



---

## Oracle9i DBA Fundamentals I

### Electronic Presentation

---

D11321GC10  
Production 1.0  
May 2001  
D32645

**ORACLE®**

**ORACLE®**

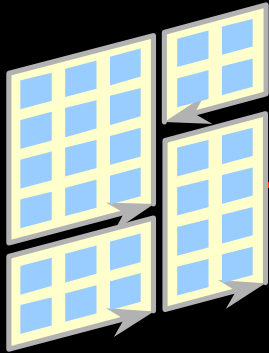
# I Introduction

# Objectives

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

- **Identify the various components of the Oracle architecture.**
- **Start and shut down an Oracle database**
- **Create an operational database**
- **Manage Oracle control files, redo log files, database files, tablespaces, segments, extents, and blocks**
- **Manage users, privileges, and resources**
- **Use Globalization Support features**

# Oracle9i Enterprise Edition



**Partitioning**

ORACLE



**Oracle Enterprise  
Manager Packs**

ORACLE



INTERNET



**Real Application  
Clusters**

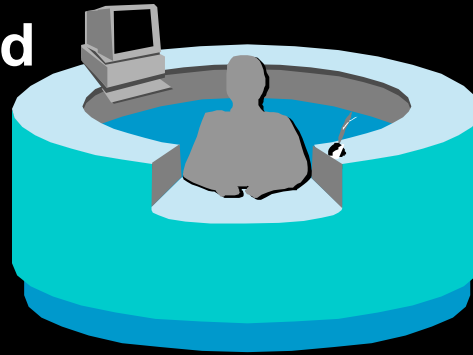


**Advanced  
Security**

ORACLE

# Database Administrator Tasks

- Plan and create databases
- Manage database availability
- Manage physical and logical structures
- Manage storage based on design
- Manage security



- Network administration
- Backup and recovery
- Database tuning

# 1

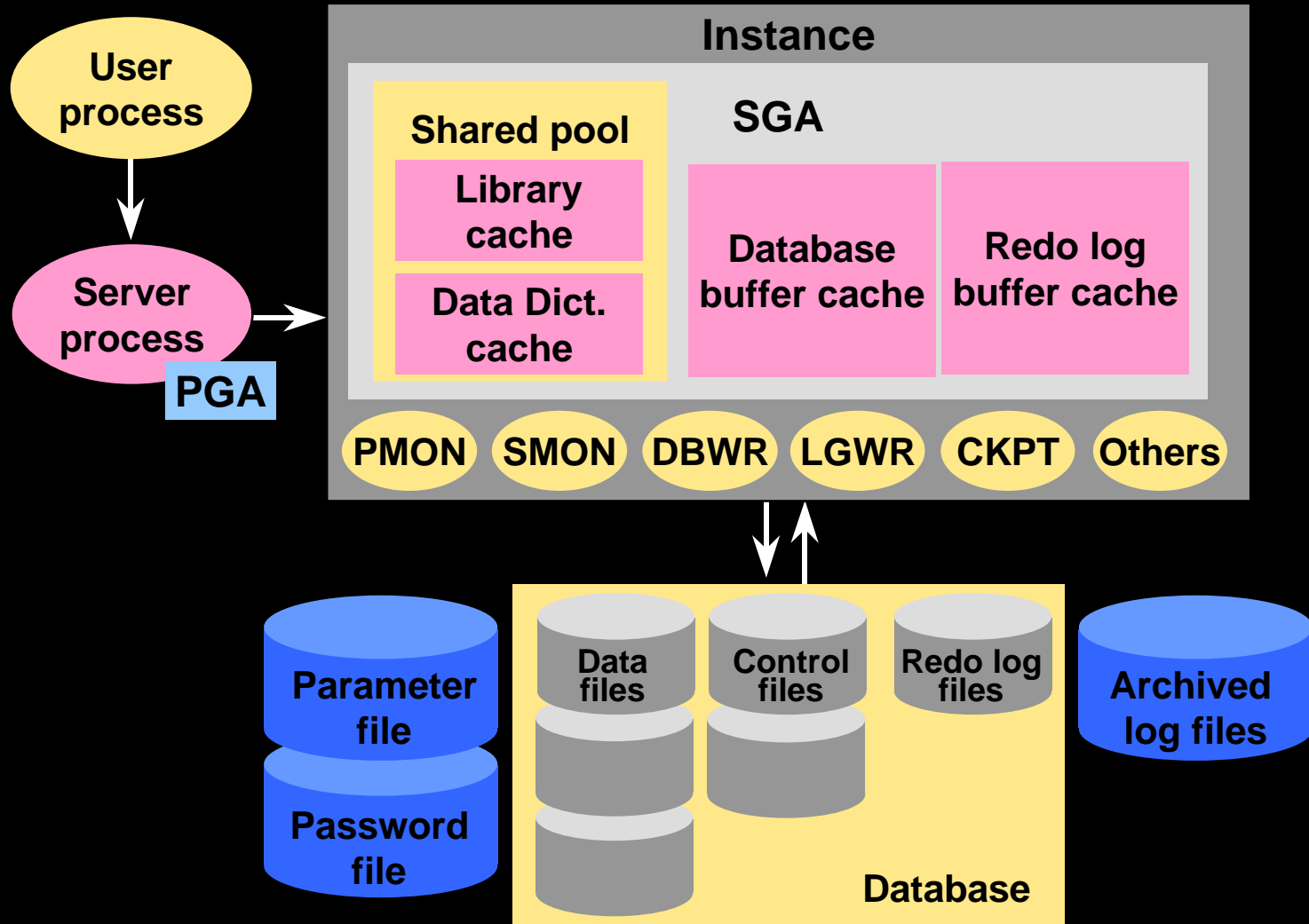
## Oracle Architectural Components

# Objectives

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

- **Outline the Oracle architecture and its main components**
- **List the structures involved in connecting a user to an Oracle instance**

# Overview of Primary Components

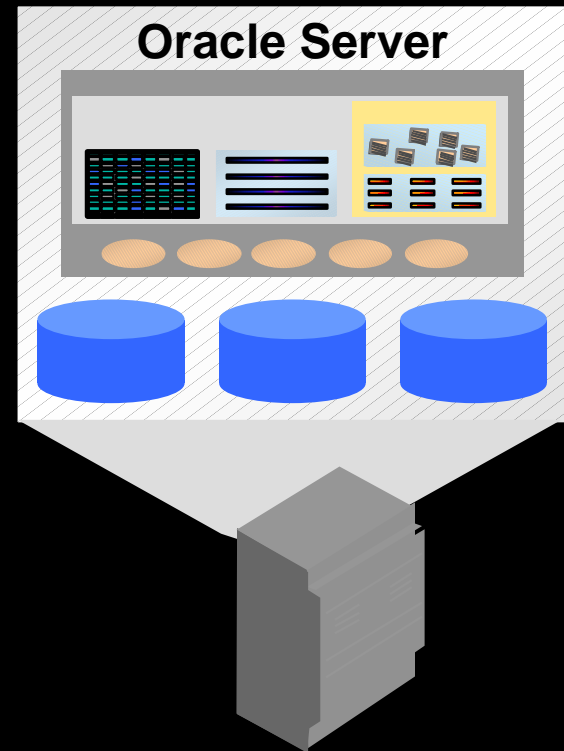




# Oracle Server

## An Oracle server:

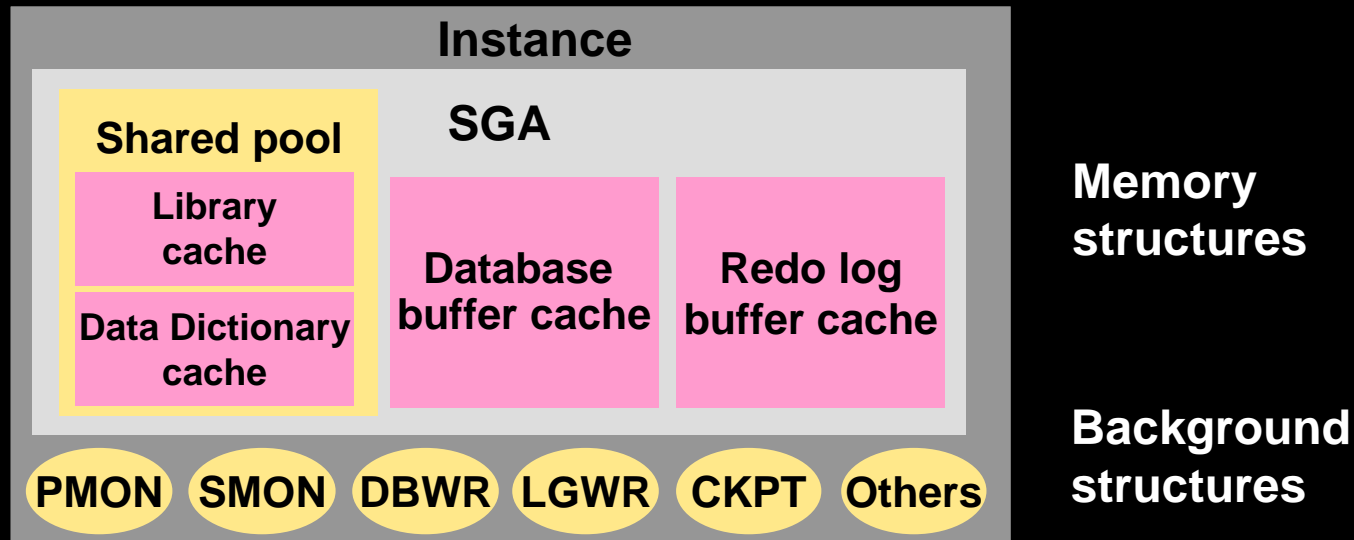
- Is a database management system that provides an open, comprehensive, integrated approach to information management
- Consists of an Oracle instance and an Oracle database



# Oracle Instance

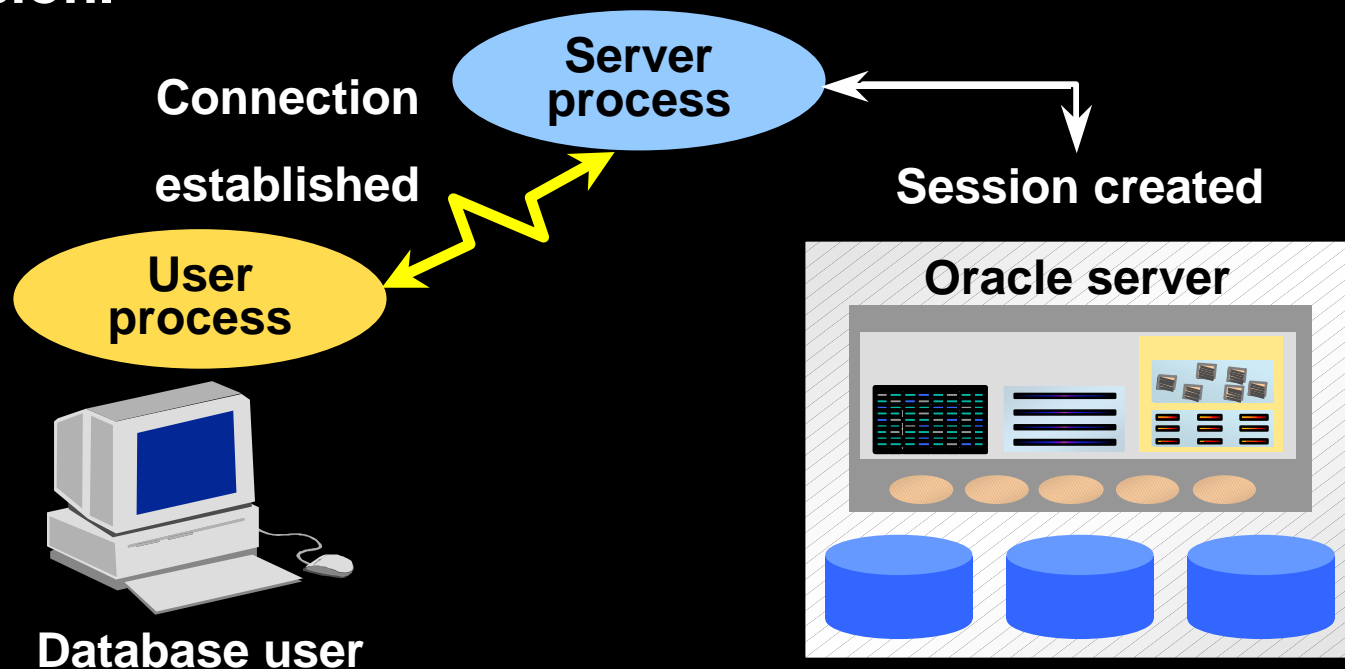
**An Oracle instance:**

- **Is a means to access an Oracle database**
- **Always opens one and only one database**
- **Consists of memory and process structures**



# Establishing a Connection and Creating a Session

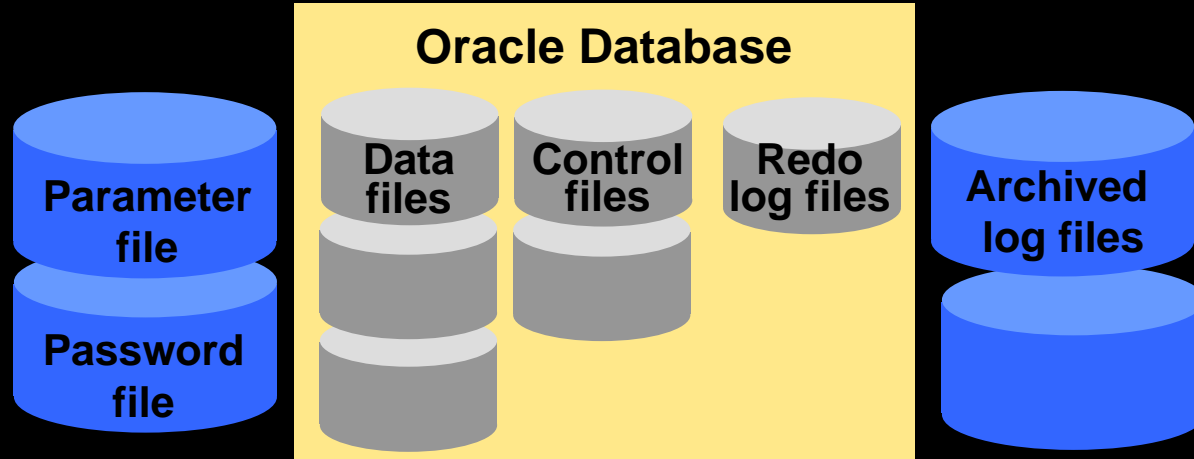
Connecting to an Oracle instance consists of establishing a user connection and creating a session.



# Oracle Database

**An Oracle database:**

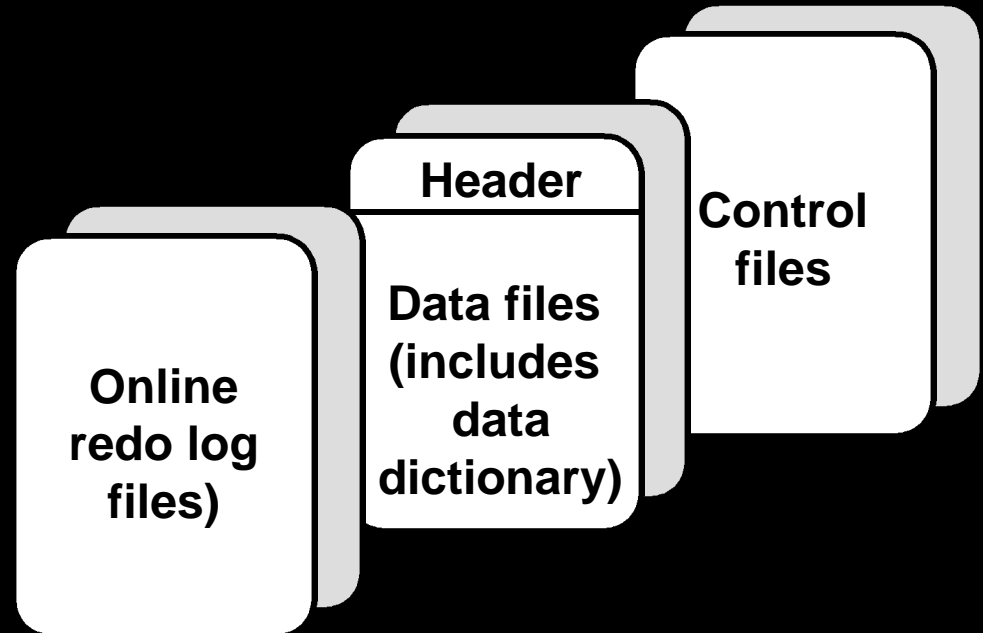
- **Is a collection of data that is treated as a unit**
- **Consists of three file types**



# Physical Structure

The physical structure of an Oracle database is determined by the operating system files that provide the actual physical storage for database information.

- Control files
- Data files
- Redo log files



# Memory Structure

**Oracle's memory structure consists of two memory areas known as:**

- **System Global Area (SGA):** Allocated at instance startup, and is a fundamental component of an Oracle Instance
- **Program Global Area (PGA):** Allocated when the server process is started

# System Global Area (SGA)

- **The SGA consists of several memory structures:**
  - Shared pool
  - Database buffer cache
  - Redo log buffer
  - Other structures (e.g. lock and latch management, statistical data)
- **There are two optional memory structures that can be configured within the SGA:**
  - Large pool
  - Java pool

# System Global Area (SGA)

- **SGA is dynamic and sized using `SGA_MAX_SIZE`.**
- **SGA memory allocated and tracked in granules by SGA components**
  - **Contiguous virtual memory allocation**
  - **Size based on `SGA_MAX_SIZE`**

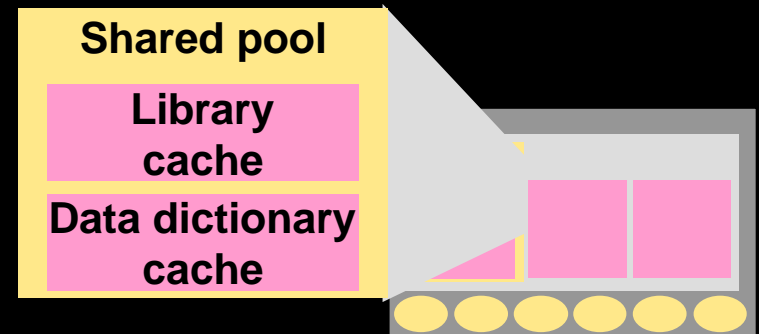


# Shared Pool

The shared pool is used to store the most recently executed SQL statements and the most recently used data definitions.

- It consists of two key performance-related memory structures:
  - Library cache
  - Data dictionary cache
- Sized by the parameter `SHARED_POOL_SIZE`.

```
ALTER SYSTEM SET  
SHARED_POOL_SIZE = 64M;
```



# Library Cache

**The library cache stores information about the most recently used SQL and PL/SQL statements. The library cache:**

- **Enables the sharing of commonly used statements**
- **Is managed by a least recently used (LRU) algorithm**
- **Consists of two structures:**
  - **Shared SQL area**
  - **Shared PL/SQL area**
- **Has its size determined by the shared pool sizing**

# Data Dictionary Cache

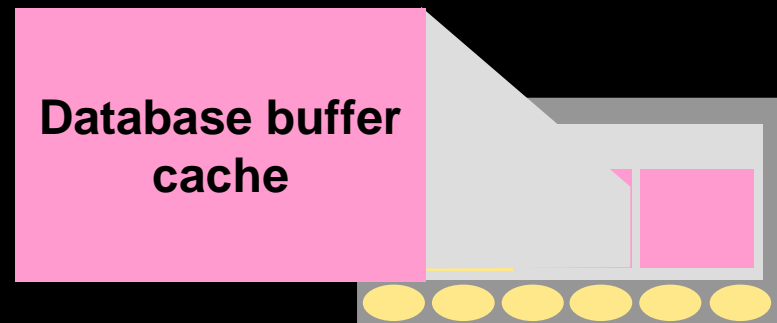
**The data dictionary cache is a collection of the most recently used definitions in the database.**

- **It includes information about database files, tables, indexes, columns, users, privileges, and other database objects.**
- **During the parse phase, the server process looks at the data dictionary for information to resolve object names and validate access.**
- **Caching the data dictionary information into memory improves response time on queries.**
- **Size is determined by the shared pool sizing.**

# Database Buffer Cache

**The database buffer cache stores copies of data blocks that have been retrieved from the data files.**

- **It enables great performance gains when you obtain and update data.**
- **It is managed through a least recently used (LRU) algorithm.**
- **DB\_BLOCK\_SIZE determines the primary block size.**



# Database Buffer Cache

- **Consists of independent sub-caches:**
  - `DB_CACHE_SIZE`
  - `DB_KEEP_CACHE_SIZE`
  - `DB_RECYCLE_CACHE_SIZE`
- **Database buffer cache can be dynamically resized to grow or shrink using `ALTER SYSTEM`.**

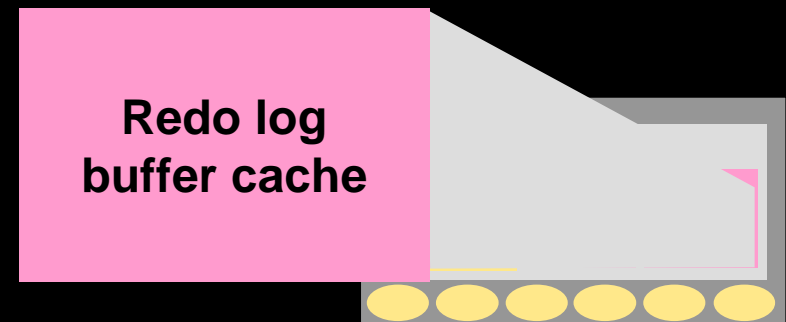
```
ALTER SYSTEM SET DB_CACHE_SIZE = 96M;
```

- **`DB_CACHE_ADVICE` can be set to gather statistics for predicting different cache size behavior.**

# Redo Log Buffer Cache

The redo log buffer cache records all changes made to the database data blocks.

- Its primary purpose is recovery.
- Changes recorded within are called redo entries.
- Redo entries contain information to reconstruct or redo changes.
- Size is defined by LOG\_BUFFER.



# Large Pool

The large pool is an optional area of memory in the SGA configured only in a shared server environment.

- It relieves the burden placed on the shared pool.
- This configured memory area is used for session memory (UGA), I/O slaves, and backup and restore operations.
- Unlike the shared pool, the large pool does not use an LRU list.
- Sized by `LARGE_POOL_SIZE`.

```
ALTER SYSTEM SET LARGE_POOL_SIZE = 64M;
```

# Java Pool

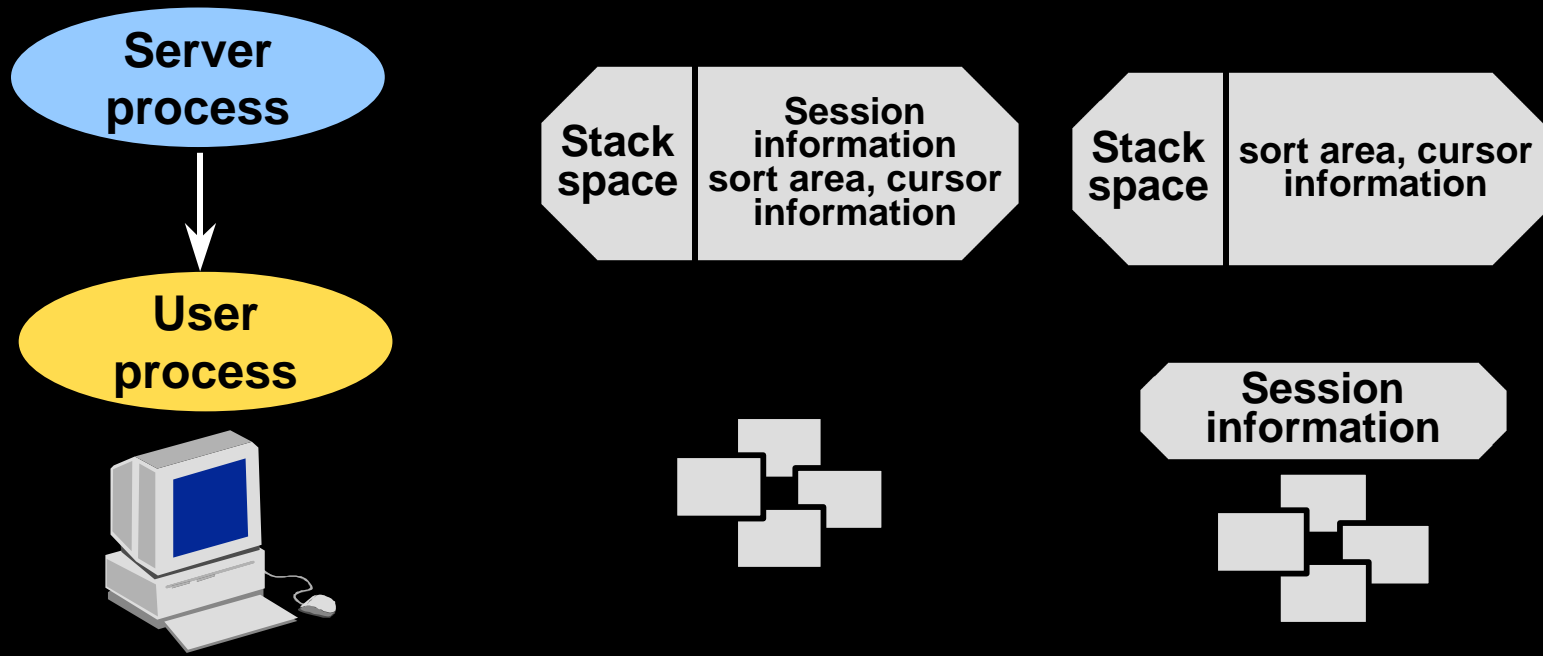
**The Java pool services the parsing requirements for Java commands.**

- **Required if installing and using Java.**
- **It is stored much the same way as PL/SQL in database tables.**
- **It is sized by the `JAVA_POOL_SIZE` parameter.**



# Program Global Area (PGA)

The PGA is memory reserved for each user process that connects to an Oracle database.



# Process Structure

**An Oracle process is a program that depending on its type can request information, execute a series of steps, or perform a specific task.**

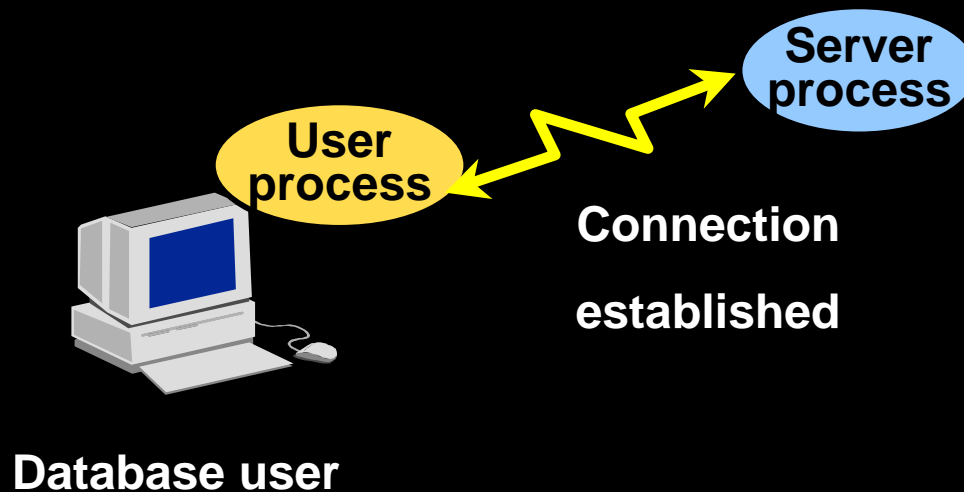
**Oracle takes advantage of various types of processes:**

- **User process: Started at the time a database user requests connection to the Oracle server**
- **Server process: Connects to the Oracle Instance and is started when a user establishes a session.**
- **Background process: Available when an Oracle instance is started**

# User Process

A user process is a program that requests interaction with the Oracle server.

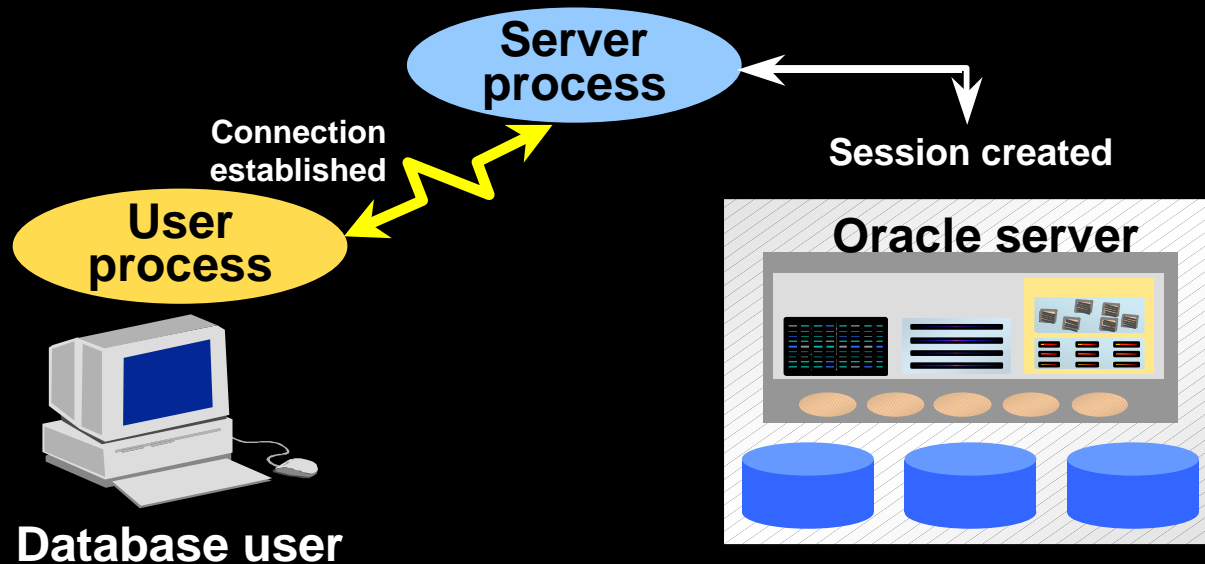
- It must first establish a connection.
- It does not interact directly with the Oracle server.



# Server Process

A server process is a program that directly interacts with the Oracle server.

- It fulfills calls generated and returns results.
- Can be dedicated or shared server.



# Background Processes

The relationship between the physical and memory structures is maintained and enforced by Oracle's background processes.

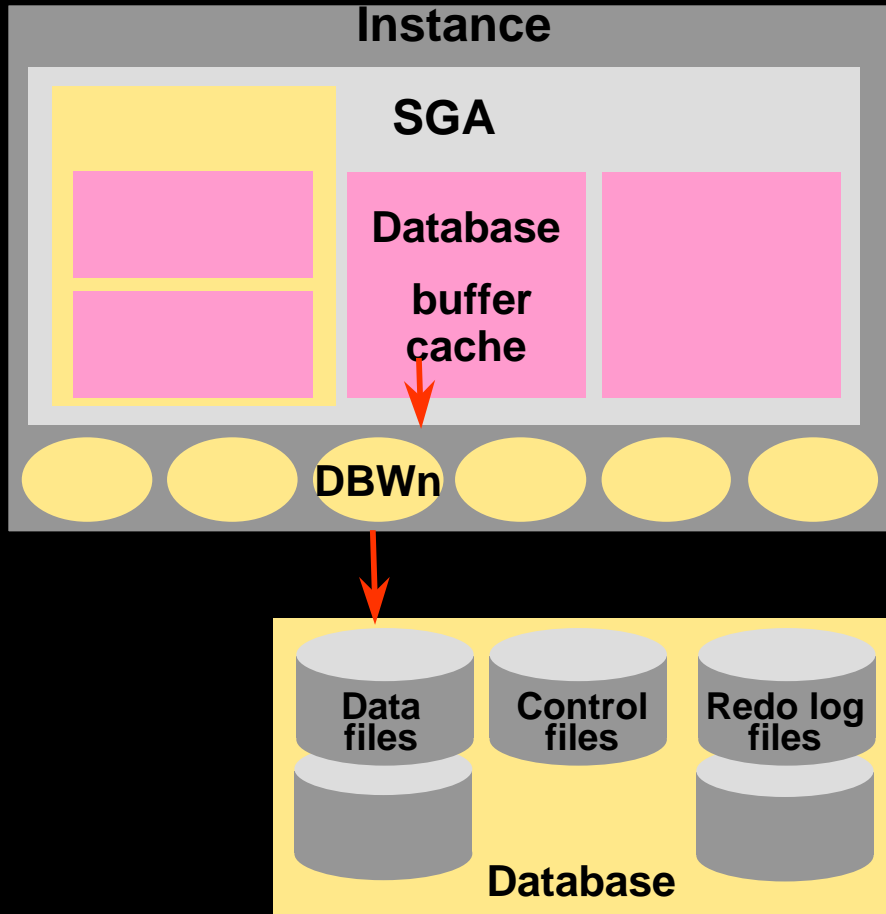
- **Mandatory background processes**

|      |      |      |
|------|------|------|
| DBWn | PMON | CKPT |
| LGWR | SMON | RECO |

- **Optional background processes**

|      |      |      |
|------|------|------|
| ARCn | LMON | Snnn |
| QMnN | LMDn |      |
| CJQ0 | Pnnn |      |
| LCKn | Dnnn |      |

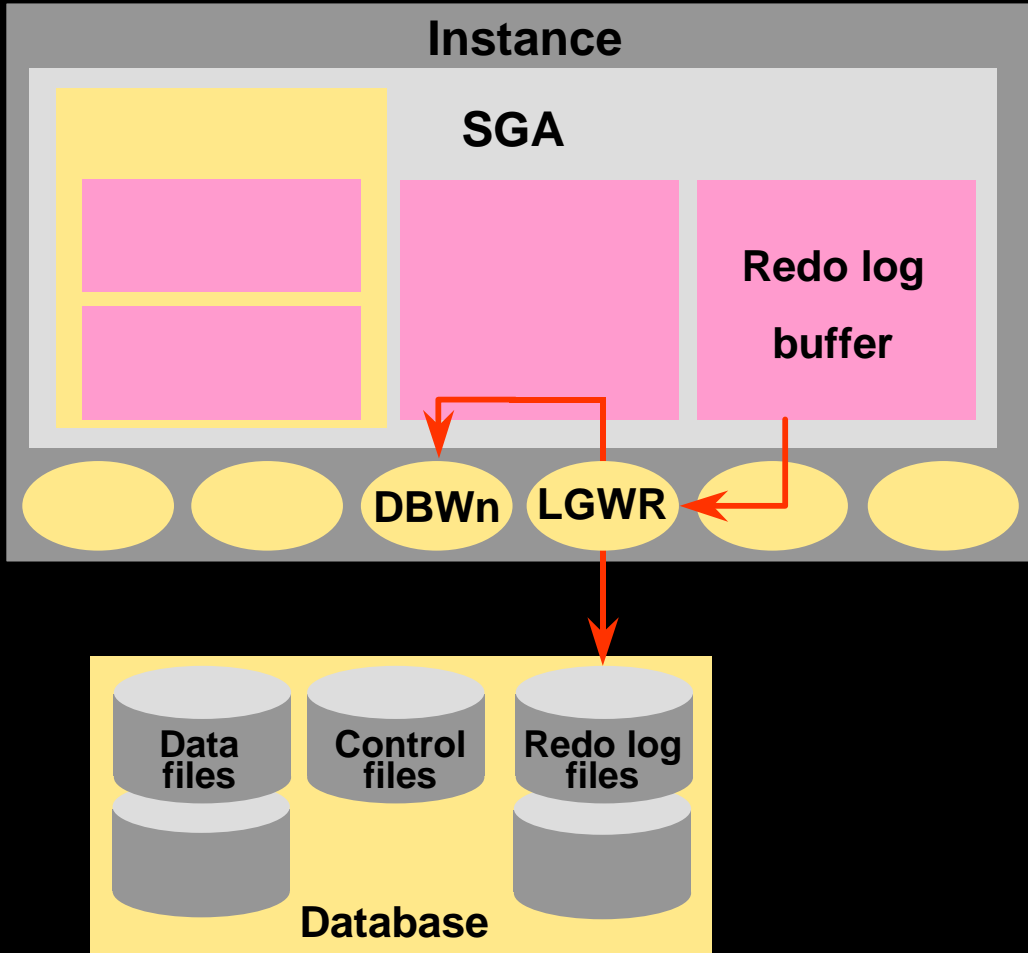
# Database Writer (DBWn)



## DBWn writes when:

- Checkpoint
- Dirty buffers threshold reached
- No free buffers
- Timeout
- RAC ping request
- Tablespace offline
- Tablespace read only
- Table DROP or TRUNCATE
- Tablespace BEGIN BACKUP

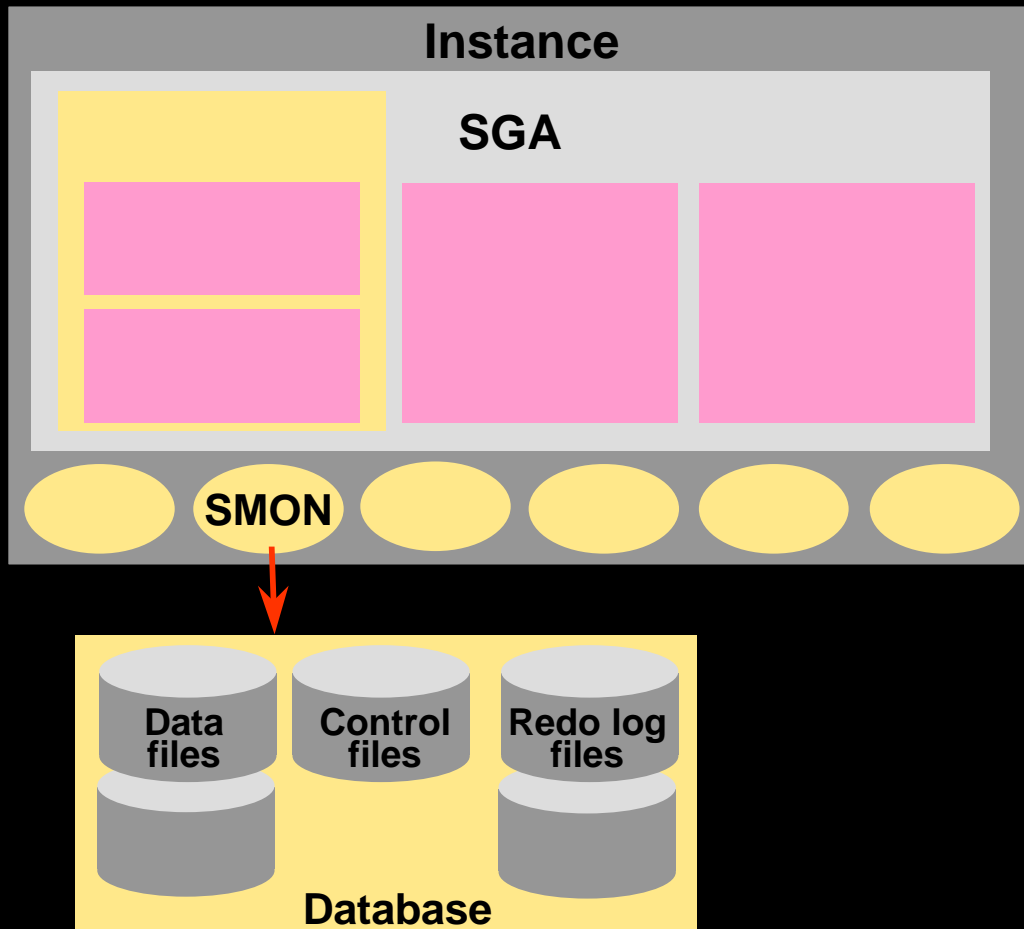
# Log Writer (LGWR)



## LGWR writes:

- At commit
- When one-third full
- When there is 1 MB of redo
- Every 3 seconds
- Before DBWn writes

# System Monitor (SMON)

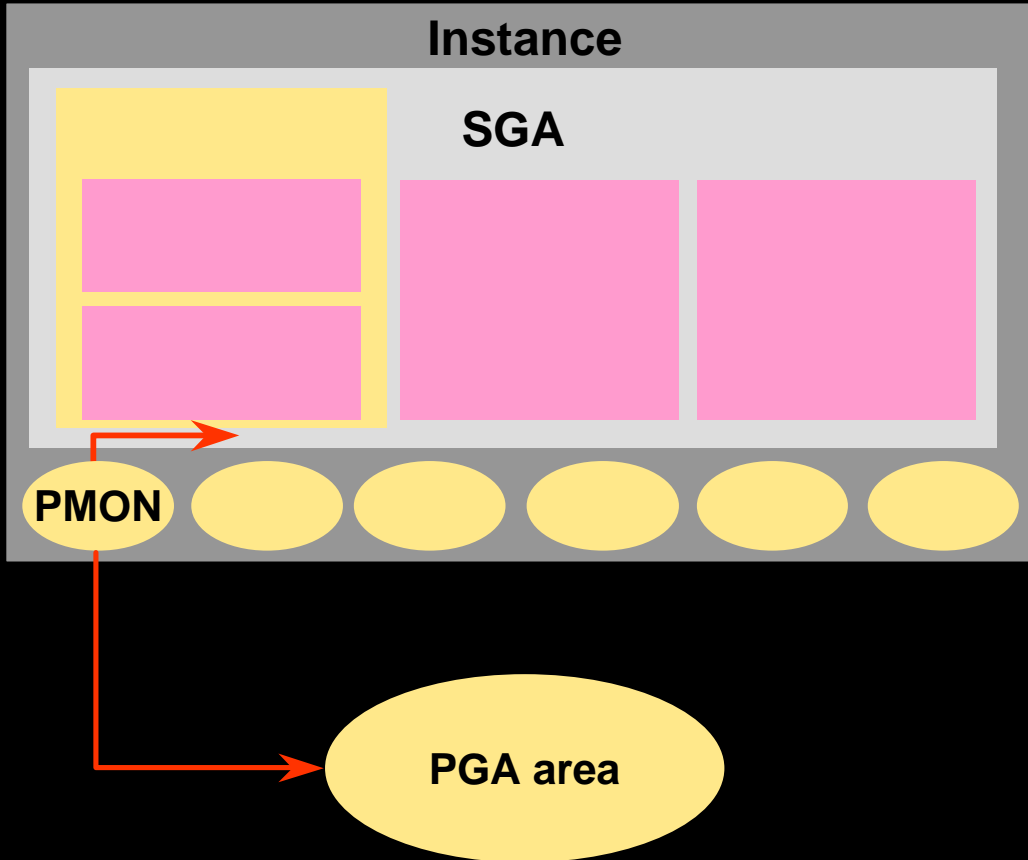


## Responsibilities:

- **Instance recovery:**
  - Rolls forward changes in the redo logs
  - Opens the database for user access
  - Rolls back uncommitted transactions
- **Coalesces free space** ever 3 sec
- **Deallocates temporary segments**



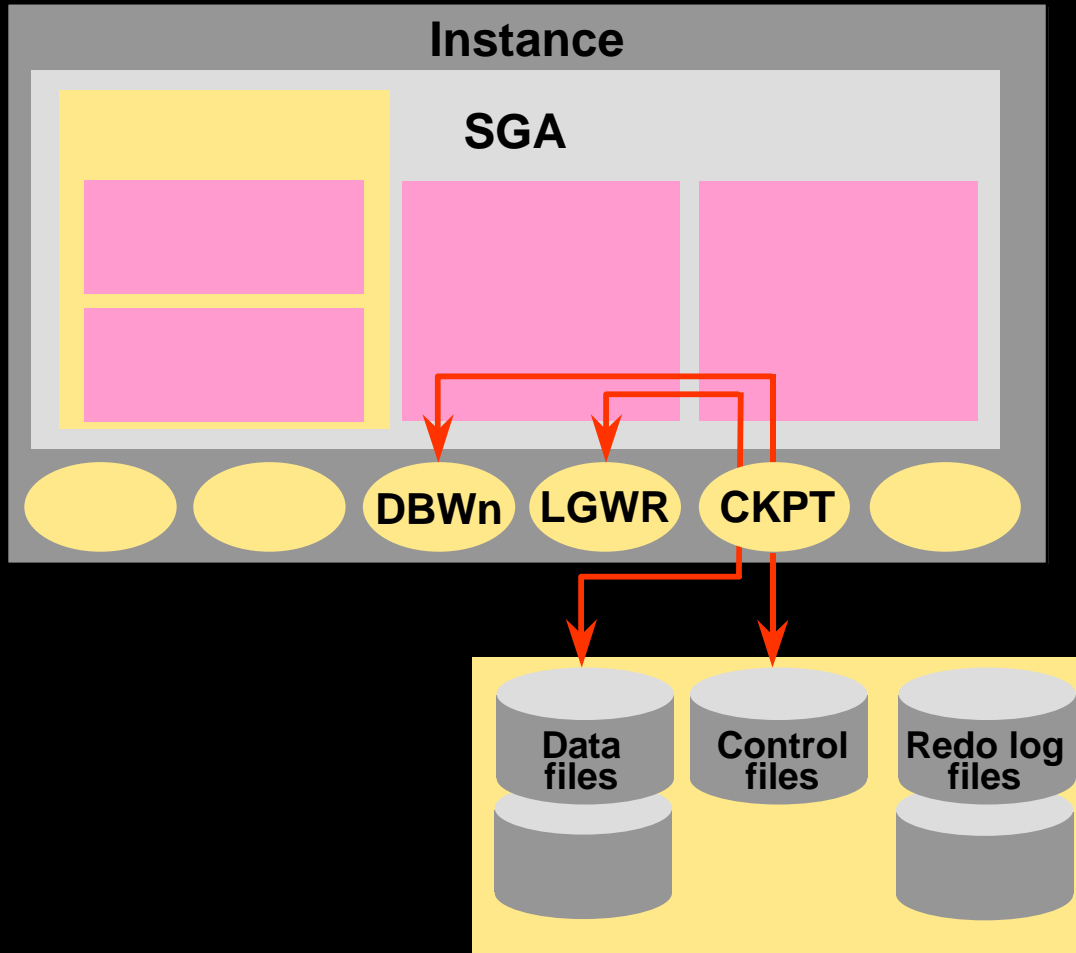
# Process Monitor (PMON)



**Cleans up after failed processes by:**

- Rolling back the transaction
- Releasing locks
- Releasing other resources
- Restarts dead dispatchers

# Checkpoint (CKPT)

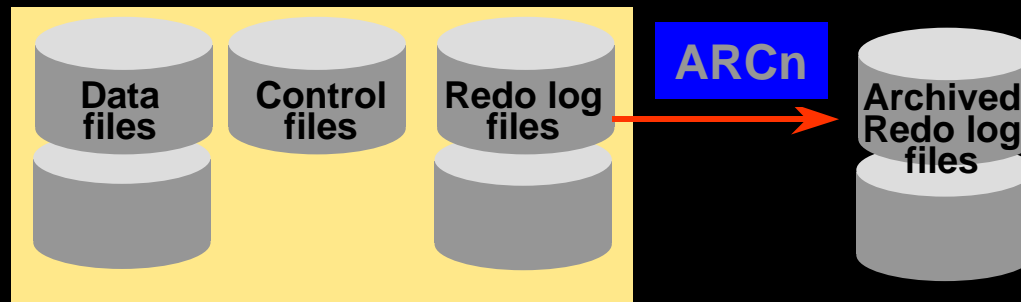


**Responsible for:**

- **Signalling DBWn at checkpoints**
- **Updating datafile headers with checkpoint information**
- **Updating control files with checkpoint information**

# Archiver (ARCn)

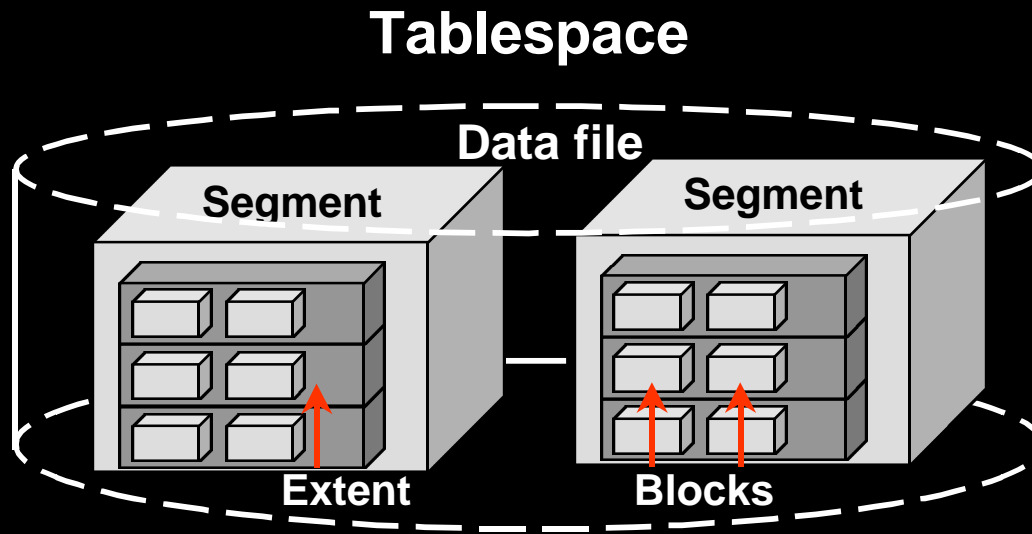
- Optional background process
- Automatically archives online redo logs when `ARCHIVELOG` mode is set
- Preserves the record of all changes made to the database



# Logical Structure

The logical structure of the Oracle architecture dictates how the physical space of a database is to be used.

A hierarchy exists in this structure that consists of tablespaces, segments, extents, and blocks.



# Processing a SQL Statement

- **Connect to an instance using:**
  - The user process
  - The server process
- **The Oracle server components that are used depend on the type of SQL statement:**
  - Queries return rows.
  - DML statements log changes.
  - Commit ensures transaction recovery.
- **Some Oracle server components do not participate in SQL statement processing.**

# Summary

**In this lesson, you should have learned how to:**

- **Explain database files: data files, control files, online redo logs**
- **Explain SGA memory structures: Database buffer cache, shared SQL pool, and redo log buffer**
- **Explain primary background processes: DBWn, LGWR, CKPT, PMON, SMON, and ARCn**

# Practice 1 Overview

**This practice covers the following topics:**

- **Review of architectural components**
- **Structures involved in connecting a user to an Oracle instance.**

# 2

## Getting Started With the Oracle Server



# Objectives

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

- **Identify common database administration tools available to a DBA**
- **Identify the features of the Oracle Universal Installer**
- **Understand the benefits of Optimal Flexible Architecture**
- **Set up password file authentication**
- **List the main components of Oracle Enterprise Manager and their uses**

# Database Administration Tools

| Tool                                    | Description   |
|---|---|
| Oracle Universal Installer (OUI)        | Used to install, upgrade, or remove software components   |
| Oracle Database Configuration Assistant | A graphical user interface tool that interacts with the OUI, or can be used independently, to create, delete or modify a database |
| Password File Utility                   | Utility for creating a database password file   |
| SQL*Plus                                | A utility to access data in an Oracle database  |
| Oracle Enterprise Manager               | A graphical interface used to administer, monitor, and tune one or more databases   |

# Oracle Universal Installer

- **Used to install, upgrade, or remove software components, and create a database**
- **Based on a Java engine**
- **Features include:**
  - **Automatic dependency resolution**
  - **Allows for Web-based installations**
  - **Tracking inventory of component and suite installations**
  - **Deinstallation of installed components**
  - **Support for multiple Oracle homes**
  - **Support for globalization technology**

# Interactive Installation

**To start Oracle Universal Installer on Unix:**

```
$ ./runInstaller
```

# Non-Interactive Installation Using Response Files

**To start Universal Installer in non-interactive mode:**

```
./runInstaller -responsefile myrespfile -silent
```

# Oracle Database Configuration Assistant

**The Oracle Database Configuration Assistant allows you to:**

- **Create a database**
- **Configure database options**
- **Delete a database**
- **Manage templates**

# Optimal Flexible Architecture (OFA)

- Oracle's recommended standard database architecture layout
- OFA involves three major rules
  - Establish a directory structure where any database file can be stored on any disk resource
  - Separate objects with different behavior into different tablespaces
  - Maximize database reliability and performance by separating database components across different disk resources

# Oracle Software and File Locations

| Software                     | Files                      |
|------------------------------|----------------------------|
| <code>oracle_base</code>     | <code>oradata/</code>      |
| <code>/product</code>        | <code>db01/</code>         |
| <code>/release_number</code> | <code>system01.dbf</code>  |
| <code>/bin</code>            | <code>control01.ctl</code> |
| <code>/dbs</code>            | <code>redo0101.log</code>  |
| <code>/rdbms</code>          | <code>...</code>           |
| <code>/sqlplus</code>        | <code>db02/</code>         |
| <code>/admin</code>          | <code>system01.dbf</code>  |
| <code>/inst_name</code>      | <code>control01.ctl</code> |
| <code>/pfile</code>          | <code>redo0101.log</code>  |
|                              | <code>...</code>           |



# Database Administrator Users

When a database is created the users `SYS` and `SYSTEM` are created automatically and granted the `DBA` role.

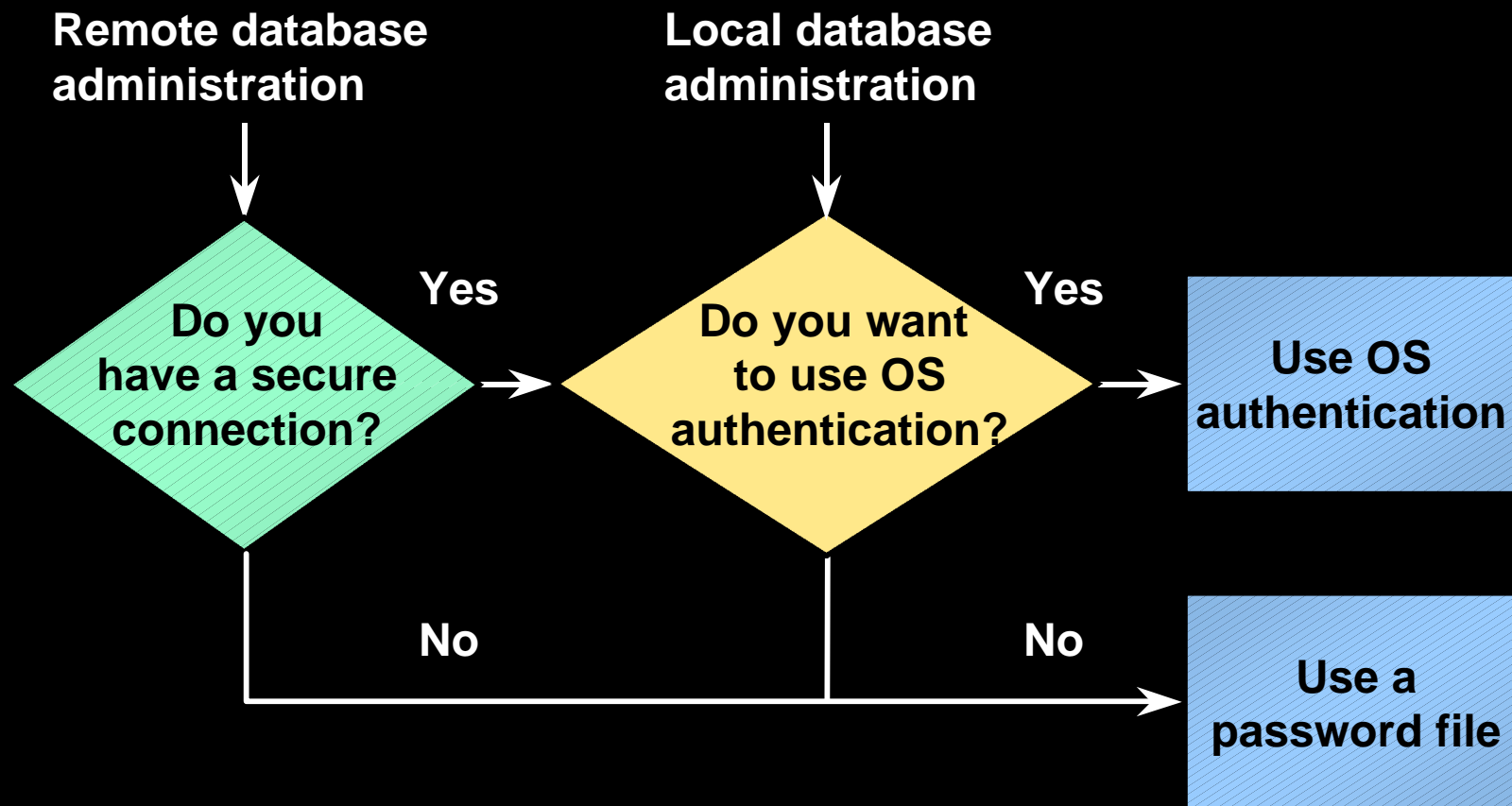
## `SYS`

- Password:  
`change_on_install`
- Owner of the database data dictionary

## `SYSTEM`

- Password:  
`manager`
- Owner of additional internal tables and views used by Oracle tools

# Authentication Methods for Database Administrators



# Using Password File Authentication

- **Create the password file using the password utility:**

```
$orapwd file=$ORACLE_HOME/dbs/orapwU15  
password=admin entries=5
```

- **Set REMOTE\_LOGIN\_PASSWORDFILE to EXCLUSIVE in the initialization parameter file**
- **Add users to the password file and assign appropriate privileges to each user**

```
GRANT SYSDBA TO HR;
```

# SQL\*Plus

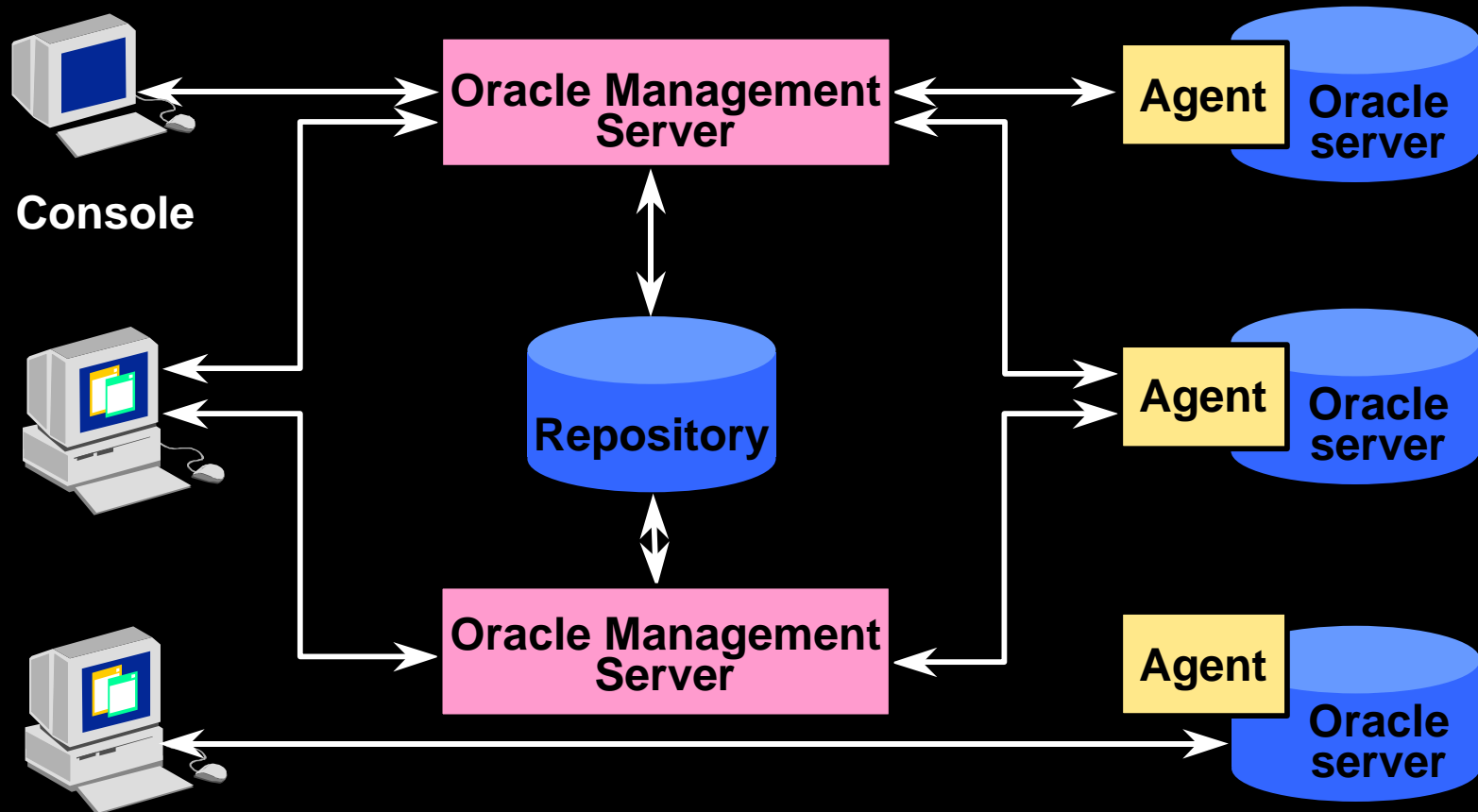
**SQL\*Plus is an Oracle tool that provides the capability to interact with and manipulate the database.**

- **Provides the ability to start up and shutdown the database, create and run queries, add rows, modify data, and write customized reports**
- **Subset of the standard SQL language with specific add ons**

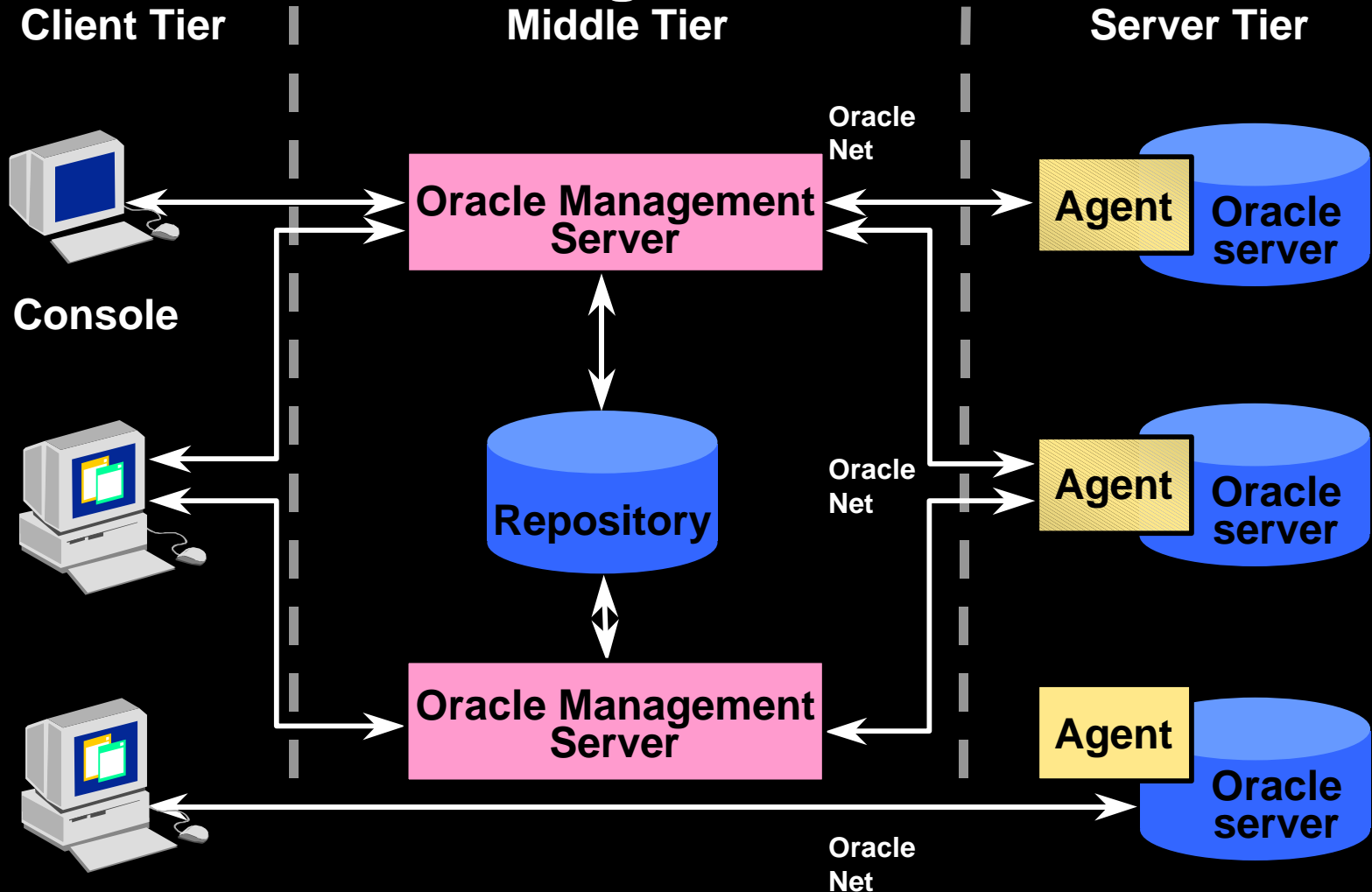
# Oracle Enterprise Manager

- **Serves as a centralized systems management console for DBAs**
- **Provides a set of common services that help manage the Oracle environment**
- **Includes integrated applications**
- **Manages the environment such as databases, web servers and listeners**
- **A tool to administer, diagnose and tune multiple databases**
- **Provides tools for administering parallel servers and replicated databases**

# Oracle Enterprise Manager Architecture: Java-Based Console and Applications



# Oracle Enterprise Manager Architecture: Managed Nodes



ORACLE®

# Oracle Enterprise Manager Console

## Console

- **Central launching point for all applications**
- **Can be run in thin mode or as a fat client**
- **Can be launched either standalone or through Oracle Management Server**



# DBA Tools

**Standard applications that can be launched from the Console:**

- **Instance Manager**
- **Security Manager**
- **Storage Manager**
- **Schema Manager**
- **SQL\*Plus Worksheet**

# Summary

**In this lesson, you should have learned to:**

- **Identify the tools available to a DBA**
- **Identify the features of the Oracle Universal Installer**
- **Outline the recommended layout defined by the Optimal Flexible Architecture**
- **Setup operating system and password file authentication**
- **Identify the main components of Oracle Enterprise Manager**

# Practice 2 Overview

**This practice covers the following topics:**

- **Executing SQL\*Plus scripts**
- **Creating password file**

# Managing an Oracle Instance

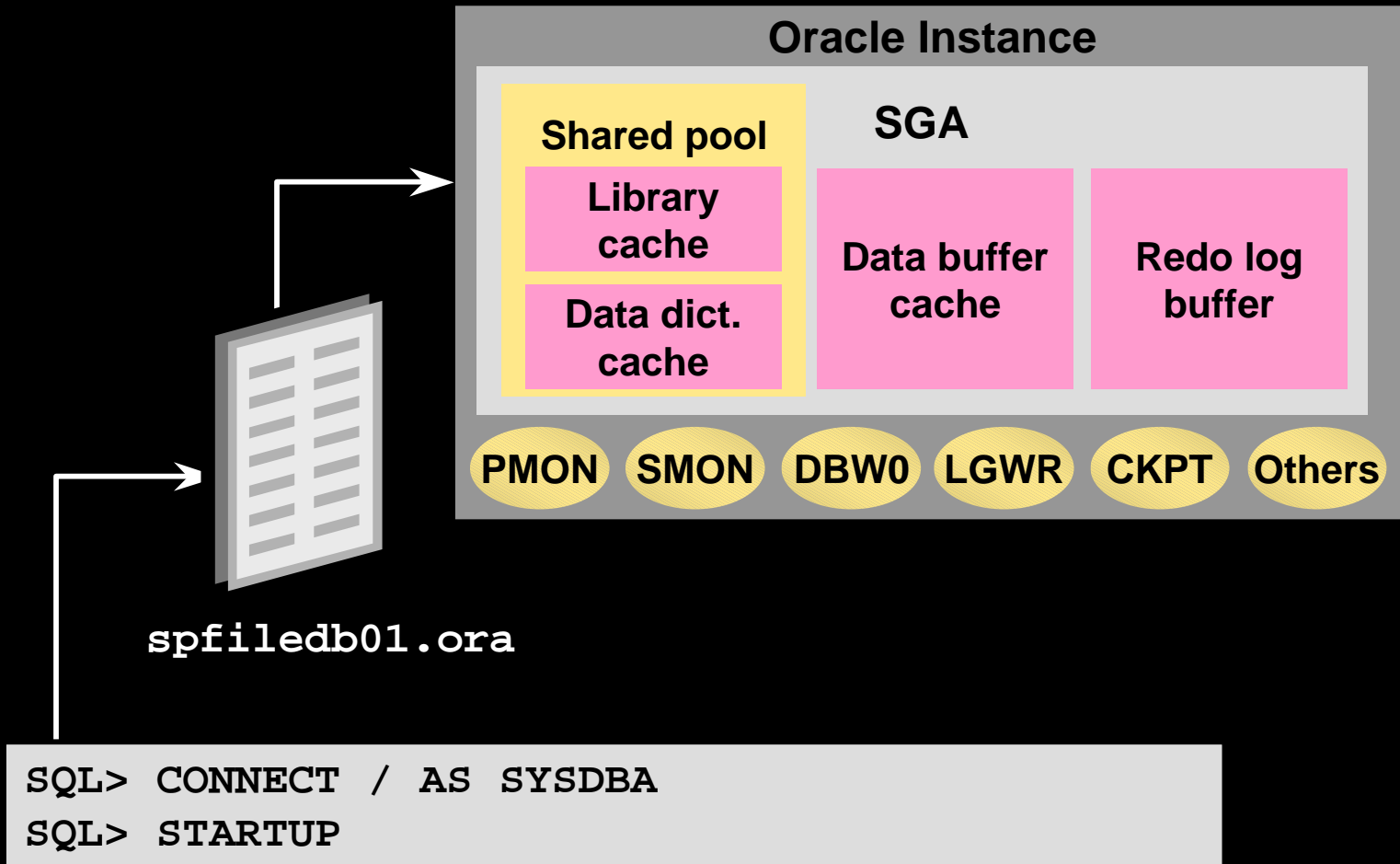


# Objectives

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

- **Create and manage initialization parameter files**
- **Configure OMF**
- **Start up and shut down an instance**
- **Monitor and use diagnostic files**

# Initialization Parameter Files



# Initialization Parameter Files

- Entries are specific to the instance being accessed
- There are two kinds of parameters:
  - Explicit: Having an entry in the file
  - Implicit: No entry within the file, but assuming the Oracle default values
- Multiple files can be used for a single database to optimize performance in different situations.
- Changes to entries in the file take effect based on the type of initialization parameter file used;
  - Static parameter file, `PFILE`
  - Persistent parameter file, `SPFILE`

# **PFILE**

## **initSID.ora**

- **The PFILE is a text file that can be modified with an operating system editor.**
- **Modifications to the file are made manually.**
- **Changes to the file take effect on the next startup.**
- **Its default location is \$ORACLE\_HOME/dbs.**



# PFILE Example

# Initialization Parameter File: initdb01.ora

```
db_name           = db01
instance_name     = db01
control_files     = ( /u03/oradata/db01/control01db01.ctl,
                     /u03/oradata/db01/control02db01.ctl )
db_block_size     = 4096
db_block_buffers  = 500
shared_pool_size  = 31457280 # 30M Shared Pool
db_files          = 1024
max_dump_file_size = 10240
background_dump_dest = /u05/oracle9i/admin/db01/bdump
user_dump_dest     = /u05/oracle9i/admin/db01/udump
core_dump_dest     = /u05/oracle9i/admin/db01/cdump
undo_management   = auto
undo_tablespace   = undtbs
. . .
```

# SPFILE

## spfileSID.ora

- Binary file with the ability to make changes persistent across shutdown and startup
- Maintained by the Oracle server
- Records parameter value changes made with the `ALTER SYSTEM` command
- Can specify whether the change being made is temporary or persistent
- Values can be deleted or reset to allow an instance to revert to the default value

```
ALTER SYSTEM SET undo_tablespace = 'UNDO2';
```

# Creating an SPFILE

**SPFILE can be created from an `initSID.ora` file using the `CREATE SPFILE` command, which can be executed before or after instance startup:**

```
CREATE SPFILE FROM PFILE;
```

# SPFILE Example

```
*.background_dump_dest='$ORACLE_HOME/admin/db01/bdump'
*.compatible='9.0.0'
*.control_files='/u03/oradata/db01/ctrl01db01.ctl', '/u03/orad
ata/db01/ctrl02db01.ctl'
*.core_dump_dest='$ORACLE_HOME/admin/db01/cdump'
*.db_block_buffers=500
*.db_block_size=4096
*.db_files=40
*.db_name='db01'
*.instance_name='db01'
*.remote_login_passwordfile='exclusive'
*.shared_pool_size=31457280 # 30M Shared Pool
*.undo_management='AUTO'
db01.undo_tablespace='UNDOTBS01'
db02.undo_tablespace='UNDOTBS02'
. . .
```

# Oracle Managed Files

**Oracle Managed Files (OMF) simplify file administration**

- **OMF are created and deleted by the Oracle server as directed by SQL commands**
- **OMF are established by setting two parameters:**
  - **DB\_CREATE\_FILE\_DEST: Set to give the default location for data files**
  - **DB\_CREATE\_ONLINE\_LOG\_DEST\_N: Set to give the default locations for online redo logs and control files, up to a maximum of 5 locations**

# Oracle Managed File Example

To create a database where data files, control files, and online redo log files are created in separate directories:

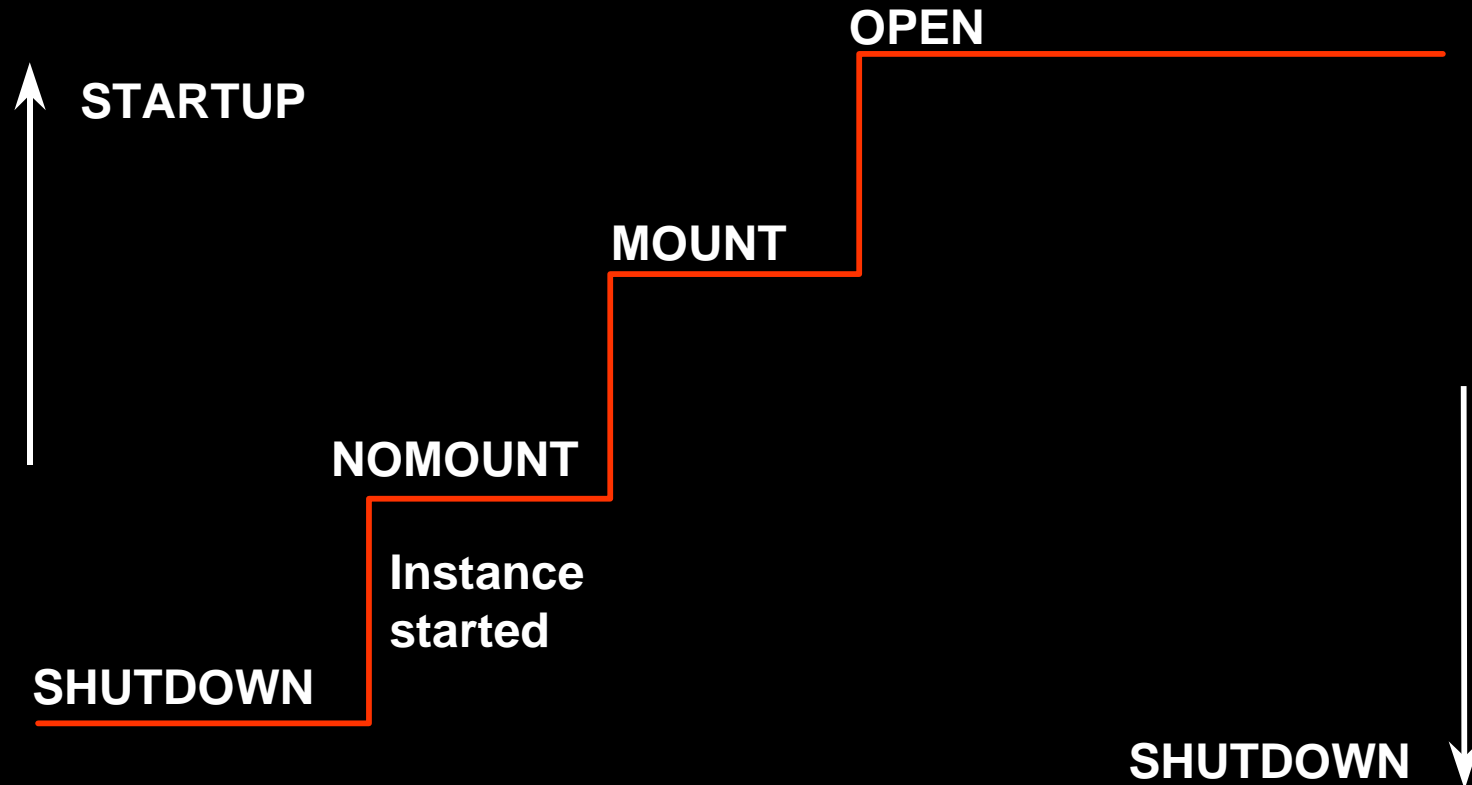
- Set the initialization parameters:

```
DB_CREATE_FILE_DEST = '/u01/oradata/'  
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata/'  
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata/'
```

- Issue the **CREATE DATABASE SQL** statement.

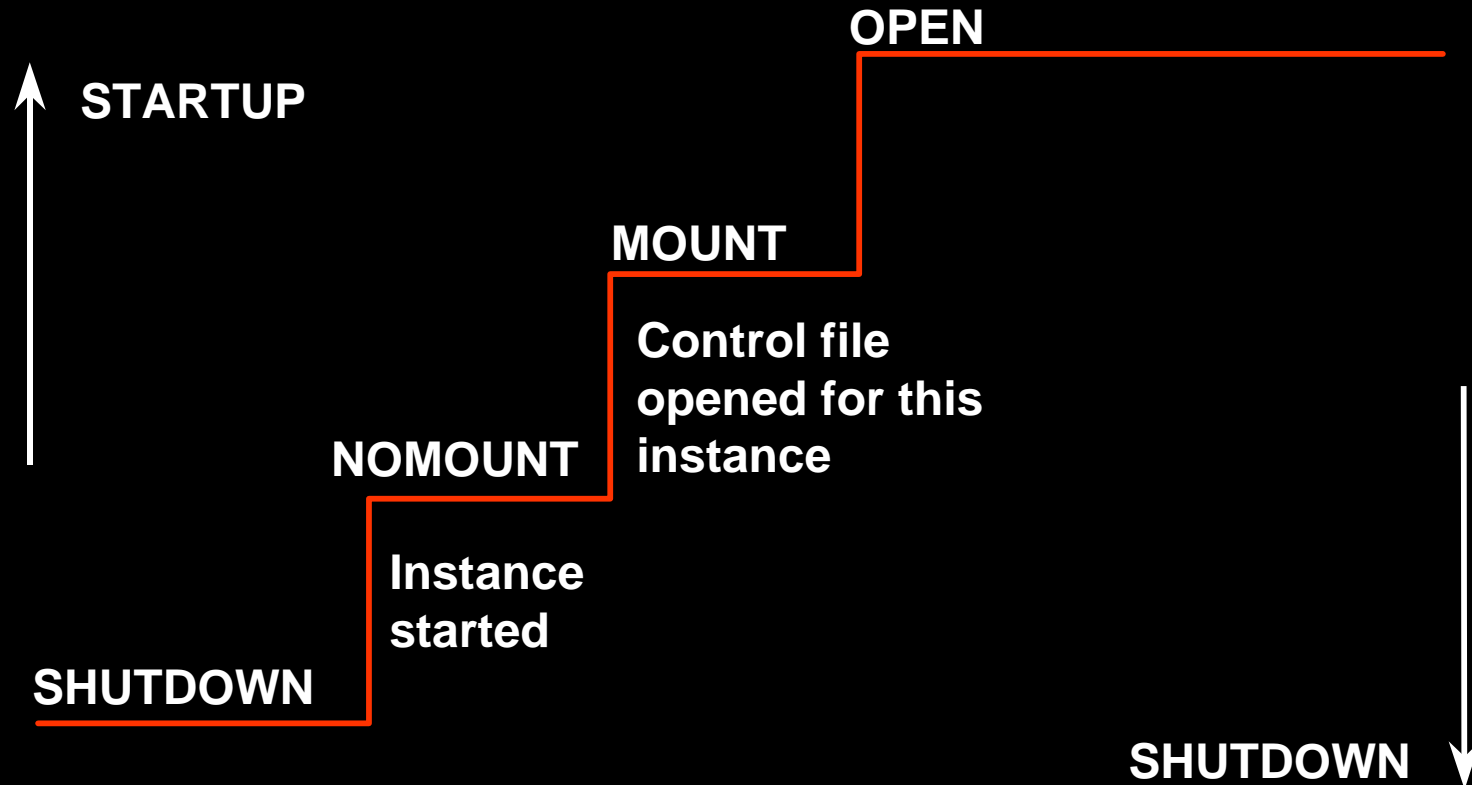
# Starting Up a Database

## NOMOUNT



# Starting Up a Database

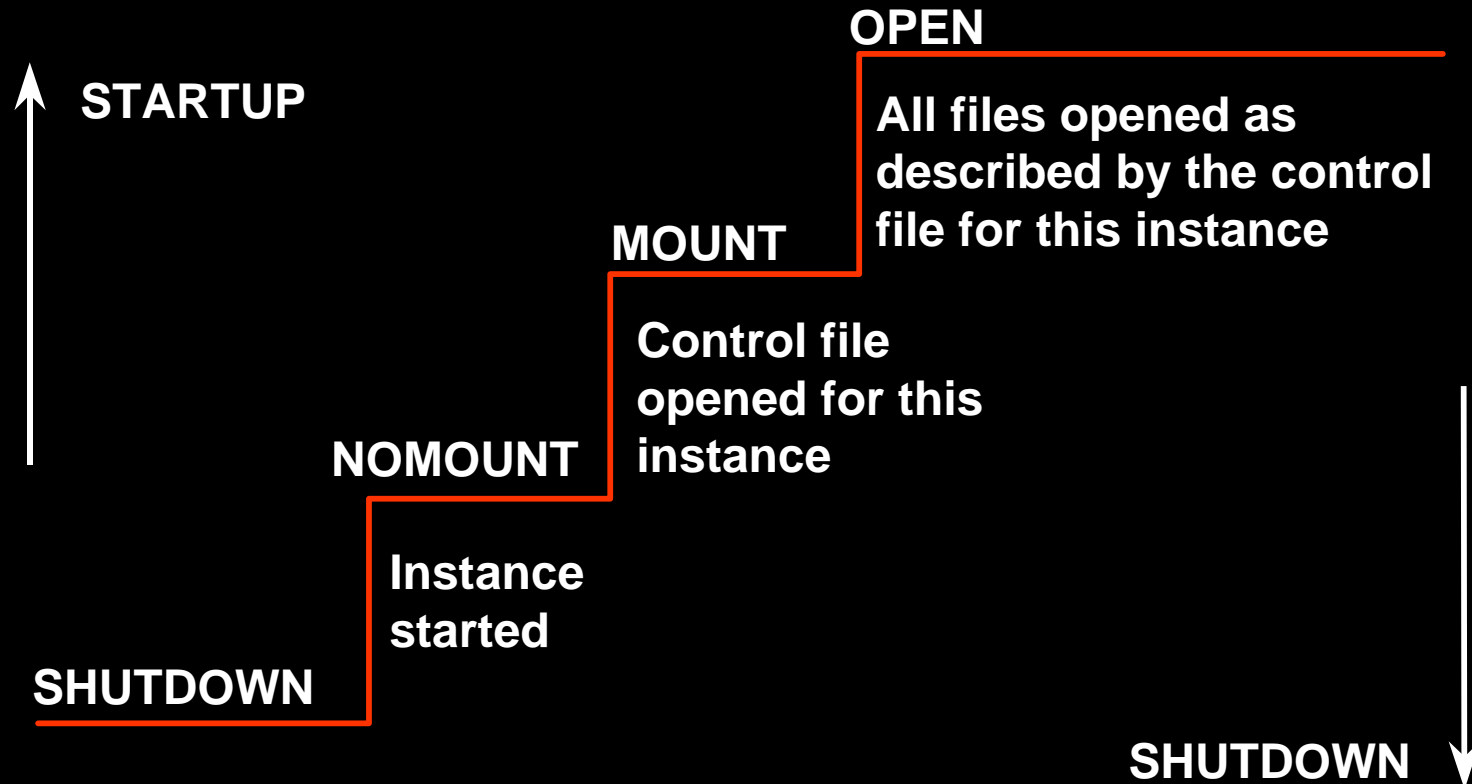
## MOUNT





# Starting Up a Database

## OPEN



# STARTUP Command

**Start up the instance and open the database:**

```
STARTUP
```

```
STARTUP PFILE=$ORACLE_HOME/dbs/initdb01.ora
```

# The ALTER DATABASE Command

- Change the state of the database from NOMOUNT to MOUNT:

```
ALTER DATABASE db01 MOUNT;
```

- Open the database as a read-only database:

```
ALTER DATABASE db01 OPEN READ ONLY;
```

# Opening a Database in Restricted Mode

- Use the **STARTUP** command to restrict access to a database:

```
STARTUP RESTRICT
```

- Use the **ALTER SYSTEM** command to place an instance in restricted mode:

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;
```

# Opening a Database in Read-Only Mode

- **A databases can be opened as a read-only database.**
- **A read-only database can be used to:**
  - **Execute queries**
  - **Execute disk sorts using locally managed tablespaces**
  - **Take data files offline and online, not tablespaces**
  - **Perform recovery of offline data files and tablespaces**

# Shutting Down the Database

| Shutdown Mode                       | A | I | T | N |
|-------------------------------------|---|---|---|---|
| Allow new connections               | x | x | x | x |
| Wait until current sessions end     | x | x | x | o |
| Wait until current transactions end | x | x | o | o |
| Force a checkpoint and close files  | x | o | o | o |

## Shutdown Mode:

- NORMAL
- TRANSACTIONAL
- IMMEDIATE
- ABORT

|   |     |
|---|-----|
| x | NO  |
| o | YES |

# Shutdown Options

## On the way down:

- Database buffer cache written to the data files
- Uncommitted changes rolled back
- Resources released.

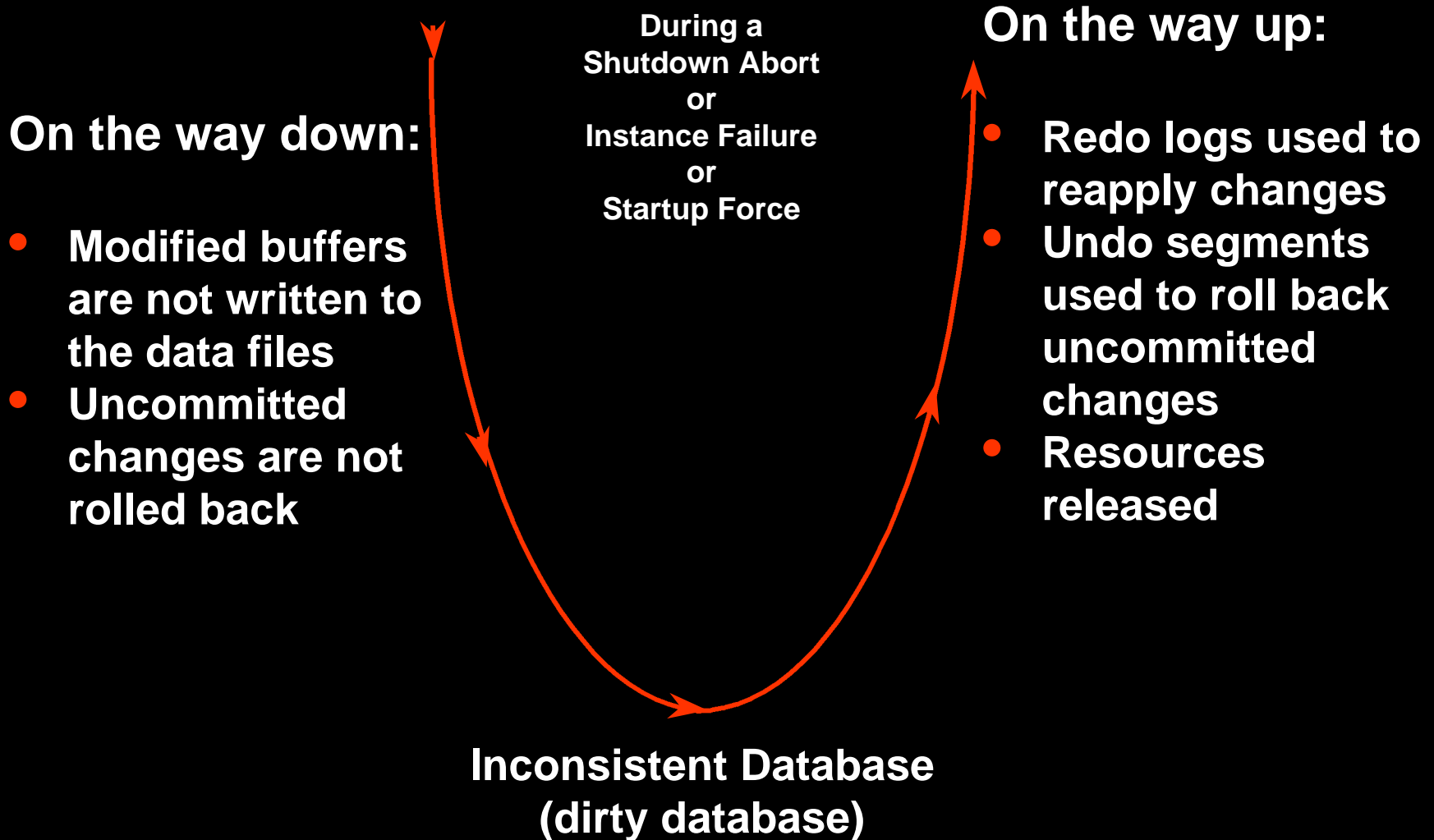
During a  
Shutdown Normal,  
Shutdown Transactional  
or  
Shutdown Immediate

## On the way up:

- No instance recovery

Consistent Database  
(clean database)

# Shutdown Options





# Managing an Instance by Monitoring Diagnostic Files

**Diagnostic files contain information about significant events encountered while the instance is operational.**

- **Used to resolve problems or to better manage the database on a day-to-day basis.**
- **Several types of diagnostic files exist:**
  - **alertSID.log file**
  - **Background trace files**
  - **User trace files**

# Alert Log File

The `alertSID.log` file records the commands and Results of major events while the database is operational.

- It is used for day-to-day operational information or diagnosing database errors.
- Each entry has a time stamp associated with it.
- The DBA manages the `alertSID.log` file.
- Its location is defined by `BACKGROUND_DUMP_DEST`.

# Background Trace Files

**Background trace files support information errors detected by any background process.**

- **They are used to diagnose and troubleshoot errors.**
- **They are created when a background process encounters an errors**
- **Their location is defined by  
BACKGROUND\_DUMP\_DEST**

# User Trace File

**A user trace file is produced by the user process connected to the Oracle server through the server process.**

- **A user trace file contains statistics for traced SQL statements or user error messages.**
- **It is created when a user encounters user session errors.**
- **It can also be generated by a you or a server process.**
- **Its location is defined by `USER_DUMP_DEST`.**
- **Its size is defined by `MAX_DUMP_FILE_SIZE` and defaults to 10M.**

# Enabling or Disabling User Tracing

User tracing is enabled or disabled at the session or instance level by using the following commands and parameter:

- **Session level using the ALTER SESSION command:**  
`ALTER SESSION SET SQL_TRACE = TRUE`
- **Session level by executing DBMS procedure:**  
`dbms_system.SET_SQL_TRACE_IN_SESSION`
- **Instance level by setting the initialization parameter:**  
`SQL_TRACE = TRUE`

# Summary

**In this lesson, you should have learned how to:**

- **Create and manage initialization parameter files**
- **Create and manage OMF files**
- **Start up and shut down an instance**
- **Monitor and use diagnostic files**

# Practice 3 Overview

**This practice covers the following topics:**

- **Creating an SPFILE**
- **Starting up and shutting down the database in different modes**

# 4

## Creating a Database



# Objectives

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

- **Understand the prerequisites necessary for database creation**
- **Create a database using Oracle Database Configuration Assistant**
- **Create a database manually**

# Managing and Organizing a Database

- **Creating a database is the first step in managing a database system**
- **A database may have been created automatically as part of Oracle9i Server installation, or you can create a new one later**
- **Oracle Data Migration Assistant is used to migrate from an earlier version of the database**

# Creation Prerequisites

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

- **A privileged account authenticated in one of the following ways:**
  - **By the operating system**
  - **Using a password file**
- **Sufficient memory to start the instance**
- **Sufficient disk space for the planned database**

# Planning Database File Locations

- **Keep at least two active copies of a database control file on at least two different devices.**
- **Multiplex the redo log files and put group members on different disks.**
- **Separate data files whose data:**
  - **Will participate in disk resource contention across different physical disk resources**
  - **Have different life spans**
  - **Have different administrative characteristics**

# Creating a Database

**An Oracle database can be created using:**

- **Oracle Database Configuration Assistant**
- **The `CREATE DATABASE` command**

# Operating System Environment

**On Unix, set the following environment variables:**

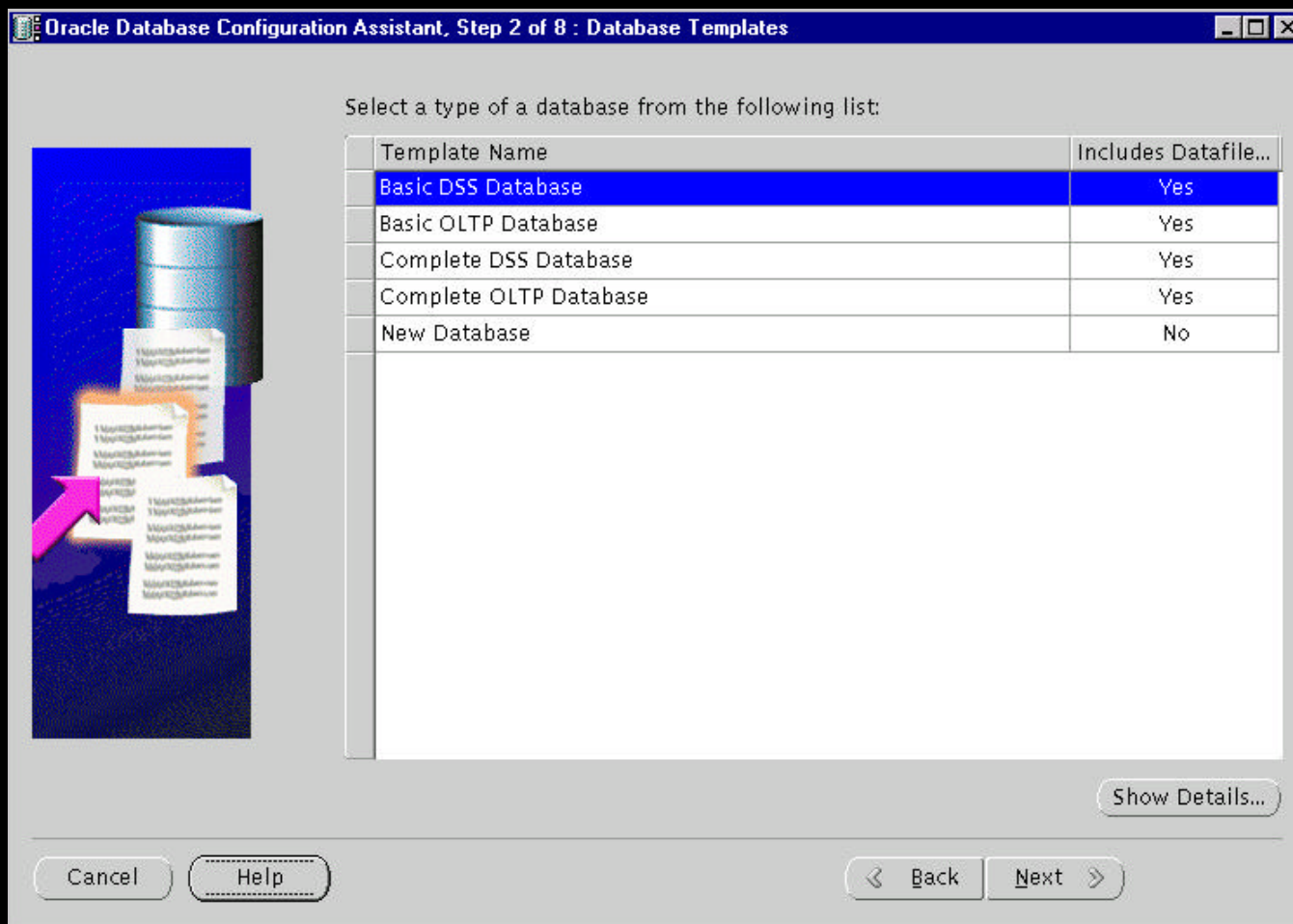
- **ORACLE\_BASE**
- **ORACLE\_HOME**
- **ORACLE\_SID**
- **ORA\_NLS33**
- **PATH**
- **LD\_LIBRARY\_PATH**

# Using the Database Configuration Assistant

**The Database Configuration Assistant allows you to:**

- **Create a database**
- **Configure database options**
- **Delete a database**
- **Manage templates**
  - **Create new template using pre-defined template settings**
  - **Create new template from an existing database**
  - **Delete database template**

# Create a Database





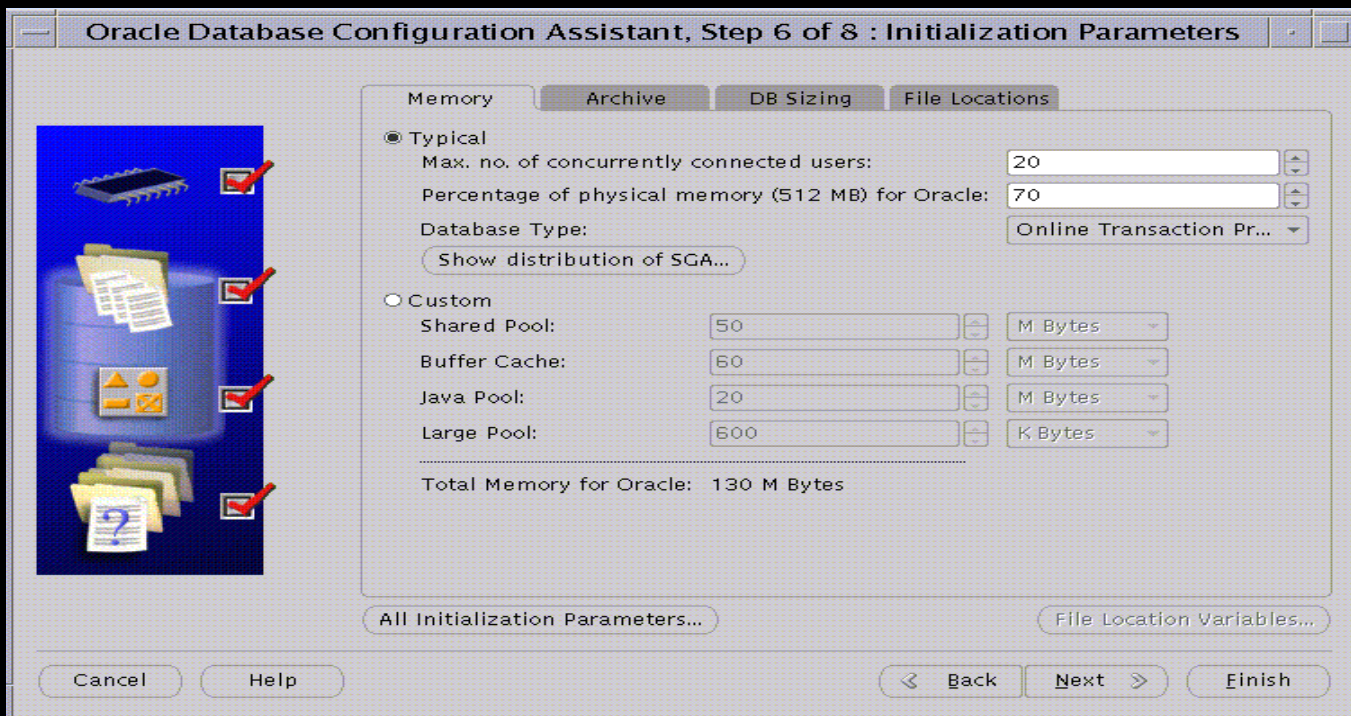
# Database Information

## Specify:

- **Global database name and SID**
- **The features you want to use for your database, such as:**
  - **Oracle Spatial**
  - **Oracle OLAP Services**
  - **Example Schemas**
- **Mode in which you want the database to operate**
  - **Dedicated server mode**
  - **Shared server mode**

# Typical or Custom Install

Choose between typical or custom install



# Other Parameters

- **Archive Parameters**
  - Use for database recovery
  - May also be used for a standby database
- **Data Block Sizing**
  - Sets the default database block size
  - Helps to determine the `SORT_AREA_SIZE`
- **File Locations**
  - Specify paths for trace files
  - Specify paths for parameter files
- **Database storage**
  - Specify storage parameters

# Complete Database Creation

**Complete database creation using the following options:**

- **Create database**
- **Save as a database template**
- **Generate database creation scripts**

# Creating a Database Manually

- **Decide on a unique instance and database name**
- **Choose a database character set**
- **Set the operating system variables**
- **Edit / Create the initialization parameter file**
- **Start the instance (`nomount`)**
- **Execute the `CREATE DATABASE` command**
- **Run scripts to generate the data dictionary and accomplish post creation steps**

# Preparing the Parameter File

- Create the new `initSID.ora`

```
$ cp init.ora $ORACLE_HOME/dbs/initdb01.ora
```

- Modify the `initSID.ora` by editing the parameters

# Creating SPFILE

**Create the SPFILE from initSID.ora**

```
CREATE SPFILE FROM PFILE;
```

# Starting the Instance

- **Connect as SYSDBA**
- **Start the instance in NOMOUNT stage**

```
STARTUP NOMOUNT
```



# Creating the Database

```
@crdbdb01.sql
```

```
SQL> create database db01
```

```
2 logfile
```

```
3 GROUP 1 ('/u01/oradata/db01/log_01_db01.rdo') SIZE 15M,
```

```
4 GROUP 2 ('/u01/oradata/db01/log_02_db01.rdo') SIZE 15M,
```

```
5 GROUP 3 ('/u01/oradata/db01/log_03_db01.rdo') SIZE 15M
```

```
6 datafile '/u01/oradata/db01/system_01_db01.dbf' SIZE 100M
```

```
7 undo tablespace UNDO
```

```
8 datafile '/u01/oradata/db01/undo_01_db01.dbf' SIZE 40M
```

```
9 default temporary tablespace TEMP
```

```
10 tempfile '/u01/oradata/db01/temp_01_db01.dbf' SIZE 20M
```

```
11 extent management local uniform size 128k
```

```
12 character set AL32UTF8
```

```
13 national character set AL16UTF16
```

```
14 set time_zone = 'America/New_York'
```

```
15 ;
```

# Creating a Database Using OMF

- Define the OMF initialization parameters in the parameter file
  - DB\_CREATE\_FILE\_DEST
  - DB\_CREATE\_ONLINE\_DEST\_*n*

```
STARTUP NOMOUNT
CREATE DATABASE
DEFAULT TEMPORARY TABLESPACE TEMP;
```

# Troubleshooting

**Creation of the database fails if:**

- **There are syntax errors in the SQL script**
- **Files that should be created already exist**
- **Operating system errors such as file or directory permission or insufficient space errors occur**

# After Database Creation

**The database contains:**

- **Datafiles, control files, and redo log files**
- **User `sys` with the password `change_on_install`**
- **User `SYSTEM` with the password `manager`**
- **Internal tables (but no data dictionary views)**

# Summary

**In this lesson, you should have learned to:**

- **Identify the prerequisites required to create a database**
- **Create a database using the Oracle Database Configuration Assistant**
- **Create a database manually**

# 5

## Data Dictionary Contents and Usage

# Objectives

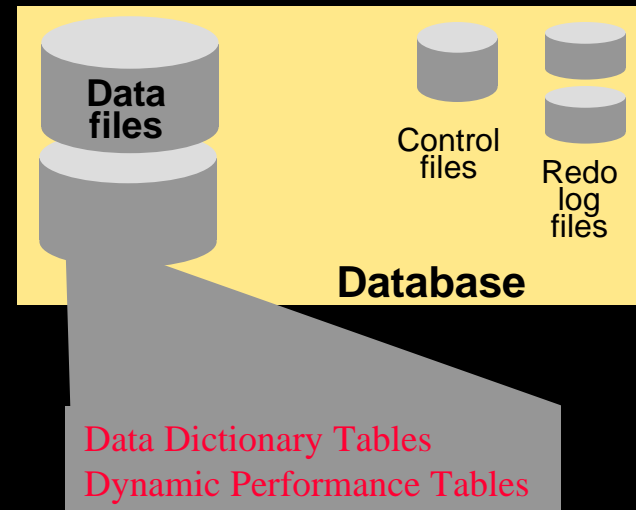
**After completing this lesson, you should be able to do the following:**

- **Identify key data dictionary components**
- **Identify the contents and uses of the data dictionary**
- **Query the data dictionary**

# Data Dictionary

During database creation, the Oracle server creates additional object structures within the data files.

- Data dictionary tables
- Dynamic performance tables





# Data Dictionary

**The data dictionary is a set of read-only tables and views that record, verify, and provide information about its associated database.**

- **Describes the database and its objects**
- **Includes two types of objects:**
  - **Base tables**
    - **Store description of database**
    - **Created with `CREATE DATABASE`**
  - **Data Dictionary views**
    - **Summarize base table information**
    - **Created using `catalog.sql` script**

# Data Dictionary Contents

**The data dictionary provides information about:**

- **Logical and physical database structure**
- **Definitions and space allocations of objects**
- **Integrity constraints**
- **Users**
- **Roles**
- **Privileges**
- **Auditing**

# How the Data Dictionary Is Used

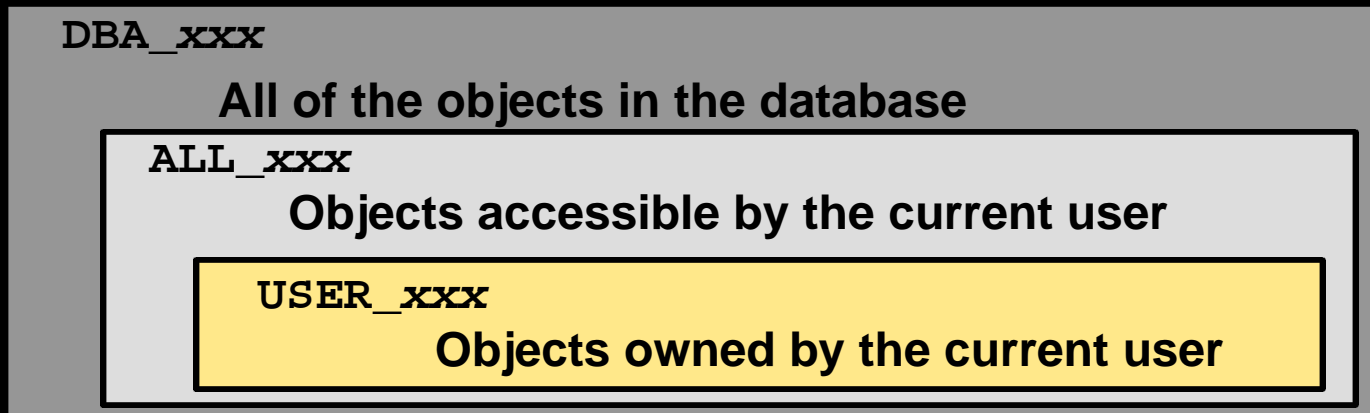
**The data dictionary has three primary uses:**

- **The Oracle server uses it to find information about:**
  - **Users**
  - **Schema objects**
  - **Storage structures**
- **The Oracle server modifies it when a DDL statement is executed.**
- **Users and DBAs can use it as a read-only reference for information about the database.**

# Data Dictionary View Categories

The data dictionary consists of three main sets of static views distinguished from each other by their scope:

- **DBA:** What is in all the schemas
- **ALL:** What the user can access
- **USER:** What is in the user's schema



# Dynamic Performance Tables

**Dynamic performance views record current database activity.**

- **Views are continually updated while the database is operational**
- **Information is accessed from:**
  - **Memory**
  - **Control file**
- **DBA uses dynamic views to monitor and tune the database**
- **Dynamic views are owned by SYS user**
- **DML is not allowed**

# Querying the Data Dictionary and Dynamic Performance Views

Data dictionary and dynamic performance views can be queried for information.

- A listing of views available can be retrieved by querying the `DICTIONARY` view.
- A listing of the columns and its contents can be accessed using `DESCRIBE` and `SELECT`.
- Column comments are available to retrieve more insight into what a column content means within a particular view.

# Data Dictionary Examples

- **General Overview**
  - `DICTIONARY`, `DICT_COLUMNS`
- **Schema objects**
  - `DBA_TABLES`, `DBA_INDEXES`, `DBA_TAB_COLUMNS`,  
`DBA_CONSTRAINTS`
- **Space allocation**
  - `DBA_SEGMENTS`, `DBA_EXTENTS`
- **Database structure**
  - `DBA_TABLESPACES`, `DBA_DATA_FILES`

# Summary

**In this lesson, you should have learned how to:**

- **Use the data dictionary views to get information about the database and instance**
- **Obtain information about data dictionary views from `DICTIONARY` and `DICT_COLUMNS`**



# Practice 5 Overview

**This practice covers the following topics:**

- **Identify the components and contents of the data dictionary**
- **Query the data dictionary**

# 6

## Maintaining the Control File

# Objectives

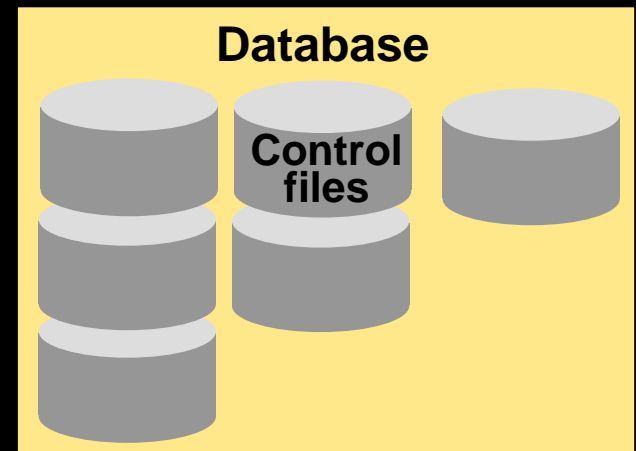
**After completing this lesson, you should be able to do the following:**

- **Explain the uses of the control file**
- **List the contents of the control file**
- **Multiplex and manage the control file**
- **Manage the control file with Oracle Managed Files**
- **Obtain control file information**

# Control File

The control file is a binary file that defines the current state of the physical database..

- Loss of the control file requires recovery
- Is read at MOUNT stage
- Is required to operate
- Is linked to a single database
- Should be multiplexed
- Maintains integrity of database
- **Sized initially by  
CREATE DATABASE**



# Control File Contents

**A control file contains the following entries:**

- **Database name and identifier**
- **Time stamp of database creation**
- **Tablespace names**
- **Names and locations of data files and redo log files**
- **Current redo log file sequence number**
- **Checkpoint information**
- **Begin and end of undo segments**
- **Redo log archive information**
- **Backup information**

# Multiplexing the Control File Using SPFILE

- **Alter the SPFILE**

```
SQL> ALTER SYSTEM SET control files =  
'$HOME/ORADATA/u01/ctrl01.ctl',  
'$HOME/ORADATA/u02/ctrl02.ctl' SCOPE=SPFILE;
```

- **Shutdown normal:**

```
SQL> shutdown
```

- **Create additional control files**

```
$ cp $HOME/ORADATA/u01/ctrl01.ctl  
$HOME/ORADATA/u02/ctrl02.ctl
```

- **Start the database:**

```
SQL> startup
```

# Multiplexing the Control File Using `init.ora`

- Shut down the database in a normal state:

```
SQL> shutdown normal
```

- Copy the existing control file to a new name and location:

```
$ cp control01.ctl .../DISK3/control02.ctl
```

- Add the new control file name to `init.ora`:

```
CONTROL_FILES = (/DISK1/control01.ctl,  
                /DISK3/control02.ctl)
```

- Start the database:

```
SQL> startup
```

# Managing Control Files with OMF

- **Control files are OMF created if the `CONTROL_FILES` parameter is not specified.**
- **OMF control files are located at `DB_CREATE_ONLINE_LOG_DEST_N`.**
- **Control file names are uniquely generated and displayed in the `alertSID.log` file when the files are created.**



# Obtaining Control File Information

Information about control file status and locations can be retrieved by querying the data dictionary.

- **V\$CONTROLFILE:** Lists the name and status of all control files associated with the instance.
- **V\$PARAMETER:** Lists status and location of all parameters.
- **V\$CONTROLFILE\_RECORD\_SECTION:** Provides information about the control file record sections.
- **SHOW PARAMETERS CONTROL\_FILES:** List the name, status, and location of the control files.

# Summary

**In this lesson, you should have learned how to:**

- **Multiplex the control file when using an `SPFILE`**
- **Multiplex the control file when using an `init.ora`**
- **Manage the control files using OMF**

# Practice 6 Overview

**This practice covers the following topics:**

- **Starting the database without a control file**
- **Multiplexing an existing control file**

# 7

## Maintaining Redo Log Files

# Objectives

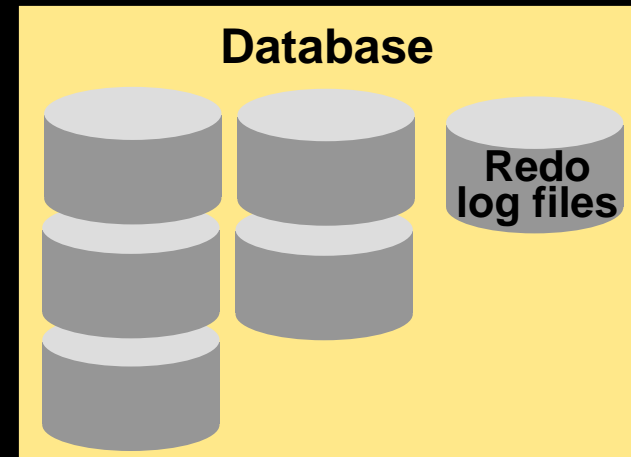
**After completing this lesson, you should be able to do the following:**

- **Explain the purpose of online redo log files**
- **Outline the structure of online redo log files**
- **Control log switches and checkpoints**
- **Multiplex and maintain online redo log files**
- **Manage online redo logs files with OMF**

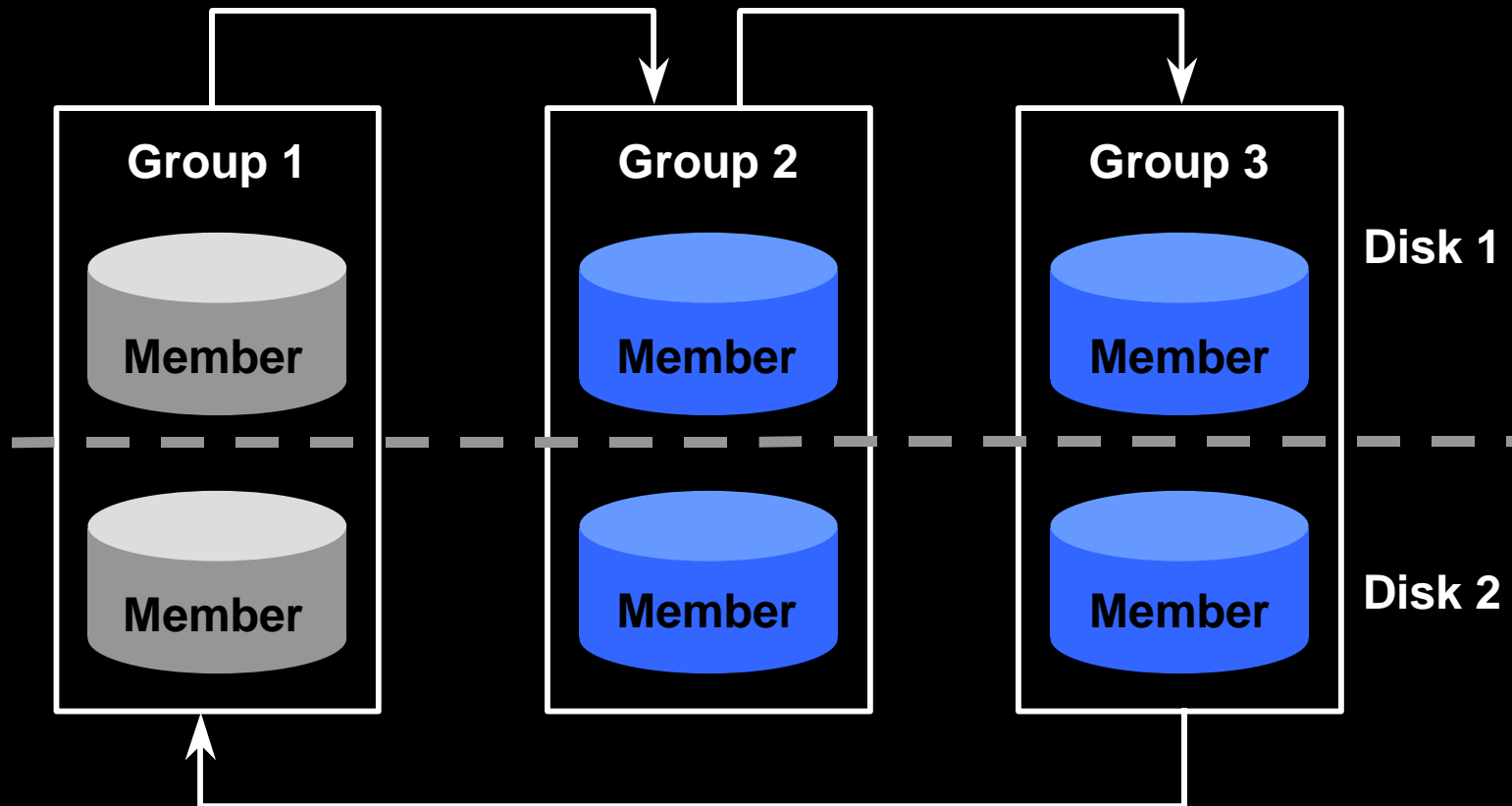
# Using Redo Log Files

Redo log files record all changes made to data and provide a recovery mechanism from a system or media failure.

- Redo log files are organized into groups.
- An Oracle database requires at least two groups.
- Each redo log within a group is called a member.



# Structure of Redo Log Files



# How Redo Logs Work

- Redo logs are used in a cyclic fashion.
- When a redo log file is full, LGWR will move to the next log group.
  - This is called a log switch
  - Checkpoint operation also occurs
  - Information is written to the control file



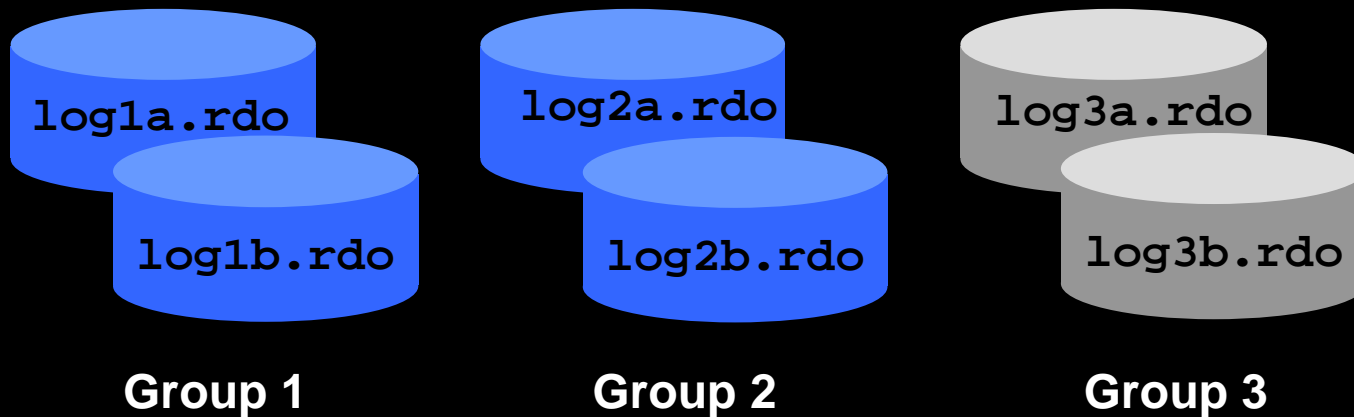
# Forcing Log Switches and Checkpoints

- Log switches can be forced using the **ALTER SYSTEM SWITCH LOGFILE** command.
- Checkpoints can be forced using:
  - Setting **FAST\_START\_MTTR\_TARGET** parameter
  - **ALTER SYSTEM CHECKPOINT** command

```
ALTER SYSTEM CHECKPOINT;
```

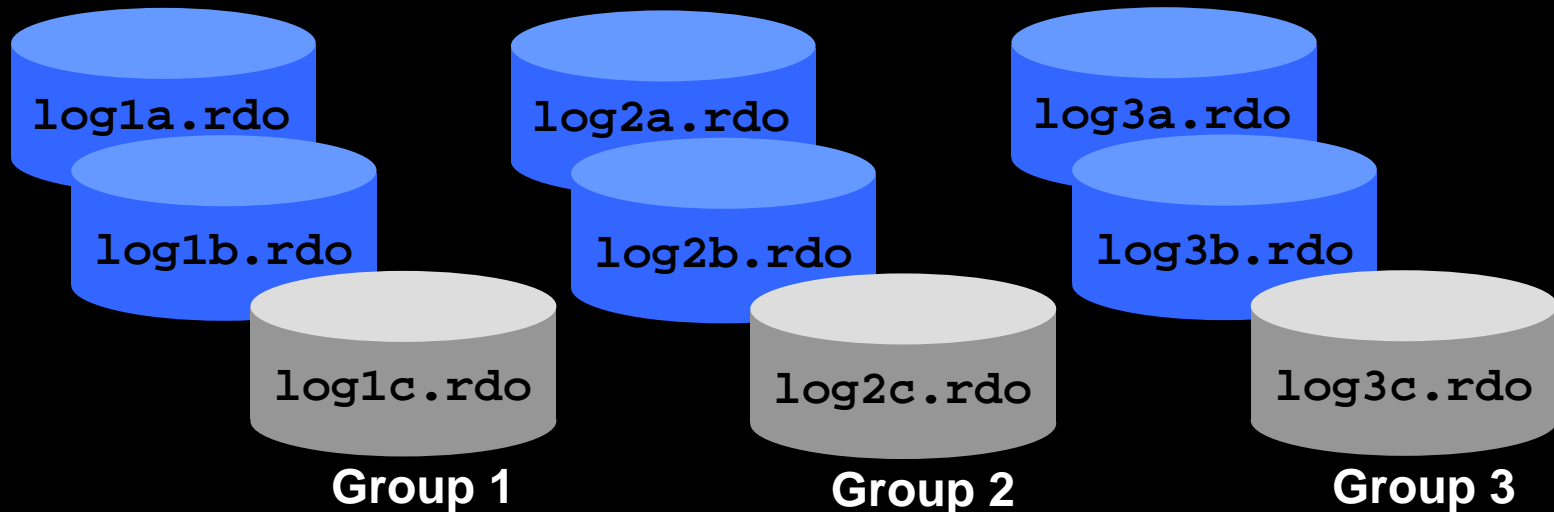
# Adding Online Redo Log Groups

```
ALTER DATABASE ADD LOGFILE GROUP 3  
( '$HOME/ORADATA/u01/log3a.rdo',  
  '$HOME/ORADATA/u02/log3b.rdo' )  
SIZE 1M;
```



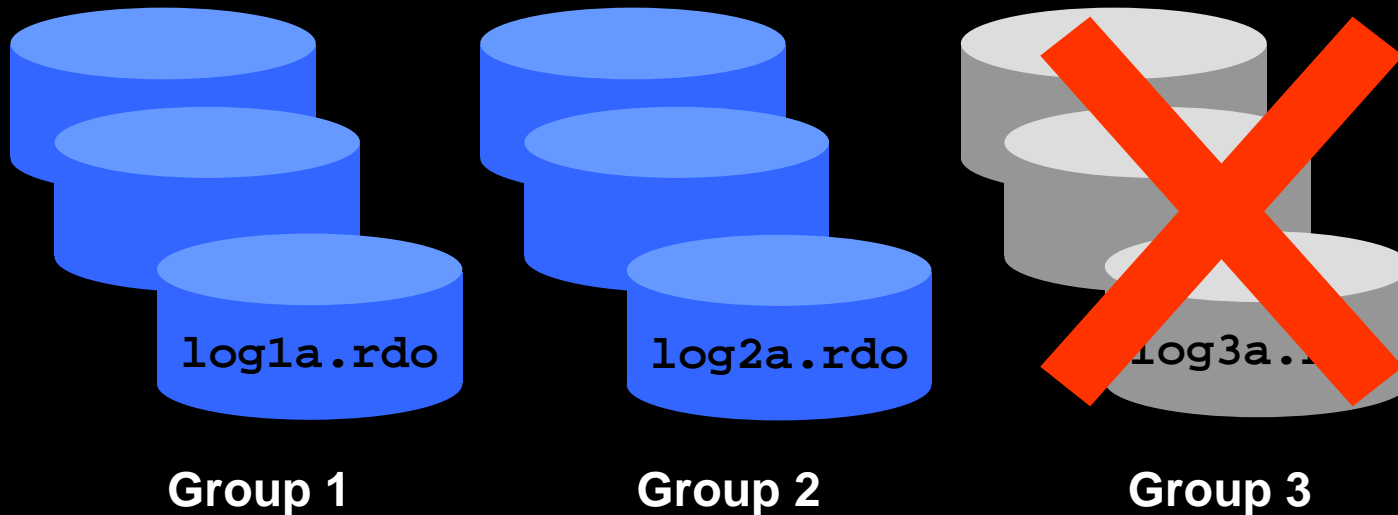
# Adding Online Redo Log Members

```
ALTER DATABASE ADD LOGFILE MEMBER  
'$HOME/ORADATA/u04/log1c.rdo' TO GROUP 1,  
'$HOME/ORADATA/u04/log2c.rdo' TO GROUP 2,  
'$HOME/ORADATA/u04/log3c.rdo' TO GROUP 3;
```



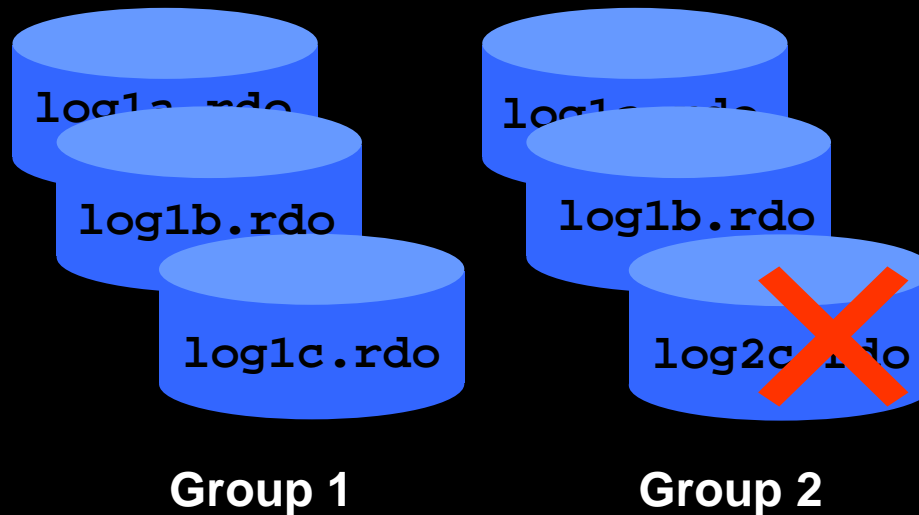
# Dropping Online Redo Log Groups

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```



# Dropping Online Redo Log Members

```
ALTER DATABASE DROP LOGFILE MEMBER  
'$HOME/ORADATA/u04/log3c.rdo';
```



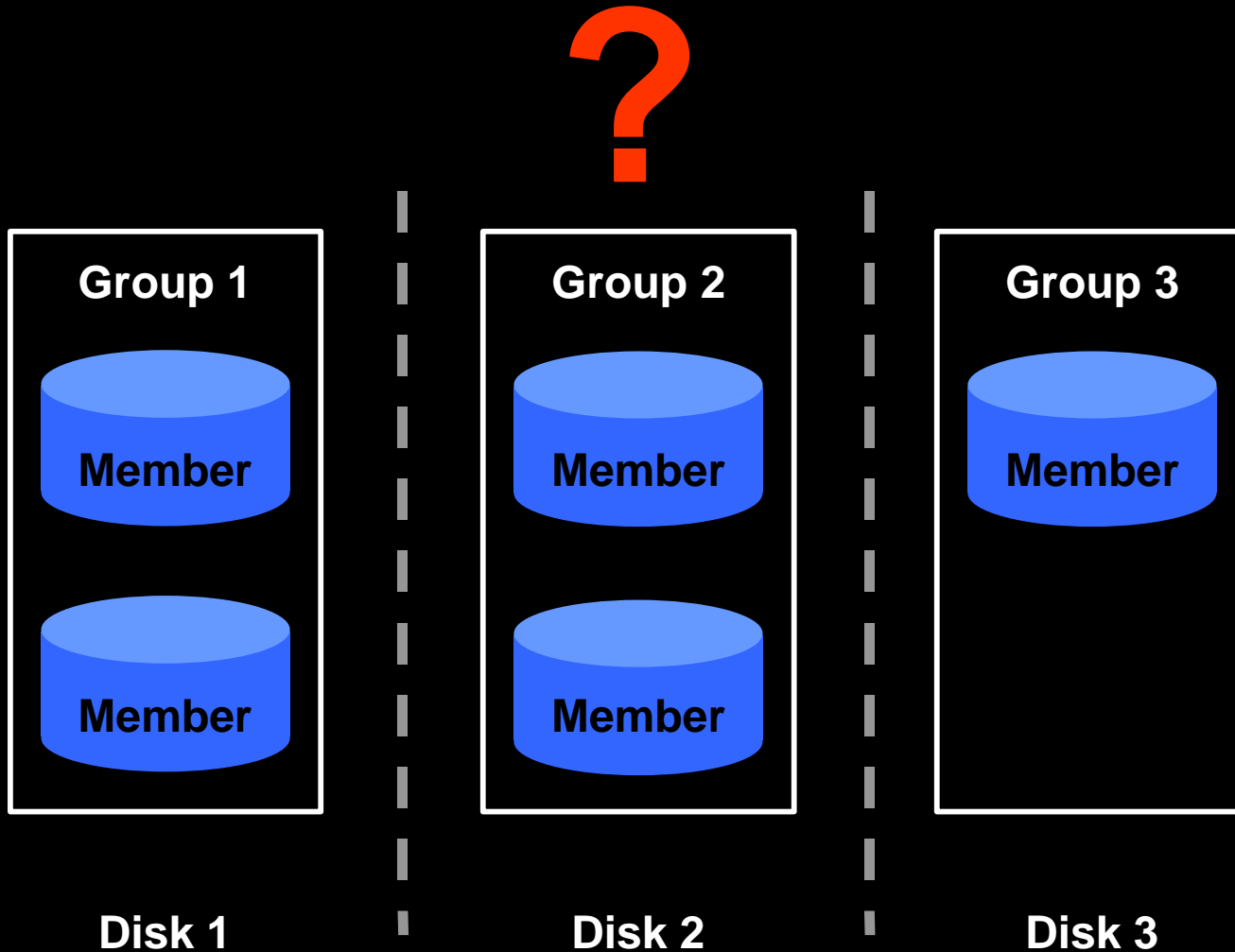
# Clearing, Relocating, or Renaming Online Redo Log Files

- **Clearing online redo log files:**

```
ALTER DATABASE CLEAR LOGFILE  
' $HOME/ORADATA/u01/log2a.rdo' ;
```

- **Relocating or renaming online redo log files can be accomplished by adding new members and dropping old members.**

# Online Redo Log Configuration



# Managing Online Redo Logs with OMF

- A complete group can be added with no file specification:

```
ALTER DATABASE ADD LOGFILE;
```

- If a group is dropped, all the corresponding OMF files are deleted at the OS level:

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```



# Obtaining Group and Member Information

Information about group and members can be obtained by querying the data dictionary.

- V\$LOG
- V\$LOGFILE

# Archived Redo Log Files

**Filled online redo log files can be archived.**

- **Two advantages exist to archiving redo logs:**
  - **Recovery:** A database backup, together with online and archived redo log files can guarantee recovery of all committed transactions.
  - **Backup:** Can be performed while the database is open.
- **By default a database is created in NOARCHIVELOG mode.**

# Archived Redo Log Files

- Archiving redo log files is accomplished by ARCn (Archiver) or manually through SQL statements.
- An entry in the control file recording the archive log name, log sequence number, and high and low SCN number is made whenever a redo log is successfully archived.
- A filled redo log file cannot be reused until a checkpoint has taken place and the redo log file has been backed up the ARCn process.
- Archived redo log files can be multiplexed.
- Archived redo log files must be maintained by the DBA.

# Summary

**In this lesson, you should have learned how to:**

- **Explain the use of online redo log files**
- **Obtain redo log information**
- **Control log switches and checkpoints**
- **Multiplex and maintain online redo log files**
- **Manage online redo log files with OMF**

# Practice 7 Overview

**This practice covers the following topics:**

- **Creating and adding redo log file groups and members.**
- **Dropping redo log file groups and members.**

# 8

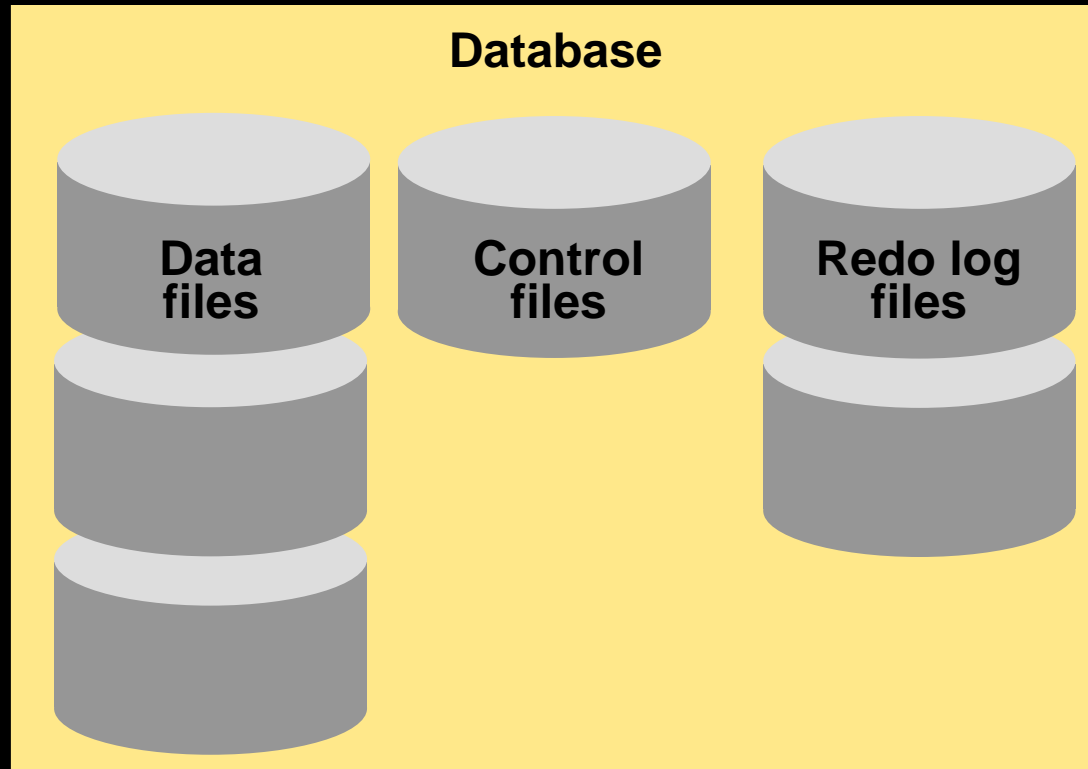
## Managing Tablespaces and Data files

# Objectives

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

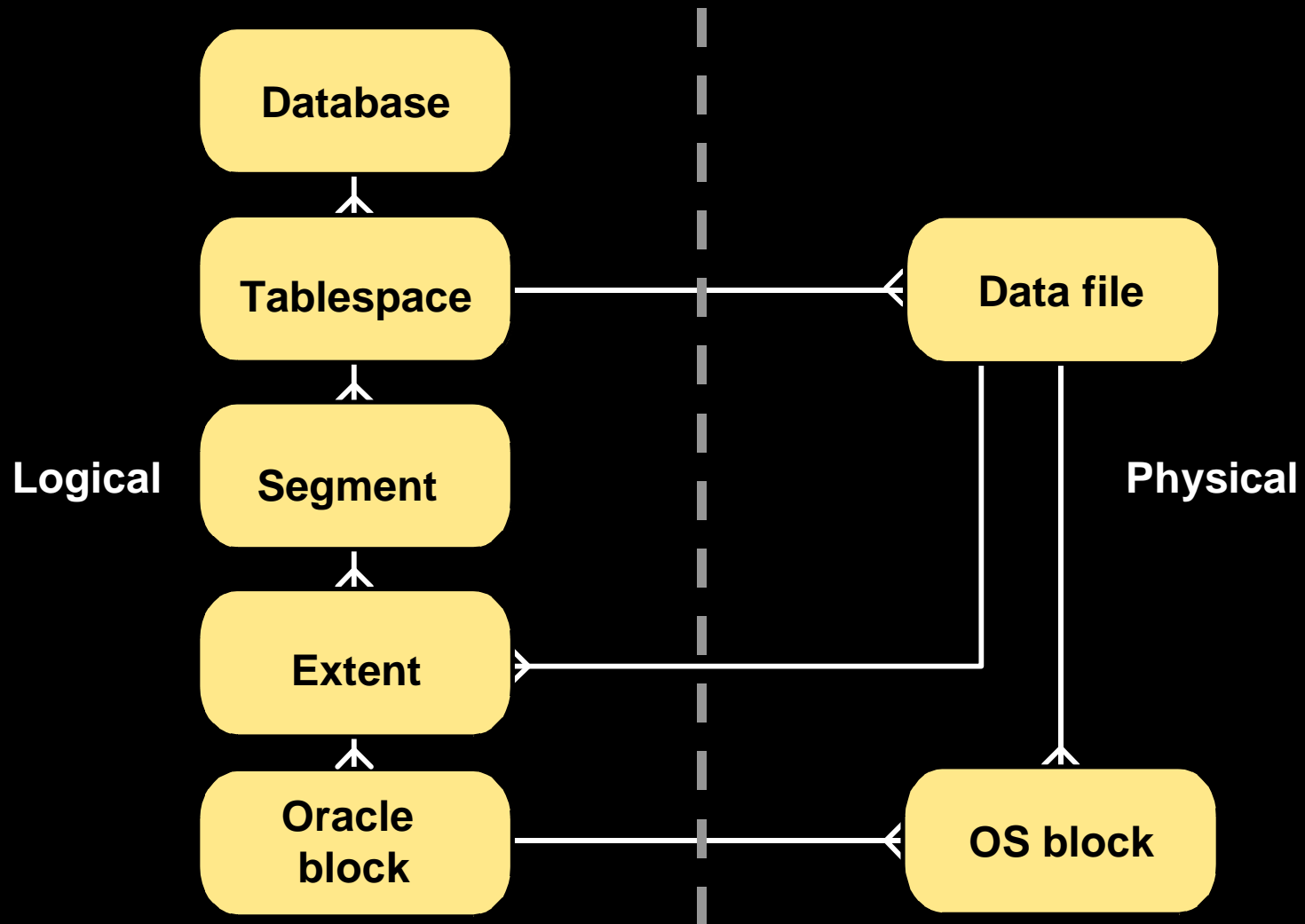
- **Describe the logical structure of the database**
- **Create tablespaces**
- **Change the size of tablespaces**
- **Allocate space for temporary segments**
- **Change the status of tablespaces**
- **Change the storage settings of tablespaces**
- **Implement Oracle Managed Files**

# Overview





# Database Storage Hierarchy



# SYSTEM and Non-SYSTEM Tablespaces

- **SYSTEM tablespace:**
  - Created with the database
  - Contains the data dictionary
  - Contains the SYSTEM undo segment
- **Non-SYSTEM tablespaces:**
  - Separate segments
  - Ease space administration
  - Control amount of space allocated to a user

# Creating Tablespaces

```
CREATE TABLESPACE userdata  
  DATAFILE '/u01/oradata/userdata01.dbf' SIZE 100M  
  AUTOEXTEND ON NEXT 5M MAXSIZE 200M;
```

# Space Management in Tablespaces

- **Locally managed tablespaces:**
  - Free extents recorded in bitmap
  - Each bit corresponds to a block or group of blocks
  - Bit value indicates free or used
- **Dictionary-managed tablespaces:**
  - Default method
  - Free extents recorded in data dictionary tables

# Locally Managed Tablespaces

```
CREATE TABLESPACE userdata  
  DATAFILE '/u01/oradata/userdata01.dbf' SIZE 500M  
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

- Reduced contention on data dictionary tables
- No undo generated when space allocation or deallocation occurs
- No coalescing required

# Dictionary Managed Tablespaces

```
CREATE TABLESPACE userdata  
  DATAFILE '/u01/oradata/userdata01.dbf' SIZE 500M  
  EXTENT MANAGEMENT DICTIONARY  
  DEFAULT STORAGE ( initial 1M NEXT 1M );
```

- Extents are managed in the data dictionary
- Each segment stored in the tablespace can have a different storage clause
- Coalescing required

# Changing the Storage Settings

```
ALTER TABLESPACE userdata  
    MINIMUM EXTENT 2M;
```

```
ALTER TABLESPACE userdata  
    DEFAULT STORAGE (  
        INITIAL      2M  
        NEXT         2M  
        MAXEXTENTS 999 );
```

# Undo Tablespace

- Used to store undo segments
- Cannot contain any other objects
- Extents are locally managed
- Can only use the **DATAFILE** and **EXTENT MANAGEMENT** clauses of the **CREATE TABLESPACE** command

```
CREATE UNDO TABLESPACE undo1  
    DATAFILE '/u01/oradata/undo101.dbf' SIZE 40M;
```



# Temporary Tablespace

- Used for sort operations
- Cannot contain any permanent objects
- Locally managed extents recommended

```
CREATE TEMPORARY TABLESPACE temp  
  TEMPFILE '/u01/oradata/temp01.dbf' SIZE 500M  
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 10M;
```

# Default Temporary Tablespace

- Allows you to specify a databasewide default temporary tablespace
- Eliminates the use of the `SYSTEM` tablespace for storing temporary data
- Can be created using the `CREATE DATABASE` or `ALTER DATABASE` command.
- When created with the `CREATE DATABASE` command, the default temporary tablespace is locally managed

# Restrictions on Default Temporary Tablespace

- It cannot be dropped until after a new default is made available.
- It cannot be taken offline.
- You cannot alter the default temporary tablespace to a permanent tablespace.

# Offline Status

- Offline tablespace is not available for data access.
- Some tablespaces must be online:
  - SYSTEM
  - Tablespaces with active undo segments
  - Default temporary

- To take a tablespace offline:

```
ALTER TABLESPACE userdata OFFLINE;
```

- To bring a tablespace online:

```
ALTER TABLESPACE userdata ONLINE;
```

# Read-Only Tablespaces

```
ALTER TABLESPACE userdata READ ONLY;
```

- **Tablespace available only for read operations**
- **Objects can be dropped from tablespace**
- **To create a read-only tablespace on a removable media drive:**
  - **ALTER TABLESPACE...READ ONLY;**
  - **Move the data file to the WORM drive**
  - **ALTER TABLESPACE...RENAME DATAFILE...;**

# Dropping Tablespaces

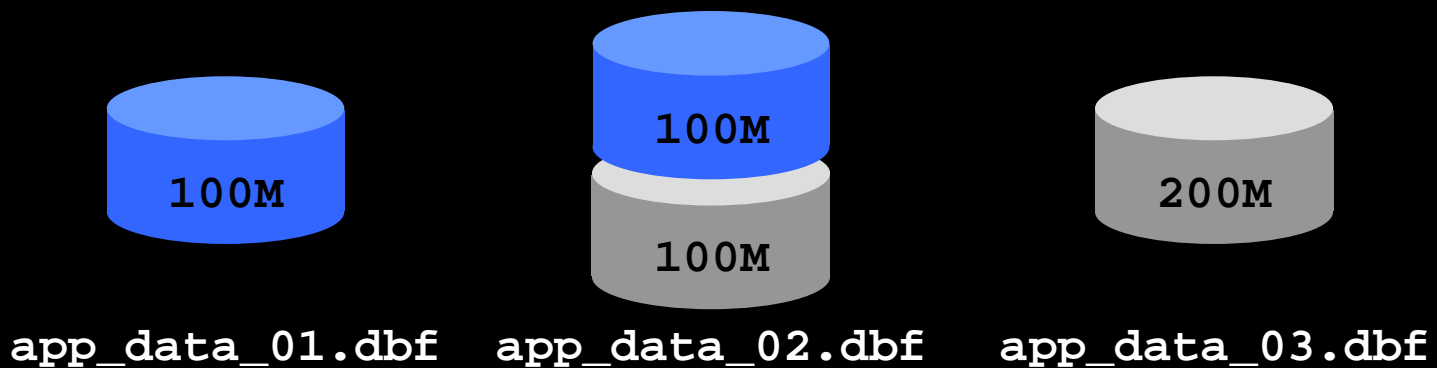
- Tablespace removed from data dictionary
- Optionally, contents removed from data dictionary
- OS files can be deleted with the optional **AND DATAFILES** clause:

```
DROP TABLESPACE userdata  
INCLUDING CONTENTS AND DATAFILES;
```

# Resizing a Tablespace

- Add a data file
- Change the size of a data file:
  - Automatically
  - Manually

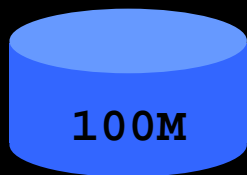
## Tablespace APP\_DATA



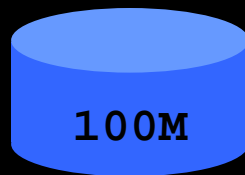
# Enabling Automatic Extension of Data Files

```
ALTER DATABASE DATAFILE  
  '/u01/oradata/userdata02.dbf' SIZE 200M  
  AUTOEXTEND ON NEXT 10M MAXSIZE 500M;
```

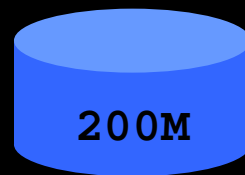
## Tablespace APP\_DATA



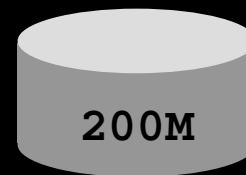
app\_data\_01.dbf



app\_data\_02.dbf



app\_data\_03.dbf



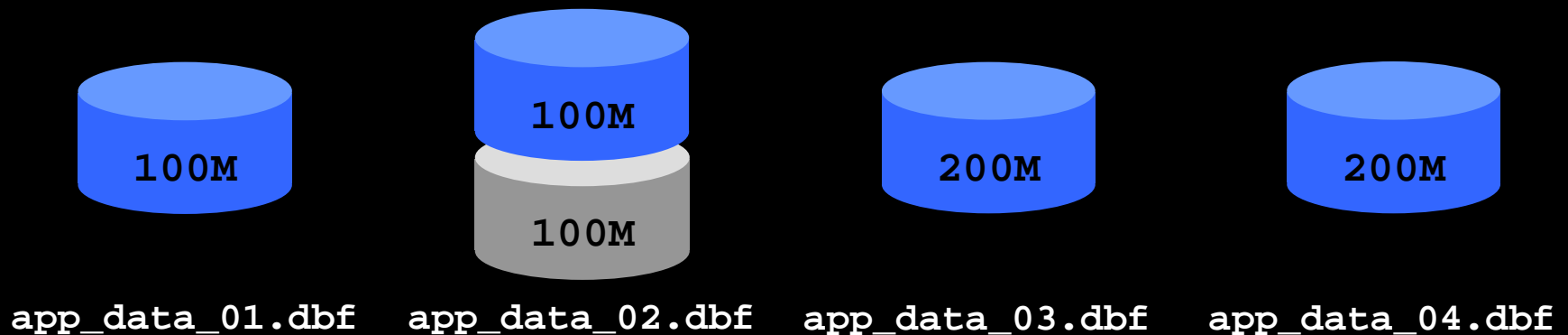
app\_data\_04.dbf



# Changing the Size of Data Files Manually

```
ALTER DATABASE  
  DATAFILE '/u03/oradata/userdata02.dbf'  
  RESIZE 200M;
```

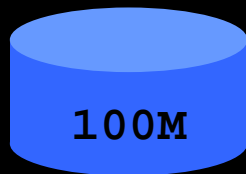
## Tablespace APP\_DATA



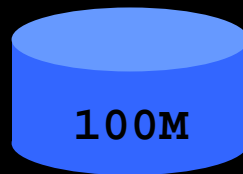
# Adding Data Files to a Tablespace

```
ALTER TABLESPACE app_data  
  ADD DATAFILE '/u01/oradata/userdata03.dbf'  
  SIZE 200M;
```

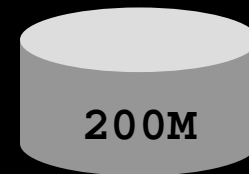
## Tablespace APP\_DATA



100M



100M



200M

app\_data\_01.dbf

app\_data\_02.dbf

app\_data\_03.dbf

# Moving Data Files:

## ALTER TABLESPACE

- The tablespace must be offline.
- The target data files must exist.

```
ALTER TABLESPACE userdata
  RENAME
    DATAFILE      '/u01/oradata/userdata01.dbf'
  TO               '/u01/oradata/userdata01.dbf' ;
```

# Moving Data Files:

## ALTER DATABASE

- The database must be mounted.
- The target data file must exist.

```
ALTER DATABASE RENAME  
  FILE '/u01/oradata/system01.dbf'  
  TO '/u03/oradata/system01.dbf';
```

# Configuring Oracle Managed Files for Tablespace Creation

- Creating a tablespace with OMF requires the configuration of one initialization parameter.
- **DB\_CREATE\_FILE\_DEST**: Set to give the default location for data files.
- The initialization parameter can be set in an initialization file or set dynamically with the **ALTER SYSTEM** command:

```
ALTER SYSTEM SET  
  db_create_file_dest = '/u01/oradata/db01';
```

# Creating Tablespaces with OMF

- With OMF configured the **DATAFILE** clause of the **CREATE TABLESPACE** command is not required.

```
CREATE TABLESPACE apps2_data DATAFILE SIZE 20M;
```

- The data file is created in the file system specified by **DB\_CREATE\_FILE\_DEST**.
- By default files are 100M in size and set to autoextend with an unlimited restriction.
- When the tablespace is dropped, all files are also deleted at the OS level.
- An OMF can be added to an existing tablespace.

# Obtaining Tablespace Information

- **Tablespace information:**
  - DBA\_TABLESPACES
  - V\$TABLESPACE
- **Data file information:**
  - DBA\_DATA\_FILES
  - V\$DATAFILE
- **Temp file information:**
  - DBA\_TEMP\_FILES
  - V\$TEMPFILE

# Summary

**In this lesson, you should have learned how to:**

- **Use tablespaces to separate data**
- **Resize tablespaces by:**
  - Adding data files
  - Extending data files
- **Use locally managed tablespaces**
- **Use temporary tablespaces**
- **Implement Oracle Managed Files**



# Practice 8 Overview

**This practice covers the following topics:**

- **Creating tablespaces**
- **Modifying tablespaces**
- **Configuring and creating a tablespace with OMF**

# 9

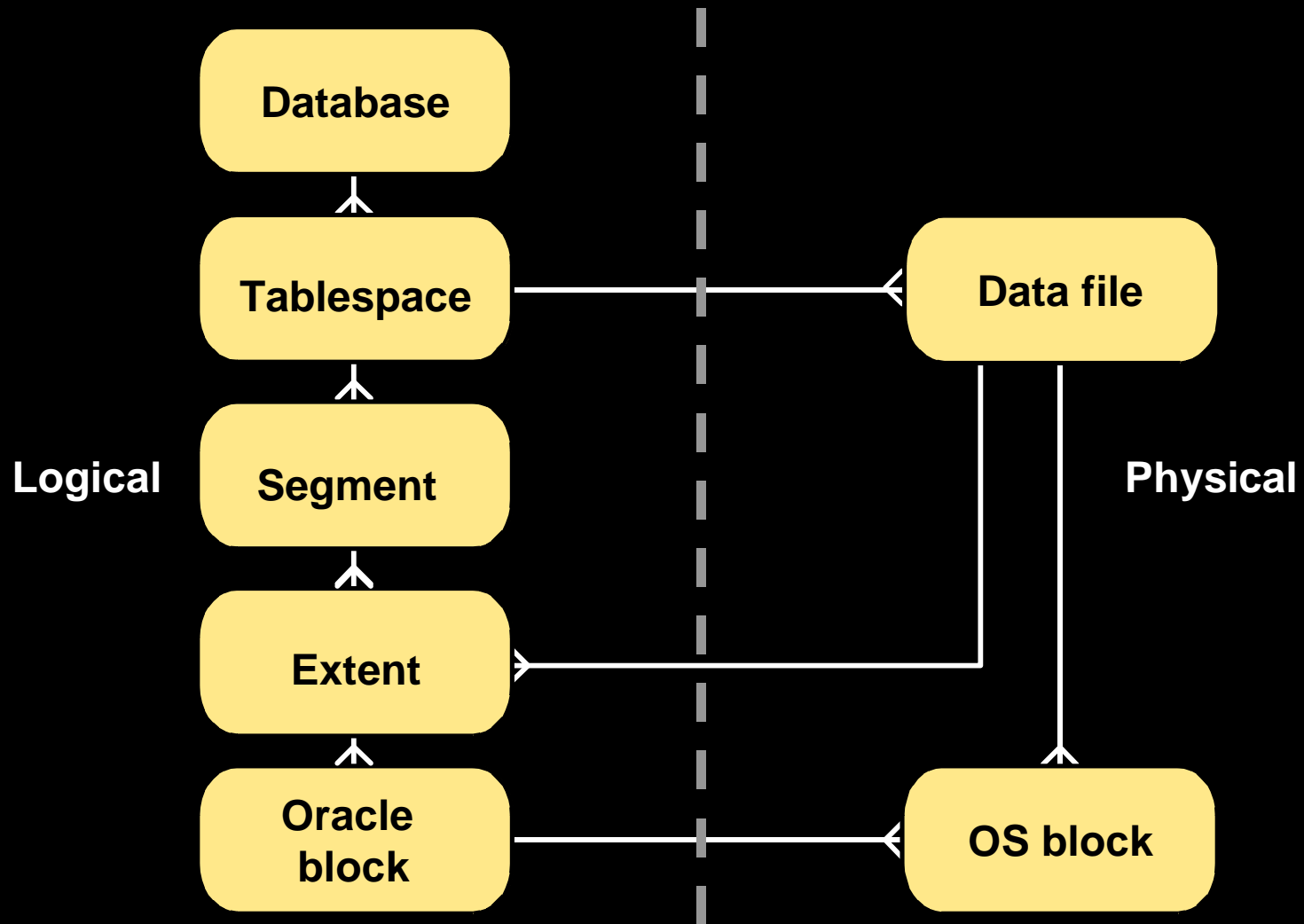
## Storage Structure and Relationships

# Objectives

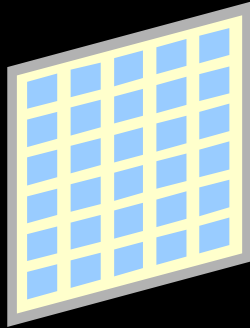
**After completing this lesson, you should be able to do the following:**

- **Describe the logical structure of the database**
- **List the segment types and their uses**
- **List the keywords that control block space usage**
- **Obtain information about storage structures from the data dictionary**
- **List the criteria for separating segments**

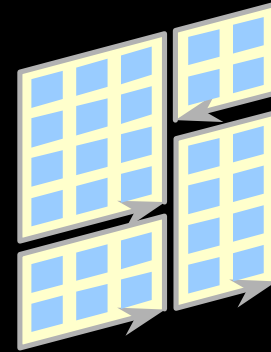
# Overview



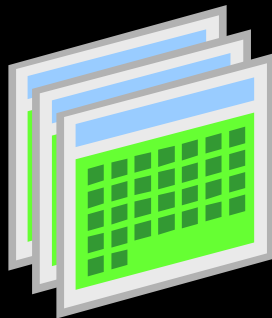
# Types of Segments



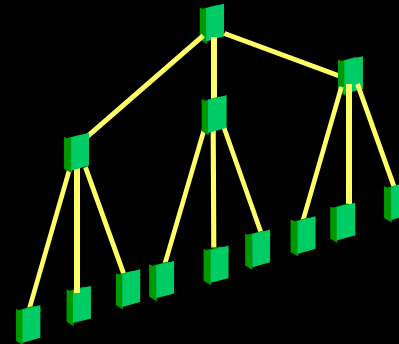
**Table**



**Table  
partition**

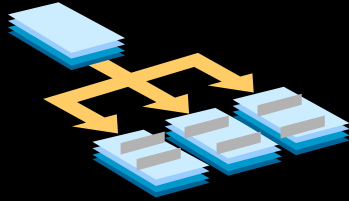


**Cluster**

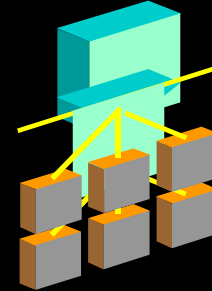


**Index**

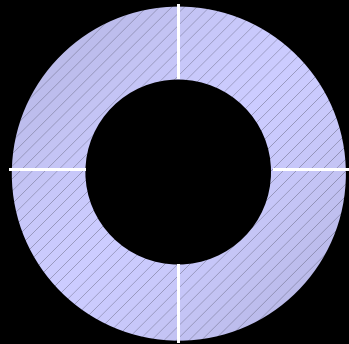
# Types of Segments



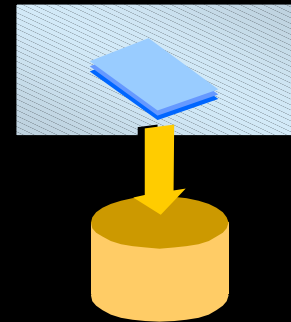
**Index-organized  
table**



**Index  
partition**

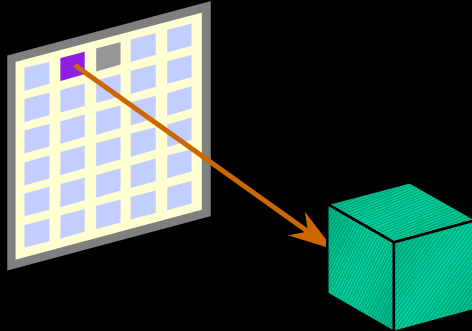


**Undo  
segment**

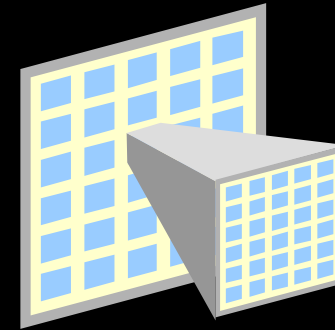


**Temporary  
segment**

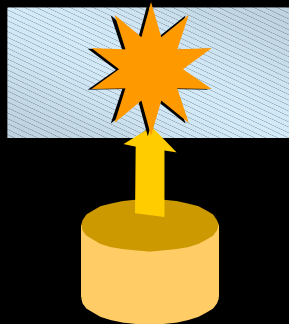
# Types of Segments



**LOB  
segment**

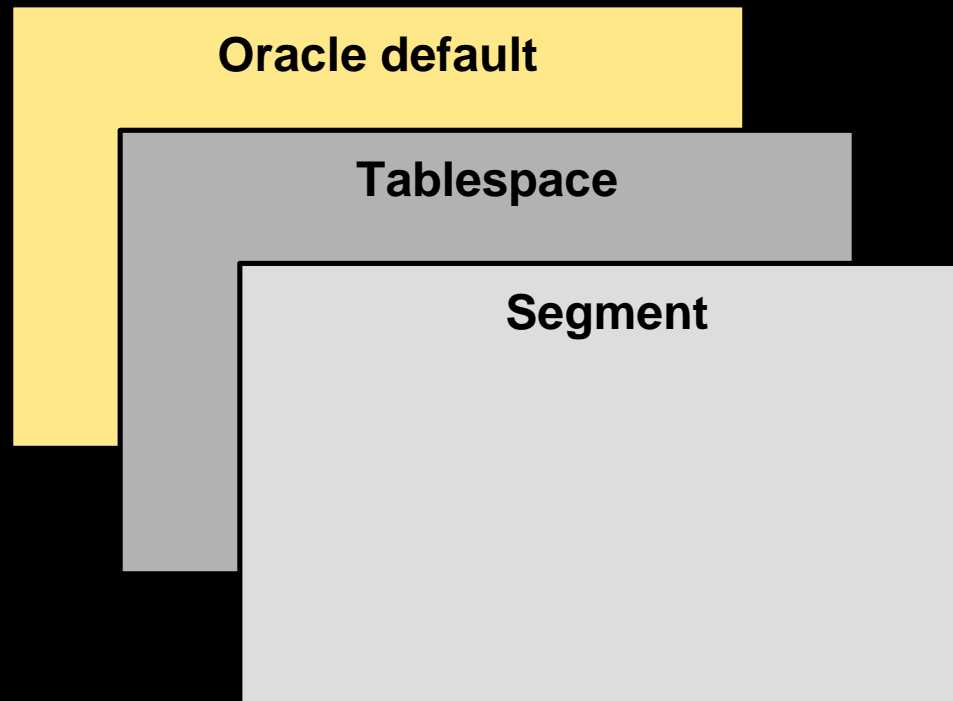


**Nested table**



**Bootstrap  
segment**

# Storage Clause Precedence



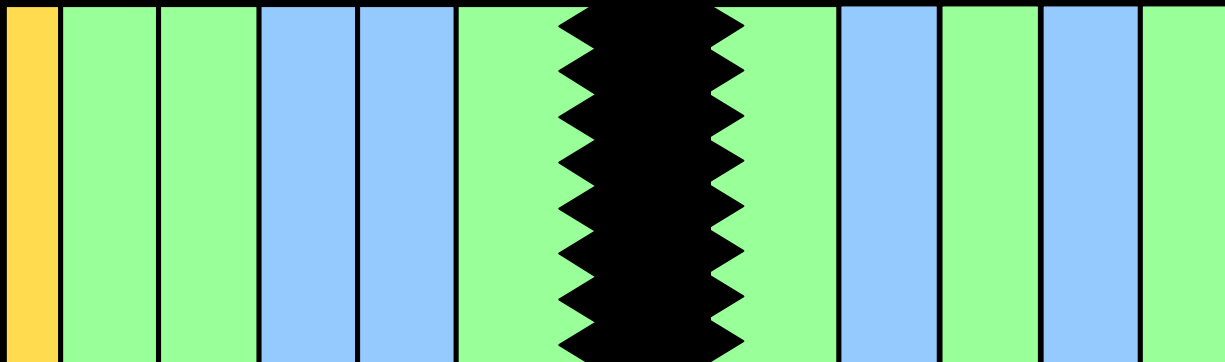


# Extent Allocation and Deallocation

- **Allocated when the segment is:**
  - Created
  - Extended
  - Altered
- **Deallocated when the segment is:**
  - Dropped
  - Altered
  - Truncated

# Used and Free Extents

Data file



File header



Used extent

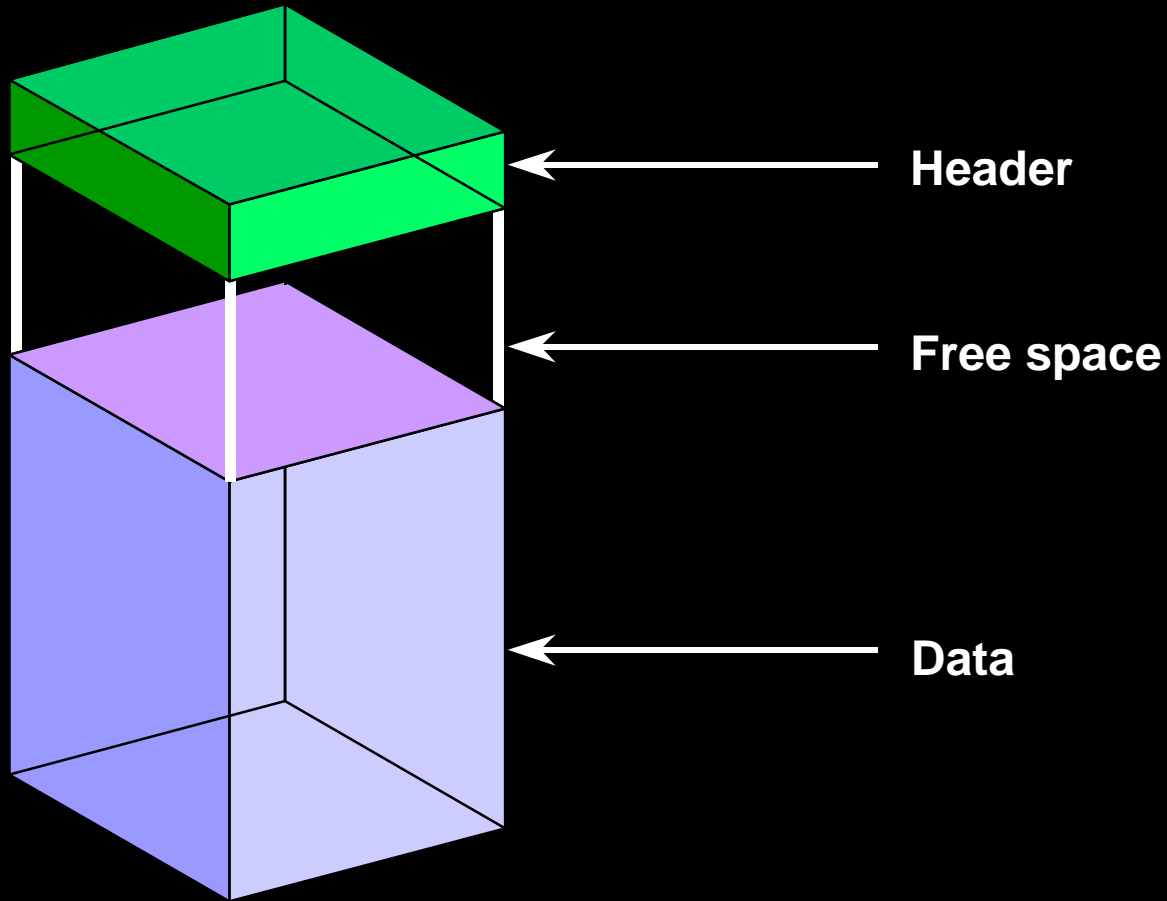


Free extent

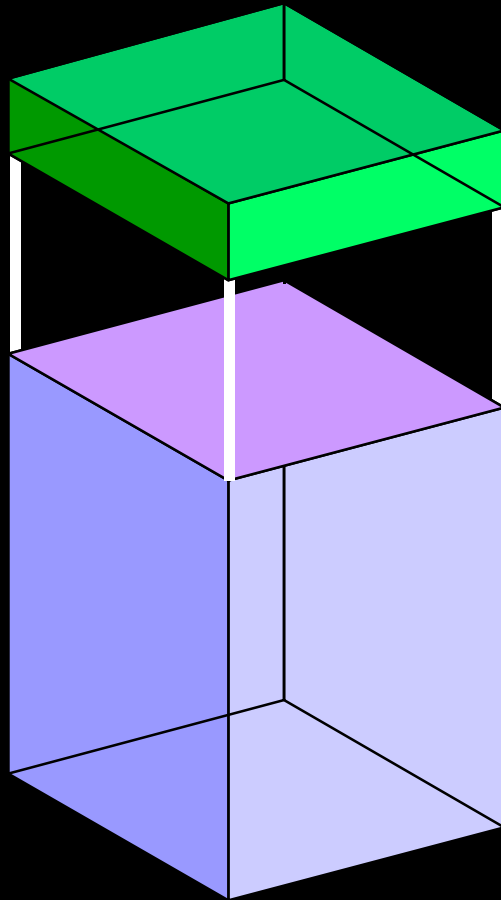
# Database Block

- **Minimum unit of I/O**
- **Consists of one or more OS blocks**
- **Set at tablespace creation**
- **DB\_BLOCK\_SIZE is the default block size**

# Database Block Contents



# Block Space Utilization Parameters



INITTRANS

MAXTRANS

PCTFREE

PCTUSED

# Data Block Management

**Two methods are available for managing data blocks:**

- **Automatic Segment-Space Management**
- **Manual Configuration**

# Automatic Segment-Space Management

- It is method of managing free space inside database segments
- Tracking in-segment free and used space is done using bitmaps as opposed to free lists
- This method provides:
  - Ease of management
  - Better space utilization
  - Better performance for concurrent INSERT operations
- Restriction: Can not be used for tablespaces which will contain LOBs.

# Automatic Segment-Space Management

- **Bitmap segments contain a bitmap that describes the status of each block in the segment with respect to its available space.**
- **The map is contained in a separate set of blocks referred to as bitmapped blocks (BMBs).**
- **When inserting a new row, the server searches the map for a block with sufficient space.**
- **As the amount of space available in a block changes, its new state is reflected in the bitmap.**



# Configuring Automatic Segment-Space Management

- Automatic segment-space management can be enabled at the tablespace level only, for locally managed tablespaces.

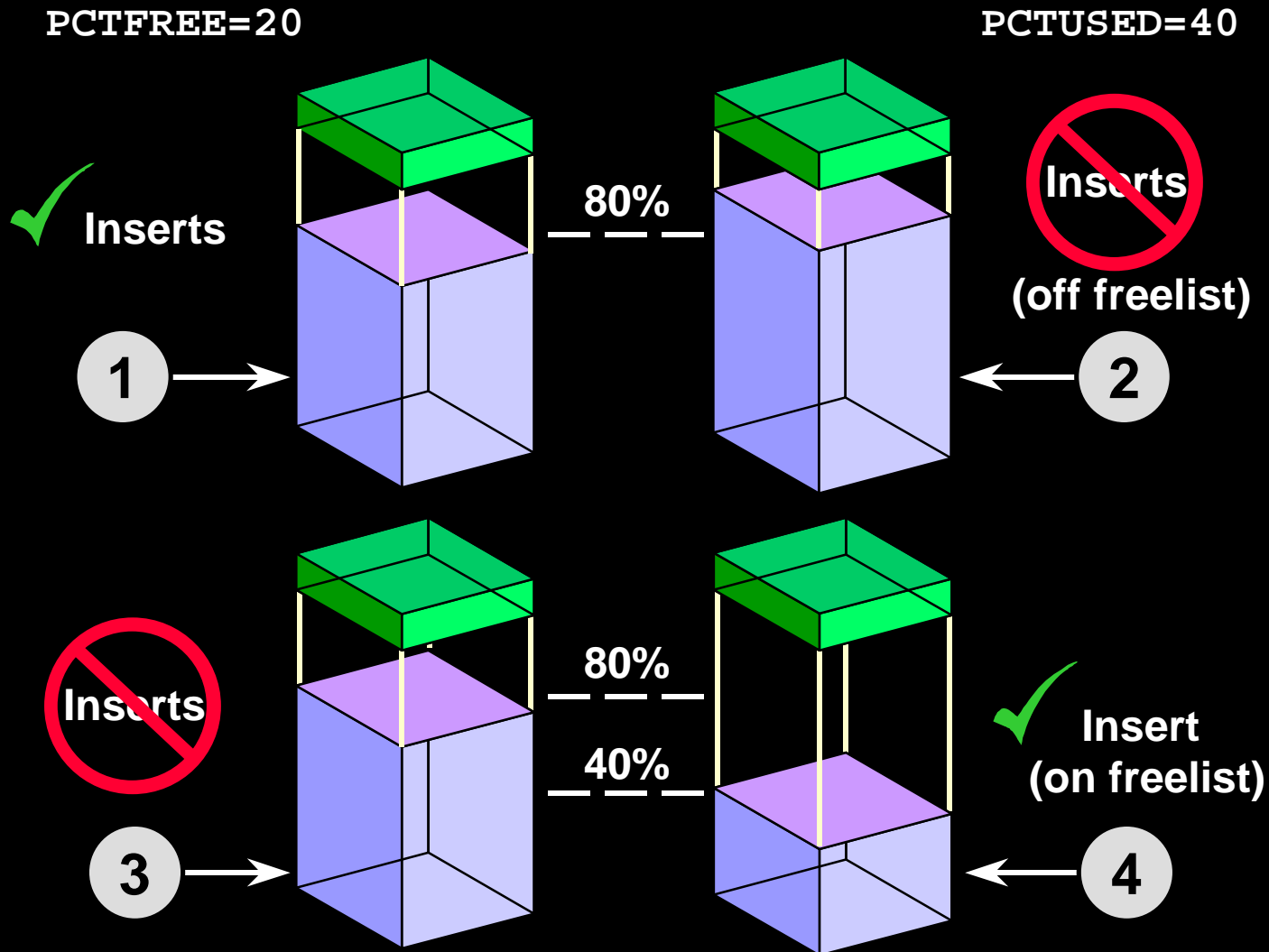
```
CREATE TABLESPACE data02  
DATAFILE '/u01/oradata/data02.dbf' SIZE 5M  
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K  
SEGMENT SPACE MANAGEMENT AUTO;
```

- After a tablespace is created, the specifications apply to all segments created in the tablespace.

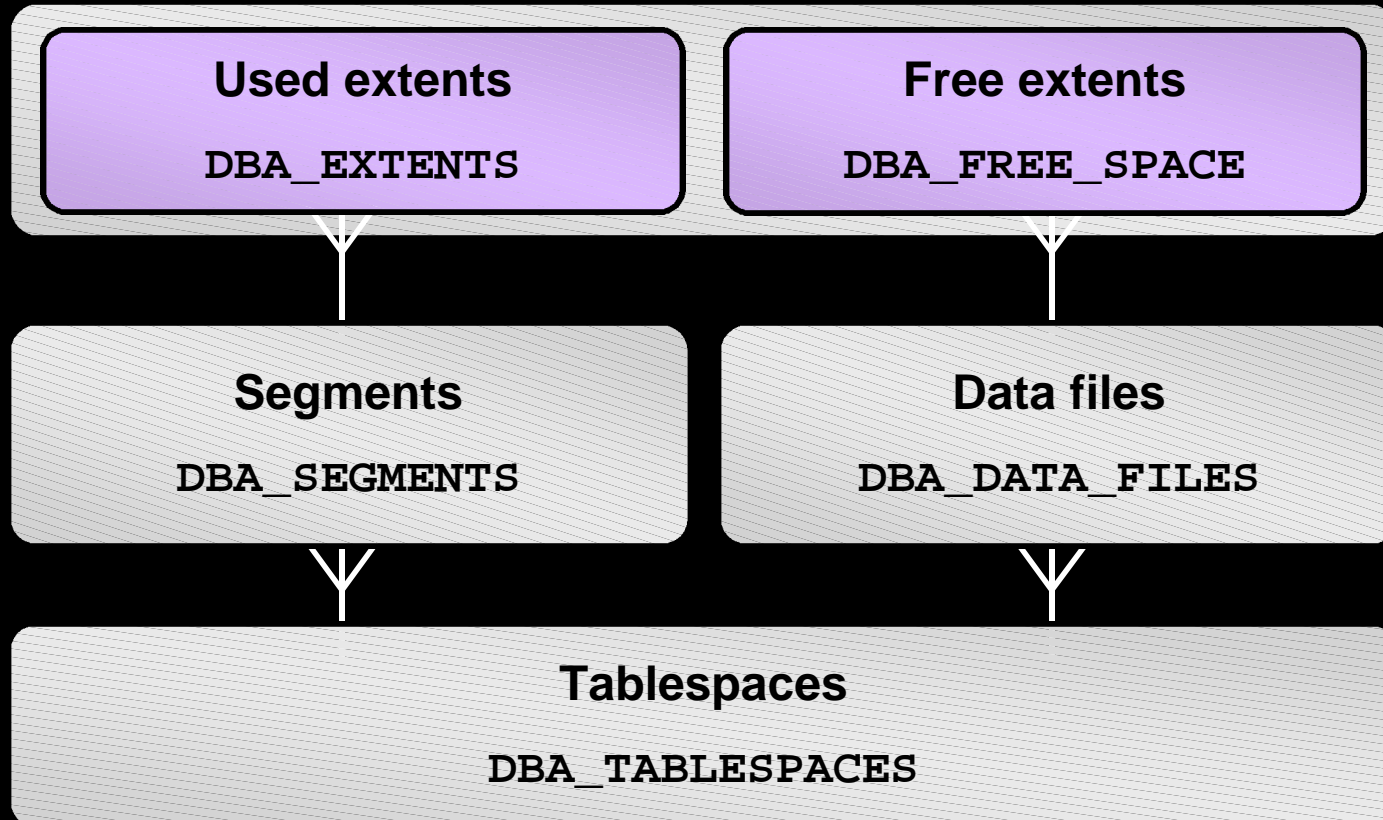
# Manual Data Block Management

- **Allows you to configure data blocks manually using parameters such as:**
  - **PCTFREE**
  - **PCTUSED**
  - **FREELIST**
- **Only method available in previous Oracle versions**

# Block Space Usage



# Data Dictionary Views



# Obtaining Storage Information

- **Data Dictionary Views**

- **DBA\_TABLESPACES**
- **DBA\_DATA\_FILES**
- **DBA\_SEGMENTS**
- **DBA\_EXTENTS**
- **DBA\_FREE\_SPACE**

# Summary

**In this lesson, you should have learned how to:**

- **Use tablespaces to:**
  - **Separate segments to ease administration**
  - **Control user's space allocation**
- **Categorize segments by the type of information stored in the segment**
- **Determine extent sizes using the storage clause**
- **Control block space utilization**

# Practice 9 Overview

**This practice covers the following topics:**

- **Creating an SPFILE**
- **Starting up and shutting down the database in different modes**

# 10

## Managing Undo Data



# Objectives

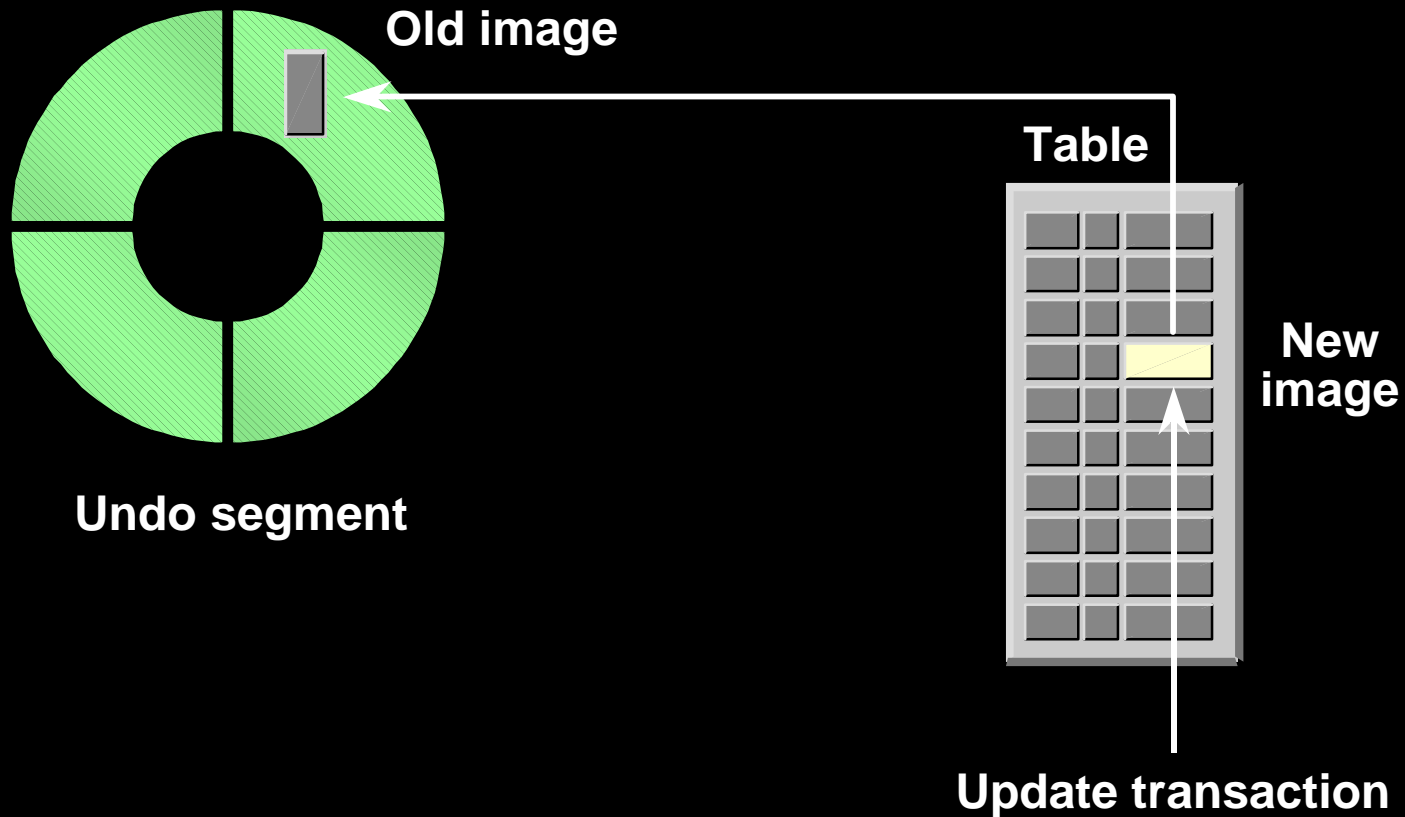
**After completing this lesson, you should be able to do the following:**

- **Describe the purpose of undo data**
- **Implement Automatic Undo Management**
- **Create and configure undo segments**
- **Obtain undo segment information from the data dictionary**

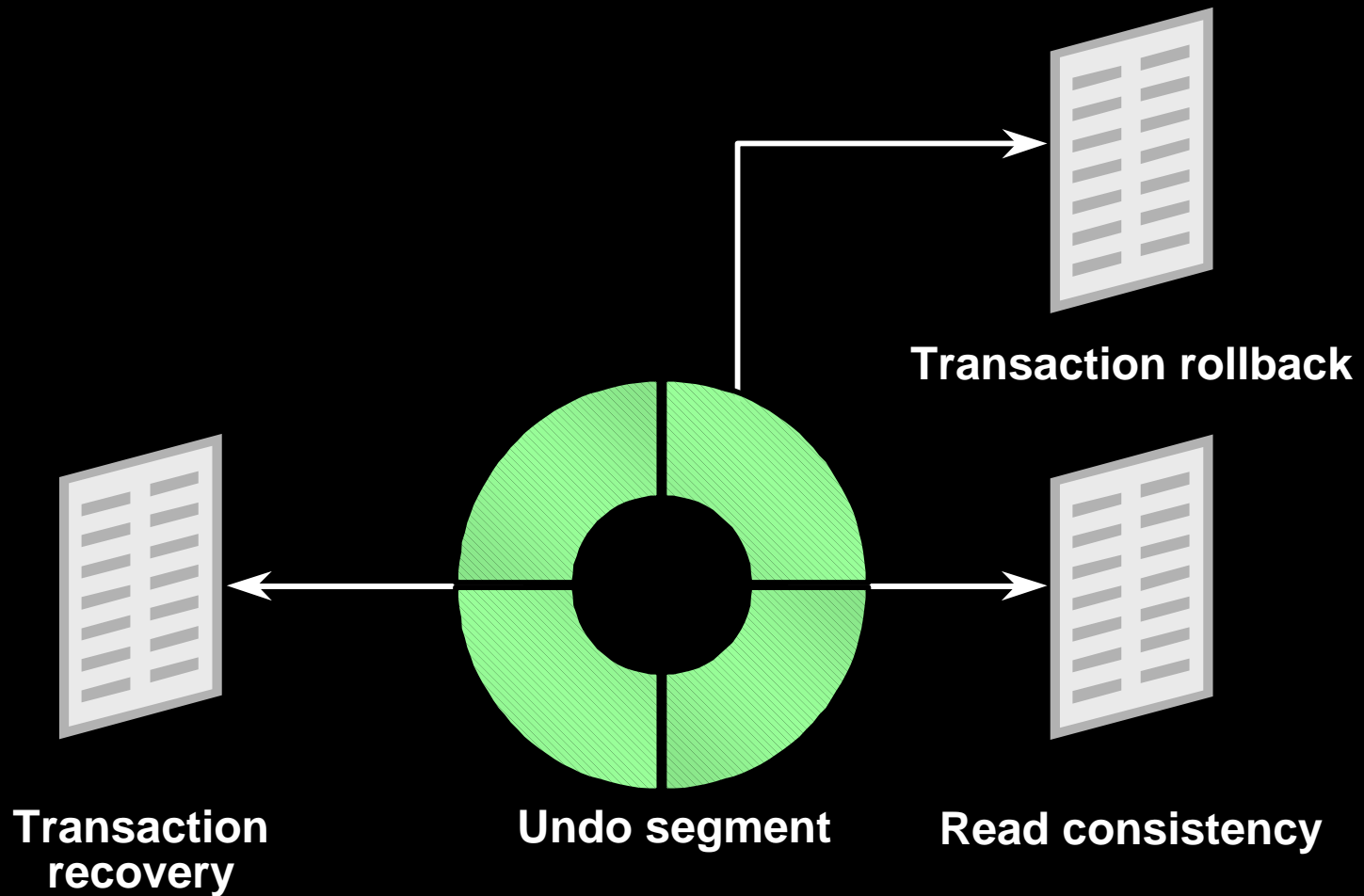
# Managing Undo Data Overview

- Two methods for managing undo data exists
  - Automatic Undo Management
  - Manual Undo Management
- This lesson discusses Automatic Undo Management
- The term *undo* replaces what was known in previous versions of Oracle as *rollback*

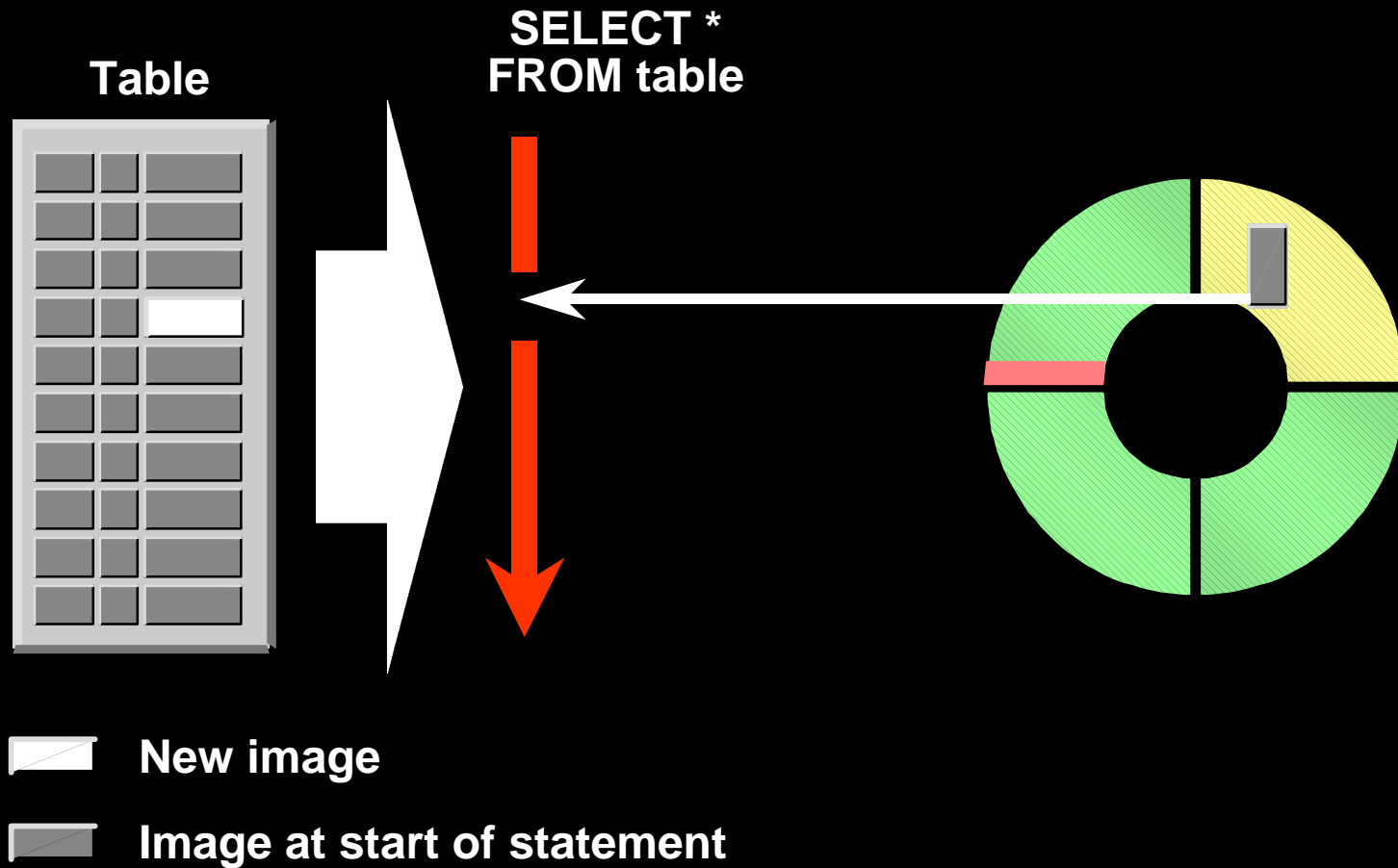
# Undo Segment



# Undo Segments: Purpose



# Read Consistency



# Types of Undo Segments

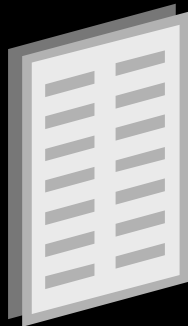
- **SYSTEM:** Used for objects in the **SYSTEM** tablespace
- **Non-SYSTEM:** Used for objects in other tablespaces:
  - **Auto Mode:** Requires an **UNDO** tablespace
  - **Manual Mode:**
    - **Private:** Acquired by a single instance
    - **Public:** Acquired by any instance
- **Deferred:** Used when tablespaces are taken offline immediate, temporary, or for recovery

# Automatic Undo Management: Concepts

- Undo data is managed using an `UNDO` tablespace
- You allocate one `UNDO` tablespace per instance with enough space for the workload of the instance.
- The Oracle server automatically maintains undo data within the `UNDO` tablespace.

# Automatic Undo Management: Configuration

- **Configure two parameters in the initialization file:**
  - `UNDO_MANAGEMENT`
  - `UNDO_TABLESPACE`
- **Create at least one UNDO tablespace.**



**Initialization  
file**



**UNDO Tablespace**



# Automatic Undo Management: Initialization Parameters

- **UNDO\_MANAGEMENT:** Specifies whether the system should use **AUTO** or **MANUAL** mode.
- **UNDO\_TABLESPACE:** Specifies a particular **UNDO** tablespace to be used.

```
UNDO_MANAGEMENT=AUTO  
UNDO_TABLESPACE=UNDOTBS
```

# Automatic Undo Management: UNDO Tablespace

- You may create the UNDO tablespace with the database by adding a clause in the **CREATE DATABASE** command

```
CREATE DATABASE db01  
.  
.  
.  
UNDO TABLESPACE undo1 DATAFILE 'undo1db01.dbf'  
SIZE 20M AUTOEXTEND ON
```

- Or create it later by using the **CREATE UNDO TABLESPACE** command

```
CREATE UNDO TABLESPACE undo1  
DATAFILE 'undo1db01.dbf' SIZE 20M;
```

# Automatic Undo Management: Altering an UNDO Tablespace

- The **ALTER TABLESPACE** command can make changes to UNDO tablespaces.
- The following example adds another data file to the UNDO tablespace:

```
ALTER TABLESPACE undotbs  
ADD DATAFILE 'undotbs2.dbf' SIZE 30M  
AUTOEXTEND ON;
```

# Automatic Undo Management: Switching UNDO Tablespaces

- You may switch from using one UNDO tablespace to another
- Only one UNDO tablespace can be assigned to an instance at a time
- More than one UNDO tablespace may exist within an instance, but only one can be active
- Use the `ALTER SYSTEM` command for dynamic switching between UNDO tablespaces

```
ALTER SYSTEM SET UNDO_TABLESPACE=UNDOTBS2;
```

# Automatic Undo Management: Dropping an UNDO Tablespace

- The **DROP TABLESPACE** command drops an UNDO tablespace.

```
DROP TABLESPACE UNDOTBS2;
```

- An UNDO tablespace can only be dropped if it is currently not in use by any instance.
- To drop an active UNDO tablespace:
  - Switch to a new UNDO tablespace
  - Drop the tablespace after all current transactions are complete

# Automatic Undo Management: Other Parameters

- **UNDO\_SUPPRESS\_ERRORS**  
Set to **TRUE**, this parameter suppresses errors while attempting to execute manual operations in **AUTO** mode.
- **UNDO\_RETENTION**  
Controls the amount of undo data to retain for consistent read

# Undo Data Statistics

```
SELECT end_time,begin_time,undoblks
FROM    v$undostat;
```

| END_TIME  |          | BEGIN_TIME |          | UNDO  |
|-----------|----------|------------|----------|-------|
| -----     |          | -----      |          | ----- |
| 22-JAN-01 | 13:44:18 | 22-JAN-01  | 13:43:04 | 19    |
| 22-JAN-01 | 13:43:04 | 22-JAN-01  | 13:33:04 | 1474  |
| 22-JAN-01 | 13:33:04 | 22-JAN-01  | 13:23:04 | 1347  |
| 22-JAN-01 | 13:23:04 | 22-JAN-01  | 13:13:04 | 1628  |
| 22-JAN-01 | 13:13:04 | 22-JAN-01  | 13:03:04 | 2249  |
| 22-JAN-01 | 13:03:04 | 22-JAN-01  | 12:53:04 | 1698  |
| 22-JAN-01 | 12:53:04 | 22-JAN-01  | 12:43:04 | 1433  |
| 22-JAN-01 | 12:43:04 | 22-JAN-01  | 12:33:04 | 1532  |
| 22-JAN-01 | 12:33:04 | 22-JAN-01  | 12:23:04 | 1075  |

# Automatic Undo Management: Sizing an UNDO Tablespace

Determining a size for the UNDO tablespace requires three pieces of information

- (UR) UNDO\_RETENTION in seconds
- (UPS) Number of undo data blocks generated per second
- (DBS) Overhead varies based on extent and file size (db\_block\_size)

$$\text{UndoSpace} = [\text{UR} * (\text{UPS} * \text{DBS})] + (\text{DBS} * 24)$$



# Automatic Undo Management

## Undo Quota

- Long transactions and improperly written transactions can consume valuable resources
- With undo quota users can be grouped and a maximum undo space limit can be assigned to the group
- `UNDO_POOL`, a Resource Manager directive, defines the amount of space allowed for a resource group
- When a group exceeds its limit no new transactions are possible, for the group, until undo space is freed by current transactions either completing or aborting

# Obtaining Undo Segments Information

- **Data Dictionary Views**
  - `DBA_ROLLBACK_SEGS`
- **Dynamic Performance Views**
  - `V$ROLLNAME`
  - `V$ROLLSTAT`
  - `V$UNDOSTAT`
  - `V$SESSION`
  - `V$TRANSACTION`

# Summary

**In this lesson, you should have learned how to:**

- **Configure Automatic Undo Management**
- **Create an Undo Tablespace**
- **Properly size an Undo Tablespace**

# Practice 10 Overview

**This practice covers the following topics:**

- **Creating an UNDO tablespace**
- **Switching between UNDO tablespaces**
- **Dropping an UNDO tablespace**

# 11

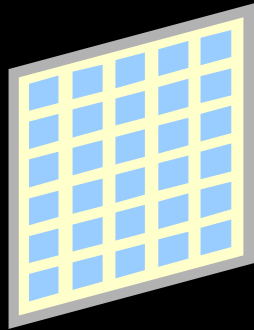
## Managing Tables

# Objectives

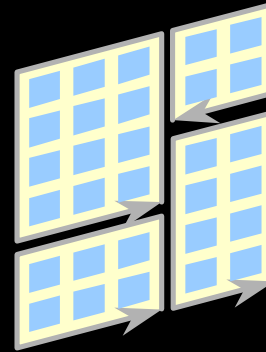
**After completing this lesson, you should be able to do the following:**

- **Identify the various methods of storing data**
- **Outline Oracle data types**
- **Distinguish between an extended versus a restricted ROWID**
- **Outline the structure of a row**
- **Create regular and temporary tables**
- **Manage storage structures within a table**
- **Reorganize, truncate, drop a table**
- **Drop a column within a table**

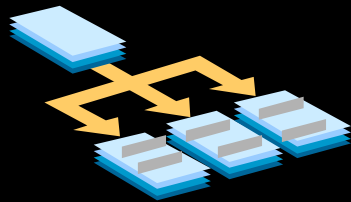
# Storing User Data



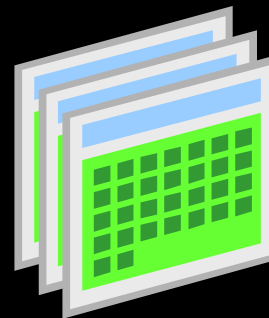
**Regular table**



**Partitioned  
table**

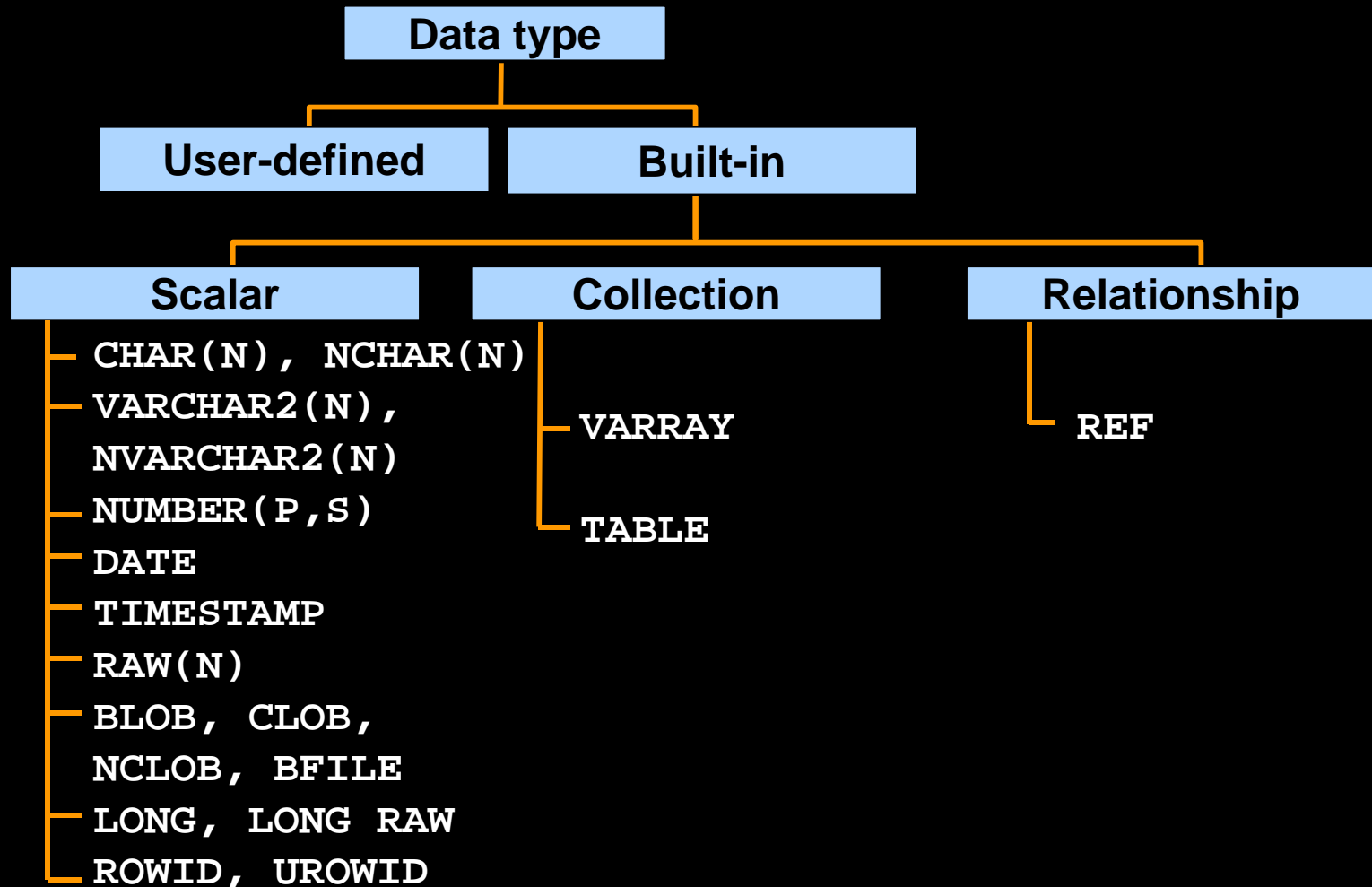


**Index-organized  
table**



**Cluster**

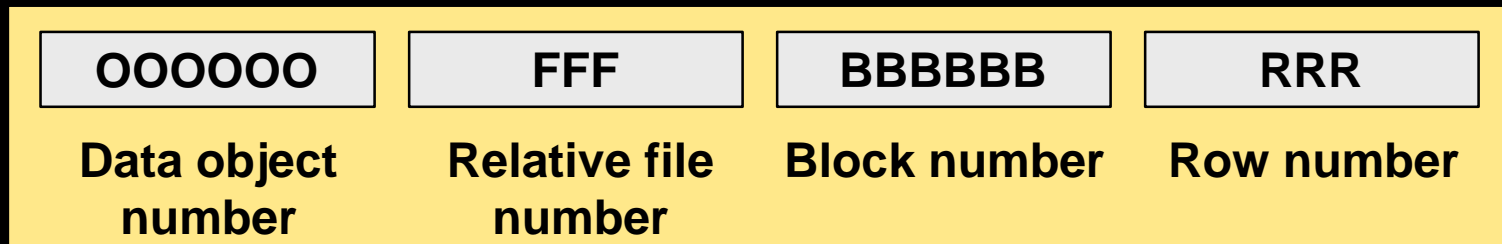
# Oracle Data Types



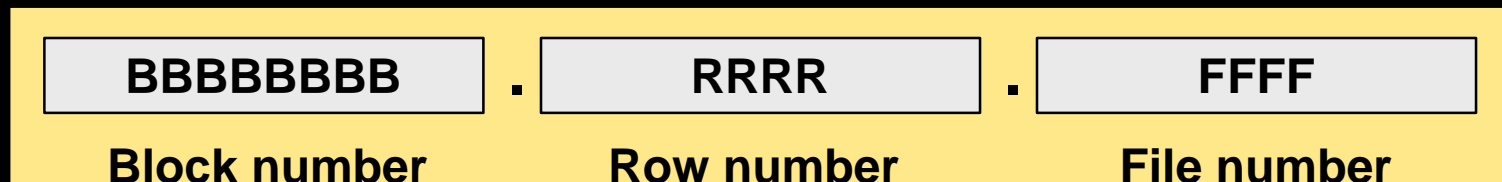


# ROWID Format

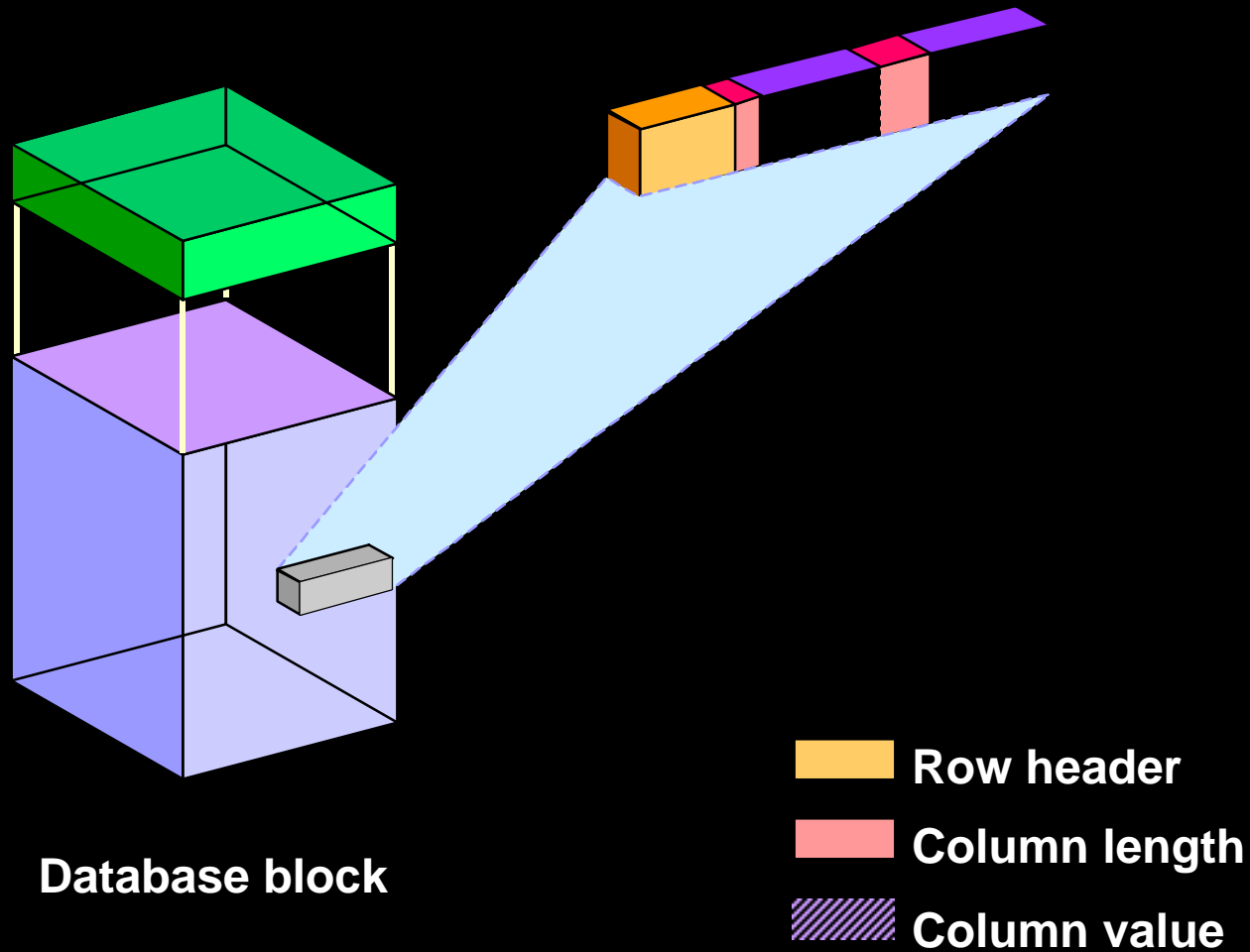
## Extended ROWID Format



## Restricted ROWID Format



# Structure of a Row



# Creating a Table

```
CREATE TABLE hr.employees(  
    employee_id  NUMBER(6),  
    first_name   VARCHAR2(20),  
    last_name    VARCHAR2(25)  
    email        VARCHAR2(25)  
    phone_number VARCHAR2(20)  
    hire_date    DATE DEFAULT SYSDATE  
    job_id       VARCHAR2(10)  
    salary       NUMBER(8,2)  
    commission_pct  NUMBER (2,2)  
    manager_id   NUMBER(6)  
    department_id NUMBER(4)  
);
```

# Creating Temporary Tables

- Created using the GLOBAL TEMPORARY clause

```
CREATE GLOBAL TEMPORARY TABLE  
hr.employees_temp  
AS SELECT * FROM hr.employees;
```

- Tables retain data only for the duration of a transaction or session
- DML locks are not acquired on the data
- DMLs do not generate redo logs
- Can create indexes, views, and triggers on temporary tables

# Creating a Table: Guidelines

- **Place tables in separate tablespaces.**
- **Use locally-managed tablespaces to avoid fragmentation.**
- **Use few standard extent sizes for tables to reduce tablespace fragmentation.**

# Changing Storage Parameters

```
ALTER TABLE hr.employees  
PCTFREE 30  
PCTUSED 50  
STORAGE(NEXT 500K  
MINEXTENTS 2  
MAXEXTENTS 100);
```

# Manually Allocating Extents

```
ALTER TABLE hr.employees  
ALLOCATE EXTENT(SIZE 500K  
DATAFILE '/DISK3/DATA01.DBF' );
```

# Nonpartitioned Table Reorganization

```
ALTER TABLE hr.employees  
MOVE TABLESPACE data1;
```

- Moves data into a new segment while preserving indexes, constraints, privileges, and so on the table
- Is being used to move a table to a different tablespace or reorganize extents



# Truncating a Table

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

- **Truncating a table deletes all rows in a table and releases used space.**
- **Corresponding indexes are truncated.**

# Dropping a Table

```
DROP TABLE hr.department  
CASCADE CONSTRAINTS;
```

# Dropping a Column

## Removing a column from a table:

```
ALTER TABLE hr.employees  
DROP COLUMN comments  
CASCADE CONSTRAINTS CHECKPOINT 1000;
```

- Removes the column length and data from each row, freeing space in the data block
- Dropping a column in a large table takes a considerable amount of time

# Using the UNUSED Option

- **Mark a column as unused**

```
ALTER TABLE hr.employees  
SET UNUSED COLUMN comments CASCADE CONSTRAINTS;
```

- **Drop unused columns**

```
ALTER TABLE hr.employees  
DROP UNUSED COLUMNS CHECKPOINT 1000;
```

- **Continue to drop column operation**

```
ALTER TABLE hr.employees  
DROP COLUMNS CONTINUE CHECKPOINT 1000;
```

# Obtaining Table Information

Information about tables can be obtained by querying the data dictionary.

- `DBA_TABLES`
- `DBA_OBJECTS`

# Summary

**In this lesson, you should have learned how to:**

- **Distinguish between an extended versus a restricted ROWID**
- **Outline the structure of a row**
- **Create regular and temporary tables**
- **Manage storage structures within a table**
- **Reorganize, truncate, drop a table**
- **Drop a column within a table**

# Practice 11 Overview

**This practice covers the following topics:**

- **Creating a table**
- **View, mark as unused, and drop columns within a table**
- **Allocate extents manually**
- **Truncate a table**
- **Obtaining table information**

# 12

## Managing Indexes



# Objectives

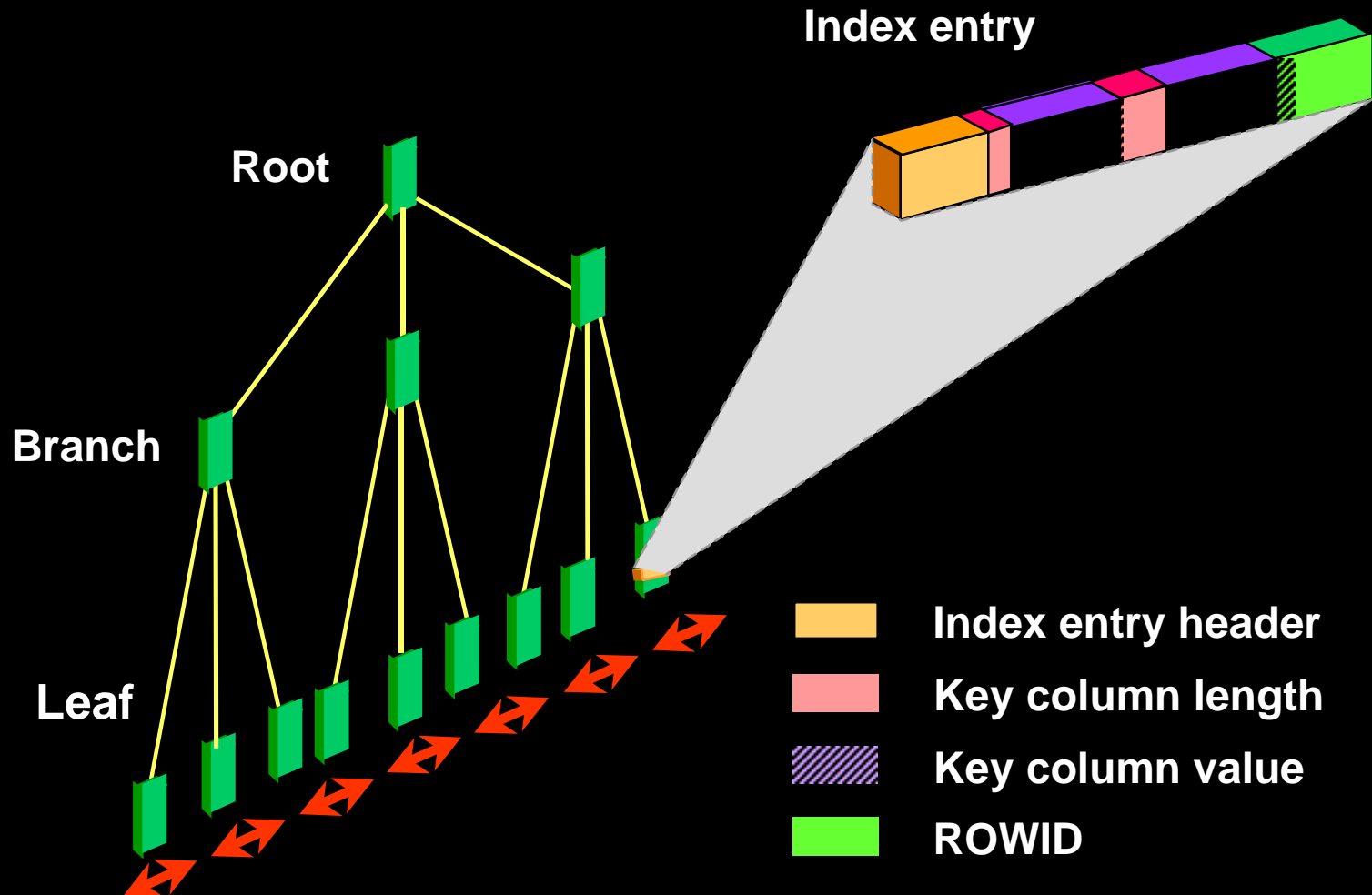
**After completing this lesson, you should be able to do the following:**

- **List the different types of indexes and their uses**
- **Create various types of indexes**
- **Reorganize indexes**
- **Drop indexes**
- **Get index information from the data dictionary**
- **Monitor the usage of an index**

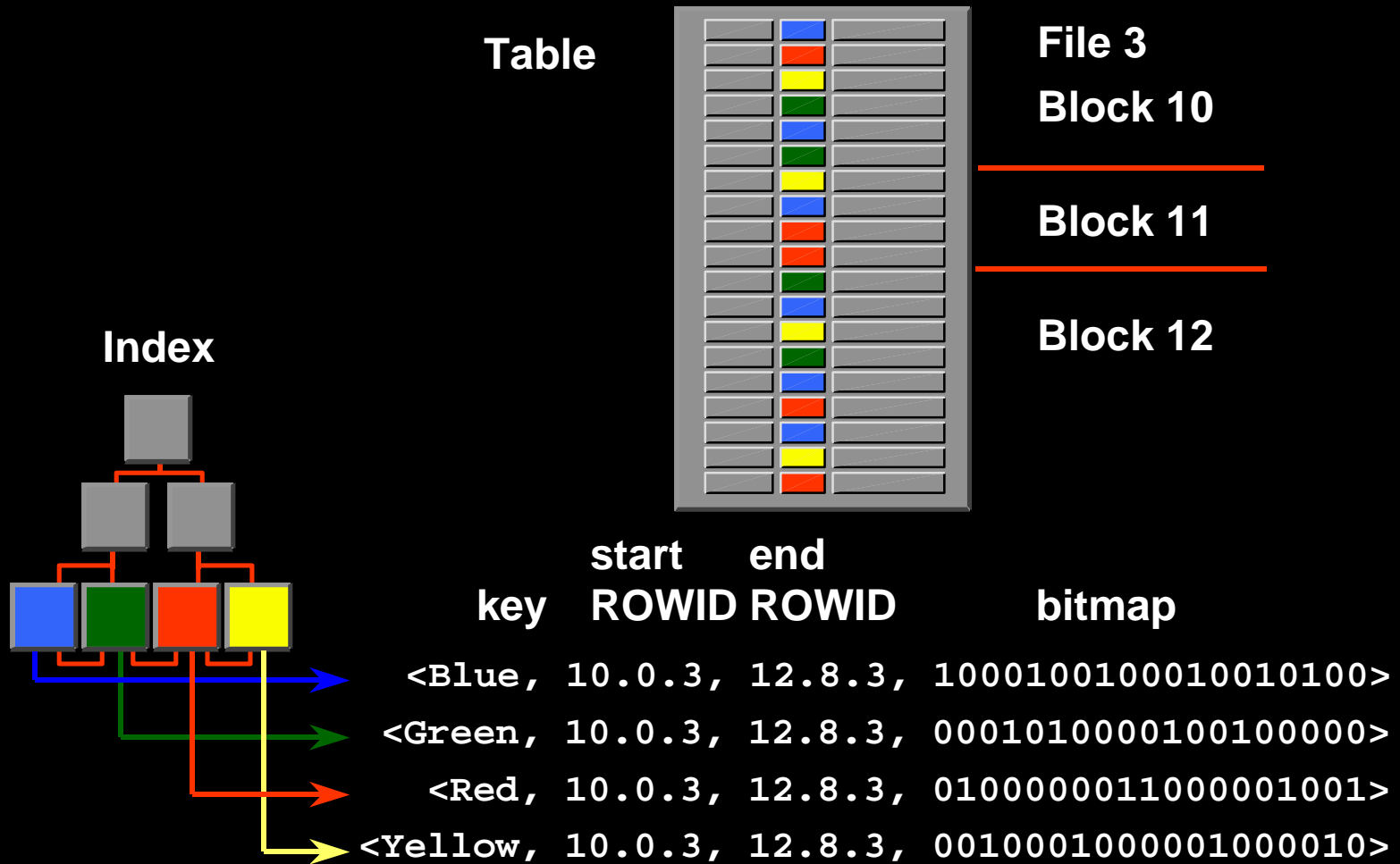
# Classification of Indexes

- **Logical**
  - **Single column or concatenated**
  - **Unique or nonunique**
  - **Function-based**
  - **Domain**
- **Physical**
  - **Partitioned or nonpartitioned**
  - **B-tree**
    - **Normal or reverse key**
  - **Bitmap**

# B-Tree Index



# Bitmap Index



# Comparing B-Tree and Bitmap Indexes

| <b>B-tree</b>                                      | <b>Bitmap</b>                                    |
|--|--|
| <b>Suitable for high-cardinality columns</b>       | <b>Suitable for low-cardinality columns</b>      |
| <b>Updates on keys relatively inexpensive</b>      | <b>Updates to key columns very expensive</b>     |
| <b>Inefficient for queries using OR predicates</b> | <b>Efficient for queries using OR predicates</b> |
| <b>Useful for OLTP</b>                             | <b>Useful for data warehousing</b>               |

# Creating Normal B-Tree Indexes

```
CREATE INDEX hr.employees_last_name_idx  
ON hr.employees(last_name)  
PCTFREE 30  
STORAGE(INITIAL 200K NEXT 200K  
PCTINCREASE 0 MAXEXTENTS 50)  
TABLESPACE indx;
```

# Creating Indexes: Guidelines

- **Balance query and DML needs**
- **Place in separate tablespace**
- **Use uniform extent sizes: Multiples of five blocks or `MINIMUM EXTENT` size for tablespace**
- **Consider `NOLOGGING` for large indexes**
- **`INITTRANS` should generally be higher on indexes than on the corresponding tables.**

# Creating Bitmap Indexes

Use the parameter **CREATE\_BITMAP\_AREA\_SIZE** to specify the amount of memory allocated for bitmap creation.

```
CREATE BITMAP INDEX orders_region_id_idx
ON orders(region_id)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K
PCTINCREASE 0    MAXEXTENTS 50)
TABLESPACE indx;
```



# Changing Storage Parameters for Indexes

```
ALTER INDEX employees_last_name_idx  
STORAGE(NEXT 400K  
MAXEXTENTS 100);
```

# Allocating and Deallocating Index Space

```
ALTER INDEX orders_region_id_idx  
ALLOCATE EXTENT (SIZE 200K  
DATAFILE '/DISK6/indx01.dbf');
```

```
ALTER INDEX orders_id_idx  
DEALLOCATE UNUSED;
```

# Rebuilding Indexes

Use the ALTER INDEX command to:

- Move an index to a different tablespace
- Improve space utilization by removing deleted entries
- Change a reverse key index to a normal B-tree index and vice versa

```
ALTER INDEX orders_region_id_idx REBUILD  
TABLESPACE indx02;
```

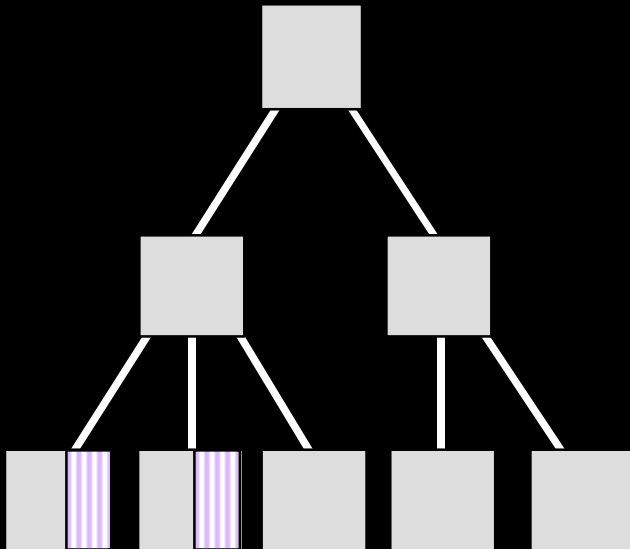
# Online Rebuild of Indexes

- Rebuilding indexes can be done with minimal table locking

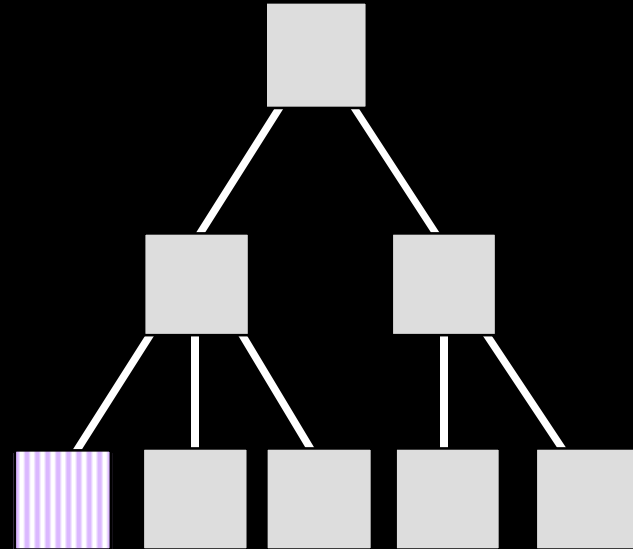
```
ALTER INDEX orders_id_idx REBUILD ONLINE;
```

- Some restrictions still apply

# Coalescing Indexes



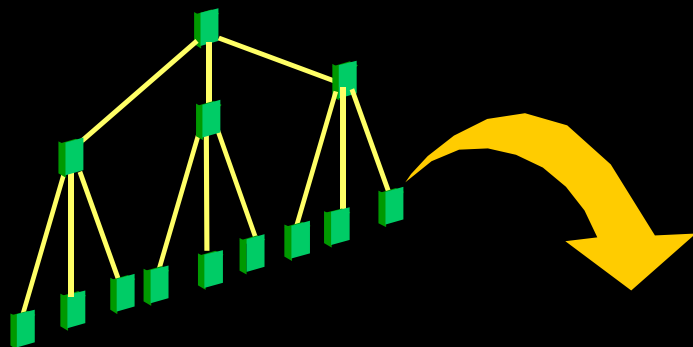
**Before coalescing**



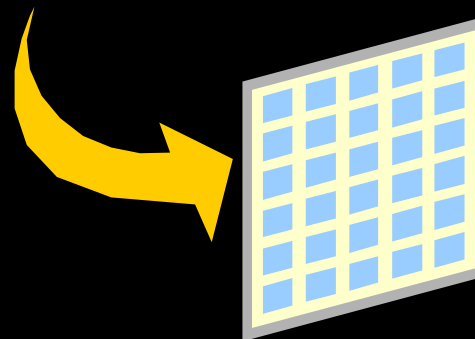
**After coalescing**

```
ALTER INDEX orders_id_idx COALESCE;
```

# Checking Index Validity



```
ANALYZE INDEX orders_region_id_idx  
VALIDATE STRUCTURE;
```



**INDEX\_STATS**

# Dropping Indexes

- Drop and re-create an index before bulk loads.
- Drop indexes that are infrequently needed and build them when necessary.
- Drop and re-create invalid indexes.

```
DROP INDEX hr.departments_name_idx;
```

# Identifying Unused Indexes

- **To start monitoring the usage of an index**

```
ALTER INDEX summit.orders_id_idx  
MONITORING USAGE
```

- **To stop monitoring the usage of an index**

```
ALTER INDEX summit.orders_id_idx  
NOMONITORING USAGE
```



# Obtaining Index Information

Information about indexes can be obtained by querying the data dictionary.

- **DBA\_INDEXES:** Provides information on the indexes
- **DBA\_IND\_COLUMNS:** Provides information on the columns indexed
- **DBA\_IND\_EXPRESSIONS:** Provides information on function based indexes
- **V\$OBJECT\_USAGE:** Provides information on the usage of an index

# Summary

**In this lesson, you should have learned how to:**

- **Create different types of indexes**
- **Reorganize indexes**
- **Drop indexes**
- **Get index information from the data dictionary**
- **Begin and end monitoring usage of indexes**

# Practice 12 Overview

**This practice covers the following topics:**

- **Creating an index on columns of a table**
- **Moving the index to another tablespace**
- **Dropping an index**
- **Obtain index information**

# 13

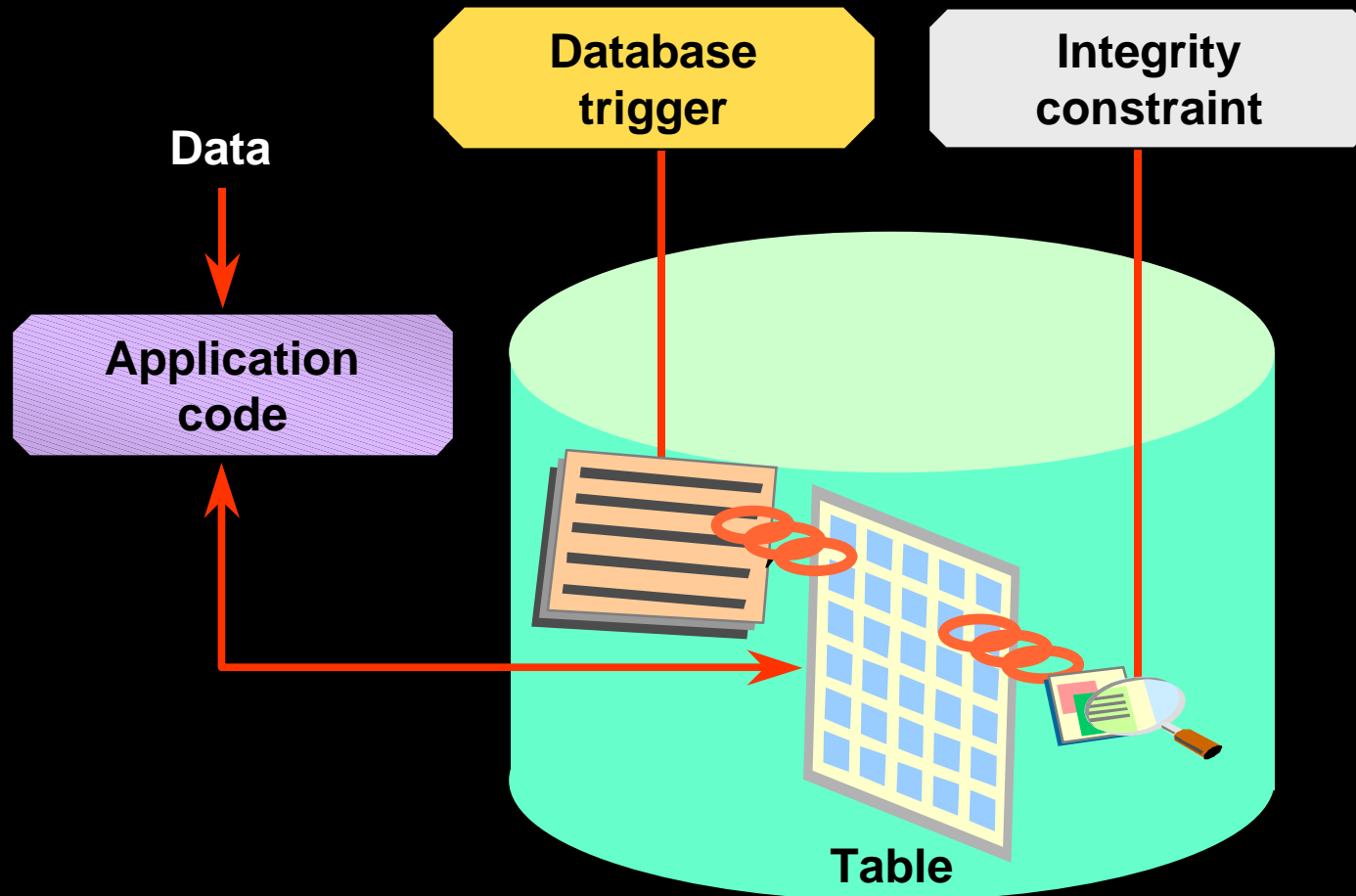
## Maintaining Data Integrity

# Objectives

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

- **Implement data integrity constraints**
- **Maintain integrity constraints**
- **Obtain constraint information from the data dictionary**

# Data Integrity



# Types of Constraints

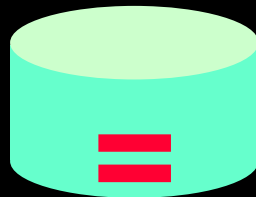
| Constraint  | Description  |
|-------------|--|
| NOT NULL    | Specifies that a column cannot contain null values   |
| UNIQUE      | Designates a column or combination of columns as unique  |
| PRIMARY KEY | Designates a column or combination of columns as the table's primary key                               |
| FOREIGN KEY | Designates a column or combination of columns as the foreign key in a referential integrity constraint |
| CHECK       | Specifies a condition that each row of the table must satisfy  |

# Constraint States

**DISABLE  
NOVALIDATE**



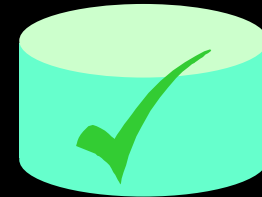
**DISABLE  
VALIDATE**



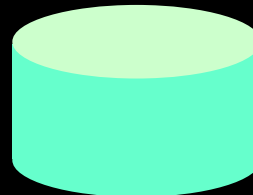
**ENABLE  
NOVALIDATE**



**ENABLE  
VALIDATE**



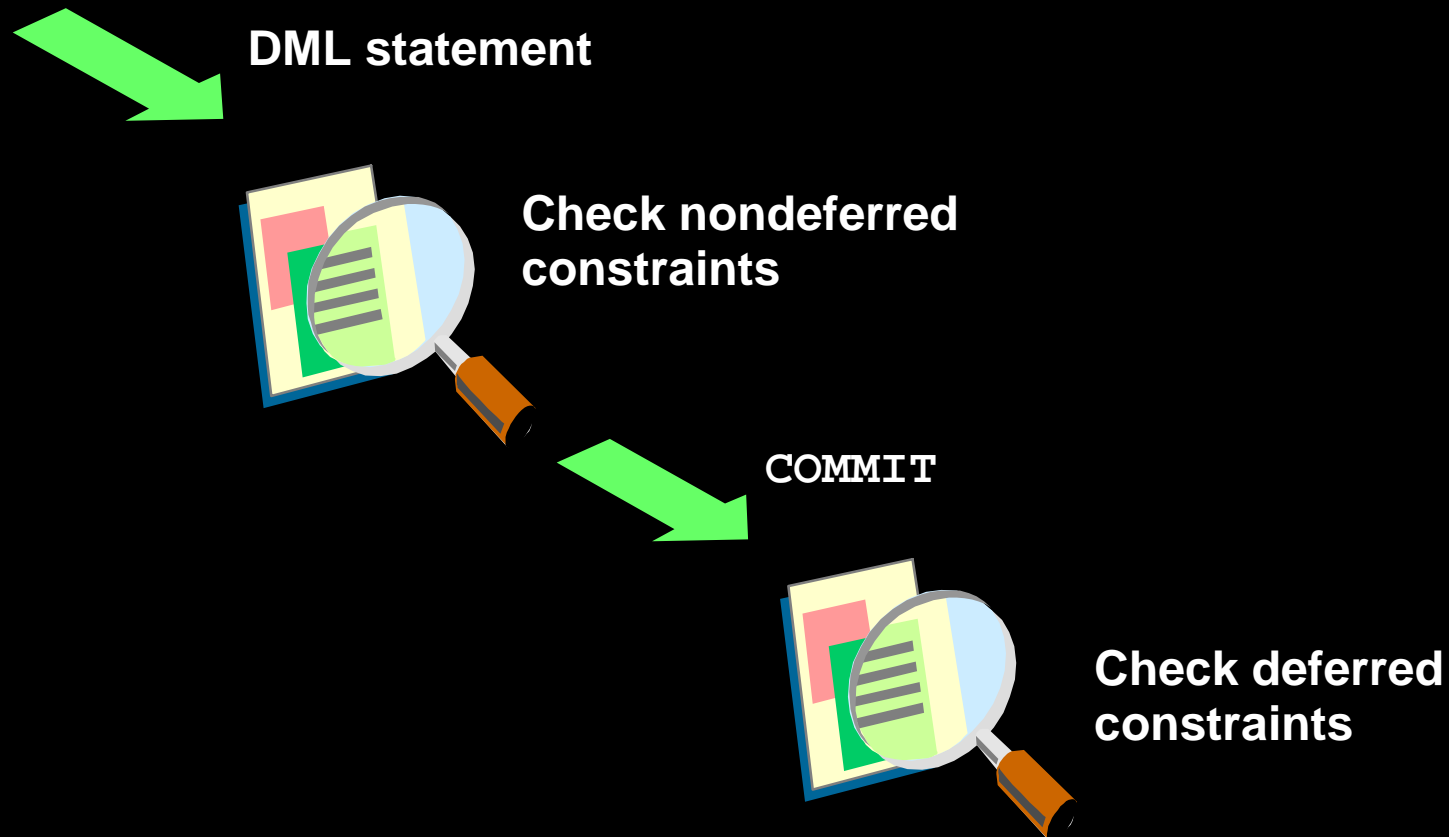
**New data**



**Existing data**



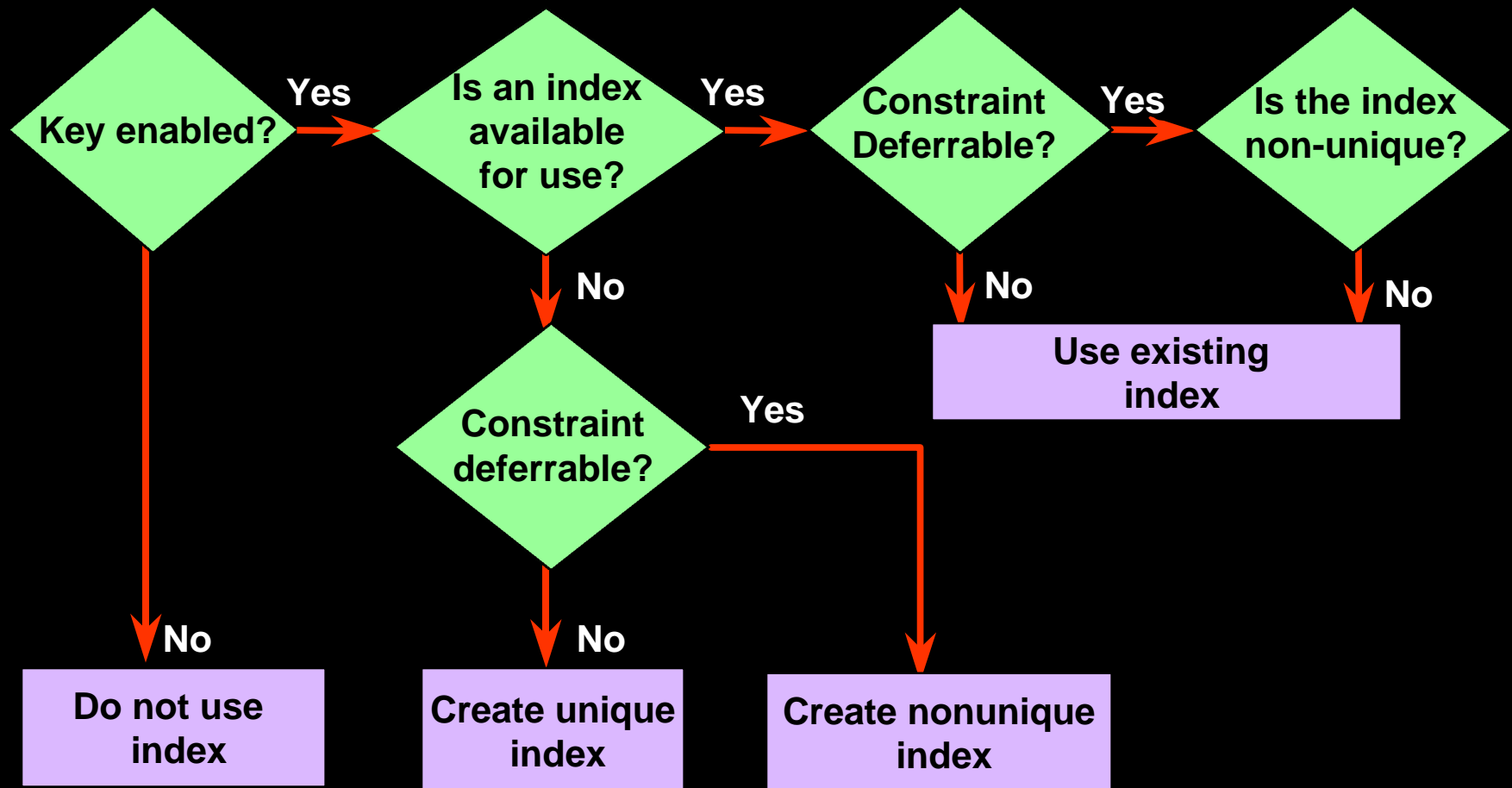
# Constraint Checking



# Defining Constraints as Immediate or Deferred

- Use the **SET CONSTRAINTS** statement to make constraints either **DEFERRED** or **IMMEDIATE**
- The **ALTER SESSION** statement also has clauses to **SET CONSTRAINTS** to **DEFERRED** or **IMMEDIATE**

# Primary and Unique Key Enforcement



# Foreign Key Considerations

| Desired Action                          | Appropriate Solution                                      |
|---|---|
| Drop parent table                       | Cascade constraints                                       |
| Truncate parent table                   | Disable or drop foreign key                               |
| Drop tablespace containing parent table | Use the <code>CASCADE CONSTRAINTS</code> clause           |
| Perform DML on child table              | Ensure the tablespace containing the parent key is online |

# Defining Constraints While Creating a Table

```
CREATE TABLE hr.employee(  
  id NUMBER(7)  
    CONSTRAINT employee_id_pk PRIMARY KEY  
    DEFERRABLE  
    USING INDEX  
    STORAGE(INITIAL 100K NEXT 100K)  
    TABLESPACE indx,  
  last_name VARCHAR2(25)  
    CONSTRAINT employee_last_name_nn NOT NULL,  
  dept_id NUMBER(7))  
TABLESPACE users;
```

# Guidelines for Defining Constraints

- **Primary and unique constraints:**
  - **Place indexes in a separate tablespace**
  - **Use nonunique indexes if bulk loads are frequent**
- **Self-referencing foreign keys:**
  - **Define or enable foreign keys after initial load**
  - **Defer constraint checking**

# Enabling Constraints



**ENABLE  
NOVALIDATE**

- **No locks on table**
- **Primary and unique keys must use nonunique indexes**

```
ALTER TABLE hr.departments  
ENABLE NOVALIDATE CONSTRAINT dept_pk;
```

# Enabling Constraints



**ENABLE  
VALIDATE**

- **Locks table**
- **Can use unique or nonunique indexes**
- **Needs valid table data**

```
ALTER TABLE hr.employees  
ENABLE VALIDATE CONSTRAINT emp_dept_fk;
```



# Using the EXCEPTIONS Table

- Create the EXCEPTIONS table by running the `utlexcpt1.sql` script
- Execute the `ALTER TABLE` statement with `EXCEPTIONS` option
- Use subquery on `EXCEPTIONS` to locate rows with invalid data
- Rectify the errors
- Reexecute `ALTER TABLE` to enable the constraint.

# Obtaining Constraint Information

## Data Dictionary Views

- **DBA\_CONSTRAINTS**
- **DBA\_CONS\_COLUMNS**

# Summary

**In this lesson, you should have learned how to:**

- **Implement data integrity**
- **Use an appropriate strategy for creating and maintaining constraints**
- **Obtain information from the data dictionary**

# Practice 13 Overview

**This practice covers the following topics:**

- **Creating constraints**
- **Enabling unique constraints**
- **Creating an Exceptions table**
- **Identifying existing constraint violations in a table, correcting the errors and re-enabling the constraints**

# 14

## Managing Password Security and Resources

# Objectives

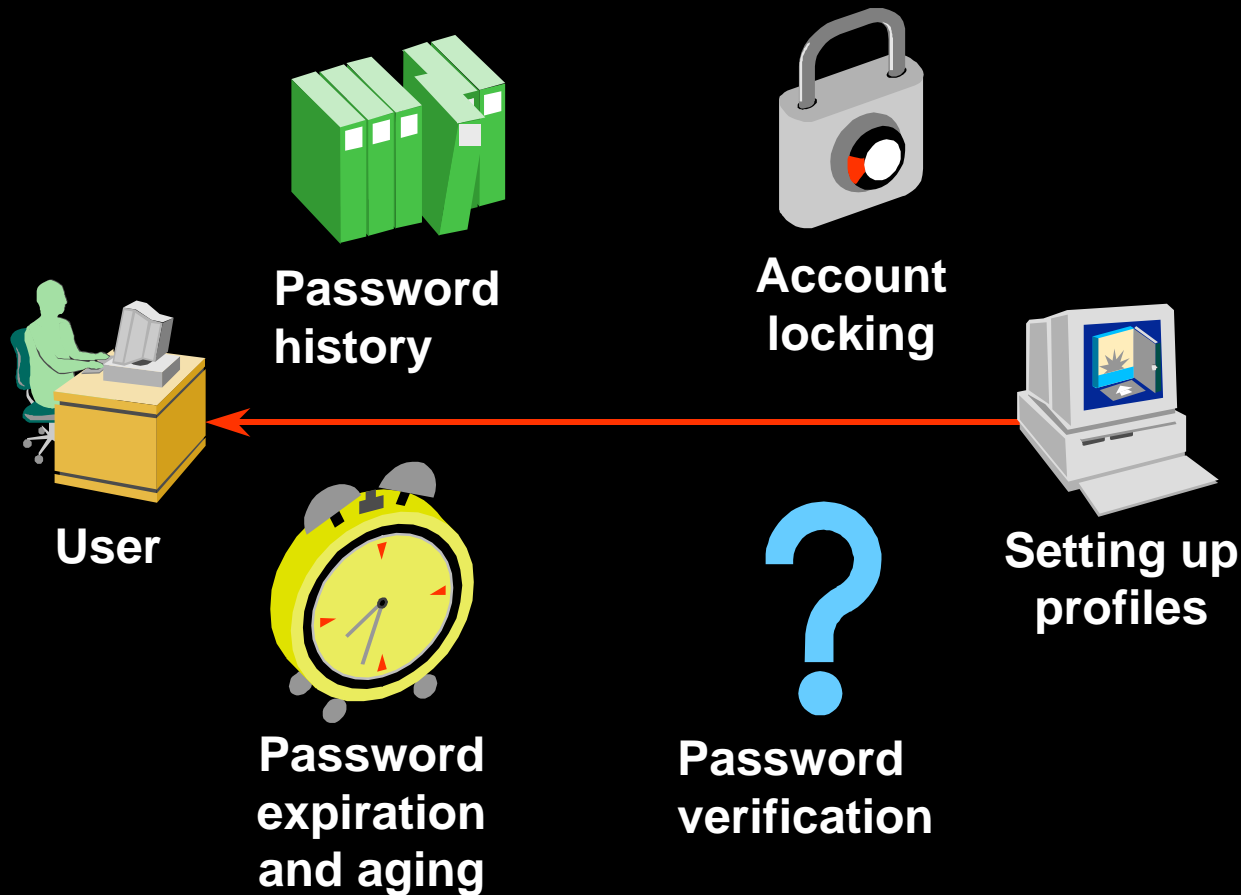
**After completing this lesson, you should be able to do the following:**

- **Manage passwords using profiles**
- **Administer profiles**
- **Control use of resources using profiles**
- **Obtain information about profiles, password management, and resources**

# Profiles

- A profile is a named set of password and resource limits
- Profiles are assigned to users by the `CREATE USER` or `ALTER USER` command
- Can be enabled or disabled
- Can relate to the `DEFAULT` profile

# Password Management



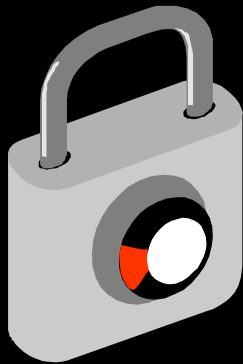


# Enabling Password Management

- Set up password management by using profiles and assign them to users.
- Lock, unlock, and expire accounts using the `CREATE USER` or `ALTER USER` command.
- Password limits are always enforced.

# Password Account Locking

| Parameter                    | Description  |
|------------------------------|--|
| <b>FAILED_LOGIN_ATTEMPTS</b> | Number of failed login attempts before lockout of the account                            |
| <b>PASSWORD_LOCK_TIME</b>    | Number of days the account is locked after the specified number of failed login attempts |



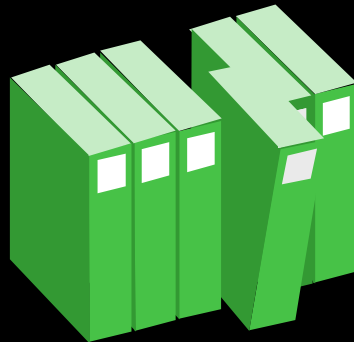
# Password Expiration and Aging

| Parameter                  | Parameter  |
|----------------------------|--|
| <b>PASSWORD_LIFE_TIME</b>  | Lifetime of the password in days after which the password expires  |
| <b>PASSWORD_GRACE_TIME</b> | Grace period in days for changing the password after the first successful login after the password has expired |



# Password History

| Parameter                  | Description                                      |
|----------------------------|--|
| <b>PASSWORD_REUSE_TIME</b> | Number of days before a password can be reused   |
| <b>PASSWORD_REUSE_MAX</b>  | Maximum number of times a password can be reused |



# Password Verification

| Parameter                       | Description  |
|---------------------------------|--|
| <b>PASSWORD_VERIFY_FUNCTION</b> | PL/SQL function that makes a password complexity check before a password is assigned |

# User-Provided Password Function

Function must be created in the `sys` schema and must have the following specification:

```
function_name(  
    userid_parameter IN VARCHAR2(30),  
    password_parameter IN VARCHAR2(30),  
    old_password_parameter IN VARCHAR2(30))  
RETURN BOOLEAN
```

# **Password Verification Function**

## **VERIFY\_FUNCTION**

- **Minimum length is four characters.**
- **Password should not be equal to username.**
- **Password should have at least one alphabetic, one numeric, and one special character.**
- **Password should differ from the previous password by at least three letters.**

# Creating a Profile: Password Settings

```
CREATE PROFILE grace_5 LIMIT  
  FAILED_LOGIN_ATTEMPTS 3  
  PASSWORD_LOCK_TIME UNLIMITED  
  PASSWORD_LIFE_TIME 30  
  PASSWORD_REUSE_TIME 30  
  PASSWORD_VERIFY_FUNCTION verify_function  
  PASSWORD_GRACE_TIME 5;
```



# Altering a Profile: Password Setting

```
ALTER PROFILE default  
FAILED_LOGIN_ATTEMPTS 3  
PASSWORD_LIFE_TIME 60  
PASSWORD_GRACE_TIME 10;
```

# Dropping a Profile: Password Setting

```
DROP PROFILE developer_prof;
```

```
DROP PROFILE developer_prof CASCADE;
```

# Resource Management

- Resource management limits can be enforced at the session level, the call level, or both.
- Limits can be defined by profiles using the `CREATE PROFILE` command.
- Enable resource limits with the:
  - `RESOURCE_LIMIT` initialization parameter
  - `ALTER SYSTEM` command

# Enabling Resource Limits

- Set the initialization parameter `RESOURCE_LIMIT` to `TRUE`
- Enforce the resource limits by enabling the parameter with the `ALTER SYSTEM` command

```
ALTER SYSTEM SET RESOURCE_LIMIT=TRUE;
```

# Setting Resource Limits at Session Level

| Resource                         | Description   |
|----------------------------------|---|
| <b>CPU_PER_SESSION</b>           | Total CPU time measured in hundredths of seconds                    |
| <b>SESSIONS_PER_USER</b>         | Number of concurrent sessions allowed for each username             |
| <b>CONNECT_TIME</b>              | Elapsed connect time measured in minutes                            |
| <b>IDLE_TIME</b>                 | Periods of inactive time measured in minutes                        |
| <b>LOGICAL_READS_PER_SESSION</b> | Number of data blocks (physical and logical reads)                  |
| <b>PRIVATE_SGA</b>               | Private space in the SGA measured in bytes (for Shared Server only) |

# Setting Resource Limits at Call Level

| Resource                      | Description                                     |
|-------------------------------|---|
| <b>CPU_PER_CALL</b>           | CPU time per call in hundredths of seconds      |
| <b>LOGICAL_READS_PER_CALL</b> | Number of data blocks that can be read per call |

# Creating a Profile: Resource Limit

```
CREATE PROFILE developer_prof LIMIT  
  SESSIONS_PER_USER 2  
  CPU_PER_SESSION 10000  
  IDLE_TIME 60  
  CONNECT_TIME 480;
```

# Managing Resources Using the Database Resource Manager

- Provides the Oracle server with more control over resource management decisions
- Elements of the Database Resource Manager
  - Resource consumer group
  - Resource plan
  - Resource allocation method
  - Resource plan directives
- DBMS\_RESOURCE\_MANAGER package is used to create and maintain elements
- Requires ADMINISTER\_RESOURCE\_MANAGER privilege



# Managing Resources Using the Database Resource Manager

- Resource plans specify the resource consumer groups belonging to the plan.
- Resource plans contain directives for how resources are to be allocated among consumer groups.

# Resource Plan Directives

- **The Database Resource Manager provides several means of allocating resources.**
  - **CPU Method**
  - **Active Session Pool and Queuing**
  - **Degree of Parallelism Limit**
  - **Automatic Consumer Group Switching**
  - **Maximum Estimated Execution Time**
  - **Undo Pool**

# Obtaining Password and Resource Limits Information

Information about password and resource limits can be obtained by querying the data dictionary.

- `DBA_USERS`
- `DBA_PROFILES`

# Summary

**In this lesson, you should have learned how to:**

- **Administer passwords**
- **Administer profiles**

# Practice Overview

**This practice covers the following topics:**

- **Enabling password management**
- **Defining profiles and assigning to users**
- **Disabling password management**

# 15

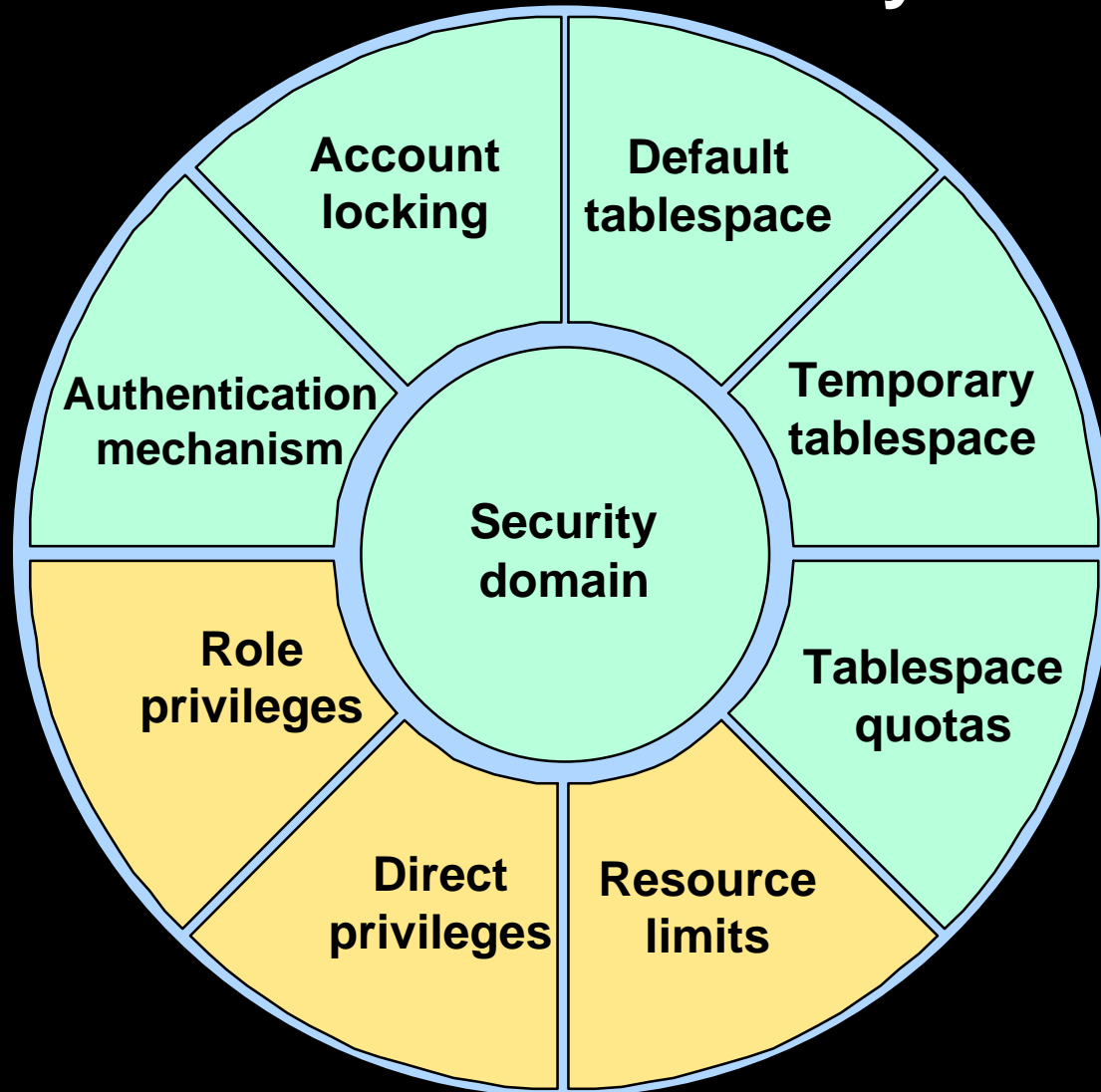
## Managing Users

# Objectives

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

- **Create new database users**
- **Alter and drop existing database users**
- **Monitor information about existing users**

# Users and Security





# Database Schema

- A schema is a named collection of objects
- A user is created, and a corresponding schema is created
- User can be associated only with one schema
- Username and schema are often used interchangeably

## Schema Objects

Tables

Triggers

Constraints

Indexes

Views

Sequences

Stored program units

Synonyms

User-defined data types

Database links

# Checklist for Creating Users

- **Identify tablespaces in which the user needs to store objects.**
- **Decide on quotas for each tablespace.**
- **Assign a default tablespace and temporary tablespace.**
- **Create a user.**
- **Grant privileges and roles to the user.**

# Creating a New User: Database Authentication

Set the initial password:

```
CREATE USER aaron  
IDENTIFIED BY soccer  
DEFAULT TABLESPACE data  
TEMPORARY TABLESPACE temp  
QUOTA 15m ON data  
PASSWORD EXPIRE;
```

# Creating a New User: Operating System Authentication

- **OS\_AUTHENT\_PREFIX** initialization parameter specifies the format of the usernames
- Defaults to OPS\$

```
CREATE USER aaron  
IDENTIFIED EXTERNALLY  
DEFAULT TABLESPACE USERS  
TEMPORARY TABLESPACE temp  
QUOTA 15m ON data  
PASSWORD EXPIRE;
```

# Changing User Quota on Tablespace

```
ALTER USER aaron  
QUOTA 0 ON USERS;
```

# Dropping a User

```
DROP USER aaron;
```

- Use the **CASCADE** clause to drop all objects in the schema if the schema contains objects.

```
DROP USER aaron CASCADE;
```

- Users currently connected to the Oracle server cannot be dropped

# Obtaining User Information

Information about users can be obtained by querying the data dictionary.

- `DBA_USERS`
- `DBA_TS_QUOTAS`

# Summary

**In this lesson, you should have learned how to:**

- **Create users specifying the appropriate password mechanism**
- **Control usage of space by users**



# Practice 15 Overview

**This practice covers the following topics:**

- **Creating users**
- **Displaying data dictionary information about users**
- **Removing user quotas**

# 16

## Managing Privileges

# Objectives

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

- **Identify system and object privileges**
- **Grant and revoke privileges**
- **Identify auditing capabilities**

# Managing Privileges

**Two types of Oracle user privileges:**

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

# System Privileges

- There are over 100 distinct system privileges
- The **ANY** keyword in the privileges signifies that users have the privilege in any schema
- The **GRANT** command adds a privilege to a user or a group of users
- The **REVOKE** command deletes the privileges

# System Privileges: Examples

| Category   | Examples  |
|------------|---|
| INDEX      | CREATE ANY INDEX<br>ALTER ANY INDEX<br>DROP ANY INDEX   |
| TABLE      | CREATE TABLE<br>CREATE ANY TABLE<br>ALTER ANY TABLE<br>DROP ANY TABLE<br>SELECT ANY TABLE<br>UPDATE ANY TABLE<br>DELETE ANY TABLE |
| SESSION    | CREATE SESSION<br>ALTER SESSION<br>RESTRICTED SESSION   |
| TABLESPACE | CREATE TABLESPACE<br>ALTER TABLESPACE<br>DROP TABLESPACE<br>UNLIMITED TABLESPACE  |

# Granting System Privileges

```
GRANT CREATE SESSION TO emi;
```

```
GRANT CREATE SESSION TO emi WITH ADMIN OPTION;
```

# **SYSDBA and SYSOPER Privileges**

| <b>Category</b> | <b>Examples</b>   |
|-----------------|---|
| <b>SYSOPER</b>  | <b>STARTUP</b><br><b>SHUTDOWN</b><br><b>ALTER DATABASE OPEN   MOUNT</b><br><b>ALTER DATABASE BACKUP CONTROLFILE TO</b><br><b>RECOVER DATABASE</b><br><b>ALTER DATABASE ARCHIVELOG</b> |
| <b>SYSDBA</b>   | <b>SYSOPER PRIVILEGES WITH ADMIN OPTION</b><br><b>CREATE DATABASE</b><br><b>ALTER DATABASE BEGIN/END BACKUP</b><br><b>RESTRICTED SSESSEION</b><br><b>RECOVER DATABASE UNTIL</b>       |



# System Privilege Restrictions

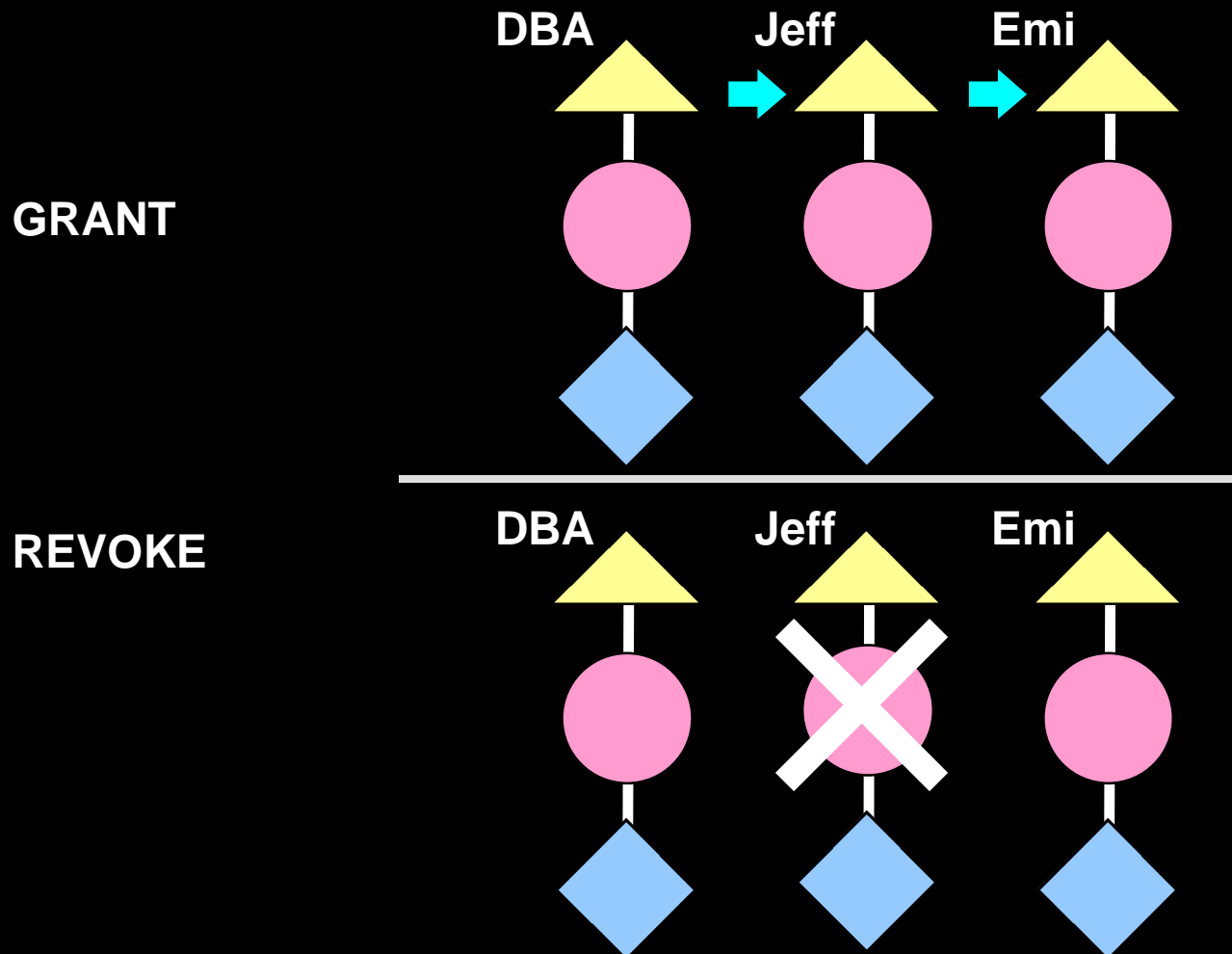
**O7\_DICTIONARY\_ACCESSIBILITY** parameter

- **Controls restrictions on SYSTEM privileges**
- **If set to TRUE, access to objects in SYS schema is allowed**
- **Default is FALSE**
  - **Ensures that system privileges that allow access to any schema do not allow access to SYS schema**

# Revoking System Privileges

```
REVOKE CREATE TABLE FROM emi;
```

# Revoking System Privileges WITH ADMIN OPTION



# Object Privileges

| Object priv. | Table | View | Sequence | Procedure |
|--------------|-------|------|----------|-----------|
| ALTER        | √     |      | √        | √         |
| DELETE       | √     | √    |          |           |
| EXECUTE      |       |      |          | √         |
| INDEX        | √     | √    |          |           |
| INSERT       | √     | √    |          |           |
| REFERENCES   | √     |      |          |           |
| SELECT       | √     | √    | √        |           |
| UPDATE       | √     | √    |          |           |

# Granting Object Privileges

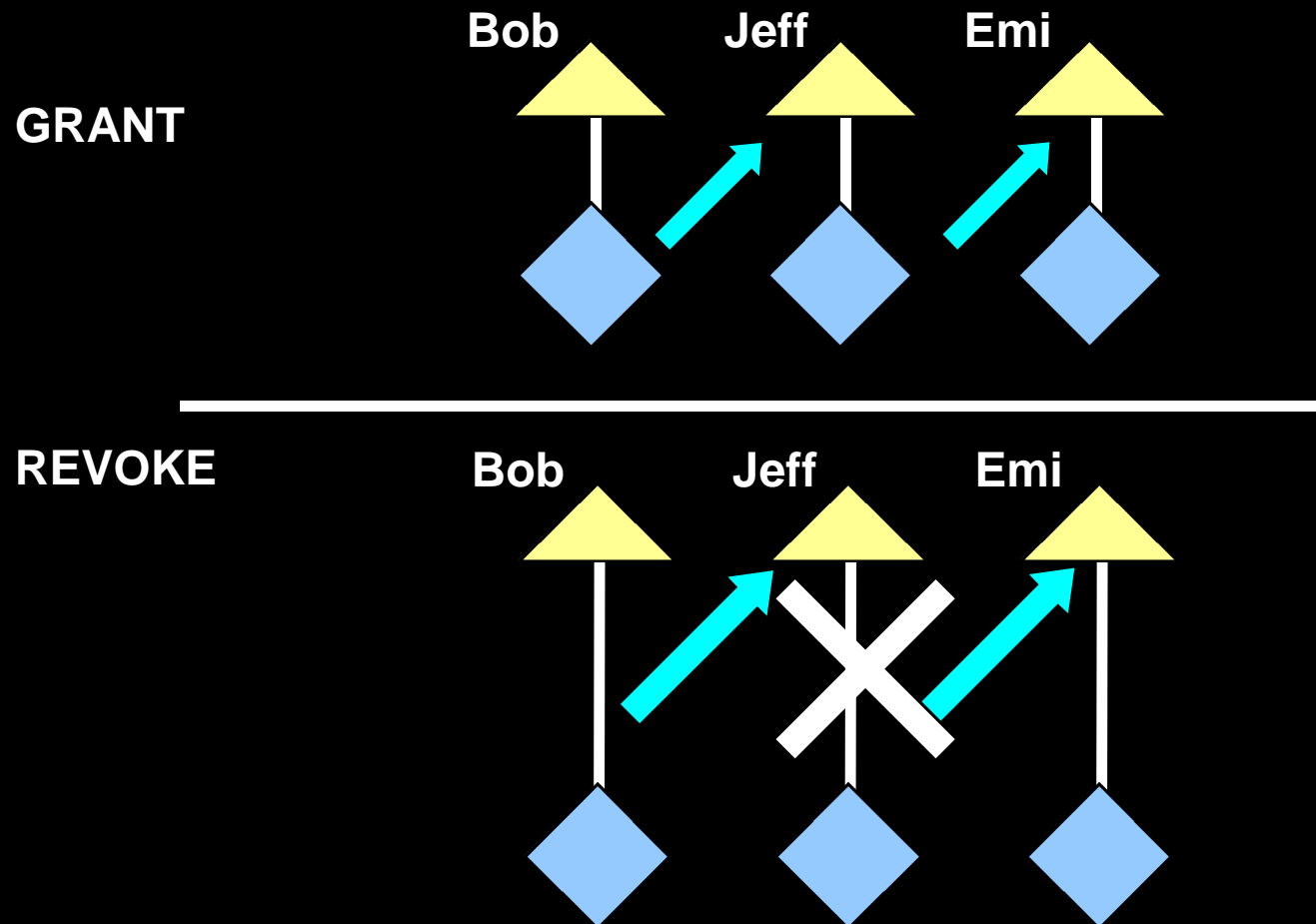
```
GRANT EXECUTE ON dbms_output TO jeff;
```

```
GRANT UPDATE ON emi.customers TO jeff WITH  
GRANT OPTION;
```

# Revoking Object Privileges

```
REVOKE SELECT ON emi.orders FROM jeff;
```

# Revoking Object Privileges WITH GRANT OPTION



# Obtaining Privileges Information

- **Data Dictionary Views**
  - `DBA_SYS_PRIVS`
  - `SESSION_PRIVS`
  - `DBA_TAB_PRIVS`
  - `DBA_COL_PRIVS`



# Auditing

- **Auditing is the monitoring of selected user database actions**
- **Used to**
  - **Investigate suspicious database activity**
  - **Gather information about specific database activities**

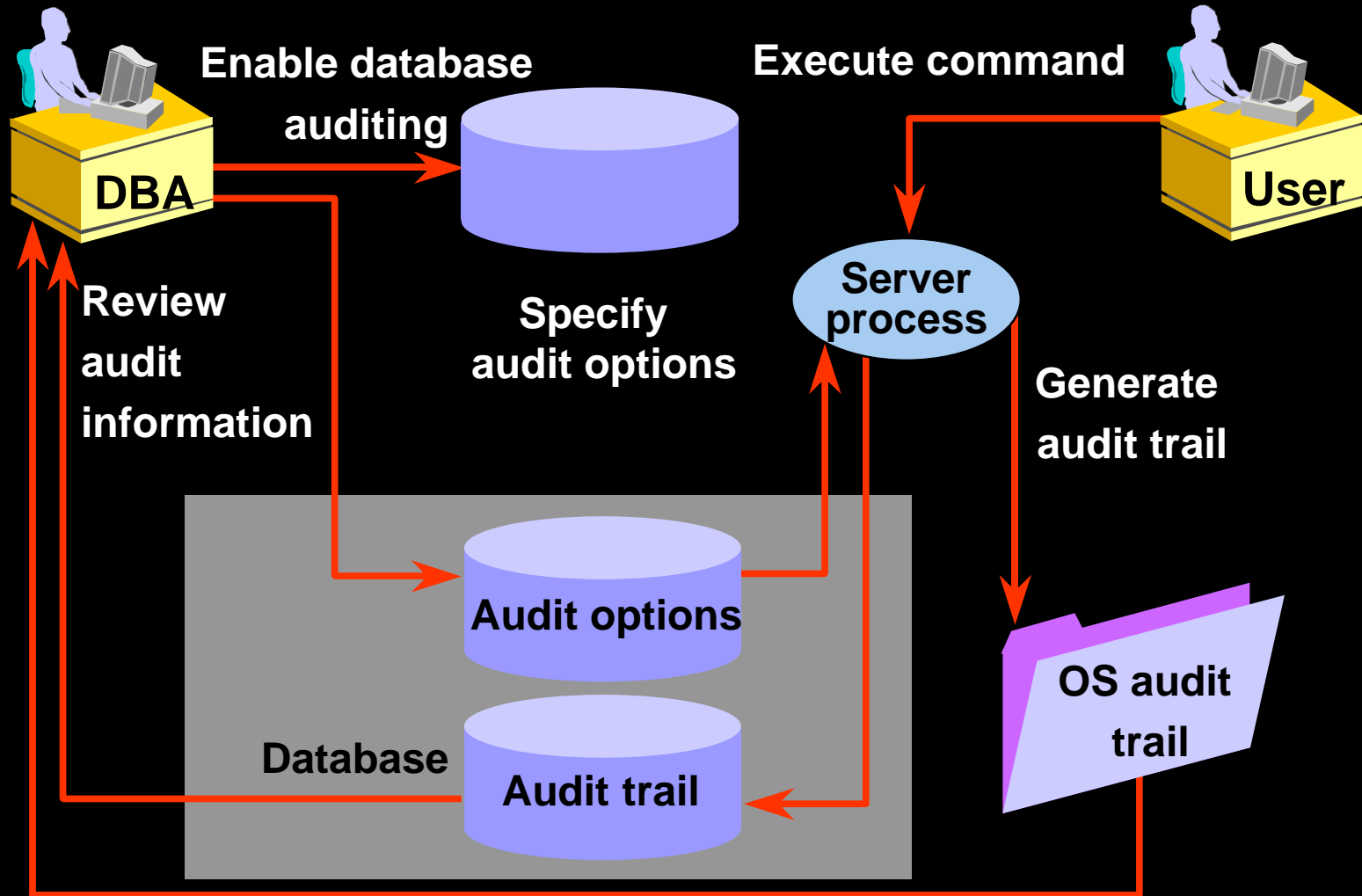
# Auditing Guidelines

- **Define what you want to audit**
  - **Audit users, statements, or objects**
  - **Statement executions**
  - **Successful statement executions, unsuccessful statement executions or both**
- **Manage your audit trail**
  - **Monitor the growth of the audit trail**
  - **Protect the audit trail from unauthorized access**

# Auditing Categories

- **Audited by default**
  - **Instance startup and Instance shutdown**
  - **Administrator privileges**
- **Database auditing**
  - **Enabled by DBA**
  - **Cannot record column values**
- **Value-based or application auditing**
  - **Implemented through code**
  - **Can record column values**
  - **Used to track changes to tables**

# Database Auditing



# Auditing Options

- **Statement auditing**

```
AUDIT TABLE;
```

- **Privilege auditing**

```
AUDIT create any trigger;
```

- **Schema object auditing**

```
AUDIT SELECT ON emi.orders;
```

# Auditing Options

## Fine-Grained Auditing

- Provides the monitoring of data access based on content
- Implemented using the `DBMS_FGA` package

# Viewing Auditing Options

## Data Dictionary Views

- ALL\_DEF\_AUDIT\_OPTS
- DBA\_STMT\_AUDIT\_OPTS
- DBA\_PRIV\_AUDIT\_OPTS
- DBA\_OBJ\_AUDIT\_OPTS

# Obtaining Audit Records

- **Data Dictionary Views**
  - DBA\_AUDIT\_TRAIL
  - DBA\_AUDIT\_EXISTS
  - DBA\_AUDIT\_OBJECT
  - DBA\_AUDIT\_SESSION
  - DBA\_AUDIT\_STATEMENT



# Summary

**In this lesson, you should have learned how to:**

- **Control system and object privileges**
- **Use database auditing**

# Practice 16 Overview

**This practice covers the following topics:**

- **Creating user and granting system privileges**
- **Granting Object privileges to users**
- **Enabling Auditing**

# 17

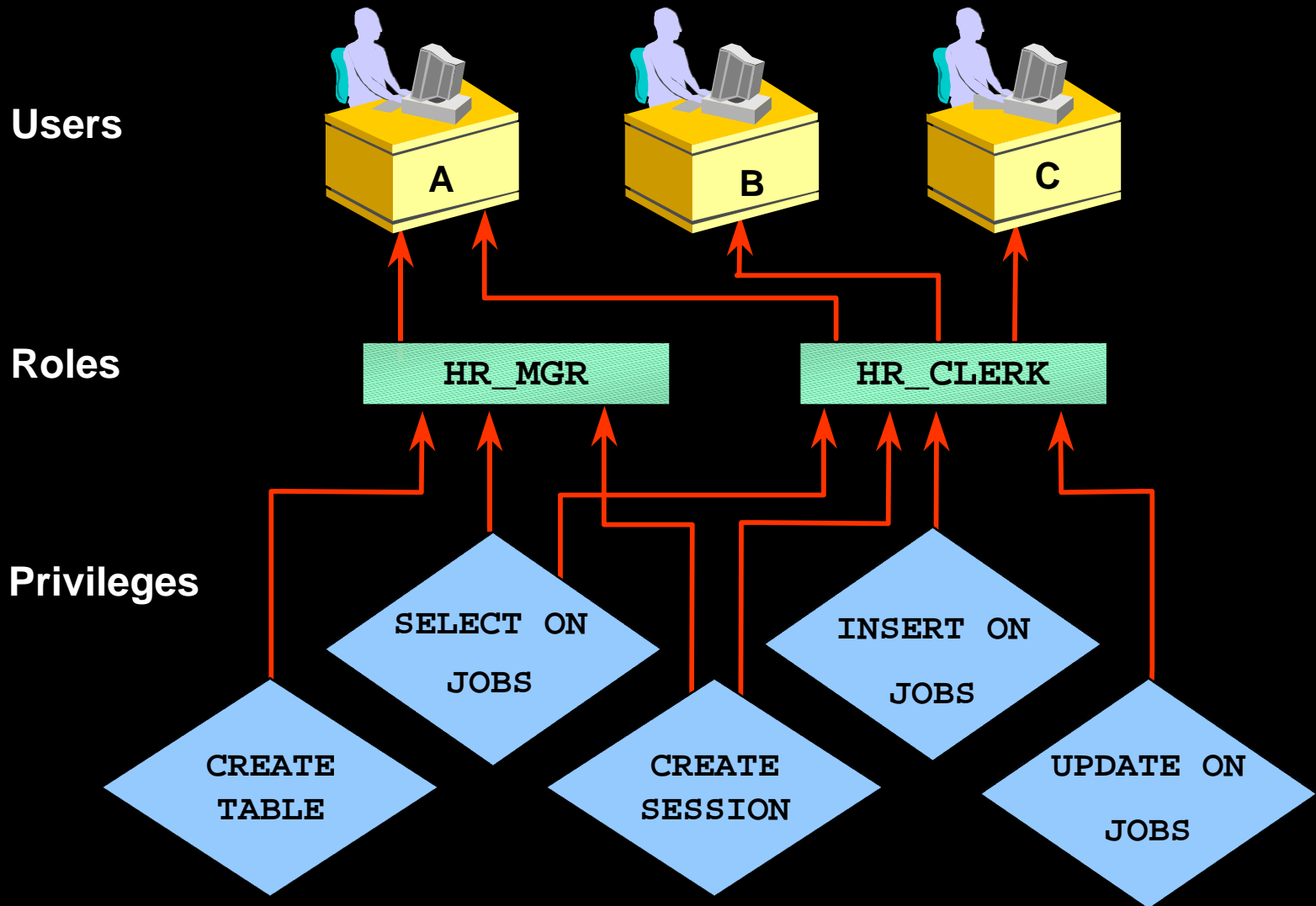
## Managing Roles

# Objectives

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

- **Create and modify roles**
- **Control availability of roles**
- **Remove roles**
- **Use predefined roles**
- **Display role information from the data dictionary**

# Roles



ORACLE

# Benefits of Roles

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

# Creating Roles

```
CREATE ROLE oe_clerk;
```

```
CREATE ROLE hr_clerk  
        IDENTIFIED BY bonus;
```

```
CREATE ROLE hr_manager  
        IDENTIFIED EXTERNALLY;
```

# Predefined Roles

| Role Name                         | Description  |
|-----------------------------------|--|
| <b>CONNECT,<br/>RESOURCE, DBA</b> | <b>These roles are provided<br/>for backward compatibility</b> |
| <b>EXP_FULL_DATABASE</b>          | <b>Privileges to export the<br/>database</b>                   |
| <b>IMP_FULL_DATABASE</b>          | <b>Privileges to import the<br/>database</b>                   |
| <b>DELETE_CATALOG_ROLE</b>        | <b>DELETE privileges on<br/>data dictionary tables</b>         |
| <b>EXECUTE_CATALOG_ROLE</b>       | <b>EXECUTE privilege on<br/>data dictionary packages</b>       |
| <b>SELECT_CATALOG_ROLE</b>        | <b>SELECT privilege on data<br/>dictionary tables</b>          |



# Modifying Roles

```
ALTER ROLE oe_clerk  
        IDENTIFIED BY order;
```

```
ALTER ROLE hr_clerk  
        IDENTIFIED EXTERNALLY;
```

```
ALTER ROLE hr_manager  
        NOT IDENTIFIED;
```

# Assigning Roles

```
GRANT oe_clerk TO scott;
```

```
GRANT hr_clerk TO hr_manager;
```

```
GRANT hr_manager TO scott WITH ADMIN  
OPTION;
```

# Establishing Default Roles

```
ALTER USER scott  
    DEFAULT ROLE hr_clerk, oe_clerk;
```

```
ALTER USER scott DEFAULT ROLE ALL;
```

```
ALTER USER scott DEFAULT ROLE ALL EXCEPT  
hr_clerk;
```

```
ALTER USER scott DEFAULT ROLE NONE;
```

# Application Roles

- Application roles can be enabled only by authorized PL/SQL packages
- The USING package clause creates an Application Role

```
CREATE ROLE admin_role  
IDENTIFIED USING hr.employee;
```

# Enabling and Disabling Roles

- **Disable a role to revoke the role from a user temporarily**
- **Enable a role to grant it temporarily**
- **The `SET ROLE` command enables and disables roles**
- **Default roles are enabled for a user at login.**
- **A password may be required to enable a role.**

# Enabling and Disabling Roles

```
SET ROLE hr_clerk;
```

```
SET ROLE oe_clerk IDENTIFIED BY order;
```

```
SET ROLE ALL EXCEPT oe_clerk;
```

# Removing Roles from Users

```
REVOKE oe_clerk FROM scott;
```

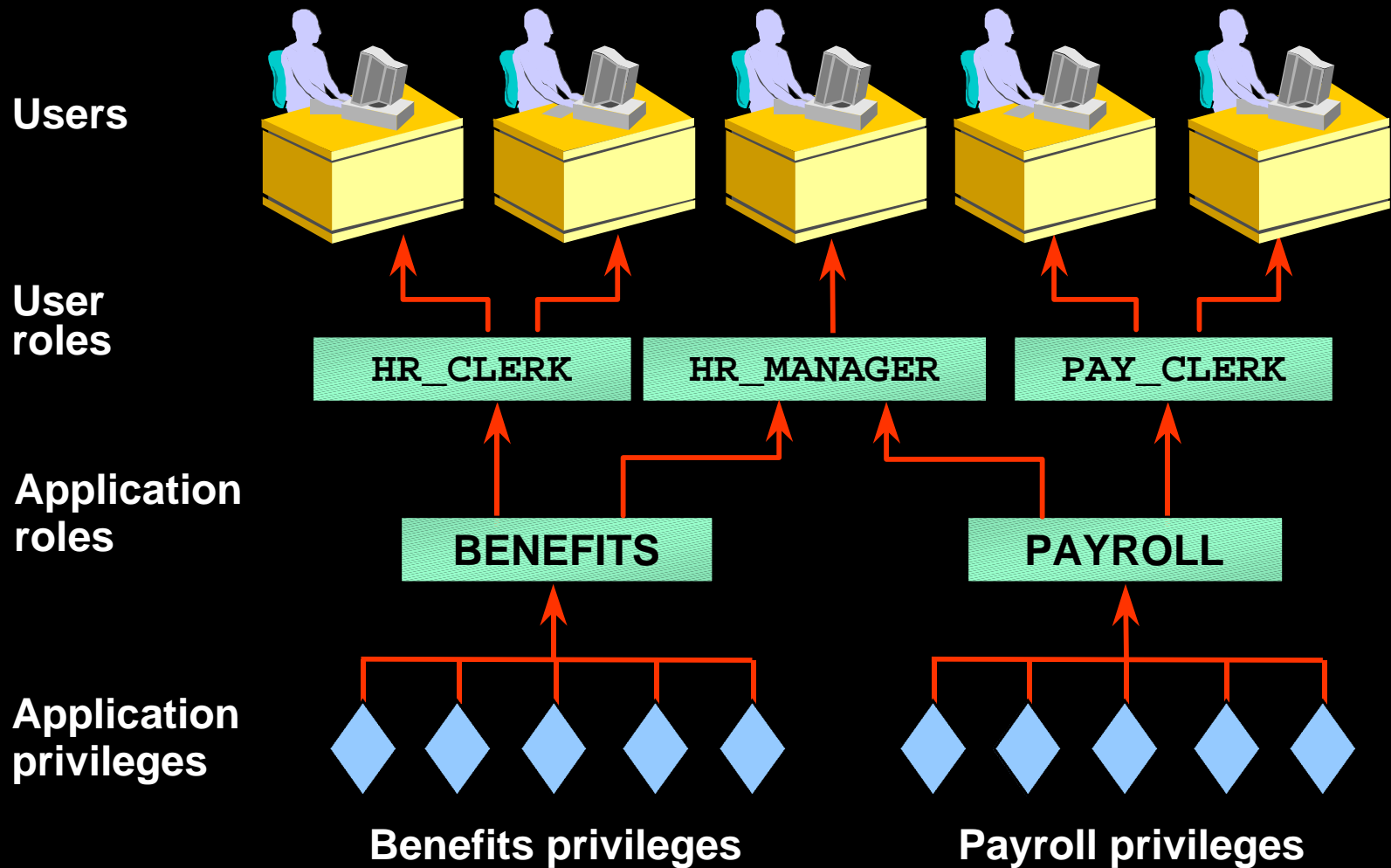
```
REVOKE hr_manager FROM PUBLIC;
```

# Removing Roles

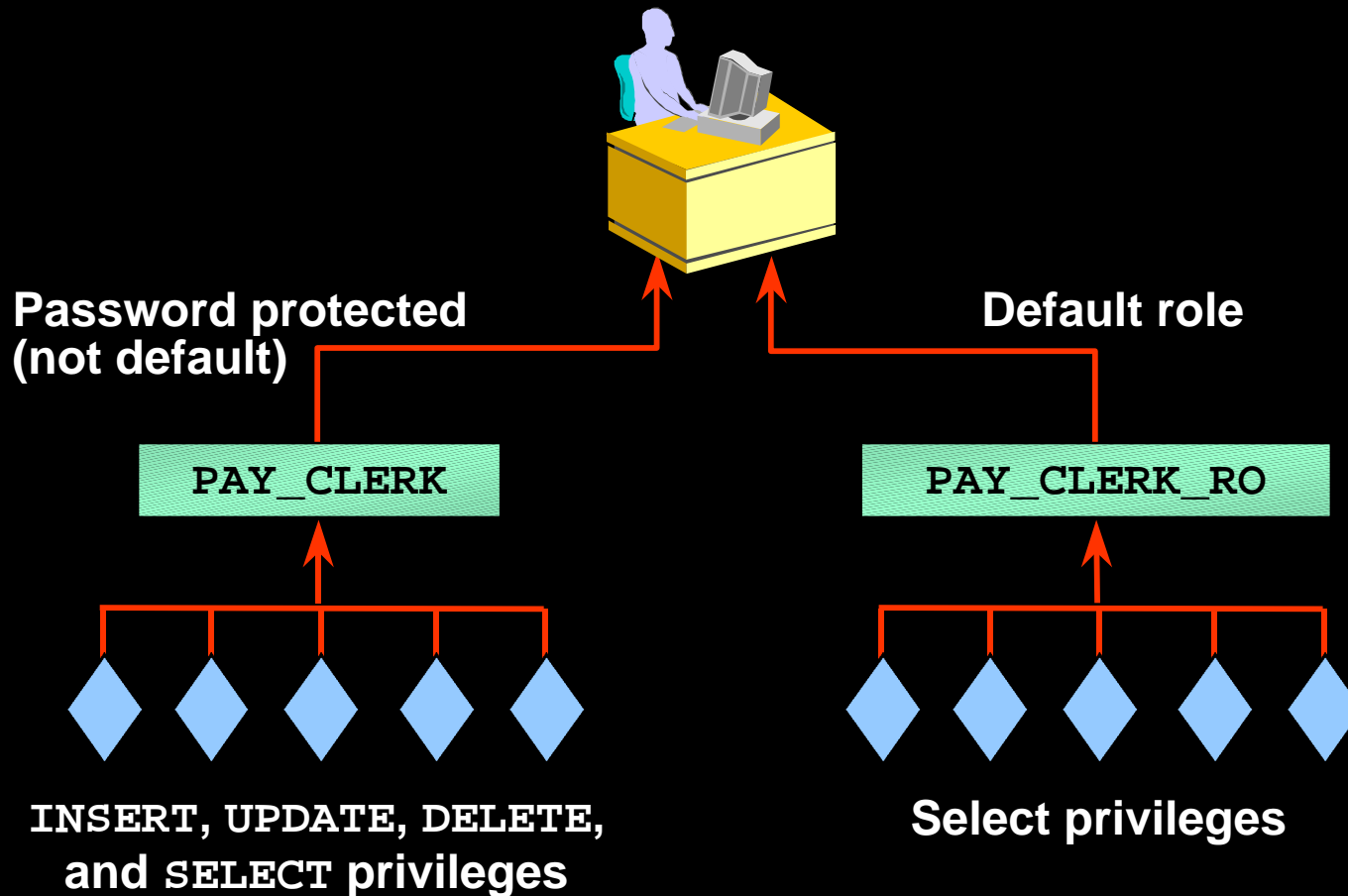
```
DROP ROLE hr_manager;
```



# Guidelines for Creating Roles



# Guidelines for Using Passwords and Default Roles



# Displaying Role Information

| Role View              | Description   |
|------------------------|---|
| <b>DBA_ROLES</b>       | <b>All roles that exist in the database</b>         |
| <b>DBA_ROLE_PRIVS</b>  | <b>Roles granted to users and roles</b>             |
| <b>ROLE_ROLE_PRIVS</b> | <b>Roles that are granted to roles</b>              |
| <b>DBA_SYS_PRIVS</b>   | <b>System privileges granted to users and roles</b> |
| <b>ROLE_SYS_PRIVS</b>  | <b>System privileges granted to roles</b>           |
| <b>ROLE_TAB_PRIVS</b>  | <b>Object privileges granted to roles</b>           |
| <b>SESSION_ROLES</b>   | <b>Roles that the user currently has enabled</b>    |

# Summary

**In this lesson, you should have learned how to:**

- **Create roles**
- **Assign privileges to roles**
- **Assign roles to users or roles**
- **Establish default roles**

# Practice 17 Overview

**This practice covers the following topics:**

- **Listing system privileges for a role**
- **Creating, assigning and dropping roles**
- **Creating Application roles**

# 18

## Using Globalization Support

# Objectives

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

- **Choose database character set and national character set for a database**
- **Specify the language-dependent behavior using initialization parameters, environment variables, and the `ALTER SESSION` command**
- **Use the different types of National Language Support (NLS) parameters**
- **Explain the influence on language-dependent application behavior**
- **Obtain information about Globalization Support usage**

# Globalization Support Features

- **Language support**
- **Territory support**
- **Character set support**
- **Linguistic sorting**
- **Message support**
- **Date and time formats**
- **Numeric formats**
- **Monetary formats**





# Different Types of Encoding Schemes

Oracle supports different classes of character encoding schemes:

- **Single-byte character sets**
  - 7-bit
  - 8-bit
- **Varying-width multibyte character sets**
- **Fixed-width multibyte character sets**
- **Unicode (AL32UFT8, AL16UTF16, UTF8)**

# Database Character Sets and National Character Sets

| Database Character Sets                               | National Character Sets                                |
|---|--|
| Defined at creation time                              | Defined at creation time                               |
| May not be change without re-creation, few exceptions | May not be changed without re-creation, few exceptions |
| Store data columns of type CHAR, VARCHAR2, CLOB, LONG | Store data columns of type NCHAR, NVARCHAR2, NCLOB     |
| Can store varying-width character sets                | Can store Unicode using either AL16UTF16 or UTF8       |

# Guidelines for Choosing an Oracle Database Character Set

## Considerations

- What language does the database need to support?
- What are interoperability concerns with system resources and applications?
- What are the performance implications?
- What are the restrictions?

# Guidelines for Choosing an Oracle National Character Set

- Two choices
  - AL16UTF16
  - UTF8
- Is space an issue?
- Is performance an issue?

# Choosing a Unicode Solution

## Unicode Database

### When Should You Use a Unicode Database?

- Easy code migration for Java or PL/SQL
- Easy data migration from ASCII based data
- Evenly distributed multilingual data
- InterMedia Text Search

# Choosing a Unicode Solution

## Unicode Datatype

**When should you use a Unicode datatype?**

- **Adding multilingual support incrementally**
- **Packaged applications**
- **Performance**

**Single byte database character set with a fixed width national character set**

- **Better support for UTF-16 with windows clients**

# Specifying Language-Dependent Behavior

**Initialization parameter**

**Environment variable**

**ALTER SESSION  
command**

# Specifying Language-Dependent Behavior for the Server

**NLS\_LANGUAGE specifies:**

- The language for messages
- Day and month names
- Symbols for A.D, B.C, A.M, P.M.
- The default sorting mechanism

**NLS\_TERRITORY specifies:**

- Day and week numbering
- Default date format, decimal character, group separator, and the default ISO and local currency symbols



# Dependent Language and Territory Default Values

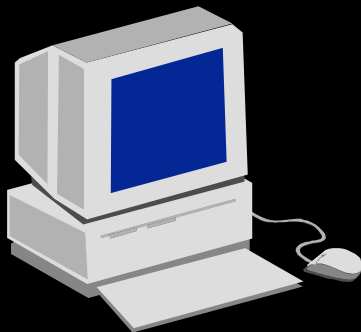
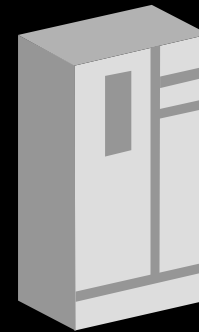
| PARAMETER              | VALUES    |
|------------------------|-----------|
| NLS_LANGUAGE           | AMERICAN  |
| NLS_DATE_LANGUAGE      | AMERICAN  |
| NLS_SORT               | BINARY    |
| NLS_TERRITORY          | AMERICA   |
| NLS_CURRENCY           | \$        |
| NLS_ISO_CURRENCY       | AMERICA   |
| NLS_DATE_FORMAT        | DD-MON-RR |
| NLS_NUMERIC_CHARACTERS | , .       |

# Specifying Language-Dependent Behavior for the Session

- **Environment variable:**  
`NLS_LANG=French_France.UTF8`
- **Additional environment variables:**
  - `NLS_DATE_FORMAT`
  - `NLS_DATE_LANGUAGE`
  - `NLS_SORT`
  - `NLS_NUMERIC_CHARACTERS`
  - `NLS_CURRENCY`
  - `NLS_ISO_CURRENCY`
  - `NLS_CALENDAR`

# Character Sets in Client-Server Architecture

```
NLS_LANG=<language>_<territory>.<charset>  
NLS_NCHAR=<ncharset>
```



```
CREATE DATABASE ...  
CHARACTER SET <charset>  
NATIONAL CHARACTER SET  
<ncharset>  
...
```

# Specifying Language-Dependent Behavior for the Session

```
ALTER SESSION SET NLS_DATE_FORMAT='DD.MM.YYYY';
```

```
DBMS_SESSION.SET-NLS('NLS_DATE_FORMAT',  
'''DD.MM.YYYY''') ;
```

# Linguistic Sorting

**Oracle provides three types of sorting**

- **Binary sorting, sorted according to the binary values of the encoded characters**
- **Monolingual sorting**
  - **Sorts in two passes**
  - **Based on a character's assigned major and minor values**
- **Multilingual sorting**
  - **Based on the new ISO 14651 and,**
  - **Unicode 3.0 Standard for multilingual collation**

# NLS Sorting

- **NLS\_SORT** specifies the type of sort for character data
  - Is defined by the **NLS\_LANG** environment variable
  - May be overridden at the session level
- **NLSORT** function
  - Specifies the type of sort for character data
  - Allows sorts to defined at the query level

# Using NLS Parameters in SQL Functions

```
SELECT TO_CHAR(hire_date, 'DD.Mon.YYYY',  
              'NLS_DATE_LANGUAGE=FRENCH')  
FROM   employees;
```

```
SELECT ename, TO_CHAR(sal, '9G999D99',  
              'NLS_NUMERIC_CHARACTERS=''', ''')  
FROM   emp;
```

# Linguistic Index Support

- Linguistic indexing
- High performance with local sorting

```
CREATE INDEX list_word ON  
  list (NLSSORT(word, 'NLS_SORT = French_M'));
```

- **NLS\_COMP** parameter for linguistic comparisons



# Import and Loading Data Using NLS

- Data will be converted from export file character set to the database character set during the import.
- **LOADER:**
  - **Conventional:** Data is converted into the session character set specified by `NLS_LANG`.
  - **DIRECT:** Data is converted directly into the database character set.

# Obtaining Information About Character Sets

**NLS\_DATABASE\_PARAMETERS:**

- **PARAMETER**  
**(NLS\_CHARACTERSET,**  
**NLS\_NCHAR\_CHARACTERSET)**
- **VALUE**

# Obtaining Information About NLS Settings

## **NLS\_INSTANCE\_PARAMETERS:**

- **PARAMETER (initialization parameters that have been explicitly set)**
- **VALUE**

## **NLS\_SESSION\_PARAMETERS:**

- **PARAMETER (session parameters)**
- **VALUE**

# Obtaining Information About NLS Settings

**V\$NLS\_VALID\_VALUES:**

- **PARAMETER**  
**(LANGUAGE, SORT, TERRITORY, CHARACTERSET)**
- **VALUE**

**V\$NLS\_PARAMETERS:**

- **PARAMETER (NLS session parameters,  
NLS\_CHARACTERSET)**
- **VALUE**

# Globalization Support Utilities

- **Character Set Scanner**
  - Scans the database to determine if the character set can be changed
  - Reports are provided detailing possible problems and fixes
- **Oracle Locale Builder**
  - Easy to use graphical interface
  - For viewing, modifying, and creating locale definitions

# Summary

**In this lesson, you should have learned how to:**

- **Choose a database character set and a national character set for the database**
- **Use the various types of National Language Support parameters for the server, or the session**

# Practice 18 Overview

**This practice covers the following topics:**

- **Checking the database and national character set**
- **Identifying valid NLS values**
- **Setting NLS parameters**