Oracle9*i* DBA Fundamentals I

Volume 1 • Student Guide

D11321GC10
Production 1.0
May 2001
D32643

ORACLE®

Authors

Sarath Chandran

Marie St. Gelais

S Matt Taylor Jr

Technical Reviewers

Howard Bradley

Ruth Baylis

Paul Cartwright

Trevor Davis

Joel Goodman

Scott Gossett

Tomoki Ishii

Peter Kilpatrick

Stefan Lindblad

Howard Ostrow

Ashesh Parekh

Radhanes Petronilla

Venceslava Pretlova

Andreas Reinhardt

Ranbir Singh

Peter Sharman

Barry Trute

Ric VanDyke

Karla Villasenor

John Watson

Teppei Yagihashi

Publisher

Shane Mattimoe

Contents

Preface

Profile

Before You Begin This Course:

How This Course Is Organized

This is an instructor-led course featuring lecture and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills introduced.

Related Publications

Additional Publications

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

Typographic Conventions

Typographic Conventions in Text

Convention	Element	Example
Bold italic	Glossary term (if there is a glossary)	The ***algorithm*** inserts the new key.
Caps and lowercase	Buttons, check boxes, triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger to the ORD block. Open the Master Schedule window.
Courier new, case sensitive (default is lowercase)	Code output, directory names, filenames, passwords, pathnames, URLs, user input, usernames	Code output: `debug.set ('I", 300);` Directory: `bin` (DOS), `$FMHOME` (UNIX) Filename: Locate the `init.ora` file. Password: User `tiger` as your password. Pathname: Open `c:\my_docs\projects` URL: Go to `http://www.oracle.com` User input: Enter `300` Username: Log on as `scott`
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (*but* Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do *not* save changes to the database. For further information, see *Oracle7 Server SQL Language Reference Manual.* Enter `user_id@us.oracle.com`, where *user_id* is the name of the user.
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select "Include a reusable module component" and click Finish. This subject is covered in Unit II, Lesson 3, "Working with Objects."
Uppercase	SQL column names, commands, functions, schemas, table names	Use the SELECT command to view information stored in the LAST_NAME column of the EMP table.

Convention	Element	Example
Arrow	Menu paths	Select File–> Save.
Brackets	Key names	Press [Enter].
Commas	Key sequences	Press and release keys one at a time: [Alternate], [F], [D]
Plus signs	Key combinations	Press and hold these keys simultaneously: [Ctrl]+[Alt]+[Del]

Typographic Conventions in Code

Convention	Element	Example
Caps and lowercase	Oracle Forms triggers	`When-Validate-Item`
Lowercase	Column names, table names	`SELECT last_name` `FROM s_emp;`
	Passwords	`DROP USER scott` `IDENTIFIED BY tiger;`
	PL/SQL objects	`OG_ACTIVATE_LAYER` `(OG_GET_LAYER ('prod_pie_layer'))`
Lowercase italic	Syntax variables	`CREATE ROLE` *`role`*
Uppercase	SQL commands and functions	`SELECT userid` `FROM emp;`

Typographic Conventions in Navigation Paths

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

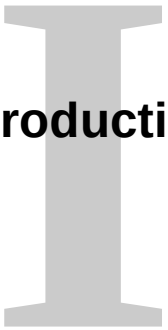
(N) Invoice—>Entry—>Invoice Batches Summary (M) Query—>Find (B) Approve

This simplified path translates to the following:

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

N = Navigator, **M** = Menu, **B** = Button

Introduction



ORACLE

Objectives

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

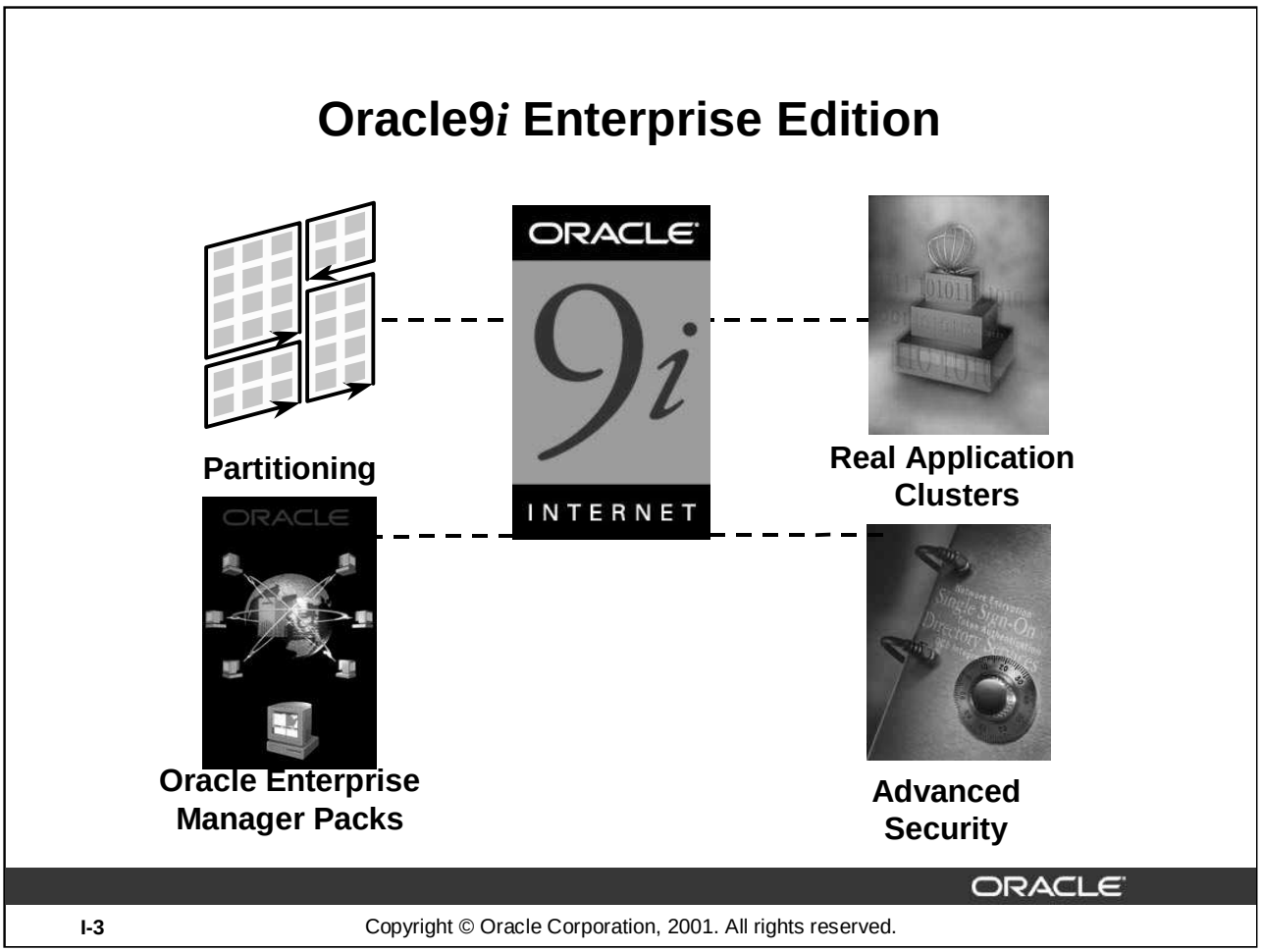
- **Identify the various components of the Oracle architecture.**
- **Start and shut down an Oracle database**
- **Create an operational database**
- **Manage Oracle control files, redo log files, database files, tablespaces, segments, extents, and blocks**
- **Manage users, privileges, and resources**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

ORACLE

Scope of the Course

This course is the first in a series of courses that cover the core database administrator tasks. The tasks covered in this course are:

- Outlining the Oracle architecture
- Planning and creating databases
- Managing the memory, process, physical, and logical structures
- Managing database users, and controlling and monitoring their actions
- Using the Globalization Support features



What Is the Oracle9*i* Enterprise Edition?

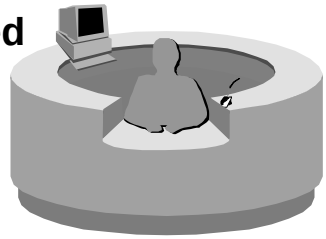
Oracle9*i* Enterprise Edition is an object relational database that is scalable and easily manageable. The administration of the basic enterprise edition is discussed in this course. However, the following options provide additional functionality.

- Partitioning: Provides facilities for implementing large, scalable applications. (It enables control over tables and indexes at a lower level of granularity than is possible with the basic enterprise edition.)
- Real Application Clusters: Improves the scalability and availability of a database by allowing multiple copies of the Oracle software to access a single database.
- Oracle Enterprise Manager Packs: Built on top of the Oracle Enterprise Manager. Oracle Enterprise Manager Diagnostics, Tuning, and Change Management Packs are add-ons that provide DBAs with a set of tools for advanced diagnostics, monitoring, tuning, and change management of Oracle environments. Oracle Enterprise Manager also provides a DBA Management Pack that enables DBAs to perform basic administration, such as creating users, starting up an instance, granting privileges, and so on.
- Advanced Security: Provides client-server, server-server network security, using encryption and data integrity checking, and supports enhanced user authentication services, using third-party security services.

Note: Options do require a license to be purchased.

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**

ORACLE

Database Administrator Tasks

Database administrators are responsible for maintaining the Oracle server so that the server can process user requests. An understanding of the Oracle architecture is necessary to maintain it effectively. This course will be focusing on outlining the Oracle architecture, and other administrator tasks such as: planning and creating databases, managing database availability, managing memory, physical, and logical structures, managing users and privileges.

Database Administrator Tasks Covered in Other Courses

The following tasks are discussed in other courses:

- Backup and Recovery in *Oracle9i DBA Fundamentals II*
- Network Administration in *Oracle9i DBA Fundamentals II*
- Database Tuning in *Oracle9i Performance Tuning*

1

Oracle Architectural Components

ORACLE

Objectives

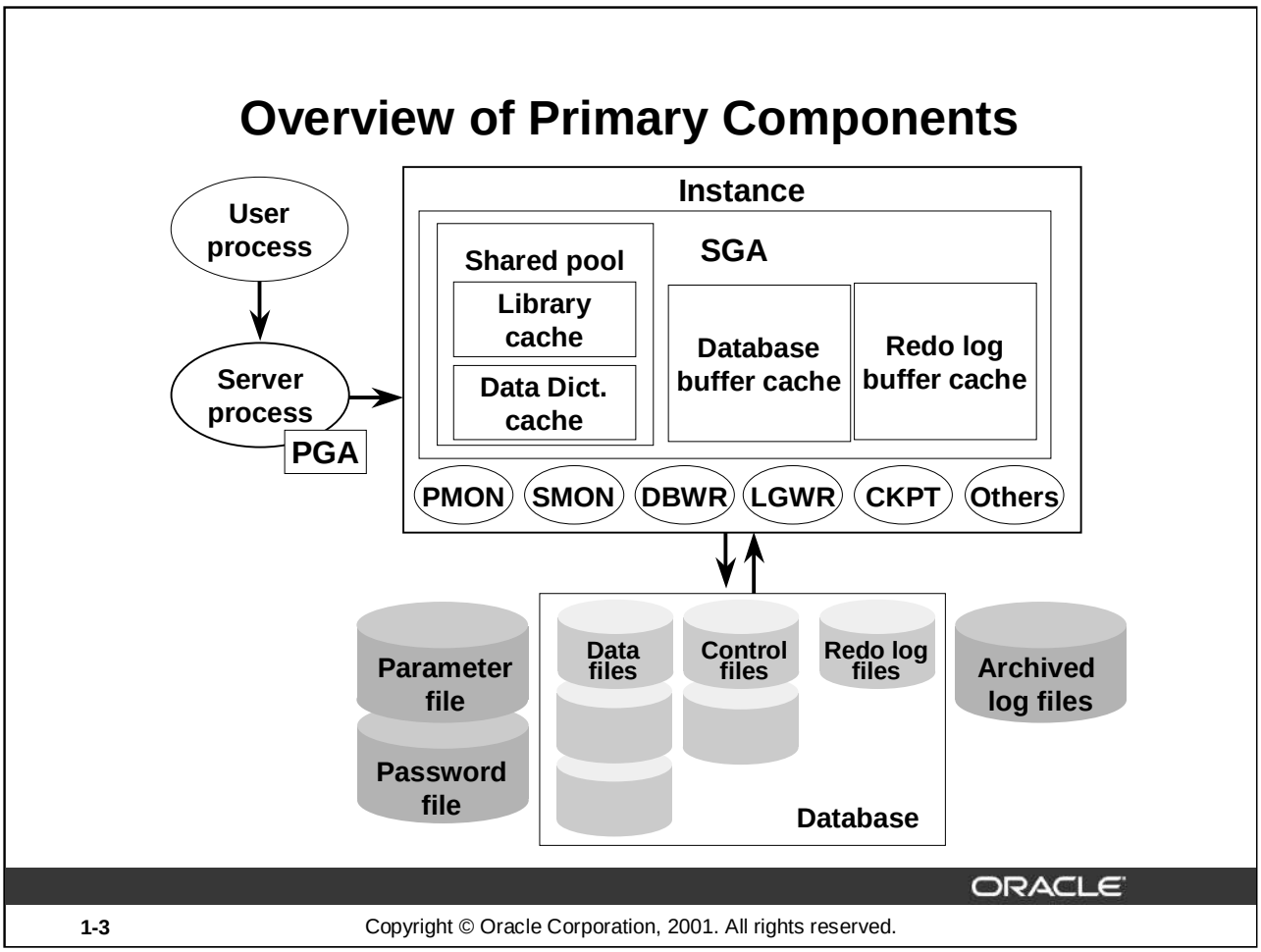
After completing this lesson, you should be able to do the following:

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

ORACLE

Objectives

This lesson introduces the Oracle server architecture by examining the physical, memory, process, and logical structures involved in establishing a database connection, creating a session, and executing SQL commands.



Overview of Primary Components

The Oracle architecture includes a number of primary components, which are discussed further in this lesson.

- Oracle server: There are several files, processes, and memory structures in an Oracle server; however, not all of them are used when processing a SQL statement. Some are used to improve the performance of the database, ensure that the database can be recovered in the event of a software or hardware error, or perform other tasks necessary to maintain the database. The Oracle server consists of an Oracle instance and an Oracle database.
- Oracle instance: An Oracle instance is the combination of the background processes and memory structures. The instance must be started to access the data in the database. Every time an instance is started, a System Global Area (SGA) is allocated and Oracle background processes are started. Background processes perform functions on behalf of the invoking process. They consolidate functions that would otherwise be handled by multiple Oracle programs running for each user. The background processes perform input/output (I/O) and monitor other Oracle processes to provide increased parallelism for better performance and reliability.
- Oracle database: An Oracle database consists of operating system files, also known as database files, that provide the actual physical storage for database information. The database files are used to ensure that the data is kept consistent and can be recovered in the event of a failure of the instance.

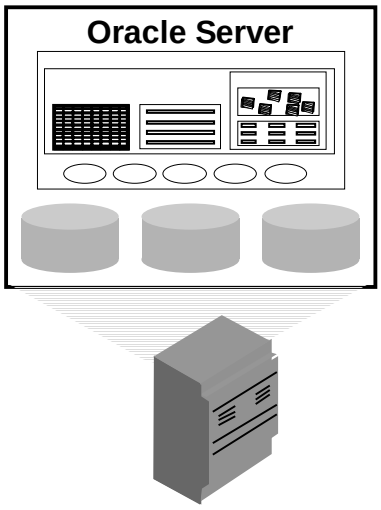
Overview of Primary Components (continued)

- Other key files: Nondatabase files are used to configure the instance, authenticate privileged users, and recover the database in the event of a disk failure.
- User and server processes: The user and server processes are the primary processes involved when a SQL statement is executed; however, other processes may help the server complete the processing of the SQL statement.
- Other processes: Many other processes exist that are used by other options within Oracle, such as Advanced Queuing, Real Application Clusters, Shared Server, Advanced Replication, and so on. These processes are discussed within their respective courses.

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

Oracle Server

The Oracle server can run on a number of different computers in one of the following ways:

- Client-Application Server-Server
- Client-Server
- Host-Based

Client-Application Server-Server: (Three-tier) Users access the database from their personal computers (clients) through an application server, which is used for the application's processing requirements.

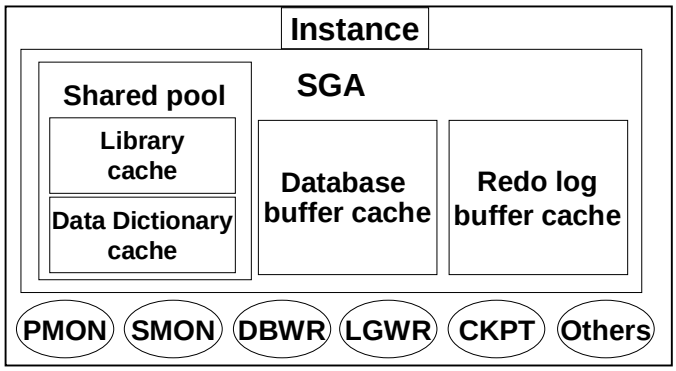
Client-Server: (Two-tier) Users access the database from their personal computer (client) over a network, and the database sits on a separate computer (server).

Host-Based: Users are connected directly to the same computer on which the database resides.

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**



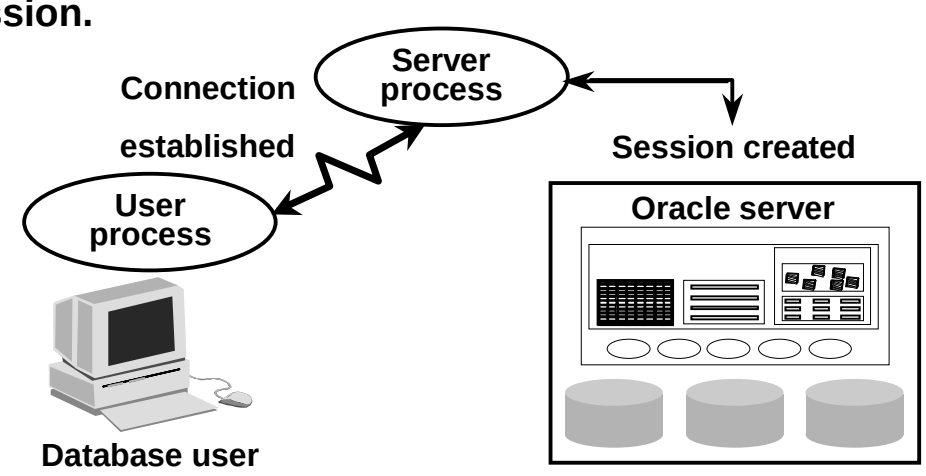
ORACLE

Oracle Instance

An Oracle instance consists of the System Global Area (SGA) memory structure and the background processes used to manage a database. An instance is identified by using methods specific to each operating system. The instance can open and use only one database at a time.

Establishing a Connection and Creating a Session

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



ORACLE

Connecting to an Oracle Instance

Before users can submit SQL statements to an Oracle database, they must connect to an instance.

- The user starts a tool such as SQL*Plus or runs an application developed using a tool such as Oracle Forms. This application or tool is executed as a user process.
- In the most basic configuration, when a user logs on to the Oracle server, a process is created on the computer running the Oracle server. This process is called a server process. The server process communicates with the Oracle instance on behalf of the user process that runs on the client. The server process executes SQL statements on behalf of the user.

Connection

A connection is a communication pathway between a user process and an Oracle server. A database user can connect to an Oracle server in one of three ways:

- The user logs on to the operating system running the Oracle instance and starts an application or tool that accesses the database on that system. The communication pathway is established using the interprocess communication mechanisms available on the host operating system.

Connecting to an Oracle Instance (continued)

- The user starts the application or tool on a local computer and connects over a network to the computer running the Oracle instance. In this configuration, called client-server, network software is used to communicate between the user and the Oracle server.
- In a three-tiered connection, the user's computer communicates over the network to an application or a network server, which is connected through a network to the machine running the Oracle instance. For example, the user runs a browser on a computer on a network to use an application residing on an NT server that retrieves data from an Oracle database running on a UNIX host.

Sessions

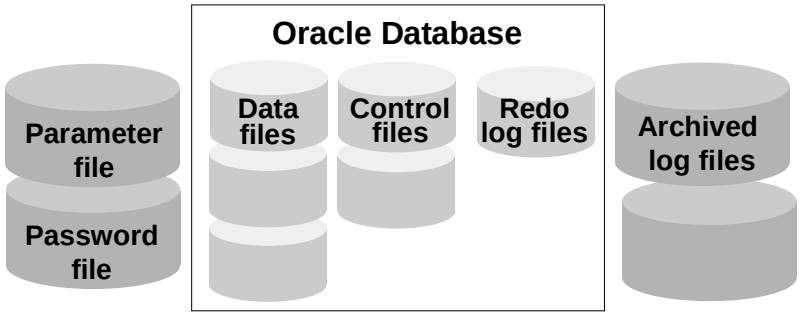
A session is a specific connection of a user to an Oracle server. The session starts when the user is validated by the Oracle server, and it ends when the user logs out or when there is an abnormal termination. For a given database user, many concurrent sessions are possible if the user logs on from many tools, applications, or terminals at the same time. Except for some specialized database administration tools, starting a database session requires that the Oracle server be available for use.

Note: The type of connection explained here, where there is a one-to-one correspondence between a user and server process, is called a dedicated server connection. When using a shared server configuration, it is possible for multiple user processes to share server processes.

Oracle Database

An Oracle database:

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



ORACLE

An Oracle Database

The general purpose of a database is to store and retrieve related information. An Oracle database has a logical and a physical structure. The physical structure of the database is the set of operating system files in the database. An Oracle database consists of three file types.

- Data files containing the actual data in the database
- Redo logs containing a record of changes made to the database to enable recovery of the data in case of failures
- Control files containing information necessary to maintain and verify database integrity

Other Key File Structures

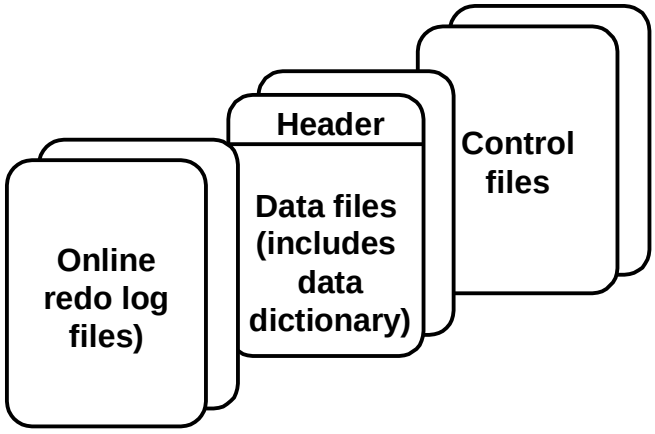
The Oracle server also uses other files that are not part of the database:

- The parameter file defines the characteristics of an Oracle instance. For example, it contains parameters that size some of the memory structures in the SGA.
- The password file authenticates users privileged to start up and shut down an Oracle instance.
- Archived redo log files are offline copies of the redo log files that may be necessary to recover from media failures.

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**



ORACLE

Physical Structure

Other key files exist and are required to start up and use a database, for example: parameter files, configuration files, password files and so on. However, the physical structure of an Oracle database includes only three types of files: control files, data files, and 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**

ORACLE

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**

ORACLE

System Global Area

The SGA is also called the shared global area. It is used to store database information that is shared by database processes. It contains data and control information for the Oracle server and is allocated in the virtual memory of the computer where Oracle resides.

The following statement can be used to view SGA memory allocations:

```
SHOW SGA;
Total System Global Area     36437964   bytes
Fixed Size                    6543794   bytes
Variable Size                19521536   bytes
Database Buffers             16777216   bytes
Redo Buffers                    73728   bytes
```

System Global Area (SGA) (continued)

Dynamic SGA

A dynamic SGA implements an infrastructure that allows the SGA configuration to change without shutting down the instance. This then allows the sizes of the database buffer cache, shared pool, and large pool to be changed without shutting down the instance. Conceivably, the database buffer cache, shared pool, and large pool could be initially under configured and would grow and shrink depending upon their respective work loads, up to a maximum of SGA_MAX_SIZE.

Sizing the SGA

The size of the SGA is determined by several initialization parameters. The parameters that most affect SGA size are:

DB_CACHE_SIZE: The size of the cache of standard blocks.

LOG_BUFFER: The number of bytes allocated for the redo log buffer cache.

SHARED_POOL_SIZE: The size in bytes of the area devoted to shared SQL and PL/SQL.

LARGE_POOL_SIZE: The size of the large pool; the default is zero.

Note: Dynamic SGA and sizing are covered in further detail in the *Oracle9i Performance Tuning* course.

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**

ORACLE

System Global Area

Unit of Allocation

A granule is a unit of contiguous virtual memory allocation. The size of a granule depends on the estimated total SGA size whose calculation is based on the value of the parameter SGA_MAX_SIZE.

- 4 MB if estimated SGA size is < 128 MB
- 16 MB otherwise

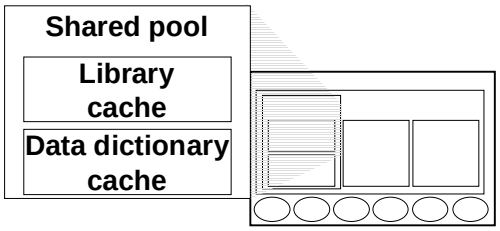
The components (buffer cache, shared pool, and large pool) are allowed to grow and shrink based on granule boundaries. For each component which owns granules, the number of granules allocated to the component, any pending operations against the component (e.g., allocation of granules via ALTER SYSTEM, freeing of granules via ALTER SYSTEM, corresponding self-tuning), and target size in granules will be tracked and displayed by the V$BUFFER_POOL view. At instance startup, the Oracle server allocates granule entries, one for each granule to support SGA_MAX_SIZE bytes of address space. As startup continues, each component acquires as many granules as it requires. The minimum SGA configuration is three granules (one granule for fixed SGA (includes redo buffers; one granule for buffer cache; one granule for shared pool).

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;
```



ORACLE

Shared Pool

The shared pool environment contains both fixed and variable structures. The fixed structures remain relatively the same size, whereas the variable structures grow and shrink based on user and program requirements. The actual sizing for the fixed and variable structures is based on an initialization parameter and the work of an Oracle internal algorithm.

Sizing the Shared Pool

Since the shared pool is used for objects that can be shared globally, such as reusable SQL execution plans; PL/SQL packages, procedures, and functions; and cursor information, it must be sized to accommodate the needs of both the fixed and variable areas. Memory allocation for the shared pool is determined by the `SHARED_POOL_SIZE` initialization parameter. It can be dynamically resized using `ALTER SYSTEM SET`. After performance analysis, this can be adjusted but the total SGA size cannot exceed `SGA_MAX_SIZE`.

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**

ORACLE

Library Cache

The library cache size is based on the sizing defined for the shared pool. Memory is allocated when a statement is parsed or a program unit is called. If the size of the shared pool is too small, statements are continually reloaded into the library cache, which affects performance. The library cache is managed by a least recently used (LRU) algorithm. As the cache fills, less recently used execution paths and parse trees are removed from the library cache to make room for the new entries. If the SQL or PL/SQL statements are not reused, they eventually are aged out.

The library cache consists of two structures:

- Shared SQL: The Shared SQL stores and shares the execution plan and parse tree for SQL statements run against the database. The second time that an identical SQL statement is run, it is able to take advantage of the parse information available in the shared SQL to expedite its execution. To ensure that SQL statements use a shared SQL area whenever possible, the text, schema, and bind variables must be exactly the same.
- Shared PL/SQL: The shared PL/SQL area stores and shares the most recently executed PL/SQL statements. Parsed and compiled program units and procedures (functions, packages, and triggers) are stored in this area.

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.**

ORACLE

Data Dictionary Cache

The data dictionary cache is also referred to as the dictionary cache or row cache. Caching data dictionary information into memory improves response time. Information about the database (user account data, data file names, segment names, extent locations, table descriptions, and user privileges) is stored in the data dictionary tables. When this information is needed by the database, the data dictionary tables are read, and the data that is returned is stored in the data dictionary cache.

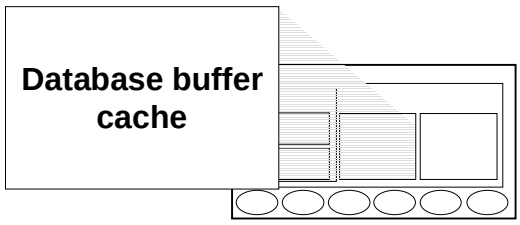
Sizing the Data Dictionary

The overall size is dependent on the size of the shared pool size and is managed internally by the database. If the data dictionary cache is too small, then the database has to query the data dictionary tables repeatedly for information needed by the database. These queries are called recursive calls and are slower than the queries that are handled by the data dictionary cache.

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.**



ORACLE

Database Buffer Cache

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

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.**

ORACLE

Database Buffer Cache (continued)

Sizing the Database Buffer Cache

The size of each buffer in the buffer cache is equal to the size of an Oracle block, and it is specified by the DB_BLOCK_SIZE parameter. The database buffer cache consists of independent sub-caches for buffer pools and for multiple block sizes. The parameter DB_BLOCK_SIZE determines the primary block size, which is used for the SYSTEM tablespace.

Three parameters define the sizes of the buffer caches:

DB_CACHE_SIZE: Sizes the default buffer cache size only, it always exists and cannot be set to zero.

DB_KEEP_CACHE_SIZE: Sizes the keep buffer cache, which is used to retain blocks in memory that are likely to be reused.

DB_RECYCLE_CACHE_SIZE: Sizes the recycle buffer cache, which is used to eliminate blocks from memory that have little change of being reused.

Multiple Block Sizes

An Oracle database can be created with a standard block size and up to four non-standard block sizes. Non-standard block sizes can have any power-of-two value between 2 KB and 32 KB.

Note: Multiple block sizing is covered further in the *Oracle9i Performance Tuning* course.

Database Buffer Cache (Continued)

Buffer Cache Advisory Parameter

The buffer cache advisory feature enables and disables statistics gathering for predicting behavior with different cache sizes. The information provided by these statistics can help DBA size the buffer cache optimally for a given workload. The buffer cache advisory information is collected and displayed through the V$DB_CACHE_ADVICE view.

The buffer cache advisory is enabled via the initialization parameter DB_CACHE_ADVICE. It is a dynamic parameter via ALTER SYSTEM. Three values (OFF, ON, READY) are available.

DB_CACHE_ADVICE Parameter Values

OFF: Advisory is turned off and the memory for the advisory is not allocated

ON: Advisory is turned on and both cpu and memory overhead is incurred

Attempting to set the parameter to this state when it is in the OFF state may lead to ORA-4031 Inability to allocate from the shared pool when the parameter is switched to ON. If the parameter is in a READY state it can be set to ON without error since the memory is already allocated.

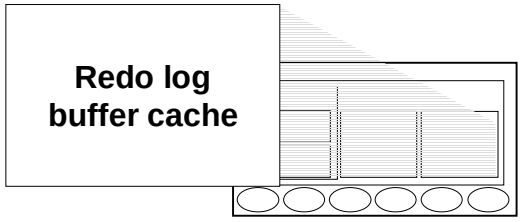
READY: Advisory is turned off but the memory for the advisory remains allocated. Allocating the memory before the advisory is actually turned on will avoid the risk of ORA-4031. If the parameter is switched to this state from OFF, it is possible than an ORA-4031 will be raised

Note: Resizing the buffer caches dynamically, the database buffer advisory, and the use and interpretation of the V$DB_CACHE_ADVICE are covered further in the *Oracle9i Performance Tuning* course

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`.**



ORACLE

Redo Log Buffer Cache

The redo log buffer cache is a circular buffer that contains changes made to data file blocks. This information is stored in redo entries. Redo entries contain the information necessary to recreate the data prior to the change made by `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `ALTER`, or `DROP` operations.

Sizing the Redo Log Buffer Cache

The size of the redo log buffer cache is defined by the initialization parameter `LOG_BUFFER`.

Note: Sizing the Redo Log Buffer Cache is covered in further detail in the *Oracle9i Performance Tuning* course. Refer to the *Managing Redo Log Files* lesson for details regarding redo log files.

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;
```

ORACLE

Large Pool

When users connect through the shared server, Oracle needs to allocate additional space in the shared pool for storing information about the connections between the user processes, dispatchers, and servers. The large pool relieves the burden on areas within the shared pool. The shared pool does not have to give up memory for caching SQL parse trees in favor of shared server session information, I/O, and backup and recover processes. The performance gain is from the reduction of overhead from increasing and shrinkage of the shared SQL cache.

Backup and Restore

Recovery Manager (RMAN) uses the large pool when the `BACKUP_DISK_IO= n` and `BACKUP_TAPE_IO_SLAVE = TRUE` parameters are set. If the large pool is configured but is not large enough, the allocation of memory from the large pool fails. RMAN writes an error message to the alert log file and does not use I/O slaves for backup or restore.

Sizing the Large Pool

The large pool is sized in bytes defined by the `LARGE_POOL_SIZE` parameter.

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.**

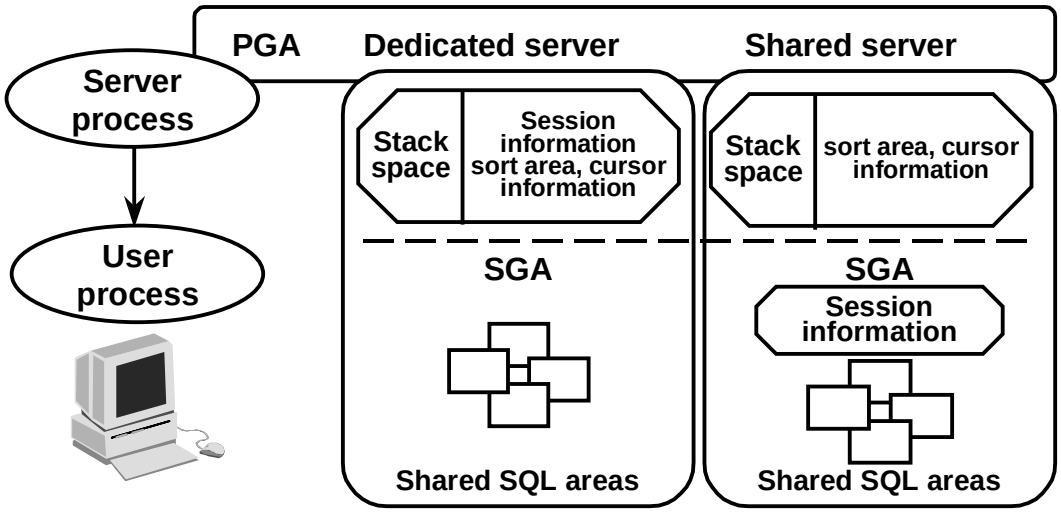
ORACLE

Java Pool

The Java pool is an optional setting but is required if installing and using Java. Its size is set, in bytes, using the `JAVA_POOL_SIZE` parameter. In Oracle9*i*, the default size of the Java Pool is 24M.

Program Global Area (PGA)

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



ORACLE

Program Global Area Components

The Program Global Area or Process Global Area (PGA) is a memory region that contains data and control information for a single server process or a single background process. The PGA is allocated when a process is created and deallocated when the process is terminated. In contrast to the SGA, which is shared by several processes, the PGA is an area that is used by only one process. In a dedicated server configuration, the PGA includes these components:

- Sort area: Used for any sorts that may be required to process the SQL statement
- Session information: Includes user privileges and performance statistics for the session
- Cursor state: Indicates the stage in the processing of the SQL statements that are currently used by the session
- Stack space: Contains other session variables

Note: Some of these structures are stored in the SGA when using a shared server configuration. If using a shared server configuration, it is possible for multiple user processes to share server processes. If a large pool is created, the structures are stored in the large pool; otherwise, they are stored in the shared pool.

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:

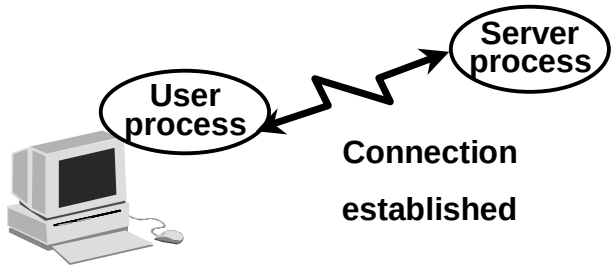
- **User process: Started at the time a database user requests connection to the Oracle server**
- **Server process: Connects to the Oracle Instance and is started when a user establishes a session.**
- **Background process: Available when an Oracle instance is started**

ORACLE

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.**



ORACLE

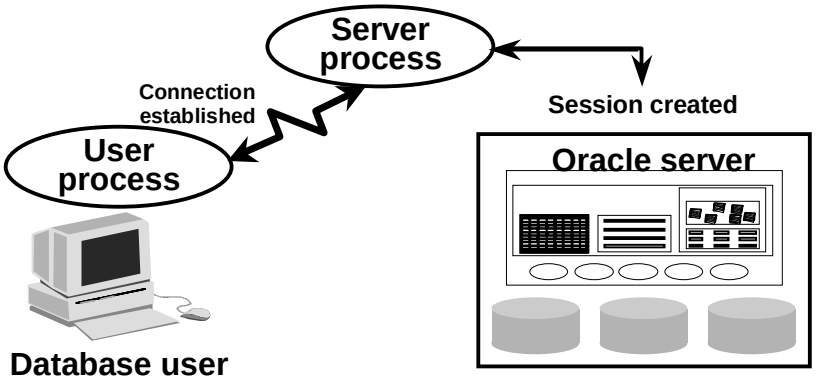
User Process

A database user who needs to request information from the database must first make a connection with the Oracle server. The connection is requested using a database interface tool, such as SQL*Plus, and beginning the user process. The user process does not interact directly with the Oracle server. Rather it generates calls through the user program interface (UPI), which creates a session and starts a server process.

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.**



ORACLE

1-27

Server Process

Once a user has established a connection, a server process is started to handle the user processes requests. A server process can be either a dedicated server process or a shared server process. In a dedicated server environment, the server process handles the request of a single user process. Once a user process disconnects, the server process is terminated. In a shared server environment, the server process handles the request of several user processes. The server process communicates with the Oracle server using the Oracle Program Interface (OPI).

Note: Allocation of server process in a dedicated environment versus a shared environment is covered in further detail in the *Oracle9i Performance Tuning* course.

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	

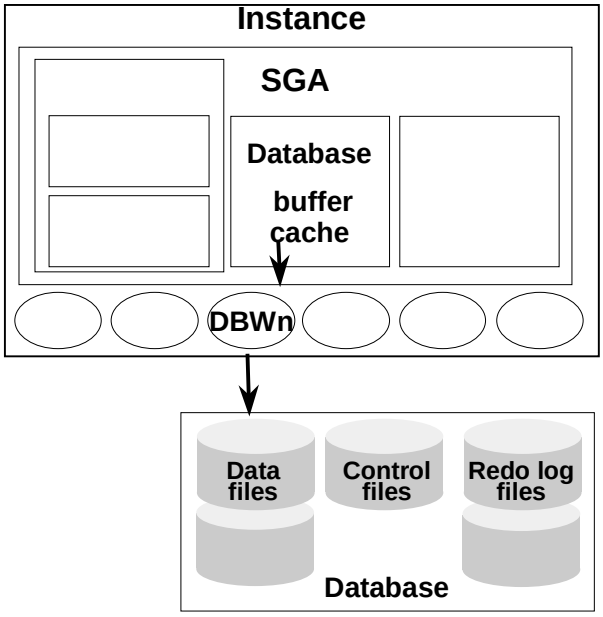
ORACLE

Background Processes

The Oracle architecture has five mandatory background processes that are discussed further in this lesson. In addition to the mandatory list, Oracle has many optional background process that are started when their option is being used. These optional processes are not within the scope of this course, with the exception of the ARCn background process. Following is a list of some optional background processes:

- RECO: Recoverer
- QMNn: Advanced Queuing
- ARCn: Archiver
- LCKn: RAC Lock Manager—Instance Locks
- LMON: RAC DLM Monitor—Global Locks
- LMDn: RAC DLM Monitor—Remote Locks
- CJQ0: Snapshot Refresh
- Dnnn: Dispatcher
- Snnn: Shared Server
- Pnnn: Parallel Query Slaves

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`**

ORACLE

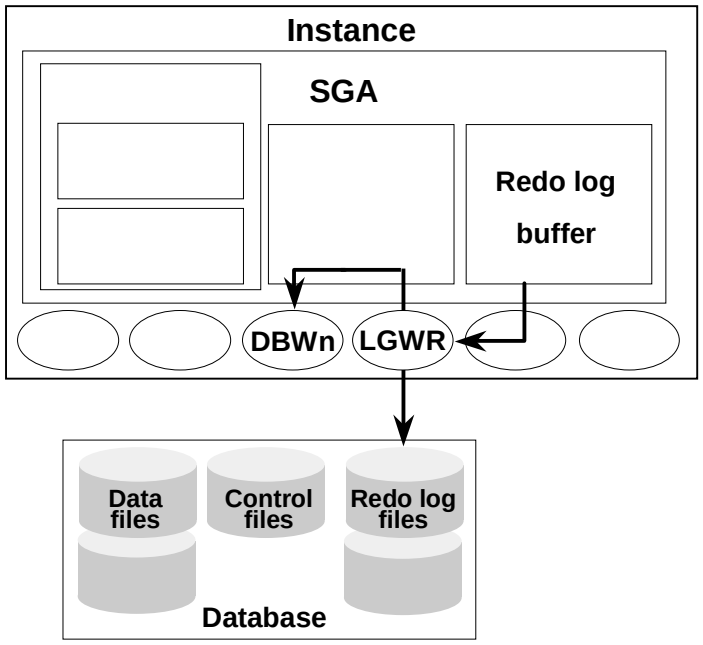
Database Writer

The server process records changes to rollback and data blocks in the buffer cache. Database Writer (DBWn) writes the dirty buffers from the database buffer cache to the data files. It ensures that a sufficient number of free buffers—buffers that can be overwritten when server processes need to read in blocks from the data files—are available in the database buffer cache. Database performance is improved because server processes make changes only in the buffer cache.

DBWn defers writing to the data files until one of the following events occurs:

- Incremental or normal checkpoint
- The number of dirty buffers reaches a threshold value
- A process scans a specified number of blocks when scanning for free buffers and cannot fine any.
- Timeout occurs.
- A ping request in Real Application Clusters environment.
- Placing a normal or temporary tablespace offline.
- Placing a tablespace in read only mode.
- Dropping or Truncating a table.
- `ALTER TABLESPACE` tablespace name `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**

ORACLE

LOG Writer

LGWR performs sequential writes from the redo log buffer cache to the redo log file under the following situations:

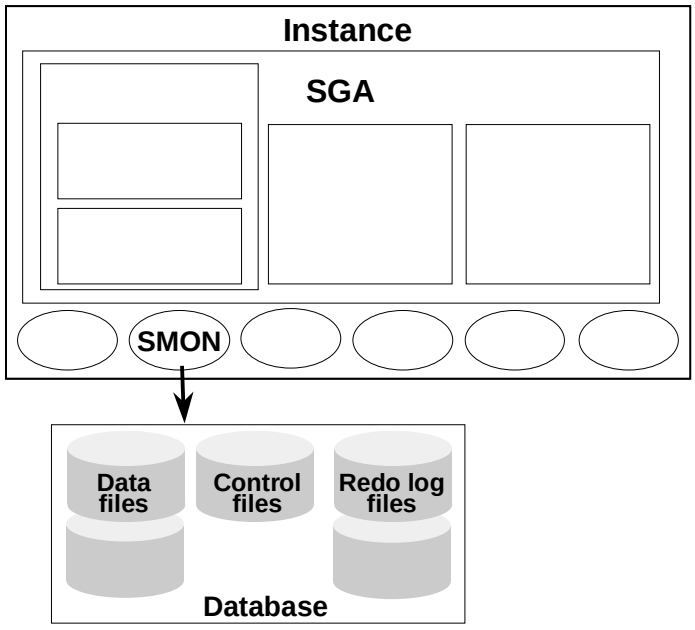
- When a transaction commits
- When the redo log buffer cache is one-third full
- When there is more than a megabyte of changes records in the redo log buffer cache
- Before DBWn writes modified blocks in the database buffer cache to the data files
- Every 3 seconds.

Because the redo is needed for recovery, LGWR confirms the commit only after the redo is written to disk.

LGWR can also call on DBWn to write to the data files.

Note: DBWn does not write to the online redo logs.

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**

ORACLE

System Monitor

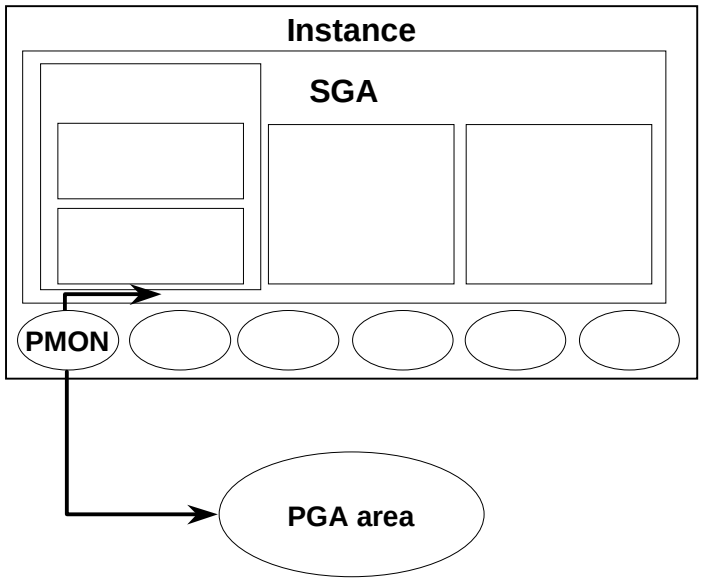
If the Oracle instance fails, any information in the SGA that has not been written to disk is lost. For example, the failure of the operating system causes an instance failure. After the loss of the instance, the background process SMON automatically performs instance recovery when the database is reopened. Instance recovery consists of the following steps:

1. Rolling forward to recover data that has not been recorded in the data files but that has been recorded in the online redo log. This data has not been written to disk because of the loss of the SGA during instance failure. During this process, SMON reads the redo log files and applies the changes recorded in the redo log to the data blocks. Because all committed transactions have been written to the redo logs, this process completely recovers these transactions.
2. Opening the database so that users can log on. Any data that is not locked by unrecovered transactions is immediately available.
3. Rolling back uncommitted transactions. They are rolled back by SMON or by the individual server processes as they access locked data.

SMON also performs some space maintenance functions:

- It combines, or coalesces, adjacent areas of free space in the data files.
- It deallocates temporary segments to return them as free space in data files. Temporary segments are used to store data during SQL statement processing.

Process Monitor (PMON)



Cleans up after failed processes by:

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

ORACLE

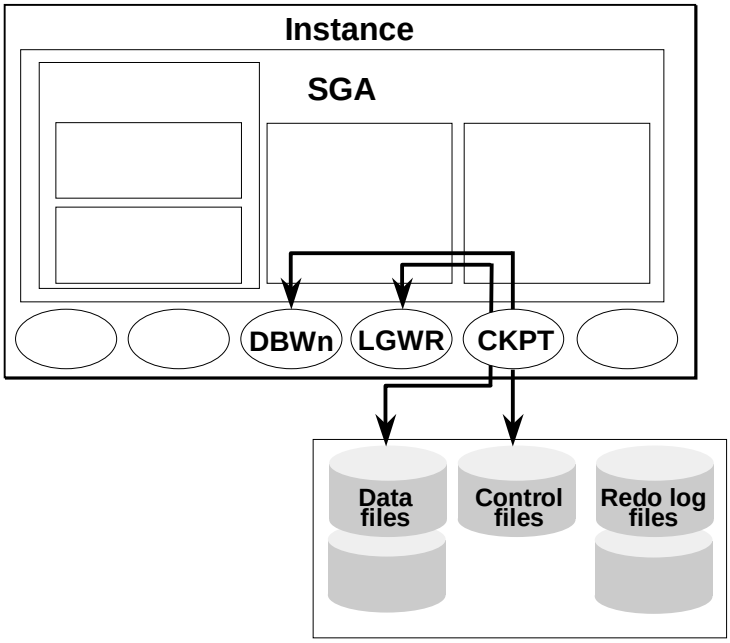
Process Monitor

The background process PMON cleans up after failed processes by:

- Rolling back the user's current transaction
- Releasing all currently held table or row locks
- Freeing other resources currently reserved by the user
- Restarts dead dispatchers

Dispatchers are covered in further detail in the *DBA Fundamentals II* course.

Checkpoint (CKPT)



Responsible for:

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

ORACLE

Checkpoint

An event called a checkpoint occurs when the Oracle background process DBWn writes all the modified database buffers in the SGA, including both committed and uncommitted data, to the data files.

Checkpoints are implemented for the following reasons:

- Checkpoints ensure that data blocks in memory that change frequently are written to data files regularly. Because of the least recently used algorithm of DBWn, a data block that changes frequently might never qualify as the least recently used block and thus might never be written to disk if checkpoints did not occur.
- Because all database changes up to the checkpoint have been recorded in the data files, redo log entries before the checkpoint no longer need to be applied to the data files if instance recovery is required. Therefore, checkpoints are useful because they can expedite instance recovery.

Checkpoint(continued)

At a checkpoint, the following information is written:

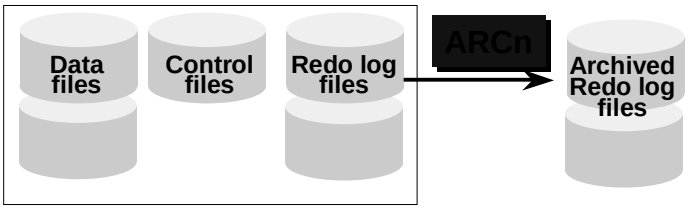
- Checkpoint number into the data file headers
- Checkpoint number, log sequence number, archived log names, and system change numbers into the control file.

CKPT does not write data blocks to disk or redo blocks to the online redo logs.

Note: The CKPT process should not be confused with the checkpoint operation discussed in the *Managing Redo Log Files* lesson.

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**



ORACLE

The Archiver Process

All other background processes are optional, depending on the configuration of the database; however, one of them, ARCn, is crucial to recovering a database after the loss of a disk. As online redo log files fill, the Oracle server begins writing to the next online redo log file. The process of switching from one redo log to another is called a log switch. The ARCn process initiates backing up, or archiving, of the filled log group at every log switch. It automatically archives the online redo log before the log can be reused, so that all of the changes made to the database are preserved. This enables the DBA to recover the database to the point of failure, even if a disk drive is damaged.

Archiving Redo Log Files

One of the important decisions that a DBA has to make is whether to configure the database to operate in `ARCHIVELOG` or in `NOARCHIVELOG` mode.

NOARCHIVELOG Mode: In `NOARCHIVELOG` mode, the online redo log files are overwritten each time a log switch occurs. LGWR does not overwrite a redo log group until the checkpoint for that group is complete. This ensures that committed data can be recovered if there is an instance crash. During the instance crash, only the SGA is lost. There is no loss of disks, only memory. For example, an operating system crash causes an instance crash.

Archiving Redo Log Files (continued)

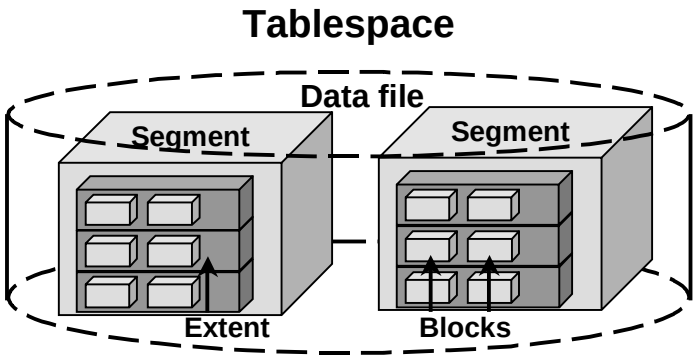
ARCHIVELOG Mode: If the database is configured to run in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived before they can be used again. Since changes made to the database are recorded in the online redo log files, the database administrator can use the physical backup of the data files and the archived online redo log files to recover the database without losing any committed data because of any single point of failure, including the loss of a disk. Usually, a production database is configured to run in ARCHIVELOG mode.

Archive log modes are covered in further detail in the *DBA Fundamentals II* course.

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.



ORACLE

Logical Structure

A logical structure hierarchy exists as follows:

- An Oracle database is a group of tablespaces.
- A tablespace may consist of one or more segments.
- A segment is made up of extents.
- An extent is made up of logical blocks.
- A block is the smallest unit for read and write operations.

Tablespaces, segments, extents, and blocks are discussed further in later lessons.

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.**

ORACLE

Processing a DML Statement

A data manipulation language (DML) statement requires only two phases of processing,

1. Parse is the same as the parse phase used for processing a query.
2. Execute requires additional processing to make data changes.

DML Execute Phase

To execute a DML statement:

1. If the data and rollback blocks are not already in the buffer cache, the server process reads them from the data files into the buffer cache.
2. The server process places locks on the rows that are to be modified. The rollback block is used to store the before-image of the data, so that the DML statements can be rolled back if necessary.
3. The data blocks record the new values of the data.

Processing a DML Statement (continued)

1. The server process records the before image to the undo block and updates the data block. Both of these changes are done in the database buffer cache. Any changed blocks in the buffer cache are marked as dirty buffers. That is, buffers that are not the same as the corresponding blocks on the disk.

   The processing of a `DELETE` or `INSERT` command uses similar steps. The before image for a `DELETE` contains the column values in the deleted row, and the before image of an `INSERT` contains the row location information.

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**

ORACLE

Practice 1 Overview

This practice covers the following topics:

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

ORACLE

Practice 1: Solutions

1. Which one of the following statements is true?
   - **a** An Oracle server is a collection of data consisting of three file types.
   - **b** A user establishes a connection with the database by starting an Oracle instance.
   - **c** A connection is a communication pathway between the Oracle Server and the Oracle Instance.
   - **d** A session starts when a user is validated by the Oracle server.
2. Which one of the following memory areas is not part of the SGA?
   - **a** Database buffer cache
   - **b** PGA
   - **c** Redo log buffer cache
   - **d** Shared Pool
3. Which two of the following statements are true about the Shared Pool?
   - **a** The shared Pool consists of the Library Cache, Data Dictionary Cache, Shared SQL area, Java Pool, and Large Pool.
   - **b** The Shared Pool is used to store the most recently executed SQL statements and the most recently used data.
   - **c** The Shared Pool is used for object that can be shared globally.
   - **d** The Library Cache consist of the Shared SQL and Shared PL/SQL areas.
4. Which one of the following memory areas is used to cache the data dictionary information?
   - **a** Database Buffer Cache
   - **b** PGA
   - **c** Redo log buffer cache
   - **d** Shared Pool
5. The primary purpose of the Redo Log Buffer Cache is to record all changes to the database data blocks.
   - **a** True
   - **b** False

   Answer: True
6. The PGA is a memory region that contains data and control information for multiple server processes or multiple background processes.
   - **a** True
   - **b** False

Practice 1: Solutions (continued)

7 Which one of the following processes is available when an Oracle instance is started.
- **a** User process
- **b** Background process
- **c** Server process
- **d** System process

8 Identify the six mandatory background processes.

_____
_____
_____
_____
_____
_____

9 Which one of the following memory areas is used to cache the data dictionary information?
- **a** Database Buffer Cache
- **b** PGA
- **c** Redo log buffer cache
- **d** Shared Pool

10 Match the process with its task.

a Database Writer	___ Assists with writing to the data file headers
b Log Writer	___ Responsible for instance recovery
c System Monitor	___ Cleans up after failed processes
d Process Monitor	___ Records database changes for recovery purposes
e Checkpoint	___ Writes dirty buffers to the data files

11 The physical structure of an Oracle database consists of control files, data files, and redo log files.
- **a** True
- **b** False

12 Place the following structures in order of hierarchy beginning with database.
- **a** Tablespaces
- **b** Extent
- **c** Segment
- **d** database
- **e** block

2

Getting Started With the Oracle Server

ORACLE

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**

ORACLE

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

Examples of Common Database Administration Tools

The tools listed will be covered in this course, but they are only a subset of the utilities supplied by Oracle.

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**

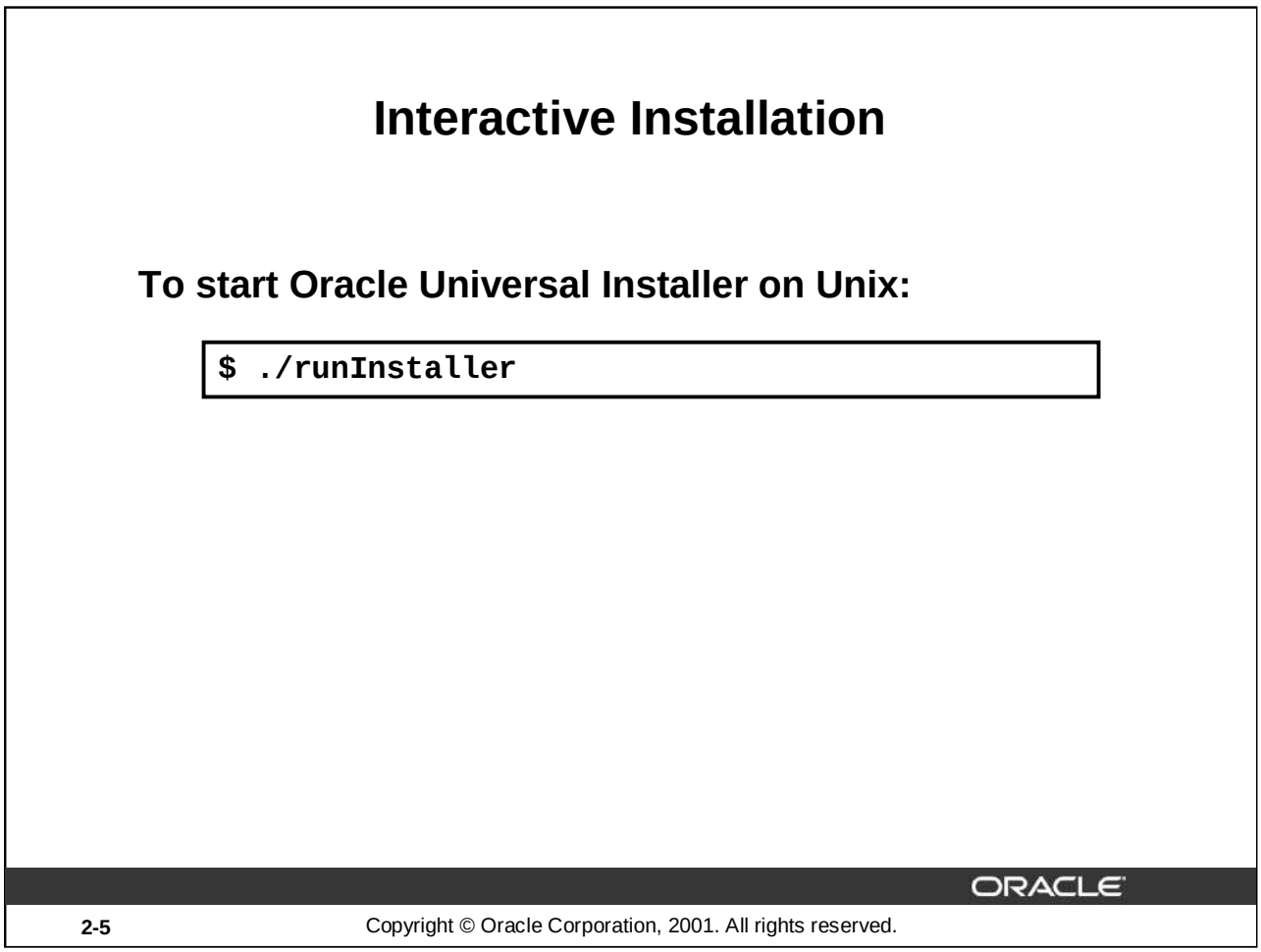
ORACLE

Features of the Oracle Universal Installer

The Java-based Oracle Universal Installer offers an installation solution for all Java-enabled platforms, allowing for a common installation flow and user experience independent of the platform.

The Universal Installer:

- Detects dependencies among components and performs an installation accordingly.
- Can be used to point to a URL where a release or staging area was defined and install software remotely over HTTP.
- Can be used to remove products installed. The deinstallation actions are the “undo” of installation actions.
- Maintains an inventory of all the Oracle homes on a target machine, their names, products, and version of products installed on them.
- Detects the language of the operating system and runs the installation session in that language.
- Can be run in interactive mode or silent mode. Oracle Universal Installer is run in silent (or non-interactive) mode using a response file.



Interactive Installation

Unix:

From the top directory of the CD-ROM

```
$ ./runInstaller
```

On Unix do not run the Installer as a root user.

Note: See your operating system specific Oracle documentation for information about installing Oracle Server on your platform.

Non-Interactive Installation Using Response Files

To start Universal Installer in non-interactive mode:

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

ORACLE

Non-Interactive Installation Using Response Files

Non-interactive installation is done when no user action is intended or if non-graphical terminals are used for installation.

Installation parameters are customized using a response file. Response file is a text file containing variables and values used by OUI during the installation process. Example of installation parameters include values for `ORACLE_HOME` and installation types (i.e., Typical or Custom Install).

The user first needs to edit the response file to specify the components to install. Response file templates are available in the `stage/response` directory on Unix platforms. Copy the response files and modify them to suit your environment.

```
./runInstaller -responsefile FILENAME [-silent] [-nowelcome]
```

Note: This is not character mode.

How to Launch the Oracle Universal Installer in Non-Interactive mode

Example : Launch the Oracle Universal Installer in non-interactive mode.

Unix

```
./runInstaller –responsefile FILENAME [-SILENT] [-NOWELCOME]
```

Where:

FILENAME :	Identifies the response file
SILENT:	Runs OUI in silent mode
NOWELCOME:	Does not display the Welcome window. If you use SILENT, this parameter is not necessary.

Sample Response File for Unix

```
[General]
RESPONSEFILE_VERSION=1.0.0.0.0
[Session]
UNIX_GROUP_NAME="dba"
FROM_LOCATION="/u01/stage/products.jar"
ORACLE_HOME="/u01/app/oracle/ora9i"
ORACLE_HOME_NAME="Ora9i"
TOPLEVEL_COMPONENT={"oracle.server", "9.0.1.0.0"}
SHOW_COMPONENT_LOCATIONS_PAGE=false
SHOW_SUMMARY_PAGE=false
SHOW_INSTALL_PROGRESS_PAGE=false
SHOW_REQUIRED_CONFIG_TOOL_PAGE=false
SHOW_OPTIONAL_CONFIG_TOOL_PAGE=false
SHOW_END_SESSION_PAGE=false
NEXT_SESSION=true
SHOW_SPLASH_SCREEN=true
SHOW_WELCOME_PAGE=false
SHOW_ROOTSH_CONFIRMATION=true
SHOW_EXIT_CONFIRMATION=true
INSTALL_TYPE="Typical”
s_GlobalDBName="u01.us.oracle.com"
s_mountPoint="/u01/app/oracle/ora9i/dbs"
s_dbSid="db09"
b_createDB=true
```

How to Launch the Oracle Universal Installer in Non-Interactive mode

Sample Response File (continued)

- The General section specifies the version number of the response file
- The Sessions section lists various dialogs of the Universal Installer. Some of the dialogs include:
  - `FROM LOCATION` specifies the location of the source of the products to be installed
  - `ORACLE_HOME`, value for `ORACLE_HOME`
  - `ORACLE_HOME_NAME`, value for `ORACLE_HOME_NAME`
  - `SHOW INSTALL PROGRESS` is the install progress page that appears during the install phase
  - `SHOW ROOTISH CONFIRMATION` is set to true if the confirmation dialog to run the root.sh script needs to be shown
  - `SHOW EXIT CONFIRMATION` is set to true if the confirmation when exiting the installer needs to be shown

Note: Refer to the Operating System specific Installation Guide for complete details on setting up a response file.

Oracle Database Configuration Assistant

The Oracle Database Configuration Assistant allows you to:

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

ORACLE

Oracle Database Configuration Assistant

Creating a database using Oracle Database Configuration Assistant is covered in lesson *Creating a Database*.

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**

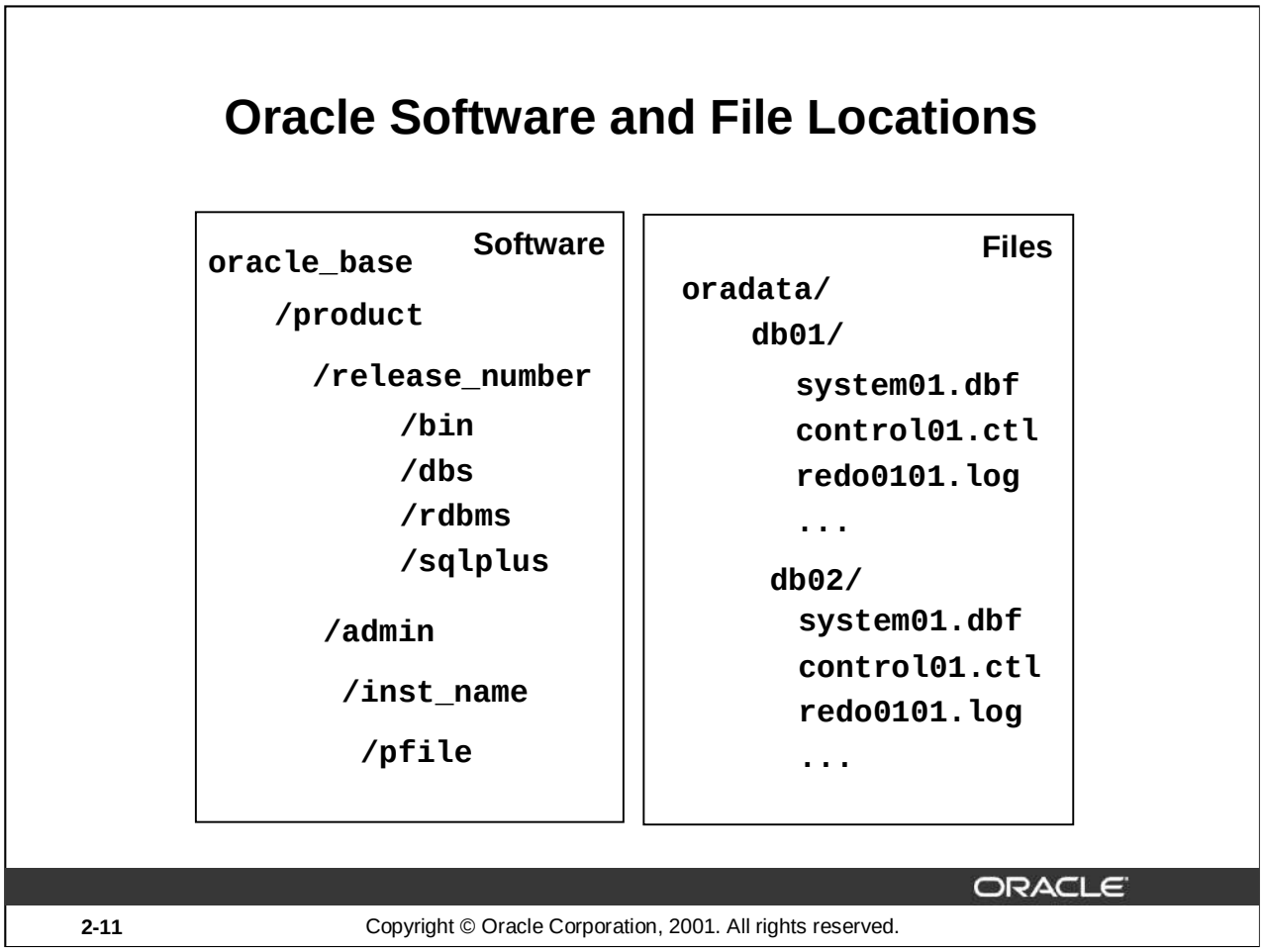
ORACLE

Optimal Flexible Architecture (OFA)

Installation and configuration on all supported platforms complies with Optimal Flexible Architecture (OFA). OFA organizes database files by type and usage. Binary files, control files, log files, and administrative files can be spread across multiple disks.

Consistent naming convention provides the following benefits:

- Database files can be easily differentiated from other files
- It is easy to identify control files, redo log files, and data files
- Easier administration of multiple Oracle Homes on the same machine by separating files on different disks and directories
- Better performance is achieved by decreasing disk contention among data files, binary files, and administrative files which can now reside on separate directories and disks



Oracle Software Locations

The directory tree above shows an example of OFA-complaint database.

Optimal Flexible Architecture

Another important issue during installation and creation of a database is organizing the file system so that it is easy to administer growth by adding data into an existing database, adding users, creating new databases, adding hardware and distributing I/O load across sufficiently many drives.

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**

ORACLE

The Database Administrator Users

Extra privileges are necessary to execute administrative duties on the Oracle server, such as creating users. Two database user accounts, SYS and SYSTEM, are created automatically with the database and granted the DBA role. That is, a predefined role that is created automatically with every database. The DBA role has all database system privileges.

SYS

When a database is created, the user SYS, identified initially by the password change_on_install, is created. SYS owns the vitally important data dictionary.

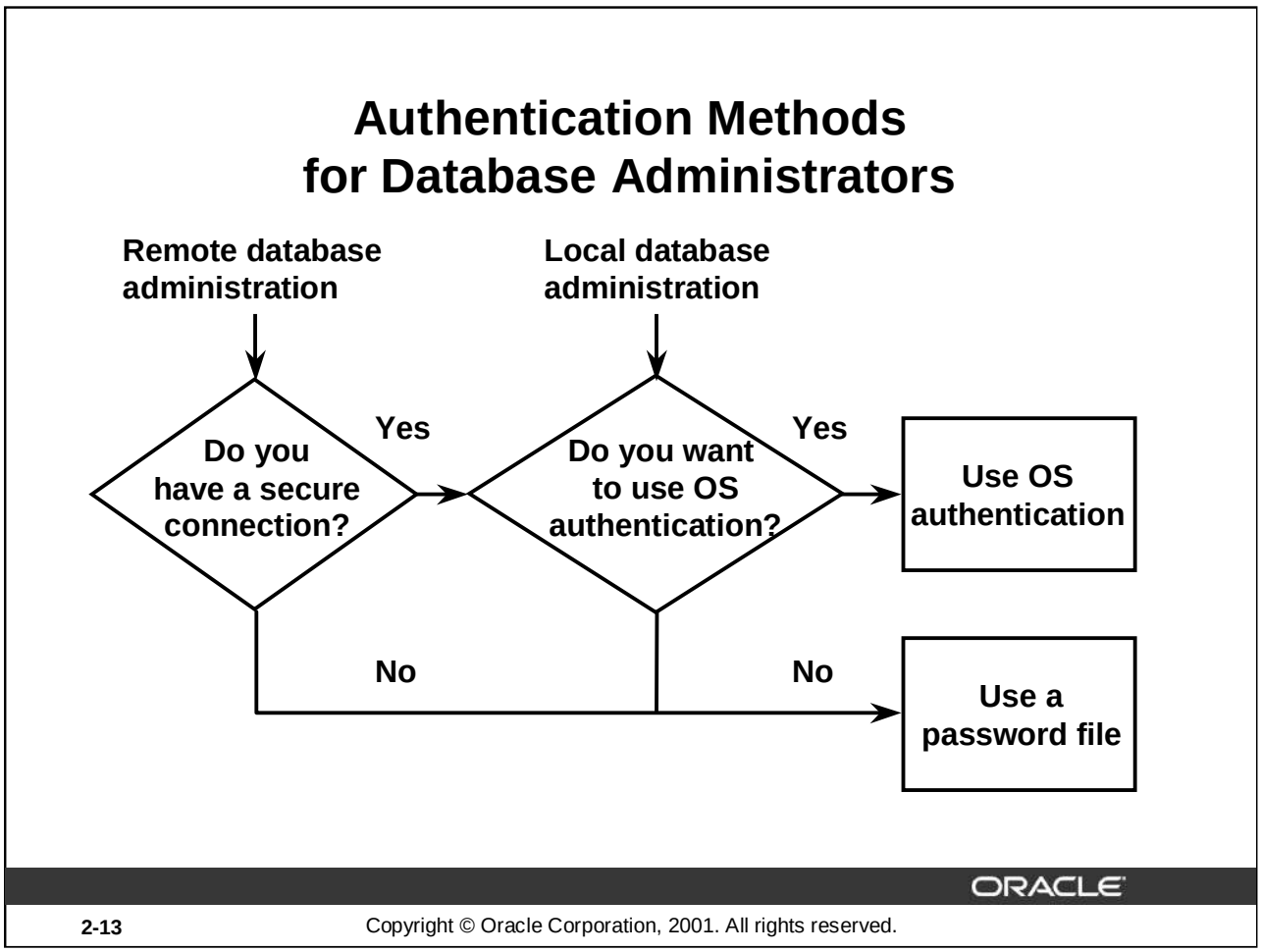
SYSTEM

When a database is created, the user SYSTEM, identified initially by the password manager, is also created automatically. Additional tables and views owned by the user SYSTEM are created. They contain administrative information used by Oracle tools.

Additional users may be created depending on the mode of database creation i.e., manually, or using Database Creation Assistant. You will want to create at least one additional administrator username to use when performing daily administrative tasks.

For security reasons, the default passwords of SYS and SYSTEM should be changed immediately after database creation.

Note: Refer to the *Managing Users* lesson for details regarding users.



Authentication Methods for Database Administrators

Depending on whether you want to administer your database locally on the same machine on which the database resides or to administer many different database servers from a single remote client, you can choose either operating system authentication or password files to authenticate database administrators.

Note: Refer to operating system specific manuals for operating system authentication.

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;
```

ORACLE

How to Use Password File Authentication

Oracle provides a password utility, `orapwd`, to create a password file. When you connect using `SYSDBA` privilege, you are connecting as `SYS` schema and not the schema associated with your username. For `SYSOPER`, you are connected to the `PUBLIC` schema.

Access to the database using the password file is provided by special `GRANT` commands issued by privileged users.

Note: Refer to the *Managing Privileges* lesson for information on grants.

How to Use Password File Authentication (continued)

Using a Password File

1. Create the password file using the password utility `orapwd`.

   ```
   orapwd file=filename password=password entries=max_users
   ```

   where:

`filename`	is the name of the password file
`password`	is the password for `SYSOPER` and `SYSDBA`
`entries`	is the maximum number of distinct users allowed to connect as `SYSDBA` or `SYSOPER`. If you exceed this number, you must create a new password file. It is safer to have a larger number.

2. Set the `REMOTE_LOGIN_PASSWORDFILE` parameter to `EXCLUSIVE` or `SHARED`

   where:

`EXCLUSIVE`	indicates that only one instance can use the password file and that the password file contains names other than `SYS`. Using an `EXCLUSIVE` password file you can grant `SYSDBA` or `SYSOPER` privileges to individual users.
`SHARED`	indicates that more than one instance can use the password file. The only user recognized by the password file is `SYS`. You cannot add users to the `SHARED` password file.

3. Connect to the database as follows:

   ```
   CONNECT sys/admin AS SYSDBA
   ```

Note: The password files are located in the `$ORACLE_HOME/dbs` directory on Unix

Maintaining the Password File

Delete the existing password file and create a new password file by using the Oracle password utility.

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

SQL*Plus

SQL*Plus is Oracle's command line tool to run the standard SQL (Structured Query Language) suite of commands. SQL is a functional language that users use to communicate with Oracle to retrieve, add, update, or modify data in the database.

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

Oracle Enterprise Manager

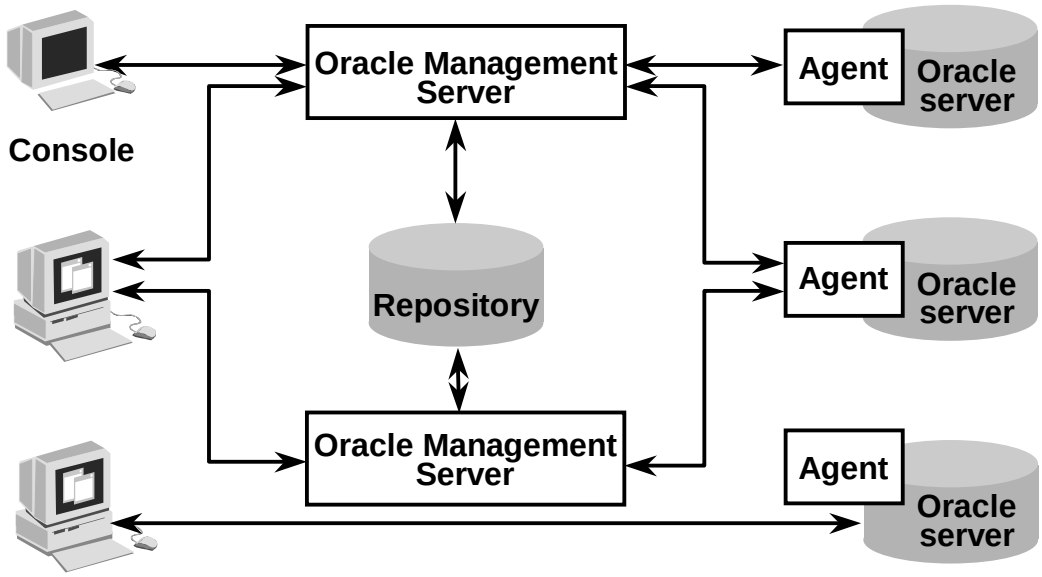
The Oracle Enterprise Manager is a unified management framework consisting of a Java-based Console, a suite of tools and services, and a network of management servers and intelligent agents. It includes both hierarchical tree and graphical representations of the objects and their relationships in the system.

The common services, which include Job Scheduling System, Event Management System, Repository, Service Discovery, Security, Paging/Email Blackouts complete the Oracle Enterprise Manager framework.

Oracle Enterprise Manager includes integrated applications that allows you to perform both routine and advanced administration tasks. They include the optional packs such as the Diagnostics Pack, Tuning Pack and Change Management Pack, and other applications such as Oracle Applications Manager.

Note: Optional packs require a separate license.

Oracle Enterprise Manager Architecture: Java-Based Console and Applications



ORACLE

Oracle Enterprise Manager Architecture

Oracle Enterprise Manager is based on a highly scalable three tier model architecture.

Console

The first tier is comprised of clients such as consoles and management applications, which present graphical user interfaces to administrators for all management tasks. The first tier depends on the second tier Oracle Management Server for the bulk of their application logic.

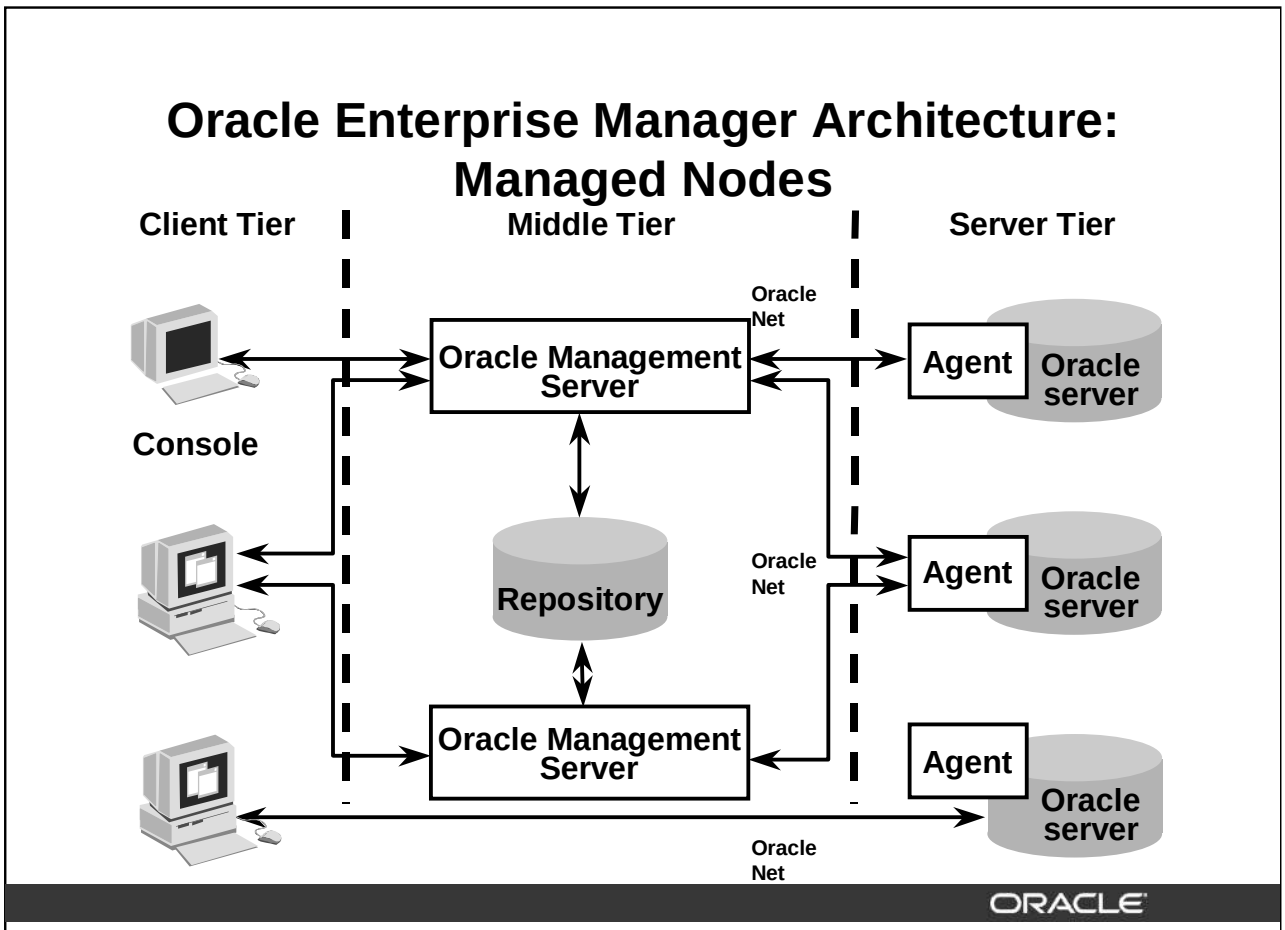
Oracle Management Server

The second tier component of Oracle Enterprise Manager is the Oracle Management Server (OMS). The OMS is the core of the Enterprise Manager framework and provides administrative user accounts, processes functions such as jobs, events and manages the flow of information between the Console (first tier) and the managed nodes (third tier).

The OMS uses a repository to store all system data, application data, information about the state of the managed nodes and information about any system managed packs.

Oracle Enterprise Manager Repository

The Repository is a set of tables, and is created when you set up the OMS. The OMS uses the Oracle Enterprise Manager repository as its persistent back-end store. If necessary, more than one OMS can be used. Multiple OMSs share a repository and provide reliability and fault tolerance.



Nodes

The third tier is comprised of managed nodes which contains targets such as databases and other managed services. Residing on each node is an Oracle Intelligent Agent, which communicates with the OMSs and performs tasks sent by consoles and client applications.

Only one Intelligent Agent is required per node.

Intelligent Agent functions independently of the database as well as the Console and Management Server. By running independently of other components, Intelligent Agents can perform such tasks as starting up and shutting down a database and staying operational if another part of the system is down.

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**

ORACLE

Console Features

The Console provides a graphical interface for administrators and a central launching point for all applications. Applications such as Instance Manager, Schema Manager, Storage Manager, SQL*Plus Worksheet can be launched from the Console.

The Console can be run either in thin mode via the web or as a fat client. Thin clients use a web browser to connect to a server where Console files are installed, whereas fat clients require Console files to be installed locally.

The Console can be launched either in standalone mode or by connecting to Oracle Management Server.

Note: Oracle Management Server requires a repository. For details on configuring the Oracle Management Server and Repository refer to the *Oracle Enterprise Manager* course.

How to Launch the Oracle Enterprise Manager Console

Example: Launch the console and add a database to it.

1. Launch the console:
   % `oemapp console`
2. At the Login dialog box choose to launch the application in standalone mode
3. Add Databases to tree: The Add Database to Tree dialog allows you to either add a database manually by supplying the Hostname, Port number, SID and Net Service name or add selected databases from your local tnsnames.ora file.
   - Select `Add Database To Tree` from the `Navigator` menu
   - Enter values for Hostname, SID, Port number and Net Service name
4. Expand your working database from the databases folder
5. Right click your working database and select `Connect`
6. Enter the username, password and service name for the database and click `OK`.

Note:

- The Add Database to Tree dialog appears automatically when you launch the Console in standalone mode for the first time. It is also available from the Navigator menu
- The Console can also be launched from the Windows NT Start menu

DBA Tools

Standard applications that can be launched from the Console:

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

ORACLE

The DBA Tools

The standard applications that are supplied with Oracle Enterprise Manager include the following:

- Instance Manager: Performs startup, shutdown and monitor databases
- Security Manager: Used to manage users and privileges
- Storage Manager: Maintains tablespaces, data files, rollback segments and log groups
- Schema Manager: Used to create and maintain objects such as tables, indexes, and views
- SQL*Plus Worksheet: Provides the capability to issue SQL statements against any database

How to Launch DBA Tools

- Launch the Console in standalone mode
- Expand the databases folder and expand the relevant database
- Select tools such as Instance Manager, Schema Manager

The DBA Tools (continued)

Note:

- The use of these applications to perform specific tasks is covered in the other lessons throughout the course.
- The use of backup management tools and wizards to back up, restore, and recover databases are discussed in the *DBA Fundamentals II* course.

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**

ORACLE

Practice 2 Overview

This practice covers the following topics:

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

ORACLE

Practice 2: Getting Started With the Oracle Server

1. List the common database administration tools available to a DBA.

   _____
   _____
   _____
   _____
   _____

2. Connect to SQL*Plus as `SYS AS SYSDBA` and start the database.

3. Select columns `OWNER`, `TABLE_NAME`, and `TABLESPACE_NAME` from data dictionary view `DBA_TABLES`, and format the output. (The display of the first ten rows is sufficient.) Exit SQL*Plus.

4. Start SQL*Plus and run a script named `lab02_03.sql`, which spools all initialization parameters to the output file `para.lst`.

5. Remove the current password file located in `$HOME/ADMIN/PFILE`. Create a new password file using the following information:

   - Password for sys: `oracle`
   - Enable five privileged users
   - Location `$HOME/ADMIN/PFILE`

   Ensure that Oracle can write to this file by changing permissions on the file:

   ```
   chmod 660 $HOME/ADMIN/PFILE/orapw$ORACLE_SID
   ```

   Hint: Use the `orapwd` utility to create the password file.

   Note: The password file is normally created in `$ORACLE_HOME/dbs`, owned by the owner of `$ORACLE_HOME`, and the permissions are not changed.

3

Managing an Oracle Instance

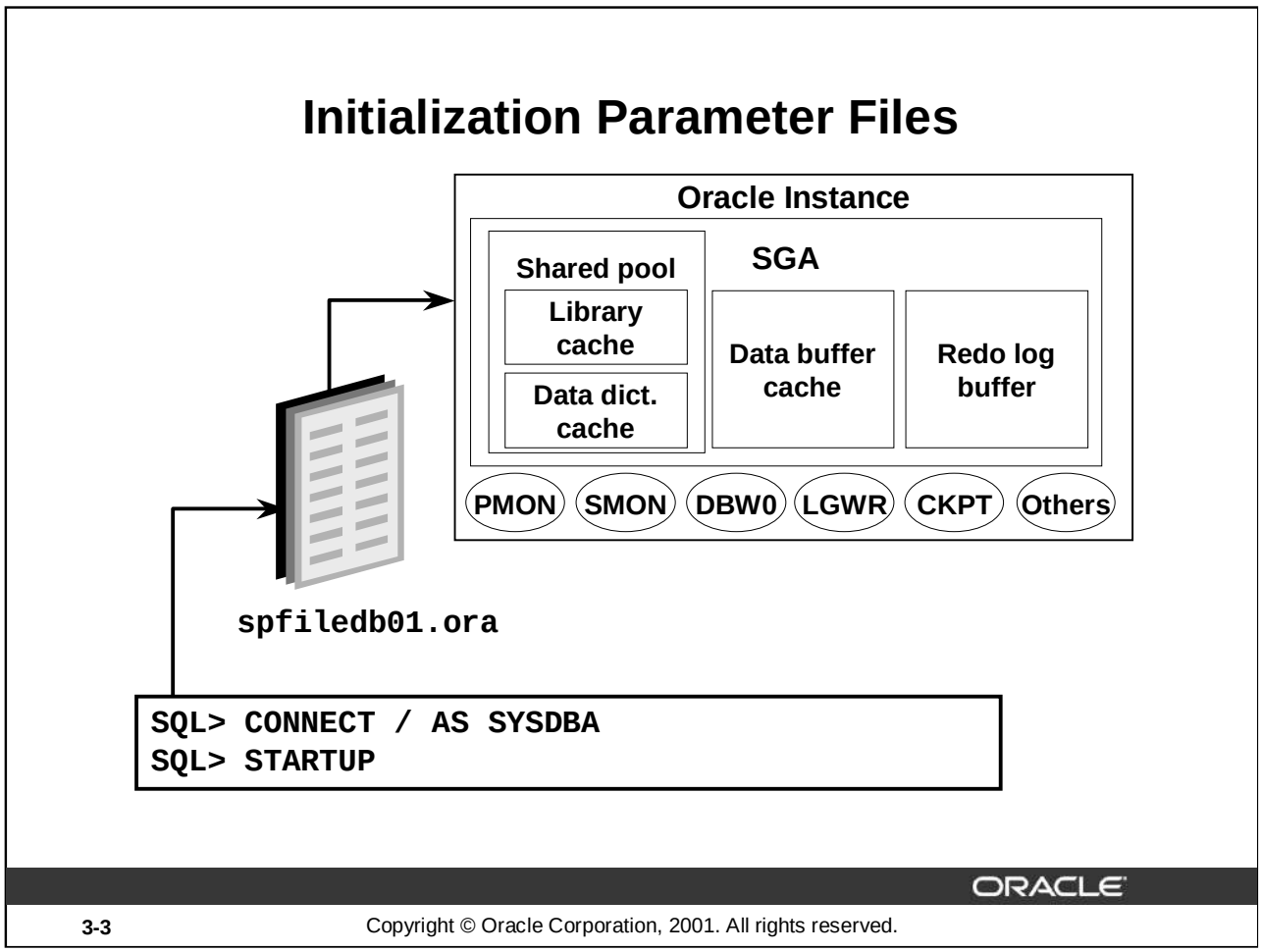
ORACLE

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**

ORACLE



Initialization Parameter Files

To start an instance, the Oracle server must read the initialization parameter file.

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`**

ORACLE

Initialization Parameter Files

To start an instance, the Oracle server reads the initialization parameter file. Two types of initialization parameter files exist:

- Static parameter file, `PFILE`, commonly referred to as `initSID.ora`
- Persistent parameter file, `SPFILE`, commonly referred to as `spfileSID.ora`

Parameter File Contents

- A list of instance parameters
- The name of the database the instance is associated with
- Allocations for memory structures of the System Global Area (SGA)
- What to do with filled online redo log files
- The names and locations of control files
- Information on undo segments

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

The PFILE is a text file that can be maintained using a standard operating system editor. The parameter file is read only during instance startup. If the file is modified, the instance must be shut down and restarted in order to make the new parameter values effective. Some parameters are dynamic, which means that they can be modified while the instance is running. Changes to dynamic parameters are not reflected in the PFILE.

By default, the PFILE is located in the $ORACLE_HOME/dbs directory on a Unix machine and named initSID.ora. A sample init.ora file is created by the Universal Installer during installation. This sample init.ora file can be used to create an instance-specific initSID.ora.

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
. . .
```

ORACLE

3-6

Rules for Specifying Parameters

- Specify the values in the following format: keyword=value.
- All parameters are optional.
- The server has a default value for each parameter. This value may be operating system dependent, depending on the parameter.
- Parameters can be specified in any order.
- Comment lines begin with the # symbol.
- Enclose parameters in double quotation marks to include character literals.
- Additional files can be included with the keyword `IFILE`.
- If case is significant for the operating system, then it is also significant in filenames.
- Multiple values are enclosed in parentheses and separated by commas.

Note: Develop a standard for listing parameters; either list them alphabetically or group them by functionality. The `PFILE` varies from instance to instance and not does necessarily look like the preceding example.

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';
```

ORACLE

SPFILE

SPFILE, new to Oracle9*i*, is a binary file. The file is not meant to be modified manually and must always reside on the server side. By default, the file is located in $ORACLE_HOME/dbs and has a default name in the format of spfileSID.ora. Once the file is created it is maintained by the Oracle server. The SPFILE provides the ability to make changes to the database persistent across shutdown and startup.

The ALTER SYSTEM command is used to change the value of instance parameters. The SCOPE setting determines the scope of the change.

- MEMORY: Changes the parameter value only in the currently running instance
- SPFILE: Changes the parameter value in the SPFILE only
- BOTH: Changes the parameter value in the currently running instance and the SPFILE

```
ALTER SYSTEM SET parameter = value [SCOPE =
MEMORY|SPFILE|BOTH]
```

The SPFILE can be modified with the ALTER SYSTEM command when the instance starts using the SPFILE or using the PFILE with the parameter SPFILE.

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;
```

ORACLE

Creating an SPFILE

An SPFILE is created from an initSID.ora file using the CREATE SPFILE command. This can be executed before or after the database is open.

```
CREATE SPFILE [='SPFILE-NAME']
FROM PFILE[='PFILE-NAME']
```

Where:

- SPFILE-NAME: Name of the SPFILE to be created: if not specified, the default SPFILE name is assumed.
- PFILE-NAME: Name of a PFILE, must be available on the server side.

The CREATE SPFILE command requires the SYSDBA role to execute.

The syntax can be reversed to create a PFILE from an SPFILE.

Creating an SPFILE (continued)

How to Modify the SPFILE Configuration

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the databases folder
4. Expand Instance folder and click Configuration
5. In the General tab, click All Initialization Parameters
6. Modify a parameter in the value column.
5. Click OK.

Note: The Console can also be launched from the Windows NT Start menu.

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

Rules Applying to SPFILE

The comments specified on the same lines as a parameter setting in the PFILE are maintained in the SPFILE. All other comments are ignored.

Although the text of an SPFILE is easily viewed in Unix the SPFILE is binary.

Rules Applying to SPFILE (continued)

Parameters That Should Be Specified

Parameter	Description
BACKGROUND_DUMP_DEST	Location where background process trace files are written (LGWR, DBW0, and so on). This is also the location for the alert log file.
COMPATIBLE	Version of the server with which this instance should be compatible.
CONTROL_FILES	Names of the control files
DB_BLOCK_BUFFERS	Number of blocks cached in the SGA
DB_NAME	Database identifier of eight characters or fewer. This is the only parameter that is required when a new database is created.
SHARED_POOL_SIZE	Size in bytes of the shared pool
USER_DUMP_DEST	Location where user debugging trace files are created on behalf of a user process

Technical Note

The default values depend on the version of the Oracle server.

Commonly Modified Parameters

Parameter	Description
IFILE	Name of another parameter file to be embedded within the current parameter file. Up to three levels of nesting is possible.
LOG_BUFFER	Number of bytes allocated to the redo log buffer in the SGA
MAX_DUMP_FILE_SIZE	Maximum size of the trace files, specified as number of operating system blocks
PROCESSES	Maximum number of operating system processes that can connect simultaneously to this instance
SQL_TRACE	Enables or disables the SQL trace facility for every user session
TIMED_STATISTICS	Enables or disables timing in trace files and in monitor screens

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

Oracle Managed Files

Oracle Managed Files (OMF) simplify file administration by eliminating the need to directly manage the files within an Oracle database.

For example: On Solaris, OMF files are named as follows:

- Control files: `ora_%u.ctl`
- Redo log files: `ora_%g_%u.log`
- Data files: `ora_%t_%u.dbf`
- Temporary data files: `ora_%t_%u.tmp`

The following characters are defined as:

- `%u` is an 8-character string that guarantees uniqueness.
- `%t` is the tablespace name, truncated if necessary to fit into the maximum length file name. Placing the tablespace name before the uniqueness string means that all the data files for a tablespace appear next to each other in an alphabetic file listing.
- `%g` is the redo log file group number.
- `ora_` identifies the file as an Oracle Managed File.

Undo files do not have a special extension. As with temporary files, they are considered to be just like any other data files.

Oracle Managed Files (continued)

The parameters DB_CREATE_FILE_DEST and DB_CREATE_ONLINE_LOG_DEST_N do not both have to be set, one or the other or both can be used.

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.**

ORACLE

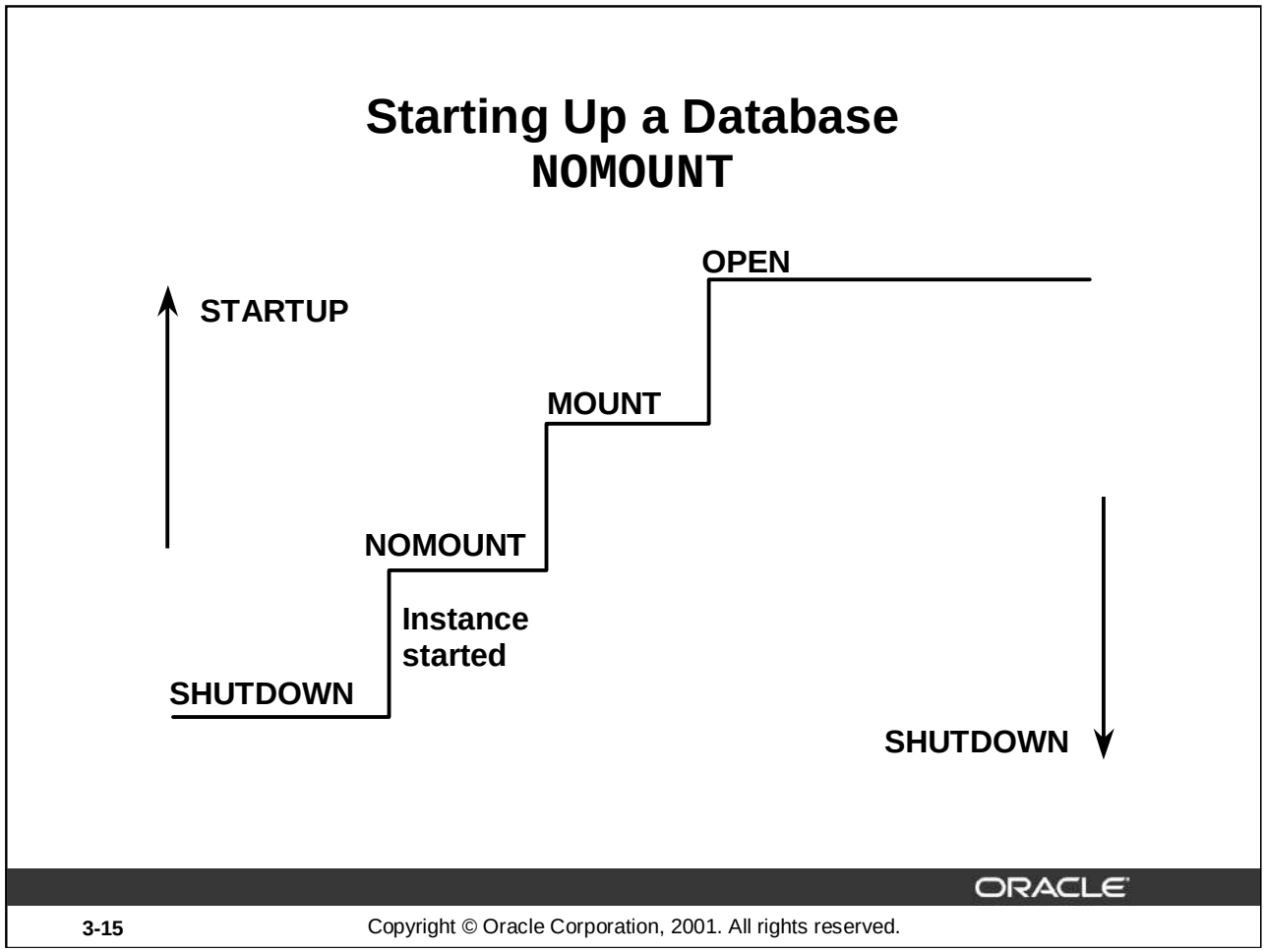
OMF Example

The DB_CREATE_FILE_DEST parameter sets the default file system directory for the data files (/u01/oradata).

The DB_CREATE_ONLINE_LOG_DEST_1 and DB_CREATE_ONLINE_LOG_DEST_2 parameters set the default file system directories for online redo log and control file creation. Each online redo log and control file is multiplexed across the two directories, /u02/oradata and /u03/oradata.

After the initialization parameters are set, the database can be created using the CREATE DATABASE command. If the command fails, any OMF files created are removed. The internally generated file names can be seen when the user selects from DBA_DATAFILES and V$LOGFILE.

Both DB_CREATE_FILE_DEST and DB_CREATE_ONLINE_LOG_DEST_N can be modified dynamically with the ALTER SYSTEM SET command.



Starting Up a Database

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

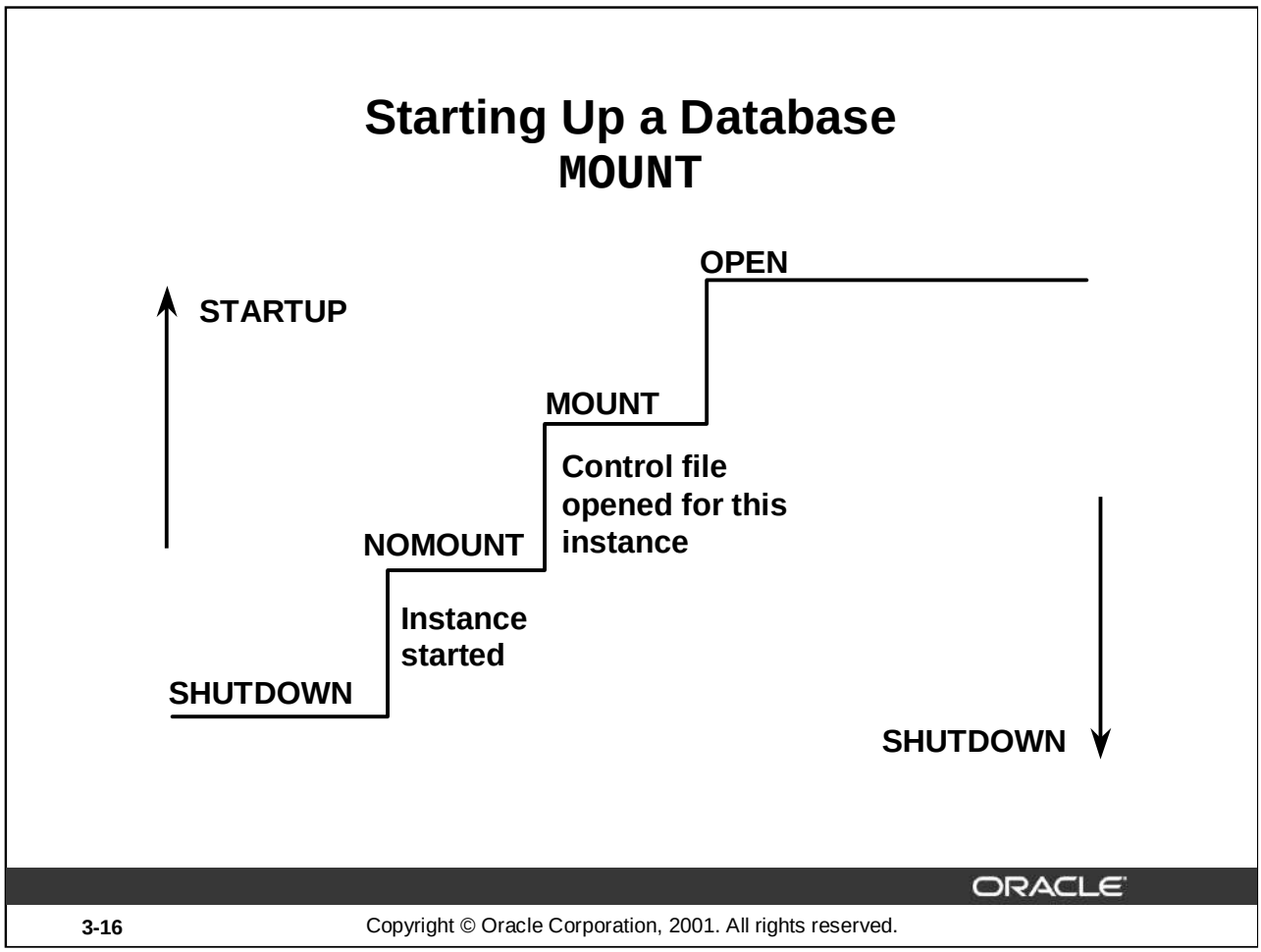
Starting the Instance (`NOMOUNT`)

Usually you would start an instance without mounting a database only during database creation or the re-creation of control files.

Starting an instance includes the following tasks:

- Reading the initialization file from `$ORACLE_HOME/dbs` in the following order:
  - First `spfileSID.ora`. If not found then
  - `spfile.ora`
  - `initSID.ora`
- Specifying the `PFILE` parameter with `STARTUP` overrides the default behavior
- Allocating the SGA
- Starting the background processes
- Opening the `alertSID.log` file and the trace files

The database must be named with the `DB_NAME` parameter either in the initialization file or in the `STARTUP` command.



Starting Up a Database

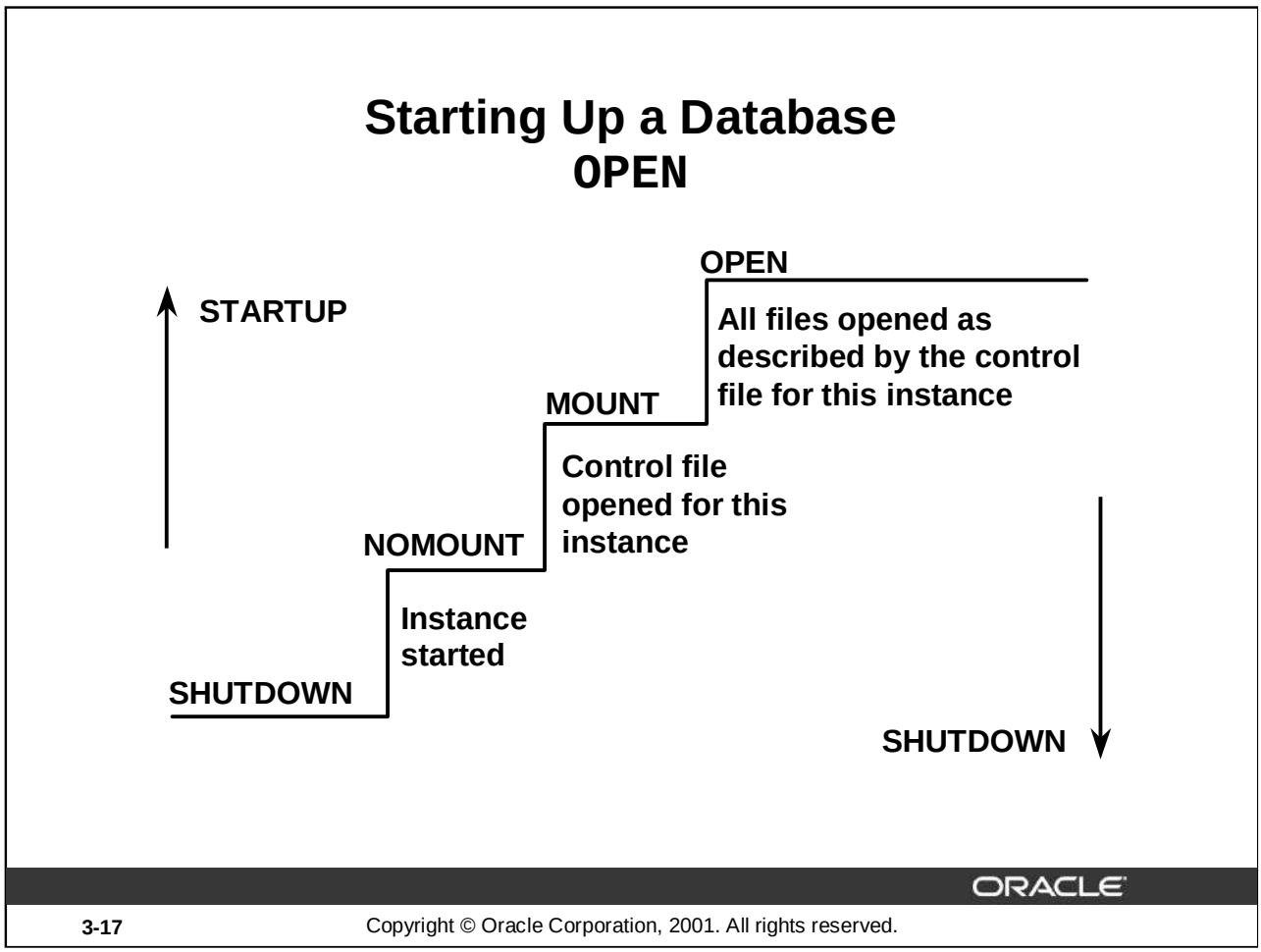
Mounting the Database To perform specific maintenance operations, you start an instance and mount a database but do not open the database.

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

- Renaming data files
- Enabling and disabling redo log archiving options
- Performing full database recovery

Mounting a database includes the following tasks:

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



Starting Up a Database

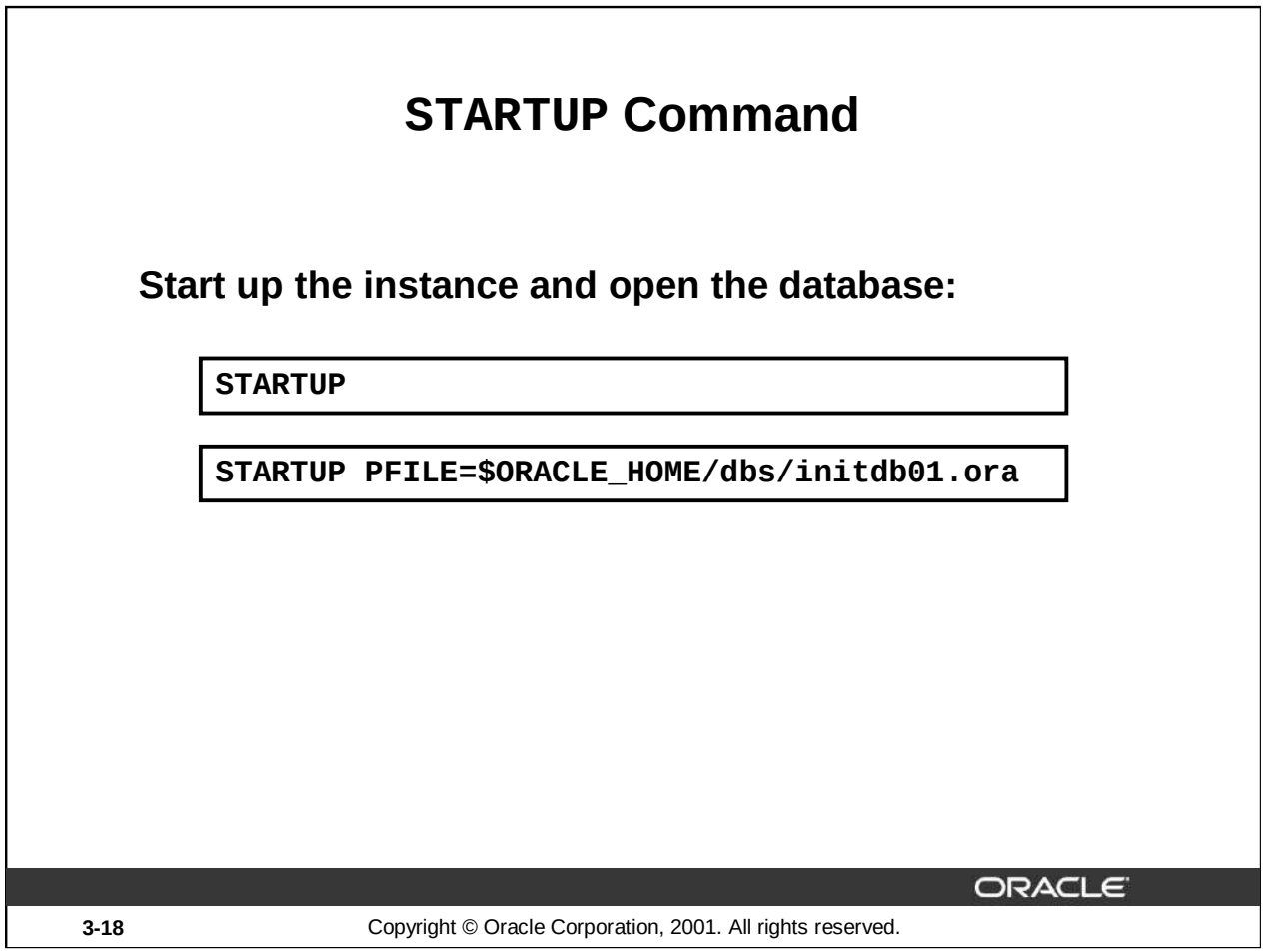
Opening the Database Normal database operation means that an instance is started and the database is mounted and open; with normal database operation, any valid user can connect to the database and perform typical data access operations.

Opening the database includes the following tasks:

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

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

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



Starting Up

To start up an instance, use the following command:

```
STARTUP [FORCE] [RESTRICT] [PFILE=filename]
        [OPEN [RECOVER][database]
        |MOUNT
        |NOMOUNT]
```

Note: This is not the complete syntax.

where:

OPEN	enables users to access the database
MOUNT	mounts the database for certain DBA activities but does not provide user access to the database
NOMOUNT	creates the SGA and starts up the background processes but does not provide access to the database
PFILE=*parfile*	enables a nondefault parameter file to be used to configure the instance

Starting Up (continued)

`FORCE`	aborts the running instance before performing a normal startup
`RESTRICT`	enables only users with `RESTRICTED SESSION` privilege to access the database
`RECOVER`	begins media recovery when the database starts

Initialization Parameter File

By default, when the database is started, the Oracle server looks in `$ORACLE_HOME/dbs` for an initialization parameter file. The Oracle server first attempts to read file `spfile`*`sid`*`.ora` and if not found then file `init`*`sid`*`.ora`. If neither file is located, an error message is received.

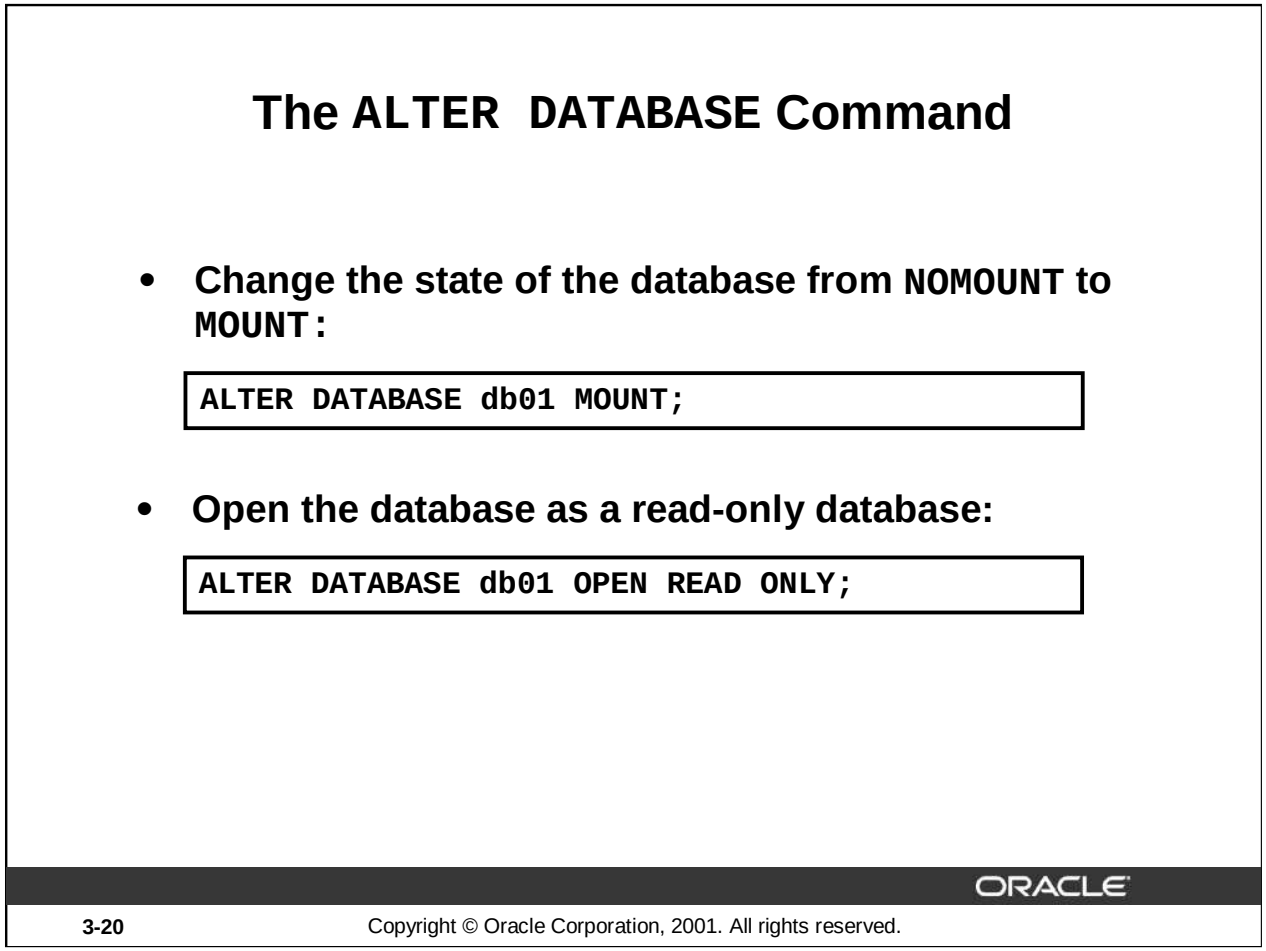
Automating Database Startup

On Unix Automating database startup and shutdown can be controlled by the entries in a special operating system file; for example, `oratab` in the `/var/opt/oracle` directory.
Note: Refer to the installation guide for your operating system for more information.

Troubleshooting

Attempting to start the Oracle Utilities without starting these services results in the following error message:

ORA-12547: TNS: lost contact



Changing the Status of the Database

To open the database from STARTUP NOMOUNT to a MOUNT stage or from MOUNT to an OPEN stage, use the ALTER DATABASE command:

```
ALTER DATABASE { MOUNT | OPEN }
```

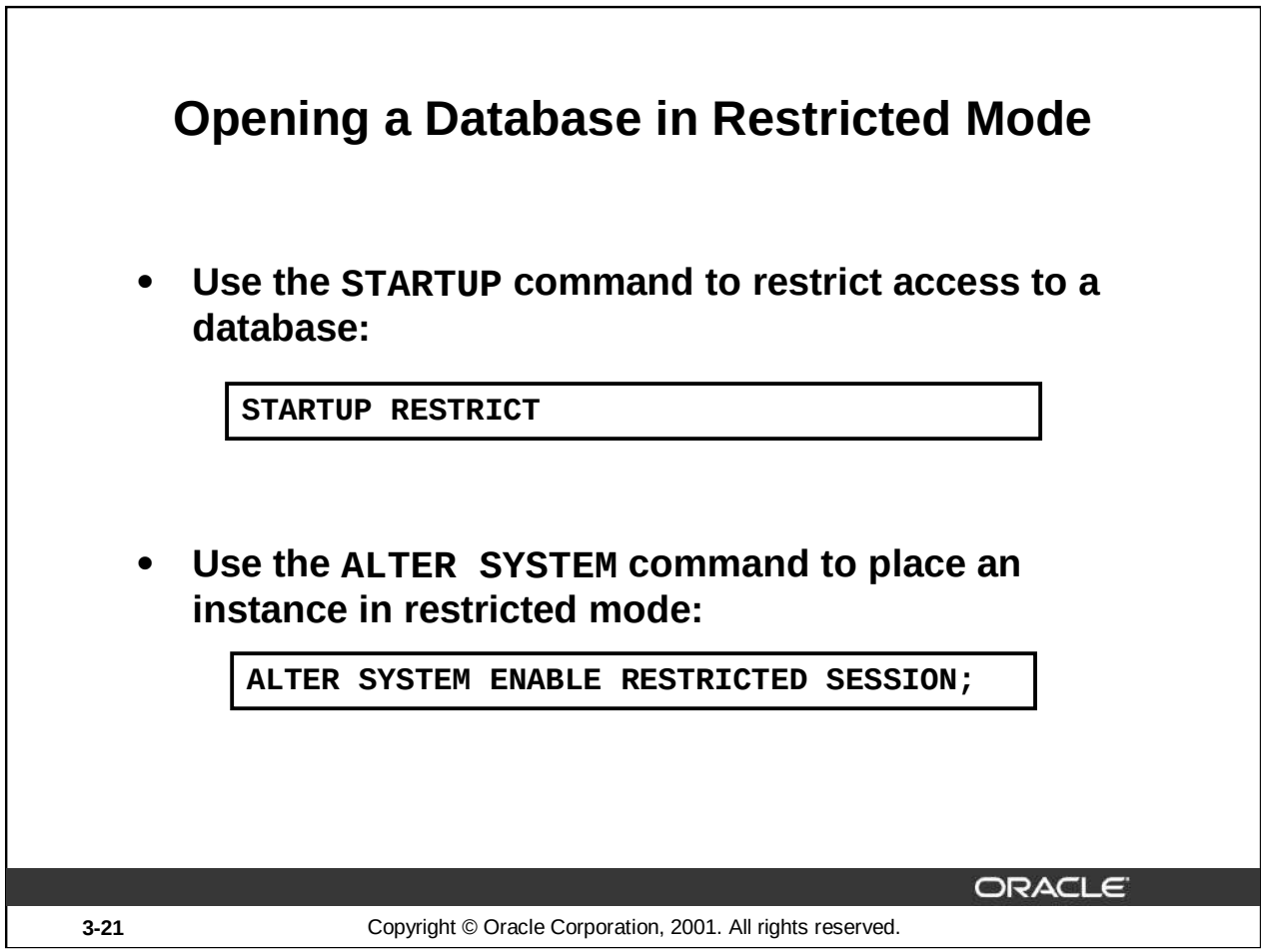
To prevent data from being modified by user transactions, the database can be opened in read-only mode.

To start up an instance, use the following command:

```
ALTER DATABASE OPEN [READ WRITE| READ ONLY]
```

where:

READ WRITE	opens the database in read-write mode, so that users can generate redo logs
READ ONLY	restricts users to read-only transactions, preventing them from generating redo log information



Opening a Database in Restricted Mode

A restricted session is useful, for example, when you perform structure maintenance or a database export and import. The database can be started in restricted mode so that it is available only to users with the RESTRICTED SESSION privilege.

The database can also be put in restricted mode by using the ALTER SYSTEM SQL command:

```
ALTER SYSTEM [ {ENABLE|DISABLE} RESTRICTED SESSION ]
```

where:

ENABLE RESTRICTED SESSION:	enables future logins only for users who have the RESTRICTED SESSION privilege
DISABLE RESTRICTED SESSION:	disables RESTRICTED SESSION so that users who do not have the privilege can log on

Terminate Sessions

After placing an instance in restricted mode, you may want to kill all current user sessions before performing administrative tasks. This can be done by the following:

```
ALTER SYSTEM KILL SESSION 'integer1,integer2'
```

where:

integer1: is the value of the SID column in the V$SESSION view

integer2: is the value of the SERIAL# column in the V$SESSION view

Opening a Database in Restricted Mode (continued)

Note: The session ID and serial number are used to uniquely identify a session. This guarantees that the `ALTER SYSTEM KILL SESSION` command is applied to the correct session even if the user logs off and a new session uses the same session ID.

Effects of Terminating a Session

The `ALTER SYSTEM KILL SESSION` command causes the background process PMON to perform the following steps upon execution:

- Roll back the user's current transaction
- Release all currently held table or row locks
- Free all resources currently reserved by the user

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**

ORACLE

Read-Only Database Features

A database can be opened as read-only, as long as it is not already open in read-write mode. The feature is especially useful for a standby database to offload query processing from the production database.

If a query needs to use a temporary tablespace—for example, to do disk sorts—the current user must have a locally managed tablespace assigned as the default temporary tablespace; otherwise, the query fails. For user SYS, a locally managed tablespace is required.

Note: Locally managed tablespaces are discussed in a later lesson.

Read-only mode does not restrict database recovery or operations that change the database state without generating redo data. For example, in read-only mode:

- Data files can be taken offline and online.
- Recovery of offline data files and tablespaces can be performed.

Disk writes to other files, such as control files, operating system audit trails, trace files, and alert log files, can continue in read-only mode.

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

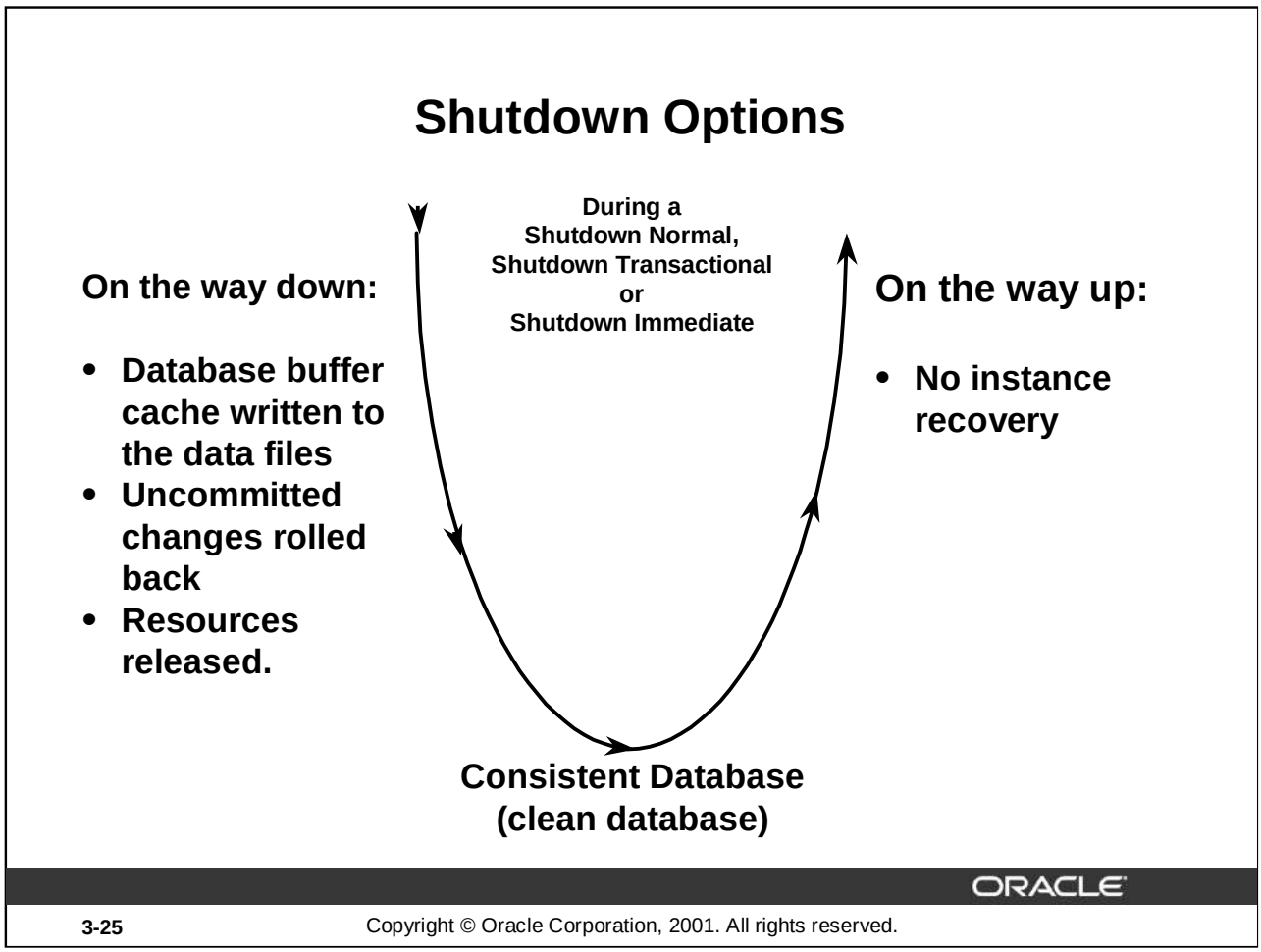
ORACLE

Shutting Down the Database

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

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

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



Shutdown Options

Shutdown Normal

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

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

Shutdown Transactional

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

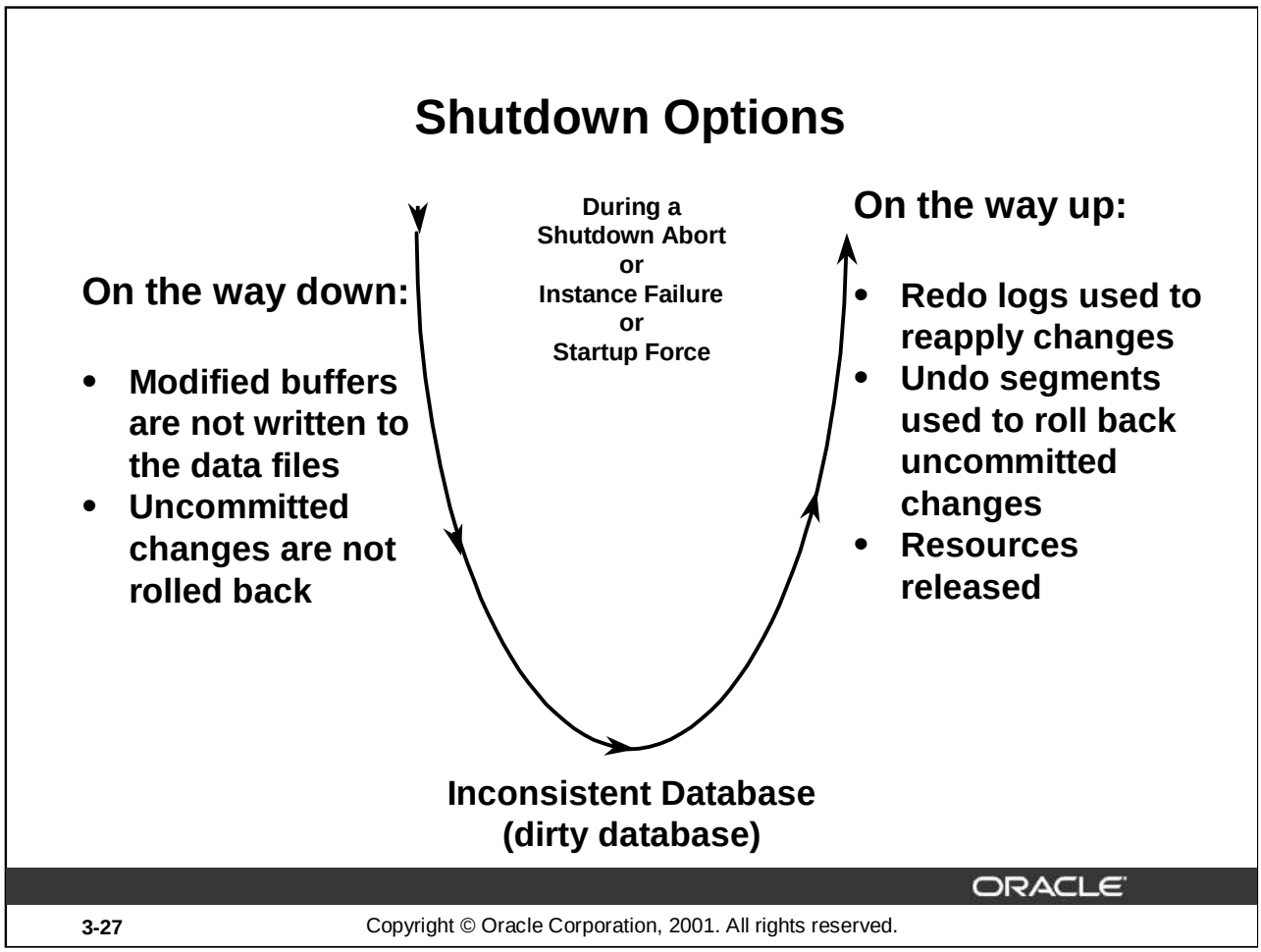
- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have finished, a shutdown immediately occurs.
- The next startup does not require an instance recovery.

Shutdown Options (continued)

Shutdown Immediate

Immediate database shutdown proceeds with the following conditions:

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



Shutdown Options

Shutdown Abort

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

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

Shutdown Options (continued)

How to Use the Console to Start Up or Shut Down

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Instance folder and click Configuration
5. From the General tab, select an Instance State, and click Apply.
6. **Note:**
   - You should be connected to the database with SYSDBA privileges to perform startup or shutdown.
   - You can also launch the Console from Windows NT Start menu

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**

ORACLE

Diagnostic Files

Diagnostic files are a means to capture information about the database's activities. They are also useful tools for you when you are managing an instance. Several types exist, and depending on the problem that occurred or the information that needs to be disseminated, they depend on the type of diagnostic file that is created.

- `alertSID.log` file: Information for day-to-day operation of the database
- Background trace files: Vital information when background processes, such as SMON, PMON, DBWn, and others fail.
- User trace files:Vital information for fatal user errors or user forced traced 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`.**

ORACLE

Alert Log File

Each Oracle instance has an alert log file. If not already created, Oracle creates one during instance startup. The alert log file is managed by you, as it continues to grow while the database continues to work. The alert log file should be the first place you look when diagnosing day-to-day operations or errors. The alert log file also contains pointers to trace files for more detailed information.

The alert log file keeps a record of the following information:

- When the database was started or shut down.
- A list of all non-default initialization parameters
- The startup of background processes
- The thread being used by the instance
- The log sequence number LGWR is writing to
- Information regarding a log switch
- Creation of tablespaces and undo segments
- Alter statements that have been issued
- Information regarding error messages such as ORA-600 and extent errors.

Alert Log File (continued)

The `alert_SID.log` location is defined by the `BACKGROUND_DUMP_DEST` initialization parameter. The default location on Unix is `$ORACLE_HOME/rdbms/log`.

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`**

ORACLE

Background Trace Files

Background trace files are used to log errors that have been encountered by a background process, such are SMON, PMON, DBWn, and other background processes. These files exist only when an error requires writing to the trace files. You use these files to diagnose and troubleshoot problems. Initially when a background trace file is created it contains header information indicating the version number of the data server and the operating system.

Naming convention for background trace files: *sid*_processname_PID.trc (db01_lgwr_23845.trc) on Unix. Its location is defined by the BACKGROUND_DUMP_DEST initialization parameter. The default location on Unix is $ORACLE_HOME/rdbms/log.

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.**

ORACLE

User Trace Files

User trace files contain statistics for traced SQL statements, which are useful for SQL tuning. In addition, user trace files contain user error messages.

Naming convention for user trace file: *sid*_ora_PID.trc (db01_ora_23845.trc) on Unix. Its location is defined by the USER_DUMP_DEST initialization parameter. The default location on Unix is $ORACLE_HOME/rdbms/log.

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`

ORACLE

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**

ORACLE

Quick Reference

Context	Reference
Initialization parameters	DB_NAME CONTROL_FILES SHARED_POOL_SIZE BACKGROUND_DUMP_DEST DB_BLOCK_BUFFERS COMPATIBLE IFILE LOG_BUFFER PROCESSES SQL_TRACE
Dynamic initialization parameters	USER_DUMP_DEST MAX_DUMP_FILE_SIZE TIMED_STATISTICS
Dynamic initialization parameters (deferred)	SORT_AREA_SIZE
Dynamic performance views	V$INSTANCE V$SESSION V$SYSTEM_PARAMETER

Quick Reference (continued)

Context	Reference
Data Dictionary Views	None
Commands	CONNECT / AS SYSDBA CONNECT / AS SYSOPER STARTUP SHUTDOWN SHOW PARAMETER ALTER SYSTEM KILL SESSION ALTER SYSTEM DISCONNECT SESSION…POST_TRANSACTION ALTER SYSTEM DISABLE RESTRICTED SESSION ALTER SESSION SET ALTER SYSTEM SET…DEFERRED ALTER SYSTEM SET…SCOPE ALTER DATABASE MOUNT ALTER DATABASE OPEN ALTER DATABASE OPEN READ ONLY ALTER DATABASE OPEN READ WRITE
Packaged procedure and functions	None

Practice 3 Overview

This practice covers the following topics:

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

ORACLE

Practice 3: Managing an Oracle Instance

1 Connect to the database as user SYS AS SYSDBA and shutdown the database.

2 With the database shutdown, create an SPFILE from the PFILE.
Place the SPFILE in directory $HOME/ADMIN/PFILE with the file name format spfileSID.ora, use your instance name in place of SID. Create the SPFILE from the PFILE located in $HOME/ADMIN/PFILE.

3 From the operating system, view the SPFILE.

4 Connect as user SYS AS SYSDBA, and start the database using the SPFILE.

5 Shutdown the database and open it in read-only mode. Connect as user HR password HR and and insert the following into the REGIONS table.

```
INSERT INTO regions VALUES ( 5, 'Mars' );
```

What happens? Put the database back in read-write mode.

6 Connect as user HR password HR and insert the following row into the table REGIONS; do not commit or exit.

```
INSERT INTO regions VALUES ( 5, 'Mars' );
```

In a new telnet session start SQL*Plus. Connect as SYS AS SYSDBA and perform a shut down transactional.

What happens? Rollback the insert in the HR session and exit, what happens?

Note: Parameter UNDO_RETENTION, which will be discussed in *Managing Undo Data*, is set to 15 minutes by default. Because of this parameter it could be 15 minutes before the database shuts down after the ROLLBACK command is executed. Connect to another session as SYS AS SYSDBA and execute the following command to speed up the process.

```
ALTER SYSTEM SET undo_retention=0 SCOPE=MEMORY;
```

7 Make sure the database is started. Keep the two SQL*Plus sessions open, one session as user SYS AS SYSDBA and one as user HR. As user SYS AS SYSDBA enable restricted session. As user HR select from the regions table. Is the select successful? In the HR session exit SQL*Plus then reconnect as HR. What happens? Disable restricted session.

4 Creating a Database

ORACLE

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**

ORACLE

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 Oracle9*i* 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**

ORACLE

Managing and Organizing a Database

Creating the database is the first step in managing and organizing a database system.

Database creation is a task that prepares several operating system files and is needed only once no matter how many data files the database has. This is a very important task, because you must decide on database settings.

Depending on the operating system, a database may have been created automatically as part of the installation.

You can use this initial database, or you can create a new one.

During migration from an older version of Oracle, database creation is necessary only if an entirely new database is needed. Otherwise you can use a migration utility—for example, Oracle Data Migration Assistant—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**

ORACLE

Creation Prerequisites

SYSDBA privileges are required to create a database. These are granted either using operating system authentication or password file authentication.

Note: Refer to the *Getting Started with the Oracle Server* lesson for more information regarding authentication.

Before you create the database, make sure that the memory for the SGA, the Oracle executable, and the processes is sufficient. Refer to your operating system installation and administration guides.

Calculate the necessary disk space for the database, including online redo log files, control files, and data files.

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**

ORACLE

Planning Database File Locations

Plan how to protect the database, including the online redo log files, control files, data files, archived redo log files, and provide a backup strategy.

Control Files

For the sake of safety, you should create at least two control files on two different disks.

Note: Refer to the *Maintaining the Control File* lesson for information regarding control files.

Online Redo Log Files

The online redo log files of a database should consist of multiplexed groups of online redo log files. A group of log files consists of identical copies, which should be located on different disks.

To distinguish between groups and their members, use a name such as `log0101.log` or `log01a.log`.

Note: Refer to the *Maintaining the Redo Log Files* lesson for information regarding redo log files.

Planning Database File Locations (continued)

Data Files

Name data files by relating to the contents as the root of the name—for example, data files such as `system01.dbf`, `temp01.dbf`, and `users01.dbf` on Unix.

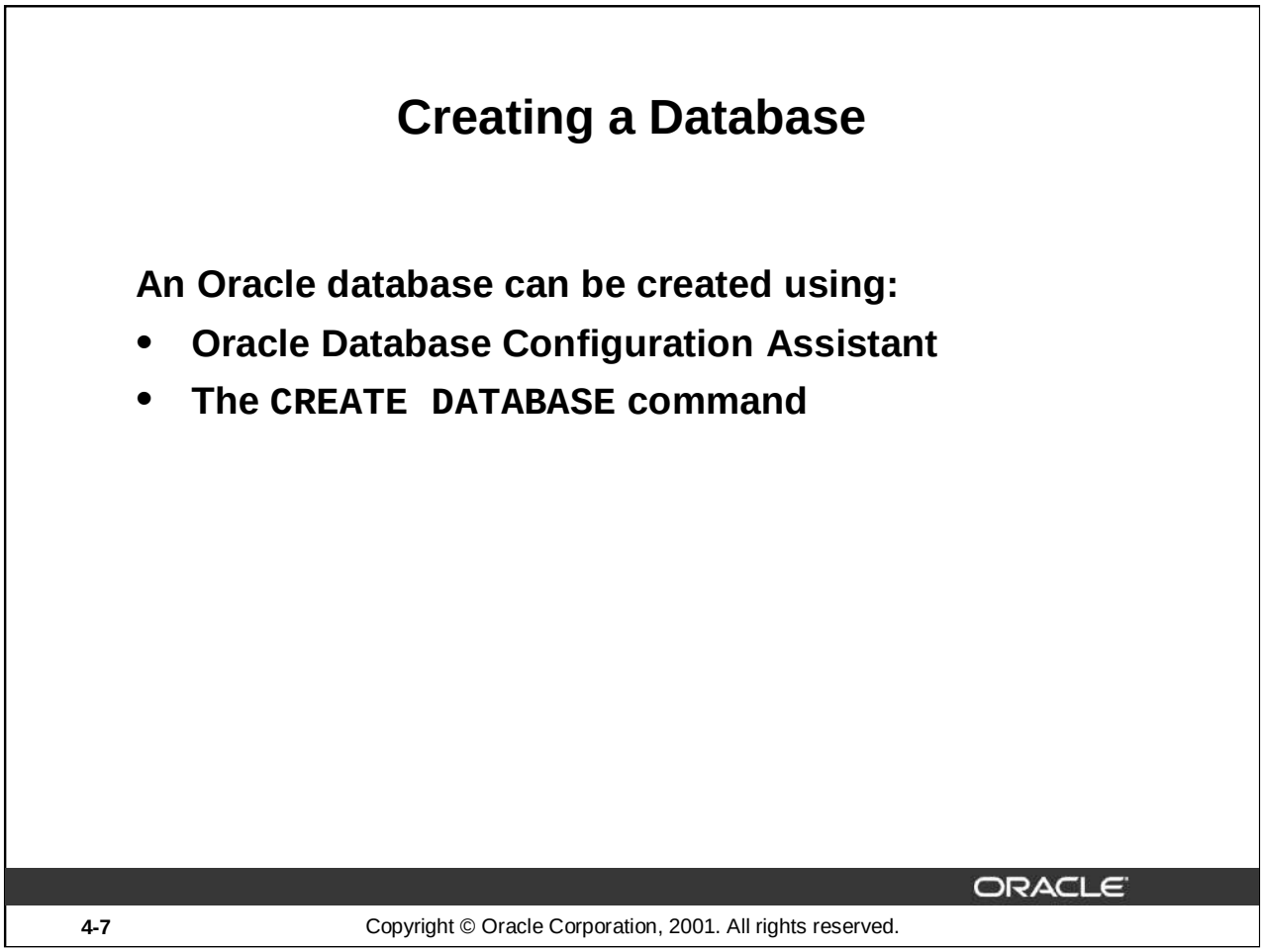
Consider the characteristics of the data to be stored before determining the structure appropriate for your database, in order to:

- Minimize fragmentation
- Minimize disk contention
- Separate objects

To minimize fragmentation of the database, you should separate database objects with different life spans, such as application data and temporary data, into different tablespaces.

To ensure well-balanced I/O loads, you should separate objects with competing I/O requirements, such as tables and indexes, into different tablespaces.

Note: Refer to the M*aintaining Tablespaces* and *Storage Structure and Relationships* lessons for information regarding these topics.



Creating a Database

Creating a database can be done either using the Oracle Database Configuration Assistant (DBCA) or by creating a SQL script using the CREATE DATABASE command.

The DBCA is a graphical user interface that interacts with the Oracle Universal Installer, or can be used stand-alone, to simplify the creation of a database. The DBCA is Java-based and can be launched from any platform with a Java engine.

During the installation of the Oracle Server, DBCA is launched by the Oracle Universal Installer and can automatically create a starter database for you. You have the option of using DBCA or not, and you also have the option to create a starter database. You also have the option to launch DBCA later as a standalone application to create a database.

You can also migrate or upgrade an existing database if you are using a previous version of Oracle.

Operating System Environment

On Unix, set the following environment variables:

- **ORACLE_BASE**
- **ORACLE_HOME**
- **ORACLE_SID**
- **ORA_NLS33**
- **PATH**
- **LD_LIBRARY_PATH**

ORACLE

Setting the Operating System Environment

Before creating a database manually or with the Database Configuration Assistant the operating system environment must be properly configured.

ORACLE_BASE: Specifies the directory at the top of the Oracle software

Example: /u01/app/oracle

ORACLE_HOME: Specifies the directory where the Oracle software is installed. The OFA-recommended value is $ORACLE_BASE/product/release.

Example: /u01/app/oracle/product/9i

ORACLE_SID: Specifies the instance name and must be unique for Oracle instances running on the same machine

ORA_NLS33: Required when creating a database with a character set other than US7ASCII

Example: $ORACLE_HOME/ocommon/nls/admin/data

PATH: Specifies the directories the operating system searches to find executables, such as SQL*Plus. The Oracle9*i* executables are located in $ORACLE_HOME/bin and needs to be added to the PATH variable.

LD_LIBRARY_PATH: Specifies the directories for the operating system and Oracle library files. Example: $ORACLE_HOME/lib

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**

ORACLE

Using the Oracle Database Configuration Assistant

Launching the Oracle Database Configuration Assistant

Unix:

- Enter `dbca` from the command prompt

Options available in Oracle Database Configuration Assistant

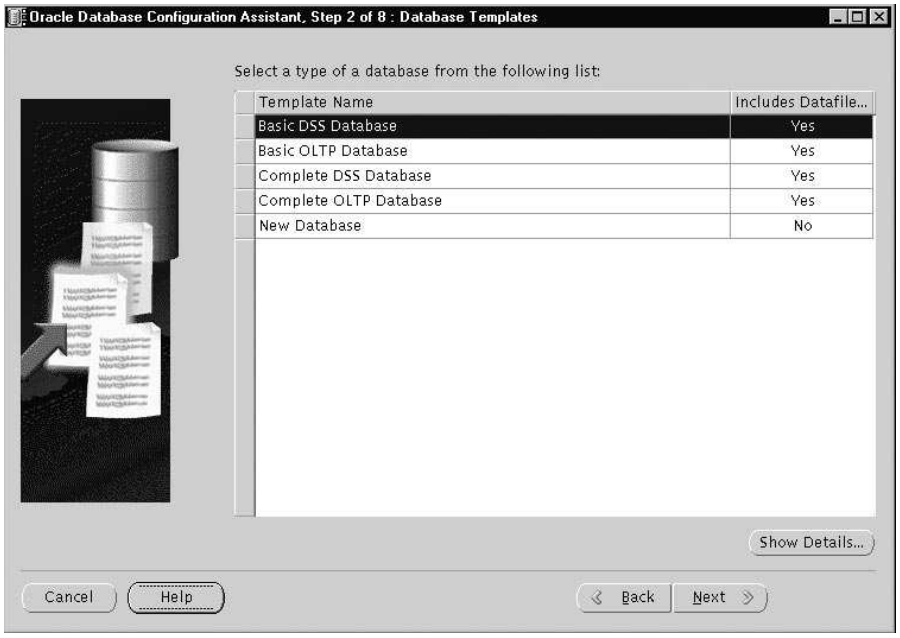
The Oracle Database Configuration Assistant allows you to do the following:

- **Create a database:** This option allows you to create a new database or template. The template allows you to choose, either, with datafiles or without datafiles.
- **Configure database options:** This option allows you to add database options that have not been previously configured, such as the following (if available): JVM, interMedia, Visual Information Retrieval, Spatial, Oracle OLAP Services, Advanced Replication and SQL*Plus help

Note: This option is not available with Oracle9*i* Real Application Clusters.

- **Delete a database:** This option allows you to delete a database.
- **Manage templates:** This option allows you to create a template through one of the following methods:
  - From pre-defined template settings: Create a new template from pre-defined template settings. You can add or change any template settings such as parameters, storage or use custom scripts
  - From an existing database (structure only): This template contains structural information similar to the existing database. This includes database options, tablespaces, datafiles and initialization parameters specified in the source database. User defined schemas and their data will not be part of the created template.
  - From an existing database (structure as well as data): This template will contain both the structure as well as the data of an existing database. Databases created using such a template will be identical to the source database. User defined schemas and their data will be part of the template. This template does not allow you to add or remove datafiles, tablespaces, or rollback segments.

Create a Database



ORACLE

Create a Database

Select a type of a database from the list of pre-defined templates. The templates can be either without datafiles or with datafiles.

Without datafiles

- Contains only the structure of the database
- Can specify and change all database parameters.

With datafiles

- Contains both the structure and the physical datafiles of the database
- All logfiles and control files are automatically created for the database and you can add/remove control files, log groups, change the destination and name of datafiles.
- You cannot add or remove datafiles, tablespaces, or rollback segments.
- Initialization parameters cannot be changed.

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**

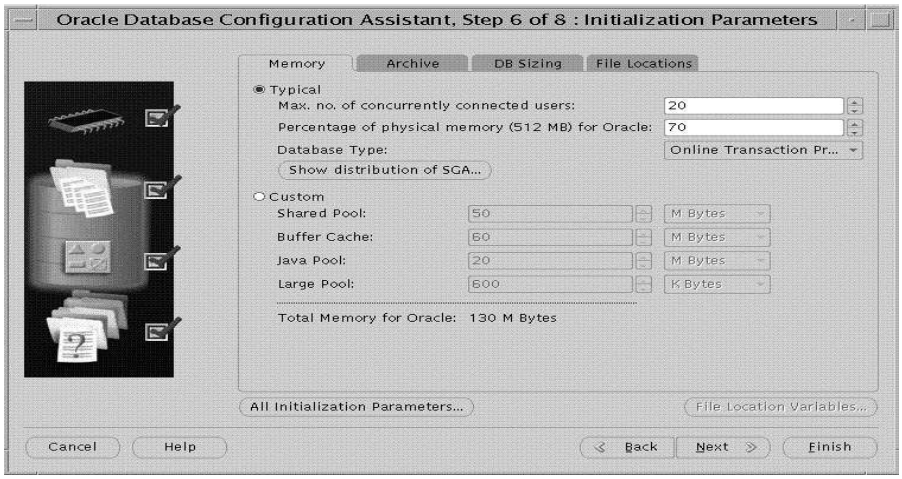
ORACLE

Database Information

- Specify global database name and `SID`
- The Example Schemas contain scripts for the following types of tables:
  - Human Resources
  - Order Entry
  - Product Media
  - Sales History
  - Shipping

Typical or Custom Install

Choose between typical or custom install



ORACLE

Create a Database

Choosing a Typical or Custom Database

Typical creates a database with minimal user input. With typical option, you can specify one of the following environments to operate the database: Online Transaction Processing (OLTP), Multipurpose, and Data Warehousing.

Online Transaction Processing (OLTP)

Databases in OLTP environments must process thousands of transactions from many concurrent users each day. These transactions consist of reading, writing, and deleting data in the database. Therefore, database performance is defined in terms of throughput and availability of data.

Data Warehousing

Databases in Data Warehousing environments must process a variety of queries (typically read-only), ranging from a simple fetch of a few records to numerous complex queries. Therefore, database performance is defined in terms of response time.

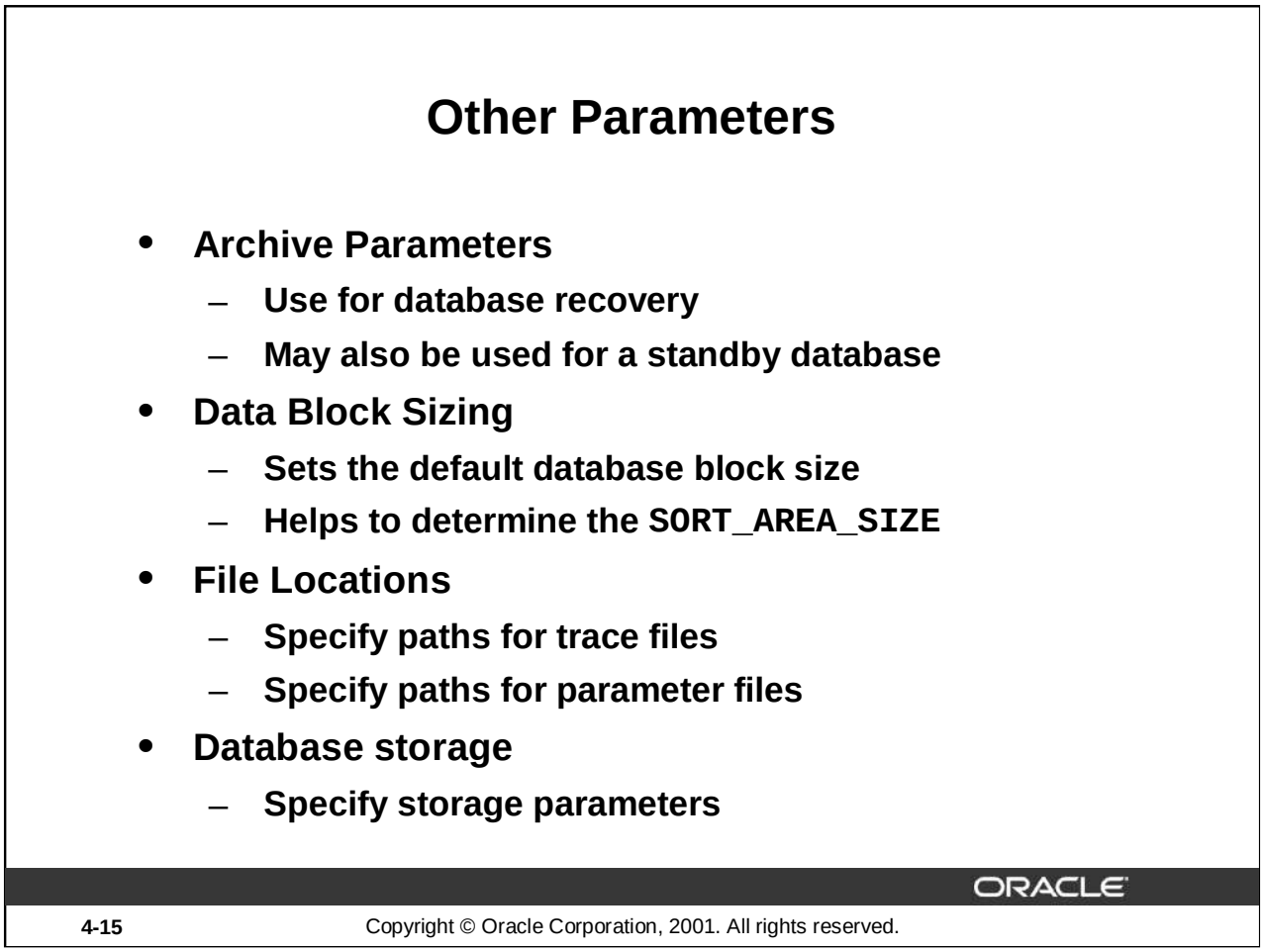
Mixed

Mixed databases support both OLTP and Data Warehousing environments. This is the default database that will be installed.

Create a Database (continued)

Custom allows you customize the creation of your database. This option is only for database administrators experienced with advanced database creation procedures, such as customizing:

- Data, control and redo settings, tablespace sizes, extent sizes, database memory parameters
- Archiving formats and destinations, trace file destinations, and character set values



Other Parameters

Archive Parameters

This option places the database in ARCHIVELOG mode and enables redo log files to be archived before being reused.

DB Sizing

This helps to define the block size and sort area size for the database. Data Block size of a database can be specified only at the time of database creation. SORT_AREA_SIZE is the maximum amount of memory used for sort operations.

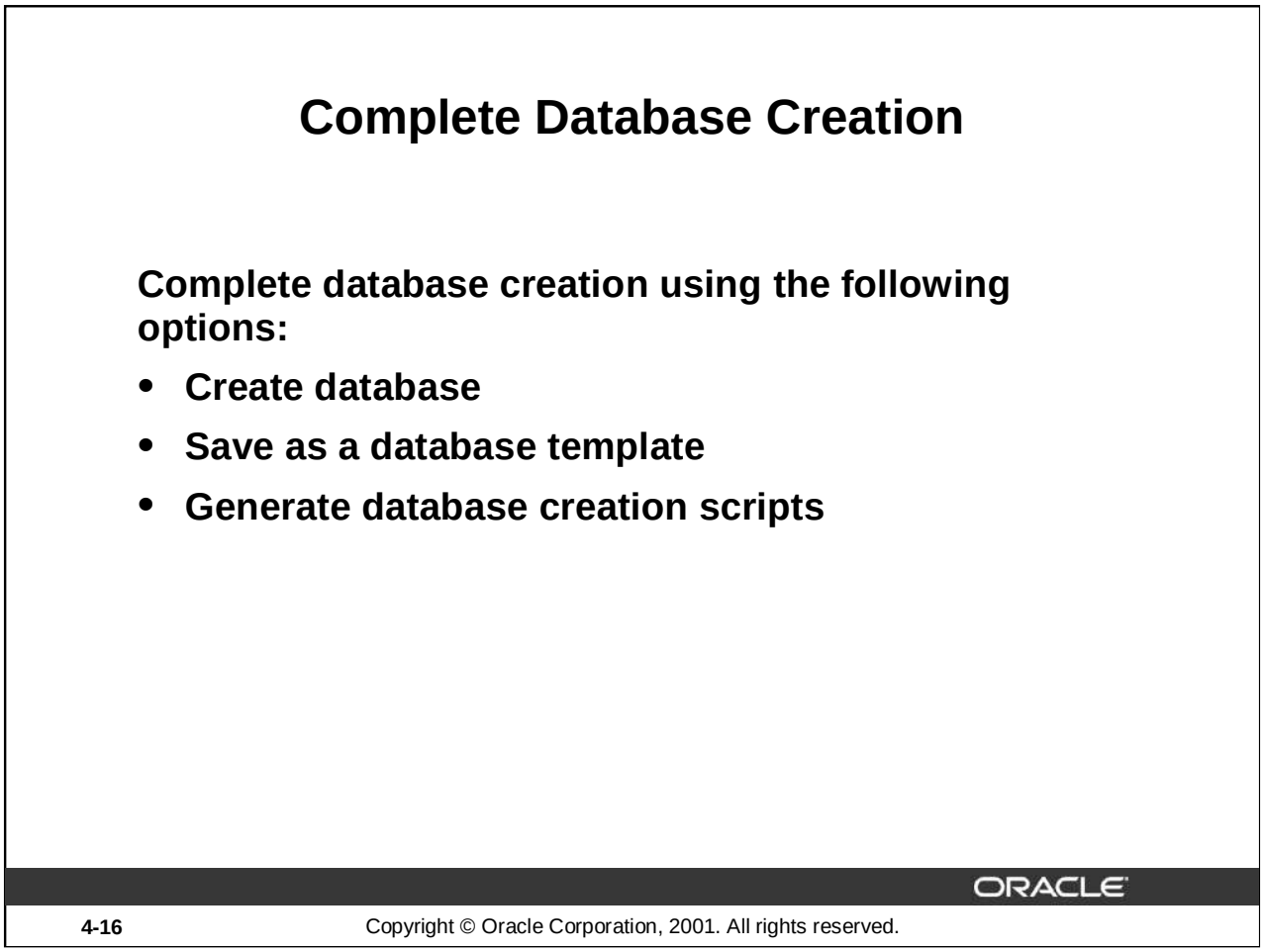
File Locations

- Specify locations for trace files
- Specify path for the initialization parameter file

Database Storage

Helps to specify storage parameters for database creation. You can view and change the following:

- Controlfiles
- Tablespaces, Datafiles
- Undo Segments, Redo Log Groups



Complete Database Creation

The following options are available to complete database creation:

- Create database – this option creates the database immediately
- Save as a database template – this option saves the database creation parameters as a template. This template will then be added to the list of available templates
- Generate database creation scripts – this option enables you to save the database creation parameters as a script file, for later use.

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

ORACLE

Creating a Database Manually

The last step, generating data dictionary, is discussed in the lesson *Data Dictionary Contents and Usage*.

Note: See your operating system specific Oracle documentation for information about creating databases on your platform.

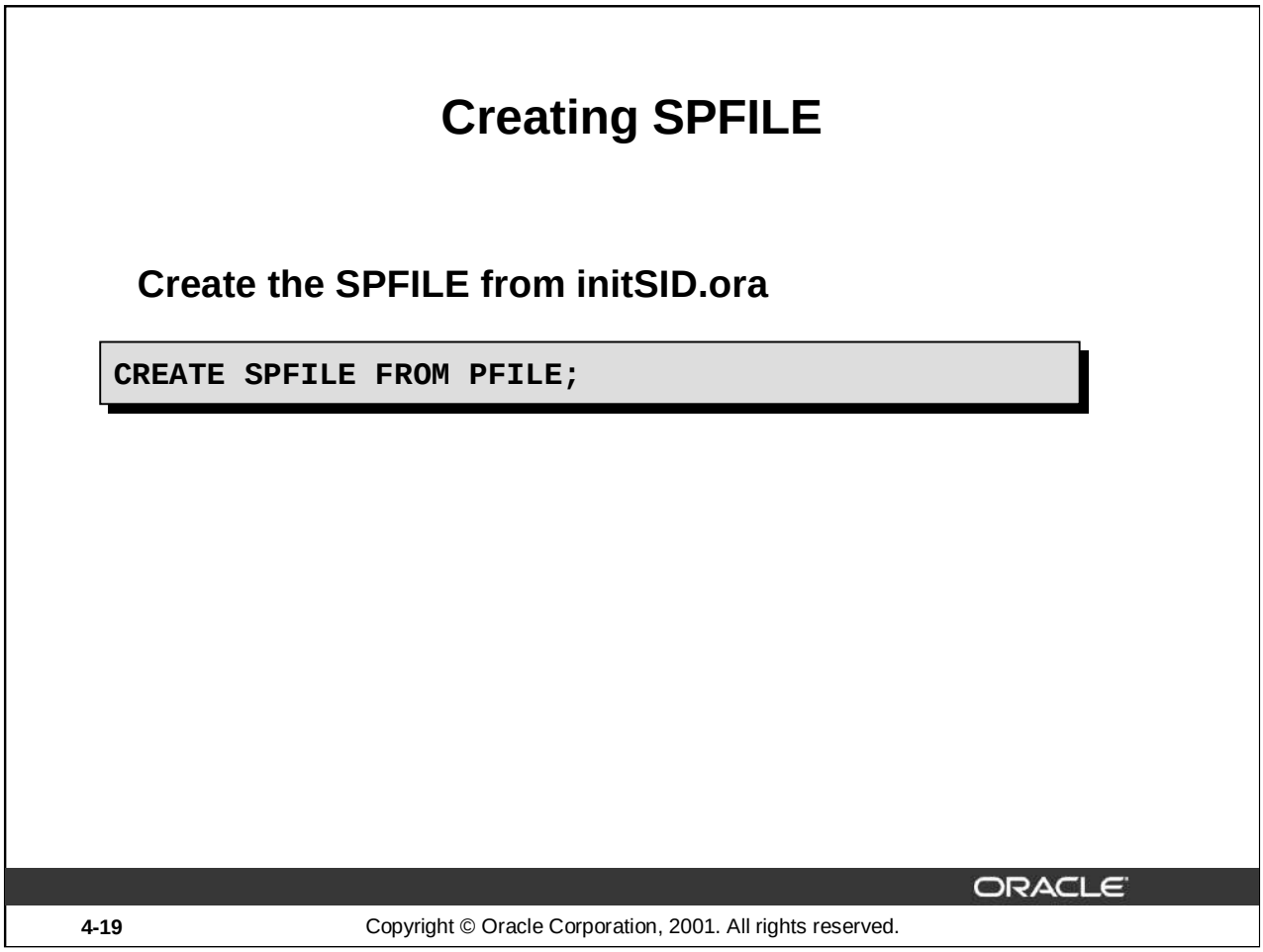
Preparing the Parameter File

- Create the new `initSID.ora`

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

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

ORACLE



Creating SPFILE

The SPFILE must initially be created from an initialization parameter file. You must have the SYSDBA or SYSOPER role to execute the CREATE SPFILE statement.

Note: Refer *Managing an Oracle Instance* for SPFILE.

Starting the Instance

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

```
STARTUP NOMOUNT
```

ORACLE

Starting the Instance

Connect as SYSDBA using operating system authentication or the password file method and start the instance using the STARTUP command.

If the parameter file is not in the default location, you may need to specify the PFILE clause in the STARTUP command.

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  ;
```

ORACLE

Creating the Database

To create a database, use the following SQL command:

```
CREATE DATABASE [database]
  [CONTROLFILE REUSE]
  [LOGFILE [GROUP integer] filespec
  [MAXLOGFILES integer]
  [MAXLOGMEMBERS integer]
  [MAXLOGHISTORY integer]
  [MAXDATAFILES integer]
  [MAXINSTANCES integer]
  [ARCHIVELOG|NOARCHIVELOG]
  [CHARACTER SET charset]
  [NATIONAL CHARACTER SET charset]
[DATAFILE filespec [autoextend_clause]
```

Creating the Database (continued)

```
   filespec :== 'filename' [SIZE integer][K|M] [REUSE]

      autoextend_clause :==

      [AUTOEXTEND {OFF|ON [NEXT integer[K|M]]

                                 [MAXSIZE {UNLIMITED|integer[K|M]} }]

   [DEFAULT TEMPORARY TABLESPACE tablespace filespec
     [temp_tablespace_extent_clause]

   temp_tablespace_extent_clause:==

   EXTENT MANAGEMENT LOCAL UNIFORM [SIZE integer][K|M] ]

   [UNDO TABLESPACE tablespace DATAFILE filespec
     [autoextend_clause] ]

   [SET TIME_ZONE [time_zone_region]]

   }

      ]
```

where:

database	is the name of the database to be created (If the name of the database is omitted, the initialization parameter DB_NAME is used.)
CONTROLFILE REUSE	specifies that an existing control file identified in the parameter file should be reused
LOGFILE GROUP	specifies the names of the log files to be used and the group to which they belong
MAXLOGFILES	specifies the maximum number of redo log file groups that can ever be created for the database
MAXLOGMEMBERS	specifies the maximum number of log file members for a log file group
MAXLOGHISTORY	specifies the maximum number of archived redo logs for automatic media recovery of the Oracle Parallel Server

Creating the Database (continued)

DATAFILE	filespec specifies the data files to be used
AUTOEXTEND	enables or disables the automatic extension of a data file
MAXDATAFILES	specifies the initial sizing of the data file section of the control file at CREATE DATABASE or CREATE CONTROLFILE time. An attempt to add a new file whose number is greater than MAXDATAFILES, but less than or equal to DB_FILES,causes the control file to expand automatically so that the data files section can accommodate more files
MAXINSTANCES	is the maximum number of instances that can simultaneously mount and open the database
ARCHIVELOG	establishes that redo logs must be archived before they can be reused
NOARCHIVELOG	establishes that redo logs can be reused without archiving their contents
CHARACTER SET	is the character set the database uses to store data
NATIONAL CHARACTER SET	specifies the national character set used to store data in columns defined as NCHAR, NCLOB, or NVARCHAR2.If not specified, the national character set is the same as the database character set
DEFAULT TEMPORARY TABLESPACE	creates a default temporary tablespace for the database. Oracle will assign to this temporary tablespace any users for whom you do not specify a different temporary tablespace
UNDO TABLESPACE	creates an undo tablespace and creates the specified datafiles as part of the undo tablespace.
SET TIME_ZONE	sets the time zone for the database.

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;
```

ORACLE

Creating a Database Using OMF

Once the OMF parameters are defined in the initialization file the syntax for creating the database becomes simple.

Troubeshooting

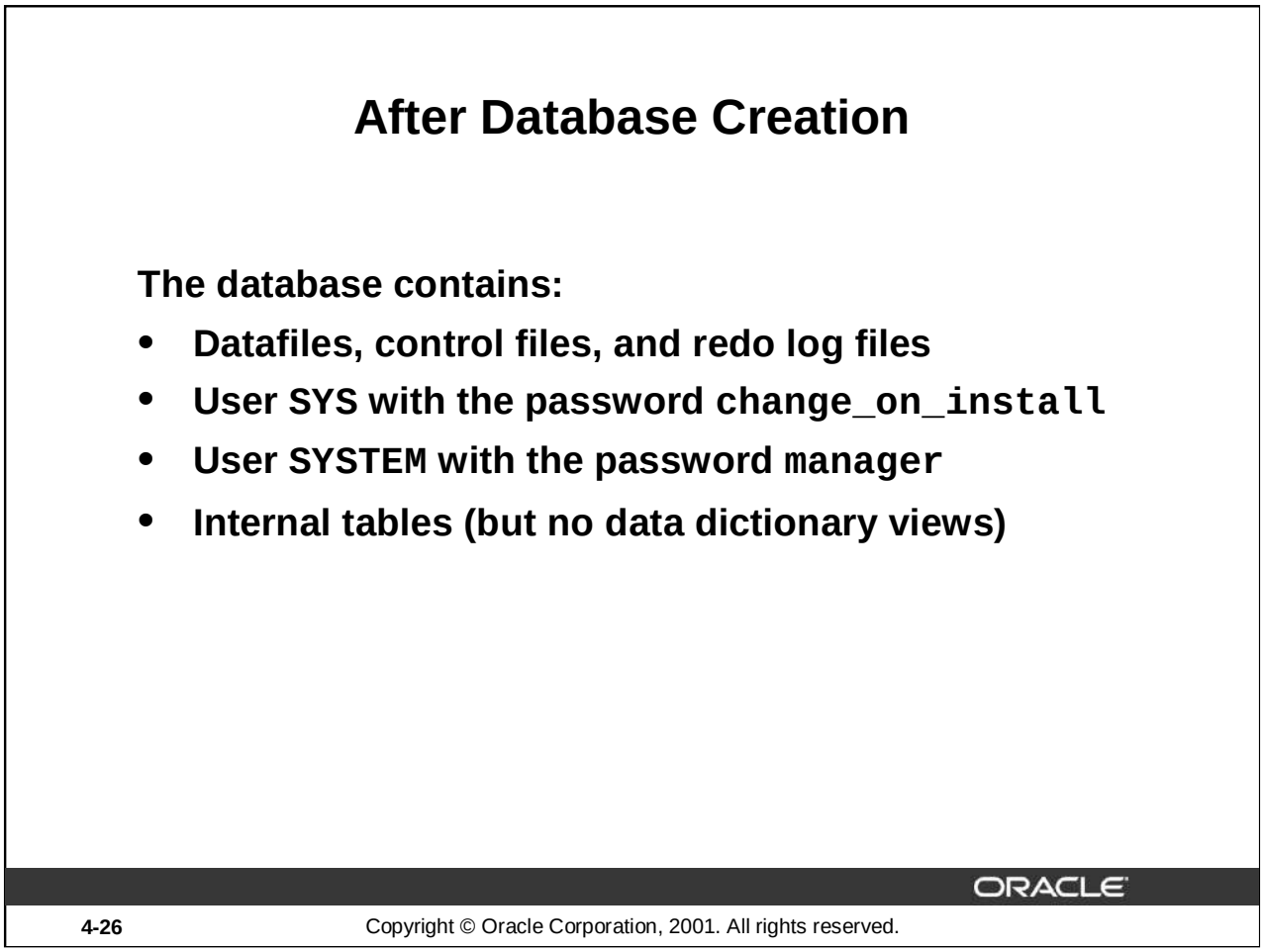
Creation of the database fails if:

- **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**

ORACLE

Troubleshooting

If one of the three problems shown on the slide occurs, the CREATE DATABASE statement fails. You should delete any files created by the CREATE DATABASE statement, correct the errors, and attempt to create again.



After Database Creation

After the database is created, the instance is left running and the database is open and available for normal use. The database contains users SYS and SYSTEM. Depending on the method of database creation, either using DBCA or manually, other users might get created. Change the passwords for SYS and SYSTEM as soon as the database is created.

You can view the dynamic performance views such as V$LOGFILE, V$CONTROLFILE, and V$DATAFILE, but no data dictionary views are created.

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**

ORACLE

5

Data Dictionary Contents and Usage

ORACLE

Objectives

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

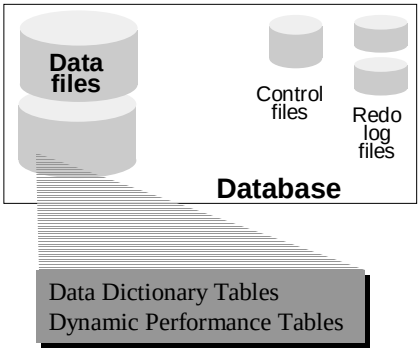
- **Identify key data dictionary components**
- **Identify the contents and uses of the data dictionary**
- **Query the data dictionary**

ORACLE

Data Dictionary

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

- **Data dictionary tables**
- **Dynamic performance tables**



ORACLE

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**

ORACLE

Data Dictionary

Base Tables: Underlying tables, which store information about the database. The base tables are the first objects created in any Oracle database. They are automatically created when the Oracle server runs the `sql.bsq` script at the time the database is created. Only the Oracle server should write to these tables. Users rarely access them directly, because most of the data is stored in a cryptic format. Never use DML commands to update the base tables directly, with the exception of the AUD$ table.

Note: Refer to the *Managing Privileges* lesson for more information on the AUD$ table.

An example of a base table is the IND$ table, which contains information about the indexes in the database.

Data Dictionary Views: Base table summaries, which provide for a more useful display of base table information. For example, in the data dictionary views, object names are used instead of only object numbers. The data dictionary views are created using the `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**

ORACLE

Data Dictionary Contents

A data dictionary contains:

- The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)
- How much space has been allocated for, and is currently used by, the schema objects
- Default values for columns
- Integrity constraint information
- The names of Oracle users
- Privileges and roles each user has been granted
- Auditing information, such as who has accessed or updated various schema objects

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.**

ORACLE

How the Data Dictionary Is Used

How the Oracle Server Uses the Data Dictionary

Data in the base tables of the data dictionary are necessary for the Oracle server to function. Therefore, only the Oracle server should write or change data dictionary information. During database operation, the Oracle server reads the data dictionary to ascertain that schema objects exist and that users have proper access to them. The Oracle server also updates the data dictionary continuously to reflect changes in database structures.

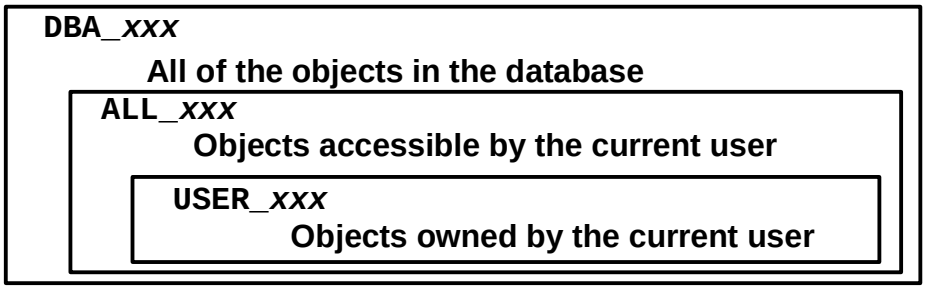
How Users and Database Administrators Can Use the Data Dictionary

The views of the data dictionary serve as a reference for all database users. Some views are accessible to all Oracle users. Others are intended for database administrators only.

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**



ORACLE

Data Dictionary View Categories

Views with the DBA Prefix

Views with the DBA prefix show a global view of the entire database. They are meant to be queried only by database administrators. Any user granted the system privilege SELECT ANY TABLE can query the DBA-prefixed views of the data dictionary.

To query on all objects in the database, the DBA could issue the following statement:

```
SELECT owner, object_name, object_type
FROM dba_objects;
```

Views with the ALL Prefix

Views with the ALL prefix refer to the user's overall perspective of the database. These views return information about schema objects to which the user has access by way of public or explicit grants of privileges and role, in addition to schema objects that the user owns.

For example, the following query returns information about all the objects to which a user has access:

```
SELECT owner, object_name, object_type
FROM all_objects;
```

Data Dictionary View Categories (continued)

The views most likely to be of interest to typical database users are those with the USER prefix.

These views:

- Refer to the user's own private environment in the database
- Generally refer to objects owned by the current user
- Have columns identical to the other views, except that the column `OWNER` is implied to be the current user
- Return a subset of the information in the ALL views
- Can have abbreviated pubic synonyms for convenience

For example, the following query returns all the objects contained in the users schema:

```
SELECT owner, object_name, object_type
FROM users_objects;
```

Data Dictionary Views

Data dictionary views are static views that answer questions such as:

- Was the object ever created?
- What is the object a part of?
- Who owns the object?
- What privileges do users have?
- What restrictions are on the object?

Note: Refer to the *Oracle9i Database Reference* document for a listing of all data dictionary views.

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**

ORACLE

Dynamic Performance Views

Throughout its operation, the Oracle server records current database activity in a set of virtual tables called dynamic performance views. These virtual tables exist in memory only when the database is running, to reflect real-time conditions of the database operation. They point to actual sources of information in memory and the control file.

The dynamic performance tables are owned by SYS.

Dynamic Performance Views

The dynamic tables answer questions such as:

- Is the object online and available?
- Is the object open?
- What locks are being held?
- Is the session active?

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.**

ORACLE

Querying the Data Dictionary

To get an overview of the data dictionary and dynamic performance views, the `DICTIONARY` view or its synonym `DICT` can be queried. For example:

```
SELECT * FROM dictionary;
```

Include the where clause to narrow your responses:

```
SELECT * FROM dictionary WHERE table_name like 'sort%'
```

The `V$FIXED_TABLE` view can also be queried to get a listing of the views:

```
SELECT * FROM V$FIXED_TABLE;
```

To get a list of columns within a view, use the `DESCRIBE` keyword:

```
DESCRIBE V$INSTANCE;
```

To get an overview of the columns in the data dictionary and dynamic performance views, the `DICT_COLUMNS` view can be queried.

To view the contents of the view use the `SELECT` command.

```
SELECT * from V$INSTANCE;
```

Note: Refer to the *Oracle9i Database Reference* document for a complete list of data dictionary views and their columns.

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

ORACLE

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`**

ORACLE

Practice 5 Overview

This practice covers the following topics:

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

ORACLE

Practice 5: Data Dictionary

1 Which of the following statements are true about the data dictionary?
- **a** The data dictionary describes the database and its objects.
- **b** The data dictionary includes two types of objects: base tables, data dictionary views.
- **c** The data dictionary is a set of read-only tables.
- **d** The data dictionary records and verifies information about its associated database.

2 Base tables are created using the catalog.sql script.
- **a** True
- **b** False

3 Which of the following statements are true about how the data dictionary is used?
- **a** The Oracle server modifies it when a DML statement is executed.
- **b** It is used to find information about users, schema objects, and storage structures.
- **c** Used by users and DBAs as a read-only reference.
- **d** The data dictionary is a necessary ingredient for the database to function.

4 Data dictionary views are static views.
- **a** True
- **b** False

5 The information for a Dynamic View is gathered from the control file and data files.
- **a** True
- **b** False

6 Which of the following questions might a Dynamic View answer?
- **a** Is the object online and available?
- **b** What locks are being held?
- **c** Who owns the object?
- **d** What privileges do users have?
- **e** Is the session active?

Practice 5: Data Dictionary

7 Find a list of the data dictionary views.

8 Identify the database name, instance name, and size of the database blocks.

Hint: Query the Dynamic Views V$DATABASE, V$THREAD, and V$PARAMETER.

9 List the name and size of the data files.

Hint: Query the view V$DATAFILE.

10 Identify the data file that makes up the SYSTEM tablespace.

Hint: Query the data dictionary view DBA_DATA_FILES to identify the data files that make up SYSTEM tablespace.

11 How much free space is available in the database and how much is already used?

Hints:

- Query the data dictionary view DBA_FREE_SPACE to show how much free space is available in the database.
- Query the data dictionary view DBA_SEGMENTS to display how much space is already used.

12 List the name and creation date of the database users.

Hint: Query the data dictionary view DBA_USERS to list the name and the creation of the database users.

6

Maintaining the Control File

ORACLE

Objectives

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

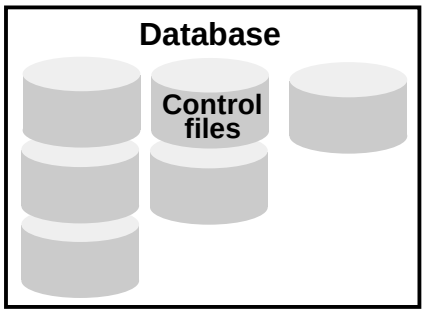
- **Explain the uses of the control file**
- **List the contents of the control file**
- **Multiplex and manage the control file**
- **Manage the control file with Oracle Managed Files**
- **Obtain control file information**

ORACLE

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`**



ORACLE

Control File

The control file is a small binary file necessary for the database to start and operate successfully. Each control file is associated with only one Oracle database. Before a database is opened, the control file is read to determine if the database is in a valid state to use. A control file is updated continuously by the Oracle server during database use, so it must be available for writing whenever the database is open. The information in the control file can be modified only by the Oracle server; no database administrator or end user can edit the control file. If for some reason the control file is not accessible, the database does not function properly. If all copies of a database's control files are lost, the database must be recovered before it can be opened. At least one control file is required, but control files can be multiplexed up to eight times.

Control File (continued)

Sizing the Control File

Keywords specified during the creation of the database affect the size of the control file. This is particularly significant when the parameters have large values. The size of the control file is influenced by the following keywords in the CREATE DATABASE or CREATE CONTROLFILE commands:

- MAXLOGFILES
- MAXLOGMEMBERS
- MAXLOGHISTORY
- MAXDATAFILES
- MAXINSTANCES

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**

ORACLE

Control File Contents

The information in the control file includes:

- Database name is taken from either the name specified by the initialization parameter `DB_NAME` or the name used in the `CREATE DATABASE` statement.
- Database identifier is recorded when the database is created.
- Time stamp of database creation is also recorded at database creation.
- Names and locations of associated data files and online redo log files are updated when a data file or redo log is added to, renamed in, or dropped from the database.
- Tablespace information is updated as tablespaces are added or dropped.
- Redo log history is recorded during log switches.
- Location and status of archived logs are recorded when archiving occurs.
- Location and status of backups are recorded by the Recovery Manager utility.
- Current log sequence number is recorded when log switches occur.
- Checkpoint information is recorded as checkpoints are made.

Contents of the Control File (continued)

The control file consists of two types of sections:

- Reusable
- Not reusable

Reusable sections store Recovery Manager information, such as backup data file names and backup redo log file names. They are used in a circular manner and can be reused only by Recovery Manager.

Note: Recovery Manager is covered in more detail in the course *Oracle9i DBA Fundamentals II*.

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> startup
```

- **Create additional control files**

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

- **Start the database:**

```
SQL> startup
```

ORACLE

Multiplexing the Control File

To safeguard against a single point of failure of the control file, it is strongly recommended that the control file be multiplexed, storing each copy on a different physical disk. If a control file is lost, a copy of the control file can be used to restart the instance without database recovery. Control files can be multiplexed up to eight times. The Oracle server creates and maintains all files listed in this parameter when the instance is started.

The database administrator can multiplex control files by:

- Creating multiple control files when the database is created by including the control file names in the CONTROL_FILES initialization parameter:

  ```
  CONTROL_FILES=$HOME/ORADATA/u01/ctrl01.ctl,
                $HOME/ORADATA/u02/ctrl02.ctl
  ```

- Adding a control file after the database is created. This steps differ depending on weather an INIT.ora file or SPFILE is being used.

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
```

ORACLE

Safeguarding Control Files

There is a `CREATE CONTROLFILE` command that allows creation of a control file from scratch. However, you must have complete knowledge of the database contents and structure to use it properly.

Backup After Database Structure Changes

Because the control file records the physical structure of the database, you should immediately make a backup of your control file after making changes to the physical structure of the database.

You can create a backup of a control file, but you cannot bring a control file back from a backup without its appropriate data files. The control file is a living file that corresponds to current database status.

```
ALTER DATABASE BACKUP CONTROLFILE TO 'FILENAME'
```

You can also backup your control file to a trace file. This will create a file with the SQL statements required to recreate your control file.

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE
```

Backup and recovery of the control file is covered in detail in the course *DBA Fundamentals II*.

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.**

ORACLE

Managing Control Files with OMF

Control files are created as OMF automatically during database creation time if the `CONTROL_FILES` parameter is not specified in the initialization parameter file. If using an `init.ora` file, the `CONTROL_FILES` parameter must be set to the OMF generated names, which can be found by selecting from `V$CONTROLFILE or from the alertSID.log`. If an `SPFILE` is used, the `CONTROL_FILES` parameter is automatically set and saved when the database is created. The control file names are uniquely generated `(ora_cmr7t30p.ctl)` and displayed in the `alertSID.log` when the files are created.

You can also create a new control file for the database using the `CREATE CONTROLFILE` command. The correct OMF names must be used in the `DATAFILE` and `LOGFILE` clauses. The `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command generates a script with the correct file names.

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.**

ORACLE

Obtaining Control File Information

To obtain the location and names of the control files, use the dynamic performance view V$CONTROLFILE.

```
SELECT name FROM V$CONTROLFILE;
NAME
------------------------------------
/u01/home/db03/ORADATA/u01/ctrl01.ctl
/u01/home/db03/ORADATA/u01/ctrl01.ctl
2 rows selected.
```

The V$PARAMETER view can also be used.

```
SELECT name, value from V$PARAMETER
WHERE name = 'control_files';
NAME            Value
-------------   -------------------------------------
control_files   /u01/home/db03/ORADATA/u01/ctrl01.ctl
```

Obtaining Control File Information (continued)

To obtain information about the different sections of the control files, query the V$CONTROLFILE_RECORD_SECTION dynamic performance view.

```
SELECT type, record_size, records_total, records_used
FROM v$controlfile_record_section
WHERE type='DATAFILE';
TYPE            RECORD_SIZ RECORDS_TO RECORDS_US
-------         ---------- -------- ------------------
DATAFILE        180        30         4
1 row selected.
```

The column RECORDS_TO specifies the number of records allocated for a special section. For example, you can view the maximum number of data files, in our example 30, which is determined by the MAXDATAFILES parameter in the CREATE DATABASE command.

The SHOW PARAMETERS command can also be used to find the location of the control files.

```
show parameters control_files;
NAME         TYPE    VALUE
------------ ------- ------------------------------
control_files string  /u01/home/db03/ORADATA/u01/ctrl01.ctl
```

Information in several of the other dynamic performance views is obtained from the control file:

- V$BACKUP
- V$DATAFILE
- V$TEMPFILE
- V$TABLESPACE
- V$ARCHIVE
- V$LOG
- V$LOGFILE
- V$LOGHIST
- V$ARCHIVED_LOG
- V$DATEBASE

Quick Reference

Context	Reference
Initialization parameters	CONTROL_FILES
Dynamic performance views	V$CONTROLFILE V$CONTROLFILE_RECORD_SECTION V$PARAMETER
Data dictionary views	None
Commands	None
Packaged procedures and functions	None

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**

ORACLE

Practice 6 Overview

This practice covers the following topics:

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

ORACLE

Practice 6: Maintaining the Control File

1. Where is the existing control file located and what is the name?

   Hint: Query the dynamic performance view `V$CONTROLFILE` or `V$PARAMETER`, or execute the `SHOW PARAMETER` command to display the name and the location of the control file.

2. Try to start the database without any control files. (Simulate this by changing the name of the control file in the parameter file or changing the control file name.) What happens?

3. Multiplex the existing control file, using the directory `u02`, and name the new control file `ctrl02.ctl`. Make sure that the Oracle Server is able to write to the new control file. For example, on Unix use the command `chmod 660`. Confirm that both control files are being used.

   Hints:

   - Before shutting down the database alter the `SPFILE` (`SCOPE=SPILE`) to add the new control file to the initialization file.
   - Shut down the database, and copy the existing control file to a new file with the name `ctrl02.ctl` in the directory `u02`. Use the command `chmod 660` on Unix. Normally the permissions on the file would not be changed, this is for the classroom environment.
   - Start up the database.
   - Query the Dynamic View `V$CONTROLFILE` or `V$PARAMETER`, or use the `SHOW PARAMETER` command to confirm that both control files are being used.

4. What is the initial sizing of the data file section in your control file?

   Hint: Query the Dynamic View `V$CONTROLFILE_RECORD_SECTION`.

7

Maintaining Redo Log Files

ORACLE

Objectives

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

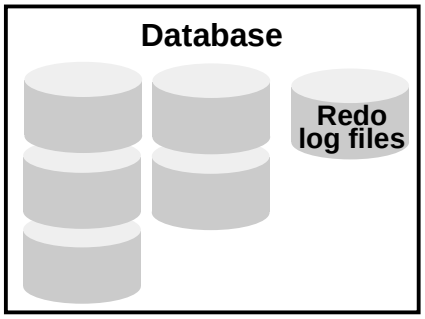
- **Explain the purpose of online redo log files**
- **Outline the structure of online redo log files**
- **Control log switches and checkpoints**
- **Multiplex and maintain online redo log files**
- **Manage online redo logs files with OMF**

ORACLE

Using Redo Log Files

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

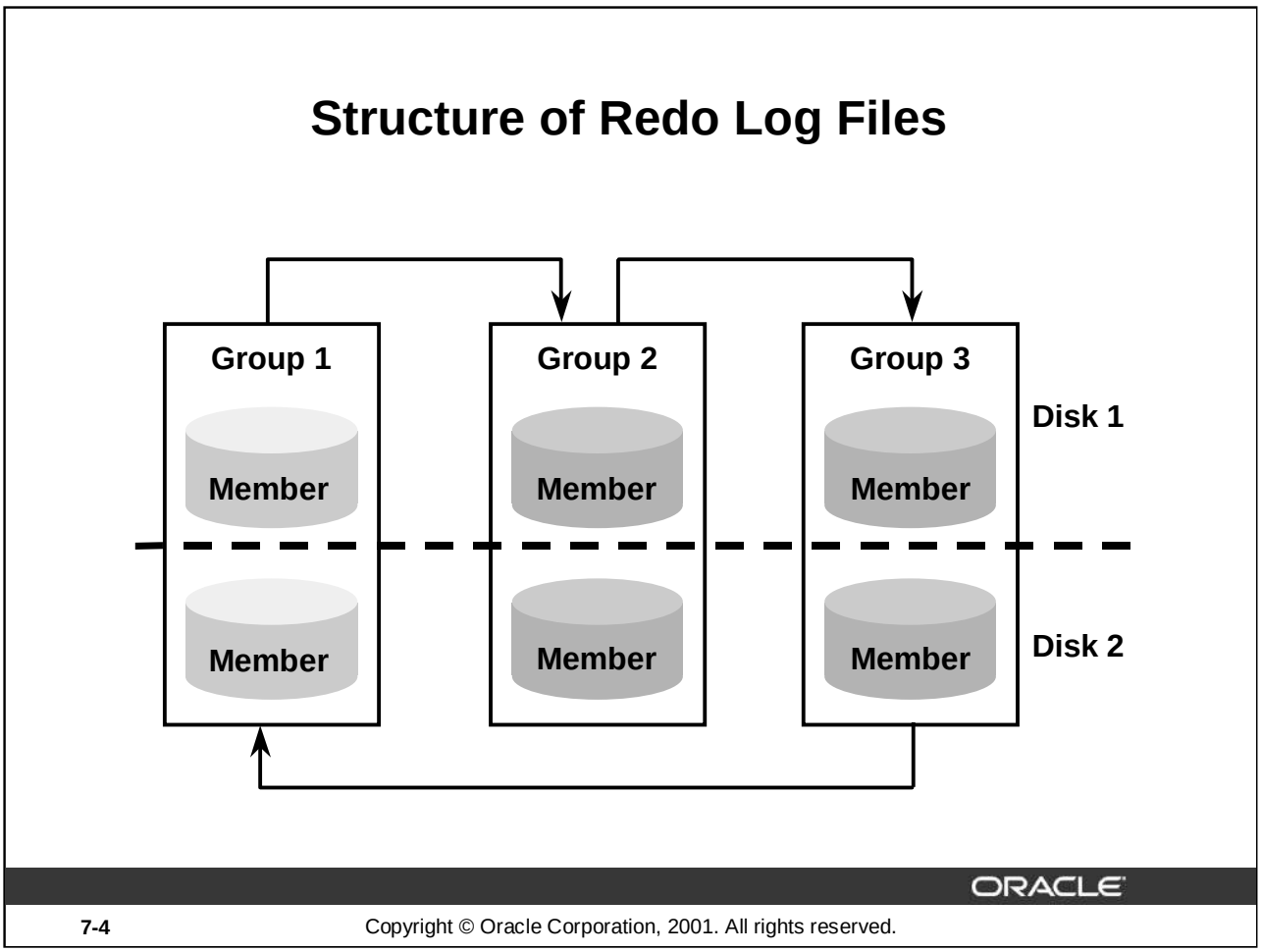
- **Redo log files are organized into groups.**
- **An Oracle database requires at least two groups.**
- **Each redo log within a group is called a member.**



ORACLE

Purposes of the Redo Log Files

Redo log files provide the means to redo transactions in the event of a database failure. Every transaction is written synchronously to the redo log files in order to provide a recovery mechanism in case of media failure. (With exceptions such as: direct loads and direct reads done with the `NOLOGGING` option.) This includes transactions that have not yet been committed, undo segment information, and schema and object management statements. Redo log files are used in a situation such as an instance failure to recover committed data that has not been written to the data files. The redo log files are used only for recovery.



Structure of the Redo Log Files

The database administrator can set up the Oracle database to maintain copies of online redo log files to avoid losing database information due to a single point of failure.

Online Redo Log Groups

- A set of identical copies of online redo log files is called an online redo log group.
- The LGWR background process concurrently writes the same information to all online redo log files in a group.
- The Oracle server needs a minimum of two online redo log file groups for the normal operation of a database.

Online Redo Log Members

- Each online redo log file in a group is called a member.
- Each member in a group has identical log sequence numbers and the same size. The log sequence number is assigned each time the Oracle server starts writing to a log group to identify each redo log file uniquely. The current log sequence number is stored in the control file and in the header of all data files.

Structure of the Redo Log Files (continued)

Creating Initial Redo Log Files

The initial set of online redo log groups and members are created during the database creation.

The following parameters limit the number of online redo log files:

- The `MAXLOGFILES` parameter in the `CREATE DATABASE` command specifies the absolute maximum of online redo log groups.
- The maximum and default value for `MAXLOGFILES` is dependent on your operating system.
- The `MAXLOGMEMBERS` parameter used in the `CREATE DATABASE` command determines the maximum number of members per group. The maximum and default value for `MAXLOGMEMBERS` is dependent on your operating system.

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**

ORACLE

How Redo Logs Work

The Oracle server sequentially records all changes made to the database in the redo log buffer. The redo entries are written from the redo log buffer to one of the online redo log groups called the current online redo log group by the LGWR process. LGWR writes under the following situations:

- When a transaction commits
- When the redo log buffer becomes one-third full
- When there is more than a megabyte of changed records in the redo log buffer
- Before the DBWn writes modified blocks in the database buffer cache to the data files

Redo logs are used in a cyclic fashion. Each redo log file group is identified by a log sequence number that is overwritten each time the log is reused.

Log Switches

LGWR writes to the online redo log files sequentially. When the current online redo log group is filled, LGWR begins writing to the next group. This is called a log switch.

When the last available online redo log file is filled, LGWR returns to the first online redo log group and starts writing again.

How Redo Logs Work (continued)

Checkpoints

During a checkpoint:

- A number of dirty database buffers covered by the log being checkpointed are written to the data files by DBWn. The number of buffers being written by DBWn is determined by the FAST_START_MTTR_TARGET parameter, if specified.

Note: The FAST_START_MTTR_TARGET parameter is covered in detail in the *Oracle9i DBA Fundamentals II* course.

- The checkpoint background process CKPT updates the headers of all data files and control files to reflect that it has completed successfully.

Checkpoints can occur for all data files in the database or for only specific data files.

A checkpoint occurs, for example, in the following situations:

- At every log switch
- When an instance has been shut down with the normal, transactional, or immediate option
- When forced by setting the initialization parameter FAST_START_MTTR_TARGET.
- When manually requested by the database administrator
- When the ALTER TABLESPACE [OFFLINE NORMAL|READ ONLY|BEGIN BACKUP] cause checkpointing on specific data files.

Information about each checkpoint is recorded in the alert_SID.log file if the LOG_CHECKPOINTS_TO_ALERT initialization parameter is set to TRUE. The default value of FALSE for this parameter does not log checkpoints.

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;
```

ORACLE

Forcing Log Switches and Checkpoints

Log switches and checkpoints are automatically done at certain points in the operation of the database as identified previously, but a DBA can force a log switch or a checkpoint to occur.

Forcing Log Switches

A log switch can be forced using the following SQL command:

```
ALTER SYSTEM SWITCH LOGFILE;
```

Forcing Checkpoints

A checkpoint can be forced using the following SQL command:

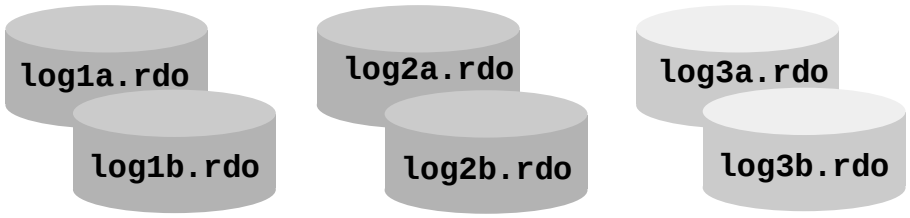
```
ALTER SYSTEM CHECKPOINT;
```

A checkpoint can be forced using the `FAST_START_MTTR_TARGET` parameter.

`FAST_START_MTTR_TARGET` is a shortcut to the deprecated parameters `FAST_START_IO_TARGET` and `LOG_CHECKPOINT_TIMEOUT.` `FAST_START_MTTR_TARGET = 600` indicates that instance recovery should not take more than 600 seconds, and the database will adjust the other parameters to this goal. `FAST_START_IO_TARGET` and `LOG_CHECKPOINT_TIMEOUT` must not be used if `FAST_START_MTTR_TARGET` is used.

Adding Online Redo Log Groups

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



ORACLE

Adding Redo Log Groups

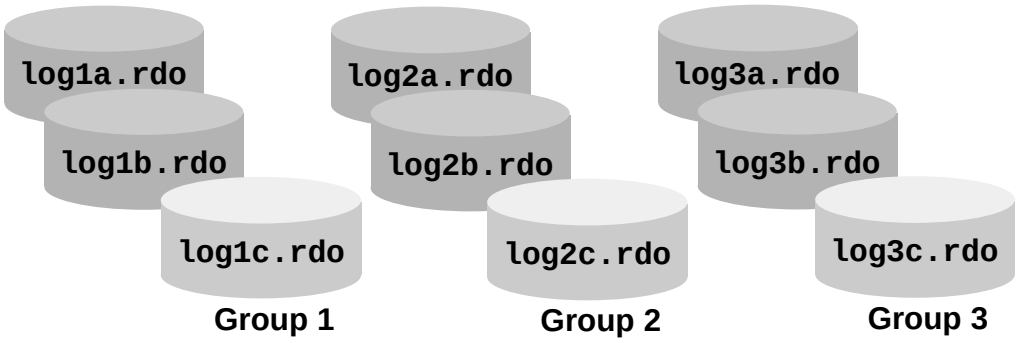
In some cases you might need to create additional log file groups. For example, adding groups can solve availability problems. To create a new group of online redo log files, use the following SQL command:

```
ALTER DATABASE [database]
      ADD LOGFILE [GROUP integer] filespec
      [,          [GROUP integer] filespec]...]
```

You specify the name and location of the members with the file specification. The value of the GROUP parameter can be selected for each redo log file group. If you omit this parameter, the Oracle server generates its value automatically.

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;
```



ORACLE

Adding Redo Log Members

You can add new members to existing redo log file groups using the following ALTER DATABASE ADD LOGFILE MEMBER command:

```
ALTER DATABASE [database]
    ADD LOGFILE MEMBER
    [     'filename' [REUSE]
       [, 'filename' [REUSE]]...
    TO {GROUP integer
       |('filename'[, 'filename']...)
       }
    ]...
```

Use the fully specified name of the log file members; otherwise the files are created in a default directory of the database server.

If the file already exists, it must have the same size, and you must specify the REUSE option. You can identify the target group either by specifying one or more members of the group or by specifying the group number.

Adding Redo Log Members (continued)

Using Storage Manager to Maintain Groups and Members

Launch Storage Manager from the Console to manage redo log groups and members.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Storage folder
4. Right-click the Redo Log Groups folder from the navigator tree, and select Create.
5. Enter your redo log group information, and specify the members. Click Create.

Note: You can also launch the Console from Windows NT Start menu

Dropping Online Redo Log Groups

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

ORACLE

Dropping a Redo Log Group

To increase or decrease the size of online redo log groups, add new online redo log groups (with the new size) and then drop the old ones.

An entire online redo log group can be dropped with the following `ALTER DATABASE DROP LOGFILE` command:

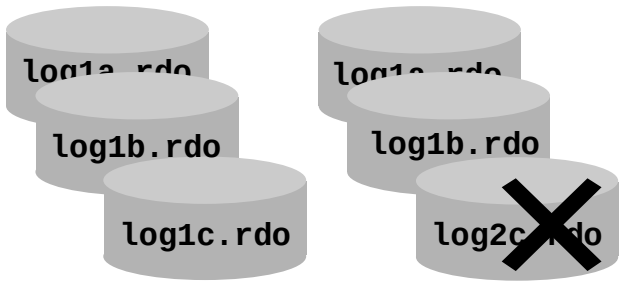
```
ALTER DATABASE [database]
DROP LOGFILE {GROUP integer|('filename'[, 'filename']...)}
                          [,{GROUP integer|('filename'[,
'filename']...)}]...
```

Restrictions

- An instance requires at least two groups of online redo log files.
- An active or current group cannot be dropped.
- When an online redo log group is dropped, the operating system files are not deleted.

Dropping Online Redo Log Members

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



ORACLE

Dropping a Redo Log Member

You may want to drop an online redo log member because it is invalid. Use the following `ALTER DATABASE DROP LOGFILE MEMBER` command if you want to drop one or more specific online redo log members:

```
ALTER DATABASE [database]
DROP LOGFILE MEMBER 'filename'[, 'filename']...
```

Restrictions

- If the member you want to drop is the last valid member of the group, you cannot drop that member.
- If the group is current, you must force a log file switch before you can drop the member.
- If the database is running in `ARCHIVELOG` mode and the log file group to which the member belongs is not archived, then the member cannot be dropped.
- When an online redo log member is dropped, the operating system file is not deleted.

Using Storage Manager to Drop Redo Log Groups and Members

Launch Storage Manager from the Console to manage redo log groups and members.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Storage folder
4. Expand the Redo Log Groups folder, and select a redo log group you want to remove.
5. Select Object—>Remove from the menu bar to remove a redo log group.

Note: You can also launch the Console from Windows NT Start menu

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.**

ORACLE

Clearing Online Redo Log Files

If a redo log file is corrupted in all members, the database administrator can solve this problem by reinitializing these log files using `ALTER DATABASE CLEAR LOGFILE`:

```
ALTER DATABASE [database]
CLEAR [UNARCHIVED] LOGFILE
       {GROUP integer|('filename'[, 'filename']...)}
      [,{GROUP integer|('filename'[, 'filename']...)}]...
```

Using this command is equivalent to adding and dropping an online redo log file. But you can issue this command even if there are only two log groups with one file each and even if the cleared group is available but not archived.

Restrictions

You can clear an online redo log file whether it is archived or not. However, when it is not archived, you must include the keyword `UNARCHIVED`. This makes backups unusable if the online redo log file is needed for recovery.

Relocating and Renaming Redo Log Files

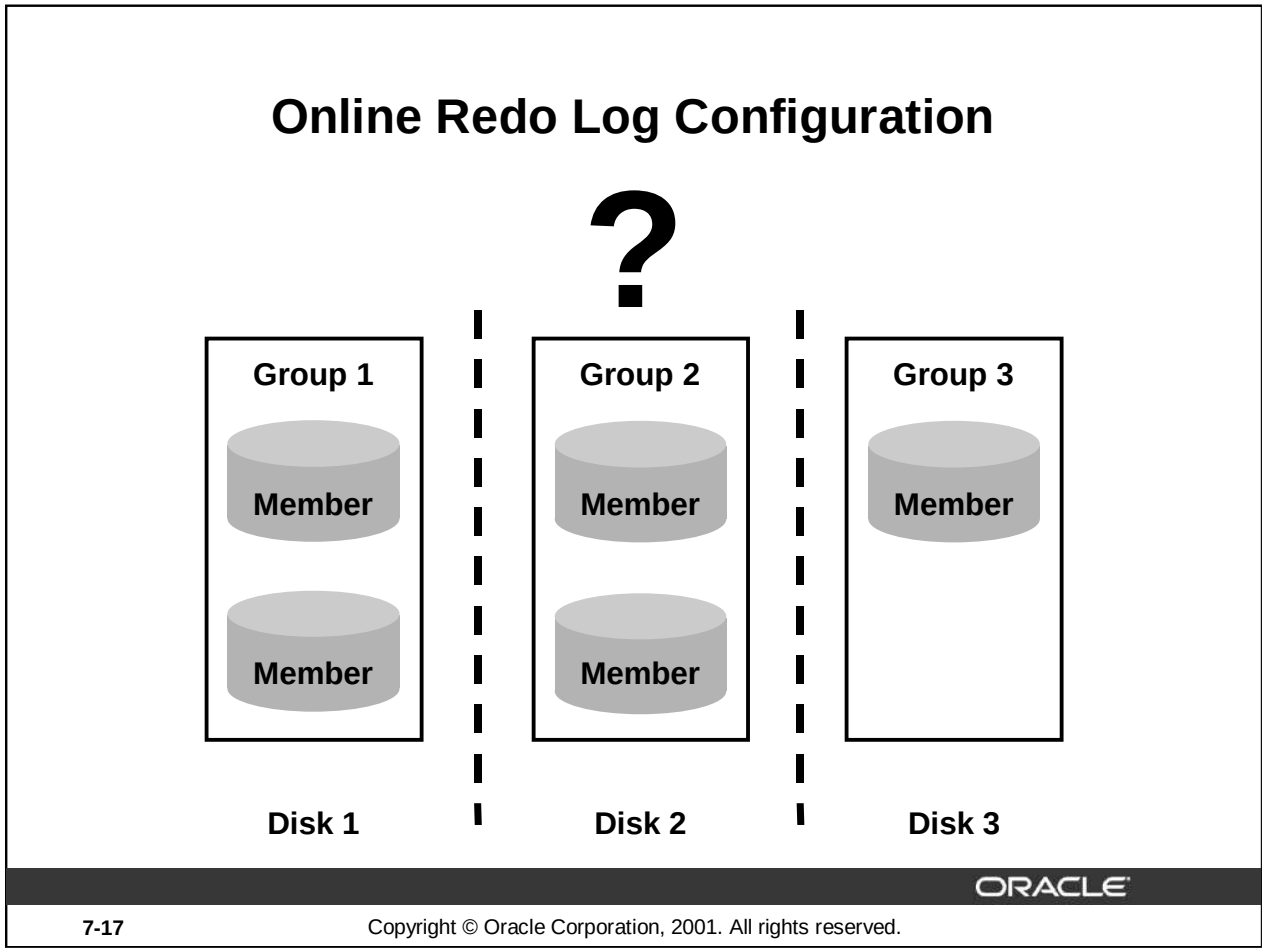
The locations of the online redo log files can be changed by adding new log files and dropping the old log files. Another method `ALTER DATABASE RENAME FILE` is available, but this requires the database to be placed in `MOUNT` mode. Therefore, it is much easier to add new ones and drop old ones.

Using Storage Manager to Relocate or Rename Redo Log Members

Launch Storage Manager from the Console to manage redo log groups and members.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Storage folder
4. Expand the Redo Log Groups folder, and select a redo log group.
5. Modify the redo log member information to rename or relocate members. Click Apply.

Note: You can also launch the Console from Windows NT Start menu



Number of Online Redo Log Files

To determine the appropriate number of online redo log files for a database instance, you have to test different configurations.

In some cases, a database instance may require only two groups. In other situations, a database instance may require additional groups to guarantee that the groups are always available to LGWR. For example, if messages in the LGWR trace file or in the alert file indicate that LGWR frequently has to wait for a group because a checkpoint has not completed or a group has not been archived, you need to add groups.

Although with the Oracle server multiplexed groups can contain different numbers of members, try to build up a symmetric configuration. An asymmetric configuration should only be the temporary result of an unusual situation such as a disk failure.

Location of Online Redo Log Files

When you multiplex the online redo log files, place members of a group on different disks. By doing this, even if one member is not available but other members are available, the instance does not shut down.

Separate archive log files and online redo log files on different disks to reduce contention between the ARCn and LGWR background processes.

Number of Online Redo Log Files (continued)

Data files and online redo log files should be placed on different disks to reduce LGWR and DBWn contention and reduce the risk of losing both data files and online redo log files in the event of media failure.

Sizing Online Redo Log Files

The minimum size of an online redo log file is 50 KB, and the maximum size is specific to the operating system. Members of different groups can have different sizes; however, there is no benefit to having different sized groups.

Different sized groups should be required as a temporary result only if you want to change the size of the members of the online redo log groups. In this case, you have to create new online redo log groups with different sizes, and then remove the old groups.

The following situations might influence the configuration of the online redo log files:

- Number of log switches and checkpoints
- Number and amount of redo entries
- Amount of space on the storage medium; for example, on a tape if archiving is enabled

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;
```

ORACLE

Managing Online Redo Log Files

Adding a Group

To create a new group of online redo log files, the DBA uses the `ALTER DATABASE ADD LOGFILE` command. The command has been modified so that the file specification is not necessary. The example in the slide adds a log file with a member in the `DB_CREATE_ONLINE_LOG_DEST_1` location and a member in the `DB_CREATE_ONLINE_LOG_DEST_2` location. Unique file names for the log file members are generated automatically.

Dropping a Group

The `GROUP` clause can be used to drop a log file. In the example above, the operating system file associated with each OMF log file member is automatically deleted.

Archived Redo Logs and OMF

Archived redo log files cannot be OMF files. A file system location for the archived log files can be specified with the `LOG_ARCHIVE_DEST_n` initialization parameters.

Obtaining Group and Member Information

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

- **V$LOG**
- **V$LOGFILE**

ORACLE

Obtaining Log Group and Member Information

V$LOG

The following query returns information about the online redo log file from the control file:

```
SQL>  SELECT group#, sequence#, bytes, members, status
   2  FROM v$log;
GROUP#          SEQUENCE#  BYTES      MEMBERS    STATUS
---------       ---------- --------   ---------  ---------
      1         688        1048576    1          CURRENT
      2         689        1048576    1          INACTIVE
2 rows selected.
```

The following items are the most common values for the STATUS column:

- UNUSED indicates that the online redo log group has never been written to. This is the state of an online redo log file that was just added.
- CURRENT indicates the current online redo log group. This implies that the online redo log group is active.
- ACTIVE indicates that the online redo log group is active but is not the current online redo log group. It is needed for crash recovery. It may be in use for block recovery. It may or may not be archived.

Obtaining Log Group and Member Information (continued)

- CLEARING indicates the log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE command. After the log is cleared, the status changes to UNUSED.
- CLEARING_CURRENT indicates that the current log file is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch, such as an I/O error writing the new log header.
- INACTIVE indicates that the online redo log group is no longer needed for instance recovery. It may or may not be archived.

V$LOGFILE

To obtain the names of all the members of a group, query the dynamic performance view V$LOGFILE.

```
SQL> SELECT member FROM V$LOGFILE;

MEMBER
-----------------------------------
/u01/home/db03/ORADATA/u03/log02a.rdo
/u01/home/db03/ORADATA/u03/log01a.rdo
```

The value of the STATUS column could be one of the following:

- INVALID indicates that the file is inaccessible.
- STALE indicates that contents of the file are incomplete.
- DELETED indicates that the file is no longer used.
- Blank indicates that the file is in use.

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.**

ORACLE

Archived Redo Log Files

One of the important decisions that a database administrator has to make is whether the database is configured to operate in ARCHIVELOG mode or in NOARCHIVELOG mode.

NOARCHIVELOG

In NOARCHIVELOG mode, the online redo log files are overwritten each time an online redo log file is filled, and log switches occur. LGWR does not overwrite a redo log group until the checkpoint for that group is completed.

ARCHIVELOG

If the database is configured to run in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived. Because all changes made to the database are recorded in the online redo log files, the database administrator can use the physical backup and the archived online redo log files to recover the database without losing any committed data.

There are two ways in which online redo log files can be archived:

- Manually
- Automatically: Recommend method

Archived Redo Log Files (continued)

ARCHIVELOG (continued)

The LOG_ARCHIVE_START initialization parameter indicates whether archiving should be automatic or manual when the instance starts up.

- TRUE indicates that archiving is automatic. ARCn initiate archiving of the filled log group at every log switch.
- FALSE, the default value, indicates that the database administrator archives filled redo log files manually. The database administrator must manually execute a command each time you want to archive an online redo log file. All or specific online redo log files can be archived manually.

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.**

ORACLE

Archived Redo Log Files

Information about archived logs can be obtained from V$INSTANCE.

```
SQL> SELECT archiver
     2 FROM v$instance;

ARCHIVE
---------
STOPPED
1 row selected.
```

Note: Archiving is covered in detail in the *DBA Fundamentals II* course.

Archived Redo Log Files (continued)

Using Instance Manager to Obtain Archive Information

Launch Instance Manager from the Console, and obtain information about archiving

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand on Instance in the navigator tree and click Configuration
4. Click the Recovery tab on the detail side of the Console to obtain archive information

Note: You can also launch the Console from Windows NT Start menu

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**

ORACLE

Quick Reference

Context	Reference
Initialization parameters	LOG_CHECKPOINTS_TO_ALERT UTL_FILE_DIR
Dynamic initialization parameters	LOG_CHECKPOINT_TIMEOUT LOG_CHECKPOINT_INTERVAL FAST_START_MTTR_TARGET
Dynamic performance views	V$THREAD V$LOG V$LOGFILE V$DATABASE
Data dictionary views	None
Commands	ALTER SYSTEM SWITCH LOGFILE ALTER SYSTEM CHECKPOINT ARCHIVE LOG LIST ALTER DATABASE ADD LOGFILE ALTER DATABASE ADD LOGFILE MEMBER ALTER DATABASE RENAME FILE ALTER DATABASE DROP LOGFILE ALTER DATABASE DROP LOGFILE MEMBER ALTER DATABASE CLEAR LOGFILE

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.**

ORACLE

Practice 7: Maintaining Redo Log Files

1 List the number and location of existing log files and display the number of redo log file groups and members your database has.

Hints:

- Query the dynamic view `V$LOGFILE`.
- Use the dynamic view `V$LOG`.

2 In which database mode is your database configured? Is archiving enabled?

Hints:

- Query the dynamic view `V$DATABASE`.
- Query the dynamic view `V$INSTANCE`.

3 Add a redo log member to each group in your database located on `u04`, using the following naming conventions:

Add member to Group 1: `log01b.rdo`
Add member to Group 2: `log02b.rdo`

Verify the result.

Hints:

- Execute the `ALTER DATABASE ADD LOGFILE MEMBER` command to add a redo log member to each group.
- Query the dynamic performance view `V$LOGFILE` to verify the result.

4 Add a redo log group in your database with two members located on `u03` and `u04` using the following naming conventions:

Add Group 3: `log03a.rdo` and `log03b.rdo`

Verify the result.

Hints:

- Execute the `ALTER DATABASE ADD LOGFILE` command to create a new group.
- Query the Dynamic View `V$LOGFILE` to display the name of the new members of the new group.
- Query the Dynamic View `V$LOG` to display the number of redo log file groups and members.

Practice 7: Maintaining Redo Log Files (Continued)

5 Remove the redo log group created in step 4.

Hints:

- Execute the `ALTER DATABASE DROP LOGFILE GROUP` command to remove the log group.
- Query the Dynamic View `V$LOG` to verify the result.
- Remove the operating system files for the group.

6 Resize all online redo log files to 1024 KB. (Because we cannot resize log files, we have to add new logs and drop the old.)

Hints:

- Execute the `ALTER DATABASE ADD LOGFILE GROUP` command to add two new groups with the size 1024 KB.
- Query the Dynamic View `V$LOG` to check the active group.
- Execute the `ALTER SYSTEM SWITCH LOGFILE` command to force log switches and change the group stage to inactive. The number of log switches required will vary.
- Execute the `ALTER DATABASE DROP LOGFILE` command to remove the inactive groups.
- Query the Dynamic View `V$LOG` to verify the result.

8

Managing Tablespaces and Data files

ORACLE

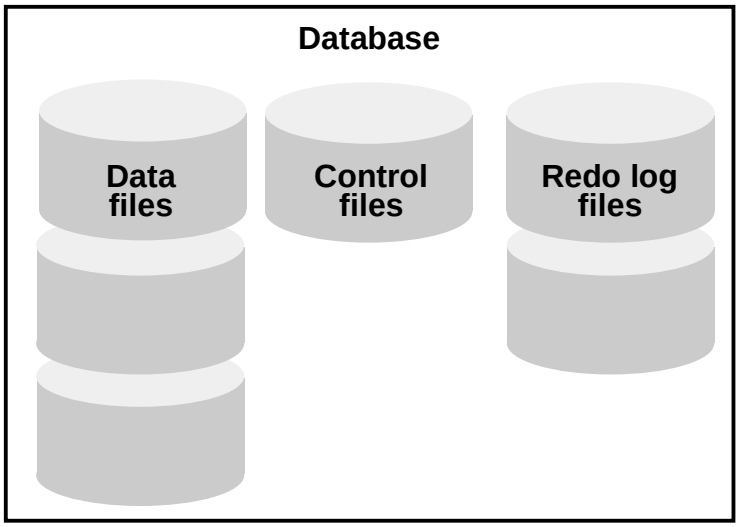
Objectives

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**

ORACLE

Overview

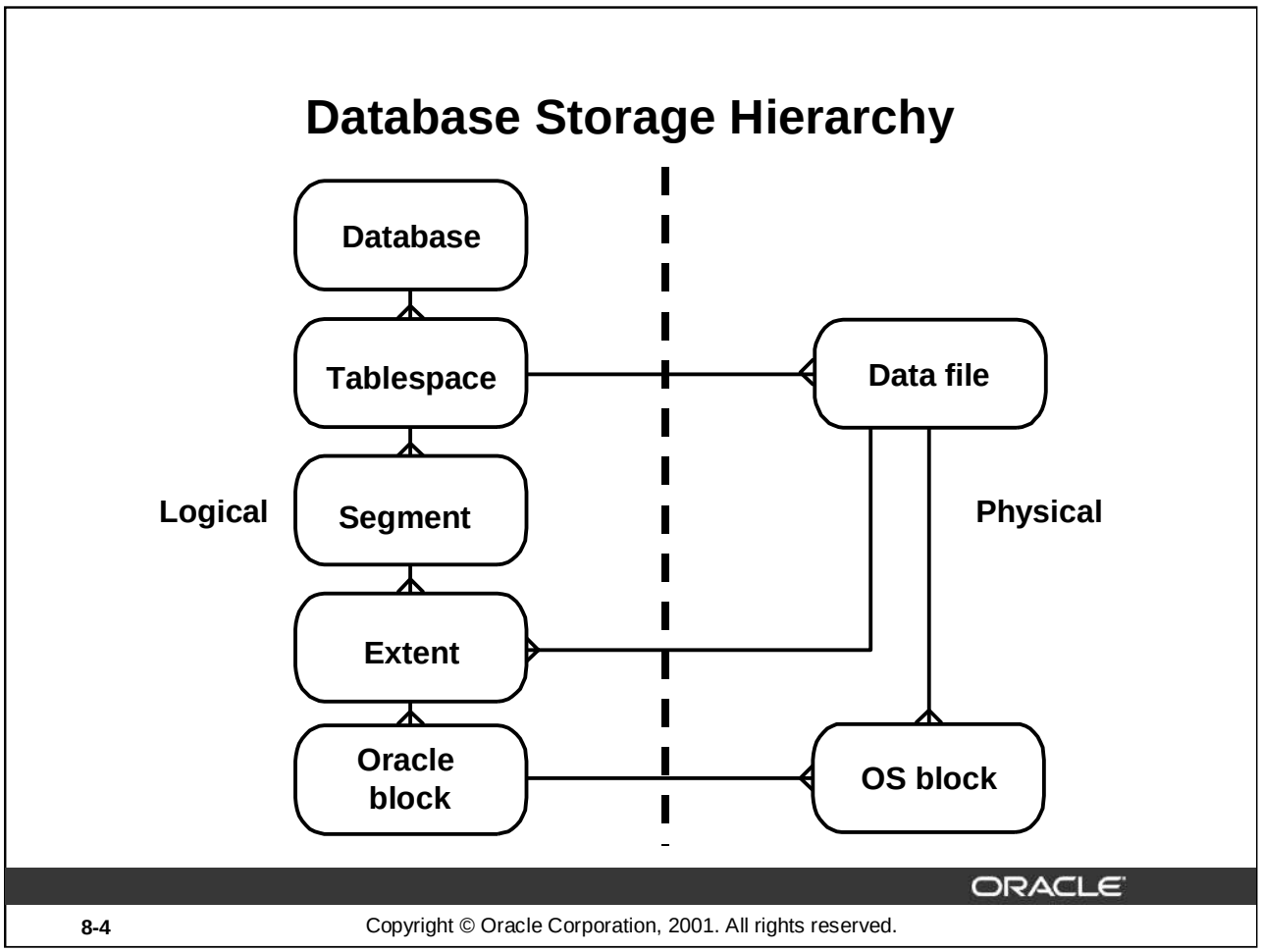


ORACLE

Overview

A small database might need only the SYSTEM tablespace; however, Oracle recommends that you create additional tablespaces to store user data, user indexes, undo segments, and temporary segments separate from data dictionary. This gives you more flexibility in various database administration operations and reduces contention among dictionary objects and schema objects for the same data files.

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



Database Architecture

The Oracle database architecture includes logical and physical structures that make up the database.

- The physical structure includes the control files, online redo log files, and data files that make up the database.
- The logical structure includes tablespaces, segments, extents, and data blocks.

The Oracle server enables fine-grained control of disk space use through tablespace and logical storage structures, including segments, extents, and data blocks.

Tablespaces

The data in an Oracle database are stored in tablespaces.

- An Oracle database can be logically grouped into smaller logical areas of space known as tablespaces.
- A tablespace can belong to only one database at a time.
- Each tablespace consists of one or more operating system files, which are called data files.

Database Architecture (continued)

- A tablespace may consist of zero or more segments.
- Tablespaces can be brought online while the database is running.
- Except for the SYSTEM tablespace or a tablespace with an active undo segment, tablespaces can be taken offline, leaving the database running.
- Tablespaces can be switched between read-write and read-only status.

Data Files

- Each tablespace in an Oracle database consists of one or more files called data files. These are physical structures that conform with the operating system on which the Oracle server is running.
- A data file can belong to only one tablespace.
- An Oracle server creates a data file for a tablespace by allocating the specified amount of disk space plus a small amount of overhead.
- The database administrator can change the size of a data file after its creation or can specify that a data file should dynamically grow as objects in the tablespace grow.

Segments

- A segment is the space allocated for a specific logical storage structure within a tablespace. For example, all of the storage allocated to a table is a segment.
- A tablespace may consist of one or more segments.
- A segment cannot span tablespaces; however, a segment can span multiple data files that belong to the same tablespace.
- Each segment is made up of one or more extents.

Extents

Space is allocated to a segment by extents.

- One or more extents make up a segment.
  - When a segment is created, it consists of at least one extent.
  - As the segment grows, extents get added to the segment.
  - The DBA can manually add extents to a segment.
- An extent is a set of contiguous Oracle blocks.
- An extent cannot span a data file but must exist in one data file.

Database Architecture (continued)

Data Blocks

The Oracle server manages the storage space in the data files in units called Oracle blocks or data blocks.

- At the finest level of granularity, the data in an Oracle database is stored in data blocks.
- Oracle data blocks are the smallest units of storage that the Oracle server can allocate, read, or write.
- One data block corresponds to one or more operating system blocks allocated from an existing data file.
- The standard data block size for an Oracle database is specified by the `DB_BLOCK_SIZE` initialization parameter when the database is created.
- The data block size should be a multiple of the operating system block size to avoid unnecessary I/O.
- The maximum data block size is dependent on the operating system.

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**

ORACLE

Types of Tablespaces

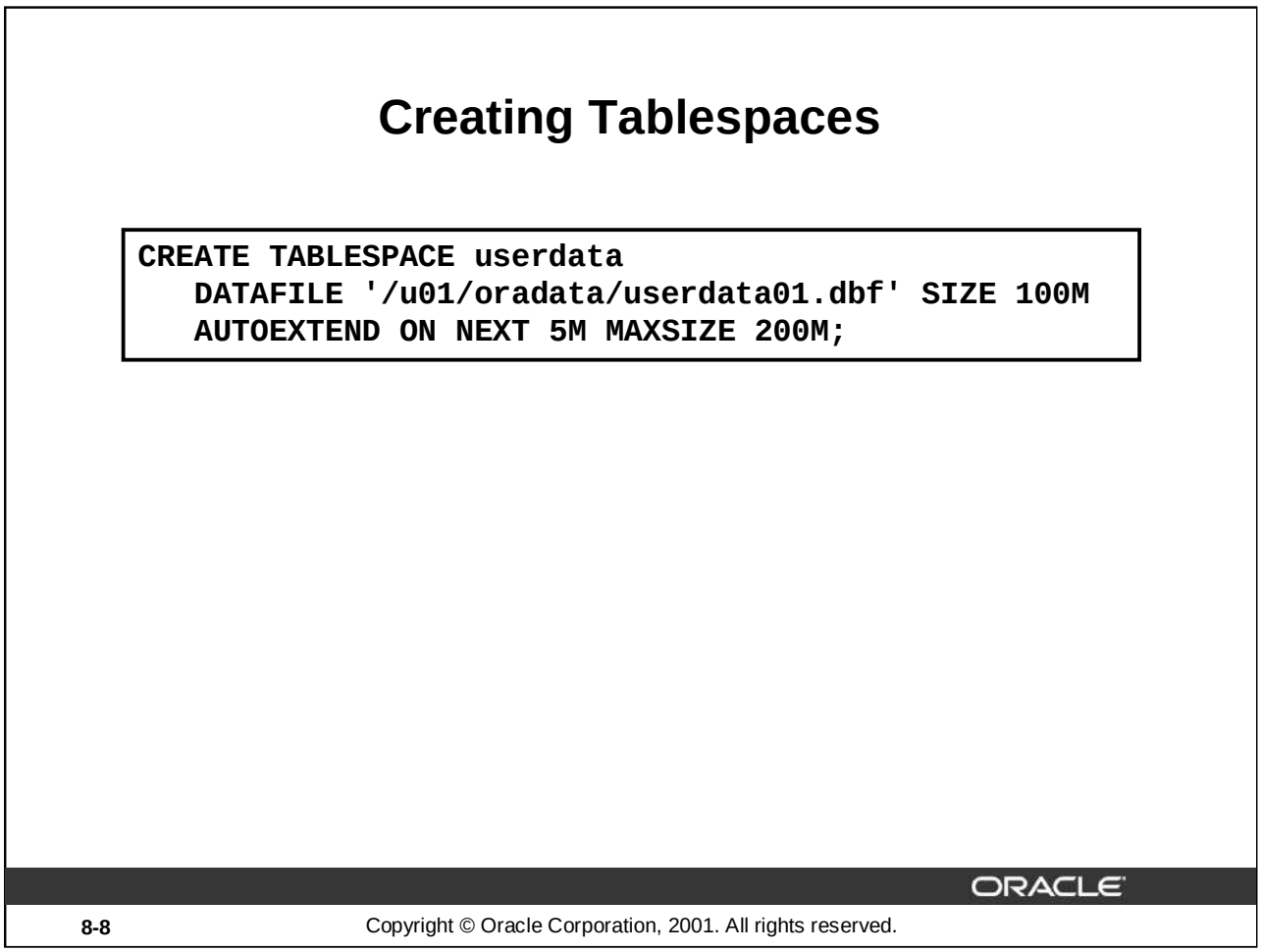
The DBA creates tablespaces for increased control and ease of maintenance. The Oracle server perceives two types of tablespaces: SYSTEM and all others.

SYSTEM Tablespace

- Created with the database
- Required in all databases
- Contains the data dictionary, including stored program units
- Contains the SYSTEM undo segment
- Should not contain user data, although it is allowed

Non-SYSTEM Tablespaces

- Enable more flexibility in database administration
- Separate undo, temporary, application data, and application index segments
- Separate data by backup requirements
- Separate dynamic and static data
- Control the amount of space allocated to user's objects



CREATE TABLESPACE Command

You create a tablespace with the CREATE TABLESPACE command:

```
CREATE TABLESPACE tablespace
    [DATAFILE clause]
    [MINIMUM EXTENT integer[K|M]]
    [BLOCKSIZE integer [K]]
    [LOGGING|NOLOGGING]
    [DEFAULT storage_clause ]
    [ONLINE|OFFLINE]
    [PERMANENT|TEMPORARY]
```

where: *tablespace* is the name of the tablespace to be created

DATAFILE specifies the data file or data files that make up the tablespace

CREATE TABLESPACE Command (continued)

`MINIMUM EXTENT`	ensures that every used extent size in the tablespace is a multiple of the *`integer`*. Use `K` or `M` to specify this size in kilobytes or megabytes.
`LOGGING`	specifies that, by default, all tables, indexes, and partitions within the tablespace have all changes written to redo. `LOGGING` is the default.
`NOLOGGING`	specifies that, by default, all tables, indexes, and partitions within the tablespace do not have all changes written to redo. `NOLOGGING` affects only some DML and DDL commands, for example, direct loads.
`DEFAULT`	specifies the default storage parameters for all objects created in the tablespace creation
`OFFLINE`	makes the tablespace unavailable immediately after creation
`PERMANENT`	specifies that the tablespace can be used to hold permanent objects
`TEMPORARY`	specifies that the tablespace be used only to hold temporary objects; for example, segments used by implicit sorts caused by an `ORDER BY` clause `extent_management_clause` specifies how the extents of the tablespace are managed. This clause is discussed in a subsequent section of this lesson.

```
datafile_clause :== filename
[SIZE integer[K|M] [REUSE] | REUSE ] [ autoextend_clause ]
```

where:		
	`filename`	is the name of a data file in the tablespace
	`SIZE`	specifies the size of the file. Use `K` or `M` to specify the size in kilobytes or megabytes.
	`REUSE`	allows the Oracle server to reuse an existing file
	`autoextend_clause`	enables or disables the automatic extension of the data file. This clause is discussed in a subsequent section of this lesson.

See Also: *Oracle9i SQL Reference* and *Oracle9i Concepts* for additional information.

CREATE TABLESPACE Command (continued)

Using Console to Create a New Tablespace

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Select the Tablespaces folder, and select Create from the right mouse menu.
6. In the General tab of the property sheet, enter the tablespace name.
7. In the Datafiles region, specify data file.
8. In the Storage Tab of the property sheet, enter storage information.
9. Click Create.

Note: You can also launch the Console from Windows NT Start menu

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**

ORACLE

Choosing a Space Management Method

Tablespace extents can be managed with data dictionary tables or bitmaps. When you create a tablespace, you choose one of these methods of space management. You cannot alter the method at a later time.

Locally Managed Tablespaces

A tablespace that manages its own extents maintains a bitmap in each datafile to keep track of the free or used status of blocks in that data file. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, the Oracle server changes the bitmap values to show the new status of the blocks.

Dictionary-Managed Tablespaces

For a tablespace that uses the data dictionary to manage its extents, the Oracle server updates the appropriate tables in the data dictionary whenever an extent is allocated or deallocated.

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**

ORACLE

Locally Managed Tablespaces

The LOCAL option of the EXTENT MANAGEMENT clause specifies that a tablespace is to be locally managed. By default a tablespace is locally managed.

```
extent_management_clause :==
    [ EXTENT MANAGEMENT
       [ DICTIONARY | LOCAL
         [ AUTOALLOCATE | UNIFORM [SIZE integer[K|M]] ] ] ]
```

where:

DICTIONARY	specifies that the tablespace is managed using dictionary tables.
LOCAL	specifies that tablespace is locally managed with a bitmap. If you specify LOCAL, you cannot specify DEFAULT storage_clause, MINIMUM EXTENT, or TEMPORARY.
AUTOALLOCATE	specifies that the tablespace is system managed. Users cannot specify an extent size. This is the default.

Locally Managed Tablespaces (continued)

UNIFORM	specifies that the tablespace is managed with uniform extents of SIZE bytes. Use K or M to specify the extent size in kilobytes or megabytes. The default size is 1 megabyte.

The EXTENT MANAGEMENT clause can be used in various CREATE commands:

- For a permanent tablespace other than SYSTEM, you can specify EXTENT MANAGEMENT LOCAL in the CREATE TABLESPACE command.
- For a temporary tablespace, you can specify EXTENT MANAGEMENT LOCAL in the CREATE TEMPORARY TABLESPACE command.

Advantages of Locally Managed Tablespaces

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

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

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

ORACLE

Dictionary Managed Tablespaces

Segments in dictionary managed tablespaces can have a customized storage, this is more flexible than locally managed tablespaces but much less efficient.

Changing the Storage Settings

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

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

ORACLE

Changing Default Storage Settings

Use the ALTER TABLESPACE command to alter the default storage definition of a tablespace:

```
ALTER TABLESPACE tablespace
    [MINIMUM EXTENT integer[K|M]
    |DEFAULT storage_clause ]
```

The storage settings for locally managed tablespaces cannot be altered.

Changing Default Storage Settings (continued)

Using Console to Change the Storage Settings

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Expand the Tablespaces folder.
6. Right-click the tablespace, and select View/Edit Details from the menu.
7. Click the Storage tab.
8. Make changes, and click Apply.

Note: You can also launch the Console from Windows NT Start menu

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;
```

ORACLE

Undo Tablespace

An undo tablespace is used with automatic undo management. Automatic undo management is covered in the lesson "Managing Undo Data." Unlike other tablespaces, the undo tablespace is limited to the DATAFILE.

```
CREATE UNDO TABLESPACE tablespace
   [DATAFILE clause]
```

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;
```

ORACLE

Temporary Segments

You can manage space for sort operations more efficiently by designating temporary tablespaces exclusively for sort segments. No permanent schema objects can reside in a temporary tablespace.

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

`CREATE TEMPORARY TABLESPACE` Command

Although the `ALTER/CREATE TABLESPACE...TEMPORARY` command can be used to create a temporary tablespace, it is recommended that the `CREATE TEMPORARY TABLESPACE` command be used.

Temporary Segments (continued)

Locally managed temporary tablespaces have temporary data files (tempfiles), which are similar to ordinary data files except that:

- Tempfiles are always set to NOLOGGING mode.
- You cannot make a tempfile read-only.
- You cannot rename a tempfile.
- You cannot create a tempfile with the ALTER DATABASE command.
- Tempfiles are required for read-only databases.
- Media recovery does not recover tempfiles.
- BACKUP CONTROLFILE does not generate any information for tempfiles.
- CREATE CONTROLFILE cannot specify any information about tempfiles.

To optimize the performance of a sort in a temporary tablespace, set the UNIFORM SIZE to be a multiple of the parameter SORT_AREA_SIZE.

Temporary Segments (continued)

Using Console to Create a Temporary Tablespace

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Select the Tablespaces folder, and select Create from the right mouse menu.
6. Supply details in the General tab of the property sheet, and select the Temporary option in the Type region
7. Click the Storage tab, and enter the storage information.
8. Click Create.

Note: You can also launch the Console from Windows NT Start menu

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**

ORACLE

Default Temporary Tablespace

When creating a database without a default temporary tablespace the default tablespace, assigned to any user created without a TEMPORARY TABLESPACE clause is the SYSTEM tablespace. Also a warning is placed in the `alert_sid.log` stating that the SYSTEM tablespace is the default temporary tablespace. Creating a default temporary tablespace during database creation prevents the SYSTEM tablespace from being used for temporary space.

After database creation, a default temporary tablespace can be set by creating a temporary tablespace and then altering the database.

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp;
```

Once defined, users not explicitly assigned to a temporary tablespace are assigned to the default temporary tablespace.

The default temporary database can be changed at any time by using the ALTER DATABASE DEFAULT TEMPORARY TABLESPACE command. When the default temporary tablespace is changed, all users assigned the default temporary tablespace are reassigned to the new default.

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.

ORACLE

Restrictions on Default Temporary Tablespace

Dropping a Default Temporary Tablespace

You cannot drop the default temporary tablespace until after a new default is made available. The ALTER DATABASE command must be used to change the default temporary tablespace to a new default. The old default temporary tablespace is then dropped only after a new default temporary tablespace is made available. Users assigned to the old default temporary tablespace are automatically reassigned to the new default temporary tablespace.

Changing to a Permanent Type Versus Temporary Type

Because a default temporary tablespace must be either the SYSTEM tablespace or a Temporary tablespace, you cannot change the default temporary tablespace to a permanent type.

Taking Default Temporary Tablespace Offline

Tablespaces are taken offline to make that part of the database unavailable to other users (for example, an offline backup, maintenance, or making a change to an application that uses the tablespace). Becuase none of these situations apply to a temporary tablespace, you cannot take a default temporary tablespace offline.

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;
```

ORACLE

Taking a Tablespace Offline

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

- Make a portion of the database unavailable, while allowing normal access to the remainder of the database
- Perform an offline tablespace backup (although a tablespace can be backed up while online and in use)
- Recover a tablespace or data file while the database is open
- Move a data file while the database is open

The Offline Status of a Tablespace

When a tablespace goes offline, Oracle does not permit any subsequent SQL statements to reference objects contained in that tablespace. Users trying to access objects in a tablespace that is offline receive an error.

When a tablespace goes offline or comes back online, the event is recorded in the data dictionary and in the control file. If a tablespace is offline when you shut down a database, the tablespace remains offline and is not checked when the database is subsequently mounted and reopened.

Taking a Tablespace Offline (continued)

The Oracle instance automatically switches a tablespace from online to offline when certain errors are encountered (for example, when the Database Writer process, DBW0, fails in several attempts to write to a data file of the tablespace). The different error situations are covered in more detail in the course *Oracle9i: Equivalent of Backup and Recovery Workshop.*

Taking Tablespaces Offline

Whenever the database is open, a database administrator can take any tablespace, except the SYSTEM tablespace or any tablespace with active undo segments or temporary segments, offline. When a tablespace is taken offline, the Oracle server takes all the associated data files offline.

```
ALTER TABLESPACE tablespace
    {ONLINE
    |OFFLINE [NORMAL|TEMPORARY|IMMEDIATE|FOR RECOVER]}
```

where:

`NORMAL`	flushes all blocks in all data files in the tablespace out of the SGA. This is the default. You need not perform media recovery on this tablespace before bringing it back online. Use the `NORMAL` clause whenever possible.
`TEMPORARY`	performs a checkpoint for all online data files in the tablespace only. Any offline files may require media recovery.
`IMMEDIATE`	does not ensure that tablespace files are available and does not perform a checkpoint. You must perform media recovery on the tablespace before bringing it back online.
`FOR RECOVER`	takes tablespaces offline for tablespace point-in-time recovery.

Taking a Tablespace Offline (continued)

Using Console to Take a Tablespace Offline

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Expand the Tablespaces folder.
6. Select the tablespace.
7. In the Status region of the General tab, select Offline.
8. Select the Mode from the drop-down menu.
9. Click Apply.

Note: You can also launch the Console from Windows NT Start menu

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…;**

ORACLE

The ALTER TABLESPACE...READ ONLY Command

Making tablespaces read-only prevents further write operations on the data files in the tablespace. Therefore, the data files can reside on read-only media, such as CD-ROMs or write-once (WORM) drives. Read-only tablespaces eliminate the need to perform backups of large, static portions of a database. Use the SQL command ALTER TABLESPACE to change a tablespace to read-only or read-write:

```
ALTER TABLESPACE tablespace READ [ONLY | WRITE]
```

To create a read-only tablespace on a write-once device:

1. ALTER TABLESPACE...READ ONLY.
2. Use an operating system command to move the data files of the tablespace to the read-only device.
3. ALTER TABLESPACE...RENAME DATAFILE.

Making Tablespaces Read-Only

The ALTER TABLESPACE...READ ONLY command places the tablespace in a transitional read-only mode. In this transitional state, no further write operations can take place in the tablespace except for the rollback of existing transactions that previously modified blocks in the tablespace. After all of the existing transactions have been either committed or rolled back, the ALTER TABLESPACE...READ ONLY command completes, and the tablespace is placed in read-only mode.

The ALTER TABLESPACE . . . READ ONLY Command (continued)

You can drop items, such as tables and indexes, from a read-only tablespace, because these commands affect only the data dictionary. This is possible because the DROP command updates only the data dictionary, but not the physical files that make up the tablespace. For locally managed tablespaces, the dropped segment is changed to a temporary segment, to prevent the bitmap from being updated.

To make a read-only tablespace writable, all of the data files in the tablespace must be online.

Making tablespaces read-only causes a checkpoint on the data files of the tablespace.

The `ALTER TABLESPACE . . . READ ONLY` Command (continued)

Using the Console to Make a Tablespace Read-Only

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Expand the Tablespaces folder.
6. Select the tablespace.
7. Select the Read Only check box in the Status region of the General tab.
8. Click Apply.

Note: You can also launch the Console from Windows NT Start menu

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;
```

ORACLE

DROP TABLESPACE Command

You can remove a tablespace from the database when the tablespace and its contents are no longer required with the following DROP TABLESPACE SQL command:

```
DROP TABLESPACE tablespace
[INCLUDING CONTENTS [AND DATAFILES] [CASCADE CONSTRAINTS]]
```

where:		
	tablespace	specifies the name of the tablespace to be dropped
	INCLUDING CONTENTS	drops all the segments in the tablespace
	AND DATAFILES	deletes the associated operating system files
	CASCADE CONSTRAINTS	drops referential integrity constraints from tables outside the tablespace that refer to primary and unique keys in the tables in the dropped tablespace

DROP TABLESPACE Command (continued)

Guidelines

- A tablespace that still contains data cannot be dropped without the INCLUDING CONTENTS option. This option may generate a lot of undo when the tablespace contains many objects.
- After a tablespace has been dropped, its data is no longer in the database.
- When a tablespace is dropped, only the file pointers in the control file of the associated database are dropped. The operating system files still exist and must be deleted explicitly using the appropriate operating system command unless the AND DATAFILES clause is used.
- Even if a tablespace is switched to read-only, it can still be dropped, along with segments within it.
- It is recommended that you take the tablespace offline before dropping it to ensure that no transactions access any of the segments in the tablespace.

DROP TABLESPACE Command (continued)

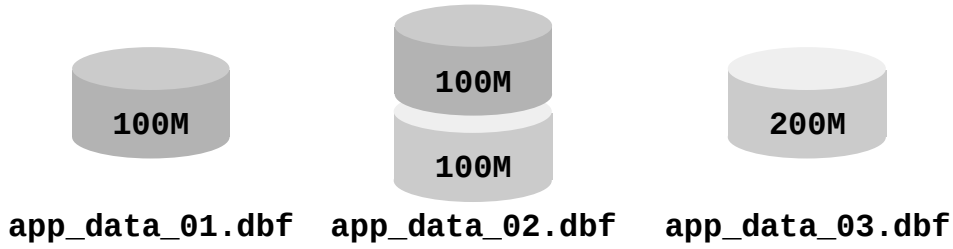
Using the Console to Drop a Tablespace

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Expand the Tablespaces folder, and select the tablespace.
6. Select Object > Remove from menu.
7. Click Yes in the dialog box to confirm.

Note: You can also launch the Console from Windows NT Start menu

Resizing a Tablespace

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



ORACLE

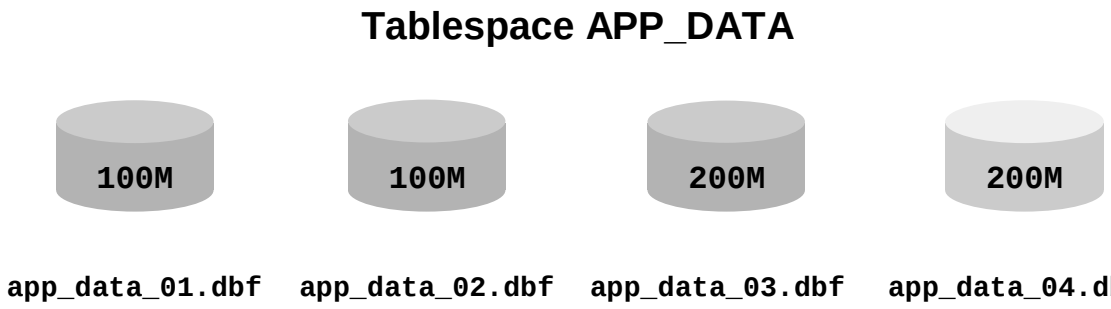
Increasing the Tablespace Size

You can enlarge a tablespace in two ways:

- Change the size of a data file, either automatically or manually.
- Add a data file to a tablespace.

Enabling Automatic Extension of Data Files

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



ORACLE

Specifying AUTOEXTEND for a New Data File

The AUTOEXTEND clause enables or disables the automatic extension of data files.

When a data file is created, the following SQL commands can be used to enable automatic extension of the data file:

- CREATE DATABASE
- CREATE TABLESPACE ... DATAFILE
- ALTER TABLESPACE ... ADD DATAFILE

Use the ALTER DATABASE command to modify a data file and enable automatic extension:

```
ALTER DATABASE DATAFILE filespec [autoextend_clause]
```

Specifying AUTOEXTEND for a New Data File (continued)

```
autoextend_clause:== [ AUTOEXTEND { OFF|ON[NEXT integer[K|M]]
    [MAXSIZE UNLIMITED | integer[K|M]] } ]
```

where:		
	AUTOEXTEND OFF	disables the automatic extension of the data file
	AUTOEXTEND ON	enables the automatic extension of the data file
	NEXT	specifies the disk space to allocate to the data file when more extents are required
	MAXSIZE	specifies the maximum disk space allowed for allocation to the data file
	UNLIMITED	sets no limit on allocating disk space to the data file

Specifying AUTOEXTEND for an Existing Data File

Use the SQL command ALTER DATABASE to enable or disable automatic file extension for existing datafiles:

```
ALTER DATABASE [database]
    DATAFILE 'filename'[, 'filename']...    autoextend_clause
```

Specifying AUTOEXTEND for a New Data File (continued)

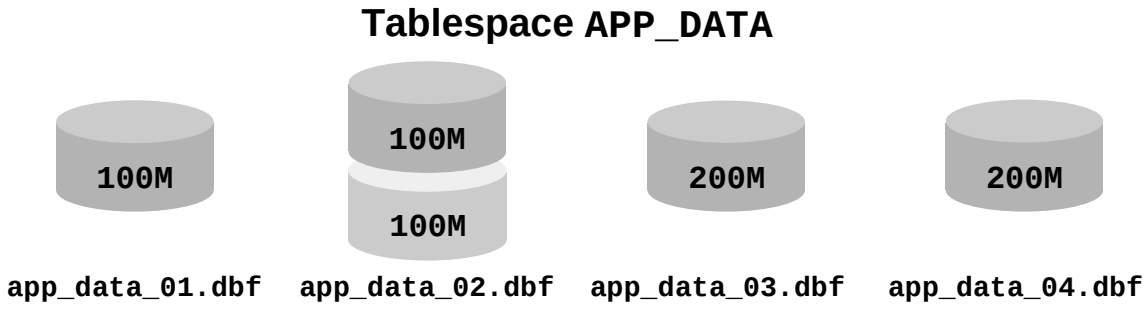
Using the Console to Enable Automatic Resizing

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand the Storage folder
5. Expand the Datafiles folder.
6. Select the data file.
7. In the Storage tab of the property sheet, select the "Automatically extend datafile when full" check box.
8. Set the values for the increment and the maximum size.
9. Click Apply.

Note: You can also launch the Console from Windows NT Start menu

Changing the Size of Data Files Manually

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



ORACLE

The ALTER DATABASE DATAFILE RESIZE Command

Instead of adding space to the database by adding data files, the DBA can change the size of a data file. Use the ALTER DATABASE command to manually increase or decrease the size of a data file:

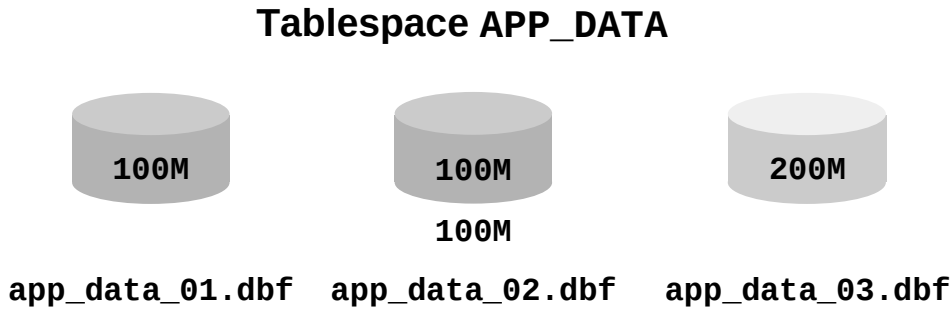
```
ALTER DATABASE [database]
    DATAFILE 'filename'[, 'filename']...
    RESIZE integer[K|M]
```

where: `integer` is the absolute size, in bytes, of the resulting data file

If there are database objects stored above the specified size, then the data file size is decreased only to the last block of the last objects in the data file.

Adding Data Files to a Tablespace

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



ORACLE

The ALTER TABLESPACE ADD DATAFILE Command

You can add data files to a tablespace to increase the total amount of disk space allocated for the tablespace with the ALTER TABLESPACE ADD DATAFILE command:

```
ALTER TABLESPACE tablespace
    ADD DATAFILE
                  filespec [autoextend_clause]
    [,            filespec [autoextend_clause]]...
```

Using the `ALTER TABLESPACE ADD DATAFILE` Command (continued)

Using the Console to Add a Data file

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Expand the Tablespaces folder.
6. Right click the tablespace, and select Add Datafile.
7. In the General tab of the property sheet, enter the file information.
8. Click Create.

Note: You can also launch the Console from Windows NT Start menu

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';
```

ORACLE

Methods for Moving Data Files

Depending on the type of tablespace, the database administrator can move data files using one of the following two methods:

The ALTER TABLESPACE Command

The following ALTER TABLESPACE command is applied only to data files in a non-SYSTEM tablespace that does not contain active undo or temporary segments:

```
ALTER TABESPACE tablespace
    RENAME DATAFILE 'filename'[, 'filename']...
                TO 'filename'[, 'filename']...
```

Use the following process to rename a data file:

1. Take the tablespace offline.
2. Use an operating system command to move or copy the files.
3. Execute the ALTER TABLESPACE RENAME DATAFILE command.
4. Bring the tablespace online.
5. Use an operating system command to delete the file if necessary.

The source filenames must match the names stored in the control file.

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';
```

ORACLE

The ALTER DATABASE Command

The ALTER DATABASE command (see the lesson "Maintaining Redo Log Files") can be used to move any type of data file:

```
ALTER DATABASE [database]
    RENAME FILE 'filename'[, 'filename']...
             TO 'filename'[, 'filename']...
```

Because the SYSTEM tablespace cannot be taken offline, you must use this method to move data files in the SYSTEM tablespace.

Use the following process to rename files in tablespaces that cannot be taken offline:

1. Shut down the database.
2. Use an operating system command to move the files.
3. Mount the database.
4. Execute the ALTER DATABASE RENAME FILE command.
5. Open the database.

Using the ALTER DATABASE Command (continued)

Using Oracle DBA Studio to Add a Data File

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand the Tablespaces folder
4. Select the data file.
5. In the General tab of the property sheet, update the file information.
6. Click Apply.

Note:

- These commands verify that the file exists in the new location; they do not create or move files.
- Always provide complete filenames (including their paths) to identify the old and new data files.

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';
```

ORACLE

OMF Configurations

When configuring Oracle Managed Files (OMF) for creating tablespaces, a single initialization parameter, `DB_CREATE_FILE_DEST`, is specified. All data files are created in the specified file system location. The parameter can be set in the initialization file or set dynamically using the `ALTER SYSTEM` command.

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.**

ORACLE

Creating Tablespaces with OMF

Creating tablespaces with OMF does not require a `DATAFILE` clause. Tablespaces omitting the `DATAFILE` clause take the defaults of a 100M data file set to autoextend with an unlimited `MAXSIZE`. Optionally a file size may be specified.

```
CREATE TABLESPACE tablespace
      [ DATAFILE [ filename ] [ SIZE integer [K|M] ] ];
```

Data files from OMF created tablespaces are deleted at the OS level when the associated tablespace is dropped.

`DB_CREATE_ONLINE_LOG_DEST_n` should be set to prevent log files and control files from being placed with data files.

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**

ORACLE

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 tablesapces**
- **Implement Oracle Managed Files**

ORACLE

Quick Reference

Context	Reference
Initialization parameters	DB_FILES
Dynamic performance views	V$DATAFILE V$TEMPFILE V$TABLESPACE
Data dictionary views	DBA_DATA_FILES DBA_TABLESPACES DBA_TEMPFILES
Commands	CREATE TABLESPACE…DATAFILE…DEFAULT STORAGE…MINIMUN EXTENT CREATE TEMPORARY TABLESPACE CREATE TABLESPACE…DATAFILE AUTOEXTEND ALTER TABLESPACE…ADD DATAFILE…AUTOEXTEND ALTER DATABASE DATAFILE…RESIZE ALTER TABLESPACE…DEFAULT STORAGE… MINIMUM EXTENT ALTER TABLESPACE…RENAME DATAFILE… ALTER TABLESPACE…READ ONLY ALTER TABLESPACE…READ WRITE ALTER TABLESPACE…OFFLINE DROP TABLESPACE
Package procedures and functions	None

Practice 8 Overview

This practice covers the following topics:

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

ORACLE

Practice 8: Managing Tablespaces and Data Files

1. Create permanent tablespaces with the following names and storage:
   a. DATA01 data dictionary managed.
   b. DATA02 locally managed with uniform sized extents (Ensure that every used extent size in the tablespace is a multiple of 100 KB.)
   c. INDX01 locally managed with uniform sized extents of 4K
   (Enable automatic extension of 500 KB when more extents are required with a maximum size of 2 MB.)
   d. RONLY for read-only tables with the default storage. DO NOT make the tablespace read only at this time.

   Display the information from the data dictionary.

Tablespace Name	Subdirectory	Datfile Name (Size)
DATA01	u04	data01.dbf (2 MB)
DATA02	u03	data02.dbf (1 MB)
INDX01	u02	indx01.dbf (1 MB)
RONLY	u01	ronly.dbf (1 MB)

2. Allocate 500K more disk space to tablespace DATA02. Verify the result.
3. Relocate tablespace INDX01 to subdirectory u06.
4. Create a table in tablespace RONLY. Make tablespace RONLY read-only. Attempt to create an additional table. Drop the first created table. What happens and why?
5. Drop tablespace RONLY and the associated datafile. Verify it.
6. Set DB_CREATE_FILE_DEST to $HOME/ORADATA/u05 in memory only. Create tablespace DATA03 size 5M. Do not specify a file location. Verify the creation of the data file.

9

Storage Structure and Relationships

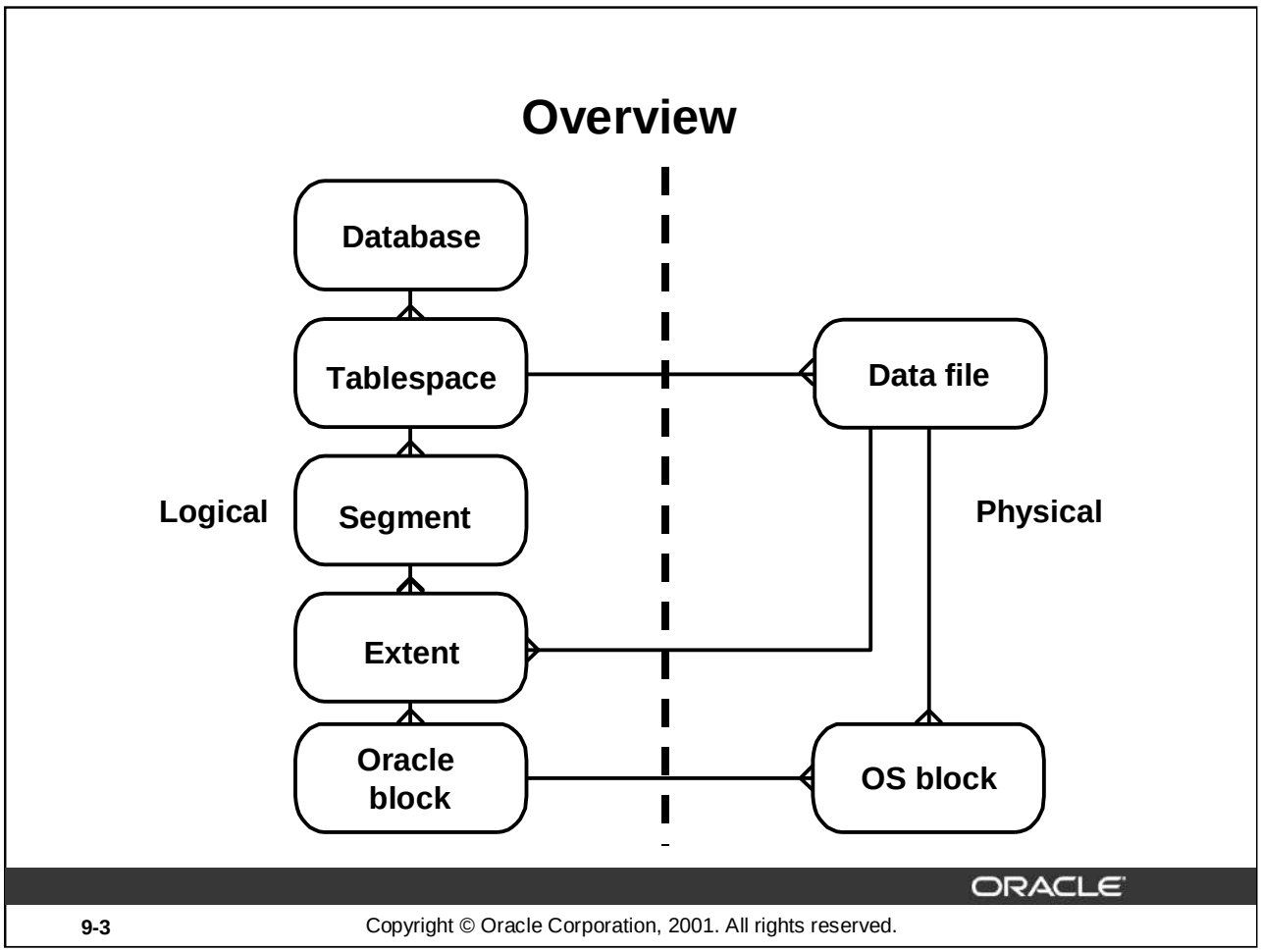
ORACLE

Objectives

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

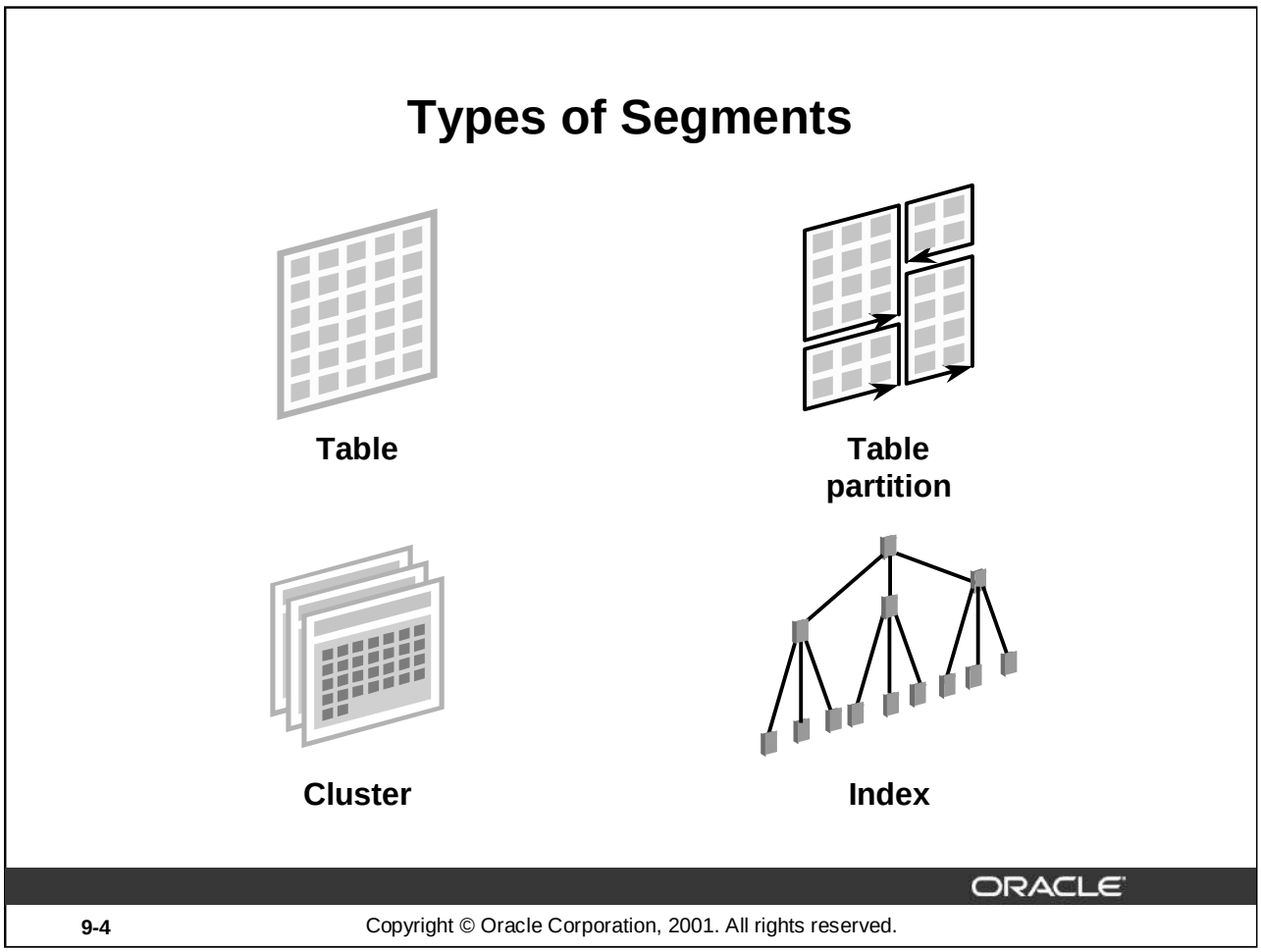
- **Describe the logical structure of the database**
- **List the segment types and their uses**
- **List the keywords that control block space usage**
- **Obtain information about storage structures from the data dictionary**
- **List the criteria for separating segments**

ORACLE



Database Architecture

The previous lesson discussed the storage structure of a database, its tablespaces, and its data files. This lesson continues the discussion of database storage by examining segments, extents, and data blocks.



Types of Segments

Segments are space-occupying objects in a database. They use space in the data files of a database. This section describes the different types of segments.

Table

A table is the most common means of storing data within a database. A table segment stores that data for a table that is neither clustered nor partitioned. Data within a table segment is stored in no particular order, and the database administrator has very little control over the location of rows within the blocks in a table. All the data in a table segment must be stored in one tablespace.

Table Partition

Scalability and availability are major concerns when there is a table in a database with high concurrent usage. In such cases, data within a table may be stored in several partitions, each of which resides in a different tablespace. The Oracle server currently supports partitioning by a range of key values or by a hashing algorithm. If a table is partitioned, each partition is a segment, and storage parameters can be specified to control them independently. Use of this type of segment requires the Partitioning option within the Oracle9*i* Enterprise Edition.

Types of Segments (continued)

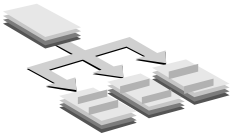
Cluster

A cluster, like a table, is a type of data segment. Rows in a cluster are stored based on key column values. A cluster may contain one or more tables. Tables in a cluster belong to the same segment and share the same storage characteristics. The rows in a clustered table can be accessed with an index or hashing algorithm.

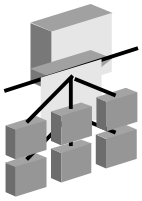
Index

All the entries for a particular index are stored within one index segment. If a table has three indexes, three index segments are used. The purpose of this segment is to look up the location of rows in a table based on a specified key.

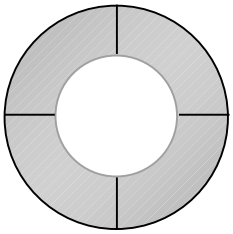
Types of Segments



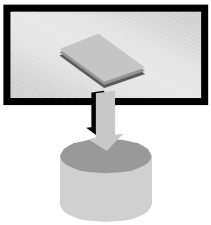
Index-organized table



Index partition



Undo segment



Temporary segment

ORACLE

Types of Segments

Index-Organized Table

In an index-organized table, data is stored within the index based on the key value. An index-organized table does not need a table lookup, because all the data can be retrieved directly from the index tree.

Index Partition

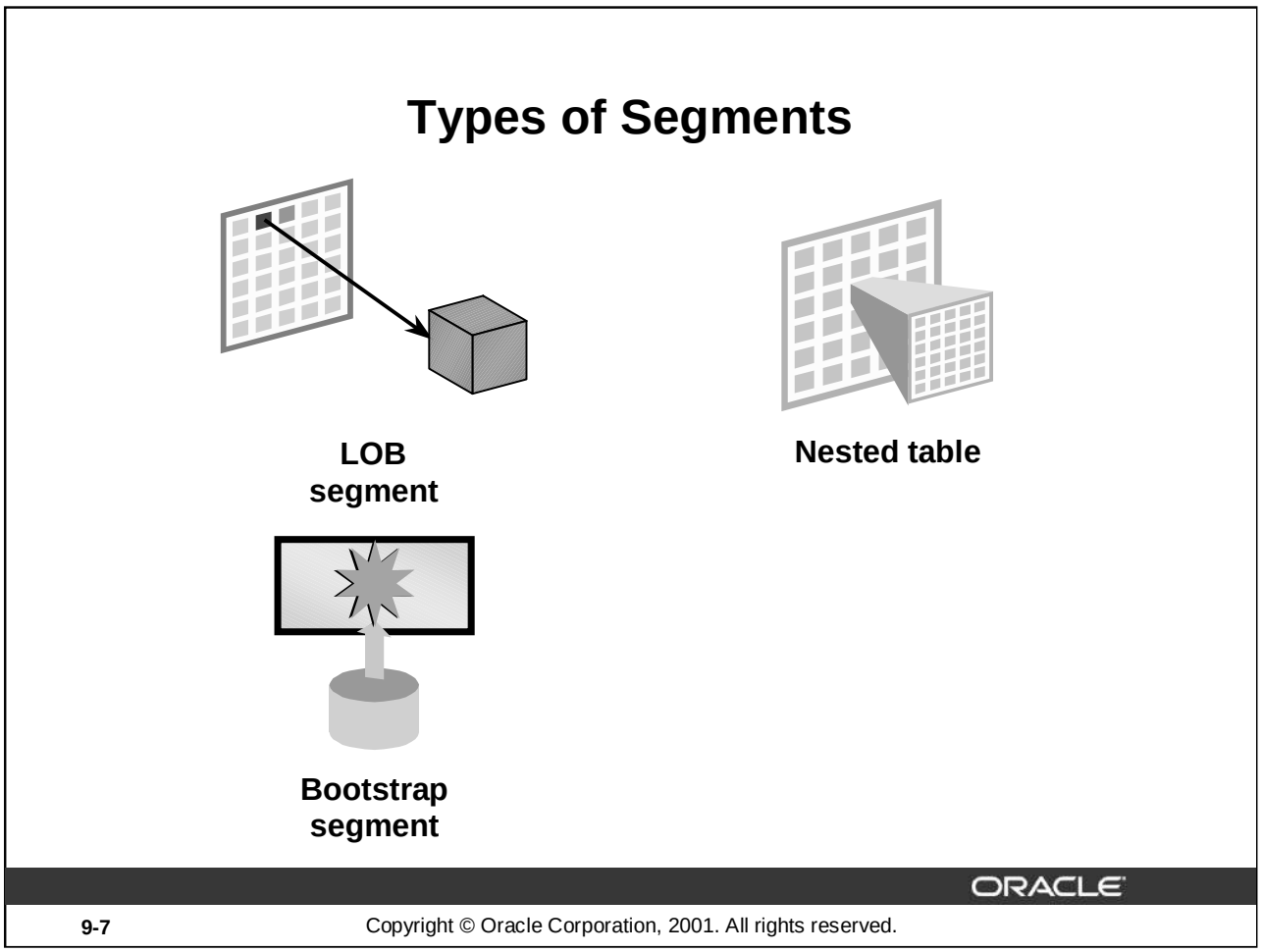
An index can be partitioned and spread across several tablespaces. In this case, each partition in the index corresponds to a segment and cannot span multiple tablespaces. The primary use of a partitioned index is to minimize contention by spreading index I/O. Use of this type of segment requires the Partitioning option within the Oracle8*i* Enterprise Edition.

Undo Segment

An undo segment is used by a transaction that is making changes to a database. Before changing the data or index blocks, the old value is stored in the undo segment. This allows a user to undo changes made.

Temporary Segment

When a user executes commands such as `CREATE INDEX`, `SELECT DISTINCT`, and `SELECT GROUP BY`, the Oracle server tries to perform sorts in memory. When a sort needs more space than the space available in memory, intermediate results are written to the disk. Temporary segments are used to store these intermediate results.



Types of Segments

LOB Segment

One or more columns in a table can be used to store large objects (LOBs) such as text documents, images, or videos. If the column is large, the Oracle server stores these values in separate segments known as LOB segments. The table contains only a locator or a pointer to the location of the corresponding LOB data.

Nested Table

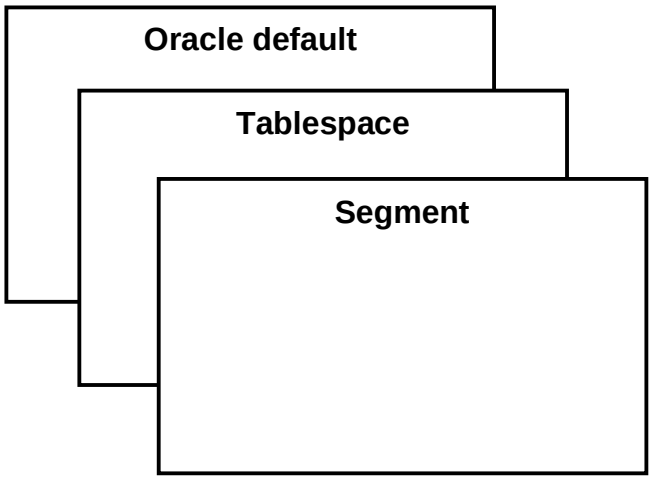
A column in a table may be made up of a user-defined table as in the case of items within an order. In such cases, the inner table, which is known as a nested table, is stored as a separate segment.

Bootstrap Segment

A bootstrap segment, also known as a cache segment, is created by the `sql.bsq` script when a database is created. This segment helps to initialize the data dictionary cache when the database is opened by an instance.

The bootstrap segment cannot be queried or updated and does not require any maintenance by the database administrator.

Storage Clause Precedence



ORACLE

Storage Parameters

A storage clause can be specified at the segment level to control how extents are allocated to a segment.

- Any storage parameter specified at the segment level overrides the corresponding option set at the tablespace level, except for the `MINIMUM EXTENT` or `UNIFORM SIZE` tablespace parameter.
- When storage parameters are not set explicitly at the segment level, they default to those at the tablespace level.
- When storage parameters are not set explicitly at the tablespace level, the Oracle server system defaults are used.

Other Considerations

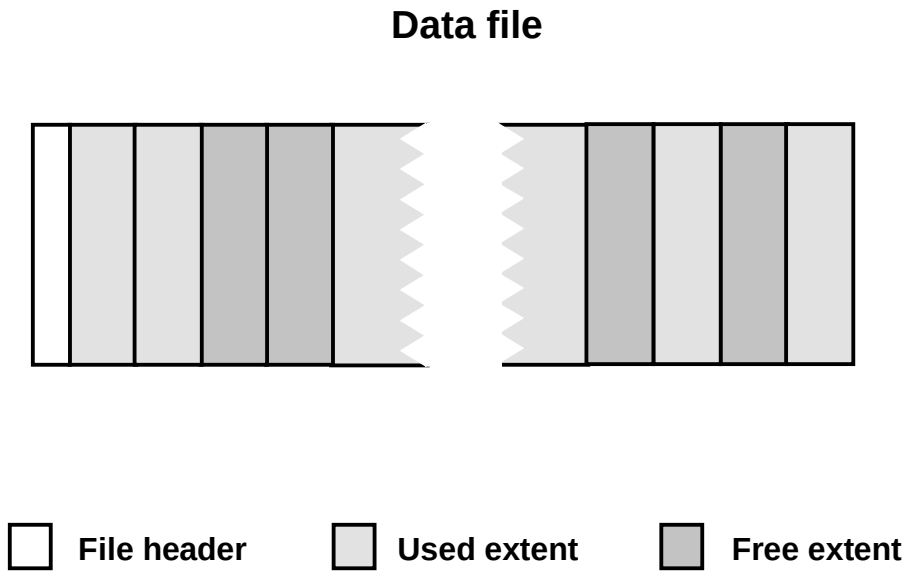
- If storage parameters are altered, the new options apply only to the extents not yet allocated.
- Some parameters cannot be specified at the tablespace level. These parameters need to be specified at the segment level only.
- If minimum extent size has been specified for the tablespace, this size applies to all extents that are allocated for segments in the tablespace in the future.

Extent Allocation and Deallocation

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

ORACLE

Used and Free Extents



ORACLE

Extents

When a tablespace is created, the data files in the tablespace contain a header, which is the first block or blocks in the file.

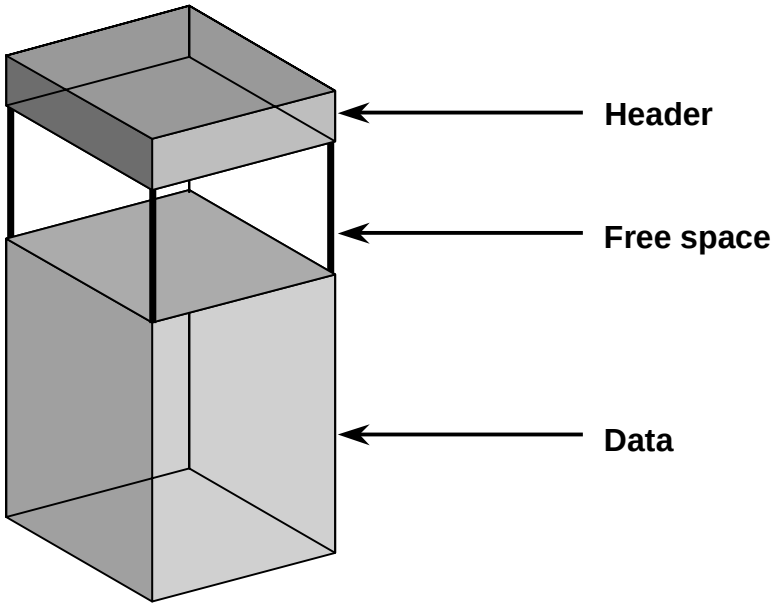
As segments are created, they are allocated space from the free extents in a tablespace. Contiguous space used by a segment is referred to as a used extent. When segments release space, the extents that are released are added to the pool of free extents available in the tablespace.

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

ORACLE

Database Block Contents



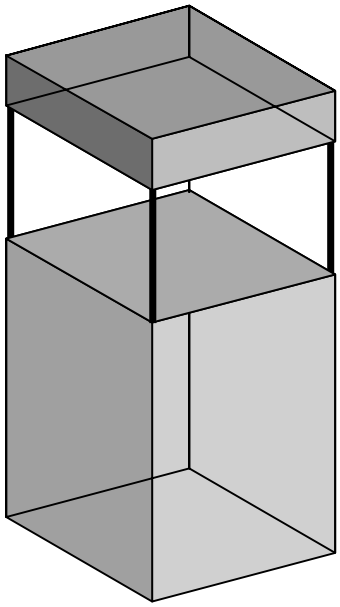
ORACLE

Data Blocks

Oracle data blocks contain:

- Block header: The header contains the data block address, table directory, row directory, and transaction slots that are used when transactions make changes to rows in the block. Block headers grow from the top down.
- Data space: Row data is inserted into the block from the bottom up.
- Free space: The free space in a block is in the middle of the block: thus both the header and the row data space can grow when necessary. The free space in a block is contiguous initially. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the Oracle server when necessary.

Block Space Utilization Parameters



INITRANS

MAXTRANS

PCTFREE

PCTUSED

ORACLE

Block Space Utilization Parameters

Block space utilization parameters can be used to control the use of space in data and index segments.

Parameters Controlling Concurrency

INITRANS and MAXTRANS specify the initial and the maximum number of transaction slots that are created in an index or a data block. The transaction slots are used to store information about transactions that are making changes to the block at a point in time. A transaction uses only one transaction slot, even if it is changing more than one row or index entry.

INITRANS, which defaults to 1 for a data segment and 2 for an index segment, guarantees a minimum level of concurrency. For example, if set to 3, INITRANS ensures that at least three transactions can concurrently make changes to the block. If necessary, additional transaction slots can be allocated from the free space in the block to permit more concurrent transactions to modify rows in the block.

MAXTRANS, which has a default value of 255, sets the limit for the number of concurrent transactions that can make changes to a data or an index block. When set, this value restricts use of space for transaction slots and therefore guarantees that there is sufficient space in the block for use by row or index data.

Block Space Utilization Parameters (continued)

Parameters Controlling the Use of Data Space

PCTFREE for a data segment specifies the percentage of space in each data block reserved for growth resulting from updates to rows in the block. The default for PCTFREE is 10%.

PCTUSED for a data segment represents the minimum percentage of used space that the Oracle server tries to maintain for each data block of the table. A block is put back on the free list when its used space falls below PCTUSED. The free list of a segment is a list of blocks that are candidates for accommodating future inserts. A segment, by default, is created with one free list. Segments can be created with a higher number of free lists by setting the FREELISTS parameter of the storage clause. The default for PCTUSED is 40%.

Both PCTFREE and PCTUSED are calculated as percentages of available data space, that is, the block space that remains after deducting the header space from the total block size.

Note: The use of these parameters for indexes is discussed in detail in the lesson "Managing Indexes."

Specifying FREELISTS is discussed in detail in the course *Oracle9i: Performance Tuning.*

Data Block Management

Two methods are available for managing data blocks:

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

ORACLE

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.

ORACLE

Automatic Segment-Space Management

Ease of Use

`PCTUSED`, `FREELISTS`, `FREELIST GROUPS` are managed automatically.

Better Space Utilization

All objects and especially objects with greatly varying row sizes utilize space more efficiently.

Better Concurrency Handling

Run-time adjustments to variations in concurrent access are improved.

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.

ORACLE

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.**

ORACLE

Configuring Automatic Segment-Space Management

Bitmapped segments are specified through the SEGMENT SPACE MANGEMENT AUTO clause of the CREATE TABLESPACE command, which cannot be subsequently altered. Any specifications of PCTUSED, FREELIST, and FREELIST GROUPS are ignored if defined.

Segments that can be bitmap-managed are heap tables, indexes, IOTs, and LOBs.

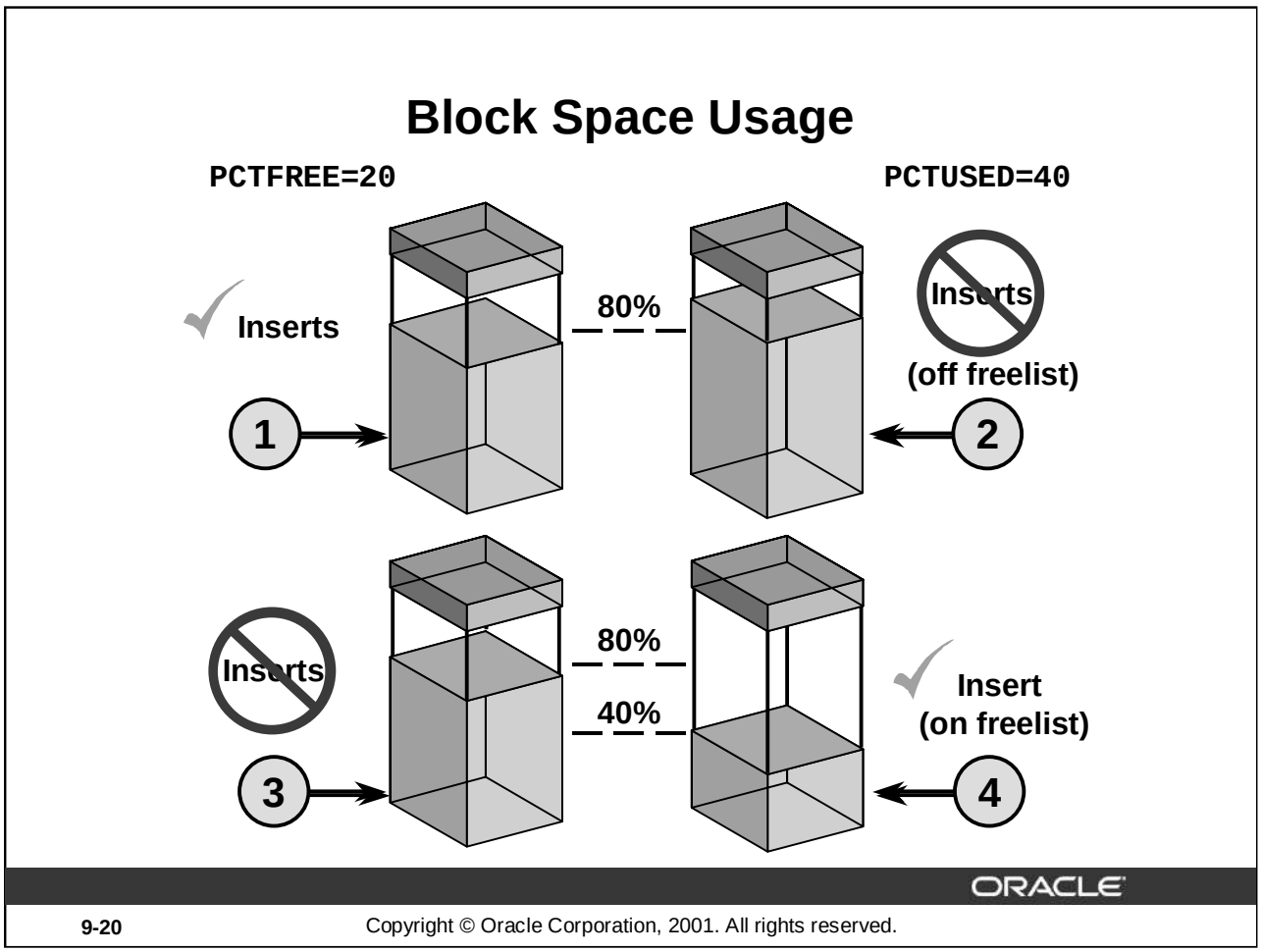
Manual Data Block Management

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

ORACLE

Manual Data Block Management

With manual database block management you can configure how block space is used and when a block is available. Parameters such as PCTFREE, PCTUSED, and FREELIST are used in manual management. Previously this was the only method available for managing data blocks.

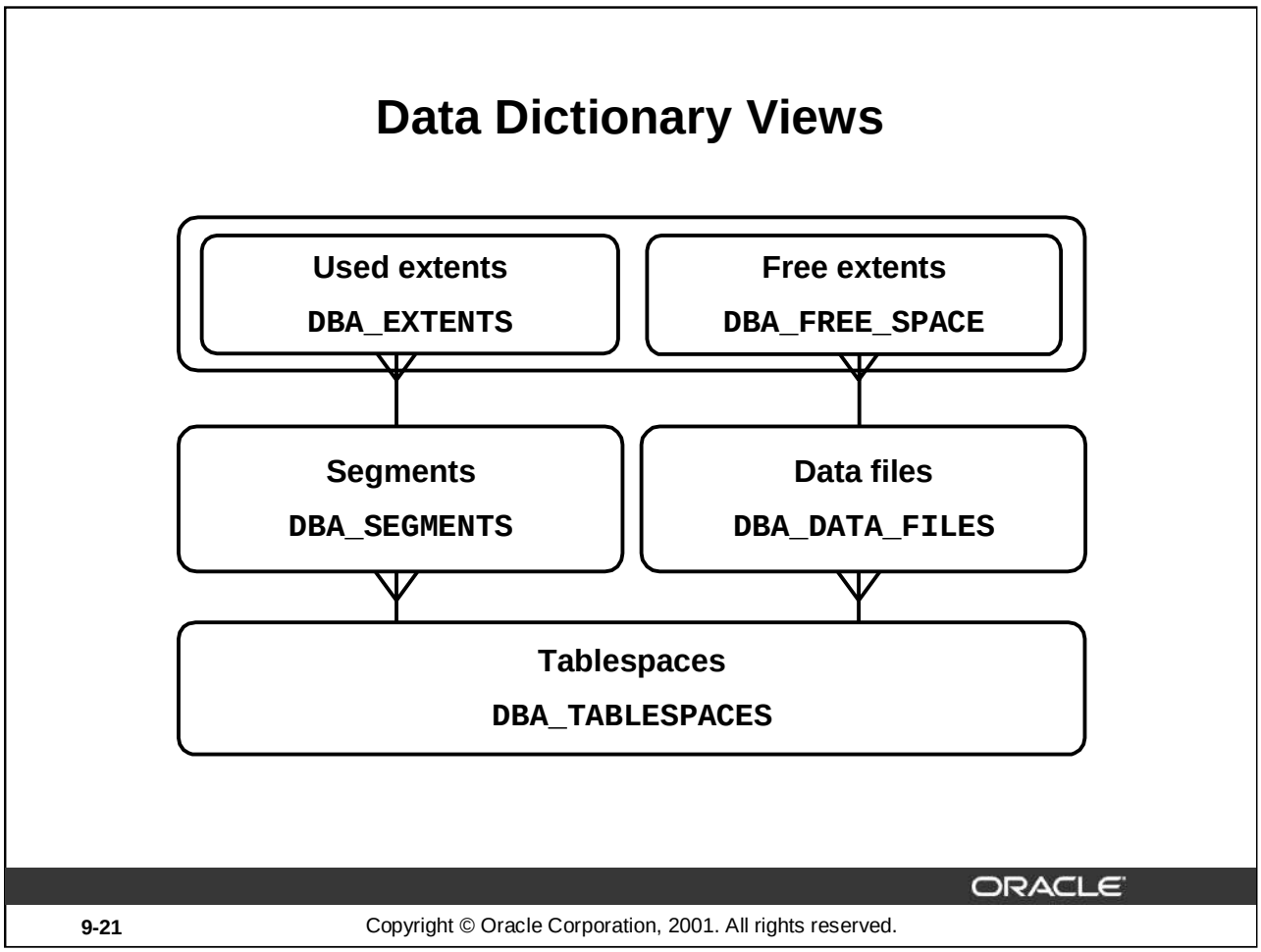


Block Space Usage

The following steps explain how space within a block is managed for a data segment with PCTFREE=20 and PCTUSED=40:

1. Rows are inserted into the block until the free space in the block is equal to or less than 20%. The block is no longer available for inserts when rows occupy 80% (100 - PCTFREE) or more of the available data space in the block.
2. The remaining 20% can be used when the size of a row increases. For example, a column that was originally NULL is updated to be assigned a value. Thus block utilization may be in excess of 80% as a result of updates.
3. If rows are deleted in the block or if rows decrease in size as a result of updates, block utilization may fall below 80%. However, a block is not used for inserts until the utilization falls below PCTUSED, which in this example, is 40%.
4. When the utilization falls below PCTUSED, the block is available for inserts. As rows are inserted into the block, the utilization of the block increases and the cycle repeats starting with step 1.

Note: Guidelines for setting PCTFREE and PCTUSED are discussed in the lessons on tables and indexes, “Managing Tables” and “Managing Indexes” respectively.



Querying the Data Dictionary

The relationships between tablespaces, data files, segments, and free and used extents can be viewed by querying the data dictionary.

When a tablespace with one or more files is created, a row is added to DBA_TABLESPACES. For each file in the database, a row is added to DBA_DATA_FILES. At this stage, the space in each data file, excluding the file header, shows up as one free extent in DBA_FREE_SPACE.

When a segment is created, a row is visible in DBA_SEGMENTS. The space allocated to the extents in this segment can be viewed from DBA_EXTENTS, while DBA_FREE_SPACE is adjusted to show lower free space in the files where the extents have been created for the segment.

All the space in a file (excluding the header block) must be accounted for either in DBA_FREE_SPACE or in DBA_EXTENTS.

Obtaining Storage Information

- **Data Dictionary Views**
  - **DBA_TABLESPACES**
  - **DBA_DATA_FILES**
  - **DBA_SEGMENTS**
  - **DBA_EXTENTS**
  - **DBA_FREE_SPACE**

ORACLE

DBA_SEGMENTS

Query the DBA_SEGMENTS view to get the number of extents and blocks allocated to a segment.

```
SQL> SELECT segment_name,tablespace_name,extents,blocks
  2  FROM   dba_segments
  3  WHERE  owner = 'HR';
SEGMENT_NAME    TABLESPACE    EXTENTS     BLOCKS
--------------- ---------- ---------- ----------
REGIONS         SAMPLE              1          8
LOCATIONS       SAMPLE              1          8
DEPARTMENTS     SAMPLE              1          8
JOBS            SAMPLE              1          8
EMPLOYEES       SAMPLE              1          8
JOB_HISTORY     SAMPLE              1          8
5 rows selected.
```

DBA_EXTENTS

Use the DBA_EXTENTS view to check the extents for a given segment.

```
SQL> SELECT extent_id,file_id,block_id,blocks
  2  FROM dba_extents
  3  WHERE owner='HR'
  4  AND segment_name='EMPLOYEES';
EXTENT_ID FILE_ID BLOCK_ID BLOCKS
--------- ------- -------- ------
        0       4        2      5
        1       4       27      5
        2       4       32     10
        3       4       42     15
        4       4       57     20
5 rows selected.
```

DBA_FREE_SPACE

Use the DBA_FREE_SPACE view to check the extents for a given segment.

```
SQLPLUS> SELECT tablespace_name, count(*),
      2>   max(blocks), sum(blocks)
      3>   FROM dba_free_space
      4>   GROUP BY tablespace_name;
TABLESPACE_NAMECOUNT(*)    MAX(BLOCKS      SUM(BLOCKS
-------------------------------------------------------
DATA01           2         1284                 1533
RBS              3         2329                 2419
SORT             1         1023                 1023
SYSTEM           1         5626                 5626
TEMP             1         2431                 2431
5 rows selected.
```

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**

ORACLE

Quick Reference

Context	Reference
Initialization Parameters	DB_BLOCK_SIZE
Data Dictionary Views	DBA_TABLESPACES DBA_DATA_FILES DBA_SEGMENTS DBA_EXTENTS DBA_FREE_SPACE DBA_FREE_SPACE_COALESCED
Commands	CREATE TABLESPACE…SEGMENT SPACE MANAGEMENT AUTO

Practice 9 Overview

This practice covers the following topics:

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

ORACLE

Practice 9: Storage Structure and Relationships

1. As user SYSTEM, run the `lab09_01.sql` script to create tables and indexes.
2. Identify the different types of segments in the database.
3. Write a query to check which segments are within five extents short of the maximum extents. Ignore the bootstrap segment. This query is useful in identifying any segments that are likely to generate errors during future data load.
4. Which files have space allocated for the EMP table?
5. Run the `lab09_05.sql` script.
6. List the free space available by tablespace. The query should display the number of fragments, the total free space, and the largest free extent in each tablespace.
7. List segments that will generate errors because of lack of space when they try to allocate an additional extent.

10

Managing Undo Data

ORACLE

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**

ORACLE

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***

ORACLE

Managing Undo Data Overview

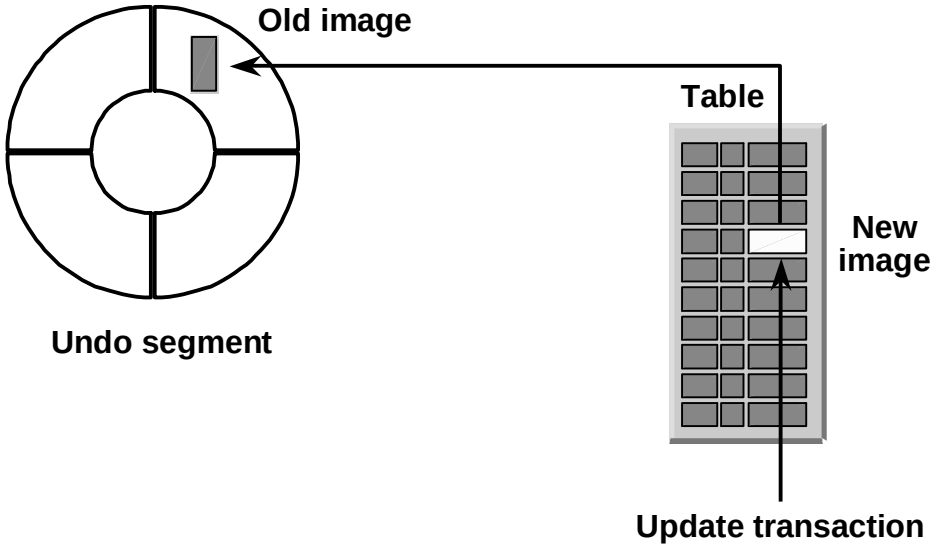
Automatic Undo Management

The Oracle server automatically manages the creation, allocation, and tuning of undo segments.

Manual Undo Management

You manually manage the creation, allocation, and tuning of undo segments. The only method available prior to Oracle9*i*. Information on manual undo management can be found in Appendix *Manually Managing Undo Data*.

Undo Segment



ORACLE

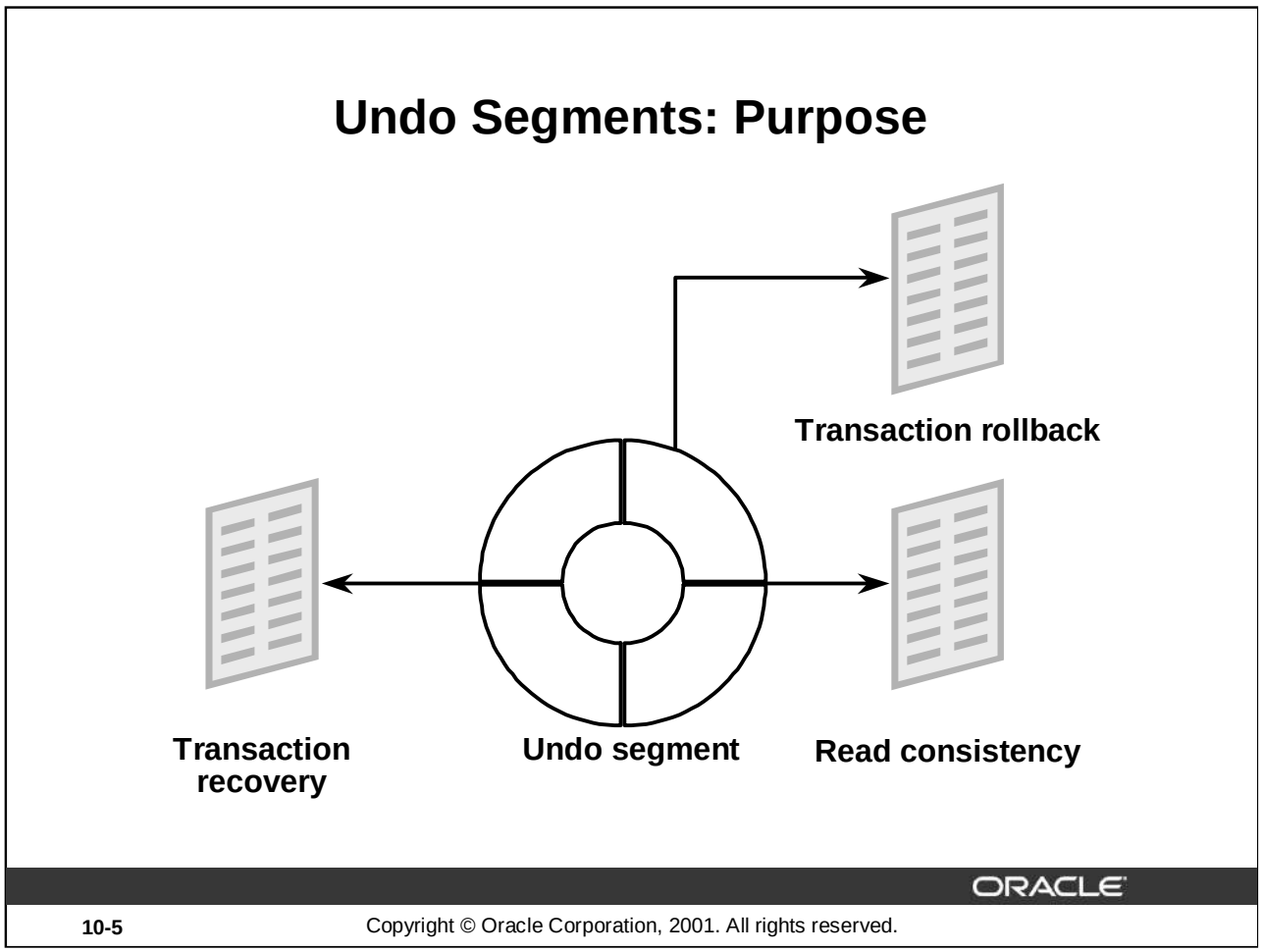
Undo Segment

An undo segment is used to save the old value (undo data) when a process changes data in a database. It stores the location of the data and the data as it existed before being modified.

The header of an undo segment contains a transaction table where information about the current transactions using the undo segment is stored.

A serial transaction uses only one undo segment to store all of its undo data.

Many concurrent transactions can write to one undo segment.



Undo Segment

Transaction Rollback

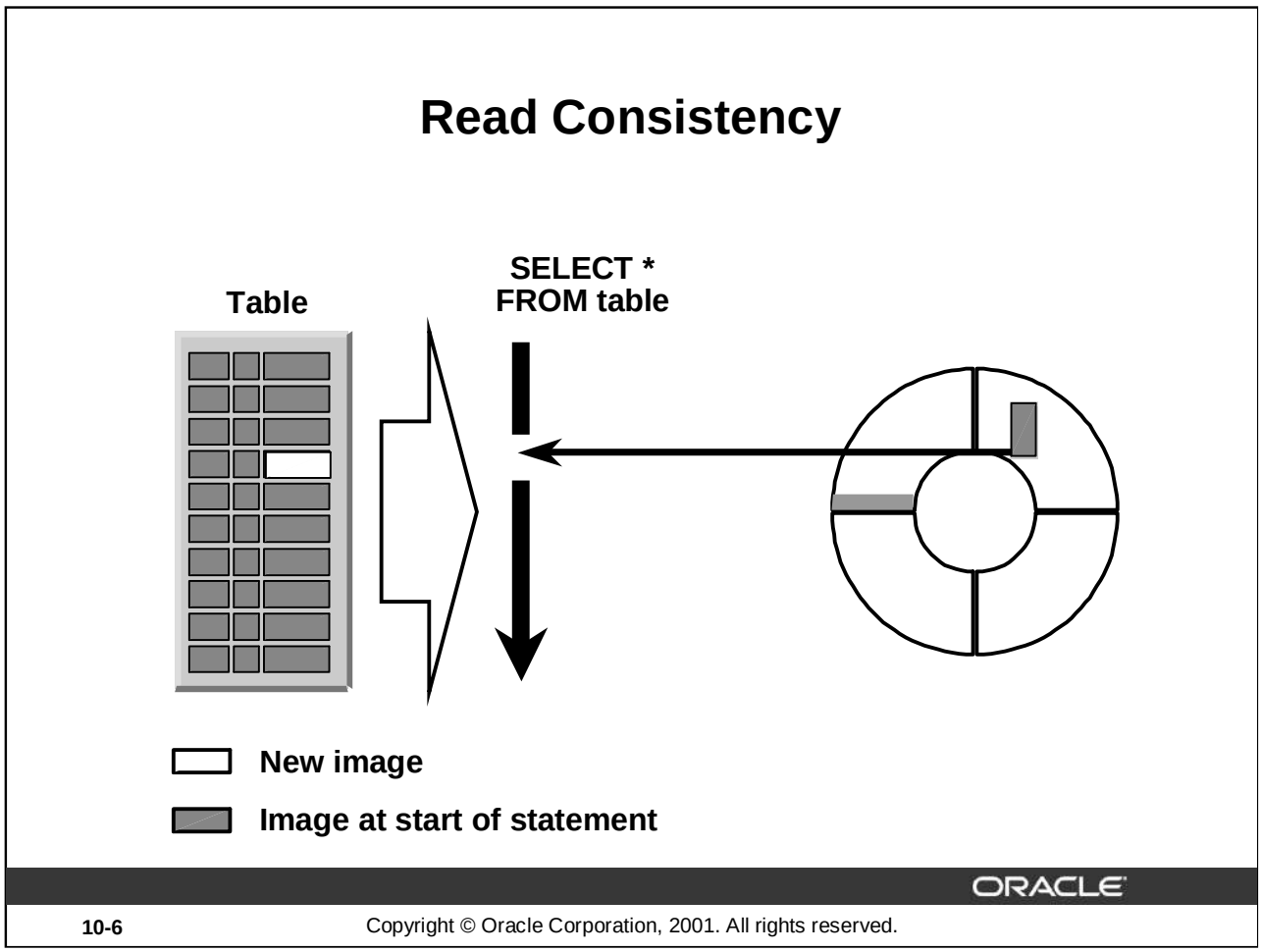
When a transaction modifies a row in a table, the old image of the modified columns (undo data) is saved in the undo segment. If the transaction is rolled back, the Oracle server restores the original values by writing the values in the undo segment back to the row.

Transaction Recovery

If the instance fails while transactions are in progress, the Oracle server needs to undo any uncommitted changes when the database is opened again. This rollback is part of transaction recovery. Recovery is possible only because changes made to the undo segment are also protected by the redo log files.

Read Consistency

While transactions are in progress, other users in the database should not see any uncommitted changes made by these transactions. In addition, a statement should not see any changes that were committed after the statement begins execution. The old values (undo data) in the undo segments are also used to provide the readers a consistent image for a given statement.



Read Consistency

The Oracle server guarantees that a statement sees data from a consistent time, even if that data is modified by other transactions.

When the Oracle server begins executing a SELECT statement, it determines the current system change number (SCN) and ensures that any changes not committed before this SCN are not processed by the statement. Consider the case where a long-running query is executed at a time when several changes are being made. If a row has changes that were not committed at the start of the query, the Oracle server constructs a read-consistent image of the row by retrieving the before image of the changes from the undo segment and applying the changes to a copy of the row in memory.

Transaction Read Consistency

Read consistency is always provided for a SQL statement. However, you can request read consistency for a read-only transaction by issuing the following command at the beginning of the transaction:

```
SET TRANSACTION READ ONLY;
```

Or, you can request read consistency for a transaction performing DML by issuing the following command at the beginning of the transaction:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

In either case, the Oracle server provides data that is read consistent from the start of the transaction. Using SERIALIZABLE can have a negative impact on performance.

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**

ORACLE

SYSTEM Undo Segment

The SYSTEM undo segment is created in the SYSTEM tablespace when a database is created. This undo segment can be used only for changes made to objects in the SYSTEM tablespace. The SYSTEM undo segment exists and works the same in both manual and auto mode.

Non-SYSTEM Undo Segments

A database that has multiple tablespaces needs at least one non-SYSTEM undo segment for manual mode or one UNDO tablespace for auto mode.

Manual Mode

In manual mode, a non-SYSTEM undo segment, which is created by the database administrator, can be used for changes made to objects in any non-SYSTEM tablespace. There are two types of non-SYSTEM undo segments.

Private

Private undo segments are segments that are brought online by an instance because they are listed in the parameter file. However, they can be brought online explicitly by issuing an ALTER ROLLBACK SEGMENT command.

SYSTEM Undo Segment (continued)

Public

Public undo segments form a pool of undo segments available in a database. Public undo segments are normally used with the Oracle Parallel Server to create a pool of undo segments that can be used by any of the Parallel Server instances.

Note: The use of public undo segments is discussed in the *Oracle9i Parallel Server Concepts and Administration* manual.

Deferred Undo Segments

Deferred undo segments may be created when a tablespace is brought offline. They are used to roll back transactions when the tablespace is brought back online. They are dropped automatically when they are no longer needed.

Because deferred undo segments are maintained by the Oracle server, no maintenance is required on your part

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.**

ORACLE

Automatic Undo Management: Concepts

Undo segments are created with the naming convention:

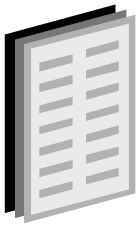
```
_SYSSMUn$
```

For example:

```
_SYSSMU1$
_SYSSMU2$
```

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

ORACLE

AUTOMATIC UNDO SEGMENT: CONFIGURATION

If only one UNDO tablespace exists in the database and UNDO_MANAGEMENT is set to AUTO, then UNDO_TABLESPACE parameter is optional; the Oracle Server will automatically choose the 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
```

ORACLE

Automatic Undo Management: Initialization Parameters

UNDO_MANAGEMENT

The UNDO_MANAGEMENT parameter determines the undo mode of the database. The parameter can be set to one of two values, AUTO or MANUAL, and must be set in the initialization file. UNDO_MANAGEMENT cannot be changed dynamically after the database starts. AUTO mode sets the database to automatic undo management and requires an UNDO tablespace. In MANUAL mode, the default value, you can create and manage undo segments when needed within the database as in previous versions of the Oracle server.

UNDO_TABLESPACE

Specifies the UNDO tablespace to be used. This parameter can be set in the initialization files or altered dynamically using the ALTER SYSTEM command.

```
ALTER SYSTEM SET 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;
```

ORACLE

Automatic Undo Management Configuration

Automatic undo management requires an UNDO tablespace. More than one UNDO tablespace may exist in the database, but only one UNDO tablespace can be active.

You can create the UNDO tablespace with the database by adding a clause to the CREATE DATABASE statement.

During database creation, if the UNDO_MANAGEMENT parameter is set to AUTO and you omit the UNDO tablespace clause from the CREATE DATABASE statement then the Oracle server creates an UNDO tablespace with the name SYS_UNDOTBS. The default data file, for data file tablespace SYS_UNDOTS, will have the name 'dbu1<*oracle_sid*>.dbf' located in $ORACLE_HOME/dbs, the size is operating system dependent, and AUTOEXTEND is ON.

After database creation, you can create an UNDO tablespace using the CREATE UNDO TABLESPACE command.

Automatic Undo Management Configuration (continued)

Using Oracle Enterprise Manager to Create a New UNDO Tablespace

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Select Create from the right mouse menu of the Tablespace folder
6. In the General tab of the property sheet, enter the file name and file size.
7. In the Type region, select Undo.
8. Click Create.

Note: You can also launch the Console from Windows NT Start menu

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;
```

ORACLE

Altering an UNDO Tablespace

The server provides support for the following clauses when altering an UNDO tablespace.

- ADD DATAFILE
- RENAME
- DATAFILE [ONLINE|OFFLINE]
- BEGIN BACKUP
- END BACKUP

Altering an UNDO Tablespace (continued)

Using Oracle Enterprise Manager to Alter an UNDO Tablespace

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Select the Tablespace folder and right-click the UNDO tablespace.
6. Select “Add a data file.”

Note: You can also launch the Console from Windows NT Start menu

Automatic Undo Management: Switching UNDO Tablespaces

- You may switch from using one UNDO tablespace to another
- Only one UNDO tablespace can be in 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;
```

ORACLE

Using Oracle Enterprise Manager to Switch an UNDO Tablespace

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand the Instance folder
5. Click on Configuration.
6. Click Undo tab from the detail side of the Console
7. Select the UNDO tablespace from the Current Undo tablespace drop down list
8. Click Apply.

Note: You can also launch the Console from Windows NT Start menu

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

ORACLE

Dropping an UNDO Tablespace

When dropping an UNDO tablespace, it cannot be in use by an instance and all transactions within the tablespace must be complete.

If tablespace UNDOTBS is the current active UNDO tablespace for the database, and it is to be dropped, then a new UNDO tablespace must be set before dropping it. First create another UNDO tablespace if one does not already exist. Then use the ALTER SYSTEM command to change the current UNDO tablespace.

```
ALTER SYSTEM SET undo_tablespace = UNDOTBS2;
```

You can drop tablespace UNDOTBS after all transactions within the tablespace are complete. To determine whether any active transactions exists use the following query:

```
SQL> SELECT a.name,b.status
  2> FROM   v$rollname a, v$rollstat b
  3> WHERE  a.name IN ( SELECT segment_name
  4>                    FROM   dba_segments
  5>                    WHERE  tablespace_name = 'UNDOTBS'
  6>                  )
  7> AND a.usn = b.usn;

NAME                           STATUS
------------------------------ ---------------
_SYSSMU4$                      PENDING OFFLINE
```

Dropping an UNDO Tablespace (continued)

An undo segment with the status `PENDING OFFLINE` still contains active transactions. When no rows return from the query, then all transactions are complete, and the tablespace can be dropped.

```
DROP TABLESPACE UNDOTBS;
```

The Oracle server may reference tablespace `UNDOTBS` after switching to another `UNDO` tablespace to provide a consistent read for queries. Queries requiring information from tablespace `UNDO1`, after it is no longer available to provide a consistent read, receive the error: `ORA-1555 snapshot too old`.

Dropping an UNDO Tablespace (continued)

Using Oracle Enterprise Manager to Drop a UNDO Tablespace

1. Launch the Console:
   % oemapp console
2. Choose Launch Standalone
3. Expand your working database from the Databases folder
4. Expand Storage folder
5. Select the Tablespace folder.
6. Right click the name of the UNDO tablespace and select Remove from the menu.

Note: You can also launch the Console from Windows NT Start menu

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

ORACLE

Automatic Undo Management: Other Parameters

UNDO_SUPPRESS_ERRORS

UNDO_SUPPRESS_ERRORS enables users to suppress errors while executing manual undo management mode operations (for example, ALTER ROLLBACK SEGMENT ONLINE) in automatic undo management mode. Setting this parameter enables users to use the undo tablespace feature before all application programs and scripts are converted to automatic undo management mode. For example, if you have a tool that uses SET TRANSACTION USE ROLLBACK SEGMENT statement, you can add the statement "ALTER SESSION SET UNDO_SUPPRESS_ERRORS = true" to the tool to suppress the ORA-30019 error.

```
ORA-30019: Illegal rollback Segment operation in Automatic
           Undo mode
```

UNDO_RETENTION

Determines how long to retain undo data to provide for consistent reads. Retaining undo data longer allows for longer queries and also requires larger data files for the UNDO tablespace. The UNDO_RETENTION parameter, defined in seconds, can be set in the initialization file or modified dynamically with an ALTER SYSTEM command.

```
ALTER SYSTEM SET UNDO_RETENTION=900;
```

A value of 900 retains undo data for 15 minutes.

UNDO_SUPPRESS_ERRORS (continued)

Even with `UNDO_RETENTION` set, if the UNDO tablespace is sized too small, undo data is not retained for the time specified. The Oracle server uses an algorithm for allocating space within an UNDO tablespace and allocates unexpired space with no active transactions before causing a new transaction to fail.

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
```

ORACLE

V$UNDOSTAT

This view displays a histogram of statistical data to show how well the system is working. Each row in the view keeps statistics collected in the instance for a 10-minute interval.You can use this view to estimate the amount of undo space required for the current workload. The database uses this view to tune undo usage in the system. This view is available in both auto mode and manual mode.

Although the time interval is normally 10 minutes, the most recent row will return the time since its interval started, usually less than 10 minutes.

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)**

```
UndoSpace = [UR * (UPS * DBS)] + (DBS * 24)
```

ORACLE

Sizing an UNDO Tablespace

Sizing an UNDO tablespace requires three pieces of data. Two can be obtained from the initialization file: UNDO_RETENTION and DB_BLOCK_SIZE. The third piece of the formula requires a query against the database. The number of undo blocks generated per second can be acquired from V$UNDOSTAT. The following formula calculates the total number of blocks generated and divides it by the amount of time monitored, in seconds:

```
SELECT (SUM(undoblks) / SUM( ((end_time - begin_time) * 86400)
FROM v$undostat;
```

Column END_TIME and BEGIN_TIME are DATE data types. When DATE data types are subtracted, the result is in days. To convert days to seconds, you multiply by 86400, the number of seconds in a day.

The result of the query returns the number of undo blocks per second. This value needs to be multiplied by the size of an undo block, which is the same size as the database block defined in DB_BLOCK_SIZE. The following query calculates the number of bytes needed:

Sizing an UNDO Tablespace (continued)

```
SQL> SELECT (UR * (UPS * DBS)) + (DBS * 24) AS "Bytes"
  2>  FROM (SELECT value AS UR
  3>  FROM v$parameter
  4>  WHERE name = 'undo_retention'),
  5>  (SELECT (SUM(undoblks)/SUM(((end_time -
begin_time)*86400))) AS UPS
  6>  FROM v$undostat),
  7>  (SELECT value AS DBS
  8>  FROM v$parameter
  9>  WHERE name = 'db_block_size');

     Bytes
----------
  19106213
```

To convert bytes to megabytes, divide by 1,048,576 bytes. The result for this database is 18.22 megabytes.

For best results, the calculation should be made during the time of day when the database has its heaviest workload.

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**

ORACLE

Undo Quota

Using resource plans, users can be grouped and limits placed on the amount of resources that can be used by the group. The amount of undo data generated by a group can be limited by setting a value for UNDO_POOL: the default value is unlimited. When a group exceeds its limit an error is received and no new transactions can be performed for the group until current transactions complete or abort.

```
ORA-30027: "Undo quota violation - failed to get %s (bytes)"
    Cause: The amount of undo assigned to the consumer group
           of this session has been exceeded.
   Action: Ask DBA to increase undo quota, or wait until other
           transactions commit before proceeding.
```

Obtaining Undo Segments Information

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

ORACLE

Obtaining Undo Segment Information

To obtain information about all the undo segments in the database, query the DBA_ROLLBACK_SEGS view:

```
SQL> SELECT segment_name,owner,tablespace_name,status
  2  FROM   dba_rollback_segs;

SEGMENT_NA OWNER  TABLESPACE STATUS
---------- ------ ---------- ------
SYSTEM     SYS    SYSTEM     ONLINE
_SYSSMU1$  PUBLIC UNDO1      ONLINE
_SYSSMU2$  PUBLIC UNDO1      ONLINE
_SYSSMU3$  PUBLIC UNDO1      ONLINE
_SYSSMU4$  PUBLIC UNDO1      ONLINE
```

Information about undo segments that are offline can be seen only in DBA_ROLLBACK_SEGS. The dynamic performance views show only undo segments that are online.

The OWNER column specifies the type of an undo segment:

- SYS refers to a private undo segment.
- PUBLIC refers to a public undo segment.

Obtaining Undo Segment Information (continued)

V$ROLLSTAT and V$ROLLNAME

Join the V$ROLLSTAT and V$ROLLNAME views to obtain the statistics of the undo segments currently used by the instance.

Example

```
SQL> SELECT n.name, s.extents, s.rssize,s.hwmsize,
  2  s.xacts, s.status
  3  FROM v$rollname n, v$rollstat s
  4  WHERE n.usn = s.usn;

NAME        EXTENTS     RSSIZE    HWMSIZE      XACTS STATUS
---------- ------- ---------- ---------- ---------- ------
SYSTEM            7     425984     425984          0 ONLINE
_SYSSMU1$         5    2289664    2289664          0 ONLINE
_SYSSMU2$        10    6549504    6549504          0 ONLINE
_SYSSMU3$         7    4386816    4386816          0 ONLINE
_SYSSMU4$         6    4321280    4321280          0 ONLINE
```

V$TRANSACTION and V$SESSION

To check the use of a undo segment by currently active transactions, join the V$TRANSACTION and V$SESSION views:

Example

```
SQL> SELECT s.username, t.xidusn, t.ubafil,
  2         t.ubablk, t.used_ublk
  3  FROM   v$session s, v$transaction t
  4  WHERE  s.saddr = t.ses_addr;

USERNAME     XIDUSN     UBAFIL     UBABLK  USED_UBLK
-------- ---------- ---------- ---------- ----------
HR                2          2       5005          1
```

Summary

In this lesson, you should have learned how to:

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

ORACLE

Quick Reference

Context	Reference
Initialization Parameters	UNDO_MANAGEMENT UNDO_TABLESPACE UNDO_RETENTION UNDO_SUPPRESS_ERRORS
Dynamic Performance Views	V$UNDOSTAT V$ROLLNAME V$ROLLSTAT V$TRANSACTION V$SESSION
Data Dictionary Views	DBA_ROLLBACK_SEGS
Commands	CREATE DATABASE…UNDO TABLESPACE CREATE UNDO TABLESPACE

Practice 10 Overview

This practice covers the following topics:

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

ORACLE

Practice 10: Managing Undo Data

1. Connect as user SYS, and list the undo segments in tablespace UNDOTBS.
2. Create undo tablespace UNDO2, size 15M, in $HOME/oradata/u03. List the rollback segments in tablespace UNDO2.
3. In a new telnet session start SQL*Plus and connect as user HR and run script lab10_03.sql to insert a row into table DEPARTMENTS.
   Do not commit, roll back, or exit the session.
4. In the session in which you are connected as SYS, using the ALTER SYSTEM command, switch the UNDO tablespace from UNDOTBS to UNDO2 for the instance.
5. As SYS drop tablespace UNDOTBS. What happened? Why?
6. List the undo segments in tablespace UNDOTBS and their status. Compare this list to the list in step 1.
7. In the session connected as HR, roll back the transaction and exit the session.
8. In the session connected as SYS drop tablespace UNDOTBS. What happened? Why?
9. As SYS issue the following command:

   ```
   ALTER SYSTEM SET undo_retention=0 SCOPE=memory;
   ```

   Now drop tablespace UNDOTBS. What happened? Why?

   Note: There still may be a delay before the tablespace is drop.

11

Managing Tables

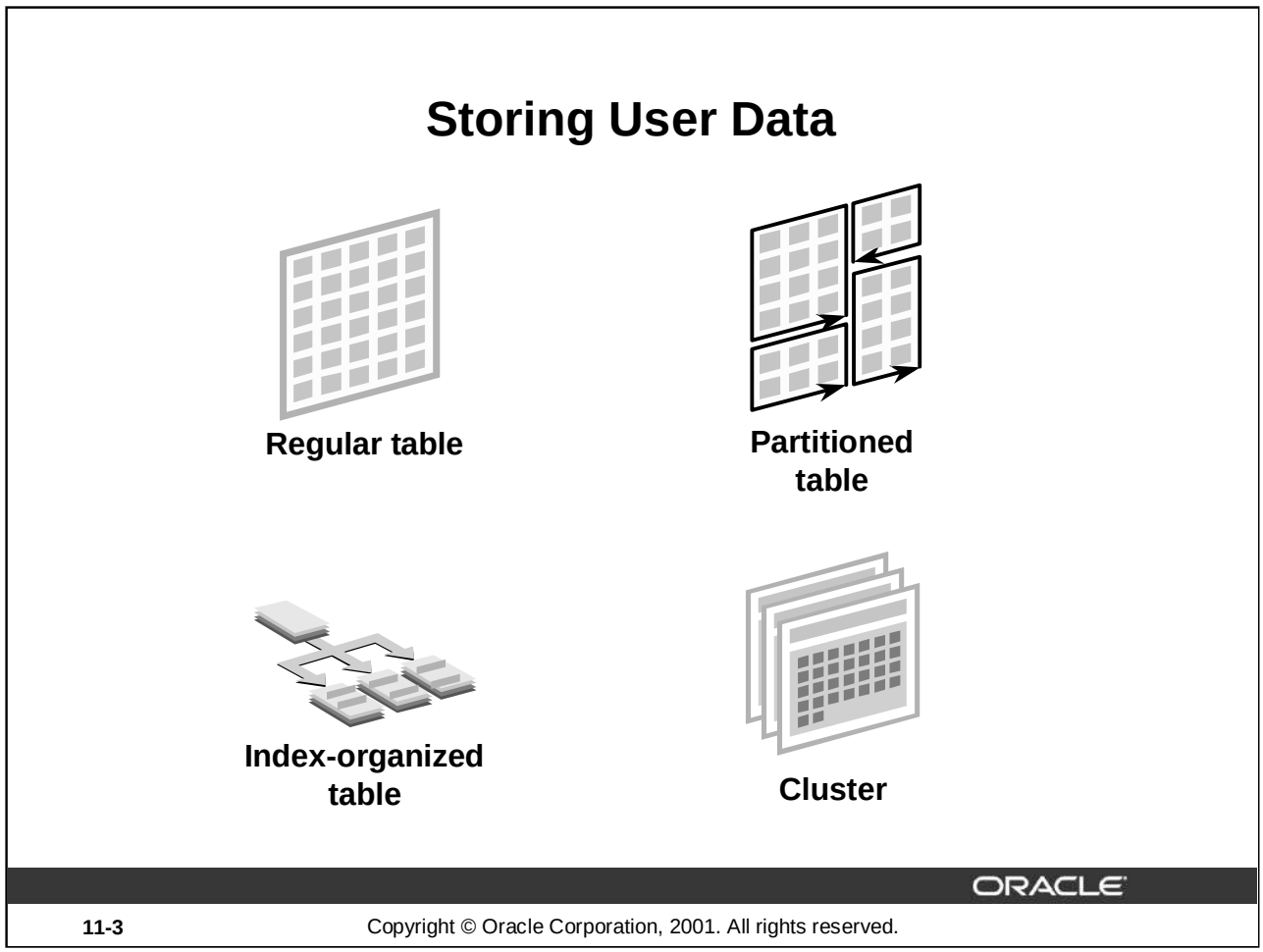
ORACLE

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**

ORACLE



Using Different Methods for Storing User Data

There are several methods for storing user data in an Oracle database:

- Regular tables
- Partitioned tables
- Index-organized tables
- Clustered tables

Note: Partitioned tables, index-organized tables, and clustered tables are covered in other courses.

Regular Table

A regular table (generally referred to as a “table”) is the most commonly used form of storing user data. This is the default table and is the main focus of discussion in this lesson. A database administrator has very limited control over the distribution of rows in an unclustered table. Rows can be stored in any order depending on the activity on the table.

Using Different Methods for Storing User Data (continued)

Partitioned Table

A partitioned table enables the building of scalable applications. It has the following characteristics:

- A partitioned table has one or more partitions, each of which stores rows that have been partitioned using range partitioning, hash partitioning, composite partitioning, or list partitioning.
- Each partition in a partitioned table is a segment and can be located in a different tablespace.
- Partitions are useful for large tables that can be queried or manipulated using several processes concurrently.
- Special commands are available to manage partitions within a table.

Index-Organized Table

An index-organized table is like a heap table with a primary key index on one or more of its columns. However, instead of maintaining two separate storage spaces for the table and a B-tree index, an index-organized table maintains a single B-tree containing the primary key of the table and other column values. An overflow segment may exist due to the `PCTTHRESHOLD` value being set and the result of longer row lengths requiring the overflow area.

Index-organized tables provide fast key-based access to table data for queries involving exact matches and range searches.

Also, storage requirements are reduced because key columns are not duplicated in the table and index. The remaining non-key columns are stored in the index unless the index entry becomes very large; in that case, the Oracle server provides an `OVERFLOW` clause to handle the problem.

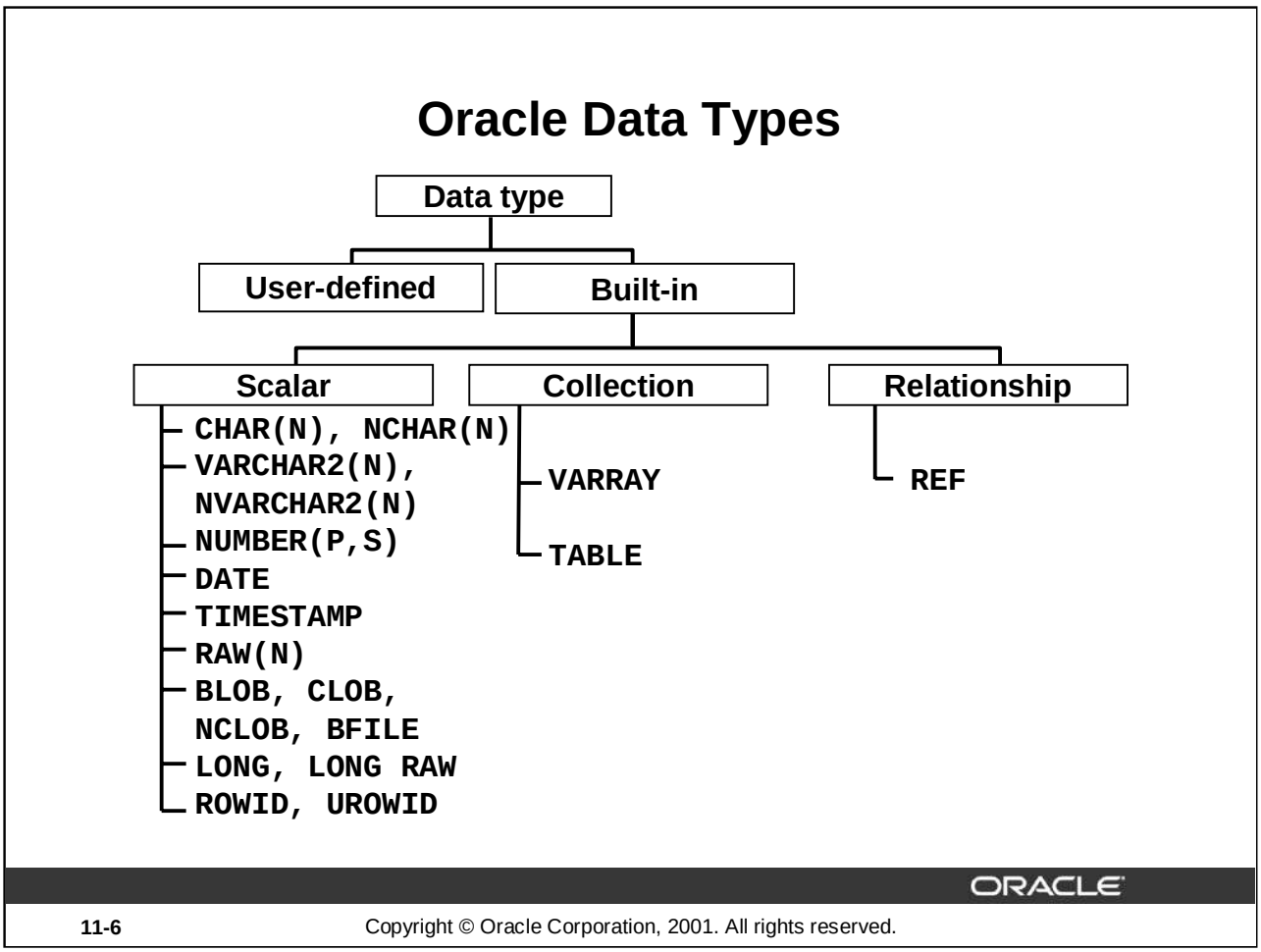
Using Different Methods for Storing User Data (continued)

Clustered Table

A clustered table provides an optional method for storing table data. A cluster is made up of a table or group of tables that share the same data blocks, which are grouped together because they share common columns and are often used together.

Clusters have the following characteristics:

- Clusters have a *cluster key*, which is used to identify the rows that need to be stored together.
- The cluster key can consist of one or more columns.
- Tables in a cluster have columns that correspond to the cluster key.
- Clustering is a mechanism that is transparent to the applications using the tables. Data in a clustered table can be manipulated as though it were stored in a regular table.
- Updating one of the columns in the cluster key may entail physically relocating the row.
- The cluster key is independent of the primary key. The tables in a cluster can have a primary key, which may be the cluster key or a different set of columns.
- Clusters are usually created to improve performance. Random access to clustered data may be faster, but full table scans on clustered tables are generally slower.
- Clusters renormalize the physical storage of tables without affecting the logical structure.



Oracle Built-in Data Types

The Oracle server provides several built-in data types to store scalar data, collections, and relationships.

Scalar Data Types

Character Data

Character data can be stored as either fixed-length or variable-length strings in the database.

Fixed-length character data types, such as CHAR and NCHAR, are stored with padded blanks. NCHAR is a Globalization Support data type that enables the storage of either fixed-width or variable-width character sets. The maximum size is determined by the number of bytes required to store one character, with an upper limit of 2,000 bytes per row. The default is 1 character or 1 byte, depending on the character set.

Variable-length character data types use only the number of bytes needed to store the actual column value, and can vary in size for each row, up to 4,000 bytes. VARCHAR2 and NVARCHAR2 are examples of variable-length character data types.

Oracle Built-in Data Types - Scalar Data Types (continued)

Numeric Data Numbers in an Oracle database are always stored as variable-length data. They can store up to 38 significant digits. Numeric data types require:

- One byte for the exponent
- One byte for every two significant digits in the mantissa
- One byte for negative numbers if the number of significant digits is less than 38 bytes

DATE Data Type The Oracle server stores dates in fixed-length fields of seven bytes. An Oracle DATE always includes the time.

TIMESTAMP Data Type This data type stores the date and time including fractional seconds up to 9 decimal places. TIMESTAMP WITH TIME ZONE and TIMESTAMP WITH LOCAL TIME ZONE can use time zones to factor items such as Daylight Savings Time. TIMESTAMP and TIMESTAMP WITH LOCAL TIME ZONE can be used in primary keys, TIMESTAMP WITH TIME ZONE can not.

RAW Data Type This data type enables the storage of small binary data. The Oracle server does not perform character set conversion when RAW data is transmitted across machines in a network or if RAW data is moved from one database to another using Oracle utilities. The number of bytes needed to store the actual column value, and can vary in size for each row, up to 2,000 bytes.

Long, Long Raw and Large Object (LOBs) Data Types

Oracle provides six data types for storing LOBs:

- CLOB and LONG for large fixed-width character data
- NCLOB for large fixed-width national character set data
- BLOB and LONG RAW for storing unstructured data
- BFILE for storing unstructured data in operating system files

LONG and LONG RAW data types were previously used for unstructured data, such as binary images, documents, or geographical information, and are primarily provided for backward compatibility. These data types are superseded by the LOB data types. LOB data types are distinct from LONG and LONG RAW, and they are not interchangeable. LOBs will not support the LONG application programming interface (API), and vice versa.

Oracle Built-in Data Types - Scalar Data Types (continued)

It is beneficial to discuss LOB functionality in comparison to the older types (LONG and LONG RAW). Below, LONGs refers to LONG and LONG RAW, and LOBs refer to all LOB data types.

LONG, LONG RAW	LOB
Single column per table	**Multiple columns per table**
Up to 2 gigabytes	**Up to 4 gigabytes**
Data stored in-line	**Data stored in-line or out-of-line**
No object type support	**Supports object types**
Sequential access to chunk	**Random access to chunks**

LOBs store a locator in the table and the data elsewhere, unless the size is less than the maximum size for a VARCHAR2 data type, which is 4,000 bytes; LONGs store all data in-line. In addition, LOBs allow data to be stored in a separate segment and tablespace, or in a host file.

LOBs support object type attributes (except NCLOBs) and replication; LONGs do not.

LONGs are primarily stored as chained row pieces, with a row piece in one block pointing to the next row piece stored in another block. Therefore, they need to be accessed sequentially. In contrast, LOBs support random piece-wise access to the data through a file-like interface.

ROWID and UROWID Data Type

ROWID is a data type that can be queried along with other columns in a table. It has the following characteristics:

- ROWID is a unique identifier for each row in the database.
- ROWID is not stored explicitly as a column value.
- Although the ROWID does not directly give the physical address of a row, it can be used to locate the row.
- ROWID provides the fastest means of accessing a row in a table.
- ROWIDs are stored in indexes to specify rows with a given set of key values.

With release 8.1, the Oracle server provides a single datatype called the universal rowid or UROWID. It supports rowids of foreign tables (non-Oracle tables) and can store all kinds of rowids. For example: A UROWID datatype is required to store a ROWID for rows stored in an IOT. The value of the parameter `COMPATIBLE` must be set to 8.1 or higher to use UROWID.

Oracle Built-in Data Types (continued)

Collections Data Types

Two types of collection data types are available to store data that is repetitive for a given row in a table. Prior to Oracle8*i*, the Objects option was needed to define and use collections. A brief discussion of these types follows.

Varying Arrays (VARRAY) Varying arrays are useful to store lists that contain a small number of elements, such as phone numbers for a customer.

VARRAYS have the following characteristics:

- An array is an ordered set of data elements.
- All elements of a given array are of the same data type.
- Each element has an index, which is a number corresponding to the position of the element in the array.
- The number of elements in an array determines the size of the array.
- The Oracle server allows arrays to be of variable size, which is why they are called VARRAYs, but the maximum size must be specified when declaring the array type.

Nested Tables Nested tables provide a means of defining a table as a column within a table. They can be used to store sets that may have a large number of records such as number of items in an order.

Nested tables generally have the following characteristics:

- A nested table is an unordered set of records or rows.
- The rows in a nested table have the same structure.
- Rows in a nested table are stored separate from the parent table with a pointer from the corresponding row in the parent table.
- Storage characteristics for the nested table can be defined by the database administrator.
- There is no predetermined maximum size for a nested table.

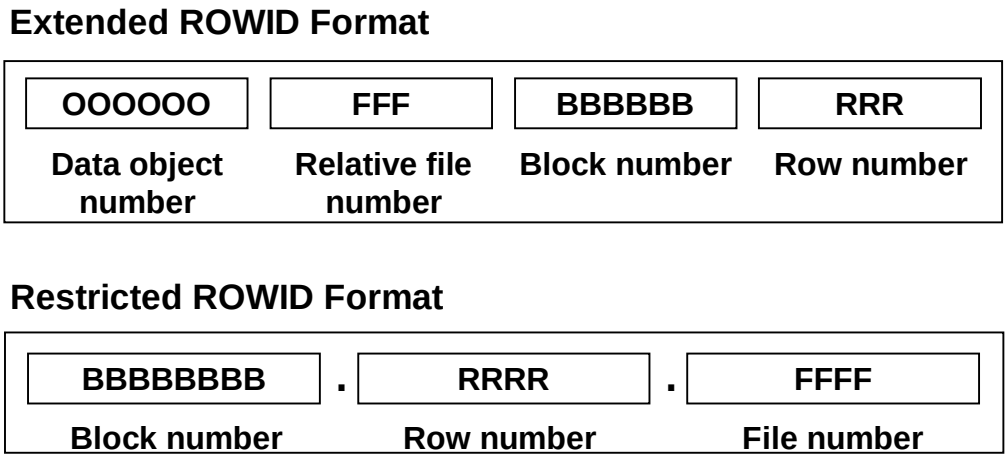
Relationship Data Types (REFs)

Relationship types are used as pointers within the database. The use of these types requires the Objects option. As an example, each item that is ordered could point to or reference a row in the `PRODUCTS` table, without having to store the product code.

Oracle User-Defined Data Types

The Oracle server allows a user to define abstract data types and use them within the application.

ROWID Format



ORACLE

ROWID Format

An Extended ROWID needs 10 bytes of storage on disk and is displayed using 18 characters. It consists of the following components:

- *Data object number* is assigned to each data object, such as table or index when it is created, and it is unique within the database.
- *Relative file number* is unique to each file within a tablespace.
- *Block number* represents the position of the block, containing the row, within the file.
- *Row number* identifies the position of the row directory slot in the block header.

Internally, the data object number needs 32 bits, the relative file number needs 10 bits, block number needs 22 bits, and the row number needs 16 bits, adding up to a total of 80 bits or 10 bytes.

An extended ROWID is displayed using a base-64 encoding scheme, which uses six positions for the data object number, three positions for the relative file number, six positions for the block number, and three positions for the row number. The base-64 encoding scheme uses characters “A-Z,” “a-z,” “0-9,” “+,” and “/”—a total of 64 characters, as in the example below:

```
SQL> SELECT department_id, rowid FROM hr.departments;
DEPARTMENT_ID ROWID
------------- ------------------
           10 AAABQMAAFAAAAA6AAA
           20 AAABQMAAFAAAAA6AAB
           30 AAABQMAAFAAAAA6AAC
           40 AAABQMAAFAAAAA6AAD
           50 AAABQMAAFAAAAA6AAE
           60 AAABQMAAFAAAAA6AAF
            ...
```

In this example:

- AAABQM is the data object number.
- AAF is the relative file number.
- AAAAA6 is the block number.
- AAA is the row number for the department with ID=10.

Restricted ROWID in Oracle7 and Earlier

Versions of Oracle prior to Oracle8 used the restricted ROWID format. A restricted ROWID used only six bytes internally and did not contain the data object number. This format was acceptable in Oracle7 or an earlier release because the file numbers were unique within a database. Thus, earlier releases did not permit more than 1,022 data files. Now it is the limit for a tablespace.

Even though Oracle8 removed this restriction by using tablespace-relative file numbers, the restricted ROWID is still used in objects like nonpartitioned indexes on nonpartitioned tables where all the index entries refer to rows within the same segment.

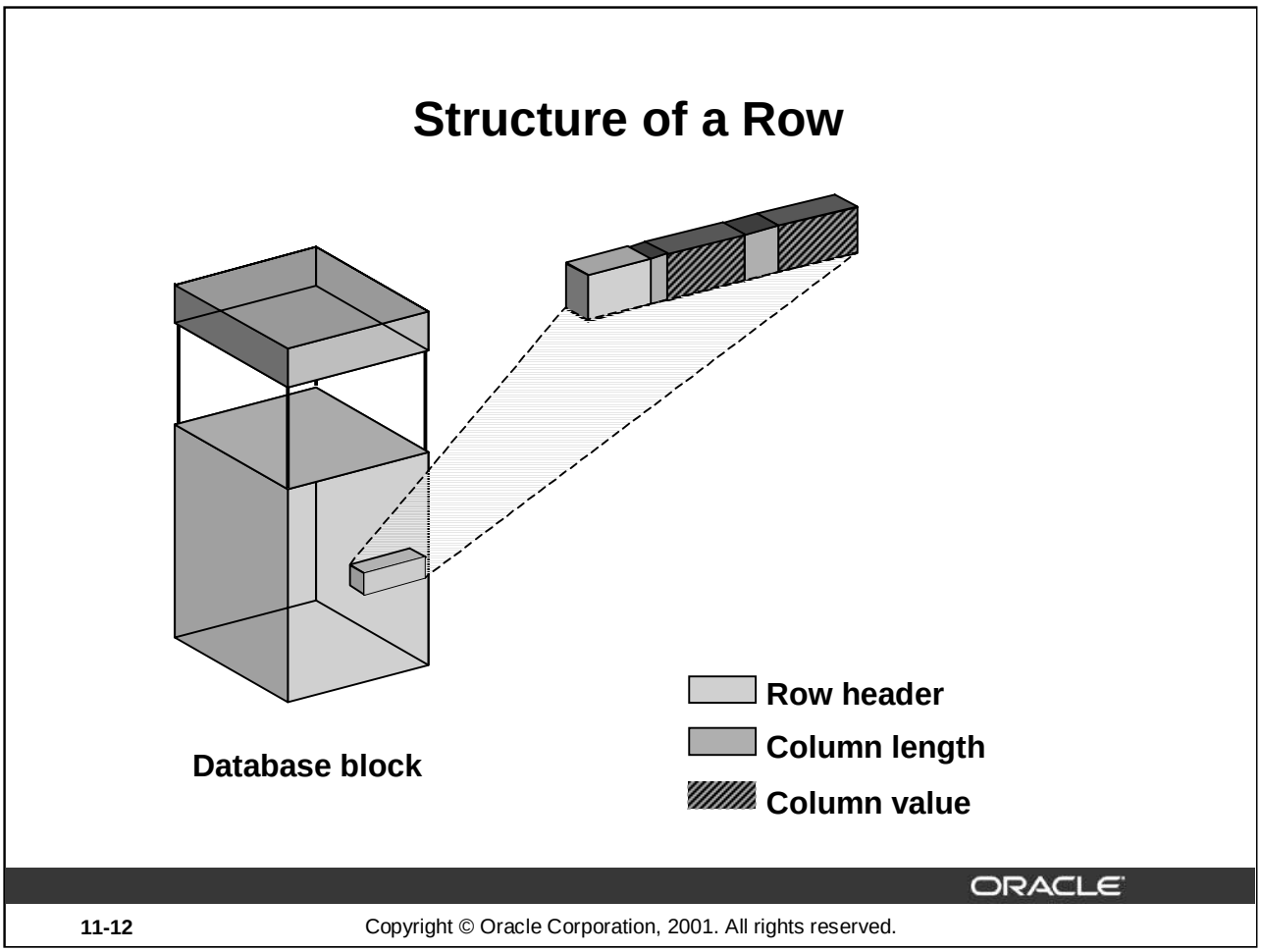
Locating a Row Using ROWID

Because a segment can only reside in one tablespace, using the data object number, the Oracle server can determine the tablespace that contains a row.

The relative file number within the tablespace is used to locate the file, the block number is used to locate the block containing the row, and the row number is used to locate the row directory entry for the row.

The row directory entry can be used to locate the beginning of the row.

Thus, ROWID can be used to locate any row within a database.



Structure of a Row

Row data is stored in database blocks as variable-length records. Columns for a row are generally stored in the order in which they are defined and any trailing `NULL` columns are not stored. Note: A single byte for column length is required for non trailing `NULL` columns. Each row in a table may have a different number of columns. Each row in a table has:

- A row header: Used to store the number of columns in the row, the chaining information, and the row lock status
- Row data: For each column, the Oracle server stores the column length and value (One byte is needed to store the column length if the column cannot exceed 250 bytes. A column that can be longer needs three length bytes. The column value is stored immediately following the column length bytes.)

Adjacent rows do not need any space between them. Each row in the block has a slot in the row directory. The directory slot points to the beginning of the 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)
   );
```

ORACLE

Create Table

The CREATE TABLE command is used to create relational tables or object tables.

Relational table: is the basic structure to hold user data.

Object table: is a table that uses an object type for a column definition. An object table is a tale explicitly defined to hold object instance of a particular type.

Note: Object tables will not be covered within this lesson.

Creating Table Guidelines

- Place tables in separate tablespaces.
- Use locally-managed tables to avoid fragmentation.

The above table contains eleven columns and has been created as locally managed.

Note: Refer to the *Oracle9i SQL Reference* document for more information on the various clauses and parameters that can be defined when using the CREATE TABLE command.

To create a relational table in your own schema, you must have the CREATE TABLE system privilege. To create a table in another user's schema, you must have CREATE ANY TABLE system privilege.

Note: Refer to the *Managing Privileges* lesson for details regarding granting privileges.

Create Table (continued)

The example below creates a departments table as data dictionary managed.

```
CREATE TABLE hr.departments(
department_id  NUMBER(4),
department_name  VARCHAR2(30),
manager_id  NUMBER(6)
location_id  NUMBER(4))
STORAGE(INITIAL 200K NEXT 200K
     PCTINCREASE 0  MINEXTENTS 1 MAXEXTENTS 5)
TABLESPACE data;
```

The above syntax is a subset of the `CREATE TABLE` clause.

STORAGE Clause

`The STORAGE` clause specifies storage characteristics for the table. The storage allocated for the first extent is 200K. When a second extent is required it will be created at 200K also defined by the `NEXT` value. When a third extent is required, it will be created at 200K because the `PCTINCREASE` has been set to 0. The maximum amount of extents that can be used is set at 5. With the minimum set to 1.

- `MINEXTENTS:` The minimum number of extents to be allocated.
- `MAXEXTENTS:` The maximum number of extents to be allocated.
- `PCTINCREASE:` The percent of increase in extent size after NEXT extent and thereafter.

Create Table (continued)

Block Utilization Parameters can also be specified in the physical_attributes_clause for the table.

- PCTFREE: Specifies the percentage of space in each data block of the table. The value of PCTFREE must be a value from 0-99. A value of 0 means that the entire block can be filled by inserts of new rows. The default value is 10. This value reserves 10% of each block for updates to existing rows and allows inserts of new rows to fill a maximum of 90% of each block.
- PCTUSED: Specifies the minimum percentage of used space that is maintained for each data block of the table. A block becomes a candidate for row insertion when its used space falls below PCTUSED. PCTUSED is specified as integer from 0-99 and defaults to 40.

  The combination of PCTFREE and PCTUSED determines whether new rows will be inserted into existing data blocks or into new bocks. The sum of these two must be equal to or less than 100. These parameters are used to utilize space within a table more efficiently.

  Note: These block utilization parameters are deprecated with the Oracle9i feature Automatic Segment-Space Management. Refer to the Storage Structures and Relationship lesson for details regarding this feature.
- INITRANS: Specifies the initial number of transaction entries allocated within each data block allocated to the table. This value can range from 1-255 and default to 1 INITRANS ensures that a minimum number of concurrent transaction can update the block. In general, this value should not be changed from its default.
- MAXTRANS: Specifies the maximum number of concurrent transaction that can update a data block allocated to the table. This limit does not apply to queries. The value can range from 1-255 and the default is a function of the data block size.

TABLESPACE Clause

The TABLESPACE clause specifies the tablespace where the table will be created. The table in the example will reside within the data tablespace. If you omit TABLESPACE, then Oracle creates the object in the default tablespace of the owner of the schema containing the table.

Note: Refer to the *Managing Tablespaces* lesson for more information regarding tablespaces.

How to Create a Table Using Schema Manager

Launch Schema Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Schema folder and select Table in the navigator tree
4. Select Object—>Create from the menu bar.
5. Choose Table in the list of objects, select the Use Wizard option, and click Create.
6. Enter your table information in the table wizard, such as table name, tablespace, owner, columns, and data types and sizes. Click Finish.
7. Expand the Tables folder to verify that your table was created.

   Alternatively, select an existing table from the navigator, and use Object—>Create Like to create a new table with the same column and storage characteristics as an existing table.

Other Options

While using Schema Manager, the user also has the option to let the tool automatically define the storage and block utilization parameters based on an estimate of the initial volume, the growth rate, and the DML activity on the table.

Note: You can also launch the Console from Windows NT Start menu

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**

ORACLE

Temporary Tables

Temporary tables can be created to hold session-private data that exists only for the duration of a transaction or session.

The `CREATE GLOBAL TEMPORARY TABLE` command creates a temporary table that can be transaction specific or session specific. For transaction-specific temporary tables, data exists for the duration of the transaction, while for session-specific temporary tables, data exists for the duration of the session. Data in a session is private to the session. Each session can only see and modify its own data. DML locks are not acquired on the data of the temporary tables. The clauses that control the duration of the rows are:

- `ON COMMIT DELETE ROWS` to specify that rows are only visible within the transaction
- `ON COMMIT PRESERVE ROWS` to specify that rows are visible for the entire session

You can create indexes, views, and triggers on temporary tables and you can also use the Export and Import utilities to export and import the definition of a temporary table. However, no data is exported, even if you use the `ROWS` option. The definition of a temporary table is visible to all sessions.

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.**

ORACLE

Guidelines for Creating a Table

Place tables in a separate tablespace—not in the tablespace that has undo segments, temporary segments, and indexes.

Place tables in locally-managed tablespaces to avoid fragmentation.

Changing Storage Parameters

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

ORACLE

Changing Storage and Block Utilization Parameters

Some of the storage parameters and any of the block utilization parameters can be modified by using the ALTER TABLE command.

Syntax

```
ALTER TABLE [schema.]table
{[ storage-clause ]
[ INITRANS integer ]
[ MAXTRANS integer]}
```

The Effects of Changing Storage Parameters

The parameters that can be modified and the implications of the modifications are as follows:

- NEXT: When the Oracle server allocates another extent for the table, the new value will be used. Subsequent extent sizes will increase by PCTINCREASE.

Effects of Changing Storage Parameters (continued)

- PCTINCREASE: A change in PCTINCREASE will be registered in the data dictionary. It will be used to recalculate NEXT when the next extent is allocated by the Oracle server. Consider a case where a table with two extents has NEXT=10K and PCTINCREASE=0. If PCTINCREASE is changed to 100, the third extent to be allocated will be 10K, the fourth extent will be 20K, the fifth extent will be 40K, and so on.
- MINEXTENTS: The value of MINEXTENTS can be changed to any value that is less than or equal to the current number of extents in the table. It will have no immediate effect on the table, but will be used if the table is truncated.
- MAXEXTENTS: The value of MAXEXTENTS can be set to any value equal to or greater than the current number of extents for the table. The value can also be set to UNLIMITED.

Restictions

- The value of INITIAL cannot be modified for a table.
- The value of NEXT specified will be rounded to a value that is a multiple of the block size greater than or equal to the value specified.

Using Oracle Enterprise Manager to Change Storage Parameters

Launch Schema Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Schema folder
4. Expand the Tables folder.
5. Expand the user name (or schema).
6. Select the table.
7. Modify the values in the Storage tab of the property sheet. Note that the minimum extents and initial number of transactions cannot be modified using this method.
8. Click Apply.

Note: You can also launch the Console from Windows NT Start menu

Manually Allocating Extents

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

ORACLE

Manually Allocating Extents

Extents may need to be allocated manually:

- To control the distribution of extents of a table across files
- Before loading data in bulk to avoid dynamic extension of tables

Syntax

Use the following command to allocate an extent to a table:

```
ALTER TABLE [schema.]table
ALLOCATE EXTENT [ ([SIZE integer [K|M]]
                [ DATAFILE 'filename' ]) ]
```

If SIZE is omitted, the Oracle server will use the NEXT_EXTENT size from DBA_TABLES to allocate the extent.

The file specified in the DATAFILE clause must belong to the tablespace that the table belongs to. Otherwise, the statement will generate an error. If the DATAFILE clause is not used, the Oracle server will allocate the extent in one of the files in the tablespace containing the table.

Note: The NEXT_EXTENT value in DBA_TABLES will not be affected by manual extent allocation. The Oracle server will not recalculate the size of the next extent when this command is executed.

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**

ORACLE

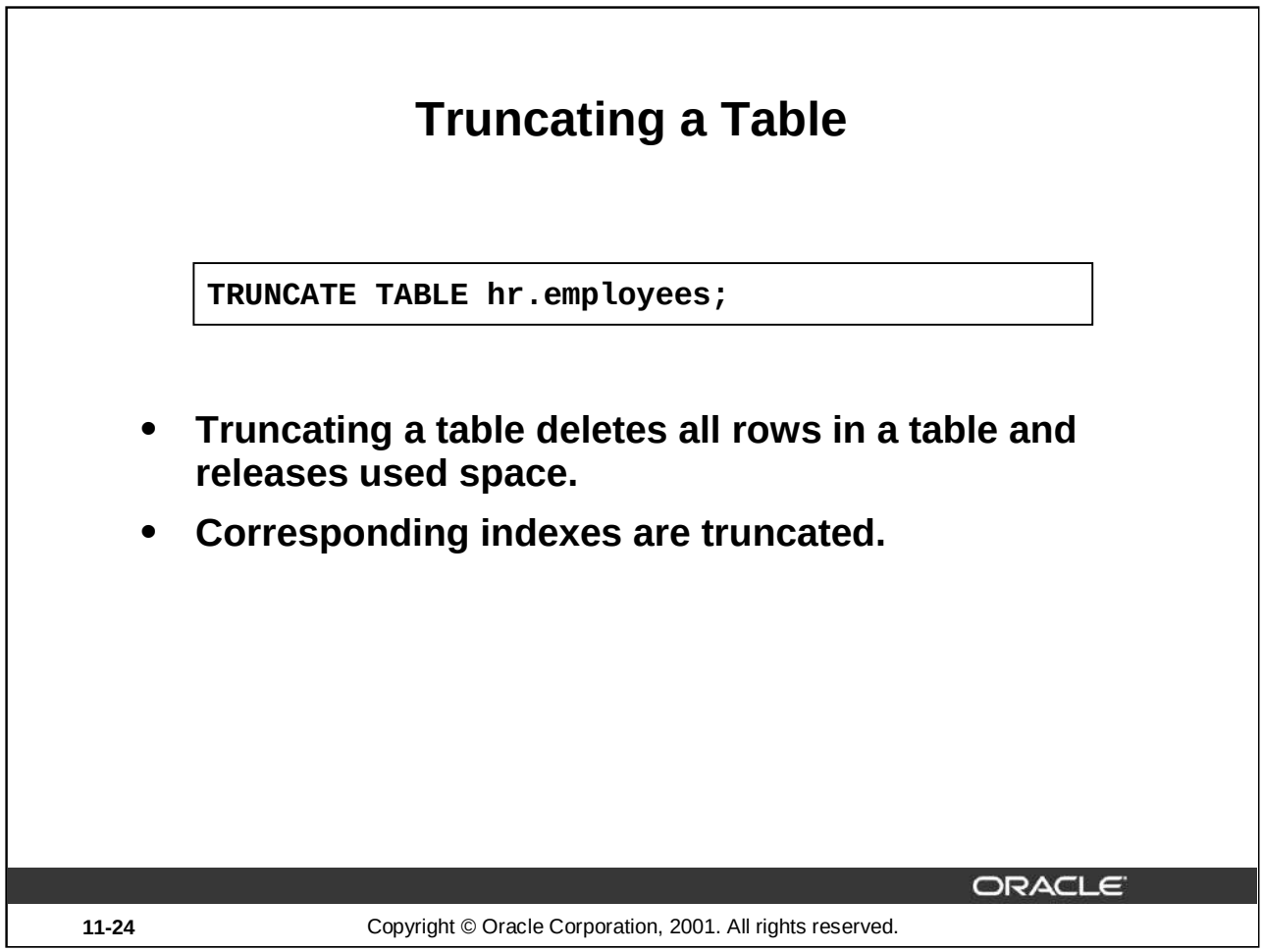
Relocating or Reorganizing a Table

A nonpartitioned table can be moved without having to run the Export or Import utility. In addition, it allows the storage parameters to be changed. This is useful when:

- Moving a table from one tablespace to another
- Reorganizing the table to eliminate row migration

After moving a table you will have to rebuild the indexes to avoid the following error:

```
SQL> select * from employee where id=23;
select * from employee where id=23
              *
ERROR at line 1:
ORA-01502: index 'SUMMIT.EMPLOYEE_ID_PK' or partition of such
index is in unusable state
```



Truncating a Table

Syntax

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

The effects of using this command are as follows:

- All rows in the table are deleted.
- No undo data is generated and the command commits implicitly because TRUNCATE TABLE is a DDL command.
- Corresponding indexes are also truncated.
- A table that is being referenced by a foreign key cannot be truncated.
- The delete triggers do not fire when this command is used.

Using Oracle Enterprise Manager to Drop a Table

1. Use Schema Manager.
2. Expand the Tables folder.
3. Expand the user name (or schema).
4. Select the table.
5. Select Object—>Remove.
6. Select Yes in the dialog box.

Dropping a Table

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

ORACLE

Dropping a Table

A table may be dropped if it is no longer needed or if it is to be reorganized.

Syntax

Use the following command to drop a table:

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

When a table is dropped, the extents used by the table are released. If they are contiguous, they may be coalesced either automatically or manually at a later stage.

The `CASCADE CONSTRAINTS` option is necessary if the table is the parent table in a foreign key relationship.

Note: Refer to the Maintaining Data Integrity lesson for details regarding `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**

ORACLE

Removing a Column From a Table

The Oracle server enables you to drop columns from rows in a table. Dropping columns cleans unused and potentially space-demanding columns without having to export or import data, and recreate indexes and constraints.

Dropping a column can take a significant amount of time because all the data for the column is deleted from the table.

Before Oracle8*i*, it was not possible to drop a column from a table.

Using a Checkpoint When Dropping a Column

Dropping a column can be time consuming and require a large amount of undo space. While dropping columns from large tables, checkpoints can be specified to minimize the use of undo space. In the example in the slide, a checkpoint occurs every 1,000 rows. The table is marked `INVALID` until the operation completes. If the instance fails during the operation, the table remains `INVALID` on start up, and the operation will have to be completed.

Use the following statement to resume an interrupted drop operation:

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

Use of this will generate an error if the table is in a `VALID` state.

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;
```

ORACLE

Marking a Column as Unused

Instead of removing a column from a table, the column can be marked as unused and then removed later. This has the advantage of being relatively quick, as it does not reclaim the disk space because the data is not removed. Columns marked as unused can be removed at a later time from the table when there is less activity on the system.

Unused columns act as if they are not part of the table. Queries cannot see data from unused columns. In addition, the names and data types of those columns are not displayed when a `DESCRIBE` command is executed. A user can add a new column with the same name as an unused column.

An example of setting a column to unused before dropping it would be if you want to drop two columns in the same table. When dropping two columns all rows in the table are updated twice, by setting the columns to unused and then drop the columns the rows will only be updated once.

Identifying Tables with Unused Columns

To identify tables with unused columns, you can query the view DBA_UNUSED_COL_TABS. It obtains the names of tables that have unused columns and the number of columns that are marked unused in them. The following query shows that the table EMPLOYEES owned by HR has one unused column:

```
SELECT * FROM dba_unused_col_tabs;

OWNER TABLE_NAME     COUNT
----- -------------- ------
HR    EMPLOYEES           1
```

To identify tables that have partially completed DROP COLUMN operations the DBA_PARTIAL_DROP_TABS view can be queried.

```
SELECT * FROM dba_partial_drop_tabs;

OWNER TABLE_NAME     COUNT
----- -------------- ------
no rows selected
```

Restrictions on Dropping a Column

You cannot do the following:

- Drop a column from an object type table
- Drop columns from nested tables
- Drop all columns in a table
- Drop a partitioning key column
- Drop a column from tables owned by SYS
- Drop a parent key column
- Drop a column from an index-organized table if the column is a primary key
- A LONG or LONG RAW column that is unused but not dropped will prevent an add of a LONG or LONG RAW column to the table. (Even though a describe of the table appears to show no LONG or LONG RAW column.

Obtaining Table Information

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

- **DBA_TABLES**
- **DBA_OBJECTS**

ORACLE

Obtaining Table Information

Information about tables can be obtained from the data dictionary. To obtain the data object number and the location of the table header for all tables owned by HR, use the following query:

```
    SELECT table_name FROM dba_tables WHERE owner = 'HR';
TABLE_NAME
--------------------
COUNTRIES
DEPARTMENTS
DEPARTMENTS_HIST
EMPLOYEES
EMPLOYEES_HIST
JOBS
JOB_HISTORY
LOCATIONS
REGIONS
```

Obtaining Data Dictionary Information (continued)

```
SQL> select object_name, created
  2  from DBA_OBJECTS
  3  where object_name like 'EMPLOYEES'
  4  and owner = 'HR';

OBJECT_NAME  CREATED
-----------  ---------
EMPLOYEES    16-APR-01
```

Quick Reference

Context	Reference
Initialization parameters	DB_BLOCK_SIZE
Dynamic initialization parameters	DB_FILE_MULTIBLOCK_READ_COUNT
Dynamic performance views	None
Data dictionary views	DBA_TABLES DBA_SEGMENTS DBA_OBJECTS DBA_EXTENTS
Commands	CREATE TABLE CREATE GLOBAL TEMPORARY TABLE ALTER TABLE ... STORAGE ALTER TABLE ... PCTFREE ... PCTUSED ALTER TABLE ... ALLOCATE EXTENT ALTER TABLE ... DEALLOCATE UNUSED TRUNCATE TABLE DROP TABLE ALTER TABLE ... DROP COLUMN ALTER TABLE ... SET UNUSED COLUMN ALTER TABLE ... DROP UNUSED COLUMNS ALTER TABLE ... DROP COLUMNS CONTINUE

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**

ORACLE

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**

ORACLE

Practice 11: Managing Tables

1. Create the following tables as user SYSTEM for an order entry system that you are implementing now. The tables and the columns are shown below:

Table	Column	Data Type and Size
CUSTOMERS	CUST_CODE	VARCHAR2(3)
	NAME	VARCHAR2(50)
	REGION	VARCHAR2(5)
ORDERS	ORD_ID	NUMBER(3)
	ORD_DATE	DATE
	CUST_CODE	VARCHAR2(3)
	DATE_OF_DELY	DATE

   You have been informed that in the table ORDERS, rows will be inserted without a value for DATE_OF_DELY, and it will be updated when the order is fulfilled. Use tablespace USERS. You can use the default storage settings.

2. Run the script lab11_02.sql to insert rows into the tables.

3. Find which files and blocks contain the rows for the orders table.

   Hint: Query data dictionary view DBA_EXTENTS.

4. Check the number of extents used by the table ORDERS.

5. Allocate an extent manually, with default size, for the table ORDERS and confirm that the extent has been added as specified.

6. Create another table, ORDERS2 as copy of the ORDERS table, but with MINEXTENTS=10. Verify that the table has been created with the specified number of extents.

7. Truncate table ORDERS without releasing space and check the number of extents to verify extents have not been deallocated.

8. Truncate the ORDERS2 table, releasing space. How many extents does the table have now?

9. Run the script lab11_09.sql to insert some rows into the ORDERS2 table.

Practice 11: Managing Tables

10 View the columns for the ORDERS2 table. Then mark the DATE_OF_DELY column as UNUSED. View the columns for the ORDERS2 table again. What happens?

11 Drop the unused column DATE_OF_DELY.

12 Drop the ORDERS2 table.