Oracle Database 10*g*: Administration Workshop I

Volume I • Student Guide

D17090GC31

Edition 3.1

December 2008

D57298

Authors

Tom Best

Maria Billings

Technical Contributors and Reviewers

Celia Antonio

Larry Baumann

Tammy Bednar

Howard Bradley

M.J. Bryksa

Sandra Cheevers

Steve Friedberg

Joel Goodman

John Hibbard

Magnus Isaksson

Sushma Jagannath

Christine Jeal

Steven Karam

Donna Keesling

Stella Kister

Pierre Labrousse

Stefan Lindblad

Dee Matishak

Paul Needham

Raza Siddiqui

James Spiller

Janet Stern

Barry Trute

Jean-Francois Verrier

Anthony Woodell

Editor

Joyce Raftery

Graphic Designer

Satish Bettegowda

Publisher

Jobi Varghese

Oracle Database 10*g*: Administration Workshop I

Volume II • Student Guide

D17090GC31

Edition 3.1

December 2008

D57299

Authors

Tom Best

Maria Billings

Technical Contributors and Reviewers

Celia Antonio

Larry Baumann

Tammy Bednar

Howard Bradley

M.J. Bryksa

Sandra Cheevers

Steve Friedberg

Joel Goodman

John Hibbard

Magnus Isaksson

Sushma Jagannath

Christine Jeal

Steven Karam

Donna Keesling

Stella Kister

Pierre Labrousse

Stefan Lindblad

Dee Matishak

Paul Needham

Raza Siddiqui

James Spiller

Janet Stern

Barry Trute

Jean-Francois Verrier

Anthony Woodell

Editor

Joyce Raftery

Graphic Designer

Satish Bettegowda

Publisher

Jobi Varghese

Preface

Profile

Before You Begin This Course

Before you begin this course, you should have the following qualifications:

- Working experience with SQL

How This Course Is Organized

Oracle Database 10g: Administration Workshop I is an instructor-led course featuring lecture and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills introduced.

Suggested Next Courses

Oracle Database 10g: Administration Workshop II (D17092GC30)

Related Publications

Oracle Publications

Title	**Part Number**
Oracle Database 2 Day DBA 10g Release 2 (10.2)	B14196-01
Oracle Database Administrator's Guide 10g Release 2 (10.2)	B14231-01
Oracle Database Backup and Recovery Basics 10g Release 2 (10.2)	B14192-01
Oracle Database Concepts 10g Release 2 (10.2)	B14220-01
Oracle Database Licensing Information 10g Release 2 (10.2)	B14199-01
Oracle Database Net Services Administrator's Guide 10g Release 2 (10.2)	B14212-01
Oracle Database Net Services Reference 10g Release 2 (10.2)	B14213-01
Oracle Database New Features Guide 10g Release 2 (10.2)	B14214-01
Oracle Database Performance Tuning Guide 10g Release 2 (10.2)	B14211-01
Oracle Database PL/SQL Packages and Types Reference 10g Release 2 (10.2)	B14258-01
Oracle Database PL/SQL User's Guide and Reference 10g Release 2 (10.2)	B14261-01
Oracle Database Recovery Manager Quick Start Guide 10g Release 2 (10.2)	B14193-01
Oracle Database Recovery Manager Reference 10g Release 2 (10.2)	B14194-01
Oracle Database Security Guide 10g Release 2 (10.2)	B14266-01
Oracle Database SQL Quick Reference 10g Release 2 (10.2)	B14195-01
Oracle Database SQL Reference 10g Release 2 (10.2)	B14200-01

Additional Publications

- System release bulletins
- Installation and user guides
- *read.me* files
- International Oracle Users Group (IOUG) articles
- *Oracle Magazine*

Typographic Conventions

Typographic Conventions in Text

Convention	Element	Example
Bold	Emphasized words and phrases in Web content only	To navigate within this application, do **not** click the Forward and Back buttons.
Bold italic	Glossary term (if there is a glossary)	The ***algorithm*** inserts the new key.
Brackets	Key names	Press [Enter].
Caps and lowercase	Buttons, check boxes, application triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger to the ORD block. Open the Master Schedule window.
Angle brackets	Menu paths	Select File > Save.
Commas	Key sequences	Press and release the following keys one at a time: [Alt], [F], [D]
Courier new, case sensitive (default is lowercase)	Code output, directory names, file names, passwords, path names, user input, usernames	Code output: `debug.set ('I', 300);` Directory: `bin` (DOS), `$FMHOME` (UNIX) File name: Locate the `init.ora` file. Password: Use `tiger` as your password. Path name: Open `c:\my_docs\projects`. User input: Enter `300`. Username: Log in as `HR`.
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (*but* Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do *not* save changes to the database. For further information, see *Oracle Database SQL Reference 10g Release 1(10.1).* Enter *`user_id`*`@us.oracle.com`, where *`user_id`* is the name of the user.

Typographic Conventions (continued)

Typographic Conventions in Text (continued)

Convention	Element	Example
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select “Include a reusable module component” and click Finish. This subject is covered in the lesson titled, “Working with Objects.”
Uppercase	SQL column names, commands, functions, schemas, table names, database trigger names	Use the `SELECT` command to view information stored in the `LAST_NAME` column of the `EMPLOYEES` table.

Typographic Conventions in Code

Convention	Element	Example

Lowercase	Column names, table names, database trigger names	`SELECT last_name`
		`FROM employees;`
		
		CREATE OR REPLACE TRIGGER `secure_employees`

	Passwords	`CREATE USER scott` `IDENTIFIED BY tiger;`
	PL/SQL objects	`items.DELETE(3);`
Lowercase italic	Syntax variables	`CREATE ROLE` *`role`*
Uppercase	SQL commands and functions	`SELECT first_name` `FROM employees;`

Typographic Conventions (continued)

Typographic Conventions in Navigation Paths

This course uses simplified navigation paths, such as the following example, to direct you through Oracle applications.

Example:

Invoice Batch Summary

(N) Invoice > Entry > Invoice Batches Summary (M) Query > Find (B) Approve

This simplified path translates to the following:

1. (N) From the Navigator window, select Invoice > Entry > Invoice Batches Summary.
2. (M) From the menu, select Query > Find.
3. (B) Click the Approve button.

Notation:

(N) = Navigator	(I) = Icon
(M) = Menu	(H) = Hyperlink
(T) = Tab	(B) = Button

Contents

15 Performing Database Backups

16 Performing Database Recovery

17 Performing Flashback

1

Introduction

Course Objectives

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

- **Install, create, and administer Oracle Database 10*g***
- **Configure the database for an application**
- **Employ basic monitoring procedures**
- **Implement a backup and recovery strategy**
- **Move data between databases and files**

ORACLE

1-2

Course Objectives

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

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

Database administration does not end after the database is configured. You also learn how to protect your database by designing a backup and recovery strategy, and how to monitor the database to ensure that it operates smoothly.

Suggested Schedule

1. Introduction
2. Installation
3. DB Creation
4. Instance

5. Storage
6. Users
7. Schema
8. Data & Concurrency

9. Undo
10. Security
11. Network
12. Proactive Maintenance

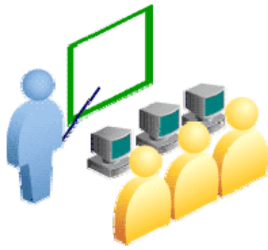
13. Performance
14. Backup & Recovery Concepts
15. Backup

16. Recovery
17. Flashback
18. Moving Data

Lesson Objectives

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

- Describe the course objectives
- Explain the Oracle Database 10*g* architecture



1-4

Oracle Products and Services

- Oracle databases
- Oracle Application Server
- Oracle applications
- Oracle Collaboration Suite
- Oracle Developer Suite
- Oracle services

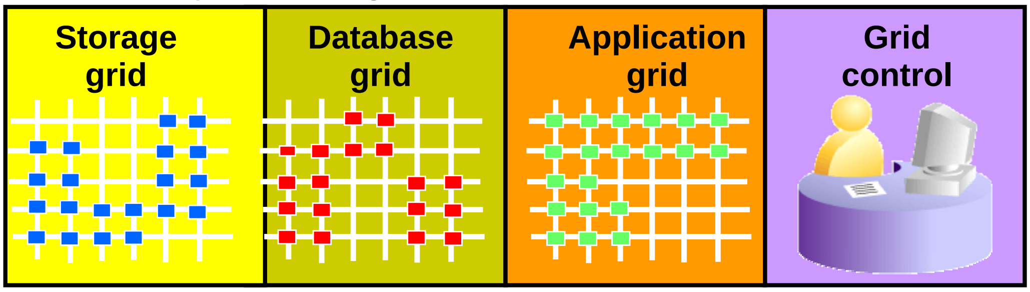
1-5

Oracle Products

- **Oracle databases:** The Oracle database is the first database that is designed for enterprise grid computing (the most flexible and cost-effective way to manage information and applications).
- **Oracle Application Server:** Oracle's Java 2 Platform, Enterprise Edition (J2EE)–certified server integrates everything that is 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 Developer Suite:** Oracle Developer Suite is a complete, integrated environment that combines application development and business intelligence tools.
- **Oracle services:** Services such as Oracle Consulting and Oracle University provide you with the necessary expertise for your Oracle projects. For useful links to a variety of resources, see the appendix titled "Next Steps, Continuing Your Education."

Oracle Database 10*g*: “*g*” Stands for Grid

- Global Grid Forum (GGF)
- Oracle’s grid infrastructure:
  – Low cost
  – High quality of service
  – Easy to manage



Automatic Storage Management | Real Application Clusters | Oracle Streams | Enterprise Manager Grid Control

Oracle Database 10*g*: “*g*” Stands for Grid

Global Grid Forum (GGF) is a standards body that develops standards for grid computing. It comprises a set of committees and working groups that focus on various aspects of grid computing. The committees and working groups are composed of participants from academia, the research community, and (increasingly) commercial companies. You can see the Web site of GGF at http://www.gridforum.org.

Oracle has created the grid computing infrastructure software that balances all types of workloads across servers and enables all those servers to be managed as one complete system. Grid computing can achieve the same very high level of reliability as mainframe computing because all components are clustered. But unlike mainframes and large UNIX symmetric multiprocessing (SMP) servers, a grid can be built with open system technologies, such as Intel processors and the Linux operating system, at a very low cost.

Oracle’s grid computing technology includes:

- Automatic Storage Management (ASM)
- Real Application Clusters (RAC)
- Oracle Streams
- Enterprise Manager Grid Control

Oracle Database 10*g*: "*g*" Stands for Grid (continued)

Automatic Storage Management spreads database data across all disks, creates and maintains a storage grid, and provides the highest input/output (I/O) throughput with minimal management costs. As disks are added or dropped, ASM redistributes the data automatically. (There is no need for a logical volume manager to manage the file system.) Data availability increases with optional mirroring, and you can add or drop disks online. For more information, see the lesson titled "Managing Database Storage Structures."

Oracle's **Real Application Clusters** runs and scales all application workloads on a cluster of servers and offers the following features:

- **Integrated clusterware:** This includes functionality for cluster connectivity, messaging and locking, cluster control, and recovery. It is available on all platforms that are supported by Oracle Database 10*g*.
- **Automatic workload management:** Rules can be defined to automatically allocate processing resources to each service both during normal operations and in response to failures. These rules can be dynamically modified to meet the changing business needs. This dynamic resource allocation within a database grid is unique to Oracle RAC.
- **Automatic event notification to the mid-tier:** When a cluster configuration changes, the mid-tier can immediately adapt to instance failover or availability of a new instance. This enables end users to continue working in the event of instance failover without the delays typically caused by network timeouts. In the event of new instance availability, the mid-tier can immediately start load balancing connections to that instance. Oracle Database 10*g* Java Database Connectivity (JDBC) drivers have the "fast connection failover" functionality that can be automatically enabled to handle these events.

Oracle Streams provides a unified framework for information sharing, combining message queuing, data replication, event notification, data warehouse loading, and publishing and subscribing functionality into a single technology. Oracle Streams can keep two or more data source copies synchronized when updates are applied at either site. It can automatically capture database changes, propagate the changes to subscribing nodes, apply changes, and detect and resolve data update conflicts. Oracle Streams can be used directly by applications as a message-queuing or workflow feature, enabling communications between applications in the grid.

Enterprise Manager Grid Control manages gridwide operations that include managing the entire stack of software, provisioning users, cloning databases, and managing patches. It can monitor the performance of all applications from the point of view of your end users. Grid Control views the performance and availability of the grid infrastructure as a unified whole rather than as isolated storage units, databases, and application servers. You can group hardware nodes, databases, and application servers into single logical entities and manage a group of targets as one unit.

Note: In this course, you use Enterprise Manager Database Console to manage one database at a time.

Oracle Database Architecture

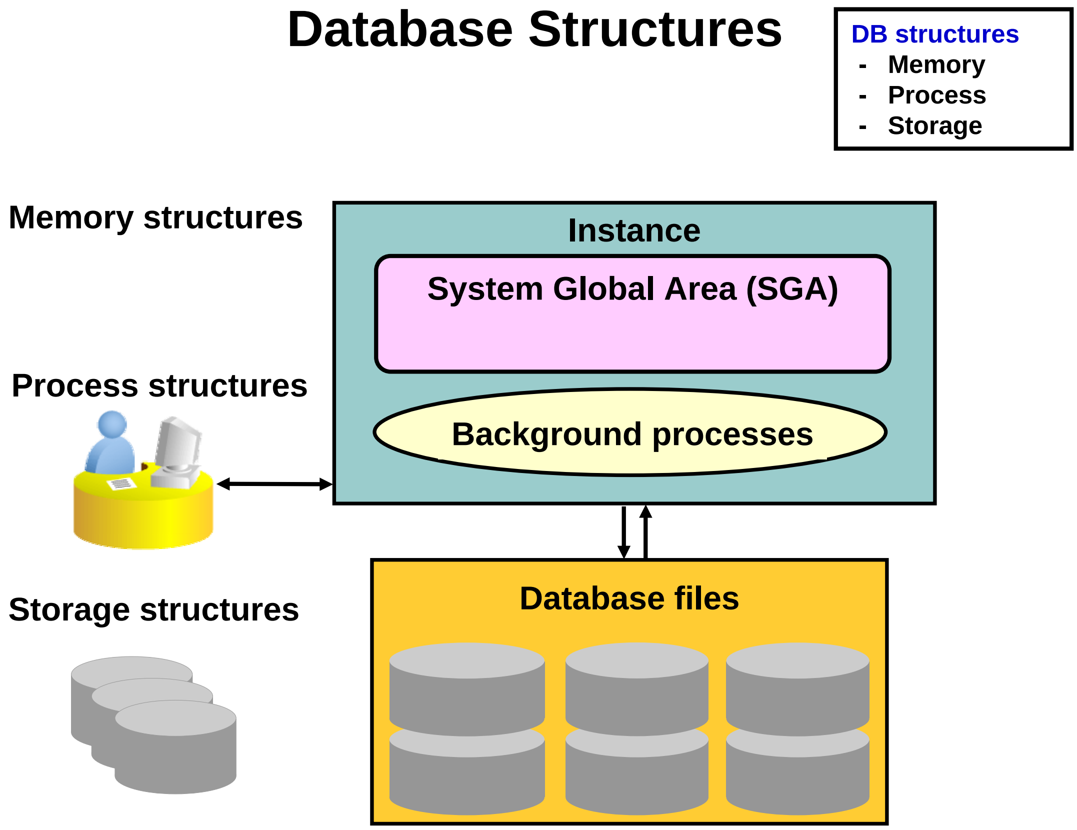
An Oracle server:

- Is a database management system that provides an open, comprehensive, integrated approach to information management
- Consists of an Oracle instance and an Oracle database

Oracle Database Architecture

The Oracle server is the key to information management. In general, an Oracle server must reliably manage a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this must be accomplished while delivering high performance. An Oracle server must also prevent unauthorized access and provide efficient solutions for failure recovery.

Database Structures



1-9

Database Structures

Each running Oracle database is associated with an Oracle instance. When a database is started on a database server, the Oracle software allocates a shared memory area called the System Global Area (SGA) and starts several Oracle background processes. This combination of the SGA and the Oracle processes is called an Oracle instance.

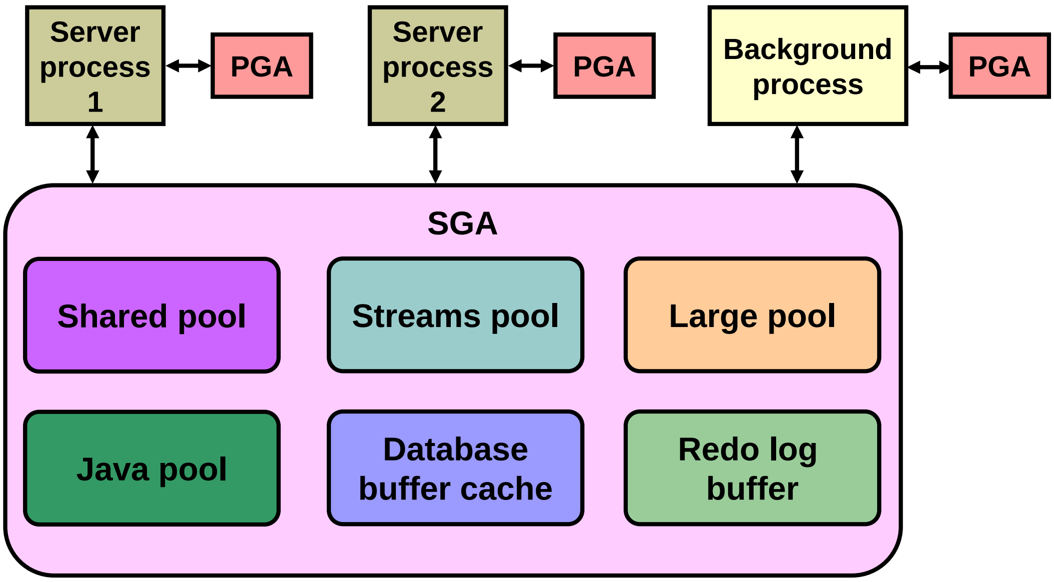
After starting an instance, the Oracle software associates the instance with a specific database. This is called mounting the database. The database is then ready to be opened, which makes it accessible to authorized users. Multiple instances can execute concurrently on the same computer, each accessing its own physical database.

You can look at the Oracle database architecture as various interrelated structural components.

An Oracle database uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of the computers that constitute the database server. Processes are jobs that work in the memory of these computers. A process is defined as a "thread of control" or a mechanism in an operating system that can run a series of steps.

Oracle Memory Structures

DB structures
> Memory
Process
Storage



ORACLE

1-10

Oracle Memory Structures

The basic memory structures associated with an Oracle instance include the following:

- **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 SGA is a 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 the disk
- **Shared pool:** Caches various constructs that can be shared among users
- **Large pool:** Is an optional area that provides large memory allocations for certain large processes, such as Oracle backup and recovery operations, and I/O server processes
- **Java pool:** Is used for all session-specific Java code and data within the Java Virtual Machine (JVM)
- **Streams pool:** Is used by Oracle Streams

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

Oracle Memory Structures (continued)

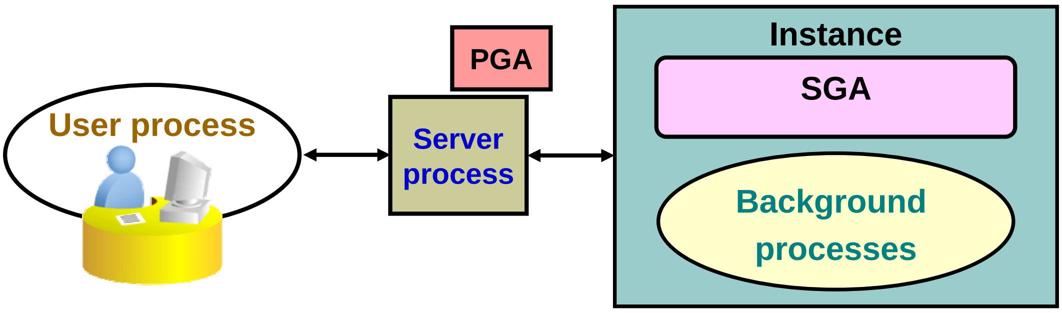
A Program Global Area (PGA) is a memory region that contains data and control information for each server process. An Oracle server process services a client's requests. Each server process has its own private PGA that is created when the server process is started. Access to the PGA is exclusive to that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

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 Oracle database uses initialization parameters to create and configure memory structures. For example, the SGA_TARGET parameter specifies the total size of the SGA. If you set `SGA_TARGET` to 0, Automatic Shared Memory Management is disabled.

Process Structures

DB structures
Memory
> Process
Storage

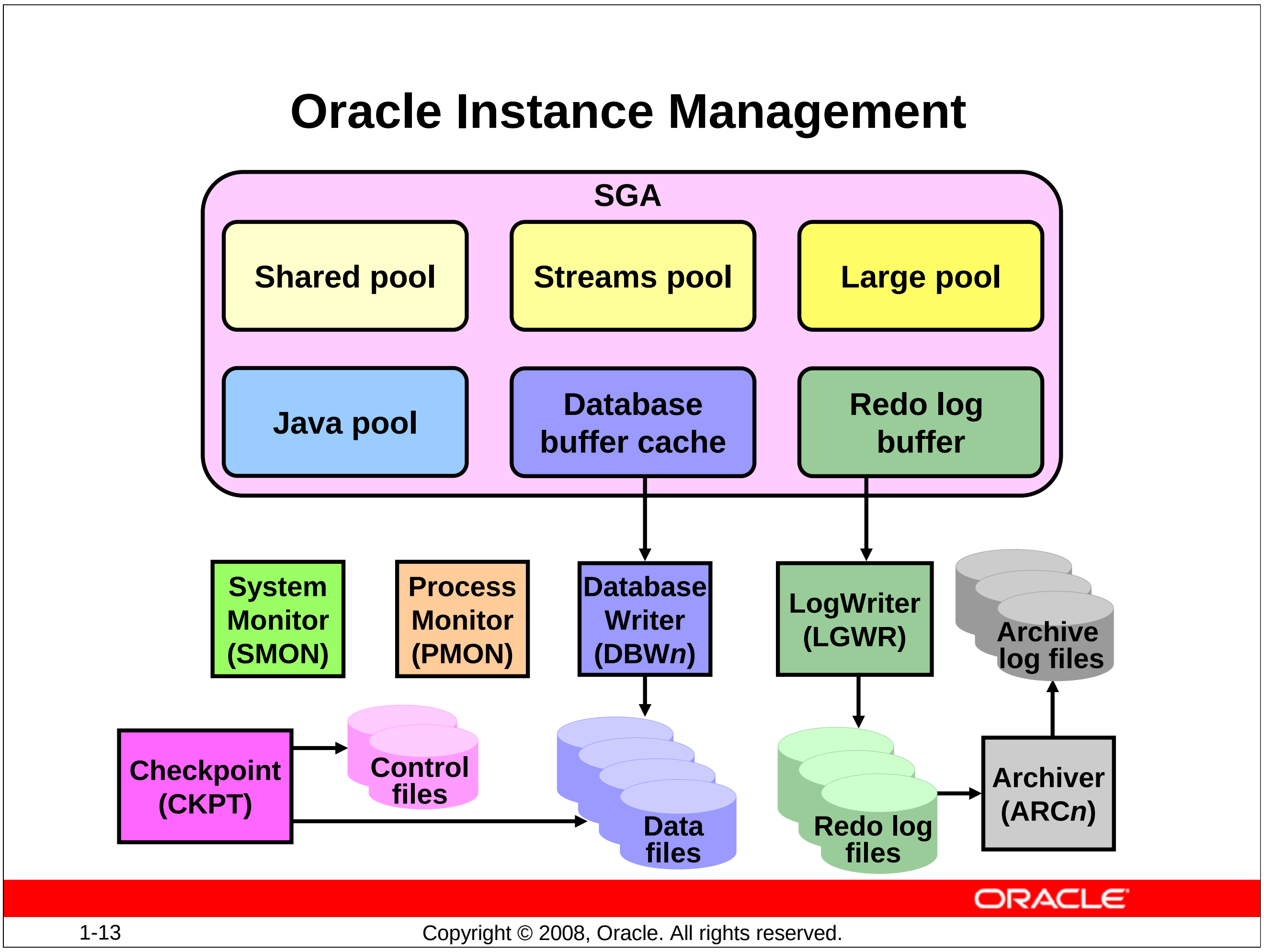


- **User process:** Is started at the time a database user requests a connection to the Oracle server
- **Server process:** Connects to the Oracle instance and is started when a user establishes a session
- **Background processes:** Are started when an Oracle instance is started

1-12

Process Structures

When you invoke an application program or an Oracle tool, such as Enterprise Manager, the Oracle server creates a server process to execute the commands issued by the application. The Oracle server 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 perform other required tasks. Which background processes are present depends on the features that are being used in the database.

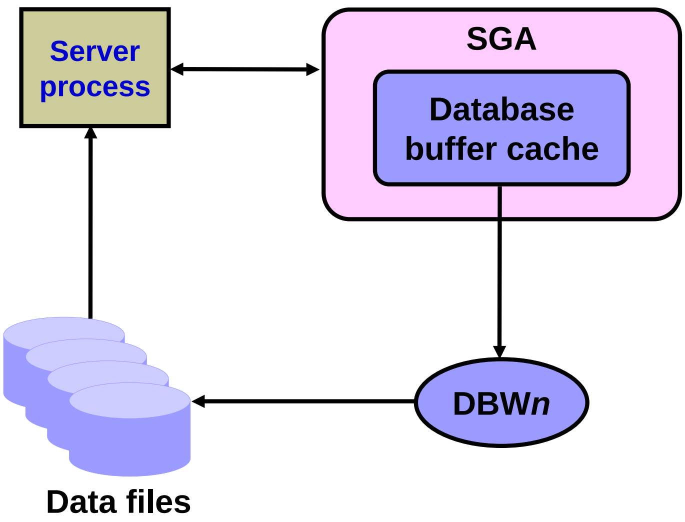


Oracle Instance Management

An Oracle database server consists of an Oracle database and an Oracle instance. An Oracle instance is made up of memory structures, 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 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 data files on the disk
- **Checkpoint (CKPT):** Updates all the data files and control files of the database to indicate the most recent checkpoint
- **LogWriter (LGWR):** Writes redo log entries to the disk
- **Archiver (ARC*n*):** Copies redo log files to the archival storage when a log switch occurs

Server Process and Database Buffer Cache



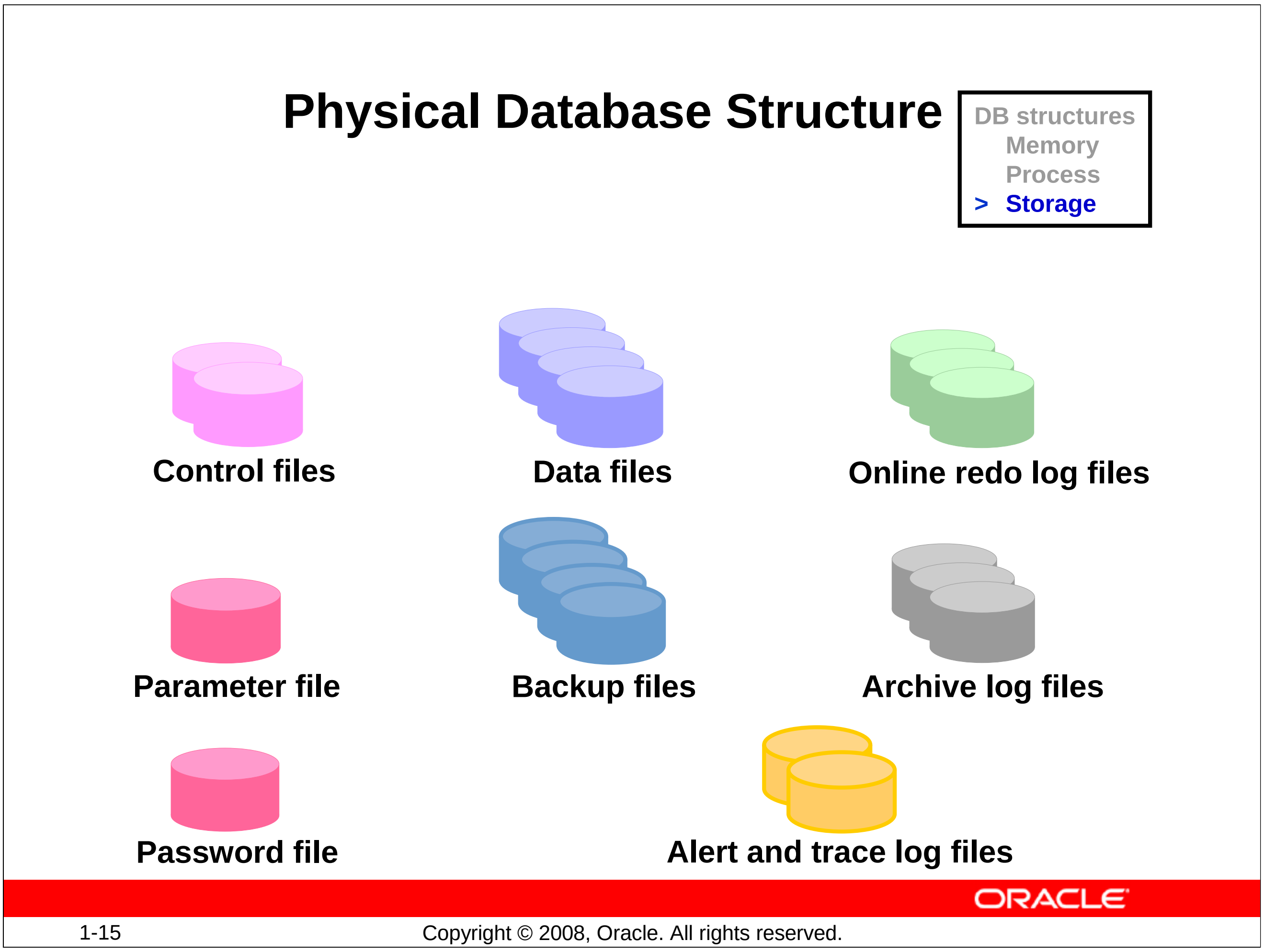
Buffers:
- Pinned
- Clean
- Free or unused
- Dirty

Server Process and Database Buffer Cache

When a query is processed, the Oracle server process looks in the database buffer cache for any blocks that it needs. If the block is not found in the database buffer cache, the server process reads the block from the data file and places a copy in the database buffer cache. Because subsequent requests for the same block may find the block in memory, the requests may not require physical reads. The Oracle server uses the least recently used algorithm to age out buffers that have not been accessed recently to make room for new blocks in the database buffer cache.

Buffers in the buffer cache can be in one of the following four states:

- **Pinned:** Multiple sessions are kept from writing to the same block at the same time. Other sessions wait to access the block.
- **Clean:** The buffer is now unpinned and is a candidate for immediate aging out, if the current contents (data block) are not referenced again. Either the contents are in sync with the block contents stored on the disk or the buffer contains a consistent read (CR) snapshot of a block.
- **Free or unused:** The buffer is empty because the instance has just started. This state is very similar to the clean state, except that the buffer has not been used.
- **Dirty:** The buffer is no longer pinned but the contents (data block) have changed and must be flushed to the disk by DBW*n* before it can be aged out.



Physical Database Structure

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

- **Control files:** Contain data about the database itself (that is, physical database structure information). These files are critical to the database. Without them, you cannot open data files to access the data within the database.
- **Data files:** Contain the user or application 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, then the instance can recover the database with the information in these files.

The following additional files are important to the successful running of the database:

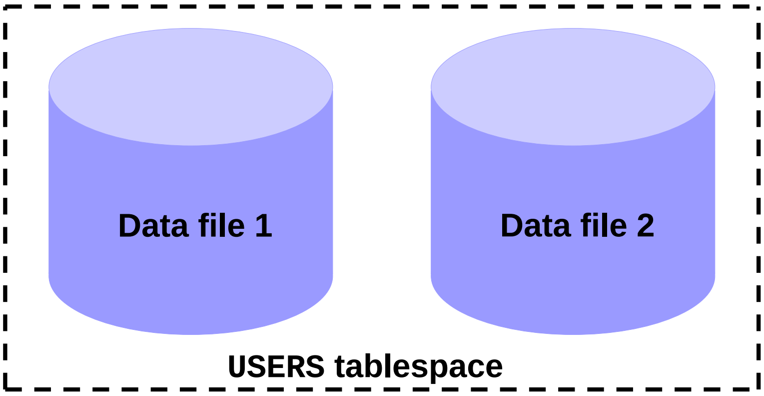
- **Parameter file:** Is used to define how the instance is configured when it starts up
- **Password file:** Allows users to connect remotely to the database and perform administrative tasks
- **Backup files:** Are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.
- **Archive log files:** Contain an ongoing history of the data changes (redo) that are generated by the instance. Using these files and a backup of the database, you can recover a lost data file. That is, archive logs enable the recovery of restored data files.

Physical Database Structure (continued)

- **Trace files:** Each server and background process can write to an associated trace file. When an internal error is detected by a process, the process dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, whereas other information is for Oracle Support Services.
- **Alert log files:** These are special trace files. They are also known as alert logs. The alert log of a database is a chronological log of messages and errors. Oracle recommends that you review these files.

Tablespaces and Data Files

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



Tablespaces and Data Files

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 the bigfile tablespaces, which are tablespaces with a single but very large (up to 4 billion data blocks) data file. The traditional smallfile tablespaces (which are the default) can contain multiple data files, but the files cannot be as large. For more information about the bigfile tablespaces, see the *Database Administrator's Guide*.

SYSTEM and SYSAUX Tablespaces

- The SYSTEM and SYSAUX tablespaces are mandatory tablespaces.
- They are created at the time of database creation.
- They must be online.
- The SYSTEM tablespace is used for core functionality (for example, data dictionary tables).
- The auxiliary SYSAUX tablespace is used for additional database components (such as the Enterprise Manager Repository).

1-18

SYSTEM and SYSAUX Tablespaces

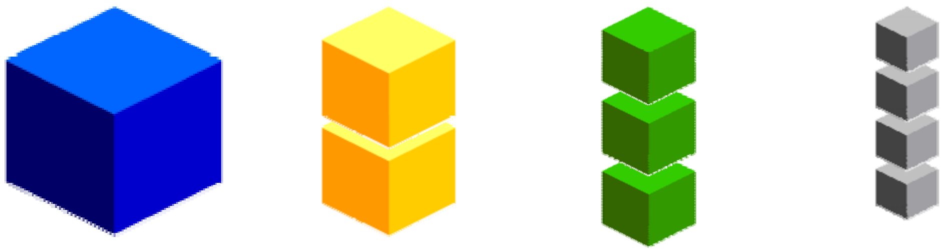
Each Oracle database contains a SYSTEM tablespace and a SYSAUX tablespace. They are automatically created when the database is created. The system default is to create a smallfile tablespace. You can also create bigfile tablespaces, which enable the Oracle database to manage ultralarge files (resulting in a database up to 8 exabytes in size).

A tablespace can be online (accessible) or offline (not accessible). The SYSTEM tablespace is always online when the database is open. It stores tables that support the core functionality of the database, such as the data dictionary tables.

The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace. The SYSAUX tablespace stores many database components, and it must be online for the correct functioning of all database components.

Segments, Extents, and Blocks

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



Segments, Extents, and Blocks

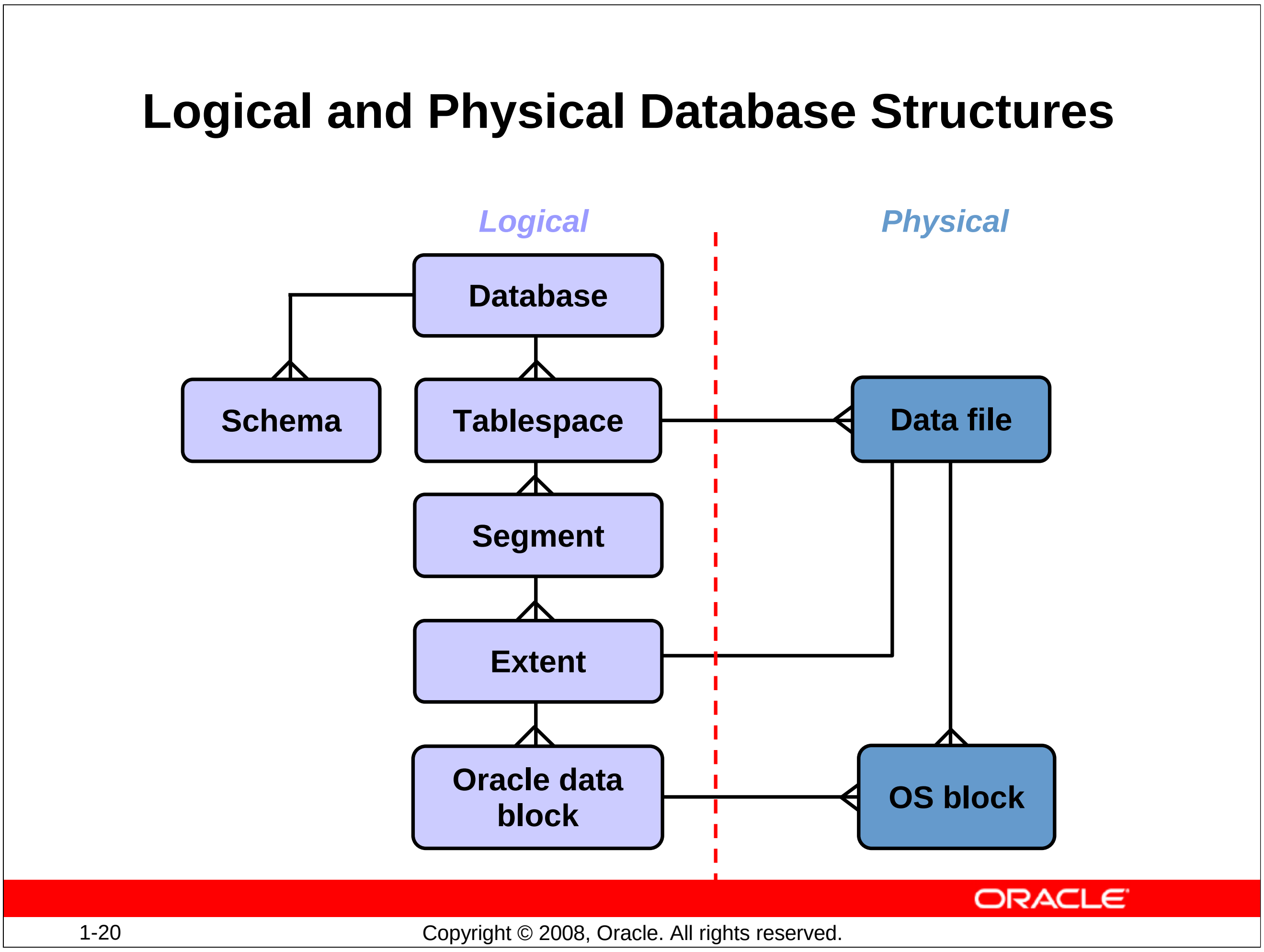
Database objects, such as tables and indexes, are stored as segments in tablespaces. Each segment contains one or more extents. An extent consists of contiguous data blocks, which means that each extent can exist only in one data file. Data blocks are the smallest unit of I/O in the database.

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

The size of the data block can be set at the time of the creation of the database. The default size of 8 KB is adequate for most databases. If your database supports a data warehouse application that has large tables and indexes, then a larger block size may be beneficial.

If your database supports a transactional application where reads and writes are random, then specifying a smaller block size may be beneficial. The maximum block size depends on your OS. The minimum Oracle block size is 2 KB and should rarely (if ever) be used.

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



Logical and Physical Database Structures

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

Tablespaces

A database is divided into logical storage units called tablespaces, which group related logical structures together. For example, tablespaces commonly group all of an application's objects to simplify some administrative operations. You may have a tablespace for application data and an additional one for application indexes.

Databases, Tablespaces, and Data Files

The relationship among databases, tablespaces, and data files is illustrated in the slide. 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. If it is a `TEMPORARY` tablespace, instead of a data file, then the tablespace has a temporary file.

Logical and Physical Database Structures (continued)

Schemas

A schema is a collection of database objects that are owned by a database user. Schema objects are the logical structures that directly refer to the database's data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, schema objects include everything that your application creates in the database.

Data Blocks

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

Extents

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

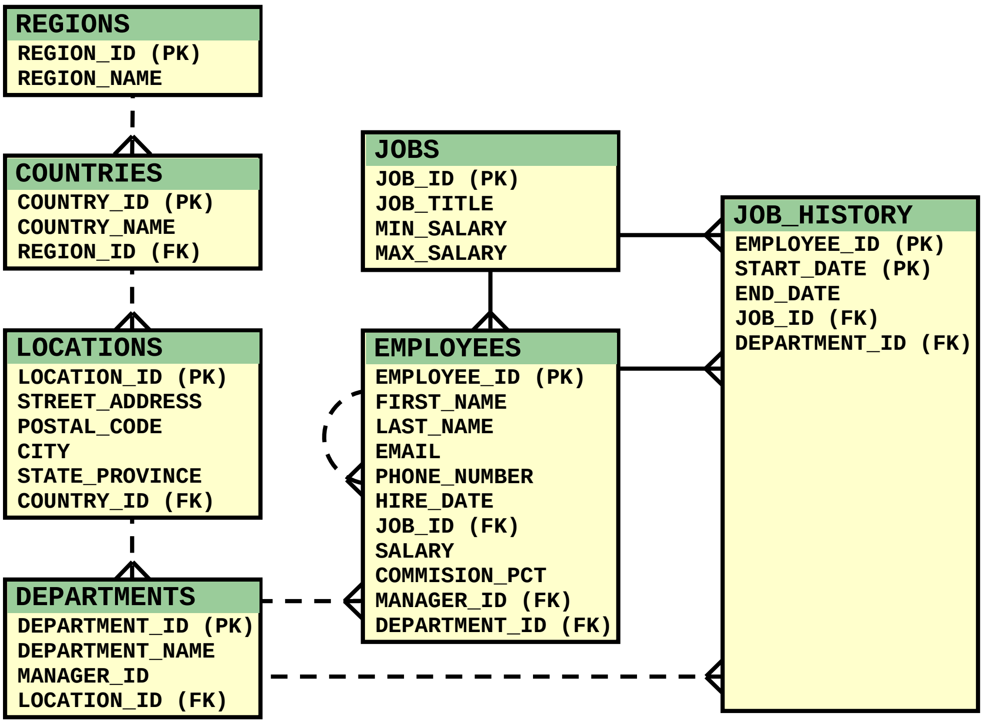
Segments

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

- **Data segments:** Each nonclustered, non-indexed-organized table has a data segment. All of the table's data is stored in the extents of its data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- **Index segments:** Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
- **Undo segments:** One `UNDO` tablespace is created by the database administrator to temporarily store *undo* information. The information in an undo segment is used to generate read-consistent database information and, during database recovery, to roll back uncommitted transactions for users.
- **Temporary segments:** Temporary segments are created by the Oracle database when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the instance for future use. Specify a default temporary tablespace for every user or a default temporary tablespace, which is used databasewide.

The Oracle database dynamically allocates space. When the existing extents of a segment are full, additional extents are added. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on the disk.

Course Examples: The HR Schema



1-22

Course Examples: The HR Sample Schema

The examples used in this course are from a human resources (HR) application, which can be created as part of the starter database.

The following are some principal business rules of the HR application:

- Each department may be the employer of one or more employees. Each employee may be assigned to one and only one department.
- Each job must be a job for one or more employees. Each employee must be currently assigned to one and only one job.
- When an employee changes his or her department or job, a record in the `JOB_HISTORY` table records the start and end dates of the past assignments.
- `JOB_HISTORY` records are identified by a composite primary key (PK): the `EMPLOYEE_ID` and the `START_DATE` columns.

Notation: PK = Primary Key, FK = Foreign Key

Solid lines represent mandatory foreign key (FK) constraints and dashed lines represent optional FK constraints.

The `EMPLOYEES` table also has an FK constraint with itself. This is an implementation of the business rule: Each employee may be reporting directly to one and only one manager. The FK is optional because the top employee does not report to another employee.

Database Architecture: Summary of Structural Components

- **Memory structures:**
  - System Global Area (SGA): Database buffer cache, redo buffer, and various pools
  - Program Global Area (PGA)
- **Process structures:**
  - User process and Server process
  - Background processes: SMON, PMON, DBW*n*, CKPT, LGWR, ARC*n*, and so on
- **Storage structures:**
  - Logical: Database, schema, tablespace, segment, extent, and Oracle block
  - Physical: Files for data, parameters, redo, and OS block

ORACLE

1-23

Database Architecture: Summary of Structural Components

In this lesson, you learned at a high level about the structural components of the Oracle database: memory, process, and storage structures. More details are covered in the following lessons.

Summary

In this lesson, you should have learned how to:

- Describe the course objectives
- Explain the Oracle Database 10*g* architecture

1-24

2

Installing the Oracle Database Software

Objectives

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

- Describe your role as a database administrator (DBA), and explain typical tasks and tools
- Plan an Oracle database installation
- Use Optimal Flexible Architecture (OFA)
- Install the Oracle software by using Oracle Universal Installer (OUI)

ORACLE

2-2

Tasks of an Oracle Database Administrator

A prioritized approach for designing, implementing, and maintaining an Oracle database involves the following tasks:

1. **Evaluating the database server hardware**
2. **Installing the Oracle software**
3. **Planning the database and security strategy**
4. **Creating, migrating, and opening the database**
5. **Backing up the database**
6. **Enrolling system users and planning for their Oracle Network access**
7. **Implementing the database design**
8. **Recovering from database failure**
9. **Monitoring database performance**

ORACLE

2-3

Tasks of an Oracle Database Administrator

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

You must 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. These tasks should be performed within the framework of a security strategy.

As you proceed through the lessons in this course, you 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.

In this lesson, you focus on installation. For this core task, consider the following subtasks:

- Understand how the installation fits into the overall technical architecture of an organization.
- Review (and update) capacity plans.
- Choose the database software (required version and options).
- Ensure that system requirements are met for all chosen elements.

Tools Used to Administer an Oracle Database

- Oracle Universal Installer
- Database Configuration Assistant
- Database Upgrade Assistant
- Oracle Net Manager
- Oracle Enterprise Manager
- SQL*Plus and *i*SQL*Plus
- Recovery Manager
- Oracle Secure Backup
- Data Pump
- SQL*Loader
- Command-line tools

ORACLE

Tools Used to Administer an Oracle Database

You can use the following tools for installation and upgrade:

- **Oracle Universal Installer (OUI):** Oracle Universal Installer installs your Oracle software and options. It can automatically launch the Database Configuration Assistant (DBCA) to create a database.
- **Database Configuration Assistant (DBCA):** It creates a database from Oracle-supplied templates. It enables you to copy a preconfigured seed database. Alternatively, you can create your own database and templates.
- **Database Upgrade Assistant (DBUA):** This tool guides you through the upgrade of your existing database to a new Oracle release.
- **Oracle Net Manager:** This is used to configure network connectivity for your Oracle databases and applications.

Tools Used to Administer an Oracle Database (continued)

The following tools are used to manage your Oracle instance and database:

- **Oracle Enterprise Manager (EM):** EM combines a graphical console, agents, common services, and tools to provide an integrated and comprehensive system management platform for managing Oracle products. After you have installed the Oracle software, created or upgraded a database, and configured the network, you can use Enterprise Manager as the single interface for managing your database. In addition to providing a Web-based user interface for executing SQL commands, it seamlessly interfaces with other Oracle components that are used to administer your database (for example, Recovery Manager and Scheduler).

  The three main Enterprise Manager tools that are used to administer an Oracle database are:
  - Enterprise Manager Database Console: Used to administer one database
  - Enterprise Manager Grid Control: Used to administer many databases at the same time
  - Enterprise Manager Java Console: Used to access tools that are not Web enabled
- **SQL*Plus:** SQL*Plus is the standard command-line interface for managing your database.
- ***i*SQL*Plus:** *i*SQL*Plus is a browser-based interface to an Oracle database.
- **Recovery Manager (RMAN):** RMAN is an Oracle tool that provides a complete solution for the backup, restoration, and recovery needs of the entire database or of specific database files.
- **Oracle Secure Backup** provides tape backup management for the Oracle ecosystem, which includes:
  - Oracle database protection to tape through integration with Recovery Manager
  - Seamless support of Oracle Real Application Clusters (RAC)
  - Central administration of distributed clients and media servers including Oracle Application Servers, Oracle Collaboration Suites, Oracle home, and binaries
- **Data Pump:** Data Pump enables the high-speed transfer of data from one database to another. For example, you may want to export a table and import it into another database.
- **SQL*Loader:** The SQL*Loader utility enables the loading of data from an external file into an Oracle database. It is one of several Oracle utilities that you can use to load data into database tables.
- **Command-line tools:**
  - To administer Enterprise Manager, use:
    ```
    emctl start | status | set | stop
    ```
  - To stop and start *i*SQL*Plus, use:
    ```
    isqlplusctl start | stop
    ```
  - To administer the listener, use:
    ```
    lsnrctl help | start | status | stop
    ```

Installation: System Requirements

- Memory requirements:
  - 1 GB for the instance with Database Control
- Disk space requirements:
  - 1.5 GB of swap space
  - 400 MB of disk space in the `/tmp` directory
  - Between 1.5 GB and 3.5 GB for the Oracle software
  - 1.2 GB for the preconfigured database (optional)
  - 2.4 GB for the flash recovery area (optional)
- Operating system: See documentation.

ORACLE

2-6

Installation: System Requirements

- A standard installation can be completed on a computer with 1 GB of RAM and 1.5 GB of swap space or larger.
- Depending on the activity level of the machine on which you are installing the Oracle database software, the standard installation can complete in 20 minutes or less.
- Some installation details:
  - Oracle Database 10*g* ships only one seed database template.
  - Duplicated files are removed.
  - Many other products and demonstrations are installable from additional CDs.

The hardware requirements listed in the slide are the minimal requirements for Linux. The minimum for Windows is 256 MB, with 512 MB recommended. Your installation may have additional requirements (especially disk space).

Note: An Enterprise Edition installation type that includes a standard seed database is referred to as a "standard installation."

Checking the System Requirements

```
[oracle@EDRSR4P1 solutions]$ cd /stage/Disk1
[oracle@EDRSR4P1 Disk1]$ ls
doc  install  response  runInstaller  stage  welcome.html
[oracle@EDRSR4P1 Disk1]$ ./runInstaller
Starting Oracle Universal Installer...

Checking installer requirements...

Checking operating system version: must be redhat-3, SuSE-9, redhat-4, UnitedLinux-1.0, asianux-1 or asianux-2
                                      Passed


All installer requirements met.

Preparing to launch Oracle Universal Installer from /tmp/OraInstall2005-10-18_02-17-50PM. Please wait ...[oracle@EDRSR4P1 Disk1]$
```

- **Adequate temporary space**
- **64-bit versus 32-bit issues**
- **Checks for the correct operating system (OS)**
- **OS patch level**
- **System packages**
- **System and kernel parameters**
- **X Server permissions**
- **Sufficient swapping**
- **Nonempty `ORACLE_HOME`**

2-7

Checking the System Requirements

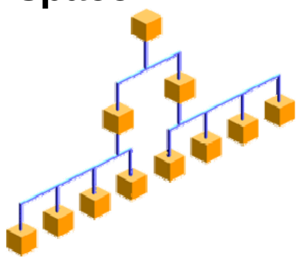
The Oracle Database 10*g* installation automates most of the prerequisite checks:

- Adequate temporary space is checked for. It is determined what the minimum temporary space requirements are for installation and configuration, and those requirements are validated during the installation process.
- 64-bit installations are prevented from being installed into Oracle homes with 32-bit software already installed (and vice versa).
- On the Linux platform, RedHat-3.0, 4.0, Asianux 1.0, 2.0, and SUSE Linux E.S. 9.0 are certified.
- The installation process checks whether all the required OS patches are installed.
- The installation process checks whether all the required system and kernel parameters are set correctly.
- The installation process verifies that the `DISPLAY` environment variable is set and that the user has sufficient permissions to display to the specified `DISPLAY`.
- The installation process verifies that the system has sufficient swapping set.
- The installation process verifies that the Oracle home into which the new installation is being performed is either empty or is one of a handful of supported releases on top of which Oracle Database 10*g* can be installed, and that they are registered in the Oracle inventory.

Optimal Flexible Architecture (OFA)

OFA is designed to:

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



2-8

Optimal Flexible Architecture (OFA)

OFA is a method for configuring the Oracle database and other databases. OFA takes advantage of the capabilities of the OS and disk subsystems 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.

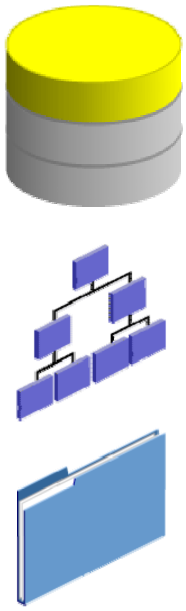
OFA is designed to:

- Organize large amounts of complicated software and data on the 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 about the goals and implementation of OFA, refer to the *Oracle Installation Guide for UNIX Systems*.

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`**



2-9

Using Optimal Flexible Architecture

At the core of OFA is a naming scheme that 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 by using the `/`*`pm`* syntax, 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. The examples of mount points are `/u01` and `/u02`.

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

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

Software directories syntax: Store each version of the Oracle software in a directory matching the pattern: `/`*`pm`*`/`*`h`*`/`*`u`*`/product/`*`v`*. Here, `product` is a literal and *`v`* is a variable for the version number. This syntax helps to enable the OFA feature of simultaneously executing multiple versions of application software. An OFA-compliant installation of the Oracle Database 10*g* version 10.2.0 looks like:

```
/u01/app/oracle/product/10.2.0
```

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/`. Here, `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. The 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/controln.ctl`
- Redo log files: `/pm/q/d/redon.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 the 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 the *d* database in the `/pm/q/d/` path.

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 (by default, `ORCL`)
- `NLS_LANG`: The language, territory, and client character set settings

2-11

Setting Environment Variables

There are many Oracle environment variables, and those 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 before the installation, you can 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, as shown in this example:

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

- `ORACLE_HOME`: Specifies the directory containing the Oracle software. It is a directory path, as shown in this example:

  ```
  $ORACLE_BASE/product/10.2.0/db_1
  ```

- `ORACLE_SID`: The initial instance name (by default, `ORCL`). 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 National Language Support (NLS) settings for a session in the form of *language_territory.character set*. For example, a setting of:

  ```
  AMERICAN_DENMARK.WE8MSWIN1252
  ```

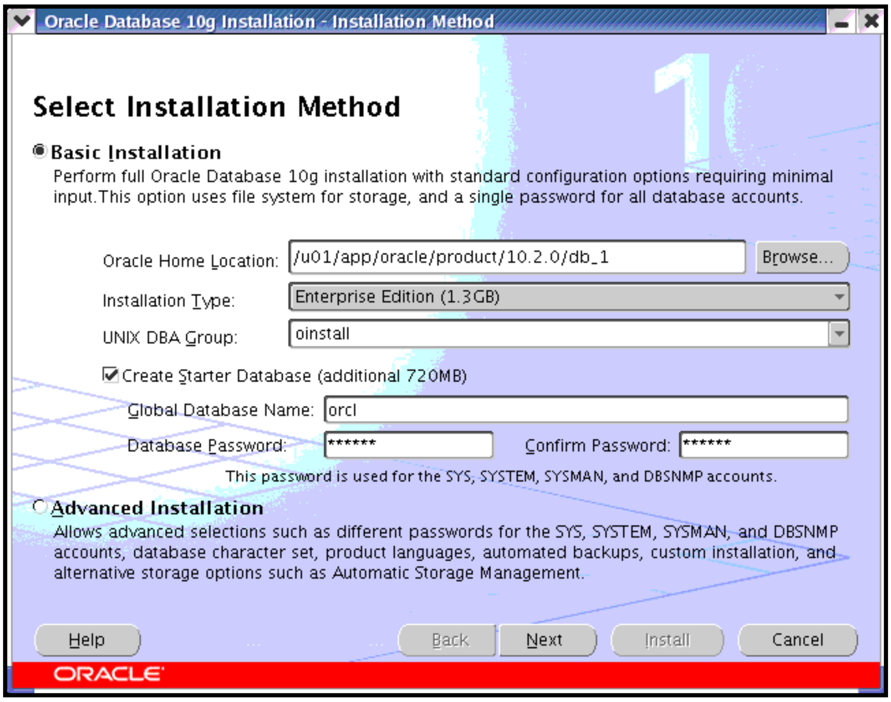
  This sets the session to use the AMERICAN language for Oracle messages, alphabetical sorting sequence, day names, and month names. The territory is DENMARK, which sets the time format, date format, and numeric and monetary conventions. The character set of WE8MSWIN1252 instructs Oracle Net to convert 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 using:

  ```
  select * from nls_session_parameters;
  ```

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

  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 the 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.

Oracle Universal Installer (OUI)



2-13

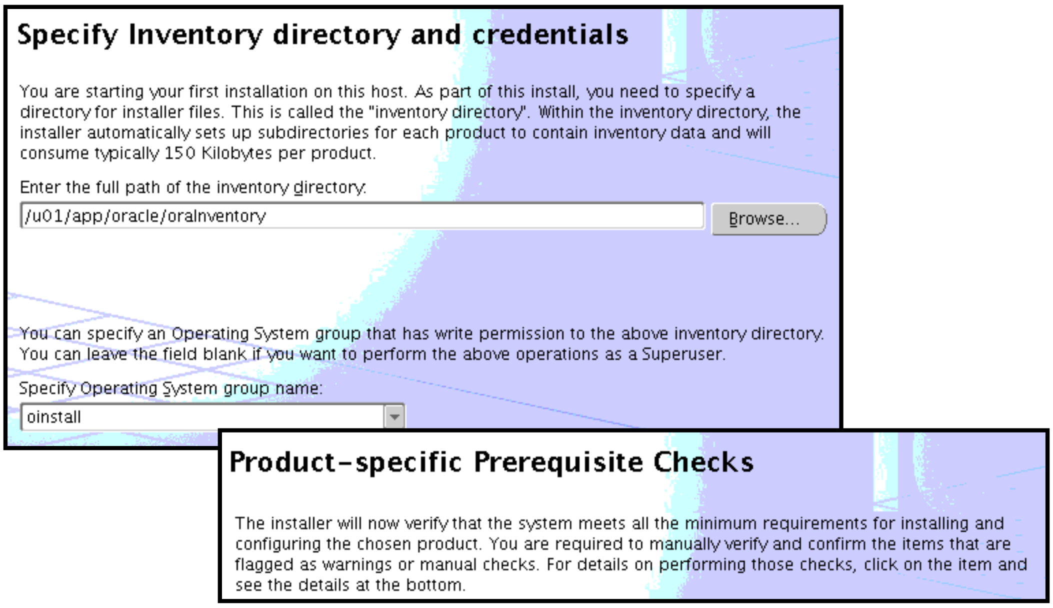
Oracle Universal Installer (OUI)

Oracle Universal Installer (OUI) 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. OUI 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
- NLS or globalization support
- Support for distributed installations
- Unattended “silent” installations that use response files

In Windows: Insert the Oracle database installation medium, navigate to the `client` directory and double-click `setup.exe` to start OUI. After the Welcome page, select your installation type: Instant Client, Administrator, Runtime, or Custom.

Installing the Oracle Software



ORACLE

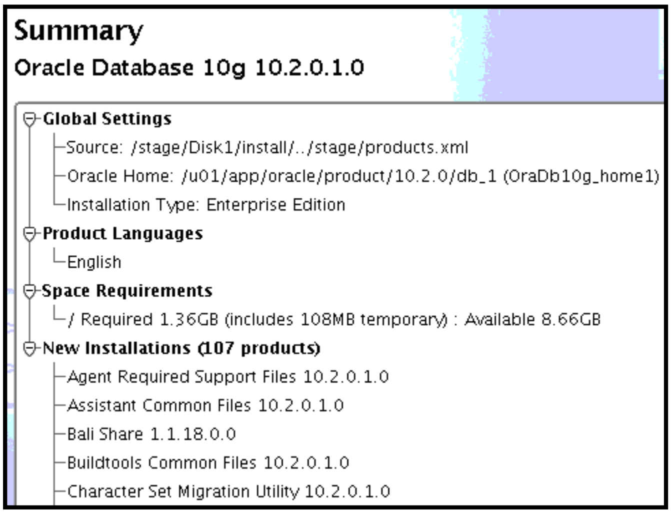
2-14

Installing the Oracle Software

You can install the Oracle software by using OUI, as follows:

1. Log on to your computer as a member of the administrative group that is authorized to install the Oracle software and to create and manage the database.
2. Insert the distribution CD for the database into your CD drive, or navigate to the Oracle database staging location.
3. Start OUI. In an XTerm window on Linux, enter `./runInstaller`. The Oracle Universal Installer page appears.
4. Navigate the OUI pages and specify your preinstallation settings according to your installation plan.
5. With the initial information, OUI executes prerequisite checks.

Database Configuration Options



2-15

Database Configuration Options

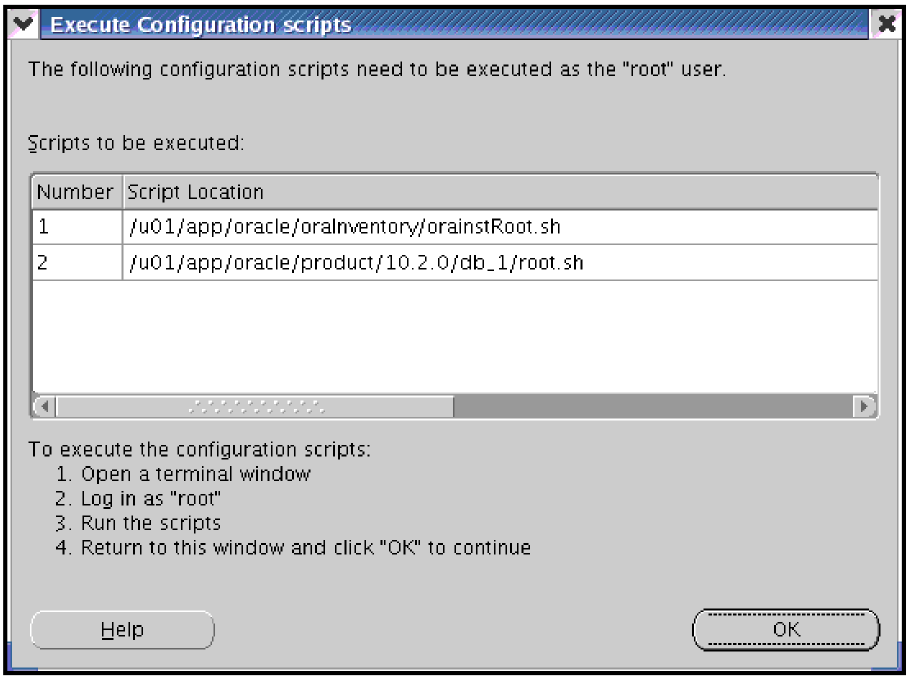
Your installation process continues:

6. Navigate through the OUI pages and specify your database configuration options. OUI displays a summary of your installation choices.
7. Click Install to begin your installation of the Oracle software.

If you chose to create a starter database as part of the installation, then OUI invokes all of these configuration assistants:

- **Oracle Net Configuration Assistant:** This configures basic network components during installation, including:
  - Listener names and protocol addresses
  - Naming methods that 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 (DBCA):** This creates the starter database that you selected. When this configuration assistant finishes, you can unlock accounts and change passwords.
- ***i*SQL*Plus Configuration Assistant:** This configures the Oracle Application Server Containers for J2EE (OC4J) instance, which is used by *i*SQLPlus, and other tools to connect to the Oracle database.

Executing Configuration Scripts



Executing Configuration Scripts

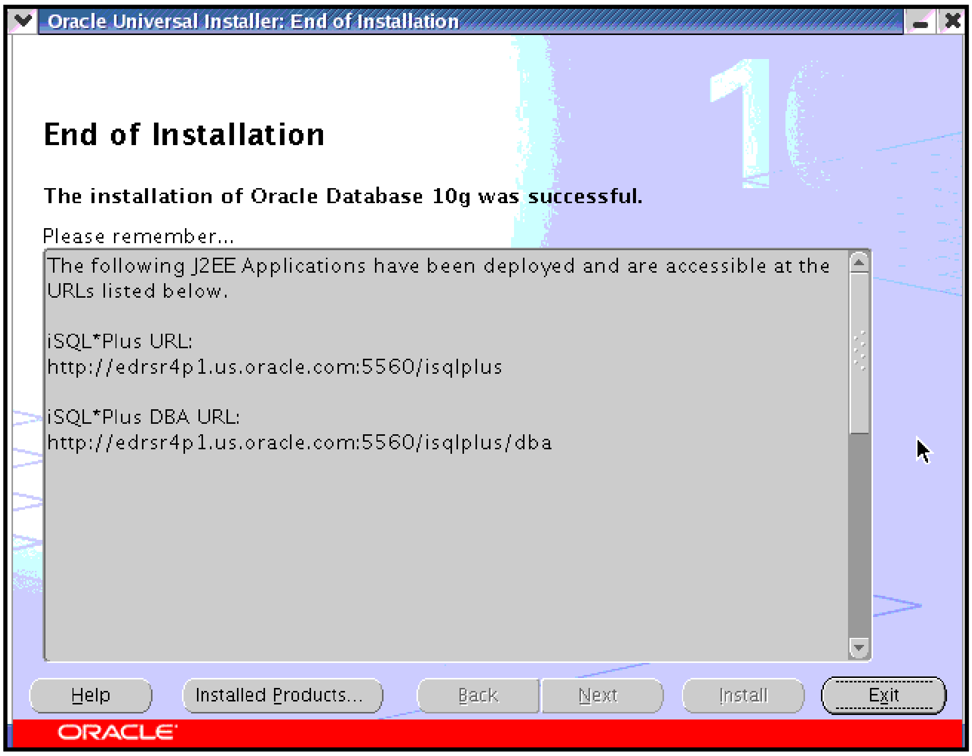
Your installation process continues:

8. When prompted during a Linux or UNIX installation, execute additional configuration scripts as the `root` user. In an XTerm window, enter:

```
$ su
# password: oracle <root password, does not appear in the window>
# cd /u01/app/oracle/oraInventory
# ./orainstRoot.sh
# cd /u01/app/oracle/product/10.2.0/db_1
# ./root.sh
```

9. Accept the default for the local `bin` directory during a Linux or UNIX installation. When the scripts are finished, exit all related accounts and windows to allow the installation to complete.

Completing Your Installation



2-17

Completing Your Installation

10. When your installation process comes to an end, note the URLs for future use.

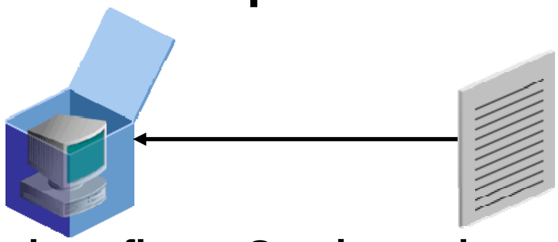
Advanced Installation Options

- Database storage options:
  - File system
  - Automatic Storage Management
  - Raw devices
- Database management options:
  - Enterprise Manager Grid Control
  - Enterprise Manager Database Control
- Database backup and recovery options
- E-mail notification options
- Cluster Ready Services
- Cloning

Advanced Installation Options

- With OUI, you can create configurations that use Automatic Storage Management.
- You can install and configure the Enterprise Manager (EM) framework. Oracle Enterprise Manager Database Control is installed in the same Oracle home as the database and is configured to run on a stand-alone OC4J instance. You have to perform a separate installation to get EM central management capabilities.
- If you choose to use Oracle Enterprise Manager Database Control, you can optionally configure the database to use the Oracle-recommended default backup strategy.
- If you choose to use Oracle Enterprise Manager Database Control during the installation, you can configure Enterprise Manager to send e-mail alerts to an e-mail address that you specify. These alerts can include issues such as disk space reaching a critical limit or a database shutting down unexpectedly.
- The Oracle Database 10*g* installation supports RAC features, particularly the installation of Cluster Ready Services (CRS).
- Oracle homes can be cloned by using the Enterprise Configuration Management tool. This tool enables users to create clone requests and then schedule and process them. This tool is available via EM Grid Control.

Installation Option: Silent Mode



To install and configure Oracle products with OUI in silent mode, perform the following steps:

1. **Create the `oraInst.loc` file, if it does not already exist.**
2. **Prepare a response file based on file templates that are delivered with the Oracle software.**
3. **Record a response file:**
   `.runInstaller –record -destinationFile <filename>`
4. **Run OUI in silent or suppressed mode.**
5. **If required, run NetCA and DBCA in silent mode.**

2-19

Installation Option: Silent Mode

To install and configure Oracle products by using OUI in silent or suppressed mode, perform these steps:

1. Create the `oraInst.loc` file, if it does not already exist. Most likely, the file is already in `/etc`, if you previously installed the Oracle software.
2. Prepare a response file. File templates for each product and installation type are provided, such as `enterprise.rsp`, `standard.rsp`, and `netca.rsp`.
3. You can use OUI in interactive mode to record a response file that you can edit and then use to complete silent-mode or suppressed-mode installations. Create the response file under Linux and UNIX with the following command:

   ```
   .runInstaller –record -destinationFile <filename>
   ```

   where `–destinationFile` is the file location.
4. Run OUI in silent or suppressed mode.
5. If you completed a software-only installation, run Oracle Net Configuration Assistant (NetCA) and Database Configuration Assistant (DBCA) in silent or noninteractive mode, if required.

For more information, see your OS-specific *Oracle Database Installation Guide*.

Summary

In this lesson, you should have learned how to:

- Describe your role as a DBA, and explain tasks and tools
- Plan your installation, starting with the appropriate documentation
- Perform preinstallation tasks, such as checking system requirements
- Install software by using OUI

2-20

Practice Overview: Installing the Oracle Software

This practice covers installing the Oracle software by using Oracle Universal Installer.

Note: Completing this practice is critical for all the subsequent practice sessions.

Creating an Oracle Database

Objectives

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

- **Create a database with the Database Configuration Assistant (DBCA)**
- **Create a database design template with the DBCA**
- **Generate database creation scripts with the DBCA**

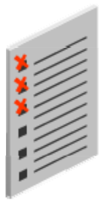
ORACLE

3-2

Planning the Database

As a DBA, you must plan:

- **The logical storage structure of the database and its physical implementation:**
  - **How many disk drives do you have for this?**
  - **How many data files will you need? (Plan for growth.)**
  - **How many tablespaces will you use?**
  - **Which type of information will be stored?**
  - **Are there any special storage requirements due to type or size?**
- **The overall database design**
- **A backup strategy for the database**



ORACLE

3-3

Planning the Database

It is important to plan how the logical storage structure of the database will affect system performance and various database management operations. For example, before creating any tablespaces for your database, you should know how many data files will make up the tablespace, what type of information will be stored in each tablespace, and on which disk drives the data files will be physically stored. When planning the overall logical storage of the database structure, take into account the effects that this structure will have when the database is actually created and running. You may have database objects that have special storage requirements due to type or size.

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

During the planning stage, develop a backup strategy for the database. You can alter the logical storage structure or design of the database to improve backup efficiency. Backup strategies are introduced in a later lesson.

These are the types of questions and considerations, which you will encounter as a DBA, and this course (in its entirety) is designed to help you answer them.

Databases: Examples

- **Data Warehouse:**
  - Research and marketing data
  - State or federal tax payments
  - Professional licensing (doctors, nurses, and so on)
- **Transaction Processing:**
  - Store checkout register system
  - Automatic teller machine (ATM) transactions
- **General Purpose:**
  - Retail billing system, for example, of a software house or a nursery

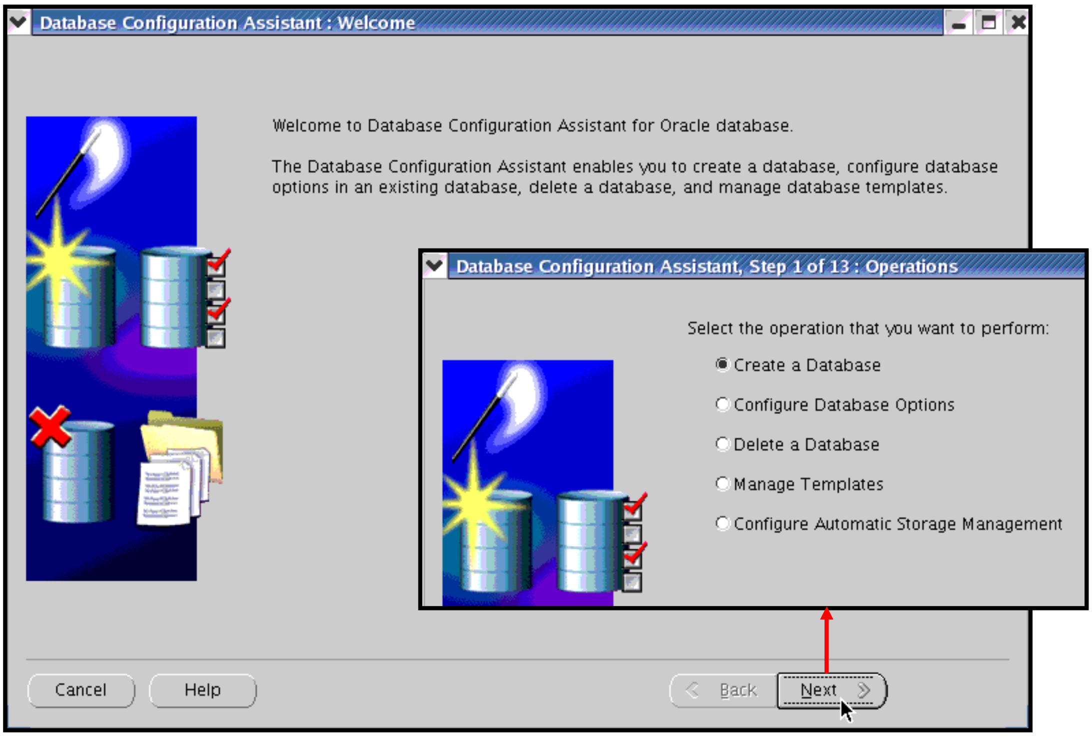
3-4

Databases: Examples

Different types of databases have their own specific instance and storage requirements. Your Oracle database software includes templates for the creation of these different types of databases. Characteristics of these examples are the following:

- **Data Warehouse:** Store data for long periods and retrieve them in read operations.
- **Transaction Processing:** Accommodate many, but usually small, transactions.
- **General Purpose:** Work with transactions and store them for a medium length of time.

Database Configuration Assistant (DBCA)



3-5

Database Configuration Assistant (DBCA)

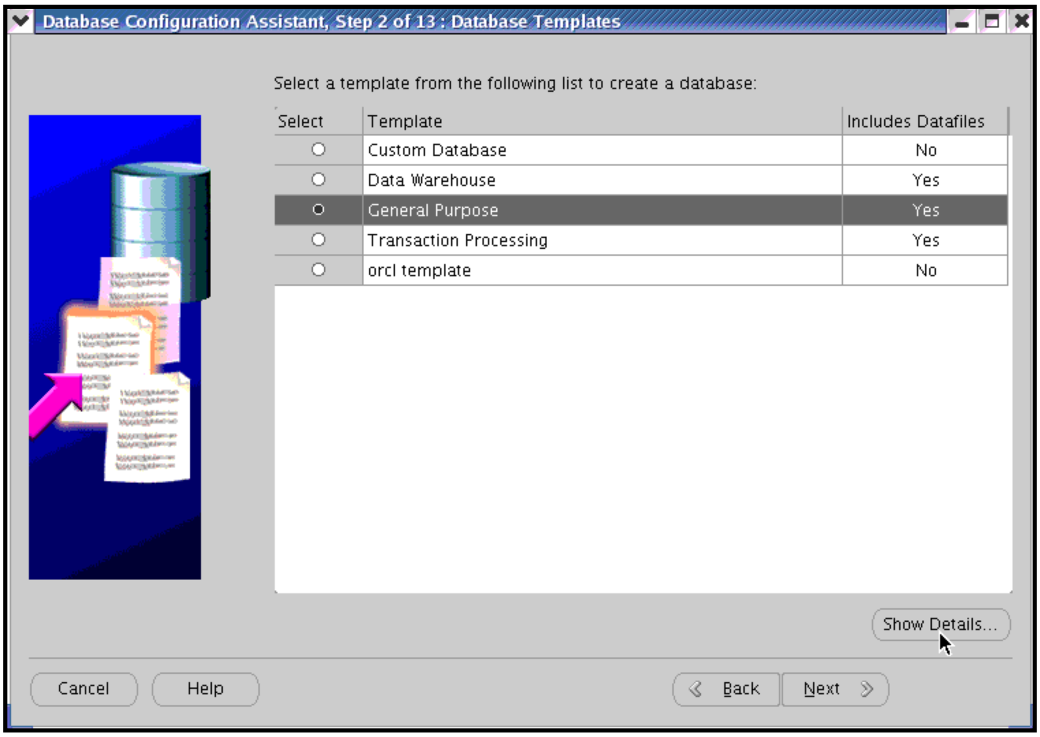
You can use the Database Configuration Assistant (DBCA) to create, change the configuration of, or delete a database. You can also create a database from a list of predefined templates or use an existing database as a sample to create a new database or template. This is sometimes referred to as “database cloning.”

You can invoke the DBCA by performing the following steps:

1. Log on to your computer as a member of the administrative group that is authorized to install the Oracle software.
2. If required, set environment variables.
3. Enter `dbca` to invoke the DBCA.
4. Click Next to continue.

DBCA offers you a choice of assisting with several operations, for example, creating a database.

Using the DBCA to Create a Database

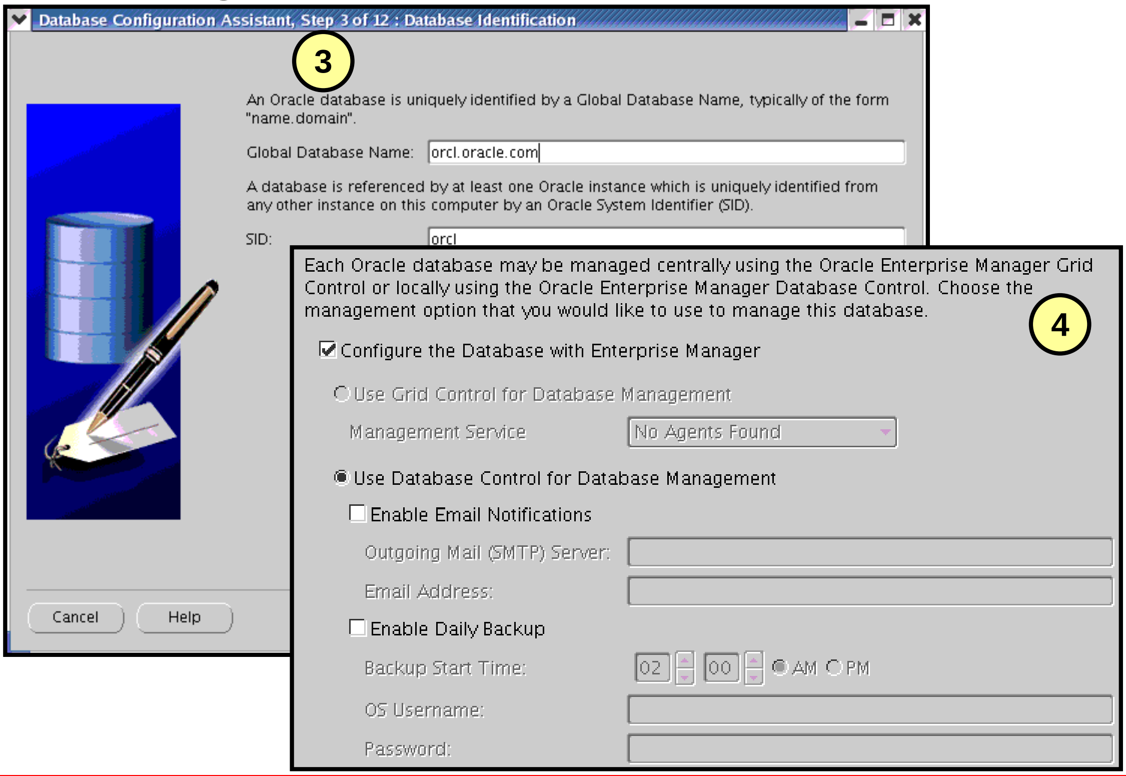


Using the DBCA to Create a Database

You can use the DBCA to create a database as follows:

1. Select Create a Database on the DBCA Operations page to invoke a wizard that enables you to configure and create a database.
   The wizard prompts you to provide configuration information as outlined in the steps that follow. On most pages, the wizard provides a default setting that you can accept.
2. Select the type of database template to be used in creating the database. There are templates for Data Warehouse, General Purpose, and Transaction Processing databases that copy a preconfigured database, including data files. These data files include control files, redo log files, and data files for various included tablespaces. Click Show Details to see the configuration for each type of database.
   For more complex environments, you may want to select the Custom Database option.

Using the DBCA to Create a Database

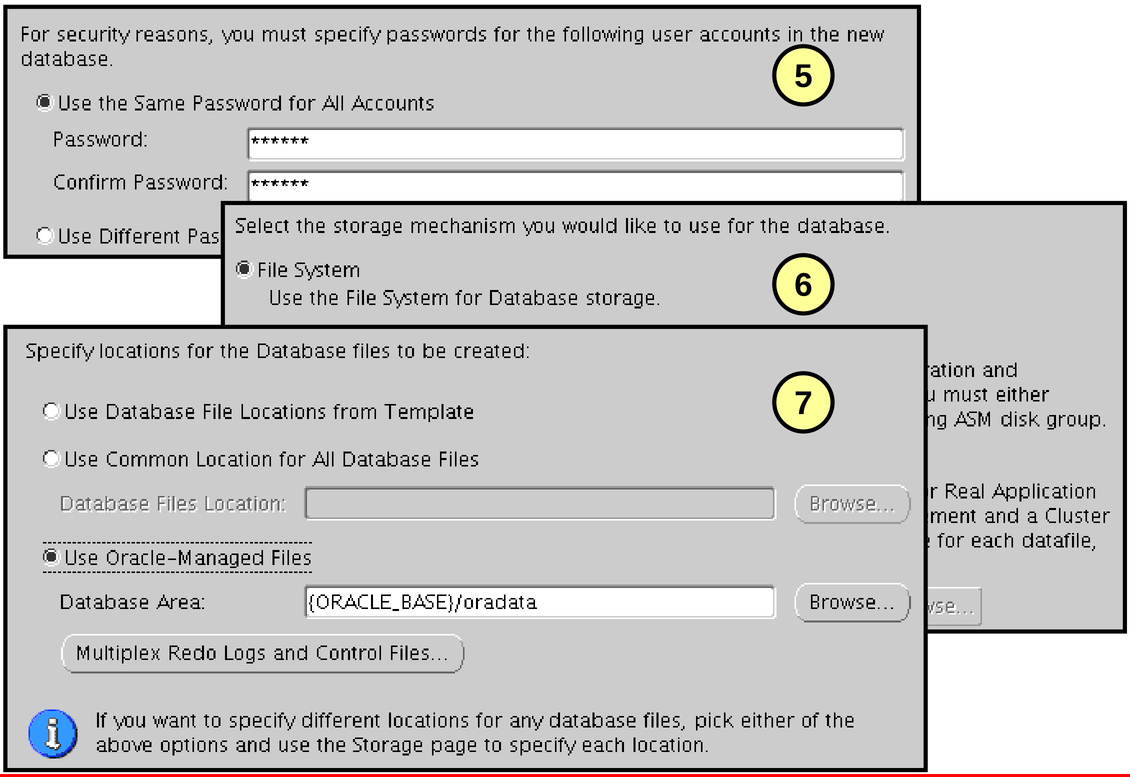


3-7

Using the DBCA to Create a Database (continued)

3. Database Identification: Enter the Global Database Name in the form `database_name.domain_name`, and the system identifier (SID). The SID defaults to the database name and uniquely identifies the instance associated with the database.
4. Management Options: Use this page to set up your database so that it can be managed with Oracle Enterprise Manager. Select the default: Configure the Database with Enterprise Manager.

Using the DBCA to Create a Database

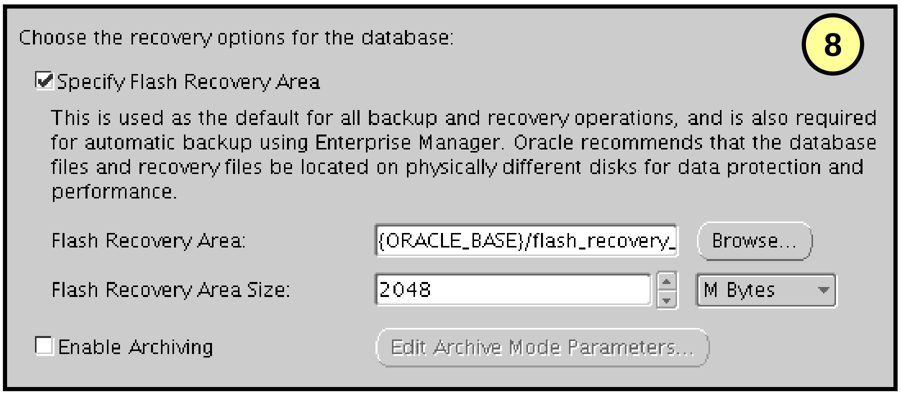


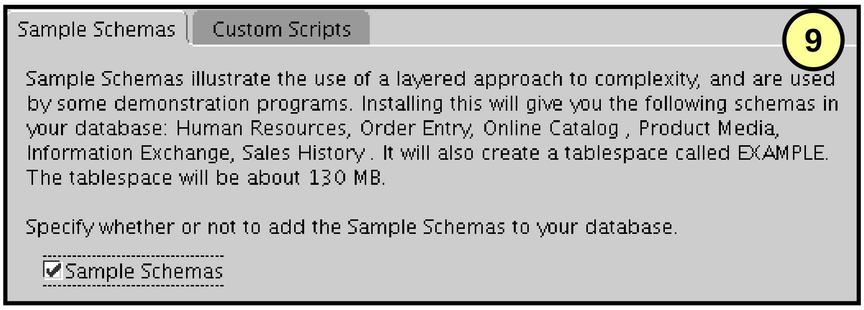
ORACLE

Using the DBCA to Create a Database (continued)

5. Database Credentials: Use this page to specify the passwords for the administrative accounts, such as `SYS` and `SYSTEM`. In class, use `oracle` as password for all administrative accounts.
6. Storage Options: Specify the type of storage mechanism (such as File System) that you would want your database to use.
7. Database File Locations: Choose according to your needs. Oracle Managed Files (OMF) eliminate the need for you to directly manage the operating system files comprising an Oracle database. You specify operations in terms of database objects rather than file names. For more details, see the lesson titled "Managing Database Storage Structures."

Using the DBCA to Create a Database



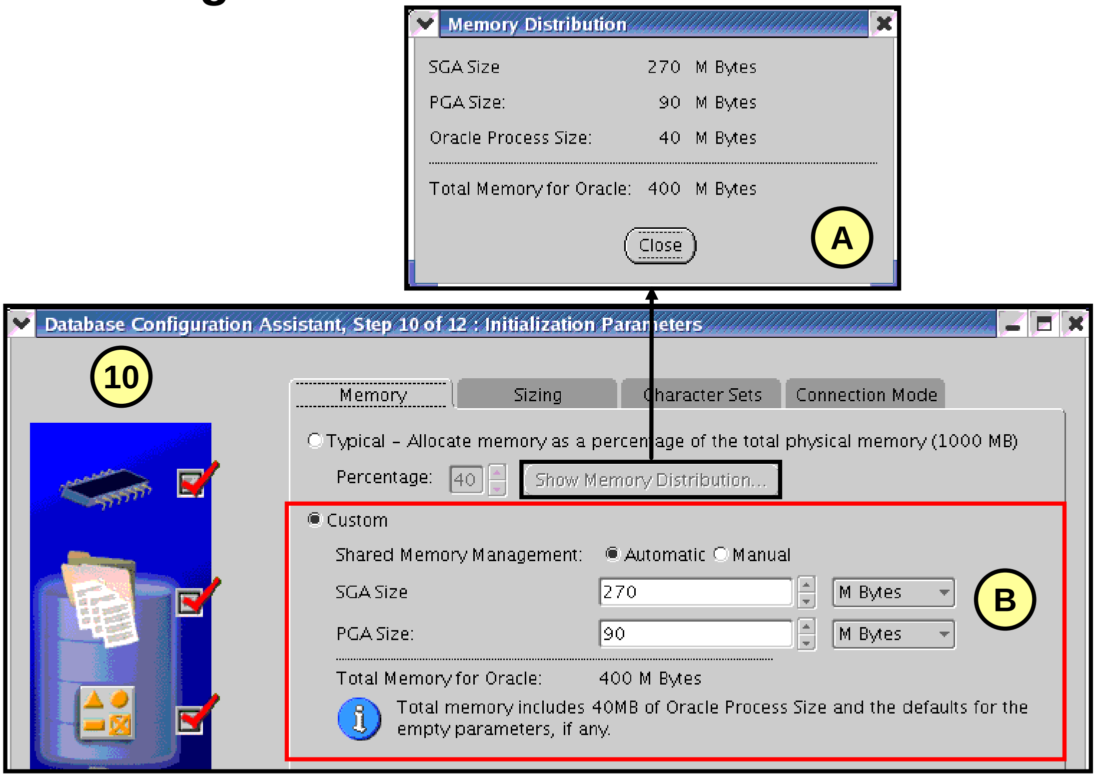


3-9

Using the DBCA to Create a Database (continued)

8. Recovery Configuration: If required, specify a flash recovery area and enable archiving.
9. Database Content: These pages provide options for selecting components, such as Sample Schemas, and for using custom scripts.

Using the DBCA to Create a Database



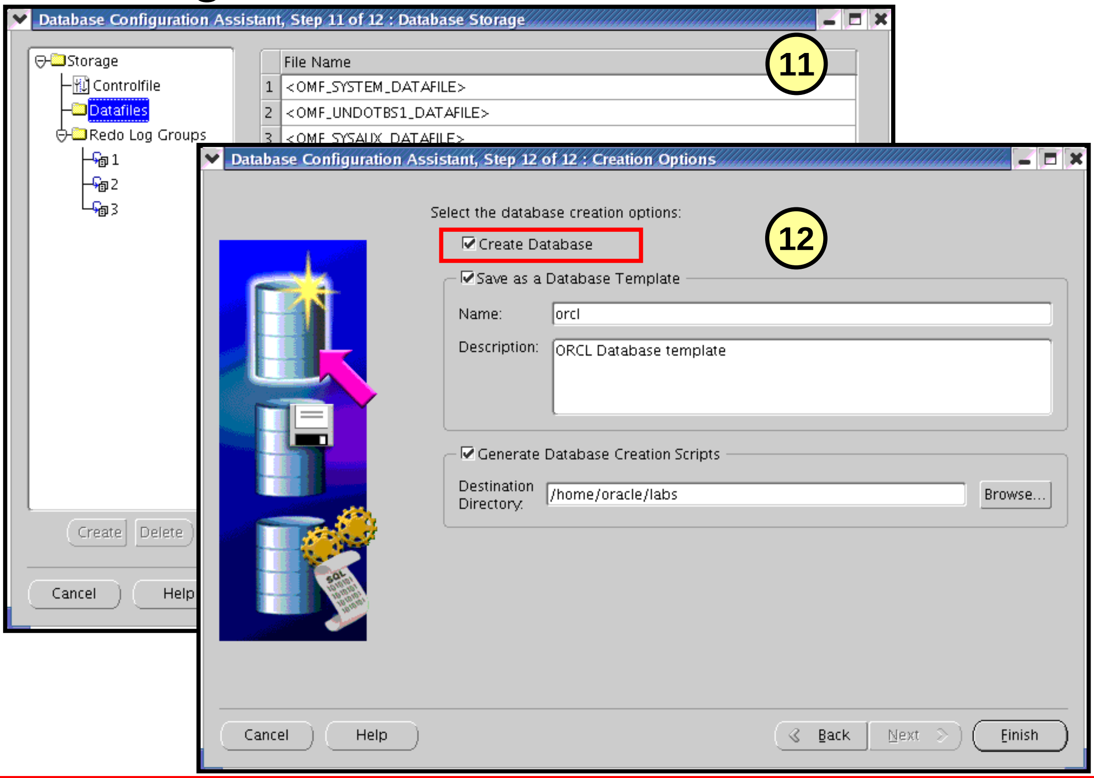
3-10

Using the DBCA to Create a Database (continued)

10. Initialization Parameters: The tabs on this page provide access to pages that enable you to change default initialization parameter settings:
    - Memory: Use this page to set the initialization parameters that control memory usage. Use either (A) Typical or (B) Custom memory allocation.
    - Sizing: To specify block size, enter the size in bytes or accept the default.
    - Character Sets: Use this page to specify the character sets for your database. **Best Practice Tip:** Oracle recommends (whenever possible) that you use Unicode for a database character set because it provides optimal flexibility for supporting Web technologies as well as many spoken languages.
    - Connection Mode: Select Dedicated or Shared Server Mode. For more details, see the lesson titled “Configuring the Oracle Network Environment.”

Note: Several initialization parameters are set for the lifetime of a database, such as the `DB_BLOCK_SIZE` and `CHARACTER_SET` parameters.

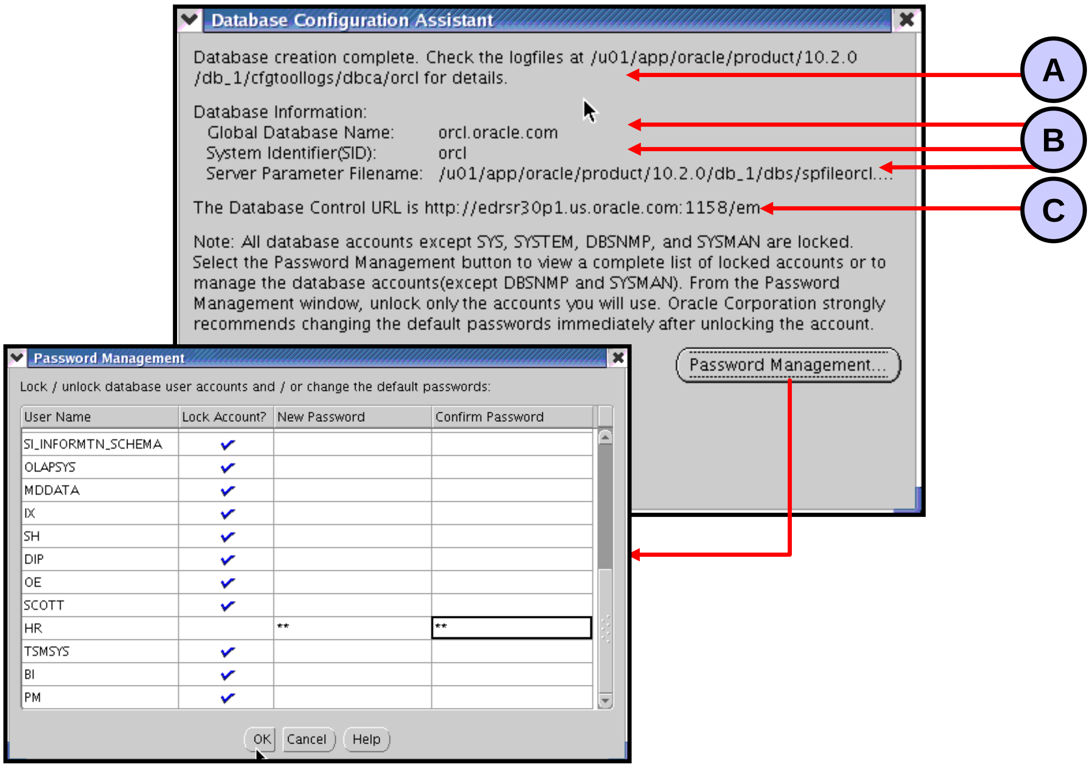
Using the DBCA to Create a Database



Using the DBCA to Create a Database (continued)

11. Database Storage: If you have selected one of the preconfigured templates for a database, then you cannot add or remove control files or data files.
    Note: You may want to save your database definition as an HTML file for easy reference.
12. Creation Options: You have the options of creating your database at this time, saving the database definition as a template, and generating scripts. If you choose all options, then the DBCA first saves the database template, then generates the scripts into your destination directory, and finally creates your database.

Password Management



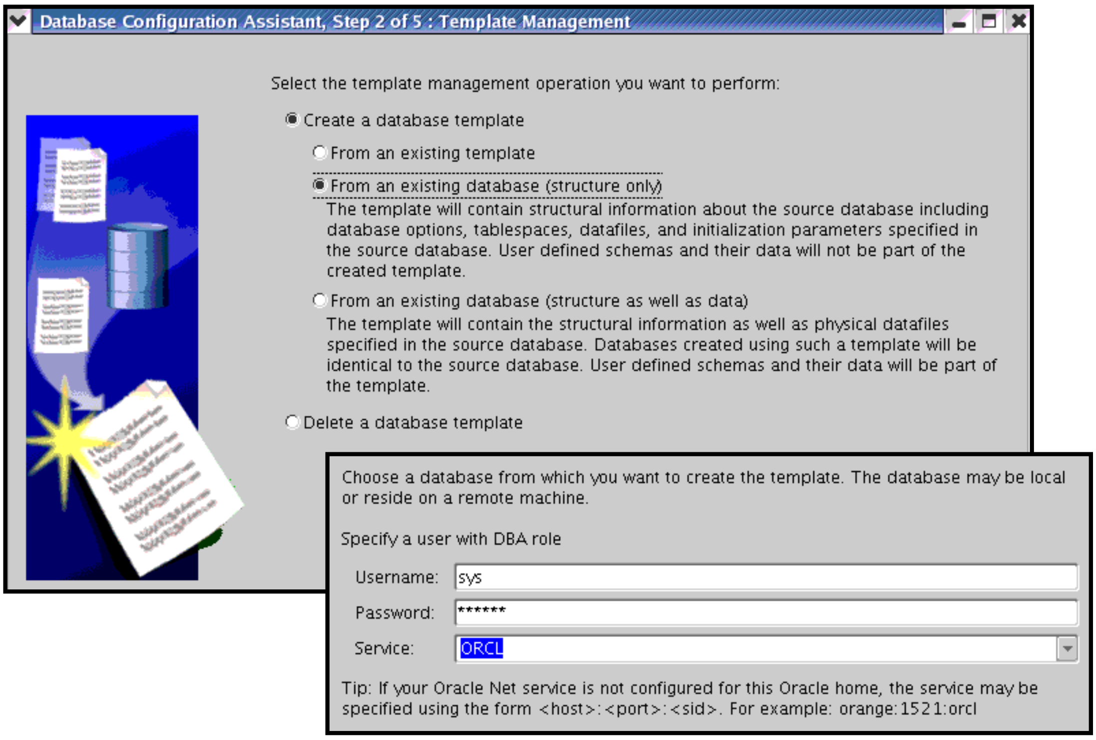
Password Management

After the DBCA finishes, note the following information for future reference:

- Location of installation log files (see A)
- Global database name (see B)
- System identifier (SID) (see B)
- Server parameter file name and location (see B)
- Enterprise Manager URL (see C)

Click Password Management to unlock database accounts that you plan to use. Provide a password when you unlock an account.

Creating a Database Design Template



3-13

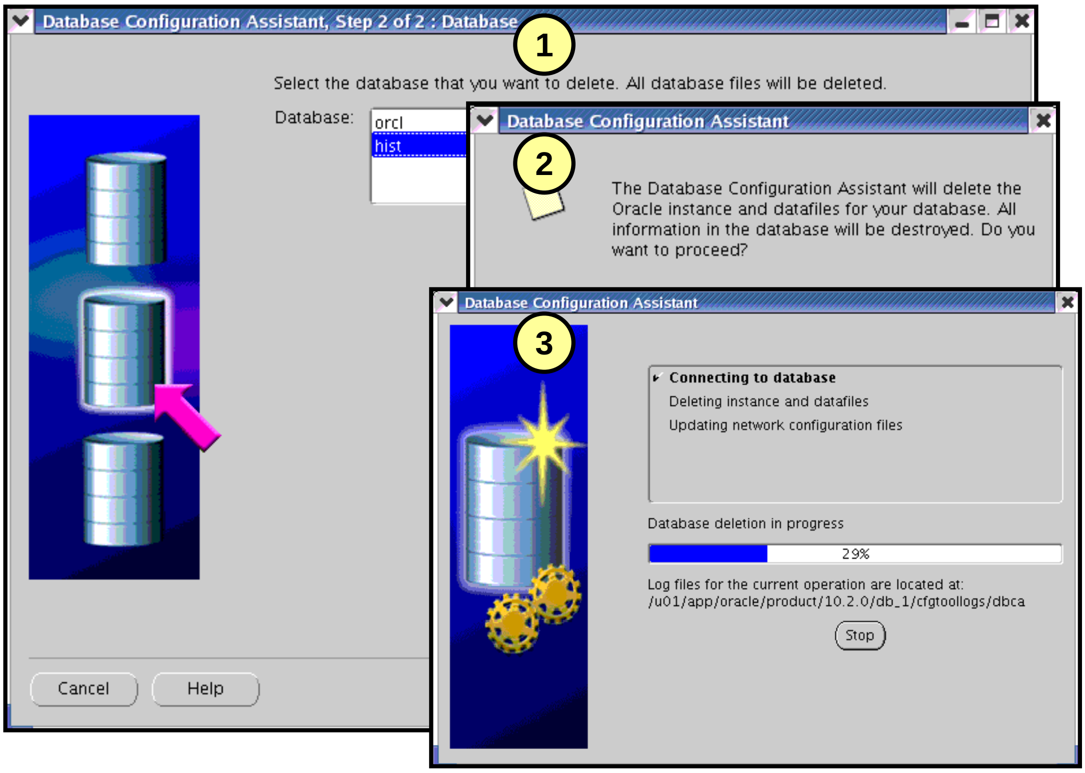
Creating a Database Design Template

A template is a predefined database definition that you use as a starting point for a new database. If you do not create a template as part of the database creation process, you can do it anytime by invoking the DBCA. 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)

The DBCA guides you through the steps to create a database design template.

Using the DBCA to Delete a Database



ORACLE

Using the DBCA to Delete a Database

To delete (or configure) a database in UNIX or Linux, you must set `ORACLE_SID` in the shell from which DBCA is launched. Start the DBCA by entering `dbca` in a terminal window, and click Next on the Welcome page. To delete the database, perform the following steps:

1. On the Operations page, select Delete a Database, and click Next.
2. Select the database that you want to delete (in class, `hist`), and click Finish.
3. Click Yes to confirm your deletion.

Using the DBCA to Delete a Database (continued)

Dropping a database involves removing its data files, redo log files, control files, and initialization parameter files. The `DROP DATABASE` statement deletes all control files and all other database files listed in the control file. To use the `DROP DATABASE` statement successfully, all the following conditions must apply:

- The database must be mounted and closed.
- The database must be mounted exclusively—not in shared mode.
- The database must be mounted as `RESTRICTED`.

An example of this statement is:

```
DROP DATABASE;
```

The `DROP DATABASE` statement has no effect on archived log files nor does it have any effect on copies or backups of the database. It is best to use Recovery Manager (RMAN) to delete such files. If the database is on raw disks, then the actual raw disk special files are not deleted.

Summary

In this lesson, you should have learned how to use the DBCA to:

- Create a database
- Create a database design template
- Generate database creation scripts

3-16

Practice Overview: Using the DBCA

This practice covers the following topics:

- Creating the `ORCL` database by using the DBCA
- Unlocking the `HR` schema

Note: Completing the database creation and unlocking the HR schema is critical for all following practice sessions.

Optionally:

- Creating the `ORCL` database design template by using the DBCA
- Creating database creation scripts with the DBCA

4

Managing the Oracle Instance

Objectives

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

- Start and stop the Oracle database and components
- Use Enterprise Manager (EM)
- Access a database with SQL*Plus and *i*SQL*Plus
- Modify database initialization parameters
- Describe the stages of database startup
- Describe the database shutdown options
- View the alert log
- Access dynamic performance views

ORACLE

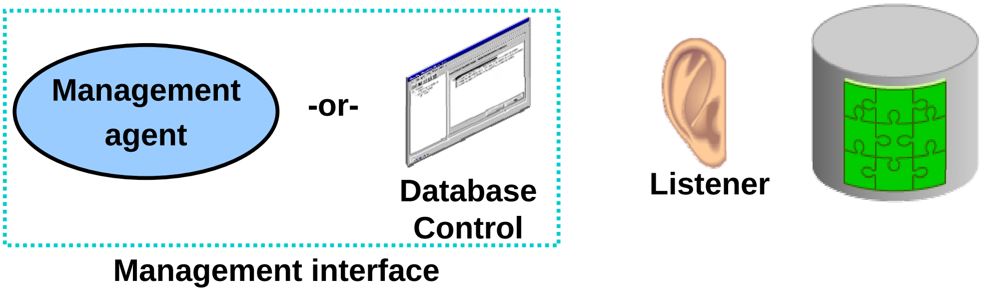
4-2

Management Framework

> Components
SQL*Plus
Init Params
DB Startup
DB Shutdown
Alert Log
Perf Views

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 management framework:

- The database instance that is being managed
- A listener that allows connections to the database
- The management interface. This may be either a management agent running on the database server (which connects it to Oracle Enterprise Manager Grid Control) or the stand-alone Oracle Enterprise Manager Database Control. This is also referred to as Database Console.

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

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

Starting and Stopping Database Control

```
$ emctl start dbconsole
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release 10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights reserved.
http://edrsr9p1.us.oracle.com:1158/em/console/aboutApplication
Starting Oracle Enterprise Manager 10g Database Control
............. started.
------------------------------------------------------------------
Logs are generated in directory
/u01/app/oracle/product/10.2.0/db_1/edrsr9p1.us.oracle.com_orcl/sy
sman/log
```

```
$ emctl stop dbconsole
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release 10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights reserved.
http://edrsr9p1.us.oracle.com:1158/em/console/aboutApplication
Stopping Oracle Enterprise Manager 10g Database Control ...
 ...  Stopped.
```

ORACLE

Starting and Stopping Database Control

Oracle provides a stand-alone management console called Database Control for databases that are not connected to the Grid Control framework. Each database that is managed with Database Control has a separate Database Control installation, and from any one Database Control, you can manage only one database. Before using Database Control, ensure that a dbconsole process is started.

To start the `dbconsole` process, use the following command:

```
emctl start dbconsole
```

To stop the `dbconsole` process, use the following command:

```
emctl stop dbconsole
```

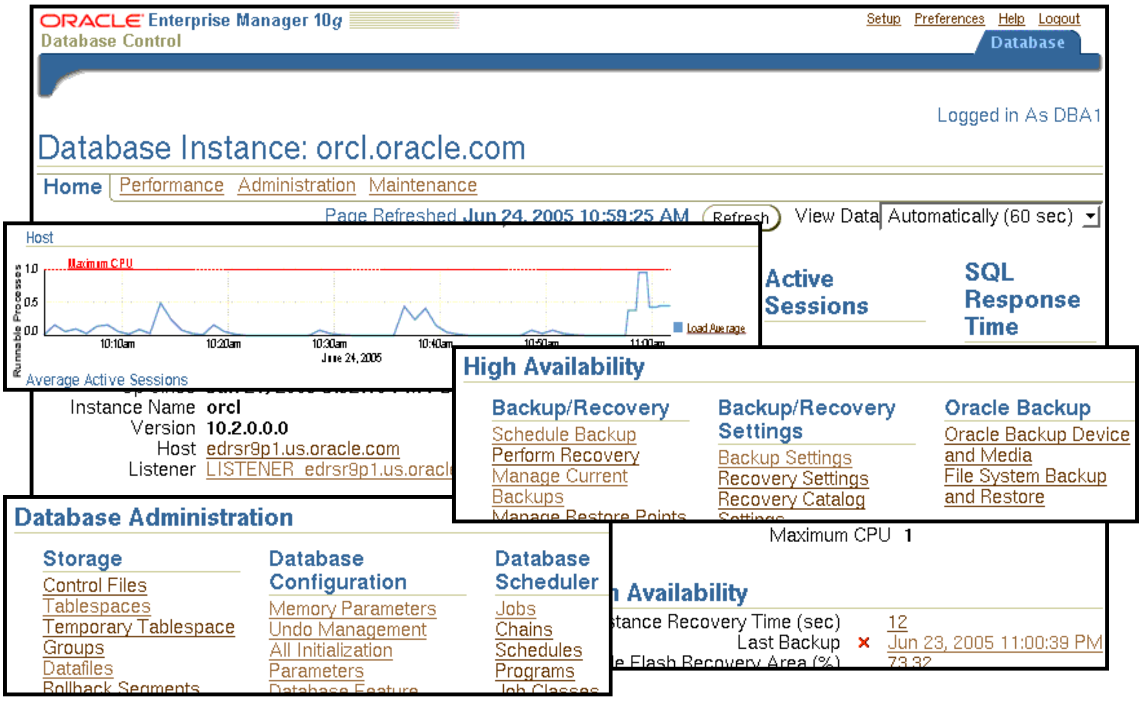
To view the status of the `dbconsole` process, use the following command:

```
emctl status dbconsole
```

Note: You may need to navigate to your `$ORACLE_HOME/bin` directory if this directory is not in your operating system (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.

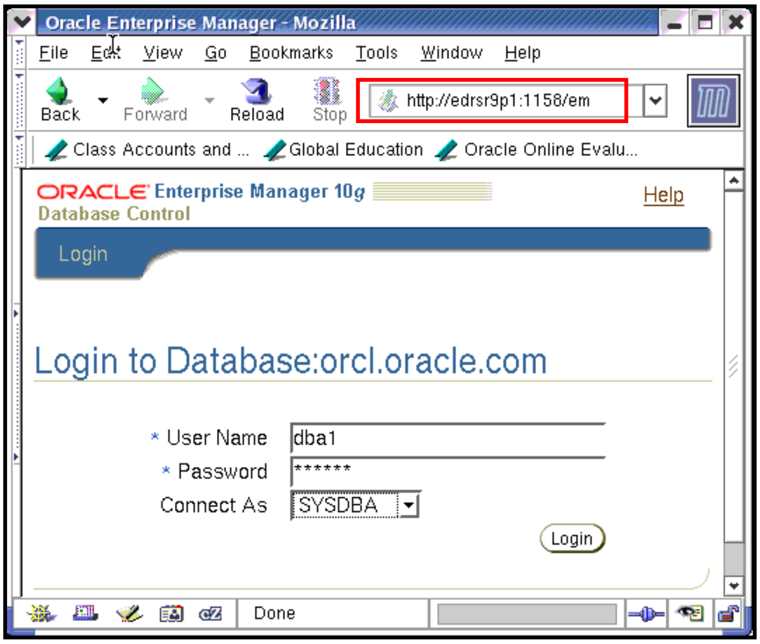
Oracle Enterprise Manager



Oracle Enterprise Manager

When you install an Oracle database, Oracle Universal Installer also installs Oracle Enterprise Manager (Enterprise Manager). Its Web-based Database Control serves as the primary tool for managing your Oracle database. You can access online help from any of the pages to assist you with the task at hand. You can drill down into links in most situations, where there is more specific information to be had about the contents of a page. Although you may sometimes want to write and execute commands that you compose yourself, Enterprise Manager provides a graphical interface for doing almost any task that you would have to do as a database administrator (DBA). Viewing alert summaries and performance graphs, creating and modifying objects, and performing backup and recovery are some of the things that you can do with Enterprise Manager.

Accessing Oracle Enterprise Manager



4-6

Accessing Oracle Enterprise Manager

Open your Web browser, and enter the following URL:

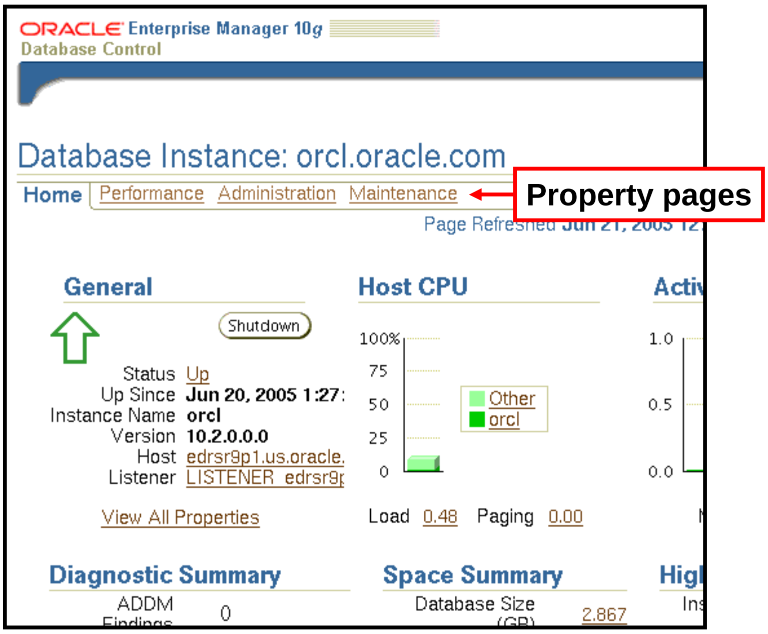
http://*host name*:*port number*/em

If the database is:

- **Up:** Enterprise Manager displays the Database Control Login page. Log in to the database by using a username that is authorized to access Database Control. Initially, this is SYS, SYSMAN, or SYSTEM. Use the password that you specified for the account during the database installation. In the Connect As option, select either SYSDBA or SYSOPER to log in to the database with special database administration privileges.
- **Down:** 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.

Note: If you have trouble starting Enterprise Manager, ensure that a listener is started.

Database Home Page



Database Home Page

The Database Home page displays the current state of the database by displaying a series of metrics that portray the overall health of the database. With the property pages, which are also referred to as tabs, you can access the Performance, Administration, and Maintenance pages for managing your database.

You can view the following performance and status information about your database instance on the Database Home page:

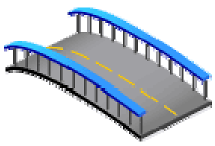
- Instance name, database version, Oracle home location, media-recovery options, and other pertinent instance data
- Current instance availability
- Outstanding alerts
- Session-related and SQL-related performance information
- Key space usage metrics
- Drill-down links (for example, `LISTENER_`<*`host_name`*>) to provide increasing levels of detail

Using SQL*Plus and *i*SQL*Plus to Access Your Database

- Components
- > SQL*Plus
- Init Params
- DB Startup
- DB Shutdown
- Alert Log
- Perf Views

SQL*Plus and *i*SQL*Plus provide additional interfaces to your database to:

- **Perform database management operations**
- **Execute SQL commands to query, insert, update, and delete data in your database**



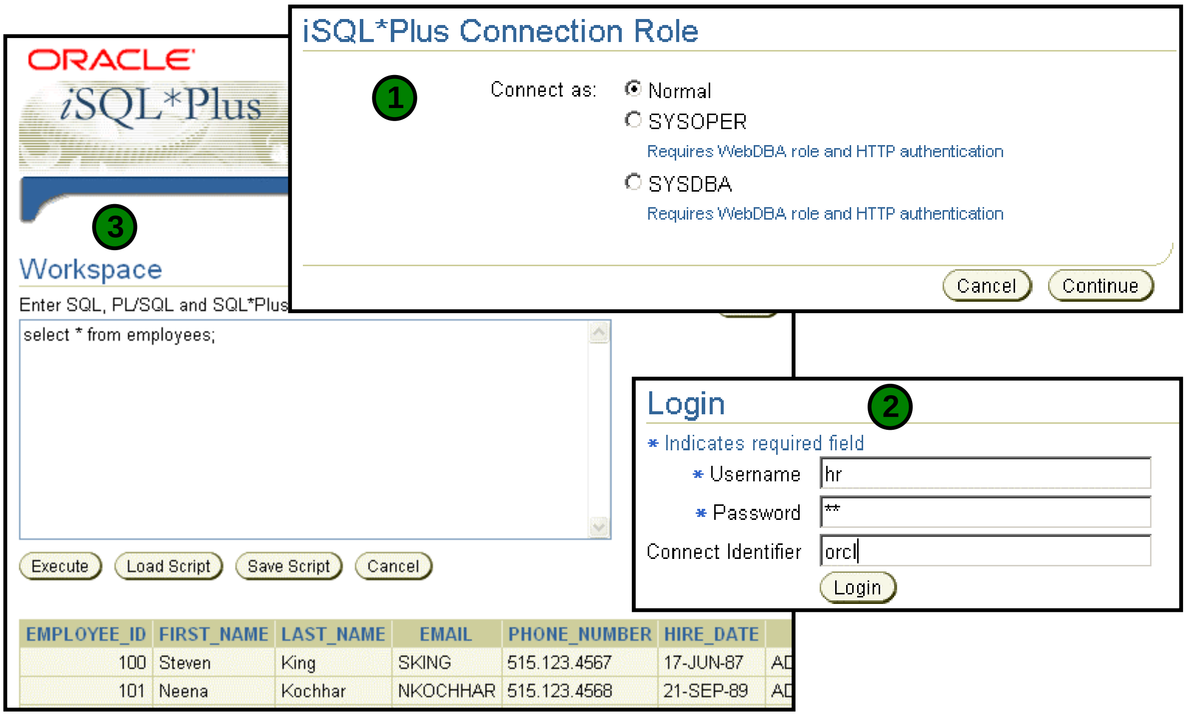
ORACLE

4-8

Using SQL*Plus and *i*SQL*Plus to Access Your Database

In addition to Enterprise Manager, you can use other Oracle tools, such as SQL*Plus and *i*SQL*Plus, to issue SQL statements. These tools enable you to perform many of the database management operations as well as to select, insert, update, or delete data in the database.

Using *i*SQL*Plus



4-9

Using *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, connect to it by entering the following URL in a browser:

http://*host name*:*port*/isqlplus

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

Setting Up *i*SQL*Plus for SYSDBA and SYSOPER Access

For a user to login to *i*SQL*Plus as SYSDBA or SYSOPER you must set up the user in the OC4J user manager by performing the following steps:

1. **Create a user**
2. **Grant the webDba role to the user**

```
$ cd $ORACLE_HOME/oc4j/j2ee/isqlplus/\
> application-deployments/isqlplus
$JAVA_HOME/bin/java \
> -Djava.security.properties=\
> $ORACLE_HOME/oc4j/j2ee/home/config/jazn.security.props \
> -jar $ORACLE_HOME/oc4j/j2ee/home/jazn.jar \
> -user "iSQL*Plus DBA/admin" -password welcome -shell
JAZN> adduser "iSQL*Plus DBA" username password
JAZN> grantrole webDba "iSQL*Plus DBA" username
```

ORACLE

4-10

Setting Up *i*SQL*Plus for SYSDBA and SYSOPER Access

When the *i*SQL*Plus Connection Role page appears, notice that the SYSOPER and SYSDBA roles require special setup and authentication for security reasons. To do this, you must set up a user in the Oracle Application Server Containers for J2EE (OC4J) user manager and grant access to the webDba role for the user. Do this by performing the following steps. Note that the JAVA_HOME OS environment variable must be set to $ORACLE_HOME/jdk.

1. Change to the correct directory:
```
cd $ORACLE_HOME/oc4j/j2ee/isqlplus/\
application-deployments/isqlplus
```
2. Run the JAZN shell:
```
$JAVA_HOME/bin/java \
-Djava.security.properties=\
$ORACLE_HOME\
/oc4j/j2ee/home/config/jazn.security.props \
-jar $ORACLE_HOME/oc4j/j2ee/home/jazn.jar \
-user "iSQL*Plus DBA/admin" \
-password welcome -shell
```

Setting Up *i*SQL*Plus for `SYSDBA` and `SYSOPER` Access (continued)

3. Create a user, choosing a username and password:
```
JAZN> adduser "iSQL*Plus DBA" username password
```
4. Grant the `webDba` role to the user:
```
JAZN> grantrole webDba "iSQL*Plus DBA" username
```
5. Exit the `JAZN` shell:
```
JAZN> exit
```

Using SQL*Plus

SQL*Plus is:
- **A command-line tool**
- **Used interactively or in batch mode**

```
$ sqlplus hr/hr

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jul 25 12:37:21 2005
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> select last_name from employees;

LAST_NAME
-------------------------
Abel
Ande
Atkinson
```

ORACLE

4-12

Using 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

To start SQL*Plus, perform the following steps:
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.
   SQL*Plus starts and connects to the default database.

Calling SQL*Plus from a Shell Script

```
$ ./batch_sqlplus.sh

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jul 25 12:47:44 2005
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL>
  COUNT(*)
----------
       107
SQL>
107 rows updated.
SQL>
Commit complete.
SQL> Disconnected from Oracle Dat
10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
[oracle@EDRSR9P1 oracle]$
```

Output

```
# Name of this file: batch_sqlplus.sh
# Count employees and give raise.
sqlplus hr/hr <<EOF
select count(*) from employees;
update employees set salary =
salary*1.10;
commit;
quit
EOF
exit
```

4-13

Calling SQL*Plus from a Shell Script

You can call SQL*Plus from a shell script or `BAT` file by invoking `sqlplus` and using the operating system scripting syntax for passing parameters.

In this example, the `SELECT`, `UPATE` and `COMMIT` statements are executed, before SQL*Plus returns control to the operating system.

Calling a SQL Script from SQL*Plus

script.sql

```
select * from departments where location_id = 1400;
quit
```

Output

```
$ sqlplus hr/hr @script.sql

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jul 25 12:57:02 2005
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
           60 IT                                    103        1400

Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

ORACLE

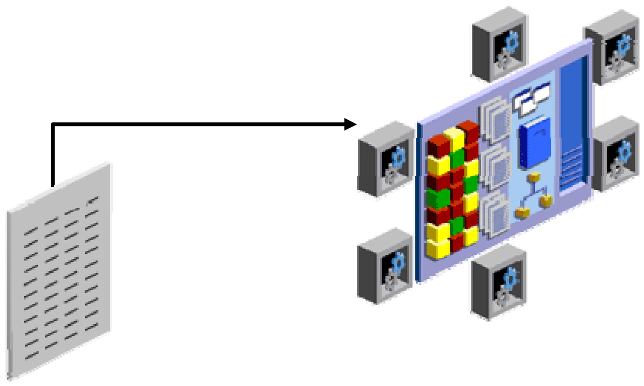
Calling a SQL Script from SQL*Plus

You can call an existing SQL script file from within SQL*Plus. This can be done at the command line when first invoking SQL*Plus, as shown in the slide. It can also be done from inside a SQL*Plus session, simply by using the "@" operator. For example, this runs the script from within an already established SQL*Plus session:

```
SQL> @script.sql
```

Initialization Parameter Files

Components
SQL*Plus
> Init Params
DB Startup
DB Shutdown
Alert Log
Perf Views



ORACLE

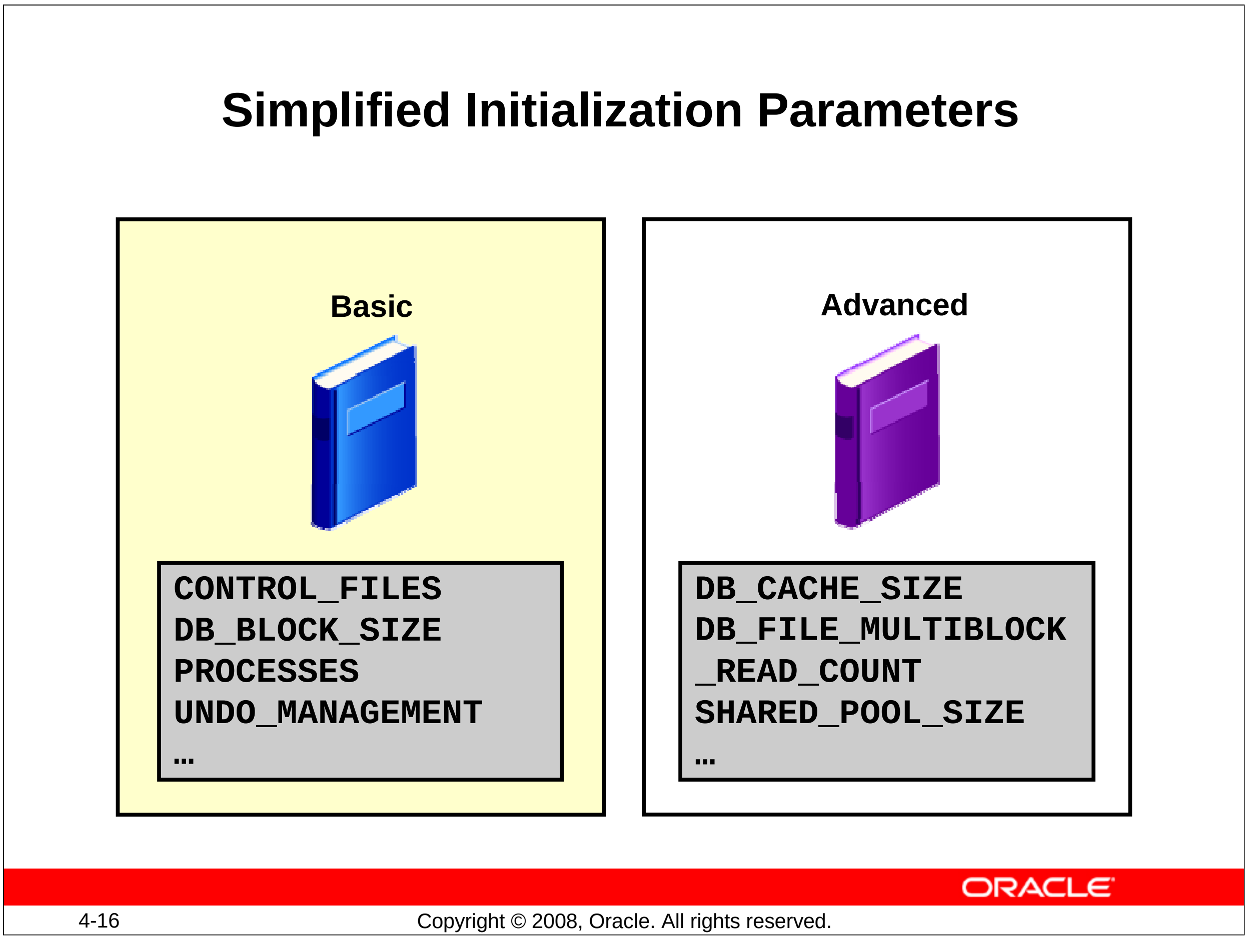
4-15

Initialization Parameter Files

When you start the instance, an initialization parameter file is read. There are two types of parameter files:

- **Server parameter file:** This is the preferred type of initialization parameter file. It is a binary file that can be written to and read by the database server and *must not be edited manually*. It resides in the server that the Oracle database is executing on, and is persistent across shutdown and startup. This is often referred to as a server parameter file (SPFILE). The default name of this file, which is automatically sought at startup, is `spfile<SID>.ora`.
- **Text initialization parameter file:** This type of initialization parameter file can be read by the database server, but it is not written to by the server. The initialization parameter settings must be set and changed manually by using a text editor so that they are persistent across shutdown and startup. The default name of this file, which is automatically sought at startup if an SPFILE is not found, is `init<SID>.ora`.

It is recommended that you create an SPFILE as a dynamic means of maintaining initialization parameters. By using an SPFILE, you can store and manage your initialization parameters persistently in a server-side disk file.



Simplified Initialization Parameters

Initialization parameters are divided into two groups: basic and advanced.

In the majority of cases, it is necessary to set and tune only the 32 basic parameters to get reasonable performance from the database. In rare situations, modification of the advanced parameters may be needed to achieve optimal performance.

A basic parameter is defined as one that you are likely to set to keep your database running with good performance. All other parameters are considered to be advanced.

The examples of basic parameters include "destinations" or directory names for specific types of files: `AUDIT_FILE_DEST`, `BACKGROUND_DUMP_DEST`, `CORE_DUMP_DEST`, `DB_CREATE_FILE_DEST`, `DB_CREATE_ONLINE_LOG_DEST_`*n*, `DB_RECOVERY_FILE_DEST`, and `USER_DUMP_DEST`.

Initialization Parameters: Examples

The `CONTROL_FILES` parameter specifies one or more control file names. Oracle strongly recommends that you multiplex and mirror control files. The range of values for this parameter is from 1 to 8 file names (with path names). The default range is OS dependent.

Simplified Initialization Parameters (continued)

Initialization Parameters: Examples (continued)

The DB_BLOCK_SIZE parameter specifies the size (in bytes) of an Oracle database block. This value is set at database creation and cannot be subsequently changed. Range of values: 2048–32768 (OS dependent). Default value: 8K (OS dependent).

The DB_CACHE_SIZE parameter specifies the size of the standard block buffer cache. Range of values: At least 16 MB. Default value: 48 MB.

The DB_FILE_MULTIBLOCK_READ_COUNT parameter specifies the maximum number of blocks read during an input/output (I/O) operation involving a full sequential scan. Range of values: Operating system dependent. Default value: OS dependent.

The DB_FILES parameter specifies the maximum number of database files that can be opened for this database. Range of values: MAXDATAFILES – OS dependent. Default value: OS dependent (200 on Solaris).

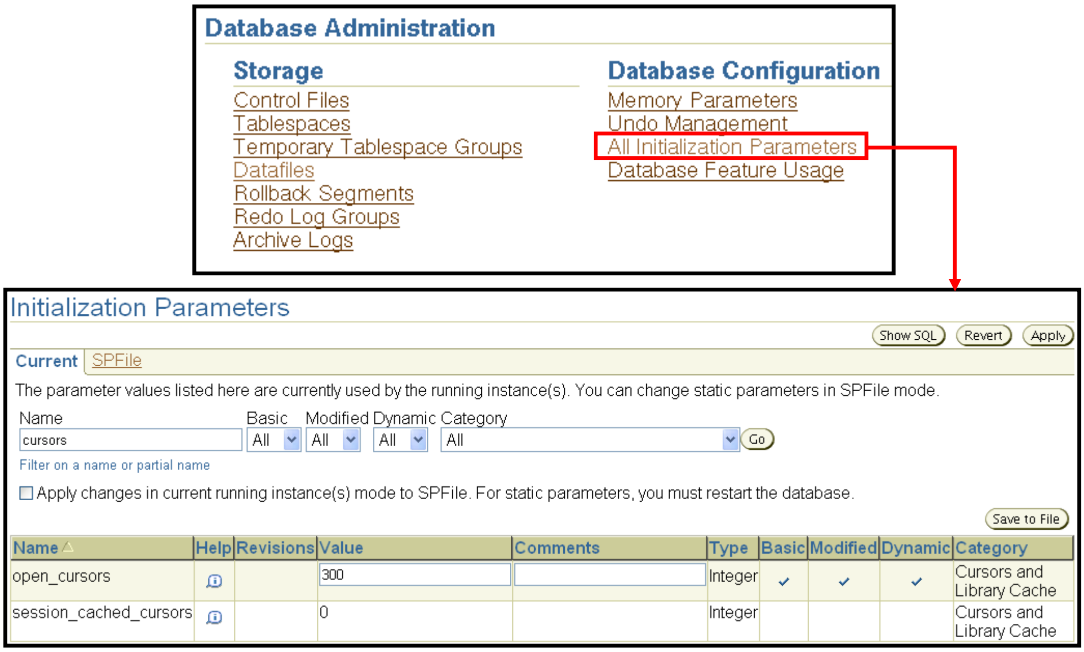
The PGA_AGGREGATE_TARGET parameter specifies the amount of Program Global Area (PGA) memory allocated to all server processes attached to the instance. Set this parameter to a positive value before enabling the automatic setting of working areas. This memory does not reside in the System Global Area (SGA). The database uses this parameter as a target amount of PGA memory to use. When setting this parameter, subtract the SGA from the total memory on the system available to the Oracle instance. The remaining memory can be assigned to PGA_AGGREGATE_MEMORY. Range of values: Integers plus letter K, M, or G to specify this limit in kilobytes, megabytes, or gigabytes. Minimum value is 10 MB and maximum value is 4096 GB. Default: 10 MB or 20% of the size of the SGA, whichever is greater.

The PROCESSES parameter specifies the maximum number of OS user processes that can simultaneously connect to an Oracle server. This value should allow for all background processes. Range of values: 6 to an OS-dependent value. Default value: OS dependent.

The SHARED_POOL_SIZE parameter specifies the size of the shared pool in bytes. The shared pool contains objects such as shared cursors, stored procedures, control structures, and parallel execution message buffers. Larger values can improve performance in multiuser systems. Range of values: The size of a granule–OS dependent. Default value: If 64 bit, then 64 MB, or else 16 MB.

The UNDO_MANAGEMENT parameter specifies which undo space management mode the system should use. When set to AUTO, the instance is started in System Managed Undo (SMU) mode. Otherwise, it is started in Rollback Undo (RBU) mode. In RBU mode, undo space is allocated externally as rollback segments. In SMU mode, undo space is allocated externally as undo tablespaces. Range of values: AUTO or MANUAL. Default value: If the UNDO_MANAGEMENT parameter is omitted when the first instance is started, the default value of MANUAL is used and the instance is started in RBU mode. If it is not the first instance, the instance is started in the same undo mode as all other existing instances.

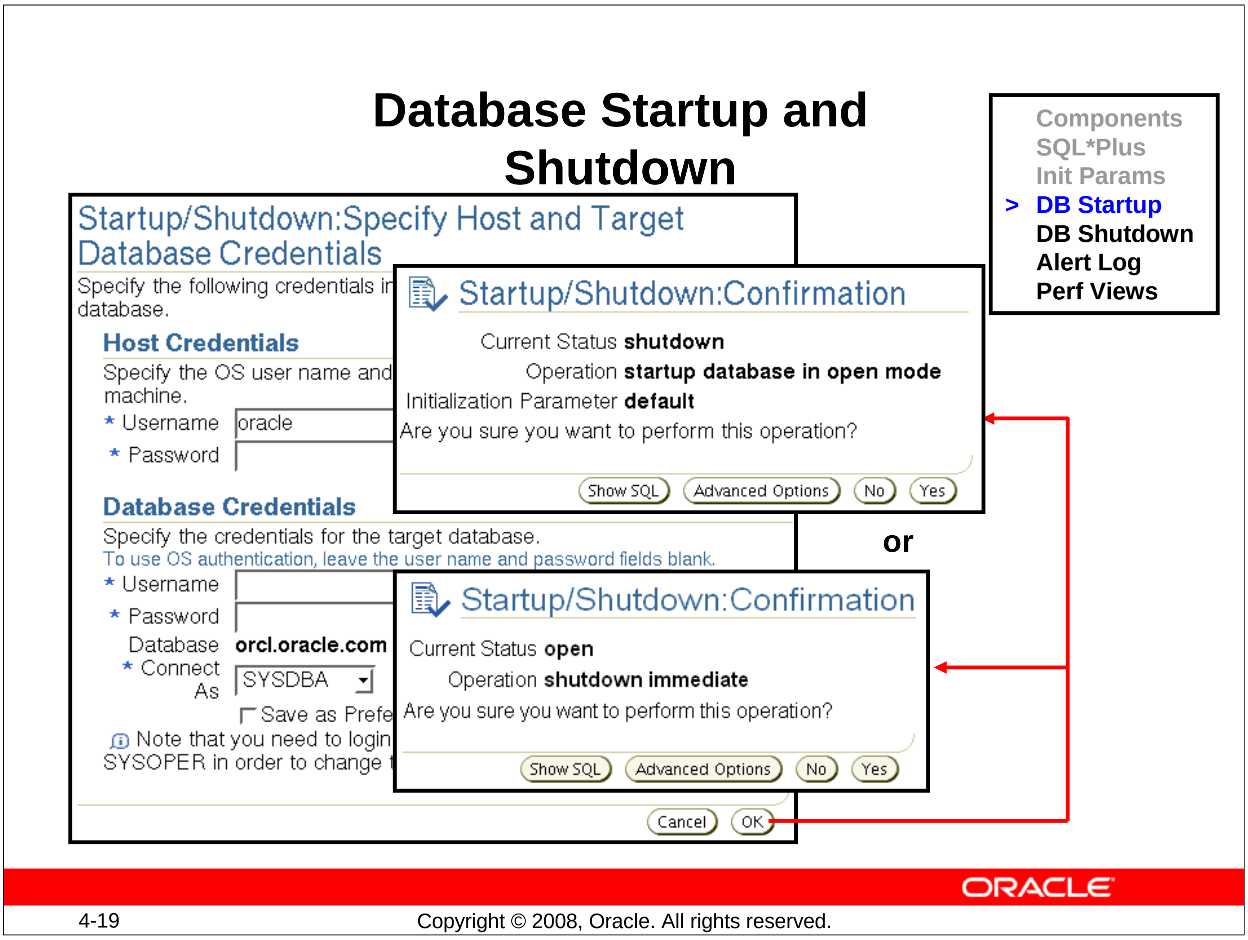
Viewing and Modifying Initialization Parameters



ORACLE

Viewing and Modifying Initialization Parameters

You can use Enterprise Manager to view and modify initialization parameters by clicking All Initialization Parameters in the Database Configuration region of the Database Administration tabbed page.

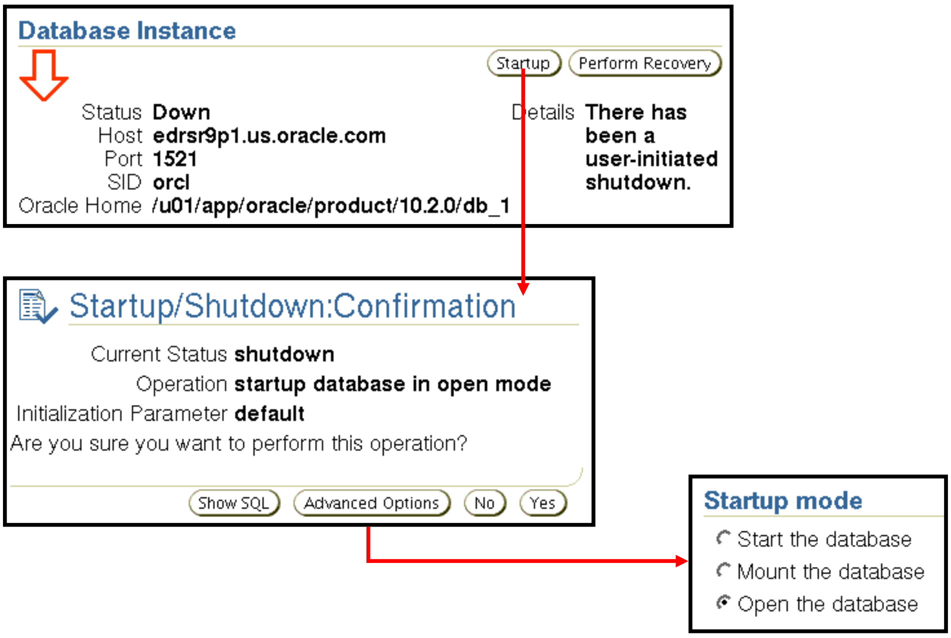


Database Startup and Shutdown

When you click either startup or shutdown, you are prompted for credentials that are used for both logging on to the host (the computer on which the database resides) and logging in to the database itself. Enter the credentials.

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

Starting Up an Oracle Database Instance

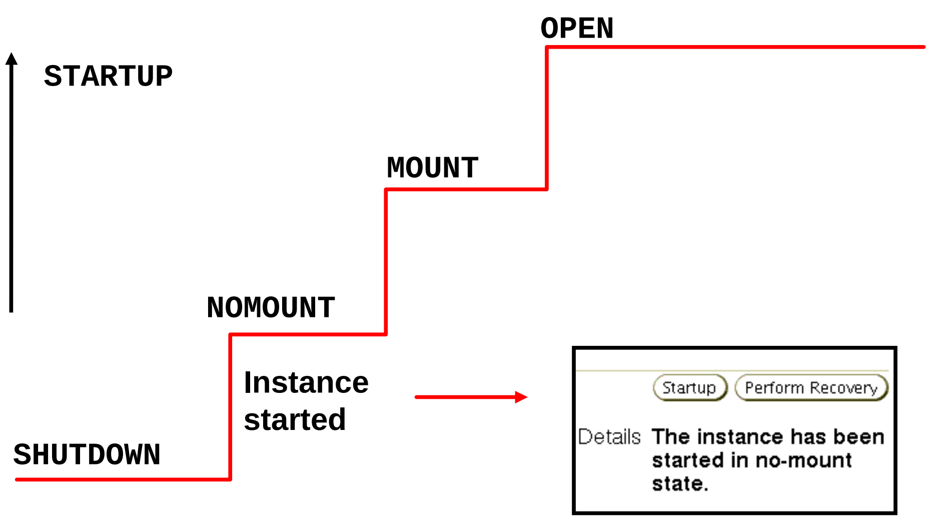


4-20

Starting Up an Oracle Database Instance

If the database is currently not started when you go to the Enterprise Manager Database Control page, click Startup to perform the startup. Enter the host credentials and, optionally, choose the startup mode.

Starting Up an Oracle Database Instance: NOMOUNT



4-21

Starting Up an Oracle Database Instance: NOMOUNT

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

An instance is typically started only in `NOMOUNT` mode during database creation, during re-creation of control files, or during certain backup and recovery scenarios.

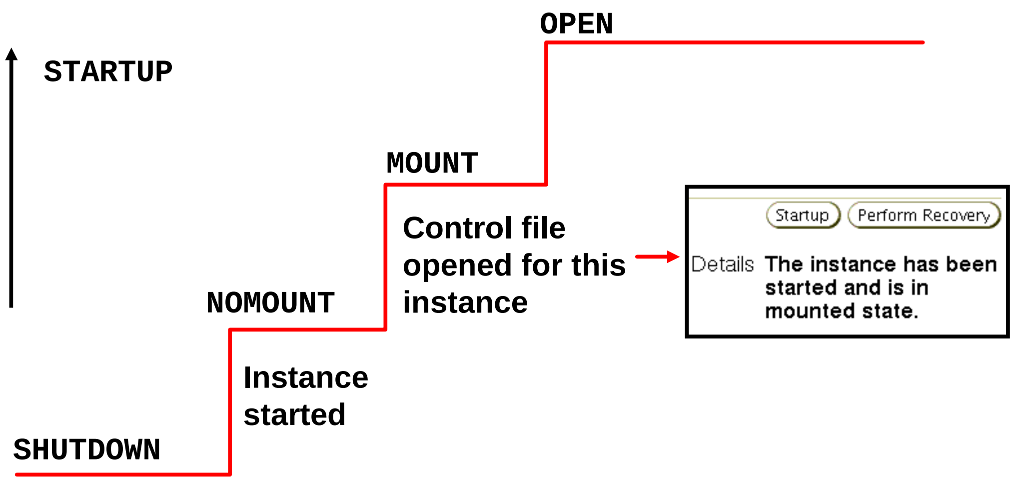
Starting an instance includes the following tasks:

- Searching `<oracle_home>/dbs` for a file of a particular name in this order:
  - `spfile<SID>.ora`
  - If not found, `spfile.ora`
  - If not found, `init<SID>.ora`

    This is the file that contains initialization parameters for the instance. Specifying the `PFILE` parameter with `STARTUP` overrides the default behavior.
- Allocating the SGA
- Starting the background processes
- Opening the `alert<SID>.log` file and the trace files

Note: `SID` is the system ID, which identifies the instance (for example, `ORCL`).

Starting Up an Oracle Database Instance: MOUNT



Starting Up an Oracle Database Instance: MOUNT

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 statuses 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.

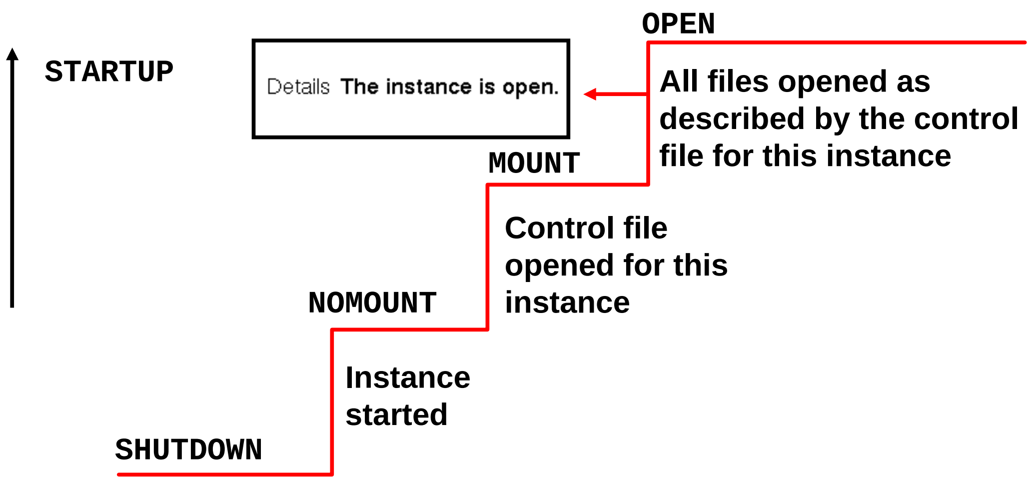
To perform specific maintenance operations, start an instance and mount a database, but do not open the database.

For example, the database must be mounted but must not be opened 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

Note: A database may be left in `MOUNT` mode even though an `OPEN` request has been made. This may be because the database needs to be recovered in some way.

Starting Up an Oracle Database Instance: OPEN



4-23

Starting Up an Oracle Database Instance: `OPEN`

A normal database operation means that an instance is started and the database is mounted and opened. With a 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

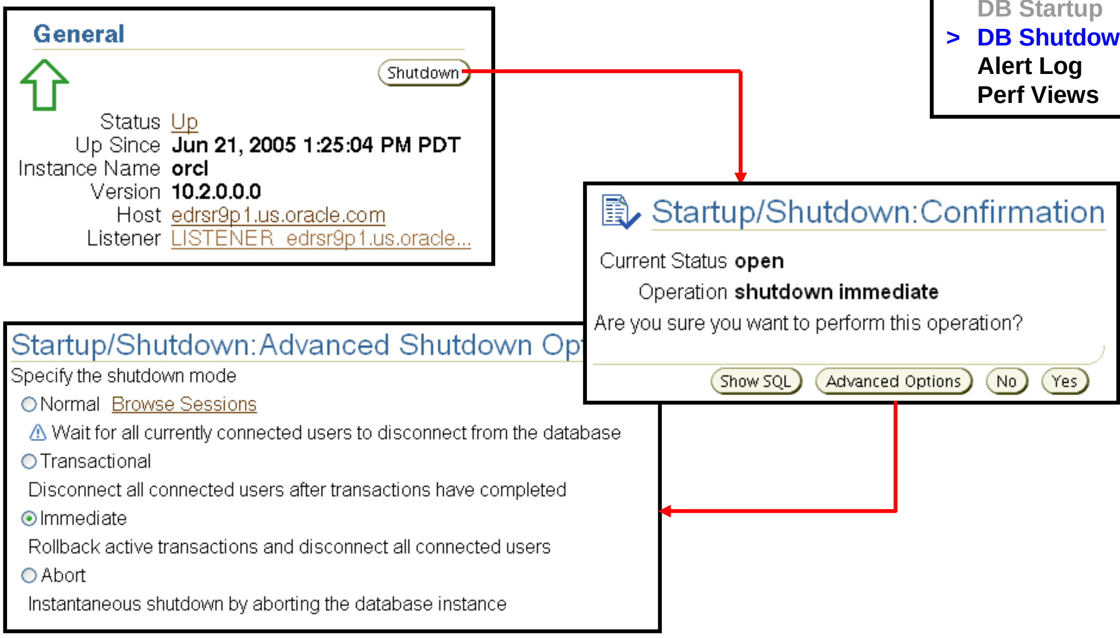
If any of the data files or online redo log files are not present when you attempt to open the database, then the Oracle server returns an error.

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 System Monitor (SMON) background process initiates instance recovery.

You can start up a database instance in restricted mode so that it is available to users with administrative privileges only. To start an instance in restricted mode, select the "Restrict access to database" option on the Advanced Startup Options page.

Shutting Down an Oracle Database Instance

Components
SQL*Plus
Init Params
DB Startup
> DB Shutdown
Alert Log
Perf Views



ORACLE

4-24

Shutting Down an Oracle Database Instance

If the instance is already started when you go to the Enterprise Manager Database Control page, you can click the Shutdown button to shut down the instance. If you then click the Advanced Options button, you can select the mode of the shutdown: Normal, Transactional, Immediate, or Abort.

Shutdown Modes

Shutdown Mode	A	I	T	N
Allows new connections	No	No	No	No
Waits until current sessions end	No	No	No	Yes
Waits until current transactions end	No	No	Yes	Yes
Forces a checkpoint and closes files	No	Yes	Yes	Yes

Shutdown mode:
- `A = ABORT`
- `I = IMMEDIATE`
- `T = TRANSACTIONAL`
- `N = NORMAL`

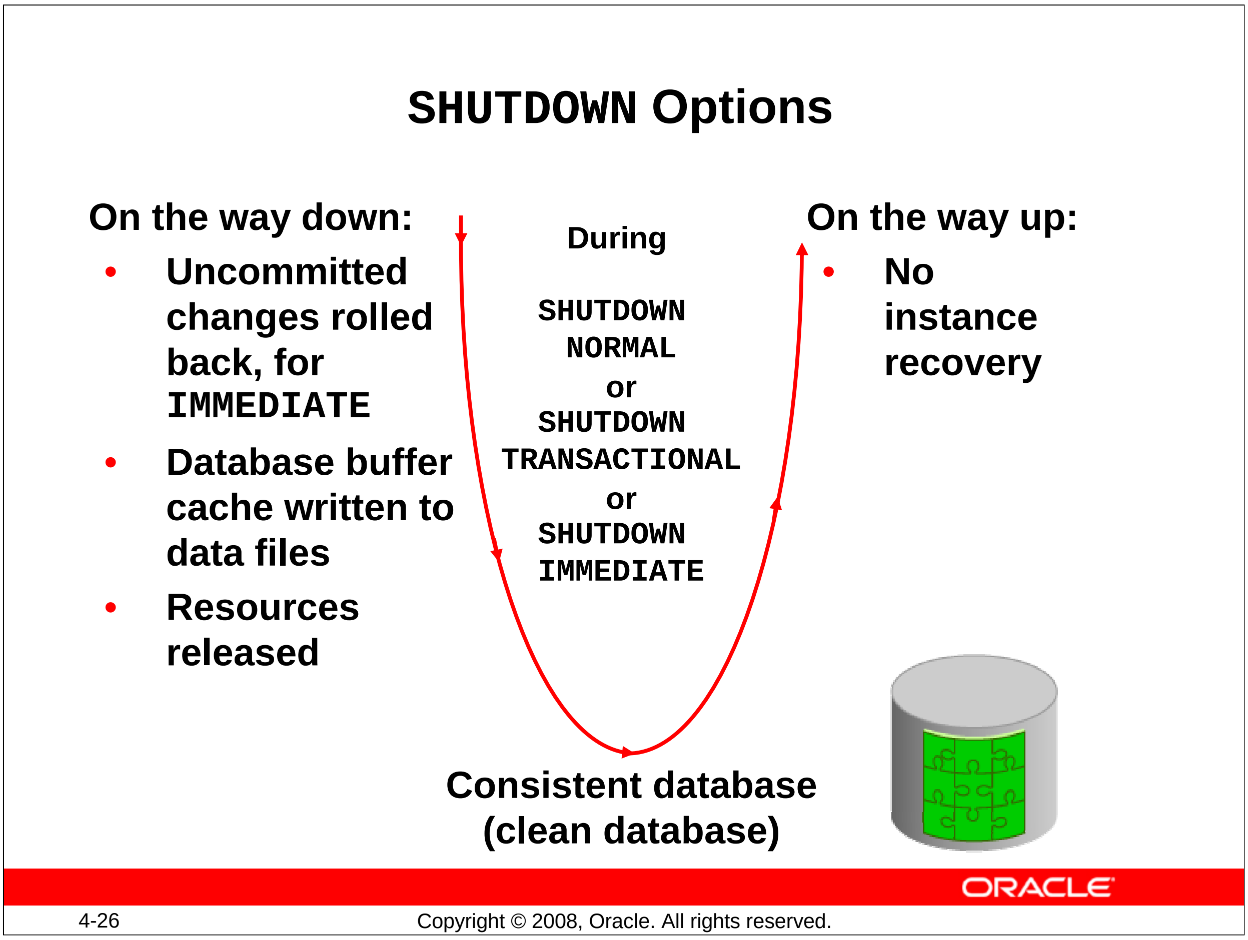
4-25

Shutdown Modes

Shutdown modes are progressively more accommodating of current activity in this order:

- `ABORT`: Performs the least amount of work before shutting down. Because this requires recovery before startup, use this only when necessary. This is typically used when no other form of shutdown works, when there are problems when starting the instance, or when you need to shut down immediately because of an impending situation, such as notice of a power outage within seconds.
- `IMMEDIATE`: Is the most typically used option. Uncommitted transactions are rolled back.
- `TRANSACTIONAL`: Allows transactions to finish
- `NORMAL`: Waits for sessions to disconnect

If you consider the amount of time that it takes to perform the shutdown, you find that `ABORT` is the fastest and `NORMAL` is the slowest.



SHUTDOWN Options

SHUTDOWN NORMAL

Normal is the default shutdown mode. A 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.
- The Oracle server 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 data, including the results from their current activity. A transactional database shutdown proceeds with the following conditions:

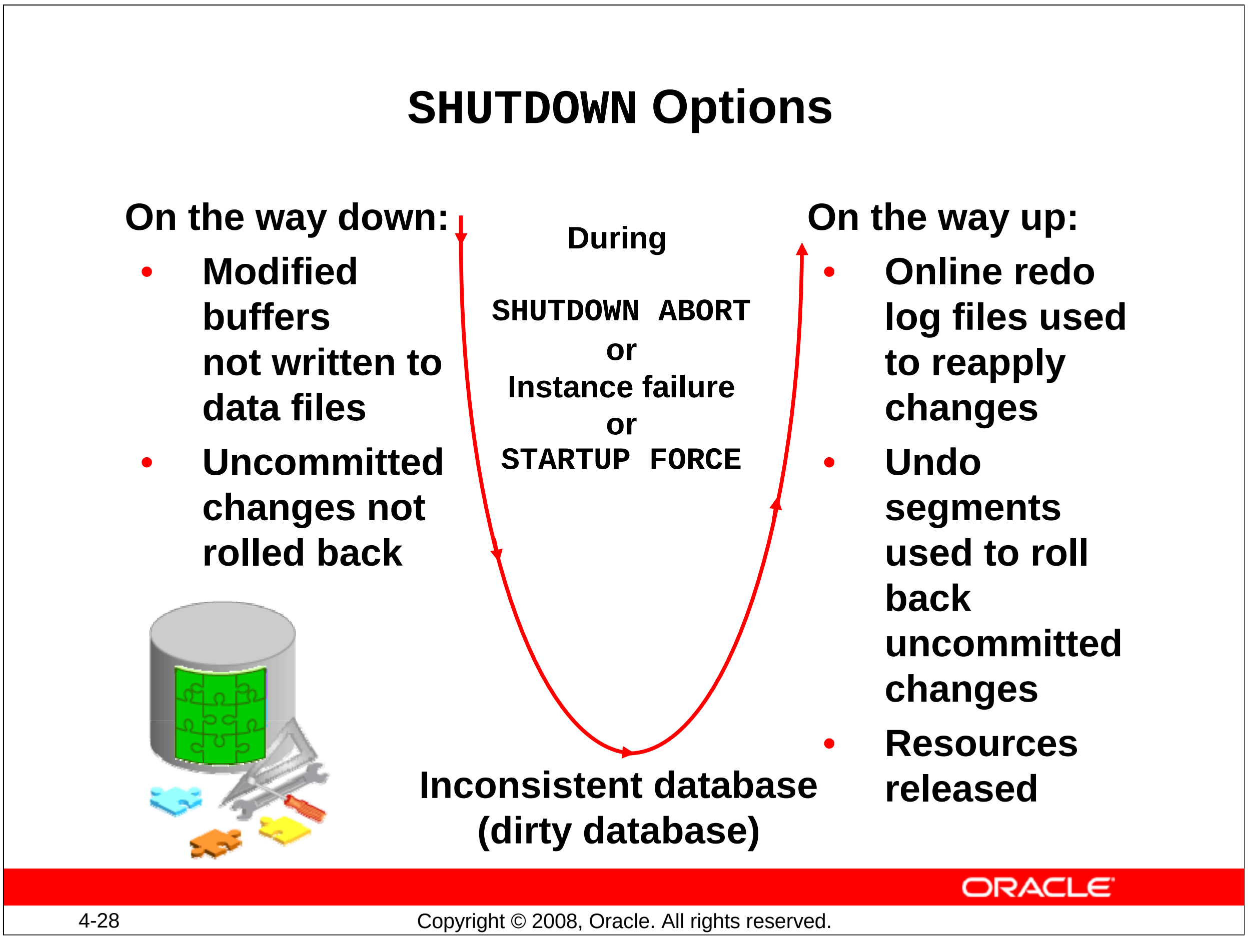
- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have been completed, a shutdown occurs immediately.
- The next startup does not require an instance recovery.

`SHUTDOWN` Options (continued)

`SHUTDOWN IMMEDIATE`

Immediate database shutdown proceeds with the following conditions:

- Current SQL statements being processed by the Oracle database are not completed.
- The Oracle server does not wait for the users who are currently connected to the database to disconnect.
- The Oracle server rolls back active transactions and disconnects all connected users.
- The Oracle server 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.
- The Oracle server 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.

Using SQL*Plus to Start Up and Shut Down

```
[oracle@EDRSR9P1 oracle]$ sqlplus dba1/oracle as sysdba

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area  285212672 bytes
Fixed Size                  1218472 bytes
Variable Size             250177624 bytes
Database Buffers           33554432 bytes
Redo Buffers                 262144 bytes
Database mounted.
Database opened.
SQL>
```


Using SQL*Plus to Start Up and Shut Down

You can also use SQL*Plus to start up, shut down, and otherwise change the state of the database. To use SQL*Plus for these tasks, you must log in as SYSDBA or SYSOPER. Then, use the equivalent commands for the Enterprise Manager functionality discussed earlier:

SHUTDOWN [<u>NORMAL</u> | TRANSACTIONAL | IMMEDIATE | ABORT]

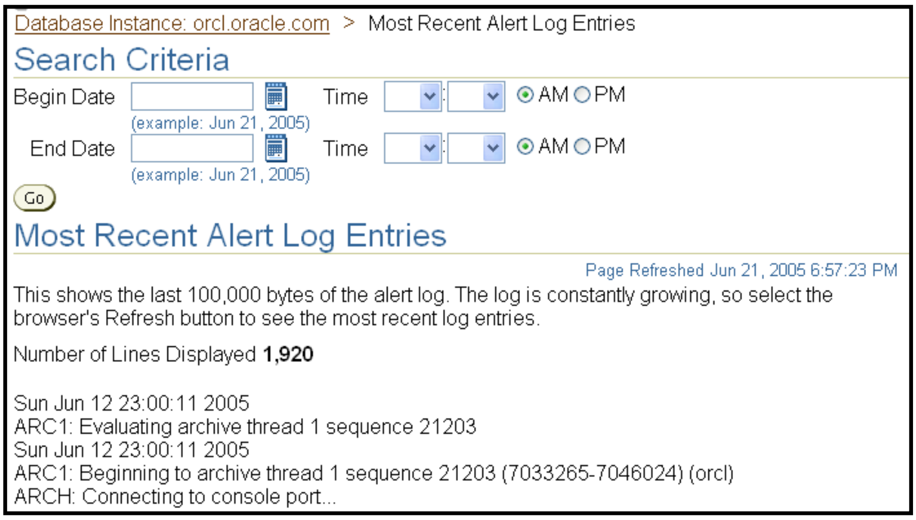
STARTUP [FORCE] [RESTRICT] [MOUNT | <u>OPEN</u> | NOMOUNT]

This enables you to include startup and shutdown operations as part of a script or batch process that performs tasks on the database, where the database needs to be in a particular state.

Viewing the Alert Log

Components
SQL*Plus
Init Params
DB Startup
DB Shutdown
> **Alert Log**
Perf Views

Database Home page > Related Links region > Alert Log Content



ORACLE

4-30

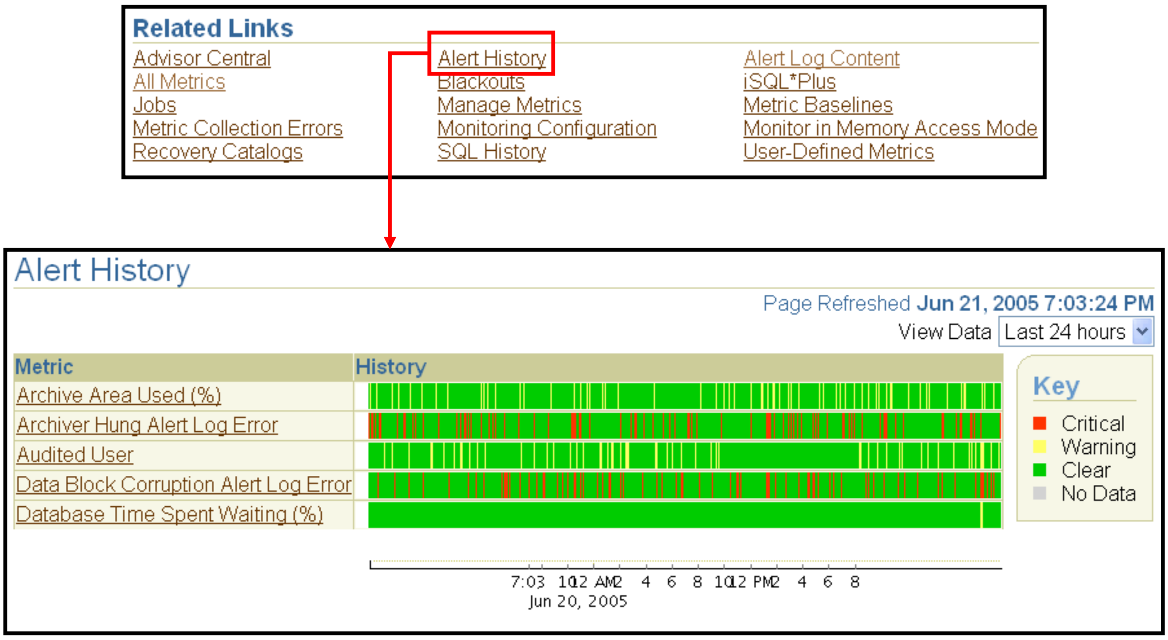
Viewing the Alert Log

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

- Any nondefault initialization parameters used at startup
- All internal errors (`ORA-600`), block corruption errors (`ORA-1578`), and deadlock errors (`ORA-60`) that occurred
- Administrative operations, such as the SQL statements `CREATE`, `ALTER`, `DROP DATABASE`, and `TABLESPACE`, 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

Enterprise Manager monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and informative messages. The file can grow to an unmanageable size. You can occasionally back up the alert file and delete the current alert file. When the database attempts to write to the alert file again, it re-creates a new one.

Viewing the Alert History



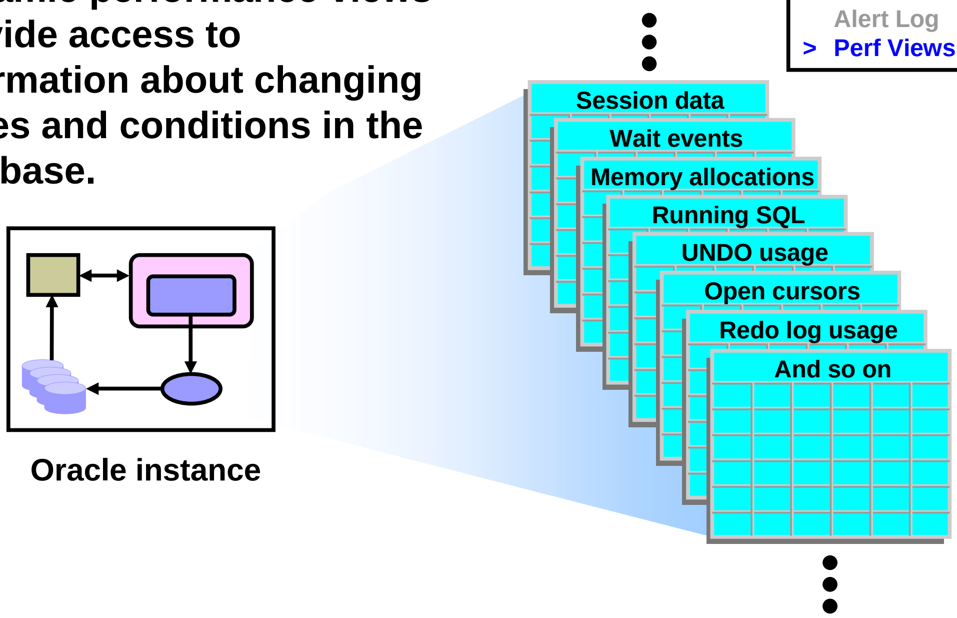
Viewing the Alert History

The Alert History page displays a chart that shows the alert history of the current database in segments of time, which you designate. An alert indicates a potential problem: either a warning or critical threshold for a monitored metric, or that a target is no longer available.

Dynamic Performance Views

Components
SQL*Plus
Init Params
DB Startup
DB Shutdown
Alert Log
> **Perf Views**

Dynamic performance views provide access to information about changing states and conditions in the database.



ORACLE

4-32

Dynamic Performance Views

The Oracle database also maintains a more dynamic set of data about the operation and performance of the database instance. These dynamic performance views are based on virtual tables that are built from memory structures inside the database server. That is, they are not conventional tables that reside in a database. This is why some of them can show you data before a database is mounted or open.

Dynamic performance views include information about:

- Sessions
- File states
- Progress of jobs and tasks
- Locks
- Backup status
- Memory usage and allocation
- System and session parameters
- SQL execution
- Statistics and metrics

Note: The `DICT` and `DICT_COLUMNS` views also contain the names of these dynamic performance views.

Dynamic Performance Views: Usage Examples

a

```
SQL> SELECT sql_text, executions FROM v$sql
WHERE cpu_time > 200000;
```

b

```
SQL> SELECT * FROM v$session WHERE machine =
'EDRSR9P1' and logon_time > SYSDATE - 1;
```

c

```
SQL> SELECT sid, ctime FROM v$lock WHERE
block > 0;
```

ORACLE

4-33

Dynamic Performance Views: Usage Examples

A frequent user of these views is Enterprise Manager, but users can also query these views as needed. The three examples shown in the slide answer the following questions:

a. What are the SQL statements and their associated number of executions where the CPU time consumed is greater than 200,000 microseconds?
b. What sessions logged in from the `EDRSR9P1` computer within the last day?
c. What are the session IDs of any sessions that are currently holding a lock that is blocking another user, and how long has that lock been held?

Dynamic Performance Views: Considerations

- These views are owned by the `SYS` user.
- Different views are available at different times:
  - The instance has been started.
  - The database is mounted.
  - The database is open.
- You can query `V$FIXED_TABLE` to see all the view names.
- These views are often referred to as "v-dollar views."
- Read consistency is not guaranteed on these views because the data is dynamic.

4-34

Dynamic Performance Views: Considerations

Some dynamic views contain data that is not applicable to all states of an instance or database. For example, if an instance has just been started, but no database is mounted, you can query `V$BGPROCESS` to see the list of background processes that are running. But you cannot query `V$DATAFILE` to see the status of database data files because it is the mounting of a database that reads the control file to find out about the data files associated with a database.

Summary

In this lesson, you should have learned how to:

- Start and stop the Oracle database and components
- Use Enterprise Manager and describe its high-level functionality
- Access a database with SQL*Plus and *i*SQL*Plus
- Modify database initialization parameters
- Describe the stages of database startup
- Describe the database shutdown options
- View the alert log
- Access dynamic performance views

4-35

Practice Overview: Managing the Oracle Instance

This practice covers the following topics:

- Navigating in Enterprise Manager
- Viewing and modifying initialization parameters
- Stopping and starting the database instance
- Viewing the alert log
- Connecting to the database by using SQL*Plus and *i*SQL*Plus

4-36

Managing Database Storage Structures

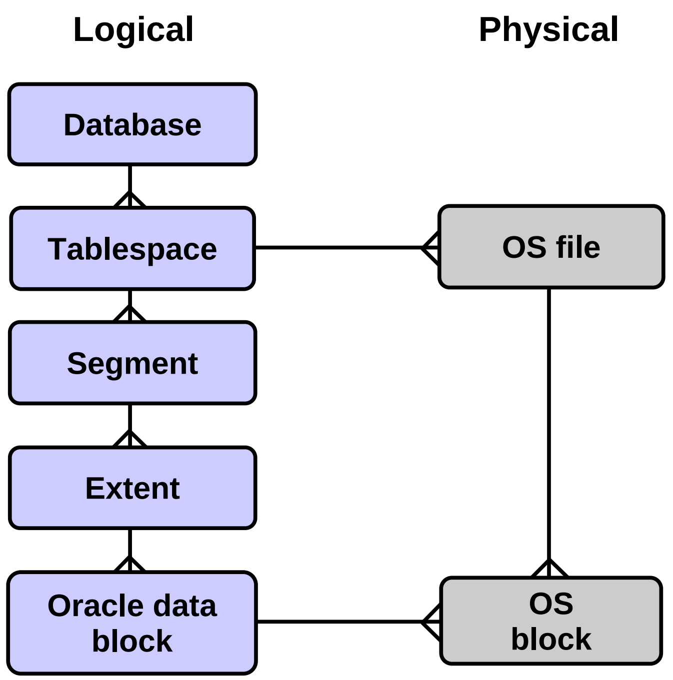
Objectives

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

- Describe how table row data is stored in blocks
- Define the purpose of tablespaces and data files
- Create and manage tablespaces
- Obtain tablespace information
- Describe the main concepts and functionality of Automatic Storage Management (ASM)

5-2

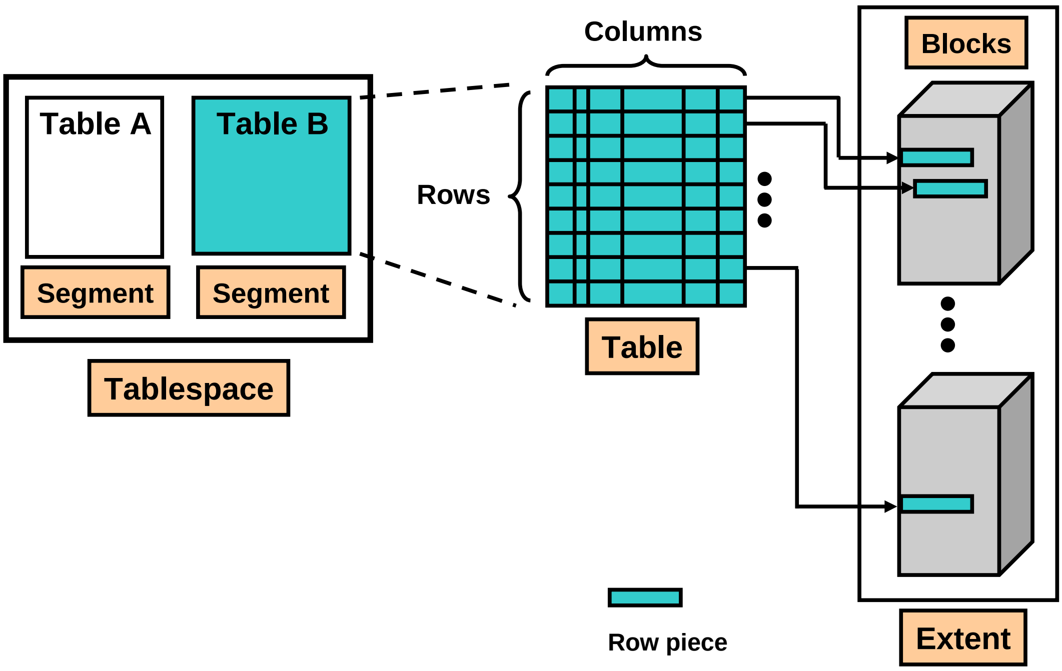
Storage Structures



Storage Structures

A database is divided into logical storage units called tablespaces. Each tablespace has many logical Oracle data blocks. The `DB_BLOCK_SIZE` parameter specifies how large a logical block is. A logical block can range from 2 KB to 32 KB in size. The default size is 8 KB. A specific number of contiguous logical blocks form an extent. A set of extents that are allocated for a certain logical structure form one segment. An Oracle data block is the smallest unit of logical I/O.

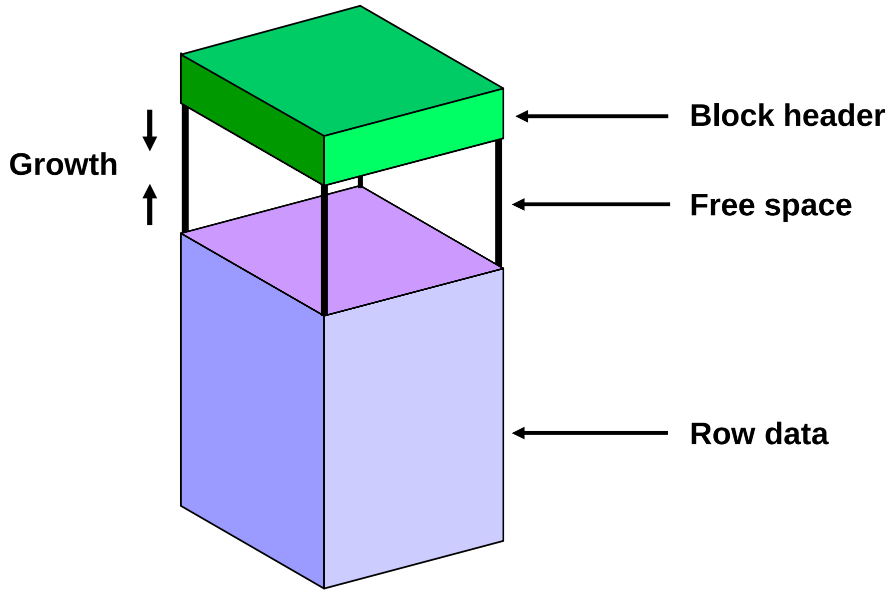
How Table Data Is Stored



How Table Data Is Stored

When a table is created, a segment is created to hold its data. A tablespace contains a collection of segments. Logically, a table contains rows of column values. A row is ultimately stored in a database block in the form of a row piece. It is called a row piece because under some circumstances the entire row may not be stored in one place. This happens when an inserted row is too large to fit into a single block or when an update causes an existing row to outgrow its current space.

Anatomy of a Database Block



Database Block: Contents

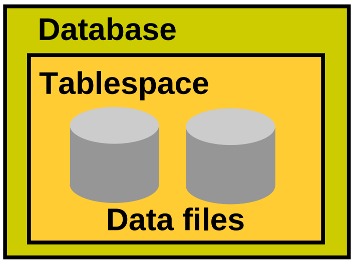
Oracle data blocks contain the following:

- **Block header:** The block header contains the segment type (such as table or index), data block address, table directory, row directory, and transaction slots of size 23 bytes each, which are used when modifications are made to rows in the block. The block header grows downward from the top.
- **Row data:** This is the actual data for the rows in the block. Row data space grows upward from the bottom.
- **Free space:** Free space is in the middle of the block. This enables the header and the row data space to grow when necessary. Row data takes up free space as new rows are inserted or columns of existing rows are updated with larger values. The examples of events that cause header growth are when the row directory needs more row entries or more transaction slots are required than initially configured. Initially, the free space in a block is contiguous. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the Oracle server when necessary.

Tablespaces and Data Files

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

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



5-6

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 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 to the operating system on which the Oracle software runs.
- 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 two tablespaces (the required `SYSTEM` and `SYSAUX` tablespaces), each with 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 can potentially have as many as 65,534 data files.

Oracle Managed Files (OMF)

Specify file operations in terms of database objects rather than file names.

Parameter	Description
`DB_CREATE_FILE_DEST`	Defines the location of the default file system directory for data files and temporary files
`DB_CREATE_ONLINE_LOG_DEST_n`	Defines the location for redo log files and control file creation
`DB_RECOVERY_FILE_DEST`	Defines the location for RMAN backups

Example:

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/oradata';
SQL> CREATE TABLESPACE tbs_1;
```

ORACLE

Oracle Managed Files (OMF)

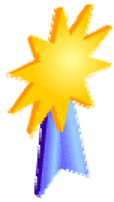
Oracle Managed files (OMF) eliminate the need for you, to directly manage the operating system files comprising an Oracle database. You specify operations in terms of database objects rather than file names. The database internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Redo log files
- Control files
- Archived logs
- Block change tracking files
- Flashback logs
- RMAN backups

A database can have a mixture of Oracle-managed and unmanaged files. The file system directory specified by either of these parameters must already exist: the database does not create it. The directory must also have permissions to allow the database to create the files in it.

The example shows that after `DB_CREATE_FILE_DEST` is set, the `DATAFILE` clause can be omitted from a `CREATE TABLESPACE` statement. The data file is created in the location specified by `DB_CREATE_FILE_DEST`.

Space Management in Tablespaces



- **Locally managed tablespace:**
  - Free extents are managed in the tablespace.
  - A bitmap is used to record free extents.
  - Each bit corresponds to a block or group of blocks.
  - The bit value indicates free or used extents.
  - The use of locally managed tablespaces is recommended.
- **Dictionary-managed tablespace:**
  - Free extents are managed by the data dictionary.
  - Appropriate tables are updated when extents are allocated or unallocated.
  - These tablespaces are supported only for backward compatibility.

ORACLE

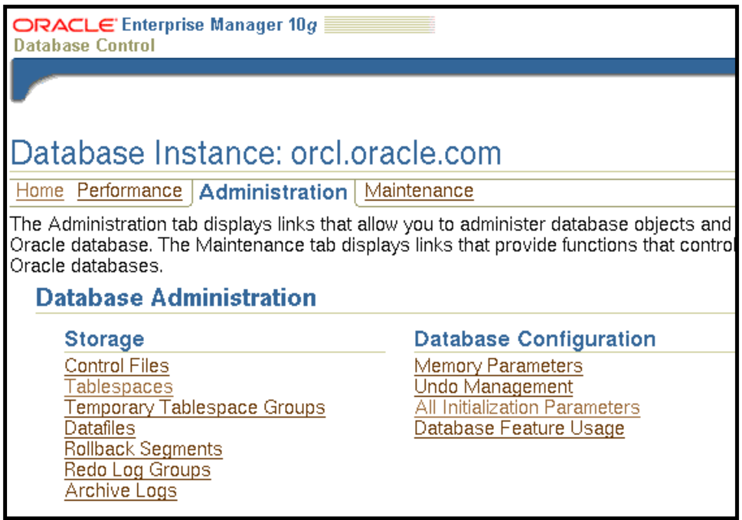
5-8

Space Management in Tablespaces

Tablespaces allocate space in extents. Tablespaces can be created to use one of the following two 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 unallocated. This is for backward compatibility; it is recommended that you use locally managed tablespaces.

Exploring the Storage Structure



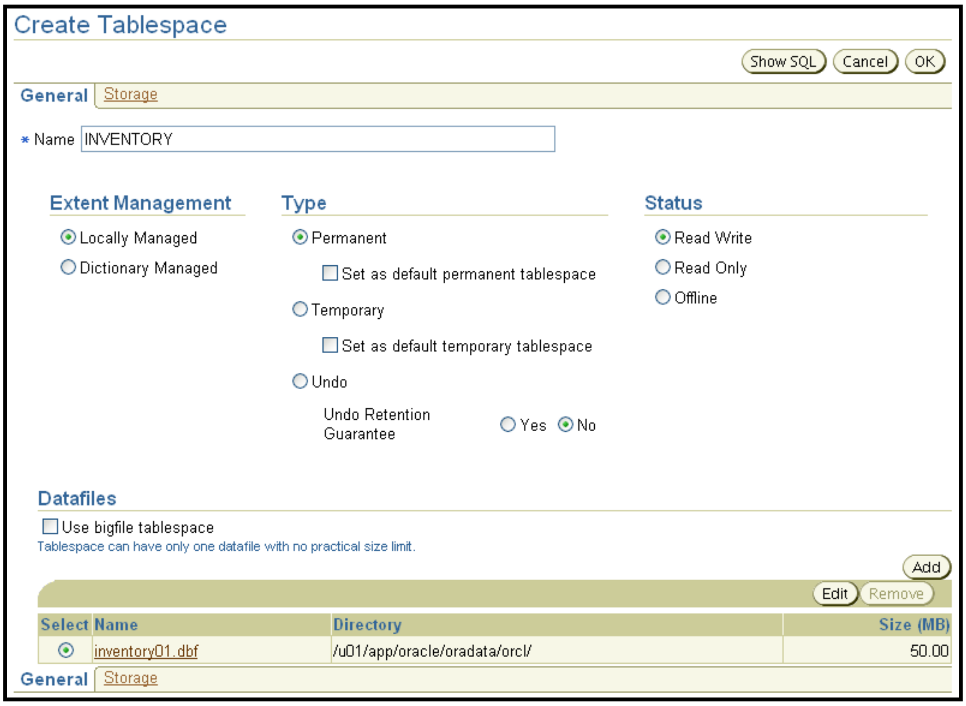
Click the links to view detailed information.

5-9

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 about each structure can be obtained by clicking the links in the Storage region of the Administration page.

Creating a New Tablespace



5-10

Creating a New Tablespace

To create a tablespace, perform the following steps:

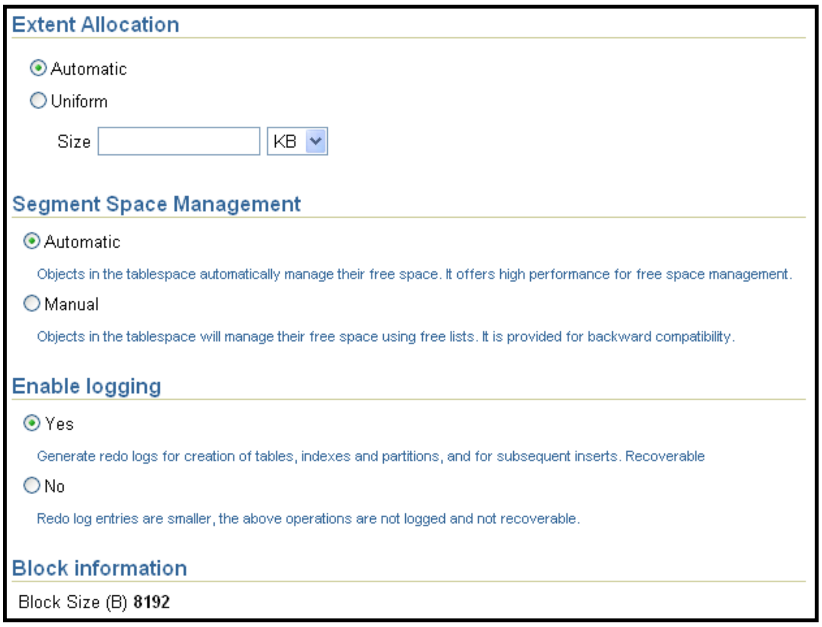
1. Click the Administration tab, and then click Tablespaces under the Storage heading.
2. Click Create.
   Note: If you want to create a tablespace that is like an existing tablespace, then select an existing tablespace and select Create Like from the Actions menu. Click Go.
   The Create Tablespace 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 manage extents more actively, and data dictionary access is required for tracking them. Dictionary-managed tablespaces are being deprecated. Oracle does not recommend their use.
5. Under the Type heading, select Permanent. Permanent tablespaces store permanent database objects that are created by the system or users.

Creating a New Tablespace (continued)

6. Under the Status heading, select Read Write. The Read Write status means that users can read and write to the tablespace after it is created. This is the default.
7. In the Datafiles region of the page, click Add to add data files to the tablespace; a tablespace must have at least one file. Bigfile tablespaces are used with extremely large databases, where Oracle's Automatic Storage Management (ASM) or other logical volume managers support striping or redundant array of independent disks (RAID) and dynamically extensible logical volumes.
8. On the Add Datafiles page, enter a file name. Accept the default for File Directory, and enter a file size.
9. In the Storage region, you can select "Automatically extend datafile when full (AUTOEXTEND)" and then specify an amount in the Increment field. This causes the data file to extend automatically each time it runs out of space. It is limited, of course, by the physical media on which it resides. Leave Maximum File Size as Unlimited. Click OK. You are returned to the Create Tablespace page.
10. Click the Storage tab. The Edit Tablespace page appears.
11. Accept all the defaults on the Storage page.

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

Storage for Locally Managed Tablespaces



ORACLE

5-12

Storage for Locally Managed Tablespaces

The extents within a locally managed tablespace can be allocated in one of these two ways:

- **Automatic:** Also called autoallocate, it specifies that the sizes of the extents within the tablespace are system managed. You cannot specify an extent size. You cannot specify Automatic for a temporary tablespace.
- **Uniform:** It specifies that the tablespace is managed with uniform extents of a size that you specify. The default size is 1 MB. All extents of temporary tablespaces are uniform and default to that value. You cannot specify Uniform for an undo tablespace.

Segment space management within a locally managed tablespace can be specified as:

- **Automatic:** The Oracle database uses bitmaps to manage the free space within segments. The bitmap describes the status of each data block within a segment with respect to the amount of space in the block that is available for inserting rows. As more or less space becomes available in a data block, its new state is reflected in the bitmap. With bitmaps, the Oracle database manages free space more automatically and, thus, this form of space management is called Automatic Segment Space Management (ASSM).

Storage for Locally Managed Tablespaces (continued)

- **Manual:** This specifies 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; it is recommended that you use ASSM.

Advantages of Locally Managed Tablespaces

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

- Local management avoids recursive space management operations. This occurs 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 tablespaces, use the `DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL` procedure to do this. For details about the use of this procedure, see *PL/SQL Packages and Types Reference* and the *Database Administrator's Guide*.

Logging

Changes made to objects in the tablespace are written to the redo log. If logging is not enabled, then any direct loads using SQL*Loader and direct load INSERT operations are not written to the redo log, and the objects are thus unrecoverable in the event of data loss. So, when an object is created without Logging enabled, you must back up those objects, if you want them to be recoverable.

For more details about the logging clause, see the *Oracle Database SQL Reference*.

Block Information

This region shows the block size that is used for the tablespace being created. It is displayed here as a read-only value. If you set any of the alternate block size initialization parameters (`DB_nK_CACHE_SIZE`), then those other values would be listed here as an option. For more information about defining other block sizes, see the *Oracle Database Administrator's Guide*.

Tablespaces in the Preconfigured Database

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

Selection Mode Single

Create

Edit View Delete Actions Add Datafile Go

Select	Name △	Size (MB)	Used (MB)	Used (%)	Free (MB)	Status	Datafiles	Type	Extent Management	Segment Management
◉	EXAMPLE	100.0	68.2	68.2	31.8	✓	1	PERMANENT	LOCAL	AUTO
○	SYSAUX	370.0	361.4	97.7	8.6	✓	1	PERMANENT	LOCAL	AUTO
○	SYSTEM	490.0	484.8	98.9	5.2	✓	1	PERMANENT	LOCAL	MANUAL
○	TEMP	20.0	0.0	0.0	20.0	✓	1	TEMPORARY	LOCAL	MANUAL
○	UNDOTBS1	35.0	12.3	35.2	22.7	✓	1	UNDO	LOCAL	MANUAL
○	USERS	40.0	38.4	95.9	1.6	✓	1	PERMANENT	LOCAL	AUTO

Total Size (MB) **1,055.0**
Total Used (MB) **965.1**
Total Free (MB) **89.9**

✓ Online ✗ Offline Read Only

Tablespaces in the Preconfigured Database

The following tablespaces are created in the preconfigured database in this course:

- **SYSTEM:** The SYSTEM tablespace is used by the Oracle 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 SYS user 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 earlier releases of the Oracle database, now use the SYSAUX tablespace. Every Oracle Database10*g* or later release must have a SYSAUX tablespace.

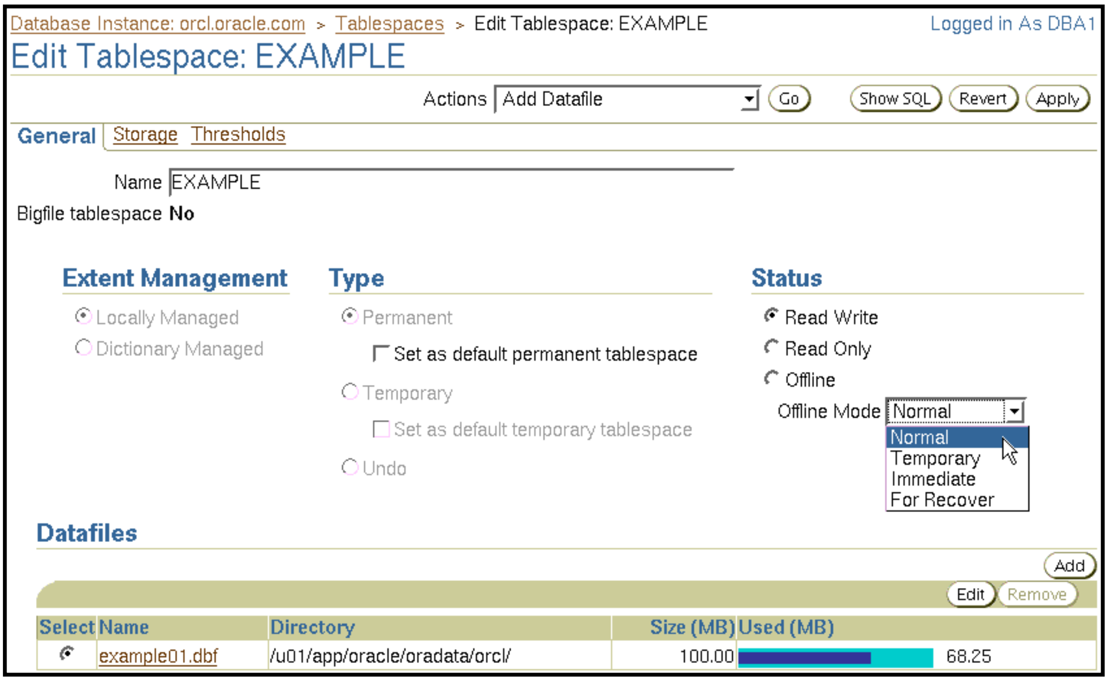
  In Enterprise Manager, you can see a pie chart of the contents of this tablespace. To do this, click Tablespaces on the Administration page. Select SYSAUX and click Edit. Then, click the Occupants tab. After creation, you can monitor the space usage of each occupant inside the SYSAUX tablespace by using EM. If you detect that a component is taking too much space in the SYSAUX tablespace, or if you anticipate that it will, you can move the occupant into a different tablespace by selecting one of the occupants and clicking Change Tablespace.

Tablespaces in the Preconfigured Database (continued)

- **`TEMP`:** Your temporary tablespace is used when you execute a SQL statement that requires the creation of temporary segments (such as a large sort or the creation of an index). Just like each user is assigned a default tablespace for storing created data objects, each user is assigned a temporary tablespace. The best practice is to define a default temporary tablespace for the database, which is assigned to any newly created user, unless otherwise specified. 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, the Oracle database assigns this tablespace to the user.
- **`UNDOTBS1`:** This is the undo tablespace used by the database server to store undo information. If a database uses Automatic Undo Management, then it must have exactly one active undo tablespace at any given time. This tablespace is created at database creation time.
- **`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 on the sample schemas.

Note: To simplify administration, it is common to have a tablespace for indexes alone.

Altering a Tablespace



ORACLE

5-16

Altering a Tablespace

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

Renaming: 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. Any of the following three states may not be available because their availability depends on the type of tablespace:

- **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 be completed (committed or rolled back), but no further data manipulation language (DML) operations are allowed on objects 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 can use the following options:
  - **Normal:** A tablespace can be taken offline normally if no error conditions exist for any of the data files of the tablespace. The Oracle database ensures that all data is written to disk by taking 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. The Oracle database takes the data files (which are not already offline) offline, performing checkpointing on 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 are 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 the Oracle database 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.

Changing the size: You can add space to an existing tablespace by either adding data files to the tablespace or by changing 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 Datafile page.
- 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 to do so, you get the following error:

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

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

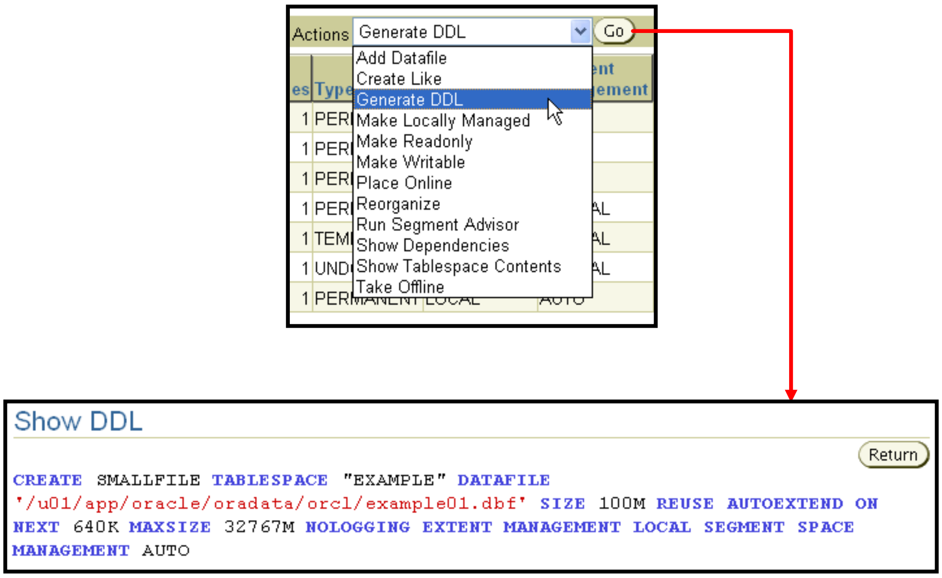
Altering a Tablespace (continued)

Thresholds: Click Thresholds to change the point at which a warning or critical level of space usage is reached on the tablespace. You have three options:

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

Note: It may take a few minutes for a threshold alert to register.

Actions with Tablespaces



ORACLE

Actions with Tablespaces

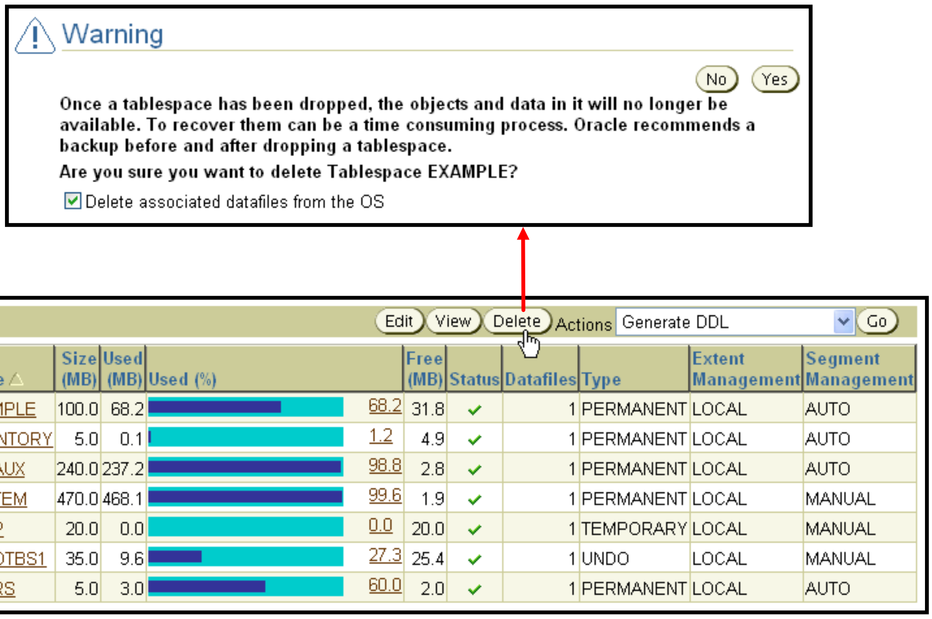
Using the Actions menu, you can perform a variety of tasks with your tablespaces. Select a tablespace and then the action that you want to perform:

- **Add Datafile:** Adds a data file to the tablespace, which makes the tablespace larger
- **Create Like:** Creates another tablespace by using the tablespace as a template
- **Generate DDL:** Generates the data definition language (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:** Converts the tablespace to locally managed if the tablespace is currently dictionary managed. This conversion is only one way; you cannot convert the tablespace back to dictionary managed.
- **Make Readonly:** Stops all writes to the tablespace. Current transactions are allowed to complete, but no new DML or other write activities are allowed to start on the tablespace. This appears only if the tablespace is currently not read-only.
- **Make Writable:** Allows DML and other write activities to be initiated on objects in the tablespace. This appears only if the tablespace is currently not writable.

Actions with Tablespaces (continued)

- **Place Online:** Brings a currently offline tablespace online
- **Reorganize:** Starts the Reorganization Wizard, which you can use to move objects around within the tablespace to reclaim space that otherwise may not be used. This is a task that should be performed during off-peak usage of the objects in the tablespace.
- **Run Segment Advisor:** Starts the Segment Advisor, which you can use to determine whether an object has space available for reclamation on the basis of the level of space fragmentation within the object. At the tablespace level, advice is generated for every segment in the tablespace.
- **Show Dependencies:** Shows objects that this tablespace depends on or objects that depend on this tablespace
- **Show Tablespace Contents:** Shows information about all the segments in the tablespace, including a graphical map of all of the extents
- **Take Offline:** Makes a currently online tablespace unavailable. The tablespace is not deleted or dropped; it is just unavailable.

Dropping Tablespaces



Select	Name	Size (MB)	Used (MB)	Used (%)	Free (MB)	Status	Datafiles	Type	Extent Management	Segment Management
◉	EXAMPLE	100.0	68.2	68.2	31.8	✓	1	PERMANENT	LOCAL	AUTO
○	INVENTORY	5.0	0.1	1.2	4.9	✓	1	PERMANENT	LOCAL	AUTO
○	SYSAUX	240.0	237.2	98.8	2.8	✓	1	PERMANENT	LOCAL	AUTO
○	SYSTEM	470.0	468.1	99.6	1.9	✓	1	PERMANENT	LOCAL	MANUAL
○	TEMP	20.0	0.0	0.0	20.0	✓	1	TEMPORARY	LOCAL	MANUAL
○	UNDOTBS1	35.0	9.6	27.3	25.4	✓	1	UNDO	LOCAL	MANUAL
○	USERS	5.0	3.0	60.0	2.0	✓	1	PERMANENT	LOCAL	AUTO

5-21

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. Also, if you are using OMF, the underlying operating system files are removed. Otherwise, without OMF, you can optionally direct the Oracle server to delete the operating system files (data files) that constitute the dropped tablespace. If you do not direct the Oracle server to delete the data files at the same time as it deletes the tablespace, you must later use the appropriate commands of your operating system if you want them to be deleted.

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

```
SELECT tablespace_name, status, contents, logging, extent_management,
allocation_type, segment_space_management
FROM dba_tablespaces
```

TABLESPACE_NAME	STATUS	CONTENTS	LOGGING	EXTENT_MAN	ALLOCATIO	SEGMEN
SYSTEM	ONLINE	PERMANENT	LOGGING	LOCAL	SYSTEM	MANUAL
UNDOTBS1	ONLINE	UNDO	LOGGING	LOCAL	SYSTEM	MANUAL
SYSAUX	ONLINE	PERMANENT	LOGGING	LOCAL	SYSTEM	AUTO
TEMP	ONLINE	TEMPORARY	NOLOGGING	LOCAL	UNIFORM	MANUAL
USERS	ONLINE	PERMANENT	LOGGING	LOCAL	SYSTEM	AUTO
EXAMPLE	ONLINE	PERMANENT	NOLOGGING	LOCAL	SYSTEM	AUTO
INVENTORY	ONLINE	PERMANENT	LOGGING	LOCAL	SYSTEM	AUTO

```
SELECT ts#, name FROM v$tablespace
```

TS#	NAME
0	SYSTEM
1	UNDOTBS1
2	SYSAUX
4	USERS
3	TEMP
6	EXAMPLE
7	INVENTORY

ORACLE

5-22

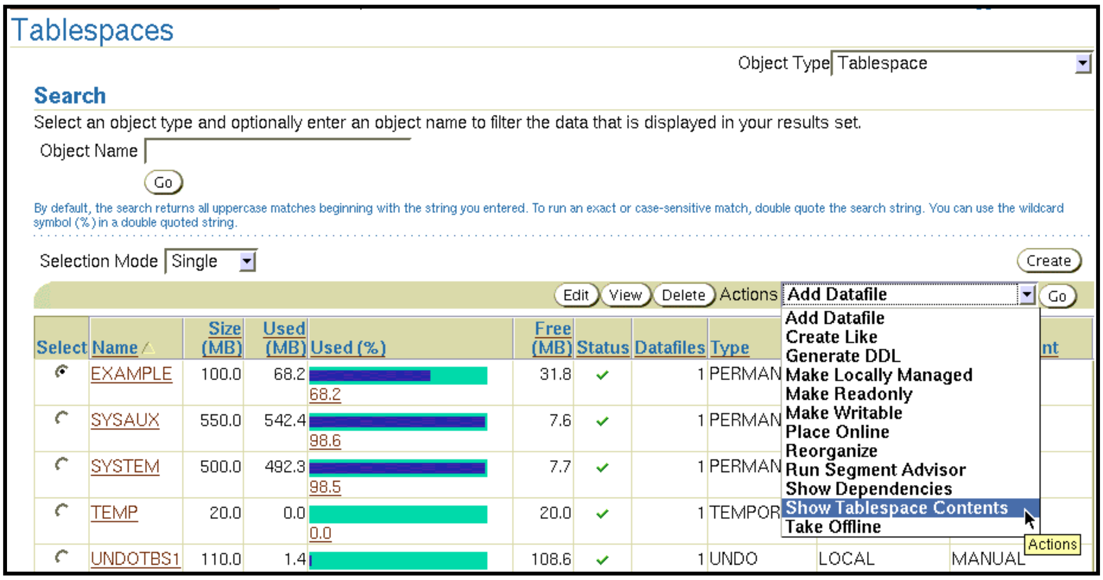
Viewing Tablespace Information

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

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`

Gathering Storage Information



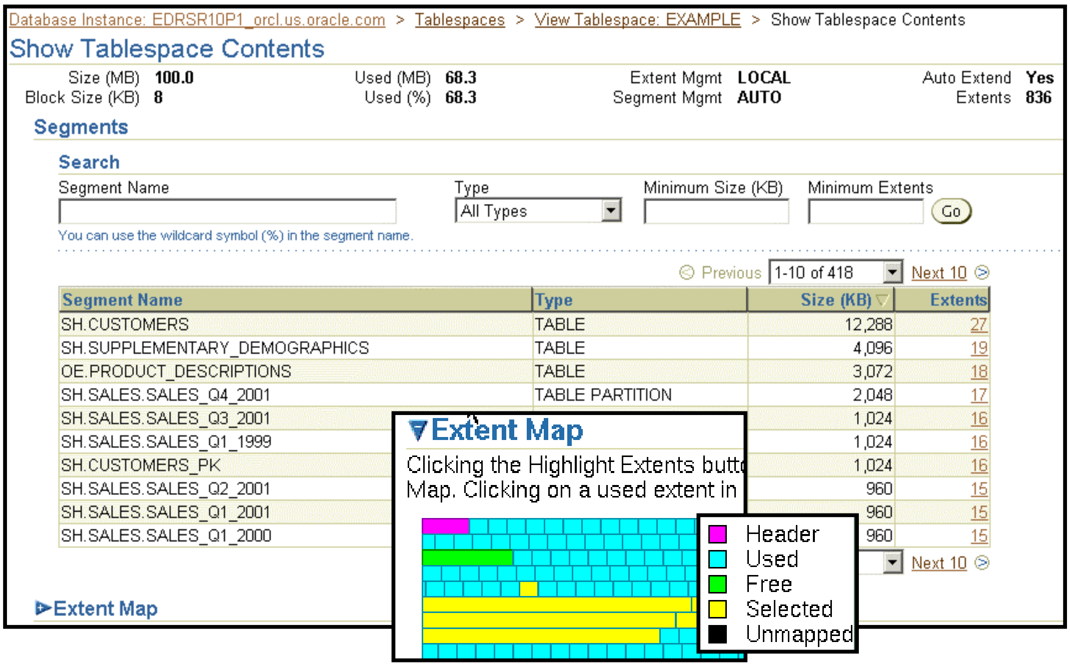
ORACLE

5-23

Gathering Storage Information

To view and modify tablespace information in EM, select Administration > Tablespaces. Use the buttons or the Actions drop-down list to navigate to your destination.

Viewing Tablespace Contents



Viewing Tablespace Contents

On the Show Tablespace Contents page, detailed information about the tablespace is displayed, including a list of the segments in the tablespace, the type of each segment, the segment size, and the number of extents that comprise each segment. Any of these four values can be used to sort the list by clicking the column header, or to filter the list by entering values in the Search region. For a dictionary-managed tablespace, additional columns are displayed:

- Max Extents
- Next
- Percent Increase

To see a list of extents, click the link in the Extents column.

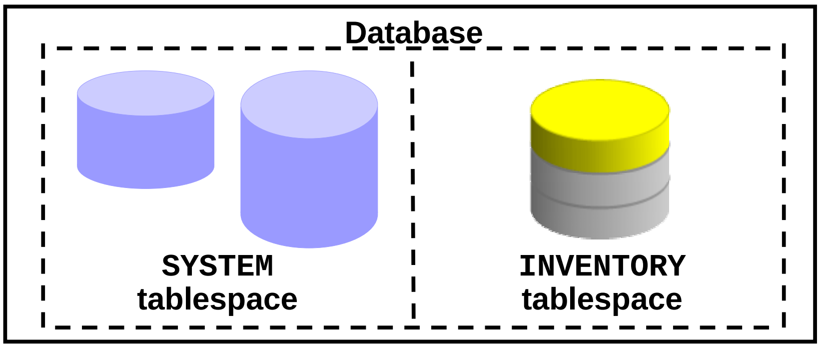
To view extents in a graphical way, expand the "Extent map" and move the cursor over individual extents. The following information is displayed:

- Name of the segment the extent belongs to
- Extent ID
- Block ID
- Extent size in blocks
- Data file in which the extent is stored

Enlarging the Database

You can enlarge the database in the following ways:

- **Creating a new tablespace**
- **Adding a data file to an existing tablespace**
- **Increasing the size of a data file**
- **Providing for the dynamic growth of a data file**



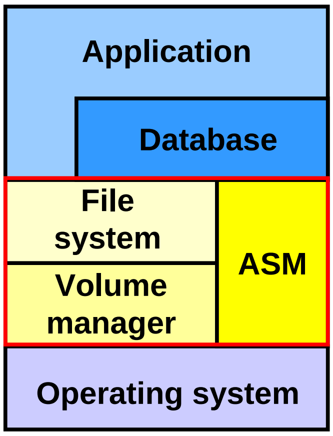
Enlarging the Database

These activities can be performed with Enterprise Manager or with SQL statements. In the end, the size of the database can be described as the sum of all of its tablespaces.

What Is Automatic Storage Management?

Automatic Storage Management

- Is a portable and high-performance cluster file system
- Manages Oracle database files
- Spreads data across disks to balance load
- Mirrors data
- Solves many storage management challenges



5-26

What Is Automatic Storage Management?

ASM provides a vertical integration of the file system and the volume manager that is specifically built for Oracle database files. ASM can provide management for single symmetric multiprocessing (SMP) machines or across multiple nodes of a cluster for Oracle Real Application Clusters (RAC) support.

ASM distributes input/output (I/O) load across all available resources to optimize performance while removing the need for manual I/O tuning. ASM helps DBAs manage a dynamic database environment by enabling them to increase the database size without having to shut down the database to adjust storage allocation.

ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied storage mechanisms. Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per-file basis.

ASM capabilities save the DBA’s time by automating manual storage and thereby increasing the DBA’s ability to manage more and larger databases with increased efficiency.

ASM: Key Features and Benefits

ASM

- Stripes files, but not logical volumes
- Provides online disk reconfiguration and dynamic rebalancing
- Allows for adjustable rebalancing speed
- Provides redundancy on a per-file basis
- Supports only Oracle database files
- Is cluster aware
- Is automatically installed

ORACLE

5-27

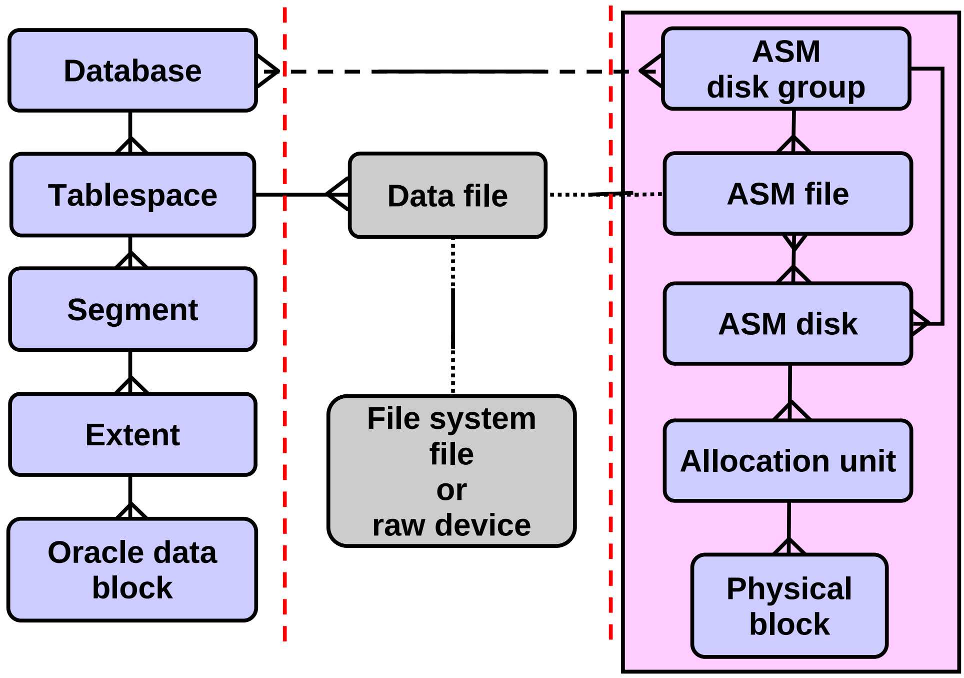
ASM: Key Features and Benefits

ASM divides files into extents (different from the data file extents discussed earlier) and spreads the extents for each file evenly across all the disks. It uses an index technique to track the placement of each extent. When the storage capacity changes, ASM does not restripe all the data but moves an amount of data proportional to the amount of storage added or removed to evenly redistribute the files and maintain a balanced load across the disks. This is done while the database is active.

You can increase the speed of a rebalance operation to cause it to finish sooner, or decrease the speed to reduce the impact on the I/O subsystem. ASM provides mirroring protection without the need to purchase a third-party Logical Volume Manager. One unique advantage of ASM is that mirroring is applied on a file basis, rather than on a volume basis. Therefore, the same disk group can contain a combination of mirrored or nonmirrored files.

ASM supports data files, log files, control files, archive logs, Recovery Manager (RMAN) backup sets, and other Oracle database file types. It supports RAC and eliminates the need for a Cluster Logical Volume Manager or a Cluster File System.

ASM: Concepts



ASM: Concepts

ASM does not eliminate any preexisting database functionality. Existing databases are able to operate as they always have. You can create new files as ASM files and leave existing files to be administered in the old way, or you can eventually migrate them to ASM.

The diagram depicts the relationships that exist between the various storage components inside an Oracle database that uses ASM. The left and middle parts of the diagram show the relationships that exist in previous releases. On the right are the new concepts introduced by ASM.

Database files can be stored as ASM files. At the top of the new hierarchy are ASM disk groups. Any single ASM file is contained in only one disk group. However, a disk group may contain files belonging to several databases, and a single database may use storage from multiple disk groups. As you can see, one disk group is made up of multiple ASM disks, and each ASM disk belongs to only one disk group. ASM files are always spread across all the ASM disks in the disk group. ASM disks are partitioned in allocation units (AU) of one megabyte each. An allocation unit is the smallest contiguous disk space that ASM allocates. ASM does not allow an Oracle block to be split across allocation units.

Note: This graphic deals with only one type of ASM file: data file. However, ASM can be used to store other database file types.

Summary

In this lesson, you should have learned how to:

- Describe how table row data is stored in blocks
- Define the purpose of tablespaces and data files
- Create and manage tablespaces
- Obtain tablespace information
- Describe the main concepts and functionality of Automatic Storage Management (ASM)

5-29

Practice Overview: Managing Database Storage Structures

This practice covers the following topics:

- Creating tablespaces
- Gathering information about tablespaces

6

Administering User Security

Objectives

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

- **Create and manage database user accounts**
  - **Authenticate users**
  - **Assign default storage areas (tablespaces)**
- **Grant and revoke privileges**
- **Create and manage roles**
- **Create and manage profiles**
  - **Implement standard password security features**
  - **Control resource usage by users**

ORACLE

6-2

Objectives

The following terms relate to administering database users and assist you in understanding the objectives:

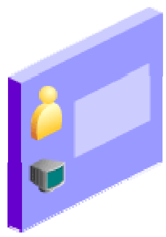
- A **database user account** is a means to organize the ownership of and access to database objects.
- A **password** is an authentication by the Oracle database.
- A **privilege** is a right to execute a particular type of SQL statement or to access another user's object.
- A **role** is a named group of related privileges that are granted to users or to other roles.
- **Profiles** impose a named set of resource limits on database usage and instance resources.
- **Quota** is a space allowance in a given tablespace. This is one of the ways by which you can control resource usage by users.

Database User Accounts

> User
Authentication
Privilege
Role
Profile
PW Security
Quota

Each database user account has:

- A unique username
- An authentication method
- A default tablespace
- A temporary tablespace
- A user profile
- A consumer group
- A lock status



6-3

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. Each database user has his or her own database account. This is Oracle's best practice recommendation to avoid potential security holes and provide meaningful data for certain audit activities. However, in rare cases, users share a common database account. In this case, operating system and applications must provide adequate security for the database. Each user account has:

- **A unique username:** Usernames cannot exceed 30 bytes, 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:** This is a place where the user creates objects if he or she does not specify some other tablespace. Note that having a default tablespace does not imply that the user has the *privilege* of creating objects in that tablespace, nor does the user have a *quota* of space within that tablespace in which to create objects. Both these are granted separately.

Database User Accounts (continued)

- **A temporary tablespace:** This is a place where the user can create temporary objects, such as sorts and temporary tables.
- **A user profile:** This is a set of resource and password restrictions assigned to the user.
- **A consumer group:** This is used by the resource manager.
- **A lock status:** Users can access only “unlocked” accounts.

Predefined Accounts: SYS and SYSTEM

- The SYS account:
  - Is granted the DBA role
  - Has all privileges with ADMIN OPTION
  - Is required for startup, shutdown, and some maintenance commands
  - Owns the data dictionary
  - Owns the Automatic Workload Repository (AWR)
- The SYSTEM account is granted the DBA role.
- These accounts are not used for routine operations.

6-5

Predefined Accounts: SYS and SYSTEM

The SYS and SYSTEM accounts have the database administrator (DBA) role granted to them by default.

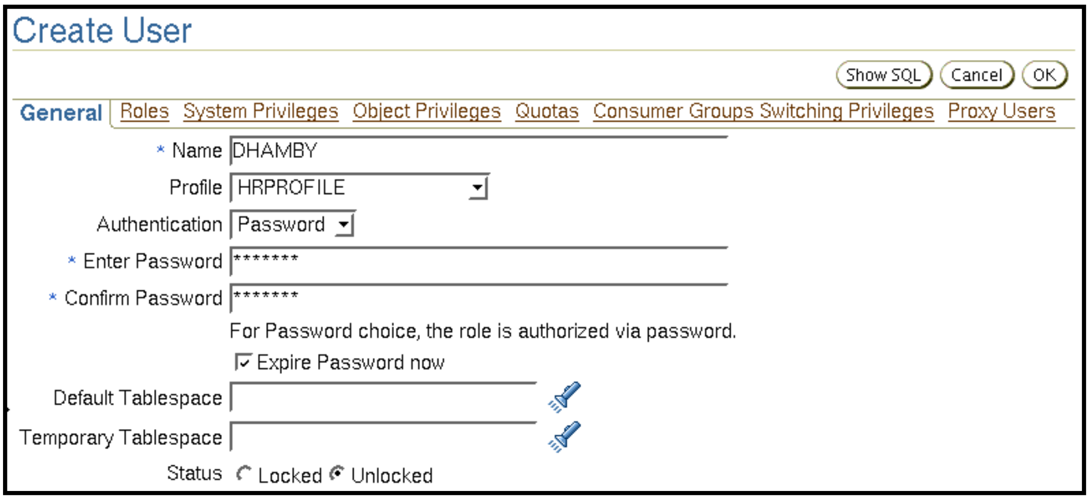
The SYS account in addition has all privileges with ADMIN OPTION and owns the data dictionary. To connect to the SYS account, you must use the AS SYSDBA clause. Any user that is granted the SYSDBA privilege can connect to the SYS account by using the AS SYSDBA clause. Only "privileged" users, who are granted the SYSDBA or SYSOPER privilege, are allowed to start up and shut down the database instance.

The SYSTEM account is granted the DBA role by default, but not the SYSDBA privilege.

Best practice tip: Applying the principle of least privilege, these accounts are not used for routine operations. Users who need DBA privileges have separate accounts with the required privileges granted to them. For example, Jim has a low privilege account called `jim` and a privileged account called `jim_dba`. This method allows the principle of least privilege to be applied, eliminates the need for account sharing, and allows individual actions to be audited.

The SYS and SYSTEM accounts are required accounts in the database. They cannot be dropped.

Creating a User



Select Administration > Schema > Users & Privileges > Users, and then click the Create button.

Creating a User

In Enterprise Manager, you can manage the list of database users, who are allowed to access the current database, by using the Users page. You can use this page to create, delete, and modify the settings of a user.

To create a database user, perform the following steps:

1. In Enterprise Manager Database Control, select Administration > Schema > Users & Privileges > Users.
2. Click the Create button.

Provide the required information. Mandatory items, such as Name, are marked with a star.

The following pages give you more information about authentication. Profiles are covered later in this lesson.

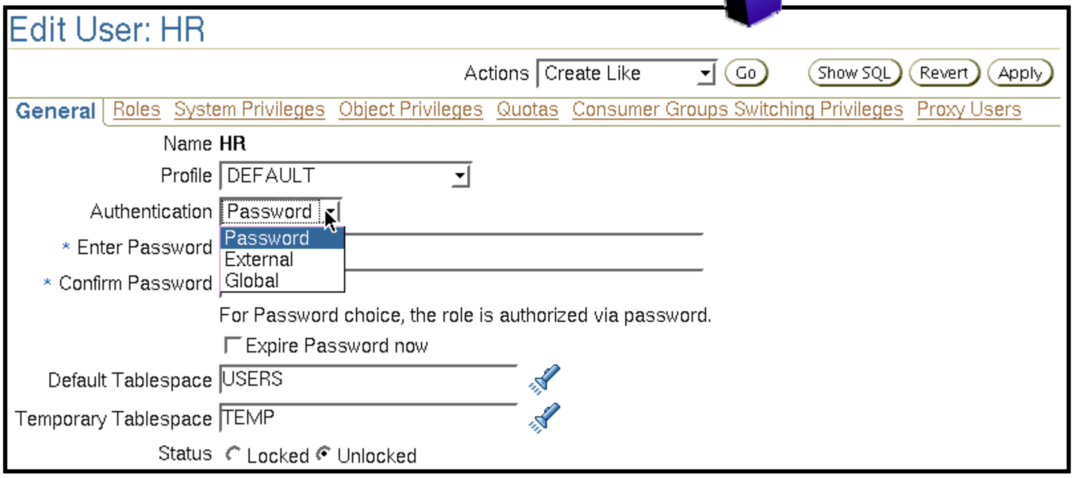
Assign a default tablespace and a temporary tablespace to each user. This allows you to control where their objects are created, if users do not specify a tablespace in the creation of an object.

If you do not choose a default tablespace, then the system-defined default permanent tablespace is used. Similarly for the temporary tablespace: if you do not specify one, then the system-defined temporary tablespace is used.

Authenticating Users

- Password
- External
- Global

User
> **Authentication**
Privilege
Role
Profile
PW Security
Quota



ORACLE

6-7

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: This is 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 decide on expiring user passwords, make sure that users have the ability to change the password. Some applications do not have this functionality.

Passwords are always automatically and transparently encrypted during network (client/server and server/server) connections, by using a modified Data Encryption Standard (DES) algorithm, before sending them across the network.

Authenticating Users (continued)

External: This is also referred to as authentication by the operating system. Users can connect to the Oracle database 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 the Oracle database adds to the beginning of each user's operating system account name. The default value of this parameter is `OPS$` for backward compatibility with the previous versions of the Oracle software. The Oracle database 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` needs to connect to an Oracle database and be authenticated by the operating system, then the Oracle database checks whether there is a corresponding database user `OPS$tsmith` and, if so, allows the user to connect. All references to a user who is authenticated by the operating system must include the prefix, as seen in `OPS$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: Using the Oracle Advanced Security option, global authentication (which is a strong authentication) allows users to be identified through the use of biometrics, x509 certificates, token devices, and Oracle Internet Directory. For more information about advanced authentication methods, refer to the *Oracle Enterprise Identity Management* course.

Administrator Authentication

Operating System Security

- **DBAs must have the OS privileges to create and delete files.**
- **Typical database users should not have the OS privileges to create or delete database files.**

Administrator Security

- **SYSBA and SYSOPER connections are authorized via password file or OS.**
  - **Password file authentication records the DBA user by name.**
  - **OS authentication does not record the specific user.**
  - **OS authentication takes precedence over password file authentication for SYSDBA and SYSOPER.**

6-9

Administrator Authentication

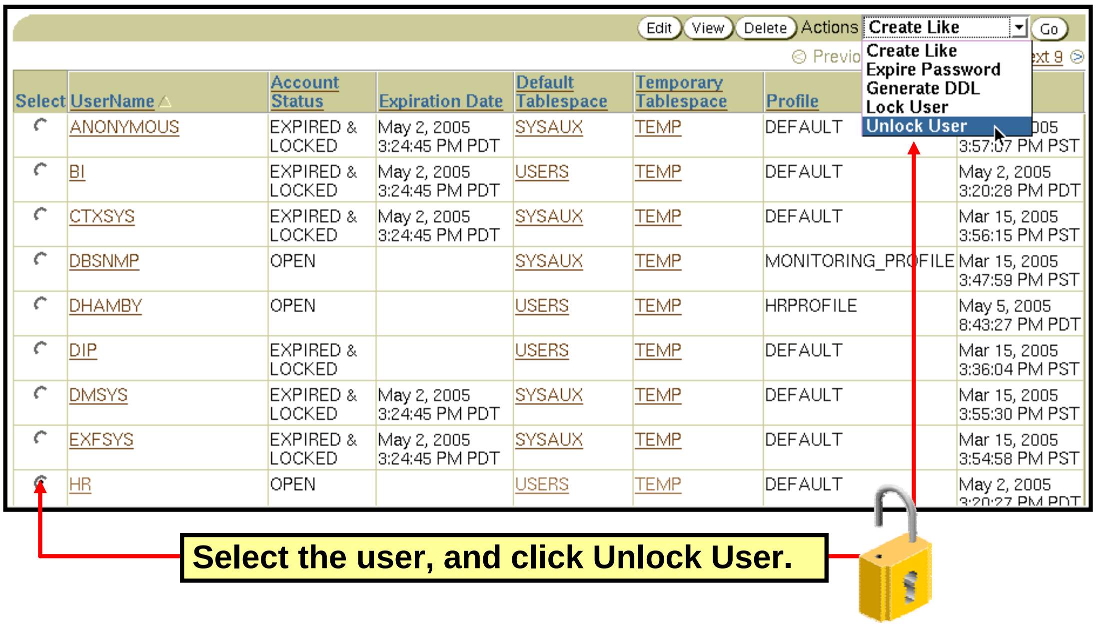
Operating System Security: In UNIX and Linux, by default, DBAs belong to the `install` OS group, which has the required privileges to create and delete database files.

Administrator Security: `SYSBA` and `SYSOPER` connections are authorized only after verification with the password file or with the operating system privileges and permissions. If operating system authentication is used, then the database does *not* use the supplied username and password. Operating system authentication is used if there is no password file, if the supplied username or password is not in that file, or if no username and password is supplied.

However, if authentication succeeds by means of the password file, then the connection is logged with the username. If authentication succeeds by means of the operating system, then it is a `CONNECT /` connection that does not record the specific user.

Note: OS authentication takes precedence over password file authentication. Specifically, if you are a member of the `OSDBA` or `OSOPER` group for the operating system, and you connect as `SYSDBA` or `SYSOPER`, you will be connected with the associated administrative privileges regardless of the username and password that you specify.

Unlocking a User Account and Resetting the Password



6-10

Unlocking a User Account and Resetting the Password

During installation and database creation, you can unlock and reset many of the Oracle-supplied database user accounts. If you have not chosen to unlock the user accounts at that time, you can unlock the users and reset the passwords by selecting the user on the Users page and clicking Unlock User.

Alternatively, if you are on the Edit Users page, perform the following steps:

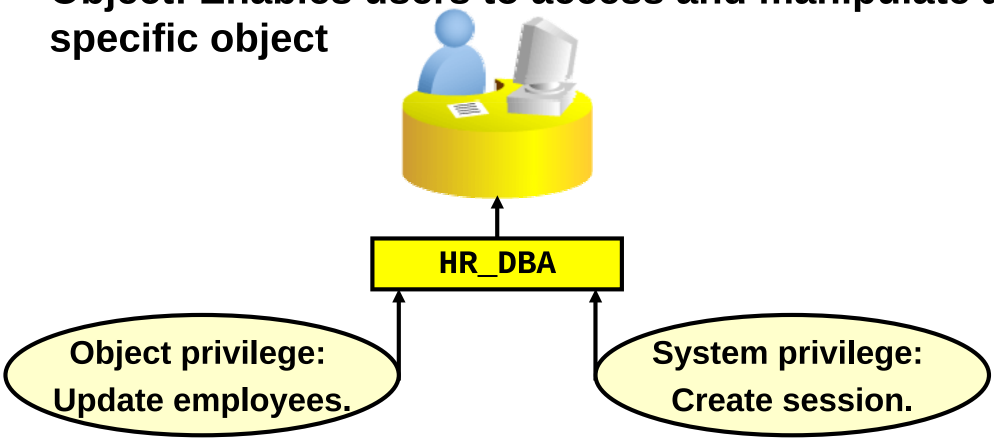
1. Enter the new password in the Enter Password and Confirm Password fields.
2. Select the Unlocked check box.
3. Click Apply to reset the password and unlock the user account.

Privileges

User Authentication
> Privilege
Role
Profile
PW Security
Quota

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



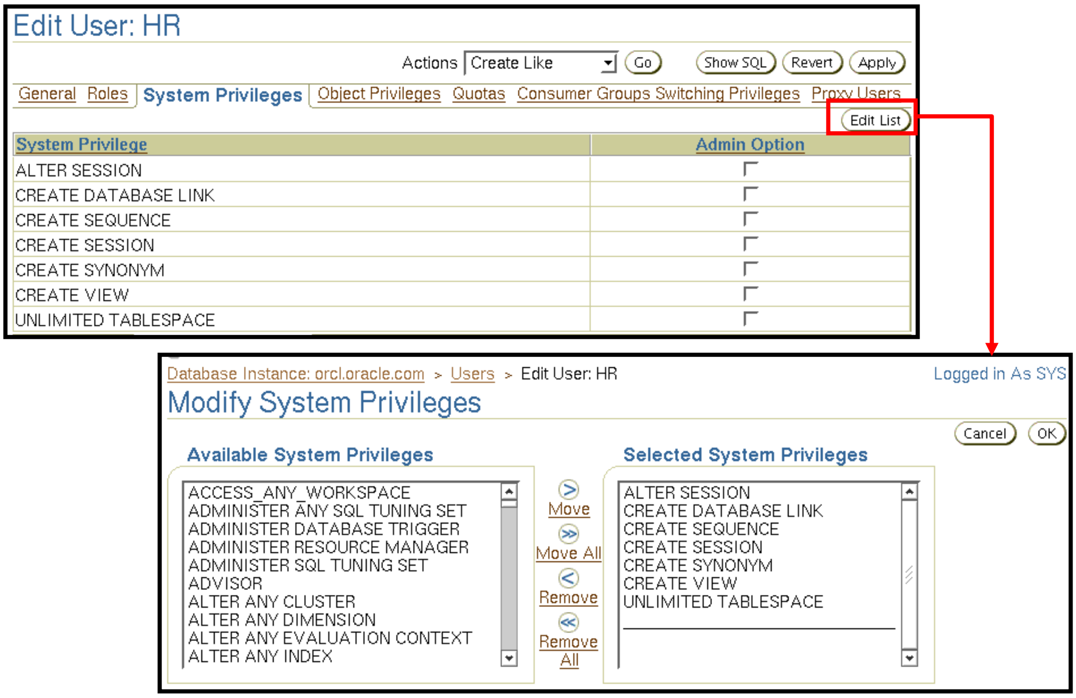
6-11

Privileges

A privilege is a right to execute a particular type of SQL statement or to access another user's object. The Oracle database enables you to control 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 more than a hundred distinct system privileges. Many system privileges contain the ANY clause.
- **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 has been explicitly given permission to grant privileges on the object.

System Privileges



System Privileges

To grant system privileges, click the Systems Privileges tab on the Edit User page. Select the appropriate privileges from the list of available privileges, and move them to the Selected System Privileges list by clicking the Move arrow.

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

Selecting the Admin Option check box enables you 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:

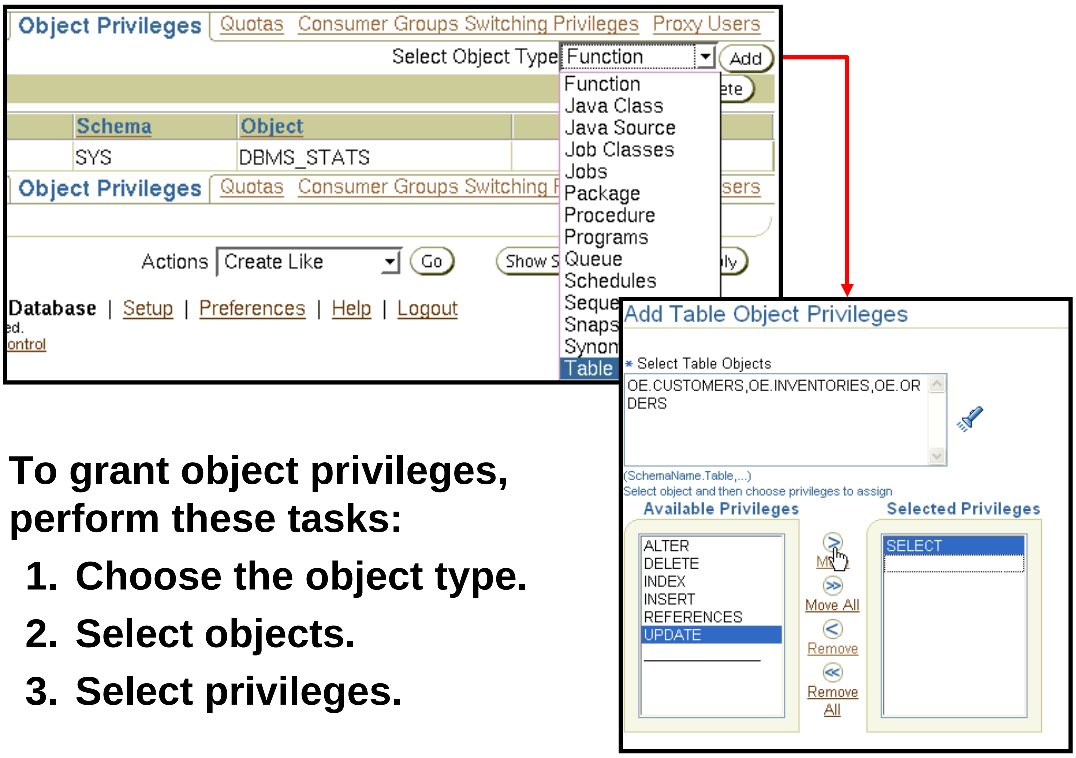
- **RESTRICTED SESSION:** This privilege allows you to log in even if the database has been opened in restricted mode.

System Privileges (continued)

- **SYSDBA and SYSOPER:** These privileges allow you to shut down, start up, and perform recovery and other administrative tasks in the database. SYSOPER allows a user to perform basic operational tasks, but without the ability to look at user data. It includes the following system privileges:
  - STARTUP and SHUTDOWN
  - CREATE SPFILE
  - ALTER DATABASE OPEN/MOUNT/BACKUP
  - ALTER DATABASE ARCHIVELOG
  - ALTER DATABASE RECOVER (Complete recovery only. Any form of incomplete recovery, such as UNTIL TIME|CHANGE|CANCEL|CONTROLFILE requires connecting as SYSDBA.)
  - RESTRICTED SESSION

  The SYSDBA system privilege additionally authorizes incomplete recovery and the deletion of a database. Effectively, the SYSDBA system privilege allows a user to connect as the SYS user.
- **DROP ANY *object*:** The DROP ANY privilege allows you to delete objects that other schema users own.
- **CREATE, MANAGE, DROP, and ALTER TABLESPACE:** These privileges allow for tablespace administration including creating, dropping, and changing their attributes.
- **CREATE ANY DIRECTORY:** The Oracle database 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, you can potentially call insecure code objects.
  The CREATE ANY DIRECTORY privilege allows a user to create a directory object (with read and write access) to any directory that the Oracle software owner can access. This means that the user can access external procedures in those directories. The user can attempt to directly read and write any database file, such as data files, redo log, and audit logs. Ensure that your organization has a security strategy that prevents misuse of powerful privileges such as this one.
- **GRANT ANY OBJECT PRIVILEGE:** This privilege allows you to grant object permissions on objects that you do not own.
- **ALTER DATABASE and ALTER SYSTEM:** These very powerful privileges allow you to modify the database and the Oracle instance, such as renaming a data file or flushing the buffer cache.

Object Privileges



To grant object privileges, perform these tasks:

1. **Choose the object type.**
2. **Select objects.**
3. **Select privileges.**

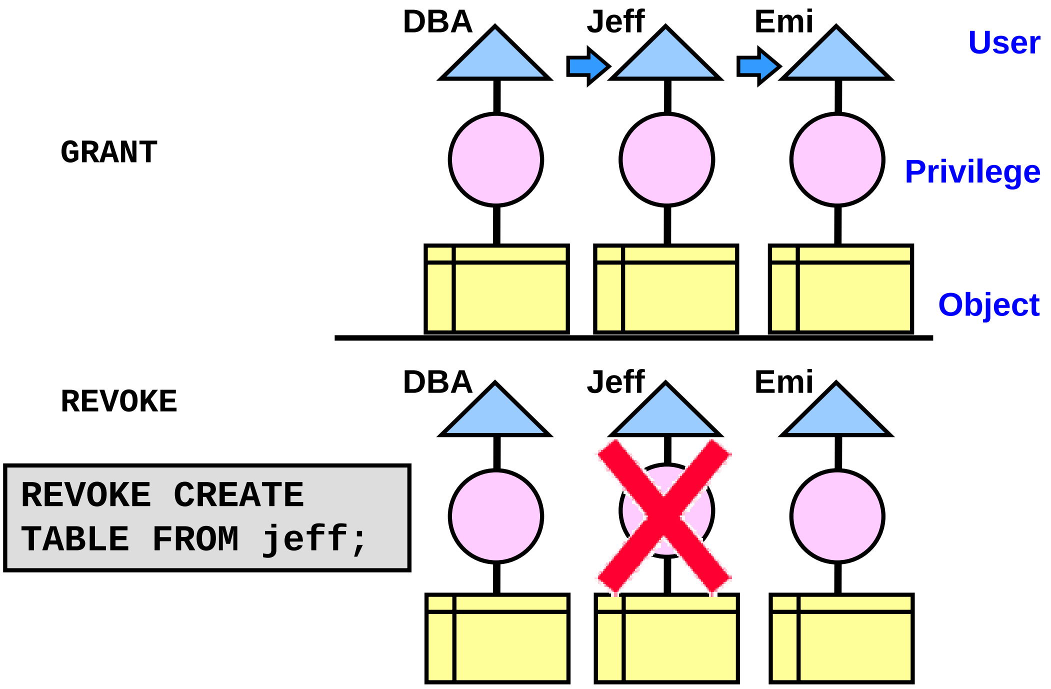
Object Privileges

To grant object privileges, click the Object Privileges tab on the Edit User page. 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 `<username.object name>` or selecting them from the list.

Next, select the appropriate privileges from the Available Privileges list, and click the Move button. When you have finished selecting privileges, click OK.

Back on the Edit User page, select the Grant check box if this user is allowed to grant other users the same access.

Revoking System Privileges with ADMIN OPTION



Revoking System Privileges

System privileges, which have been granted directly with a `GRANT` command, can be revoked by using the `REVOKE` SQL statement. Users with `ADMIN OPTION` for a system privilege can revoke the privilege from any other database user. The revoker does not have to be the same user who originally granted the privilege.

There are no cascading effects when a system privilege is revoked, regardless of whether it is given the `ADMIN OPTION`.

Read through the following steps that illustrate this:

Scenario

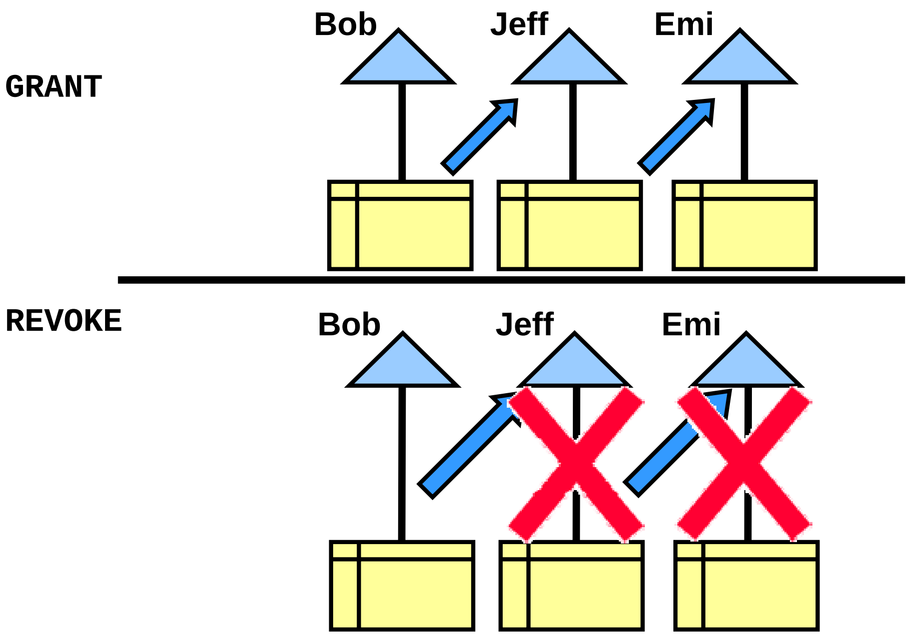
1. The DBA grants the `CREATE TABLE` system privilege to Jeff with `ADMIN OPTION`.
2. Jeff creates a table.
3. Jeff grants the `CREATE TABLE` system privilege to Emi.
4. Emi creates a table.
5. The DBA revokes the `CREATE TABLE` system privilege from Jeff.

The result

Jeff's table still exists, but no new tables can be created.

Emi's table still exists, and she still has the `CREATE TABLE` system privilege.

Revoking Object Privileges with GRANT OPTION



Revoking Object Privileges

Cascading effects can be observed when revoking a system privilege that is related to a data manipulation language (DML) operation. For example, if the `SELECT ANY TABLE` privilege is granted to a user, and that user has created procedures that use the table, all procedures that are contained in the user's schema must be recompiled before they can be used again.

Revoking object privileges also cascades when given `WITH GRANT OPTION`.

Read through the following steps that illustrate this:

Scenario

1. Jeff is granted the `SELECT` object privilege on `EMPLOYEES` with `GRANT OPTION`.
2. Jeff grants the `SELECT` privilege on `EMPLOYEES` to Emi.
3. Later, the `SELECT` privilege is revoked from Jeff. This revoke is cascaded to Emi as well.

Benefits of Roles

User
Authentication
Privilege
> Role
Profile
PW Security
Quota

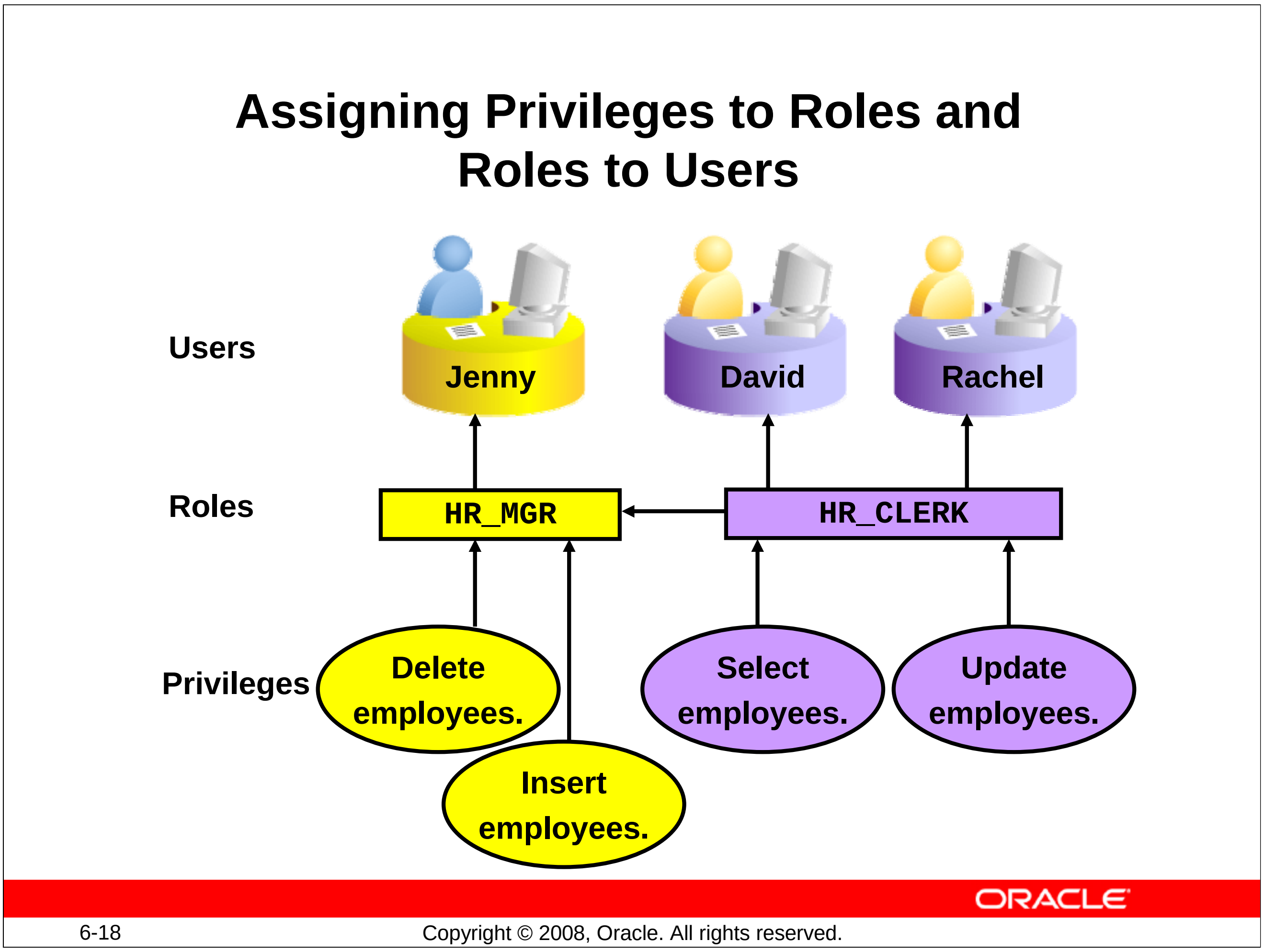
- Easier privilege management
- Dynamic privilege management
- Selective availability of privileges

ORACLE

6-17

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.



Assigning Privileges to Roles and Roles to Users

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. The Oracle software 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. Roles are designed to ease the administration of privileges in the database and, therefore, improve security.

Role characteristics

- Privileges are granted to and revoked from roles as though the role were a user.
- Roles can be granted to and revoked from users or other roles as though 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 slide example, the `HR_CLERK` role is granted the `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.

Predefined Roles

CONNECT	CREATE SESSION
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
SCHEDULER_ ADMIN	CREATE ANY JOB, CREATE EXTERNAL 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 HS_ADMIN_ROLE and over 1,700 object privileges on the data dictionary

ORACLE

6-19

Predefined Roles

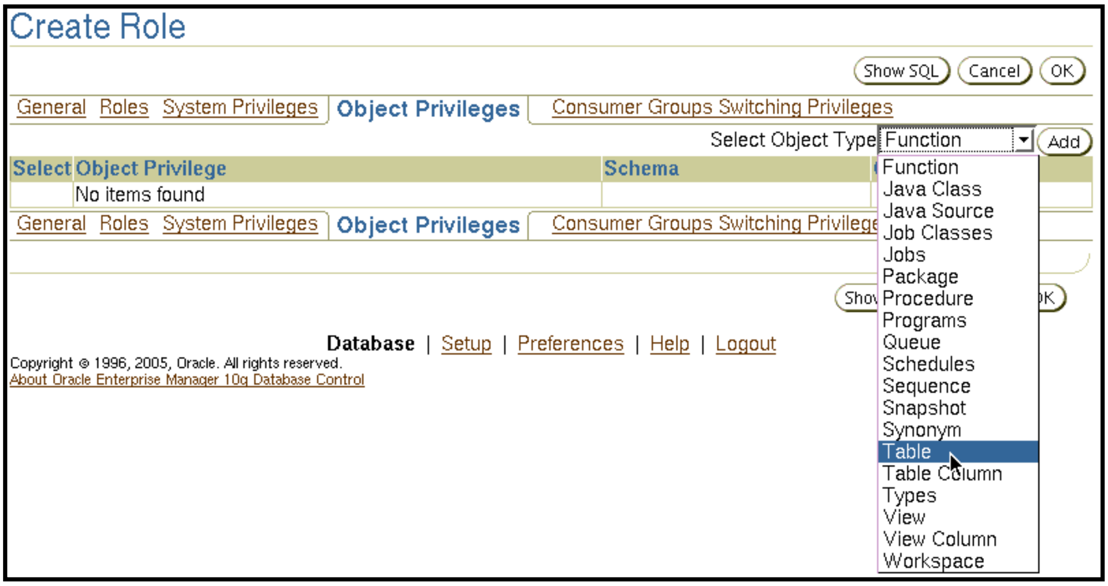
There are several roles that are defined automatically for Oracle databases when you run database creation scripts. `CONNECT` is granted automatically to any user created with Enterprise Manager. In earlier versions of the database (before Oracle Database 10*g* Release 2), the `CONNECT` role included more privileges, such as `CREATE TABLE` and `CREATE DATABASE LINK`, which have been removed for security reasons.

Note: Be aware that granting the `RESOURCE` role includes granting the `UNLIMITED TABLESPACE` privilege.

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 Extensible Markup Language (XML) database if that feature is installed. `AQ_ADMINISTRATOR_ROLE` provides privileges to administer advanced queuing. `HS_ADMIN_ROLE` includes the privileges needed to administer heterogeneous services. You must not alter the privileges granted to these functional roles without the assistance of Oracle support because you may inadvertently disable the needed functionality.

Creating a Role



Select Administration > Schema > Users & Privileges > Roles.

Creating a Role

A **role** is a named group of related privileges that are granted to users or to other roles. A DBA manages privileges through roles.

To create a role, perform the following steps:

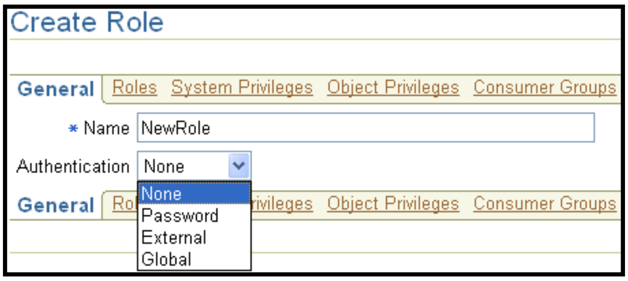
1. In Enterprise Manager Database Control, select Administration > Schema > Users & Privileges > Roles.
2. Click the Create button.

Secure Roles

- Roles may be nondefault.

```
SET ROLE vacationdba;
```

- Roles may be protected through authentication.



- Roles may also be secured programmatically.

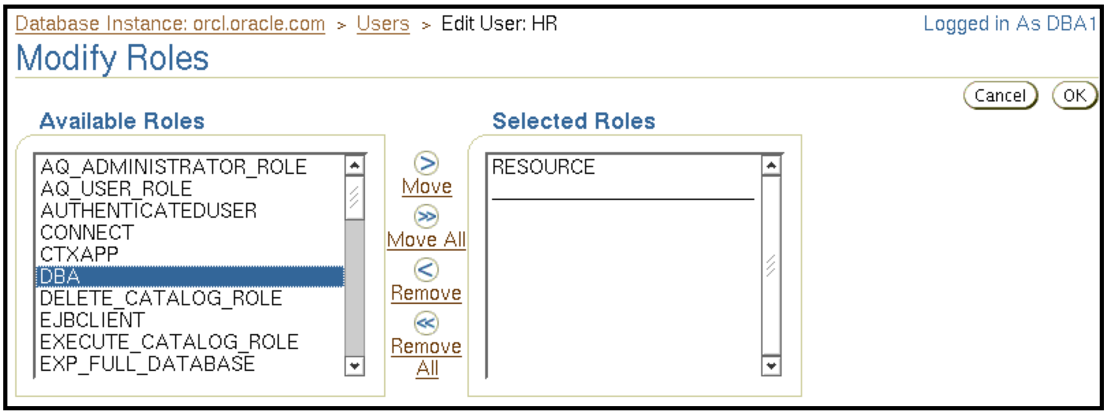
```
CREATE ROLE secure_application_role
IDENTIFIED USING <security_procedure_name>;
```


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, deselect the DEFAULT check box. The user must now 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 other elements needed to properly secure a group of permissions.

Assigning Roles to Users



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 schemas.

To assign a role to a user, perform the following steps:

1. In Enterprise Manager Database Control, choose Administration > Schema > Users & Privileges > Users.
2. Select the user, and click the Edit button.
3. Click the Roles tab, and then click the Edit List button.
4. Select the desired role under Available Roles and move it under Selected Roles.
5. When you have assigned all appropriate roles, click the OK button.

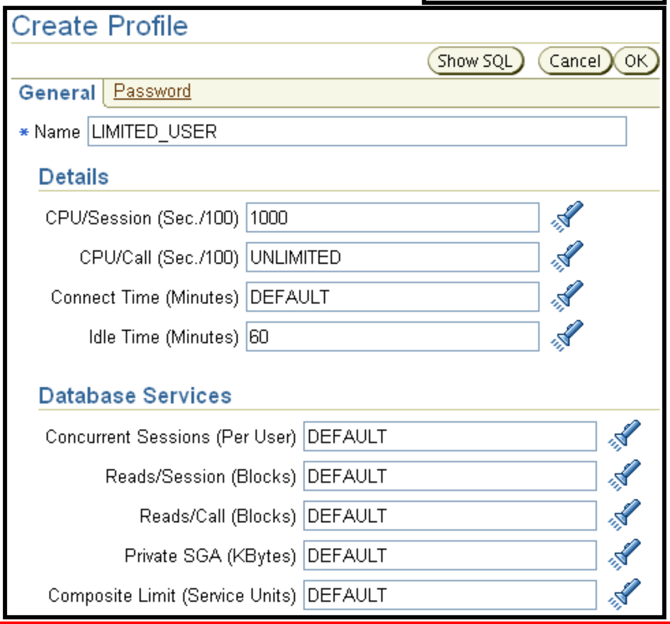
Profiles and Users

User
Authentication
Privilege
Role
> **Profile**
PW Security
Quota

Users are assigned only one profile at any given time.

Profiles:

- **Control resource consumption**
- **Manage account status and password expiration**



Profiles and Users

Profiles impose a named set of resource limits on database usage and instance resources. Profiles also manage the account status and place limitations on users' passwords (length, expiration time, and so on). Every user is assigned a profile and may belong to only one profile at any given time. If users have already logged in when you change their profile, the change does not take effect until their next login.

The default profile serves as the basis for all other profiles. As illustrated in the slide, limitations for a profile can be implicitly specified (as in CPU/Session), be 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 enable 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 1,000 means that if any individual session that uses this profile consumes more than 10 seconds of CPU time (CPU time limitations are in hundredths 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 who is not local to the server) network resources. You can specify the following:
  - Connect Time: Indicates for how many minutes a user can be connected before being automatically logged off
  - Idle Time: Indicates for 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: Indicates how many concurrent sessions can be created by using a database user account
  - Private SGA: Limits the amount of space consumed within the System Global Area (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 the lesson titled "Configuring the Oracle Network Environment").
- **Disk I/O:** This 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 the disk. This can be done to ensure that no input/output (I/O)-intensive statements overuse memory and disks.

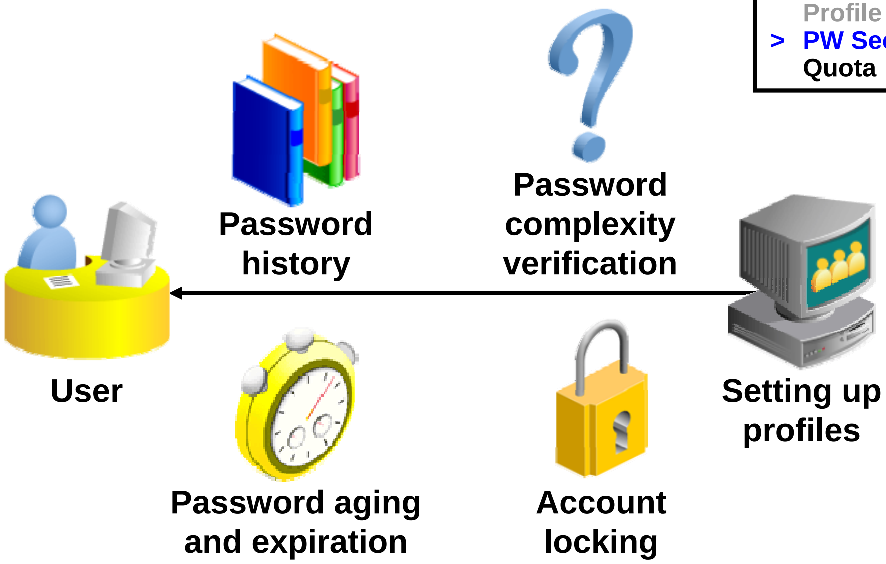
Profiles also allow a composite limit. Composite limits are based on a weighted combination of CPU/Session, Reads/Session, Connect Time, and Private SGA. Composite limits are discussed in more detail in the *Oracle Database Security Guide*.

To create a profile, select Administration > Schema > Users & Privileges > Profiles, and click the Create button.

Note: Resource Manager is an alternative to many of the profile settings. For more details about Resource Manager, see the *Oracle Database Administrator's Guide*.

Implementing Password Security Features

User
Authentication
Privilege
Role
Profile
> **PW Security**
Quota



Note: Do not use profiles that cause the passwords for SYS, SYSMAN, and DBSNMP to expire and, subsequently, cause those accounts to get locked.

Implementing Password Security Features

Oracle password management is implemented with user profiles. Profiles can provide many standard security features including the following:

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.

- The `FAILED_LOGIN_ATTEMPTS` parameter specifies the number of failed login attempts before the lockout of the account.
- The `PASSWORD_LOCK_TIME` parameter specifies the number of days for which the account is locked after the specified number of failed login attempts.

Password aging and expiration: Enables user passwords to have a lifetime, after which the passwords expire and must be changed

- The `PASSWORD_LIFE_TIME` parameter determines the lifetime of the password in days, after which the password expires.
- The `PASSWORD_GRACE_TIME` parameter specifies a grace period in days for changing the password after the first successful login after the password has expired.

Note: Expiring passwords and locking the `SYS`, `SYSMAN`, and `DBSNMP` accounts prevent Enterprise Manager from functioning properly. The applications must catch the “password expired” warning message and handle the password change; otherwise, the grace period expires and the user is locked out without knowing the reason.

Implementing Password Security Features (continued)

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. These checks can be implemented by using one of the following:

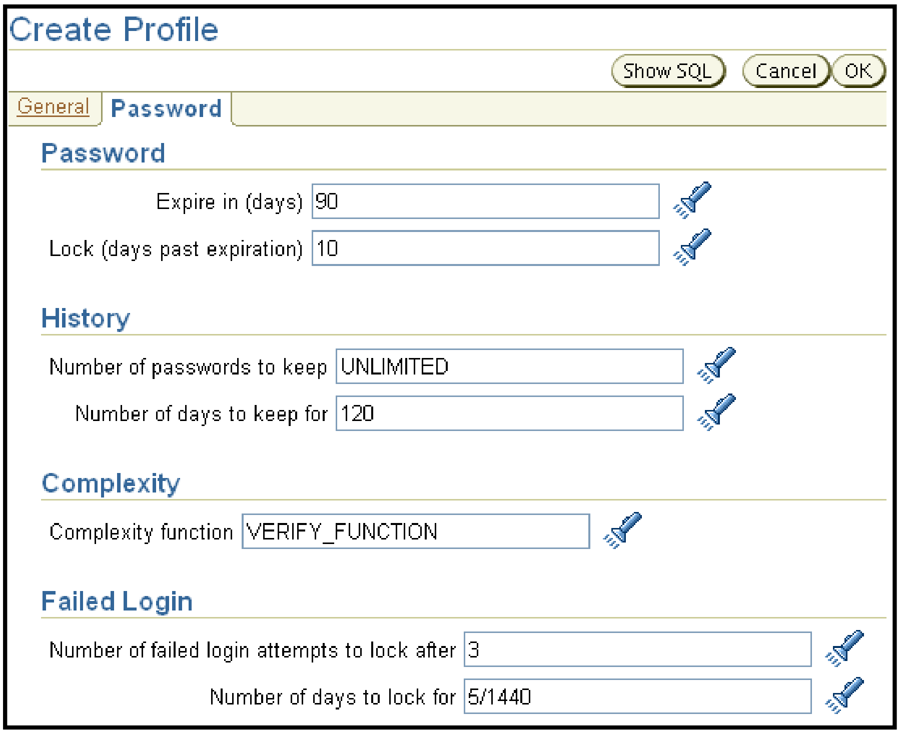
- `PASSWORD_REUSE_TIME`: Specifies that a user cannot reuse a password for a given number of days
- `PASSWORD_REUSE_MAX`: Specifies the number of password changes that are required before the current password can be reused

These two parameters are mutually exclusive, and 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 complexity verification: Makes a complexity check on the password to verify that it meets certain rules. The check must ensure that the password is complex enough to provide protection against intruders who may try to break into the system by guessing the password.

The `PASSWORD_VERIFY_FUNCTION` parameter names a PL/SQL function that performs a password complexity check before a password is assigned. Password verification functions must be owned by the `SYS` user and must return a Boolean value (`TRUE` or `FALSE`).

Creating a Password Profile



6-27

Creating a Password Profile

To create a password profile, select Administration > Schema > Users & Privileges > 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 also. There are 1,440 minutes in a day, and so 5/1440 is five minutes.

Enterprise Manager can also be used to edit existing password profiles.

Dropping a Password Profile

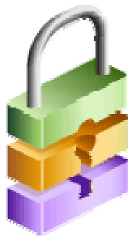
In Enterprise Manager, you cannot drop a profile that is used by users. However, if you drop a profile with the `CASCADE` option (for example, in SQL*Plus), then all users who have that profile are automatically assigned the `DEFAULT` profile.

Supplied Password Verification Function: VERIFY_FUNCTION

The supplied password verification function enforces these password restrictions:

- **The minimum length is four characters.**
- **The password cannot be the same as the username.**
- **The password must have at least one alphabetic, one numeric, and one special character.**
- **The password must differ from the previous password by at least three letters.**

Tip: Use this function as a template to create your own customized password verification.



ORACLE

6-28

Supplied Password Verification Function: VERIFY_FUNCTION

The Oracle server provides a password complexity verification function named VERIFY_FUNCTION. This function is created with the <oracle_home>/rdbms/admin/utlpwdmg.sql script. The password complexity verification function must be created in the SYS schema. It can be used as a template for your customized password verification.

In addition to creating 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;
```

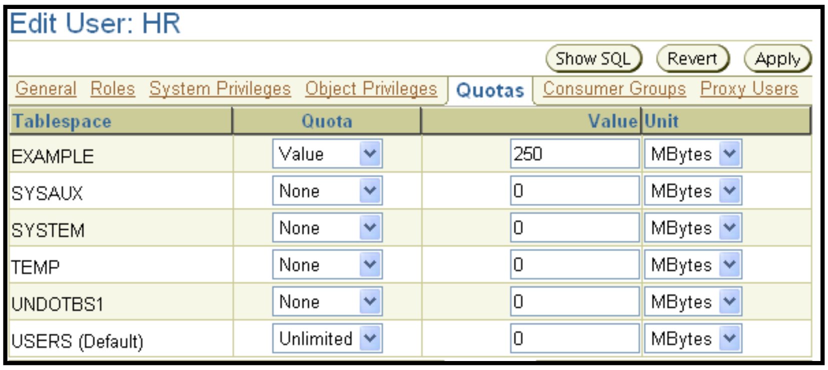
Remember that when users are created, they are assigned the DEFAULT profile, unless another profile is specified.

Assigning Quota to Users

User
Authentication
Privilege
Role
Profile
PW Security
> **Quota**

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:

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



Tablespace	Quota	Value	Unit
EXAMPLE	Value	250	MBytes
SYSAUX	None	0	MBytes
SYSTEM	None	0	MBytes
TEMP	None	0	MBytes
UNDOTBS1	None	0	MBytes
USERS (Default)	Unlimited	0	MBytes

ORACLE

6-29

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 does not guarantee that the space is set aside for the user. This value can be larger or smaller than the current space that is 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 must be granted with caution.
  Note: Be aware that granting the `RESOURCE` role includes granting this privilege.

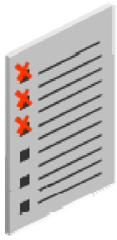
You must not provide quota to users on the `SYSTEM` or `SYSAUX` tablespace. Typically, only the `SYS` and `SYSTEM` users must be able to create objects in the `SYSTEM` or `SYSAUX` tablespace.

You do not need quota on an assigned temporary tablespace or any undo tablespaces.

Assigning Quota to Users (continued)

- When does the Oracle instance use quota?
  Quotas are used when a user creates or extends a segment.
- Which activities do not count against the quota?
  Activities that do not use space in the assigned tablespace do not affect the quota, such as creating views or using temporary tablespace.
- When is the quota replenished?
  The quota is replenished when objects owned by the user are dropped with the `PURGE` clause or the objects in the recycle bin are automatically purged.

Summary



In this lesson, you should have learned how to:

- Create and manage database user accounts
  - Authenticate users
  - Assign default storage areas (tablespaces)
- Grant and revoke privileges
- Create and manage roles
- Create and manage profiles
  - Implement standard password security features
  - Control resource usage by users

Practice Overview: Administering Users

This practice covers the following topics:

- Creating a profile to limit resource consumption
- Creating two roles:
  - `HRCLERK`
  - `HRMANAGER`
- Creating four new users:
  - One manager and two clerks
  - One schema user for the next practice session

Managing Schema Objects

Objectives

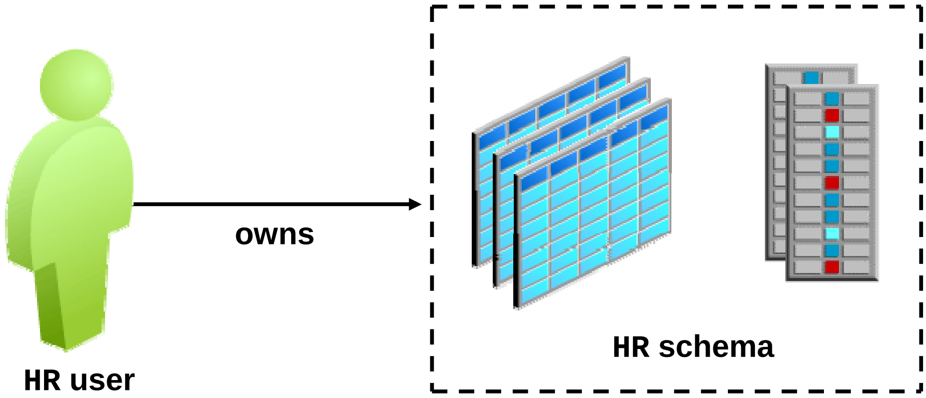
After completing this lesson, you should be able to do the following:

- **Define schema objects and data types**
- **Create and modify tables**
- **Define constraints**
- **View the columns and contents of a table**
- **Create indexes**
- **Create views**
- **Create sequences**
- **Explain the use of temporary tables**
- **Use the data dictionary**

7-2

What Is a Schema?

> **Schema**
Constraints
Indexes
Views
Sequences
Temp Tables
Data Dict



ORACLE

7-3

What Is a Schema?

A schema is a collection of database objects that are owned by a particular user. Typically, for a production database, this user does not represent a person, but an application. A schema has the same name as the user that owns the schema. Schema objects are the logical structures that directly refer to database's data. Schema objects include structures such as tables, views, and indexes.

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

Note: A schema does not necessarily have to be directly related to a single tablespace. You can define configurations such that objects in a single schema can be in different tablespaces, and a tablespace can hold objects from different schemas.

What Is a Schema? (continued)

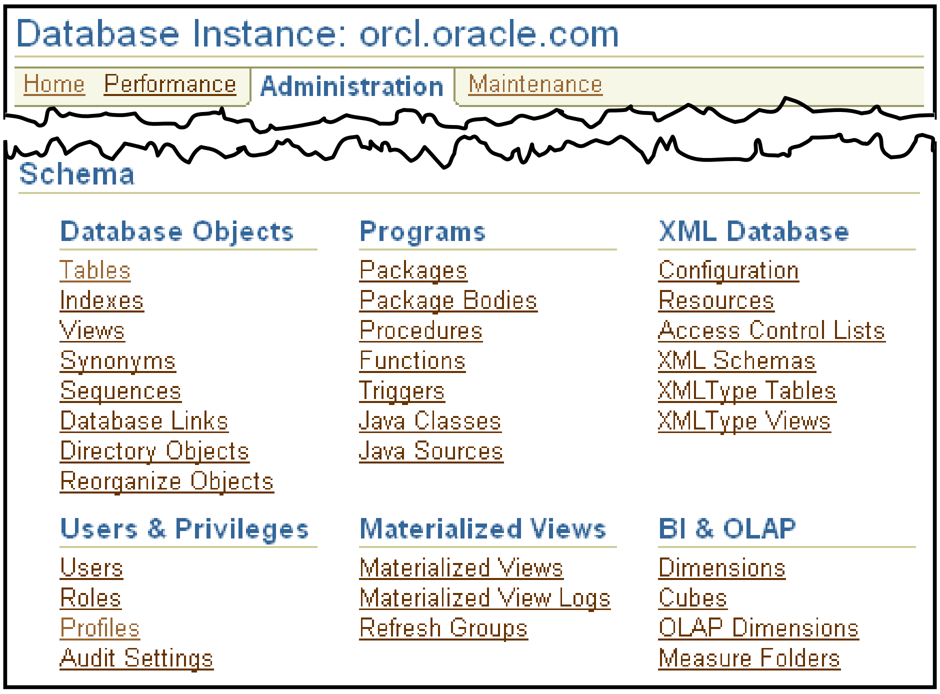
When you create the database, several schemas are created for you. Two of particular importance are the following:

- **SYS schema:** This contains the data dictionary, as described in the lesson titled "Administering User Security."
- **SYSTEM schema:** It contains additional tables and views that store administrative information. This is described in the lesson titled "Administering User Security."

During a complete installation of an Oracle database, sample schemas are installed automatically. Sample schemas serve the purpose of providing a common platform for examples in Oracle documentation and curricula. They are a set of interlinked schemas aimed at providing examples of different levels of 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 deals with matters of intermediate complexity. A multitude of data types are 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 allows demonstrations with larger amounts of data. An extension to this schema provides support for advanced analytic processing.

Accessing Schema Objects



7-5

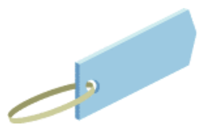
Accessing Schema Objects

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 list. The drop-down list includes additional object types that are not shown as links on the Database Administration page.

Naming Database Objects

- **The length of names must be from 1 to 30 bytes, 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.**
- **Quoted names are not recommended.**



ORACLE

7-6

Naming Database Objects

When you name an object in the database, you can enclose the name in double quotation marks (“”). If you do this, 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 quotation marks 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 enter the name in the wrong case, then you get an error:

```
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 (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 punctuation marks as well as spaces. However, neither quoted nor nonquoted identifiers can contain double quotation marks.

Specifying Data Types in Tables

Common data types:

- **`CHAR(`*`size`*` [BYTE|CHAR])`: Fixed-length character data of *`size`* bytes or characters**
- **`VARCHAR2(`*`size`*` [BYTE|CHAR])`: Variable-length character string having a maximum length of *`size`* bytes or characters**
- **`DATE`: Valid date ranging from January 1, 4712 B.C. through A.D. December 31, 9999**
- **`NUMBER(`*`p,s`*`)`: Number with 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.

Built-in data types in the Oracle database include the following:

- **`CHAR`:** Fixed-length character data of *`size`* bytes or characters. The maximum size is 2,000 bytes or characters; the default and minimum size is 1 byte.
  - `BYTE` indicates that the column has byte length semantics.
  - `CHAR` indicates that the column has character semantics.
- **`VARCHAR2`:** Variable-length character string having maximum length *`size`* bytes or characters. The maximum size is 4,000 bytes. You must specify the size for `VARCHAR2`.
- **`DATE`:** Valid date ranging from January 1, 4712 B.C. through A.D. December 31, 9999. It also stores the time: hours, minutes, and seconds.
- **`NUMBER`:** Number with precision *p* and scale *s*. The precision can range from 1 through 38. The scale can range from –84 through 127.

Specifying Data Types in Tables (continued)

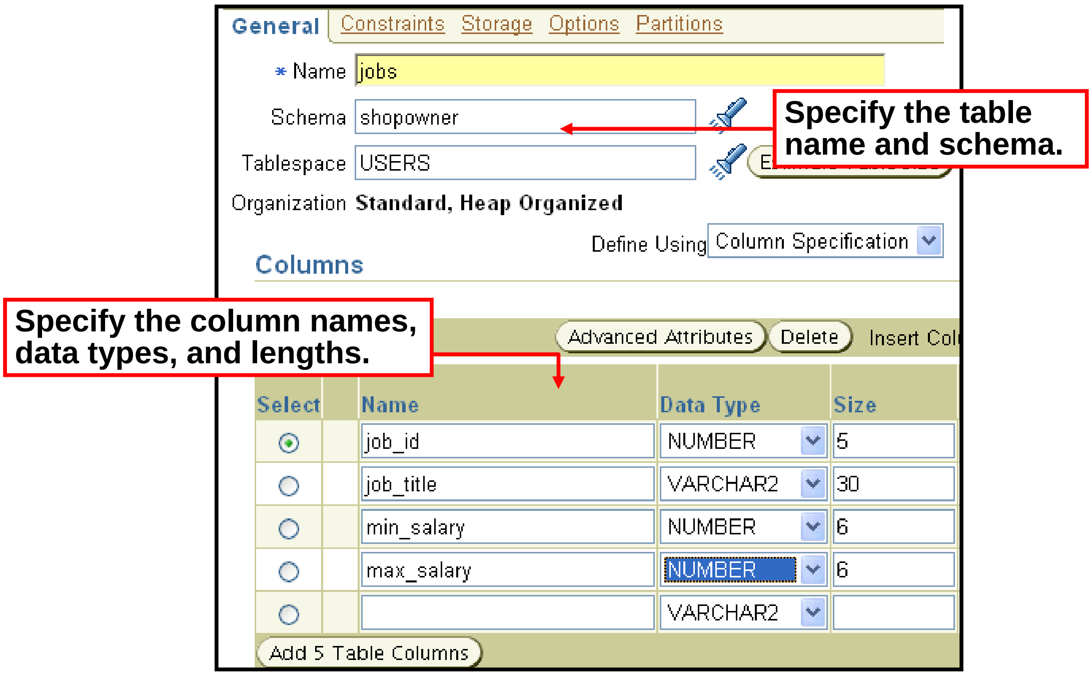
- **BINARY_FLOAT:** This is a 32-bit floating-point number. This data type requires 5 bytes, including the length byte.
- **BINARY_DOUBLE:** This is a 64-bit floating-point number. This data type requires 9 bytes.
- **FLOAT(*p*):** This is an American National Standards Institute (ANSI) data type. The 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. The width specifications of the NCHAR data type refer to the number of characters. The maximum column size allowed is 2,000 bytes. If you insert a value that is less than the column length, the Oracle database pads the value with blanks for full column length. You cannot insert a CHAR value into an NCHAR column, nor can you insert an NCHAR value into a CHAR column.
- **NVARCHAR2(size [BYTE|CHAR]):** The NVARCHAR2 data type is a Unicode-only data type. It is like NCHAR except that its maximum length is 4,000 bytes and it is not blank-padded.
- **LONG:** This is a character data of variable length of up to 2 gigabytes or 2^{31} –1 bytes. The LONG data type is deprecated; use the large object (LOB) data type instead.
- **LONG RAW:** This is raw binary data of variable length of up to 2 gigabytes.
- **RAW(*size*):** This is raw binary data of length *size* bytes. The maximum size is 2,000 bytes. You must specify the size for a RAW value.
- **ROWID:** This is a base 64 string representing the unique address of a row in the database. This data type is primarily for values returned by the ROWID pseudocolumn.
- **UROWID:** This is a 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 the UROWID type. The maximum size and default is 4,000 bytes.
- **BLOB:** This is a binary large object.
- **CLOB:** This is 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.

Specifying Data Types in Tables (continued)

- **NCLOB:** This is 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. It stores national character set data.
  Note: The maximum size for all LOB data types (BLOB, CLOB, and NCLOB) is:
  (4 gigabytes – 1) * (the value of CHUNK).
  CHUNK is an optional attribute that you can set when defining a LOB. CHUNK specifies the number of bytes to be allocated for LOB manipulation. If the size is not a multiple of the database block size, then the database rounds up the size in bytes to the next multiple. For example, if the database block size is 2,048 and the CHUNK size is 2,050, then the database allocates 4,096 bytes (2 blocks). The maximum value is 32,768 (32 KB), which is the largest Oracle database block size allowed. The default CHUNK size is one Oracle database block.
- **BFILE:** The BFILE data type contains a locator to a large binary file stored outside the database. It enables byte stream I/O access to external LOBs residing on the database server. The maximum size is 4 gigabytes.
- **TIMESTAMP(*fractional_seconds_precision*):** With this data type, you can specify the 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 a second. The accepted values are 0 to 9. The default is 6.

For a complete list of built-in data types and user-defined types, refer to *Oracle Database SQL Reference*.

Creating and Modifying Tables



7-11

Creating and Modifying Tables

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

Creating a Table

To create a table by using Enterprise Manager, perform the following steps:

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 browse through the schema names and select the one that 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 region, enter the column name and data types.
9. Click OK. An update message appears indicating that the table has been successfully created.

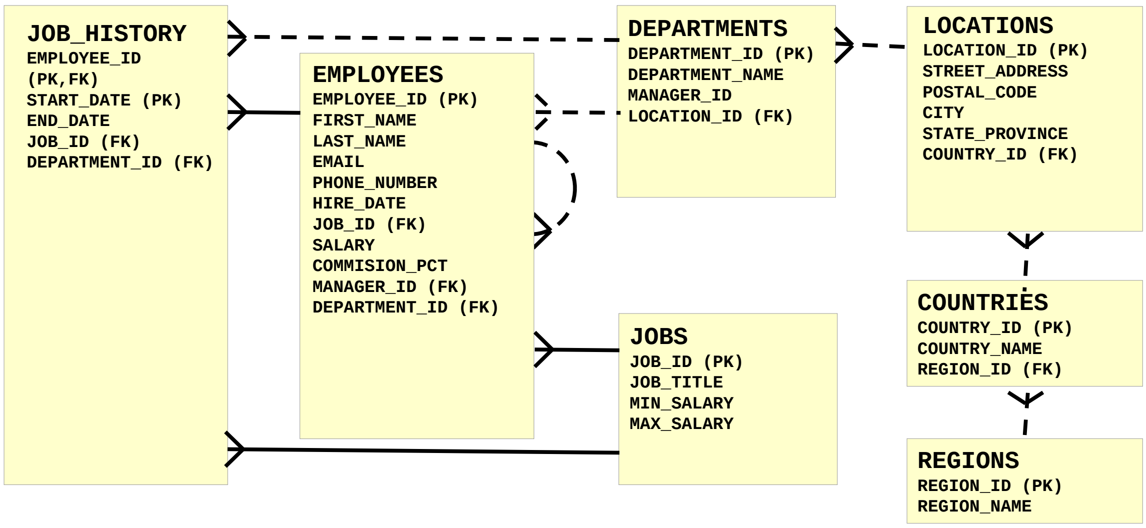
Modifying a Table

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

1. On the Tables page, select the table in the results list and click Edit.
2. On 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.
4. Click Apply. An update message appears indicating that the table has been modified successfully.

Understanding Data Integrity

Schema
> Constraints
Indexes
Views
Sequences
Temp Tables
Data Dict



ORACLE

7-13

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 must 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. Except for special circumstances, this is enforced with a unique index.
- **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 server's implementation of the `PRIMARY KEY` integrity constraint guarantees that both 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.

Under normal circumstances, the database enforces the `PRIMARY KEY` constraints by 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.

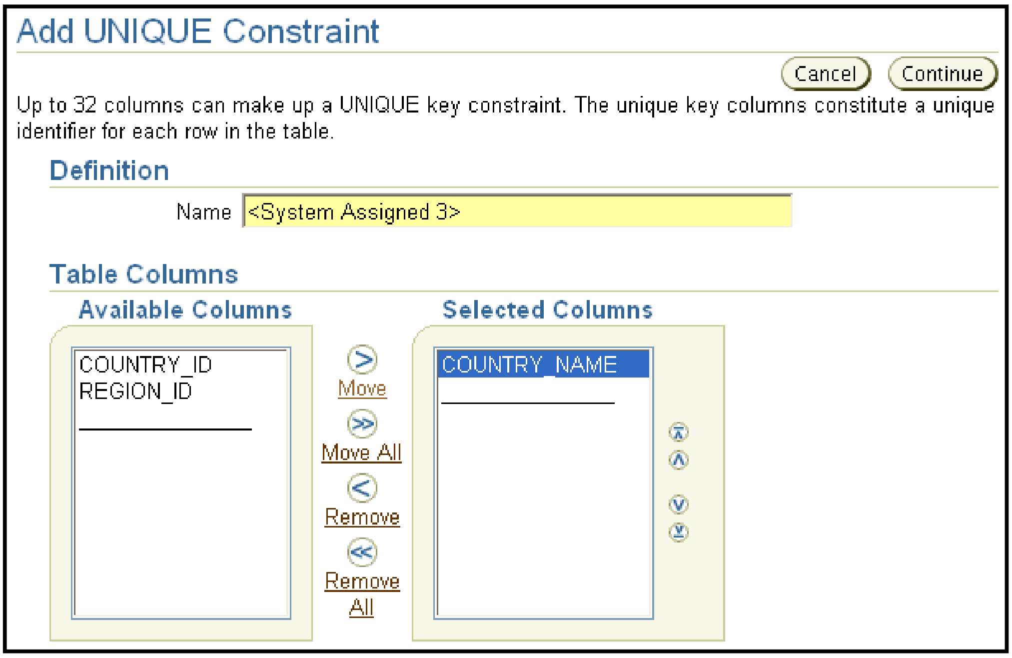
A referential integrity constraint requires that for each row of a table, the value in the foreign key must match a value in a parent key.

As an example, a foreign key 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. Therefore, no erroneous department numbers can exist in the `DEPARTMENT_ID` column of the `EMPLOYEES` 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 data manipulation language (DML) statement results in the condition of the `CHECK` constraint evaluating to false, then the statement is rolled back.

Defining Constraints



Defining Constraints

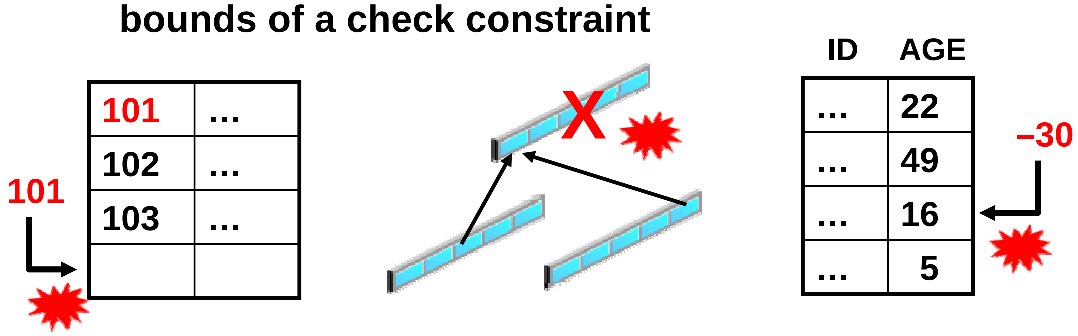
To add a constraint to a table by using Enterprise Manager, perform the following steps:

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 that you want to add from the drop-down list, and click Add.
4. Enter the appropriate information for the type of constraint that you are defining. Click OK.

Constraint Violations

Examples of how a constraint can be violated are:

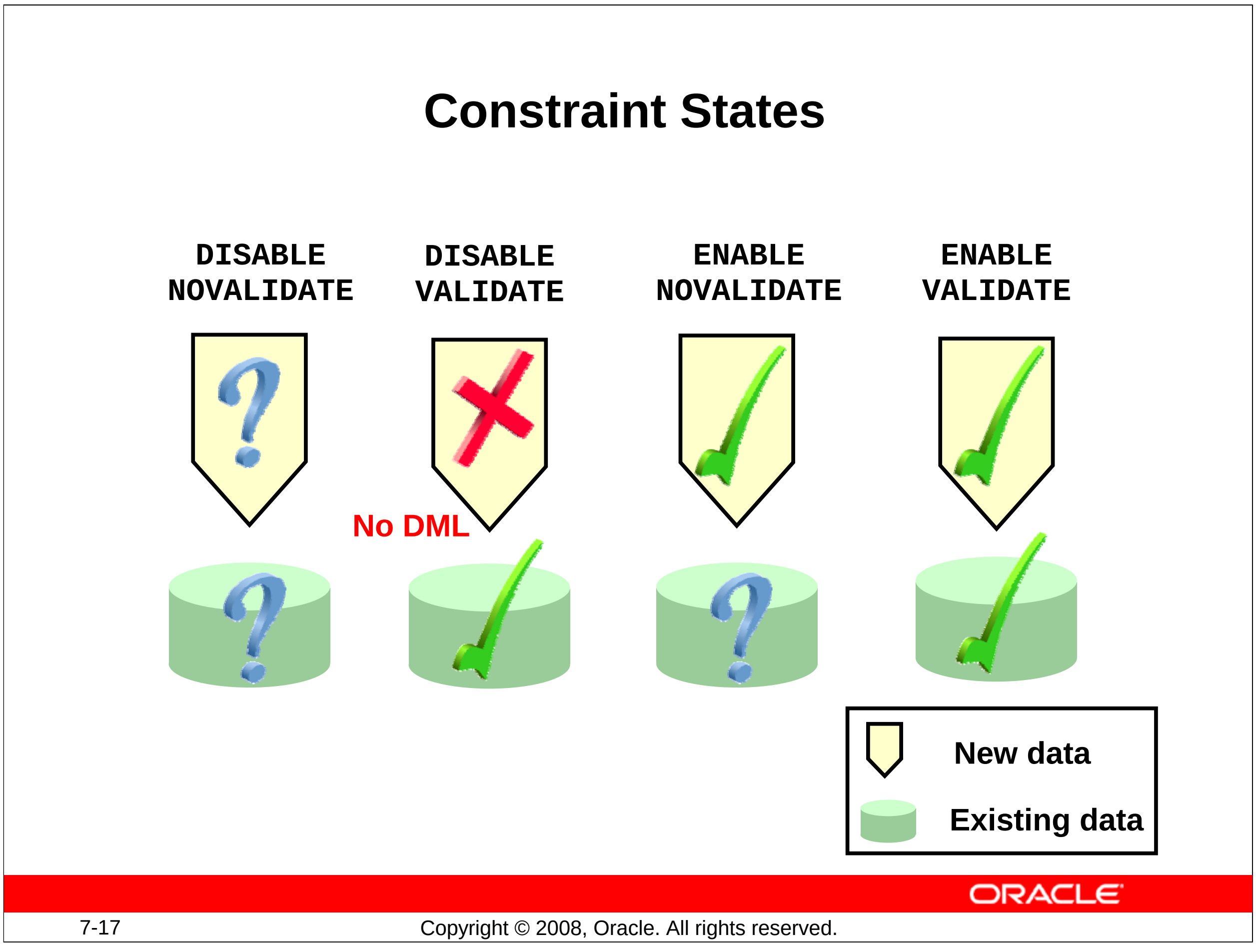
- **Inserting a duplicate primary key value**
- **Deleting the parent of a child row in a referential integrity constraint**
- **Updating a column to a value that is out of the bounds of a check constraint**



Constraint Violations

A constraint violation occurs when DML is submitted, which does not comply with the constraint. Constraint violations can come in many forms. Among these are the following:

- **Uniqueness:** An attempt is made to have duplicate values in a column that has a unique constraint, such as in the case where a column is the primary key, or it is uniquely indexed.
- **Referential integrity:** The rule of every child row having a parent row is violated.
- **Check:** An attempt is made to store a value in a column that does not follow the rules defined for that column. For example, an AGE column could have a check constraint on it enforcing it to be a positive number.



Constraint States

To better deal with situations where data must be temporarily in violation of a constraint, you can designate a constraint to be in various states. An integrity constraint can be enabled (`ENABLE`) or disabled (`DISABLE`). If a constraint is enabled, the data is checked as it is entered or updated in the database. Data that does not conform to the constraint's rule is prevented from being entered. If a constraint is disabled, then the nonconforming data can be entered into the database. An integrity constraint can be in one of the following states:

- `DISABLE NOVALIDATE`
- `DISABLE VALIDATE`
- `ENABLE NOVALIDATE`
- `ENABLE VALIDATE`

Constraint States (continued)

- **`DISABLE NOVALIDATE`:** New as well as existing data may not conform to the constraint because it is not checked. This is often used when the data is from an already validated source and the table is read-only, so no new data is being entered into the table.
- **`DISABLE VALIDATE`:** If a constraint is in this state, then any modification of the constrained columns is not allowed because it would be inconsistent to have validated the existing data and then allow unchecked data to enter the table. This is often used when the existing data must be validated but the data is not going to be modified and the index is not otherwise needed for performance.
- **`ENABLE NOVALIDATE`:** New data conforms to the constraint but existing data is in an unknown state. This is frequently used so that existing constraint violations can be corrected, and at the same time, new violations are not allowed to enter the system.
- **`ENABLE VALIDATE`:** Both new and existing data conform to the constraint. This is the typical and default state of a constraint.

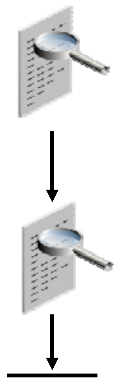
Constraint Checking

Constraints are checked at the time of:

- Statement execution, for *nondeferred* constraints
- `COMMIT`, for *deferred* constraints

Case: DML statement, followed by `COMMIT`

1. Nondeferred constraints checked
2. `COMMIT` issued
3. Deferred constraints checked
4. `COMMIT` complete



Constraint Checking

You can defer checking constraints for validity until the end of the transaction.

Nondeferred constraints, also known as immediate constraints, are enforced at the end of every DML statement. A constraint violation causes the statement to be rolled back. If a constraint causes an action such as `delete cascade`, the action is taken as part of the statement that caused it. A constraint that is defined as nondeferrable cannot be changed to a deferrable constraint.

Deferred constraints are constraints that are checked only when a transaction is committed. If any constraint violations are detected at commit time, then the entire transaction is rolled back. These constraints are most useful when both the parent and child rows in a foreign key relationship are entered at the same time, as in the case of an order entry system, where the order and the items in the order are entered at the same time.

A constraint that is defined as deferrable can be specified as one of the following:

- `Initially immediate` specifies that by default it must function as an immediate constraint unless explicitly set otherwise.
- `Initially deferred` specifies that by default the constraint must be enforced only at the end of the transaction.

Creating Constraints with SQL: Examples

a
```
ALTER TABLE countries
ADD (UNIQUE(country_name) ENABLE NOVALIDATE);
```

b
```
ALTER TABLE employees ADD CONSTRAINT pk PRIMARY KEY
(employee_id)
```

c
```
CREATE TABLE t1 (pk NUMBER PRIMARY KEY, fk NUMBER, c1 NUMBER,
c2 NUMBER,
CONSTRAINT ri FOREIGN KEY (fk) REFERENCES t1, CONSTRAINT ck1
CHECK (pk > 0 and c1 > 0));
```

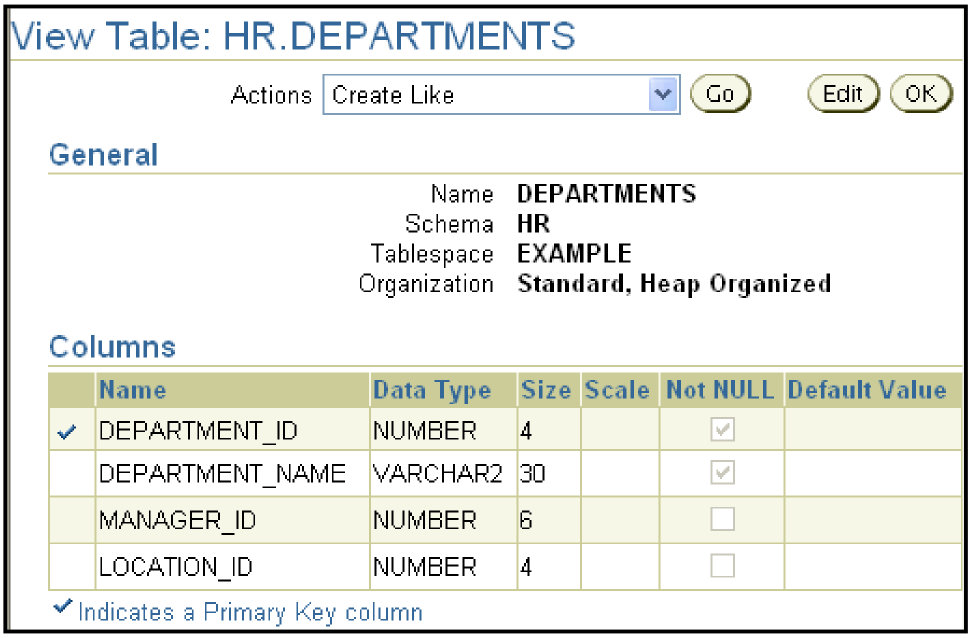

Creating Constraints with SQL: Examples

Three examples of constraint creation are shown in the slide:

a. After this statement executes, any inserts or updates done on the `COUNTRIES` table are required to have a `COUNTRY_NAME` value that is unique. But it is possible that when this statement is issued, there already exist `COUNTRY_NAME` values in the table that are not unique. The `NOVALIDATE` keyword indicates that they should be ignored. Only new rows are constrained.
b. This statement adds a primary key to the employee table. The constraint name is `PK` and the primary key is the `EMPLOYEE_ID` column.
c. This statement defines constraints at the time the table is created, rather than using an `ALTER TABLE` statement later. The `RI` constraint enforces that the values in the `FK` column must be present in the primary key column of the `T1` table. The `CK1` constraint enforces that the `PK` and `C1` columns are greater than zero.

Note: Each constraint has a name. If one is not supplied in the DDL statement, then a system-supplied name is assigned, which starts with `SYS_`.

Viewing the Columns in a Table



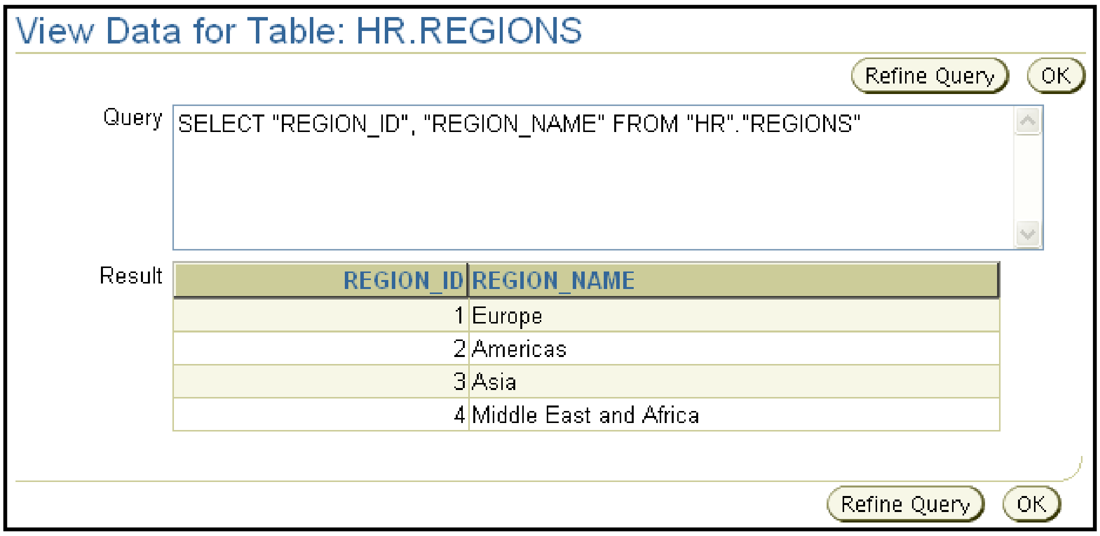
7-21

Viewing the Columns in a Table

To view the attributes of a table by using Enterprise Manager, perform the following steps:

1. Click the Tables link in the Schema region 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

7-22

Viewing the Contents of a Table

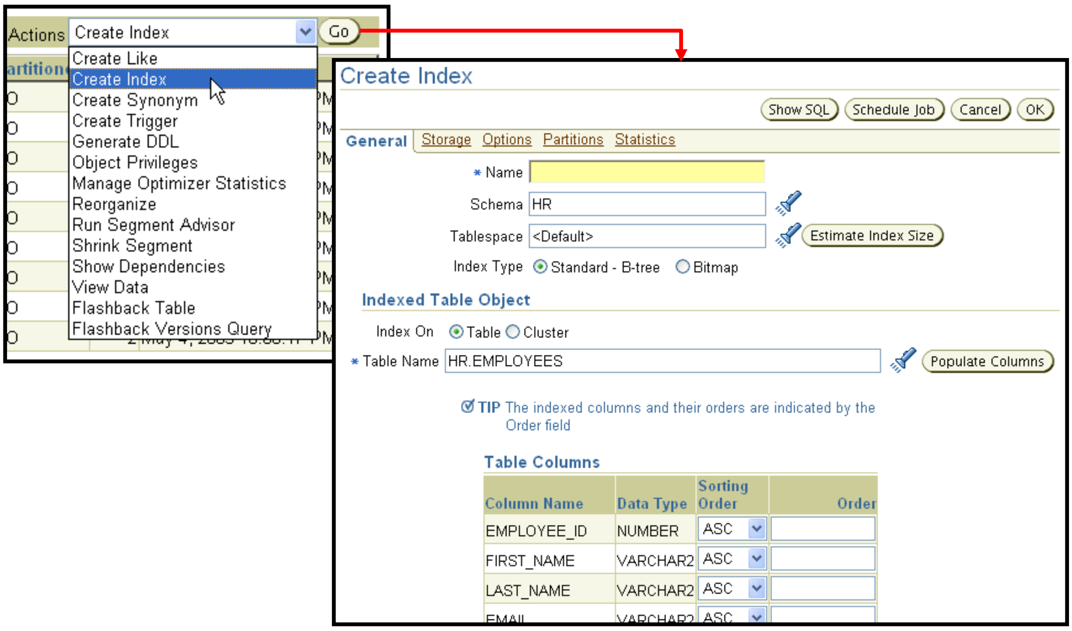
To view the rows in a table by using Enterprise Manager, perform the following steps:

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 Result region. The Query box displays the SQL query that is 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.

For more information about the `WHERE` clauses in SQL statements, refer to *Oracle Database SQL Reference*.

Actions with Tables



ORACLE

7-23

Actions with Tables

You can select a table and then perform actions on that table. Here are some of those actions:

- **Create Like:** With this action, you can create a table that has the same structure as the selected table. You must change the constraint names. You can add or delete columns and make other changes to the table structure before it is created.
- **Create Index:** Use this option to create indexes on a table.
- **Generate DDL:** This generates the DDL that represents the table as it already exists. This can then be copied to 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 privileges to other users in order for them to perform DML or possibly DDL on the table.
- **Show Dependencies:** This shows objects that this table depends on or objects that depend on this table.
- **View Data:** This selects and displays data from the table in a read-only manner.

Dropping a Table

Dropping a table removes:

- **Data**
- **Table structure**
- **Database triggers**
- **Corresponding indexes**
- **Associated object privileges**

```
DROP TABLE hr.employees PURGE;
```

Optional clauses for the `DROP TABLE` statement:

- **`CASCADE CONSTRAINTS`: Dependent referential integrity constraints**
- **`PURGE`: No flashback possible**

7-24

Dropping a Table

Syntax:

```
DROP TABLE [schema.] table [CASCADE CONSTRAINTS] [PURGE]
```

The `DROP TABLE` command removes data, the table structure, and associated object privileges. Some `DROP TABLE` considerations are as follows:

- Without the `PURGE` clause, the table definition, associated indexes, and triggers are placed in a recycle bin. The table data still exists, but is inaccessible without the table definition. If you drop a table through Enterprise Manager, the `PURGE` clause is not used.
- Use the `FLASHBACK TABLE` command to recover schema objects from the recycle bin. The `PURGE RECYCLEBIN` command empties the recycle bin.
- The `CASCADE CONSTRAINTS` option is required to remove all dependent referential integrity constraints.

Note: If you do not use the `PURGE` option, the space taken up by the table and its indexes still counts against the user's allowed quota for the tablespaces involved. That is, the space is still considered as being used.

Truncating a Table

```
TRUNCATE TABLE hr.employees;
```

- **Truncating a table makes its row data unavailable, and optionally releases used space.**
- **Corresponding indexes are truncated.**

Truncating a Table

Syntax:

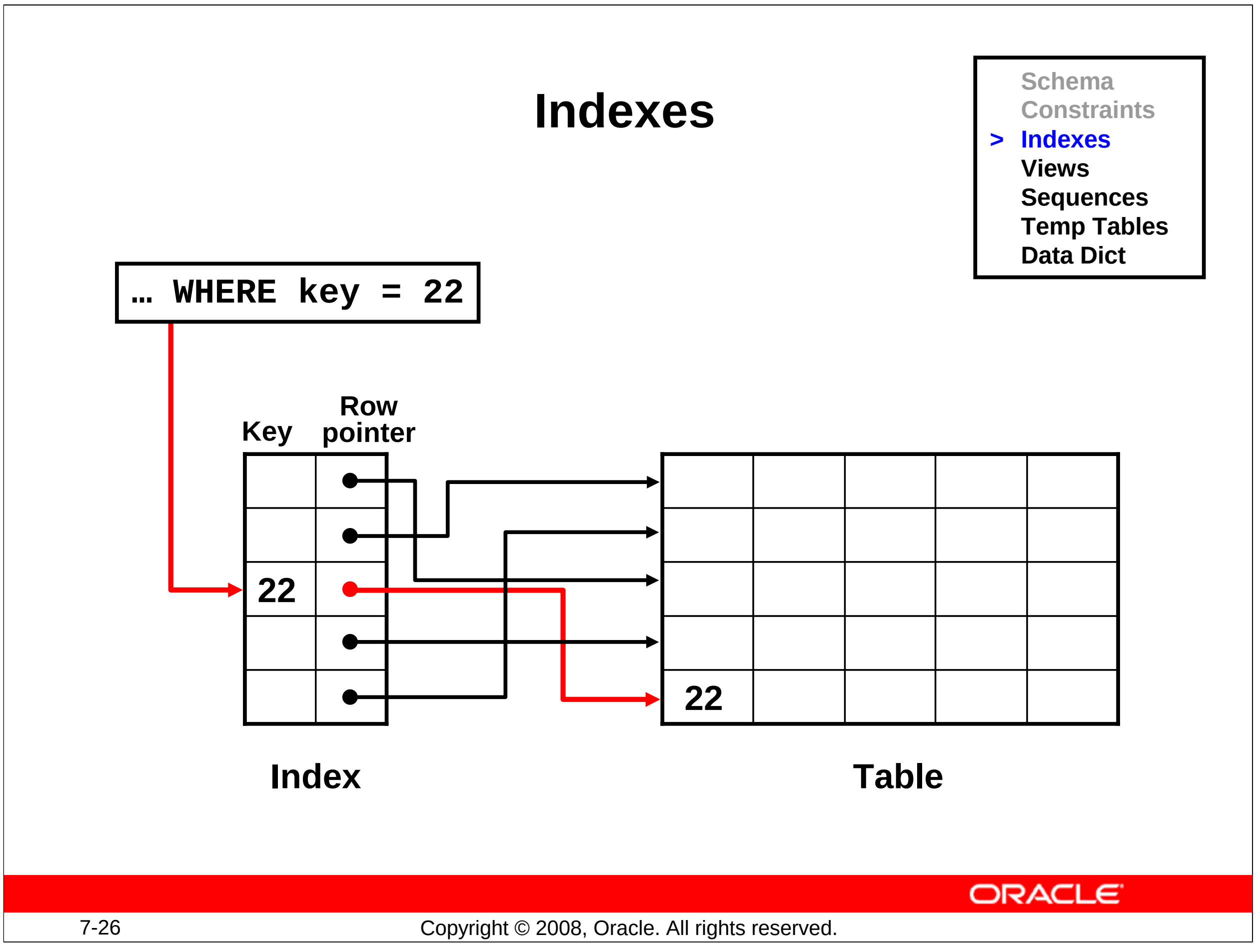
```
TRUNCATE TABLE [schema.] table [{DROP | REUSE} STORAGE]
```

The effects of using this command are as follows:

- The table is marked as empty by setting the high-water mark (HWM) to the beginning of the table, making its rows unavailable.
- No undo data is generated and the command commits implicitly because `TRUNCATE TABLE` is a DDL command.
- Corresponding indexes are also truncated.
- A table that is being referenced by a foreign key cannot be truncated.
- The delete triggers do not fire when this command is used.

This is usually many times faster than issuing a `DELETE` statement to delete all the rows of the table due to the following reasons:

- The Oracle database resets the table's HWM instead of processing each row as a `DELETE` operation.
- No undo data is generated.



Indexes

Indexes are optional structures associated with tables. They can be created to improve the performance of data update and retrieval. An Oracle index provides a direct access path to a row of 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 users.

Types of Indexes

These are several types of index structures available to you, depending on the need:

- **A B-tree index is in the form of a binary tree and is the default index type.**
- **A bitmap index has a bitmap for each distinct value indexed, and each bit position represents a row that may or may not contain the indexed value. This is best for low-cardinality columns.**

7-27

Types of Indexes

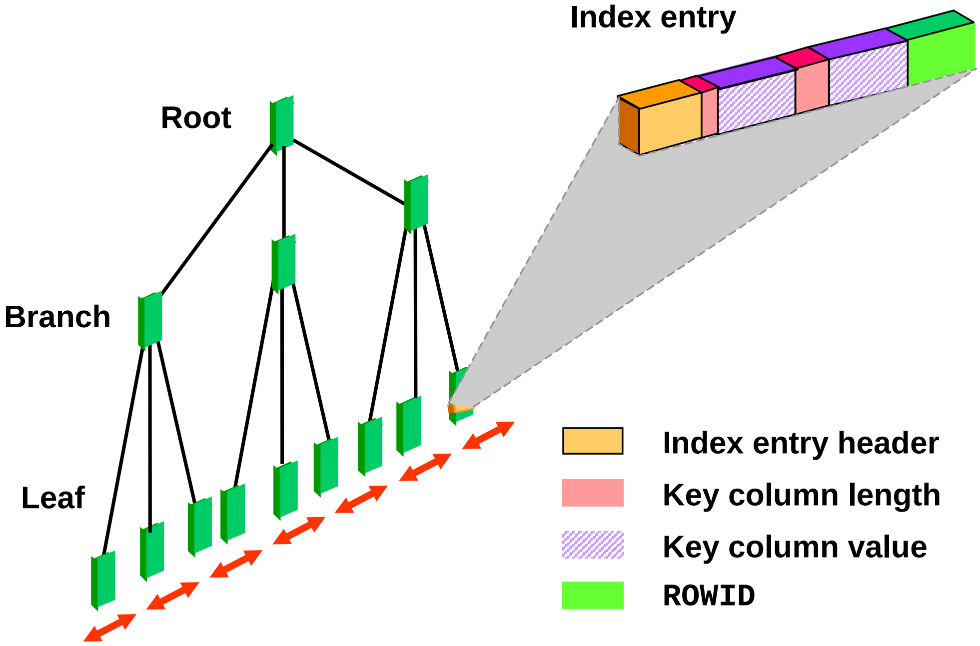
The following are the most common forms of indexes:

- B-tree
- Bitmap

A B-tree index has its key values stored in a balanced tree (B-tree), allowing for fast binary searches.

A bitmap index has a bitmap for each distinct key value being indexed. Within each bitmap, there is a bit set aside for each row in the table being indexed. This allows for fast lookups when there are few distinct values; that is, the indexed column has low cardinality. An example of this is a gender indicator. It can have values of "M" and "F" only. So, there are only two bitmaps to search. For example, if a bitmap index were used for a `phone_number` column, there would be so many bitmaps to manage and search that it would be very inefficient. Use bitmap indexes for low-cardinality columns.

B-Tree Index



B-Tree Index

Structure of a B-tree index

At the top of the index is the root, which contains entries that point to the next level in the index. At the next level are branch blocks, which in turn point to blocks at the next level in the index. At the lowest level are the leaf nodes, which contain the index entries that point to rows in the table. The leaf blocks are doubly linked to facilitate the scanning of the index in an ascending as well as descending order of key values.

Format of index leaf entries

An index entry is made up of the following components:

- An entry header, which stores the number of columns and locking information
- Key column length-value pairs, which define the size of a column in the key followed by the value for the column (The number of such pairs is a maximum of the number of columns in the index.)
- `ROWID` of a row that contains the key values

B-Tree Index (continued)

Index leaf entry characteristics

In a B-tree index on a nonpartitioned table:

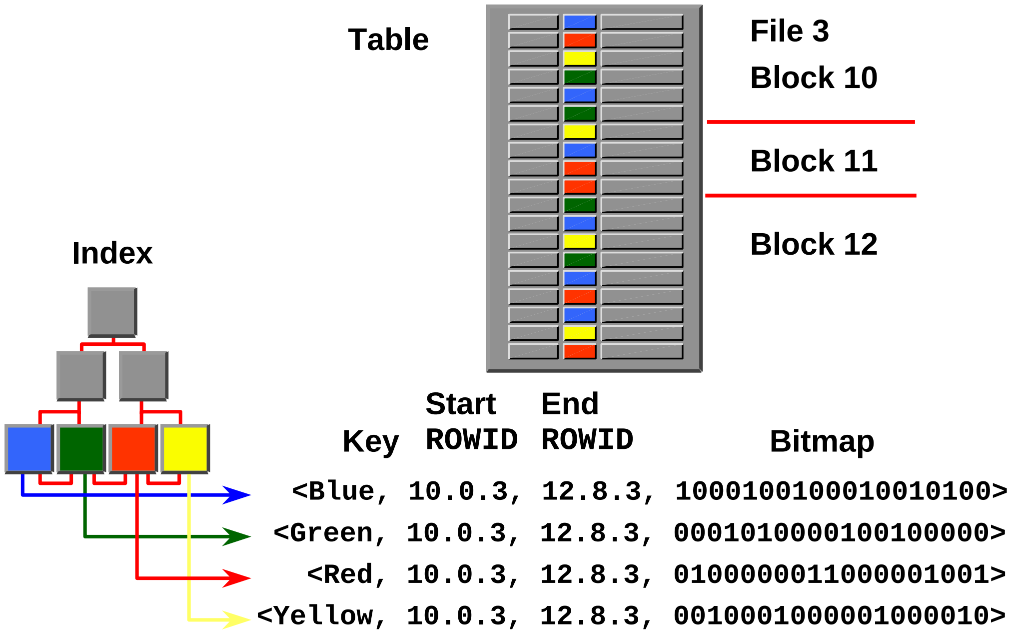
- Key values are repeated if there are multiple rows that have the same key value unless the index is compressed
- There is no index entry corresponding to a row that has all key columns that are `NULL`. Therefore, a `WHERE` clause specifying `NULL` will always result in a full table scan.
- Restricted `ROWID` is used to point to the rows of the table because all rows belong to the same segment

Effect of DML operations on an index

The Oracle server maintains all the indexes when DML operations are carried out on the table. Here is an explanation of the effect of a DML command on an index:

- Insert operations result in the insertion of an index entry in the appropriate block.
- Deleting a row results only in a logical deletion of the index entry. The space used by the deleted row is not available for new entries until all the entries in the block are deleted.
- Updates to the key columns result in a logical delete and an insert to the index. The `PCTFREE` setting has no effect on the index except at the time of creation. A new entry may be added to an index block even if it has less space than that specified by `PCTFREE`.

Bitmap Indexes



7-30

Bitmap Indexes

Bitmap indexes are more advantageous than B-tree indexes in certain situations:

- When a table has millions of rows and the key columns have low cardinality—that is, there are very few distinct values for the column. For example, bitmap indexes may be preferable to B-tree indexes for the gender and marital status columns of a table containing passport records.
- When queries often use a combination of multiple `WHERE` conditions involving the `OR` operator
- When there is read-only or low update activity on the key columns

Structure of a bitmap index

A bitmap index is also organized as a B-tree, but the leaf node stores a bitmap for each key value instead of a list of `ROWID`s. Each bit in the bitmap corresponds to a possible `ROWID`, and if the bit is set, it means that the row with the corresponding `ROWID` contains the key value.

As shown in the diagram, the leaf node of a bitmap index contains the following:

- An entry header that contains the number of columns and lock information

Bitmap Indexes (continued)

Structure of a bitmap index (continued)

- Key values consisting of length and value pairs for each key column. In the example, the key consists of only one column, and the first entry has a key value of Blue.
- Start `ROWID`, which in the example specifies block number ten, row number zero, and file number three
- End `ROWID`, which in the example specifies block number twelve, row number eight, and file number three
- A bitmap segment consisting of a string of bits. (The bit is set when the corresponding row contains the key value and is unset when the row does not contain the key value. The Oracle server uses a patented compression technique to store bitmap segments.)

The start `ROWID` is the `ROWID` of the first row pointed to by the bitmap segment of the bitmap—that is, the first bit of the bitmap corresponds to that `ROWID`, the second bit of the bitmap corresponds to the next row in the block, and the end `ROWID` is a pointer to the last row in the table covered by the bitmap segment. Bitmap indexes use restricted `ROWID`s.

Using a bitmap index

The B-tree is used to locate the leaf nodes that contain bitmap segments for a given value of the key. Start `ROWID` and the bitmap segments are used to locate the rows that contain the key value.

When changes are made to the key column in the table, bitmaps must be modified. This results in the locking of the relevant bitmap segments. Because locks are acquired on the whole bitmap segment, a row that is covered by the bitmap cannot be updated by other transactions until the first transaction ends.

Index Options

- **A unique index ensures that every indexed value is unique.**
- **An index can have its key values stored in ascending or descending order.**
- **A reverse key index has its key value bytes stored in reverse order.**
- **A composite index is one that is based on more than one column.**
- **A function-based index is an index based on a function's return value.**
- **A compressed index has repeated key values removed.**

ORACLE

7-32

Index Options

For efficiency of retrieval, it may be advantageous to have an index store the keys in descending order. This decision is made on the basis of how the data is accessed most frequently.

A reverse key index has the bytes of the indexed value stored in reverse order. This can reduce activity in a particular hot spot in the index. If many users are processing data in the same order, then the prefix portions of the key values (that are currently being processed) are close in value at any given instant. Consequently, there is a lot of activity in that area of the index structure. A reverse key index spreads that activity out across the index structure by indexing a reversed-byte version of the key values.

An index created by the combination of more than one column is called a composite index. For example, you can create an index based on a person's last name and first name:

```
CREATE INDEX name_ix ON employees
(last_name, first_name);
```

Index Options (continued)

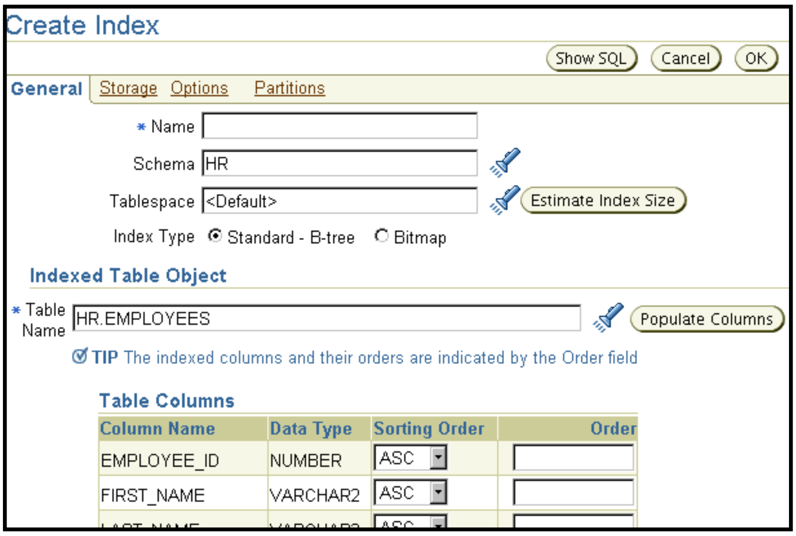
A function-based index indexes a function's return value. This function can be a built-in SQL function, a supplied PL/SQL function, or a user-written function. This relieves the server from having to invoke the function for every key value as it performs a search on the indexed expression. The following example indexes the returned tree volume that is computed by the function, based on each tree's species, height, and volume (which are columns in the `TREES` table):

```
CREATE INDEX tree_vol_ix ON
TREES(volume(species,height,circumference));
```

Then, any query that contains the expression `volume(species,height,circumference)` in the `WHERE` clause may be able to take advantage of this index, and execute much more quickly because the volume computation is already done for each tree. Function-based indexes are maintained automatically, as are normal indexes.

You can use a compressed index to reduce disk consumption at execution time. Because repeated key values are removed, more index entries can fit in a given amount of disk space, resulting in the ability to read more entries from the disk in the same amount of time. Compression and decompression must be performed for the writing and reading of the index, respectively.

Creating Indexes




```
CREATE INDEX my_index ON
employees(last_name, first_name);
```


Creating Indexes

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. An example of an implicitly created index is the definition of a primary key, in which case a unique index would be automatically created to enforce uniqueness on the column.

What Is a View?

Schema
Constraints
Indexes
> **Views**
...

LOCATION table

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
2200	12-98 Victoria Street	2901	Sydney	New South Wales	AU
2800	Rua Frei Caneca 1360	01307-002	Sao Paulo	Sao Paulo	BR
1000	1297 Via Cola di Rie	00989	Roma		IT
1100	93091 Calle della Testa	10934	Venice		IT

COUNTRY table

CO	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2

View

LOCATION_ID	COUNTRY_NAME
2200	Australia
2800	Brazil

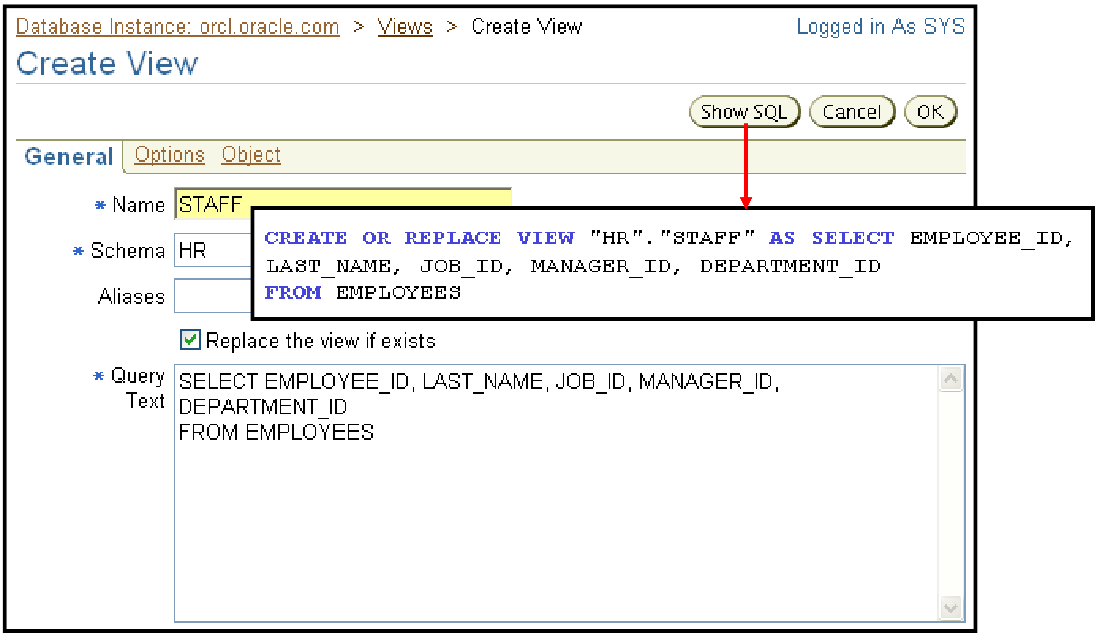
```
CREATE VIEW v AS SELECT location_id, country_name FROM
locations l, countries c
WHERE l.country_id = c.country_id AND c.country_id in
('AU','BR');
```

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 because they can hide very complex conditions, joins, and other complex expressions and SQL constructs. Views do not actually contain data; instead, they derive their data from the tables on which they are based. These tables are referred to as the base tables of the view.

Creating Views



7-36

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.

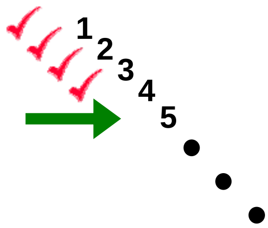
To see the views defined in the database, click the Views link under the Schema heading on the Administration page.

Sequences

Schema
Constraints
Indexes
Views
> **Sequences**
Temp Tables
Data Dict

A sequence is a mechanism for automatically generating integers that follow a pattern.

- **A sequence has a name, which is how it is referenced when the next value is requested.**
- **A sequence is not associated with any particular table or column.**
- **The progression can be ascending or descending.**
- **The interval between numbers can be of any size.**
- **A sequence can cycle when a limit is reached.**



ORACLE

7-37

Sequences

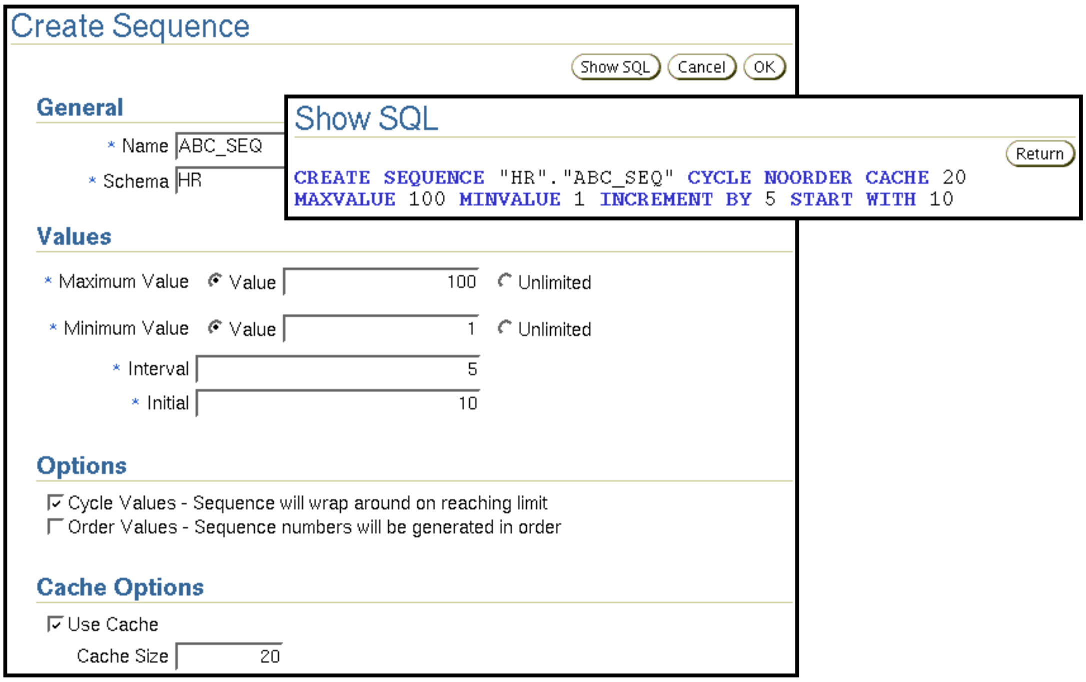
To retrieve the next value from a sequence, you reference it by its name; there is no association of a sequence to a table or a column.

After a given number is issued, it will not be issued again, unless the sequence is defined as cyclical. Sometimes an application requests a value it never ends up using or storing in the database. This may result in gaps in the numbers that reside in the table that they are being stored into.

Caching of sequence numbers improves performance because a set of numbers is preallocated in memory for faster access. If there is an instance failure, any cached sequence numbers are not used, which results in gaps.

Note: If an application requires that there be no gaps, then the application should implement a custom number generator. However, this method can result in very poor performance. If you use a table to store a value, and increment that value and update the table for each request, that process would be a systemwide bottleneck. This is because every session would have to wait for that mechanism, which, to guarantee no duplicates or gaps, can handle only a single request at a time.

Creating a Sequence



Creating a Sequence

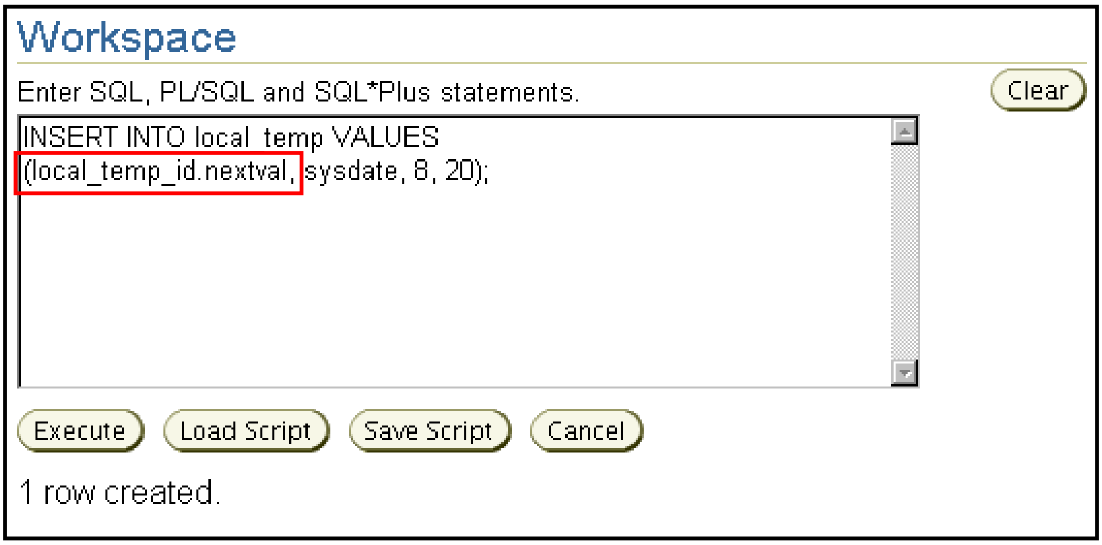
You can view and create sequences with Enterprise Manager by clicking the Sequences link under the Schema heading of the Administration page. Here is a summary of the sequence creation options:

- **Name:** This is the name of the sequence, which is how it is referenced.
- **Schema:** This is the owner of the sequence.
- **Maximum Value:** Specify the maximum value that the sequence can generate. This integer value can have 28 or fewer digits. It must be greater than Minimum Value and Initial. Using Unlimited indicates the 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 than or equal to Initial and less than Maximum Value. Using Unlimited indicates the minimum value of 1 for an ascending sequence or -10^{26} for a descending sequence. The default is Unlimited.

Creating 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. It can have 28 or fewer digits. The default value 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 is returned when you attempt to retrieve a value after the sequence has been exhausted.
- **Order Values:** This guarantees that sequence numbers are generated in the order of request. This clause is useful if you are using sequence numbers as timestamps. Guaranteeing order is usually not important for sequences that are used to generate primary keys. This option is necessary only to guarantee ordered generation if you are using the Oracle database with Real Application Clusters.
- **Cache Options:** Specify how many values of the sequence the Oracle database 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 what would fit in a given cycle of sequence numbers.

Using a Sequence



Using a Sequence

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
```

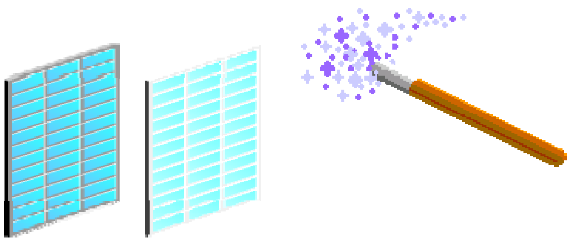
The first reference to NEXTVAL returns the initial value of the sequence. Subsequent references to NEXTVAL increment the sequence value by the defined increment and return the new value. Any reference to CURRVAL always returns the current value of the sequence, which is the value returned by the last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

Temporary Tables

- Schema
- Constraints
- Indexes
- Views
- Sequences
- > **Temp Tables**
- Data Dict

A temporary table:

- **Provides storage of data that is automatically cleaned up when the session or transaction ends**
- **Provides private storage of data for each session**
- **Is available to all sessions for use without affecting each other's private data**



ORACLE

7-41

Temporary Tables

You can take advantage of temporary tables when you need to privately store data for the purpose of performing a task, and you want the data to be cleaned up when that task is performed, at the end of either a transaction or a session. Temporary tables provide this functionality while relieving you of the responsibilities of hiding your data from other sessions, and removing the generated data when you have finished. The only temporary table data visible to a session is the data that the session has inserted.

A temporary table can be transaction specific or session specific. For transaction-specific temporary tables, data exists for the duration of the transaction whereas for session-specific temporary tables, data exists for the duration of the session. In both cases, the data inserted by a session is private to the session. Each session can view and modify only its own data. As a result, DML locks are never acquired on the data of temporary tables. The following clauses control the lifetime of the rows:

- **`ON COMMIT DELETE ROWS`:** To specify that the lifetime of the inserted rows is for the duration of the transaction only
- **`ON COMMIT PRESERVE ROWS`:** To specify that the lifetime of the inserted rows is for the duration of the session

Temporary Tables (continued)

The `CREATE GLOBAL TEMPORARY TABLE` statement creates a temporary table. You can create indexes, views, and triggers on temporary tables, and you can also use Export and Import or Data Pump to export and import the definition of a temporary table. However, no data is exported, even if you use the `ROWS` option.

In addition to the already mentioned events that cause the data to be deleted, you can force the data to be removed efficiently with the `TRUNCATE TABLE` command. This removes all the data that you have inserted. It is more efficient than using the `DELETE` command.

You can create indexes, views, and triggers on temporary tables.

Temporary tables can be created using Enterprise Manager by clicking the Temporary option on the Create Table: Table Organization page. Click Continue, and the next page enables you to specify whether the temporary table is session specific or transaction specific. The Tablespace field is disabled because a temporary table is always created in the user's temporary tablespace; no other tablespace can be specified.

Note: The `GLOBAL` keyword is based on the terminology specified in the International Organization for Standardization (ISO) standard for SQL.

Temporary Tables: Considerations

- Use the `GLOBAL TEMPORARY` clause to create temporary tables:

```
CREATE GLOBAL TEMPORARY TABLE employees_temp
ON COMMIT PRESERVE ROWS
AS SELECT * FROM employees;
```

- Use the `TRUNCATE TABLE` command to delete the contents of the table.
- You can create the following on temporary tables:
  – Indexes
  – Views
  – Triggers

ORACLE

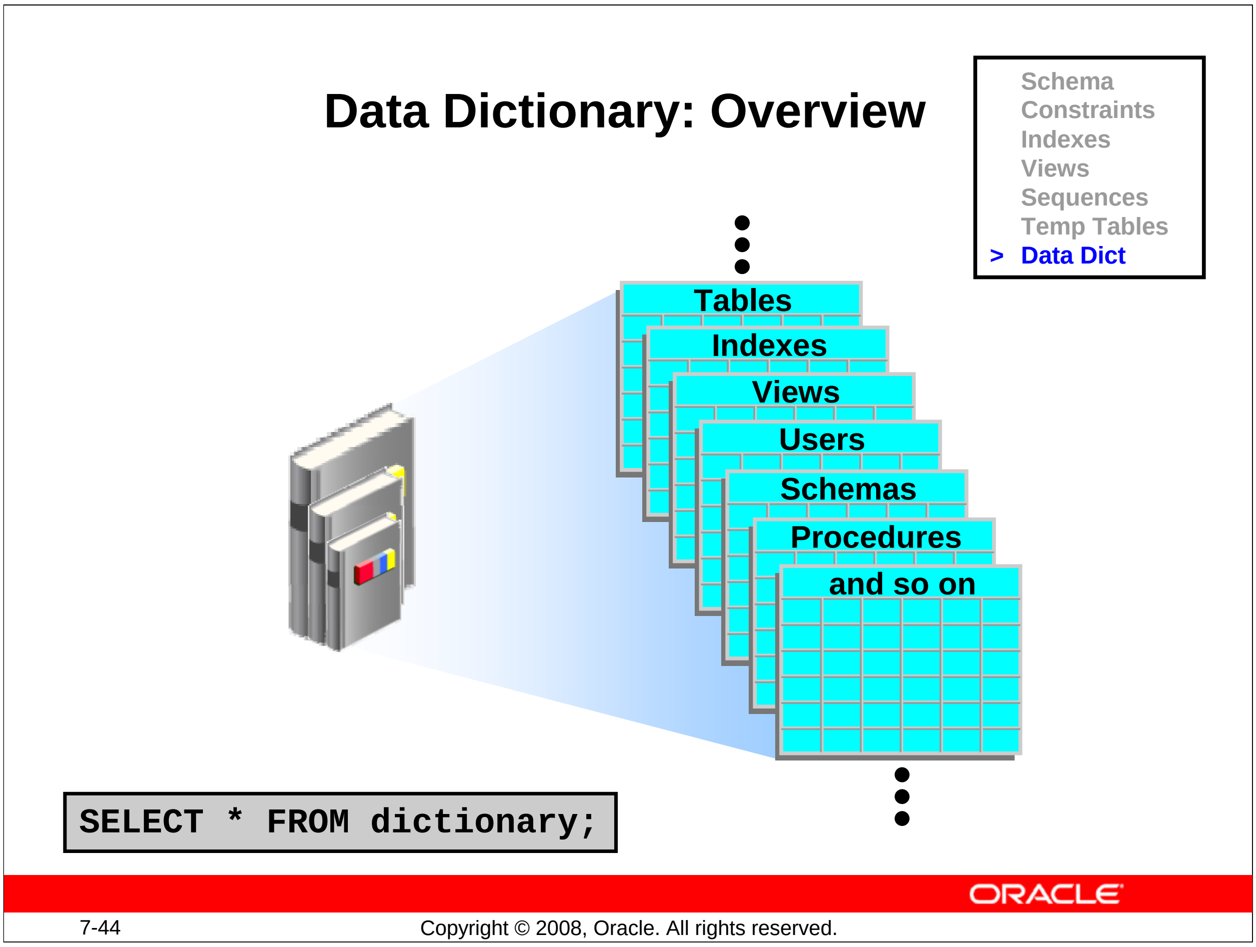
7-43

Temporary Tables: Considerations

The `CREATE GLOBAL TEMPORARY TABLE` statement creates a temporary table. You can create indexes, views, and triggers on temporary tables, and you can also use Export and Import or Data Pump to export and import the definition of a temporary table. However, no data is exported even if you use the `ROWS` option.

In addition to the already mentioned events that cause the data to be deleted, you can force the data to be removed efficiently with the `TRUNCATE TABLE` command. This removes all the data that you have inserted. It is more efficient than using the `DELETE` command.

Note: The `GLOBAL` keyword is based on the terminology specified in the International Organization for Standardization (ISO) standard for SQL.



Data Dictionary: Overview

Oracle's data dictionary is the description of a database. It contains the names and attributes of all objects in the database. The creation or modification of any object causes an update to the data dictionary that reflects those changes. This information is stored in the base tables that are maintained by the Oracle database, but you access these tables by using predefined views rather than reading the tables directly.

The data dictionary:

- Is used by the Oracle database server to find information about users, objects, constraints, and storage
- Is maintained by the Oracle database server as object structures or definitions are modified
- Is available for use by any user to query information about the database
- Is owned by the `SYS` user
- Should never be modified directly using SQL

Note: The `DICTIONARY` data dictionary view, or the `DICT` synonym for this, contains the names and descriptions of everything in the data dictionary. Use the `DICT_COLUMNS` view to see the view columns and their definitions. For complete definitions of each view, see the *Oracle Database Reference* documentation.

Data Dictionary Views

	Who Can Query	Contents	Subset of	Notes
DBA_	DBA	Everything	N/A	May have additional columns meant for DBA use only
ALL_	Everyone	Everything that the user has privileges to see	DBA_ views	Includes user's own objects
USER_	Everyone	Everything that the user owns	ALL_ views	Is usually the same as ALL_ except for the missing OWNER column. Some views have abbreviated names as PUBLIC synonyms.

Data Dictionary Views

The view prefixes indicate what or how much data a given user can see. The global view of everything is accessed only by users with DBA privileges, using the DBA_ prefix. The next level of privilege is at the ALL_ prefix level, which represents all objects that the querying user is privileged to see, whether he or she owns them or not. For example, if USER_A has been granted access to a table owned by USER_B, then USER_A sees that table listed in any ALL_ view dealing with table names. The USER_ prefix represents the smallest scope of visibility. This shows only those objects that the querying user owns; that is, those that are present in his or her own schema.

Data Dictionary Views (continued)

Generally, each view set is a subset of the higher privileged view set, row-wise and columnwise. Not all views in a given view set have a corresponding view in the other view sets. This is dependent on the nature of the information in the view. For example, there is a `DBA_LOCK` view, but there is no `ALL_LOCK` view. This is because only a DBA would have interest in data about locks. You should be certain to choose the appropriate view set to meet the need that you have. If you have the privilege to access the `DBA` views, you still may want to query only the `USER` version of the view because you know that it is something that you own and you do not want other objects to be added to your result set.

The `DBA_` views can be queried by users with the `SYSDBA` or `SELECT ANY DICTIONARY` privilege.

Data Dictionary: Usage Examples

a
```
SELECT table_name, tablespace_name FROM
user_tables;
```

b
```
SELECT sequence_name, min_value, max_value,
increment_by FROM all_sequences WHERE
sequence_owner IN ('MDSYS','XDB');
```

c
```
SELECT USERNAME, ACCOUNT_STATUS FROM
dba_users WHERE ACCOUNT_STATUS = 'OPEN';
```

d
```
DESCRIBE dba_indexes;
```

ORACLE

Static Data Dictionary: Usage Examples

The examples in the slide show queries that answer these questions:

a. What are the names of the tables (along with the name of the tablespace where they reside) that have been created in your schema?
b. What is the significant information about any sequences in the database that you have access to?
c. What users in this database are currently able to log in?
d. What are the columns of the DBA_INDEXES view? This shows you what information you can view about all the indexes in the database. The following is a partial output of this command:

```
SQL> DESCRIBE dba_indexes;
 Name            Null?    Type
 --------------- -------- -------------
 OWNER           NOT NULL VARCHAR2(30)
 INDEX_NAME      NOT NULL VARCHAR2(30)
 INDEX_TYPE               VARCHAR2(27)
 TABLE_OWNER     NOT NULL VARCHAR2(30)
 TABLE_NAME      NOT NULL VARCHAR2(30)
```

Summary

In this lesson, you should have learned how to:

- Define schema objects and data types
- Create and modify tables
- Define constraints
- View the columns and contents of a table
- Create indexes
- Create views
- Create sequences
- Explain the use of temporary tables
- Use the data dictionary

Practice Overview: Administering Schema Objects

This practice covers the following topics:

- Creating tables with columns
- Creating constraints:
  - Primary Key
  - Foreign Key
  - Check constraint
- Creating indexes

7-49

Managing Data and Concurrency

8

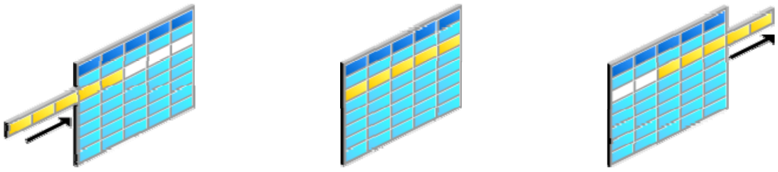
Objectives

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

- Manage data through the use of SQL
- Identify and administer PL/SQL objects
- Describe triggers and triggering events
- Monitor and resolve locking conflicts

Manipulating Data Through SQL

> SQL
PL/SQL
Locks



```
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

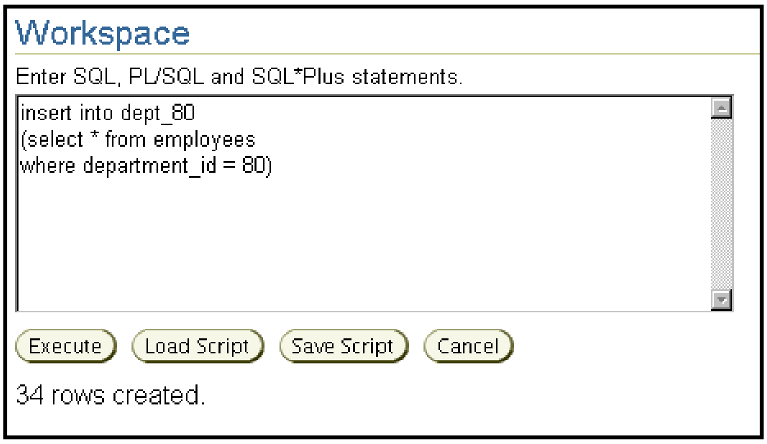
8-3

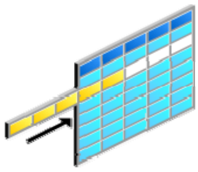
Manipulating Data Through SQL

The basic data manipulation language (DML) statements are the way data is manipulated in the database. Although these statements are briefly mentioned in the lesson titled “Moving Data,” they are covered in more detail in this lesson.

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` command to copy rows from one table to another. This method is also referred to as an `INSERT SELECT` statement. The example in 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 exactly the same structure as the `employees` table. If this is not the case, you can name the columns in each table. The values selected in the `SELECT` statement are associated with the columns of the table being inserted into, respectively. The column values match in the order as named in the `INSERT` and `SELECT` statements. All that is required is that the data types match. For example:

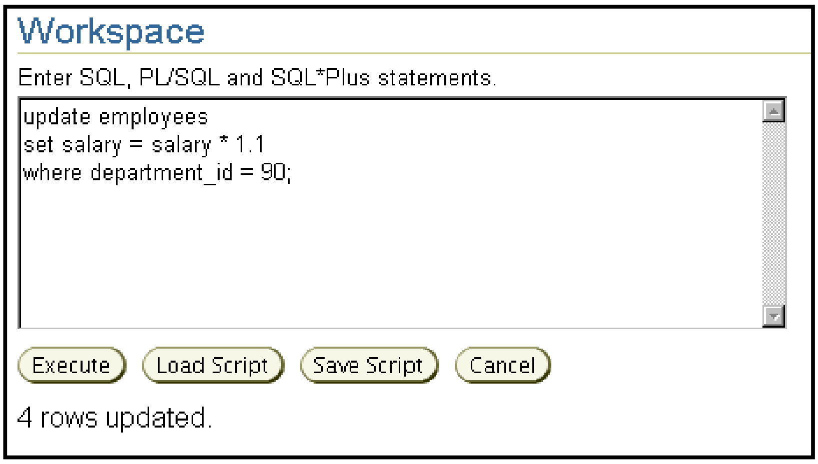
```
insert into just_names (first, last)
(select first_name, last_name from employees);
```

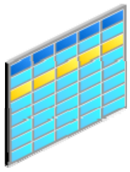
Here, the `just_names` table has only two columns that have the same data type as the `first_name` and `last_name` columns in the `employees` table.

Using the `INSERT SELECT` method is a way to load data in bulk from one or more tables into another table.

The UPDATE Command

Use the UPDATE command to change zero or more rows of a table.





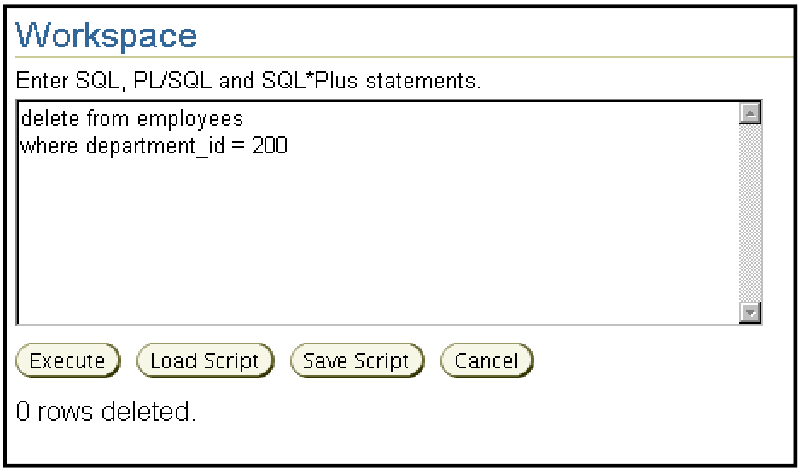
ORACLE

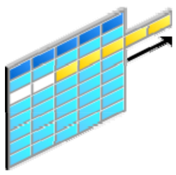
The UPDATE Command

The UPDATE command is used to modify existing rows in a table. The number of rows modified by the UPDATE command depends on the WHERE condition. If the WHERE clause is omitted, then all rows are changed. If no rows satisfy the WHERE condition, then no rows are modified.

The DELETE Command

Use the DELETE command 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 of rows modified by the DELETE command depends on the WHERE condition. If the WHERE clause is omitted, then all rows are removed. If no rows satisfy the WHERE condition, then no rows are removed. Note that 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 MERGE Command

Use the MERGE command to perform both INSERT and UPDATE in a single command.

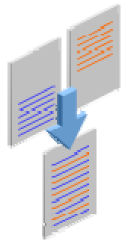
Workspace

Enter SQL, PL/SQL and SQL*Plus statements.

```
MERGE INTO jobs j
  USING (SELECT * FROM jobs_acquisition) a
  ON (j.job_id = a.job_id)
  WHEN MATCHED THEN UPDATE SET j.job_title = a.job_title
  WHEN NOT MATCHED THEN INSERT
    (j.job_id, j.job_title, j.min_salary, j.max_salary)
  VALUES (a.job_id, a.job_title, a.min_salary, a.max_salary)
```

Execute | Load Script | Save Script | Cancel

5 rows merged.



ORACLE

The MERGE Command

The MERGE command performs UPDATE and INSERT in the same command. You can merge data from one source into another, optionally inserting new rows and updating certain columns if a row already exists.

Consider this example. Some of the data in the JOBS table looks like this:

```
JOB_ID     JOB_TITLE      MIN_SALARY MAX_SALARY
---------- -------------- ---------- ----------
AD_PRES    President      20000      40000
FI_ACCOUNT Accountant      4200       9000
ST_CLERK   Stock Clerk     2000       5000
IT_PROG    Programmer      4000      10000
```

The MERGE Command (continued)

The following are the contents of the JOBS_ACQUISITION table:

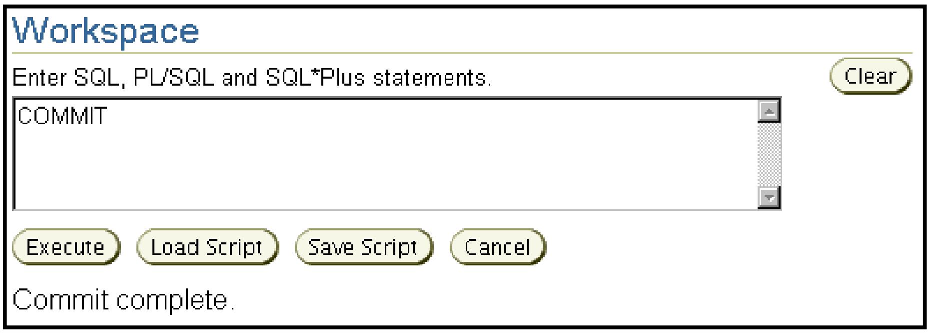
```
JOB_ID     JOB_TITLE      MIN_SALARY MAX_SALARY
---------- -------------- ---------- ----------
AD_PRES    VP             20000      40000
DBA        DB Admin        4200       9000
SA         Sys Admin       2000       5000
```

The MERGE command inserts into the JOBS table any rows with a new JOB_ID, and updates the existing JOBS row with JOB_TITLE, if JOB_ID already exists. The result is that the "President" job title is changed to "VP" and the new jobs of "SA" and "DBA" are added.

The COMMIT and ROLLBACK Commands

The following are used to finish a transaction:

- **COMMIT: Makes the change permanent**
- **ROLLBACK: Undoes the change**



The COMMIT and ROLLBACK Commands

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

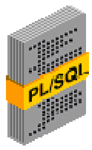
By default, a user trying to make a change on the same row as another user waits until the first user either commits or rolls back the change. This is controlled automatically by Oracle database’s locking mechanism. Because the locking mechanism is built into the row itself, there is no way the database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

PL/SQL

SQL
> PL/SQL
Locks

Oracle's Procedural Language extension to SQL (PL/SQL) is a fourth-generation programming language (4GL). It provides:

- **Procedural extensions to SQL**
- **Portability across platforms and products**
- **Higher level of security and data integrity protection**
- **Support for object-oriented programming**



ORACLE

8-10

PL/SQL

PL/SQL is an Oracle proprietary fourth-generation programming language that provides procedural extensions to SQL. 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.

PL/SQL also provides protection of data. The caller need not know the data structures being read or manipulated to make the call. The caller also need not have permission to access those objects; if the caller has permission to execute the PL/SQL program, that is all that is necessary. Optionally, there is another mode of permissions for calling PL/SQL, where the caller must have permission to perform every statement being executed during the called program.

PL/SQL (continued)

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 about procedural constructs and uses of PL/SQL, refer to the *PL/SQL User's Guide and Reference* documentation.

Administering PL/SQL Objects

Database administrators should be able to:

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

Administering PL/SQL Objects

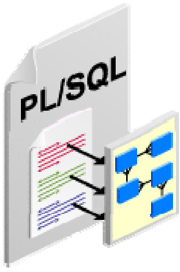
As a DBA, you are not usually responsible for loading PL/SQL code into the database and for assisting developers in troubleshooting. You are also not usually expected to write applications by using PL/SQL, but you should be familiar with the different PL/SQL objects sufficiently to make recommendations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

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

PL/SQL Objects

There are many types of PL/SQL database objects:

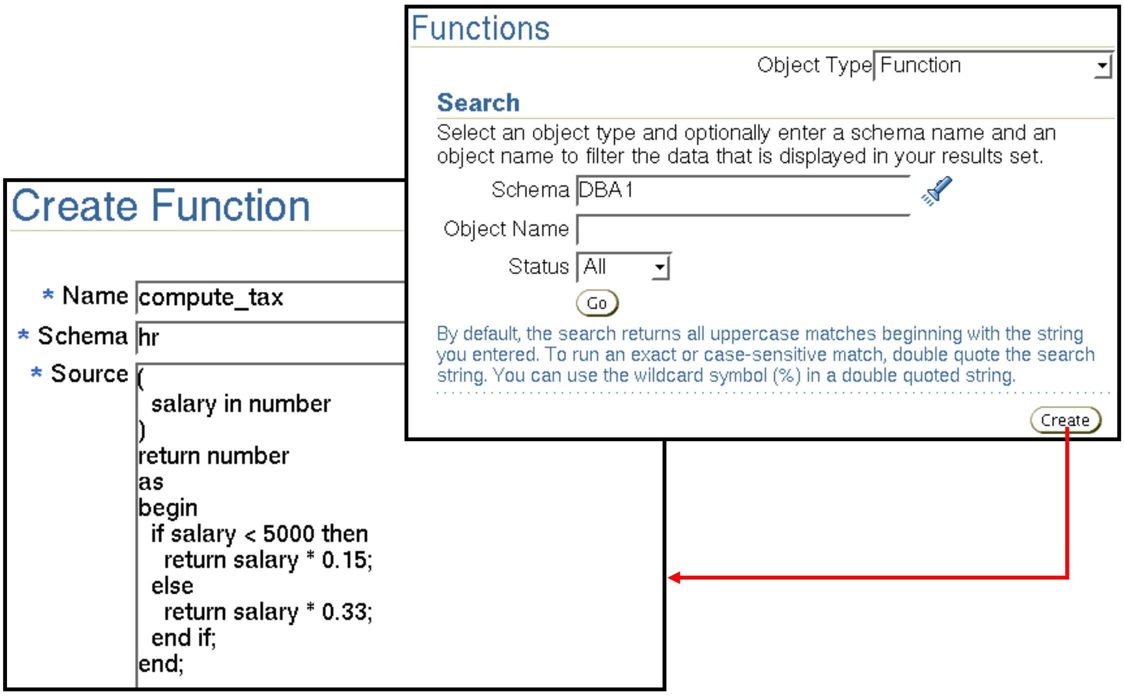
- **Package**
- **Package body**
- **Type body**
- **Procedure**
- **Function**
- **Trigger**



PL/SQL Objects

- **Package:** 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 describes 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 contains implementation details and private declarations, which are hidden from the caller.
- **Type body:** It is a collection of methods (procedures and functions) associated with user-defined data types. For more information about user-defined data types, refer to *Oracle Database Application Developer's Guide—Object Relational Features*.
- **Procedure:** A procedure is a PL/SQL block that performs a specific action.
- **Function:** A function is a PL/SQL block that returns a single value by using the `RETURN` PL/SQL command. It is a procedure that has a return value.
- **Trigger:** A trigger is a PL/SQL block that is executed when a particular event occurs 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

8-14

Functions

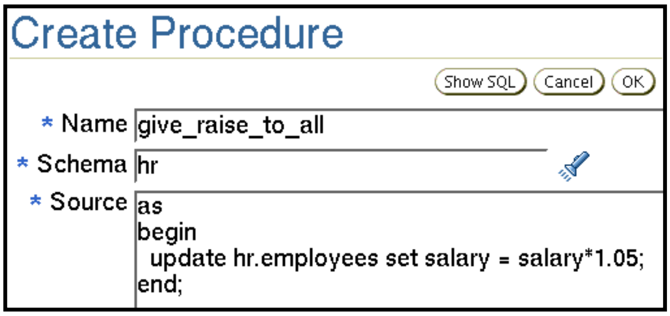
PL/SQL functions are typically used to compute a value. There are many built-in functions such as SYSDATE, SUM, AVG, and TO_DATE. Developers also create their own functions when writing applications. The code for a PL/SQL function *must* contain a RETURN statement. PL/SQL functions are created by entering a name, schema, and source code as shown in the slide.

The compute_tax function shown in the slide is created with the following SQL command:

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

Procedures

- Are used to perform a specific action
- Pass values in and out by using an argument list
- Can be invoked using:
  – The `CALL` command, which is a SQL statement
  – The `EXECUTE` command, which is a SQL*Plus command



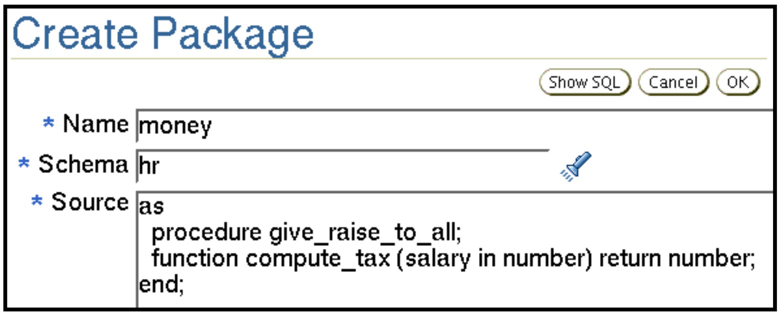
Procedures

PL/SQL procedures perform a specific action. Like functions, procedures can accept input values and perform conditional statements such as `IF-THEN`, `CASE`, and `LOOP`.

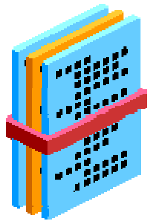
Packages

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

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



Package specification



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:** This object (sometimes known as the package header) has an object type of `PACKAGE` and contains only 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.

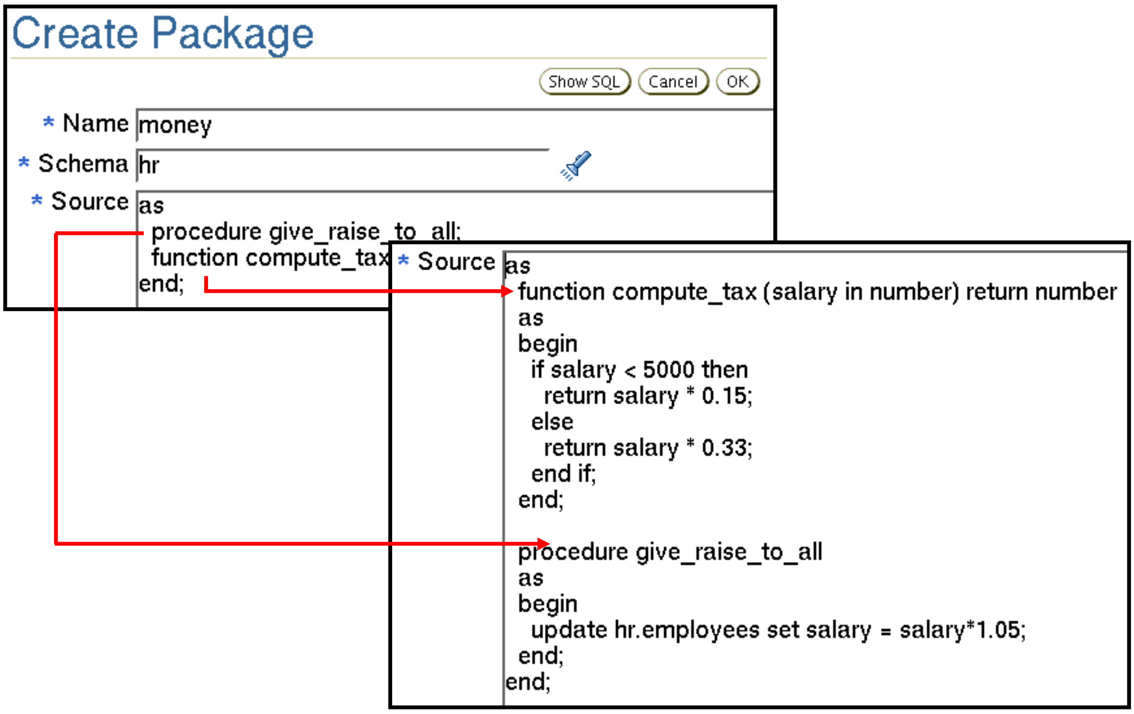
Procedures and functions called from within a package are called using dot notation:

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

In the package shown in the slide, the subprograms can be invoked as follows:

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

Package Specification and Body



8-17

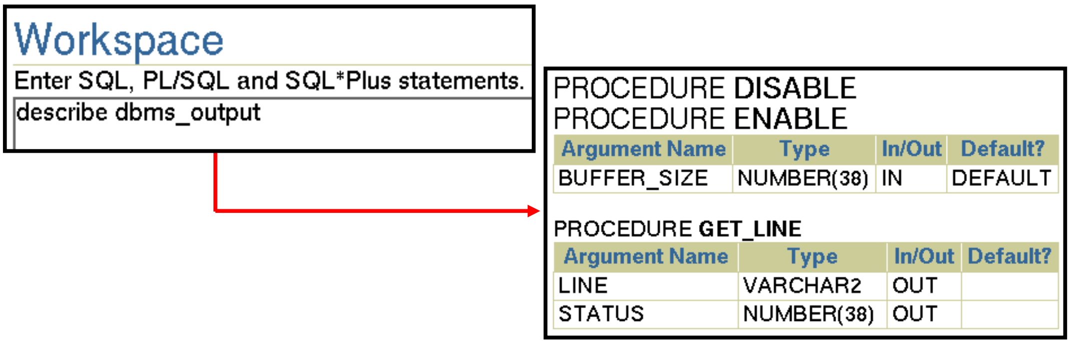
Package Body

Package bodies:

- Are separate from package specifications. Because of this, the code of the body can be changed and recompiled, and other objects that are dependent 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 about the use of wrap, see the *PL/SQL User’s Guide and Reference*.

Built-in Packages

- The Oracle database comes with over 350 built-in PL/SQL packages, which provide:
  - Administration and maintenance utilities
  - Extended functionality
- Use the `DESCRIBE` command to view subprograms.



Built-in Packages

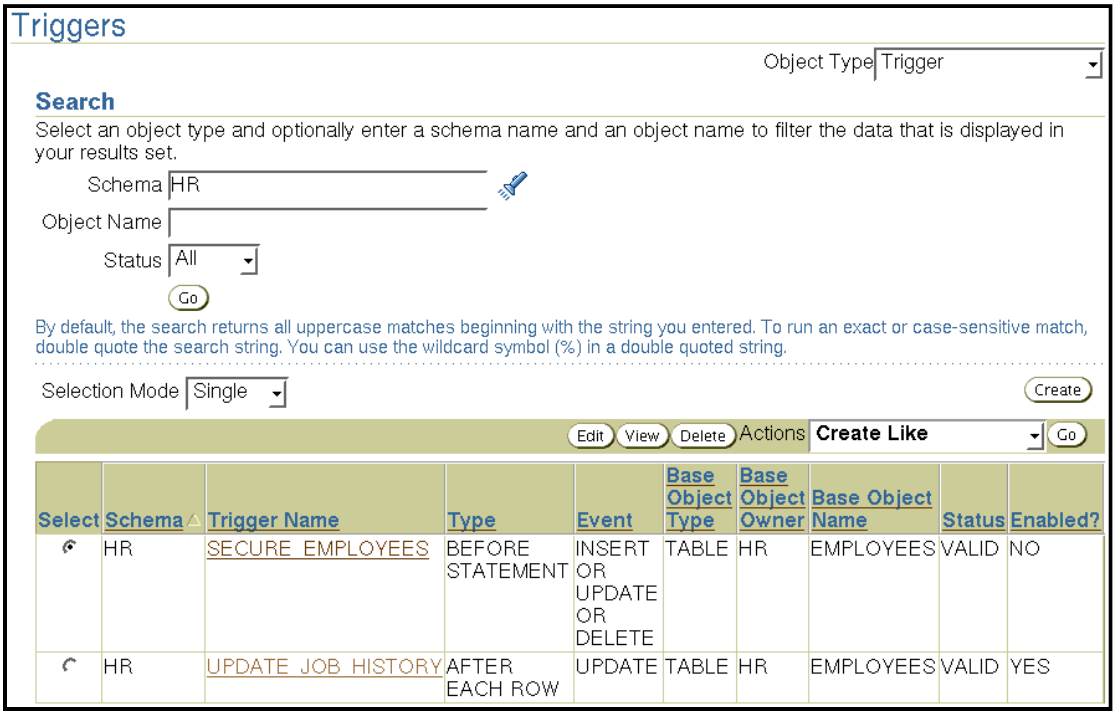
The built-in PL/SQL packages that are supplied with the Oracle database provide access to extended database functionalities, such as advanced queuing, encryption, and file input/output (I/O). They also include many administration and maintenance utilities.

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

- `DBMS_STATS`: Gather, view, and modify optimizer statistics
- `DBMS_OUTPUT`: Generate output from PL/SQL
- `DBMS_SESSION`: PL/SQL access to the `ALTER SESSION` and `SET ROLE` statements
- `DBMS_RANDOM`: Generate random numbers
- `DBMS_UTILITY`: Get time, CPU time, and version information; compute a hash value, and perform many other miscellaneous functionalities
- `DBMS_SCHEDULER`: Schedule functions and procedures that are callable from PL/SQL
- `DBMS_CRYPTO`: Encrypt and decrypt database data
- `UTL_FILE`: Read and write to operating system files from PL/SQL

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

Triggers



8-19

Triggers

Triggers are PL/SQL code objects that are stored in the database and which automatically run or “fire” when something happens. The Oracle database allows many actions to serve as triggering events, including an insert into a table, a user logging in to the database, and someone trying to drop a table or change audit settings.

Triggers may call other procedures or functions. It is best to keep the trigger’s code very short and place anything that requires lengthy code in a separate package.

DBAs use triggers to assist in value-based auditing (discussed in the lesson titled “Implementing Oracle Database Security”), to enforce complex constraints, and to automate many tasks. For example, the `SECURE_EMPLOYEES` trigger that is shown in the slide logs all DML statements to a holding table.

Triggering Events

Event Type	Examples of Events
DML	`INSERT`, `UPDATE`, `DELETE`
DDL	`CREATE`, `DROP`, `ALTER`, `GRANT`, `REVOKE`, `RENAME`
Database	`LOGON`, `LOGOFF`, `STARTUP`, `SHUTDOWN`, `SERVERERROR`, `SUSPEND`

8-20

Triggering Events

There are many events that can be used to fire a trigger, and they are divided into three categories.

- DML event triggers fire when statements modify data.
- DDL event triggers fire when statements create or in some way modify an object.
- Database event triggers fire when certain things happen in the database.

Most triggers can be specified to fire either before the event is allowed to occur or after it has occurred. For DML events, the trigger can be designed to fire once for the statement or with each row that is modified.

Locks

SQL
PL/SQL
> Locks

- Locks prevent multiple sessions from changing the same data at the same time.
- They are automatically obtained at the lowest possible level for a given statement.
- They do not escalate.

Transaction 1

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3 WHERE employee_id=100;
```


Locks

Before the database allows a session to modify data, the session must first lock the data that is 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 the transaction ends (with the `COMMIT` or `ROLLBACK` operation)

Transaction 1

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id=101;
```

ORACLE

8-22

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 block-level 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 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 `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 that transaction.

Data Concurrency

Time:		
09:00:00	Transaction 1	`UPDATE hr.employees SET salary=salary+100 WHERE employee_id=100;`
	Transaction 2	`UPDATE hr.employees SET salary=salary+100 WHERE employee_id=101;`
	Transaction 3	`UPDATE hr.employees SET salary=salary+100 WHERE employee_id=102;`
	...	...
	Transaction *x*	`UPDATE hr.employees SET salary=salary+100 WHERE employee_id=xxx;`

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 another.

Although the default model is to lock at the row level, the Oracle database supports manual locking at higher levels, if needed:

```
SQL> LOCK TABLE employees IN EXCLUSIVE MODE;
Table(s) Locked.
```

With the preceding statement, any other transaction that tries to update a row in the locked table must wait until the transaction that issued the lock request completes. `EXCLUSIVE` is the strictest lock mode. The following are the other lock modes:

- `ROW SHARE`: Permits concurrent access to the locked table, but prohibits sessions from locking the entire table for exclusive access
- `ROW EXCLUSIVE`: Is the same as `ROW SHARE`, but also prohibits locking in `SHARE` mode. The `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`: Is 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 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
```

It is usually not necessary to manually lock objects. The automatic locking mechanism provides the data concurrency needed for most applications.

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:

- **`EXCLUSIVE` row lock for the row or rows being updated**
- **`ROW EXCLUSIVE` table-level lock for the table containing the rows**

DML Locks

Each DML transaction obtains two locks:

- An `EXCLUSIVE` row lock for the row or rows being updated
- A `ROW EXCLUSIVE` 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**

8-26

Enqueue Mechanism

Requests for locks are automatically queued. As soon as the transaction holding a lock is completed, the next session in line receives the lock.

The enqueue mechanism tracks the order in which locks are 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 `SHARE` lock on a table. The session can request to convert the `SHARE` lock to an `EXCLUSIVE` lock. As long as no one else already has an `EXCLUSIVE` or `SHARE` lock on the table, the session holding the `SHARE` lock is granted an `EXCLUSIVE` lock without having to wait in the queue again.

Lock Conflicts

Transaction 1	Time	Transaction 2
`UPDATE employees SET salary=salary+100 WHERE employee_id=100;` `1 row updated.`	9:00:00	`UPDATE employees SET salary=salary+100 WHERE employee_id=101;` `1 row updated.`

`UPDATE employees SET COMMISION_PCT=2 WHERE employee_id=101;`
Session waits enqueued due to lock conflict.
	9:00:05 	`SELECT sum(salary) FROM 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;`

8-27

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 in the slide, 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

Possible Causes of Lock Conflicts

The most common cause of lock conflicts is an uncommitted change, but there are a few other possible causes also:

- **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, they 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 applications with artificially high locking levels so that the Oracle database behaves similarly to these less capable database systems. Developers who are new to Oracle also sometimes unnecessarily code in higher locking levels than required by the Oracle database.

Detecting Lock Conflicts

Select Blocking Sessions from the Performance page.

Blocking Sessions

Page Refreshed **Jun 23, 2005 2:41:04 PM**

View Session | Kill Session

Expand All | Collapse All

Select	Username	Sessions Blocked	Session ID	Session Serial Number	SQL Hash Value	Wait Class	Wait Event	P1	P2	P3	Seconds in Wait
○	▼ Blocking Sessions										
◉	▼ HR	1	130	308	duf40r50uy5gd	Idle	SQL*Net message from client	1413697536	1	0	81
○	HR	0	133	5361	duf40r50uy5gd	Application	enq: TX - row lock contention	1415053318	589840	1672	72

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 then below that, any sessions enqueued for the lock.

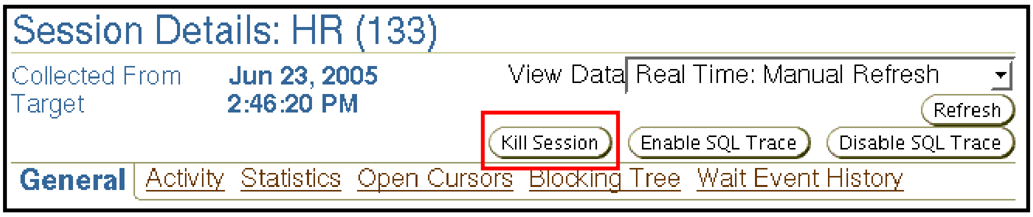
For each session involved in the conflict, you are given the username, session ID, and the number of seconds for which the session has been waiting. Drill down to the session ID to see actual SQL statements that are currently being executed or requested by the session.

The Automatic Database Diagnostic Monitor (ADDM) also automatically detects lock conflicts and can advise you on 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. Remember that when a session is killed, all work within the current transaction is lost (rolled back). A user whose session is killed must log in again and redo all work since the killed session’s last commit.

Users, whose sessions have been killed, receive the following error the next time they try to issue a SQL statement:

```
ORA-03135: connection lost contact
```

Resolving Lock Conflicts Using SQL

SQL statements can be used to determine the blocking session and kill it.

1

```
SQL> select sid, serial#, username
      from v$session where sid in
      (select blocking_session from v$session)
```

Result:

SID	SERIAL#	USERNAME
144	8982	HR

2

```
SQL> alter system kill session '144,8982' immediate;
```


Resolving Lock Conflicts Using SQL

Session manipulation, just like most other tasks performed in Enterprise Manager, can also be done by issuing SQL statements. The `v$session` table contains details of all connected sessions. `blocking_session` is the session ID of the session that is blocking. So, if you query for `SID` and `SERIAL#`, where `SID` matches a blocking session ID, then you have the information needed to perform the `kill session` operation.

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

8-32

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 for the other, neither can complete their transaction to resolve the conflict.

The Oracle database automatically detects deadlocks and terminates the statement with an error. The proper response to that error is either commit or rollback, which releases any other locks in that session so that the other session can continue its transaction.

In the example in the slide, Transaction 1 must either commit or roll back in response to the deadlock detected error. If it commits, it would need to resubmit the second update to complete its transaction. If it performs a rollback, it must resubmit both statements to accomplish its transaction.

Summary

In this lesson, you should have learned how to:

- Manage data through the use of SQL
- Identify and administer PL/SQL objects
- Describe triggers and triggering events
- Monitor and resolve locking conflicts

8-33

Practice Overview: Managing Data and Concurrency

This practice covers the following topics:

- **Identifying locking conflicts**
- **Resolving locking conflicts**

9

Managing Undo Data

Objectives

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

- **Explain DML and undo data generation**
- **Monitor and administer undo data**
- **Describe the difference between undo data and redo data**
- **Configure undo retention**
- **Guarantee undo retention**
- **Use the Undo Advisor**

9-2

Data Manipulation

- Data manipulation language (DML) consists of the following SQL statements:
  - `INSERT`
  - `UPDATE`
  - `DELETE`
  - `MERGE`
- DML always executes as part of a transaction, which can be:
  - Rolled back, using the `ROLLBACK` command
  - Committed, using the `COMMIT` command

Data Manipulation

Data is manipulated, or modified, by the DML class of SQL statements: `INSERT`, `UPDATE`, `DELETE`, and `MERGE`. These statements execute as part of a transaction, which starts with the first successful DML statement and ends with either a `COMMIT` or `ROLLBACK` command. A transaction is either entirely committed or entirely rolled back.

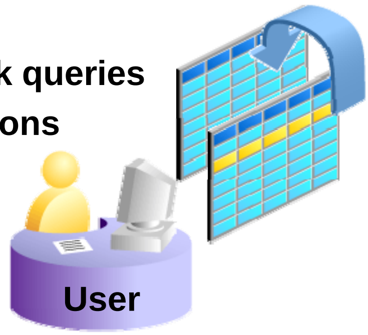
Rollback may also occur if there is a process or system failure.

Note: The `MERGE` command performs a combination of inserts and updates to merge data from one table into another. It is covered in the lesson titled “Managing Data and Concurrency.”

Undo Data

Undo data is:

- A copy of original, premodified 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



Undo Data

The Oracle database 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 enables you to roll back your uncommitted data. Undo also supports read-consistent and flashback queries.

Read-consistent queries provide results that are 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, the Oracle database can reconstruct data to satisfy read-consistent queries.

Flashback queries are queries that purposely ask for a version of the data as it existed at some time in the past. As long as undo information for that past time still exists, 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 because of network errors or a failure on the client computer) before the user decides to commit or roll back the transaction. Failed transactions may also occur when the instance crashes.

Undo Data (continued)

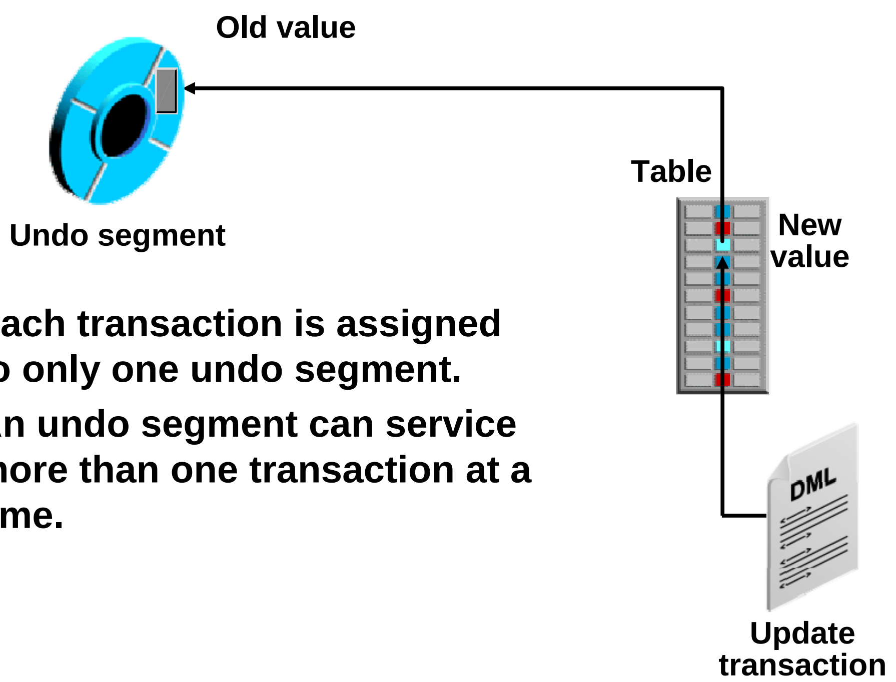
In case of a failed transaction, the safest behavior is chosen, and the Oracle database reverses all changes made by a user, restoring the original data.

Undo information is retained for all transactions, at least until the transaction is ended by:

- Users undoing the transaction (rolls back)
- Users ending a transaction (commits)
- User session abnormally terminating (rolls back)
- User session normally terminating with an exit (commits)

The amount of undo data that is retained and the time for which it is retained depend on the amount of database activity and the database configuration.

Transactions and Undo Data



- Each transaction is assigned to only one undo segment.
- An undo segment can service more than one transaction at a time.

ORACLE

9-6

Transactions and Undo Data

When a transaction starts, it is assigned to an undo segment. Throughout the life of the transaction, when data is changed, the original (before the change) values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the `v$transaction` dynamic performance view.

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.

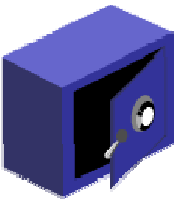
Transactions fill extents in their undo segments until a transaction is completed or all space is consumed. If an extent fills up and more space is needed, the transaction acquires that space from the next extent in the segment. After all extents have been consumed, the transaction either wraps around back into the first extent or requests a new extent to be allocated to the undo segment.

Note: Parallel DML operations can actually cause a transaction to use more than one undo segment. To learn more about parallel DML execution, see the *Oracle Database Administrator's Guide 10g*.

Storing Undo Information

Undo information is stored in undo segments, which are, in turn, stored in an undo tablespace. Undo tablespaces:

- **Are used only for undo segments**
- **Have special recovery considerations**
- **May be associated with only a single instance**
- **Require that only one of them be the current writable undo tablespace for a given instance at any given time**



Storing Undo Information

Undo segments can exist only in a specialized form of tablespace called an undo tablespace. Although a database may have many undo tablespaces, only one of them at a time can be designated as the current one to which undo data is written.

Undo segments are always owned by SYS. Because the segments act as a circular buffer, each segment has 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-KB 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 those that may occur when an instance crashes), undo tablespaces can be recovered only while the instance is in the `MOUNT` state. Recovery considerations for undo tablespaces are covered in the lesson titled “Performing Database Recovery.”

Undo Data Versus Redo Data

	Undo	Redo
Record of	How to undo a change	How to reproduce a change
Used for	Rollback, read-consistency, flashback	Rolling forward database changes
Stored in	Undo segments	Redo log files
Protects against	Inconsistent reads in multiuser systems	Data loss

ORACLE

9-8

Undo Data Versus Redo Data

Undo data and redo data seem similar at first, but they serve different purposes. Undo data is needed in case there is the need to undo a change, and this occurs for read-consistency and rollback. Redo data is needed in case there is the need to perform the changes again, in case they are lost for some reason.

The process of committing entails a verification that the changes in the transaction have been written to the redo log file, which is persistent storage on the disk, as opposed to memory. In addition, it is typically multiplexed. So, there are multiple copies of the redo data on the disk. Even though the changes may not have yet been written to the data files where the table’s blocks are actually stored, guaranteeing that the changes have been written to the redo log file is enough.

A power outage that occurs just before committed changes have been reflected into the data files does not cause a problem because the transaction has been committed. So, when the system starts up again, it is able to roll forward any redo records that are not yet reflected in data files at the time of the outage.

Monitoring Undo

Undo usually requires little management. The areas to monitor include:

- **Free space in an undo tablespace**
- **“Snapshot too old” errors**

9-9

Monitoring Undo

Most of the time, undo is managed automatically by the instance with little need for database administrator (DBA) intervention. A few things that may require administrator involvement include:

- Insufficient space for undo
- Users receiving the `ORA-01555 snapshot too old` error messages

Undo information is always retained until a transaction ends. This 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 being committed, the undo tablespace must be equally large to contain the original data. Imagine a case where a 50-GB table had all rows deleted with the following 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 this 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
```

Proactive monitoring detects space problems in an undo tablespace before they affect users.

Monitoring Undo (continued)

Another problem that the administrator may encounter with undo information is when a query needs to access undo information that has already been overwritten. This may 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
```

This can happen because the Oracle database presents the user with a consistent view of the data as it exists at the time the query starts running. If there are uncommitted changes to the table being queried, the Oracle database reads the undo data to get the committed version of data. This is read consistency. If the query runs so long that in the meantime those modifications are indeed committed, and subsequently their undo data is released and overwritten, then the long-running query no longer can see a consistent view of the data as of when it first began to run. For this reason, undo retention should be configured to accommodate the longest-running query.

Administering Undo

Administration of undo should include preventing:

- **Space errors in an undo tablespace:**
  - **Size the undo tablespace properly.**
  - **Ensure that large transactions commit periodically.**
- **“Snapshot too old” errors:**
  - **Configure an appropriate undo retention interval.**
  - **Size the undo tablespace properly.**
  - **Consider guaranteeing undo retention.**

Use automatic undo management:

```
UNDO_MANAGEMENT=AUTO
UNDO_TABLESPACE=UNDOTBS1
```

9-11

Administering Undo

It is recommended that you use automatic undo management, configured by setting the `UNDO_MANAGEMENT` initialization parameter to `AUTO`. Manual undo management is supported for backward compatibility with Oracle8*i* and earlier versions, but requires more DBA interaction.

With automatic undo management, the DBA manages undo at the tablespace level, controlling, with the `UNDO_TABLESPACE` initialization parameter, which undo tablespace an instance uses. After selecting the undo tablespace, the administrator need worry only 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 to handle transactions (In manual mode, undo segments are known as rollback segments.)
- Choosing a tablespace to contain the rollback segments (Undo tablespaces are used only with automatic undo management.)

Configuring Undo Retention

`UNDO_RETENTION` **specifies (in seconds) how long already committed undo information is to be retained. The only time you must set this parameter is when:**

- **The undo tablespace has the `AUTOEXTEND` option enabled**
- **You want to set undo retention for LOBs**
- **You want to guarantee retention**

Configuring Undo Retention

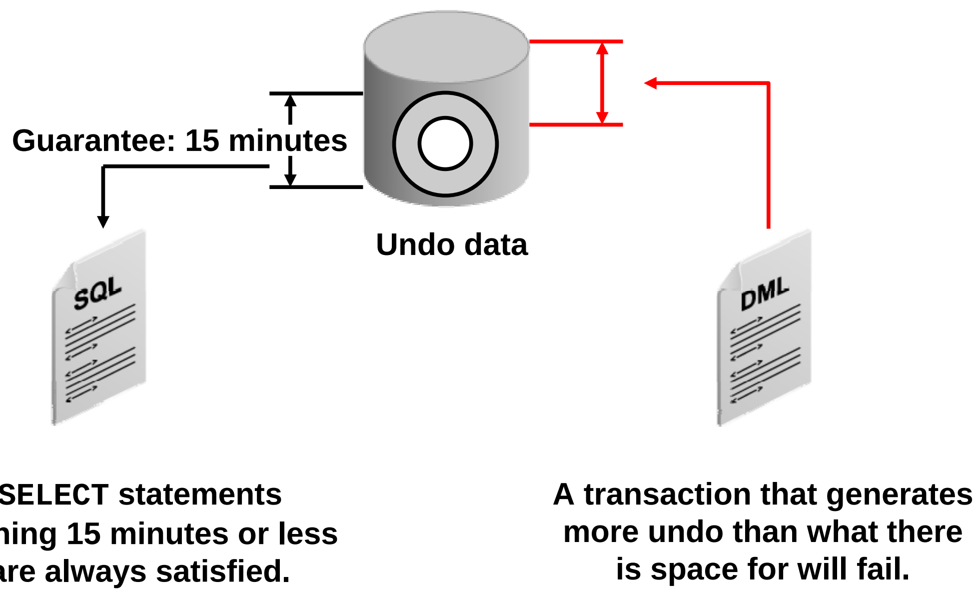
`UNDO_RETENTION` specifies (in seconds) the low threshold value of undo retention. For the `AUTOEXTEND` undo tablespaces, the system retains undo for at least the time specified in this parameter, and automatically tunes the undo retention period to meet the undo requirements of the queries. For fixed-size undo tablespaces, the system automatically tunes for the maximum possible undo retention period on the basis of undo tablespace size and usage history; it ignores `UNDO_RETENTION` unless retention guarantee is enabled. So for automatic undo management, for the three cases listed, the `UNDO_RETENTION` setting is used. In cases other than these three, this parameter is ignored.

Configuring Undo Retention (continued)

Undo information is divided into three categories:

- **Uncommitted undo information:** Supports a currently running transaction, and it is required if a user wants to roll back or if the transaction has failed. Uncommitted undo information is never overwritten.
- **Committed undo information:** Is no longer needed to support a running transaction, but it is still needed to meet the undo retention interval. It is also known as “unexpired” undo information. Committed undo information is retained when possible without causing an active transaction to fail because of lack of space.
- **Expired undo information:** Is no longer needed to support a running transaction. Expired undo information is overwritten when space is required by an active transaction.

Guaranteeing Undo Retention



Guaranteeing Undo Retention

The default undo behavior is to overwrite committed transactions that have not yet expired rather than to allow an active transaction to fail because of 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:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```

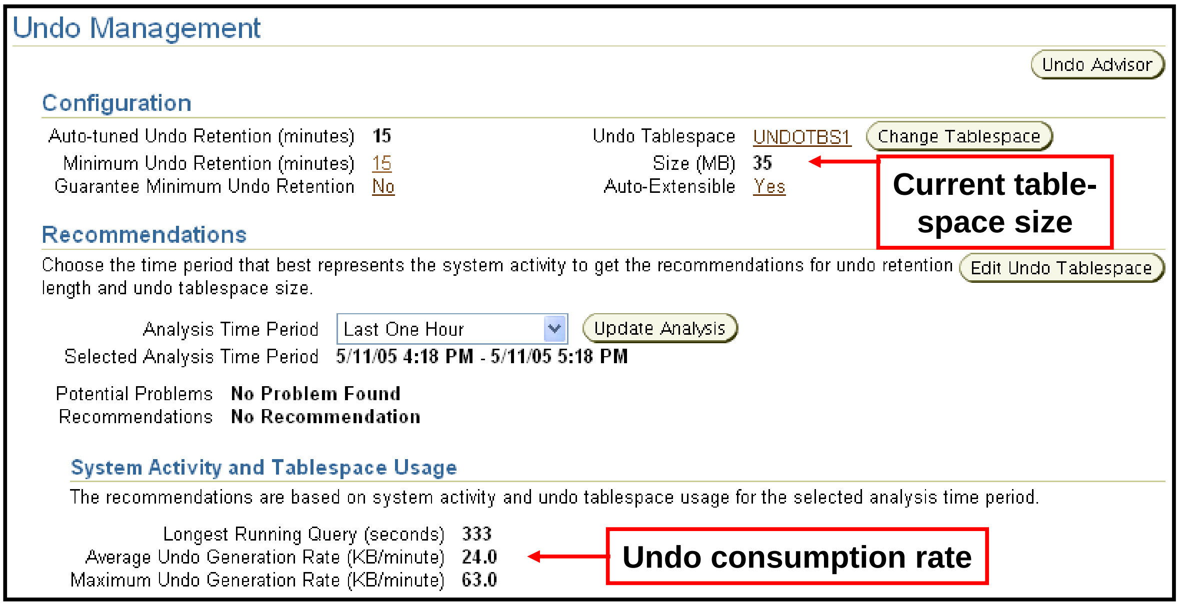
To return a guaranteed undo tablespace to its normal setting, use the following command:

```
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 the following error:

```
SQL> ALTER TABLESPACE example RETENTION GUARANTEE;
ERROR at line 1:
ORA-30044: 'Retention' can only specified for undo
tablespace
```

Sizing the Undo Tablespace



ORACLE

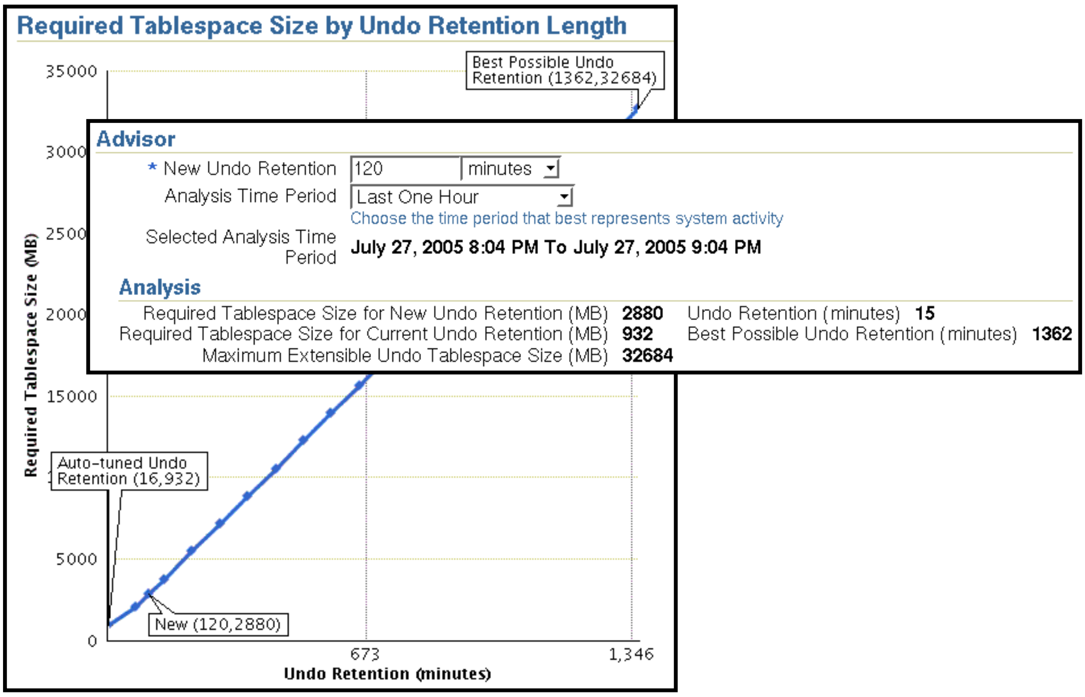
9-15

Sizing the Undo Tablespace

Undo tablespaces must be sized so that they can contain the original information for all transactions. Clicking the Undo Management link on the Enterprise Manager's Administration 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 can automatically extend when they run out of free space. Oracle Corporation recommends that data files that are associated with undo tablespaces, unlike other tablespaces, should 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 must 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



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 the analysis region 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:

- Explain DML and undo data generation
- Monitor and administer undo segments
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Use the Undo Advisor

ORACLE

Practice Overview: Managing Undo Segments

This practice covers the following topics:

- Calculating undo tablespace sizing to support a 48-hour retention interval
- Modifying an undo tablespace to support a 48-hour retention interval

10

Implementing Oracle Database Security

ORACLE

Objectives

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

- **Describe your DBA responsibilities for security**
- **Apply the principle of least privilege**
- **Enable standard database auditing**
- **Specify audit options**
- **Review audit information**
- **Maintain the audit trail**

Objectives

This lesson is a starting point to learn about Oracle Security. Additional information is provided in the following documentation:

- Oracle Database Concepts 10*g* Release 2 (10.2)
- Oracle Database Administrator's Guide 10*g* Release 2 (10.2)
- Oracle Database Security Guide 10*g* Release 2 (10.2)

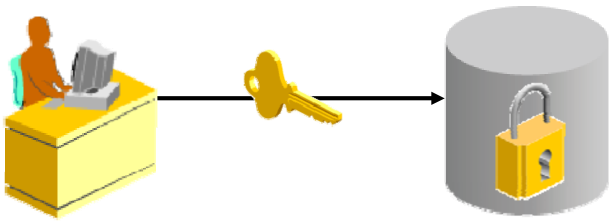
Additional training is provided in the following courses:

- Oracle Database 10*g*: Administration Workshop II (D17092GC30)
- Oracle Database 10*g*: Security (D17499GC10)

Industry Security Requirements

> **Requirements**
> Least Privilege
> Auditing
> Value-based
> FGA
> DBA
> Sec. Updates

- **Legal:**
  - **Sarbanes-Oxley Act (SOX)**
  - **Health Information Portability and Accountability Act (HIPAA)**
  - **California Breach Law**
  - **UK Data Protection Act**
- **Auditing**



ORACLE

Industry Security Requirements

Security requirements have been a matter of individual concern until recently. Unless you were handling government or military data, there were few legal requirements. This is rapidly changing. A variety of laws have been passed to enforce the privacy and accuracy of data. Along with these laws, there is a requirement to audit the security measures that are in place.

Legal: Each of the laws listed here has some specific requirements. This list is representative of many other laws that are being passed worldwide. Of course, security laws vary from place to place.

- **Sarbanes-Oxley Act (SOX)** requires that public companies strengthen and document internal controls to prevent individuals from committing fraudulent acts that may compromise an organization’s financial position or the accuracy of its financial statements. The chief executive officer and the chief financial officer must attest to the adequacy of the internal controls and accuracy of the financial report. These officers are subject to fines and imprisonment for fraudulent reports. The details of SOX include requirements for providing the information that is used to generate the reports, and internal controls that are used to assure the integrity of the financial information.

Industry Security Requirements (continued)

- **Health Information Portability and Accountability Act (HIPAA)** is intended to protect personally identifiable health information from release or misuse. Information holders must provide audit trails of all who access this data.
- **UK Data Protection Act** is intended to protect individual privacy by restricted access to individually identifiable data. It has eight points, one of which requires that data be kept secure.
- **Other laws:**
  - **Family Educational Rights and Privacy Act (FERPA)** covers health and personal information held by schools
  - **California Breach Law** requires that an organization holding a variety of personal identity information (PII) (for example, credit card, driver's license, and government identity numbers) must protect that information. If the information may have been compromised, the organization must notify all individuals involved. There are two laws, CA-SB-1386 and CA-AB-1950, that apply to organizations that hold PII.
  - **Federal Information Security Management Act (FISMA)** is creating security guidance and standards through Federal Information Processing Standard (FIPS) documents that are managed by the National Institute of Standards (NIST). These standards are applied to organizations that are processing information for the U.S. government.

Auditing: Many of these laws include provisions requiring that security plans (internal controls) be audited periodically. SOX requirements are vague and subject to interpretation by the officers of the organization. The implementation details can vary widely, depending on the level of details that the officers require. Because SOX is vague, but the penalties are severe, it is important to protect your company. The cost of security measures must be balanced against the risk. No one will certify that you are 100% secure. A very good solution is industry consensus. If you meet the agreed upon minimum security practices and have accomplished due diligence, then you may be safe from the worst penalties of the law. Some good resources for industry standard practices are SysAdmin, Audit, Network, Security (SANS) Institute, CERT/CC operated by Carnegie Mellon University for the Department of Defense, and the ISO-17799 certification standard:

http://www.sans.org/index.php
http://www.cert.org/nav/index.html
http://www.iso17799software.com/

The ISO-17799 certification standard is an international standard of security practices. It includes best practices, certification, and risk assessment. It covers a broad range of issues and includes prewritten policies.

Separation of Responsibilities

- Users with DBA privileges must be trusted. Consider:
  - Abuse of trust
  - That audit trails protect the trusted position
- DBA responsibilities must be shared.
- Accounts must never be shared.
- The DBA and the system administrator must be different people.
- Separate operator and DBA responsibilities.

10-5

Separation of Responsibilities

These are the main requirements to satisfy separation of duties.

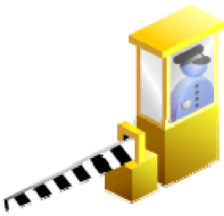
DBAs must be trusted: It is difficult to restrict a DBA. To do his or her job, the DBA requires high-level privileges. A DBA has a position of trust and must be thoroughly vetted. Even a trusted DBA must have accountability. Consider the following:

- **Abuse of trust:** A DBA can potentially misuse the encrypted passwords from the `DBA_USERS` view.
- **Audit trails protect the trusted position:** When auditing is carefully implemented and the guidelines have been followed, the audit trail can show that a particular person has not violated procedures or committed a damaging act. If a malicious user tries to cast suspicion on a trusted user, well-designed audit trails catch the act.

Database Security

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

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



10-6

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 must follow best practices and continually monitor database activity.

Restricting Access to Data and Services

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

Database Security (continued)

Authenticating Users

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 users to provide something that 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 users to provide something that they have, such as a token or public key infrastructure (PKI) certificate. An even stronger form of authentication is to identify users through a unique biometric characteristic such as a fingerprint, iris scan, bone structure patterns, and so on. The Oracle database supports advanced authentication techniques, such as token-, biometric-, and certificate-based identification, through the Advanced Security Option. User accounts that are not in use must be locked to prevent attempts to compromise authentication.

Monitoring for Suspicious Activity

Even authorized and 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. The Oracle database provides a rich set of auditing tools to track user activity and identify suspicious trends.

Principle of Least Privilege

- Install only required software on the machine.
- Activate only required services on the machine.
- Give OS and database access to only those users that require access.
- Limit access to the root or administrator account.
- Limit access to the SYSDBA and SYSOPER accounts.
- Limit users' access to only the database objects required to do their jobs.

Principle of Least Privilege

Apply the principle of least privilege starting at the lowest levels, and continue at every level. There are always new security exploits that cannot be anticipated. By applying this principle, the possibility of the exploit is reduced and the damage may be contained.

- **Install only required software on the machine:** By reducing the number of software packages, you reduce maintenance, upgrades, the possibility of security holes, and software conflicts.
- **Activate only required services on the machine:** Fewer services imply fewer open ports and fewer attack vectors.
- **Give operating system (OS) and database access to only those users that require access:** Fewer users mean fewer passwords and accounts. This reduces the possibility of open or stale accounts. Fewer accounts make it easier for the administrator to keep the accounts current.
- **Limit access to the root or administrator account:** The administrator account must be carefully guarded, audited, and never shared.
- **Limit access to the SYSDBA and SYSOPER accounts:** Users who require access to these roles must each have their own account and be audited.
- **Limit users' access to only the database objects required to do their jobs:** Users who have access to more objects and services than they require have an opportunity for mischief.

Applying the Principle of Least Privilege

- Protect the data dictionary:

```
O7_DICTIONARY_ACCESSIBILITY=FALSE
```

- Revoke unnecessary privileges from `PUBLIC`:

```
REVOKE EXECUTE ON UTL_SMTP, UTL_TCP, UTL_HTTP,
UTL_FILE FROM PUBLIC;
```

- Restrict the directories accessible by users.
- Limit users with administrative privileges.
- Restrict remote database authentication:

```
REMOTE_OS_AUTHENT=FALSE
```


Applying the Principle of Least Privilege

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

Protect the data dictionary: The `O7_DICTIONARY_ACCESSIBILITY` parameter is set by default to `FALSE`. You must not allow this to be changed without a very good reason because it prevents users with the `ANY TABLE` system privileges to access the data dictionary base tables. It also ensures that the `SYS` user can log in only as `SYSDBA`.

Revoke unnecessary privileges from `PUBLIC`: The following packages are extremely useful to applications that need them, but require proper configuration to be used securely. Revoke the `EXECUTE` privilege from `PUBLIC` and grant it to roles when required for the following packages: `UTL_SMTP`, `UTL_TCP`, `UTL_HTTP`, and `UTL_FILE`.

Applying the Principle of Least Privilege (continued)

The more powerful packages that may potentially be misused include:

- **`UTL_SMTP`:** Permits arbitrary e-mail messages to be sent by using the database as a Simple Mail Transfer Protocol (SMTP) mail server. Granting this package to `PUBLIC` may permit unauthorized exchange of e-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.
- **`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.

Restrict access to OS directories: The `DIRECTORY` object inside the database enables DBAs to map directories to OS paths and to grant privileges on those directories to individual users.

Limit users with administrative privileges: Do not provide database users more privileges than necessary. Nonadministrators must not be granted the DBA role. 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`

Restrict remote database authentication: The `REMOTE_OS_AUTHENT` parameter is set to `FALSE` by default. It must not be changed unless all clients can be trusted to authenticate users appropriately.

In the remote authentication process:

- The database user is authenticated externally
- The remote system authenticates the user
- The user logs in to the database without further authentication

Monitoring for Suspicious Activity

Requirements
Least Privilege
> Auditing
Value-based
FGA
DBA
Sec. Updates

Monitoring or auditing must be an integral part of your security procedures. Review the following:

- **Mandatory auditing**
- **Standard database auditing**
- **Value-based auditing**
- **Fine-grained auditing (FGA)**
- **DBA auditing**

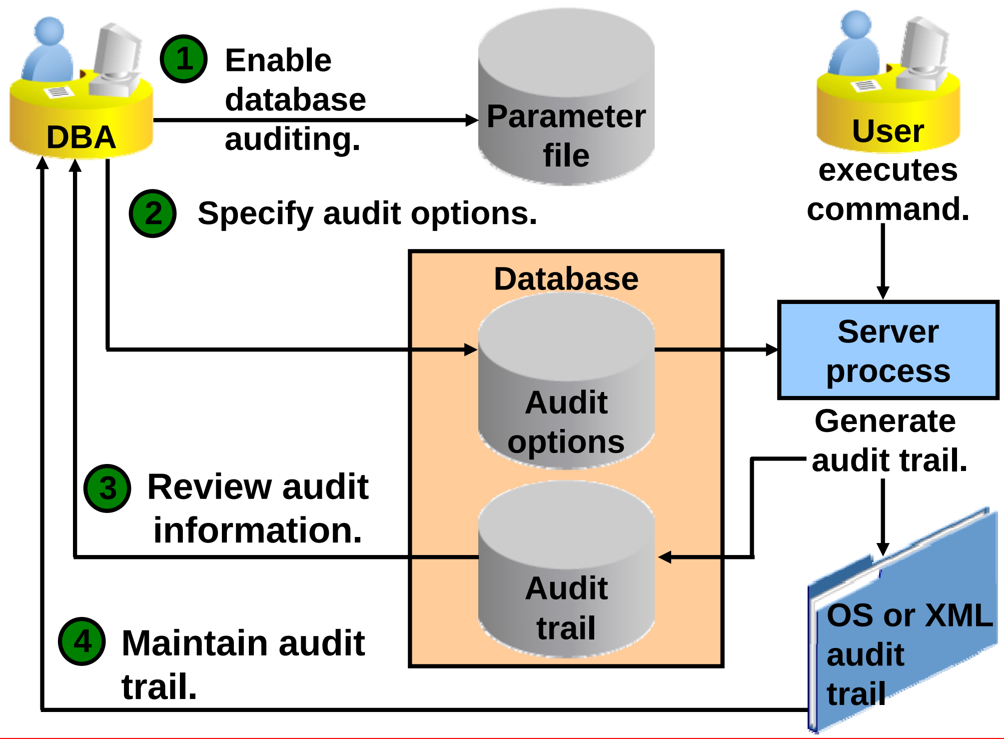
10-11

Monitoring for Suspicious Activity

Auditing, which means capturing and storing information about what is happening in the system, increases the amount of work the system must do. Auditing must 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.

- **Mandatory auditing:** All Oracle databases audit certain actions regardless of other audit options or parameters. The reason for mandatory audit logs is that the database needs to record some database activities, such as system startup and shutdown.
- **Standard database auditing:** This is set at the system level by using the `AUDIT_TRAIL` initialization parameter. After you enable auditing, select the objects and privileges that you want to audit.
- **Value-based auditing:** It extends standard database auditing, capturing not only the audited event that has occurred but also the actual values that have been inserted, updated, or deleted. Value-based auditing is implemented through database triggers.
- **Fine-grained auditing (FGA):** FGA extends standard database auditing, capturing the actual SQL statement that has been issued rather than only that the event has occurred.
- **DBA auditing:** Separate the auditing duties between the DBA and an auditor or security administrator who monitors the DBA activities in an operating system audit trail.

Standard Database Auditing



ORACLE

10-12

Standard Database Auditing

After you have enabled database auditing and specified auditing options (login events, exercise of system and object privileges, or the use of SQL statements), the database begins collecting audit information.

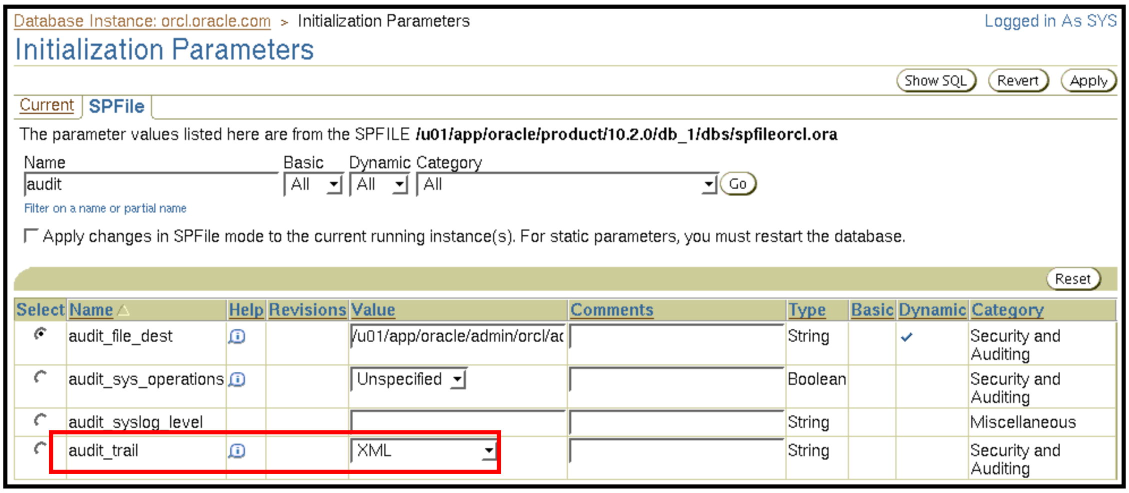
If `AUDIT_TRAIL` is set to `OS`, the audit records are stored in the operating system's audit system. In a Windows environment, this is the event log. In a UNIX or Linux environment, audit records are stored in a file. The location of that file is specified with the `AUDIT_FILE_DEST` parameter.

Assuming that the `AUDIT_TRAIL` parameter is set to `DB`, you can review audit records in the `DBA_AUDIT_TRAIL` view, which is part of the `SYS` schema.

If `AUDIT_TRAIL` is set to `XML` or `XML,EXTENDED`, the audit records are written to XML files in the directory, to which the `AUDIT_FILE_DEST` parameter points. The `V$XML_AUDIT_TRAIL` view allows you to view all the XML files in this directory.

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.

Enabling Auditing




```
ALTER SYSTEM SET audit_trail="XML" SCOPE=SPFILE;
```

Restart database after modifying a static initialization parameter.

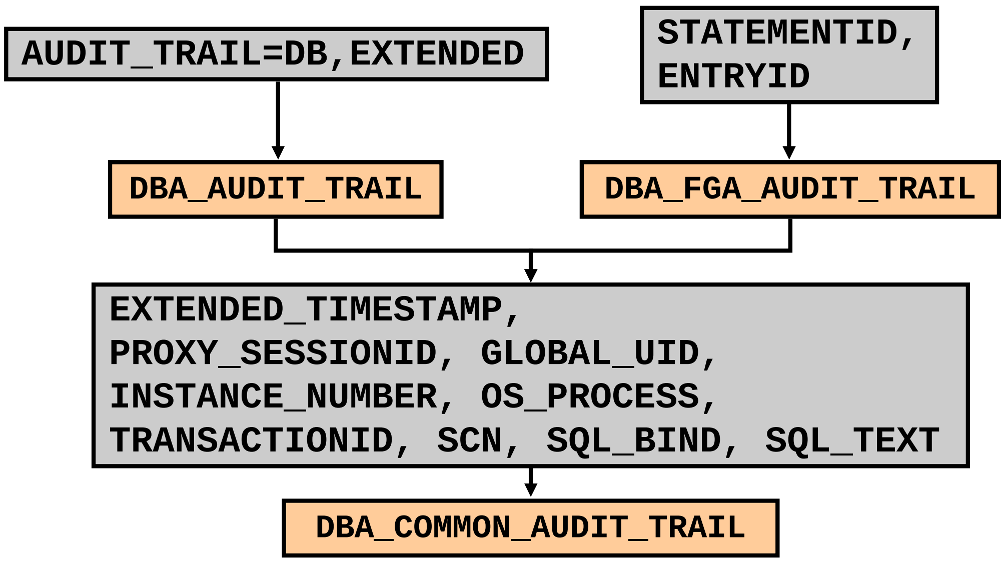
10-13

Enabling Auditing

You must enable database auditing before you specify audit settings.

Uniform Audit Trails

Use `AUDIT_TRAIL` to enable database auditing



Uniform Audit Trails

To use database auditing, you must first set the nondynamic `AUDIT_TRAIL` parameter to point to a storage location for audit records. This enables database auditing.

Oracle Database 10*g* tracks the same fields for standard and fine-grained auditing. This enables you to easily analyze database activities. To accomplish this, both the standard audit trail and the fine-grained audit trail have attributes to complement each other.

The extra information that is collected by standard auditing includes:

- The system change number (SCN), which records every change to the system
- The exact SQL text executed by the user and the bind variables used with the SQL text. These columns appear only if you have specified `AUDIT_TRAIL=DB,EXTENDED` in your initialization parameter file.

The extra information that is collected by fine-grained auditing includes:

- A serial number for each audit record
- A statement number that links multiple audit entries that originate from a single statement

Common attributes include:

- A global time stamp in Universal Time Coordinates (UTC). This field is useful for monitoring across servers in separate geographic locations and time zones.
- An instance number that is unique for each Real Application Clusters (RAC) instance
- A transaction identifier that helps you group audit records of a single transaction

The `DBA_COMMON_AUDIT_TRAIL` view combines standard and fine-grained audit log records.

Enterprise Manager Audit Page

Enterprise Manager Audit Page

You can reach the Audit page from the Database Control Home page by clicking the Administration tab, and then the Audit Settings link in the Users & Privileges region.

The Audit page contains the following regions:

- **Configuration:** Shows the current configuration parameter values and contains links to edit the parameter values
- **Audit Trails:** Provides an easy-to-use access to the audit information, which has been collected

Use these tabbed pages to set and unset audit options:

- **Audited Privileges:** Shows the privileges that are audited
- **Audited Objects:** Shows the objects that are audited
- **Audited Statements:** Shows the statements that are audited

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;
```


Specifying Audit Options

SQL statement auditing: The statement shown in the slide can audit any data definition language (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 or failure:

```
SQL> AUDIT TABLE BY hr WHENEVER NOT SUCCESSFUL;
```

System-privilege auditing: It can be used to audit the exercise of any system privilege (such as `DROP ANY TABLE`). It can be focused by username or by success or failure. By default, auditing is `BY ACCESS`. Each time an audited system privilege is exercised, an audit record is generated. You can choose to group those records with the `BY SESSION` clause so that only one record is generated per session. (This way, if a user updates 100,000 records in a table belonging to another user, you gather only one audit record.) Consider using the `BY SESSION` clause to limit the performance and storage impact of system-privilege auditing.

Object-privilege auditing: It 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 or 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.

Using and Maintaining Audit Information

Audited Objects

Filter Result | Return

▼Hide SQL

```
SELECT "OBJECT_SCHEMA", "OBJECT_NAME", "DB_USER", "STATEMENT_TYPE",
"EXTENDED_TIMESTAMP" FROM SYS.DBA_COMMON_AUDIT_TRAIL WHERE (action between 1 and 16) or
(action between 19 and 29) or (action between 32 and 41) or (action = 43) or (action between 51 and 99) or
(action = 103) or (action between 110 and 113) or (action between 116 and 121) or (action between 123 and 128)
or (action between 160 and 162)
```

Schema	Object Name	User Name	Action	Time (In Session's Time Zone)
HR	JOBS	AUDIT_USER	SESSION REC	2005-10-21 17.52.33.783793000 -7:0
HR	JOBS	HR	SESSION REC	2005-10-21 17.52.34.147582000 -7:0

Disable audit options if you are not using them.

Confirmation

No | Yes

Are you sure you want to remove the 3 selected audited objects?

The audited statements you remove will no longer be audited on the objects.

▼Hide SQL

```
NOAUDIT DELETE ON HR.JOBS
NOAUDIT INSERT ON HR.JOBS
NOAUDIT UPDATE ON HR.JOBS
```

```
ALTER SYSTEM SET audit_trail = "NONE" SCOPE=SPFILE
```

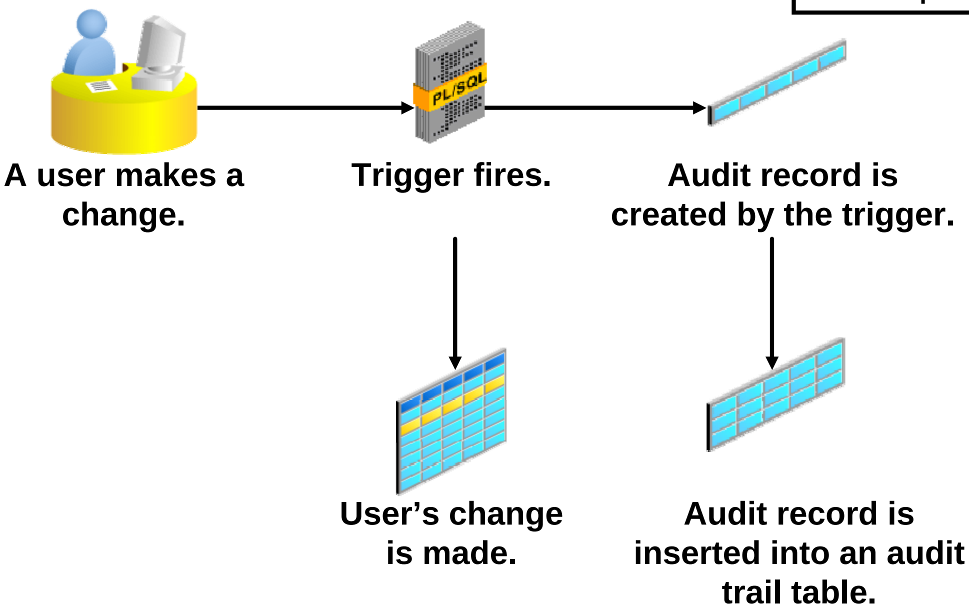
ORACLE

Using and Maintaining Audit Information

Best Practice Tip: Because auditing adds to your system load, disable whatever you are not using.

Value-Based Auditing

Requirements
Least Privilege
Auditing
> **Value-based**
FGA
DBA
Sec. Updates



ORACLE

Value-Based Auditing

Database auditing records the inserts, updates, and deletes that have occurred in audited objects, but does not capture the actual values that are changed. Value-based auditing extends database auditing by capturing the actual values that are 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 that is 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 must 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 as follows:

```
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 includes the username, the IP address from which the change is made, the primary key identifying which record is changed, and the actual salary values that are 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 login triggers, the administrator can capture data that identifies the user who is connecting to the database. Examples include the following:

- IP address of the person logging in
- The first 48 characters of the program name that is used to connect to the instance
- Terminal name that is used to connect to the instance

For a complete list of user parameters, see the section titled “`SYS_CONTEXT`” in *Oracle Database SQL Reference*.

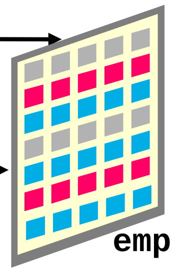
Fine-Grained Auditing

Requirements
Least Privilege
Auditing
Value-based
> FGA
DBA
Sec. Updates

- Monitors data access on the basis of content
- Audits `SELECT`, `INSERT`, `UPDATE`, `DELETE`, and `MERGE`
- Can be linked to a table or view, to one or more columns
- May fire a procedure
- Is administered with the `DBMS_FGA` package

Policy: `AUDIT_EMPS_SALARY`

```
SELECT name, salary
  FROM employees
  WHERE
    department_id = 10;
```



ORACLE

10-20

Fine-Grained Auditing

FGA 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. More than one relevant column is supported for an FGA policy. By default, if any one of these columns is present in the SQL statement, it is audited. `DBMS_FGA.ALL_COLUMNS` and `DBMS_FGA.ANY_COLUMNS` are provided to audit on the basis of whether any or all of the relevant columns are used in the statement.

Use 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 specified 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, and so a `SELECT` statement that returns thousands of rows generates only one audit record.

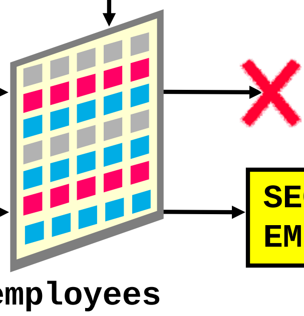
FGA Policy

- **Defines:**
  - **Audit criteria**
  - **Audit action**
- **Is created with `DBMS_FGA .ADD_ POLICY`**

```
dbms_fga.add_policy (
 object_schema    => 'HR',
 object_name      => 'EMPLOYEES',
 policy_name => 'audit_emps_salary',
 audit_condition=> 'department_id=10',
 audit_column     => 'SALARY',
 handler_schema   => 'secure',
 handler_module   => 'log_emps_salary',
 enable           => TRUE,
 statement_types => 'SELECT' );
```

```
SELECT name, job_id
  FROM employees;
```

```
SELECT name, salary
  FROM employees
  WHERE
    department_id = 10;
```



`SECURE.LOG_ EMPS_SALARY`

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` by using the following argument:

```
policy_name => 'audit_emps_salary'
```

Audit Condition

The audit condition is a SQL predicate that defines when the audit event must fire. In the slide example, all rows in department 10 are audited by using the following condition argument:

```
audit_condition => 'department_id = 10'
```

FGA Policy (continued)

Audit Column

The audit column defines the data that is being audited. An audit event occurs if this column is included in the `SELECT` statement or if the audit condition allows the selection. The example in the slide audits two columns by 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 must occur.

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 in the slide audits the `hr.employees` table by 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 must be taken during auditing. For example, the event handler can 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 in the slide executes the `SECURE.LOG_EMPS_SALARY` procedure by using the following arguments:

```
handler_schema => 'secure'
handler_module => 'log_emps_salary'
```

By default, audit trail always writes the SQL text and SQL bind information to LOBs. The default can be changed (for example, if the system would suffer performance degradation).

Status

The status indicates whether the FGA policy is enabled. In the slide example, the following argument enables the policy:

```
enable => TRUE
```

Audited DML Statement: Considerations

- Records are audited if the FGA predicate is satisfied and the relevant columns are referenced.
- `DELETE` statements are audited regardless of any specified columns.
- `MERGE` statements are audited with the underlying `INSERT` or `UPDATE` generated statements.

```
UPDATE hr.employees
SET salary = 10
WHERE department_id = 10;
```

```
UPDATE hr.employees
SET salary = 10
WHERE employee_id = 111;
```


Audited DML Statement: Considerations

With an FGA policy defined for DML statements, a DML statement is audited if data rows (both new and old) being manipulated meet the policy predicate criteria.

However, if relevant columns are also specified in the policy definition, the statement is audited when the data meets the FGA policy predicate and the statement references the relevant columns defined.

For `DELETE` statements, specifying relevant columns during policy definition is not useful because all columns in a table are touched by a `DELETE` statement. Therefore, a `DELETE` statement is always audited regardless of the relevant columns.

`MERGE` statements are supported by FGA. The underlying `INSERT` or `UPDATE` statements are audited if they meet any defined `INSERT` or `UPDATE` FGA policies.

Using the previously defined FGA policy, the first statement is audited whereas the second one is not.

FGA Guidelines

- To audit all statements, use a `null` condition.
- Policy names must be unique.
- The audited table or view must already exist when you create the policy.
- If the audit condition syntax is invalid, an `ORA-28112` error is raised when the audited object is accessed.
- If the audited 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 record is still created.

ORACLE

10-24

FGA Guidelines

For the `SELECT` statements, FGA captures the statement itself, and not the actual rows. However, when FGA is combined with Flashback Query, the rows can be reconstructed as they existed at that point in time.

For more details about Flashback Query, see the lesson titled "Performing Flashback."

For more details about the `DBMS_FGA` package, see *Oracle Database, PL/SQL Packages and Types Reference*.

DBA Auditing

Users with the SYSDBA or SYSOPER privileges can connect when the database is closed.

- **Audit trail must be stored outside the database.**
- **Connections as SYSDBA or SYSOPER are always audited.**
- **You can enable additional auditing of SYSDBA or SYSOPER actions with `audit_sys_operations`.**
- **You can control the audit trail with `audit_file_dest`.**

DBA Auditing

The SYSDBA and SYSOPER users have privileges to start up 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. The Oracle database captures login events by the SYSDBA and SYSOPER users automatically. This provides a valuable way to track authorized or unauthorized SYSDBA and SYSOPER actions, but it is useful only if the OS audit trail is reviewed.

The Oracle database does not capture anything other than the login events unless auditing is specifically enabled. Enable auditing of the SYSDBA and SYSOPER users by setting the initialization parameter:

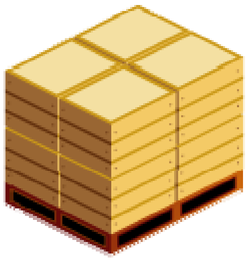
`audit_sys_operations=TRUE` (The default is FALSE.)

If the SYS operations are audited, then the `audit_file_dest` initialization parameter controls the storage location of the audit records. 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`.

Maintaining the Audit Trail

The audit trail should be maintained. Follow these best practice guidelines:

- **Review and store old records.**
- **Prevent storage problems.**
- **Avoid loss of records.**



ORACLE

10-26

Maintaining the Audit Trail

Each type of audit trail must be maintained. Basic maintenance must include reviewing the audit records and removing older records from the database or operating system. Audit trails can grow to fill the available storage. If the file system is full, the system may crash or just cause performance problems. If the database audit trail fills the tablespace, audited actions do not complete. If the audit trail fills the system tablespace, the performance of other operations is affected before audit operations halt.

The audit trail for standard auditing is stored in the `AUD$` table. The audit trail for FGA is the `FGA_LOG$` table. Both these tables are created in the `SYSTEM` tablespace by default. You can move these table to another tablespace by using the export and import utilities.

Note: Moving the audit tables out of the `SYSTEM` tablespace is not supported.

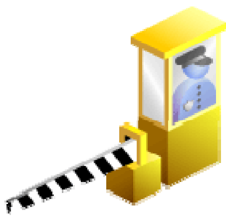
Audit records can be lost during the process of removing records from the audit tables.

Best practice tip: Use an export based on a time stamp, and then delete rows from the audit trail, based on the same time stamp.

Security Updates

- Requirements
- Least Privilege
- Auditing
- Value-based
- FGA
- DBA
- > Sec. Updates

- **Oracle posts security alerts on the Oracle Technology Network Web site at:**
  http://www.oracle.com/technology/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.**



10-27

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. With Enterprise Manager, you can manage your patches. Oracle includes an acknowledgement of the individual or organization that notified it of the vulnerability.

Security alerts are posted on the Oracle Technology Network Web site and on Oracle *MetaLink* (*MetaLink*). 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, notify Oracle by submitting a service request through *MetaLink* or by sending an e-mail to `secalert_us@oracle.com`.

Applying Security Patches

- Use the Critical Patch Update process.
- Apply all security patches and workarounds.
- Contact the Oracle security products team.

ORACLE

10-28

Applying Security Patches

Critical Patch Update (CPU) Process

Oracle initiated the CPU process in January 2005. The process bundles together critical patches on a quarterly basis. This program replaces the Security Alert patch releases. These patches are cumulative, and include commonly requested and required prerequisite patches. The quarterly patch release comes with a risk assessment matrix to enable you to determine for your site the impact and security risks. See *MetaLink* Note: 290738.1 “Oracle Critical Patch Update Program General FAQ.” You must subscribe to *MetaLink* to receive these updates.

Apply All Security Patches and Workarounds

Always apply all relevant and current security patches for both the operating system on which the database resides and the Oracle software, and for all installed options and components.

Contact the Oracle Security Products Team

If you believe that you have found a security vulnerability in the Oracle software, follow the instructions provided from the Security Alerts link at http://otn.oracle.com or at http://www.oracle.com/technology/deploy/security/alerts.htm.

Summary

In this lesson, you should have learned how to:

- Describe your DBA responsibilities for security
- Apply the principle of least privilege
- Enable standard database auditing
- Specify audit options
- Review audit information
- Maintain the audit trail

Practice Overview: Implementing Oracle Database Security

This practice covers the following topics:

- Enabling standard database auditing
- Specifying audit options for the `HR.JOBS` table
- Updating the table
- Reviewing audit information
- Maintaining the audit trail

11

Configuring the Oracle Network Environment

Objectives

After completing this lesson, you should be able to:

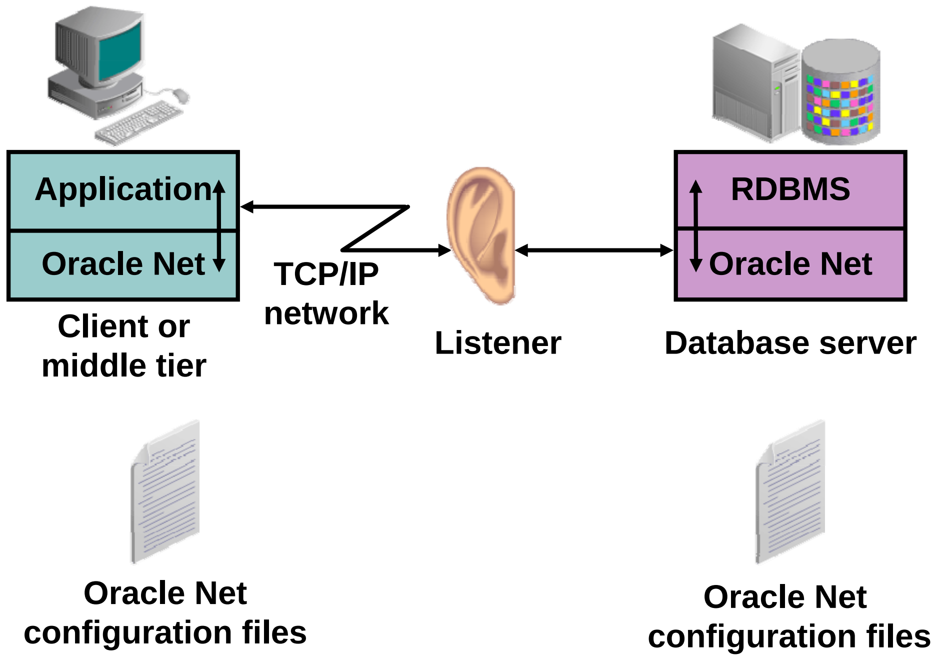
- Use Enterprise Manager to:
  - Create additional listeners
  - Create Oracle Net Service aliases
  - Configure connect-time failover
  - Control the Oracle Net Listener
- Use `tnsping` to test Oracle Net connectivity
- Identify when to use shared servers versus dedicated servers

11-2

Resources

- *Oracle Database, Net Services Administrator's Guide, 10g Release 2 (10.2),* Part No. B14212-01
- *Oracle Database, Net Services Reference, 10g Release 2 (10.2)*, Part No. B14213-01

Oracle Net Services



Oracle Net Services

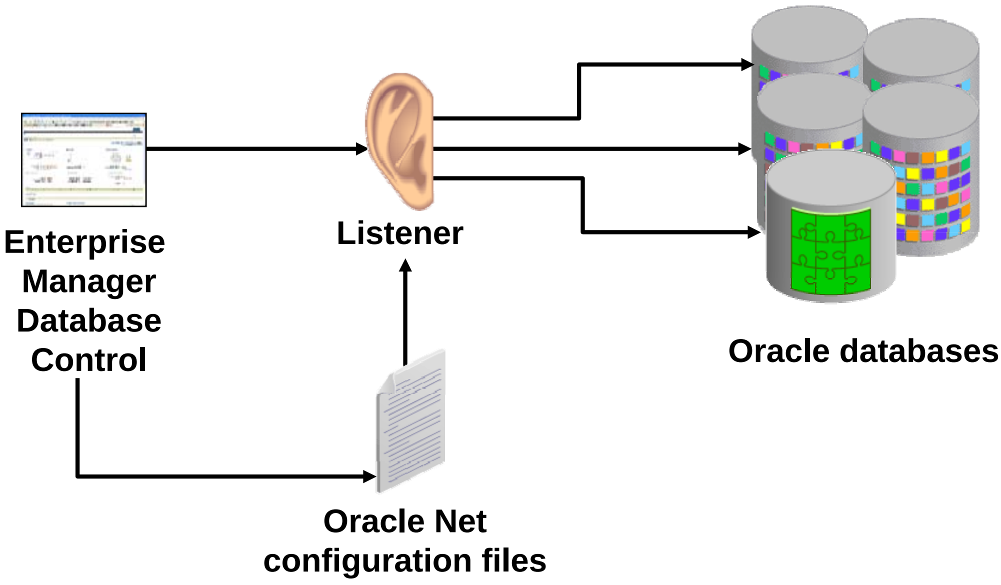
Oracle Net Services enable network connections from a client or middle-tier application to the Oracle server. After a network session is established, Oracle Net acts as the data courier for both the client application and the database server. It is responsible for establishing and maintaining the connection 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 computer, Oracle Net is a background component for application connections to the database.

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.

The most common use of Oracle Net Services is to allow incoming database connections. You can configure additional net services to allow access to external code libraries (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_home>/network/admin/listener.ora`
`sqlnet.ora`

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.

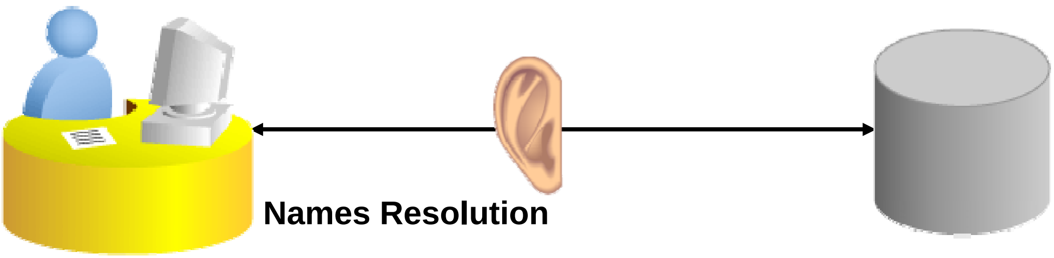
Enterprise Manager is one of the ways to access the listener. You can control the configuration of the actual listener as well as general parameters such as password protection and log file locations.

Advanced administrators can also configure Oracle Net Services by manually editing the configuration files with a standard operating system (OS) text editor, such as `vi` or `gedit`, if necessary.

Establishing 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 that the listener is monitoring
- Protocol that the listener is using
- Name of the service that the listener is handling

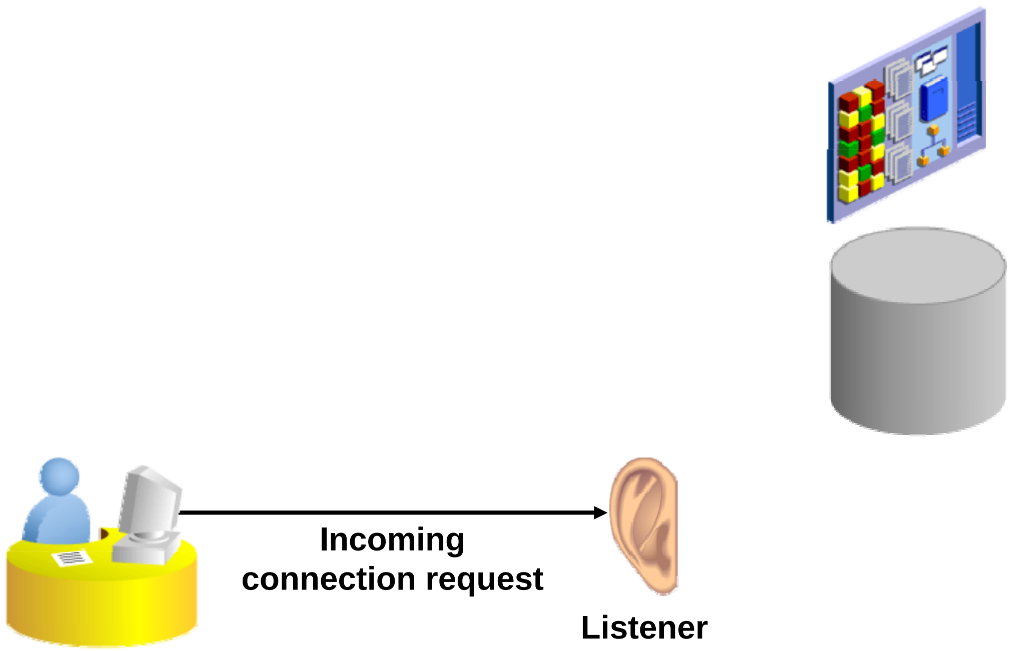


Establishing Net Connections

For an application to connect to a service through an Oracle Net Listener, the application must have information about that service, including the address or host where the listener resides, the protocol that the listener accepts, and the port that the listener monitors. After the listener is located, the final piece of information that the application needs is the name of the service that it wants to connect to.

The process of determining this connection information is called “names resolution”.

Establishing a Connection

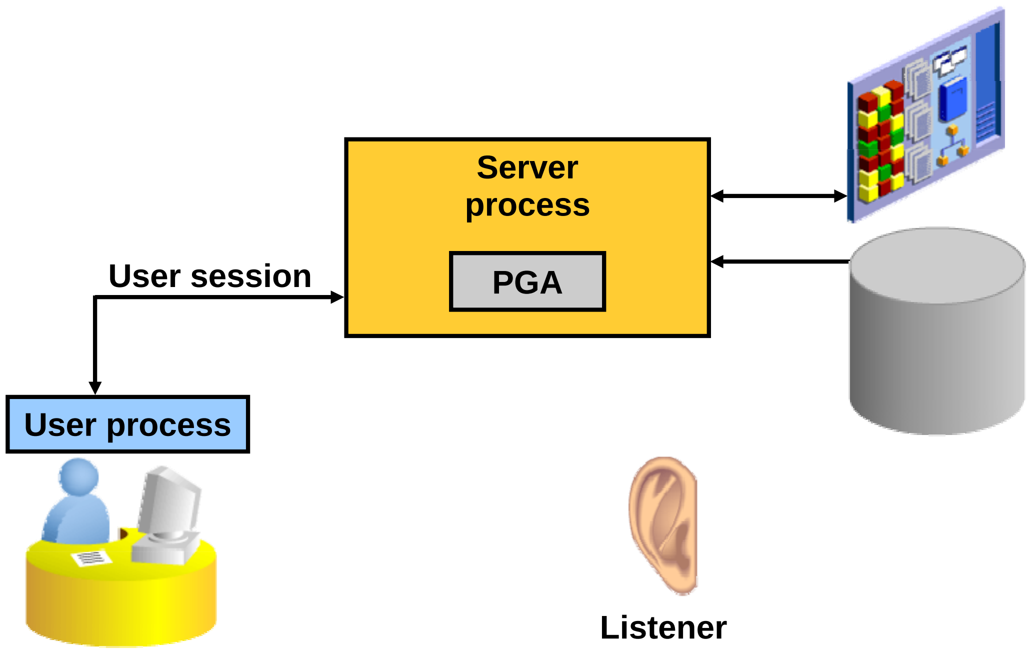


Establishing a Connection

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 onward) to the Oracle Net Listener. The listener receives a `CONNECT` packet and checks 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.

User Sessions



User Sessions

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." The listener connects to the process and passes it the 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 passed on to the server process.

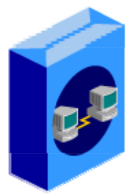
The server process checks the user's authentication credentials (usually a password), and if the credentials are valid, a user session is created.

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 data files on the disk into the database buffer cache portion of the System Global Area (SGA), if the blocks are not already present in the SGA
- Managing all sorting 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
- Reading auditing options and reporting user processes to the audit destination

Tools for Configuring and Managing the Oracle Network

- **Enterprise Manager Net Services Administration page**
- **Oracle Net Manager**
- **Oracle Net Configuration Assistant launched by Oracle Universal Installer**
- **Command line**



11-8

Tools for Configuring and Managing the Oracle Network

You can use any of the following tools or applications to manage your Oracle Network configuration:

- **Enterprise Manager:** Provides an integrated environment for configuring and managing Oracle Net Services. Use Enterprise Manager to configure Oracle Net Services for any Oracle home across multiple file systems and to administer listeners.
- **Oracle Net Manager:** Provides a graphical user interface (GUI) through which you can configure Oracle Net Services for an Oracle home on a local client or a server host
- **Oracle Net Configuration Assistant:** Is launched by Oracle Universal Installer when you install the Oracle software. The Oracle Net Configuration Assistant enables you to configure the listening protocol address and service information for an Oracle database.
- **Command Line:** Is used to start, stop, and view the status of the listener process. It is an OS user (in class, `oracle`) that starts or stops the listener. If the listener is not started, you cannot use Enterprise Manager.

Listener Control Utility

Oracle Net listeners can be controlled with the command-line `lsnrctl` utility (or from EM).

```
$lsnrctl
LSNRCTL for Linux: Version 10.2.0.0.0 on 12-MAY-2005 13:27:51
Copyright (c) 1991, 2004, 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*
```

ORACLE

Listener Control Utility

When an instance starts, a listener process establishes a communication pathway to the Oracle database. The listener is then able to accept database connection requests.

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
- Reinitialize the listener from the configuration file parameters
- Dynamically configure many listeners
- Change the listener password

The basic command syntax for this utility is as follows:

```
LSNRCTL> command [listener_name]
```

When the `lsnrctl` command is issued, the command acts on the default listener (named “`LISTENER`”) unless a different listener name is specified or the `SET CURRENT_LISTENER` command is executed. If the listener name is `LISTENER`, the *`listener_name`* argument can be omitted.

The valid commands for `lsnrctl` are shown in the slide.

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
```


Listener Control Utility Syntax

The `lsnrctl` commands can be issued from within the utility (prompt syntax) or the command line. The following two commands have the identical effect—first, by using the command-line syntax:

```
$ lsnrctl start
```

and then by using the prompt syntax:

```
$ lsnrctl
LSNRCTL for Linux: Version 10.2.0.0.0 on 12-MAY-2005
Copyright (c) 1991, 2004, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> start
```

Usually, the command-line syntax is used to execute an individual command or scripted commands. If you plan to execute several consecutive `lsnrctl` commands, then the prompt syntax is the most efficient. Notice that the `listener_name` argument is omitted, and 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 is named `BACKUP`. Following are two examples of stopping a listener named `BACKUP` using prompt syntax:

```
LSNRCTL> stop backup
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel)(PORT=5521)))
The command completed successfully
```

This produces the same results as follows:

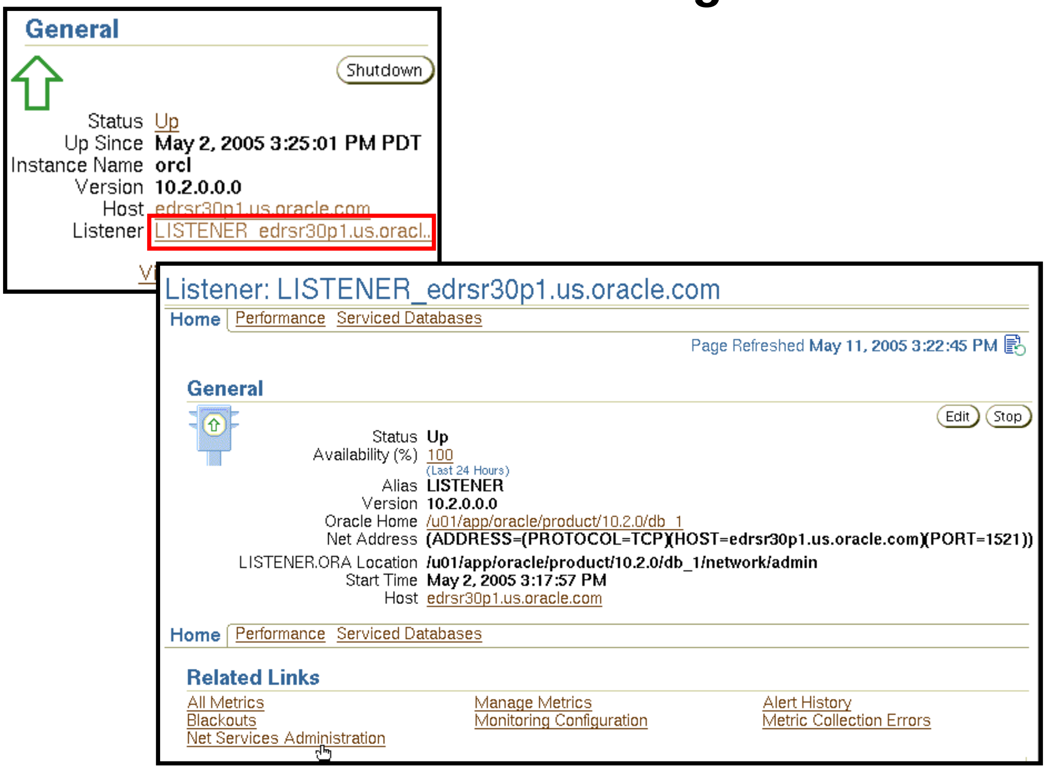
```
LSNRCTL> set cur backup
Current Listener is backup
LSNRCTL> stop
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel)(PORT=5521)))
The command completed successfully
```

Note: In the preceding syntax, `current_listener` can be abbreviated as `cur`.

You can also achieve the same results with the command-line syntax:

```
/home/oracle> lsnrctl stop backup
LSNRCTL for Linux:Version 10.2.0.0.0 on 12-MAY-2005 15:19:33
Copyright (c) 1991, 2004, Oracle. All rights reserved.
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel)(PORT=5521)))
The command completed successfully
```

Listener Home Page



11-12

Listener Home Page

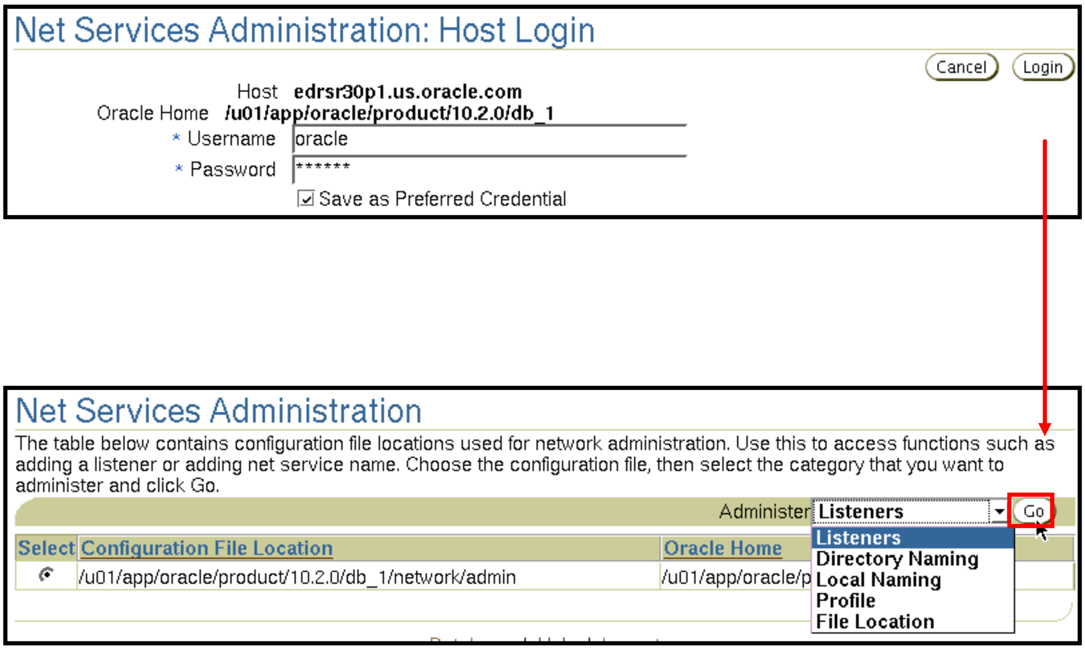
Click the Listener link on the Enterprise Manager Database Home page to access the Listener Home page.

On this page, you can see:

- The listener status and availability within the last 24 hours
- The listener version and Oracle home
- The first listening address for the listener
- The location of the configuration files that are used to start the listener
- The listener start time and host information

To start the listener, go to the Database home page, and click the listener name to open the Listener home page. Click Stop to stop the listener if it is running, or click Start to start the listener if it is not running. Log on to the host as the OS user that can start and stop the listener.

Net Services Administration Pages

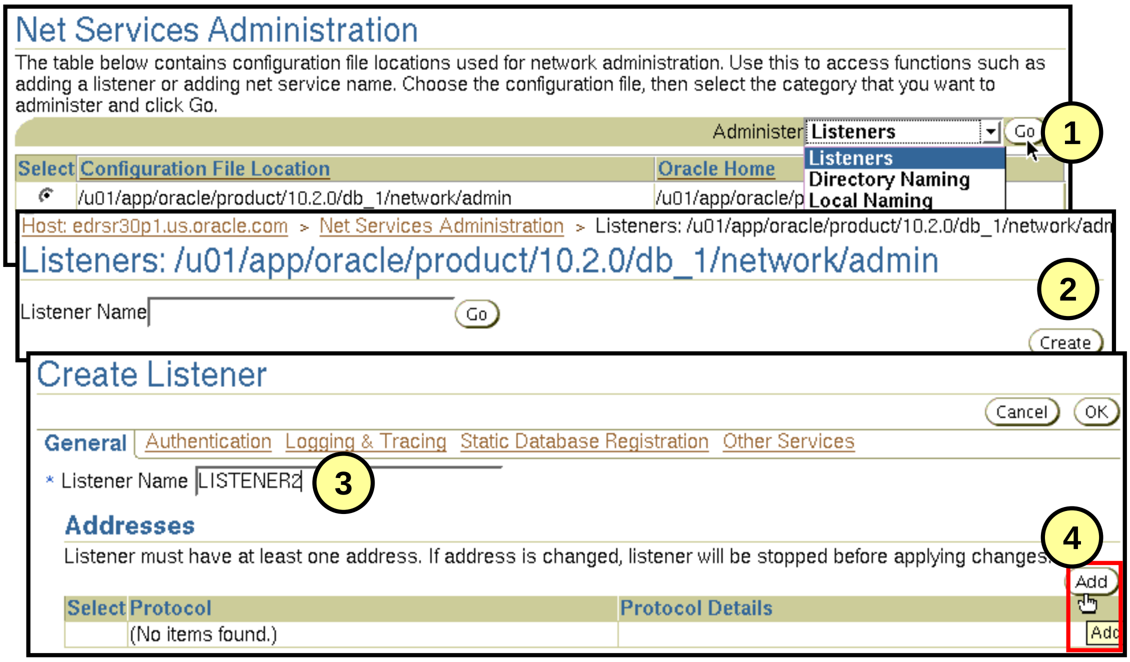


Net Services Administration Pages

The Net Services Administration page enables you to configure Oracle Net Services for any Oracle home across multiple file systems. It also provides common administration functions for listeners. You can use Net Services Administration to configure and administer the following:

- **Listeners:** You can add, remove, start, and stop a listener, as well as change its tracing and logging characteristics. You can also look at a listener's control status report.
- **Directory Naming:** Define simple names and connect identifiers, and map them to connect descriptors to identify the network location and identification of a service. Save database services, Net Services, and Net Service aliases in a centralized directory service.
- **Local Naming:** Save Net Service names in the `tnsnames.ora` file.
- **Profiles:** Configure the `sqlnet.ora` parameters.
- **File Location:** Change the location of the configuration files of Net Services.

Creating a Listener

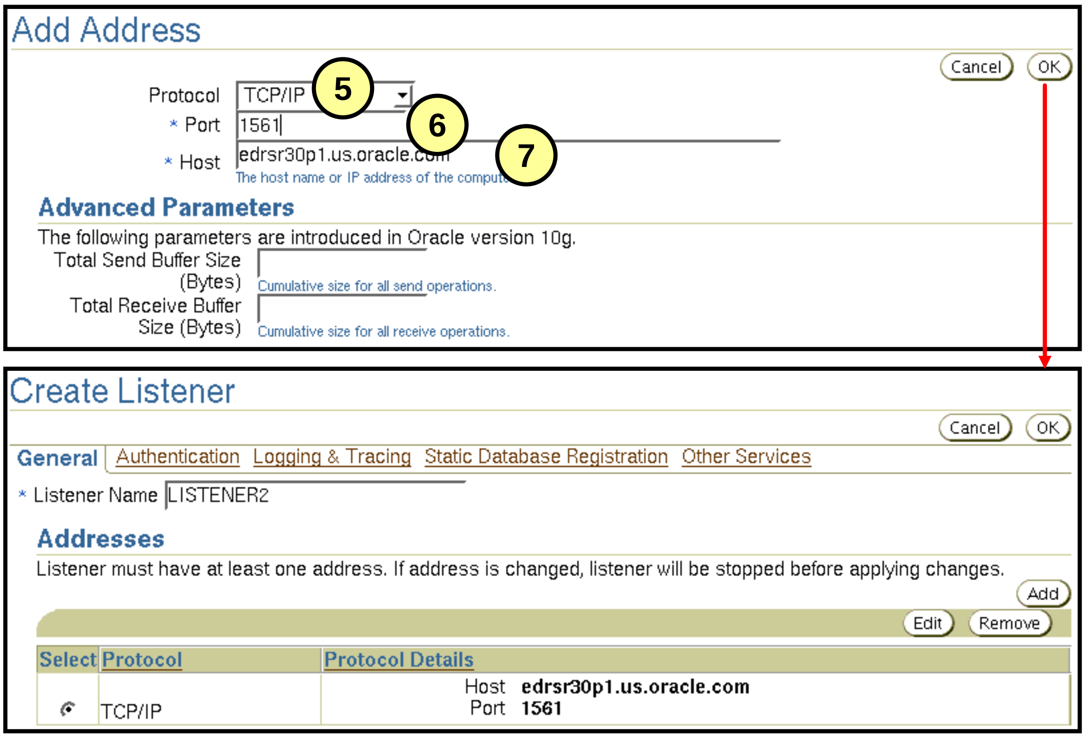


Creating a Listener

To create an Oracle Net Listener, click Net Services Administration in the Related Links region of the Listener properties page. Perform the following steps:

1. Select Listeners from the Administer drop-down list, and click Go.
2. Click Create.
3. Enter a listener name. The name must be unique for this server.
4. Add a listener address. Each listener must have at least one listener address.

Adding Listener Addresses



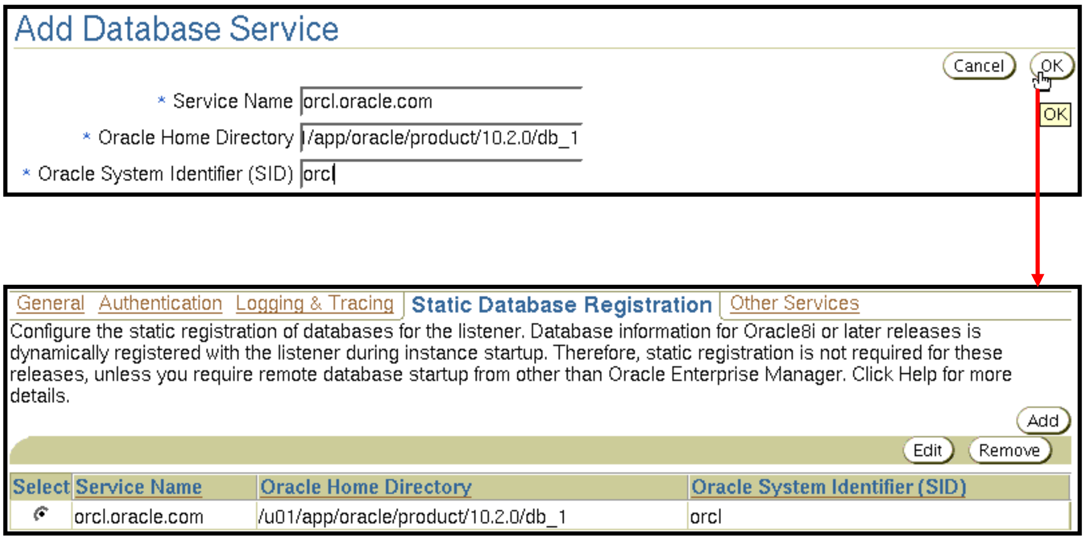
11-15

Adding Listener Addresses

The workflow to create a listener continues:

5. Select the network protocol. TCP/IP is the 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.
   Note: The NMP and EXTPROC protocols are configured by using the Other Services tab.
6. Enter the port that you want the listener to monitor. Oracle Net’s default port is 1521. If you choose to use a port other than 1521, then additional configuration is required for the listener or for the instance.
7. Enter the name or IP address of the server that 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.

Database Service Registration



Database Service 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:** The earlier 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 that 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 want to configure your instance to register with a nondefault port
  - Your application requires static service registration

To add a static database service, click Static Database Registration on the Edit Listener page, and click the Add 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 must reload (use the `RELOAD` command) or restart your listener for the changes to take effect.

Naming Methods

Oracle Net supports several methods of resolving connection information:

- Easy connect naming: Uses a TCP/IP connect string
- Local naming: Uses a local configuration file
- Directory naming: Uses a centralized LDAP-compliant directory server
- External naming: Uses a supported non-Oracle naming service

Naming Methods

Oracle Net provides support for the following naming methods:

- **Easy connect naming:** The easy connect naming method enables clients to connect to an Oracle database server by using a TCP/IP connect string consisting of a host name, and optional port and service name as follows:

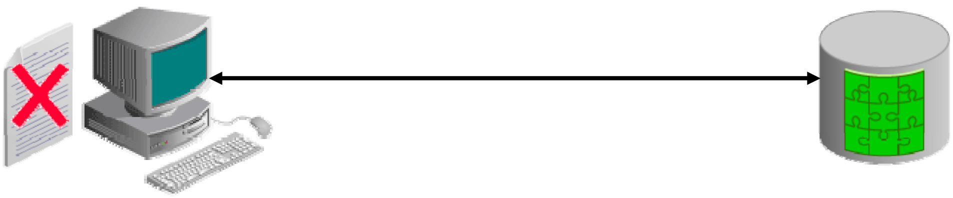
  ```
  CONNECT username/password@host[:port][/service_name]
  ```

  The easy connect naming method requires no configuration.
- **Local naming:** The local naming method stores connect descriptors, identified by their net service name, in a local configuration file named `tnsnames.ora` on the client.
- **Directory naming:** The directory naming method stores connect identifiers in a centralized Lightweight Directory Access Protocol (LDAP)–compliant directory server to access a database service.
- **External naming:** The external naming method stores net service names in a supported non-Oracle naming service. Supported third-party services include:
  - Network Information Service (NIS) External Naming
  - Distributed Computing Environment (DCE) Cell Directory Services (CDS)

Easy Connect

- Is enabled by default
- Requires no client-side configuration
- Supports only TCP/IP (no SSL)
- Offers no support for advanced connection options, such as:
  - Connect-time failover
  - Source routing
  - Load balancing

```
SQL> CONNECT hr/hr@db.us.oracle.com:1521/dba10g
```



No Oracle Net configuration files

Easy Connect

With Easy Connect, you supply 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, Oracle Net assumes that the default port of 1521 is being used. If the service name is not provided, Oracle Net assumes that the database service name and host name provided in the connect string are identical.

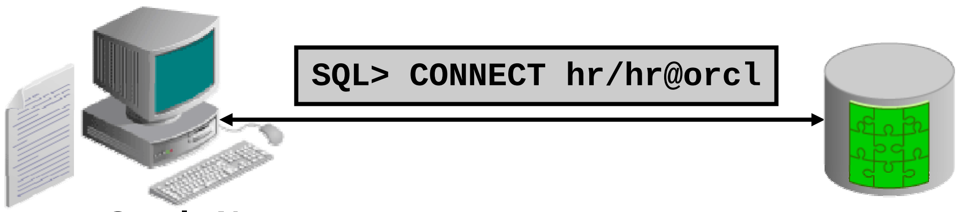
Assuming that the listener uses TCP to listen on port 1521 and the `SERVICE_NAMES=db` and `DB_DOMAIN=us.oracle.com` instance parameters, the connect string shown in the slide can be shortened to:

```
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 must 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, such as:
  - Connect-time failover
  - Source routing
  - Load balancing



Oracle Net
configuration files

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.

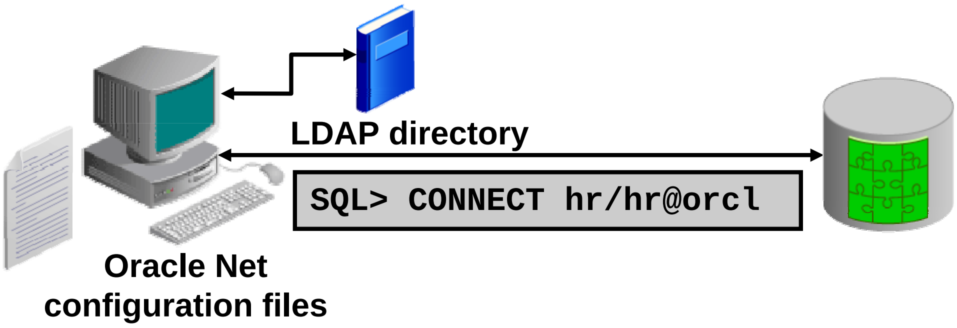
One advantage of local naming is that the database users need to remember only a short alias rather than the long connect string required by Easy Connect.

The local list of known services is stored in the text configuration file: <oracle_home>/network/admin/tnsnames.ora. This is the default location of the `tnsnames.ora` file, but the file can be located elsewhere using the `TNS_ADMIN` environment variable.

Local naming is appropriate for organizations where Oracle Net service configurations do not change often.

Directory Naming

- Requires LDAP with Oracle Net Names Resolution information loaded:
  - Oracle Internet Directory
  - Microsoft Active Directory Services
- Supports all Oracle Net protocols
- Supports advanced connection options



Directory Naming

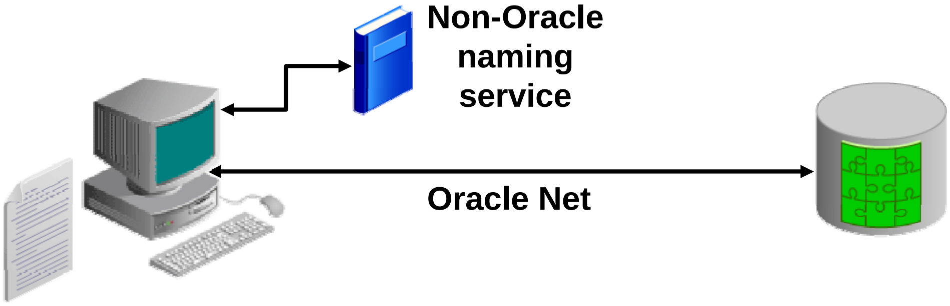
With directory naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against an external list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name. Like local naming, database users need to remember only a short alias.

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 database administrator (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.

External Naming Method

- Uses a supported non-Oracle naming service
- Includes:
  - Network Information Service (NIS) External Naming
  - Distributed Computing Environment (DCE) Cell Directory Services (CDS)



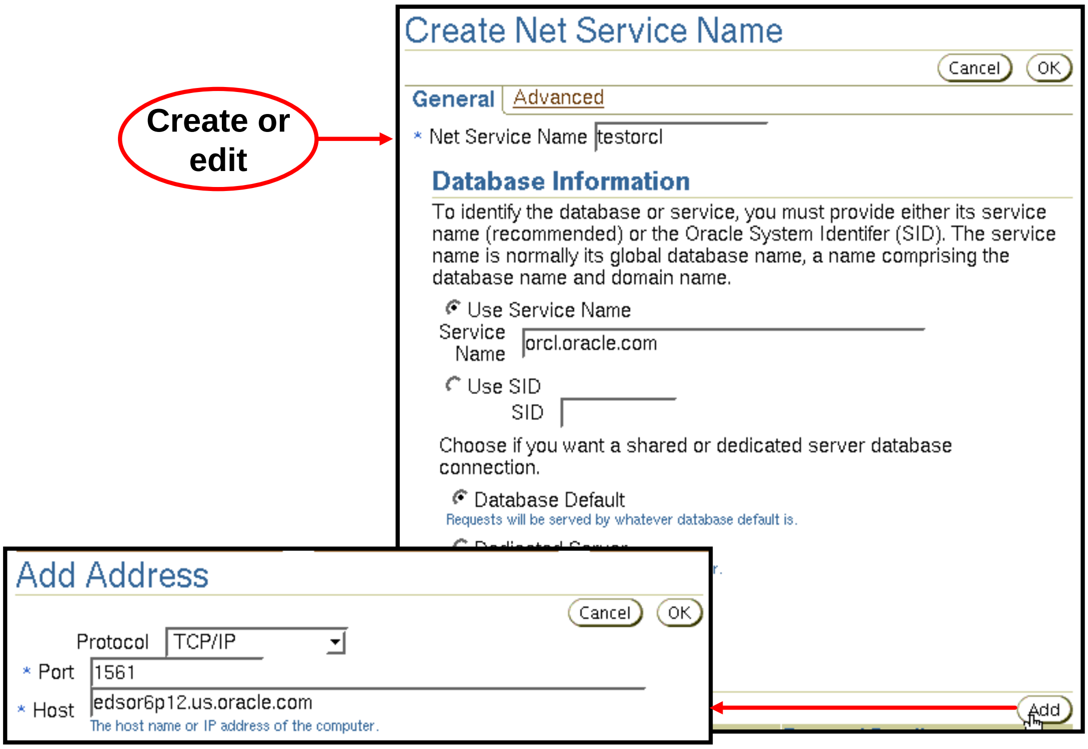
External Naming Method

The external naming method stores net service names in a supported non-Oracle naming service. Supported third-party services include:

- Network Information Service (NIS) External Naming
- Distributed Computing Environment (DCE) Cell Directory Services (CDS)

Conceptually, external naming is similar to directory naming.

Configuring Service Aliases



ORACLE

11-22

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.

You can configure service aliases for directory naming by selecting Directory Naming instead of Local Naming.

Note: If directory naming has not already been configured, you cannot select the Directory Naming option. Directory naming is discussed in the *Oracle Enterprise Identity Management* course as well as in the *Oracle Advanced Security Administration* manual.

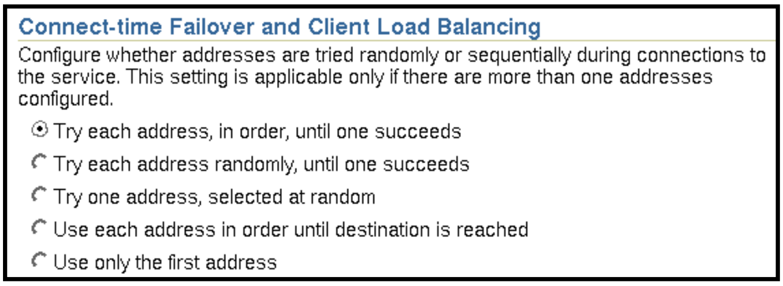
On the Create Net Service Name page, enter a unique name in the Net Service Name field (This is the name that users enter when they want to use this alias.). Enter the service name or system identifier (SID) of the database that you want to connect to, and click the Add button to enter the address for the service name.

For the address, enter the protocol, port, and host used by the listener for the service that you want to connect to.

Advanced Connection Options

Oracle Net supports the following advanced connection options with local and directory naming:

- **Connect-time failover**
- **Load balancing**
- **Source routing**



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 tried. Oracle Net keeps trying addresses in the listed 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.

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.

Advanced Connection Options (continued)

Note that there are five choices for connect-time failover and load balancing. The five options translate to the following:

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 the destination is reached.	Source routing
Use only the first address.	None

Testing Oracle Net Connectivity

The `tnsping` utility that tests Oracle Net service aliases:

- Ensures connectivity between the 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 orcl
```

ORACLE

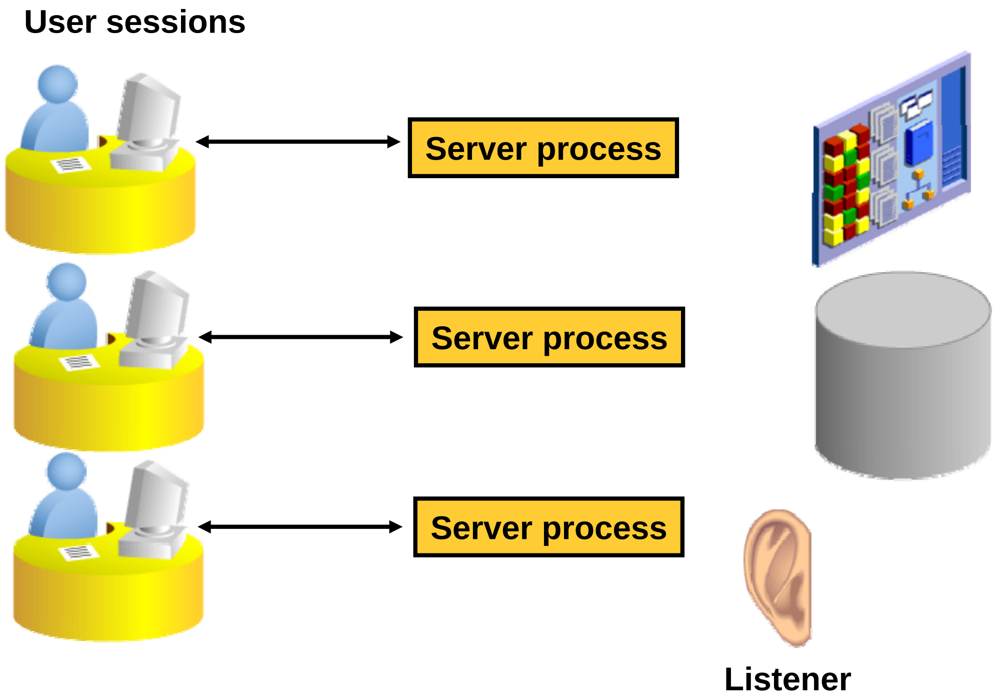
11-25

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 to a destination is good. For example, enter `tnsping orcl` in a command-line window.

The utility validates that the host name, port, and protocol reach a listener. It does not actually check whether the listener handles the service name. Another useful thing that `tnsping` reveals is the location of the configuration files. In a system with multiple `ORACLE_HOME` locations, this can be helpful.

User Sessions: Dedicated Server



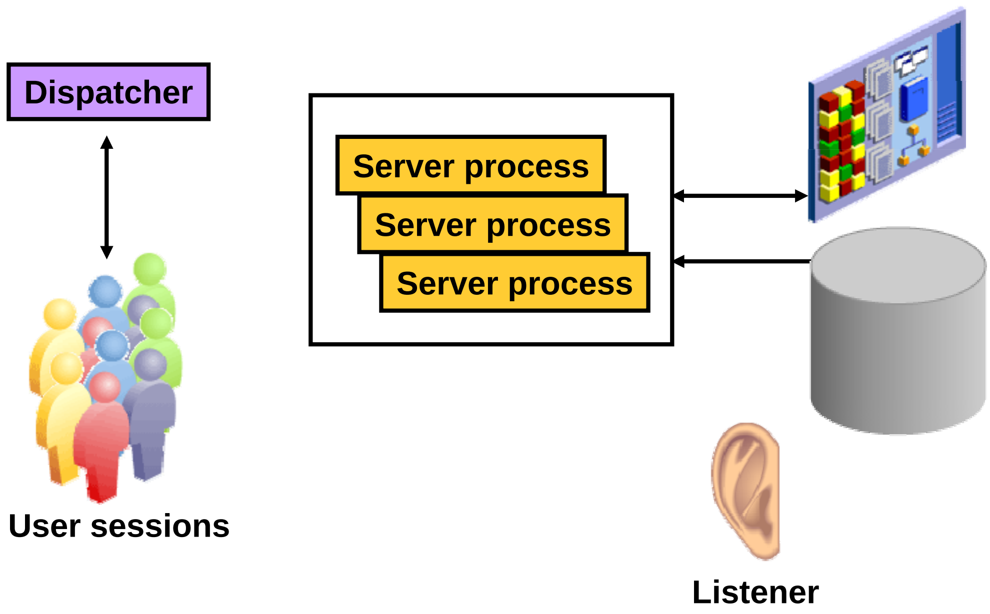
User Sessions: Dedicated Server

With dedicated server processes, there is a one-to-one ratio between server processes and user processes. Each server process uses system resources, including CPU cycles and memory.

In a heavily loaded system, the memory and CPU resources that are used by dedicated server processes can be prohibitive and can negatively affect the system's scalability. If your system is being negatively impacted by the resource demands of the dedicated server architecture, you have the following two options:

- Increasing system resources by adding more memory and additional CPU capability
- Using the Oracle Shared Server architecture

User Sessions: Shared Servers



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 that are 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.

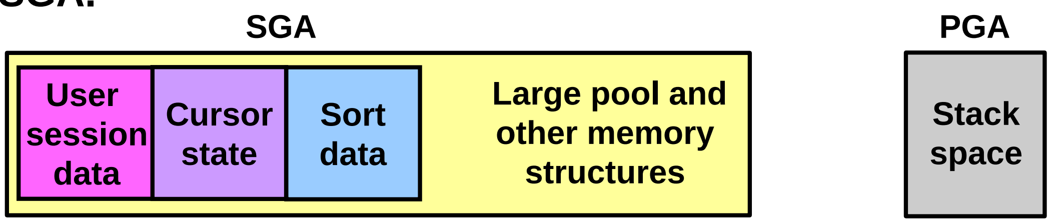
Dispatchers do not actually handle the work of user requests. Instead, they pass user requests to a common queue located in the shared pool portion of the SGA.

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 that are usually stored in the PGA must be in a shared memory location (by default, in the shared pool). However, if the large pool is configured or `SGA_TARGET` is set for Automatic Memory Management, these memory structures are stored in the large pool portion of the SGA.

SGA and PGA

Oracle Shared Server: User session data is held in the SGA.



Remember to factor in shared server memory requirement when sizing the SGA.

SGA and PGA

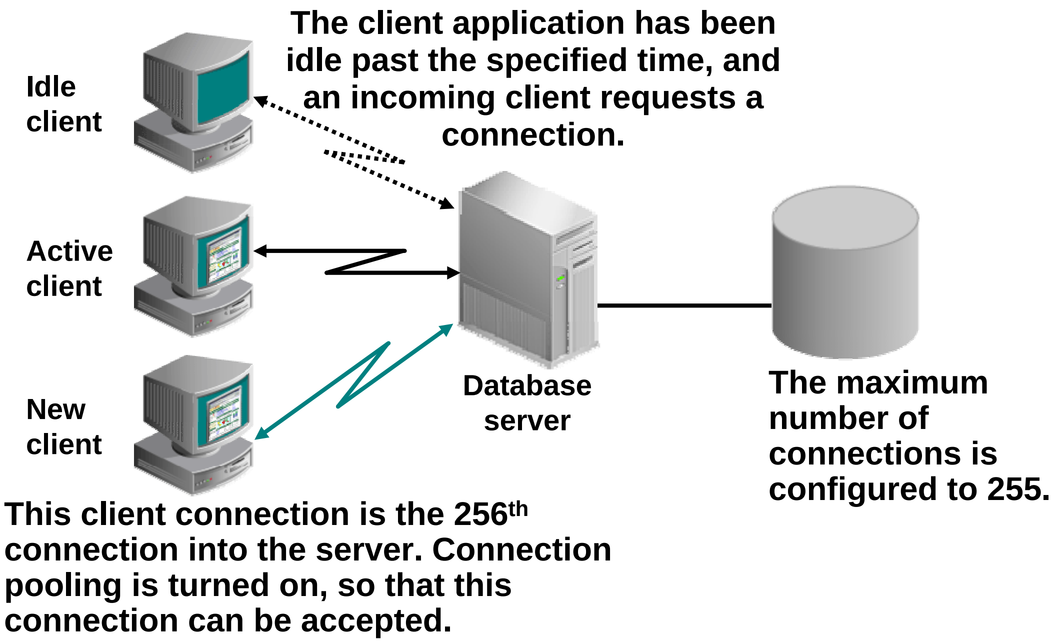
The contents of the SGA and the PGA differ when dedicated servers or shared servers are used:

- 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 the PGA is transparent to the user; however, if you are supporting multiple users, you need to increase the `LARGE_POOL_SIZE` initialization parameter. Each shared server process must 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.

Shared Server: Connection Pooling



Shared Server: Connection Pooling

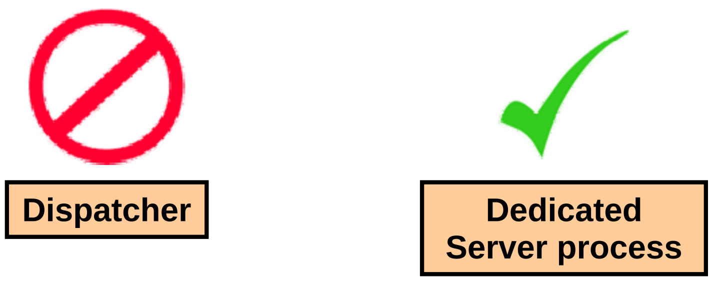
The connection pooling feature enables the database server to time out an idle session and use the connection to service an active session. The idle logical session remains open, and the physical connection is automatically reestablished when the next request comes from that session. Therefore, Web applications can allow larger numbers of concurrent users to be accommodated with existing hardware. Connection pooling is configurable through the shared server.

In this example, the Oracle database server has been configured with 255 connections. One of the clients has been idle past the specified time. Connection pooling makes this connection available to an incoming client connection, which is the 256th connection. When the idle client has more work to do, the connection is reestablished for that client with another client's idle connection.

When Not to Use a Shared Server

Certain types of database work must not be performed by using shared servers:

- **Database administration**
- **Backup and recovery operations**
- **Batch processing and bulk load operations**
- **Data warehouse operations**



When Not to Use a Shared Server

The Oracle Shared Server architecture is an efficient process and memory use model, but it 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 that use Oracle Recovery Manager (discussed in later lessons) also deal with very large data sets and must make use of dedicated connections.

Many administration tasks must not (and cannot) be performed by using shared server connections. These include starting up and shutting down the instance, creating tablespaces or data files, maintaining indexes and tables, analyzing statistics, and many other tasks that are commonly performed by the DBA. All DBA sessions must choose dedicated servers.

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager to:
  - Create additional listeners
  - Create Oracle Net Service aliases
  - Configure connect-time failover
  - Control the Oracle Net Listener
- Use `tnsping` to test Oracle Net connectivity
- Identify when to use shared servers versus dedicated servers

11-31

Practice Overview: Working with Oracle Network Components

This practice covers:

- Configuring local Names Resolution to connect to another database
- Creating a second listener for connect-time failover

12

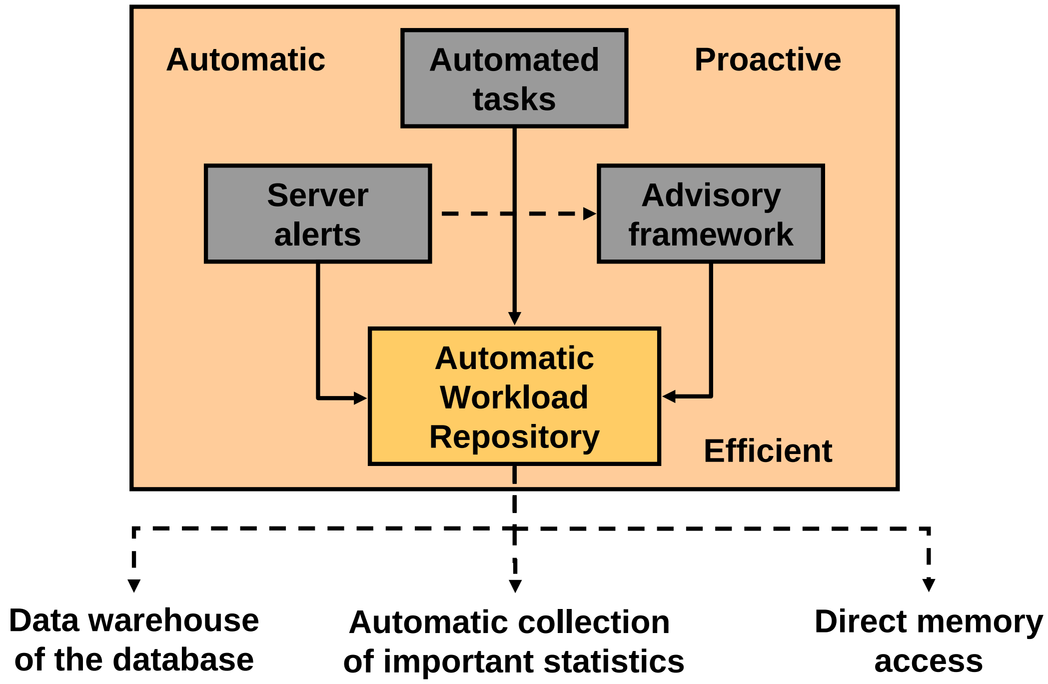
Proactive Maintenance

Objectives

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

- Use statistics
- Manage the Automatic Workload Repository (AWR)
- Use the Automatic Database Diagnostic Monitor (ADDM)
- Describe the advisory framework
- Set alert thresholds
- Use server-generated alerts
- Use automated tasks

Proactive Maintenance



12-3

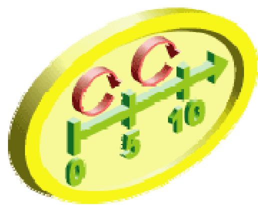
Proactive Maintenance

Proactive maintenance is made easy by the sophisticated infrastructure of the Oracle database. The main elements are as follows:

- The Automatic Workload Repository (AWR) is a built-in repository in each Oracle database. At regular intervals, the Oracle database makes a snapshot of all its vital statistics and workload information and stores them in the AWR. The captured data can be analyzed by you, by the database itself, or by both.
- By analyzing the information that is stored in the AWR, the database can identify the need to perform routine maintenance tasks, such as performing regular backups to maximize availability or refreshing statistics, which are used to optimize the execution of SQL statements.
- For problems that cannot be resolved automatically and require administrators to be notified (such as running out of space), the Oracle database provides server-generated alerts. The Oracle database can monitor itself and send out alerts to notify you of any problem. The alerts not only notify you of the problem, they also provide recommendations on how the reported problem can be resolved.
- Recommendations are generated from a number of advisors, each of which is responsible for a subsystem. For example, there are memory and SQL advisors.

Introducing Terminology

- **Automatic Workload Repository (AWR): Infrastructure for data gathering, analysis, and solutions recommendations**
- **Baseline: Data gathered of a “normal running database” for performance comparison**
- **Metric: Rate of change in a cumulative statistic**
- **Statistics: Data collections used for optimizing internal operations, such as execution of a SQL statement**
- **Threshold: A boundary value against which metric values are compared**



12-4

Introducing Terminology

Automatic Workload Repository (AWR) provides services to internal Oracle server components to collect, process, maintain, and utilize performance statistics for problem detection and self-tuning purposes.

Active Session History (ASH) is the history of recent session activity, stored in the AWR.

Statistics are a collection of data that provide more details about the database and the objects in the database. Optimizer statistics are used by the query optimizer to choose the best execution plan for each SQL statement.

Baseline data should include:

- Application statistics (transaction volumes, response time)
- Database statistics
- Operating system statistics
- Disk I/O statistics
- Network statistics

Optimizer Statistics

> **Statistics**
> AWR
> ADDM
> Advisors
> Alerts
> AutoTasks

Optimizer statistics are:

- Not real time
- Persistent across instance restarts
- Collected automatically

```
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
```


Optimizer Statistics

Optimizer statistics include table, column, index, and system statistics. 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.

The statistics that are 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 facts 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.

Optimizer statistics are collected automatically by the preconfigured `GATHER_STATS_JOB`, which runs during predefined maintenance windows, once per day.

Optimizer Statistics (continued)

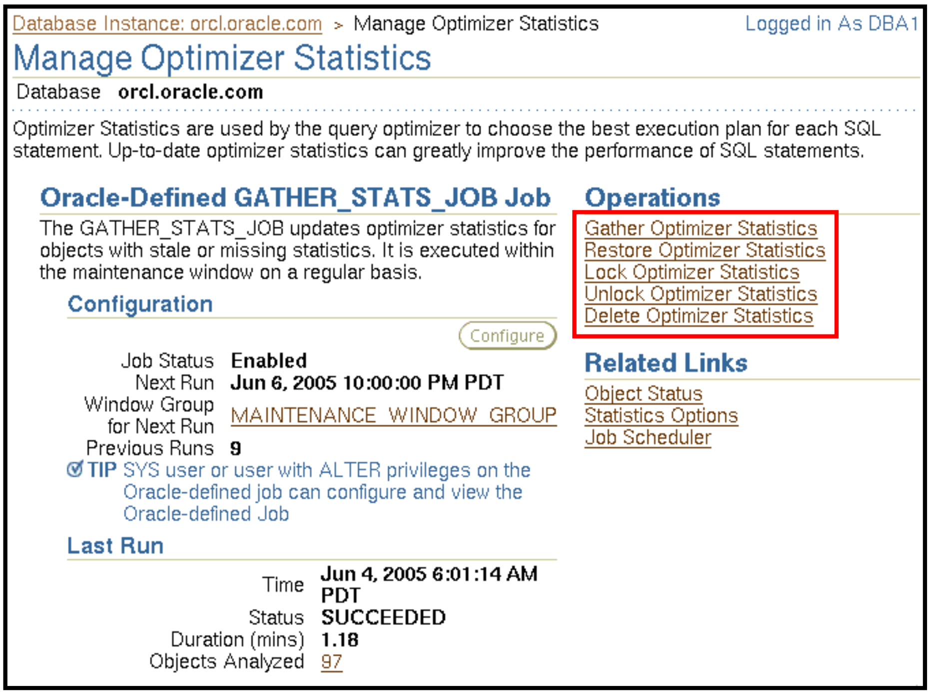
A large table that experiences 10 percent growth (or reduction) within a 24-hour period is usually considered too volatile for statistics collection once per day to be sufficient. For tables that experience this level of change, Oracle recommends collecting statistics more frequently, preferably often enough that the table never changes by more than about 10 percent between collection periods. This requires manual statistics collection.

Statistics can be manually collected by using Enterprise Manager or through the use of the `DBMS_STATS` package as shown here:

```
SQL> EXEC dbms_stats.gather_table_stats(‘HR’,’EMPLOYEES’);
SQL> SELECT num_rows FROM dba_tables
  2  WHERE owner='HR' AND table_name = 'EMPLOYEES';
  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 enables manual collection of statistics for an entire schema or even the whole database.

Using the Manage Optimizer Statistics Page



ORACLE

12-7

Using the Manage Optimizer Statistics Page

Go to the Enterprise Manager page for managing optimizer statistics by clicking Manage Optimizer Statistics on the Administration tabbed page. Note that (as this page shows) `GATHER_STATS_JOB` is enabled; it has run nine times; and last time, it successfully ran against 97 objects, which took a little over a minute. For `GATHER_STATS_JOB` to work properly, you must be sure that the `STATISTICS_LEVEL` initialization parameter is set to at least `TYPICAL`.

Note: The default window for this job is 10:00 p.m. to 6:00 a.m. on weekdays, and 12:00 a.m. Saturday to 12:00 a.m. Monday, on weekends. When the maintenance window closes, by default, the Scheduler terminates `GATHER_STATS_JOB`. The remaining objects are then processed in the next maintenance window.

Using the Manage Optimizer Statistics Page (continued)

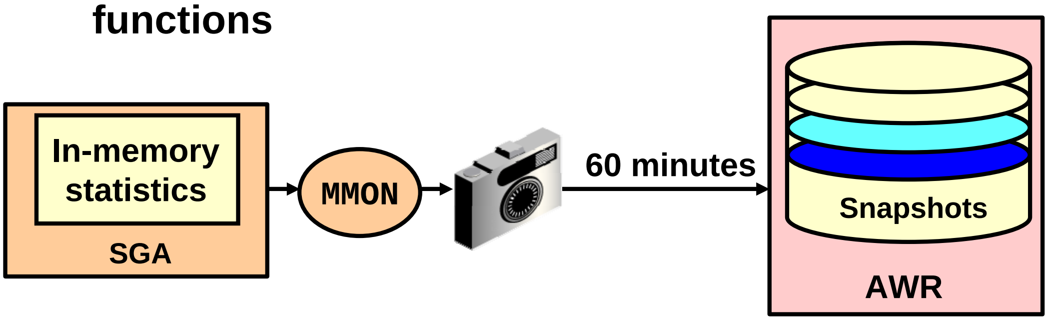
From this page, you can perform the following tasks on statistics:

- Gather optimizer statistics manually. This action submits the job that `GATHER_STATS_JOB` automatically does. This should be done in the case where a table's contents have changed so much between automatic gathering jobs that the statistics no longer represent the table accurately. Examples of this are a table that is truncated in the middle of the day and a batch job that runs and adds large amounts of data to a table.
- Restore optimizer statistics to a point in the past. The point in time chosen must be within the optimizer statistics retention period, which defaults to 30 days.
- Lock optimizer statistics to guarantee that the statistics for certain objects are never overwritten. This is useful if statistics have been calculated for a certain table at a time when well-representative data is present, and you want to always have those statistics. No fluctuation in the table affects the statistics if they are locked.
- Unlock optimizer statistics to undo the previously done lock
- Delete optimizer statistics to delete statistics

Automatic Workload Repository (AWR)

Statistics
> AWR
ADDM
Advisors
Alerts
AutoTasks

- Built-in repository of performance information
- Snapshots of database metrics taken every 60 minutes and retained for 7 days
- Foundation for all self-management functions



ORACLE

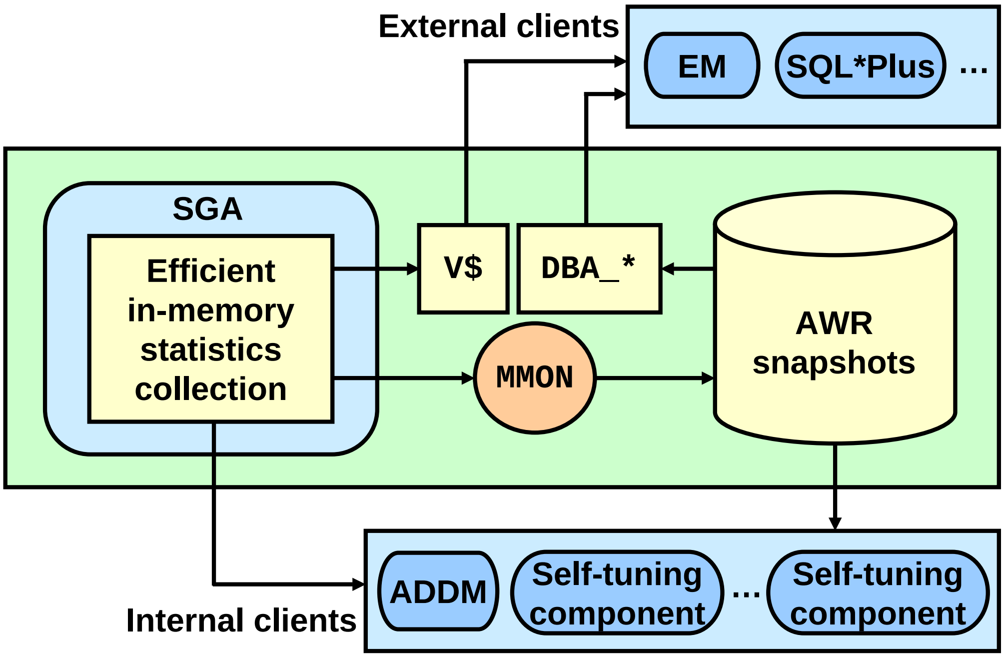
Automatic Workload Repository (AWR)

The AWR is the infrastructure that provides services to Oracle Database 10*g* components to collect, maintain, and utilize statistics for problem detection and self-tuning purposes. You can view it as a data warehouse for database statistics, metrics, and so on.

By default, every 60 minutes, the database automatically captures statistical information from the SGA and stores it inside the AWR in the form of snapshots. These snapshots are stored on the disk by a background process called Manageability Monitor (`MMON`). By default, snapshots are retained for seven days. You can modify both the snapshot interval and retention intervals.

The AWR contains hundreds of tables, all belonging to the `SYSMAN` schema and stored in the `SYSAUX` tablespace. The Oracle database does not support direct SQL access to the repository. Instead, use Enterprise Manager or the `DBMS_WORKLOAD_REPOSITORY` package to work with the AWR.

AWR Infrastructure



AWR Infrastructure

The AWR infrastructure consists of two major parts:

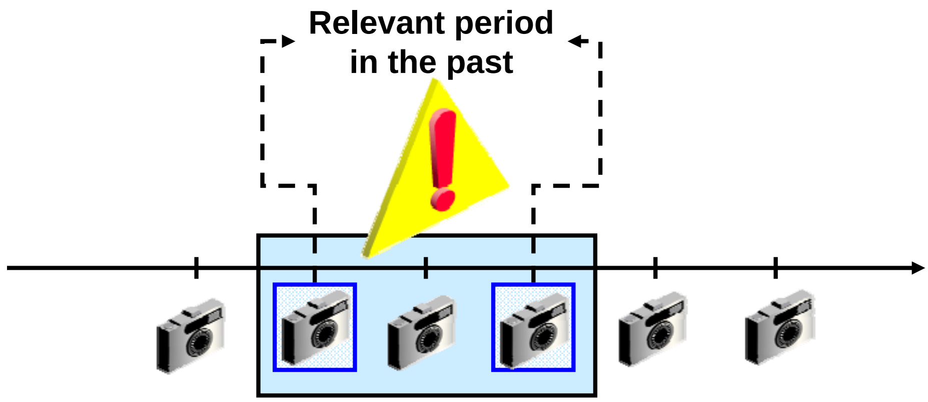
- An in-memory statistics collection facility that is used by Oracle Database 10*g* components to collect statistics. These statistics are stored in memory for performance reasons. Statistics stored in memory are accessible through dynamic performance (V$) views.
- The AWR snapshots that represent the persistent portion of the facility. AWR snapshots are accessible through data dictionary views and Enterprise Manager Database Control.

Statistics are stored in persistent storage for several reasons:

- The statistics need to survive instance crashes.
- Some analyses need historical data for baseline comparisons.
- A memory overflow can occur. When old statistics are replaced by new ones because of memory shortage, the replaced data can be stored for later use.

The memory version of the statistics is transferred to the disk on a regular basis by the `MMON` background process. With the AWR, the Oracle database provides a way to capture historical statistics data automatically, without the intervention of DBAs.

AWR Snapshot Sets




```
DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE ( -
        start_snap_id IN NUMBER   ,
        end_snap_id   IN NUMBER   ,
        baseline_name IN VARCHAR2);
```


AWR Snapshot Sets

Using snapshot sets is the mechanism for you to tag sets of snapshot data for important periods. A snapshot set is defined on a pair of snapshots; the snapshots are identified by their snapshot sequence numbers (`snap_id`). Each snapshot set corresponds to one and only one pair of snapshots.

A snapshot set can be identified by either a user-supplied name or a system-generated identifier. You simply create a snapshot set by executing the `DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE` procedure and specifying a name and a pair of snapshot identifiers. A snapshot set identifier is assigned to the newly created snapshot set. Snapshot set identifiers are unique for the life of a database.

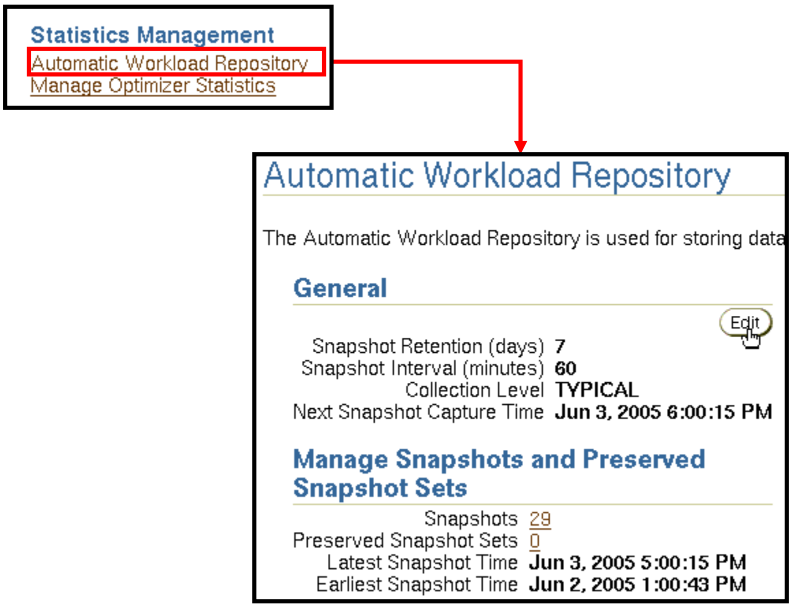
Snapshot sets are used to retain snapshot data. Therefore, snapshots belonging to snapshot sets are retained until the snapshot sets are dropped.

You set up snapshot sets, usually from some representative periods in the past, to be used for comparisons with current system behavior. You can also set up threshold-based alerts by using snapshot sets from Database Control.

You can get `snap_ids` directly from `DBA_HIST_SNAPSHOT` or Enterprise Manager Database Control.

Note: For more information about the `DBMS_WORKLOAD_REPOSITORY` package, see the *Oracle Database PL/SQL Packages and Types Reference* guide.

Enterprise Manager and AWR



12-12

Enterprise Manager and AWR

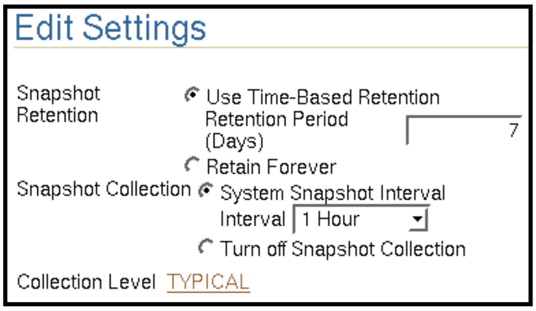
Select Administration > Database Administration > Statistics Management > Automatic Workload Repository and click Edit to change the settings.

From the Automatic Workload Repository page, you can:

- Edit the workload repository settings
- Look at the detailed information about created snapshots and manually create new ones
- Create baselines, also called Preserved Snapshot Sets
- Generate an AWR report

Managing the AWR

- **Retention period**
  - **The default is 7 days**
  - **Consider storage needs**
- **Collection interval**
  - **The default is 60 minutes**
  - **Consider storage needs and performance impact**
- **Collection level**
  - **Basic (disables most of ADDM functionality)**
  - **Typical (recommended)**
  - **All (adds additional SQL tuning information to snapshots)**



ORACLE

12-13

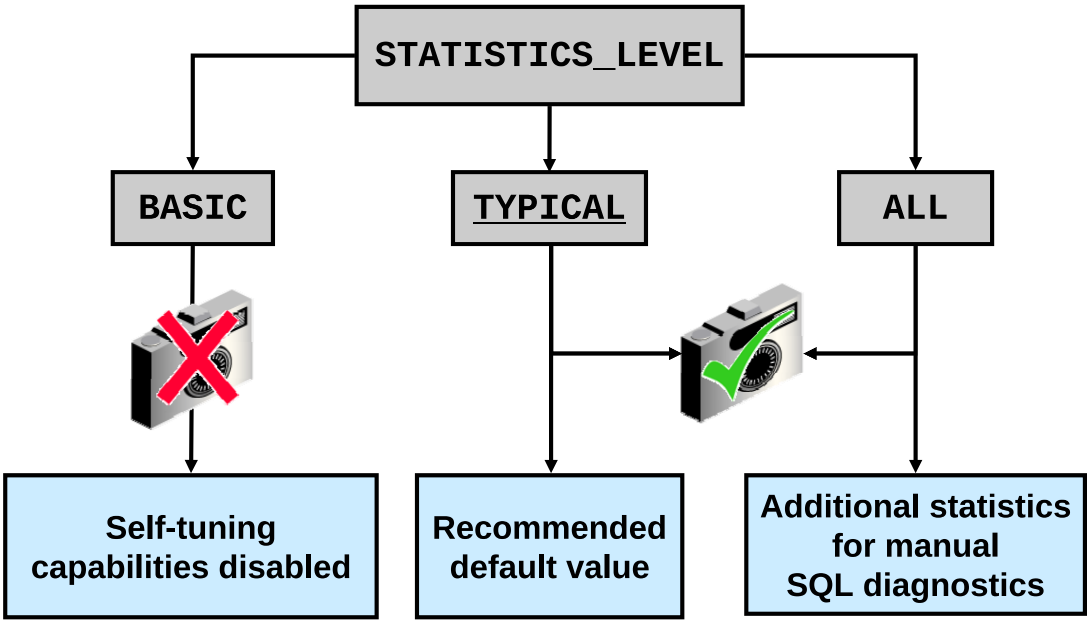
Managing the AWR

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 at the cost of the space required to store the snapshots and the performance expended in 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.

Statistic Levels



ORACLE

12-14

Statistic Levels

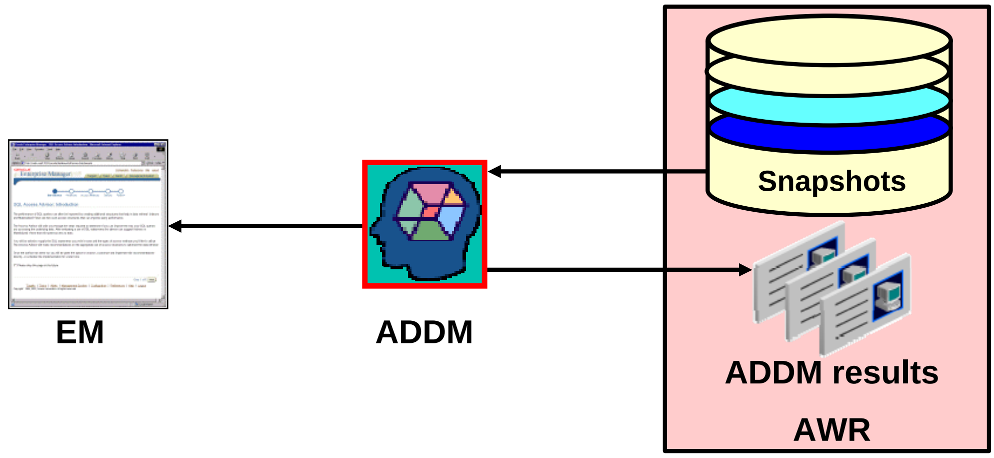
You can control the set of statistics to capture by using the `STATISTICS_LEVEL` initialization parameter, which has the following levels of capture:

- **`BASIC`:** The computation of AWR statistics and metrics is turned off.
- **`TYPICAL`:** Only some of the statistics are collected. They represent what is typically needed to monitor the Oracle database behavior. This automatic gathering of statistics reduces the likelihood of poorly performing SQL statements due to stale or invalid statistics.
- **`ALL`:** All possible statistics are captured. This level of capture should not be used except in certain rare cases for which you need extra SQL diagnostics information.

Automatic Database Diagnostic Monitor (ADDM)

Statistics
AWR
> ADDM
Advisors
Alerts
AutoTasks

- Runs after each AWR snapshot
- Monitors the instance; detects bottlenecks
- Stores results within the AWR



ORACLE

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 performs an analysis of the period corresponding to the last two snapshots. The ADDM proactively monitors the instance and detects most bottlenecks before they become a significant problem.

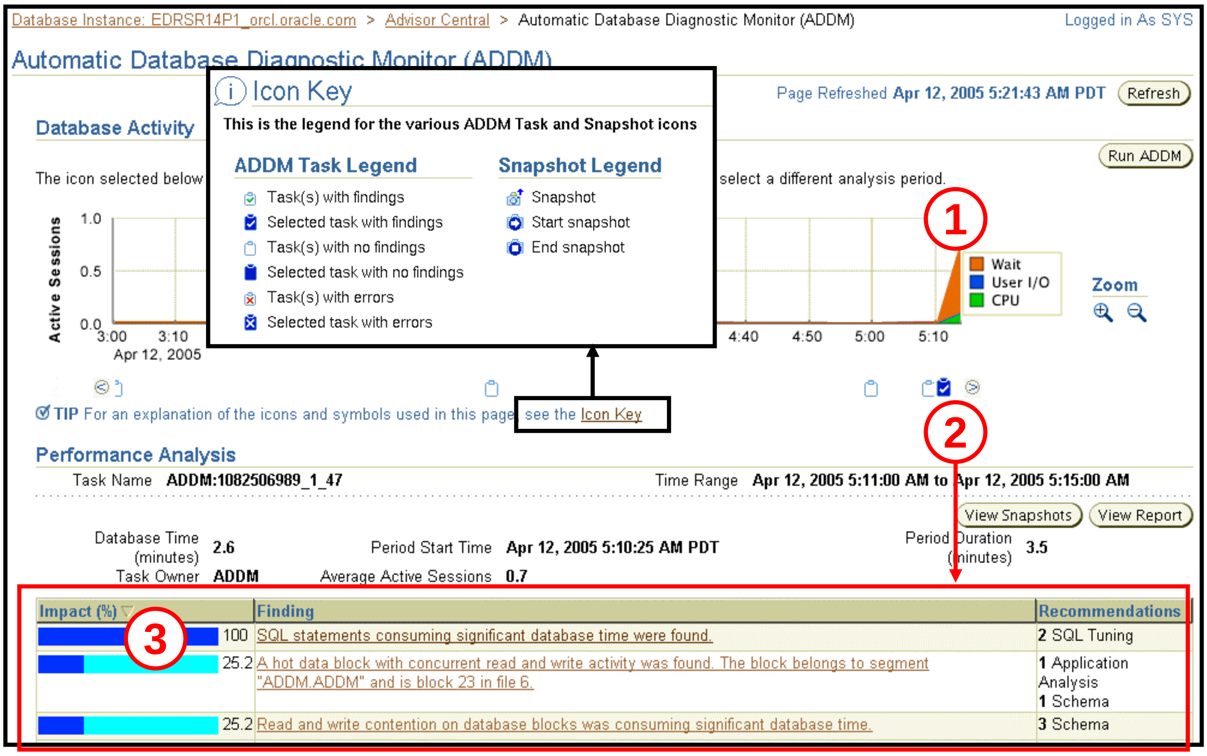
In many cases, the ADDM recommends solutions for detected problems and even quantifies the benefits for the recommendations.

Some common problems that are detected by the ADDM include:

- CPU bottlenecks
- Poor Oracle Net connection management
- Lock contention
- Input/output (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 AWR and are also accessible through Enterprise Manager.

ADDM Findings



12-16

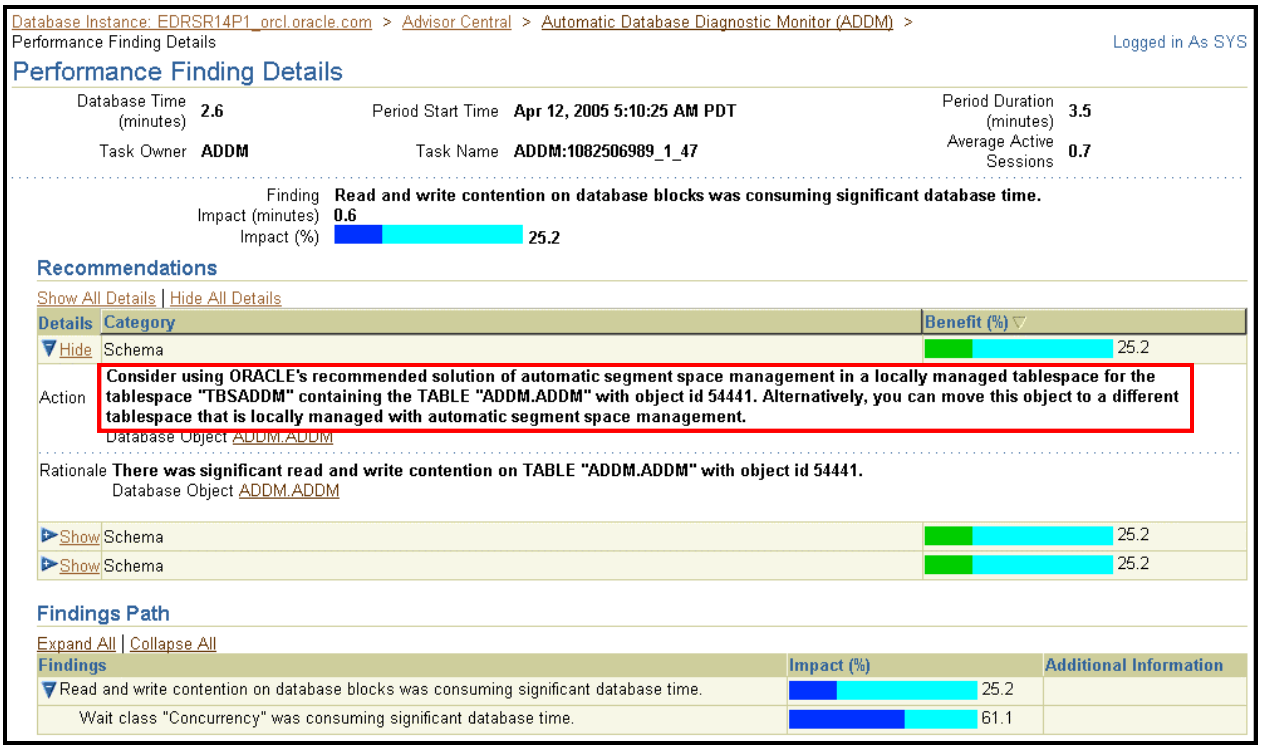
ADDM Findings

On the Automatic Database Diagnostic Monitor (ADDM) page, you can see the detailed findings for the latest ADDM run. Database Time represents the sum of the nonidle 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 as compared with the database time for the analysis period. In this slide, you see the following:

1. The graphic shows that the number of average active users increased dramatically at this point. Also, the major problem was a `Wait` problem.
2. The icon shows that the ADDM output displayed at the bottom of the page corresponds to this point in time. You can go into the past (to view previous analysis) by clicking the other icons.
3. The findings give you a short summary of what ADDM has found as tunable areas. By clicking a particular issue, you are directed to the Performance Finding Details page.

You can click the View Report button to get details about the performance analysis in the form of a text report.

ADDM Recommendations



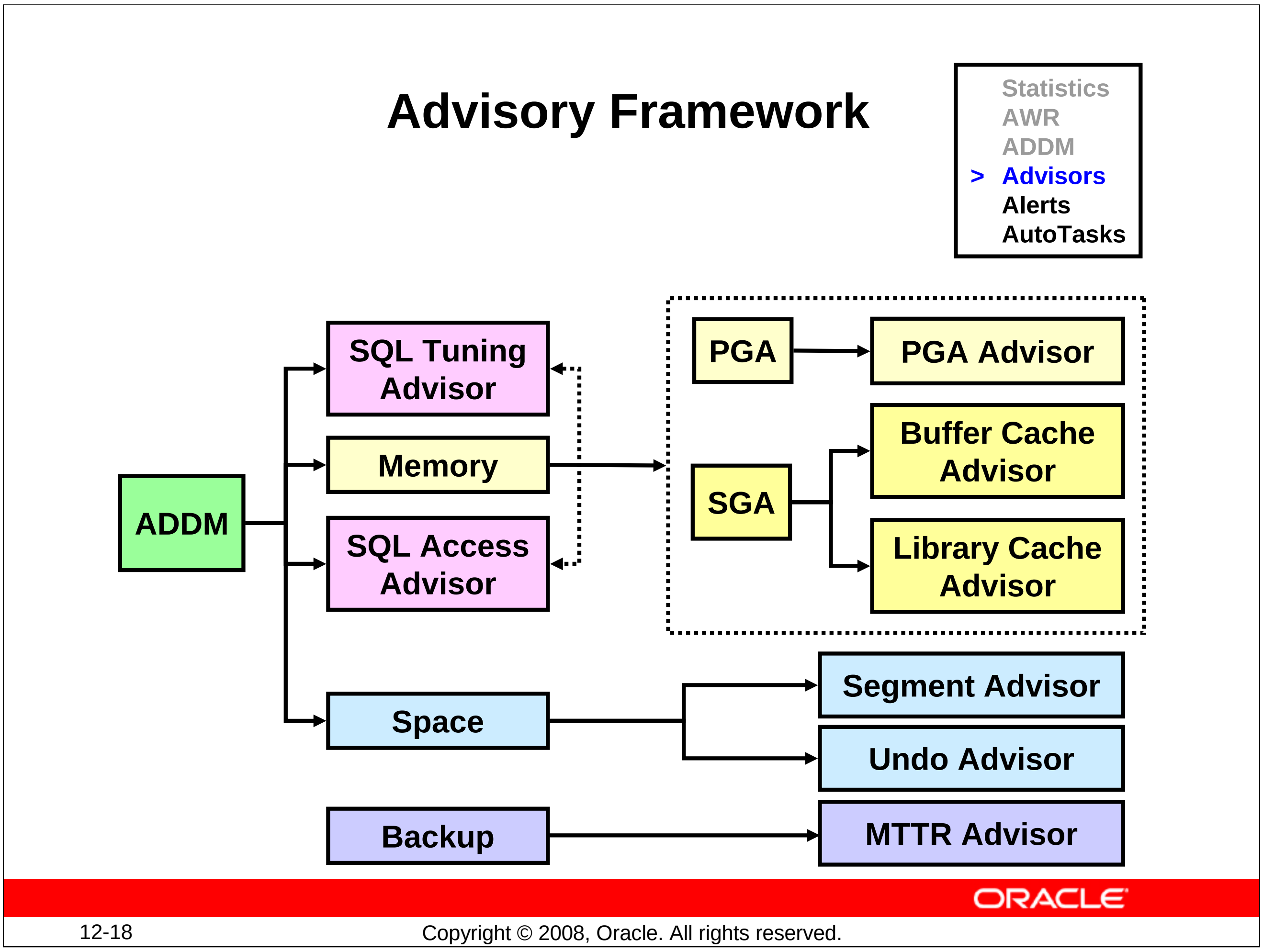
12-17

ADDM Recommendations

On the Performance Finding Details page, you are given some recommendations to solve the corresponding issue. Recommendations are grouped into Schema, SQL Tuning, Database Configuration, and many other categories. The `Benefit (%)` column gives you the maximum reduction in database elapsed time, if the recommendation is implemented.

ADDM considers a variety of changes to a system, and its recommendations can include:

- **Hardware changes:** Adding CPUs or changing the I/O subsystem configuration
- **Database configuration:** Changing initialization parameter settings
- **Schema changes:** Hash-partitioning a table or index, or using Automatic Segment Space Management (ASSM)
- **Application changes:** Using the cache option for sequences or using bind variables
- **Using other advisors:** Running the SQL Tuning Advisor on high-load SQL or running the Segment Advisor on hot objects



Advisory Framework

Advisors are server components that provide you with useful feedback about resource utilization and performance of their respective components.

By building upon the data captured in the AWR, the ADDM enables the Oracle database to diagnose its own performance and determine how identified problems can be resolved. ADDM runs automatically after each AWR statistics capture. It can potentially call other advisors.

Here are the major benefits that are provided by the advisor infrastructure:

- It uses a uniform interface for all advisors.
- All advisors have a common data source and results storage by using the workload repository.

Advisory Framework (continued)

Automatic Database Diagnostic Monitor (ADDM)

It is 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.

Memory Advisors

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 (PGA). In addition to the advisory functions, this page provides a central point of control for the large pool and the Java pool.

Mean-Time-To-Recover (MTTR) Advisor

Using the MTTR Advisor, you can set the length of time required for the database to recover after an instance crash.

Segment Advisor

This advisor looks for tables and indexes that consume 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.

SQL Access Advisor

This advisor analyzes all SQL statements that are issued within a given period and suggests the creation of additional indexes or materialized views that will improve performance.

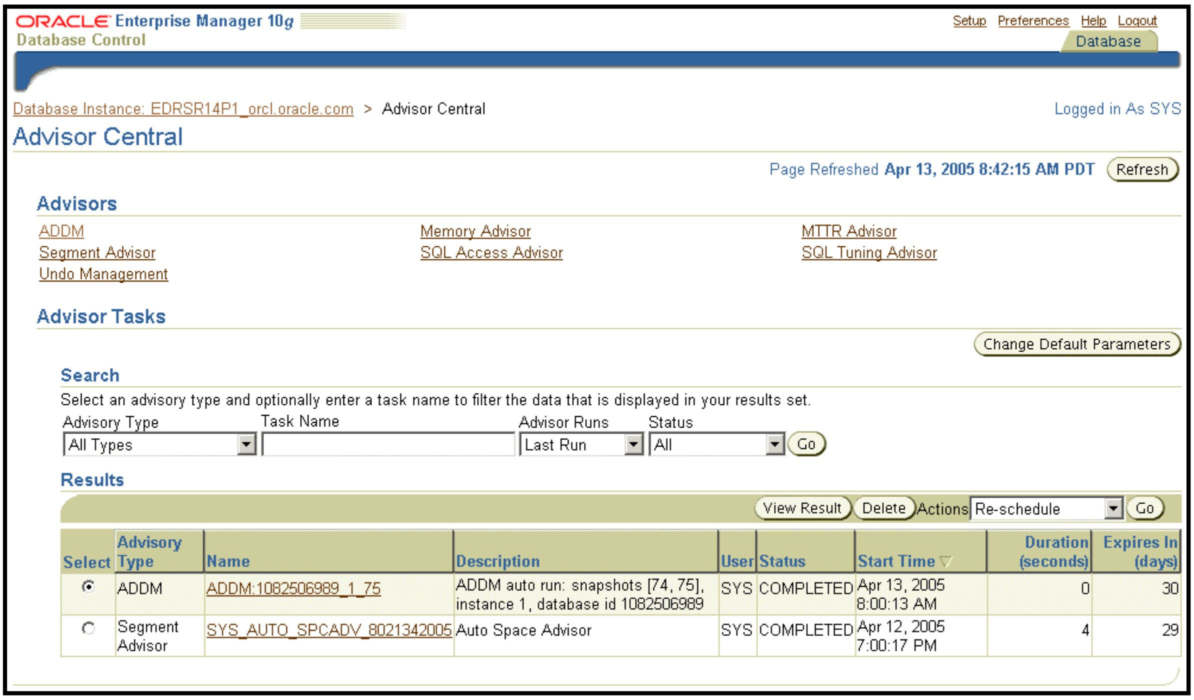
SQL Tuning Advisor

This advisor analyzes an individual SQL statement and makes recommendations for improving its performance. Recommendations may 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.

Undo Management Advisor

With the Undo Management Advisor, you can determine the undo tablespace size that is required to support a given retention period. Undo management and the use of the advisor is covered in the lesson titled "Managing Undo Data."

Enterprise Manager and Advisors



Enterprise Manager and Advisors

The Advisor Central page is the main page of all advisors. You can reach this page by clicking the Advisor Central link in the list of Related Links of the Database Control Home page. However, this is not the only place in Database Control where advisors can be invoked. It is also possible to have access to advisors in certain contexts.

On the Advisor Central page, you can list all the advisor tasks that are registered in the workload repository. You can also filter this list by advisor type and for predefined time periods.

Some advisors are described in greater detail in the lessons titled "Managing Undo Data", "Performance Management", and "Backup and Recovery Concepts".

Note: Use the Change Default Parameters page to change the default expiration in days for all future tasks. You can also use this page to change some important advisor's parameters.

The DBMS_ADVISOR Package

Procedure	Description
CREATE_TASK	Creates a new task in the repository
DELETE_TASK	Deletes a task from the repository
EXECUTE_TASK	Initiates execution of the task
INTERRUPT_TASK	Suspends a task that is currently executing
GET_TASK_REPORT	Creates and returns a text report for the specified task
RESUME_TASK	Causes a suspended task to resume
UPDATE_TASK_ATTRIBUTES	Updates task attributes
SET_TASK_PARAMETER	Modifies a task parameter
MARK_RECOMMENDATION	Marks one or more recommendations as accepted, rejected, or ignored
GET_TASK_SCRIPT	Creates a script of all the recommendations that are accepted

ORACLE

12-21

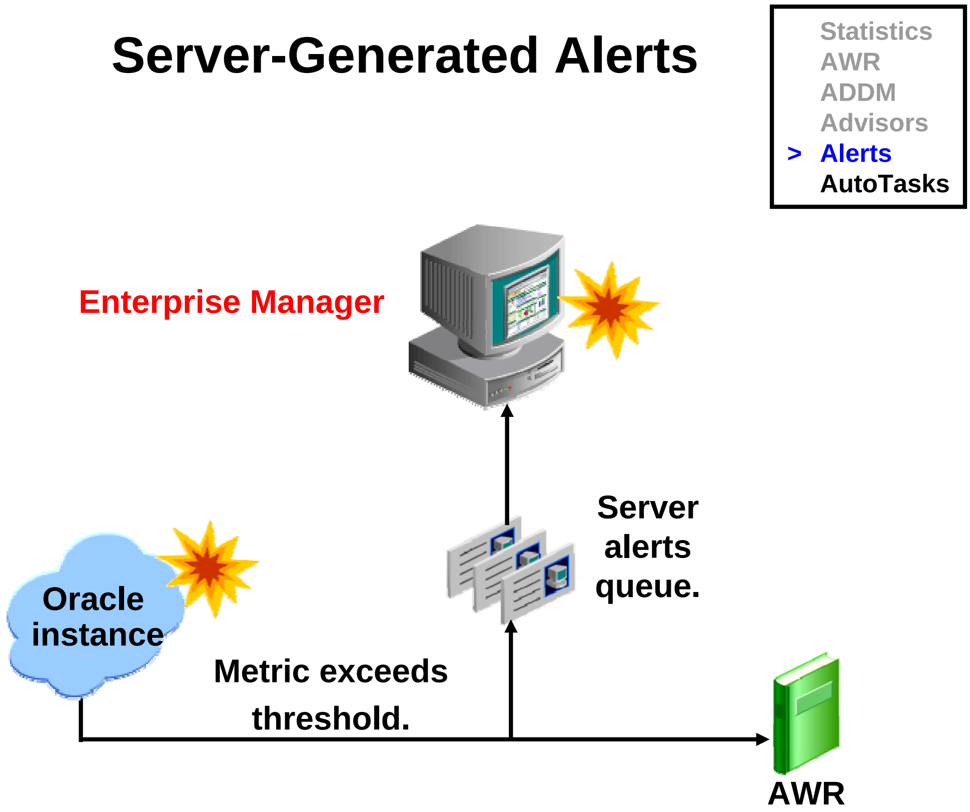
The DBMS_ADVISOR Package

The DBMS_ADVISOR package contains all constants and procedure declarations for all advisor modules. You can use this package to execute tasks via command line.

To execute advisor procedures, you must be granted the ADVISOR privilege. The ADVISOR privilege permits full access to the advisor procedures and views.

Note: For more information about all the procedures found in the DBMS_ADVISOR package, see the *Oracle Database PL/SQL Packages and Types Reference* guide.

Server-Generated Alerts



12-22

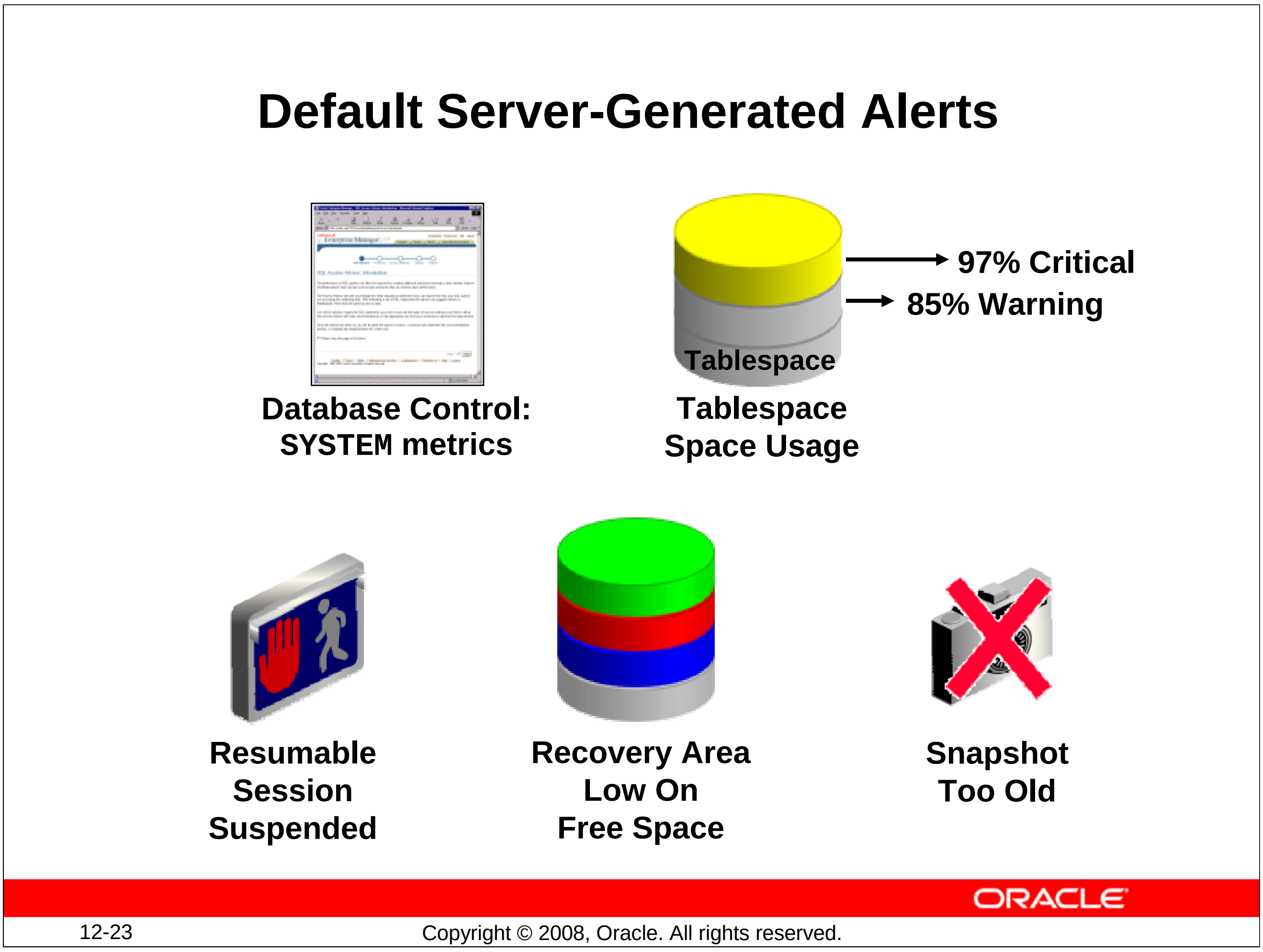
Server-Generated Alerts

Alerts are notifications of when a database is in an undesirable state and needs your attention. By default, the Oracle database provides alerts via Enterprise Manager Database Control. Optionally, Enterprise Manager can be configured to send an e-mail to 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* proactively notifies you if the database deviates from normal readings enough to reach those thresholds. An early notification of potential problems enables 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 (centiseconds)
- Dump Area Used (%)
- Response Time (per transaction)
- SQL Response Time (%)
- Tablespace Used (%)
- Wait Time (%)



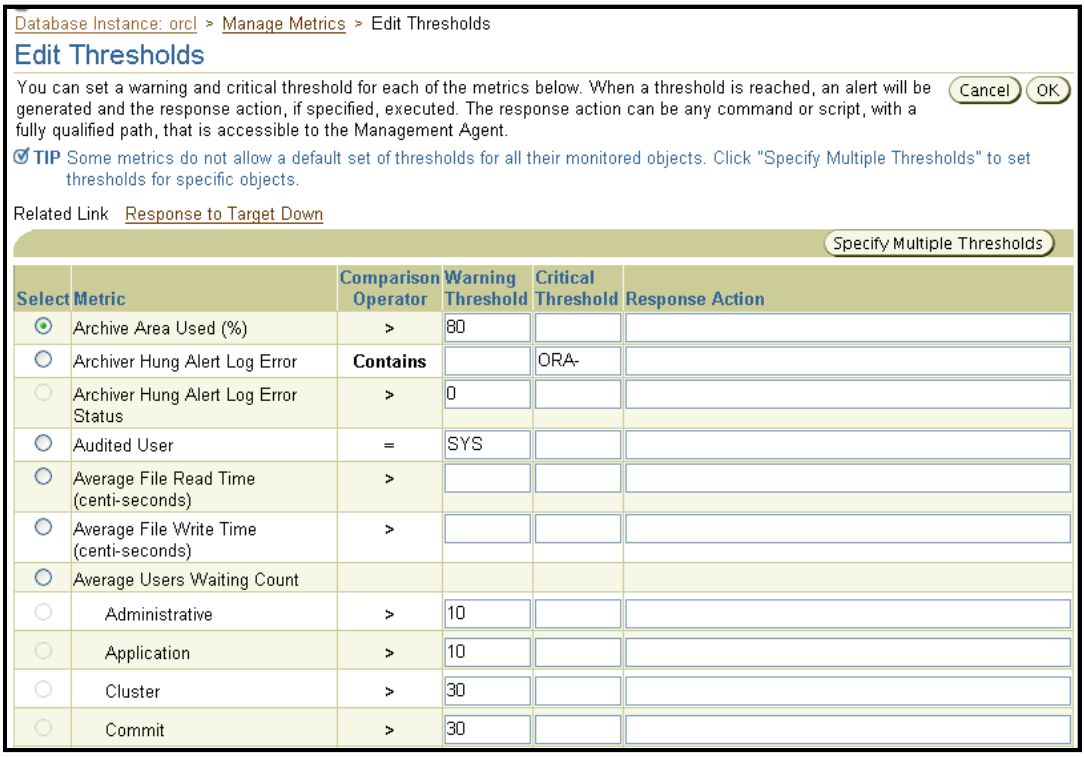
Default Server-Generated Alerts

By default, the following server-generated alerts are enabled:

- Tablespace Space Usage (warning 85%, critical 97%)
- Snapshot Too Old
- Recovery Area Low On Free Space
- Resumable Session Suspended

Note: Enterprise Manager Database Control automatically sets thresholds on server metrics with the `SYSTEM` object type.

Setting Thresholds

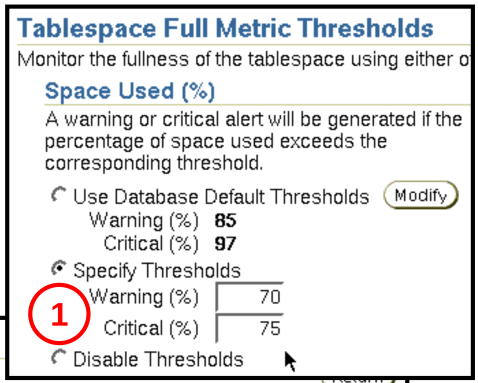


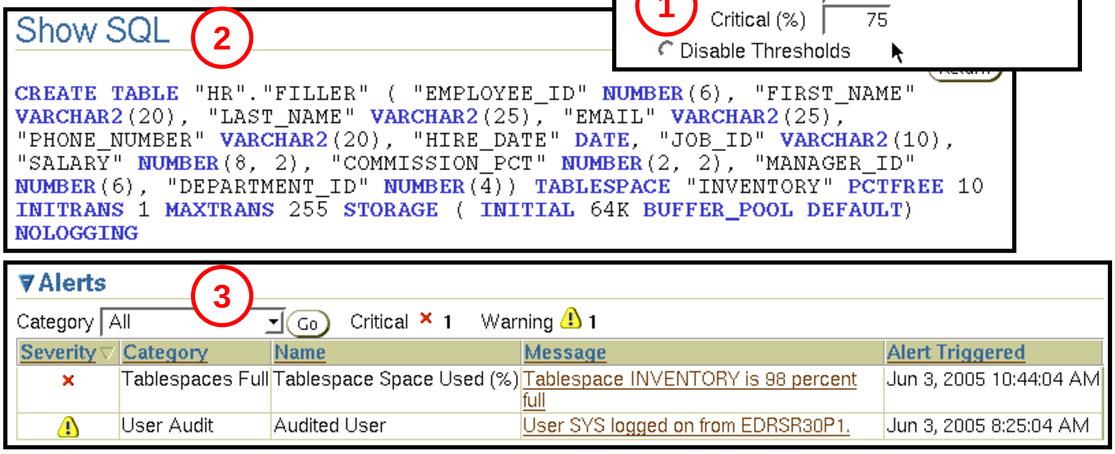
Setting Thresholds

To set or edit a threshold for your whole database, select Manage Metrics in the Related Links region of the database home page. Click Edit Threshold. Enter your desired Warning and Critical Threshold values. The appropriate alerts appear when the database reaches your specified values. If required, you can specify an additional response action.

Creating and Testing an Alert

1. Specify a threshold.
2. Create a test case.
3. Check for an alert.





12-25

Creating and Testing an Alert

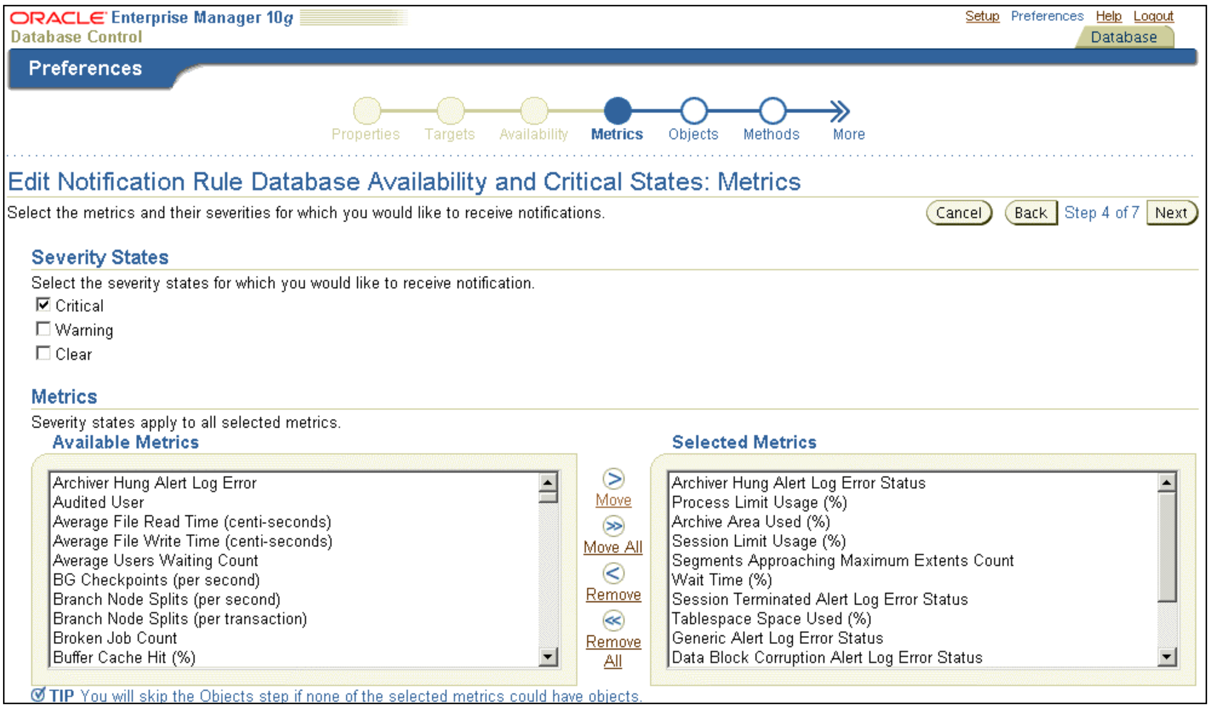
You can also set thresholds for a specific object.

Example: You decide that you need to receive a critical alert if the space used in the `INVENTORY` tablespace exceeds 75%. (This tablespace does not allow its data files to automatically extend.) To create and test the alert, perform the following steps:

1. In Enterprise Manager, navigate to the tablespace administration, and set your desired threshold.
2. Use the “Create like” action to duplicate an existing table, and fill it using SQL*Plus.
3. After you receive an error that this table is unable to extend, check the Database Instance home page for the associated alert.

Most alerts contain a name of an associated advisor that can be invoked to give you more detailed advice. For each corresponding alert message, Database Control provides a link to invoke the corresponding advisor.

Alerts Notification



ORACLE

12-26

Alerts Notification

The notification mechanism uses the user interface because it is already available in Enterprise Manager. It is based on the concept of a notification rule that establishes the appropriate notification mechanism for a set of upcoming alerts.

Using Database Control, you can edit the notification rules. On the home page, click the Preferences link. This displays the General page, where you can specify your e-mail address at which you want to receive notifications.

On the General page, click the Rules link in the Notification region. Select the Database Availability and Critical States rule, and then click the Edit button. This takes you to the Edit Notification Rule Database Availability and Critical States wizard page, where you can select the metrics (and their severities) for which you want to receive notifications.

Alerts Notification (continued)

You can optionally specify that Enterprise Manager provide you with direct notification when specific alerts arise. For example, if you specify that you want e-mail notification for critical alerts, and you have a critical threshold set for the system response time for each call metric, then you may send an e-mail containing a message similar to the following:

```
Host Name=mydb.us.mycompany.com
Metric=Response Time per Call
Timestamp=08-NOV-2005 10:10:01 (GMT -7:00)
Severity=Critical
Message=Response time per call has exceeded the threshold.
  See the latest ADDM analysis.
Rule Name= Rule
Owner=SYSMAN
```

The e-mail contains a link to the host name and the latest ADDM analysis.

By default, alerts in critical state such as DB Down, Generic Alert Log Error Status, and Tablespace Used are set up for notification. However, to receive these notifications, you must set up your e-mail information by following these steps:

1. On any Database Control page, click the Setup link, which is visible in the header and footer area.
2. On the Setup page, select Notification Methods.
3. Enter the required information in the Mail Server region of the Notifications Methods page.

There are other methods of notification, including scripts and Simplified Network Management Protocol (SNMP) traps. The latter can be used to communicate with third-party applications.

To receive notifications, perform the following steps:

1. On any Database Control page, click the Preferences link, which is visible in the header and footer area.
2. On the Preferences page, select General. Enter your e-mail address in the E-mail Addresses region.
3. You can optionally edit notification rules, such as to change the severity state for receiving notification. To do so, select Notification Rules. The Notification Rules page appears. For more information about configuring notification rules, see the *Oracle Enterprise Manager Advanced Configuration* documentation.

Reacting to Alerts

- If needed, gather more input, for example, by running ADDM or another advisor.
- Take corrective measures.
- Acknowledge alerts, which are not automatically cleared.

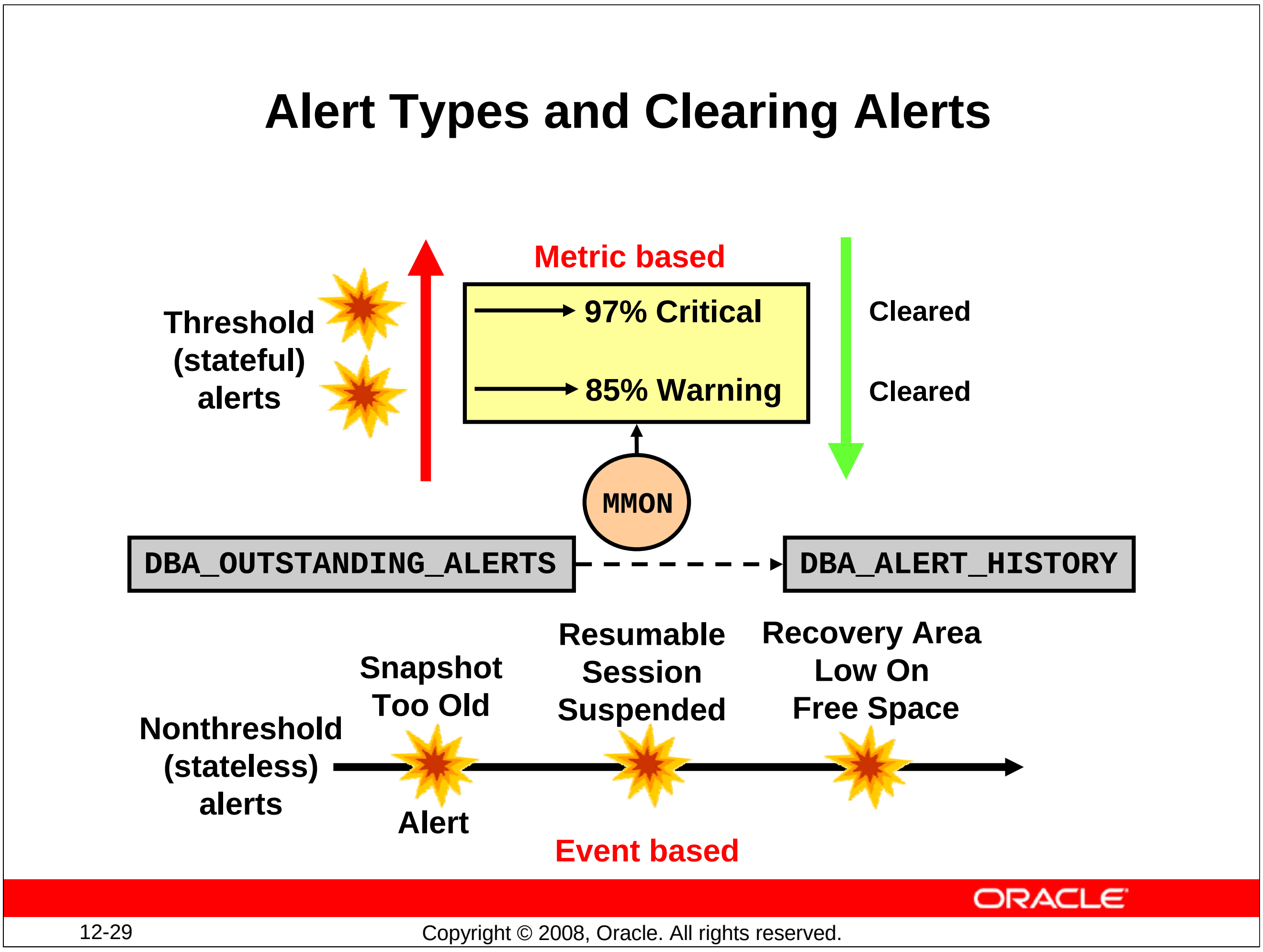
12-28

Reacting to Alerts

When you receive an alert, follow any recommendations it provides, or consider running ADDM or another advisor as appropriate to get more detailed diagnostics of system or object behavior.

Most alerts, such as the Out of Space alert, are cleared automatically when the cause of the problem disappears. However, other alerts such as Generic Alert Log Error are sent to you for notification and need to be acknowledged by you. After taking the necessary corrective measures, you can acknowledge an alert by clearing or purging it. Clearing an alert sends the alert to the Alert History, which is viewable from the home page under Related Links. Purging an alert removes it from the Alert History.

To clear an alert such as Generic Alert Log Error, from the home page under Diagnostic Summary, click the Alert Log link. The Alert Log Errors page appears. Select the alert to clear and click Clear. To purge an alert, select it and click Purge. You can also Clear Every Open Alert or Purge Every Alert using these buttons.



Alert Types and Clearing Alerts

There are two kinds of server-generated alerts: threshold and nonthreshold.
Most server-generated alerts are configured by setting a warning and critical threshold values on database metrics. You can define thresholds for more than 120 metrics. For example:

- Physical Reads Per Sec
- User Commits Per Sec
- SQL Service Response Time

Except for the Tablespace Space Usage metric, which is database related, the other metrics are instance related. Threshold alerts are also referred to as stateful alerts. These alerts are automatically cleared when an alert condition clears. Stateful alerts appear in `DBA_OUTSTANDING_ALERTS` and, when cleared, go to `DBA_ALERT_HISTORY`.

Other server-generated alerts correspond to specific database events such as Snapshot Too Old errors, Recovery Area Low On Free Space, and Resumable Session Suspended. These are non-threshold-based alerts, also referred to as stateless alerts. Stateless alerts go directly to the history table. Clearing a stateless alert makes sense only in the Database Control environment because Database Control stores stateless alerts in its own repository.

Automated Maintenance Tasks

Statistics
AWR
ADDM
Advisors
Alerts
\> AutoTasks

- Scheduler initiates jobs
- Jobs run in the default maintenance window
- Limit maintenance impact on normal operation by using Resource Manager

Examples of maintenance:

- Gathering optimizer statistics
- Gathering segment information
- Backing up database

ORACLE

12-30

Automated Maintenance Tasks

By analyzing the information stored in the AWR, the database can identify the need to perform routine maintenance tasks, such as optimizer statistics refresh. The automated maintenance tasks infrastructure enables the Oracle database to automatically perform such operations. It uses the Scheduler to run such tasks in a predefined "maintenance window."

By default, the maintenance window starts at 10 p.m. every night and lasts until 6 a.m. the next morning and throughout the weekend. All attributes of the maintenance window are customizable, including the start and end time, frequency, days of the week, and so on. Also, the impact of automated maintenance tasks on normal database operations can be limited by associating a Database Resource Manager resource plan to the maintenance window.

The examples of maintenance are as follows:

- Optimizer statistics are automatically refreshed by using the automatic maintenance task infrastructure.
- The Segment Advisor has default jobs, which run in the maintenance window.
- When creating a database with the DBCA, you can initiate regular database backups.

Summary

In this lesson, you should have learned how to:

- Use statistics
- Manage the Automatic Workload Repository
- Use the Automatic Database Diagnostic Monitor
- Describe the advisory framework
- Set alert thresholds
- Use server-generated alerts
- Use automated tasks

Practice Overview: Proactive Maintenance

This practice covers the following topics:

- Proactively managing your database by using ADDM
  - Setting up an issue for analysis
  - Reviewing your database performance
  - Implementing a solution

12-32

13

Performance Management

Objectives

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

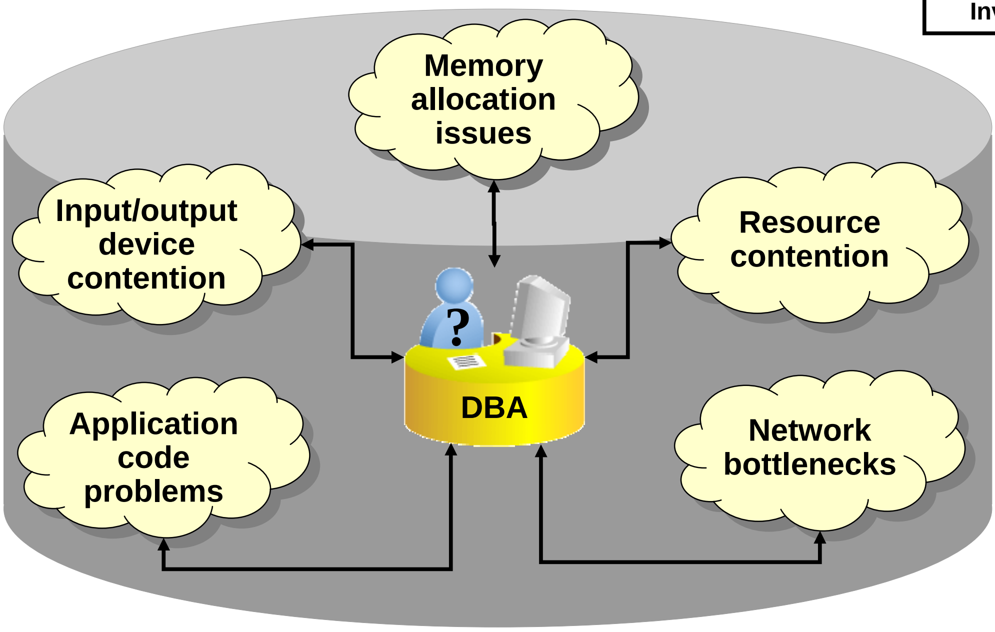
- Use Enterprise Manager to monitor performance
- Tune SQL by using the SQL Tuning Advisor
- Tune SQL by using the SQL Access Advisor
- Use Automatic Shared Memory Management (ASSM)
- Use the Memory Advisor to size memory buffers
- View performance-related dynamic views
- Troubleshoot invalid and unusable objects

ORACLE

13-2

Performance Monitoring

> **Perf Mon**
Tuning Adv
Access Adv
Memory
Stats
Invalid Obj



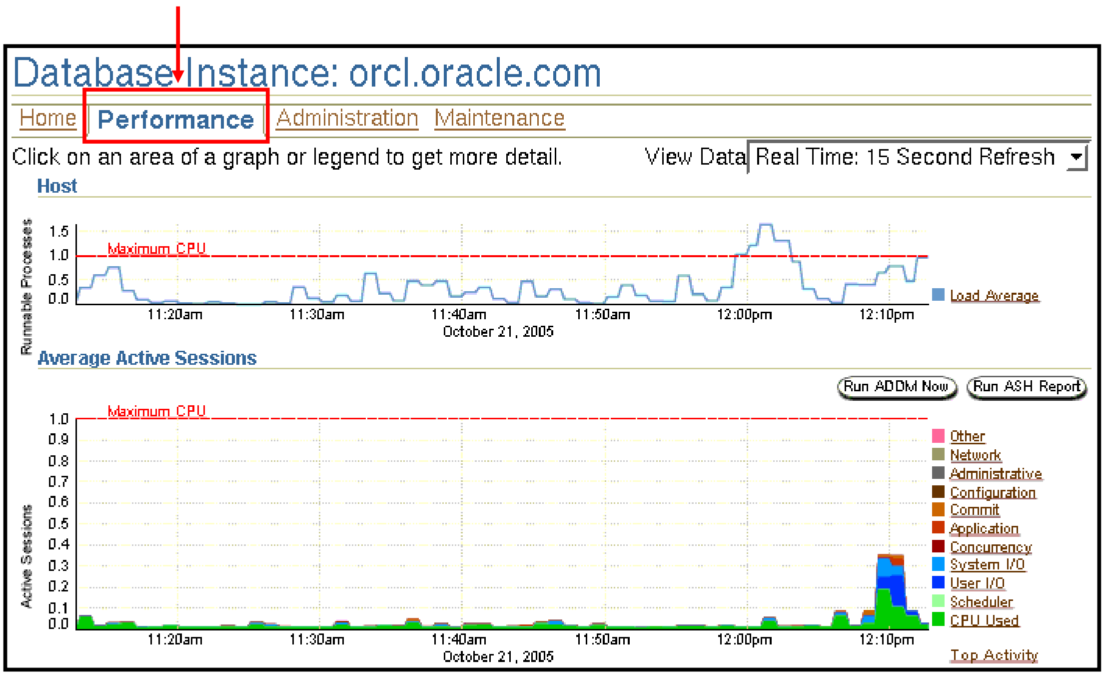
Performance Monitoring

To administer Oracle Database 10*g* and keep it running smoothly, the database administrator (DBA) must regularly monitor its performance to locate bottlenecks and correct problem areas.

There are hundreds of performance measurements the DBA can look at, covering everything from network performance and disk input/output (I/O) speed to the time spent working on individual application operations. These performance measurements are commonly referred to as database metrics.

Note: For more information about Oracle database performance, see the *Oracle Database 10g: SQL Tuning Workshop* course.

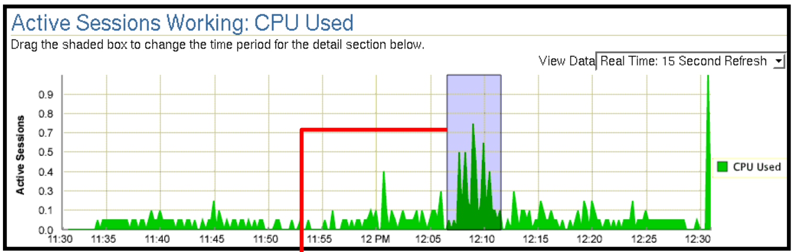
Performance Monitoring



Performance Monitoring (continued)

The Performance tabbed page in Enterprise Manager is the portal to a powerful set of performance monitoring and tuning tools. The first screen on this page summarizes processes and active session activity. The Average Active Sessions graph shows the level of CPU usage and what resources are causing the most wait events. In the screen in the slide, you see that there was a recent increase in CPU usage and waits for User I/O, System I/O, and Concurrency. You can click any one of these categories to see more details about the waits. The I/O data is broken down into types of I/O—for example, log file read, control file write, and so on.

Performance Monitoring



Detail for Selected 5 Minute Interval

Start Time **Oct 21, 2005 12:06:35 PM PDT**

Top Working SQL

Schedule SQL Tuning Advisor | Create SQL Tuning Set

Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
	30.19	a0q0ya8fxx52s	INSERT
	9.43	257rmrxgvaj4z	SELECT
	7.55	8f4zf0m1b7b6u	INSERT
	7.55	9c3326865m2h9	SELECT
	7.55	cakg0hdjjw2wf	SELECT
	3.77	fsz8wz5pmvamh	SELECT
	3.77	6uvk7uc8m4mf0	SELECT
	3.77	4c1xvq9ufwcjc	SELECT
	1.89	f787fyhjmkp61	INSERT

Total Sample Count: 53

Top Working Sessions

View Top Sessions

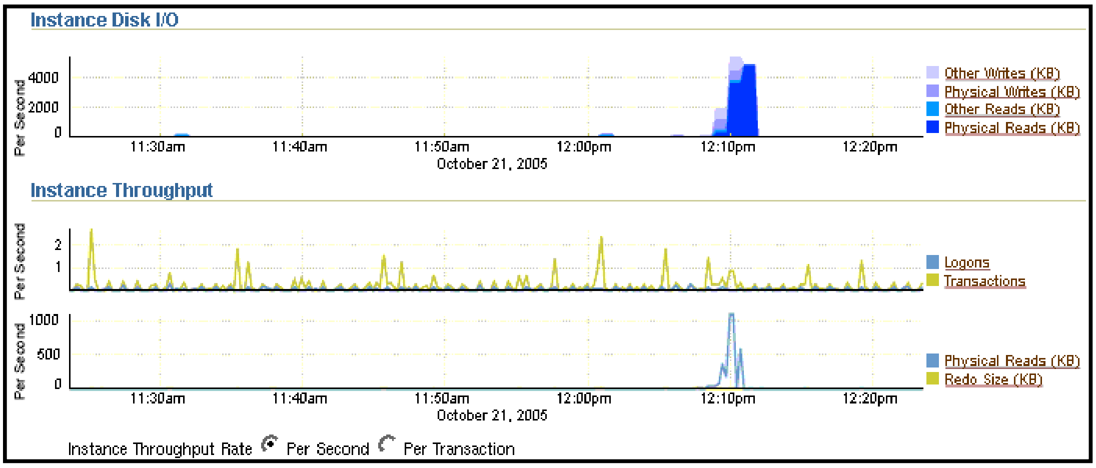
Activity (%)	Session ID	User Name	Program
41.43	132	HR	sqlplus.exe
22.86	159	DBSNMP	OMS
11.43	167	SYS	oracle@edrsr9p1 (DBW0)
10.00	145	SYS	oracle@edrsr9p1 (m000)
4.29	128	SYSMAN	OMS
2.86	141	SYSMAN	OMS
2.86	137	SYSMAN	OMS
1.43	146	SYS	oracle@edrsr9p1 (q000)

Total Sample Count: 70

Performance Monitoring (continued)

When you drill down to a particular wait category, you can view details of specific five-minute intervals, and also see the Top Working SQL and the Top Working Sessions associated with that particular wait event during that time. This enables you to perform after-the-fact analysis of system slowdowns, and determine potential causes.

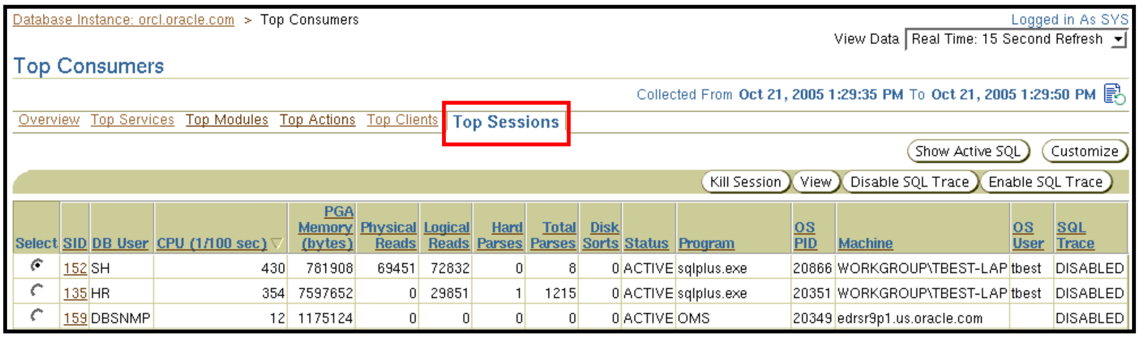
Performance Monitoring



Performance Monitoring (continued)

Instance Disk I/O and Instance Throughput are also reflected on the main Performance tabbed page.

Performance Monitoring: Top Sessions



Performance Monitoring: Top Sessions

If you click one of the I/O category names, you see the Top Consumers page, which lists the top services, modules, actions, clients, and sessions, including critical statistics such as logical and physical read and write count, parse count, and sort count. If you click an I/O category name, the associated statistic is the ordering value for the list.

The table on this page lists the sessions, sorted by CPU usage. This shows that the user SH in session 152 is the greatest consumer of CPU at this particular time.

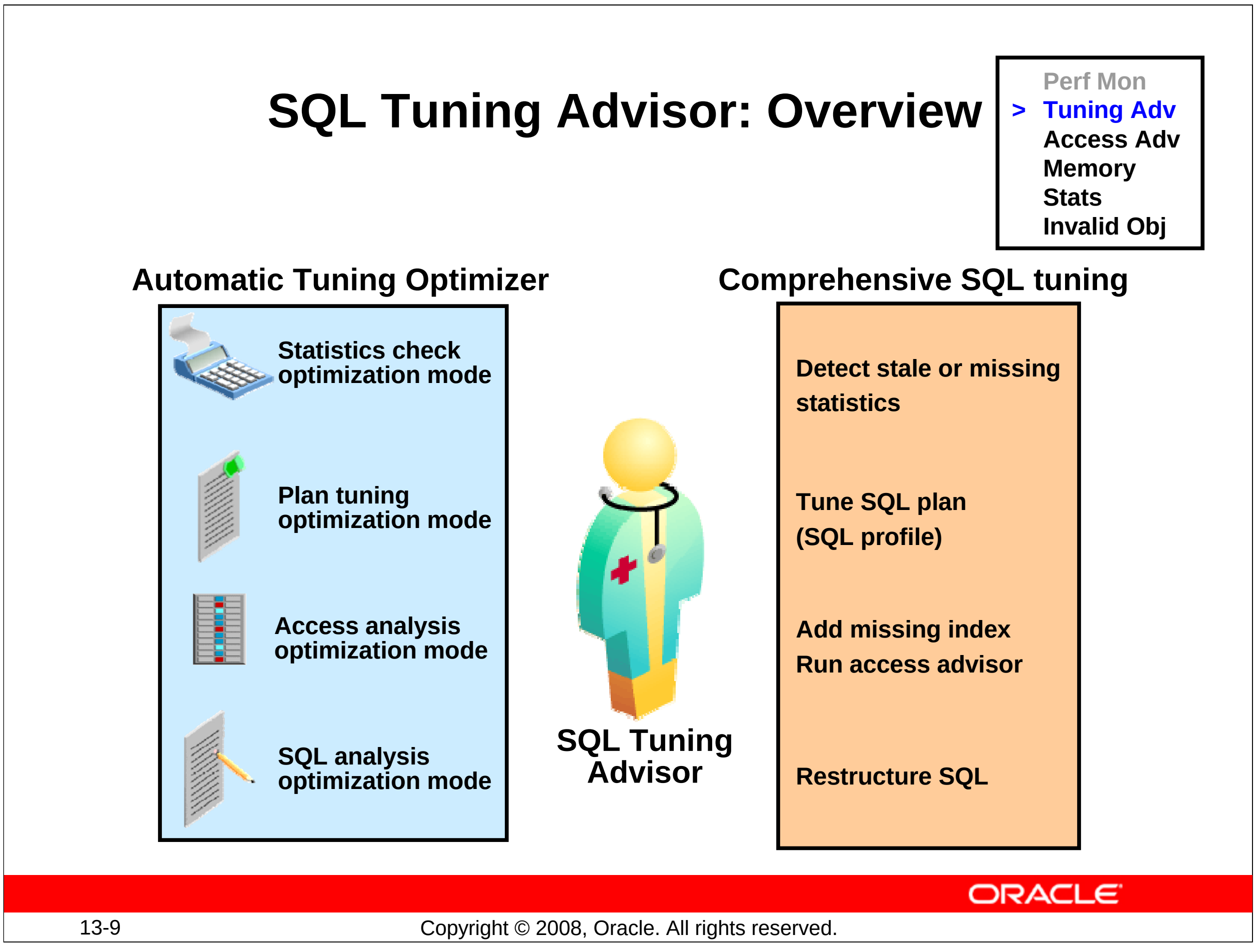
Performance Monitoring: Top Services

Select	Service	Activity (% for the last 5 minutes)	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Time (seconds)	Delta CPU Time (seconds)	Cumulative CPU Time (seconds)	Delta Physical I/O (blocks)	Cumulative Physical I/O (blocks)
	SYS$USERS	37.8	FALSE	1	4874	1	1774	9518	362289
	SYS$BACKGROUND	27.0	FALSE	0	0	0	0	1	328437
	inventory.oracle.com	24.3	FALSE	0	262	0	58	0	10250
	orcl.oracle.com	8.1	FALSE	0	2486	0	1186	0	4977
	hr.oracle.com	2.7	FALSE	1	1124	0	73	5874	55841

ORACLE

Performance Monitoring: Top Services

In multitier systems, where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately. In the example in the slide, there are three services: `inventory`, `orcl`, and `hr`. Regardless of what session was used for a particular request, if it connected via one of these services, then the session's performance data is captured under that service name. It is clear from this listing that, of the three application services, the `inventory` service was the most active during this five-minute interval.

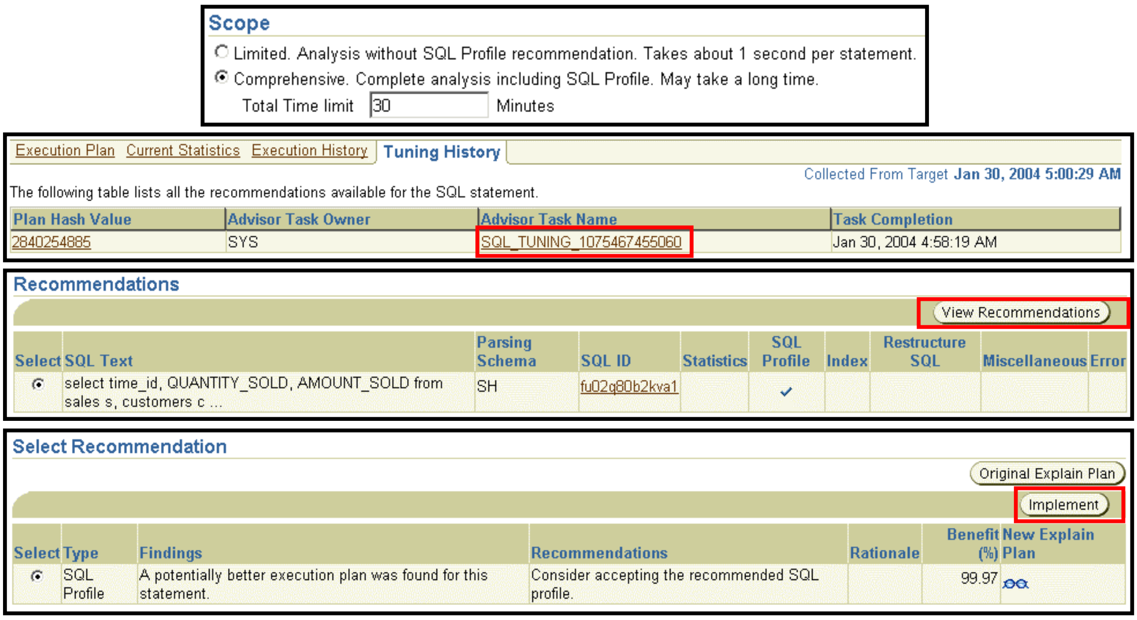


SQL Tuning Advisor: Overview

The SQL Tuning Advisor is the primary driver of the tuning process. It calls the Automatic Tuning Optimizer (ATO) to perform four specific types of analyses:

- **Statistics Analysis:** The Automatic Tuning Optimizer checks each query object for missing or stale statistics, and makes recommendations to gather relevant statistics.
- **SQL Profiling:** The ATO verifies its own estimates and collects auxiliary information to remove estimation errors. It builds a SQL profile using the auxiliary information and makes a recommendation to create it. When a SQL profile is created, it enables the query optimizer to generate a well-tuned plan.
- **Access Path Analysis:** The ATO explores whether a new index can be used to significantly improve access to each table in the query and, when appropriate, makes recommendations to create such indexes.
- **SQL Structure Analysis:** The ATO tries to identify SQL statements that use bad plans and makes relevant suggestions to restructure them. The suggested changes can be syntactic as well as semantic.

SQL Tuning Advisor Options and Recommendations



SQL Tuning Advisor Options and Recommendations

After the SQL Tuning Advisor is launched, Enterprise Manager automatically creates a tuning task, provided that the user has appropriate `ADVISOR` privileges to do so. Enterprise Manager shows the tuning task and automatic default options on the SQL Tuning Options page. On this page, the user can change the automatic defaults for a tuning task. It is important to choose the appropriate scope for the tuning task If you choose the Limited option, then the SQL Tuning Advisor produces recommendations based on statistics check, access path analysis, and SQL structure analysis. No SQL profile recommendation is generated with the Limited option. If you choose the Comprehensive option, the SQL Tuning Advisor produces all the recommendations that the Limited option produces, but it also invokes the optimizer under the SQL profiling mode to build a SQL profile, if applicable. With the Comprehensive option, you can also specify a time limit for the tuning task, which by default is 60 minutes. After you select Run SQL Tuning Advisor, configure your tuning task using the SQL Tuning Options page. Go back to the Top SQL page and click the tuned statement to go to the SQL Details page on which the Recommendations history is displayed. The Recommendations history shows you the completed tuning task. Click the task to see its general recommendation information. Click View Recommendations to see the details about the task .

Using the SQL Tuning Advisor

- Use the SQL Tuning Advisor to analyze SQL statements and obtain performance recommendations.
- Sources for SQL Tuning Advisor to analyze
  - Top SQL: Analyzes the top SQL statements currently active
  - SQL Tuning Sets: Analyzes a set of SQL statements you provide
  - Snapshots: Analyzes a snapshot
  - Baselines: Analyzes a baseline

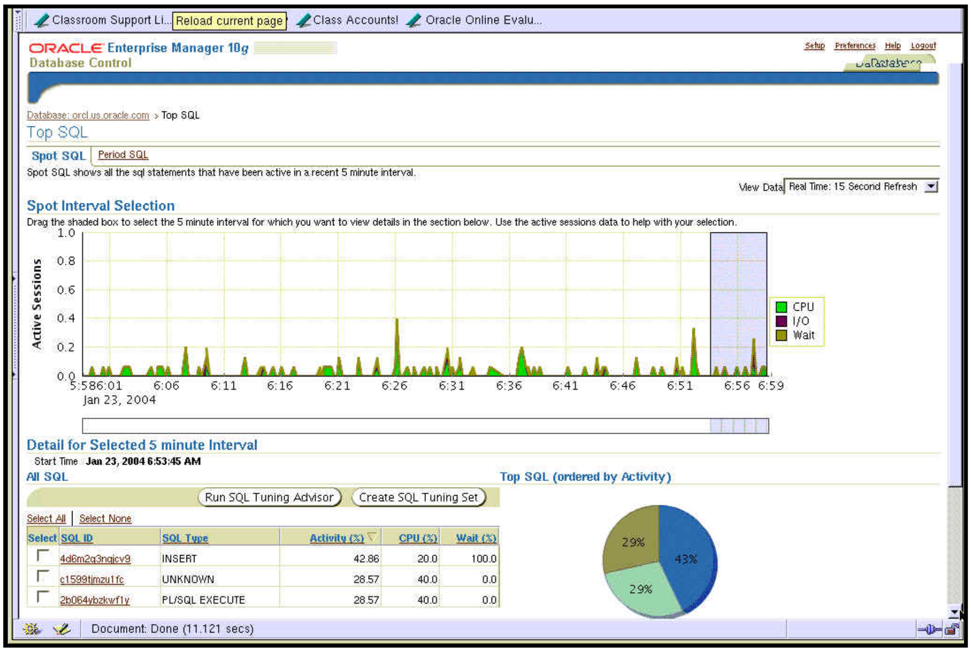
13-11

Using the SQL Tuning Advisor

You can use the SQL Tuning Advisor to analyze SQL statements and obtain performance recommendations. Typically, you run this advisor as an ADDM performance–finding action.

Additionally, you can run the SQL Tuning Advisor when you want to analyze the Top SQL statements consuming the most CPU time, I/O, and memory.

Using the SQL Tuning Advisor: Example



Using the SQL Tuning Advisor: Example

You can invoke the SQL Tuning Advisor by performing the following steps:

1. Click Advisor Central in the Related Links region on the Database home page.
2. Click SQL Tuning Advisor. The SQL Tuning Advisor Links page appears.
   The advisor can be run on one of the following sources:
   - **Top Activity:** The highest load SQL statements that were run during the last hour
   - **Period SQL:** A set of SQL statements that span any 24-hour period
   - **SQL Tuning Sets:** A set of SQL statements you provide
   - **Snapshots:** A set of SQL statements from an AWR snapshot
   - **Preserved Snapshot Sets:** A set of SQL statements from a preserved snapshot set
3. Select Top SQL. Select a five-minute interval to analyze by dragging the shaded box over the target time period. Select one or more statements to analyze during the selected period.
4. Click Run SQL Tuning Advisor. The SQL Tuning Options page appears showing the SQL statements in the interval. Give your task a name and description, select Comprehensive as the scope, and select Immediately for start time. Click OK.

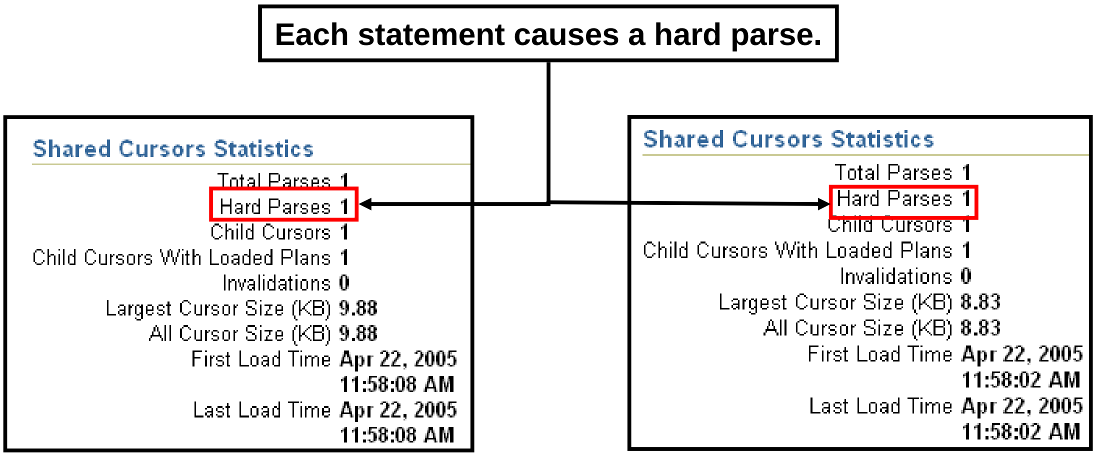
Using the SQL Tuning Advisor: Example (continued)

5. Navigate back to the Advisor Central page. The status of Advisor Tasks is listed under this heading in the results region. Wait until your task status is completed. Check the status by clicking Refresh in your browser. Select your task and click View Result. The SQL Tuning Result page appears.
6. Select the SQL statement and click View Recommendations.

SQL Tuning Advisor: SQL Statistics

```
select count(*) from x
where object_id < 340
```

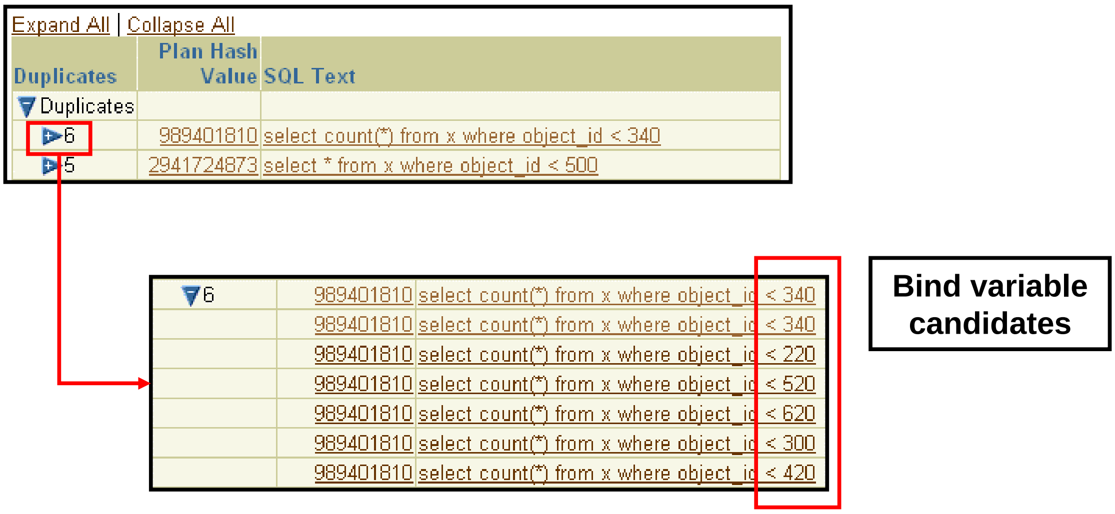
```
select count(*) from x
where object_id < 220
```



SQL Statistics

The SQL Tuning Advisor also shows you the statistics for a cursor that represents a SQL statement. By viewing statistics for each of these two cursors, you can see that each of them causes a hard parse of the statement. This means that the statement is not found to be a match in the Library Cache. This is because of the use of literals instead of bind variables.

SQL Tuning Advisor: Identifying Duplicate SQL



Expand All | Collapse All

Duplicates	Plan Hash Value	SQL Text
Duplicates		
6	989401810	select count(*) from x where object_id < 340
5	2941724873	select * from x where object_id < 500

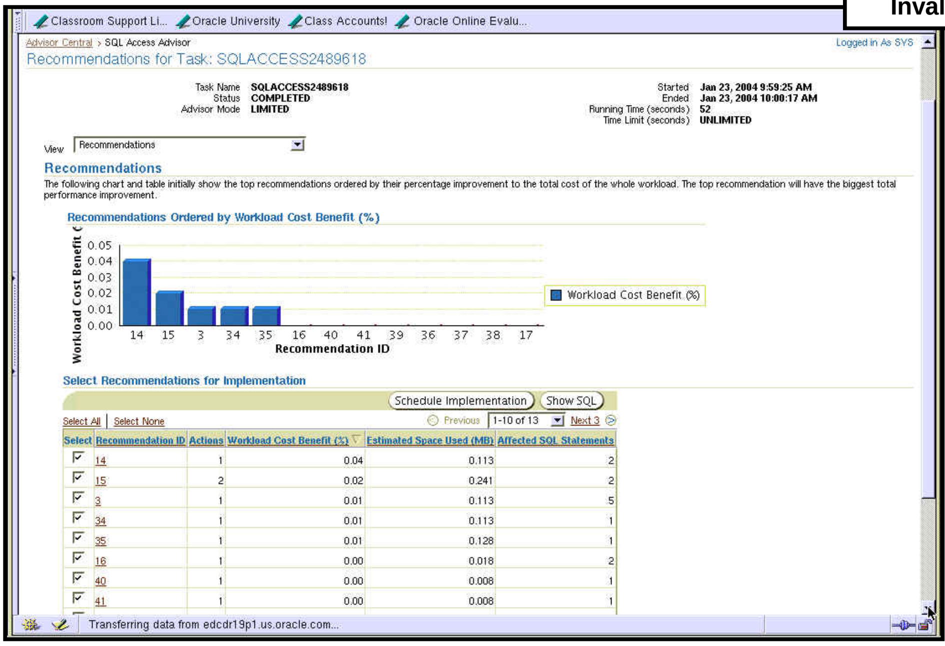
6	989401810	select count(*) from x where object_id < 340
	989401810	select count(*) from x where object_id < 340
	989401810	select count(*) from x where object_id < 220
	989401810	select count(*) from x where object_id < 520
	989401810	select count(*) from x where object_id < 620
	989401810	select count(*) from x where object_id < 300
	989401810	select count(*) from x where object_id < 420

Identifying Duplicate SQL

Duplicate SQL can be identified by clicking Duplicate SQL on the Performance tabbed page. SQL that is determined to be duplicate, except for formatting or literal differences, is listed together. This helps you determine which SQL in your application can be consolidated, thus lowering the requirements on the Library Cache and speeding up the execution of the statement.

Using the SQL Access Advisor

Perf Mon
Tuning Adv
> **Access Adv**
Memory
Stats
Invalid Obj



Using the SQL Access Advisor

You can use the SQL Access Advisor to tune your schema and improve your query performance. This advisor requires that you identify a SQL workload, which is a representative set of SQL statements that access the schema. You can select your workload from different sources including current and recent SQL activity, a SQL repository, or a user-defined workload such as from a development environment.

The SQL Access Advisor may make recommendations such as creating indexes or materialized views to improve your query performance for the given workload.

You can invoke the SQL Access Advisor by performing the following steps:

1. Click Advisor Central in the Related Links region on the Database home page.
2. Click SQL Access to begin a wizard. The SQL Access Advisor: Workload Source page appears.
3. Specify your workload source and click Next. The SQL Access Advisor: Recommendation Options page appears.
4. Specify whether you want the advisor to recommend indexes, materialized views, or both.
5. Specify limited or comprehensive mode. Limited mode runs faster by concentrating on highest cost statements.

Using the SQL Access Advisor (continued)

6. Click Next. The SQL Access Advisor: Schedule page appears. Accept the default of immediate execution, or schedule the execution for a later time.
7. Click Next. The SQL Access Advisor: Review page appears.
8. Review the options you have selected and click Submit to start your job.

Results are posted on the Advisor Central page. The SQL Access Advisor recommendations are ordered by cost benefit. For example, a recommendation may consist of a SQL script with one or more `CREATE INDEX` statements, which you can implement by clicking Schedule Implementation.

Managing Memory Components

Perf Mon
Tuning Adv
Access Adv
> Memory
Stats
Invalid Obj

- **Automatic Shared Memory Management:**
  - Is recommended to simplify management
  - Enables you to specify the total SGA memory through one initialization parameter
  - Enables the Oracle server to manage the amount of memory allocated to the shared pool, Java pool, buffer cache, streams pool, and the large pool
- **Manually setting shared memory management:**
  - Sizes the components through multiple individual initialization parameters
  - Uses the Memory Advisor to make recommendations

ORACLE

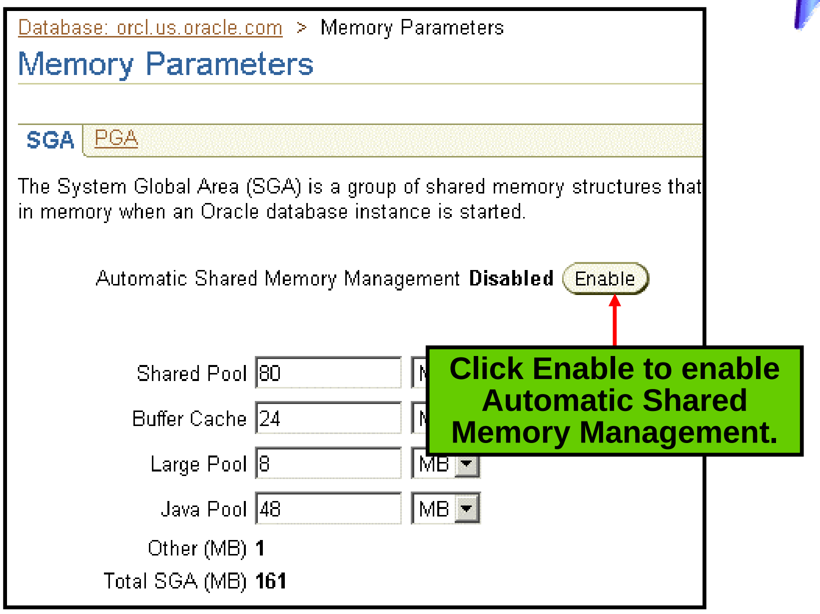
13-18

Managing Memory Components

The SGA comprises several components. The size of many of these components can be managed by the Oracle server through the use of the Automatic Shared Memory Management (ASMM) feature. This simplifies your memory management tasks.

Alternatively, you can manage the size of the components manually by setting other multiple initialization parameters. If, at a later time, the Oracle server notifies you of a performance problem that is related to the size of the Shared Global Area (SGA) or Program Global Area (PGA), you can use the Memory Advisor to determine appropriate new settings. The Memory Advisor can model the effect of parameter changes. You can also specify that the Oracle server must automatically tune the important memory parameters as conditions change. Automatic tuning is recommended.

Enabling Automatic Shared Memory Management (ASMM)



13-19

Enabling Automatic Shared Memory Management

If you have not enabled this feature when you configured your database, you can enable it by performing the following steps:

1. Click Memory Parameters in the Database Configuration region of the Administration page.
2. Click Enable.
   The Enable Automatic Shared Memory Management page appears.
3. Specify the total SGA size. Click OK.

You can increase the total SGA size at a later time by increasing the value of the `SGA_TARGET` initialization parameter, but you cannot set it higher than the value specified by the `SGA_MAX_SIZE` parameter. For more information, refer to the *Oracle Database Administrator's Guide*.

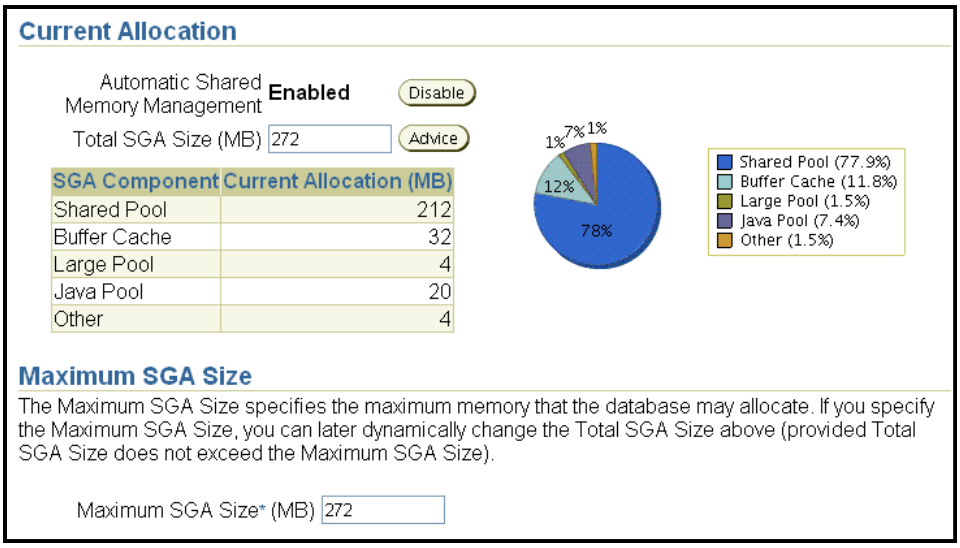
Note: Oracle recommends that you use Automatic Shared Memory Management to simplify your memory management tasks.

Enabling Automatic Shared Memory Management (continued)

If ASMM is enabled, you should not, initially, set initialization parameters for the specific components for which it manages memory. If after seeing the effects of the ASMM allocations, you decide that you want to adjust certain component allocations, then you can specify values for those components. Those values are treated as minimum memory sizes for their respective components. Doing this limits the amount of memory available for automatic adjustment, but the capability is available if your environment requires special sizing not accommodated by ASMM. The initialization parameters of concern are the following:

- `SHARED_POOL_SIZE`
- `LARGE_POOL_SIZE`
- `JAVA_POOL_SIZE`
- `DB_CACHE_SIZE`
- `STREAMS_POOL_SIZE`

Manually Setting Shared Memory Management



SGA Component	Current Allocation (MB)
Shared Pool	212
Buffer Cache	32
Large Pool	4
Java Pool	20
Other	4

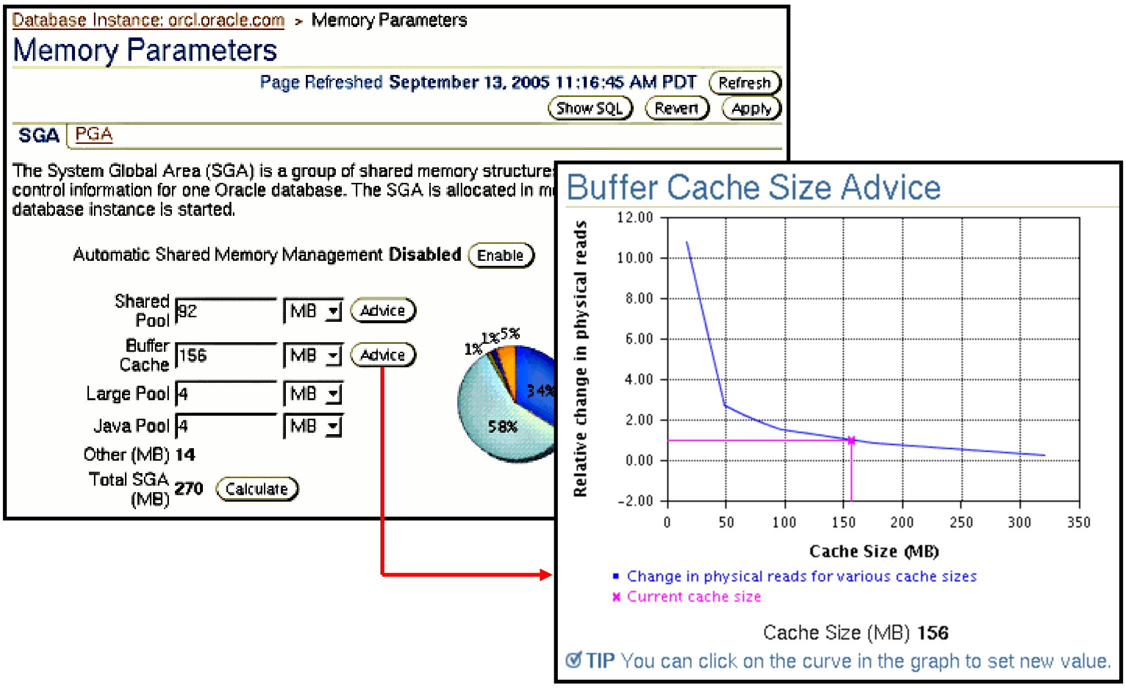
13-21

Manually Setting Shared Memory Management

If you do not use Automatic Shared Memory Management, you must provide values for each component of the SGA on installation and database creation. Perform the following:

1. Access the Memory Parameters page by clicking the Memory Parameters link in the Database Configuration region of the Administration page.
2. Invoke any of the memory advisors by clicking Advice.
3. Click Help to view the online Help for additional information about how the Memory Advisor works.

Using the Memory Advisor



13-22

Using the Memory Advisor

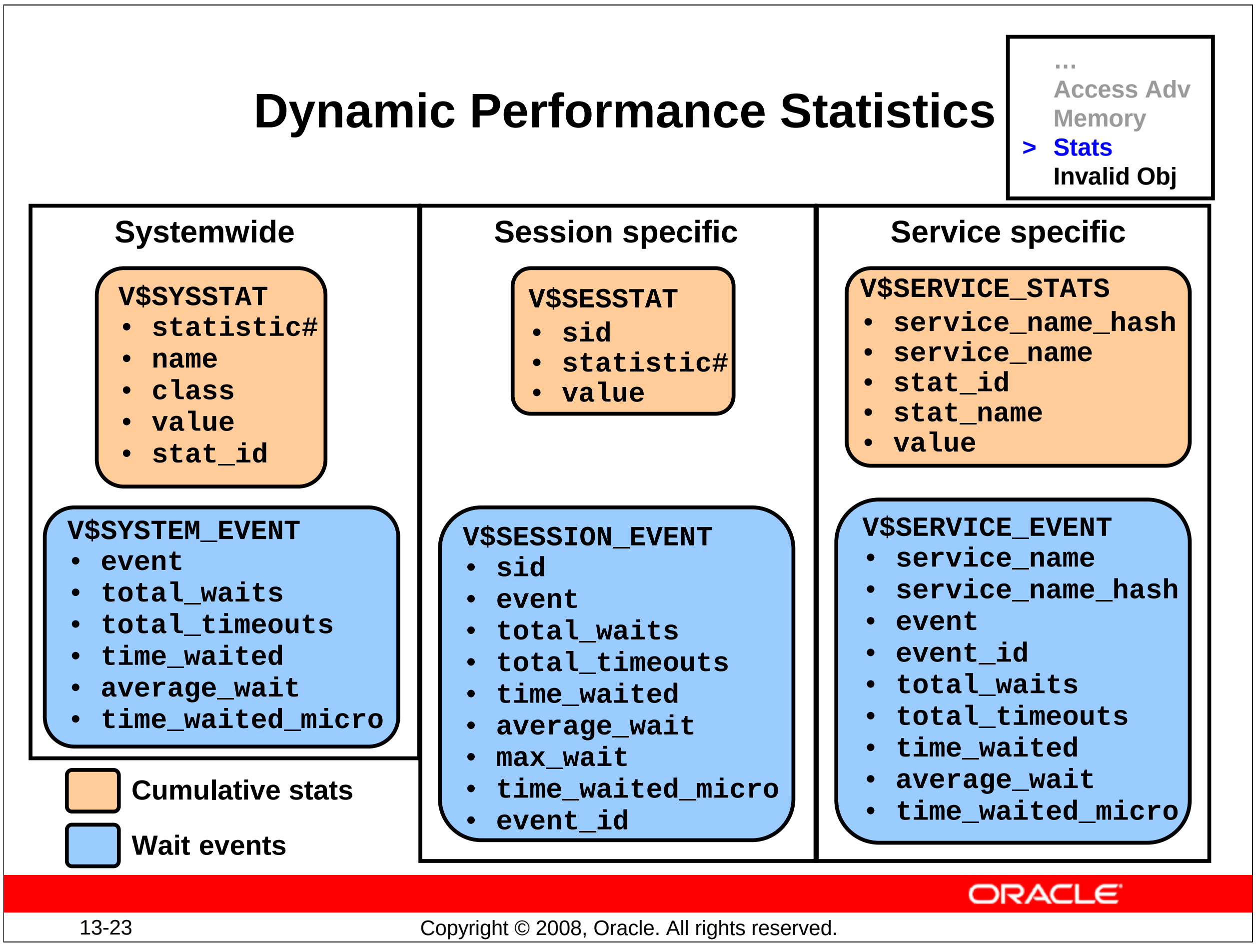
The Memory Advisor helps you tune the size of your memory structures. You can use this advisor only when automatic memory tuning is disabled.

The Memory Advisor comprises three advisors that give you recommendations on the following memory structures:

- Shared pool in the System Global Area (SGA)
- Buffer cache in the SGA
- Program Global Area (PGA)

You can invoke the Memory Advisors by performing the following steps:

1. Click Advisor Central in the Related Links region on the Database home page.
2. Click Memory Advisor on the Advisor Central page. The Memory Parameters page appears. This page provides a breakdown of memory usage for the SGA.
   Note: The Automatic Shared Memory Management setting should be disabled in order to run the advisor.
3. Click Advice next to the Shared Pool value or Buffer Cache value to invoke the respective advisors.
4. Click PGA to access the PGA property page. Click Advice to invoke the PGA Advisor.



Dynamic Performance Statistics

To effectively diagnose performance problems, statistics must be available. Oracle generates many types of statistics for different levels of granularity. At the systemwide level, the session level, and the service level, both wait events and accumulated statistics are computed. The top row of views are the cumulative statistics. The bottom row is made up of the wait event views. When analyzing a performance problem in any of these scopes, you typically look at the change in statistics (delta value) over the period of time you are interested in. All the possible wait events are cataloged in the V$EVENT_NAME view. All statistics are cataloged in the V$STATNAME view; about 360 statistics are available in Oracle database.

Dynamic Performance Statistics (continued)

Displaying Systemwide Statistics

For example:

```
SQL>  SELECT name, class, value FROM v$sysstat;
NAME                              CLASS      VALUE
------------------------------  ------ ----------
...
table scans (short tables)           64     135116
table scans (long tables)            64        250
table scans (rowid ranges)           64          0
table scans (cache partitions)       64          3
table scans (direct read)            64          0
table scan rows gotten               64   14789836
table scan blocks gotten             64     558542
...
```

Systemwide statistics are classified by the tuning topic and the debugging purpose. The classes include general instance activity, redo log buffer activity, locking, database buffer cache activity, and so on.

Troubleshooting and Tuning Views

Instance/Database
V$DATABASE
V$INSTANCE
V$PARAMETER
V$SPPARAMETER
V$SYSTEM_PARAMETER
V$PROCESS
V$BGPROCESS
V$PX_PROCESS_SYSSTAT
V$SYSTEM_EVENT

Memory
V$BUFFER_POOL_STATISTICS
V$LIBRARYCACHE
V$SGAINFO
V$PGASTAT

Disk
V$DATAFILE
V$FILESTAT
V$LOG
V$LOG_HISTORY
V$DBFILE
V$TEMPFILE
V$TEMPSEG_USAGE
V$SEGMENT_STATISTICS

Contention
V$LOCK
V$UNDOSTAT
V$WAITSTAT
V$LATCH

ORACLE

13-25

Troubleshooting and Tuning Views

The slide lists some of the views that you may access to determine the cause of performance problems or analyze the current status of your database.

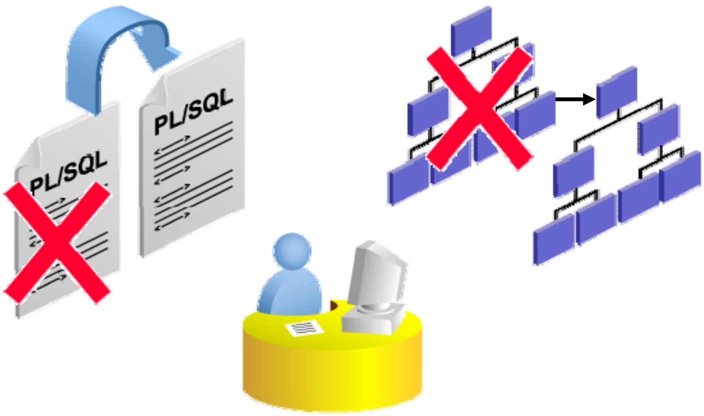
For a complete description of these views, refer to the *Oracle Database Reference Manual*.

Invalid and Unusable Objects

Perf Mon
Tuning Adv
Access Adv
Memory
Stats
> Invalid Obj

Effect on Performance:

- **PL/SQL code objects are recompiled.**
- **Indexes are rebuilt.**



Invalid and Unusable Objects

The current status of certain database objects can be viewed by querying the data dictionary, which is described in the lesson titled "Managing Schema Objects." If you find PL/SQL objects with a status of `INVALID`, the first question that 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 sometime in the past and has recently become invalid, you have two options to fix the problem:

- Do nothing. Most PL/SQL objects automatically recompile if needed when they are called. Users experience a slight delay while the objects recompile. (In most cases, this delay is not even noticeable.)
- Manually recompile the invalid object.

Invalid PL/SQL objects can be manually recompiled by using Enterprise Manager or through SQL commands:

```
ALTER PROCEDURE HR.add_job_history COMPILE;
```

Manually recompiling PL/SQL packages requires two steps:

```
ALTER PACKAGE HR.maintainemp COMPILE;
ALTER PACKAGE 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, and then drops the unusable index. This can be done either by 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 enables 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.

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager to monitor performance
- Tune SQL using the SQL Tuning Advisor
- Tune SQL using the SQL Access Advisor
- Use Automatic Shared Memory Management
- Use the Memory Advisor to size memory buffers
- View performance-related dynamic views
- Troubleshoot invalid and unusable objects

Practice Overview: Monitoring and Improving Performance

This practice covers the following topics:

- **Detecting and repairing unusable indexes**
- **Using the SQL Tuning Advisor**
- **Using the Performance page in Enterprise Manager**

13-29

14

Backup and Recovery Concepts

Objectives

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

- Identify 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 archive log files
- Configure `ARCHIVELOG` mode

Part of Your Job

The administrator's duties are 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**

Part of Your Job

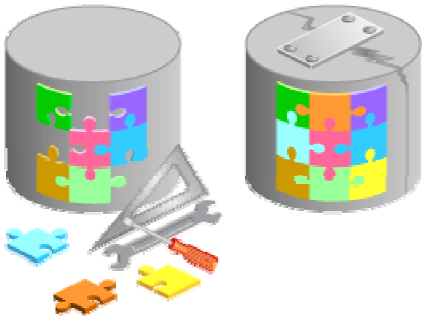
The database administrator's (DBA's) goal is to ensure that the database is open and available when users need it. To achieve that goal, the DBA (usually working with the system administrator):

- Anticipates and works to avoid common causes of failure
- Works to increase the Mean-Time-Between-Failures (MTBF), ensuring that hardware is as reliable as possible, that critical components are protected by redundancy, and that operating system maintenance is performed in a timely manner. The Oracle database provides advanced configuration options to increase MTBF, including:
  - Real Application Clusters (discussed in the *Oracle Database 10g: Real Application Clusters* course)
  - Streams (discussed in the *Oracle Database 10g: Implement Streams* course)
- Decreases the Mean-Time-To-Recover (MTTR), practicing recovery procedures in advance and configuring backups so that 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. Entities that assist in guaranteeing this include:
  - Archive log files (discussed later in this lesson)
  - Standby databases and Oracle Data Guard (discussed in the *Oracle Database 10g: Data Guard Administration* 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



Categories of Failures

Failures can be divided into a few broad categories:

- **Statement failure:** A single database operation (select, insert, update, or 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 (dropping a table or entering bad data) is incorrect .
- **Instance failure:** The database instance shuts down unexpectedly.
- **Media failure:** One or more of the database files are lost (that is, the files have been deleted or the disk has failed).

Statement Failure

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 owner quota. • Add space to tablespace.
Logic errors in applications	Work with developers to correct program errors.

14-5

Statement Failure

When a single database operation fails, a DBA's involvement may be needed to correct errors with user privileges or database space allocation.

User Process Failure

Typical Problems	Possible Solutions
A user performs an abnormal disconnect.	A DBA’s action is not usually needed to resolve user process failures. Instance background processes roll back uncommitted changes and release locks.
A user’s session is abnormally terminated.	
A user experiences a program error that terminates the session.	Watch for trends.

14-6

User Process Failure

User processes that abnormally disconnect from the instance may have uncommitted work in progress that needs to be rolled back. The Process Monitor (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; it also rolls back uncommitted changes and releases any locks that are held by the failed session.

A DBA’s intervention should not be required to recover from user process failure, but the administrator must 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 (which includes teaching them to log out rather than just terminate 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.

14-7

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 Error

Typical Causes	Possible Solutions
A user inadvertently deletes or modifies data.	Roll back or use flashback query to recover.
A user drops a table.	Recover table from the recycle bin.

Oracle LogMiner

User Error

Users may inadvertently delete or modify data. When that happens, the DBA may need to assist the users in recovering from the error. If the users have not yet committed or exited their program, they can simply roll back their operation. If the users have 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 Error (continued)

You can use Oracle LogMiner to query your online redo logs and archived redo logs through a SQL interface. Transaction data may persist in online redo logs longer than it does in undo segments, 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 a table can recover it from the recycle bin by flashing the table back to before the drop. For detailed instructions, see the lesson titled “Performing Flashback.”

If the recycle bin has already been purged, or if the user dropped the table with the `PURGE` option, then 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

Typical Causes	Possible Solutions
Power outage	Restart the instance by using the "startup" command. Recovering from instance failure is automatic, including rolling forward changes in the redo logs and then rolling back any uncommitted transactions.
Hardware failure	
Failure of one of the background processes	
Emergency shutdown procedures	Investigate the causes of failure by using the alert log, trace files, and Enterprise Manager.

ORACLE

14-10

Instance Failure

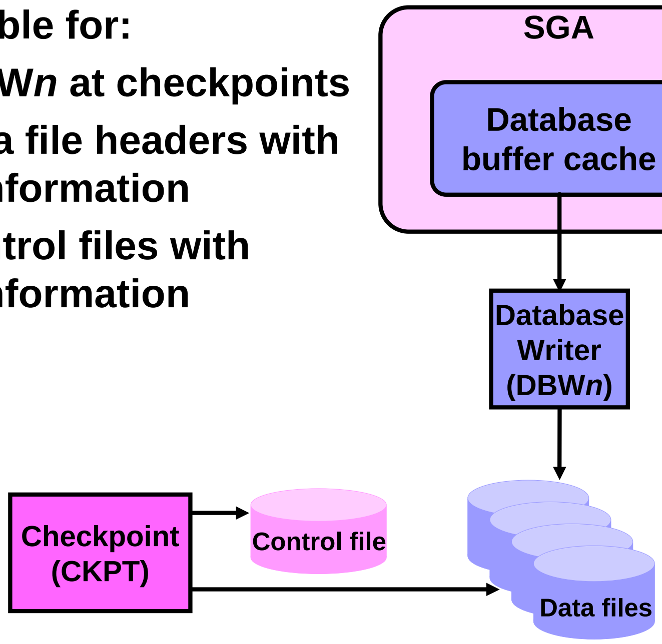
Instance failure occurs when the database instance is shut down before synchronizing all database files. An instance failure can occur because of hardware or software failure or through the use of the emergency `SHUTDOWN ABORT` and `STARTUP FORCE` shutdown commands.

Administrator involvement in recovering from instance failure is usually limited to restarting the instance and working to prevent future occurrences.

Background Processes and Recovery: Checkpoint (CKPT)

CKPT is responsible for:

- **Signaling DBW*n* at checkpoints**
- **Updating data file headers with checkpoint information**
- **Updating control files with checkpoint information**



Background Processes and Recovery: Checkpoint (CKPT)

To understand instance recovery, you need to understand the functioning of certain background processes.

Every three seconds (or more frequently), the CKPT process stores data in the control file to document which modified data blocks DBW*n* has written from the SGA to disk. This is called a "checkpoint." The purpose of a checkpoint is to identify that place in the online redo log file where instance recovery is to begin (which is called the "checkpoint position").

In the event of a log switch, the CKPT process also writes this checkpoint information to the headers of data files.

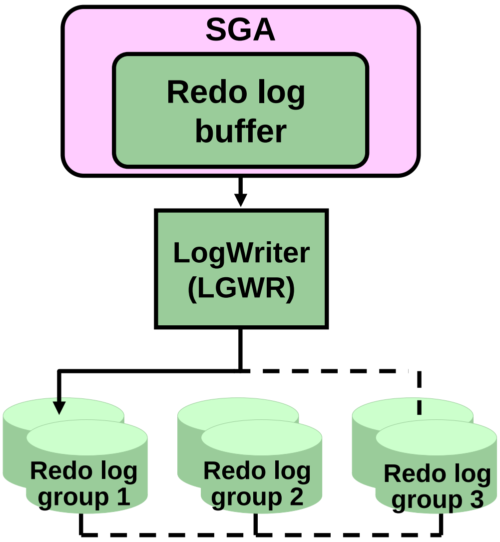
Checkpoints exist for the following reasons:

- To ensure that modified data blocks in memory are written to the disk regularly so that data is not lost in case of a system or database failure
- To reduce the time required for instance recovery. Only the online redo log file entries following the last checkpoint need to be processed for recovery.
- To ensure that all committed data has been written to data files during shutdown

The checkpoint information written by the CKPT process includes checkpoint position, system change number, location in the online redo log file to begin recovery, information about logs, and so on.

Note: The CKPT process does not write data blocks to the disk or redo blocks to the online redo log files.

Background Processes and Recovery: Redo Log Files and LogWriter



Redo log files:

- Record changes to the database
- Should be multiplexed to protect against loss

LogWriter writes:

- At commit
- When one-third full
- Every three seconds
- Before DBW*n* writes

Background Processes and Recovery: Redo Log Files and LogWriter

Redo log files record changes to the database as a result of transactions and internal Oracle server actions. (A transaction is a logical unit of work, consisting of one or more SQL statements run by a user.) Redo log files protect the database from the loss of integrity because of system failures caused by power outages, disk failures, and so on. Redo log files must be multiplexed to ensure that the information stored in them is not lost in the event of a disk failure.

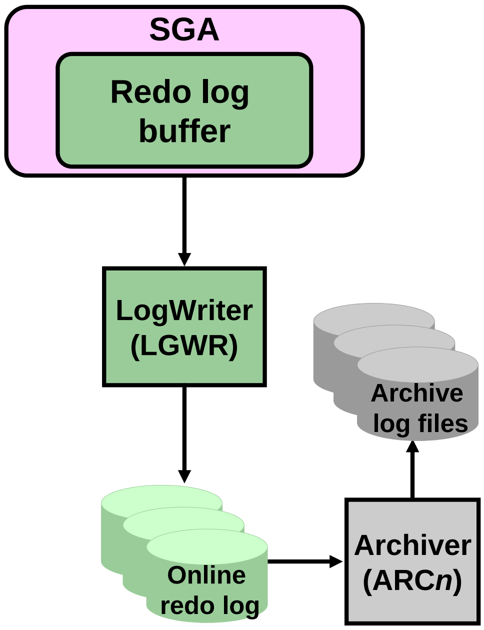
The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group, and each group is identified by a number. The LogWriter (LGWR) process writes redo records from the redo log buffer to all members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

Best practice tip: If possible, multiplexed redo log files should reside on different disks.

Background Processes and Recovery: Archiver (ARC*n*)

Archiver (ARC*n*):

- **Is an optional background process**
- **Automatically archives online redo log files when `ARCHIVELOG` mode is set for the database**
- **Preserves the record of all changes made to the database**



14-13

Background Processes and Recovery: Archiver (ARC*n*)

ARC*n* is an optional background process. However, it is crucial to recovering a database after the loss of a disk. When an online redo log group gets filled, the Oracle instance begins writing to the next online redo log group. The process of switching from one online redo log group to another is called a log switch. The ARC*n* process initiates archiving of the filled log group at every log switch. It automatically archives the online redo log group before the log group can be reused, so all the changes made to the database are preserved. This enables recovery of the database to the point of failure even if a disk drive is damaged.

One of the important decisions that a DBA has to make is whether to configure the database to operate in `ARCHIVELOG` mode or in `NOARCHIVELOG` mode.

- In `NOARCHIVELOG` mode, the online redo log files are overwritten each time a log switch occurs.
- In `ARCHIVELOG` mode, inactive groups of filled online redo log files must be archived before they can be used again.

Note: `ARCHIVELOG` mode is essential for most backup strategies (and is very easy to configure).

Instance Recovery

Instance or crash recovery:

- **Is caused by attempts to open a database whose files are 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.**

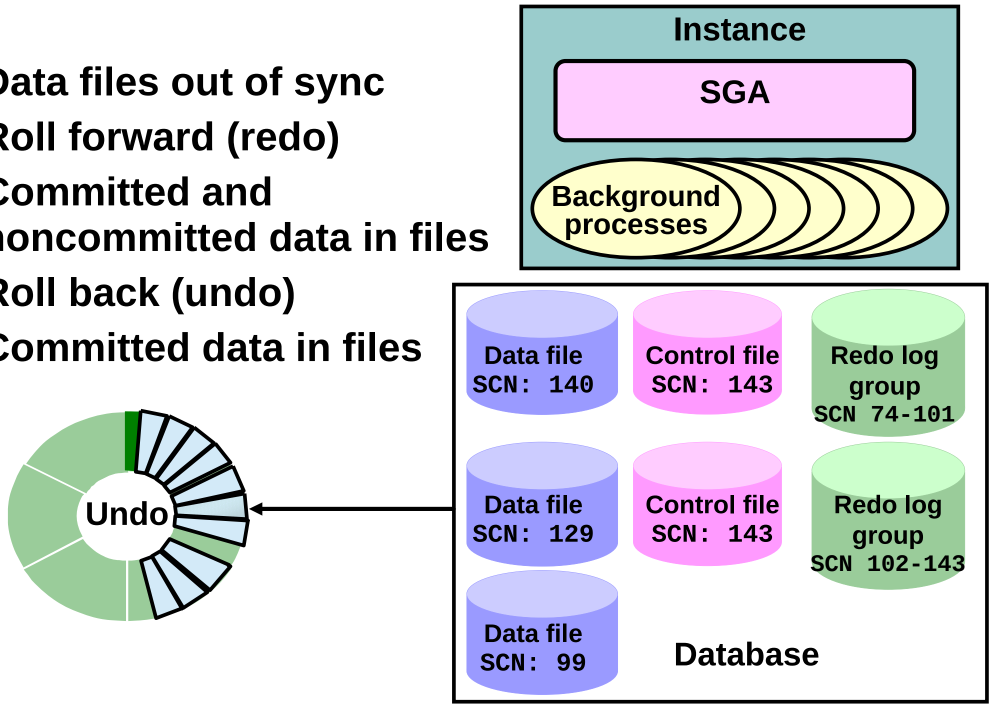
14-14

Instance Recovery

Oracle Database 10*g* automatically recovers from instance failure. All that the DBA needs to do is start the instance normally. The instance mounts the control files and then attempts to open the data files. When it discovers that 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 is also rolled forward) rolls back any uncommitted transactions.

Phases of Instance Recovery

1. Data files out of sync
2. Roll forward (redo)
3. Committed and noncommitted data in files
4. Roll back (undo)
5. Committed data in files



Phases of Instance Recovery

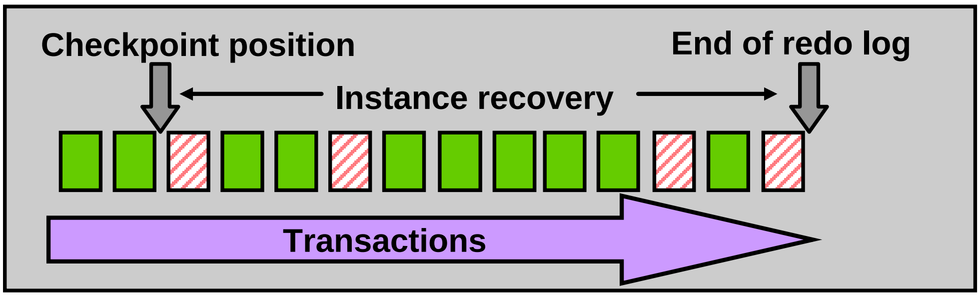
For an instance to open a data file, the system change number (SCN) contained within the data file's header must match the current SCN stored in the database's control files.

If the numbers do not match, the instance applies redo data 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 logs are applied, *all* transactions are applied to bring the database up to the state as of the time of failure. This usually includes transactions that are in progress but have 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 contain only committed data.

Tuning Instance Recovery

- During instance recovery, the transactions between the checkpoint position and the end of redo log must be applied to data files.
- You tune instance recovery by controlling the difference between the checkpoint position and the end of redo log.



Tuning Instance Recovery

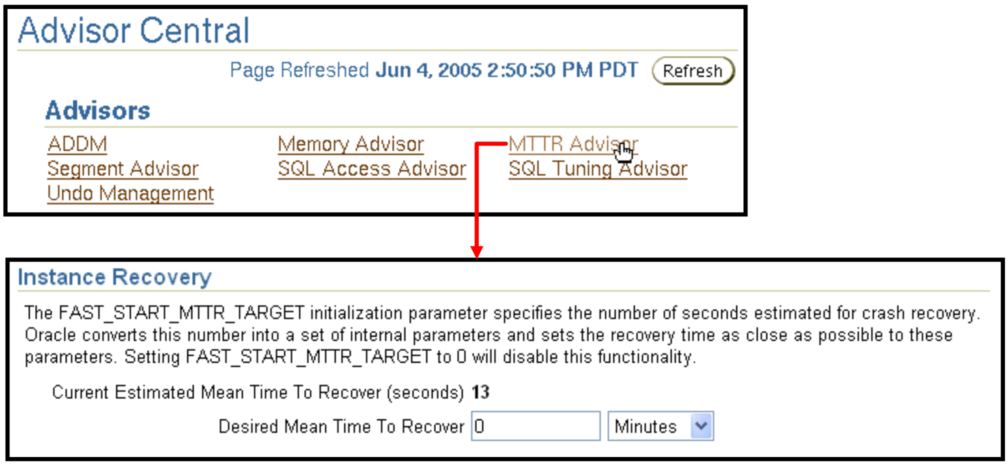
Transaction information is always recorded in the redo log groups before the instance returns `commit complete` for a transaction. The information in the redo log groups guarantees that the transaction can be recovered in case of a failure. The transaction information also needs to be written to the data file. The data file write usually happens sometime after the information is recorded in 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.)

Every three seconds, the checkpoint process records information in the control file about the checkpoint position in the redo log. Therefore, the Oracle database knows that all redo log entries recorded before this point are not necessary for database recovery. In the graphic in the slide, the striped blocks have not yet been written to the disk.

The time required for instance recovery is the time required to bring data files from their last checkpoint to the latest SCN recorded in the control file. The administrator controls that time by setting an MTTR target (in seconds) and through the size of 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.
- The default value is 0 (disabled).
- The maximum value is 3,600 seconds (one hour).



14-17

MTTR Advisor

For assistance in setting the MTTR target, select Enterprise Manager > Administration > Advisor Central > MTTR Advisor. This advisor converts the `FAST_START_MTTR_TARGET` value into several parameters to enable instance recovery in the desired time or as close to it as possible.

Explicit setting of the `FAST_START_MTTR_TARGET` parameter to 0 disables automatic checkpoint tuning. Explicit setting of the `FAST_START_MTTR_TARGET` parameter to a value other than 0 also enables the Redo Log Advisor.

The `FAST_START_MTTR_TARGET` parameter must be set to a value that supports the service level agreement for your system. A small value for MTTR target increases I/O overhead because of additional data file writes (impacting performance). However, if you set the MTTR target too large, then 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.
Failure of disk controller	2. If necessary, inform the database about a new file location.
Deletion or corruption of database file	3. If necessary, recover the file by applying redo information.

Media Failure

Oracle Corporation defines media failure as any failure that 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 that your database can be recovered from media failure, follow best practices as outlined in the next few pages.

Configuring for Recoverability

To configure your database for maximum recoverability, you must:

- **Schedule regular backups**
- **Multiplex control files**
- **Multiplex redo log groups**
- **Retain archived copies of redo logs**

14-19

Configuring for Recoverability

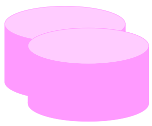
To provide the best protection for your data, you must:

- **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 a 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 at least three copies.
- **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 that data is likely to be lost. Ensure that there are at least two copies of each redo log group, if possible, under different disk controllers.
- **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. With the default setting, the database can overwrite redo information after 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. It is suggested that your database has:

- **At least two copies (Oracle recommends three) of the control file**
- **Each copy on a separate disk**
- **At least one copy on a separate disk controller**



Control files

ORACLE

Control Files

A 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 opened. Without this file, the database cannot be mounted and recovery or re-creation of the control file is required. Your database must 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 is created with the Database Configuration Assistant (DBCA) using Oracle Managed Files (OMF), you have two control files. If you do not use OMF, there are three control files.

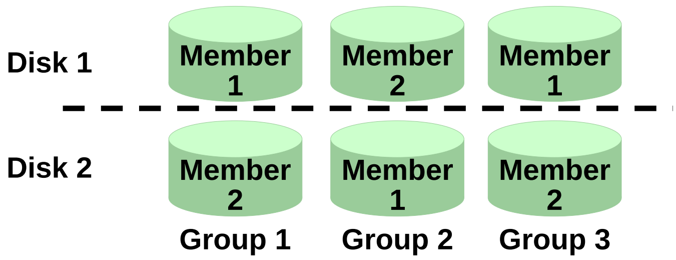
The loss of a single control file causes the instance to fail because all control files must be available at all times, but recovery is a simple matter of copying one of the other control files. The loss of all control files is slightly more difficult to recover from, but is not usually catastrophic.

Redo Log Files

Multiplex redo log groups to protect against media failure and loss of data. It is suggested that redo log groups have:

- **At least two members (files) per group**
- **Each member on a separate disk drive**
- **Each member on a separate disk controller**

Note: Performance is heavily influenced by writing to redo logs.



ORACLE

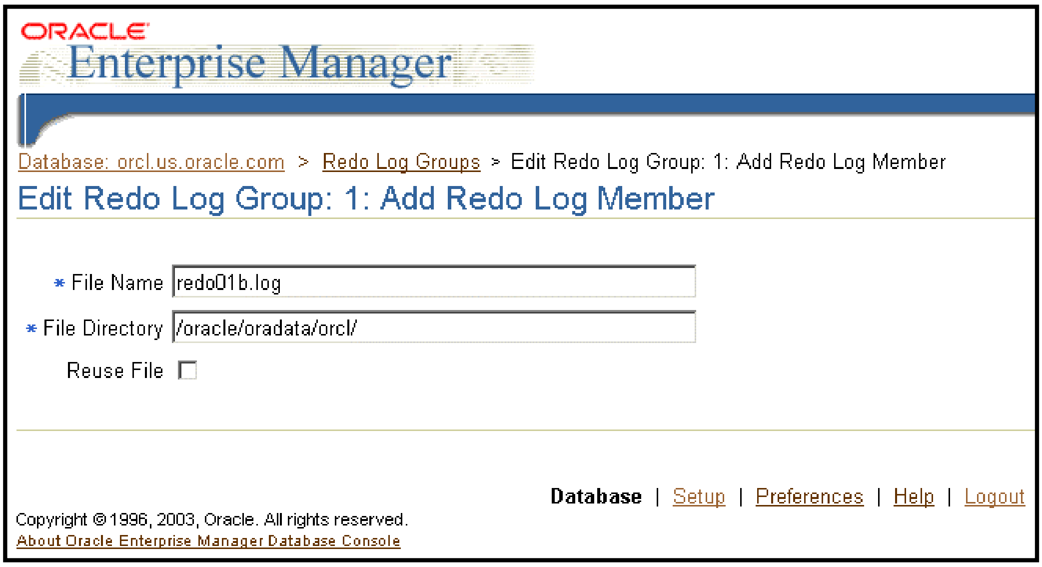
Redo Log Files

Redo log groups are made up of 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 or controllers so that no single equipment failure destroys an entire log group.

The loss of an entire log group is one of the most serious possible media failures because it can result in loss of data. The loss of a single member within a multiple-member log group is trivial and does not affect database operation, other than causing an alert to be published in the alert log. Recovery from the loss of an entire log group requires advanced recovery techniques and is discussed in *Oracle Database 10g: Administration 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 must 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



14-22

Multiplexing the Redo Log

You can multiplex your redo log by adding a member to an existing log group. To add a member to a redo log group (with open database and no impact on user performance), perform the following steps:

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 region, click Add. The Add Redo Log Member page appears.
4. Enter the file name and the file directory. Click Continue.

   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.

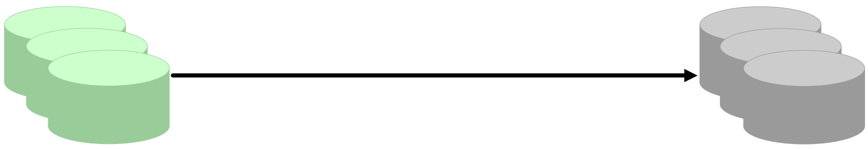
Repeat these steps for every existing group.

When you add the redo log member to a group, the added log file status is set to `INVALID`, as can be seen in the `v$logfile` view. 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`.

Archive Log Files

To preserve redo information, create archived copies of redo log files by performing the following steps.

1. **Specify archive log file naming convention.**
2. **Specify one or more archive log file locations.**
3. **Switch the database to `ARCHIVELOG` mode.**



Archive 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.

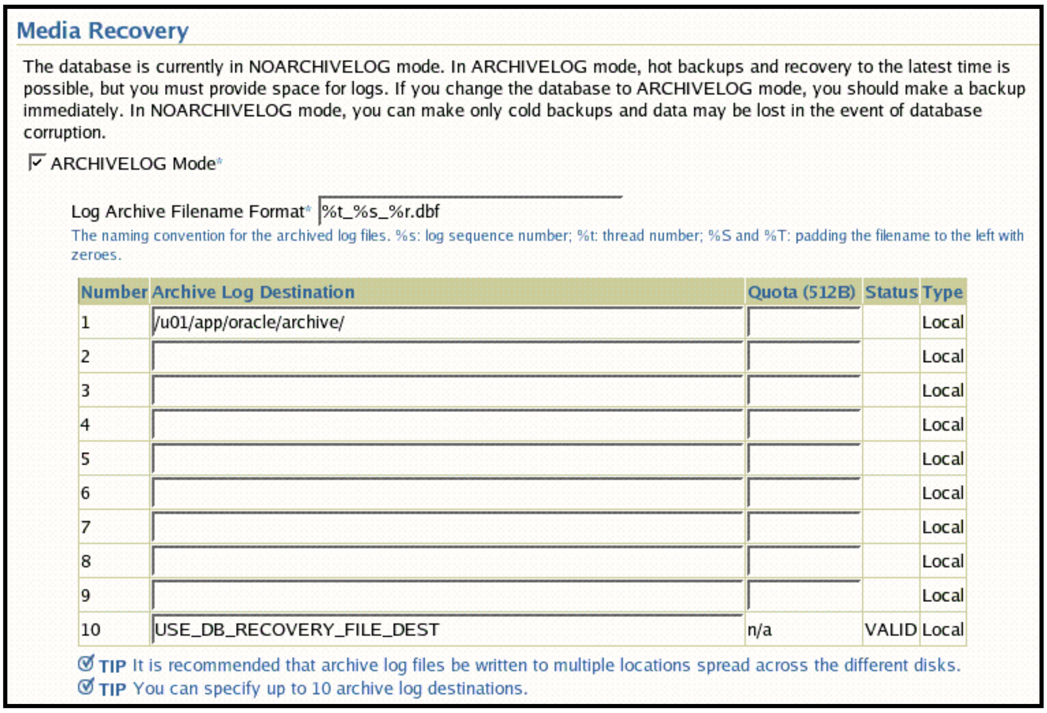
To configure your database for maximum recoverability, you must instruct the database to make a copy of the online redo log group before allowing it to be overwritten. These copies are known as archived logs. To facilitate the creation of archive log files, you must perform the following steps:

1. Specify a naming convention for your archive logs.
2. Specify a destination or destinations for storing your archive logs.
3. Place the database in `ARCHIVELOG` mode.

Note

- Steps 1 and 2 are not necessary if you are using a flash recovery area.
- The destination must exist before placing the database in `ARCHIVELOG` mode. When a directory is specified as a destination, there should be a slash at the end of the directory name.

Archive Log File: Naming and Destinations



Archive Log File: Naming and Destinations

Configure archive log file names and destinations by clicking Configure Recovery Settings on the Maintenance page.

Each archive log file must have a unique name to avoid overwriting older log files. Specify the naming format as shown in the slide. 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:** Includes resetlogs ID to ensure 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 must be included if multiple databases share the same archive log destination.

Archive 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 must end in a slash "/" (or a backslash "\" if using Windows).

Archive Log File: Naming and Destinations (continued)

The default destination (number 10) sends archive 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 Flash Recovery Area Location. If you do not want archives sent to this location, delete USE_DB_RECOVERY_FILE_DEST.

To change recovery settings, you must be connected as SYSDBA or SYSOPER.

ARCHIVELOG Mode

- To place the database in ARCHIVELOG mode, perform the following steps:
  1. Select the ARCHIVELOG Mode check box.
  2. Click Apply. The database can be set to ARCHIVELOG mode only from the MOUNT state.
  3. Click Yes when asked whether you want to restart the database.
  4. Back up your database.
- Databases in ARCHIVELOG mode have access to the full range of backup and recovery options.

ARCHIVELOG Mode

Placing the database in ARCHIVELOG mode prevents redo logs from being overwritten until they have been archived. The following SQL command is used to place the database in ARCHIVELOG mode:

```
SQL> ALTER DATABASE ARCHIVELOG;
```

This command can be issued only while the database is in the MOUNT state, and so the instance must be restarted to complete this last step. You are prompted for operating system and database credentials during the restart of the database. The database credentials *must* be for a user with the SYSDBA privileges.

After the instance is restarted, the changes that you have made to the archive processes, log format, and log destinations are in effect.

With the database in NOARCHIVELOG mode (the default), recovery is possible only until the time of the last backup. All transactions made after that backup are lost.

In ARCHIVELOG mode, recovery is possible until the time of the last commit. Most production databases are run in ARCHIVELOG mode.

Note: Back up your database after switching to ARCHIVELOG mode because your database is only recoverable from the last backup taken in that mode.

Summary

In this lesson, you should have learned how to:

- Identify 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 archive log files
- Configure `ARCHIVELOG` mode

Practice Overview: Configuring for Recoverability

This practice covers the following topics:

- Multiplexing control files
- Multiplexing redo log groups
- Placing your database in `ARCHIVELOG` mode
- Ensuring that redundant archive logs are created

15

Performing Database Backups

Objectives

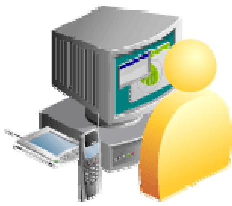
After completing this lesson, you should be able to do the following:

- Create consistent database backups
- Back up your database without shutting it down
- Create incremental backups
- Automate database backups
- Monitor the flash recovery area

Backup Solutions: Overview

Backups can be performed by using:

- **Recovery Manager**
- **Oracle Secure Backup**
- **A user-managed scenario**



Backup Solutions: Overview

As you see in the remainder of this lesson, Recovery Manager (RMAN) is the recommended method of backing up your Oracle database.

Oracle Secure Backup complements existing functionality by adding backup to tape and network backup capabilities.

User-managed backups are based on scripts, which a DBA would have to write. This option is being phased out because it is more labor intensive.

Oracle Secure Backup

- Oracle Secure Backup and RMAN provide an end-to-end backup solution for Oracle environments:
  - Centralized tape backup management for file system data and the Oracle database
  - Most well-integrated media management layer for RMAN backups
  - Backup of any data anywhere on the network
- A single technical support resource for the entire backup solution expedites problem resolution.
- This ensures reliable data protection at lower cost and complexity.

ORACLE

15-4

Oracle Secure Backup

Oracle’s current backup and recovery product for the database is Recovery Manager. Oracle Secure Backup complements existing functionality in the following ways:

- **Complete backup solution:** Oracle Secure Backup provides data protection for the database and nondatabase data to protect the whole Oracle environment.
- **Media management:** Oracle Secure Backup provides the media management layer for RMAN database backups to tape. Before Oracle Secure Backup, customers had to purchase expensive third-party media management products offering integration with RMAN tape backups.
- **Backup anywhere on the network:** Oracle Secure Backup backs up data from multiple network-attached computer systems to tertiary storage resources on the network. Oracle Secure Backup supports diverse configurations of servers, clients, Network Attached Storage (NAS) servers, and tertiary storage devices and protects network storage environments.

The combination of RMAN and Oracle Secure Backup provides an end-to-end backup solution, entirely within the Oracle product stack. This allows for better customer support because Oracle Corporation is responsible for the entire backup solution.

User-Managed Backup

A user-managed scenario:

- **Is a manual process of tracking backup needs and status.**
- **Requires the DBA to write scripts.**
- **Requires that database files be put in the correct mode for backup.**
- **Relies on operating system commands to make backups of files.**

User-Managed Backup

A user-managed backup entails the writing of scripts to perform the backup. There are several scenarios that can be run, and scripts must be written to handle them. The following are some of the steps that the scripts must take:

- Query `v$datafile` to determine the data files that need to be backed up and their current state.
- Query `v$logfile` to identify the online redo log files.
- Query `v$controlfile` to identify the control file to back up.
- Place each tablespace in online backup mode.
- Query `v$backup` to see what data files are part of a tablespace that has been placed in online backup mode.
- Issue operating system copy commands to copy the data files to the backup location.
- Bring each tablespace out of online backup mode.

Terminology

- **Backup strategy may include:**
  - **The entire database (whole)**
  - **A portion of the database (partial)**
- **Backup type may indicate inclusion of:**
  - **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)**

ORACLE

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 each data block that contains data and which is within the files being backed up.

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 and 1). A level 0 or baseline backup, like a full backup, contains all data blocks. A level 1 incremental back up can be one of two types: cumulative or differential. A cumulative backup backs up all changes since the last level 0 backup. A differential backup backs up all changes since the last incremental backup (which could be either a level 0 or a level 1).

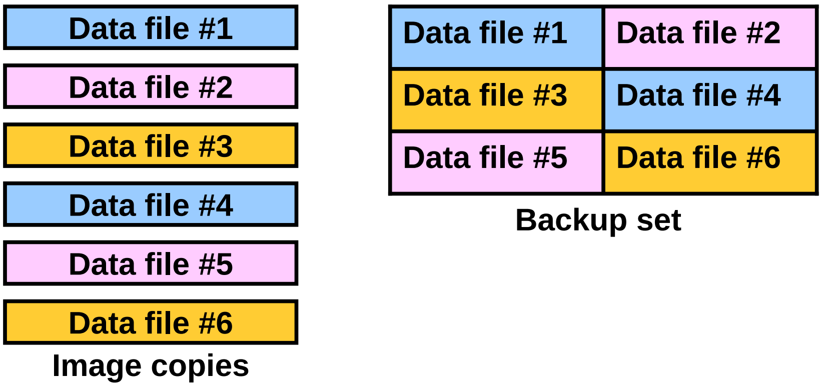
Offline backups (also known as **consistent backups**) are taken while the database is not open. They are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

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 by using operating system commands).

Backup sets are collections of one or more binary files that contain 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 the disk or tape. Backup sets can be compressed to further reduce the space requirements of the backup.

Image copies must be backed up to the disk. Backup sets can be sent either to the disk or directly to the 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 the tape. With backup sets, the entire backup set must be retrieved from the 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 up every single data block, 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.

Recovery Manager (RMAN)

- Enterprise Manager uses Recovery Manager (RMAN) to perform backup and recovery operations.
- RMAN:
  - Is a command-line client for advanced functions
  - Has powerful control and scripting language
  - Has a published API that enables interface with most popular backup software
  - Backs up data, control, archived log, and server parameter files
  - Backs up files to the disk or tape

15-8

Recovery Manager (RMAN)

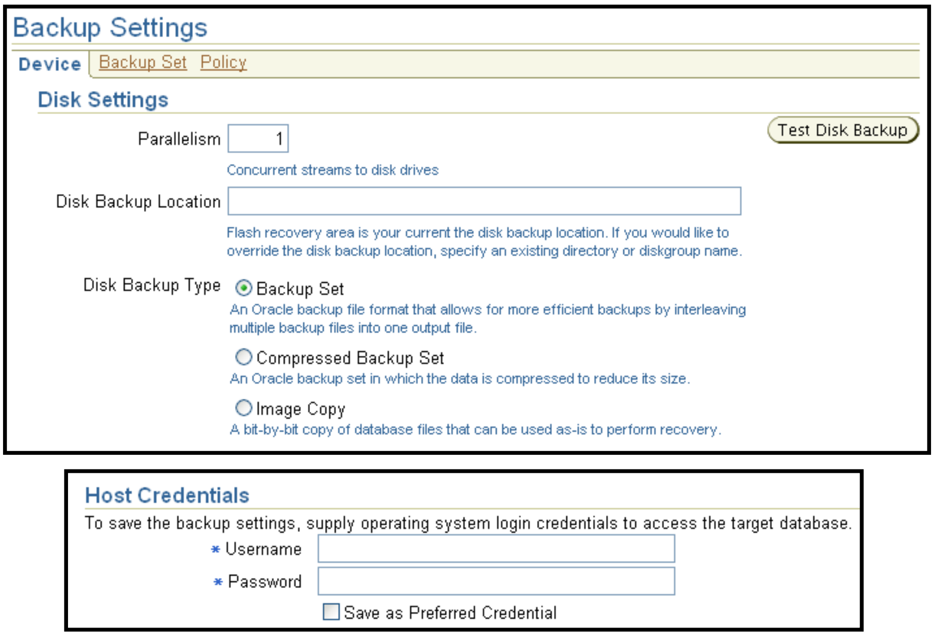
RMAN is the component of Oracle Database 10*g* that is used to perform backup and recovery operations. It can make consistent and inconsistent backups, perform incremental or full backups, and back up either the whole database or a portion of it.

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 the disk for quick recovery or place them on the tape for long-term storage. In order for RMAN to store backups on the tape, an interface to the tape device known as a Media Management Library (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 about 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

15-9

Configuring Backup Settings

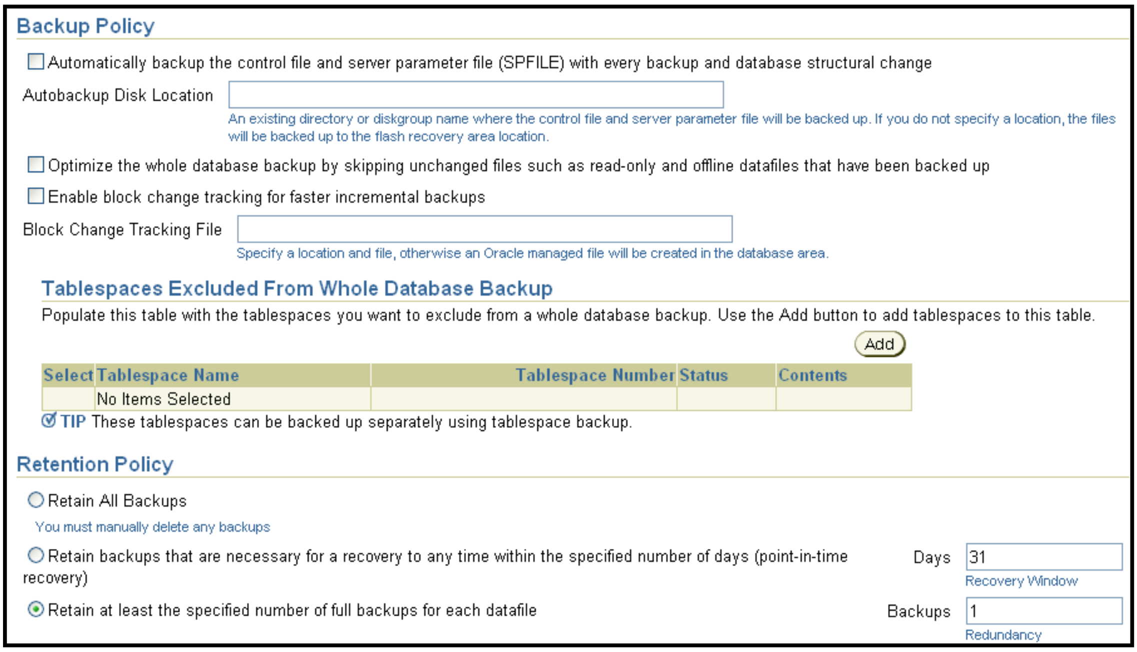
Navigate to the Maintenance page and click Configure Backup Settings. On this property page, you can manage the persistent backup settings that are used for creating backups. There are separate settings for the disk and the tape. Tape settings depend on the media management library capabilities. Disk 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. This maximum is used to divide the backup sets into what is referred to as “backup pieces,” making archiving easier.

Host credentials are required for Enterprise Manager to save any changes to the backup settings.

Configuring Backup Settings



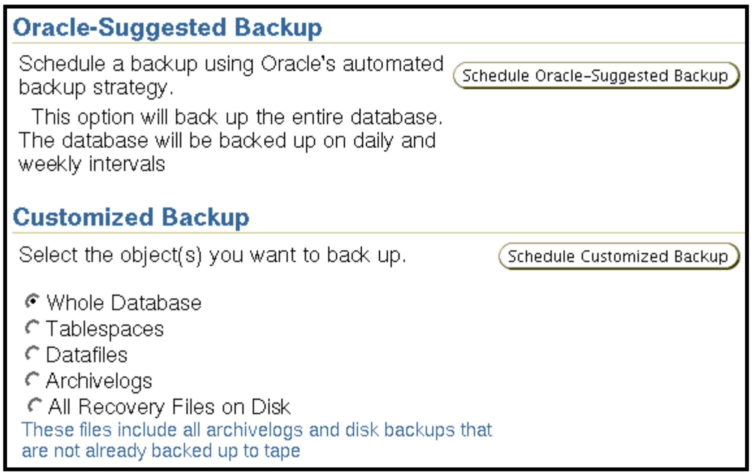
Configuring Backup Settings (continued)

Click the Policy tab to:

- Automatically back up the control file and server parameter file (`SPFILE`) with each backup. You can also specify a location for these backups, if you do not 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 enables 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 be easily re-created (such as indexes or data that is batch-loaded frequently).
- Specify a 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

Select whole or partial database backup.

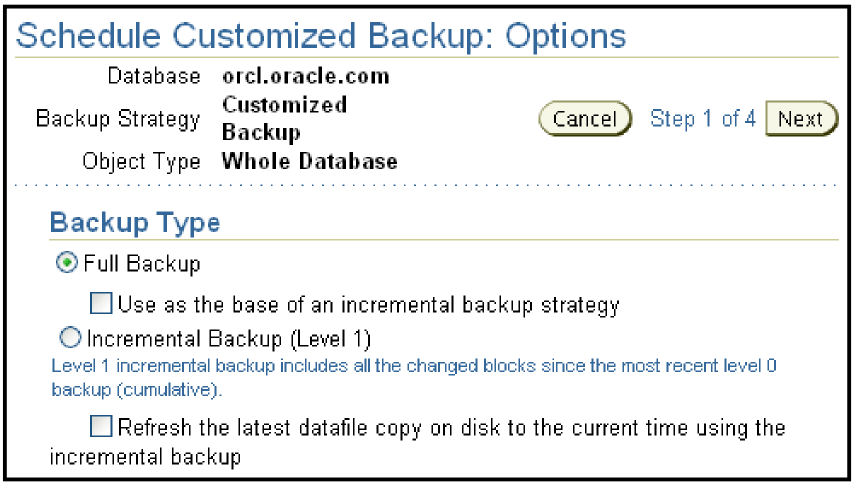


Scheduling Backups: Strategy

Click Schedule Backup in the Backup/Recovery region 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 backup, which is performed online. This is a baseline incremental level 0 backup. The automated backup strategy then schedules incremental level 1 backups for each following day.

By selecting Customized, you gain access to a wider range of configuration options. Select the objects that you want to back up—the whole database (the default) or individual tablespaces, data files, archived logs, or any Oracle backups currently residing on the disk (to move them to the tape).

Scheduling Backups: Options



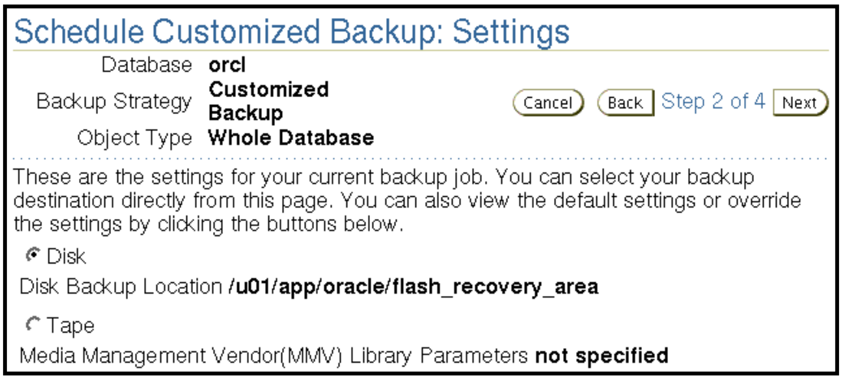
15-12

Scheduling Backups: Options

Select full or incremental backup type. 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 backup. If using image copies, you can select the “Refresh the latest datafile copy on disk to the current time using the incremental backup” check box to update the existing backup rather than creating a new image copy.

Click “Delete obsolete backups” to remove any backups that fall outside the retention policy that you configured earlier. RMAN automatically removes obsolete backups if you are backing up to the flash recovery area.

Scheduling Backups: Settings

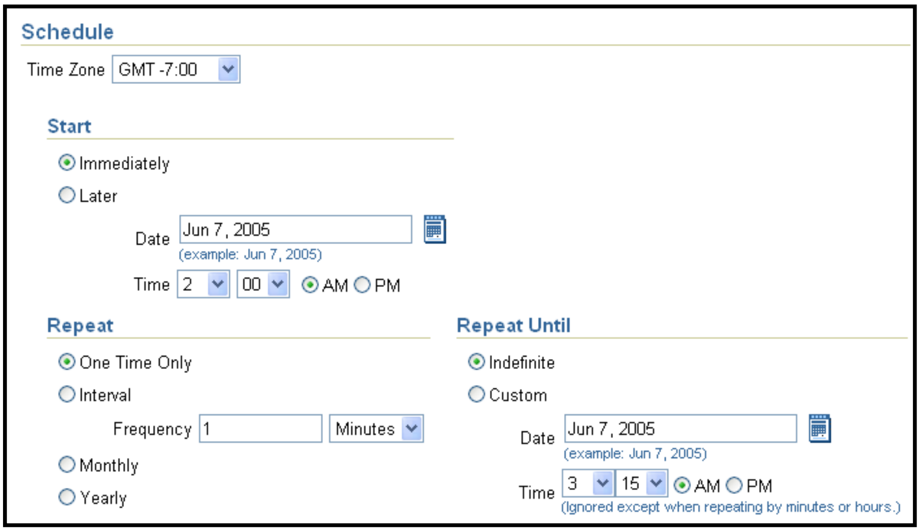


15-13

Scheduling Backups: Settings

Select whether the backup is to go to the disk or to the tape.

Scheduling Backups: Schedule

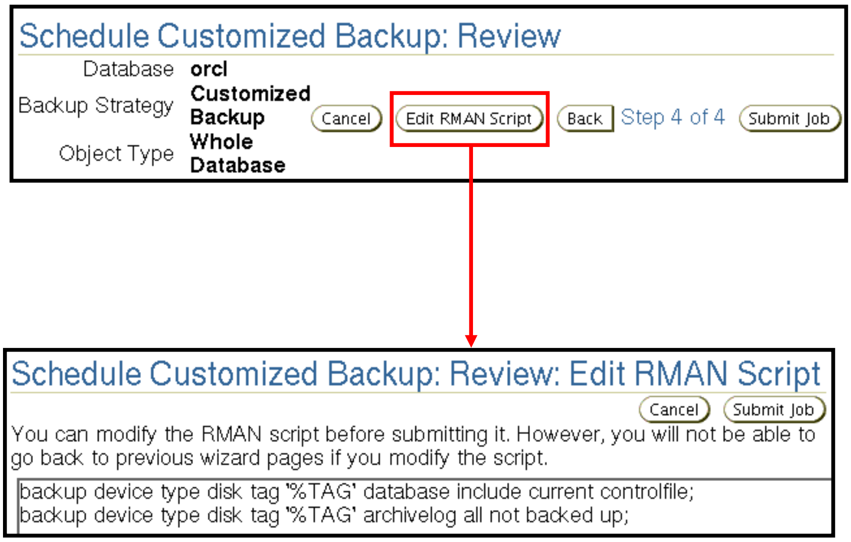


Scheduling Backups: Schedule

Choose how you want the backup to be scheduled—either as a one-time job or as an automated, recurring 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.

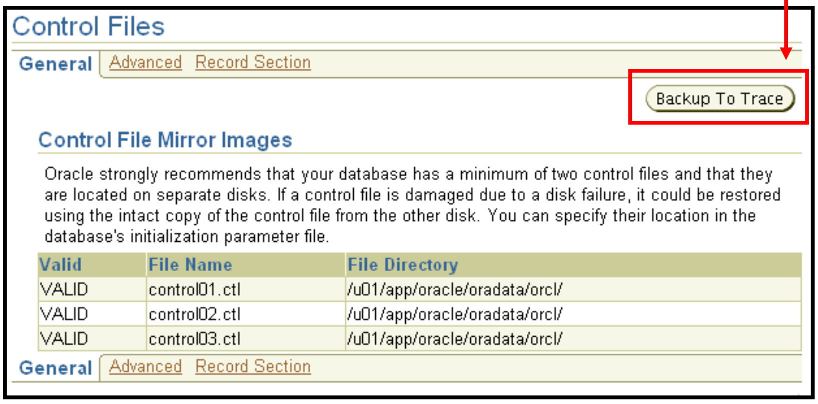
Scheduling Backups: Review

RMAN uses its own command syntax and scripting language. Click the Edit RMAN Script button to see the commands that the backup scheduler has generated on the basis of your inputs.

Using this page, you can customize the RMAN scripts if needed, or copy them for recording purposes.

Backing Up the Control File to a Trace File

Control files have an additional backup option.



Control file trace backups may be used to recover from loss of all control files.

Backing Up the Control File to a Trace File

Click Control Files in the Storage region 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 a trace file. 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. Therefore, the administrator should back up the control file to a trace file after each change to the physical structure of the database (adding tablespaces or data files, or adding additional redo log groups).

Trace copies of the control file can be created by using Enterprise Manager (as shown in the slide) by clicking Control Files on the Administration properties page, or with the following SQL command:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

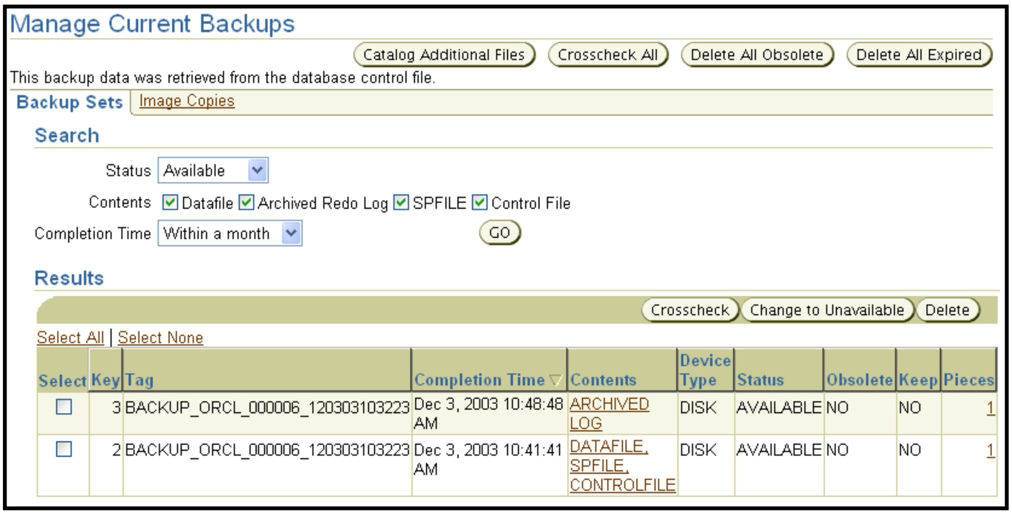
The trace backup is created in the location specified by the `USER_DUMP_DEST` initialization parameter with a file name such as *`sid_ora_pid.trc`*.

Backing Up the Control File to a Trace File (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_m
  f_1_1_%u_.arc';
-- ALTER DATABASE REGISTER LOGFILE
  '/oracle/flash_recovery_area/ORCL/archivelog/2003_12_05/o1_m
  f_1_1_%u_.arc';
-- Recovery is required if any of the data files 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;
```

Managing Backups



15-18

Managing Backups

Click Manage Current Backups on the Maintenance properties page to manage your existing backups. On this page, you can see when a backup was completed, where it was created (disk or tape), and whether it is still available.

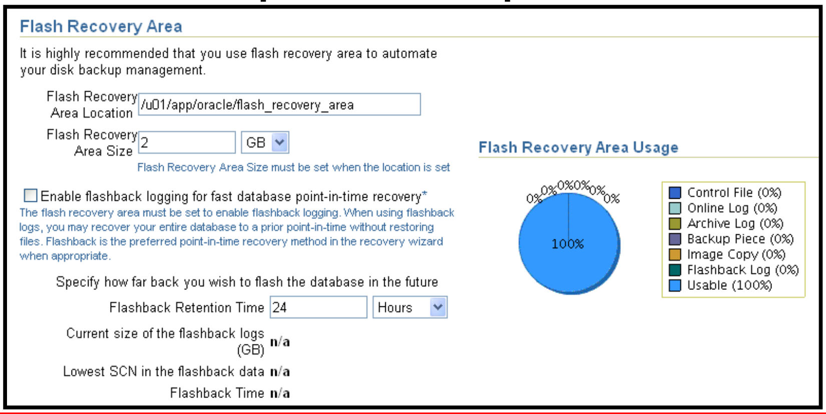
At the top of the Manage Current Backups page, you can see four buttons that enable you to work with existing backups:

- **Catalog Additional Files:** Although RMAN (working through Enterprise Manager) is the recommended way to create backups, it is possible that you may have image copies or backup sets created by some other means or in some other environment such that RMAN is not aware of them. This task identifies those files and adds 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 without using RMAN, the catalog does not know whether the backup is missing until you perform a crosscheck between the catalog and what is really there.
- **Delete All Obsolete:** This deletes backups older than the retention policy.
- **Delete All Expired:** This deletes the catalog listing for any backups that are not found when the crosscheck is performed.

Flash Recovery Area

Monitor the flash recovery area to:

- **Configure flashback logging**
- **Size the recovery area**
- **View current space consumption**



Flash Recovery Area

The flash recovery area is a space set aside on the disk to contain archived logs, backups, flashback logs, mirrored control files, and mirrored redo 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), then it is important to monitor this space to ensure that it does not reach its capacity. If the instance is unable to create an archived log because of lack of space, then it pauses until the administrator corrects the situation.

Clicking Recovery Settings on the Maintenance property page takes you to the Flash Recovery Area settings. On this page, you can:

- Specify the location of the flash recovery area
- Specify the size of the flash recovery area (Oracle recommends that this be at least twice the size of the database so that it can hold one backup and several archived logs.)
- Verify how much of the flash recovery area has been consumed
- Configure flashback database. Flashback database is discussed in the lesson titled “Performing Flashback.”

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

15-20

Practice Overview: Creating Database Backups

This practice covers the following topics:

- 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

16

Performing Database Recovery

Objectives

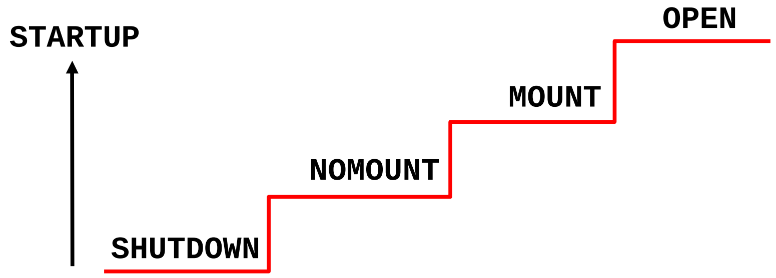
After completing this lesson, you should be able to recover from the loss of a:

- Control file
- Redo log file
- Data file

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**



Opening a Database

As a database moves from the shutdown stage to being fully open, it performs internal consistency checks. The stages are:

- `NOMOUNT`: In order for an instance to reach the `NOMOUNT` (also known as `STARTED`) status, the instance must read the initialization parameter file. No database files are checked while the instance enters the `NOMOUNT` state.
- `MOUNT`: As the instance moves to the `MOUNT` status, it checks whether all control files listed in the initialization parameter file are present and synchronized. If even one control file is missing or corrupt, the instance returns an error (noting the missing control file) to the administrator and remains in the `NOMOUNT` state.
- `OPEN`: When the instance moves from the `MOUNT` state to the `OPEN` state, it:
  - Checks whether 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 in 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 that need recovery, the administrator can check the `v$recover_file` dynamic performance view to get a complete list of the files that need 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
```

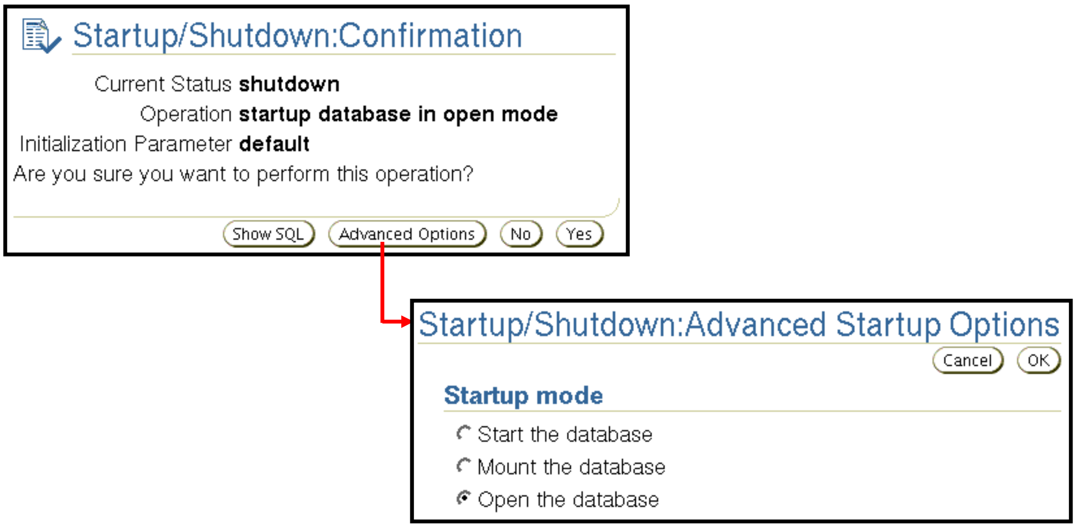
- Verifies that all data files that are not offline or read-only are synchronized with the control file. If necessary, instance recovery is automatically performed. However, if a file is out of synchronization to the extent that it cannot be recovered by using the online redo log groups, then the administrator 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 in 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 that need attention. Files that are present and which need media recovery are listed, but no error message is displayed .

Changing Instance Status

Use Database Control to alter the instance's status.



ORACLE

16-5

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 the instance in another mode. The Advanced Startup Options properties page enables 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

After the instance is open, it fails in the case of the loss of:

- **Any control file**
- **A data file belonging to the system or undo tablespaces**
- **An entire redo log group. As long as at least one member of the group is available, the instance remains open.**

16-6

Keeping a Database Open

After an instance is open, media failure that causes the loss of a control file, loss of an entire redo log group, or loss of a data file belonging to the `SYSTEM` or `UNDO` tablespaces causes the instance to fail.

In many cases, the failed instance does not completely shut down, but is unable to continue to perform work. Recovering from these types of media failures must be done with the database down, and so the administrator must use the `SHUTDOWN ABORT` command before beginning recovery efforts.

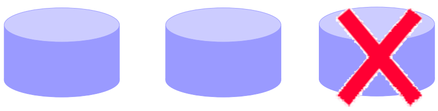
The loss of data files belonging to other tablespaces does not cause instance failure, and the database can be recovered while open, with work continuing in other tablespaces.

These errors can be detected by inspecting the alert log file.

Loss of a Control File

If a control file is lost or corrupted, the instance normally aborts, at which time you must perform the following steps:

1. **Shut down the instance, if it is still open.**
2. **Restore the missing control file by copying an existing control file.**
3. **Start the instance.**



Control files

16-7

Loss of a Control File

Recovering from the loss of a control file (as long as at least one control file remains) can be accomplished by performing the following steps:

1. If the instance has not already failed, shut it down by using `SHUTDOWN ABORT`.
2. Copy one of the remaining control files to the missing file's location. If the media failure is due to the 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 recommends having *at least* two control files at all times.
3. Start the instance.

Recovering from the 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 redo log file group is lost, as long as the group still has at least one member, then:

- **The normal operation of the instance is not affected.**
- **You receive a message in the alert log notifying you that a member cannot be found.**
- **You can restore the missing log file by dropping the lost redo log member and adding a new member.**
- **If the group with the missing log file has been archived, you can clear the log group to re-create the missing file.**

ORACLE

Loss of a Redo Log File

Recovering from the loss of a single redo log group member should not affect the running instance. To perform this recovery, perform the following steps:

1. Determine whether there is a missing log file by examining the alert log.
2. Restore the missing file by first dropping the lost redo log member:
   ```
   SQL> ALTER DATABASE DROP LOGFILE MEMBER 'redo01a.log';
   ```
   Then add a new member to replace the lost redo log member:
   ```
   SQL> ALTER DATABASE ADD LOGFILE MEMBER 'redo01a.log'
   TO GROUP 2;
   ```
   Note: If you are using OMF for your redo log files and you use the above syntax to add a new redo log member to an existing group, that new redo log member file will not be an OMF file. If you want to ensure that the new redo log member is an OMF file, the easiest recovery option would be to create a new redo log group and then drop the redo log group that had the missing redo log member.
3. If the media failure is due to the loss of a disk drive or controller, rename the missing file.

Loss of a Redo Log File (continued)

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 following command:

```
SQL> ALTER DATABASE CLEAR LOGFILE GROUP #;
```

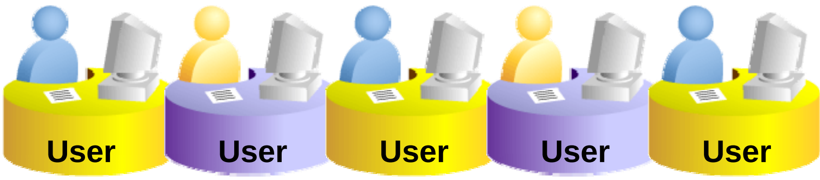
Note: Database Control does not allow you to clear a log group that has not been archived. Doing so breaks the chain of redo information. If you must clear an unarchived log group, you should *immediately* take a full backup of the whole database. Failure to do so may result in a loss of data if another failure occurs. To clear an unarchived log group, use the following 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, perform the following tasks:

1. **Shut down the instance if it is not already down.**
2. **Restore the entire database, including all data and control files, from the backup.**
3. **Open the database.**
4. **Have users reenter all changes made since the last backup.**



16-10

Loss of a Data File in NOARCHIVELOG Mode

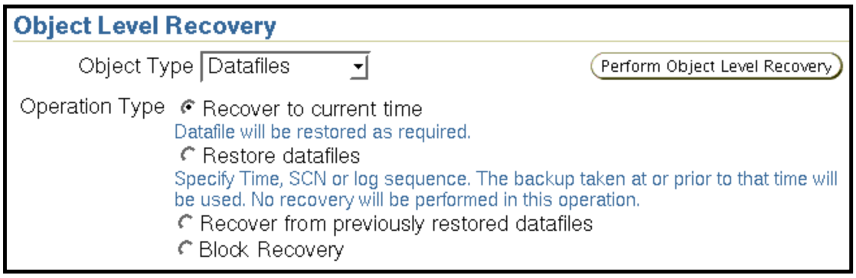
The loss of *any* data file from a database in NOARCHIVELOG mode requires complete restoration of the database, including control files and all data files.

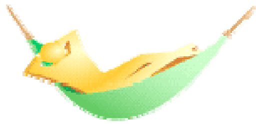
With the database in NOARCHIVELOG mode, recovery is possible only up to the time of the last backup. So users must reenter all changes made since that backup. For this type of recovery, perform the following tasks:

1. Shut down the instance if it is not already down.
2. Click Perform Recovery on 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

ORACLE

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. To restore and recover the missing data file, perform the following steps:

1. Click Perform Recovery on the Maintenance properties page.
2. Select "Datafiles" as the recovery type, and select "Restore to current time."
3. Add all data files that need recovery.
4. Determine whether you want to restore the files to the default location or (if a disk or controller is missing) 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 to 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

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). To perform this recovery, perform the following steps:

1. If the instance is not already shut down, shut it down.
2. Mount the database.
3. Click Perform Recovery on the Maintenance properties page.
4. Select “Datafiles” as the recovery type, and select “Restore to current time.”
5. Add all data files that need recovery.
6. Determine whether you want to restore the files to the default location or (if a disk or controller is missing) 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 is up to the time of the last commit.

Summary

In this lesson, you should have learned how to recover from the loss of a:

- Control file
- Redo log file
- Data file

Practice Overview: Performing Database Recovery

This practice covers recovering from the loss of a:

- **Control file**
- **Redo log file**
- **Noncritical data file**
- **System-critical data file**

16-14

17

Performing Flashback

Objectives

After completing this lesson, you should be able to:

- Describe Flashback Database
- Restore the table content to a specific point in the past with Flashback Table
- Recover from a dropped table
- View the contents of the database as of any single point in time with Flashback Query
- See versions of a row over time with Flashback Versions Query
- View transaction history or a row with Flashback Transaction Query

ORACLE

Flashback Technology: Benefits

> **Overview**
Database
Table
Drop
Query
Versions
Transaction

- **The Flashback technology is a revolutionary advance in recovery.**
- **Traditional recovery techniques are slow.**
  - **The entire database or a file (not just the incorrect data) has to be restored.**
  - **Every change in the database log must be examined.**
- **Flashback is *fast*.**
  - **Changes are indexed by row and by transaction.**
  - **Only the changed data is restored.**
- **Flashback commands are *easy*.**
  - **No complex multiple-step procedures are involved.**

ORACLE

17-3

Flashback Technology: Benefits

The Oracle Database 10*g* architecture leverages the unique technological advances in the area of database recovery from the loss of data due to human errors. The Flashback technology provides a set of new features to view and rewind data back and forth in time.

The Flashback technology revolutionizes recovery by simply operating on the changed data. The time it takes to recover from the error is equal to the amount of time it takes to make the error. When applicable, the Flashback technology provides significant benefits over media recovery in terms of ease of use, availability, and restoration time.

When to Use the Flashback Technology

Object Level	Scenario Examples	Flashback Technology	Uses	Affects Data
Database	Truncate table; Undesired multitable changes made	Database	Flashback logs	True
Table	Drop table	Drop	Recycle bin	True
	Update with the wrong `WHERE` clause	Table	Undo data	True
	Compare current data with data from the past	Query	Undo data	False
	Compare versions of a row	Version	Undo data	False
Tx	Investigate several historical states of data	Transaction	Undo data	False

When to Use the Flashback Technology

The Flashback technology must be used when a logical corruption occurs in the Oracle database, and you need to recover data quickly and easily. As with human errors, it is difficult to identify the objects and rows that are affected by an erroneous transaction. With the Flashback technology, you can diagnose how errors are introduced into the database, and then you can repair the damage. You can view the transactions that have contributed to specific row modifications, view the entire set of versions of a given row during some time period, or just view data as it appeared at a specific time in the past. The table in the slide shows typical uses of the Flashback technology.

Flashback Database uses the flashback logs to perform flashback. Flashback Drop uses the recycle bin. All others use undo data.

Not all flashback features modify the database. Some are simply methods to query other versions of data. Those are tools for you to use to investigate a problem and aid in recovery. The results of those flashback queries can help you perform one of these two things:

- Determine which type of database-modifying flashback operation to perform to fix the problem
- Feed the result set of these queries into an `INSERT`, `UPDATE`, or `DELETE` statement that enables you to easily repair the erroneous data.

Flashing Back Any Error

- Flashback Database brings the database to an earlier point in time by undoing all changes made since that time.
- Flashback Table recovers a table to a point in time in the past without having to restore from a backup.
- Flashback Drop restores accidentally dropped tables.

17-5

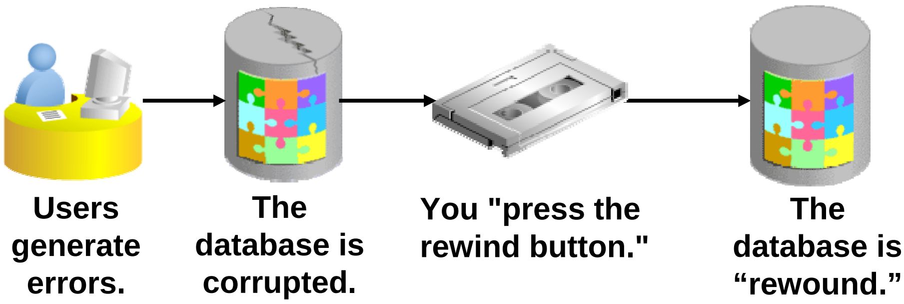
Flashing Back Any Error

Oracle Database 10*g* introduces expanded database flashback capabilities. If a major error occurs that makes unisolated changes, such as a batch job being run twice in succession, then you can request a flashback operation that quickly recovers the entire database to a previous point in time. This eliminates the need to restore backups and perform a point-in-time recovery. In addition to flashback operations at the database level, it is also possible to flash back a single table or recover a table dropped in error.

Flashback Database: Overview

Overview
> **Database**
Table
Drop
Query
Versions
Transaction

The Flashback Database operation:
- **Works like a rewind button for the database**
- **Can be used in cases of logical data corruptions made by users**

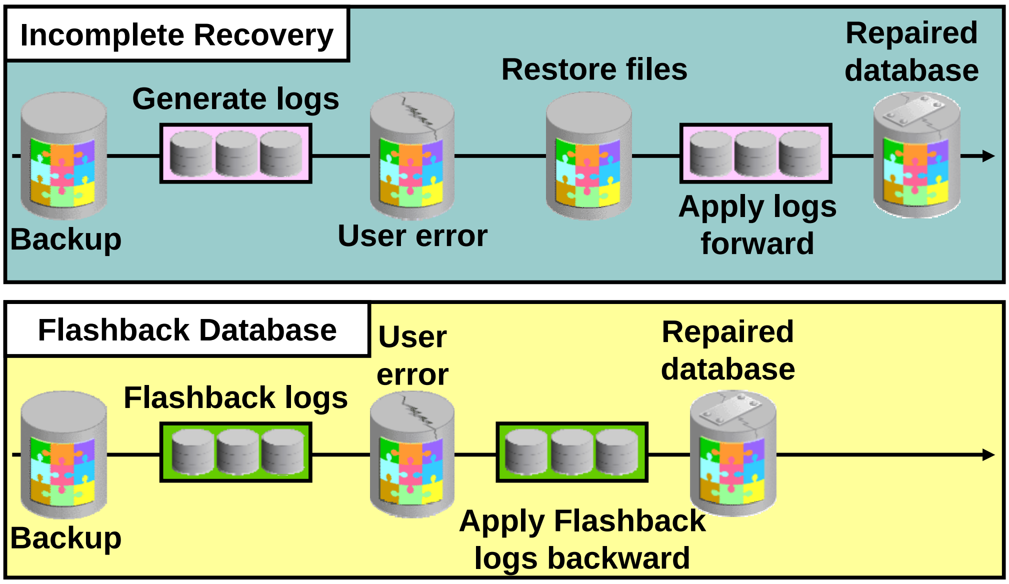


Flashback Database: Overview

With Flashback Database, you can quickly bring your database to an earlier point in time by undoing all the changes that have taken place since that time. This operation is fast because you do not need to restore backups. You can use this feature to undo changes that have resulted in logical data corruptions.

If you have experienced a loss of media or physical corruption in your database, then you must use traditional recovery methods.

Flashback Database: Reducing Restore Time



17-7

Flashback Database: Reducing Restore Time

Flashback Database is faster than the traditional point-in-time recovery that uses restored files and redo log files. As a database grows in size, the length of time required to restore all the data files to perform a traditional point-in-time recovery becomes prohibitive. With Flashback Database, the time to recover a database is now proportional to the number of changes that need to be backed out (and not to the size of the database) because you do not have to restore data files.

Flashback Database is implemented by using a type of log file called Flashback Database logs. The Oracle database periodically logs "before images" of data blocks in the Flashback Database logs. Block images can be reused to quickly back out the data file changes to any time at which flashback logs are captured just before the desired target time. Then, changes from the redo log files are applied to fill in the gap. The Flashback Database logs are automatically created and managed in the flash recovery area.

Flashback Database: Considerations

- When the Flashback Database operation completes, the database must be opened by using one of these methods:
  - In read-only mode to verify that the correct target time or SCN has been used
  - With the `RESETLOGS` parameter to allow for updates
- The opposite of flash back is recover.

17-8

Flashback Database: Considerations

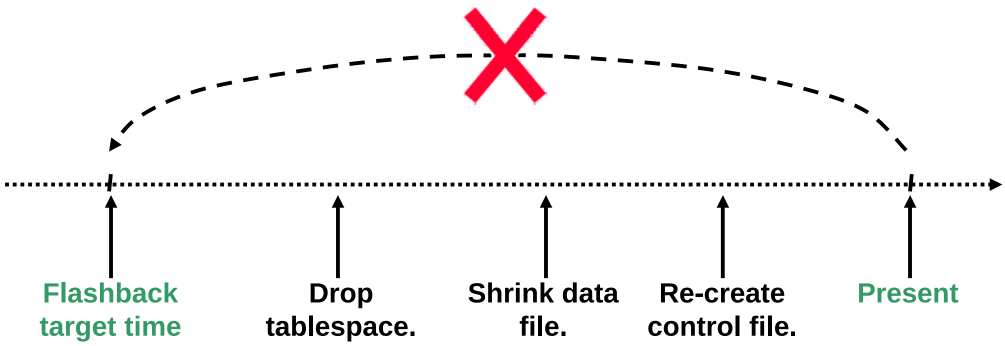
In situations where you cannot use the Flashback Database feature, you must use an incomplete recovery operation to return the database to a specific time. After the Flashback Database operation is complete, you can open the database in read-only mode to verify that the correct target time or system change number (SCN) has been used. If not, you can flash back the database again or perform a recovery to roll forward the database. So, to undo a Flashback Database operation, you must recover the database forward.

Note: The flashback retention target is not an absolute guarantee that flashback is available. If space is needed for required files in the flash recovery area, then flashback logs may be deleted automatically.

Flashback Database: Limitations

You cannot use Flashback Database in the following situations:

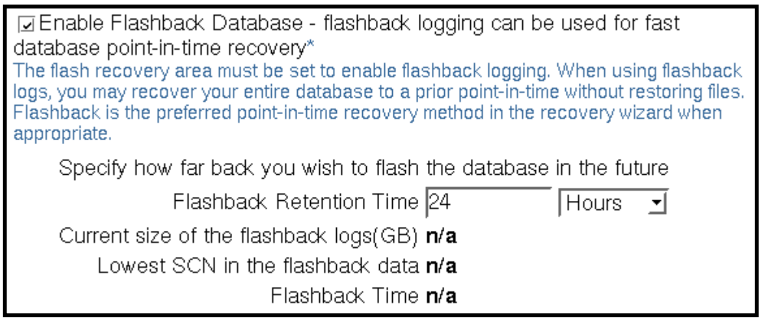
- **The control file has been restored or re-created.**
- **A tablespace has been dropped.**
- **A data file has been shrunk.**



Flashback Database: Limitations

You cannot use Flashback Database to recover a data file that has been dropped since the Flashback target time. The dropped data file is added to the control file and marked offline, but it is not flashed back. Flashback Database cannot flash back a data file if it has been shrunk since the Flashback target time. Any such data files must be taken offline before performing the flashback operation.

Enabling Flashback Database



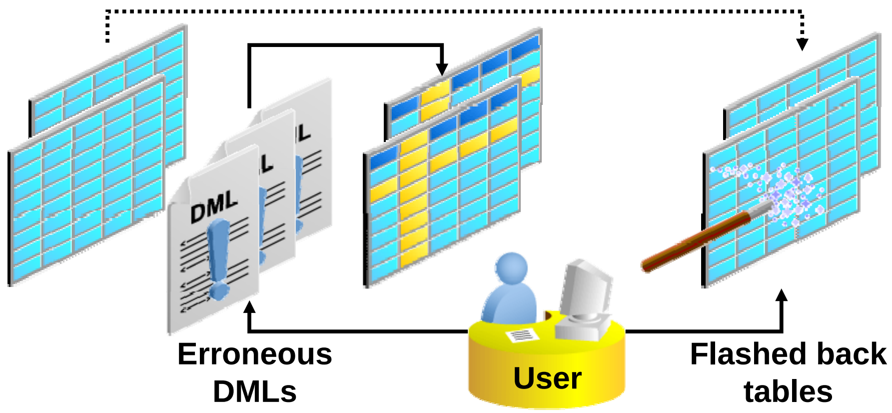
17-10

Enabling Flashback Database

Flashback Database can be enabled in Enterprise Manager, using the Recovery Settings link on the Maintenance tabbed page. Turn on the checkmark in the Flash Recovery region of the page, and specify the retention time, which is how far back in the past you want to be able to flash back the database to.

Flashback Table: Overview

- Flashback Table recovers tables to a specific point in time.
- Flashback Table is an in-place operation.
- The database stays online.



Flashback Table: Overview

Using Flashback Table, you can recover a set of tables to a specific point in time without having to perform traditional point-in-time recovery operations.

A Flashback Table operation is done in-place, while the database is online, by rolling back only the changes that are made to the given tables and their dependent objects.

A Flashback Table statement is executed as a single transaction. All tables must be flashed back successfully, or the entire transaction is rolled back.

Note: You can use Flashback Versions Query and Flashback Transaction Query to determine the appropriate flashback time.

Flashback Table

- Using Flashback Table, you can recover a table or tables to a specific point in time without restoring a backup.
- Data is retrieved from the undo tablespace to perform a Flashback Table operation.
- The `FLASHBACK` object privilege and the `FLASHBACK ANY TABLE` system privilege can be granted to allow a non-owner of a table to flashback that table.
- Row movement must be enabled on the table that you are performing the flashback operation on.

Flashback Table

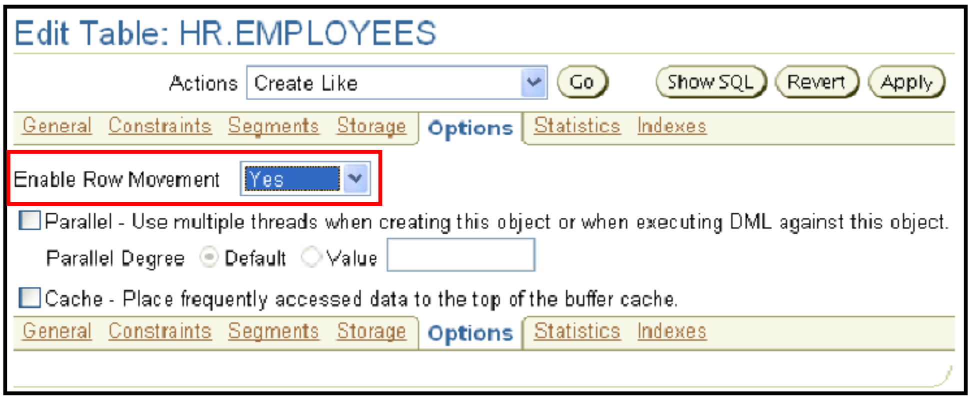
With Flashback Table, you can recover a table or tables to a specific point in time without restoring a backup. When you use this feature, the data in tables and their associated objects (indexes, constraints, triggers, and so on) is restored. The data used to satisfy a Flashback Table request is retrieved from the undo tablespace.

You can use Flashback Versions Query and Flashback Transaction Query to determine the appropriate flashback time. For more information about using these features, refer to *Oracle Database Concepts*.

Flashback Table provides a way for users to easily and quickly recover from accidental modifications without a database administrator's (DBA's) involvement. You may grant the `FLASHBACK` object privilege to other users for a specific table to allow them to flashback that table. There is also the `FLASHBACK ANY TABLE` system privilege that allows a user to flashback any table.

You can use Enterprise Manager to flash back a table. The wizard guides you through the process.

Enabling Row Movement on a Table




```
ALTER TABLE employees ENABLE ROW MOVEMENT;
```

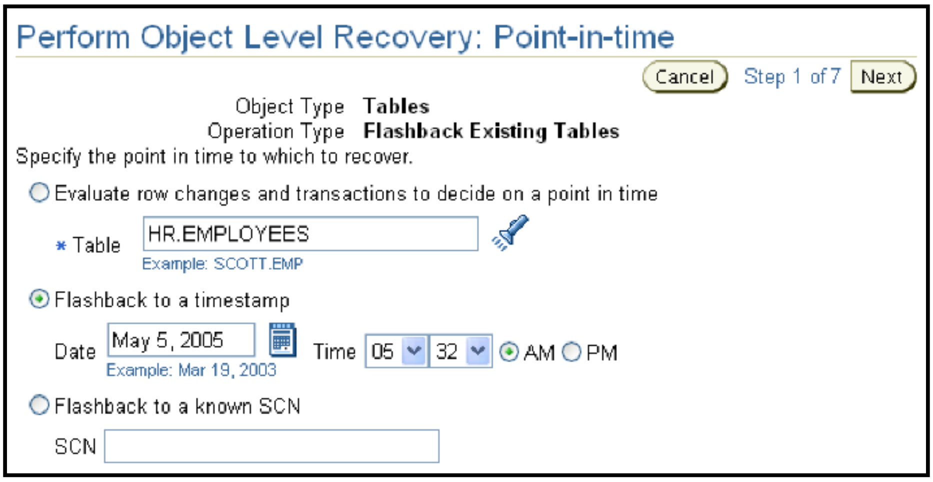

Enabling Row Movement on a Table

You must enable row movement on a table to be able to flash back the table. When you enable row movement, the Oracle server can move a row in the table. You can use Enterprise Manager to enable row movement.

Using Enterprise Manager, you can enable row movement on a table by performing the following steps:

1. Select Tables in the Schema region on the Administration property page. Enter the schema name to search for the table, and click Go.
2. Click the table name of the table for which you want to enable row movement. You are now on the View Table page.
3. Click Edit, which takes you to the Edit Table page.
4. Click the Options tab, where you can change the Enable Row Movement setting for the table.
5. Set Enable Row Movement to Yes, and click Apply.
   The update confirmation message is displayed.

Performing Flashback Table




```
FLASHBACK TABLE hr.employees TO TIMESTAMP
TO_TIMESTAMP('2005-05-05 05:32:00',
'YYYY-MM-DD HH24:MI:SS');
```

ORACLE

17-14

Performing Flashback Table

You can flash back a table through Enterprise Manager by performing the following steps:

1. Select Perform Recovery in the Backup/Recovery region on the Maintenance property page. The Perform Recovery page appears.
2. In the Object Level Recovery region, select Tables from the Object Type drop-down list.
3. Select Flashback Existing Tables as the Operation Type. Click Perform Object Level Recovery. The “Perform Object Level Recovery: Point-in-time” page is displayed.
4. Select “Flashback to a timestamp” or “Flashback to a known SCN” and then specify a time stamp or SCN to flash back to, and click Next.
5. Click Add Tables to add tables to the list for the flashback operation. Click Next.
6. The Dependency Options page appears if there are dependent tables. Select the desired option for dealing with dependent tables. Typically, you would choose “Cascade” to ensure a consistent flashback. Click Next.

Performing Flashback Table (continued)

7. The “Perform Object Level Recovery: Review” page appears. Review the information and click Submit. The Confirmation page appears.

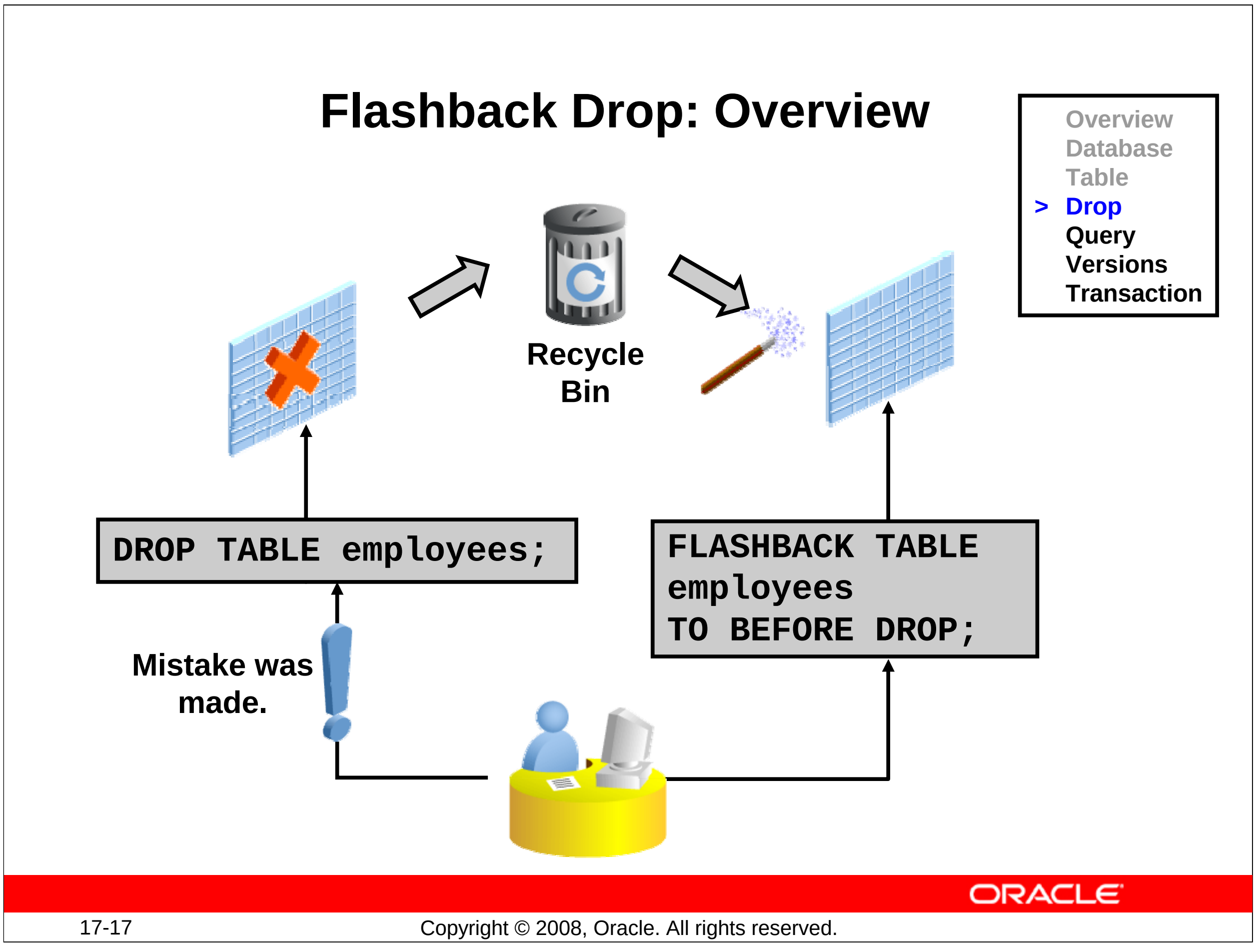
Note: You can also flash back tables from the Tables link in the Schema region of the Administration page.

Flashback Table: Considerations

- The `FLASHBACK TABLE` command executes as a single transaction, acquiring exclusive DML locks.
- Statistics are not flashed back.
- Current indexes and dependent objects are maintained.
- Flashback Table operations:
  - Cannot be performed on system tables
  - Cannot span DDL operations
  - Generate undo and redo data

Flashback Table: Considerations

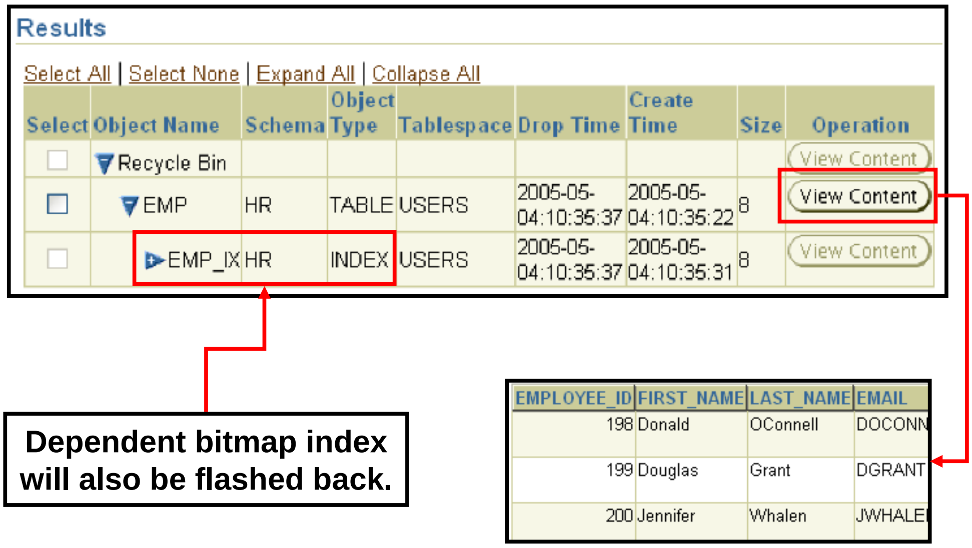
- The entire `FLASHBACK TABLE` statement is executed within a single transaction. All or none of the specified tables are flashed back.
- Flashback Table acquires exclusive data manipulation language (DML) locks on all the tables that are specified in the statement over the period of time when the operation is in progress.
- Statistics of impacted objects are not flashed back.
- All existing indexes are maintained. Dropped indexes are not re-created. Dependent on-commit materialized views are also maintained automatically.
- A `FLASHBACK TABLE` statement is written to the alert log file.
- Tables specified in the `FLASHBACK TABLE` statement are flashed back, provided that none of the table constraints are violated. If any constraints are violated during flashback execution, the operation is aborted and the tables are left in the same state as they were just before the `FLASHBACK TABLE` statement invocation.
- You cannot perform Flashback Table to a particular time that is older than the time of the execution of a data definition language (DDL) operation that altered the structure of or shrunk a table that would be involved in the flashback operation. This restriction does not apply to DDL statements that only change storage attributes of the tables.
- Flashback Table cannot be performed on system tables, remote tables, and fixed tables.



Flashback Drop: Overview

With the Flashback Drop feature, you can undo the effects of a `DROP TABLE` statement without resorting to the traditional point-in-time recovery. This is made possible because of the recycle bin, which can be queried via the `DBA_RECYCLEBIN` view.

Flashing Back Dropped Tables Through Enterprise Manager



ORACLE

17-18

Flashing Back Dropped Tables Through Enterprise Manager

To flash back dropped tables with the Database Control Console, select Perform Recovery in the Backup/Recovery region of the Maintenance page. Select Tables for the Object Type in the Type region, and choose Flashback Dropped Tables in the Operation Type region. After this is done, click Perform Object Level Recovery.

You should now see the “Perform Object Level Recovery: Dropped Objects Selection” page, where you can select dropped tables from the recycle bin. You can also query the content of dropped tables by clicking View Content. Select the tables that you want to recover, and click Next.

The “Perform Object Level Recovery: Rename” page appears, where you can rename the table if a table with the same name currently exists in the same schema. Click Next to continue. On the “Perform Object Level Recovery: Review” page, you can review the details of your operation as well as display the corresponding SQL statements. After you are ready, click Submit. You should now see the Confirmation page. Click OK to go back to the Maintenance page.

Note: You can also flash back dropped tables from the Tables link of the Schema region of the Administration page. On the Tables page, click the Recycle Bin button.

Flashback Drop: Considerations

- Flashback Drop does not work for tables that:
  - Reside in the `SYSTEM` tablespace
  - Use fine-grained auditing or Virtual Private Database
  - Reside in a dictionary-managed tablespace
  - Have been purged, either by manual purging or automatic purging under space pressure
- The following dependencies are not protected:
  - Bitmap-join indexes
  - Materialized view logs
  - Referential integrity constraints
  - Indexes dropped before tables

ORACLE

17-19

Flashback Drop: Considerations

Flashback Drop only works for tables in non-`SYSTEM`, locally managed tablespaces (LMT). However, dependent objects that reside in dictionary-managed tablespaces are flashed back as part of the flashback operation of the LMT-dwelling parent object.
Tables with fine-grained auditing (FGA) or Virtual Private Database (VPD) policies defined on them are not eligible for a Flashback Drop operation. Also, you cannot flash back a dropped table if it has been purged. It could have been purged manually with the `PURGE` statement or automatically as the result of a need for space for other objects in the tablespace.

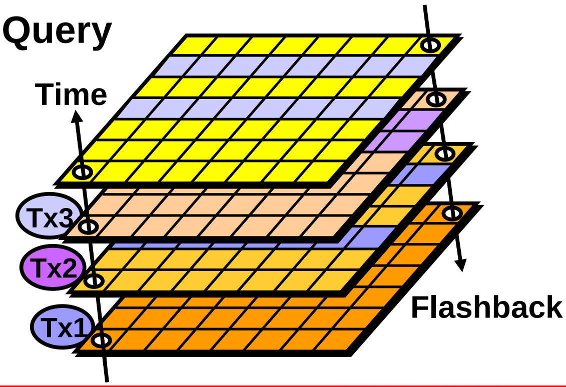
When you perform Flashback Drop on a table, all objects dependent on that table are also flashed back from the recycle bin. There are exceptions to this: Bitmap-join indexes, referential integrity constraints, and materialized view logs are not flashed back, even though their parent table is.

Note: If you drop an index before its associated table, recovery of the index is not supported when you flash back the dropped table.

Flashback Time Navigation

Overview
Database
Table
Drop
> Query
Versions
Transaction

- **Flashback Query**
  - Query all data at a specified point in time.
- **Flashback Versions Query**
  - See all versions of a row between two times.
  - See the transactions that changed the row.
- **Flashback Transaction Query**
  - See all changes made by a transaction.



ORACLE

17-20

Flashback Time Navigation

The Flashback technology offers the capability to query past versions of schema objects, query historical data, and perform change analysis. Every transaction logically generates a new version of the database. With the Flashback technology, you can navigate through these versions to find an error and its cause:

- **Flashback Query:** Query all data as it existed at a specific point in time.
- **Flashback Versions Query:** See all versions of rows between two times and the transactions that changed the row.
- **Flashback Transaction Query:** See all changes made by a transaction.

Flashback Query: Overview


```
SELECT employee_id, salary FROM employees
AS OF TIMESTAMP t1
WHERE employee_id = 200
```

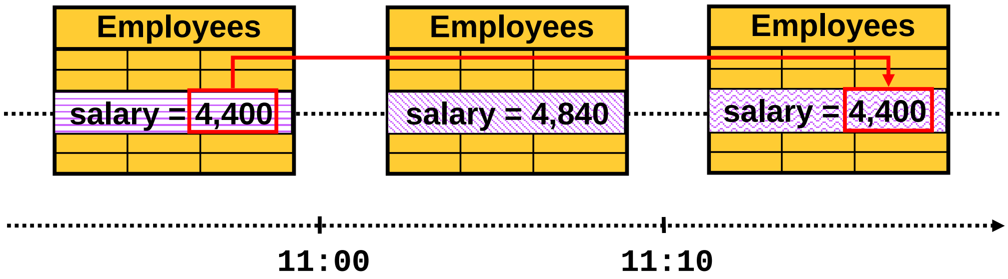
17-21

Flashback Query: Overview

With the Flashback Query feature, you can perform queries on the database as of a certain time. By using the `AS OF` clause of the `SELECT` statement, you can specify the time stamp for which to view the data. This is useful for analyzing a data discrepancy.

Note: The `AS OF` clause can be followed by either `TIMESTAMP` or `SCN`.

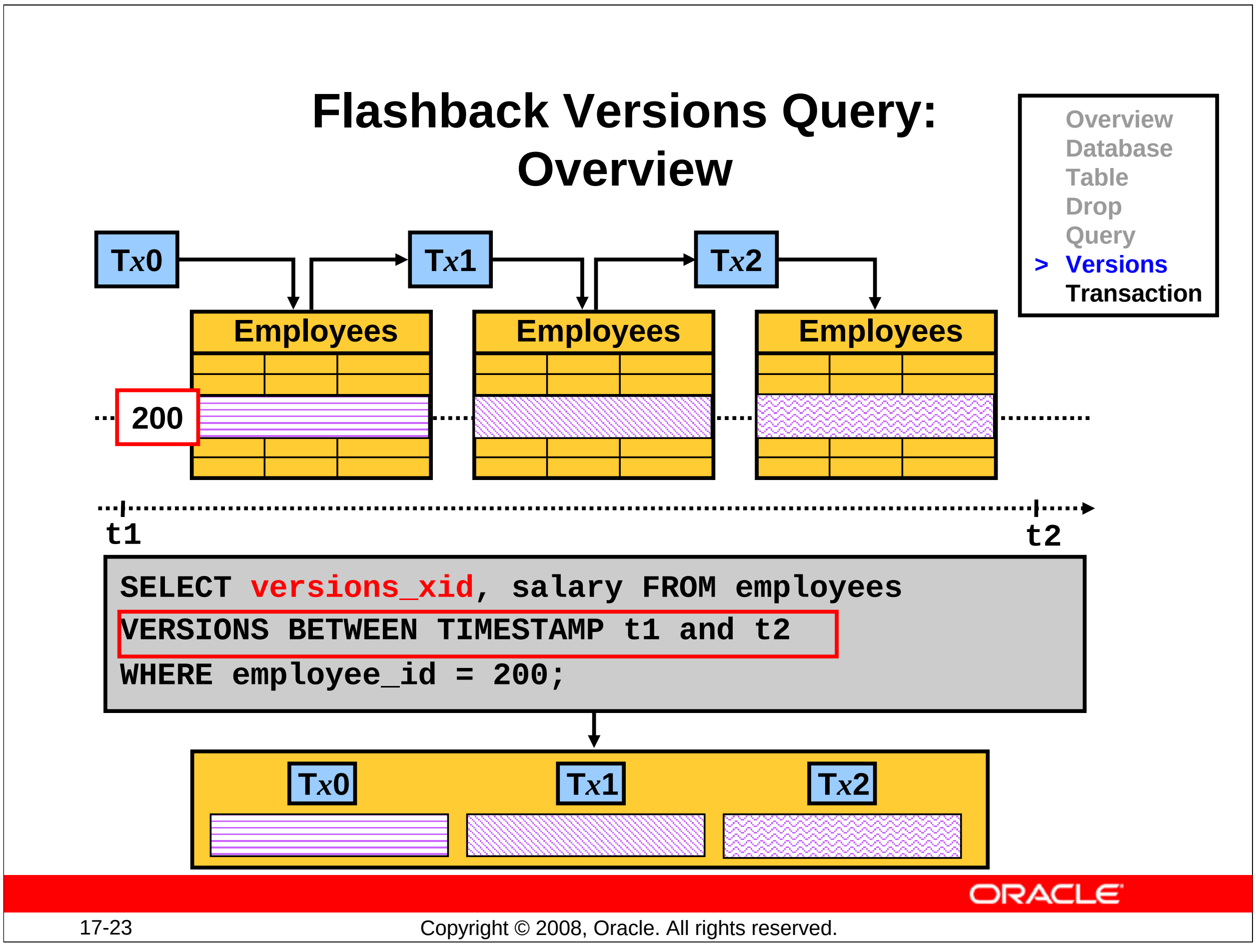
Flashback Query: Example




```
UPDATE employees SET salary =
      (SELECT salary FROM employees
      AS OF TIMESTAMP TO_TIMESTAMP
    ('2005-05-04 11:00:00', 'yyyy-mm-dd hh24:mi:ss')
      WHERE employee_id = 200)
WHERE employee_id = 200
```


Flashback Query: Example

If a raise has been erroneously given to a particular employee recently, you can update the salary again, assigning the salary provided by a subquery that returns the flashed back value.



Flashback Versions Query: Overview

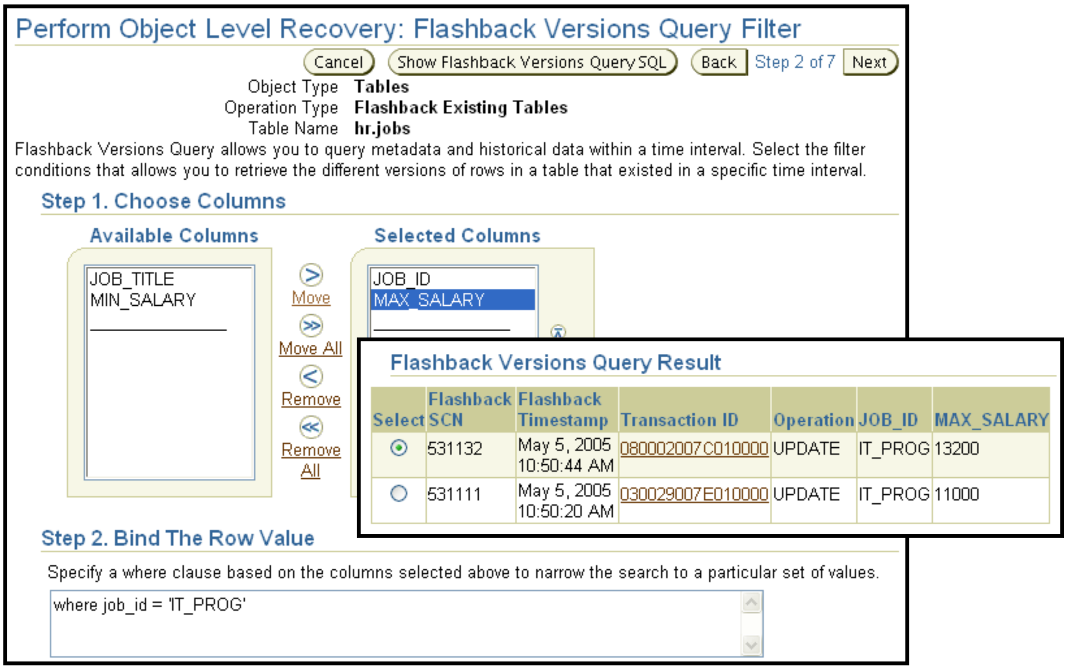
With the Flashback Query feature, you can perform queries on the database as of a certain time span or range of user-specified system change numbers (SCNs). The Flashback Versions Query feature enables you to use the `VERSIONS` clause to retrieve all the versions of the rows that exist between two points in time or two SCNs.

The rows returned by Flashback Versions Query represent a history of changes for the rows across transactions. Flashback Versions Query retrieves only committed occurrences of the rows. Uncommitted row versions within a transaction are not shown. The rows returned also include deleted and subsequently reinserted versions of the rows.

You can use Flashback Versions Query to retrieve row history. It provides you with a way to audit the rows of a table and retrieve information about the transactions that affected the rows. You can then use the returned transaction identifier either to perform transaction mining by using LogMiner or to perform a Flashback Transaction Query, which is covered later in this lesson.

Note: In the example, `VERSIONS_XID` is a pseudocolumn that returns the transaction identifier of the corresponding version of a row.

Flashback Versions Query Through Enterprise Manager



ORACLE

17-24

Flashback Versions Query Through Enterprise Manager

Flashback Versions Query can also be performed through Enterprise Manager. On the Maintenance page, select Perform Recovery.

On the Perform Recovery page, select Tables for the Object Type and select Flashback Existing Tables for the Operation Type. Click Perform Object Level Recovery. On the “Perform Object Level Recovery: Point-in-Time” page, select “Evaluate row changes and transactions to decide on a point in time,” and specify the name of the target table.

Select the columns that you want to view in the Available Columns box, and then enter a search clause in the Bind The Row Value box. Select “Show all row history,” and then click Next.

Flashback Versions Query: Considerations

- The `VERSIONS` clause cannot be used to query:
  - External tables
  - Temporary tables
  - Fixed tables
  - Views
- The `VERSIONS` clause cannot span DDL commands.
- Segment shrink operations are filtered out.

Flashback Versions Query: Considerations

The `VERSIONS` clause cannot be used to query the following special tables:

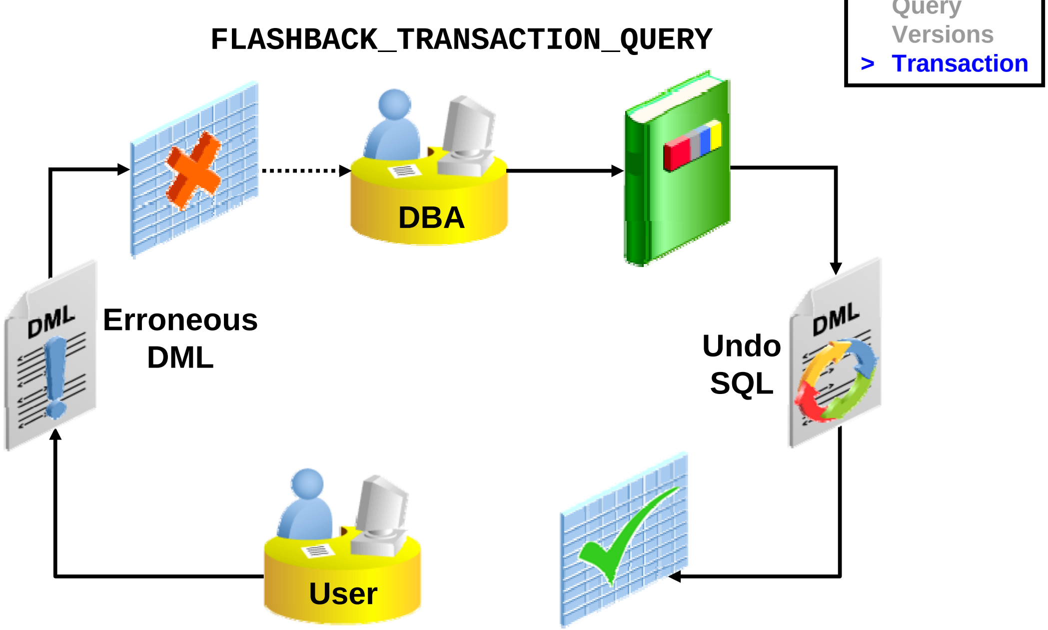
- External tables
- Temporary tables
- Fixed tables

You cannot use the `VERSIONS` clause to query a view. However, a view definition can use the `VERSIONS` clause.

The `VERSIONS` clause in a `SELECT` statement cannot produce versions of rows across the DDL statements that change the structure of the corresponding tables. This means that the query stops producing rows after it reaches a time in the past when the table structure was changed.

Certain maintenance operations, such as a segment shrink, may move table rows across blocks. In this case, the version query filters out such phantom versions because the row data remains the same.

Flashback Transaction Query: Overview



Flashback Transaction Query: Overview

Flashback Transaction Query is a diagnostic tool that you can use to view changes made to the database at the transaction level. This enables you to diagnose problems in your database and perform analysis and audits of transactions.

You can use the `FLASHBACK_TRANSACTION_QUERY` view to determine all the necessary SQL statements that can be used to undo the changes made either by a specific transaction or during a specific period of time.

Flashback Transaction Query Through Enterprise Manager

Flashback Versions Query Result

Select	Flashback SCN	Flashback Timestamp	Transaction ID	Operation	JOB_ID	MIN_SALARY
◉	489358	Aug 5, 2005 11:54:29 AM	090003002D010000	UPDATE	AD_PRES	30000
○	489347	Aug 5, 2005 11:54:11 AM	0A001C00CF000000	UPDATE	AD_PRES	25000
○	489318	Aug 5, 2005 11:53:17 AM	0800110002010000	UPDATE	AD_PRES	22000

Choose SCN: Transaction Details

OK

Transaction ID **0A001C00CF000000**
User **HR**
Commit SCN **489348**
Commit Time **Aug 5, 2005 12:00:00 AM**

Operation	Table Owner	Table Name	Undo SQL
UPDATE	HR	JOBS	update "HR"."JOBS" set "MIN_SALARY" = '22000' where ROWID = 'AAAMg1AAFAAAABIAAA';
UPDATE	HR	EMPLOYEES	update "HR"."EMPLOYEES" set "SALARY" = '4400' where ROWID = 'AAAMg3AAFAAAABUAAC';

ORACLE

Flashback Transaction Query Through Enterprise Manager

This feature is used in conjunction with the Flashback Versions Query feature with the help of the Perform Recovery Wizard. On the “Perform Object Level Recovery: Choose SCN” page, click the corresponding Transaction ID link in the Flashback Versions Query Result region.

In the example in the slide, a Flashback Versions Query is performed on the `JOBS` table to retrieve the three versions of the `JOBS` row for `JOB_ID` = '`AD_PRES`'. Then, one of the transaction IDs is clicked, showing all the changes that were part of that transaction. Notice that, besides the `JOBS` table update, there was also an update to the `EMPLOYEES` table in that transaction.

Flashback Transaction Query: Considerations

- DDLs are seen as dictionary updates.
- Dropped objects appear as object numbers.
- Dropped users appear as user identifiers.

17-28

Flashback Transaction Query: Considerations

- Within the database, DDL operations are nothing but a series of space management operations and changes to the data dictionary. Flashback Transaction Query on a transaction underlying a DDL displays the changes made to the data dictionary.
- When Flashback Transaction Query involves tables that have been dropped from the database, the table names are not reflected. Instead, object numbers are used.
- If the user who executed a transaction is dropped, Flashback Transaction Query of that transaction displays the corresponding user ID only, and not the username.

Note: When there is not enough undo data for a specific transaction, a row with a value of `UNKNOWN` in the `OPERATION` column of `FLASHBACK_TRANSACTION_QUERY` is returned.

Summary

In this lesson, you should have learned how to:

- Describe Flashback Database
- Restore the table content to a specific point in the past with Flashback Table
- Recover from a dropped table
- View the contents of the database as of any single point in time with Flashback Query
- See versions of a row over time with Flashback Versions Query
- View transaction history or a row with Flashback Transaction Query

17-29

Practice Overview: Using Flashback

This practice covers the following topics:

- Using Flashback to recover a dropped table
- Performing Flashback Versions Query

18

Moving Data

Objectives

Directory Obj.
SQL*Loader
Data Pump
- Export
- Import
External Table

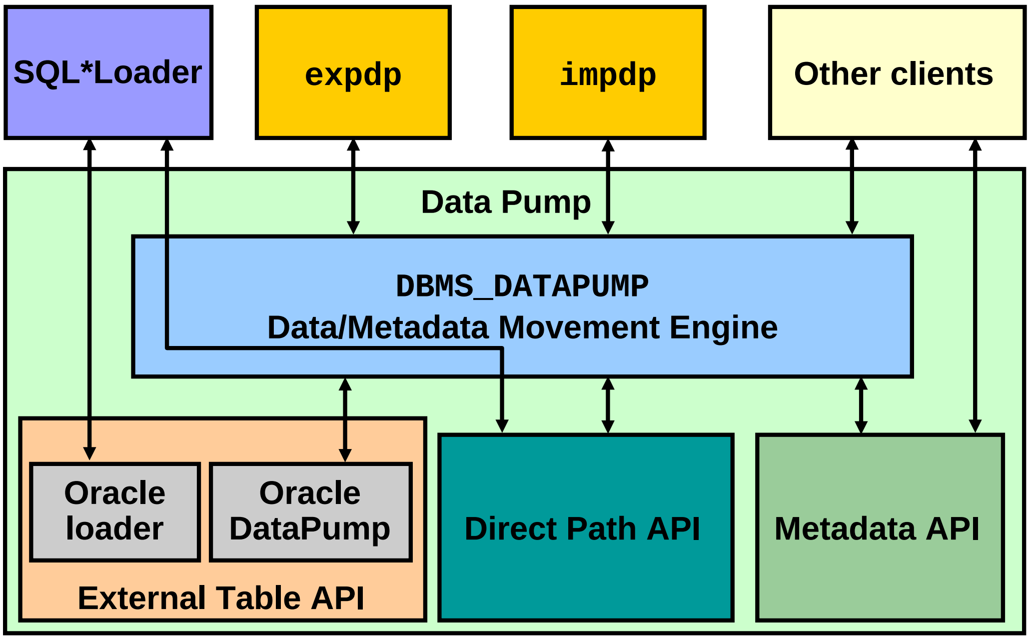
After completing this lesson, you should be able to do the following:

- Describe available ways for moving data
- Create and use directory objects
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Explain the general architecture of Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Use external tables to move data via platform-independent files

ORACLE

18-2

Moving Data: General Architecture



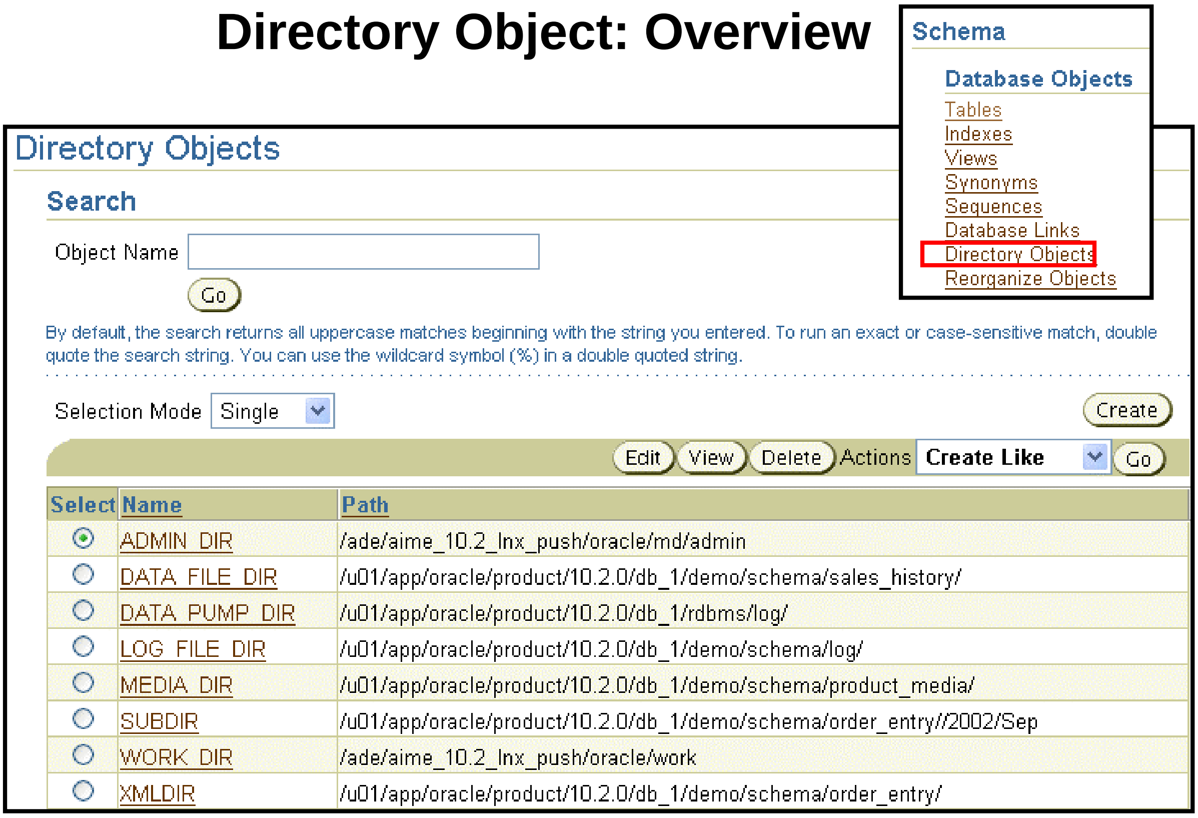
18-3

Moving Data: General Architecture

This is a block diagram of the major functional components:

- **DBMS_DATAPUMP:** This package embodies the API for high-speed export and import utilities for bulk data and metadata movement.
- **Direct Path API (DPAPI):** Oracle Database 10*g* supports a direct path API interface that minimizes data conversion and parsing at both unload and load time.
- **DBMS_METADATA:** This package is used by worker processes for all metadata unloading and loading. Database object definitions are stored using XML rather than SQL.
- **External Table API:** With the ORACLE_DATAPUMP and ORACLE_LOADER access drivers, you can store data in external tables (that is, in platform-independent files). The SELECT statement reads external tables as though they were stored in an Oracle database.
- **SQL*Loader:** The **SQL*Loader** client has been integrated with external tables, thereby providing automatic migration of loader control files to external table access parameters.
- **expdp** and **impdp:** The **expdp** and **impdp** clients are thin layers that make calls to the DBMS_DATAPUMP package to initiate and monitor Data Pump operations.
- **Other clients:** They are applications, such as Database Control, replication, transportable tablespaces, and user applications, that benefit from this infrastructure. SQL*Plus may also be used as a client of DBMS_DATAPUMP for simple status queries against ongoing operations.

Directory Object: Overview



18-4

Directory Object: Overview

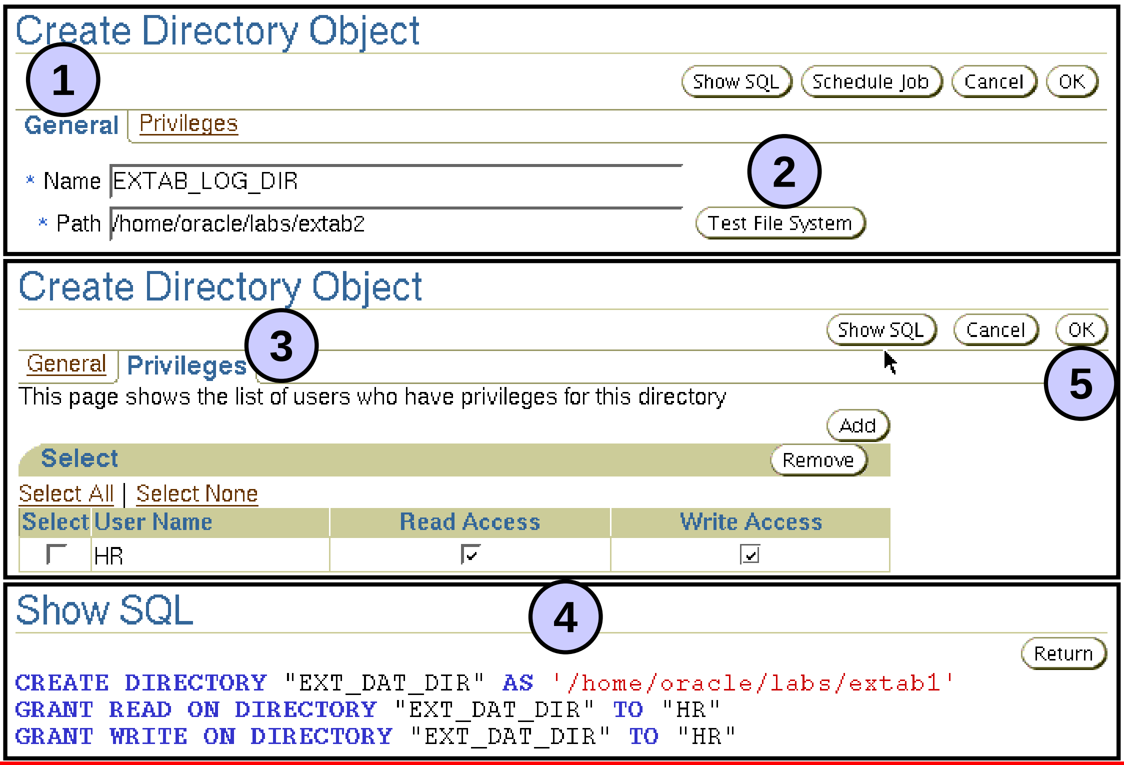
Directory objects are logical structures that represent a physical directory on the server's file system. They contain the location of a specific operating system directory. This directory object name can be used in Enterprise Manager, so you do not need to hard-code directory path specifications. Therefore, you get greater file management flexibility. Directory objects are owned by the SYS user. Directory names are unique across the database because all the directories are located in a single name space (that is, SYS).

Directory objects are required when you specify file locations for Data Pump because it accesses files on the server rather than on the client.

In Enterprise Manager, select Administration > Directory Objects.

To edit or delete a directory object, select the directory object and click the appropriate button.

Creating Directory Objects

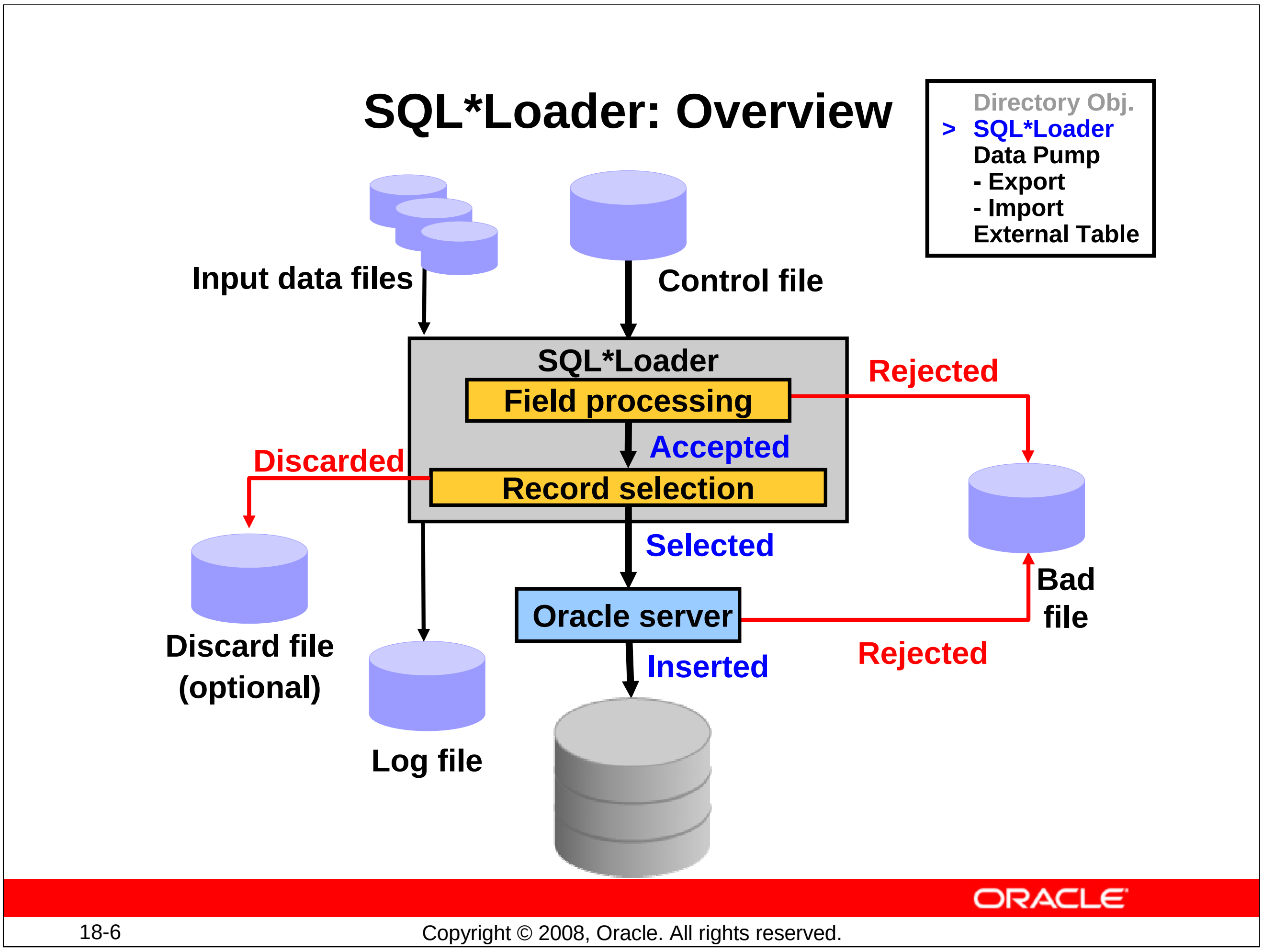


ORACLE

18-5

Creating Directory Objects

1. On the Directory Objects page, click the Create button.
2. Enter the name of the directory object and the OS path to which it maps. OS directories should be created before they are used. You can test this by clicking the “Test File System” button. For the test, provide the host login credentials (that is, the OS user who has privileges on this OS directory).
3. Permissions for the directory objects are not the same as the OS permissions on the physical directory on the server file system. You can manage user privileges on individual directory objects. This increases the level of security and gives you granular control over these objects. On the Privileges tabbed page, click Add to select the user to which you give read or write privileges or both.
4. Click Show SQL to view the underlying statements.
5. Click OK to create the object.



SQL*Loader: Overview

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 that are 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) that are 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 that is written in a language that SQL*Loader understands. The control file indicates to SQL*Loader where to find the data, how to parse and interpret the data, where to insert the data, and so on. Although not precisely defined, a control file can be said to have three sections.

- The first section contains sessionwide information, for example:
  - Global options, such as the input data file name, and records to be skipped.
  - `INFILE` clauses to specify where the input data is located
  - Data to be loaded

SQL*Loader: Overview (continued)

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

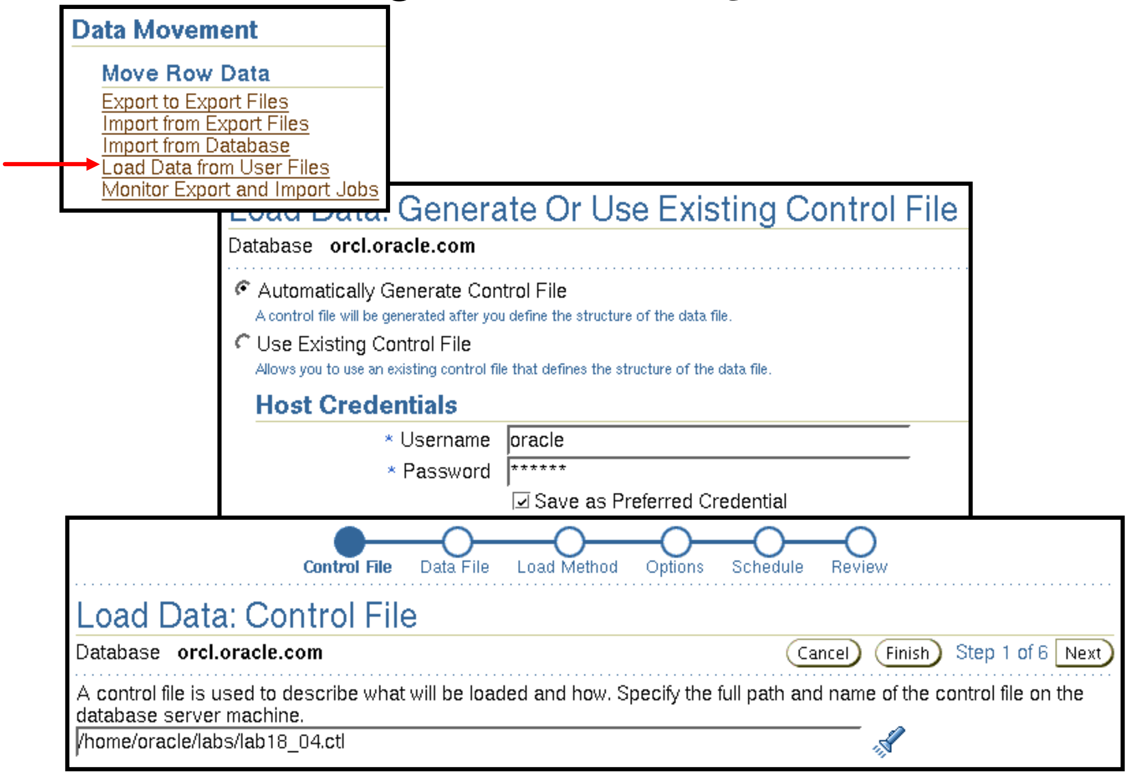
Log file: When SQL*Loader begins execution, it creates a log file. If it cannot create a log file, execution terminates. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load.

Bad file: The bad file contains records that are 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 are filtered out of the load because they do not match any record-selection criteria specified in the control file.

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

Loading Data with SQL*Loader



Loading Data with SQL*Loader

Use the Load Data from User Files Wizard to load data from a flat file into an Oracle database. To display the wizard, select Enterprise Manager Maintenance > Data Movement > Move Row Data > Load Data from User Files. The wizard guides you through the required steps.

SQL*Loader Control File

The SQL*Loader control file instructs SQL*Loader about:

- **Location of the data to be loaded**
- **The data format**
- **Configuration details:**
  - **Memory management**
  - **Record rejection**
  - **Interrupted load handling details**
- **Data manipulation details**

18-9

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 is being configured (including memory management, selection and rejection criteria, interrupted load handling, and so on) as it loads the data
- How SQL*Loader manipulates the data being loaded

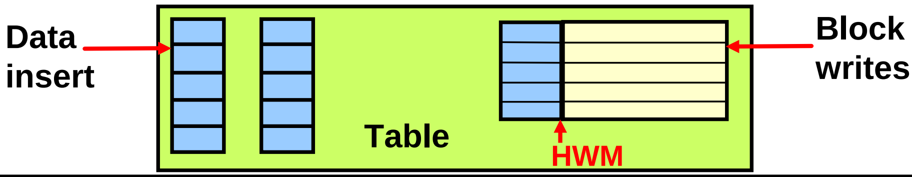
SQL*Loader Control File (continued)

```
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"
   )
```

The explanation of a sample control file by line numbers is as follows:

1. Comments can appear anywhere in the command section of the file, but they must 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 indicates to SQL*Loader that this is the beginning of a new data load. If you are continuing a load that has been interrupted in progress, use the `CONTINUE LOAD DATA` statement.
3. The `INFILE` keyword specifies the name of a data file containing data that you want to load.
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 that you can use when loading data into a table that is not empty. To load data into a table that is empty, 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 loads the data. In this example, SQL*Loader loads the record only 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 prompts 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.

Loading Methods



Conventional Load	Direct Path Load
Uses `COMMIT`	Uses data saves (faster operation)
Always generates redo entries	Generates redo only under specific conditions
Enforces all constraints	Enforces only `PRIMARY KEY`, `UNIQUE`, and `NOT NULL`
Fires `INSERT` triggers	Does not fire `INSERT` triggers
Can load into clustered tables	Does not load into clusters
Allows other users to modify tables during load operation	Prevents other users from making changes to tables during load operation

Comparing Direct and Conventional Path Loads

Method of Saving Data

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

Direct path loads use data saves to write blocks of data to Oracle data files. This is why the direct path loads are faster than the conventional ones. The following features differentiate a data save from `COMMIT`:

- During a data save, only full database blocks are written to the database.
- The blocks are written after the high-water mark (HWM) of the table.
- After a data save, the high-water mark (HWM) 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.

Note: Direct path and parallel direct path loads are so similar regarding DML activities that they are not separated in this comparison.

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 they can be enabled at the end of the load by using the appropriate commands in the control file. The `FOREIGN KEY` constraints are disabled because they reference other rows or tables, and the `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, then use conventional loads.
- `PRIMARY KEY` and `UNIQUE` constraints are checked during and at the end of the load, and they may be disabled if they are violated.

Firing the `INSERT` Triggers

The `WHILE INSERT` triggers are fired during conventional loads; they are disabled before a direct path load and reenabled at the end of the load. 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 the `INSERT` triggers.

Loading into Clustered Tables

Direct Loads cannot be used to load rows into clustered tables. Clustered tables can be loaded with 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.

Data Pump: Overview

As a server-based facility for high-speed data and metadata movement, Data Pump:

- Is callable via `DBMS_DATAPUMP`
- Provides the following tools:
  - `expdp`
  - `impdp`
  - Web-based interface
- Provides data access methods:
  - Direct path
  - External tables
- Detaches from and reattaches to long-running jobs
- Restarts Data Pump jobs

Data Pump: Overview

Data Pump enables very high-speed data and metadata loading and unloading of Oracle databases. The Data Pump infrastructure is callable via the `DBMS_DATAPUMP` PL/SQL package. Thus, custom data movement utilities can be built by using Data Pump.

Oracle Database 10*g* provides the following tools:

- Command-line export and import clients called `expdp` and `impdp` respectively
- A Web-based export and import interface that is accessible from Database Control

Data Pump automatically decides the data access methods to use; these can be either direct path or external tables. Data Pump uses direct path load and unload when a table's structure allows it and when maximum single-stream performance is desired. However, if there are clustered tables, referential integrity constraints, encrypted columns, or a number of other items, Data Pump uses external tables rather than direct path to move the data.

The ability to detach from and reattach to long-running jobs without affecting the job itself enables you to monitor jobs from multiple locations while they are running. All stopped Data Pump jobs can be restarted without loss of data as long as the metainformation remains undisturbed. It does not matter whether the job is stopped voluntarily or involuntarily due to a crash.

Note: Data Pump is an integral feature of Oracle Database 10*g* and is, therefore, available in all configurations. However, parallelism is available in Enterprise Edition only.

Data Pump: Benefits

- Fine-grained object and data selection
- Explicit specification of database version
- Parallel execution
- Estimation of the export job space consumption
- Network mode in a distributed environment
- Remapping capabilities during import
- Data sampling and metadata compression

18-14

Data Pump: Benefits

The `EXCLUDE`, `INCLUDE`, and `CONTENT` parameters are used for fine-grained object and data selection.

You can specify the database version for objects to be moved (using the `VERSION` parameter) to create a dump file set that is compatible with a previous release of the Oracle database that supports Data Pump.

You can use the `PARALLEL` parameter to specify the maximum number of threads of active execution servers operating on behalf of the export job.

You can estimate how much space an export job would consume (without actually performing the export) by using the `ESTIMATE_ONLY` parameter.

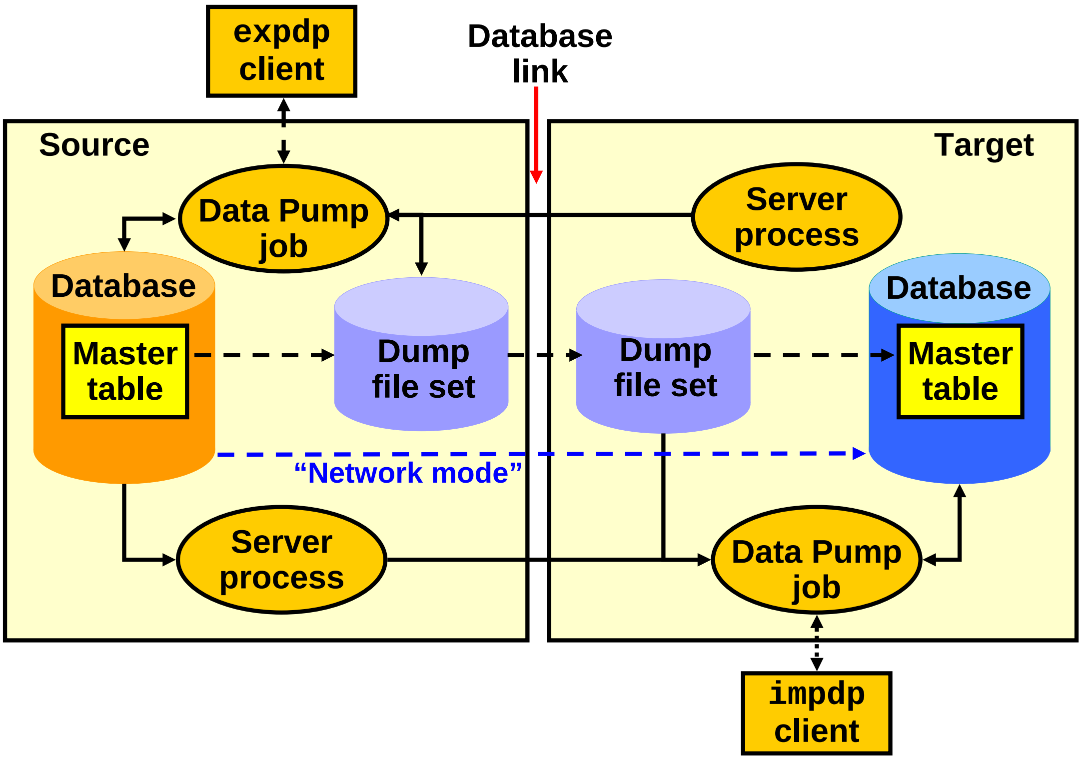
Network mode enables you to export from a remote database directly to a dump file set. This can be done by using a database link to the source system.

During import, you can change the target data file names, schemas, and tablespaces.

In addition, Oracle Database 10*g* enables you to specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export. This can be done by specifying the `SAMPLE` parameter.

You can use the `COMPRESSION` parameter to indicate whether the metadata should be compressed in the export dump file so that it consumes less disk space. If you compress the metadata, it is automatically uncompressed during import.

Data Pump Export and Import: Overview



Data Pump Export and Import: Overview

Data Pump Export is a utility for unloading data and metadata into a set of operating system files called dump file sets. Data Pump Import is used to load metadata and data stored in an export dump file set into a target system.

The Data Pump API accesses its files on the server rather than on the client.

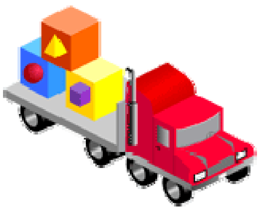
These utilities can also be used to export from a remote database directly to a dump file set, or to load the target database directly from a source database with no intervening files. This is known as network mode. This mode is particularly useful to export data from a read-only source database.

At the center of every Data Pump operation is the master table (MT), which is a table created in the schema of the user running the Data Pump job. The MT maintains all aspects of the job. The MT is built during a file-based export job and is written to the dump file set as the last step. Conversely, loading the MT into the current user's schema is the first step of a file-based import operation and is used to sequence the creation of all objects imported.

Note: The MT is the key to Data Pump's restart capability in the event of a planned or unplanned stopping of the job. The MT is dropped when the Data Pump job finishes normally.

Data Pump Utility: Interfaces and Modes

- Data Pump Export and Import interfaces:
  – Command line
  – Parameter file
  – Interactive command line
  – Database Control
- Data Pump Export and Import modes:
  – Full
  – Schema
  – Table
  – Tablespace
  – Transportable tablespace



ORACLE

18-16

Data Pump Utility: Interfaces and Modes

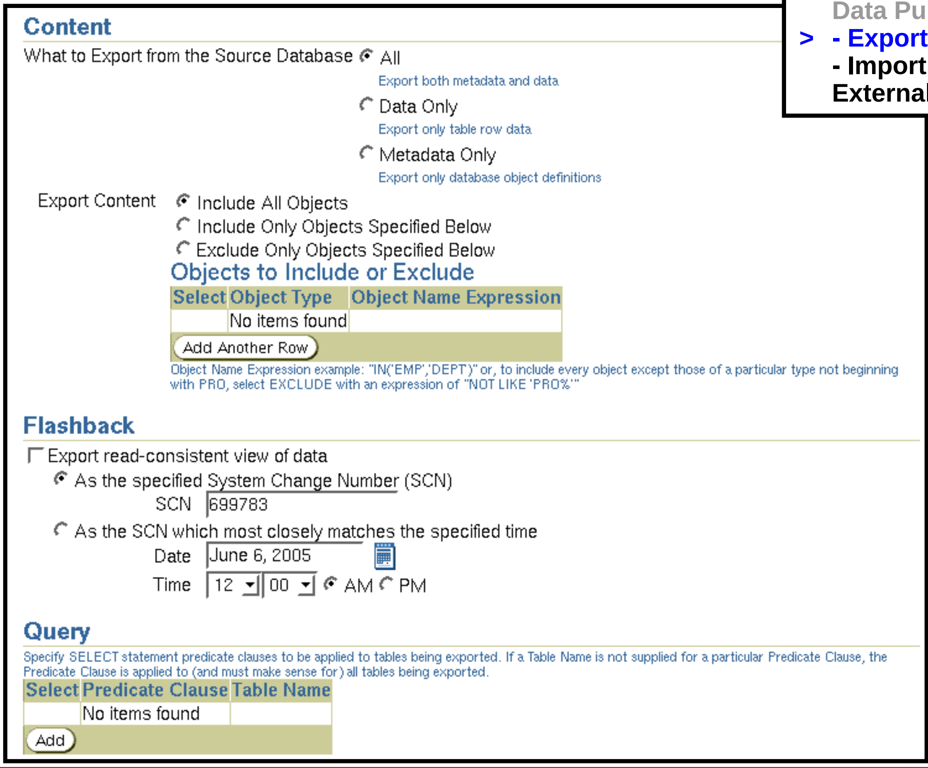
You can interact with Data Pump Export and Import by using one of the following interfaces:

- The command-line interface enables you to specify most of the export parameters directly on the command line.
- The parameter file interface enables you to specify all command-line parameters in a parameter file. The only exception is the `PARFILE` parameter.
- The interactive-command interface stops logging to the terminal and displays the export or import prompts, where you can enter various commands. This mode is enabled by pressing [Ctrl] + [C] during an export operation that is started with the command-line interface or the parameter file interface. Interactive-command mode is also enabled when you attach to an executing or stopped job.
- You can also access the Web interface. On the Database Control home page, click the Maintenance tab, and then select one of the following links from the Utilities region: Export to Files, Import from Files, or Import from Database.

Data Pump Export and Import provide different modes for unloading and loading different portions of the database. The mode is specified on the command line by using the appropriate parameter.

Fine-Grained Object Selection

Directory Obj.
SQL*Loader
Data Pump
> - Export
- Import
External Table



ORACLE

Fine-Grained Object Selection

The Data Pump job can include or exclude virtually any type of object.

The `EXCLUDE` parameter enables any database object type to be excluded from an export or import operation. The optional name qualifier enables you to have finer selectivity within each object type that is specified. Examples:

```
EXCLUDE=VIEW
EXCLUDE=PACKAGE
EXCLUDE=INDEX:"LIKE 'EMP%'"
```

The `INCLUDE` parameter includes only the specified object types and objects in an operation. Syntax:

```
INCLUDE  = object_type[:"name_expr"]
```

The `CONTENT` parameter enables you to request for the current operation, only the metadata, only the data, or both. Syntax:

```
CONTENT = ALL | METADATA_ONLY | DATA_ONLY
```

The `QUERY` parameter operates in a similar manner as the original export utility, with two significant enhancements: It can be qualified with a table name so that it applies to only that table, and it can be used during import as well. Example:

```
QUERY=hr.employees:"WHERE department_id in (10,20) and salary
< 1600 ORDER BY department_id"
```

Advanced Feature: Sampling

- **Task: Create test data.**
- **Method: Specify a percentage of data to be sampled and unloaded from the source database.**

Example to unload 44% of the `HR.EMPLOYEES` table:

```
SAMPLE="HR"."EMPLOYEES":44
```

Example to unload 30% of the entire export job (because no table name is specified):

```
expdp hr/hr DIRECTORY=DATA_PUMP_DIR
DUMPFILE=sample1.dmp SAMPLE=30
```


Advanced Feature: Sampling

With the `SAMPLE` parameter, you can specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export.

Syntax:

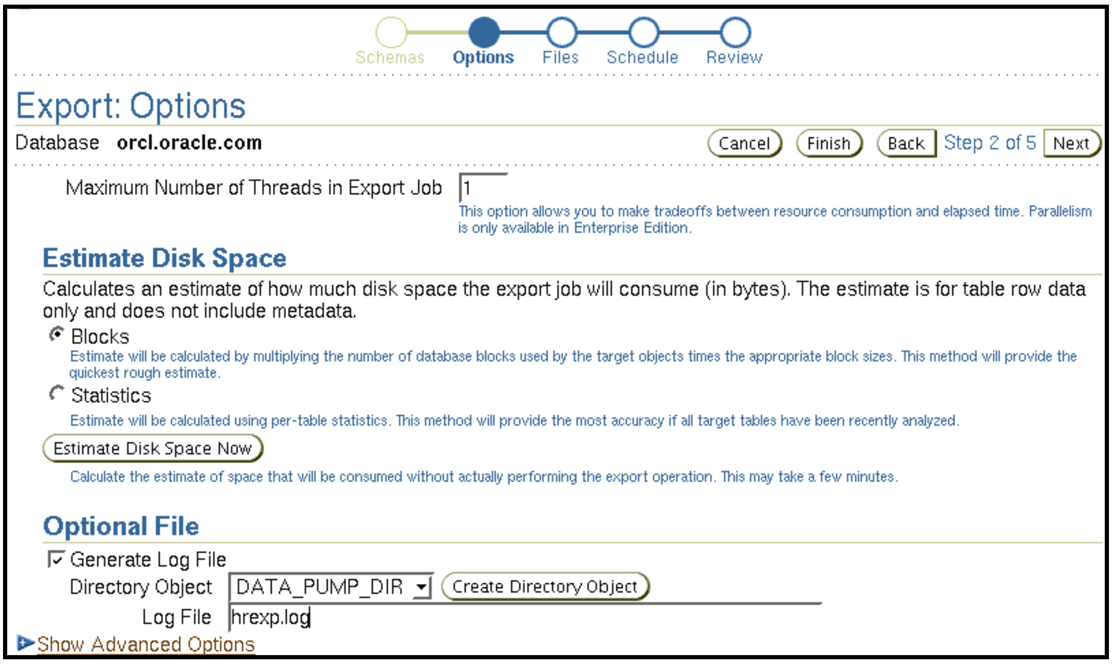
```
SAMPLE=[[schema_name.]table_name:]sample_percent
```

Range for *`sample_percent`*: `.000001` to (but not including) `100`

Sample percentage indicates the likelihood that a block of rows will be included.

Note: The `SAMPLE` parameter is not valid for network exports.

Export Options: Files



18-19

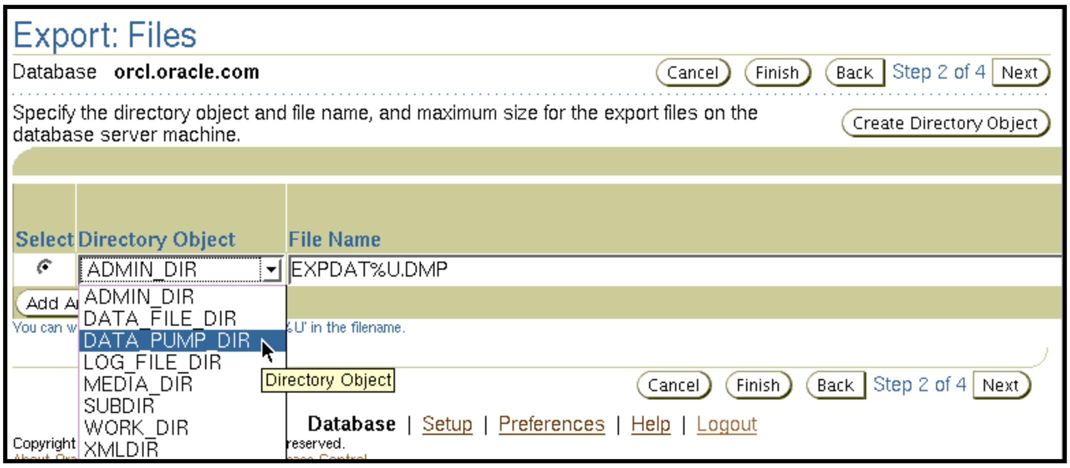
Export Options: Files

There are three types of files that are managed by Data Pump jobs:

- Dump files for data and metadata that is to be moved
- Log files for messages
- SQL files for the output of a `SQLFILE` operation

Because Data Pump is server based and not client based, Data Pump files are accessed relative to Oracle directory paths. Absolute paths are not supported for security reasons.

Data Pump File Locations



The order of precedence of file locations:

- **Per-file directory**
- **The `DIRECTORY` parameter**
- **The `DATA_PUMP_DIR` environment variable**
- **`DATA_PUMP_DIR` directory object**

18-20

Data Pump File Locations

The slide shows you the order of precedence used by Data Pump clients to locate these files:

- Per-file directory objects may be specified for each dump file, log file, and SQL file. If specified, they are separated from the file name by a colon (:).
- The Data Pump Export and Import clients provide a `DIRECTORY` parameter, which specifies the name of a directory object. These directory objects describe the location in which the files are accessed.
- You can alternatively define an environment variable, `DATA_PUMP_DIR`, to specify the directory object name rather than use the `DIRECTORY` parameter. The Data Pump clients look for this environment variable if no explicit directory object is specified.
- There is a default directory object created for every database. This directory object is named `DATA_PUMP_DIR`. Access to the `DATA_PUMP_DIR` directory is granted automatically to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles.

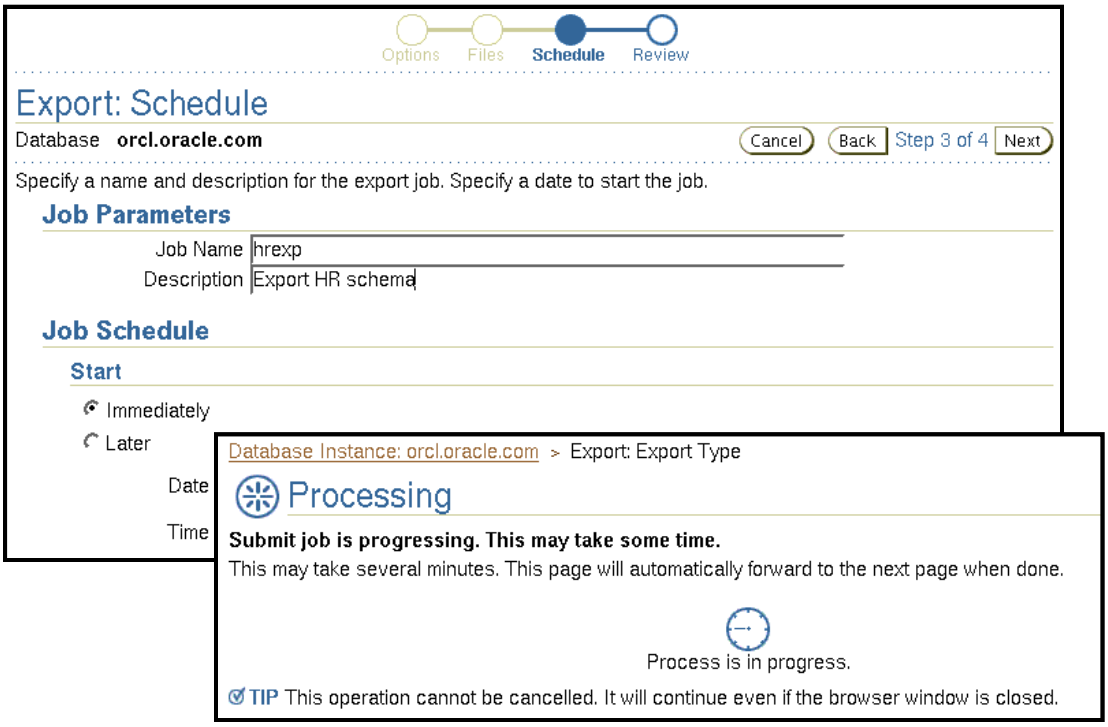
Data Pump File Locations (continued)

- You do not need to create a directory object manually before using Data Pump Export. There is a default directory object created for every database, whether newly created or upgraded by a script on UNIX or Windows platforms. This directory object is named DATA_PUMP_DIR. Access to the DATA_PUMP_DIR directory is granted automatically to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles. The DATA_PUMP_DIR directory is created in one of the following locations:
  - <ORACLE_BASE>/admin/ORACLE_SID
  - <ORACLE_HOME>/admin/ORACLE_SID

  The exact directory path specification for DATA_PUMP_DIR varies, depending on the value of the ORACLE_BASE and ORACLE_HOME system environment variables and on the existence of the DATA_PUMP_DIR subdirectory. If ORACLE_BASE is defined on the target system, then that value is used. Otherwise, the value of ORACLE_HOME is used. If, for some reason, the DATA_PUMP_DIR subdirectory is not found, the following default path is used: ORACLE_HOME/rdbms/log.

Note: In all cases, you must have the appropriate access privileges to the directory object for the operation attempted. For export, you need write access for all files; for import, you need read access for dump files and write access for log files and SQL files.

Scheduling and Running a Job

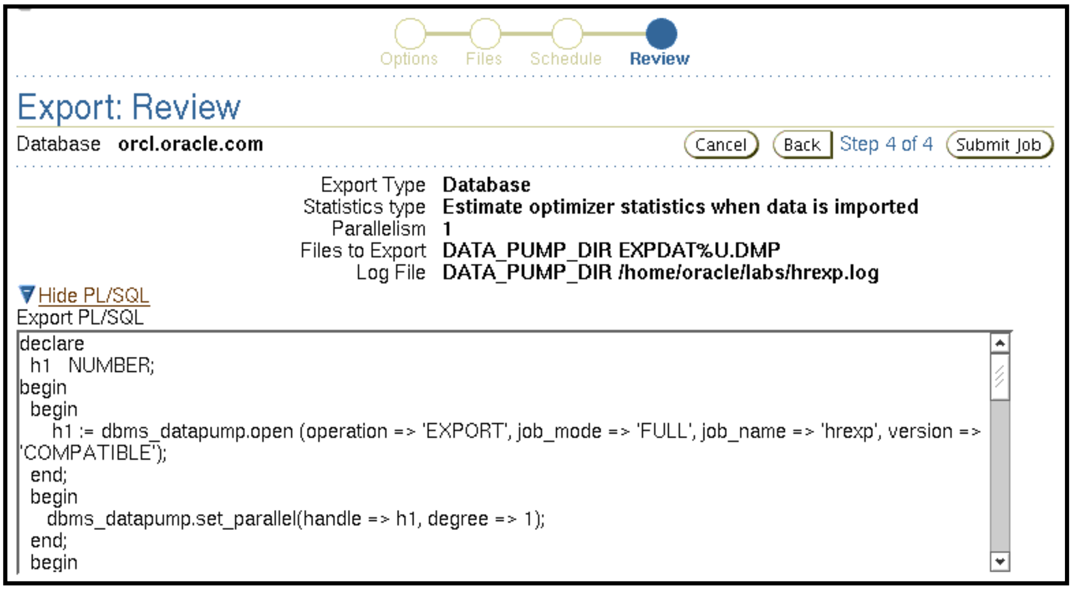


18-22

Scheduling and Running a Job

Data Pump jobs (created through this wizard) are scheduled as repeatable jobs by Enterprise Manager Database Control.

Data Pump File Naming and Size



Data Pump File Naming and Size

The DUMPFILE parameter specifies the names and (optionally) directories of disk-based dump files. Multiple file specifications may be provided as a comma-separated list or in separate DUMPFILE parameter specifications. File names may contain the substitution variable %U, which implies that multiple files may be generated. %U is expanded in the resulting file names into a two-character, fixed-width, monotonically increasing integer starting at 01. If no DUMPFILE is specified, expdat.dmp is used by default. By default, created dump files are autoextensible.

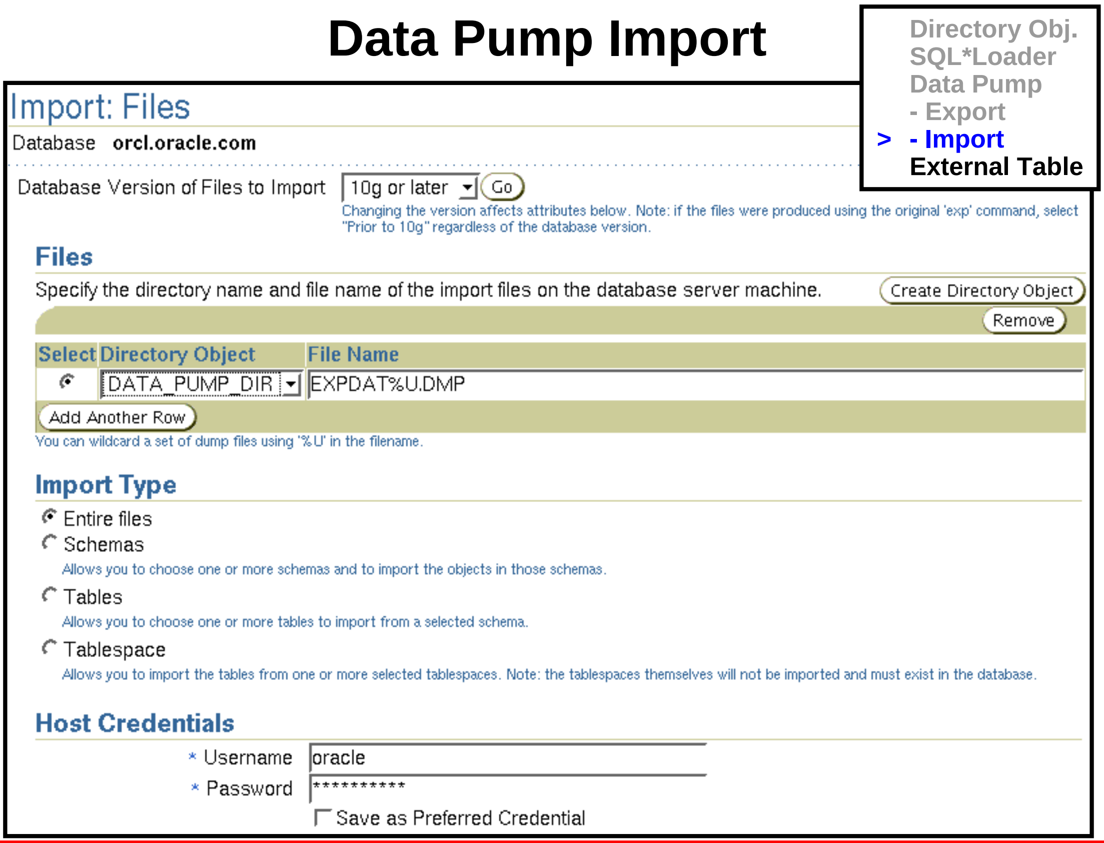
If FILESIZE is specified, each file is FILESIZE bytes in size and nonextensible. If more dump space is required and a template with %U has been supplied, then a new file is automatically created with FILESIZE bytes; otherwise, the client receives a message to add a new file.

If a template with %U is specified, the number of files initially created is equal to the PARALLEL parameter.

Preexisting files that match the resulting file names are not overwritten; they result in an error and cause the job to be aborted.

Note: If multiple dump file templates are provided, then they are used to generate dump files in a circular fashion.

Data Pump Import



18-24

Data Pump Import

Data Pump Import is a utility for loading an export dump file set into a target system. The dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format. During an import operation, the Data Pump Import utility uses these files to locate each database object in the dump file set.

You can interact with Data Pump Import by using a command line, a parameter file, or an interactive-command mode:

- You can use the `impdp` command and specify parameters directly on the command line.
- You can enter command-line parameters in a file (the `PARFILE` parameter is excluded because parameter files cannot be nested).
- In interactive-command mode, the current job continues running, but logging to the terminal is stopped and the Import prompt is displayed. You can, for example, attach additional jobs to an executing or stopped job.

Data Pump Import: Transformations

You can remap:

- Data files by using `REMAP_DATAFILE`
- Tablespaces by using `REMAP_TABLESPACE`
- Schemas by using `REMAP_SCHEMA`

```
REMAP_DATAFILE = 'C:\oradata\tbs6.f':'/u1/tbs6.f'
```

18-25

Data Pump Import: Transformations

Because object metadata is stored as XML in the dump file set, it is easy to apply transformations when DDL is being formed during import. Data Pump Import supports several transformations:

- `REMAP_DATAFILE` is useful when moving databases across platforms that have different file-system semantics.
- `REMAP_TABLESPACE` allows objects to be moved from one tablespace to another.
- `REMAP_SCHEMA` provides the old `FROMUSER` /`TOUSER` capability to change object ownership.

Data Pump Import: Transformations

Using `TRANSFORM`, you can also :

- **Exclude from tables and indexes:**
  - **`STORAGE` and `TABLESPACE` clauses**
  - **`STORAGE` clause only**
- **Re-create object identifiers of abstract data types**
- **Change extent allocations and file size**

```
TRANSFORM =
SEGMENT_ATTRIBUTES|STORAGE|OID|PCTSPACE:{y|n|v}[:object type]
```

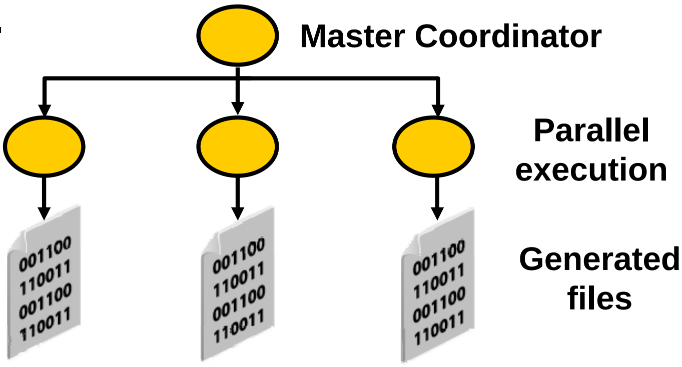

Data Pump Import: Transformations (continued)

The `TRANSFORM` parameter enables you to alter the object creation DDL for specific objects or for all applicable objects being loaded. Specify the `TRANSFORM` parameter as shown in the slide. The following are possible options:

- `SEGMENT_ATTRIBUTES`: If the value is specified as `Y`, segment attributes (physical attributes, storage attributes, tablespaces, and logging) are included.
- `STORAGE`: If the value is specified as `Y`, the `STORAGE` clauses are included.
- `OID`: You can use this parameter to determine whether the object ID (OID) of abstract data types is reused or created anew. If the value is specified as `N`, then the generation of the export OID clause for object types is suppressed. This is useful when you need to duplicate schemas across databases by using export and import, but you cannot guarantee that the object types will have identical OID values in those databases.
- `PCTSPACE`: You can use the `PCTSPACE` parameter to reduce the amount of space required for tablespaces by performing a shrink operation on tablespace storage allocation. The value supplied for this transformation must be a number greater than zero. It represents the percentage multiplier used to alter extent allocations and the size of data files.

Data Pump: Performance Consideration

Maximizing job performance with the PARALLEL parameter.



Example:

```
expdp hr/hr FULL=y
DUMPFILE=dp_dir1:full1%U.dmp, dp_dir2:full2%U.dmp
FILESIZE=2G PARALLEL=3
LOGFILE=dp_dir1:expfull.log JOB_NAME=expfull
```

18-27

Data Pump: Performance Consideration

You can improve throughput of a job with the `PARALLEL` parameter. The parallelism setting is enforced by the master process, which allocates work to be executed to worker processes that perform the data and metadata processing within an operation. These worker processes operate in parallel. In general, the degree of parallelism should be set to more than twice the number of CPUs on an instance. To maximize parallelism, you must supply at least one file for each degree of parallelism. If there are not enough dump files, the performance will not be optimal because multiple threads of execution will be trying to access the same dump file. The degree of parallelism can be reset at any time during a job.

The example in the slide shows a full database export. All data and metadata in the database will be exported. Dump files (`full101.dmp`, `full201.dmp`, `full102`.dmp, and so on) will be created in a round-robin fashion in the directories pointed to by the `dp_dir1` and `dp_dir2` directory objects. For best performance, these should be on separate I/O channels. Each file will be up to 2 gigabytes in size, as necessary. Initially, up to three files will be created. More files will be created, if needed. The job and master table have the same name: `expfull`. The log file will be written to `expfull.log` in the `dp_dir1` directory.

Performance Initialization Parameters

- Performance of Data Pump can be affected by:
  - `DISK_ASYNCH_IO=TRUE`
  - `DB_BLOCK_CHECKING=FALSE`
  - `DB_BLOCK_CHECKSUM=FALSE`
- The following should be set high to allow for maximum parallelism:
  - `PROCESSES`
  - `SESSIONS`
  - `PARALLEL_MAX_SERVERS`
- The following should be sized generously:
  - `SHARED_POOL_SIZE`
  - `UNDO_TABLESPACE`

ORACLE

18-28

Performance Initialization Parameters

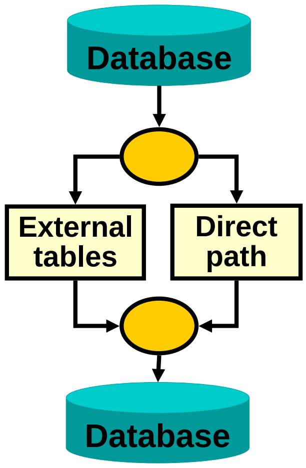
You can try using the parameters (shown in the slide) to improve performance, although the effect may not be the same on all platforms.

Additionally, the `SHARED_POOL_SIZE` and `UNDO_TABLESPACE` initialization parameters should be generously sized. The exact values will depend upon the size of your database.

Data Pump Access Path: Considerations

One of the following access paths is automatically selected by Data Pump:

- **Direct path**
- **External tables, if data includes:**
  - **Encrypted columns**
  - **Clustered tables**
  - **Different partition at unload and load time, and others (see Notes)**



Data Pump Direct Path: Considerations

Data Pump automatically selects the most appropriate access method for each table.

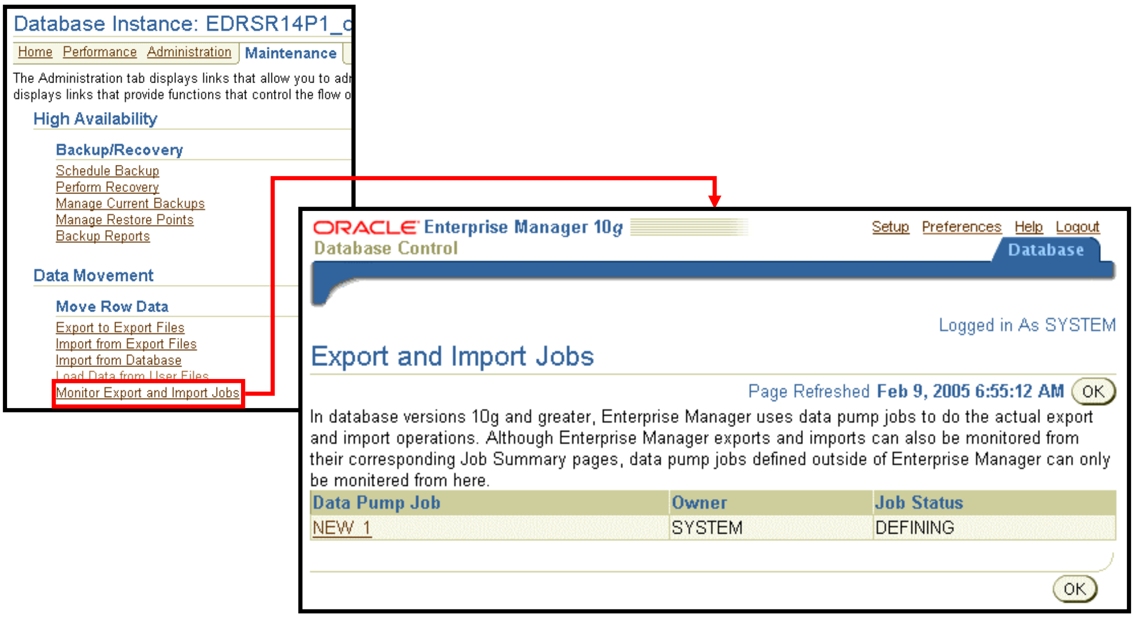
Data Pump uses direct path load and unload when a table's structure allows it and when maximum single-stream performance is desired.

Data Pump uses external tables, if any of the conditions exist:

- Tables with fine-grained access control enabled in insert and select modes
- Domain index, which exists for a LOB column
- Tables with active triggers defined
- Global index on partitioned tables with a single-partition load
- `BFILE` or opaque type columns
- Referential integrity constraint
- VARRAY columns with an embedded opaque type

Note: Because both methods support the same external data representation, data that is unloaded with one method can be loaded using the other method.

Using Enterprise Manager to Monitor Data Pump Jobs



Using Enterprise Manager to Monitor Data Pump Jobs

You can use the Enterprise Manager graphical user interface (GUI) to monitor all Data Pump jobs, including those created by using the `expdp` or `impdp` command-line interfaces or by using the `DBMS_DATAPUMP` package.

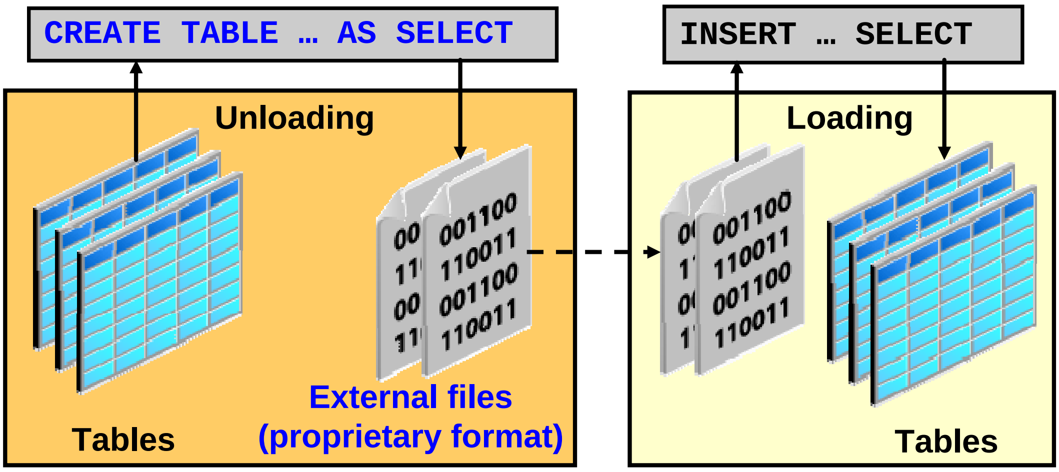
You can view the current status of the job and also change the status to `EXECUTE`, `STOP`, or `SUSPEND`.

To access the Export and Import Jobs page, click the Monitor Export and Import Jobs link in the Move Row Data region of the Maintenance page.

External Table Population

Directory Obj.
SQL*Loader
Data Pump
- Export
- Import
> External Table

- Unloading of data to external files with the `ORACLE_DATAPUMP` access driver
- No modifications of external tables



External Table Population

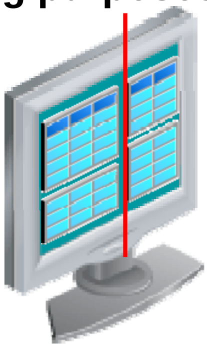
An "external table" is composed of proprietary format (that is, Direct Path API) flat files that are operating system independent. As data is extracted from the Oracle database and "unloaded" into files, it is transparently converted from its Oracle internal representation into an equivalent Oracle native external representation (that is, DPAPI).

You may use the `CREATE TABLE AS SELECT` command to populate an external table. After an external table has been created and populated, no rows may be added, updated, or deleted from the external table. Any attempt to modify the data in the external table fails. An external table may not have indexes.

The Data Pump access driver enables the unloading and loading operations for external tables.

Using External Tables

- Data can be used directly from the external file or loaded into another database.
- Resulting files can be read only with the `ORACLE_DATAPUMP` access driver.
- You can combine generated files from different sources for loading purposes.



Using External Tables

The data files created for the external table can be moved and used as the data files for another external table in the same database or different database. They can be read only by the `ORACLE_DATAPUMP` access driver. You can choose whether your applications should directly access external tables with the `SELECT` command, or if the data should first be loaded into a target database.

Data files populated by different external tables can all be specified in the `LOCATION` clause of another external table. This provides an easy way of aggregating data from multiple sources. The only restriction is that the metadata for all the external tables must be exactly the same.

External Table Population with ORACLE_DATAPUMP

```
CREATE TABLE emp_ext
  (first_name, last_name, department_name)
ORGANIZATION EXTERNAL
  (
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY ext_dir
    LOCATION ('emp1.exp','emp2.exp','emp3.exp')
  )
PARALLEL
AS
SELECT e.first_name,e.last_name,d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id AND
       d.department_name in
                     ('Marketing', 'Purchasing');
```


External Table Population with ORACLE_DATAPUMP

This example shows you how the new external table population operation can help to export a selective set of records resulting from the join of the EMPLOYEES and DEPARTMENTS tables.

Because the external table can be large, you can use a parallel populate operation to unload your data to an external table. As opposed to a parallel query from an external table, the degree of parallelism of a parallel populate operation is constrained by the number of concurrent files that can be written to by the access driver. There is never more than one parallel execution server writing into one file at a particular point in time.

The number of files in the LOCATION clause must match the specified degree of parallelism because each input/output (I/O) server process requires its own file. Any extra files that are specified are ignored. If there are not enough files for the specified degree of parallelism, the degree of parallelization is lowered to match the number of files in the LOCATION clause.

Note: For more information about the ORACLE_DATAPUMP access driver parameters, see the *Oracle Database Utilities* guide.

External Table Population with ORACLE_LOADER

```
CREATE TABLE extab_employees
                 (employee_id       NUMBER(4),
                  first_name        VARCHAR2(20),
                  last_name         VARCHAR2(25),
                  hire_date         DATE)
ORGANIZATION EXTERNAL
    ( TYPE ORACLE_LOADER DEFAULT DIRECTORY extab_dat_dir
      ACCESS PARAMETERS
      ( records delimited by newline
        badfile extab_bad_dir:'empxt%a_%p.bad'
        logfile extab_log_dir:'empxt%a_%p.log'
        fields terminated by ','
        missing field values are null
    ( employee_id, first_name, last_name,
     hire_date char date_format date mask "dd-mon-yyyy“))
    LOCATION ('empxt1.dat', 'empxt2.dat') )
    PARALLEL  REJECT LIMIT UNLIMITED;
```

18-34

External Table Population with ORACLE_LOADER

The ORACLE_LOADER access driver uses the SQL*Loader syntax to create external tables.

The example shown in the slide assumes that three directory objects (extab_dat_dir, extab_bad_dir, and extab_log_dir) are created and mapped to existing OS directories, to which the user is granted access.

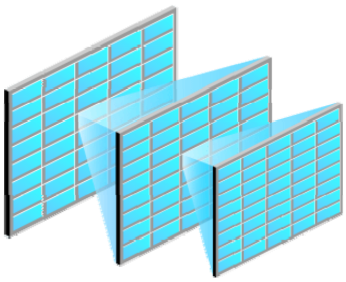
Tip: If you have a lot of data to load, enable PARALLEL for the load operation:

```
ALTER SESSION ENABLE PARALLEL DML;
```

Data Dictionary

View information about external tables in:

- **`[DBA| ALL| USER]_EXTERNAL_TABLES`**
- **`[DBA| ALL| USER]_EXTERNAL_LOCATIONS`**
- **`[DBA| ALL| USER]_TABLES`, and others**



Data Dictionary

`[DBA| ALL| USER]_EXTERNAL_TABLES` list the specific attributes of external tables in the database.

`[DBA| ALL| USER]_EXTERNAL_LOCATIONS` list the data sources for external tables.

`[DBA| ALL| USER]_TABLES` describe relational tables in the database.

`[DBA| ALL| USER]_TAB_COLUMNS` describe the columns of tables, views, and clusters in the database.

Summary

In this lesson, you should have learned how to:

- Describe available ways for moving data
- Create and use directory objects
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Explain the general architecture of Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Use external tables to move data via platform-independent files

ORACLE

Practice Overview: Moving Data

This practice covers the following topics:

- Using the Data Pump Export Wizard to select database objects to be exported
- Monitoring a Data Pump Export job
- Using the Data Pump Import Wizard to import tables in your database
- Using the Load Data Wizard to load data into your database
- Loading data by using the command line

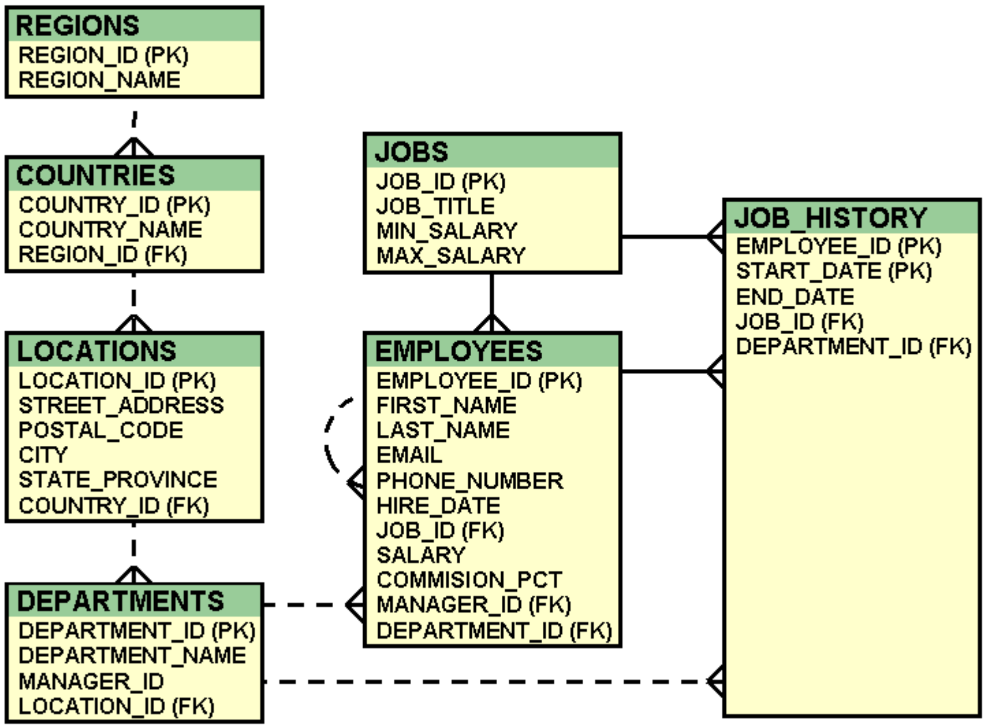
ORACLE

Appendix A

Practices

Practice Sessions: Overview

The HR schema:



Data Definition Language (DDL) Scripts

DDL scripts can be used to create the initial HR schema.

Table Definitions

```
PROMPT Creating Table 'REGIONS'
CREATE TABLE REGIONS
 (REGION_ID FLOAT(53) NOT NULL
 ,REGION_NAME VARCHAR2(25)
 )
/

PROMPT Creating Table 'JOBS'
CREATE TABLE JOBS
 (JOB_ID VARCHAR2(10) NOT NULL
 ,JOB_TITLE VARCHAR2(35) NOT NULL
 ,MIN_SALARY NUMBER(10,0)
 ,MAX_SALARY NUMBER(10,0)
 )
/

PROMPT Creating Table 'LOCATIONS'
CREATE TABLE LOCATIONS
 (LOCATION_ID NUMBER(5,0) NOT NULL
 ,STREET_ADDRESS VARCHAR2(40)
 ,POSTAL_CODE VARCHAR2(12)
 ,CITY VARCHAR2(30) NOT NULL
 ,STATE_PROVINCE VARCHAR2(25)
 ,COUNTRY_ID CHAR(2)
 )
/

PROMPT Creating Table 'JOB_HISTORY'
CREATE TABLE JOB_HISTORY
 (EMPLOYEE_ID NUMBER(10,0) NOT NULL
 ,START_DATE DATE NOT NULL
 ,END_DATE DATE NOT NULL
 ,JOB_ID VARCHAR2(10) NOT NULL
 ,DEPARTMENT_ID NUMBER(5,0)
 )
/

PROMPT Creating Table 'DEPARTMENTS'
CREATE TABLE DEPARTMENTS
 (DEPARTMENT_ID NUMBER(5,0) NOT NULL
 ,DEPARTMENT_NAME VARCHAR2(30) NOT
NULL
 ,MANAGER_ID NUMBER(10,0)
 ,LOCATION_ID NUMBER(5,0)
 )
/
```

Practice Sessions: Overview (continued)

```
PROMPT Creating Table 'EMPLOYEES'
CREATE TABLE EMPLOYEES
 (EMPLOYEE_ID NUMBER(10,0) NOT NULL
 ,FIRST_NAME VARCHAR2(20)
 ,LAST_NAME VARCHAR2(25) NOT NULL
 ,EMAIL VARCHAR2(25) NOT NULL
 ,PHONE_NUMBER VARCHAR2(20)
 ,HIRE_DATE DATE NOT NULL
 ,JOB_ID VARCHAR2(10) NOT NULL
 ,SALARY NUMBER(8,2)
 ,COMMISSION_PCT NUMBER(2,2)
 ,MANAGER_ID NUMBER(10,0)
 ,DEPARTMENT_ID NUMBER(5,0)
 )
/

PROMPT Creating Table 'COUNTRIES'
CREATE TABLE COUNTRIES
 (COUNTRY_ID CHAR(2) NOT NULL
 ,COUNTRY_NAME VARCHAR2(40)
 ,REGION_ID FLOAT(53)
 )
/
```

Constraint Definitions

```
PROMPT Creating Primary Key on
'REGIONS'
ALTER TABLE REGIONS
 ADD (CONSTRAINT
PK__REGIONS__76CBA758 PRIMARY KEY
  (REGION_ID))
/

PROMPT Creating Primary Key on 'JOBS'
ALTER TABLE JOBS
 ADD (CONSTRAINT PK__JOBS__023D5A04
PRIMARY KEY
  (JOB_ID))
/

PROMPT Creating Primary Key on
'LOCATIONS'
ALTER TABLE LOCATIONS
 ADD (CONSTRAINT
PK__LOCATIONS__7B905C75 PRIMARY KEY
  (LOCATION_ID))
/

PROMPT Creating Primary Key on
'JOB_HISTORY'
ALTER TABLE JOB_HISTORY
 ADD (CONSTRAINT
PK__JOB_HISTORY__0AD2A005 PRIMARY KEY
  (EMPLOYEE_ID
  ,START_DATE))
/

PROMPT Creating Primary Key on
'DEPARTMENTS'
ALTER TABLE DEPARTMENTS
 ADD (CONSTRAINT
PK__DEPARTMENTS__7E6CC920 PRIMARY KEY
  (DEPARTMENT_ID))
/

PROMPT Creating Primary Key on
'EMPLOYEES'
ALTER TABLE EMPLOYEES
 ADD (CONSTRAINT
PK__EMPLOYEES__0425A276 PRIMARY KEY
  (EMPLOYEE_ID))
/

PROMPT Creating Primary Key on
'COUNTRIES'
ALTER TABLE COUNTRIES
 ADD (CONSTRAINT
PK__COUNTRIES__78B3EFCA PRIMARY KEY
  (COUNTRY_ID))
/

PROMPT Creating Unique Key on
'EMPLOYEES'
ALTER TABLE EMPLOYEES
 ADD (CONSTRAINT
UQ__EMPLOYEES__0519C6AF UNIQUE
  (EMAIL))
/

PROMPT Creating Check Constraint on
'JOB_HISTORY'
ALTER TABLE JOB_HISTORY
 ADD (CONSTRAINT
CK__JOB_HISTORY__0EA330E9 CHECK
(END_DATE > START_DATE))
/
```

Practice Sessions: Overview (continued)

```
PROMPT Creating Check Constraint on
'EMPLOYEES'
ALTER TABLE EMPLOYEES
 ADD (CONSTRAINT
CK__EMPLOYEES__SALAR__08EA5793 CHECK
(SALARY > 0))
/


PROMPT Creating Foreign Key on
'LOCATIONS'
ALTER TABLE LOCATIONS ADD (CONSTRAINT
 FK__LOCATIONS__COUNT__7C8480AE
FOREIGN KEY
  (COUNTRY_ID) REFERENCES COUNTRIES
  (COUNTRY_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'JOB_HISTORY'
ALTER TABLE JOB_HISTORY ADD
(CONSTRAINT
 FK__JOB_HISTO__DEPAR__0BC6C43E
FOREIGN KEY
  (DEPARTMENT_ID) REFERENCES
DEPARTMENTS
  (DEPARTMENT_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'JOB_HISTORY'
ALTER TABLE JOB_HISTORY ADD
(CONSTRAINT
 FK__JOB_HISTO__JOB_I__0DAF0CB0
FOREIGN KEY
  (JOB_ID) REFERENCES JOBS
  (JOB_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'JOB_HISTORY'
ALTER TABLE JOB_HISTORY ADD
(CONSTRAINT
 FK__JOB_HISTO__EMPLO__0CBAE877
FOREIGN KEY
  (EMPLOYEE_ID) REFERENCES EMPLOYEES
  (EMPLOYEE_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'DEPARTMENTS'
ALTER TABLE DEPARTMENTS ADD
(CONSTRAINT
 FK__DEPARTMEN__LOCAT__7F60ED59
FOREIGN KEY
  (LOCATION_ID) REFERENCES LOCATIONS
  (LOCATION_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'EMPLOYEES'
ALTER TABLE EMPLOYEES ADD (CONSTRAINT
 FK__EMPLOYEES__MANAG__07F6335A
FOREIGN KEY
  (MANAGER_ID) REFERENCES EMPLOYEES
  (EMPLOYEE_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'EMPLOYEES'
ALTER TABLE EMPLOYEES ADD (CONSTRAINT
 FK__EMPLOYEES__JOB_I__07020F21
FOREIGN KEY
  (JOB_ID) REFERENCES JOBS
  (JOB_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'EMPLOYEES'
ALTER TABLE EMPLOYEES ADD (CONSTRAINT
 FK__EMPLOYEES__DEPAR__060DEAE8
FOREIGN KEY
  (DEPARTMENT_ID) REFERENCES
DEPARTMENTS
  (DEPARTMENT_ID) ON DELETE CASCADE)
/

PROMPT Creating Foreign Key on
'COUNTRIES'
ALTER TABLE COUNTRIES ADD (CONSTRAINT
 FK__COUNTRIES__REGIO__79A81403
FOREIGN KEY
  (REGION_ID) REFERENCES REGIONS
  (REGION_ID) ON DELETE CASCADE)
```

Practice 2: Installing the Oracle Database Software

Background: In the practices of this course, you assume the role of a database administrator (DBA). The operating system (OS) accounts on your computer 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 installation, and the installation media is staged at `/stage/Disk1`. Perform the following tasks as the default `oracle` OS user, unless otherwise indicated.

Use the Oracle Universal Installer (OUI) to begin your software installation of the Enterprise Edition of Oracle Database 10*g* Release 2. This is a "Basic Installation" of the Oracle software.

After installing the software, execute the `lab_02_05.sh` script, which creates a listener for you.

Note: Completing this practice is critical for all following practice sessions.

Your Tasks

1. Install the Oracle database software as the `oracle` user. Navigate to the `/stage/Disk1` directory, and start the OUI by entering `./runInstaller`.

2. Select your installation method by entering and accepting the following settings:

Object	Setting
Installation Method	Basic Installation
Database Home Location	`/u01/app/oracle/product/10.2.0/db_1`
Installation Type	Enterprise Edition
UNIX DBA Group	`oinstall`
Create Starter Database	*Deselected*
Inventory Directory	`/u01/app/oracle/oraInventory`

3. After entering the initial settings, OUI checks product-specific prerequisites for you. When these checks are finished, install the Oracle software.

   Estimated installation time is 10–15 minutes. However, varying environments can greatly influence this estimate.

Practice 2: Installing the Oracle Database Software (continued)

4. When the Execute Configuration scripts page appears, follow the instructions on that page, accept the default for the local `bin` directory, and then finish your installation with OUI.

   Make a note of your End of Installation URLs. You will use them in later practice sessions.

   *i*SQL*PlusURL:____

   *i*SQL*Plus DBA URL:____

   Optionally, review the product inventory, and then exit.

5. Execute the `lab_02_05.sh` script, which is in the `/home/oracle/labs` directory. This script uses the `netca` utility to create a listener for you. If you create your first database with OUI (as part of your installation), then OUI invokes the network configuration assistant, which creates the first listener for you. For more details on listeners, see the lesson titled “Configuring the Oracle Network Environment.”

   You see an activity log, which should end with a success message. If not, resolve any errors that might have occurred.

Practice 3: Creating an Oracle Database

Background: You are about to begin creating your first Oracle database. You anticipate that several similar databases will be needed in the near future. Therefore, you decide to create your `ORCL` database, as well as a database template and the database creation scripts. Locate the scripts in the `/home/oracle/labs` directory (which is the directory that you use most often throughout this course).

Note: Completing the database creation is critical for all following practice sessions.

Your Tasks

1. Start the Database Configuration Assistant (DBCA) by entering `dbca` in a terminal window.

2. Begin the `ORCL` database creation. Use the General Purpose database template. Review the template's details and answer the following questions.

   Question 1: How many control files are created?

   ____

   Question 2: Would it maximize database availability to multiplex them?

   ____

   Question 3: How many redo log groups are created?

   ____

   Question 4: Would it maximize database availability to mirror them?

   ____

   ____

   Question 5: What is the database block size (`db_block_size`)?

   ____

   Question 6: What is the value of Sample Schemas?

   ____

   Note: You will change this setting later in this practice to create and unlock the `HR` sample schema.

Practice 3: Creating an Oracle Database (continued)

Question 7: What is the template default for the Database Character Set?

____

Note: You will change this setting later in this practice to use a Unicode database character set.

3. Create the `ORCL` database, as well as the `ORCL` template and the database generation scripts.

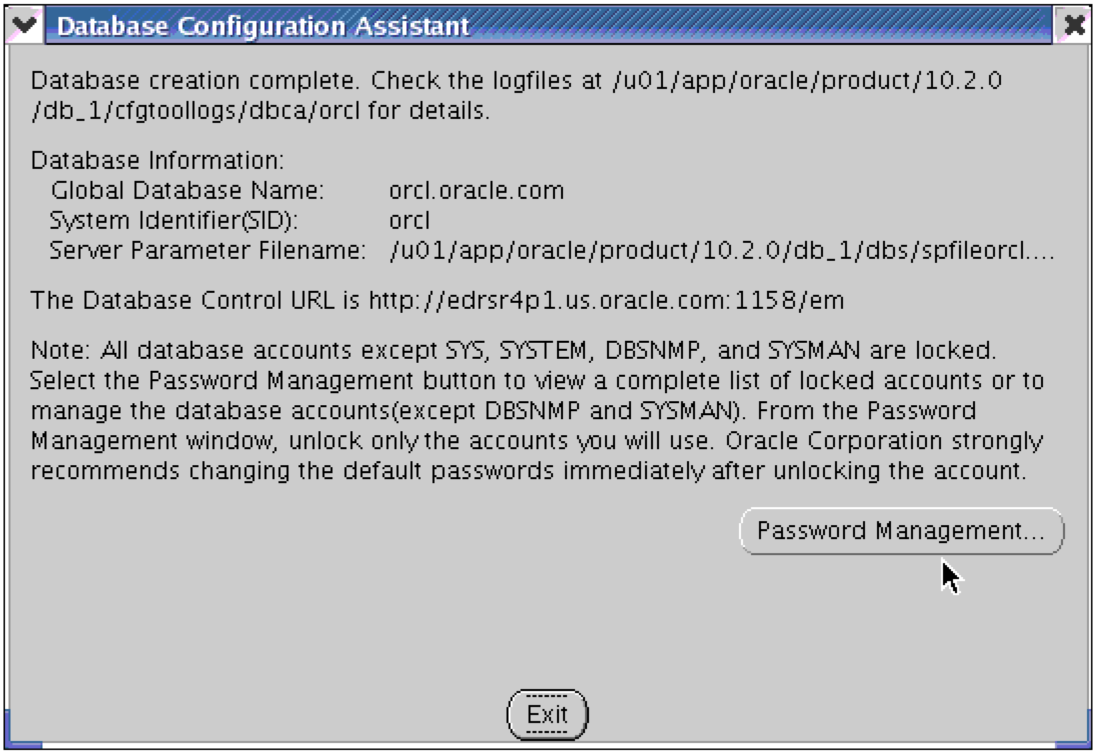
   Create a database with the following settings:

Object	**Setting**
Global Database Name	`orcl.oracle.com`
SID	`orcl`
Configuration	Configure the Database with Enterprise Manager
Database Management	Use Database Control for Database Management
Enable Email Notifications	*deselect*
Enable Daily Backup	*deselect*
Password for All Accounts	`oracle`
Storage Options	File System
File Location	Use Oracle-Managed Files
Recovery Configuration	Specify Flash Recovery Area
Enable Archiving	*deselect*
Database Content	Sample Schemas enabled
Initialization Parameters Memory	Custom and Automatic Shared Memory Management
Character Sets	Use Unicode (AL32UTF8)
Creation Options:	Select all: - Create Database - Save as a Database Template (optional) - Generate Database Creation Scripts (optional)
Template Name	`orcl`
Description	`ORCL Database template`
Destination Directory	`/home/oracle/labs`

   Review and confirm options and parameters, such as `Sample Schemas (true)`, `db_block_size (8KB)`, `sga_target (270MB)`, `undo_management (AUTO)`, and `Database Character Set (AL32UTF8)`.

Practice 3: Creating an Oracle Database (continued)

The DBCA displays the progress of the various installation steps. When the database itself has been created, the DBCA displays essential information about the database. Make note of this information. The Database Control URL will be used in several of the following practice sessions.



Unlock the `HR` user with `HR` as password.

You completed your task to create a database and (optionally) a database template and database generation scripts.

Practice 4: Managing the Oracle Instance

Background: You have just installed the Oracle software and created a database. You want to ensure that you can start and stop the database and see the application data.

Your Tasks

1. Invoke Enterprise Manager with Mozilla and the `oracle` profile. Enter the URL that you wrote down in Practice 3, and log in as the `SYS` user with the `oracle` password and `SYSDBA` as "Connect As." Which port number does this database use? You noted this in Practice 3.

   ____

   The first time you log in after installing the software, the "Oracle Database 10*g* Licensing Information" page appears. Acknowledge this information.

2. View the initialization parameters (Navigation aid: Adminstration > All Initialization Parameters). Set the `JOB_QUEUE_PROCESSES` parameter to `15`. What SQL statement is run to do this?

   ____

3. *Question:* What is the significance of a check in the Dynamic column?

   ____

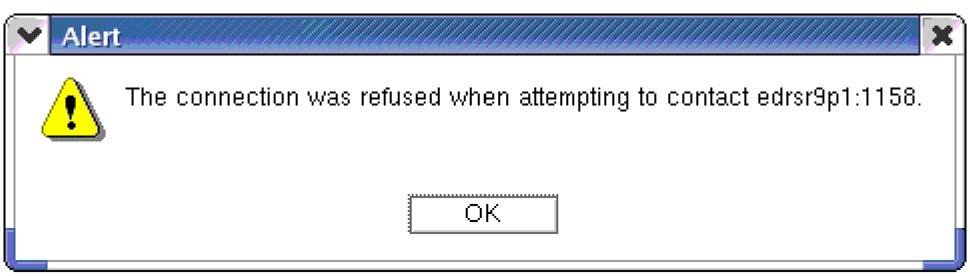
4. Shut down the database instance by using Enterprise Manager.

   For Host Credentials, enter `oracle` as Username and `oracle` as Password.

   Question: What SQL is executed to shut down the database instance?

   ____

   If you click Refresh during the shutdown operation, you may see the following error: Click **OK** and continue to refresh. The error will resolve itself.



   Click **OK** and continue to click Refresh. Note that the Status of the instance is now "Down."

Practice 4: Managing the Oracle Instance (continued)

5. When you note that the Status of the instance is “Down,” use SQL*Plus to verify that you are **not** able to connect as the `HR` user.

6. Use Enterprise Manager to restart the database instance with the `oracle` host credentials.
   Question: What SQL is run to accomplish the database startup?

   ____

7. In the alert log, view the phases that the database went through during startup. (Navigation aid: Database > Alert Log Content).
   Question: What are the database instance startup phases?

   ____

8. Test access to *i*SQL*Plus for your HR application developers. (Navigation aid: Database > iSQL*Plus). Use the `Normal` role, `hr` username and password, and the default setting as Connect Identifier. If there is an error accessing *i*SQL*Plus, then start the isqlplus process using the `isqlplusctl start` command at the OS prompt, and then reattempt. After connecting, select the contents of the `EMPLOYEES` table.

   When you have finished reviewing the information, log out of *i*SQL*Plus and close the *i*SQL*Plus window.

Practice 5: Managing Database Storage Structures

Background: You need to create a new tablespace for the `INVENTORY` application. You also need to create a database user that is not as privileged as the `SYS` user.

Your Tasks

1. Enter `./lab_05_01.sh` to run a script that creates the `DBA1` user. It is located at `/home/oracle/labs`. The password for `DBA1` is `oracle`. Leave the command shell window open. You will use it again later.

2. Use the Setup link in the top-right corner of Enterprise Manager (EM) to define the `DBA1` user as one who can perform administrative tasks in EM. Use the following settings:

Object	**Setting**
Name	`dba1`
Password	`Oracle`
Confirm Password	`Oracle`
Super Administrator	*Selected*

   When the non-SYS user is configured, log out as `SYS` user and log in as `DBA1` user. Use the `DBA1` user to perform the rest of these tasks, unless otherwise indicated.

3. Using Enterprise Manager, view information about the `EXAMPLE` tablespace (Navigation aid: Administration > Tablespaces). Answer the following questions about it:

   Question 1: What percentage of free space can be used up before the Warning threshold is reached?

   ____

   Question 2: How many segments are there in the `EXAMPLE` tablespace? (Navigation aid: “Show Tablespace Contents” Actions drop-down list selection).

   ____

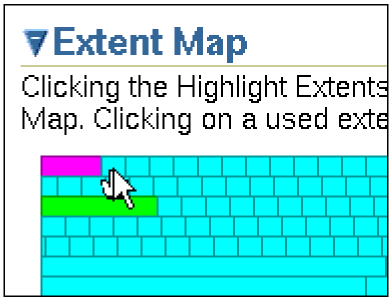
   Question 3: Which index in the `EXAMPLE` tablespace takes up the most space? (Navigation aid: Search type “INDEX”)

   ____

Practice 5: Managing Database Storage Structures (continued)

Question 4: Which segment is stored physically first in the tablespace? That is, which one is stored right after the tablespace header?

Look at all extent types on the Extent Map. Note the extent just to the right of the tablespace header extent.



Scroll to the top of the page again, and note the segment that is being pointed to.

____

4. Create a new, locally managed tablespace (LMT) called `INVENTORY` (Navigation aid: Administration > Tablespaces). Use the following specifications:

Object	**Setting**
Tablespace name	`INVENTORY`
Extent Management	Locally Managed
Type	Permanent
Status	Read Write
Use Bigfile tablespace	*deselected*
Data File Name	`inventory01.dbf`
File Size	`5 MB`
Extent Allocation	Automatic
Segment Space Management	Automatic
Enable Logging	*selected*

Review the SQL that will be run to create this tablespace.

Note: The `INVENTORY` tablespace will be used in later practice sessions during this class.

5. Run the `lab_05_05.sql` script to create and populate a table (called `X`) in the `INVENTORY` tablespace. What error do you eventually see?

____

Practice 5: Managing Database Storage Structures (continued)

6. Go to the Enterprise Manager window and define space for 50 MB in the tablespace instead of 5 MB, while keeping the same single data file in the tablespace. What is the `ALTER` statement that is executed to make this change?

   ____

7. Go back to the terminal window and run the `lab_05_07.sql` script that drops the table and reexecutes the original script that previously returned the space error.

   Note that the same number of row inserts are attempted, and there is no error because of the increased size of the tablespace.

8. In a terminal window, run the `lab_05_08.sql` script to clean up the tablespace for later practice sessions.

Practice 6: Administering User Security

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 must be able to log in to the `ORCL` database, select data from, and update records in the `HR.EMPLOYEES` 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 out after 15 minutes. You also need to create a new user account for the inventory application that you are installing.

Your Tasks

1. **Mandatory:** Review and run the `lab_06_01.sh` script (located in the `/home/oracle/labs` directory) to create the `INVENTORY` schema user, which you will use in the next practice.

2. Create a profile named `HRPROFILE`, allowing 15 minutes idle time (Navigation aid: Administration > Profiles). Optionally, review the underlying SQL statement.

3. Set the `RESOURCE_LIMIT` initialization parameter to TRUE, so that your profile limits will be enforced time (Navigation aid: Administration > All Initialization Parameters).

4. Use EM to create the role named `HRCLERK` that has permission to select from and update the `HR.EMPLOYEES` table (Navigation aid: Administration > Roles). Review your underlying SQL statement:

```
CREATE ROLE "HRCLERK" NOT IDENTIFIED
GRANT SELECT ON "HR"."EMPLOYEES" TO "HRCLERK"
GRANT UPDATE ON "HR"."EMPLOYEES" TO "HRCLERK"
```

5. Use EM to create the role named `HRMANAGER` that has permissions to insert into and delete from the `HR.EMPLOYEES` table. Grant the `HRCLERK` role to the `HRMANAGER` role. Review your underlying SQL statement:

```
CREATE ROLE "HRMANAGER" NOT IDENTIFIED
GRANT DELETE ON "HR"."EMPLOYEES" TO "HRMANAGER"
GRANT INSERT ON "HR"."EMPLOYEES" TO "HRMANAGER"
GRANT "HRCLERK" TO "HRMANAGER"
```

Practice 6: Administering User Security (continued)

6. Use EM to create an account for David Hamby, a new HR clerk (Navigation aid: Administration > Users). Use the following specifications:

Object	Setting
Name	DHAMBY
Profile	HRPROFILE
Password Authentication	*selected*
Password	newuser
Expire Password now	*selected*
Roles	CONNECT and HRCLERK

7. Use EM to create an account for Rachel Pandya, another new HR clerk. Repeat the steps from step 6 with RPANDYA as the username.

8. Use EM to create an account for Jenny Goodman, the new HR manager. Repeat the steps from step 6 with JGOODMAN as the username and selecting the HRMANAGER role instead of the HRCLERK role. Review your underlying SQL statement:

```
CREATE USER "JGOODMAN" PROFILE "HRPROFILE" IDENTIFIED BY "*******" PASSWORD
EXPIRE ACCOUNT UNLOCK
GRANT "CONNECT" TO "JGOODMAN"
GRANT "HRMANAGER" TO "JGOODMAN"
```

9. Test the new users in SQL*Plus. Connect to the ORCL database as the DHAMBY user. Use oracle as the new password. Select the row with EMPLOYEE_ID=197 from the HR.EMPLOYEES table. Then, attempt to delete it. (You should get the "insufficient privileges" error.)

10. Repeat the delete attempt as the JGOODMAN user. After deleting the row, issue a rollback, so that you still have the original 107 rows.

    Question 1: Where was the row stored after deletion?

    ____

    Question 2: When you created the new users, you did not select a default or temporary tablespace. What determines the tablespaces that the new users will use?

    ____

    Question 3: You did not grant the CREATE SESSION system privilege to any of the new users, but they can all connect to the database. Why?

    ____

11. Review the lab_05_01.sql script and the lab_05_01.txt log file that it generated when you created the DBA1 user.

Practice 6: Administering User Security (continued)

12. Use SQL*Plus to connect to the ORCL database as the RPANDYA user. Change the password to oracle. (You must change the password, because this is the first connection as RPANDYA.) Leave RPANDYA connected during the next lesson or at the end of the day. HRPROFILE specifies that users whose sessions are inactive for more than 15 minutes will automatically be logged out. Verify that the user was automatically logged out by trying to select from the HR.EMPLOYEES table again.

```
ERROR at line 1:
ORA-02396: exceeded maximum idle time, please connect again
```

Practice 7: Managing Schema Objects

Background: You need to create schema objects for the new inventory application. Work as DBA1 user in the SYSDBA role for your ORCL database.

Your Tasks

1. Return to the Enterprise Manager browser session, or invoke EM as the DBA1 user in the SYSDBA role for your ORCL database.

2. In the INVENTORY tablespace, create the PRODUCT_MASTER table in the INVENTORY schema. The specifications of the table are:

   PRODUCT_ID number(7). This is the primary key field. (Constraint name: PK_INV)
   PRODUCT_NAME varchar2(50) with a Not NULL constraint
   CODE varchar2(10) with a Not NULL constraint
   REORDER_THRESHOLD number(5) with a check constraint ensuring that the number is always greater than zero (Constraint name: CHK_REORDER)
   COST number(5,2)
   PRICE number(5,2)

3. In the INVENTORY tablespace, create the PRODUCT_ON_HAND table in the INVENTORY schema. You have been given the lab_07_03.sql script to create the table, but there is a problem with it (*intentionally created to enhance your learning experience*). Fix the problem, and run the script. If you cannot find the error right away, then go ahead and run the original script in SQL*Plus to see the error message. This helps you discover and solve the problem. The specifications of the table are:

   PRODUCT_ID number(7). 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)
   LAST_UPDATE date

4. In the INVENTORY tablespace, create the OBSOLETE_PRODUCTS table in the INVENTORY schema. This table definition is very much like that of the PRODUCT_MASTER table, so you can use Enterprise Manager's ability to "Define Using SQL" rather than using "Column Specification." The specifications of the table are:

   PRODUCT_ID number(7). 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)

Practice 7: Managing Schema Objects (continued)

5. In the INVENTORY tablespace, create an index called OBS_CODE on the CODE column of the OBSOLETE_PRODUCTS table in the INVENTORY schema. Choose an appropriate index type: either B-tree or Bitmap. Explain the reason for your choice.

   Question: Which type of index is appropriate, and why?

   ____

   ____

6. In the INVENTORY tablespace, create an index called PROD_NAME_CODE on the combined PRODUCT_NAME and CODE columns of the PRODUCT_MASTER table. Use the lab_07_06.sql script *(which contains an error, intentionally created to enhance your learning experience*). Correct the error and run the script. If you cannot find the error right away, then run the original script in SQL*Plus to see the error message. This will help you discover and solve the problem.

7. In the INVENTORY tablespace, use SQL*Plus to create a combined index on the PRODUCT_ID and QUANTITY columns of the PRODUCT_ON_HAND table. The index name should be POH_PROD_ID_QTY.

8. You receive an update for the inventory application that requires you to add two columns to the PRODUCT_MASTER table. Add a column called PRIMARY_SOURCE of the data type varchar2(50). Add another column called SECONDARY_SOURCE of the data type varchar2(50). What is the SQL that executes to do this?

   ____

   ____

9. You receive another update for the inventory application. This change request instructs you to drop the OBSOLETE_PRODUCTS table and add the OBSOLETED column to the PRODUCT_MASTER table, with data type DATE. Do this using EM. What clause is added to the end of the DROP TABLE statement to also remove the table constraints?

   ____

10. Another change request to the inventory application instructs you to create a view called WAREHOUSE_VW (Navigation aid: Administration > Views). The view is in the INVENTORY schema and displays (in this order):

    - The name of the product
    - The amount of "product on hand"
    - The warehouse city name

    You will have to join two tables together to create this view.

Practice 7: Managing Schema Objects (continued)

11. You receive a notice from developers that there is a certain type of query that will be run very frequently, and they want to be sure that it runs in less than one second. You need to run the query and see how it performs. First, run the `lab_07_11_a.sql` script to generate some test data of the volume indicated in the request. Then, run the query in the `lab_07_11_b.sql` script several times to see the average run time. Note that it takes several seconds to run each time. Create a function-based index on the `CODE` column that will improve the performance of this query.

12. Use *i*SQL*Plus to identify the data dictionary view name that you would use to list all constraints that the `INVENTORY` user can see. What is the view name?

    ____

13. How many indexes are owned by the `INVENTORY` user? You are looking for database objects owned by the `INVENTORY` user, so you know that the data dictionary view begins with the "`USER_`" prefix. The view name is `USER_INDEXES`.

    ____

Practice 8: Managing Data and Concurrency

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 could not do anything else. SQL script files are provided for you in the `/home/oracle/labs` directory.

Your Tasks

1. Make an uncommitted update to the row in question by running the `lab_08_01.sql` script. Do not worry if the session seems to “hang”—this is the condition you are trying to create.

2. Make an attempt to update the same row in a separate session by running, in a separate terminal window, the `lab_08_02.sql` script. Make sure you see the message “Update is being attempted now” before moving on.

3. Using the **Blocking Sessions** link on the Performance page, detect which session is causing the locking conflict.

4. What was the last SQL statement that the blocking session executed?

5. Resolve the conflict in favor of the user who complained, by killing the *blocking* session. What SQL statement resolves the conflict?

   ____

6. Return to the SQL*Plus command window, and note that `SMAVRIS`’s update has now completed successfully. It may take a few seconds for the success message to appear.

Practice 9: Managing Undo Data

Background: A new version of your application will include several reports based on very long-running queries. Configure your system to support these reports.

Your Tasks

1. Use the Undo Advisor to calculate the amount of undo space required to support a report that takes two days to run, on the basis of an analysis period of the last seven days. (Navigation aid: Administration > Advisor Central > Undo Management > Undo Advisor).

   Question: What does the analysis recommend as "Required Tablespace Size for New Undo Retention"?

   ____

2. Resize the undo tablespace to support the retention period required by the new reports (or 1 GB, whichever is smaller). Do this by increasing the size of the existing data file.

   Question: What are the two ways to add space to a tablespace?

   ____

Practice 10: Implementing Oracle Database Security

Background: You have just been informed of suspicious activities in the HR.JOBS table in your ORCL database. All maximum salaries seem to fluctuate in a strange way. You decide to enable standard database auditing and monitor data manipulation language (DML) activities in this table.

Log in as the DBA1 user (with oracle password, connect as SYSDBA) and perform the necessary tasks either through Enterprise Manager Database Control or through SQL*Plus. All scripts for this practice are in the /home/oracle/labs directory.

Your Tasks

1. Use Enterprise Manager to enable database auditing. Set the AUDIT_TRAIL parameter to XML. This setting should be stored in the SPFILE.

2. Because you changed a static parameter, you must restart the database. Do so by running the lab_10_02.sh script.

   Continue with the next step when you see that the database is restarted and the script has exited out of SQL*Plus.

3. Back in Enterprise Manager, select HR.JOBS as the audited object and DELETE, INSERT, and UPDATE as **Selected Statements**. Gather audit information by session. (Navigation aid: Administration > Audit Settings).

4. Provide input for the audit, by executing the lab_10_04.sh script. This script creates the AUDIT_USER user, connects to SQL*Plus as this user, and multiplies the values in the MAX_SALARY column by 10. Then, the HR user connects and divides the column values by 10. Finally, the AUDIT_USER user is dropped again.

5. In Enterprise Manager, review the audited objects. Use the **Audited Objects** tab.

6. Undo your audit settings for HR.JOBS, disable database auditing, and then restart the database by using the lab_10_06.sh script.

7. Maintain your audit trail: Because you are completely finished with this task, delete all audit files from the /u01/app/oracle/admin/orcl/adump directory.

Practice 11: Configuring the Oracle Network Environment

Background: Users need to connect to your `ORCL` database. Work with them to enable connections by using different methods. Ensure that users can use connect-time failover to take advantage of a backup listener.

Your Tasks

1. Make a copy of your `listener.ora` and `tnsnames.ora` files. They are in the `$ORACLE_HOME/network/admin` directory.

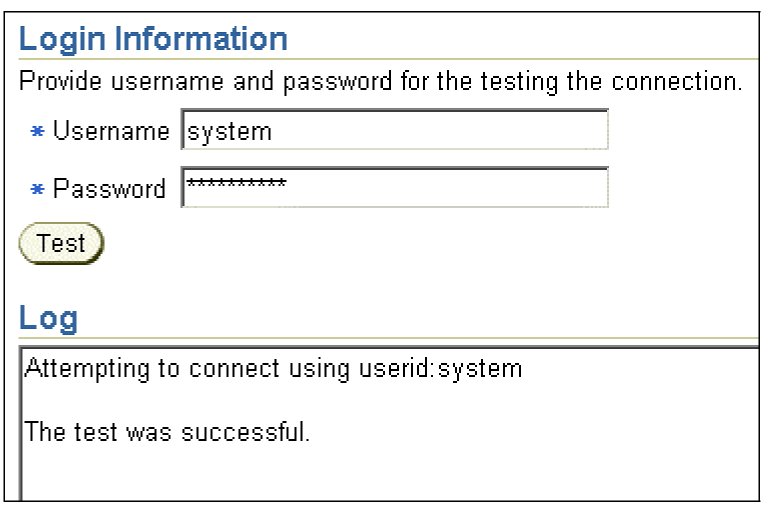
2. Navigate to the Net Services Administration page. Start by clicking the **Listener** link on the Database home page.

3. Modify your local Names Resolution file so that you can connect to another database. (Navigation aid: Listener link > Net Services Administration > Administer > Local Naming). Use the following information to define the connection:

Object	**Setting**
Connection name	`Testorcl`
DB ID Method	`SID`
SID value	`Orcl`
Protocol	TCP/IP
Port	`1521`
Host	IP or name of other student's computer

4. In Enterprise Manager, test access to your partner's `ORCL` database as the **`system`** user with the **`oracle`** password by using the **`testorcl`** Local Naming.

   The Processing page displays status information. Then, it is followed by a success message. *If you receive any errors or warnings, resolve them.*



Practice 11: Configuring the Oracle Network Environment (continued)

5. Test your changes to the network configuration by using SQL*Plus or *i*SQL*Plus. Again, use: `system/oracle@testorcl`. To see your partner's information, select the `instance_name` and `host_name` columns from the `v$instance` table. You should see your partner's host name.

6. Create a `LISTENER2` listener to support connect-time failover. Use port 1561 for this listener. First, log out of Enterprise Manager and run the `lab_11_06.sh` script to configure the `NetProperties` file. (Navigation aid: Listener link > Net Services Administration > Administer > Listeners). Use the **Static Database Registration** tab on the Create Listener page to connect the listener to your database. Use the following information:

Object	**Setting**
Listener name	`LISTENER2`
Host	`<your computer's name>`
Service name	`orcl`
Protocol	TCP/IP
Port	`1561`
SID	`orcl`
Oracle Home Directory	`/u01/app/oracle/product/10.2.0/db_1`

7. Start the `LISTENER2` listener.

Practice 12: Proactive Maintenance

Background: You want to proactively monitor your ORCL database so that common problems can be fixed before they affect users. This practice session invents some issues so that you can familiarize yourself with the tools that are available. First, execute scripts to set up your Automatic Database Diagnostic Management (ADDM) environment.

Your Tasks

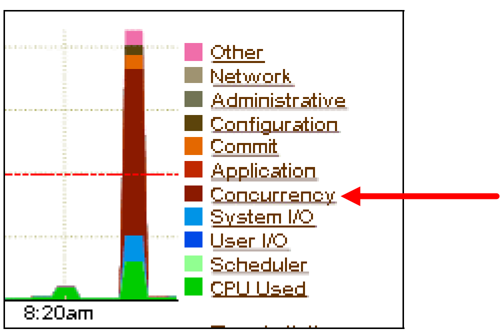
1. Create a new locally managed tablespace called TBSADDM. Its addm1.dbf data file is 50 MB. Ensure that the TBSADDM tablespace does not use Automatic Segment Space Management (ASSM). Execute the lab_12_01.sh script to perform these tasks.

2. Create a new ADDM user, identifed by ADDM. Assign the TBSADDM tablespace as default tablespace. Assign the TEMP tablespace as temporary tablespace. Grant the following roles to the ADDM user: CONNECT, RESOURCE, and DBA. Execute the lab_12_02.sh script to perform these tasks.

3. Use the DBMS_ADVISOR package to set the database activity time to 30 minutes. As an ADDM user, drop and create the ADDM table and gather statistics for this table. Create a snapshot in Automatic Workload Repository (AWR). Execute the lab_12_03.sh script to perform these tasks.

4. Create activity to be analyzed. Execute the lab_12_04.sh script to perform these tasks.

5. In Enterprise Manager, review the Performance page as a user connected as SYSDBA. View performance data in real time with a 15-seconds refresh cycle. After a while, you should see a spike on the “Average Active Sessions” graph. This is your activity to be analyzed. Looking at the graph, you can already determine that this instance is suffering from concurrency problems. If this is the first time that you accessed the Performance page, you need to accept the Adobe license agreement. Follow the directions in the pop-up window to accept the agreement.

   Note: Depending on when you run the workload, you may see differences between your graph and the one provided as a possible solution.

   After the spike is finished, execute the lab_12_05.sh script. This script forces the creation of a new snapshot and gathers statistics on your ADDM table.

Practice 12: Proactive Maintenance (continued)

6. Look at the **Performance Analysis** findings in order of their impact. There are several access paths to this information.

   Looking at the Performance Analysis section, you see that the first finding (in the SQL Tuning Recommendations category) has a 100% impact on the system. So your first impluse is to look at this finding in more detail. However, looking at this SQL statement does not yet help you to understand the concurrency problem of your database.

   Research the next finding under Schema Recommendations: **Read and write contention of database blocks was consuming significant database time**. Here, you are advised to use the Automatic Segment Space Management (ASSM) feature for your `ADDM` table.

7. To implement the recommendation, you must re-create the object. Create a new locally managed tablespace, called `TBSADDM2` with a 50 MB data file, called `addm2_1.dbf`. Ensure that the `TBSADDM2` tablespace uses the the ASSM feature. Then, execute the `lab_12_07.sh` script to drop the `ADDM` table, to re-create it in the new tablespace, to gather statistics and to take a new snapshot.

8. Execute your workload again by running the `lab_12_08.sh` script. (The `lab_12_08.sh` script is identical to the `lab_12_04.sh` script.)

9. In Enterprise Manager, review the Performance page as a user connected as `SYSDBA`. View performance data in real time with a 15-seconds refresh cycle. After a while, you should see a spike on the “Average Active Sessions” graph.

   Note: Depending on when you run the workload, you may see differences between your graph and the one provided as a possible solution.

   After the spike is finished, execute the `lab_12_09.sh` script. (The `lab_12_09.sh` script is identical to the `lab_12_05.sh` script.) This script forces the creation of a new snapshot and gathers statistics on your `ADDM` table.

10. Review the Performance Analysis on the Database home page. View the information for this last ADDM task.

    You see that there are no longer any schema-related recommendations. By moving the `ADDM` table to the locally managed `TBSADDM2` tablespace, which uses the Automatic Autoextend Segment feature, you obviously fixed the root cause of this problem.

11. To not affect other practice session, execute the `lab_12_11.sh` script to clean up your environment.

Practice 13: Performance Management

Background: Users are complaining about slower-than-normal performance for operations involving the human resources and order-entry applications. When you question other members of the DBA staff, you find that maintenance was recently performed on some of the tables belonging to the HR schema. You need to troubleshoot and make changes as appropriate to resolve the performance problems. SQL script files are provided for you in the /home/oracle/labs directory. Other directories are individually named.

Your Tasks

1. Log in to SQL*Plus as the DBA1 user and perform maintenance on tables in the HR schema by running the lab_13_01.sql script.

2. You get calls from HR application users saying that a particular query is taking longer than normal to execute. The query is in the lab_13_02.sql script. As the HR user, run it.

3. Using Enterprise Manager (EM), locate the HR session in which the above statement was just executed, and view the execution plan for that statement. (Navigation aid: Performance > Search Sessions).

4. Using EM, check to see the status of the EMPLOYEE table's index on EMPLOYEE_ID. See if it is VALID. (Navigation aid: Administration > Indexes).

5. Now that you have seen one index with a non-VALID status, you decide to check all indexes. Using SQL*Plus, as the HR user, find out which HR schema indexes do not have STATUS of VALID. To do this, you can query a data dictionary view with a condition on the STATUS column.

6. Using EM, reorganize all the indexes in the HR schema that are marked as UNUSABLE.

7. Return to the SQL*Plus session where the HR user is logged in, and run the lab_13_07.sql script to execute the same kind of query. Then, repeat the steps to see the plan of the last SQL statement executed by this session, to see if the plan has changed.

8. What is the difference in execution plans, and why?

   ____

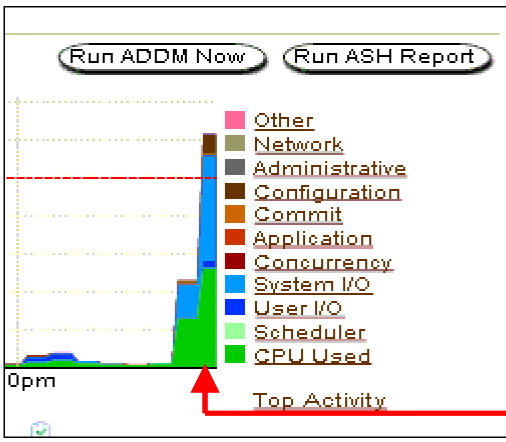
Practice 13: Performance Management (continued)

9. Simulate a working load on your instance by running the `lab_13_09.sql` script as the `DBA1` user. Note the SID value for task 10.

   This script takes about 20 minutes to complete. So, run it in a separate terminal window and continue with this practice exercise while it runs.

   Note: Because this script generates a fairly heavy load in terms of CPU and disk I/O, you will notice that response time for Database Control is slower.

   Use EM to view overall instance performance and answer the following questions:



   Wait to see the beginning of a spike in the **Average Active Sessions** graph before proceeding.

   Question 1: In the **Average Active Sessions** graph, which are the two main categories that active sessions are waiting for?

   Question 2: In the Configuration category of waits, what is one of the contributors to the wait time? Click **Configuration** to see the graph.

   Question 3: Review the **Physical Writes** on the **Instance Disk I/O** graph. Determine which process is doing the most writing to the disk.

   Question 4: Review the **Top Activity** under **Additional Monitoring Links**. Which SQL statement is causing the most waits?

   ---

10. Kill the session that is generating the load. Use the SID value from step 9 in Enterprise Manager's **Search Sessions** link on the **Performance** tab to locate and kill the session.

Practice 14: Backup and Recovery Concepts

Background: Your ORCL database is ready to move from test or development into production. Configure your database to reduce the chances of failure or data loss.

Your Tasks

1. Verify that you have two control files to ensure redundancy. (Navigation aid: Administration > Control Files).

   Question 1: How would you add another control file if you needed to?

   ____

   ____

   ____

   ____

   ____

2. Check how many members each redo log group has. Ensure that there are at least two redo log members in each group. In what directory or directories are the redo log files stored?

   ____

   ____

   Note: In a production database, you want to ensure that the two members are on different hard drives, preferably with different disk controllers, to minimize the risk of any single hardware failure destroying an entire log group.

3. You notice that, for each log group, the `Archived` column has a value of `No`. This means that your database is not retaining copies of redo logs to use for database recovery, and in the event of a failure, you will lose all data since your last backup. Place your database in `ARCHIVELOG` mode, so that redo logs will be archived.

   In the OS, create a new directory `/u01/app/oracle/archive` as the destination for the redo log files. In Enterprise Manager, verify that Log Archive Filename Format contains `%t`, `%s`, and `%r`.

   Notice that the database is preconfigured to save archived logs to the Flash Recovery Area by default (Archive Log Destination 10).

4. Configure redundant archive log destinations—one to the Flash Recovery Area and the other to `/u01/app/oracle/archive/`. Do not forget the trailing slash. Leave **Quota** blank.

   Optionally, review the underlying SQL statement:

   Show SQL

   ```
   ALTER SYSTEM SET log_archive_dest_1 =
   "LOCATION=/u01/app/oracle/archive/ OPTIONAL REOPEN=300" SCOPE=BOTH
   ALTER SYSTEM SET log_archive_dest_10 =
   "LOCATION=USE_DB_RECOVERY_FILE_DEST OPTIONAL REOPEN=300" SCOPE=BOTH
   ```

   When prompted, restart your database with `oracle` Host Credentials and the `SYS` user.

   Now that your database is in `ARCHIVELOG` mode, it will continually archive a copy of each online redo log file before reusing it for additional redo data.

   Note: Remember that this consumes space on the disk and that you must regularly back up older archive logs to some other storage.

Practice 15: Performing Database Backups

Background: Your database is ready to move from development and test into production. Ensure that your database is configured so that recovery is possible without loss of data.

Your Tasks

1. What is the difference between a backup set and an image copy?

   ____

2. What is the destination of any disk backups that are done?

   ____

3. Test making a backup to disk, as a backup set, with `oracle` for Host Credentials. (Navigation aid: Maintenance > Backup Settings)

4. Back up your entire database, without archived logs, while the database is open for user activity. This backup should be the base for an incremental backup strategy (Navigation aid: Maintenance > Schedule Backup).

   Question: What prerequisite *must* be met to create a valid backup of a database without shutting it down?

   ____

   Use the following specifications to perform the backup:

Object	**Setting**
Object(s)	Whole Database
Host Credentials Username	`oracle`
Host Credentials Password	`oracle`
Backup Type	Full Backup
Use as the base of an incremental backup strategy	*selected*
Backup Mode	Online Backup
Backup Location	Disk
Disk Backup Location	Flash Recovery Area
Also backup all archived logs on disk	*deselected*
Schedule	*Defaults selected*

   Submit and monitor the job. This backup takes approximately 15 minutes to complete.

Practice 15: Performing Database Backups (continued)

5. Schedule nightly disk-based incremental online backups for your whole database, without archived logs `backup` (Navigation aid: Maintenance > Schedule Backup). Schedule it for 11:00 p.m.. The schedule should be in effect indefinitely.

   Use the following specifications:

Object	**Setting**
Object(s)	Whole Database
Host Credentials Username	`oracle`
Host Credentials Password	`oracle`
Backup Type	Incremental Backup (Level 1)
Use as the base of an incremental backup strategy	*selected*
Backup Mode	Online Backup
Also backup all archived logs on disk	*deselected*
Backup Location	Disk – Flash Recovery Area
Job Name	`Nightly_Backup`
Job Description	*Default selected*
Start	Later
Time	11:00 p.m.
Repeat	Interval
Frequency	1 Days

Practice 16: Performing 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. SQL script files are provided for you in the `/home/oracle/labs` directory. If needed, use appendix C for Linux and appendix D for SQL syntax. Note that where OS file names are mentioned, your system may possibly have different file names than shown here.

Your Tasks

1. Recover from the loss of a control file.

   a) As the `DBA1` user, run the `lab_16_01_a.sql` script to prepare some procedures to be called by the rest of this practice.

   b) Now run the `lab_16_01_b.sql` script. 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. Use Enterprise Manager's Database page to attempt to start up the database, and use SQL*Plus, if needed.

   d) The startup of the instance fails with Enterprise Manager, and you can get no other information to explain the failure. So use the command-line tools.

      - Connect to the instance with SQL*Plus `as sysdba` and check the current status of the instance.

      - Attempt to mount the database.

   e) The instance cannot move to the mount stage because it cannot find one of the control files. Check the last 10 rows of the alert log to see which control file is the problem.

   f) The control file in the Flash Recovery Area is missing. Restore the missing control file by copying the existing control file, and then mount and open the database.

   Question 1: Why did you have to use two commands to move the instance state from NOMOUNT to OPEN?

   ____

   ____

   Question 2:Why did you use operating system commands to restore the control file instead of using Oracle Recovery Manager?

   ____

   ____

Practice 16: Performing Database Recovery (continued)

2. Recover from the loss of an application data file.

   a) Start a SQL*Plus session as the `DBA1` user, and run the `lab_16_02.sql` script. This script deletes one of your application data files.

   b) The Help desk has received a call from a user who is unable to access the `COUNTRIES` table in the `HR` application schema. Count the rows in the table to see whether there is a problem.

   c) Troubleshoot and recover as necessary. The error message suggests that the data file for the `EXAMPLES` tablespace is corrupt or missing. Using operating system commands, verify that there is a problem with the file.

   d) Recover the data file to the current time, specifying the missing data file to be recovered. (Navigation aid: Maintenance > Perform recovery > Object Type > Datafiles).

   e) Verify that the `COUNTRIES` table is now accessible.

3. Recover from the loss of a system data file.

   a) *Question 3*: Why is recovery from the loss of a system data file or a data file belonging to an undo tablespace different from recovering an application data file?

   ____

   ____

   b) As `SYSDBA`, run the `lab_16_03.sql` script. This script deletes the system data file.

   c) In Enterprise Manager, review the Database home page. The database is shut down, so you click **Startup** to try to open it.

      - Enter the host credentials as `oracle` and `oracle` for the host **Username** and **Password**, and enter `DBA1` and `oracle` for the database credentials, and then click **OK**.

   d) This command will fail with the database left in the `MOUNT` state, because there is a data file missing from the `SYSTEM` tablespace.

   e) Recover the missing data file.

   f) Open the database.

   g) Verify that the database is open and operating normally, by logging into EM as `DBA1/oracle, as SYSDBA`, and reviewing the Database home page.

Practice 17: Performing Flashback

Background: You decide to gain hands-on experience in some of the flashback functionality. To avoid impacting other users, you will first copy the DEPARTMENTS table of the HR schema to DEPARTMENTS2.

Your Tasks

1. Log in to SQL*Plus as DBA1 user and create a new HR.DEPARTMENTS2 table based on the HR.DEPARTMENTS table. Count the rows in the DEPARTMENTS2 table. There should be 27 rows.

2. Drop the HR.DEPARTMENTS2 table, and then verify that it has indeed been dropped.

3. Use the FLASHBACK TABLE command to restore the table. Count the rows in the DEPARTMENTS2 table.

4. Run the lab_17_04.sql script to insert three rows into the HR.DEPARTMENTS2 table by using three separate transactions. The new rows have DEPARTMENT_ID values of 280, 290, and 300.

5. Use Enterprise Manager to perform flashback to the version of the table where only the first of the three new rows is present (with DEPARTMENT_ID = 280). (Navigation aid: Maintenance > Perform Recovery). First, evaluate row changes to decide on a point in time. Limit your analysis to the new rows just added: where department_id >= 280. If you receive an error while performing the flashback, you may need to enable row movement on the table. See the next step. Use the following specifications:

Object	**Setting**
Object Type	Tables
Operation Type	Flashback Existing Tables
Evaluate row changes and transactions to decide on a point in time	*selected*
Table	HR.DEPARTMENTS2
Choose Columns > Selected Columns	DEPARTMENT_ID
Bind The Row Value	where department_id >= 280

Review the Flashback Versions Query Result.

Flashback Versions Query Result

Select	Flashback SCN	Flashback Timestamp	Transaction ID	Operation	DEPARTMENT_ID
(•)	6800080	Jun 9, 2005 5:15:32 PM	0A002B00A6030000	INSERT	300
( )	6800061	Jun 9, 2005 5:15:15 PM	03002400CB030000	INSERT	290
( )	6800049	Jun 9, 2005 5:14:53 PM	05000600FA030000	INSERT	280

Select the row with the DEPARTMENT_ID = 290 and continue your flashback operation. Review the SQL statement that you are about to execute and submit your operation.

Practice 17: Performing Flashback (continued)

6. You find that the operation fails because row movement is not enabled for the table. You may recall from the lesson that row movement must be enabled for this feature to work.

   Enable row movement (Navigation aid: Administration > Tables). Review the underlying SQL statement:

```
ALTER TABLE "HR"."DEPARTMENTS2" ENABLE ROW MOVEMENT
```

   After the table modification is confirmed, you can perform the flashback operation.

   Repeat step 5 to perform the version flashback of the table. Because row movement has been enabled, it should succeed this time.

   Then, in SQL*Plus, count the rows of the `HR.DEPARTMENTS2` table to confirm the flashback operation. Note that there is only one additional row now, not three. Then, display the rows where `DEPARTMENT_ID >= 280`. Note that only one of the original three is remaining.

```
SQL> select count(*) from hr.departments2;

  COUNT(*)
----------
        28

SQL> select * from hr.departments2 where department_id >= 280;

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
          280 DUMMY1

SQL>
```

Practice 18: Moving Data

Background: In the recent past, you received a number of questions about the `HR` schema. To analyze them, without interfering in the daily activities, you decide to use the Data Pump Wizard to export the `HR` schema to file. When you perform the export, you are not sure into which database you will be importing this schema.

In the end, you find out that the only database for which management approves an import, is the `ORCL` database. So, you perform the import with the Data Pump Wizard, remapping the `HR` schema to a newly created `HR_TEST` schema in the `HR_TEST` tablespace. To follow best practice guidelines, you also decide to create a `DP` user who will be a DBA performing Data Pump jobs. For your convenience in class, the creation of the `DP` user is included in the `lab_18_01.sql` script. SQL scripts are in the `/home/oracle/labs` directory. However, there is one step missing.

Then you receive two data load requests for which you decide to use SQL*Loader.

Your Tasks

1. Review the `lab_18_01.sql` script, which creates the `HR_TEST` tablespace, the `HR_TEST_ROLE` role, and the `HR_TEST` and `DP` users. Note the passwords for these users.

   `HR_TEST` password: ________________________

   `DP` password: ________________________

   Which additional step do you need to perform to allow the `DP` user access to Enterprise Manager as Administrator?

   ____

2. Execute the `lab_18_02.sh` script. Then, perform the required step to make the `DP` user an EM administrator.

Practice 18: Moving Data (continued)

3. Log in to Enterprise Manager as the DP user in the Normal role and export the HR schema (Navigation aid: Maintenance > Export to Export Files). Use the following specifications:

Object	Setting
Username	oracle
Password	oracle
Save as Preferred Credential	*selected*
Export: Schemas	HR
Export Options: Directory Objects	DATA_PUMP_DIR
Log File	hrexp.log
Export Files: Directory Objects	DATA_PUMP_DIR
File Name	HREXP%U.DMP
Export Schedule: Job Name	hrexp
Description	Export HR schema
Job Start Time	Immediate

   Review the PL/SQL that the Export Wizard helped you to create and submit the job. A processing message appears, and then a success message. If not, resolve any errors, which may have occurred.

   Note: Please wait, not only for the job to be created, but also for the job to complete execution. (It may take two minutes.)

4. As the DP user, import the exported HR schema back into the ORCL database, remapping it to the previously created HR_TEST schema. (Navigation aid: Maintenance > Import from Export Files). Use the following import specifications:

Object	Setting
Import Files: Directory Objects	DATA_PUMP_DIR
File Name	HREXP%U.DMP
Import Type	Schemas
Host credentials	oracle

   The Data Pump Wizard reads the specified import file and gives you a success message. If not, resolve any errors that may have occurred.

Practice 18: Moving Data (continued)

Use the following import remapping specifications:

Object	**Setting**
Re-Map Schemas: Source Schema	HR
Re-Map Schemas: Destination Schema	HR_TEST
Re-Map Tablespaces: Destination Tablespace	HR_TEST
Import Options: Directory Objects	DATA_PUMP_DIR
Log file	hrimport.log
Import Schedule: Job Name	hrimp
Description	Import HR schema for test purposes
Start	Later
Time	Enter a time between 2 and 5 minutes from now (to give yourself time for submitting and monitoring the job)

(Navigation aid: Maintenance > Monitor Export and Import Jobs) Monitor the job while it is executing.

Optional, as this requires quick action: Suspend and resume the job.

In the end, you want to see that your job executed 100% without any errors.

Verify the import succeeded by viewing the log file.

```
$ cat </u01/app/oracle/product/10.2.0/db_1/rdbms/log/hrimport.log
```

Note: You may see an error saying that the hr_test object already exists. This is because that user existed when you did the export, and still exists. This is not a problem.

Select data from tables in the `hr_test` schema, for verification of the import.

Practice 18: Moving Data (continued)

5. As the DP user, load data into the PRODUCT_MASTER table by using SQL*Loader via Enterprise Manager Database Control. (Navigation aid: Maintenance > Load Data from User Files). Use the following loading specifications:

Object	**Setting**
Use Existing Control File	*selected*
Username	oracle
Password	oracle
Save as Preferred Credential	*selected*
Load Data Control File Name	/home/oracle/labs/lab_18_05.ctl
Data File: Provide the full path and name on the database server machine	*selected*
Data File Name	/home/oracle/labs/lab_18_05.dat
Load Method	Conventional Path
Load Data Options: Log File Name	/home/oracle/labs/lab_18_05.log
Load Data Schedule: Job Name	lab_18_05
Description	Load data into the PRODUCT_MASTER table
Start	Immediately

   Submit your job and confirm your results by viewing your lab_18_05.log file.

6. As the INVENTORY user, load data into the PRODUCT_ON_HAND table by using SQL*Loader command line. (Enter the command in continuation, without pressing [Enter] before reaching the end of the command: **sqlldr userid=inventory/verysecure control=lab_18_06.ctl log=lab-_18_06.log data=lab_18_06.dat**).

   Confirm your results by viewing your lab_18_06.log file in your /home/oracle/labs directory.

Appendix B
Solutions

Solutions for Practice 2: Installing the Oracle Database Software

Background: In the practices of this course, you assume the role of a database administrator (DBA). The operating system (OS) accounts on your computer 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 installation, and the installation media is staged at `/stage/Disk1`. Perform the following tasks as the default `oracle` OS user, unless otherwise indicated.

After installing the software, execute the `lab_02_05.sh` script, which creates a listener for you.

Note: Completing this practice is critical for all following practice sessions.

1. Install the Oracle database software as the `oracle` user. Navigate to the `/stage/Disk1` directory, and start the Oracle Universal Installer (OUI) by entering `./runInstaller`.

   a) Right-click your desktop and select **Open Terminal**, and then enter:

   ```
   $ cd /stage/Disk1
   $ ./runInstaller
   ```

2. Select your installation method for OUI.

   a) On the Installation Method page, select **Basic Installation**, and confirm the following settings:

Object	Setting
Database Home Location	`/u01/app/oracle/product/10.2.0/db_1`
Installation Type	Enterprise Edition
UNIX DBA Group	`oinstall`
Create Starter Database	*Deselected*

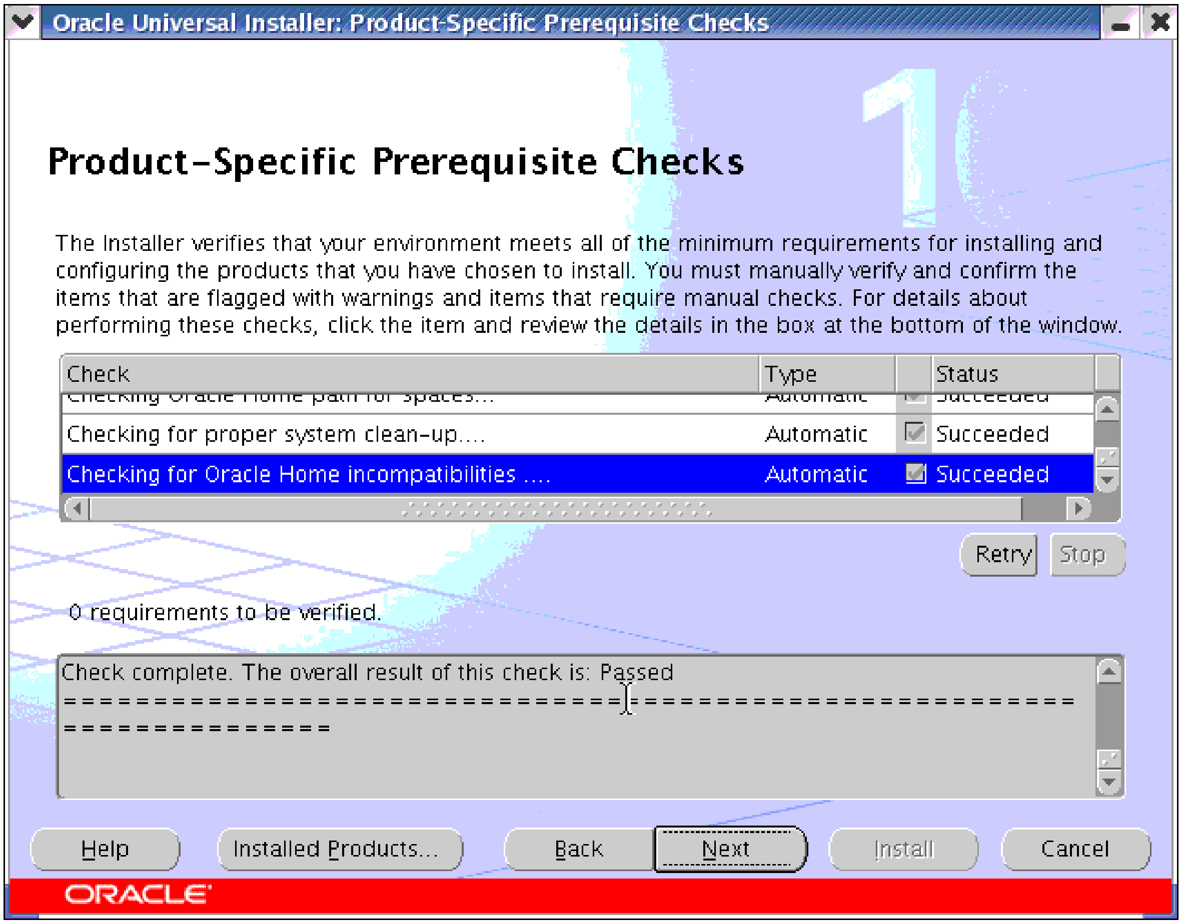
   Note: Ensure that you deselect the Create Starter Database option.

   b) Click Next.

   c) You are now on the page titled “Specify Inventory directory and credentials.” Accept `/u01/app/oracle/oraInventory` as **inventory directory** and `oinstall` as **Operating System group name**. Click **Next**.

Solutions for Practice 2: Installing the Oracle Database Software (continued)

OUI is loading the products list and checking product-specific prerequisites.



d) After OUI has finished its prerequisite checks on the Product-Specific Prerequisite Checks page, click **Next**.

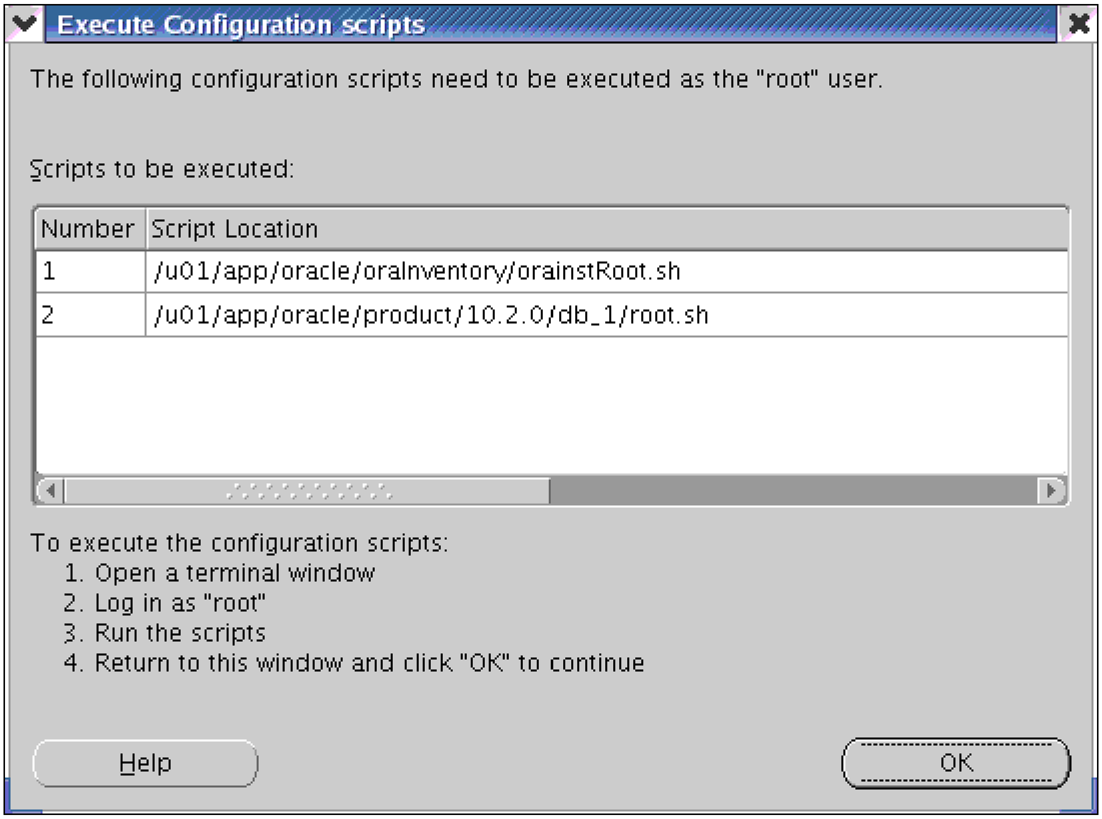
3. When the prerequisite checks are finished, the Summary page is displayed.

   a) Click **Install** to begin your installation.

   Estimated installation time is 10–15 minutes. However, varying environments can greatly influence this estimate.

4. When the “Execute Configuration scripts” page appears, follow the instructions on that page, accept the default for the local `bin` directory, and then finish your installation with OUI.

Solutions for Practice 2: Installing the Oracle Database Software (continued)



a) Right-click your desktop and select **Open Terminal**.

b) Run the `orainstRoot.sh` and `root.sh` scripts as `root`, and then click **Continue**.

```
$ su
# password: oracle <root password, does not appear on the screen>
# cd /u01/app/oracle/oraInventory
# ./orainstRoot.sh
# cd /u01/app/oracle/product/10.2.0/db_1
# ./root.sh
```

c) Accept the default for the local `bin` directory.

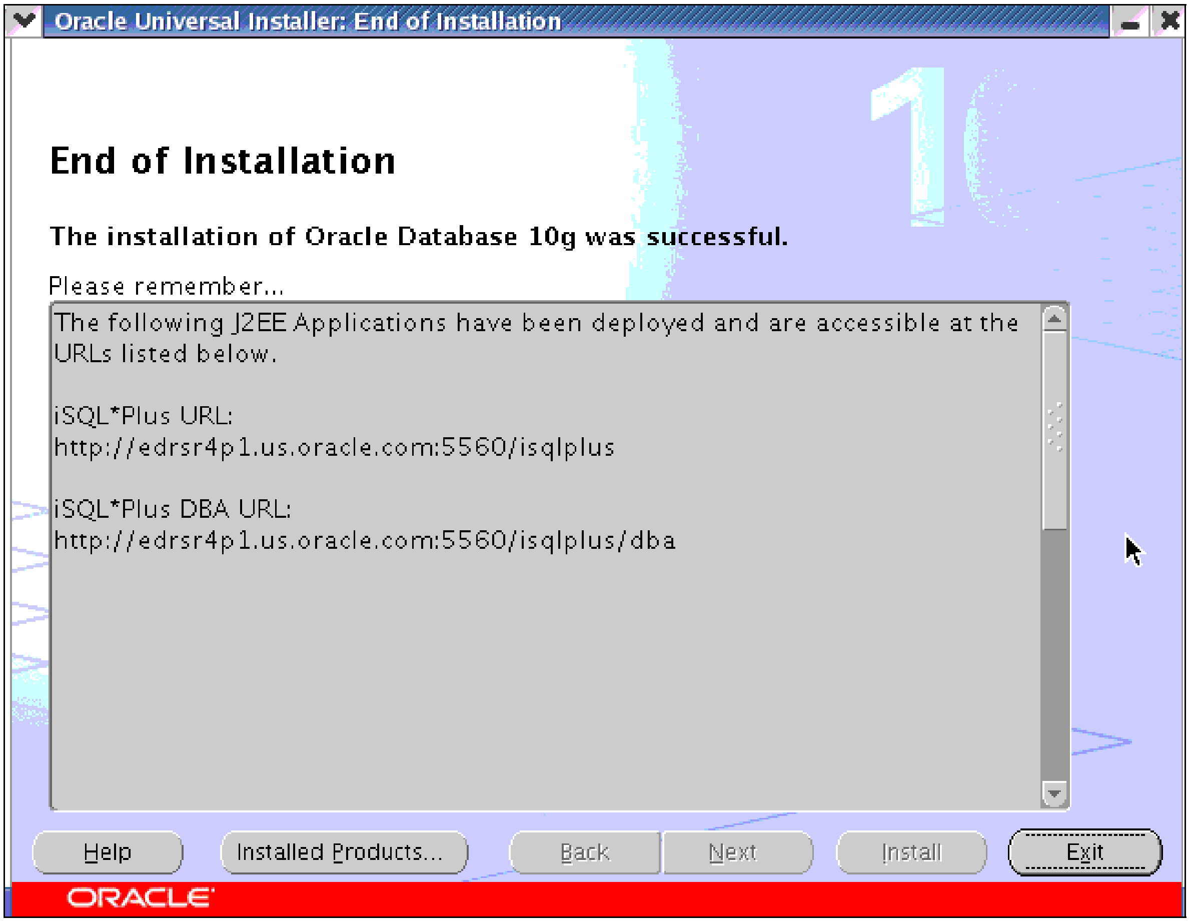
Solutions for Practice 2: Installing the Oracle Database Software (continued)

```
[oracle@EDRSR4P1 oracle]$ su
Password:
[root@EDRSR4P1 oracle]# cd /u01/app/oracle/oraInventory
[root@EDRSR4P1 oraInventory]# ./orainstRoot.sh
Changing permissions of /u01/app/oracle/oraInventory to 770.
Changing groupname of /u01/app/oracle/oraInventory to oinstall.
The execution of the script is complete
[root@EDRSR4P1 oraInventory]# cd /u01/app/oracle/product/10.2.0/db_1
[root@EDRSR4P1 db_1]# ./root.sh
Running Oracle10 root.sh script...

The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /u01/app/oracle/product/10.2.0/db_1

Enter the full pathname of the local bin directory: [/usr/local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying coraenv to /usr/local/bin ...


Creating /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
[root@EDRSR4P1 db_1]# 
```

d) Enter **`exit`** to exit the `root` OS user.

e) Close the terminal window, and then click **OK** on the “Execute Configuration scripts” page.

The End of Installation page appears.



f) Make a note of your URLs. You will use them in later practice sessions.

*i*SQL*Plus URL:____

*i*SQL*Plus DBA URL:____

g) Optionally, click **Installed Products**, review the product inventory, then click **Close**.

h) Click **Exit**, and then click **Yes** to leave OUI.

5. Execute the `lab_02_05.sh` script, which is in the `/home/oracle/labs` directory. This script uses the `netca` utility to create a listener for you. If you create your first database with OUI (as part of your installation), then OUI invokes the network configuration assistant, which creates the first listener for you.

a) In a terminal window, enter:

```
cd /home/oracle/labs
./lab_02_05.sh
```

You see an activity log, which should end with a success message. If not, resolve any errors that might have occurred.

Solutions for Practice 3: Creating an Oracle Database

Background: You are about to begin creating your first Oracle database. You anticipate that several similar databases will be needed in the near future. Therefore, you decide to create your `ORCL` database, as well as a database template and the database creation scripts. Locate the scripts in the `/home/oracle/labs` directory (which is the directory that you use most often throughout this course).

After you create the `ORCL` database, you execute the `lab_03_03.sh` script, which creates a listener for you.

Note: Completing the database creation is critical for all following practice sessions.

1. Start the Database Configuration Assistant (DBCA).

   a) Open a terminal window as the `oracle` user; that is, right-click your desktop and select **Open Terminal**.

   b) To start the DBCA, enter:

```
$ dbca
```

2. Begin the `ORCL` database creation. Use the General Purpose database template.

   a) In the DBCA, click **Next** on the Welcome page.

   b) On the Operations page, select **Create a Database,** and then click **Next**.

   c) On the Database Templates page, select **General Purpose,** and then click **Show Details**.

   d) Review the template's details and answer the following questions.

   Question 1: How many control files are created?

   Answer: 3

   Question 2: Would it maximize database availability to multiplex them?

   Answer: Yes. (*This will be done in a later practice.)*

   Question 3: How many redo log groups are created?

   Answer: 3

   Question 4: Would it maximize database availability to mirror them?

   It depends: No, not in class, because there are already three groups and you have only one physical storage device; but yes, if you can put each group on a different physical storage device.

Solutions for Practice 3: Creating an Oracle Database (continued)

Question 5: What is the database block size (`db_block_size`)?

Answer: 8 KB

Question 6: What is the value of Sample Schemas?

Answer: Sample Schemas is set to False.

Note: You will change this setting later in this practice to create the `HR` sample schema.

Question 7: What is the template default for the Database Character Set?

Answer: WE8ISO8859P1

Note: You will change this setting later in this practice to use a Unicode database character set.

e) Click **Close** to close the Template Details window.

f) Click **Next**.

3. Create the `ORCL` database, as well as the `ORCL` template and the database generation scripts.

a) On the Database Identification page, enter **`orcl.oracle.com`** as Global Database Name. The SID defaults to the database name **`orcl`**. Click **Next**.

b) On the Management Options page, ensure that the following items are selected:

- Configure the Database with Enterprise Manager
- Use Database Control for Database Management

c) Click **Next**.

d) On the Database Credentials page, select **Use the Same Password for All Accounts** and enter **`oracle`** as Password and Confirm Password. Then, click **Next**.

e) On the Storage Options page, select **File System**, and then click **Next**.

f) On the Database File Locations page, select **Use Oracle-Managed Files**. Accept the default Database Area, and then click **Next**.

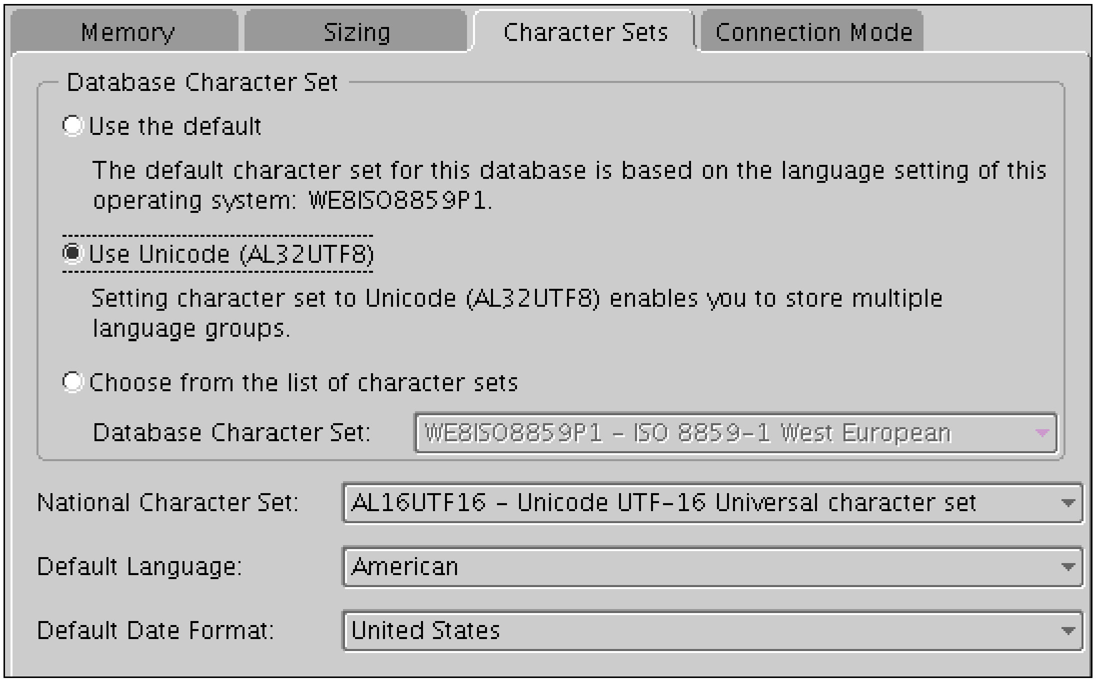
g) On the Recovery Configuration page, select **Specify Flash Recovery Area,** and then click **Next**.

h) On the Database Content page, select **Sample Schemas,** and then click **Next**.

i) On the Memory tabbed page of the Initialization Parameters page, select **Custom** and then select **Automatic** for the **Shared Memory Management** setting.

Solutions for Practice 3: Creating an Oracle Database (continued)

j) On the Initialization Parameters Character Sets tabbed, select **Use Unicode (AL32UTF8)**.

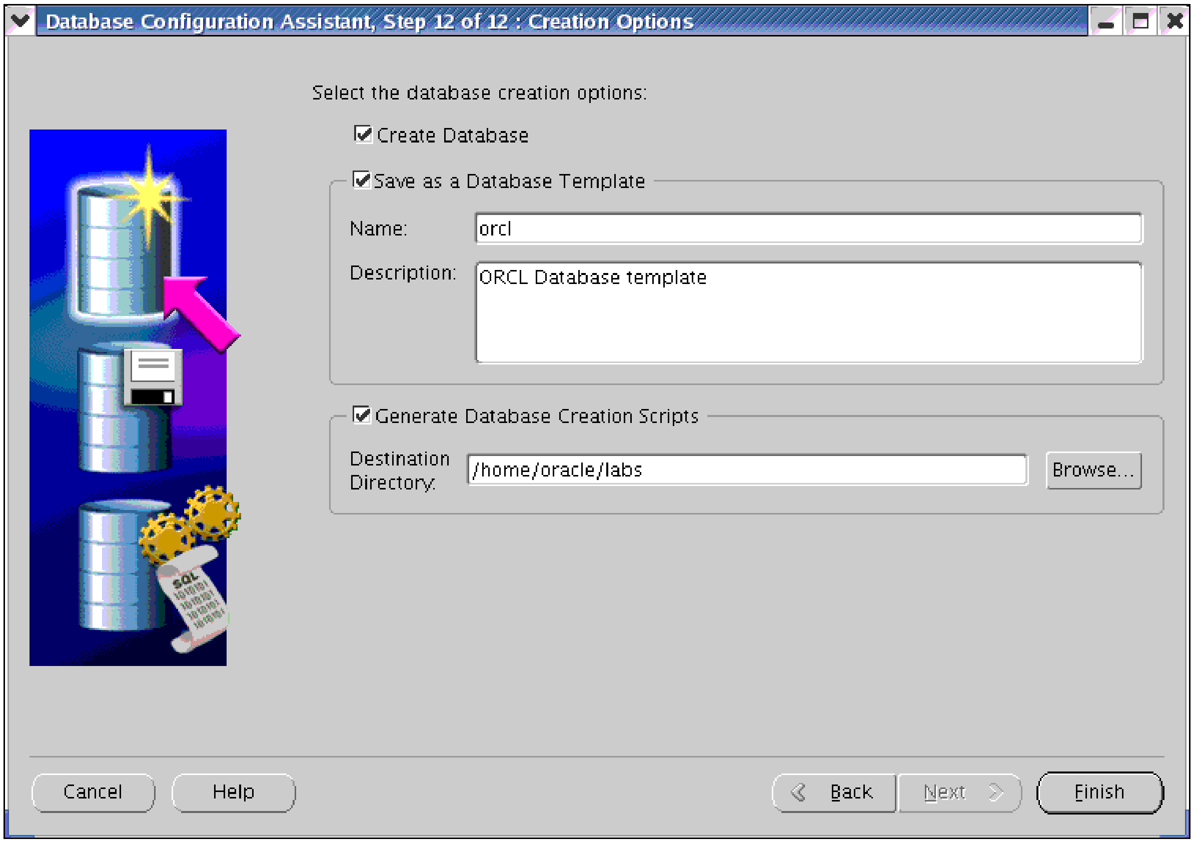


k) Review the Sizing and Connection Mode tabbed pages, but do not change any values. Then, click **Next**.

l) On the Database Storage page, review your file names and locations. Then, click **Next**.

Solutions for Practice 3: Creating an Oracle Database (continued)

m) On the Creation Options page, select **Create Database**.

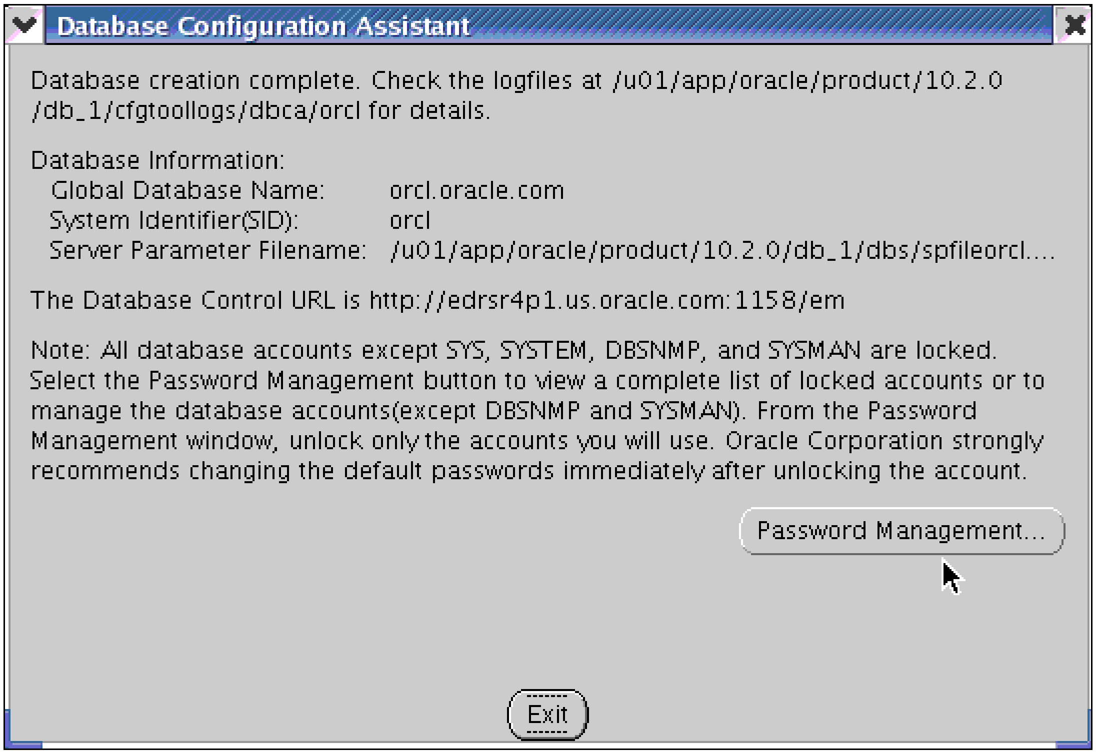


n) Optionally, select all creation options and enter `orcl` as **Name** for the database template, `ORCL Database template` as **Description**, and `/home/oracle/labs` as **Destination Directory**. Then, click Finish.

o) The Confirmation page appears. Review options and parameters, such as `Sample Schemas (true)`, `db_block_size (8KB)`, `sga_target (270MB)`, `undo_management (AUTO)`, `Database Character Set (AL32UTF8)`, and then click **OK**.

p) Click **OK** to acknowledge that the template has been created. Then, acknowledge the generation of the database scripts (if you selected those options).

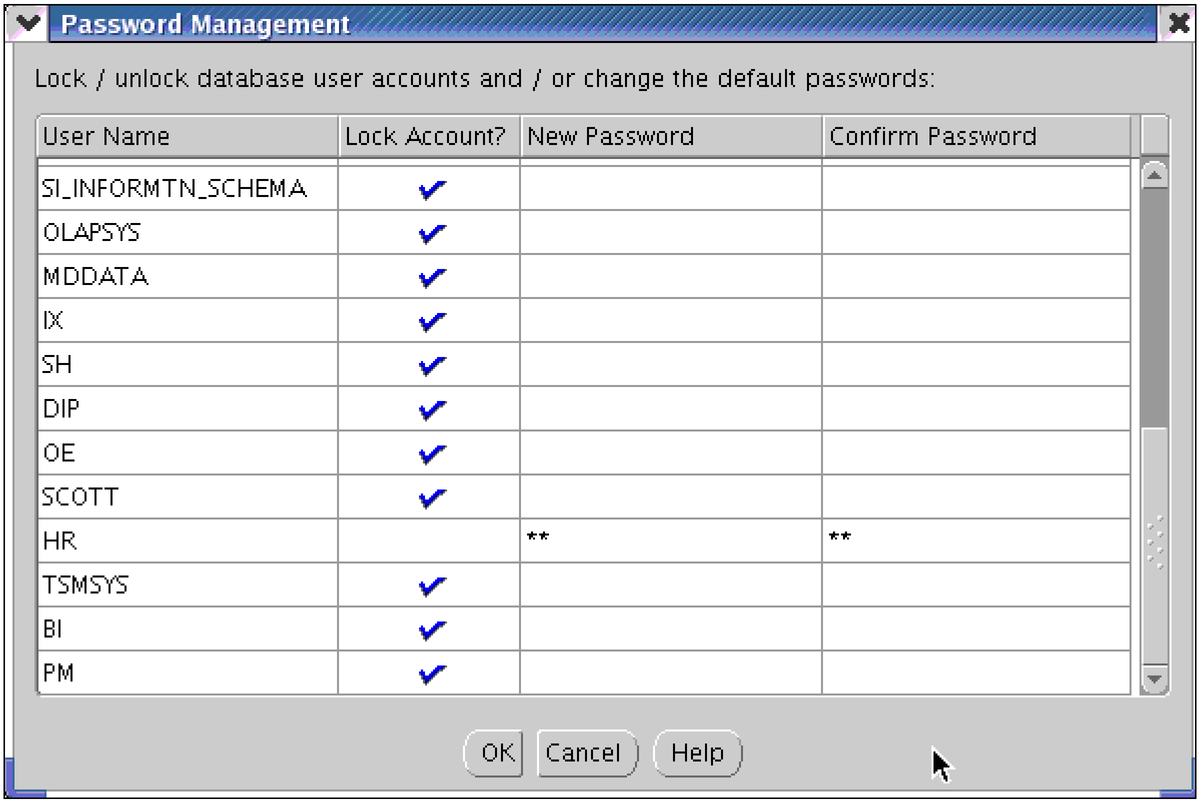
q) The DBCA displays the progress of the various installation steps. When the database itself has been created, the DBCA displays essential information about the database. Make note of this information. The Database Control URL will be used in several of the following practice sessions.



r) Click the **Password Management** button.

s) Scroll down the Password Management page until you see the HR User Name.

t) Deselect **Lock Account?** and enter **`hr`** as New Password and Confirm Password. Then, click **OK**.



u) Click **Exit** to close the DBCA.

You completed your task to create a database and (optionally) a database template and database generation scripts.

Solutions for Practice 4: Managing the Oracle Instance

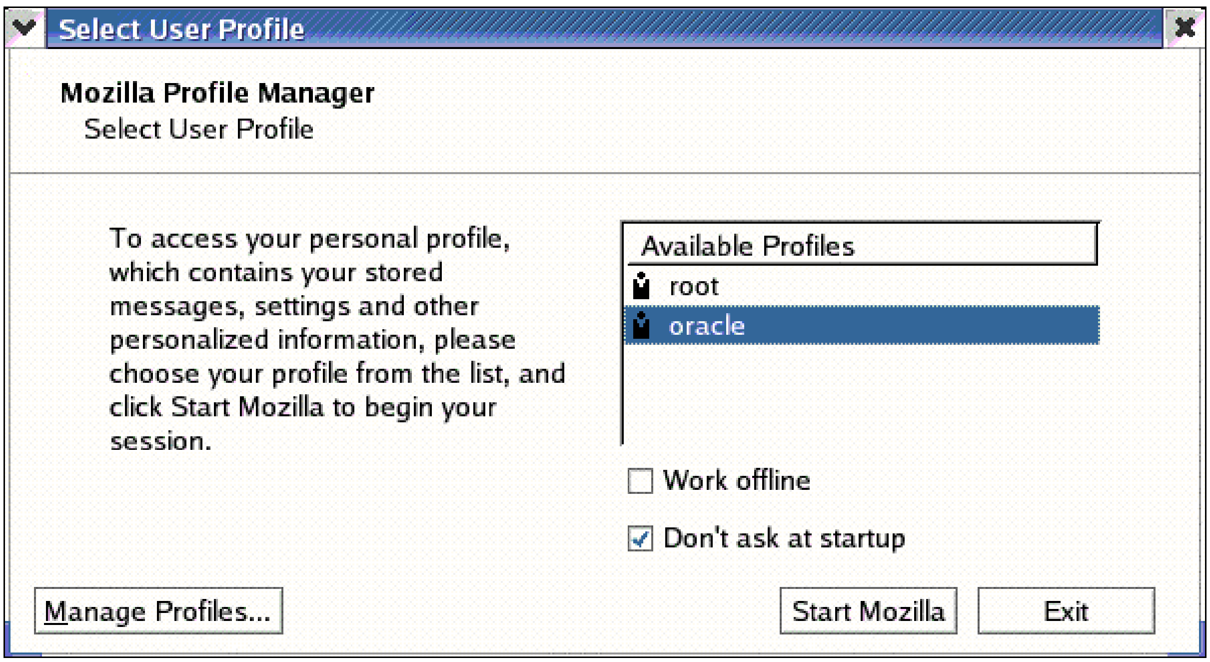
Background: You have just installed the Oracle software and created a database. You want to ensure that you can start and stop the database and see the application data.

1. Invoke Enterprise Manager, and log in as the `SYS` user. Which port number does this database use? You noted this in Practice 3.

   Answer: `1158`

   a) Open a Web browser.

   b) If using Mozilla, the Select User Profile dialog box appears.



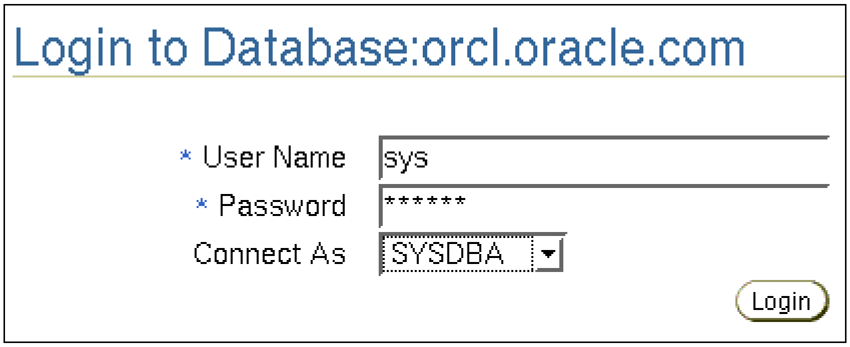
   Select **oracle** as the profile, and select **Don’t ask at startup**, and then click **Start Mozilla**.

   c) Enter the URL that you wrote down in Practice 3. It has the following format:

   http://*hostname*:*portnumber*/em

   The Oracle Enterprise Manager window appears.

d) Log in by entering `sys` as **User Name** and `oracle` as **Password**, and selecting `SYSDBA` as **Connect As**. Then, click **Login**.



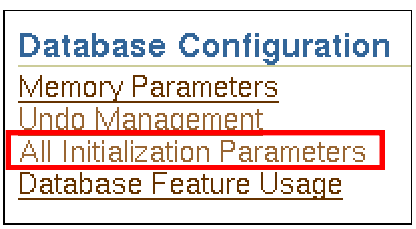
e) You may receive a security warning. In class, deselect **Alert me whenever I submit information that's not encrypted.**, and then click **Continue**.

The first time you log in after installing the software, the Oracle Database 10*g* Licensing Information page appears.

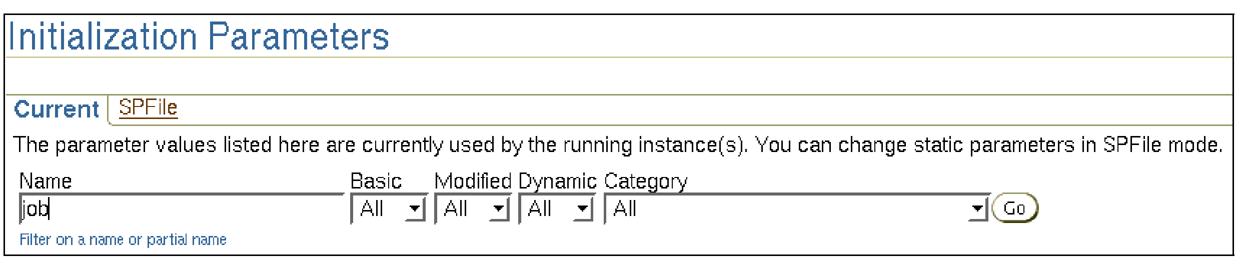
f) To acknowledge this information, click **I agree** at the bottom of the page.

2. View the initialization parameters and set the `JOB_QUEUE_PROCESSES` parameter to `15`. What SQL statement is run to do this?

a) Select **Administration** > Database Administration > Database Configuration > **All Initialization Parameters**.

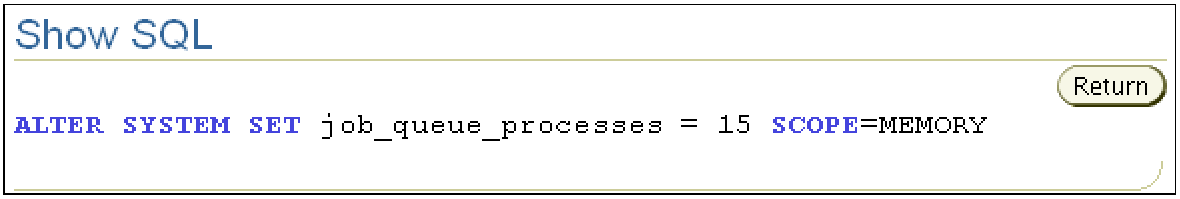


b) Enter `job` in the **Name** field, and then click **Go**.



c) When the `JOB_QUEUE_PROCESSES` initialization parameter appears, change its value from 10 to 15.

d) Click **Show SQL** and note the SQL statement that is going to be run.



e) Click **Return**, and then click **Apply**.

3. *Question:* What is the significance of a check in the Dynamic column?

   Answer: A "dynamic" parameter can be modified while the database is active.

4. Shut down the database instance by using Enterprise Manager.

   Question: What SQL is executed to do this?

   a) In the Enterprise Manager browser session, click the **Database** tab.

   b) Click **Shutdown**.

   c) For **Host Credentials**, enter `oracle` as **Username** and `oracle` as **Password**.

   d) Click OK.

   The Startup/Shutdown: Confirmation page appears.

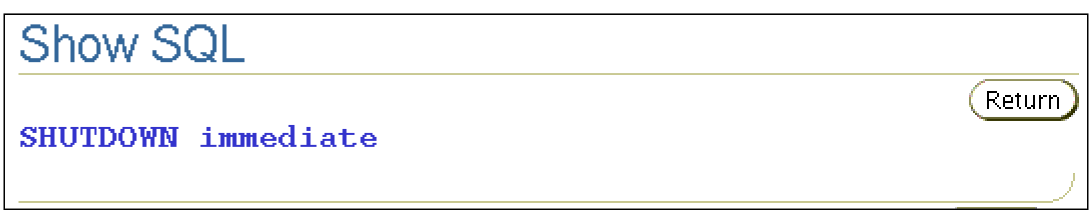
   e) Click **Advanced Options** to see the mode for shutting down, but do not change the mode; it should remain as "Immediate."

   f) Click **Cancel** to return to the previous page.

   g) Click **Show SQL** to view the SQL that is going to be executed to perform the shutdown operation.

   Question: What SQL is executed to do this?

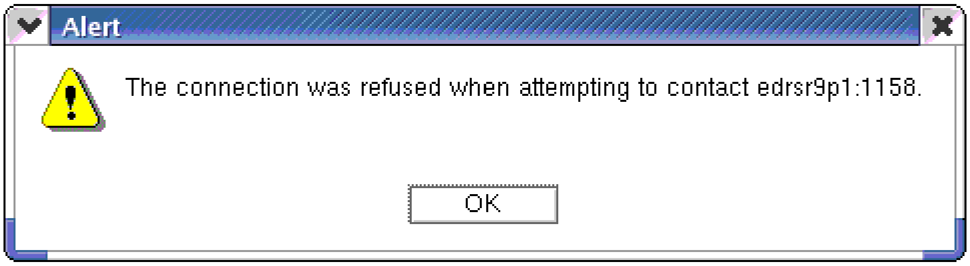
   Answer: `SHUTDOWN IMMEDIATE`



   h) Click **Return**.

i) Click **Yes** to confirm the shutdown operation.

j) Click Refresh. If you see the following error during the refresh, click **OK** and continue to refresh. The error will resolve itself.



k) Note that the **Status** of the instance is now “Down.”

5. Using SQL*Plus, verify that you are <u>not</u> able to connect as the `HR` user to a database that has been shut down.

a) In the Linux command window, enter the following to attempt to log in to the database:

```
$ sqlplus hr
```

b) Enter `hr` for the password.

c) Note the “ORACLE not available” error message.

d) Press [Ctrl], [D] to exit the username prompt.

6. Use Enterprise Manager to restart the database instance, and then log in as the `SYS` user again.

Question: What SQL is run to accomplish the database startup?

a) In Enterprise Manager, click the **Startup** button.

b) Enter `oracle` for both **Username** and **Password** in the **Host Credentials** region.

c) Click **OK**.

Solutions for Practice 4: Managing the Oracle Instance (continued)

d) The Startup/Shutdown: Confirmation page appears. Click **Show SQL** to view the SQL that is about to run.

Question: What SQL is run to accomplish the database startup?

Answer: STARTUP

e) Click **Return**.

f) Click **Yes** to confirm the startup operation.

g) The Startup/Shutdown: Activity Information page appears. Wait for the login page to appear, at which time you can log in as SYS user with the oracle password and the SYSDBA privilege.

7. In the alert log, view the phases that the database went through during startup. What are they?

a) Select **Database** > Related Links > **Alert Log Content**.

b) Scroll toward the bottom of the log and review the phases of the database during startup. Your alert log may look different from this screenshot, based on different system activities.

```
Fri May 20 09:47:26 2005
Database mounted in Exclusive Mode
Completed: ALTER DATABASE   MOUNT
Fri May 20 09:47:26 2005
ALTER DATABASE OPEN
Fri May 20 09:47:26 2005
Thread 1 opened at log sequence 209
  Current log# 1 seq# 209 mem# 0: /u01/app/oracle/oradata/orcl/redo01.log
Successful open of redo thread 1
Fri May 20 09:47:26 2005
MTTR advisory is disabled because FAST_START_MTTR_TARGET is not set
Fri May 20 09:47:26 2005
SMON: enabling cache recovery
Fri May 20 09:47:27 2005
Successfully onlined Undo Tablespace 1.
Fri May 20 09:47:27 2005
SMON: enabling tx recovery
Fri May 20 09:47:27 2005
Database Characterset is AL32UTF8
replication_dependency_tracking turned off (no async multimaster replication found)
Starting background process QMNC
QMNC started with Fri May 20 09:47:30 2005
db_recovery_file_dest_size of 2048 MB is 0.00% used. This is a
user-specified limit on the amount of space that will be used by this
database for recovery-related files, and does not reflect the amount of
space available in the underlying filesystem or ASM diskgroup.
Fri May 20 09:47:33 2005
Completed: ALTER DATABASE OPEN
```

c) Note that the modes the database goes through during startup are `MOUNT` and `OPEN`. Click **OK** to close the alert log.

8. Test access to *i*SQL*Plus for your HR application developers. (Navigation aid: Database > *i*SQL*Plus). Use the `Normal` role, `hr` username and password, and the default setting as Connect Identifier. If there is an error accessing *i*SQL*Plus, then start the isqlplus process using the `isqlplusctl start` command at the OS prompt, and then reattempt. After connecting, select the contents of the `EMPLOYEES` table.

a) Select **Database** > Related Links > **iSQL*Plus**.

The *i*SQL*Plus Connection Role page appears.

b) Notice that the `SYSOPER` and `SYSDBA` roles require special setup and authentication for security reasons. Select **Normal**, and then click **Continue**.

Solutions for Practice 4: Managing the Oracle Instance (continued)

c) If you see an error message saying that the connection was refused, that means you need to start the supporting process. To do this, enter the `isqlplusctl start` command at the OS prompt, and then retry step (b).

```
$ isqlplusctl start
iSQL*Plus 10.2.0.1.0
Copyright (c) 2003, 2005, Oracle.  All rights reserved.
Starting iSQL*Plus ...
iSQL*Plus started.
```

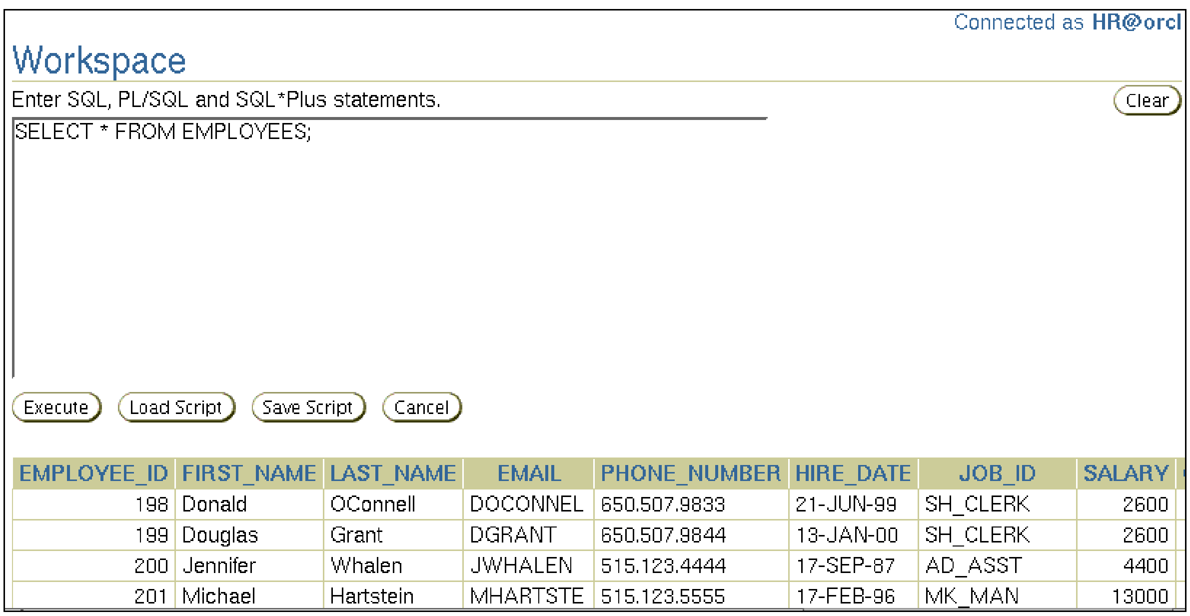
d) On the Login page, enter `hr` as **Username** and **Password** and leave the **Connect Identifier** set to its default value. Click the **Login** button.

The Confirm window appears, with the Password Manager offering to remember this login for you. Click **No**.

e) Enter the following statement in the workspace, and then click the **Execute** button:

```
SELECT * FROM EMPLOYEES;
```

The content of the `EMPLOYEES` table appears in a tabular form.



f) When you have finished reviewing the information, click **Logout** in the top-right corner of the page.

g) Click X in the top-right window frame to close the *i*SQL*Plus window.

Solutions for Practice 5: Managing Database Storage Structures

Background: You need to create a new tablespace for the INVENTORY application. You also need to create a database user that is not as privileged as the SYS user.

1. Enter ./lab_05_01.sh to run a script that creates the DBA1 user. It is located at /home/oracle/labs. The password for DBA1 is oracle.

   a) Start a Linux command shell by right-clicking your desktop and selecting **Open Terminal**.

   b) Change the current directory to the labs directory by entering:

   ```
   $ cd labs
   ```

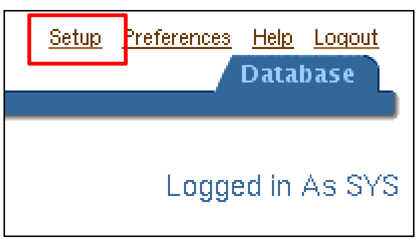
   c) Enter the following command to run the script that creates the DBA1 user:

   ```
   $ ./lab_05_01.sh
   ```

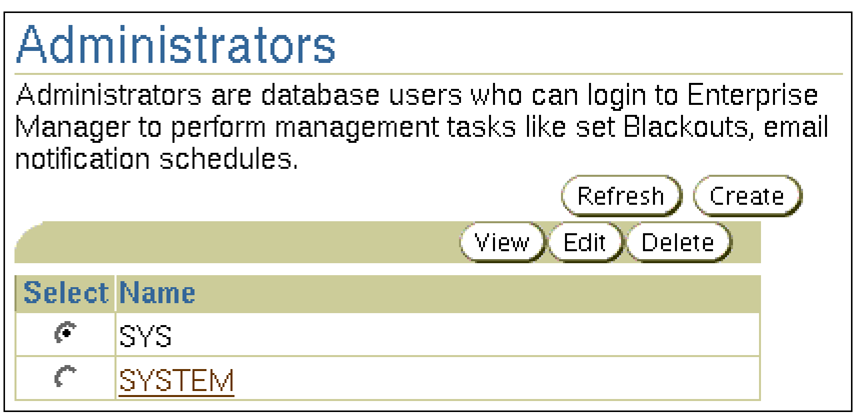
   d) Leave the command shell window open. You will use it again later.

2. Use the Setup link in the top-right corner of Enterprise Manager (EM) to define the DBA1 user as one who can perform administrative tasks in EM. When the non-SYS user is configured, log out as the SYS user and log in as the DBA1 user. Use the DBA1 user to perform the rest of these tasks, unless otherwise indicated.

   a) In the far top-right corner of the EM window, click **Setup**.



   b) Click **Create** to add the DBA1 user to the Administrators list. This will enable the DBA1 user to perform management tasks by using Enterprise Manager.



c) Enter `dba1` as **Name**, and `oracle` as **Password** and **Confirm Password**. Leave **Email Address** blank, and leave **Super Administrator** selected, and then click **Finish**.

d) On the Create Administrator: Review page, click **Finish** again.

e) Now that there is a non-`SYS` user, click **Logout** in the top-right corner, and then click **Login**.

f) Enter `DBA1` as **User Name** and `ORACLE` as **Password**, and select `SYSDBA` as **Connect As**. Then, click **Login**.

The first time you log in as a new user, the licensing page appears.

g) To acknowledge this information, click **I agree** at the bottom of the page.

The Database Home page appears.

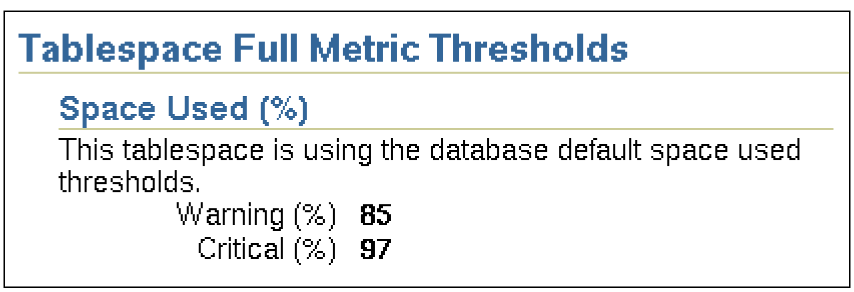
3. Using Enterprise Manager, view information about the `EXAMPLE` tablespace. Answer the following questions about it:

a) In Enterprise Manager, select **Administration** > Database Administration **>** Storage > **Tablespaces**.

b) Click the `EXAMPLE` tablespace name.

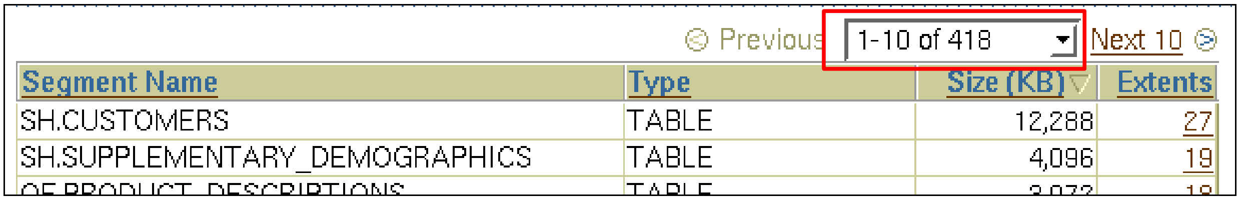
c) *Question 1:* What percentage of free space can be used up before the Warning threshold is reached?

Answer: 85%



d) From the **Actions** drop-down list, select **Show Tablespace Contents**, and then click **Go**.

e) The Show Tablespace Contents page appears.

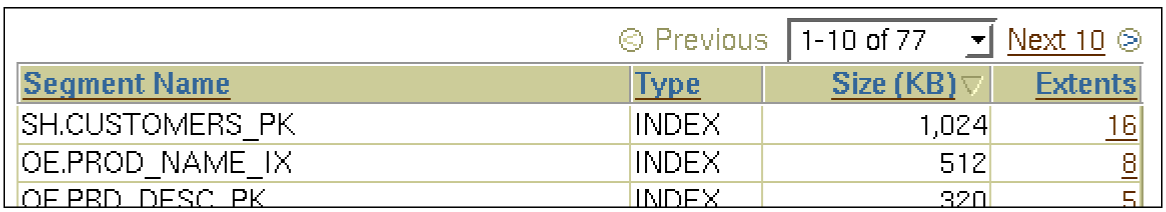

Previous 1-10 of 418 Next 10

Segment Name	Type	Size (KB)	Extents
SH.CUSTOMERS	TABLE	12,288	27
SH.SUPPLEMENTARY_DEMOGRAPHICS	TABLE	4,096	19

f) *Question 2:* How many segments are there in the `EXAMPLE` tablespace?

Answer: 418

g) Select INDEX from the **Type** drop-down list in the Search region, and then click **Go**.


Previous 1-10 of 77 Next 10

Segment Name	Type	Size (KB)	Extents
SH.CUSTOMERS_PK	INDEX	1,024	16
OE.PROD_NAME_IX	INDEX	512	8

h) *Question 3:* Which index in the `EXAMPLE` tablespace takes up the most space?

Answer: `SH.CUSTOMERS_PK`

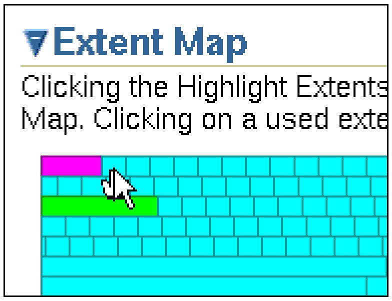
Question 4: Which segment is stored physically first in the tablespace? That is, which one is stored right after the tablespace header?

i) Scroll to the bottom of the page, and then click the plus icon to the left of the Extent Map label.

j) After several seconds, the extent map appears. Note that the map legend indicates that pink is the tablespace header.

k) Scroll back to the top of the page, select **All Types** from the **Type** drop-down list, and then click **Go**.

l) Click the extent just to the right of the tablespace header extent.



m) Scroll to the top of the page again, and note the segment that is being pointed to:

	PM.SYS_LOB0000051820C00004$$	LOBSEGMENT	128	2
	PM.SYS_LOB0000051819C00015$$	LOBSEGMENT	128	2
	HR.REGIONS	TABLE	64	1
	HR.LOCATIONS	TABLE	64	1

Answer: `HR.REGIONS`

4. Create a new, locally managed tablespace (LMT) called `INVENTORY` of size 5 MB.

Note: The `INVENTORY` tablespace will be used in later practice sessions during this class.

a) In Enterprise Manager, select **Administration** > Database Administration > Storage > **Tablespaces**.

b) Click **Create**.

c) Enter `INVENTORY` as the tablespace name, and verify that **Extent Management** is **Locally Managed**, **Type** is **Permanent**, **Status** is **Read Write**, and **Use Bigfile tablespace** is <u>not</u> selected.

d) Click **Add** in the Datafiles region.

e) On the Add Datafile page, enter `inventory01.dbf` for **File Name**, and `5 MB` as **File Size**.

f) Click **Continue**.

g) Click the **Storage** tab, and verify that **Extent Allocation** is **Automatic**, **Segment Space Management** is **Automatic**, and **Logging** is enabled.

h) Click the **General** tab.

i) Click **Show SQL** to see the SQL that will be run, and then click **Return**.

j) Click **OK**, and a successful Update Message appears.

5. As the `DBA1` user, run the `lab_05_05.sql` script to create and populate a table (called `X`) in the `INVENTORY` tablespace. What error do you eventually see?

   a) Right-click the desktop and select **Open Terminal** to start a command shell. Then enter:

```
$ cd labs
```

   b) Enter the following to run the script:

```
$ sqlplus dba1/oracle @lab_05_05.sql
```

   c) Note that there is eventually an error ORA-01653 stating that the table cannot be extended. There is not enough space to accommodate all of the rows to be inserted.

```
1024 rows created.

SQL> insert into x select * from x
  2  /
insert into x select * from x
*
ERROR at line 1:
ORA-01653: unable to extend table DBA1.X by 128 in tablespace INVENTORY


SQL> commit
  2  /

Commit complete.
```

6. Go to the Enterprise Manager window and define space for 50 MB in the tablespace instead of 5 MB, while keeping the same single data file in the tablespace. What is the `ALTER` statement that is executed to make this change?

   a) Select **Administration** > Database Administration **>** Storage > **Tablespaces**.

   b) Select the `INVENTORY` tablespace, and then click **Edit**.

   c) In the **Datafiles** region, click **Edit**.

   d) Change **File Size** from 5 MB to 50 MB.

   e) Click **Continue** to return to the General tabbed page.

   f) Click **Show SQL** to see the SQL that will be run. Note that it is an `ALTER DATABASE` statement. Click **Return**.

Solutions for Practice 5: Managing Database Storage Structures (continued)

Show SQL

Return

```
ALTER DATABASE DATAFILE
'/u01/app/oracle/oradata/ORCL/datafile/inventory01.dbf'
RESIZE 50M
```

g) Click **Apply**.

7. Go back to the terminal window and run the `lab_05_07.sql` script. It drops the table and reexecutes the original script that previously returned the space error.

   a) Go to the SQL*Plus window.

   b) Enter the following to run the script:

   ```
   $ sqlplus dba1/oracle @lab_05_07.sql
   ```

   c) Note that the same number of row inserts are attempted, and there is no error because of the increased size of the tablespace.

8. In an terminal window, run the `lab_05_08.sql` script to clean up the tablespace for later practice sessions.

   a) Enter the following to run the script:

   ```
   $ sqlplus dba1/oracle @lab_05_08.sql
   ```

Solutions for Practice 6: Administering User Security

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 must be able to log in to the ORCL database, select data from, and update records in the HR.EMPLOYEES 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 out after 15 minutes. You also need to create a new user account for the inventory application that you are installing.

1. **Mandatory task:** Review and run the lab_06_01.sh script (located in the /home/oracle/labs directory) to create the INVENTORY user, which you will use in the next practice.

   a) In a terminal window, enter:

```
cd /home/oracle/labs
more lab_06_01.sh
./lab_06_01.sh
```

2. Create a profile named HRPROFILE, allowing 15 minutes idle time.

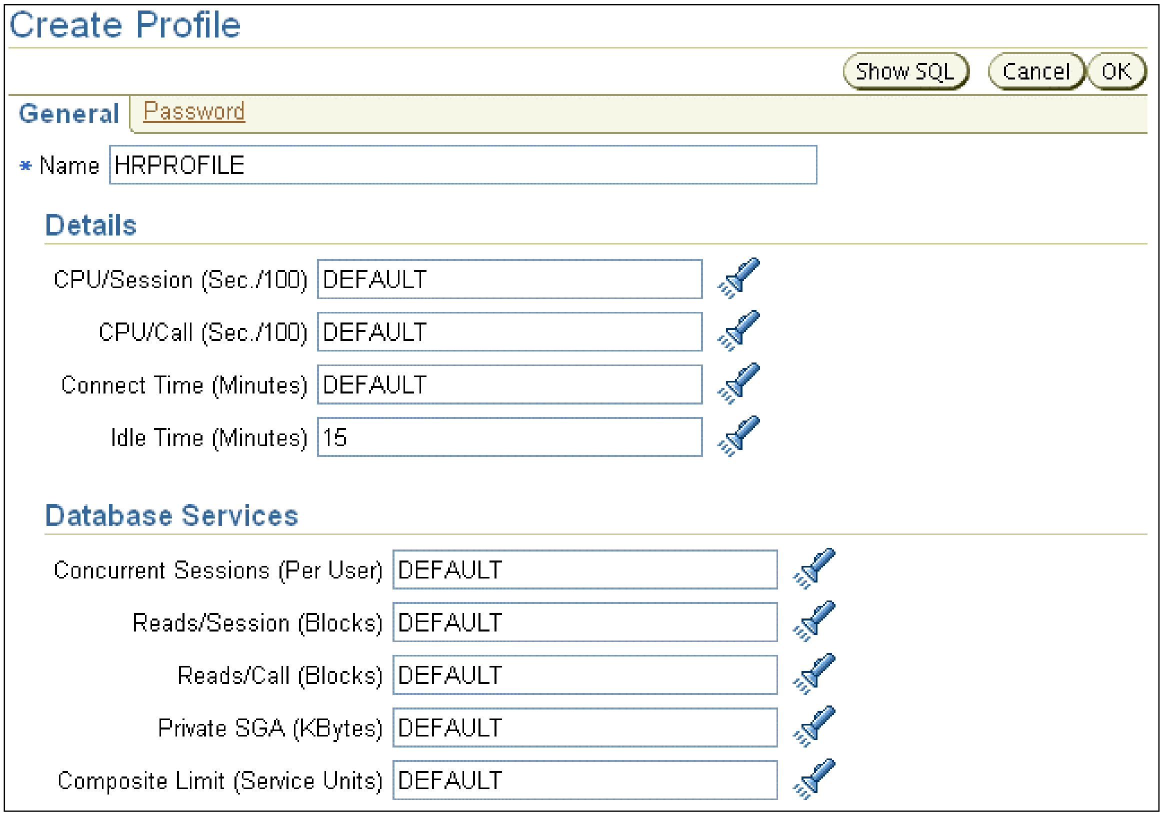
   a) Invoke Enterprise Manager as the DBA1 user in the SYSDBA role for your ORCL database.

   b) Select **Administration** > Schema > Users & Privileges > **Profiles**.

   c) Click the **Create** button.

   d) Enter **HRPROFILE** in the **Name** field.

   e) Enter **15** in the **Idle Time (Minutes)** field.



f) Leave all the other fields set to DEFAULT.

g) Click the **Password** tab, and review the Password options, which are currently all set to DEFAULT.

h) Optionally, click the **Show SQL** button, review your underlying SQL statement, and then click **Return**.

i) Finally, click **OK** to create your profile.

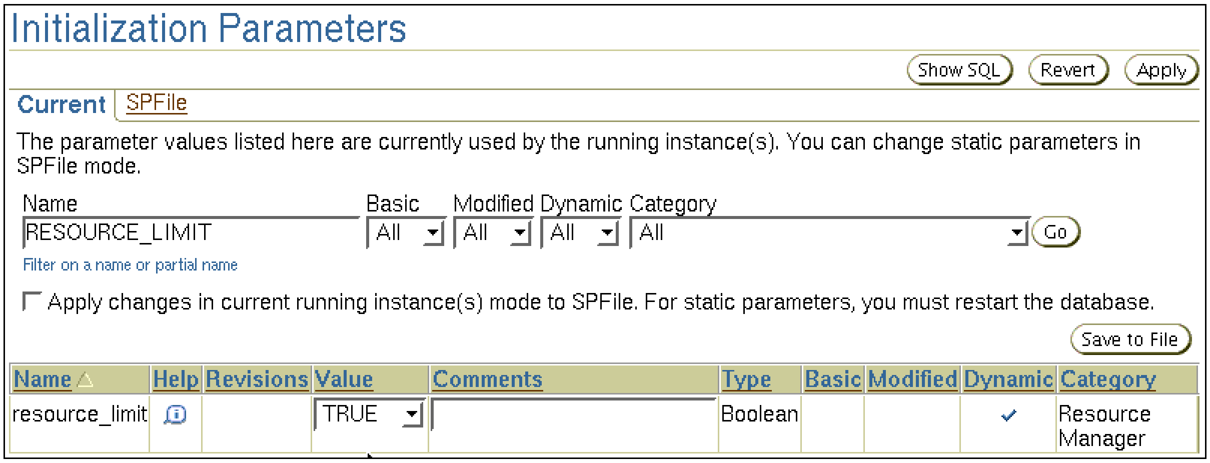
3. Set the `RESOURCE_LIMIT` initialization parameter to TRUE so that your profile limits will be enforced.

a) Select **Administration** > Database Administration > Database Configuration > **All Initialization Parameters**.

b) Enter **`RESOURCE_LIMIT`** in the **Filter** field, and then click **Go**.

c) Select **TRUE** from the **Value** drop-down list, and then click **Apply**.

Solutions for Practice 6: Administering User Security (continued)



4. Create the role named `HRCLERK` with `SELECT` and `UPDATE` permissions on the `HR.EMPLOYEES` table.

   a) Select **Administration** > Schema > Users & Privileges > **Roles**.

   b) Click the **Create** button at the top-right of the page.

   c) Enter **`HRCLERK`** in the **Name** field. This role is not authenticated.

   d) Click **Object Privileges**.

   e) Select **Table** from the **Select Object Type** drop-down list, and then click **Add**.

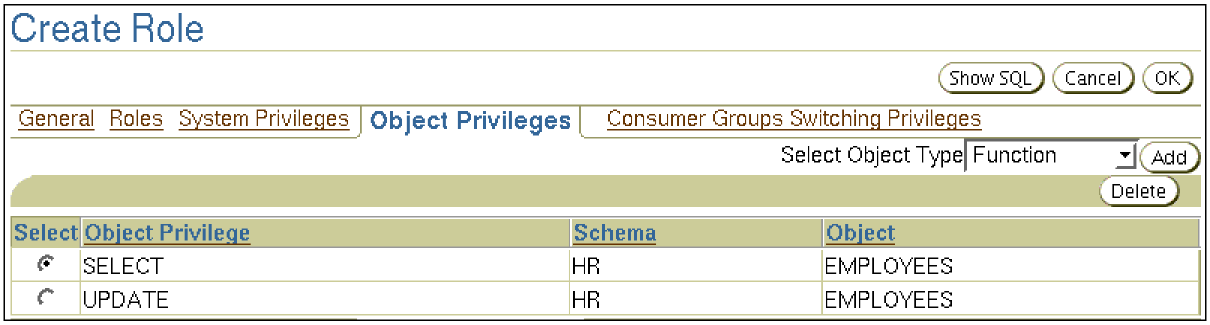
   f) Enter **`HR.EMPLOYEES`** in the **Select Table Objects** field.

   g) Move the `SELECT` and `UPDATE` privileges to the **Selected Privileges** box. Click **OK**.

   h) Click the **Show SQL** button, and review your underlying SQL statement.

```
CREATE ROLE "HRCLERK" NOT IDENTIFIED
GRANT SELECT ON "HR"."EMPLOYEES" TO "HRCLERK"
GRANT UPDATE ON "HR"."EMPLOYEES" TO "HRCLERK"
```

   i) Click **Return**, and then click **OK** to create the role.



5. Create the role named `HRMANAGER` with `INSERT` and `DELETE` permissions on the `HR.EMPLOYEES` table. Grant the `HRCLERK` role to the `HRMANAGER` role.

   a) Select **Administration** > Schema > Users & Privileges > **Roles**.

   b) Click **Create**.

   c) Enter **`HRMANAGER`** in the **Name** field. This role is not authenticated.

   d) Click **Object Privileges**.

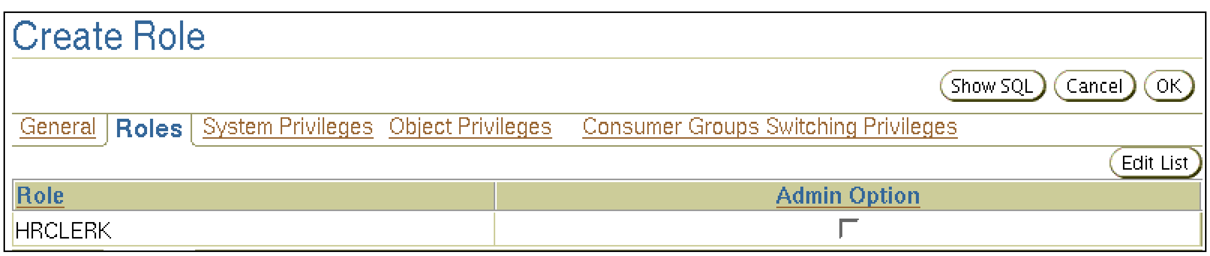
   e) Select **Table** from the **Select Object Type** drop-down list, and then click **Add**.

   f) Enter **`HR.EMPLOYEES`** in the **Select Table Objects** field.

   g) Move the `INSERT` and `DELETE` privileges to the **Selected Privileges** box. Click **OK**.

   h) Click **Roles**, and then click **Edit List**.

   i) Move the `HRCLERK` role into the **Selected Roles** box, and then click **OK**.



   j) Click the **Show SQL** button, and review your underlying SQL statement.

```
CREATE ROLE "HRMANAGER" NOT IDENTIFIED
GRANT DELETE ON "HR"."EMPLOYEES" TO "HRMANAGER"
GRANT INSERT ON "HR"."EMPLOYEES" TO "HRMANAGER"
GRANT "HRCLERK" TO "HRMANAGER"
```

   k) Click **Return**, and then click **OK** to create the role.

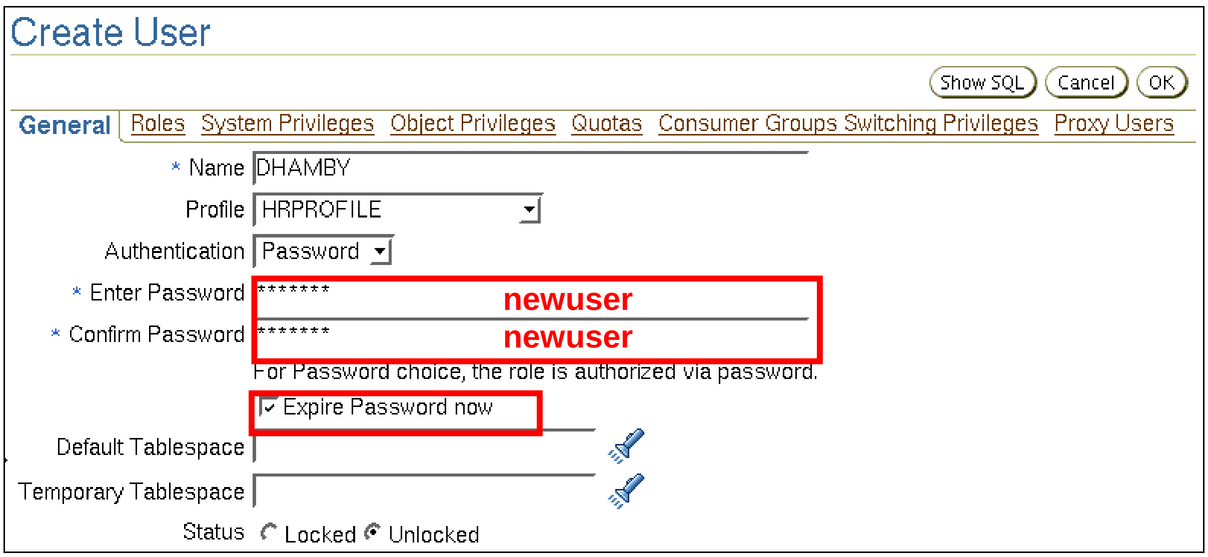
6. Create an account for David Hamby, a new HR clerk.

   a) Select **Administration** > Schema > Users & Privileges > **Users**.

   b) Click **Create**, and enter **`DHAMBY`** in the Name field.

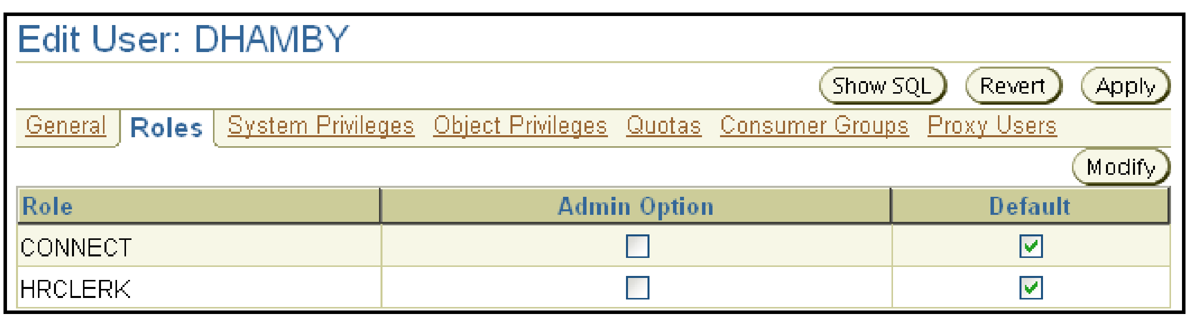
   c) Select **`HRPROFILE`** for the Profile.

   d) Select Password Authentication, and enter **`newuser`** as password. Enter it into the **Confirm Password** field also. Select the **Expire Password now** check box so that David will have to change the password the first time he logs in.



   e) Click **Roles**. Notice that the `CONNECT` role has automatically been assigned to the user.

   f) Add the `HRCLERK` role by clicking **Edit List** and moving the `HRCLERK` role into the **Selected Roles** box. Click **OK** to close the Modify Roles window.



   g) Click **OK** again to create the user.

7. Create an account for Rachel Pandya, another new HR clerk. Repeat the steps under step 6 with `RPANDYA` as the username.

8. Create an account for Jenny Goodman, the new HR manager. Repeat the steps under step 6 with `JGOODMAN` as the username and selecting the `HRMANAGER` role instead of the `HRCLERK` role.

   a) Click the **Show SQL** button and review your underlying SQL statement.

```
CREATE USER "JGOODMAN" PROFILE "HRPROFILE" IDENTIFIED BY "*******" PASSWORD
EXPIRE ACCOUNT UNLOCK
GRANT "CONNECT" TO "JGOODMAN"
GRANT "HRMANAGER" TO "JGOODMAN"
```

   b) Click **Return**, and then click **OK** to create the user.

9. Test the new users in SQL*Plus. Connect to the `ORCL` database as the `DHAMBY` user. Use `oracle` as the new password. Select the row with `EMPLOYEE_ID=197` from the `HR.EMPLOYEES` table. Then, attempt to delete it. (You should get the "insufficient privileges" error.)

   a) In a terminal window, enter:

```
sqlplus dhamby/newuser
```

   or if you are already in SQL*Plus, use the `CONNECT` command. If you reconnect as `dhamby` in SQL*Plus, the login and change-of-password session looks like this:

```
SQL> CONNECT dhamby/newuser

ERROR:
ORA-28001: the password has expired

Changing password for dhamby
New password: oracle    <<<Password does not appear on screen
Retype new password: oracle    <<<Password does not appearon screen
Password changed

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL>
```

   b) Select the salary for `EMPLOYEE_ID=197` from the `HR.EMPLOYEES` table.

```
SQL> SELECT salary FROM hr.employees WHERE EMPLOYEE_ID=197;

          SALARY
      ----------
            3000
```

c) Now attempt to delete the same record from the `hr.employees` table.

```
SQL> DELETE FROM hr.employees WHERE EMPLOYEE_ID=197;
DELETE FROM  hr.employees WHERE EMPLOYEE_ID=197
                *
ERROR at line 1:
ORA-01031: insufficient privileges
```

10. Repeat the delete attempt as the `JGOODMAN` user. After deleting the row, issue a rollback, so that you still have the original 107 rows.

a) Connect to the `ORCL` database as the `JGOODMAN` user.

```
SQL> connect jgoodman/newuser
ERROR:
ORA-28001: the password has expired
<Change the password as shown above>
```

b) Select the row with `EMPLOYEE_ID=197` from the `HR.EMPLOYEES` table.

```
SQL> SELECT salary FROM hr.employees WHERE EMPLOYEE_ID=197;

     SALARY
   ----------
      3000
```

c) Now delete the same row from the `HR.EMPLOYEES` table.

```
SQL> DELETE FROM  hr.employees WHERE EMPLOYEE_ID=197;

1 row deleted.
```

d) Roll back the delete operation (because this was just a test).

```
SQL> rollback;

Rollback complete.
```

e) Confirm that you still have 107 rows in this table.

```
SQL> SELECT COUNT(*) FROM hr.employees;

  COUNT(*)
----------
       107

SQL>
```

Question 1: Where was the row stored after deletion?

Answer: It was stored in the Undo tablespace.

Solutions for Practice 6: Administering User Security (continued)

Question 2: When you created the new users, you did not select a default or temporary tablespace. What determines the tablespaces that the new users will use?

Answer: The system-defined default permanent and temporary tablespaces

Question 3: You did not grant the `CREATE SESSION` system privilege to any of the new users, but they can all connect to the database. Why?

Answer: Because Enterprise Manager automatically assigns the `CONNECT` role to the new users, and `CREATE SESSION` is contained within that role

11. Review the `lab_05_01.sql` script and the `lab_05_01.txt` log file that it generated when you created the `DBA1` user.

    a) Double-click the **oracle's Home** icon on your desktop.

    b) Navigate to the **`labs`** directory.

    c) Double-click the **`lab_05_01.sql`** file, and review its content.

    d) When you have finished reviewing the file, click the **Up** icon to return to the `labs` directory.

    e) Now, double-click the **`lab_05_01.txt`** file, which was created when you executed the `lab_05_01.sql` file.

    f) When you have finished reviewing the file, click the X (Close Window) icon.

12. Use SQL*Plus to connect to the `ORCL` database as the `RPANDYA` user. Change the password to `oracle`. (You must change the password, because this is the first connection as `RPANDYA`.) Leave `RPANDYA` connected during the next lesson or at the end of the day. `HRPROFILE` specifies that users whose sessions are inactive for more than 15 minutes will automatically be logged out. Verify that the user was automatically logged out by trying to select from the `HR.EMPLOYEES` table again.

```
ERROR at line 1:
ORA-02396: exceeded maximum idle time, please connect again
```

Solutions for Practice 7: Managing Schema Objects

Background: You need to create schema objects for the new inventory application. Work as the DBA1 user in the SYSDBA role for your ORCL database.

1. Return to the Enterprise Manager browser session, or invoke EM as the DBA1 user in the SYSDBA role for your ORCL database.

2. In the INVENTORY tablespace, create the PRODUCT_MASTER table in the INVENTORY schema. The specifications of the table are:

   PRODUCT_ID number(7). This is the primary key field. (Constraint name: PK_INV)
   PRODUCT_NAME varchar2(50) with a Not NULL constraint
   CODE varchar2(10) with a Not NULL constraint
   REORDER_THRESHOLD number(5) with a check constraint ensuring that the number is always greater than zero (Constraint name: CHK_REORDER)
   COST number(5,2)
   PRICE number(5,2)

   a) Select **Administration** > Schema > Database Objects > **Tables**.

   b) Click **Create**.

   c) Leave the default table type to be **Standard, Heap Organized permanent**, and then click **Continue**.

   d) Enter PRODUCT_MASTER as table name and set **Schema** to INVENTORY, and leave the Tablespace setting as the default for the user, which is INVENTORY.

   e) Enter the first five columns' information (including Data Type, Size, Scale, and Not NULL); but no constraints yet.

   f) Click **Add 5 Table Columns**, and then enter the last column, PRICE, along with its type and size.

   g) Click the **Constraints** tab.

   h) Set the drop-down list value to PRIMARY, and then click **Add**.

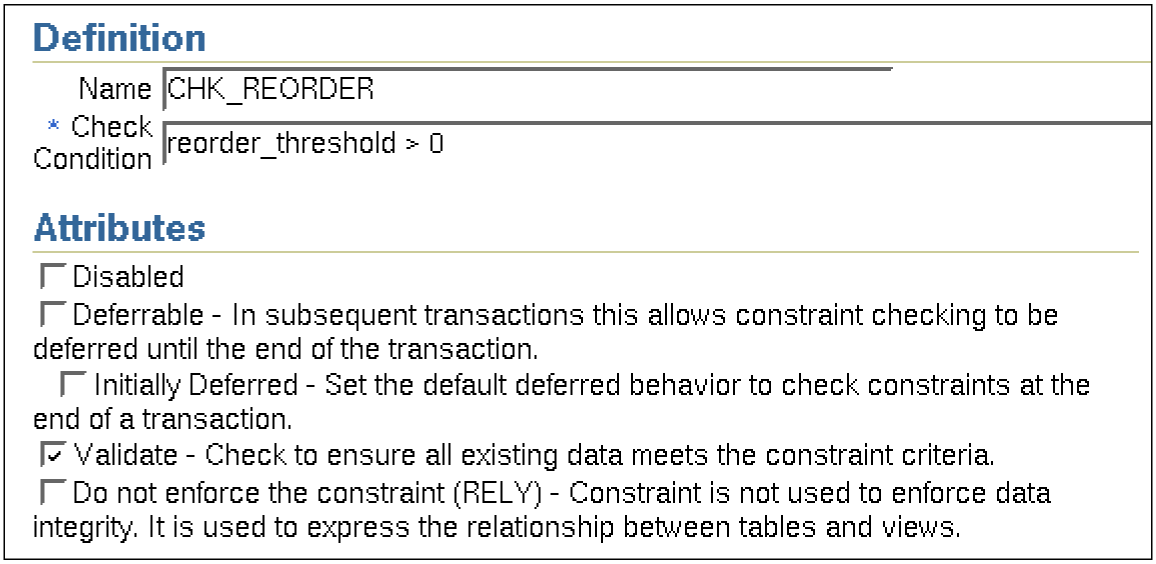
   i) Give the constraint a name by entering PK_INV in the **Name** field.

   j) Double-click PRODUCT_ID in the left list to move it to the right list, to make it alone the primary key. Then, click **Continue**.

k) Set the constraint type drop-down list value to CHECK, and then click **Add**.

l) Enter `CHK_REORDER` for the name of the check constraint.

m) Enter "`reorder_threshold > 0`" (without the quotation marks) in the **Check Condition** field.



n) Leave the Attributes settings at their default settings, and then click **Continue**.

o) Your constraint list should now look like this:

Select	Name	Type	Table Columns	Disabled	Deferrable	Initially Deferred	Validate	RELY
○	PK_INV	PRIMARY	PRODUCT_ID	NO	NO	NO	YES	NO
●	CHK_REORDER	CHECK		NO	NO	NO	YES	NO

p) Click **OK** to create the table. If you receive errors, correct them, and then click **OK** again.

Solutions for Practice 7: Managing Schema Objects (continued)

3. In the `INVENTORY` tablespace, create the `PRODUCT_ON_HAND` table in the `INVENTORY` schema. You have been given the `lab_07_03.sql` script to create the table, but there is a problem with it (*intentionally created to enhance your learning experience*). Fix the problem, and run the script. If you cannot find the error right away, then go ahead and run the original script in SQL*Plus to see the error message. This helps you discover and solve the problem. The specifications of the table are:

   `PRODUCT_ID` number(7). 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)
   `LAST_UPDATE` date

   a) Edit the `lab_07_03.sql` in the `/home/oracle/labs` directory. It has an error in it. If you can spot the error, make the change to correct it. Run the script to create the table by entering this on the OS command line:

   ```
   $ sqlplus dba1/oracle @lab_07_03.sql
   ```

   b) The error in the script is that "`(PRODUCT_ID)`" is missing after "`FOREIGN KEY`." So, add "`(PRODUCT_ID)`".

   ```
   SQL> CREATE TABLE INVENTORY.PRODUCT_ON_HAND
     2  (
     3  PRODUCT_ID NUMBER(7),
     4  QUANTITY NUMBER(5),
     5  WAREHOUSE_CITY VARCHAR2(30),
     6  LAST_UPDATE DATE,
     7    CONSTRAINT FK_PROD_ON_HAND_PROD_ID
     8      FOREIGN KEY REFERENCES
     9      INVENTORY.PRODUCT_MASTER (PRODUCT_ID) VALIDATE
    10  )
    11  /
       FOREIGN KEY REFERENCES
                   *
   ERROR at line 8:
   ORA-00906: missing left parenthesis
   ```

   (PRODUCT_ID)

   c) Remove the `prompt` commands:

   ```
   prompt  There is an error in this statement. It will not
   prompt  run successfully unless fixed.
   ```

   d) Run the script. The table should be created without error.

Solutions for Practice 7: Managing Schema Objects (continued)

4. In the INVENTORY tablespace, create the OBSOLETE_PRODUCTS table in the INVENTORY schema. This table definition is very much like that of the PRODUCT_MASTER table, so you can use Enterprise Manager's ability to "Define Using SQL" rather than using "Column Specification." The specifications of the table are:

   PRODUCT_ID number(7). 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)

   a) In Enterprise Manager, select **Administration** > Schema > Database Objects > **Tables**.

   b) Click **Create**.

   c) Leave the default setting for **Table Organization**, and then click **Continue**.

   d) Enter OBSOLETE_PRODUCTS for **Name**.

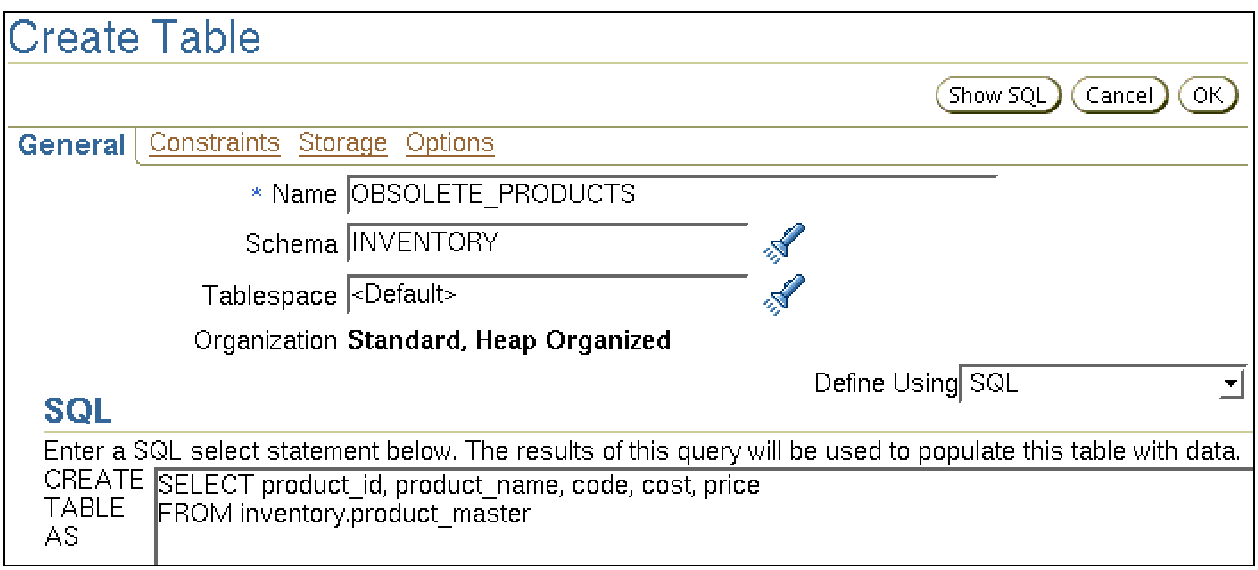
   e) Enter INVENTORY for **Schema**.

   f) Leave the Tablespace setting as the default for this schema.

   g) Set the **Define Using** drop-down list to **SQL**.

   h) In the SQL region, enter the following statement:

```
SELECT product_id, product_name, code, cost, price
FROM inventory.product_master
```



   i) Click **OK** to create the table.

5. In the `INVENTORY` tablespace, create an index called `OBS_CODE` on the `CODE` column of the `OBSOLETE_PRODUCTS` table in the `INVENTORY` schema. Choose an appropriate index type: either B-tree or Bitmap. Explain the reason for your choice.

   a) *Question:* Which type of index is appropriate, and why?

   Answer: B-tree, because the `CODE` column can contain many different values, not just a small, finite set.

   b) Select **Administration** > Schema > Database Objects > **Indexes**, and the click **Create**.

   c) Enter `OBS_CODE` as **Name**.

   d) Enter `INVENTORY` as **Schema**.

   e) Enter `INVENTORY.OBSOLETE_PRODUCTS` as **Table Name**.

   f) Click **Populate Columns**. The column names have been filled into the table. If your browser prompts you to remember entered values, click **No**.

   g) Enter an `ORDER` value of `1` for the `CODE` column.

Table Columns

Column Name	Data Type	Sorting Order	Order
PRODUCT_ID	NUMBER	ASC	
PRODUCT_NAME	VARCHAR2	ASC	
CODE	VARCHAR2	ASC	1
COST	NUMBER	ASC	
PRICE	NUMBER	ASC	

Add Column Expression

   h) Click **Show SQL**, confirm the SQL statement looks like the statement below, and then click **Return**.

```
CREATE INDEX "INVENTORY"."OBS_CODE" ON
"INVENTORY"."OBSOLETE_PRODUCTS" ("CODE")
```

   i) Click **OK** to create the index.

Solutions for Practice 7: Managing Schema Objects (continued)

6. In the `INVENTORY` tablespace, create an index called `PROD_NAME_CODE` on the combined `PRODUCT_NAME` and `CODE` columns of the `PRODUCT_MASTER` table. Use the `lab_07_06.sql` script *(which contains an error, intentionally created to enhance your learning experience*). Correct the error and run the script. If you cannot find the error right away, then run the original script in SQL*Plus to see the error message. This will help you discover and solve the problem.

   a) Edit the `lab_07_06.sql` in `the /home/oracle/labs` directory. It has an error in it. If you can spot the error, make the change to correct it.

```
There is an error in this script.
Correct the error to have the index
created successfully.
create composite index inventory.prod_name_code
       *
ERROR at line 1:
ORA-00901: invalid CREATE command
```

Remove this

   The corrected statement is:

```
create index inventory.prod_name_code
on inventory.product_master(product_name,code)
```

   b) Remove the `prompt` commands:

```
prompt  There is an error in this script.
prompt  Correct the error to have the index
prompt  created successfully.
```

   c) Run the script to create the index by entering the following on the OS command line:

```
$ sqlplus dba1/oracle @lab_07_06.sql
```

   d) The error in the script is that it should not have the word "`COMPOSITE`". Delete the word, run the script, and the index should be created without error.

7. In the `INVENTORY` tablespace, use SQL*Plus to create a combined index on the `PRODUCT_ID` and `QUANTITY` columns of the `PRODUCT_ON_HAND` table. The index name should be `POH_PROD_ID_QTY`.

   a) At the SQL*Plus prompt, enter the following command:

```
SQL> create index inventory.poh_prod_id_qty on
inventory.product_on_hand(product_id, quantity);
```

Solutions for Practice 7: Managing Schema Objects (continued)

8. You receive an update for the inventory application that requires you to add two columns to the `PRODUCT_MASTER` table. Add a column called `PRIMARY_SOURCE` of the data type `varchar2(50)`. Add another column called `SECONDARY_SOURCE` of the data type `varchar2(50)`. What is the SQL that executes to do this?

   a) Select **Administration** > Schema > Database Objects > **Tables**.

   b) Enter `INVENTORY` in the **Schema** field, and then click **Go**.

   c) Select the `PRODUCT_MASTER` table, and then click **Edit**.

   d) Enter `PRIMARY_SOURCE` in the **Name** field under `PRICE`, and set **Data Type** to `VARCHAR2`, and **Size** to `50`.

   e) Enter `SECONDARY_SOURCE` in the next available **Name** field, and set **Data Type** to `VARCHAR2`, and **Size** to `50`.

   f) Click **Show SQL**.

```
ALTER TABLE "INVENTORY"."PRODUCT_MASTER"
ADD ( "PRIMARY_SOURCE" VARCHAR2(50),
"SECONDARY_SOURCE" VARCHAR2(50))
```

   g) Click **Return**, and then click **Apply**.

9. You receive another update for the inventory application. This change request instructs you to drop the `OBSOLETE_PRODUCTS` table and add the `OBSOLETED` column to the `PRODUCT_MASTER` table, with data type `DATE`. Do this using EM. What clause is added to the end of the `DROP TABLE` statement to also remove the table constraints?

   a) Select **Administration** > Schema > Database Objects > **Tables**.

   b) Enter `INVENTORY` in the **Schema** field, and then click **Go**.

   c) Select the `OBSOLETE_PRODUCTS` table, and then click **Delete With Options**.

   d) Keep the default settings for performing `DROP`, and then click **Show SQL** to see the `CASCADE CONSTRAINTS` option.

```
DROP TABLE "INVENTORY"."OBSOLETE_PRODUCTS" CASCADE CONSTRAINTS
```

   e) Click **Return**, and then click **Yes**.

   f) Select the `PRODUCT_MASTER` table, and then click **Edit**.

g) In the first empty **Name** field, enter `OBSOLETED`, and set **Data Type** to `DATE`.

h) Click **Apply**.

10. Another change request to the inventory application instructs you to create a view called `WAREHOUSE_VW` (Navigation aid: Administration > Views). The view is in the `INVENTORY` schema and displays (in this order):

- The name of the product
- The amount of "product on hand"
- The warehouse city name

You will have to join two tables together to create this view.

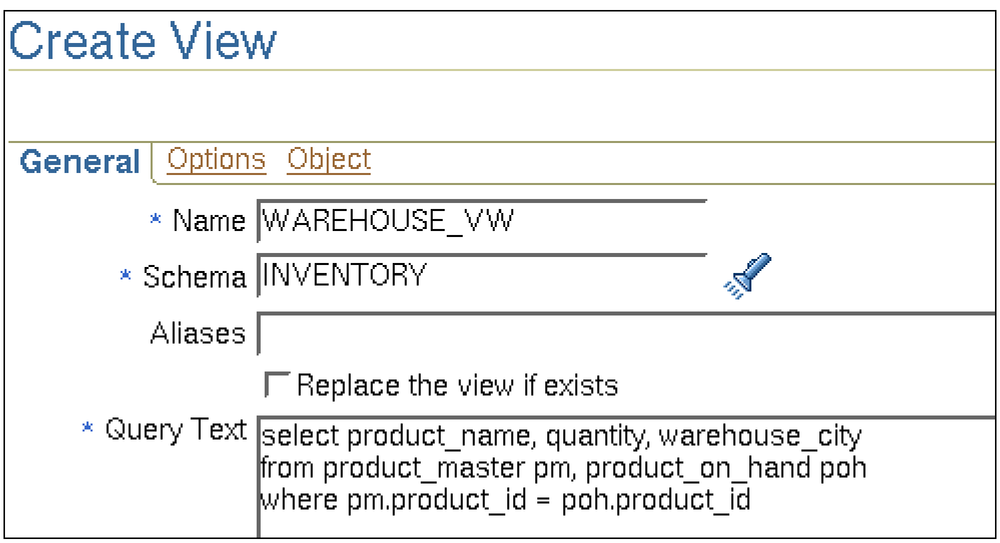
a) Select **Administration** > Schema > Database Objects > **Views**.

b) Click **Create**.

c) Enter `WAREHOUSE_VW` in the **Name** field, and `INVENTORY` in the **Schema** field.

d) Enter the following in the **Query Text** field:

```
select product_name, quantity, warehouse_city
from product_master pm, product_on_hand poh
where pm.product_id = poh.product_id
```



e) Click **OK**.

11. You receive a notice from developers that there is a certain type of query that will be run very frequently, and they want to be sure that it runs in less than one second. You need to run the query and see how it performs. First, run the `lab_07_11_a.sql` script to generate some test data of the volume indicated in the request. Then, run the query in the `lab_07_11_b.sql` script several times to see the average run time. Note that it takes several seconds to run each time. Create a function-based index on the `CODE` column that will improve the performance of this query.

    a) Right-click the desktop and select **Open Terminal**. Then enter the following at the command prompt to populate the table with test data. This will take three to five minutes to run.

    ```
    $ cd /home/oracle/labs
    $ sqlplus dba1/oracle @lab_07_11_a.sql
    ```

    b) Enter the following on the command line to run the test query.

    ```
    SQL> @lab_07_11_b.sql
    ```

    c) Enter `/` (a slash), and press [Enter] to run it again. Repeat this several times until you establish an average run time. It should take several seconds each time.

    d) Select **Administration** > Schema > Database Objects > **Indexes**.

    e) Click **Create**.

    f) Enter `CODE_FUNC` in the **Name** field.

    g) Enter `INVENTORY` in the **Schema** field.

    h) Enter `INVENTORY.PRODUCT_MASTER` in the **Table Name** field.

    i) Click **Populate Columns**.

    j) Click **Add Column Expression**.

    k) Enter `upper(substr(code,5,2))` in the newly added empty **Column Name** field.

    l) Enter `1` in the **Order** field beside the expression that you just entered.

    m) Click **Show SQL** to confirm that the SQL statement looks like this:

    ```
    CREATE INDEX "INVENTORY"."CODE_FUNC" ON "INVENTORY"."PRODUCT_MASTER"
    (upper(substr(code,5,2)))
    ```

n) Click **Return**.

o) Click **OK**.

p) Return to the SQL*Plus command line and run the benchmark query a few more times. Note that the execution time is greatly reduced.

12. Use *i*SQL*Plus to identify the data dictionary view name that you would use to list all constraints that the `INVENTORY` user can see.

a) In the Enterprise Manager browser session, select **Database** > Related Links > **iSQL*Plus**.

The *i*SQL*Plus Connection Role page appears.

b) Select **Normal**, and then click **Continue**.

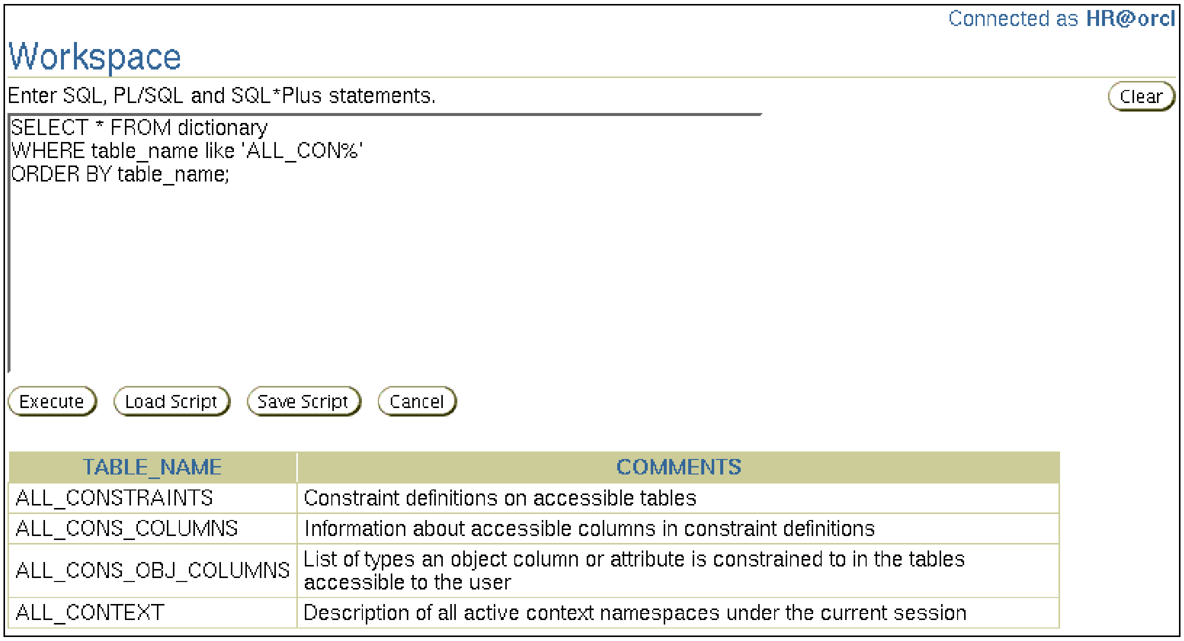
c) On the Login page, enter `inventory` as **Username** and `verysecure` as the **Password**, leave the **Connect Identifier** set to its default value, and then click **Login**.

The Confirm window appears, with the Password Manager offering to remember this login for you. Click **No**.

d) You are looking for things that the `INVENTORY` user has access to. So, you must use a view with the `ALL_` prefix. You can assume that the name of the view that shows constraint information begins with "`ALL_CON`." In the *i*SQL*Plus Workspace, enter:

```
SELECT * FROM dictionary
WHERE table_name like 'ALL_CON%'
ORDER BY table_name;
```

e) Click the **Execute** button.



TABLE_NAME	COMMENTS
ALL_CONSTRAINTS	Constraint definitions on accessible tables
ALL_CONS_COLUMNS	Information about accessible columns in constraint definitions
ALL_CONS_OBJ_COLUMNS	List of types an object column or attribute is constrained to in the tables accessible to the user
ALL_CONTEXT	Description of all active context namespaces under the current session

f) Review the returned rows. One of them has the `COMMENTS` value “Constraints definitions on accessible tables.” The object name is `ALL_CONSTRAINTS`.

13. How many indexes are owned by the `INVENTORY` user? You are looking for database objects owned by the `INVENTORY` user, so you know that the data dictionary view begins with the “`USER_`” prefix. The view name is `USER_INDEXES`.

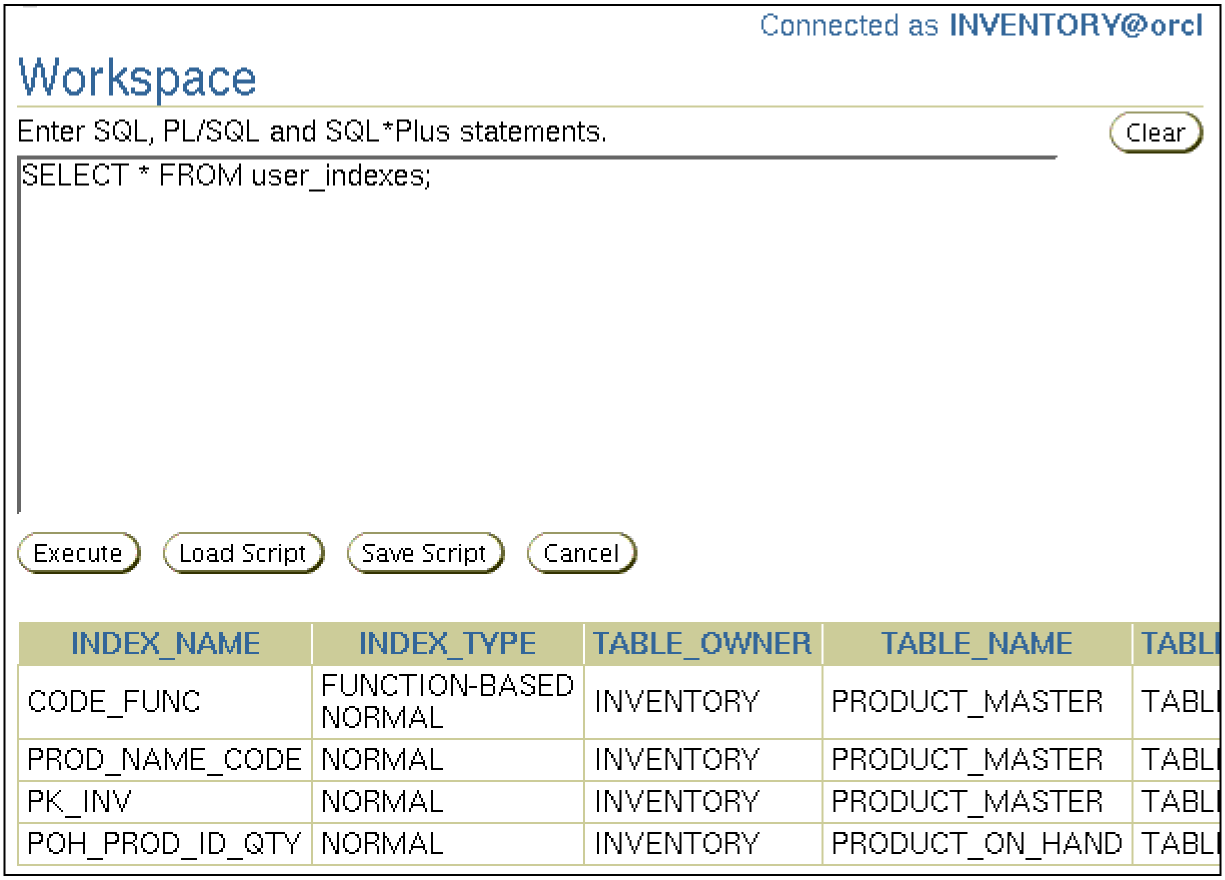
a) In the *i*SQL*Plus Workspace, enter:

```
SELECT * FROM user_indexes;
```

b) Click the **Execute** button.

How many indexes are owned by the `INVENTORY` user?

Answer: 4



INDEX_NAME	INDEX_TYPE	TABLE_OWNER	TABLE_NAME	TABLI
CODE_FUNC	FUNCTION-BASED NORMAL	INVENTORY	PRODUCT_MASTER	TABLI
PROD_NAME_CODE	NORMAL	INVENTORY	PRODUCT_MASTER	TABLI
PK_INV	NORMAL	INVENTORY	PRODUCT_MASTER	TABLI
POH_PROD_ID_QTY	NORMAL	INVENTORY	PRODUCT_ON_HAND	TABLI

c) Note that there are four rows returned.

d) Log out of *i*SQL*Plus by clicking **Logout** in the top-right corner of the window. Then, close the window by clicking **X** (Close icon) in the top-right corner.

Solutions for Practice 8: Managing Data and Concurrency

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 could not do anything else. SQL script files are provided for you in the `/home/oracle/labs` directory.

1. Make an uncommitted update to the row in question by running the `lab_08_01.sql` script. Do not worry if the session seems to “hang”—this is the condition you are trying to create.

   a) Enter the following to run the script. When the script completes executing, you will see a note stating that an uncommitted update has been made.

```
$ sqlplus dba1/oracle @lab_08_01.sql
```

```
SQL> show user
USER is "NGREENBERG"
SQL> update hr.employees set phone_number='650.555.1212' where employee_id = 110;

1 row updated.

SQL> prompt User "ngreenberg" made an update and left it uncommitted in this session.
User "ngreenberg" made an update and left it uncommitted in this session.
SQL>
SQL>
SQL>
```

2. Make an attempt to update the same row in a separate session by running, in a separate terminal window, the `lab_08_02.sql` script. Make sure you see the message “Update is being attempted now” before moving on.

   a) Right-click the desktop and select **Open Terminal** to start another command shell. Then enter the following to run the second script.

```
$ sqlplus dba1/oracle @lab_08_02.sql
```

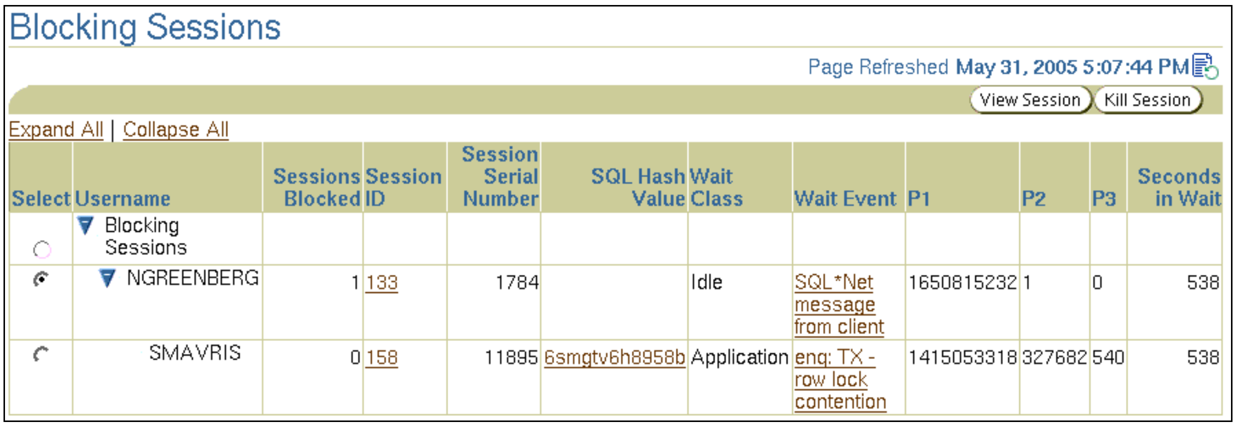
```
Sleeping for 20 seconds to ensure first process gets the lock first.

PL/SQL procedure successfully completed.

Sleep is finished.
Connected.
USER is "SMAVRIS"
Update is being attempted now.
```

3. Using the **Blocking Sessions** link on the Performance page, detect which session is causing the locking conflict.

   a) In Enterprise Manager, click the Performance page.

   b) The first time you invoke the Performance page, a Software License Agreement is displayed. Press [A] to accept and continue.

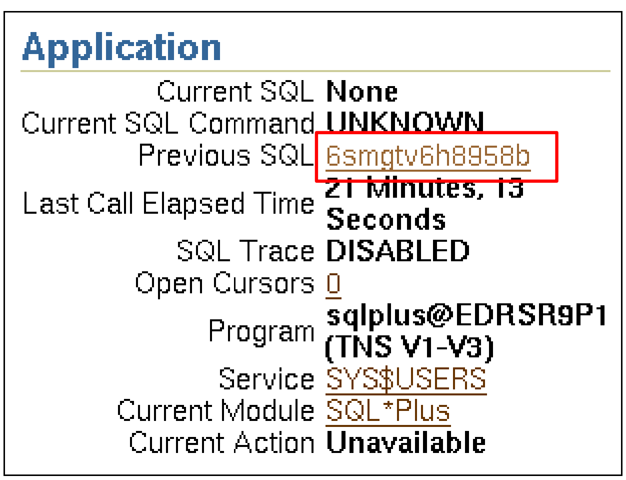
   c) Click **Blocking Sessions** in the **Additional Monitoring Links** area.



4. What was the last SQL statement that the blocking session executed?

   a) Select the NGREENBERG session, and then click **View Session**.

   b) Click the hash value link named "**Previous SQL**."



Solutions for Practice 8: Managing Data and Concurrency (continued)

c) Note the SQL that was most recently run.

Text

```
update hr.employees set phone_number='650.555.1212'
where employee_id = 110
```

5. Resolve the conflict in favor of the user who complained, by killing the *blocking* session. What SQL statement resolves the conflict?

a) Click the browser's **Back** button.

b) Now, on the Session Details: NGREENBERG page, click **Kill Session**.

c) Leave the Options set to **Kill Immediate**, and then click **Show SQL** to see the statement that is going to be executed to kill the session.

Note: Your session and serial number are most likely to be different from those shown here.

```
ALTER SYSTEM KILL SESSION '133,1784' IMMEDIATE
```

d) Click **Return**, and then click **Yes** to carry out the `KILL SESSION` command.

6. Return to the SQL*Plus command window, and note that `SMAVRIS`'s update has now completed successfully. It may take a few seconds for the success message to appear.

```
USER is "SMAVRIS"
Update is being attempted now.

1 row updated.

Update is completed.
SQL>
```

a) Close all open SQL sessions by entering `exit`, and then close the terminal windows.

Solutions for Practice 9: Managing Undo Data

Background: A new version of your application will include several reports based on very long-running queries. Configure your system to support these reports.

1. Use the Undo Advisor to calculate the amount of undo space required to support a report that takes two days to run, on the basis of an analysis period of the last seven days.

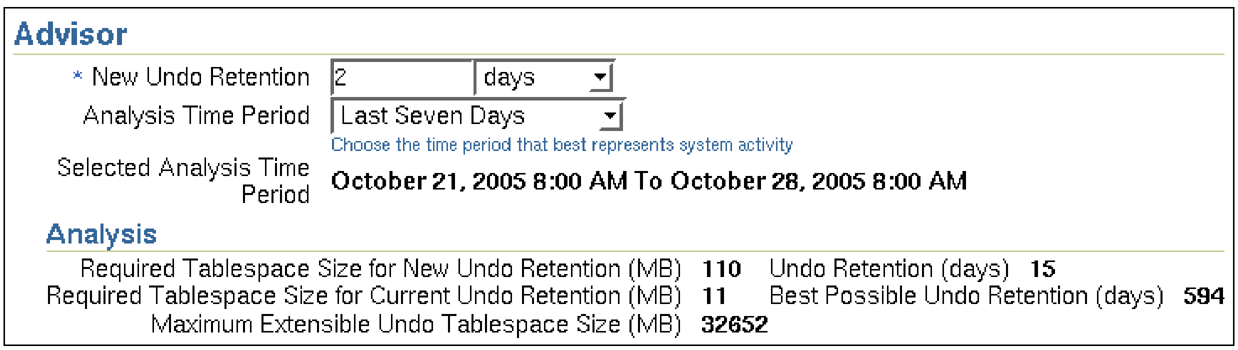
   a) In Enterprise Manager, select **Administration** > Related Links > **Advisor Central**.

   b) Click **Undo Management**.

   c) Click **Undo Advisor**.

   d) In the **New Undo Retention** field, enter `2`. Then select **days** from the drop-down list.

   e) Select Last Seven Days in the **Analysis Time Period** drop-down list. Results of the analysis will be displayed.
   Note: The values that you see are likely to be different from those shown here.



   f) *Question:* What does the analysis recommend as “Required Tablespace Size for New Undo Retention”?

   Answer: 110 MB

   g) Click **OK**.

Solutions for Practice 9: Managing Undo Data (continued)

2. Resize the undo tablespace to support the retention period required by the new reports (or 1 GB, whichever is smaller). Do this by increasing the size of the existing data file.

   a) Click **Edit Undo Tablespace**.

   b) *Question:* What are the two ways to add space to a tablespace?

   Answer: Add a new data file or increase the size of an existing data file.

   c) To increase the size of the existing data file, select the data file, and then click **Edit**.

   d) Enter the new value from step 1 (f) in the **File Size** field. Round off to the nearest 100 MB (not to exceed 1 GB). Make sure the box for "Automatically extend datafile when full" is not selected. Then, click **Continue**.

   e) Click **Show SQL** to confirm that the SQL statement looks similar to this:

```
ALTER DATABASE DATAFILE
'/u01/app/oracle/oradata/ORCL/datafile/o1_mf_undotbs1_1pzfonlk_.dbf'
RESIZE 200M
ALTER DATABASE DATAFILE
'/u01/app/oracle/oradata/ORCL/datafile/o1_mf_undotbs1_1pzfonlk_.dbf'
AUTOEXTEND OFF
```

   f) Click **Return**, then click **Apply** to implement your change.

Solutions for Practice 10: Implementing Oracle Database Security

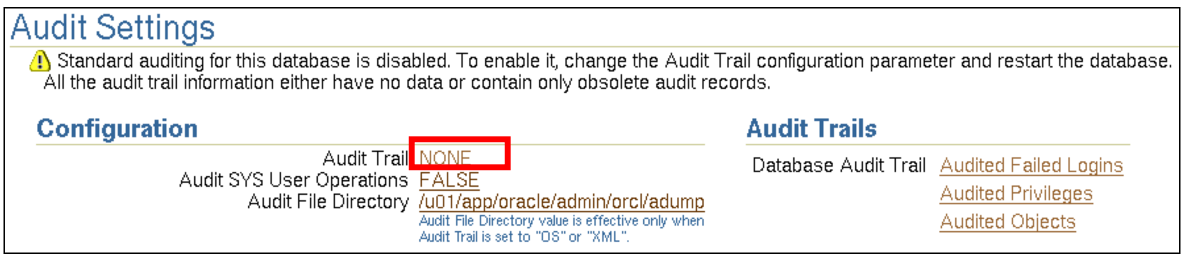
Background: You have just been informed of suspicious activities in the `HR.JOBS` table in your `ORCL` database. All maximum salaries seem to fluctuate in a strange way. You decide to enable standard database auditing and monitor data manipulation language (DML) activities in this table.

Log in as the `DBA1` user (with `oracle` password, connect as `SYSDBA`) and perform the necessary tasks either through Enterprise Manager Database Control or through SQL*Plus. All scripts for this practice are in the `/home/oracle/labs` directory.

1. Use Enterprise Manager to enable database auditing. Set the `AUDIT_TRAIL` parameter to `XML`.

   a) Invoke Enterprise Manager as the `DBA1` user in the `SYSDBA` role for your `ORCL` database.

   b) Select **Administration** > Schema > Users & Privileges > **Audit Settings**.



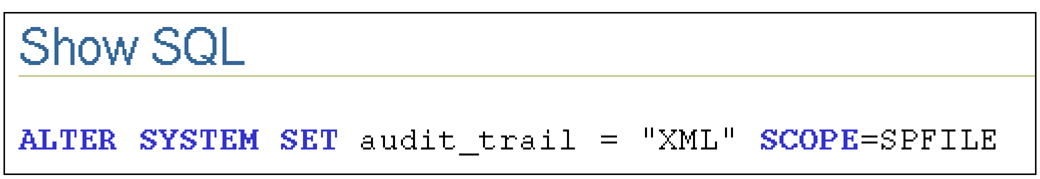
   c) Click the **NONE** link.

   d) On the Initialization Parameters page, click the **SPFile** tab.

   e) Enter `audit` in the **Name** field and then click **Go**.

   f) For the **`audit_trail`** parameter, select the **`XML`** value.

   g) Click **Show SQL**.



   h) Review the statement and then click **Return**.

   i) On the Initialization Parameters page, click **Apply**.

2. Because you changed a static parameter, you must restart the database. Do so by running the `lab_10_02.sh` script.

   a) In a terminal window, enter:

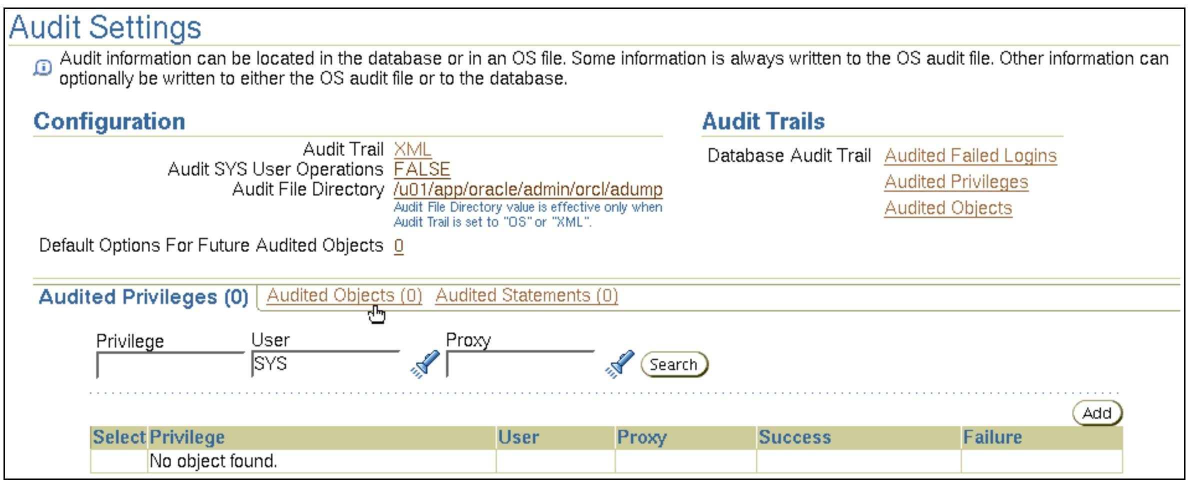
```
cd /home/oracle/labs
./lab_10_02.sh
```

   Continue with the next step when you see that the database is restarted and the script has exited out of SQL*Plus.

3. Back in Enterprise Manager, select `HR.JOBS` as the audited object and `DELETE`, `INSERT`, and `UPDATE` as **Selected Statements**. Gather audit information by session.

   a) Click the Database home page tab to ensure that Enterprise Manager had time to update the status of the database and its agent connections. Because the database has been restarted, you have to log in to EM again as the `DBA1` user.

   b) Then, select **Administration** > Schema > Users & Privileges > **Audit Settings**.

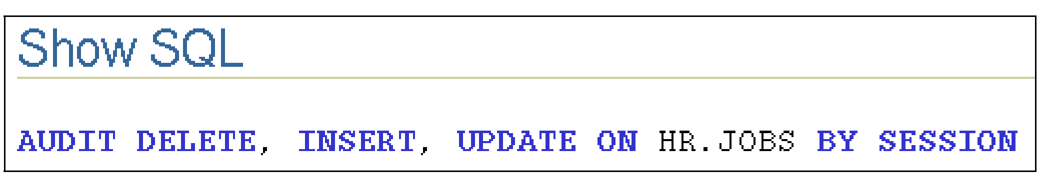


   c) Click the **Audited Objects** tab, and then click the **Add** button.

   d) On the Add Audited Object page, ensure that the **Object Type** is Table, and enter `HR.JOBS` in the **Table** field (or use the flashlight icon to retrieve this table).

   e) Move `DELETE`, `INSERT`, and `UPDATE` into the **Selected Statements** area by double-clicking each of them.

   f) Click **Show SQL**.



g) Review the statement, and then click **Return**.

h) Click **OK** to activate this audit.

4. Provide input for the audit, by executing the `lab_10_04.sh` script. This script creates the `AUDIT_USER` user, connects to SQL*Plus as this user, and multiplies the values in the `MAX_SALARY` column by `10`. Then, the `HR` user connects and divides the column values by `10`. Finally, the `AUDIT_USER` user is dropped again.

a) In a terminal window, enter:

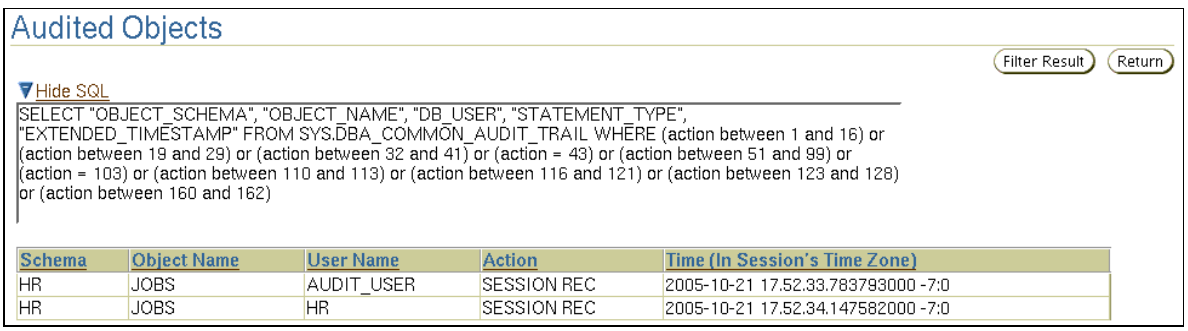
```
cd /home/oracle/labs
./lab_10_04.sh
```

5. In Enterprise Manager, review the audited objects.

a) Select **Administration** > Schema > Users & Privileges > **Audit Settings**.

b) Click **Audited Objects** in the **Audit Trails** area, which is on the right side of the page.

c) On the Audited Objects page, review the collected information, and optionally click **Show SQL**.



Schema	Object Name	User Name	Action	Time (In Session's Time Zone)
HR	JOBS	AUDIT_USER	SESSION REC	2005-10-21 17.52.33.783793000 -7:0
HR	JOBS	HR	SESSION REC	2005-10-21 17.52.34.147582000 -7:0

d) Click **Return**.

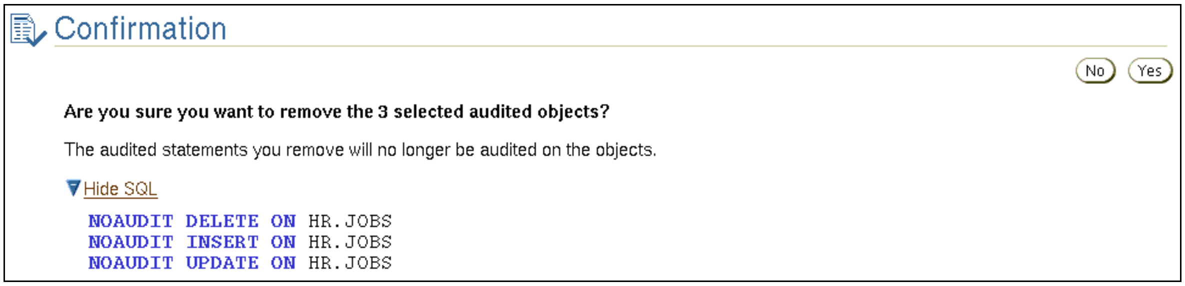
6. Undo your audit settings for `HR.JOBS`, disable database auditing, and then restart the database by using the `lab_10_06.sh` script.

a) On the Audit Settings page, click the **Audited Objects** tab.

b) Enter `HR` as **Schema,** and then click **Search**.

c) Select all three rows, and then click **Remove**.

d) On the Confirmation page, click **Show SQL**.



e) Review the statements, then click **Yes** to confirm your removal.

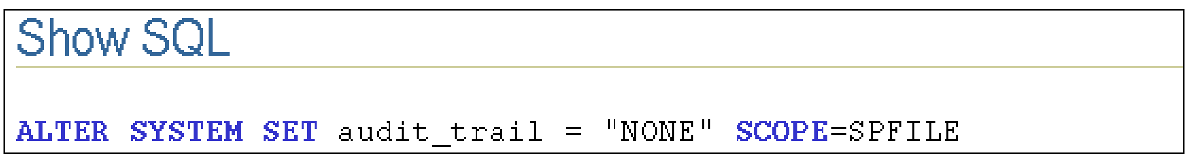
f) On the Audit Settings page, click **XML** in the **Configuration** region.

g) On the Initialization Parameters page, click the **SPFile** tab.

h) On the SPFile page, enter `audit` in the **Name** field, and then click **Go**.

i) For the **`audit_trail`** parameter, select the **`NONE`** value.

j) Click **Show SQL**.



k) Review the statement, and then click **Return**.

l) On the Initialization Parameters page, click **Apply**.

m) Because you changed a static parameter, you must restart the database. Do so by running the `lab_10_06.sh` script. In a terminal window, enter:

```
cd /home/oracle/labs
./lab_10_06.sh
```

7. Maintain your audit trail: Because you are completely finished with this task, delete all audit files from the `/u01/app/oracle/admin/orcl/adump` directory.

a) In a terminal window, enter:

```
cd /u01/app/oracle/admin/orcl/adump
ls
rm -f *
```

b) Close the terminal window.

Solutions for Practice 11: Configuring the Oracle Network Environment

Background: Users need to connect to your `ORCL` database. Work with them to enable connections by using different methods. Ensure that users can use connect-time failover to take advantage of a backup listener.

1. Make a copy of your `listener.ora` and `tnsnames.ora` files. They are in the `$ORACLE_HOME/network/admin` directory.

   a) In an terminal window, enter `cd $ORACLE_HOME/network/admin` to navigate to the `/u01/app/oracle/product/10.2.0/db_1/network/admin` directory.

   b) Enter `cp listener.ora listener.old` to create a copy of the `listener.ora` file.

   c) Enter `cp tnsnames.ora tnsnames.old` to create a copy of the `tnsnames.ora` file.

   d) Enter `ls -l`, if you want to see the copies and their privileges in your directory.

2. Navigate to the Net Services Administration page. Start by clicking the **Listener** link on the Database home page.

   a) Invoke Enterprise Manager as the `DBA1` user in the `SYSDBA` role for your `ORCL` database.

   b) On the Database Instance – Home page, click the **Listener** link in the **General** region.

   c) In the **Related Links** region, click **Net Services Administration**.

3. Modify your local Names Resolution file so that you can connect to another database. Name the connection to a partner's `ORCL` database **`testorcl`**.

   a) On the Net Services Administration page, select **Local Naming** from the Administer drop-down list, and then click **Go**.

   The **Netservices Administration: Host Login** page appears.

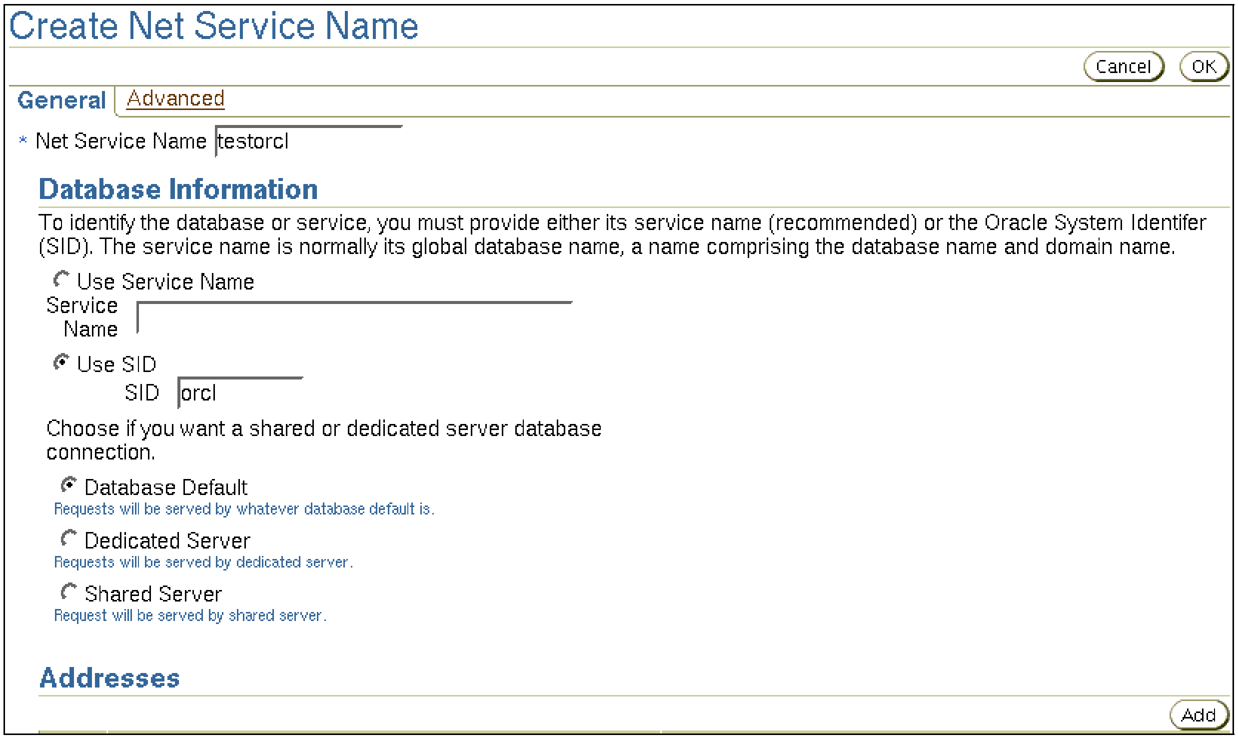
   b) If you previously saved the `oracle` username and `oracle` password as preferred credentials for your host login, then they appear on the screen. If not, enter **`oracle`** as Username and Password, select the **Save as Preferred Credential** check box, and then click **Login**.

   c) On the Local Naming page, click **Create** to enter a new network service name.

   d) Enter **`testorcl`** as Net Service Name.

   e) Select **Use SID**, and enter **`orcl`** as SID.

f) Select **Database Default**.

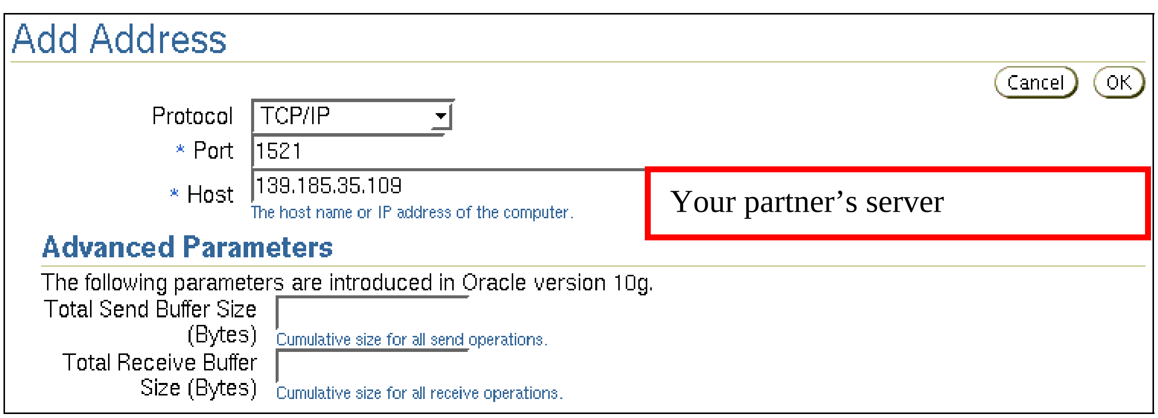


g) Click **Add** in the **Addresses** region.

Click **No** if you are asked, "**Do you want to remember the values you filled in?**"

h) On the Add Address page, select the following values:

- Protocol: **TCP/IP**
- Port: **1521**
- Host: <*Enter the host name or IP address of* ***your partner's computer****. It could be something like* `edrsr9p1.us.oracle.com` *or like* `139.185.35.109`>



Solutions for Practice 11: Configuring the Oracle Network Environment (continued)

i) Click **OK** to return to the Create Net Service Name properties page.

j) Click **OK**.

The Creation Message appears: Net Service “testorcl” created successfully.

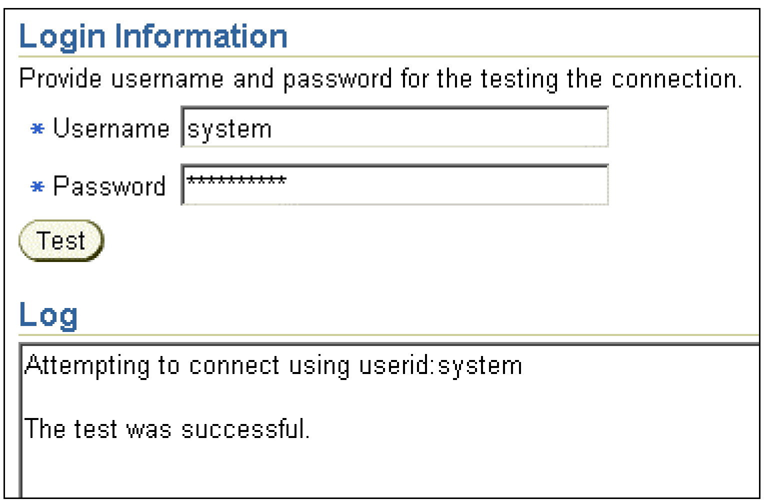
4. In Enterprise Manager, test access to your partner’s `ORCL` database as the **`system`** user with the **`oracle`** password by using the **`testorcl`** Local Naming.

a) Select **`testorcl`** on the Local Naming page, and then click **Test Connection**.

The “Test Connection To Net Service Name: testorcl” appears.

b) Enter **`system`** as Username and **`oracle`** as Password, and then click **Test**.

The Processing page displays status information. Then, it is followed by a success message. *If you receive any errors or warnings, resolve them.*



5. Test your changes to the network configuration by using SQL*Plus or *i*SQL*Plus. Again, use: `system/oracle@testorcl`. To see your partner’s information, select the `instance_name` and `host_name` columns from the `v$instance` table.

a) In an terminal window, enter:

```
sqlplus system/oracle@testorcl
```

The Oracle SQL*Plus window opens. *If you receive any errors or warnings, resolve them.*

b) At the SQL> prompt, enter the following command:

```
select instance_name, host_name from v$instance;
```

```
[oracle@EDRSR30P1 oracle]$ sqlplus system/oracle@testorcl

SQL*Plus: Release 10.2.0.0.0 - Beta on Mon May 16 16:38:46 2005

Copyright (c) 1982, 2004, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.0.0 - Beta
With the Partitioning, OLAP and Data Mining options

SQL> select instance_name, host_name from v$instance;

INSTANCE_NAME
----------------
HOST_NAME
----------------------------------------------------------------
orcl
EDRSR9P1


SQL>
```

You should see your partner's host name.

c) Enter **`exit`** to exit your SQL*Plus session.

6. Create a `LISTENER2` listener to support connect-time failover. Use port 1561 for this listener. First, log out of Enterprise Manager and run the `lab_11_06.sh` script to configure the `NetProperties` file.

   a) Logout of Enterprise Manager.

   b) Run the `lab_11_06.sh` script at the operating system command prompt.

   c) Log in to Enterprise Manager as the `DBA1` user in the `SYSDBA` role for your `ORCL` database.

   d) On the Database Instance – Home page, click the **Listener** link in the **General** region.

   e) In the **Related Links** region, click **Net Services Administration**.

   f) On the Net Services Administration page, select **Listeners** from the **Administer** drop-down list, and then click **Go**. Enter host credentials as `oracle` and `oracle` for username and password, and then click **Login**.

g) On the Listeners page, which gives you an overview of the existing listeners, click the **Create** button.

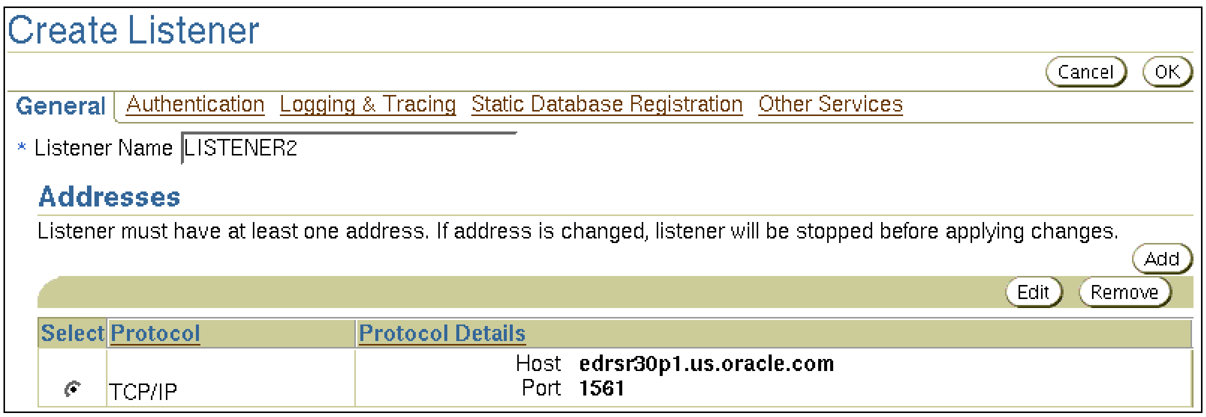
The Create Listener page appears.

h) Enter **`LISTENER2`** as Listener Name, and then click **Add** to add a listener address.

i) Enter or confirm the following values:

- Protocol: `TCP/IP`
- Port: **`1561`**
- Host: <Your computer’s host name; for example, `edrsr30p1.us.oracle.com`>

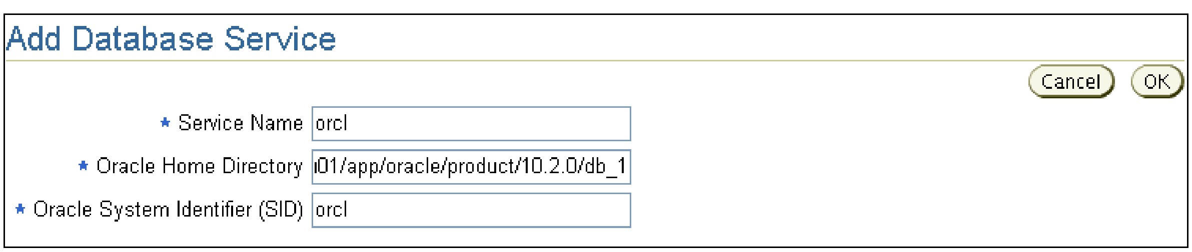
j) Click **OK**.



k) Click the **Static Database Registration** tab, and then click the **Add** button to connect the new listener with your `ORCL` database.

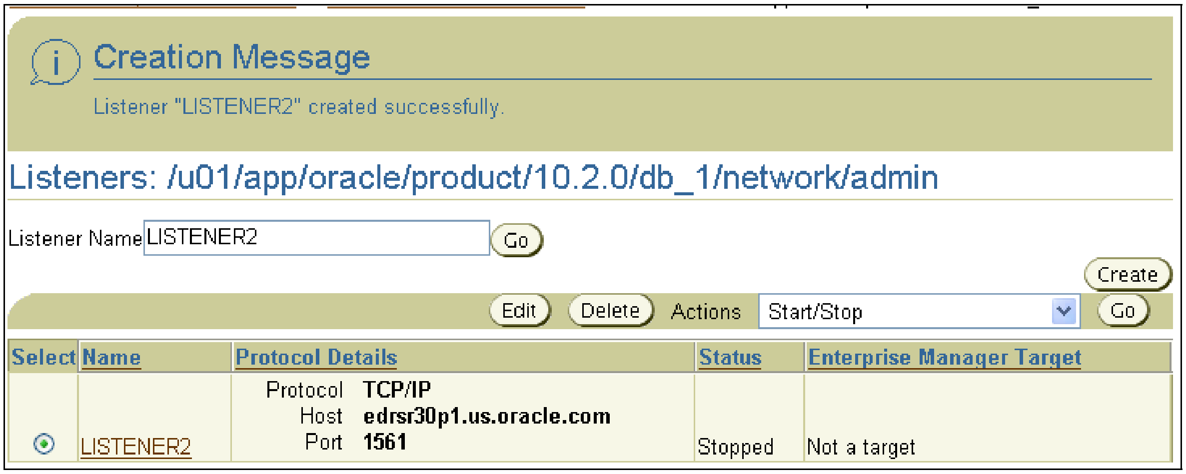
l) Enter the following values:

- Service Name: **`orcl`**
- Oracle Home Directory: **`/u01/app/oracle/product/10.2.0/db_1`**
- Oracle System Identifier (SID): **`orcl`**



m) Click **OK** to add the database service.

n) Click **OK** to create the `LISTENER2` listener.



7. Start the `LISTENER2` listener.

   a) Confirm that the **`LISTENER2`** listener and **Start/Stop** Actions are selected, and then click **Go**.

   b) Click **OK** on the Start/Stop page.

   A confirmation message appears with a **View Details** link.

   c) Optionally, click the **View Details** link, review the listener status information, and use the **Back** icon of your browser to return to the previous page.

Solutions for Practice 12: Proactive Maintenance

Background: You want to proactively monitor your ORCL database so that common problems can be fixed before they affect users. This practice session invents some issues so that you can familiarize yourself with the tools that are available. First, execute scripts to set up your Automatic Database Diagnostic Management (ADDM) environment.

1. Create a new, locally managed tablespace called TBSADDM. Its addm1.dbf data file is 50 MB. Ensure that the TBSADDM tablespace does not use Automatic Segment Space Management (ASSM). Execute the lab_12_01.sh script to perform these tasks.

   a) In a terminal window, enter:

```
cd /home/oracle/labs
./lab_12_01.sh
```

2. Create a new ADDM user, identifed by ADDM. Assign the TBSADDM tablespace as default tablespace. Assign the TEMP tablespace as temporary tablespace. Grant the following roles to the ADDM user: CONNECT, RESOURCE, and DBA. Execute the lab_12_02.sh script to perform these tasks.

   a) In a terminal window, enter:

```
./lab_12_02.sh
```

3. Use the DBMS_ADVISOR package to set the database activity time to 30 minutes. As an ADDM user, drop and create the ADDM table and gather statistics for this table. Create a snapshot in Automatic Workload Repository (AWR). Execute the lab_12_03.sh script to perform these tasks.

   a) In a terminal window, enter:

```
./lab_12_03.sh
```

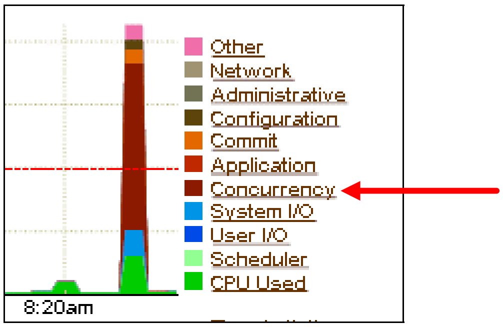
4. Create activity to be analyzed. Execute the lab_12_04.sh script to perform these tasks.

   a) In a terminal window, enter the following. You may have to press [Enter] after you see that eight PL/SQL procedures have completed, in order to see the command prompt again.

```
./lab_12_04.sh
```

Solutions for Practice 12: Proactive Maintenance (continued)

5. In Enterprise Manager, review the Performance page as a user connected as SYSDBA. View performance data in real time with a 15-seconds refresh cycle. After a while, you should see a spike on the "Average Active Sessions" graph. This is your activity to be analyzed. Looking at the graph, you can already determine that this instance is suffering from concurrency problems.



Note: Depending on when you run the workload, you may see differences between your graph and the one provided as a possible solution.

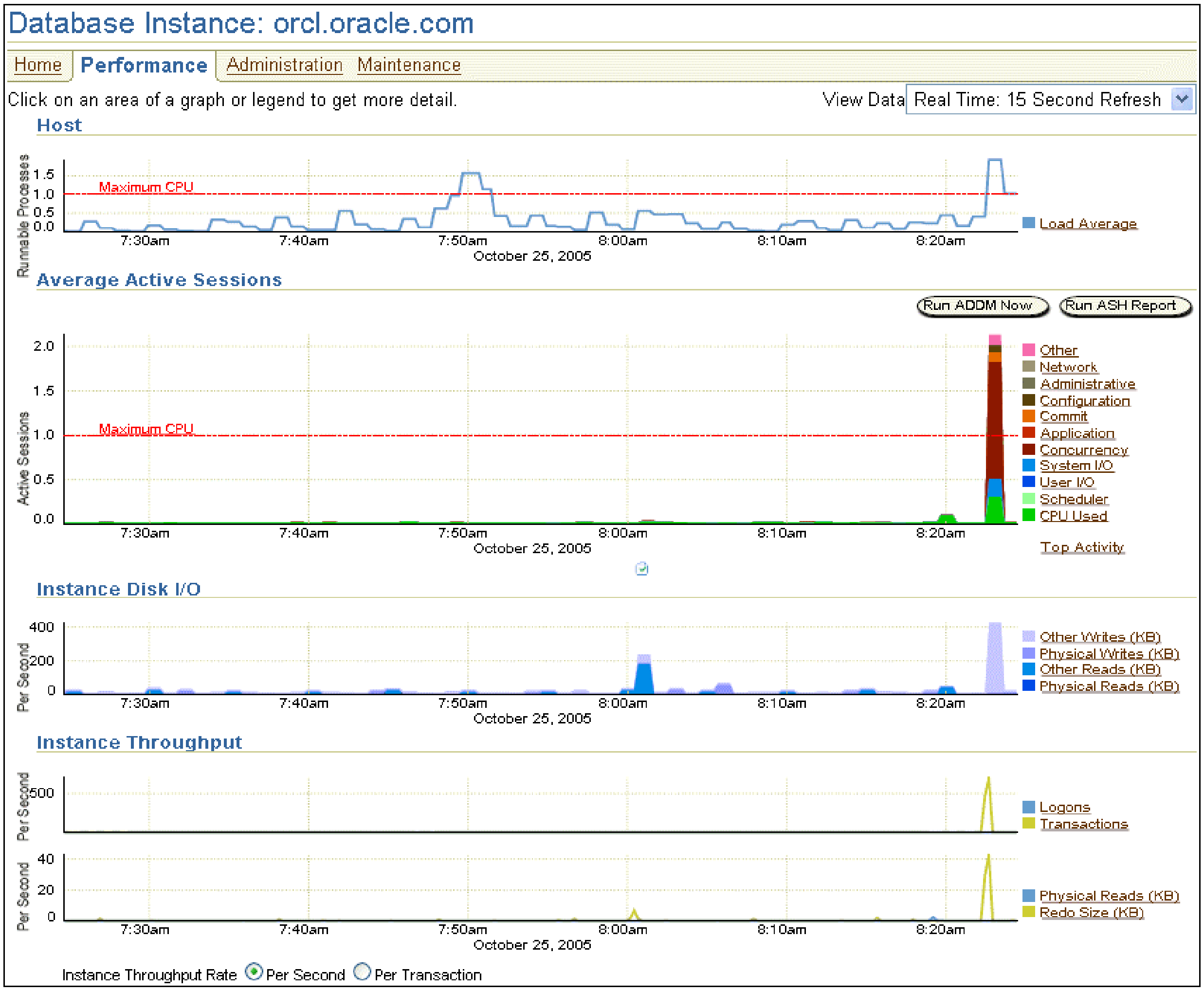
After the spike is finished, execute the `lab_12_05.sh` script. This script forces the creation of a new snapshot and gathers statistics on your ADDM table.

a) Invoke Enterprise Manager as the DBA1 user in the SYSDBA role for your ORCL database.

b) Click the **Performance** tab.

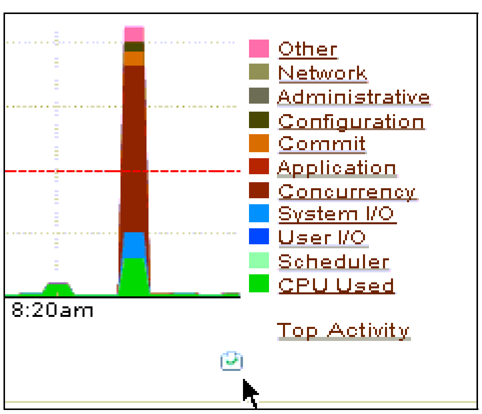
   If this is the first time that you accessed the Performance page, you need to accept the Adobe license agreement. Follow the directions in the pop-up window to accept the agreement.

Solutions for Practice 12: Proactive Maintenance (continued)



c) After the spike has finished, in a terminal window, enter:

```
./lab_12_05.sh
```



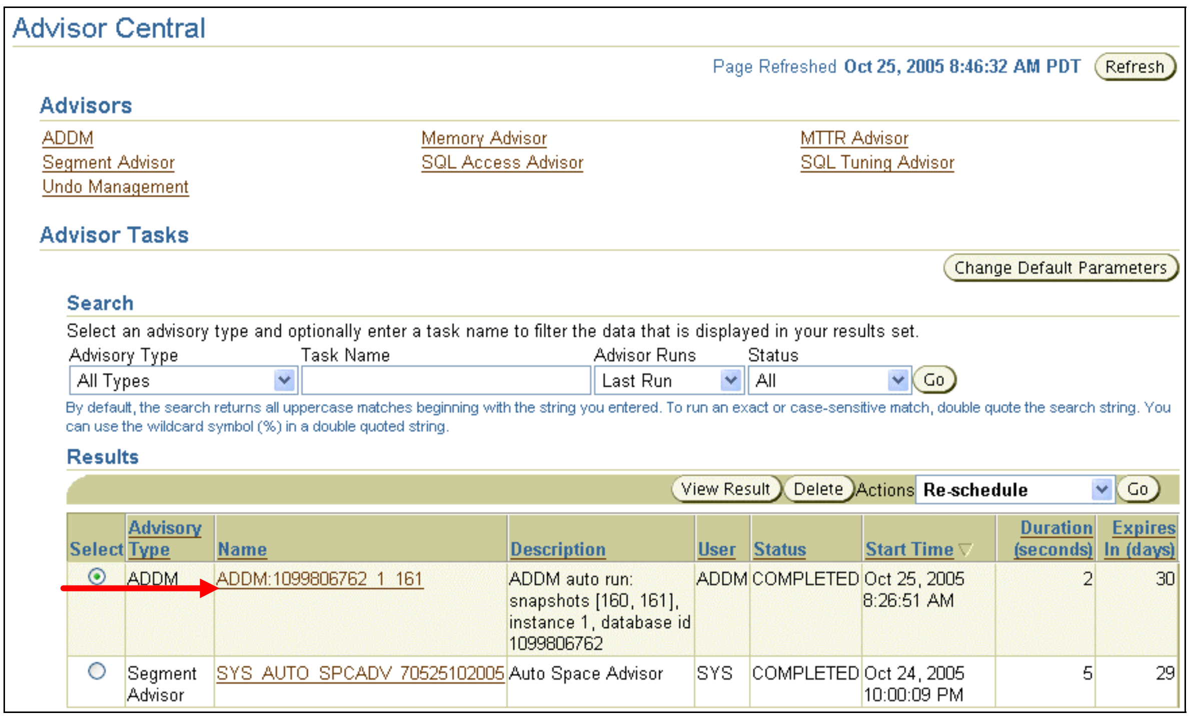
Note: The icon next to the cursor in this screenshot is a shortcut to the Automatic Database Diagnostic Monitor (ADDM) page.

6. Look at the **Performance Analysis** findings in order of their impact. There are several access paths to this information.

   Looking at the Performance Analysis section, you see that the first finding (in the SQL Tuning Recommendations category) has a 100% impact on the system. So your first impluse is to look at this finding in more detail. However, looking at this SQL statement does not yet help you to understand the concurrency problem of your database.

   Research the next finding under Schema Recommendations: **Read and write contention of database blocks was consuming significant database time**. Here you are advised to use the Automatic Segment Space Management (ASSM) feature for your `ADDM` table.

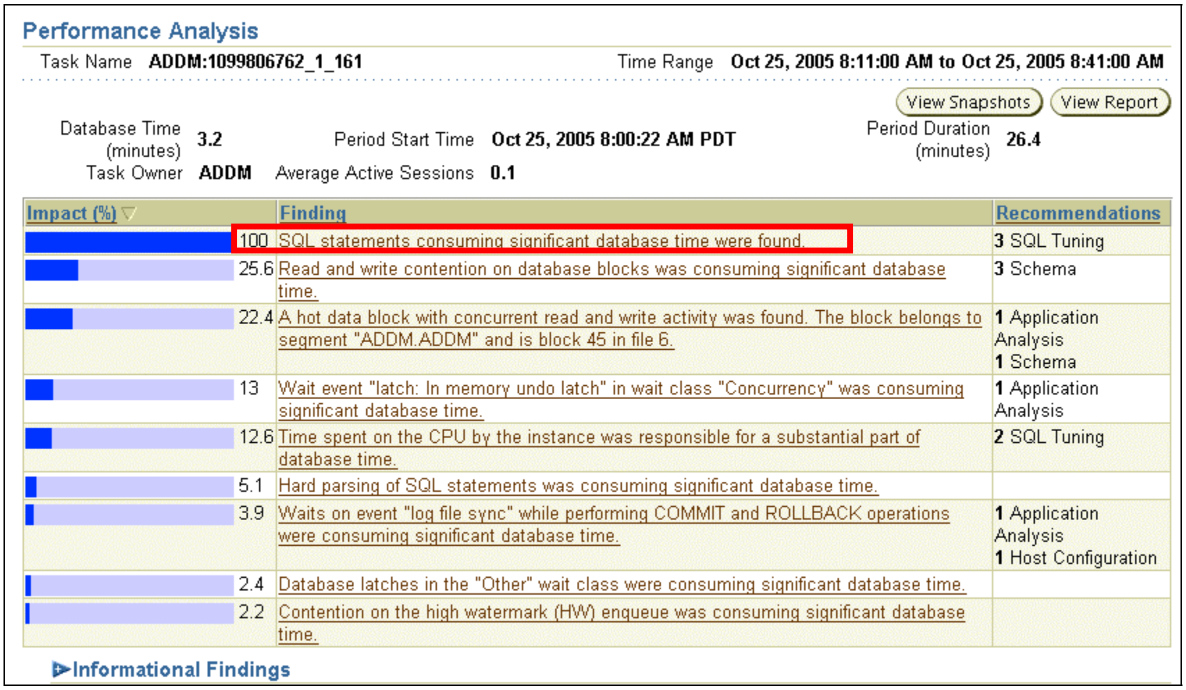
   a) Navigate to the Database home page, and then click **Advisor Central** at the bottom of the page.



   b) Your ADDM task should already be displayed. If not, search for it and display it on this page.

   c) Select the task, and then click the **View Result** button (or alternatively, click the name of the task).

Solutions for Practice 12: Proactive Maintenance (continued)


Performance Analysis

Task Name ADDM:1099806762_1_161 Time Range Oct 25, 2005 8:11:00 AM to Oct 25, 2005 8:41:00 AM

View Snapshots | View Report

Database Time (minutes) 3.2 | Period Start Time Oct 25, 2005 8:00:22 AM PDT | Period Duration (minutes) 26.4

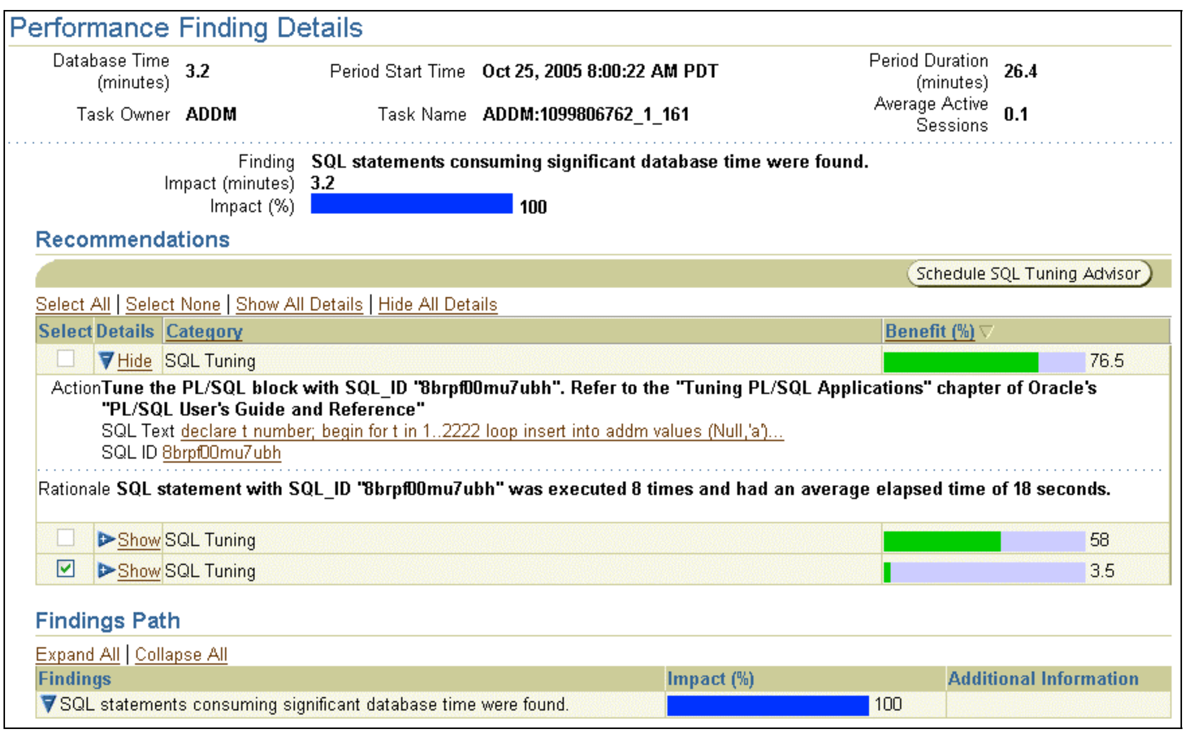
Task Owner ADDM | Average Active Sessions 0.1

Impact (%)	Finding	Recommendations
100	SQL statements consuming significant database time were found.	3 SQL Tuning
25.6	Read and write contention on database blocks was consuming significant database time.	3 Schema
22.4	A hot data block with concurrent read and write activity was found. The block belongs to segment "ADDM.ADDM" and is block 45 in file 6.	1 Application Analysis 1 Schema
13	Wait event "latch: In memory undo latch" in wait class "Concurrency" was consuming significant database time.	1 Application Analysis
12.6	Time spent on the CPU by the instance was responsible for a substantial part of database time.	2 SQL Tuning
5.1	Hard parsing of SQL statements was consuming significant database time.	
3.9	Waits on event "log file sync" while performing COMMIT and ROLLBACK operations were consuming significant database time.	1 Application Analysis 1 Host Configuration
2.4	Database latches in the "Other" wait class were consuming significant database time.	
2.2	Contention on the high watermark (HW) enqueue was consuming significant database time.	

Informational Findings

Looking at the Performance Analysis section, you see that the first finding has a 100% impact on the system. So your first impluse is to look at this finding in more detail.

d) Click the **SQL statements consuming significant database time were found** link.


Performance Finding Details

Database Time (minutes) 3.2 | Period Start Time Oct 25, 2005 8:00:22 AM PDT | Period Duration (minutes) 26.4

Task Owner ADDM | Task Name ADDM:1099806762_1_161 | Average Active Sessions 0.1

Finding SQL statements consuming significant database time were found.

Impact (minutes) 3.2

Impact (%) 100

Recommendations

Schedule SQL Tuning Advisor

Select All | Select None | Show All Details | Hide All Details

Select	Details	Category	Benefit (%)
	Hide	SQL Tuning	76.5
	Show	SQL Tuning	58
☑	Show	SQL Tuning	3.5

Action Tune the PL/SQL block with SQL_ID "8brpf00mu7ubh". Refer to the "Tuning PL/SQL Applications" chapter of Oracle's "PL/SQL User's Guide and Reference"

SQL Text declare t number; begin for t in 1..2222 loop insert into addm values (Null,'a')...

SQL ID 8brpf00mu7ubh

Rationale SQL statement with SQL_ID "8brpf00mu7ubh" was executed 8 times and had an average elapsed time of 18 seconds.

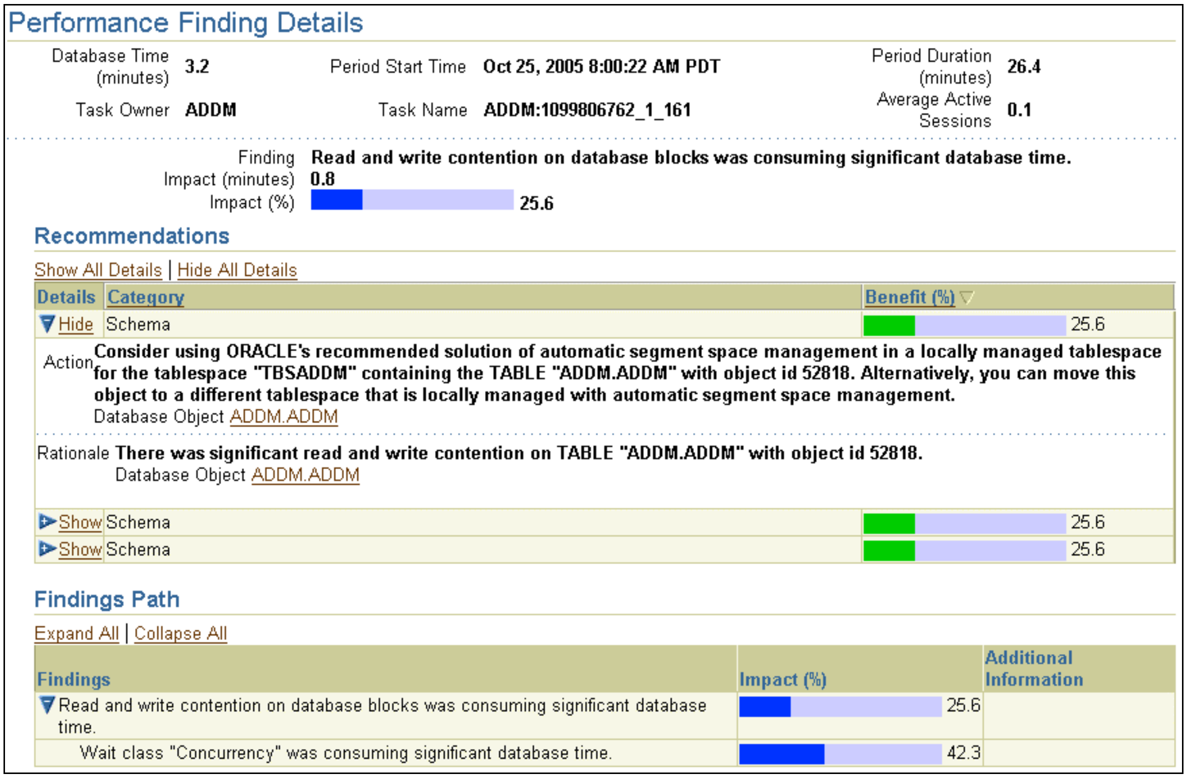
Findings Path

Expand All | Collapse All

Findings	Impact (%)	Additional Information
SQL statements consuming significant database time were found.	100	

e) Review the recommendations on the Performance Finding Details page. However, looking at this SQL statement does not yet help you to understand the concurrency problem of your database. Click the **Back** icon in your Web browser.

f) On the Automatic Database Diagnostic Monitor (ADDM) page, click the **Read and write contention of database blocks was consuming significant database time** link . This finding appears as type **Schema** under the **Recommendations** heading.



g) You are advised to use the Automatic Segment Space Management feature for your `ADDM` table.

7. To implement the recommendation, you must re-create the object. Create a new, locally managed tablespace, called `TBSADDM2` with a 50 MB datafile, called `addm2_1.dbf`. Ensure that the `TBSADDM2` tablespace uses the the Automatic Segment Space Management feature. Then, execute the `lab_12_07.sh` script to drop the `ADDM` table, to re-create it in the new tablespace, to gather statistics and to take a new snapshot.

   a) In Enterprise Manager, select **Administration** > Database Administration > Storage > **Tablespaces**.

   b) Click **Create**.

   c) Enter `TBSADDM2` as the tablespace name, and verify that **Extent Management** is **Locally Managed**, **Type** is **Permanent**, **Status** is **Read Write**, and **Use Bigfile tablespace** is <u>not</u> selected.

d) Click **Add** in the **Datafiles** region.

e) On the Add Datafile page, enter `addm2_1.dbf` for **File Name**, and `50 MB` as **File Size**.

f) Click **Continue**.

g) Click the **Storage** tab, and verify that **Extent Allocation** is **Automatic**, **Segment Space Management** is **Automatic**, and **Logging** is enabled.

h) Click the **General** tab.

i) Click **Show SQL** to see the SQL that will be run, and then click **Return**.

j) Click **OK**, and a successful Update Message appears.

k) In a terminal window, enter:

```
./lab_12_07.sh
```

8. Execute your workload again. (The `lab_12_08.sh` script is identical to the `lab_12_04.sh` script.)

a) In a terminal window, enter the following. You may have to press [Enter] after you see that eight PL/SQL procedures have completed, in order to see the command prompt again.

```
./lab_12_08.sh
```

9. In Enterprise Manager, review the Performance page as a user connected as `SYSDBA`. View performance data in real time with a 15-seconds refresh cycle. After a while, you should see a spike on the “Average Active Sessions” graph.

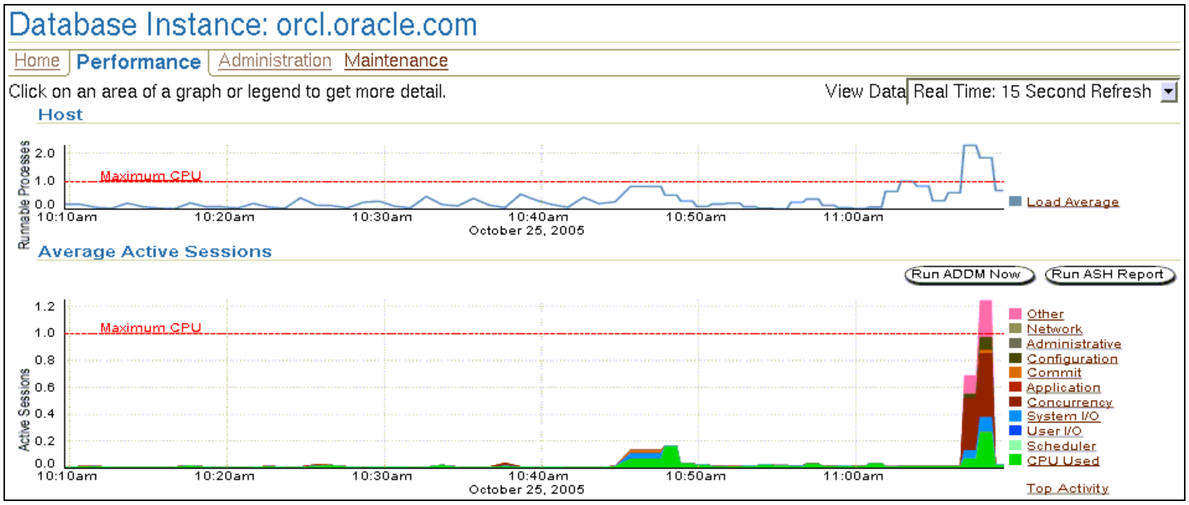
Note: Depending on when you run the workload, you may see differences between your graph and the one provided as a possible solution.

After the spike is finished, execute the `lab_12_09.sh` script. (The `lab_12_09.sh` script is identical to the `lab_12_05.sh` script.) This script forces the creation of a new snapshot and gathers statistics on your `ADDM` table.

a) Invoke Enterprise Manager as the `DBA1` user in the `SYSDBA` role for your `ORCL` database.

b) Click on the **Performance** tabbed page.

Solutions for Practice 12: Proactive Maintenance (continued)



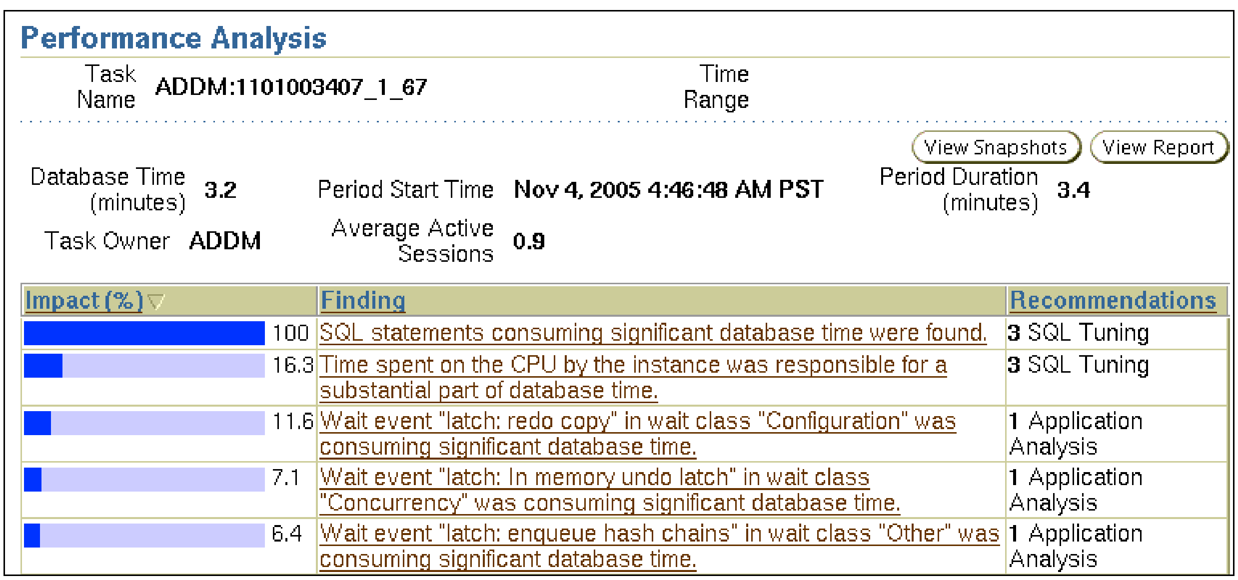
c) After the spike is finished, enter the following in a terminal window:

```
./lab_12_09.sh
```

10. Review the Performance Analysis on the Database home page.

a) Navigate to the Database home page, and then click **Advisor Central** at the bottom of the page.

b) Click the topmost ADDM task name.



Performance Analysis

Task Name ADDM:1101003407_1_67 Time Range

View Snapshots | View Report

Database Time (minutes) 3.2 | Period Start Time Nov 4, 2005 4:46:48 AM PST | Period Duration (minutes) 3.4

Task Owner ADDM | Average Active Sessions 0.9

Impact (%)	Finding	Recommendations
100	SQL statements consuming significant database time were found.	3 SQL Tuning
16.3	Time spent on the CPU by the instance was responsible for a substantial part of database time.	3 SQL Tuning
11.6	Wait event "latch: redo copy" in wait class "Configuration" was consuming significant database time.	1 Application Analysis
7.1	Wait event "latch: In memory undo latch" in wait class "Concurrency" was consuming significant database time.	1 Application Analysis
6.4	Wait event "latch: enqueue hash chains" in wait class "Other" was consuming significant database time.	1 Application Analysis

You see that there are no longer any schema-related recommendations. By moving the `ADDM` table to the locally managed `TBSADDM2` tablespace, which uses the Automatic Autoextend Segment feature, you obviously fixed the root cause of this problem.

Solutions for Practice 12: Proactive Maintenance (continued)

11. To not affect other practice session, execute the `lab_12_11.sh` script to clean up your environment.

    a) In a terminal window, enter:

```
./lab_12_11.sh
```

Solutions for Practice 13: Performance Management

Background: Users are complaining about slower-than-normal performance for operations involving the human resources and order-entry applications. When you question other members of the DBA staff, you find that maintenance was recently performed on some of the tables belonging to the `HR` schema. You need to troubleshoot and make changes as appropriate to resolve the performance problems. SQL script files are provided for you in the `/home/oracle/labs` directory. Other directories are individually named.

1. Log in to SQL*Plus as the `DBA1` user and perform maintenance on tables in the `HR` schema by running the `lab_13_01.sql` script.

   a) Connect to the database as the `DBA1` user and perform some maintenance on the `employees` table by running the `lab_13_01.sql` script.

```
$ cd labs
$ sqlplus dba1/oracle
SQL> @lab_13_01.sql
```

2. You get calls from HR application users saying that a particular query is taking longer than normal to execute. The query is in the `lab_13_02.sql` script. As the HR user, run it.

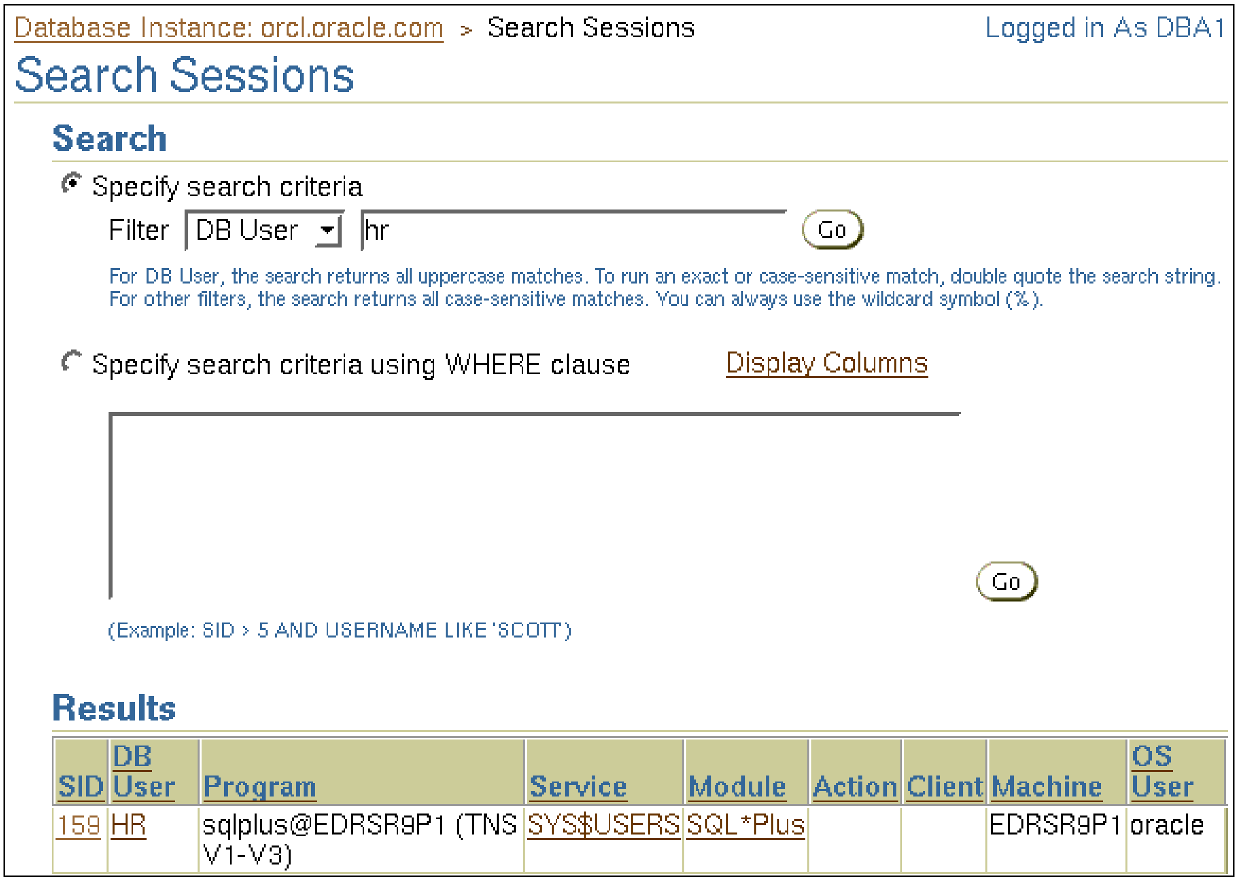
   a) Enter the following in SQL*Plus , while logged in as the HR user:

```
SQL> @lab_13_02.sql
```

3. Using Enterprise Manager, locate the `HR` session in which the above statement was just executed, and view the execution plan for that statement.

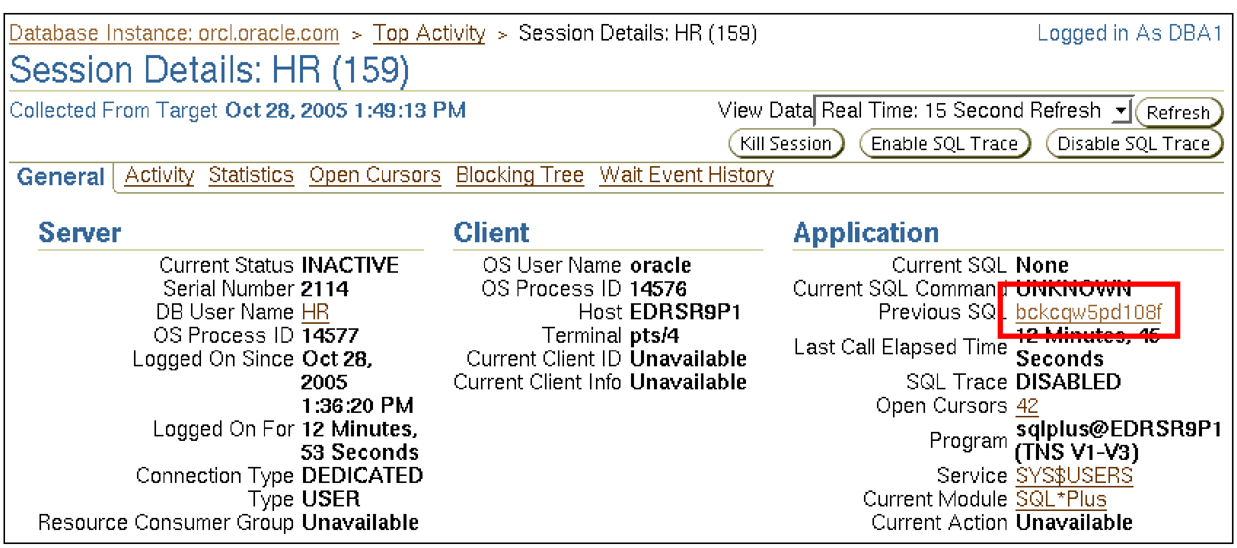
   a) In Enterprise Manager, click **Search Sessions** on the Performance tabbed page.

b) Change the Filter for the search criteria to “DB User,” enter `HR` in the field to the right of that, and then click **Go**.



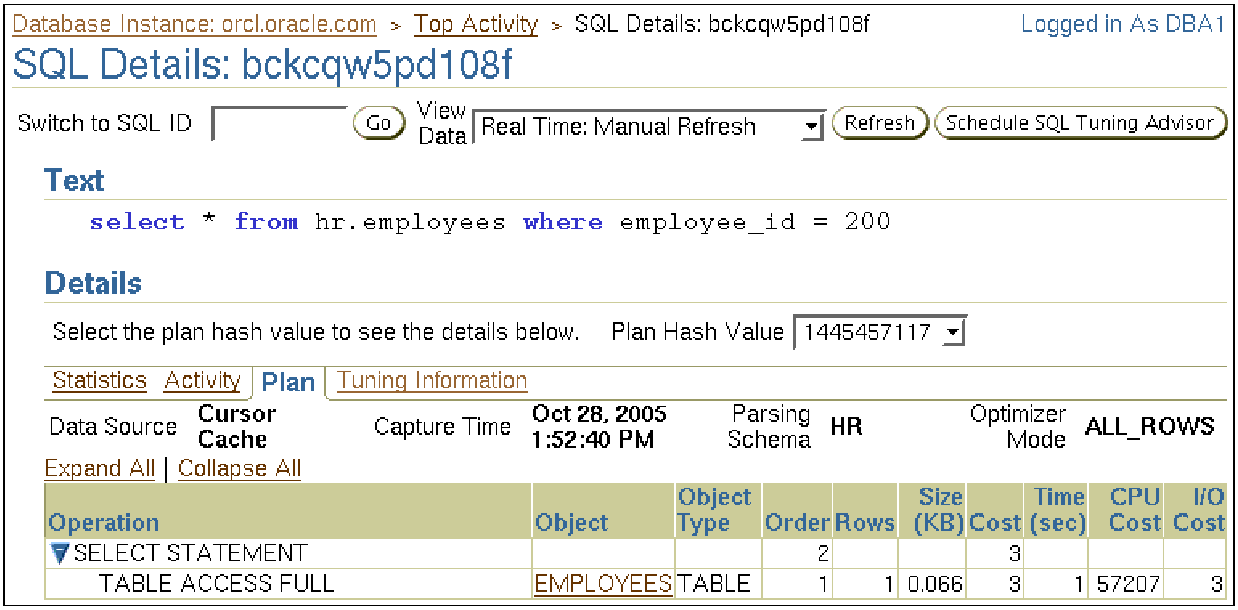
c) Click the **SID** number in the **Results** listing.

d) You now see the Session Details page for this session. Click the hash value link to the right of the **Previous SQL** label.



Solutions for Practice 13: Performance Management (continued)

e) That shows you the SQL Details page for the last SQL statement executed by that session, which is the one in question. Click the **Plan** tab to see the execution plan for the query.



f) You see in the Operation column that this query is doing a full table scan (TABLE ACCESS FULL). Because you know that the query's condition is an equality comparison on the primary key (`EMPLOYEE_ID`), you decide to investigate the status of the primary key index.

4. Using Enterprise Manager, check to see the status of the `EMPLOYEE` table's index on `EMPLOYEE_ID`. See if it is `VALID`.

   a) Select **Administration** > Schema > Database Objects > **Indexes**.

   b) Select Table Name as the **Search By** value.

   c) Enter `HR` as **Schema Name**.

   d) Enter `employees` for **Object Name**.

   e) Click **Go**, and the list of six indexes appears.

   f) Click the index named `EMP_EMP_ID_PK`.

   g) Click the index link and note that the status of the index is `UNUSABLE`.

5. Now that you have seen one index with a non-`VALID` status, you decide to check all indexes. Using SQL*Plus, as the `HR` user, find out which `HR` schema indexes do not have `STATUS` of `VALID`. To do this, you can query a data dictionary view with a condition on the `STATUS` column.

   a) Go to the SQL*Plus session where you are still logged in as the `HR` user, and run this query:

```
SQL> select index_name, table_name, status
     from user_indexes where status <> ‘VALID’;
```

   b) See that the output lists six indexes, all on the `EMPLOYEES` table.

```
SQL> select index_name, table_name, status
  2  from user_indexes where status <> 'VALID';

INDEX_NAME                     TABLE_NAME                     STATUS
------------------------------ ------------------------------ --------
EMP_EMAIL_UK                   EMPLOYEES                      UNUSABLE
EMP_EMP_ID_PK                  EMPLOYEES                      UNUSABLE
EMP_DEPARTMENT_IX              EMPLOYEES                      UNUSABLE
EMP_JOB_IX                     EMPLOYEES                      UNUSABLE
EMP_MANAGER_IX                 EMPLOYEES                      UNUSABLE
EMP_NAME_IX                    EMPLOYEES                      UNUSABLE

6 rows selected.

SQL>
```

6. Using Enterprise Manager, reorganize all the indexes in the `HR` schema that are marked as `UNUSABLE`.

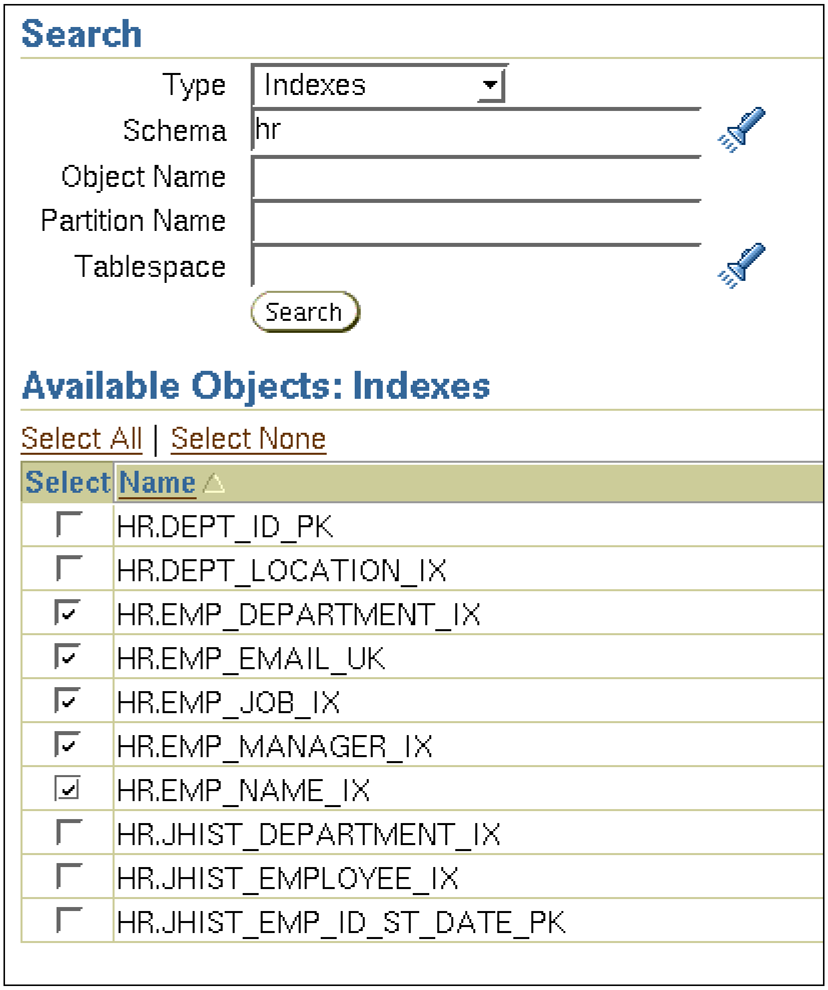
   a) In Enterprise Manager, on the page displaying the `EMP_EMP_ID_PK` index, select **Reorganize** in the **Actions** list, and then click **Go**.

   b) Click **Add**, to add each of the other five indexes to the reorganization operation.

   c) In the Add screen, choose Indexes for the **Type** drop-down list, and enter `hr` in the **Schema** field.

   d) Click **Search**.

e) Select the five other indexes whose names start with "`EMP_`."



f) Click **OK**.

g) Click **Next**.

h) Keep all the default settings for **Options**, and then click **Next**.

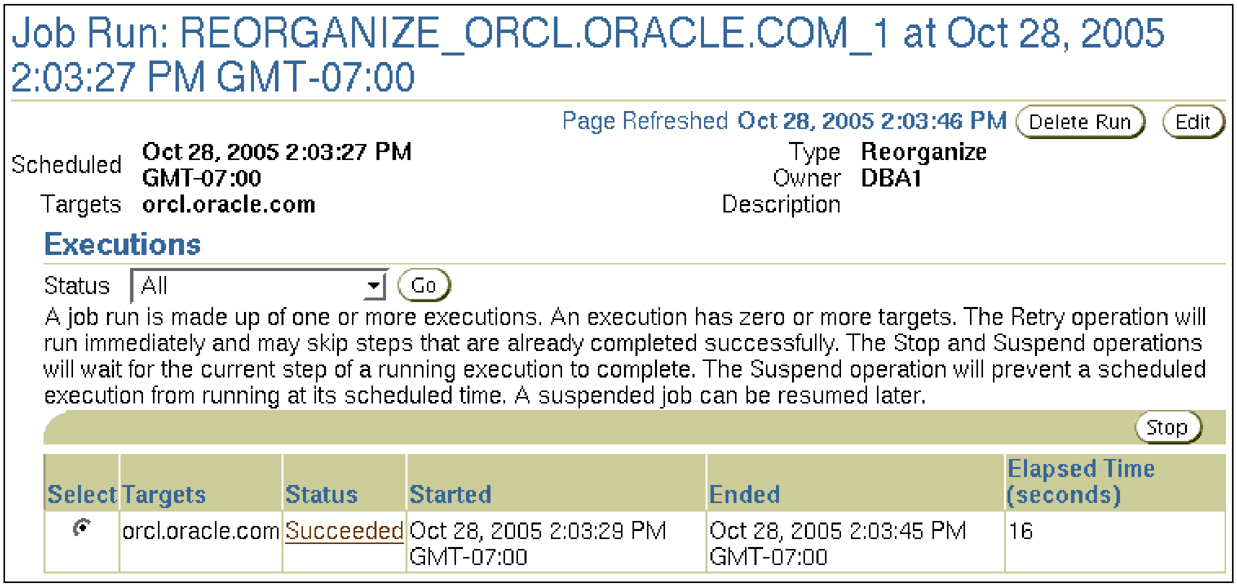
i) Note that there are no problems reported on **Impact Report**, and then click **Next**.

j) On the Schedule page, enter `oracle` and `oracle` for Username and Password under **Host Credentials**.

k) Click **Next**.

l) On the Review page, click **Submit Job**.

m) After the Confirmation page appears, click the job name to see its status. Click **Reload** on your browser until you see the job has succeeded.

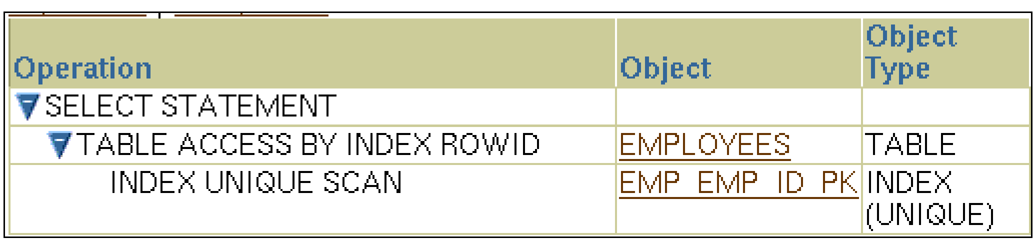


7. Return to the SQL*Plus session where the `HR` user is logged in, and run the `lab_13_07.sql` script to execute the same kind of query. Then, repeat the steps to see the plan of the last SQL statement executed by this session, to see if the plan has changed.

a) Enter the following at the SQL*Plus prompt:

```
SQL> @lab_13_07.sql
```

b) Repeat the tasks listed in step 3. Note that the plan now uses the index.



Operation	Object	Object Type
SELECT STATEMENT		
TABLE ACCESS BY INDEX ROWID	EMPLOYEES	TABLE
INDEX UNIQUE SCAN	EMP_EMP_ID_PK	INDEX (UNIQUE)

c) Quit the SQL*Plus session.

8. What is the difference in execution plans, and why?

Answer: The statement execution uses a unique index scan instead of a full table scan, because the index is usable after your index reorganization.

Solutions for Practice 13: Performance Management (continued)

9. Simulate a working load on your instance by running the `lab_13_09.sql` script as the `DBA1` user. Please note the SID value for task 10.

   Possible answer: 147   (Your answer is most likely a different one).

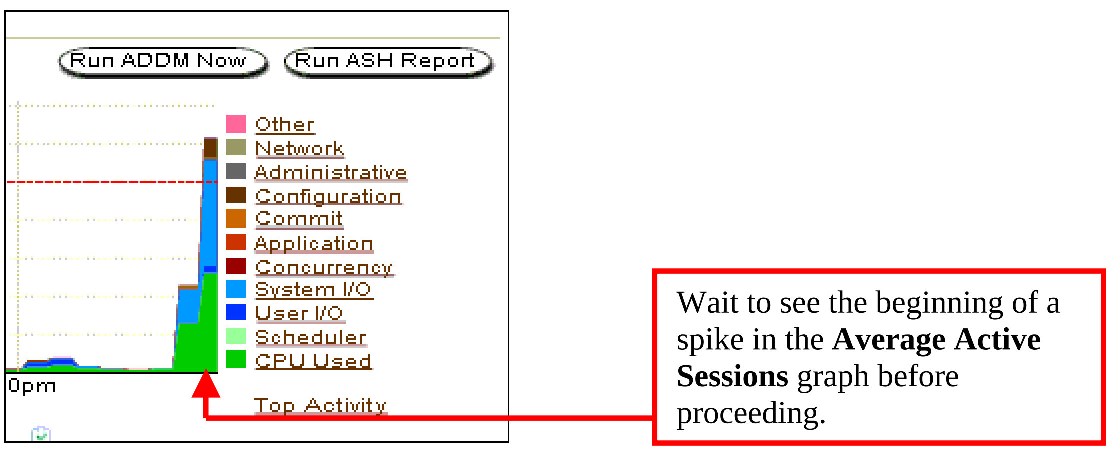
   This script takes about 20 minutes to complete. So, run it in a separate terminal window and continue with this practice exercise while it runs.

   Note: Because this script generates a fairly heavy load in terms of CPU and disk I/O, you will notice that response time for Database Control is slower.

```
$ sqlplus dba1/oracle
SQL> @lab_13_09.sql
```

   a) In Enterprise Manager, navigate to the Performance page, and investigate system performance.

   b) You may need to wait a minute or two to see the effects of the load generation script appear on the graphs.



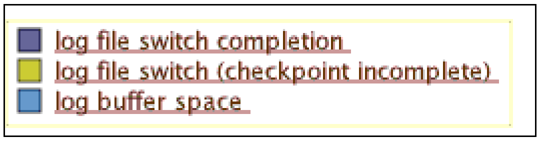
   Question 1: In the **Average Active Sessions** graph, which are the two main categories that active sessions are waiting for?

   Answer: System I/O and CPU Used

Solutions for Practice 13: Performance Management (continued)

Question 2: In the Configuration category of waits, what is one of the contributors to the wait time? Click Configuration to see the graph.

Answer: Any one of these:



Question 3: Click Back, and then click Physical Writes on the Instance Disk I/O graph. Determine which process is doing the most writing to the disk.

Answer: `DBW0`

c) Click **Back**.

d) Click **Top Activity** in the **Additional Monitoring Links** region.

e) Click the **SQL ID** of the first SQL statement listed in the **Top SQL** region.

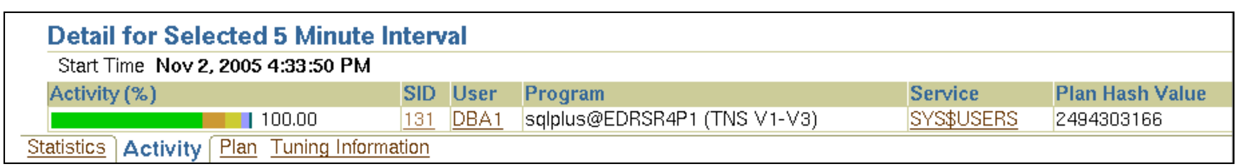
f) See the first SQL statement.

Question 4: What SQL statement is causing the most waits?

Answer: `delete from sh.sales_copy`

10. Kill the session that is generating the load. Use the session ID recorded in step 9. The session ID should be listed at the top of the list on the right side of the page, under the **Top Sessions** region.

    a) Click the SID number for the session ID recorded earlier. This is found under the heading **Detail for Selected 5 Minute Interval.**



Detail for Selected 5 Minute Interval

Start Time Nov 2, 2005 4:33:50 PM

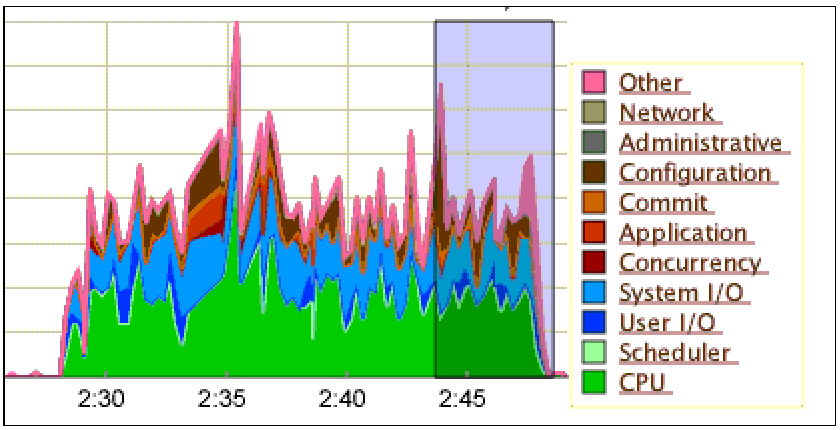
Activity (%)	SID	User	Program	Service	Plan Hash Value
100.00	131	DBA1	sqlplus@EDRSR4P1 (TNS V1-V3)	SYS$USERS	2494303166

Statistics | Activity | Plan | Tuning Information

b) On the Session Details page, click **Kill Session**, and then click **Yes** to confirm.

Note: If you remain on this Session Details page long enough for a few automatic refreshes to be done, you may see the following warning, which means you are attempting to refresh information about a session that's already been killed. You can ignore this warning.

c) Click **Top Activity** in the navigation history at the top of the page. Note that the session activity in the database has declined considerably.



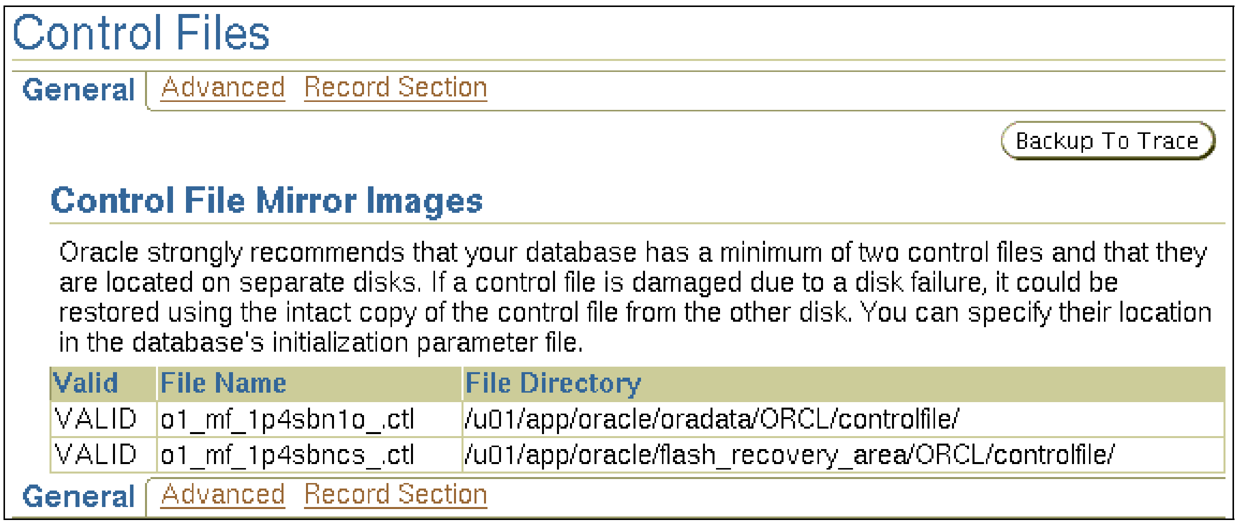
Solutions for Practice 14: Backup and Recovery Concepts

Background: Your `ORCL` database is ready to move from test or development into production. Configure your database to reduce the chances of failure or data loss.

1. Verify that you have two control files to ensure redundancy.

   a) Invoke Enterprise Manager as the `DBA1` user in the `SYSDBA` role for your `ORCL` database.

   b) Select **Administration** > Database Administration > Storage > **Control Files**.



   c) *Question 1:* How would you add another control file if you needed to?

   Answer: Adding a control file is a manual operation. To perform this, you must:

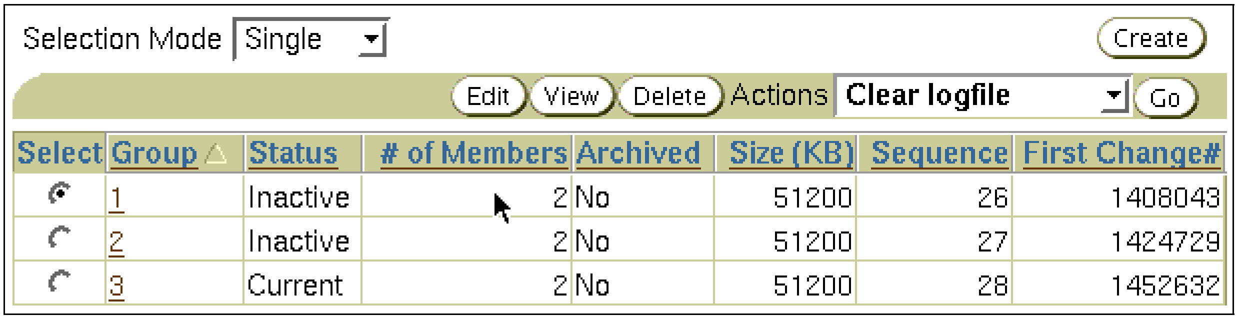
   - Shut down the database
   - Use the operating system to copy an existing control file to the location where you want your new file to be
   - Start the database by using Enterprise Manager. Unlike a normal startup, you would use Advanced Options to select a different startup mode. Select "Start the instance" to leave the instance in the `NOMOUNT` state.
   - Edit the `CONTROL_FILES` initialization parameter to point to the new control file
   - Continue the `STARTUP` database operation until the database is in an open state

   Note: This answer does not apply to an OMF database, as the control files in that case would have to all be recreated.

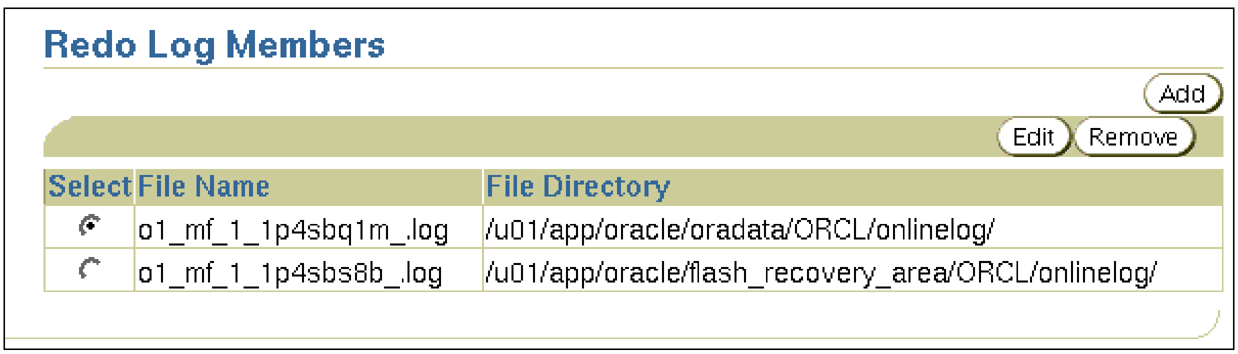
2. Check how many members each redo log group has. Ensure that there are at least two redo log members in each group. In what directory or directories are the redo log files stored?

Solutions for Practice 14: Backup and Recovery Concepts (continued)

a) Select **Administration** > Database Administration > Storage > **Redo Log Groups**, and note how many members are in the “# of Members” column. There should be two per group.



b) Select the first group, and then click **Edit** to see the member file names. Note that one member is in directories under the `oradata` directory, and another is in the Flash Recovery Area. Click the browser’s **Back** button.



Note: In a production database, you want to ensure that the two members are on different hard drives, preferably with different disk controllers, to minimize the risk of any single hardware failure destroying an entire log group.

3. You notice that, for each log group, the `Archived` column has a value of `No`. This means that your database is not retaining copies of redo logs to use for database recovery, and in the event of a failure, you will lose all data since your last backup. Place your database in `ARCHIVELOG` mode, so that redo logs will be archived.

a) Create a new directory `/u01/app/oracle/archive` as the destination for the redo log files.

- In a terminal window, enter **`cd /u01/app/oracle`**.

- Then, enter **`mkdir archive`**.

- Optionally, enter **`ls -l`** to view your new directory and its OS permissions.

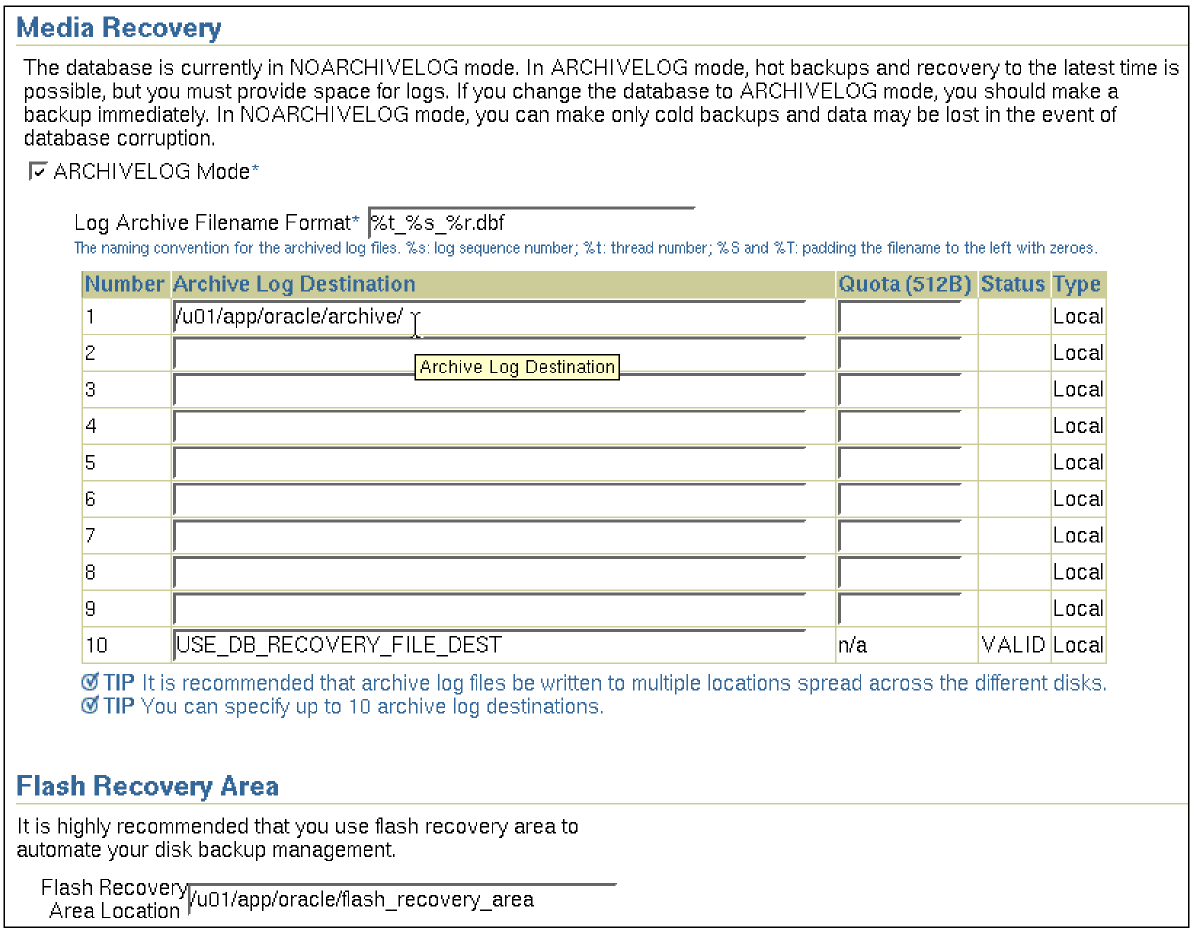
Solutions for Practice 14: Backup and Recovery Concepts (continued)

b) In Enterprise Manager, select **Maintenance** > High Availability > Backup/Recovery Settings > **Recovery Settings**.

c) In the **Media Recovery** region, select the **ARCHIVELOG Mode** check box.

d) Verify that Log Archive Filename Format contains `%t`, `%s`, and `%r`.

4. Configure redundant archive log destinations—one to the Flash Recovery Area and the other to `/u01/app/oracle/archive`.



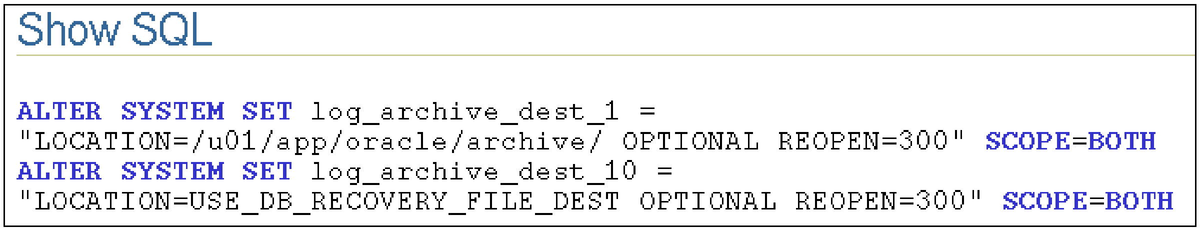
Notice that the database is preconfigured to save archived logs to the Flash Recovery Area by default (Archive Log Destination 10). Add an additional Archive Log Destination so that you will have redundant copies of your log files.

a) In **Archive Log Destination** number 1, enter `/u01/app/oracle/archive/`. The directory path should end with a slash. Leave **Quota** blank.

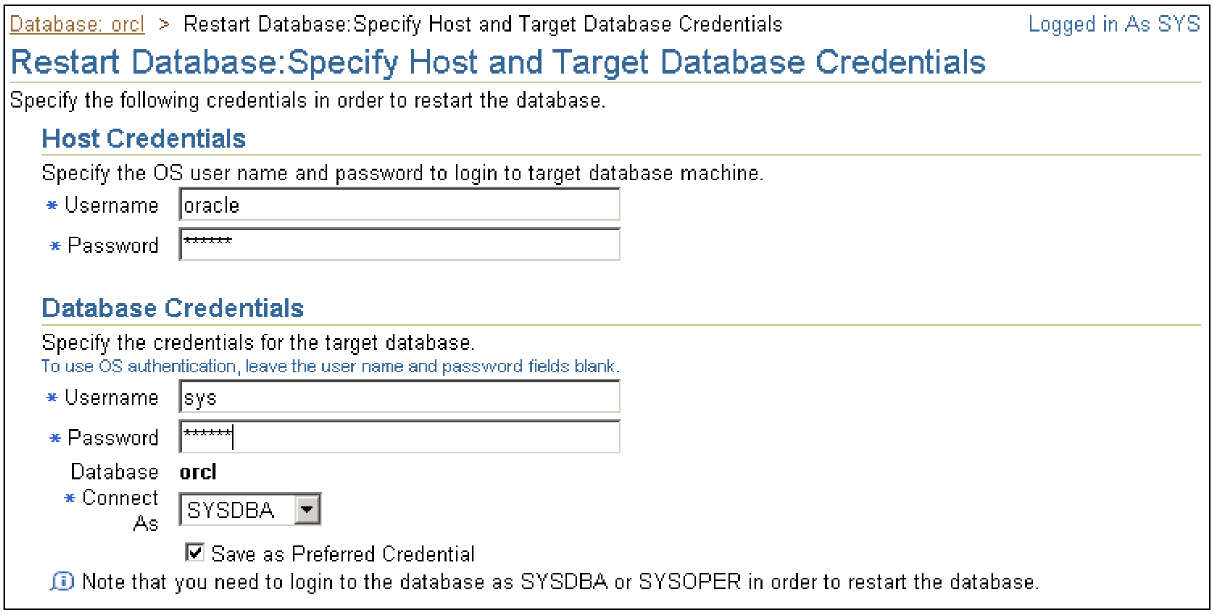
Note: You must create the directory, if it does not already exist. You already did this in step (3).

Solutions for Practice 14: Backup and Recovery Concepts (continued)

b) Optionally, click **Show SQL**, review the statements, and then click **Return**.



c) Click **Apply**. When prompted whether you want to restart the database now, click **Yes**.



d) Enter the credentials to restart the database, and then click **OK**.

e) When asked to confirm, click **Yes** again.

Now that your database is in `ARCHIVELOG` mode, it will continually archive a copy of each online redo log file before reusing it for additional redo data.

Note: Remember that this consumes space on the disk and that you must regularly back up older archive logs to some other storage.

Solutions for Practice 15: Performing Database Backups

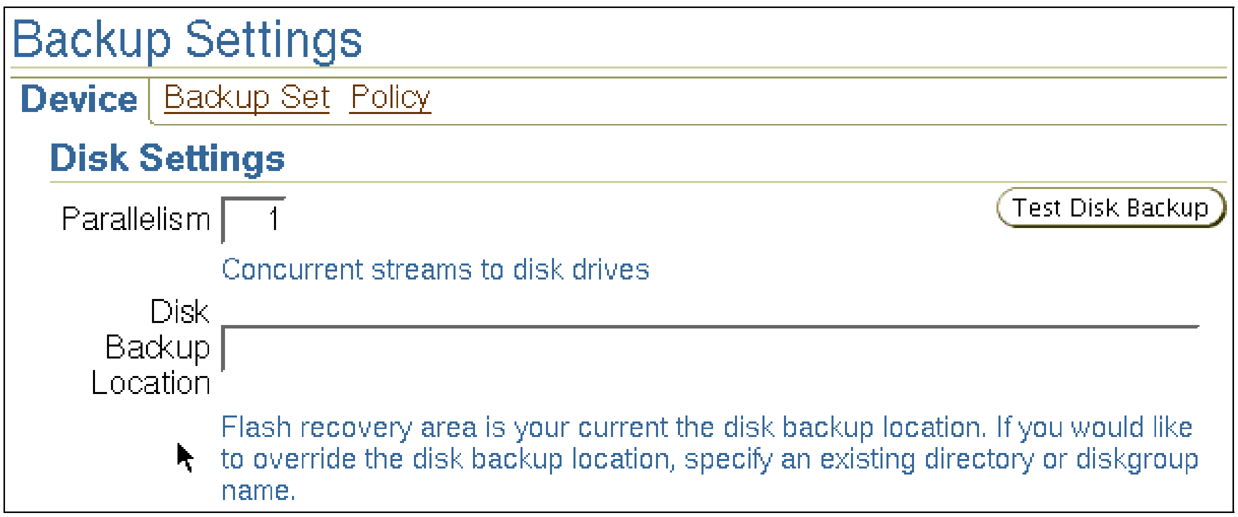
Background: Your database is ready to move from development and test into production. Ensure that your database is configured so that recovery is possible without loss of data.

1. What is the difference between a backup set and an image copy?

   Answer: A backup set contains data and archive log files packed in an Oracle proprietary format. Files must be extracted before use. Image copies are the equivalent of operating system file copies and can be used to restore operations immediately.

2. What is the destination of any disk backups that are done?

   a) In Enterprise Manager, select **Maintenance** > High Availability > Backup/Recovery Settings > **Backup Settings**. Note the message under the Disk Backup Location that says the Flash Recovery Area is the current disk backup location.



3. Test making a backup to disk, as a backup set, with `oracle` for Host Credentials.

   a) Select **Backup Set** as your Disk Backup Type.

   b) Scroll to the bottom and enter `oracle` and `oracle` for **Host Credentials Username** and **Password** for your server.

   c) Click **Test Disk Backup**.

   d) When the test finishes, click **OK**.

Solutions for Practice 15: Performing Database Backups (continued)

4. Back up your entire database, without archived logs, while the database is open for user activity. This backup should be the base for an incremental backup strategy.

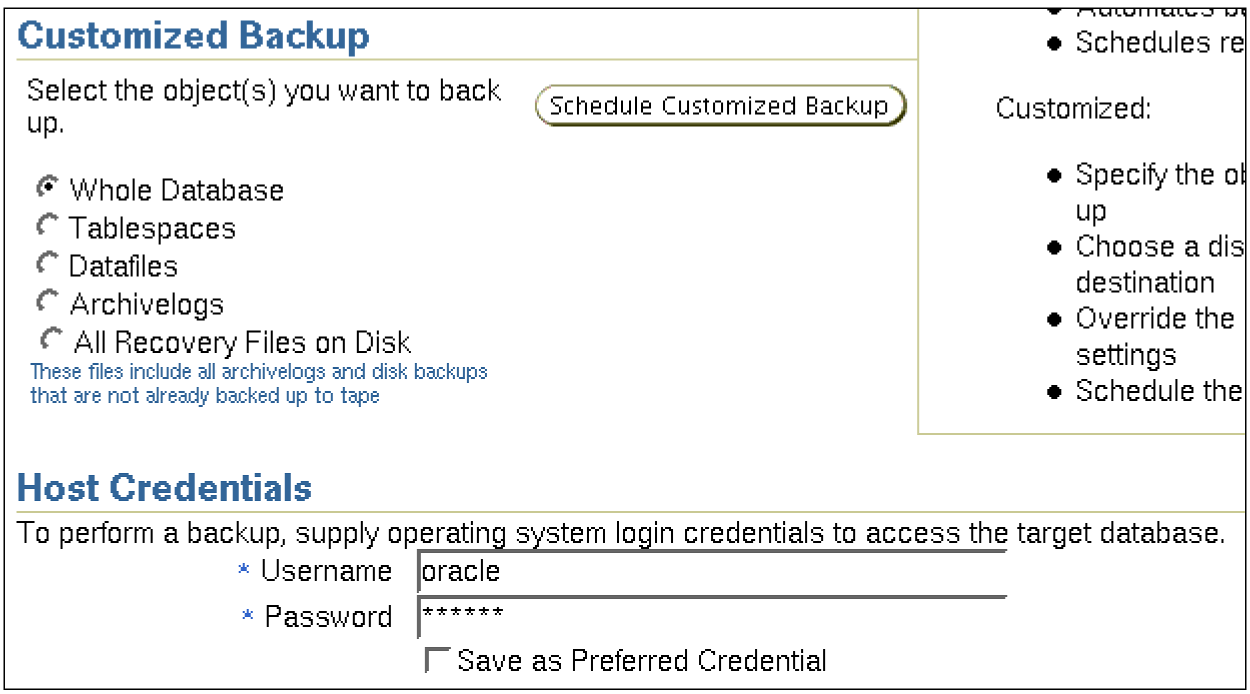
   Question: What prerequisite *must* be met to create a valid backup of a database without shutting it down?

   Answer: The database must be in `ARCHIVELOG` mode. Backups made with the database open but not in `ARCHIVELOG` mode cannot be used for recovery.

   a) Select **Maintenance** > High Availability > Backup/Recovery > **Schedule Backup**.

   b) Select **Whole Database** as the object to be backed up.

   c) Enter `oracle` and `oracle` for **Host Credentials Username** and **Password** for your server.



   d) Click **Schedule Customized Backup**.

   e) Select **Full Backup** for your Backup Type, and select the **Use as the base of an incremental backup strategy** check box.

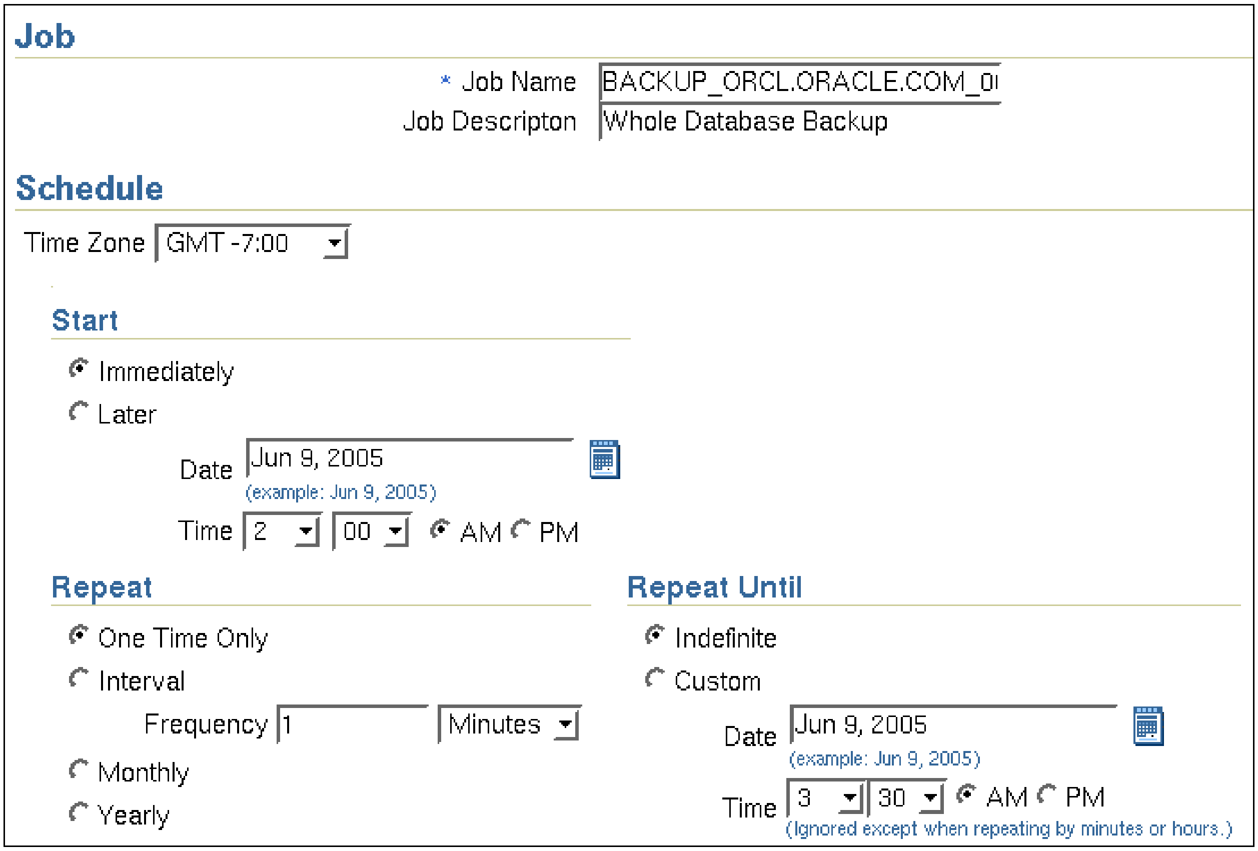
   f) Select **Online Backup** as Backup Mode.

   g) In the **Advanced** region, deselect the **Also backup all archived logs on disk** check box, and then click **Next** to continue.

Solutions for Practice 15: Performing Database Backups (continued)

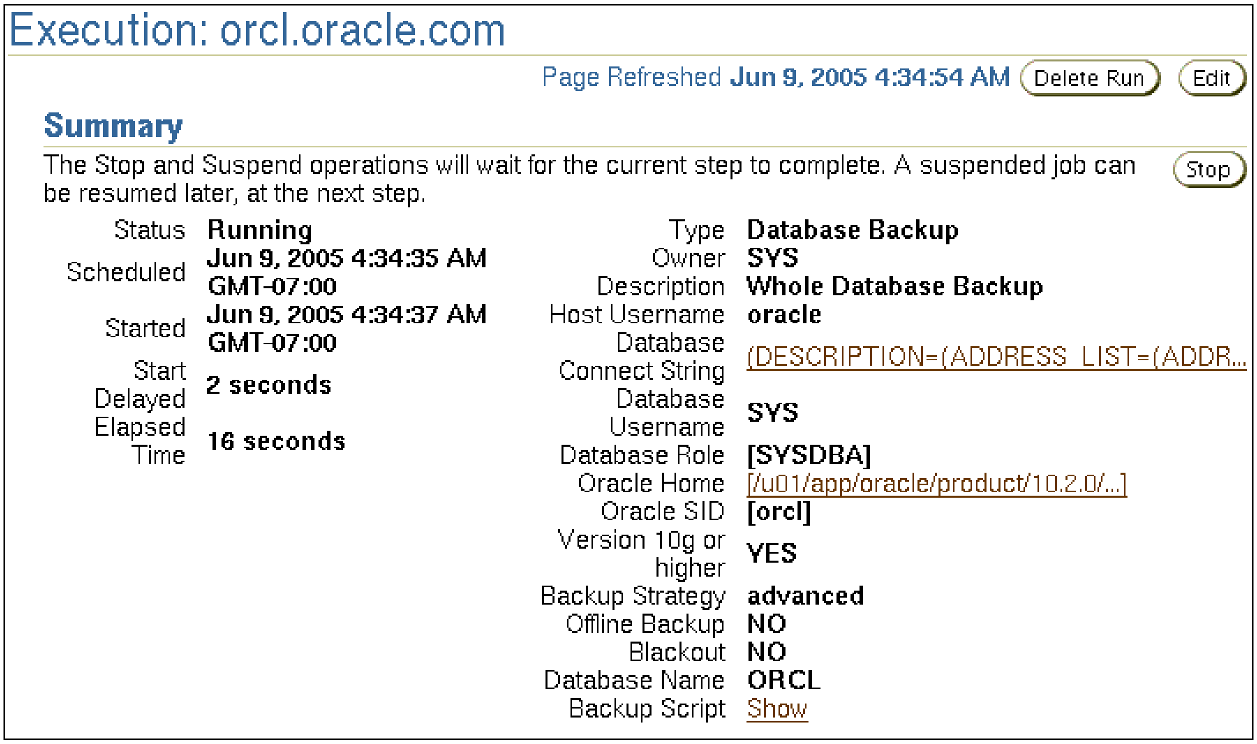
h) Select **Disk** for your backup location (notice that your Disk Backup Location is retained from step [2]). Click **Next** to continue.

i) Accept all the defaults on the Schedule page, and then click **Next** to continue.



j) Click **Submit Job** to perform the online database backup.

k) Click **View Job** to monitor the status of the backup job. This backup takes approximately five minutes to complete.



5. Schedule nightly disk-based incremental online backups for your whole database, without archived logs. Schedule it for 11:00 p.m. The schedule should be in effect indefinitely.

   a) In Enterprise Manager, select **Maintenance** > High Availability > Backup/Recovery > **Schedule Backup**.

   b) Select **Whole Database** as the object to be backed up.

   c) Enter `oracle` and `oracle` for **Host Credentials Username** and **Password** for your server, and then click **Schedule Customized Backup**.

   d) Select **Incremental Backup (Level 1)** for your Backup Type.

   e) Select **Online Backup** as Backup Mode.

   f) In the **Advanced** region, deselect the **Also backup all archived logs on disk** check box, and then click **Next** to continue.

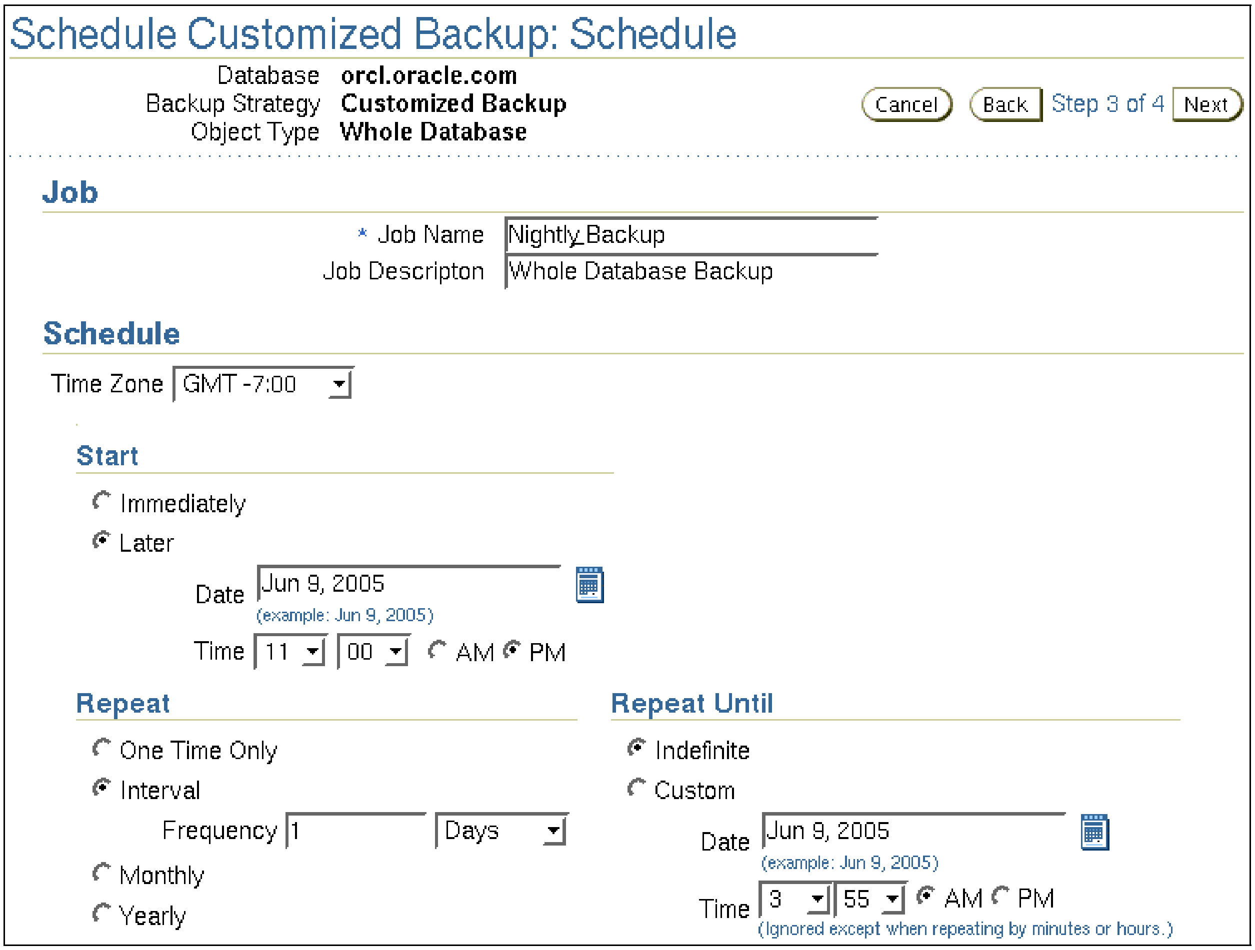
   g) Select **Disk** as your backup location, and then click **Next** to continue.

h) Change **Job Name** to `Nightly_Backup` and accept the default value for **Job Description**.

i) Select **Later** in the **Start** region. Accept today's date and use the drop-down lists and option buttons to select 11:00 p.m. for **Time**.

j) In the **Repeat** area, select **Interval**, and **Frequency** **`1 Days`**.

k) Select **Indefinite** in the **Repeat Until** region, and then click **Next**.



l) Click **Submit Job**, and then click **OK**.

m) Navigate to **Maintenance** > Related Links > **Jobs** to see the scheduled job in the **Job Activity** list.

Solutions for Practice 16: Performing 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. SQL script files are provided for you in the `/home/oracle/labs` directory. If needed, use appendix C for Linux and appendix D for SQL syntax. Note that where OS file names are mentioned, your system may possibly have different file names than shown here.

1. Recover from the loss of a control file.

   a) As the `DBA1` user, run the `lab_16_01_a.sql` script to prepare some procedures to be called by the rest of this practice.

   ```
   @$HOME/labs/lab_16_01_a.sql
   ```

   b) Now run the `lab_16_01_b.sql` script. This script deletes one of your control files.

   ```
   @$HOME/labs/lab_16_01_b.sql
   ```

   c) The Help desk begins receiving calls saying that the database appears to be down. Troubleshoot and recover as necessary. Use Enterprise Manager's Database page to attempt to start up the database, and use SQL*Plus, if needed.

   - In Enterprise Manager, navigate to the Database page. It reports that the database is down and offers you the chance to start it up again. Attempt to do so by clicking **Startup**. You may need to wait one or two minutes before the page appears as described. If you see a Connection Refused message, ignore it; the connection will eventually be established.

   - Enter host credentials as `oracle` for **Username** and **Password**, and then click **OK**.

   - When asked whether you are sure that you want to start the database, click **Yes**.

d) The startup of the instance fails with Enterprise Manager, and you can get no other information to explain the failure. So use the command-line tools.

- Connect to the instance with SQL*Plus `as sysdba` and check the current status of the instance.

```
sqlplus / as sysdba
select status from v$instance;
```

- The instance status is `STARTED`, which means that the database is in the `NOMOUNT` stage. Attempt to mount the database by entering this:

```
SQL> alter database mount;
```

```
SQL> alter database mount;
alter database mount
*
ERROR at line 1:
ORA-00205: error in identifying control file, check alert log for more info
```

e) The instance cannot move to the mount stage because it cannot find one of the control files. Check the last 10 rows of the alert log to see which control file is the problem.

```
SQL> host tail -10 $ORACLE_BASE/admin/orcl/bdump/aler*
```

Output:

```
SQL> host tail -10 $ORACLE_BASE/admin/orcl/bdump/aler*
ORA-205 signalled during: ALTER DATABASE   MOUNT...
Sun Oct 30 13:35:01 2005
alter database mount
Sun Oct 30 13:35:01 2005
ORA-00202: control file:
'/u01/app/oracle/flash_recovery_area/ORCL/controlfile/o1_mf_1p4sbncs_.ct
l'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
Additional information: 3
Sun Oct 30 13:35:01 2005
ORA-205 signalled during: alter database mount...
```

f) The control file in the Flash Recovery Area is missing. Restore the missing control file by copying the existing control file, and then mount and open the database.

```
SQL> host
$ cd /u01/app/oracle/oradata/ORCL/controlfile
$ ls -l
total 6908
-rw-r-----  1 oracle  oinstall  7061504 Oct 30 13:32 o1_mf_1p4sbn1o_.ctl
$ cp o1_mf_1p4sbn1o_.ctl
/u01/app/oracle/flash_recovery_area/ORCL/controlfile/o1_mf_1p4sbncs_.ctl
$ exit
```

```
SQL> alter database mount;
```

```
SQL> alter database open;
```

g) Why did you have to use two commands to move the instance state from `NOMOUNT` to `OPEN`?

Answer: Because the `ALTER DATABASE` command enables you to change only one state level per command

h) Why did you use operating system commands to restore the control file instead of using Oracle Recovery Manager?

Answer: Because all control files are identical. As long as any one control file is intact, it can be used to restore the others.

Solutions for Practice 16: Performing Database Recovery (continued)

2. Recover from the loss of an application data file.

   a) Start a SQL*Plus session as the `DBA1` user, and run the `lab_16_02.sql` script. This script deletes one of your application data files.

```
$ sqlplus dba1/oracle @$HOME/labs/lab_16_02.sql
```

   b) The Help desk has received a call from a user who is unable to access the `COUNTRIES` table in the `HR` application schema. Count the rows in the table to see whether there is a problem.

```
SQL> select count(*) from HR.COUNTRIES;
select count(*) from HR.COUNTRIES
                        *
ERROR at line 1:
ORA-01116: error in opening database file 5
ORA-01110: data file 5:
'/u01/app/oracle/oradata/ORCL/datafile/o1_mf_example_1p4sd3y2_.dbf'
ORA-27041: unable to open file
Linux Error: 2: No such file or directory
Additional information: 3
```

   c) Troubleshoot and recover as necessary. The error message suggests that the data file for the `EXAMPLES` tablespace is corrupt or missing. Using operating system commands, verify that there is a problem with the file.

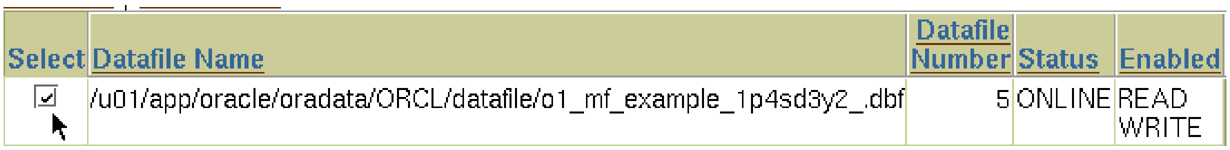
```
SQL> host ls
/u01/app/oracle/oradata/ORCL/datafile/o1_mf_example_1p4sd3y2_.dbf
ls: /u01/app/oracle/oradata/ORCL/datafile/o1_mf_example_1p4sd3y2_.dbf:
No such file or directory
```

   d) Recover the data file to the current time, specifying the missing data file to be recovered.

      - In Enterprise Manager, select **Maintenance** > High Availability > Backup/Recovery > **Perform Recovery**.
      - Select **Datafiles** from the **Object Type** drop-down list.
      - In the Object Level Recovery region, select **Recover to current time** for the **Operation Type**.
      - Enter the host credentials as `oracle` and `oracle` for **Username** and **Password**, and then click **Perform Object Level Recovery**.

Solutions for Practice 16: Performing Database Recovery (continued)

- On the Datafiles page, select the data file in question.



Select	Datafile Name	Datafile Number	Status	Enabled
☑	/u01/app/oracle/oradata/ORCL/datafile/o1_mf_example_1p4sd3y2_.dbf	5	ONLINE	READ WRITE

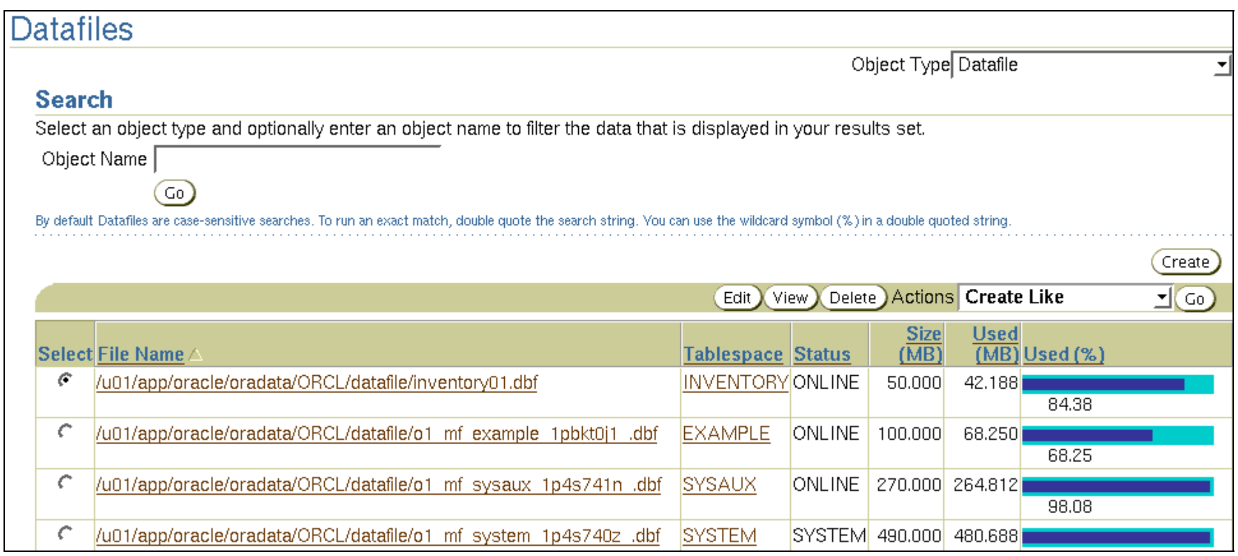
- Click **Next**.

- Because the problem is simply a deleted file rather than a bad hard drive, there is no need to restore the file to a different location. Select **No. Restore the files to the default location**, and then click **Next**.

- Click **Submit**. (*It will take one or two minutes for the operation to complete.)*

e) When you see the Operation Succeeded message, ensure that the restored data file is online.

- In Enterprise Manager, select **Administration** > Database Administration > Storage > **Datafiles**.



Select	File Name	Tablespace	Status	Size (MB)	Used (MB)	Used (%)
◉	/u01/app/oracle/oradata/ORCL/datafile/inventory01.dbf	INVENTORY	ONLINE	50.000	42.188	84.38
○	/u01/app/oracle/oradata/ORCL/datafile/o1_mf_example_1pbkt0j1_.dbf	EXAMPLE	ONLINE	100.000	68.250	68.25
○	/u01/app/oracle/oradata/ORCL/datafile/o1_mf_sysaux_1p4s741n_.dbf	SYSAUX	ONLINE	270.000	264.812	98.08
○	/u01/app/oracle/oradata/ORCL/datafile/o1_mf_system_1p4s740z_.dbf	SYSTEM	SYSTEM	490.000	480.688	

f) Verify that the `COUNTRIES` table is now accessible.

```
SQL> select count(*) from HR.COUNTRIES;

  COUNT(*)
----------
        25
```

3. Recover from the loss of a system 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 an application data file?

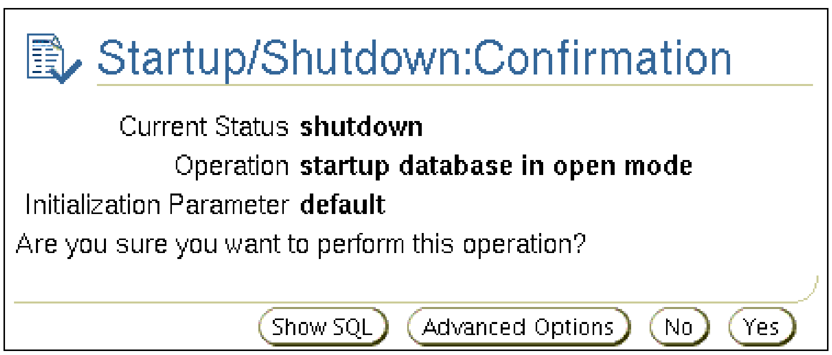
   Answer: Because recovery of system or undo data files must be done with the database closed, whereas recovery of an application data file can be done with the database open and available to users.

   b) As `SYSDBA`, run the `lab_16_03.sql` script. This script deletes the system data file.

```
SQL> @lab_16_03.sql

PL/SQL procedure successfully completed.

ORACLE instance shut down.
SQL>
```

   c) In Enterprise Manager, review the Database home page. The database is shut down, so you click **Startup** to try to open it. If you see a message that says the connection was refused, dismiss it, and reenter the EM home page URL in the browser.

   - Enter the host credentials as `oracle` and `oracle` for the host **Username** and **Password**, and enter `DBA1` and `oracle` for the database credentials, and then click **OK**.



   - Click **Yes**.

Solutions for Practice 16: Performing Database Recovery (continued)

d) This command will fail with the database left in the `MOUNT` state, because there is a data file missing from the `SYSTEM` tablespace.

e) Click **Perform Recovery**.

- If prompted, enter host (`oracle/oracle`) and database (`dba1/oracle AS SYSDBA`) credentials, and then click **Continue**.
- In the **Object Level Recovery** region, select **Datafiles** for Object Type.
- Select **Recover to current time** for **Operation Type**.
- Fill in the host credentials if not already set, and then click **Perform Object Level Recovery**.
- Select the data file for the `SYSTEM` tablespace, and then click **Next**.
- Because the problem is simply a deleted file rather than a bad hard drive, there is no need to restore to a different location. Select **No. Restore the files to the default location**, and then click **Next**.
- Click **Submit**. It will take three to four minutes for the operation to complete.

f) When you see the Operation Succeeded message, click **Open Database**.

g) After you see the success message, click **OK**, and then verify that the database is open and operating normally by logging into EM as `DBA1/oracle, as SYSDBA`, and reviewing the Database home page.

Solutions for Practice 17: Performing Flashback

Background: You decide to gain hands-on experience in some of the flashback functionality. To avoid impacting other users, you will first copy the `DEPARTMENTS` table of the `HR` schema to `DEPARTMENTS2`.

1. Log in to SQL*Plus as `DBA1` user and create a new `HR.DEPARTMENTS2` table based on the `HR.DEPARTMENTS` table.

   a) Log in to SQL*Plus as `dba1/oracle`.

```
$ sqlplus dba1/oracle
```

   b) Enter the following command to create the copy table:

```
SQL> create table hr.departments2 as select * from hr.departments;
```

   c) Count the rows in the `DEPARTMENTS2` table. There should be 27 rows.

```
SQL> SELECT COUNT(*) FROM HR.DEPARTMENTS2;

  COUNT(*)
----------
        27
```

2. Drop the `HR.DEPARTMENTS2` table, and then verify that it has indeed been dropped.

```
SQL> DROP TABLE HR.DEPARTMENTS2;

Table dropped.

SQL> SELECT * FROM HR.DEPARTMENTS2;
SELECT * FROM HR.DEPARTMENTS2
                 *
ERROR at line 1:
ORA-00942: table or view does not exist
```

3. Use the `FLASHBACK TABLE` command to restore the table. Count the rows in the `DEPARTMENTS2` table.

```
SQL> FLASHBACK TABLE hr.departments2 TO BEFORE DROP;

Flashback complete.

SQL> SELECT COUNT(*) FROM hr.departments2;

  COUNT(*)
----------
        27
```

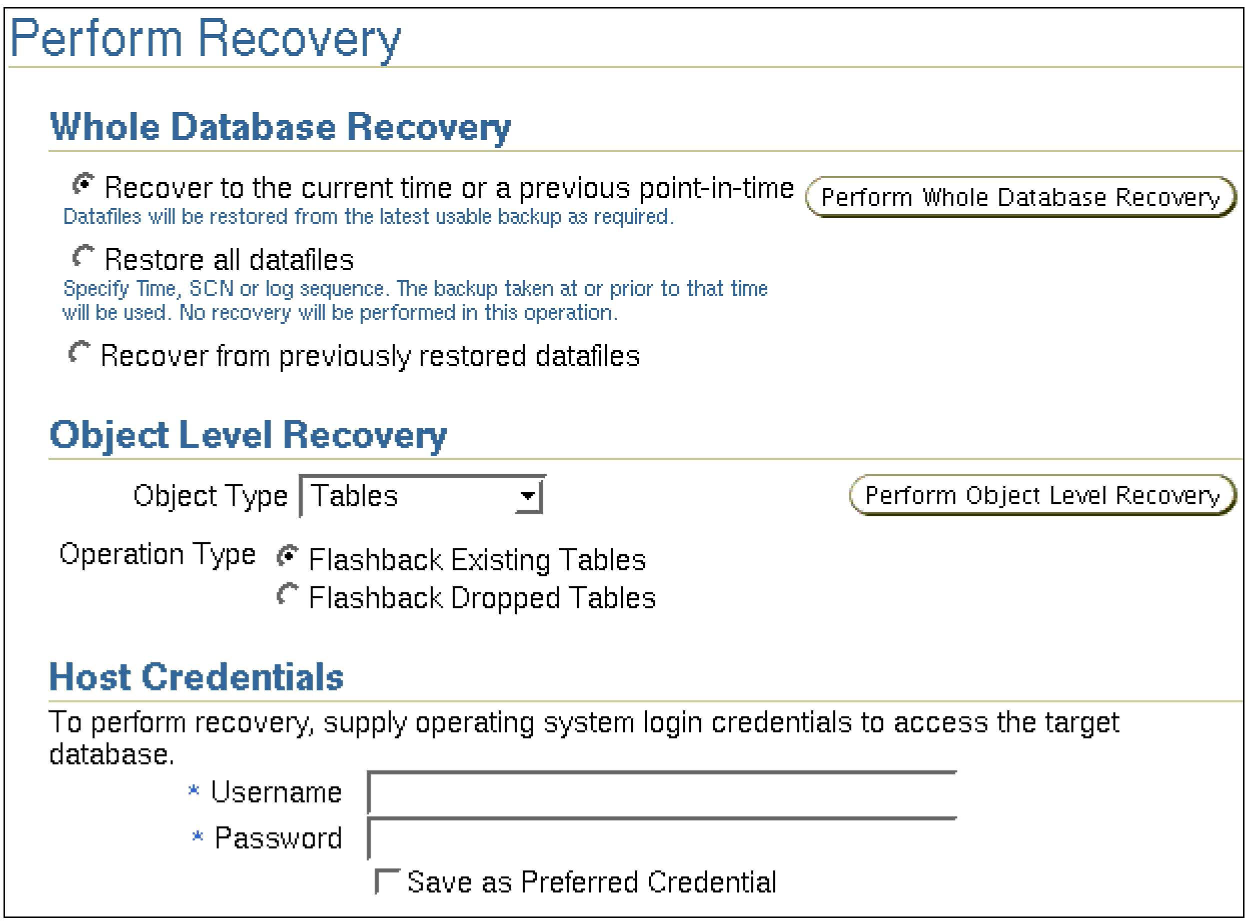
4. Run the `lab_17_04.sql` script to insert three rows into the `HR.DEPARTMENTS2` table by using three separate transactions. The new rows have `DEPARTMENT_ID` values of `280`, `290`, and `300`.

```
SQL> @lab_17_04.sql
```

5. Use Enterprise Manager to perform flashback to the version of the table where only the first of the three new rows is present (with `DEPARTMENT_ID` = 280). First, evaluate row changes to decide on a point in time. Limit your analysis to the new rows just added: where `department_id >= 280`. If you receive an error while performing the flashback, you may need to enable row movement on the table. See the next step.

   a) In Enterprise Manager, select **Maintenance** > High Availability > Backup/Recovery > **Perform Recovery**.

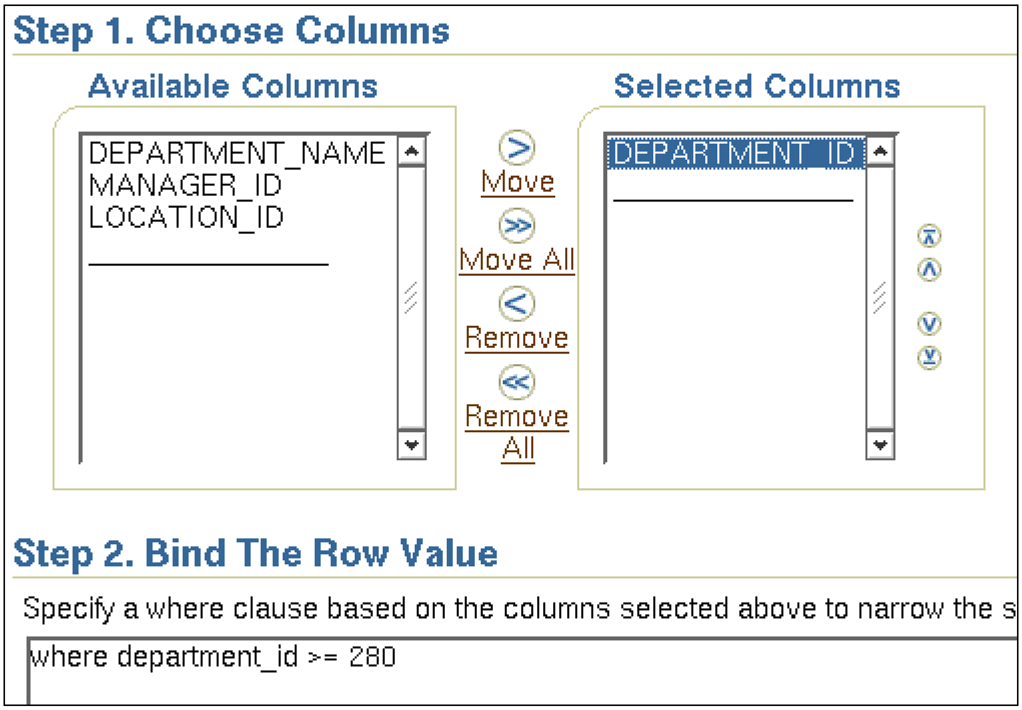
   b) Select Tables from the **Object Type** drop-down list, and then select **Flashback Existing Tables** for **Operation Type**. Click **Perform Object Level Recovery**.



   c) Select **Evaluate row changes and transactions to decide on a point in time**, and enter `HR.DEPARTMENTS2` as the fully qualified name of the table in the **Table** field, and then click **Next**.

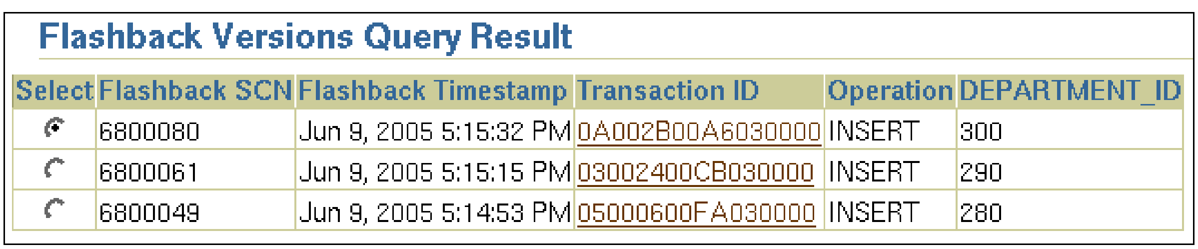
Solutions for Practice 17: Performing Flashback (continued)

d) Highlight `DEPARTMENT_ID` under **Available Columns,** and then click the **Move** button to move it under **Selected Columns**. Under Step 2, enter a `WHERE` clause that will select the added rows. For example, earlier you added rows with `DEPARTMENT_IDs` of `280`, `290`, and `300`, so the “`WHERE department_id >= 280`” clause is suitable.



e) Click **Next** to continue.

f) You can now review the rows under Flashback Versions Query Result.



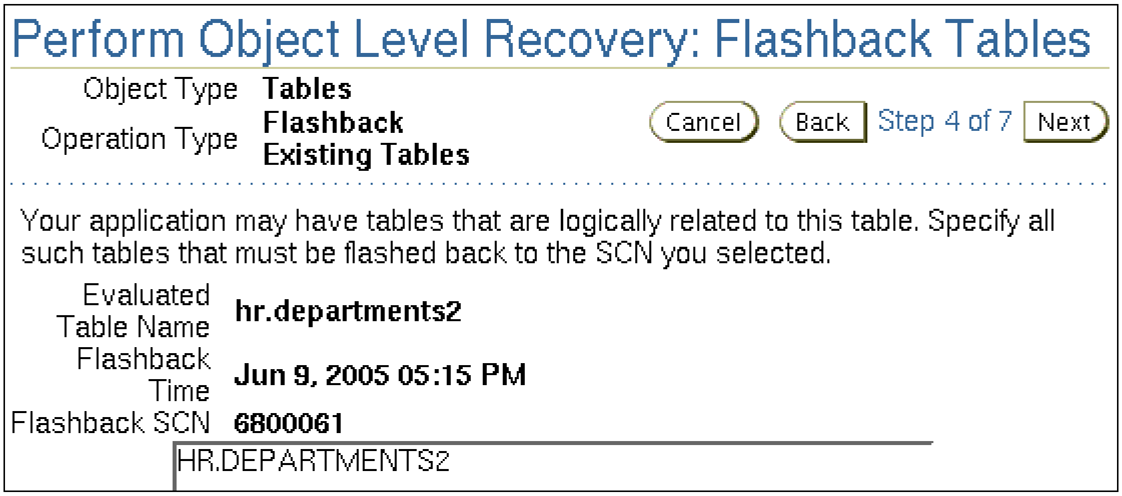
Flashback Versions Query Result

Select	Flashback SCN	Flashback Timestamp	Transaction ID	Operation	DEPARTMENT_ID
◉	6800080	Jun 9, 2005 5:15:32 PM	0A002B00A6030000	INSERT	300
○	6800061	Jun 9, 2005 5:15:15 PM	03002400CB030000	INSERT	290
○	6800049	Jun 9, 2005 5:14:53 PM	05000600FA030000	INSERT	280

g) Under **Flashback Versions Query Result**, select the middle transaction from the list to flash back. Click **Next** to continue.

Solutions for Practice 17: Performing Flashback (continued)

h) On the next page, you are asked to include any tables related to or dependent on the table that you are flashing back. Because there are none, click **Next** to continue.

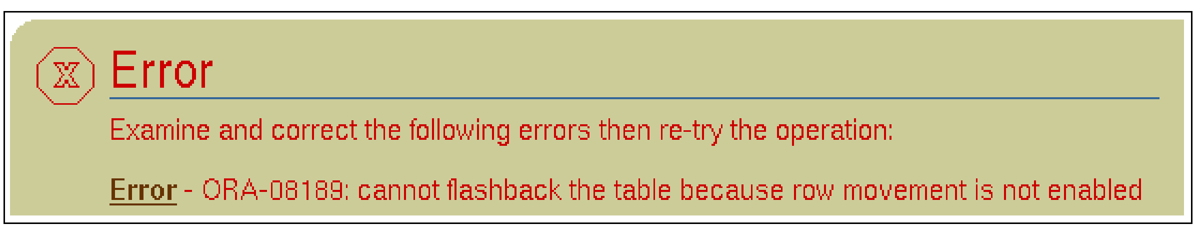


i) Review the SQL statement that you are about to execute. Click **Show SQL**, view the SQL statement, click **OK**, and then click **Submit**.

```
FLASHBACK TABLE HR.DEPARTMENTS2 TO SCN 6800061
```

6. On the next page, you find that the operation has failed because row movement is not enabled for the table. You may recall from the lesson that row movement must be enabled for this feature to work.

   a) Note this error:

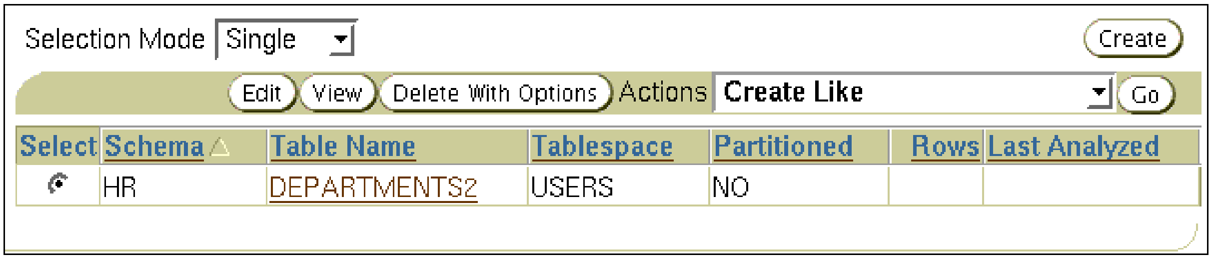


   b) To enable row movement for this table, select **Administration** > Schema > Database Objects > **Tables**.

   c) Enter `HR` in the **Schema** field and `DEPARTMENTS2` in the **Object Name** field, and then click **Go**.

Solutions for Practice 17: Performing Flashback (continued)

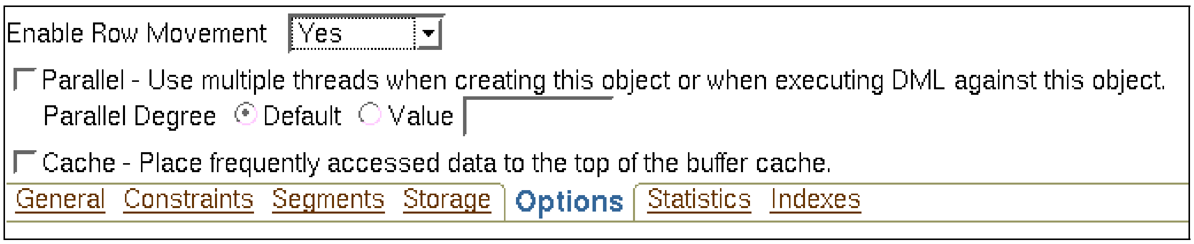
d) One table is displayed in the **Results** region and is selected by default. Click **Edit**.



The Edit Table page appears.

e) Click the **Options** tab.

f) From the **Enable Row Movement** drop-down list, select **Yes**.



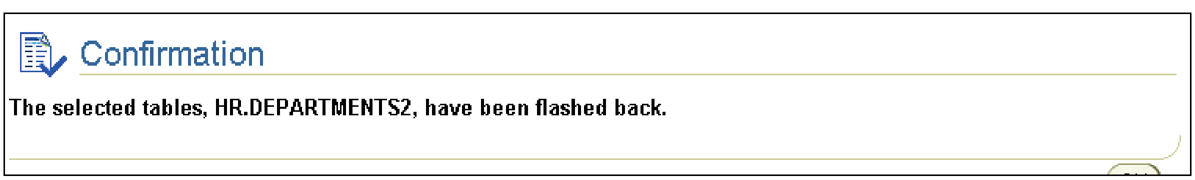
g) Click **Show SQL**, and then click **Return**.

```
ALTER TABLE "HR"."DEPARTMENTS2" ENABLE ROW MOVEMENT
```

h) Click **Apply** to apply the change.

After the table modification is confirmed, you can perform the flashback operation.

Repeat step 5 to perform the version flashback of the table. Because row movement has been enabled, it should succeed this time.



Solutions for Practice 17: Performing Flashback (continued)

i) In SQL*Plus, count the rows of the `HR.DEPARTMENTS2` table to confirm the flashback operation. Note that there is only one additional row now, not three. Then display the rows where `DEPARTMENT_ID >= 280`. Note that only one of the original three is remaining.

```
SQL> select count(*) from hr.departments2;

  COUNT(*)
----------
        28

SQL> select * from hr.departments2 where department_id >= 280;

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
          280 DUMMY1

SQL>
```

Solutions for Practice 18: Moving Data

Background: In the recent past, you received a number of questions about the HR schema. To analyze them, without interfering in the daily activities, you decide to use the Data Pump Wizard to export the HR schema to file. When you perform the export, you are not sure into which database you will be importing this schema.

In the end, you find out that the only database for which the management approves an import for, is the ORCL database. So you perform the import with the Data Pump Wizard, remapping the HR schema to a newly created HR_TEST schema in the HR_TEST tablespace. To follow best practice guidelines, you also decide to create a DP user who will be a DBA performing Data Pump jobs. For your convenience in class, the creation of the DP user is included in the lab_18_01.sql script. SQL scripts are in the /home/oracle/labs directory. However, there is one step missing.

Then, you receive two data load requests for which you decide to use SQL*Loader.

1. Review the lab_18_01.sql script, which creates the HR_TEST tablespace, the HR_TEST_ROLE role, and the HR_TEST and DP users.

    a) Double-click the **oracle's Home** icon on your desktop, double-click **labs**, and then double-click **lab_18_01.sql** to review this script.

    b) Note the passwords for these users.

       HR_TEST password: **hr_4test**

       DP password: **dp_4test**

    c) Which additional step do you need to perform to allow the DP user access to Enterprise Manager as Administrator?

    You need to log in to Enterprise Manager as the SYS user in the SYDBA role and make the DP user an EM Administrator (as you learned in the lesson titled "Managing the Oracle Instance").

    d) Close the lab_18_01.sql window.

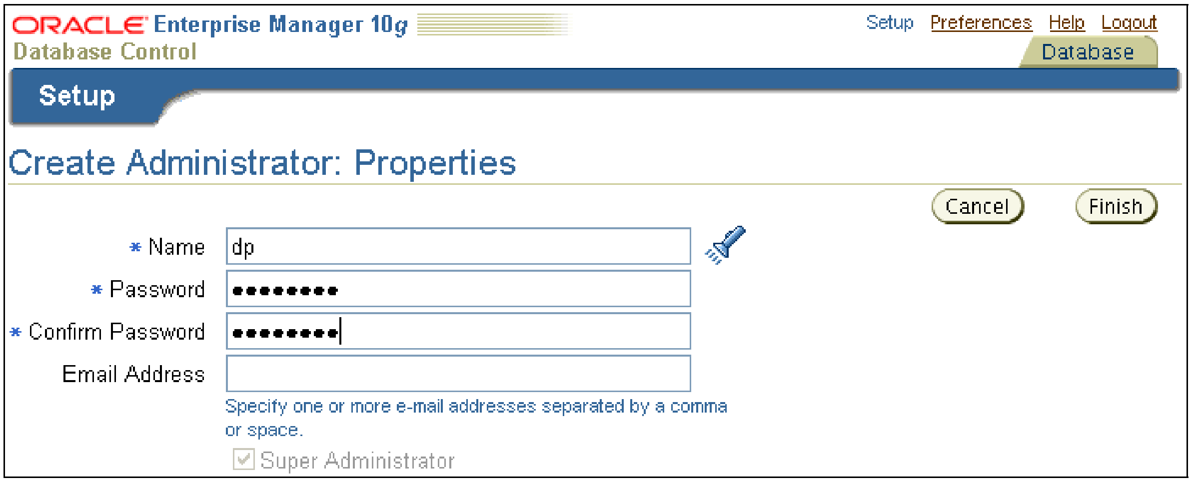
2. Execute the lab_18_02.sh script. Then, perform the required step to make the DP user an EM administrator.

    a) Right-click your desktop and select **Open Terminal**. Then navigate to the /home/oracle/labs directory by entering **cd labs**.

    b) Enter **./lab_18_02.sh** to create the HR_TEST tablespace, the HR_TEST_ROLE role, and the HR_TEST and DP users.

    c) Log in to Enterprise Manager as the SYS user in the SYSDBA role.

d) Make the DP user an EM Administrator (as you learned in the lesson titled "Managing the Oracle Instance"). (Select Setup > Create > Enter Name: DP, Password: dp_4test, Super Administrator, checked > Click Finish > Finish > Logout.)



3. Log in to Enterprise Manager as the DP user in the Normal role and export the HR schema.

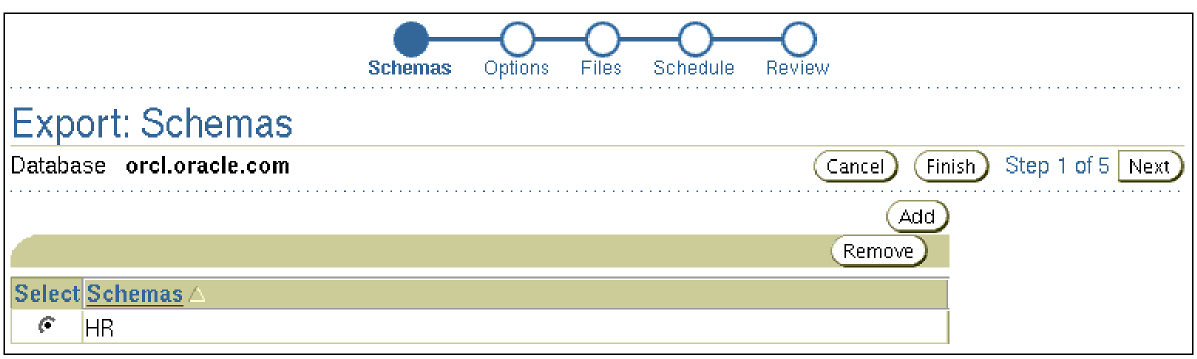
a) Invoke Enterprise Manager as the DP user in the Normal role for your ORCL database. The **Connect As** setting should be Normal.

If this is the first time that the DP user logs in, click **I agree** to accept the licensing agreement.

b) Select **Maintenance** > Data Movement > Move Row Data > **Export to Export Files**.

c) Select **Schemas**, enter **oracle** as **Username** and **Password**, select **Save as Preferred Credential**, and then click **Continue**.

d) On the Export: Schemas page, click **Add**, select the HR schema, and then click the **Select** button.



e) Click **Next**.

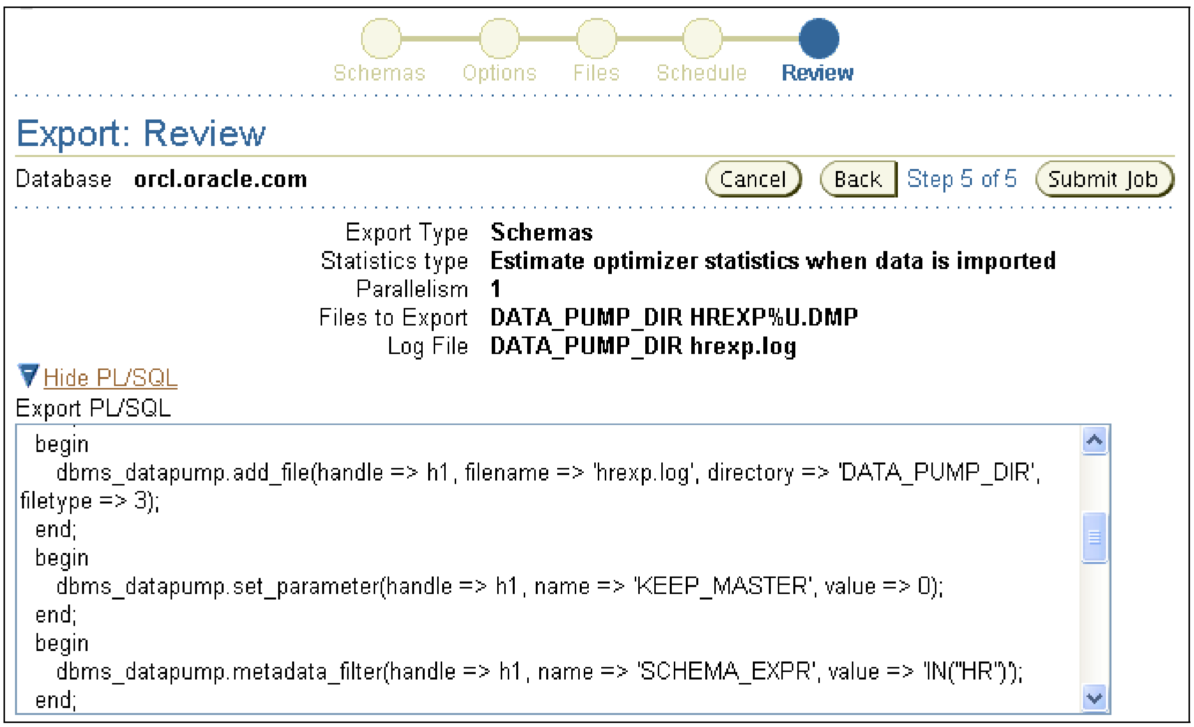
f) On the Export: Options page, select `DATA_PUMP_DIR` from the **Directory Object** drop-down list, and enter **`hrexp.log`** as **Log File**.

g) Review Advanced Options, but do not change, and then click **Next**.

h) On the Export: Files page, select `DATA_PUMP_DIR` from the **Directory Object** drop-down list, enter **`HREXP%U.DMP`** as **File Name**, and then click **Next**.

i) On the Export: Schedule page, enter **`hrexp`** as **Job Name** and **`Export HR schema`** as **Description**, accept the immediate job start time, and then click **Next**.

j) On the Export: Review page, click **Show PL/SQL** and review the PL/SQL that the Export Wizard helped you to create.



k) Click **Submit Job**.

A processing message appears, and then a success message. If not, resolve any errors, which may have occurred.

l) When the Job Activity Confirmation page appears, click the HREXP job name, and then monitor the job progress by clicking the browser's **Reload** button.

Note: Please wait, not only for the job to be created, but also for the job to complete execution. (It may take two minutes.)

4. As the `DP` user, import the exported `HR` schema back into the `ORCL` database, remapping it to the previously created `HR_TEST` schema.

a) Invoke Enterprise Manager as the `DP` user in the `Normal` role for your `ORCL` database.

b) Select **Maintenance** > Data Movement > Move Row Data > **Import from Export Files**.

c) On the Import: Files page, select `DATA_PUMP_DIR` from the **Directory Object** drop-down list, and enter **`HREXP%U.DMP`** as **File Name**.

d) Select **Schemas** as the Import Type.

e) Confirm `oracle` as your **Host Credentials**, and then click **Continue**.

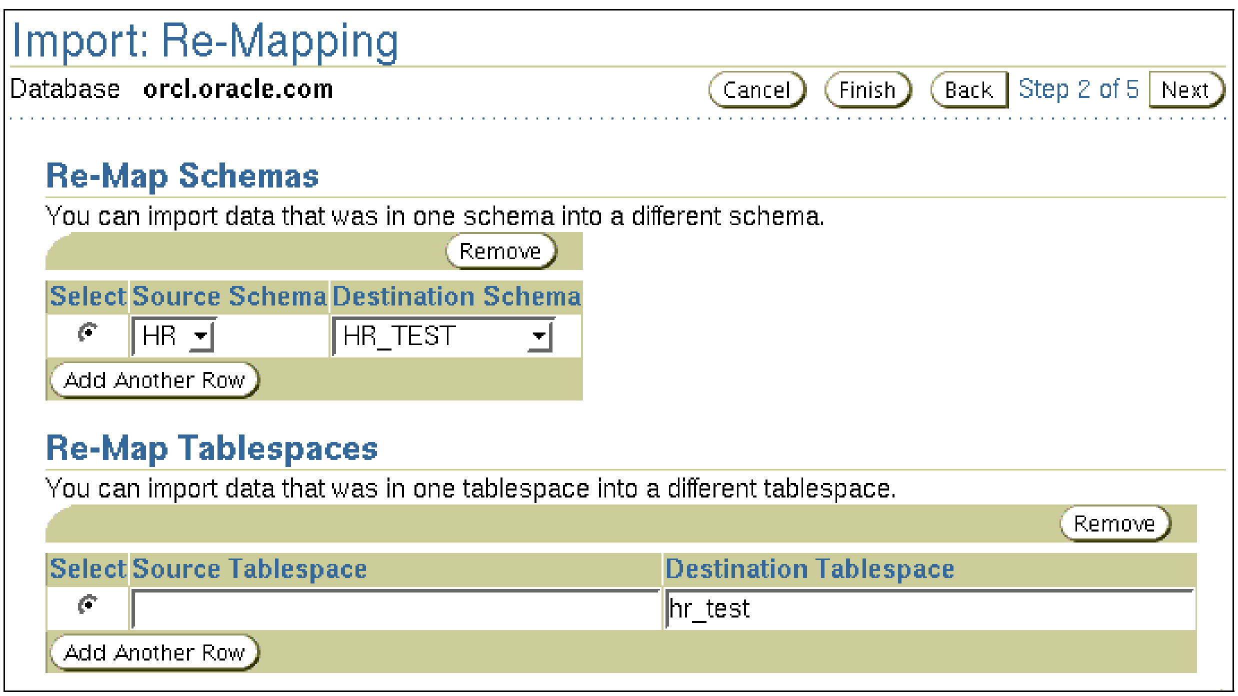
At this point, the export file is read, to verify the contents. Wait for this to complete.

f) On the Import: Schemas page, click **Add**, select HR, and then click **Select**.

g) Click **Next**.

h) On the Import: Re-Mapping page, click **Add Another Row** under **Re-Map Schemas**. Then select **`HR_TEST`** as **Destination Schema**.

i) Click **Add Another Row** under **Re-Map Tablespaces**, enter **`HR_TEST`** as **Destination Tablespace.**



j) Click **Next**.

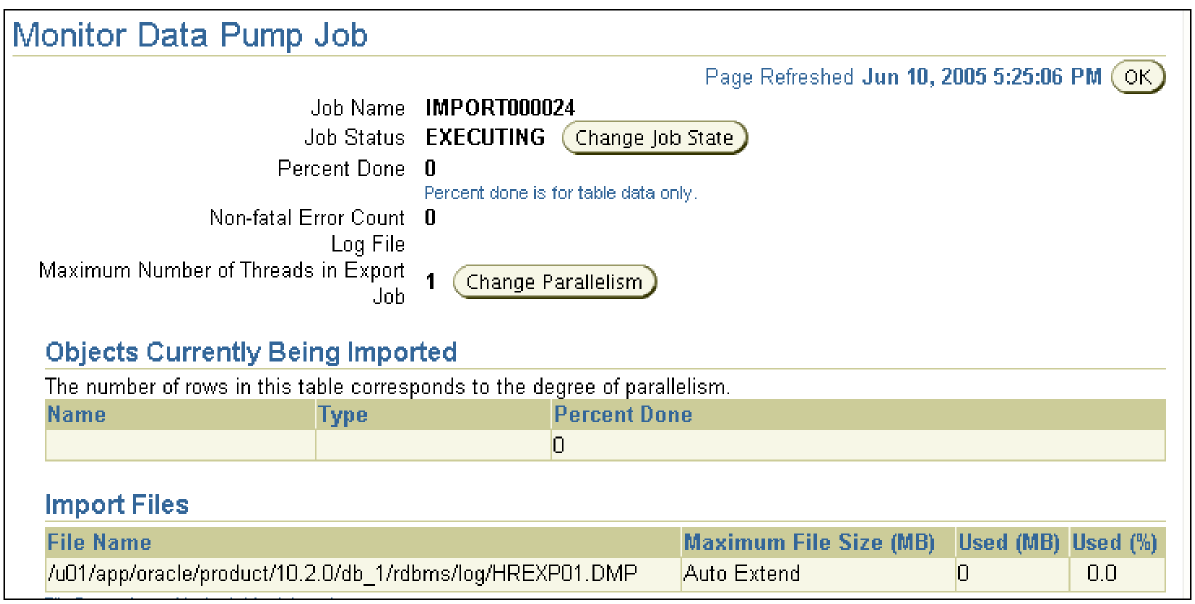
Solutions for Practice 18: Moving Data (continued)

k) On the Import: Options page, select **`DATA_PUMP_DIR`** from the **Directory Object** drop-down list, enter **`hrimport.log`** as **Log File**, review the advanced options, but leave them at their default values, and then click **Next**.

l) On the Import: Schedule page, enter **`hrimp`** as **Job Name** and **`Import HR schema for test purposes`** as **Description**.

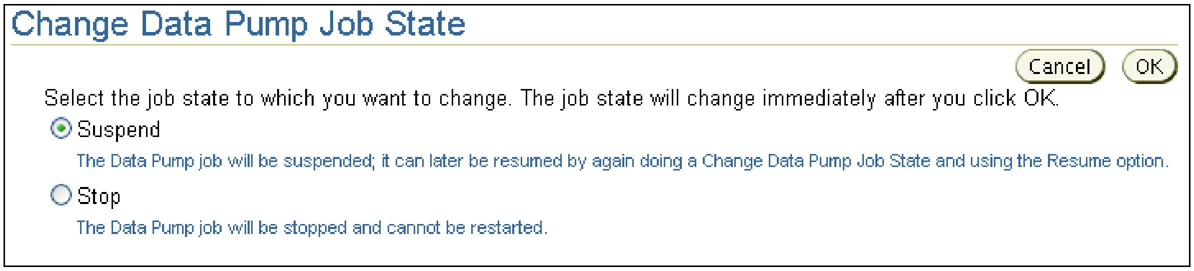
m) Select **Later** as **Start** and enter a time between 2 and 5 minutes from now (to give yourself time for the following steps):

- Click **Next**.
- On the Import: Review page, review the PL/SQL that the Data Pump Wizard creates for you, and then click **Submit Job**.
- After the confirmation that the job was successfully created, note the job name (for example `HRIMP`), and navigate to **Maintenance** > Data Movement > Move Row Data > **Monitor Export and Import Jobs**.
- Click your last job (for example, `HRIMP)`.
- On the "Monitor Data Pump job" page, click **Reload** in your browser, when it is time to execute your job.



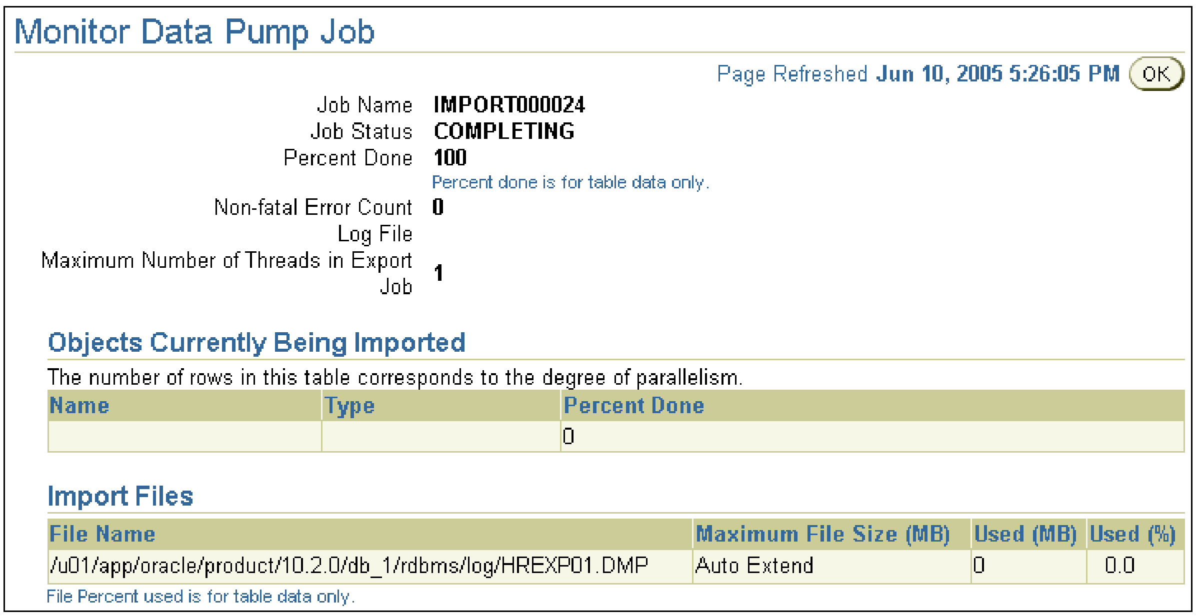
Solutions for Practice 18: Moving Data (continued)

- *Optional,as this requires quick action:* Click **Change Job State**.



- *Optional (continued):* Click **Suspend** and later **Resume** to halt the job and then to continue it again.

n) In the end, you want to see that your job executed 100% without any errors.



o) Click **OK**.

p) Verify that the import succeeded by viewing the log file.

```
$ cat </u01/app/oracle/product/10.2.0/db_1/rdbms/log/hrimport.log
```

Note: You may see an error saying that the hr_test object already exists. This is because that user existed when you did the export, and still exists. This is not a problem.

q) Using SQL*Plus, connect to the database as the `HR_TEST` user.

```
$ sqlplus hr_test/hr_4test
```

r) Select data from tables in the `hr_test` schema, for verification of the import.

```
SQL> select * from jobs;
```

Solutions for Practice 18: Moving Data (continued)

5. As the `DP` user, load data into the `PRODUCT_MASTER` table by using SQL*Loader via Enterprise Manager Database Control.

   a) Invoke Enterprise Manager as the `DP` user in the `Normal` role for your `ORCL` database.

   b) Select **Maintenance** > Data Movement > Move Row Data > **Load Data from User Files**.

   c) Click **Use Existing Control File**.

   d) If you have not done so before, enter **`oracle`** as **Username** and as **Password**, click **Save as Preferred Credential**, and then click **Continue**.

   e) On the Load Data: Control File page, enter **`/home/oracle/labs/lab_18_05.ctl`** as control file name and path, or use the flashlight icon to select this control file. Click **Next**.

   f) On the Load Data: Data File page, click **Provide the full path and name on the database server machine** and enter **`/home/oracle/labs/lab_18_05.dat`** as data file name and path, or use the flashlight icon to select this data file. Click **Next**.

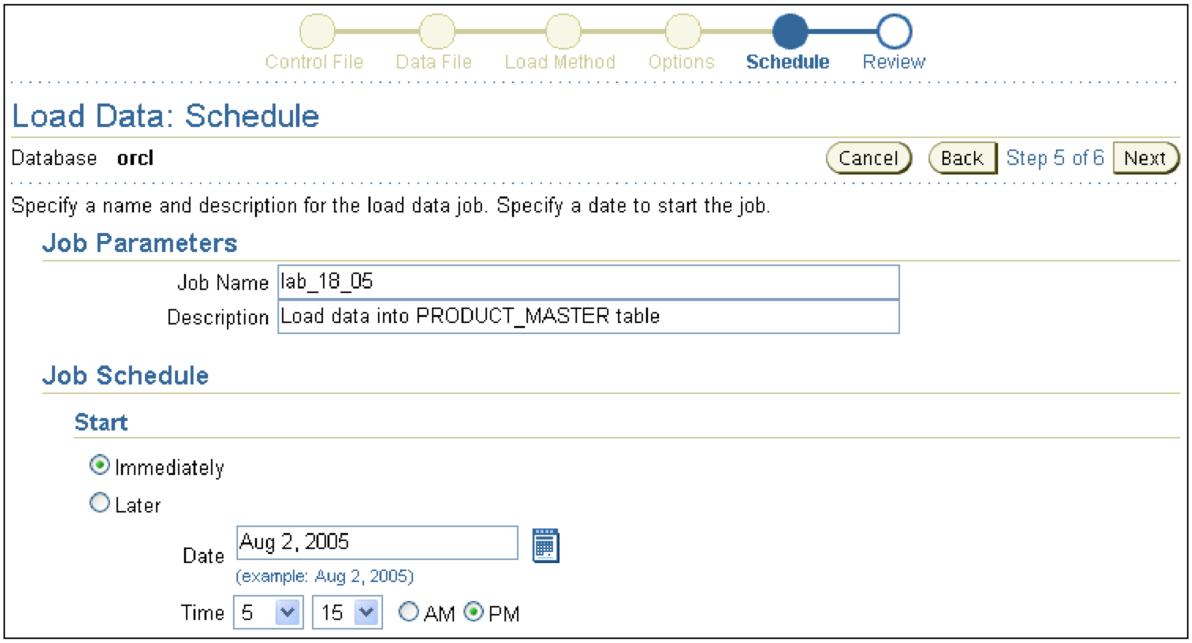
   g) On the Load Data: Load Method page, accept **Conventional Path**, and then click **Next**.

   h) On the Load Data: Options page, accept all defaults, but enter **`/home/oracle/labs/lab_18_05.log`** as log file name and path.

   i) Review the advanced options, but do not change any, and then click **Next**.

Solutions for Practice 18: Moving Data (continued)

j) On the Load Data: Schedule page, enter **`lab_18_05`** as **Job Name** and **`Load data into the PRODUCT_MASTER table`** as **Description**. Let the job start immediately, and then click **Next**.



k) On the Load Data: Review page, review the loading information and parameters, and then click **Submit Job**.

l) Confirm your results by viewing your `lab_18_05.log` file in your `/home/oracle/labs` directory.

6. As the `INVENTORY` user, load data into the `PRODUCT_ON_HAND` table by using SQL*Loader command line.

   a) Invoke a terminal window and navigate to the `/home/oracle/labs` directory.

   b) Enter the following SQL*Loader command (in continuation, without pressing [Enter] before reaching the end of the command): **`sqlldr userid=inventory/verysecure control=lab_18_06.ctl log=lab-_18_06.log data=lab_18_06.dat`**.

```
sqlldr userid=inventory/verysecure control=lab_18_06.ctl
log=lab_18_06.log data=lab_18_06.dat

SQL*Loader: Release 10.2.0.1.0 - Production on Tue Aug 02 22:24:44
2005

Copyright © 1982, 2005, Oracle. All rights reserved.

Commit point reached - logical record count 64
Commit point reached - logical record count 82
Commit point reached - logical record count 83
```

Solutions for Practice 18: Moving Data (continued)

c) Confirm your results by viewing your `lab_18_06.log` file in your `/home/oracle/labs` directory.

Congratulations!

You completed all practices for the

Oracle Database 10g: Administration Workshop I

Appendix C
Basic Linux and vi Commands

vi Commands

The Visual Interpreter/Editor (*vi*) is the most widely used text editor available for the UNIX environment. While almost everybody curses its unwieldy command syntax, it is still the only editor almost certain to be included with every version of the UNIX and Linux operating system. The following are a partial list of available *vi* commands.

vi has two modes. Command line (where anything typed is taken as an editing command) and input mode (where everything typed will be treated as part of the file being edited. To enter the input mode, type a, A, i, I, o, O, c, C, s, S, r, or R. To return to the command line mode, use the <ESC> key. To access the *vi* editor from SQLPlus, enter the following command:

```
SQL>define _editor=vi
```

To edit a file from SQLPlus prompt, edit <*filename*> (press enter), from the Linux command prompt, vi <*filename*> (press enter)

To MOVE the cursor:

h - move left   j - move down   k - move up   l - move right

w - one word forward   b - one word backward   e - end of current word

W, B, or E - same as lower case but ignores punctuation

0 (zero) - Move to beginning of current line $ - end of current line

G - go to last line of file   H - go to top line on the screen

L - go to last line on screen   M - go to middle line on the screen

/<string> - Search forward to the next occurrence of <string>

?<string> - Search backward to the next occurrence of <string>

n - Repeat previous search   N - Repeat previous search in opposite direction

<ctrl> f - Scroll forward one page   <ctrl> b - Scroll backward one page

To UNDO previous changes:

u - Will undo the most recent change.   U - Will undo the most recently deleted text.

:e! - re-edit current file without saving any changes made since last change

To ENTER NEW text:

a - Append text after the current cursor position.

A - Append text to the end of a line (jumps to end of line and begin appending).

c - Change object   C - Change from current cursor position to end of the line

i - Insert text before the current cursor position.   I - Insert text at the beginning of a line.

o - Insert a blank line BELOW the current cursor position.

O - Insert a blank line ABOVE the current cursor position.

r - Replace character at current cursor position   R - Replace all characters until <ESC> is pressed

s - substitute text for character under cursor

:s/A/B/opt substitutes string B for string A. %s/A/B/opt is global replace

options include: g (change all occurences on current line)   c (confirm prior to each change)

p (print changed lines)   S - Substitute entire line to the end

. <period> - repeat last change   *n*. <integer><period> repeat last change *n* times

To leave the input mode, press <ESC>

To DELETE existing text:

x - Will delete the character directly under the current cursor location.

dd - Will delete the entire line where the cursor is located.

d*n*d (where *n* is some integer) will delete *n* lines from current cursor position

dw - delete current word   D - delete to end of current line

J - Delete return at end of current line. Join this line and the next

<int> J - Join the next <int> lines

COPY, CUT, and PASTE: *vi* uses a single buffer where the last changed or deleted text is stored. This text may be manipulated with the following commands:

Y - Yank a copy of the current line   y <integer> - Yank a copy of next <int> lines

yw - Yank a copy of the current word   yb - Yank a copy of the previous word

p - Put buffer contents after cursor   P - Put buffer contents before cursor

Also, see the s and S commands under the input section

To SAVE edited changes to an operating system file:

zz - Will terminate edit mode.   :w filename - Will save changes to the filename specified.

:wq - Write all changes and quit the edit mode

To QUIT without saving changes:

ZZ - Will terminate edit mode.   :q! - Will terminate the file without saving changes.

Basic Linux Commands

This appendix is meant to serve only as a quick reference while you are in class. For more details on these commands, consult the man pages, your Linux documentation, or other Linux command reference books.

Files and Directories	**Linux Commands**	**Description/Comments**

Command manual	`man <command>`
	
	`man –k <string>`
	
	`man man`
		Find the manual entry for this <command>.
		Show all the manual entries that contain this <string>.
		Displays the manual page for `man`.

Command information	`info <command>`	Show the information system entry for this command. Using `info info` shows a tutorial of the `info` documentation system.
Print to standard out	`cat <file>`	Concatenate and print – print the named file to the terminal screen.
List users	`cat /etc/password`	
Change working directory	`cd <directory>`	Change working directory to specified directory cd with no parameters changes to $HOME.
Copy a file	`cp <source_file> <destination_file>`	Copy a source file to a destination file.
View a file	`less <file>`	View a file a page at a time. This is a GNU version of `more`, or `pg`.
View a file	`more <file>`	View a file a page at a time. BSD version.
List directory	`ls <directory>`	Options –l long listing, -R recursive, -a show hidden files, -t sort by time, -r reverse sort, default directory is current working directory.
Create a directory	`mkdir <directory>`	Make a directory defaults into the current working directory, full path may be specified.
Move or rename a file	`mv <old_file> <new_file>`	Move changes the name of a file or moves it to a different directory.

Process List	ps ps -ef	Shows the processes report Shows all processes on the system with a full listing. Many option exist see the man page for details.
Print working directory	pwd	Print to stdout the current working directory.
Remove or erase a file	rm <file>	Removing a file on Linux is permanent. Options –r recursive, and –f force (including subdirectories) are *very dangerous*. Often the rm command is aliased with rm –i The option –i asks ‘Are you sure?’
Create an empty file	touch <file>	Create a file.
Name of the machine	hostname	Returns the name of the machine.
The IP address of the machine	host <machine_name>	Queries the Domain Name Server, and returns the IP address of the machine name.
Remote shell	rsh <host> <command>	Execute a <command> on <host>. Rsh is not secure, use ssh instead.
Remote shell	ssh <host>	Secure shell, has features to replace rsh, rcp, ftp, and telnet.
Remote shell	telnet <host>	Start a terminal session on <host>. Telnet is not secure use ssh instead.
Search a file for a pattern	grep <option> <pattern> <file>	Search a <file> or stream for a regular expression defined by <pattern> and show the line that contains that pattern. A common option is –i for case insensitive. grep can accept input from a file or stdin through a pipe as in: netstat –a| grep ESTABLISHED
Source a script	. <script_file>	In the bash shell this command ‘.’ forces the script to run in the shell. Normal behavior is for the script to run in a child shell.

An interpreter	`awk`	A macro language for reformatting or interpreting input. For each line of input, a variety of actions can be taken. May be referred to as nawk – for “new awk.”
Sort a file	`sort`	Sort a file takes input from stdin or a filename argument, many options to sort by a particular column, field, etc. See man page.
Command line editor	`sed`	Sed is a command line editor, with many possible commands and options that are very good for editing from a shell script.
Visual editor	`vi <file>`	Terminal based editor available on every Unix system, Linux provides vim, an improved vi, that is a superset of vi.
Gnu editor	`emacs <file>`	This is a GPL editor with extensive customizable features available on most UNIX and Linux distributions.
WSIWIG editor	`gedit <file>`	A full-screen editor, requiring X. Available under Gnome.
WSIWIG	`kate <file>`	A full-screen editor, requires X. Available under KDE
Terminal output	`stdout`	Standard out (stdout), is not a command but a concept, most Linux commands write to stdout by default unless redirected.
Terminal input (keyboard)	`stdin`	Standard in (stdin), is not a command but a concept, most Linux commands read from stdin by default unless redirected.
Alias	`alias <command> <alias>`	Make a substitution when a user types <command> substitute and execute <alias>, common alias is alias ‘rm’ ‘rm –i’. These aliases are set in the .bashrc file.
Show shell variables	`set`	Prints all of the variables that are currently defined in the shell.

Show environment variables	`printenv or env`	Prints all the environment variables – an environment variable has been ‘exported’ so that it will be inherited by child processes.
File Creation mask	`umask –S u=rwx,g=rx,o=rx`	Set the default permissions for all files created by this shell or its children. The –S option uses the symbolic notation, the numeric notation is obsolete.
Clock	`xclock`	An X client that shows a clock on the screen. Often used to test the X windows system.
X access control	`xhost` `xhost +<Xclient>`	Show the current access control in place. Add a Xclient that is allowed to access the local DISPLAY, if no <Xclient> is given all are allowed.

System Administration	Linux Commands	Description / Comments
Root file system	`/`	The root directory for the system directory tree.
Home Directory	`/home`	Typically the directory in which all user home directories placed. For example: `/home/oracle`.
Tmp directory	`/tmp`	A temporary storage area. Do notput anything here you want to keep. SA often have a cron job to remove everything periodically.
Boot directory	`/boot`	A small partition to hold the kernel image(s) and boot loader instructions.
Log directory	`/var/log`	The location of most system log files.
Sample configuration files	`/etc/inittab`	Configuration files are located per the application. Any configuration file that you change after installation should be included in the backup.
Password files	`/etc/passwd` `/etc/shadow`	The `/etc/passwd` file holds user information and must be readable by others; even with encrypted passwords this can be a security hole. The `/etc/shadow` file holds the encrypted passwords and is only readable by root.
Groups file	`/etc/group`	The `/etc/groups` file defines the groups on a server and the users that are members of the group; primary group for a user is defined in the `/etc/passwd` file.
X configuration file	`/etc/X11/XF86Config`	The file that sets the X server settings for your video card, monitor, mouse, and keyboard. Usually set up with a vendor supplied tool, such as `sax2`.

Schedule a command to run at a regularly scheduled time	`crontab -e`	Use this command to edit the `crontab` file, to create the specification for the `cron` daemon to use.
Schedule a script to run at a particular frequency	`/etc/anacrontab`	Edit the file to specify a script to run at a particular frequency (see `man anacrontab` for details).
Schedule a command to run at a single specified time	`at <options> TIME`	Runs a job specified by <options> at a specified TIME parameter.
Schedule a command	`batch <options> <TIME>`	Run a command when the load average drops below .8, optionally after a set TIME.
Mount a file system	`mount <opt> <dev> <mount_point>`	Mount a file system on device <dev> at <mount_point> with the options specified by <dev>.
Unmount a file system	`umount <dev>` `umount <mount_point>`	Unmount the file system or device.
Maximum # of user ID	`65535`	

Recover root password	`{lilo}`
	`control-x`
	`linux S`
	`passwd root`
	
	`{grub}`
	`c`
	`kernel vmlinuz-2.4.9-13 single ro root=/dev/hda8`
	`initrd /initrd-2.4.9-13.img`
	`boot`
	`passwd root`
		This is a procedure to recover the root password if is lost. This requires physical access to the machine and system console. You start by rebooting the machine, then during the LILO boot press and hold [Ctrl] + [x] to get a prompt and command LILO to boot linux to runlevel S.
		
		The second procedure uses the grub boot loader.

Create new user	`useradd`	The –D option alone shows the defaults. –D with other options changes the defaults options; without –D override, the default (e.g., –g) sets a primary group.

Delete user	`userdel`	Remove a user and optionally all files belonging to the user.
Modify user account	`usermod`	Change `/etc/password` information.
Create new group	`groupadd`	–g sets the group id; default is first free value above 500.
Delete group	`groupdel`	Remove a group from the system. May not remove a group that is a primary group for a user. Files owned by deleted group must be manually changed with chown.
Change run levels	`init <runlevel>`	The `init` command causes the rcN.d scripts to be evaluated, for the change in run level. `init 6` forces a reboot.
Synchronize the disks	`sync`	Forces the buffer cache and page cache to write all dirty buffers to disk. Used just before a reboot to prevent disk corruption.
Shutdown the Linux system	`shutdown <mode> <delay>`	Do a graceful shutdown of the system, shut down processes, run all shutdown scripts, and sync disks. The modes are –r, reboot and –h, halt. The delay is a required parameter is a number in seconds or 'now'. Option shutdown warning message may be sent as well.
Error logs	`dmesg`	View boot messages. This log is circular, and limited system errors could overwrite boot information after a time.
Network IP configuration	`/etc/sysconfig/network-scripts/`	This directory holds scripts executed as part of the boot up sequence by `rc.sysinit`.
Hosts IP addresses	`/etc/hosts`	A list of hosts that your machine knows about. Must at minimum include the name of the local machine and loopback IP.
Name service switch	`/etc/nsswitch.conf`	

<table>
<tr><td>Network parameters</td><td>sysctl -a | grep net</td><td>View all net parameters that are set for the kernel.</td></tr>
<tr><td>Routing daemon</td><td>routed</td><td></td></tr>
<tr><td>NIC Configurations</td><td>ifconfig -a</td><td>Show all the network devices currently configured.</td></tr>
<tr><td rowspan="2">Secondary IP Address</td><td>modprobe ip_alias</td><td rowspan="2"></td></tr>
<tr><td>ifconfig eth0:1 IP</td></tr>
<tr><td>Login prompt</td><td>/etc/issue</td><td>Banner message user sees when issued the login prompt.</td></tr>
<tr><td>YP/NIS service binder</td><td>/sbin/ypbind</td><td>Finds and attaches to a NIS server for name resolution and other services.</td></tr>
<tr><td>Module information</td><td>modinfo <options> <module></td><td>Display information about kernel modules: –l shows license, –p parameters, –d description.</td></tr>
<tr><td>List modules</td><td>lsmod</td><td>Show currently loaded modules.</td></tr>
<tr><td>Load module</td><td>insmod</td><td>Load a loadable module.</td></tr>
<tr><td>Unload module</td><td>rmmod</td><td>Unload a loadable module.</td></tr>
<tr><td>Install Software</td><td>rpm -ivh package</td><td>Install –i, verbose –v, with progress hash marks –h.</td></tr>
<tr><td>Uninstall software</td><td>rpm -e package</td><td>Erase package –e; will not uninstall if dependencies exist.</td></tr>
<tr><td>List installed software</td><td>rpm -qa</td><td>Query –q, All –a, lists all installed packages.</td></tr>
<tr><td>Verify installed software</td><td>rpm -V package</td><td>Compares installed files with the rpm database information.</td></tr>
<tr><td>List all files</td><td>rpm -ql package</td><td>List all the files that are part of a package.</td></tr>
<tr><td>Package owner</td><td>rpm -qf file</td><td>List the package when given the full file name.</td></tr>
<tr><td>Machine model</td><td>uname -m</td><td>Shows CPU level (e.g., i686).</td></tr>
<tr><td>OS Level</td><td>uname -r</td><td>Shows kernel version.</td></tr>
<tr><td>Run Level</td><td>runlevel</td><td>Shows previous and current runlevel.</td></tr>
<tr><td>Kernel Parameters</td><td>sysctl -a</td><td>Show settings of all settable kernel parameters.</td></tr>
<tr><td>Max # File Descriptors</td><td>sysctl fs.file-max</td><td>Shows the value of maximum number of file descriptor per process.</td></tr>
</table>

Kernel parameter settings	/etc/sysctl.conf	Compiled in kernel parameters; may be reset at bootup by setting them in this file.
Change Kernel Parameter	echo <value> > </proc/<file>	Write the new value of a kernel parameter into the /proc file system.
	echo 2147483648 >/proc/sys/kernel/shmmax	Set the value of the maximum size of a shared memory segment.
Shared Memory	sysctl kernel.shmmax	Show the shmmax parameter.
Change Kernel Parameter	sysctl –w <parameter>=<value>	Change a kernel parameter; the –p option reads the setting from a file and sets them. The default file is /etc/sysctl.conf
Set Process limits	ulimit <option> <value>	Set limits on a shell and processes started by the shell. Users can make limits more restrictive; generally only root can make limit less restrictive; some options require root privilege. Options: –u sets number of processes, –n number of file handles; many others (see man bash).
Show process limits	ulimit	Without options ulimit show the current limit settings.
Interprocess Communication (Shared Memory and Semaphores)	ipcs <option>	Options: –m the current usage of shared memory; –s usage of semaphores; –a shows all.
Remove a shared memory segment	ipcrm shm <shmid>	Releases the shared memory segment identified by <shmid>. *This is very dangerous*. You can corrupt a database that is using the segment that is released.

System Performance	Linux Commands	Description / Comments
Performance monitor	`top`	View real-time OS and process statistics.
System activity reporter	`sar –<options> <interval> <count>`	Options: –q shows CPU queue, –u CPU utilization, –d device activity, –n DEV network device activity, many more (see `man` page). Interval is in seconds.
Virtual Memory statistics	`vmstat <interval> < count>`	Interval is in seconds.
Virtual Memory statistics	`cat /proc/meminfo`	Shows instantaneous virtual memory usage.
Kernel Cache statistics	`cat /proc/slabinfo`	Kernel slab allocator statistics: frequently allocated cache objects such as inode, dentries, and asynchronous IO buffers.
I/O statistics	`iostat <option> <interval> <count>`	Options: –d device activity, –c CPU activity, –x extended disk activity statistics. The interval is in seconds.
Multiprocessor Statistics	`mpstat –P <cpu> <count> <interval>`	Return CPU statistics for particular processor or *all* CPUs in an smp system.
Physical RAM	`64 GB(Theoretical)`	Maximum physical RAM requires enterprise kernel (Red Hat Enterprise Linux AS 21 only supports up to 16 GB).
Swap device	`swapon -s`	Shows devices currently in use for swap. The swap device is arbitrary designated at install. It may be changed or added to. Multiple swap devices may be created; swap size should be at least as large as physical memory.

Display swap size	`free`	Show the current memory and swap usage.
Activate Swap	`swapon -a`	Turn on swap.
Free disk blocks	`df -k`	Measured in KB; use –m for MB units.
Device listing	`cat /proc/devices`	List devices known to the system by major and minor number.

Disk information	`cat /proc/scsi/scsi0/sda/model`
	
	`cat /proc/ide/ide0/hda/model`
		View SCSI disk information.
		
		View IDE disk information.

Print network statistics	`netstat <options>`	Print a wide variety of network statistics (see `man netstat`).
Graphical system statistics viewer	`xosview`	An X-based display of recent OS statistics.

Misc System Information	**Linux Commands**	**Description / Comments**
NFS exported	/etc/exports	Database file are not supported on simple NFS.
NFS Client mounted directories	/var/lib/nfs/xtab	
Max File System	2 TB with 4KB block size (on 32 kernel)	With ext3 and ext2, others vary.
Max File Size File size can not exceed file system	2 GB {512B block size} 2 TB {4KB block size}	The oracle database can create files up to 64 GB with a 16 KB database block size. The 32-bit kernel limits file and block devices to 2 TB.
File System Block size	dumpe2fs <device>	Dump the file system properties to stdout.
Filesystem table	/etc/fstab	Mounts these file systems at boot up.
Journal Filesystem types	ext3 reiserfs	
Disk Label	fdisk -l	fdisk is not available on all distributions.
Extend File system	resize2fs resize_reiserfs	Extending a file system is applicable to only some file system types.
Backup	tar cvf /dev/rst0 /	Create a backup of the root / file system.
Restore	tar xvf /dev/rst0	Restore the root / file system.
Prepare boot volumes	/sbin/lilo	Must be run after changing /etc/lilo.conf to push changes to boot loader.
Startup script	/etc/rc.d/rc	
Kernel	/boot/vmlinuz	
Kernel Bits	getconf WORD_BIT	POSIX call to get kernel information. There are many other variables besides WORD_BIT.

Boot single user	{lilo} control-x linux S {grub} c kernel vmlinuz-2.4.9-13 **single** ro root=/dev/hda8 initrd /initrd-2.4.9-13.img boot	Use LILO facility. Use GRUB Boot Loader.
Time zone Management	/etc/sysconfig/clock	
SW Directory	/var/lib/rpm	Directory where rpm database are kept.
Devices	/dev	This directory holds all the device files.
CPU	cat /proc/cpuinfo	Shows CPU static information.
Whole Disk	/dev/sda	Device name.
CDROM	/dev/cdrom	Usually mounted at `/mnt/cdrom`.
CDROM file type	iso9660	
Floppy drive	/dev/fd0	Usually mounted at `/mnt/floppy`.
System information	/proc	The /proc filesystem is a memory-based file system that allows access to process and kernel settings and statistics.
Compile and link a executable	make –f <file> <command>	Use a make file <file> to determine which parts of a large program need to be recompiled, and issue the commands required to compile, link, and prepare the executable for use.

LVM	**Linux (UnitedLinux)**	**Description / Comments**
LVM	`Logical Volume Manager`	This package is not provided by Red Hat Enterprise Linux AS 2.1 and may not be added without tainting the kernel. Kernel support is provided in United Linux.
LVM Concepts	`logical extents`	A Logical volume is made up of logical extents.
	`logical volume`	A set of logical extents taken from a volume group and presented to the OS as a disk volume. These extents may be striped across multiple disks.
	`volume group`	A set of physical disk partitions created by `fdisk` or the like, initialized with `pvcreate`, then grouped into a physical volume with `vgcreate`.
Display volume group	`vgdisplay -v`	
Modify physical volume	`pvchange`	
Prepare physical disk	`pvcreate`	
List physical volume	`pvdisplay`	
Remove disk from volume group	`vgreduce`	
Move logical volumes to another physical volumes	`pvmove`	
Create volume group	`vgcreate`	
Remove volume group	`vgremove`	
Volume group availability	`vgchange`	
Restore volume group	`vgcfgrestore`	

Exports volume group	`vgexport`	
Imports volume group	`vgimport`	
Volume group listing	`vgscan`	
Change logical volume characteristics	`lvchange`	
List logical volume	`lvdisplay`	
Make logical volume	`lvcreate`	
Extend logical volume	`lvextend`	
Reduce logical volume	`lvreduce`	
Remove logical volume	`lvremove`	
Create striped volumes	`lvcreate -i 3 -I 64`	

Appendix D
SQL Statement Syntax

SQL Statements

This excerpt from the Oracle Database SQL Quick Reference guide presents the syntax for Oracle SQL statements. SQL statements are the means by which programs and users access data in an Oracle database.

Table 1 shows each SQL statement and its related syntax. Table 2 shows the syntax of the subclauses found in the table 1.

See Also: Oracle Database SQL Reference for detailed information about Oracle SQL

Table 1: Syntax for SQL Statements

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td><code>ALTER CLUSTER</code></td><td><pre>
ALTER CLUSTER [schema.]cluster
  { physical_attributes_clause
  | SIZE size_clause
  | allocate_extent_clause
  | deallocate_unused_clause
  | { CACHE | NOCACHE }
  }
    [physical_attributes_clause
    | SIZE size_clause
    | allocate_extent_clause
    | deallocate_unused_clause
    | { CACHE | NOCACHE }
    ]...
  [parallel_clause] ;
</pre></td></tr>
<tr><td><code>ALTER DATABASE</code></td><td><pre>
ALTER DATABASE [database]
  { startup_clauses
  | recovery_clauses
  | database_file_clauses
  | logfile_clauses
  | controlfile_clauses
  | standby_database_clauses
  | default_settings_clauses
  | redo_thread_clauses
  | security_clause
  } ;
</pre></td></tr>
<tr><td><code>ALTER DIMENSION</code></td><td><pre>
ALTER DIMENSION [schema.]dimension
  { ADD
    { level_clause
    | hierarchy_clause
    | attribute_clause
    | extended_attribute_clause
    }
    [ADD
</pre></td></tr>
</table>

SQL Statement	Syntax

	{ level_clause
	      | hierarchy_clause
	      | attribute_clause
	      | extended_attribute_clause
	      }
	    ]...
	  | DROP
	    { LEVEL level
	        [RESTRICT | CASCADE]
	    | HIERARCHY hierarchy
	    | ATTRIBUTE attribute
	        [LEVEL level [COLUMN column
	                        [, COLUMN column]...]
	    }
	    [DROP
	      { LEVEL level
	          [RESTRICT | CASCADE]
	      | HIERARCHY hierarchy
	      | ATTRIBUTE attribute
	          [LEVEL level [COLUMN column
	                          [, COLUMN column]...]
	      }
	    ]...
	  | COMPILE
	  } ;

ALTER DISKGROUP	ALTER DISKGROUP
	  { disk_clauses | diskgroup_clauses }
	    [{ disk_clauses | diskgroup_clauses }]... ;

ALTER FUNCTION	ALTER FUNCTION [schema.]function
	  COMPILE [DEBUG]
	  [compiler_parameters_clause
	    [compiler_parameters_clause] ...]
	  [REUSE SETTINGS] ;

ALTER INDEX	ALTER INDEX [schema.]index
	  { { deallocate_unused_clause
	    | allocate_extent_clause
	    | shrink_clause
	    | parallel_clause
	    | physical_attributes_clause
	    | logging_clause
	    }
	      [deallocate_unused_clause
	      | allocate_extent_clause
	      | shrink_clause
	      | parallel_clause
	      | physical_attributes_clause
	      | logging_clause
	      ]...
	  | rebuild_clause

SQL Statement	Syntax
	| PARAMETERS ('ODCI_parameters') | { ENABLE | DISABLE } | UNUSABLE | RENAME TO new_name | COALESCE | { MONITORING | NOMONITORING } USAGE | UPDATE BLOCK REFERENCES | alter_index_partitioning } ;
ALTER INDEXTYPE	ALTER INDEXTYPE [schema.]indextype { { ADD | DROP } [schema.]operator (parameter_types) [, { ADD | DROP } [schema.]operator (parameter_types)]... [using_type_clause] | COMPILE } ;
ALTER JAVA	ALTER JAVA { SOURCE | CLASS } [schema.]object_name [RESOLVER ((match_string [,] { schema_name | - }) [(match_string [,] { schema_name | - })]...)] { { COMPILE | RESOLVE } | invoker_rights_clause } ;
ALTER MATERIALIZED VIEW	ALTER MATERIALIZED VIEW [schema.](materialized_view) [physical_attributes_clause | table_compression | LOB_storage_clause [, LOB_storage_clause]... | modify_LOB_storage_clause [, modify_LOB_storage_clause]... | alter_table_partitioning | parallel_clause | logging_clause | allocate_extent_clause | shrink_clause | { CACHE | NOCACHE }] [alter_iot_clauses] [USING INDEX physical_attributes_clause] [MODIFY scoped_table_ref_constraint | alter_mv_refresh]

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td></td><td><pre>
  [{ ENABLE | DISABLE } QUERY REWRITE
  | COMPILE
  | CONSIDER FRESH
  ] ;
</pre></td></tr>
<tr><td>ALTER MATERIALIZED VIEW LOG</td><td><pre>
ALTER MATERIALIZED VIEW LOG [FORCE]
  ON [schema.]table
  [physical_attributes_clause
  | alter_table_partitioning
  | parallel_clause
  | logging_clause
  | allocate_extent_clause
  | shrink_clause
  | { CACHE | NOCACHE }
  ]
  [ADD
      { { OBJECT ID
        | PRIMARY KEY
        | ROWID
        | SEQUENCE
        }
        [(column [, column]...)]
      | (column [, column]...)
      }
        [, { { OBJECT ID
             | PRIMARY KEY
             | ROWID
             | SEQUENCE
             }
             [(column [, column]...)]
           | (column [, column]...)
           }
        ]...
      [new_values_clause]
  ] ;
</pre></td></tr>
<tr><td>ALTER OPERATOR</td><td><pre>
ALTER OPERATOR [schema.]operator
  { add_binding_clause
  | drop_binding_clause
  | COMPILE
  } ;
</pre></td></tr>
<tr><td>ALTER OUTLINE</td><td><pre>
ALTER OUTLINE
  [PUBLIC | PRIVATE] outline
  { REBUILD
  | RENAME TO new_outline_name
  | CHANGE CATEGORY TO new_category_name
  | { ENABLE | DISABLE }
  }
    [REBUILD
    | RENAME TO new_outline_name
    | CHANGE CATEGORY TO new_category_name
</pre></td></tr>
</table>

SQL Statement	Syntax
	| { ENABLE | DISABLE }]... ;
ALTER PACKAGE	ALTER PACKAGE [schema.]package COMPILE [DEBUG] [PACKAGE | SPECIFICATION | BODY] [compiler_parameters_clause [compiler_parameters_clause] ...] [REUSE SETTINGS] ;
ALTER PROCEDURE	ALTER PROCEDURE [schema.]procedure COMPILE [DEBUG] [compiler_parameters_clause [compiler_parameters_clause] ...] [REUSE SETTINGS] ;
ALTER PROFILE	ALTER PROFILE profile LIMIT { resource_parameters | password_parameters } [resource_parameters | password_parameters]... ;
ALTER RESOURCE COST	ALTER RESOURCE COST { CPU_PER_SESSION | CONNECT_TIME | LOGICAL_READS_PER_SESSION | PRIVATE_SGA } integer [{ CPU_PER_SESSION | CONNECT_TIME | LOGICAL_READS_PER_SESSION | PRIVATE_SGA } integer] ... ;
ALTER ROLE	ALTER ROLE role { NOT IDENTIFIED | IDENTIFIED { BY password | USING [schema.]package | EXTERNALLY | GLOBALLY } } ;
ALTER ROLEBACK SEGMENT	ALTER ROLLBACK SEGMENT rollback_segment { ONLINE | OFFLINE | storage_clause

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td></td><td><pre>
  | SHRINK [TO integer [K | M]]
  };
</pre></td></tr>
<tr><td>ALTER SEQUENCE</td><td><pre>
ALTER SEQUENCE [schema.]sequence
  { INCREMENT BY integer
  | { MAXVALUE integer | NOMAXVALUE }
  | { MINVALUE integer | NOMINVALUE }
  | { CYCLE | NOCYCLE }
  | { CACHE integer | NOCACHE }
  | { ORDER | NOORDER }
  }
    [INCREMENT BY integer
    | { MAXVALUE integer | NOMAXVALUE }
    | { MINVALUE integer | NOMINVALUE }
    | { CYCLE | NOCYCLE }
    | { CACHE integer | NOCACHE }
    | { ORDER | NOORDER }
    ]... ;
</pre></td></tr>
<tr><td>ALTER SESSION</td><td><pre>
ALTER SESSION
  { ADVISE { COMMIT | ROLLBACK | NOTHING }
  | CLOSE DATABASE LINK dblink
  | { ENABLE | DISABLE } COMMIT IN PROCEDURE
  | { ENABLE | DISABLE } GUARD
  | { ENABLE | DISABLE | FORCE } PARALLEL
    { DML | DDL | QUERY } [PARALLEL integer]
  | { ENABLE RESUMABLE
      [TIMEOUT integer] [NAME string]
    | DISABLE RESUMABLE
    }
  | alter_session_set_clause
  } ;
</pre></td></tr>
<tr><td>ALTER SYSTEM</td><td><pre>
ALTER SYSTEM
  { archive_log_clause
  | checkpoint_clause
  | check_datafiles_clause
  | DUMP ACTIVE SESSION HISTORY [MINUTES integer]
  | distributed_recov_clauses
  | restricted_session_clauses
  | FLUSH { SHARED_POOL | BUFFER_CACHE }
  | end_session_clauses
  | SWITCH LOGFILE
  | { SUSPEND | RESUME }
  | quiesce_clauses
  | shutdown_dispatcher_clause
  | REGISTER
  | SET alter_system_set_clause
        [alter_system_set_clause]...
  | RESET alter_system_reset_clause
          [alter_system_reset_clause]...
  } ;
</pre></td></tr>
</table>

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td>ALTER TABLE</td><td><pre>ALTER TABLE [schema.]table
  [alter_table_properties
  | column_clauses
  | constraint_clauses
  | alter_table_partitioning
  | alter_external_table_clauses
  | move_table_clause
  ]
  [enable_disable_clause
  | { ENABLE | DISABLE }
    { TABLE LOCK | ALL TRIGGERS }
    [enable_disable_clause
    | { ENABLE | DISABLE }
      { TABLE LOCK | ALL TRIGGERS }
    ]...
  ] ;</pre></td></tr>
<tr><td>ALTER TABLESPACE</td><td><pre>ALTER TABLESPACE tablespace
  { DEFAULT
      [table_compression] storage_clause
  | MINIMUM EXTENT integer [K | M]
  | RESIZE size_clause
  | COALESCE
  | RENAME TO new_tablespace_name
  | { BEGIN | END } BACKUP
  | datafile_tempfile_clauses
  | tablespace_logging_clauses
  | tablespace_group_clause
  | tablespace_state_clauses
  | autoextend_clause
  | flashback_mode_clause
  | tablespace_retention_clause
  } ;</pre></td></tr>
<tr><td>ALTER TRIGGER</td><td><pre>ALTER TRIGGER [schema.]trigger
  { ENABLE
  | DISABLE
  | RENAME TO new_name
  | COMPILE [DEBUG]
      [compiler_parameters_clause
        [compiler_parameters_clause] ...]
      [REUSE SETTINGS]
  } ;</pre></td></tr>
<tr><td>ALTER TYPE</td><td><pre>ALTER TYPE [schema.]type
  { compile_type_clause
  | replace_type_clause
  | { alter_method_spec
    | alter_attribute_definition
    | alter_collection_clauses
    | [NOT] { INSTANTIABLE | FINAL }
    }</pre></td></tr>
</table>

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td></td><td>

```
    [ dependent_handling_clause ]
  } ;
```

</td></tr>
<tr><td>ALTER USER</td><td>

```
ALTER USER
  { user
    { IDENTIFIED
      { BY password [ REPLACE old_password ]
      | EXTERNALLY
      | GLOBALLY AS 'external_name'
      }
    | DEFAULT TABLESPACE tablespace
    | TEMPORARY TABLESPACE
         { tablespace | tablespace_group_name }
    | QUOTA { integer [ K | M ]
            | UNLIMITED
            } ON tablespace
      [ QUOTA { integer [ K | M ]
              | UNLIMITED
              } ON tablespace
      ]...
    | PROFILE profile
    | DEFAULT ROLE { role [, role ]...
                   | ALL [ EXCEPT
                            role [, role ]... ]
                   | NONE
                   }
    | PASSWORD EXPIRE
    | ACCOUNT { LOCK | UNLOCK }
    }
(continued)
UPDATE MANUALLY too big to fit on page, split manually
```

</td></tr>
<tr><td>(cont.) ALTER USER</td><td>

```
      [  { IDENTIFIED
           { BY password [ REPLACE old_password ]
           | EXTERNALLY
           | GLOBALLY AS 'external_name'
           }
         | DEFAULT TABLESPACE tablespace
         | TEMPORARY TABLESPACE
              { tablespace | tablespace_group_name }
         | QUOTA { integer [ K | M ]
                 | UNLIMITED
                 } ON tablespace
           [ QUOTA { integer [ K | M ]
                   | UNLIMITED
                   } ON tablespace
           ]...
         | PROFILE profile
         | DEFAULT ROLE { role [, role ]...
                        | ALL [ EXCEPT
                                 role [, role ]... ]
                        | NONE
                        }
         | PASSWORD EXPIRE
```

</td></tr>
</table>

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td></td><td><pre> | ACCOUNT { LOCK | UNLOCK }
        }
     ]...
  | user [, user]... proxy_clause ;
UPDATE MANUALLY too big to fit on page, split manually</pre></td></tr>
<tr><td>ALTER VIEW</td><td><pre>ALTER VIEW [schema.]view
  { ADD out_of_line_constraint
  | MODIFY CONSTRAINT constraint
      { RELY | NORELY }
  | DROP { CONSTRAINT constraint
         | PRIMARY KEY
         | UNIQUE (column [, column]...)
         }
  | COMPILE
  } ;</pre></td></tr>
<tr><td>ANALYZE</td><td><pre>ANALYZE
  { TABLE [schema.]table
      [PARTITION (partition)
      | SUBPARTITION (subpartition)
      ]
  | INDEX [schema.]index
      [PARTITION (partition)
      | SUBPARTITION (subpartition)
      ]
  | CLUSTER [schema.]cluster
  }
  { validation_clauses
  | LIST CHAINED ROWS [into_clause]
  | DELETE [SYSTEM] STATISTICS
  | compute_statistics_clause
  | estimate_statistics_clause
  } ;</pre></td></tr>
<tr><td>ASSOCIATE STATISTICS</td><td><pre>ASSOCIATE STATISTICS WITH
  { column_association | function_association } ;</pre></td></tr>
<tr><td>AUDIT</td><td><pre>AUDIT
  { sql_statement_clause | schema_object_clause }
  [BY { SESSION | ACCESS }]
  [WHENEVER [NOT] SUCCESSFUL] ;</pre></td></tr>
<tr><td>CALL</td><td><pre>CALL
  { routine_clause
  | object_access_expression
  }
  [INTO :host_variable
    [[INDICATOR] :indicator_variable]] ;</pre></td></tr>
<tr><td>COMMENT</td><td><pre>COMMENT ON</pre></td></tr>
</table>

SQL Statement	Syntax

	{ TABLE [schema.]
	  { table | view }
	| COLUMN [schema.]
	  { table. | view. | materialized_view. } column
	| OPERATOR [schema.] operator
	| INDEXTYPE [schema.] indextype
	| MATERIALIZED VIEW materialized_view
	}
	IS 'text' ;

COMMIT	COMMIT [WORK]
	  [COMMENT 'text'
	  | FORCE 'text' [, integer]
	  ] ;

CREATE CLUSTER	CREATE CLUSTER [schema.]cluster
	  (column datatype [SORT]
	    [, column datatype [SORT]]...
	  )
	  [{ physical_attributes_clause
	    | SIZE size_clause
	    | TABLESPACE tablespace
	    | { INDEX
	      | [SINGLE TABLE]
	        HASHKEYS integer [HASH IS expr]
	      }
	    }
	      [physical_attributes_clause
	      | SIZE size_clause
	      | TABLESPACE tablespace
	      | { INDEX
	        | [SINGLE TABLE]
	          HASHKEYS integer [HASH IS expr]
	        }
	      ]...
	  ]
	  [parallel_clause]
	  [NOROWDEPENDENCIES | ROWDEPENDENCIES]
	  [CACHE | NOCACHE] ;

CREATE CONTEXT	CREATE [OR REPLACE] CONTEXT namespace
	  USING [schema.] package
	  [INITIALIZED { EXTERNALLY | GLOBALLY }
	  | ACCESSED GLOBALLY
	  ] ;

CREATE CONTROLFILE	CREATE CONTROLFILE
	  [REUSE]
	  [SET]
	  DATABASE database
	  [logfile_clause]

SQL Statement	Syntax
	{ RESETLOGS | NORESETLOGS } [DATAFILE file_specification [, file_specification]...] [{ MAXLOGFILES integer | MAXLOGMEMBERS integer | MAXLOGHISTORY integer | MAXDATAFILES integer | MAXINSTANCES integer | { ARCHIVELOG | NOARCHIVELOG } | FORCE LOGGING } [MAXLOGFILES integer | MAXLOGMEMBERS integer | MAXLOGHISTORY integer | MAXDATAFILES integer | MAXINSTANCES integer | { ARCHIVELOG | NOARCHIVELOG } | FORCE LOGGING]...] [character_set_clause] ;
CREATE DATABASE	CREATE DATABASE [database] { USER SYS IDENTIFIED BY password | USER SYSTEM IDENTIFIED BY password | CONTROLFILE REUSE | MAXDATAFILES integer | MAXINSTANCES integer | CHARACTER SET charset | NATIONAL CHARACTER SET charset | SET DEFAULT { BIGFILE | SMALLFILE } TABLESPACE | database_logging_clauses | tablespace_clauses | set_time_zone_clause }... ;
CREATE DATABASE LINK	CREATE [SHARED] [PUBLIC] DATABASE LINK dblink [CONNECT TO { CURRENT_USER | user IDENTIFIED BY password [dblink_authentication] } | dblink_authentication] [USING 'connect_string'] ;
CREATE DIMENSION	CREATE DIMENSION [schema.]dimension level_clause [level_clause]... { hierarchy_clause | attribute_clause

SQL Statement	Syntax

	`| extended_attribute_clause`
	`}`
	`[ hierarchy_clause`
	`| attribute_clause`
	`| extended_attribute_clause`
	`]... ;`

CREATE DIRECTORY	`CREATE [ OR REPLACE ] DIRECTORY directory` `AS 'path_name' ;`
CREATE DISKGROUP	`CREATE DISKGROUP diskgroup_name` `[{ HIGH	NORMAL	EXTERNAL } REDUNDANCY]` `[FAILGROUP failgroup_name]` `DISK qualified_disk_clause` `[, qualified_disk_clause]...` `[[FAILGROUP failgroup_name]` `DISK qualified_disk_clause` `[, qualified_disk_clause]...` `]... ;`
CREATE FUNCTION	`CREATE [ OR REPLACE ] FUNCTION [ schema. ]function` `[ (argument [ IN | OUT | IN OUT ]` `[ NOCOPY ] datatype` `[, argument [ IN | OUT | IN OUT ]` `[ NOCOPY ] datatype` `]...` `)` `]` `RETURN datatype` `[ { invoker_rights_clause` `| DETERMINISTIC` `| parallel_enable_clause` `}` `[ invoker_rights_clause` `| DETERMINISTIC` `| parallel_enable_clause` `]...` `]` `{ { AGGREGATE	PIPELINED }` `USING [schema.]implementation_type` `| [PIPELINED]` `{ IS	AS }` `{ pl/sql_function_body	call_spec }` `} ;`
CREATE INDEX	`CREATE [UNIQUE	BITMAP] INDEX [schema.]index` `ON { cluster_index_clause` `	table_index_clause` `	bitmap_join_index_clause` `} ;`
CREATE INDEXTYPE	`CREATE [ OR REPLACE ] INDEXTYPE`

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td></td><td><pre>
  [schema.]indextype FOR
  [schema.]operator (paramater_type
                      [, paramater_type]...)
    [, [schema.]operator (paramater_type
                           [, paramater_type]...)
    ]...
  using_type_clause ;
</pre></td></tr>
<tr><td>CREATE JAVA</td><td><pre>
CREATE [OR REPLACE]
  [AND { RESOLVE | COMPILE }]
  [NOFORCE]
  JAVA { { SOURCE | RESOURCE }
         NAMED [schema.]primary_name
       | CLASS [SCHEMA schema]
       }
  [invoker_rights_clause]
  [RESOLVER
    ((match_string [,] { schema_name | - })
      [(match_string [,] { schema_name | - })]...
)
  ]
  { USING { BFILE (directory_object_name ,
                   server_file_name)
          | { CLOB | BLOB | BFILE }
            subquery
          | 'key_for_BLOB'
          }
  | AS source_text
  } ;
</pre></td></tr>
<tr><td>CREATE LIBRARY</td><td><pre>
CREATE [OR REPLACE] LIBRARY [schema.]libname
  { IS | AS } 'filename' [AGENT 'agent_dblink'] ;
</pre></td></tr>
<tr><td>CREATE MATERIALIZED VIEW</td><td><pre>
CREATE MATERIALIZED VIEW
  [schema.]materialized_view
  [OF [schema.]object_type]
  [(scoped_table_ref_constraint)]
  { ON PREBUILT TABLE
    [{ WITH | WITHOUT } REDUCED PRECISION]
  | physical_properties materialized_view_props
  }
  [USING INDEX
    [physical_attributes_clause
    | TABLESPACE tablespace
    ]
      [physical_attributes_clause
      | TABLESPACE tablespace
      ]...
  | USING NO INDEX
  ]
  [create_mv_refresh]
  [FOR UPDATE]
</pre></td></tr>
</table>

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td></td><td>[{ DISABLE | ENABLE }
QUERY REWRITE
]
AS subquery ;</td></tr>
<tr><td>CREATE MATERIALIZED VIEW LOG</td><td>CREATE MATERIALIZED VIEW LOG
ON [schema.] table
[physical_attributes_clause
| TABLESPACE tablespace
| logging_clause
| { CACHE | NOCACHE }
[physical_attributes_clause
| TABLESPACE tablespace
| logging_clause
| { CACHE | NOCACHE }
]...
]
[parallel_clause]
[table_partitioning_clauses]
[WITH { OBJECT ID
| PRIMARY KEY
| ROWID
| SEQUENCE
| (column [, column]...)
}
[, { OBJECT ID
| PRIMARY KEY
| ROWID
| SEQUENCE
| (column [, column]...)
}
]...
[new_values_clause]
] ;</td></tr>
<tr><td>CREATE OPERATOR</td><td>CREATE [OR REPLACE] OPERATOR
[schema.] operator binding_clause ;</td></tr>
<tr><td>CREATE OUTLINE</td><td>CREATE [OR REPLACE]
[PUBLIC | PRIVATE] OUTLINE [outline]
[FROM [PUBLIC | PRIVATE] source_outline]
[FOR CATEGORY category]
[ON statement] ;</td></tr>
<tr><td>CREATE PACKAGE</td><td>CREATE [OR REPLACE] PACKAGE [schema.]package
[invoker_rights_clause]
{ IS | AS } pl/sql_package_spec ;</td></tr>
<tr><td>CREATE PACKAGE BODY</td><td>CREATE [OR REPLACE] PACKAGE BODY
[schema.]package
{ IS | AS } pl/sql_package_body ;</td></tr>
</table>

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td>CREATE PFILE</td><td><pre>CREATE PFILE [= 'pfile_name']
   FROM SPFILE [= 'spfile_name'] ;</pre></td></tr>
<tr><td>CREATE PROCEDURE</td><td><pre>CREATE [OR REPLACE] PROCEDURE [schema.]procedure
   [(argument [IN | OUT | IN OUT]
               [NOCOPY]
               datatype
       [, argument [IN | OUT | IN OUT]
                   [NOCOPY]
                   datatype
       ]...
)
   ]
   [invoker_rights_clause]
   { IS | AS }
   { pl/sql_subprogram_body | call_spec } ;</pre></td></tr>
<tr><td>CREATE PROFILE</td><td><pre>CREATE PROFILE profile
   LIMIT { resource_parameters
         | password_parameters
         }
           [resource_parameters
           | password_parameters
           ]... ;</pre></td></tr>
<tr><td>CREATE ROLE</td><td><pre>CREATE ROLE role
   [NOT IDENTIFIED
   | IDENTIFIED { BY password
                | USING [schema.] package
                | EXTERNALLY
                | GLOBALLY
                }
   ] ;</pre></td></tr>
<tr><td>CREATE ROLEBACK SEGMENT</td><td><pre>CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment
  [{ TABLESPACE tablespace | storage_clause }
      [TABLESPACE tablespace | storage_clause]...
  ];</pre></td></tr>
<tr><td>CREATE SCHEMA</td><td><pre>CREATE SCHEMA AUTHORIZATION schema
   { create_table_statement
   | create_view_statement
   | grant_statement
   }
     [create_table_statement
     | create_view_statement
     | grant_statement
     ]... ;</pre></td></tr>
</table>

SQL Statement	Syntax
CREATE SEQUENCE	CREATE SEQUENCE [schema.]sequence [{ INCREMENT BY | START WITH } integer | { MAXVALUE integer | NOMAXVALUE } | { MINVALUE integer | NOMINVALUE } | { CYCLE | NOCYCLE } | { CACHE integer | NOCACHE } | { ORDER | NOORDER }] [{ INCREMENT BY | START WITH } integer | { MAXVALUE integer | NOMAXVALUE } | { MINVALUE integer | NOMINVALUE } | { CYCLE | NOCYCLE } | { CACHE integer | NOCACHE } | { ORDER | NOORDER }]... ;
CREATE SPFILE	CREATE SPFILE [= 'spfile_name'] FROM PFILE [= 'pfile_name'] ;
CREATE SYNONYM	CREATE [OR REPLACE] [PUBLIC] SYNONYM [schema.]synonym FOR [schema.]object [@ dblink] ;
CREATE TABLE	{ relational_table | object_table | XMLType_table }
CREATE TABLESPACE	CREATE [BIGFILE | SMALLFILE] { permanent_tablespace_clause | temporary_tablespace_clause | undo_tablespace_clause } ;
CREATE TRIGGER	CREATE [OR REPLACE] TRIGGER [schema.]trigger { BEFORE | AFTER | INSTEAD OF } { dml_event_clause | { ddl_event [OR ddl_event]... | database_event [OR database_event]... } ON { [schema.]SCHEMA | DATABASE } } [WHEN (condition)] { pl/sql_block | call_procedure_statement } ;
CREATE TYPE	{ create_incomplete_type | create_object_type | create_varray_type | create_nested_table_type }

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td>CREATE TYPE BODY</td><td><pre>CREATE [OR REPLACE] TYPE BODY [schema.]type_name
   { IS | AS }
   { subprogram_declaration
   | map_order_func_declaration
   }
     [; { subprogram_declaration
        | map_order_func_declaration
        }
     ]...
   END ;</pre></td></tr>
<tr><td>CREATE USER</td><td><pre>CREATE USER user
   IDENTIFIED { BY password
              | EXTERNALLY
              | GLOBALLY AS 'external_name'
              }
   [DEFAULT TABLESPACE tablespace
   | TEMPORARY TABLESPACE
        { tablespace | tablespace_group_name }
   | QUOTA { integer [K | M]
           | UNLIMITED
           }
           ON tablespace
     [QUOTA { integer [K | M]
             | UNLIMITED
             }
             ON tablespace
     ]...
   | PROFILE profile
   | PASSWORD EXPIRE
   | ACCOUNT { LOCK | UNLOCK }
     [DEFAULT TABLESPACE tablespace
     | TEMPORARY TABLESPACE
          { tablespace | tablespace_group_name }
     | QUOTA { integer [K | M]
             | UNLIMITED
             }
             ON tablespace
       [QUOTA { integer [K | M]
               | UNLIMITED
               }
               ON tablespace
       ]...
     | PROFILE profile
     | PASSWORD EXPIRE
     | ACCOUNT { LOCK | UNLOCK }
     ]...
  ] ;</pre></td></tr>
<tr><td>CREATE VIEW</td><td><pre>CREATE [OR REPLACE] [[NO] FORCE] VIEW
   [schema.]view
   [(alias [inline_constraint
              [inline_constraint]...]</pre></td></tr>
</table>

<table>
<tr><th>SQL Statement</th><th>Syntax</th></tr>
<tr><td></td><td>| out_of_line_constraint
[, alias [inline_constraint
[inline_constraint]...]
| out_of_line_constraint
]...
)
| object_view_clause
| XMLType_view_clause
]
AS subquery [subquery_restriction_clause] ;</td></tr>
<tr><td>DELETE</td><td>DELETE [hint]
[FROM]
{ dml_table_expression_clause
| ONLY (dml_table_expression_clause)
}
[t_alias]
[where_clause]
[returning_clause] ;</td></tr>
<tr><td>DISASSOCIATE STATISTICS</td><td>DISASSOCIATE STATISTICS FROM
{ COLUMNS [schema.]table.column
[, [schema.]table.column]...
| FUNCTIONS [schema.]function
[, [schema.]function]...
| PACKAGES [schema.]package
[, [schema.]package]...
| TYPES [schema.]type
[, [schema.]type]...
| INDEXES [schema.]index
[, [schema.]index]...
| INDEXTYPES [schema.]indextype
[, [schema.]indextype]...
}
[FORCE] ;</td></tr>
<tr><td>DROP CLUSTER</td><td>DROP CLUSTER [schema.]cluster
[INCLUDING TABLES [CASCADE CONSTRAINTS]] ;</td></tr>
<tr><td>DROP CONTEXT</td><td>DROP CONTEXT namespace ;</td></tr>
<tr><td>DROP DATABASE</td><td>DROP DATABASE ;</td></tr>
<tr><td>DROP DATABASE LINK</td><td>DROP [PUBLIC] DATABASE LINK dblink ;</td></tr>
<tr><td>DROP DIMENSION</td><td>DROP DIMENSION [schema.]dimension ;</td></tr>
<tr><td>DROP DIRECTORY</td><td>DROP DIRECTORY directory_name ;</td></tr>
</table>

SQL Statement	Syntax
DROP DISKGROUP	DROP DISKGROUP diskgroup_name [{ INCLUDING | EXCLUDING } CONTENTS] ;
DROP FUNCTION	DROP FUNCTION [schema.]function_name ;
DROP INDEX	DROP INDEX [schema.]index [FORCE] ;
DROP INDEXTYPE	DROP INDEXTYPE [schema.]indextype [FORCE] ;
DROP JAVA	DROP JAVA { SOURCE | CLASS | RESOURCE } [schema.]object_name ;
DROP LIBRARY	DROP LIBRARY library_name ;
DROP MATERIALIZED VIEW	DROP MATERIALIZED VIEW [schema.]materialized_view [PRESERVE TABLE] ;
DROP MATERIALIZED VIEW LOG	DROP MATERIALIZED VIEW LOG ON [schema.]table ;
DROP OPERATOR	DROP OPERATOR [schema.]operator [FORCE] ;
DROP OUTLINE	DROP OUTLINE outline ;
DROP PACKAGE	DROP PACKAGE [BODY] [schema.]package ;
DROP PROCEDURE	DROP PROCEDURE [schema.]procedure ;
DROP PROFILE	DROP PROFILE profile [CASCADE] ;
DROP ROLE	DROP ROLE role ;
DROP ROLLBACK SEGMENT	DROP ROLLBACK SEGMENT rollback_segment ;
DROP SEQUENCE	DROP SEQUENCE [schema.]sequence_name ;
DROP SYNONYM	DROP [PUBLIC] SYNONYM [schema.]synonym [FORCE] ;

SQL Statement	Syntax

DROP TABLE	`DROP TABLE [ schema. ]table`
	`   [ CASCADE CONSTRAINTS ]`
	`   [ PURGE ] ;`

DROP TABLESPACE	`DROP TABLESPACE tablespace`
	`   [ INCLUDING CONTENTS [ AND DATAFILES ]`
	`     [ CASCADE CONSTRAINTS ]`
	`   ] ;`

DROP TRIGGER	`DROP TRIGGER [ schema. ]trigger ;`
DROP TYPE	`DROP TYPE [ schema. ]type_name` `   [ FORCE | VALIDATE ] ;`
DROP TYPE BODY	`DROP TYPE BODY [ schema. ]type_name ;`
DROP USER	`DROP USER user [ CASCADE ] ;`
DROP VIEW	`DROP VIEW [ schema. ] view` `   [ CASCADE CONSTRAINTS ] ;`

EXPLAIN PLAN	`EXPLAIN PLAN`
	`   [ SET STATEMENT_ID = 'text' ]`
	`   [ INTO [ schema. ]table [ @ dblink ] ]`
	`   FOR statement ;`

FLASHBACK DATABASE	`FLASHBACK [ STANDBY ] DATABASE [ database ]`
	`   { TO { SCN | TIMESTAMP } expr`
	`   | TO BEFORE { SCN | TIMESTAMP } expr`
	`   };`

FLASHBACK TABLE	`FLASHBACK TABLE`
	`   [ schema. ]table`
	`     [, [ schema. ]table ]...`
	`   TO { { SCN | TIMESTAMP } expr`
	`        [ { ENABLE | DISABLE } TRIGGERS ]`
	`      | BEFORE DROP [ RENAME TO table ]`
	`      } ;`

GRANT	`GRANT { grant_system_privileges`
	`      | grant_object_privileges`
	`      } ;`

INSERT	`INSERT [ hint ]` `   { single_table_insert | multi_table_insert } ;`

SQL Statement	Syntax

LOCK TABLE	LOCK TABLE
	   [schema.] { table | view }
	   [{ PARTITION (partition)
	     | SUBPARTITION (subpartition)
	     }
	   | @ dblink
	   ]
	     [, [schema.] { table | view }
	        [{ PARTITION (partition)
	          | SUBPARTITION (subpartition)
	          }
	        | @ dblink
	        ]
	     ]...
	   IN lockmode MODE
	   [NOWAIT] ;

MERGE	MERGE [hint]
	   INTO [schema.]table [t_alias]
	   USING [schema.] { table | view | subquery }
	     [t_alias]
	   ON (condition)
	   [merge_update_clause]
	   [merge_insert_clause] ;

NOAUDIT	NOAUDIT
	   { sql_statement_clause
	     [, sql_statement_clause]...
	   | schema_object_clause
	     [, schema_object_clause]...
	   }
	   [WHENEVER [NOT] SUCCESSFUL] ;

PURGE	PURGE
	   { { TABLE table
	     | INDEX index
	     }
	   | { RECYCLEBIN | DBA_RECYCLEBIN }
	   | TABLESPACE tablespace
	     [USER user]
	   } ;
	DO NOT IMPORT AS INSET problem importing file, locks up FrameMaker

RENAME	RENAME old_name TO new_name ;

REVOKE	REVOKE { revoke_system_privileges
	       | revoke_object_privileges
	       } ;

SQL Statement	Syntax
ROLLBACK	ROLLBACK [WORK] [TO [SAVEPOINT] savepoint | FORCE 'text'] ;
SAVEPOINT	SAVEPOINT savepoint ;
SELECT	subquery [for_update_clause] ;
SET CONSTRAINT[S]	SET { CONSTRAINT | CONSTRAINTS } { constraint [, constraint]... | ALL } { IMMEDIATE | DEFERRED } ;
SET ROLE	SET ROLE { role [IDENTIFIED BY password] [, role [IDENTIFIED BY password]]... | ALL [EXCEPT role [, role]...] | NONE } ;
SET TRANSACTION	SET TRANSACTION { { READ { ONLY | WRITE } | ISOLATION LEVEL { SERIALIZABLE | READ COMMITTED } | USE ROLLBACK SEGMENT rollback_segment } [NAME 'text'] | NAME 'text' } ;
TRUNCATE	TRUNCATE { TABLE [schema.]table [{ PRESERVE | PURGE } MATERIALIZED VIEW LOG] | CLUSTER [schema.]cluster } [{ DROP | REUSE } STORAGE] ;
UPDATE	UPDATE [hint] { dml_table_expression_clause | ONLY (dml_table_expression_clause) } [t_alias] update_set_clause [where_clause] [returning_clause] ;

Table 2: Syntax for Subclauses

Subclause	Syntax
activate_standby_db_clause	ACTIVATE [PHYSICAL | LOGICAL] STANDBY DATABASE [SKIP [STANDBY LOGFILE]]
add_binding_clause	ADD BINDING (parameter_type [, parameter_type]...) RETURN (return_type) [implementation_clause] using_function_clause
add_column_clause	ADD (column datatype [DEFAULT expr] [{ inline_constraint [inline_constraint]... | inline_ref_constraint }] [, column datatype [DEFAULT expr] [{ inline_constraint [inline_constraint]... | inline_ref_constraint }]]...) [column_properties]
add_disk_clause	ADD [FAILGROUP failgroup_name] DISK qualified_disk_clause [, qualified_disk_clause]... [[FAILGROUP failgroup_name] DISK qualified_disk_clause [, qualified_disk_clause]...]...
add_hash_index_partition	ADD PARTITION [partition_name] [TABLESPACE tablespace_name] [parallel_clause]
add_hash_partition_clause	ADD PARTITION [partition] partitioning_storage_clause [update_index_clauses] [parallel_clause]

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>add_hash_subpartition</td><td><pre>ADD subpartition_spec
   [update_index_clauses]
   [parallel_clause]</pre></td></tr>
<tr><td>add_list_partition_clause</td><td><pre>ADD PARTITION [partition]
   list_values_clause
   [table_partition_description]
   [update_index_clauses]</pre></td></tr>
<tr><td>add_list_subpartition</td><td><pre>ADD subpartition_spec
   [update_index_clauses]</pre></td></tr>
<tr><td>add_logfile_clauses</td><td><pre>ADD [STANDBY] LOGFILE
   { [INSTANCE 'instance_name' | THREAD
integer]
     [GROUP integer] redo_log_file_spec
       [, [GROUP integer]
redo_log_file_spec]...
   | MEMBER 'filename' [REUSE]
            [, 'filename' [REUSE]]...
        TO logfile_descriptor
           [, logfile_descriptor]...
   }</pre></td></tr>
<tr><td>add_overflow_clause</td><td><pre>ADD OVERFLOW [segment_attributes_clause]
[(PARTITION [segment_attributes_clause]
   [, PARTITION
[segment_attributes_clause]]...
)
]</pre></td></tr>
<tr><td>add_range_partition_clause</td><td><pre>ADD PARTITION [partition]
   range_values_clause
   [table_partition_description]
   [update_index_clauses]</pre></td></tr>
<tr><td>add_table_partition</td><td><pre>{ add_range_partition_clause
| add_hash_partition_clause
| add_list_partition_clause
}</pre></td></tr>
<tr><td>alias_file_name</td><td><pre>+diskgroup_name [(template_name)] /alias_name</pre></td></tr>
<tr><td>allocate_extent_clause</td><td><pre>ALLOCATE EXTENT
  [({ SIZE size_clause
      | DATAFILE 'filename'
      | INSTANCE integer
      }</pre></td></tr>
</table>

Subclause	Syntax
	[SIZE size_clause | DATAFILE 'filename' | INSTANCE integer]...)]
alter_attribute_definition	{ { ADD | MODIFY } ATTRIBUTE { attribute [datatype] | (attribute datatype [, attribute datatype]...) } | DROP ATTRIBUTE { attribute | (attribute [, attribute]...) } }
alter_collection_clauses	MODIFY { LIMIT integer | ELEMENT TYPE datatype }
alter_datafile_clause	DATAFILE { 'filename' | filenumber } [, 'filename' | filenumber]... } { ONLINE | OFFLINE [FOR DROP] | RESIZE size_clause | autoextend_clause | END BACKUP }
alter_external_table_clauses	{ add_column_clause | modify_column_clauses | drop_column_clause | parallel_clause | external_data_properties | REJECT LIMIT { integer | UNLIMITED } | PROJECT COLUMN { ALL | REFERENCED } } [add_column_clause | modify_column_clauses | drop_column_clause | parallel_clause | external_data_properties | REJECT LIMIT { integer | UNLIMITED } | PROJECT COLUMN { ALL | REFERENCED }]...

Subclause	Syntax
alter_index_partitioning	{ modify_index_default_attrs | add_hash_index_partition | modify_index_partition | rename_index_partition | drop_index_partition | split_index_partition | coalesce_index_partition | modify_index_subpartition }
alter_iot_clauses	{ index_org_table_clause | alter_overflow_clause | alter_mapping_table_clauses | COALESCE }
alter_mapping_table_clauses	MAPPING TABLE { UPDATE BLOCK REFERENCES | allocate_extent_clause | deallocate_unused_clause }
alter_method_spec	{ ADD | DROP } { map_order_function_spec | subprogram_spec } [{ ADD | DROP } { map_order_function_spec | subprogram_spec }]...
alter_mv_refresh	REFRESH { { FAST | COMPLETE | FORCE } | ON { DEMAND | COMMIT } | { START WITH | NEXT } date | WITH PRIMARY KEY | USING { DEFAULT MASTER ROLLBACK SEGMENT | MASTER ROLLBACK SEGMENT rollback_segment } | USING { ENFORCED | TRUSTED } CONSTRAINTS }
alter_overflow_clause	{ OVERFLOW { allocate_extent_clause | deallocate_unused_clause } [allocate_extent_clause | deallocate_unused_clause

Subclause	Syntax
	]... | add_overflow_clause }
alter_session_set_clause	SET parameter_name = parameter_value [parameter_name = parameter_value]...
alter_system_reset_clause	parameter_name [SCOPE = { MEMORY | SPFILE | BOTH }] SID = 'sid'
alter_system_set_clause	parameter_name = parameter_value [, parameter_value]... [COMMENT 'text'] [DEFERRED] [SCOPE = { MEMORY | SPFILE | BOTH }] [SID = { 'sid' | * }]
alter_table_partitioning	{ modify_table_default_attrs | set_subpartition_template | modify_table_partition | modify_table_subpartition | move_table_partition | move_table_subpartition | add_table_partition | coalesce_table_partition | drop_table_partition | drop_table_subpartition | rename_partition_subpart | truncate_partition_subpart | split_table_partition | split_table_subpartition | merge_table_partitions | merge_table_subpartitions | exchange_partition_subpart }
alter_table_properties	{ { physical_attributes_clause | logging_clause | table_compression | supplemental_table_logging | allocate_extent_clause | deallocate_unused_clause | shrink_clause | { CACHE | NOCACHE } | upgrade_table_clause | records_per_block_clause | parallel_clause | row_movement_clause } [physical_attributes_clause

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td><pre>
    | logging_clause
    | table_compression
    | supplemental_table_logging
    | allocate_extent_clause
    | deallocate_unused_clause
    | shrink_clause
    | { CACHE | NOCACHE }
    | upgrade_table_clause
    | records_per_block_clause
    | parallel_clause
    | row_movement_clause
    ]...
| RENAME TO new_table_name
}
[alter_iot_clauses]
</pre></td></tr>
<tr><td>alter_tempfile_clause</td><td><pre>
TEMPFILE
   { 'filename' [, 'filename']...
   | filenumber [, filenumber]...
   }
   { RESIZE size_clause
   | autoextend_clause
   | DROP [INCLUDING DATAFILES]
   | ONLINE
   | OFFLINE
   }
</pre></td></tr>
<tr><td>alter_varray_col_properties</td><td><pre>
MODIFY VARRAY varray_item
   (modify_LOB_parameters)
</pre></td></tr>
<tr><td>analytic_clause</td><td><pre>
[query_partition_clause]
[order_by_clause [windowing_clause]]
</pre></td></tr>
<tr><td>archive_log_clause</td><td><pre>
ARCHIVE LOG
   [INSTANCE 'instance_name' | THREAD
integer]
   { { SEQUENCE integer
     | CHANGE integer
     | CURRENT [NOSWITCH]
     | GROUP integer
     | LOGFILE 'filename'
          [USING BACKUP CONTROLFILE]
     | NEXT
     | ALL
     | START
     }
     [TO 'location']
   | STOP
   }
</pre></td></tr>
</table>

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>array_DML_clause</td><td><pre>[WITH | WITHOUT]
ARRAY DML
[([schema.]type
   [, [schema.]varray_type])
    [, ([schema.]type
        [, [schema.]varray_type])...
]</pre></td></tr>
<tr><td>attribute_clause</td><td><pre>ATTRIBUTE level DETERMINES
   { dependent_column
   | (dependent_column
       [, dependent_column]...)
   }</pre></td></tr>
<tr><td>auditing_by_clause</td><td><pre>BY { proxy [, proxy]...
   | user [, user]...
   }</pre></td></tr>
<tr><td>auditing_on_clause</td><td><pre>ON { [schema.]object
   | DIRECTORY directory_name
   | DEFAULT
   }</pre></td></tr>
<tr><td>autoextend_clause</td><td><pre>AUTOEXTEND
   { OFF
   | ON [NEXT size_clause]
        [maxsize_clause]
   }</pre></td></tr>
<tr><td>binding_clause</td><td><pre>BINDING
   (parameter_type [, parameter_type]...)
   RETURN return_type
   [implementation_clause]
   using_function_clause
    [, (parameter_type [, parameter_type]...)
       RETURN return_type
       [implementation_clause]
       using_function_clause
    ]...</pre></td></tr>
<tr><td>bitmap_join_index_clause</td><td><pre>[schema.]table
   ([[schema.]table. | t_alias.]column
     [ASC | DESC]
       [, [[schema.]table. | t_alias.]column
          [ASC | DESC]
       ]...
)
   FROM [schema.]table [t_alias]
          [, [schema.]table [t_alias]
        ]...</pre></td></tr>
</table>

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td><pre>   WHERE condition
      [local_partitioned_index]
index_attributes</pre></td></tr>
<tr><td>build_clause</td><td><pre>BUILD { IMMEDIATE | DEFERRED }</pre></td></tr>
<tr><td>C_declaration</td><td><pre>C [NAME name]
   LIBRARY lib_name
   [AGENT IN (argument[, argument]...)]
   [WITH CONTEXT]
   [PARAMETERS (parameter[, parameter]...)]</pre></td></tr>
<tr><td>call_spec</td><td><pre>LANGUAGE { Java_declaration | C_declaration }</pre></td></tr>
<tr><td>cancel_clause</td><td><pre>CANCEL [IMMEDIATE] [WAIT | NOWAIT]</pre></td></tr>
<tr><td>cell_assignment</td><td><pre>measure_column [{ { condition
                   | expr
                   | single_column_for_loop
                   }
                     [, { condition
                        | expr
                        | single_column_for_loop
                        }
                     ]...
                 | multi_column_for_loop
                 }
               ]

Note: The outer square brackets are part of the
syntax.
      In this case, they do not indicate
optionality.</pre></td></tr>
<tr><td>cell_reference_options</td><td><pre>[{ IGNORE | KEEP } NAV]
[UNIQUE { DIMENSION | SINGLE REFERENCE }]</pre></td></tr>
<tr><td>character_set_clause</td><td><pre>CHARACTER SET character_set</pre></td></tr>
<tr><td>check_datafiles_clause</td><td><pre>CHECK DATAFILES [GLOBAL | LOCAL]</pre></td></tr>
<tr><td>check_diskgroup_clauses</td><td><pre>CHECK
{ ALL
| DISK
    disk_name
    [, disk_name]...
| DISKS IN FAILGROUP
    failgroup_name</pre></td></tr>
</table>

Subclause	Syntax

	[, failgroup_name]...
	| FILE
	    filename
	    [, filename]...
	}
	[CHECK
	  { ALL
	  | DISK
	      disk_name
	      [, disk_name]...
	  | DISKS IN FAILGROUP
	      failgroup_name
	      [, failgroup_name]...
	  | FILE
	      filename
	      [, filename]...
	  }
	]...
	[REPAIR | NOREPAIR]

checkpoint_clause	CHECKPOINT [GLOBAL | LOCAL]
cluster_index_clause	CLUSTER [schema.] cluster index_attributes
coalesce_index_partition	COALESCE PARTITION [parallel_clause]

coalesce_table_partition	COALESCE PARTITION
	   [update_index_clauses]
	   [parallel_clause]

column_association	COLUMNS [schema.]table.column
	          [, [schema.]table.column]...
	   using_statistics_type

column_clauses	{ { add_column_clause
	  | modify_column_clause
	  | drop_column_clause
	  }
	    [add_column_clause
	    | modify_column_clause
	    | drop_column_clause
	    ]...
	| rename_column_clause
	| modify_collection_retrieval
	  [modify_collection_retrieval]...
	| modify_LOB_storage_clause
	| alter_varray_col_properties
	}

Subclause	Syntax

column_properties	{ object_type_col_properties
	| nested_table_col_properties
	| { varray_col_properties | LOB_storage_clause }
	  [(LOB_partition_storage
	      [, LOB_partition_storage]...
	    )
	  ]
	| XMLType_column_properties
	}
	  [{ object_type_col_properties
	    | nested_table_col_properties
	    | { varray_col_properties |
	LOB_storage_clause }
	      [(LOB_partition_storage
	          [, LOB_partition_storage]...
	        )
	      ]
	    | XMLType_column_properties
	    }
	  ]...

commit_switchover_clause	{ PREPARE | COMMIT } TO SWITCHOVER
	[TO { { PHYSICAL | LOGICAL } PRIMARY
	     | PHYSICAL STANDBY
	       [{ WITH | WITHOUT } SESSION SHUTDOWN
	         { WAIT | NOWAIT }
	       ]
	     | LOGICAL STANDBY
	     }
	| CANCEL
	]

compile_type_clause	COMPILE
	   [DEBUG]
	   [SPECIFICATION | BODY]
	   [compiler_parameters_clause
	     [compiler_parameters_clause] ...]
	   [REUSE SETTINGS]

compiler_parameters_clause	parameter_name = parameter_value

composite_partitioning	PARTITION BY RANGE (column_list)
	  [subpartition_by_list |
	subpartition_by_hash]
	  (PARTITION [partition]
	       range_values_clause
	       table_partition_description
	    [, PARTITION [partition]
	          range_values_clause
	          table_partition_description] ...
	  )

Subclause	Syntax
compute_statistics_clause	COMPUTE [SYSTEM] STATISTICS [for_clause]
conditional_insert_clause	[ALL | FIRST] WHEN condition THEN insert_into_clause [values_clause] [error_logging_clause] [insert_into_clause [values_clause] [error_logging_clause]]... [WHEN condition THEN insert_into_clause [values_clause] [error_logging_clause] [insert_into_clause [values_clause] [error_logging_clause]]...]... [ELSE insert_into_clause [values_clause] [error_logging_clause] [insert_into_clause [values_clause] [error_logging_clause]]...]
constraint	{ inline_constraint | out_of_line_constraint | inline_ref_constraint | out_of_line_ref_constraint }
constraint_clauses	{ ADD { out_of_line_constraint [out_of_line_constraint]... | out_of_line_REF_constraint } | MODIFY { CONSTRAINT constraint | PRIMARY KEY | UNIQUE (column [, column]...) } constraint_state | RENAME CONSTRAINT old_name TO new_name | drop_constraint_clause }
constraint_state	[[[NOT] DEFERRABLE] [INITIALLY { IMMEDIATE | DEFERRED }] | [INITIALLY { IMMEDIATE | DEFERRED }]

Subclause	Syntax
	[[NOT] DEFERRABLE]] [RELY | NORELY] [using_index_clause] [ENABLE | DISABLE] [VALIDATE | NOVALIDATE] [exceptions_clause]
constructor_declaration	[FINAL] [INSTANTIABLE] CONSTRUCTOR FUNCTION datatype [[SELF IN OUT datatype,] parameter datatype [, parameter datatype]...] RETURN SELF AS RESULT { IS | AS } { pl/sql_block | call_spec }
constructor_spec	[FINAL] [INSTANTIABLE] CONSTRUCTOR FUNCTION datatype [([SELF IN OUT datatype,] parameter datatype [, parameter datatype]...)] RETURN SELF AS RESULT [{ IS | AS } call_spec]
context_clause	[WITH INDEX CONTEXT, SCAN CONTEXT implementation_type [COMPUTE ANCILLARY DATA]] [WITH COLUMN CONTEXT]
controlfile_clauses	{ CREATE [LOGICAL | PHYSICAL] STANDBY CONTROLFILE AS 'filename' [REUSE] | BACKUP CONTROLFILE TO { 'filename' [REUSE] | trace_file_clause } }
create_datafile_clause	CREATE DATAFILE { 'filename' | filenumber } [, 'filename' | filenumber]... } [AS { file_specification [, file_specification]... | NEW }]

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>create_incomplete_type</td><td><pre>CREATE [OR REPLACE]
   TYPE [schema.]type_name ;</pre></td></tr>
<tr><td>create_mv_refresh</td><td><pre>{ REFRESH
  { { FAST | COMPLETE | FORCE }
  | ON { DEMAND | COMMIT }
  | { START WITH | NEXT } date
  | WITH { PRIMARY KEY | ROWID }
  | USING
       { DEFAULT [MASTER | LOCAL]
            ROLLBACK SEGMENT
       | [MASTER | LOCAL]
            ROLLBACK SEGMENT rollback_segment
       }
         [DEFAULT [MASTER | LOCAL]
              ROLLBACK SEGMENT
         | [MASTER | LOCAL]
              ROLLBACK SEGMENT rollback_segment
         ]...
  | USING
       { ENFORCED | TRUSTED }
       CONSTRAINTS
  }
    [{ FAST | COMPLETE | FORCE }
    | ON { DEMAND | COMMIT }
    | { START WITH | NEXT } date
    | WITH { PRIMARY KEY | ROWID }
    | USING
         { DEFAULT [MASTER | LOCAL]
              ROLLBACK SEGMENT
         | [MASTER | LOCAL]
              ROLLBACK SEGMENT rollback_segment
         }
           [DEFAULT [MASTER | LOCAL]
                ROLLBACK SEGMENT
           | [MASTER | LOCAL]
                ROLLBACK SEGMENT
rollback_segment
           ]...
    | USING
         { ENFORCED | TRUSTED }
         CONSTRAINTS
    ]...
| NEVER REFRESH
}</pre></td></tr>
<tr><td>create_nested_table_type</td><td><pre>CREATE [OR REPLACE]
   TYPE [schema.]type_name
   [OID 'object_identifier']
   { IS | AS } TABLE OF datatype ;</pre></td></tr>
</table>

Subclause	Syntax
create_object_type	CREATE [OR REPLACE] TYPE [schema.]type_name [OID 'object_identifier'] [invoker_rights_clause] { { IS | AS } OBJECT | UNDER [schema.]supertype } [sqlj_object_type] [(attribute datatype [sqlj_object_type_attr] [, attribute datatype [sqlj_object_type_attr]... [, element_spec [, element_spec]...])] [[NOT] FINAL] [[NOT] INSTANTIABLE] ;
create_varray_type	CREATE [OR REPLACE] TYPE [schema.]type_name [OID 'object_identifier'] { IS | AS } { VARRAY | VARYING ARRAY } (limit) OF datatype ;
database_file_clauses	{ RENAME FILE 'filename' [, 'filename']... TO 'filename' | create_datafile_clause | alter_datafile_clause | alter_tempfile_clause }
database_logging_clauses	{ LOGFILE [GROUP integer] file_specification [, [GROUP integer] file_specification]... | MAXLOGFILES integer | MAXLOGMEMBERS integer | MAXLOGHISTORY integer | { ARCHIVELOG | NOARCHIVELOG } | FORCE LOGGING }
datafile_tempfile_clauses	{ ADD { DATAFILE | TEMPFILE } [file_specification [, file_specification]...] | RENAME DATAFILE 'filename' [, 'filename']... TO

Subclause	Syntax
	'filename' [, 'filename']... | { DATAFILE | TEMPFILE } { ONLINE | OFFLINE } }
datafile_tempfile_spec	['filename'] [SIZE size_clause] [REUSE] [autoextend_clause]
dblink	database[.domain [.domain]...] [@ connect_descriptor]
dblink_authentication	AUTHENTICATED BY user IDENTIFIED BY password
deallocate_unused_clause	DEALLOCATE UNUSED [KEEP size_clause]
default_cost_clause	DEFAULT COST (cpu_cost, io_cost, network_cost)
default_selectivity_clause	DEFAULT SELECTIVITY default_selectivity
default_tablespace	DEFAULT TABLESPACE tablespace [DATAFILE datafile_tempfile_spec] extent_management_clause
default_settings_clauses	{ SET DEFAULT { BIGFILE | SMALLFILE } TABLESPACE | DEFAULT TABLESPACE tablespace | DEFAULT TEMPORARY TABLESPACE { tablespace | tablespace_group_name } | RENAME GLOBAL_NAME TO database.domain [.domain]... | { ENABLE BLOCK CHANGE TRACKING [USING FILE 'filename' [REUSE]] | DISABLE BLOCK CHANGE TRACKING } | flashback_mode_clause | set_time_zone_clause }
default_temp_tablespace	[BIGFILE | SMALLFILE] DEFAULT TEMPORARY TABLESPACE tablespace [TEMPFILE file_specification [, file_specification]...] extent_management_clause

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>dependent_handling_clause</td><td><pre>
{ INVALIDATE
| CASCADE [{ [NOT] INCLUDING TABLE DATA
            | CONVERT TO SUBSTITUTABLE
            }
          ]
     [[FORCE] exceptions_clause]
}
</pre></td></tr>
<tr><td>dimension_join_clause</td><td><pre>
JOIN KEY
   { child_key_column
   | (child_key_column [, child_key_column]...)
   }
REFERENCES parent_level
  [JOIN KEY
       { child_key_column
       | (child_key_column [,
child_key_column]...)
       }
     REFERENCES parent_level
  ]...
</pre></td></tr>
<tr><td>disk_clauses</td><td><pre>
{ diskgroup_name
    { add_disk_clause
    | drop_disk_clauses
    | resize_disk_clauses
    }
| { diskgroup_name | ALL }
      undrop_disk_clause
}
</pre></td></tr>
<tr><td>diskgroup_alias_clauses</td><td><pre>
{ ADD ALIAS
    alias_name FOR filename
    [, alias_name FOR filename]...
| DROP ALIAS
    alias_name
    [, alias_name]...
| RENAME ALIAS
    old_alias_name TO new_alias_name
    [, old_alias_name TO new_alias_name]...
}
</pre></td></tr>
<tr><td>diskgroup_availability</td><td><pre>
{ MOUNT
| DISMOUNT [FORCE | NOFORCE]
}
</pre></td></tr>
<tr><td>diskgroup_clauses</td><td><pre>
{ diskgroup_name
    { rebalance_diskgroup_clause
    | check_diskgroup_clauses
    | diskgroup_template_clauses
    | diskgroup_directory_clauses
    | diskgroup_alias_clauses
</pre></td></tr>
</table>

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td> | drop_diskgroup_file_clause
    }
| { diskgroup_name | ALL }
  diskgroup_availability
}</td></tr>
<tr><td>diskgroup_directory_clauses</td><td>{ ADD DIRECTORY
    filename
    [, filename]...
| DROP DIRECTORY
    filename [FORCE | NOFORCE]
    [, filename [FORCE | NOFORCE]]...
| RENAME DIRECTORY
    old_dir_name TO new_dir_name
    [, old_dir_name TO new_dir_name]...
}</td></tr>
<tr><td>diskgroup_file_spec</td><td>[' { fully_qualified_file_name
    | numeric_file_name
    | incorporate_file_name
    | alias_file_name
    }
  '
]
[SIZE size_clause]
[REUSE]
[autoextend_clause]</td></tr>
<tr><td>diskgroup_template_clauses</td><td>{ { ADD | ALTER } TEMPLATE
      qualified_template_clause
      [, qualified_template_clause]...
| DROP TEMPLATE
       template_name
       [, template_name]...
}</td></tr>
<tr><td>distributed_recov_clauses</td><td>{ ENABLE | DISABLE } DISTRIBUTED RECOVERY</td></tr>
<tr><td>dml_event_clause</td><td>{ DELETE | INSERT | UPDATE
     [OF column [, column]...]
}
  [OR { DELETE | INSERT | UPDATE
       [OF column [, column]...]
       }
  ]...
ON { [schema.]table
   | [NESTED TABLE nested_table_column OF]
         [schema.] view
   }
[referencing_clause]
[FOR EACH ROW]</td></tr>
</table>

Subclause	Syntax

dml_table_expression_clause	{ [schema.]
	  { table
	    [{ PARTITION (partition)
	      | SUBPARTITION (subpartition)
	      }
	    | @ dblink
	    ]
	  | { view | materialized view } [@ dblink]
	  }
	| (subquery [subquery_restriction_clause])
	| table_collection_expression
	}

domain_index_clause	INDEXTYPE IS indextype
	   [parallel_clause]
	   [PARAMETERS ('ODCI_parameters')]

drop_binding_clause	DROP BINDING (parameter_type [, parameter_type]...) [FORCE]

drop_column_clause	{ SET UNUSED { COLUMN column
	             | (column [, column]...)
	             }
	  [{ CASCADE CONSTRAINTS | INVALIDATE }
	      [CASCADE CONSTRAINTS | INVALIDATE]...
	  ]
	| DROP { COLUMN column
	       | (column [, column]...)
	       }
	  [{ CASCADE CONSTRAINTS | INVALIDATE }
	      [CASCADE CONSTRAINTS | INVALIDATE]...
	  ]
	  [CHECKPOINT integer]
	| DROP { UNUSED COLUMNS
	       | COLUMNS CONTINUE
	       }
	  [CHECKPOINT integer]
	}

drop_constraint_clause	DROP
	   { { PRIMARY KEY
	     | UNIQUE (column [, column]...)
	     }
	     [CASCADE]
	     [{ KEEP | DROP } INDEX]
	   | CONSTRAINT constraint
	     [CASCADE]
	   }

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>drop_disk_clauses</td><td><pre>DROP
{ DISK
    disk_name [FORCE | NOFORCE]
    [, disk_name [FORCE | NOFORCE]]...
| DISKS IN FAILGROUP
    failgroup_name [FORCE | NOFORCE]
    [, failgroup_name [FORCE | NOFORCE]]...
}</pre></td></tr>
<tr><td>drop_diskgroup_file_clause</td><td><pre>DROP FILE
  filename
  [, filename]...</pre></td></tr>
<tr><td>drop_index_partition</td><td><pre>DROP PARTITION partition_name</pre></td></tr>
<tr><td>drop_logfile_clauses</td><td><pre>DROP [STANDBY] LOGFILE
   { logfile_descriptor
     [, logfile_descriptor]...
   | MEMBER 'filename'
            [, 'filename']...
   }</pre></td></tr>
<tr><td>drop_table_partition</td><td><pre>DROP PARTITION partition
   [update_index_clauses [parallel_clause]]</pre></td></tr>
<tr><td>drop_table_subpartition</td><td><pre>DROP SUBPARTITION subpartition
   [update_index_clauses [parallel_clause]]</pre></td></tr>
<tr><td>element_spec</td><td><pre>[inheritance_clauses]
{ subprogram_spec
| constructor_spec
| map_order_function_spec
}
  [subprogram_clause
  | constructor_spec
  | map_order_function_spec
  ]...
[, pragma_clause]</pre></td></tr>
<tr><td>else_clause</td><td><pre>ELSE else_expr</pre></td></tr>
<tr><td>enable_disable_clause</td><td><pre>{ ENABLE | DISABLE }
[VALIDATE | NOVALIDATE]
{ UNIQUE (column [, column]...)
| PRIMARY KEY
| CONSTRAINT constraint
}
[using_index_clause]</pre></td></tr>
</table>

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td><pre>[exceptions_clause]
[CASCADE]
[{ KEEP | DROP } INDEX]</pre></td></tr>
<tr><td>end_session_clauses</td><td><pre>{ DISCONNECT SESSION 'integer1, integer2'
     [POST_TRANSACTION]
| KILL SESSION 'integer1, integer2'
}
[IMMEDIATE]</pre></td></tr>
<tr><td>estimate_statistics_clause</td><td><pre>ESTIMATE [SYSTEM] STATISTICS [for_clause]
[SAMPLE integer { ROWS | PERCENT }]</pre></td></tr>
<tr><td>exceptions_clause</td><td><pre>EXCEPTIONS INTO [schema.]table</pre></td></tr>
<tr><td>exchange_partition_subpart</td><td><pre>EXCHANGE { PARTITION partition
         | SUBPARTITION subpartition
         }
   WITH TABLE table
   [{ INCLUDING | EXCLUDING } INDEXES]
   [{ WITH | WITHOUT } VALIDATION]
   [exceptions_clause]
   [update_index_clauses [parallel_clause]]</pre></td></tr>
<tr><td>expr</td><td><pre>{ simple_expression
| compound_expression
| case_expression
| cursor_expression
| datetime_expression
| function_expression
| interval_expression
| object_access_expression
| scalar_subquery_expression
| model_expression
| type_constructor_expression
| variable_expression
}</pre></td></tr>
<tr><td>expression_list</td><td><pre>{ expr [, expr]...
| (expr [, expr]...)
}</pre></td></tr>
<tr><td>extended_attribute_clause</td><td><pre>ATTRIBUTE attribute
LEVEL level
DETERMINES { dependent_column
           | (dependent_column
               [, dependent_column]...
)
[LEVEL level</pre></td></tr>
</table>

Subclause	Syntax
	DETERMINES { dependent_column | (dependent_column [, dependent_column]...)]...
extent_management_clause	EXTENT MANAGEMENT { DICTIONARY | LOCAL [AUTOALLOCATE | UNIFORM [SIZE size_clause]] }
external_data_properties	DEFAULT DIRECTORY directory [ACCESS PARAMETERS { (opaque_format_spec) | USING CLOB subquery }] LOCATION ([directory:] 'location_specifier' [, [directory:] 'location_specifier']...)
external_table_clause	([TYPE access_driver_type] external_data_properties) [REJECT LIMIT { integer | UNLIMITED }]
file_specification	{ datafile_tempfile_spec | diskgroup_file_spec | redo_log_file_spec }
finish_clause	[DISCONNECT [FROM SESSION]] [parallel_clause] FINISH [SKIP [STANDBY LOGFILE]] [WAIT | NOWAIT]
flashback_mode_clause	FLASHBACK { ON | OFF }
flashback_query_clause	[VERSIONS BETWEEN { SCN | TIMESTAMP } { expr | MINVALUE } AND { expr | MAXVALUE }

Subclause	Syntax
	] AS OF { SCN | TIMESTAMP } expr
for_clause	FOR { TABLE | ALL [INDEXED] COLUMNS [SIZE integer] | COLUMNS [SIZE integer] { column | attribute } [SIZE integer] [{ column | attribute } [SIZE integer]]... | ALL [LOCAL] INDEXES } [FOR { TABLE | ALL [INDEXED] COLUMNS [SIZE integer] | COLUMNS [SIZE integer] { column | attribute } [SIZE integer] [{ column | attribute } [SIZE integer]]... | ALL [LOCAL] INDEXES }]...
for_update_clause	FOR UPDATE [OF [[schema.] { table | view } .]column [, [[schema.] { table | view } .]column]...] [NOWAIT | WAIT integer]
full_database_recovery	[STANDBY] DATABASE [{ UNTIL { CANCEL | TIME date | CHANGE integer } | USING BACKUP CONTROLFILE } [UNTIL { CANCEL | TIME date | CHANGE integer } | USING BACKUP CONTROLFILE]...]

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>fully_qualified_file_name</td><td>+diskgroup_name/db_name/file_type/
   file_type_tag.filenumber.incarnation_number</td></tr>
<tr><td>function_association</td><td>{ FUNCTIONS
     [schema.]function [,
[schema.]function]...
| PACKAGES
     [schema.]package [,
[schema.]package]...
| TYPES
     [schema.]type [, [schema.]type]...
| INDEXES
     [schema.]index [, [schema.]index]...
| INDEXTYPES
     [schema.]indextype [,
[schema.]indextype]...
}
{ using_statistics_type
| { default_cost_clause
    [, default_selectivity_clause]
  | default_selectivity_clause
    [, default_cost_clause]
  }
}</td></tr>
<tr><td>function_declaration</td><td>FUNCTION name
   (parameter datatype[, parameter
datatype]...)
   RETURN datatype
   { IS | AS } { pl/sql_block | call_spec }</td></tr>
<tr><td>function_spec</td><td>FUNCTION name
   (parameter datatype [, parameter
datatype]...)
   return_clause</td></tr>
<tr><td>general_recovery</td><td>RECOVER
[AUTOMATIC]
[FROM 'location']
{ { full_database_recovery
  | partial_database_recovery
  | LOGFILE 'filename'
  }
  [{ TEST
    | ALLOW integer CORRUPTION
    | parallel_clause
    }
      [TEST
      | ALLOW integer CORRUPTION
      | parallel_clause
      ]...</td></tr>
</table>

Subclause	Syntax
	] | CONTINUE [DEFAULT] | CANCEL }

global_partitioned_index	GLOBAL PARTITION BY
	   { RANGE
	        (column_list)
	        (index_partitioning_clause)
	   | HASH
	        (column_list)
	        { individual_hash_partitions
	        | hash_partitions_by_quantity
	        }
	   }

grant_object_privileges	{ object_privilege | ALL [PRIVILEGES] }
	[(column [, column]...)]
	  [, { object_privilege | ALL [PRIVILEGES] }
	     [(column [, column]...)]
	  ]...
	on_object_clause
	TO grantee_clause
	[WITH HIERARCHY OPTION]
	[WITH GRANT OPTION]

grant_system_privileges	{ system_privilege
	| role
	| ALL PRIVILEGES
	}
	  [, { system_privilege
	     | role
	     | ALL PRIVILEGES
	     }
	  ]...
	TO grantee_clause
	[IDENTIFIED BY password]
	[WITH ADMIN OPTION]

grantee_clause	{ user | role | PUBLIC } [, { user | role | PUBLIC }]...

group_by_clause	GROUP BY
	   { expr
	   | rollup_cube_clause
	   | grouping_sets_clause
	   }
	     [, { expr
	        | rollup_cube_clause
	        | grouping_sets_clause
	        }

Subclause	Syntax
	]... [HAVING condition]
grouping_expression_list	expression_list [, expression_list]...
grouping_sets_clause	GROUPING SETS ({ rollup_cube_clause | grouping_expression_list })
hash_partitioning	PARTITION BY HASH (column [, column] ...) { individual_hash_partitions | hash_partitions_by_quantity }
hash_partitions_by_quantity	PARTITIONS hash_partition_quantity [STORE IN (tablespace [, tablespace]...)] [OVERFLOW STORE IN (tablespace [, tablespace]...)]
hierarchical_query_clause	[START WITH condition] CONNECT BY [NOCYCLE] condition
hierarchy_clause	HIERARCHY hierarchy (child_level CHILD OF parent_level [CHILD OF parent_level]... [dimension_join_clause])
implementation_clause	{ ANCILLARY TO primary_operator (parameter_type [, parameter_type]...) [, primary_operator (parameter_type [, parameter_type]...)]... | context_clause }
incomplete_file_name	+diskgroup_name [(template_name)]
index_attributes	[{ physical_attributes_clause | logging_clause | ONLINE | COMPUTE STATISTICS | TABLESPACE { tablespace | DEFAULT } | key_compression

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td> | { SORT | NOSORT }
  | REVERSE
  | parallel_clause
  }
    [physical_attributes_clause
    | logging_clause
    | ONLINE
    | COMPUTE STATISTICS
    | TABLESPACE { tablespace | DEFAULT }
    | key_compression
    | { SORT | NOSORT }
    | REVERSE
    | parallel_clause
    ]...
]</td></tr>
<tr><td>index_expr</td><td>{ column | column_expression }</td></tr>
<tr><td>index_org_overflow_clause</td><td>[INCLUDING column_name]
OVERFLOW
[segment_attributes_clause]</td></tr>
<tr><td>index_org_table_clause</td><td>[{ mapping_table_clause
  | PCTTHRESHOLD integer
  | key_compression
  }
    [mapping_table_clause
    | PCTTHRESHOLD integer
    | key_compression
    ]...
]
[index_org_overflow_clause]</td></tr>
<tr><td>index_partition_description</td><td>PARTITION
[partition
   [{ segment_attributes_clause
     | key_compression
     }
     [segment_attributes_clause
     | key_compression
     ]...
   ]
]</td></tr>
<tr><td>index_partitioning_clause</td><td>PARTITION [partition]
   VALUES LESS THAN (value[, value...])
   [segment_attributes_clause]</td></tr>
<tr><td>index_properties</td><td>[{ { global_partitioned_index
    | local_partitioned_index</td></tr>
</table>

Subclause	Syntax
	} | index_attributes } [{ { global_partitioned_index | local_partitioned_index } | index_attributes }]... | domain_index_clause]
index_subpartition_clause	{ STORE IN (tablespace[, tablespace]...) | (SUBPARTITION [subpartition [TABLESPACE tablespace]] [, SUBPARTITION [subpartition [TABLESPACE tablespace]]]...) }
individual_hash_partitions	(PARTITION [partition partitioning_storage_clause] [, PARTITION [partition partitioning_storage_clause]]...)
inheritance_clauses	[NOT] { OVERRIDING | FINAL | INSTANTIABLE } [[NOT] { OVERRIDING | FINAL | INSTANTIABLE }]...
inline_constraint	[CONSTRAINT constraint_name] { [NOT] NULL | UNIQUE | PRIMARY KEY | references_clause | CHECK (condition) } [constraint_state]
inline_ref_constraint	{ SCOPE IS [schema.] scope_table | WITH ROWID | [CONSTRAINT constraint_name] references_clause [constraint_state] }

Subclause	Syntax

inner_cross_join_clause	table_reference
	{ [INNER] JOIN table_reference
	    { ON condition
	    | USING (column [, column]...)
	    }
	| { CROSS
	  | NATURAL [INNER]
	  }
	  JOIN table_reference
	}

insert_into_clause	INTO dml_table_expression_clause [t_alias] [(column [, column]...)]
integer	[+ | -] digit [digit]...

interval_day_to_second	INTERVAL
	  '{ integer | integer time_expr | time_expr }'
	{ { DAY | HOUR | MINUTE }
	      [(leading_precision)]
	| SECOND
	    [(leading_precision
	        [, fractional_seconds_precision]
	      )
	    ]
	}
	[TO { DAY | HOUR | MINUTE | SECOND
	       [(fractional_seconds_precision)]
	     }
	]

interval_year_to_month	INTERVAL 'integer [- integer]' { YEAR | MONTH } [(precision)] [TO { YEAR | MONTH }]
into_clause	INTO [schema.] table
invoker_rights_clause	AUTHID { CURRENT_USER | DEFINER }
Java_declaration	JAVA NAME 'string'
join_clause	{ inner_cross_join_clause | outer_join_clause }
key_compression	{ COMPRESS [integer] | NOCOMPRESS }

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td><pre>level_clause</pre></td><td><pre>LEVEL level IS
   { level_table.level_column
   | (level_table.level_column
      [, level_table.level_column]...
)
   }</pre></td></tr>
<tr><td><pre>list_partitioning</pre></td><td><pre>PARTITION BY LIST (column)
(PARTITION [partition]
    list_values_clause
    table_partition_description
  [, PARTITION [partition]
        list_values_clause
        table_partition_description
  ]...
)</pre></td></tr>
<tr><td><pre>list_values_clause</pre></td><td><pre>VALUES ({ value | NULL
          [, { value | NULL }...)
       | DEFAULT
)</pre></td></tr>
<tr><td><pre>LOB_parameters</pre></td><td><pre>{ TABLESPACE tablespace
| { ENABLE | DISABLE } STORAGE IN ROW
| storage_clause
| CHUNK integer
| PCTVERSION integer
| RETENTION
| FREEPOOLS integer
| { CACHE
  | { NOCACHE | CACHE READS } [logging_clause]
  }
}
  [TABLESPACE tablespace
  | { ENABLE | DISABLE } STORAGE IN ROW
  | storage_clause
  | CHUNK integer
  | PCTVERSION integer
  | RETENTION
  | FREEPOOLS integer
  | { CACHE
    | { NOCACHE | CACHE READS }
[logging_clause]
    }
  ]...</pre></td></tr>
<tr><td><pre>LOB_partition_storage</pre></td><td><pre>PARTITION partition
{ LOB_storage_clause | varray_col_properties }
  [LOB_storage_clause |
varray_col_properties]...
[(SUBPARTITION subpartition</pre></td></tr>
</table>

Subclause	Syntax
	{ LOB_storage_clause | varray_col_properties } [LOB_storage_clause | varray_col_properties]...)]
LOB_storage_clause	LOB { (LOB_item [, LOB_item]...) STORE AS (LOB_parameters) | (LOB_item) STORE AS { LOB_segname (LOB_parameters) | LOB_segname | (LOB_parameters) } }
local_partitioned_index	LOCAL [on_range_partitioned_table | on_list_partitioned_table | on_hash_partitioned_table | on_comp_partitioned_table]
logfile_clause	LOGFILE [GROUP integer] file_specification [, [GROUP integer] file_specification]...
logfile_clauses	{ { ARCHIVELOG [MANUAL] | NOARCHIVELOG } | [NO] FORCE LOGGING | RENAME FILE 'filename' [, 'filename']... TO 'filename' | CLEAR [UNARCHIVED] LOGFILE logfile_descriptor [, logfile_descriptor]... [UNRECOVERABLE DATAFILE] | add_logfile_clauses | drop_logfile_clauses | supplemental_db_logging }
logfile_descriptor	{ GROUP integer | ('filename' [, 'filename']...) | 'filename' }

Subclause	Syntax
logging_clause	{ LOGGING | NOLOGGING }
main_model	[MAIN main_model_name] model_column_clauses [cell_reference_options] model_rules_clause
managed_standby_recovery	RECOVER MANAGED STANDBY DATABASE [recover_clause | cancel_clause | finish_clause]
map_order_func_declaration	{ MAP | ORDER } MEMBER function_declaration
map_order_function_spec	{ MAP | ORDER } MEMBER function_spec
mapping_table_clauses	{ MAPPING TABLE | NOMAPPING }
materialized_view_props	[column_properties] [table_partitioning_clauses] [CACHE | NOCACHE] [parallel_clause] [build_clause]
maximize_standby_db_clause	SET STANDBY DATABASE TO MAXIMIZE { PROTECTION | AVAILABILITY | PERFORMANCE }
maxsize_clause	MAXSIZE { UNLIMITED | size_clause }
merge_insert_clause	WHEN NOT MATCHED THEN INSERT [(column [, column]...)] VALUES ({ expr [, expr]... | DEFAULT }) [where_clause]

merge_table_partitions	MERGE PARTITIONS partition_1, partition_2
	   [INTO partition_spec]
	   [update_index_clauses]
	   [parallel_clause]

merge_table_subpartitions	MERGE SUBPARTITIONS subpart_1, subpart_2
	   [INTO subpartition_spec]
	   [update_index_clauses]
	   [parallel_clause]

merge_update_clause	WHEN MATCHED THEN UPDATE SET column = { expr | DEFAULT }

Subclause	Syntax
	[, column = { expr | DEFAULT }]... [where_clause] [DELETE where_clause]

model_clause	MODEL
	   [cell_reference_options]
	   [return_rows_clause]
	   [reference_model]
	     [reference_model]...
	   main_model

model_column	expr [[AS] c_alias]

model_column_clauses	[query_partition_clause [c_alias]]
	DIMENSION BY (model_column
	               [, model_column]...)
	MEASURES (model_column
	           [, model_column]...)

model_rules_clause	RULES
	[UPSERT | UPDATE]
	[{ AUTOMATIC | SEQUENTIAL } ORDER]
	[ITERATE (number) [UNTIL (condition)]]
	([UPDATE | UPSERT]
	 cell_assignment [order_by_clause] = expr
	   [[UPDATE | UPSERT]
	     cell_assignment [order_by_clause] = expr
	   ]...
	)

modify_col_properties	(column [datatype]
	         [DEFAULT expr]
	         [inline_constraint
	           [inline_constraint]...]
	         [LOB_storage_clause]
	  [, column [datatype]
	            [DEFAULT expr]
	            [inline_constraint
	              [inline_constraint]...]
	            [LOB_storage_clause]
	  ]
	)

modify_col_substitutable	COLUMN column [NOT] SUBSTITUTABLE AT ALL LEVELS [FORCE]
modify_collection_retrieval	MODIFY NESTED TABLE collection_item RETURN AS { LOCATOR | VALUE }

Subclause	Syntax
modify_column_clauses	MODIFY { modify_col_properties | modify_col_substitutable }
modify_hash_partition	MODIFY PARTITION partition { partition_attributes | alter_mapping_table_clause | [REBUILD] UNUSABLE LOCAL INDEXES }
modify_hash_subpartition	{ { allocate_extent_clause | deallocate_unused_clause | shrink_clause | { LOB LOB_item | VARRAY varray } modify_LOB_parameters [{ LOB LOB_item | VARRAY varray } modify_LOB_parameters]... } | [REBUILD] UNUSABLE LOCAL INDEXES }
modify_index_default_attrs	MODIFY DEFAULT ATTRIBUTES [FOR PARTITION partition] { physical_attributes_clause | TABLESPACE { tablespace | DEFAULT } | logging_clause } [physical_attributes_clause | TABLESPACE { tablespace | DEFAULT } | logging_clause]...
modify_index_partition	MODIFY PARTITION partition { { deallocate_unused_clause | allocate_extent_clause | physical_attributes_clause | logging_clause | key_compression } [deallocate_unused_clause | allocate_extent_clause | physical_attributes_clause | logging_clause | key_compression]... | PARAMETERS ('ODCI_parameters')

Subclause	Syntax
	| COALESCE | UPDATE BLOCK REFERENCES | UNUSABLE }
modify_index_subpartition	MODIFY SUBPARTITION subpartition { UNUSABLE | allocate_extent_clause | deallocate_unused_clause }

modify_list_partition	MODIFY PARTITION partition
	  { partition_attributes
	  | {ADD | DROP} VALUES
	    (partition_value[, partition_value]...)
	  | [REBUILD] UNUSABLE LOCAL INDEXES
	  }

modify_list_subpartition	{ allocate_extent_clause
	| deallocate_unused_clause
	| shrink_clause
	| { LOB LOB_item | VARRAY varray }
	  modify_LOB_parameters
	    [{ LOB LOB_item | VARRAY varray }
	      modify_LOB_parameters
	    ] ...
	| [REBUILD] UNUSABLE LOCAL INDEXES
	| { ADD | DROP } VALUES (value[, value]...)
	}

modify_LOB_parameters	{ storage_clause
	| PCTVERSION integer
	| RETENTION
	| FREEPOOLS integer
	| REBUILD FREEPOOLS
	| { CACHE
	  | { NOCACHE | CACHE READS } [logging_clause]
	  }
	| allocate_extent_clause
	| deallocate_unused_clause
	| shrink_clause
	
	}
	  [storage_clause
	  | PCTVERSION integer
	  | RETENTION
	  | FREEPOOLS integer
	  | REBUILD FREEPOOLS
	  | { CACHE
	    | { NOCACHE | CACHE READS }
	[logging_clause]

Subclause	Syntax
	} | allocate_extent_clause | deallocate_unused_clause | shrink_clause]...
modify_LOB_storage_clause	MODIFY LOB (LOB_item) (modify_LOB_parameters)
modify_range_partition	MODIFY PARTITION partition { partition_attributes | { add_hash_subpartition | add_list_subpartition } | COALESCE SUBPARTITION [update_index_clauses] [parallel_clause] | alter_mapping_table_clause | [REBUILD] UNUSABLE LOCAL INDEXES }
modify_table_default_attrs	MODIFY DEFAULT ATTRIBUTES [FOR PARTITION partition] [segment_attributes_clause] [table_compression] [PCTTHRESHOLD integer] [key_compression] [alter_overflow_clause] [{ LOB (LOB_item) | VARRAY varray } (LOB_parameters) [{ LOB (LOB_item) | VARRAY varray } (LOB_parameters)]...]
modify_table_partition	{ modify_range_partition | modify_hash_partition | modify_list_partition }
modify_table_subpartition	MODIFY SUBPARTITION subpartition { modify_hash_subpartition | modify_list_subpartition }

Subclause	Syntax
move_table_clause	MOVE [ONLINE] [segment_attributes_clause] [table_compression] [index_org_table_clause] [{ LOB_storage_clause | varray_col_properties } [{ LOB_storage_clause | varray_col_properties }]...] [parallel_clause]
move_table_partition	MOVE PARTITION partition [MAPPING TABLE] [table_partition_description] [update_index_clauses] [parallel_clause]
move_table_subpartition	MOVE SUBPARTITION subpartition_spec [update_index_clauses] [parallel_clause]

multi_column_for_loop	FOR (dimension_column
	[, dimension_column]...)
	IN ({ (literal [, literal]...)
	[(literal [, literal]...)...]
	| subquery
	}
	)

multi_table_insert	{ ALL insert_into_clause [values_clause] [insert_into_clause [values_clause]]... | conditional_insert_clause } subquery
multiset_except	nested_table1 MULTISET EXCEPT [ALL | DISTINCT] nested_table2
multiset_intersect	nested_table1 MULTISET INTERSECT [ALL | DISTINCT] nested_table2

Subclause	Syntax
multiset_union	nested_table1 MULTISET UNION [ALL | DISTINCT] nested_table2

nested_table_col_properties	NESTED TABLE
	{ nested_item | COLUMN_VALUE }
	[substitutable_column_clause]
	STORE AS storage_table
	[({ (object_properties)
	    | [physical_properties]
	    | [column_properties]
	    }
	      [(object_properties)
	      | [physical_properties]
	      | [column_properties]
	      ]...
	  )
	]
	[RETURN AS { LOCATOR | VALUE }]

new_values_clause	{ INCLUDING | EXCLUDING } NEW VALUES
number	[+ | -] { digit [digit]... [.] [digit [digit]...] | . digit [digit]... } [e [+ | -] digit [digit]...] [f | d]
numeric_file_name	+diskgroup_name.filenumber.incarnation_number

object_properties	{ { column | attribute }
	  [DEFAULT expr]
	  [inline_constraint [inline_constraint]...
	  | inline_ref_constraint
	  ]
	| { out_of_line_constraint
	  | out_of_line_ref_constraint
	  | supplemental_logging_props
	  }
	}

object_table	CREATE [GLOBAL TEMPORARY] TABLE
	   [schema.]table OF
	   [schema.]object_type
	   [object_table_substitution]
	   [(object_properties)]
	   [ON COMMIT { DELETE | PRESERVE } ROWS]
	   [OID_clause]

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td>[OID_index_clause]
[physical_properties]
[table_properties] ;</td></tr>
<tr><td>object_table_substitution</td><td>[NOT] SUBSTITUTABLE AT ALL LEVELS</td></tr>
<tr><td>object_type_col_properties</td><td>COLUMN column substitutable_column_clause</td></tr>
<tr><td>object_view_clause</td><td>OF [schema.]type_name
{ WITH OBJECT IDENTIFIER
{ DEFAULT | (attribute
[, attribute]...)
}
| UNDER [schema.]superview
}
({ out_of_line_constraint
| attribute inline_constraint
[inline_constraint]...
}
[, { out_of_line_constraint
| attribute inline_constraint
[inline_constraint]...
}
]...
)</td></tr>
<tr><td>OID_clause</td><td>OBJECT IDENTIFIER IS
{ SYSTEM GENERATED | PRIMARY KEY }</td></tr>
<tr><td>OID_index_clause</td><td>OIDINDEX [index]
({ physical_attributes_clause
| TABLESPACE tablespace
}
[physical_attributes_clause
| TABLESPACE tablespace
]...
)</td></tr>
<tr><td>on_comp_partitioned_table</td><td>[STORE IN (tablespace [, tablespace]...)]
(PARTITION
[partition
[{ segment_attribute_clause
| key_compression
}
[segment_attribute_clause
| key_compression
]...
]
[index_subpartition_clause]
]</td></tr>
</table>

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td><pre>
      [, PARTITION
           [partition
             [{ segment_attribute_clause
               | key_compression
               }
                 [segment_attribute_clause
                 | key_compression
                 ]...
             ]
             [index_subpartition_clause]
           ]...
      ]
)
</pre></td></tr>
<tr><td>on_hash_partitioned_table</td><td><pre>
{ STORE IN (tablespace[, tablespace]...)
| (PARTITION
      [partition [TABLESPACE tablespace]]
   [, PARTITION
         [partition [TABLESPACE tablespace]]
   ]...
)
}
</pre></td></tr>
<tr><td>on_list_partitioned_table</td><td><pre>
(PARTITION
    [partition
      [{ segment_attributes_clause
        | key_compression
        }
          [segment_attributes_clause
          | key_compression
          ]...
      ]
    ]
  [, PARTITION
       [partition
         [{ segment_attributes_clause
           | key_compression
           }
             [segment_attributes_clause
             | key_compression
             ]...
         ]
       ]
  ]...
)
</pre></td></tr>
<tr><td>on_object_clause</td><td><pre>
{ schema.object
| { DIRECTORY directory_name
  | JAVA { SOURCE | RESOURCE } [schema.]object
  }
}
</pre></td></tr>
</table>

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>on_range_partitioned_table</td><td><pre>
(PARTITION
    [partition
      [{ segment_attributes_clause
        | key_compression
        }
          [segment_attributes_clause
          | key_compression
          ]...
      ]
    ]
  [, PARTITION
       [partition
         [{ segment_attributes_clause
           | key_compression
           }
             [segment_attributes_clause
             | key_compression
             ]...
         ]
       ]
  ]...
)
</pre></td></tr>
<tr><td>order_by_clause</td><td><pre>
ORDER [SIBLINGS] BY
{ expr | position | c_alias }
[ASC | DESC]
[NULLS FIRST | NULLS LAST]
  [, { expr | position | c_alias }
     [ASC | DESC]
     [NULLS FIRST | NULLS LAST]
  ]...
</pre></td></tr>
<tr><td>out_of_line_constraint</td><td><pre>
[CONSTRAINT constraint_name]
{ UNIQUE (column [, column]...)
| PRIMARY KEY (column [, column]...)
| FOREIGN KEY (column [, column]...)
     references_clause
| CHECK (condition)
}
[constraint_state]
</pre></td></tr>
<tr><td>out_of_line_ref_constraint</td><td><pre>
{ SCOPE FOR
     ({ ref_col | ref_attr })
     IS [schema.]scope_table
| REF
     ({ ref_col | ref_attr })
     WITH ROWID
| [CONSTRAINT constraint_name]
  FOREIGN KEY
     ({ ref_col | ref_attr })
  references_clause
  [constraint_state]
}
</pre></td></tr>
</table>

Subclause	Syntax
outer_join_clause	table_reference [query_partition_clause] { outer_join_type JOIN | NATURAL [outer_join_type] JOIN } table_reference [query_partition_clause] [ON condition | USING (column [, column]...)]
outer_join_type	{ FULL | LEFT | RIGHT } [OUTER]
parallel_clause	{ NOPARALLEL | PARALLEL [integer] }

parallel_enable_clause	PARALLEL_ENABLE
	[(PARTITION argument BY
	      { ANY
	      | { HASH | RANGE } (column [, column]...)
	      }
	  )
	 [streaming_clause]
	]

partial_database_recovery	{ TABLESPACE tablespace [, tablespace]...
	| DATAFILE { 'filename' | filenumber }
	             [, 'filename' | filenumber]...
	           }
	| STANDBY
	  { TABLESPACE tablespace [, tablespace]...
	  | DATAFILE { 'filename' | filenumber }
	               [, 'filename' | filenumber]...
	             }
	  }
	  UNTIL [CONSISTENT WITH] CONTROLFILE
	}

partition_attributes	[{ physical_attributes_clause
	  | logging_clause
	  | allocate_extent_clause
	  | deallocate_unused_clause
	  | shrink_clause
	  }
	    [physical_attributes_clause
	    | logging_clause
	    | allocate_extent_clause
	    | deallocate_unused_clause
	    | shrink_clause
	    ]...
	]
	[OVERFLOW

Subclause	Syntax
	{ physical_attributes_clause | logging_clause | allocate_extent_clause | deallocate_unused_clause } [physical_attributes_clause | logging_clause | allocate_extent_clause | deallocate_unused_clause]...] [table_compression] [{ LOB LOB_item | VARRAY varray } modify_LOB_parameters [{ LOB LOB_item | VARRAY varray } modify_LOB_parameters]...]
partition_extended_name	[schema.] { table | view } [PARTITION (partition) | SUBPARTITION (subpartition)]
partition_level_subpartition	{ SUBPARTITIONS hash_subpartition_quantity [STORE IN (tablespace[, tablespace]...)] | (subpartition_spec[, subpartition_spec]...) }
partition_spec	PARTITION [partition] [table_partition_description]
partitioning_storage_clause	[{ TABLESPACE tablespace | OVERFLOW [TABLESPACE tablespace] | LOB (LOB_item) STORE AS { LOB_segname [(TABLESPACE tablespace)] | (TABLESPACE tablespace) } | VARRAY varray_item STORE AS LOB LOB_segname } [{ TABLESPACE tablespace | OVERFLOW [TABLESPACE tablespace] | LOB (LOB_item) STORE AS { LOB_segname [(TABLESPACE tablespace)] | (TABLESPACE tablespace) } | VARRAY varray_item STORE AS LOB LOB_segname }]...]

Subclause	Syntax

password_parameters	{ { FAILED_LOGIN_ATTEMPTS
	  | PASSWORD_LIFE_TIME
	  | PASSWORD_REUSE_TIME
	  | PASSWORD_REUSE_MAX
	  | PASSWORD_LOCK_TIME
	  | PASSWORD_GRACE_TIME
	  }
	  { expr | UNLIMITED | DEFAULT }
	| PASSWORD_VERIFY_FUNCTION
	     { function | NULL | DEFAULT }
	}

permanent_tablespace_clause	{ MINIMUM EXTENT integer [K | M]
	| BLOCKSIZE integer [K]
	| logging_clause
	| FORCE LOGGING
	| DEFAULT [table_compression]
	  storage_clause
	| { ONLINE | OFFLINE }
	| extent_management_clause
	| segment_management_clause
	| flashback_mode_clause
	  [MINIMUM EXTENT integer [K | M]
	  | BLOCKSIZE integer [K]
	  | logging_clause
	  | FORCE LOGGING
	  | DEFAULT [table_compression]
	    storage_clause
	  | { ONLINE | OFFLINE }
	  | extent_management_clause
	  | segment_management_clause
	  | flashback_mode_clause
	  ]...
	}

physical_attributes_clause	[{ PCTFREE integer
	  | PCTUSED integer
	  | INITRANS integer
	  | storage_clause
	  }
	    [PCTFREE integer
	    | PCTUSED integer
	    | INITRANS integer
	    | storage_clause
	    ]...
	]

physical_properties	{ segment_attributes_clause
	  [table_compression]
	| ORGANIZATION
	     { HEAP
	          [segment_attributes_clause]

Subclause	Syntax
	[table_compression] | INDEX [segment_attributes_clause] index_org_table_clause | EXTERNAL external_table_clause } | CLUSTER cluster (column [, column]...) }
pragma_clause	PRAGMA RESTRICT_REFERENCES ({ method_name | DEFAULT } , { RNDS | WNDS | RNPS | WNPS | TRUST } [, { RNDS | WNDS | RNPS | WNPS | TRUST }]...)
procedure_declaration	PROCEDURE name (parameter datatype [, parameter datatype]...) { IS | AS } { pl/sql_block | call_spec }
procedure_spec	PROCEDURE name (parameter datatype [, parameter datatype]...) [{ IS | AS } call_spec]
proxy_authentication	{ AUTHENTICATION REQUIRED | AUTHENTICATED USING { PASSWORD | DISTINGUISHED NAME | CERTIFICATE [TYPE 'type'] [VERSION 'version'] } }
proxy_clause	{ GRANT | REVOKE } CONNECT THROUGH proxy [WITH { ROLE { role_name [, role_name]... | ALL EXCEPT role_name [, role_name]... } | NO ROLES }] [proxy_authentication]
qualified_disk_clause	search_string [NAME disk_name] [SIZE size_clause] [FORCE | NOFORCE]

Subclause	Syntax
qualified_template_clause	template_name ATTRIBUTES ([MIRROR | UNPROTECTED] [FINE | COARSE])
query_partition_clause	PARTITION BY { value_expr[, value_expr]... | (value_expr[, value_expr]...)
query_table_expression	{ query_name | [schema.] { table [{ PARTITION (partition) | SUBPARTITION (subpartition) } [sample_clause] | [sample_clause] | @ dblink] | { view | materialized view } [@ dblink] } | (subquery [subquery_restriction_clause]) | table_collection_expression }
quiesce_clauses	QUIESCE RESTRICTED | UNQUIESCE
range_partitioning	PARTITION BY RANGE (column[, column]...) (PARTITION [partition] range_values_clause table_partition_description [, PARTITION [partition] range_values_clause table_partition_description]...)
range_values_clause	VALUES LESS THAN ({ value | MAXVALUE } [, { value | MAXVALUE }]...)
rebalance_diskgroup_clause	REBALANCE [POWER integer]
rebuild_clause	REBUILD [{ PARTITION partition | SUBPARTITION subpartition } | { REVERSE | NOREVERSE }

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td>]
[parallel_clause
| TABLESPACE tablespace
| PARAMETERS ('ODCI_parameters')
| ONLINE
| COMPUTE STATISTICS
| physical_attributes_clause
| key_compression
| logging_clause
]
[parallel_clause
| TABLESPACE tablespace
| PARAMETERS ('ODCI_parameters')
| ONLINE
| COMPUTE STATISTICS
| physical_attributes_clause
| key_compression
| logging_clause
]...</td></tr>
<tr><td>records_per_block_clause</td><td>{ MINIMIZE | NOMINIMIZE } RECORDS_PER_BLOCK</td></tr>
<tr><td>recover_clause</td><td>{ { DISCONNECT [FROM SESSION]
| { TIMEOUT integer | NOTIMEOUT }
}
| { NODELAY | DEFAULT DELAY | DELAY integer }
| NEXT integer
| { EXPIRE integer | NO EXPIRE }
| parallel_clause
| USING CURRENT LOGFILE
| UNTIL CHANGE integer
| THROUGH { [THREAD integer] SEQUENCE integer
| ALL ARCHIVELOG
| { ALL | LAST | NEXT } SWITCHOVER
}
}
[{ DISCONNECT [FROM SESSION]
| { TIMEOUT integer | NOTIMEOUT }
}
| { NODELAY | DEFAULT DELAY | DELAY integer }
| NEXT integer
| { EXPIRE integer | NO EXPIRE }
| parallel_clause
| USING CURRENT LOGFILE
| UNTIL CHANGE integer
| THROUGH { [THREAD integer] SEQUENCE
integer
| ALL ARCHIVELOG
| { ALL | LAST | NEXT } SWITCHOVER
}
] ...</td></tr>
</table>

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>recovery_clauses</td><td>{ general_recovery
| managed_standby_recovery
| BEGIN BACKUP
| END BACKUP
}</td></tr>
<tr><td>redo_log_file_spec</td><td>['filename'
| ('filename' [, 'filename']...)
]
[SIZE size_clause]
[REUSE]</td></tr>
<tr><td>redo_thread_clauses</td><td>{ ENABLE | DISABLE }
  { INSTANCE 'instance_name'
  | [PUBLIC] THREAD integer
  }</td></tr>
<tr><td>reference_model</td><td>REFERENCE reference_spreadsheet_name
ON (subquery)
spreadsheet_column_clauses
[cell_reference_options]</td></tr>
<tr><td>references_clause</td><td>REFERENCES [schema.] { object_table | view }
[(column [, column]...)]
[ON DELETE { CASCADE | SET NULL }]
[constraint_state]</td></tr>
<tr><td>referencing_clause</td><td>REFERENCING
 { OLD [AS] old
 | NEW [AS] new
 | PARENT [AS] parent }
   [OLD [AS] old
   | NEW [AS] new
   | PARENT [AS] parent]...</td></tr>
<tr><td>register_logfile_clause</td><td>REGISTER
[OR REPLACE]
[PHYSICAL | LOGICAL]
LOGFILE
[file_specification
  [, file_specification]...
]
FOR logminer_session_name</td></tr>
<tr><td>relational_properties</td><td>{ column datatype [SORT]
    [DEFAULT expr]
    [inline_constraint
      [inline_constraint]...
    | inline_ref_constraint</td></tr>
</table>

Subclause	Syntax

	]
	| { out_of_line_constraint
	  | out_of_line_ref_constraint
	  | supplemental_logging_props
	  }
	}
	  [, { column datatype [SORT]
	         [DEFAULT expr]
	         [inline_constraint
	           [inline_constraint]...
	         | inline_ref_constraint
	         ]
	     | { out_of_line_constraint
	       | out_of_line_ref_constraint
	       | supplemental_logging_props
	       }
	  ]...

relational_table	CREATE [GLOBAL TEMPORARY] TABLE
	[schema.]table
	   [(relational_properties)]
	   [ON COMMIT { DELETE | PRESERVE } ROWS]
	   [physical_properties]
	   [table_properties] ;

rename_column_clause	RENAME COLUMN old_name TO new_name

rename_index_partition	RENAME { PARTITION partition
	       | SUBPARTITION subpartition }
	   TO new_name

rename_partition_subpart	RENAME { PARTITION | SUBPARTITION } current_name TO new_name

replace_type_clause	REPLACE [invoker_rights_clause] AS OBJECT
	   (attribute datatype [, attribute
	datatype]...
	   [, element_spec [, element_spec]...])

resize_disk_clauses	RESIZE
	{ ALL [SIZE size_clause]
	| DISK
	   disk_name [SIZE size_clause]
	   [, disk_name [SIZE size_clause]]...
	| DISKS IN FAILGROUP
	    failgroup_name [SIZE size_clause]
	    [, failgroup_name [SIZE size_clause]]...
	}

Subclause	Syntax

resource_parameters	{ { SESSIONS_PER_USER
	  | CPU_PER_SESSION
	  | CPU_PER_CALL
	  | CONNECT_TIME
	  | IDLE_TIME
	  | LOGICAL_READS_PER_SESSION
	  | LOGICAL_READS_PER_CALL
	  | COMPOSITE_LIMIT
	  }
	  { integer | UNLIMITED | DEFAULT }
	| PRIVATE_SGA
	  { integer [K | M] | UNLIMITED | DEFAULT }
	}

restricted_session_clauses	{ ENABLE | DISABLE } RESTRICTED SESSION
return_clause	{ RETURN datatype [{ IS | AS } call_spec] | sqlj_object_type_sig }
return_rows_clause	RETURN { UPDATED | ALL } ROWS
returning_clause	RETURNING expr [, expr]... INTO data_item [, data_item]...
revoke_object_privileges	{ object_privilege | ALL [PRIVILEGES] } [, { object_privilege | ALL [PRIVILEGES] }]... on_object_clause FROM grantee_clause [CASCADE CONSTRAINTS | FORCE]

revoke_system_privileges	{ system_privilege
	| role
	| ALL PRIVILEGES
	}
	  [, { system_privilege
	     | role
	     | ALL PRIVILEGES
	     }
	  ]...
	FROM grantee_clause

rollup_cube_clause	{ ROLLUP | CUBE } (grouping_expression_list)
routine_clause	[schema.] [type. | package.] { function | procedure | method } [@dblink_name] ([argument [, argument]...])

Subclause	Syntax
row_movement_clause	{ ENABLE | DISABLE } ROW MOVEMENT

sample_clause	SAMPLE [BLOCK]
	       (sample_percent)
	       [SEED (seed_value)]

schema_object_clause	{ object_option [, object_option]... | ALL } auditing_on_clause

scoped_table_ref_constraint	{ SCOPE FOR
	  ({ ref_column | ref_attribute })
	  IS [schema.] { scope_table_name | c_alias }
	}
	  [, SCOPE FOR
	     ({ ref_column | ref_attribute })
	     IS [schema.] { scope_table_name |
	c_alias }
	  ]...

searched_case_expression	WHEN condition THEN return_expr [WHEN condition THEN return_expr]...
security_clause	GUARD { ALL | STANDBY | NONE }

segment_attributes_clause	{ physical_attributes_clause
	| TABLESPACE tablespace
	| logging_clause
	}
	  [physical_attributes_clause
	  | TABLESPACE tablespace
	  | logging_clause
	  ]...

segment_management_clause	SEGMENT SPACE MANAGEMENT { MANUAL | AUTO }

select_list	{ *
	| { query_name.*
	  | [schema.]
	    { table | view | materialized view } .*
	  | expr [[AS] c_alias]
	  }
	    [, { query_name.*
	       | [schema.]
	         { table | view | materialized view } .*
	       | expr [[AS] c_alias]
	       }
	    ]...
	}

Subclause	Syntax
set_subpartition_template	SET SUBPARTITION TEMPLATE { (SUBPARTITION subpartition [list_values_clause] [partitioning_storage_clause] [, SUBPARTITION subpartition [list_values_clause] [partitioning_storage_clause]...]) | hash_subpartition_quantity }
set_time_zone_clause	SET TIME_ZONE = '{ { + | - } hh : mi | time_zone_region }'
shrink_clause	SHRINK SPACE [COMPACT] [CASCADE]
shutdown_dispatcher_clause	SHUTDOWN [IMMEDIATE] dispatcher_name
simple_case_expression	expr WHEN comparison_expr THEN return_expr [WHEN comparison_expr THEN return_expr]...
single_column_for_loop	FOR dimension_column { IN ({ literal [, literal]... | subquery }) | [LIKE pattern] FROM literal TO literal { INCREMENT | DECREMENT } literal }
single_table_insert	insert_into_clause { values_clause [returning_clause] | subquery }
size_clause	integer [K | M | G | T]
split_index_partition	SPLIT PARTITION partition_name_old AT (value [, value]...) [INTO (index_partition_description, index_partition_description)]

Subclause	Syntax
	[parallel_clause] UPDATE MANUALLY problem importing file
split_table_partition	SPLIT PARTITION current_partition { AT | VALUES } (value [, value]...) [INTO (partition_spec, partition_spec)] [update_index_clauses] [parallel_clause] UPDATE MANUALLY problem importing file
split_table_subpartition	SPLIT SUBPARTITION subpartition VALUES ({ value | NULL } [, value | NULL]...) [INTO (subpartition_spec, subpartition_spec)] [update_index_clauses] [parallel_clause] UPDATE MANUALLY problem importing file
sql_statement_clause	{ { statement_option | ALL } [, { statement_option | ALL }]... | { system_privilege | ALL PRIVILEGES } [, { system_privilege | ALL PRIVILEGES }]... } [auditing_by_clause]
sqlj_object_type	EXTERNAL NAME java_ext_name LANGUAGE JAVA USING (SQLData | CustomDatum | OraData)
sqlj_object_type_attr	EXTERNAL NAME 'field_name'
sqlj_object_type_sig	RETURN { datatype | SELF AS RESULT } EXTERNAL { VARIABLE NAME 'java_static_field_name' | NAME 'java_method_sig' }
standby_database_clauses	(activate_standby_db_clause | maximize_standby_db_clause | register_logfile_clause | commit_switchover_clause | start_standby_clause | stop_standby_clause) [parallel_clause]
start_standby_clause	START LOGICAL STANDBY APPLY [IMMEDIATE] [NODELAY] [NEW PRIMARY dblink

Subclause	Syntax
	| INITIAL [scn_value] | { SKIP FAILED TRANSACTION | FINISH }]

startup_clauses	{ MOUNT [{ STANDBY | CLONE } DATABASE]
	| OPEN { [READ WRITE]
	         [RESETLOGS | NORESETLOGS]
	         [UPGRADE | DOWNGRADE]
	       | READ ONLY
	       }
	}

stop_standby_clause	{ STOP | ABORT } LOGICAL STANDBY APPLY

storage_clause	STORAGE
	   ({ INITIAL integer [K | M]
	    | NEXT integer [K | M]
	    | MINEXTENTS integer
	    | MAXEXTENTS { integer | UNLIMITED }
	    | PCTINCREASE integer
	    | FREELISTS integer
	    | FREELIST GROUPS integer
	    | OPTIMAL [integer [K | M]
	              | NULL
	              ]
	    | BUFFER_POOL { KEEP | RECYCLE | DEFAULT }
	    }
	      [INITIAL integer [K | M]
	      | NEXT integer [K | M]
	      | MINEXTENTS integer
	      | MAXEXTENTS { integer | UNLIMITED }
	      | PCTINCREASE integer
	      | FREELISTS integer
	      | FREELIST GROUPS integer
	      | OPTIMAL [integer [K | M]
	                | NULL
	                ]
	      | BUFFER_POOL { KEEP | RECYCLE | DEFAULT }
	      ]...
	   )

streaming_clause	{ ORDER | CLUSTER } BY (column [, column]...)

subpartition_by_hash	SUBPARTITION BY HASH (column [, column]...)
	   [SUBPARTITIONS quantity
	        [STORE IN (tablespace [,
	tablespace]...)]
	   | subpartition_template
	   ]

Subclause	Syntax
subpartition_by_list	SUBPARTITION BY LIST (column) [subpartition_template]
subpartition_spec	SUBPARTITION [subpartition] [list_values_clause] [partitioning_storage_clause]
subpartition_template	SUBPARTITION TEMPLATE (SUBPARTITION subpartition [list_values_clause] [partitioning_storage_clause] [, SUBPARTITION subpartition [list_values_clause] [partitioning_storage_clause]]) | hash_subpartition_quantity
subprogram_declaration	{ MEMBER | STATIC } { procedure_declaration | function_declaration | constructor_declaration }
subprogram_spec	{ MEMBER | STATIC } { procedure_spec | function_spec }
subquery	[subquery_factoring_clause] SELECT [hint] [{ { DISTINCT | UNIQUE } | ALL }] select_list FROM table_reference [, table_reference]... [where_clause] [hierarchical_query_clause] [group_by_clause] [HAVING condition] [model_clause] [{ UNION [ALL] | INTERSECT | MINUS } (subquery)] [order_by_clause]

Subclause	Syntax
subquery_factoring_clause	WITH query_name AS (subquery) [, query_name AS (subquery)]...

subquery_restriction_clause	WITH { READ ONLY
	     | CHECK OPTION [CONSTRAINT constraint]
	     }

substitutable_column_clause	[ELEMENT] IS OF [TYPE] ([ONLY] type) | [NOT] SUBSTITUTABLE AT ALL LEVELS
supplemental_db_logging	{ ADD | DROP } SUPPLEMENTAL LOG { DATA | supplemental_id_key_clause }

supplemental_id_key_clause	DATA
	({ ALL
	 | PRIMARY KEY
	 | UNIQUE
	 | FOREIGN KEY
	 }
	   [, { ALL
	      | PRIMARY KEY
	      | UNIQUE
	      | FOREIGN KEY
	      }
	  ]...
	)
	COLUMNS

supplemental_log_grp_clause	GROUP log_group (column [NO LOG] [, column [NO LOG]]...) [ALWAYS]
supplemental_logging_props	{ supplemental_log_grp_clause | supplemental_id_key_clause }

supplemental_table_logging	{ ADD SUPPLEMENTAL LOG
	        { supplemental_log_grp_clause
	        | supplemental_id_key_clause
	        }
	      [, SUPPLEMENTAL LOG
	           { supplemental_log_grp_clause
	           | supplemental_id_key_clause
	           }
	      ]...
	| DROP SUPPLEMENTAL LOG
	         { supplemental_id_key_clause
	         | GROUP log_group

Subclause	Syntax

	}
	       [, SUPPLEMENTAL LOG
	            { supplemental_id_key_clause
	            | GROUP log_group
	            }
	       ]...
	}

table_collection_expression	TABLE (collection_expression) [(+)]
table_compression	{ COMPRESS | NOCOMPRESS }
table_index_clause	[schema.]table [t_alias] (index_expr [ASC | DESC] [, index_expr [ASC | DESC]]...) [index_properties]

table_partition_description	[segment_attributes_clause]
	[table_compression | key_compression]
	[OVERFLOW [segment_attributes_clause]]
	[{ LOB_storage_clause
	  | varray_col_properties
	  }
	    [LOB_storage_clause
	    | varray_col_properties
	    ]...
	]
	[partition_level_subpartition]

table_partitioning_clauses	{ range_partitioning | hash_partitioning | list_partitioning | composite_partitioning }
table_properties	[column_properties] [table_partitioning_clauses] [CACHE | NOCACHE] [parallel_clause] [ROWDEPENDENCIES | NOROWDEPENDENCIES] [enable_disable_clause] [enable_disable_clause]... [row_movement_clause] [AS subquery]

table_reference	{ ONLY
	  (query_table_expression)
	  [flashback_query_clause]
	  [t_alias]

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td></td><td>| query_table_expression
  [flashback_query_clause]
  [t_alias]
| (join_clause)
| join_clause
}</td></tr>
<tr><td>tablespace_clauses</td><td>{ EXTENT MANAGEMENT LOCAL
| DATAFILE file_specification
         [, file_specification]...
| SYSAUX DATAFILE file_specification
                [, file_specification]...
| default_tablespace
| default_temp_tablespace
| undo_tablespace
}</td></tr>
<tr><td>tablespace_group_clause</td><td>TABLESPACE GROUP { tablespace_group_name | '' }</td></tr>
<tr><td>tablespace_logging_clauses</td><td>{ logging_clause
| [NO] FORCE LOGGING
}</td></tr>
<tr><td>tablespace_retention_clause</td><td>RETENTION { GUARANTEE | NOGUARANTEE }</td></tr>
<tr><td>tablespace_state_clauses</td><td>{ ONLINE
| OFFLINE [NORMAL | TEMPORARY | IMMEDIATE]
}
| READ { ONLY | WRITE }
| { PERMANENT | TEMPORARY }</td></tr>
<tr><td>temporary_tablespace_clause</td><td>TEMPORARY TABLESPACE tablespace
  [TEMPFILE file_specification
           [, file_specification]...
  ]
  [tablespace_group_clause]
  [extent_management_clause]</td></tr>
<tr><td>text</td><td>[N | n]
{ 'c [c]...'
| { Q | q }
  'quote_delimiter c [c]... quote_delimiter'
}</td></tr>
<tr><td>trace_file_clause</td><td>TRACE
[AS 'filename' [REUSE]]
[RESETLOGS | NORESETLOGS]</td></tr>
</table>

Subclause	Syntax

truncate_partition_subpart	TRUNCATE { PARTITION partition
	         | SUBPARTITION subpartition
	         }
	   [{ DROP | REUSE } STORAGE]
	   [update_index_clauses [parallel_clause]]

undo_tablespace	[BIGFILE | SMALLFILE] UNDO TABLESPACE tablespace [TABLESPACE file_specification [, file_specification]...]

undo_tablespace_clause	UNDO TABLESPACE tablespace
	  [DATAFILE file_specification
	             [, file_specification]...
	  ]
	  [extent_management_clause]
	  [tablespace_retention_clause]

undrop_disk_clause	UNDROP DISKS

update_all_indexes_clause	UPDATE INDEXES
	   [(index ({ update_index_partition
	              | update_index_subpartition
	              }
	            )
	     )
	       [, (index ({ update_index_partition
	                   | update_index_subparition
	                   }
	                 )
	          )
	       ]...

update_global_index_clause	{ UPDATE | INVALIDATE } GLOBAL INDEXES
update_index_clauses	{ update_global_index_clause | update_all_indexes_clause }

update_index_partition	PARTITION [partition]
	   [index_partition_description
	     [index_subpartition_clause]
	   ]
	[, PARTITION [partition]
	      [index_partition_description
	        [index_subpartition_clause]
	      ]
	]...

<table>
<tr><th>Subclause</th><th>Syntax</th></tr>
<tr><td>update_index_subpartition</td><td>SUBPARTITION [subpartition]
[TABLESPACE tablespace]
[, SUBPARTITION [subpartition]
[TABLESPACE tablespace]
]...</td></tr>
<tr><td>update_set_clause</td><td>SET
{ { (column [, column]...) = (subquery)
| column = { expr | (subquery) | DEFAULT }
}
[, { (column [, column]...) = (subquery)
| column = { expr | (subquery) |
DEFAULT }
}
]...
| VALUE (t_alias) = { expr | (subquery) }
}</td></tr>
<tr><td>upgrade_table_clause</td><td>UPGRADE [[NOT] INCLUDING DATA]
[column_properties]</td></tr>
<tr><td>using_function_clause</td><td>USING [schema.] [package. |
type.]function_name</td></tr>
<tr><td>using_index_clause</td><td>USING INDEX
{ [schema.]index
| (create_index_statement)
| index_properties
}</td></tr>
<tr><td>using_statistics_type</td><td>USING { [schema.] statistics_type | NULL }</td></tr>
<tr><td>using_type_clause</td><td>USING [schema.]implementation_type
[array_DML_clause]</td></tr>
<tr><td>validation_clauses</td><td>{ VALIDATE REF UPDATE
[SET DANGLING TO NULL]
| VALIDATE STRUCTURE
[CASCADE]
[into_clause]
{ OFFLINE| ONLINE }
}</td></tr>
<tr><td>values_clause</td><td>VALUES ({ expr | DEFAULT }
[, { expr | DEFAULT }]...
)</td></tr>
</table>

Subclause	Syntax
varray_col_properties	VARRAY varray_item { [substitutable_column_clause] STORE AS LOB { [LOB_segname] (LOB_parameters) | LOB_segname } | substitutable_column_clause }
where_clause	WHERE condition
windowing_clause	{ ROWS | RANGE } { BETWEEN { UNBOUNDED PRECEDING | CURRENT ROW | value_expr { PRECEDING | FOLLOWING } } AND { UNBOUNDED FOLLOWING | CURRENT ROW | value_expr { PRECEDING | FOLLOWING } } | { UNBOUNDED PRECEDING | CURRENT ROW | value_expr PRECEDING } }
XML_attributes_clause	XMLATTRIBUTES (value_expr [AS c_alias] [, value_expr [AS c_alias]...)
XMLSchema_spec	[XMLSCHEMA XMLSchema_URL] ELEMENT { element | XMLSchema_URL # element }
XMLType_column_properties	XMLTYPE [COLUMN] column [XMLType_storage] [XMLSchema_spec]
XMLType_storage	STORE AS { OBJECT RELATIONAL | CLOB [{ LOB_segname [(LOB_parameters)] | LOB_parameters }]
XMLType_table	CREATE TABLE [GLOBAL TEMPORARY] TABLE [schema.]table OF XMLTYPE

Subclause	Syntax
	[(oject_properties)] [XMLTYPE XMLType_storage] [XMLSchema_spec] [ON COMMIT { DELETE | PRESERVE } ROWS] [OID_clause] [OID_index_clause] [physical_properties] [table_properties] ;
XMLType_view_clause	OF XMLTYPE [XMLSchema_spec] WITH OBJECT IDENTIFIER { DEFAULT | (expr [, expr]...) }

Appendix E

Acronyms and Terms

Term	Definition
Active session pool	Number of current active sessions allowed for a resource group or subplan
ADDM	Automatic Database Diagnostic Monitor
ASM	Automatic Storage Management
ASM	Automatic Summary Management
ASMM	Automatic Shared Memory Management
ASSM	Automatic Segment Space Management
ATO	Automatic Tuning Optimizer
Automatic PGA Memory Management	A feature of the Oracle database that simplifies and improves the way PGA memory is allocated
Automatic Shared Memory Management	A feature of the Oracle database that automates the management of the most important shared memory structures used by an Oracle database instance
Automatic Storage Management	A mechanism that provides a vertical integration of the file system and the volume manager, specifically built for the Oracle database files
Automatic Database Diagnostic Management	A utility that performs a top-down instance analysis, identifies problems and potential causes, and makes recommendations for fixing the problems
Automatic Tuning Optimizer	A database feature that performs various analyses of SQL performance within the database
Automatic Workload Repository	An infrastructure that collects, processes, and maintains performance statistics for problem detection and self-tuning purposes
Auxiliary database	A database that is used when creating a duplicate database or performing tablespace point-in-time recovery
AWR	Automatic Workload Repository
Backup piece	An individual file that is part of a backup set
Backup set	A copy of one or more data or archived log files. It differs from image copies in that empty blocks are not stored.
Block change tracking	A feature that uses the change tracking writer (CTWR) background process to record the physical location of all database changes in a separate file
Block corruption	Corruption of a data block. A corrupted data block is a block that is not in a recognized Oracle format, or whose contents are not internally consistent.
Block Media Recovery	A recovery method that reduces the smallest recoverable unit of media recovery from a data file to a block
BMR	Block Media Recovery
Buffer cache	A region of memory that caches blocks of data retrieved from the database
CFS	Cluster File Storage

Term	Definition
Change tracking file	A file used to store the physical location of database changes made since the last backup
Channel	A link or connection to a target database
CLI	Command-line interpreter
Cluster	A group of one or more tables that share the same data blocks
CMAN	Oracle Connection Manager. It functions as a net traffic firewall and proxy server.
Control file	A file that contains information about the physical structure of the database, including the locations of all data and redo log files
CRS	Cluster Ready Services
Data block	The smallest unit of physical storage within the database. Data blocks contain rows of data, index information, and so on.
Data dictionary cache	An area of memory within the shared pool that holds the definitions of dictionary objects in memory
Data file	A file that contains data for the database
Database Character Set Scanner	A utility that assesses the feasibility of migrating an Oracle database to a new database character set
DBA	Database administrator
DBA	Data block address, which is used to uniquely identify a data block within the database
DBCA	Database Configuration Assistant
DBVERIFY	An external command-line utility that performs a physical data structure integrity check on an offline database
DDL	Data definition language. It is the class of SQL statements that define and manipulate database objects.
DML	Data manipulation language. It is the class of SQL statements that query and manipulate data.
EM	Enterprise Manager
`emctl`	Enterprise Manager Control. It is a utility for starting, stopping, and checking the status of Database Control, the Oracle Agent, and Oracle Management servers.
Encoded character set	A character set that maps numeric codes to characters that a computer or terminal can display and receive
Enterprise Manager Database Control Console	A graphical interface used to manage the database
EXTPROC	External code libraries
FGA	Fine-grained auditing
FGAC	Fine-Grained Access Control
Flash recovery area	A unified storage location for all recovery-related files and activities in an Oracle database
Flashback buffer	An area in memory that stores Flashback Database data
Flashback Database	A new recovery method that uses Undo data, instead of Redo data, to recover the database

Term	Definition
Flashback Drop	A feature that enables you to undo the effects of a `DROP TABLE` statement without resorting to traditional point-in-time recovery
Flashback Table	A command that enables you to recover a table and all its dependent objects from the recycle bin
Flashback Transaction Query	A diagnostic tool that you can use to view changes made to the database at the transaction level
Flashback Versions Query	A query syntax that provides a history of changes made to a row along with the corresponding identifier of the transaction that made the change
Format mask elements	A character literal that describes the format of datetime or numeric data stored in a character string
Growth trend report	Analysis of the growth of database segments
Globalization support	A feature set that ensures that utilities and error messages, sort order, alphabet, calendar, date, time, money, and numbers automatically adapt to the native language
Image copy	A bit-for-bit identical copy of a database file
Incarnation	A separate version of a physical database. The incarnation of the database changes when you open it with the `RESETLOGS` option, but you can recover backups from an earlier incarnation so long as the necessary redo is available.
Index-organized tables	A database structure that has the appearance of a table but stores its data in a B*Tree structure
`init.ora` or `init`<*sid*>`.ora`	The initialization parameter file that controls how the database instance is configured and run at startup time. Also known as “parameter file”.
Instance	The collection of shared memory and processes used to access the Oracle database
IPC	Internal Process Communication
`isqlplusctl`	Control utility for starting and stopping *i*SQL*Plus listener processes
ISV	Independent software vendor
Java pool	A region of memory in the SGA that is used for all session-specific Java code and data within the Java Virtual Machine (JVM)
JDBC	Java Database Connectivity
`j`*nnn*	Job Queue Processes. They execute scheduled jobs.
Keep buffer cache	An area of memory in the SGA used to cache data in the buffer cache for longer periods of time
Language and Character Set File Scanner	A statistic-based utility for determining the language and character set for unknown file text
Large pool	An optional memory storage area used for buffering large I/O requests
LCSSCAN	Language and Character Set File Scanner

Term	Definition
LEGATO® NetWorker, Single-Server Version	Software included with Oracle Database 10*g* that enables the Recovery Manager utility to write to tape drives
Library cache	An area of memory within the shared pool that contains the fully parsed or compiled representations of PL/SQL blocks and SQL statements
Linguistic sort	A feature that produces a sort sequence that matches the alphabetic sequence of characters, and not their numeric values in the character encoding scheme
Listener	The gateway to the Oracle instance for all nonlocal user connections
Locale	A collection of information about the linguistic and cultural preferences from a particular region
Locale variants	A language-dependent territory definition
LSSV	LEGATO® NetWorker, Single-Server Version
Media management library	An interface used by RMAN when writing to or reading from tapes
Memory Advisor	A feature of Enterprise Manager that helps you tune the size of your memory structures
Memory Manager (MMAN)	A database background process that serves as the SGA memory broker and coordinates the sizing of memory components
Metric	A measurement of some database or instance characteristic
MML	Media management library
MMON	Management Monitor Process. This process issues alerts whenever a metric violates its threshold value. It captures statistics for SQL objects that have been recently modified.
National Language Support	Parameters and files that determine the locale-specific behavior of the database client and the database server
*n*K block size buffer	A region of memory in the SGA, which caches data blocks that are of a different size than the default database block size. It is used to support transportable tablespaces.
NLS	National Language Support
NLS Runtime Library	A comprehensive suite of language-independent functions that allow proper text and character processing and language-convention manipulations
`NLS_LANG`	An environment variable used to specify the language, territory, and character set used by a database
NLSRTL	National Language Support Runtime Library
NMP	Named Pipes
OC4J	Oracle Application Server Containers for J2EE
OMF	Oracle Managed Files
Optimizer statistics	Statistics that describe the database and the objects in the database, and are used by the query optimizer to choose the best execution plan for each SQL statement
OUI	Oracle Universal Installer

Term	Definition
Oracle Locale Builder	A feature that provides a graphical user interface through which you can easily view, modify, and define locale-specific data
Oracle Managed Files	A feature of the Oracle database, which manages the creation, naming, and deletion of Oracle database files within dedicated areas of the disk
Oracle Net	An interface that enables network connections between Oracle Database 10*g* and client or middle-tier applications
Oracle Shared Server	A database server configuration that allows many user processes to share a small number of server processes, minimizing the number of server processes and maximizing the use of available system resources
`ORACLE_BASE`	Environment variable used to point to the base of the OFA structure
`ORACLE_HOME`	Environment variable used to identify a directory containing Oracle software
`ORACLE_SID`	Environment variable used to specify the default database instance name
Package	A collection of procedures and function definitions that are logically related. The procedures and functions are implemented by the package body.
Parallelization	Allocation of multiple channels for RMAN backup and recovery operations
PGA	Program Global Area
PGA Advisor	A feature of Enterprise Manager that gives detailed statistics for the work areas and provides recommendations about optimal usage of Program Global Area (PGA) memory on the basis of workload characteristics
Pipe	An area of memory used by one process to pass information to another
Private SQL area	An area of memory in the PGA that contains data such as bind information and run-time memory structures
Privilege	The right to execute a particular type of SQL statement. There are two basic forms of privileges: object and system.
Proactive Tablespace Monitoring	A feature of Oracle Database 10*g* that manages tablespace disk space usage
Program Global Area	Private memory area for use by a process
Recovery catalog	A separate database that keeps historical data concerning backup activities
Recovery Manager	The Oracle utility used to back up and restore database files
Recycle bin	A data dictionary table that maintains the relationships between the original names of dropped objects and their system-generated names
Recycle buffer cache	A region of memory in the SGA, which holds data that is quickly aged out of the buffer cache

Term	Definition
Redo log buffer	A region of memory that caches redo information until it can be written to disk
Redo Log File Sizing Advisor	A feature of Enterprise Manager that offers redo log file-sizing advice
Resource Manager	A feature of the Oracle database that gives the Oracle database server more control over resource management decisions, thus circumventing problems resulting from inefficient operating system management
Resumable space allocation	A means for suspending, and later resuming, the execution of large database operations in the event of space allocation failures
RMAN	Recovery Manager
RMAN Repository	A storage structure that maintains metadata about a database's backup and recovery operations
Scheduler	A new database feature that enables database administrators and application developers to control when and where various tasks take place in the database environment
SCN	System change number
Segment Advisor	An advisor that monitors object space issues and analyzes growth trends
Segment Resource Estimator	The new segment resource estimation feature that enables you to estimate the amount of resources that the creation of a new segment would require
Server sessions	The server processes (UNIX) or threads (Windows NT/2000) invoked by a client utility to connect to the target database
Session memory	Memory in the PGA that is allocated to hold session variables and other information related to the session
SGA	System Global Area. It is the memory area shared by all server and background processes.
SGA Advisor	An advisor that makes recommendations for SGA-related parameter settings
Shared pool	A region of memory that caches various constructs that can be shared among users
Shrink Advisor	See the Segment Advisor.
SID	System Identifier. It defaults to the database name and uniquely identifies the instance on a given server.
SQL	Structured Query Language
SQL Access Advisor	A tool that determines optimal data access path (for example, the use of indexes and materialized views)
SQL Tuning Advisor	A tool that provides tuning advice for SQL statements
Statspack	A set of SQL, PL/SQL, and SQL*Plus scripts that allow the collection, automation, storage, and viewing of performance data. This feature has been replaced by the Automatic Workload Repository.
Streams pool	An optional region of memory in the SGA that is used by Oracle Streams

Term	Definition
System statistics	Statistics that describe the system’s hardware characteristics, such as I/O and CPU performance and utilization, to the query optimizer
Tablespace	A logical grouping of data files
Target database	The database that you are attempting to connect to
Threshold	A boundary value against which metric values are compared
Undo Advisor	A feature of Enterprise Manager that suggests parameter values and the amount of additional space that is needed to support flashback for a specified time
Undo data	A copy of original data stored whenever a DML transaction changes data. Undo data is used to roll back a transaction and to provide read-consistent views of changing data.
User Global Area	An area of memory within the shared pool or large pool that contains the session information for the Oracle shared server sessions
UTC	Universal Time Coordinates. This is a global time stamp in the Uniform Audit Trail.
VPD	Virtual Private Database
Wait event	Statistics that are incremented by a server process or thread to indicate that the process had to wait for an event to complete before being able to continue processing
Work area	A private allocation of memory in the PGA, used for sorts, hash joins, and other operations that are memory intensive
Workload repository	See AWR.

F

Next Steps
Continuing Your Education

Where Do You Go from Here?

"To stay competitive in the tech industry, never stop learning. Always be on the lookout for better ways of doing things and new technologies. Our industry does not reward people who let themselves stagnate"

–John Hall, Senior Vice President, Oracle University

Here are a few resources to help you with continuing your education.

F-2

Continuing Education Resources

The resources to learn more about administering Oracle Database 10*g* include:

- **Oracle University Web site**
- **Oracle University University Knowledge Center**
- **Oracle Technology Network:**
  - **Oracle by Example**
  - **Oracle Magazine**
- **Technical support: Oracle *Meta*Link**

F-3

Oracle University

Learn Oracle from Oracle! No one knows Oracle technology better than Oracle University.

- **Worldwide education services**
- **Consistently high customer satisfaction**
- **Learn with the format that best suits *your* needs:**
  - **Instructor-Led Inclass Training**
  - **Live Web Class**
  - **Self-Study CD-ROMs**

Oracle University

http:// www.oracle.com/education

ORACLE

F-4

Oracle University

Oracle University is the world's largest corporate educator with education centers around the globe and has one of the highest customer satisfaction ratings in the industry (over 95%). Oracle University instructors are experts on Oracle technology and can help you get the most from your software.

Continuing Your Education

- Recommended follow-on classes:
  - Oracle Database 10*g*: Administration Workshop II
  - Oracle Database 10*g*: SQL Fundamentals I & II
  - Oracle Database 10*g*: PL/SQL Fundamentals
- Grid technology specialty courses:
  - Oracle Enterprise Manager 10*g* Grid Control
  - Oracle Database 10*g*: Real Application Clusters
  - Oracle Database 10*g*: Implement Streams
  - Oracle Database 10*g*: Data Guard Administration
- Other specialty courses

ORACLE

F-5

Continuing Your Education

The *Oracle Database 10g: Administration Workshop II* course continues your training as a database administrator. You cover advanced database recovery strategies, performance monitoring and tuning, and distributed data concepts.

In this course, SQL and PL/SQL are discussed. Because both of these topics are vast, you are provided with only an overview. You can find additional training on these topics that can enhance your abilities as an administrator.

Oracle recommends that you complete the *Oracle Database 10g: Administration Workshop II* course, before beginning specialty courses.

Consult Oracle University's website for an up-to-date list of all courses. Other specialty courses include:

- *Oracle Database 10g:Security*
- *Oracle Database 10g:Implement and Administer a Data Warehouse*

Oracle University Knowledge Center

Extend your learning with the Knowledge Center to gain access to thousands of hours of training on everything from fundamental SQL to advanced database topics. Topics of interest include:

- **Oracle Database 10*g*: Manage Jobs with the Scheduler**
- **Oracle Database 10*g*: Maximize Performance—General Enhancements**
- **OCA and OCP preparation material**

LEARN ORACLE FROM ORACLE | Oracle University

http://ilearning.oracle.com

ORACLE

F-6

Oracle University Knowledge Center

Oracle University Knowledge Center contains thousands of hours of content on a variety of topics. This content can be accessed 24/7 via the Web and is a great way to refresh your memory on things that you may not have worked with for a while or to learn about new concepts.

Additional content for the Oracle University Knowledge Center is being developed continuously, so make sure to check back often to see what is new.

Oracle Technology Network

Oracle Technology Network is a *free* resource with information about the core Oracle software products, including database, Application Server, Collaboration Suite, and development tools. You can have access to:

- **Technology centers**
- **Discussion forums**
- **Software downloads**
- **Online documentation**
- **Oracle by Example**
- **Code samples**

and much more!

http://www.oracle.com/technology

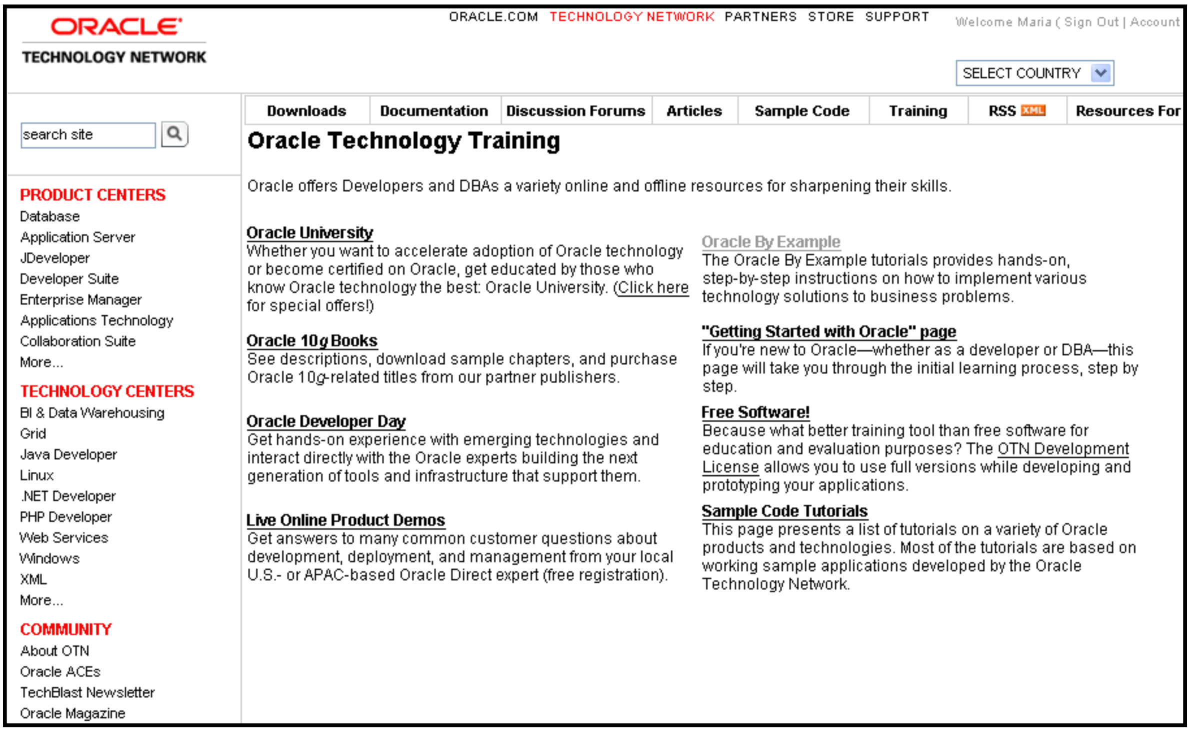
F-7

Oracle Technology Network

Oracle Technology Network (OTN) hosts the latest news about Oracle technology and products. Additionally, OTN provides peer-to-peer forums, white papers, security bulletins, and other vital information for the Oracle professional.

In addition to tips, tricks, and techniques for getting the most out of your Oracle software, you can download that software from OTN. Remember: All software downloads are free, and each comes with a development license that allows you to use full versions of the products only when developing and making prototypes your applications.

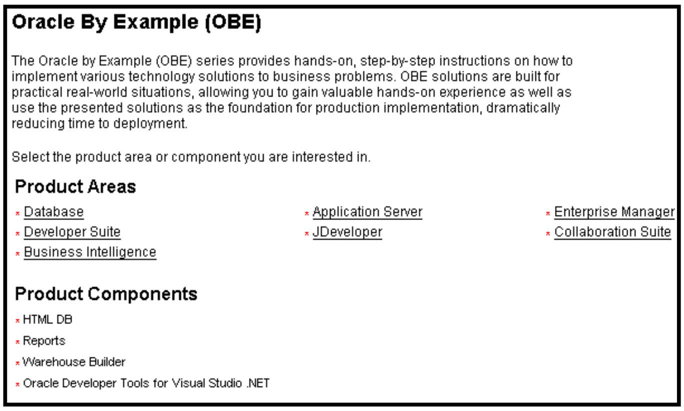
Oracle Technology Training



ORACLE

Oracle by Example

- **What is an OBE?**
  - **A set of hands-on, step-by-step instructions**
- **Where can I find them?**
  - **http://www.oracle.com//technology/obe**
- **What is available?**
  - **Over 100 database OBEs**



ORACLE

F-9

Oracle by Example

The Oracle by Example (OBE) series provides hands-on, step-by-step instructions on how to use various new features of Oracle products. OBEs help to reduce the time spent on learning a new product capability and enhance the users’ understanding of how the feature can be implemented in their environment. Currently, OBEs are available for the Oracle database, Oracle Application Server, and Oracle Collaboration Suite. OBEs can be accessed at http://www.oracle.com/technology/obe.

Oracle Magazine

- Free subscription
- Oracle Magazine Archives

  http://www.oracle.com/technology/oramag/index.html

Oracle Magazine

Among the many different types of resources to which you have access from OTN is Oracle Magazine. You can receive your free subscription also by mail.

Oracle Applications Community

Oracle Technology Network is a resource for Oracle Applications users and implementers. You can have access to:

- **Discussion Forums**
- **User Groups**
- **Online Chat**
- **Documentation**
- **Training**
- **Upgrade Information**

and much more!

http://www.oracle.com/technology/community/apps/index.html

F-11

Technical Support: Oracle *Meta*Link

Access to Oracle *Meta*Link is included as part of your annual support maintenance fees. In addition to the most up-to-date technical information available, *Meta*Link gives you access to:

- **Service requests (SRs)**
- **Certification matrices**
- **Technical forums monitored by Oracle experts**
- **Software patches**
- **Bug reports**

MetaLink

http://metalink.oracle.com

ORACLE

Oracle *Meta*Link

Oracle *Meta*Link is your gateway to Oracle's Support resources. Here, you find answers to the most common issues facing Oracle administrators and developers, as well as resources to solve many of those issues.

Like Oracle Technology Network, *Meta*Link includes the most recent headlines about issues that affect the Oracle professional.

Thank You!

We hope your experience with Oracle University has been enjoyable. We welcome your feedback on how we can improve to better meet your needs:

- **End-of-course evaluations**
- **Oracle University Office of Customer Satisfaction**
- **Oracle Education Services**

We hope to see you in class again soon.

ORACLE

F-13

Thank You!

Oracle University's mission is to enhance the adoption of Oracle technology. Our goal is to partner with you, providing information that is pertinent, timely, and relevant to your needs.

Please take a minute to complete the end-of-course evaluation and let us know how we can serve you better. In the U.S., feel free to e-mail our office of customer satisfaction at:

`customersat_us@oracle.com`

If you have questions about continuing your Oracle education, need help finding a class, or want to arrange for on-site training at your company, contact Oracle Education Services for assistance. In the U.S., dial 800.529.0165. For contact numbers outside the U.S., visit the following Web site:

http://www.oracle.com/education/index.html?contact.html

Thanks again and hope to see you in another class!

Index

A

B

C

D

E

F

G

I

J

L

L

M

N

O

S