

Oracle Database 11g: Administration Workshop I

Volume 1 - Student Guide

D50102GC10

Edition 1.0

September 2007

D52683 L\$

ORACLE®

Authors

Priya Vennapusa
James Spiller
Maria Billings

Technical Contributors and Reviewers

Christian Bauwens
Sangram Dash
Andy Fortunak
Gerlinde Frenzen
Steve Friedberg
Joel Goodman
Magnus Isaksson
Akira Kinutani
Pete Jones
Pierre Labrousse
Gwen Lazenby
Hakan Lindfors
Srinivas Putrevu
Andreas Reinhardt
Ira Singer
Jenny Tsai

Editors

Richard Wallis
Amitha Narayan

Graphic Designer

Rajiv Chandrabhanu

Publishers

Nita Brozowski
Michael Sebastian Almeida

Copyright © 2007, Oracle. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

I Introduction

ORACLE®

Copyright © 2007, Oracle. All rights reserved.

Course Objectives

After completing this course, you should be able to:

- **Install, create, and administer Oracle Database 11g**
- **Configure the database for an application**
- **Employ basic monitoring procedures**
- **Implement a backup and recovery strategy**
- **Move data between databases and files**

ORACLE

I - 2

Copyright © 2007, Oracle. All rights reserved.

Course Objectives

In this course, you install the Oracle Database 11g 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 you configure your database. You also learn how to protect it by designing a backup and recovery strategy and how to monitor it to ensure that it operates smoothly.

Suggested Schedule

- | | | | |
|----------|---------------------------------------------------------------------------|----------|----------------------------------------------------------------------------------|
| 1 | 1. Introduction
2. Installation
3. Database Creation
4. Instance | 4 | 12. Proactive Maintenance
13. Performance
14. Backup and Recovery Concepts |
| 2 | 5. Network
6. Storage
7. Users
8. Schema | 5 | 15. Backup
16. Recovery
17. Moving Data
18. Enhancing Database |
| 3 | 9. Data and Concurrency
10. Undo
11. Security | | |

Oracle Products and Services

- **Oracle Database**
- **Oracle Application Server**
- **Oracle Applications**
- **Oracle Collaboration Suite**
- **Oracle Developer Suite**
- **Oracle Services**



ORACLE®

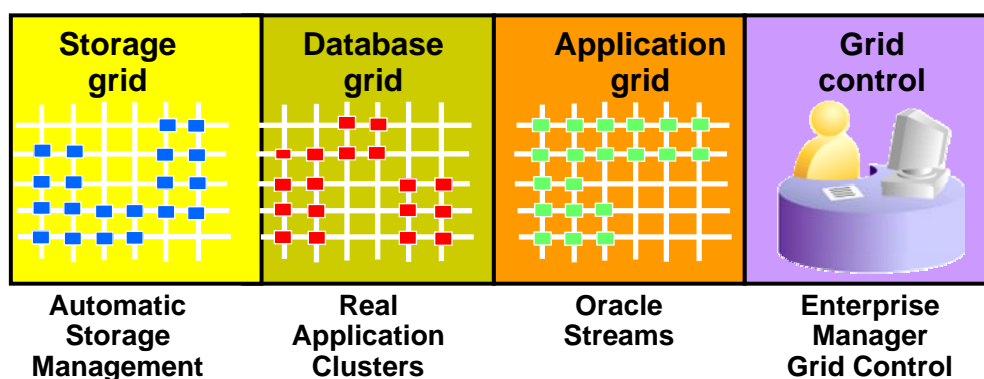
ORACLE®

Oracle Products and Services

- **Oracle Database:** 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 such as 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, email, 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 links to a variety of resources, see the appendix titled “Next Steps: Continuing Your Education.”

Oracle Database 11g: “g” Stands for Grid

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



I - 5

Copyright © 2007, Oracle. All rights reserved.

ORACLE

Oracle Database 11g: “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 11g: “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. 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:

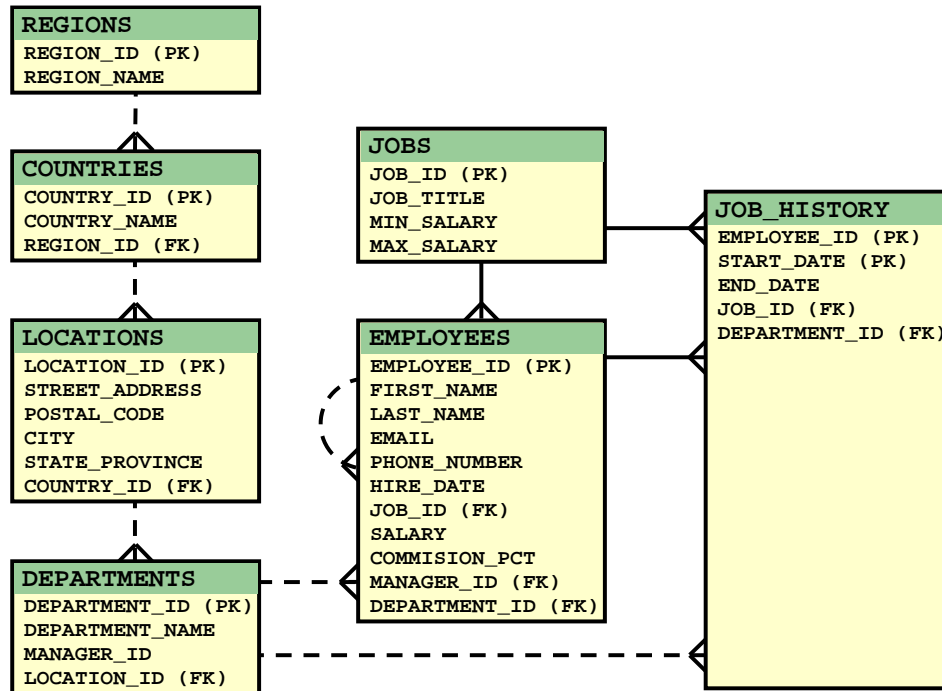
- **Integrated clusterware:** This includes functionality for cluster connectivity, messaging and locking, cluster control, and recovery. It is available on all platforms that are supported by Oracle Database 10g.
- **Automatic workload management:** Rules can be defined to automatically allocate processing 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 10g 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 functionalities 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.

Course Examples: HR Sample Schema



ORACLE

I - 7

Copyright © 2007, Oracle. All rights reserved.

Course Examples: 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 only one department.
- Each job must be a job for one or more employees. Each employee must be currently assigned to 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 only one manager. The FK is optional because the top employee does not report to another employee.

1

Exploring the Oracle Database Architecture

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- **List the major architectural components of Oracle Database**
- **Explain the memory structures**
- **Describe the background processes**
- **Correlate the logical and physical storage structures**



ORACLE

Objectives

This lesson provides a detailed overview of the Oracle Database architecture. You learn about the physical and logical structures and about various components.

Oracle Database

The Oracle relational database management system (RDBMS) provides an open, comprehensive, integrated approach to information management



ORACLE

1 - 3

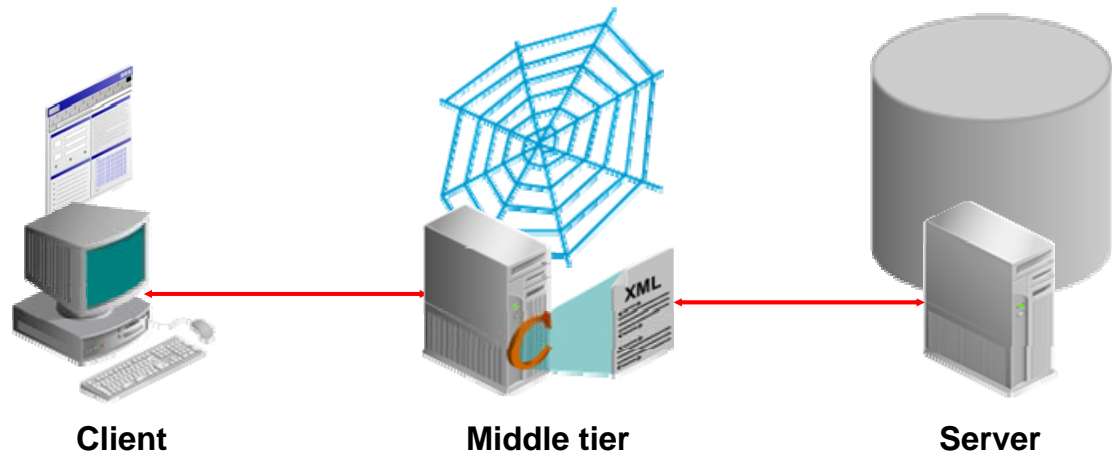
Copyright © 2007, Oracle. All rights reserved.

Oracle Database

A database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information.

Oracle Database reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. This is accomplished while delivering high performance. At the same time, it prevents unauthorized access and provides efficient solutions for failure recovery.

Connecting to a Server



Multitier architecture shown

ORACLE

1 - 4

Copyright © 2007, Oracle. All rights reserved.

Connecting to a Server

A database user can connect to an Oracle server in one of three ways:

- The user logs on to the operating system running the Oracle instance and starts an application or tool that accesses the database on that system. The communication pathway is established using the interprocess communication mechanisms available on the host operating system.
- The user starts the application or tool on a local computer and connects over a network to the computer running the Oracle database. In this configuration (called *client/server*), network software is used to communicate between the user and the back-end server.

The client/server architecture database system has two parts: a front end (client) and a back end (server) connected through a network. Network software is used to communicate between the user and the Oracle server.

- The client is a database application that initiates a request for an operation to be performed on the database server. It requests, processes, and presents data managed by the server. The client workstation can be optimized for its job. For example, the client might not need large disk capacity, or it might benefit from graphic capabilities. Often, the client runs on a different computer than the database server. Many clients can simultaneously run against one server.

Connecting to a Server (continued)

- The server runs Oracle Database software and handles the functions required for concurrent, shared data access. The server receives and processes requests that originate from client applications. The computer that manages the server can be optimized for its duties. For example, the server computer can have large disk capacity and fast processors.
- The user accesses an application server through a tool (such as a Web browser) on the local computer (client). The application server then interacts with a back-end database server on behalf of the client.

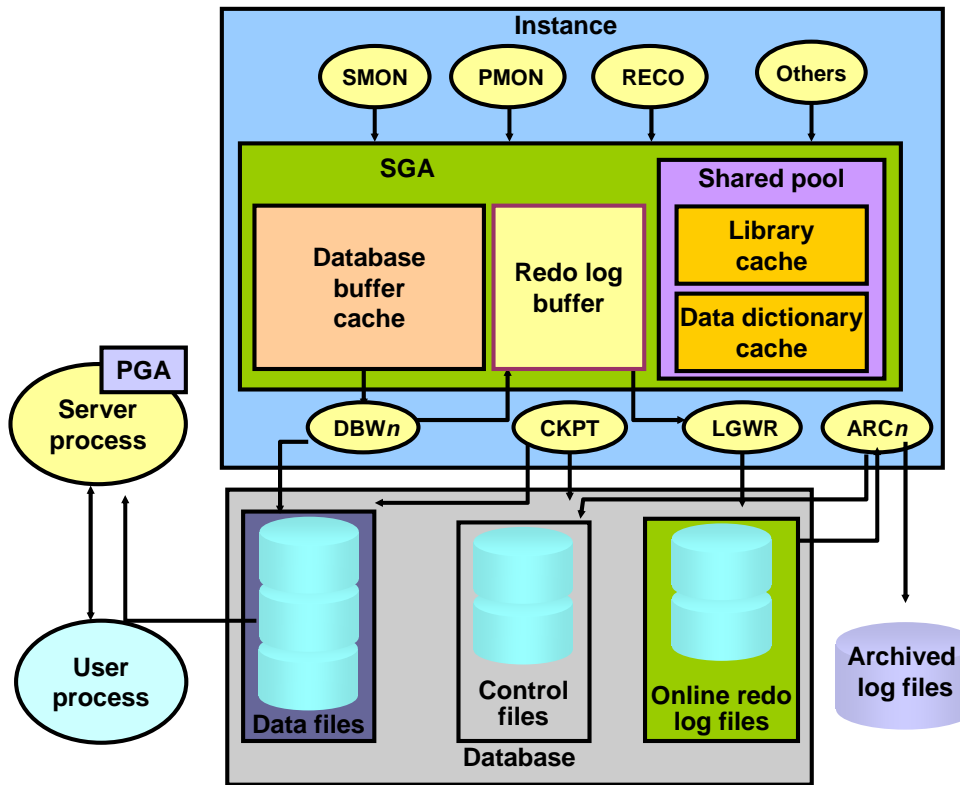
A traditional multitier architecture has the following components:

- A client or initiator process that starts an operation
- One or more application servers that perform parts of the operation. An application server contains a large part of the application logic, provides access to the data for the client, and performs some query processing, thus removing some of the load from the database server. The application server can serve as an interface between clients and multiple database servers and can provide an additional level of security.
- An end server or database server that stores most of the data used in the operation

This architecture enables use of an application server to do the following:

- Validate the credentials of a client (such as a Web browser)
- Connect to an Oracle Database server
- Perform the requested operation on behalf of the client

Oracle Database Architecture: Overview



1 - 6

Copyright © 2007, Oracle. All rights reserved.

ORACLE

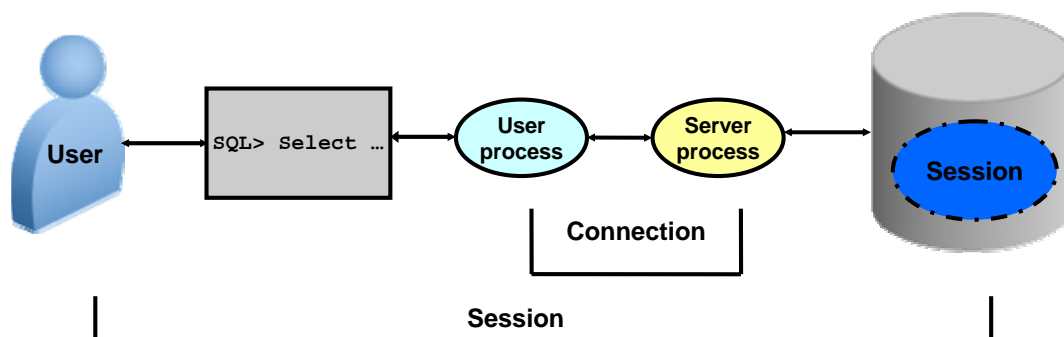
Oracle Database Architecture

An Oracle database consists of an instance and its associated databases. The instance consists of memory structures and background processes. Every time an instance is started, a shared memory area called the System Global Area (SGA) is allocated and the background processes are started.

The database consists of both physical structures and logical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting access to logical storage structures.

Connecting to the Database

- **Connection:** Communication between a user process and an instance
- **Session:** Specific connection of a user to an instance through a user process



Connecting to the Database

Connections and sessions are closely related to user processes but are very different in meaning.

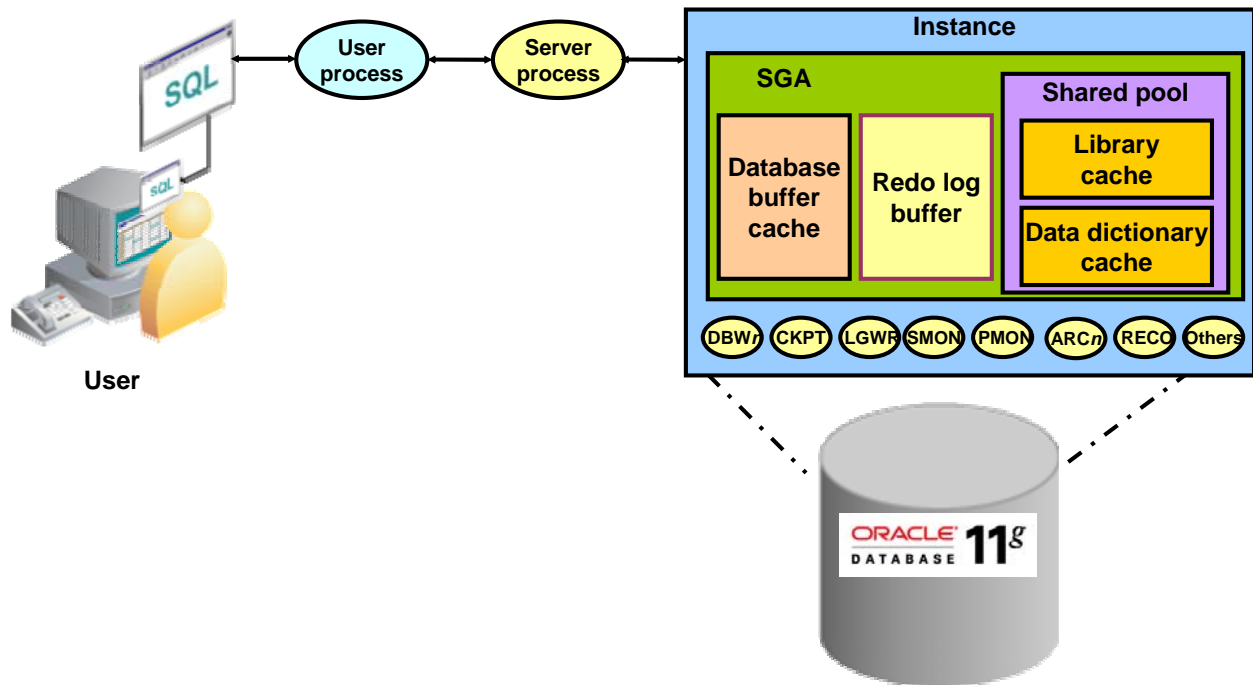
A *connection* is a communication pathway between a user process and an Oracle Database instance. A communication pathway is established using available interprocess communication mechanisms (on a computer that runs both the user process and Oracle Database) or network software (when different computers run the database application and Oracle Database, and communicate through a network).

A *session* represents the state of a current user login to the database instance. For example, when a user starts SQL*Plus, the user must provide a valid username and password, and then a session is established for that user. A session lasts from the time a user connects until the user disconnects or exits the database application.

In the case of a dedicated connection, the session is serviced by a permanent dedicated process. The session is serviced by an available server process selected from a pool, either by the middle tier or by Oracle shared server architecture.

Multiple sessions can be created and exist concurrently for a single Oracle database user using the same username. For example, a user with the username/password of HR/HR can connect to the same Oracle Database instance several times.

Interacting with an Oracle Database



Interacting with an Oracle Database

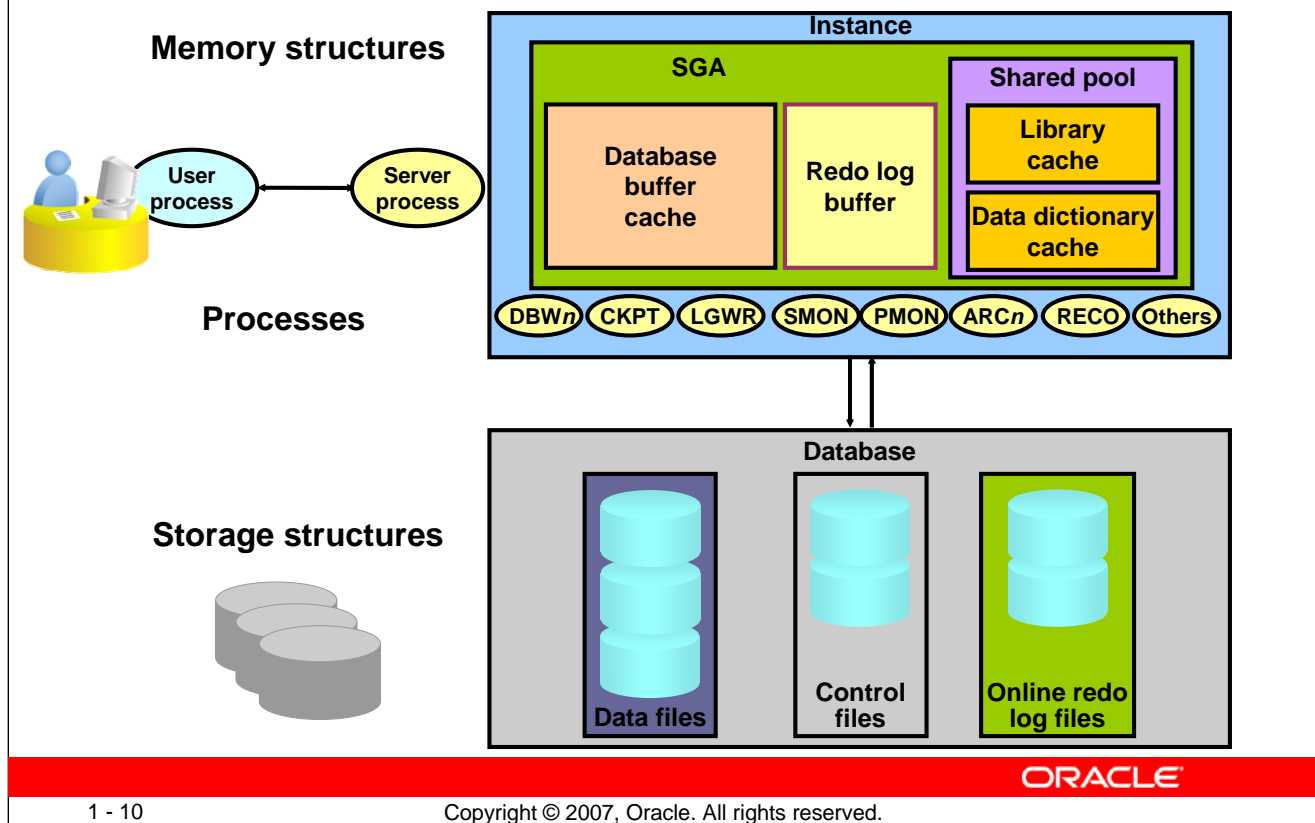
The following example describes Oracle database operations at the most basic level. It illustrates an Oracle database configuration in which the user and associated server process are on separate computers, connected through a network.

1. An instance has started on a node where Oracle Database is installed, often called the *host* or *database server*.
2. A user starts an application spawning a user process. The application attempts to establish a connection to the server. (The connection may be local, client/server, or a three-tier connection from a middle tier.)
3. The server runs a listener that has the appropriate Oracle Net Services handler. The server detects the connection request from the application and creates a dedicated server process on behalf of the user process.
4. The user runs a DML-type SQL statement and commits the transaction. For example, the user changes the address of a customer in a table and commits the change.
5. The server process receives the statement and checks the shared pool (an SGA component) for any shared SQL area that contains a similar SQL statement. If a shared SQL area is found, the server process checks the user's access privileges to the requested data, and the existing shared SQL area is used to process the statement. If a shared SQL area is not found, a new shared SQL area is allocated for the statement so that it can be parsed and processed.

Interacting with an Oracle Database (continued)

6. The server process retrieves any necessary data values, either from the actual data file (table) or from values stored in the SGA.
7. The server process modifies data in the SGA. Because the transaction is committed, the LogWriter process (LGWR) immediately records the transaction in the redo log file. The Database Writer process (DBW*n*) writes modified blocks permanently to disk when it is efficient to do so.
8. If the transaction is successful, the server process sends a message across the network to the application. If it is not successful, an error message is transmitted.
9. Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

Oracle Database Server Structures

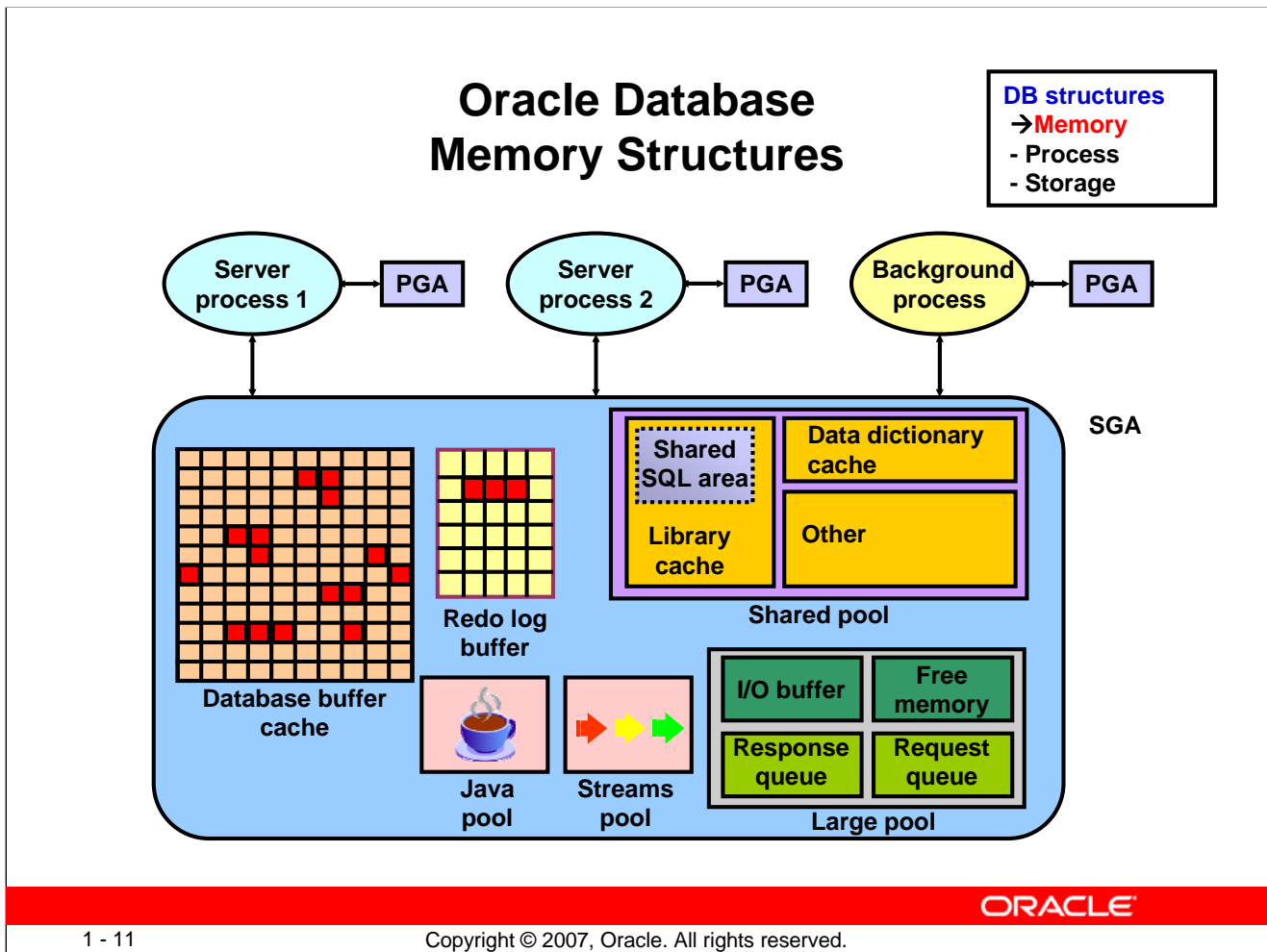


Oracle Database Server Structures

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 instance 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 Database Memory Structures

Oracle Database creates and uses memory structures for various purposes. For example, memory stores program code being run, data that is shared among users, and private data areas for each connected user.

Two basic memory structures are associated with an instance:

- **System Global Area (SGA):** Group of shared memory structures, known as SGA components, that contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.
- **Program Global Areas (PGA):** Memory regions that contain data and control information for a server or background process. A PGA is nonshared memory created by Oracle Database when a server or background process is started. Access to the PGA is exclusive to the server process. Each server process and background process has its own PGA.

Oracle Database Memory Structures (continued)

The SGA is the 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:** 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:** Used for all session-specific Java code and data in the Java Virtual Machine (JVM)
- **Streams pool:** Used by Oracle Streams to store information required by capture and apply

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

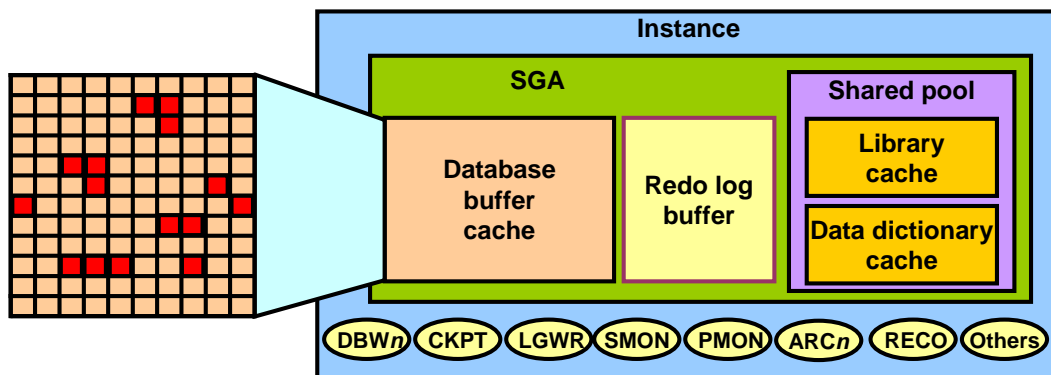
A Program Global Area (PGA) is a memory region that contains data and control information for each server process. An Oracle server process services a client's requests. Each server process has its own private PGA that is created when the server process is started. Access to the PGA is exclusive to that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

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

The Oracle database uses initialization parameters to create and manage memory structures. The simplest way to manage memory is to allow the database to automatically manage and tune it for you. To do so (on most platforms), you only have to set a target memory size initialization parameter (MEMORY_TARGET) and a maximum memory size initialization parameter (MEMORY_MAX_TARGET).

Database Buffer Cache

- Is part of the SGA
- Holds copies of data blocks that are read from data files
- Is shared by all concurrent users



ORACLE

1 - 13

Copyright © 2007, Oracle. All rights reserved.

Database Buffer Cache

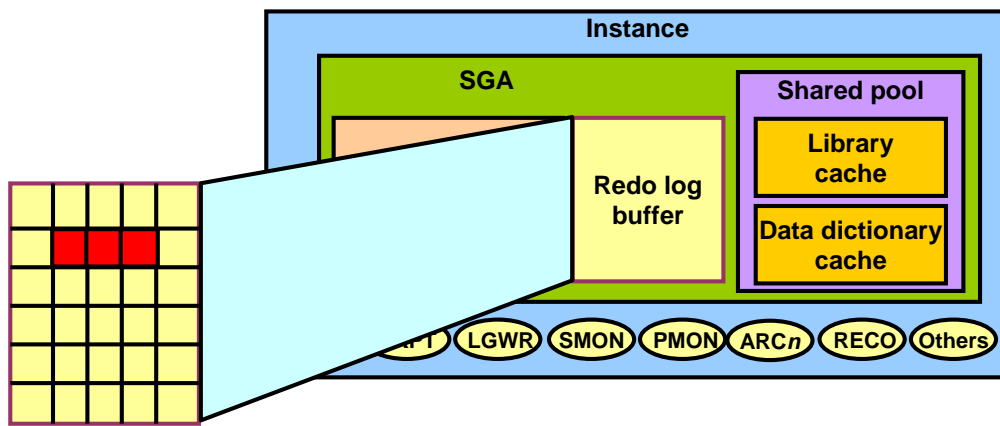
The database buffer cache is the portion of the SGA that holds copies of data blocks that are read from data files. All users who are concurrently connected to the instance share access to the database buffer cache.

The first time an Oracle Database user process requires a particular piece of data, it searches for the data in the database buffer cache. If the process finds the data already in the cache (a cache hit), it can read the data directly from memory. If the process cannot find the data in the cache (a cache miss), it must copy the data block from a data file on disk into a buffer in the cache before accessing the data. Accessing data through a cache hit is faster than data access through a cache miss.

The buffers in the cache are managed by a complex algorithm that uses a combination of least recently used (LRU) lists and touch count.

Redo Log Buffer

- Is a circular buffer in the SGA
- Holds information about changes made to the database
- Contains redo entries that have the information to redo changes made by operations such as DML and DDL



ORACLE

1 - 14

Copyright © 2007, Oracle. All rights reserved.

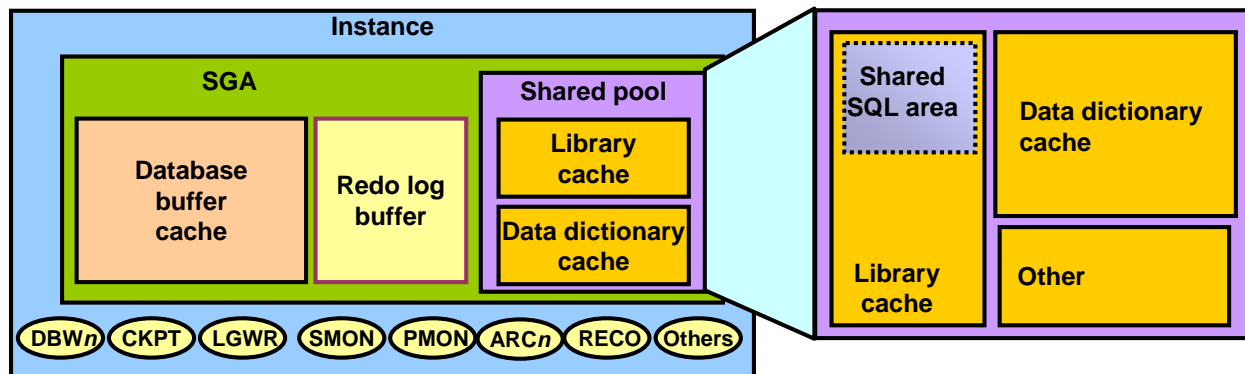
Redo Log Buffer

The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct (or redo) changes that are made to the database by DML, DDL, or internal operations. Redo entries are used for database recovery if necessary.

Redo entries are copied by Oracle Database processes from the user's memory space to the redo log buffer in the SGA. The redo entries take up continuous, sequential space in the buffer. The LGWR background process writes the redo log buffer to the active redo log file (or group of files) on disk.

Shared Pool

- Is a portion of the SGA
- Contains:
 - Library cache
 - Shared SQL area
 - Data dictionary cache
 - Control structures



ORACLE

1 - 15

Copyright © 2007, Oracle. All rights reserved.

Shared Pool

The shared pool portion of the SGA contains the library cache, the data dictionary cache, the SQL query result cache, the PL/SQL function result cache, buffers for parallel execution messages, and control structures.

The *data dictionary* is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database.

The data dictionary is accessed so often by Oracle Database that two special locations in memory are designated to hold dictionary data. One area is called the *data dictionary cache*, also known as the row cache because it holds data as rows instead of buffers (which hold entire blocks of data). The other area in memory to hold dictionary data is the *library cache*. All Oracle Database user processes share these two caches for access to data dictionary information.

Oracle Database represents each SQL statement that it runs with a shared SQL area (as well as a private SQL area kept in the PGA). Oracle Database recognizes when two users are executing the same SQL statement and reuses the shared SQL area for those users.

Shared Pool (continued)

A *shared SQL area* contains the parse tree and execution plan for a given SQL statement. Oracle Database saves memory by using one shared SQL area for SQL statements run multiple times, which often happens when many users run the same application.

When a new SQL statement is parsed, Oracle Database allocates memory from the shared pool to store in the shared SQL area. The size of this memory depends on the complexity of the statement.

Oracle Database processes PL/SQL program units (procedures, functions, packages, anonymous blocks, and database triggers) in much the same way it processes individual SQL statements. Oracle Database allocates a shared area to hold the parsed, compiled form of a program unit. Oracle Database allocates a private area to hold values specific to the session that runs the program unit, including local, global, and package variables (also known as *package instantiation*) and buffers for executing SQL. If more than one user runs the same program unit, then a single, shared area is used by all users, while all users maintain separate copies of their own private SQL areas, holding values specific to their own sessions.

Individual SQL statements contained in a PL/SQL program unit are processed just like other SQL statements. Despite their origins in a PL/SQL program unit, these SQL statements use a shared area to hold their parsed representations and a private area for each session that runs the statement.

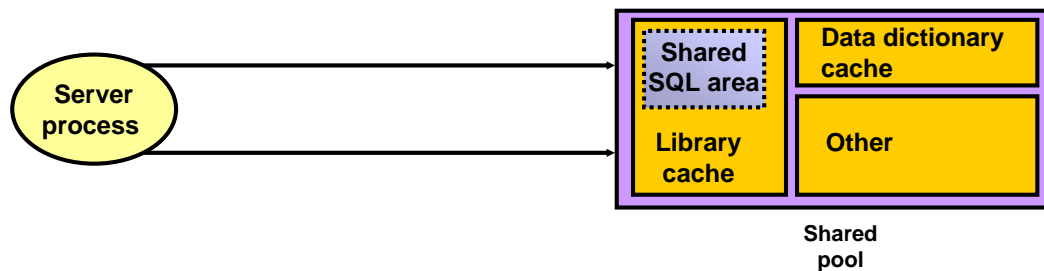
The SQL query result cache and PL/SQL function result cache are new to Oracle Database 11g. They share the same infrastructure, appear in the same dynamic performance (\$V) views, and are administered using the same supplied package.

Results of queries and query fragments can be cached in memory in the *SQL query result cache*. The database can then use cached results to answer future executions of these queries and query fragments. Because retrieving results from the SQL query result cache is faster than rerunning a query, frequently run queries experience a significant performance improvement when their results are cached.

A PL/SQL function is sometimes used to return the result of a computation whose inputs are one or several parameterized queries issued by the function. In some cases, these queries access data that changes very infrequently compared to the frequency of calling the function. You can include syntax in the source text of a PL/SQL function to request that its results be cached in the *PL/SQL function result cache* and (to ensure correctness) that the cache be purged when tables in a list of tables experience DML.

Allocation and Reuse of Memory in the Shared Pool

- Server process checks the shared pool to see if a shared SQL area already exists for an identical statement.
- Server process allocates a private SQL area on behalf of the session.



Allocation and Reuse of Memory in the Shared Pool

In general, any item (shared SQL area or dictionary row) in the shared pool remains until it is flushed according to a modified LRU (least recently used) algorithm. The memory for items that are not being used regularly is freed if space is required for new items that must be given some space in the shared pool. A modified LRU algorithm allows shared pool items that are used by many sessions to remain in memory as long as they are useful, even if the process that originally created the item terminates. As a result, the overhead and processing of SQL statements associated with a multiuser Oracle Database system are minimized.

When a SQL statement is submitted to Oracle Database for execution, the following memory allocation steps are automatically performed:

1. Oracle Database checks the shared pool to see if a shared SQL area already exists for an identical statement. If so, that shared SQL area is used for the execution of the subsequent new instances of the statement. If there is no shared SQL area for a statement, Oracle Database allocates a new shared SQL area in the shared pool. In either case, the user's private SQL area is associated with the shared SQL area that contains the statement.

Allocation and Reuse of Memory in the Shared Pool (continued)

2. Oracle Database allocates a private SQL area on behalf of the session. The location of the private SQL area depends on the type of connection established for the session.

Note: A shared SQL area can be flushed from the shared pool even if the shared SQL area corresponds to an open cursor that has not been used for some time. If the open cursor is subsequently used to run its statement, Oracle Database reparses the statement and a new shared SQL area is allocated in the shared pool.

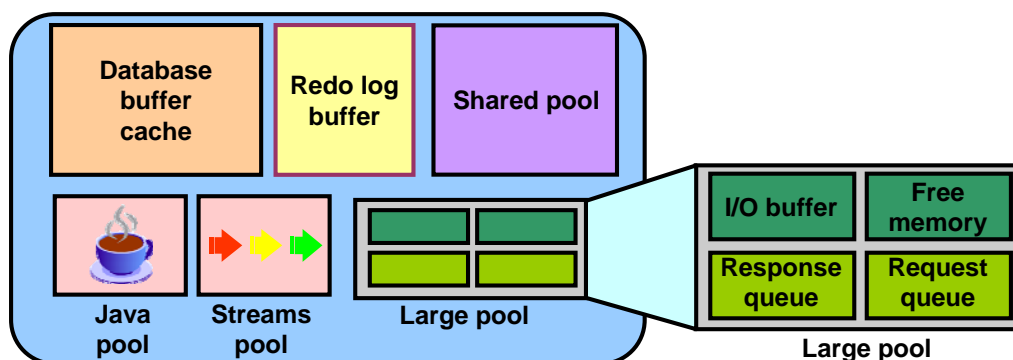
Oracle Database also flushes a shared SQL area from the shared pool in these circumstances:

- When the `DBMS_STATS` package is used to update or delete the statistics of a table, cluster, or index, all shared SQL areas that contain statements referencing the analyzed schema object are flushed from the shared pool. The next time a flushed statement is run, the statement is parsed in a new shared SQL area to reflect the new statistics for the schema object.
- If a schema object is referenced in a SQL statement and that object is later modified in any way, the shared SQL area is invalidated (marked invalid) and the statement must be reparsed the next time it is run.
- If you change a database's global database name, all information is flushed from the shared pool.
- The administrator can manually flush all information in the shared pool to assess the performance (with respect to the shared pool, not the data buffer cache) that can be expected after instance startup without shutting down the current instance. The `ALTER SYSTEM FLUSH SHARED_POOL` statement is used to do this.

Large Pool

Provides large memory allocations for:

- Session memory for the shared server and the Oracle XA interface
- I/O server processes
- Oracle Database backup and restore operations



Large Pool

The database administrator can configure an optional memory area called the *large pool* to provide large memory allocations for:

Session memory for the shared server and the Oracle XA interface (used where transactions interact with more than one database):

- I/O server processes
- Oracle Database backup and restore operations

By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle Database can use the shared pool primarily for caching shared SQL and avoid the performance overhead that is caused by shrinking the shared SQL cache.

In addition, the memory for Oracle Database backup and restore operations, for I/O server processes, and for parallel buffers is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such large memory requests than the shared pool.

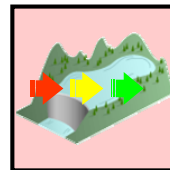
The large pool does not have an LRU list. It is different from reserved space in the shared pool, which uses the same LRU list as other memory allocated from the shared pool.

Java Pool and Streams Pool

- Java pool memory is used in server memory for all session-specific Java code and data in the JVM.
- Streams pool memory is used exclusively by Oracle Streams to:
 - Store buffered queue messages
 - Provide memory for Oracle Streams processes



Java pool



Streams pool

ORACLE

1 - 20

Copyright © 2007, Oracle. All rights reserved.

Java Pool and Streams Pool

Java pool memory is used in server memory for all session-specific Java code and data in the JVM. Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

The Java Pool Advisor statistics provide information about library cache memory used for Java and predict how changes in the size of the Java pool can affect the parse rate. The Java Pool Advisor is internally turned on when `statistics_level` is set to `TYPICAL` or higher. These statistics reset when the advisor is turned off.

The Streams pool is used exclusively by Oracle Streams. The Streams pool stores buffered queue messages, and it provides memory for Oracle Streams capture processes and apply processes.

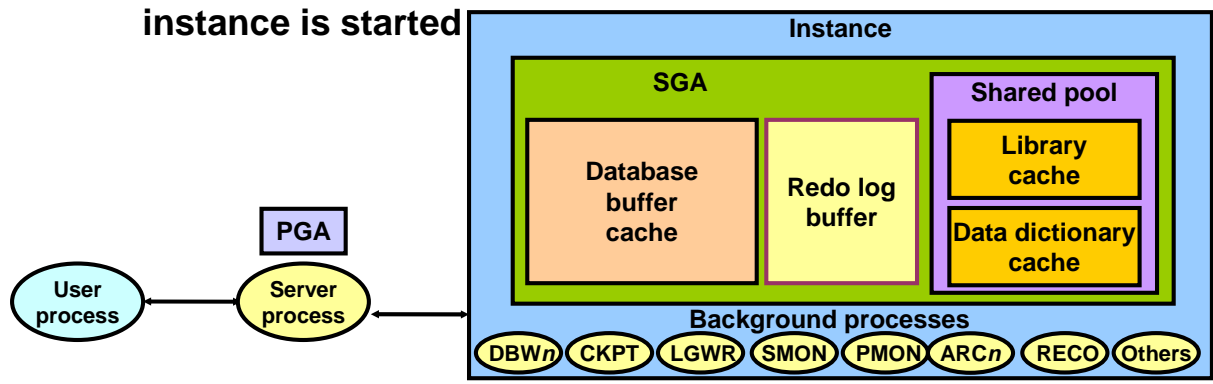
Unless you specifically configure it, the size of the Streams pool starts at zero. The pool size grows dynamically as needed when Oracle Streams is used.

Note: A detailed discussion of Java programming and Oracle Streams is beyond the scope of this class

Process Architecture

DB structures
 - Memory
 → **Process**
 - Storage

- **User process**
 - Is started when a database user or a batch process connects to Oracle Database
- **Database processes**
 - **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



Process Architecture

The processes in an Oracle Database system can be divided into two major groups:

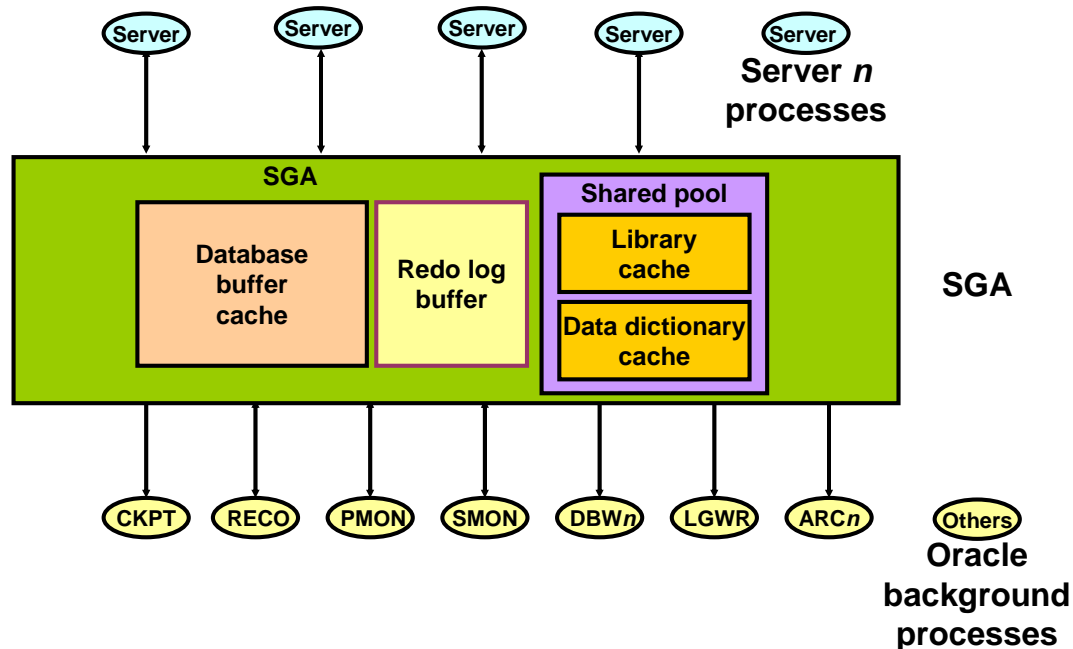
- User processes that run the application or Oracle tool code
- Oracle Database processes that run the Oracle database server code (including server processes and background processes)

When a user runs an application program or an Oracle tool such as SQL*Plus, Oracle Database creates a *user process* to run the user's application. Oracle Database also creates a *server process* to execute the commands issued by the user process. In addition, the Oracle server also has 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.

The process structure varies for different Oracle Database configurations, depending on the operating system and the choice of Oracle Database options. The code for connected users can be configured as a dedicated server or a shared server.

- **Dedicated server:** For each user, the database application is run by a user process that is served by a dedicated server process that executes Oracle database server code.
- **Shared server:** Eliminates the need for a dedicated server process for each connection. A dispatcher directs multiple incoming network session requests to a pool of shared server processes. A shared server process serves any client request.

Process Structures



ORACLE

1 - 22

Copyright © 2007, Oracle. All rights reserved.

Process Structures

Server Processes

Oracle Database creates server processes to handle the requests of user processes connected to the instance. In some situations, when the application and Oracle Database operate on the same computer, it is possible to combine the user process and corresponding server process into a single process to reduce system overhead. However, when the application and Oracle Database operate on different computers, a user process always communicates with Oracle Database through a separate server process.

Server processes created on behalf of each user's application can perform one or more of the following:

- Parse and run SQL statements issued through the application
- Read necessary data blocks from data files on disk into the shared database buffers of the SGA (if the blocks are not already present in the SGA)
- Return results in such a way that the application can process the information

Background Processes

To maximize performance and accommodate many users, a multiprocess Oracle Database system uses some additional Oracle Database processes called *background processes*. An Oracle Database instance can have many background processes.

Process Structures (continued)

The background processes commonly seen in non-RAC, non-ASM environments can include the following:

- Database writer process (DBW*n*)
- Log writer process (LGWR)
- Checkpoint process (CKPT)
- System Monitor process (SMON)
- Process monitor process (PMON)
- Recoverer process (RECO)
- Job queue processes
- Archiver processes (ARC*n*)
- Queue monitor processes (QMN*n*)

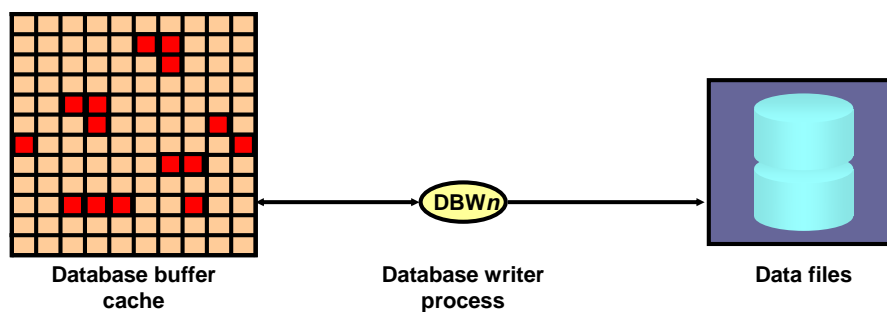
Other background processes may be found in more advanced configurations such as RAC. See the V\$BGPROCESS view for more information on the background processes.

On many operating systems, background processes are created automatically when an instance is started.

Database Writer Process (DBWn)

Writes modified (dirty) buffers in the database buffer cache to disk:

- **Asynchronously while performing other processing**
- **Periodically to advance the checkpoint**



Database Writer Process (DBWn)

The Database Writer process (DBWn) writes the contents of buffers to data files. The DBWn processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk. Although one Database Writer process (DBW0) is adequate for most systems, you can configure additional processes (DBW1 through DBW9 and DBWa through DBWj) to improve write performance if your system modifies data heavily. These additional DBWn processes are not useful on uniprocessor systems.

When a buffer in the database buffer cache is modified, it is marked dirty and is added to the LRUW (LRU write) list of dirty buffers that is kept in SCN order. This order therefore matches the order of redo that is written to the redo logs for these changed buffers. When the number of available buffers in the buffer cache falls below an internal threshold (to the extent that server processes find it difficult to obtain available buffers), DBWn writes dirty buffers to the data files in the order that they were modified by following the order of the LRUW list.

Database Writer Process (DBWn) (continued)

The SGA contains a memory structure that has the redo byte address (RBA) of the position in the redo stream where recovery should begin in the case of an instance failure. This structure acts as a pointer into the redo and is written to the control file by the CKPT process once every three seconds. Because the DBWn writes dirty buffers in SCN order, and because the redo is in SCN order, every time DBWn writes dirty buffers from the LRUW list, it also advances the pointer held in the SGA memory structure so that instance recovery (if required) begins reading the redo from approximately the correct location and avoids unnecessary I/O. This is known as *incremental checkpointing*.

Note: There are other cases when DBWn may write (for example, when tablespaces are made read-only or are placed offline). In such cases, no incremental checkpoint occurs because dirty buffers belonging only to the corresponding data files are written to the database unrelated to the SCN order.

The LRU algorithm keeps more frequently accessed blocks in the buffer cache so that, when a buffer is written to disk, it is unlikely to contain data that will soon be useful.

The DB_WRITER_PROCESSES initialization parameter specifies the number of DBWn processes. The maximum number of DBWn processes is 20. If it is not specified by the user during startup, Oracle Database determines how to set DB_WRITER_PROCESSES based on the number of CPUs and processor groups.

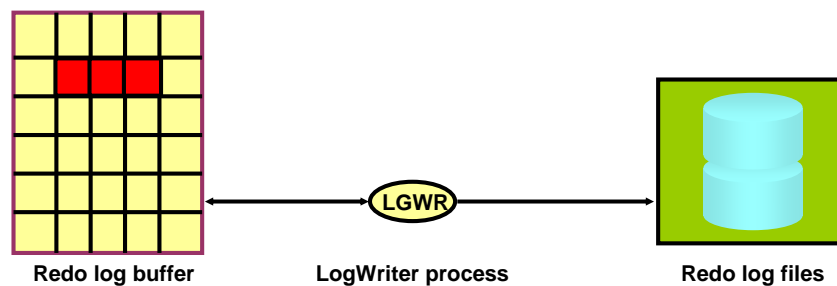
The DBWn process writes dirty buffers to disk under the following conditions:

- When a server process cannot find a clean reusable buffer after scanning a threshold number of buffers, it signals DBWn to write. DBWn writes dirty buffers to disk asynchronously while performing other processing.
- DBWn periodically writes buffers to advance the checkpoint, which is the position in the redo thread (log) from which instance recovery begins. This log position is determined by the oldest dirty buffer in the buffer cache.

In all cases, DBWn performs batched (multiblock) writes to improve efficiency. The number of blocks written in a multiblock write varies by operating system.

LogWriter Process (LGWR)

- **Writes the redo log buffer to a redo log file on disk**
- **Writes:**
 - **When a user process commits a transaction**
 - **When the redo log buffer is one-third full**
 - **Before a DBWn process writes modified buffers to disk**



ORACLE

1 - 26

Copyright © 2007, Oracle. All rights reserved.

LogWriter Process (LGWR)

The LogWriter process (LGWR) is responsible for redo log buffer management by writing the redo log buffer entries to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote.

The redo log buffer is a circular buffer. When LGWR writes redo entries from the redo log buffer to a redo log file, server processes can then copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy. LGWR writes one contiguous portion of the buffer to disk.

LGWR writes:

- When a user pLogWriter process commits a transaction
- When the redo log buffer is one-third full
- Before a DBWn process writes modified buffers to disk (if necessary)

Log Writer Process (LGWR) (continued)

Before DBW_n can write a modified buffer, all redo records that are associated with the changes to the buffer must be written to disk (the write-ahead protocol). If DBW_n finds that some redo records have not been written, it signals LGWR to write the redo records to disk and waits for LGWR to complete writing the redo log buffer before it can write out the data buffers. LGWR writes to the current log group. If one of the files in the group is damaged or unavailable, LGWR continues writing to other files in the group and logs an error in the LGWR trace file and in the system alert log. If all files in a group are damaged, or if the group is unavailable because it has not been archived, LGWR cannot continue to function.

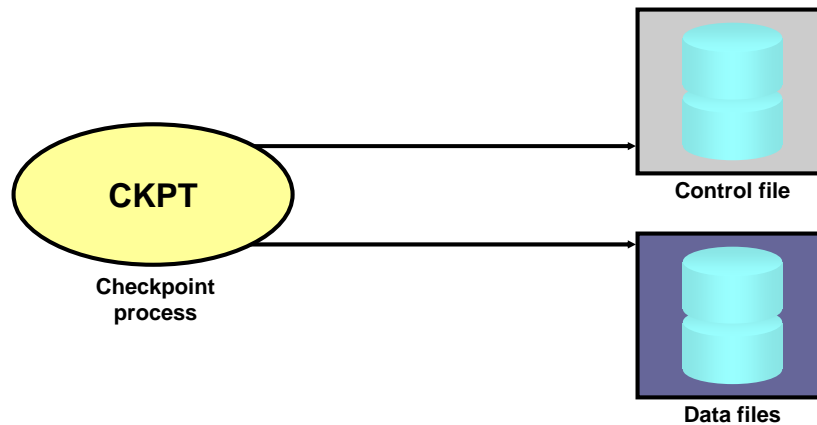
When a user issues a COMMIT statement, LGWR puts a commit record in the redo log buffer and writes it to disk immediately, along with the transaction's redo entries. The corresponding changes to data blocks are deferred until it is more efficient to write them. This is called a *fast commit mechanism*. The atomic write of the redo entry containing the transaction's commit record is the single event that determines whether the transaction has committed. Oracle Database returns a success code to the committing transaction, although the data buffers have not yet been written to disk.

If more buffer space is needed, LGWR sometimes writes redo log entries before a transaction is committed. These entries become permanent only if the transaction is later committed. When a user commits a transaction, the transaction is assigned a system change number (SCN), which Oracle Database records along with the transaction's redo entries in the redo log. SCNs are recorded in the redo log so that recovery operations can be synchronized in Real Application Clusters and distributed databases.

In times of high activity, LGWR can write to the redo log file by using group commits. For example, suppose that a user commits a transaction. LGWR must write the transaction's redo entries to disk. As this happens, other users issue COMMIT statements. However, LGWR cannot write to the redo log file to commit these transactions until it has completed its previous write operation. After the first transaction's entries are written to the redo log file, the entire list of redo entries of waiting transactions (not yet committed) can be written to disk in one operation, requiring less I/O than do transaction entries handled individually. Therefore, Oracle Database minimizes disk I/O and maximizes performance of LGWR. If requests to commit continue at a high rate, every write (by LGWR) from the redo log buffer can contain multiple commit records.

Checkpoint Process (CKPT)

- Records checkpoint information in
 - Control file
 - Each data file header



Checkpoint Process (CKPT)

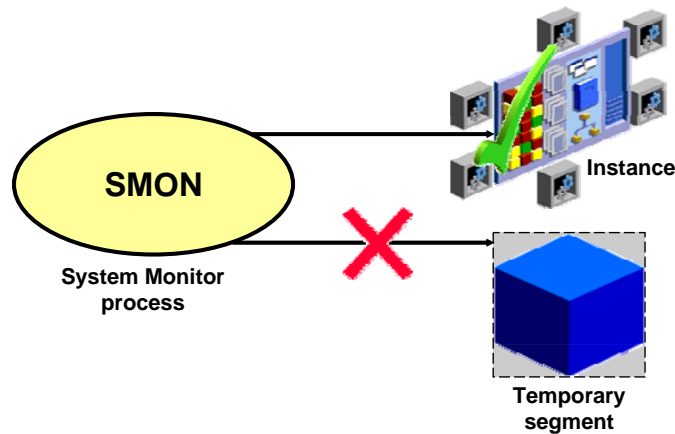
A *checkpoint* is a data structure that defines a system change number (SCN) in the redo thread of a database. Checkpoints are recorded in the control file and in each data file header. They are a crucial element of recovery.

When a checkpoint occurs, Oracle Database must update the headers of all data files to record the details of the checkpoint. This is done by the CKPT process. The CKPT process does not write blocks to disk; DBWn always performs that work. The SCNs recorded in the file headers guarantee that all changes made to database blocks prior to that SCN have been written to disk.

The statistic DBWR checkpoints displayed by the `SYSTEM_STATISTICS` monitor in Oracle Enterprise Manager indicate the number of checkpoint requests that have completed.

System Monitor Process (SMON)

- Performs recovery at instance startup
- Cleans up unused temporary segments



ORACLE

1 - 29

Copyright © 2007, Oracle. All rights reserved.

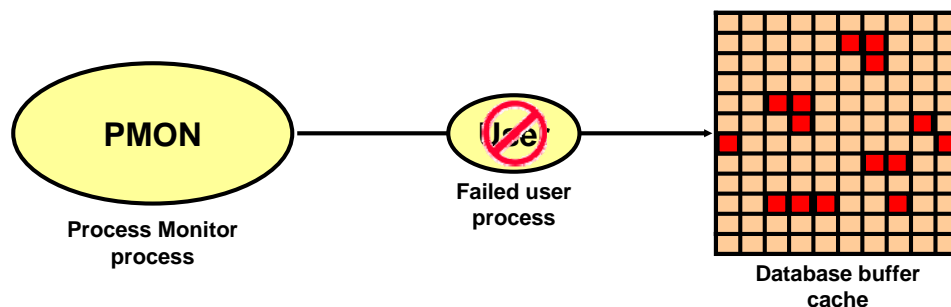
System Monitor Process (SMON)

The System Monitor process (SMON) performs recovery at instance startup if necessary. SMON is also responsible for cleaning up temporary segments that are no longer in use. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online.

SMON checks regularly to see whether the process is needed. Other processes can call SMON if they detect a need for it.

Process Monitor Process (PMON)

- **Performs process recovery when a user process fails**
 - Cleans up the database buffer cache
 - Frees resources that are used by the user process
- **Monitors sessions for idle session timeout**
- **Dynamically registers database services with listeners**



Process Monitor Process (PMON)

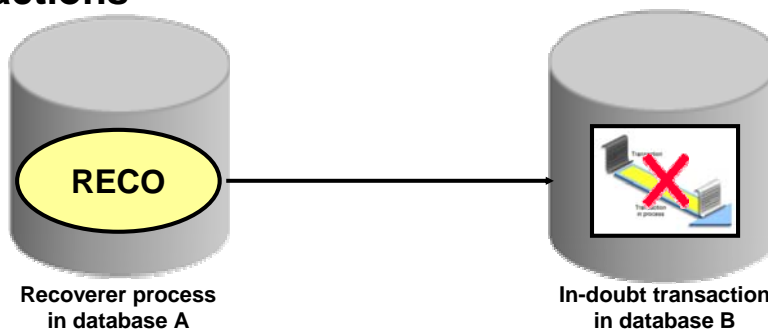
The Process Monitor process (PMON) performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using. For example, it resets the status of the active transaction table, releases locks, and removes the process ID from the list of active processes.

PMON periodically checks the status of dispatcher and server processes, and restarts any that have stopped running (but not any that Oracle Database has terminated intentionally). PMON also registers information about the instance and dispatcher processes with the network listener.

Like SMON, PMON checks regularly to see whether it is needed; it can be called if another process detects the need for it.

Recoverer Process

- Used with the distributed database configuration
- Automatically connects to other databases involved in in-doubt distributed transactions
- Automatically resolves all in-doubt transactions
- Removes any rows that correspond to in-doubt transactions



ORACLE

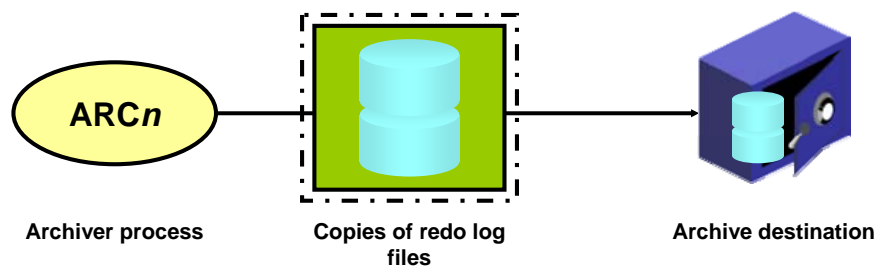
Recoverer Process (RECO)

The Recoverer process (RECO) is a background process that is used with the distributed database configuration that automatically resolves failures involving distributed transactions. The RECO process of an instance automatically connects to other databases involved in an in-doubt distributed transaction. When the RECO process reestablishes a connection between involved database servers, it automatically resolves all in-doubt transactions, removing from each database's pending transaction table any rows that correspond to the resolved in-doubt transactions.

If the RECO process fails to connect with a remote server, RECO automatically tries to connect again after a timed interval. However, RECO waits an increasing amount of time (growing exponentially) before it attempts another connection.

Archiver Processes (ARCn)

- Copy redo log files to a designated storage device after a log switch has occurred
- Can collect transaction redo data and transmit that data to standby destinations



Archiver Processes (ARCn)

The archiver processes (ARCn) copy redo log files to a designated storage device after a log switch has occurred. ARCn processes are present only when the database is in ARCHIVELOG mode and automatic archiving is enabled.

If you anticipate a heavy workload for archiving (such as during bulk loading of data), you can increase the maximum number of archiver processes with the LOG_ARCHIVE_MAX_PROCESSES initialization parameter. The ALTER SYSTEM statement can change the value of this parameter dynamically to increase or decrease the number of ARCn processes.

Other Processes

- **MMON: Performs manageability-related background tasks**
- **MMNL: Performs frequent and lightweight manageability-related tasks**
- **MMAN: Performs automatic memory management tasks**
- **CJQ0: Runs user jobs used in batch processing**
- **QMNx: Monitors the Streams Advanced Queuing message queues**

ORACLE

1 - 33

Copyright © 2007, Oracle. All rights reserved.

Other Processes

There are several other background processes that might be running. These can include the following:

The Manageability Monitor process (MMON) performs various manageability-related background tasks, for example:

- Issuing alerts whenever a given metrics violates its threshold value
- Taking snapshots by spawning additional process (MMON slaves)
- Capturing statistics value for SQL objects that have been recently modified

The Lightweight Manageability Monitor process (MMNL) performs frequent tasks related to lightweight manageability, such as session history capture and metrics computation.

The Memory Manager process (MMAN) is used for internal database tasks. It manages automatic memory management processing to help allocate memory where it is needed dynamically in an effort to avoid out-of-memory conditions or poor buffer cache performance.

Other Processes (continued)

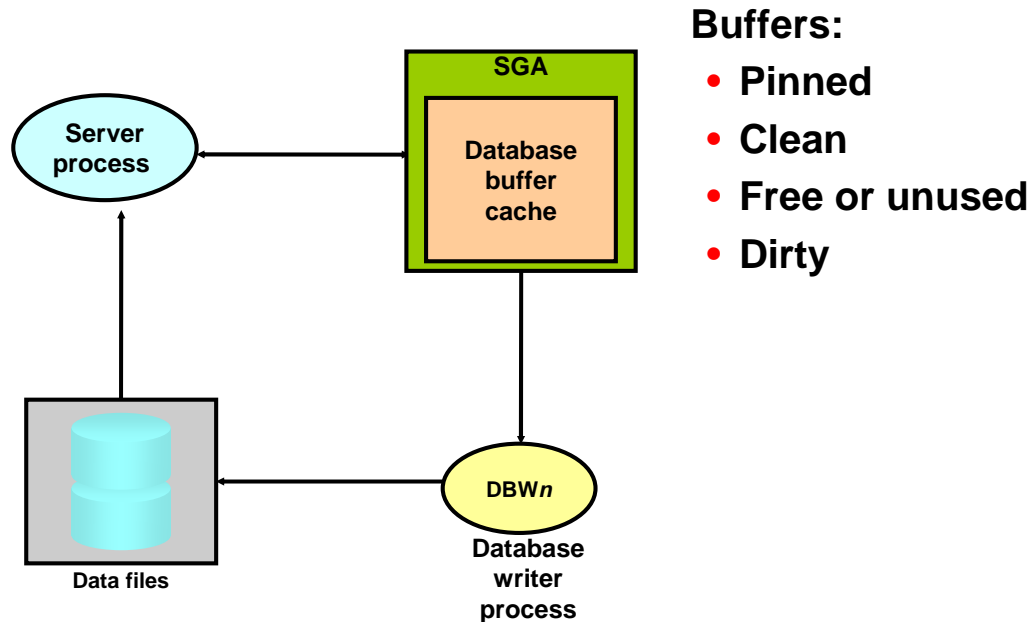
The Rebalance process (RBAL) coordinates rebalance activity for disk groups in an Automatic Storage Management instance. It performs a global open on Automatic Storage Management disks. ORBn performs the actual rebalance data extent movements in an Automatic Storage Management instance. There can be many of these at a time, named ORB0, ORB1, and so on.

The Automatic Storage Management process (ASMB) is present in a database instance using an Automatic Storage Management disk group. It communicates with the Automatic Storage Management instance.

Job queue processes are used for batch processing. They run user jobs and can be viewed as scheduler services that are used to schedule jobs as PL/SQL statements or procedures on an Oracle Database instance. The coordinator process, named CJQ0, periodically selects jobs that need to be run from the system `JOB$` table. The CJQ0 process dynamically spawns job queue slave processes (J000 through J999) to run the jobs. The job queue process runs one of the jobs that was selected by the CJQ0 process for execution. The processes run one job at a time.

The Queue Monitor process (QMNX) is an optional background process for Oracle Streams Advanced Queuing, which monitors the message queues. You can configure up to 10 queue monitor processes.

Server Process and Database Buffer Cache



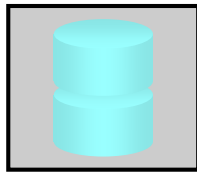
Server Process and Database Buffer Cache

When a query is processed, the Oracle server process looks in the database buffer cache for images of any blocks that it needs. If the block image 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. Buffers in the buffer cache can be in one of the following four states:

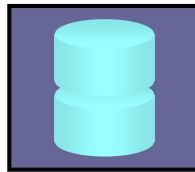
- **Pinned:** Multiple sessions are kept from writing to the same block at the same time. Other sessions wait to access the block.
- **Clean:** The buffer is now unpinned and is a candidate for immediate aging out, if the current contents (data block) are not referenced again. Either the contents are 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 DBWn before it can be aged out.

Database Storage Architecture

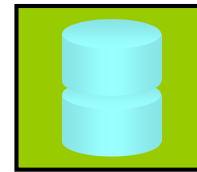
DB structures
- Memory
- Process
→ Storage



Control files



Data files



Online redo log files



Parameter file



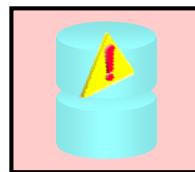
Backup files



Archived redo log files



Password file



Alert log and trace files

ORACLE

1 - 36

Copyright © 2007, Oracle. All rights reserved.

Database Storage Architecture

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 in the database.
- **Data files:** Contain the user or application data of the database, as well as metadata and the data dictionary
- **Online redo log files:** Allow for instance recovery of the database. If the database server crashes and does not lose any data files, 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:

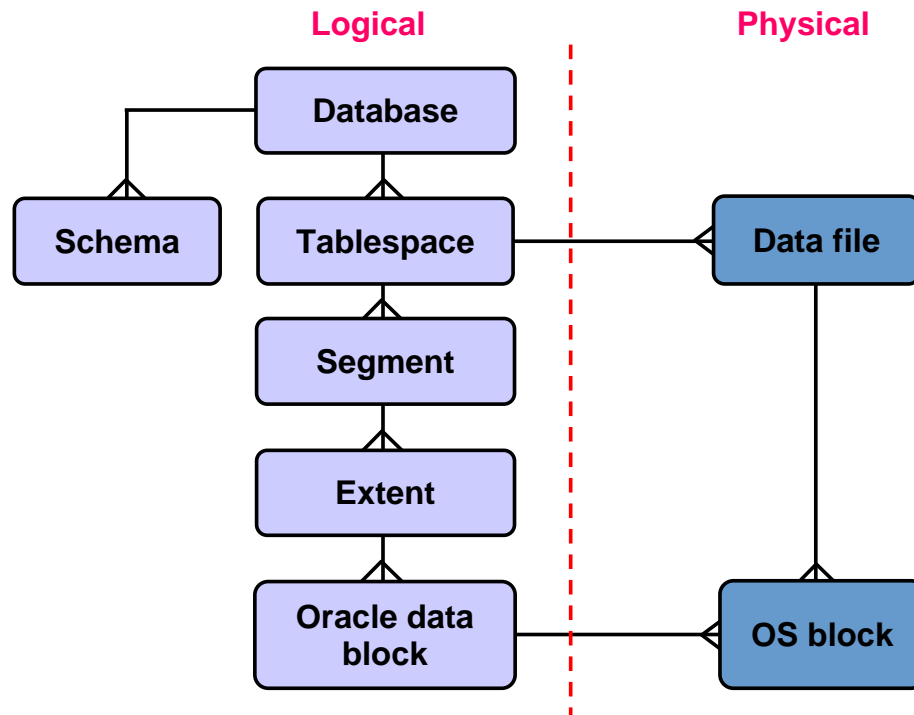
- **Parameter file:** Is used to define how the instance is configured when it starts up
- **Password file:** Allows `sysdba`, `sysoper`, and `sysasm` to connect remotely to the database and perform administrative tasks
- **Backup files:** Are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.
- **Archived redo 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.

Database Storage Architecture (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 file:** These are special trace entries. The alert log of a database is a chronological log of messages and errors. Each instance has one alert log file. Oracle recommends that you review this periodically.

Note: Parameter, password, alert, and trace files are covered in other lessons.

Logical and Physical Database Structures



1 - 38

Copyright © 2007, Oracle. All rights reserved.

ORACLE

Logical and Physical Database Structures

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, 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 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

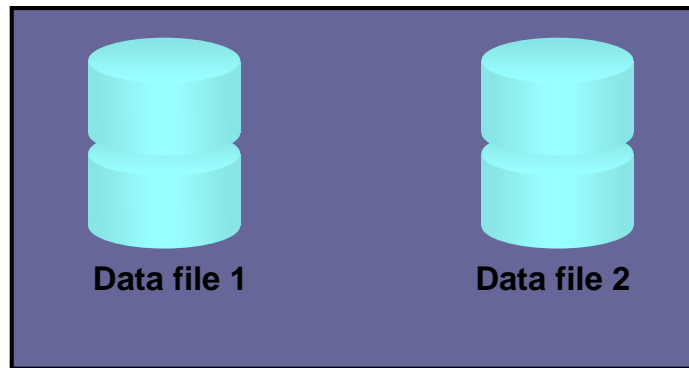
The level of logical database storage above an extent is called a *segment*. A segment is a set of extents that are allocated for a certain logical structure. Different types of segments include:

- **Data segments:** Each nonclustered, non-index-organized table has a data segment, with the exception of external tables, global temporary tables, and partitioned tables in which each table has one or more segments. 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 for each database instance. This tablespace contains numerous undo segments 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 either a default temporary tablespace for every user, or a default temporary tablespace that is used database-wide.

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.

Tablespaces and Data Files

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



USERS tablespace

ORACLE

1 - 40

Copyright © 2007, Oracle. All rights reserved.

Tablespaces and Data Files

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

Note: You can also create bigfile tablespaces, which have only one file that is often very large. The file may be any size up to the maximum that the row ID architecture permits. The maximum size is the block size for the tablespace multiplied by 2^{36} , or 128 TB for a 32 KB block size. Traditional smallfile tablespaces (which are the default) usually contain multiple data files, but the files cannot be as large. For more information about bigfile tablespaces, see the *Oracle Database Administrator's Guide*.

SYSTEM and SYSAUX Tablespaces

- The **SYSTEM** and **SYSAUX** tablespaces are mandatory tablespaces that 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).

ORACLE

1 - 41

Copyright © 2007, Oracle. All rights reserved.

SYSTEM and SYSAUX Tablespaces

Each Oracle database must contain a **SYSTEM** tablespace and a **SYSAUX** tablespace, which 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 (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.

Note: The **SYSAUX** tablespace may be offlined to do tablespace recovery, whereas this is not possible for the **SYSTEM** tablespace. Neither of them may be made read-only.

Segments, Extents, and Blocks

- **Segments exist in a tablespace.**
- **Segments are collections of extents.**
- **Extents are collections of data blocks.**
- **Data blocks are mapped to disk blocks.**



Segment



Extents



**Data
blocks**



**Disk
blocks**

ORACLE

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 do not need to 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 database creation. 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, a larger block size may be beneficial.

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

You can have tablespaces with a nonstandard block size. For details, see the *Oracle Database Administrator's Guide*.

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:** Data files, control files, and redo log files

ORACLE

1 - 43

Copyright © 2007, Oracle. All rights reserved.

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. The details are covered in the following lessons.

Summary

In this lesson, you should have learned how to:

- **List the major architectural components of Oracle Database**
- **Explain the memory structures**
- **Describe the background processes**
- **Correlate the logical and physical storage structures**

ORACLE

Practice 1: Overview

This is a paper practice with questions about:

- Database architecture
- Memory
- Processes
- File structures

ORACLE

2

Preparing the Database Environment

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- 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)



Tasks of an Oracle Database Administrator

The 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

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 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:

- Understanding how the installation fits into the overall technical architecture of an organization
- Reviewing (and updating) capacity plans
- Choosing the database software (required version and options)
- Ensuring that system requirements are met for all chosen elements

Tools for Administering an Oracle Database

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

Tools for Administering an Oracle Database

You can use the following tools for installation and upgrade:

- **Oracle Universal Installer (OUI):** Installs your Oracle software and options; can automatically launch the Database Configuration Assistant to create a database
- **Database Configuration Assistant (DBCA):** Creates a database from Oracle-supplied templates, enabling you to copy a preconfigured seed database (Alternatively, you can create your own database and templates.)
- **Database Upgrade Assistant (DBUA):** Guides you through the upgrade of your existing database to a new Oracle release
- **Oracle Net Manager:** Configures network connectivity for your Oracle databases and applications

Tools for Administering an Oracle Database (continued)

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

- **Oracle Enterprise Manager (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 EM as the single interface for managing your database. In addition to providing a Web-based user interface for executing SQL commands, it interfaces with other Oracle components that are used to administer your database (for example, Recovery Manager and Scheduler).

The main EM tools that are used to administer an Oracle database are:

- **Enterprise Manager Database Console:** To administer one database
- **Enterprise Manager Grid Control:** To administer many databases at the same time
- **SQL*Plus:** Standard command-line interface for managing your database
- **Recovery Manager (RMAN):** 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:** 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:** Enables the loading of data from an external file into an Oracle database; one of several Oracle utilities that you can use to load data into database tables
- **Command-line tools:**
 - To administer Enterprise Manager:
`emctl start | status | set | stop dbconsole`
 - To administer the listener:
`lsnrctl help | start | status | stop dbconsole`

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)**



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 11g 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 minimal requirements across all platforms. 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

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

```
[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, UnitedLin
ux-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]$
```

Checking the System Requirements

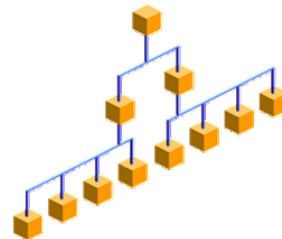
The Oracle Database 11g installation automates most of the prerequisite checks to verify the following:

- Minimum temporary space requirements for installation and configuration are checked. 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).
- Oracle Database 11g is certified against several versions of the Linux platform.
- All required OS patches are installed.
- All required system and kernel parameters are set correctly.
- The DISPLAY environment variable is set and the user has sufficient permissions to display to the specified DISPLAY.
- The system has a sufficient swapping set.
- The Oracle home for the new installation either is empty or is one of a handful of supported releases on top of which Oracle Database 11g can be installed. The installation process also verifies that those releases 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



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.

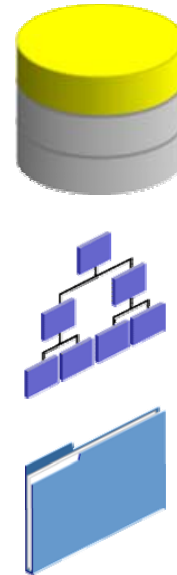
OFA is designed to:

- 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) that are often vulnerable to data corruption
- Facilitate switching between multiple Oracle databases
- 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, see the *Oracle Installation Guide for UNIX Systems*.

Optimal Flexible Architecture: Naming Scheme

- **Mount points:**
 - /u01
 - /disk01
- **Directories:**
 - /u01/app/oracle
 - /u01/app/applmgr
- **Files:**
 - **Control files:** `controln.ctl`
 - **Redo log files:** `redon.log`
 - **Data files:** `tn.dbf`



Optimal Flexible Architecture: Naming Scheme

At the core of OFA is a scheme that provides naming standards for your mount points (which are often the physical disks), the directories and subdirectories on those mount points, and 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) that is used to distinguish each mount point. Examples of OFA-compliant 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. 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 Oracle Database 11g version 11.1.0 is the following:

```
/u01/app/oracle/product/11.1.0/db_1
```

Using Optimal Flexible Architecture

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.)
- **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

Note: `bdump` and `udump` no longer exist in `$ORACLE_BASE/admin`. Their contents have been modified and moved to the `DIAGNOSTIC_DEST/diag/db_name/sid/` subdirectories. The `cdump` destination has been moved to `DIAGNOSTIC_DEST/diag/rdbms/db_name/sid/cdump`.

Database file 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:** Mount point name (as described previously)
- **q:** String distinguishing the Oracle data from all other files (commonly named `ORACLE` or `oradata`)
- **d:** Value of the initialization parameter `DB_NAME` (the database name)
- **t:** Oracle tablespace name
- **n:** Two-digit string

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

Setting Environment Variables

- **ORACLE_BASE:** Base of the Oracle directory structure for OFA
- **ORACLE_HOME:** Directory containing the Oracle software
- **ORACLE_SID:** Initial instance name (default: ORCL)
- **NLS_LANG:** Language, territory, and client character set settings



ORACLE

Setting Environment Variables

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

- **ORACLE_BASE:** Specifies the base of the Oracle directory structure for OFA. Use is optional; if used, it can facilitate future installations and upgrades. This is a directory path, as shown in the following example:
`/u01/app/oracle`
- **ORACLE_HOME:** Specifies the directory containing the Oracle software. This is a directory path, as shown in the following example:
`$ORACLE_BASE/product/11.1.0/db_1`
- **ORACLE_SID:** The initial instance name (default: ORCL). This 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*, as in the following example:

```
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 as follows:

```
select * from nls_session_parameters;
```

For more information about valid languages, territories, character sets, and language support, see the *Oracle Database Globalization Support Guide*.

Note: A Windows installation defaults the NLS_LANG values in the registry, where the *language* part originates from the keyboard language. The result is 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)

Oracle Database 11g Installation - Select Installation Method

Select Installation Method

Basic Installation
Perform full Oracle Database 11g installation with standard configuration options requiring minimal input. This option uses file system for storage, and a single password for all database accounts.

Oracle Base Location: /u01/app/oracle

Oracle Home Location: /u01/app/oracle/product/11.1.0/db_1

Installation Type: Enterprise Edition (3.3GB)

UNIX DBA Group: dba

Create Starter Database (additional 720MB)

Global Database Name: orcl

Database Password: ***** Confirm Password: *****
This password is used for the SYS, SYSTEM, SYSMAN, and DBSNMP accounts.

Advanced Installation
Allows advanced selections such as different passwords for the SYS, SYSTEM, SYSMAN, and DBSNMP accounts, database character set, product languages, automated backups, custom installation, and alternative storage options such as Automatic Storage Management.

Oracle Universal Installer (OUI)

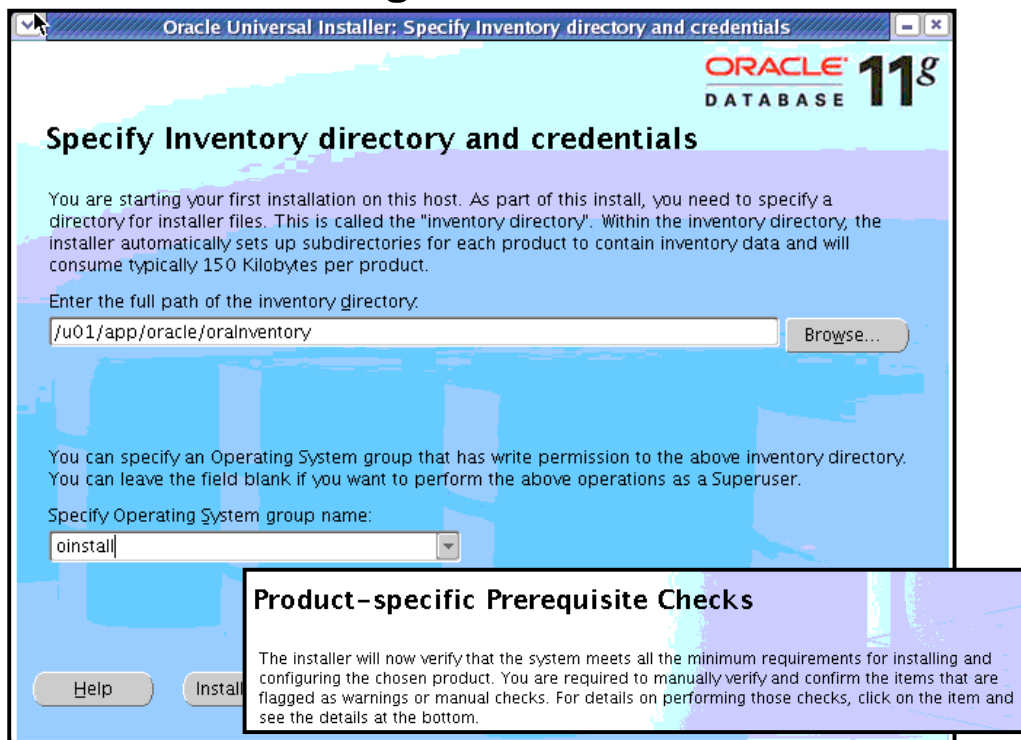
Oracle Universal Installer (OUI) is a Java application that performs component-based installations and enables different levels of integrated bundle, suite, and Web-based installations as well as complex logic in a single package. The installation engine is easily portable across all Java-enabled platforms, 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: To start OUI, insert the Oracle Database installation disk, navigate to the client directory, and double-click `setup.exe`. On the Welcome page, select your installation type: Instant Client, Administrator, Runtime, or Custom.

Installing the Oracle Software



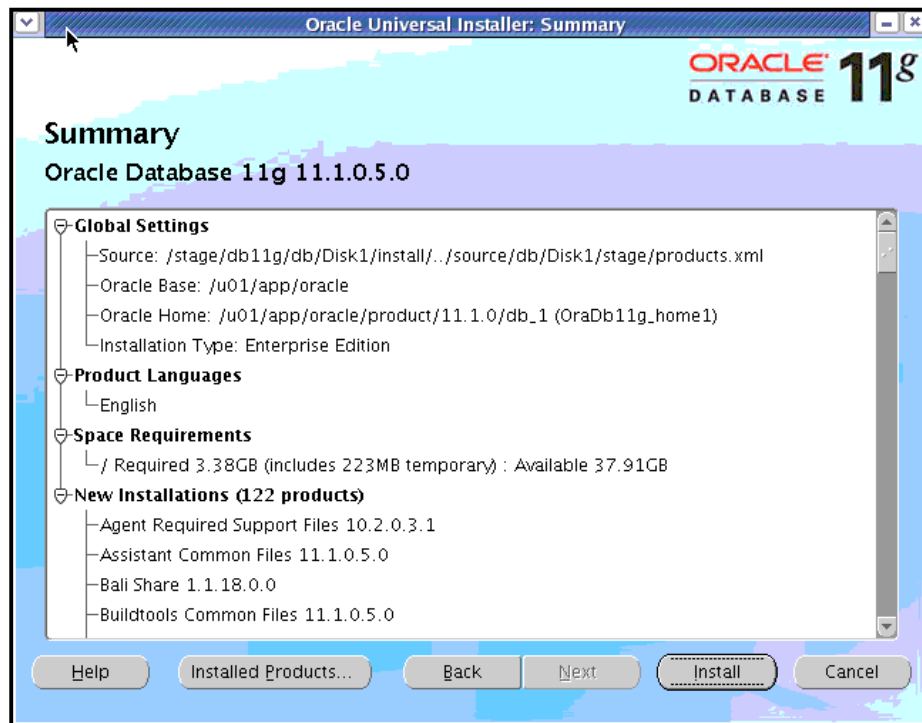
Installing the Oracle Software

To install the Oracle software by using OUI:

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 the preinstallation settings according to your installation plan.

With the initial information, OUI executes prerequisite checks.

Database Configuration Options



Database Configuration Options

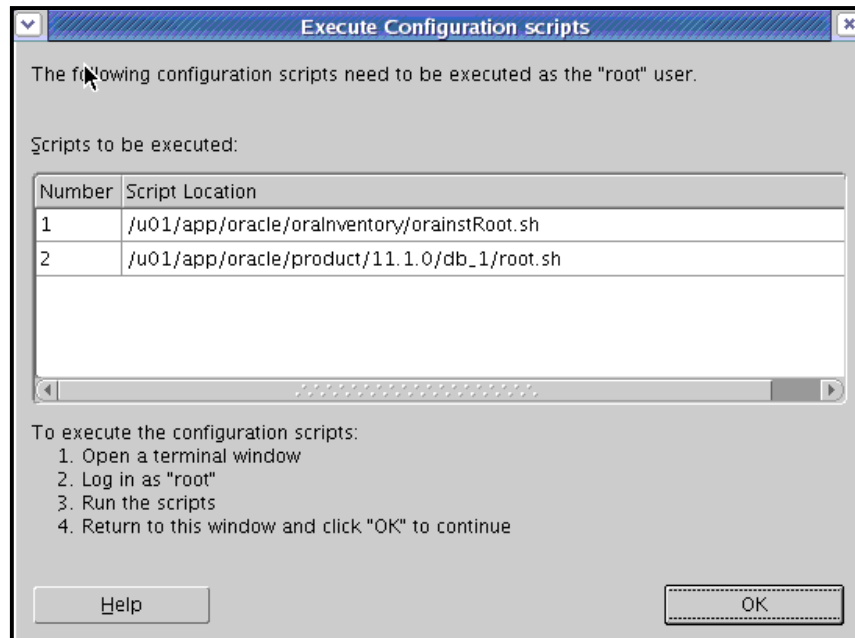
Your installation process continues:

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

If you chose to create a starter database as part of the installation, 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.

Executing Configuration Scripts



Executing Configuration Scripts

Your installation process continues:

7. 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/11.1.0/db_1
# ./root.sh
```
8. 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.
9. 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**
- **Email notification options**
- **Oracle Portable Clusterware**
- **Cloning**

ORACLE

2 - 17

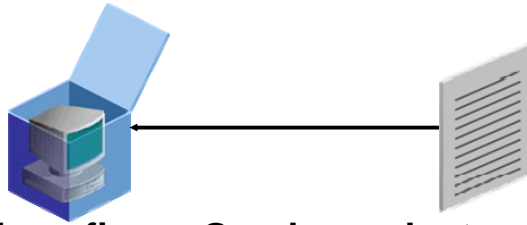
Copyright © 2007, Oracle. All rights reserved.

Advanced Installation Options

- With OUI, you can create configurations that use Automatic Storage Management.
- You can install and configure the Oracle 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 EM Database Control, you can optionally configure the database to use the Oracle-recommended default backup strategy.
- If you choose to use EM Database Control during the installation, you can configure Enterprise Manager to send email alerts to an email 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 11g installation supports RAC features, particularly the installation of clusterware.
- Oracle homes can be cloned by using the Enterprise Configuration Management tool, which enables users to create clone requests and then schedule and process them. This tool is available through EM Grid Control.

Note: These options are available only when creating a database. If you perform the installation without creating the database, these options are unavailable. When you create the database by using the DBCA, you have the opportunity to select these options.

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 the DBCA in silent mode.

ORACLE

Installation Option: Silent Mode

To install and configure Oracle products by using OUI in silent or suppressed mode:

1. Prepare a response file. File templates for each product and installation type are provided, such as `enterprise.rsp`, `standard.rsp`, and `netca.rsp`.
2. 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>
```

In the syntax, `-destinationFile` is the file location.
3. Run OUI in silent or suppressed mode. Run the `$ORACLE_BASE/oraInventory/orainstRoot.sh` and `$ORACLE_HOME/root.sh` at the end of the installation.
4. If you completed a software-only installation, run Oracle Net Configuration Assistant (NetCA) and the 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 the Oracle software by using OUI**

ORACLE

Practice 2 Overview: Preparing the Database Environment

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

Note: Completing this practice is critical for all subsequent practices.

3

Creating an Oracle Database

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

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

- Create a database by using the Database Configuration Assistant (DBCA)
- Create a listener by using the Oracle Net Configuration Assistant (NetCA)
- Create a database design template with the DBCA
- Generate database creation scripts with the DBCA

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? What type of storage is being used?
 - How many data files will you need? (Plan for growth.)
 - How many tablespaces will you use?
 - What types of information will be stored?
 - Are there any special storage requirements due to type or size?
- **Overall database design**
- **Database backup strategy**



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. Information such as the availability of network attached storage (NAS) and the bandwidth for the private storage network are important. If storage area networks (SAN) are going to be used, knowing how the logical volumes are configured and the stripe size is essential.

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.

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, for a software house or a nursery)

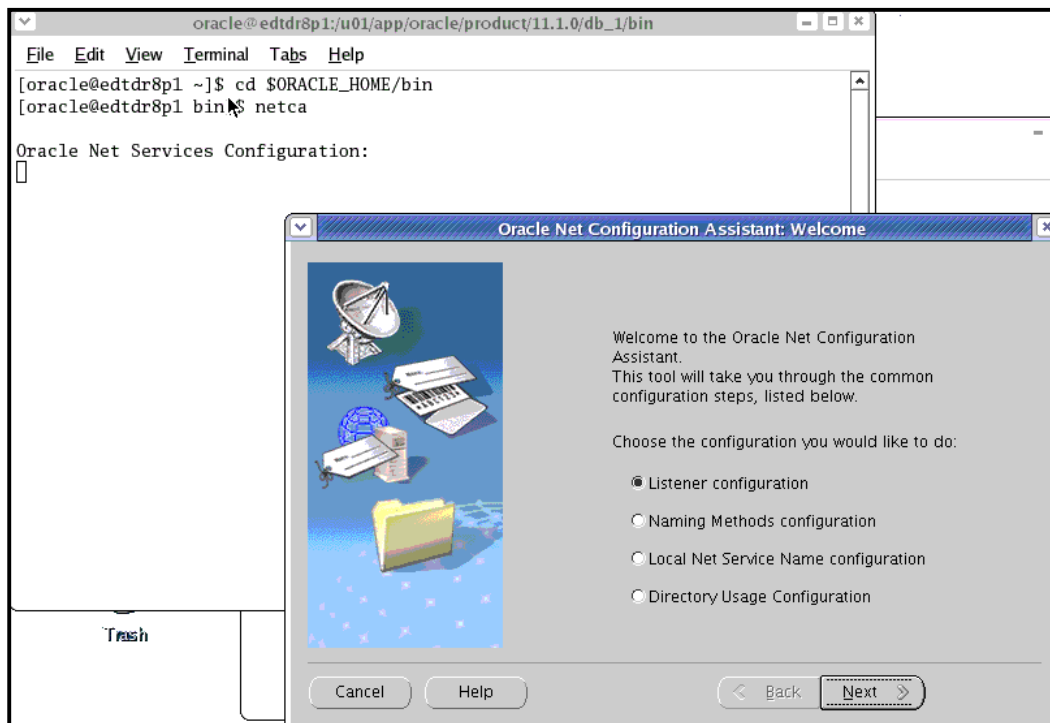
Databases: Examples

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:** For storing data for long periods and retrieving them in read operations
- **Transaction processing:** For accommodating many (usually small) transactions
- **General purpose:** For working with transactions and storing them for a medium length of time

The information on this page and the previous one present considerations that you will encounter as a DBA. This course (in its entirety) is designed to help you address them.

Configuring the Listener



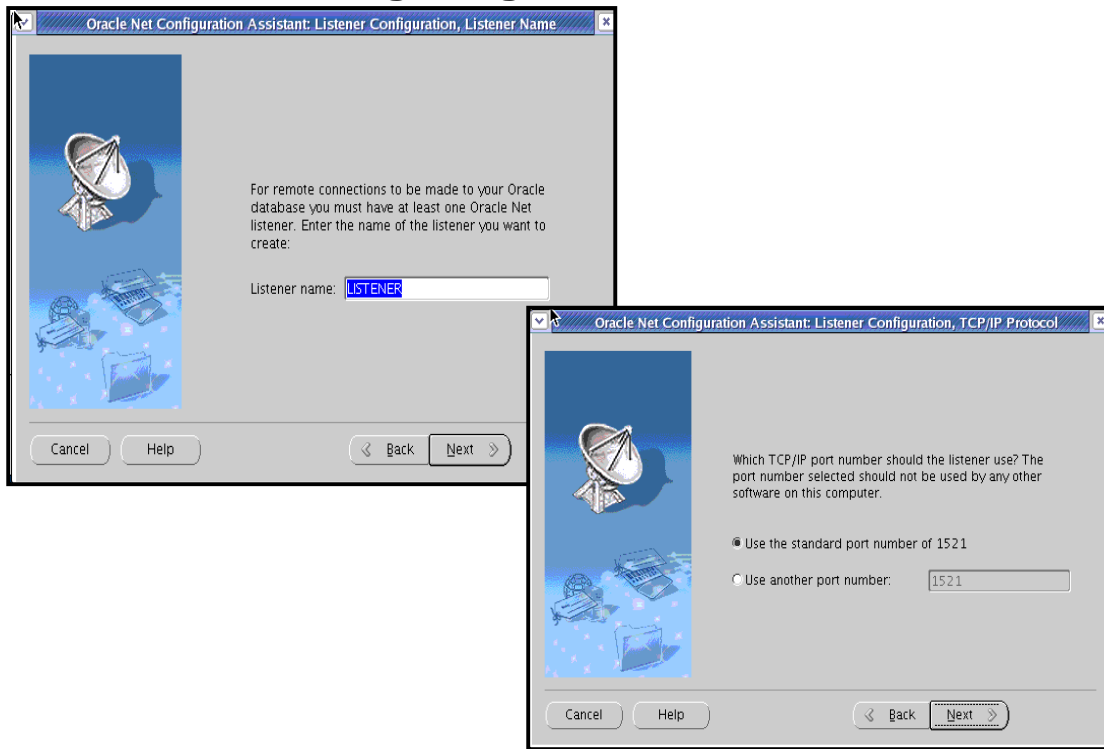
Configuring the Listener

When an instance starts, a listener process establishes a communication pathway to Oracle Database. When a user process makes a connection request, the listener determines whether it should use a shared server dispatcher process or a dedicated server process, and establishes an appropriate connection.

The listener also establishes a communication pathway between databases. When multiple databases or instances run on one computer, as in Oracle Real Application Clusters, service names enable instances to register automatically with other listeners on the same computer. A service name can identify multiple instances, and an instance can belong to multiple services. Clients connecting to a service do not need to specify which instance they require.

A listener has to be configured prior to configuring EM using DBCA. The Oracle Net Configuration Assistant (NetCA) is a simple tool for configuring the listener. The NetCA enables you to configure the listening protocol address and service information for an Oracle database. Use the NetCA for initial network configuration after database installation.

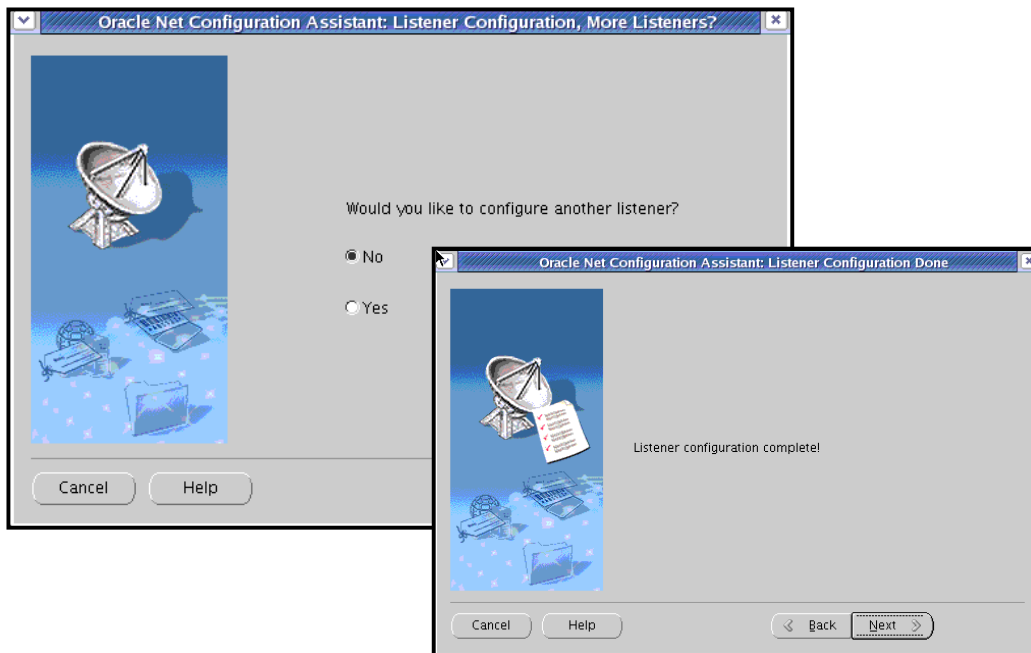
Configuring the Listener



Configuring the Listener (continued)

1. Start `netca` from a command prompt.
2. Select Listener Configuration.
3. When prompted, enter the name of the listener.
4. Specify the port number for the listener.

Configuring the Listener

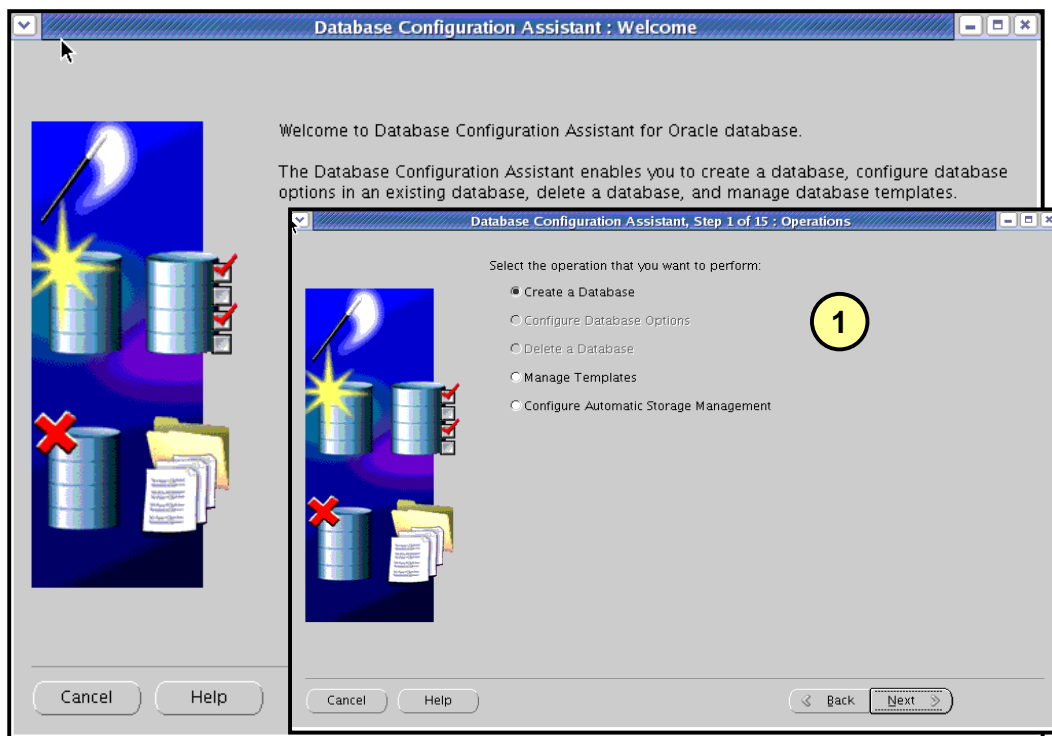


Configuring the Listener (continued)

5. Select No when you are asked if you want to configure another listener. Your listener configuration is complete.

You are now ready to begin database creation by using the DBCA.

Database Configuration Assistant (DBCA)



3 - 8

Copyright © 2007, Oracle. All rights reserved.

ORACLE

Database Configuration Assistant (DBCA)

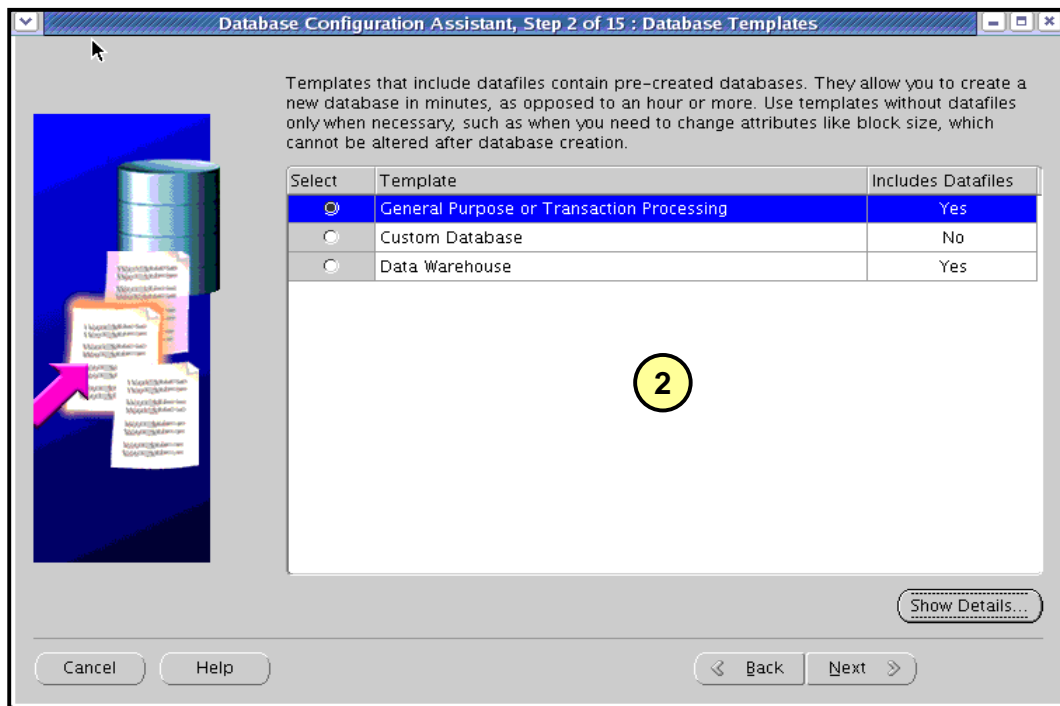
You can use the Database Configuration Assistant (DBCA) to create, change the configuration of, or delete a database. You can also create a database from a list of predefined templates or use an existing database as a sample to create a new database or template. This is sometimes referred to as “database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

To invoke the DBCA:

1. Log on to your computer as a member of the OS DBA 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



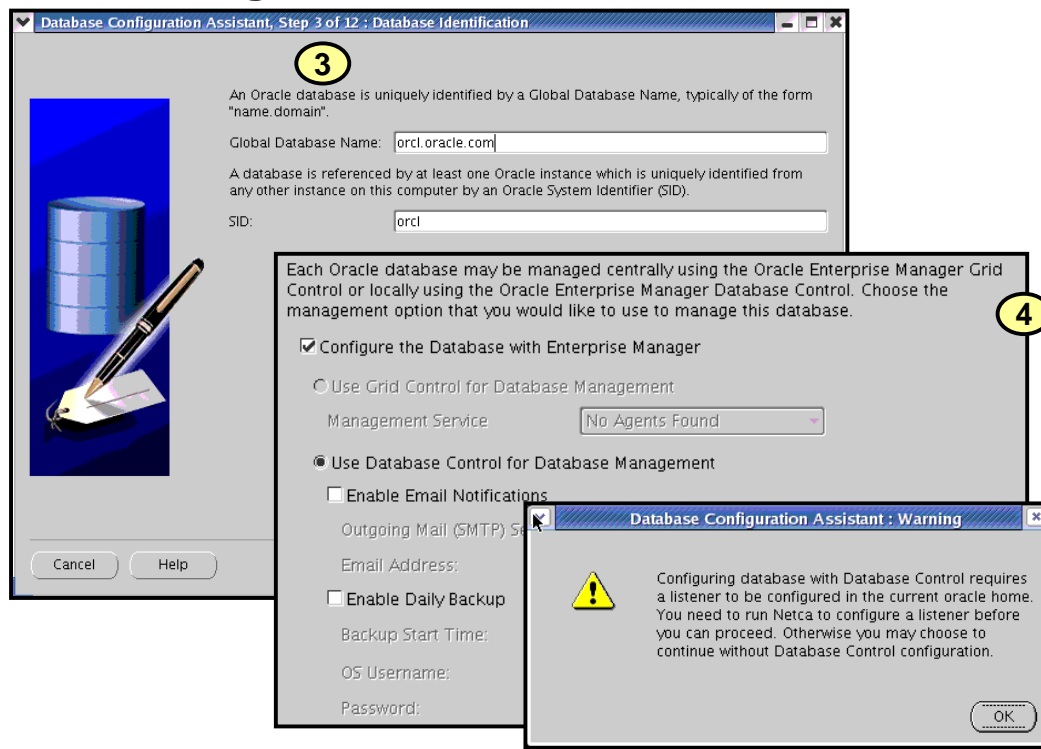
Using the DBCA to Create a Database

To use the DBCA to create a database:

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 - 10

Copyright © 2007, Oracle. All rights reserved.

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.”

Note: You must configure the listener before you can configure Enterprise Manager (as shown earlier).

Using the DBCA to Create a Database

The image displays three overlapping screenshots from the Oracle Database Configuration Assistant (DBCA) interface, illustrating the steps to create a database:

- Step 5:** A dialog box titled "For security reasons, you must specify passwords for the following user accounts in the new database." It offers two options: "Use the Same Password for All Accounts" (selected) and "Use Different Passwords". The "Password:" and "Confirm Password:" fields both contain "*****".
- Step 6:** A dialog box titled "Select the storage mechanism you would like to use for the database." It offers two options: "File System" (selected) and "Automatic Storage Management". The "File System" option is selected, and the text "Use the File System for Database storage." is displayed.
- Step 7:** A dialog box titled "Database Configuration Assistant, Step 7 of 14 : Database File Locations". It asks to "Specify locations for the Database files to be created:" and offers three options: "Use Database File Locations from Template" (selected), "Use Common Location for All Database Files", and "Use Oracle-Managed Files". The "Use Database File Locations from Template" option is selected, and the "Database Files Location:" field is empty. There is also a "Database Area:" field and a "Multiplex Redo Logs and Control Files..." button. An information icon (i) is present with a note: "If you want to specify different locations for any database files, options except Oracle-Managed Files and use the Storage page for each file location. If you use Oracle-Managed Files, Oracle automatically generates the names for database files, which can not be changed on the fly."

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 the password for all administrative accounts.
- 6. Storage Options:** Specify the type of storage mechanism (such as File System) that you want your database to use.
- 7. Database File Locations:** Choose according to your needs. Using Oracle Managed Files (OMF) eliminates the need for you to directly manage the operating system files in an Oracle database. You specify operations in terms of database objects rather than file names. For details, see the lesson titled "Managing Database Storage Structures."

Using the DBCA to Create a Database

Choose the recovery options for the database: 8

Specify Flash Recovery Area

This is used as the default for all backup and recovery operations, and is also required for automatic backup using Enterprise Manager. Oracle recommends that the database files and recovery files be located on physically different disks for data protection and performance.

Flash Recovery Area:

Flash Recovery Area Size:

Enable Archiving

Sample Schemas 9

Sample Schemas illustrate the use of a layered approach to complexity, and are used by some demonstration programs. Installing this will give you the following schemas in your database: Human Resources, Order Entry, Online Catalog, Product Media, Information Exchange, Sales History. It will also create a tablespace called EXAMPLE. The tablespace will be about 130 MB.

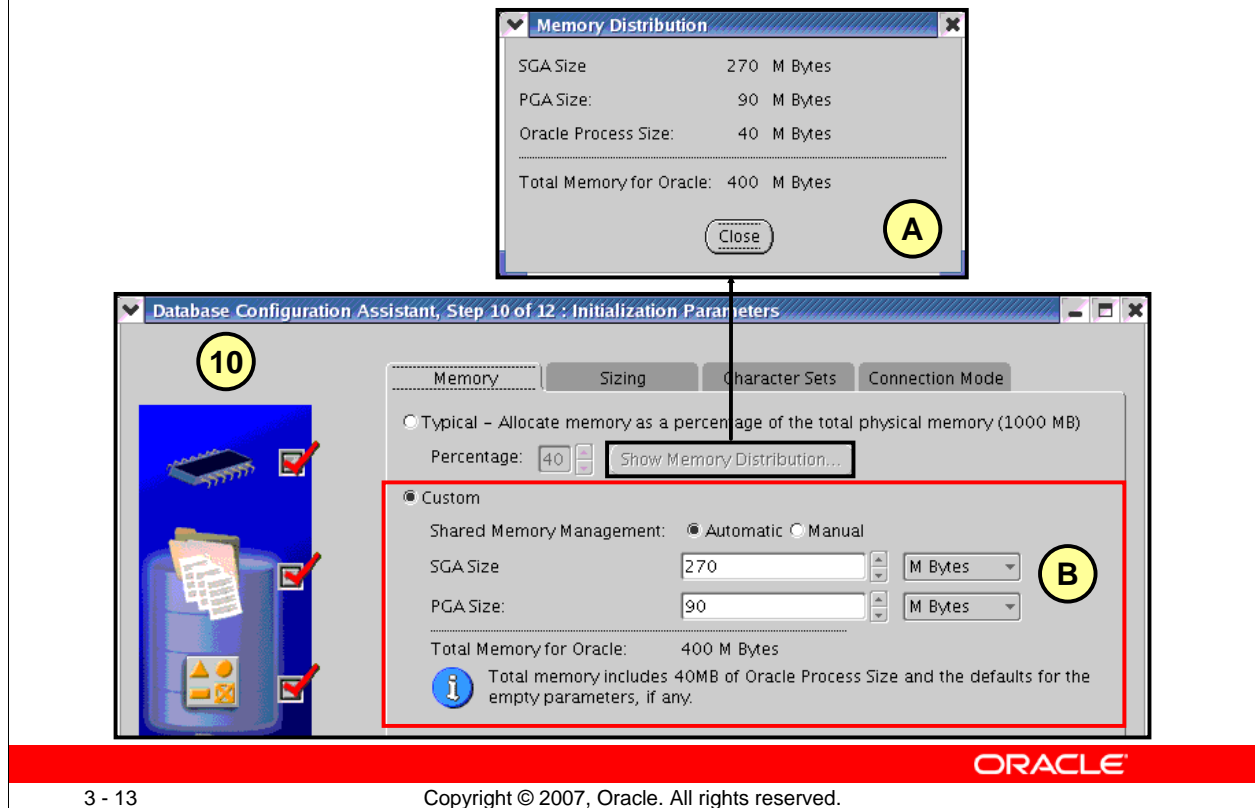
Specify whether or not to add the Sample Schemas to your database.

Sample Schemas

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 using custom scripts.

Using the DBCA to Create a Database



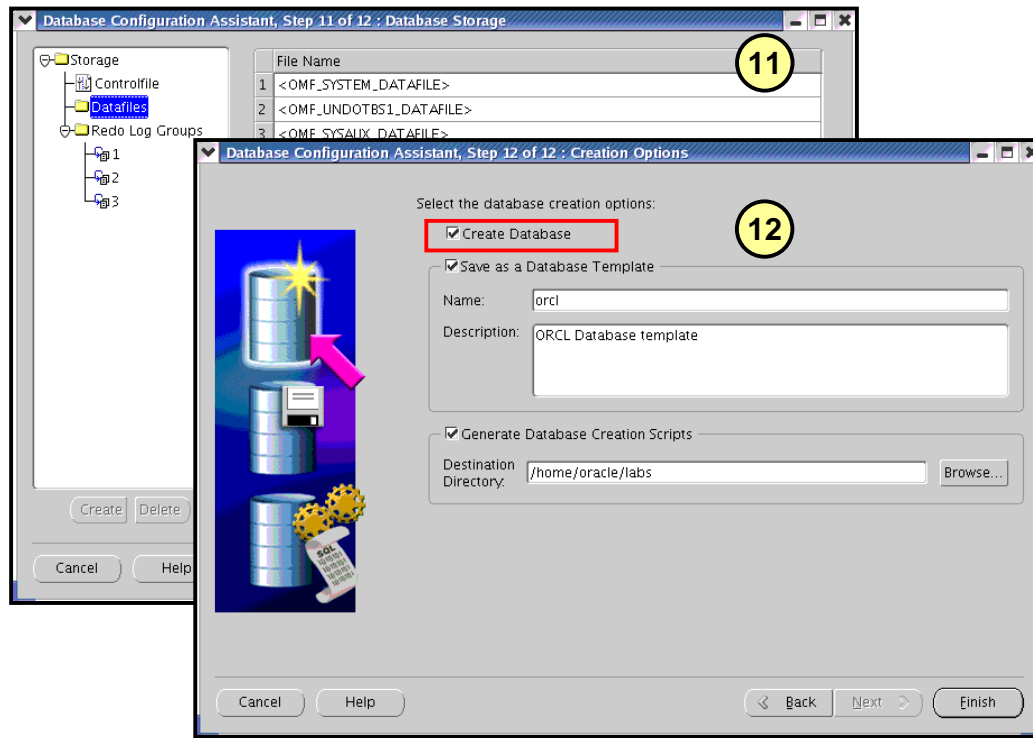
Using the DBCA to Create a Database (continued)

10. **Initialization Parameters:** The tabs on this page provide access to pages that enable you to change default database 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 Corporation 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.

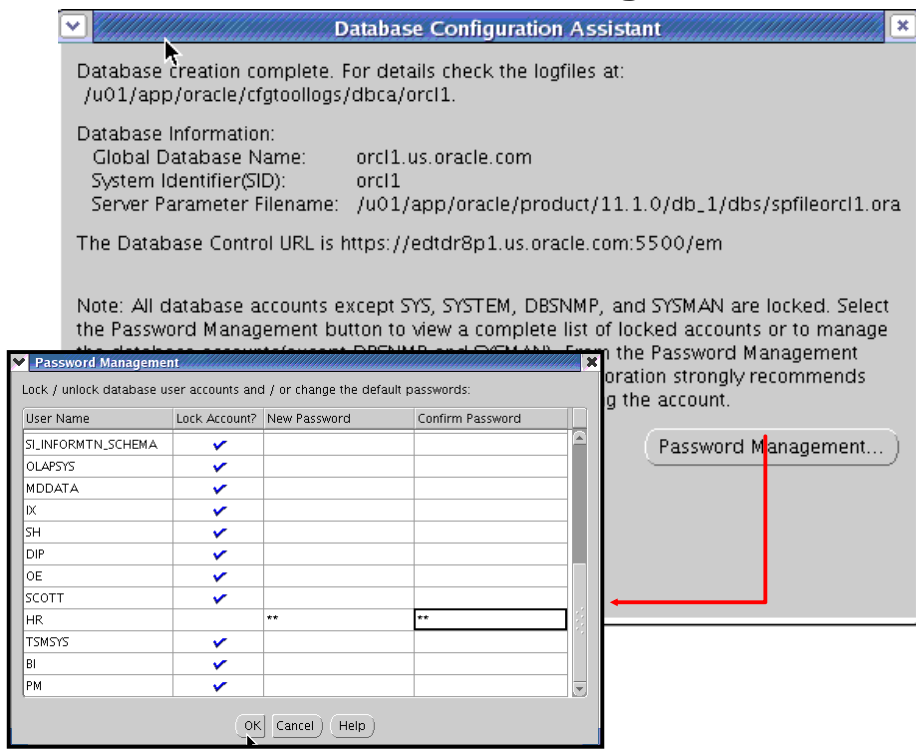
Using the DBCA to Create a Database



Using the DBCA to Create a Database (continued)

- 11. Database Storage:** If you selected one of the preconfigured templates for a database, 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 following options: create your database at this time, save the database definition as a template, and generate scripts. If you choose all options, the DBCA first saves the database template, then generates the scripts into your destination directory, and finally creates your database.

Password Management



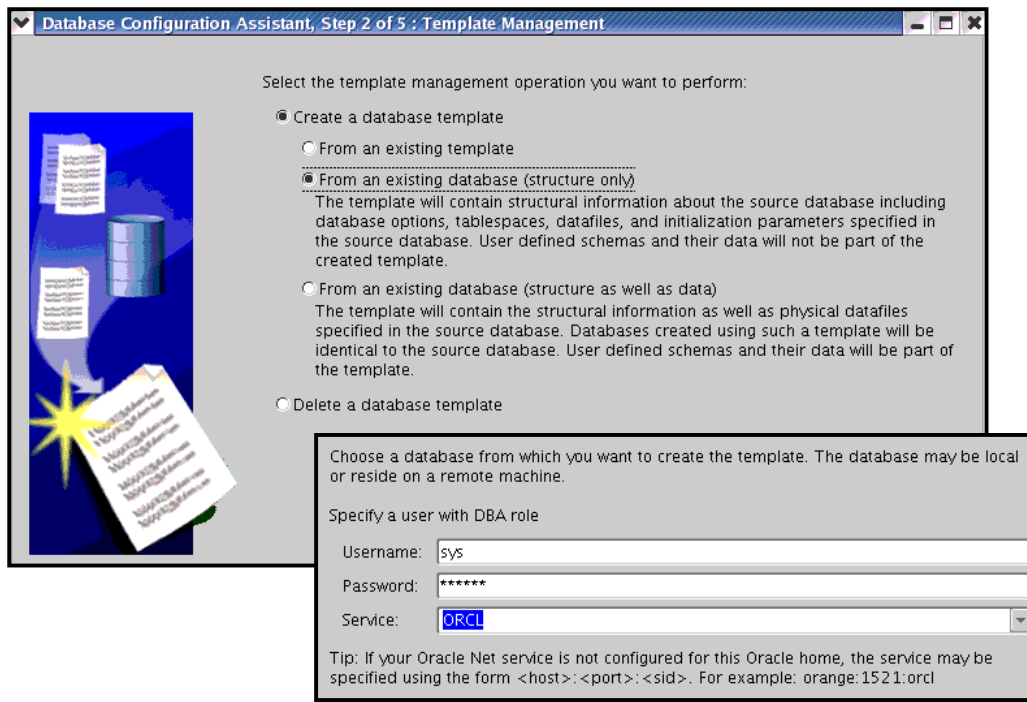
Password Management

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

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

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



Database Configuration Assistant, Step 2 of 5 : Template Management

Select the template management operation you want to perform:

- Create a database template
 - From an existing template
 - From an existing database (structure only)
The template will contain structural information about the source database including database options, tablespaces, datafiles, and initialization parameters specified in the source database. User defined schemas and their data will not be part of the created template.
 - From an existing database (structure as well as data)
The template will contain the structural information as well as physical datafiles specified in the source database. Databases created using such a template will be identical to the source database. User defined schemas and their data will be part of the template.
- Delete a database template

Choose a database from which you want to create the template. The database may be local or reside on a remote machine.

Specify a user with DBA role

Username:

Password:

Service:

Tip: If your Oracle Net service is not configured for this Oracle home, the service may be specified using the form <host>:<port>:<sid>. For example: orange:1521:orcl

Creating a Database Design Template

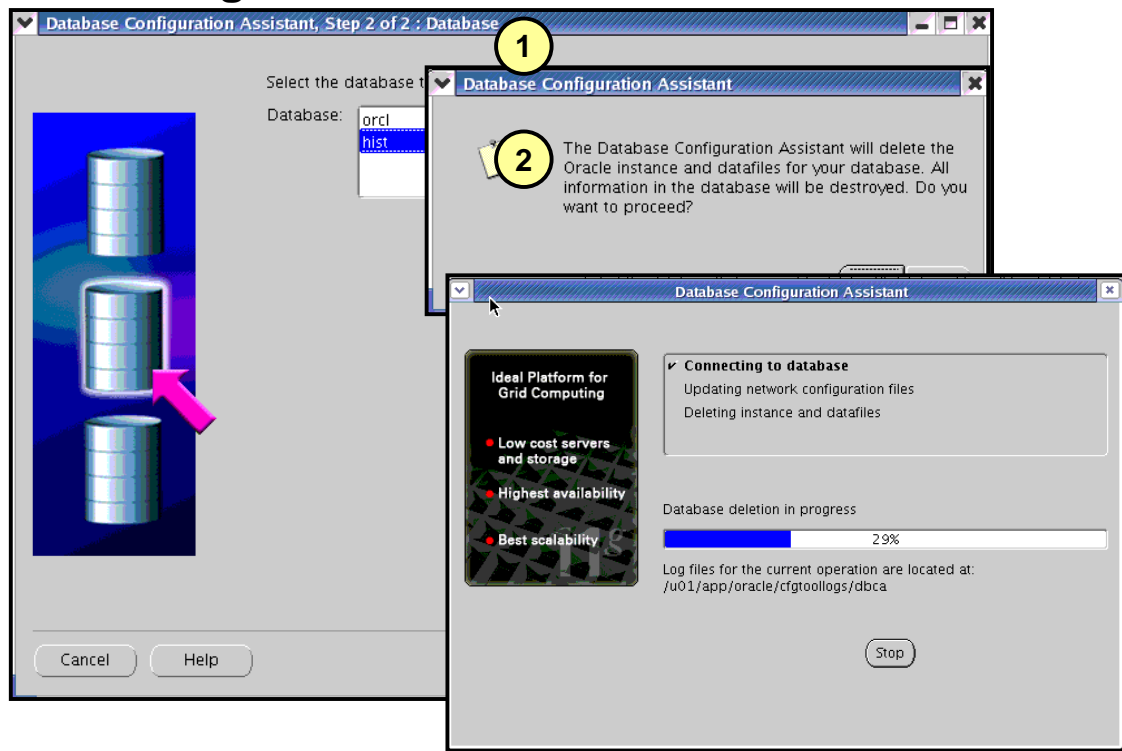
A template is a predefined database definition that you use as a starting point for a new database. If you do not create a template as part of the database creation process, you can do it at any time by invoking the DBCA.

There are 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



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 the 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. Then 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 of the following conditions must apply:

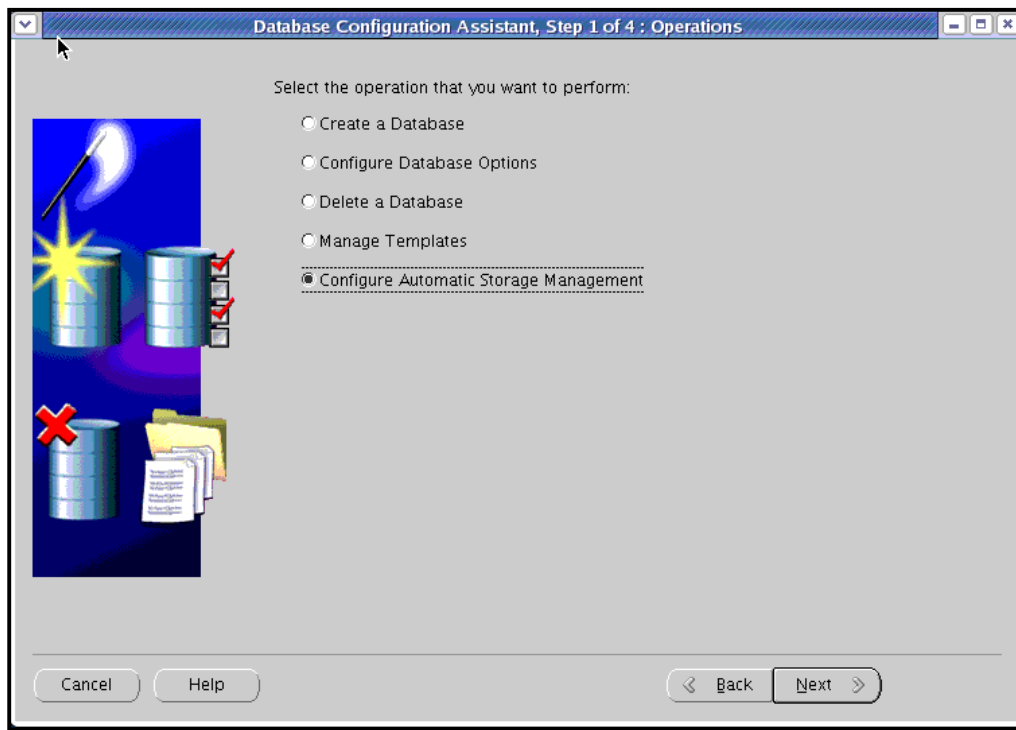
- The database must be mounted and closed.
- The database must be mounted exclusively (not in shared mode).
- The database must be mounted as `RESTRICTED`.

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, the actual raw disk special files are not deleted.

Using the DBCA for Additional Tasks



Using the DBCA for Additional Tasks

You can use the DBCA to configure for Automatic Storage Management or the Manage templates. You can also use Oracle installer to create an ASM instance and database. Oracle recommends that you install Automatic Storage Management in its own Oracle home, regardless of whether you plan to have only one or multiple database instances. Installing Automatic Storage Management in its own Oracle home helps ensure higher availability and manageability.

With separate Oracle homes, you can upgrade Automatic Storage Management and databases independently, and you can remove database software without affecting the Automatic Storage Management instance.

Note: For more information about installing or configuring ASM, see the Oracle Database 11g installation guide for your operating system.

Summary

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

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

Practice 3 Overview: Using the DBCA

This practice covers the following topics:

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

Note: Completing database creation and unlocking the HR schema are critical for all following practices.

Optional:

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

ORACLE

4

Managing the Oracle Instance

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

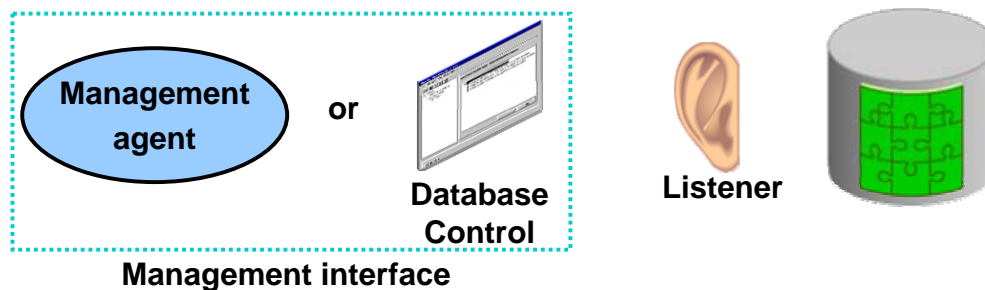
- **Start and stop the Oracle database and components**
- **Use Oracle Enterprise Manager**
- **Access a database with SQL*Plus**
- **Modify database initialization parameters**
- **Describe the stages of database startup**
- **Describe database shutdown options**
- **View the alert log**
- **Access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

ORACLE

Management Framework

Oracle Database 11g management framework components:

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



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 node where the database server runs (which connects it to Oracle Enterprise Manager Grid Control) or the stand-alone Oracle Enterprise Manager Database Control. This is also referred to as the *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 it is activated, the management interface can be used to start the other components.

Starting and Stopping Database Control

```
$ emctl start dbconsole
Oracle Enterprise Manager 11g Database Control Release 11.1.0.1.0
Copyright (c) 1996, 2006 Oracle Corporation. All rights reserved.
https://edrsr17pl.us.oracle.com:1158/em/console/aboutApplication
Starting Oracle Enterprise Manager 11g Database Control .....
started.
-----
Logs are generated in directory /u01/app/oracle/product/11.1.0/db_1/
edrsr17pl.us.oracle_orcl/sysman/log
```

```
$ emctl stop dbconsole
Oracle Enterprise Manager 11g Database Control Release 11.1.0.1.0
Copyright (c) 1996, 2006 Oracle Corporation. All rights reserved.
https://edrsr17pl.us.oracle.com:1158/em/console/aboutApplication
Stopping Oracle Enterprise Manager 11g Database Control ...
... Stopped.
```

ORACLE

Starting and Stopping Database Control

Oracle Database provides *Database Control*, which is a stand-alone management console 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; from any one Database Control, you can manage only one database. Before using Database Control, ensure that a `dbconsole` process is started.

Command to start the `dbconsole` process:

```
emctl start dbconsole
```

Command to stop the `dbconsole` process:

```
emctl stop dbconsole
```

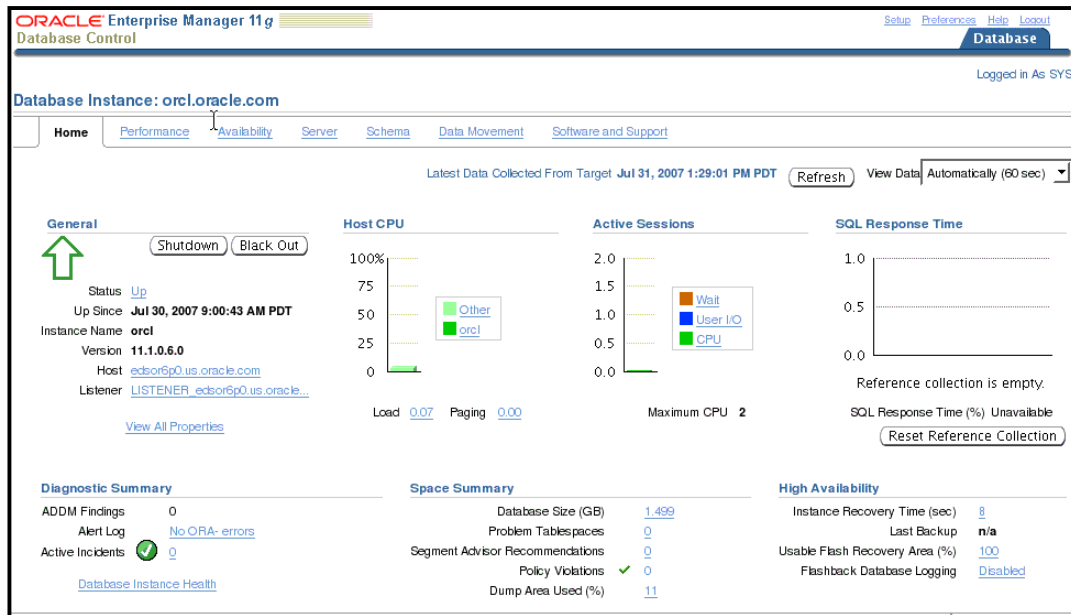
Command to view the status of the `dbconsole` process:

```
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

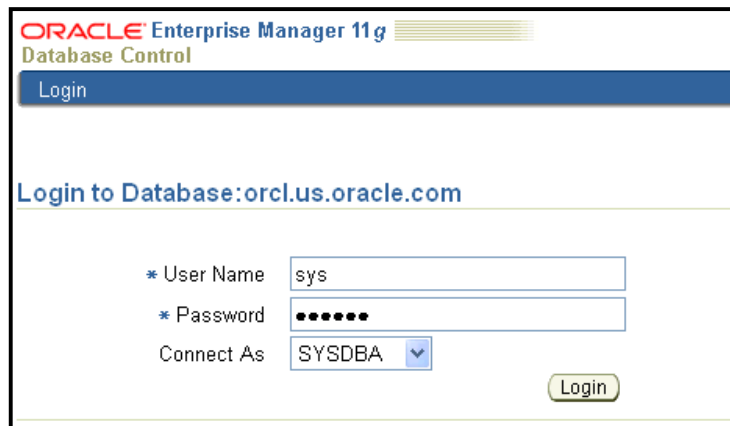
When you install an Oracle database, Oracle Universal Installer also installs Oracle Enterprise Manager (Enterprise Manager, or EM). Its Web-based Database Control serves as the primary tool for managing your Oracle database. EM provides a graphical interface for doing almost any task that you need to 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 EM. In most cases, you can click links in EM to find more specific information about the contents of a page.

Note: In Oracle Database 11g, the URL to access EM uses HTTPS (instead of HTTP) as the protocol to enable a secure connection. To reach the EM dbconsole, you must therefore enter a URL in the following format:

```
https://machine_name:port/em
```

The default port number is 1158.

Accessing Oracle Enterprise Manager



ORACLE Enterprise Manager 11g
Database Control

Login

Login to Database: orcl.us.oracle.com

* User Name

* Password

Connect As ▼

Login

Accessing Oracle Enterprise Manager

In a Web browser, enter the following URL:

`https://host name:port number/em`

Although the default port number is 1158, it is possible to have different numbers, especially if there are multiple databases on the same host. To determine the port number, check the `portlist.ini` file. In addition, ports for some Oracle Database applications are listed in the `portlist.ini` file. This file is located in the

`ORACLE_BASE\ORACLE_HOME\install` directory.

- 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 database installation. For the Connect As option, select either SYSDBA or SYSOPER to log in to the database with special database administration privileges.
- If the database is *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

ORACLE Enterprise Manager 11g Database Control

Database Instance: orcl.us.oracle.com

Home Performance Availability Server Schema Data Movement Software and Support **Property pages**

Latest Data Collected From Target May 30, 2007 9:45:35 PM GMT+07:00 Refresh View Data Automatically (60 sec)

General Shutdown Black Out

Status Up
Up Since May 10, 2007 10:21:07 AM GMT+07:00
Instance Name orcl
Version 11.1.0.3.0
Host edrsr17p1.us.oracle.com
Listener LISTENER_edrsr17p1.us.orac...

View All Properties

Health Meter

1 issues
0 Incidents
1 Alerts
Health History

Host CPU

100%
75%
50%
25%
0%
Other
orcl
Load 1.08 Paging 0.00 Maximum CPU 1

Active Sessions

1.0
0.5
0.0
Wait
User I/O
CPU

Diagnostic Summary

ADDM Findings 0
Alert Log No ORA errors
SQL Response Time Unavailable
Reset Reference Collection

Space Summary

Database Size (GB) 1,664
Problem Tablespaces 0
Segment Advisor Recommendations 0
Space Violations 0
Dump Area Used (%) 43

High Availability

Instance Recovery Time (sec) 25
Last Backup n/a
Usable Flash Recovery Area (%) 100
Flashback Logging Disabled

Alerts

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 (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

Other Oracle Tools

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

SQL*Plus provides an additional interface to your database so that you can:

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

SQL Developer:

- **Is a graphical user interface for accessing your instance of Oracle Database**
- **Supports development in both SQL and PL/SQL**
- **Is available in the default installation of Oracle Database**

Other Oracle Tools

In addition to Enterprise Manager, you can use SQL*Plus and SQL Developer 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.

SQL*Plus is a command-line program that you use to submit SQL and PL/SQL statements to an Oracle database. You can submit statements interactively or as SQL*Plus scripts. SQL*Plus is installed with the database and is located in your \$ORACLE_HOME/bin directory.

You can start SQL*Plus from the command line, or from the Start menu on a Windows client.

SQL Developer is a graphical user interface for accessing your instance of Oracle Database. SQL Developer supports development in both the SQL and PL/SQL languages. It is available in the default installation of Oracle Database.

With SQL Developer, you can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. You can also run any number of provided reports, as well as create and save your own.

Note: This course uses EM and SQL*Plus.

Using SQL*Plus

SQL*Plus is:

- A command-line tool
- Used interactively or in batch mode

```
$ sqlplus hr/hr

SQL*Plus: Release 11.1.0.3.0 - Beta on Wed May 30 21:41:24 2007
Copyright (c) 1982, 2006, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.3.0 - Beta
With the Partitioning, OLAP and Data Mining options

SQL> select last_name from employees;

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

ORACLE

Using SQL*Plus

You can use the command-line interface of 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:

1. Open a terminal window.
2. At the command-line prompt, enter the SQL*Plus command in the following form:

```
$ sqlplus <userid>/<pwd> or nolog
```
3. If you use the NOLOG option, you must enter CONNECT followed by the username you want to connect as.
4. When prompted, enter the user's password. SQL*Plus starts and connects to the default database.

Note: Entering the username and password at the command line exposes the password to anyone with privileges to run the `ps` command at the OS level on UNIX or Linux systems. It is therefore preferable to use the NOLOG option to log in.

Calling SQL*Plus from a Shell Script

```
$ ./batch_sqlplus.sh
```

```
SQL*Plus: Release 11.1.0.3.0 - Beta on Wed May 30 21:41:24 2007  
Copyright (c) 1982, 2006, Oracle. All rights reserved.
```

```
Connected to:
```

```
Oracle Database 11g Enterprise Edition Release 11.1.0.3.0 - Beta  
With the Partitioning, OLAP and Data Mining options
```

```
SQL>
```

```
  COUNT(*)
```

```
-----  
          107
```

```
SQL>
```

```
107 rows updated.
```

```
SQL>
```

```
Commit complete.
```

```
SQL> Disconnected from Oracle Dat
```

```
11.1.0.3.0 - Beta
```

```
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
```

ORACLE

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 11.1.0.3.0 - Beta on Wed May 30 21:41:24 2007
```

```
Copyright (c) 1982, 2006, Oracle. All rights reserved.
```

```
Connected to:
```

```
Oracle Database 11g Enterprise Edition Release 11.1.0.3.0 - Beta
With the Partitioning, OLAP and Data Mining options
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
60	IT	103	1400

```
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.3.0 - Beta
With the Partitioning, OLAP and Data Mining options
```

ORACLE

4 - 11

Copyright © 2007, Oracle. All rights reserved.

Calling a SQL Script from SQL*Plus

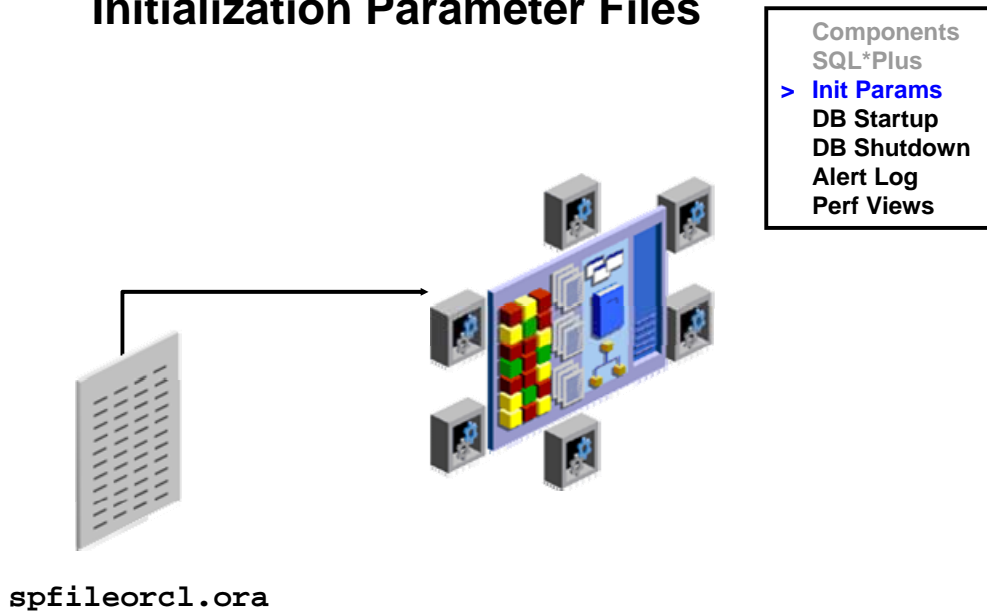
You can call an existing SQL script file from within SQL*Plus. This can be done at the command line when first invoking SQL*Plus, as shown in the slide. It can also be done from inside a SQL*Plus session simply by using the “@” operator. For example, this runs the script from within an already established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

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

Note: The default file extension for script files is .sql. When a script is saved from SQL*Plus by using the save command, this extension is automatically supplied. Scripts with this extension can be executed even without supplying the extension at execution time, as in the following example:

```
SQL> @script
```

Initialization Parameter Files



Initialization Parameter Files

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

- **Server parameter file (SPFILE):** 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 on which the Oracle database is executing; it is persistent across shutdown and startup. 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 way to maintain initialization parameters. By using an SPFILE, you can store and manage your initialization parameters persistently in a server-side disk file.

Initialization Parameter Files (continued)

Types of Initialization Parameters

The Oracle database server has the following types of initialization parameters:

Derived Parameters

Some initialization parameters are derived, meaning that their values are calculated from the values of other parameters. Normally, you should not alter values for derived parameters. But if you do, the value that you specify overrides the calculated value.

For example, the default value of the `SESSIONS` parameter is derived from the value of the `PROCESSES` parameter. If the value of `PROCESSES` changes, the default value of `SESSIONS` changes as well unless you override it with a specified value.

Operating System–Dependent Parameters

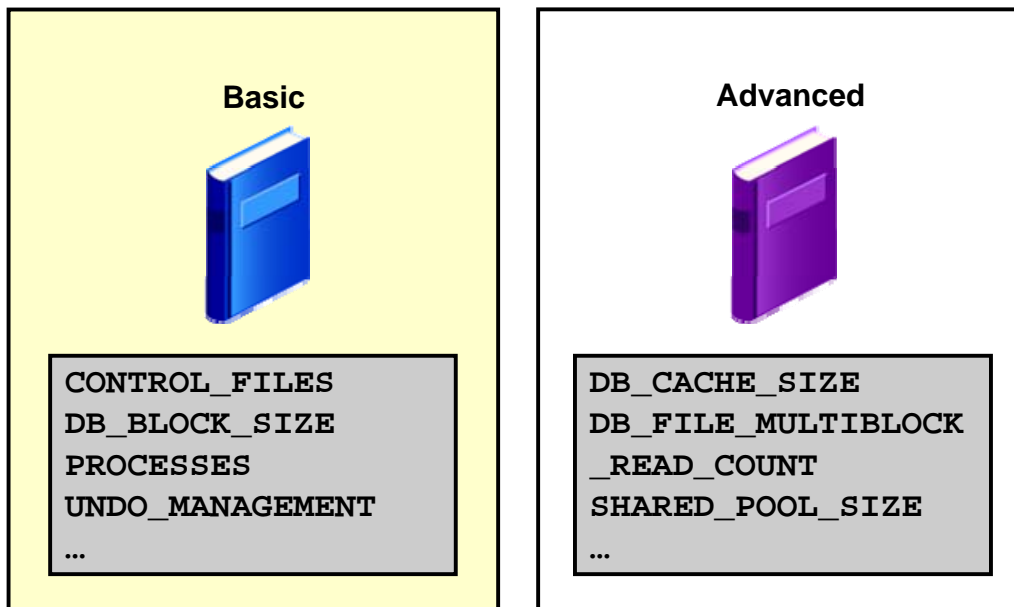
The valid values or value ranges of some initialization parameters depend on the host operating system. For example, the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter specifies the maximum number of blocks that are read in one I/O operation during a sequential scan; this parameter is platform dependent. The size of those blocks, which is set by `DB_BLOCK_SIZE`, has a default value that depends on the operating system.

Setting Parameter Values

Initialization parameters offer the most potential for improving system performance. Some parameters set capacity limits but do not affect performance. For example, when the value of `OPEN_CURSORS` is 10, a user process attempting to open its eleventh cursor receives an error. Other parameters affect performance but do not impose absolute limits. For example, reducing the value of `MAX_OPEN_CURSORS` does not prevent work even though it may slow the performance.

Increasing the values of parameters may improve your system's performance, but increasing most parameters also increases the system global area (SGA) size. A larger SGA can improve database performance up to a point. In virtual memory operating systems, an SGA that is too large can degrade performance if it is swapped in and out of memory. Operating system parameters that control virtual memory working areas should be set with the SGA size in mind. The operating system configuration can also limit the maximum size of the SGA.

Simplified Initialization Parameters



Simplified Initialization Parameters

Initialization parameters are of two types: basic and advanced.

In the majority of cases, it is necessary to set and tune only the 30 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.

Examples of basic parameters:

- Determining the global database name: DB_NAME and DB_DOMAIN
- Specifying a flash recovery area: DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE
- Specifying the DDL lock timeout: DDL_LOCK_TIMEOUT
- Specifying the method of undo space management: UNDO_MANAGEMENT
- COMPATIBLE initialization parameter and irreversible compatibility

Note: Some of the initialization parameters are listed on the following pages. For a complete list, see the *Oracle Database Reference*.

Initialization Parameters: Examples

Parameter	Specifies
<code>CONTROL_FILES</code>	One or more control file names
<code>DB_FILES</code>	Maximum number of database files
<code>PROCESSES</code>	Maximum number of OS user processes that can simultaneously connect
<code>DB_BLOCK_SIZE</code>	Standard database block size used by all tablespaces
<code>DB_CACHE_SIZE</code>	Size of the standard block buffer cache

Initialization Parameters: Examples

CONTROL_FILES parameter: Specifies one or more control file names. Oracle strongly recommends that you multiplex and mirror control files. Range of values: from one to eight file names (with path names). Default range: OS dependent.

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

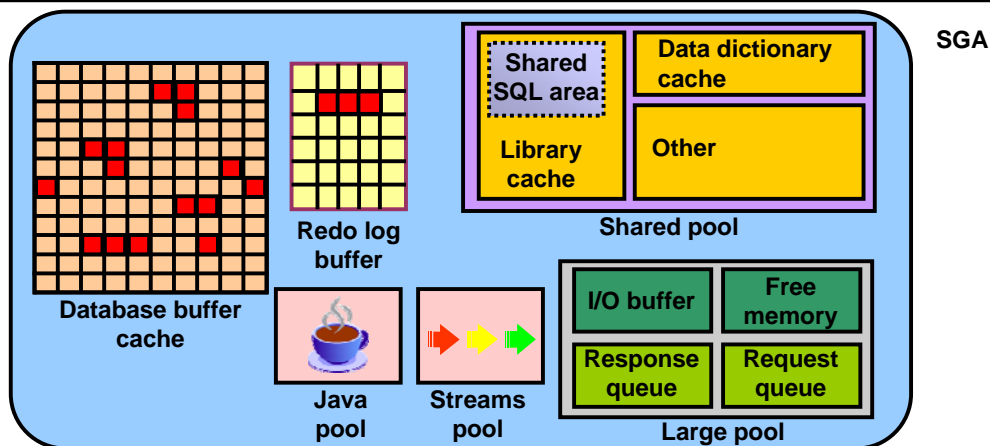
PROCESSES parameter: Specifies the maximum number of OS user processes that can simultaneously connect to an Oracle server. This value should allow for all background processes. Range of values: from 6 to an OS-dependent value. Default value: OS dependent.

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. This specifies the Standard block size for the database. All tablespaces will use this size by default. Range of values: 2048 to 32768 (OS dependent). Default value: 8 KB (OS dependent).

DB_CACHE_SIZE parameter: Specifies the size of the standard block buffer cache. Range of values: at least 16 MB. Default value: 48 MB.

Initialization Parameters: Examples

Parameter	Specifies
SGA_TARGET	Total size of all SGA components
MEMORY_TARGET	Oracle systemwide usable memory



Initialization Parameters: Examples (continued)

SGA_TARGET specifies the total size of all SGA components. If SGA_TARGET is specified, the following memory pools are automatically sized:

- Buffer cache (DB_CACHE_SIZE)
- Shared pool (SHARED_POOL_SIZE)
- Large pool (LARGE_POOL_SIZE)
- Java pool (JAVA_POOL_SIZE)
- Streams pool (STREAMS_POOL_SIZE)

If these automatically tuned memory pools are set to nonzero values, the values are used as minimum levels by Automatic Shared Memory Management (ASMM). You set minimum values if an application component needs a minimum amount of memory to function properly.

The following pools are manually sized components and are not affected by ASMM:

- Log buffer
- Other buffer caches (such as KEEP and RECYCLE) and other block sizes
- Fixed SGA and other internal allocations

The memory allocated to these pools is deducted from the total available memory for SGA_TARGET when ASMM computes the values of the automatically tuned memory pools.

Initialization Parameters: Examples (continued)

`MEMORY_TARGET` specifies the Oracle systemwide usable memory. The database tunes memory to the `MEMORY_TARGET` value, reducing or enlarging the SGA and PGA as needed.

In a text-based initialization parameter file, if you omit `MEMORY_MAX_TARGET` and include a value for `MEMORY_TARGET`, the database automatically sets `MEMORY_MAX_TARGET` to the value of `MEMORY_TARGET`. If you omit the line for `MEMORY_TARGET` and include a value for `MEMORY_MAX_TARGET`, the `MEMORY_TARGET` parameter defaults to zero. After startup, you can then dynamically change `MEMORY_TARGET` to a nonzero value if it does not exceed the value of `MEMORY_MAX_TARGET`. This parameter is modifiable with the `ALTER SYSTEM` command. Values range from 152 MB to `MEMORY_MAX_TARGET`.

Initialization Parameters: Examples

Parameter	Specifies
PGA_AGGREGATE_TARGET	Amount of PGA memory allocated to all server processes
SHARED_POOL_SIZE	Size of shared pool (in bytes)
UNDO_MANAGEMENT	Undo space management mode to be used

ORACLE

4 - 18

Copyright © 2007, Oracle. All rights reserved.

Initialization Parameters: Examples (continued)

PGA_AGGREGATE_TARGET parameter: Specifies the amount of Program Global Area (PGA) memory allocated to all server processes attached to the instance. 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 that is available to the Oracle instance. The range of values comprises integers plus the letters *K*, *M*, or *G* (to specify this limit in kilobytes, megabytes, or gigabytes). The minimum value is 10 MB and the maximum value is 4096 GB. The default is 10 MB or 20% of the size of the SGA, whichever is greater.

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: OS dependent. Default value: 64 MB if 64 bit; 16 MB otherwise.

UNDO_MANAGEMENT parameter: Specifies the undo space management mode that the system should use. When set to *AUTO*, the instance is started in Automatic Undo Management (AUM) mode. Otherwise, it is started in Rollback Undo (RBU) mode. In RBU mode, undo space is allocated externally as rollback segments. In AUM mode, undo space is allocated externally as undo tablespaces. Range of values: *AUTO* or *MANUAL*. If the `UNDO_MANAGEMENT` parameter is omitted when the first instance is started, the default value *AUTO* is used.

Using SQL*Plus to View Parameters

```
SQL> SELECT name , value FROM V$PARAMETER;
NAME                                VALUE
-----
lock_name_space                     2
processes                           150
sessions                             170
timed_statistics                     TRUE
timed_os_statistics                  0
...

SQL> SHOW PARAMETER SHARED_POOL_SIZE
NAME                                TYPE                                VALUE
-----
shared_pool_size                    big integer 0

SQL> show parameter para
NAME                                TYPE                                VALUE
-----
fast_start_parallel_rollback       string                               LOW
parallel_adaptive_multi_user        boolean                              TRUE
parallel_automatic_tuning           boolean                              FALSE
parallel_execution_message_size     integer                              2148
parallel_instance_group             string
...
```

ORACLE

4 - 19

Copyright © 2007, Oracle. All rights reserved.

Using SQL*Plus to View Parameters

The slide shows examples of using SQL*Plus to view parameters. You can query the data V\$PARAMETER dictionary view to find the values of the various parameters. V\$PARAMETER displays the current parameter values in the current session. You can also use the SHOW PARAMETER command with any string to view parameters that contain that string.

The query in the following example is requesting the name and values of the parameters. Use a WHERE clause to specify specific parameter names:

```
SQL> SELECT name, value FROM V$PARAMETER WHERE name LIKE
'%pool%';
```

```
NAME                                TYPE  VALUE
-----
shared_pool_size                    6    0
large_pool_size                     6    0
java_pool_size                      6    0
streams_pool_size                   6    0
shared_pool_reserved_size           6    6710886
buffer_pool_keep                     2
...
```

9 rows selected.

Using SQL*Plus to View Parameters (continued)

Description of the view:

```
SQL> desc V$parameter
Name                                                    Null?    Type
-----
NUM                                                    NUMBER
NAME                                                    VARCHAR2(80)
TYPE                                                    NUMBER
VALUE                                                    VARCHAR2(4000)
DISPLAY_VALUE                                           VARCHAR2(4000)
ISDEFAULT                                               VARCHAR2(9)
ISSES_MODIFIABLE                                       VARCHAR2(5)
ISSYS_MODIFIABLE                                       VARCHAR2(9)
ISINSTANCE_MODIFIABLE                                  VARCHAR2(5)
ISMODIFIED                                              VARCHAR2(10)
ISADJUSTED                                             VARCHAR2(5)
ISDEPRECATED                                           VARCHAR2(5)
ISBASIC                                                 VARCHAR2(5)
DESCRIPTION                                             VARCHAR2(255)
UPDATE_COMMENT                                         VARCHAR2(255)
HASH                                                    NUMBER
```

The second example shows the use of the SQL*Plus `SHOW PARAMETER` command to view parameter settings. You can also use this command to find all parameters that contain a text string (*para*). For example, you can find all parameter names that include the string `db` by using the following command:

```
SQL> show parameter db
NAME                                                    TYPE     VALUE
-----
...
db_8k_cache_size                                       big integer 0
db_block_buffers                                       integer    0
db_block_checking                                       string    FALSE
db_block_checksum                                       string    TYPICAL
db_block_size                                           integer    8192
db_cache_advice                                         string    ON
db_cache_size                                           big integer 0
db_create_file_dest                                     string
...
```

Other Views Containing Information About Parameters

- **V\$SPPARAMETER**: Displays information about the contents of the server parameter file. If a server parameter file was not used to start the instance, each row of the view will contain `FALSE` in the `ISSPECIFIED` column.
- **V\$PARAMETER2**: Displays information about the initialization parameters that are currently in effect for the session, with each parameter value appearing as a row in the view. A new session inherits parameter values from the instance-wide values displayed in the `V$SYSTEM_PARAMETER2` view.

Changing Initialization Parameter Values

- **Static parameters:**
 - Can be changed only in the parameter file
 - Require restarting the instance before taking effect
- **Dynamic parameters:**
 - Can be changed while database is online
 - Can be altered at:
 - Session level
 - System level
 - Are valid for duration of session or based on `SCOPE` setting
 - Are changed by using `ALTER SESSION` and `ALTER SYSTEM` commands

Changing Initialization Parameter Values

There are two types of initialization parameters.

Static parameters: Affect the instance or entire database and can be modified only by editing `init.ora` and the `SPFILE`. Static parameters require the database to be shut down and restarted to take effect. They cannot be changed for the current instance.

Dynamic parameters: Can be changed while database is online. There are two types:

- *Session-level parameters* affect only a user session. Examples include national language support (NLS) parameters that can be used to specify national language settings for sorts, date parameters, and so on. You can use these in a given session; they expire when the session ends.
- *System-level parameters* affect the entire database and all sessions. Examples include flushing the shared pool and setting archive log destinations. These parameters stay in effect based on the `SCOPE` specification. To make them permanent, you have to add these parameter settings to `init.ora` and the `SPFILE` by specifying the `SCOPE =both` option or (manually) editing `init.ora`. If you do the latter, you must restart the database.

Dynamic parameters can be changed by using the `ALTER SESSION` and `ALTER SYSTEM` commands.

Changing Initialization Parameter Values (continued)

Use the `SET` clause of the `ALTER SYSTEM` statement to set or change initialization parameter values. The optional `SCOPE` clause specifies the scope of a change as follows:

- **SCOPE=SPFILE:** The change is applied in the server parameter file only. No change is made to the current instance. For both dynamic and static parameters, the change is effective at the next startup and is persistent. This is the only `SCOPE` specification allowed for static parameters.
- **SCOPE=MEMORY:** The change is applied in memory only. The change is made to the current instance and is effective immediately. For dynamic parameters, the effect is immediate but not persistent because the server parameter file is not updated. For static parameters, this specification is not allowed.
- **SCOPE=BOTH:** The change is applied in both the server parameter file and memory. The change is made to the current instance and is effective immediately. For dynamic parameters, the effect is persistent because the server parameter file is updated. For static parameters, this specification is not allowed.

It is an error to specify `SCOPE=SPFILE` or `SCOPE=BOTH` if the instance did not start up with a server parameter file. The default is `SCOPE=BOTH` if a server parameter file was used to start up the instance, and the default is `MEMORY` if a text initialization parameter file was used to start up the instance.

For dynamic parameters, you can also specify the `DEFERRED` keyword. When it is specified, the change is effective only for future sessions.

When you specify `SCOPE` as `SPFILE` or as `BOTH`, an optional `COMMENT` clause lets you associate a text string with the parameter update. The comment is written to the server parameter file.

Changing Parameter Values: Examples

```
SQL> ALTER SESSION
      SET NLS_DATE_FORMAT = 'mon dd yyyy';
```

Session altered.

```
SQL> SELECT SYSDATE FROM dual;
```

```
SYSDATE
-----
jun 12 2007
```

```
SQL> ALTER SYSTEM SET
      SEC_MAX_FAILED_LOGIN_ATTEMPTS=2 COMMENT='Reduce
      from 10 for tighter security.' SCOPE=SPFILE;
```

System altered.

ORACLE

Changing Parameter Values: Examples

The first statement in the slide is an example of changing a session-level parameter. The user is setting the session date format to be `mon dd yyyy`. As a result, any queries on the date will display dates in that format. Session-level parameters can also be set in applications by using PL/SQL.

The second statement changes the maximum number of failed login attempts before the connection is dropped. It includes a comment and explicitly states that the change is to be made only in the server parameter file. After the specified number of failure attempts, the connection is automatically dropped by the server process.

Database Startup and Shutdown

The screenshot displays the Oracle Enterprise Manager interface for database startup and shutdown. On the left, the 'Startup/Shutdown: Specify Host and Credentials' page is visible, showing fields for Host Credentials (Username: oracle, Password: *****) and Database Credentials (Username: sys, Password: *****). The main area shows two confirmation dialog boxes. The top dialog, titled 'Startup/Shutdown: Confirmation', shows 'Current Status shutdown' and 'Operation startup database in open mode' with an 'Initialization Parameter default'. The bottom dialog, also titled 'Startup/Shutdown: Confirmation', shows 'Current Status open' and 'Operation shutdown immediate'. A red box on the right lists components: SQL*Plus, Init Params, DB Startup, DB Shutdown, Alert Log, and Perf Views. A red line connects the 'DB Startup' component to the top dialog and the 'DB Shutdown' component to the bottom dialog. The Oracle logo is at the bottom right, and the page number '4 - 24' and copyright notice are at the bottom left.

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. You can also click Show SQL to see the SQL statements that are used for startup or shutdown.

Note: The default option to shut down by using EM is IMMEDIATE. The default when issuing the SHUTDOWN command from SQL*Plus is NORMAL.

Starting Up an Oracle Database Instance

The screenshot illustrates the steps to start an Oracle database instance. It features three main components:

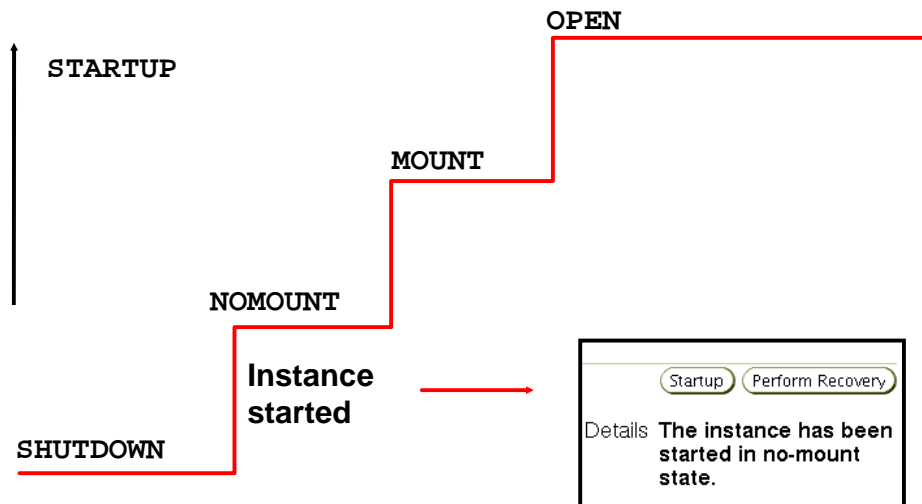
- Database Instance Panel:** Shows the instance status as **Down**. A red arrow points to the **Startup** button. The instance details include Host: `edrsr9p1.us.oracle.com`, Port: `1521`, SID: `orcl`, and Oracle Home: `/u`. A **Details** section notes: "There has been a user-initiated shutdown."
- Startup/Shutdown: Confirmation Dialog:** A modal dialog box with the following text:
 - Current Status: **shutdown**
 - Operation: **startup database in open mode**
 - Initialization Parameter: **default**
 - Question: "Are you sure you want to perform this operation?"
 - Buttons: **Show SQL**, **Advanced Options**, **No**, and **Yes**.
- Startup mode Panel:** A panel with the title **Startup mode** and three radio button options:
 - Start the database
 - Mount the database
 - Open the database

Red lines indicate the flow: from the **Startup** button to the **Confirmation** dialog, and from the **Yes** button to the **Startup mode** panel.

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**. Then enter the host credentials and, optionally, choose the startup mode.

Starting Up an Oracle Database Instance: NOMOUNT



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:

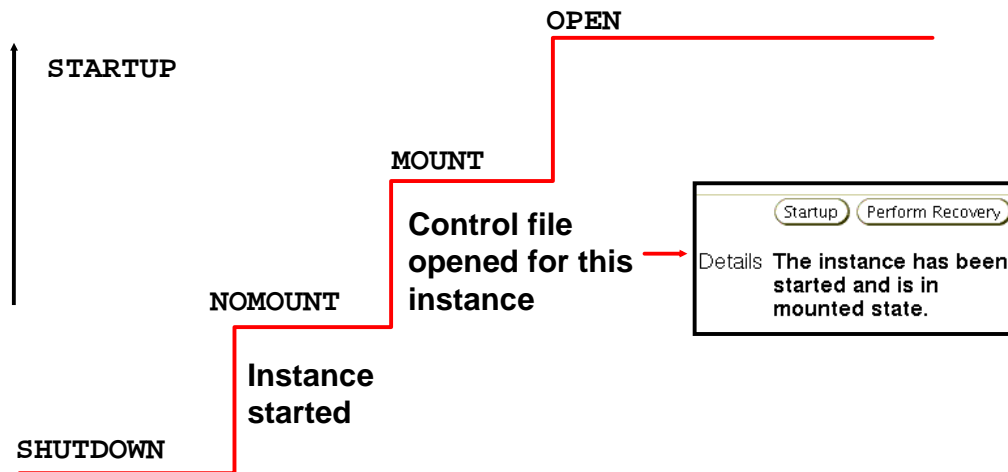
- Searching `$ORACLE_HOME/dbs` for a file of a particular name in this sequence:
 1. Search for `spfile<SID>.ora`.
 2. If `spfile<SID>.ora` is not found, search for `spfile.ora`.
 3. If `spfile.ora` is not found, search for `init<SID>.ora`.

This is the file that contains initialization parameters for the instance. Specifying the `PFIL` 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.)

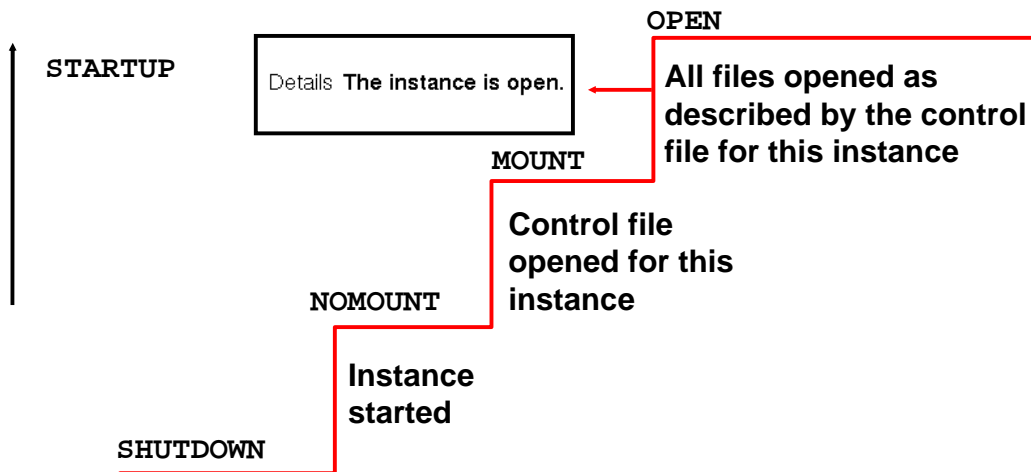
To perform specific maintenance operations, start an instance and mount a database, but do not open the database.

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. If recovery is performed while in the MOUNT state, the redo logs are open for reads and the data files are open as well to read the blocks needing recovery and to write blocks if required during recovery.

Starting Up an Oracle Database Instance: OPEN



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 data files
- Opening the online redo log files

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

During this final stage, the Oracle server verifies that all 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.

Startup Options: Examples

```
SQL> startup
```

1

```
SQL> startup nomount
```

2

```
SQL> alter database mount;
```

3

```
SQL> alter database open;
```

4

ORACLE

Startup Options: Examples

The slide shows the SQL*Plus syntax to start up the database.

1. This command starts the instance, associates the database files to it, and mounts and opens the database
2. This command starts the instance and the database is not mounted
- 3,4. An ALTER DATABASE command mounts and opens the database from the NOMOUNT state.

Shutting Down an Oracle Database Instance

The screenshot displays the Oracle Enterprise Manager Database Control interface for shutting down a database instance. It is divided into several sections:

- General:** Shows the instance status as 'Up' since Jun 21, 2005 1:25:04 PM PDT. Instance Name is 'orcl', Version is '10.2.0.0.0', Host is 'edrsr9p1.us.oracle.com', and Listener is 'LISTENER_edrsr9p1.us.oracle...'. A 'Shutdown' button is visible.
- Components:** A list of components including SQL*Plus, Init Params, DB Startup, **DB Shutdown** (highlighted), Alert Log, and Perf Views.
- Startup/Shutdown: Confirmation:** Shows the current status as 'open' and the operation as 'shutdown immediate'. It asks 'Are you sure you want to perform this operation?' with buttons for 'Show SQL', 'Advanced Options', 'No', and 'Yes'.
- Startup/Shutdown: Advanced Shutdown Options:** Allows specifying the shutdown mode. Options include:
 - Normal [Browse Sessions](#)
 - Wait for all currently connected users to disconnect from the database
 - Transactional: Disconnect all connected users after transactions have completed
 - Immediate: Rollback active transactions and disconnect all connected users
 - Abort: Instantaneous shutdown by aborting the database instance

At the bottom of the page, there is a red banner with the Oracle logo and the text 'ORACLE'. Below the banner, the page number '4 - 30' and the copyright notice 'Copyright © 2007, Oracle. All rights reserved.' are displayed.

Shutting Down an Oracle Database Instance

If the instance is already started when you go to the Enterprise Manager Database Control page, 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 modes:

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

ORACLE

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 mode requires recovery before startup, use it only when necessary. It 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. NORMAL and TRANSACTIONAL can take a long time depending on the number of sessions and transactions.

Shutdown Options

On the way down:

- Uncommitted changes rolled back, for **IMMEDIATE**
- Database buffer cache written to data files
- Resources released

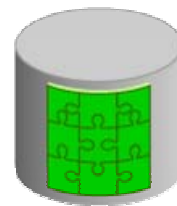
During:

SHUTDOWN
NORMAL
or
SHUTDOWN
TRANSACTIONAL
or
SHUTDOWN
IMMEDIATE

On the way up:

- No instance recovery

**Consistent database
(clean database)**



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 shutdown in **TRANSACTIONAL** mode prevents clients from losing data, including 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: Examples

```
SQL> shutdown
```

```
SQL> shutdown transactional
```

```
SQL> shutdown immediate
```

```
SQL> shutdown abort
```

ORACLE

Shutdown Options (continued)

SHUTDOWN IMMEDIATE

A shutdown in IMMEDIATE mode 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

On the way down:

- **Modified buffers not written to data files**
- **Uncommitted changes not rolled back**



During:

SHUTDOWN ABORT
or
Instance failure
or
STARTUP FORCE

On the way up:

- **Online redo log files used to reapply changes**
- **Undo segments used to roll back uncommitted changes**
- **Resources released**

**Inconsistent database
(dirty database)**

ORACLE

Shutdown Options (continued)

SHUTDOWN ABORT

If shutdown in `NORMAL` and `IMMEDIATE` modes does 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 who are 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

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

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

STARTUP [NOMOUNT | MOUNT | OPEN (open options)] [FORCE]
[RESTRICT]
```

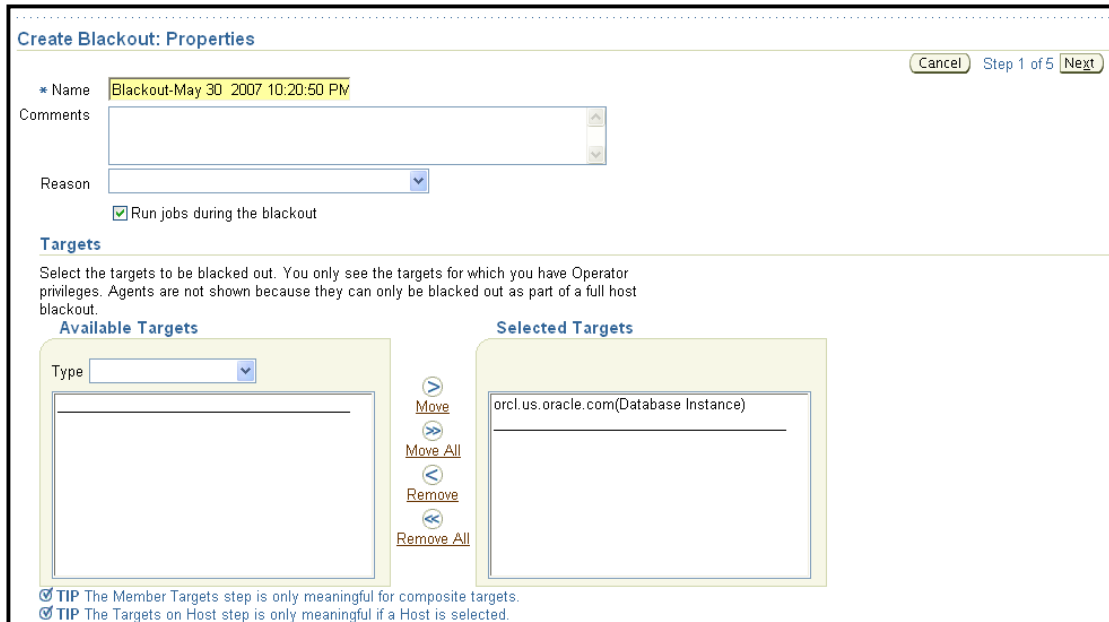
This enables you to include startup and shutdown operations as part of a script or batch process that performs tasks on the database (where the database needs to be in a particular state).

Note: The open options enable you to specify the mode of access in which you want the specified database to start. Possible modes are:

- READ ONLY
- READ WRITE

You can also use the PFILE option for STARTUP. This specifies the PFILE initialization parameter file to be used when the database instance is started.

Blackout Administration



Create Blackout: Properties Cancel Step 1 of 5 Next

* Name: Blackout-May 30 2007 10:20:50 PM

Comments: [Text Area]

Reason: [Dropdown]

Run jobs during the blackout

Targets

Select the targets to be blacked out. You only see the targets for which you have Operator privileges. Agents are not shown because they can only be blacked out as part of a full host blackout.

Available Targets **Selected Targets**

Type: [Dropdown]

orcl.us.oracle.com(Database Instance)

> Move
>> Move All
< Remove
<< Remove All

TIP The Member Targets step is only meaningful for composite targets.
TIP The Targets on Host step is only meaningful if a Host is selected.

Blackout Administration

Blackouts suspend the collection of database monitoring data and the sending of alerts by Database Control. This enables you to perform scheduled maintenance on the database without receiving needless alerts and without skewing the monitoring data. For example, you can stop data collections during a database backup or a hardware upgrade. If you continue monitoring during these periods, the collected data will show trends and other monitoring information that are not the result of normal day-to-day operations. To get a more accurate, long-term picture of database performance, you can use blackouts to exclude these special-case situations from data analysis.

To define a blackout:

1. Click Setup at the top of any Database Control page. The Enterprise Manager Configuration page appears, showing the “Overview of Setup” page.
2. In the left-hand pane, click Blackouts. The Blackouts page appears.
3. Click Create to start the Create Blackout Wizard. The Create Blackout: Properties page appears.
4. You can replace the default blackout name with a name of your choosing. In the Comments field, enter text that describes the purpose of the blackout.

Blackout Administration (continued)

5. In the Reason list, select the blackout reason that is most appropriate.
6. In the Type list in the Available Targets section, select Database Instance. Your database instance ID (SID) appears in the Available Targets list.
7. In the Available Targets list, select your instance ID and then click the Move icon. Click Next. The Create Blackout Schedule page appears.
8. In the Start section, schedule the blackout to start either immediately or at a later date and time.
9. In the Duration section, indicate the duration of the blackout.
10. To repeat the blackout periodically, select a repeat frequency from the Repeat list in the Repeating section. Click Next. The review page appears.
11. Review what you have entered. You can click Back to change a setting. Click Finish. The Confirmation page appears, with the new blackout period shown in the list.

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

Jul 25, 2007 9:34:52 PM GMT+07:00	NOTIFICATION	16	admin_ddl@plex:2924:4222364190	ALTER DATABASE OPEN
Jul 25, 2007 9:34:52 PM GMT+07:00	NOTIFICATION	16	admin_ddl@plex:2995:2802784106	Completed: ALTER DATABASE MOUNT
Jul 25, 2007 9:34:52 PM GMT+07:00	UNKNOWN	16		Lost write protection disabled
Jul 25, 2007 9:34:52 PM GMT+07:00	UNKNOWN	16		Successful mount of redo thread 1, with mount id 1156682488
Jul 25, 2007 9:34:52 PM GMT+07:00	UNKNOWN	16		Database mounted in Exclusive Mode
Jul 25, 2007 9:34:52 PM GMT+07:00	UNKNOWN	16		Setting recovery target incarnation to 2
Jul 25, 2007 9:34:45 PM GMT+07:00	NOTIFICATION	16	admin_ddl@plex:2924:4222364190	ALTER DATABASE MOUNT
Jul 25, 2007 9:34:45 PM GMT+07:00	NOTIFICATION	16	startup ksu_setup_oracle_base:21703:2787919602	ORACLE_BASE from environment = /u01/app/oracle
Jul 25, 2007 9:34:45 PM GMT+07:00	UNKNOWN	16		starting up 1 shared server(s) ...

ORACLE

4 - 38

Copyright © 2007, Oracle. All rights reserved.

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 `$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/trace` by default if `$ORACLE_BASE` is set.

The alert file of a database is a chronological log of messages such as 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

Oracle Database uses the alert log to keep a record of these events as an alternative to displaying the information on an operator's console. (Many systems also display this information on the console.) If an administrative operation is successful, a message is written in the alert log as "completed" along with a time stamp.

Viewing the Alert Log (continued)

EM monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and information messages. Because the file can grow to an unmanageable size, you can periodically 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.

Note: There is an XML version of the alert log in the `$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/alert` directory.

To view the alert log with a text editor:

- Connect to the database with SQL*Plus (or another query tool such as SQL Developer).
- Query the `V$DIAG_INFO` view.

To view the text-only alert log without the XML tags:

- In the `V$DIAG_INFO` query results, note the path that corresponds to the Diag Trace entry. Change directory to that path.
- Open the `alert_SID.log` file with a text editor.

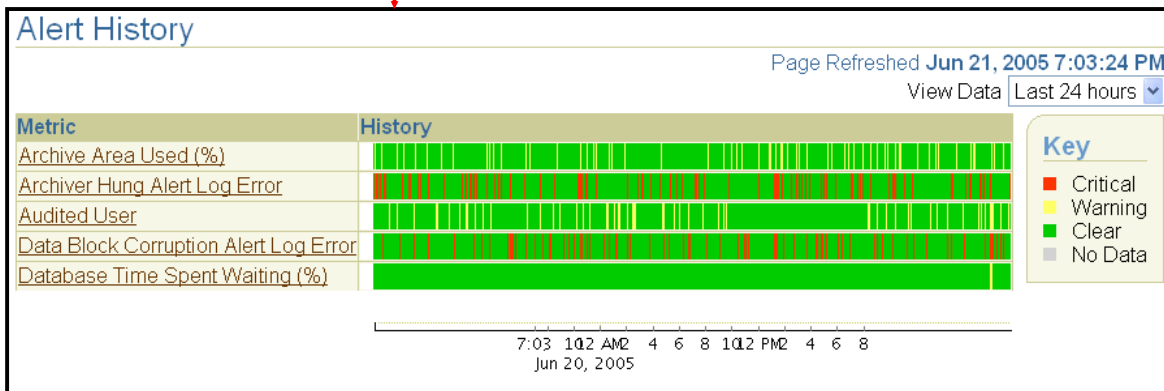
To view the XML-formatted alert log:

- In the `V$DIAG_INFO` query results, note the path that corresponds to the Diag Alert entry. Change directory to that path.
- Open the `log.xml` file with a text editor.

Viewing the Alert History

Related Links

Advisor Central	Alert History	Alert Log Content
All Metrics	Blackouts	iSQL*Plus
Jobs	Manage Metrics	Metric Baselines
Metric Collection Errors	Monitoring Configuration	Monitor in Memory Access Mode
Recovery Catalogs	SQL History	User-Defined Metrics



ORACLE

Viewing the Alert History

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

Using Trace Files

- **Each server and background process can write to an associated trace file.**
- **Error information is written to the corresponding trace file.**
- **Automatic diagnostic repository (ADR)**
 - **Is a systemwide central tracing and logging repository**
 - **Stores database diagnostic data such as:**
 - **Traces**
 - **Alert log**
 - **Health monitor reports**

Using Trace Files

Each server and background process can write to an associated trace file. When a process detects an internal error, it dumps information about the error to its trace file. If an internal error occurs and information is written to a trace file, the administrator should contact Oracle Support Services.

All file names of trace files associated with a background process contain the name of the process that generated the trace file. The one exception to this is trace files that are generated by job queue processes (Jnnn).

Additional information in trace files can provide guidance for tuning applications or an instance. Background processes always write this information to a trace file when appropriate.

Beginning with Oracle Database 11g, an advanced fault diagnosability infrastructure is included for preventing, detecting, diagnosing, and resolving problems. In particular, problems that are targeted include critical errors such as those caused by database code bugs, metadata corruption, and customer data corruption.

Using Trace Files (continued)

When a critical error occurs, an incident number is assigned to it; diagnostic data for the error (such as trace files) is immediately captured and tagged with this number. The data is then stored in the automatic diagnostic repository (ADR)—a file-based repository outside the database—where it can later be retrieved by incident number and analyzed.

The ADR is a systemwide tracing and logging central repository for database diagnostic data such as traces, the alert log, health monitor reports, and more.

The ADR root directory is known as *ADR base*. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter. If this parameter is omitted or left null, the database sets `DIAGNOSTIC_DEST` upon startup as follows:

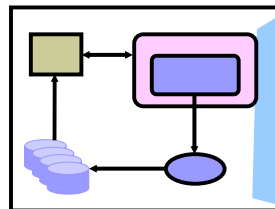
- If the `ORACLE_BASE` environment variable is set, `DIAGNOSTIC_DEST` is set to the directory designated by `ORACLE_BASE`.
- If the `ORACLE_BASE` environment variable is not set, `DIAGNOSTIC_DEST` is set to `ORACLE_HOME/log`.

The location of an ADR home is given by the following path, which starts at the ADR base directory:

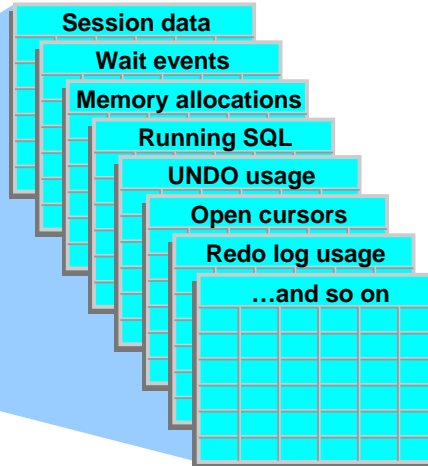
```
diag/product_type/product_id/instance_id
```

Dynamic Performance Views

Provide access to information about changing states of the instance memory structures



Oracle instance



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

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 `DICTIONARY` and `DICTIONARY_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 - 44

Copyright © 2007, Oracle. All rights reserved.

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:

- For which SQL statements (and their associated numbers of executions) is the CPU time consumed greater than 200,000 microseconds?
- Which current sessions are logged in from the EDRSR9P1 computer on the last day?
- What are the session IDs of those sessions that are currently holding a lock that is blocking another user, and how long have those locks 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.**

ORACLE

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.

Some `V$` views contain information that is similar to information in the corresponding `DBA_` views. For example, `V$DATAFILE` is similar to `DBA_DATAFILES`. Note also that `V$` view names are generally singular and `DBA_` view names are plural.

Data Dictionary: Overview

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

```
SELECT * FROM dictionary;
```

4 - 46 Copyright © 2007, Oracle. All rights reserved.

Data Dictionary: Overview

The Oracle data dictionary is the description of a database and 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*.

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.)

ORACLE

4 - 47

Copyright © 2007, Oracle. All rights reserved.

Data Dictionary Views

The view prefixes indicate the data (and how much of that 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 the user 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 type of view shows only those objects that the querying user owns (that is, those that are present in the user's own schema).

Data Dictionary Views (continued)

Generally, each view set is a subset of the higher-privileged view set, row-wise and column-wise. 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.

Be sure 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;`

Data Dictionary: Usage Examples

The example queries in the slide answer the following questions:

- What are the names of the tables (along with the name of the tablespace where they reside) that have been created in your schema?
- What is the significant information about the sequences in the database that you have access to?
- What users in this database are currently able to log in?
- 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:

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

ORACLE

Practice 4 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**

ORACLE

5

Configuring the Oracle Network Environment

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

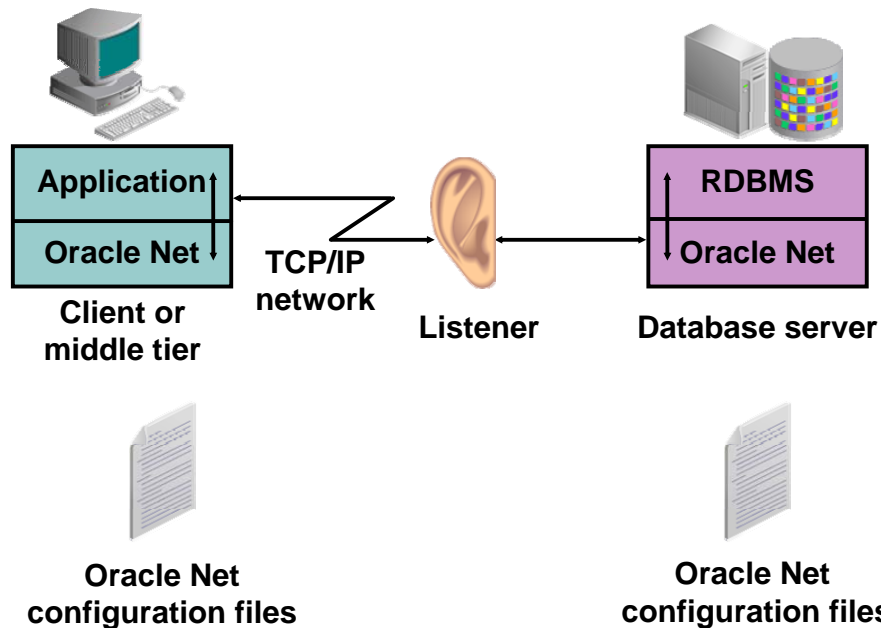
After completing this lesson, you should be able to:

- **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 and when to use dedicated servers**

Resources

- *Oracle Database Net Services Administrator's Guide 11g Release 1*
- *Oracle Database Net Services Reference 11g Release 1*

Oracle Net Services



5 - 3

Copyright © 2007, Oracle. All rights reserved.

Oracle Net Services

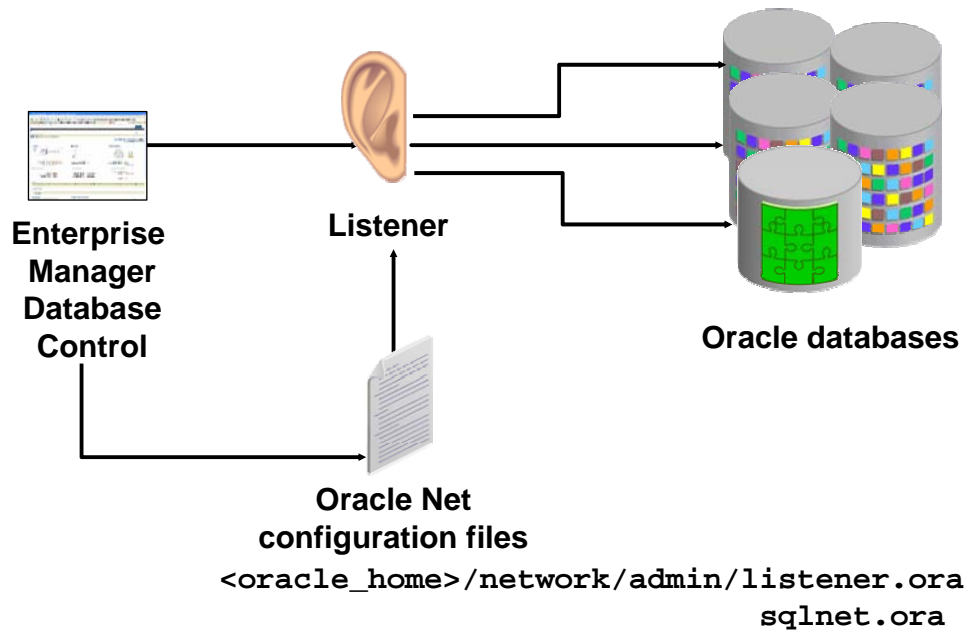
Oracle Net Services enables network connections from a client or middle-tier application to the Oracle server. After a network session is established, Oracle Net acts as the data courier for both the client application and the database server. It is responsible for establishing and maintaining the connection 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) 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 *Oracle Net Listener*, which 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 Net Listener

The Oracle Net Listener (or simply *the 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.

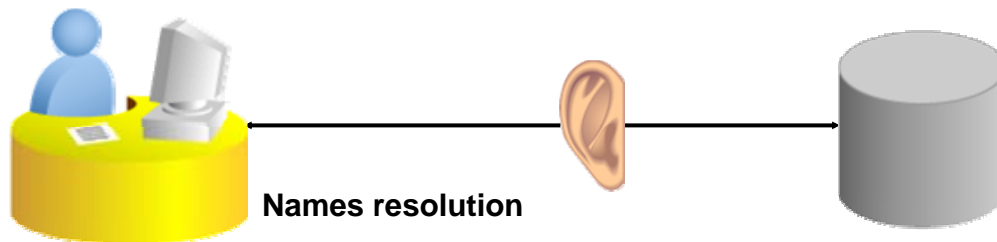
Enterprise Manager is one of the ways to access the listener. You can control the configuration of the actual listener as well as general parameters such as password protection and log file locations.

Advanced administrators can also configure Oracle Net Services by manually editing the configuration files, if necessary, with a standard operating system (OS) text editor such as `vi` or `gedit`.

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

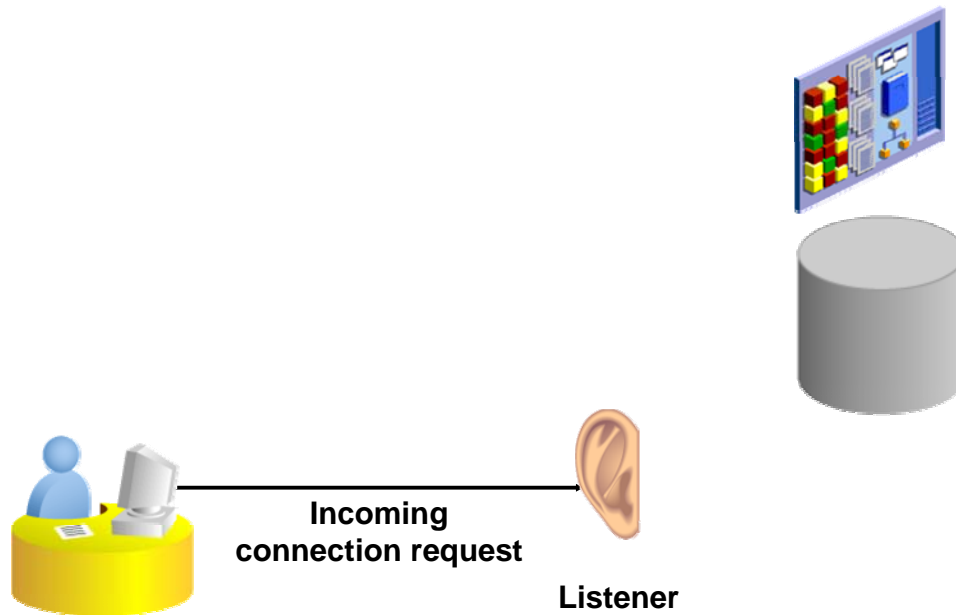


Establishing Net Connections

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 to which it wants to connect.

Oracle Net Names Resolution is the process of determining this connection information.

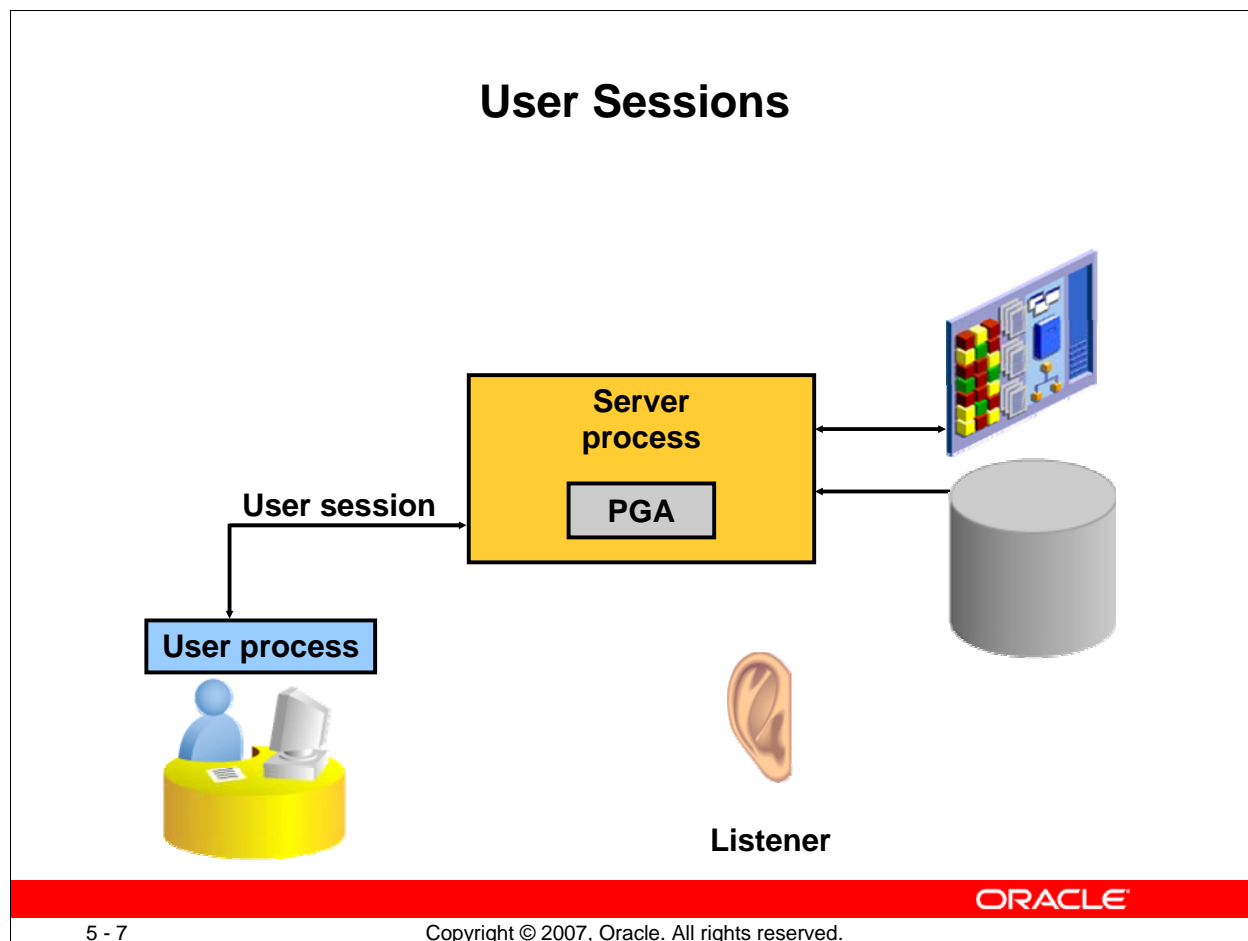
Establishing a Connection



Establishing a Connection

After Oracle Net Names Resolution is complete, a connection request is passed from the user or middle-tier application (hereafter referred to as the *user process*) to the 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

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 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 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. The Sort Area is a memory area that is used to work with sorting; it is contained in a portion of memory that is associated with the Program Global Area (PGA).
- 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**



ORACLE

5 - 8

Copyright © 2007, Oracle. All rights reserved.

Tools for Configuring and Managing the Oracle Network

Use the following tools and 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 Manager enables you to configure Oracle Net Services for an Oracle home on a local client or server host. You can use Oracle Net Manager to configure the following network components:
 - **Naming:** Define simple names, connect identifiers, and map them to connect descriptors to identify the network location and identification of a service. Oracle Net Manager supports configuration of connect descriptors in local `tnsnames.ora` files or a centralized directory service.
 - **Naming methods:** Configure the different ways in which connect identifiers are resolved into connect descriptors
 - **Profiles:** Configure preferences for enabling and configuring Oracle Net features on the client or server
 - **Listeners:** Create and configure listeners to receive client connections

Tools for Configuring and Managing the Oracle Network (continued)

- **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. During a typical database installation, Oracle Net Configuration Assistant automatically configures a listener called `LISTENER` that has a TCP/IP listening protocol address for the database. If you do a custom installation, Oracle Net Configuration Assistant prompts you to configure a listener name and protocol address of your choice. Use the Oracle Net Configuration Assistant for initial network configuration after database installation. Thereafter, you can use the Oracle Enterprise Manager and Oracle Net Manager to configure and administer your networks.
- **Command line:** Is used to start, stop, and view the status of the listener process. It is an OS user (in this course, `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 `lsnrctl` command-line utility (or from EM).

```
[oracle@edrsr17p1 ~]$ lsnrctl
LSNRCTL for Linux: Version 11.1.0.3.0 - Beta on 30-MAY-2007 22:38:19
Copyright (c) 1991, 2006, 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*
```

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:

```
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 from the command line. The following two commands have the same effect but use command-line syntax and prompt syntax, respectively:

Command-line syntax:

```
$ lsnrctl start
```

Prompt syntax:

```
$ lsnrctl
LSNRCTL for Linux: Version 11.1.0.3.0 - Beta on 22-JUN-2007
01:35:30
Copyright (c) 1991, 2006, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> start
```

The command-line syntax is typically used to execute an individual command or scripted commands. If you plan to execute several consecutive `lsnrctl` commands, the prompt syntax is more efficient. Note that the `listener_name` argument is omitted, and the stop command would thus 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 that your listener is named flovr_lis. Here are two examples of stopping a listener named flovr_lis by using prompt syntax:

```
LSNRCTL> stop flovr_lis
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rhel)(PORT=5521)))
The command completed successfully
```

This produces the same results as the following:

```
LSNRCTL> set cur flovr_lis
Current Listener is flovr_lis
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 to cur.

Using command-line syntax produces the same results:

```
/home/oracle> lsnrctl stop flovr_lis
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

Database Instance: orcl.us.oracle.com

Home Performance Availability Server

Latest Data Collected

General

Shutdown Black Out

Status Up

Up Since **May 10, 2007 10:21:07 AM GMT+07:00**

Instance Name **orcl**

Version **11.1.0.3.0**

Host **edrsr17p1.us.oracle.com**

Listener **LISTENER_edrsr17p1.us.orac...**

General

Edit Stop Black Out

Status **Up**

Availability (%) **4** (Last 24 Hours)

Alias **LISTENER**

Version **11.1.0.3.0**

Oracle Home **/u01/app/oracle/product/11.1.0/db_1**

Net Address **(ADDRESS=(PROTOCOL=TCP)(HOST=edrsr17p1.us.oracle.com)(PORT=1521))**

LISTENER.ORA Location **/u01/app/oracle/product/11.1.0/db_1/network/admin**

Start Time **May 10, 2007 10:20:58 AM**

Host **edrsr17p1.us.oracle.com**

State

TNS Ping (ms) **10**

Established Connections per minute **53**

Refused Connections per minute **0**

Policy Violations

Current **230** Distinct Rules Violated **230** Compliance Score (%) **93** [Policy Trend Overview](#)

Home Performance Serviced Databases

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:

- Listener status and availability in the last 24 hours
- Listener version and Oracle home
- First listening address for the listener
- Location of the configuration files that are used to start the listener
- 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 who can start and stop the listener.

Net Services Administration Pages

ORACLE Enterprise Manager 11g Database Control

Host: edrsr17p1.us.oracle.com >

Net Services Administration

Net Services Administration allows you to configure or administer the following network components:

- Listener: Allows configuration and administration functions on listeners.
- Directory Naming: Allows configuration and administration of Net service names on a Directory server.
- Local Naming: Allows configuration and administration of Net service names on a client's tnsnames.ora file.
- Network Profile: Allows configuration of preferences for Oracle Net Services features on the client or server.

Choose a configuration file location, then select the feature that you want to administer and click 'Go'.

Administer Listeners Go

Select Configuration File Location	Oracle Home
<input type="radio"/> /u01/app/oracle/product/11.1.0/db_1/network/admin	<input type="radio"/> /u01/app/oracle/product/11.1.0/db_1

Net Services Administration: Host Login

Host: edrsr17p1.us.oracle.com

Oracle Home: /u01/app/oracle/product/11.1.0/db_1

* Username: oracle

* Password: ●●●●●●

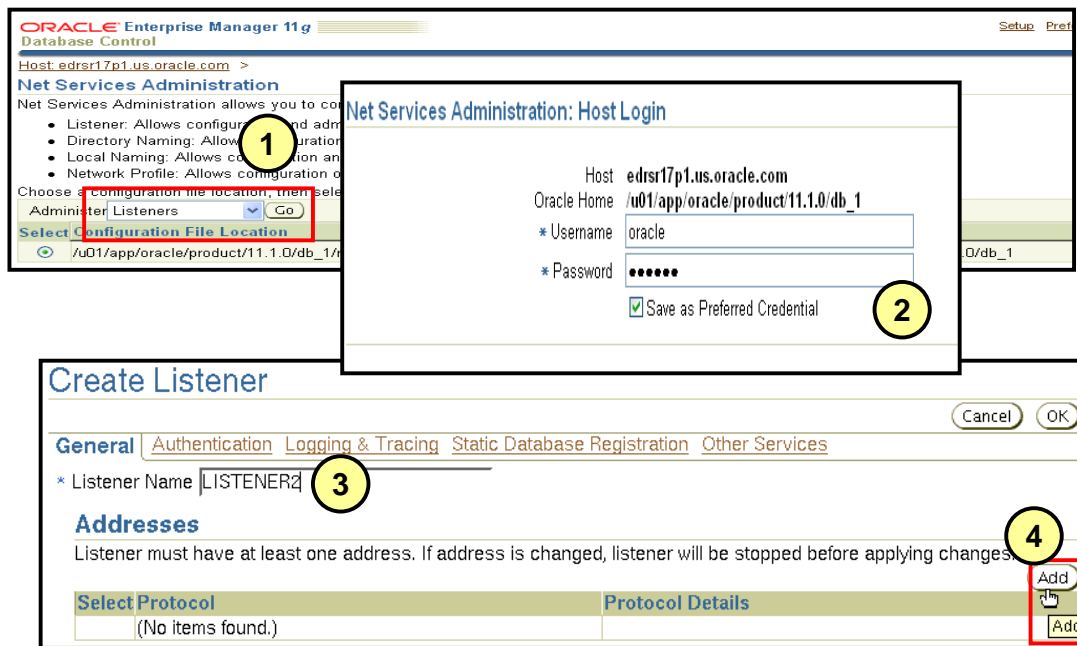
Save as Preferred Credential

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:** 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. Then 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

Add Address

Protocol: TCP/IP (5)
* Port: 1561 (6)
* Host: edrsr30p1.us.oracle.com (7)
The host name or IP address of the computer.

Advanced Parameters
The following parameters are introduced in Oracle version 10g.
Total Send Buffer Size (Bytes): Cumulative size for all send operations.
Total Receive Buffer Size (Bytes): Cumulative size for all receive operations.

Create Listener

General | Authentication | Logging & Tracing | Static Database Registration | Other Services

* Listener Name: LISTENER2

Addresses
Listener must have at least one address. If address is changed, listener will be stopped before applying changes.

Buttons: Add, Edit, Remove

Select	Protocol	Protocol Details
<input checked="" type="checkbox"/>	TCP/IP	Host: edrsr30p1.us.oracle.com Port: 1561

Adding Listener Addresses

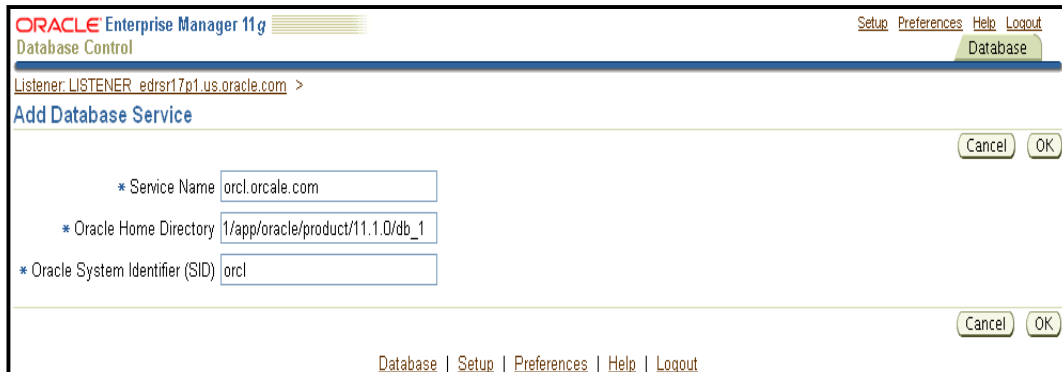
The workflow to create a listener continues:

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, 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 then click Go.

Database Service Registration



The screenshot shows the Oracle Enterprise Manager 11g Database Control interface. The main window title is "ORACLE Enterprise Manager 11g Database Control". The breadcrumb navigation shows "Listener LISTENER_edrsr17p1.us.oracle.com >". The "Add Database Service" dialog box is open, with the following fields:

- * Service Name: orcl.oracle.com
- * Oracle Home Directory: f:/app/oracle/product/11.1.0/db_1
- * Oracle System Identifier (SID): orcl

There are "Cancel" and "OK" buttons at the bottom right of the dialog box. The bottom of the window shows the navigation menu: "Database | Setup | Preferences | Help | Logout".

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:** Oracle8i and later instances automatically register with the default listener on database startup. No additional listener configuration is required for the default listener.
- **Static service registration:** 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 then 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. For the changes to take effect, you must reload (use the RELOAD command) or restart your listener.

Database Service Registration (continued)

Service Names

The `SERVICE_NAMES` initialization parameter specifies one or more names by which clients can connect to the instance. The instance registers its service names with the listener. When a client requests a service, the listener determines which instances offer the requested service and routes the client to the appropriate instance.

You can specify multiple service names to distinguish among different uses of the same database, as in this example:

```
SERVICE_NAMES = sales.acme.com, eurosales.acme.com
```

You can also use service names to identify a single service that is available from two different databases through the use of replication.

If you do not qualify the names in this parameter with a domain, Oracle qualifies them with the value of the `DB_DOMAIN` parameter. If `DB_DOMAIN` is not specified, no domain will be applied to the nonqualified `SERVICE_NAMES` values.

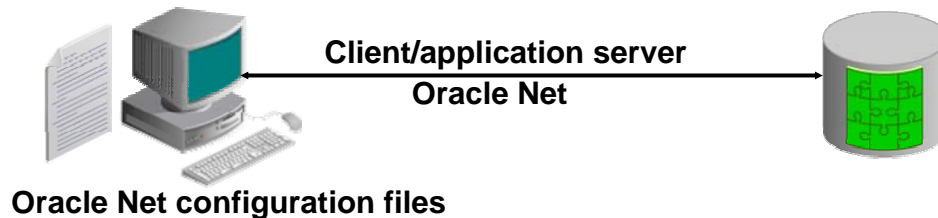
In Oracle Net Manager, the Service Name is the `GLOBAL_DBNAME` initialization parameter. This parameter identifies the database service.

While processing a client connection request, the listener tries to match the value of this parameter with the value of the `SERVICE_NAME` parameter in the client connect descriptor. If the client connect descriptor uses the `SID` parameter, the listener does not attempt to map the values. The value for this parameter is typically obtained from the combination of the `DB_NAME` and `DB_DOMAIN` parameters (`DB_NAME.DB_DOMAIN`) in the initialization parameter file, but the value can also contain any valid name used by clients to identify the service.

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



ORACLE

5 - 19

Copyright © 2007, Oracle. All rights reserved.

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:

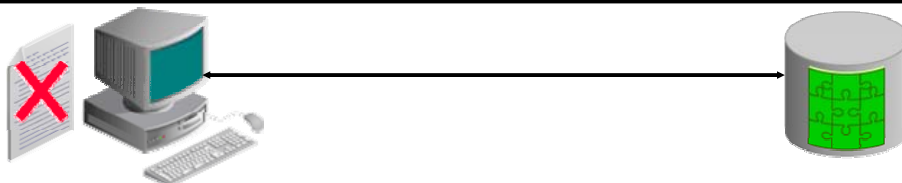
```
CONNECT username/password@host[:port][/]service_name]
```

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:** To access a database service, the directory naming method stores connect identifiers in a centralized directory server that is compliant with the Lightweight Directory Access Protocol (LDAP).
- **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/dba11g
```



No Oracle Net configuration files

ORACLE

5 - 20

Copyright © 2007, Oracle. All rights reserved.

Easy Connect

With Easy Connect, you supply all information that is required for the Oracle Net connection as part of the connect string. Easy Connect connection strings take the following form:

```
<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.

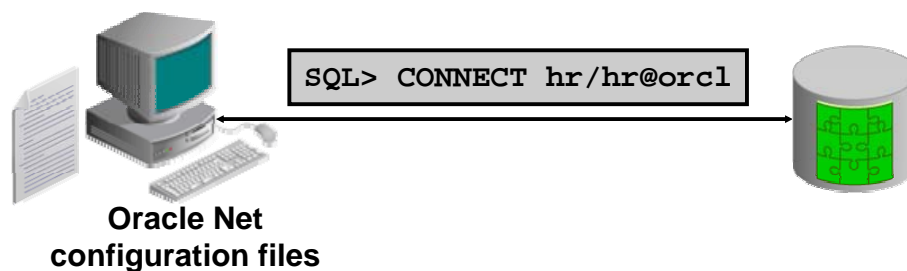
Assuming that the listener uses TCP to listen on port 1521 and the `SERVICE_NAMES=db` and `DB_DOMAIN=us.oracle.com` instance parameters, the connect string shown in the slide can be shortened:

```
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



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 following text configuration file:

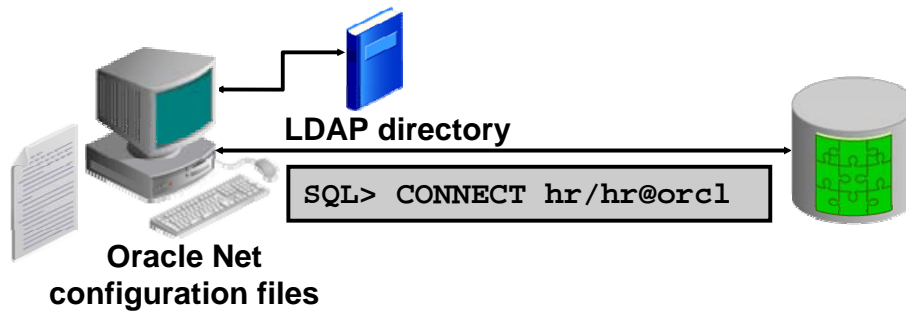
```
<oracle_home>/network/admin/tnsnames.ora
```

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 in which 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

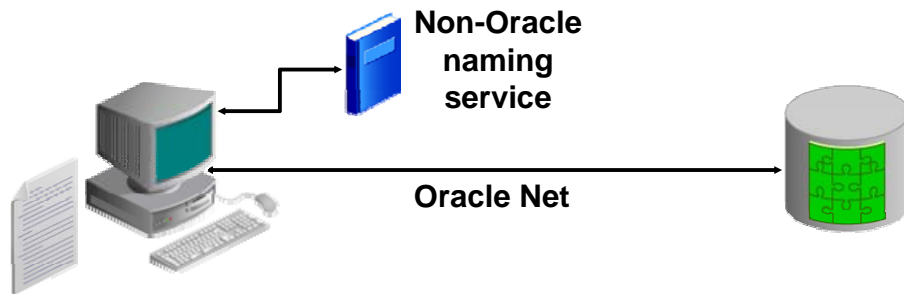
With directory naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against an external list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name. Like local naming, database users need to remember only a short alias.

One advantage of directory naming is that the service name is available for users to connect with as soon as a new service name is added to the LDAP directory. 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 in which 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)**



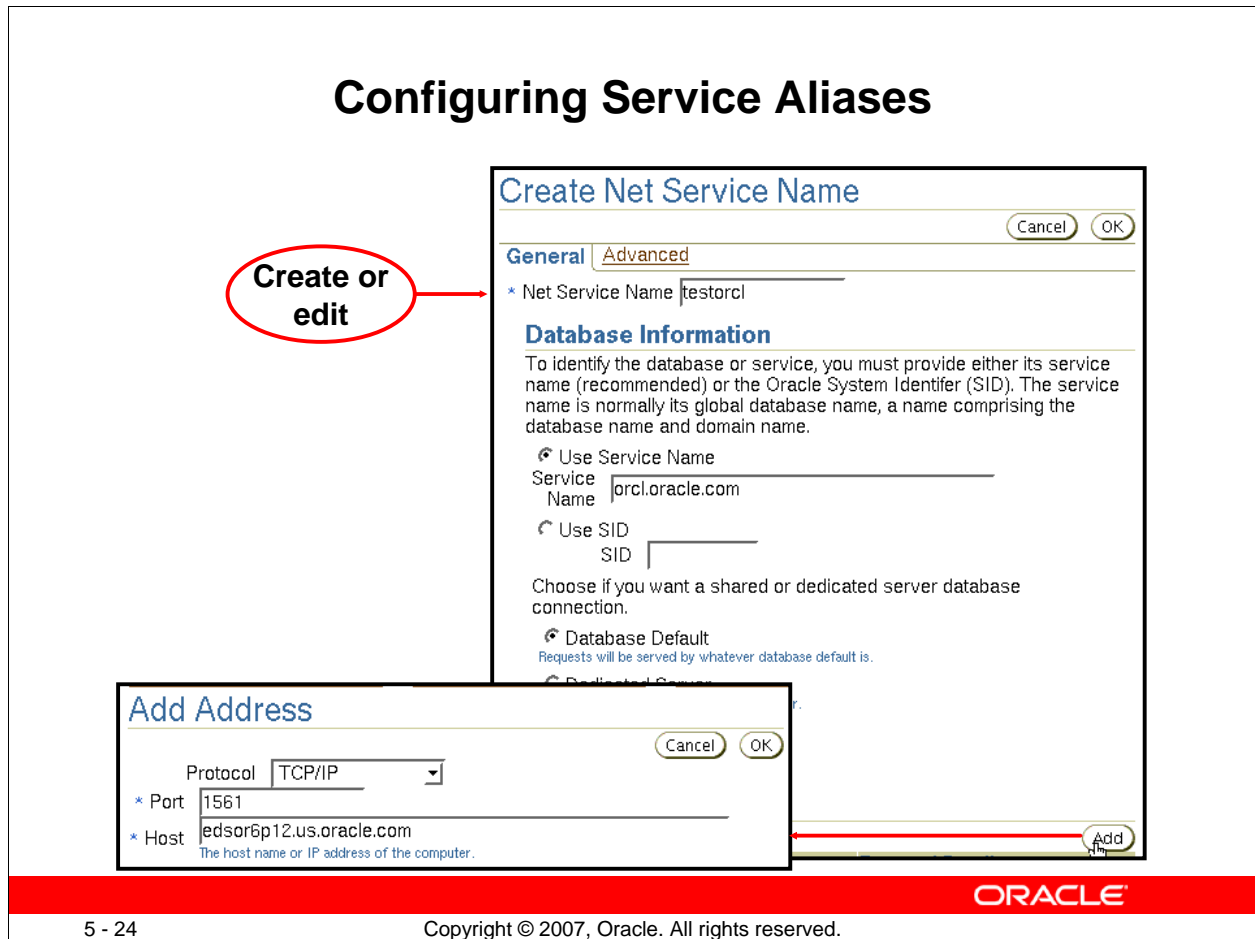
External Naming Method

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



Configuring Service Aliases

To create a local Oracle Net Service alias, select Local Naming in the Administer drop-down list and click Go. 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 to which you want to connect.

Advanced Connection Options

Oracle Net supports the following advanced connection options with local and directory naming:

- **Connect-time failover**
- **Load balancing**
- **Source routing**

Connect-time Failover and Client Load Balancing

Configure whether addresses are tried randomly or sequentially during connections to the service. This setting is applicable only if there are more than one addresses configured.

- Try each address, in order, until one succeeds
- Try each address randomly, until one succeeds
- Try one address, selected at random
- Use each address in order until destination is reached
- Use only the first address

Advanced Connection Options

When a database service is accessible by multiple listener protocol addresses, you can specify the order in which the addresses are to be used. The addresses can be chosen randomly or tried sequentially. In cases in which more than one listener is available, such as Oracle Real Application Clusters (RAC) configurations, 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. Transparent Application Failover (TAF) is a client-side feature that allows clients to reconnect to surviving databases in the event of a database instance failure. Notifications are used by the server to trigger TAF callbacks on the client-side.

With *load balancing* enabled, Oracle Net picks an address at random from the list of addresses. The run-time connection load-balancing feature improves connection performance by balancing the number of active connections among multiple dispatchers. In a RAC environment, connection pool load balancing also has the capability to balance the number of active connections among multiple instances.

Advanced Connection Options (continued)

Source routing is used with Oracle Connection Manager, which 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.

Note that there are five choices for connect-time failover and load balancing:

Option	Advanced Functionality
Try each address in sequence 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 sequence 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/dba11g
```

- Supports local and directory naming:

```
tnsping orcl
```

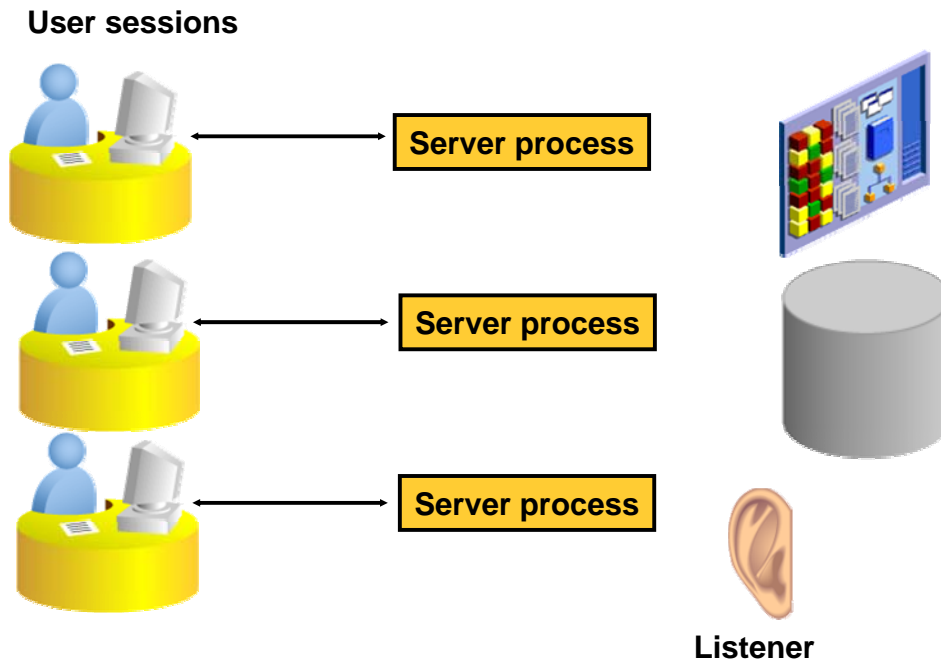
ORACLE

Testing Oracle Net Connectivity

`tnsping` is the Oracle Net equivalent of the TCP/IP ping utility. It offers a quick test to verify that the network path 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. The `tnsping` utility also reveals the location of the configuration files. In a system with multiple `ORACLE_HOME` locations, this can be helpful.

User Sessions: Dedicated Server



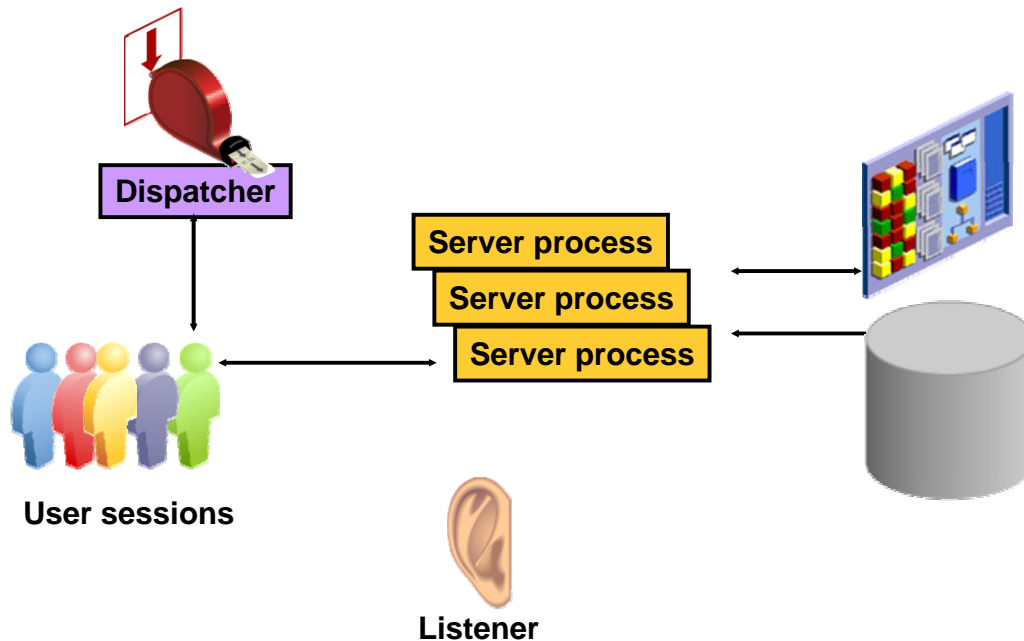
User Sessions: Dedicated Server

With dedicated server processes, there is a one-to-one ratio of server processes to 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 affected by the resource demands of the dedicated server architecture, you have the following 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 dispatcher process (and usually more). 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.

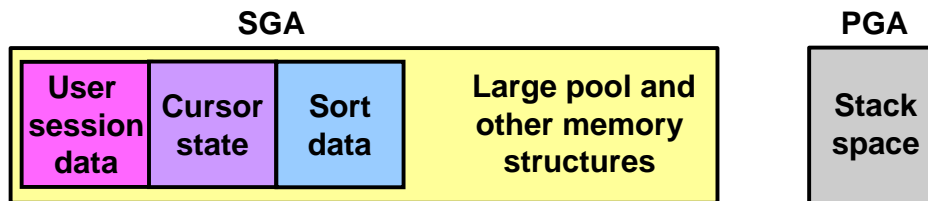
Dispatchers do not actually handle the work of user requests. Instead, they pass user requests to a common queue located in the shared pool portion of the SGA.

Shared server processes take over most of the work of dedicated server processes, pulling requests from the queue and processing them until they are 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 if `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 consider shared server memory requirements when sizing the SGA.

SGA and PGA

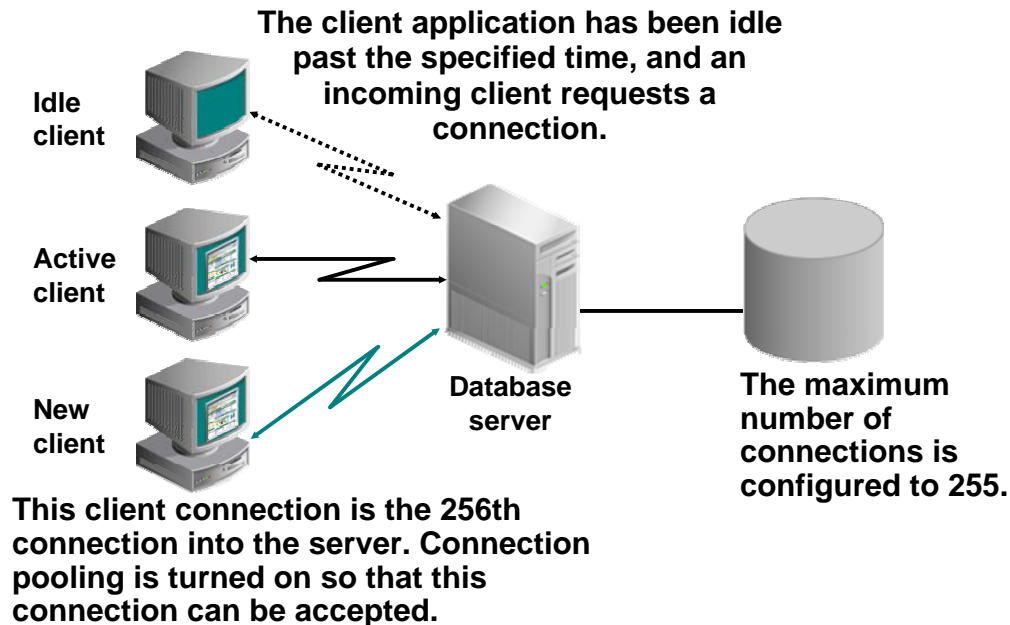
The contents of the SGA and PGA differ when dedicated servers or shared servers are used:

- Text and parsed forms of all SQL statements are stored in the SGA.
- The cursor state contains run-time memory values for the SQL statement, such as rows retrieved.
- User-session data includes security and resource usage information.
- The stack space contains local variables for the process.

Technical Note

The change in the SGA and PGA is transparent to the user; however, if 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 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 using shared servers:

- **Database administration**
- **Backup and recovery operations**
- **Batch processing and bulk load operations**
- **Data warehouse operations**



Dispatcher



**Dedicated
server process**

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 and 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.

Configuring Communication Between Databases

- **Sending data or messages between sites requires network configuration on both sites.**
- **You must configure the following:**
 - Network connectivity (for example, `TNSNAMES.ora`)
 - Database links

```
CREATE DATABASE LINK <remote_global_name>  
CONNECT TO <user> IDENTIFIED BY <pwd>  
USING '<connect_string_for_remote_db>';
```

Configuring Communication Between Databases

A database link is a schema object in one database that enables you to access objects on another database. The other database need not be an Oracle database system. However, to access non-Oracle systems, you must use Oracle Heterogeneous Services.

To create a private database link, you must have the `CREATE DATABASE LINK` system privilege. To create a public database link, you must have the `CREATE PUBLIC DATABASE LINK` system privilege. You must also have the `CREATE SESSION` system privilege on the remote Oracle database.

When an application uses a database link to access a remote database, Oracle Database establishes a database session in the remote database on behalf of the local request. The `CONNECT TO` clause that is used in creating a database link determines how the connection is established on the remote database. You can create fixed user, current user, and connected user database links. Current user links are available only through the Oracle Advanced Security option. The example in the slide shows the syntax to create a fixed user database link.

After you create a database link, you can use it to refer to tables and views on the other database. In SQL statements, you can refer to a table or view on the other database by appending `@dblink` to the table or view name. You can query a table or view on the other database or use any `INSERT`, `UPDATE`, `DELETE`, or `LOCK TABLE` statement for the table.

Connecting to Another Database

```
HQ =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)
      (HOST = edtdr8p1.us.oracle.com)
      (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl.us.oracle.com)
    )
  )
)
```

tnsnames.ora

```
CREATE DATABASE LINK hq.oracle.com
CONNECT TO HR IDENTIFIED BY HR
USING 'hq';

CONNECT hr/hr@hq

SELECT * from HR.employees@hq;
```

SQL*Plus

ORACLE

5 - 34

Copyright © 2007, Oracle. All rights reserved.

Connecting to Another Database

The slide shows the `tnsnames.ora` entry that is needed before creating a database link. The example shows a fixed user database link called `HQ` that is connecting to the user `HR` by using the connect string `HQ`. After you create a database link, you can use it to refer to tables and views on the other database.

The description of the view is as follows:

```
SQL> DESC DBA_DB_LINKS
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
DB_LINK	NOT NULL	VARCHAR2(128)
USERNAME		VARCHAR2(30)
HOST		VARCHAR2(2000)
CREATED	NOT NULL	DATE

```
SQL> select owner, db_link, username from dba_db_links;
```

OWNER	DB_LINK	USERNAME
SYS	HQ.ORACLE.COM	HR

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 and when to use dedicated servers**

ORACLE

Practice 5 Overview: Working with Oracle Network Components

This practice covers the following topics:

- **Configuring local Names Resolution to connect to another database**
- **Creating a second listener for connect-time failover**

6 Managing Database Storage Structures

ORACLE

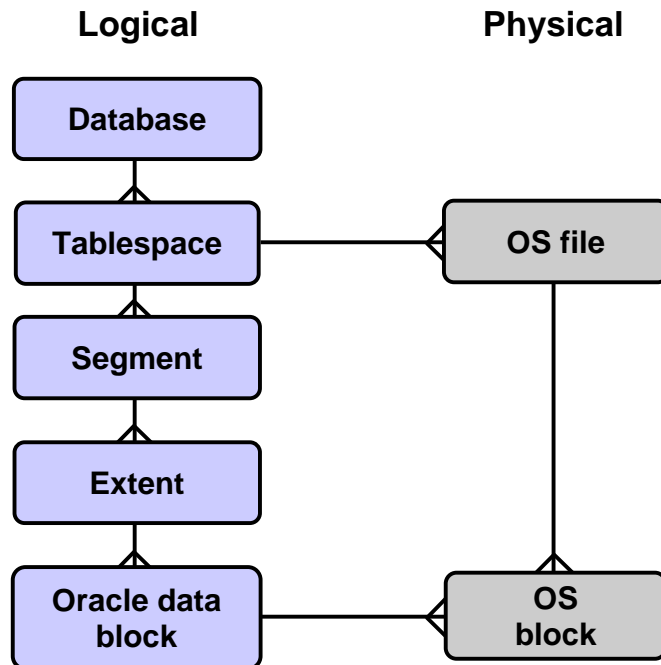
Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- **Describe the storage of table row data 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)**

Storage Structures



Storage Structures

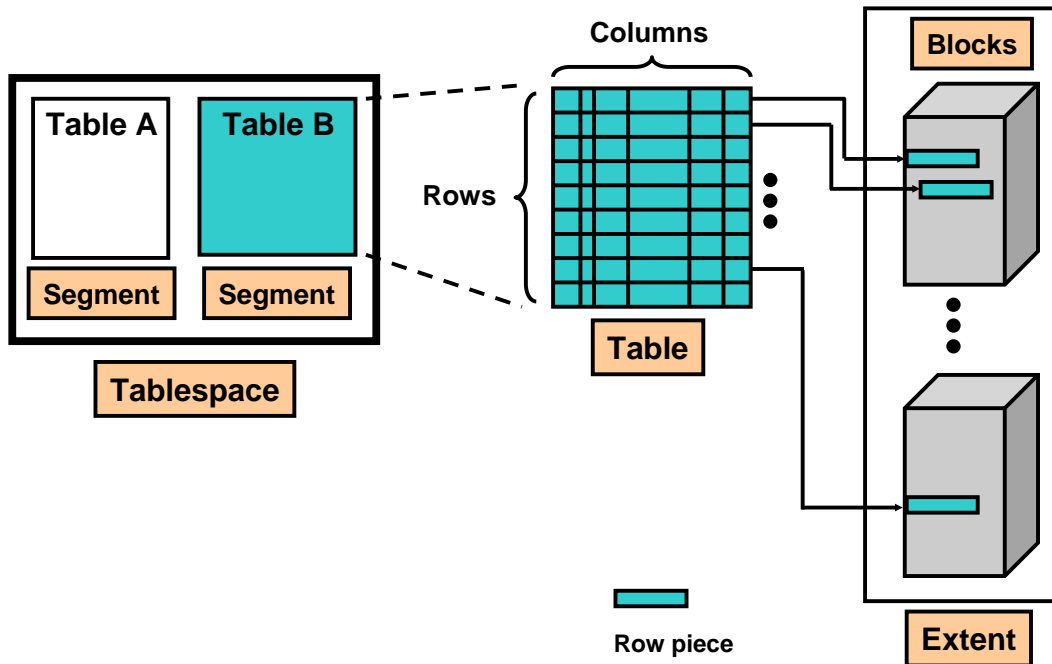
A database is divided into logical storage units called *tablespaces*. Each tablespace has many logical Oracle data blocks. The `DB_BLOCK_SIZE` parameter specifies how large a logical block is. A logical block can range from 2 KB to 32 KB in size. The default size is 8 KB. An Oracle data block is the smallest unit of logical I/O.

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.

The most commonly used block size should be picked as the standard block size. In many cases, this is the only block size that you need to specify. Typically, `DB_BLOCK_SIZE` is set to either 4 KB or 8 KB. If you do not set a value for this parameter, the default data block size is operating system specific, which is generally adequate. You cannot change the block size after database creation except by re-creating the database.

If the database block size is different from the operating system block size, ensure that the database block size is a multiple of the operating system block size.

How Table Data Is Stored

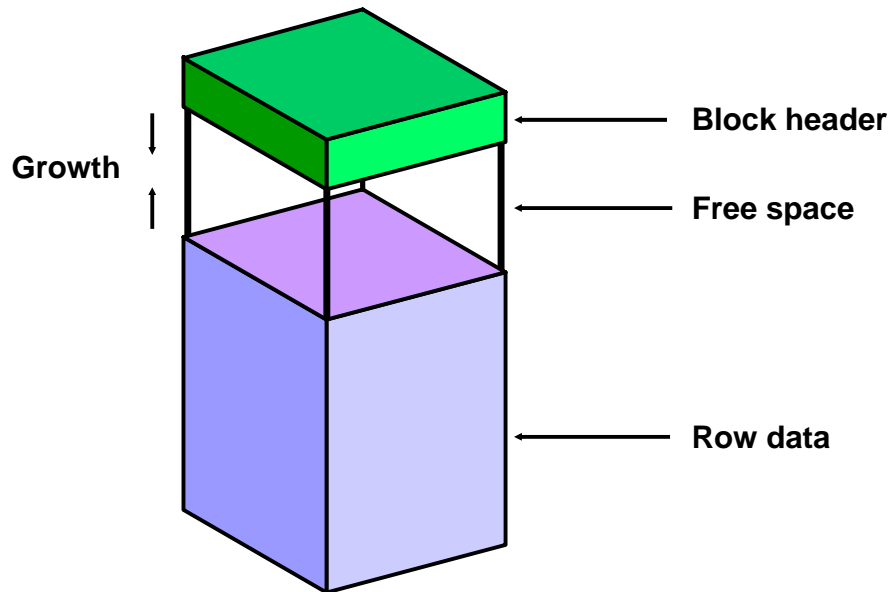


How Table Data Is Stored

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.

Database Block: Contents



Database Block: Contents

- **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 24 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, enabling the header and the row data space to grow when necessary. Row data takes up free space as new rows are inserted or as columns of existing rows are updated with larger values.

Examples of events that cause header growth:

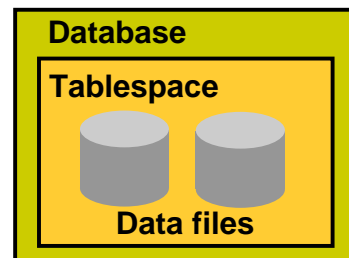
- Row directories that need more row entries
- More transaction slots required than initially configured

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
- **Data files:**
 - Can belong to only one tablespace and one database
 - Are a repository for schema object data



Tablespaces and Data Files

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

- An Oracle database consists of one or more logical storage units—its tablespaces—that collectively store all the database’s data.
- Each tablespace in an Oracle database consists of one or more 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. An Oracle database must have a minimum of 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 have as many as 65,534 data files. If a tablespace consists for its lifetime of exactly one (and only one) data file, it is defined as a *bigfile tablespace*. This is a special case that is useful for data warehouse applications.
- A tempfile is a file that belongs to a temporary tablespace; it is created with the `TEMPFILE` option. Temporary tablespaces cannot contain permanent database objects such as tables, and are typically used for sorting.

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.
 - Use of locally managed tablespaces is recommended.
- **Dictionary-managed tablespace:**
 - Free extents are managed by Oracle.
 - Appropriate tables are updated when extents are allocated or unallocated.
 - These tablespaces are supported only for backward compatibility.



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 in 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



The screenshot shows the Oracle Enterprise Manager 11g Database Control interface. At the top, it displays "ORACLE Enterprise Manager 11g" and "Database Control". Below this, it shows "Database Instance: orcl.us.oracle.com". A navigation bar includes "Home", "Performance", "Availability", "Server", "Schema", and "Data Movement". The main content area is divided into two columns: "Storage" and "Database Configuration". The "Storage" column contains links for "Control Files", "Tablespaces", "Temporary Tablespace Groups", "Datafiles", "Rollback Segments", "Redo Log Groups", and "Archive Logs". The "Database Configuration" column contains links for "Memory Parameters", "Undo Management", "All Initialization Parameters", and "Database Feature Usage".

Click the links to view detailed information.

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 (EM). Detailed information about each structure can be obtained by clicking the links in the Storage region of the Administration page.

Creating a New Tablespace

Create Tablespace

General Storage

Show SQL Cancel OK

* Name INVENTORY

Extent Management **Type** **Status**

Locally Managed Permanent Read Write

Dictionary Managed Set as default permanent tablespace Read Only

Temporary Set as default temporary tablespace Offline

Undo Undo Retention Guarantee Yes No

Datafiles

Use bigfile tablespace
Tablespace can have only one datafile with no practical size limit.

Add Edit Remove

Select	Name	Directory	Size (MB)
<input checked="" type="radio"/>	inventory01.dbf	/u01/app/oracle/oradata/orcl/	50.00

General Storage

Creating a New Tablespace

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, select an existing tablespace and then 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 in 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.
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, in which Automatic Storage Management (ASM) or other logical volume managers support the 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 then enter a file size.
9. In the Storage region, 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.
10. Leave Maximum File Size as Unlimited. Click OK to return to the Create Tablespace page.
11. Click the Storage tab.
The Edit Tablespace page appears.
12. Accept all the defaults on the Storage page.

Note: These steps 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

Extent Allocation

Automatic
 Uniform

Size KB

Segment Space Management

Automatic
Objects in the tablespace automatically manage their free space. It offers high performance for free space management.

Manual
Objects in the tablespace will manage their free space using free lists. It is provided for backward compatibility.

Enable logging

Yes
Generate redo logs for creation of tables, indexes and partitions, and for subsequent inserts. Recoverable

No
Redo log entries are smaller, the above operations are not logged and not recoverable.

Block information

Block Size (B) 8192

ORACLE

6 - 11

Copyright © 2007, Oracle. All rights reserved.

Storage for Locally Managed Tablespaces

The extents in 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 in the tablespace are system managed. You cannot specify Automatic for a temporary tablespace. With Automatic, the extent size may be specified but, because the bits each represent 64 KB, any requested extent size is rounded up to a multiple of 64 KB.
- **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 in a locally managed tablespace can be specified as:

- **Automatic:** The Oracle database uses bitmaps to manage the free space in segments. The bitmap describes the status of each data block in 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, the new state is reflected in the bitmap. With bitmaps, the Oracle database manages free space more automatically. As a result, 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 in segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space in 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 this procedure, see the *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, 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. 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), 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**

Tablespaces

Object Type: Tablespace

Search
Select an object type and optionally enter an object name to filter the data that is displayed in your results set.
Object Name:

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Selection Mode: Single

Select	Name	Allocated Size(MB)	Space Used(MB)	Allocated Space Used(%)	Allocated Free Space(MB)	Status	Datafiles	Type	Extent Management	Segment Management
<input checked="" type="radio"/>	EXAMPLE	100.0	77.7	77.7	22.3	✓	1	PERMANENT LOCAL	AUTO	
<input type="radio"/>	SYSAUX	738.0	700.9	95.0	37.1	✓	1	PERMANENT LOCAL	AUTO	
<input type="radio"/>	SYSTEM	670.0	669.0	99.9	1.0	✓	1	PERMANENT LOCAL	MANUAL	
<input type="radio"/>	TEMP	60.0	0.0	0.0	60.0	✓	1	TEMPORARY LOCAL	MANUAL	
<input type="radio"/>	UNDOTBS1	135.0	12.2	9.1	122.8	✓	1	UNDO LOCAL	MANUAL	
<input type="radio"/>	USERS	5.0	3.0	60.0	2.0	✓	1	PERMANENT LOCAL	AUTO	

Total Allocated Size (MB) 1,708.0
Total Used (MB) 1,462.8
Total Allocated Free Space (MB) 245.2

✓ Online ✗ Offline Ⓜ Read Only

ORACLE

6 - 13

Copyright © 2007, Oracle. All rights reserved.

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 Oracle Database now use the SYSAUX tablespace. Every Oracle Database 10g (or later release) database must have a SYSAUX tablespace.

In Enterprise Manager (EM), 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 in 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 as 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 all newly created users 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, 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, it must have exactly one active undo tablespace per instance 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

Database Instance: orcl.oracle.com > Tablespaces > Edit Tablespace: EXAMPLE Logged in As DBA1

Edit Tablespace: EXAMPLE

Actions: Add Datafile

General | Storage | Thresholds

Name: EXAMPLE

Bigfile tablespace: No

Extent Management

- Locally Managed
- Dictionary Managed

Type

- Permanent
 - Set as default permanent tablespace
- Temporary
 - Set as default temporary tablespace
- Undo

Status

- Read Write
- Read Only
- Offline

Offline Mode: (Dropdown menu open: Normal, Temporary, Immediate, For Recover)

Datafiles

Select	Name	Directory	Size (MB)	Used (MB)
<input checked="" type="checkbox"/>	example01.dbf	/u01/app/oracle/oradata/orcl/	100.00	68.25

ORACLE

6 - 15

Copyright © 2007, Oracle. All rights reserved.

Altering a Tablespace

After you create a tablespace, you can 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 tablespaces 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. 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. 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 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.

Note: System tablespaces may not be taken offline.

Changing the size: You can add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file.

- To add a new data file to the tablespace, click Add. Then enter 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.

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

Selection Mode ▾

▾

Select	Name ▲	Allocated Size(MB)	Used(%)	Allocated Free Space(MB)	Status
<input checked="" type="radio"/>	EXAMPLE	100.0	77.7	22.3	✓
<input type="radio"/>	SYSAUX	738.0	95.0	37.1	✓
<input type="radio"/>	SYSTEM	670.0	99.9	1.0	✓
<input type="radio"/>	TEMP	60.0	0.0	60.0	✓
<input type="radio"/>	UNDOTBS1	135.0	7.5	124.8	✓
<input type="radio"/>	USERS	5.0	60.0	2.0	✓

Total Allocated Size (MB) 1,708.0 Online Offline Read Only

Total Used (MB) 1,460.8

Total Allocated Free Space (MB) 247.2

Show DDL

```
CREATE SMALLFILE TABLESPACE "EXAMPLE" DATAFILE
'/u01/app/oracle/oradata/orcl/example01.dbf' SIZE 100M REUSE AUTOEXTEND ON
NEXT 640K MAXSIZE 32767M NOLOGGING EXTENT MANAGEMENT LOCAL SEGMENT SPACE
MANAGEMENT AUTO
```

Actions with Tablespaces

Using the Actions menu, you can perform a variety of tasks with your tablespaces. Select a tablespace and then select 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 one-way only; you cannot convert the tablespace back to dictionary managed. You can use the PL/SQL package `DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL` to convert to dictionary managed if needed.
- **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 in 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 in the object. At the tablespace level, advice is generated for every segment in the tablespace.
- **Show Dependencies:** Shows objects on which this tablespace depends, 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

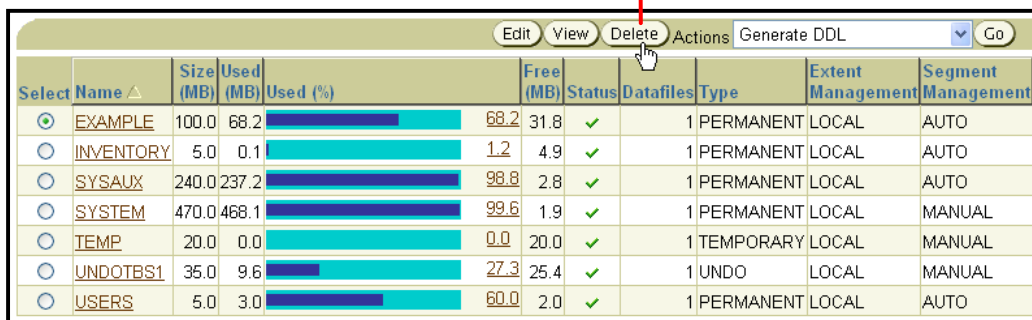
 **Warning**








Once a tablespace has been dropped, the objects and data in it will no longer be available. To recover them can be a time consuming process. Oracle recommends a backup before and after dropping a tablespace.

Are you sure you want to delete Tablespace EXAMPLE?

Delete associated datafiles from the OS

No Yes



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

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. If you are using OMF, the underlying operating system files are also 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 that 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 active segments. For example, if a table in the tablespace is currently being used or if 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	SEGMENT
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

6 - 20

Copyright © 2007, Oracle. All rights reserved.

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

Note: The V\$DBFILE view displays all data files in the database. This view is retained for historical compatibility. Use of V\$DATAFILE is recommended instead.

- **Temp file information:**
 - DBA_TEMP_FILES
 - V\$TEMPFILE

Gathering Storage Information

Tablespaces Object Type | Tablespace

Search
 Select an object type and optionally enter an object name to filter the data that is displayed in your results set.
 Object Name

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Selection Mode | Single

Select	Name	Size (MB)	Used (MB)	Used (%)	Free (MB)	Status	Datafiles	Type	Actions
<input type="checkbox"/>	EXAMPLE	100.0	68.2	68.2	31.8	✓	1	PERMAN	<ul style="list-style-type: none"> Add Datafile Create Like Generate DDL Make Locally Managed Make Readonly Make Writable Place Online Reorganize Run Segment Advisor Show Dependencies Show Tablespace Contents Take Offline
<input type="checkbox"/>	SYSAUX	550.0	542.4	98.6	7.6	✓	1	PERMAN	
<input type="checkbox"/>	SYSTEM	500.0	492.3	98.5	7.7	✓	1	PERMAN	
<input type="checkbox"/>	TEMP	20.0	0.0	0.0	20.0	✓	1	TEMPOR	
<input type="checkbox"/>	UNDOTBS1	110.0	1.4		108.6	✓	1	UNDO	LOCAL MANUAL

Gathering Storage Information

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

Viewing Tablespace Contents

Database Instance: EDRSR10P1_orcl.us.oracle.com > Tablespaces > View Tablespace: EXAMPLE > Show Tablespace Contents

Show Tablespace Contents

Size (MB) **100.0** Used (MB) **68.3** Extent Mgmt **LOCAL** Auto Extend **Yes**
 Block Size (KB) **8** Used (%) **68.3** Segment Mgmt **AUTO** Extents **836**

Segments

Search

Segment Name Type Minimum Size (KB) Minimum Extents

You can use the wildcard symbol (%) in the segment name.

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
OE.PRODUCT_DESCRIPTIONS	TABLE	3,072	18
SH.SALES.SALES_Q4_2001	TABLE PARTITION	2,048	17
SH.SALES.SALES_Q3_2001		1,024	16
SH.SALES.SALES_Q1_1999		1,024	16
SH.CUSTOMERS_PK		1,024	16
SH.SALES.SALES_Q2_2001		960	15
SH.SALES.SALES_Q1_2001		960	15
SH.SALES.SALES_Q1_2000		960	15

[Extent Map](#)

Extent Map

Clicking the Highlight Extents button in the Extent Map. Clicking on a used extent in the Extent Map.

- Header
- Used
- Free
- Selected
- Unmapped

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

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	Default location for the flash recovery area

Example:

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

ORACLE

Oracle Managed Files (OMF)

Oracle Managed Files eliminates the need for you to directly manage the operating system files in 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

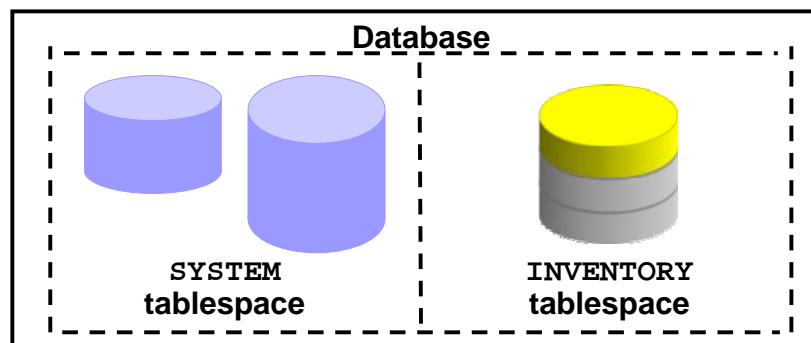
A database can have a mixture of Oracle-managed and unmanaged files. The file system directory specified by either of these parameters must already exist; the database does not create it. The directory must also have permissions for the database to create the files in it.

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. When you create a tablespace as shown, default values are assigned to all parameters.

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

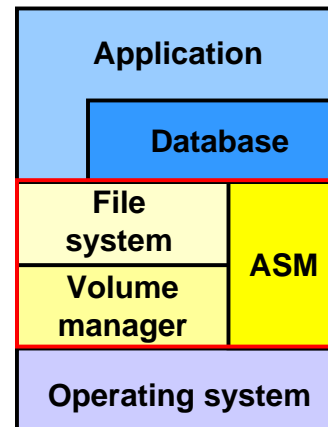


Enlarging the Database

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

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 storage-management challenges



ORACLE

6 - 25

Copyright © 2007, Oracle. All rights reserved.

Automatic Storage Management

Automatic Storage Management (ASM) provides 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 administrator's ability to manage more and larger databases with increased efficiency.

ASM: Key Features and Benefits

- **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**

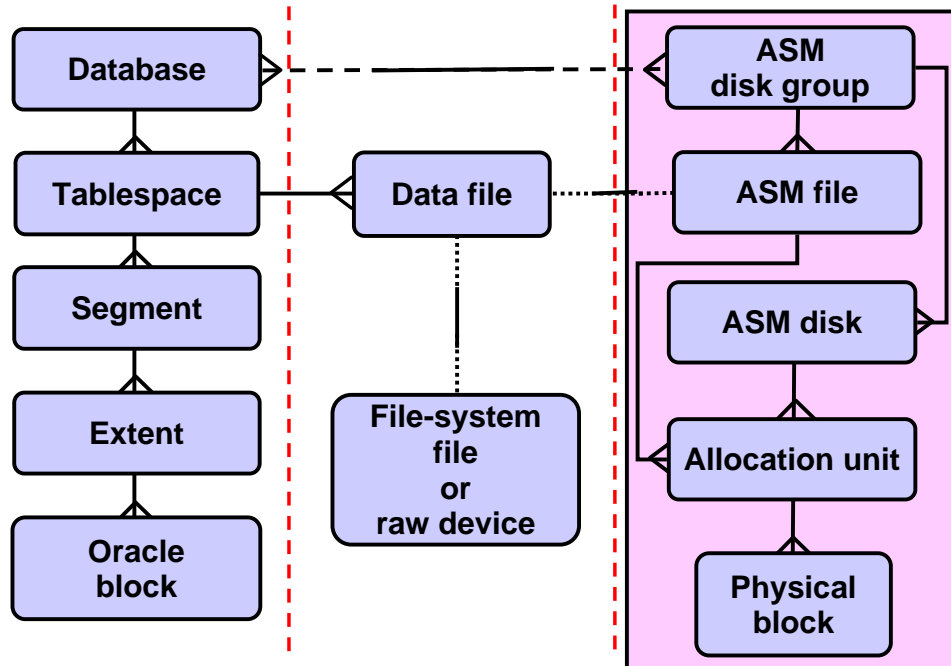
ASM: Key Features and Benefits

ASM divides files into extents (different from the data file extents discussed earlier) and spreads the extents for each file evenly across all disks. It uses an index technique to track the placement of each extent. When 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 files and maintain a balanced load across 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 you can decrease the speed to reduce impact on the I/O subsystem. ASM provides mirroring protection without the need to purchase a third-party Logical Volume Manager. A unique advantage of ASM is that mirroring is applied on a file basis rather than 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 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 among the various storage components in an Oracle database that uses ASM. The left and middle parts of the diagram show the relationships that existed 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 composed of multiple ASM disks, and each ASM disk belongs to only one disk group. ASM files are always spread across all ASM disks in the disk group. ASM disks are partitioned in allocation units. An allocation unit (AU) is the smallest contiguous disk space that ASM allocates. When you create a disk group, you can set the ASM AU size to be between 1 MB and 64 MB in powers of two (1, 2, 4, 8, 16, 32, or 64). Larger AU sizes typically provide performance advantages for data warehouse applications that use large sequential reads.

Note: The graphic in the slide deals with only one type of ASM file—the data file. However, ASM can be used to store other database file types.

Summary

In this lesson, you should have learned how to:

- **Describe the storage of table row data 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)**

Practice 6 Overview: Managing Database Storage Structures

This practice covers the following topics:

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



Administering User Security

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able 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**

ORACLE

7 - 2

Copyright © 2007, Oracle. All rights reserved.

Objectives

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

A *database user account* is a way 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.

A *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

Each database user account has:

- **A unique username**
- **An authentication method**
- **A default tablespace**
- **A temporary tablespace**
- **A user profile**
- **An initial consumer group**
- **An account status**



ORACLE

Database User Accounts

To access the database, a user must specify a valid database user account and successfully authenticate as required by that user account. Each database user has a unique database account. This is Oracle's best practice recommendation to avoid potential security holes and provide meaningful data for certain audit activities. However, users may sometimes share a common database account. In these rare cases, the 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 11g supports several other authentication methods, including biometric, certificate, and token authentication.
- **A default tablespace:** This is a place where a user creates objects if the user 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 in that tablespace in which to create objects. Both of these are granted separately.
- **A temporary tablespace:** This is a place where temporary objects, such as sorts and temporary tables, are created on behalf of the user by the instance. No quota is applied to temporary tablespaces.

Database User Accounts (continued)

- **A user profile:** This is a set of resource and password restrictions assigned to the user.
- **An initial consumer group:** This is used by the resource manager.
- **An account status:** Users can access only “open” accounts. The `account_status` may be in various combinations of “locked” and “expired.”

Note: A database user is not necessarily a person. It is a common practice to create a user that owns the database objects of a particular application, such as HR. The database user can be a device, an application, or just a way to group database objects for security purposes. The personal identifying information of a person is not needed for a database user.

Predefined Accounts: SYS and SYSTEM

- **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)
- **SYSTEM account is granted the DBA role.**
- **These accounts are not used for routine operations.**

ORACLE

7 - 5

Copyright © 2007, Oracle. All rights reserved.

Predefined Accounts: SYS and SYSTEM

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

In addition, the SYS account has all privileges with ADMIN OPTION and owns the data dictionary. To connect to the SYS account, you must use the AS SYSDBA clause for a database instance and AS SYSASM for an Automatic Storage Management (ASM) instance. 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, SYSOPER, or SYSASM privileges are allowed to start up and shut down instances.

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

Database Instance: orcl > Users > Logged in As SYS

Create User

[Show SQL](#) [Cancel](#) [OK](#)

General [Roles](#) [System Privileges](#) [Object Privileges](#) [Quotas](#) [Consumer Group Privileges](#) [Proxy Users](#)

* Name

Profile

Authentication

* Enter Password

* Confirm Password

For Password choice, the role is authorized via password.

Expire Password now

Default Tablespace

Temporary Tablespace

Status Locked Unlocked

Select Server > Users, and then click the Create button.

ORACLE

7 - 6

Copyright © 2007, Oracle. All rights reserved.

Creating a User

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

To create a database user:

1. In Enterprise Manager Database Control, click the Server tab and then click Users in the Security section.
2. Click the Create button.

Provide the required information. Mandatory items (such as Name) are marked with an asterisk (*).

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. If users do not specify a tablespace when creating an object, the object will be created in the default tablespace assigned to the object owner. This enables you to control where their objects are created.

If you do not choose a default tablespace, the system-defined default permanent tablespace is used. The case is similar for the temporary tablespace: If you do not specify a tablespace, the system-defined temporary tablespace is used.

Authenticating Users

- Password
- External
- Global



Edit User: HR

Actions: Create Like [Go] Show SQL Revert Apply

General Roles System Privileges Object Privileges Quotas Consumer Group Privileges Proxy Users

Name: HR

Profile: DEFAULT

Authentication: Password

* Enter Password: Password

* Confirm Password: External

Global

For Password choice, the role is authorized via password.

Expire Password now

Default Tablespace: USERS

Temporary Tablespace: TEMP

Status: Locked Unlocked

ORACLE

7 - 7

Copyright © 2007, Oracle. All rights reserved.

Authenticating Users

Authentication means verifying the identity of someone or something (a user, device, or other entity) that 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 that are 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. All passwords created in Oracle Database 11g are case-sensitive by default. These passwords may also contain multibyte characters and are limited to 30 bytes. Each password created in a database that is upgraded to Oracle Database 11g remains case-insensitive until the password is changed.

Passwords are always automatically and transparently encrypted using the Advanced Encryption Standard (AES) algorithm during network (client/server and server/server) connections before sending them across the network.

Authenticating Users (continued)

External: This is authentication by a method outside the database (operating system, Kerberos, or Radius). The Advanced Security Option is required for Kerberos or Radius. Users can connect to the Oracle database without specifying a username or password. The Advanced Security Option (which is a strong authentication) allows users to be identified through the use of biometrics, x509 certificates, and token devices. With external authentication, your database relies on the underlying operating system, network authentication service, or external 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 use operating system authentication, 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, suppose 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, 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 the Oracle documentation that is specific to your operating system for more information about this initialization parameter.

Global: With the Oracle Advanced Security option, global authentication enables users to be identified through the use of Oracle Internet Directory.

For more information about advanced authentication methods, see the *Oracle Database Security* 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:

- **For SYSDBA, SYSOPER, and SYSASM connections:**
 - **DBA user by name is audited for password file and strong authentication methods**
 - **OS account name is audited for OS authentication**
 - **OS authentication takes precedence over password file authentication for privileged users**
 - **Password file uses case-sensitive passwords**

ORACLE

7 - 9

Copyright © 2007, Oracle. All rights reserved.

Administrator Authentication

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

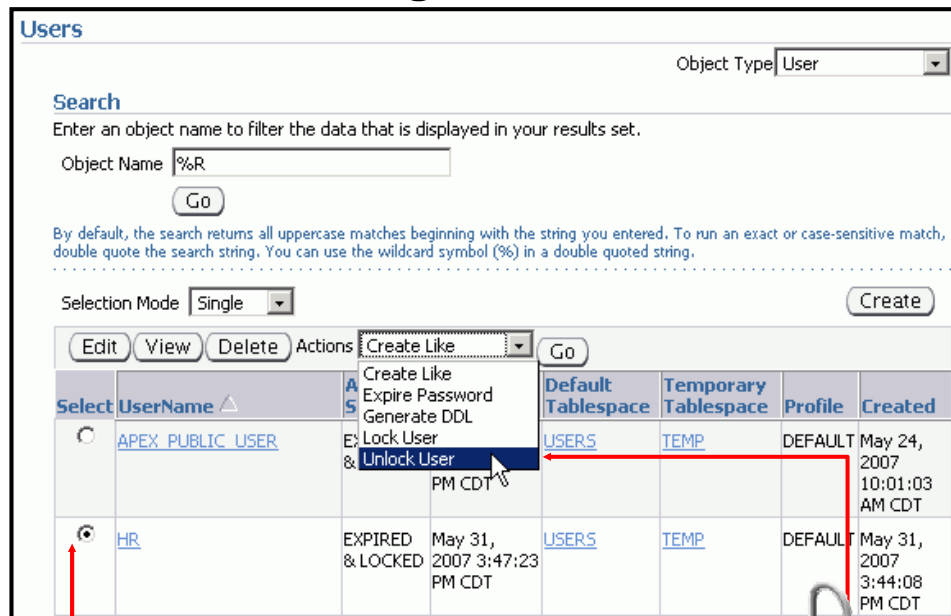
Administrator security: Connections for the privileged users `SYSDBA`, `SYSOPER`, and `SYSASM` are authorized only after verification with the password file or with the OS privileges and permissions. If OS authentication is used, the database does *not* use the supplied username and password. OS authentication 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 are supplied. The password file in Oracle Database 11g uses case-sensitive passwords by default.

However, if authentication succeeds by means of the password file, the connection is logged with the username. If authentication succeeds by means of the operating system, 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`, `SYSOPER`, or `SYSASM`, you will be connected with the associated administrative privileges regardless of the username and password that you specify.

In Oracle Database 11g, a privileged user may use strong authentication methods: Kerberos, SSL, or directory authentication if the Advanced Security Option is licensed.

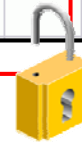
Unlocking a User Account and Resetting the Password



The screenshot shows the Oracle 'Users' page. At the top, there is a search bar with 'Object Name' set to '%R' and a 'Go' button. Below the search bar, there is a 'Selection Mode' dropdown set to 'Single' and a 'Create' button. A table of users is displayed with columns: Select, UserName, Actions, Default Tablespace, Temporary Tablespace, Profile, and Created. The 'HR' user is selected, and the 'Unlock User' option is highlighted in the 'Actions' dropdown menu. A red arrow points from the 'Unlock User' option to the 'HR' user row. A yellow callout box with a black border and a yellow background contains the text 'Select the user and click Unlock User.' with a red arrow pointing to the 'HR' user row. A yellow padlock icon is positioned to the right of the callout box.

Select	UserName	Actions	Default Tablespace	Temporary Tablespace	Profile	Created
<input type="radio"/>	APEX_PUBLIC_USER	Create Like Expire Password Generate DDL Lock User & Unlock User	USERS	TEMP	DEFAULT	May 24, 2007 10:01:03 AM CDT
<input checked="" type="radio"/>	HR	EXPIRED & LOCKED	USERS	TEMP	DEFAULT	May 31, 2007 3:44:08 PM CDT

Select the user and click Unlock User.



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 did not choose 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.

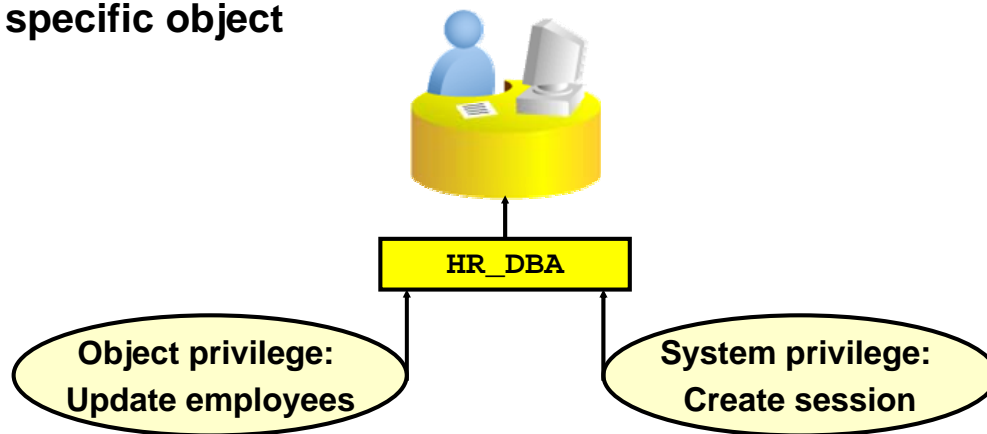
Alternatively, perform the following steps on the Edit Users page:

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

There are two types of user privileges:

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



ORACLE

7 - 11

Copyright © 2007, Oracle. All rights reserved.

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 the users can and cannot do in 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 has been given explicit permission to administer the privilege. There are more than one 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

The screenshot shows the Oracle Database Administration console interface for editing user privileges. The main window is titled 'Edit User: HR' and has several tabs: General, Roles, System Privileges (selected), Object Privileges, Quotas, Consumer Group Privileges, and Proxy Users. The 'System Privileges' tab displays a table with two columns: 'System Privilege' and 'Admin Option'. The 'Admin Option' column contains checkboxes for each privilege. A red box highlights the 'Edit List' button in the top right corner of the table. A 'Modify System Privileges' dialog box is open in the foreground, showing a list of 'Available System Privileges' on the left and a list of 'Selected System Privileges' on the right. The 'Selected System Privileges' list includes ALTER SESSION, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE VIEW, and UNLIMITED TABLESPACE. The dialog also has 'Move', 'Move All', 'Remove', and 'Remove All' buttons between the two lists.

System Privilege	Admin Option
ALTER SESSION	<input type="checkbox"/>
CREATE DATABASE LINK	<input type="checkbox"/>
CREATE SEQUENCE	<input type="checkbox"/>
CREATE SESSION	<input type="checkbox"/>
CREATE SYNONYM	<input type="checkbox"/>
CREATE VIEW	<input type="checkbox"/>
UNLIMITED TABLESPACE	<input type="checkbox"/>

ORACLE

7 - 12

Copyright © 2007, Oracle. All rights reserved.

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, if you have the CREATE TABLE privilege, you can create a table—but only in your own schema. The SELECT ANY TABLE privilege allows you to select from tables owned by other users. The SYS user and users with the DBA role are granted all of the ANY privileges; they can therefore do anything to any data object. The scope of the ANY system privileges can be controlled using the Oracle Database Vault Option.

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

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

- **RESTRICTED SESSION:** This privilege allows 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.

- **SYSASM:** This privilege allows you to start up, shut down and administer an ASM instance.
- **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 tablespace attributes.
- **CREATE LIBRARY:** The Oracle database allows developers to create and call external code (for example, a C library) from PL/SQL. The library must be named by a LIBRARY object in the database. The CREATE LIBRARY privilege allows a user to create an arbitrary code library that is executable from PL/SQL.
- **CREATE ANY DIRECTORY:** 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 (for example, renaming a data file or flushing the buffer cache).

Object Privileges

Edit User: HR

Actions: Create Like Go Show SQL Revert Apply

General Roles System Privileges **Object Privileges** Quotas Consumer Group Privileges

Select Object Type: Table Add

Select Object Privilege	Schema	Object
<input checked="" type="checkbox"/> EXECUTE	SYS	DBMS_STATS

General Roles System Privileges **Object Privileges** Quotas Consumer Group Privileges

Add Table Object Privileges Cancel OK

* Select Table Objects

OE.CUSTOMERS,OE.INVENTORIES,OE.ORDERS
,OE.ORDER_ITEMS

(SchemaName.Table,...)
Select object and then choose privileges to assign

Available Privileges	Selected Privileges
ALTER DELETE INDEX INSERT REFERENCES UPDATE	SELECT

- To grant object privileges:**
- 1. Choose the object type.**
 - 2. Select objects.**
 - 3. Select privileges.**

ORACLE

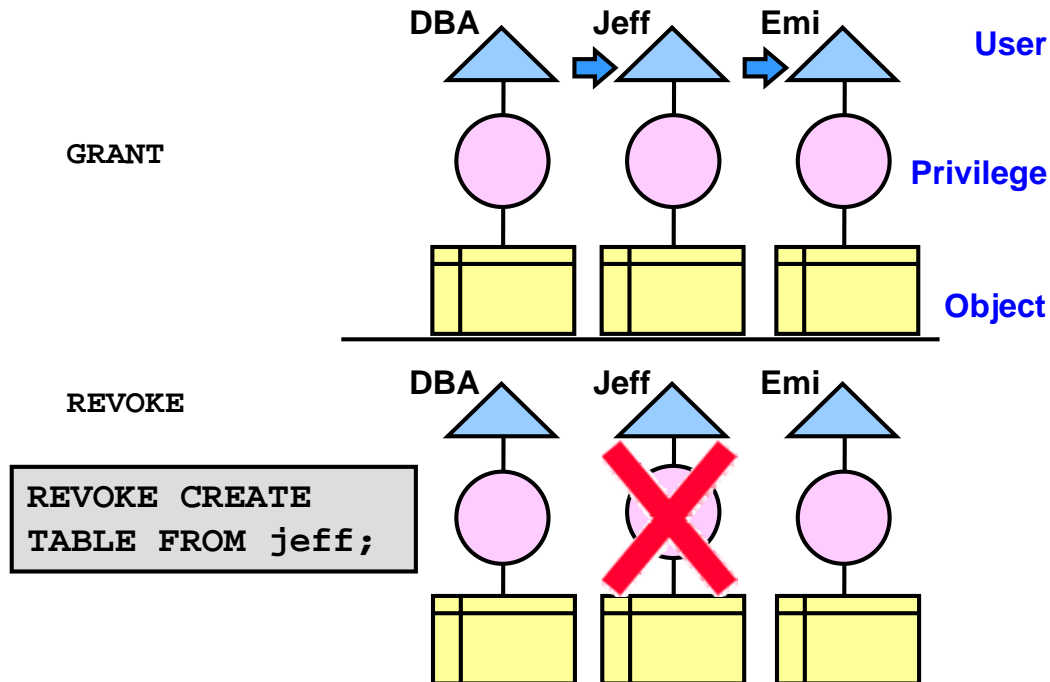
Object Privileges

To grant object privileges, click the Object Privileges tab on the Edit User page. Select the type of object on which you want to grant privileges, and then click the Add button. Choose the objects by either entering `<username.object name>` or selecting them from the list.

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

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



ORACLE

7 - 15

Copyright © 2007, Oracle. All rights reserved.

Revoking System Privileges

System privileges that 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.

The slide illustrates the following situation.

Scenario

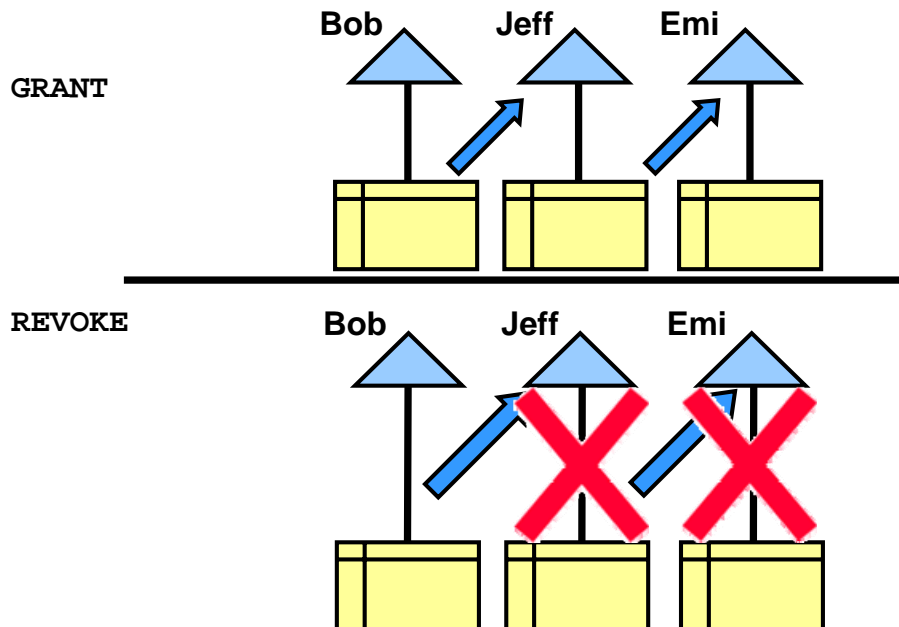
1. The DBA grants the CREATE TABLE system privilege to Jeff with ADMIN OPTION.
2. Jeff creates a table.
3. Jeff grants the CREATE TABLE system privilege to Emi.
4. Emi creates a table.
5. The DBA revokes the CREATE TABLE system privilege from Jeff.

Result

Jeff's table still exists, but Jeff cannot create new tables.

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

Revoking Object Privileges with GRANT OPTION



ORACLE

7 - 16

Copyright © 2007, Oracle. All rights reserved.

Revoking Object Privileges with GRANT OPTION

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 if 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`. As a user, you can revoke only those privileges that you have granted. For example, Bob cannot revoke the object privilege that Jeff granted to Emi. Even a DBA cannot revoke object privileges that were not granted by that DBA.

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. The `SELECT` privilege is revoked from Jeff. This revoke is cascaded to Emi as well.

Benefits of Roles

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



ORACLE

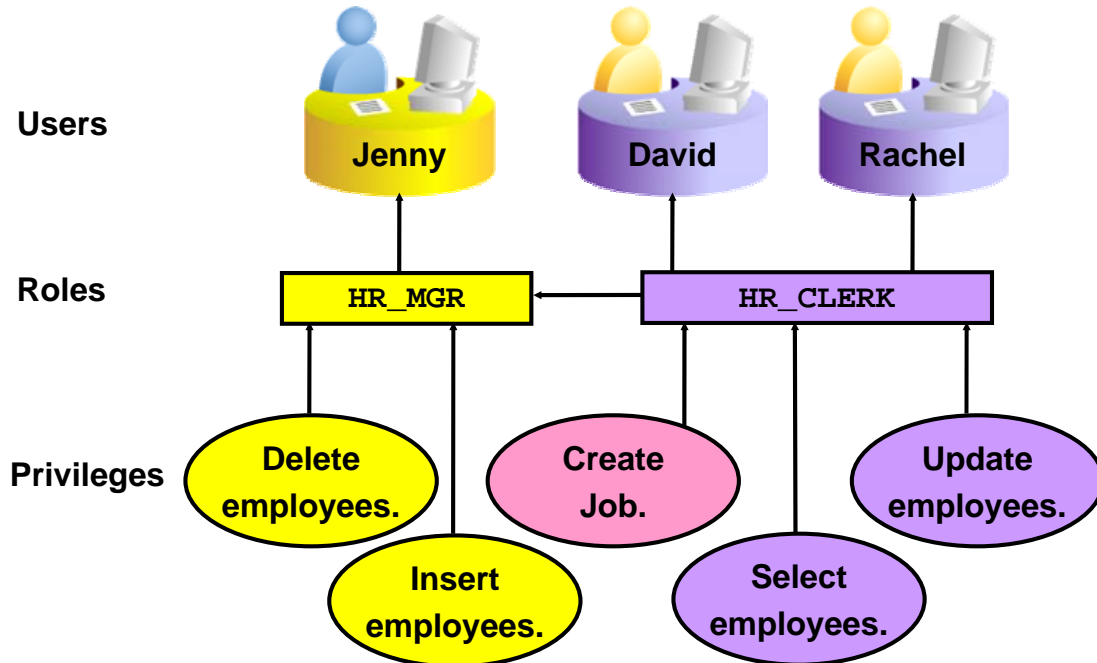
7 - 17

Copyright © 2007, Oracle. All rights reserved.

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 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. This allows the privileges of the user to be controlled in a given situation.

Assigning Privileges to Roles and Assigning Roles to Users



ORACLE

7 - 18

Copyright © 2007, Oracle. All rights reserved.

Assigning Privileges to Roles and Assigning Roles to Users

In most systems, it is time consuming and error prone to grant necessary privileges to each user individually. 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 are 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 SELECT and UPDATE privileges on the employees table *and* the CREATE JOB system privilege are granted to the HR_CLERK role. DELETE and INSERT privileges on the employees table *and* the HR_CLERK role are granted to the HR_MGR 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; HS_ADMIN_ROLE and over 1,700 object privileges on the data dictionary

ORACLE

7 - 19

Copyright © 2007, Oracle. All rights reserved.

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 that is created with Enterprise Manager. For security reasons, the `CONNECT` role has contained only the `CREATE SESSION` privilege since version 10.2.0 of the Oracle Database.

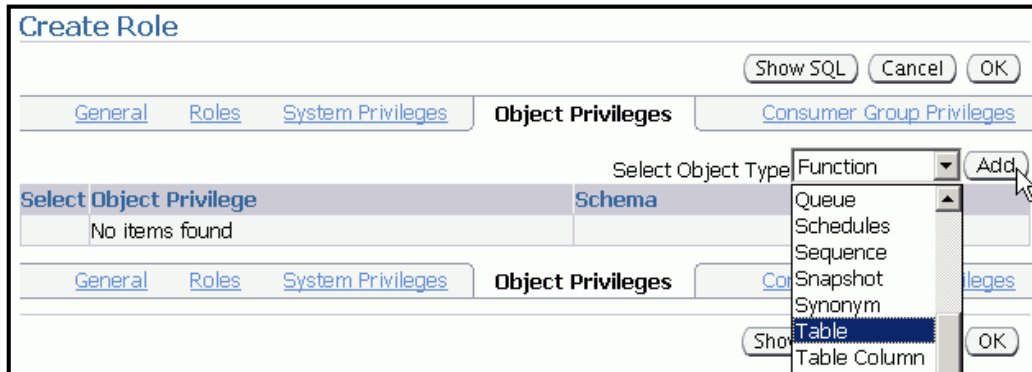
Note: Be aware that granting the `RESOURCE` role includes granting the `UNLIMITED TABLESPACE` privilege.

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 Server > Roles.

ORACLE

7 - 20

Copyright © 2007, Oracle. All rights reserved.

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:

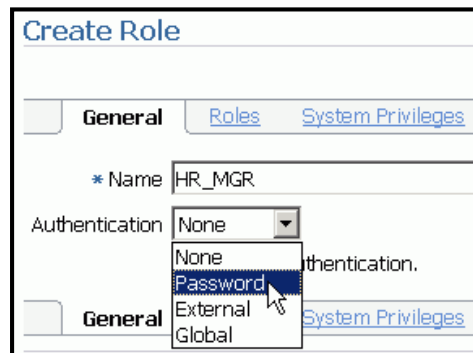
1. In Enterprise Manager Database Control, click the Server tab and then click Roles under the Security heading.
2. Click the Create button.

Secure Roles

- Roles can be nondefault.

```
SET ROLE vacationdba;
```

- Roles can be protected through authentication.



The screenshot shows the 'Create Role' dialog box with the following details:

- Title: Create Role
- Tab: General (selected), Roles, System Privileges
- * Name: HR_MGR
- Authentication: None (selected in dropdown)
- Authentication dropdown options: None, Password (highlighted), External, Global
- Bottom tabs: General (selected), System Privileges

- Roles can also be secured programmatically.

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

ORACLE

7 - 21

Copyright © 2007, Oracle. All rights reserved.

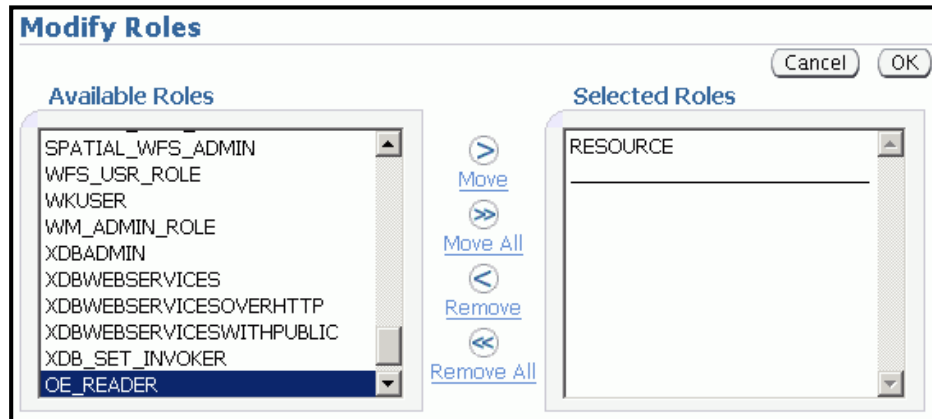
Secure Roles

Roles are usually enabled by default, which means that if a role is granted to a user, then that user can exercise the privileges given to the role. Default roles are assigned to the user at connect time.

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, the program that the user is running, the time of day, and other elements needed to properly secure a group of permissions.
- Administer roles easily using the Oracle Database Vault option. Secure application roles are simplified, and traditional roles can be further restricted.

Assigning Roles to Users



ORACLE

Assigning Roles to Users

You can use roles to administer database privileges. You can add privileges to a role and 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 that are granted to that role and all privileges of other roles that are 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:

1. In Enterprise Manager Database Control, click the Server tab and then click Users under the Security heading.
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

Users are assigned only one profile at a time.

Profiles:

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

Section	Parameter	Value
Details	CPU/Session (Sec./100)	1000
	CPU/Call (Sec./100)	UNLIMITED
	Connect Time (Minutes)	DEFAULT
	Idle Time (Minutes)	60
Database Services	Concurrent Sessions (Per User)	DEFAULT
	Reads/Session (Blocks)	DEFAULT
	Reads/Call (Blocks)	DEFAULT
	Private SGA (KBytes)	DEFAULT
	Composite Limit (Service Units)	DEFAULT

ORACLE

7 - 23

Copyright © 2007, Oracle. All rights reserved.

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), can be unlimited (as in CPU/Call), or can 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 resource limitations are ignored. Profile password settings are always enforced

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), 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. The user receives an error message such as the following:

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

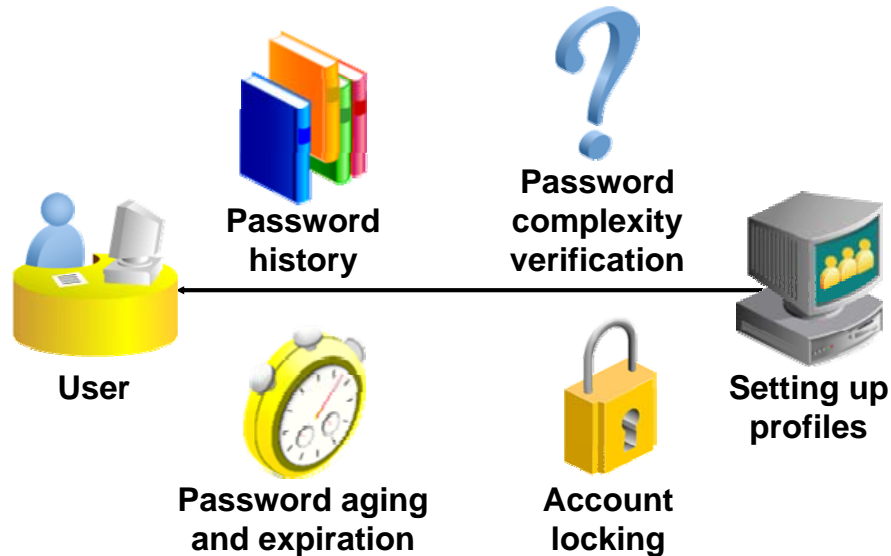
- **Network/Memory:** Each database session consumes system memory resources and (if the session is from a user who is not local to the server) network resources. You can specify the following:
 - **Connect Time:** Indicates for how many minutes a user can be connected before being automatically logged 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 in 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 covered in the lesson titled "Configuring the Oracle Network Environment.")
- **Disk I/O:** This limits the amount of data a user can read at the per-session level 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 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, click the Server tab and then click Profiles under the Security heading. On the Profiles page, 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



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

ORACLE

7 - 25

Copyright © 2007, Oracle. All rights reserved.

Implementing Password Security Features

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

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

- **FAILED_LOGIN_ATTEMPTS:** Specifies the number of failed login attempts before the lockout of the account
- **PASSWORD_LOCK_TIME:** 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

- **PASSWORD_LIFE_TIME:** Determines the lifetime of the password in days, after which the password expires
- **PASSWORD_GRACE_TIME:** 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

Recall that the values of the profile parameters are either set or inherited from the `DEFAULT` profile.

If both password history parameters have a value of `UNLIMITED`, Oracle Database ignores both. The user can reuse any password at any time, which is not a good security practice.

If both parameters are set, password reuse is allowed—but only after meeting both conditions. The user must have changed the password the specified number of times, and the specified number of days must have passed since the old password was last used.

For example, the profile of user `ALFRED` has `PASSWORD_REUSE_MAX` set to 10 and `PASSWORD_REUSE_TIME` set to 30. User `ALFRED` cannot reuse a password until he has reset the password 10 times and until 30 days have passed since the password was last used.

If one parameter is set to a number and the other parameter is specified as `UNLIMITED`, then the user can never reuse a password.

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`). A model password verification function is provided in the `utlpwdmg.sql` script found in the following directories:

- Unix and Linux platforms: `$ORACLE_HOME/rdbms/admin`
- Windows platforms: `%ORACLE_HOME%\rdbms\admin`

Creating a Password Profile

Create Profile

Show SQL Cancel OK

General Password

Password

Expire in (days) 90

Lock (days past expiration) 10

History

Number of passwords to keep 4

Number of days to keep for 90

Complexity

Complexity function VERIFY_FUNCTION_11G

Failed Login

Number of failed login attempts to lock after 3

Number of days to lock for 5/1440

ORACLE

7 - 27

Copyright © 2007, Oracle. All rights reserved.

Creating a Password Profile

To create a password profile, click the Server tab and then click Profiles under the Security heading. On the Profiles page, click the Create button. Click the Password tab to set the password limits.

You can choose common values for each of the settings 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 also be expressed as fractions. There are 1,440 minutes in a day; 5/1,440 is therefore five minutes.

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

If the `utlpwdmg.sql` script has been run, the `VERIFY_FUNCTION` and `VERIFY_FUNCTION_11G` functions are available. If you have created your own complexity function, the name of that function may be entered. The function name does not appear in the Select list. If the function produces run-time errors, the user is unable to change the password.

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`), all users who have that profile are automatically assigned the `DEFAULT` profile.

Supplied Password Verification Function: VERIFY_FUNCTION_11G

The VERIFY_FUNCTION_11G function insures that the password is:

- At least eight characters
- Different from the username, username with a number, or username reversed
- Different from the database name or the database name with a number
- A string with at least one alphabetic and one numeric character
- Different from the previous password by at least three letters

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



ORACLE

Supplied Password Verification Function: VERIFY_FUNCTION_11G

The Oracle server provides two password complexity verification functions named VERIFY_FUNCTION and VERIFY_FUNCTION_11g. These functions are created with the <oracle_home>/rdbms/admin/utlpwdmg.sql script. The VERIFY_FUNCTION is provided for those who prefer the password function provided with previous versions. 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 180
PASSWORD_GRACE_TIME 7
PASSWORD_REUSE_TIME UNLIMITED
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 10
PASSWORD_LOCK_TIME 1
PASSWORD_VERIFY_FUNCTION verify_function_11g;
```

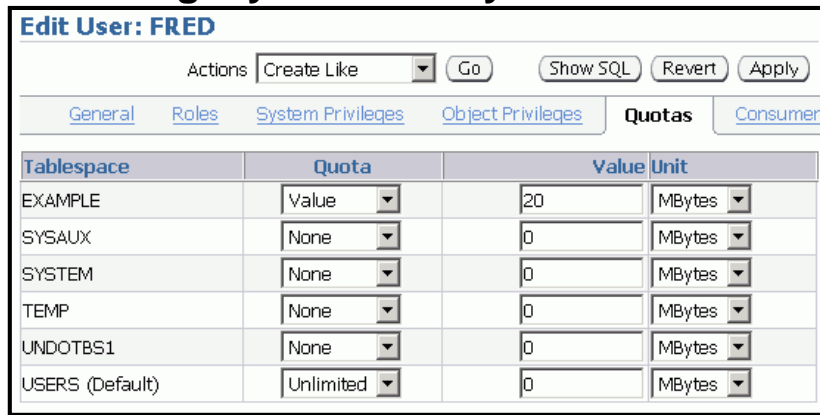
Remember that when users are created, they are assigned the DEFAULT profile unless another profile is specified.

Assigning Quotas to Users

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

Quotas can be:

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



Tablespace	Quota	Value	Unit
EXAMPLE	Value	20	MBytes
SYSAUX	None	0	MBytes
SYSTEM	None	0	MBytes
TEMP	None	0	MBytes
UNDOTBS1	None	0	MBytes
USERS (Default)	Unlimited	0	MBytes

ORACLE

7 - 29

Copyright © 2007, Oracle. All rights reserved.

Assigning Quotas to Users

A *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 quota for a user on a tablespace.

- **Unlimited:** Allows the user to use as much space as is available in the tablespace
- **Value:** 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:** 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 a quota to users on the SYSTEM or SYSAUX tablespaces. Typically, only the SYS and SYSTEM users are able to create objects in the SYSTEM or SYSAUX tablespaces.

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

Assigning Quotas to Users (continued)

- The Oracle instance checks the quota when a user creates or extends a segment.
- For activities that are assigned to a user schema, only those activities that use space in a tablespace 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 tablespaces).
- The quota is replenished when objects owned by the user are dropped with the `PURGE` clause or when the objects owned by the user 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**

ORACLE

Practice 7 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**

ORACLE

8

Managing Schema Objects

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Objectives

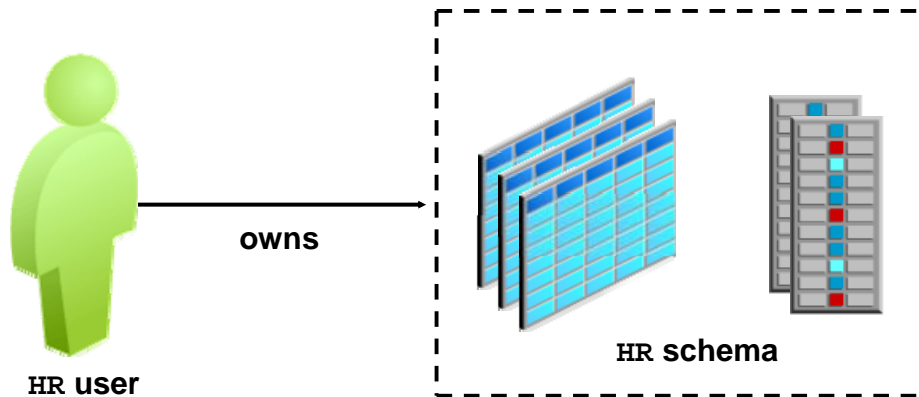
After completing this lesson, you should be able to:

- **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**

ORACLE

What Is a Schema?

> Schema
Constraints
Indexes
Views
Sequences
Temp Tables
Data Dict



What Is a Schema?

A *schema* is a collection of database objects that are owned by a particular user. For a production database, this user typically represents not 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 a 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 need to be directly related to a single tablespace. You can define configurations so that objects in a single schema can be in different tablespaces and so that a tablespace can hold objects from different schemas.

When you create the database, several schemas are created for you, including the following two important schemas:

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

What Is a Schema? (continued)

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

The screenshot displays the Oracle Enterprise Manager 11g Database Control interface. At the top, it shows 'ORACLE Enterprise Manager 11g' and 'Database Control'. Below this, the 'Database Instance: orcl.us.oracle.com' is identified. A navigation bar includes links for Home, Performance, Availability, Server, Schema (which is selected), Data Movement, and Software and Support. The main content area is organized into several sections:

- Database Objects:** Tables, Indexes, Views, Synonyms, Sequences, Database Links, Directory Objects, Reorganize Objects.
- Users & Privileges:** Users, Roles, Profiles, Audit Settings.
- User Defined Types:** Array Types, Object Types, Table Types.
- Enterprise Manager Administration:** Administrators, Notification Schedule, Blackouts.
- Programs:** Packages, Package Bodies, Procedures, Functions, Triggers, Java Classes, Java Sources.
- Materialized Views:** Materialized Views, Materialized View Logs, Refresh Groups, Dimensions.
- Change Management:** Dictionary Baselines, Dictionary Comparisons.
- XML Database:** Configuration, Resources, Access Control Lists, XML Schemas, XMLType Tables, XMLType Views, XML Indexes, XML Repository Events.
- Workspace Manager:** Workspaces.
- Text Manager:** Text Indexes.

Accessing Schema Objects

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

After you click any link, 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 Schema 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.**



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, you must do the following if you name a table "Local Temp":

```
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, 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 all-uppercase characters 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 December 31, 9999 (A.D.)
- **NUMBER(*p*,*s*):** Number with precision *p* and scale *s*

ABC



42

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 that each argument can have.

Built-in data types in the Oracle database include:

- **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 (that is, the length of the column is measured in bytes).
 - CHAR indicates that the column has character semantics (that is, it treats strings as a sequence of characters).
- **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 December 31, 9999 (A.D.). 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**: 32-bit floating-point number. This data type requires 5 bytes, including the length byte.
- **BINARY_DOUBLE**: 64-bit floating-point number. This data type requires 9 bytes.
- **FLOAT(*p*)**: 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**: Equivalent to **NUMBER(*p*,0)**
- **NCHAR(*length*)**: 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**])**: Unicode-only data type. It is like **NCHAR** except that its maximum length is 4,000 bytes and it is not blank-padded.
- **LONG**: Character data of variable length of up to 2 GB (or $2^{31} - 1$ bytes). The **LONG** data type is deprecated; use the large object (LOB) data type instead.
- **LONG RAW**: Raw binary data of variable length of up to 2 GB
- **RAW(*size*)**: Raw binary data of length *size* bytes. The maximum size is 2,000 bytes. You must specify the size for a **RAW** value.
- **ROWID**: 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**: 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. **UROWID** data type in a column has a type code for 1 byte followed by the row ID of a matching type. The three types are physical row ID, logical row ID, and foreign row IDs (which occur when selecting a row ID column over a dblink via a heterogeneous service gateway).
- **BLOB**: Binary large object
- **CLOB**: 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.

Note: For single byte character sets, columns that are defined in character semantics are basically the same as those defined in byte semantics. Character semantics are useful for defining varying-width multibyte strings, reducing the complexity when defining the actual length requirements for data storage. For example, in a Unicode database (UTF8), you need to define a **VARCHAR2** column that can store up to five Chinese characters together with five English characters. In byte semantics, this would require $(5 * 3 \text{ bytes}) + (1 * 5 \text{ bytes}) = 20$ bytes; in character semantics, the column requires 10 characters.

Specifying Data Types in Tables (continued)

- **NCLOB:** Character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, and both use the NCHAR database character set. This data type stores national character set data.
Note: Maximum size for all LOB data types (BLOB, CLOB, and NCLOB) is $(4 \text{ GB} - 1) * (\text{the value of CHUNK})$.
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, 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, 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:** 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 GB.
- **TIMESTAMP(*fractional_seconds_precision*):** Specifies 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, see the *Oracle Database SQL Reference*.

Creating and Modifying Tables

Database Instance: orcl.us.oracle.com > Tables > Logged in As SYS

Create Table [Show SQL] [Cancel] [OK]

General Constraints Storage Options Partitions

* Name jobs
Schema SHOPOWNER
Tablespace USERS
Organization Standard, Heap Organized
Wallet Status CLOSED

Define Using Column Specification [v]
[Set Default LOB Attributes]

[Advanced Attributes] [Delete] Insert Column: Abstract Data Type [v] [Insert] [Encryption Options]

Select	Name	Data Type	Size	Scale	Not NULL	Default Value	Encrypted
<input type="checkbox"/>	job_id	NUMBER	5		<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	job_title	VARCHAR2	30		<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	min_salary	NUMBER	6		<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	max_salary	NUMBER	6		<input type="checkbox"/>		<input type="checkbox"/>

Creating and Modifying Tables

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

Creating a Table

To create a table by using Enterprise Manager:

1. Click Tables on the Schema 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 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

```
CREATE TABLE SHOPOWNER.JOBS (  
  Job_id NUMBER(5),  
  Job_title VARCHAR2(30),  
  MIN_SALARY NUMBER(6),  
  MAX_SALARY NUMBER(6)  
)  
TABLESPACE USERS;
```

```
ALTER TABLE SHOPOWNER.JOBS add bonus NUMBER(6);
```

ORACLE

8 - 12

Copyright © 2007, Oracle. All rights reserved.

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 indicates that the table has been successfully created.

Modifying a Table

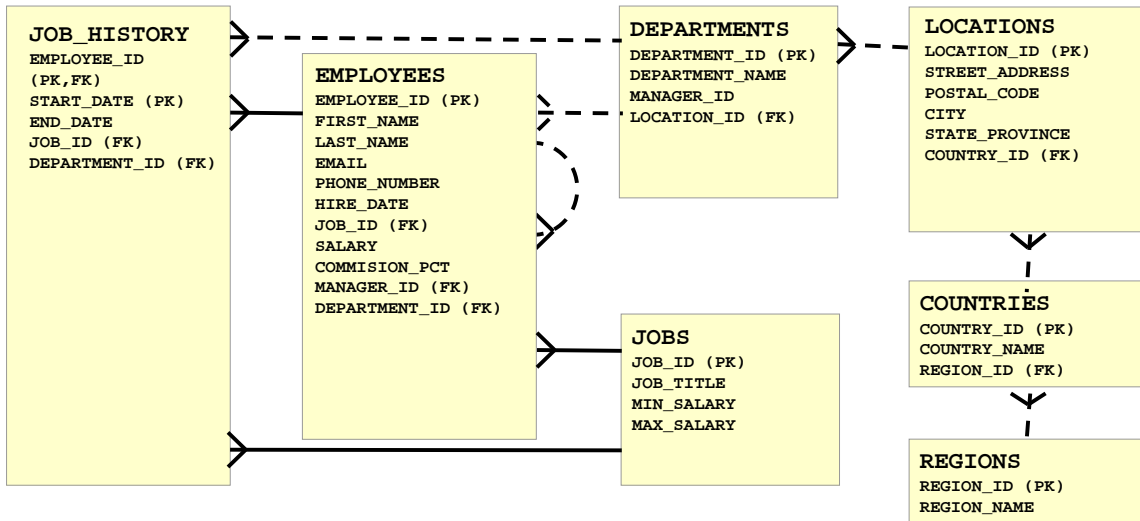
You can modify a table by using Enterprise Manager. For example, you add a column to a table as follows:

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. Then click Apply. An update message indicates that the table has been modified successfully.

You can also use SQL to create tables. In the example in the slide, you create the JOBS table in the SHOPOWNER schema, and the table is created in the USERS tablespace.

Understanding Data Integrity

Schema
> Constraints
Indexes
Views
Sequences
Temp Tables
Data Dict



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. The word *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 that are 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 a constraint evaluating to FALSE, the statement is rolled back if the constraint is IMMEDIATE. If the constraint is deferrable and is set to DEFERRED, rollback occurs at commit rather than at DML execution.

Defining Constraints

Add UNIQUE Constraint

Up to 32 columns can make up a UNIQUE key constraint. The unique key columns constitute a unique identifier for each row in the table.

Cancel Continue

Definition

Name <System Assigned 3>

Table Columns

Available Columns		Selected Columns
COUNTRY_ID REGION_ID	> Move	COUNTRY_NAME
	>> Move All	
	< Remove	
	<< Remove All	

Defining Constraints

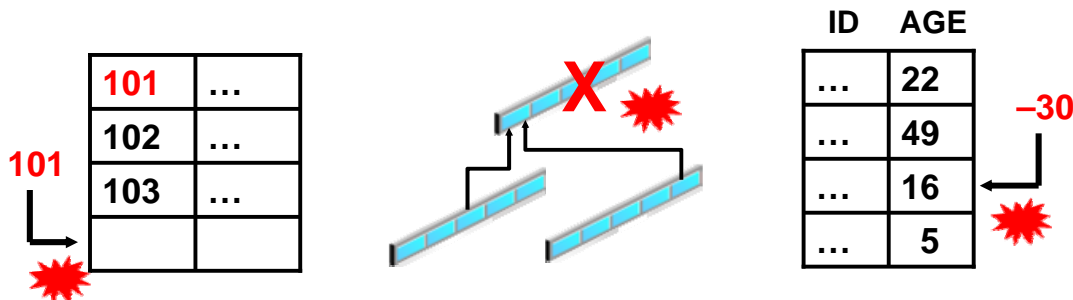
To add a constraint to a table by using Enterprise Manager:

1. Select the table on the Tables page and click Edit.
2. Click Constraints. The Constraints page shows 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 then 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:

- 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

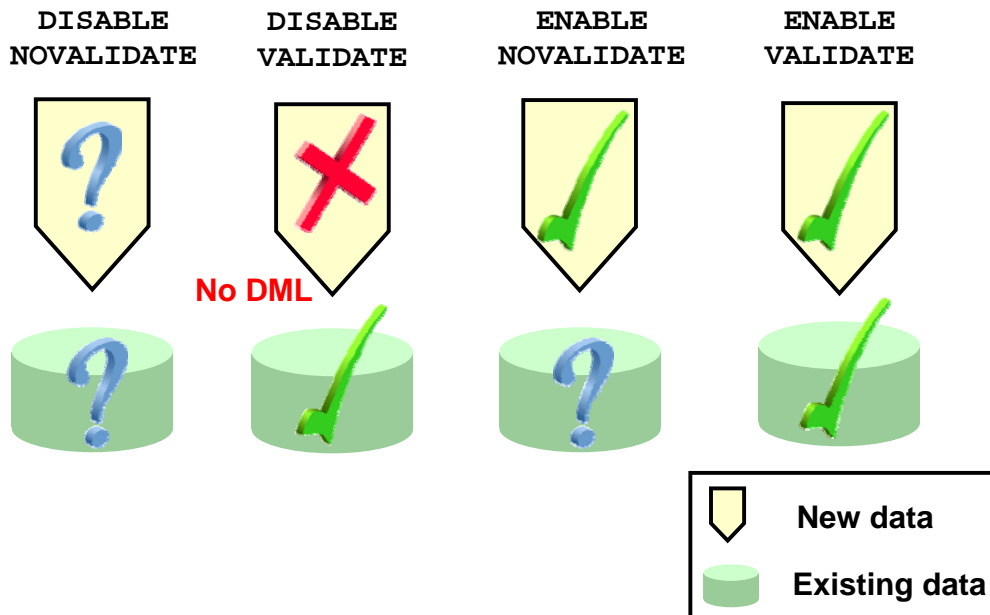


Constraint Violations

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

- **Uniqueness:** An attempt is made to have duplicate values in a column that has a unique constraint (for example, where a column is the primary key or is uniquely indexed).
- **Referential integrity:** The rule that every child row has 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 might have a check constraint that enforces it to be a positive number.

Constraint States



Constraint States

To better deal with situations in which 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, 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. NOVALIDATE is used in data warehousing situations where the data has already been cleaned up. No validation is needed, thereby saving time.

DISABLE VALIDATE: If a constraint is in this state, modification of the constrained columns is not allowed because it would be inconsistent to validate the existing data and then allow unchecked data to enter the table. This is often used when existing data must be validated but not modified and when 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 when it is known that clean and conforming data exists in the table so there is no need for validation. However, 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.

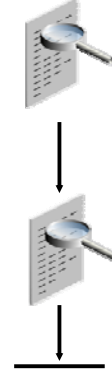
Constraint Checking

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



ORACLE

Constraint Checking

You can defer the checking of 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. For nondeferrable constraints, the primary key and unique key constraints need unique indexes; if the column or columns already have a non-unique index, constraint creation fails because those indexes cannot be used for a unique or primary key.

Deferred constraints are constraints that are checked only when a transaction is committed. If constraint violations are detected at commit time, 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 in which the order and the items in the order are entered at the same time. For deferrable constraints, primary key and unique keys need non-unique indexes; if the column or columns already have a unique index on them, constraint creation fails because those indexes cannot be deferred.

Constraint Checking (continued)

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.

Note: If an appropriate index already exists on the column, it is used for the constraint. An additional index does not need to be created for primary keys and unique keys.

Creating Constraints with SQL: Examples

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

b `ALTER TABLE shopowner.jobs ADD CONSTRAINT job_pk
PRIMARY KEY (job_id);`

c `CREATE TABLE emp (emp_no NUMBER PRIMARY KEY, Last_name
VARCHAR2(30), first_name VARCHAR2(30), dept_no NUMBER,
Mgr_no NUMBER, hire_date date, salary NUMBER,
CONSTRAINT Mgr_FK FOREIGN KEY (mgr_no) REFERENCES
emp(emp_no), CONSTRAINT ck1 CHECK (salary > 0));`

ORACLE

8 - 21

Copyright © 2007, Oracle. All rights reserved.

Creating Constraints with SQL: Examples

- 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 JOBS table. The constraint name is JOB_PK, and the primary key is the JOB_ID column.
- c. This statement defines constraints at the time the table is created, rather than using an ALTER TABLE statement later. The MGR_FK constraint ensures that the values in the MGR_NO column must be present in the primary key column of the table. The CK1 constraint ensures that the SALARY is greater than zero.

Creating Constraints with SQL: Examples (continued)

When a constraint is violated, you receive an error such as:

```
INSERT INTO emp
(Select employee_id , last_name, first_name,department_id,
manager_id, hire_date, salary FROM HR.employees where
department_id =30);
  2  INSERT INTO emp
*
ERROR at line 1:
ORA-02291: integrity constraint (SYS.MGR_FK) violated -
parent key not found
```

Note: Every constraint has a name. If a name is not specified while creating the constraint, a system-assigned name that starts with `SYS_` is provided.

Viewing the Columns in a Table

View Table: HR.DEPARTMENTS

Actions: Create Like [Go] [Edit] [OK]

General

Name: DEPARTMENTS
 Schema: HR
 Tablespace: EXAMPLE
 Organization: Standard, Heap Organized

Columns

	Name	Data Type	Size	Scale	Not NULL	Default Value	Encrypted
⚡	DEPARTMENT_ID	NUMBER	4		<input checked="" type="checkbox"/>		<input type="checkbox"/>
	DEPARTMENT_NAME	VARCHAR2	30		<input checked="" type="checkbox"/>		<input type="checkbox"/>
	MANAGER_ID	NUMBER	6		<input type="checkbox"/>		<input type="checkbox"/>
	LOCATION_ID	NUMBER	4		<input type="checkbox"/>		<input type="checkbox"/>

⚡ Indicates a Primary Key column
 ⚡ Indicates a Unique Key column

Constraints

Name	Type	Table Columns	Disabled	Deferrable	Initially Deferred	Validate	RELY	Check Condition	Referenced Schema	Referenced Table	Referenced Table Columns
DEPT_ID_PK	PRIMARY	DEPARTMENT_ID	NO	NO	NO	YES	NO				
DEPT_LOC_FK	FOREIGN	LOCATION_ID	NO	NO	NO	YES	NO		HR	LOCATIONS	LOCATION_ID
DEPT_MGR_FK	FOREIGN	MANAGER_ID	NO	NO	NO	YES	NO		HR	EMPLOYEES	EMPLOYEE_ID
DEPT_NAME_NN	CHECK	DEPARTMENT_NAME	NO	NO	NO	YES	NO	"DEPARTMENT_NAME" IS NOT NULL			

Indexes

Number of Indexes: 2

Schema	Index	Indexed Columns	Column Position

Viewing the Columns in a Table

To view the attributes of a table by using Enterprise Manager:

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.

You can also use the SQL*Plus DESCRIBE command to view the table descriptions:

```
SQL> desc hr.departments
Name                               Null?      Type
-----
DEPARTMENT_ID                       NOT NULL  NUMBER(4)
DEPARTMENT_NAME                       NOT NULL  VARCHAR2(30)
MANAGER_ID                             NUMBER(6)
LOCATION_ID                             NUMBER(4)
```

Viewing the Contents of a Table

View Data for Table: HR.DEPARTMENTS

Query: `SELECT "DEPARTMENT_ID", "DEPARTMENT_NAME", "MANAGER_ID", "LOCATION_ID"
FROM "HR"."DEPARTMENTS"`

Result:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700

Viewing the Contents of a Table

To view the rows in a table by using Enterprise Manager:

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

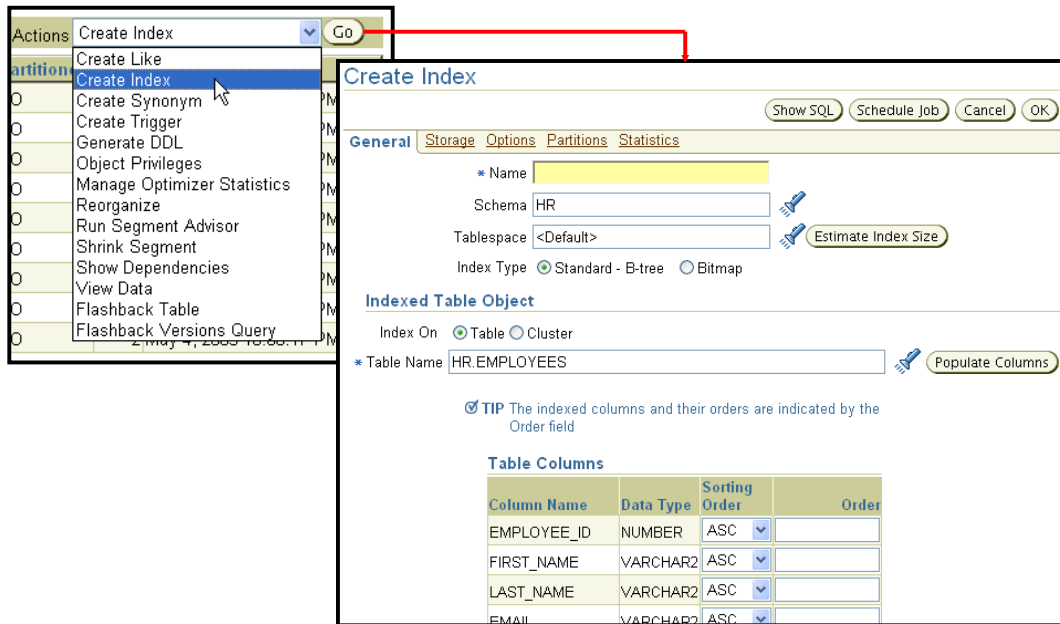
The View Data for Table page appears. The row data for the table is shown in the Result region. The Query box displays the SQL query that is executed to produce the results. On this page, you can click any column name and sort the data in the column in either ascending or descending order. If you want 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, see the *Oracle Database SQL Reference*.

You can also execute the query from a SQL prompt:

```
SQL> SELECT department_id , department_name, manager_id,  
manager_name  
FROM departments;
```


Actions with Tables



ORACLE

Actions with Tables

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

- **Create Like:** 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:** Create indexes on a table
- **Generate DDL:** Generate 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, only the owner can do anything with a table when it is created. The owner must grant privileges to other users for them to perform DML or possibly DDL on the table.
- **Show Dependencies:** Show objects that this table depends on or objects that depend on this table
- **View Data:** Select and display 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**

Dropping a Table

Syntax:

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

The DROP TABLE command removes data, the table structure, and associated object privileges and partitions if any exist.

Some DROP TABLE considerations:

- 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

- **Truncating a table removes the data and releases used space.**
- **Corresponding indexes are truncated.**

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

ORACLE

8 - 27

Copyright © 2007, Oracle. All rights reserved.

Truncating a Table

Syntax

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

Effects of using this command:

- 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.

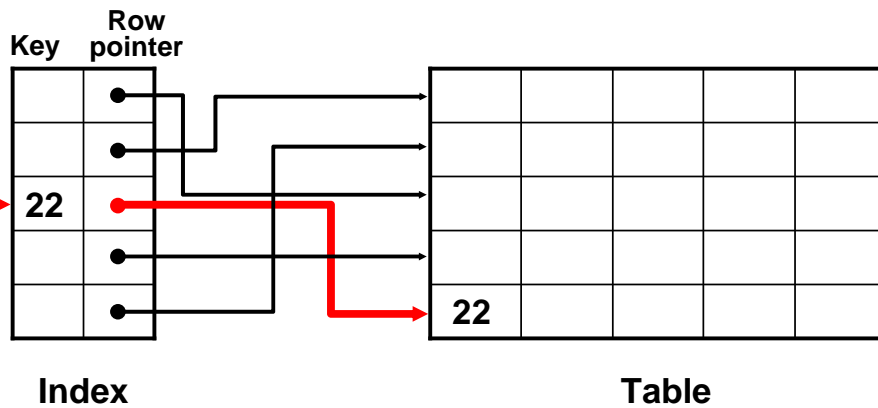
This is usually many times faster than issuing a DELETE statement to delete all the rows of the table due to the following reasons:

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

Indexes

Schema
Constraints
> Indexes
Views
Sequences
Temp Tables
Data Dict

... WHERE key = 22



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.

Indexes can also improve performance in the enforcement of primary key and unique key constraints. Without indexes, the entire table is scanned (full table scan) with each DML operation on the table.

Types of Indexes

These are several types of index structures that are available depending on your needs. Two of the most common are:

- **B-tree index**
 - Default index type; in the form of a balanced tree
- **Bitmap index:**
 - Has a bitmap for each distinct value indexed
 - Each bit position represents a row that may or may not contain the indexed value.
 - Best for low-cardinality columns

ORACLE

8 - 29

Copyright © 2007, Oracle. All rights reserved.

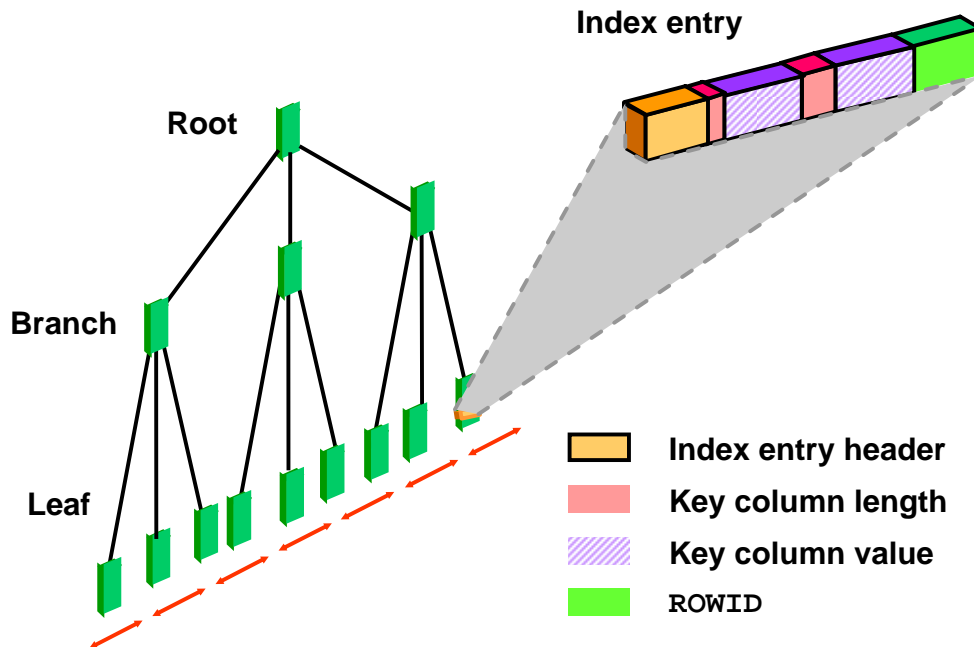
Types of Indexes

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

A bitmap index has a bitmap for each distinct key value being indexed. In 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. Use bitmap indexes for low-cardinality columns.

An example of this is a gender indicator that has values of “M” and “F” only. As a result, there are only two bitmaps to search. 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.

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 has the following components:

- **Entry header:** Stores the number of columns and locking information
- **Key column length-value pairs:** Define the size of a column in the key followed by the value for the column (The number of such pairs is the maximum of the number of columns in the index.)
- **ROWID:** Row ID 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 always results in a full table scan.
- A restricted ROWID is used to point to the rows of the table because all rows belong to the same segment

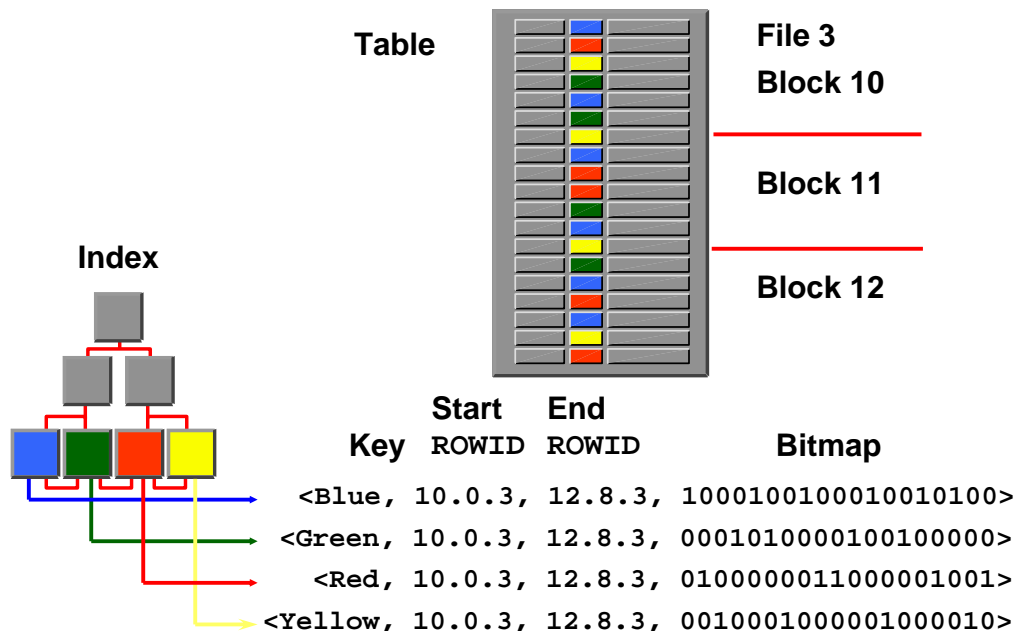
Effect of DML Operations on an Index

The Oracle server maintains all the indexes when DML operations are carried out on a 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 available for new sequential leaf entries.
- 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



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, when 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 row IDs. Each bit in the bitmap corresponds to a possible row ID, and if the bit is set, this means that the row with the corresponding row ID 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
- Key values consisting of length and value pairs for each key column (In the slide example, the key consists of only one column; the first entry has a key value of Blue).

Bitmap Indexes (continued)

Structure of a Bitmap Index (continued)

- Start ROWID, which in the example specifies block number 10, row number 0, and file number 3
- End ROWID, which in the example specifies block number 12, row number 8, and file number 3
- 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 row ID of the first row pointed to by the bitmap segment of the bitmap—that is, the first bit of the bitmap corresponds to that row ID, 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 row IDs.

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. The 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

- **Unique index:** Ensures that every indexed value is unique
- **Reverse key index:** Has its key value bytes stored in reverse order
- **Composite index:** Is based on more than one column
- **Function-based index:** Is based on a function's return value
- **Compressed index:** Has repeated key values removed
- **Order:** An index can have its key values stored in ascending or descending order.

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, 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));
```

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.

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

General | Storage | Options | Partitions

* Name:

Schema: HR

Tablespace: <Default>

Index Type: Standard - B-tree Bitmap

Estimated Index Size:

Indexed Table Object

* Table Name: HR.EMPLOYEES

TIP The indexed columns and their orders are indicated by the Order field

Column Name	Data Type	Sorting Order	Order
EMPLOYEE_ID	NUMBER	ASC	<input type="text"/>
FIRST_NAME	VARCHAR2	ASC	<input type="text"/>
LAST_NAME	VARCHAR2	ASC	<input type="text"/>

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

ORACLE

Creating Indexes

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 they can be created 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 is automatically created to enforce uniqueness on the column.

Views

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');
```

Views

Views are representations of queries of data from one or more tables or other views. Views are stored queries because they can hide very complex conditions and joins as well as 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

Database Instance: orcl.oracle.com > Views > Create View Logged in As SYS

Create View

Show SQL Cancel OK

General Options Object

* Name

* Schema

Aliases

Replace the view if exists

* Query Text
SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, MANAGER_ID,
DEPARTMENT_ID
FROM EMPLOYEES

**CREATE OR REPLACE VIEW "HR"."STAFF" AS SELECT EMPLOYEE_ID,
LAST_NAME, JOB_ID, MANAGER_ID, DEPARTMENT_ID
FROM EMPLOYEES**

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.

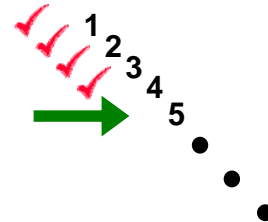
To see the views that are defined in the database, click the Views link under the Schema heading on the Administration page.

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.**



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

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

Note: If an application requires that there be no gaps, 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 if you 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. Gaps can also happen when cached values are aged out of the shared pool. The `DBMS_SHARED_POOL.KEEP` procedure allows the "Q" flag in the flag parameter to indicate that the name of the object being kept is a sequence. Doing this prevents the sequence from aging out.

Creating a Sequence

Create Sequence

Show SQL Cancel OK

General

* Name ABC_SEQ

* Schema HR

Values

* Maximum Value Value 100 Unlimited

* Minimum Value Value 1 Unlimited

* Interval 5

* Initial 10

Options

Cycle Values - Sequence will wrap around on reaching limit

Order Values - Sequence numbers will be generated in order

Cache Options

Use Cache

Cache Size 20

Show SQL

Return

```
CREATE SEQUENCE "HR"."ABC_SEQ" CYCLE NOORDER CACHE 20
MAXVALUE 100 MINVALUE 1 INCREMENT BY 5 START WITH 10
```

ORACLE

8 - 40

Copyright © 2007, Oracle. All rights reserved.

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:** Name of the sequence, which is how it is referenced
- **Schema:** Owner of the sequence
- **Maximum Value:** Specifies 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:** Specifies 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:** Specifies the interval between sequence numbers. This integer value can be any positive or negative integer, but it cannot be 0. It can have 28 or fewer digits. The default value is 1.
- **Initial:** Specifies 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:** Guarantees that sequence numbers are generated in the order of request. This clause is useful if you are using sequence numbers as time stamps. 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 `CACHE` option with Oracle Database with Real Application Clusters. If you are not caching, the sequence is in order by default.
- **Cache Options:** Specifies 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 would fit in a given cycle of sequence numbers.

Using a Sequence

```
SQL> CREATE TABLE orders
  (id NUMBER,
   ord_date DATE,
   prod_id NUMBER,
   prod_desc VARCHAR2(30)
  );
Table created.

SQL> INSERT INTO orders VALUES ( abc_seq.NEXTVAL,
sysdate, 1245009, 'Gizmo X');

1 row created.
```

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

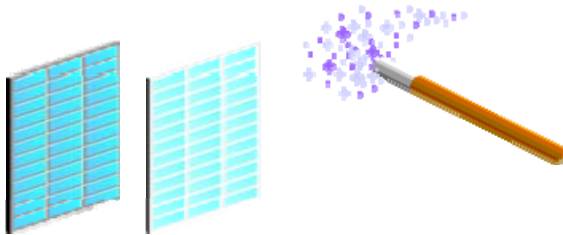
The first reference to NEXTVAL returns the initial value of the sequence. Subsequent references to NEXTVAL increment the sequence value by the defined increment and return the new value. A reference to CURRVAL always returns the current value of the sequence, which is the value returned by the last reference 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 for use to all sessions without affecting the private data of each session



Temporary Tables

You can take advantage of temporary tables when you need to store data privately 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 that is 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. 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: 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

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

Temporary tables can be created with Enterprise Manager by clicking the Temporary option on the Create Table: Table Organization page. Click Continue. 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.

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

ORACLE

Practice 8 Overview: Administering Schema Objects

This practice covers the following topics:

- **Creating tables with columns**
- **Creating constraints:**
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK
- **Creating indexes**

Managing Data and Concurrency



ORACLE

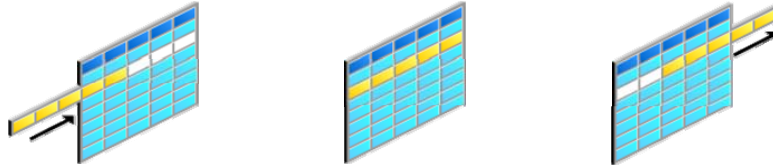
Copyright © 2007, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- **Manage data by using SQL**
- **Identify and administer PL/SQL objects**
- **Describe triggers and triggering events**
- **Monitor and resolve locking conflicts**

Manipulating Data by Using SQL



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

1 row created.

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

1 row updated.

SQL> DELETE from employees
  2  WHERE EMPLOYEE_ID = 9999;

1 row deleted.
```

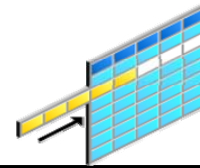
ORACLE

Manipulating Data by Using SQL

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

INSERT Command

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



```
SQL> INSERT INTO emp VALUES  
      (9999,'Bob','Builder',100,SYSDATE, 10000);
```

1

```
SQL>INSERT INTO emp(select employee_id , last_name,  
first_name,department_id, hire_date, salary FROM HR.employees  
where department_id =30);
```

2

```
6 rows created.
```

ORACLE

9 - 4

Copyright © 2007, Oracle. All rights reserved.

INSERT Command

The basic INSERT statement creates one row at a time. Using what is called a *subselect*, you cause the INSERT command to copy rows from one table to another. This method is also referred to as an INSERT SELECT statement. The first example in the slide inserts one row into the EMP table.

In this case, the EMP 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. The column values match in the same order named in the INSERT and SELECT statements. All that is required is that the data types match:

```
INSERT INTO emp (first_name, last_name)  
      (SELECT first_name, last_name From employees);
```

In this example, the columns that are populated are only the two columns in the EMP table.

Using the INSERT SELECT method is a way to load data in bulk from one or more tables into another table. This is shown in the second example. Here you are selecting all rows in department 30 from the EMPLOYEES table and inserting them into the EMP table.

UPDATE Command

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

```
SQL> UPDATE EMP SET SALARY= salary +500  
Where emp_no =117;
```

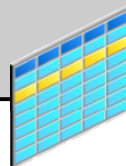
1

1 row updated.

```
SQL> UPDATE EMP  
Set dept_no= (select dept_no from emp where emp_no =117);
```

2

6 rows updated.



ORACLE

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, all rows are changed. If no rows satisfy the WHERE condition, no rows are modified. In the slide, the first example shows an update to EMP that changes the salary for employee 117.

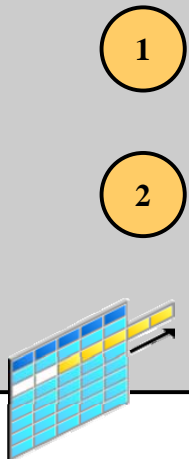
In the second example, you set all department numbers to be the same as the department of employee 117.

DELETE Command

Use the **DELETE** command to remove zero or more rows from a table.

```
SQL> DELETE FROM emp WHERE emp_no =9000;
0 rows deleted.

SQL> DELETE emp;
6 rows deleted.
```



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, all rows are removed. If no rows satisfy the **WHERE** condition, no rows are removed.

In the first example, it is not an error when no rows are deleted. The message that is returned states only that zero rows were removed from the table.

In the second example, a **WHERE** clause is not specified. All rows are therefore deleted.

MERGE Command

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

```
MERGE INTO EMP a USING
  (Select * FROM HR.employees) b
ON (a.emp_no =b. employee_id )
WHEN MATCHED THEN UPDATE SET a.salary =b.salary
WHEN NOT MATCHED THEN INSERT
  (emp_no, last_name, first_name, dept_no, hire_date,
  salary)
VALUES
  (employee_id,last_name, first_name, department_id,
  hire_date, salary);
```



ORACLE

MERGE Command

The **MERGE** command performs **UPDATE**, **INSERT**, or **DELETE** 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 the following example of some of the data in the **EMPS** table:

EMPNO	LAST_NAME	FIRST_NAME	DEPT_NO	HIRE_DATE	SALARY
114	Raphaely	Den	30	07-DEC-94	11000
115	Khoo	Alexander	30	18-MAY-95	3100
116	Baida	Shelli	30	24-DEC-97	2900
117	Tobias	Sigal	30	24-JUL-97	2800
118	Himuro	Guy	30	15-NOV-98	2600
119	Colmenares	Karen	30	10-AUG-99	2500

MERGE Command (continued)

The EMPLOYEES table has more rows and has the following description:

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

The MERGE command inserts into the EMPS table any rows with a new EMPLOYEE_ID, and updates the existing EMPS table row with new values for the existing columns if the row already exists.

COMMIT and ROLLBACK Commands

To finish a transaction:

- COMMIT makes the change permanent
- ROLLBACK undoes the change

```
SQL> COMMIT;  
Commit complete.
```

COMMIT and ROLLBACK Commands

By default, each DML command that is entered is not committed. Several tools (including *iSQL*Plus*) have options that can be set to commit on each command or 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 they 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 that is 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 the locking mechanism in Oracle Database. Because the locking mechanism is built into the row itself, there is no way for the database to run out of locks.

PL/SQL

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

9 - 10

Copyright © 2007, Oracle. All rights reserved.

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 declare constants and variables, define procedures and functions, use collections and object types, and trap run-time errors. A PL/SQL program can call programs written in other languages, such as C, C++, and Java.

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

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

For details about procedural constructs and uses of PL/SQL, see the *PL/SQL Packages and Types Reference*.

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
- Use supplied packages for administrative and database maintenance tasks



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 sufficiently familiar with the different PL/SQL objects 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:** 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 that are available for use.
- **Package body:** Fully defines cursors and subprograms, and thus implements the spec. The body contains implementation details and private declarations that are hidden from the caller.
- **Type body:** Collection of methods (procedures and functions) associated with user-defined data types. For more information about user-defined data types, see the *Oracle Database Object-Relational Developer's Guide*.
- **Procedure:** PL/SQL block that performs a specific action
- **Function:** PL/SQL block that returns a single value by using the PL/SQL RETURN command. It is a procedure that has a return value.
- **Trigger:** 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

Create Function

- * Name: compute_tax
- * Schema: hr
- * Source:

```
( salary in number
)
return number
as
begin
  if salary < 5000 then
    return salary * 0.15;
  else
    return salary * 0.33;
  end if;
end;
```

Functions

Object Type: Function

Search

Select an object type and optionally enter a schema name and an object name to filter the data that is displayed in your results set.

Schema: DBA1

Object Name:

Status: All

Go

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Create

9 - 13

Copyright © 2007, Oracle. All rights reserved.

ORACLE

Functions

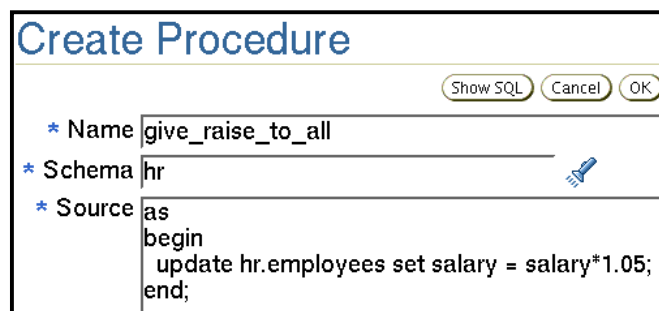
PL/SQL functions are typically used to compute a value. There are many built-in functions, such as SYSDATE, SUM, AVG, and TO_DATE. Developers also create their own functions when writing applications. The code for a PL/SQL function must contain a RETURN statement. PL/SQL functions are created by entering 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 specific actions
- Pass values in and out by using an argument list
- Can be called from within other programs using the `CALL` command



The screenshot shows a dialog box titled "Create Procedure". It has three input fields: "Name" with the value "give_raise_to_all", "Schema" with the value "hr", and "Source" with the code "as begin update hr.employees set salary = salary*1.05; end;". There are buttons for "Show SQL", "Cancel", and "OK".

Procedures

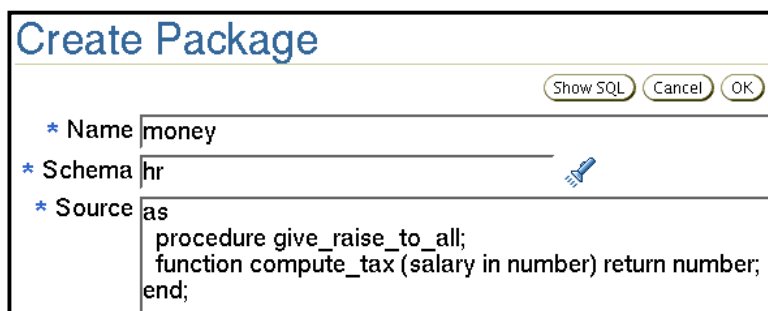
PL/SQL procedures perform a specific action. Like functions, procedures can accept input values and perform conditional statements such as `IF-THEN`, `CASE`, and `LOOP`.

Note: Procedures do not necessarily require input parameters and do not need to return anything.

Packages

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

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



Create Package	
* Name	money
* Schema	hr
* Source	as procedure give_raise_to_all; function compute_tax (salary in number) return number; end;

Package specification



ORACLE

9 - 15

Copyright © 2007, Oracle. All rights reserved.

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 typically comprises 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.

Note: The specification must exist, the body is optional

Procedures and functions that are 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

Create Package

Show SQL Cancel OK

* Name money

* Schema hr

* Source as

```
procedure give_raise_to_all;
function compute_tax
end;

function compute_tax (salary in number) return number
as
begin
  if salary < 5000 then
    return salary * 0.15;
  else
    return salary * 0.33;
  end if;
end;

procedure give_raise_to_all
as
begin
  update hr.employees set salary = salary*1.05;
end;
end;
```

9 - 16

Copyright © 2007, Oracle. All rights reserved.

ORACLE

Package Specification and 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 Packages and Types Reference*.

Built-in Packages

- Oracle Database comes with several built-in PL/SQL packages that provide:
 - Administration and maintenance utilities
 - Extended functionality
- Use the `DESCRIBE` command to view subprograms.

```
SQL>DESC DBMS_LOCK
```

```
View Package: SYS.DBMS_LOCK

Name DBMS_LOCK
Schema SYS
Status Valid
Methods
procedure allocate_unique( lockname in varchar2, lockhandle out varchar2, expiration_secs in integer);
function request( kl in integer, lockmode in integer, timeout in integer, release_on_commit in boolean);
function request( lockhandle in varchar2, lockmode in integer, timeout in integer, release_on_commit in boolean);
function convert( kl in integer, lockmode in integer, timeout in number);
function convert( lockhandle in varchar2, lockmode in integer, timeout in number);
function release( kl in integer);
function release( lockhandle in varchar2);
procedure sleep( seconds in number);

Number of Procedures: 2, Number of Functions: 6.
Source
is
-----
-- OVERVIEW
--
-- These routines allow the user to request, convert and release locks.
-- The locks are managed by the rdbms lock management services. All
```

Built-in Packages

The built-in PL/SQL packages that are supplied with 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 PL/SQL administration and maintenance packages:

- **DBMS_STATS:** Gathering, viewing, and modifying optimizer statistics
- **DBMS_OUTPUT:** Generating output from PL/SQL
- **DBMS_SESSION:** Accessing the `ALTER SESSION` and `SET ROLE` statements
- **DBMS_SHARED_POOL:** Managing the shared pool (for example, flushing it)
- **DBMS_UTILITY:** Getting time, CPU time, and version information; computing a hash value; performing many other miscellaneous functionalities
- **DBMS_SCHEDULER:** Scheduling functions and procedures that are callable from PL/SQL
- **DBMS_REDEFINITION:** Redefining objects online
- **UTL_FILE:** Reading and writing 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*.

Triggers

Triggers

Object Type: Trigger

Search

Enter a schema name and an object name to filter the data that is displayed in your results set.

Schema: HR

Object Name:

Status: All

Go

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Selection Mode: Single

Create

Edit View Delete Actions Create Like Go

Select	Schema	Trigger Name	Type	Event	Base Object Type	Base Object Owner	Base Object Name	Status	Enabled?
	HR	SECURE_EMPLOYEES	BEFORE STATEMENT	INSERT OR UPDATE OR DELETE	TABLE	HR	EMPLOYEES	VALID	NO
	HR	UPDATE_JOB_HISTORY	AFTER EACH ROW	UPDATE	TABLE	HR	EMPLOYEES	VALID	YES

ORACLE

9 - 18

Copyright © 2007, Oracle. All rights reserved.

Triggers

Triggers are PL/SQL code objects that are stored in the database and that 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

ORACLE

Triggering Events

There are three categories of events that can be used to fire a trigger:

- 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 specified triggering events occur 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

- Prevent multiple sessions from changing the same data at the same time
- Are automatically obtained at the lowest possible level for a given statement
- 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;
```

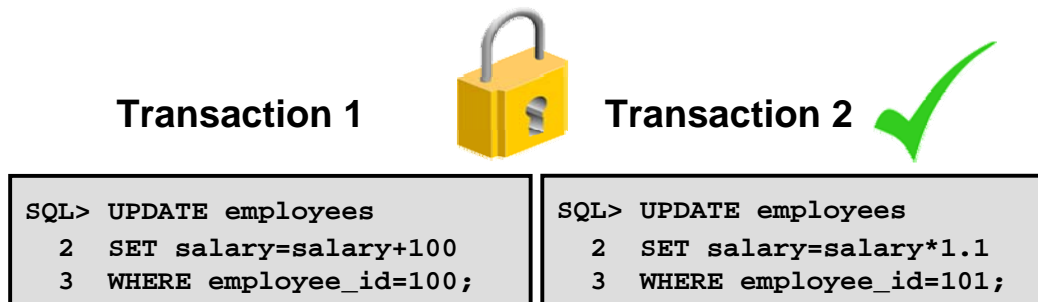
Locks

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 supports both manual and automatic locking. Automatically acquired locks always choose the lowest possible level of locking 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)**



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 automatic locking is released on commit. Transactions are completed when a COMMIT or ROLLBACK is issued. In the case of a failed transaction, the same background process that automatically rolls back any changes from the failed transaction releases all locks held by that transaction.

Data Concurrency (continued)

Share locks allow multiple readers and no writers. Share locks are also used transparently when deleting or updating rows in a parent table that has a child table with foreign key constraints on the parent.

- **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. Oracle recommends that you avoid using manual locks, especially when developing applications. Severe performance issues often occur from unnecessarily high locking levels.

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

ORACLE

DML Locks

Each DML transaction obtains two locks:

- An **EXCLUSIVE** row lock on the row or rows being updated
- A **ROW EXCLUSIVE** table-level lock on the table being updated. This prevents another session from locking the whole table (possibly to drop or truncate it) while the change is being made.

A **ROW EXCLUSIVE** lock on the table stops DDL from changing the dictionary metadata in the middle of an uncommitted transaction. This preserves dictionary integrity and read consistency across the life of a transaction.

Enqueue Mechanism

The enqueue mechanism keeps track of:

- Sessions waiting for locks
- Requested lock mode
- Order in which sessions requested the lock



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 that the lock be *converted* 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 that the `SHARE` lock be converted to an `EXCLUSIVE` lock. If nobody 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.

Note: There are two categories of waiters for enqueues: those waiting without shared ownership and those with shared ownership that do not choose to escalate the lock level. Waiters in the second category are known as *converters*, which are always given priority over normal waiters even if they have been waiting less time.

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 COMMISSION_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;

ORACLE

9 - 26

Copyright © 2007, Oracle. All rights reserved.

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.

A 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:

- **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 that are intended to run on many different databases often write their applications with artificially high locking levels so that 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 are required by Oracle Database.

Detecting Lock Conflicts

Select Blocking Sessions on the Performance page.

Select	Username	Sessions Blocked	Session ID	Serial Number	SQL ID	Wait Class	Wait Event	P1 Value	P2 Value	P3 Value	Seconds in Wait
▼	Blocking Sessions										
▼	NGREENBERG	1	170	17463		Idle	SQL*Net message from client	1650815232	1	0	90
▼	SMAVRIS	0	124	20568	6smqtv6h8958b	Application	enq: TX - row lock contention	1415053318	458762	691	39

Click the Session ID link to view information about the locking session, including the actual SQL statement.

ORACLE

Detecting Lock Conflicts

Use the Blocking Sessions page in Enterprise Manager to locate lock conflicts. Conflicting lock requests are shown in hierarchical layout, showing the session holding the lock at the top and, below that, any sessions that are enqueued for the lock.

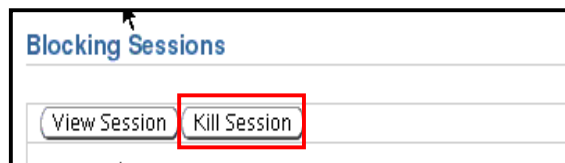
For each session involved in the conflict, you are given the username, session ID, and number of seconds for which the session has been waiting. Drill down to the session ID to see actual SQL statements that are currently being executed or requested by the session.

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 (in an emergency)



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
```

Note: The PMON session sniper can automatically kill sessions due to idle timeout, and this can be done by using profiles or Resource Manager.

Resolving Lock Conflicts with 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, like most other tasks 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. If you query for `SID` and `SERIAL#` (where `SID` matches a blocking session ID), you have the information needed to perform the `kill session` operation.

Note: Database Resource Manager can be used to automatically log out sessions that block others and are idle.

Deadlocks

Transaction 1		Transaction 2
<code>UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 1000;</code>	9:00	<code>UPDATE employees SET manager = 1342 WHERE employee_id = 2000;</code>
<code>UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 2000;</code>	9:15	<code>UPDATE employees SET manager = 1342 WHERE employee_id = 1000;</code>
<code>ORA-00060: Deadlock detected while waiting for resource</code>	9:16	

Deadlocks

A deadlock is a special example of a lock conflict. Deadlocks arise when two or more sessions wait for data that has been locked by the other. Because each is waiting for the other, neither can complete their transaction to resolve the conflict.

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 must 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 by using SQL**
- **Identify and administer PL/SQL objects**
- **Describe triggers and triggering events**
- **Monitor and resolve locking conflicts**

Practice 9 Overview: Managing Data and Concurrency

This practice covers the following topics:

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

