Oracle9*i* Database Administration Fundamentals I

Volume 2 • Student Guide

D11321GC20
Production 2.0
September 2002
D37262

ORACLE®

Author

Marie St. Gelais

Technical Reviewers

Louise Beijer

Dairy Chan

Trevor Davies

Donna Hamby

Lutz Hartmann

Kuljit Jassar

Patricia Mesa

Sabiha Miri

Howard Ostrow

Caroline Pereda

Andreas Reinhardt

Ajai Sahni

Jaco Verheul

Publisher

Shane Mattimoe

Contents

Preface

Profile

This course is designed to give the Oracle database administrator (DBA) a firm foundation in basic administrative tasks. The primary goal of this course is to give the DBA the necessary knowledge and skills to set up, maintain, and troubleshoot an Oracle database. This course has been designed for junior database administrators, technical support analysts, system administrators, application developers, MIS managers, and other Oracle user.

This preface covers the following sections:

- Before You Begin This Course
- Prerequisites
- How This Course Is Organized
- Related Publications
- Typographic Conventions

Before You Begin This Course

The specific skills you as a participant must have in order to derive the maximum value from attending this course are:

- Familiarity with relational database concepts
- Thorough knowledge of SQL, SQL*Plus and OS commands (UNIX and NT)
- Basic operating system knowledge
- Some working experience with the Oracle environment

Prerequisites

- Introduction to Oracle9*i*: SQL Basics

How This Course Is Organized

Oracle9*i* Database Administration Fundamentals I is an instructor-led course featuring lecture and hands-on exercises. This course also allows the capability to accomplish the practices using SQL*Plus or the Oracle Enterprise Manager (OEM). In addition, this course contains clearly defined objectives designed to support preparation for the Oracle Certified Professional examination.

Related Publications

Oracle Publications

Title	Part Number
Oracle9*i* Backup and Recovery Concepts	A96519-01
Oracle9*i* Database Administrator's Guide	A96521-01
Oracle9*i* Database Concepts	A96524-01
Oracle9*i* Database Error Messages	A96525-01
Oracle9*i* Database New Features	A96531-01
Oracle9*i* Database Reference	A96536-01
Oracle9*i* Database Utilities	A96652-01
Oracle9*i* Enterprise Manager Administrator's Guide	A96670.01
Oracle9*i* Enterprise Manager Concepts Guide	A96674-01
Oracle9*i* Enterprise Manager Configuration Guide	A96673-01
Oracle9*i* Net Services Administrator's Guide	A96580-01
Oracle9*i* Net Services Reference Guide	A96581-01
Oracle9*i* Recovery Manager Reference	A96565-01
Oracle9*i* Recovery Manager User's Guide	A96566-01
Oracle9*i* SQL Reference	A96540-01
Oracle9*i* User-Managed Backup and Recovery Guide	A96572-01

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.
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;`

15

Managing Users

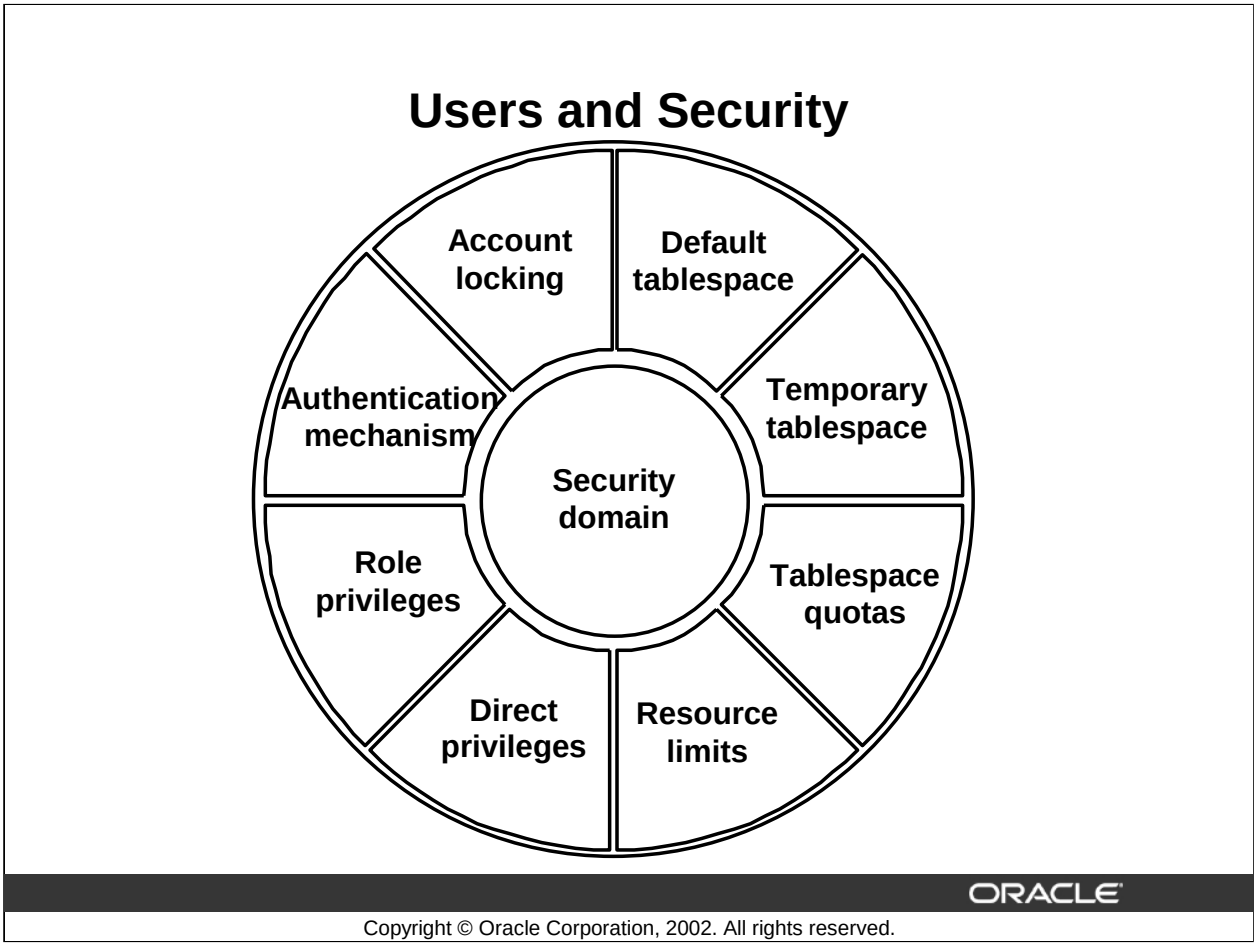
ORACLE

Objectives

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

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

ORACLE



Users and Security

Security domain

The database administrator defines the names of the users who are allowed to access a database. A security domain defines the settings that apply to the user.

Authentication mechanism

A user who requires access to the database can be authenticated by one of the following:

- Data dictionary
- Operating system
- Network

The means of authentication is specified at the time the user is defined in the database and can be altered later. This lesson covers authentication by database and by operating system only.

Note: Refer to the “Getting Started with the Oracle Server” lesson for details regarding operating system authentication using roles.

Authentication through the network is covered in the *Oracle9i Database Administration Fundamentals II* course.

Users and Security (continued)

Tablespace quotas

Tablespace quotas control the amount of physical storage space that is allocated to a user in the tablespaces in the database.

Default tablespace

The default tablespace defines the location where segments that are created by a user are stored if the user does not explicitly specify a tablespace at the time the segment is created.

Temporary tablespace

Temporary tablespace defines where extents will be allocated by the Oracle server if the user performs an operation that requires writing sort data to the disk.

Account locking

Accounts can be locked to prevent a user from logging on to the database. This can be set to occur automatically, or the database administrator can lock and unlock accounts manually.

Resource limits

Limits can be placed on the use of resources such as CPU time, logical input/output (I/O), and the number of sessions that a user opens.

Direct privileges

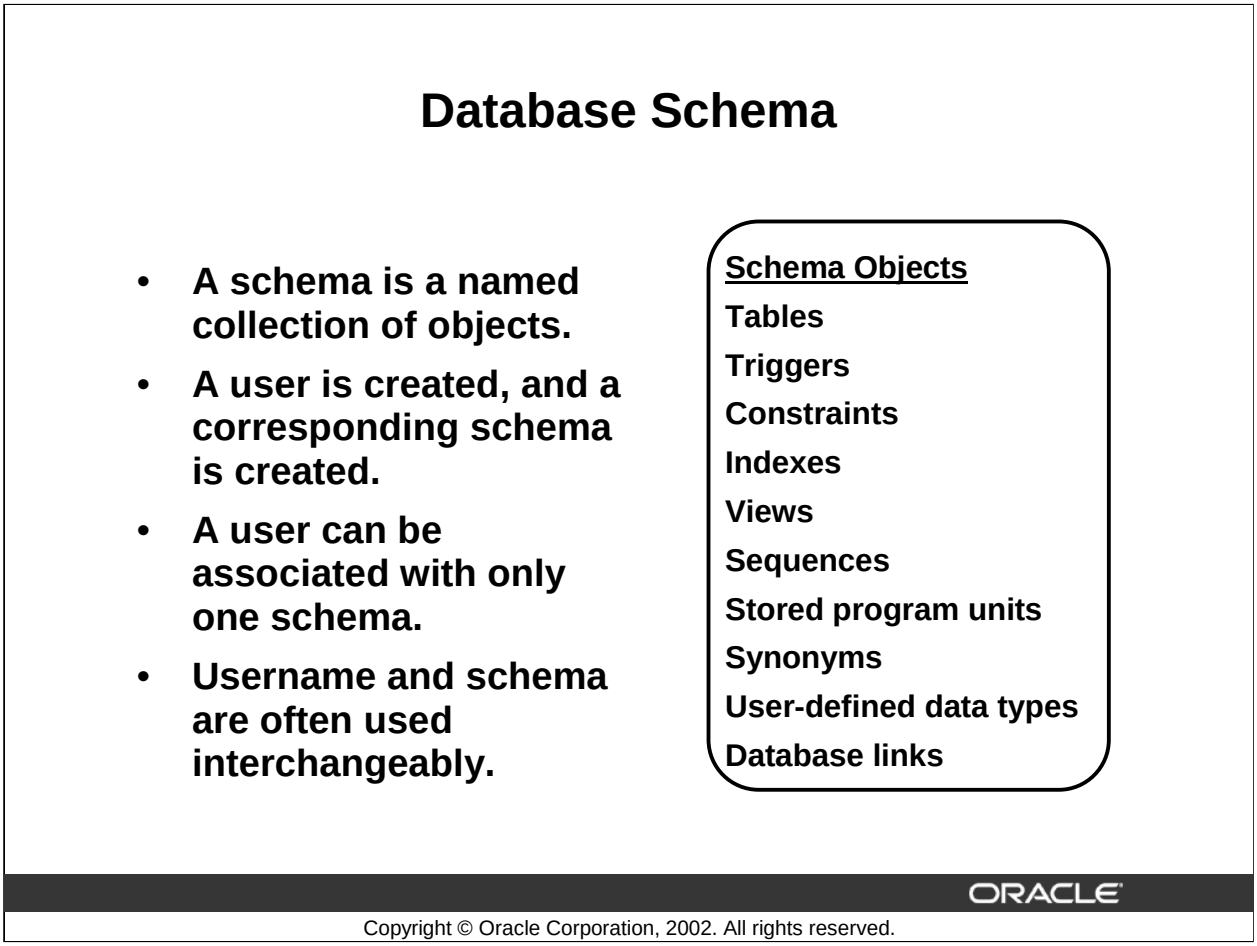
Privileges are used to control the actions that a user can perform in a database.

Role privileges

A user can be granted privileges indirectly through the use of roles.

Note: Refer to the lessons “Managing Privileges” and “Managing Roles” for information regarding role privileges.

This lesson covers defining a user with the appropriate authentication mechanism, limiting the use of space by the users in the system, and the manual control of account locking.



Database Schema

A schema is a named collection of objects such as tables, views, clusters, procedures, and packages that are associated with a particular user. When a database user is created, a corresponding schema with the same name is created for that user. A user can be associated only with a schema of the same name, and therefore *username* and *schema* are often used interchangeably.

The slide shows some of the objects that users can own in an Oracle database.

Checklist for Creating Users

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

ORACLE

Creating a New User: Database Authentication

Set the initial password:

```
CREATE USER aaron
IDENTIFIED BY soccer
DEFAULT TABLESPACE data
TEMPORARY TABLESPACE temp
QUOTA 15M ON data
QUOTA 10M ON users
PASSWORD EXPIRE;
```

ORACLE

Creating a New User: Database Authentication

Syntax

Use the following command to create a new user:

```
CREATE USER user
IDENTIFIED {BY password | EXTERNALLY|GLOBALLY AS
  external name}
[ DEFAULT TABLESPACE tablespace ]
[ TEMPORARY TABLESPACE tablespace ]
[ QUOTA {integer [K | M ] | UNLIMITED } ON tablespace
[ QUOTA {integer [K | M ] | UNLIMITED } ON tablespace
]...]
[ PASSWORD EXPIRE ]
[ ACCOUNT { LOCK | UNLOCK }]
[ PROFILE { profile | DEFAULT }]
```

Creating a New User: Database Authentication (continued)

Syntax (continued):

where:

`user`: Is the name of the user

`BY password`: Specifies a user who is authenticated by the database and must supply a password while logging on

`EXTERNALLY`: Specifies that the user is authenticated by the operating system

`GLOBALLY AS`: Specifies that the user is authenticated globally

`DEFAULT` or `TEMPORARY TABLESPACE`: Identifies the default or temporary tablespace for the user

`QUOTA`: Defines the maximum space allowed for objects owned by the user in the tablespace. (Quota can be defined as integer bytes or kilobytes and megabytes. The `UNLIMITED` keyword is used to specify that the objects owned by the user can use as much space as is available in the tablespace. By default, no user has any quota on any tablespace.)

`PASSWORD EXPIRE`: Forces the user to reset the password when the user logs on to the database using SQL*Plus. (This option is valid only if the user is authenticated by the database.)

`ACCOUNT LOCK/UNLOCK`: Can be used to lock or unlock the user's account explicitly. (`UNLOCK` is the default.)

`PROFILE`: Is used to control resource usage and to specify the password control mechanism to be used for the user

Note: Refer to the **"**Managing Profiles" lesson for information about creating profiles.

A password authentication method is mandatory. If a password is specified, it is maintained by the Oracle server in the data dictionary. Password control mechanisms provided by the Oracle server are available when users are authenticated by the server.

After the password has expired, and when the user logs on using SQL*Plus, the user will be prompted to enter a new password:

```
ERROR:
ORA-28001: the account has expired
Changing password for PETER
Old password:
New password:
Retype new password:
Password changed
```

Creating a New User: Database Authentication (continued)

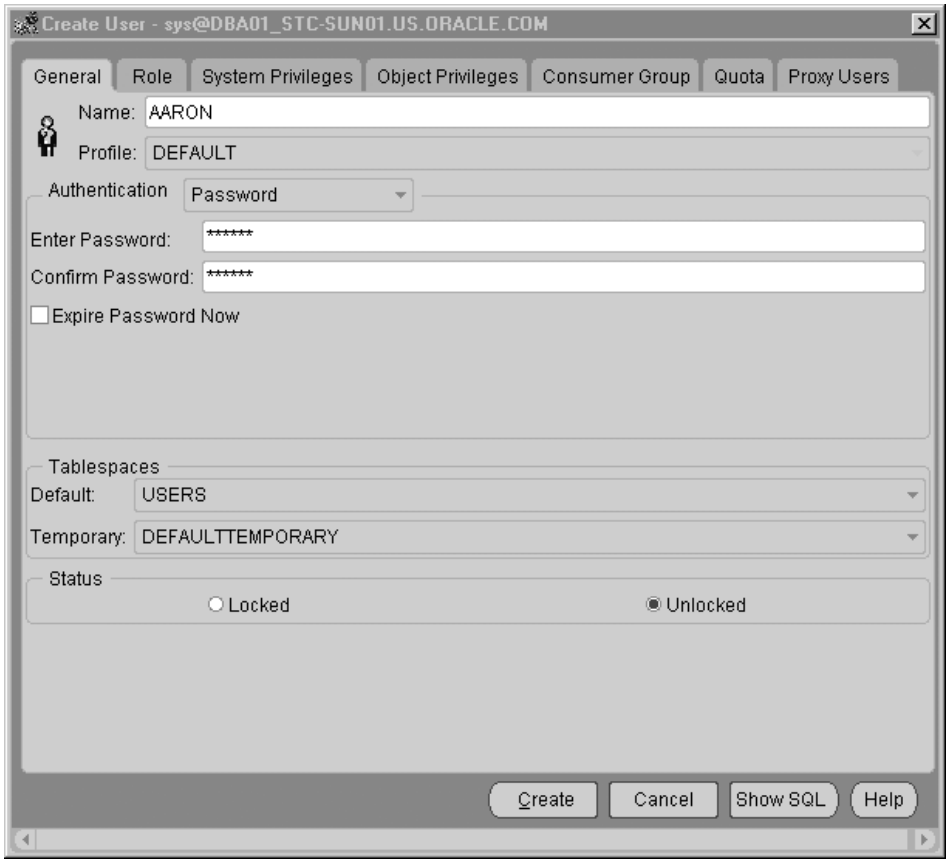
Using Oracle Enterprise Manager to Create a New User

From the OEM Console:

1. Navigate to Security > Users.
2. Select Create from the right-mouse menu.
3. Enter information to create user.
4. Click Create.

Oracle Security Manager automatically grants the `CONNECT` role to any user who is created by using the tool.

Note: Refer to the "Managing Roles" lesson for information about the `CONNECT` role.



Creating a New User: Operating System Authentication

- The OS_AUTHENT_PREFIX initialization parameter specifies the format of the usernames.
- It defaults to OPS$.

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

ORACLE

Creating a New User: Operating System Authentication

Operating system authentication

Use the IDENTIFIED EXTERNALLY clause of the CREATE USER command to specify that a user must be authenticated by the operating system. This option is generally useful when the user logs on directly to the machine where the Oracle server is running.

Username for operating system authentication

The OS_AUTHENT_PREFIX initialization parameter is used to specify the format of the usernames for operating system authentication. This value defaults to OPS$ to make it backward compatible with earlier releases of the Oracle server. To set the prefix to a NULL value, specify this initialization parameter as:

```
OS_AUTHENT_PREFIX = " "
```

The example in the slide shows how a user, aaron, is defined in the database. This specifies that the operating system user aaron will be allowed access to the database without being validated by the Oracle server. Thus, to use SQL*Plus to log on to the system, the UNIX user aaron must enter the following command from the operating system:

```
$ sqlplus /
```

Creating a New User: Operating System Authentication (continued)

Username for operating system authentication

Note

- Using `OS_AUTHENT_PREFIX=OPS$`: Provides the flexibility of having a user authenticated by either the operating system or the Oracle server. In this case, the DBA can create the user by entering a command of the form:
  ```
  CREATE USER ops$user
  IDENTIFIED BY password ...
  ```
- A user who logs on to the machine running the Oracle server need not supply a password. If the user connects from a remote client, he or she can connect by supplying the password.
- Setting another initialization parameter, `REMOTE_OS_AUTHENT=TRUE`, specifies that a user can be authenticated by a remote operating system. The default value of False indicates that a user can be authenticated only by the machine running the Oracle server. Use this parameter with care because there is a potential security problem.
- If there are users in the database who are authenticated by the operating system, changing the `OS_AUTHENT_PREFIX` parameter may prevent these users from logging on to the database.

Changing User Quota on Tablespaces

- **A user's tablespace quotas can be modified for any of the following situations:**
  - **Tables owned by a user exhibit unanticipated growth.**
  - **An application is enhanced and requires additional tables or indexes.**
  - **Objects are reorganized and placed in different tablespaces.**
- **To modify a user's tablespace quota, enter the following:**

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

ORACLE

Changing User Quota on Tablespaces

Use the following command to modify tablespace quotas or to reassign tablespaces:

```
ALTER USER user
[ DEFAULT TABLESPACE tablespace]
[ TEMPORARY TABLESPACE tablespace]
[ QUOTA {integer [K | M] | UNLIMITED } ON tablespace
[ QUOTA {integer [K | M] | UNLIMITED } ON tablespace ]
...]
```

After a quota of `0` is assigned, the objects owned by the user remain in the revoked tablespace, but they cannot be allocated a new space. For example, if a table that is 10 MB exists in the `USERS` tablespace, and the `USERS` tablespace quota is altered to `0`, then no more new extents can be allocated for that table.

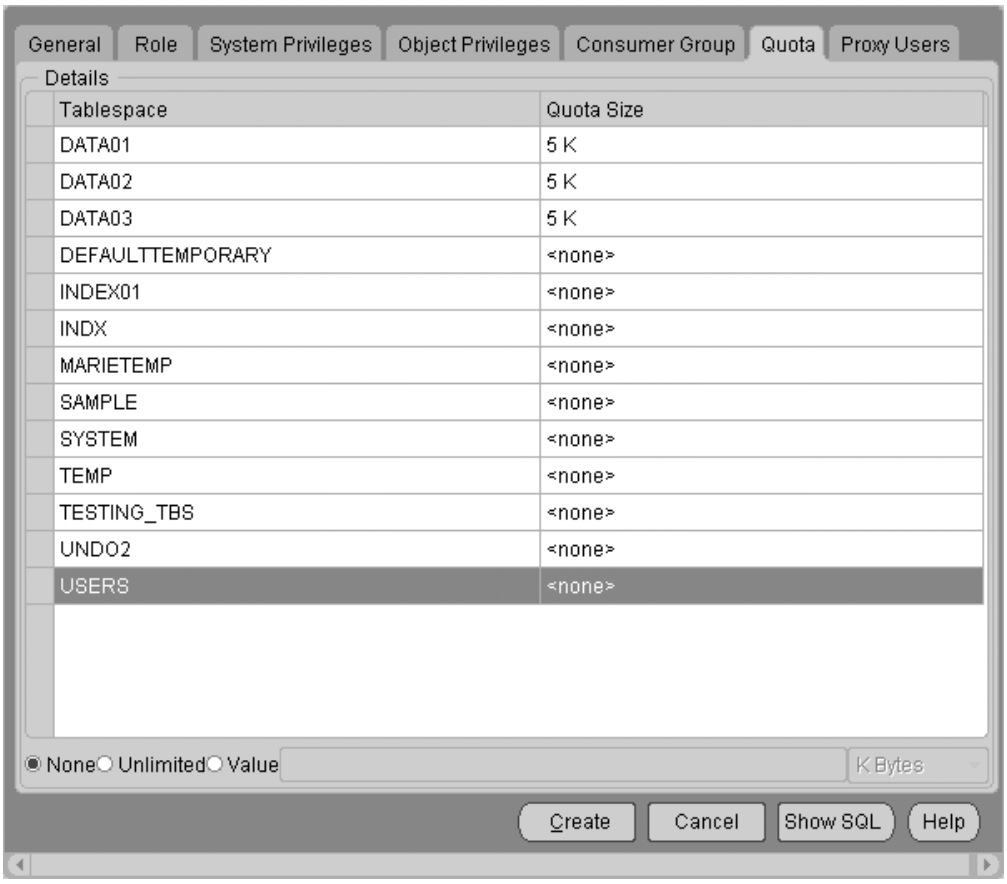
Any unchanged options remain unchanged.

Note: Beware of `UNLIMITED TABLESPACE` privileges because it takes priority over quota settings.

Changing User Quota on Tablespaces (continued)

Using Oracle Enterprise Manager to Modify Tablespace Quota for a User

1. Navigate to Security > Users.
2. Highlight the user.
3. Enter the quota size in the Quota tabbed page.
4. Click Apply.



Dropping a User

- **Use the DROP command to remove a user.**

```
DROP USER aaron;
```

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

```
DROP USER aaron CASCADE;
```

- **Users who are currently connected to the Oracle server cannot be dropped.**

ORACLE

Dropping a User

```
DROP USER user [CASCADE]
```

Guidelines

- The CASCADE option drops all objects in the schema before dropping the user. This must be specified if the schema contains any objects.
- A user who is currently connected to the Oracle server cannot be dropped.

Dropping a User (continued)

Using Oracle Enterprise Manager to Drop a User

From the OEM Console:

1. Navigate to Security > Users.
2. Highlight the user.
3. Select Remove from the right-mouse menu.
4. Click Yes to confirm drop.

Obtaining User Information

Information about users can be obtained by querying the following views:

- **DBA_USERS**
- **DBA_TS_QUOTAS**

Obtaining User Information

Use the following query to find the default tablespace for all users:

```
SQL> SELECT username, default_tablespace
  2  FROM dba_users;

USERNAME   DEFAULT_TABLESPACE
---------  ------------------
SYS        SYSTEM
SYSTEM     SYSTEM
OUTLN      SYSTEM
DBSNMP     SYSTEM
HR         SAMPLE
OE         SAMPLE
```

Summary

In this lesson, you should have learned how to:

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

ORACLE

Practice 15 Overview

This practice covers the following topics:

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

ORACLE

Practice 15 Overview

Note: Practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 15: Managing Users

1. Create user Bob with a password of CRUSADER. Make sure that any objects and temporary segments created by Bob are not created in the system tablespace. Also, make sure that Bob can log in and create objects up to one megabyte in size in the USERS and INDX tablespaces. Use lab15_01.sql script to grant Bob the ability to create sessions.
   Hint: Assign Bob the default tablespace of USERS and temporary tablespace TEMP.
2. Create a user Emi with a password of Mary. Make sure that any objects and sort segments created by Emi are not created in the system tablespace.
3. Display the information on Bob and Emi from the data dictionary.
   Hint: This can be obtained by querying DBA_USERS.
4. From the data dictionary, display the information on the amount of space that Bob can use in tablespaces.
   Hint: This can be obtained by querying DBA_TS_QUOTAS.
5. **a** As user Bob, change his temporary tablespace. What happens?
   b As user Bob, change his password to Sam.
6. As user SYSTEM, remove Bob's quota on his default tablespace.
7. Remove the Emi account from the database.
8. Bob has forgotten his password. Assign him a password of OLINK and require that Bob change his password the next time he logs on.

16

Managing Privileges

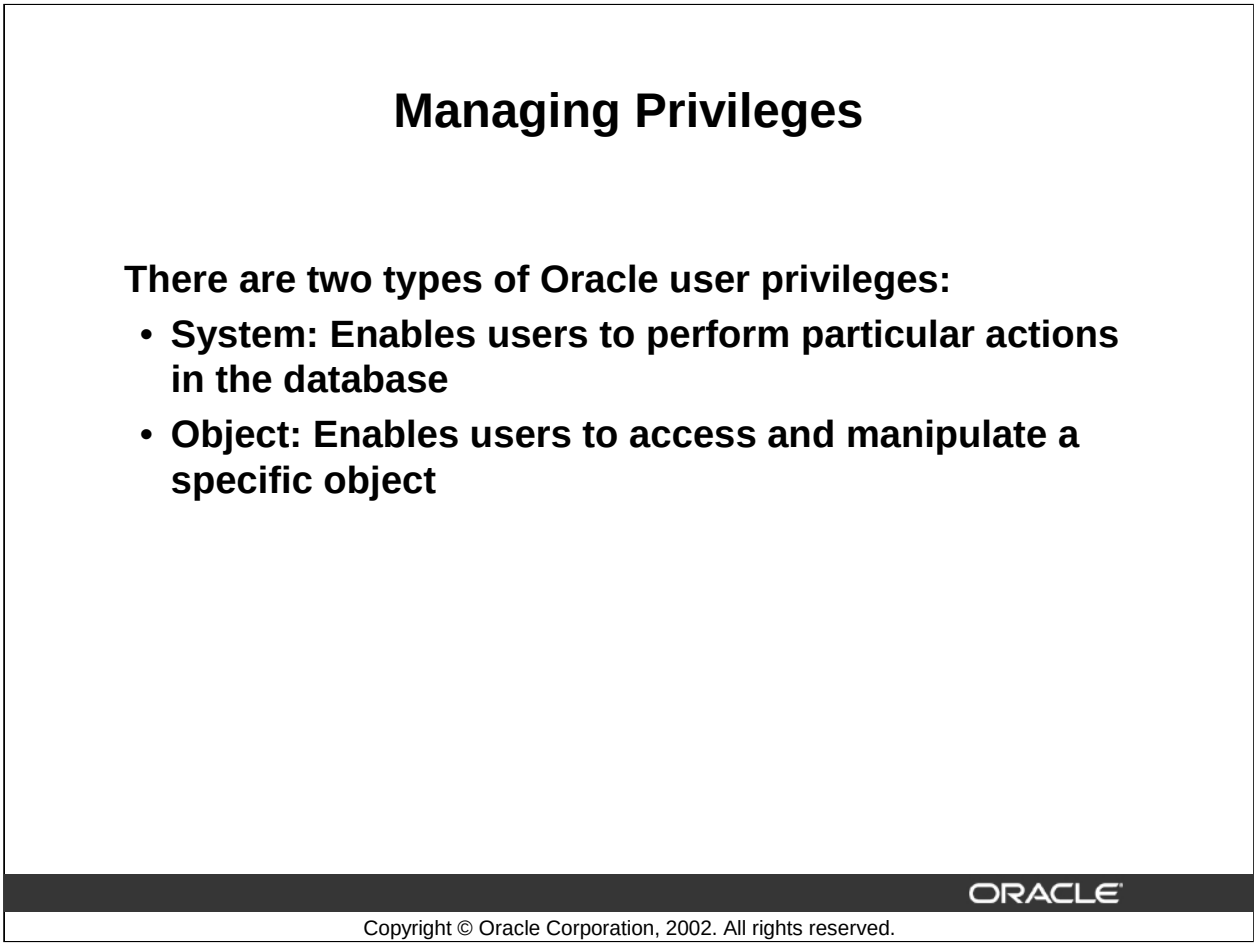
ORACLE

Objectives

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

- **Identify system and object privileges**
- **Grant and revoke privileges**
- **Obtain privilege information**

ORACLE



Managing Privileges

A privilege is a right to execute a particular type of SQL statement or to access another user's object. These include the right to:

- Connect to a database
- Create a table
- Select rows from another user's table
- Execute another user's stored procedure

System privileges

Each system privilege allows a user to perform a particular database operation or class of database operations; for example, the privilege to create tablespaces is a system privilege.

Object privileges

Each object privilege allows a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package.

A DBA's control of privileges includes:

- Providing a user the right to perform a type of operation
- Granting and revoking access to perform system functions
- Granting privileges directly to users or to roles
- Granting privileges to all users (`PUBLIC`)

System Privileges

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

ORACLE

System Privileges

Privileges can be classified as follows:

- Privileges that enable system-wide operations; for example, `CREATE SESSION`, `CREATE TABLESPACE`
- Privileges that enable management of objects in a user's own schema; for example, `CREATE TABLE`
- Privileges that enable management of objects in any schema; for example, `CREATE ANY TABLE`

Privileges can be controlled with the DDL commands `GRANT` and `REVOKE`, which add and revoke system privileges to the user or to a role. For more information on roles, refer to the "Managing Roles" lesson.

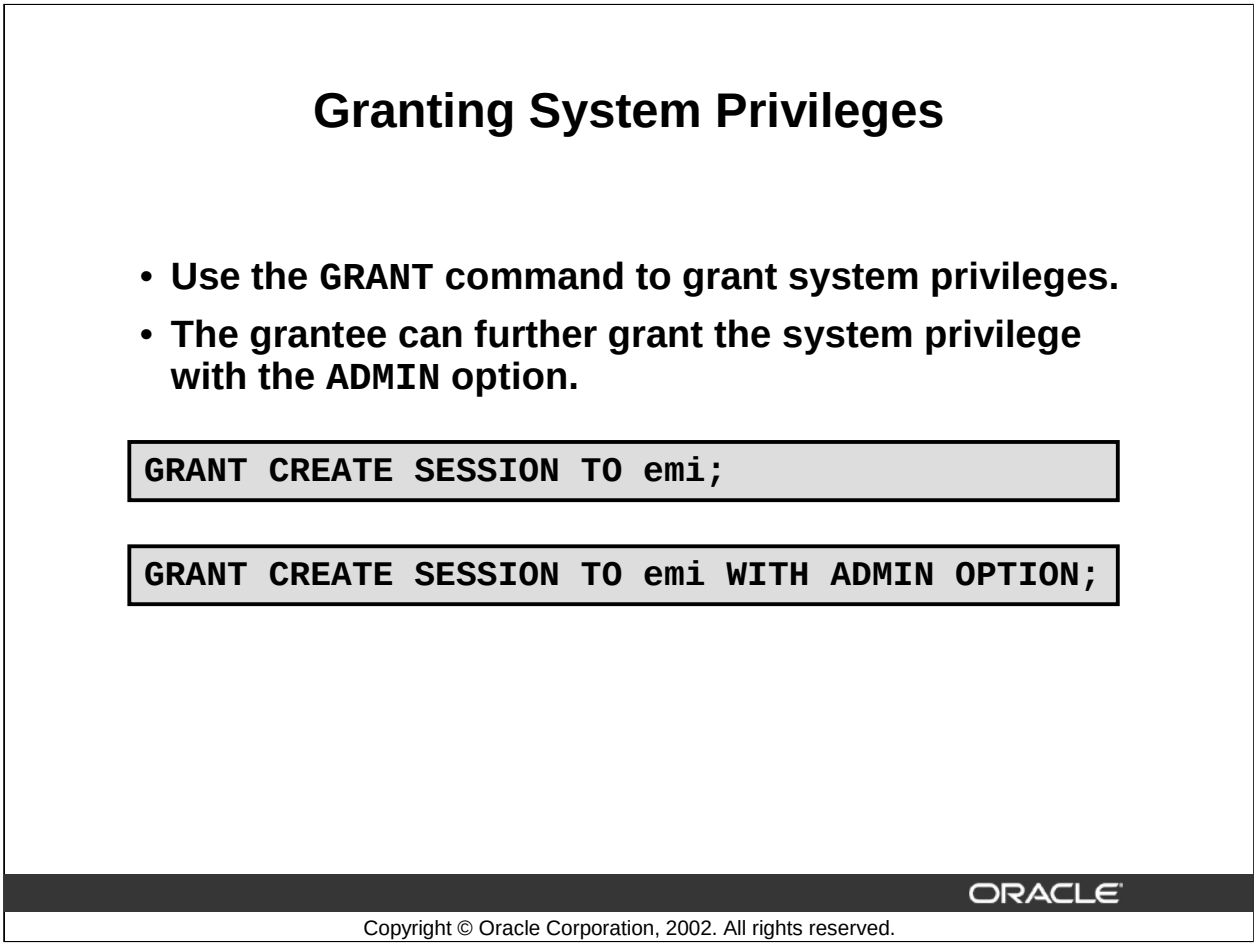
System Privileges: Examples

Category	Examples
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX
TABLE	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE SELECT ANY TABLE UPDATE ANY TABLE DELETE ANY TABLE
SESSION	CREATE SESSION ALTER SESSION RESTRICTED SESSION
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE

ORACLE

System Privileges: Examples

- There is no `CREATE INDEX` privilege.
- `CREATE TABLE` includes the `CREATE INDEX` and the `ANALYZE` commands. The user must have a quota for the tablespace or must have been granted `UNLIMITED TABLESPACE`.
- Privileges such as `CREATE TABLE`, `CREATE PROCEDURE`, or `CREATE CLUSTER` include dropping these objects.
- `UNLIMITED TABLESPACE` cannot be granted to a role.
- The `DROP ANY TABLE` privilege is necessary to truncate a table in another schema.



Granting System Privileges

Use the SQL statement GRANT to grant system privileges to users.

The grantee can further grant the system privilege to other users with the ADMIN option. Exercise caution when granting system privileges with the ADMIN option. Such privileges are usually reserved for security administrators and are rarely granted to other users.

```
GRANT {system_privilege|role}
     [, {system_privilege|role} ]...
    TO {user|role|PUBLIC}
     [, {user|role|PUBLIC} ]...
    [WITH ADMIN OPTION]
```

where:

system_privilege: Specifies the system privilege to be granted

role: Specifies the role name to be granted

PUBLIC: Grants system privilege to all users

Granting System Privileges (continued)

WITH ADMIN OPTION: Enables the grantee to further grant the privilege or role to other users or roles

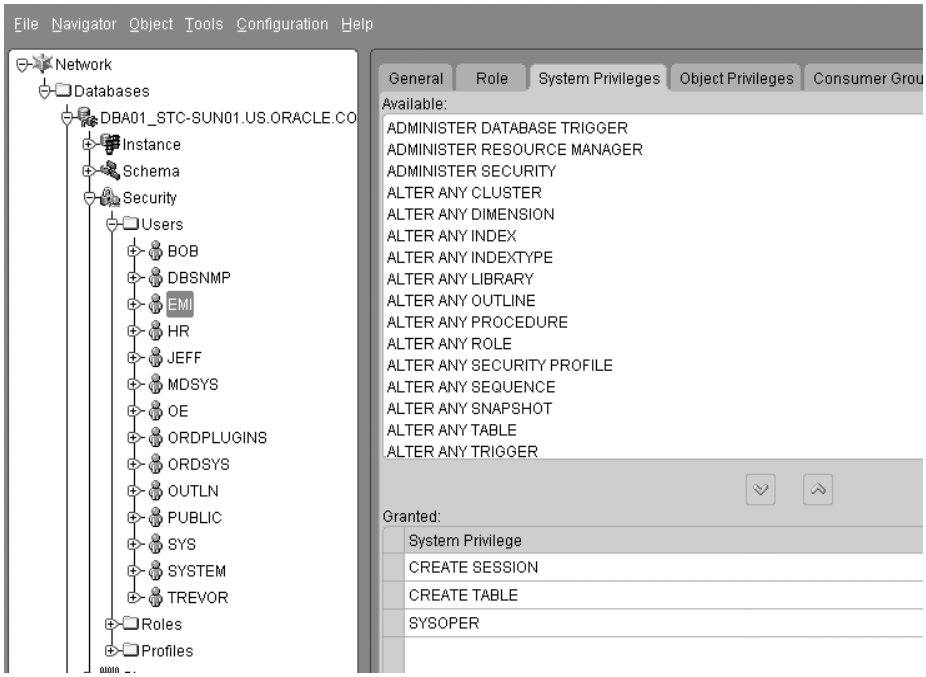
GRANT ANY OBJECT PRIVILEGE: This system privilege allows users to grant and revoke any object privilege on behalf of the object owner. This provides a convenient means for database and application administrators to grant access to objects in any schema without requiring that they connect to the schema. This eliminates the need to maintain login credentials for schema owners so that they can grant access to objects, and it reduces the number of connections required during configuration. This system privilege is part of the DBA role and is thus granted (with the ADMIN OPTION) to any user connecting as SYSDBA. As with other system privileges, the GRANT ANY OBJECT PRIVILEGE system privilege can only be granted by a user who possesses the ADMIN OPTION. When you exercise the GRANT ANY OBJECT PRIVILEGE system privilege to grant an object privilege to a user, if you already possess the object privilege with the GRANT OPTION, then the grant is performed in the usual way. In this case, you become the grantor of the grant. If you do not possess the object privilege, then the object owner is shown as the grantor, even though you, with the GRANT ANY OBJECT PRIVILEGE system privilege, actually performed the grant.

Granting System Privileges (continued)

Using Oracle Enterprise Manger to Grant System Privilege

From the OEM Console:

1. Navigate to Security > Users.
2. Select the user who will be granted the privilege.
3. Select the System tabbed page on the detail side of the console.
4. Select the system privileges that you want to grant. Optionally, select the Admin Option check box.
5. Click Apply.



SYSDBA and SYSOPER Privileges

Category	Examples
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE OPEN | MOUNT ALTER DATABASE BACKUP CONTROLFILE TO RECOVER DATABASE ALTER DATABASE ARCHIVELOG RESTRICTED SESSION
SYSDBA	SYSOPER PRIVILEGES WITH ADMIN OPTION CREATE DATABASE ALTER TABLESPACE BEGIN/END BACKUP RESTRICTED SESSION RECOVER DATABASE UNTIL

ORACLE

SYSDBA and SYSOPER Privileges

Only database administrators should have the capability to connect to a database with administrator privileges. Connecting as SYSDBA provides a user with unrestricted privileges to perform any operation on a database or on the objects within a database.

System Privilege Restrictions

The O7_DICTIONARY_ACCESSIBILITY parameter:

- **Controls restrictions on SYSTEM privileges**
- **If set to TRUE, allows access to objects in SYS schema**
- **The default is FALSE: ensures that system privileges that allow access to any schema do not allow access to SYS schema**

System Privilege Restrictions

The dictionary protection mechanism in Oracle9*i* prevents unauthorized users from accessing dictionary objects.

Access to dictionary objects is restricted to the roles SYSDBA and SYSOPER. System privileges that provide access to objects in other schemas do not give you access to dictionary objects. For example, the SELECT ANY TABLE privilege allows you to access views and tables in other schemas, but does not enable you to select dictionary objects (base tables, views, packages, and synonyms).

If the parameter is set to True, access to objects in SYS schema is allowed (Oracle7 behavior). If this parameter is set to False, SYSTEM privileges that allow access to objects in other schemas do not allow access to objects in the dictionary schema.

For example, if O7_DICTIONARY_ACCESSIBILITY=FALSE, then the SELECT ANY TABLE statement will allow access to views or tables in any schema except the SYS schema (for example, dictionaries could not be accessed). The EXECUTE ANY PROCEDURE system privilege will allow access on the procedures in any other schema except in the SYS schema.

Revoking System Privileges

- **Use the `REVOKE` command to remove a system privilege from a user.**
- **Users with `ADMIN OPTION` for system privilege can revoke system privileges.**
- **Only privileges granted with a `GRANT` command can be revoked.**

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

ORACLE

Revoking System Privileges

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

```
REVOKE {system_privilege|role}
 [, {system_privilege|role} ]...
FROM {user|role|PUBLIC}
 [, {user|role|PUBLIC} ]...
```

Note:

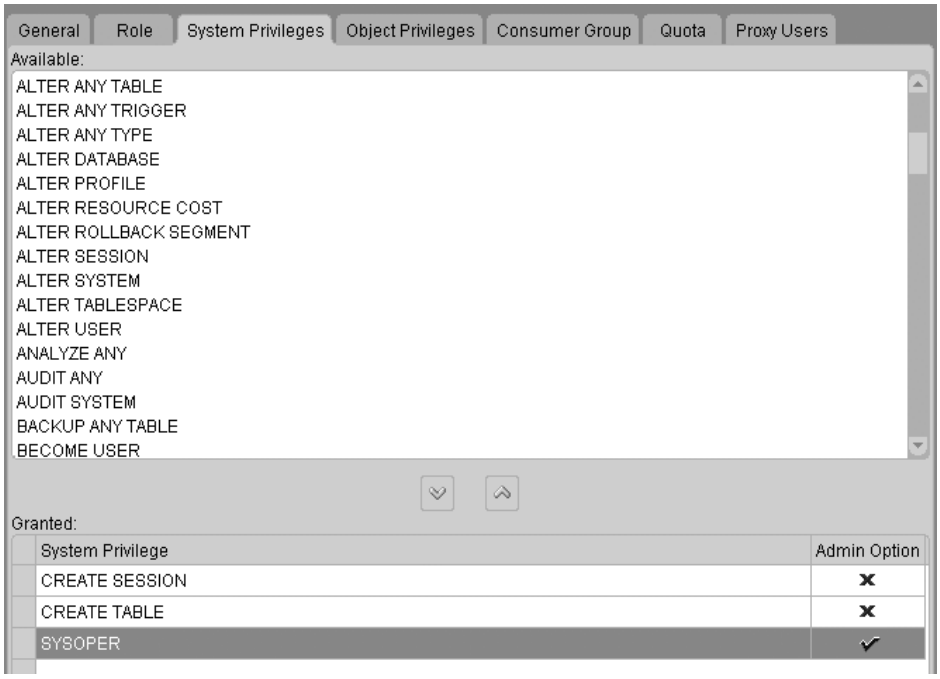
- The `REVOKE` command can only revoke privileges that have been granted directly with a `GRANT` command.
- Revoking system privileges may have an effect on some dependent objects. For example, if `SELECT ANY TABLE` is granted to a user, and that user has created procedures or views that use a table in some other schema, then revoking the privilege invalidates those procedures or views.

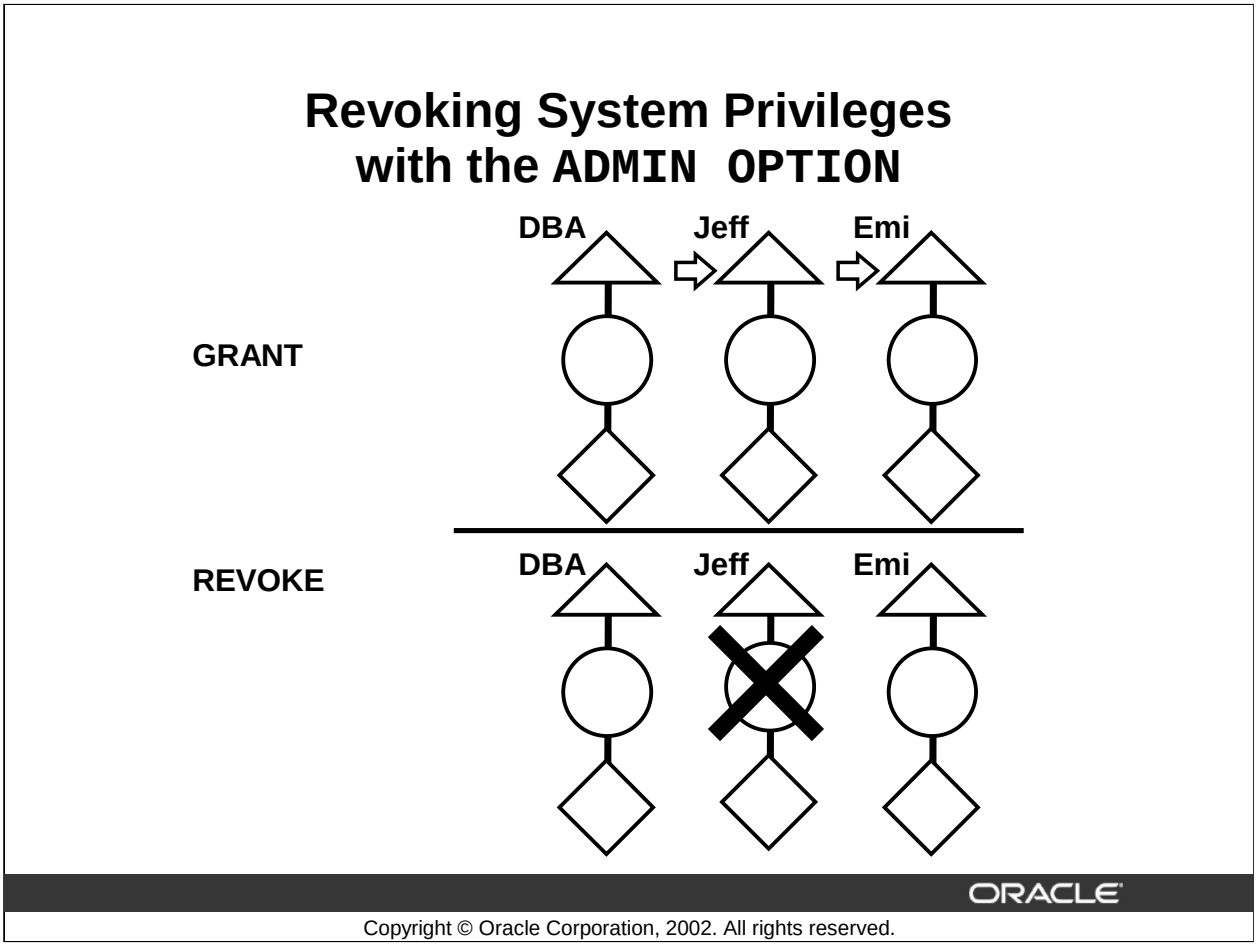
Revoking System Privileges (continued)

Using Oracle Enterprise Manager to Revoke System Privileges

From the OEM Console:

1. Navigate to Security > Users.
2. Select the user for whom the privilege is to be revoked.
3. Select the system privilege that is to be revoked from the System Privileges tabbed page.
4. Click the up arrow to remove the privilege from the Granted area.
5. Click Apply.





Revoking System Privileges (continued)

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

Read through the following steps that illustrate this.

Scenario

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

The result

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

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

Object Privileges

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

ORACLE

Object Privileges

An object privilege is a privilege or right to perform a particular action on a specific table, view, sequence, procedure, function, or package. Each object has a particular set of grantable privileges. The table above lists the privileges for various objects. Note that the only privileges that apply to a sequence are SELECT and ALTER. UPDATE, REFERENCES, and INSERT can be restricted by specifying a subset of updatable columns. SELECT can be restricted by creating a view with a subset of columns and by granting the SELECT privilege on the view. A grant on a synonym is converted to a grant on the base table that is referenced by the synonym.

Note: This slide does not provide a complete list of object privileges.

Granting Object Privileges

- **Use the `GRANT` command to grant object privileges.**
- **Grant must be in grantor's schema or grantor must have `GRANT OPTION`.**

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

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

Granting Object Privileges

```
GRANT { object_privilege [(column_list)]
     [, object_privilege [(column_list)] ]...
     |ALL [PRIVILEGES]}
ON  [schema.]object
TO  {user|role|PUBLIC}[, {user|role|PUBLIC} ]...
    [WITH GRANT OPTION]
```

where:

`object_privilege`: Specifies the object privilege to be granted

`column_list`: Specifies a table or view column. (This can be specified only when granting the `INSERT`, `REFERENCES`, or `UPDATE` privileges.)

`ALL`: Grants all privileges for the object that have been granted `WITH GRANT OPTION`

`ON object`: Identifies the object on which the privileges are to be granted

`WITH GRANT OPTION`: Enables the grantee to grant object privileges to other users or roles

Granting Object Privileges (continued)

The GRANT statement is used to grant object privileges to roles and users. To grant an object privilege, one of the following conditions must be met:

- You own the object specified.
- You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.
- The WITH GRANT OPTION clause was specified when you were granted the object privilege by its owner.

Note: System privileges and roles cannot be granted along with object privileges in the same GRANT statement.

WITH GRANT OPTION clause

Specify WITH GRANT OPTION to enable the grantee to grant the object privileges to other users and roles. The user whose schema contains an object is automatically granted all associated object privileges with the GRANT OPTION. This special privilege allows the grantee several expanded privileges:

- The grantee can grant the object privilege to any users in the database, with or without the GRANT OPTION, or to any role in the database.
- If both of the following are true, the grantee can create views on the table and grant the corresponding privileges on the views to any user or role in the database:
  - The grantee receives object privileges for the table with the GRANT OPTION.
  - The grantee has the CREATE VIEW or CREATE ANY VIEW system privilege.

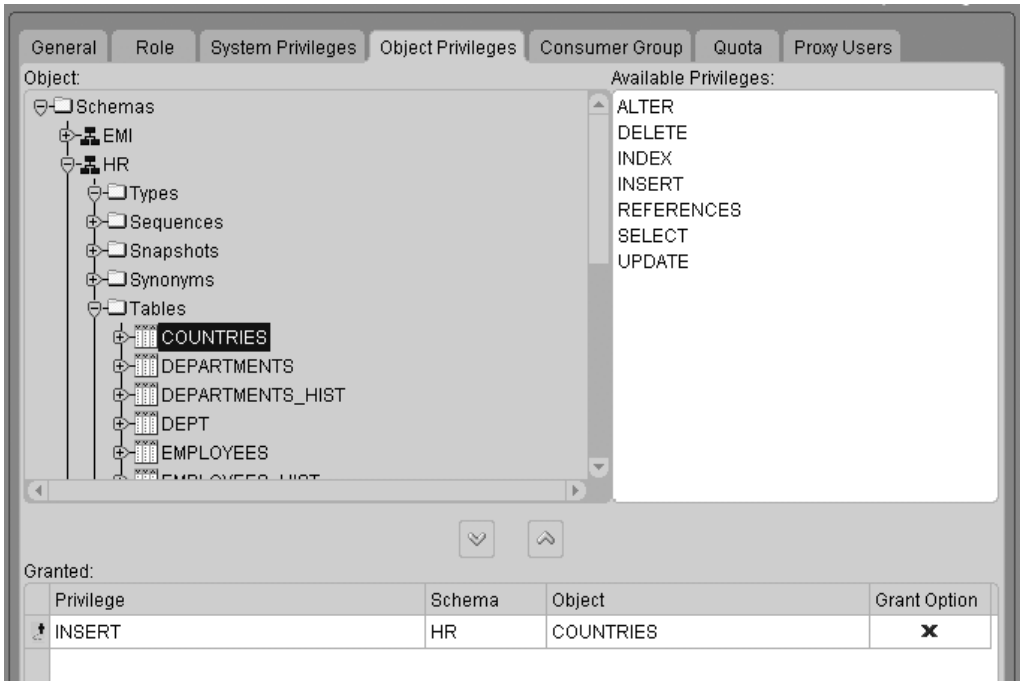
The GRANT OPTION is not valid when granting an object privilege to a role.

Granting Object Privileges (continued)

Using Oracle Enterprise Manager to Grant Object Privileges

From the OEM Console:

1. Navigate to Security > Users.
2. Select the user who will be granted the privilege.
3. Select the Object tabbed page on the detail side of the console.
4. Expand the schema and object folders for which the object privilege is being granted.
5. Select the privilege to be granted from the Available Privileges list and click the down arrow.
6. Optionally, select the Grant Option check box.
7. Click Apply.



Revoking Object Privileges

- **Use the REVOKE command to revoke object privileges.**
- **User revoking the privilege must be the original grantor of the object privilege being revoked.**

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

Revoking Object Privileges

The REVOKE statement is used to revoke object privileges. To revoke an object privilege, the revoker must be the original grantor of the object privilege being revoked.

Use the following command to revoke an object privilege:

```
REVOKE { object_privilege
       [, object_privilege ]...
       | ALL [PRIVILEGES] }
ON  [schema.]object
FROM {user|role|PUBLIC}
       [, {user|role|PUBLIC} ]...
       [CASCADE CONSTRAINTS]
```

Revoking Object Privileges (continued)

where:

`object_privilege`: Specifies the object privilege to be revoked

`ALL`: Revokes all object privileges that are granted to the user

`ON`: Identifies the object on which the object privileges are revoked

`FROM`: Identifies users or roles from which the object privileges are revoked

`CASCADE CONSTRAINTS`: Drops any referential integrity constraints that the revoke has defined using `REFERENCES` or `ALL` privileges

Restriction

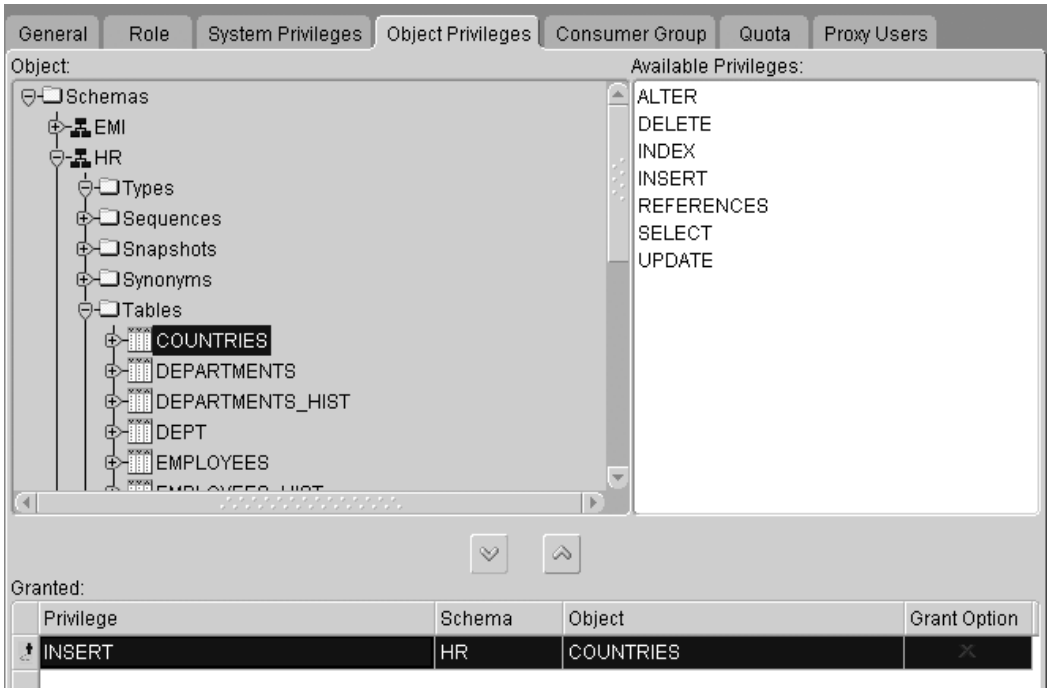
Grantors can revoke object privileges from only those users to whom they have granted privileges.

Revoking Object Privileges (continued)

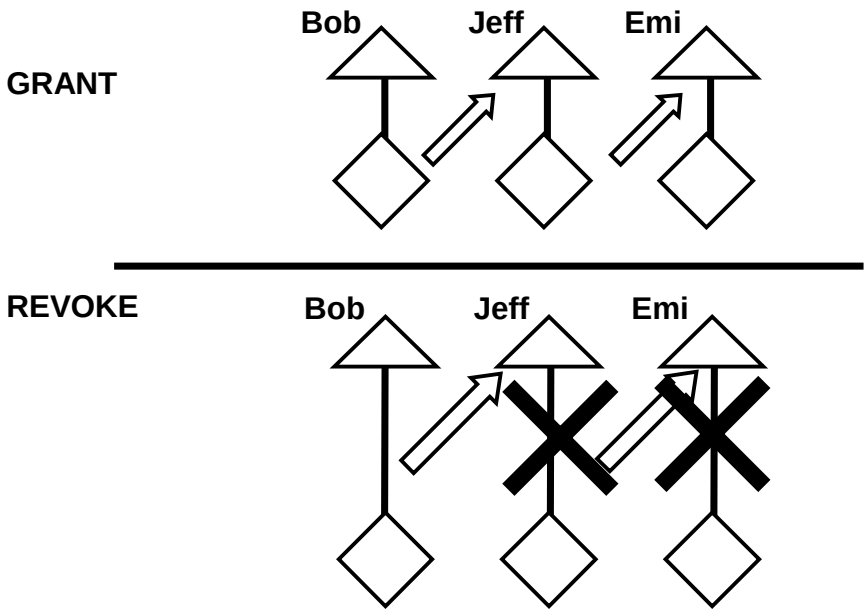
Using Oracle Enterprise Manager to Revoke Object Privileges

From the OEM Console:

1. Navigate to Security > Users.
2. Select the user for whom the privilege is to be revoked.
3. Click the Object tabbed page on the detail side of the console.
4. Select the object privilege that is to be revoked and click the up arrow.
5. Click Apply.



Revoking Object Privileges with GRANT OPTION



ORACLE

Revoking Object Privileges (continued)

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

Revoking object privileges will also cascade when given WITH GRANT OPTION.

Read through the following steps that illustrate this.

Scenario

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

Obtaining Privileges Information

Information about privileges can be obtained by querying the following views:

- **DBA_SYS_PRIVS**
- **SESSION_PRIVS**
- **DBA_TAB_PRIVS**
- **DBA_COL_PRIVS**

ORACLE

Obtaining Privileges Information

DBA_SYS_PRIVS: Lists system privileges granted to users and roles

SESSION_PRIVS: Lists the privileges that are currently available to the user

DBA_TAB_PRIVS: Lists all grants on all objects in the database

DBA_COL_PRIVS: Describes all object-column grants in the database

Summary

In this lesson, you should have learned how to:

- **Identify system and object privileges**
- **Grant and revoke privileges**
- **Obtain privilege information**

ORACLE

Practice 16 Overview

This practice covers the following topics:

- **Creating user and granting system privileges**
- **Granting object privileges to users**

ORACLE

Practice 16 Overview

Note: Practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 16: Managing Privileges

1. As user SYSTEM, create user Emi with the password of abcd12. Give her the capability to log on to the database and create schema objects. Assign her to the default tablespace DATA01, and the temporary tablespace TEMP. Make her quota on DATA01 1M.
2. **a** Run the script lab16_02a.sql to connect as Emi and create the tables CUSTOMERS1 and ORDERS1.
   b Connect as SYSTEM and copy the data from SYSTEM.CUSTOMERS to Emi's CUSTOMERS1 table. Verify that records have been inserted.
   c As user SYSTEM give Bob the ability to select from Emi's CUSTOMERS1 table. What happens?
3. Reconnect as Emi and give Bob the ability to select from Emi's CUSTOMERS1 table. Also, enable Bob to give the select capability to other users. As user SYSTEM, examine the data dictionary views that record these actions.
   Hint: Query the DBA_TAB_PRIVS data dictionary view to examine.
4. Create user Trevor identified by diamond1$ with the capability to log on to the database.
5. **a** As Bob, enable Trevor to access Emi's CUSTOMERS1 table.
   Note: A password has expired message will be received due to actions in step 8 in Lesson 15. Give Bob the new password aaron$1.
5. **b** As Emi, remove Bob's privilege to read Emi's CUSTOMERS1 table.
   c As Trevor, query Emi's CUSTOMERS1 table. What happens?
6. Enable Emi to start up and shut down the database without the ability to create a new database.

17

Managing Roles

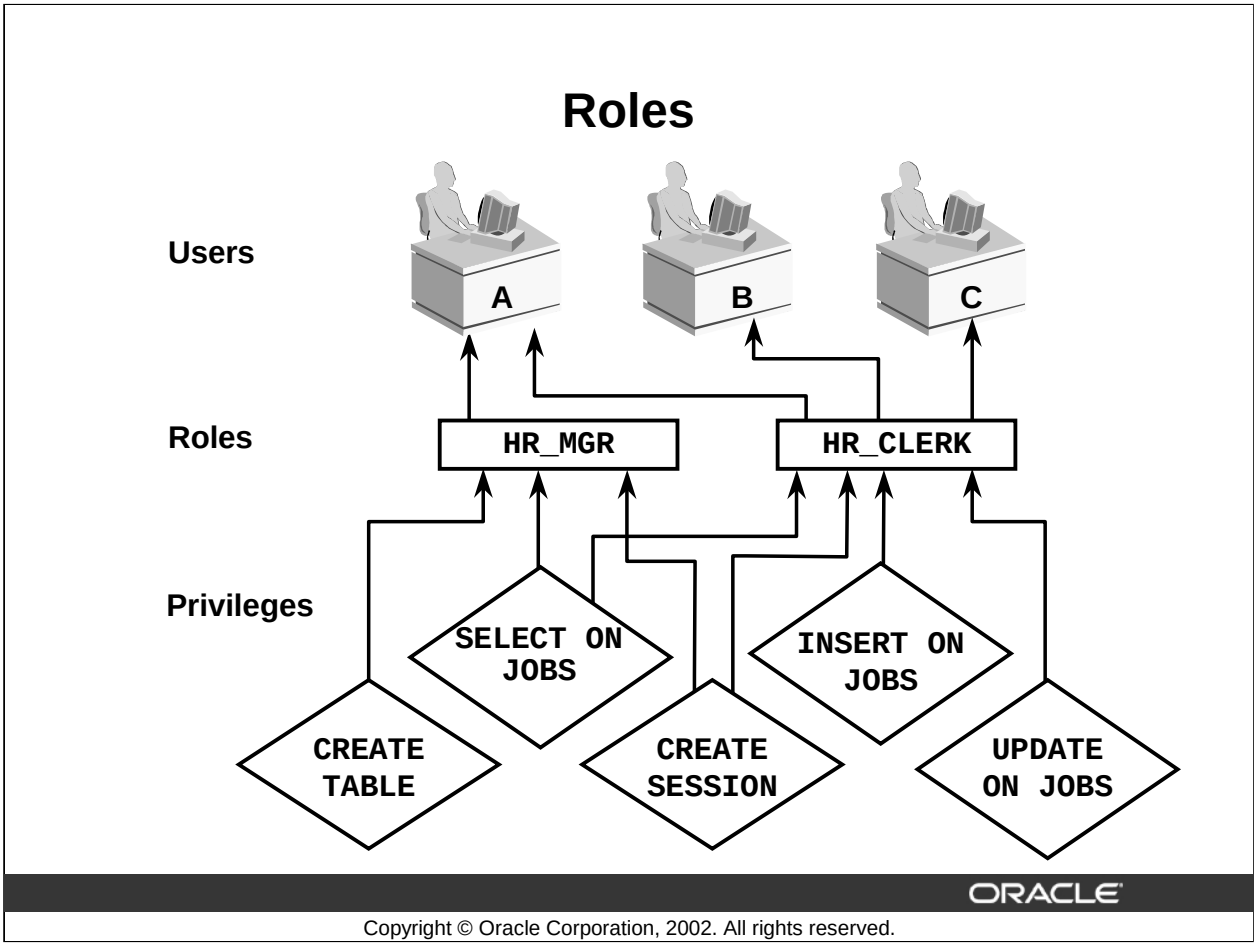
ORACLE

Objectives

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

- **Create and modify roles**
- **Control availability of roles**
- **Remove roles**
- **Use predefined roles**
- **Obtain role information**

ORACLE



Roles

Oracle provides for easy and controlled privilege management through roles. Roles are named groups of related privileges that are granted to users or to other roles. They are designed to ease the administration of privileges in the database.

Role characteristics

- Roles can be granted to and revoked from users with the same commands that are used to grant and revoke system privileges.
- Roles can be granted to any user or role. However, a role cannot be granted to itself and cannot be granted circularly.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Each role name must be unique among existing usernames and role names.
- Roles are not owned by anyone; and they are not in any schema.
- Roles have their descriptions stored in the data dictionary

Benefits of Roles

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

ORACLE

Benefits of Roles

Easier privilege management

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

Dynamic privilege management

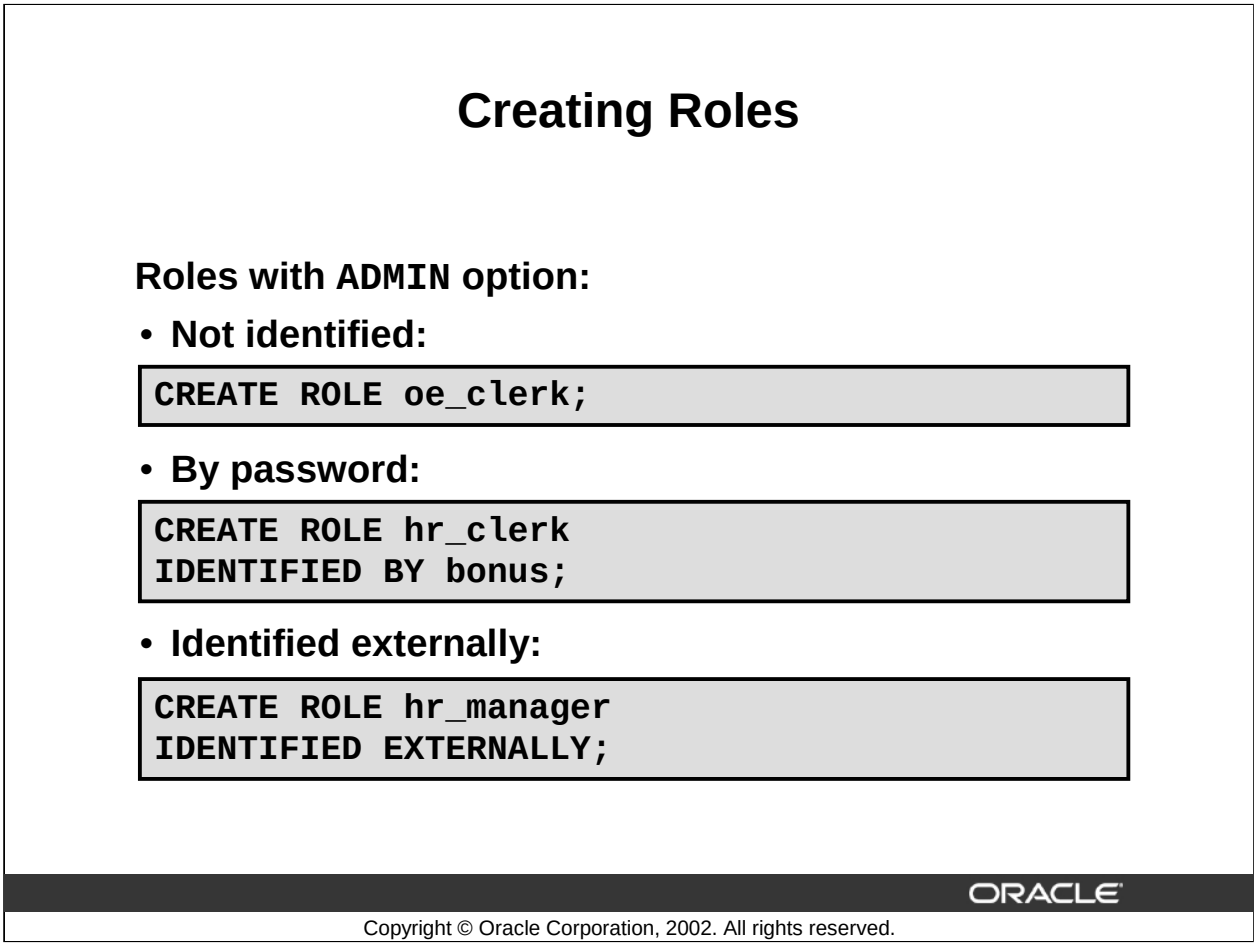
If the privileges associated with a role are modified, all the users who are granted the role acquire the modified privileges automatically and immediately.

Selective availability of privileges

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

Granting through the operating system

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



Creating Roles

Use the `CREATE ROLE` statement to create roles. You must have the `CREATE ROLE` system privilege to create roles. When you create a role that is not identified or is identified externally or by password, the role is granted with the `ADMIN` option.

Use the following command to create a role:

```
CREATE ROLE role [NOT IDENTIFIED | IDENTIFIED
  {BY password | EXTERNALLY | GLOBALLY | USING package}]
```

where:

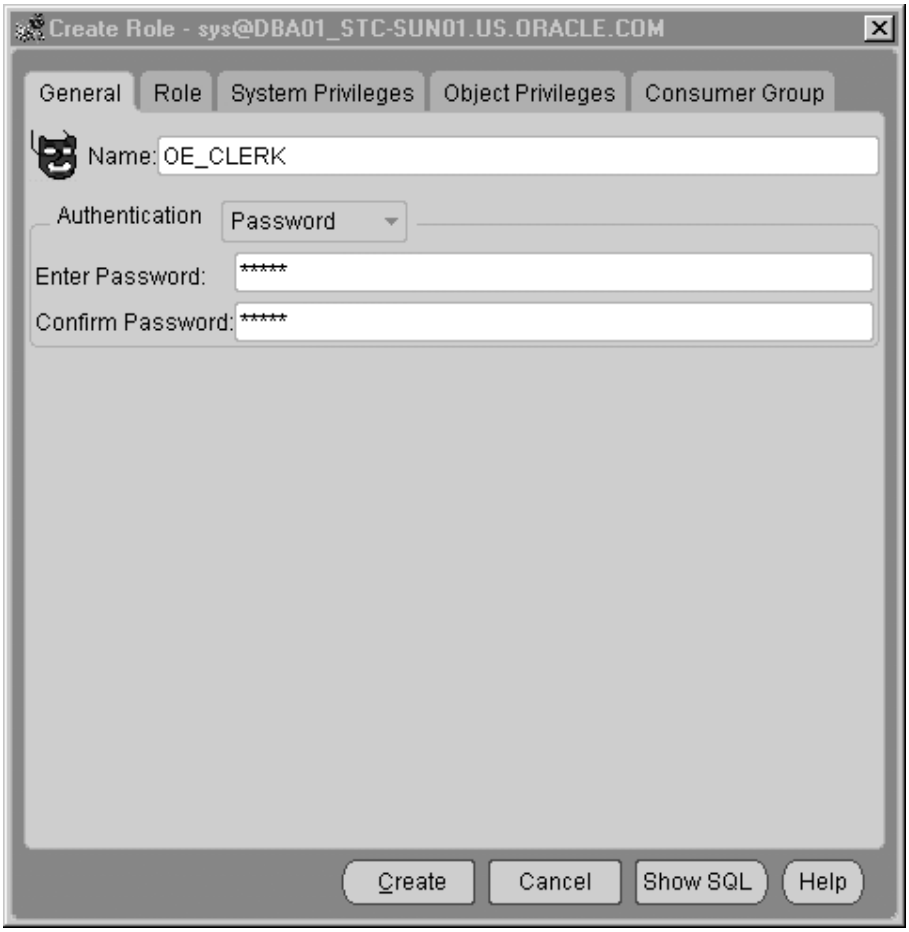
- `role`: Is the name of the role
- `NOT IDENTIFIED`: Indicates that no verification is required when enabling the role
- `IDENTIFIED`: Indicates that verification is required when enabling the role
- `BY password`: Provides the password that the user must specify when enabling the role
- `USING package`: Creates an application role, which is a role that can be enabled only by applications using an authorized package
- `EXTERNALLY`: Indicates that a user must be authorized by an external service (such as the operating system or a third-party service) before enabling the role
- `GLOBALLY`: Indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the `SET ROLE` statement, or at login.

Creating Roles (continued)

Using Oracle Enterprise Manager to Create a Role

From the OEM Console:

1. Navigate to Security > Roles.
2. Select Create from the right-mouse menu.
3. Complete the information for creating a role.
4. Click Create.



Predefined Roles

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

ORACLE

Predefined Roles

The roles listed are defined automatically for Oracle databases when you run database creation scripts. CONNECT, RESOURCE, and DBA roles are provided for backward compatibility to earlier versions of the Oracle server.

The EXP_FULL_DATABASE and IMP_FULL_DATABASE roles are provided for convenience in using the Import and Export utilities.

The roles DELETE_CATALOG_ROLE, EXECUTE_CATALOG_ROLE, and SELECT_CATALOG_ROLE are provided for accessing data dictionary views and packages. These roles can be granted to users who do not have the DBA role but who require access to the views and tables in the data dictionary.

Other special roles

The Oracle server also creates other roles that authorize you to administer the database. On many operating systems, these roles are called OSOPER and OSDBA. Their names may be different on your operating system.

Other roles are defined by SQL scripts provided with the database. For example, AQ_ADMINISTRATOR_ROLE provides privileges to administer advanced queuing. AQ_USER_ROLE is obsolete but is kept mainly for release 8.0 compatibility.

Modifying Roles

- **ALTER ROLE modifies the authentication method.**
- **Modifying roles requires the ADMIN option or ALTER ANY ROLE privilege.**

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

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

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

ORACLE

Modifying Roles

You can modify a role only to change its authentication method. You must have either been granted the role with the ADMIN option or have the ALTER ANY ROLE system privilege.

Use the following command to modify a role:

```
ALTER ROLE role {NOT IDENTIFIED | IDENTIFIED
 {BY password |USING package| EXTERNALLY | GLOBALLY }};
```

where:

- role: Is the name of the role
- NOT IDENTIFIED: Indicates that no verification is required when enabling the role
- IDENTIFIED: Indicates that verification is required when enabling the role
- BY password: Provides the password used when enabling the role
- EXTERNALLY: Indicates that a user must be authorized by an external service (such as the operating system or a third-party service) before enabling the role
- GLOBALLY: Indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the SET ROLE statement, or at login

Modifying Roles (continued)

Using Oracle Enterprise Manager to Modify a Role

From the OEM Console:

1. Navigate to Security > Roles.
2. Select Role to be modified.
3. Select View/Edit Details from the right-mouse menu.
4. Make the modifications.
5. Click OK.

Assigning Roles

Use the `GRANT` command to assign a role.

```
GRANT oe_clerk TO scott;
```

```
GRANT hr_clerk TO hr_manager;
```

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

Assigning Roles

To grant a role to a user, use the same syntax command that you used to grant a system privilege to a user:

```
GRANT role [, role ]...
TO {user|role|PUBLIC}
 [, {user|role|PUBLIC} ]...
[WITH ADMIN OPTION]
```

where:

- `role`: Is a collection of roles to be granted
- `PUBLIC`: Grants the role to all users
- `WITH ADMIN OPTION`: Enables the grantee to grant the role to other users or roles. (If you grant a role with this option, then the grantee can grant and revoke the role from other users and alter or drop the role.)

Assigning Roles (continued)

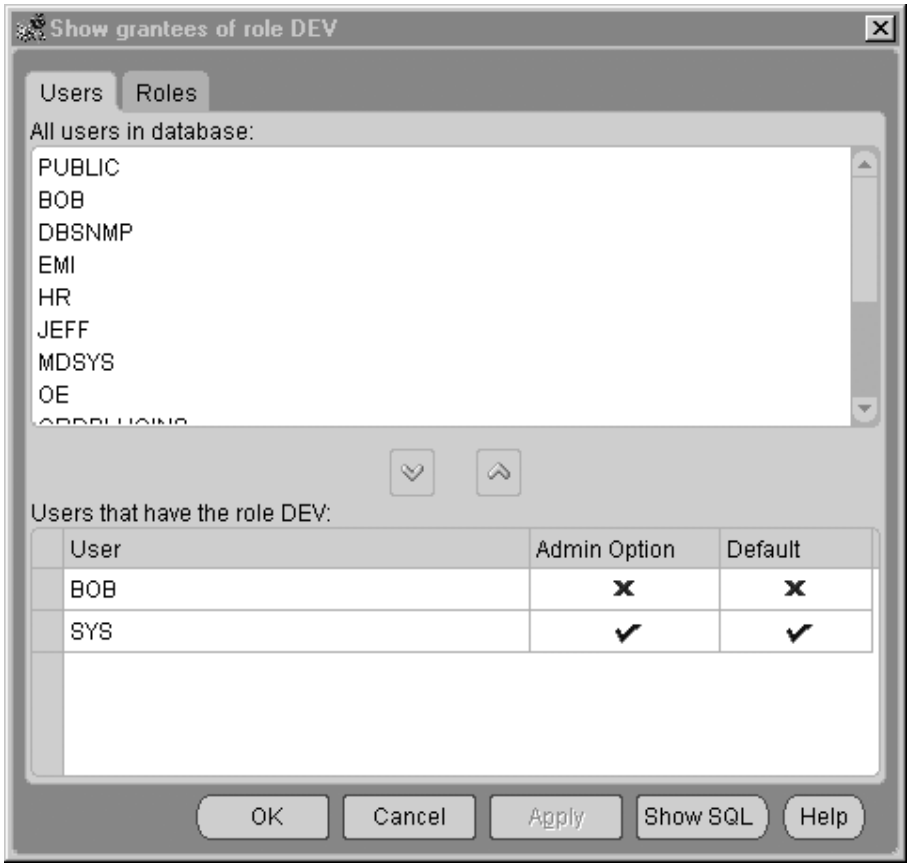
The user who creates a role is implicitly assigned the role with `ADMIN OPTION`. A user who has not been granted a role with `ADMIN OPTION` requires the `GRANT ANY ROLE` system privilege to grant and revoke roles to and from others.

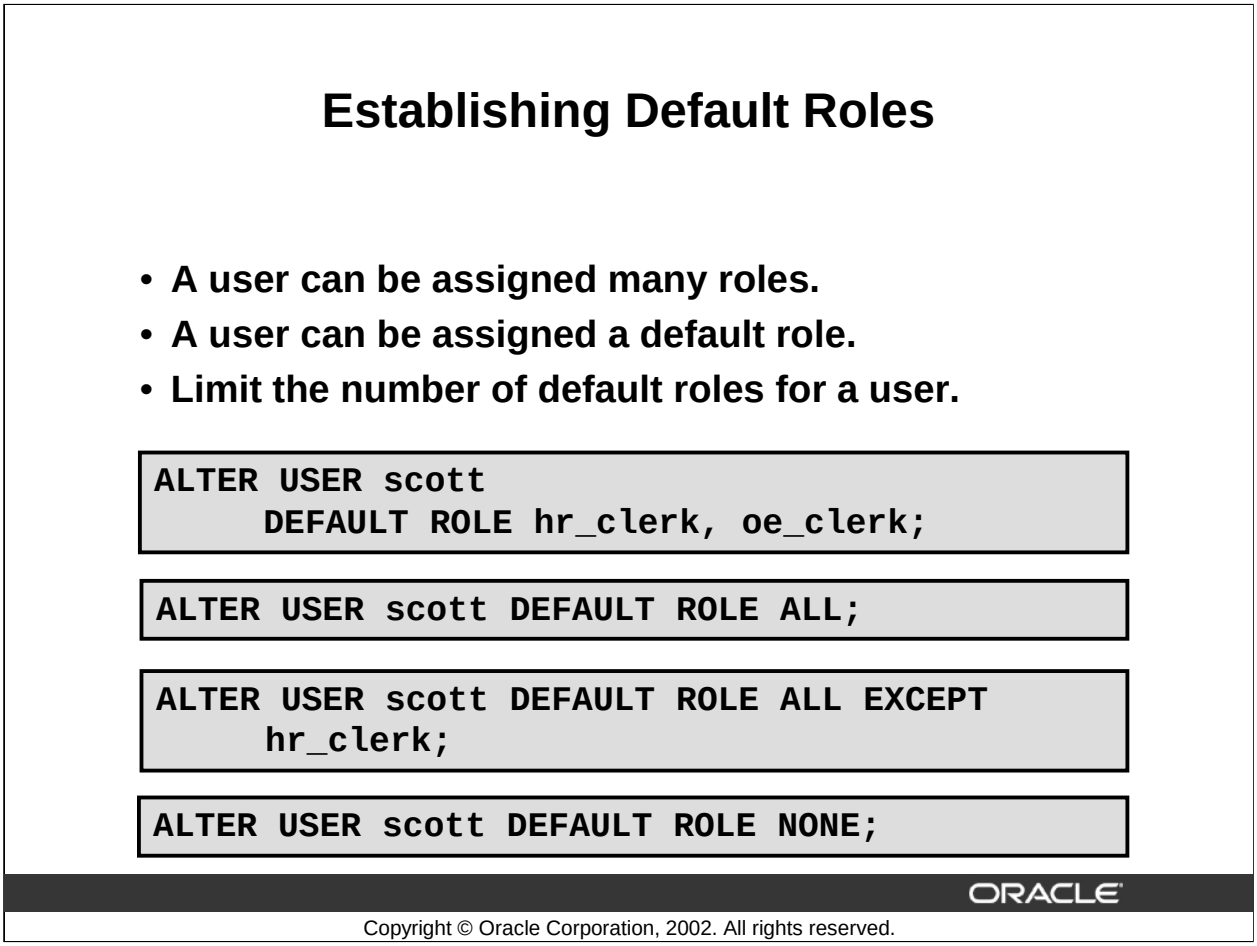
Note: The maximum number of database roles that users can enable is set by the `MAX_ENABLED_ROLES` initialization parameter.

Assigning Roles (continued)

Using Oracle Enterprise Manager to Assign a Role

1. Navigate to Security > Roles.
2. Select Role to be assigned.
3. Select Show Grantees from the right-mouse menu.
4. Select the user to be assigned the role from the Users tabbed page.
5. Click the down arrow to move the user into the “Users that have the role” window.
6. Click OK.





Establishing Default Roles

A user can be assigned many roles. A default role is a subset of these roles that is automatically enabled when the user logs on. By default, all the roles assigned to a user are enabled at logon without the need of a password. Limit the default roles for a user with the `ALTER USER` command.

The `DEFAULT ROLE` clause applies only to roles that have been granted directly to the user with a `GRANT` statement. The `DEFAULT ROLE` clause cannot be used to enable the following:

- Roles not granted to the user
- Roles granted through other roles
- Roles managed by an external service (such as the operating system)

Use the following syntax to assign default roles to a user:

```
ALTER USER user DEFAULT ROLE
{role [,role]... | ALL [EXCEPT role [,role]... ] | NONE}
```

where:

- `user`: Is the name of the user who is granted the roles
- `role`: Is the role to be made the default role for the user

Establishing Default Roles (continued)

- ALL: Makes all of the roles granted to the user default roles, except those listed in the EXCEPT clause. (This is the default.)
- EXCEPT: Indicates that the following roles should not be included in the default roles
- NONE: Makes none of the roles that are granted to the user default roles. (The only privileges that the user has at login are those privileges that are assigned directly to the user.)

Because the roles must be granted before they can be made defaults, you cannot set default roles with the CREATE USER command.

Application Roles

- **Application roles can be enabled only by authorized PL/SQL packages.**
- **The `USING` package clause creates an application role.**

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


Application Roles

The `USING` package clause in the `CREATE ROLE` statement creates an application role. An application role can be enabled only by applications by using an authorized PL/SQL package. Application developers do not need to secure a role by embedding passwords inside applications. Instead, they can create an application role and specify which PL/SQL package is authorized to enable the role.

```
SQL> CREATE ROLE admin_role
  2  IDENTIFIED USING hr.employees;
```

In this example, `admin_role` is an application role and the role can be enabled only by modules that are defined inside the `hr.employee` PL/SQL package.

Enabling and Disabling Roles

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

ORACLE

Enabling and Disabling Roles

Enable or disable roles to temporarily activate and deactivate the privileges associated with the roles. To enable a role, the role must first be granted to the user.

When a role is enabled, the user can use the privileges granted to that role. If a role is disabled, the user cannot use the privileges associated with that role unless those privileges are granted directly to the user or to another role enabled for that user. Roles are enabled for a session. At the next session, the user's active roles revert to default roles.

Specifying roles to be enabled

The `SET ROLE` command and the `DBMS_SESSION.SET_ROLE` procedure enable all of the roles included in the command and disable all other roles. Roles can be enabled from any tool or program that allows PL/SQL commands; however, a role cannot be enabled in a stored procedure.

You can use the `ALTER USER...DEFAULT ROLE` command to indicate which roles will be enabled for a user at login. All other roles are disabled.

A password may be required to enable a role. The password must be included in the `SET ROLE` command to enable the role. Default roles assigned to a user do not require a password; they are enabled at login, the same as a role without a password.

Enabling and Disabling Roles (continued)

Restrictions

A role cannot be enabled from a stored procedure, because this action may change the security domain (set of privileges) that allowed the procedure to be called in the first place. So, in PL/SQL, roles can be enabled and disabled in anonymous blocks and application procedures (for example, Oracle Forms procedures), but not in stored procedures.

If a stored procedure contains the command `SET ROLE`, the following error is generated at run time:

```
ORA-06565: cannot execute SET ROLE from within stored
           procedure
```

Enabling and Disabling Roles

```
SET ROLE hr_clerk;
```

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

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


Enabling and Disabling Roles

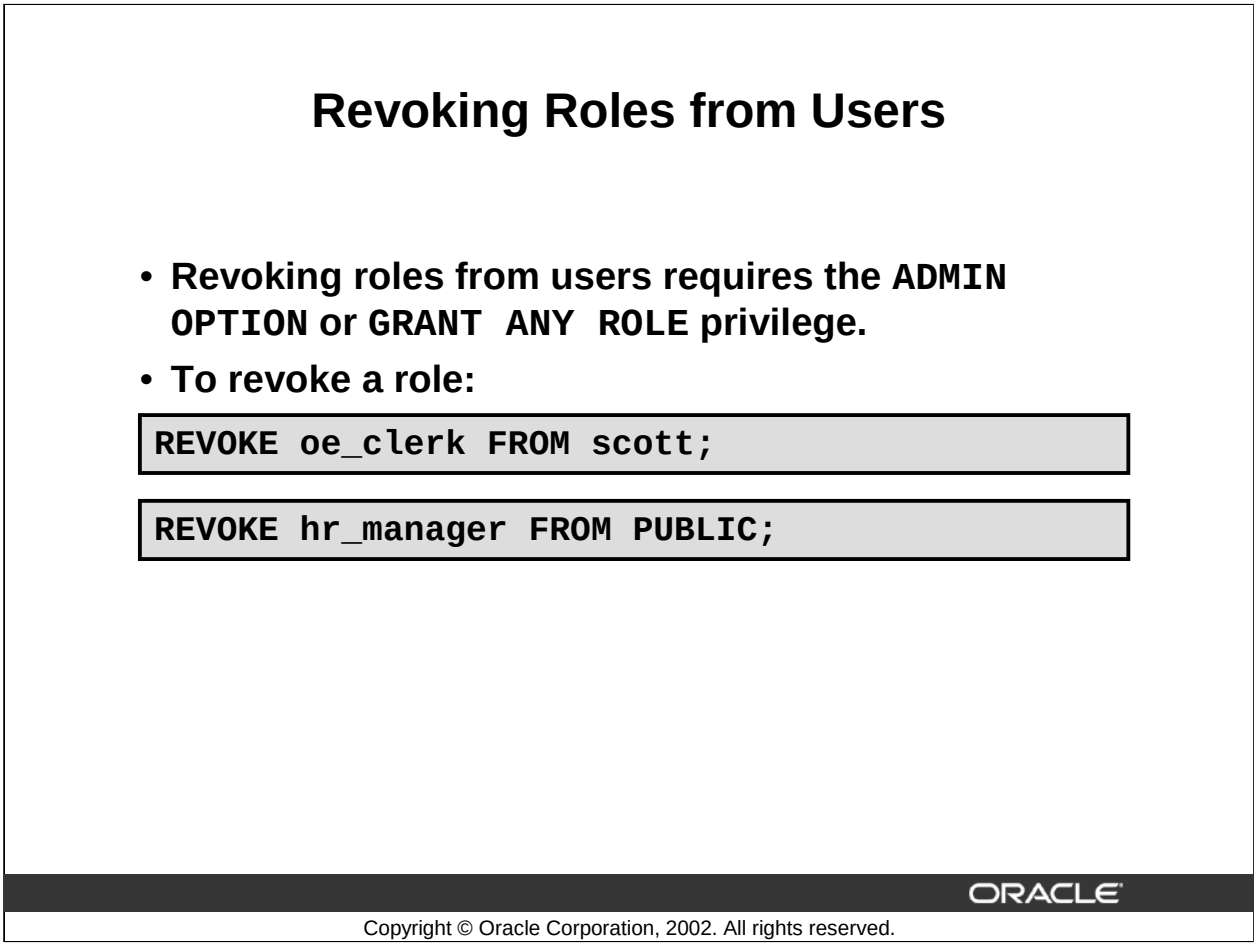
The SET ROLE command turns off any other roles granted to the user.

```
SET ROLE {role [ IDENTIFIED BY password ]
   [, role [ IDENTIFIED BY password ]]...
   | ALL [ EXCEPT role [, role ] ...]
   | NONE }
```

where:

- role: Is the name of the role
- IDENTIFIED BY password: Provides the password required when enabling the role
- ALL: Enables all roles that are granted to the current user, except those listed in the EXCEPT clause. (You cannot use this option to enable roles with passwords.)
- EXCEPT role: Does not enable these roles
- NONE: Disables all roles for the current session. (Only privileges granted directly to the user are active.)

The ALL option without the EXCEPT clause works only when every role that is enabled does not have a password.



Revoking Roles from Users

To revoke a role from a user, use the SQL statement REVOKE. Any user with the ADMIN option for a role can revoke the role from any other database user or role. Also, users with the GRANT ANY ROLE privilege can revoke any role.

```
REVOKE role [, role ]
   FROM {user|role|PUBLIC}
   [, {user|role|PUBLIC} ]
```

where

- role: Is the role to be revoked or the role from which roles are revoked
- user: Is the user from which the system privileges or roles are revoked
- PUBLIC: Revokes the privilege or role from all users

Revoking Roles from Users (continued)

Using Oracle Enterprise Manager to Revoke a Role from a User

From the OEM Console:

1. Navigate to Security > Users.
2. Highlight the user for whom a role is to be revoked.
3. Navigate to Roles Granted.
4. Select the role to be revoked.
5. Select Revoke from the right-mouse menu.
6. Select Yes to confirm revocation.

Removing Roles

- **Dropping a role:**
  - **Removes it from all users and roles it was granted**
  - **Removes it from the database**
- **Requires the `ADMIN OPTION` or `DROP ANY ROLE` privilege**
- **To drop a role:**

```
DROP ROLE hr_manager;
```

ORACLE

Removing Roles

To remove a role from the database, use the following syntax:

```
SQL> DROP ROLE role
```

When you drop a role, the Oracle server revokes it from all users and roles to whom it has been granted and removes it from the database.

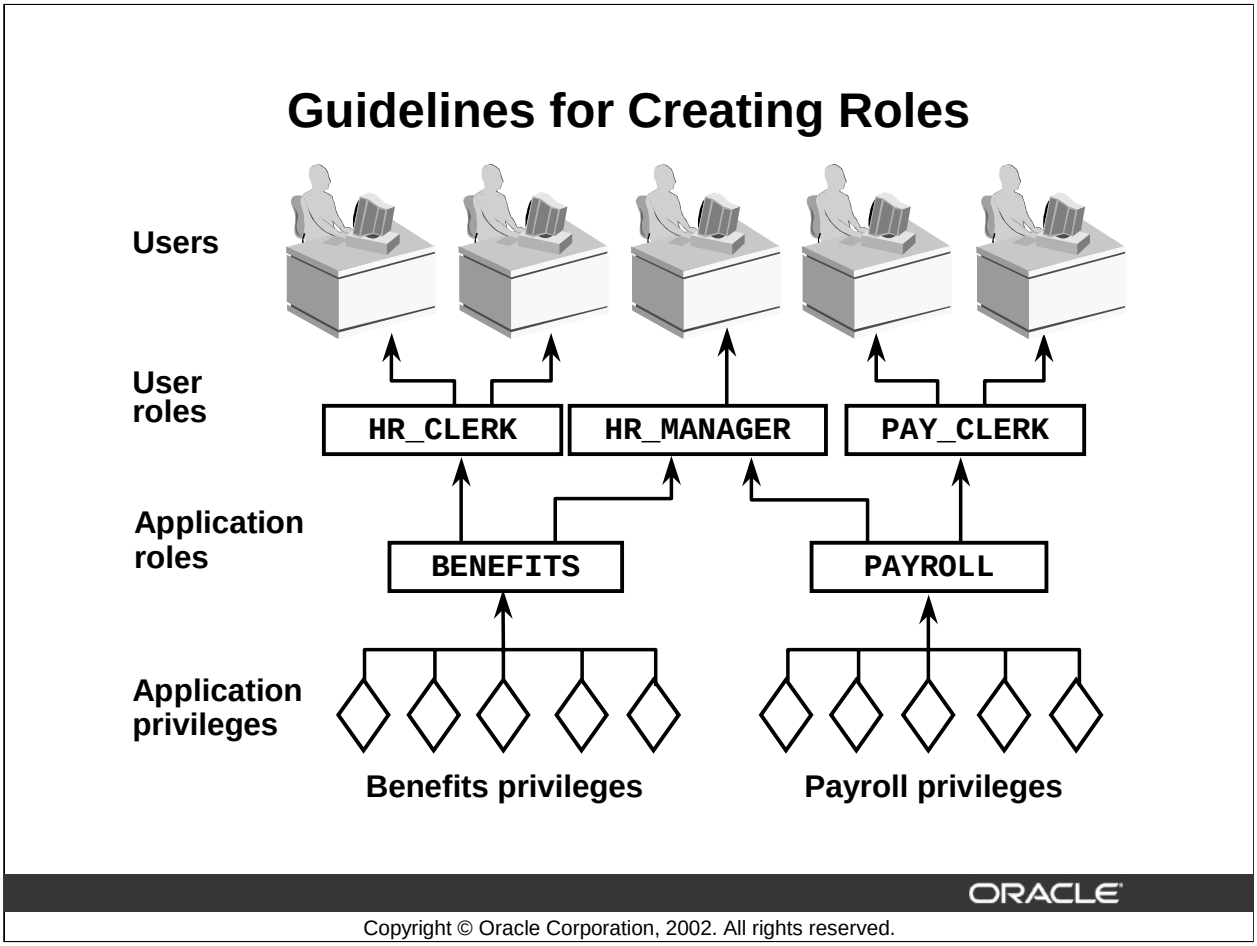
In order to drop the role, you must have been granted the role with `ADMIN OPTION` or have the `DROP ANY ROLE` system privilege.

Removing Roles (continued)

Using Oracle Enterprise Manager to Remove a Role

From the OEM Console:

1. Navigate to Security > Roles.
2. Select the role to be removed.
3. Select Remove from the right-mouse menu.
4. Verify the Role is not assigned.
5. Select Yes to confirm removal.

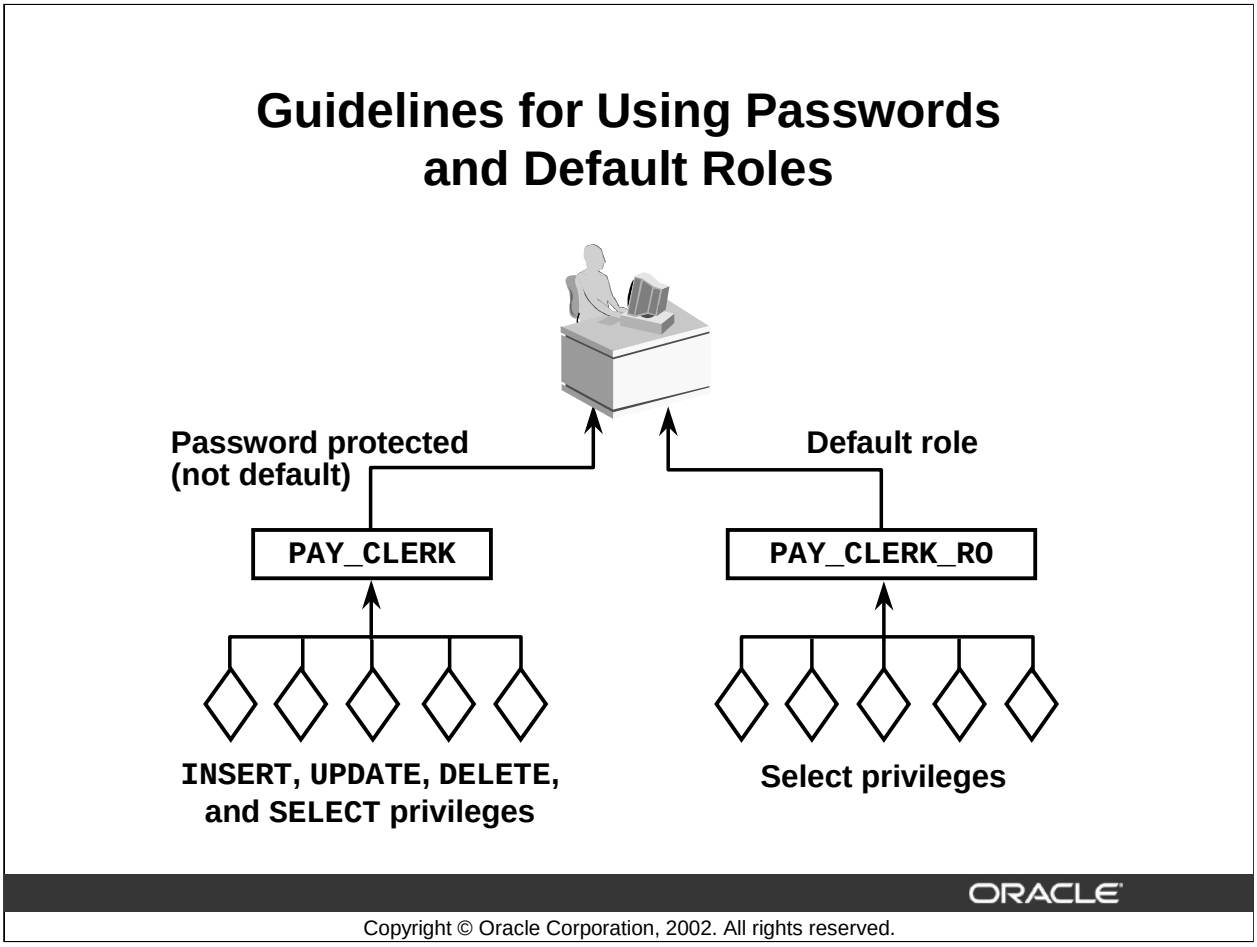


Guidelines for Creating Roles

Because a role includes the privileges that are necessary to perform a task, the role name is usually an application task or a job title. The example in the slide uses both application tasks and job titles for role names. Use the following steps to create, assign, and grant users roles:

1. Create a role for each application task. The name of the application role corresponds to a task in the application, such as PAYROLL.
2. Assign the privileges necessary to perform the task to the application role.
3. Create a role for each type of user. The name of the user role corresponds to a job title, such as PAY_CLERK.
4. Grant application roles to user's roles.
5. Grant user's roles to users.

If a modification to the application requires that new privileges are needed to perform the payroll task, then the DBA only needs to assign the new privileges to the PAYROLL application role. All of the users that are currently performing this task will receive the new privileges.



Guidelines for Using Passwords and Default Roles

Passwords provide an additional level of security when enabling a role. For example, the application might require a user to enter a password when enabling the PAY_CLERK role, because this role can be used to issue checks.

Passwords allow a role to be enabled only through an application. This technique is shown in the example in the slide.

- The DBA has granted the user two roles, PAY_CLERK and PAY_CLERK_RO.
- The PAY_CLERK has been granted all of the privileges that are necessary to perform the payroll clerk function.
- The PAY_CLERK_RO (RO for read only) has been granted only SELECT privileges on the tables required to perform the payroll clerk function.
- The user can log in to SQL*Plus to perform queries, but cannot modify any of the data, because the PAY_CLERK is not a default role, and the user does not know the password for PAY_CLERK.
- When the user logs in to the payroll application, it enables the PAY_CLERK by providing the password. It is coded in the program; the user is not prompted for it.

Obtaining Role Information

Information about roles can be obtained by querying the following views:

- **DBA_ROLES: All roles that exist in the database**
- **DBA_ROLE_PRIVS: Roles granted to users and roles**
- **ROLE_ROL_PRIVS: Roles that are granted to roles**
- **DBA_SYS_PRIVS: System privileges granted to users and roles**
- **ROLE_SYS_PRIVS: System privileges granted to roles**
- **ROLE_TAB_PRIVS: Object privileges granted to roles**
- **SESSION_ROLES: Roles that the user currently has enabled**

ORACLE

Obtaining Role Information

Many of the data dictionary views that contain information on privileges that are granted to users also contain information about whether the role requires a password.

```
SQL> SELECT role, password_required
  2   FROM dba_roles;
ROLE                           PASSWORD
------------------------------ -----------
CONNECT                        NO
RESOURCE                       NO
DBA                            NO
SELECT_CATALOG_ROLE            NO
EXECUTE_CATALOG_ROLE           NO
DELETE_CATALOG_ROLE            NO
IMP_FULL_DATABASE              NO
EXP_FULL_DATABASE              NO
SALES_CLERK                    YES
HR_CLERK                       EXTERNAL
```

Summary

In this lesson, you should have learned how to:

- **Create roles**
- **Assign privileges to roles**
- **Assign roles to users or roles**
- **Establish default roles**
- **Obtain role information**

ORACLE

Practice 17 Overview

This practice covers the following topics:

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

ORACLE

Practice 17 Overview

Note: Practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 17: Managing Roles

1. Examine the data dictionary view and list the system privileges of the RESOURCE role.
2. Create a role called DEV, which will enable a user assigned the role to create a table, create a view, and select from Emi's CUSTOMERS1 table.
3. a. Assign the RESOURCE and DEV roles to Bob, but make only the RESOURCE role automatically enabled when he logs on.
   b. Give Bob the ability to read all the data dictionary information.
4. Bob needs to check the undo segments that are currently used by the instance. Connect as Bob and list the undo segments used.
   Hint: Use SET ROLE SELECT_CATALOG_ROLE
5. As SYSTEM, try to create a CUST_VIEW view on Emi's CUSTOMERS1 table. What happens?
6. As user Emi, grant SELECT on CUSTOMERS1 to SYSTEM. As SYSTEM, create a CUST_VIEW view on Emi's CUSTOMERS1 table.

18

Auditing

ORACLE

Objectives

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

- **Outline auditing categories**
- **Enable auditing for an instance**
- **Outline auditing options**
- **Obtain audit information**

ORACLE

Auditing

- **Auditing is the monitoring of selected user database actions, and is used to:**
  - **Investigate suspicious database activity**
  - **Gather information about specific database activities**
- **Auditing can be performed by session or access**

ORACLE

Auditing

If an unauthorized user is deleting data, then the DBA might decide to audit all connections to the database and all successful and unsuccessful deletions from all tables in the database. The DBA can gather statistics about which tables are being updated, how many logical inputs/outputs (I/Os) are performed, and how many concurrent users connect at peak times.

Auditing Guidelines

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

ORACLE

Auditing Guidelines

Restrict auditing by first identifying the auditing requirements, and setting minimal auditing options that will cater to the requirements. Object auditing must be used where possible to reduce the number of entries generated. If statement or privilege auditing must be used, you can minimize audit generation by using the following settings:

- Specifying users to audit
- Auditing by session, and not by access
- Auditing either successes or failures, but not both

Note: Audit records can be written to either `SYS.AUD$` or the operating system's audit trail. The ability to use the operating system's audit trail is operating system dependent.

Monitoring the growth of the audit trail

If the audit trail becomes full, no more audit records can be inserted, and audited statements will not execute successfully. Errors are returned to all users who issue an audited statement. You must free some space in the audit trail before these statements can be executed.

Auditing Guidelines (continued)

Monitoring the growth of the audit trail (continued)

To ensure that the audit trail does not grow too rapidly, do the following:

- Enable auditing only when necessary.
- Be selective about which audit options are specified.
- Tightly control schema object auditing. Users can turn on auditing for the objects that they own.
- Grant the `AUDIT ANY` privilege sparingly because it also enables a user to turn on auditing.

Periodically remove audit records from the audit trail with the `DELETE` or `TRUNCATE` command. Audit files are located in `$ORACLE_HOME/rdbms/audit` directory.

Protecting the audit trail

You should protect the audit trail so that audit information cannot be added, modified, or deleted. Issue the command:

```
SQL> AUDIT delete ON sys.aud$ BY ACCESS;
```

to protect the audit trail from unauthorized deletions; only the DBA should have the `DELETE_CATALOG_ROLE` role.

Moving the audit trail out of the system tablespace

As new records are inserted into the database audit trail, the `AUD$` table can grow without limit. Although you should not drop the `AUD$` table, you can delete or truncate from it because the rows are for information only and are not necessary for the Oracle Instance to run. Because the `AUD$` table grows and then shrinks, it should be stored outside of the system tablespace.

To move `AUD$` to the `AUDIT_TAB` tablespace:

- Ensure that auditing is currently disabled.
- Enter the following command:
```
SQL> ALTER TABLE aud$ MOVE TABLESPACE AUDIT_TAB;
```
- Enter the following command:
```
SQL> CREATE INDEX i_aud1 ON aud$(sessionid, ses$tid)
 2   TABLESPACE AUDIT_IDX;
```
- Enable auditing for the instance.

Auditing Categories

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

ORACLE

Auditing Categories

Regardless of whether database auditing is enabled, Oracle always records some database operations into the operating system audit trail. These are:

- **Instance startup:** The audit record details the operating system user who is starting the instance, the user's terminal identifier, the date and time stamp, and whether database auditing was enabled or disabled.
- **Instance shutdown:** This details the operating system user who is shutting down the instance, the user's terminal identifier, and the date and time stamp.
- **Administrator privileges:** This details the operating system user who is connecting to Oracle with administrator privileges.

Database auditing

Database auditing monitors and records selected user database actions. Information about the event is stored in the audit trail.

The audit trail can be used to investigate suspicious activity. For example, if an unauthorized user is deleting data from tables, the DBA may decide to audit all connections to the database in conjunction with successful and unsuccessful deletions of rows from tables in the database.

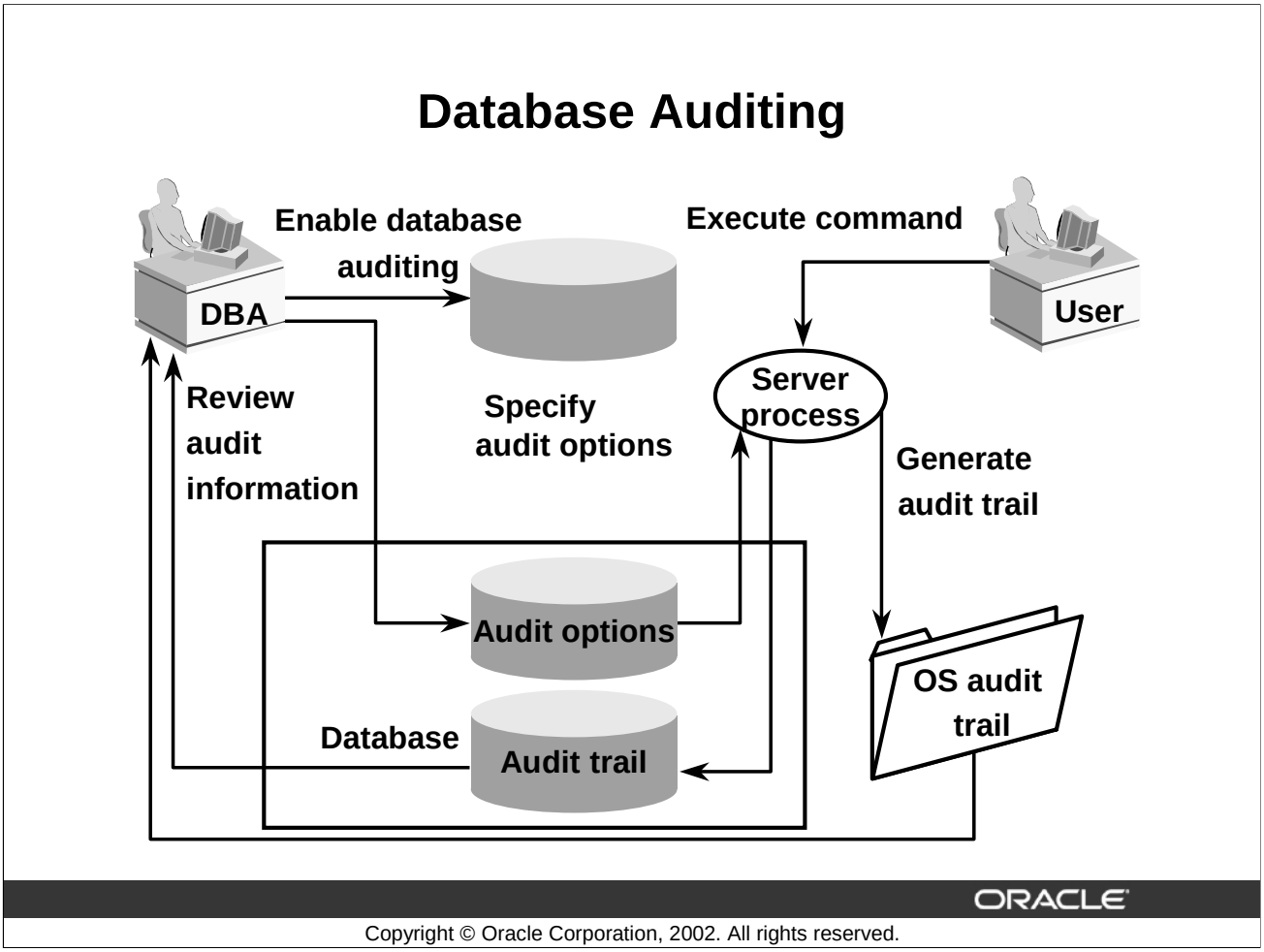
Auditing Categories (continued)

Database auditing (continued)

Auditing might also be used to monitor and gather data about specific database activities. For example, the DBA can gather statistics about which tables are being updated, how many logical I/Os are performed, and how many concurrent users connect at peak times.

Value-based auditing

Database auditing cannot record column values. If the changes to database columns must be tracked and column values must be stored for each change, then use application auditing. Application auditing can be done either through client code, stored procedures, or database triggers.



Database Auditing

Enabling and disabling database auditing

After you have decided what to audit, you set the `AUDIT_TRAIL` initialization parameter to enable auditing for the instance. This parameter indicates whether the audit trail is written to a database table or the operating system audit trail.

```
AUDIT_TRAIL = value
```

where value can be one of the following:

- `TRUE` or `DB`: Enables auditing and directs all audit records to the database audit trail (`SYS.AUD$`)
- `OS`: Enables auditing and directs all audit records to the operating system audit trail (if permitted on the operating system)
- `FALSE` or `NONE`: Disables auditing

Note: There is no default value.

Database Auditing (continued)

Audit records will not be written to the audit trail unless the DBA has set the AUDIT_TRAIL parameter to DB or OS. Although the SQL statements AUDIT and NOAUDIT can be used at any time, records will only be written to the audit trail if the DBA has set the AUDIT_TRAIL parameter in the initialization file.

Note: The *Installation and Configuration Guide* for your operating system provides information on writing audit records to the OS audit trail.

Specifying audit options

Next, you set specific auditing options using the AUDIT command. With the AUDIT command, you indicate which commands, users, objects, or privileges to audit. You can also indicate whether an audit record should be generated for each occurrence or once per session. If an auditing option is no longer required, you can turn off the option with the NOAUDIT command.

Execution of statements

When users execute PL/SQL and SQL statements, the server process examines the auditing options to determine if the statement being executed should generate an audit record. SQL statements inside PL/SQL program units are individually audited, as necessary, when the program unit is executed. Because views and procedures may refer to other database objects, several audit records may be generated as the result of executing a single statement.

Generating audit data

The generation and insertion of an audit trail record is independent of a user's transaction. Therefore, if a user's transaction is rolled back, the audit trail record remains intact. Because the audit record is generated during the execute phase, a syntax error that occurs during the parse phase will not cause an audit trail record to be generated.

Reviewing audit information

Examine the information that is generated during auditing by selecting from the audit trail data dictionary views or by using an operating system utility to view the operating system audit trail. This information is used to investigate suspicious activity and to monitor database activity.

Auditing Options

- **Statement auditing:**

```
AUDIT TABLE;
```

- **Privilege auditing:**

```
AUDIT create any trigger;
```

- **Schema object auditing:**

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

ORACLE

Auditing Options

Statement auditing: This is the selective auditing of SQL statements, not the specific schema objects on which it operates. For example, `AUDIT TABLE` tracks several DDL statements regardless of the table on which they are issued. You can set statement auditing to audit selected users or every user in the database.

Privilege auditing

This is the selective auditing of system privileges to perform corresponding actions, such as `AUDIT CREATE ANY TRIGGER`. You can set privilege auditing to audit a selected user or every user in the database.

Schema object auditing

This is the selective auditing of specific statements on a particular schema object, such as `AUDIT SELECT ON HR.EMPLOYEES`. Schema object auditing always applies to all users of the database.

You can specify any auditing option, and specify the following conditions:

- `WHENEVER SUCCESSFUL` / `WHENEVER NOT SUCCESSFUL`
- `BY SESSION` / `BY ACCESS`

Auditing Options

Fine-grained auditing:

- **Provides the monitoring of data access based on content**
- **Is implemented using the `DBMS_FGA` package**

Auditing Options

Fine-grained auditing: This provides the monitoring of data access based on content. A PL/SQL package `DBMS_FGA` administers value-based audit policies. Using `DBMS_FGA`, the DBA creates an audit policy on the target table. If any of the rows returned from a query block matches the audit condition, then an audit event entry, including username, SQL text, bind variable, policy name, session ID, timestamp, and other attributes, is inserted into the audit trail.

Disabling auditing

Use the `NOAUDIT` statement to stop auditing chosen by the `AUDIT` command.

Note: A `NOAUDIT` statement reverses the effect of a previous `AUDIT` statement. The `NOAUDIT` statement must have the same syntax as the previous `AUDIT` statement and it only reverses the effects of that particular statement. Therefore, if one `AUDIT` statement (statement A) enables auditing for a specific user, and a second (statement B) enables auditing for all users, then a `NOAUDIT` statement to disable auditing for all users reverses statement B, but leaves statement A in effect and continues to audit the user that statement A had specified.

Auditing User SYS

- **Auditing user SYS provides:**
  - **Extra level of security**
  - **Set AUDIT_SYS_OPERATIONS to True**
- **Non-auditing of user SYS:**
  - **Set AUDIT_SYS_OPERATIONS to False**
  - **This is the default value.**

ORACLE

Auditing User SYS

Beginning with Oracle9*i* Database Release 2, you can audit user SYS for all commands. Auditing SYS provides an extra level of security on databases where user SYS is a widely used account.

The static parameter AUDIT_SYS_OPERATIONS determines the level at which you want user SYS to be audited. A value of False retains the level of auditing of SYS used in previous versions. The commands audited include instance startup and shutdown operations, and connections to the database using administrator privileges.

Setting AUDIT_SYS_OPERATIONS to True allows you to increase auditing of SYS to include all commands. Because auditing for user SYS is recorded in an operating system audit file, the location of the audit file is determined by the parameter AUDIT_FILE_DEST. This file, as with Oracle trace files, requires frequent monitoring due to growth.

Obtaining Auditing Information

Information about auditing can be obtained by querying the following views:

- **ALL_DEF_AUDIT_OPTS**
- **DBA_STMT_AUDIT_OPTS**
- **DBA_PRIV_AUDIT_OPTS**
- **DBA_OBJ_AUDIT_OPTS**

ORACLE

Obtaining Auditing Information

Data Dictionary View	Description
ALL_DEF_AUDIT_OPTS	Default audit options
DBA_STMT_AUDIT_OPTS	Statement auditing options
DBA_PRIV_AUDIT_OPTS	Privilege auditing options
DBA_OBJ_AUDIT_OPTS	Schema object auditing options

Obtaining Audit Records Information

Information about auditing records can be obtained by querying the following views:

- **DBA_AUDIT_TRAIL**
- **DBA_AUDIT_EXISTS**
- **DBA_AUDIT_OBJECT**
- **DBA_AUDIT_SESSION**
- **DBA_AUDIT_STATEMENT**

ORACLE

Obtaining Audit Records Information

Listing audit records

The database audit trail (SYS.AUD$) is a single table in each Oracle database's dictionary. Several predefined views are available. Some of the views are listed in the slide. These views are created by the DBA.

Data Dictionary View	Description
DBA_AUDIT_TRAIL	All audit trail entries
DBA_AUDIT_EXISTS	Records for AUDIT EXISTS/NOT EXISTS
DBA_AUDIT_OBJECT	Records concerning schema objects
DBA_AUDIT_SESSION	All connect and disconnect entries
DBA_AUDIT_STATEMENT	Statement auditing records

Summary

In this lesson, you should have learned how to:

- **Outline auditing needs**
- **Enable and disable auditing**
- **Identify and use the various auditing options**
- **Obtain audit information**

ORACLE

Practice 18 Overview

There is no practice for this lesson.

ORACLE

19

Loading Data into a Database

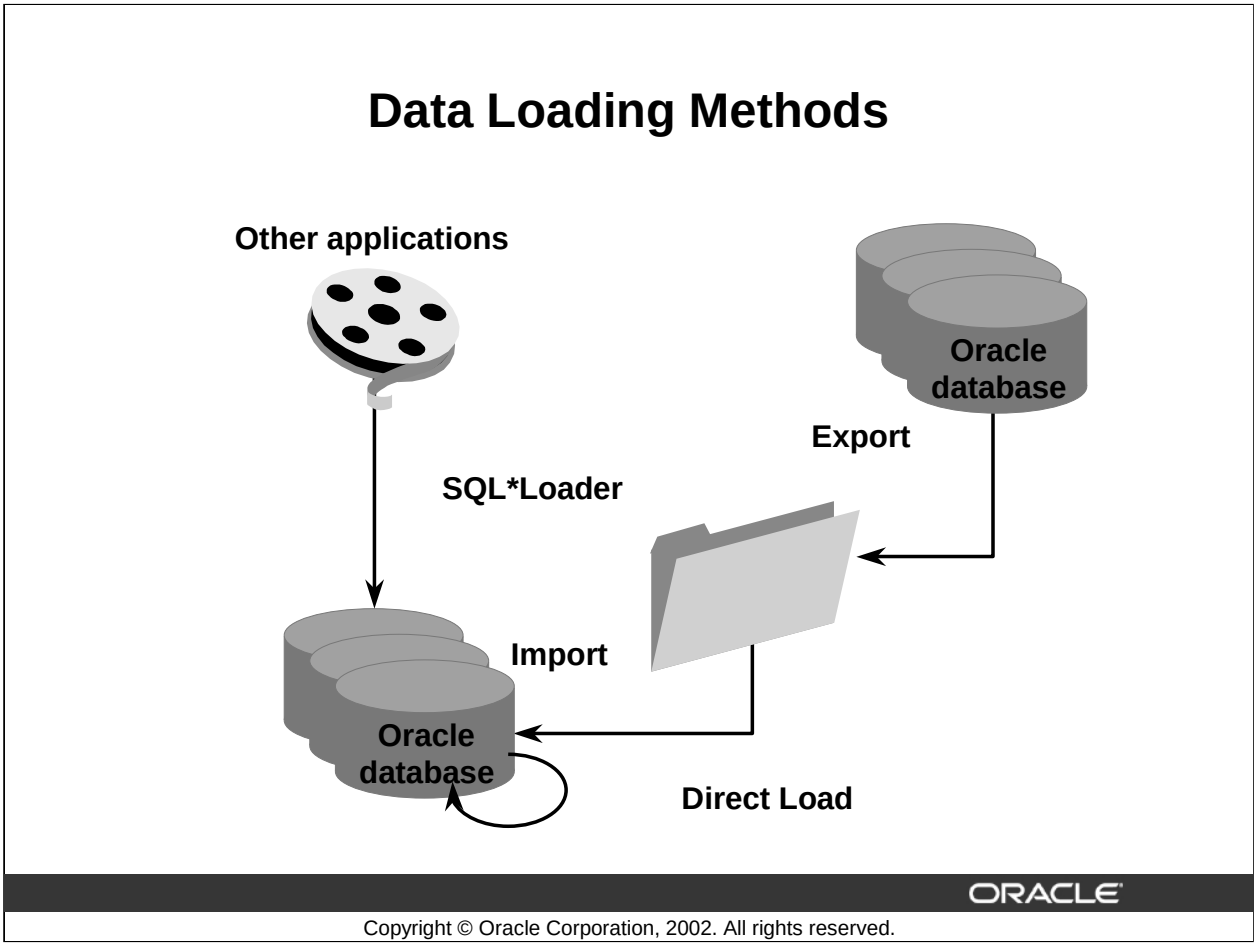
ORACLE

Objectives

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

- **Demonstrate usage of Direct Load operations**
- **Describe the usage of SQL*Loader**
- **Perform basic SQL*Loader operations**
- **List guidelines for using SQL*Loader and Direct Load**

ORACLE



Data Loading Methods

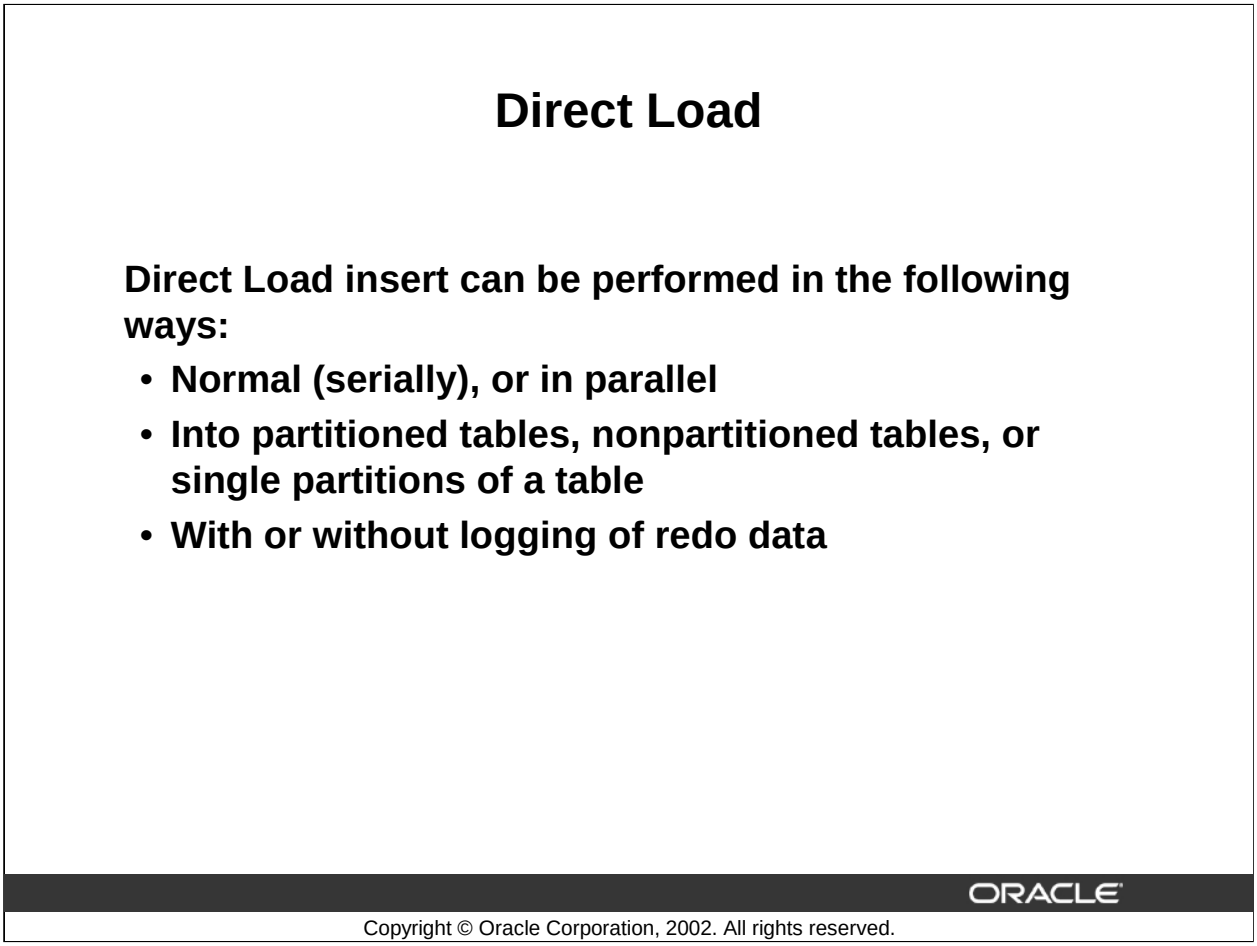
Several methods are available for loading data into tables in an Oracle database. Of the methods available, Direct Load insert and SQL*Loader are discussed here. Export and Import are covered in the *Oracle9i Database Administration Fundamentals II* course.

SQL*Loader

SQL*Loader loads data from external files into tables of an Oracle database. It has a powerful data parsing engine that places little limitation on the format of the data in the data file.

Direct Load

Direct Load insert can be used to copy data from one table to another table within the same database. It speeds up the insert operation, bypassing the database buffer cache and writing data directly into the data files.



Direct Load

Direct Load insert (serial or parallel) can only support the `INSERT ... SELECT` syntax of an `INSERT` statement, not the `INSERT` ... Values syntax. The parallelism for `INSERT` ... `SELECT` is determined from either parallel hints or parallel table definition. Oracle9*i* provides syntax extensions that extend the scope of the `INSERT` ... `SELECT` statement, so that you can insert rows into multiple tables as part of a single DML statement.

A Direct Load insert can be invoked by using the `APPEND` hint, as shown in following command:

```
INSERT /*+APPEND */ INTO [ schema. ] table
   [ [NO]LOGGING ]
sub-query;
```

where:

`schema`: Is the owner of the table

`table`: Is the name of the table

`sub-query`: Is the sub-query used to select the columns and rows for insert

Direct Load (continued)

`LOGGING` mode

When inserting using the `LOGGING` option, which is the default, the operation generates redo entries, making complete recovery possible in case of failures. If the `NOLOGGING` option is used, changes to data are not recorded in the redo log buffer. Some minimal logging still occurs for operations that update the data dictionary. The `NOLOGGING` mode is used if this attribute has been set for the table.

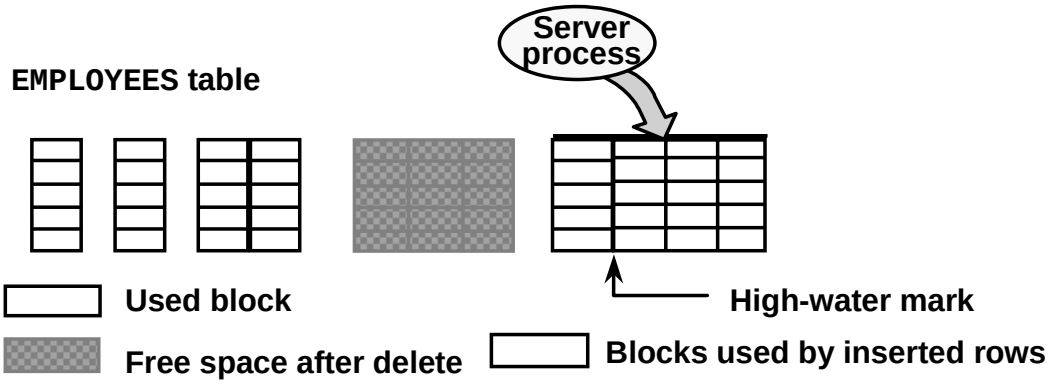
If several online modifications to the data in the table are likely to occur subsequently, then it is advisable to set the `NOLOGGING` attribute before the load and reset it to `LOGGING` after the load is completed.

Other considerations

All data that is loaded using Direct Load `insert` is loaded above the high-water mark. If the table contains many blocks where rows have been deleted, space may be wasted and full table scans may be slower.

Serial Direct Load

```
INSERT /*+ APPEND */ INTO emp
NOLOGGING
SELECT * FROM t_employees;
COMMIT;
```



ORACLE

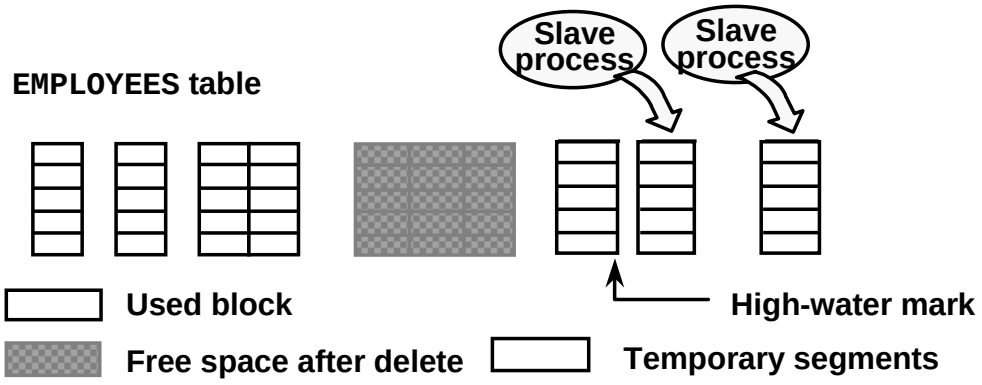
Serial Direct Load

Serial Direct Load insert into a nonpartitioned, partitioned, or subpartitioned table

Data is inserted beyond the current high-water mark of the table segment or each partition segment. The high-water mark is the level at which blocks have never been formatted to receive data. When a statement executes, the high-water mark is updated to the new value, making the data visible to others. When loading a partitioned or subpartitioned table, SQL*Loader partitions the rows and maintains indexes (which can also be partitioned). A Direct path load of a partitioned or subpartitioned table can be resource-intensive.

Parallel Direct Load

```
ALTER SESSION ENABLE PARALLEL DML;
INSERT /*+PARALLEL(hr.employees,2) */
INTO hr.employees NOLOGGING
SELECT * FROM hr.old_employees;
```



ORACLE

Parallel Direct Load

Direct Load inserts can be made in parallel by one of the following methods:

- Using a PARALLEL hint in the INSERT statement, as in the above example
- Creating the table or altering it to specify the PARALLEL clause

When parallel Direct Load inserts are made, the Oracle server uses several processes, known as parallel query slaves, to insert data into the table. Temporary segments are allocated to store the data inserted by each slave process. When the transaction commits, the extents in these individual segments become a part of the table in which records are inserted.

Note

- The ALTER SESSION ENABLE PARALLEL DML command must be executed at the beginning of a transaction.
- An object that is modified using parallel Direct Load insert cannot be queried or modified again within the same transaction.

For a detailed discussion of parallel Direct Load inserts, see the “Parallel Execution” section in *Oracle9i Database Reference*.

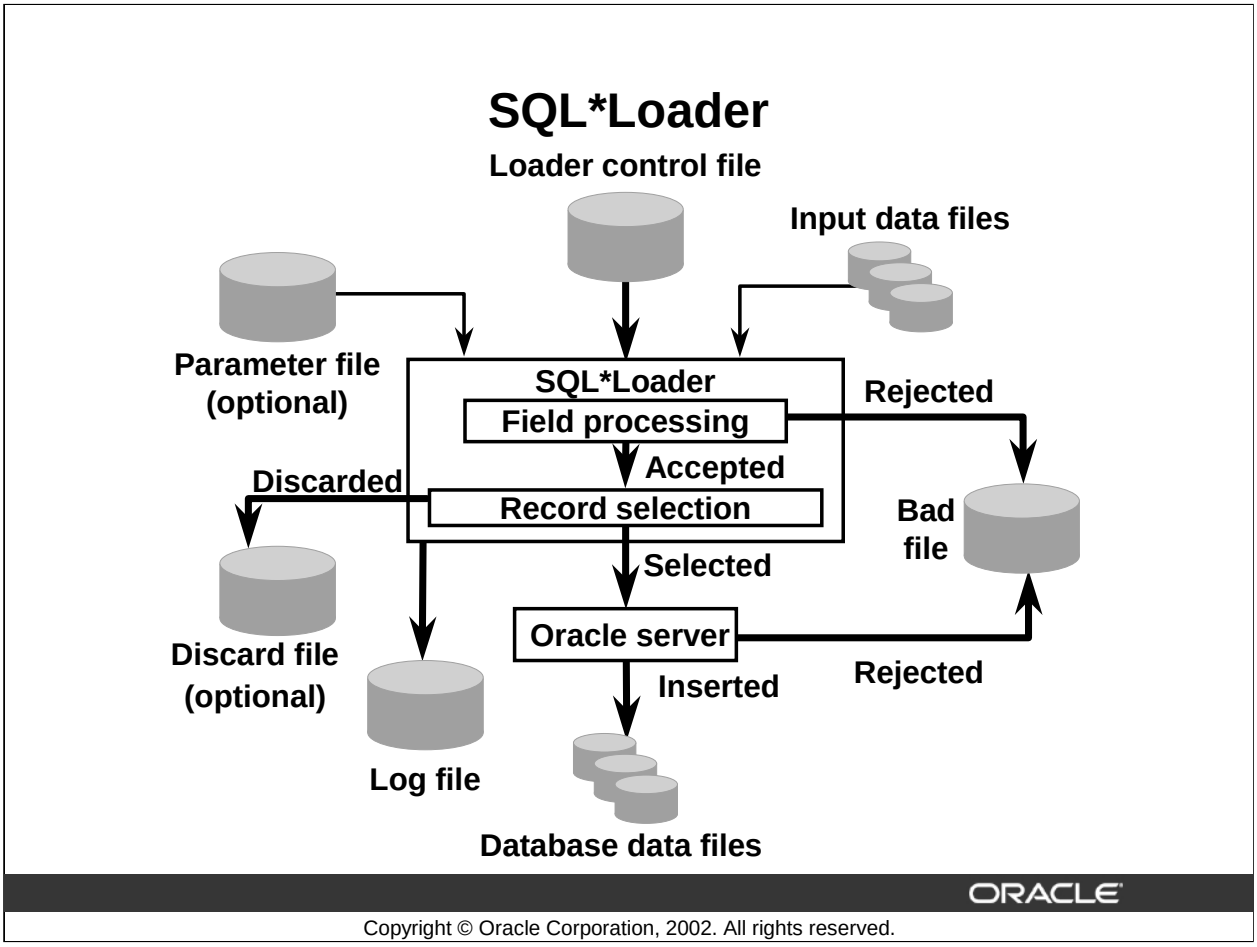
Parallel Direct Load (continued)

Parallel Direct Load insert into a nonpartitioned table

Each parallel execution server allocates a new temporary segment and inserts data into the temporary segment. When a statement executes, the parallel execution coordinator merges the new temporary segments into the primary table segment.

Parallel Direct Load insert into a partitioned table

Each parallel execution server is assigned one or more partitions, with no more than one process working per partition. The parallel execution server inserts data beyond the current high-water mark of the partition segments assigned to it. When a statement executes, the high-water mark of each partition segment is updated by the parallel execution coordinator to the new value, making the data visible to others.



SQL*Loader

SQL*Loader loads data from external files into tables in an Oracle database. SQL*Loader has the following features:

- SQL*Loader can use one or more input files.
- Several input records can be combined into one logical record for loading.
- Input fields can be of fixed or variable lengths.
- Input data can be in any format: character, binary, packed decimal, date, and zoned decimal.
- Data can be loaded from different types of media such as disk, tape, or named pipes.
- Data can be loaded into several tables in one run.
- Options are available to replace or to append to existing data in the tables.
- SQL functions can be applied on the input data before the row is stored in the database.
- Column values can be auto-generated based on rules. For example, a sequential key value can be generated and stored in a column.
- Data can be loaded directly into the table, bypassing the database buffer cache.

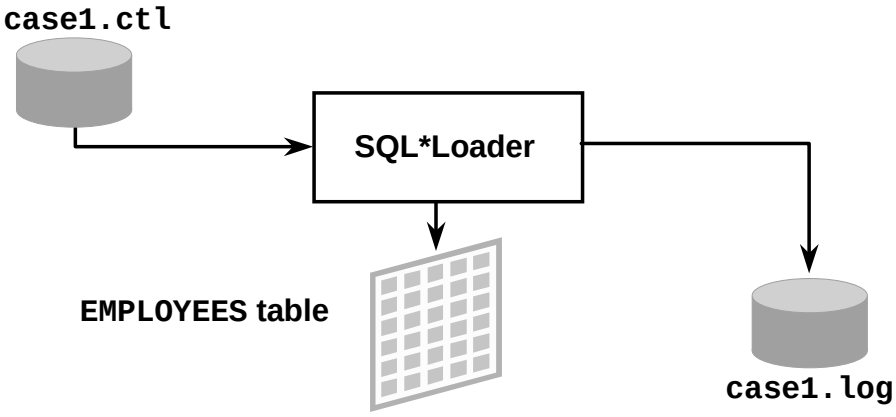
Files Used by SQL*Loader

SQL*Loader uses the following files:

- Loader control file: Specifies the input format, output tables, and optional conditions that can be used to load only part of the records found in the input data files
- Input data files: Contain the data in the format defined in the control file
- Parameter file: Is an optional file that can be used to define the command line parameters for the load
- Log file: Is created by SQL*Loader and contains a record of the load
- Bad file: Is used by the utility to write the records that are rejected during the load. (This can occur during input record validation by the utility or during record insertion by the Oracle server.)
- Discard file: Is a file that can be created, if necessary, to store all records that did not satisfy the selection criteria

Using SQL*Loader

```
$sqlldr hr/hr \
> control=case1.ctl \
> log=case1.log direct=Y
```



ORACLE

Using SQL*Loader

Command line

When you invoke SQL*Loader, you can specify parameters that establish session characteristics. Parameters can be entered in any order, optionally separated by commas. You can specify values for parameters, or in some cases, you can accept the default without entering a value.

If you invoke SQL*Loader without specifying any parameters, SQL*Loader displays a Help screen that lists the available parameters and their default values.

Using SQL*Loader (continued)

Using Oracle Enterprise Manager to Perform a Load using the Load Wizard

From the OEM Console:

1. Navigate to Databases > Schema > [Schema Name] > Tables.
2. Expand the table into which you will load the data.
3. Select Data Management > Load from the right-mouse menu.
4. This Load Wizard Introduction page will appear.
5. Click Next to begin.
   Note: The remaining slides within this lesson provide discussion of information requested within the Load Wizard as follows:
6. Using the Control File page
7. Using the Data File page
8. Using the Load Method page to define the load method
9. Using the Load Method page to define optional files
10. Using the Schedule page
11. Using the Job Information page
12. Using the Summary page

SQL*Loader Control File

The loader control file tells SQL*Loader:

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

ORACLE

SQL*Loader Control File

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

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

Although not precisely defined, a loader control file can be said to have three sections:

- The first section contains session-wide information, for example:
  - Global options such as bind size, rows, records to skip, and so on
  - `INFILE` clauses to specify where the input data is located
  - How data is to be loaded
- The second section consists of one or more `INTO TABLE` blocks. Each of these blocks contains information about the table into which the data is to be loaded, such as the table name and the columns of the table.
- The third section is optional and, if present, contains input data.

SQL*Loader Control File (continued)

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

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

Sample control file explanation:

1. This is how comments are entered in a control file. Comments can appear anywhere in the command section of the file, but they should not appear within the data.
2. The `LOAD DATA` statement tells SQL*Loader that this is the beginning of a new data load. If you were continuing a load that had been interrupted in progress, you would use the `CONTINUE LOAD DATA` statement.
3. The `INFILE` keyword specifies the name of a data file containing data that you want to load.

SQL*Loader Control File (continued)

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

SQL*Loader Control File (continued)

Using OEM Console to perform a Load Using the Load Wizard

From the OEM Console:

1. Navigate to Databases > Schema > [Schema Name] > Tables.
2. Expand the table into which you will load the data.
3. Select Data Management > Load from the right mouse menu.
4. The Load Wizard Introduction page will appear.
5. Click Next to begin.
   Note: The remaining slides within this lesson provide discussion of information requested within the Load Wizard as follows.
6. Using the Control File page: Specify the full path and name of the control file on the database server machine.
7. Select Next.
8. Using the Data File page
9. Using the Load Method page to define the load method
10. Using the Load Method page to define optional files
11. Using the Schedule page
12. Using the Job Information page
13. Using the Summary page

Control File Syntax Considerations

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

ORACLE

Control File Syntax Considerations

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

Input Data and Data Files

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

Input Data and Data Files

Fixed-record format

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

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

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

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

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

Input Data and Data Files (continued)

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

Variable-record format

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

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

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

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

Stream-record format

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

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

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

where:

'`char_string`' is a string of characters enclosed in single or double quotation marks

X'`hex_string`' is a byte string in hexadecimal format

Input Data and Data Files (continued)

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

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

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

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

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

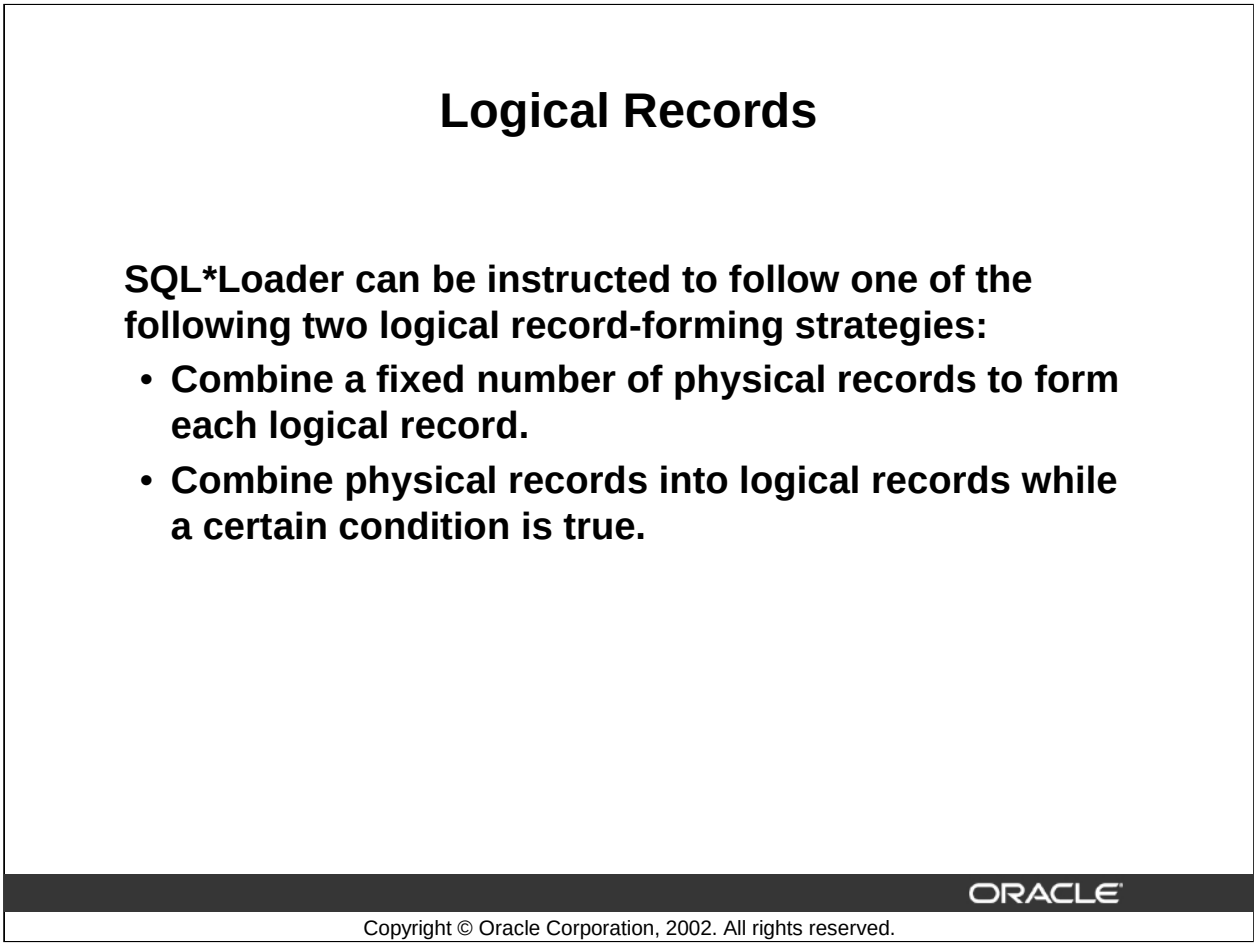
```
load data
infile 'example.dat' "str '|\n'"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1 char(5),
col2 char(7))
example.dat:
hello,world,|
james,bond,|
```

Input Data and Data Files (continued)

Using Oracle Enterprise Manager to Perform a Load using the Load Wizard

From the OEM Console:

1. Navigate to Databases > Schema > [Schema Name] > Tables.
2. Expand the table that you will load the data into.
3. Select Data Management > Load from the right-mouse menu.
4. This Load Wizard Introduction page will appear.
5. Click Next to begin.
   Note: The remaining slides within this lesson provide discussion of information requested within the Load Wizard.
6. Using the Control File page: Specify the full path and name of the control file on the database server machine. Click Next.
7. Using the Data File page: This page defines how you want to specify the file containing the data. You can either:
   - Use the data as specified in the control file (identified above)
   - Or provide the full path of the database server machine where the data file resides. If using this method, please note case sensitivity on UNIX environments.
8. Using the Load Method page to define load method
9. Using the Load Method page to define optional files
10. Using the Schedule page
11. Using the Job Information page
12. Using the Summary page



Logical Records

SQL*Loader organizes the input data into physical records, according to the specified record format. By default, a physical record is a logical record. However, for added flexibility, SQL*Loader can be instructed to combine a number of physical records into a logical record. SQL*Loader can do this in one of two ways:

- Combine a fixed number of physical records to form each logical record
- Combine physical records into logical records while a certain condition is true

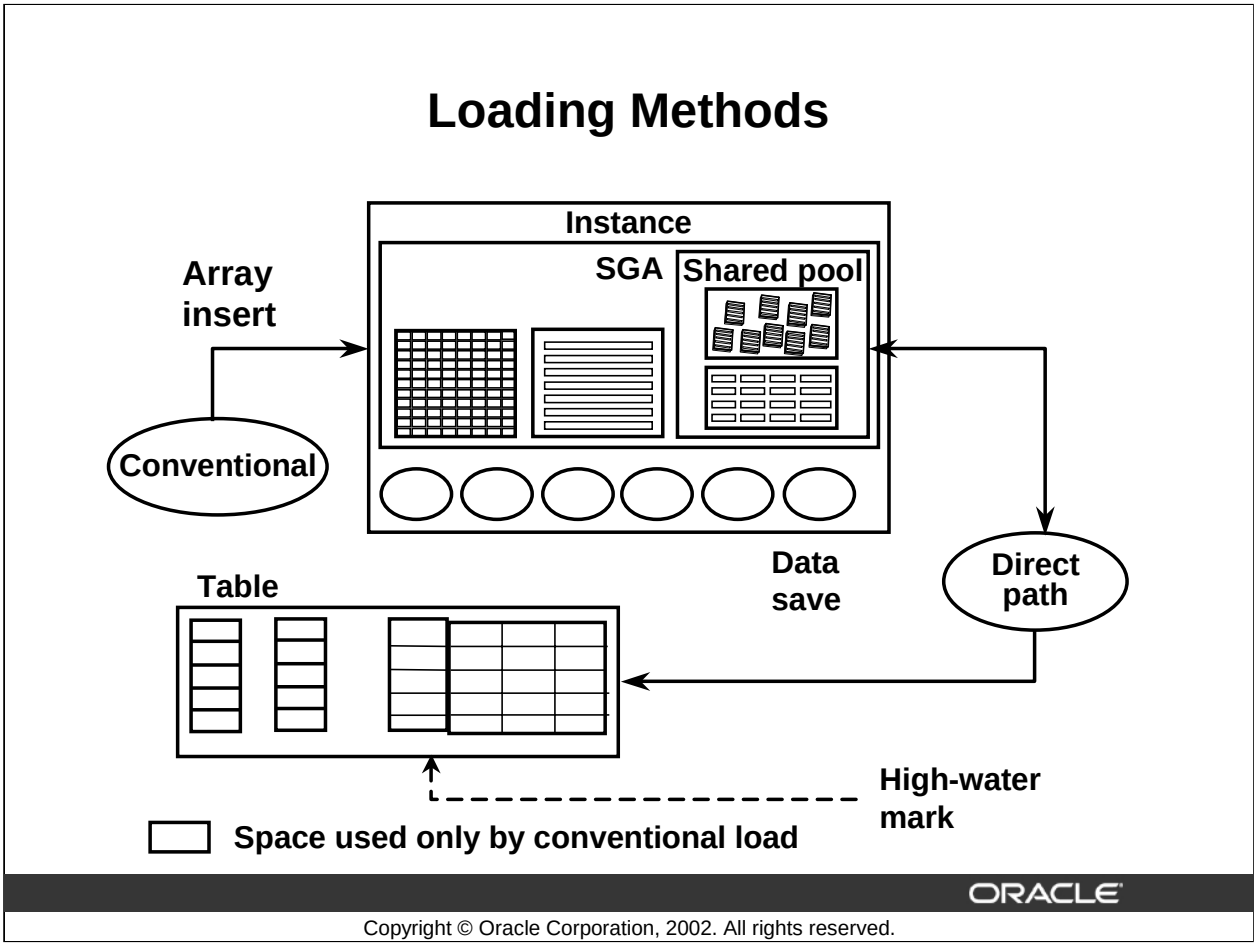
Using `CONCATENATE` to assemble logical records

`CONCATENATE` is used when SQL*Loader should always add the same number of physical records to form one logical record. The following is an example of using `CONCATENATE`, in which `integer` specifies the number of physical records to combine:

```
CONCATENATE integer
```

Using `CONTINUEIF` to assemble logical records

`CONTINUEIF` must be used if the number of physical records to be continued varies. The `CONTINUEIF` keyword is followed by a condition that is evaluated for each physical record as it is read. For example, two records might be combined if there was a pound sign (#) in character position 80 of the first record. If there was any other character, the second record would not be added to the first.



Loading Methods

SQL*Loader provides two methods for loading data:

- Conventional path
- Direct path

Conventional path load

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

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

Loading Methods (continued)

Direct path load

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

Direct path load is generally faster than conventional path load, but cannot be used in all situations. The next section compares conventional path to Direct path loading, and provides examples of situations in which each of them can be used.

Note: The `catldr.sql` script, supplied by Oracle, creates views that are used by Direct path load. It is automatically invoked when the `catalog.sql` script is run.

Loading Methods (continued)

Using Oracle Enterprise Manager to Perform a Load using the Load Wizard

From the OEM Console:

1. Navigate to Databases > Schema > [Schema Name] > Tables.
2. Expand the table into which you will load the data
3. Select Data Management > Load from the right-mouse menu.
4. This Load Wizard Introduction page will appear.
5. Click Next to begin.
   Note: The remaining slides within this lesson provide discussion of information requested within the Load Wizard.
6. Using the Control File page: Specify the full path and name of the control file on the database server machine. Click Next.
7. Using the Data File page: This page defines how you want to specify the file containing the data. You can either:
   - Use the data as specified in the control file (identified above)
   - Or provide the full path of the database server machine where the data file resides. If you are using this method, please note case sensitivity on UNIX environments.
8. Using the Load Method page to define the load method:
   This page defines the load method that you want to use. You can choose one of the following methods:
   - Conventional Path
   - Direct Load
   - Parallel Direct Load
9. Using the Load Method page to define optional files
10. Using the Schedule page
11. Using the Job Information page
12. Using the Summary page

Comparing Direct and Conventional Path Loads

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

ORACLE

Comparing Direct and Conventional Path Loads

Method of saving data

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

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

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

Comparing Direct and Conventional Path Loads (continued)

Logging changes

Conventional path loading generates redo entries just as any DML statement. When using a Direct path load, redo entries are not generated if:

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

Enforcing constraints

During a conventional path load, all enabled constraints are enforced as they would be during any DML operation.

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

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

Firing `INSERT` triggers

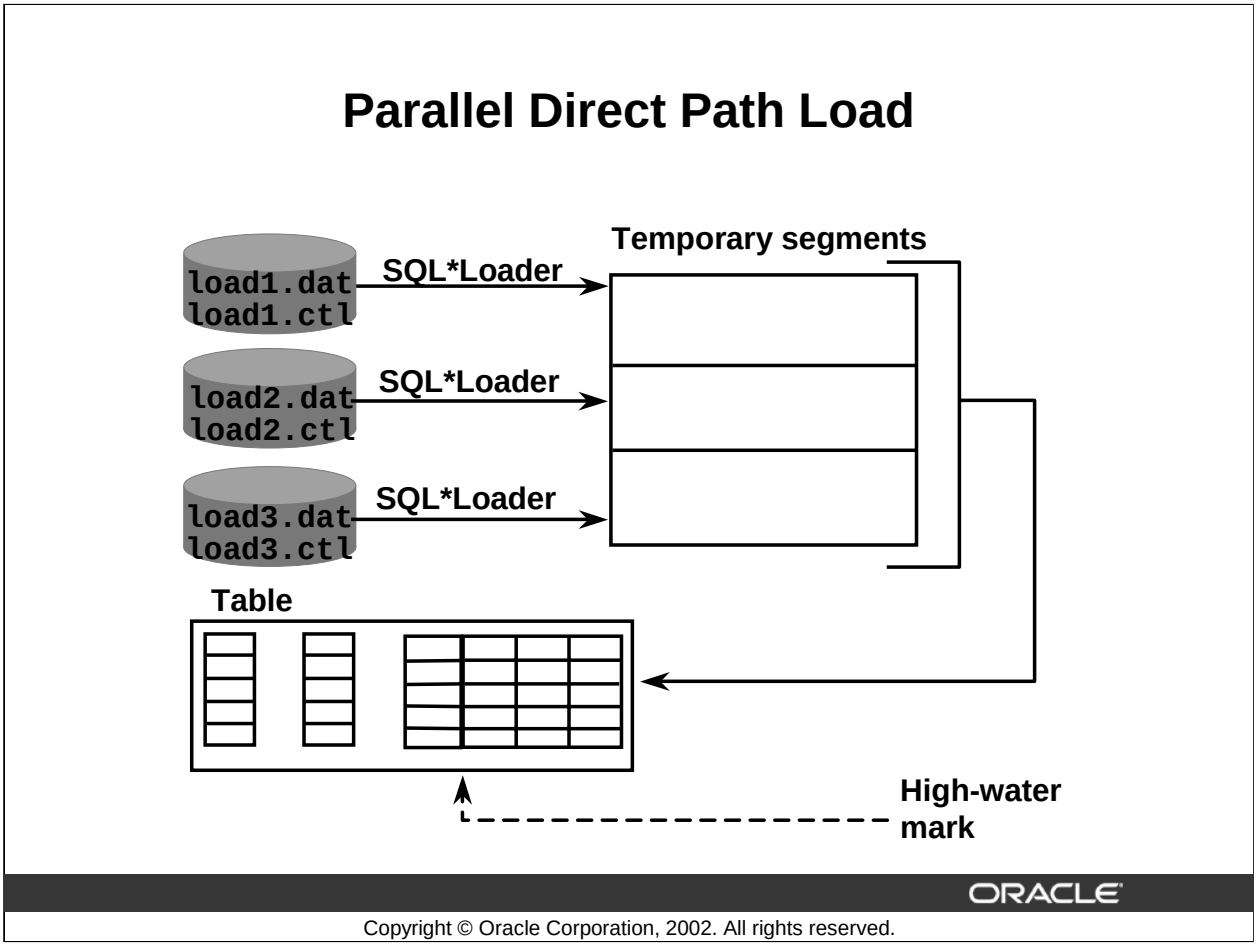
`WHILE INSERT` triggers are fired during conventional loads; they are disabled before a Direct path load and reenabled at the end of the run. They may remain disabled if a referenced object is not accessible at the end of the run. Consider using conventional path loads to load data into tables with `INSERT`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

Loading into clustered tables

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

Locking

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



Parallel Direct Path Load

Multiple SQL*Loader sessions improve the performance of a Direct path load. Three models of concurrency can be used to minimize the time required for data loading:

- Parallel conventional path loads
- Intersegment concurrency with Direct path load method
- Intrasegment concurrency with Direct path load method

Concurrent conventional path

If triggers or integrity constraints pose a problem, but faster loading is desired, consider using concurrent conventional path loads. Use multiple load sessions executing concurrently on a multiple-CPU system. Split the input data files into separate files on logical record boundaries, then load each such input data file with a conventional path load session.

Intersegment concurrency

Intersegment concurrency can be used for concurrent loading of different objects. This technique can be applied to concurrent Direct path loading of different tables, or to concurrent Direct path loading of different partitions of the same table.

Intrasegment concurrency

A parallel Direct path load allows multiple Direct path load sessions to concurrently load the data into the same table, or into the same partition of a partitioned table allowing intrasegment parallelism.

Data Conversion

During a conventional path load, data fields in the data file are converted into columns in the database in two steps:

- **The field specifications in the control file are used to interpret the format of the data file and convert it to a SQL `INSERT` statement using that data.**
- **The Oracle database server accepts the data and executes the `INSERT` statement to store the data in the database.**

ORACLE

Discarded or Rejected Records

- **Bad file:**
  - **SQL*Loader rejects records when the input format is invalid.**
  - **If the Oracle database finds that the row is invalid, then the record is rejected and SQL*Loader puts it in the bad file.**
- **Discard file:**
  - **This can be used only if it has been enabled.**
  - **This file contains records that were filtered out because they did not match any record-selection criteria specified in the control file.**

ORACLE

Discarded or Rejected Records

The bad file

The bad file contains records that were rejected, either by SQL*Loader or by the Oracle database.

SQL*Loader rejects

Records are rejected by SQL*Loader when the input format is invalid. For example, if the second enclosure delimiter is missing, or if a delimited field exceeds its maximum length, then SQL*Loader rejects the record. Rejected records are placed in the bad file.

Oracle rejects

After a record is accepted for processing by SQL*Loader, a row is sent to Oracle for insertion. If Oracle determines that the row is valid, then the row is inserted into the database. If not, then the record is rejected, and SQL*Loader puts it in the bad file. The row may be rejected, for example, because a key is not unique, because a required field is null, or because the field contains invalid data for the Oracle data type.

Discarded or Rejected Records (continued)

The discard file

As SQL*Loader executes, it may create a file called the discard file. This file is created only when it is needed, and only if you have specified that a discard file should be enabled. The discard file contains records that were filtered out of the load because they did not match any record-selection criteria specified in the control file. The discard file therefore contains records that were not inserted into any table in the database. You can specify the maximum number of such records that the discard file can accept.

Discarded or Rejected Records (continued)

Using Oracle Enterprise Manager to Perform a Load using the Load Wizard

From the OEM Console:

1. Navigate to Databases > Schema > [Schema Name] > Tables.
2. Expand the table into which you will load the data.
3. Select Data Management > Load from the right-mouse menu.
4. This Load Wizard Introduction window will appear.
5. Click Next to begin.
   Note: The remaining slides within this lesson provide discussion of information requested within the Load Wizard.
6. Using the Control File page: Specify the full path and name of the control file on the database server machine. Click Next.
7. Using the Data File page: This page defines how you want to specify the file containing the data. You can either:
   - Use the data as specified in the control file (identified above)
   - Or provide the full path of the database server machine where the data file resides. If you are using this method, please note case sensitivity on UNIX environments.
8. Using the Load Method page to define the load method:
   This page defines the load method you want to use. You can choose one of the following methods:
   - Conventional Path
   - Direct Load
   - Parallel Direct Load

   After the above information is provided in the Load Wizard, the load will run as an OEM job.
9. Using the `Load Method` page to define optional files: You can define additional loading requirements, tuning options, and optional files. In order to define the path for the bad file, discard file, and the log file:
   - Click the Advanced button on this page.
   - Open the Optional Files page.
   - Select to generate any one of the files and provide a full path for each selected.
   - Click OK.

   After the above information is provided in the Load Wizard, the load will run as an OEM job.

Discarded or Rejected Records (continued)

Using Oracle Enterprise Manager to Perform a Load using the Load Wizard (continued)

Note: After the preceding information is provided in the Load Wizard, the load will run as an OEM job.

10. Using the Schedule page: This page specifies when you want the job to run.
11. Using the Job Information page: This page designates a specific name for the job and your exact requirements of the job. It also allows you to override any preferred credentials. It is a good idea to define the schema owner at this point.
12. Using the Summary page: This page provides a summary of the Load Wizard page information submitted by you. This page gives you an opportunity to review the information and make any necessary changes. When you are satisfied with its accuracy, select OK and the job is submitted.
13. Navigate to and select Jobs from the list.
14. View the History page to check your job.

Log File Contents

- **Header information**
- **Global information**
- **Table information**
- **Data file information**
- **Table load information**
- **Summary statistics**
- **Additional statistics for Direct path loads and multithreading information**

ORACLE

Log File Contents

The Header Information section contains the following entries:
- Date of the run
- Software version number

The Global Information section contains the following entries:
- Names of all input/output files
- Echo of command-line arguments
- Continuation character specification

The Table Information section provides the following entries for each table loaded:
- Table name
- Load conditions, if any. That is, whether all records were loaded or only those meeting the `WHEN` clause criteria.
- `INSERT`, `APPEND`, or `REPLACE` specification
- The following column information:
  - If found in data file, the position, length, data type, and delimiter
  - If specified, `RECNUM`, `SEQUENCE`, `CONSTANT`, or `EXPRESSION`
  - If specified, `DEFAULTIF` or `NULLIF`

Log File Contents (continued)

If the SQL*Loader control file contains any directives for loading date time or interval data types, then the log file contains the `DATETIME` or `INTERVAL` keyword under the data type heading. If applicable, the `DATETIME` or `INTERVAL` keyword is followed by the corresponding mark.

The Datafile Information section appears only for data files with data errors and provides the following entries:

- SQL*Loader and Oracle data record errors
- Records discarded

The Table Load Information section provides the following entries for each table that was loaded:

- Number of rows loaded
- Number of rows that qualified for loading but were rejected due to data errors
- Number of rows that were discarded because they met no `WHEN`-clause tests
- Number of rows whose relevant fields were all null

The Summary Statistics section displays the following data:

- Amount of space used:
  - For bind array (what was actually used based on `BINDSIZE` specified)
  - For other overhead (always required, independent of `BINDSIZE`)
- Cumulative load statistics; that is, for all data files, the number of records that were skipped, read, or rejected

The following statistics are logged when a table is loaded:

- Direct path load of a partitioned table reports per-partition statistics.
- Conventional-path load cannot report per-partition statistics.

If media recovery is not enabled, the load is not logged. That is, if media recovery is disabled, then the request for a logged operation is overridden.

SQL*Loader Guidelines

- **Use a parameter file to specify commonly used command line options.**
- **Place data within the control file only for a small, one-time load.**
- **Improve performance by:**
  - **Allocating sufficient space**
  - **Sorting the data on the largest index**
  - **Specifying different files for temporary segments for parallel loads**

ORACLE

SQL*Loader Guidelines

Use the following guidelines when using SQL*Loader to minimize errors and improve performance:

- Use a parameter file to specify commonly used command line options. For example, if loading into a data warehouse every week, all options except the names of the files can be the same.
- Separate the control file and the data file to permit reusing control files for several load sessions.
- Preallocate space based on the expected data volume to prevent dynamic allocation of extents during the load and to improve the speed of the load.
- When Direct loads are used, temporary segments are used to generate indexes for the new data. These indexes are merged with the existing indexes at the end of the load. By sorting the input data on the keys of the largest index, use of sort space can be minimized.
- With parallel Direct Loads, you can specify the location of temporary segments that are used for inserting data. For each load session, specify a different database file to achieve maximum performance.

Summary

In this lesson, you should have learned how to:

- **Describe the usage of SQL*Loader**
- **Perform basic SQL*Loader operations**
- **Demonstrate proficiency using Direct Load operations**
- **List guidelines for using SQL*Loader and Direct Load operations**

ORACLE

Practice 19 Overview

This practice covers the following topics:

- **Using SQL*Loader to restore data:**
  - **Using a control file**
  - **Using a data file**
- **Using Direct Load to load data**

ORACLE

Practice 19 Overview

Note: Practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 19: Loading Data into a Database

1 a Examine the data files to be used in the load in order to become familiar with the control and data file formats.
- Examine the following files if using SQL*Plus:
  `lcase1.ctl`, `lcase2.ctl`, and `lcase2.dat`
- Examine the following files if using Oracle Enterprise Manager:
  `OEMsqlcase1.ctl`, `OEMsqlcase2.ctl`, and `OEMsqlcase2.`

b As user `HR`, run the `lab19_01.sql` script to create the `DEPARTMENTS2` table.

2 a Run SQL*Loader using the Conventional Path method to load data into the `DEPARTMENTS2` table. Use the following control file:
- Using SQL*Plus: `lcase1.ctl`
- Using Oracle Enterprise Manager: `OEMsqlcase1.ctl`

Files are located in the `LABS` directory.

Note: This runs using a control file that contains the input data.

2 b Examine the log file using operating system command.

c Query the `DEPARTMENTS2` table to check the data.

3 Delete all the records in the `DEPARTMENTS2` table.

4 a Run SQL*Loader in Direct-Path mode to load data into the `DEPARTMENTS2` table. Use the following control files:
- Using SQL*Plus: `lcase2.ctl`
- Using Oracle Enterprise Manager: `OEMsqlcase2.ctl`

Files are located in the `LABS` directory.

Note: This runs using an input data file to load data.

b Examine the log file using operating system command.

c Query the `DEPARTMENTS2` table to check the data.

5 a As user `HR`, create a table `EMPLOYEES2` as select from the `EMPLOYEES` table. Truncate the table. Then select from the `EMPLOYEES2` table to verify no data is in the table.

b Perform a direct-load insert into `EMPLOYEES2` from `EMPLOYEES` and query `EMPLOYEES2` to verify the load.

6 a Truncate the `EMPLOYEES2` table once again. Then select from the `EMPLOYEES2` table to verify no data is in the table.

b Restore the data with a parallel direct-load insert from the `EMPLOYEES` table. Specify a degree of parallelism of two. Verify the data.

20

Using Globalization Support

ORACLE

Objectives

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

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

ORACLE

Globalization Support Features

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

ORACLE

Globalization Support Features

Globalization Support ensures that database utilities and error messages, sort order, date, time, monetary, numeric, and calendar conventions automatically adapt to the native language.

Oracle currently supports 57 languages, 88 territories, 84 linguistic sorts (71 monolingual and 13 multilingual), and 235 encoded character sets.

The language-dependent operations are controlled by a number of parameters and environment variables on both the client and the server sides.

The server and the client may run in the same or different locations. When the client and the server use different character sets, the Oracle server handles character set conversion automatically.

Globalization Support Features (continued)

- Users can interact, store, process, and retrieve data in their native languages, including Western European, Eastern European, Middle Eastern, East Asian, and Southeast Asian languages.
- Different countries and geographies dictate different cultural conventions that directly affect data formats.
- Many different character encoding schemes, including single-byte, multibyte, and fixed-width encoded character sets are supported.
- The Oracle server provides many different linguistic sorts for linguistically accurate sorting.
- Database utilities and error messages appear in the supported native language. Oracle products are translated into 30 different languages.
- Date and time formats can be expressed according to International Organization for Standardization (ISO) conventions for fractional seconds, second, minute, hour, day, month, and year. Time zone regions can be used to support daylight savings time.
- National calendars such as Gregorian, Japanese, Imperial, and Thai Buddha are supported.
- Numeric data is represented in the appropriate local formats.
- Currency symbols reflect the local economy and ISO conventions. Credit and debit symbols also differ from location to location.

Encoding Schemes

Oracle supports different classes of character encoding schemes:

- **Single-byte character sets**
  - **7-bit**
  - **8-bit**
- **Varying-width multibyte character sets**
- **Fixed-width multibyte character sets**
- **Unicode (AL32UTF8, AL16UTF16, UTF8)**

ORACLE

Encoding Schemes

A character encoding scheme specifies numeric codes corresponding to characters that a computer or terminal can display and receive.

Character encoding schemes are used to interpret data into meaningful symbols from a terminal to a host machine.

Oracle provides different classes of encoding schemes:

- Single-byte
- Varying-width
- Fixed-width
- Unicode

Single-byte character sets

In a single-byte character set, each character occupies one byte. Single-byte 7-bit encoding schemes can define up to 128 (2^7) characters; single-byte 8-bit encoding schemes can define up to 256 (2^8) characters.

Encoding Schemes (continued)

Examples of single-byte schemes

7-bit character set: ASCII 7-bit American (US7ASCII)

8-bit character set:

- ISO 8859-1 West European (WE8ISO8859P1)
- EBCDIC Code Page 500 8-bit West European (WE8EBCDIC500)
- DEC 8-bit West European (WE8DEC)

Varying-width multibyte character sets

A varying-width multibyte character set is represented by one or more bytes per character. Multibyte character sets are commonly used for Asian language support. Some multibyte encoding schemes use the value of the most significant bit to indicate if a byte represents a single byte or is part of a series of bytes representing a character. However, other character encoding schemes differentiate single-byte from multibyte characters. A shift-out control code, sent by a device, indicates that the following bytes are double-byte characters, until a shift-in code is encountered.

Examples of varying-width multibyte schemes

- Japanese Extended UNIX Code (JEUC)
- Chinese GB2312-80 (CGB2312-80)
- AL32UTF8 (UTF-8)

Fixed-width multibyte character sets

Fixed-width multibyte character sets provide support similar to multibyte character sets, except that the format is a fixed number of bytes for each character.

The resulting benefit is having a uniform byte size representation for each character.

Only one fixed-width multibyte character set is supported and it is only in the National Character Set, AL16UTF16.

Example of fixed-width multibyte character sets

AL16UTF16, 16-bit Unicode (fixed width 2-byte Unicode)

Encoding Schemes (continued)

Unicode character set

Unicode is a worldwide character-encoding standard that can represent all characters for computer usage, including technical symbols and characters used in publishing. Unicode Standard, version 3.0 contains 49,149 characters, with a capacity for over one million characters.

The Unicode character repertoire can be represented in a number of different encoding formats. UTF-16 (Universal Character Set Transformation Format) is a two-byte, fixed-width format; UTF-8 is a multibyte, varying-width format.

Oracle provides AL32UTF8, UTF8, and UTFE as database character sets and AL16UTF16 and UTF8 as national character sets. The advantage of UTF-8 based character sets is that they include ASCII using the same single-byte encoding. Because UTF8 is a superset of ASCII, database character set migration is easier when upgrading ASCII-based characters sets to Unicode.

Note: Notice above that UTF-16 and UTF-8, with hyphens, refer to the Unicode Standard encodings, while UTF8, AL32UTF8, and AL16UTF16, without hyphens, refer to Oracle character sets based on the Unicode Standard.

Database Character Sets and National Character Sets

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

ORACLE

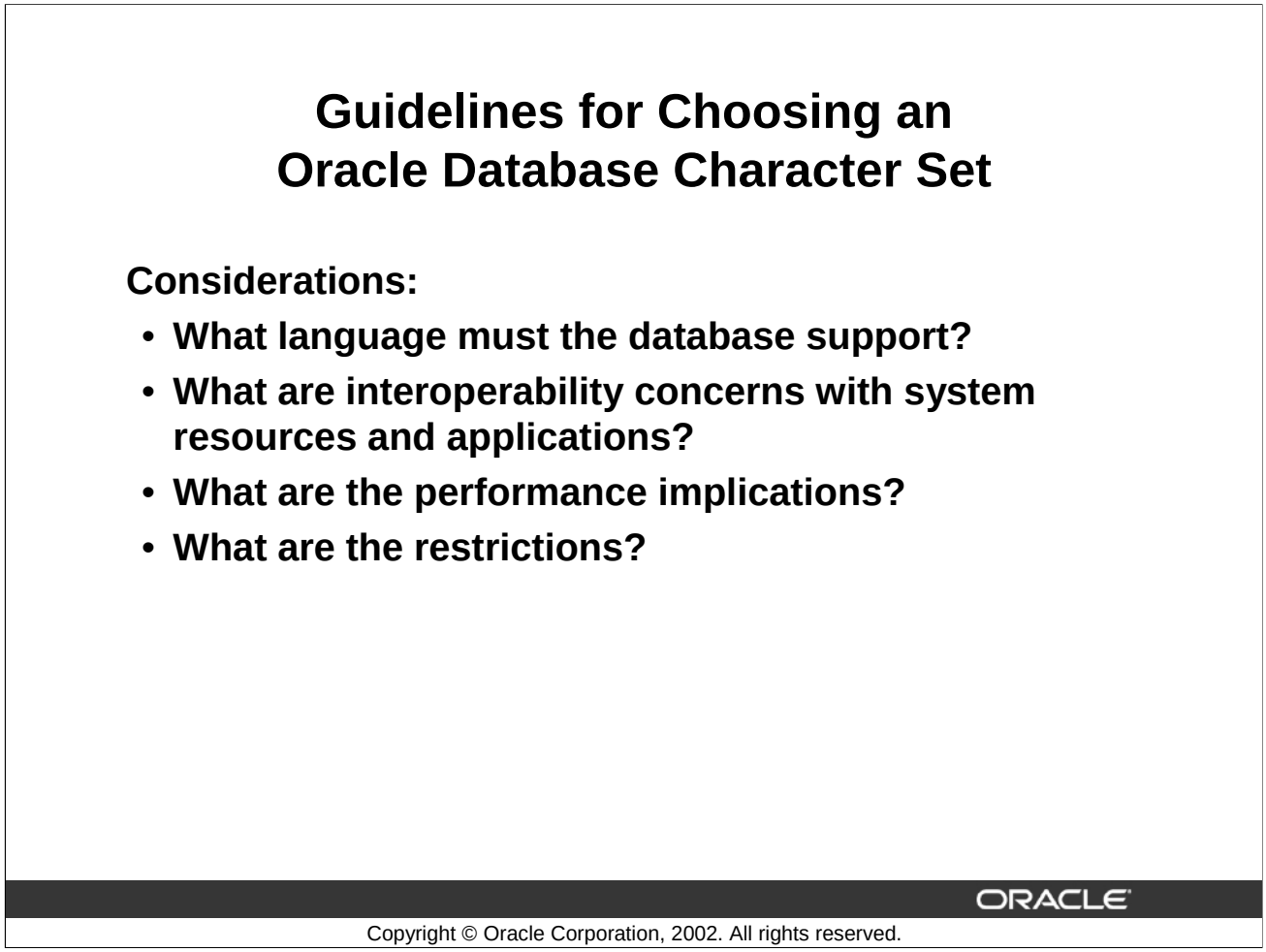
Database Character Sets and National Character Sets

Character set types

The CREATE DATABASE statement contains the CHARACTER SET clause and the additional optional NATIONAL CHARACTER SET clause that declare the character set to be used as the database character set and the national character set. If no NATIONAL CHARACTER SET clause is present, the national character set defaults to AL16UTF16.

Because the database character set is used to identify and to hold SQL and PL/SQL source code, it must have either EBCDIC or 7-bit ASCII as a subset, whichever is native to the platform. Therefore, it is not possible to use a fixed-width, multibyte character set as the database character set, only as the national character set.

The National Character set is for Unicode storage only and the SQL NCHAR data types (NCHAR, NVARCHAR2, and NCLOB) are Unicode Datatypes in Oracle9*i*.



Guidelines for Choosing an Oracle Database Character Set

What language must the database support?

Although several character sets might meet your current language requirements, you should consider future requirements as well. If you know that you will need to expand support in the future for different languages, picking a character set with a wider range now will eliminate the need for migration later.

What are interoperability concerns with system resources and applications?

Whereas the database maintains and processes the actual character data, there are other resources that you must depend on from the operating system. For instance, the operating system supplies fonts that correspond to the character set you have chosen. Input methods that support the languages desired and application software must also be compatible with a particular character set.

If you choose a character set that is not available on the operating system, Oracle can convert the operating system character set to the database character set. However, there is some character set conversion overhead, and you should make sure that the operating system character set has an equivalent character repertoire to avoid any possible data loss.

Guidelines for Choosing an Oracle Database Character Set (continued)

What are the performance implications?

There can be different performance overheads in handling different encoding schemes, depending on the character set chosen. For best performance, you should try to choose a character set that avoids character set conversion and uses the most efficient encoding for the languages desired. Single-byte character sets are more optimal for performance than multibyte character sets, and they are also the most efficient in terms of space requirements. However, single-byte character sets limit how many languages you can use.

What are the restrictions?

The restrictions are as follows:

- You cannot choose a database character set that is fixed-width multibyte.
- You cannot have an ASCII-based character set as the database character set on EBCDIC platforms.
- You cannot have an EBCDIC based database character set on ASCII platforms.

Guidelines for Choosing an Oracle National Character Set

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

ORACLE

Guidelines for Choosing an Oracle National Character Set

Two choices are available for the National Character Set, AL16UTF16 and UTF8. AL16UFT16 is a fixed width 2-byte Unicode character set. UTF8 is a variable width, one to three byte, Unicode character set. European characters, in UTF8, are stored in one to two bytes and thus save space over AL16UTF16 which stores characters in two bytes. Asian characters in UTF8 are stored in three bytes and require more space than AL16UTF16.

Because AL16UTF16 is a fixed-width encoding, it performs faster than the variable width UTF8.

AL16UTF16 supports the latest version of the Unicode standard, which is 3.0 in Oracle9*i*.

UTF8 support Unicode version 3.0 in Oracle9*i*, and will remain at Unicode version 3.0 in future releases.

Choosing a Unicode Solution: Unicode Database

When should you use a Unicode database?

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

ORACLE

Choosing a Unicode Solution: Unicode Database

Easy code migration for Java or PL/SQL

A Unicode database minimizes code changes when implementing multiple languages by storing multilingual data in existing SQL `CHAR` data types (`CHAR`, `VARCHAR2`, `CLOB`, and `LONG`). It is not necessary to recode for the SQL `NCHAR` data types.

Easy data migration from ASCII-based data

If the current database character set and data are strict `US7ASCII`, the database can be migrated with a simple `ALTER DATABASE` statement.

Evenly distributed multilingual data

If multilingual data is distributed throughout the database, select a Unicode database solution because it does not require you to identify which columns store multilingual data.

Oracle text

To use multilingual BLOBs with Oracle Text, a Unicode database solution is required.

Choosing a Unicode Solution: Unicode Data Type

When should you use a Unicode data type?

- **While adding multilingual support incrementally**
- **Packaged applications**
- **Performance: Single byte database character set with a fixed-width national character set**
- **Better support for UTF-16 with Windows clients**

ORACLE

Choosing a Unicode Solution: Unicode Data Type

Add multilingual support incrementally

To add Unicode support without migrating the database, you can add SQL NCHAR data types to new and existing tables.

Package application

Use the SQL NCHAR data type for packaged applications because it is a reliable Unicode data type in which the data is always stored in Unicode, and the length of the data is always specified in UTF-16 code units. As a result, you need to test the application only once, and your application will run on customer databases of any database character set.

Performance

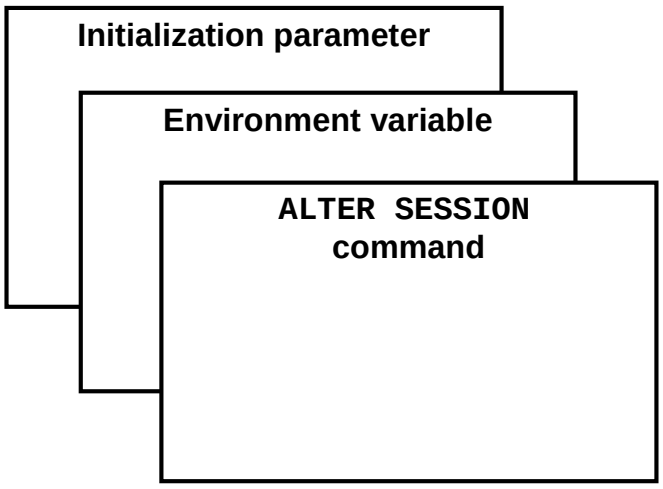
For performance concerns, consider using a single-byte character set for the database character set, and SQL NCHAR data types using AL16UTF16 for multilingual data. There is a performance overhead for using a UTF-8 encoding, which is a variable-width format; fixed-width single byte and multibyte character sets perform more efficiently.

Choosing a Unicode Solution: Unicode Data Type (continued)

Better support for UTF-16 with windows clients

If your applications are written in Visual C/C++ or Visual Basic running on Windows, then you may want to use the SQL `NCHAR` data types because you can store UTF-16 data in these data types in the same way as you store it in the `wchar_t` buffer in Visual C/C++ and string buffer in Visual Basic. You can avoid buffer overflow in your client applications because the length of the `wchar_t` and string data types match the length of the SQL `NCHAR` data types in the database.

Specifying Language-Dependent Behavior



ORACLE

Specifying Language-Dependent Behavior

There are three ways to specify National Language Support (NLS) parameters:

- As initialization parameters on the server side to specify the default server environment. (These default settings have no effect on the client side.)
- As environment variables for the client to specify locale-dependent behavior overriding the defaults set for the server
- As the `ALTER SESSION` parameter to override the default set for the client or the server

Specifying Language-Dependent Behavior for the Server

- **NLS_LANGUAGE specifies:**
  - **The language for messages**
  - **Day and month names**
  - **Symbols for A.D., B.C., a.m., p.m.**
  - **The default sorting mechanism**
- **NLS_TERRITORY specifies:**
  - **Day and week numbering**
  - **Default date format, decimal character, group separator, and the default ISO and local currency symbols**

ORACLE

Specifying Language-Dependent Behavior for the Server

NLS initialization parameters

The NLS_LANGUAGE initialization parameter defines the value for language-dependent conventions, such as:

- Language used for Oracle messages
- Language used for day and month names and their abbreviations
- Symbols used for language-equivalents of a.m., p.m., A.D., and B.C.
- Default sorting sequence of character data

The NLS_TERRITORY initialization parameter defines values for territory-dependent conventions, which include:

- Default date format
- Decimal character and group separator
- Local currency symbol
- ISO currency symbol
- ISO week number calculation
- Week start day

Note: When the territory name contains a space, as in The Netherlands, the territory name should be enclosed in double quotes, for example “The Netherlands.”

Dependent Language and Territory Default Values

Parameter	Values
NLS_LANGUAGE NLS_DATE_LANGUAGE NLS_SORT	AMERICAN AMERICAN BINARY
NLS_TERRITORY NLS_CURRENCY NLS_ISO_CURRENCY NLS_DATE_FORMAT NLS_NUMERIC_CHARACTERS	AMERICA $ AMERICA DD-MON-RR ,.

Dependent Language and Territory Default Values

NLS initialization parameters

The NLS_LANGUAGE initialization parameter determines the default values of the following parameters:

Column	Description
NLS_DATE_LANGUAGE	Explicitly changes the language for day and month names and abbreviations and spelled values of other date format elements
NLS_SORT	Changes the linguistic sort sequence that the Oracle server uses to sort character values. (The sort value must be the name of a linguistic sort sequence.)

Dependent Language and Territory Default Values (continued)

NLS initialization parameters (continued)

NLS_TERRITORY determines the default values for the following parameters:

Column	Description
NLS_CURRENCY	Explicitly specifies a new local currency symbol
NLS_ISO_CURRENCY	Explicitly specifies the territory whose ISO currency symbol should be used
NLS_DATE_FORMAT	Explicitly specifies a new default date format. (The value must be a date format model.)
NLS_NUMERIC_CHARACTERS	Explicitly specifies a new decimal character and group separator

Dual currency support for the euro

On January 1, 1999, the new currency of the European union, the euro, made its debut. To support the new European Union currency, dual currency support has been added for given territories. A NLS_DUAL_CURRENCY initialization parameter sets an alternate currency symbol for the user session.

The following territories have the euro symbol added for dual currency support:

Austria	Italy
Belgium	Luxembourg
Denmark	Netherlands
Finland	Portugal
France	Spain
Germany	Sweden
Greece	United Kingdom
Ireland	

The ISO character sets, such as WE8ISO8859P15 and MS Code Page WE8MSWIN1252, have specified code points for the euro symbol.

Specifying Language-Dependent Behavior for the Session

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

ORACLE

Specifying Language-Dependent Behavior for the Session

The environment variable NLS_LANG

The desired cultural convention for an individual user is specified with the NLS_LANG environment. The value of NLS_LANG overrides any values of the NLS initialization parameters.

Each component controls a subset of NLS features:

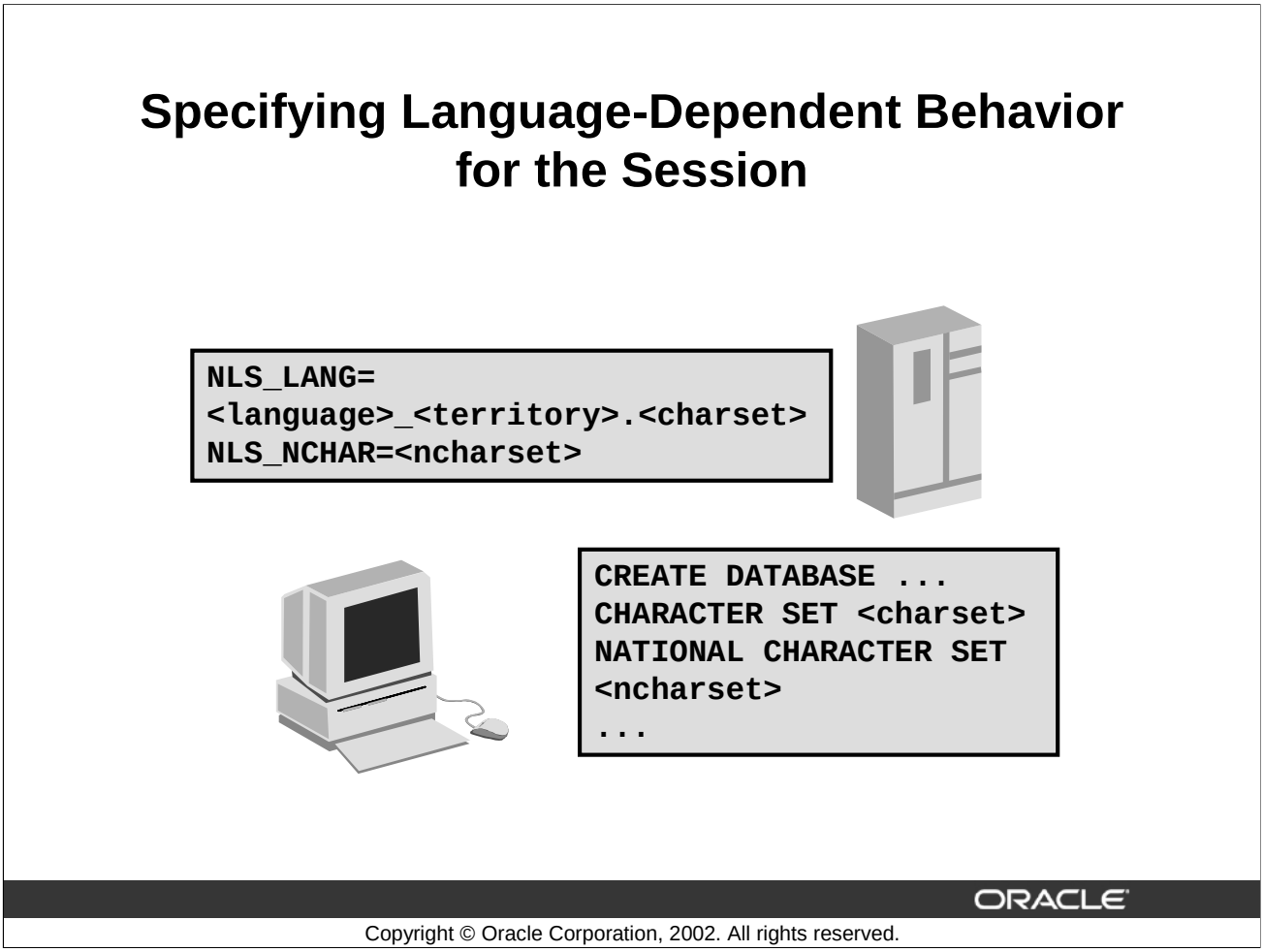
```
NLS_LANG=<language>_<territory>.<charset>
```

where:

Language: Overrides the value of NLS_LANGUAGE and controls the same features as NLS_LANGUAGE

Territory: Overrides the value of NLS_TERRITORY and controls the same features as NLS_TERRITORY

Charset: Specifies the character encoding scheme used by client application (usually that of the user's terminal)



Specifying Language-Dependent Behavior for the Session (continued)

The environment variable NLS_LANG (continued)

NLS_LANG defines a client terminal's character encoding scheme. Different clients can use different encoding schemes. Data passed between the client and the server is converted automatically between the two encoding schemes. The database encoding scheme should be a superset, or equivalent, of all the client encoding schemes. The conversion is transparent to the client application.

When the database character set and the client character set are the same, Oracle assumes that the data being sent or received is of the same character set, so no validations or conversions are performed. Although the benefit of this scenario is better performance, misuse can lead to possible data inconsistency problems, such as storing data from another character set that is different from the database character set.

For example, your database character set is US7ASCII and you are using Simplified Chinese Windows as your client terminal. By setting NLS_LANG to SIMPLIFIED CHINESE_HONGKONG.US7ASCII as the client character set, it is possible for you to store multibyte Simplified Chinese characters inside a single byte database. This means that Oracle will treat these characters as single-byte US7ASCII characters, and therefore, all SQL string manipulation functions such as SUBSTR or LENGTH will be based on bytes rather than characters. All of your non-ASCII characters could be lost following an export and import into another database.

Specifying Language-Dependent Behavior for the Session (continued)

Additional environment variables

All NLS initialization parameters are available as environment variables, making it possible to specify individual NLS characteristics for each client.

In addition, NLS_CALENDAR can be used to specify which calendar system the Oracle server uses; for example, Gregorian, Persian, or Thai Buddha.

The following variables can be set only in the client environment:

- NLS_CREDIT
- NLS_DEBIT
- NLS_DISPLAY
- NLS_LANG
- NLS_LIST_SEPARATOR
- NLS_MONETARY

Note: A description of these parameters can be found in the *Oracle9i Globalization Support Manual*.

If the environment variable ORA_NLS33 is set to an invalid directory, it is possible to create the database only with the US7ASCII default character set. ORA_NLS33 should be set on UNIX as follows:

```
$ORACLE_HOME/ocommon/nls/admin/data
```

This is also the default setting if ORA_NLS33 is not set.

Specifying Language-Dependent Behavior for the Session

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

```
DBMS_SESSION.SET_NLS('NLS_DATE_FORMAT',
'''DD.MM.YYYY''') ;
```

ORACLE

Specifying Language-Dependent Behavior for the Session

Changing NLS parameters

Change individual NLS characteristics for a session with the `ALTER SESSION` command. All environment variables that can be set on both the client and server sides can also be modified by issuing the `ALTER SESSION` command.

In the above example, the date format is changed for the session.

Also, tools such as SQL*Plus read the environment variables and issue the corresponding `ALTER SESSION` command.

In addition to explicitly issuing `ALTER SESSION` commands, there is a `DBMS_SESSION.SET_NLS` database package that takes the name and the value of the parameter.

Linguistic Sorting

Three types of sorting:

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

ORACLE

Linguistic Sorting

A binary sort is a conventional sorting mechanism by which letters are sorted according to the binary values that are used to encode the characters. The alphabetic position of a character may vary for different languages.

With monolingual sorting, Oracle makes two passes when comparing strings in monolingual sorts. The first pass is to compare the major value of entire string from the major table and the second pass is to compare the minor value from the minor table. Usually, letters with the same appearance will have the same major value. Oracle defines letters with diacritic and case differences for the same major value but different minor values. Although this provides better sorting than binary, it is still limited.

For multilingual sorting, Oracle provides a sorting mechanism based on an ISO standard (ISO14651) and the Unicode 3.0 standard. This enables each language to properly sort each encoded character.

Oracle currently supports 84 linguistic sorts (68 monolingual and 13 multilingual). For a complete list, refer to the *Oracle9i Globalization Support Manual*.

NLS Sorting

- **NLS_SORT specifies the type of sort for character data:**
  - **Is defined by the NLS_LANG environment variable**
  - **Can be overridden at the session level**
- **NLSSORT function:**
  - **Specifies the type of sort for character data**
  - **Allows sorts to be defined at the query level**

ORACLE

NLS Sorting

To overcome the limitations of binary sorting, the Oracle server provides linguistic sorts by setting the NLS_SORT parameter.

How NLS affects sorts

The following example demonstrates all three types of sorts:

- Binary
- Monolingual, using French
- Multilingual, using French_M

The example creates a list of four French words, base on table list.

```
SQL> CREATE TABLE list ( num NUMBER(1),
 2   word VARCHAR2(5),
 3   def VARCHAR2(7)
 4   );
Table created.
```

NLS Sorting (continued)

How NLS affects sorts (continued)

```
SQL> INSERT INTO list VALUES (1, 'gelée', 'frost');
1 row created.
SQL> INSERT INTO list VALUES (2, 'gelé', 'frozen');
1 row created.
SQL> INSERT INTO list VALUES (3, 'gèle', 'freezes');
1 row created.
SQL> INSERT INTO list VALUES (4, 'gelez', 'freeze');
1 row created.
```

First, NLS_SORT is set to BINARY. Notice that in BINARY, e is sorted before è. This occurs because e has a binary value lower than è in this character encoding.

```
SQL> ALTER SESSION SET NLS_SORT = BINARY;
Session altered.
SQL> SELECT   num, word, def
 2   FROM     list
 3   ORDER BY word;
NUM WORD  DEF
--- ----- -------
 4  gelez freeze
 2  gelé  frozen
 1  gelée frost
 3  gèle  freezes
```

Next, NLS_SORT is set to French. French is a monolingual sort. Because monolingual sorting is limited to a two-pass sort, it cannot encompass all of nuances of the French language. For example, the French sort letters from left to right and accents from right to left. This can be seen in the multilingual sort.

```
SQL> ALTER SESSION SET NLS_SORT = FRENCH;
Session altered.
SQL> SELECT   num, word, def
 2   FROM     list
 3   ORDER BY word;
NUM WORD  DEF
--- ----- -------
 2  gelé  frozen
 3  gèle  freezes
 1  gelée frost
 4  gelez freeze
```

NLS Sorting (continued)

How NLS affects sorts (continued):

Finally, there is the `French_M` multilingual sort. Notice the differences between this sort and the previous sort.

```
SQL> ALTER SESSION SET NLS_SORT = FRENCH_M;
Session altered.
SQL> SELECT   num, word, def
 2   FROM     list
 3   ORDER BY word;
NUM WORD  DEF
--- ----- -------
 3  gèle  freezes
 2  gelé  frozen
 1  gelée frost
 4  gelez freeze
```

`NLSSORT` allows sorting to be defined at the query level. The following example sets `NLS_SORT` to `BINARY` at the session level but changes the sort at the query level. Notice that the results are the same as in the previous example.

```
SQL> ALTER SESSION SET NLS_SORT=BINARY;
Session altered.
SQL> SELECT     num, word, def
 2   FROM       list
 3   ORDER BY NLSSORT(word,'NLS_SORT=FRENCH_M');

NUM WORD  DEF
--- ----- -------
 3  gèle  freezes
 2  gelé  frozen
 1  gelée frost
 4  gelez freeze
```

Using NLS Parameters in SQL Functions

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

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

ORACLE

Using NLS Parameters in SQL Functions

SQL character functions support single-byte and multibyte characters.

Some SQL functions require NLS parameters to be specified explicitly as part of their parameter list. Therefore SQL functions can override the behavior specified by the environment.

Using NLS Parameters in SQL Functions (continued)

Examples

```
SQL> SELECT TO_CHAR(hire_date, 'DD.Mon.YYYY',
  2 'NLS_DATE_LANGUAGE=FRENCH') AS "Hire Date"
  3 FROM employees;

Hire Date
-----------
15.Dec.1997
03.Nov.1998
11.Nov.1997
19.Mar.1999
24.Jan.2000
23.Fev.2000
24.Mar.2000
21.Avr.2000
11.Mar.1997
23.Mar.1998
24.Jan.1998
23.Fev.1999
24.Mar.1999
21.Avr.2000
11.Mai.1996
19.Mar.1997
24.Mar.1998
23.Avr.1998
24.Mai.1999
04.Jan.2000
```

Using NLS Parameters in SQL Functions (continued)

Examples

```
SQL> SELECT last_name,
  2  TO_CHAR(salary,'99G999D99',
  3  'NLS_NUMERIC_CHARACTERS='',.''')
  4  FROM employees;

LAST_NAME         TO_CHAR(SA
----------        ------------
Doran             7.500,00
Sewall            7.000,00
Vishney           10.500,00
Greene            9.500,00
Marvins           7.200,00
Lee               6.800,00
Ande              6.400,00
Banda             6.200,00
Ozer              11.500,00
Bloom             10.000,00
Fox               9.600,00
Smith             7.400,00
Bates             7.300,00
Kumar             6.100,00
Abel              11.000,00
Hutton            8.800,00
Taylor            8.600,00
Livingston        8.400,00
Grant             7.000,00
```

Using NLS Parameters in SQL Functions (continued)

Examples

The following SQL functions use NLS parameters:

Function	NLS Parameter
TO_DATE	NLS_DATE_LANGUAGE NLS_CALENDAR
TO_NUMBER	NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY
TO_CHAR	NLS_DATE_LANGUAGE NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY NLS_CALENDAR
NLS_UPPER, NLS_LOWER, NLS_INITCAP, NLSSORT	NLS_SORT

Several format mask elements have been defined for functions such as TO_CHAR, TO_DATE, and TO_NUMBER.

Number format mask elements

- “D” for decimal separator
- “G” for group (thousands) separator
- “L” for local currency symbol
- “C” for local ISO currency symbol
- “U” for the dual currency symbol, used for the euro

Date format mask elements

- “RM, rm” for Roman month number
- “IW” for ISO week number
- “IYYY, IYY, IY,” and “I” for ISO year

Linguistic Index Support

- **Linguistic indexing**
- **High performance with local sorting:**

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

- **NLS_COMP parameter for linguistic comparisons**

ORACLE

Linguistic Index Support

Functional indexes can be specialized to create linguistically sorted indexes. The SQL function NLSSORT returns the string of bytes used to sort the first parameter in the given linguistic sorting sequence. In this example, an index is created on WORD that is sorted according to the French_M sorting order. This enables you to perform index-based queries on data that is sorted according to the rules of each language.

Linguistic behavior of comparison operators

NLS_COMP is a dynamic initialization parameter that controls how comparison operators such as <, >, and = handle linguistic ordering. When set to BINARY (the default), comparison is based on the binary value of the string. When set to ANSI, comparison operators use linguistic sorting sequences to determine the outcome of the operation according to the NLS_SORT session parameter.

Import and Loading Data Using NLS

- **Data is converted from the export file character set to the database character set during the import.**
- **SQL*Loader:**
  - **Conventional Path: Data is converted into the session character set specified by `NLS_LANG`.**
  - **Direct Path: Data is converted directly into the database character set.**

ORACLE

Import and Loading Data Using NLS

The export file is exported with the source database character set. During the import, the data is automatically converted from the character set of the export file to the target database character set.

SQL*Loader also has the capability to convert data from the data file character set to the database character set.

When using Conventional path, data is converted into the session character set specified by the `NLS_LANG` parameter for that session.

With Direct path, data is converted directly into the database character set.

The control file tells SQL*Loader how to interpret the data file.

The parameter character set tells what character set is used in each data file.

Example

```
$sqlldr control=utl1case.ctl characterset=WE8ISO9959P1
```

Using NLS Parameters in SQL Functions

- **Character set scanner:**
  - **Scans the database to determine whether the character set can be changed**
  - **Provides reports that detail possible problems and fixes**
- **Oracle locale builder:**
  - **Easy to use graphical interface**
  - **For viewing, modifying, and creating locale definitions**

ORACLE

Using NLS Parameters in SQL Functions

Character set scanner

The character set scanner is a command line utility which assists in character set conversion. The scanner identifies areas of possible character set conversion and truncation of data, the amount of effort required, and column widths which should be expanded. It provides assessment of feasibility, reports potential migration issues, checks all character data, and generates a summary of database scan. The character set scanner should be used prior to any character set conversion.

Oracle locale builder

The Oracle9*i* server provides an extensive set of locale definitions including languages, territories, character sets, and linguistic sorts. If you must customize any of these existing locale definitions, or create a new one, the new Oracle Locale Builder provides an easy-to-use graphical user interface through which you can easily view, customize, and define the various locales.

Obtaining Character Set Information

NLS_DATABASE_PARAMETERS:

- **PARAMETER (NLS_CHARACTERSET, NLS_NCHAR_CHARACTERSET)**
- **VALUE**

ORACLE

Obtaining Character Set Information

View the database and the national character set with the following query:

```
SQL> SELECT parameter, value
  2   FROM nls_database_parameters
  3   WHERE parameter LIKE '%CHARACTERSET%';
PARAMETER                VALUE
-----------------------  -------------
NLS_CHARACTERSET         WE8ISO8859P1
NLS_NCHAR_CHARACTERSET   AL16UTF16
2 rows selected.
```

Obtaining NLS Settings Information

- **NLS_INSTANCE_PARAMETERS:**
  - **PARAMETER (initialization parameters that have been explicitly set)**
  - **VALUE**
- **NLS_SESSION_PARAMETERS:**
  - **PARAMETER (session parameters)**
  - **VALUE**

ORACLE

Obtaining NLS Settings Information

This view displays only values for parameters that have been explicitly set in the `init<SID>.ora` file.

```
SQL> SELECT * FROM nls_instance_parameters;
PARAMETER                      VALUE
------------------------------ ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_SORT
NLS_DATE_LANGUAGE
NLS_DATE_FORMAT
NLS_CURRENCY
NLS_NUMERIC_CHARACTERS
NLS_ISO_CURRENCYNLS_CALENDAR
NLS_TIME_FORMAT
NLS_TIMESTAMP_FORMAT
```

Obtaining NLS Settings Information (continued)

```
NLS_TIME_TZ_FORMAT
NLS_TIMESTAMP_TZ_FORMAT
NLS_DUAL_CURRENCY
NLS_COMP
NLS_LENGTH_SEMANTICS    BYTE
NLS_NCHAR_CONV_EXCP     FALSE

17 rows selected.
```

The following view shows session parameters:

```
SQL> SELECT * FROM nls_session_parameters;
PARAMETER                      VALUE
------------------------------ ----------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY               AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-RR
NLS_DATE_LANGUAGE              AMERICAN
NLS_SORT                       BINARY
NLS_TIME_FORMAT                HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT           DD-MON-RR HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT             HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT        DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY              $
NLS_COMP                       BINARY
NLS_LENGTH_SEMANTICS           BYTE
NLS_NCHAR_CONV_EXCP            FALSE

17 rows selected.
```

Obtaining NLS Settings Information

- **V$NLS_VALID_VALUES:**
  - **PARAMETER (LANGUAGE, SORT, TERRITORY, CHARACTERSET)**
  - **VALUE**
- **V$NLS_PARAMETERS:**
  - **PARAMETER (NLS session parameters, NLS_CHARACTERSET)**
  - **VALUE**

ORACLE

Obtaining NLS Settings Information

List all valid values for NLS parameters.

```
SQL>  SELECT * FROM v$nls_valid_values
  2   WHERE parameter='LANGUAGE';
PARAMETER    VALUE
---------    --------------
LANGUAGE     AMERICAN
LANGUAGE     GERMAN
LANGUAGE     FRENCH
LANGUAGE     CANADIAN FRENCH
LANGUAGE     SPANISH
LANGUAGE     ITALIAN
LANGUAGE     DUTCH
LANGUAGE     SWEDISH
LANGUAGE     NORWEGIAN
...
```

Obtaining NLS Settings Information (continued)

Note: The V$NLS_VALID_VALUES view shows the contents of the NLS data boot file. This returns a list of all character sets, languages, linguistic sorts and territory definitions that are shipped with a given database release. This list may contain desupported or internally used definitions.

To display current values of NLS parameters.

```
SQL> SELECT * FROM v$nls_parameters;
PARAMETER                      VALUE
------------------------------ ----------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY                AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-RR
NLS_DATE_LANGUAGE              AMERICAN
NLS_CHARACTERSET               WE8ISO8859P1
NLS_SORT                       BINARY
NLS_TIME_FORMAT                HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT           DD-MON-RR HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT             HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT        DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY              $
NLS_NCHAR_CHARACTERSET         AL16UTF16
NLS_COMP                       BINARY
NLS_LENGTH_SEMANTICS           BYTE
NLS_NCHAR_CONV_EXCP            FALSE

19 rows selected.
```

Note: Various views will contain a new column, CHARACTER_SET_NAME, which shows the name of the character set: CHAR_CS for database character set and NCHAR_CS for national character set.

For example, DBA_TAB_COLUMNS builds this column from COL$.

Summary

In this lesson, you should have learned how to:

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

ORACLE

Practice 20 Overview

This practice covers the following topics:

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

ORACLE

Practice 20 Overview

Note: Practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 20: Using Globalization Support

1. Check the database and the national character set.
2. Which are the valid values for the database character set?
3. Check that all dates in this session are displayed using a four-digit year. Change `NLS_LANGUAGE` to `FRENCH`. Select `sysdate` from `DUAL`.

How to Create an Oracle9*i* Database in a UNIX Environment

ORACLE

Introduction

Steps to Create a Database

There are six steps to creating a useable database: three steps to create the database and three additional steps to make it usable.

1. Set the operating system environment variables `ORACLE_HOME, ORACLE_SID, PATH,` and `LD_LIBRARY_PATH`.
2. Edit/Create an `init`*`sid`*`.ora` parameter file.
3. Execute the `CREATE DATABASE` command in SQL*Plus.
4. Run the required `catalog.sql` and `catproc.sql` scripts.
5. Run the `pupbld.sql` script.
6. Create a tablespace for user data and any other tablespaces that might be required to meet the needs of the database.

Note: The contents of this document are written with the assumption that the Oracle9*i* Server has been installed in an `ORACLE_HOME`. Installation of the Oracle9*i* Server is not covered in this appendix.

Setting the Environment

Before a database is created, the UNIX environment must be configured and the Oracle9*i* server must have already been installed.

Four environment variables must be set: ORACLE_HOME, ORACLE_SID, PATH, LD_LIBRARY_PATH.

ORACLE_HOME is the full path to the top directory in which the Oracle9*i* Server is installed. The directory for ORACLE_HOME should be supplied by the person who installed the server, usually the UNIX administrator or the DBA.

ORACLE_SID is a user-definable name assigned to an instance of a database. The ORACLE_SID (system identifier) is used by the operating system to distinguish different database instances running on the machine.

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

LD_LIBRARY_PATH defines the directories in which required library files are stored.

Example

Bourne or Korn shell:

```
$ ORACLE_HOME=/u01/oracle9i/product/9.0.1; export ORACLE_HOME
$ ORACLE_SID=db01; export ORACLE_SID
$ PATH=/usr/bin:/usr/ccs/bin:$ORACLE_HOME/bin; export PATH
$ LD_LIBRARY_PATH=/usr/lib:$ORACLE_HOME/lib; export
  LD_LIBRARY_PATH
```

C shell:

```
% setenv ORACLE_HOME /u01/oracle9i/product/9.0.1
% setenv ORACLE_SID db01
% setenv PATH $PATH:$ORACLE_HOME/bin
% setenv LD_LIBRARY_PATH /usr/lib:$ORACLE_HOME/lib
```

Editing initsid.ora

Edit/Create initsid.ora

The initsid.ora file, a user-configured text file, is read each time the database starts. The parameters in the file initialize the database settings. The parameter settings in the initsid.ora file affect not only the database at startup but also how the database is created. Prior to creating the database, the initsid.ora file must be configured.

When the Oracle9*i* server is installed, a sample init.ora file is placed in $ORACLE_HOME/dbs. Keep this file as a backup and do not modify it. Create a copy of the file containing the name of the ORACLE_SID.

Example

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

The sample init.ora file has many comments containing suggestions for parameter settings.

The parameters in the initsid.ora need not be in any order. However, if a parameter is listed more than once, the last setting is the one used. Oracle9*i* Reference suggests placing the parameters in alphabetical order to prevent duplication.

The following parameters must be configured: db_name, control_files, background_dump_dest, user_dump_dest, core_dump_dest, and undo_management.

The background_dump_dest, user_dump_dest, and core_dump_dest parameters are set to the full path locations where trace files are to be placed:

- core_dump_dest — contains core dumps generated by the database
- user_dump_dest — contains user trace files
- background_dump_dest — contains trace files for the background processes and the alert.log.

The db_name is the name of the database, which is used for a purpose different from ORACLE_SID. The ORACLE_SID is the name that is used to designate an instance of the database. Many times the db_name and ORACLE_SID are the same but it is not a requirement. The db_name in the initsid.ora must be identical (case sensitive) to the name of the database that is used in the CREATE DATABASE command when the database is created.

The control_files initialization parameter designates the full path and filename of each control file for the database. For the creation of the database, it identifies the control files that must be created.

The undo_management initialization parameter determines whether the Oracle server automatically or the DBA manually handles undo data. Set undo_management to AUTO in the initialization file.

At the end of Appendix A is an example initdb01.ora file.

Creating the Database

After setting the environment and configuring the initsid.ora, the database can be created. An Oracle database is created by executing a CREATE DATABASE command. The CREATE DATABASE command, specifies the number and location of the log files, the location and size of the SYSTEM tablespace, UNDO tablespace, and TEMP tablespace, and the character set for the database (this is not an exhaustive list).

The Oracle utility SQL*Plus is used when creating the database. The UNIX executable for SQL*Plus is sqlplus.

When creating a database, the Oracle9*i* server is only aware of the SYS user and the SYSDBA role. To create a database, you must connect to SQL*Plus as the user SYS and the role SYSDBA. This can be accomplished in one of two methods:

1. If the UNIX user who usually connects to SQL*Plus is part of the administrator's group, defined during the installation of the Oracle9*i* server, then the following syntax can be used:

   ```
   $ sqlplus  '/ as sysdba '
   ```

   or

   ```
   $ sqlplus /nolog
   SQL> connect / as sysdba
   ```

2. If the UNIX user who connects to SQL*Plus is not part of the administrator's group, then a password file must be created for the database by an administrator, and the password assigned to SYS in the password file must be used. The following syntax assumes the password assigned to SYS in the password file is oracle:

   ```
   $ sqlplus 'sys/oracle as sysdba'
   ```

   or

   ```
   $ sqlplus /nolog
   SQL> connect sys/oracle as sysbda
   ```

The steps to create and execute the SQL statement are:

1. Create a SQL script that contains the CREATE DATABASE command. (At the end of Appendix A is a sample create database script.)
2. Connect to SQL*Plus as SYS AS SYSDBA in one of the two methods shown above.
3. Start up the database in NOMOUNT mode.
4. Execute the SQL script.

Creating the Database (continued)

Example

```
% sqlplus 'sys/oracle as sysdba'
SQL> startup nomount
ORACLE instance started.
Total System Global Area   21790532 bytes
Fixed Size                   278340 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
SQL> @crdbdb01.sql
SQL> CREATE DATABASE db01
  2     LOGFILE
  3       GROUP 1 ('$HOME/ORADATA/u03/log_01_01_db01.rdo') SIZE
          10M,
  4       GROUP 2 ('$HOME/ORADATA/u03/log_02_01_db01.rdo') SIZE
          10M
  5     DATAFILE '$HOME/ORADATA/u01/system_01_db01.dbf' SIZE 100M
  6        AUTOEXTEND ON NEXT 5M MAXSIZE 150M
  7     DEFAULT TEMPORARY TABLESPACE temp
  8        TEMPFILE '$HOME/ORADATA/u02/temp_01_db01.dbf' SIZE 15M
  9        AUTOEXTEND ON NEXT 5M MAXSIZE 30M
 10     CHARACTER SET WE8ISO8859P1
 11     NATIONAL CHARACTER SET AL16UTF16
 12  ;
Statement processed.
```

Running Scripts

The `catalog.sql` and `catproc.sql` scripts that are located in `$ORACLE_HOME/rdbms/admin`, must be run after the database is created. The `catalog.sql` script creates the data dictionary views and `catproc.sql` creates the packages and procedures required to use PL/SQL.

Both scripts must be run as SYS. Before executing the scripts, make sure the database is open.

Example

```
% sqlplus /nolog
SQL> CONNECT / AS SYSDBA
SQL> @$ORACLE_HOME/rdbms/admin/catalog.sql
SQL> @$ORACLE_HOME/rdbms/admin/catproc.sql
```

On a system that is not busy, the total time for both scripts to complete is between 35 and 65 minutes.

After these scripts have run, verify that the objects are valid. The following query returns any invalid objects.

```
SQL> SELECT  owner,object_name,object_type
  2  FROM     dba_objects
  3  WHERE    status = 'INVALID'
  4  ORDER BY owner,object_type,object_name;
```

Running `pupbld.sql`

The `pupbld.sql` script that is located in the `$ORACLE_HOME/sqlplus/admin` directory creates the Product User Profile table and related procedures. Running this script, among other purposes, prevents a warning message each time a user connects to SQL*Plus. The script must be run as user SYSTEM.

```
$ sqlplus system/manager
SQL> @$ORACLE_HOME/sqlplus/admin/pupbld.sql
```

Creating Tablespaces

Create the other tablespaces that are required for the installation.

In a database installation, the following tablespaces are usually created:

- `users` user data
- `tools` objects created by the user `SYSTEM` (optional)

These should be created.

Example

```
SQL> create tablespace USERS
  2  datafile '$HOME/ORADATA/u03/users_01_db01.dbf' SIZE 25M
  3  PERMANENT
  4  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K
  5  SEGMENT SPACE MANAGEMENT auto;
```

Summary

- Set `ORACLE_HOME`, `ORACLE_SID`, `PATH`, and `LD_LIBRARY_PATH`
- Edit the `initsid.ora`.
- Execute the `CREATE DATABASE` command.
- Run the `catalog.sql` and `catproc.sql` scripts.
- Run the `pupbld.sql` script.
- Create the tablespaces that are required for the database.

Sample `initdb01.ora`

```
background_dump_dest=$HOME/ADMIN/BDUMP
compatible=9.0.0
control_files=$HOME/ORADATA/u01/ctrl_01_sid.ctl
core_dump_dest=$HOME/ADMIN/CDUMP
db_block_size=4096
db_cache_size=4M
db_domain=world
db_name=db01
global_names=TRUE
instance_name=db01
max_dump_file_size=10240
remote_login_passwordfile=exclusive
service_names=db01
shared_pool_size=8M
undo_management=AUTO
user_dump_dest=$HOME/ADMIN/UDUMP
```

Sample Script to a Create Database

```
CREATE DATABASE db01
   LOGFILE
      GROUP 1 ('$HOME/ORADATA/u03/log_01_01_db01.rdo') SIZE 10M,
      GROUP 2 ('$HOME/ORADATA/u03/log_02_01_db01.rdo') SIZE 10M
   DATAFILE '$HOME/ORADATA/u01/system_01_db01.dbf' SIZE 100M
      AUTOEXTEND ON NEXT 5M MAXSIZE 150M
   DEFAULT TEMPORARY TABLESPACE temp
      TEMPFILE '$HOME/ORADATA/u02/temp_01_db01.dbf' SIZE 15M
      AUTOEXTEND ON NEXT 5M MAXSIZE 30M
   CHARACTER SET WE8ISO8859P1
   NATIONAL CHARACTER SET AL16UTF16
;
```

B

Manually Managing Undo Data (Rollback Segments)

ORACLE

Objectives

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

- **Create rollback segments using appropriate storage settings**
- **Maintain rollback segments**
- **Plan the number and size of rollback segments**
- **Troubleshoot common rollback segment problems**

ORACLE

Creating Rollback Segments

```
CREATE ROLLBACK SEGMENT rbs01
  TABLESPACE rbs
  STORAGE (
    INITIAL    100K
    NEXT      100K
    MINEXTENTS   20
    MAXEXTENTS   100
    OPTIMAL    2000K );
```

ORACLE

Creating Rollback Segments

Use the following command to create a rollback segment:

```
CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment
    [TABLESPACE tablespace]
    [STORAGE (     [INITIAL     integer[K|M]]
              [NEXT        integer[K|M]]
              [MINEXTENTS  integer]
              [MAXEXTENTS {integer|UNLIMITED}]
              [OPTIMAL    {integer[K|M]|NULL}]
         )
]
```

Creating Rollback Segments (continued)

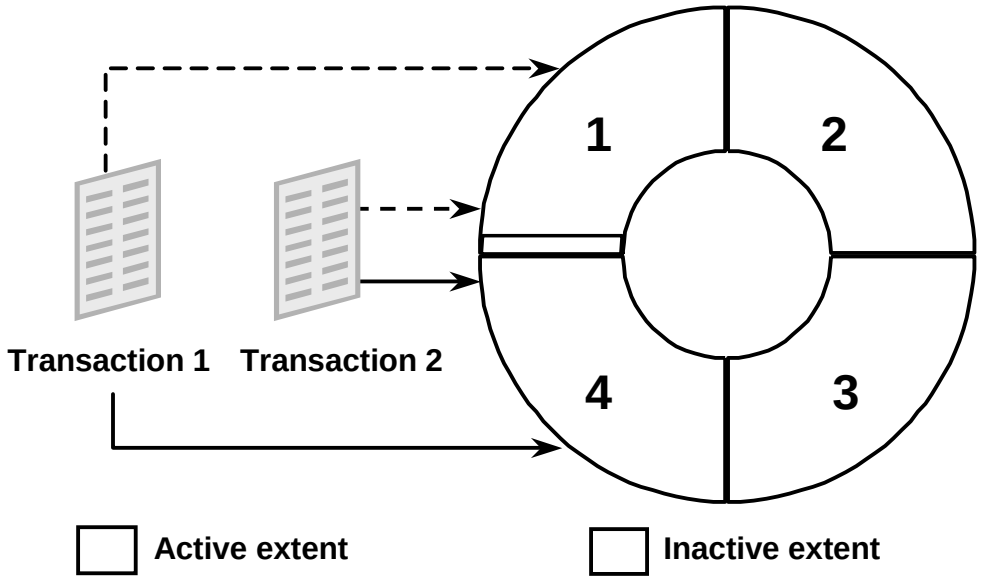
Restrictions

- A rollback segment can be specified as either public or private (the default) at the time of creation and cannot be changed.
- For a rollback segment, `MINEXENTS` must be at least two.
- `PCTINCREASE` cannot be specified for a rollback segment and is always set to 0.
- `OPTIMAL`, if specified, must be at least equal to the initial size of the rollback segment, which is the space used by the number of extents defined by `MINEXTENTS`.

Guidelines

- Always use `INITIAL = NEXT` for rollback segments to ensure that all extents are of the same size.
- Set the `OPTIMAL` value to minimize the allocation and deallocation of rollback segment extents.
- Avoid setting `MAXEXTENTS` to `UNLIMITED`. This could cause unnecessary extension of a rollback segment and possibly of data files due to a program error.
- Always place rollback segments in a separate, exclusive tablespace to minimize contention and fragmentation.

Transactions and Rollback Segments



ORACLE

Allocation of a Rollback Segment

When a transaction begins, a rollback segment must be assigned to this transaction. A transaction can request a specific rollback segment using the following command:

```
SET TRANSACTION USE ROLLBACK SEGMENT rollback_segment
```

If no such request is made, the Oracle server chooses the rollback segment with the fewest transactions and assigns it to the transaction.

Using Extents

Transactions use extents of a rollback segment in a sequential, circular fashion, moving from one to the next after the current extent is full. A transaction writes an entry to its current location in the rollback segment and advances the current pointer by the size of the entry.

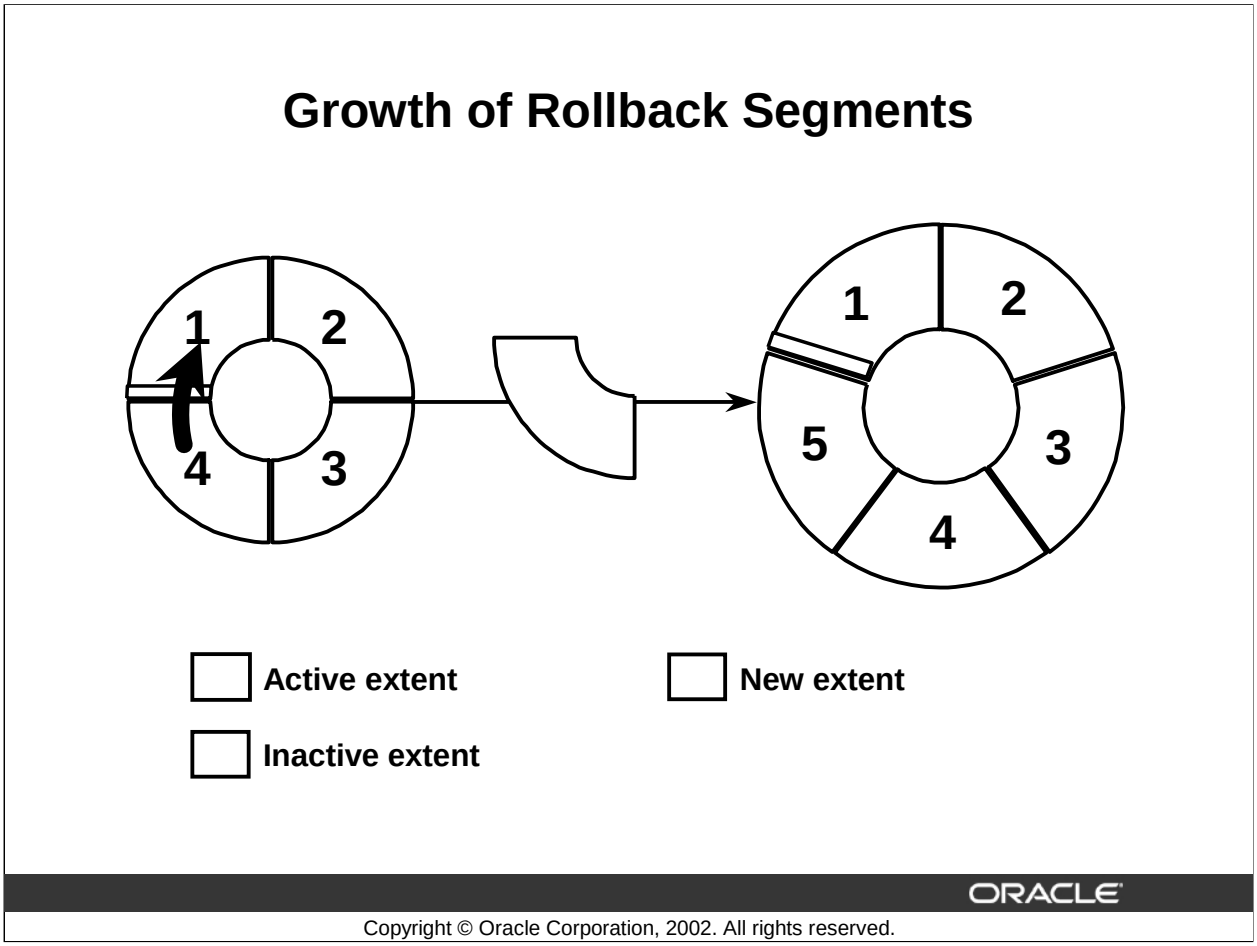
Although more than one transaction can write to the same extent of a rollback segment, each rollback segment block contains information from one and only one transaction.

Allocation of a Rollback Segment (continued)

Example

In the example in the slide, two transactions have been assigned to a rollback segment, which has four extents.

1. When the transactions commence, they begin writing to Extent 3 of the rollback segment.
2. As the two transactions generate more rollback information, they continue to write into Extent 3.
3. When Extent 3 is full, the transactions write to the next extent in the ring, which is Extent 4. When transactions start writing to a new extent as in this step, it is called a wrap.
4. When the last extent for the rollback segment (Extent 4) is full, the transactions can use the first in the ring (Extent 1) if it is free or inactive. An extent is free or inactive only if there are currently no active transactions using the extent; that is, all transactions that wrote to the extent have completed.



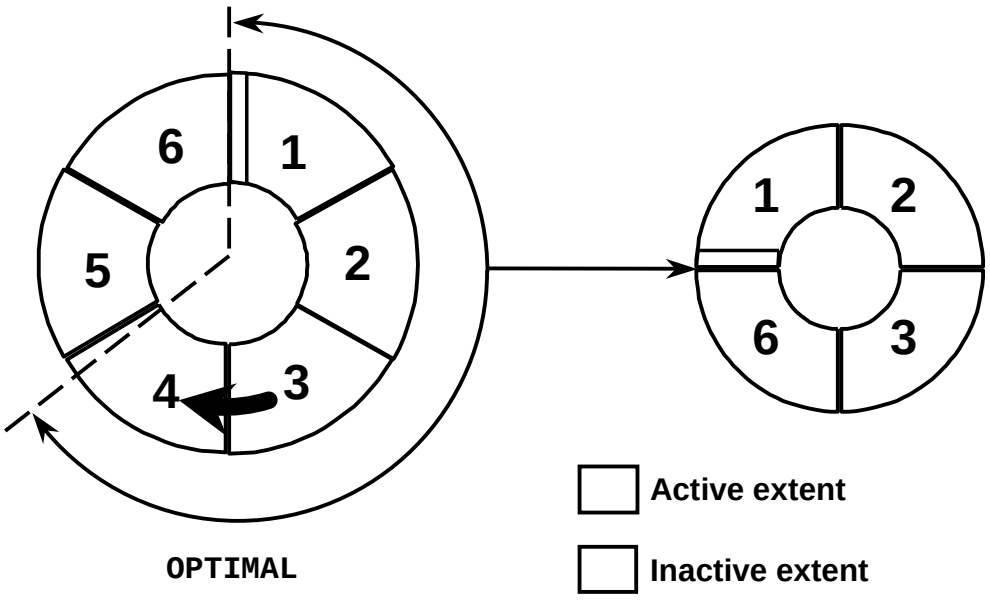
Growth of Rollback Segments

The pointer or the head of the rollback segment moves to the next extent when all blocks of the current extent are used and a transaction requires another block for more space. When the last extent is full, the pointer moves to the front of the first extent.

The pointer can move to the next extent only if that extent has no active transactions. The pointer cannot skip over an extent. If the next extent is being used, the transaction allocates an additional extent for the rollback segment. This is called an *extend*.

A rollback segment can grow in this manner until it reaches the maximum number of extents specified by the MAXEXTENTS parameter.

Shrinkage of Rollback Segments



ORACLE

OPTIMAL Parameter

The OPTIMAL parameter specifies the size in bytes to which a rollback segment must shrink, if possible. Specifying OPTIMAL minimizes the waste of space in a rollback segment. If the OPTIMAL parameter is specified, a rollback segment can release space when transactions that caused the growth are completed.

The deallocation of extents is not performed as soon as transactions are completed. The process of deallocating extents is performed only when the head moves from one extent to the next. Extents are deallocated if both of the following conditions are true:

- The current size of the rollback segment exceeds OPTIMAL.
- There are contiguous inactive extents.

The Oracle server tries to deallocate the size of the rollback segment until it is equal to OPTIMAL, but may have to stop short if the next extent to be deallocated is in use.

The Oracle server always deallocates the oldest inactive extents, because they are least likely to be used for read consistency.

Bringing Rollback Segments Online

- **Use the following command to make a rollback segment available:**

```
ALTER ROLLBACK SEGMENT rbs01 ONLINE;
```

- **Specify the following initialization parameter to ensure that rollback segments are brought online at startup:**

```
ROLLBACK_SEGMENTS=(rbs01, rbs02)
```

ORACLE

The `ALTER ROLLBACK SEGMENT` Command

When a rollback segment is created, it is offline and cannot be used. To make the rollback segment available for use by transactions, use the `ALTER ROLLBACK SEGMENT` command and bring it online.

Syntax

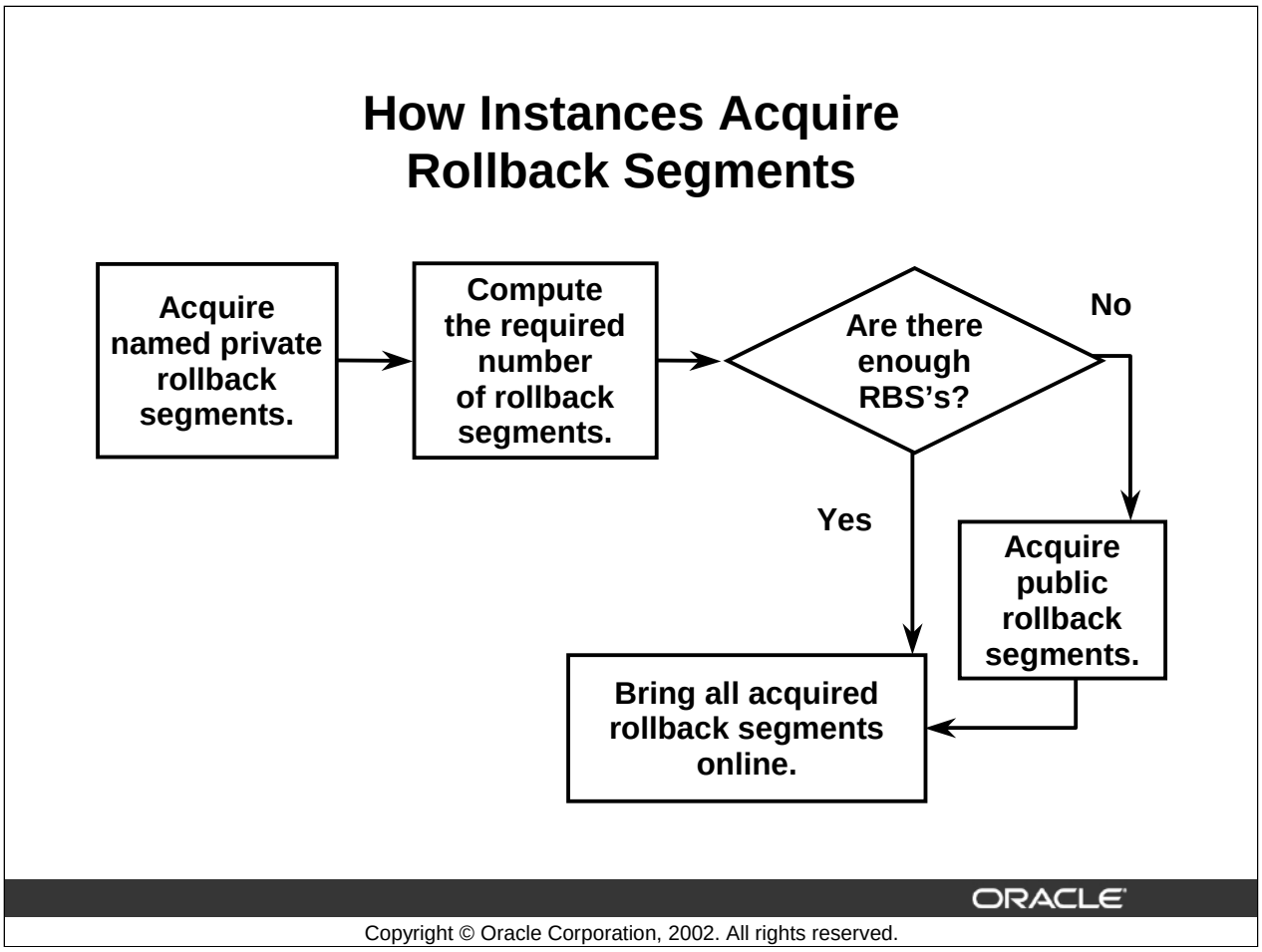
Use the following command to make a rollback segment available:

```
ALTER ROLLBACK SEGMENT rollback_segment ONLINE;
```

The number of rollback segments that can be brought online by an instance is limited by the `MAX_ROLLBACK_SEGMENTS` parameter. Set this value to one more than the number of non-`SYSTEM` rollback segments required for the instance.

A rollback segment is online only until the instance is shut down. To ensure that a rollback segment is always brought online by an instance, specify the name of the rollback segment in the parameter file as shown in the following example:

```
ROLLBACK_SEGMENTS=(rbs01, rbs02)
```



How Instances Acquire Rollback Segments

The following steps explain how rollback segments are acquired by an instance when it opens a database:

- The instance acquires all rollback segments that are named in the `ROLLBACK_SEGMENTS` initialization parameter.
- The `TRANSACTIONS init.ora` parameter is divided by the `TRANSACTIONS_PER_ROLLBACK_SEGMENT init.ora` parameter, and the result is the number of rollback segments that are needed by the instance. If this value is greater than the non-`SYSTEM` rollback segments already brought online by the instance, then the instance acquires additional public rollback segments to make up for the shortfall. If there are insufficient public rollback segments, the database is opened and available to users. No errors are generated.

Changing Rollback Segment Storage Settings

- **Use the `ALTER ROLLBACK SEGMENT` command.**
- **You can change `OPTIMAL` or `MAXEXTENTS`.**

```
ALTER ROLLBACK SEGMENT rbs01
  STORAGE( MAXEXTENTS 200 );
```

Changing Rollback Segment Storage

The storage parameters for a rollback segment can be changed using the `ALTER ROLLBACK SEGMENT` command:

```
ALTER ROLLBACK SEGMENT rollback_segment
[STORAGE ( [NEXT        integer[K|M]]
           [MINEXTENTS  integer]
           [MAXEXTENTS {integer|UNLIMITED}]
           [OPTIMAL    {integer[K|M]|NULL}]
         )
]
```

Use this command to redefine the `OPTIMAL` or `MAXEXTENTS` parameters.

Deallocating Space From Rollback Segments

- **Use the `ALTER ROLLBACK SEGMENT` command.**
- **If extents are active, they might not shrink to the requested size.**

```
ALTER ROLLBACK SEGMENT rbs01
  SHRINK TO 4M;
```

Deallocating Space from Rollback Segments

If `OPTIMAL` has been specified for a rollback segment, the Oracle server attempts to deallocate extents to release space above the optimal size.

Manual deallocation

To manually deallocate space from a rollback segment, use the following command:

```
ALTER ROLLBACK SEGMENT rollback_segment
   SHRINK [ TO integer [ K|M ]];
```

This command attempts to reduce the size of the rollback segment to the specified size but stops short if an extent cannot be deallocated because it is active.

If *`integer`* is not specified, then the Oracle server attempts to deallocate extents until the size of the rollback segment is equal to `OPTIMAL`.

If the *`integer`* specified is larger than the current size of the rollback segment, then this command is ignored.

Taking Rollback Segment Offline

- **Take a rollback segment offline to make it unavailable.**
- **If transactions are using the rollback segment, the status is temporarily changed to `PENDING OFFLINE`.**

```
ALTER ROLLBACK SEGMENT rbs01
  OFFLINE;
```

ORACLE

Taking a Rollback Segment Offline

Take a rollback segment offline:

- To prevent new transactions from using a rollback segment
- If the rollback segment must be dropped

Syntax

Use the following command to take a rollback segment offline:

```
ALTER ROLLBACK SEGMENT rollback_segment OFFLINE;
```

If there are transactions using the rollback segment at the time this statement is executed, then the status of the rollback segment is set to `PENDING OFFLINE`, as seen from the `V$ROLLSTAT` dynamic performance view. As soon as all existing transactions complete, the segment is taken offline.

Dropping Rollback Segments

- **A rollback segment must be offline before it can be dropped.**
- **To drop a rollback segment:**

```
DROP ROLLBACK SEGMENT rbs01;
```

ORACLE

Dropping Rollback Segments

Use the following command to drop a rollback segment:

```
DROP ROLLBACK SEGMENT rollback_segment;
```

A rollback segment may need to be dropped if it is no longer needed or if it needs to be re-created with different storage settings for `INITIAL`, `NEXT`, or `MINEXTENTS`.

A rollback segment must be offline before it can be dropped.

Planning Rollback Segments: Number

- **OLTP**
  - **Many small rollback segments**
  - **Four transactions per rollback segment**
  - **Up to ten transactions per rollback segment**
- **Batch**
  - **Few large rollback segments**
  - **One per transaction**

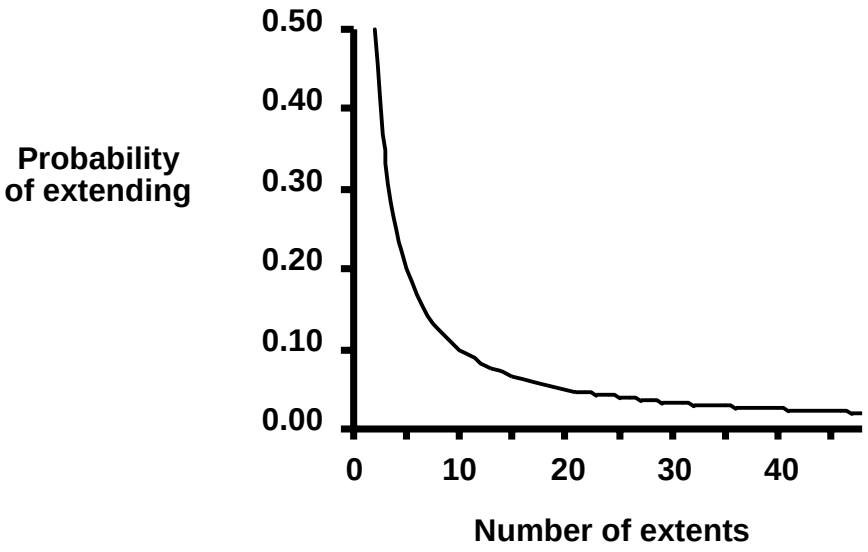
ORACLE

Number of Rollback Segments

The header block of a rollback segment contains transaction table entries that define the state of each transaction. Every transaction that uses a rollback segment must update the transaction table frequently. This could cause contention on the header, especially in an OLTP environment. Because OLTP environments typically use short transactions, many small rollback segments are recommended in this environment. If possible, create one rollback segment for every four concurrent transactions.

Batch environments generally run fewer jobs that may need to carry out several changes. These jobs require large rollback segments. Therefore, in a batch environment, allow for the growth of the rollback segments by creating them in large tablespaces.

Planning Rollback Segments: Number of Extents



ORACLE

Size of a Rollback Segment

The number of bytes required to store information that is needed in case of rollback depends on two things:

- The type of transaction that is being performed (insert, update, delete, and so on)
- The actual data that is being processed

In general, inserting a given record into a table generates less rollback than deleting the same record. Inserts must store only the row ID in the rollback, whereas deletions must store the actual row itself.

You can estimate the size of the rollback segment by running the longest transaction expected and checking the size of the rollback segment or the amount of rollback generated. The *Oracle8i: Performance Tuning* and *Oracle8i: SQL Statement Tuning* courses discuss monitoring statistics.

Number of Extents

A dynamic extension of a rollback segment can be minimized by creating rollback segments with a large number of extents. Creating rollback segments with `MINEXTENTS=20` is recommended to reduce the possibility of extension.

Rollback Segment Problems

- Insufficient space for transactions
- Read-consistency errors
- Blocking sessions
- Errors in taking a tablespace offline

ORACLE

Insufficient Space for Transactions

- **No space in tablespace:**
  - **Extend data files**
  - **Enables automatic extension of data files**
  - **Add data files**
- **MAXEXTENTS reached for segment**
  - **Increase MAXEXTENTS**
  - **Re-create segments with larger extent sizes**

ORACLE

Possible Causes

A transaction uses a single rollback segment and may fail if there is insufficient space in the rollback segment (ORA-01562). This could be caused by one of the following:

- There is insufficient space in the tablespace for the rollback segments to extend (ORA-01560).
- The number of extents in the rollback segment has reached MAXEXTENTS, and additional extents cannot be allocated (ORA-01628).

Solution

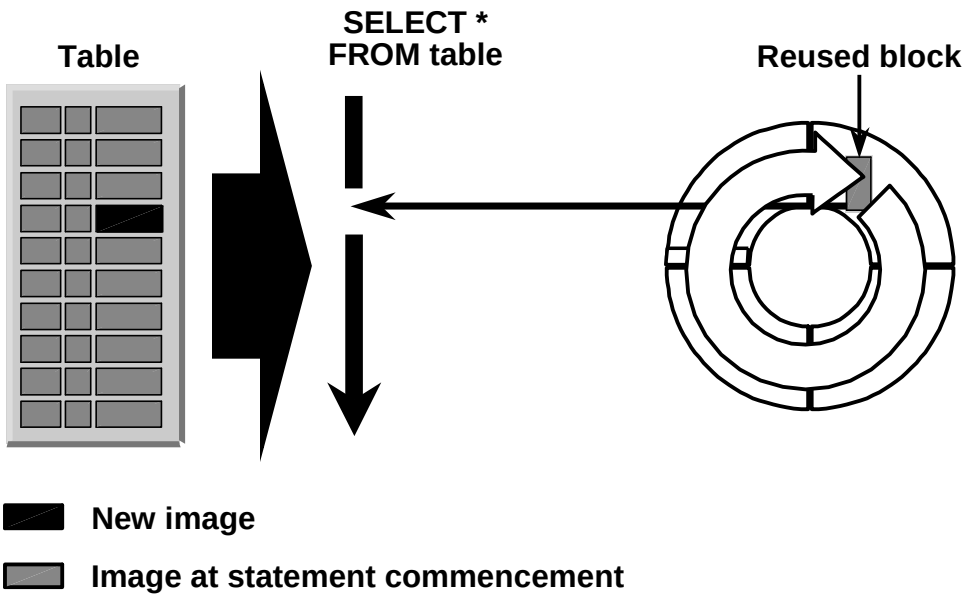
If the tablespace does not contain free space, increase the space available by using the following methods:

- Setting OPTIMAL to ensure that a single rollback segment does not use all of the free space in the tablespace
- Shrinking rollback segments back to their optimal size
- Increasing the size of the tablespace

If a rollback segment cannot allocate more extents because the limit imposed by MAXEXTENTS has been reached:

- Increase MAXEXTENTS for the rollback segment.
- Drop and re-create the rollback segment with larger extent sizes to avoid a recurrence of the problem.

Read-Consistency Errors



ORACLE

Possible Causes

To provide read consistency, the Oracle server guarantees that changes made by other users that are not committed when the statement begins, or are made after the statement begins execution, are not seen by the statement. If the Oracle server cannot construct a read-consistent image of data, the user will receive an `ORA-01555 SNAPSHOT TOO OLD` error. This error can occur when the transaction that made the change has already committed, and:

- The transaction slot in the rollback header has been reused.
- The before image in the rollback segment has been overwritten by another transaction.

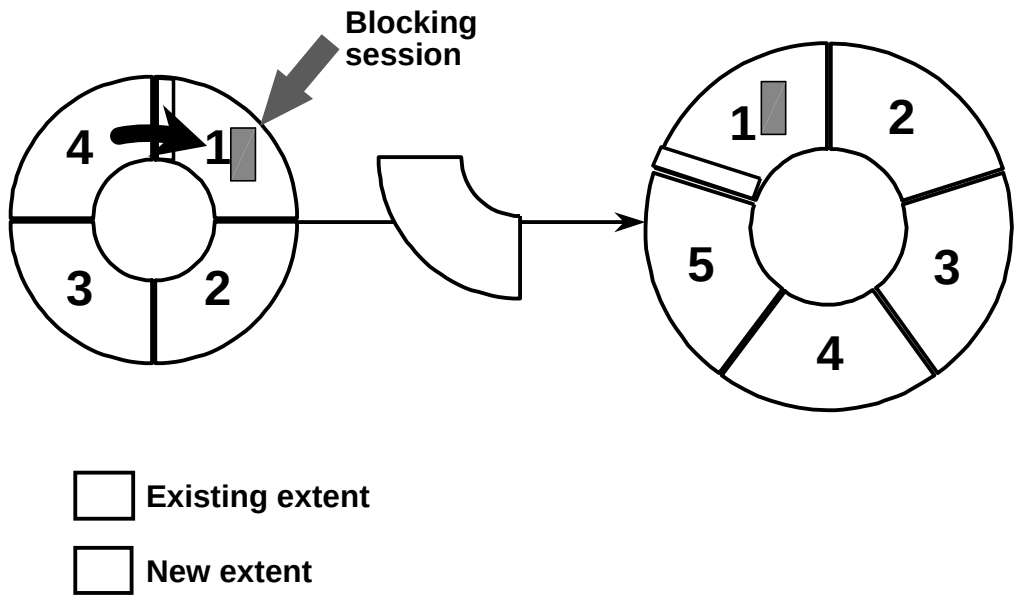
Solution

Read-consistency errors can be minimized by ensuring that rollback segments are created with:

- A higher `MINEXTENTS` value
- Larger extent sizes
- A higher `OPTIMAL` value

Note: These errors cannot be avoided by increasing `MAXEXTENTS`.

Blocking Sessions



ORACLE

Possible Causes

When an extent in a rollback segment is full, the Oracle server attempts to reuse the next extent in the segment. If this new extent contains one active entry, that is, an entry by a transaction that is still active, it cannot be used. In these cases, a rollback segment allocates an additional extent. The transaction cannot skip an extent in the ring and continue writing to a subsequent extent. A transaction that has made only a few changes, but has been idle for a long time could cause rollback segments to grow even though there are many free extents. In this situation, a lot of space is wasted and a database administrator may need to intervene to avoid excessive rollback segment growth.

Possible Causes (continued)

Solution

Query V$ROLLSTAT, V$SESSION, and V$TRANSACTION views to find any blocking transactions.

Example

```
SQL> SELECT s.sid, s.serial#, t.start_time, t.xidusn, s.username
     2 FROM v$session s, v$transaction t, v$rollstat r
     3 WHERE s.saddr = t.ses_addr
     4 AND t.xidusn = r.usn
     5 AND ((r.curext = t.start_uext-1) OR
     6 ((r.curext = r.extents-1) AND t.start_uext=0));
    SID   SERIAL#    START_TIME           XIDUSN  USERNAME
    ---   -------    ------------------    ------- ------
    9     27         10/30/97 21:10:41     2       SYSTEM
    1 row selected.
```

Check whether the transaction can be ended by the user. If not, it may be necessary to kill the session.

Errors in Taking a Tablespace Offline

You cannot take a tablespace offline if it contains an active rollback segment.

1. **Determine which rollback segments are in the tablespace.**
2. **Take all of these rollback segments offline.**
3. **Find active transactions using these rollback segments.**
4. **Find the session ID and serial number.**
5. **Terminate the session, if necessary.**
6. **Take the tablespace offline.**

ORACLE

Problem Diagnosis and Resolution

If a tablespace contains one or more active rollback segments, it cannot be taken offline. The session executing the statement receives an ORA-01546 error message.

Solution

Perform the following steps:

1. Query `DBA_ROLLBACK_SEGS` to find which rollback segments are in the tablespace.
2. Take all rollback segments in the tablespace offline.
3. Check `V$TRANSACTION` to find which transactions are currently using these rollback segments.
4. Use `V$SESSION` to obtain the username and session information.
5. Kill the session or have the user end the transaction.
6. Take the tablespace offline.

Practice Solutions for SQL*Plus

Practice Solution Conventions

The following solutions are designed for the student interested in using SQL*Plus to accomplish the practices.

For those students interested in using SQL*Worksheet, the commands are identical. Where exceptions exist, notations have been made under the heading Using SQL*Plus Worksheet heading. Keep in mind, each time you connect as a different user the service name must be included in the connection string. For example:

```
CONNECT SYS/ORACLE@[service name] AS SYSDBA.
```

Running SQL Lab Scripts

Lab scripts have been included in the lab exercises to perform certain tasks for you. The task being accomplished may be one that you have not learned at the point of the exercise or it may be that the task is being accomplished for your convenience. When asked to run a `.sql` script, take time to review the script before running it. This will provide you with a thorough understanding of what is actually being created for you, and how it affects the task you are about to accomplish in the practice.

Using Operating System Command Line

On occasion you will need to perform some tasks using the operating system command line. An [!] will be used from the SQL*Plus command line to get you out of SQL*Plus and into the operating system command line mode. Once you have completed your work on the operating system command line you can return to the SQL*Plus command line by using `exit`. For example: Remove the current password file located at `$HOME/ADMIN/PFILE`.

```
SQL > !rm $HOME/ADMIN/PFILE/orapw$ORACLE_SID
$ > exit
```

Connecting as SYS

When asked to connect as user `SYS` it will always be with `SYSDBA` privilege.

```
CONNECT / AS SYSDBA
```

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.

Answer: D

2 Which one of the following memory areas is not part of the SGA?
- **a** Database Buffer Cache
- **b** PGA
- **c** Redo Log Buffer
- **d** Shared Pool

Answer: B

3 Which three 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.
- **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.

Answer: B,C,D

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
- **d** Shared Pool

Answer: D

5 The primary purpose of the Redo Log Buffer 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

Answer: False, A PGA is a memory region that contains data and control information for a single server process or a single background process.

Practice 1: Solutions (continued)

7 Which of the following becomes available when an Oracle instance is started?
- **a** User process
- **b** Server process
- **c** Background processes

Answer: C

8 Identify five mandatory background processes.

_____
_____
_____
_____
_____

Answer: DBWR, LGWR, PMON, SMON, CKPT

9 Match the process with its task.

a Database Writer	1. Assists with writing to the data file headers
b Log Writer	2. Responsible for instance recovery
c System Monitor	3. Cleans up after failed processes
d Process Monitor	4. Records database changes for recovery purposes
e Checkpoint	5. Writes dirty buffers to the data files

Answer: a = 5, b = 4, c = 2, d = 3, e =1

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

Answer: True

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

Answer: D,A,C,B,E

Practice 1: Solutions (continued)

12 Identify the components of an Oracle server.

_____
_____

Answer: Oracle Instance, Oracle database

13 Identify the components of an Oracle Instance.

_____
_____

Answer: SGA, Background processes

14 Identify three file types that make up an Oracle database.

_____
_____
_____

Answer: data files, control files, online redo log files

Practice 2: Solutions

This practice is Instructor led. Your instructor will provide you with login accounts and walk you through logging into your account. Write the information provided by your Instructor below.

Host name: _____

SID name: _____

1 Connect to SQL*Plus as SYSDBA.

Hint:

- Connect to your account using instructions provided by your Instructor.
- Start SQL*Plus.
- Connect as the user SYS.

```
$ sqlplus /nolog
SQL*Plus: Release 9.2.0.1.0 - Production on Mon Aug 5 10:52:10
2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights
reserved.
SQL> CONNECT / AS SYSDBA
Connected.
```

2 Using SQL*Plus, run the following query to verify the connection to the database has been made.

```
SQL> SELECT * FROM DUAL;
D
-
X
```

3 Launch Oracle Enterprise Manager in Standalone mode.

- Navigate to `Start > Programs > Oracle-OraHome92 > Enterprise Manager Console`.
- Select `Launch standalone` option.
- Click OK.

Practice 2: Solutions (continued)

4 If you are in an Oracle classroom you must perform the following four steps that are particular to the Oracle classroom setup:

a. Click the `omsconfig file` update icon on the desktop.

Enter the name of the UNIX server your class is using. Your instructor will provide the server name. Enter it exactly as it is given to you. This is a case-sensitive entry.

b. Open an MSDOS window.

c. At the command prompt enter: `oemctl start oms`. Wait for the message:

```
"The Oracle92_homeManagementServer service was
started successfully."
```

d. Close the MSDOS window.

Launch Oracle Enterprise Manager using Oracle Management Server.

- Start the OEM Console and select the Login to the Oracle Management Server option. Log in as follows:

  Administrator: `sysman` **Note:** Case is important.

  Password: `oem_temp` **Note:** Case is important.

  When prompted, change the password to `oracle`. **Note:** Case is important.

  Management Server: (Instructor supplied)

- Navigate to Navigator > Discover Nodes from the main menu after the OEM. Console is opened. The Discovery Wizard dialog box will appear.
- Select Next.
- Enter the name of the node you want to manage: that is, designated database server host name. (Instructor supplied)
- Click Next.
- Select Finish after discovery is complete.
- Click OK. **Note:** Alert the instructor if discovery is not successful.
- Expand the Database folder.
- Double click your designated database provided by your instructor.
- Enter the connection information:

  User: `sys`

  Password: `secure`

  As: `SYSDBA`

Practice 2: Solutions (continued)

4 Launch Oracle Enterprise Manager using Oracle Management Service. (continued)

- Set the node credentials for running jobs.
- Navigate to Configuration > Preferences from the main menu
- Select the Preferred Credentials page.
- Scroll down and select the entry for your designated database.
- Supply the following:

  Username: (Instructor supplied)

  Password: (Instructor supplied)

  Confirm Password.

  Role: SYSDBA
- Click OK.

5 Start SQL*Plus Worksheet

SQL*Plus Worksheet can be started within the Oracle Enterprise Manager Console using the following navigation:

- Navigate to Tools > Database Applications > SQL*Plus Worksheet

SQL*Plus Worksheet can also be started from the Windows NT menu by doing the following:

- Navigate to Start > Programs > Oracle-OraHome92 > Application Development > SQLPlus Worksheet
  - Connect directly to the database defined by the instructor, enter
    - Username
    - Password
    - Service
  - Connect as: SYSDBA
  - Click OK.

Note: Each time you log in as a different user (within SQL*Plus Worksheet) the service name must be included in the connection string. For example:

```
CONNECT SYS/ORACLE@[service name] AS SYSDBA.
```

Practice 3: Solutions

1 Connect to the database as user SYS and shut down the database.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
```

2 With the database shut down, create an SPFILE from the PFILE. The SPFILE will be created in $ORACLE_HOME/dbs.

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> CREATE SPFILE FROM PFILE;
File created.
```

Practice 3: Solutions (continued)

3 From the operating system, view the SPFILE.

```
SQL> !more $ORACLE_HOME/dbs/spfileU481.ora
*.background_dump_dest='/home1/user481/ADMIN/BDUMP'
*.compatible='9.2.0'
*.control_files='/home1/user481/ORADATA/u01/ctrl01.ctl'
*.core_dump_dest='/home1/user481/ADMIN/CDUMP'
*.db_block_size=4096
*.db_cache_size=16M
*.db_domain='world'
*.db_name='U481'
*.global_names=TRUE
*.job_queue_processes=2
*.log_buffer=64000
*.max_dump_file_size='10240'
*.shared_pool_size=48M
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS'
*.user_dump_dest='/home1/user481/ADMIN/UDUMP'
…
```

Practice 3: Solutions (continued)

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

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> STARTUP
ORACLE instance started.

Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
```

Practice 3: Solutions (continued)

5 a Shut down the database and open it in read-only mode.

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT
Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
SQL> ALTER DATABASE OPEN READ ONLY;
Database altered.
```

b Connect as user HR password HR and insert a row into the REGIONS table as follows:

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

What happens?

```
SQL> CONNECT HR/HR
Connected.
SQL> INSERT INTO regions VALUES (5, 'Mars');
INSERT INTO regions VALUES (5, 'Mars')
            *
ERROR at line 1:
ORA-01552: cannot use system rollback segment for non-system
tablespace
'SAMPLE'
```

Practice 3: Solutions (continued)

5 c Put the database back in read-write mode.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
Database opened.
```

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

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

HR SESSION

```
SQL> CONNECT HR/HR
Connected.
SQL> INSERT INTO regions VALUES (5, 'Mars');
1 row created.
```

Practice 3: Solutions (continued)

5 b In a new telnet session start SQL*Plus. Connect user SYS and perform a SHUTDOWN TRANSACTIONAL.

```
SYS SESSION
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN TRANSACTIONAL
```

What happens? The SHUTDOWN TRANSACTIONAL is waiting on the HR session transaction to complete.

Practice 3: Solutions (continued)

6 c Roll back the insert in the HR session and exit.

HR SESSION

```
SQL> ROLLBACK;
Rollback complete.
SQL> EXIT;
ERROR:
ORA-01089: immediate shutdown in progress - no operations are
permitted
JServer Release 9.0.0.0.0 - Beta (with complications)
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0
- 64bit Productn
With the Partitioning, Oracle Label Security, OLAP and Oracle
Data Mining optios
JServer Release 9.2.0.1.0 - Production
```

What happens to the HR session? An ORA-01089 message is received. The HR user cannot EXIT because the SYS user has issued a SHUTDOWN TRANSACTIONAL. No other operations are allowed.

What happens to the SYS session? Once the HR session completes a ROLLBACK, the SYS session will shut down the database.

SYS SESSION

```
Database closed.
Database dismounted.
ORACLE instance shut down.
```

The HR session will be disconnected based on its EXIT command.

HR SESSION

```
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0
- 64bit Productn
With the Partitioning, Oracle Label Security, OLAP and Oracle
Data Mining optios
JServer Release 9.2.0.1.0 - Production
```

Practice 3: Solutions (continued)

7 a In the user SYS session start the database.

```
SYS SESSION
SQL> STARTUP
ORACLE instance started.
Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
Database opened.
```

b In the open telnet session start SQL*Plus and connect as user HR.

Note: Keep the two SQL*Plus sessions open, one session as user SYS and one as user HR.

```
HR SESSION
$ sqlplus /nolog
SQL*Plus: Release 9.2.0.1.0 - Production on Wed Jul 24
16:52:51 2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights
reserved.
SQL> CONNECT HR/HR
Connected.
```

c As user SYS enable restricted session.

```
SYS SESSION
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
System altered.
```

Practice 3: Solutions (continued)

7 d As user HR, SELECT from the REGIONS table. Is the SELECT successful? Yes

HR SESSION

```
SQL> SELECT * FROM regions;
REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
```

e Exit the session, then reconnect as HR. What happens? The user HR does not have RESTRICTED SESSION privilege, and therefore, cannot log in.

HR SESSION

```
SQL> EXIT
Disconnected from Oracle9i Enterprise Edition Release
9.2.0.1.0 - 64bit Productn
With the Partitioning, Oracle Label Security, OLAP and
Oracle Data Mining optios
JServer Release 9.2.0.1.0 - Production
$ sqlplus /nolog
SQL*Plus: Release 9.2.0.1.0 - Production on Wed Jul 24
17:00:35 2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights
reserved.
SQL> CONNECT HR/HR
ERROR:
ORA-01035: ORACLE only available to users with RESTRICTED
SESSION privilege
Warning: You are no longer connected to ORACLE.
```

Practice 3: Solutions (continued)

7 f As user SYS disable restricted session.

```
SYS SESSION
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
System altered.
```

g Exit HR telnet session.

```
HR SESSION
$ EXIT
This session is no longer connected.
```

Practice 5: Solutions

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 tables.
   - **d** The data dictionary records and verifies information about its associated database.

   Answer: All of the Above

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

   Answer: False, The base tables are created with the `sql.bsq` script

3. Which three 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 reference.
   - **d** The data dictionary is a necessary ingredient for the database to function.

   Answer: B,C,D

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

   Answer: True

5. The information for a dynamic performance view is gathered from the control file.
   - **a** True
   - **b** False

   Answer: True

6. Which of the following questions might a dynamic performance 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?

   Answer: A,B,E

Practice 5: Solutions (continued)

7 Connect as SYSTEM/MANGER and find a list of the data dictionary views.

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> SELECT table_name FROM dictionary;

TABLE_NAME
------------------------------
ALL_ALL_TABLES
ALL_ARGUMENTS
ALL_ASSOCIATIONS
ALL_AUDIT_POLICIES
ALL_BASE_TABLE_MVIEWS
ALL_CATALOG
ALL_CLUSTERS
ALL_CLUSTER_HASH_EXPRESSIONS
ALL_COLL_TYPES
.
.
.
1256 rows selected.
```

Practice 5: Solutions (continued)

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

Hint: Query the dynamic performance views V$DATABASE, V$THREAD, and V$PARAMETER.

```
SQL> SELECT name FROM v$database;
NAME
-----
 U481
SQL> SELECT instance FROM v$thread;
INSTANCE
--------
U481
SQL> SELECT value FROM v$parameter
  2  WHERE name='db_block_size';
VALUE
-----
4096
```

9 List the name of the data files.

Hint: Query the dynamic performance view V$DATAFILE.

```
SQL> SELECT name FROM v$datafile;
NAME
----------------------------------------------
/home1/user481/ORADATA/u01/system01.dbf
/home1/user481/ORADATA/u02/undotbs01.dbf
/home1/user481/ORADATA/u03/users01.dbf
/home1/user481/ORADATA/u03/indx01.dbf
/home1/user481/ORADATA/u02/sample01.dbf
/home1/user481/ORADATA/u01/querydata01.dbf

6 rows selected.
```

Practice 5: Solutions (continued)

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

Hint: Query the data dictionary view DBA_DATA_FILES to identify the SYSTEM tablespace data file.

```
SQL> SELECT file_name FROM dba_data_files
  2  WHERE tablespace_name = 'SYSTEM';

FILE_NAME
----------------------------------------
/home1/user481/ORADATA/u01/system01.dbf
```

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.

```
SQL> SELECT sum(bytes)/1024 "free space in KB"
  2  FROM dba_free_space;
free space in KB
----------------
           22716
SQL> SELECT sum(bytes)/1024 "used space in KB"
  2  FROM dba_segments;
used space in KB
----------------
          212448
```

Practice 5: Solutions (continued)

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.

```
SQL> SELECT username, created FROM dba_users;
USERNAME                       CREATED
------------------------------ ---------
SYS                            16-Jul-02
SYSTEM                         16-Jul-02
DBSNMP                         16-Jul-02
OUTLN                          16-Jul-02
MDSYS                          16-AUG-02
HR                             16-AUG-02
OE                             16-AUG-02
GC9201DBAIS3U$81               16-AUG-02

8 rows selected.
```

Practice 6: Solutions

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

Hint: Query the dynamic performance view V$CONTROLFILE.

Note: You can also use V$PARAMETER, or execute the SHOW PARAMETER command to display the name and the location of the control file.

```
SQL> COL name FORMAT a50
SQL> SELECT * FROM   v$controlfile;
STATUS  NAME
------- ------------------------------------
         /home1/user481/ORADATA/u01/ctrl01.ctl
```

Practice 6: Solutions (continued)

2 a 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?

Hints:
- Connect as user SYS.
- Shutdown the database using the `IMMEDIATE` option.
- Using OS command line, copy the control file `.ctl` as a `.bak` extension.
- Remove the control file `.ctl.`
- Start the database.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.ctl
       $HOME/ORADATA/u01/ctrl01.bak
SQL> !rm $HOME/ORADATA/u01/ctrl01.ctl
SQL> STARTUP
ORACLE instance started.
Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
ORA-00205: error in identifying controlfile, check alert log
for more info
```

Practice 6: Solutions (continued)

2 b To resolve the ORA-00205 error:

- Shutdown the database.
- Rename the copied control file to the appropriate name.
- Startup the database.

```
SQL> SHUTDOWN IMMEDIATE
ORA-01507: database not mounted
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.bak
       $HOME/ORADATA/u01/ctrl01.ctl
SQL> STARTUP
ORACLE instance started.
Total System Global Area  131040632 bytes
Fixed Size                   730488 bytes
Variable Size             113246208 bytes
Database Buffers           16777216 bytes
Redo Buffers                 286720 bytes
Database mounted.
Database opened.
```

Practice 6: Solutions (continued)

3 a 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`.

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. **Note:** Normally the permissions on the file would not be changed, this is for the classroom environment.
- Start up the database.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET control_files =
          '$HOME/ORADATA/u01/ctrl01.ctl',
        '$HOME/ORADATA/u02/ctrl02.ctl' SCOPE=SPFILE;
System altered.
SQL> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.ctl
           $HOME/ORADATA/u02/ctrl02.ctl
SQL> !chmod 660 $HOME/ORADATA/u02/ctrl02.ctl
SQL> STARTUP
ORACLE instance started.
Total System Global Area  131040632 bytes
Fixed Size                   730488 bytes
Variable Size             113246208 bytes
Database Buffers           16777216 bytes
Redo Buffers                 286720 bytes
Database mounted.
Database opened.
```

Practice 6: Solutions (continued)

3 b Confirm that the control file was multiplexed and is now being used.

Hint: Query the dynamic performance views `V$CONTROLFILE` or `V$PARAMETER`, or use the `SHOW PARAMETER` command to confirm that both control files are being used.

```
SQL> SELECT name FROM v$controlfile;
NAME
-------------------------------------
/home1/user481/ORADATA/u01/ctrl01.ctl
/home1/user481/ORADATA/u02/ctrl02.ctl
```

Practice 6: Solutions (continued)

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

Hint: Query the dynamic performance view `V$CONTROLFILE_RECORD_SECTION`.

```
SQL> SELECT records_total
  2  FROM   v$controlfile_record_section
  3  WHERE  type = 'DATAFILE';

RECORDS_TOTAL
-------------
           30
```

Practice 7: Solutions

1 a List the number and location of existing log files.

Hint: Query the dynamic performance view `V$LOGFILE`.

```
SQL> SELECT member FROM v$logfile;
MEMBER
-----------------------------------
/home1/user481/ORADATA/u03/log01a.rdo
/home1/user481/ORADATA/u03/log02a.rdo
```

b Display the number of online redo log file groups and members your database has.

Hint: Query the dynamic performance view `V$LOGFILE`.

```
SQL> SELECT group#, members FROM v$log;
    GROUP#    MEMBERS
---------- ----------
         1          1
         2          1
```

Practice 7: Solutions (continued)

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

Hints:

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

```
SQL> SELECT log_mode FROM v$database;
LOG_MODE
------------
NOARCHIVELOG

SQL> SELECT archiver FROM v$instance;
ARCHIVE
-------
STOPPED
```

Practice 7: Solutions (continued)

3 Add an online redo log file 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 an online redo log file member to each group.
- Query the dynamic performance view `V$LOGFILE` to verify the result.

```
SQL> ALTER DATABASE ADD LOGFILE MEMBER
  2  '$HOME/ORADATA/u04/log01b.rdo' to Group 1,
  3  '$HOME/ORADATA/u04/log02b.rdo' to Group 2;
Database altered.
SQL> COLUMN GROUP# FORMAT 99
SQL> COLUMN MEMBER FORMAT a40
SQL> SELECT * FROM v$logfile;

GROUP# STATUS TYPE    MEMBER
------ ------ ------- ------------------------------------
    1          ONLINE /home1/user481/ORADATA/u03/log01a.rdo
    2          ONLINE /home1/user481/ORADATA/u03/log02a.rdo
    1 INVALID  ONLINE  /home1/user481/ORADATA/u04/log01b.rdo
    2 INVALID  ONLINE  /home1/user481/ORADATA/u04/log02b.rdo
```

Practice 7: Solutions (continued)

4 Add an online redo log file group with two members located on `u03` and `u04` using the following naming conventions and verify the result.

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

Hints:

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

```
SQL> ALTER DATABASE ADD
  2  LOGFILE GROUP 3('$HOME/ORADATA/u03/log03a.rdo',
  3  '$HOME/ORADATA/u04/log03b.rdo') SIZE 1024K;
Database altered.
SQL> COLUMN GROUP# FORMAT 99
SQL> COLUMN MEMBER FORMAT a40
SQL> SELECT * FROM v$logfile;
GROUP# STATUS  TYPE    MEMBER
------ ------- ------- ----------------------------------
     1          ONLINE  /home1/user481/ORADATA/u03/log01a.rdo
     2         ONLINE  /home1/user481/ORADATA/u03/log02a.rdo
     1 INVALID ONLINE  /home1/user481/ORADATA/u04/log01b.rdo
     3         ONLINE  /home1/user481/ORADATA/u03/log03a.rdo
     3         ONLINE  /home1/user481/ORADATA/u04/log03b.rdo
6 rows selected.
SQL> SELECT group#, members FROM v$log;
GROUP#    MEMBERS
------ ----------
     1          2
     2          2
     3          2
```

Practice 7: Solutions (continued)

5 Remove the online redo log file group created in step 4.

Hints:

- Use `ALTER SYSTEM SWITCH LOGFILE` if the log files are active. The number of log switches required will vary. **Note:** Query the database to see which log file is active then decide how many times you need to perform the `ALTER SYSTEM SWITCH LOGFILE` command.
- Execute the `ALTER DATABASE DROP LOGFILE GROUP` command to remove the log group.
- Query the dynamic performance view `V$LOG` to verify the result.
- Remove the operating system files for the group.

.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER DATABASE DROP LOGFILE GROUP 3;
Database altered.
SQL> SELECT group#, members FROM v$log;
GROUP#    MEMBERS
------ ----------
    1          2
    2          2
SQL> !rm $HOME/ORADATA/u03/log03a.rdo
SQL> !rm $HOME/ORADATA/u04/log03b.rdo
```

Practice 7: Solutions (continued)

6 Resize all online redo log files to 1024 KB.

Hints:

- We cannot resize log files, therefore, add new logs and drop the old.
- Execute the `ALTER DATABASE ADD LOGFILE GROUP` command to add two new groups with the size 1024 KB.
- Query the dynamic performance view `V$LOG` to check the active group.
- Execute the `ALTER SYSTEM SWITCH LOGFILE` command to force log switches and change the group stage to inactive. The number of log switches required will vary. **Note:** Query the database to see which log file is active then decide how many times you need to perform the `ALTER SYSTEM SWITCH LOGFILE` command.
- Execute the `ALTER DATABASE DROP LOGFILE` command to remove the unused groups.
- Query the dynamic performance view `V$LOG` to verify the result.

```
SQL> ALTER DATABASE ADD LOGFILE
  2     GROUP 3( '$HOME/ORADATA/u03/log03a.rdo',
  3              '$HOME/ORADATA/u04/log03b.rdo' )
  4               SIZE 1024K,
  5     GROUP 4( '$HOME/ORADATA/u03/log04a.rdo',
  6              '$HOME/ORADATA/u04/log04b.rdo' )
  7               SIZE 1024K;
Database altered.
SQL> SELECT group#, status
  2  FROM   v$log;

GROUP# STATUS
------ ----------------
     1 INACTIVE
     2 CURRENT
     3 UNUSED
     4 UNUSED
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
```

Practice 7: Solutions (continued)

```
                          - continued -

SQL> ALTER DATABASE DROP LOGFILE GROUP 1, GROUP 2;
Database altered.
SQL> SELECT group#, status, bytes FROM v$log;
GROUP# STATUS                BYTES
------ ---------------- ----------
     3 CURRENT             1048576
     4 INACTIVE            1048576
```

Practice 8: Solutions

1 Create permanent tablespaces with the following names and storage:

a Tablespace Name: `DATA01`

Datafile Name: `data01.dbf`

Size: `2M`

Extent Management: `dictionary`

Location: `u04`

```
SQL> CREATE TABLESPACE DATA01
  2  DATAFILE '$HOME/ORADATA/u04/data01.dbf' SIZE 2M
  3  EXTENT MANAGEMENT DICTIONARY;
Tablespace created.
```

b Tablespace Name: `DATA02`

Datafile Name: `data02.dbf`

Size: `1M`

Extent Management: `local uniform size 100K`

Location: `u03`

```
SQL> CREATE TABLESPACE DATA02
  2  DATAFILE '$HOME/ORADATA/u03/data02.dbf' SIZE 1M
  3  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 100K;
Tablespace created.
```

c Tablespace Name: `INDEX01`

Datafile Name: `index.dbf`

Size: `1M`

Extent Management: `local uniform size 4K`

Location: `u02`

Enable automatic extension of 500 KB when more extents are required with a maximum size of 2 MB.

```
SQL> CREATE TABLESPACE INDEX01
  2  DATAFILE '$HOME/ORADATA/u02/index01.dbf' SIZE 1M
  3  AUTOEXTEND ON NEXT 500K MAXSIZE 2M
  4  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4K;
Tablespace created.
```

Practice 8: Solutions

1 Create permanent tablespaces with the following names and storage (continued):

d Tablespace Name: `RONLY`

Datafile Name: `ronly01.dbf`

Size: `1M`

Location: `u01`

Default storage. DO NOT make the tablespace read only at this time.

```
SQL> CREATE TABLESPACE RONLY
  2  DATAFILE '$HOME/ORADATA/u01/ronly01.dbf' SIZE 1M;
Tablespace created.
```

Practice 8: Solutions (continued)

e Display the information from the data dictionary.

Hint: Information about tablespaces can be viewed using any of the following queries.

- DBA_TABLESPACES
- V$TABLESPACE
- V$DATAFILE

```
SQL> SELECT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
UNDOTBS
TEMP
USERS
INDX
SAMPLE
QUERY_DATA
DATA01
DATA02
INDEX01
RONLY

11 rows selected.
```

Practice 8: Solutions (continued)

2 Allocate 500K more disk space to tablespace `DATA02.` Verify the result.

```
SQL> ALTER DATABASE
  2  DATAFILE '$HOME/ORADATA/u03/data02.dbf' RESIZE 1500K;
Database altered.
SQL> COLUMN name FORMAT a40
SQL> SELECT name, bytes, create_bytes
  2  FROM   v$datafile
  3  WHERE  name LIKE '%data02%';
NAME                                     BYTES   CREATE_BYTES
---------------------------------------- ------- ------------
/home1/user481/ORADATA/u03/data02.dbf 1536000  1048576
```

Practice 8: Solutions (continued)

3 Relocate tablespace `INDEX01` to subdirectory `u06`. Verify relocation and status of `INDEX01`.

Hints:

- Take the `INDEX01` tablespace offline.
- Use `V$DATAFILE` to verify status.
- Use operating system move command to move the tablespace to `u06`.
- Use `ALTER TABLESPACE` to relocate the tablespace.
- Place the `INDEX01` tablespace online.
- Use `V$DATAFILE` to verify status.

```
SQL> ALTER TABLESPACE index01 OFFLINE;
Tablespace altered.
SQL> SELECT status, name FROM v$datafile;
STATUS  NAME
------- ---------------------------------------
SYSTEM  /home1/user481/ORADATA/u01/system01.dbf
ONLINE  /home1/user481/ORADATA/u02/undotbs01.dbf
ONLINE  /home1/user481/ORADATA/u03/users01.dbf
ONLINE  /home1/user481/ORADATA/u03/indx01.dbf
ONLINE  /home1/user481/ORADATA/u02/sample01.dbf
ONLINE  /home1/user481/ORADATA/u01/querydata01.dbf
ONLINE  /home1/user481/ORADATA/u02/example01.dbf
ONLINE  /home1/user481/ORADATA/u04/data01.dbf
ONLINE  /home1/user481/ORADATA/u03/data02.dbf
OFFLINE /home1/user481/ORADATA/u02/index01.dbf
ONLINE  /home1/user481/ORADATA/u01/ronly01.dbf
11 rows selected.
SQL> !mv $HOME/ORADATA/u02/index01.dbf
        $HOME/ORADATA/u06/index01.dbf
SQL> ALTER TABLESPACE index01
  2     RENAME DATAFILE
  3     '$HOME/ORADATA/u02/index01.dbf' TO
  4     '$HOME/ORADATA/u06/index01.dbf';
Tablespace altered.
SQL> ALTER TABLESPACE index01 ONLINE;
Tablespace altered.
                      - continued -
```

Practice 8: Solutions (continued)

```
                        - continued -

SQL> SELECT status, name FROM v$datafile;
STATUS  NAME
------- ----------------------------------------
SYSTEM  /home1/user481/ORADATA/u01/system01.dbf
ONLINE  /home1/user481/ORADATA/u02/undotbs01.dbf
ONLINE  /home1/user481/ORADATA/u03/users01.dbf
ONLINE  /home1/user481/ORADATA/u03/indx01.dbf
ONLINE  /home1/user481/ORADATA/u02/sample01.dbf
ONLINE  /home1/user481/ORADATA/u01/querydata01.dbf
ONLINE  /home1/user481/ORADATA/u02/example01.dbf
ONLINE  /home1/user481/ORADATA/u04/data01.dbf
ONLINE  /home1/user481/ORADATA/u03/data02.dbf
ONLINE  /home1/user481/ORADATA/u02/index01.dbf
ONLINE  /home1/user481/ORADATA/u01/ronly01.dbf
11 rows selected.
```

Practice 8: Solutions (continued)

4 a Connect as user SYSTEM and create a table in tablespace RONLY. Make tablespace read-only. Run a query to verify it.

```
SQL> Connect SYSTEM/MANAGER
SQL> CREATE TABLE table1 (x CHAR(1)) TABLESPACE ronly;
Table created.
SQL> ALTER TABLESPACE ronly READ ONLY;
Tablespace altered.
SQL> SELECT enabled, status, name FROM v$datafile;
ENABLED    STATUS  NAME
---------- ------- ----------------------------------------
READ WRITE SYSTEM  /home1/user481/ORADATA/u01/system01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u02/undotbs01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u03/users01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u03/indx01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u02/sample01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u01/querydata01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u02/example01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u04/data01.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u03/data02.dbf
READ WRITE ONLINE  /home1/user481/ORADATA/u06/index01.dbf
READ ONLY  ONLINE  /home1/user481/ORADATA/u01/ronly01.dbf

11 rows selected.
```

Practice 8: Solutions (continued)

4 b Attempt to create an additional table called TABLE2. Drop the first created table, TABLE1. What happens?

```
SQL> CREATE TABLE table2 ( y CHAR(1))
  2  TABLESPACE ronly;
CREATE TABLE table2 ( y CHAR(1))
*
ERROR at line 1:
ORA-01647: tablespace 'RONLY' is read only, cannot allocate
space in it
SQL> DROP TABLE table1;
Table dropped.
```

Practice 8: Solutions (continued)

5 Drop tablespace RONLY and the associated data file. Verify it.

Hints:

- Use the INCLUDING CONTENTS AND DATAFILES clause.
- View the V$TABLESPACE to verify the tablespace was dropped.
- Use the ! f from the SQL prompt to see a list of the data files in u01.

```
SQL> DROP TABLESPACE ronly INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
SQL> SELECT * FROM v$tablespace;
    TS# NAME          INC
------- ------------- ---
      0   SYSTEM        YES
      1   UNDOTBS       YES
      2   TEMP          YES
      3   USERS         YES
      4   INDX          YES
      5   SAMPLE        YES
      6   QUERY_DATA    YES
      7   EXAMPLE       YES
      8   DATA01        YES
      9   DATA02        YES
      10  INDEX01       YES
11 rows selected.
SQL> !ls $HOME/ORADATA/u01/*
/home1/user481/ORADATA/u01/ctrl01.ctl
/home1/user481/ORADATA/u01/querydata01.dbf
/home1/user481/ORADATA/u01/system01.dbf
```

Practice 8: Solutions (continued)

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.

.

```
SQL> ALTER SYSTEM SET
       DB_CREATE_FILE_DEST='$HOME/ORADATA/u05'
       SCOPE=MEMORY;
System altered.
SQL> CREATE TABLESPACE data03   DATAFILE SIZE 5M;
Tablespace created.
SQL> SELECT * FROM v$tablespace;
       TS# NAME                                           INC
---------- ---------------------------------------- ---
         0 SYSTEM                                     YES
         1 UNDOTBS                                    YES
         2 TEMP                                       YES
         3 USERS                                      YES
         4 INDX                                       YES
         5 SAMPLE                                     YES
         6 QUERY_DATA                                 YES
         7 EXAMPLE                                    YES
         8 DATA01                                     YES
         9 DATA02                                     YES
        10 INDEX01                                    YES
        12 DATA03                                     YES
12 rows selected.
SQL> !ls $HOME/ORADATA/u05
o1_mf_data03_yogw5908_.dbf
```

Practice 9: Solutions

1 As user SYSTEM, run the `lab09_01.sql` script to create tables and indexes.

```
SQL> @$HOME/STUDENT/LABS/lab09_01.sql
```

2 Identify the different types of segments in the database.

```
SQL> SELECT DISTINCT segment_type FROM dba_segments;
SEGMENT_TYPE
------------------
CACHE
CLUSTER
INDEX
INDEX PARTITION
LOBINDEX
LOBSEGMENT
NESTED TABLE
ROLLBACK
TABLE
TABLE PARTITION
TYPE2 UNDO
11 rows selected.
```

Practice 9: Solutions (continued)

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.

Hints:
- Select from DBA_EXTENTS.
- Use the segment_name, segment_type, max_extents, extents keywords.

```
SQL> COLUMN segment_name FORMAT a20
SQL> COLUMN segment_type FORMAT a15
SQL> SELECT segment_name,segment_type,
  2         max_extents, extents
  3  FROM   dba_segments
  4  WHERE  extents+5 > max_extents
  5  AND    segment_type<>'CACHE';

SEGMENT_NAME         SEGMENT_TYPE    MAX_EXTENTS    EXTENTS
-------------------- --------------- ----------- ----------
EMP                  TABLE                    10          8
```

4 Which files have space allocated for the EMP table?

```
SQL> SELECT DISTINCT f.file_name
  2  FROM   dba_extents e,dba_data_files f
  3  WHERE  e.segment_name='EMP'
  4  AND    e.file_id=f.file_id;

FILE_NAME
-------------------------------------
/home1/user481/ORADATA/u04/data01.dbf
```

Practice 9: Solutions (continued)

5 Run the `lab09_05.sql` script.

```
SQL> @$HOME/STUDENT/LABS/lab09_05.sql
```

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.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT tablespace_name,COUNT(*) AS fragments,
  2    SUM(bytes) AS total,
  3    MAX(bytes) AS largest
  4  FROM dba_free_space
  5  GROUP BY tablespace_name;
TABLESPACE_NAME FRAGMENTS  TOTAL        LARGEST
--------------- ---------- ------       ---------
 DATA01                  3  147456      126976
 DATA02                  1  1433600     1433600
 DATA03                  1  5177344     5177344
 EXAMPLE                 1  5242880     5242880
 INDEX01                1925696         925696
 INDX                    1    5120000    5120000
 QUERY_DATA              1     884736     884736
 SAMPLE                  1    9437184    9437184
 SYSTEM                  1   15224832   15224832
 UNDOTBS                 1   46071808   46071808
 USERS                   1    5177344    5177344

11 rows selected.
```

Practice 9: Solutions (continued)

7 List segments that will generate errors because of lack of space when they try to allocate an additional extent.

```
SQL> SELECT s.segment_name, s.segment_type, s.tablespace_name,
  2    s.next_extent
  3  FROM   dba_segments s
  4  WHERE  NOT EXISTS
  5    ( SELECT 1
  6      FROM dba_free_space f
  7      WHERE s.tablespace_name=f.tablespace_name
  8      HAVING  max(f.bytes) > s.next_extent ) ;

SEGMENT_NAME SEGMENT_TYPE TABLESPACE_NAME NEXT_EXTENT
------------ ------------ --------------- -----------
BIG_EMP      TABLE        DATA01              1048576
```

Practice 10: Solutions

1 Connect as user SYSTEM/MANAGER, and list the undo segments in tablespace UNDOTBS.

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
_SYSSMU9$
_SYSSMU10$

10 rows selected.
```

Practice 10: Solutions (continued)

2 Create undo tablespace UNDO2, size 15M, in $HOME/ORADATA/u03. List the undo segments in tablespace UNDO2.

```
SQL> CREATE UNDO TABLESPACE undo2
  2  DATAFILE '$HOME/ORADATA/u03/undo2.dbf' size 15M;
Tablespace created.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDO2';
SEGMENT_NAME
--------------------
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
_SYSSMU17$
_SYSSMU18$
_SYSSMU19$
_SYSSMU20$

10 rows selected.
```

Practice 10: Solutions (continued)

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.

```
SQL> @$HOME/STUDENT/LABS/lab10_03.sql
SQL> CONNECT hr/hr
Connected.
SQL> INSERT INTO departments
  2    (department_id,department_name)
  3  VALUES (9999,'x');
1 row created.
```

4 In the session in which you are connected as SYS, using the ALTER SYSTEM command, switch the UNDO tablespace from UNDOTBS to UNDO2, using SCOPE=BOTH.

```
SQL> ALTER SYSTEM SET undo_tablespace='UNDO2' SCOPE=BOTH;
System altered.
```

5 As SYS drop tablespace UNDOTBS. Use the INCLUDING CONTENTS AND DATAFILES clause. What happened?

```
SQL> DROP TABLESPACE undotbs
  2    INCLUDING CONTENTS AND DATAFILES;
DROP TABLESPACE undotbs INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-30013: undo tablespace 'UNDOTBS' is currently in use
```

Practice 10: Solutions (continued)

6 List the undo segments in tablespace UNDOTBS and their status. Compare this list to the list in step 1.

Hint: Query data dictionary view DBA_ROLLBACK_SEGS.

```
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
_SYSSMU9$
_SYSSMU10$
10 rows selected.
SQL> SELECT a.usn,a.name,b.status
 2  FROM v$rollname a, v$rollstat b
 3  WHERE a.name
 4   IN ( SELECT segment_name
 5        FROM   dba_segments
 6        WHERE  tablespace_name = 'UNDOTBS' )
 7  AND a.usn = b.usn;

  USN NAME         STATUS
----- ------------ ---------------
 10 _SYSSMU10$   PENDING OFFLINE
```

Practice 10: Solutions (continued)

7 In the session connected as HR, roll back the transaction and exit the session.

```
SQL> ROLLBACK;
Rollback complete.
SQL> EXIT;
Disconnected from Oracle9i Enterprise Edition Release
9.2.0.1.0 - 64bit Productn
With the Partitioning, Oracle Label Security, OLAP and Oracle
Data Mining optios
JServer Release 9.2.0.1.0 - Production
```

8 In the session connected as SYS drop tablespace UNDOTBS. What happened?

```
SQL> DROP TABLESPACE undotbs;
DROP TABLESPACE undotbs
*
ERROR at line 1:
ORA-30013: undo tablespace 'UNDOTBS' is currently in use
```

9 As SYS issue the following command:

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

Now drop tablespace UNDOTBS. What happened?

Note: There still may be a delay before the tablespace is drop.

```
SQL> ALTER SYSTEM SET undo_retention=0 SCOPE=MEMORY;
System altered.
SQL> DROP TABLESPACE undotbs
  2  INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
```

Practice 11: Solutions

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. **Note:** When using OEM be sure to set DATE_OF_DELY to NULL.
In addition, 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.

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

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE TABLE customers
  2    ( cust_code VARCHAR2(3),
  3      name VARCHAR2(50),
  4      region VARCHAR2(5) )
  5  TABLESPACE users;
Table created.
SQL> CREATE TABLE orders
  2    ( ord_id NUMBER(3),
  2      ord_date DATE,
  3      cust_code VARCHAR2(3),
  4      date_of_dely DATE )
  5  TABLESPACE users;
Table created.
```

Practice 11: Solutions (continued)

2 Run the script `lab11_02.sql` to insert rows into the tables.

```
SQL> @$HOME/STUDENT/LABS/lab11_02.sql
```

3 Find which files and blocks contain the rows for the ORDERS table.

Hint: Query data dictionary view `DBA_EXTENTS`.

```
SQL> SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  owner = 'SYSTEM'
  4  AND    segment_name = 'ORDERS'
  5  AND    segment_type = 'TABLE';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         3         25          8
```

4 Check the number of extents used by the `ORDERS` table .

```
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         1
```

Practice 11: Solutions (continued)

5 Allocate an extent manually, with default size, for the table ORDERS and confirm that the extent has been added as specified.

```
SQL> ALTER TABLE orders ALLOCATE EXTENT;
Table altered.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         2
```

6 Create another table, ORDERS2 as copy of the ORDERS table in the USERS tablespace, with MINEXTENTS equal to 10. Verify that the table has been created with the specified number of extents.

```
SQL> CREATE TABLE orders2
  2  TABLESPACE users
  3  STORAGE(MINEXTENTS 10)
  4  AS SELECT * FROM orders;
Table created.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS2'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
        10
```

Practice 11: Solutions (continued)

7 Truncate table ORDERS without releasing space and check the number of extents to verify extents have not been deallocated.

```
SQL> TRUNCATE TABLE orders REUSE STORAGE;
Table truncated.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         2
```

8 Truncate the ORDERS2 table, releasing space. How many extents does the table have now?

```
SQL> TRUNCATE TABLE orders2;
Table truncated.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS2'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
        10
```

9 Run the script lab11_09.sql to insert some rows into the ORDERS2 table.

```
SQL> @$HOME/STUDENT/LABS/lab11_09.sql
```

Practice 11: Solutions (continued)

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?

```
SQL> DESCRIBE orders2;
 Name                      Null?    Type
 ------------------------- -------- -----------------
 ORD_ID                             NUMBER(3)
 ORD_DATE                           DATE
 CUST_CODE                          VARCHAR2(3)
 DATE_OF_DELY                       DATE
SQL> ALTER TABLE orders2
  2  SET UNUSED COLUMN date_of_dely
  3  CASCADE CONSTRAINTS;
Table altered.
SQL> DESCRIBE orders2;
 Name                                      Null?    Type
 ------------------------- -------- -----------------
 ORD_ID                             NUMBER(3)
 ORD_DATE                           DATE
 CUST_CODE                          VARCHAR2(3)
```

11 Drop the unused column DATE_OF_DELY.

```
SQL> ALTER TABLE orders2
  2  DROP UNUSED COLUMNS;
Table altered.
```

12 Drop the ORDERS2 table.

```
SQL> DROP TABLE orders2;
Table dropped.
```

Practice 12: Solutions

1 You are considering creating indexes on the NAME and REGION columns of the CUSTOMERS table. What types of index are appropriate for the two columns? Create two indexes, naming them CUST_NAME_IDX and CUST_REGION_IDX, respectively, and placing them in the INDEX01 tablespace.

Hint: A B-tree index is suitable for a column with many distinct values, and a bitmap index is suitable for columns with only a few distinct values.
Note: CUSTOMERS table is in the SYSTEM schema.

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE INDEX cust_name_idx
  2  ON customers(name)
  3  TABLESPACE index01;
Index created.
SQL> CREATE BITMAP INDEX cust_region_idx
  2  ON system.customers(region)
  3  TABLESPACE index01;
Index created.
```

2 Move the CUST_REGION_IDX index to another tablespace.

Hint: The index can be rebuilt specifying a different tablespace.

```
SQL> ALTER INDEX cust_region_idx REBUILD
  2  TABLESPACE indx;
Index altered.
```

Practice 12: Solutions (continued)

3 Note the files and blocks used by the extents for the CUST_REGION_IDX index.

Hint: Use the view DBA_EXTENTS to get this information. **Note**: The owner is SYSTEM.

```
SQL> SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  segment_name='CUST_REGION_IDX'
  4  AND    owner='SYSTEM';
   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         4         17        125
```

4 Re-create the CUST_REGION_IDX index without dropping and re-creating it, and retain it in the same tablespace as before. Does the new index use the same blocks that were used earlier?

Hint: Rebuild the index.

Note: The new index does not reuse the same space as seen from the location of the extent after rebuild. This is because Oracle server builds a temporary index, drops the old one, and renames the temporary index.

```
SQL> ALTER INDEX cust_region_idx REBUILD;
Index altered.
SQL> SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  segment_name='CUST_REGION_IDX'
  4  AND    owner='SYSTEM';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         4        142        125
```

Practice 12: Solutions (continued)

5 a As user SYSTEM, run the script `lab12_05a.sql` to create and populate the NUMBERS table.

```
SQL> @$HOME/STUDENT/LABS/lab12_05a.sql
```

b Query the NUMBERS table to find the number of distinct values in the two columns in the table.

```
SQL> SELECT count(DISTINCT no) NO,
  2    count(DISTINCT odd_even) OE
  3  FROM   numbers;

        NO         OE
---------- ----------
     10000          2
```

c Using uniform extent sizes of 4KB, create two indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively. Place the indexes in the INDEX01 tablespace. Check the total sizes of the indexes and write the number of blocks in box below.

Hint: Use PCTINCREASE equal to zero to create extents of equal size. Check the total blocks allocated to the extents from DBA_SEGMENTS.

Index	Column	Blocks
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

Practice 12: Solutions (continued)

5 c (continued)

```
SQL> CREATE INDEX numb_oe_idx
  2  ON numbers(odd_even)
  3  TABLESPACE index01;
Index created.
SQL> CREATE INDEX numb_no_idx
  2  ON numbers(no)
  3  TABLESPACE index01;
Index created.
SQL> COLUMN segment_name FORMAT a15
SQL> SELECT segment_name, blocks
  2  FROM   dba_segments
  3  WHERE  segment_name LIKE 'NUMB%'
  4  AND    segment_type='INDEX';

SEGMENT_NAME        BLOCKS
--------------- ----------
NUMB_OE_IDX             40
NUMB_NO_IDX             46
```

Practice 12: Solutions (continued)

5 d After you have recorded the blocks above, drop the two indexes, NUMB_OE_IDX and NUMB_NO_IDX . Using uniform extent sizes of 4KB, create bitmap indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively. Place the indexes in the INDEX01 tablespace. Re-execute the query to check the total blocks allocated to the extents from DBA_SEGMENTS. Check the total sizes of the indexes and write in box below.

Index	Column	Blocks
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

```
SQL> DROP INDEX numb_oe_idx;
Index dropped.
SQL> DROP INDEX numb_no_idx;
Index dropped.
SQL> CREATE BITMAP INDEX numb_oe_idx
  2  ON numbers(odd_even)
  3  TABLESPACE index01;
Index created.
SQL> CREATE BITMAP INDEX numb_no_idx
  2  ON numbers(no)
  3  TABLESPACE index01;
Index created.
SQL> SELECT segment_name, blocks
  2  FROM   dba_segments
  3  WHERE  segment_name LIKE 'NUMB%'
  4  AND    segment_type='INDEX';

SEGMENT_NAME        BLOCKS
--------------- ----------
NUMB_OE_IDX              2
NUMB_NO_IDX             72
```

What can you conclude about the relationship between cardinality and sizes of the two types of indexes?

Answer: It can be seen from the results that a bitmap index is compact for a low-cardinality column, while a B-tree index is compact for a high-cardinality column.

Practice 13: Solutions

1 Examine and run the script `lab13_01.sql` to create the constraints.

```
SQL> @$HOME/STUDENT/LABS/lab13_01.sql
```

2 As user SYSTEM, query the data dictionary to:

a Check for constraints, whether they are deferrable, and their status.

Hint: Use the `DBA_CONSTRAINTS` view to get this information.

```
SQL> COLUMN constraint_name FORMAT a25
SQL> COLUMN table_name      FORMAT a10
SQL> COLUMN constraint_type FORMAT a1
SQL> COLUMN deferrable      FORMAT a15
SQL> COLUMN status          FORMAT a10
SQL> SELECT constraint_name, table_name,
  2    constraint_type, deferrable, status
  3  FROM   dba_constraints
  4  WHERE  table_name IN
  5 ('PRODUCTS','ORDERS','CUSTOMERS')
  6  AND owner='SYSTEM';

CONSTRAINT_NAME           TABLE_NAME C DEFERRABLE      STATUS
------------------------- ---------- - --------------- --------
CUSTOMERS_REGION_CK       CUSTOMERS  C NOT DEFERRABLE  ENABLED
CUSTOMERS_CUST_CODE_PK    CUSTOMERS  P DEFERRABLE      ENABLED
ORDERS_CUST_CODE_FK       ORDERS     R DEFERRABLE      ENABLED
ORDERS_DATE_OF_DELY_CK    ORDERS     C NOT DEFERRABLE  ENABLED
ORDERS_ORD_ID_PK          ORDERS     P NOT DEFERRABLE  ENABLED
PRODUCTS_PROD_CODE_UK     PRODUCTS   U DEFERRABLE     DISABLED

6 rows selected.
```

2 b Check the names and types of indexes created to validate the constraints.

Hint: The indexes are only created for primary key and unique constraints and have the same name as the constraints

```
SQL> SELECT index_name,table_name,uniqueness
  2  FROM   dba_indexes
  3  WHERE  index_name in
  4    ( SELECT constraint_name
  5      FROM   dba_constraints
  6      WHERE  table_name IN ('PRODUCTS', 'ORDERS',
  7                            'CUSTOMERS')
  8      AND    owner='SYSTEM'
  9      AND    constraint_type in ('P','U')
 10     ) ;

INDEX_NAME                     TABLE_NAME UNIQUENES
------------------------------ ---------- ---------
CUSTOMERS_CUST_CODE_PK         CUSTOMERS  NONUNIQUE
ORDERS_ORD_ID_PK               ORDERS     UNIQUE
```

3 As user SYSTEM, run the script `lab13_03.sql` to insert two records into the PRODUCTS table.

```
SQL> @$HOME/STUDENT/LABS/lab13_03.sql
```

Practice 13: Solutions (continued)

4 Enable the unique constraint on the PRODUCTS table. Was it successful?

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk;
ALTER TABLE system.products
*
ERROR at line 1:
ORA-02299: cannot validate (SYSTEM.PRODUCTS_PROD_CODE_UK) -
duplicate keys found
```

5 a Ensure any new rows added to the table do not violate the constraint on the PRODUCTS table.

Hint: This can be done by enabling the constraint NOVALIDATE.

```
SQL> ALTER TABLE system.products
  2  ENABLE NOVALIDATE CONSTRAINT products_prod_code_uk;
Table altered.
```

b Query the data dictionary to verify the effect of the change.

```
SQL> SELECT constraint_name, table_name,
  2    constraint_type, validated, status
  3  FROM   dba_constraints
  4  WHERE  table_name = 'PRODUCTS'
  5  AND    owner='SYSTEM';

CONSTRAINT_NAME           TABLE_NAME C VALIDATED     STATUS
----------------------     ---------- - ------------- ------
PRODUCTS_PROD_CODE_UK     PRODUCTS   U NOT VALIDATED ENABLED
```

5 c Test that the constraint disables inserts that violate the change by adding a row with the following values:

PRODUCT_ID	PRODUCT_DESCRIPTION	LIST_PRICE
4000	Monitor	3000

```
SQL> INSERT INTO system.products
  2  VALUES(4000,'Monitor',3000);
INSERT INTO system.products
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.PRODUCTS_PROD_CODE_UK)
violated
```

6 Take the necessary steps to identify existing constraint violations in the PRODUCTS table, modify product codes as needed, and guarantee that all existing as well as new data do not violate the constraint. (Assume that the table has several thousands of rows and it is too time-consuming to verify each row manually.)

Hint: Use the following steps:

a Create the EXCEPTIONS table.

```
SQL> CONNECT system/manager
Connected.
SQL> @?/rdbms/admin/utlexcpt
```

b Run the command to enable the constraint and trap the exceptions.

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk
  3  EXCEPTIONS INTO system.exceptions;
ALTER TABLE system.products
*
ERROR at line 1:
ORA-02299: cannot validate (SYSTEM.PRODUCTS_PROD_CODE_UK) -
duplicate keys found
```

6 c Use the ROWID in the EXCEPTIONS table to list the rows in the PRODUCTS table that violate the constraint. Do not list LOB columns.

```
SQL> SELECT rowid, prod_code, description
  2  FROM   system.products
  3  WHERE  rowid IN
  4    ( SELECT row_id
  5      FROM   exceptions
  6      WHERE  table_name='PRODUCTS' ) ;
ROWID              PROD_CODE DESCRIPTION
------------------ ---------- ------------------------------
AAAF/MAAJAAAAASAAA       4000 UNIX Monitor
AAAF/MAAJAAAAASAAB       4000 NT Monitor
```

d Rectify the errors.

```
SQL> UPDATE   system.products
  2  SET      prod_code='4001'
  3  WHERE    rowid = ( SELECT max(row_id)
  4                     FROM   exceptions
  5                     WHERE  table_name='PRODUCTS' ) ;
1 row updated.
```

e Enable the constraint.

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk
  3  EXCEPTIONS INTO system.exceptions;
Table altered.
```

Practice 13: Solutions (continued)

7 Run the script `lab13_07.sql` to insert rows into the table. Were the inserts successful?

```
SQL> @$HOME/STUDENT/LABS/lab13_07.sql
SQL> INSERT INTO system.orders
  2  VALUES (800,'01-JAN-98','J01',NULL);
INSERT INTO system.orders
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.ORDERS_CUST_CODE_FK)
violated - parent key not found
SQL> INSERT INTO system.customers
  2  VALUES ('J01','Sports Store', 'East');
1 row created.
SQL> ROLLBACK;
Rollback complete.
```

8 Now examine the script `lab13_08`. Notice that this script also performs the inserts in the same sequence. Run the script and check if it executes successfully.

```
SQL> @$HOME/STUDENT/LABS/lab13_08.sql
SQL> ALTER SESSION SET CONSTRAINTS=deferred;
Session altered.
SQL> INSERT INTO system.orders
  2  VALUES (800,'01-JAN-98','J01',NULL);
1 row created.
SQL> INSERT INTO system.customers
  2  VALUES ('J01','Sports Store', 'East');
1 row created.
SQL> COMMIT;
Commit complete.
```

9 Truncate the `CUSTOMERS` table. Was it successful?

```
SQL> TRUNCATE TABLE system.customers;
TRUNCATE TABLE system.customers
                      *
ERROR at line 1:
ORA-02266: unique/primary keys in table referenced by enabled
foreign keys
```

Practice 14: Solutions

1 a Run the `lab14_01.sql` script to create user `Jeff`.

```
SQL> @$HOME/STUDENT/LABS/lab14_01.sql
```

b Connect as user SYS and enable password management by running script `$ORACLE_HOME/rdbms/admin/utlpwdmg.sql`.

```
SQL> connect / as sysdba
SQL> @?/rdbms/admin/utlpwdmg dmg
```

2 Try to change the password for user `Jeff` to `Jeff`. What happens?

```
SQL> ALTER USER jeff IDENTIFIED BY jeff;
ALTER USER jeff IDENTIFIED BY jeff
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20001: Password same as or similar to user
```

3 Try changing the password for `Jeff` to follow the password management format.

Hint: Password should contain at least one digit, one character, and one punctuation.

```
SQL> ALTER USER jeff
  2  IDENTIFIED BY super1$;
User altered.
```

Practice 14: Solutions (continued)

4 Alter the DEFAULT profile to ensure the following applies to users assigned the DEFAULT profile:

- After two login attempts, the account should be locked.
- The password should expire after 30 days.
- The same password should not be reused for at least one minute.
- The account should have a grace period of five days to change an expired password.
- Ensure that the requirements given have been implemented.

Hints:

- Use the ALTER PROFILE command to change the default profile limits.
- Query the data dictionary view DBA_PROFILES to verify the result.

```
SQL> ALTER PROFILE default LIMIT
  2     FAILED_LOGIN_ATTEMPTS 2
  3     PASSWORD_LIFE_TIME 30
  4     PASSWORD_REUSE_TIME 1/1440
  5     PASSWORD_GRACE_TIME 5;
Profile altered.
SQL> SELECT resource_name, limit
  2  FROM   dba_profiles
  3  WHERE  profile='DEFAULT'
  4  AND    resource_type='PASSWORD';
RESOURCE_NAME                    LIMIT
-------------------------------- ----------------
FAILED_LOGIN_ATTEMPTS            2
PASSWORD_LIFE_TIME               30
PASSWORD_REUSE_TIME              .0006
PASSWORD_REUSE_MAX               UNLIMITED
PASSWORD_VERIFY_FUNCTION         VERIFY_FUNCTION
PASSWORD_LOCK_TIME               .0006
PASSWORD_GRACE_TIME              5

7 rows selected.
```

Practice 14: Solutions (continued)

5 Log in as user `Jeff` supplying an invalid password. Try this twice, then log in again, this time supplying the correct password. What happens?

```
SQL> CONNECT jeff/superman
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.
SQL> CONNECT jeff/super
ERROR:
ORA-01017: invalid username/password; logon denied
SQL> CONNECT jeff/super1$
ERROR:
ORA-28000: the account is locked
```

Practice 14: Solutions (continued)

6 Using data dictionary view DBA_USERS verify user Jeff is locked. Unlock the account for user Jeff. After unlocking user Jeff connect as Jeff.

Hint: Execute the ALTER USER command to unlock the account.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT username, account_status
  2  FROM   dba_users;
USERNAME        ACCOUNT_STATUS
--------------- ---------------
SYS              OPEN
SYSTEM           OPEN
DBSNMP           OPEN
OUTLN            OPEN
OE               OPEN
GC9201DBAIS3U481 OPEN
HR               OPEN
MDSYS            OPEN
JEFF             LOCKED(TIMED)
9 rows selected.
SQL> ALTER USER jeff
  2  ACCOUNT UNLOCK;
User altered.
SQL> CONNECT jeff/super1$
Connected.
```

Practice 14: Solutions (continued)

7 Connect as user SYS and disable password checks for the DEFAULT profile.

Hint: Execute the ALTER PROFILE command to disable the password checks.

```
SQL> ALTER PROFILE default LIMIT
  2     FAILED_LOGIN_ATTEMPTS     UNLIMITED
  3     PASSWORD_LIFE_TIME        UNLIMITED
  4     PASSWORD_REUSE_TIME       UNLIMITED
  5     PASSWORD_REUSE_MAX        UNLIMITED
  6     PASSWORD_VERIFY_FUNCTION  NULL
  7     PASSWORD_LOCK_TIME        UNLIMITED
  8     PASSWORD_GRACE_TIME       UNLIMITED;

Profile altered.
```

8 Log in to user Jeff supplying an invalid password. Try this twice, then log in again, this time supplying the correct password. What happens? Why?

```
SQL> CONNECT jeff/superman
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.
SQL> CONNECT jeff/super
ERROR:
ORA-01017: invalid username/password; logon denied
SQL> CONNECT jeff/super1$
Connected.
```

Answer: Account is not locked on the third attempt to log in when using the correct password as it was in step 5. This is because the PASSWORD_VERIFY_FUNCTION was disabled in step 7 when set to NULL.

Practice 15: Solutions

1 Create user Bob with a password of CRUSADER. Make sure that any objects and temporary segments created by Bob are not created in the system tablespace. Also, ensure that Bob can log in and create objects up to one megabyte in size in the USERS and INDX tablespaces. Use lab15_01.sql script to grant Bob the ability to create sessions.

Hint: Assign Bob the default tablespace of USERS and temporary tablespace TEMP.

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE USER bob
  2     IDENTIFIED BY crusader
  3     DEFAULT TABLESPACE USERS
  4     TEMPORARY TABLESPACE temp
  5     QUOTA 1M ON USERS
  6     QUOTA 1M ON INDX;
User created.
SQL> @$HOME/STUDENT/LABS/lab15_01.sql
SQL> GRANT CREATE SESSION TO bob;
Grant succeeded.
```

Practice 15: Solutions (continued)

2 Create a user `Emi` with a password of `Mary`. Make sure that any objects and sort segments created by `Emi` are not created in the system tablespace.

```
SQL> CREATE USER emi
  2     IDENTIFIED BY mary
  3     DEFAULT TABLESPACE users
  4     TEMPORARY TABLESPACE temp;
User created.
```

3 Display the information on `Bob` and `Emi` from the data dictionary.

Hint: This can be obtained by querying `DBA_USERS`.

```
SQL> SELECT username, default_tablespace,
  2     temporary_tablespace
  3  FROM   dba_users
  4  WHERE  username IN ('BOB', 'EMI');

USERNAME   DEFAULT_TABLESPACE   TEMPORARY_TABLE
---------- -------------------- ---------------
EMI        USERS                TEMP
BOB        USERS                TEMP
```

Practice 15: Solutions (continued)

4 From the data dictionary, display the information on the amount of space that Bob can use in tablespaces.

Hint: This can be obtained by querying DBA_TS_QUOTAS.

```
SQL> COLUMN tablespace_name FORMAT a15
SQL> COLUMN user            FORMAT a10
SQL> SELECT *
  2  FROM   dba_ts_quotas
  3  WHERE username = 'BOB';
TABLESPACE_NAME USERNAME BYTES  MAX_BYTES BLOCKS MAX_BLOCKS
--------------- -------- ----- ---------- ------ ----------
INDX            BOB          0    1048576      0        256
USERS           BOB          0    1048576      0        256
```

5 a As user Bob change his temporary tablespace. What happens?

```
SQL> connect bob/crusader
Connected.
SQL> ALTER USER bob
  2  TEMPORARY TABLESPACE users;
ALTER USER bob
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Practice 15: Solutions (continued)

5 b As user Bob, change his password to Sam.

```
SQL> CONNECT bob/crusader
Connected.
SQL> ALTER USER bob
  2  IDENTIFIED BY sam;
User altered.
```

6 As user SYSTEM, remove Bob's quota on his default tablespace.

```
SQL> CONNECT system/manager
Connected.
SQL> ALTER USER bob QUOTA 0 ON users;
User altered.
```

Practice 15: Solutions (continued)

7 Remove Emi's account from the database.

```
SQL> DROP USER emi;
User dropped.
```

8 Bob has forgotten his password. Assign him a password of OLINK and require that Bob change his password the next time he logs on.

```
SQL> ALTER USER bob
  2  IDENTIFIED BY olink
  3  PASSWORD EXPIRE;
User altered.
```

Practice 16: Solutions

1 As user SYSTEM, create user Emi with the password of abcd12. Give her the capability to log on to the database and create schema objects. Assign her to the default tablespace DATA01, and the temporary tablespace TEMP. Make her quota on DATA01 1M.

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE USER emi
  2  IDENTIFIED BY "abcd12"
  3  DEFAULT TABLESPACE data01
  4  TEMPORARY TABLESPACE temp
  5  QUOTA 1M ON data01;
User created.
SQL> GRANT create session, create table TO emi;
Grant succeeded.
```

2 **a** Run the script lab16_02a.sql to connect as Emi and create the tables CUSTOMERS1 and ORDERS1.

```
SQL> @$HOME/STUDENT/LABS/lab16_02a.sql;
```

Practice 16: Solutions (continued)

2 b Connect as SYSTEM and copy the data from SYSTEM.CUSTOMERS to Emi's CUSTOMERS1 table. Verify that records have been inserted.

```
SQL> CONNECT system/manager
Connected.
SQL> INSERT INTO emi.customers1
  2  SELECT *
  3  FROM system.customers;
9 rows created.
SQL> SELECT * FROM emi.customers1;

CUS NAME                                               REGIO
--- -------------------------------------------------- -----
A01 TKB SPORT SHOP                                     West
A02 VOLLYRITE                                          North
A03 JUST TENNIS                                        North
A04 EVERY MOUNTAIN                                     South
A05 SHAPE UP                                           South
A06 SHAPE UP                                           West
A07 WOMENS SPORTS                                      South
A08 NORTH WOODS HEALTH AND FITNESS SUPPLY CENTER       East
J01 Sports Store                                       East

9 rows selected.
```

c As user SYSTEM give Bob the ability to select from Emi's CUSTOMERS1 table. What happens?

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT select ON emi.customers1 TO bob;
GRANT succeeded.
```

Practice 16: Solutions (continued)

3 Reconnect as `Emi` and give `Bob` the ability to select from `Emi`'s `CUSTOMERS1` table. Also, enable `Bob` to give the select capability to other users. As user `SYSTEM`, examine the data dictionary views that record these actions.

Hint: Query the data dictionary view `DBA_TAB_PRIVS` to examine.

```
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers1
  2  TO bob WITH GRANT OPTION;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> COLUMN grantee    FORMAT a8
SQL> COLUMN owner      FORMAT a8
SQL> COLUMN table_name FORMAT a10
SQL> COLUMN grantor    FORMAT a8
SQL> COLUMN privilege  FORMAT a10
SQL> COLUMN grantable  FORMAT a3
SQL> COLUMN hierarchy   FORMAT a3
SQL> SELECT *
  2  FROM   dba_tab_privs
  3  WHERE  grantee='BOB';
GRANTEE OWNER TABLE_NAME GRANTOR  PRIVILEGE  GRANT HIERARCHY
------- ----- ---------- -------- ---------- ----- ---------
BOB      EMI  CUSTOMERS1 EMI      SELECT     YES   NO
```

Practice 16: Solutions (continued)

4 Create user `Trevor` identified by `diamond1$` with the capability to log on to the database.

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE USER trevor IDENTIFIED BY “diamond1$";
User created.
SQL> GRANT create session TO trevor;
Grant succeeded.
```

5 a As `Bob`, enable `Trevor` to access `Emi`'s `CUSTOMERS1` table.

Note: A password has expired message will be received due to actions in step 8 in Lesson 15. Give `Bob` the new password `aaron$1`.

```
SQL> CONNECT bob/olink
ERROR:
ORA-28001: the password has expired
Changing password for bob
New password: aaron1$
Retype new password: aaron1$
Password changed
Connected.
SQL> GRANT select ON emi.customers1 TO trevor;
Grant succeeded.
```

Practice 16: Solutions (continued)

5 b As Emi, remove Bob's privilege to read Emi's CUSTOMERS1 table.

```
SQL> CONNECT emi/abcd12
Connected.
SQL> REVOKE select ON customers1 FROM bob;
Revoke succeeded.
```

c As Trevor, query Emi's CUSTOMERS1 table. What happens?

```
SQL> CONNECT trevor/diamond1$
Connected.
SQL> SELECT *
  2  FROM emi.customers1;
FROM emi.customers1
         *
ERROR at line 2:
ORA-00942: table or view does not exist
```

Practice 16: Solutions (continued)

6 Enable `Emi` to start up and shut down the database without the ability to create a new database.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT sysoper TO emi;
Grant succeeded.
```

Practice 17: Solutions

1 Examine the data dictionary view and list the system privileges of the RESOURCE role.

```
SQL> CONNECT system/manager
Connected.
SQL> COLUMN privilege FORMAT a20
SQL> COLUMN grantee   FORMAT a10
SQL> SELECT  *
  2  FROM    dba_sys_privs
  3  WHERE   grantee = 'RESOURCE';
GRANTEE    PRIVILEGE            ADM
---------- -------------------- ---
RESOURCE   CREATE TYPE          NO
RESOURCE   CREATE TABLE         NO
RESOURCE   CREATE CLUSTER       NO
RESOURCE   CREATE TRIGGER       NO
RESOURCE   CREATE OPERATOR      NO
RESOURCE   CREATE SEQUENCE      NO
RESOURCE   CREATE INDEXTYPE     NO
RESOURCE   CREATE PROCEDURE     NO
8 rows selected.
```

Practice 17: Solutions (continued)

2 Create a role called DEV, which will enable a user assigned the role to create a table, create a view, and select from Emi's CUSTOMERS1 table.

```
SQL> CREATE ROLE dev;
Role created.
SQL> GRANT create table, create view TO dev;
Grant succeeded.
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers1 TO dev;
Grant succeeded.
```

Practice 17: Solutions (continued)

3 a Assign the RESOURCE and DEV roles to Bob, but make only the RESOURCE role automatically enabled when he logs on.

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT dev, resource TO bob;
Grant succeeded.
SQL> ALTER USER bob
  2     DEFAULT ROLE resource;
User altered.
```

b Give Bob the ability to read all the data dictionary information.

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT select_catalog_role TO bob;
Grant succeeded.
```

Practice 17: Solutions (continued)

4 Bob needs to check the undo segments that are currently used by the instance. Connect as Bob and list the undo segments used.

Hint: Use SET ROLE SELECT_CATALOG_ROLE

```
SQL> CONNECT bob/aaron1$
Connected.
SQL> SET ROLE select_catalog_role;
Role set.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  status='ONLINE';
SEGMENT_NAME
---------------
SYSTEM
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
_SYSSMU17$
_SYSSMU18$
_SYSSMU19$
_SYSSMU20$

11 rows selected.
```

5 As SYSTEM, try to create a view CUST_VIEW on Emi's CUSTOMERS1 table. What happens?

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2    SELECT *
  3    FROM   emi.customers1;
   FROM   emi.customers1
               *
ERROR at line 3:
ORA-01031: insufficient privileges
```

Practice 17: Solutions (continued)

6 As user Emi grant select on CUSTOMERS1 to SYSTEM. As SYSTEM create a view CUST_VIEW on Emi's CUSTOMERS1 table.

```
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers1 TO system;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2     SELECT *
  3     FROM   emi.customers1;

View created.
```

Practice 19: Solutions

1 a Examine the data files to be used in the load in order to become familiar with the control and data file formats.

- Examine the following files if using SQL*Plus:
  `lcase1.ctl`, `lcase2.ctl`, and `lcase2.dat`
- Examine the following files if using Oracle Enterprise Manager:
  `OEMsqlcase1.ctl`, `OEMsqlcase2.ctl`, and `OEMsqlcase2.`

b As user HR, run the `lab19_01.sql` script to create the DEPARTMENTS2 table.

```
SQL> CONNECT hr/hr
Connected.
SQL> @$HOME/STUDENT/LABS/lab19_01.sql
Table created.
```

2 a Run SQL*Loader using the Conventional Path method to load data into the DEPARTMENTS2 table. Use the following control file:

- Using SQL*Plus: `lcase1.ctl`
- Using Oracle Enterprise Manager: `OEMsqlcase1.ctl`

Files are located in the LABS directory.

Note: This runs using a control file that contains the input data.

```
$ cd $HOME/STUDENT/LABS
$ sqlldr hr/hr control=lcase1.ctl log=$HOME/lcase1.log
  SQL*Loader: Release 9.2.0.1.0 - Production on Wed Jul 24
  23:18:29 2001
  (c) Copyright 2001 Oracle Corporation.  All rights
  reserved.
  Commit point reached - logical record count 27
  ...
```

Practice 19: Solutions (continued)

2 b Examine the log file using operating system command.

```
$ cd $HOME
$ more lcase1.log
   -- Note:  Path used:  Conventional
```

c Query the DEPARTMENTS2 table to check the data.

```
SQL> SELECT * FROM DEPARTMENTS2;
   DEPT_ID DEPT_NAME
   ------- ------------------
        10 Administration
        20 Marketing
        30 Purchasing
        40 Human Resources
        50 Shipping
        60 IT
        70 Public Relations
        …
        27 rows selected
```

3 Delete all the records in the DEPARTMENTS2 table.

```
SQL> TRUNCATE TABLE departments2;
Table truncated.
```

Practice 19: Solutions (continued)

4 a Run SQL*Loader in Direct-Path mode to load data into the DEPARTMENTS2 table. Use the following control files:

- Using SQL*Plus: `lcase2.ctl`
- Using Oracle Enterprise Manager: `OEMsqlcase2.ctl`

Files are located in the LABS directory.

Note: This runs using an input data file to load data.

```
$ cd $HOME/STUDENT/LABS
$ sqlldr hr/hr control=lcase2.ctl direct=true
    log=$HOME/STUDENT/LABS/lcase2.log
  SQL*Loader: Release 9.2.0.1.0 - Production on Wed Jul 24
  10:40:10 2001
  (c) Copyright 2001 Oracle Corporation.  All rights
  reserved.
  Load completed - logical record count 27.
  ...
```

b Examine the log file using operating system command.

```
$ cd $HOME
$ more lcase2.log
  -- Note:  Path used:  Direct
```

c Query the DEPARTMENTS2 table to check the data.

```
SQL> SELECT * FROM DEPARTMENTS2;
   DEPT_ID DEPT_NAME
   ------- ------------------
        10 Administration
        20 Marketing
        30 Purchasing
        40 Human Resources
        50 Shipping
        60 IT
        70 Public Relations
        80 Sales
        90 Executive
        …
        27 rows selected.
```

Practice 19: Solutions (continued)

5 a As user HR, create a table EMPLOYEES2 as select from the EMPLOYEES table. Truncate the table. Then select from the EMPLOYEES2 table to verify no data is in the table.

```
SQL> CONNECT hr/hr
SQL> CREATE TABLE employees2
  2  AS SELECT * FROM employees;
Table created.
SQL> TRUNCATE TABLE employees2;
Table truncated.
SQL> SELECT * FROM employees2;
no rows selected
```

b Perform a direct-load insert into EMPLOYEES2 from EMPLOYEES and query EMPLOYEES2 to verify the load.

```
SQL> INSERT /*+ append */ INTO employees2
  2  NOLOGGING
  3  SELECT * from employees;
107 rows created.
SQL> COMMIT;
SQL> SELECT employee_id, first_name, last_name
  2  FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ------------------ -------------------------
139   John                  Seo
140   Joshua                Patel
141   Trenna                Rajs
142   Curtis                Davies
…
107 rows selected.
```

Practice 19: Solutions (continued)

6 a Truncate the EMPLOYEES2 table once again. Then select from the EMPLOYEES2 table to verify no data is in the table.

```
SQL> TRUNCATE TABLE employees2;
Table truncated.
SQL> SELECT * FROM employees2;
no rows selected
```

b Restore the data with a parallel direct-load insert from the EMPLOYEES table. Specify a degree of parallelism of two. Verify the data.

```
SQL> ALTER SESSION ENABLE PARALLEL DML;
Session altered
SQL> INSERT /*+ parallel(employees2,2) */
  2  INTO employees2 NOLOGGING
  3  SELECT * FROM employees;
107 rows created.
SQL> COMMIT;
Commit complete.
SQL> SELECT employee_id, first_name, last_name
  2 FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ----------------- -------------------------
139   John                  Seo
140   Joshua                Patel
141   Trenna                Rajs
142   Curtis                Davies
…
107 rows selected.
```

Practice 20: Solutions

1 Check the database and the national character set.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT parameter, value
  2  FROM   nls_database_parameters
  3  WHERE  parameter LIKE '%CHARACTERSET%';

PARAMETER                      VALUE
------------------------------ --------------
NLS_CHARACTERSET               WE8ISO8859P1
NLS_NCHAR_CHARACTERSET         AL16UTF16
```

2 Which are valid values for the database character set.

```
SQL> SELECT   value
  2  FROM     v$nls_valid_values
  3  WHERE    parameter = 'CHARACTERSET'
  4  ORDER BY value;

VALUE
--------------------
AL16UTF16
AL24UTFFSS
AL32UTF8
AR8ADOS710
AR8ADOS710T
AR8ADOS720
AR8ADOS720T
.
.
.
257 rows selected.
```

Practice 20: Solutions (continued)

3 Make sure that all dates in this session are displayed using a 4-digit year. Change NLS_LANGUAGE to FRENCH. Select sysdate from DUAL.

```
SQL> ALTER SESSION SET nls_date_format = 'DD-MON-YYYY';
Session altered.
SQL> ALTER SESSION SET nls_language = FRENCH;
Session modifiée.
SQL> SELECT sysdate
  2  FROM   dual;
SYSDATE
-----------
24-Jul-2002
```

Practice Solutions for Oracle Enterprise Manager

Practice Solution Conventions

The following solutions are designed for the student wishing to use Oracle Enterprise Manager.

Using OEM Console

When a specific OEM Console solution can be used as an alternative to using SQL*Plus, notations have been made under the heading Using OEM Console.

Using SQL*Plus and OS Command Line

On occasion you will need to perform some tasks using SQL*Plus and/or the operating system command line. In Practice 2, you will start an SQL*Plus session. Continue to keep the SQL*Plus session open to accomplish those tasks done in SQL*Plus. When performing tasks that require the operating system command line, an [!] will be used from the SQL*Plus command line to get you out of SQL*Plus and into the operating system command line mode. Once you have completed your work on the operating system command line you can return to the SQL*Plus command line by using `Exit`. For example: Remove the current password file located at `$HOME/ADMIN/PFILE`.

```
SQL > !rm $HOME/ADMIN/PFILE/orapw$ORACLE_SID
$ > exit
```

Running SQL Lab Scripts

Lab scripts have been included in the lab exercises to perform certain tasks for you. The task being accomplished may be one that you have not learned at the point of the exercise or it may be that the task is being accomplished for your convenience. When asked to run a `.sql` script, take time to review the script before running it. This will provide you with a thorough understanding of what is actually being created for you, and how it affects the task you are about to accomplish in the practice.

Connecting as SYS

When asked to connect as user `SYS` it will always be with `SYSDBA` privilege.

```
CONNECT / AS SYSDBA
```

Starting and Shutting Down the Database

Although the OEM Console can be used to start and shutdown the database, for the purposes of this course in the Oracle classroom, you will use SQL*Plus to start and shutdown the database.

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.

Answer: D

2 Which one of the following memory areas is not part of the SGA?

a Database Buffer Cache
b PGA
c Redo Log Buffer
d Shared Pool

Answer: B

3 Which three 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.
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.

Answer: B,C,D

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
d Shared Pool

Answer: D

5 The primary purpose of the Redo Log Buffer 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

Answer: False, A PGA is a memory region that contains data and control information for a single server process or a single background process.

Practice 1: Solutions (continued)

7 Which of the following becomes available when an Oracle instance is started?

a User process
b Server process
c Background processes

Answer: C

8 Identify five mandatory background processes.

_____
_____
_____
_____
_____

Answer: DBWR, LGWR, PMON, SMON, CKPT

9 Match the process with its task.

a Database Writer	1. Assists with writing to the data file headers
b Log Writer	2. Responsible for instance recovery
c System Monitor	3. Cleans up after failed processes
d Process Monitor	4. Records database changes for recovery purposes
e Checkpoint	5. Writes dirty buffers to the data files

Answer: a = 5, b = 4, c = 2, d = 3, e =1

10 The physical structure of an Oracle database consists of control files, data files, and online redo log files.

a True
b False

Answer: True

11 Place the following structures in order of hierarchy beginning with database.

a Tablespaces
b Extent
c Segment
d Database
e Block

Answer: D,A,C,B,E

Practice 1: Solutions (continued)

12 Identify the components of an Oracle server.

_____

_____

Answer: Oracle Instance, Oracle database

13 Identify the components of an Oracle Instance.

_____

_____

Answer: SGA, Background processes

14 Identify three file types that make up an Oracle database.

_____

_____

_____

Answer: data files, control files, online redo log files

Practice 2: Solutions

This practice is Instructor led. Your instructor will provide you with login accounts and walk you through logging into your account. Write the information provided by your Instructor below.

Host name: _____

SID name: _____

1 Connect to SQL*Plus as SYSDBA.

Hint:

- Connect to your account using instructions provided by your Instructor.
- Start SQL*Plus.
- Connect as the user SYS.

```
$ sqlplus /nolog
SQL*Plus: Release 9.2.0.1.0 - Production on Mon Aug 5 10:52:10
2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights
reserved.
SQL> CONNECT / AS SYSDBA
Connected.
```

2 Using SQL*Plus, run the following query to verify the connection to the database has been made.

```
SQL> SELECT * FROM DUAL;
D
-
X
```

3 Launch Oracle Enterprise Manager in Standalone mode.

- Navigate to Start > Programs > Oracle-OraHome92 > Enterprise Manager Console.
- Select Launch standalone option.
- Click OK.

Practice 2: Solutions (continued)

4 If you are in an Oracle classroom you must perform the following four steps that are particular to the Oracle classroom setup:

a. Click the `omsconfig file` update icon on the desktop.
   Enter the name of the UNIX server your class is using. Your instructor will provide the server name. Enter it exactly as it is given to you. This is a case-sensitive entry.

b. Open an MSDOS window.

c. At the command prompt enter: `oemctl start oms`. Wait for the message:
   ```
   "The Oracle92_homeManagementServer service was
   started successfully."
   ```

d. Close the MSDOS window.

Launch Oracle Enterprise Manager using Oracle Management Server.

- Start the OEM Console and select the Login to the Oracle Management Server option. Log in as follows:
  Administrator: `sysman` **Note:** Case is important.
  Password: `oem_temp` **Note:** Case is important.
  When prompted, change the password to `oracle`. **Note:** Case is important.
  Management Server: (Instructor supplied)
- Navigate to Navigator > Discover Nodes from the main menu after the OEM. Console is opened. The Discovery Wizard dialog box will appear.
- Select Next.
- Enter the name of the node you want to manage: that is, designated database server host name. (Instructor supplied)
- Click Next.
- Select Finish after discovery is complete.
- Click OK. **Note:** Alert the instructor if discovery is not successful.
- Expand the Database folder.
- Double click your designated database provided by your instructor.
- Enter the connection information:
  User: `sys`
  Password: `secure`
  As: `SYSDBA`

Practice 2: Solutions (continued)

4 Launch Oracle Enterprise Manager using Oracle Management Service. (continued)

- Set the node credentials for running jobs.
- Navigate to Configuration > Preferences from the main menu
- Select the Preferred Credentials page.
- Scroll down and select the entry for your designated database.
- Supply the following:

  Username: (Instructor supplied)

  Password: (Instructor supplied)

  Confirm Password.

  Role: SYSDBA
- Click OK.

5 Start SQL*Plus Worksheet

SQL*Plus Worksheet can be started within the Oracle Enterprise Manager Console using the following navigation:

- Navigate to Tools > Database Applications > SQL*Plus Worksheet

SQL*Plus Worksheet can also be started from the Windows NT menu by doing the following:

- Navigate to Start > Programs > Oracle-OraHome92 > Application Development > SQLPlus Worksheet
  - Connect directly to the database defined by the instructor, enter
    - Username
    - Password
    - Service
  - Connect as: SYSDBA
  - Click OK.

Note: Each time you log in as a different user (within SQL*Plus Worksheet) the service name must be included in the connection string. For example:

```
CONNECT SYS/ORACLE@[service name] AS SYSDBA.
```

Practice 3: Solutions

1 Connect to the database as user SYS and shut down the database.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Note: Although the OEM Console can be used to start and shutdown the database, for the purposes of this course, you will use SQL*Plus to start and shutdown the database. The following instructions have been provided for your knowledge only.

Using OEM Console

- Navigate to Instance > Configuration.
- Select the General page.
- Under Instance State, select the Shutdown option.
- Click Apply.
- Select Immediate.
- Click OK.
- Click Close.

2 With the database shut down, create an SPFILE from the PFILE. The SPFILE will be created in $ORACLE_HOME/dbs.

Using OEM Console

- Navigate to Instance > Configuration
- Select Object > Create spfile from the main menu bar.
- The Create an spfile from a pfile dialog box will appear.
- Enter pfile name to be used to create spfile.
- Click OK.

Practice 3: Solutions (continued)

3 From the operating system, view the SPFILE.

Using OEM Console

- Navigate to Instance > Configuration.
- Select All Initialization Parameters .
- Select the SPFile radio button.
- The Create an spfile from a pfile dialog box will appear.
- Enter pfile name to be used to create spfile.
- Click OK.

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

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> STARTUP
ORACLE instance started.

Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
```

Note: Although the OEM Console can be used to start and shutdown the database, for the purposes of this course, you will use SQL*Plus to start and shutdown the database. The following instructions have been provided for your knowledge only.

Using OEM Console

- Navigate to Instance > Configuration.
- Select the General page.
- Under Instance State, select the Open option.
- Click Apply.

Practice 3: Solutions (continued)

5 a Shut down the database and open it in read-only mode.

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT
Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
SQL> ALTER DATABASE OPEN READ ONLY;
Database altered.
```

Note: Although the OEM Console can be used to start and shutdown the database, for the purposes of this course, you will use SQL*Plus to start and shutdown the database. The following instructions have been provided for your knowledge only.

Using OEM Console

- Navigate to Instance > Configuration.
- Select the General page.
- Under Instance State, select the Shutdown option.
- Click Apply.
- Select Immediate from the Shutdown Options dialog box.
- Click OK.
- Click Close when processing complete.
- Under Instance State, select Show All States option.
- Select Open.
- Click Apply.
- Select Read Only Mode from the Startup Options dialog box.
- Click OK. The Starting Up Database dialog will appear.
- Select Close when processing complete.

5 b Connect as user HR password HR and insert a row into the REGIONS table as follows:

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

What happens?

```
SQL> CONNECT HR/HR
Connected.
SQL> INSERT INTO regions VALUES (5, 'Mars');
INSERT INTO regions VALUES (5, 'Mars')
            *
ERROR at line 1:
ORA-01552: cannot use system rollback segment for non-system
tablespace
'SAMPLE'
```

Practice 3: Solutions (continued)

5 c Put the database back in read-write mode.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
Database opened.
```

Note: Although the OEM Console can be used to start and shutdown the database, for the purposes of this course, you will use SQL*Plus to start and shutdown the database. The following instructions have been provided for your knowledge only.

Using OEM Console

- Navigate to Instance > Configuration.
- Select the General page.
- Under Instance State, select the Shutdown option.
- Click Apply.
- Select Immediate from the Shutdown Options dialog box.
- Click OK.
- Click Close when processing complete.
- Under Instance State, select Show All States option.
- Select Open.
- Click Apply.
- Click OK. The Starting Up Database dialog will appear.
- Select Close.

Practice 3: Solutions (continued)

6 **Note:** For the purposes of this exercise, use SQL*Plus sessions to accomplish the tasks.

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

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

```
HR SESSION
SQL> CONNECT HR/HR
Connected.
SQL> INSERT INTO regions VALUES (5, 'Mars');
1 row created.
```

b In a new telnet session start SQL*Plus. Connect user SYS and perform a SHUTDOWN TRANSACTIONAL.

```
SYS SESSION
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN TRANSACTIONAL
```

What happens? The SHUTDOWN TRANSACTIONAL is waiting on the HR session transaction to complete.

6 c Roll back the insert in the HR session and exit.

HR SESSION

```
SQL> ROLLBACK;
Rollback complete.
SQL> EXIT;
ERROR:
ORA-01089: immediate shutdown in progress - no operations are
permitted
JServer Release 9.0.0.0.0 - Beta (with complications)
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0 -
64bit Productn
With the Partitioning, Oracle Label Security, OLAP and Oracle
Data Mining optios
JServer Release 9.2.0.1.0 - Production
```

What happens to the HR session? An ORA-01089 message is received. The HR user cannot EXIT because the SYS user has issued a SHUTDOWN TRANSACTIONAL. No other operations are allowed.

What happens to the SYS session? Once the HR session completes a ROLLBACK, the SYS session will shut down the database.

SYS SESSION

```
Database closed.
Database dismounted.
ORACLE instance shut down.
```

The HR session will be disconnected based on its EXIT command.

HR SESSION

```
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0 -
64bit Productn
With the Partitioning, Oracle Label Security, OLAP and Oracle
Data Mining optios
JServer Release 9.2.0.1.0 - Production
```

Practice 3: Solutions (continued)

7 **Note:** For the purposes of this exercise, use SQL*Plus sessions to accomplish the tasks.

a In the user SYS session start the database.

```
SYS SESSION
SQL> STARTUP
ORACLE instance started.
Total System Global Area   26706720 bytes
Fixed Size                   729888 bytes
Variable Size              20971520 bytes
Database Buffers            4194304 bytes
Redo Buffers                 811008 bytes
Database mounted.
Database opened.
```

b In the open telnet session start SQL*Plus and connect as user HR.

Note: Keep the two SQL*Plus sessions open, one session as user SYS and one as user HR.

```
HR SESSION
$ sqlplus /nolog
SQL*Plus: Release 9.2.0.1.0 - Production on Wed Jul 24
16:52:51 2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights
reserved.
SQL> CONNECT HR/HR
Connected.
```

c As user SYS enable restricted session.

```
SYS SESSION
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
System altered.
```

Practice 3: Solutions (continued)

7 d As user HR, SELECT from the REGIONS table. Is the SELECT successful?

Yes.

```
HR SESSION
SQL> SELECT * FROM regions;
REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
```

e Exit the session, then reconnect as HR. What happens? The user HR does not have RESTRICTED SESSION privilege, and therefore, cannot log in.

```
HR SESSION
SQL> EXIT
Disconnected from Oracle9i Enterprise Edition Release
9.2.0.1.0 - 64bit Productn
With the Partitioning, Oracle Label Security, OLAP and Oracle
Data Mining optios
JServer Release 9.2.0.1.0 - Production
$ sqlplus /nolog
SQL*Plus: Release 9.2.0.1.0 - Production on Wed Jul 24
17:00:35 2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights
reserved.
SQL> CONNECT HR/HR
ERROR:
ORA-01035: ORACLE only available to users with RESTRICTED
SESSION privilege
Warning: You are no longer connected to ORACLE.
```

Practice 3: Solutions (continued)

7 f As user SYS disable restricted session.

```
SYS SESSION
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
System altered.
```

g Exit HR telnet session.

```
HR SESSION
$ EXIT
This session is no longer connected.
```

Practice 5: Solutions

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 tables.
   d. The data dictionary records and verifies information about its associated database.

   Answer: All of the Above

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

   Answer: False, The base tables are created with the `sql.bsq` script

3. Which three 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 reference.
   d. The data dictionary is a necessary ingredient for the database to function.

   Answer: B,C,D

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

   Answer: True

5. The information for a dynamic performance view is gathered from the control file.
   a. True
   b. False

   Answer: True

6. Which of the following questions might a dynamic performance 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?

   Answer: A,B,E

Practice 5: Solutions (continued)

7 Connect as SYSTEM/MANGER and find a list of the data dictionary views.

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> SELECT table_name FROM dictionary;

TABLE_NAME
------------------------------
ALL_ALL_TABLES
ALL_ARGUMENTS
ALL_ASSOCIATIONS
ALL_AUDIT_POLICIES
ALL_BASE_TABLE_MVIEWS
ALL_CATALOG
ALL_CLUSTERS
ALL_CLUSTER_HASH_EXPRESSIONS
ALL_COLL_TYPES
.
.
.
1256 rows selected.
```

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

Hint: Query the dynamic performance views V$DATABASE, V$THREAD, and V$PARAMETER.

Using OEM Console

- Navigate to Instance > Configuration.
- Select the General page.
- Click the All Initialization Parameters button.
- View: db_name, instance_name, db_block_size parameters.
- Select Cancel to exit.

Practice 5: Solutions (continued)

9 List the name of the data files.

Hint: Query the dynamic performance view `V$DATAFILE`.

Using OEM Console

- Navigate to Storage > Datafiles.
- Data files will be displayed in the right window pane.

.

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

Hint: Query the data dictionary view `DBA_DATA_FILES` to identify the `SYSTEM` tablespace data file.

Using OEM Console

- Navigate to Storage > Tablespaces > SYSTEM > Datafiles.
- The SYSTEM data file will be displayed in the right window pane.

.

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.

```
SQL> SELECT sum(bytes)/1024 "free space in KB"
  2  FROM dba_free_space;
free space in KB
----------------
            22716
SQL> SELECT sum(bytes)/1024 "used space in KB"
  2  FROM dba_segments;
used space in KB
----------------
           212448
```

Practice 5: Solutions (continued)

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.

Using OEM Console

- Navigate to Security > Users.
- Users will be displayed in the right window pane.

.

Practice 6: Solutions

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

Hint: Query the dynamic performance view `V$CONTROLFILE`.

Note: You can also use `V$PARAMETER`, or execute the `SHOW PARAMETER` command to display the name and the location of the control file.

Using OEM Console

- Navigate to Storage > Controlfile.
- Control files will be displayed in the right window pane.

2 **Note:** For the purposes of this exercise, use SQL*Plus sessions to accomplish the tasks.

a 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?

Hints:

- Connect as user `SYS`.
- Shutdown the database using the `IMMEDIATE` option.
- Using OS command line, copy the control file `.ctl` as a `.bak` extension.
- Remove the control file `.ctl`.
- Start the database.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.ctl
        $HOME/ORADATA/u01/ctrl01.bak
SQL> !rm $HOME/ORADATA/u01/ctrl01.ctl
SQL> STARTUP
ORACLE instance started.
Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
ORA-00205: error in identifying controlfile, check alert log
for more info
```

Practice 6: Solutions (continued)

2 b To resolve the ORA-00205 error:

- Shutdown the database
- Rename the copied control file to the appropriate name
- Startup the database.

```
SQL> SHUTDOWN IMMEDIATE
ORA-01507: database not mounted
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.bak
       $HOME/ORADATA/u01/ctrl01.ctl
SQL> STARTUP
ORACLE instance started.
Total System Global Area  131040632 bytes
Fixed Size                   730488 bytes
Variable Size             113246208 bytes
Database Buffers           16777216 bytes
Redo Buffers                 286720 bytes
Database mounted.
Database opened.
```

Practice 6: Solutions (continued)

3 **Note:** For the purposes of this exercise, use SQL*Plus sessions to accomplish the tasks.

a 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`.

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. **Note:** Normally the permissions on the file would not be changed, this is for the classroom environment.
- Start up the database.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET control_files =
       '$HOME/ORADATA/u01/ctrl01.ctl',
       '$HOME/ORADATA/u02/ctrl02.ctl' SCOPE=SPFILE;
System altered.
SQL> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.ctl
        $HOME/ORADATA/u02/ctrl02.ctl
SQL> !chmod 660 $HOME/ORADATA/u02/ctrl02.ctl
SQL> STARTUP
ORACLE instance started.
Total System Global Area  131040632 bytes
Fixed Size                   730488 bytes
Variable Size             113246208 bytes
Database Buffers           16777216 bytes
Redo Buffers                 286720 bytes
Database mounted.
Database opened.
```

Practice 6: Solutions (continued)

3 b Confirm that the control file was multiplexed and is now being used.

Hint: Query the dynamic performance views `V$CONTROLFILE` or `V$PARAMETER`, or use the `SHOW PARAMETER` command to confirm that both control files are being used.

Using OEM Console
- Navigate to Storage > Controlfile.
- Control files will be displayed in the right window pane.

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

Hint: Query the dynamic performance view `V$CONTROLFILE_RECORD_SECTION`.

Using OEM Console
- Navigate to Storage > Controlfile.
- Select the Records Section page.

Practice 7: Solutions

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

Hint: Query the dynamic performance view `V$LOGFILE`.

Using OEM Console
- Navigate to Storage > Redo Log Groups.
- Expand the Redo Log Groups.
- Select A Redo Log Group.
- Online redo log file information will be displayed in the right window pane.

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

Hint: Query the dynamic performance view `V$LOGFILE`.

Using OEM Console
- Navigate to `Storage > Redo Log Groups.`
- Online redo log file information will be displayed in the right window pane.

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

Hints:

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

Using OEM Console
- Navigate to Instance > Configuration.
- Under the Database and Instance Information section, you will see the Archive Log Mode.

Practice 7: Solutions (continued)

3 Add an online redo log file 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 an online redo log file member to each group.
- Query the dynamic performance view `V$LOGFILE` to verify the result.

Using OEM Console
- Navigate to Storage > Redo Log Groups.
- Expand the redo log group.
- In the Redo Log Members section, enter File Name and File Directory.
- Click Apply.

4 Add an online redo log file group with two members located on `u03` and `u04` using the following naming conventions and verify the result.

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

Hints:

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

Using OEM Console
- Navigate to Storage > Redo Log Groups.
- Select Create from right mouse menu.
- Enter the name and location of the online redo log file members to be created.
- Click Create.

Practice 7: Solutions (continued)

5 Remove the online redo log file group created in step 4.

Hints:

- Use `ALTER SYSTEM SWITCH LOGFILE` if the log files are active. The number of log switches required will vary. **Note:** Query the database to see which log file is active then decide how many times you need to perform the `ALTER SYSTEM SWITCH LOGFILE` command.
- Execute the `ALTER DATABASE DROP LOGFILE GROUP` command to remove the log group.
- Query the dynamic performance view `V$LOG` to verify the result.
- Remove the operating system files for the group.

Using OEM Console

- Navigate to Storage > Redo Log Groups
- Select Group 3 if active, and select Switch Logfile from right mouse menu.
- Select Group 3, and choose Remove from right mouse menu.
- Remove the OS files for the group. See below.

Using OS Command Line

```
$ rm $HOME/ORADATA/u03/log03a.rdo
$ rm $HOME/ORADATA/u04/log03b.rdo
```

Practice 7: Solutions (continued)

6 Resize all online redo log files to 1024 KB.

Hints:

- We cannot resize log files, therefore, add new logs and drop the old.
- Execute the `ALTER DATABASE ADD LOGFILE GROUP` command to add two new groups with the size 1024 KB.
- Query the dynamic performance view `V$LOG` to check the active group.
- Execute the `ALTER SYSTEM SWITCH LOGFILE` command to force log switches and change the group stage to inactive. The number of log switches required will vary. **Note:** Query the database to see which log file is active then decide how many times you need to perform the `ALTER SYSTEM SWITCH LOGFILE` command.
- Execute the `ALTER DATABASE DROP LOGFILE` command to remove the unused groups.
- Query the dynamic performance view `V$LOG` to verify the result.

Using OEM Console

- Navigate to Storage > Redo Log Groups.
- Use Create from right mouse menu to add the two log groups.
- Use Switch Logfile form right mouse menu until Group 1 and Group 2 become inactive.
- Use Remove from the right mouse menu to drop Group1 and Group 2.
- Remove the OS files for the group.

Practice 8: Solutions

1 Create permanent tablespaces with the following names and storage:

a Tablespace Name: `DATA01`

Datafile Name: `data01.dbf`

Size: `2M`

Extent Management: `dictionary`

Location: `u04`

Using OEM Console

- Navigate to Storage > Tablespaces.
- Select Create from right mouse menu.
- On the General page, enter Name, File Name, File Directory, and Size.
- Select Status: Online, and Type: Permanent.
- On the Storage page, select Extent Management: Managed in the Dictionary.
- Select Create.
- Click OK.

b Tablespace Name: `DATA02`

Datafile Name: `data02.dbf`

Size: `1M`

Extent Management: `local uniform size 100K`

Location: `u03`

Using OEM Console

- Navigate to Storage > Tablespaces.
- Select Create from right mouse menu.
- On the General page, enter Name, File Name, File Directory, and Size.
- Select Status: Online, and Type: Permanent.
- On the Storage page, select Extent Management: Locally Managed.
- Select Uniform Allocation, and enter size.
- Select Create.
- Click OK.

Practice 8: Solutions

1 c Tablespace Name: `INDEX01`

Datafile Name: `index01.dbf`

Size: `1M`

Extent Management: `local uniform size 4K`

Location: `u02`

Enable automatic extension of 500 KB when more extents are required with a maximum size of 2 MB.

Using OEM Console

- Navigate to Storage > Tablespaces.
- Select Create from right mouse menu.
- On the General page, enter Name, File Name, File Directory, and Size.
- Select Status: Online, and Type: Permanent.
- On the Storage page, select Extent Management: Locally Managed
- Select Uniform Allocation, and enter size.
- Select Create.
- Click OK.
- Select Edit Datafile.
- On the Storage page, select Automatic extend data filess when full (AUTOEXTEND).
- Enter increment size and maximum value size.
- Click OK.

d Tablespace Name: `RONLY`

Datafile Name: `ronly01.dbf`

Size: `1M`

Location: `u01`

Default storage. DO NOT make the tablespace read only at this time.

Using OEM Console

- Navigate to Storage > Tablespaces.
- Select Create from right mouse menu.
- On the General page, enter Name, File Name, File Directory, and Size.
- Select Status: Online, and Type: Permanent.
- On the Storage page, select Extent Management: Locally Managed.
- Select Uniform Allocation, and enter size.
- Select Create.
- Click OK.

e Display the information from the data dictionary.

Hint: Information about tablespaces can be viewed using any of the following queries.

- `DBA_TABLESPACES`
- `V$TABLESPACE`
- `V$DATAFILE`

Using OEM Console

- Navigate to Storage > Tablespaces.
- Tablespaces will be displayed in the right window pane.

2 Allocate 500K more disk space to tablespace `DATA02.` Verify the result.

Using OEM Console

- Navigate to Storage > Tablespaces > DATA02.
- In the General tab, modify the file size.
- Click Apply.

Practice 8: Solutions (continued)

3 Relocate tablespace `INDEX01` to subdirectory `u06`. Verify relocation and status of `INDEX01`.

Hints:

- Take the `INDEX01` tablespace offline.
- Use `V$DATAFILE` to verify status.
- Use operating system move command to move the tablespace to `u06`.
- Use `ALTER TABLESPACE` to relocate the tablespace.
- Place the `INDEX01` tablespace online.
- Use `V$DATAFILE` to verify status.

Using OEM Console

- Navigate Storage > Tablespaces > INDEX01.
- Select Offline and Normal from the General tab Satus section.
- Click Apply.
- Move the physical data file using OS command line. See below.
- Return back to the General tab of INDEX01 and update the File Directory to u06.
- Click Apply.
- Select Online button from the General tab.
- Click Apply.

Note: The name change can also be accomplished by:

- Navigate Storage > Datafiles and selecting the file belonging to INDEX01 tablespace.

Using OS Command Line

```
SQL> !mv $HOME/ORADATA/u02/index01.dbf
        $HOME/ORADATA/u06/index01.dbf
```

Practice 8: Solutions (continued)

4 a Connect as user `SYSTEM` and create a table in tablespace `RONLY`. Make tablespace read-only. Run a query to verify it.

Using OEM Console

- Navigate to Schema > SYSTEM.
- Select Create from the right mouse menu.
- Select Table.
- Select &Create. The Table Wizard will appear to guide you through creating a table.
- Enter Table Name: table1, Schema: SYSTEM, Tablespace: RONLY.
- Select Next.
- Enter Column Definition. Create one column named X, datatype CHAR, and size 1.
- Select Next.
- Select Yes to confirm one column only.
- Select Next on the remaining pages.
- Select Finish to create the table.
- Click OK.
- Make the tablespace read only using Storage > Tablespace > RONLY.
- Select Make Read-Only from the right mouse menu.
- Select Yes in the dialog box.
- Verify from the General page that the tablespace is read-only. **Note:** If the Read Only box is not checked, try refreshing the screen.

4 b Attempt to create an additional table called `TABLE2`. Drop the first created table, `TABLE1`. What happens?

Using OEM Console

- Create second table using `Schema > Table.`
- Select `Create` from the right mouse menu.
- In the `General` page fill in the appropriate information to create the table.
- Click `Create.`
- Select the table in the `Table` directory and use `Remove` from the right mouse menu to drop the table.

Using OEM Console

- Navigate to Schema > SYSTEM.
- Select Create from the right mouse menu.
- Select Table.
- Select &Create. The Table Wizard will appear to guide you through creating a table.
- Enter Table Name: table2, Schema: SYSTEM, Tablespace: RONLY.
- Select Next.
- Enter Column Definition. Create one column named X, datatype CHAR, and size 1.
- Select Next.
- Select Yes to confirm one column only.
- Select Next on the remaining pages.
- Select Finish. An ORA-01647 error message will be received.
- Select Cancel.

To remove TABLE1.

- Navigate to Schema > SYSTEM > Tables.
- Highlight TABLE1.
- Select Remove from right mouse menu.
- Click Yes, to confirm Remove.

Practice 8: Solutions (continued)

5 Drop tablespace `RONLY` and the associated data file. Verify it.

Hints:

- Use the `INCLUDING CONTENTS AND DATAFILES` clause
- View the `V$TABLESPACE` to verify the tablespace was dropped.
- Use the `! f`rom the SQL prompt to see a list of the data filess in `u01`.

Using OEM Console

- Storage > Tablespaces.
- Highlight RONLY tablespace.
- Select Remove from right mouse menu. The Drop Tablespace dialog box will appear.
- Select Delete associated data filess from the OS.
- Yes to confirm drop.
- Verify it has been dropped by selecting Tablespaces from the tree.

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.

Using OEM Console

- Navigate Instance > Configuration.
- Select ALL INITIALIZATION PARAMETERS.
- Verify the Running radio button is selected.
- Move down to `DB_CREATE_FILE_DEST` and modify the location to u05. Example: `$HOME/ORADATA/u05`.
- Click Apply. A dialog box will appear indicating parameter has been changed.
- Click OK.

To create a tablespace: DATA03

- Navigate to Storage > Tablespaces.
- Select Create from right mouse menu.
- On the General page, enter Name, File Name, File Directory, and Size.
- Select all defaults.
- Select Create.
- Click OK.

Practice 9: Solutions

1 As user SYSTEM, run the `lab09_01.sql` script to create tables and indexes.

```
SQL> @$HOME/STUDENT/LABS/lab09_01.sql
```

2 Identify the different types of segments in the database.

```
SQL> SELECT DISTINCT segment_type FROM dba_segments;
SEGMENT_TYPE
------------------
CACHE
CLUSTER
INDEX
INDEX PARTITION
LOBINDEX
LOBSEGMENT
NESTED TABLE
ROLLBACK
TABLE
TABLE PARTITION
TYPE2 UNDO
11 rows selected.
```

Practice 9: Solutions (continued)

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.

Hints:
- Select from DBA_EXTENTS.
- Use the segment_name, segment_type, max_extents, extents keywords.

```
SQL> COLUMN segment_name FORMAT a20
SQL> COLUMN segment_type FORMAT a15
SQL> SELECT segment_name,segment_type,
  2         max_extents, extents
  3  FROM   dba_segments
  4  WHERE  extents+5 > max_extents
  5  AND    segment_type<>'CACHE';

SEGMENT_NAME         SEGMENT_TYPE    MAX_EXTENTS    EXTENTS
-------------------- --------------- ----------- ----------
EMP                  TABLE                    10          8
```

4 Which files have space allocated for the EMP table?

```
SQL> SELECT DISTINCT f.file_name
  2  FROM   dba_extents e,dba_data_files f
  3  WHERE  e.segment_name='EMP'
  4  AND    e.file_id=f.file_id;

FILE_NAME
-----------------------------------
/home1/user481/ORADATA/u04/data01.dbf
```

Practice 9: Solutions (continued)

5 Run the `lab09_05.sql` script.

```
SQL> @$HOME/STUDENT/LABS/lab09_05.sql
```

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.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT tablespace_name,COUNT(*) AS fragments,
  2    SUM(bytes) AS total,
  3    MAX(bytes) AS largest
  4  FROM dba_free_space
  5  GROUP BY tablespace_name;
TABLESPACE_NAME FRAGMENTS  TOTAL           LARGEST
--------------- ---------- ------          ---------
 DATA01                  3  147456         126976
 DATA02                  1  1433600        1433600
 DATA03                  1  5177344        5177344
 EXAMPLE                 1  5242880    5242880
 INDEX01                1 925696           925696
 INDX                    1    5120000    5120000
 QUERY_DATA              1     884736     884736
 SAMPLE                  1    9437184    9437184
 SYSTEM                  1   15224832   15224832
 UNDOTBS                 1   46071808   46071808
 USERS                   1    5177344    5177344

11 rows selected.
```

Practice 9: Solutions (continued)

7 List segments that will generate errors because of lack of space when they try to allocate an additional extent.

```
SQL> SELECT s.segment_name, s.segment_type, s.tablespace_name,
  2    s.next_extent
  3  FROM   dba_segments s
  4  WHERE  NOT EXISTS
  5    ( SELECT 1
  6      FROM dba_free_space f
  7      WHERE s.tablespace_name=f.tablespace_name
  8      HAVING  max(f.bytes) > s.next_extent ) ;

SEGMENT_NAME SEGMENT_TYPE TABLESPACE_NAME NEXT_EXTENT
------------ ------------ --------------- -----------
BIG_EMP      TABLE        DATA01              1048576
```

Practice 10: Solutions

1 Connect as user SYSTEM/MANAGER, and list the undo segments in tablespace UNDOTBS.

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
_SYSSMU9$
_SYSSMU10$

10 rows selected.
```

2 Create undo tablespace UNDO2, size 15M, in $HOME/ORADATA/u03. List the undo segments in tablespace UNDO2.

Using OEM Console

- Navigate to Storage > Tablespaces.
- Select Create from right mouse menu.
- On the General page, enter Name, File Name, File Directory, and Size.
- Select Status: Online, and Type: Undo.
- On the Storage page, select Extent Management: Locally Managed.
- Select Create.
- Click OK.
- Highlight Tablespaces on the tree to view.

Practice 10: Solutions (continued)

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.

```
SQL> @$HOME/STUDENT/LABS/lab10_03.sql
SQL> CONNECT hr/hr
Connected.
SQL> INSERT INTO departments
  2    (department_id,department_name)
  3  VALUES (9999,'x');
1 row created.
```

4 In the session in which you are connected as SYS, using the `ALTER SYSTEM` command, switch the UNDO tablespace from UNDOTBS to UNDO2, using `SCOPE=BOTH`.

Using OEM Console

- Navigate to Instance > Configuration.
- Select the UNDO page.
- Select UNDO2 from the Current Undo Tablespace drop down menu.
- Click Apply.
- Click OK.

5 As SYS drop tablespace `UNDOTBS`. Use the `INCLUDING CONTENTS AND DATAFILES` clause. What happened?

Using OEM Console

- Navigate to Storage > Tablespaces > UNDOTBS.
- Select Remove from right mouse menu. A Drop Tablespace dialog box will appear.
- Select Delete associated from from the OS.
- Select Yes. ORA-30013 error message will appear.
- Click OK.

Practice 10: Solutions (continued)

6 List the undo segments in tablespace UNDOTBS and their status. Compare this list to the list in step 1.

Hint: Query data dictionary view DBA_ROLLBACK_SEGS.

```
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
_SYSSMU9$
_SYSSMU10$
10 rows selected.
SQL> SELECT a.usn,a.name,b.status
 2  FROM v$rollname a, v$rollstat b
 3  WHERE a.name
 4   IN ( SELECT segment_name
 5        FROM   dba_segments
 6        WHERE  tablespace_name = 'UNDOTBS' )
 7  AND a.usn = b.usn;

  USN NAME         STATUS
----- ------------ ---------------
 10 _SYSSMU10$   PENDING OFFLINE
```

Practice 10: Solutions (continued)

7 In the session connected as HR, roll back the transaction and exit the session.

```
SQL> ROLLBACK;
Rollback complete.
SQL> EXIT;
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0
- 64bit Productn
With the Partitioning, Oracle Label Security, OLAP and Oracle
Data Mining optios
JServer Release 9.2.0.1.0 - Production
```

8 In the session connected as SYS drop tablespace UNDOTBS. What happened?

```
SQL> DROP TABLESPACE undotbs;
DROP TABLESPACE undotbs
*
ERROR at line 1:
ORA-30013: undo tablespace 'UNDOTBS' is currently in use
```

9 As SYS issue the following command:

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

Now drop tablespace UNDOTBS. What happened?

Note: There still may be a delay before the tablespace is drop.

Using OEM Console

- Navigate to Instance > Configuration.
- Select ALL INITIALIZATION PARAMETERS.
- Select Running.
- Move to the undo_retention parameter and modify value to 0.
- Click Apply.
- Click OK.

Practice 11: Solutions

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. **Note:** When using OEM be sure to set DATE_OF_DELY to NULL.
In addition, 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.

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

Using OEM Console

- Navigate to Schema > SYSTEM > Tables.
- Select Create from the right mouse menu.
- In the General page fill in the appropriate information to create the CUSTOMERS table.
- Click Create.
- Click OK.
- Repeat steps to create the ORDERS table.

2 Run the script lab11_02.sql to insert rows into the tables.

```
SQL> @$HOME/STUDENT/LABS/lab11_02.sql
```

Practice 11: Solutions (continued)

3 Find which files and blocks contain the rows for the ORDERS table.

Hint: Query data dictionary view `DBA_EXTENTS`.

Using OEM Console

- Navigate to Tablespaces > USERS.
- Select Show Tablespace map from the right mouse menu.
- Select the ORDERS table.
- Drag your mouse across the colored bar to ORDERS table on the Tablespace Map tab.
- Select File > Close.

4 Check the number of extents used by the `ORDERS` table .

Using OEM Console

- Navigate to Tablespaces > USERS.
- Select Show Tablespace map from the right mouse menu.
- Select the ORDERS table.
- View the bottom of the Tablespace Map tab for extent usage information.
- Select File > Close.

5 Allocate an extent manually, with default size, for the table `ORDERS` and confirm that the extent has been added as specified.

```
SQL> ALTER TABLE orders ALLOCATE EXTENT;
Table altered.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         2
```

Practice 11: Solutions (continued)

6 Create another table, `ORDERS2` as copy of the `ORDERS` table in the `USERS` tablespace, with `MINEXTENTS` equal to `10`. Verify that the table has been created with the specified number of extents.

Using OEM Console

- Navigate to Schema > SYSTEM > Tables > ORDERS.
- Select Create Like from the right mouse menu. A Create Table dialog box will appear.
- Enter Name in General page.
- Enter MINEXTENTS as 10 under the Extents section of the Storage page.
- Click Create.
- Click OK.
- View Tables folder to verify.

Practice 11: Solutions (continued)

7 Truncate table ORDERS without releasing space and check the number of extents to verify extents have not been deallocated.

```
SQL> TRUNCATE TABLE orders REUSE STORAGE;
Table truncated.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         2
```

8 Truncate the ORDERS2 table, releasing space. How many extents does the table have now?

```
SQL> TRUNCATE TABLE orders2;
Table truncated.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS2'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
        10
```

Using OEM Console

- Navigate to Tablespaces > USERS.
- Select Show Tablespace map from the right mouse menu.
- Select the ORDERS2 table.
- View the the Tablespace Map for extent usage information.
- Select File > Close.

9 Run the script `lab11_09.sql` to insert some rows into the ORDERS2 table.

```
SQL> @$HOME/STUDENT/LABS/lab11_09.sql
```

Practice 11: Solutions (continued)

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?

Using OEM Console

- Navigate to Schema > SYSTEM > Tables.
- Double click on the ORDERS2 table in the right column. An Edit Table dialog box will appear.
- Select the DATE_OF_DELY column on the General page.
  Click the Drop Column icon (located at the bottom left of the General page).
- Highlight the DATE_OF_DELY column and use the right promote arrow [>].
  This will promote the column from the Available Columns to the Selected Columns.
- Select the Mark selected columns as unused radio button under the Operations section.
- Click OK.

To view the columns for the ORDERS2 table.

- Navigate to Schema > SYSTEM > Tables > ORDERS2.
- Verify the DATE_OF_DELY is no longer available.

11 Drop the unused column DATE_OF_DELY.

Using OEM Console

- Navigate to Schema > SYSTEM > Tables > ORDERS2.
- Select Drop Unused Columns from the Unused Columns section on the General page.
- Click Apply.

12 Drop the ORDERS2 table.

Using OEM Console

- Navigate to Schema > SYSTEM > Tables > ORDERS2.
- Select Remove from the right mouse menu.
- Click Yes to confirm drop.

Practice 12: Solutions

1 You are considering creating indexes on the `NAME` and `REGION` columns of the `CUSTOMERS` table. What types of index are appropriate for the two columns? Create two indexes, naming them `CUST_NAME_IDX` and `CUST_REGION_IDX`, respectively, and placing them in the `INDEX01` tablespace.

Hint: A B-tree index is suitable for a column with many distinct values, and a bitmap index is suitable for columns with only a few distinct values.
Note: `CUSTOMERS` table is in the `SYSTEM` schema.

Using OEM Console

Create the CUST_NAME_IDX index.

- Navigate to Schema > SYSTEM > Indexes.
- Select Create from the right mouse menu.
- On the General page complete the necessary information to create the CUST_NAME_IDX index.
- Highlight the column you want to create the index on (name). This will place a [1] in the Order column.
- Click Create.
- Click OK.

Repeat the steps to create the CUST_REGION_IDX index.

- Navigate to Schema > SYSTEM > Indexes.
- Select Create from the right mouse menu.
- On the General page complete the necessary information to create the CUST_REGION_IDX index.
- Highlight the column you want to create the index on (region). This will place a [1] in the Order column.
- Select Bitmap from the Options section of the General page.
- Click Create.
- Click OK.

2 Move the `CUST_REGION_IDX` index to another tablespace.

Hint: The index can be rebuilt specifying a different tablespace.

Using OEM Console

- Navigate to Schema > SYSTEM > Indexes > CUST_REGION_IDX.
- Select Reorganize from the right mouse menu. The Reorganize Wizard dialog box will appear. Note: You must be using OEM via an OMS to use this option.
- Enter the appropriate information on the Wizard pages.
- Click Finish.
- Click OK.

Practice 12: Solutions (continued)

3 Note the files and blocks used by the extents for the CUST_REGION_IDX index.

Hint: Use the view DBA_EXTENTS to get this information. **Note**: The owner is SYSTEM.

Using OEM Console

- Navigate to Storage > Tablespaces > INDEX01.
- Select Show Tablespace map from the right mouse menu.

4 Re-create the CUST_REGION_IDX index without dropping and re-creating it, and retain it in the same tablespace as before. Does the new index use the same blocks that were used earlier?

Hint: Rebuild the index.

Note: The new index does not reuse the same space as seen from the location of the extent after rebuild. This is because Oracle server builds a temporary index, drops the old one, and renames the temporary index.

```
SQL> ALTER INDEX cust_region_idx REBUILD;
Index altered.
SQL> SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  segment_name='CUST_REGION_IDX'
  4  AND    owner='SYSTEM';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         4        142        125
```

5 **a** As user SYSTEM, run the script lab12_05a.sql to create and populate the NUMBERS table.

```
SQL> @$HOME/STUDENT/LABS/lab12_05a.sql
```

5 b Query the NUMBERS table to find the number of distinct values in the two columns in the table.

```
SQL> SELECT count(DISTINCT no) NO,
  2    count(DISTINCT odd_even) OE
  3  FROM   numbers;

        NO         OE
---------- ----------
     10000          2
```

c Using uniform extent sizes of 4KB, create two indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively. Place the indexes in the INDEX01 tablespace. Check the total sizes of the indexes and write the number of blocks in box below.

Hint: Use PCTINCREASE equal to zero to create extents of equal size. Check the total blocks allocated to the extents from DBA_SEGMENTS.

Index	Column	Blocks
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

Using OEM Console

- Navigate to Schema > SYSTEM > Indexes.
- Select Create from the right mouse menu.
- On the General and Storage page complete the necessary information to create the index.
- Highlight the column you want to create the index on. This will place a [1] in the Order column.
- Click Create.
- Click OK.

Practice 12: Solutions (continued)

5 d After you have recorded the blocks above, drop the two indexes, NUMB_OE_IDX and NUMB_NO_IDX . Using uniform extent sizes of 4KB, create bitmap indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively. Place the indexes in the INDEX01 tablespace. Re-execute the query to check the total blocks allocated to the extents from DBA_SEGMENTS. Check the total sizes of the indexes and write in box below.

Index	Column	Blocks
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

Using OEM Console

To Drop the Indexes

- Navigate to Schema > SYSTEM > Indexes.
- Highlight the index on the right column, select Remove from the right mouse menu to drop the index.
- Click OK to confirm drop.
- Repeat for second index.

To Create the Bitmap Indexes

- Navigate to Schema > SYSTEM > Indexes.
- Select Create from the right mouse menu.
- On the General and Storage page complete the necessary information to create the index.
- Highlight the column you want to create the index on. This will place a [1] in the Order column.
- Select Bitmap from the Options section of the General page.
- Click Create.
- Click OK.

What can you conclude about the relationship between cardinality and sizes of the two types of indexes?

Answer: It can be seen from the results that a bitmap index is compact for a low-cardinality column, while a B-tree index is compact for a high-cardinality column.

Practice 13: Solutions

1 Examine and run the script `lab13_01.sql` to create the constraints.

```
SQL> @$HOME/STUDENT/LABS/lab13_01.sql
```

2 As user SYSTEM, query the data dictionary to:

a Check for constraints, whether they are deferrable, and their status.

Hint: Use the `DBA_CONSTRAINTS` view to get this information.

```
SQL> COLUMN constraint_name FORMAT a25
SQL> COLUMN table_name      FORMAT a10
SQL> COLUMN constraint_type FORMAT a1
SQL> COLUMN deferrable      FORMAT a15
SQL> COLUMN status          FORMAT a10
SQL> SELECT constraint_name, table_name,
  2    constraint_type, deferrable, status
  3  FROM   dba_constraints
  4  WHERE  table_name IN
  5 ('PRODUCTS','ORDERS','CUSTOMERS')
  6  AND owner='SYSTEM';

CONSTRAINT_NAME           TABLE_NAME C DEFERRABLE      STATUS
------------------------- ---------- - --------------- --------
CUSTOMERS_REGION_CK       CUSTOMERS  C NOT DEFERRABLE  ENABLED
CUSTOMERS_CUST_CODE_PK    CUSTOMERS  P DEFERRABLE      ENABLED
ORDERS_CUST_CODE_FK       ORDERS     R DEFERRABLE      ENABLED
ORDERS_DATE_OF_DELY_CK    ORDERS     C NOT DEFERRABLE  ENABLED
ORDERS_ORD_ID_PK          ORDERS     P NOT DEFERRABLE  ENABLED
PRODUCTS_PROD_CODE_UK     PRODUCTS   U DEFERRABLE     DISABLED

6 rows selected.
```

Using OEM Console

A portion of the constraint information can be viewed via OEM as follows.

- Navigate to Schema > SYSTEM > Tables.
- Select the tables CUSTOMERS, PRODUCTS, ORDERS (individually).
- Select View/Edit Details.
- Select the Contraints page.
- Click OK.

2 b Check the names and types of indexes created to validate the constraints.

Hint: The indexes are only created for primary key and unique constraints and have the same name as the constraints

Using OEM Console
- Navigate to Schema > SYSTEM > Tables.
- Select the tables CUSTOMERS, PRODUCTS, ORDERS (individually).
- Select View/Edit Details.
- Select the Constraints page.
- Click OK.

3 As user `SYSTEM`, run the script `lab13_03.sql` to insert two records into the `PRODUCTS` table.

```
SQL> @$HOME/STUDENT/LABS/lab13_03.sql
```

4 Enable the unique constraint on the `PRODUCTS` table. Was it successful?

Using OEM Console
- Navigate to Schema > SYSTEM > Tables > PRODUCTS.
- Deselect the Disable checkmark on the Constraints page. An ORA-2299 error message will be received.
- Click OK.
- Select Revert.

5 a Ensure any new rows added to the table do not violate the constraint on the PRODUCTS table.

Hint: This can be done by enabling the constraint NOVALIDATE.

Using OEM Console

- Navigate to Schema > SYSTEM > Tables > PRODUCTS.
- Select Novalidate on the Constraints page.
- Click Apply.

b Query the data dictionary to verify the effect of the change.

```
SQL> SELECT constraint_name, table_name,
  2    constraint_type, validated, status
  3  FROM   dba_constraints
  4  WHERE  table_name = 'PRODUCTS'
  5  AND    owner='SYSTEM';

CONSTRAINT_NAME           TABLE_NAME C VALIDATED     STATUS
----------------------      ---------- - ------------- ------
PRODUCTS_PROD_CODE_UK     PRODUCTS   U NOT VALIDATED ENABLED
```

5 c Test that the constraint disables inserts that violate the change by adding a row with the following values:

PRODUCT_ID	PRODUCT_DESCRIPTION	LIST_PRICE
4000	Monitor	3000

```
SQL> INSERT INTO system.products
  2  VALUES(4000,'Monitor',3000);
INSERT INTO system.products
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.PRODUCTS_PROD_CODE_UK)
violated
```

6 Take the necessary steps to identify existing constraint violations in the PRODUCTS table, modify product codes as needed, and guarantee that all existing as well as new data do not violate the constraint. (Assume that the table has several thousands of rows and it is too time-consuming to verify each row manually.)

Hint: Use the following steps:

a Create the EXCEPTIONS table.

```
SQL> CONNECT system/manager
Connected.
SQL> @?/rdbms/admin/utlexcpt
```

b Run the command to enable the constraint and trap the exceptions.

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk
  3  EXCEPTIONS INTO system.exceptions;
ALTER TABLE system.products
*
ERROR at line 1:
ORA-02299: cannot validate (SYSTEM.PRODUCTS_PROD_CODE_UK) -
duplicate keys found
```

6 c Use the ROWID in the EXCEPTIONS table to list the rows in the PRODUCTS table that violate the constraint. Do not list LOB columns.

```
SQL> SELECT rowid, prod_code, description
  2  FROM   system.products
  3  WHERE  rowid IN
   4    ( SELECT row_id
   5      FROM   exceptions
  6      WHERE  table_name='PRODUCTS' ) ;
ROWID              PROD_CODE DESCRIPTION
------------------ ---------- ------------------------------
AAAF/MAAJAAAAASAAA       4000 UNIX Monitor
AAAF/MAAJAAAAASAAB       4000 NT Monitor
```

d Rectify the errors.

```
SQL> UPDATE   system.products
  2  SET      prod_code='4001'
  3  WHERE    rowid = ( SELECT max(row_id)
  4                     FROM   exceptions
  5                     WHERE  table_name='PRODUCTS' ) ;
1 row updated.
```

e Enable the constraint.

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk
  3  EXCEPTIONS INTO system.exceptions;
Table altered.
```

7 Run the script `lab13_07.sql` to insert rows into the table. Were the inserts successful?

```
SQL> @$HOME/STUDENT/LABS/lab13_07.sql
```

8 Now examine the script `lab13_08`. Notice that this script also performs the inserts in the same sequence. Run the script and check if it executes successfully.

```
SQL> @$HOME/STUDENT/LABS/lab13_08.sql
```

9 Truncate the `CUSTOMERS` table. Was it successful?

```
SQL> TRUNCATE TABLE system.customers;
TRUNCATE TABLE system.customers
                      *
ERROR at line 1:
ORA-02266: unique/primary keys in table referenced by enabled
foreign keys
```

Practice 14: Solutions

1 a Run the `lab14_01.sql` script to create user `Jeff`.

```
SQL> @$HOME/STUDENT/LABS/lab14_01.sql
```

b Connect as user SYS and enable password management by running script `$ORACLE_HOME/rdbms/admin/utlpwdmg.sql`.

```
SQL> connect / as sysdba
SQL> @?/rdbms/admin/utlpwdmg dmg
```

2 Try to change the password for user `Jeff` to `Jeff`. What happens?

Using OEM Console

- Navigate to Security > Users > Jeff.
- Enter Password and Confirm Password in the General page.
- Click Apply. An error will be displayed.

3 Try changing the password for `Jeff` to follow the password management format.

Hint: Password should contain at least one digit, one character, and one punctuation.

Using OEM Console

- Navigate to Security > Users > Jeff.
- Enter Password and Confirm Password in the General page.
- Click Apply.

Practice 14: Solutions (continued)

4 Alter the `DEFAULT` profile to ensure the following applies to users assigned the `DEFAULT` profile:

- After two login attempts, the account should be locked.
- The password should expire after 30 days.
- The same password should not be reused for at least one minute.
- The account should have a grace period of five days to change an expired password.
- Ensure that the requirements given have been implemented.

Hints:

- Use the `ALTER PROFILE` command to change the default profile limits.
- Query the data dictionary view `DBA_PROFILES` to verify the result.

Using OEM Console

- Navigate to Security > Profiles > DEFAULT.
- Select Password page.
- Enter the following for Expire Password section:
  - Expire in: 30 (PASSWORD_LIFE_TIME)
  - Lock: 5 (PASSWORD_GRACE_TIME)
- Enter the following for Keep Password History section:
  - Keep: Unlimited (PASSWORD_REUSE_MAX)
  - Keep for: 1/1440 (PASSWORD_REUSE_TIME)
- Enter the following for Enforce Password Complexity section:
  - Complexity Function: NULL (PASSWORD_VERIFY_FUNCTION
- Enter the following for Lock Account on Failed Logon section:
  - Lock after: 2 failed login attempts (FAILED_LOGIN_ATTEMPTS)
  - Lock for: Unlimited days
- Click Apply.

Practice 14: Solutions (continued)

5 Log in as user `Jeff` supplying an invalid password. Try this twice, then log in again, this time supplying the correct password. What happens?

```
SQL> CONNECT jeff/superman
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.
SQL> CONNECT jeff/super
ERROR:
ORA-01017: invalid username/password; logon denied
SQL> CONNECT jeff/super1$
ERROR:
ORA-28000: the account is locked
```

6 Using data dictionary view `DBA_USERS` verify user `Jeff` is locked. Unlock the account for user `Jeff`. After unlocking user `Jeff` connect as `Jeff`.

Hint: Execute the `ALTER USER` command to unlock the account.

Using OEM Console

- Navigate to Security > Users > Jeff.
- Verify the Status for user Jeff is Locked on the General page.
- Select Unlocked on the General page.
- Click Apply.

Using SQL*Plus to connect as Jeff.

```
SQL> CONNECT jeff/superman
Connected.
```

Practice 14: Solutions (continued)

7 Connect as user SYS and disable password checks for the DEFAULT profile.

Hint: Execute the ALTER PROFILE command to disable the password checks.

Using OEM Console

- Navigate to Security > Profiles > Default.
- Select Password page.
- Enter Unlimited for items within the following sections
  - Expire Password
  - Keep Password History
  - Enforce Password Complexity
  - Lock Account on Failed Logon
- Click Apply.

8 Log in to user Jeff supplying an invalid password. Try this twice, then log in again, this time supplying the correct password. What happens? Why?

```
SQL> CONNECT jeff/superman
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.
SQL> CONNECT jeff/super
ERROR:
ORA-01017: invalid username/password; logon denied
SQL> CONNECT jeff/super1$
Connected.
```

Answer: Account is not locked on the third attempt to log in when using the correct password as it was in step 5. This is because the PASSWORD_VERIFY_FUNCTION was disabled in step 7 when set to NULL.

Practice 15: Solutions

1 Create user `Bob` with a password of `CRUSADER`. Make sure that any objects and temporary segments created by `Bob` are not created in the system tablespace. Also, ensure that `Bob` can log in and create objects up to one megabyte in size in the `USERS` and `INDX` tablespaces. Use `lab15_01.sql` script to grant `Bob` the ability to create sessions.

Hint: Assign `Bob` the default tablespace of `USERS` and temporary tablespace `TEMP`.

Using OEM Console

- Navigate to Security > Users.
- Select Create from the right mouse menu. The Create User dialog box will appear.
- Use the General page to identify the user and to assign default and temporary tablespaces.
- Use the the Quota page to define the quota on the USERS and INDX tablespaces.
  - Highlight the tablespace.
  - Select Value radio button.
  - Enter quota size (1) and type (M).
- Click Create.
- Click OK.

2 Create a user `Emi` with a password of `Mary`. Make sure that any objects and sort segments created by `Emi` are not created in the system tablespace.

Using OEM Console

- Navigate to Security > Users.
- Select Create from the right mouse menu. The Create User dialog box will appear.
- Use the General page to identify the user and to assign default and temporary tablespaces.
- Click Create.
- Click OK.

3 Display the information on `Bob` and `Emi` from the data dictionary.

Hint: This can be obtained by querying `DBA_USERS`.

Using OEM Console

- Navigate to Security > Users.
- User information displayed in right window pane.

Practice 15: Solutions (continued)

4 From the data dictionary, display the information on the amount of space that Bob can use in tablespaces.

Hint: This can be obtained by querying DBA_TS_QUOTAS.

Using OEM Console

- Navigate to Security > Users > BOB.
- Select Quota page.

5 a As user Bob change his temporary tablespace. What happens?

Using OEM Console

- Navigate to Security > Users > BOB.
- Select a tablespace from the Temporary drop down menu on the General page.

Note: No error would occur using OEM Console as would if connected as Bob in SQL*Plus. In OEM you are SYS / AS SYSDBA and therefore have privilege to perform this task.

5 b As user Bob, change his password to Sam.

Using OEM Console

- Navigate to Security > Users > BOB.
- Enter Password, Confirm Password.
- Click Apply.

6 As user SYSTEM, remove Bob's quota on his default tablespace.

Using OEM Console

- Navigate to Security > Users > BOB.
- Identify Bob's default tablespace from the General page.
- Select Quota page, and remove quota from Bob's default tablespace.
- Click Apply.

Practice 15: Solutions (continued)

7 Remove `Emi`'s account from the database.

Using OEM Console

- Navigate to Security > Users > EMI.
- Select Remove from right mouse menu.
- Select Yes to confirm remove.
- Click Apply.

8 `Bob` has forgotten his password. Assign him a password of `OLINK` and require that `Bob` change his password the next time he logs on.

Using OEM Console

- Navigate to Security > Users >BOB.
- Select General page.
- Enter Password, Confirm Password.
- Select Expire Password Now from the General page.
- Click Apply.

Practice 16: Solutions

1 As user `SYSTEM`, create user `Emi` with the password of `abcd12`. Give her the capability to log on to the database and create schema objects. Assign her to the default tablespace `DATA01`, and the temporary tablespace `TEMP`. Make her quota on `DATA01` 1M.

Using OEM Console

- Navigate to Security > Users.
- Select Create from the right mouse menu. The Create User dialog box will appear.
- Use the General page to identify the user and to assign default and temporary tablespaces.
- Use the Quota page to identify 1M on DATA01.
- Select CREATE SESSION and CREATE TABLE privileges from SYSTEM page.
- Click Create.
- Click OK.

2 a Run the script `lab16_02a.sql` to connect as Emi and create the tables `CUSTOMERS1` and `ORDERS1`.

```
SQL> @$HOME/STUDENT/LABS/lab16_02a.sql;
```

Practice 16: Solutions (continued)

2 b Connect as SYSTEM and copy the data from SYSTEM.CUSTOMERS to Emi's CUSTOMERS1 table. Verify that records have been inserted.

```
SQL> CONNECT system/manager
Connected.
SQL> INSERT INTO emi.customers1
  2  SELECT *
  3  FROM system.customers;
9 rows created.
SQL> SELECT * FROM emi.customers1;

CUS NAME                                               REGIO
--- -------------------------------------------------- -----
A01 TKB SPORT SHOP                                     West
A02 VOLLYRITE                                          North
A03 JUST TENNIS                                        North
A04 EVERY MOUNTAIN                                     South
A05 SHAPE UP                                           South
A06 SHAPE UP                                           West
A07 WOMENS SPORTS                                      South
A08 NORTH WOODS HEALTH AND FITNESS SUPPLY CENTER       East
J01 Sports Store                                       East

9 rows selected.
```

c As user SYSTEM give Bob the ability to select from Emi's CUSTOMERS1 table. What happens? You will receive an insufficient privilege error using SQL*Plus since you are logged in as SYSTEM and the table is owned by Emi. No error will be received when performing this task via OEM since you are logged in as SYS / AS SYSDBA.

Using OEM Console

- Navigate to Security > Users > BOB.
- Select the Object page.
- Navigate to Emi > Tables > CUSTOMERS1.
- Highlight the Select privilege from the Available Privileges window of the Object Privileges page and click on the down arrow. This will place the privilege in the Granted window.
- Click Apply.

Practice 16: Solutions (continued)

3 Reconnect as Emi and give Bob the ability to select from Emi's CUSTOMERS1 table. Also, enable Bob to give the select capability to other users. As user SYSTEM, examine the data dictionary views that record these actions.

Hint: Query the data dictionary view DBA_TAB_PRIVS to examine.

```
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers1
  2  TO bob WITH GRANT OPTION;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> COLUMN grantee    FORMAT a8
SQL> COLUMN owner      FORMAT a8
SQL> COLUMN table_name FORMAT a10
SQL> COLUMN grantor    FORMAT a8
SQL> COLUMN privilege  FORMAT a10
SQL> COLUMN grantable  FORMAT a3
SQL> COLUMN hierarchy   FORMAT a3
SQL> SELECT *
  2  FROM   dba_tab_privs
  3  WHERE  grantee='BOB';
GRANTEE OWNER TABLE_NAME GRANTOR  PRIVILEGE  GRANT HIERARCHY
------- ----- ---------- -------- ---------- ----- ---------
BOB      EMI  CUSTOMERS1 EMI      SELECT     YES   NO
```

Using OEM Console

This task has already been accomplished successfully in the previous task.

Practice 16: Solutions (continued)

4 Create user Trevor identified by diamond1$ with the capability to log on to the database.

Using OEM Console

- Navigate to `Security > Users`.
- Select `Create` from the right mouse menu. The `Create User` dialog box will appear.
- Use the `General` page to create the user.
- select `CREATE SESSION` privilege from the System page.
- Click `Create`.
- Click OK.

5 a As Bob, enable Trevor to access Emi's CUSTOMERS1 table.

Note: A password has expired message will be received due to actions in step 8 in Lesson 15. Give Bob the new password aaron$1.

```
SQL> CONNECT bob/olink
ERROR:
ORA-28001: the password has expired
Changing password for bob
New password: aaron1$
Retype new password: aaron1$
Password changed
Connected.
SQL> GRANT select ON emi.customers1 TO trevor;
Grant succeeded.
```

5 b As Emi, remove Bob's privilege to read Emi's CUSTOMERS1 table.

```
SQL> CONNECT emi/abcd12
Connected.
SQL> REVOKE select ON customers1 FROM bob;
Revoke succeeded.
```

5 c As Trevor, query Emi's CUSTOMERS1 table. What happens?

```
SQL> CONNECT trevor/diamond1$
Connected.
SQL> SELECT *
  2  FROM emi.customers1;
FROM emi.customers1
         *
ERROR at line 2:
ORA-00942: table or view does not exist
```

6 Enable Emi to start up and shut down the database without the ability to create a new database.

Using OEM Console

- Navigate to Security > Users > EMI.
- Select the System page.
- Double click on the SYSOPER privilege from the Available window. This will place the privilege in the Granted window.
- Click Apply.

Practice 17: Solutions

1 Examine the data dictionary view and list the system privileges of the `RESOURCE` role.

Using OEM Console

- Navigate to Security > Roles > Resource.
- Use the System page to view the list of system privileges assigned to the Resource role.

2 Create a role called `DEV`, which will enable a user assigned the role to create a table, create a view, and select from `Emi's CUSTOMERS1` table.

Using OEM Console

- Navigate to Security > Roles.
- Select Create from the right mouse menu.
- Identify the role DEV on the General page.
- Select the System page to grant CREATE TABLE, and CREATE VIEW.
- Select the privilege and use the down arrow to move the privlege into the Granted window.
- Select the Object page to grant SELECT on CUSTOMERS1 to DEV.
- Navigate to Emi > Tables > CUSTOMERS1.
- Highlight the SELECT privilege in the Granted window.
- Select Create.
- Click OK.

3 a Assign the `RESOURCE` and `DEV` roles to `Bob`, but make only the `RESOURCE` role automatically enabled when he logs on.

Using OEM Console

- Navigate to Security > Users > BOB.
- Use the Role page to assign the RESOURCE and DEV roles.
- Select Default in the Granted window for the RESOURCE role. This places a check in the Default box.
- Click Apply.

b Give `Bob` the ability to read all the data dictionary information.

Using OEM Console

- Navigate to Security > Users > BOB.
- Select the Role page to grant SELECT_CATALOG_ROLE.
- Deselect the check in the box for default catalog role.
- Click Apply.

Practice 17: Solutions (continued)

4 Bob needs to check the undo segments that are currently used by the instance. Connect as Bob and list the undo segments used.

Hint: Use SET ROLE SELECT_CATALOG_ROLE

```
SQL> CONNECT bob/aaron1$
Connected.
SQL> SET ROLE select_catalog_role;
Role set.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  status='ONLINE';
SEGMENT_NAME
---------------
SYSTEM
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
_SYSSMU17$
_SYSSMU18$
_SYSSMU19$
_SYSSMU20$

11 rows selected.
```

5 As SYSTEM, try to create a view CUST_VIEW on Emi's CUSTOMERS1 table. What happens?

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2    SELECT *
  3    FROM   emi.customers1;
   FROM   emi.customers1
               *
ERROR at line 3:
ORA-01031: insufficient privileges
```

Practice 17: Solutions (continued)

6 As user Emi grant select on CUSTOMERS1 to SYSTEM. As SYSTEM create a view CUST_VIEW on Emi's CUSTOMERS1 table.

```
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers1 TO system;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2     SELECT *
  3     FROM   emi.customers1;

View created.
```

Practice 19: Solutions

1 **a** Examine the data files to be used in the load in order to become familiar with the control and data file formats.

- Examine the following files if using SQL*Plus: `lcase1.ctl`, `lcase2.ctl`, and `lcase2.dat`
- Examine the following files if using Oracle Enterprise Manager: `OEMsqlcase1.ctl`, `OEMsqlcase2.ctl`, and `OEMsqlcase2.`

b As user `HR`, run the `lab19_01.sql` script to create the `DEPARTMENTS2` table.

```
SQL> CONNECT hr/hr
Connected.
SQL> @$HOME/STUDENT/LABS/lab19_01.sql
Table created.
```

Practice 19: Solutions

2 a Run SQL*Loader using the Conventional Path method to load data into the `DEPARTMENTS2` table. Use the following control file:

- Using SQL*Plus: `lcase1.ctl`
- Using Oracle Enterprise Manager: `OEMsqlcase1.ctl`

Files are located in the `LABS` directory.

Note: This runs using a control file that contains the input data.

Using OEM Console

The OEM Console must be started with an OMS to run this task.

- Navigate down the database tree to Schema > HR > Tables > DEPARTMENTS2.
- Select Data Management > Load from the right mouse menu.
- Select Next from the Load Wizard Introduction page.
- In the Control File page, specify the full path and name of the OEMsqlcase1.ctl file., and select Next. (LABS directory provided by Instructor.)
  Note: Case sensitivity in entering path and name.
- In the Data File page, select The data file is specified in the control file radio button.
- In the Load Method page, select Conventional Path radio button, and select Next.
- In the Schedule page, select Immediately radio button to run the job now, and select Next.
- In the Job Information page, specify the name for the job: LoadOEMsqlcase1, then select Submit the job now radio button, then check the box to Override Preferred Credentials: Username: HR Password: HR Connect As: SYSDBA then select Finish.
- A summary page will appear listing all of the details of the load. Review the information and select OK to continue. Note: A log file will be created to list the actions of the load.
  Note: A log file will be created to list any problems that occur with the load.
- A dialog box will appear indicating that your job has been successfully submitted, select OK to continue.
- Navigate to Jobs on the tree. In the right window, select the History tab. A listing of jobs will appear. Check the Status field for LOAD OEMsqlcase1. It will indicate Completed indicating the job was successful.

2 b Examine the log file using operating system command.

```
$  cd $HOME
$  more lcase1.log
   -- Note:  Path used:  Conventional
```

c Query the DEPARTMENTS2 table to check the data.

```
SQL> SELECT * FROM DEPARTMENTS2;
   DEPT_ID DEPT_NAME
   ------- ------------------
        10 Administration
        20 Marketing
        30 Purchasing
        40 Human Resources
        50 Shipping
        60 IT
        70 Public Relations
        …
        27 rows selected
```

3 Delete all the records in the DEPARTMENTS2 table.

```
SQL> TRUNCATE TABLE departments2;
Table truncated.
```

Practice 19: Solutions (continued)

4 a Run SQL*Loader in Direct-Path mode to load data into the `DEPARTMENTS2` table. Use the following control files:

- Using SQL*Plus: `lcase2.ctl`
- Using Oracle Enterprise Manager: `OEMsqlcase2.ctl`

Files are located in the `LABS` directory.

Note: This runs using an input data file to load data.

Using OEM Console

- Navigate down the database tree to Schema > HR > Tables > DEPARTMENTS2.
- Select Data Management > Load from the right mouse menu.
- Select Next from the Load Wizard Introduction page.
- In the Control File page, specify the full path and name of the OEMsqlcase2.ctl file (LABS directory provided by Instructor.), and select Next. **Note:** Case sensitivity in entering path and name.
- In the Data File page, select Provide the full path and name on the database server machine radio button.
- Enter the full path where the OEMsqlcase2.dat data file exists, and select Next. **Note:** Case sensitivity in entering path and name.
- In the Load Method page, select Direct Path radio button, and select Next.
- In the Schedule page, select Immediately radio button to run the job now, and select Next.
- In the Job Information page, specify the name for the job: LoadOEMsqlcase2, then select Submit the job now radio button, then check the box to Override Preferred Credentials: Username: HR Password: HR Connect As: SYSDBA then select Finish.
- A summary page will appear listing all of the details of the load. Review the information and select OK to continue. Note: A log file will be created to list the actions of the load.
- A dialog box will appear indicating that your job has been successfully submitted, select OK to continue.
- Navigate to Jobs on the tree. In the right window, select the History tab. A listing of jobs will appear. Check the Status field for Load OEMsqlcase2. It will indicate Completed indicating the job was successful.

4 b Examine the log file using operating system command.

```
$ cd $HOME
$ more lcase2.log
   -- Note:  Path used:  Direct
```

c Query the DEPARTMENTS2 table to check the data.

```
SQL> SELECT * FROM DEPARTMENTS2;
   DEPT_ID DEPT_NAME
   ------- ------------------
        10 Administration
        20 Marketing
        30 Purchasing
        40 Human Resources
        50 Shipping
        60 IT
        70 Public Relations
        80 Sales
        90 Executive
        …
        27 rows selected.
```

Practice 19: Solutions (continued)

5 a As user HR, create a table EMPLOYEES2 as select from the EMPLOYEES table. Truncate the table. Then select from the EMPLOYEES2 table to verify no data is in the table.

```
SQL> CONNECT hr/hr
SQL> CREATE TABLE employees2
  2  AS SELECT * FROM employees;
Table created.
SQL> TRUNCATE TABLE employees2;
Table truncated.
SQL> SELECT * FROM employees2;
no rows selected
```

b Perform a direct-load insert into EMPLOYEES2 from EMPLOYEES and query EMPLOYEES2 to verify the load.

```
SQL> INSERT /*+ append */ INTO employees2
  2  NOLOGGING
  3  SELECT * from employees;
107 rows created.
SQL> COMMIT;
SQL> SELECT employee_id, first_name, last_name
  2  FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ------------------ -------------------------
139   John                  Seo
140   Joshua                Patel
141   Trenna                Rajs
142   Curtis                Davies
…
107 rows selected.
```

Practice 19: Solutions (continued)

6 a Truncate the EMPLOYEES2 table once again. Then select from the EMPLOYEES2 table to verify no data is in the table.

```
SQL> TRUNCATE TABLE employees2;
Table truncated.
SQL> SELECT * FROM employees2;
no rows selected
```

b Restore the data with a parallel direct-load insert from the EMPLOYEES table. Specify a degree of parallelism of two. Verify the data.

```
SQL> ALTER SESSION ENABLE PARALLEL DML;
Session altered
SQL> INSERT /*+ parallel(employees2,2) */
  2  INTO employees2 NOLOGGING
  3  SELECT * FROM employees;
107 rows created.
SQL> COMMIT;
Commit complete.
SQL> SELECT employee_id, first_name, last_name
  2 FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ------------------ -------------------------
139   John                  Seo
140   Joshua                Patel
141   Trenna                Rajs
142   Curtis                Davies
…
107 rows selected.
```

Practice 20: Solutions

1 Check the database and the national character set.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT parameter, value
  2  FROM   nls_database_parameters
  3  WHERE  parameter LIKE '%CHARACTERSET%';

PARAMETER                      VALUE
------------------------------ --------------
NLS_CHARACTERSET               WE8ISO8859P1
NLS_NCHAR_CHARACTERSET         AL16UTF16
```

2 Which are valid values for the database character set.

```
SQL> SELECT   value
  2  FROM     v$nls_valid_values
  3  WHERE    parameter = 'CHARACTERSET'
  4  ORDER BY value;

VALUE
------------------
AL16UTF16
AL24UTFFSS
AL32UTF8
AR8ADOS710
AR8ADOS710T
AR8ADOS720
AR8ADOS720T
.
.
.
257 rows selected.
```

Practice 20: Solutions (continued)

3 Make sure that all dates in this session are displayed using a 4-digit year. Change `NLS_LANGUAGE` to `FRENCH`. Select `sysdate` from `DUAL`.

```
SQL> ALTER SESSION SET nls_date_format = 'DD-MON-YYYY';
Session altered.
SQL> ALTER SESSION SET nls_language = FRENCH;
Session modifiée.
SQL> SELECT sysdate
  2  FROM   dual;
SYSDATE
-----------
24-Jul-2002
```