again, v$recover_file gives a complete list of files needing attention. Files that are present but need media recovery will be listed, but will not have an error message.

Changing Instance Status

Use Database Control to alter the instance's status:



ORACLE

Changing Instance Status

When starting the instance, the default start mode is OPEN. You may choose to start the instance in some other mode, or problems with the database may force you to start in another mode. The Advanced Startup Options properties screen allows you to select a state other than OPEN when starting the instance and to alter the state if the instance has already started in another mode. You may also use SQL commands to modify an instance's status:

```
SQL> STARTUP NOMOUNT
ORACLE instance started.

Total System Global Area  188743680 bytes
Fixed Size                   778036 bytes
Variable Size             162537676 bytes
Database Buffers           25165824 bytes
Redo Buffers                 262144 bytes

SQL> ALTER DATABASE MOUNT
Database altered.

SQL> ALTER DATABASE OPEN
```

Keeping a Database Open

Once open, the instance fails with:

- **Loss of any control file**
- **Loss of a data file belonging to the system or undo tablespaces**
- **Loss of an entire redo log group. As long as at least one member of the group is available, the instance remains open.**

ORACLE

Keeping a Database Open

After an instance is open, media failure causing loss of a control file, loss of a member of a redo log group, or loss of a data file belonging to the `SYSTEM` or `UNDO` tablespaces will cause the instance to fail.

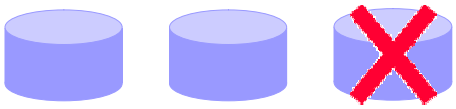
In many cases the failed instance will not completely shut down, but will be unable to continue to perform work. Recovering from these types of media failures must be done with the database down, so the administrator should use the `SHUTDOWN ABORT` command before beginning recovery efforts.

Loss of data files belonging to other tablespaces does not cause instance failure, and the database can be recovered while open with work continuing in other tablespaces.

Loss of a Control File

If a control file is lost or corrupted:

1. **The instance will normally abort. If it is still open, shut it down.**
2. **Restore the missing control file by copying an existing control file.**
3. **Start the instance.**



Control files

ORACLE

Loss of a Control File

Recovering from loss of a control file (as long as at least one control file remains) can be accomplished as follows:

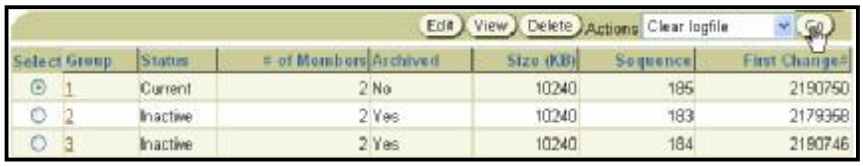
1. If the instance did not already fail, shut it down using `SHUTDOWN ABORT`.
2. Copy one of the remaining control files to the missing file's location. If the media failure was due to loss of a disk drive or controller, copy one of the remaining control files to some other location and update the instance's parameter file to point to the new location. Alternatively, you can delete the reference to the missing control file from the initialization parameter file. Remember that Oracle suggests having *at least* two control files at all times.
3. Start the instance.

Recovering from loss of all control files is covered in *Oracle Database 10g: Administration Workshop II*.

Loss of a Redo Log File

If a member of a log file group is lost, as long as the group still has at least one member:

1. **Normal operation of the instance will not be affected.**
2. **You will receive a message in the alert log notifying you that a member can not be found.**
3. **Restore the missing log file by copying one of the remaining files from the same group.**



ORACLE

Loss of a Redo Log File

Recovering from the loss of a single log group member should not affect the running instance.

1. Determine there is a missing log file by examining the alert log.
2. Restore the missing file by copying one of the remaining files from the same group.
3. If the media failure was due to loss of a disk drive or controller, rename the missing file.
4. If the group has already been archived, or if you are in `noarchivelog` mode, you may choose to solve the problem by clearing the log group to re-create the missing file or files. Select the appropriate group and select the Clear Logfile action. You can also clear the affected group manually with the command:

```
SQL> ALTER DATABASE CLEAR LOGFILE GROUP #;
```

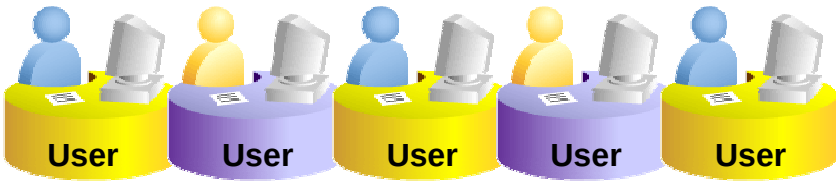
Note: Database Control will not allow you to clear a log group that has not been archived. Doing so breaks the chain of redo information. If you must clear an unarchived log group, you should *immediately* take a full backup of the whole database. Failure to do so may result in a loss of data if another failure occurs. To clear an unarchived log group use the command:

```
SQL> ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP #;
```

Loss of a Data File in NOARCHIVELOG Mode

If the database is in NOARCHIVELOG mode, and any data file is lost:

1. **Shut the instance down if it is not already down.**
2. **Restore the entire database, including all data and control files, from backup.**
3. **Open the database.**
4. **Have users reenter all changes made since the last backup.**



ORACLE

Loss of a Data File in NOARCHIVELOG Mode

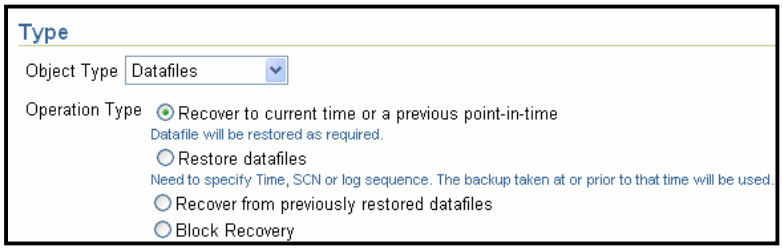
The loss of *any* data file from a database in NOARCHIVELOG mode requires complete restoration of the database including control files and all data files.

With the database in NOARCHIVELOG mode, recovery is possible only up until the time of the last backup, so users must reenter all changes made since that backup.

1. Shut down the instance if it is not already down.
2. Click Perform Recovery from the Maintenance properties page.
3. Select “Whole Database” as the type of recovery.

Loss of a Noncritical Data File in ARCHIVELOG Mode

If a data file is lost or corrupted, and that file does not belong to the SYSTEM or UNDO tablespace, then restore and recover the missing data file.



Users

Loss of a Noncritical Data File in ARCHIVELOG Mode

With the database in ARCHIVELOG mode, the loss of any data file not belonging to the SYSTEM or UNDO tablespaces only affects objects that are in the missing file. The rest of the database remains available for users to continue work. For the missing data file,

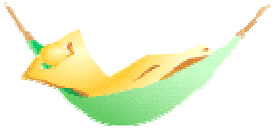
1. Click Perform Recovery from the Maintenance properties page.
2. Select "Datafiles" as the recovery type and select "Restore to current time."
3. Add all data files needing recovery.
4. Determine whether you want to restore the files to the default location or (if missing a disk or controller) to a new location.
5. Submit the RMAN job to restore and recover the missing files..

Because the database is in ARCHIVELOG mode, recovery up until the time of the last commit is possible and users are not required to reenter any data.

Loss of a System-Critical Data File in ARCHIVELOG Mode

If a data file is lost or corrupted, and that file belongs to the SYSTEM or UNDO tablespace :

1. **The instance may or may not shut down automatically. If it does not, use SHUTDOWN ABORT to bring the instance down.**
2. **Mount the database.**
3. **Restore and recover the missing data file.**
4. **Open the database.**



Users

ORACLE

Loss of a System-Critical Data File

Data files belonging to the SYSTEM tablespace or containing UNDO data are considered system-critical. A loss of one of these files requires the database to be restored from the MOUNT state (unlike other data files that may be restored with the database open).

1. If the instance is not already shut down, shut it down.
2. Mount the database.
3. Click Perform Recovery from the Maintenance properties page.
4. Select “Datafiles” as the recovery type and select “Restore to current time.”
5. Add all data files needing recovery.
6. Determine whether you want to restore the files to the default location or (if missing a disk or controller) to a new location.
7. Submit the RMAN job to restore and recover the missing files.
8. Open the database. Users are not required to reenter data because the recovery was to the time of the last commit.

Summary

In this lesson you should have learned how to recover from loss of a:

- **Control file**
- **Redo log file**
- **Data file**

ORACLE

Practice Overview: Database Recovery

This practice covers recovering from loss of a:

- **Control file**
- **Redo log file**
- **Noncritical data file**
- **System-critical data file**

ORACLE

Practice 20: Database Recovery

Background: Many failures of the Oracle database can be traced to some sort of media failure such as disk or controller failure. Recover your database from a variety of simulated media failures.

Tasks:

- Recover from loss of a control file.
- Recover from loss of a redo log member.
- Recover from loss of a non-system critical data file.
- Recover from loss of a system-critical data file.

1. Recover from loss of a control file..
   a) As user SYSTEM run the SQL script $HOME/labs/lab20_01_a.sql.
      ```
      SQL> @$HOME/labs/lab20_01_a.sql
      ```
   b) Now run the SQL script $HOME/labs/lab20_01_b.sql. This script deletes one of your control files.
   c) The Help desk begins receiving calls saying that the database appears to be down. Troubleshoot and recover as necessary.
   d) Why did you have to use two commands to move the instance state from NOMOUNT to OPEN?
   e) Why did you use operating system commands to restore the control file instead of using Oracle Recovery Manager?
   f) What is another way you could have solved this problem?
2. Recover from loss of a redo log member.
   a) Run the SQL script $HOME/labs/lab20_02.sql. This script deletes one of your redo log files.
      ```
      SQL> @$HOME/labs/lab20_02.sql
      ```
   b) During a routine check of the alert log for ORA- errors, you notice the following message:
      ```
      ORA-00313: open failed for members of log group 1 of thread 1
      ORA-00312: online log 1 thread 1:
      '/oracle/oradata/orcl/redo01.log' ORA-27037: unable to obtain
      file status Linux Error: 2: No such file or directory
      ```
   c) Troubleshoot and recover as necessary.
   d) When clearing a log file, what determines whether a complete backup is required immediately following the clear command?
   e) Why doesn't Enterprise Manager produce a critical alert for a missing log file?
3. Recover from loss of a nonsystem or undo data file.
   a) Run the SQL script $HOME/labs/lab20_03.sql. This script deletes one of your nonsystem or undo data files.
      ```
      SQL> @$HOME/labs/lab20_03.sql
      ```

b) The Help desk has received a call from a user complaining that they are unable to access the `countries` table in the `HR` application schema. Check the table to see if there is a problem.
c) Troubleshoot and recover as necessary.

4. Recover from loss of a system/undo data file.
   a) Why is recovery from the loss of a system data file or a data file belonging to an undo tablespace different from recovering a nonsystem or undo data file?
   b) Run the SQL script `$HOME/labs/lab20_04.sql`. This script deletes one of your system or undo data files.
      ```
      SQL> @$HOME/labs/lab20_04.sql
      ```
   c) The Help desk begins receiving calls saying that the database appears to be down. Troubleshoot and recover as necessary.