Oracle9*i* DBA Fundamentals I

Volume 2 • Student Guide

D11321GC10
Production 1.0
May 2001
D32644

Authors

Sarath Chandran

Marie St. Gelais

S Matt Taylor Jr

Technical Reviewers

Howard Bradley

Ruth Baylis

Paul Cartwright

Trevor Davis

Joel Goodman

Scott Gossett

Tomoki Ishii

Peter Kilpatrick

Stefan Lindblad

Howard Ostrow

Ashesh Parekh

Radhanes Petronilla

Venceslava Pretlova

Andreas Reinhardt

Ranbir Singh

Peter Sharman

Barry Trute

Ric VanDyke

Karla Villasenor

John Watson

Teppei Yagihashi

Publisher

Shane Mattimoe

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

Contents

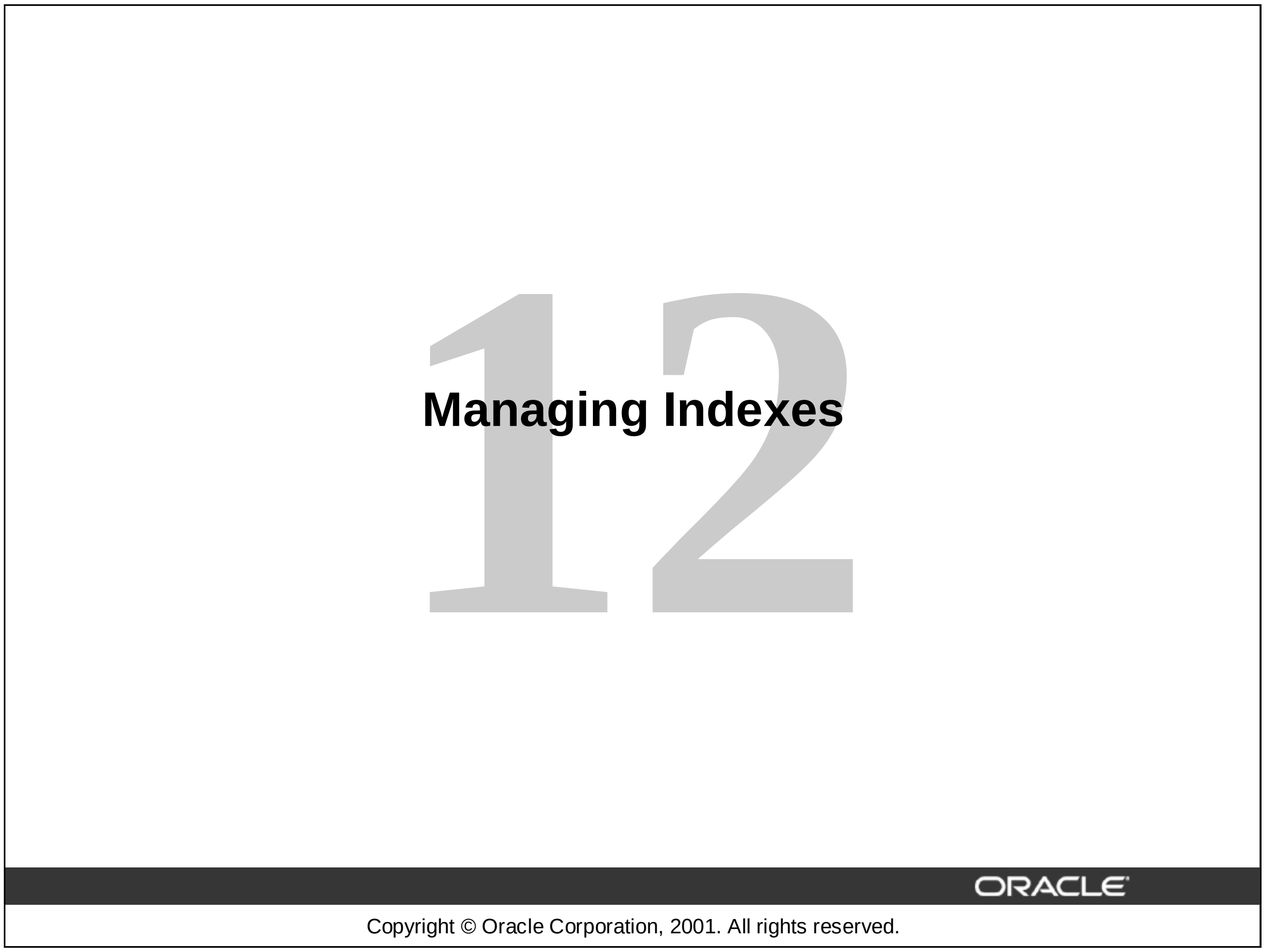
Introduction

1 Oracle Architectural Components

2 Getting Started With the Oracle Server

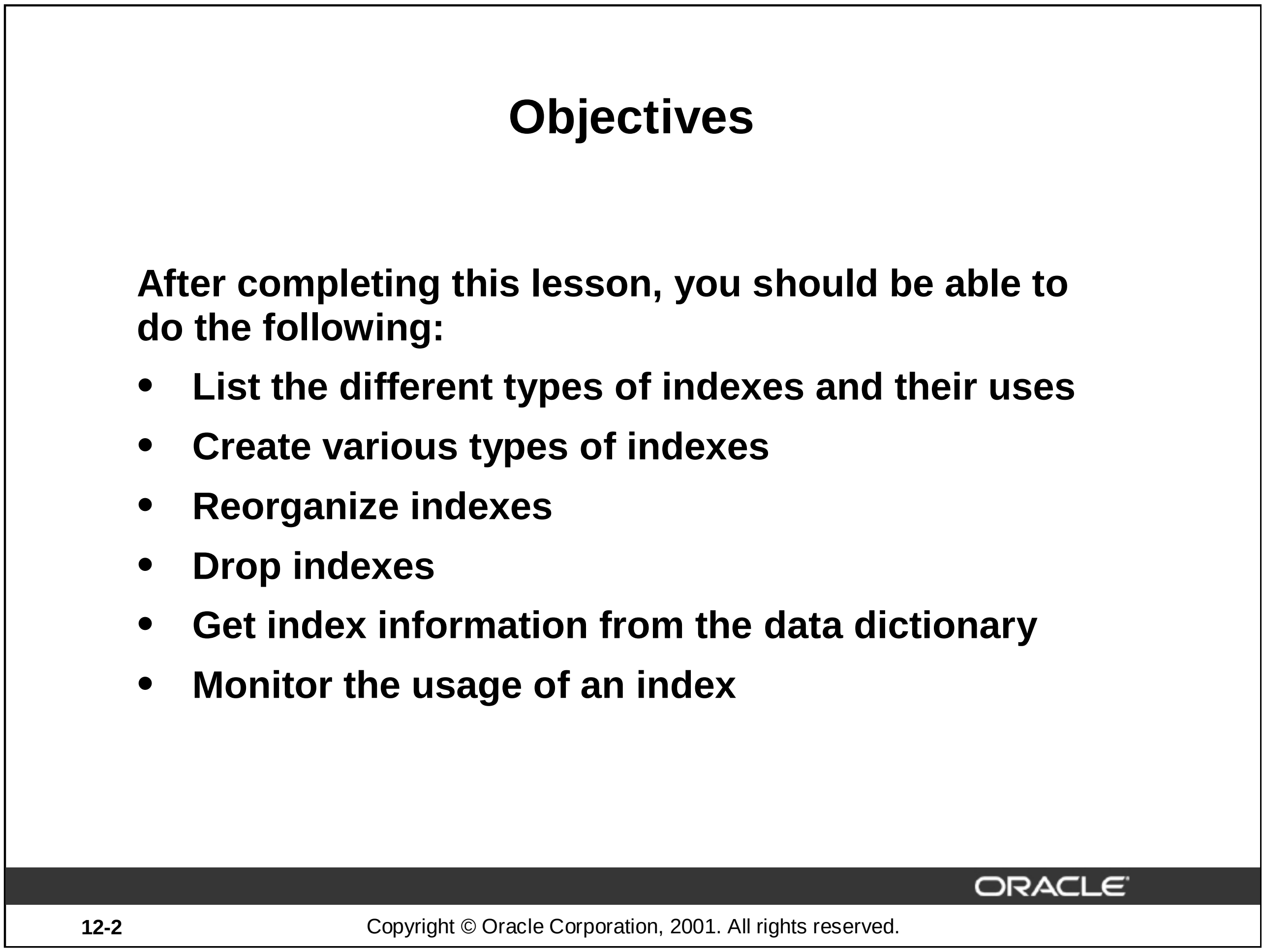
7 Maintaining Redo Log Files

8 Managing Tablespaces and Data files



12

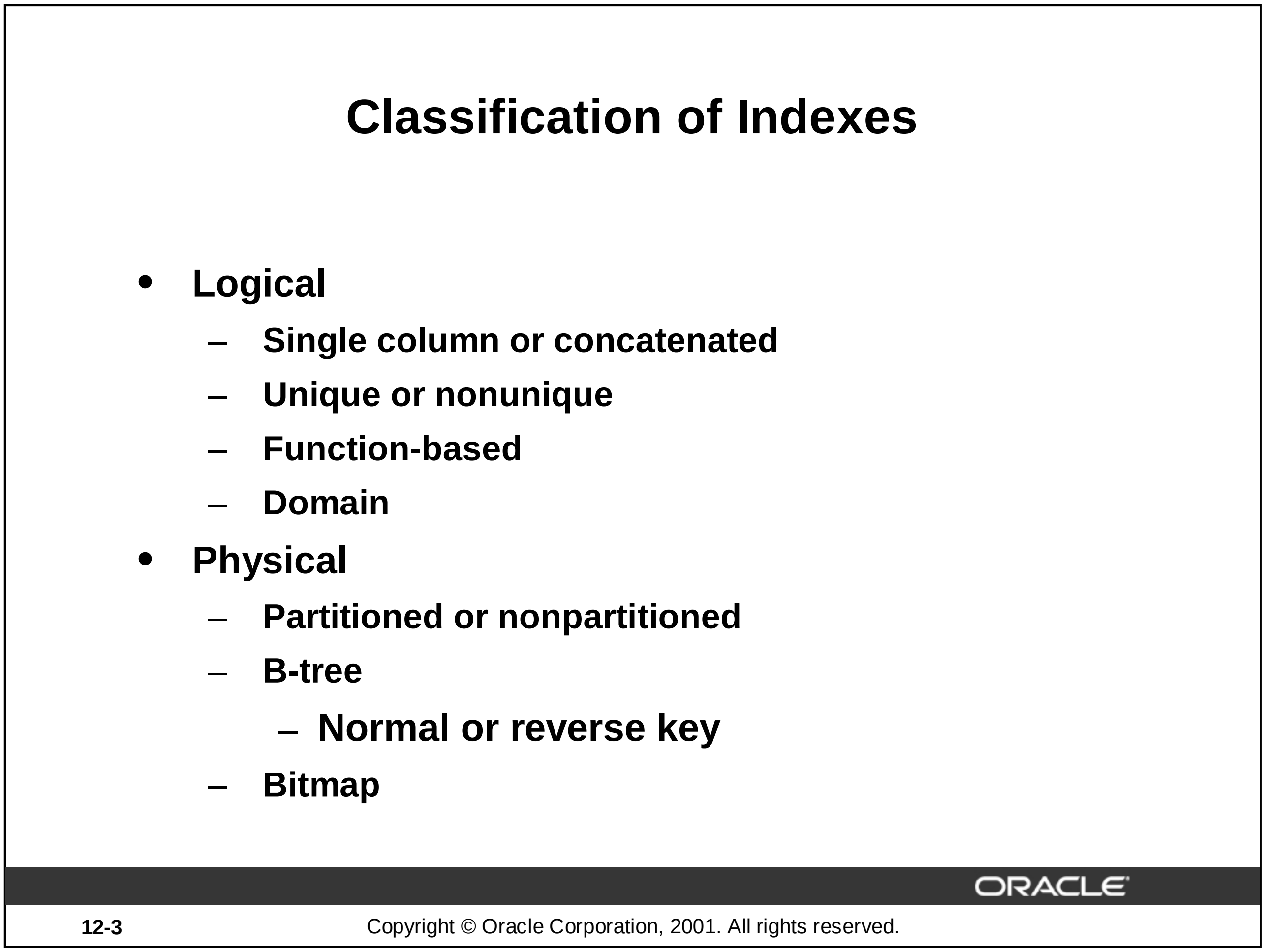
Managing Indexes



Objectives

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

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



Classification of Indexes

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

12-3

Classification of Indexes

An index is a tree structure that allows direct access to a row in a table. Indexes can be classified based on their logical design or on their physical implementation. The logical classification groups indexes from an application perspective, while the physical classification is derived from the way the indexes are stored.

Single Column and Concatenated Indexes

A *single column index* has only one column in the index key—for example, an index on the `employees` number column of an `employees` table.

A *concatenated index*, also known as a composite index, is created on multiple columns in a table. Columns in a concatenated index do not need to be in the same order as the columns in the table, nor do they need to be adjacent—for example, an index on the department and job columns of an employee table.

The maximum number of columns in a composite key index is 32. However, the combined size of all the columns cannot exceed roughly one-third of the data block size.

Classification of Indexes (continued)

Unique and Nonunique Indexes

A unique index guarantees that no two rows of a table have duplicate values in the column that defines the index. An index key in a unique index can point to only one row in the table.

In a nonunique index, a single key can have multiple rows associated with it, and can be used to enforce uniqueness.

Function-Based Indexes

A function-based index is created when using functions or expressions that involve one or more columns in the table being indexed. A function-based index precomputes the value of the function or expression and stores it in the index. Function-based indexes can be created as either a B-tree or a bitmap index.

Domain Indexes

A domain index is an application-specific (Text, Spatial) index that is created, managed, and accessed by routines supplied by an indextype. It is called a domain index because it indexes data in application-specific domains.

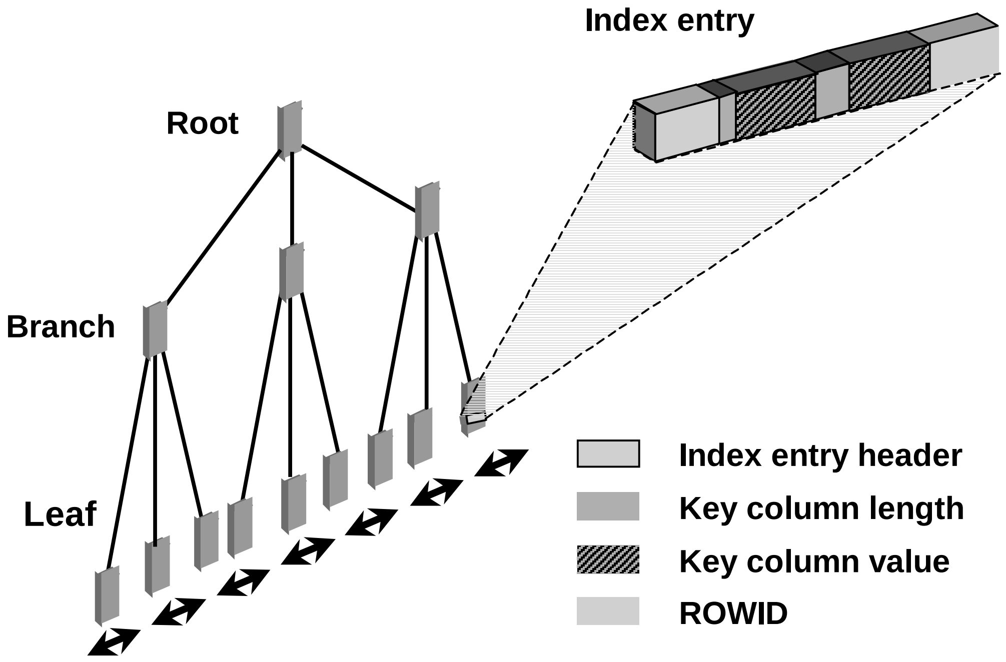
Only single-column domain indexes are supported. You can build single-column domain indexes on columns having scalar, object, or LOB datatypes.

Partitioned and Nonpartitioned Indexes

Partitioned indexes are used for large tables to store index entries corresponding to an index in several segments. Partitioning allows an index to be spread across many tablespaces, decreasing contention for index lookup, and increasing manageability. Partitioned indexes are often used with partitioned tables to improve scalability and manageability. An index partition can be created for each table partition.

This lesson discusses the creation and maintenance of nonpartitioned B-tree and bitmap indexes.

B-Tree Index



How Indexes Are Stored

Although all the indexes use a B-tree structure, the term B-tree index is usually associated with an index that stores a list of ROWIDS for each key.

Structure of a B-Tree Index

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

Format of Index Leaf Entries

An index entry is made up of the following components:

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

Index Leaf Entry Characteristics

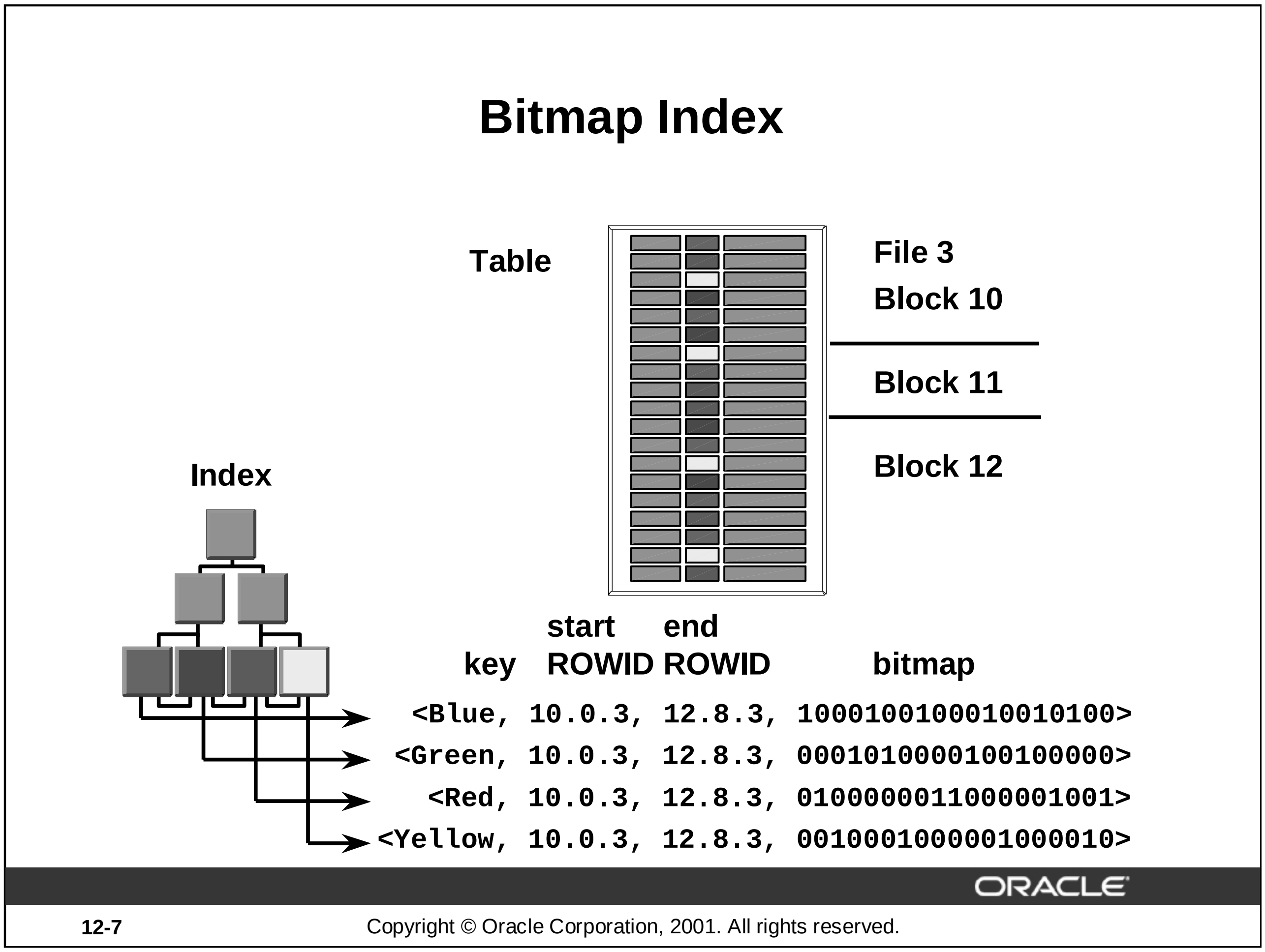
In a B-tree index on a nonpartitioned table:

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

Effect of DML Operations on an Index

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

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



Bitmap Index

```
             start   end
        key  ROWID   ROWID          bitmap
  <Blue, 10.0.3, 12.8.3, 1000100100010010100>
 <Green, 10.0.3, 12.8.3, 0001010000100100000>
   <Red, 10.0.3, 12.8.3, 0100000011000001001>
<Yellow, 10.0.3, 12.8.3, 0010001000001000010>
```

12-7 Copyright © Oracle Corporation, 2001. All rights reserved.

Bitmap Indexes

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

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

Structure of a Bitmap Index

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

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

- An entry header, containing the number of columns and lock information
- Key values consisting of length and value pairs for each key column (In the example, the key consists of only one column, and the first entry has a key value of Blue.)

Structure of a Bitmap Index (continued)

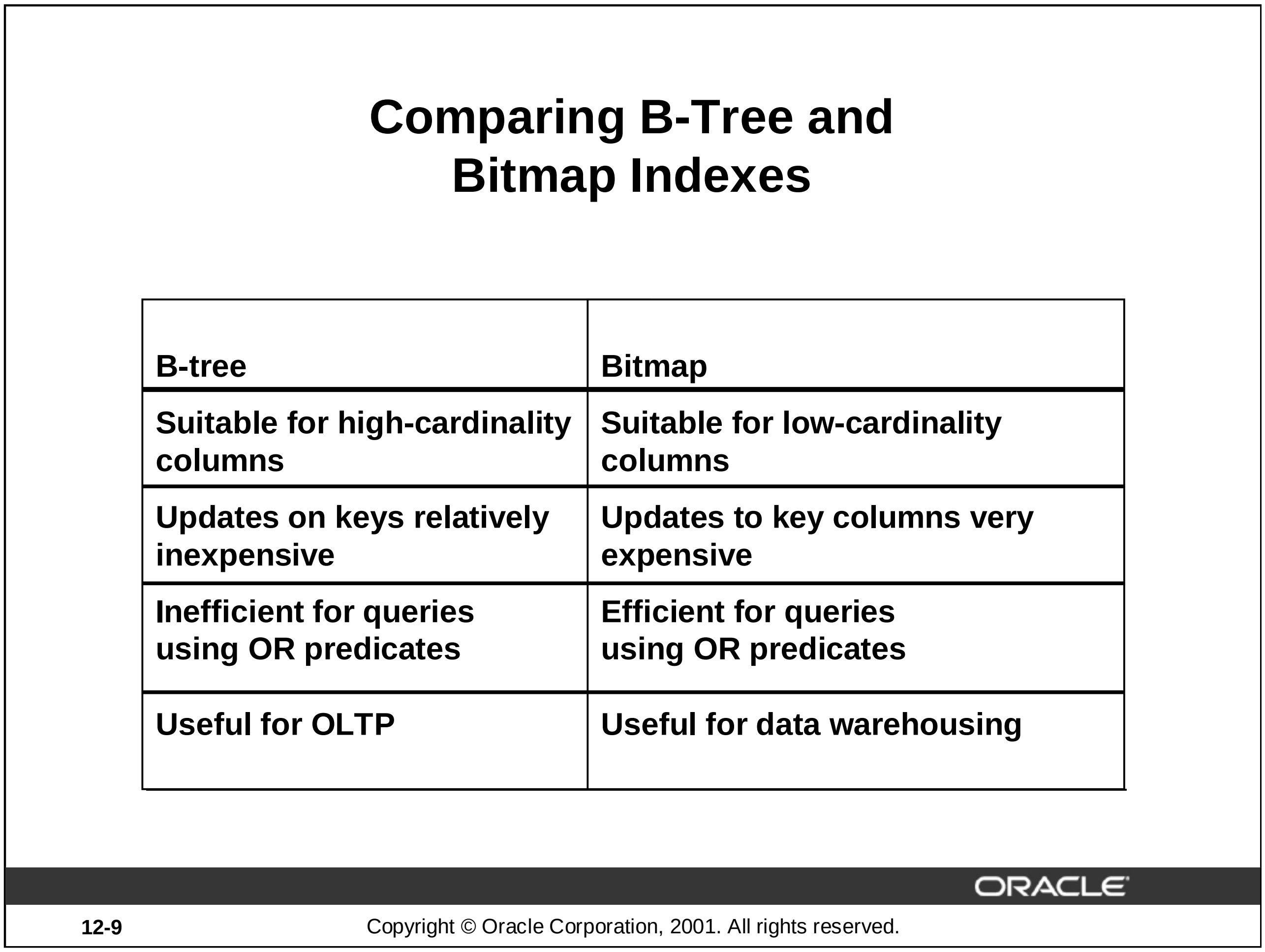
- Start ROWID, which in the example contains a file number 3, a block number 10, and a row number 0
- End ROWID, which in the example includes a block number 12 and a row number 8
- A bitmap segment consisting of a string of bits (The bit is set when the corresponding row contains the key value and is unset when the row does not contain the key value. The Oracle server uses a patented compression technique to store bitmap segments.)

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

Using a Bitmap Index

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

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



Comparing B-Tree and Bitmap Indexes

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

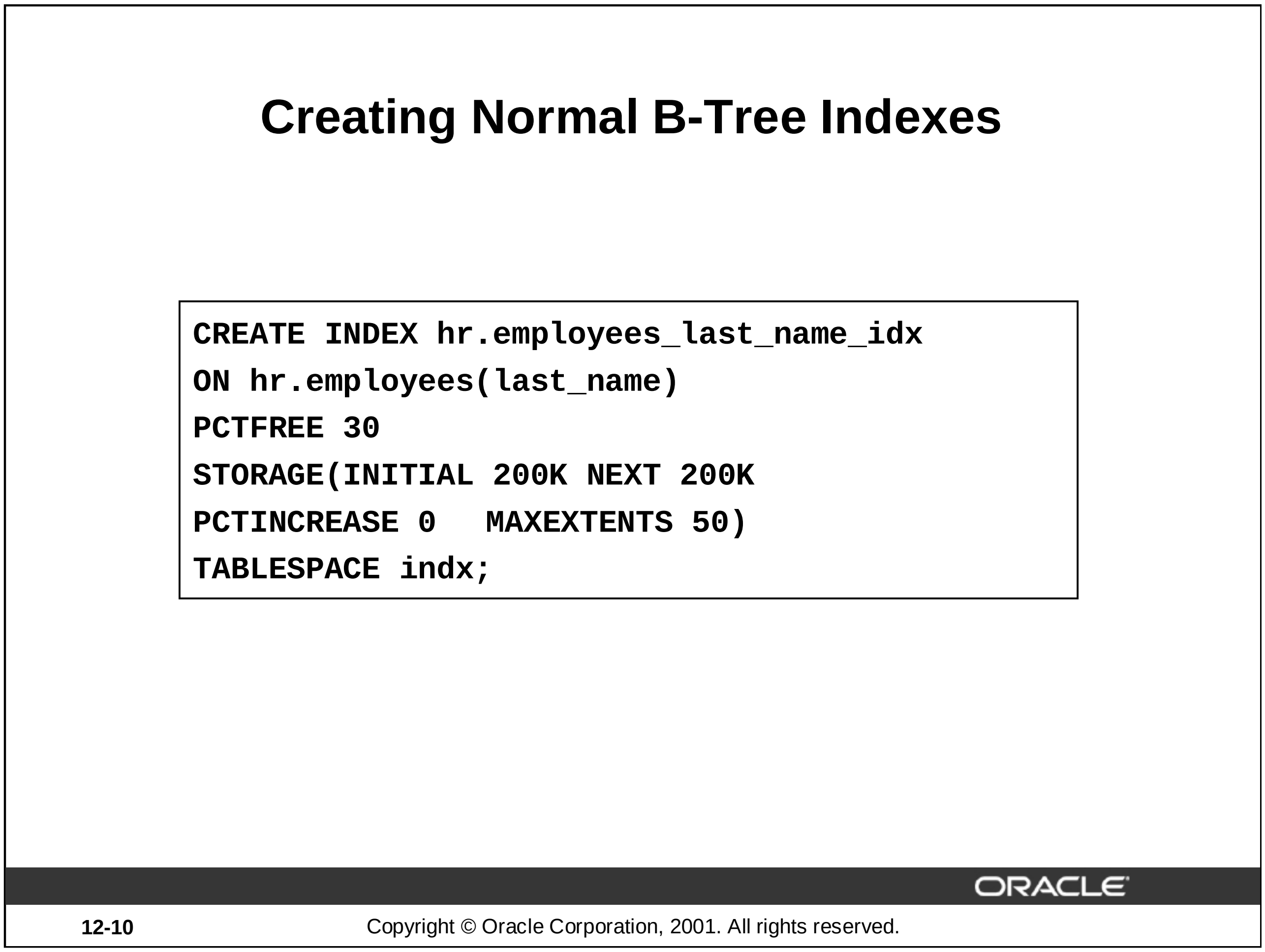
Comparing B-Tree and Bitmap Indexes

Bitmap indexes are more compact than B-tree indexes when used with low-cardinality columns.

Updates to key columns in a bitmap index are more expensive because bitmaps use bitmap-segment-level locking, whereas in a B-tree index, locks are on entries corresponding to individual rows of the table.

Bitmap indexes can be used to perform operations such as Bitmap Boolean. The Oracle server can use two bitmap segments to perform a bitwise boolean and get a resulting bitmap. This allows efficient use of bitmaps in queries that use the boolean predicate.

In summary, B-tree indexes may be more suitable in an OLTP environment for indexing dynamic tables, whereas bitmap indexes may be useful in data warehouse environments where complex queries are used on large, static tables.



Creating Normal B-Tree Indexes

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

12-10

Creating Normal B-Tree Indexes

An index can be created either in the account of the user who owns the table or in a different account, although it is generally created in the same account as the table.

The syntax above creates an index on the EMPLOYEES table using the LAST_NAME column.

Syntax (continued)

where:

UNIQUE
is used to specify a unique index
(Nonunique is the default.)

schema
is the owner of the index/table

index
is the name of the index

table
is the name of the table

column
is the name of the column

ASC/ DESC
indicates whether the index should be created in ascending or decending order.

TABLESPACE
identifies the tablespace where the index will be created

PCTFREE
is the amount of space reserved in each block (in percentage of total space minus the block header) at the time of creation for accommodating new index entries

INITRANS
specifies the number of transaction entries preallocated in each block
(The default and the minimum value is 2.)

MAXTRANS
limits the number of transaction entries that can be allocated to each block
(The default is 255.)

STORAGE clause
identifies the storage clause that determines how extents are allocated to the index

LOGGING
specifies that the creation of the index and subsequent operations on the index are logged in the redo log file
(This is the default.)

NOLOGGING
specifies that the creation and certain types of data loads are not logged in the redo log file

NOSORT
specifies that the rows are stored in the database in ascending order, and therefore, the Oracle server does not have to sort the rows while creating the index

Syntax (continued)

Note

- If MINIMUM EXTENT has been defined for the tablespace, the extent sizes for the index are rounded up to the next higher multiple of the MINIMUM EXTENT value.
- If the [NO]LOGGING clause is omitted, the logging attribute of the index defaults to the logging attribute of the tablespace in which it resides.
- PCTUSED cannot be specified for an index. Because index entries must be stored in the correct order, the user cannot control when an index block is used for inserts.
- If the NOSORT keyword is used when the data is not sorted on the key, the statement terminates with an error. This option is likely to fail if the table has had several DML operations on it.
- The Oracle server uses existing indexes to create a new index, if possible. This happens when the key for the new index corresponds to the leading part of the key of an existing index.

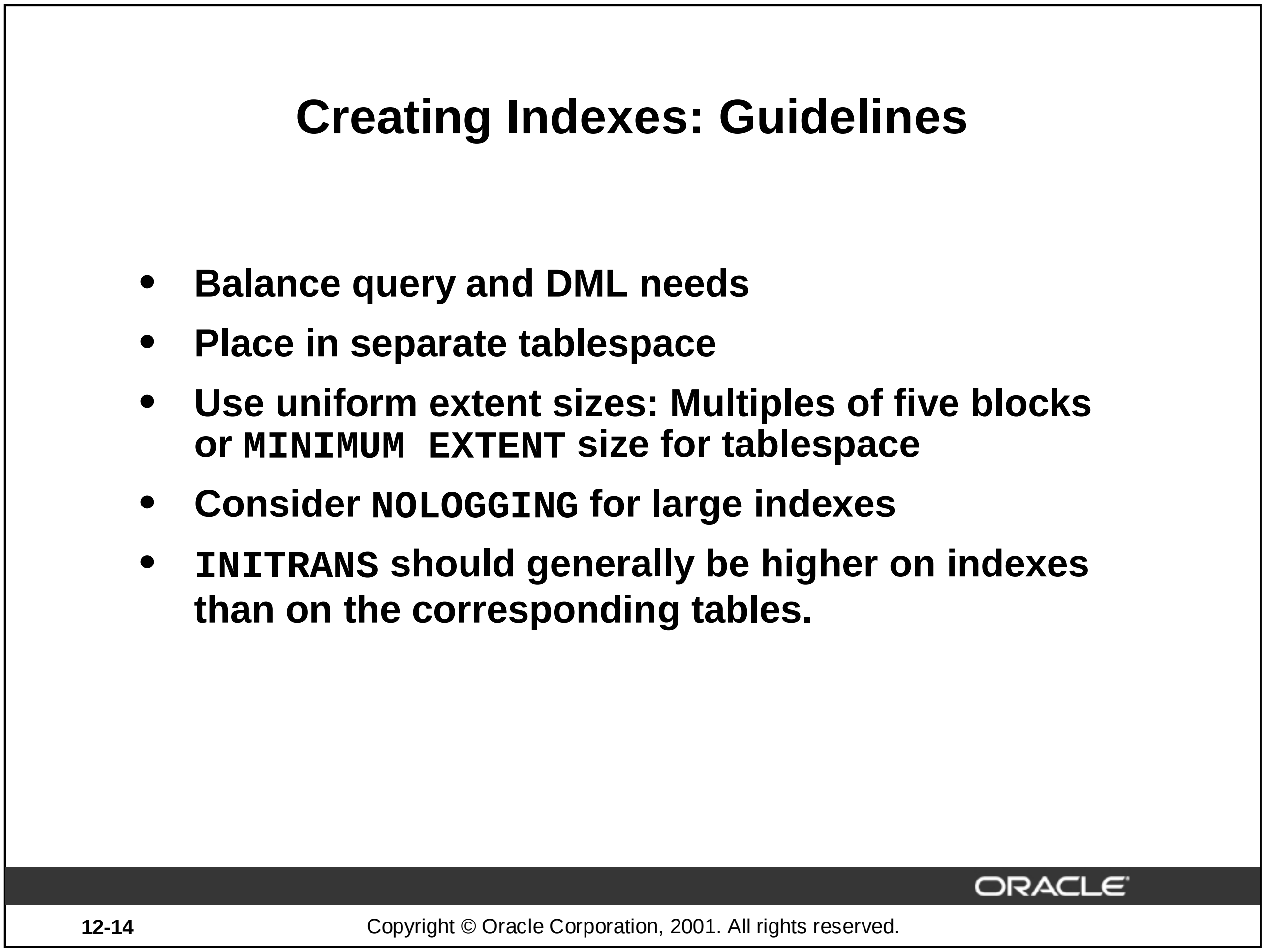
Using Oracle Enterprise Manager to Create an Index

Launch Schema Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Schema folder and select Index in the navigator tree
4. Choose Object—>Create.
5. Select Index from the list of values and then click Create
6. Enter General, Partitions, Storage, and Options information in the property sheet.
7. Click Create.

   While using Schema Manager, the user also has the option to let the tool automatically define the storage and block utilization parameters based on an estimate of the initial volume, the growth rate, the insert activity on the table, and the order in which rows are inserted.

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



Creating Indexes: Guidelines

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

12-14

Guidelines When Creating Indexes

Consider the following while creating an index:

- Indexes speed up query performance and slow down DML operations. Always minimize the number of indexes needed on volatile tables.
- Place indexes in a separate tablespace, not in a tablespace that has rollback segments, temporary segments, and tables.
- There could be significant performance gain for large indexes by avoiding redo generation. Consider using the `NOLOGGING` clause for creating large indexes.
- Because index entries are smaller compared to the rows they index, index blocks tend to have more entries per block. For this reason, `INITRANS` should generally be higher on indexes than on the corresponding tables.

Indexes and PCTFREE

The `PCTFREE` parameter for an index works differently from that of a table. This parameter is used only during creation of the index to reserve space for index entries that may need to be inserted into the same index block. Index entries are not updated. When a key column is updated, this involves a logical delete of the index entry and an insert.

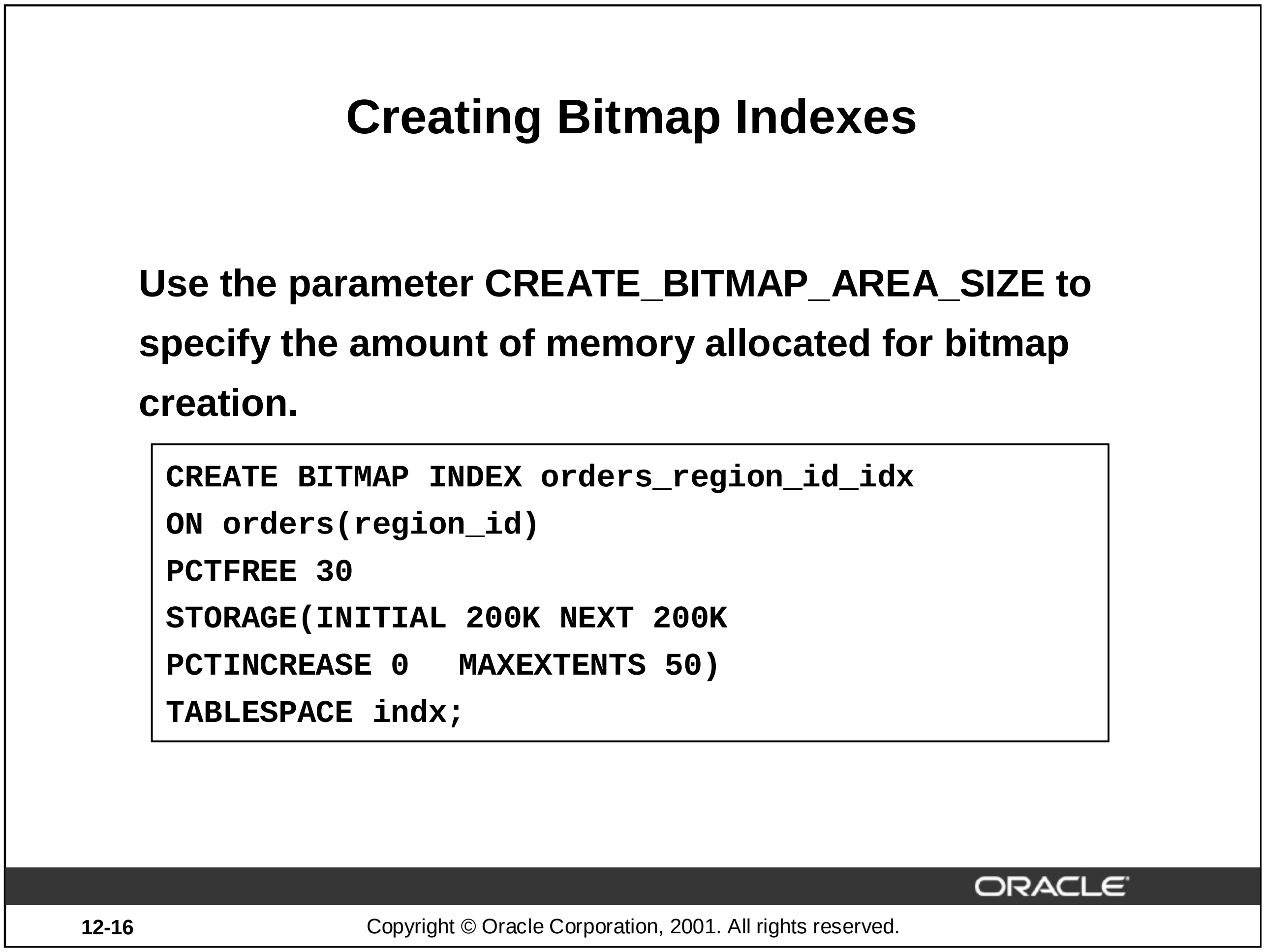
Indexes and PCTFREE (continued)

Use a low `PCTFREE` for indexes on columns that are monotonically increasing, such as a system-generated invoice number. In these cases, new index entries are always appended to the existing entries and there is no need to insert a new entry between two existing index entries.

Where the value for an indexed column of an inserted row can take on any value, that is, the new value can fall within the current range of values—you should provide for a higher `PCTFREE.` An example of an index requiring a high `PCTFREE` is an index on the customer code column of an invoice table. In this case, it is useful to specify a value of `PCTFREE` as indicated by the following equation:

$$\frac{\text{Maximum number of rows} - \text{Initial number of rows}}{\text{Maximum number of rows}} \text{ X } 100$$

The maximum value can cater to a specific time period, such as a year.



Creating Bitmap Indexes

Use the parameter CREATE_BITMAP_AREA_SIZE to specify the amount of memory allocated for bitmap creation.

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

Syntax

Use the following command to create a bitmap index:

```
CREATE BITMAP INDEX [schema.] index
ON [schema.] table
(column [ ASC | DESC ] [ , column [ASC | DESC ] ] ...)
[ TABLESPACE tablespace ]
    [ PCTFREE integer ]
    [ INITRANS integer ]
    [ MAXTRANS integer ]
    [ storage-clause ]
    [ LOGGING| NOLOGGING ]
    [ NOSORT ]
```

Notice that a bitmap index cannot be unique.

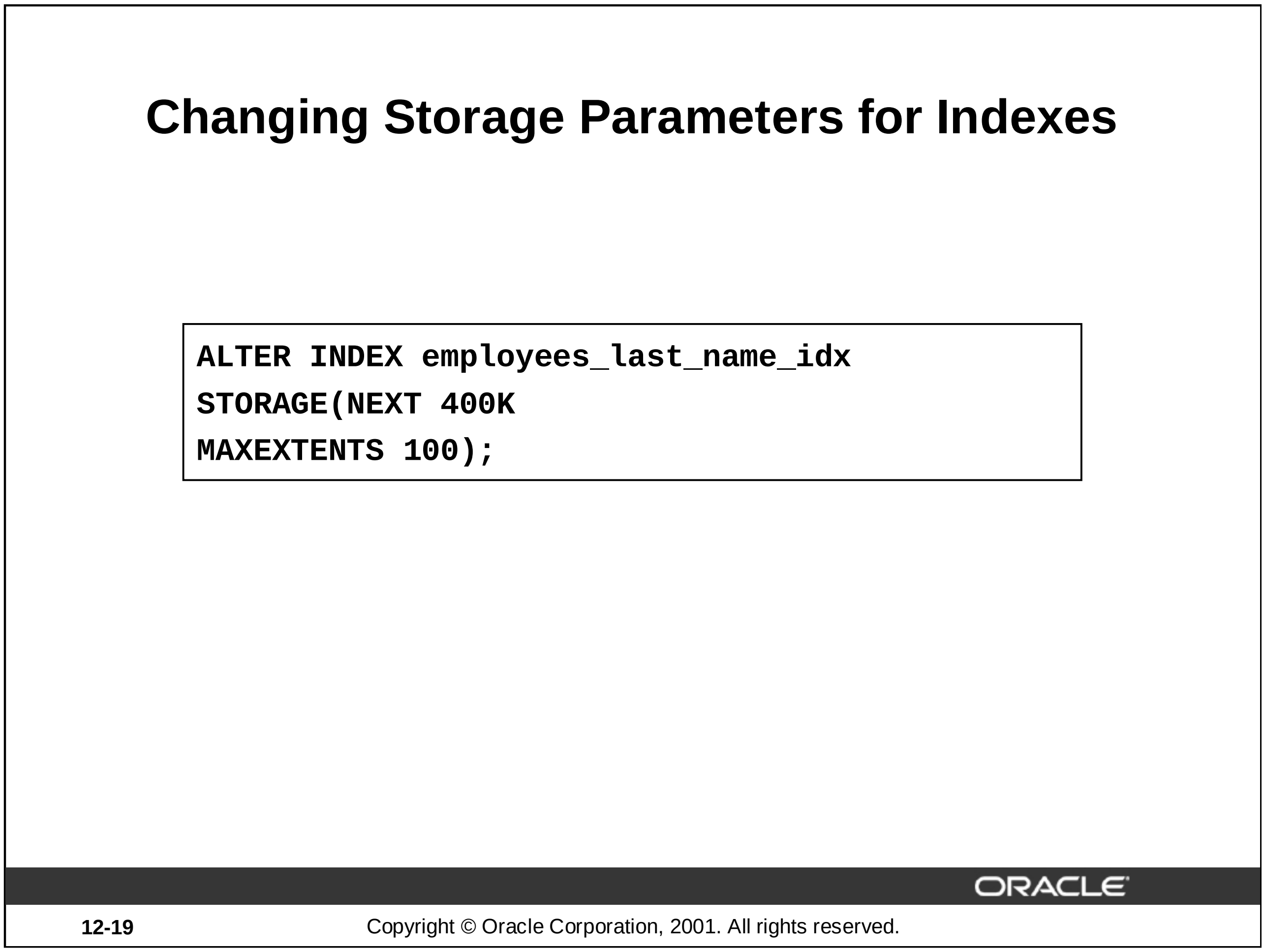
CREATE_BITMAP_AREA_SIZE

The initialization parameter `CREATE_BITMAP_AREA_SIZE` determines the amount of space that will be used for storing bitmap segments in memory. The default value is 8 MB. A larger value may lead to a faster index creation. If cardinality is very small, this value can be set to a small value. For example, if cardinality is only 2, then the value can be in the order of kilobytes rather than megabytes. As a general rule, for a higher cardinality, more memory is needed for optimal performance.

Using Oracle Enterprise Manager to Create a Bitmap Index

Launch Schema Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Schema folder and select Index in the navigator tree
4. Select Object—>Create from the menu
5. Select Index from the list of values and then click Create.
6. Enter General, Partitions, Storage, and Options information in the property sheet, and make sure that the Bitmap option is selected in the General page.
7. Click Create.



Changing Storage Parameters for Indexes

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

Changing Storage Parameters for Indexes

Some of the storage parameters and block utilization parameters can be modified by using the `ALTER INDEX` command.

Syntax

```
ALTER INDEX [schema.]index
[ storage-clause ]
[ INITRANS integer ]
[ MAXTRANS integer ]
```

The implications of changing the storage parameters for an index are the same as the implications of changing them for a table. A common use of this change is to increase the `MAXEXTENTS` for an index.

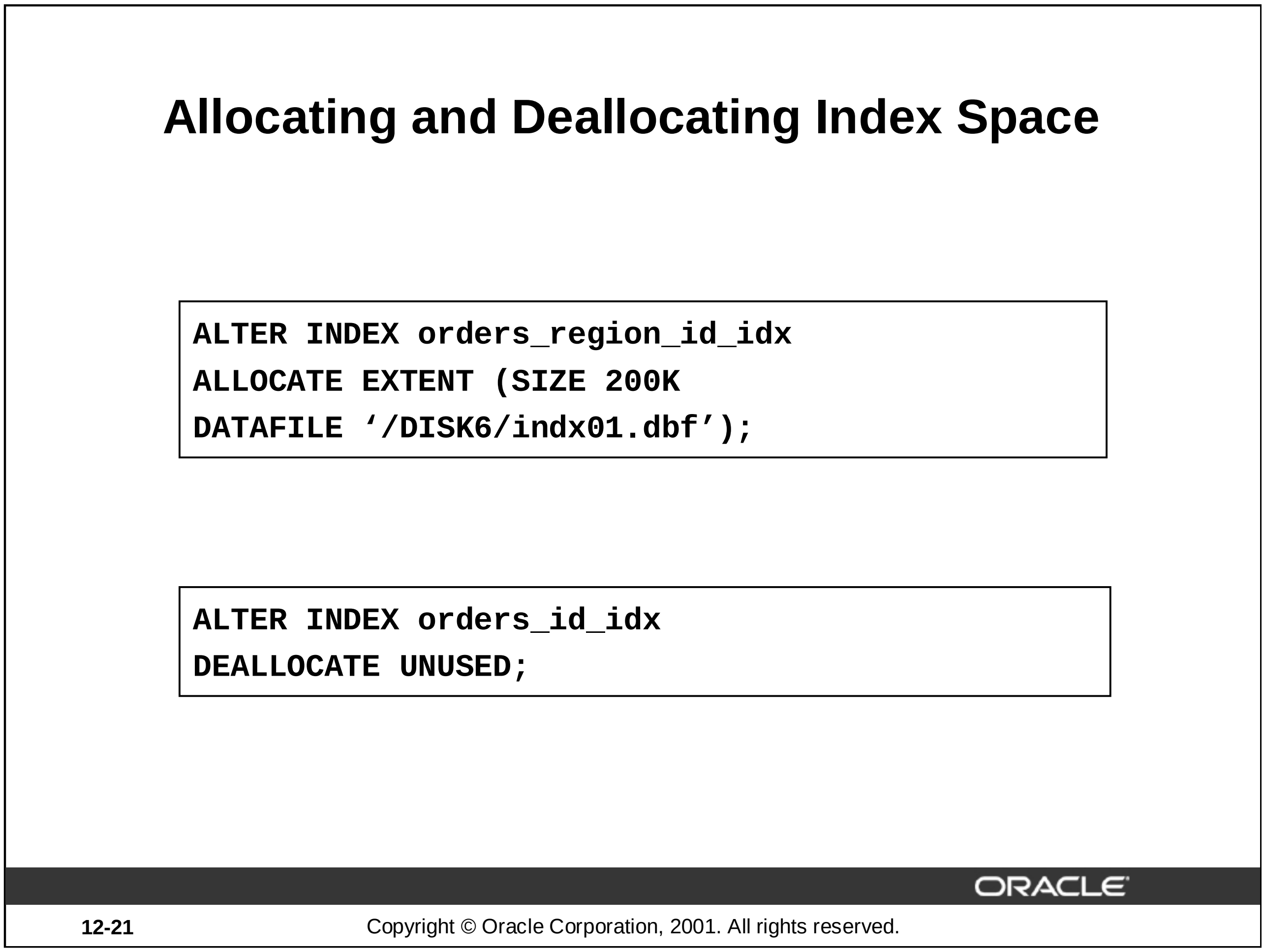
Block utilization parameters may be changed to guarantee higher levels of concurrency on an index block.

Using Oracle Enterprise Manager to Change Storage Parameters

Launch Schema Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand Schema folder and expand on Index folder in the navigator tree
4. Expand the user name (or schema).
5. Select the index.
6. Modify the values in the Storage tab of the property sheet.
7. Click Apply.

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



Allocating and Deallocating Index Space

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

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

Manual Allocation of Space to an Index

It may be necessary to add extents to an index before a period of high insert activity on a table. Adding extents prevents dynamic extension of indexes and the resulting degradation in performance.

Manual Deallocation of Space from an Index

Use the `DEALLOCATE` clause of the `ALTER INDEX` command to release unused space above the high-water mark in an index.

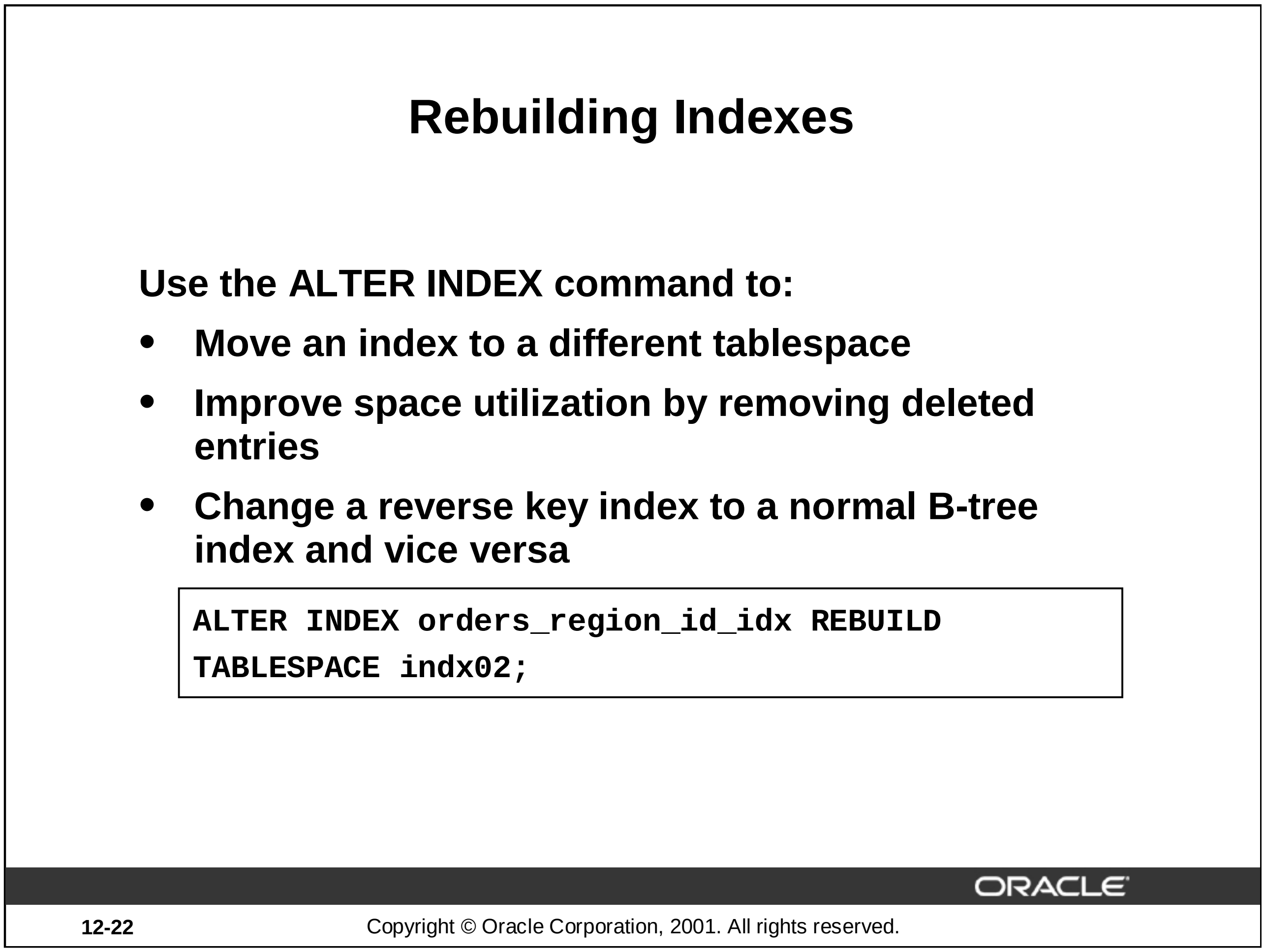
Syntax

Use the following command to allocate or deallocate index space:

```
ALTER INDEX [schema.]index
{ALLOCATE EXTENT ([SIZE integer [K|M]]
                 [ DATAFILE ‘filename’ ])
| DEALLOCATE UNUSED [KEEP integer [ K|M ] ] }
```

Manual allocation and deallocation of space for an index follow the same rules as those that are used when using these commands against a table.

Note: Index space is deallocated when the table on which the index built is truncated. Truncating a table results in truncation of the associated index.



Rebuilding Indexes

Use the ALTER INDEX command to:

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

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

Rebuilding Indexes

Index rebuilds have the following characteristics:

- A new index is built using an existing index as the data source.
- Sorts are not needed when an index is built using an existing index, resulting in better performance.
- The old index is deleted after the new index is built. During the rebuild, sufficient space is needed to accommodate both the old and the new index in their respective tablespaces.
- The resulting index does not contain any deleted entries. Therefore, this index uses space more efficiently.
- Queries can continue to use the existing index while the new index is being built.

Possible Rebuild Situations

Rebuild an index in the following situations:

- The existing index needs to be moved to a different tablespace. This may be necessary if the index is in the same tablespace as the table or if objects need to be redistributed across disks.

Possible Rebuild Situations (continued)

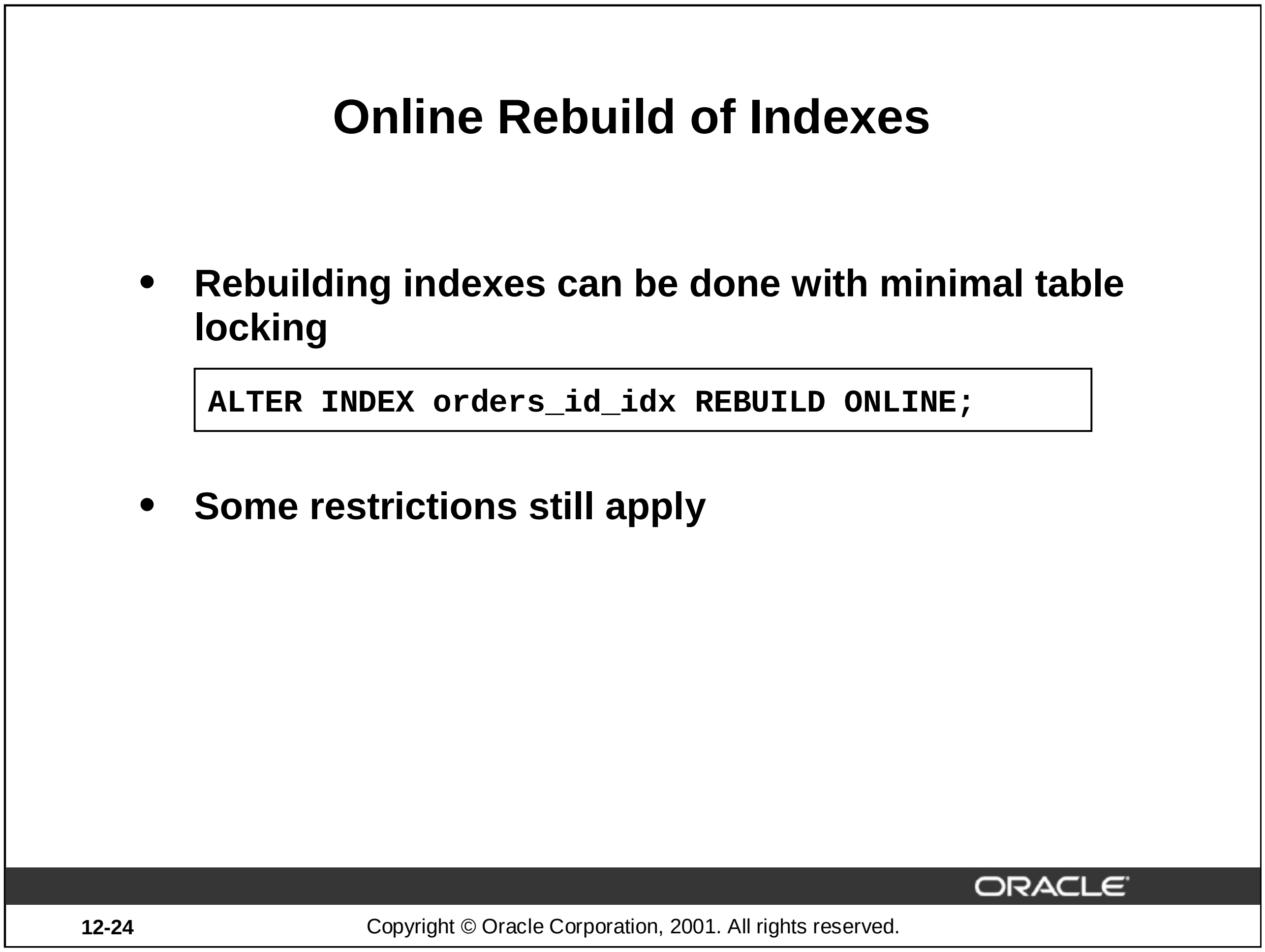
- An index contains many deleted entries. This is a typical problem with sliding indexes, such as an index on the order number of an orders table, where completed orders are deleted and new orders with higher numbers are added to the table. If a few old orders are outstanding, there may be several index leaf blocks with all but a few deleted entries.
- An existing normal index needs to be converted into a reverse key index. This may be the case when migrating applications from an earlier release of the Oracle server.
- The table of the index has been moved to another tablespace using the `ALTER TABLE ... MOVE TABLESPACE` command.

Syntax

Use the following command to rebuild an index:

```
ALTER INDEX [schema.] index REBUILD
[ TABLESPACE tablespace ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ storage-clause ]
[ LOGGING| NOLOGGING ]
[ REVERSE | NOREVERSE ]
```

The `ALTER INDEX ... REBUILD` command cannot be used to change a bitmap index to B-tree and vice versa. The `REVERSE` or `NOREVERSE` keyword, can only be specified for B-tree indexes.



Online Rebuild of Indexes

- Rebuilding indexes can be done with minimal table locking

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

- Some restrictions still apply

Rebuilding Indexes Online

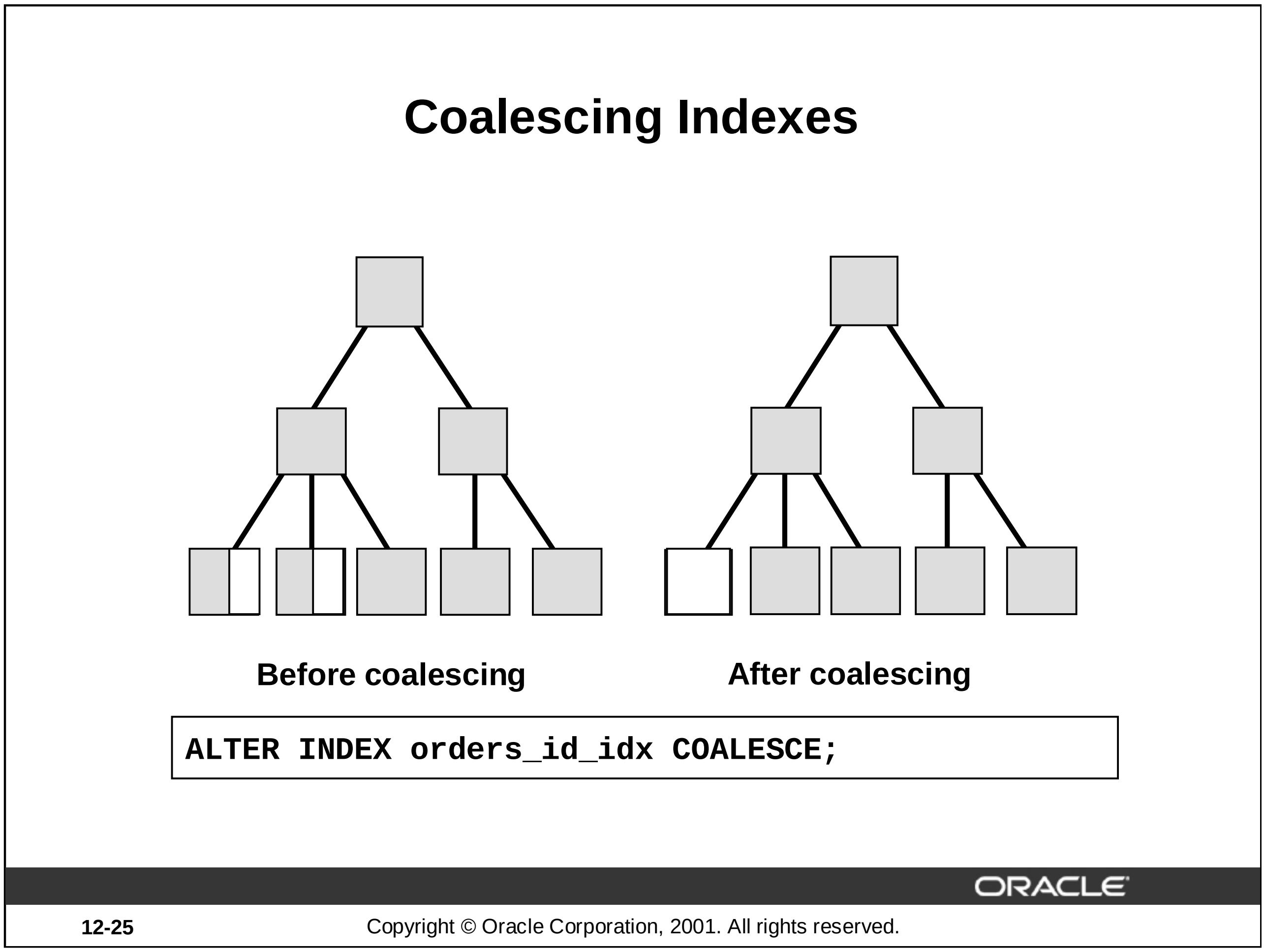
Building or rebuilding an index can be a time-consuming task, especially if the table is very large. Before Oracle8*i*, creating or rebuilding indexes required a lock on the table and prevented concurrent DML operations.

Oracle8*i* offers a method of creating or re-creating an index while allowing concurrent operations on the base table, but performing large DML operations during this procedure is not recommended .

Note: There are still DML locks, which means you cannot perform other DDL operations during an online index build.

Restrictions

- You cannot rebuild an index on a temporary table
- You cannot rebuild an entire partitioned index. You must rebuild each partition or subpartition.
- You cannot also deallocate unused space.
- You cannot change the value of the `PCTFREE` parameter for the index as a whole.



Coalescing Indexes

```
ALTER INDEX orders_id_idx COALESCE;
```

12-25

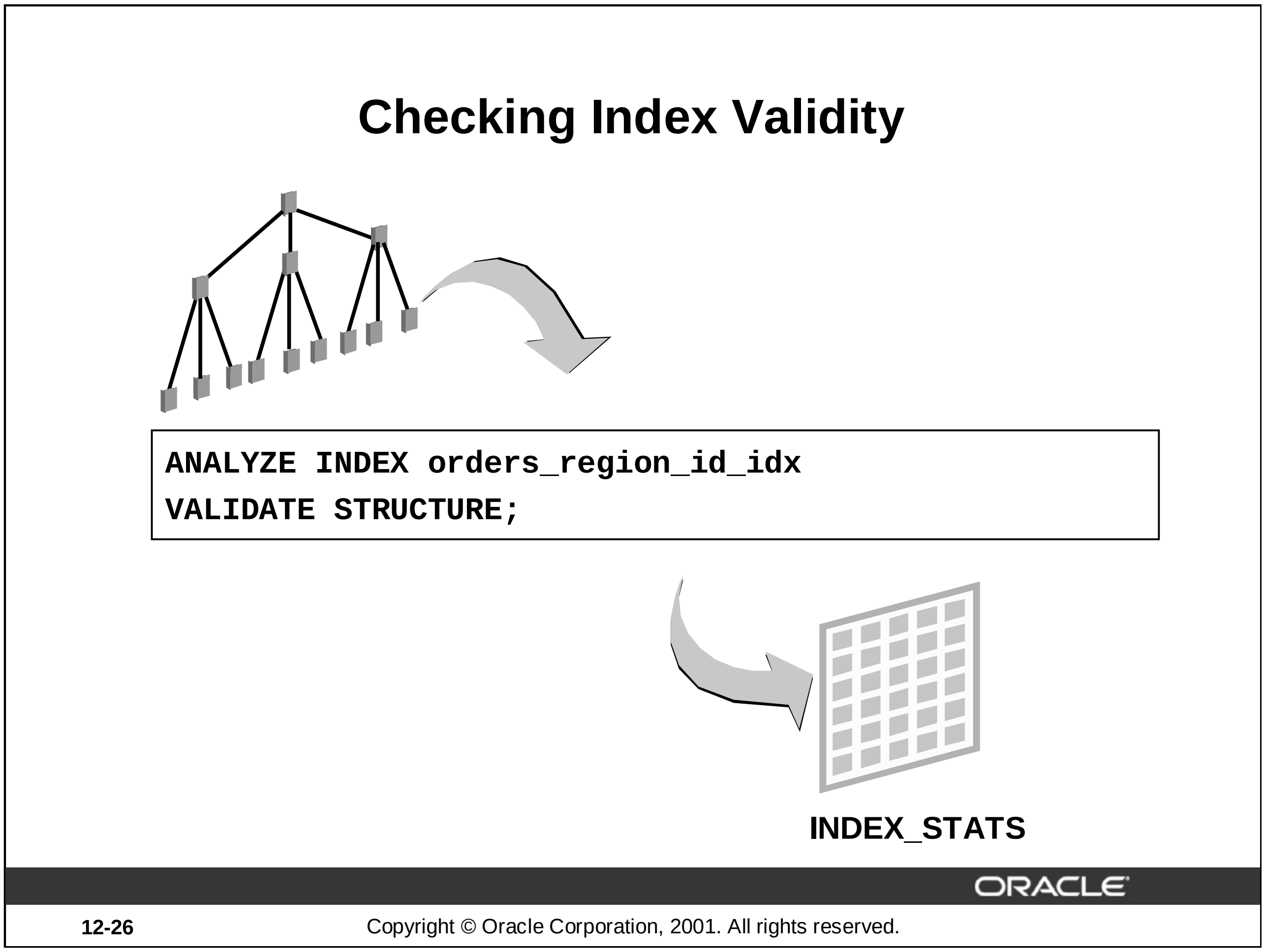
Coalescing Indexes

When you encounter index fragmentation, you can rebuild or coalesce the index. Before you perform either task, you should consider the cost and benefits of each option and choose the one that works best for your situation. Coalesce on an index is a block rebuild that is done online.

In situations where you have B-tree index leaf blocks that can be freed up for reuse, you can merge those leaf blocks using the following SQL statement:

```
ALTER INDEX hr.employees_idx COALESCE;
```

The figure above shows the effect of `ALTER INDEX … COALESCE` on the index hr.employees_idx. Before performing the `COALESCE` operation, the first two leaf blocks are 50% full. This means the index is fragmented and can be coalesced to completely filling the first block, reducing fragmentation.



Checking Index Validity

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

12-26

Analyzing an Index

Analyze the index to perform the following:

- Check all the index blocks for block corruption. Note that this command does not verify whether index entries correspond to data in the table.
- Populate the `INDEX_STATS` view with information about the index.

Syntax

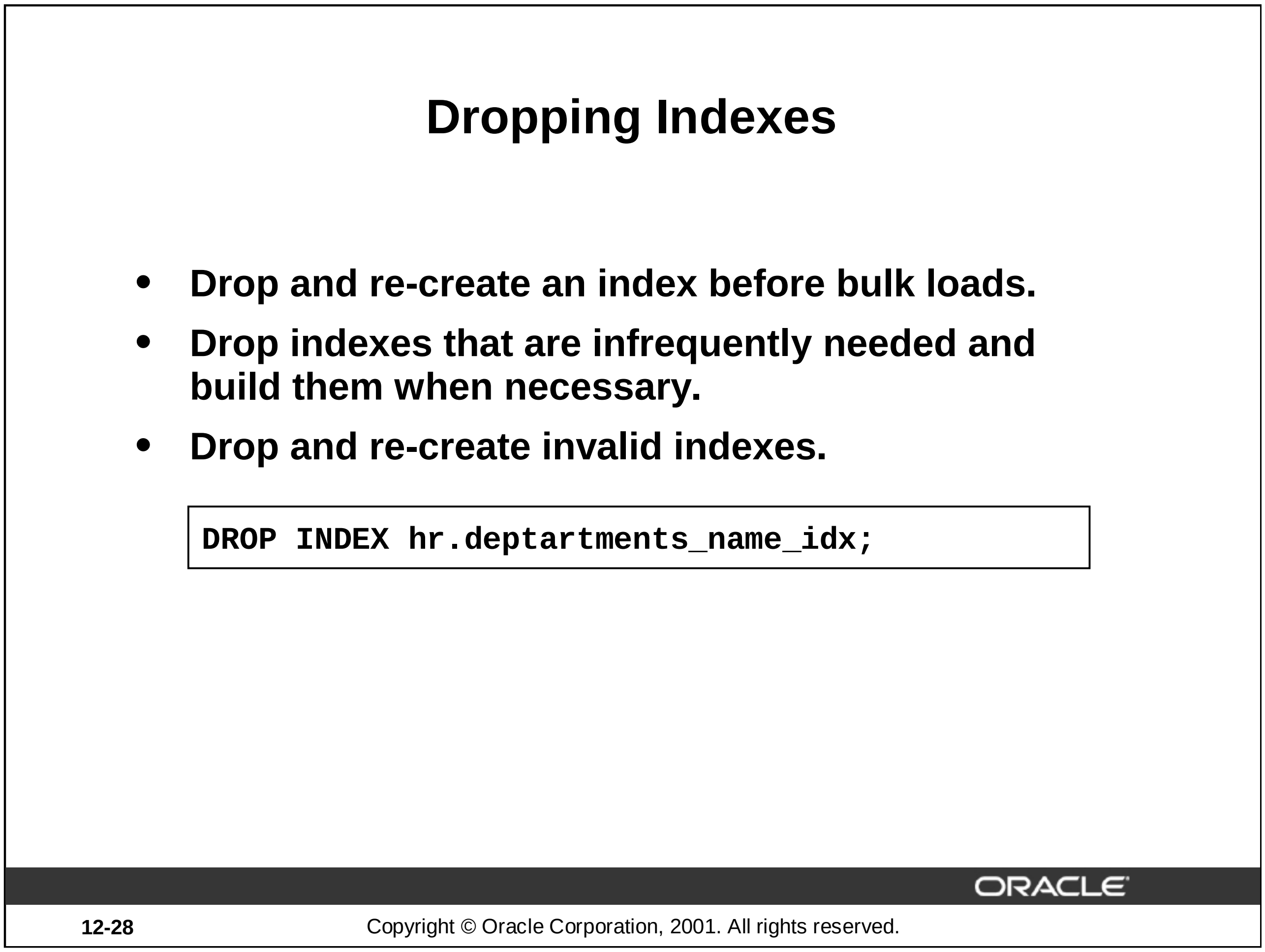
```
ANALYZE INDEX [ schema.]index VALIDATE STRUCTURE
```

After running this command, query `INDEX_STATS` to obtain information about the index as shown in the following example:

Checking Indexes and Their Validity (continued)

```
SELECT blocks, pct_used, distinct_keys
    lf_rows, del_lf_rows
    FROM index_stats;
          BLOCKS     PCT_USED   LF_ROWS    DEL_LF_ROWS
          ------     ---------  --------   ------------
          25         11         14         0
    1 row selected.
```

Reorganize the index if it has a high proportion of deleted rows. For example: when the ratio of DEL_LF_ROWS to LF_ROWS exceeds 30%.



Dropping Indexes

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

```
DROP INDEX hr.deptartments_name_idx;
```

When Should Indexes Be Dropped?

Indexes may need to be dropped in the following scenarios:

- An index that is no longer needed by applications can be removed.
- An index may be dropped prior to performing bulk loads. Dropping an index prior to large data loads and re-creating them after the load:
  - Improves performance of the load
  - Uses index space more efficiently
- Indexes that are used only periodically do not need to be maintained unnecessarily, especially if they are based on volatile tables. This is generally the case in an OLTP system, where ad hoc queries are generated at year-end or quarter-end to gather information for review meetings.
- An index may be marked `INVALID` when there is an instance failure during certain types of operations such as loading. In this case, the index needs to be dropped and re-created.
- The index is corrupt.

Indexes that are required for constraints cannot be dropped, therefore, the dependent constraint must be disabled or dropped first.

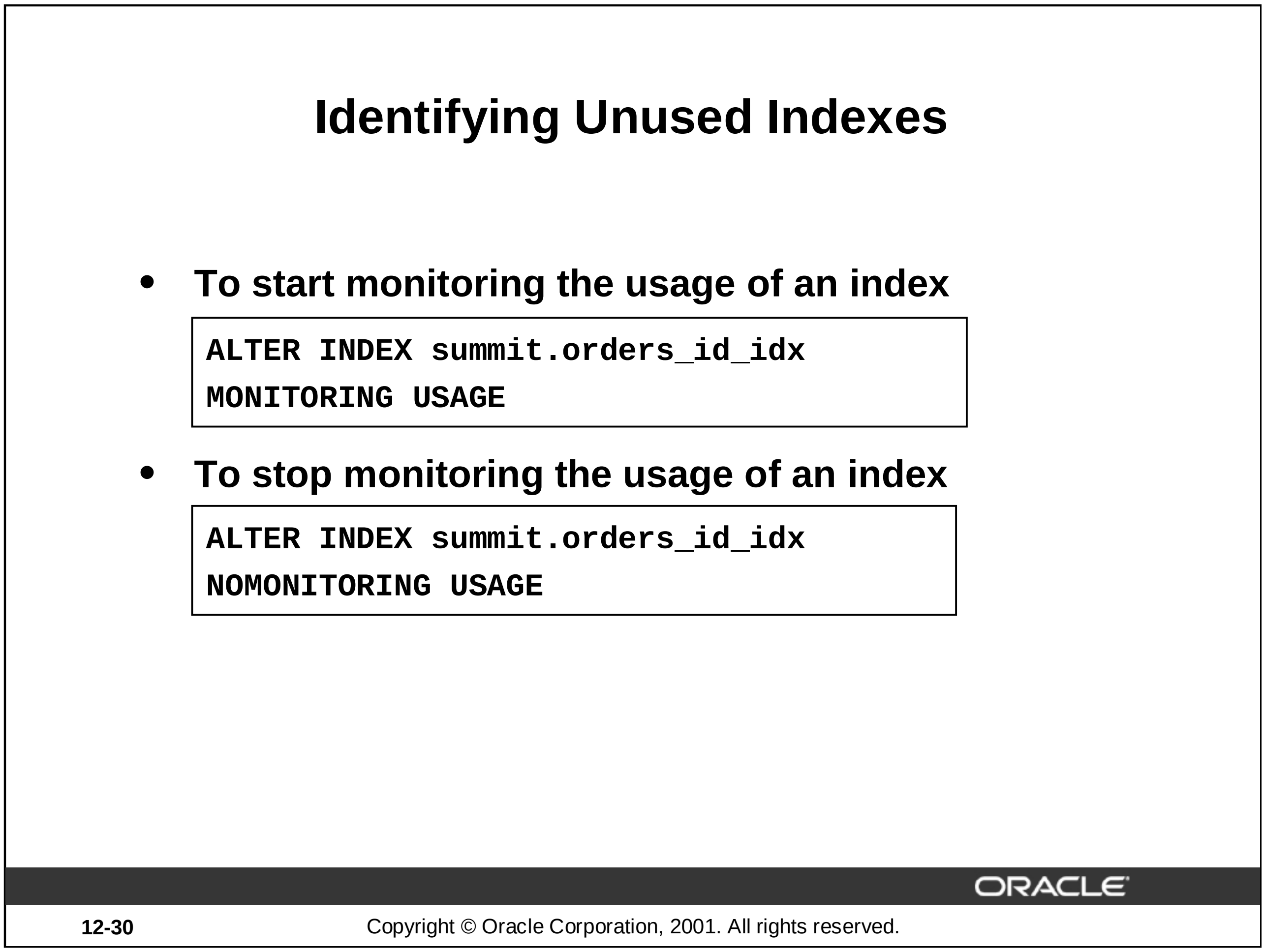
Using Oracle Enterprise Manager to Drop an Index

Launch Schema Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone

   You can also launch the Console from Windows NT Start menu
2. Expand your working database from the databases folder
3. Expand Schema folder and expand the Index folder in the navigator tree
4. Expand the user name (or schema).
5. Select the index.
6. Select Object—>Remove.
7. Select Yes in the dialog box.

Note: An index cannot be dropped if it is used to implement an integrity constraint that is enabled. Constraints are discussed in the lesson “Maintaining Data Integrity.”



Identifying Unused Indexes

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

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

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

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

Identifying Unused Indexes

Beginning with Oracle9*i*, statistics about the usage of an index can be gathered and displayed in V$OBJECT_USAGE. If the information gathered indicates that an index is never used, the index can be dropped. In addition, eliminating unused indexes cuts down on overhead that the Oracle server has to do for DML, thus performance is improved. Each time the MONITORING USAGE clause is specified, V$OBJECT_USAGE will be reset for the specified index. The previous information is cleared or reset, and a new start time is recorded.

V$OBJECT_USAGE Columns

INDEX_NAME: The index name.

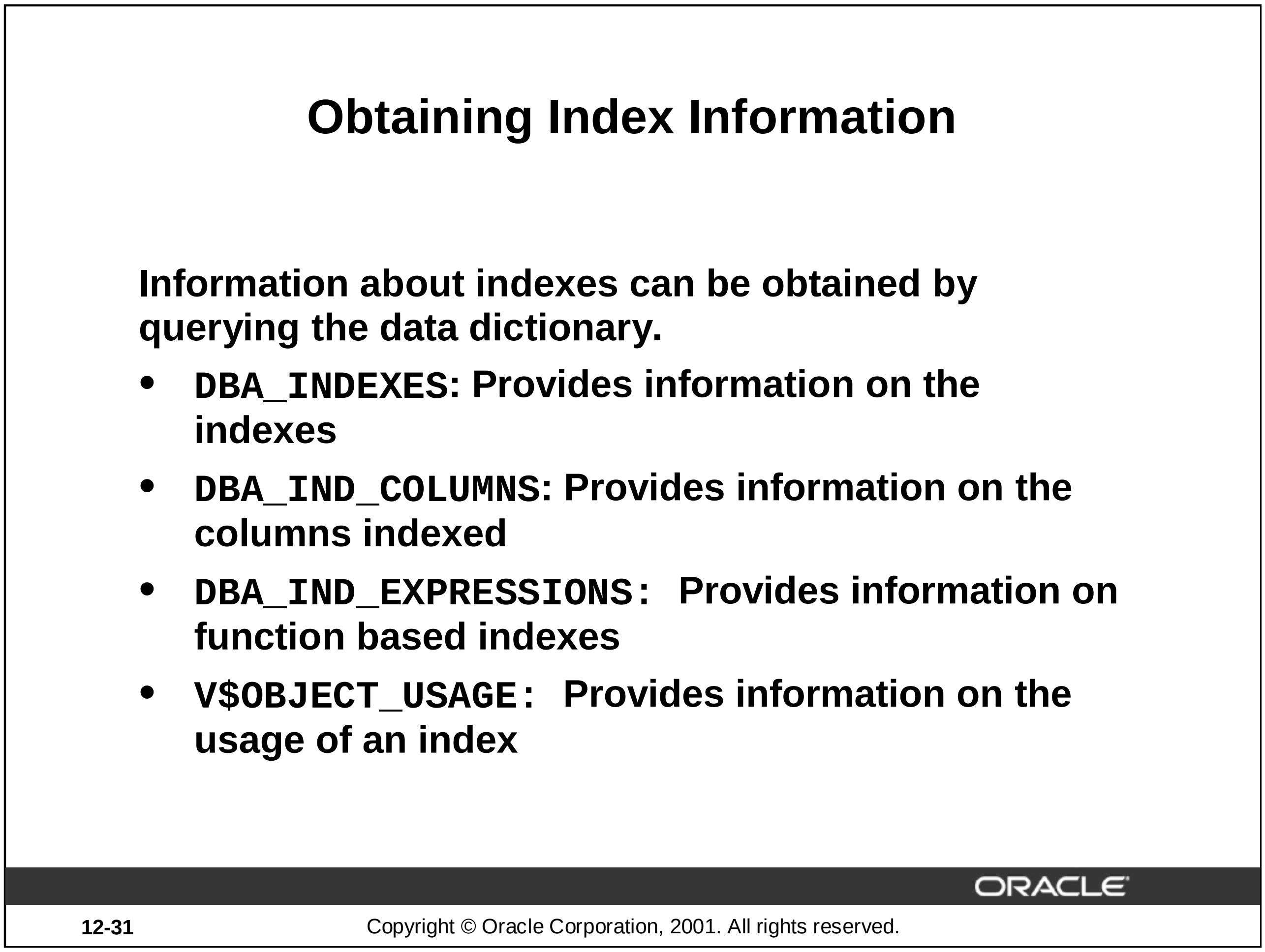
TABLE_NAME: The corresponding table.

MONITORING: Indicates whether monitoring is ON or OFF.

USED: Indicates YES or NO whether index has been used during the monitoring time.

START_MONITORING: Time monitoring began on index.

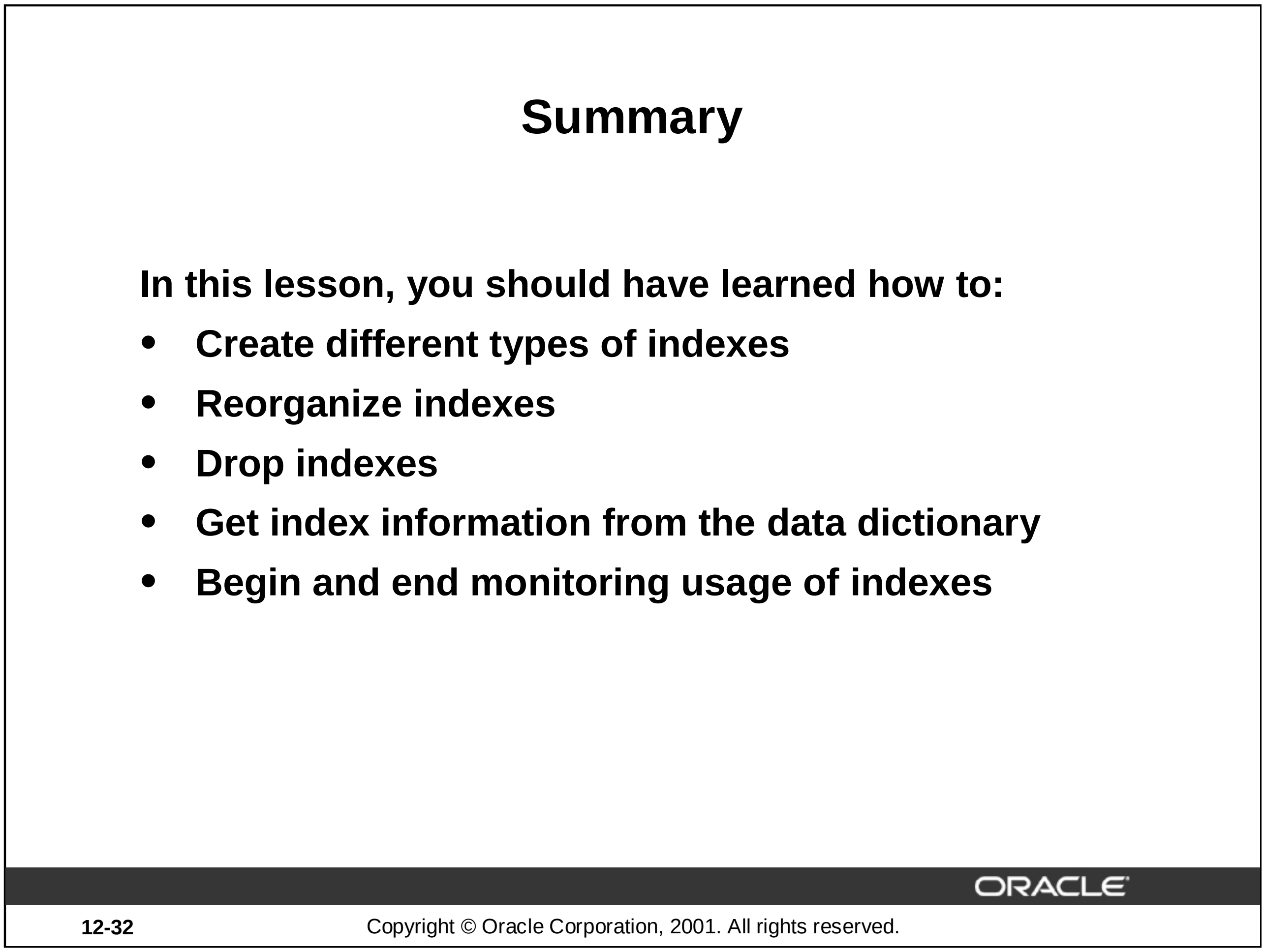
END_MONITORING: Time monitoring stopped on index.



Obtaining Index Information

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

- **`DBA_INDEXES`: Provides information on the indexes**
- **`DBA_IND_COLUMNS`: Provides information on the columns indexed**
- **`DBA_IND_EXPRESSIONS:` Provides information on function based indexes**
- **`V$OBJECT_USAGE:` Provides information on the usage of an index**



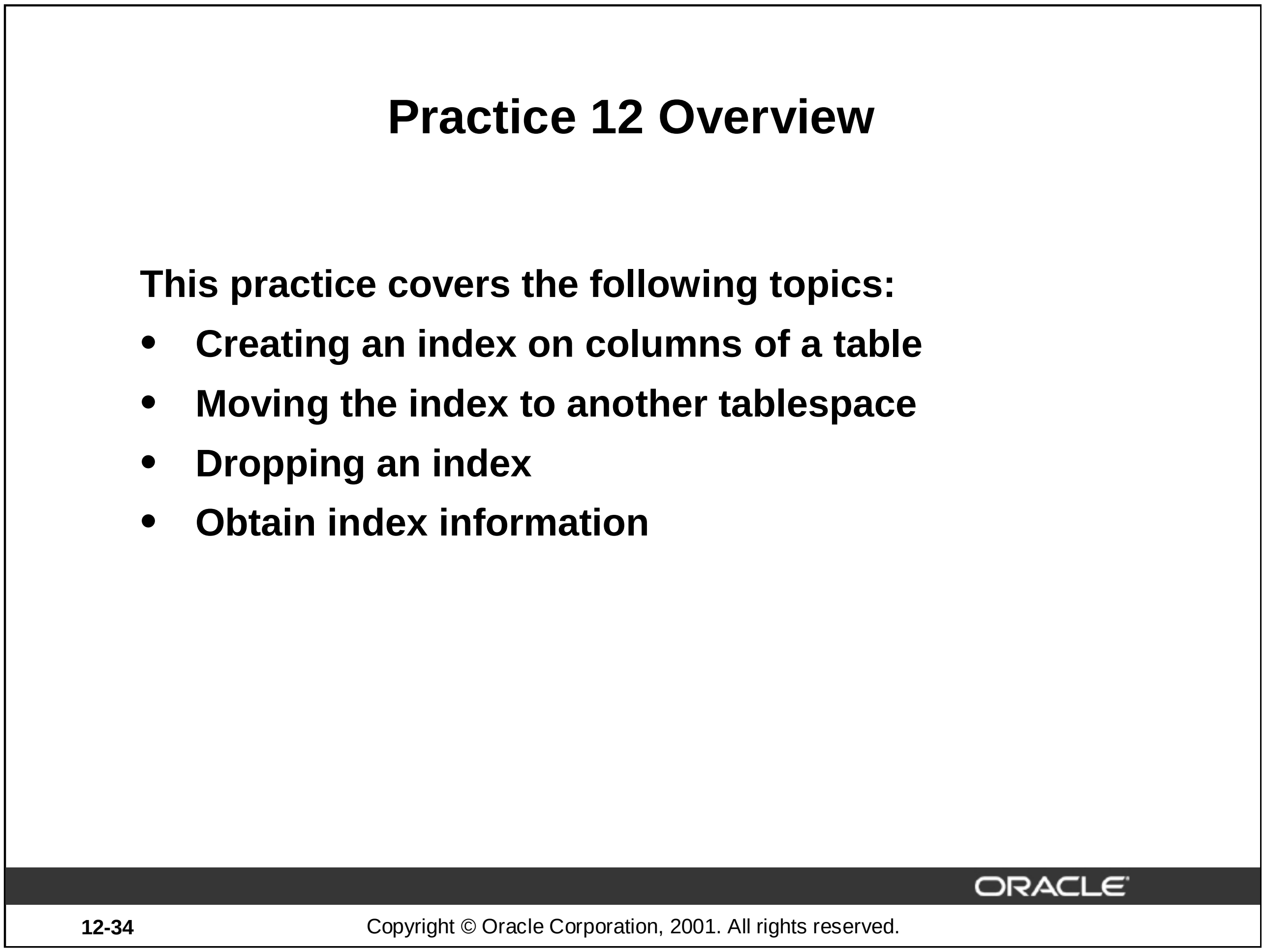
Summary

In this lesson, you should have learned how to:

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

Quick Reference

Context	**Reference**
Initialization parameters	CREATE_BITMAP_AREA_SIZE
Dynamic performance views	None
Data dictionary tables/views	DBA_INDEXES DBA_IND_COLUMNS DBA_OBJECTS IND$ INDEX_STATS
Commands	CREATE INDEX CREATE UNIQUE INDEX CREATE BITMAP INDEX CREATE INDEX ... REVERSE ALTER INDEX ... STORAGE ALTER INDEX ... INITRANS ... MAXTRANS ALTER INDEX ... ALLOCATE EXTENT ALTER INDEX ... DEALLOCATE UNUSED ALTER INDEX .... REBUILD ALTER INDEX .... REBUILD ... REVERSE ALTER INDEX .... REBUILD ... NOREVERSE ANALYZE INDEX ... VALIDATE STRUCTURE DROP INDEX
Packaged procedures and functions	None



Practice 12 Overview

This practice covers the following topics:

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

Practice 12: Managing Indexes

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 the indexes, naming them CUST_NAME_IDX and CUST_REGION_IDX, respectively, and placing them in the appropriate tablespaces.

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.

2 Move the CUST_REGION_IDX index to another tablespace.

Hint: The index can be rebuilt specifying a different tablespace.

3 Note the files and blocks used by the extents by CUST_REGION_IDX index.

Hint: Use the view DBA_EXTENTS to get this information.

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.

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.

5 a As user SYSTEM, run the script lab12_05a.sql to create and populate the NUMBERS table.

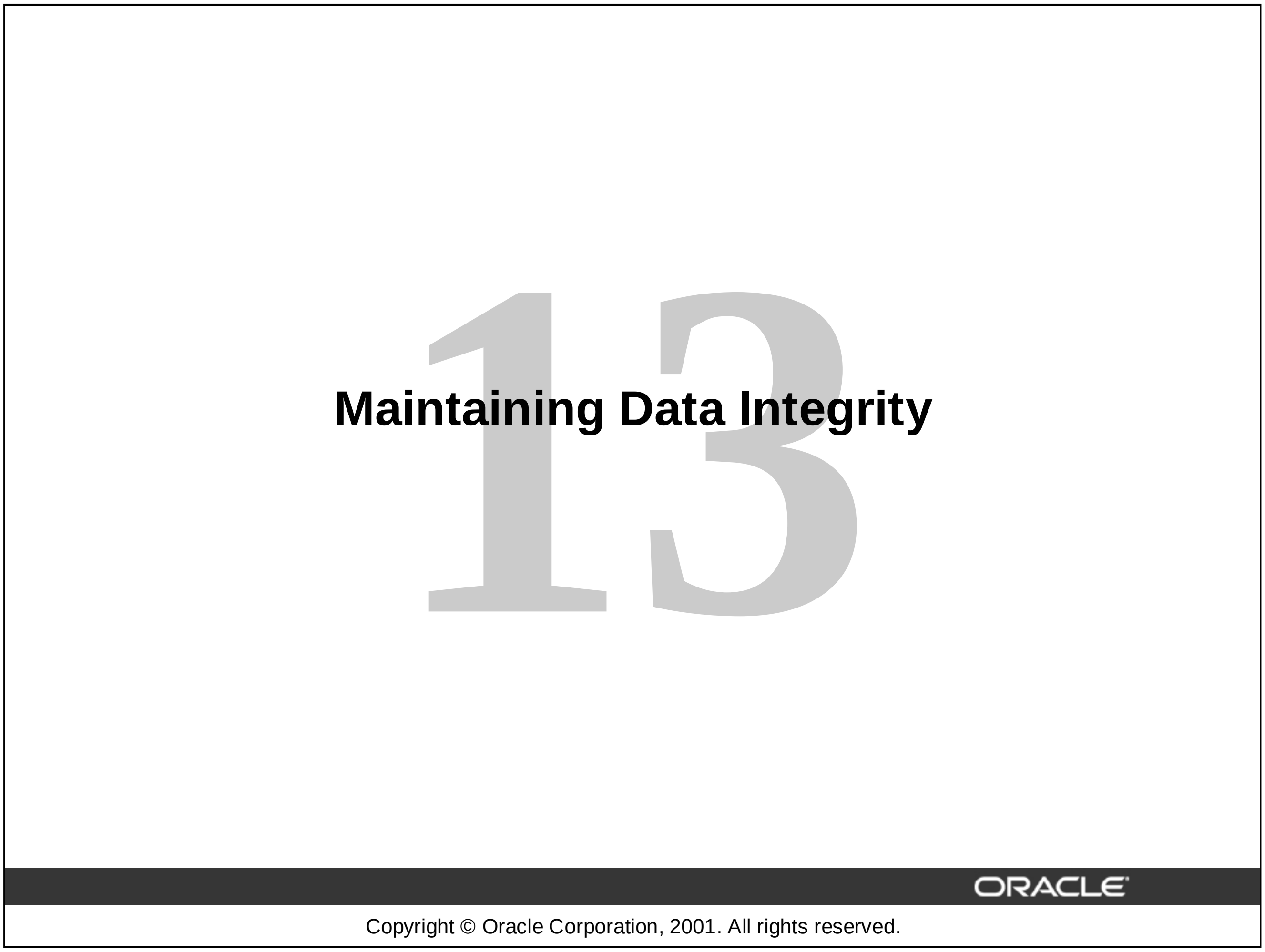
b Query the table NUMBERS to find the number of distinct values in the two columns in the table.

c Create B-tree indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively, and check the total sizes of the indexes. Put the indexes in tablespace INDX01.

Hint: Check the total blocks allocated to the extents from DBA_SEGMENTS.

d Create bitmap indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively, and check the total sizes of the indexes. Put the indexes in tablespace INDX01.
What can you conclude about the relationship between cardinality and sizes of the two types of indexes?

Hint: The existing indexes need to be dropped before creating the new indexes. Now re-execute the query to check the sizes of the indexes.



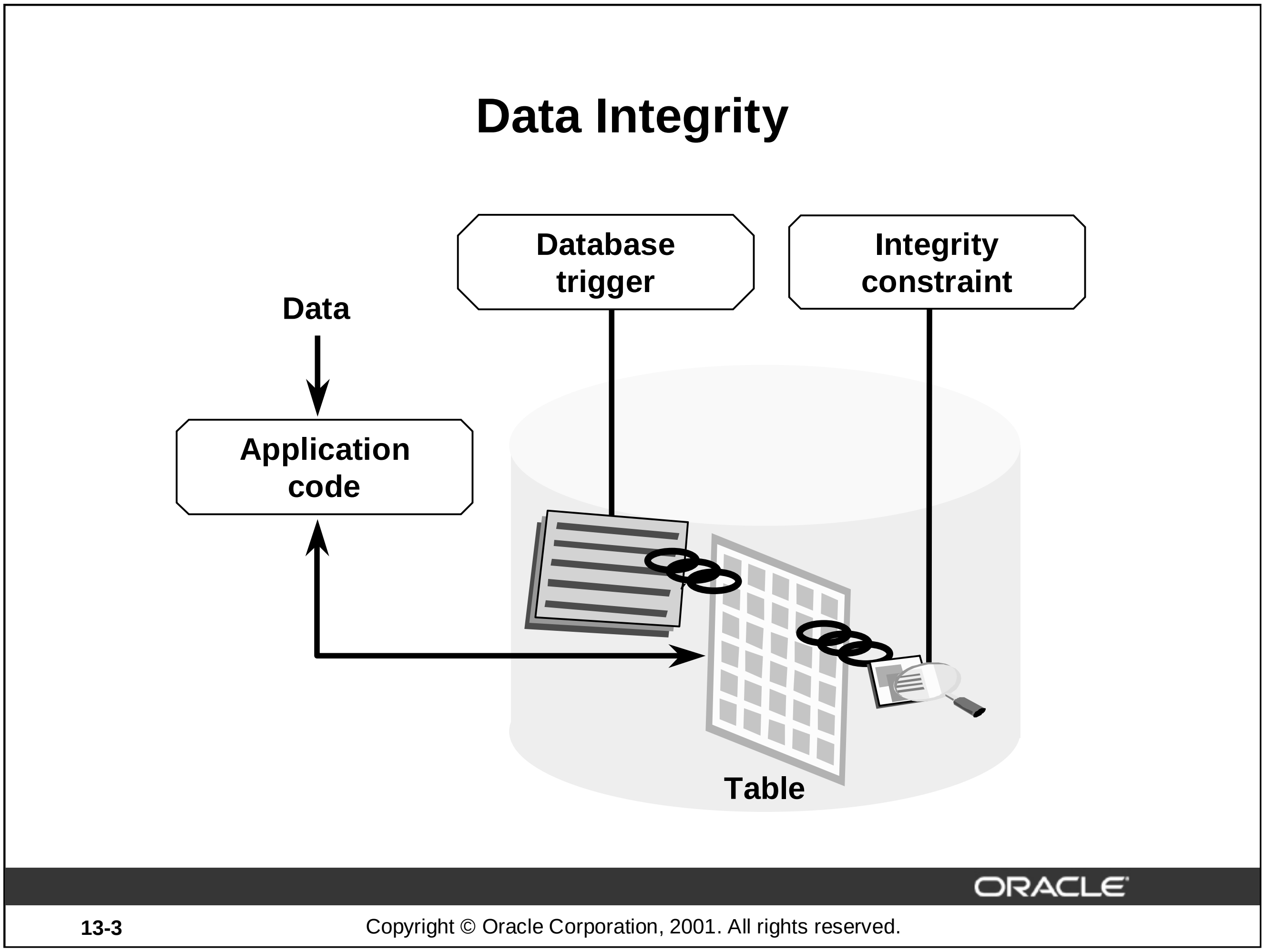
13

Maintaining Data Integrity

Objectives

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

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



Data Integrity

Methods to Guarantee Data Integrity

Data integrity means that data in a database adheres to business rules. There are three primary ways in which data integrity can be maintained:

- Application code
- Database triggers
- Declarative integrity constraints

Mapping the business rules using one of the three methods is a design decision made by the designer. The database administrator is primarily concerned with implementing the methods chosen by the designer and balancing the performance needs against integrity requirements.

Application code can be implemented either as stored procedures within the database or as applications running on the client. This lesson focuses on the use of integrity constraints.

Methods to Guarantee Data Integrity (continued)

Database Triggers

Database triggers are PL/SQL programs that are executed when a certain event such as an insert or an update of a column occurs on a table. Triggers can be enabled or disabled—that is, they can be set to execute when the event occurs, or they can be set not to execute even though they are defined. Database triggers are usually created only to enforce a complex business rule that cannot be defined as an integrity constraint.

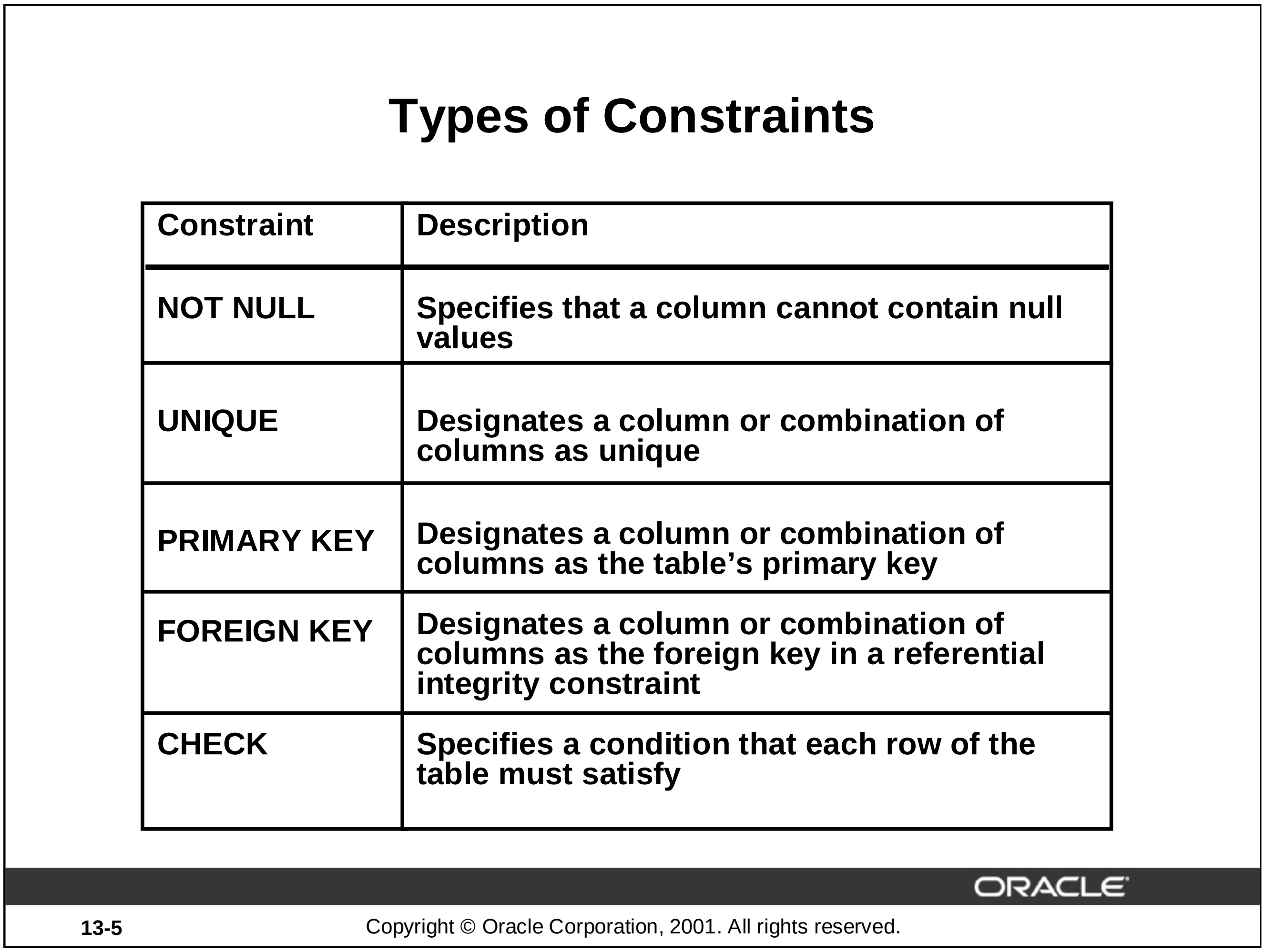
Note: Database triggers are covered in other Oracle courses.

Integrity Constraints

Integrity constraints are the preferred mechanism for enforcing business rules because they:

- Provide improved performance
- Are easy to declare and modify—they do not require extensive coding
- Centralize rules
- Are flexible (enabled or disabled)
- Are fully documented in the data dictionary

The following sections explain the behavior of integrity constraints and discuss how they are implemented by the Oracle server.



Types of Constraints

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

13-5

Types of Constraints

By default, all columns in a table allow nulls. Null means absence of a value. A `NOT NULL` constraint requires a column of a table to contain values.

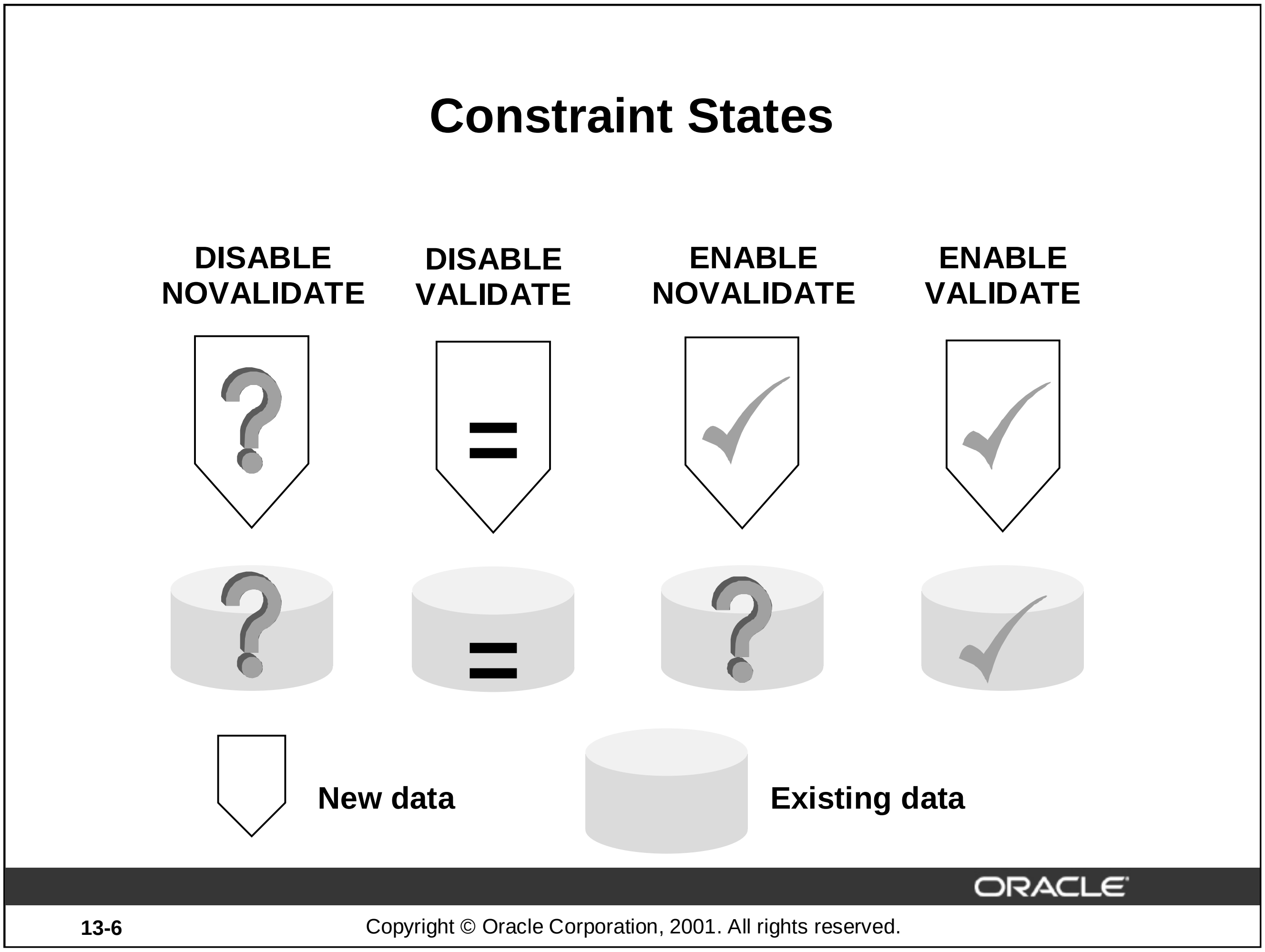
A `UNIQUE` key constraint requires that every value in a column or set of columns (key) be unique. No two rows of a table can have duplicate values in a specified column or set of columns.

Each table in the database can have at most one `PRIMARY KEY` constraint. A `PRIMARY KEY` constraint ensures that both of the following are true:

- No two rows of a table have duplicate values in the specified column.
- Primary key columns do not contain nulls.

A `CHECK` integrity constraint on a column or a set of columns requires that a specified condition be true or unknown for every row of the table.

Although the `NOT NULL` and `CHECK` constraints do not directly require DBA attention, the primary key, unique, and foreign key constraints must be managed to ensure high availability and acceptable performance levels.



Constraint States

DISABLE NOVALIDATE

DISABLE VALIDATE

ENABLE NOVALIDATE

ENABLE VALIDATE

New data

Existing data

13-6

Constraint States

An integrity constraint can be enabled (ENABLE) or disabled (DISABLE). If a constraint is enabled, data is checked as it is entered or updated in the database. Data that does not confirm to the constraint’s rule is prevented from being entered. If a constraint is disabled, then data that does not confirm can be entered into the database. An integrity constraint can be in one of the following states:

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

Disable Novalidate A constraint that is disable novalidate is not checked. Data in the table, as well as new data that is entered or updated, may not conform to the rules defined by the constraint.

Disable Validate If a constraint is in this state, then any modification of the constrained columns is not allowed. In addition, the index on the constraint is dropped and the constraint is disabled.

Constraint States (continued)

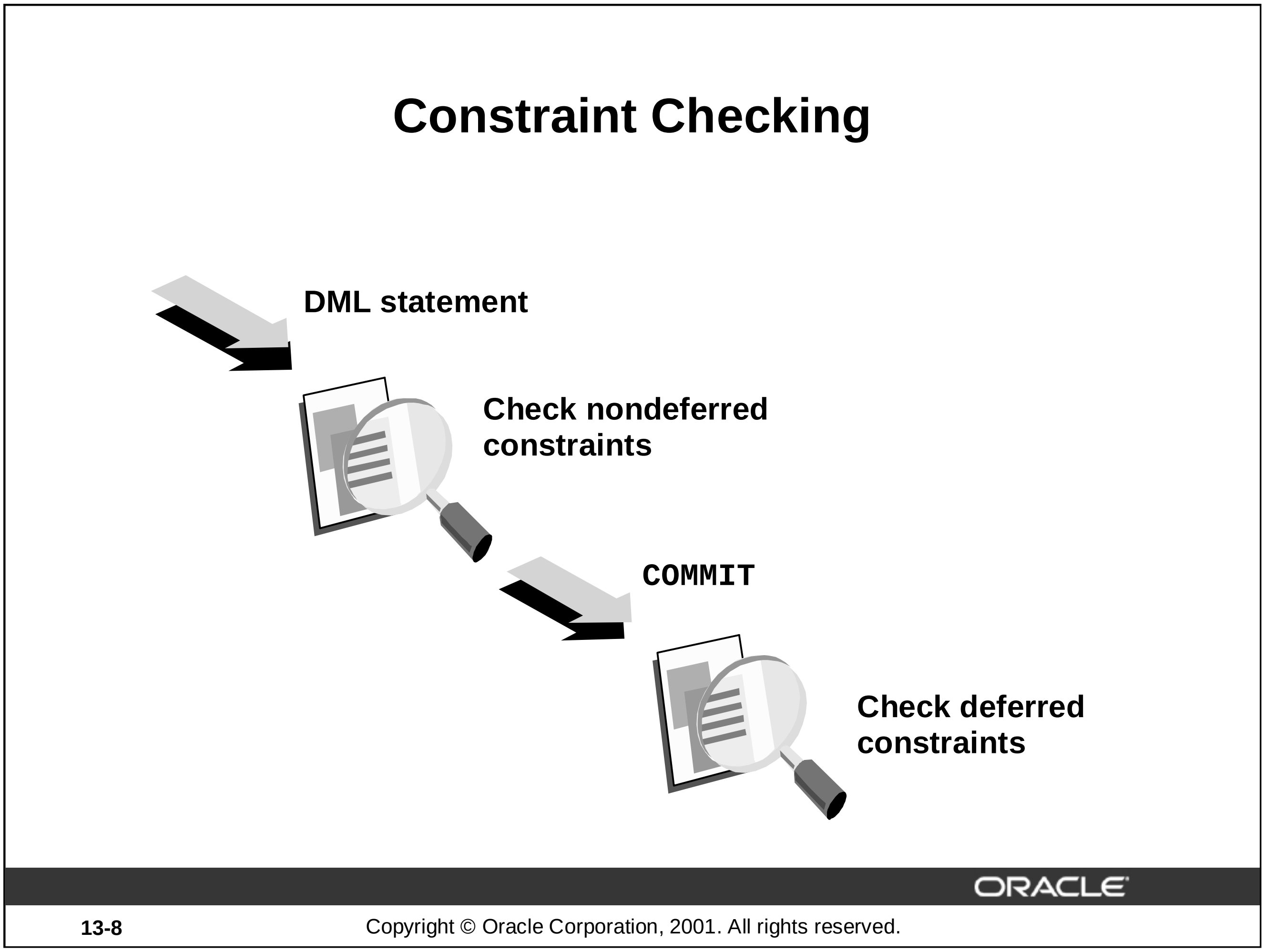
Enable Novalidate If a constraint is in this state, new data that violates the constraint cannot be entered. However, the table may contain data that is invalid—that is, data that violates the constraint. Enabling constraints in the novalidated state is most useful in data warehouse configurations that are uploading valid OLTP data.

Enable Validate If a constraint is in this state, no row violating the constraint can be inserted into the table. However, while the constraint is disabled such a row can be inserted. This row is known as an exception to the constraint. If the constraint is in the enable novalidate state, violations resulting from data entered while the constraint was disabled remain. The rows that violate the constraint must be either updated or deleted in order for the constraint to be put in the validated state.

When a constraint changes to enable validate from a disabled state, the table is locked and all data in the table is checked for conformity. This may cause DML operations such as a data load to wait, so it is advisable to move first from a disabled state to enable novalidate, and then to enable validate.

Transitions between these states are governed by the following rules:

- `ENABLE` implies `VALIDATE`, unless `NOVALIDATE` is specified.
- `DISABLE` implies `NOVALIDATE`, unless `VALIDATE` is specified.
- `VALIDATE` and `NOVALIDATE` do not have default implications for the `ENABLE` and `DISABLE` states.
- When a unique or primary key moves from the `DISABLE` state to the `ENABLE` state and there is no existing index, a unique index is created automatically. Similarly, when a unique or primary key moves from `ENABLE` to `DISABLE` and it is enabled with a unique index, the unique index is dropped.
- When any constraint is moved from the `NOVALIDATE` state to the `VALIDATE` state, all data must be checked. However, moving from `VALIDATE` to `NOVALIDATE` simply forgets that the data was ever checked.
- Moving a single constraint from the `ENABLE NOVALIDATE` state to the `ENABLE VALIDATE` state does not block reads, writes, or other DDL statements.



Constraint Checking

DML statement

Check nondeferred constraints

COMMIT

Check deferred constraints

13-8

Constraint Checking

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

Nondeferred or Immediate Constraints

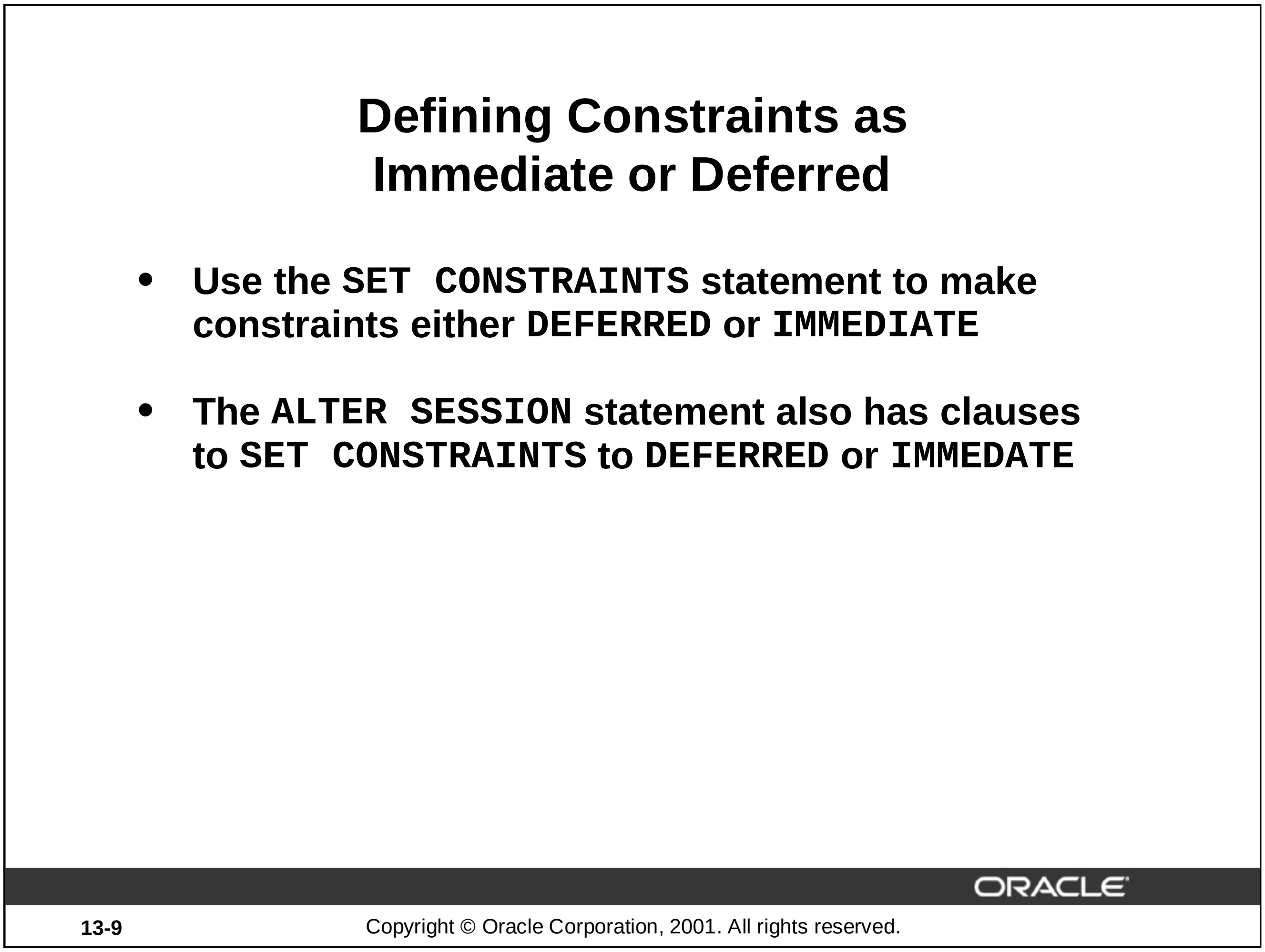
Nondeferred constraints, also known as immediate constraints, are enforced at the end of every DML statement. A constraint violation causes the statement to be rolled back. If a constraint causes an action such as `delete cascade`, the action is taken as part of the statement that caused it. A constraint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

Deferred Constraints

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

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

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



Defining Constraints as Immediate or Deferred

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

Changing the Enforcement of Constraints

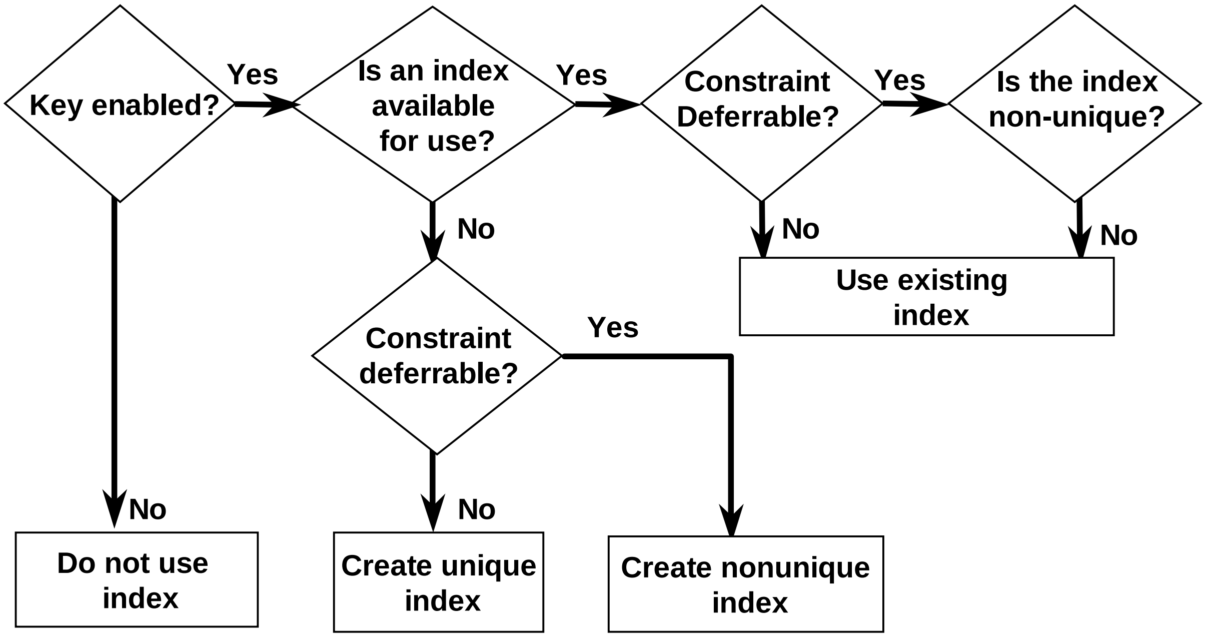
The `SET CONSTRAINTS` statement makes constraints either `DEFERRED` or `IMMEDIATE` for a particular transaction. You can use this statement to set the mode for a list of constraint names or for constraints. The `SET CONSTRAINTS` mode lasts for the duration of the transaction or until another `SET CONSTRAINTS` statement resets the mode. The `SET CONSTRAINTS` statement is disallowed inside triggers.

The `ALTER SESSION` statement also has clauses to `SET CONSTRAINTS` to `IMMEDIATE` or `DEFERRED`. These clauses imply setting ALL deferrable constraints (list of constraint names cannot be specified). The `ALTER SESSION SET CONSTRAINTS` statement applies to a current session only.

```
ALTER SESSION
    SET CONSTRAINT[S] =
    {IMMEDIATE|DEFERRED|DEFAULT}


SET CONSTRAINT | CONSTRAINTS
    {constraint |ALL }
    {IMMEDIATE|DEFERRED}
```

Primary and Unique Key Enforcement

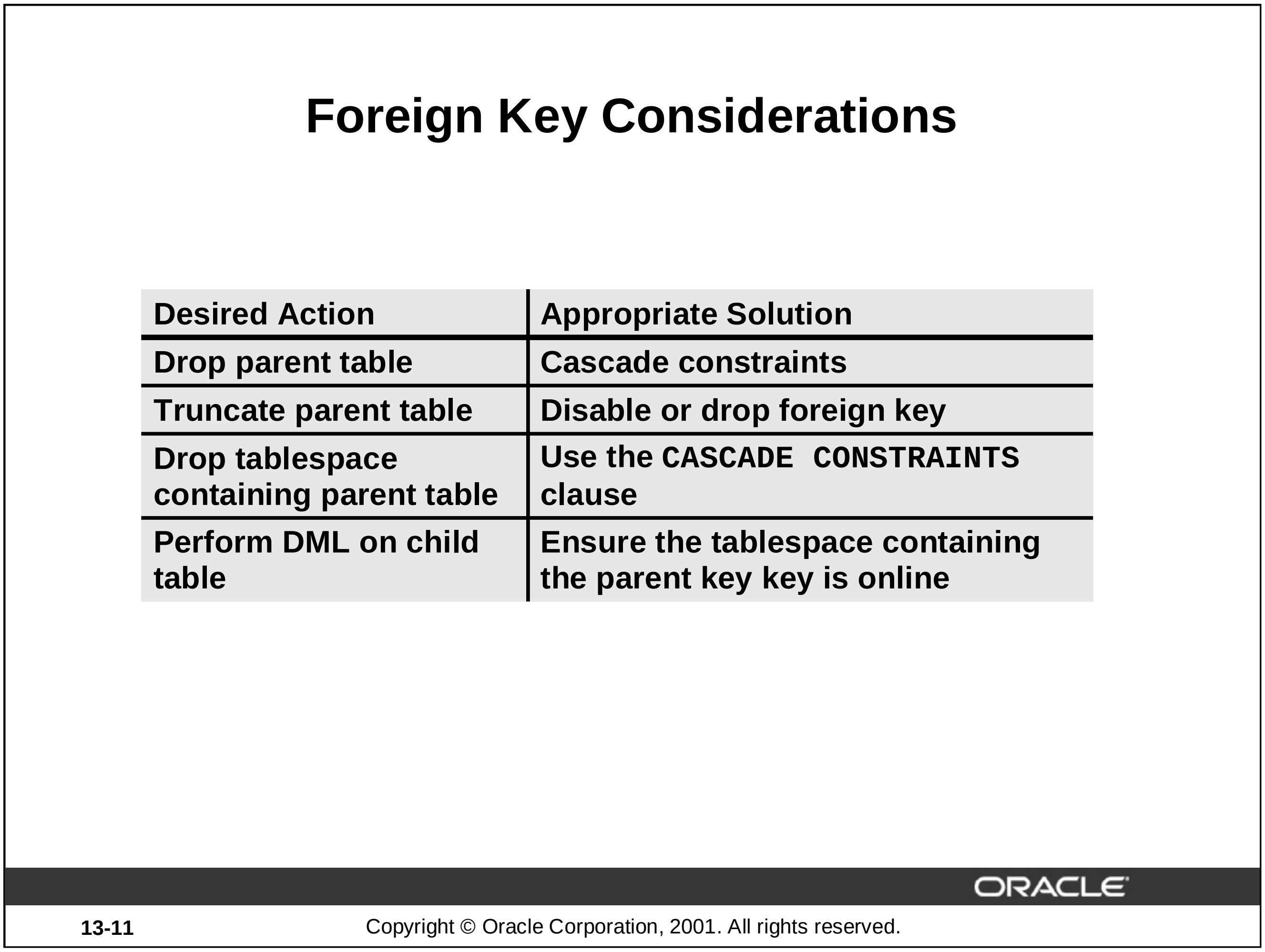


How to Enforce Primary and Unique Key Constraints

Primary and unique keys are enforced using indexes. You can control the location and type of index that is used for enforcing these constraints.

The Oracle server uses the following procedure to implement unique and primary key constraints:

- If the constraint is disabled, no indexes are needed.
- If the constraint is enabled and the columns in the constraint form the leading part of an index, the index is used to enforce the constraint whether the index itself was created as unique or non-unique.
- If the constraint is enabled and there is no index that uses the constraint columns as a leading part of the index, then an index with the same name as the constraint is created using the following rules:
  - If the key is deferrable, a non-unique index on the key column is created.
  - If the key is non-deferrable, a unique index is created.
- If an index is available for use and constraint is non-deferrable, use existing index. If the constraint is deferrable and the index is non-unique, use existing index.



Foreign Key Considerations

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

13-11

Foreign Key Considerations

You need to consider several factors in maintaining tables that are in a foreign key relationship.

DDL Involving Parent Table

The foreign key must be dropped before dropping the parent table. Use the following command to perform both actions using a single statement:

```
DROP TABLE table CASCADE CONSTRAINTS
```

The parent table cannot be truncated without dropping or disabling the foreign key.

The foreign key must be dropped before the tablespace containing the parent is dropped. The following command can be used to achieve this:

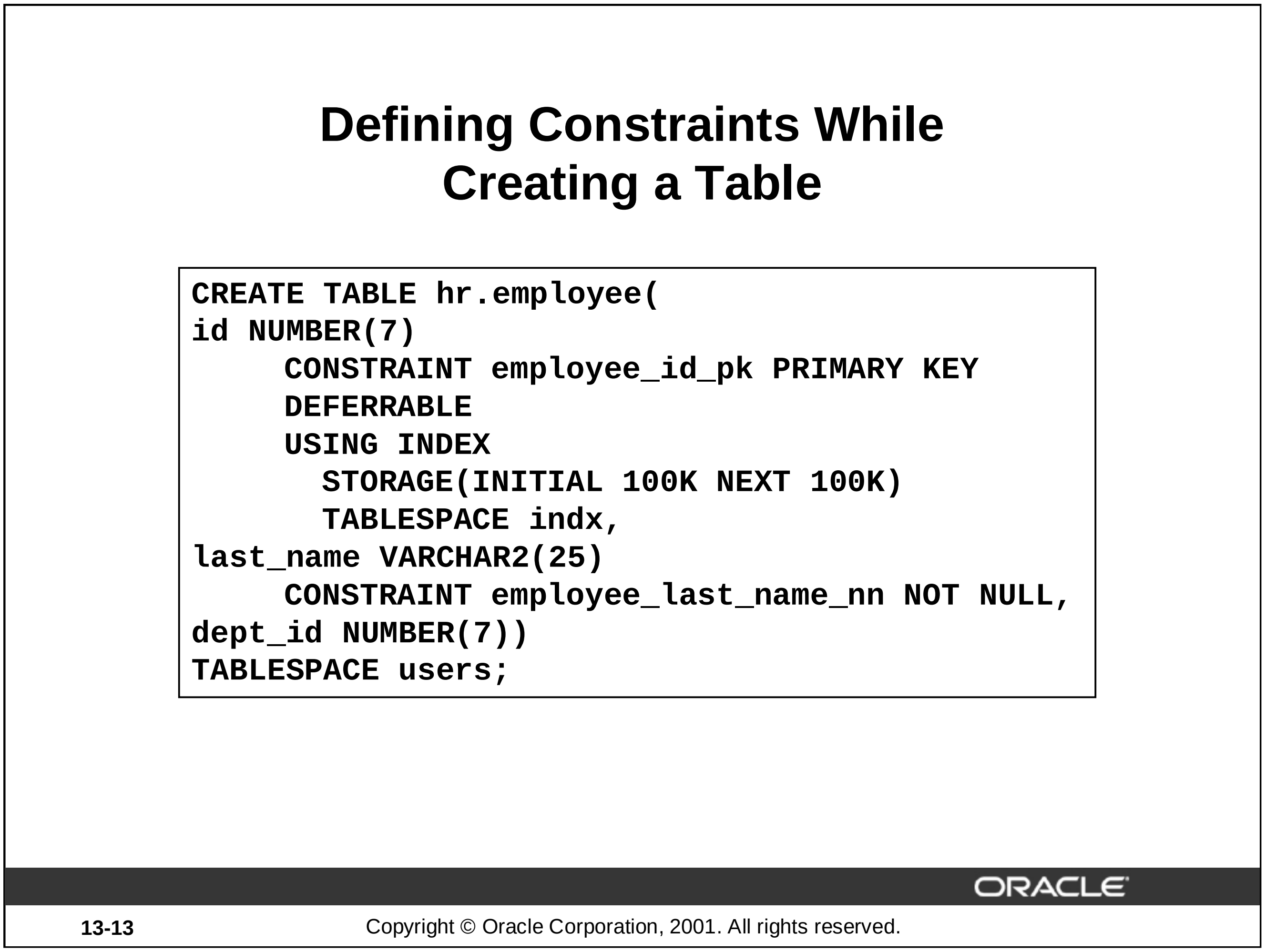
```
DROP TABLESPACE tablespace INCLUDING CONTENTS
CASCADE CONSTRAINTS
```

Foreign Key Considerations (continued)

If the `DELETE CASCADE` option is not used when rows are deleted from the parent table, the Oracle server needs to ensure that there are no rows in the child table with the corresponding foreign key. Similarly, an update to the parent key is permitted only when there are no child rows with the old key value. If there is no index on the foreign key on the child table, the Oracle server locks the child table and prevents changes to ensure referential integrity. If there is an index on the table, the referential integrity is maintained by locking the index entries and avoiding more restrictive locks on the child table. If both tables need to be updated concurrently from different transactions, create an index on the foreign key columns.

When data is inserted into or the foreign key column is updated in the child table, the Oracle server checks the index on the parent table that is used for enforcing the referenced key. Therefore, the operation succeeds only if the tablespace containing the index is online. Note that the tablespace containing the parent table does not need to be online to perform DML operations on the child table.

Oracle9i no longer requires a share lock on unindexed foreign keys when doing an update or delete on the primary key. It still obtains the table-level share lock, but then releases it immediately after obtaining it. If multiple primary keys are update or deleted, the lock is obtained and released once per row.



Defining Constraints While Creating a Table

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

Defining Constraints While Creating a Table

A constraint can be defined either when a table is created or when a table is altered. Use the `constraint_clause` clause in a `CREATE TABLE` or `ALTER TABLE` statement to define a constraint. You must have the requisite privileges to define an integrity constraint. To create a referential integrity constraint, the parent table must be in your own schema, or you must have the `REFERENCES` privilege on the columns of the referenced key in the parent table.

Defining Constraints While Creating a Table (continued)

The `column_constraint` syntax is part of the table definition. At the time the table is created, the constraint can be defined using the following syntax:

```
column datatype [CONSTRAINT constraint]
              {[NOT] NULL
              |UNIQUE      [USING INDEX index_clause]
              |PRIMARY KEY [USING INDEX index_clause]
              |REFERENCES  [schema.]table [(column)]
                   [ON DELETE CASCADE]
              |CHECK       (condition)
              }
              constraint_state :==
              [NOT DEFERRABLE|DEFERRABLE [INITIALLY
    {IMMEDIATE|DEFERRED}]
              ]
              [DISABLE|ENABLE [VALIDATE|NOVALIDATE]]
```

where: `CONSTRAINT` identifies the integrity constraint by the name *`constraint`* stored in data dictionary

`USING INDEX` specifies that the parameters defined in the *`index-clause`* should be used for the index the Oracle server uses to enforce a unique or primary key constraint (The name of the index is the same as the name of the constraint.)

`DEFERRABLE` indicates that constraint checking can be deferred until the end of the transaction by using the `SET CONSTRAINT` command

`NOT DEFERRABLE` indicates that this constraint is checked at the end of each DML statement
(A `NOT DEFERRABLE` constraint cannot be deferred by sessions or transactions. `NOT DEFERRABLE` is the default.)

Defining Constraints While Creating a Table (continued)

`INITIALLY IMMEDIATE` indicates that at the start of every transaction, the default is to check this constraint at the end of every DML statement (If no `INITIALLY` clause is specified, `INITIALLY IMMEDIATE` is the default.)

`INITIALLY DEFERRED` implies that this constraint is `DEFERRABLE` and specifies that, by default, the constraint is checked only at the end of each transaction

`DISABLE` disables the integrity constraint
(If an integrity constraint is disabled, the Oracle server does not enforce it.)

Defining Constraints While Creating a Table (continued)

Using Enterprise Manager to define Constraints

1. Launch the Console:
   %oemapp console
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Right-click on the your working database and click Connect.
5. Supply the username, password and service name for your working database and click OK.
6. Expand the Schema folder.
7. Select the Table folder and select Create from the Object menu
8. Select Table from the list and click Create
9. Supply values for the table name, schema owner, tablespace and define the columns for the table
10. Click the Constraints tab and define integrity constraints and click Create

Defining Constraints While Creating a Table (continued)

Table Constraint

A table constraint is part of the table definition. This can define on any type of constraint except a NOT NULL constraint. This is defined using the following syntax:

```
[CONSTRAINT constraint]
{PRIMARY KEY (column [, column ]... )
             [USING INDEX index_clause]
|UNIQUE      (column [, column ]... )
             [USING INDEX index_clause]
|FOREIGN KEY (column [, column ]... )
 REFERENCES  [schema.]table [(column [, column ]... )]
             [ON DELETE CASCADE]
|CHECK       (condition)
}
[constraint_state]
```

Note

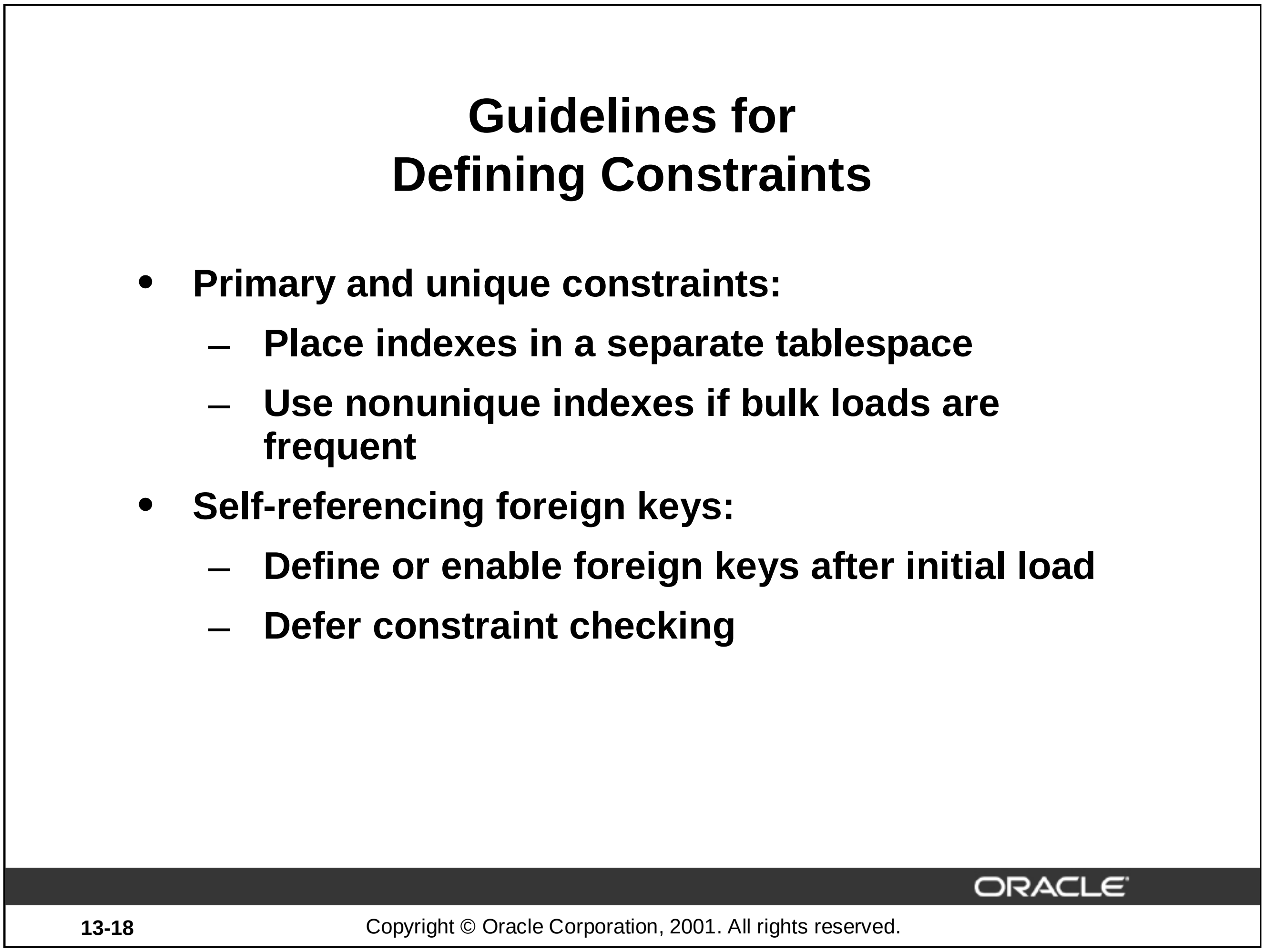
- It is a good practice to adopt a standard naming convention for constraints. This is especially true with `CHECK` constraints because the same constraint can be created several times with different names.
- Table constraints are needed in the following cases:
  - When a constraint names two or more columns
  - When a table is altered to add any constraint other than the NOT NULL constraint
- Defining a constraint from the type NOT NULL after creating a table is only possible with:

  ```
  ALTER TABLE table MODIFY column CONSTRAINT constraint NOT
  NULL;
  ```

Defining Constraints After Creating a Table: Example

```
ALTER TABLE hr.employee
   ADD(CONSTRAINT employee_dept_id_fk FOREIGN KEY(dept_id)
   REFERENCES hr.department(id)
   DEFERRABLE INITIALLY DEFERRED);
```

Note: The `EXCEPTIONS` clause, discussed under “Enabling Constraints” later in this lesson, can be used to identify rows violating a constraint that is added using the `ALTER TABLE` command.



Guidelines for Defining Constraints

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

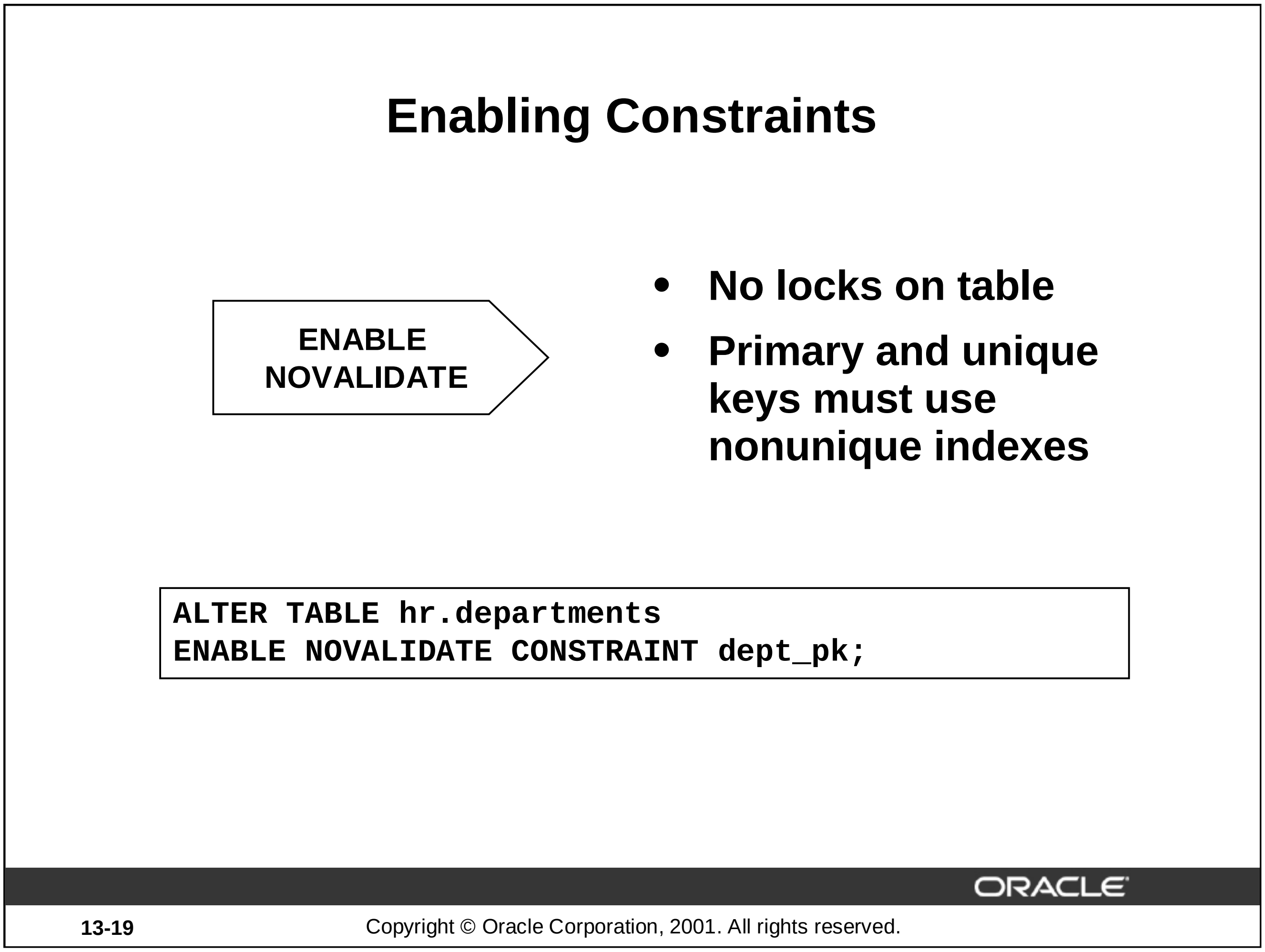
13-18

Guidelines for Defining Constraints

The following guidelines are useful when defining constraints:

- Place indexes used for enforcing primary key and unique constraints in a tablespace different from that of the table. This can be done either by specifying the `USING INDEX` clause or by creating the table, creating the index, and altering the table to add or enable the constraint.
- If data is frequently loaded in bulk into a table, it is preferable to disable the constraints, perform the load, and then enable the constraints. If a unique index is used for enforcing a primary key or unique constraint, this index needs to be dropped when the constraint is disabled. Performance can be enhanced by using a nonunique index for enforcement of primary key or unique constraints in such situations: either create the key as deferrable or create the index before defining or enabling the key.
- If a table contains a self-referencing foreign key, use one of the following methods to load data:
  - Define or enable the foreign key after the initial load
  - Define the constraint as a deferrable constraint

  The second method is useful if data loads are performed frequently.



Enabling Constraints

ENABLE NOVALIDATE

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

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

13-19

Enabling Constraints

A constraint that is currently disabled can be enabled in one of the two ways: `ENABLE NOVALIDATE` or `ENABLE VALIDATE`

Enable NOVALIDATE

For `PRIMARY KEY` and `UNIQUE` constraints which have an existing index, enabling a constraint `NOVALIDATE` is much faster than enabling a constraint `VALIDATE` because existing data is not checked for constraint violation if the constraint is deferrable. If this option is used for enabling a constraint, no locks are required on the table. This method is appropriate where there is a lot of DML activity on a table, as in the case of an OLTP environment.

The following command can be used to enable a constraint `ENABLE NOVALIDATE`:

```
ALTER TABLE [ schema. ] table
ENABLE NOVALIDATE {CONSTRAINT constraint
                    | PRIMARY KEY
                    | UNIQUE ( column [, column ] ... ) }
    [ USING INDEX index_clause ]
```

Enabling Constraints (continued)

Restrictions

The USING INDEX clause is applicable only for primary key or unique constraints that were created as deferrable, and when one of the following is true:

- The constraints were created disabled
- The constraints were disabled and the index was dropped.

However, if the index needs to be created, using this method of enabling a constraint does not offer any significant benefit over ENABLE VALIDATE because the Oracle server locks the table to build the index.

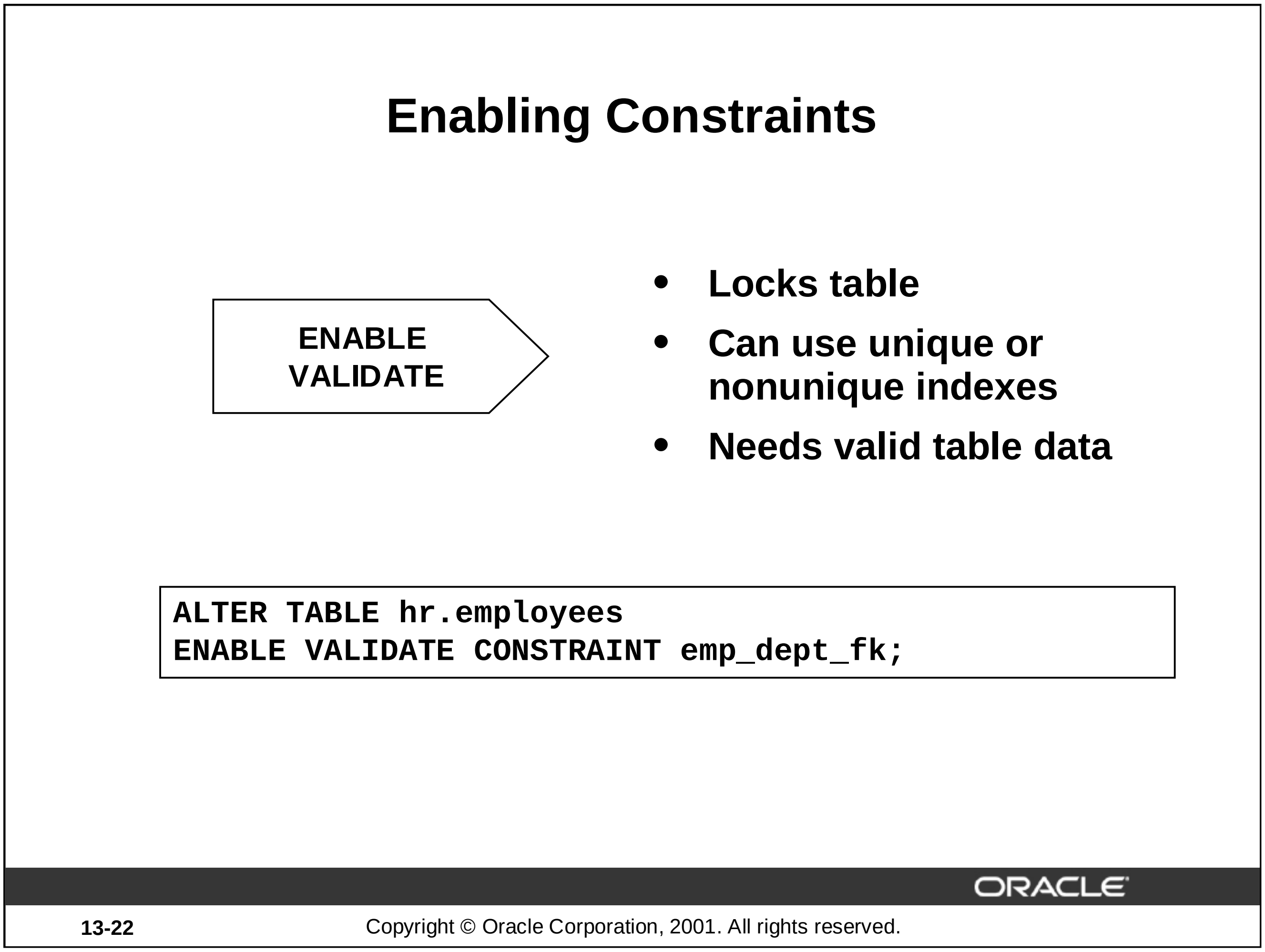
Note: Disabling constraints is covered in the course *Introduction to SQL and PL/SQL.*

Enabling Constraints (continued)

Using Enterprise Manager to modify Constraints

1. Launch the Console:
   `%oemapp console`
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Right-click on the your working database and click Connect.
5. Supply the username, password and service name for your working database and click OK.
6. Expand the Schema folder.
7. Expand the Table folder and select the Table in which Constraints are to be modified
8. Click the Constraints tab, make the requisite modifications
9. Click Apply

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



Enabling Constraints

ENABLE VALIDATE

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

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

Enabling Constraints

Enabling a constraint `VALIDATE` checks existing data for constraint violation. This is the default when a constraint is enabled. If executed when the constraint is disabled, it has the following effects:

- The table is locked and changes to the table are prevented until validation of existing data is complete.
- The Oracle server creates an index if one does not exist on the index columns. It creates a unique index while enabling a primary key or unique constraint that is nondeferrable. A nonunique index is built for a deferrable primary key or a unique constraint.

If this command is executed when a constraint is enforced, it does not require any table locks during validation. The enforced constraint guarantees that no violations are introduced during validation. This has the following advantages:

- All constraints are enabled concurrently.
- Each constraint is internally parallelized.
- Concurrent activity on the table is permitted.

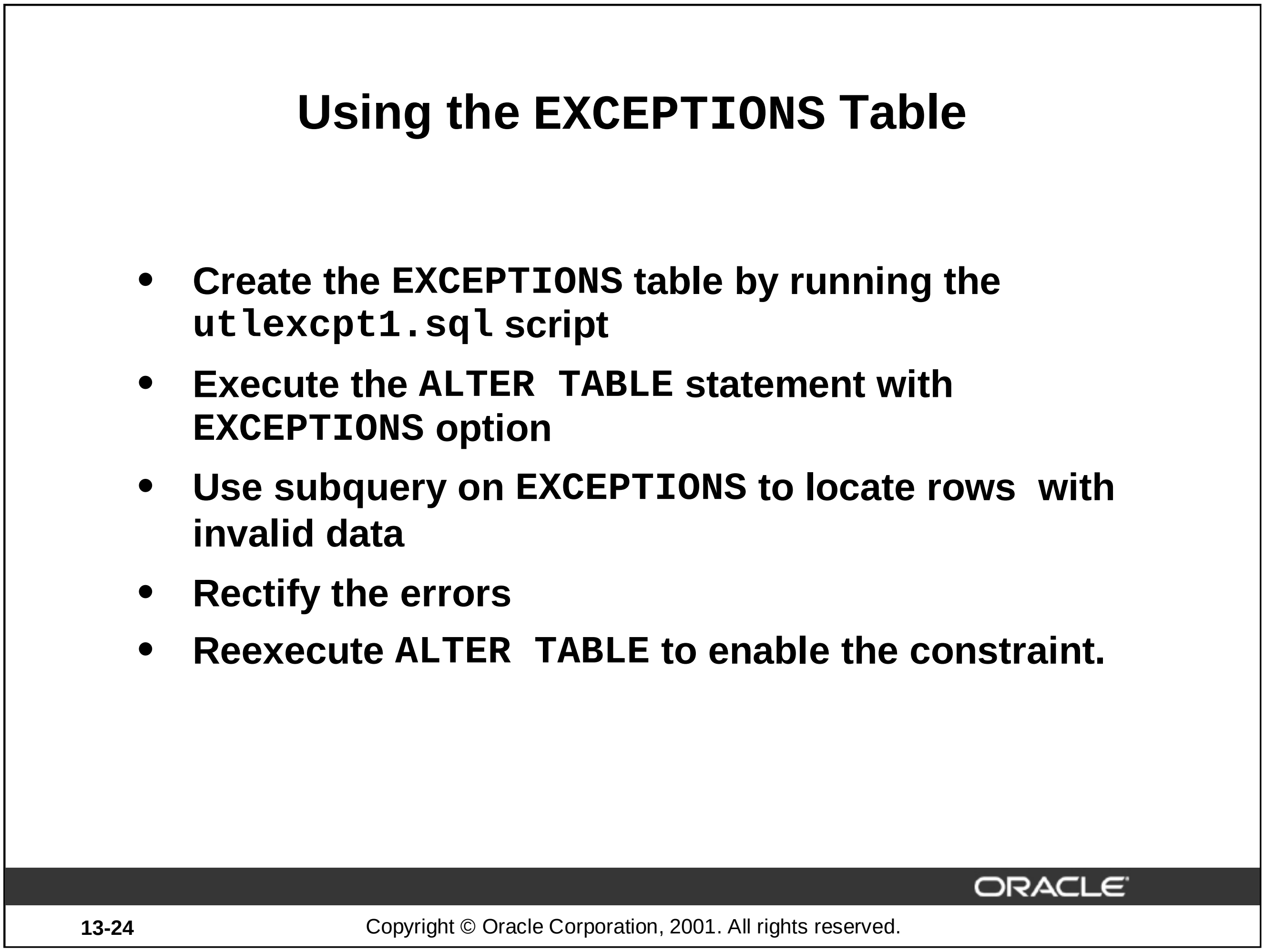
Enabling Constraints (continued)

The following command is used to enable a constraint ENABLE VALIDATE:

```
ALTER TABLE [ schema. ] table
ENABLE [ VALIDATE ]{CONSTRAINT constraint
                    | PRIMARY KEY
                    | UNIQUE ( column [, column ] ... ) }
    [ USING INDEX index_clause ]
[ EXCEPTIONS INTO [ schema. ] table ]
```

Note

- The VALIDATE option is the default and does not need to be specified when enabling a constraint that is disabled.
- If data in the table violates the constraint, then the statement is rolled back and the constraint remains disabled.
- The use of the EXCEPTIONS clause is discussed in the following section.



Using the EXCEPTIONS Table

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

13-24

How to Identify Row Violation

The EXCEPTIONS clause helps to identify any row that violates an enabled constraint. Use the following procedure to detect constraint violations, rectify them, and reenable a constraint:

1. If the EXCEPTIONS is not already created, run the utlexcpt1.sql script:

```
SQL> @?/rdbms/admin/utlexcpt1
Statement processed.
SQL> DESCRIBE exceptions
Name                          Null?     Type
--------------------------    -------   ----------------
ROW_ID                        ROWID
OWNER                                   VARCHAR2(30)
TABLE_NAME                              VARCHAR2(30)
CONSTRAINT                              VARCHAR2(30)
```

Note: The exact name and location of the utlexcpt1.sql script is specific to the operating system. For more information, see your operating system specific Oracle documentation.

How to Identify Row Violation (continued)

2. Execute the ALTER TABLE command using the EXCEPTIONS clause:

```
SQL> ALTER TABLE hr.employee
  2  ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
  3  EXCEPTIONS INTO system.exceptions;
ALTER TABLE hr.employee
*
ORA-02298: cannot enable (hr.EMP_DEPT_FK) - parent keys not
  found
```

If the EXCEPTIONS table is not qualified with the name of the owner, it must belong to the owner of the table being altered.

Rows are inserted into the EXCEPTIONS table. If you are rerunning the command, truncate the EXCEPTIONS table to remove all existing rows.

3. Identify invalid data by using a subquery on the EXCEPTIONS table:

```
SQL> SELECT rowid, id, last_name, dept_id
  2  FROM hr.employee
  3  WHERE ROWID in (SELECT row_id
  4  FROM exceptions)
  5  FOR UPDATE;
ROWID                  ID    LAST_NAME  DEPT_ID
------------------  ----  ---------  -------
AAAAeyAADAAAAA1AAA  1003  Pirie      50
1 row selected.
```

4. Correct the errors in the data:

```
SQL> UPDATE hr.employee
  2  SET id=10
  3  WHERE rowid=’AAAAeyAADAAAAA1AAA’;
1 row processed.
SQL> COMMIT;
Statement processed.
```

How to Identify Row Violation (continued)

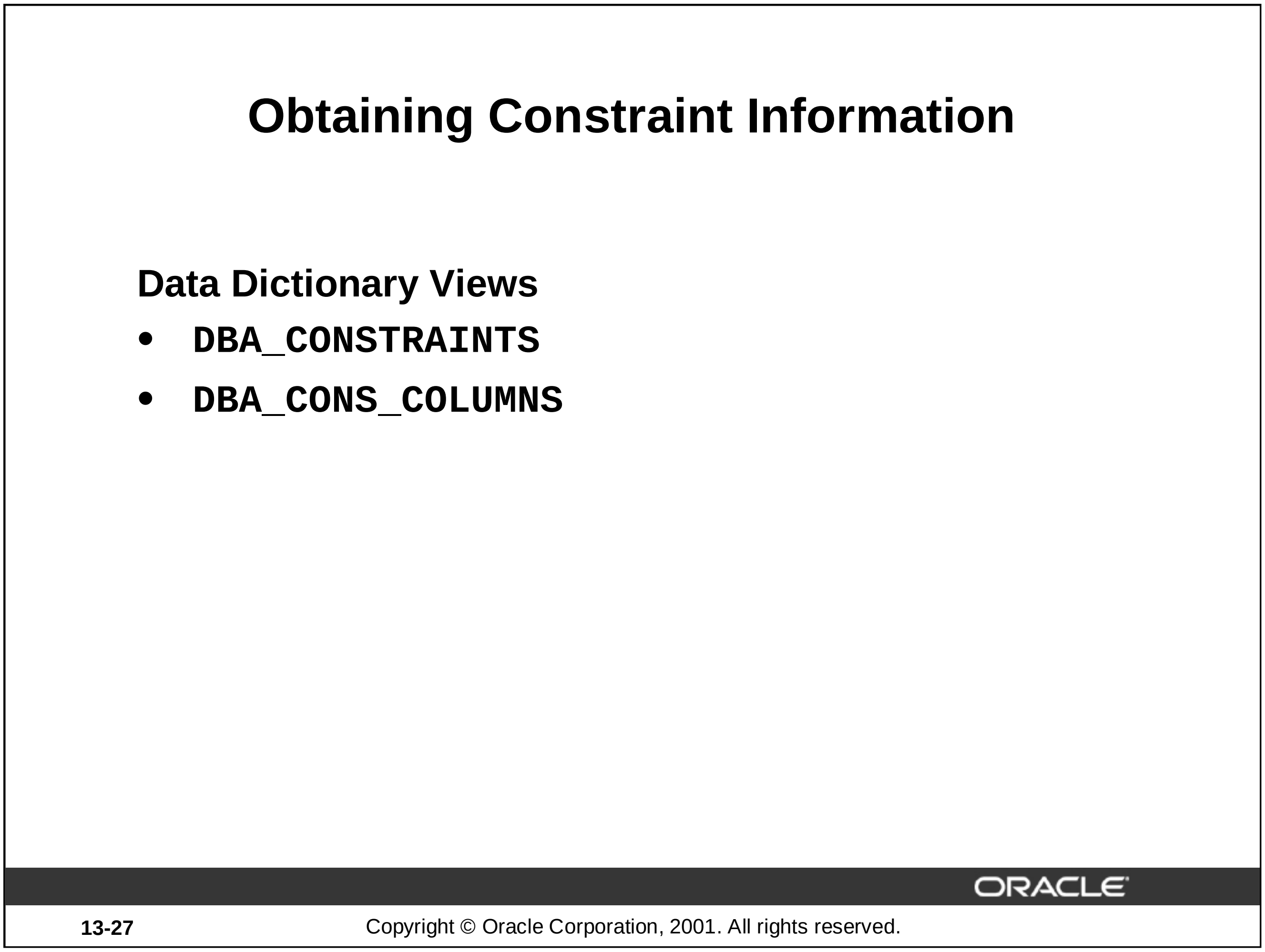
5. Truncate the EXCEPTIONS table and reenable the constraint:

```
SQL> TRUNCATE TABLE exceptions;

Statement processed.

SQL> ALTER TABLE hr.employee
  2  ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
  3  EXCEPTIONS INTO system.exceptions;

Statement processed.
```



Obtaining Constraint Information

Data Dictionary Views

- **DBA_CONSTRAINTS**
- **DBA_CONS_COLUMNS**

ORACLE

13-27

Obtaining Constraint Information

Use the following query to obtain the names, types, and status of all constraints on HR’s EMPLOYEE table:

```
SQL> SELECT constraint_name, constraint_type, deferrable,
  2         deferred, validated
  3  FROM   dba_constraints
  4  WHERE  owner='HR'
  5  AND    table_name='EMPLOYEE';

CONSTRAINT_NAME   C  DEFERRABLE      DEFERRED     VALIDATED
----------------  -  --------------- -----------  ----------
EMPLOYEE_DEPT..   R  DEFERRABLE      DEFERRED     VALIDATED
EMPLOYEE_ID_PK    P  DEFERRABLE      IMMEDIATE    VALIDATED
SYS_C00565        C  NOT DEFERRABLE  IMMEDIATE    VALIDATED

3 rows selected.
```

Obtaining Constraint Information (continued)

The following table shows the columns in the `DBA_CONSTRAINTS` view that are not self-evident:

Name	Description
`CONSTRAINT_TYPE`	The type of constraint is P if Primary Key, U if Unique, R if foreign key, or C if Check constraint. NOT NULL constraints are stored as check constraints.
`SEARCH_CONDITION`	Show the condition specified for a check constraint
`R_OWNER` `R_CONSTRAINT_NAME`	Defines the owner and name of the referenced constraint for foreign keys
`GENERATED`	Indicates whether the constraint name is system-generated (Valid values are ‘USERNAME’ and ‘GENERATED NAME.’)
`BAD`	Indicates that the constraint is to be rewritten to avoid such situations as Year 2000 problems.
`RELY`	If set, this flag will be used in the optimizer
`LAST_CHANGE`	The date when the constraint was last enabled or disabled

Columns in Constraints

To get the columns in the constraints on HR’s EMPLOYEES table, use the following query:

```
SQL> SELECT c.constraint_name, c.constraint_type,
  2         cc.column_name
  3  FROM   dba_constraints c, dba_cons_columns cc
  4  WHERE  c.owner='HR'
  5  AND    c.table_name='EMPLOYEE'
  6  AND    c.owner = cc.owner
  7  AND    c.constraint_name = cc.constraint_name
  8  ORDER BY cc.position;

CONSTRAINT_NAME   C  COLUMN_NAME
----------------  -  ---------------
EMPLOYEE_DEPT...  R  DEPT_ID
EMPLOYEE_ID_PK    P  ID
SYS_C00565        C  LAST_NAME

3 rows selected.
```

Obtaining Constraint Information (continued)

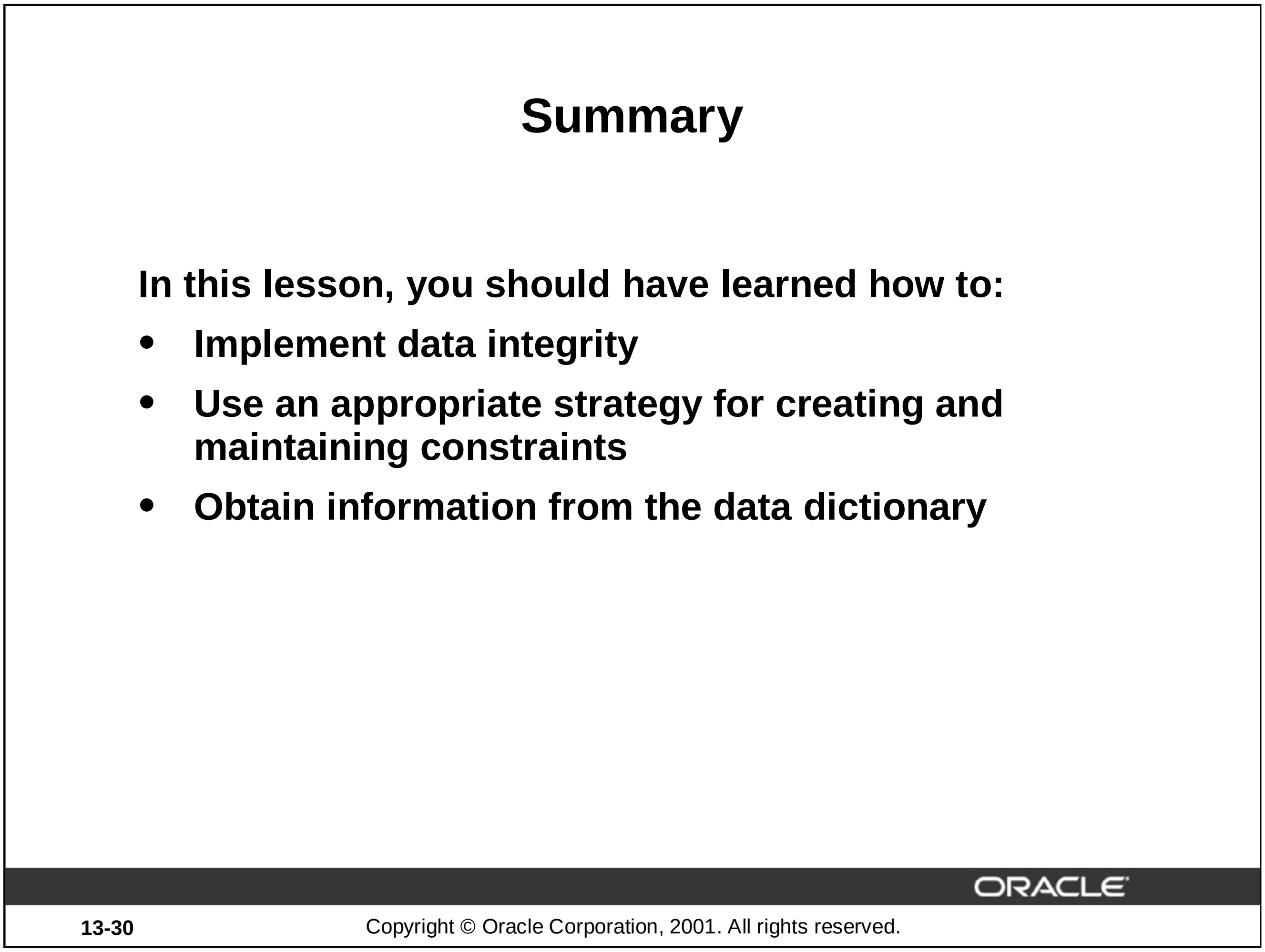
Finding Primary Key–Foreign Key Relationships

To find foreign keys on hr’s EMPLOYEE table and the parent constraints, use the following query:

```
SQL> SELECT c.constraint_name AS "Foreign Key",
  2         p.constraint_name AS "Referenced Key",
  3         p.constraint_type,
  4         p.owner,
  5         p.table_name
  6  FROM   dba_constraints c, dba_constraints p
  7  WHERE  c.owner='HR'
  8  AND    c.table_name='EMPLOYEE'
  9  AND    c.constraint_type='R'
  10 AND    c.r_owner=p.owner
  11 AND    c.r_constraint_name = p.constraint_name;
```

```
Foreign Key       Referenced Key  C  OWNER       TABLE_NAME
------------      --------------  -  ----------  ----------
EMPLOYEES_DEPT..   DEPT_PK        P  HR          DEPARTMENT
```

```
1 row selected.
```



Summary

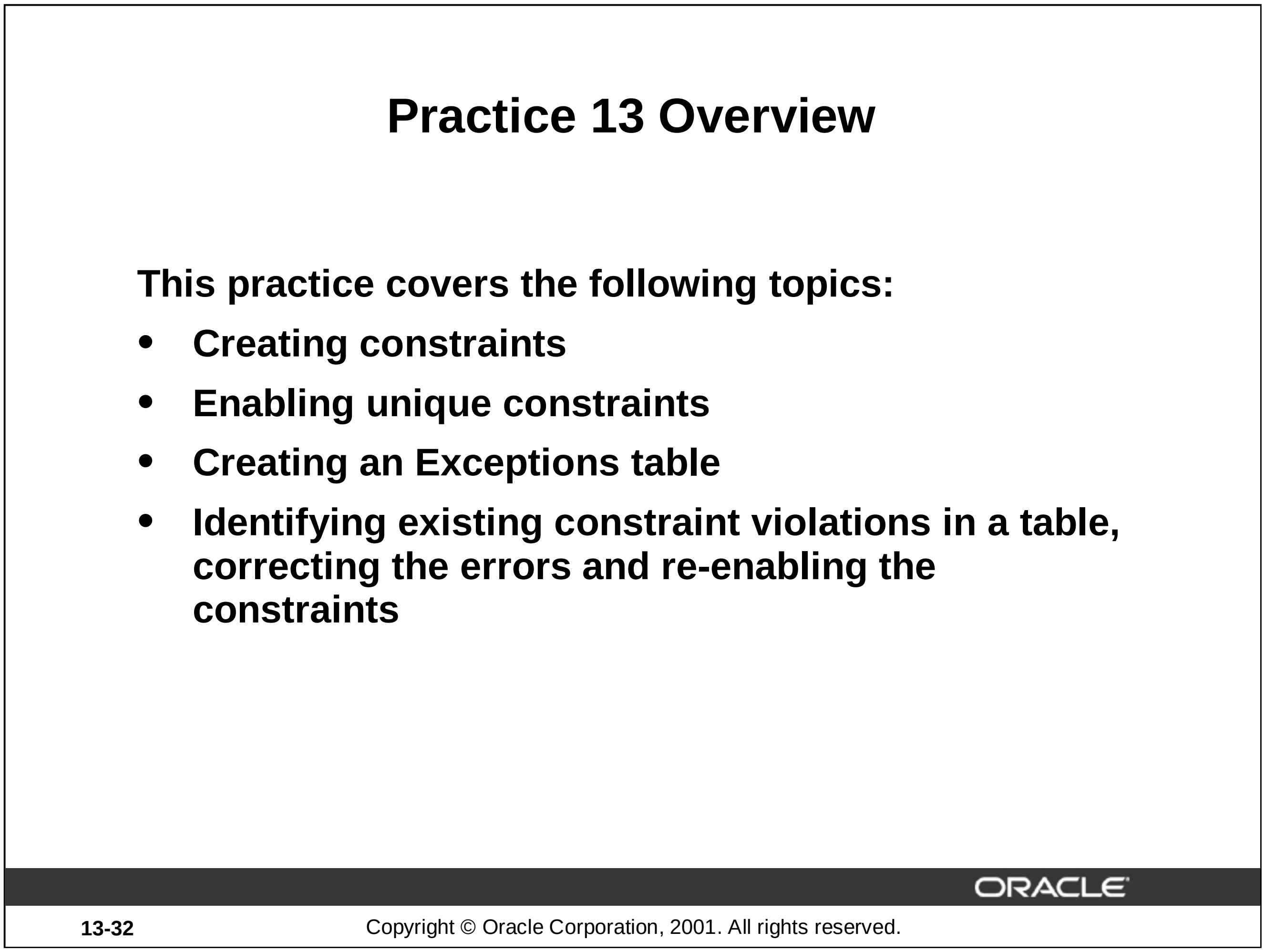
In this lesson, you should have learned how to:

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

13-30

Quick Reference

Context	**Reference**
Initialization parameters	None
Dynamic performance views	None
Data dictionary views	DBA_CONSTRAINTS DBA_CONS_COLUMNS
COMMANDS	CREATE TABLE…CONSTRAINT ALTER TABLE ADD CONSTRAINT…EXCEPTIONS INTO ALTER TABLE…DISABLE CONSTRAINT ALTER TABLE…ENABLE NOVALIDATE CONSTRAINT ALTER TABLE…ENABLE VALIDATE CONSTRAINT…EXCEPTIONS INTO
Packages, procedures, and functions	None



Practice 13 Overview

This practice covers the following topics:

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

Practice 13: Maintaining Data Integrity

1 Examine the script `lab13_01.sql`. Run the script to create the constraints.

2 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

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

3 Insert two records with the following values into the `PRODUCTS` table:

PRODUCT_ID	PRODUCT_DESCRIPTION	LIST_PRICE
4000	UNIX Monitor	3620
4000	NT Monitor	2400

4 Enable the unique constraint on the `PRODUCT` table. Was it successful? Why or why not?

5 **a** Ensure that new rows added to the table do not violate the constraint on the `PRODUCT` table.

Hint: This can be done by enabling the constraint `NOVALIDATE`.

b Query the data dictionary to verify the effect of the change.

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

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.

b Run the command to enable the constraint and trap the exceptions.

c Use the `ROWID`s in the `EXCEPTIONS` table to list the rows in the `PRODUCTS` table that violate the constraint. (Do not list LOB columns.)

d Rectify the errors.

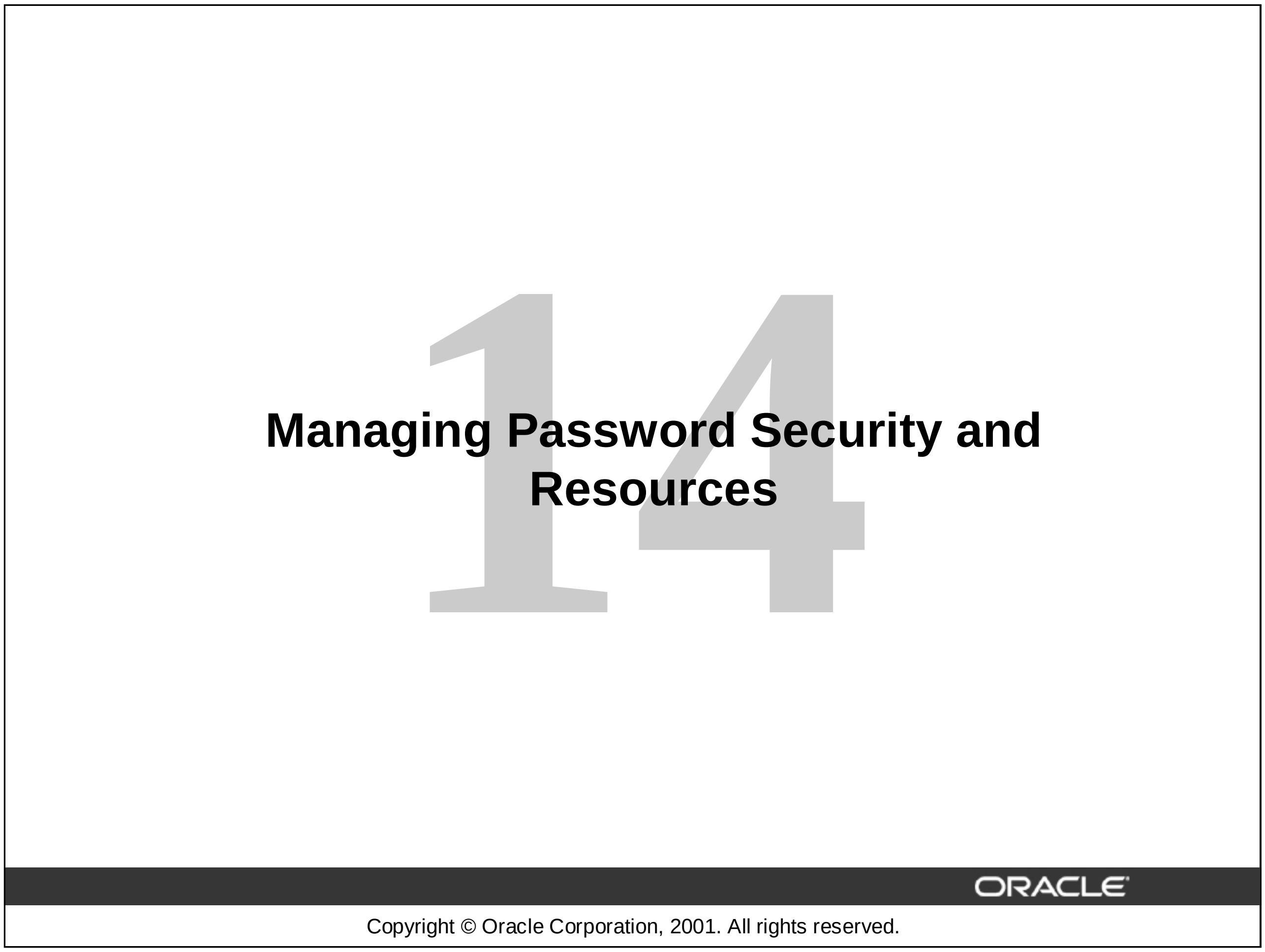
e Enable the constraint.

Practice 13: Maintaining Data Integrity

7 Run the script `lab13_07.sql` to insert rows into the table. Were the inserts successful? Roll back the changes.

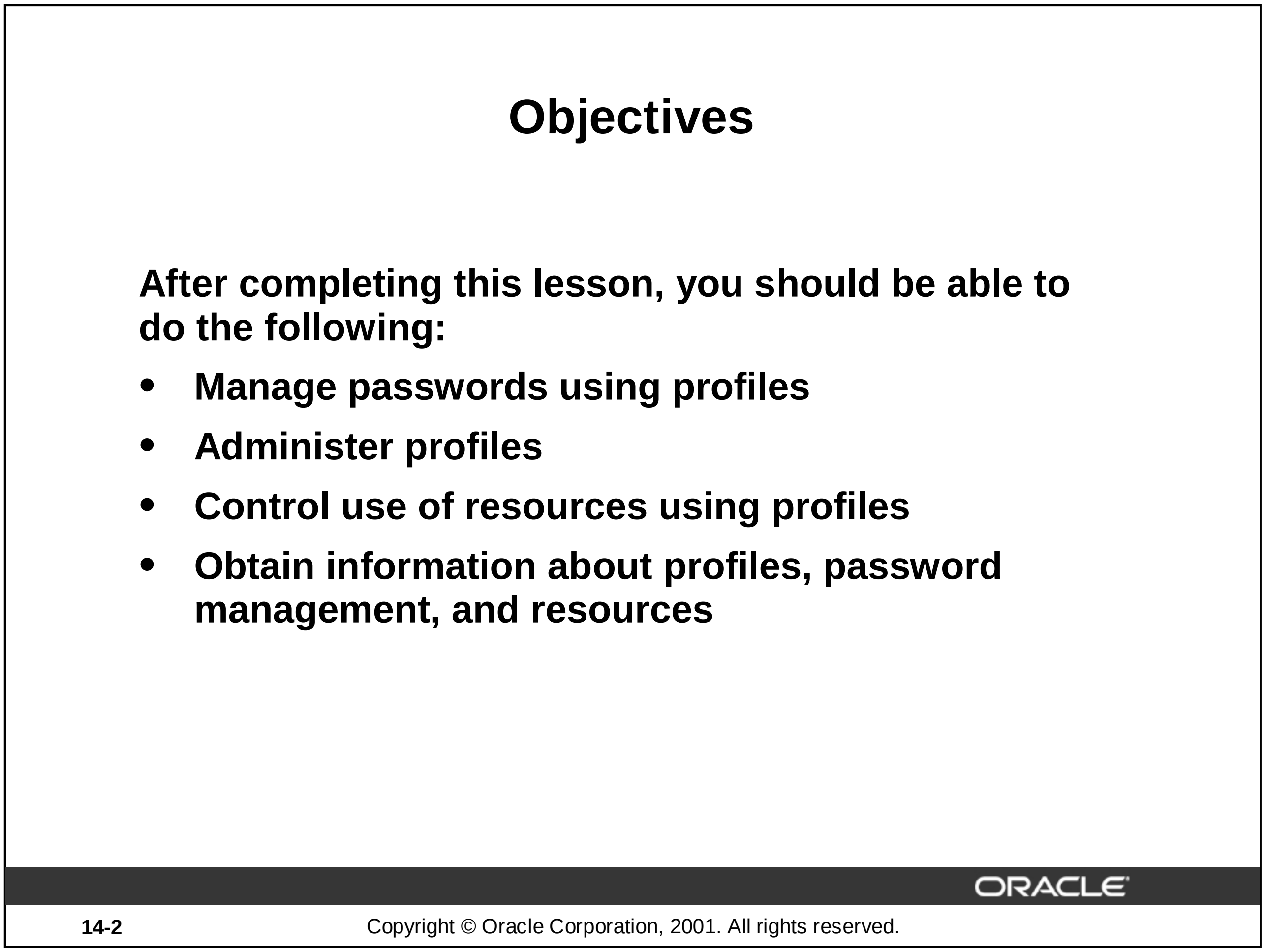
8 Now examine the script `lab13_08`. Notice that this script also performs the inserts in the same sequence. Run the script and check if it executes successfully.

9 Truncate the `CUSTOMERS` table. Was it successful? Why or why not?



14

Managing Password Security and Resources

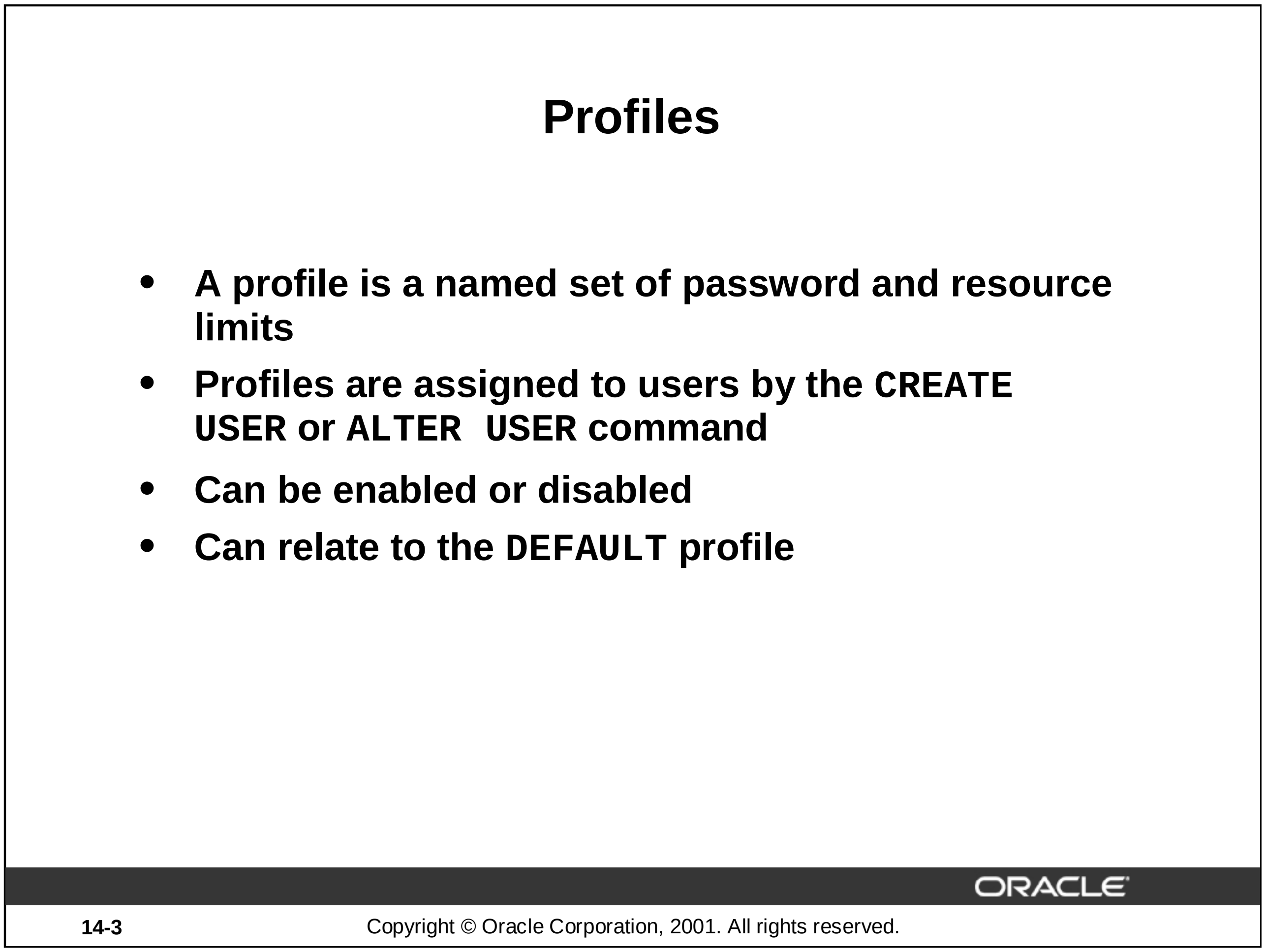


Objectives

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

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

14-2



Profiles

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

14-3

What Is a Profile?

A profile is a named set of the following password and resource limits:

- Password aging and expiration
- Password history
- Password complexity verification
- Account locking
- CPU time
- I/O operations
- Idle time
- Connect time
- Memory space (private SQL area for MTS only)
- Concurrent sessions

After a profile has been created, the database administrator can assign it to each user. If resource limits are enabled, the Oracle server limits the database usage and resources to the defined profile of the user.

DEFAULT Profile

The Oracle server automatically creates a `DEFAULT` profile when the database is created.

The users who have not been explicitly assigned a specific profile conform to all the limits of the `DEFAULT` profile. All limits of the `DEFAULT` profile are initially unlimited. However, the database administrator can change the values so that limits are applied to all users by default.

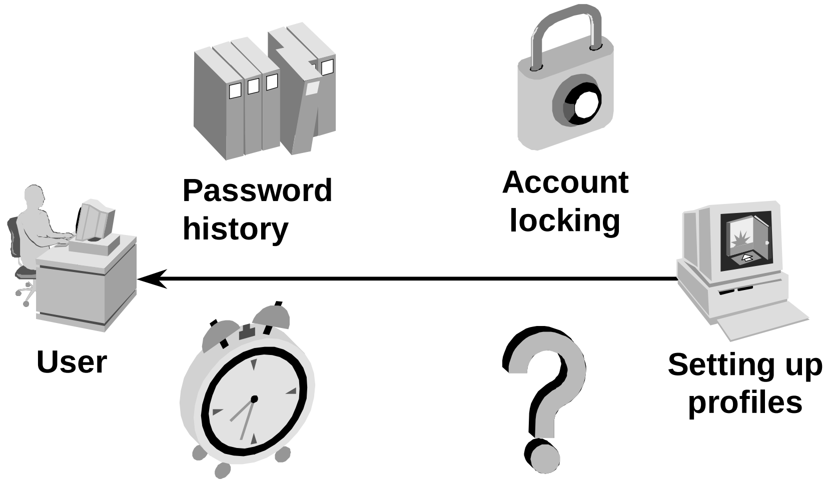
Profile Usage

- Restrict users from performing some operations that require heavy use of resources
- Ensure that users log off the database when they have left their session idle for some time
- Enable group resource limits for similar users
- Easily assign resource limits to users
- Manage resource usage in large, complex multiuser database systems
- Control the use of passwords

Profile Characteristics

- Profile assignments do not affect current sessions.
- Profiles can be assigned only to users and not to roles or other profiles.
- If you do not assign a profile when creating a user, the user is automatically assigned the `DEFAULT` profile.

Password Management



Password history

Account locking

User

Setting up profiles

Password expiration and aging

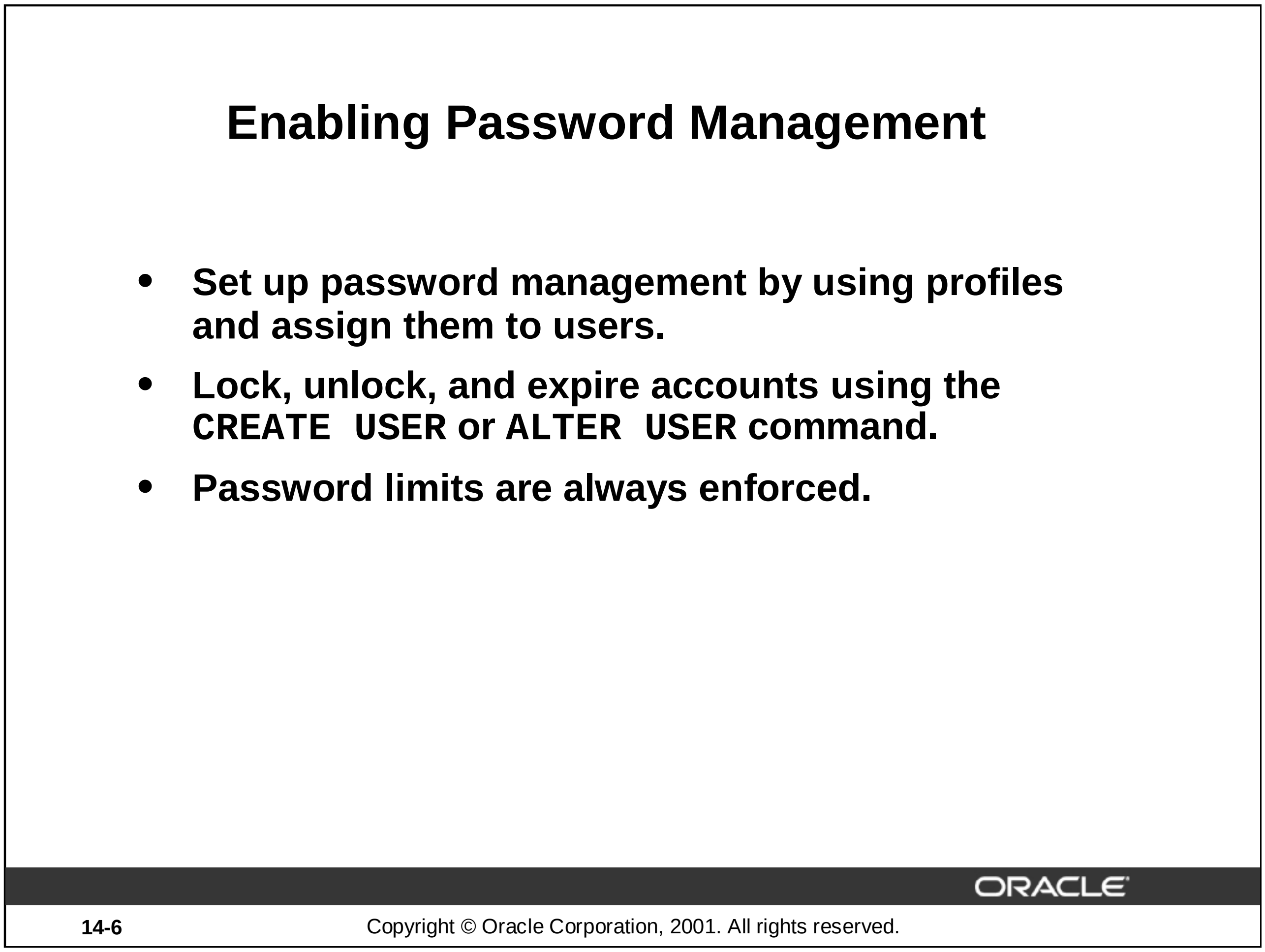
Password verification

Password Management Features

For greater control over database security, Oracle password management is controlled by database administrators with profiles.

This lesson describes the available password management features:

- Account locking: Enables automatic locking of an account when a user fails to log into the system in the specified number of attempts
- Password aging and expiration: Enables the password to have a lifetime, after which it expires and must be changed
- Password history: Checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes
- Password complexity verification: Makes a complexity check on the password to verify that it is complex enough to provide protection against intruders who might try to break into the system by guessing the password



Enabling Password Management

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

14-6

How to Enable Password Management

Create the profile to limit the password settings, and assign the profile to the user by using the CREATE USER or ALTER USER command.

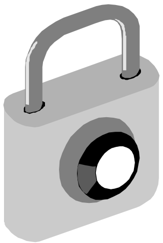
Password limit settings in profiles are always enforced.

When password management is enabled, the user account can be locked or unlocked by using the CREATE USER or ALTER USER command.

Note: Refer to the Managing Users lesson for details regarding the CREATE USER command.

Password Account Locking

Parameter	Description
FAILED_LOGIN_ATTEMPTS	**Number of failed login attempts before lockout of the account**
PASSWORD_LOCK_TIME	**Number of days the account is locked after the specified number of failed login attempts**



14-7

Account Locking

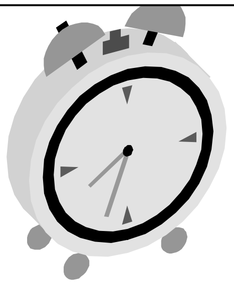
The Oracle server automatically locks an account after the FAILED_LOGIN_ATTEMPTS value is reached. The account is either automatically unlocked after a specified time PASSWORD_LOCK_TIME or it must be unlocked by the database administrator using the ALTER USER command.

The database account can be explicitly locked with the ALTER USER command. When this happens, the account is not automatically unlocked.

Note: The ALTER USER command will be demonstrated later in this lesson.

Password Expiration and Aging

Parameter	Parameter
PASSWORD_LIFE_TIME	Lifetime of the password in days after which the password expires
PASSWORD_GRACE_TIME	Grace period in days for changing the password after the first successful login after the password has expired



Password Aging and Expiration

The PASSWORD_LIFE_TIME parameter sets the maximum lifetime after which the password must be changed.

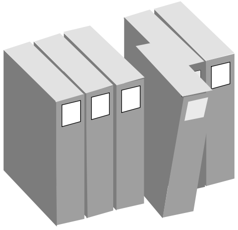
The database administrator can specify a grace period PASSWORD_GRACE_TIME, which begins after the first attempt to log in to the database after password expiration. A warning message is generated every time the user tries to log in until the grace period is over. The user is expected to change the password within the grace period.

If the password is not changed, the account is locked.

The user’s account status is changed to EXPIRED by explicitly setting the password to be expired.

Password History

Parameter	Description
PASSWORD_REUSE_TIME	Number of days before a password can be reused
PASSWORD_REUSE_MAX	Maximum number of times a password can be reused

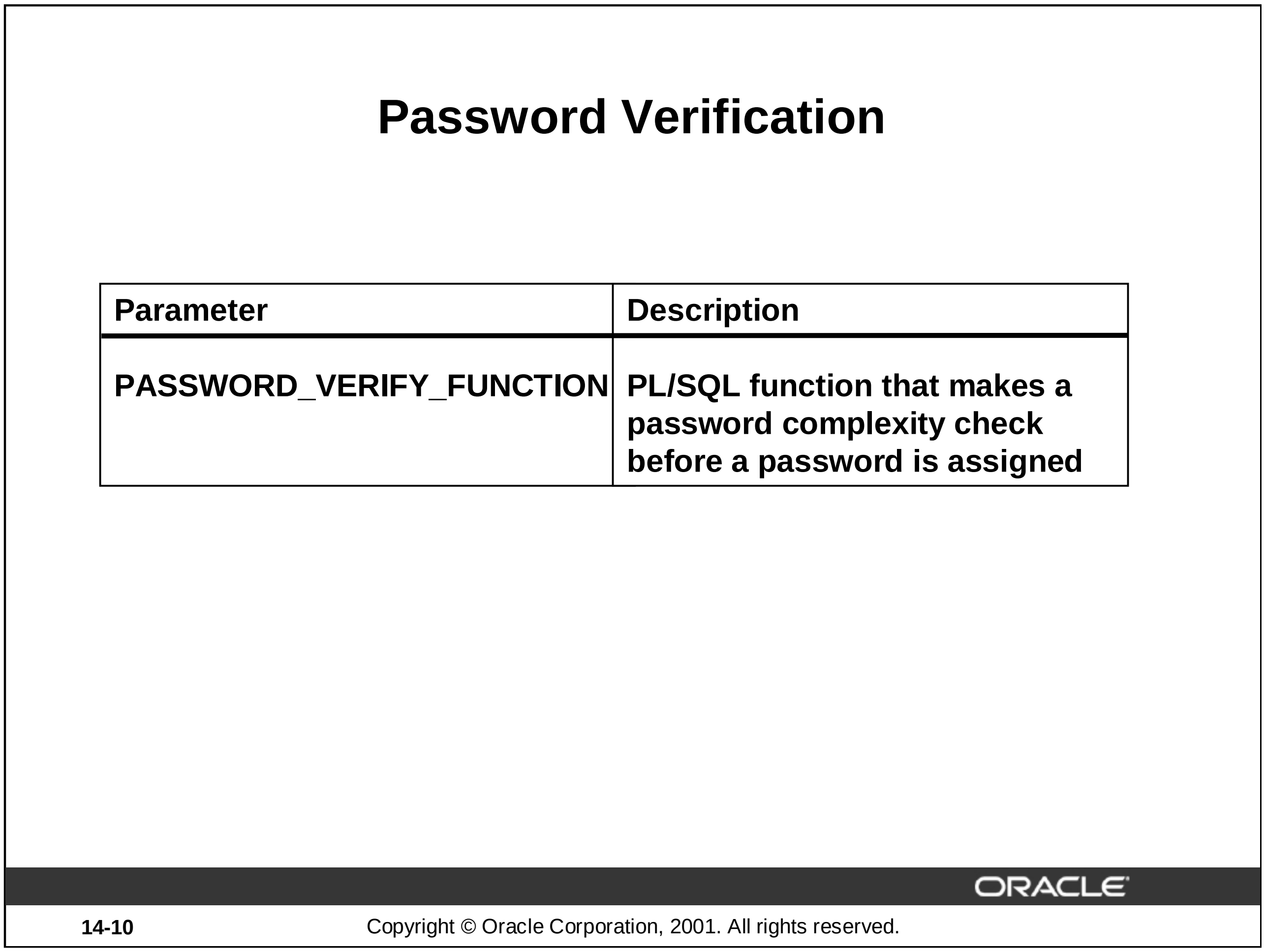


Password History

Password history checks ensure that a user cannot reuse a password for a specified time interval. These checks can be implemented using one of the following:

- PASSWORD_REUSE_TIME to specify that a user cannot reuse a password for a given number of days
- PASSWORD_REUSE_MAX to force a user to define a password that is not identical to earlier passwords

When one parameter is set to a value other than DEFAULT or UNLIMITED, the other parameter must be set to UNLIMITED.



Password Verification

Parameter	Description
PASSWORD_VERIFY_FUNCTION	PL/SQL function that makes a password complexity check before a password is assigned

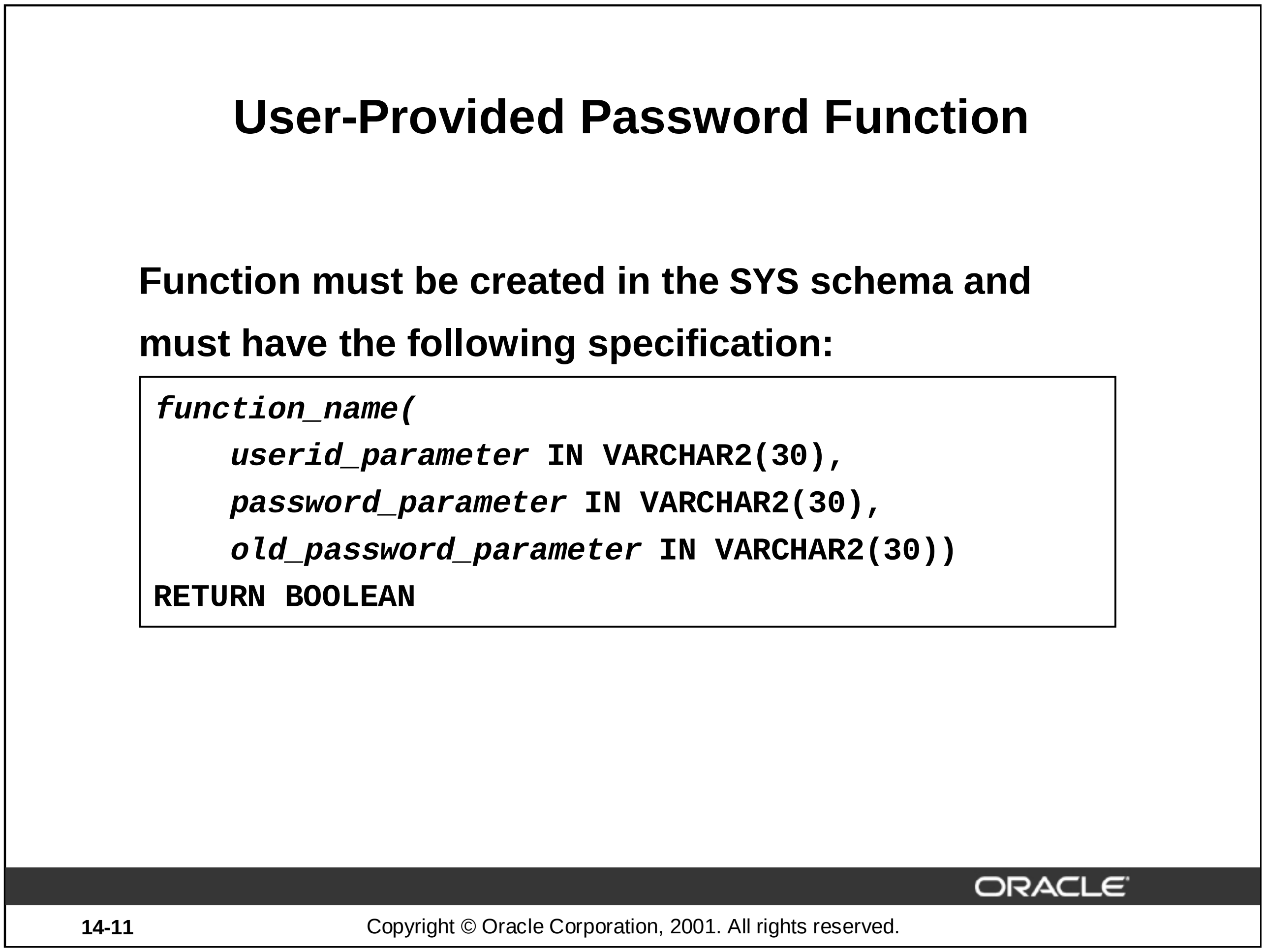
ORACLE

14-10

Password Verification

Before assigning a new password to a user, a PL/SQL function can be invoked to verify the validity of the password.

The Oracle server provides a default verification routine or the database administrator can write a PL/SQL function.



User-Provided Password Function

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

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

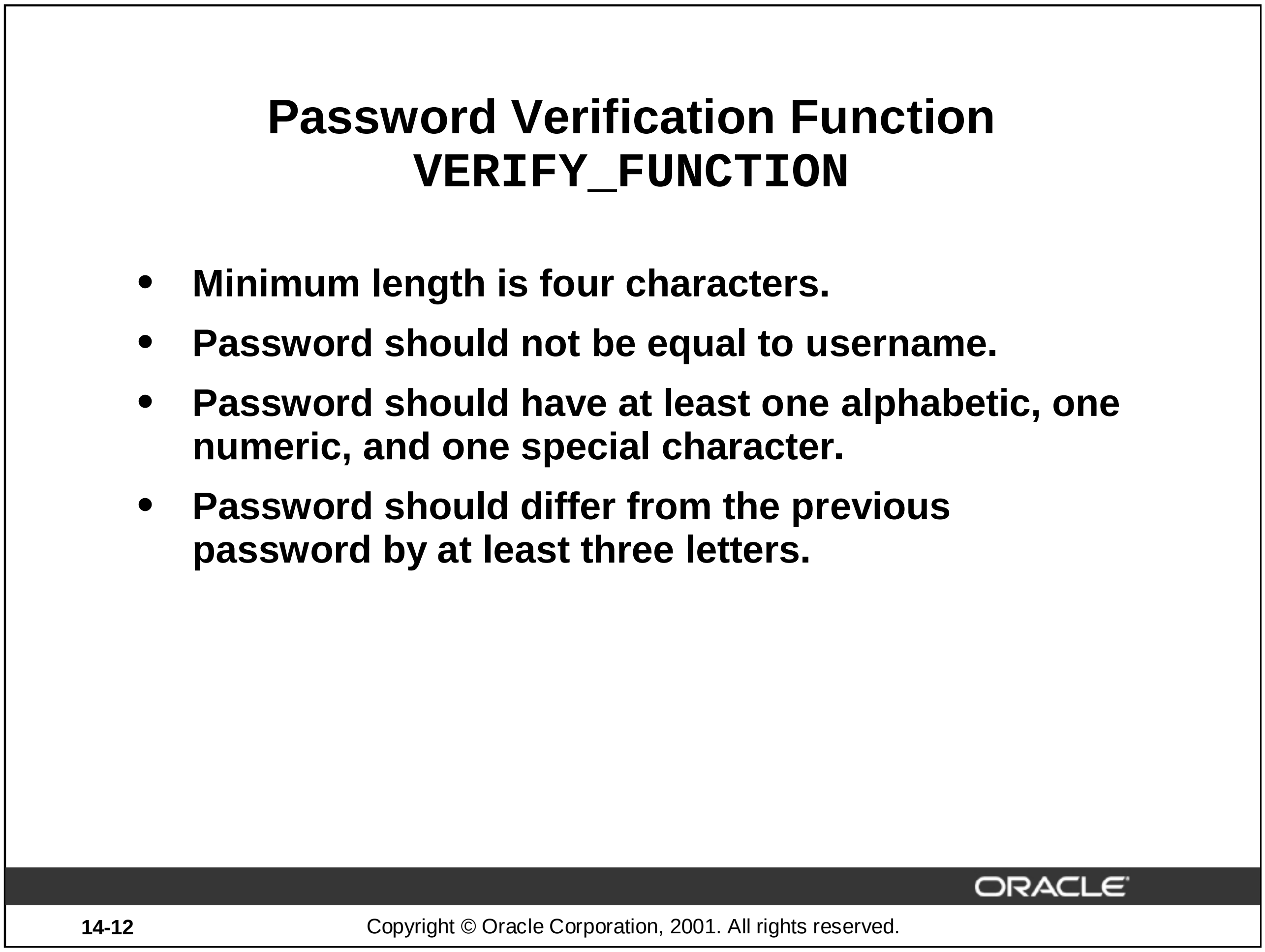
14-11

How to Define a Function to Verify a Password

When a new password verification function is added, the database administrator must consider the following restrictions:

- The procedure must use the specification indicated in the slide.
- The procedure returns the value TRUE for success and FALSE for failure.
- If the password function raises an exception, an error is returned and the ALTER USER or CREATE USER command is terminated.
- The password function is owned by SYS.
- If the password function becomes invalid, an error message is returned and the ALTER USER or CREATE USER command is terminated.

Note: Refer to the Managing Users lesson for details regarding CREATE USER.



Password Verification Function VERIFY_FUNCTION

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

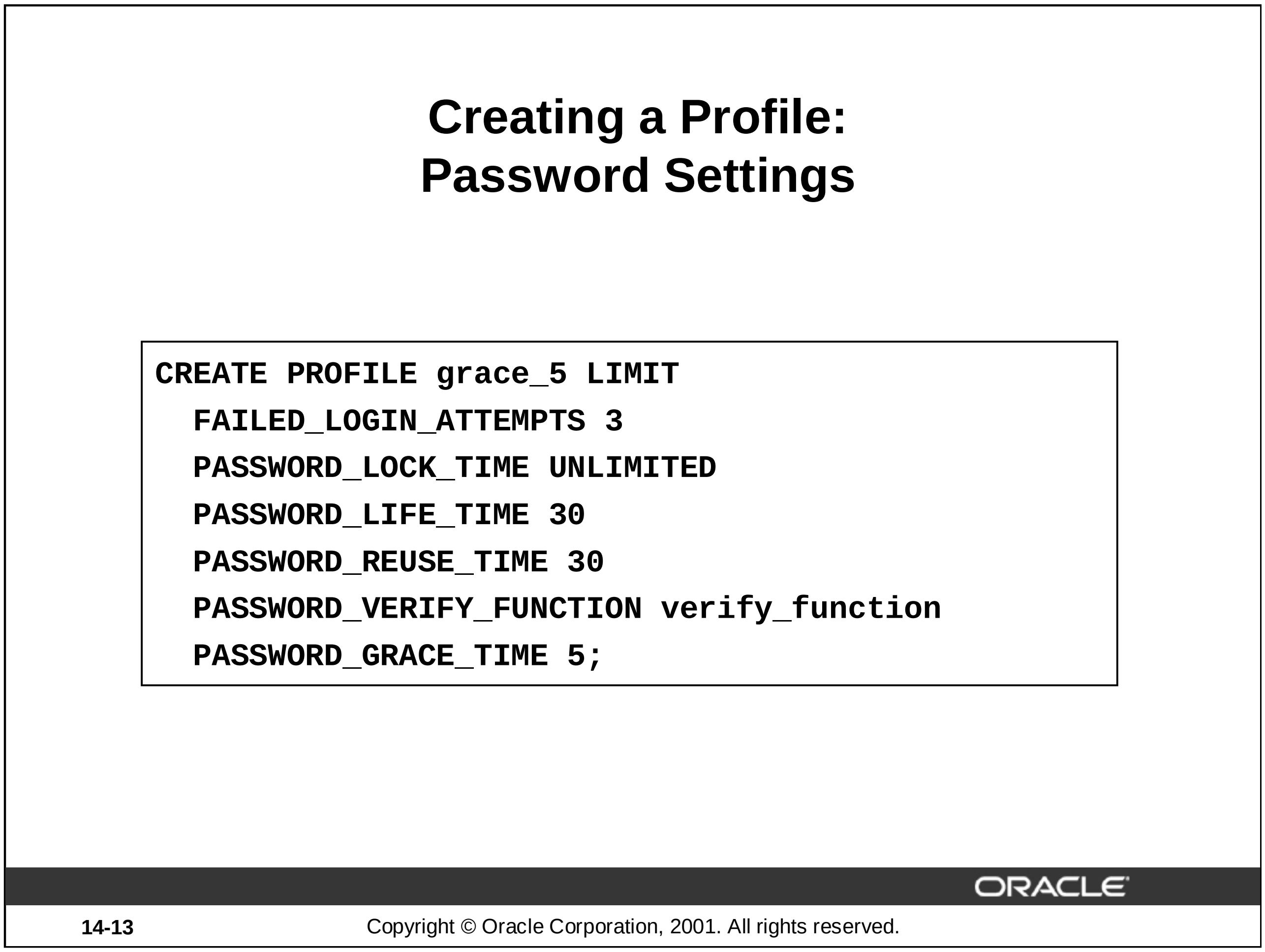
14-12

The Default Verification Function

The Oracle server provides a complexity verification function, in the form of a default PL/SQL function called `VERIFY_FUNCTION` of the script utlpwdmg.sql, which must be run in the `SYS` schema.

During the execution of the script utlpwdmg.sql, the Oracle server creates `VERIFY_FUNCTION` and changes the `DEFAULT` profile with the following `ALTER PROFILE` command:

```
ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIFY_FUNCTION verify_function;
```



Creating a Profile: Password Settings

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

14-13

How to Create a Profile

Use the following `CREATE PROFILE` command to administer passwords:

```
CREATE PROFILE profile LIMIT
     [FAILED_LOGIN_ATTEMPTS    max_value]
     [PASSWORD_LIFE_TIME       max_value]
     [ {PASSWORD_REUSE_TIME
       |PASSWORD_REUSE_MAX}    max_value]
     [PASSWORD_LOCK_TIME        max_value]
     [PASSWORD_GRACE_TIME      max_value]
     [PASSWORD_VERIFY_FUNCTION
      {function|NULL|DEFAULT} ]
```

How to Create a Profile (continued)

Where

`PROFILE:` is the name of the profile to be created

`FAILED_LOGIN_ATTEMPTS:` specifies the number of failed attempts to log in to the user account before the account locked

`PASSWORD_LIFE_TIME:` limits the number of days the same password can be used for authentication. The password expires if it is not changed within this period, and further connections are rejected.

`PASSWORD_REUSE_TIME:` specifies the number of days before a password can be reused. If you set `PASSWORD_REUSE_TIME` to an integer value, then you must set `PASSWORD_REUSE_MAX` to `UNLIMITED`.

`PASSWORD_REUSE_MAX:` specifies the number of password changes required before the current password can be reused. If you set `PASSWORD_REUSE_MAX` to an integer value, then you must set `PASSWORD_REUSE_TIME` to `UNLIMITED`.

`PASSWORD_LOCK_TIME:` specifies the number of days an account will be locked after the specified number of consecutive failed login attempts

`PASSWORD_GRACE_TIME:` specifies the number of days after the grace period begins during which a warning is issued and login is allowed. If the password is not changed during the grace period, the password expires.

`PASSWORD_VERIFY_FUNCTION:` allows a PL/SQL password complexity verification script to be passed as an argument to the `CREATE PROFILE` statement

How to Use Oracle Enterprise Manager to Create a Profile

Launch Security Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone

   You can also launch the Console from Windows NT Start menu
2. Expand your working database from the databases folder
3. Expand Security folder and select the Profiles folder
4. Select Create from the Object menu
5. Select Profile in the list and click Create.
6. Enter a name for the Profile and complete other fields or accept the default values
7. Select the Password tab and enter the account password parameters
8. Click Create.

Assigning a Profile

With the `CREATE USER` command or the `ALTER USER` command, a profile can be assigned. Each user can be assigned only one profile at a time.

Note: Refer to the Managing Users lesson for details regarding `CREATE USER`.

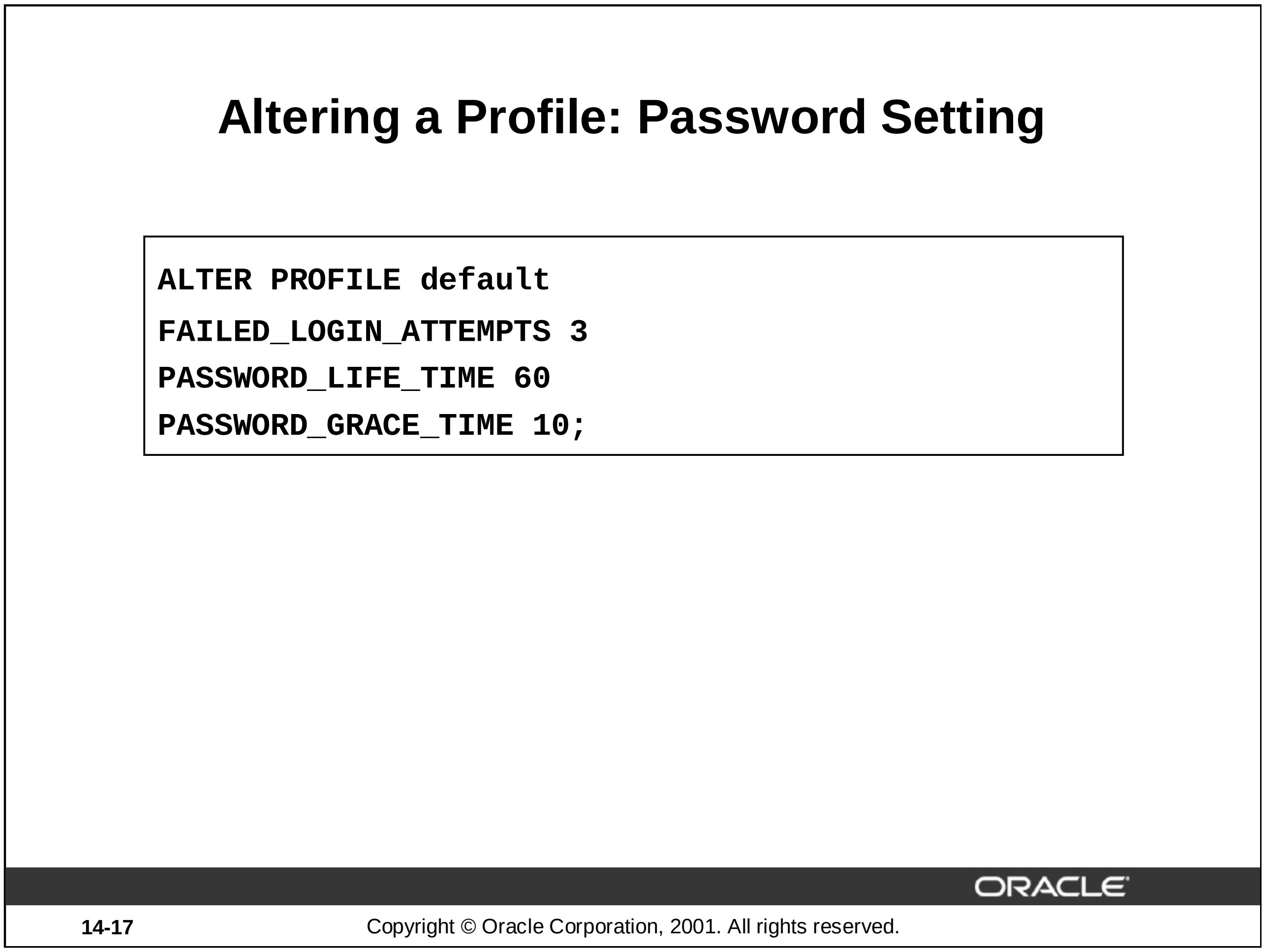
How to Use Oracle Enterprise Manager to Assign a Profile to a User

To generate the ALTER USER command with Oracle Enterprise Manager use the following steps:

Launch Security Manager from the Console.

1. Launch the Console
   - %oemapp console
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand the Security folder
4. From the Profiles folder, select Object—>Assign a Profile to User(s).
5. In the Assign Profile dialog box select the user(s)
6. Click OK

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



Altering a Profile: Password Setting

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

14-17

Altering a Profile

The ALTER PROFILE command is used to change the password limits assigned to a profile:

```
ALTER PROFILE profile LIMIT
  [FAILED_LOGIN_ATTEMPTS    max_value]
  [PASSWORD_LIFE_TIME       max_value]
  [ {PASSWORD_REUSE_TIME
    |PASSWORD_REUSE_MAX}    max_value]
  [PASSWORD_LOCK_TIME        max_value]
  [PASSWORD_GRACE_TIME      max_value]
  [PASSWORD_VERIFY_FUNCTION
   {function|NULL|DEFAULT} ]
```

If you want to set the password parameters to less than a day:

1 hour: `PASSWORD_LOCK_TIME = 1/24`

10 minutes: `PASSWORD_LOCK_TIME = 10/1400`

5 minutes: `PASSWORD_LOCK_TIME = 5/1440`

How to Use Oracle Enterprise Manager to Alter a Profile

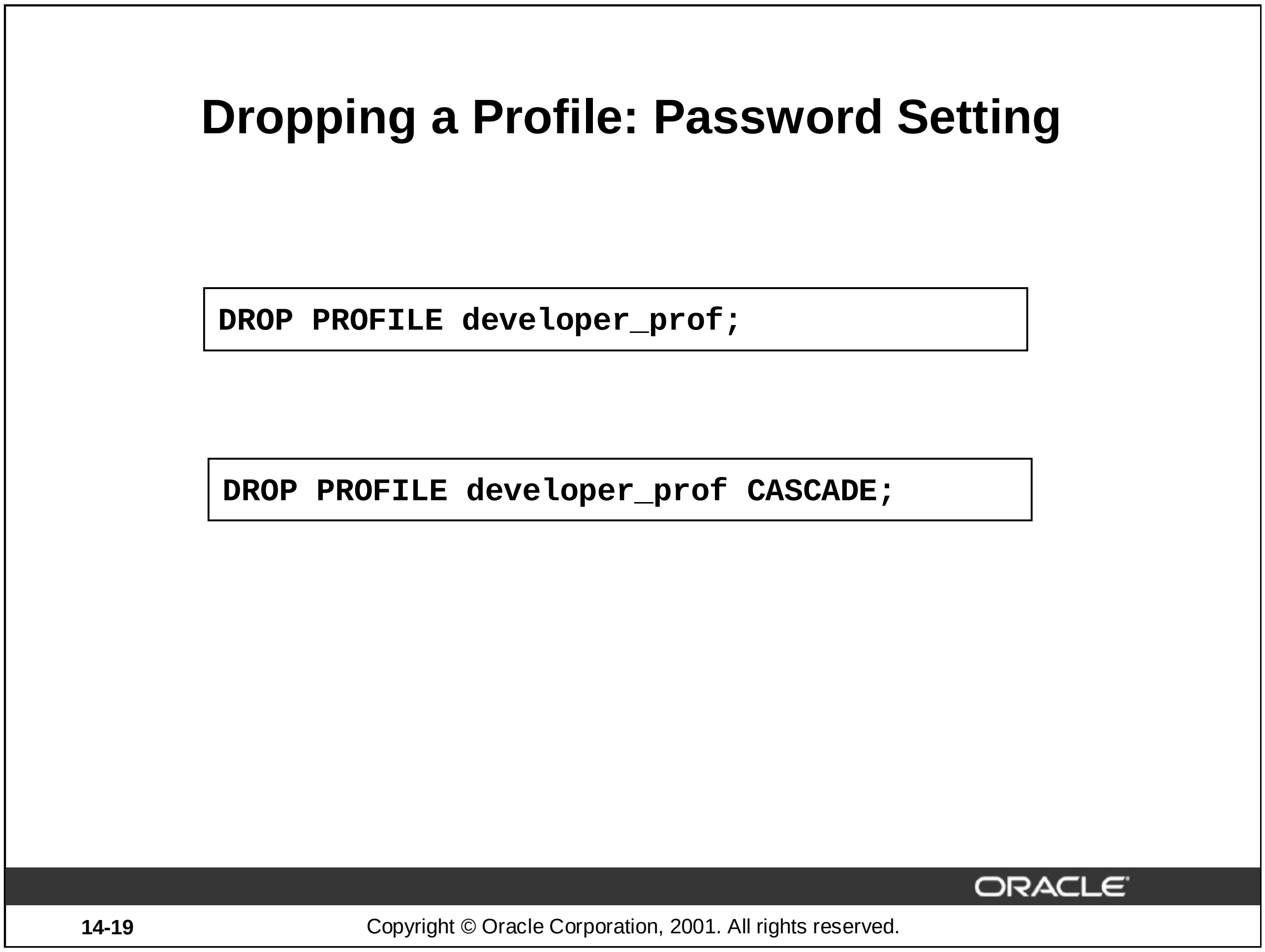
Launch Security Manager from the Console.

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand the Security folder
4. Expand the Profiles folder.
5. Select the profile.
6. Click the Password tab in the detail side of the Console and change the details on the password parameters
7. Click Apply.

Guidelines

Changes to a profile do not affect current sessions. Changes are used in subsequent sessions only.

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



Dropping a Profile: Password Setting

```
DROP PROFILE developer_prof;
```

```
DROP PROFILE developer_prof CASCADE;
```

14-19

Dropping a Profile

Drop a profile using the `DROP PROFILE` command:

```
DROP PROFILE profile [CASCADE]
```

where:

`profile:` is the name of the profile to be dropped

`CASCADE:` revokes the profile from users to whom it is assigned (The Oracle server automatically assigns the DEFAULT profile to such users. Specify this option to drop a profile that is currently assigned to users.)

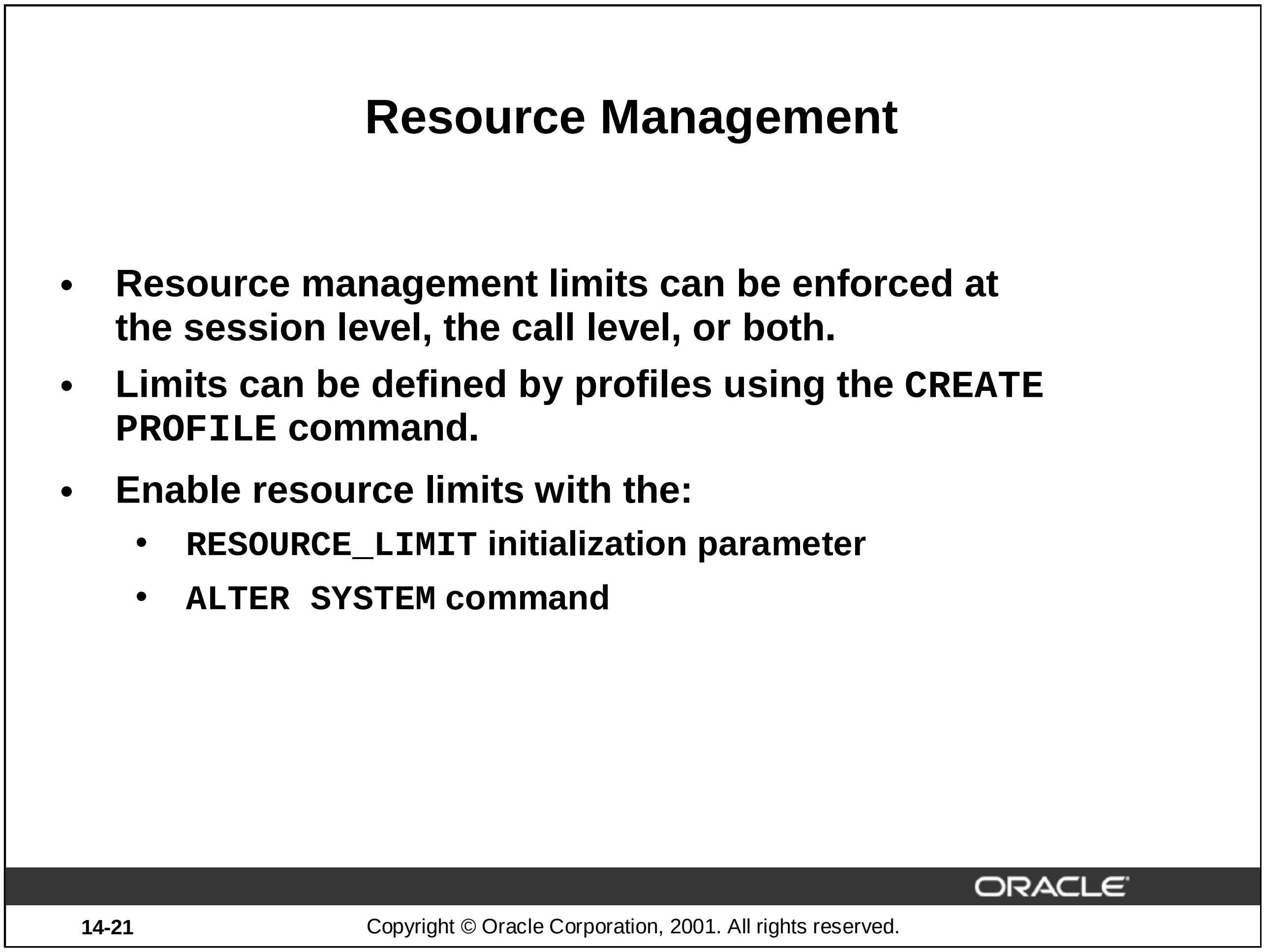
Guidelines

- The `DEFAULT` profile cannot be dropped.
- When a profile is dropped, this change applies to subsequently created sessions only and not to the current sessions.

How to Use Oracle Enterprise Manager to Drop a Profile

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand the Security folder
4. Expand the Profiles folder.
5. Select the profile.
6. Select Object—>Remove

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



Resource Management

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

14-21

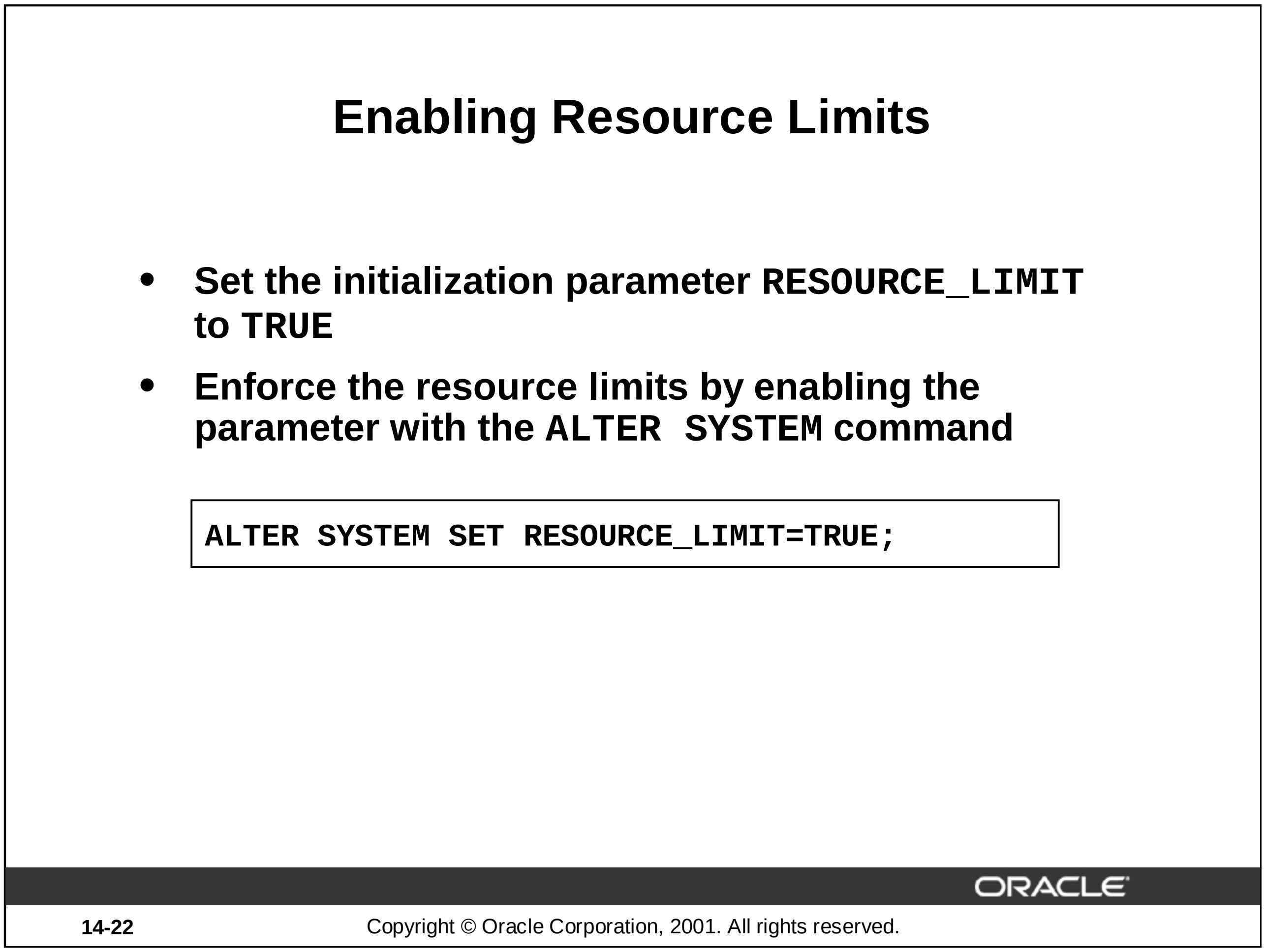
Steps for Using Resource Limits

Use the following steps to control the usage of resources with profiles:

1. Create a profile with the `CREATE PROFILE` command to determine the resource and password limits.
2. Assign profiles with the `CREATE USER` or `ALTER USER` command.
3. Enforce resource limits with the `ALTER SYSTEM` command or by editing the initialization parameter file (and stopping and restarting the instance).

These steps are discussed in detail in the following section.

Note: Enforcing the resource limits is not required for enabling Oracle password management.



Enabling Resource Limits

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

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

Controlling Enforcement of Resource Limits

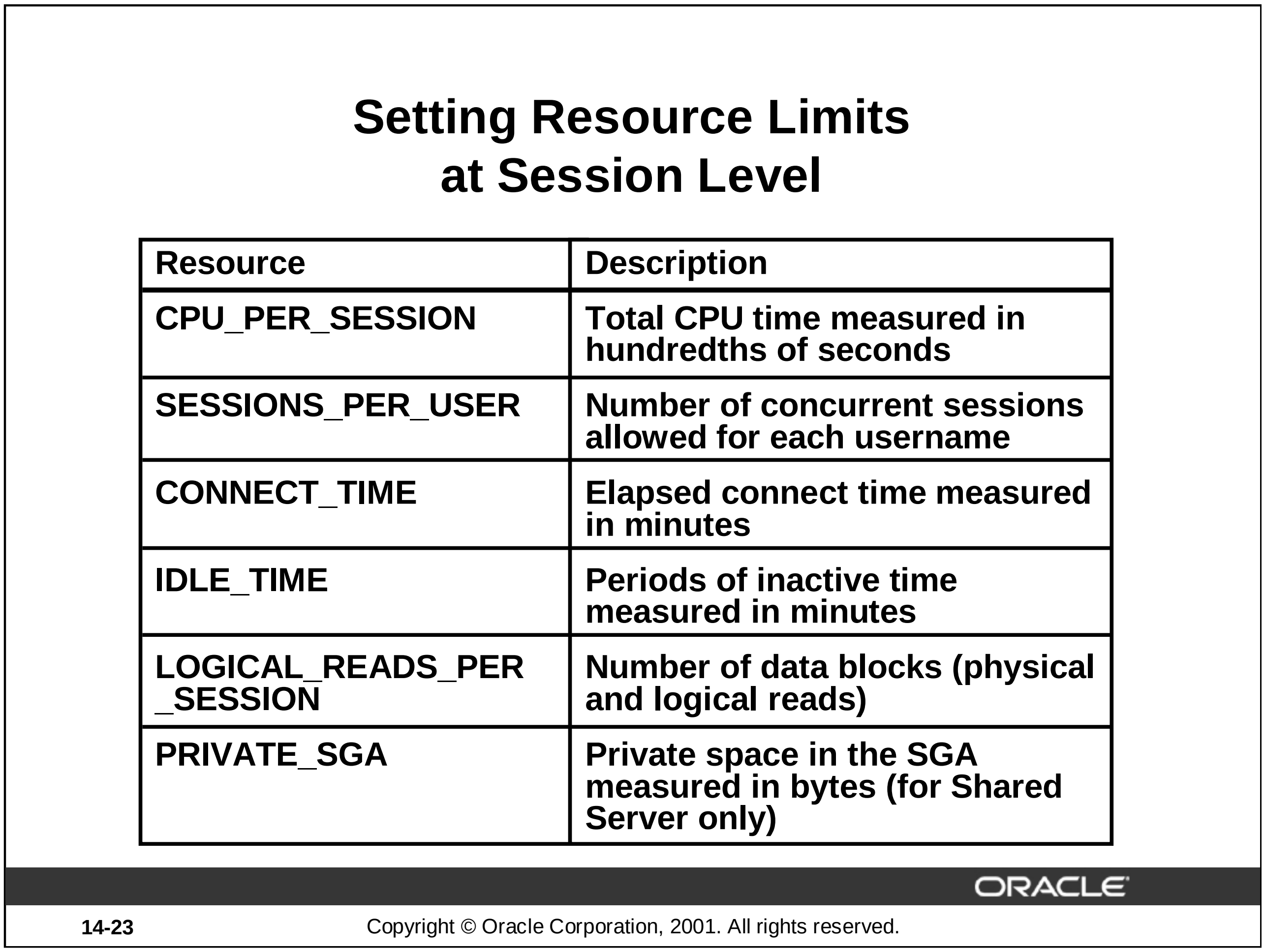
Enable or disable the enforcement of resource limits by altering the `RESOURCE_LIMIT` initialization parameter or by using the `ALTER SYSTEM` command.

`RESOURCE_LIMIT` Initialization Parameter

- To enable or disable enforcement of resource limits, alter this parameter in the initialization file and restart the instance.
- A value of `TRUE` enables enforcement.
- A value of `FALSE` disables enforcement (default).
- Use this parameter to enable enforcement when the database can be shut down.

`ALTER SYSTEM` Command

- To enable or disable enforcement of resource limits for an instance, use the `ALTER SYSTEM` command.
- The setting specified using the `ALTER SYSTEM` command remains in effect until altered again or until the database is shut down.
- Use this command to enable or to disable enforcement when the database cannot be shut down.



Setting Resource Limits at Session Level

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

14-23

Guidelines

Profile limits can be enforced at the session level, the call level, or both. Session-level limits are enforced for each connection.

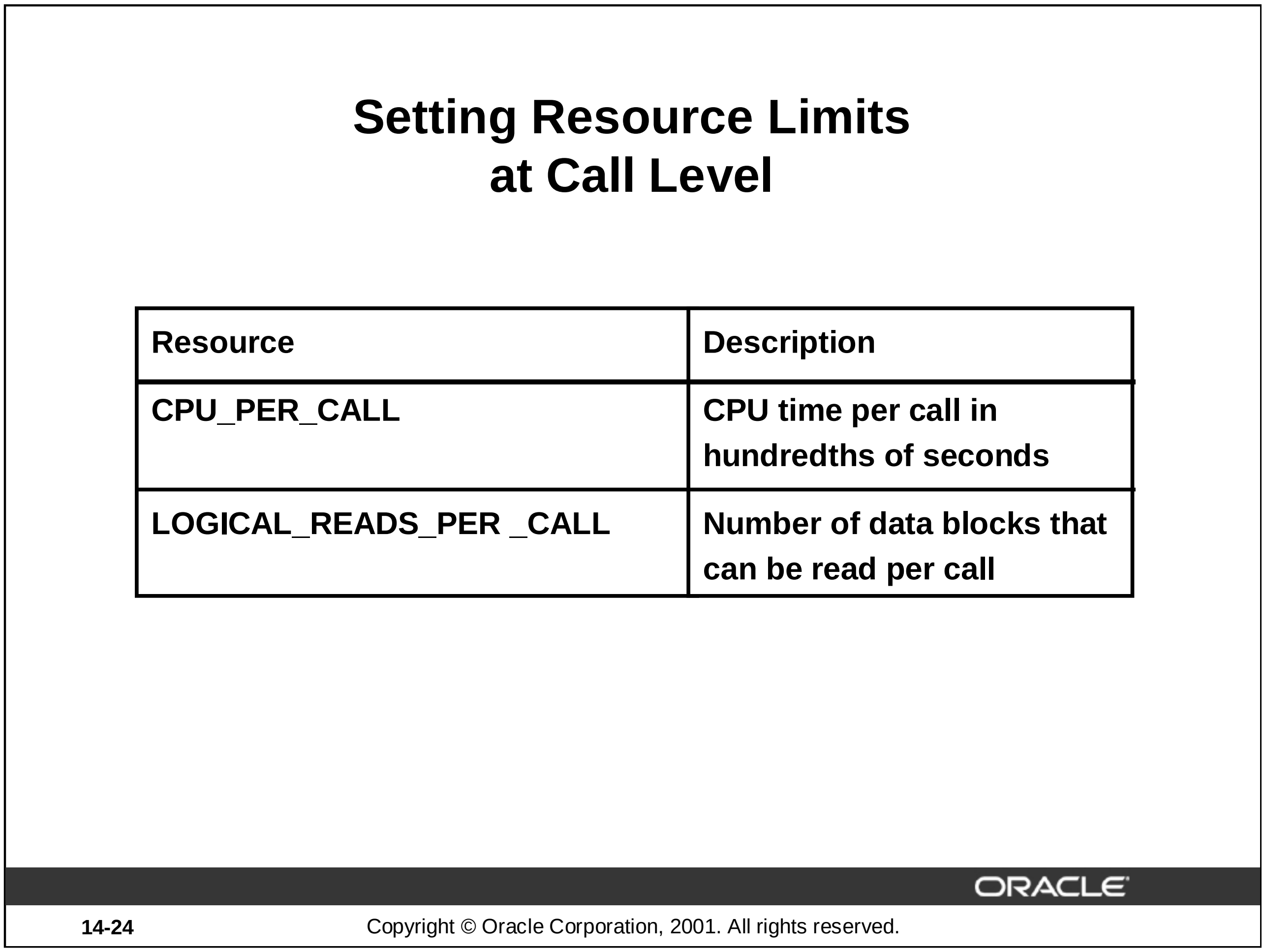
When a session-level limit is exceeded:

- An error message returns; for example:
  ORA-02391: exceeded simultaneous SESSION_PER_USER limit.
- The Oracle server disconnects the user.

Guidelines

- `IDLE_TIME` is calculated for the server process only. It does not take into account application activity. The `IDLE_TIME` limit is not affected by long-running queries and other operations.
- `LOGICAL_READS_PER_SESSION` is a limitation on the total number of reads from both memory and disk. This might be done to ensure that no I/O intensive statements can hoard memory and tie up the disk.
- `PRIVATE_SGA` applies only when running the shared server architecture and can be specified in M or K.

  Note: The shared server architecture is covered in detail in the *DBA Fundamentals II* course.



Setting Resource Limits at Call Level

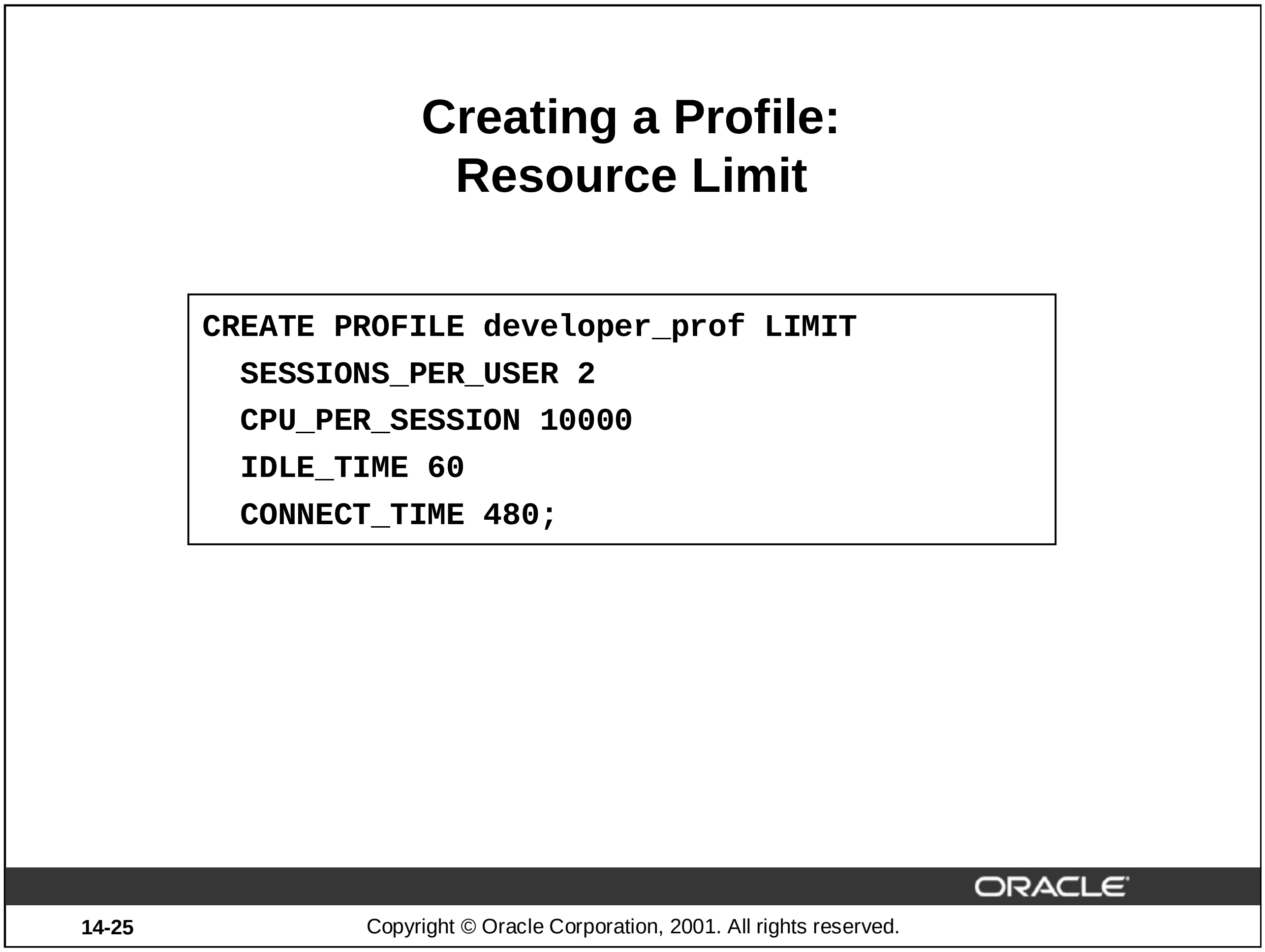
Resource	Description
CPU_PER_CALL	CPU time per call in hundredths of seconds
LOGICAL_READS_PER _CALL	Number of data blocks that can be read per call

Call-Level and Session-Level Limits

Call-level limits are enforced for each call made while executing a SQL statement.

When a call-level limit is exceeded:

- The processing of the statement is halted.
- The statement is rolled back.
- All previous statements remain intact.
- The user’s session remains connected.



Creating a Profile: Resource Limit

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

14-25

The CREATE PROFILE Command Syntax

Create a profile using the following CREATE PROFILE command:

```
CREATE PROFILE profile LIMIT
  [SESSIONS_PER_USER          max_value]
  [CPU_PER_SESSION            max_value]
  [CPU_PER_CALL               max_value]
  [CONNECT_TIME               max_value]
  [IDLE_TIME                  max_value]
  [LOGICAL_READS_PER_SESSION max_value]
  [LOGICAL_READS_PER_CALL     max_value]
  [COMPOSITE_LIMIT            max_value]
  [PRIVATE_SGA                max_bytes]
```

where:

profile is the name of the profile

max_value is an integer, UNLIMITED, or DEFAULT

max_bytes is an integer optionally followed by K or M UNLIMITED, or DEFAULT

The CREATE PROFILE Command Syntax (continued)

UNLIMITED: indicates that a user assigned this profile can use an unlimited amount of this resource

DEFAULT: indicates this profile is subject to the limit for this resource, as specified in the DEFAULT profile

COMPOSITE_LIMIT: limits the total resource cost for a session expressed in service units; Oracle calculates the resource cost as a sum of:

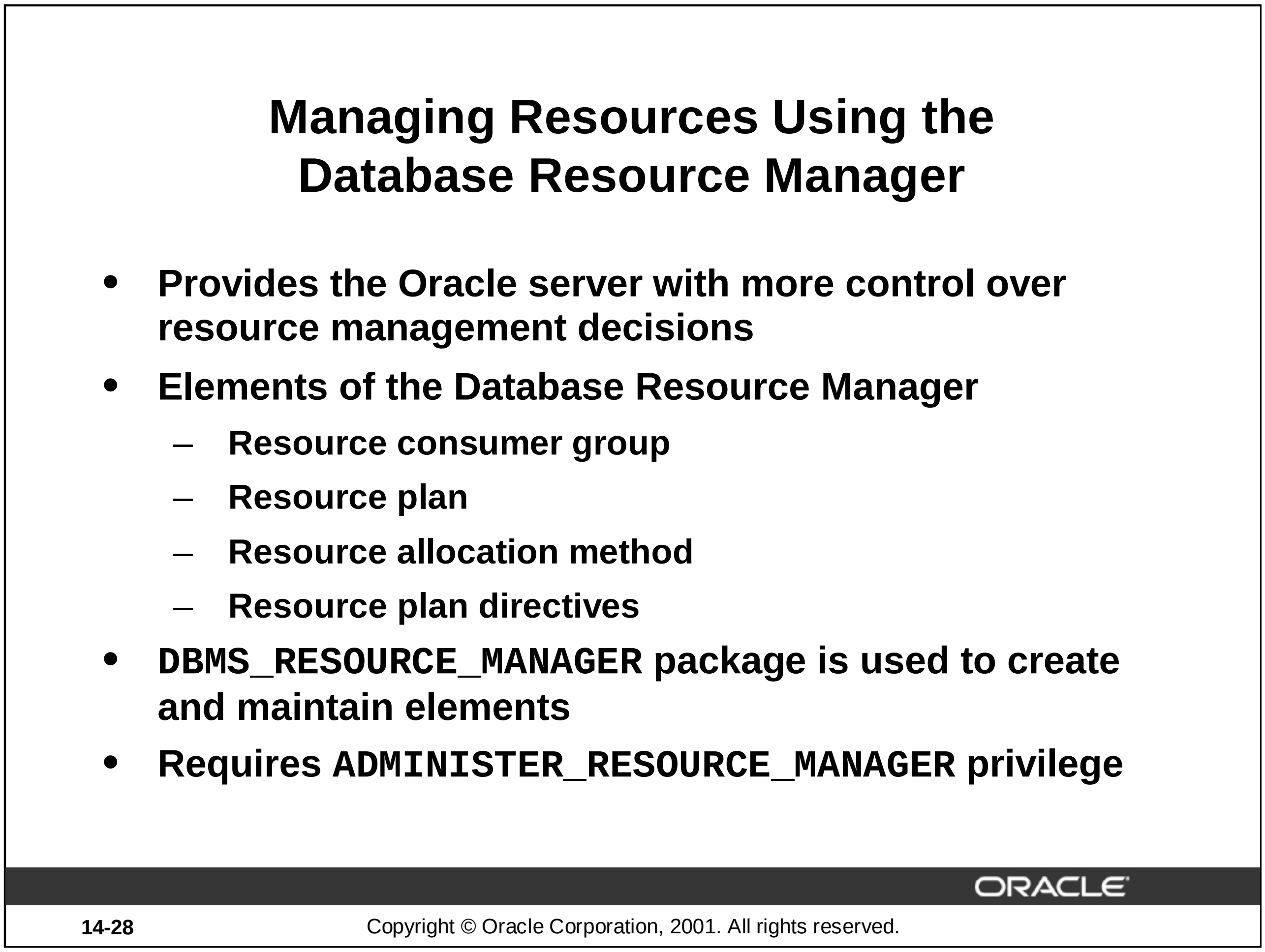
- CPU_PER_SESSION
- CONNECT_TIME
- LOGICAL_READS_PER_SESSION
- PRIVATE_SGA

The data dictionary view RESOURCE_COST provides resource limits assigned to different resources.

Note: Refer to the *Oracle9i SQL Reference* document for information on how to specify the weight for each session resource, ALTER RESOURCE COST command.

How to Use Oracle Enterprise Manger to Set Resource Limits

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone
2. Expand your working database from the databases folder
3. Expand the Security folder
4. Select the Profiles folder.
5. Select Create from the Object menu
6. Select Profile in the list and click Create.
7. Enter the resource parameters
8. Click Create



Managing Resources Using the Database Resource Manager

- **Provides the Oracle server with more control over resource management decisions**
- **Elements of the Database Resource Manager**
  - **Resource consumer group**
  - **Resource plan**
  - **Resource allocation method**
  - **Resource plan directives**
- **`DBMS_RESOURCE_MANAGER` package is used to create and maintain elements**
- **Requires `ADMINISTER_RESOURCE_MANAGER` privilege**

14-28

Managing Resources Using Database Resource Manager

The goal of the Database Resource Manager is to give the Oracle server more control over resource management decisions, thus circumventing problems resulting from inefficient operating system management.

Elements of the Database Resource Manager

Resource consumer group: Groups of users, or sessions, grouped together based on resource processing requirements.

Resource plan: Contains directives that specify how resources are allocated to resource consumer groups.

Resource allocation method: The method or policy used by the Database Resource Manager when allocating for a particular resource.

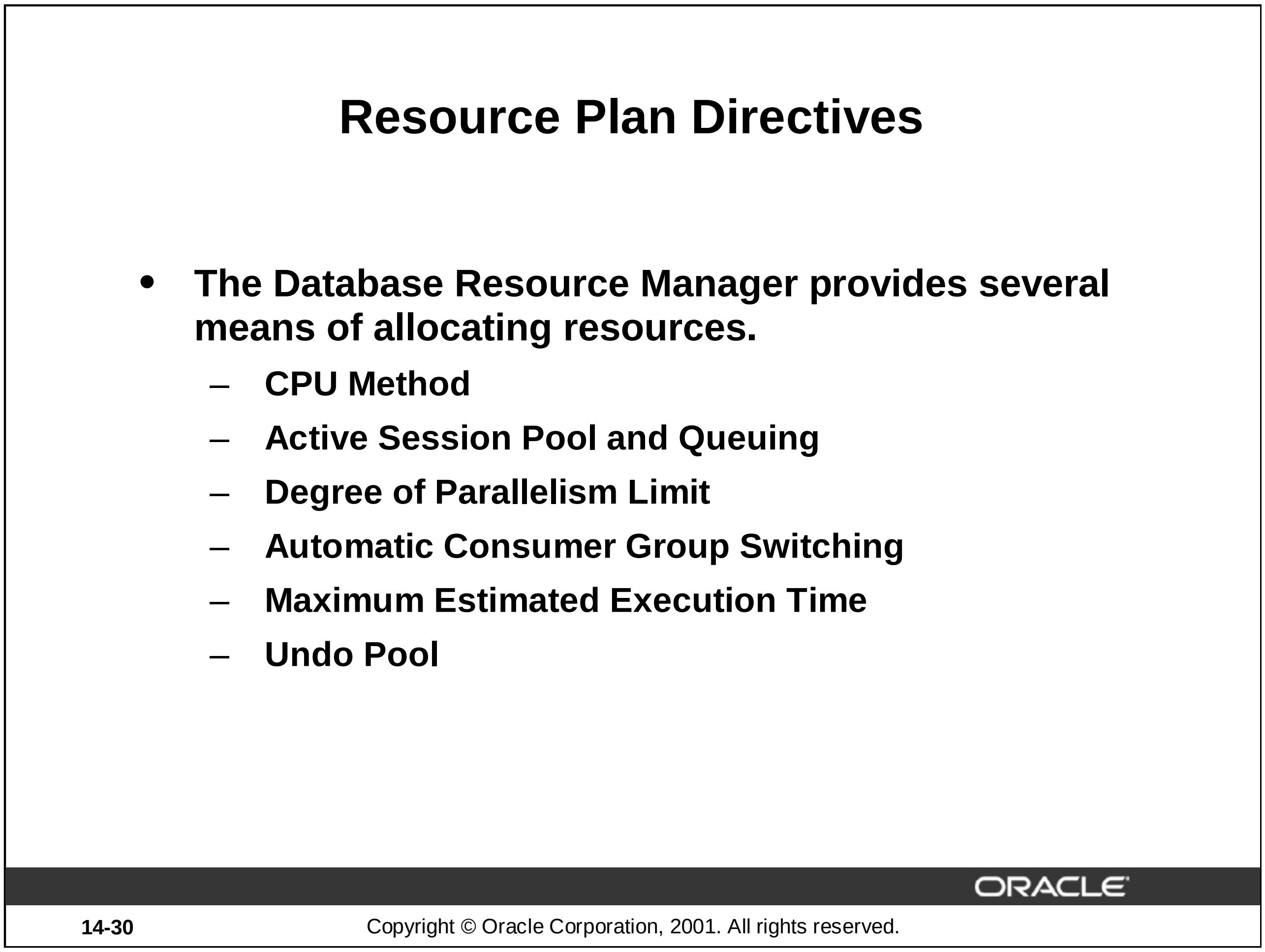
Resource plan directive: Used by administrators to associate resource consumer groups with particular plans and allocate resources among resource consumer groups.

Administering the Database Resource Manager

You must have the system privilege `ADMINISTER_RESOURCE_MANAGER` to administer the Database Resource Manager `(DBMS_RESOURCE_MANAGER).` Typically, DBAs will have this privilege with the ADMIN option as part of the DBA role.

Managing Resources Using the Database Resource Manager

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



Resource Plan Directives

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

14-30

Resource Plan Directives

CPU Method: Allows you to specify how CPU resources are to be allocated among consumer groups.

Active Session Pool with Queuing: You can control the maximum number of concurrently active sessions allowed within a consumer group. This maximum designates the active session pool. When the session cannot be initiated because the pool is full, the session is placed into a queue. When an active session completes, the first session in the queue is scheduled for execution. A timeout period can also be defined so that a job in the queue will timeout, causing it to abort with an error.

Degree of Parallelism Limit: Specifies a parallel degree limit for any operation within a consumer group.

Automatic Consumer Group Switching: Allows you to control resources by specifying criteria. If the criteria is not met, this would cause the automatic switching of sessions to another consumer group. The criteria used to determine switching are:

Switch group: Group being switched to.

Switch time: Switch time in seconds.

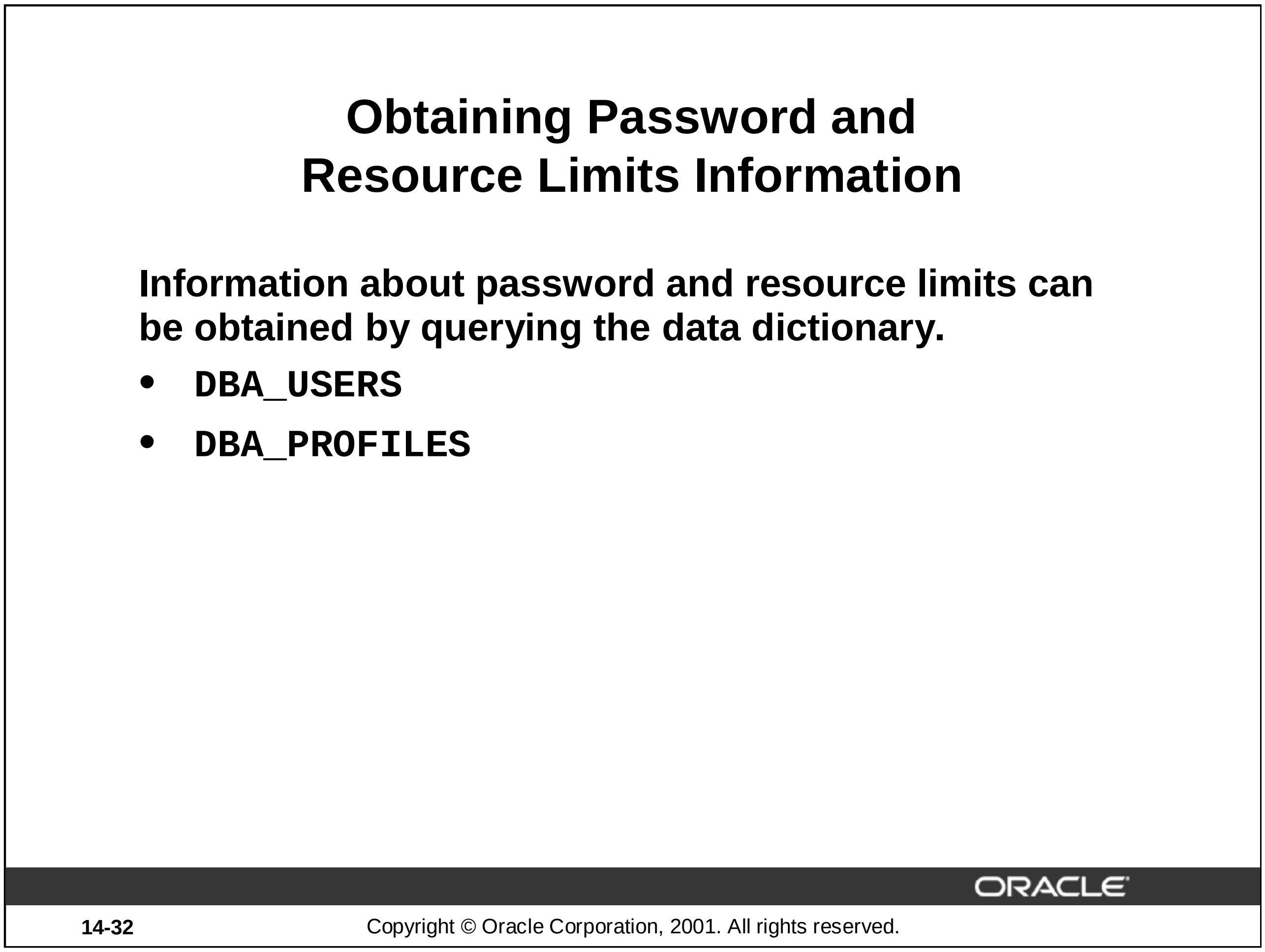
Switch estimate: Estimate of how long the operation will take to complete, which is used to decide whether to switch an operation even before it starts.

Resource Plan Directives (continued)

Maximum Estimated Execution Time: Estimates the execution time of an operation proactively. A DBA can define the maximum estimated execution time any operation can take at any given time by setting the resource plan directive parameter `MAX_ESTIMATED_EXEC_TIME`. If the operation's estimate is more than the MAX_ESTIMATED_EXEC_TIME defined, the operation will not start, therefore, eliminating the exceptionally large job that would utilize too much of the system resources.

Undo Pool: An undo pool for each consumer group can be specified to control the amount of total undo that can be generated by a consumer group. When a consumer group exceeds its limit, the current DML statement generating the redo is terminated. The undo pool is defined by the resource plan directive parameter called `UNDO_POOL`.

Note: The Database Resource Manager is covered further in the *Oracle9i Performance Tuning* course.



Obtaining Password and Resource Limits Information

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

- **`DBA_USERS`**
- **`DBA_PROFILES`**

14-32

Viewing User Information

Use DBA_USERS to obtain information about account status.

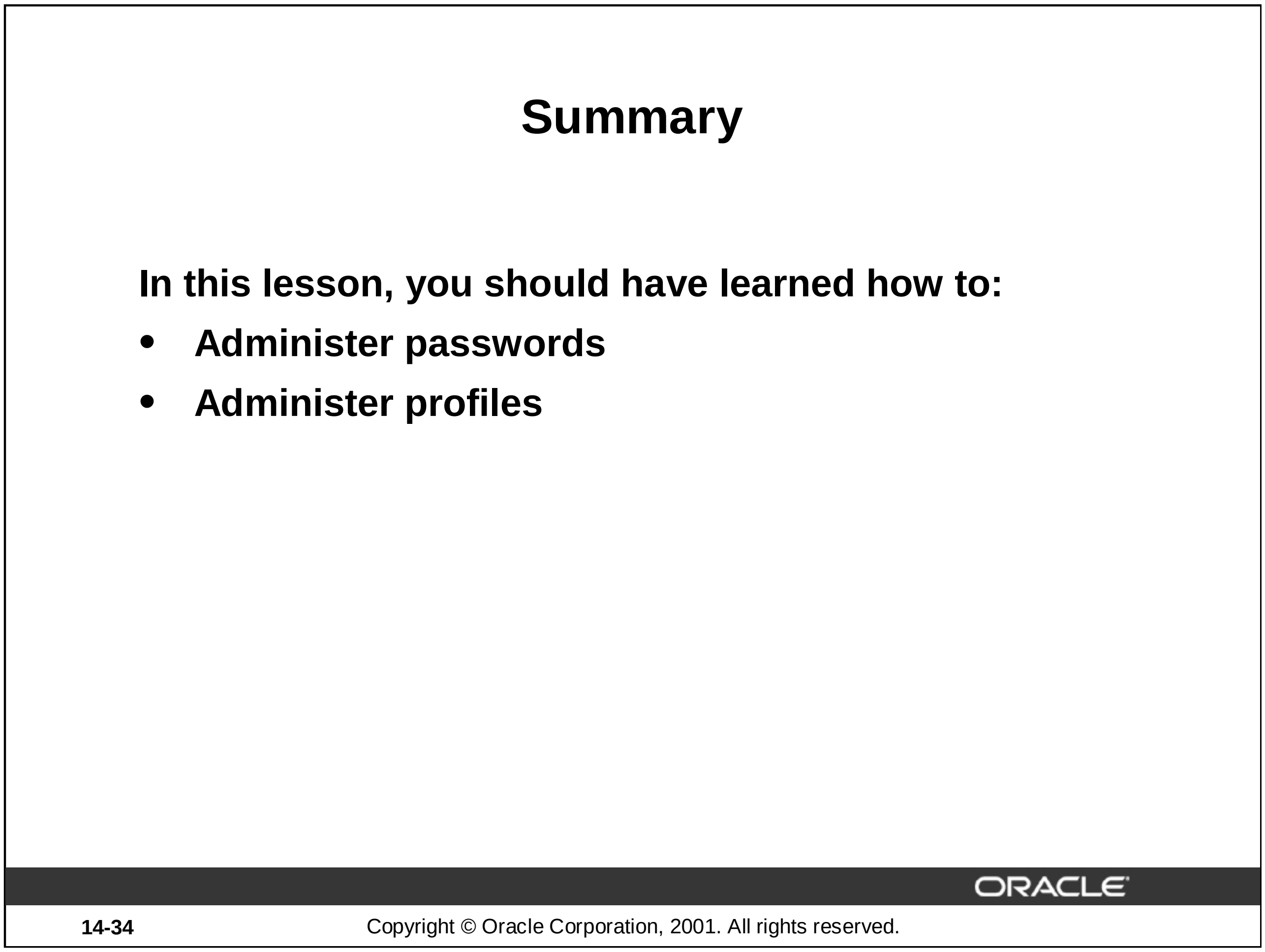
```
    SELECT username, password, account_status,
    FROM dba_users;
USERNAME  PASSWORD     ACCOUNT_STATUS
-------    --------     --------------
SYS       8A8F025737A9097A      OPEN
SYSTEM    D4DF7931AB130E37      OPEN
OUTLN     4A3BA55E08595C81      OPEN
DBSNMP    E066D214D5421CCC      OPEN
HR        BB69FBB77CFA6B9A      OPEN
OE        957C7EF29CC223FC      LOCKED
```

Viewing Profile Information

Query the DBA_PROFILES view to display password profile information:

```
SELECT * FROM dba_profiles
  WHERE resource_type='PASSWORD'
  AND profile=’GRACE_5’;

PROFILE RESOURCE_NAM             RESOURCE         LIMIT
--------------------             ---------        -------
GRACE_5 FAILED_LOGIN_ATTEMPTS    PASSWORD         3
GRACE_5 PASSWORD_LIFE_TIME       PASSWORD         30
GRACE_5 PASSWORD_REUSE_TIME      PASSWORD         30
GRACE_5 PASSWORD_REUSE_MAX       PASSWORD         UNLIMITED
GRACE_5 PASSWORD_VERIFY_FUNCTION PASSWORD         DEFAULT
GRACE_5 PASSWORD_LOCK_TIME       PASSWORD         UNLIMITED
GRACE_5 PASSWORD_GRACE_TIME      PASSWORD         5
```



Summary

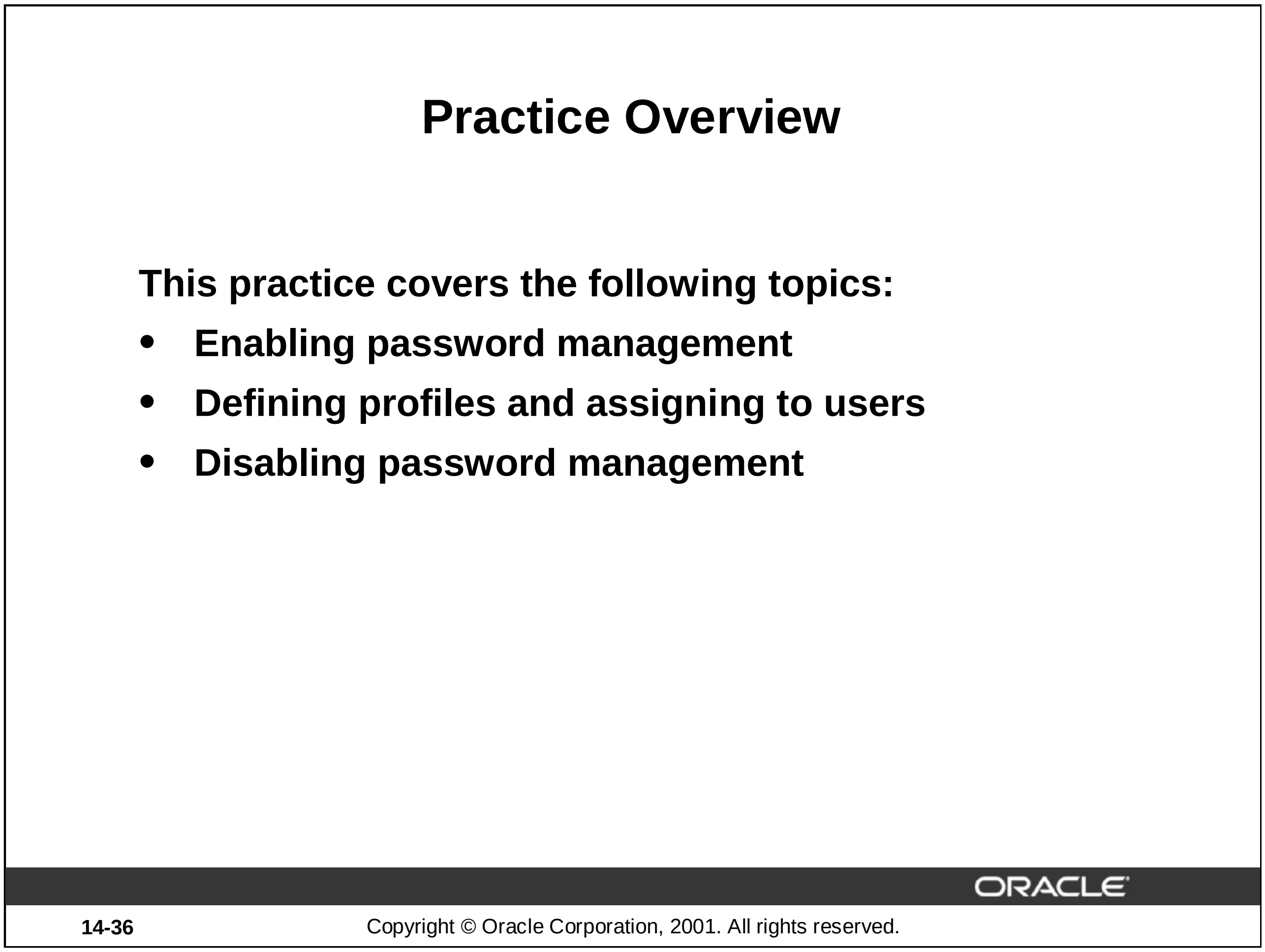
In this lesson, you should have learned how to:

- Administer passwords
- Administer profiles

14-34

Quick Reference

Context	Reference
Initialization parameters	RESOURCE_LIMIT
Dynamic performance views	None
Data dictionary view	DBA_PROFILES DBA_USERS
Commands	CREATE PROFILE ALTER PROFILE DROP PROFILE ALTER USER
Stored procedures and functions	VERIFY_FUNCTION



Practice Overview

This practice covers the following topics:

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

Practice 14: Managing Password Security and Resources

1. Run the `lab14_01.sql` script to create user Jeff. Enable password management by running script `@$ORACLE_HOME/rdbms/admin/utlpwdmg.sql`.

2. Try to change the password of user Jeff to `Jeff`. What happens?

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.

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.

5. 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?

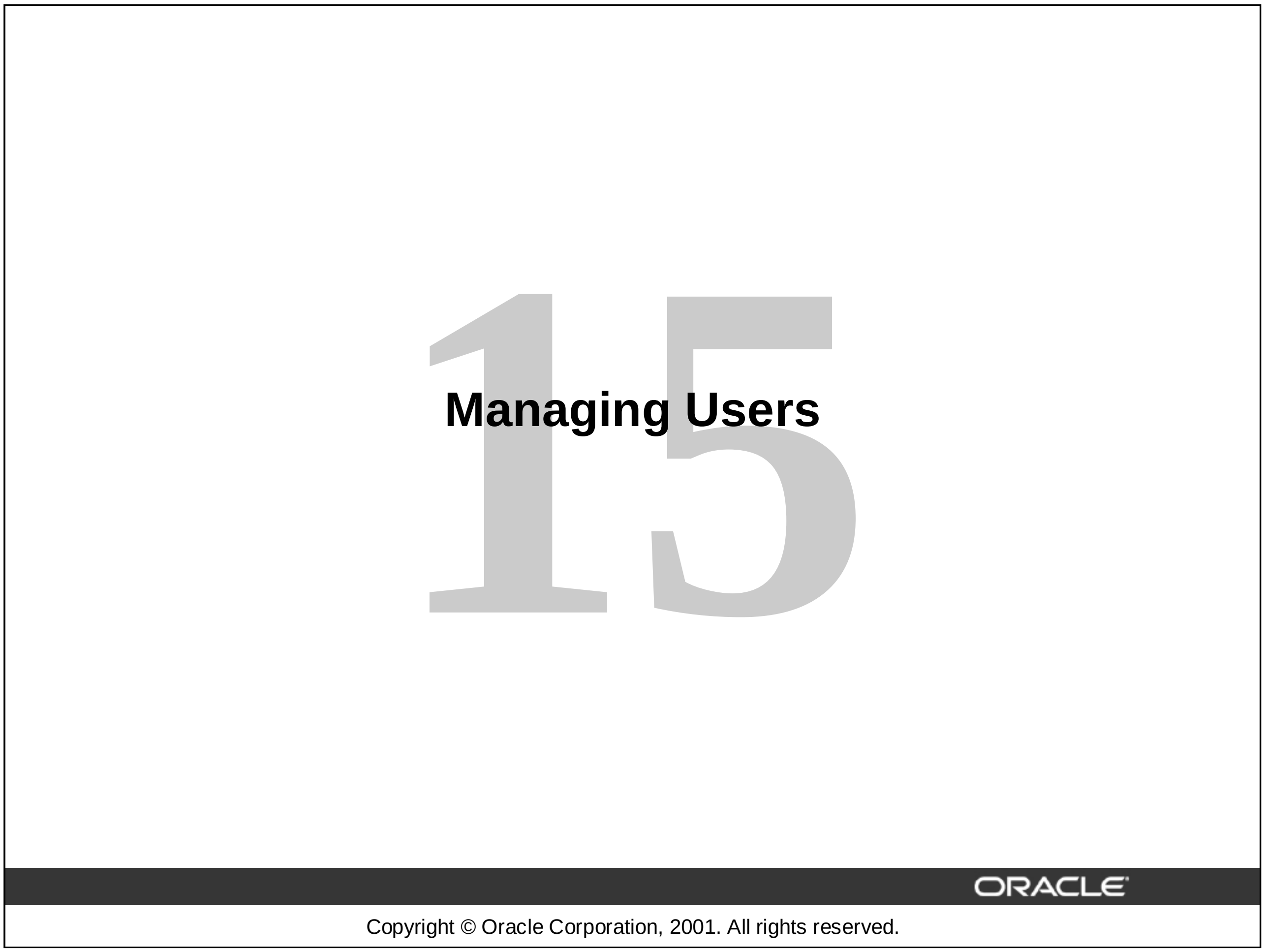
6. Using data dictionary view DBA_USERS verify user Jeff is locked. Unlock the account for the user Jeff. After unlocking user Jeff connect as Jeff.

   Hint: Execute the `ALTER USER` command to unlock the account.

7. Disable password checks for the `DEFAULT` profile.

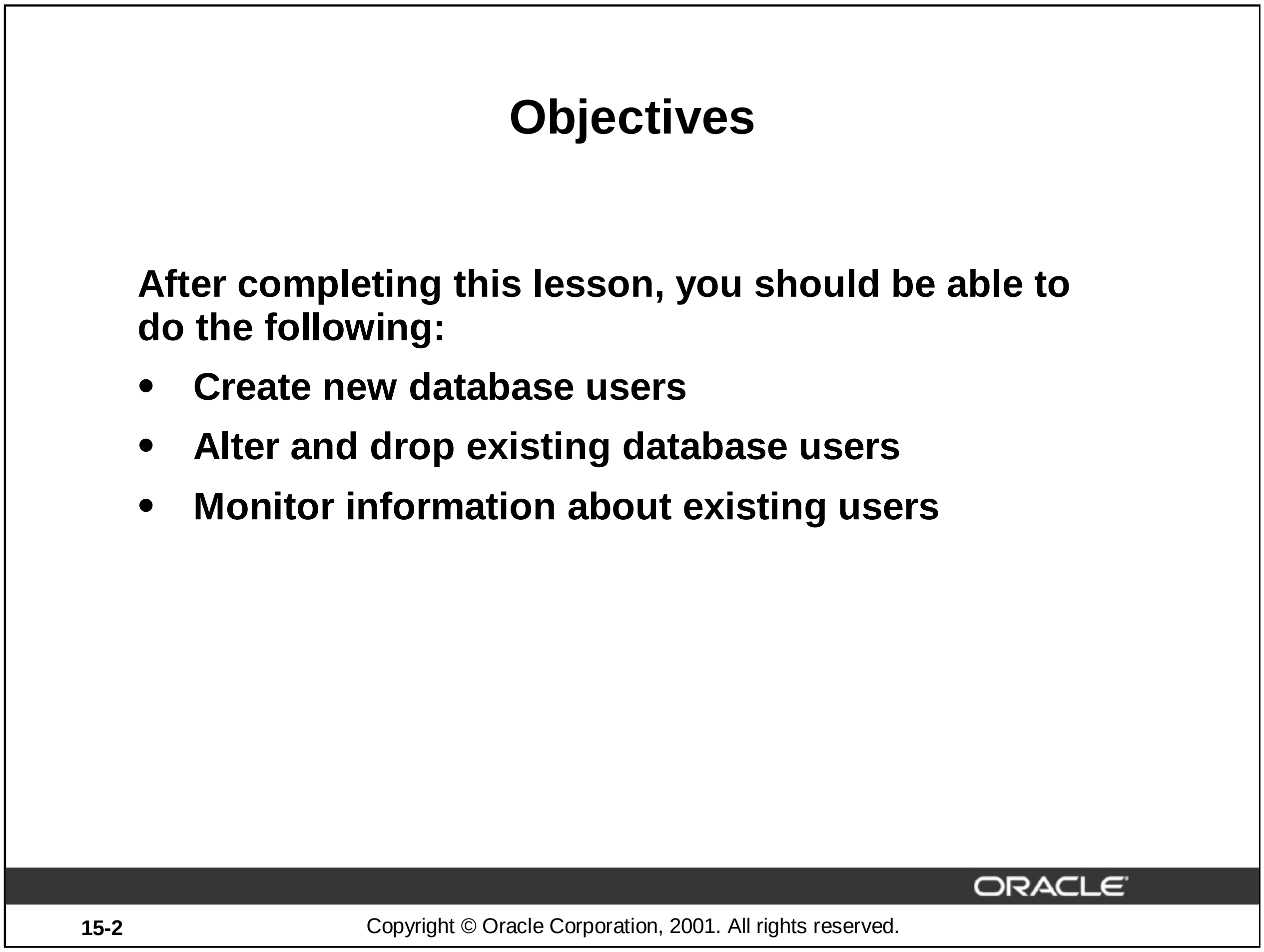
   Hint: Execute the `ALTER PROFILE` command to disable the password checks.

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?



15

Managing Users

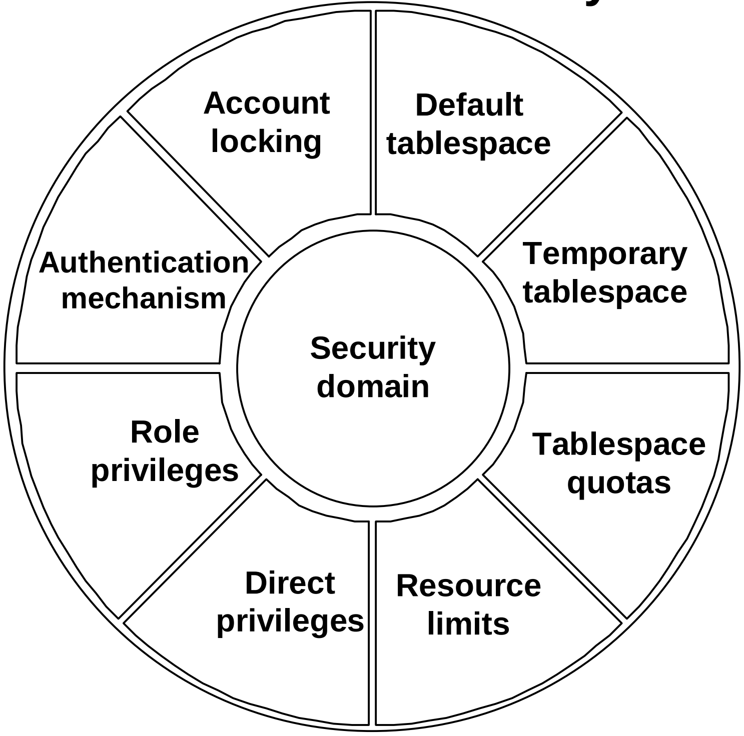


Objectives

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

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

Users and Security



Security domain

Account locking

Default tablespace

Temporary tablespace

Tablespace quotas

Resource limits

Direct privileges

Role privileges

Authentication mechanism

Security Domain

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

Authentication Mechanism

A user who needs 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 Getting Started with Oracle lesson for details regarding operating system authentication using roles.

Authentication through the network is covered in the course *DBA Fundamentals II*.

Tablespace Quotas

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

Default Tablespace

The default tablespace defines the location where segments 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 or unlock accounts manually.

Resource Limits

Limits can be placed on the use of resources such as CPU time, logical I/O, and the number of sessions opened by a user.

Direct Privileges

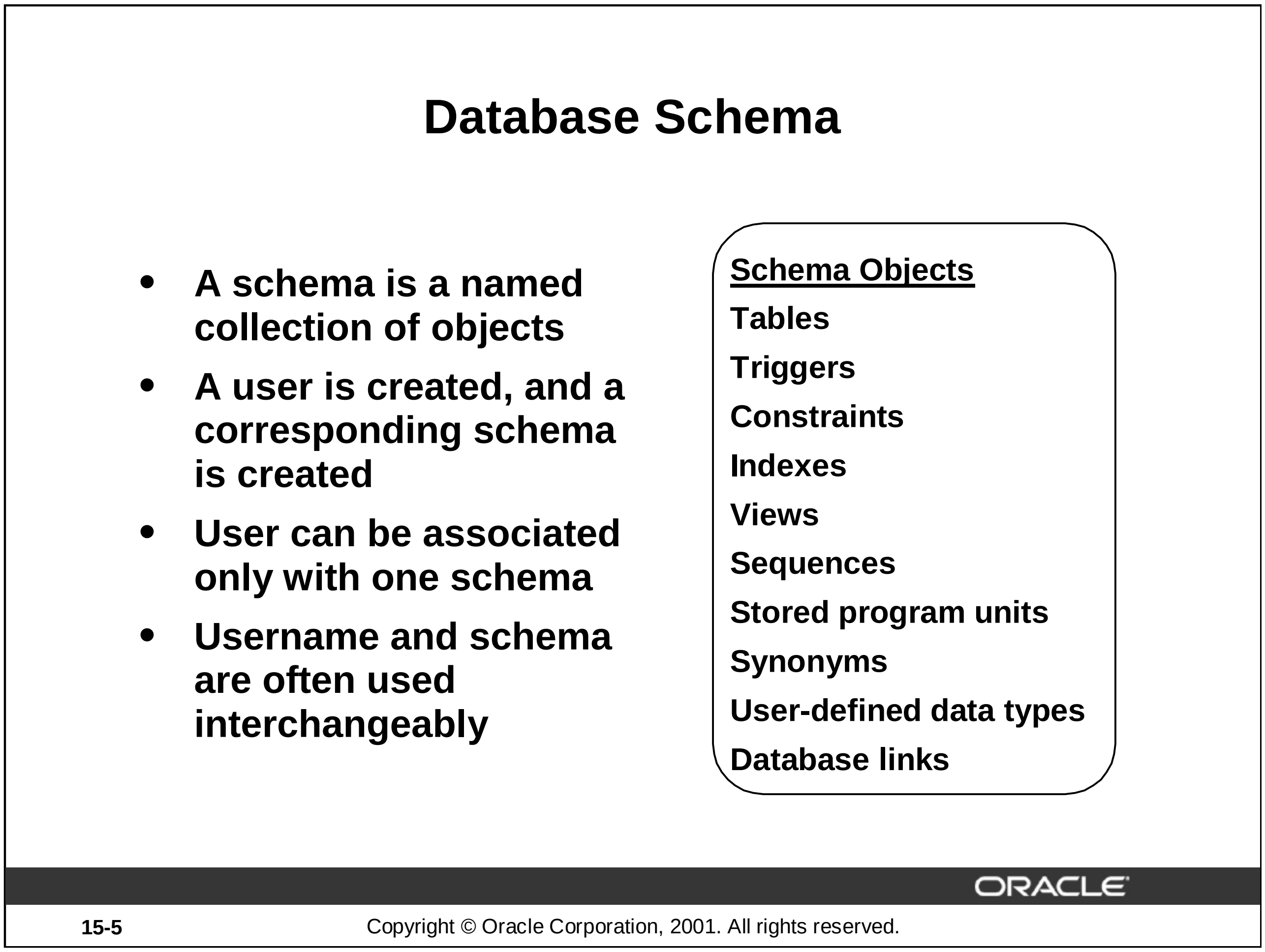
Privileges are used to control the actions a user can perform in a database.

Role Privileges

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

Note: Refer to the *Managing Privileges* and *Managing Roles* lessons 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 manually controlling account locking.



Database Schema

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

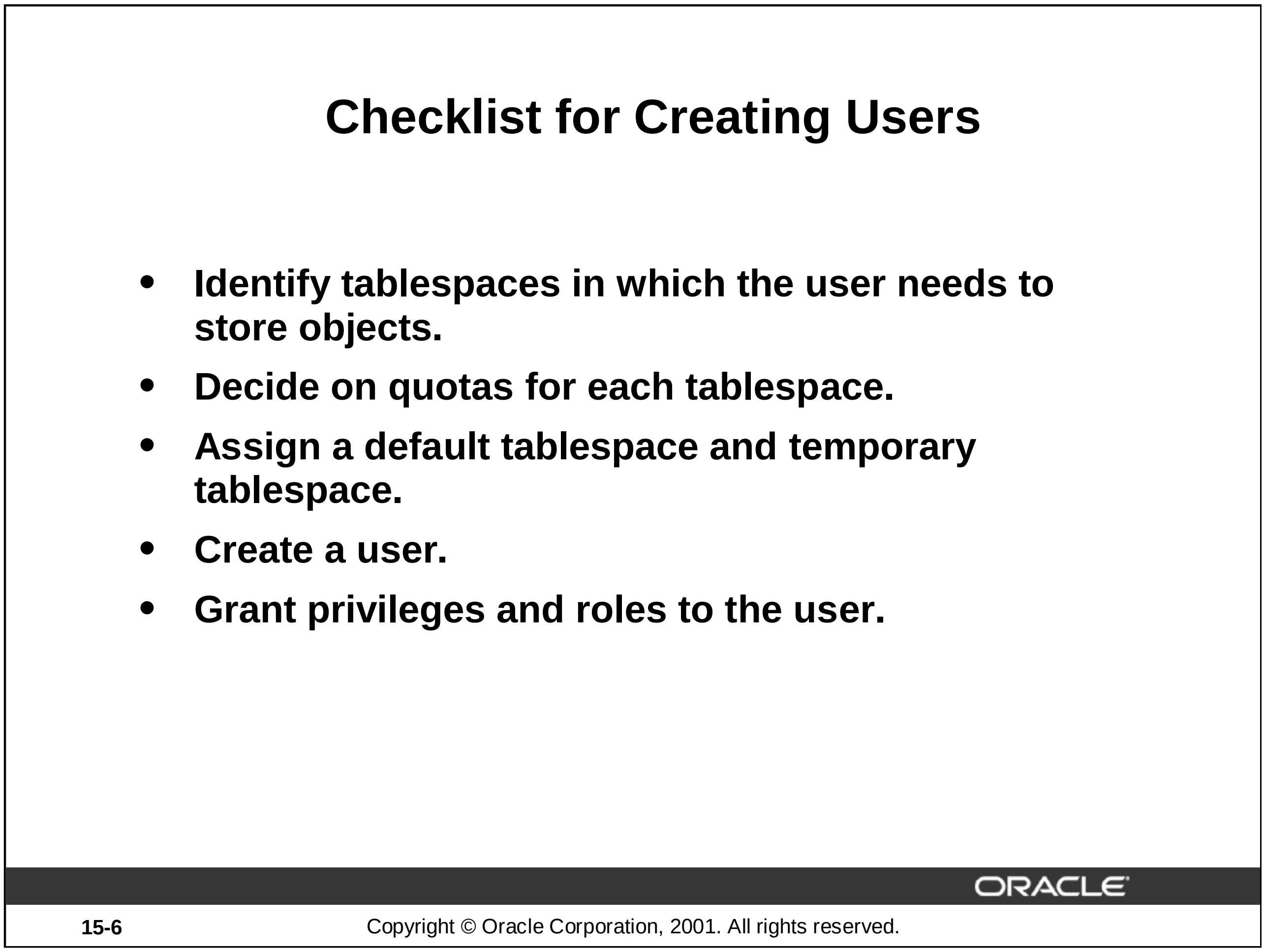
Schema Objects

Tables
Triggers
Constraints
Indexes
Views
Sequences
Stored program units
Synonyms
User-defined data types
Database links

What Is a Schema?

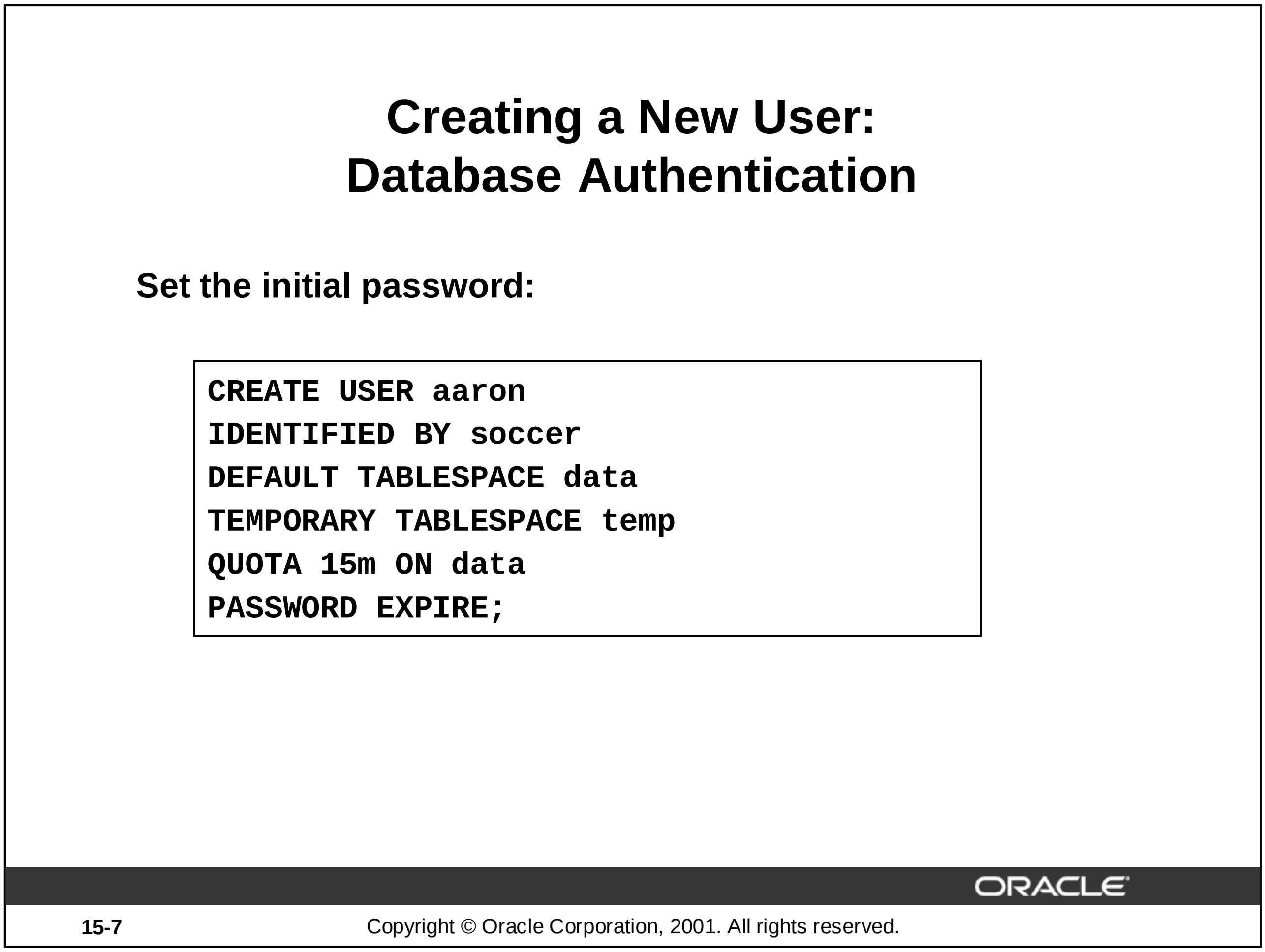
A schema is a named collection of objects such as tables, views, clusters, procedures, and packages associated with a particular user. When a database user is created, a corresponding schema with the same name is created for that user. A user can be associated only with a schema of the same name, and 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 needs to store objects.
- Decide on quotas for each tablespace.
- Assign a default tablespace and temporary tablespace.
- Create a user.
- Grant privileges and roles to the user.



Creating a New User: Database Authentication

Set the initial password:

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

Syntax

Use the following command to create a new user:

```
CREATE USER user
IDENTIFIED {BY password | EXTERNALLY}
[ 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 }]
```

Syntax (continued)

where:

`user:` is the name of the user

`BY password:` specifies user authenticated by the database and needs to supply *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 TEMPORARY TABLESPACE:` identifies the default or temporary tablespace for the user if a temporary tablespace has not been assigned to the user.

`QUOTA:` defines the maximum space allowed for objects owned by the user in the tablespace *tablespace (*Quota can be defined as *integer* bytes or kilobytes and megabytes. The keyword UNLIMITED 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 on 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.

Once the password expiry is set, when the user logs on using SQL Plus, the user receives the following message at logon, and is 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
```

How to Use Oracle Enterprise Manager to Create a New User

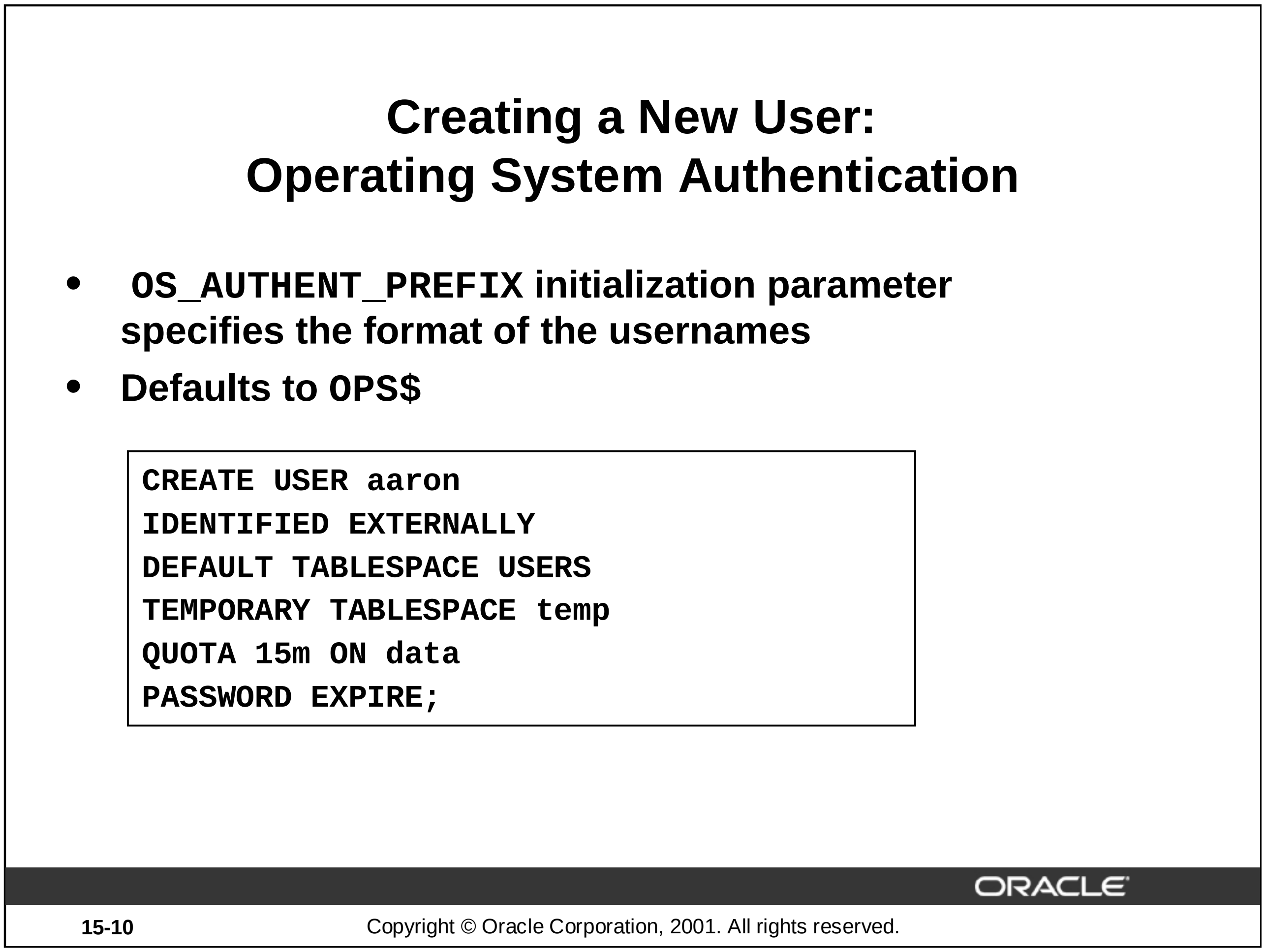
1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone

   You can also launch the Console from Windows NT Start menu.
2. Expand your working database from the databases folder
3. Expand the Security folder
4. Select the Users folder and select Create from the right mouse menu.
5. Enter user information in the General page of the property sheet.
6. Specify quotas using the Quotas page.
7. Click Create.

   You can also select a user and then select Object—>Create Like from the menu bar to create a user with the same quotas and privileges as an existing database user.

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

   Note: Refer to the *Managing Roles* lesson for information on the `CONNECT` role.



Creating a New User: Operating System Authentication

- `OS_AUTHENT_PREFIX` **initialization parameter specifies the format of the usernames**
- **Defaults to `OPS$`**

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

OS Authenitication

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 having to go through any validation by the Oracle server. Thus, to use SQL Plus to log on to the system UNIX user `aaron` needs to enter the following command from the operating system:

```
$ sqlplus /
```

Username for Operating System Authentication (continued)

Note

- Using `OS_AUTHENT_PREFIX=OPS$` gives 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 `OS_AUTHENT_PREFIX` may prevent these users from logging on to the database.

Changing User Quota on Tablespace

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

Modifying Tablespace Quotas

You may need to modify tablespace quotas in the following situations:

- When tables owned by a user exhibit unanticipated growth
- When an application is enhanced and requires additional tables or indexes
- When objects are reorganized and placed in different 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 ]
...]
```

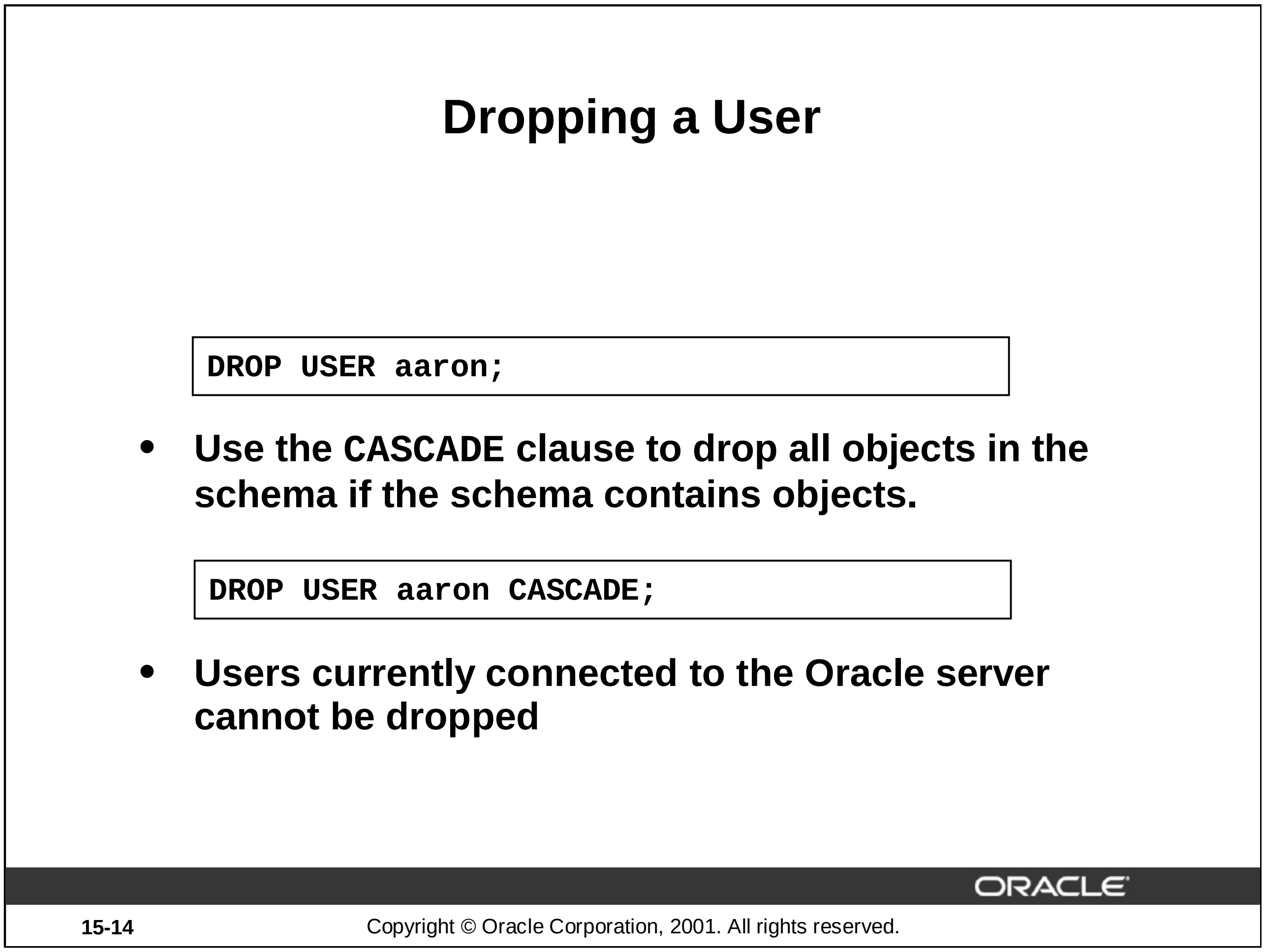
Once a quota of 0 is assigned, the objects owned by the user remain in the revoked tablespace, but they cannot be allocated any new space. For example, if a table that is 10 MB exists in tablespace USERS, and the tablespace USERS quota is altered to 0, no more new extents can be allocated for that table.

Any unchanged options remain unchanged.

Manager to Modify Tablespace Quota

1. Launch the Console
   - `%oemapp console`
   - Choose to Launch standalone

   You can also launch the Console from Windows NT Start menu
2. Expand your working database from the databases folder
3. Expand the Security folder
4. Expand the Users folder.
5. Select the username.
6. Enter the quota size in the Quota page of the property sheet
7. Click Apply.



Dropping a User

```
DROP USER aaron;
```

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

```
DROP USER aaron CASCADE;
```

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

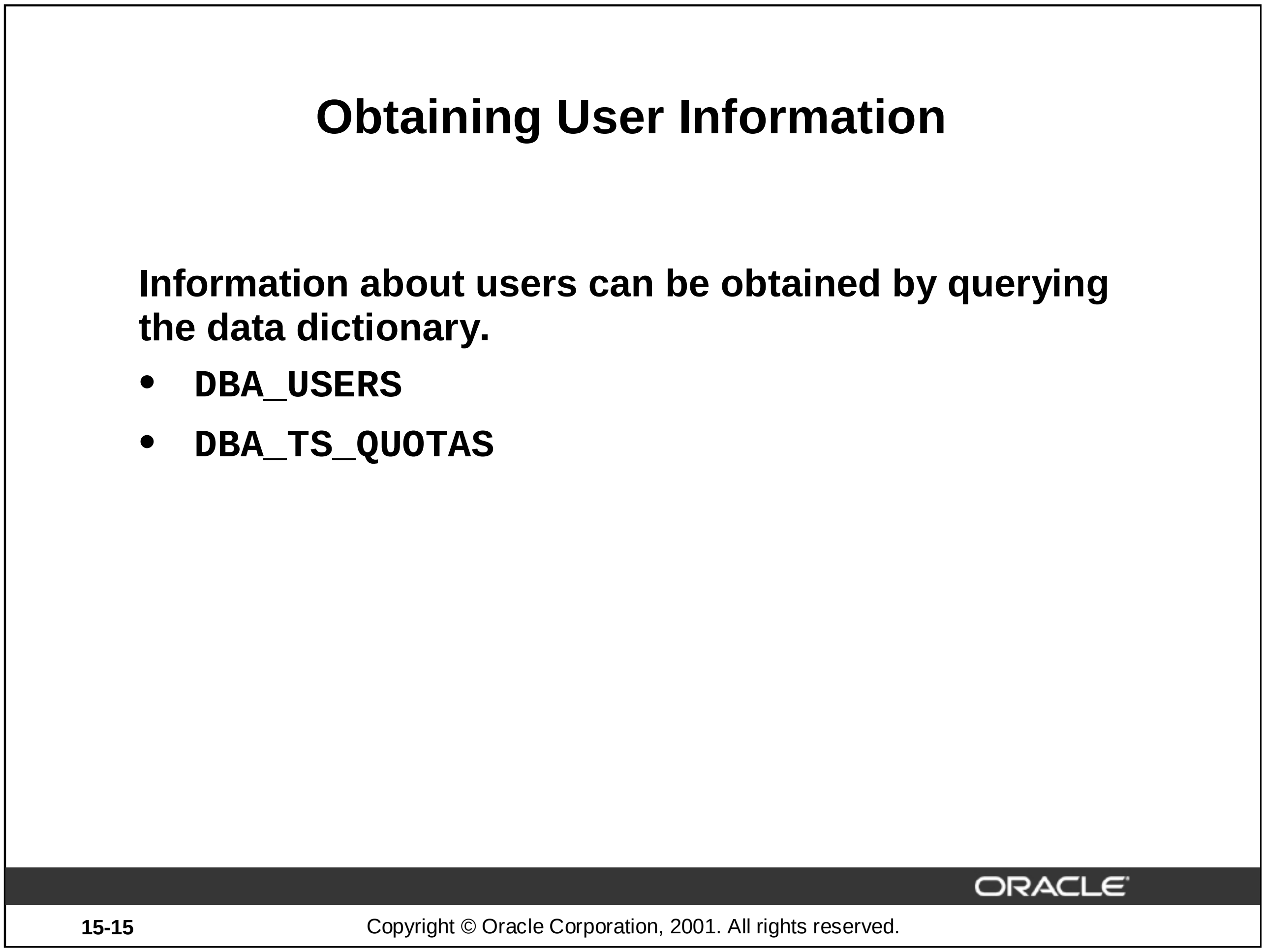
15-14

Syntax

```
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.



Obtaining User Information

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

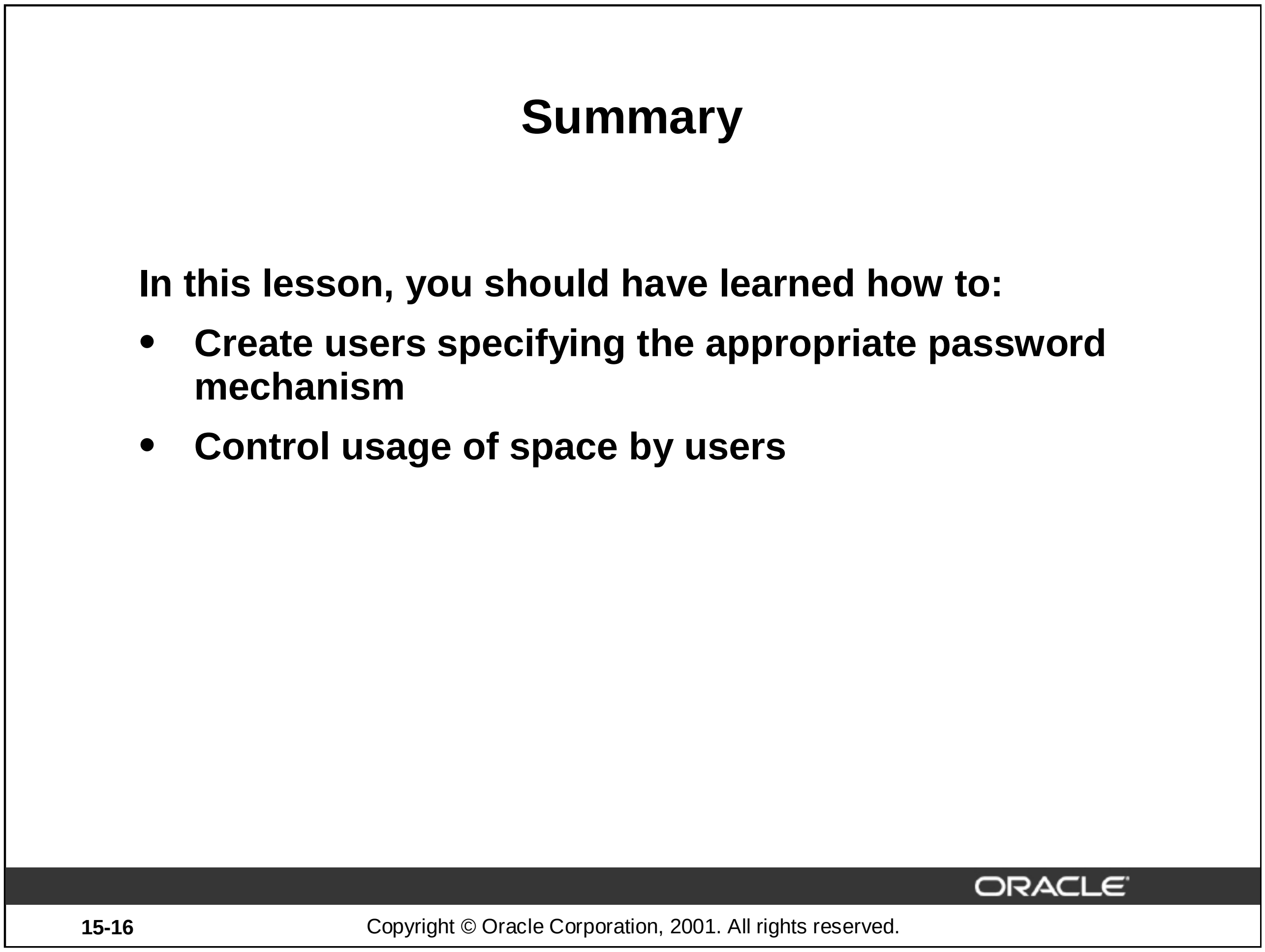
- **`DBA_USERS`**
- **`DBA_TS_QUOTAS`**

Tablespace Quotas

Use the following query to find the default_tablespace for all users.

```
SELECT username, default_tablespace
FROM dba_users;
```

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



Summary

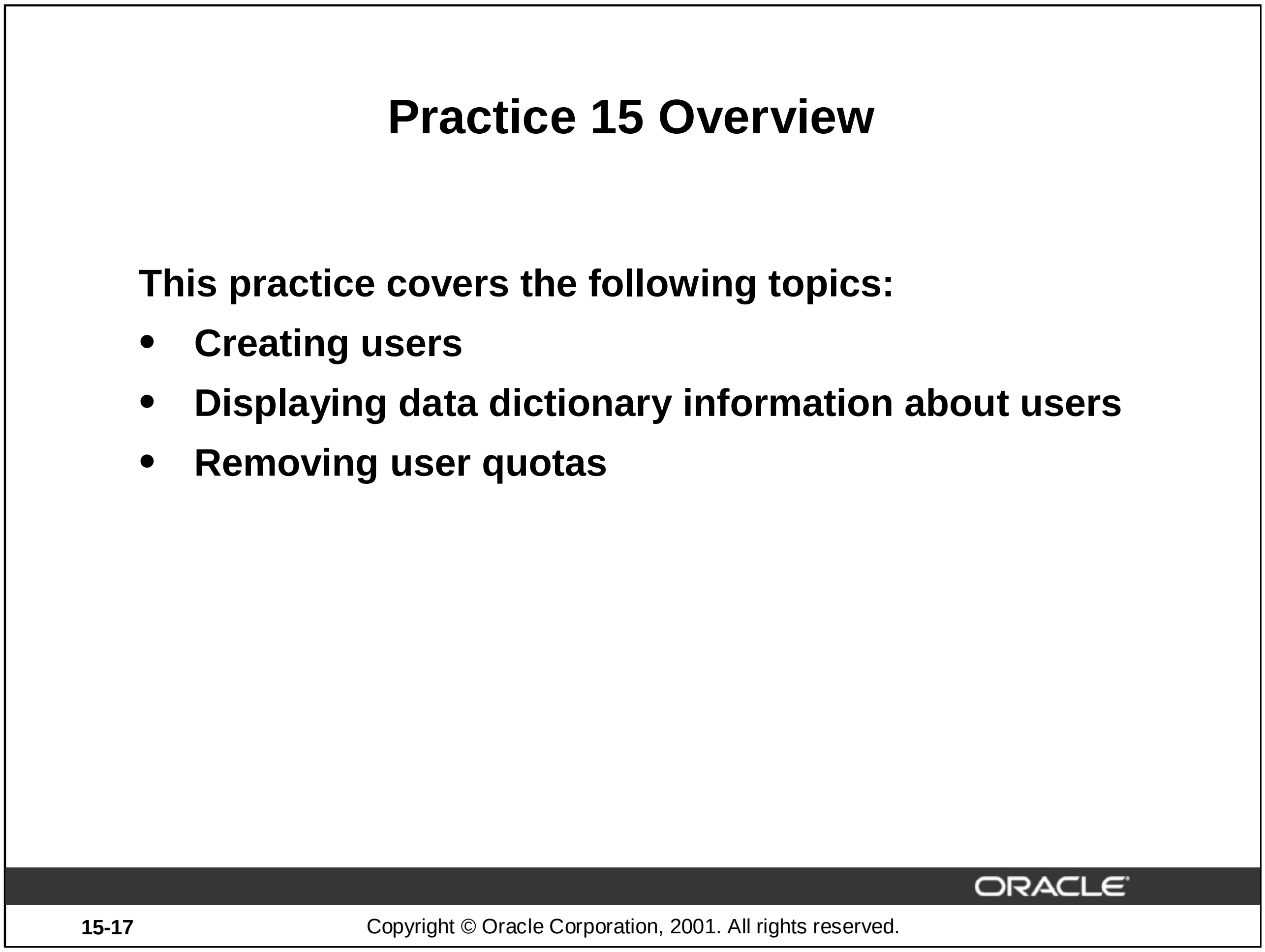
In this lesson, you should have learned how to:

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

15-16

Quick Reference

Context	Reference
Initialization parameters	OS_AUTHENT_PREFIX REMOTE_OS_AUTHENT
Data dictionary views	DBA_USERS DBA_TS_QUOTA
Commands	CREATE USER ALTER USER DROP USER



Practice 15 Overview

This practice covers the following topics:

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

15-17

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, ensure that Bob can log in and create objects up to one megabyte in size in USERS and INDX tablespaces.

   Hint: Ensure that the temporary tablespace temp is assigned. Use the lab15_01.sql script to grant Bob the ability to create sessions.

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? Why?

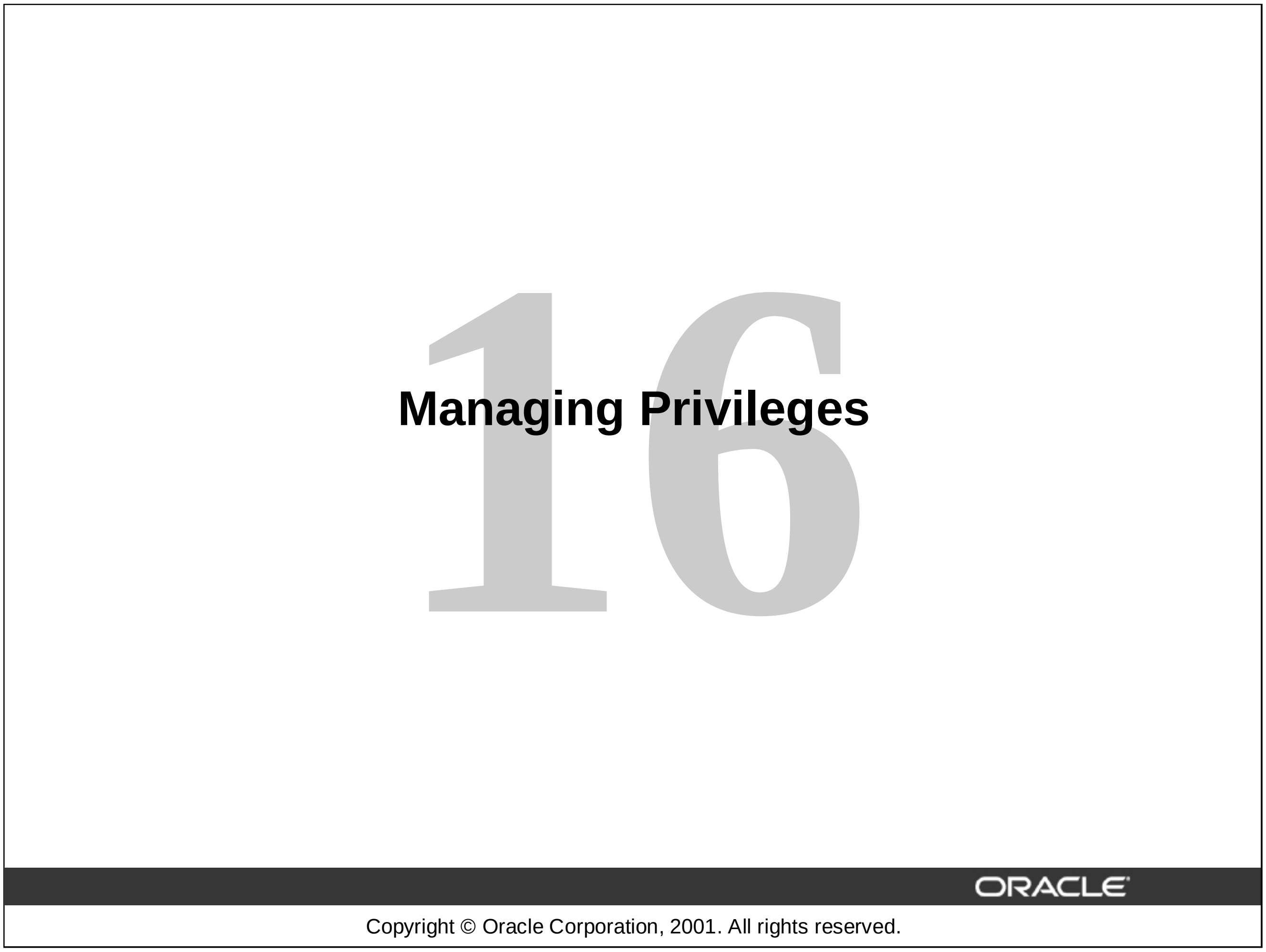
   b As Bob, change his password to SAM.

6. As SYSTEM, remove Bob's quota on his default tablespace.

7. Remove Emi's account from the database.

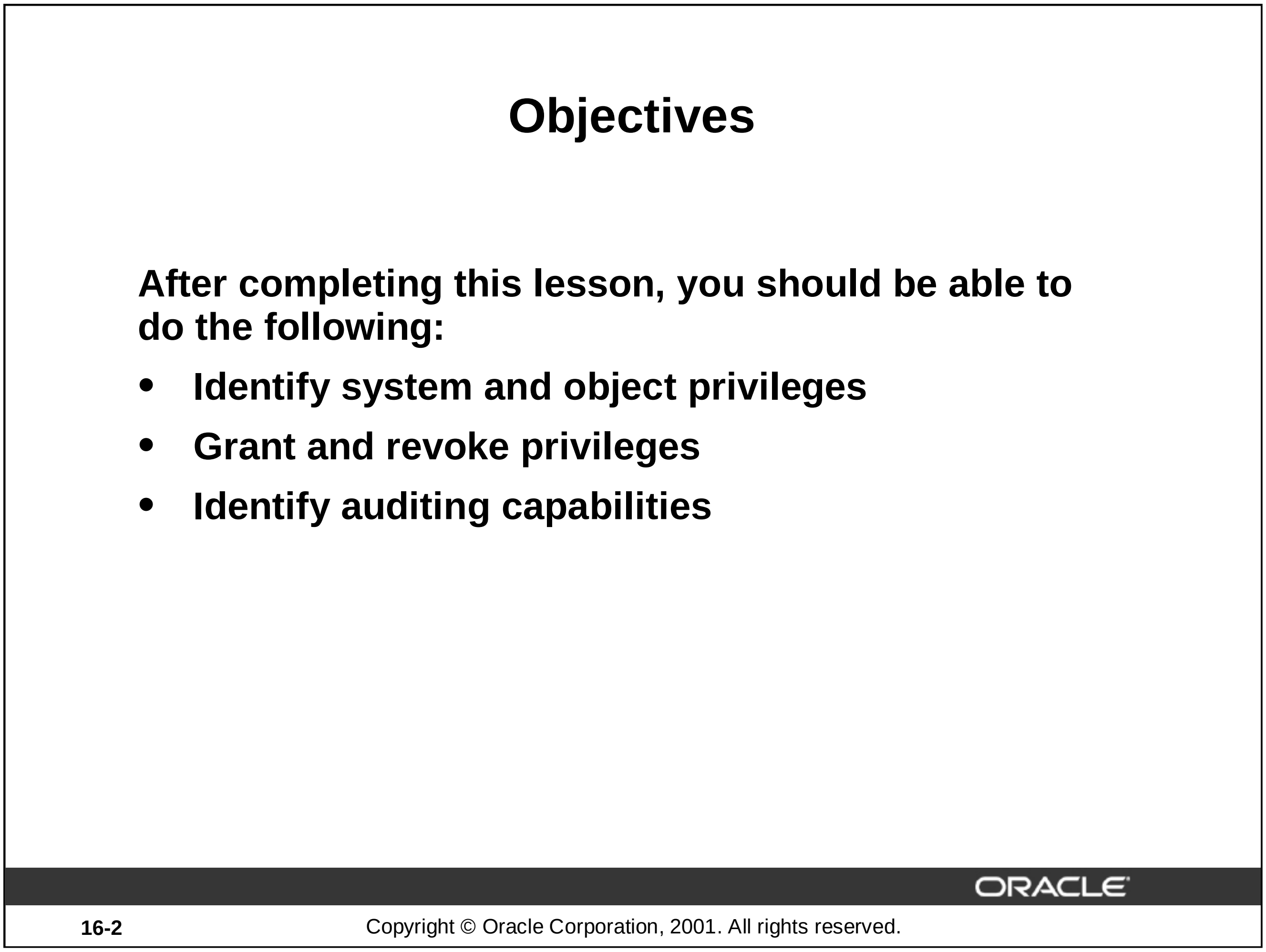
   Hint: Because Emi owns tables, you need to use the CASCADE option.

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

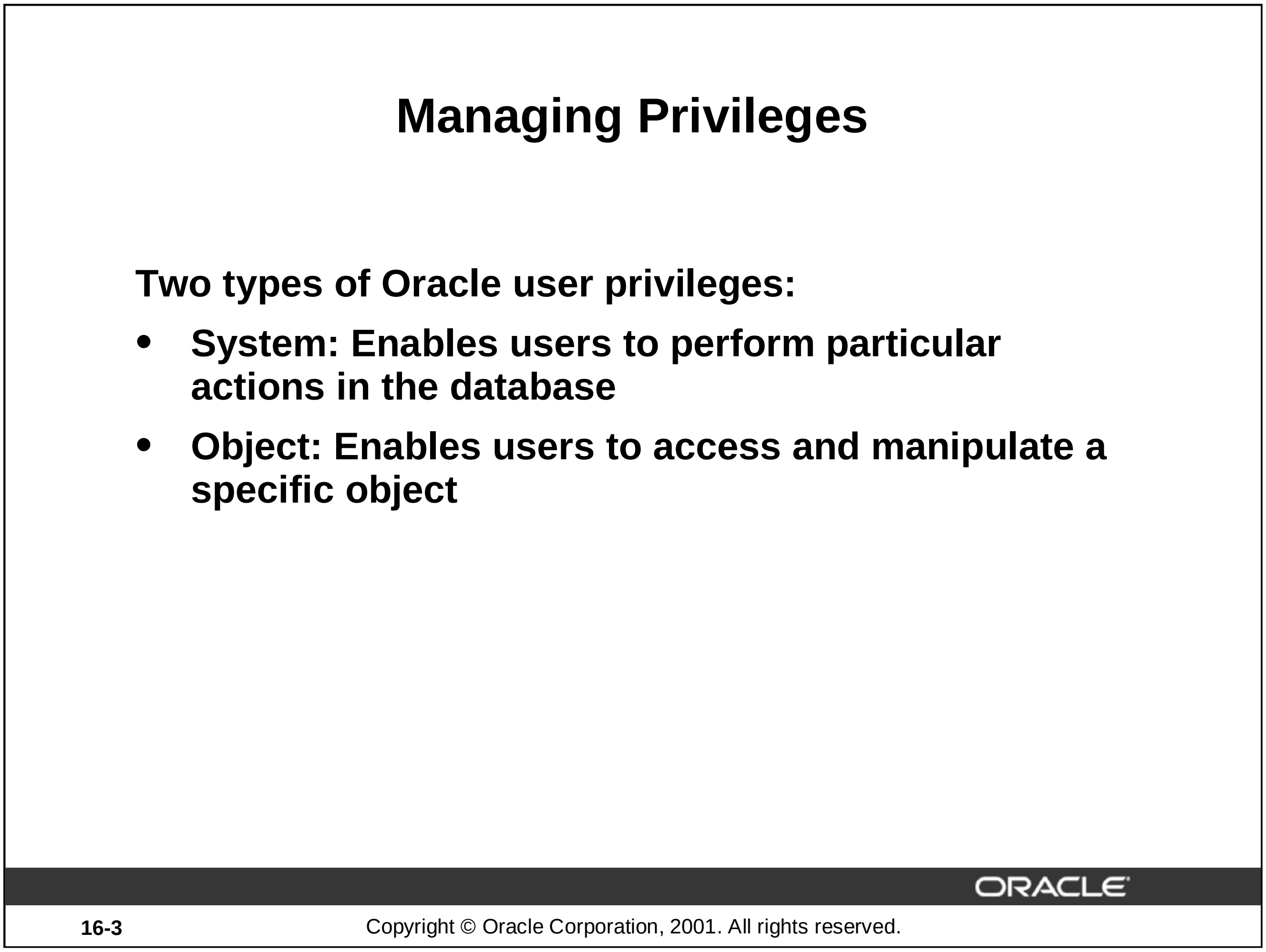
Managing Privileges



Objectives

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

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



Managing Privileges

Two types of Oracle user privileges:

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

16-3

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

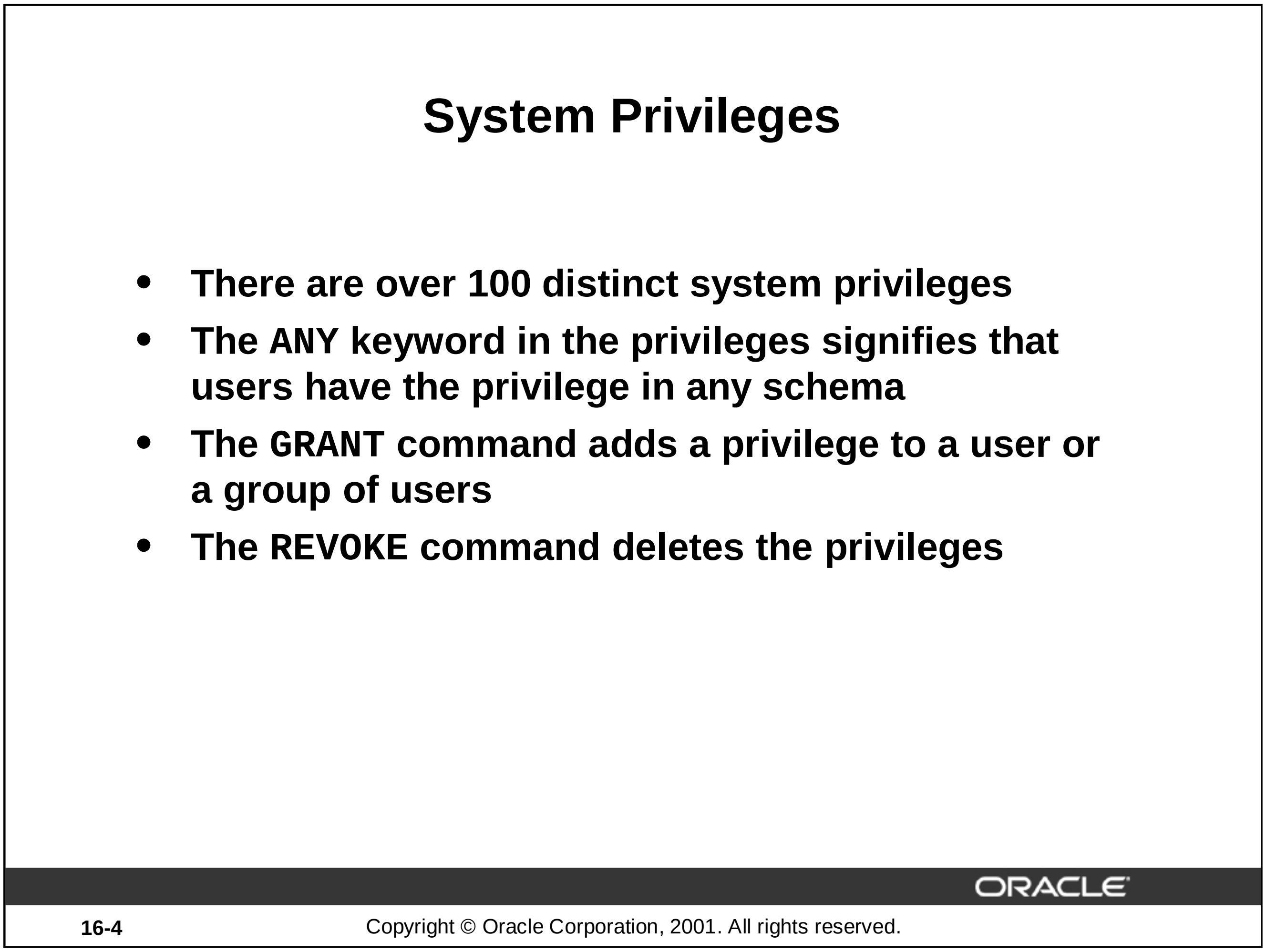
Each system privilege allows a user to perform a particular database operation or class of database operations. For example, the privilege to create tablespaces is a system privilege.

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

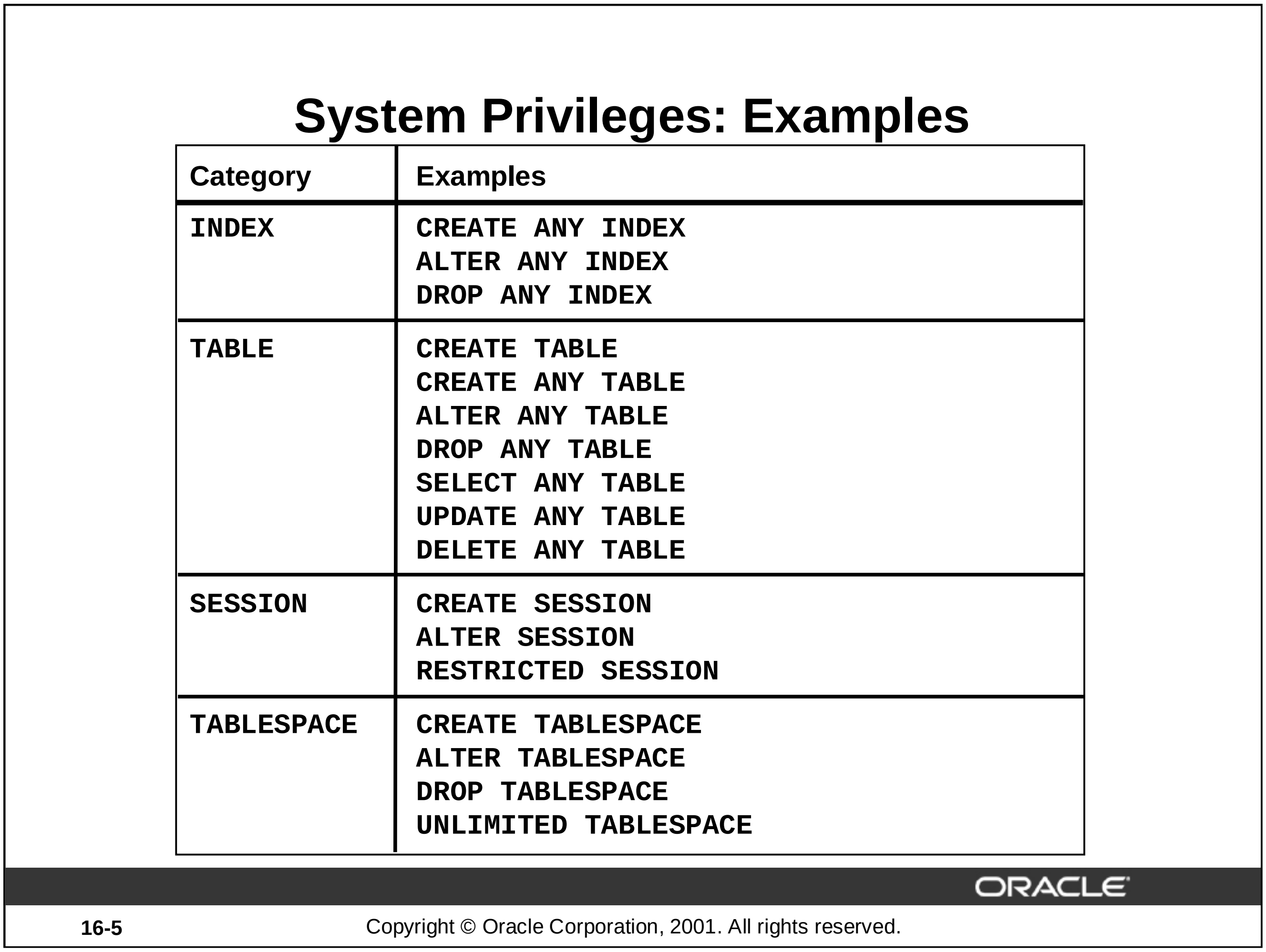
- There are over 100 distinct system privileges
- The `ANY` keyword in the privileges signifies that users have the privilege in any schema
- The `GRANT` command adds a privilege to a user or a group of users
- The `REVOKE` command deletes the privileges

System Privileges

The privileges can be classified as follows:

- Privileges enabling system wide operations; for example, `CREATE SESSION`, `CREATE TABLESPACE`
- Privileges enabling management of objects in a user's own schema; for example, `CREATE TABLE`
- Privileges enabling 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 Roles, see the lesson *Managing Roles*)



System Privileges: Examples

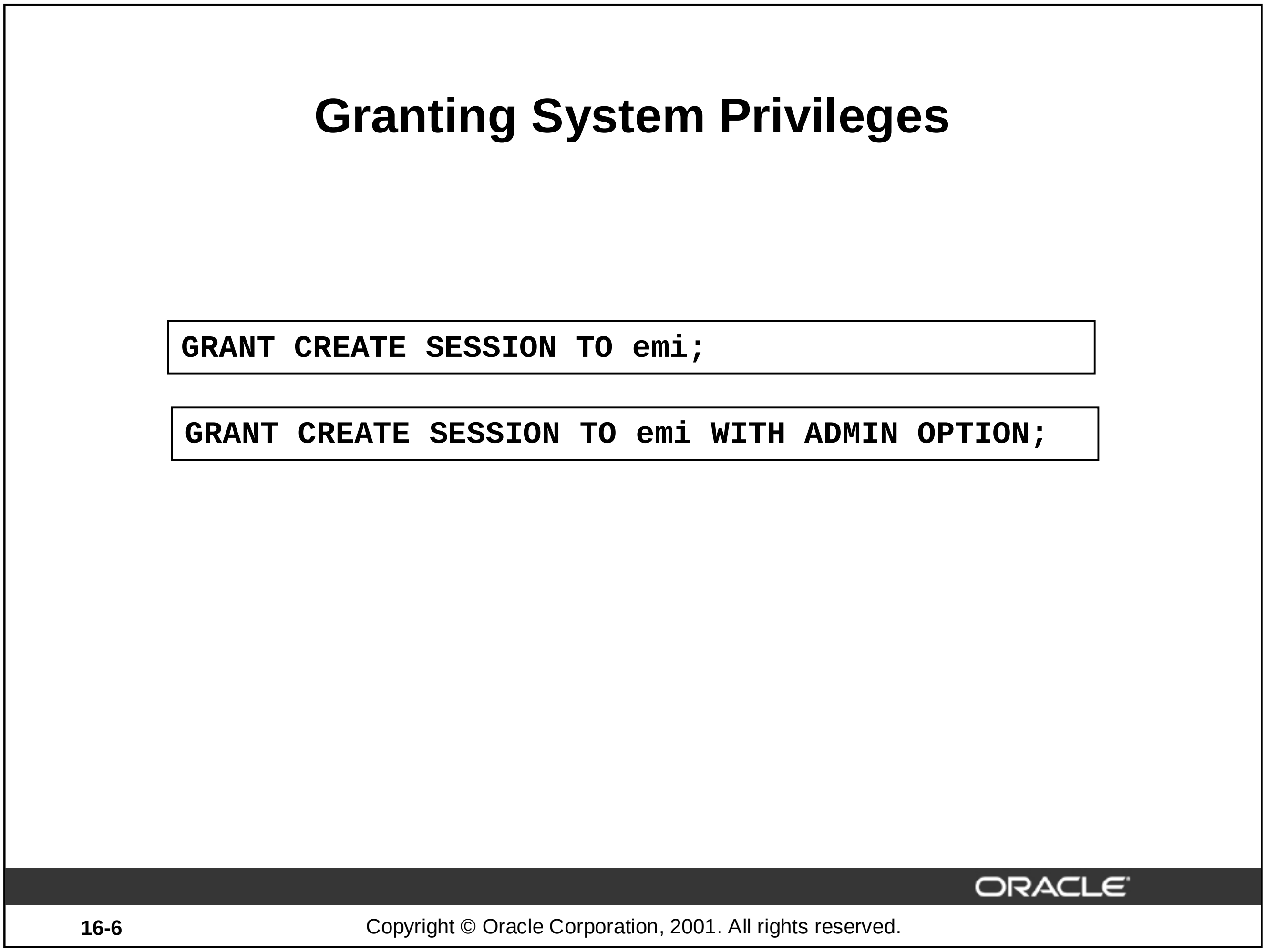
Category	Examples
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX

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

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 the dropping of these objects.
- UNLIMITED TABLESPACE cannot be granted to a role.
- For truncating a table in another schema, the DROP ANY TABLE privilege is necessary.



Granting System Privileges

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

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

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 administrator and rarely granted to other users.

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

Granting System Privileges (continued)

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

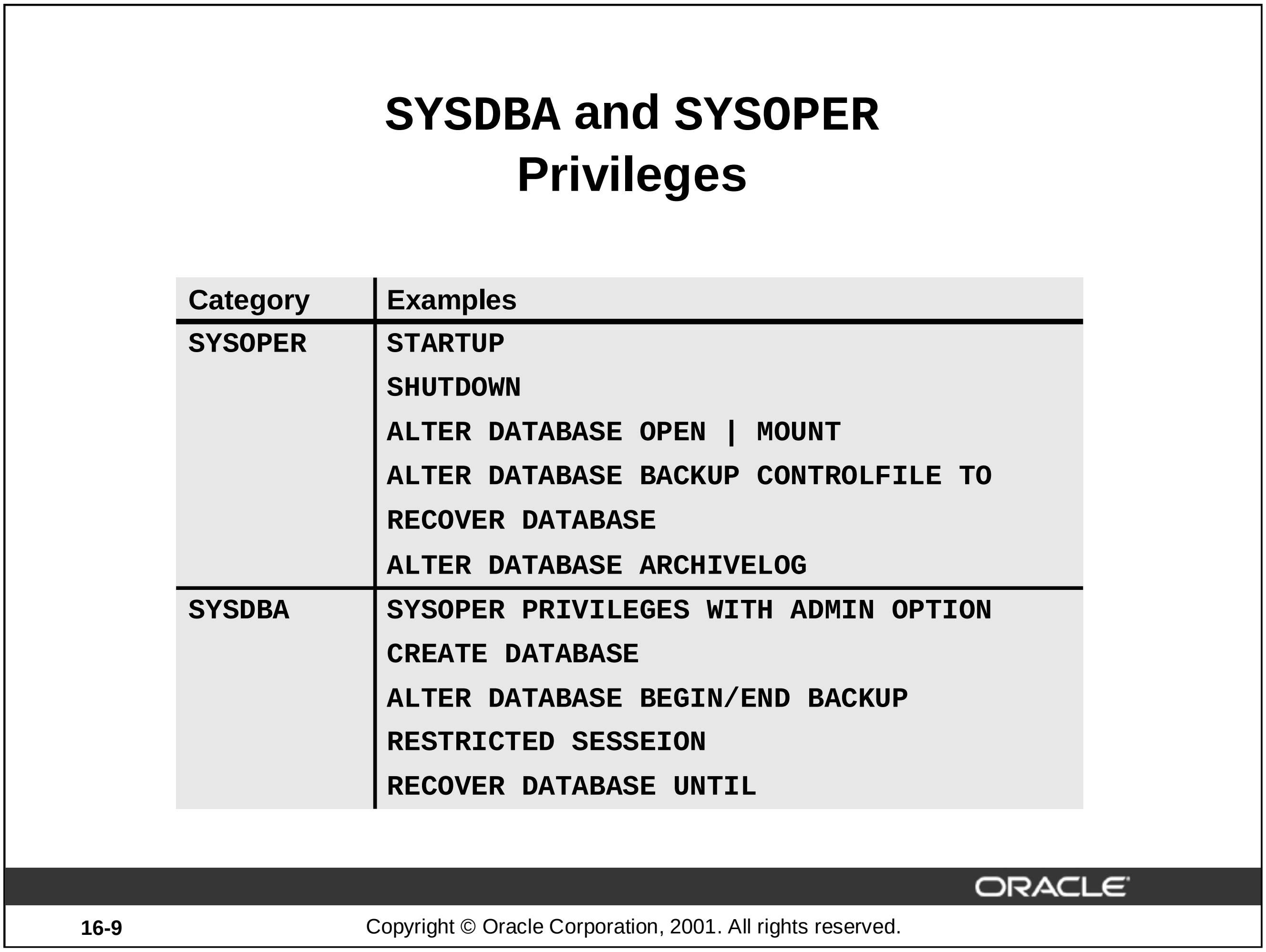
WITH ADMIN OPTION enables the grantee to further grant the privilege or role to other users or roles

Granting System Privileges (continued)

Using Enterprise Manager to grant System Privileges

1. Launch the Console:
   %oemapp console
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Right-click on the your working database and click Connect.
5. Supply the username, password and service name for your working database and click OK.
6. Expand the Security folder.
7. Expand the Users folder and select the user who needs the privilege.
8. Click the System Privileges tab on the detail side of the console.
9. Select the system privileges that you want to grant. Optionally, check the Admin Option box and click Apply.

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



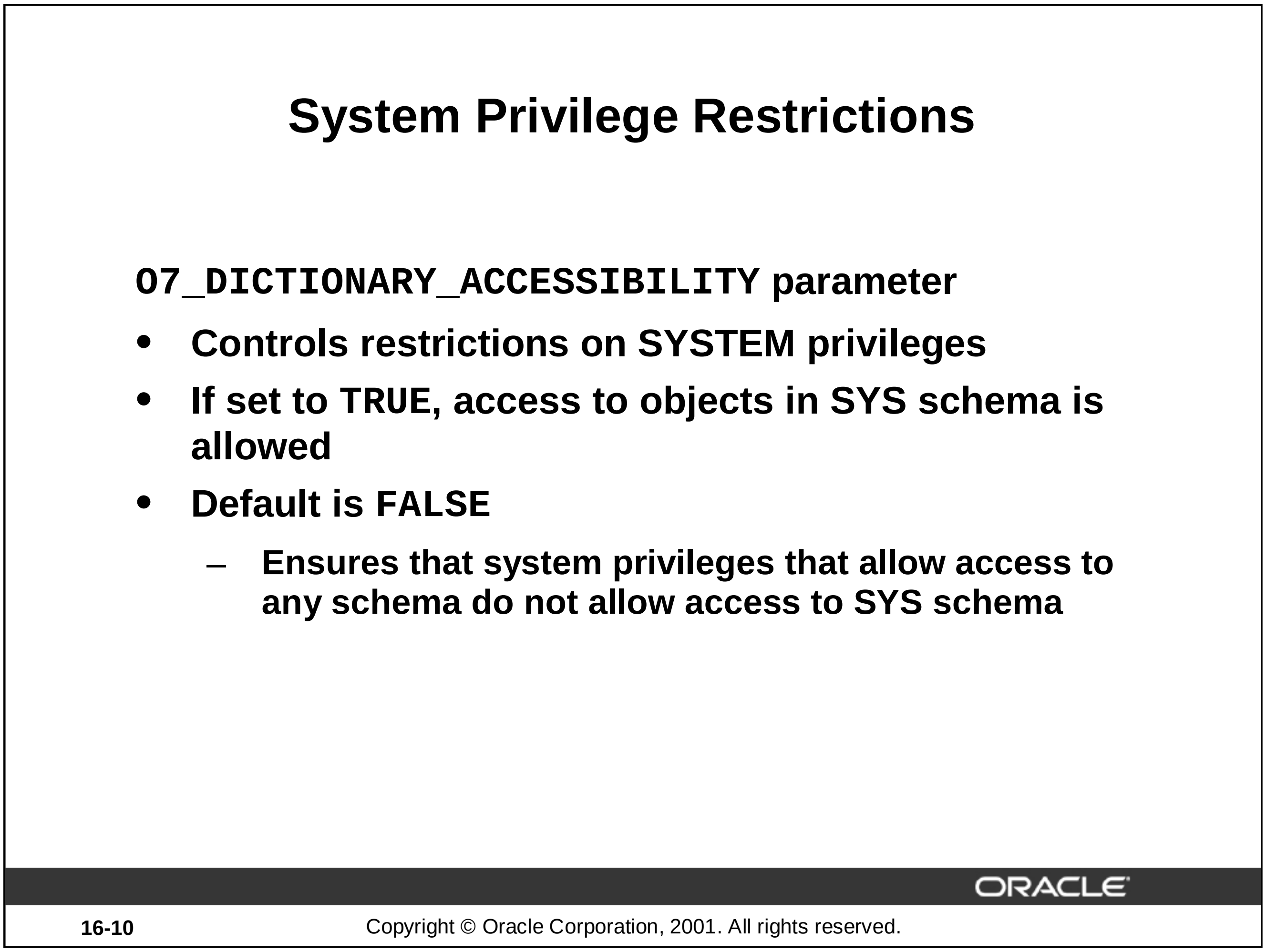
SYSDBA and SYSOPER Privileges

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

SYSDBA and SYSOPER Privileges

In the lesson Getting Started with Oracle, the system privileges SYSDBA and SYSOPER were introduced to specify the authentication by using a password file.

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



System Privilege Restrictions

`O7_DICTIONARY_ACCESSIBILITY` **parameter**

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

16-10

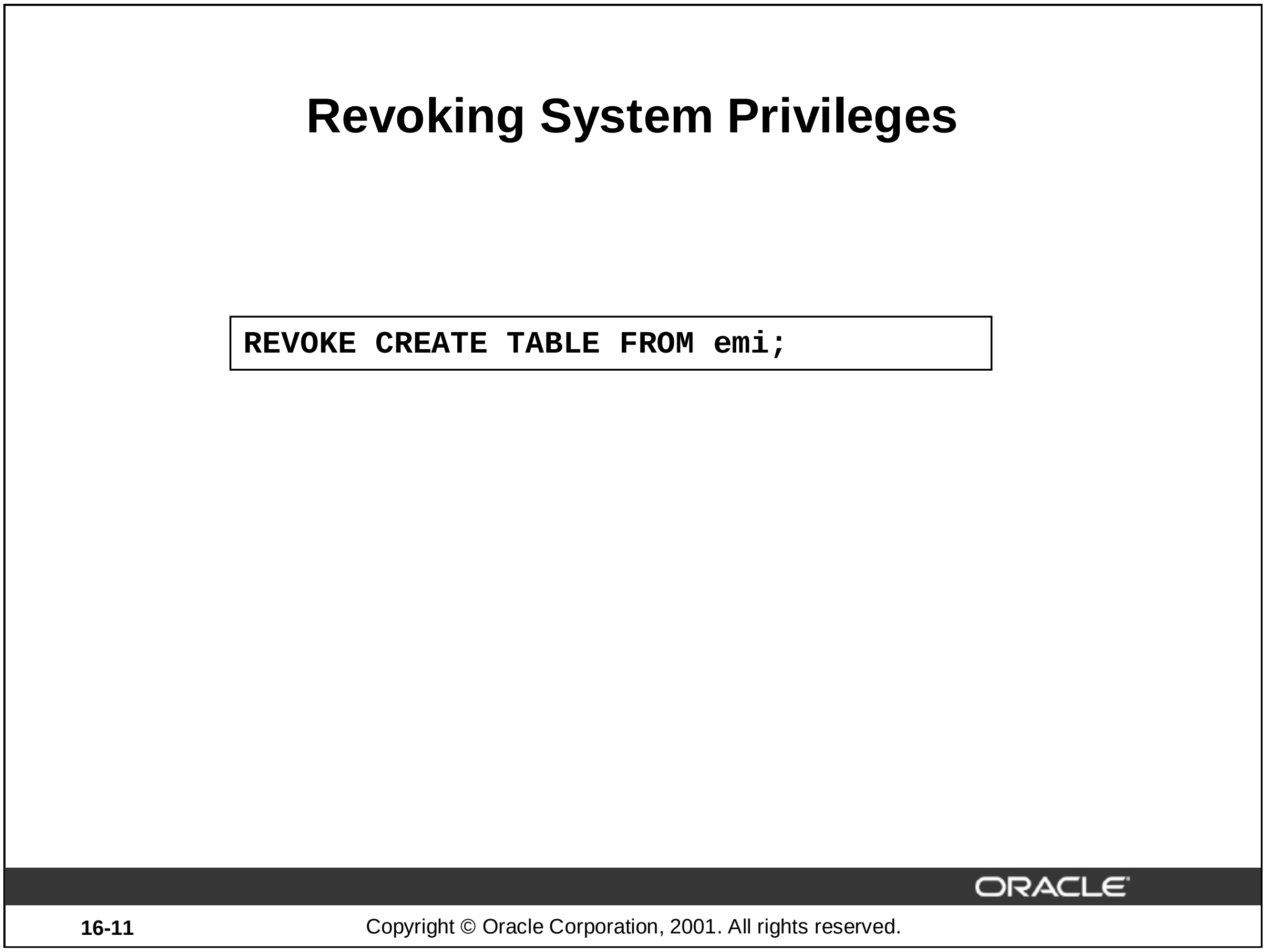
System Privilege Restrictions

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 providing 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 `SYS` schema (for example, dictionaries could not be accessed). The system privilege, `EXECUTE ANY PROCEDURE` will allow access on the procedures in any other schema except in `SYS` schema.



Revoking System Privileges

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

16-11

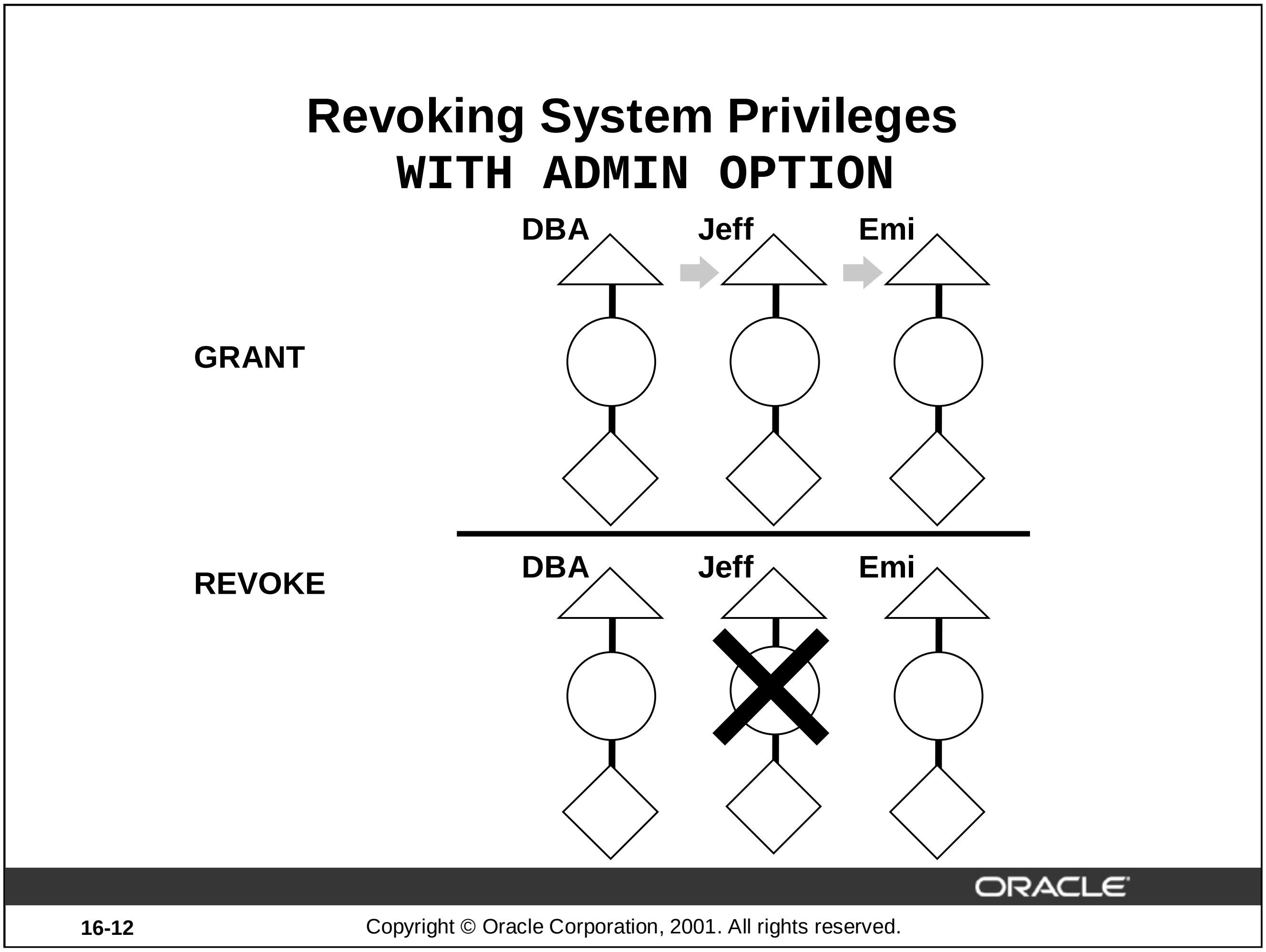
Revoking System Privileges

System privileges can be revoked using the SQL statement `REVOKE`. Any user 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 user that 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 any procedures or views that use a table in some other schema, revoking the privilege invalidates the procedures or views.



Revoking System Privileges WITH ADMIN OPTION

GRANT

REVOKE

Revoking System Privileges (continued)

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

To illustrate this, read through the following steps.

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.

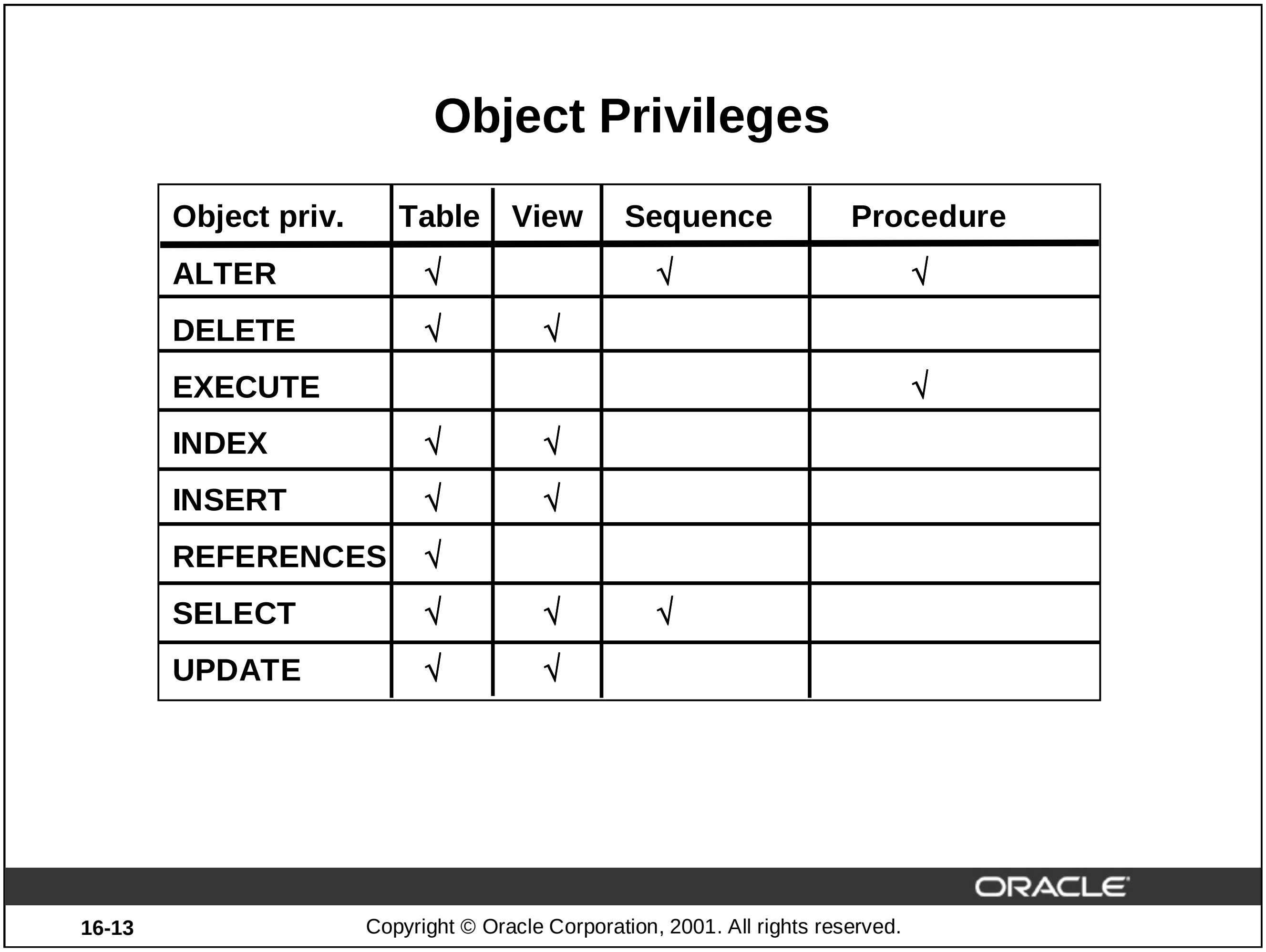
4. Emi creates a table.

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

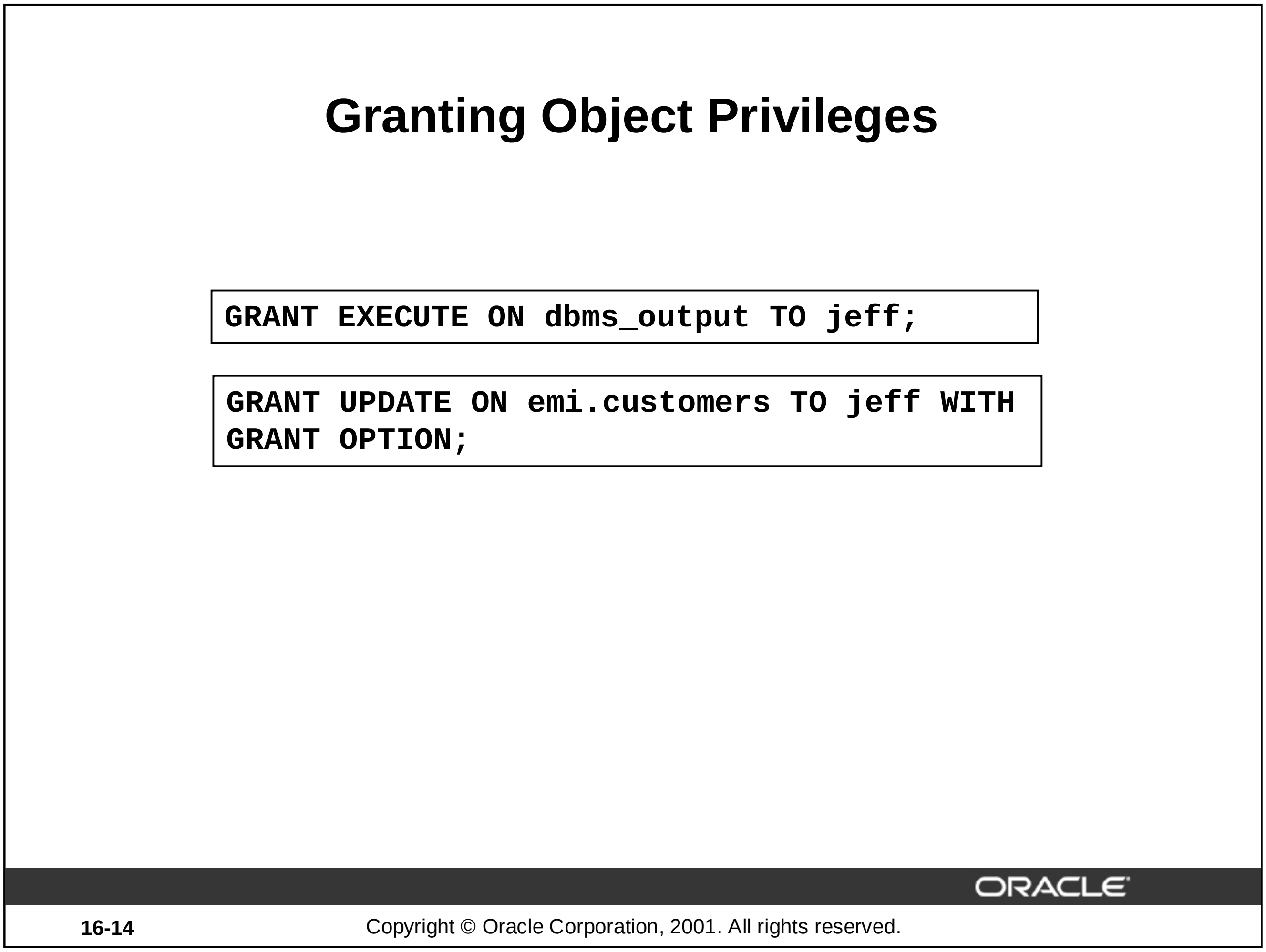
Object priv.	Table	View	Sequence	Procedure
ALTER	√		√	√
DELETE	√	√		
EXECUTE				√
INDEX	√	√		
INSERT	√	√		
REFERENCES	√			
SELECT	√	√	√	
UPDATE	√	√		

16-13

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 updateable columns. A `SELECT` can be restricted by creating a view with a subset of columns and granting `SELECT` privilege on the view. A grant on a synonym is converted to a grant upon the base table referenced by the synonym.

Note: This slide does not provide an exhaustive list of object privileges.



Granting Object Privileges

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

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

ORACLE

16-14 Copyright © Oracle Corporation, 2001. All rights reserved.

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 the object privileges to other users or roles

Granting Object Privileges (continued)

Use the GRANT statement to grant object privileges.

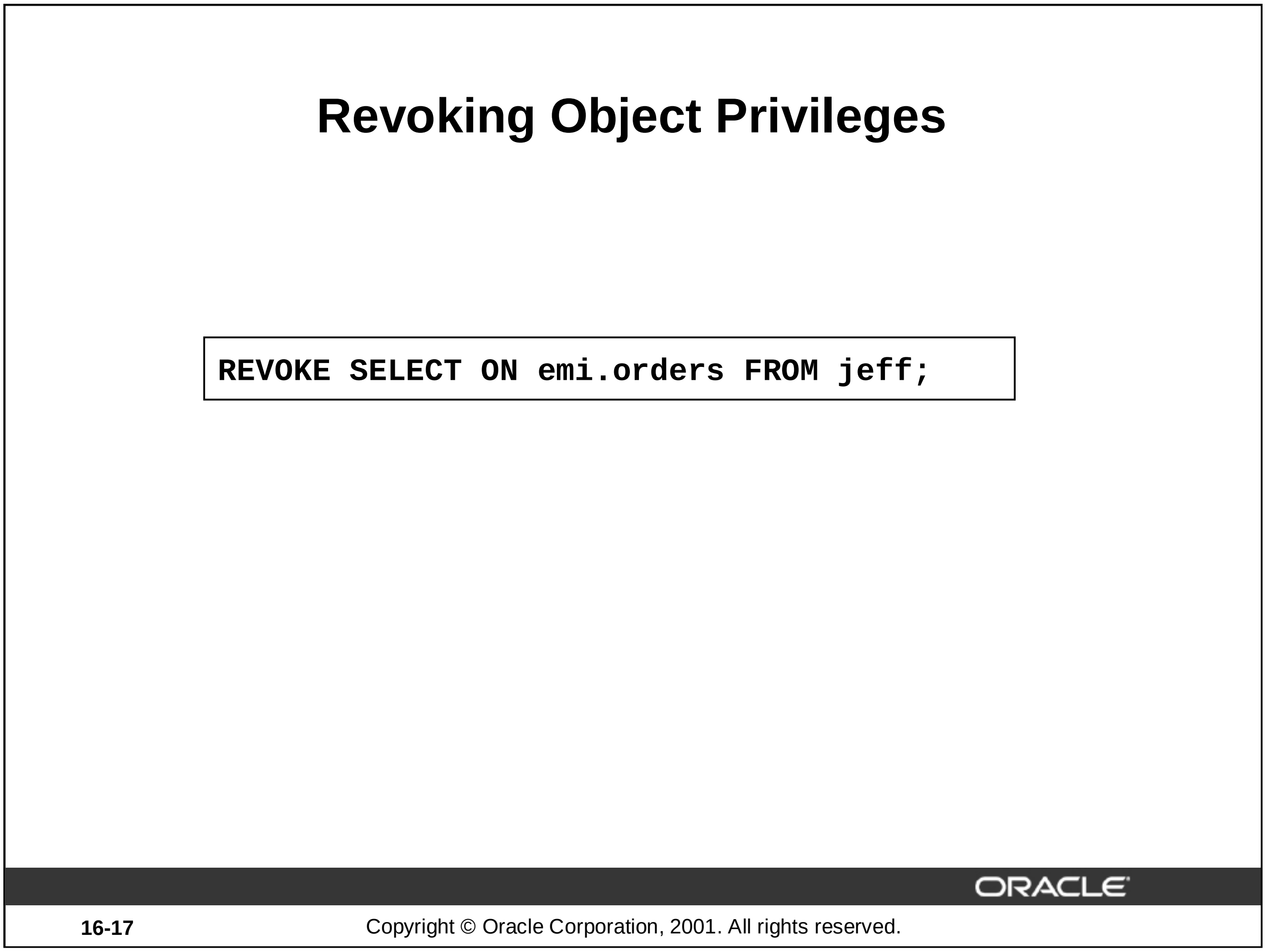
- To grant privileges, the object must be in your schema or you must have been given the privilege WITH GRANT OPTION.
- By default, if you own an object, all privileges on that object are automatically acquired.
- Use caution when granting privileges on your objects to other users when security is a concern.

Granting Object Privileges (continued)

Using Enterprise Manager to grant Object Privileges

In this example, user `HR` grants ‘`UPDATE`’ on `EMPLOYEES` table to user `Jeff`.

1. Launch the Console:
   `%oemapp console`
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Right-click on the your working database and click Connect.
5. Connect as user `HR` by supplying the username, password and service name and click OK.
6. Expand the Security folder.
7. Expand the Users folder and select the user who needs the privilege.
8. Click the Object Privileges tab on the detail side of the console.
9. Expand user `HR` and expand the Tables folder
10. Click the Table on which Object Privileges are to be granted, say `EMPLOYEES` table
11. Select `UPDATE` from the Available Privileges field and click the down arrow
12. Optionally, check the Grant Option box and click Apply.



Revoking Object Privileges

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

16-17

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 granted

`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

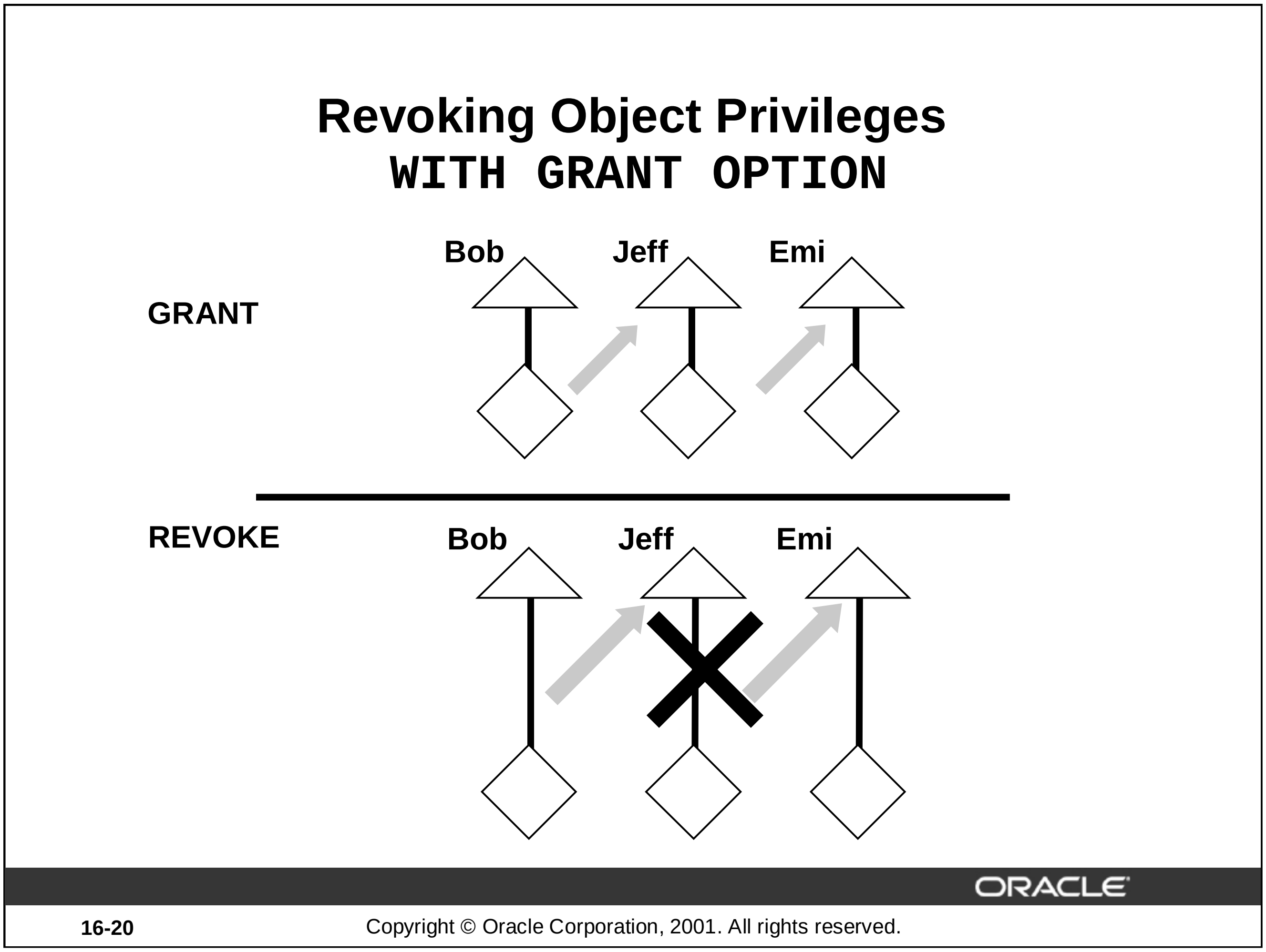
Grantors can revoke privileges from only those users to whom they have granted privileges.

Revoking Object Privileges (continued)

Using Enterprise Manager to Revoke Object Privileges

In this example, user `HR` is revoking the Object Privilege.

1. Launch the Console:
   `%oemapp console`
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Right-click on the your working database and click Connect.
5. Connect as user HR by supplying the username, password and service name and click OK.
6. Expand the Security folder.
7. Expand the Users folder and select the user from whom the privilege is to be revoked
8. Click the Object Privileges tab on the detail side of the console.
9. Select the object privilege that is to be revoked and click the Up arrow
10. Click Apply.



Revoking Object Privileges `WITH GRANT OPTION`

Revoking Object Privileges (continued)

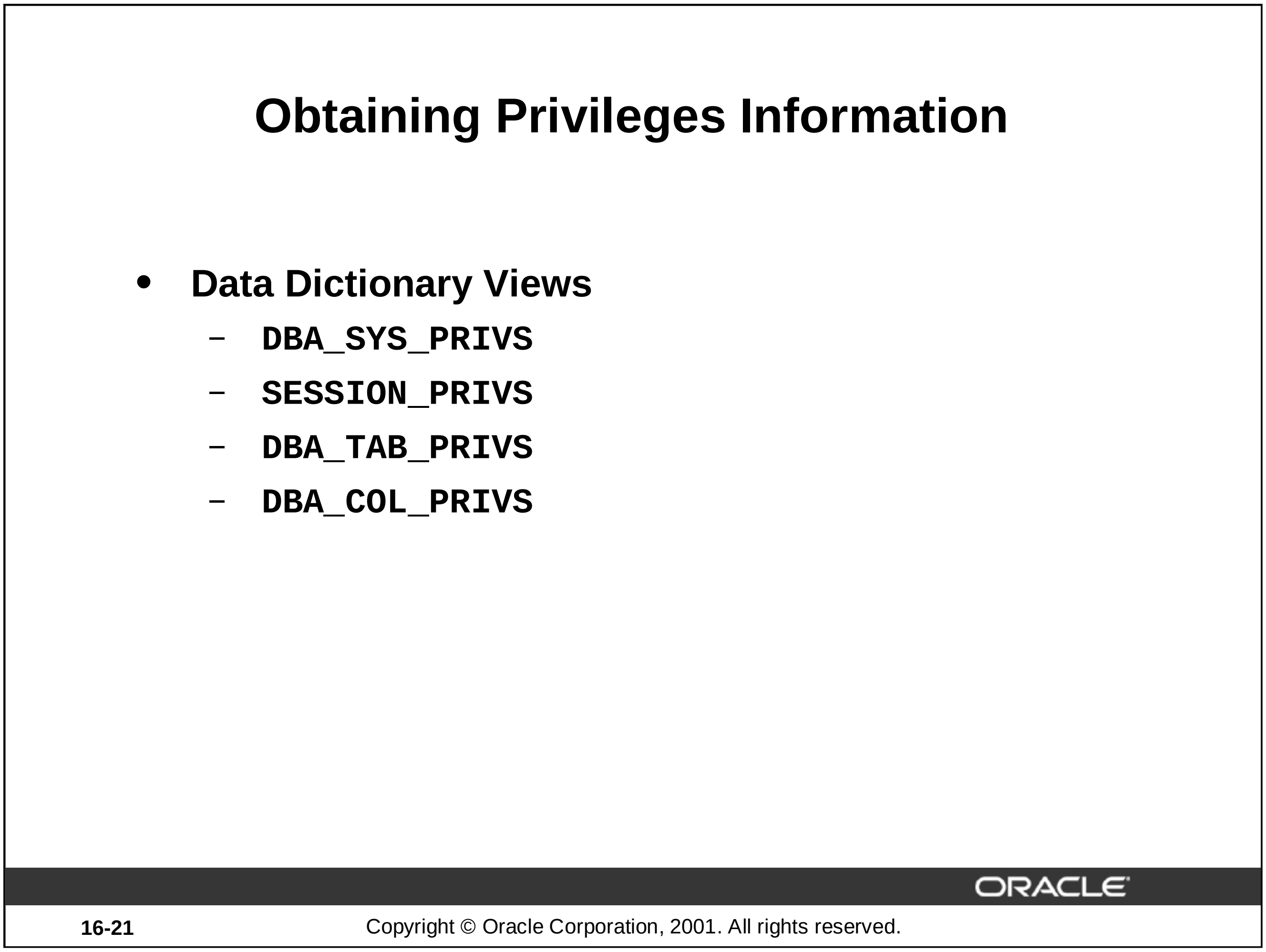
Cascading effects can be observed when revoking a system privilege related to a DML operation. For example, if `SELECT ANY TABLE` is granted to a user, and that user has created any procedures that use the table, all procedures contained in the user's schema must be recompiled before they can be used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

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

To illustrate this, read through the following steps.

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.



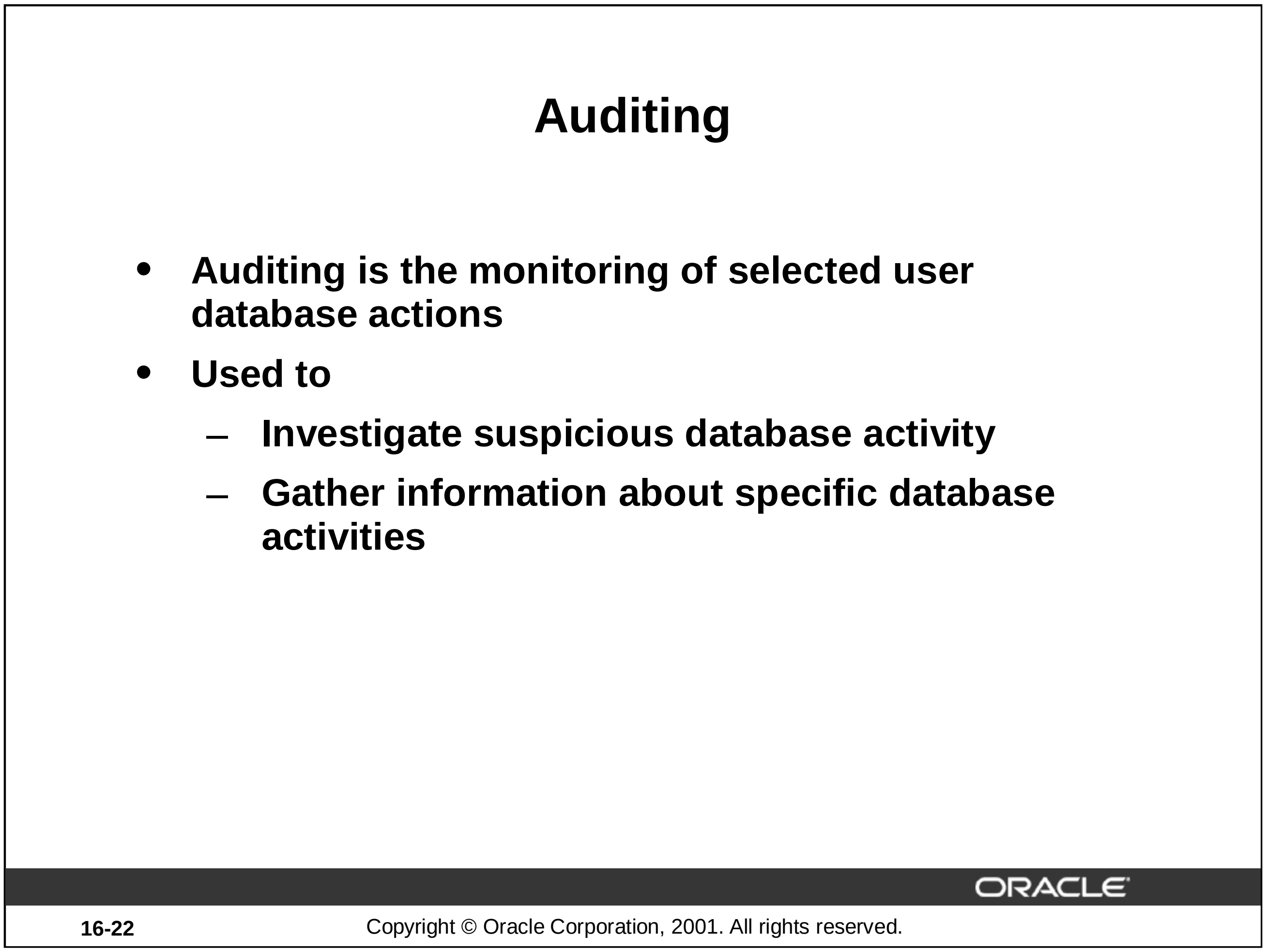
Obtaining Privileges Information

- Data Dictionary 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.



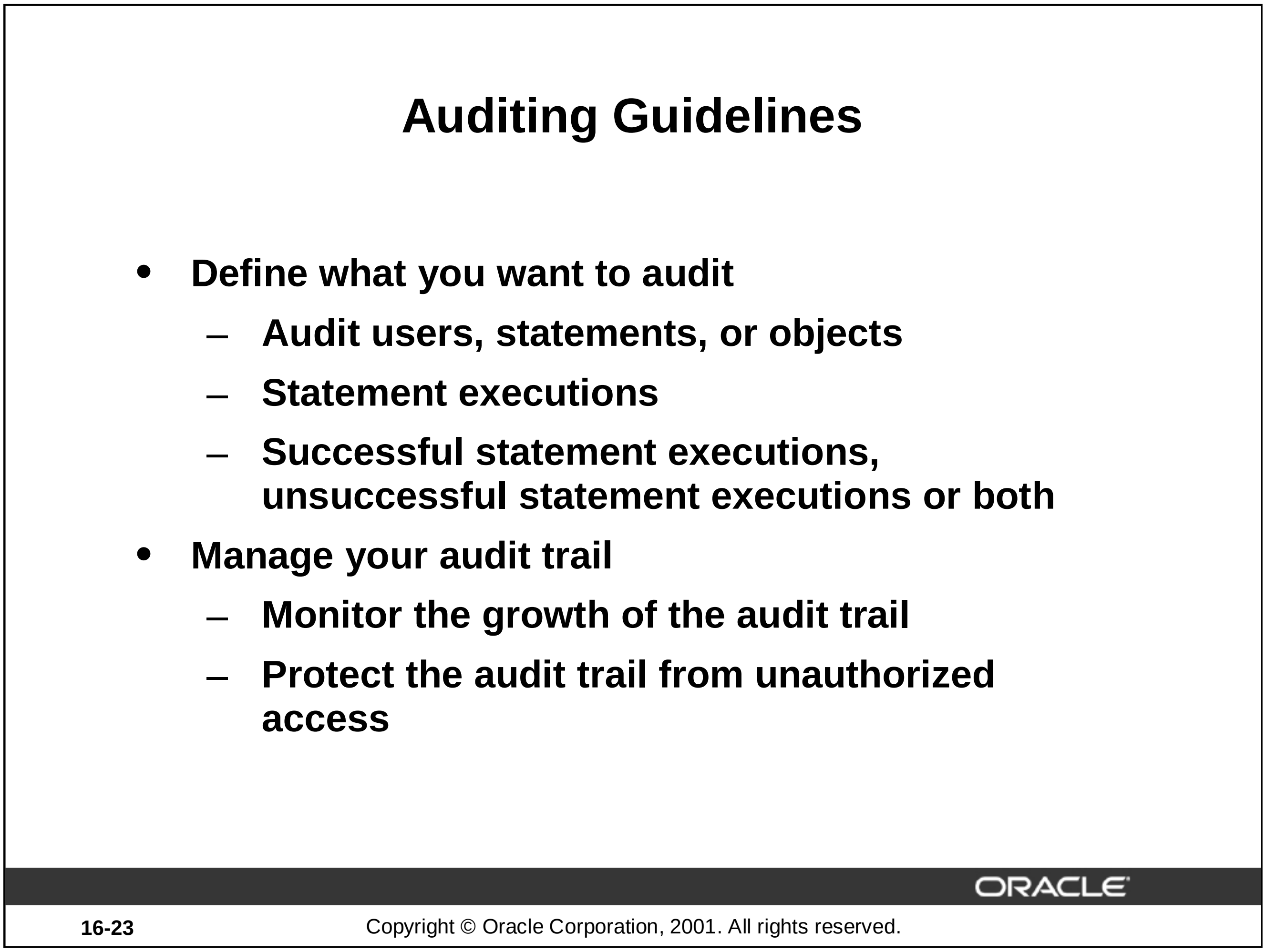
Auditing

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

16-22

Auditing

If an unauthorized user is deleting data, 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 I/Os are performed, and how many concurrent users connect at peak times.



Auditing Guidelines

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

Auditing 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 needs to be used, the following settings can minimize audit generation:

- Specifying users to audit
- Auditing by session, and not by access
- Auditing either successes or failures, but not both
- Audit records may 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 that issue an audited statement. You must free some space in the audit trail before these statements can be executed.

Monitoring the Growth of the Audit Trail (continued)

To ensure the audit trail does not grow too rapidly:

- 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.
- The AUDIT ANY privilege also enables a user to turn on auditing, so grant it sparingly.

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:

```
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 get inserted into the database audit trail, the AUD$ table can grow without bound. 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:

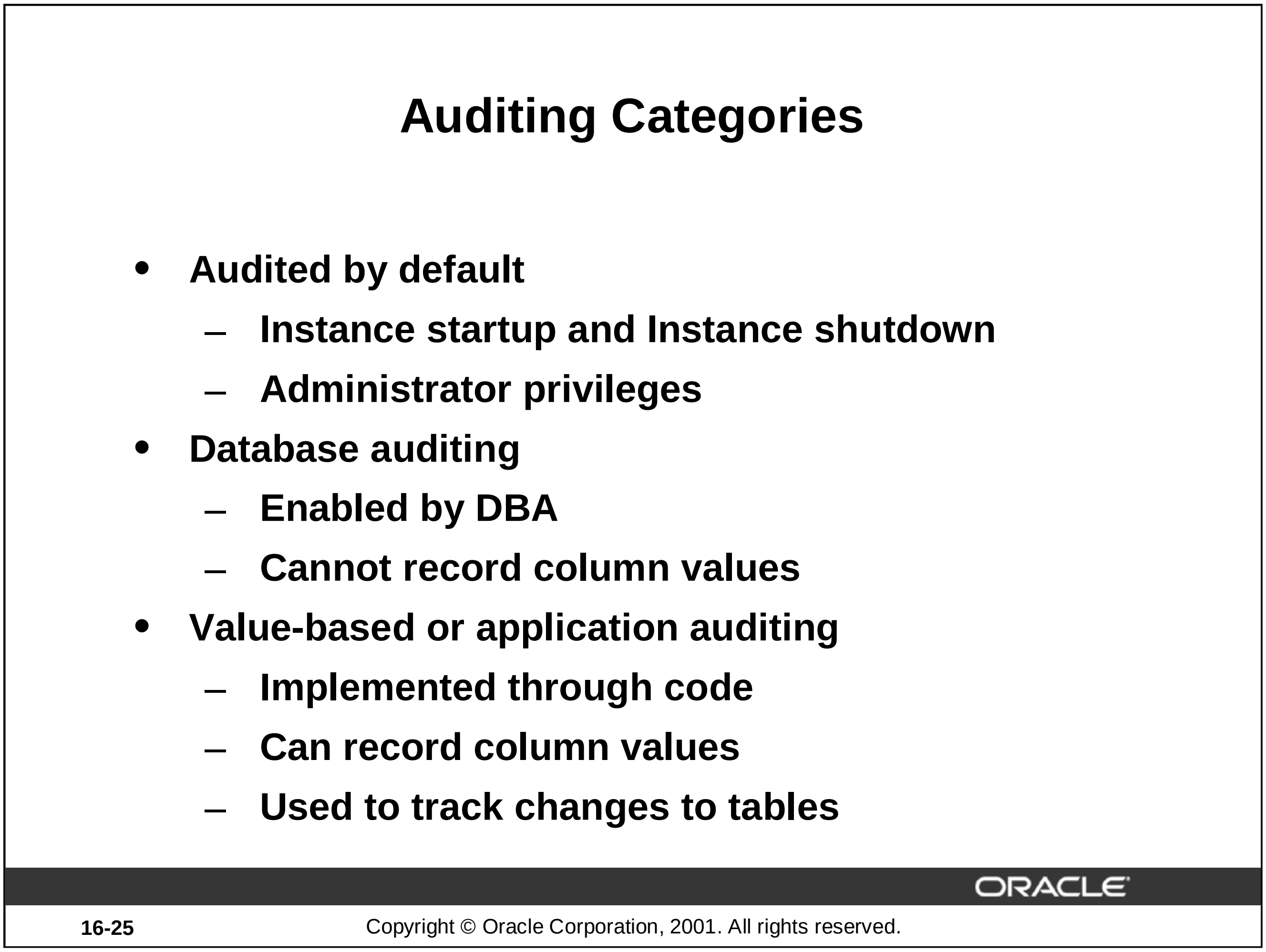
- Ensure that auditing is currently disabled.
- Enter the following command:

```
ALTER TABLE aud$ MOVE TABLESPACE AUDIT_TAB;
```

- Enter the following command:

```
CREATE INDEX i_aud1 ON aud$(sessionid, ses$tid)
TABLESPACE AUDIT_IDX;
```

- Enable auditing for the instance.



Auditing Categories

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

16-25

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

Database Auditing

Database auditing is the monitoring and recording of selected user database actions. Information about the event is stored in the audit trail.

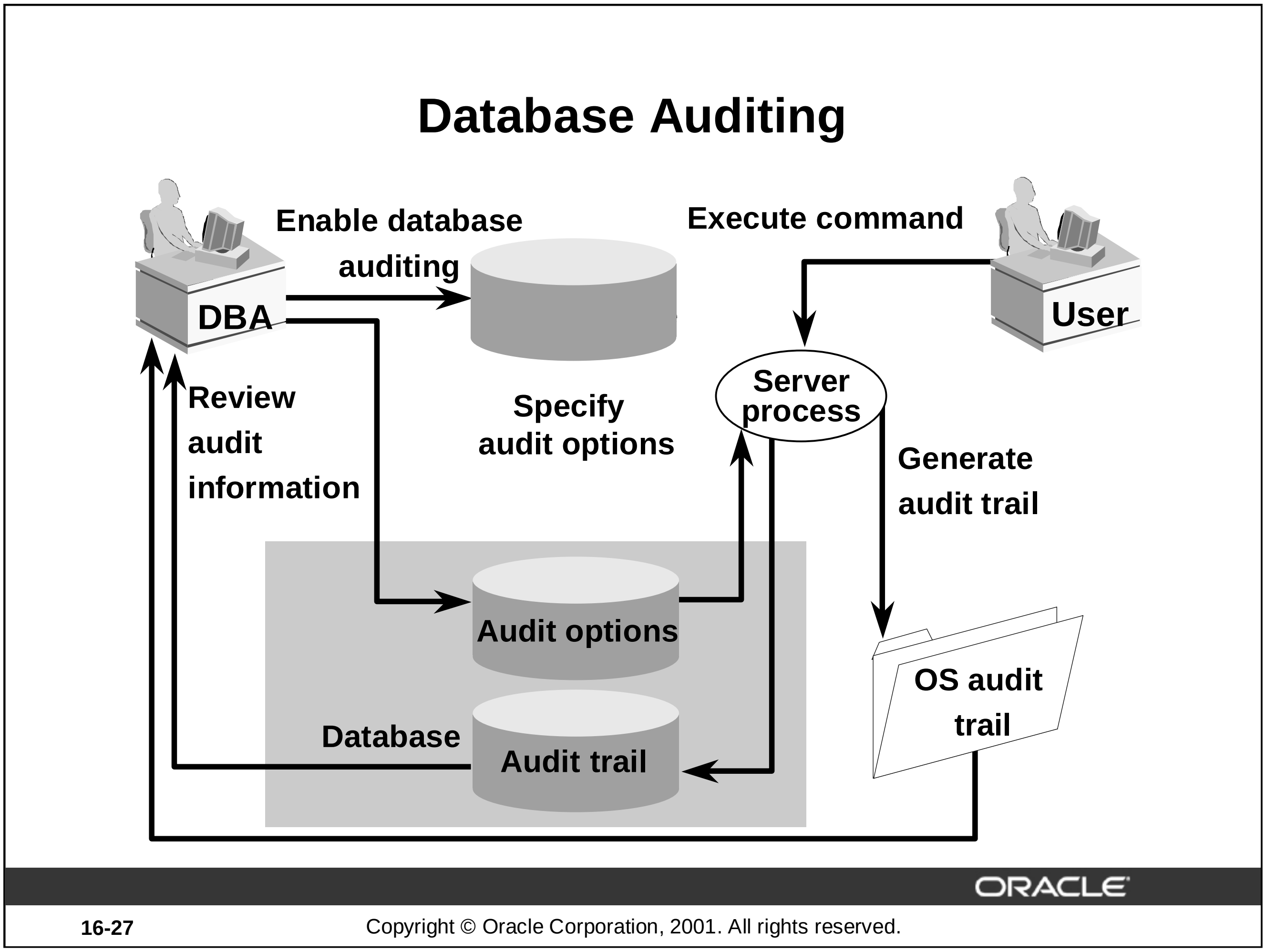
The audit trail can be used to investigate suspicious activity. For example, if an unauthorized user is deleting data from tables, the database administrator may decide to audit all connections to the database in conjunction with successful and unsuccessful deletions of rows from tables in the database.

Database Auditing (continued)

Auditing might also be used to monitor and gather data about specific database activities. For example, the database administrator 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 need to be tracked and column values need to be stored for each change, use application auditing. Application auditing can be done either through client code, stored procedures, or database triggers.



Database Auditing

16-27

Enabling and Disabling Database Auditing

Once 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:

`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)
`NONE`	disables auditing (this is the default value)

Enabling 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

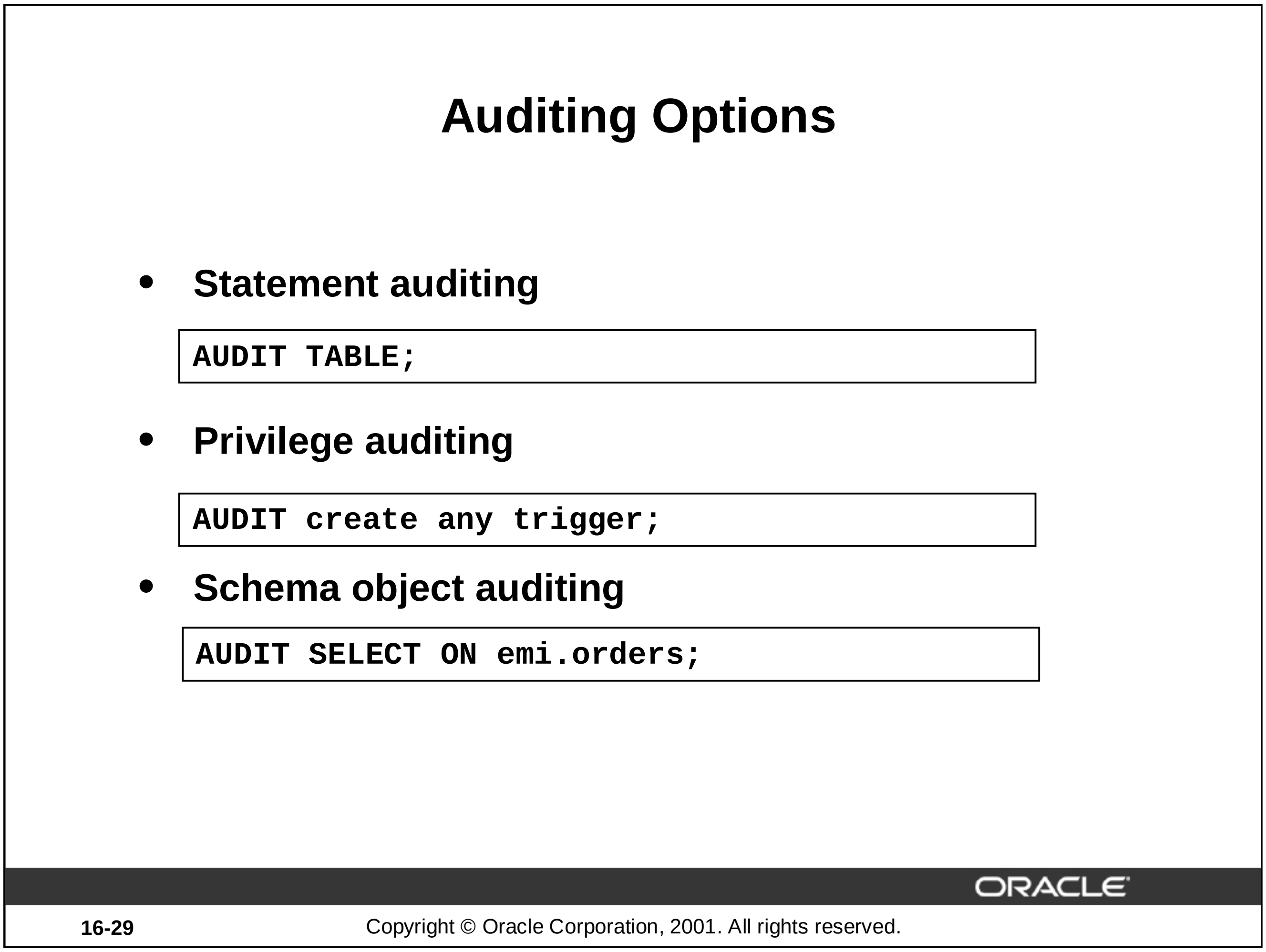
When users execute PL/SQL and SQL statements, the server process examines the auditing options to determine if the statement being executed should generate an audit record. SQL statements inside PL/SQL program units are individually audited, as necessary, when the program unit is executed. Because views 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. Since the audit record is generated during the execute phase, a syntax error, which occurs during the parse phase, will not cause an audit trail record to be generated.

Reviewing Audit Information

Examine the information 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;
```

16-29

Audit 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.

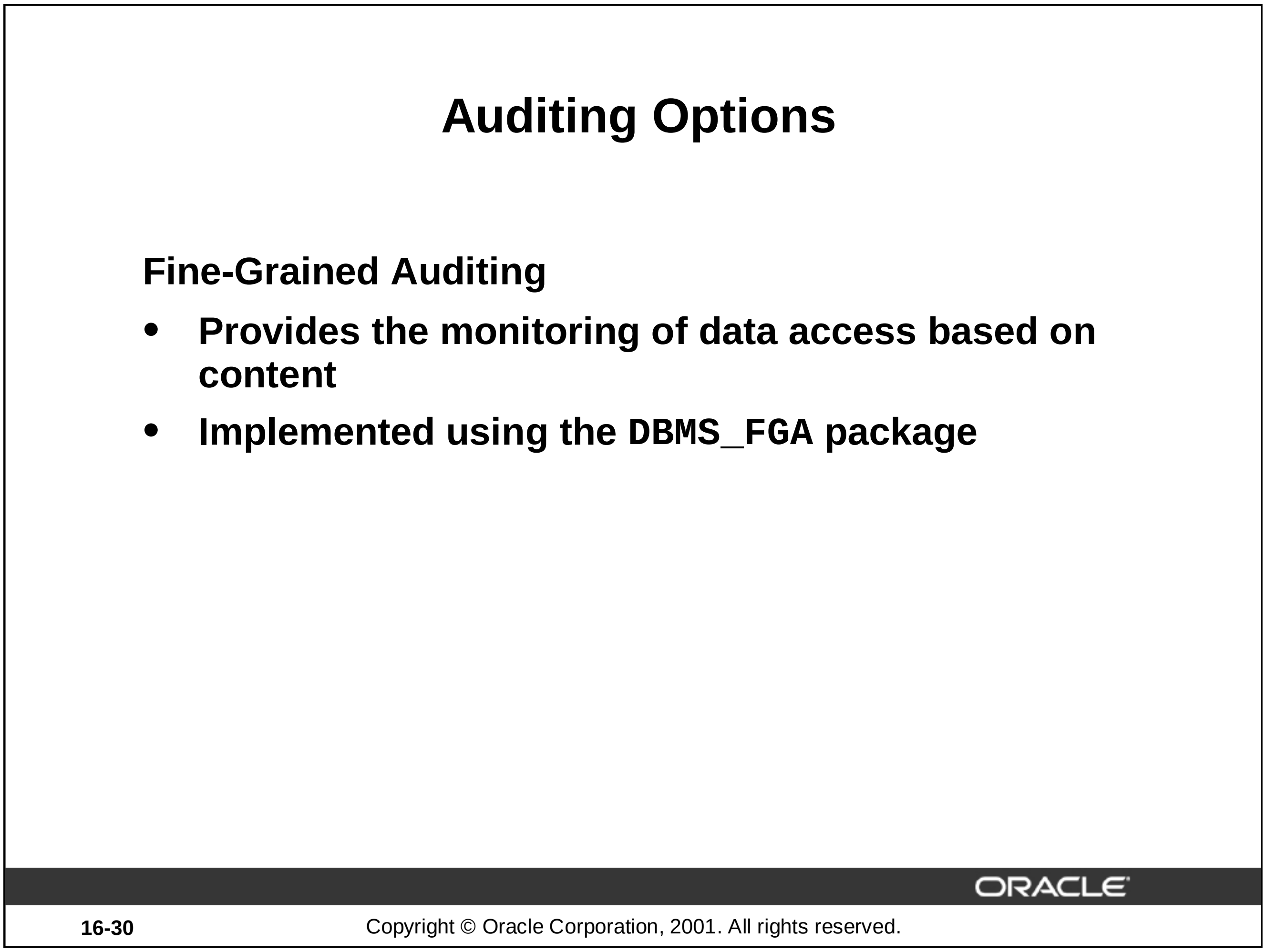
Privilege auditing: This is the selective auditing of system privileges to perform corresponding actions, such as `AUDIT CREATE ANY TRIGGER`. You can set privilege auditing to audit a selected user or every user in the database.

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`

For specific users or for all users in the database (statement and privilege auditing only).



Auditing Options

Fine-Grained Auditing

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

16-30

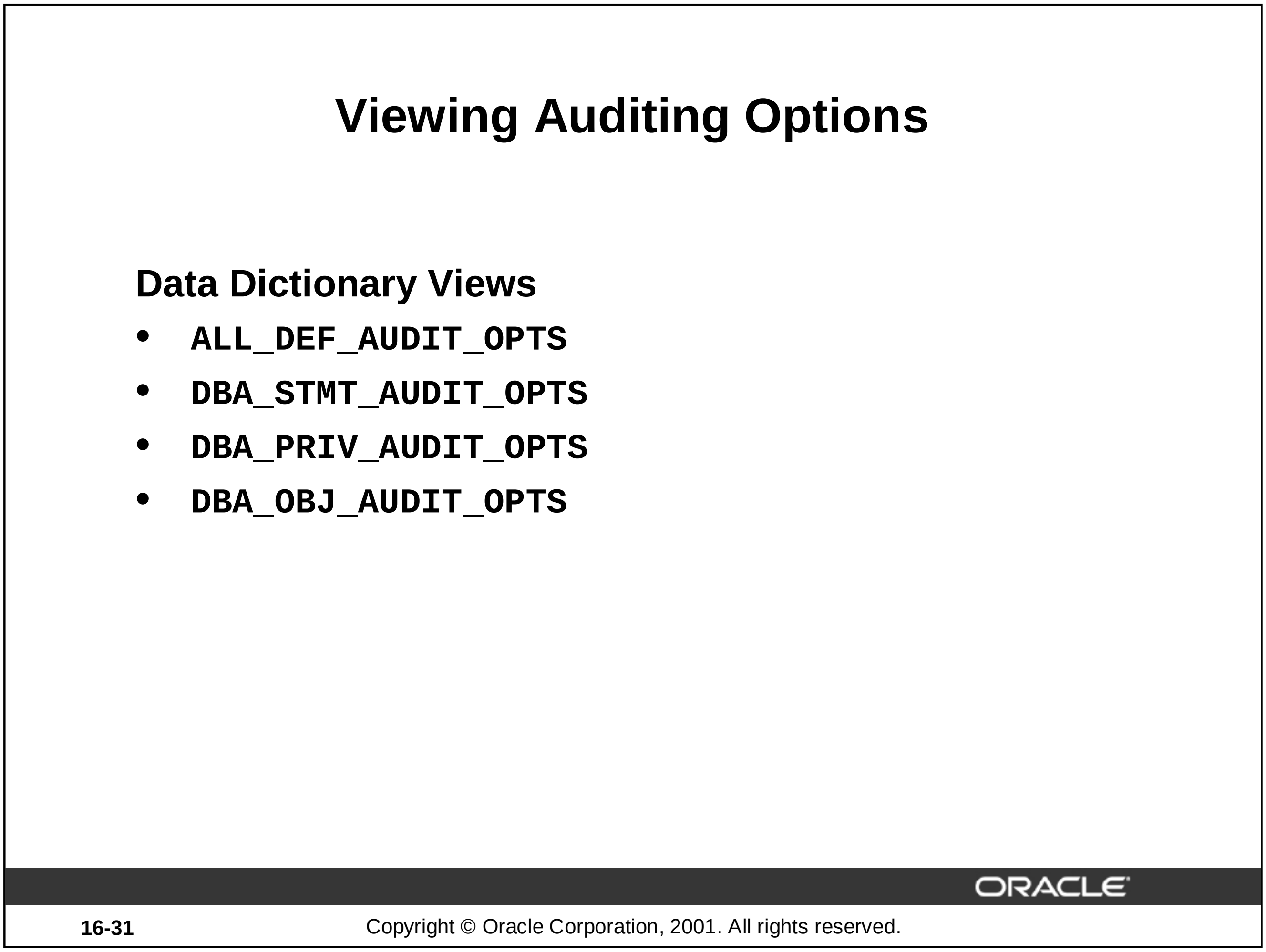
Audit Options

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, an audit event entry, including username, SQL text, bind variable, policy name, session id, timestamp, and other attributes are 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. Note that the NOAUDIT statement must have the same syntax as the previous `AUDIT` statement and that 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 specified.



Viewing Auditing Options

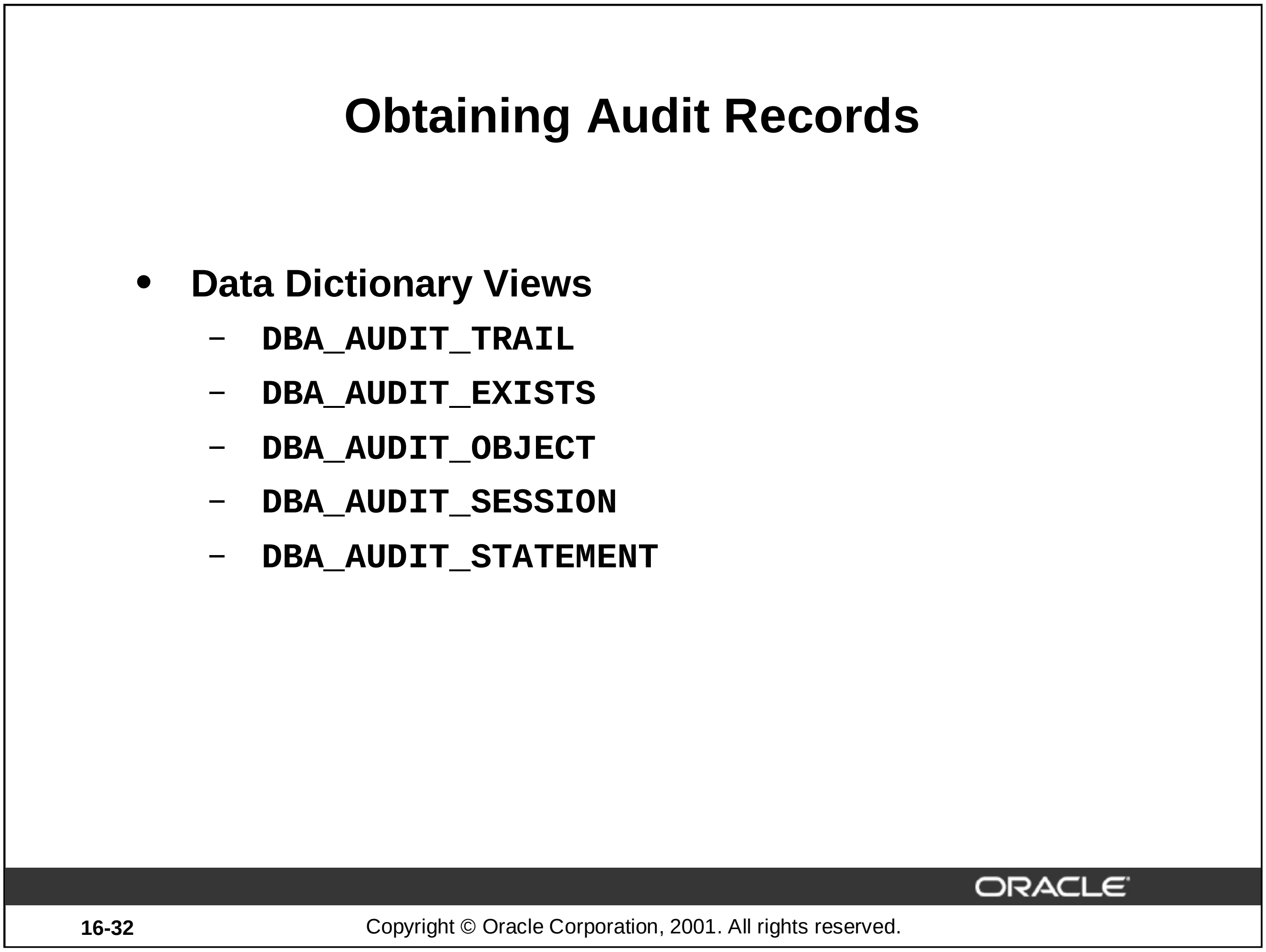
Data Dictionary Views

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

16-31

Viewing Auditing Options

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

- Data Dictionary Views
  - **`DBA_AUDIT_TRAIL`**
  - **`DBA_AUDIT_EXISTS`**
  - **`DBA_AUDIT_OBJECT`**
  - **`DBA_AUDIT_SESSION`**
  - **`DBA_AUDIT_STATEMENT`**

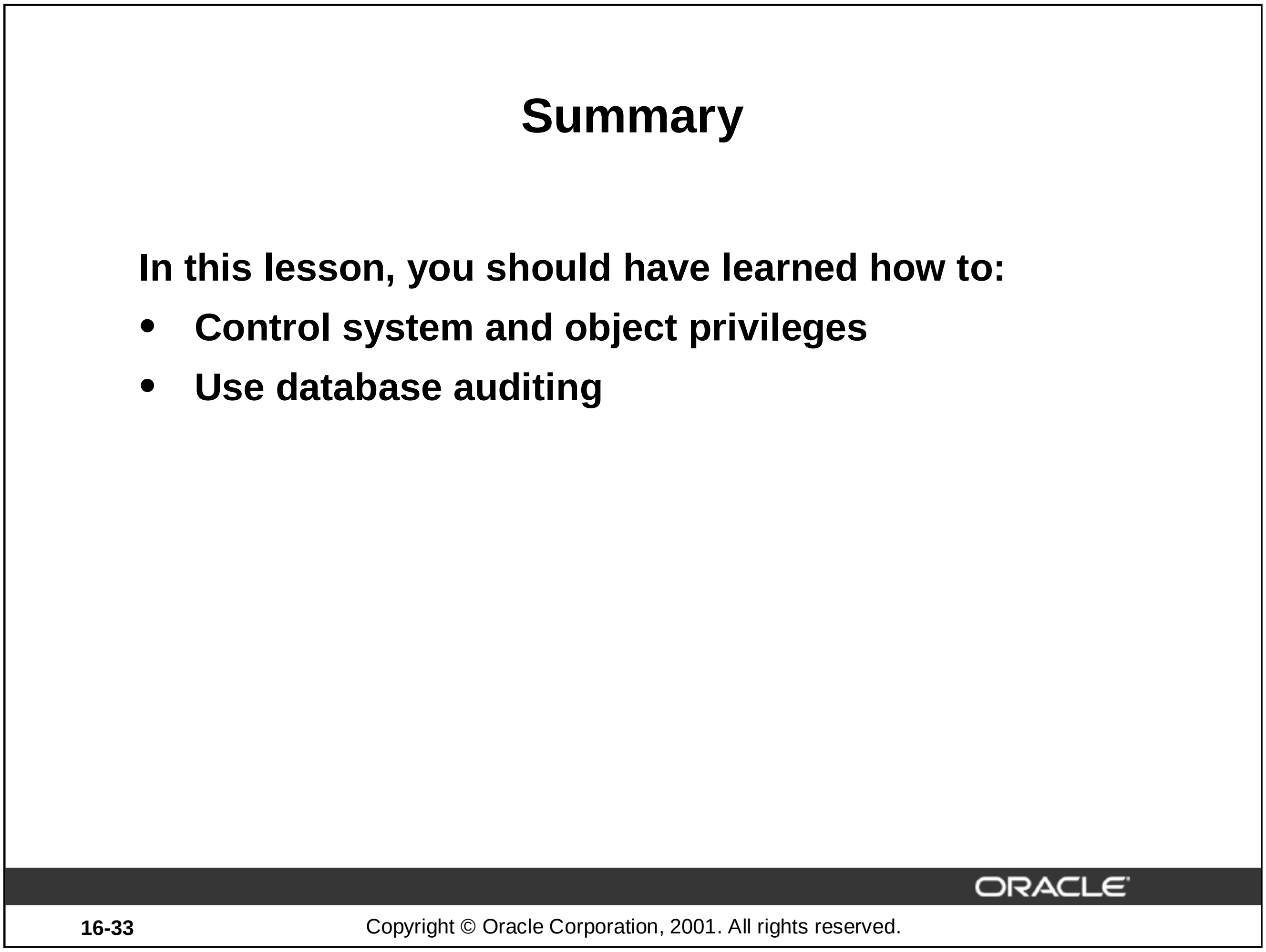
ORACLE

16-32

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

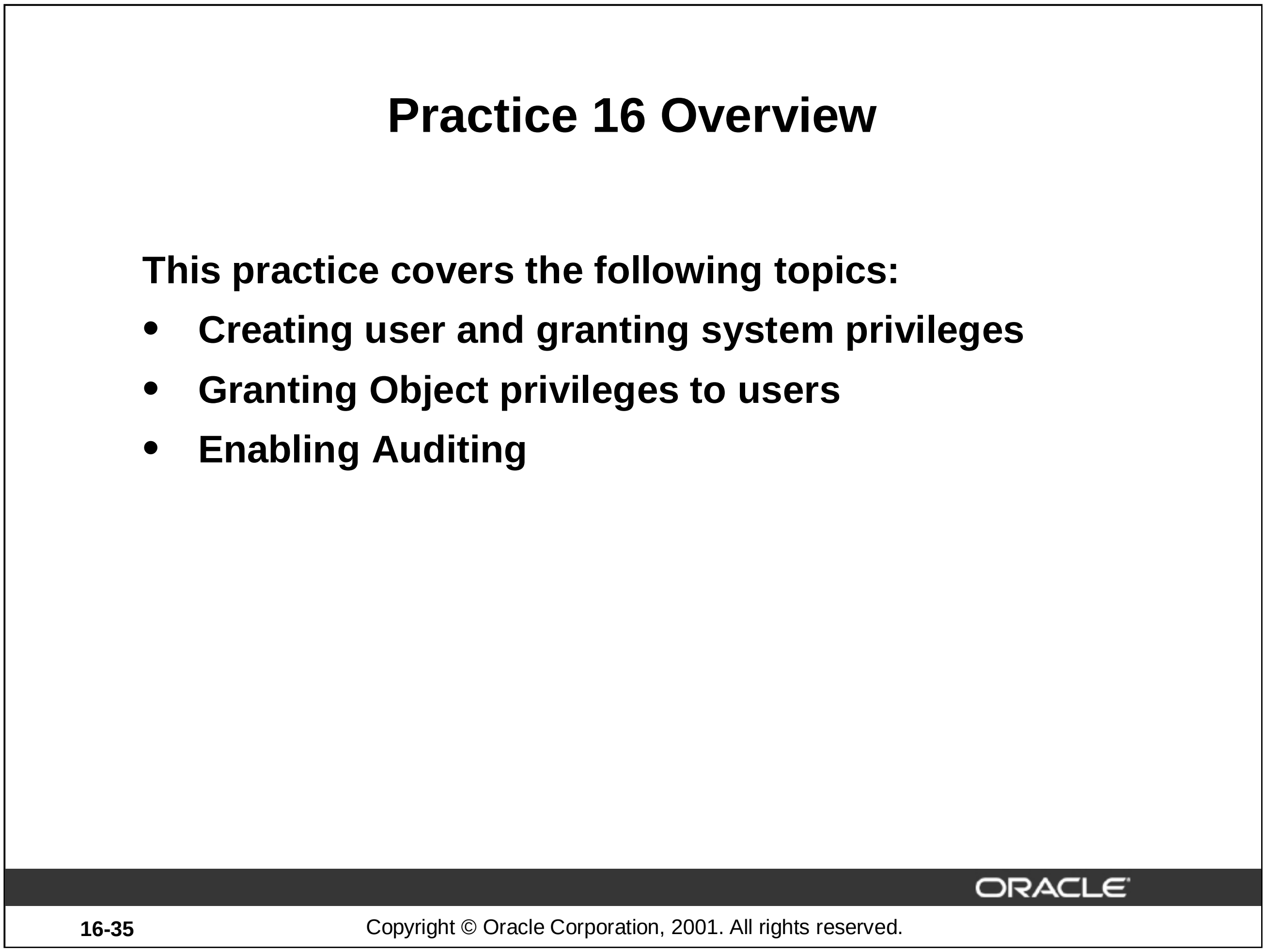
In this lesson, you should have learned how to:

- Control system and object privileges
- Use database auditing

16-33

Quick Reference

Context	Reference
Initialization parameters	O7_DICTIONARY_ACCESSIBILITY AUDIT_TRAIL
Data dictionary views	DBA_SYS_PRIVS SESSION_PRIVS DBA_TAB_PRIVS DBA_COL_PRIVS ALL_DEF_AUDIT_OPTS AUDIT_ACTIONS DBA_AUDIT_OBJECT DBA_AUDIT_SESSION DBA_AUDIT_STATEMENT DBA_AUDIT_TRAIL DBA_OBJ_AUDIT_OPTS DBA_PRIV_AUDIT_OPTS DBA_STMT_AUDIT_OPTS
Commands	GRANT REVOKE AUDIT NOAUDIT



Practice 16 Overview

This practice covers the following topics:

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

Practice 16: Managing Privileges

1 As `SYSTEM`, create user `Emi` and give her the capability to log on to the database and create objects in her schema.

2 a Connect as `Emi`, and create tables using the script `lab16_02a.sql` to create the tables `CUSTOMERS` and `ORDERS`.

b Connect as `SYSTEM` and copy the data from `SYSTEM.CUSTOMERS` to `Emi`’s `CUSOMTERS` table. Verify that records have been inserted.

c As `SYSTEM` give `Bob` the ability to select from `Emi`'s `CUSTOMERS` table. What happens and why?

3 Reconnect as `Emi` and give `Bob` the ability to select from `Emi`'s `CUSTOMERS` table. Also, enable `Bob` to give the select capability to other users. Examine the data dictionary views that record these actions.

4 Create user `Trevor` with the capability to log on to the database

5 a As `Bob`, enable `Trevor` to access `Emi`’s `CUSTOMERS` table. Give `Bob` the new password `sam`.

b As `Emi`, remove `Bob`’s privilege to read `Emi`’s `CUSTOMERS` table.

c As `Trevor`, query `Emi`’s `CUSTOEMRS` table. What happens and why?

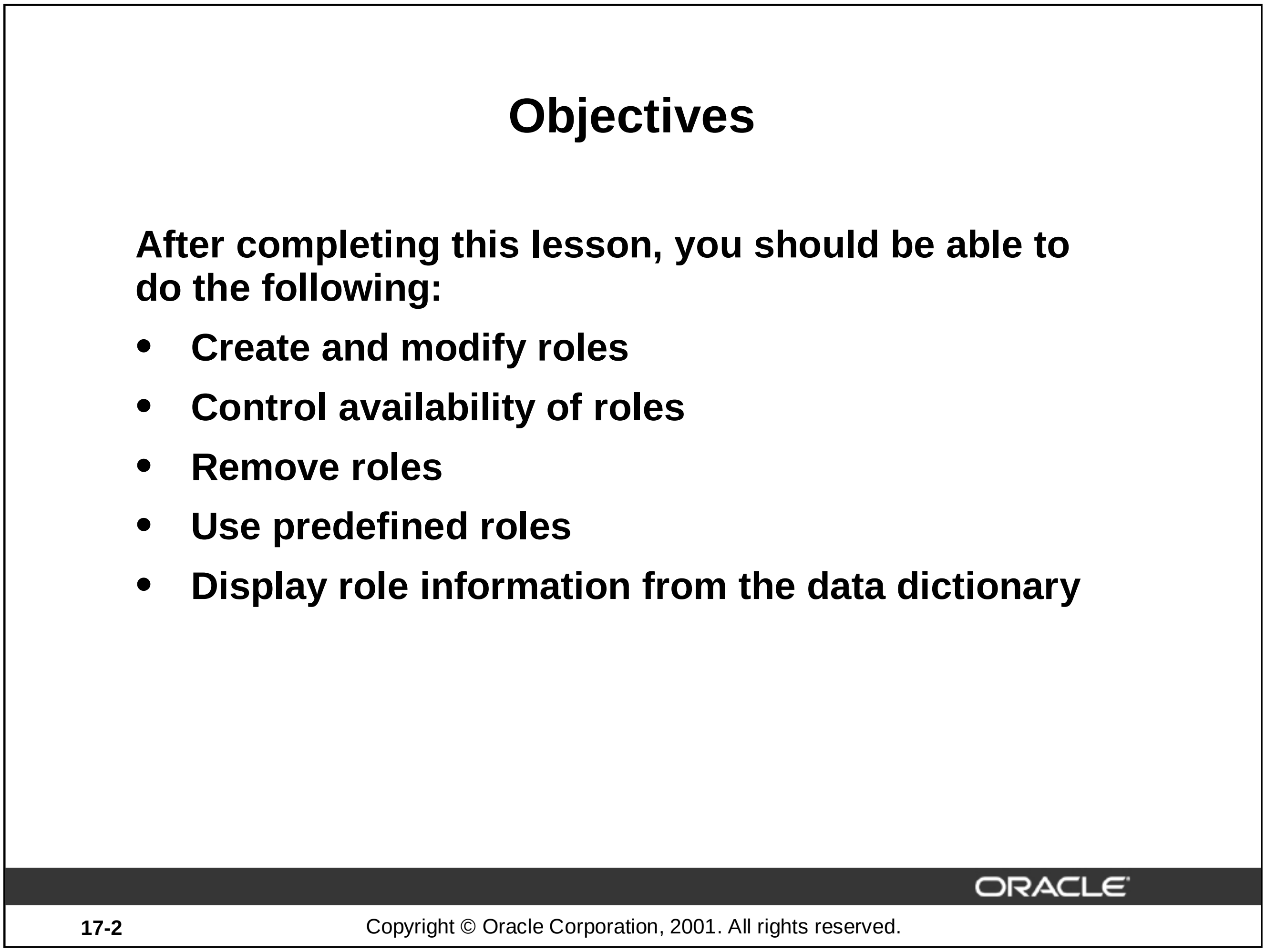
6 a Enable `Emi` to create tables in any schema. As `Emi`, create the table `ORDERS` in `Bob`’s schema as a copy of `EMI.ORDERS`. What happened and why?

b As `SYSTEM`, examine the data dictionary view `DBA_TABLES` to check the result.

7 Enable `Emi` to start up and shut down the database without the ability to create a new database.

17

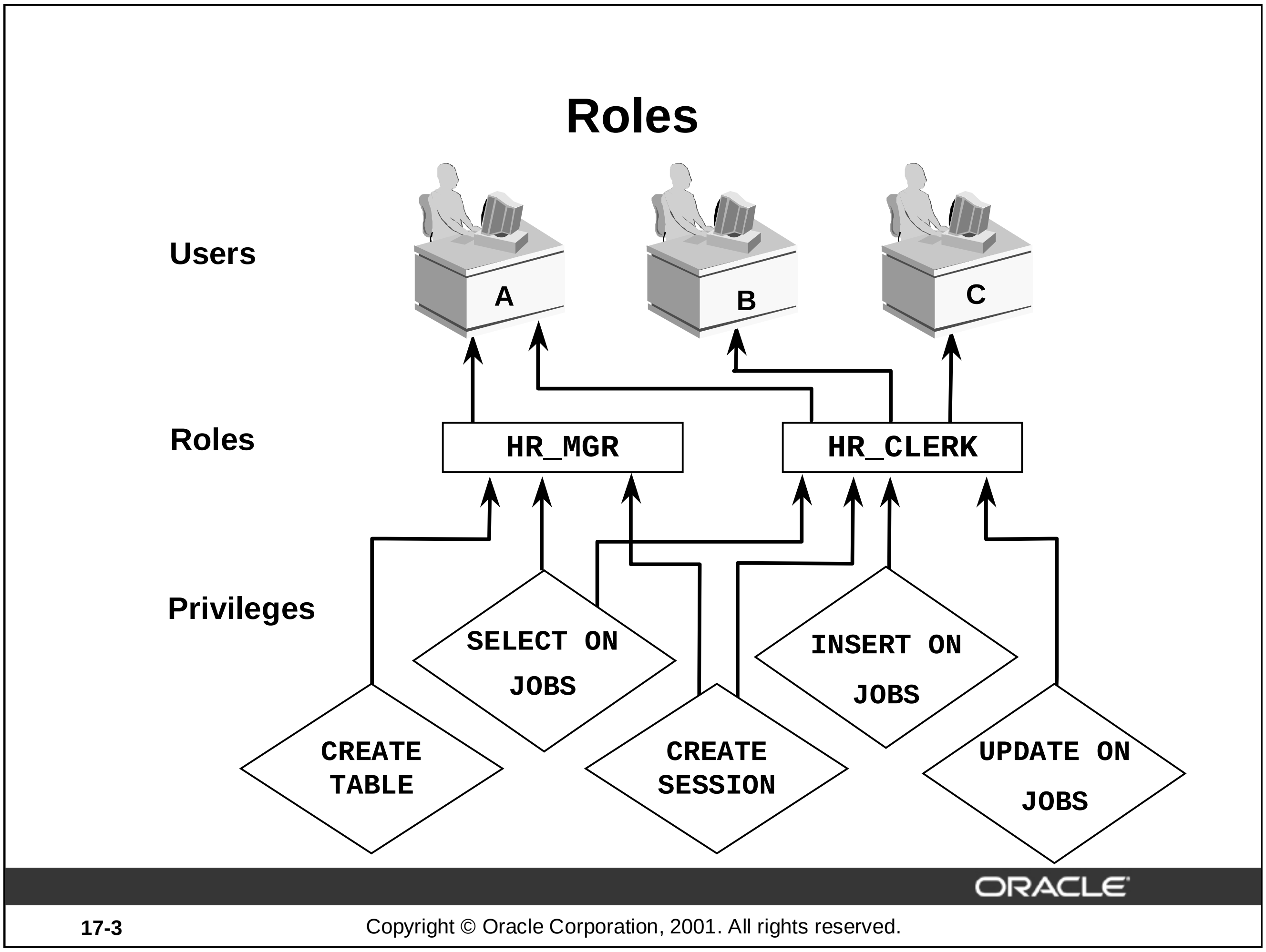
Managing Roles



Objectives

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

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



Roles

Users

A

B

C

Roles

HR_MGR

HR_CLERK

Privileges

SELECT ON JOBS

INSERT ON JOBS

CREATE TABLE

CREATE SESSION

UPDATE ON JOBS

ORACLE

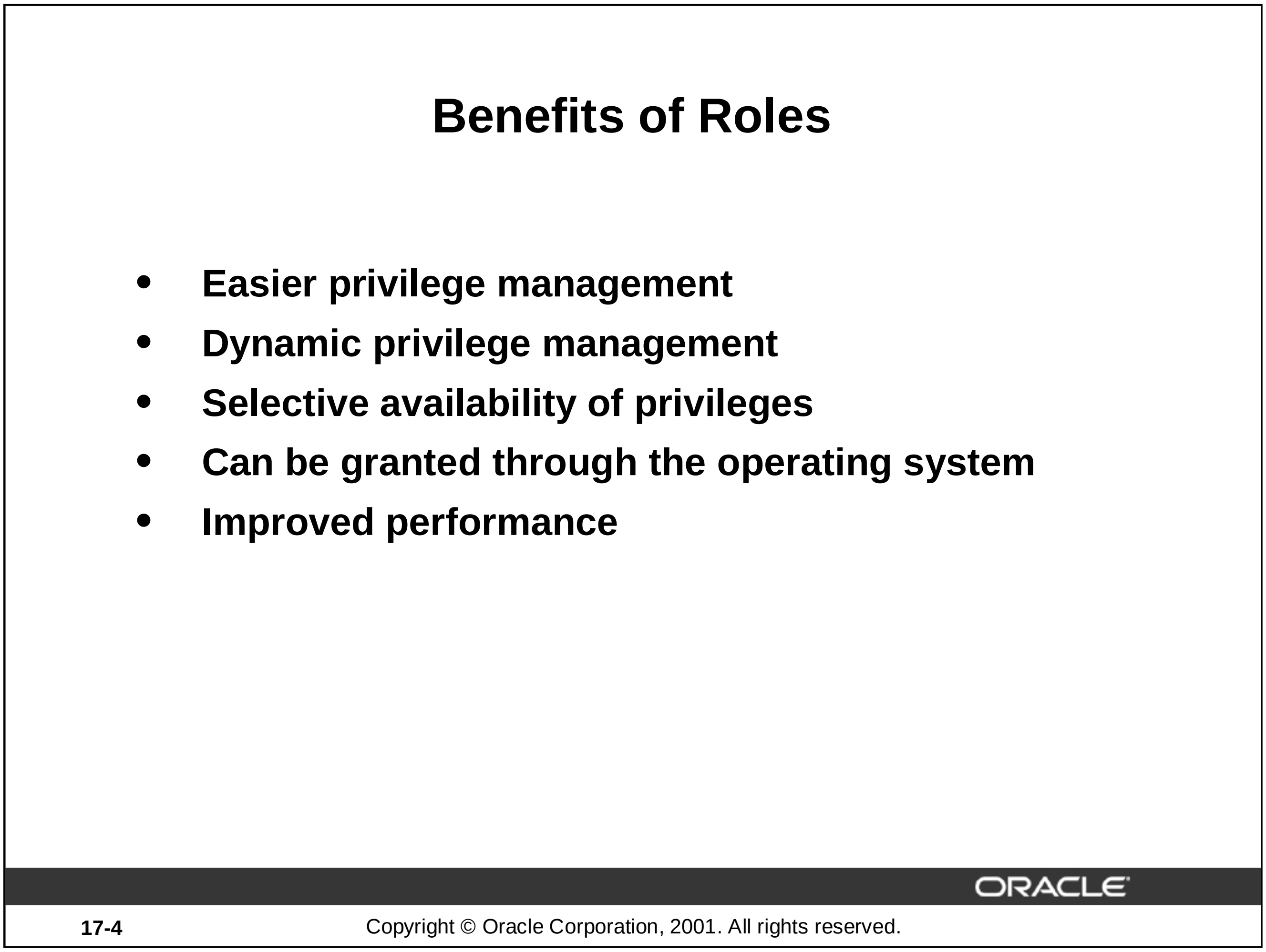
17-3

What is a Role?

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

Role Characteristics

- Granted to and revoked from users with the same commands used to grant and revoke system privileges.
- May be granted to any user or role. However, a role cannot be granted to itself and cannot be granted circularly.
- Can consist of both system and object privileges.
- May be enabled or disabled for each user granted the role.
- Can require a password to enable.
- Each role name must be unique among existing usernames and role names.
- Are not owned by anyone; are not in any schema.
- 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
- Improved performance

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 automatically and immediately acquire the modified privileges.

Selective Availability of Privileges

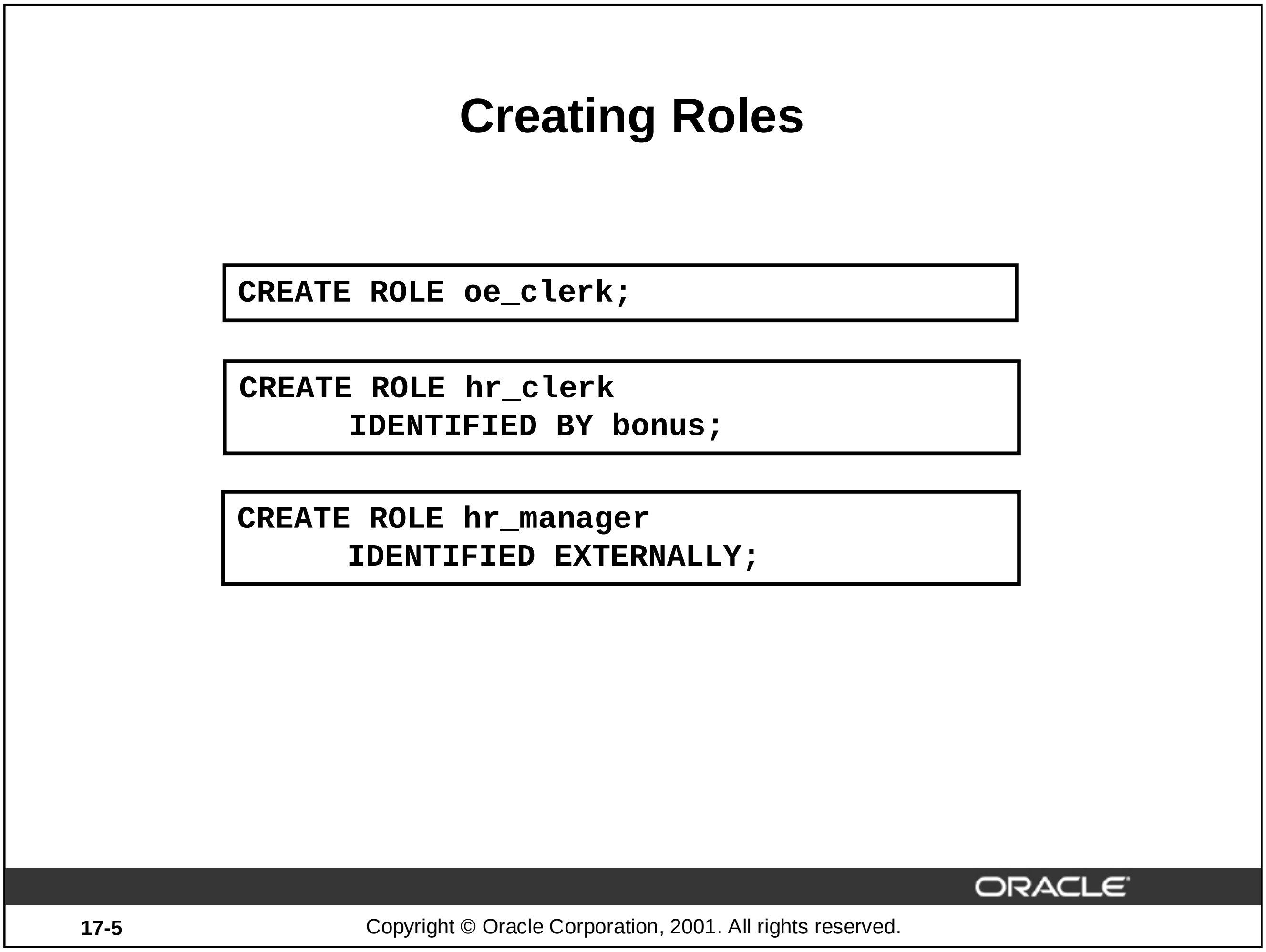
Roles can be enabled and disabled to turn privileges on and off temporarily. Enabling a role can also be used to verify that a user has been granted that role.

Can be granted Through the Operating System

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

Improved Performance

By disabling roles, there are fewer privileges to verify during statement execution. Using roles reduces the number of grants stored in the data dictionary.



Creating Roles

```
CREATE ROLE oe_clerk;
```

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

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

17-5

Creating Roles

Use the CREATE ROLE statement to create roles. CREATE ROLE system privilege is required. When you create a role that is NOT IDENTIFIED or is IDENTIFIED EXTERNALLY or BY Password, Oracle grants the role with 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 row
`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

Creating Roles (continued)

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)

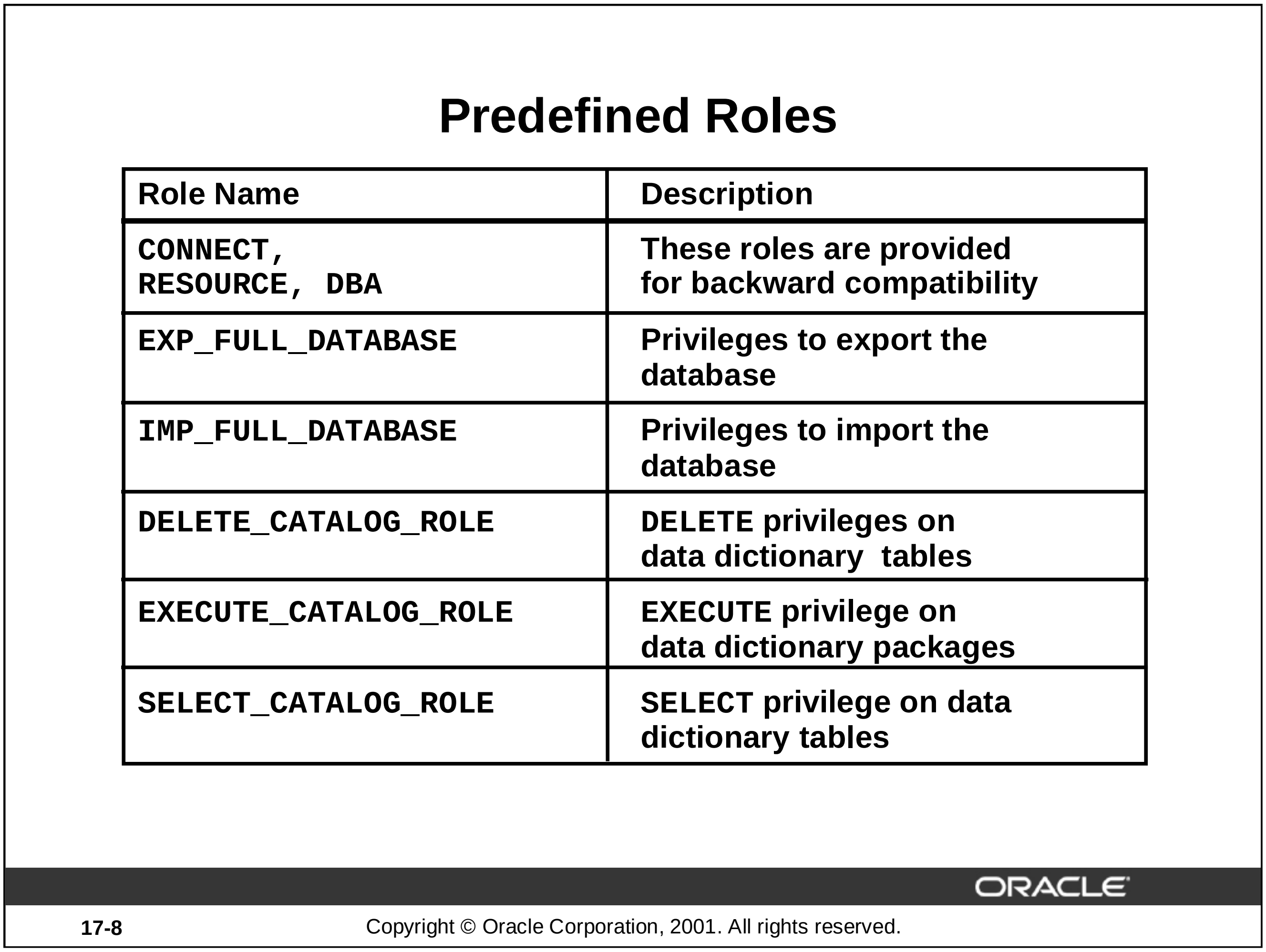
GLOBALLY indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the SET ROLE statement, or at login

Creating Roles (continued)

Using Enterprise Manager to Create a Role

1. Launch the Console:
   %oemapp console
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Right-click on the your working database and click Connect.
5. Supply the username, password and service name for your working database and click OK.
6. Expand the Security folder.
7. Right-click on Roles folder and click Create.
8. Enter a name for the role.
9. Optionally, GRANT roles and privileges to the new role.
10. Click Create.

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



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

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.

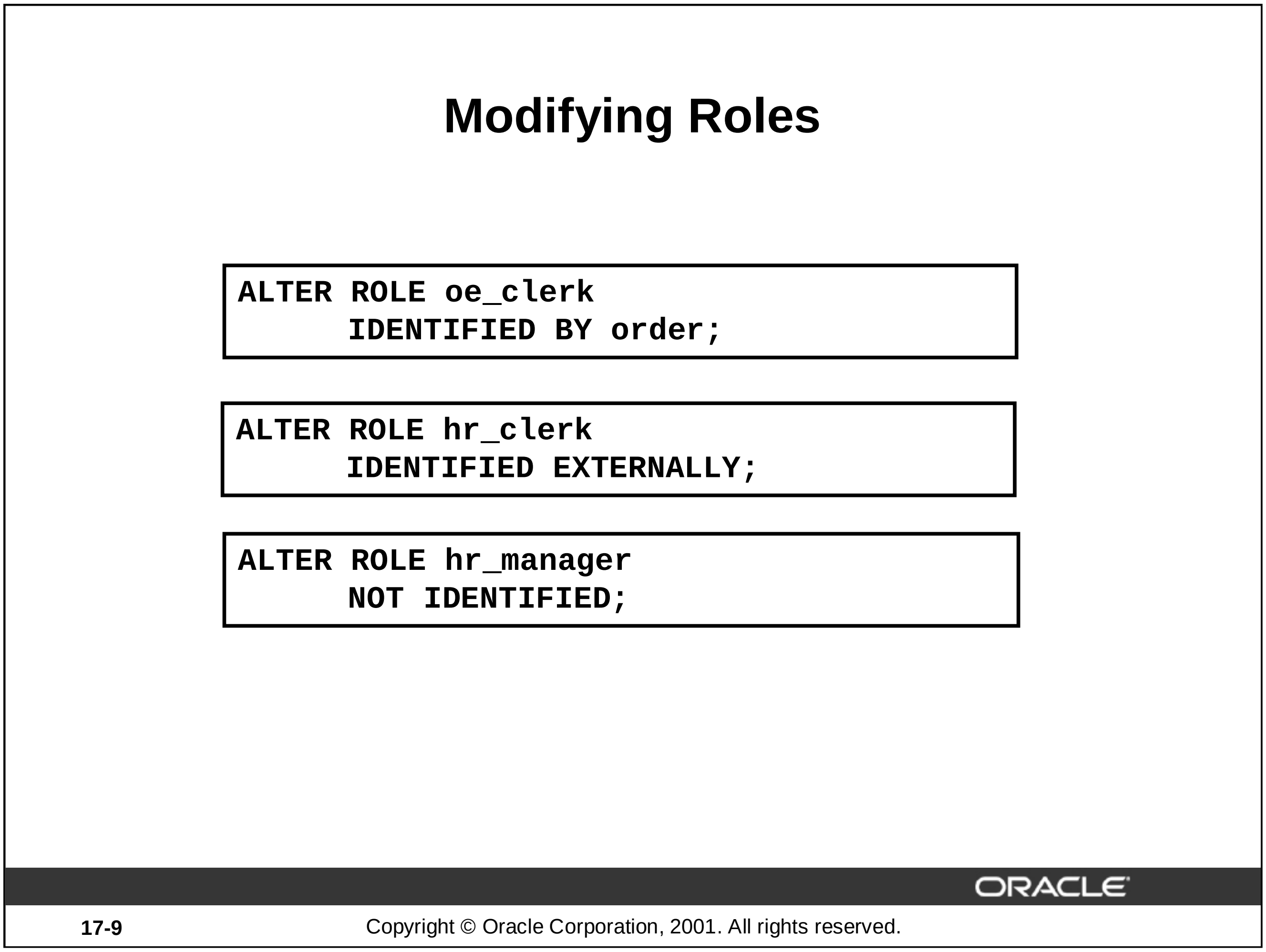
The EXP_FULL_DATABASE and IMP_FULL_DATABASE roles are provided for convenience in using the Import and Export utilities.

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 obsoleted but mainly kept for release 8.0 compatibility.



Modifying Roles

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

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

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

Modifying Roles

A role can only be modified to change its authentication method. You must have either been granted the role with the `ADMIN` option or have `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

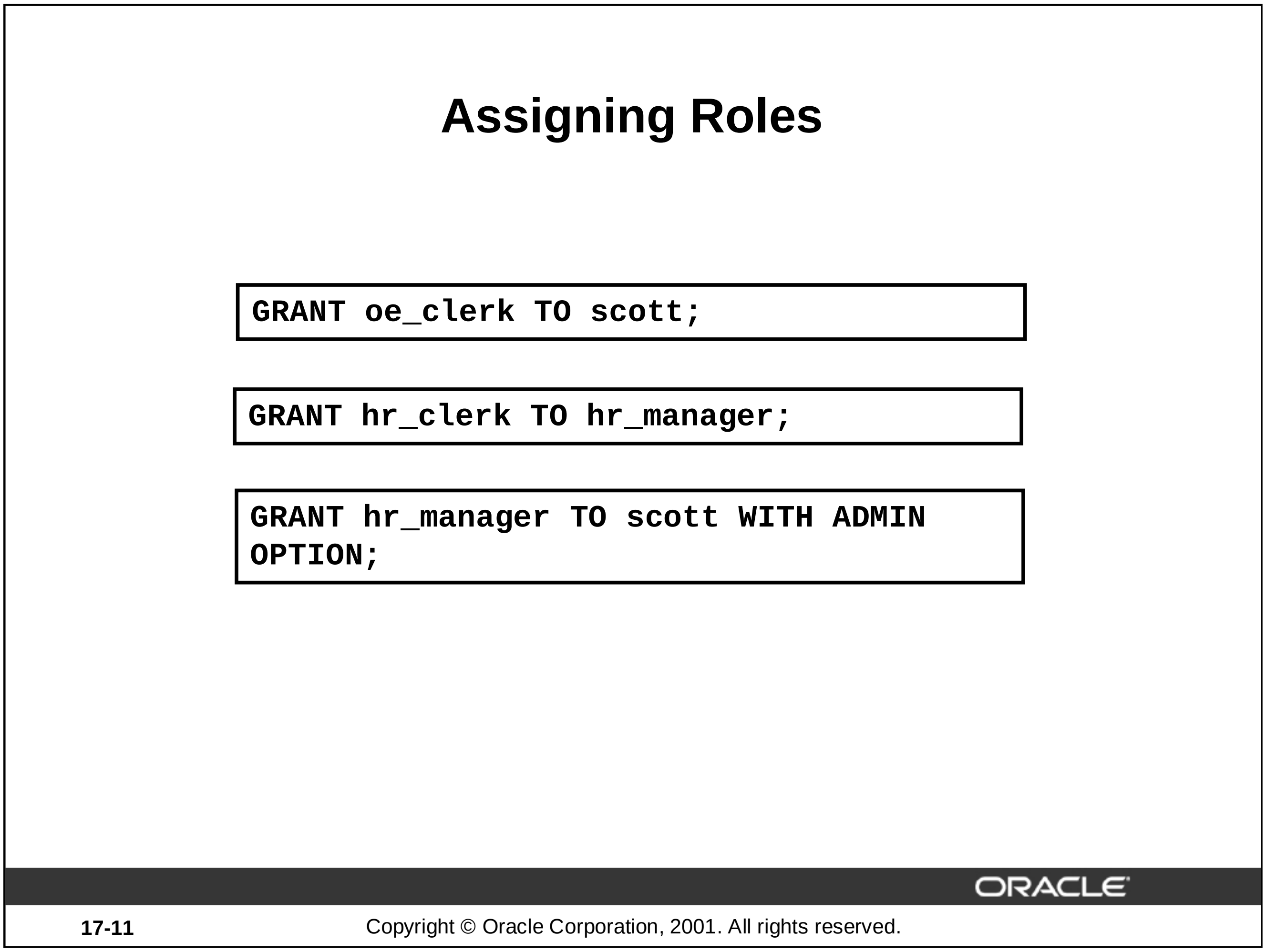
`GLOBALLY` indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the `SET ROLE` statement, or at login.

Modifying Roles (continued)

Using Enterprise Manager to Modify Roles

1. Launch the Console:
   ```
   %oemapp console
   ```
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Right-click on the your working database and click `Connect`.
5. Supply the username, password and service name for your working database and click `OK`.
6. Expand the `Security` folder.
7. Expand the `Roles` folder and select the role to be modified.
8. Make the requisite modifications and click `Apply`.

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



Assigning Roles

```
GRANT oe_clerk TO scott;
```

```
GRANT hr_clerk TO hr_manager;
```

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

ORACLE

17-11

Assigning Roles

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

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

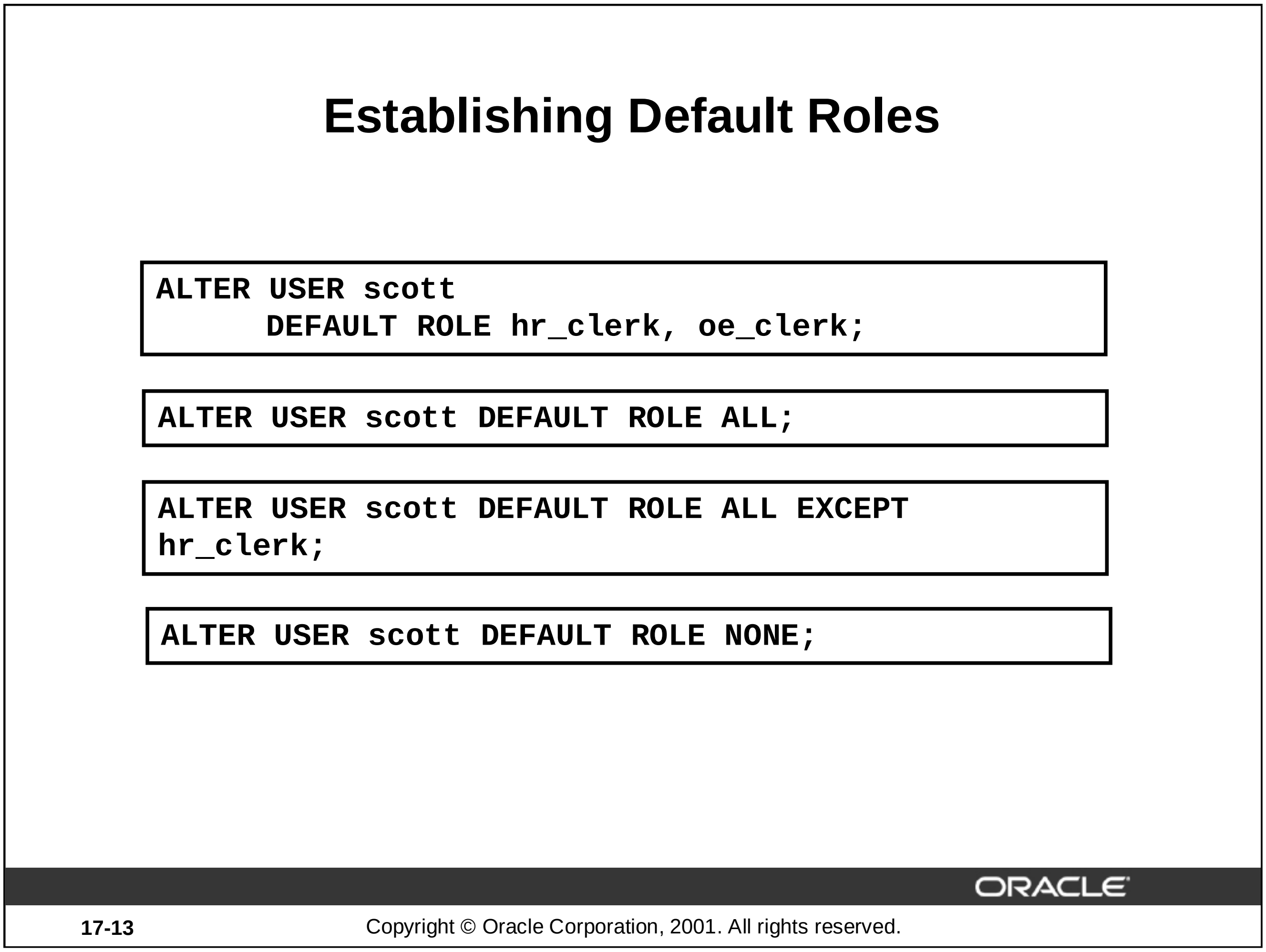
where:

`role`	is a role to be granted or a role receiving the role granted
	user is a user receiving a role
`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, 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 initialization parameter `MAX_ENABLED_ROLES`.



Establishing Default Roles

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

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

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

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

17-13

Default Roles

A user may have many roles assigned. 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:

- 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}
```

Default Roles (continued)

where:

`user` is the name of the user granted the roles

`role` is the role to be made the default role for the user

`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

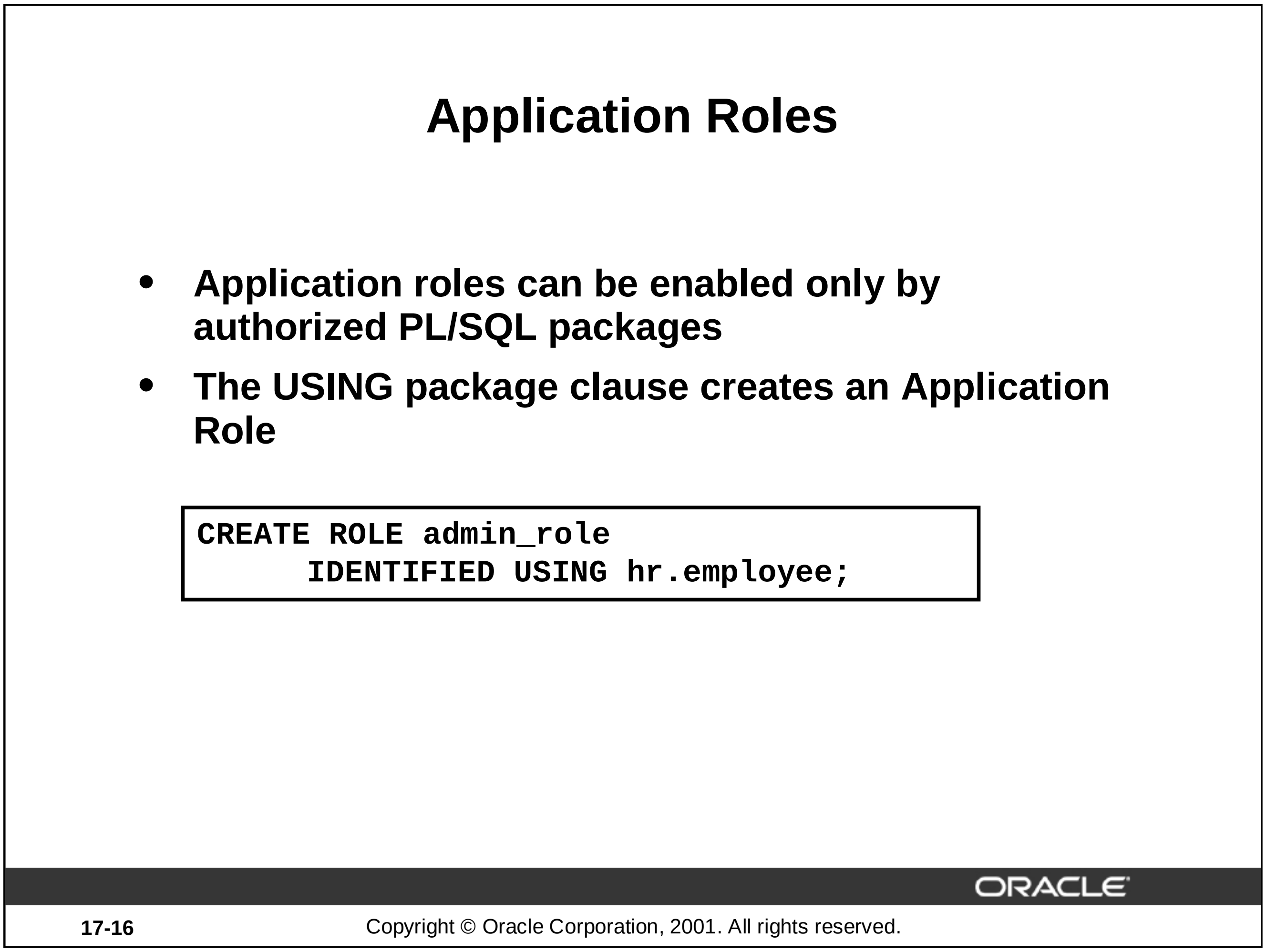
`NONE` makes none of the roles granted to the user default roles (The only privileges that the user has at login are those privileges assigned directly to the user.)

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

Default Roles (continued)

Using Oracle Enterprise Manager to Assign Roles

1. Launch the Console:
   %oemapp console
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Expand the Security folder.
5. Expand the Users folder and select the user for whom a role is to be assigned
6. Click on the Role tab on the detail side of he Console.
7. Select the Role to be assigned and click the down arrow.
8. Optionally assign the role with Admin option
9. Click Apply.



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;
```

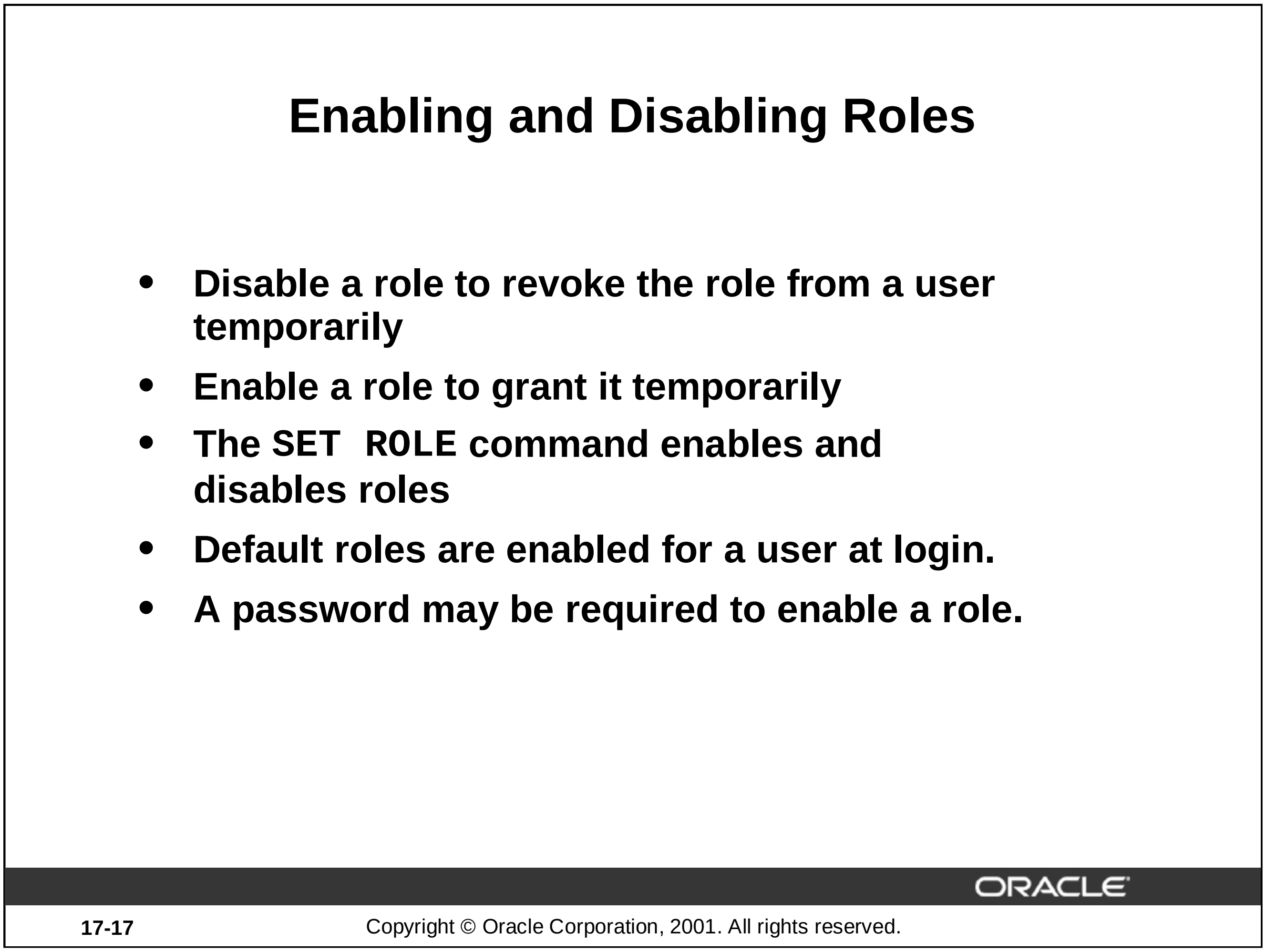
17-16

Application Roles

The `USING` package clause in the `CREATE ROLE` statement creates an application role. Application role can be enabled only by applications 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.

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

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



Enabling and Disabling Roles

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

17-17

Enabling and Disabling Roles

Enable or disable roles to activate and deactivate temporarily 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 will 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.

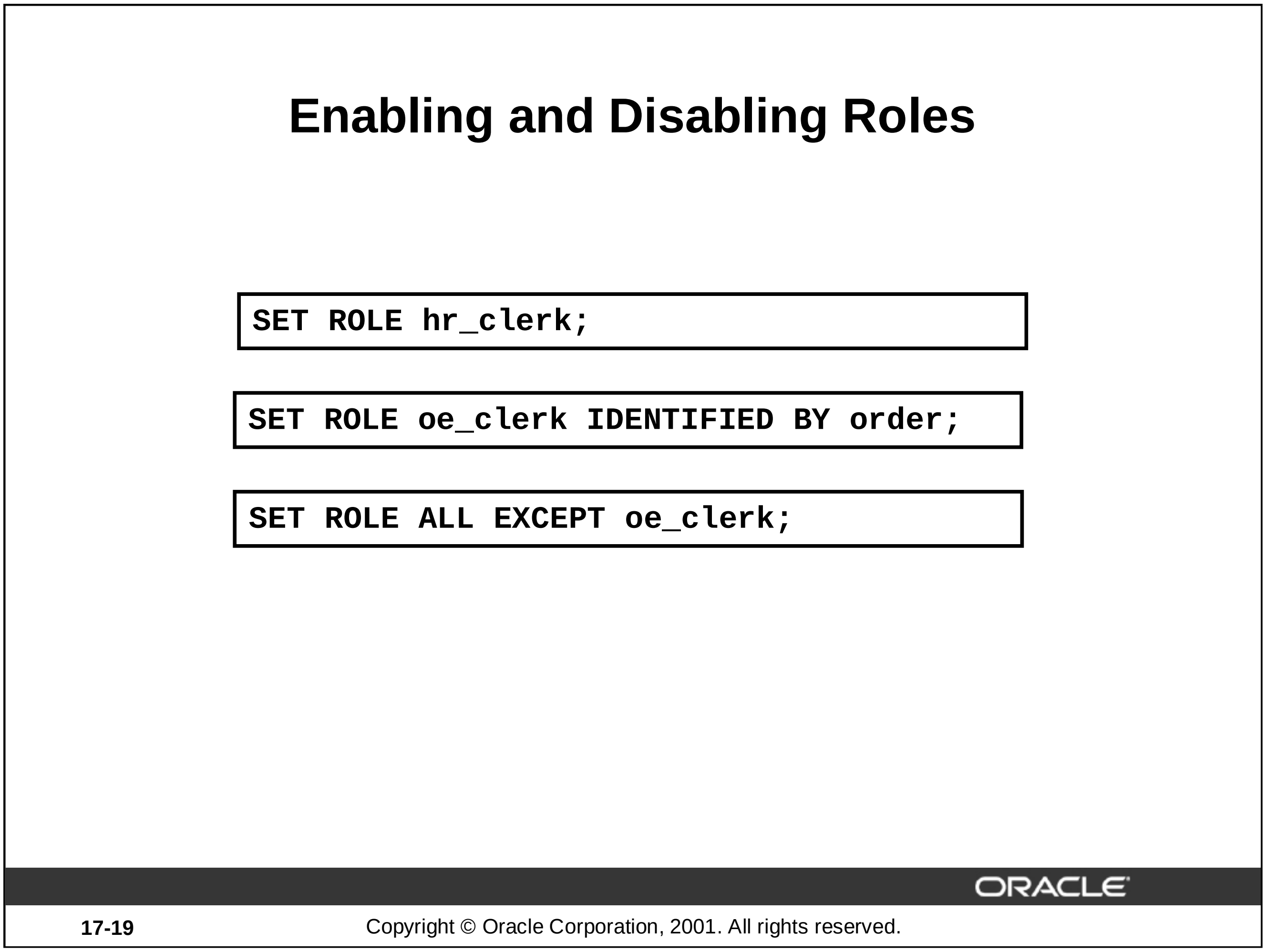
Enabling and Disabling Roles (continued)

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.

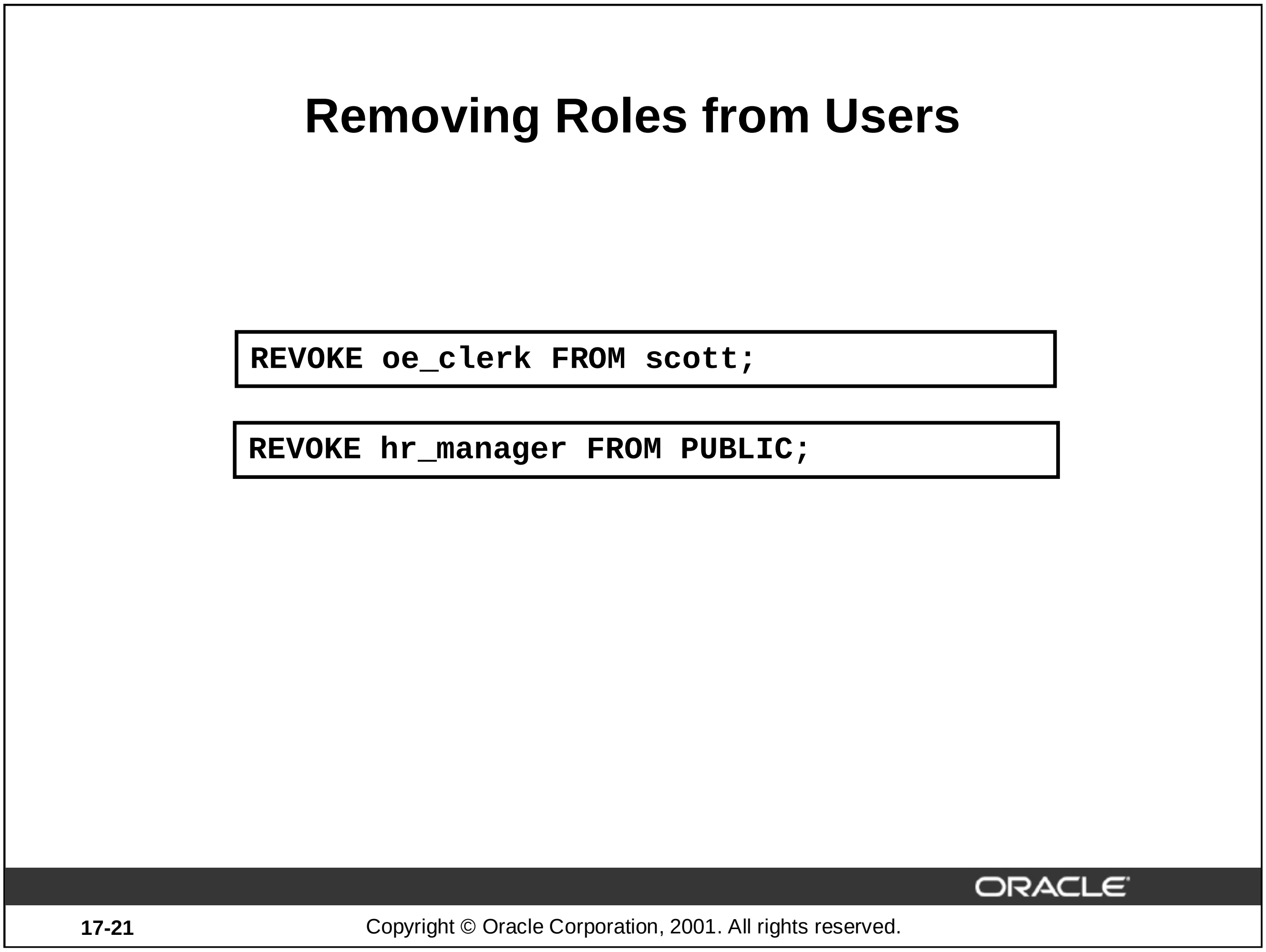
```
SET ROLE {role [ IDENTIFIED BY password ]
        [, role [ IDENTIFIED BY password ]]...
        | ALL [ EXCEPT role [, role ] ...]
        | NONE }
```

where:

role	is the name of the role
IDENTIFIED BY password	provides the password required when enabling the role
ALL	enables all roles 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.)

Enabling and Disabling Roles (continued)

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



Removing Roles from Users

```
REVOKE oe_clerk FROM scott;
```

```
REVOKE hr_manager FROM PUBLIC;
```

Removing 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 can revoke any role.

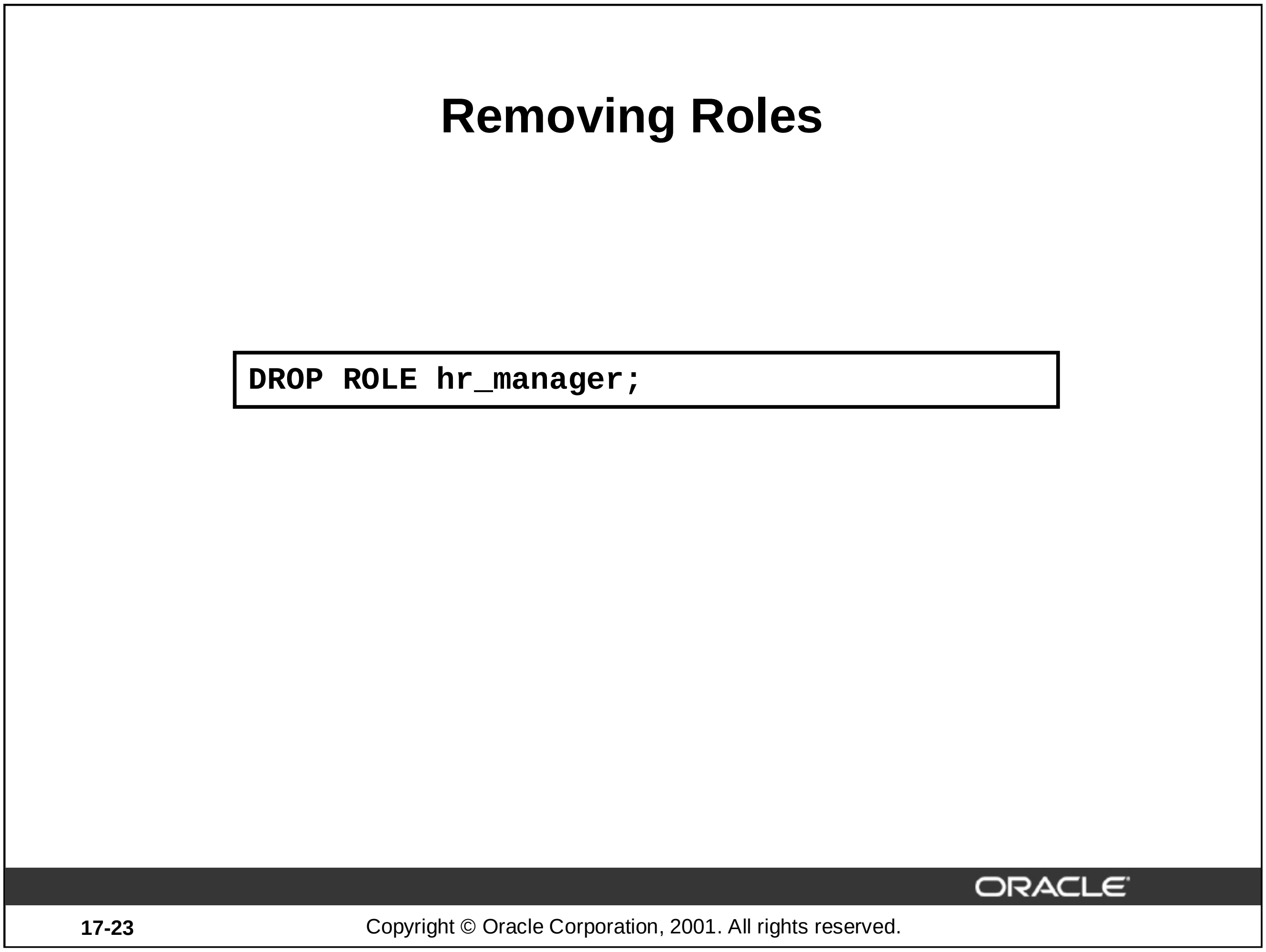
```
REVOKE role [, role ]...
  FROM   {user|role|PUBLIC}
      [, {user|role|PUBLIC} ]...
where:
```

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

Removing Roles from Users (continued)

How to use Oracle Enterprise Manager to Revoke a Role

1. Launch the Console:
   %oemapp console
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Expand the Security folder.
5. Expand the Users folder and select the user for whom a Role is to be revoked.
6. Click on the Role tab on the detail side of the Console.
7. Select the role to be revoked from the ‘Granted’ box and click on the up arrow. Click Apply.



Removing Roles

```
DROP ROLE hr_manager;
```

17-23

Removing Roles

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

```
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.

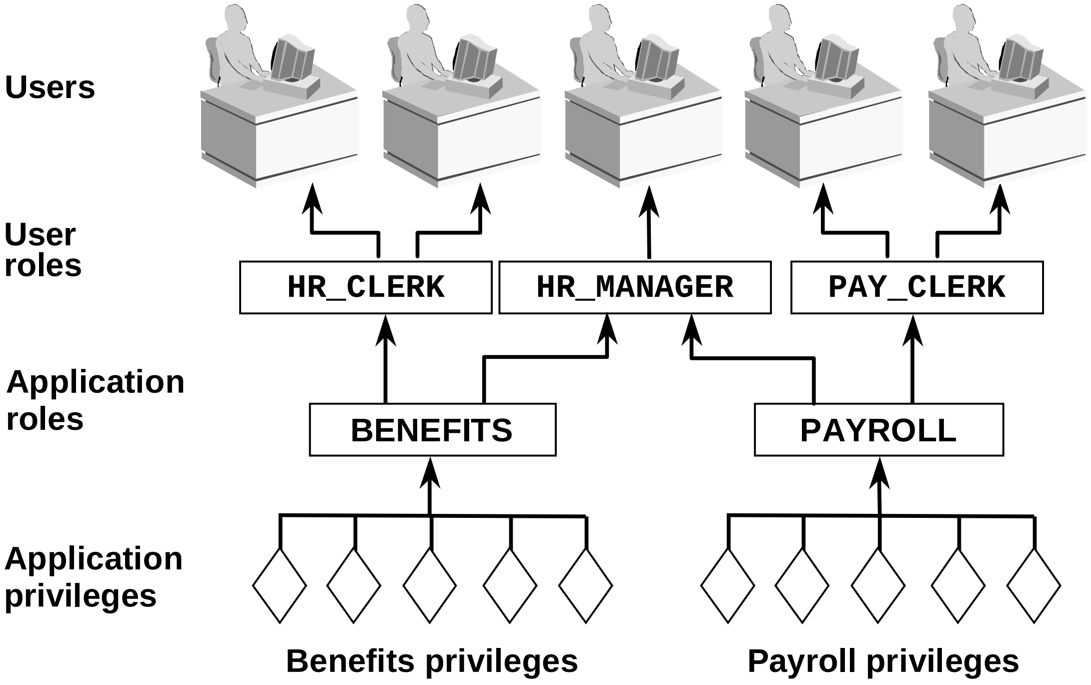
You must have been granted the role with `ADMIN OPTION` or have the `DROP ANY ROLE` system privilege to drop the role.

Removing Roles (continued)

How to use Oracle Enterprise Manager to Remove a Role

1. Launch the Console:
   %oemapp console
2. Choose to Launch the Console standalone.
3. Expand your working database from the databases folder
4. Expand the Security folder.
5. Expand the Roles folder.
6. Right-click on the role to be removed and select Remove. Or, select the role to be removed and select Remove from the Object menu.

Guidelines for Creating Roles



Users

User roles

HR_CLERK

HR_MANAGER

PAY_CLERK

Application roles

BENEFITS

PAYROLL

Application privileges

Benefits privileges

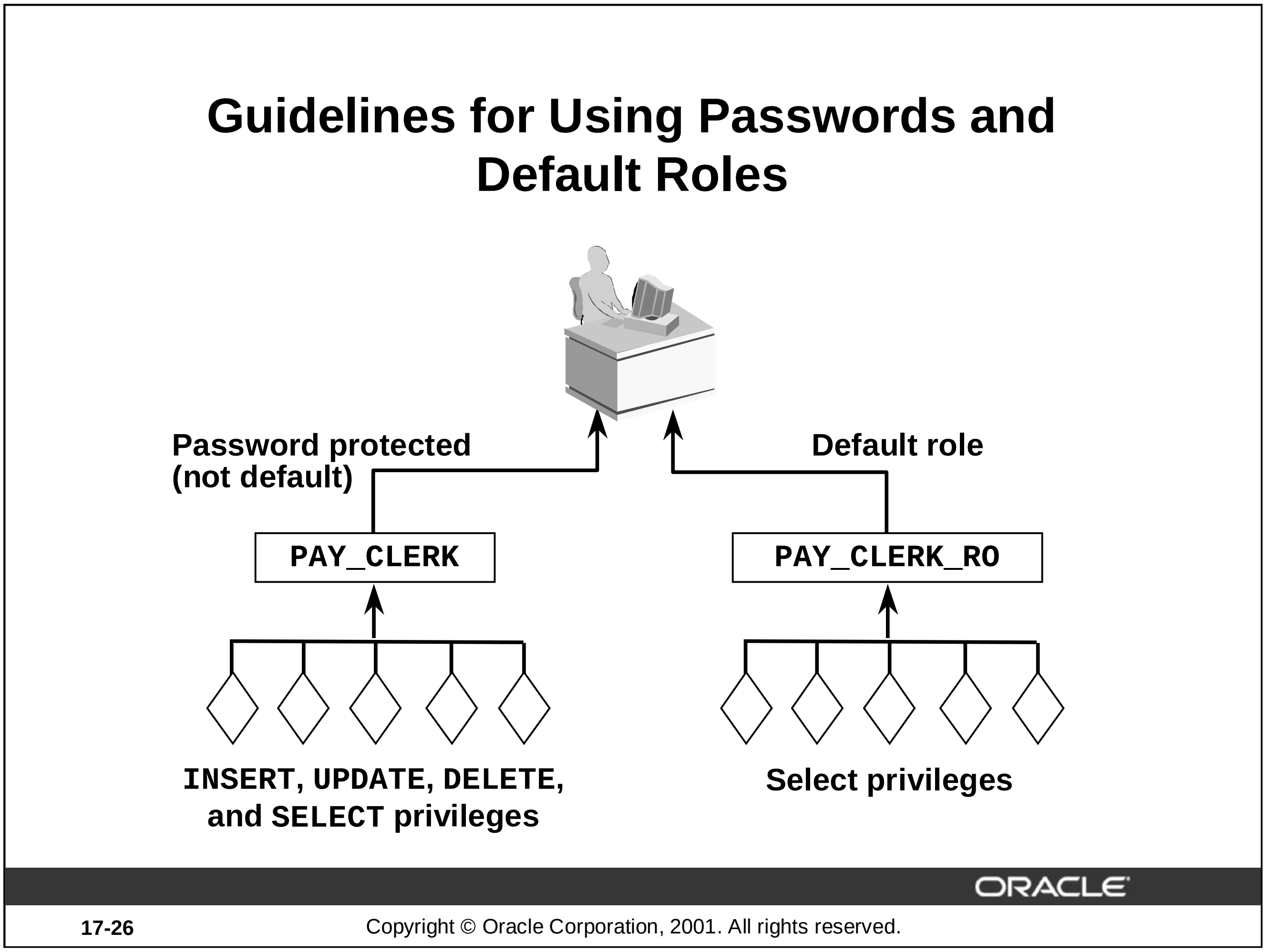
Payroll privileges

Guidelines for Creating Roles

Because a role includes the privileges necessary to perform a task, the role name is usually an application task or a job title. The example above uses both application tasks and job titles for role names.

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 application role, PAYROLL. All of the users that are currently performing this task will receive the new privileges.



Guidelines for Using Passwords and Default Roles

Password protected (not default)

Default role

PAY_CLERK

PAY_CLERK_RO

INSERT, UPDATE, DELETE, and SELECT privileges

Select privileges

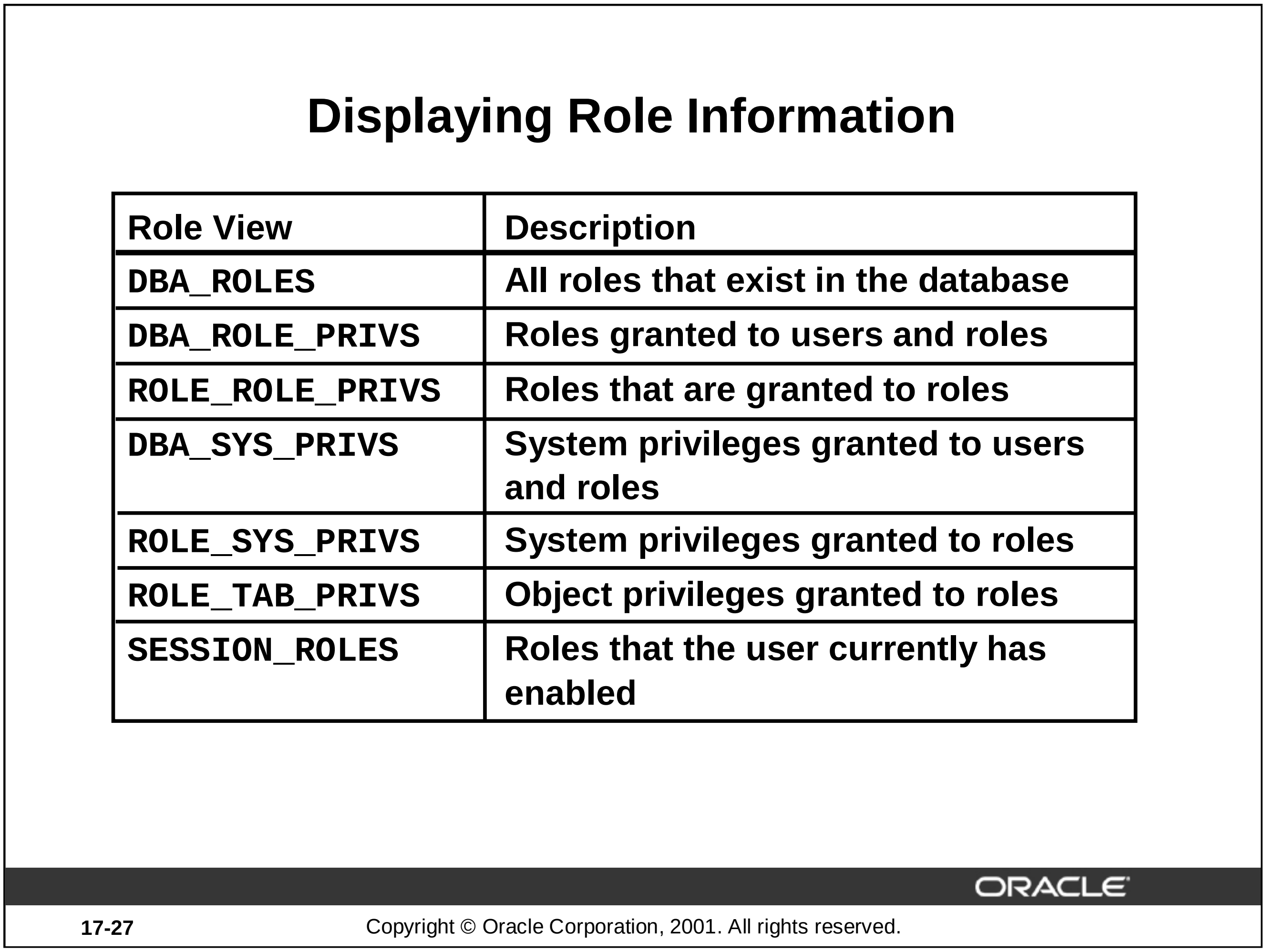
17-26

Using Passwords

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 above

- The DBA has granted the user two roles, PAY_CLERK and PAY_CLERK_RO.
- The PAY_CLERK has been granted all of the privileges needed 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 on 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.



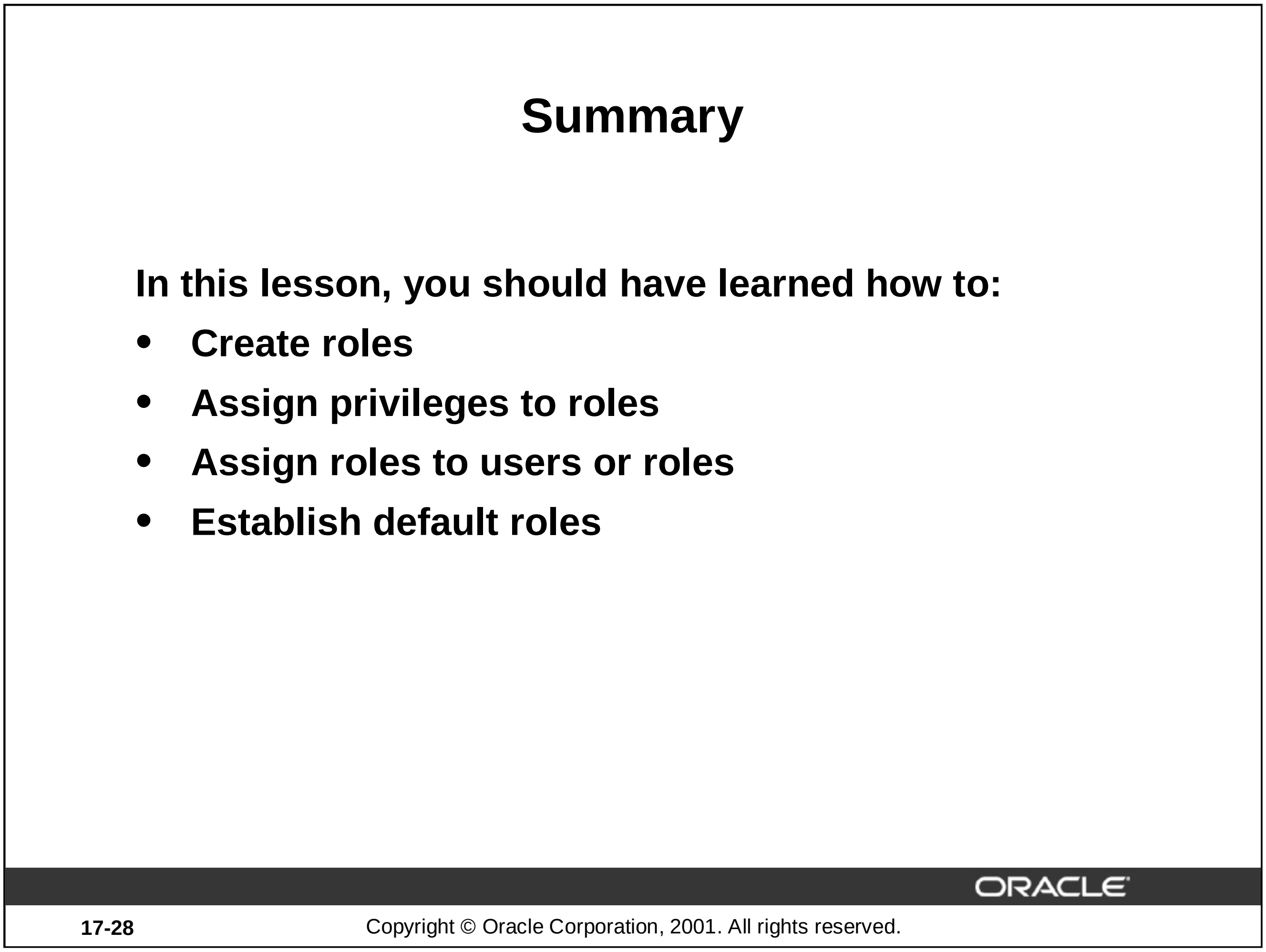
Displaying Role Information

Role View	Description
DBA_ROLES	All roles that exist in the database
DBA_ROLE_PRIVS	Roles granted to users and roles
ROLE_ROLE_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

Query Role Information

Many of the data dictionary views that contain information on privileges granted to users also contain information on privileges to roles.

```
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

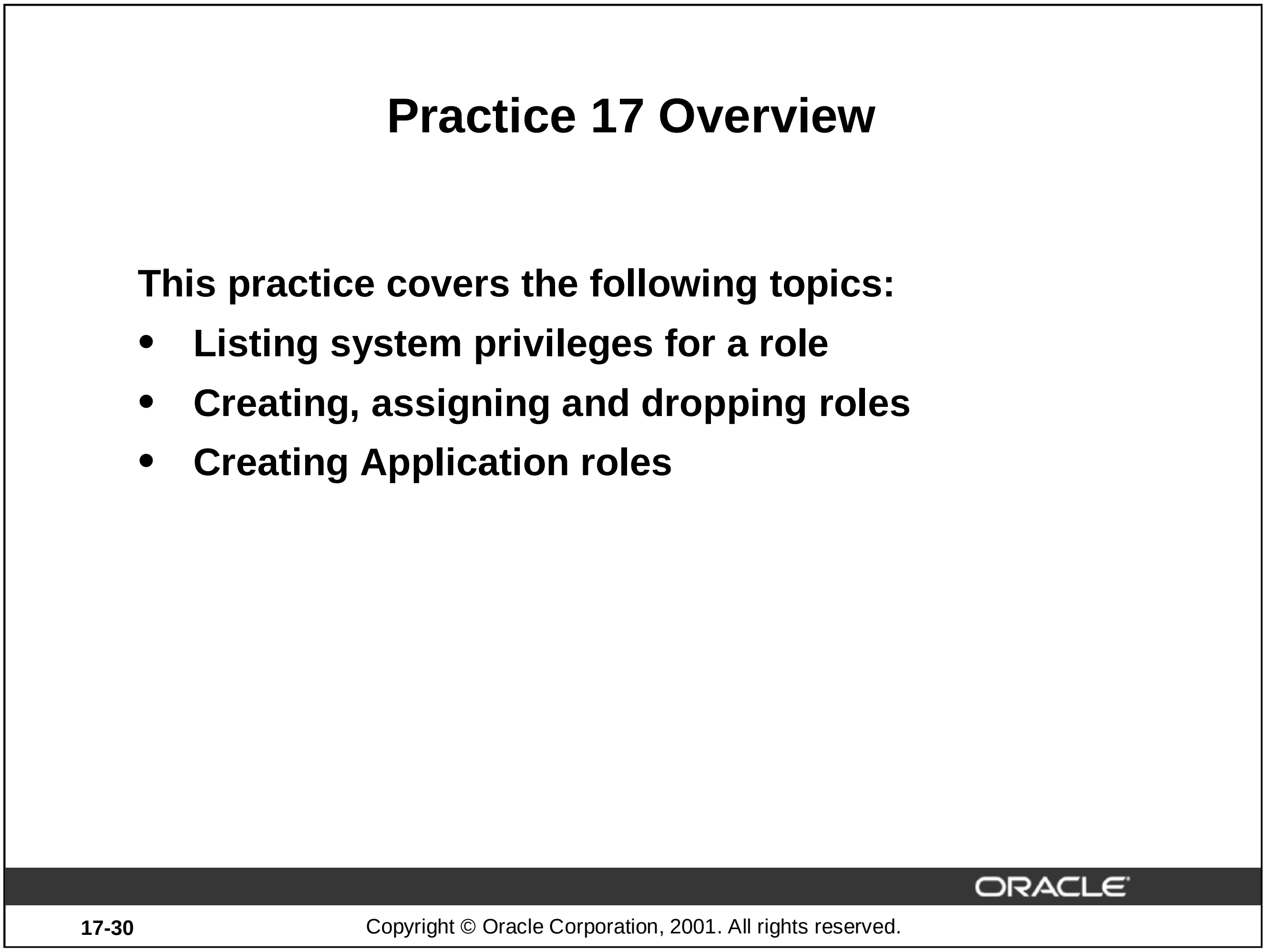
In this lesson, you should have learned how to:

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

ORACLE

Quick Reference

Context	Reference
Initialization parameters	MAX_ENABLED_ROLES
Data dictionary views	DBA_ROLES DBA_ROLE_PRIVS DBA_SYS_PRIVS ROLE_ROLE_PRIVS ROLE_SYS_PRIVS ROLE_TAB_PRIVS SESSION_ROLES
Commands	CREATE ROLE ALTER ROLE DROP ROLE SET ROLE ALTER USER…DEFAULT ROLES GRANT REVOKE



Practice 17 Overview

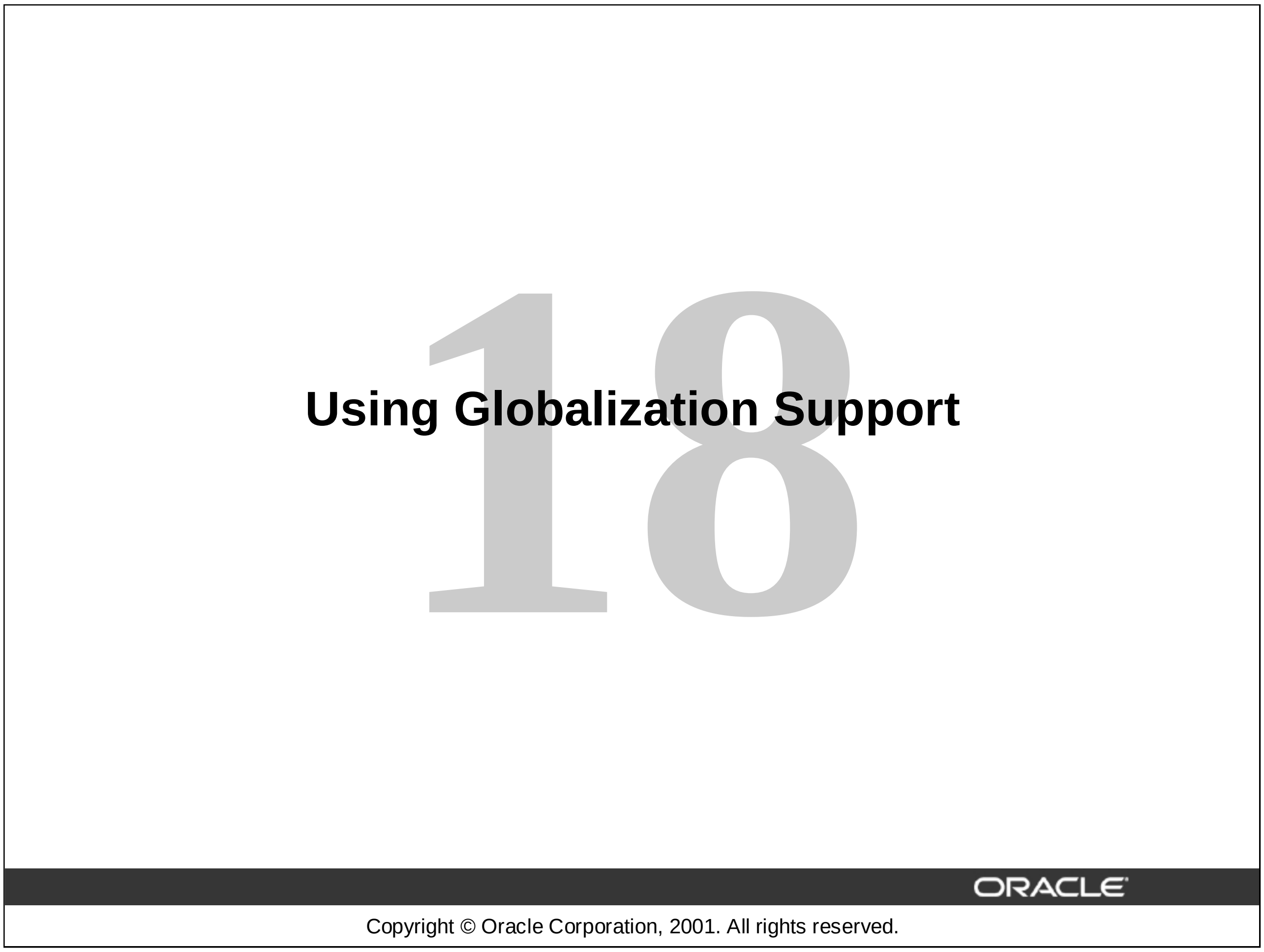
This practice covers the following topics:

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

Practice 17: Managing Roles

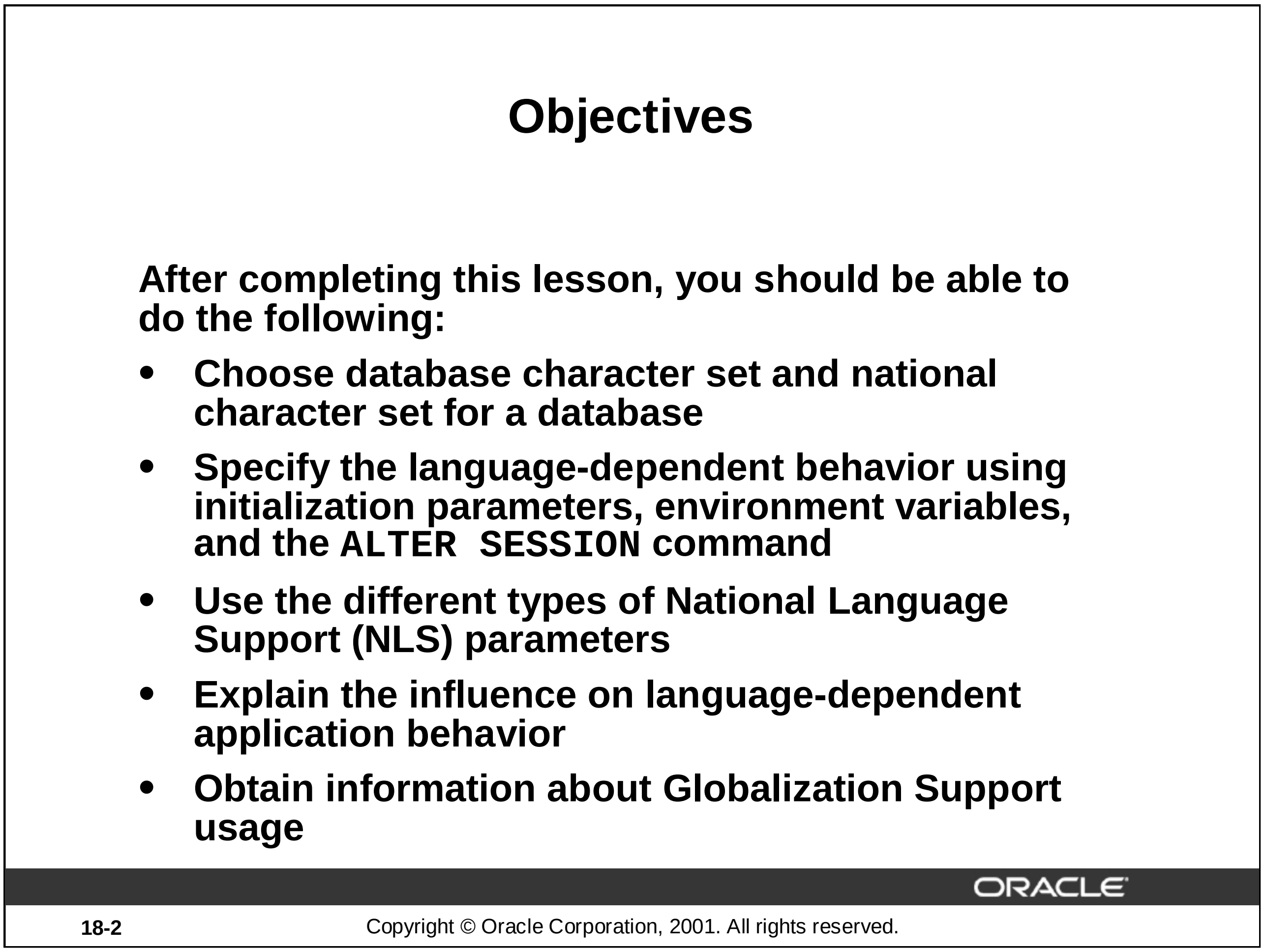
1. Examine the data dictionary view and list the system privileges of the RESOURCE role.
2. Create a role called DEV, which enables a user to create a table, create a view and select from Kay's CUSTOMERS table.
3. **a** Assign the RESOURCE and DEV roles to Bob, but make only the RESOURCE role to be automatically 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.
5. As SYSTEM, try to create a view CUST_VIEW on Kay's CUSTOMERS table. What happens and why?
6. As user Emi grant select on customers to SYSTEM. As SYSTEM try to create view CUST_VIEW on Emi's CUSTOMERS table. What happens and why?



18

Using Globalization Support



Objectives

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

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

18-2

Globalization Support Features

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

Database

Globalization Support

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

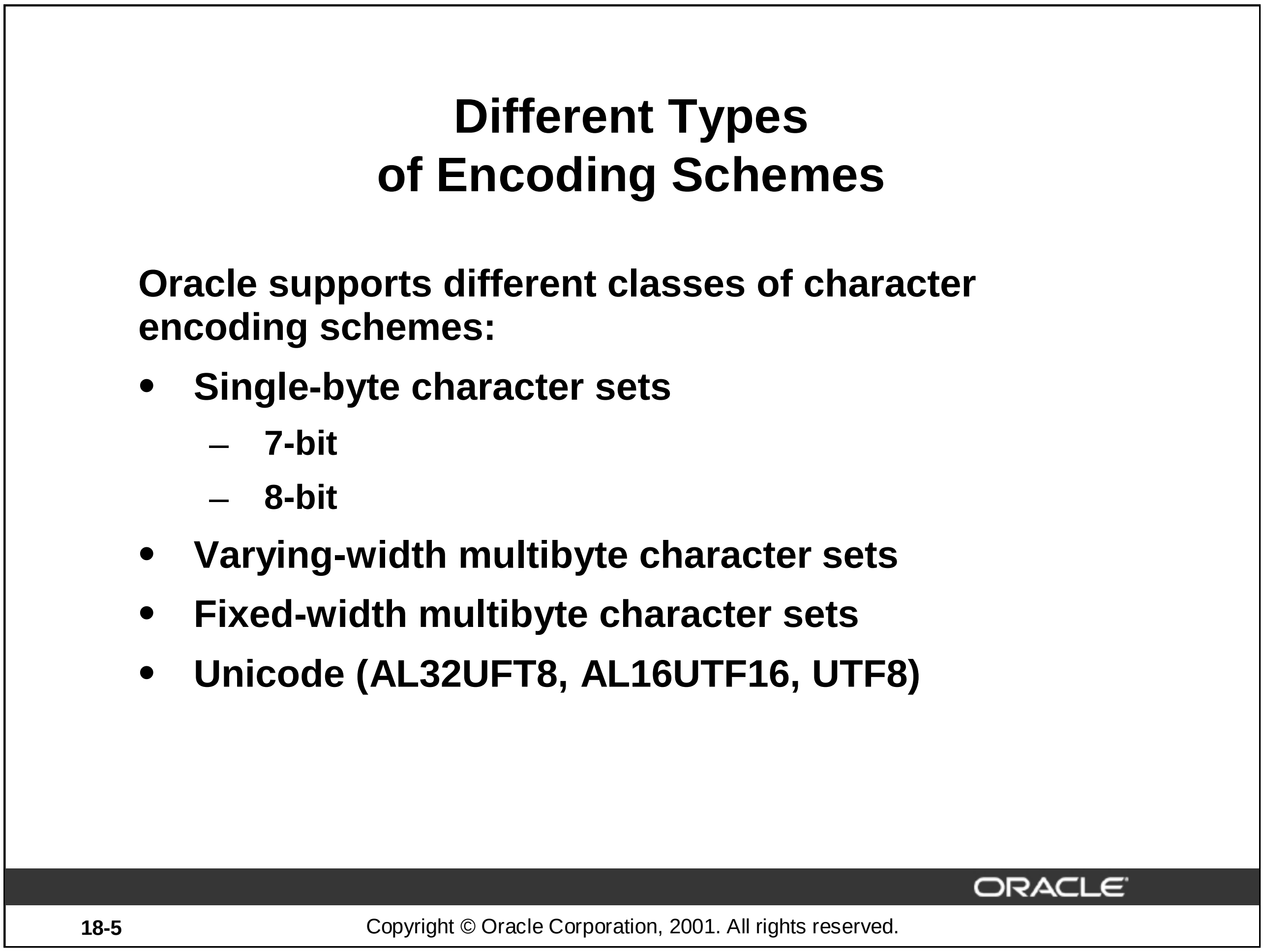
Oracle currently supports 57 languages, 88 territories, 84 linguistic sorts (71 monolingual and 13 multilingual), and 235 encoded character sets.

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

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

Globalization Support (continued)

- Users can interact, store, process, and retrieve data in their native language, 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. The Oracle products are translated into 30 different languages.
- Date and time formats can be expressed according to 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 the ISO conventions. Credit and debit symbols also differ from location to location.



Different Types of Encoding Schemes

Oracle supports different classes of character encoding schemes:

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

18-5

Character 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.

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.

This provides the benefits of 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)

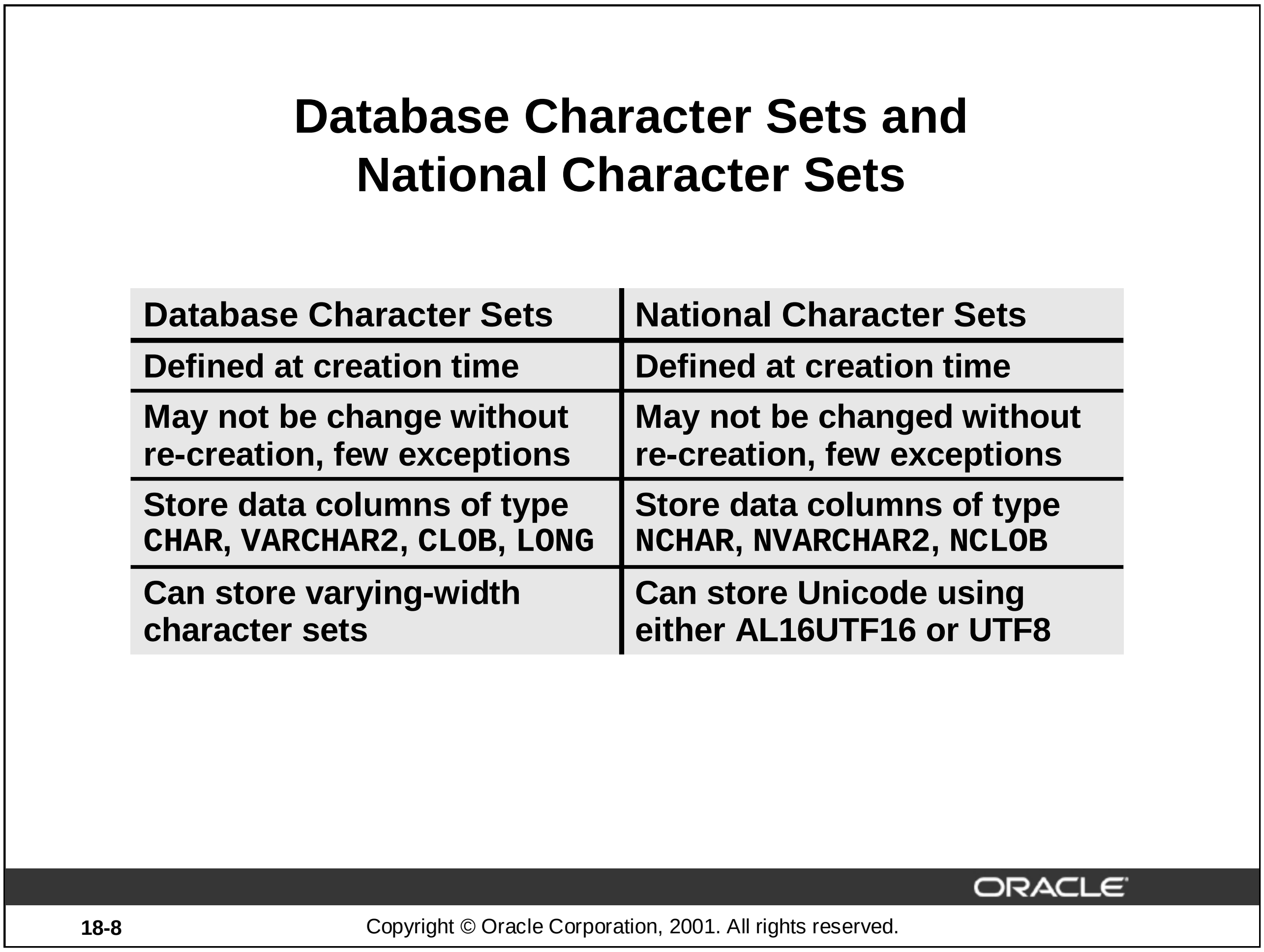
Unicode Character Set

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. UTF8 is a superset of ASCII, hence this makes database character set migration easier when upgrading ASCII based characters sets to Unicode.

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



Database Character Sets and National Character Sets

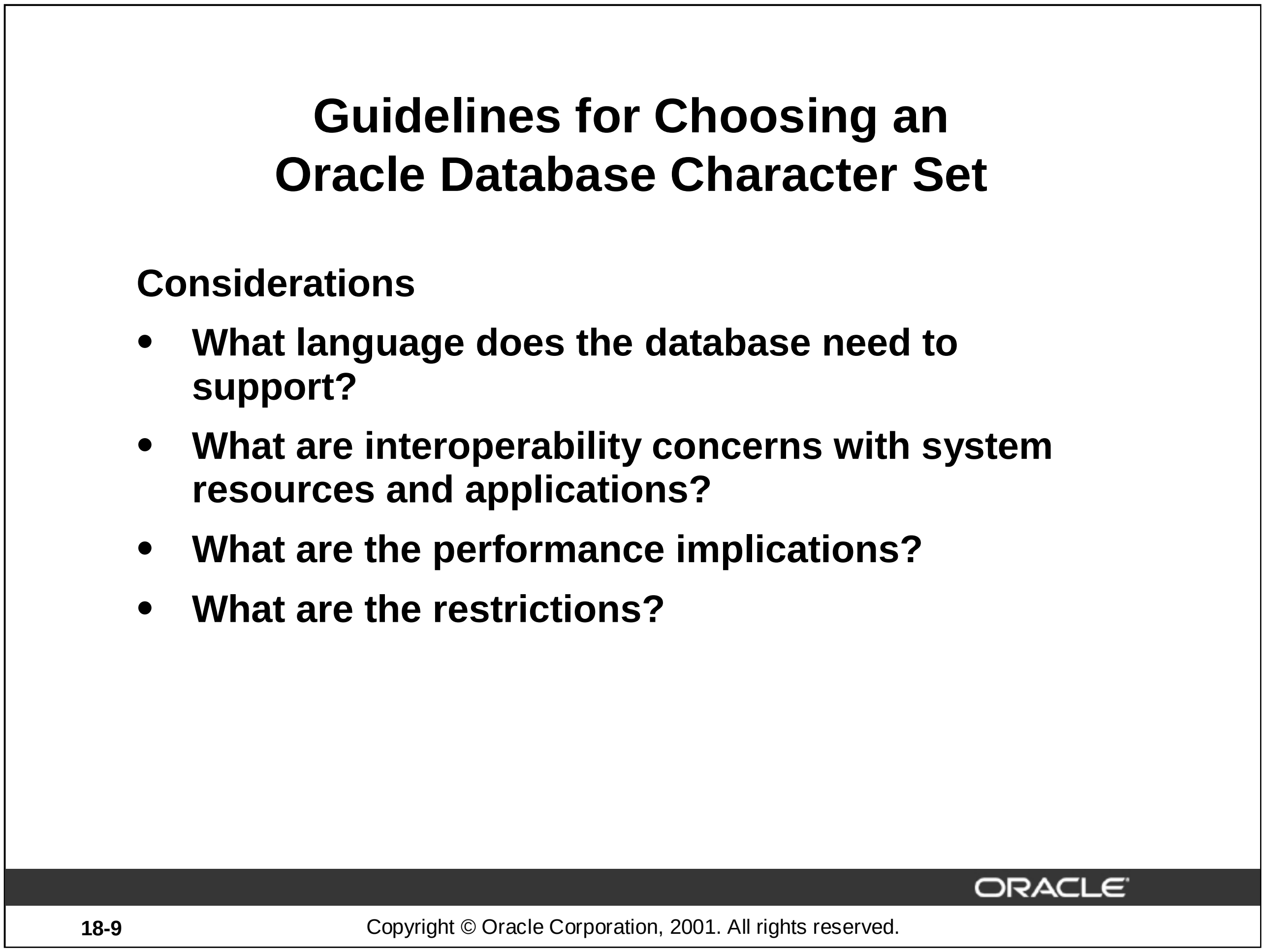
Database Character Sets	National Character Sets
Defined at creation time	Defined at creation time
May not be change without re-creation, few exceptions	May not be changed without re-creation, few exceptions
Store data columns of type `CHAR`, `VARCHAR2`, `CLOB`, `LONG`	Store data columns of type `NCHAR`,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

Character Set Types

The `CREATE DATABASE` statement has the `CHARACTER SET` clause and the additional optional clause `NATIONAL CHARACTER SET` to 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` datatypes (`NCHAR`, `NVARCHAR2`, and `NCLOB`) are Unicode Datatypes in Oracle9*i*.



Guidelines for Choosing an Oracle Database Character Set

Considerations

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

18-9

What language does the database need to support?

Several character sets may meet your current language requirements, but 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?

While 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 language(s) desired and application software must also be compatible with a particular character set.

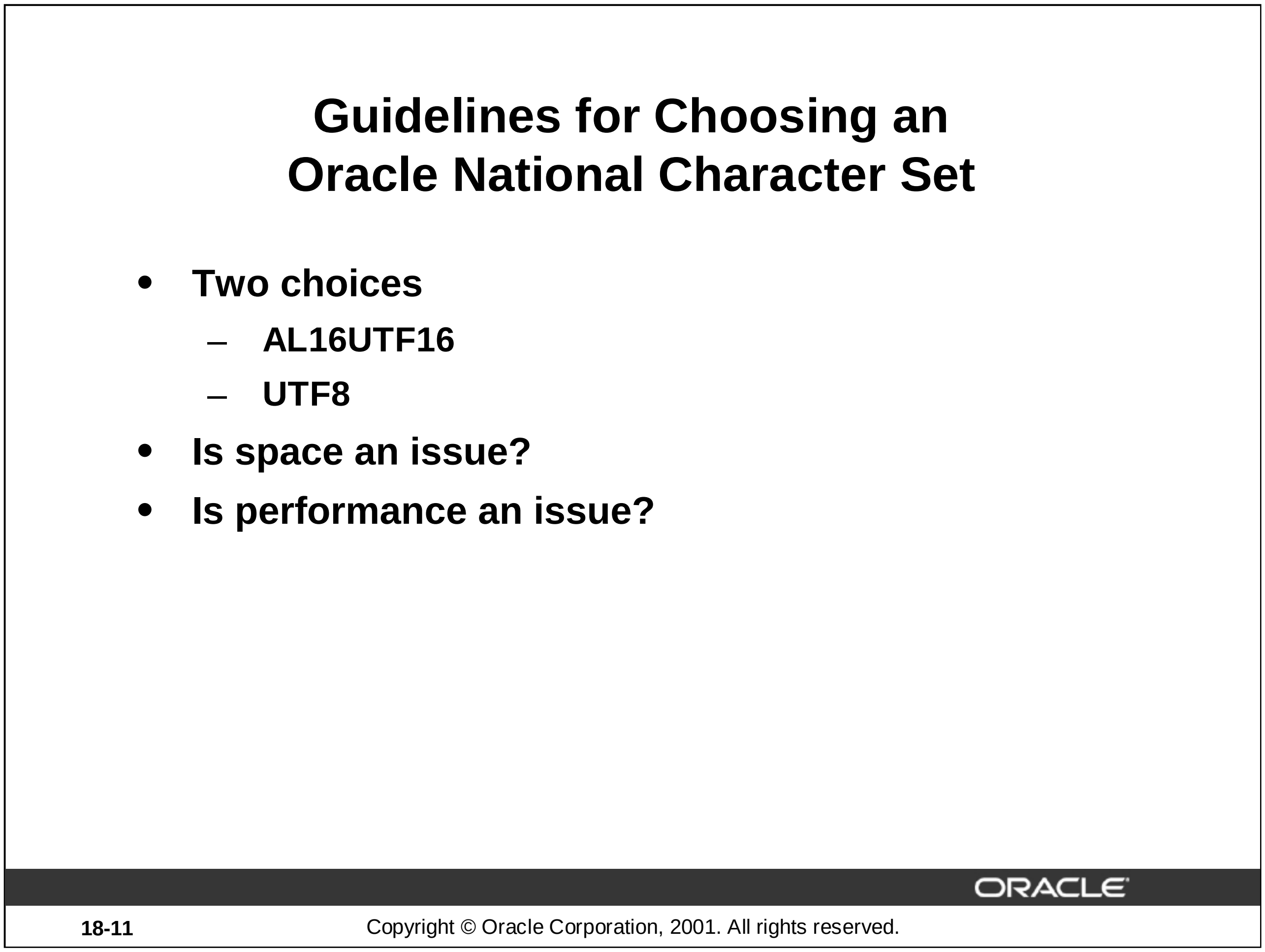
If you choose a character set that is different from what is available on the operating system, Oracle can convert the operating system character set to the database character set. However, there is some character set conversion overhead, and you need to make sure that the operating system character set has an equivalent character repertoire to avoid any possible data loss.

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 also are the most efficient in terms of space requirements. However, single-byte character sets limit how many languages you can use.

What are the restrictions?

You cannot choose a database character set that is fixed-width multibyte.



Guidelines for Choosing an Oracle National Character Set

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

18-11

Choosing a 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 1 to 3 byte Unicode character set.

European characters, in UTF8, are stored in 1 to 2 bytes and thus save space over AL16UTF16 which stores characters in 2 bytes. Asian characters, in UTF8, are stored in 3 bytes and require more space than AL16UTF16.

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

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

18-12

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` datatypes (`CHAR`, `VARCHAR2`, `CLOB`, and `LONG`). It is not necessary to recode for the SQL `NCHAR` datatypes.

Easy Data Migration from ASCII based data

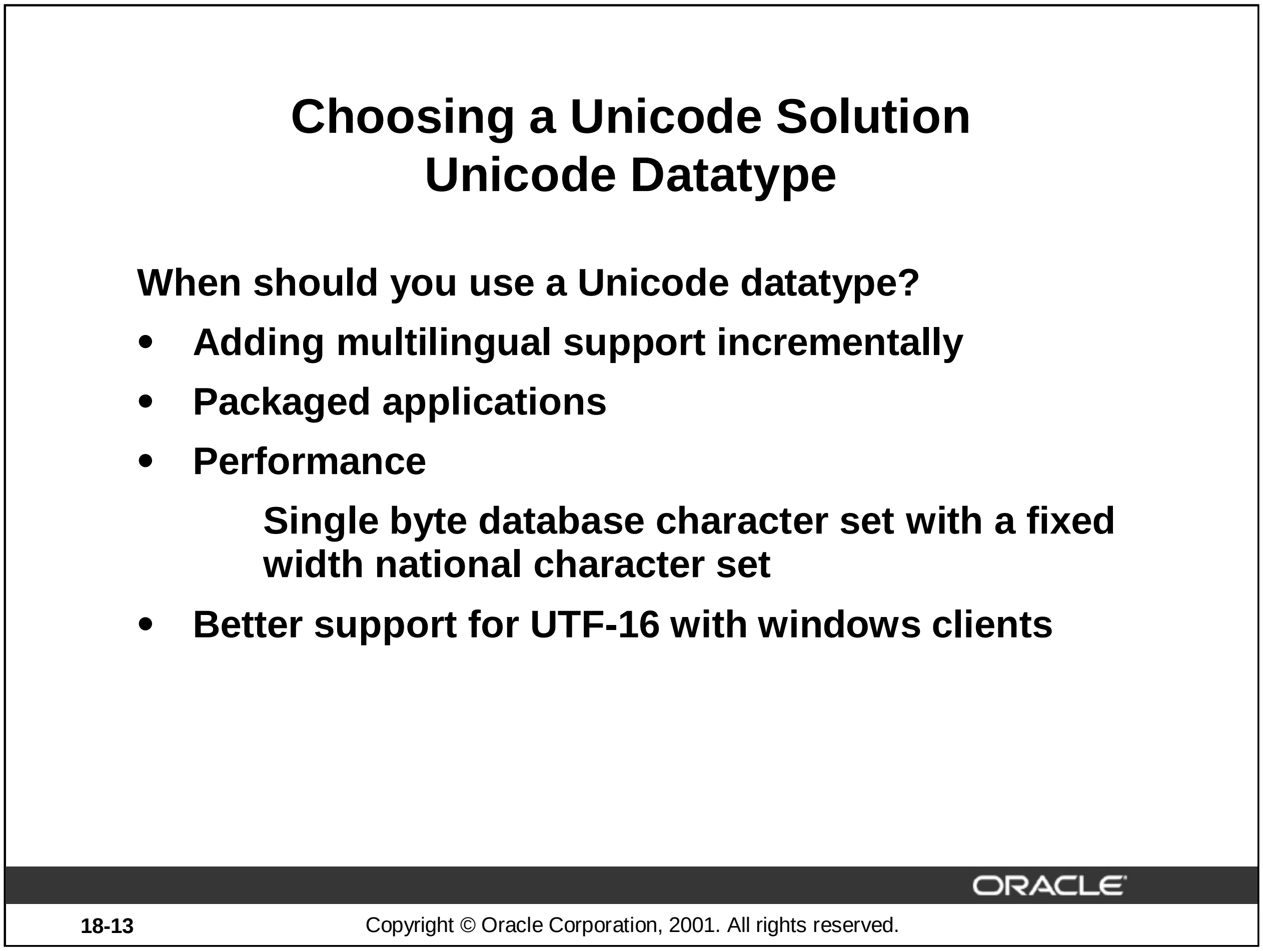
If the current database character set and data are strict `US7ASCII`, the database can be migrated with a simple `ALTER DATABASE` statement.

Evenly Distributed Multilingual data

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

Oracle Text

To use multilingual `BLOB`s with Oracle Text, a Unicode database solution is required.



Choosing a Unicode Solution Unicode Datatype

When should you use a Unicode datatype?

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

  Single byte database character set with a fixed width national character set
- **Better support for UTF-16 with windows clients**

18-13 Copyright © Oracle Corporation, 2001. All rights reserved.

Add Multilingual Support Incrementally

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

Package Application

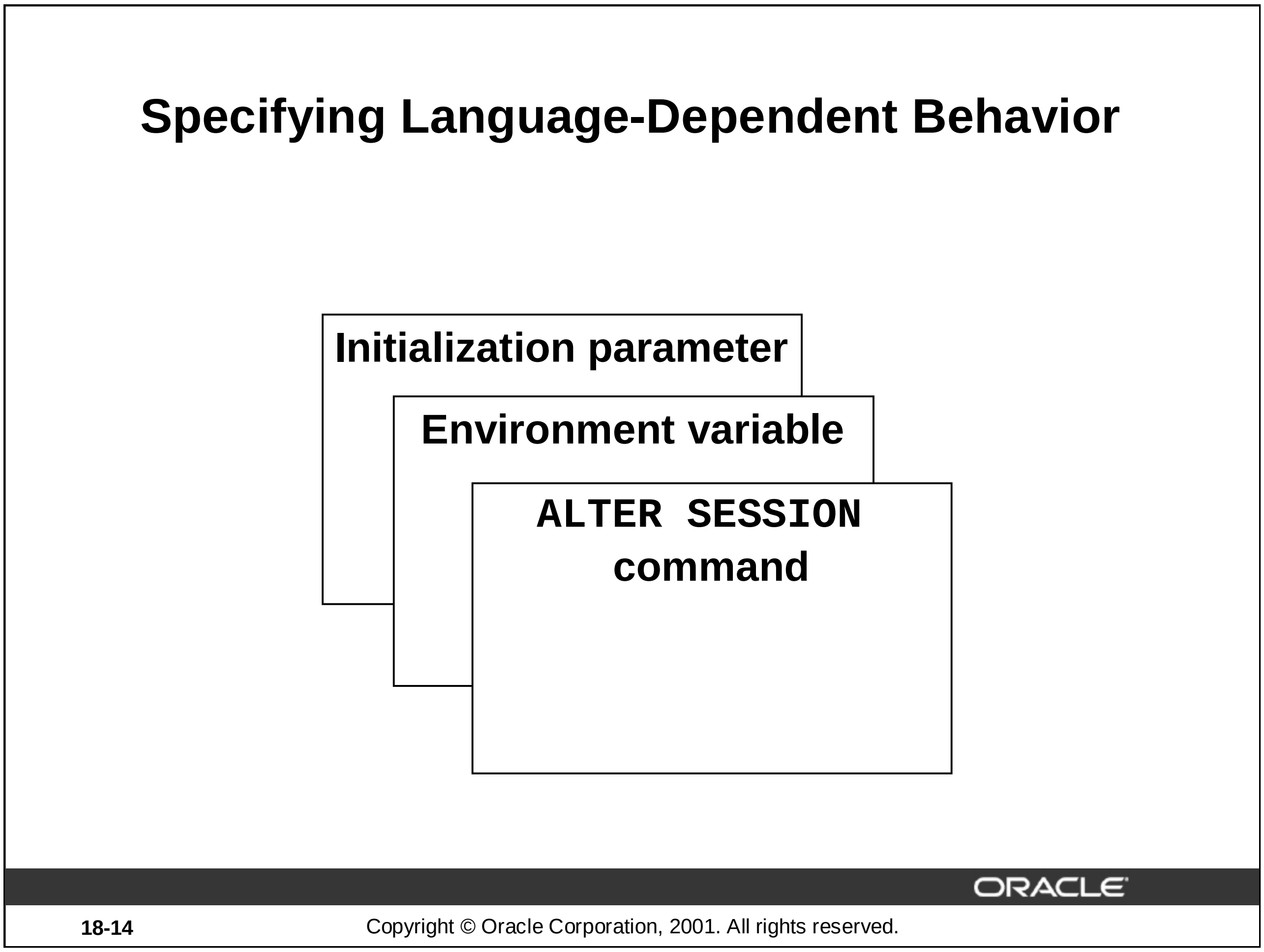
Use the SQL `NCHAR` datatype for packaged applications because it is a reliable Unicode datatype in which the data is always stored in Unicode, and the length of the data is always specified in UTF-16 code units. As a result, you need only test the application once, and your application will run on customer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

Better support for UTF-16 with windows clients

If your applications are written in Visual C/C++ or Visual Basic running on Windows, you may want to use the SQL `NCHAR` datatypes because you can store UTF-16 data in these datatypes in the same way 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 datatypes match the length of the SQL `NCHAR` datatypes in the database.



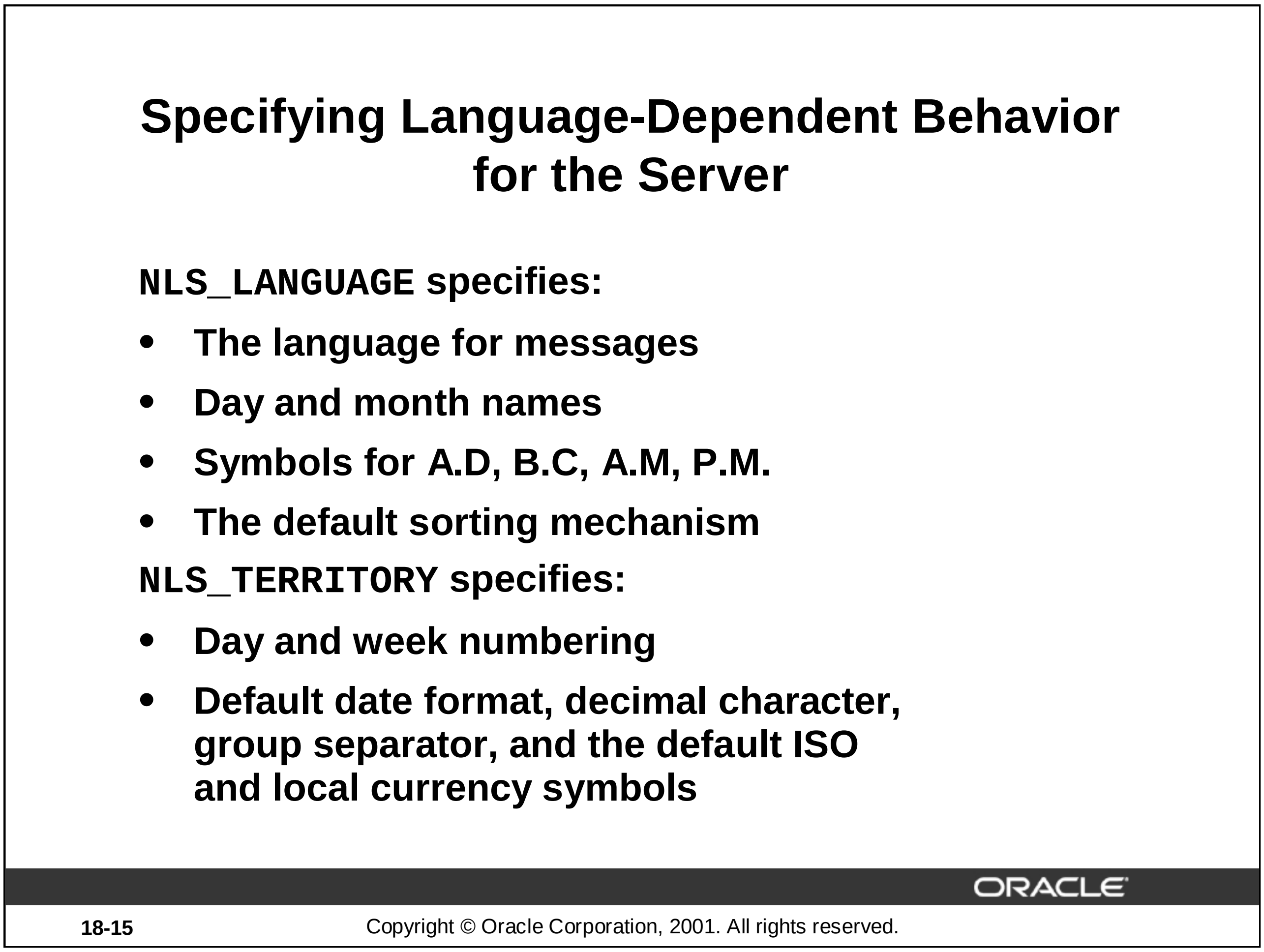
Specifying Language-Dependent Behavior

18-14

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 session 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

18-15

NLS Initialization Parameters

The initialization parameter `NLS_LANGUAGE` 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 initialization parameter `NLS_TERRITORY` 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	**AMERICAN**
NLS_DATE_LANGUAGE	AMERICAN
NLS_SORT	BINARY
NLS_TERRITORY	**AMERICA**
NLS_CURRENCY	$
NLS_ISO_CURRENCY	AMERICA
NLS_DATE_FORMAT	DD-MON-RR
NLS_NUMERIC_CHARACTERS	,.

ORACLE

NLS Initialization Parameters

The initialization parameter `NLS_LANGUAGE` 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.)

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

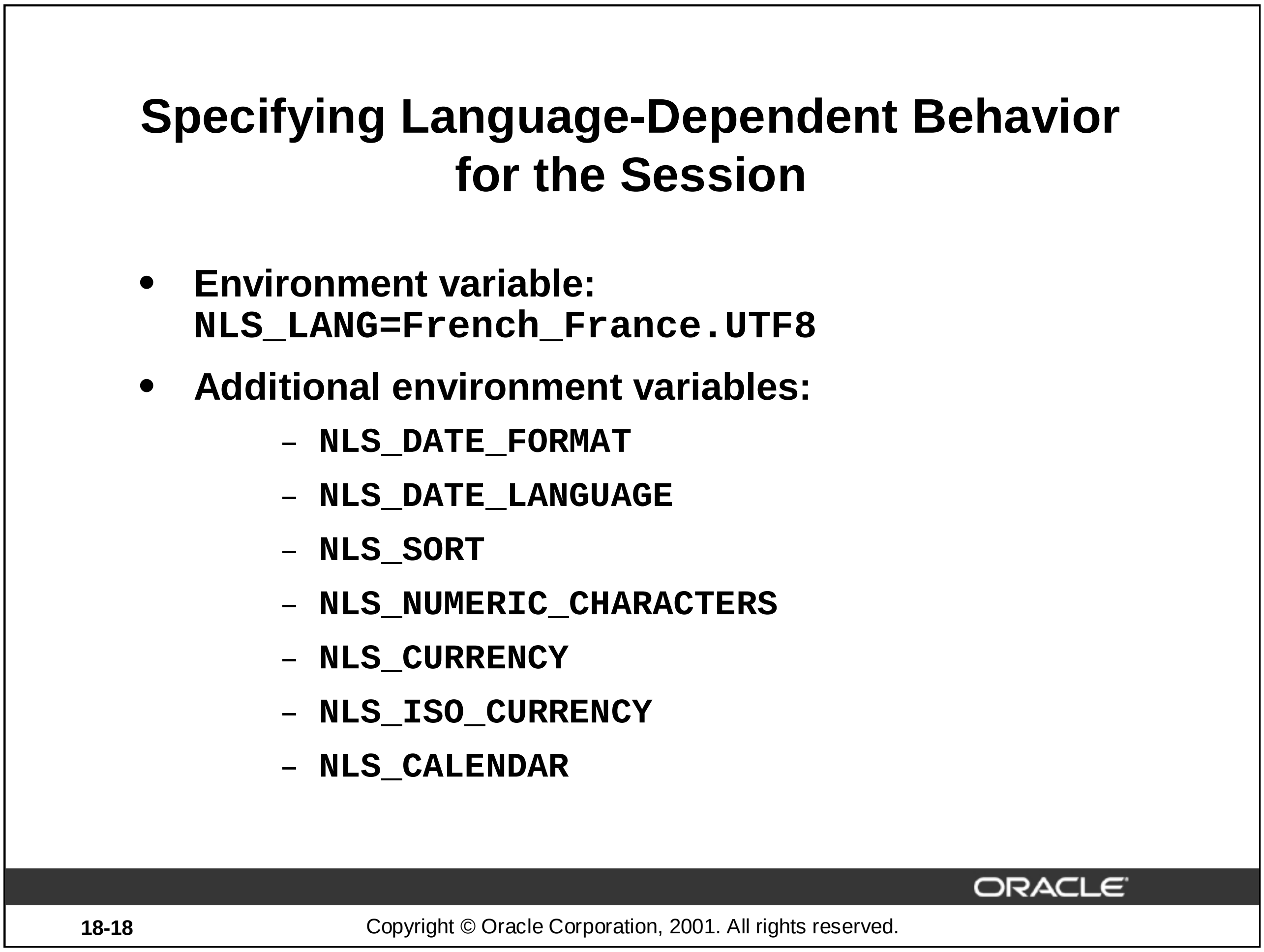
Dual Currency Support for the Euro

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

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

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

The ISO character sets, such as WE8ISO8859P15 and MS Code Page WE8MSWIN1525, 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`**

18-18

The Environment Variable `NLS_LANG`

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

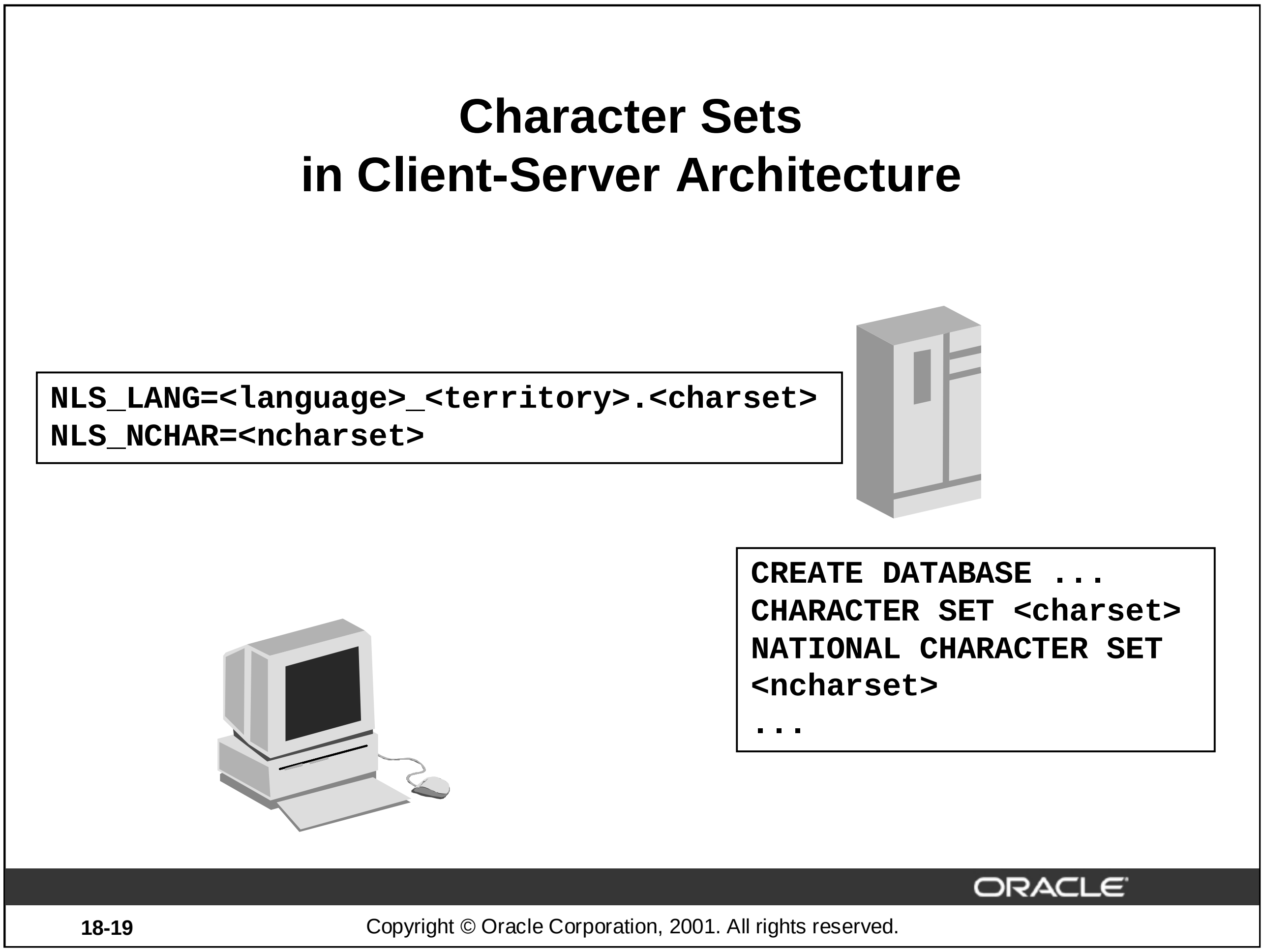
Each component controls a subset of NLS features:

```
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`

`characterset` specifies the character encoding scheme used by client application (normally that of the user’s terminal)



Character Sets in Client-Server Architecture

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

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

18-19

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 client and 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. The benefit in this scenario is better performance, but if misused it can lead to possible data inconsistency problems, such as storing data from another character set different then 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, hence all SQL string manipulation functions such as `SUBSTR` or `LENGTH` will be based on bytes rather than characters. And all your non-ASCII characters could be lost following an export and import into another database.

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`

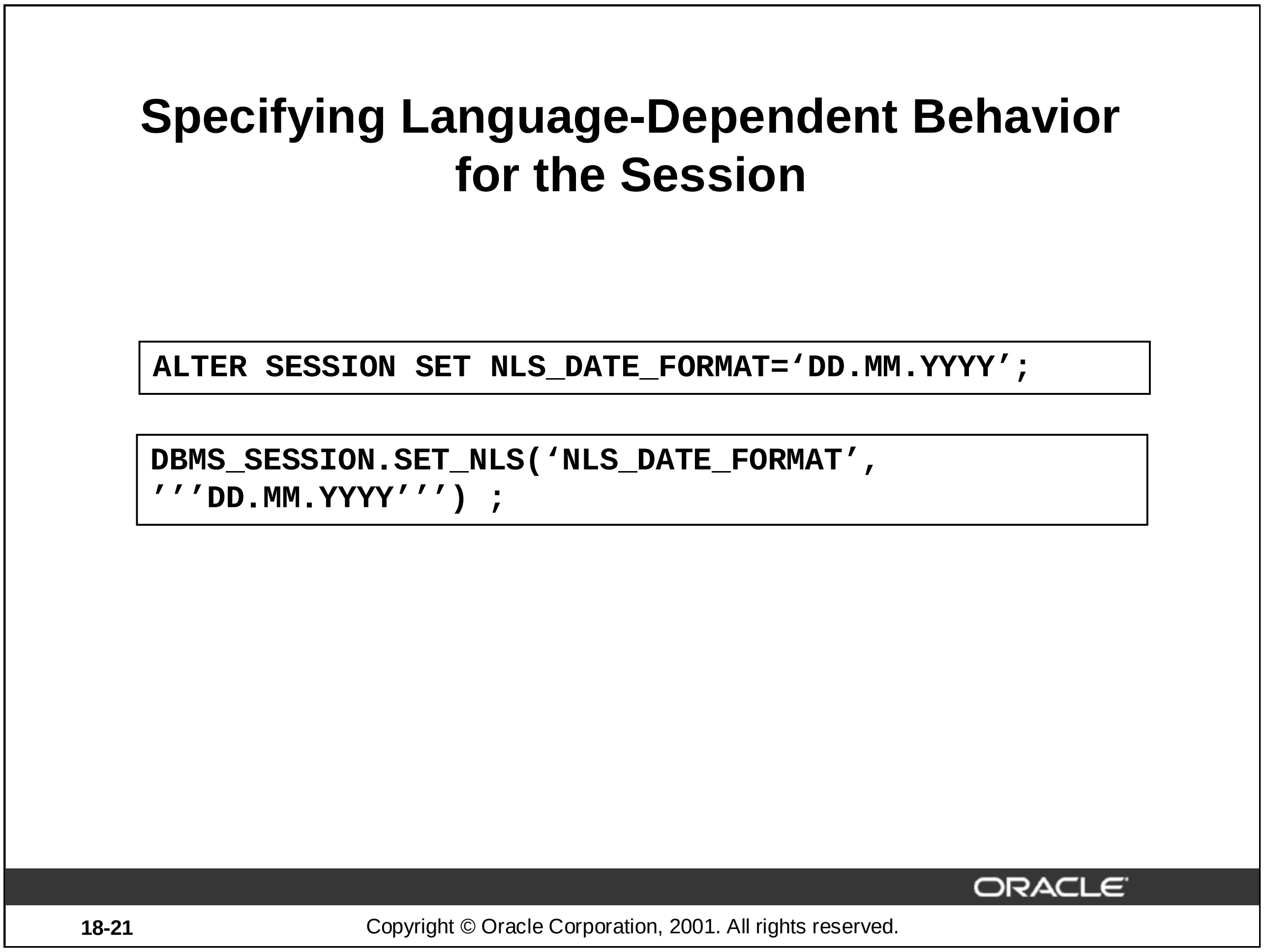
Note

The 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 default character set `US7ASCII`. `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’’’) ;
```

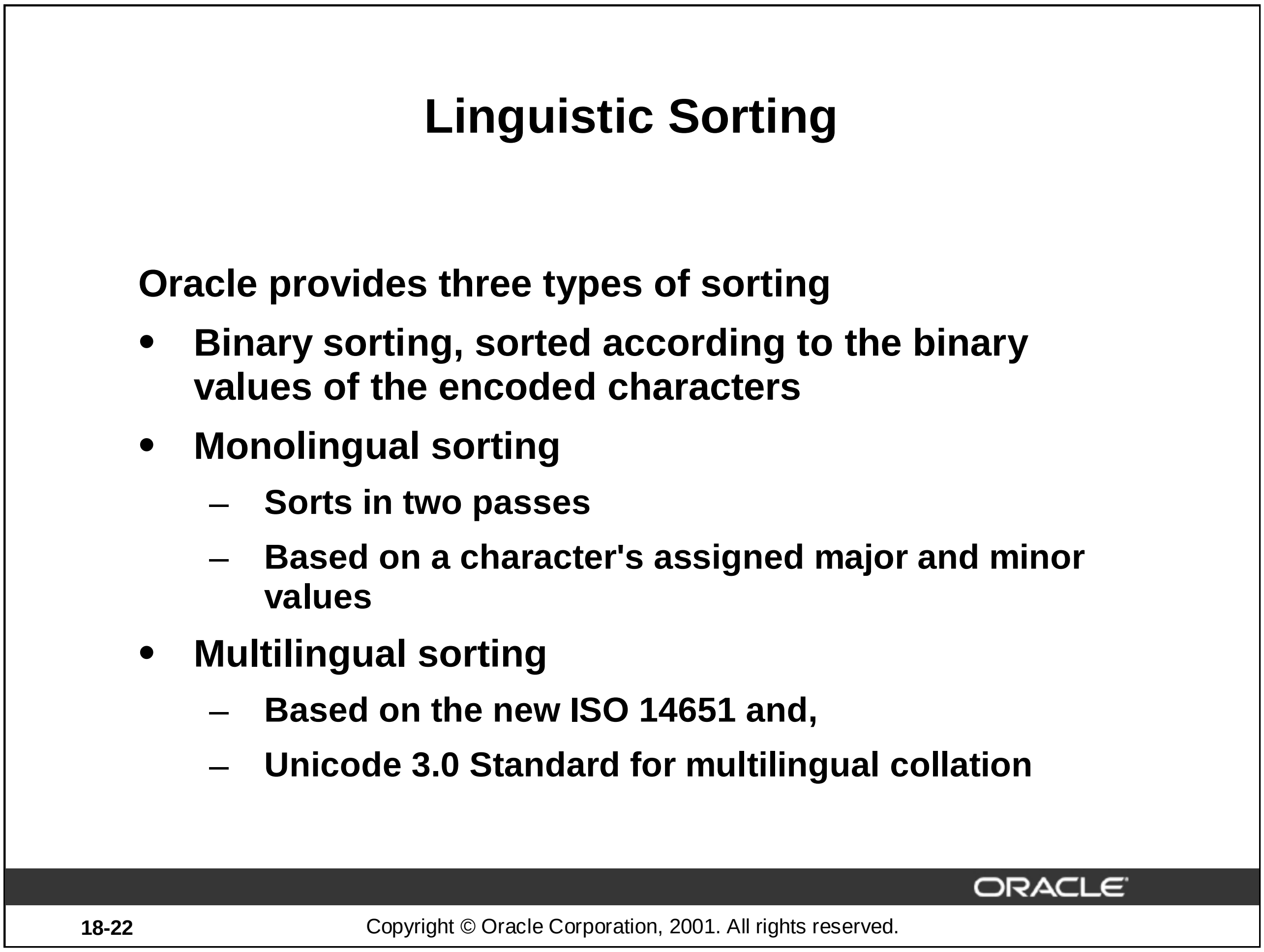
Changing NLS Parameters

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 reads the environment variables and issue the corresponding `ALTER SESSION` command.

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



Linguistic Sorting

Oracle provides three types of sorting

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

18-22

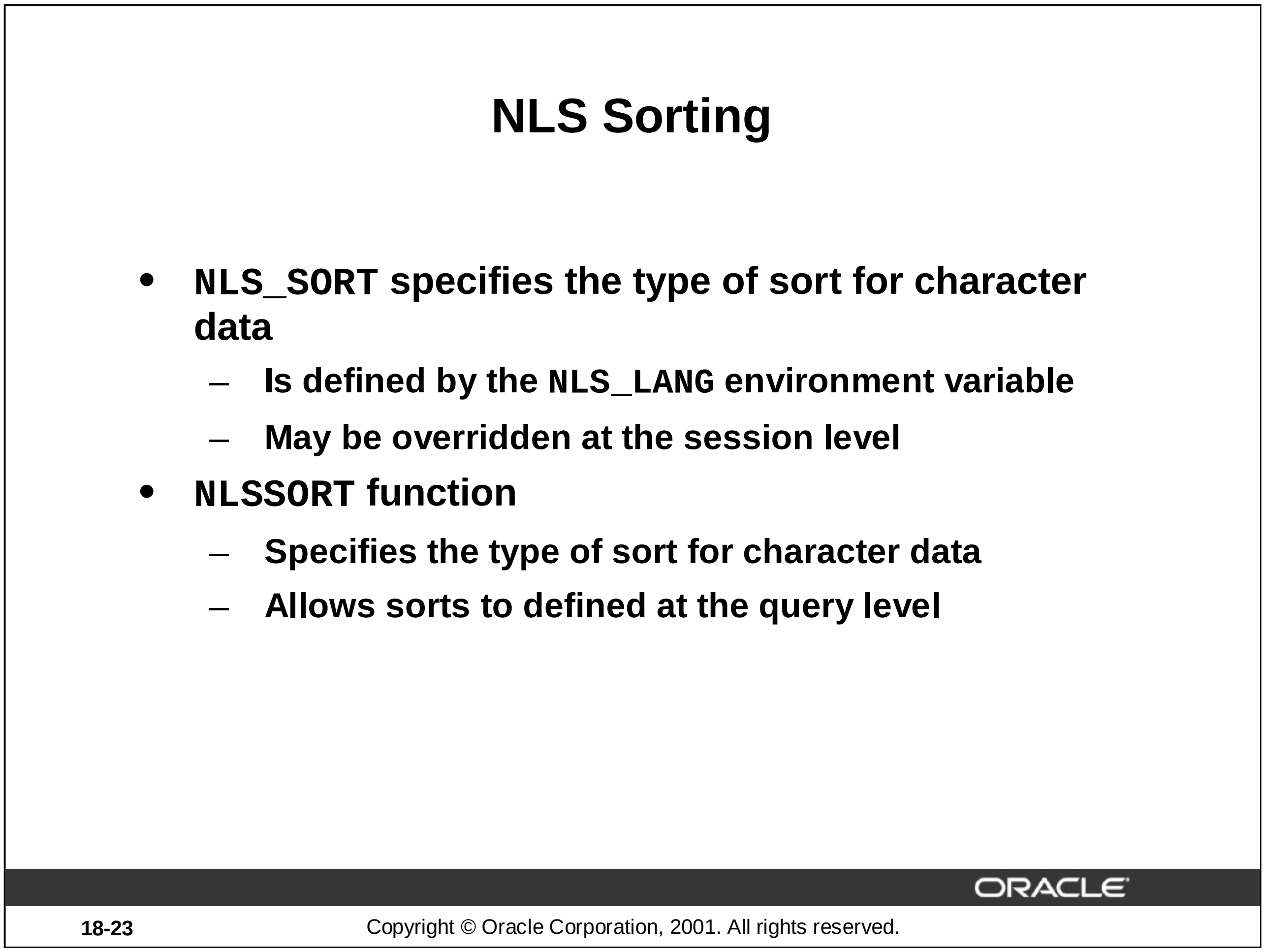
Linguistic Sorting

A binary sort is a conventional sorting mechanism by which letters are sorted according to the binary values 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. This provides better sorting than binary but is still limited.

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

Oracle currently support 84 linguistic sorts (68 monolingual and 13 multilingual). For a complete list please 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**
  - **May be overridden at the session level**
- `NLSSORT` **function**
  - **Specifies the type of sort for character data**
  - **Allows sorts to defined at the query level**

18-23

How NLS Affects Sorts

For example *ä* is sorted before *b* in French, but *ä* occurs after *z* if you use a binary sort.

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

In the following example demonstrates all three type 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.
```

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 since 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. Since monolingual sorting is limited to a two pass sort it can not 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 will 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
```

How NLS Affects Sorts (continued)

Finally the multilingual sort, `French_M`. 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
```

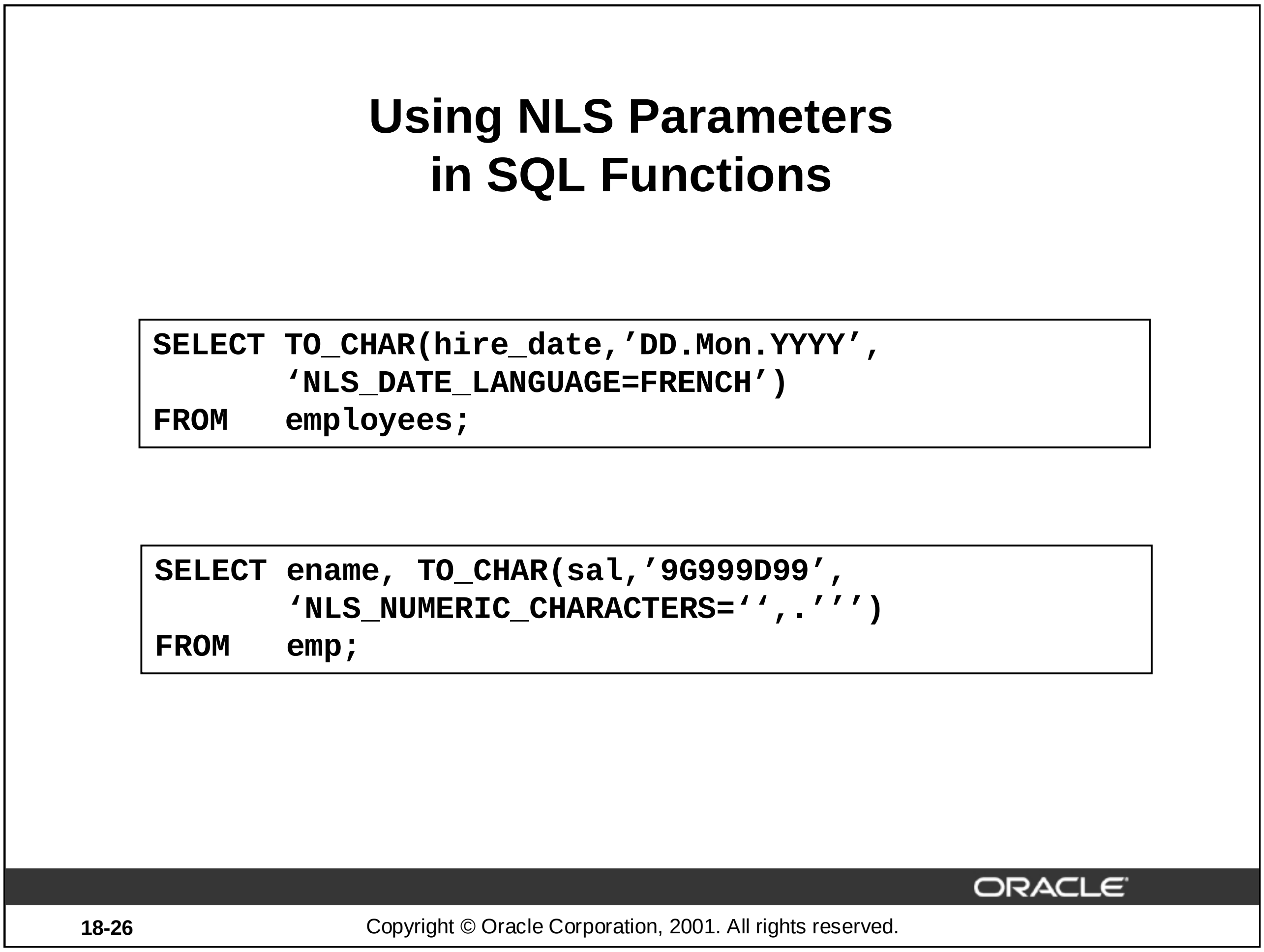
`NLSSORT` allows sorting to be defined at the query level. The following example sets `NLS_SORT` to `BINARY` at the session level but changes the sort at the query level. Notice the results are the same as the above example.

```
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;
```

SQL Functions with NLS Parameters

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.

Examples Using NLS Parameters in SQL Functions

```
 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
```

Examples Using NLS Parameters in SQL Functions (continued)

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

Examples Using NLS Parameters in SQL Functions (continued)

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

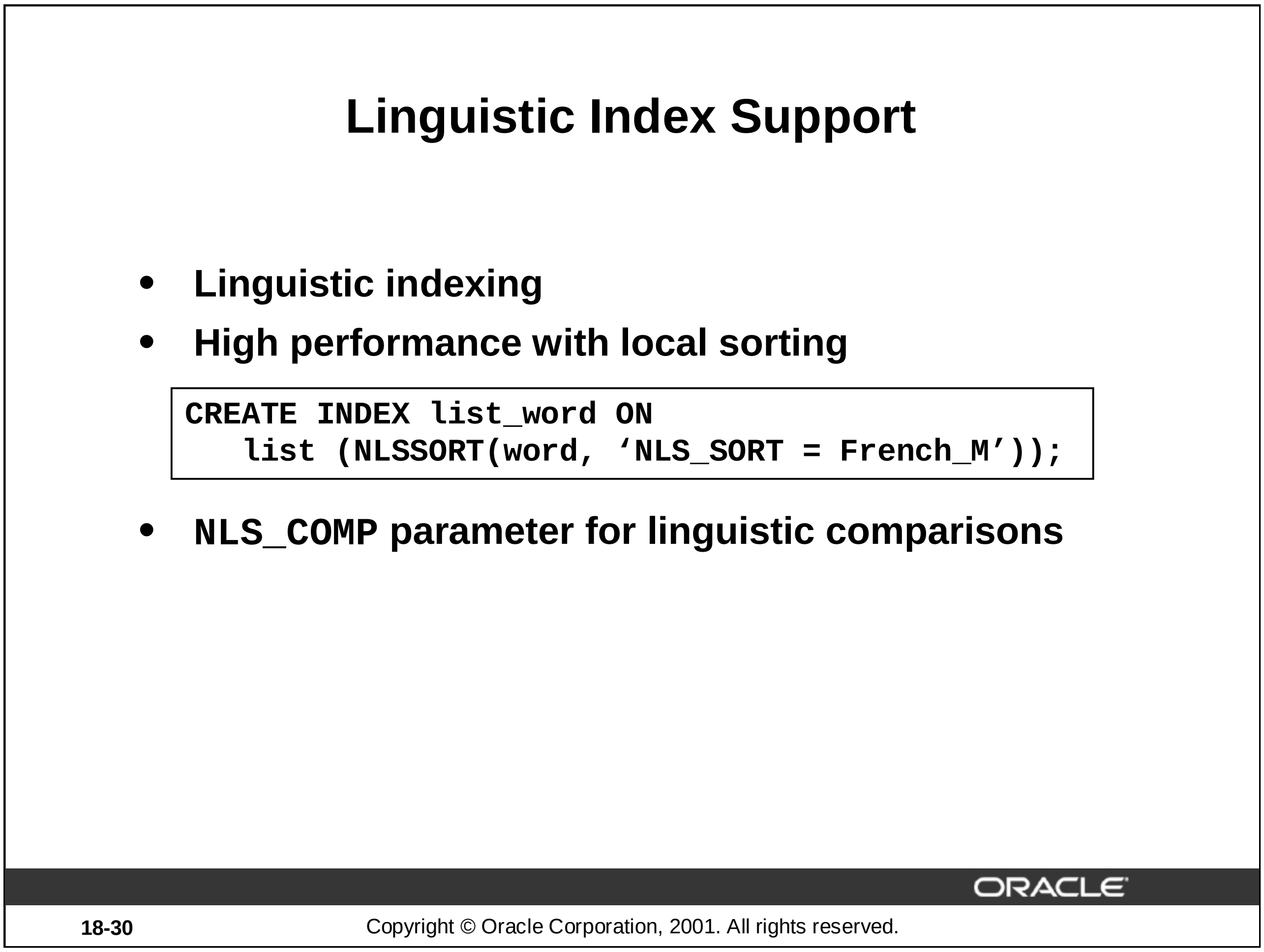
Several format mask elements have been defined for functions such as `TO_CHAR`, `TO_DATE`, and `TO_NUMBER`.

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**

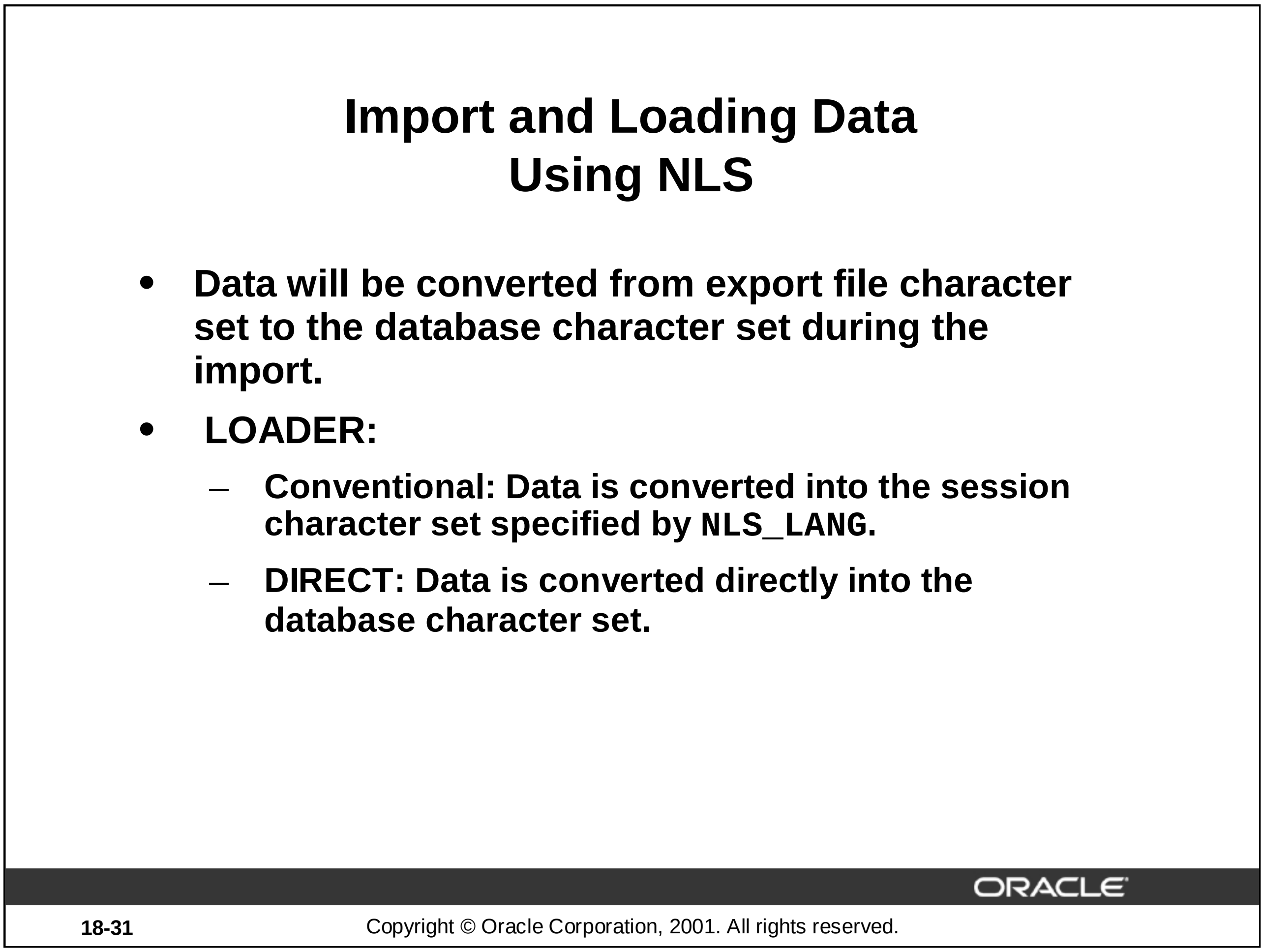
18-30

Linguistic Indexing

Functional indexes, described in the lesson Indexes and Index-Organized Tables, 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 French_M sorting order. This allows you to perform index-based queries on data sorted according to each languages rules.

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

NLS with Import and SQL*Loader

The export file is exported with the source database character set. During 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.

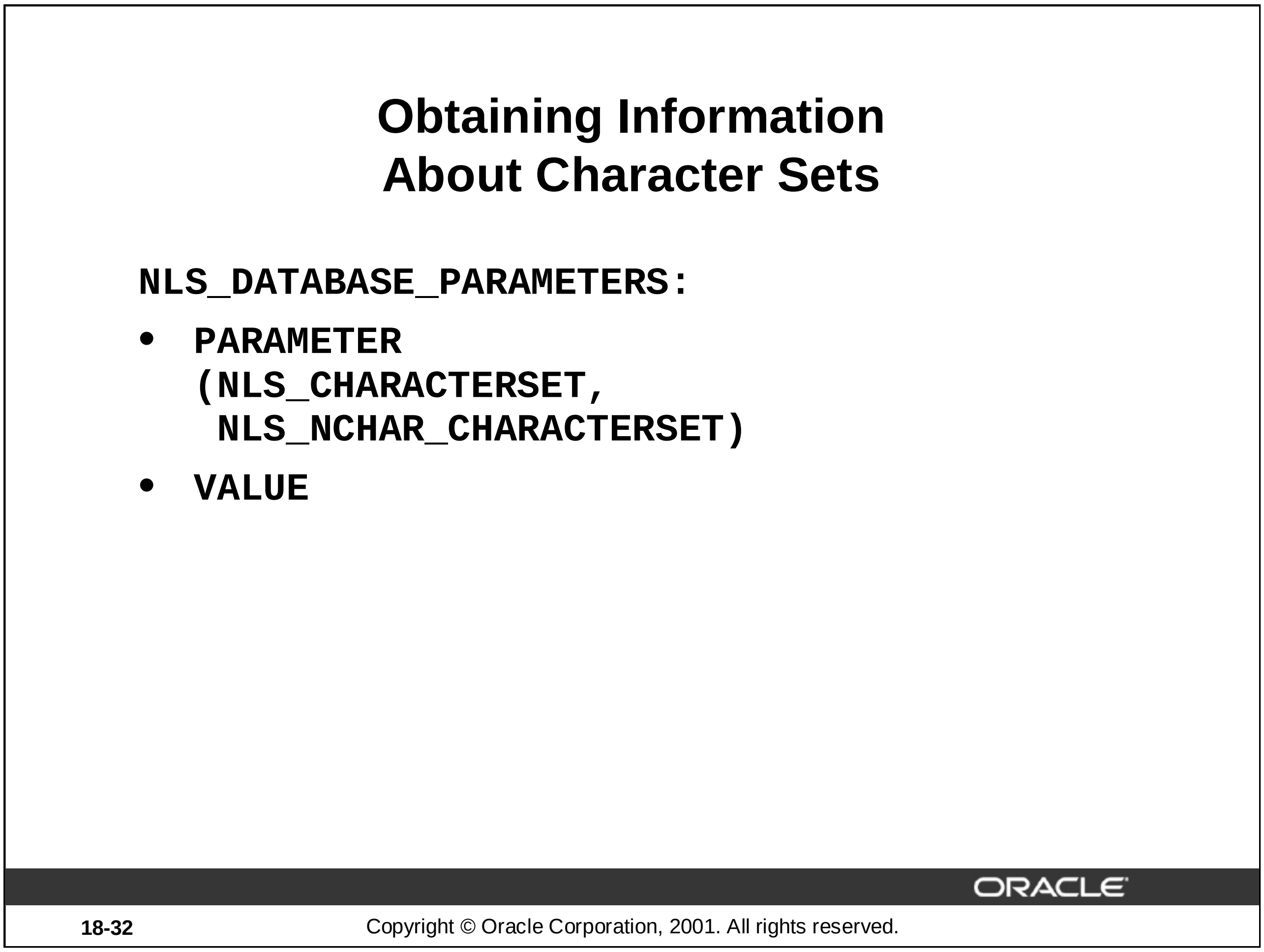
On the direct path, data is converted directly into the database character set.

The control file of the SQL*Loader shows 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
```



Obtaining Information About Character Sets

NLS_DATABASE_PARAMETERS:

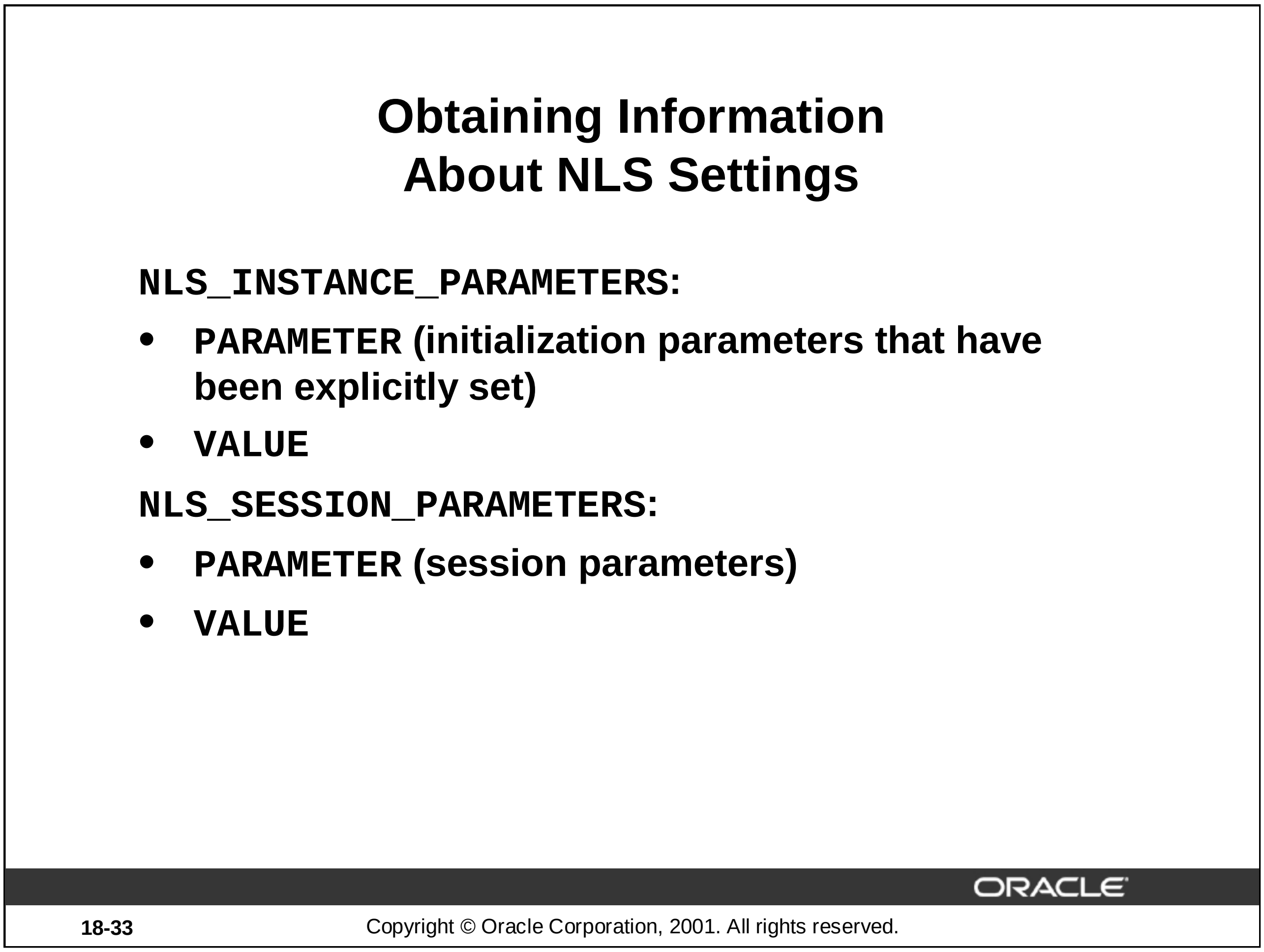
- **PARAMETER (NLS_CHARACTERSET, NLS_NCHAR_CHARACTERSET)**
- **VALUE**

18-32

Querying the Data Dictionary for NLS 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 Information About NLS Settings

`NLS_INSTANCE_PARAMETERS`:

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

`NLS_SESSION_PARAMETERS`:

- `PARAMETER` (session parameters)
- `VALUE`

18-33

Querying the Data Dictionary for NLS 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
```

Querying the Data Dictionary for NLS Information (continued)

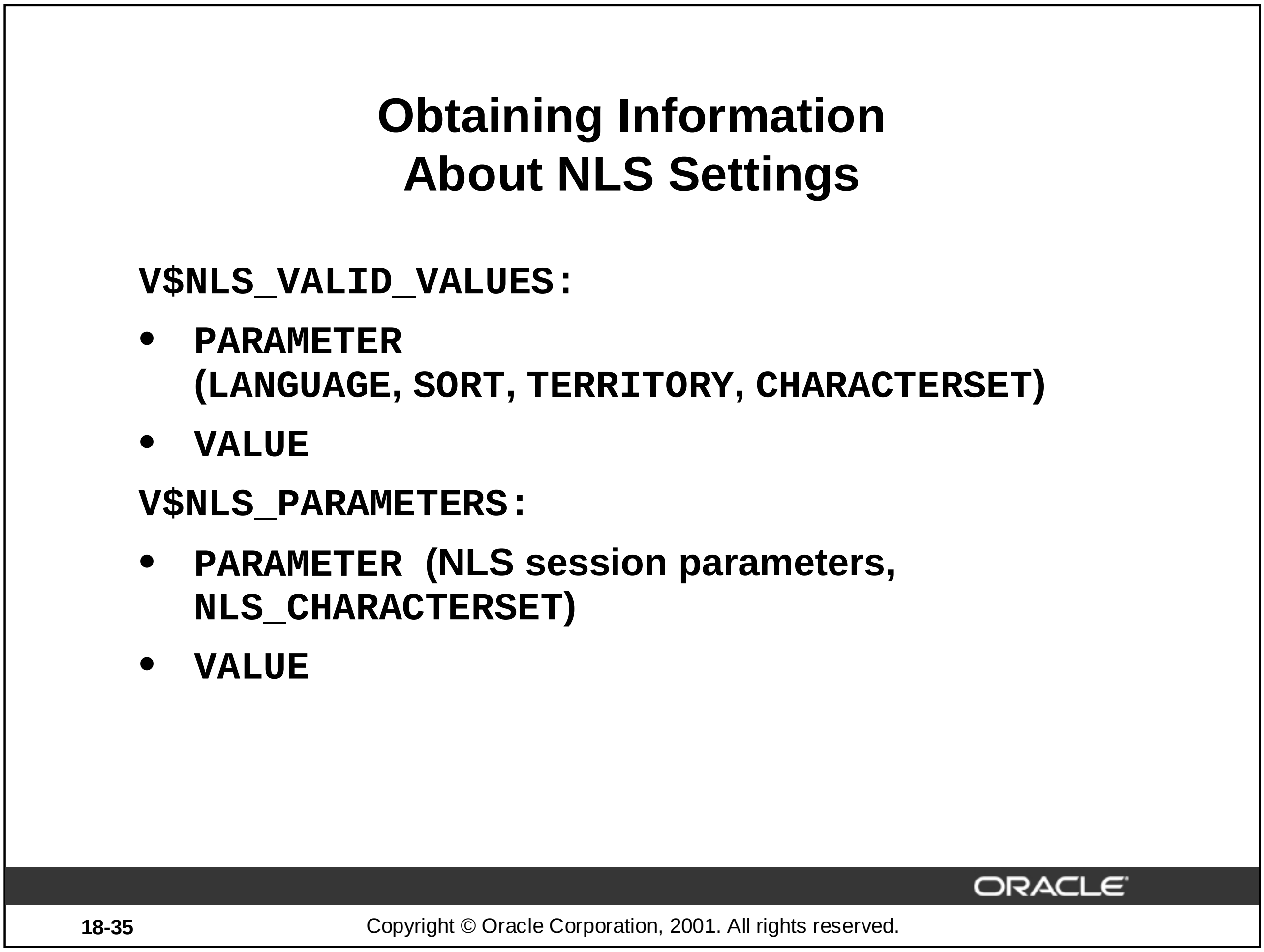
```
NLS_TIME_TZ_FORMAT
NLS_TIMESTAMP_TZ_FORMAT
NLS_DUAL_CURRENCY
NLS_COMP
NLS_LENGTH_SEMANTICS           BYTE
NLS_NCHAR_CONV_EXCP            FALSE

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 Information About NLS Settings

V$NLS_VALID_VALUES:

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

V$NLS_PARAMETERS:

- **PARAMETER (NLS session parameters, NLS_CHARACTERSET)**
- **VALUE**

Querying the Data Dictionary for NLS 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
  ...
```

Querying the Data Dictionary for NLS Information (continued)

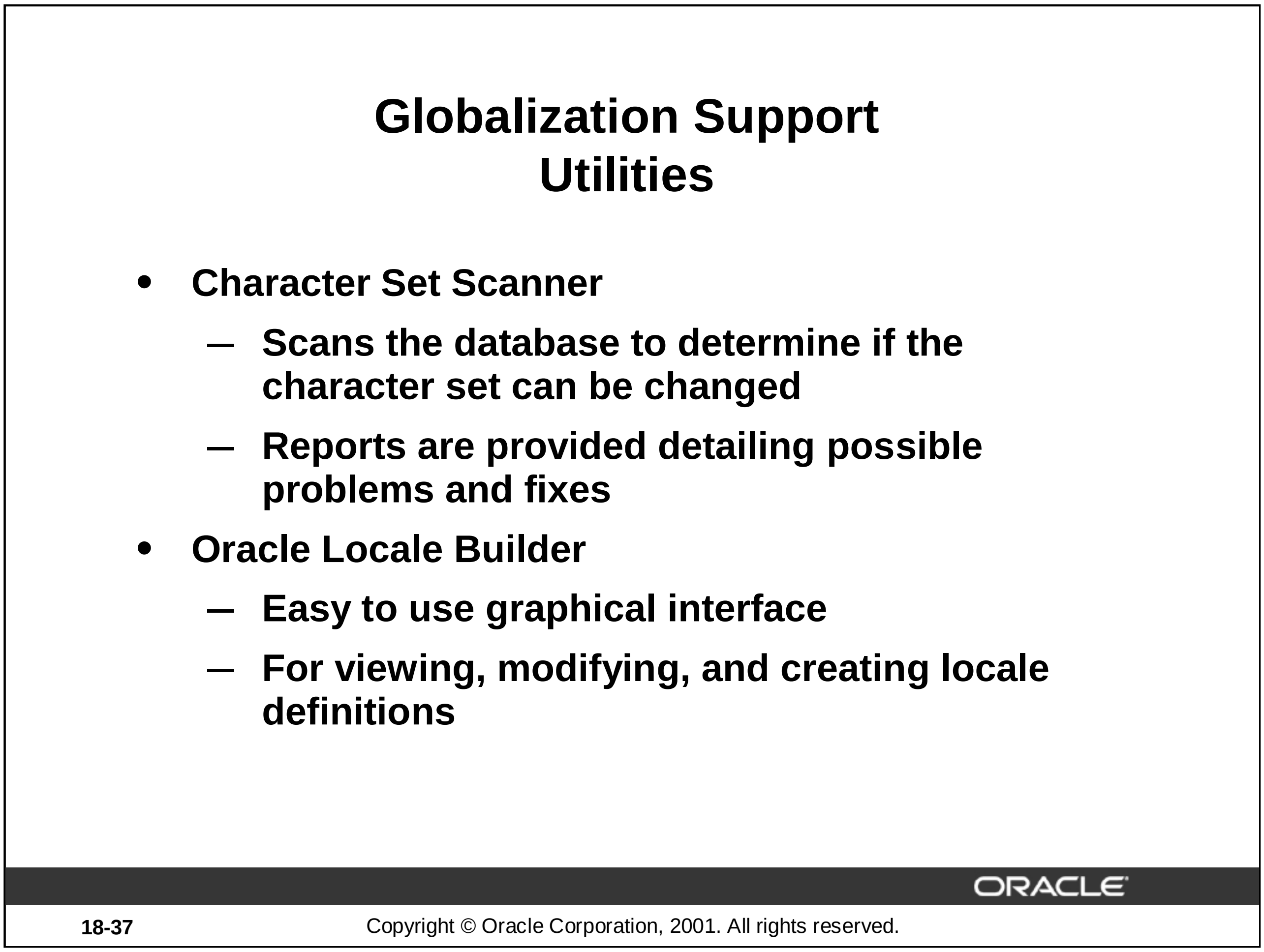
Display current values of NLS parameters.

```
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$`.



Globalization Support Utilities

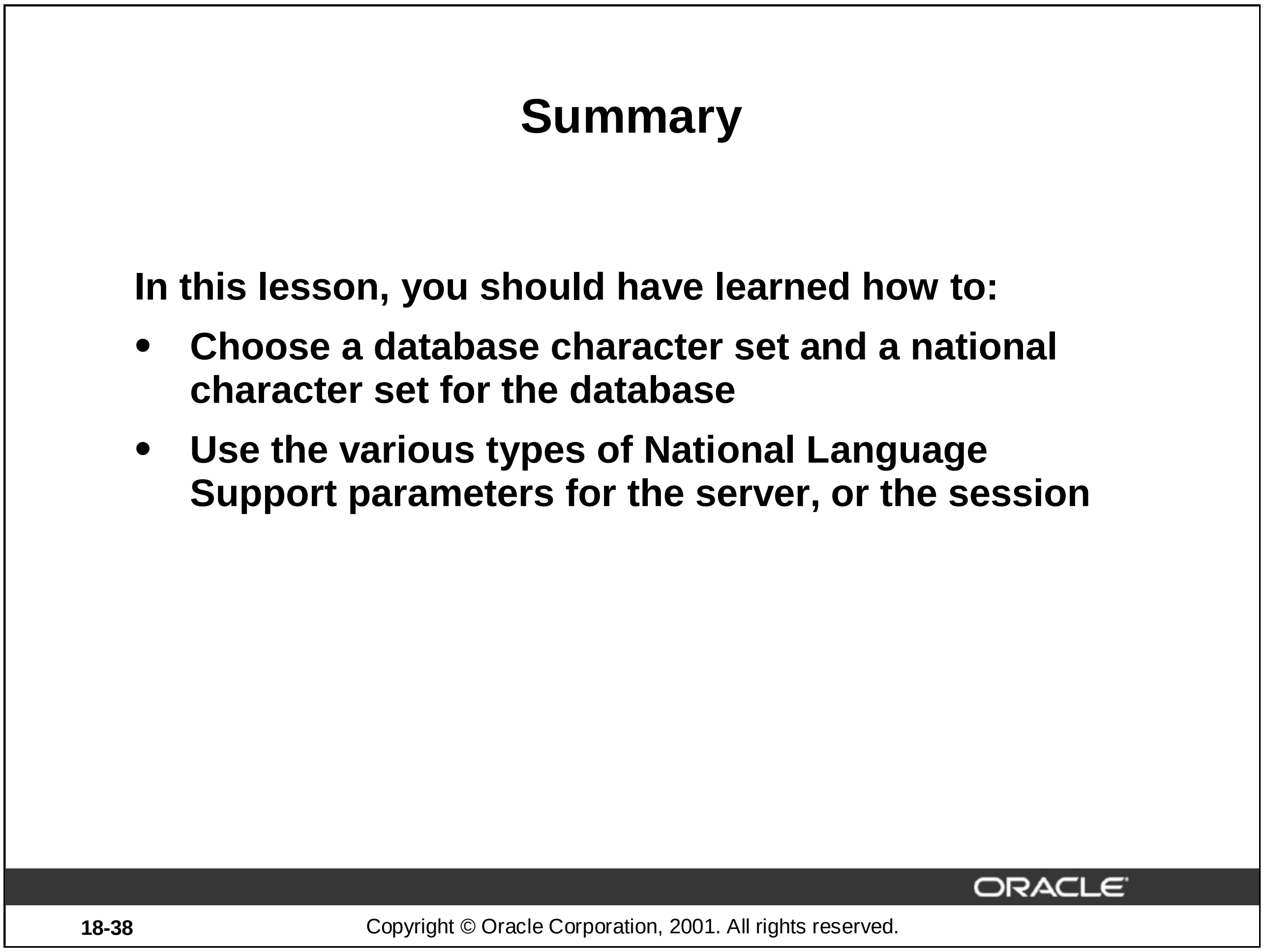
- **Character Set Scanner**
  - **Scans the database to determine if the character set can be changed**
  - **Reports are provided detailing possible problems and fixes**
- **Oracle Locale Builder**
  - **Easy to use graphical interface**
  - **For viewing, modifying, and creating locale definitions**

Character Set Scanner

The character set scanner is a command line utility which assists in character set conversion. The scanner identifies area 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. Prior to any character set conversion the character set scanner should be used.

Oracle Locale Builder

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



Summary

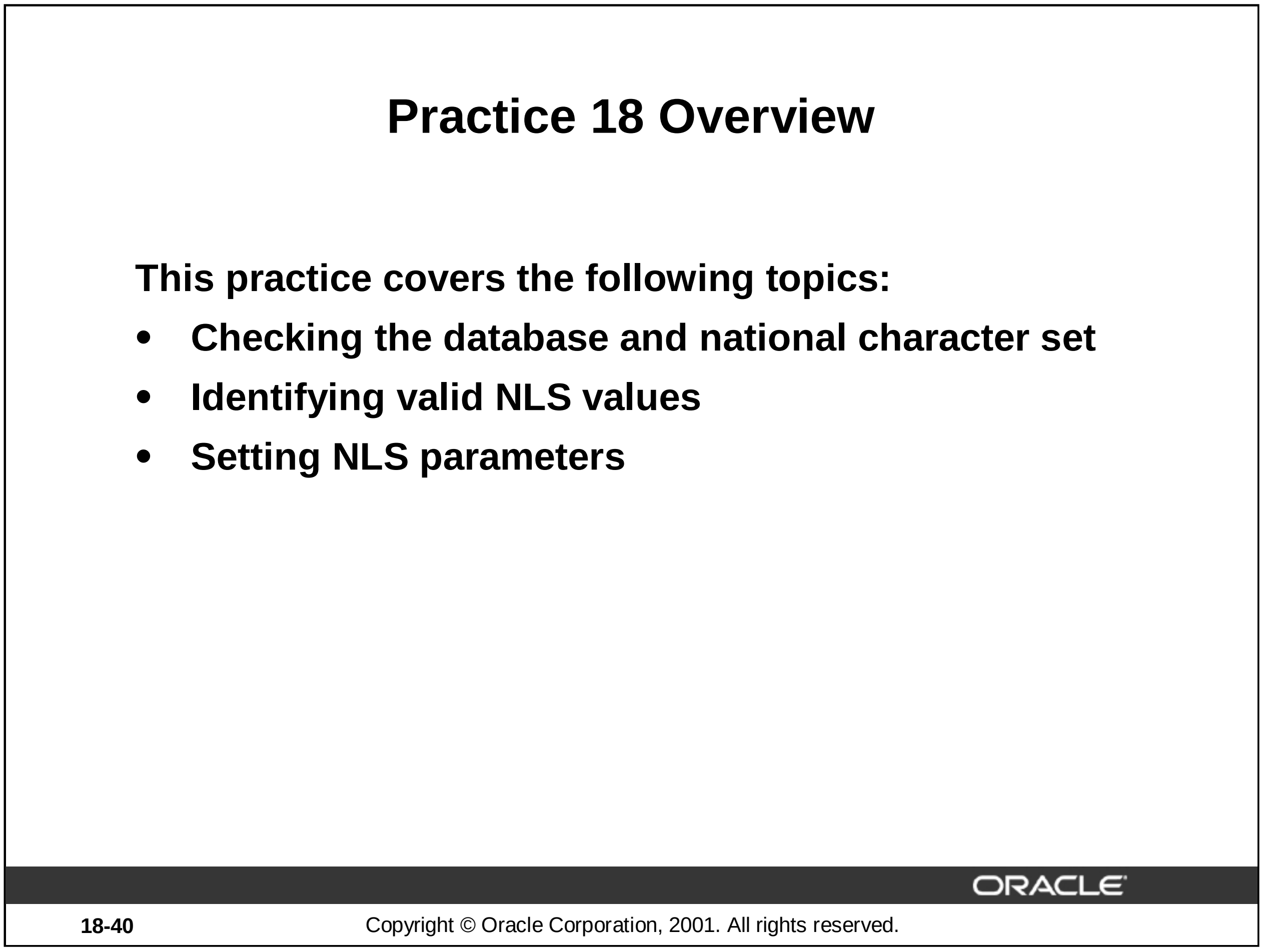
In this lesson, you should have learned how to:

- 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

18-38

Quick Reference

Context	Reference
Initialization parameters	NLS_LANGUAGE NLS_TERRITORY NLS_DATE_FORMAT NLS_DATE_LANGUAGE NLS_CURRENCY NLS_ISO_CURRENCY NLS_SORT NLS_NUMERIC_CHARACTERS NLS_CALENDAR
Dynamic performance views	V$NLS_VALID_VALUES V$NLS_PARAMETERS
Data dictionary views	NLS_DATABASE_PARAMETERS NLS_INSTANCE_PARAMETERS NLS_SESSION_PARAMETERS
Commands	ALTER SESSION SET
Packaged procedures and functions	DBMS_SESSION.SET_NLS



Practice 18 Overview

This practice covers the following topics:

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

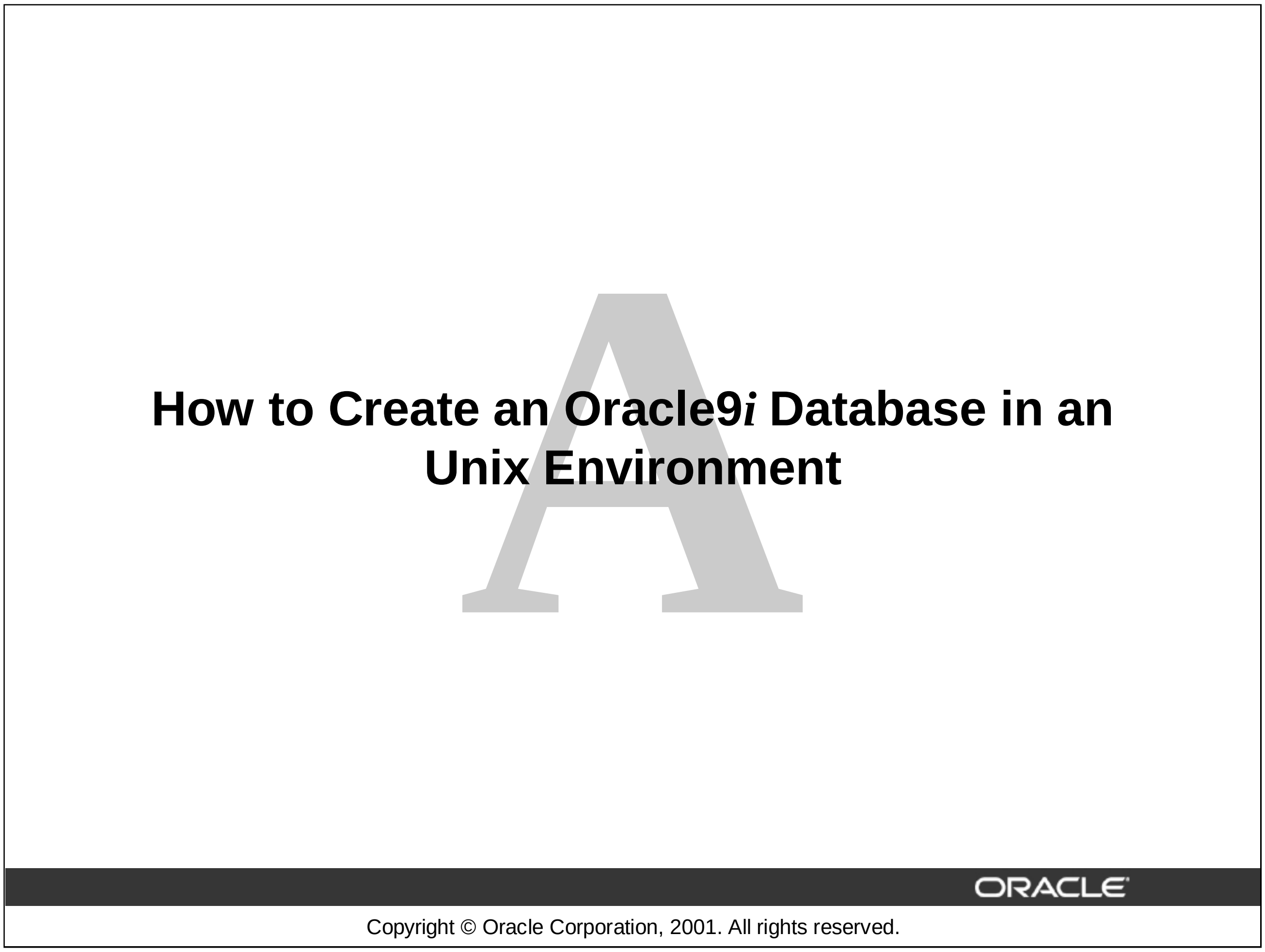
18-40

Practice 18: Using National Language Support

1 Check the database and the national character set.

2 Which are valid values for the database character set.

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`.



How to Create an Oracle9*i* Database in an Unix Environment

Introduction

Steps to Create a Database

There are six steps to creating a useable database: three steps to create a 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 scripts `catalog.sql` and `catproc.sql.`
5. Run the script `pupbld.sql`.
6. Create a tablespace for user data and any other tablespaces that might be required to meet the needs of the database.

Note: This paper makes the assumption that the Oracle9*i* Server has been installed in an `ORACLE_HOME`. Installation of the Oracle9*i* Server does not fall into the scope of this paper.

Setting 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 need to 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 needs to 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 do not have to be in any order but 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.

There are a few parameters that should be configured; these include db_name, control_files, background_dump_dest, user_dump_dest, core_dump_dest, and undo_management.

The parameters background_dump_dest, user_dump_dest, and core_dump_dest 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 different purpose than ORACLE_SID. The ORACLE_SID is the name used to designate an instance of the database. Many times the db_name and ORACLE_SID are the same but it is not required. The db_name in the initsid.ora must be identical (case sensitive) to the name of the database used in the CREATE DATABASE command when the database is created.

The control_files initialization parameter designates the full path and filename of each controlfile for the database. For the creation of the database, it identifies the controlfiles that need to 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 logfiles, 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 used to connect to SQL*Plus is part of the administrator's group, defined during the installation of the Oracle9*i* server, then the following syntax may be used:

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

   or

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

2. If the Unix user being used to connect 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 will need to be used. The syntax below 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 for creating and executing 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. Startup 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 scripts catalog.sql and catproc.sql, located in $ORACLE_HOME/rdbms/admin, must be run after the database is created. The script catalog.sql 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 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 script pupbld.sql, located in directory $ORACLE_HOME/sqlplus/admin, 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 other tablespaces needed for the installation.

In a database installation the following tablespaces are normally 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

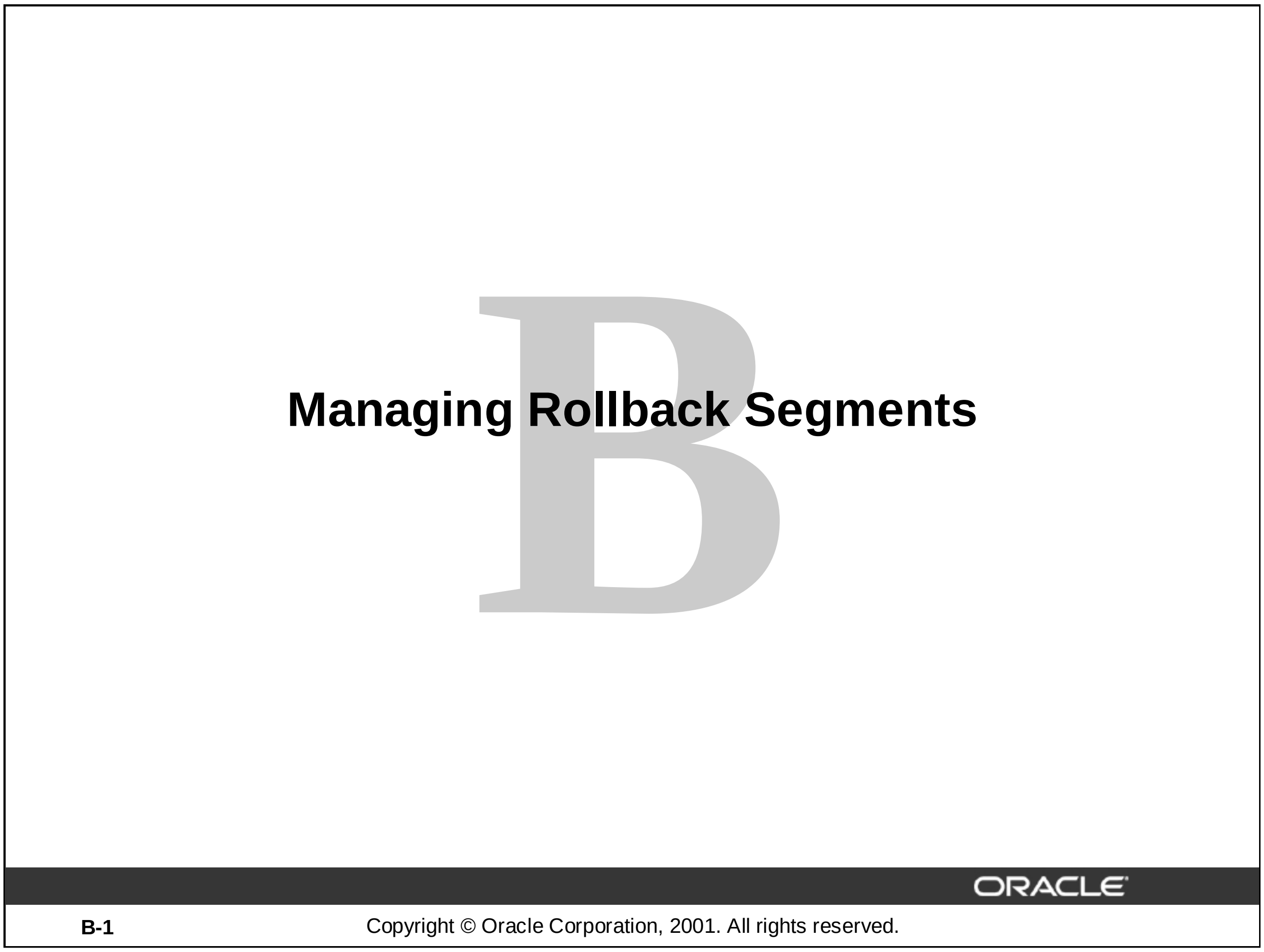
- Set `ORACLE_HOME`, `ORACLE_SID`, `PATH`, and `LD_LIBRARY_PATH`
- Edit the `initsid.ora`.
- Execute the `CREATE DATABASE` command.
- Run the scripts `catalog.sql` and `catproc.sql`.
- Run the script `pupbld.sql`.
- Create the tablespaces needed for the database.

Sample initdb01.ora:

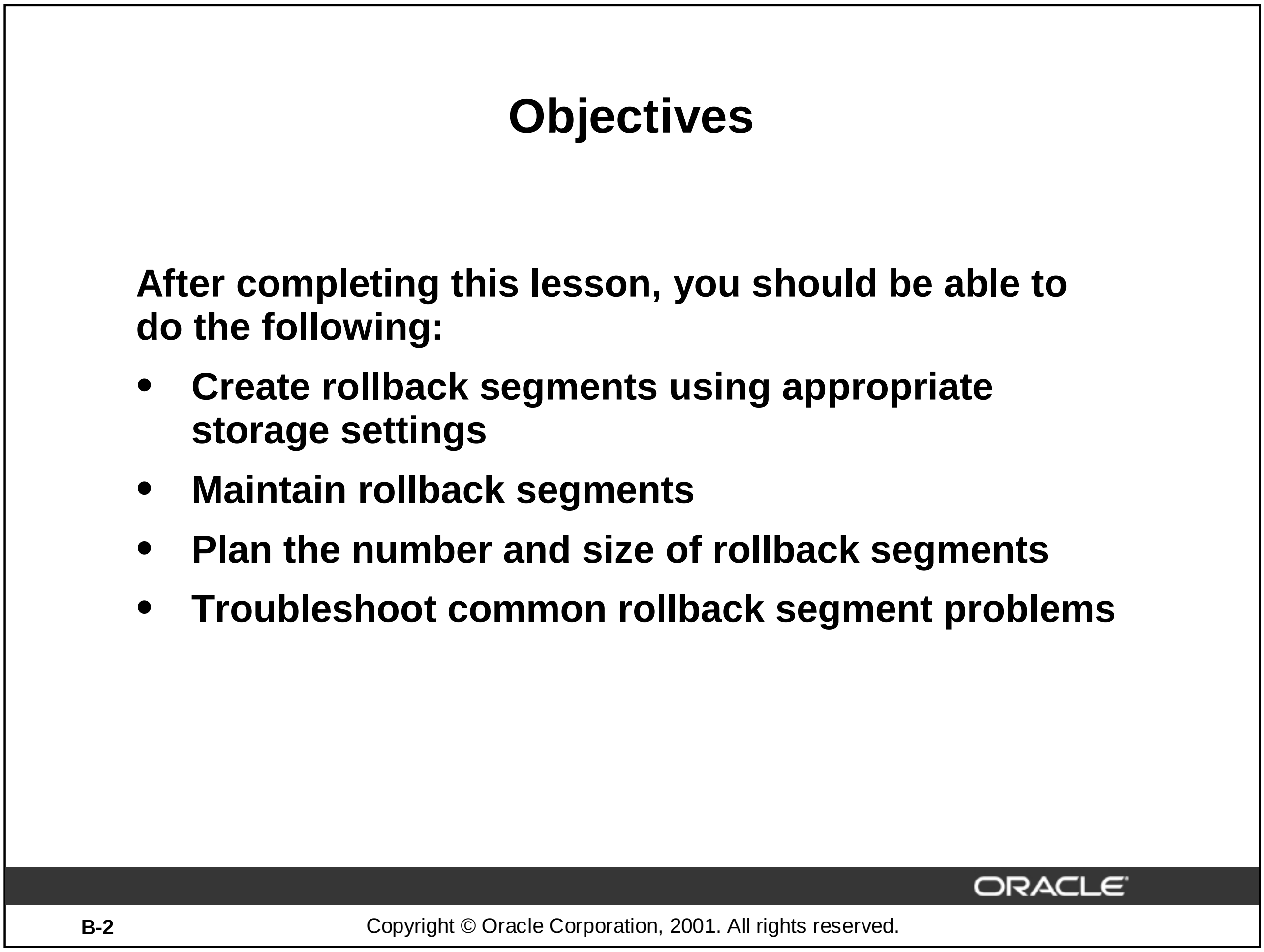
```
background_dump_dest=$HOME/ADMIN/BDUMP
compatible=9.0.0
control_files=$HOME/ORADATA/u01/ctrl_01_sid.ctl
core_dump_dest=$HOME/ADMIN/CDUMP
db_block_size=4096
db_cache_size=4M
db_domain=world
db_name=db01
global_names=TRUE
instance_name=db01
max_dump_file_size=10240
remote_login_passwordfile=exclusive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



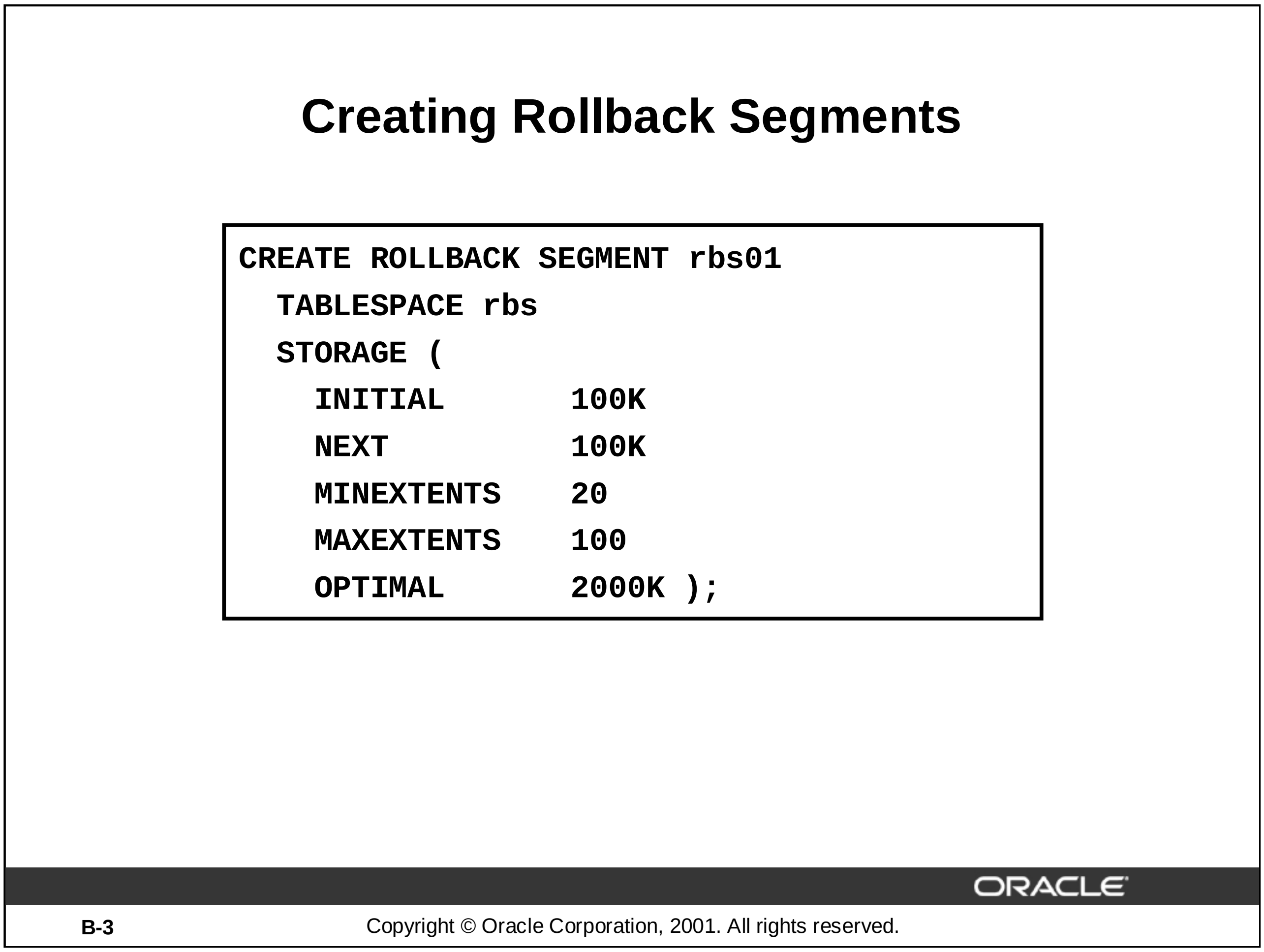
Managing Rollback Segments



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



Creating Rollback Segments

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

B-3

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)

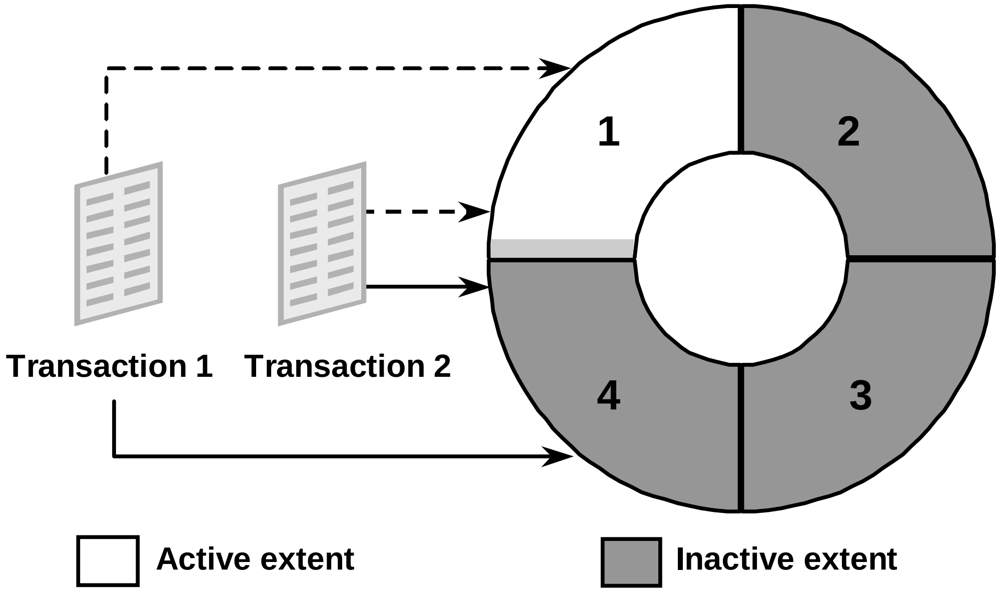
Restrictions

- 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



Allocation of a Rollback Segment

When a transaction begins, a rollback segment needs to 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.

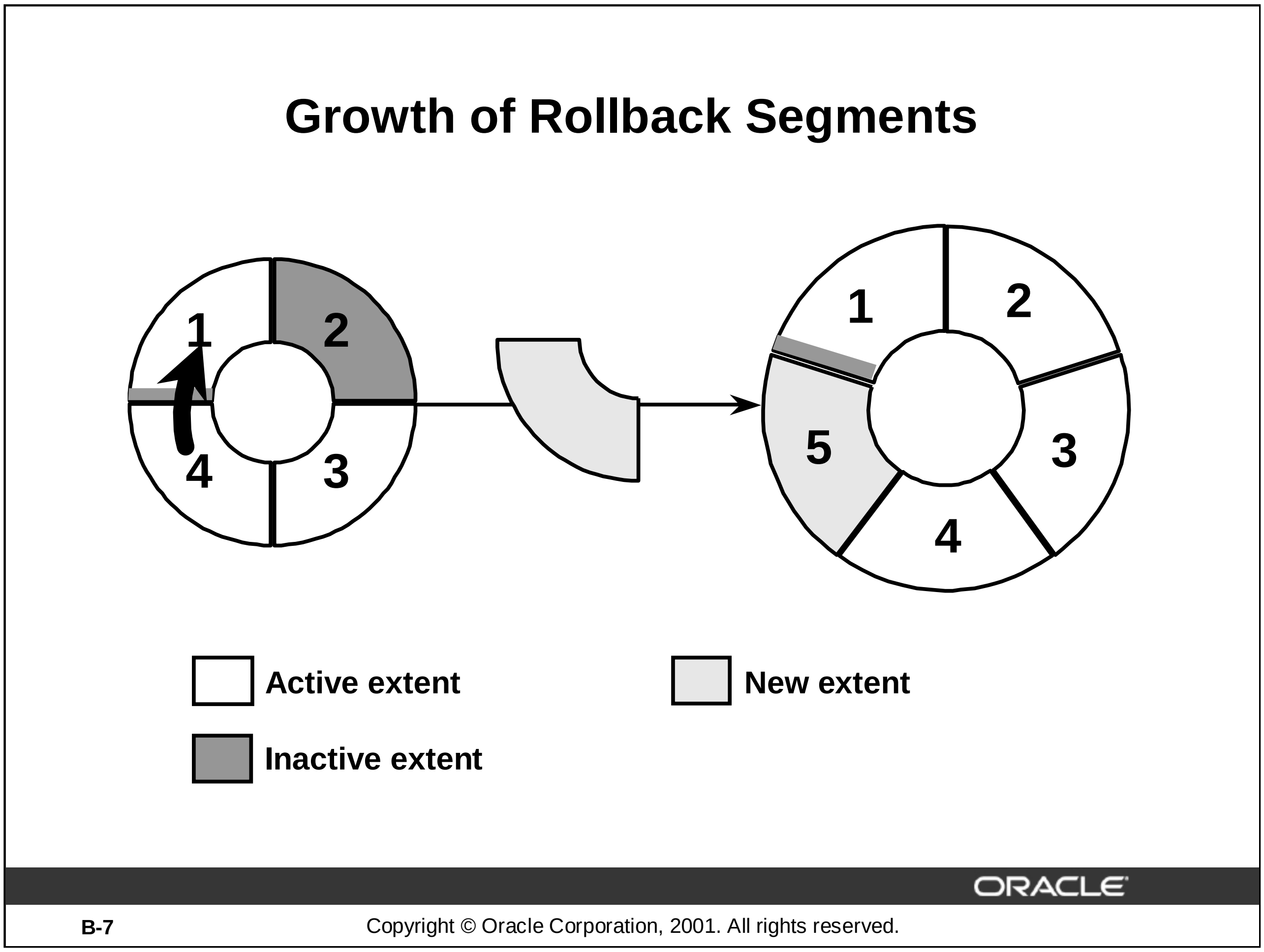
More than one transaction can write to the same extent of a rollback segment; however, each rollback segment block contains information from one and only one transaction.

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

□ Active extent

▨ Inactive extent

░ New extent

B-7

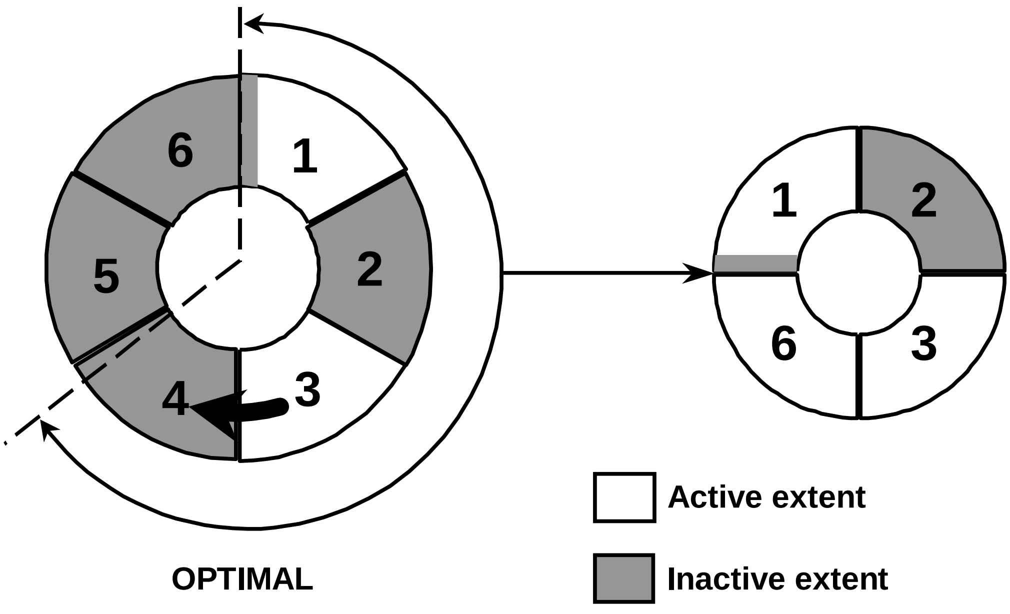
Growth of Rollback Segments

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 needs 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



OPTIMAL Parameter

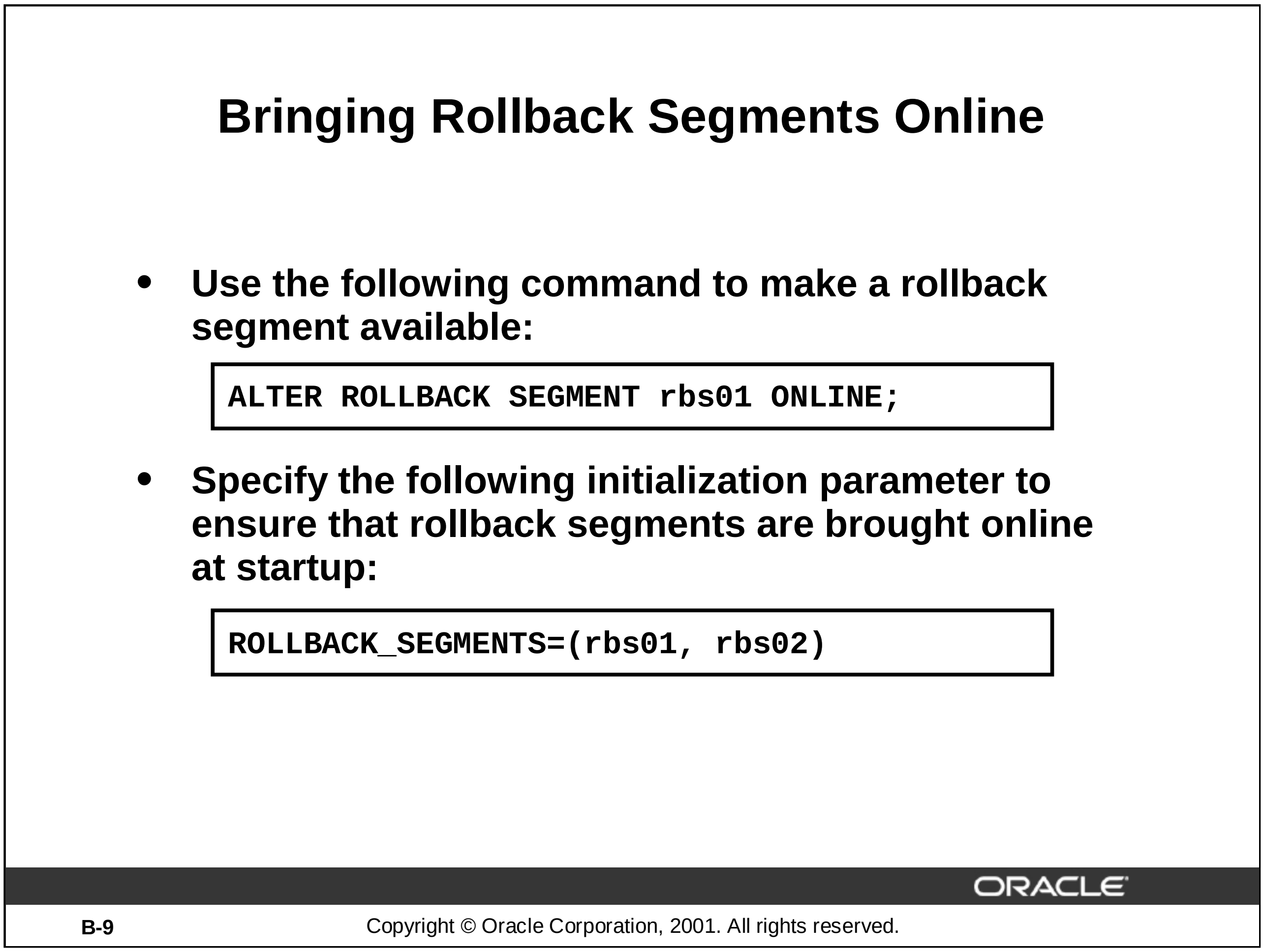
The `OPTIMAL` parameter specifies the size in bytes that a rollback segment must shrink to, if possible. Specifying `OPTIMAL` minimizes the waste of space in a rollback segment. If the `OPTIMAL` parameter is specified, a rollback segment can release space on completion of transactions that caused the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

The deallocation of extents is not done 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 these 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, as 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)
```

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

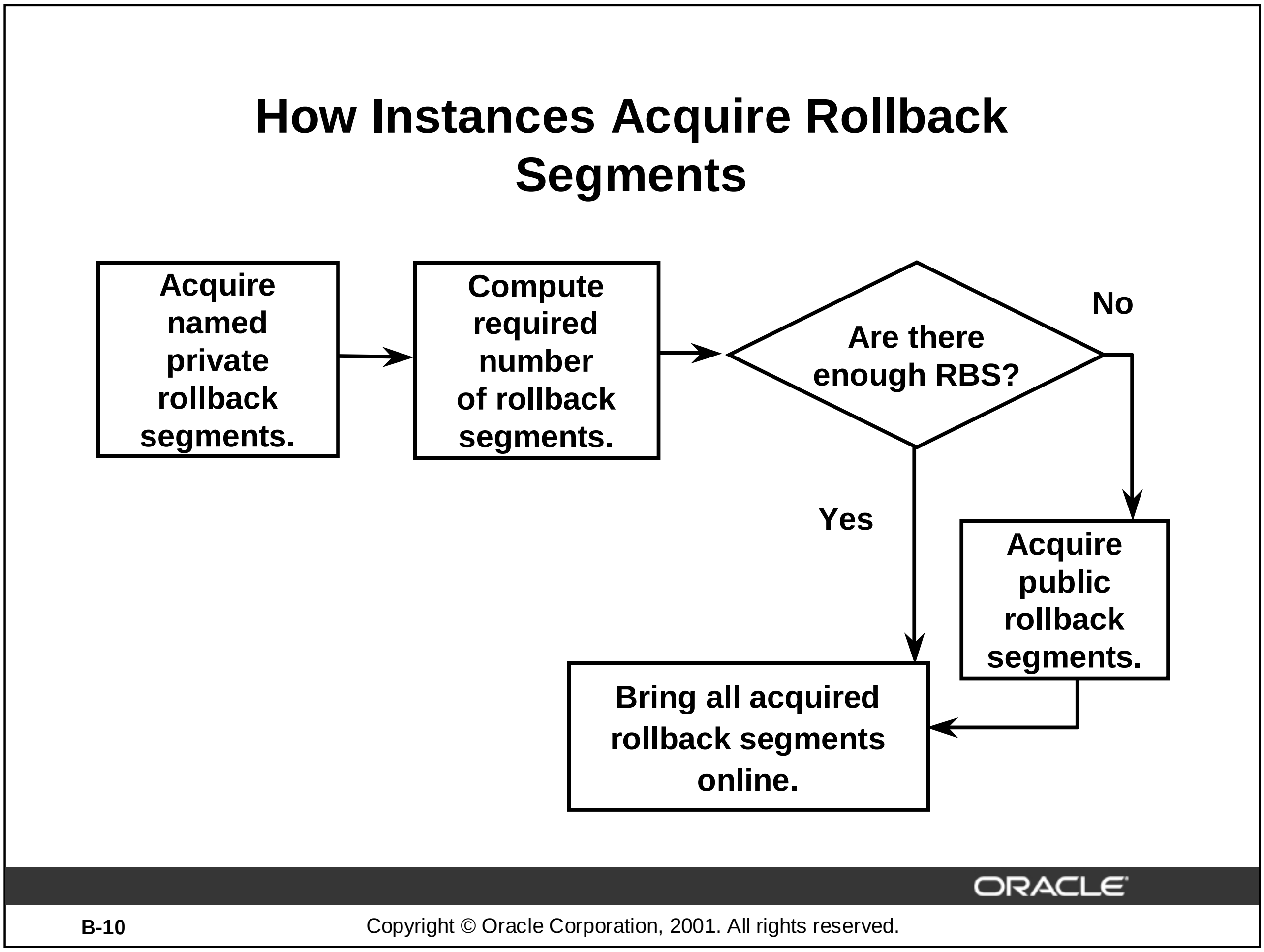
Use the following command to make a rollback segment available:

```
    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

Acquire named private rollback segments.

Compute required number of rollback segments.

Are there enough RBS?

No

Acquire public rollback segments.

Yes

Bring all acquired rollback segments online.

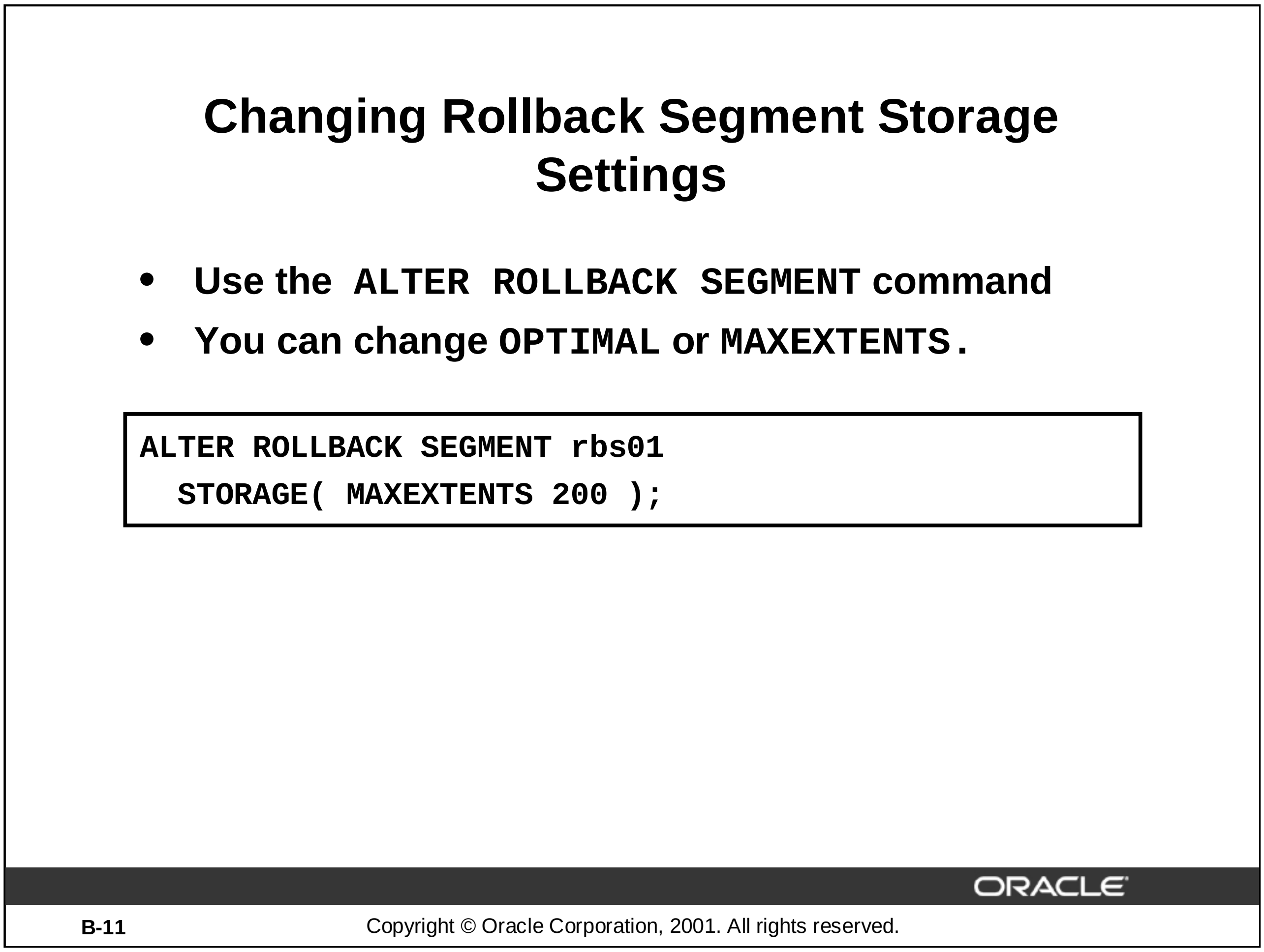
ORACLE

B-10

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 initialization parameter `ROLLBACK_SEGMENTS`.
- 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 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.`

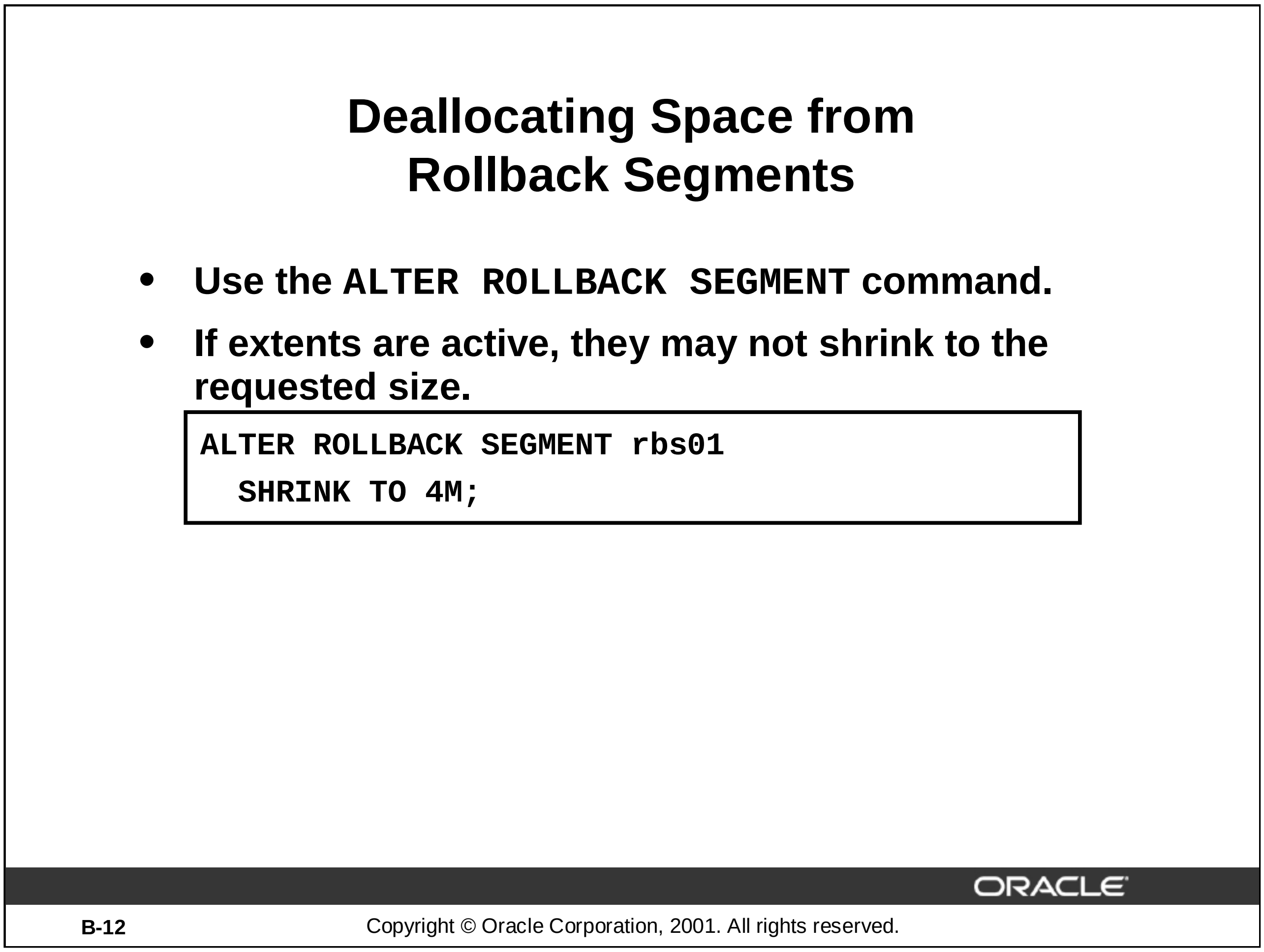
```
ALTER ROLLBACK SEGMENT rbs01
  STORAGE( MAXEXTENTS 200 );
```

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 may 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

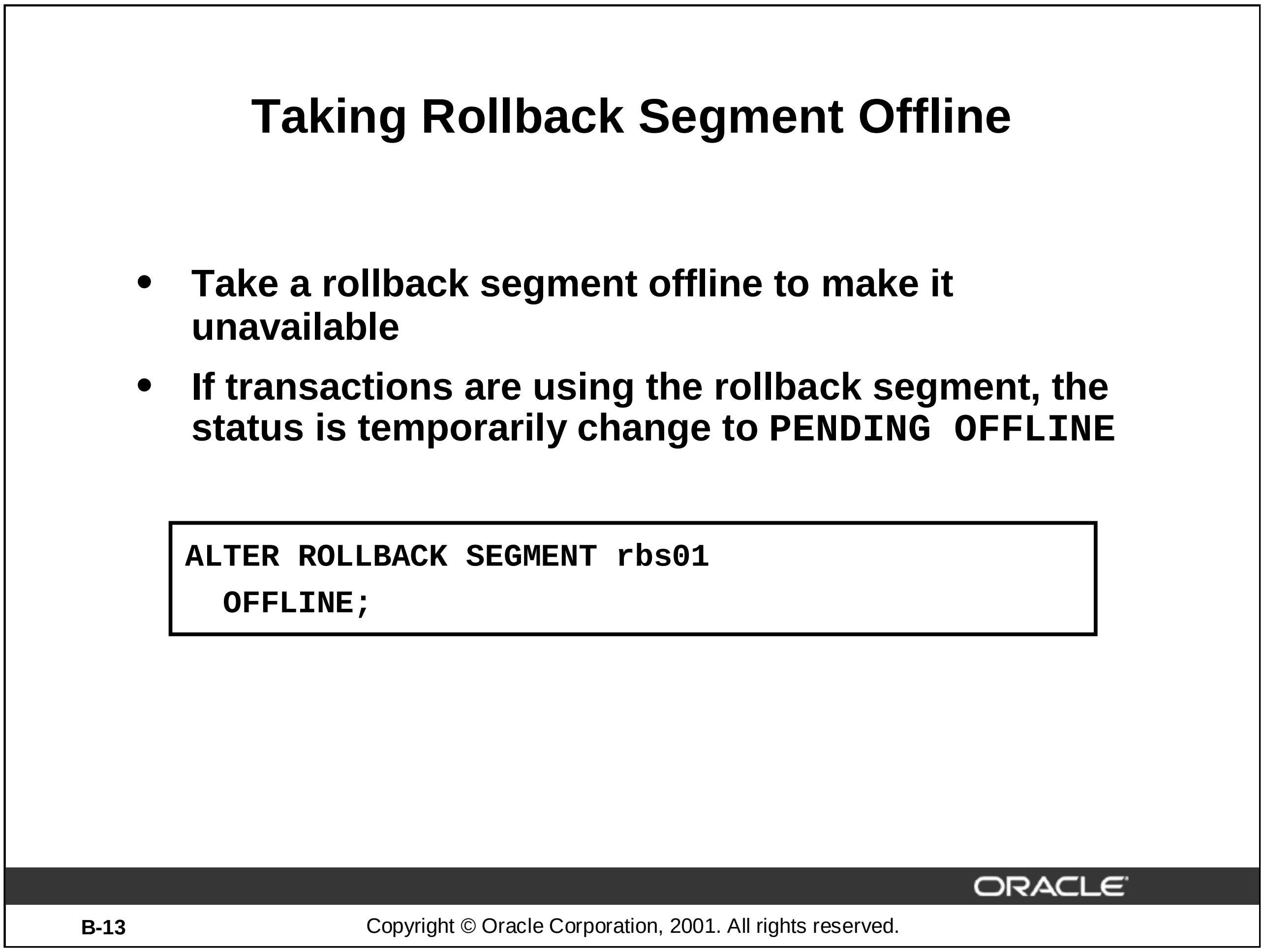
To manually deallocate space from a rollback segment, use the following command:

```
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, 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, 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 change to `PENDING OFFLINE`**

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

B-13

Taking a Rollback Segment Offline

Take a rollback segment offline:

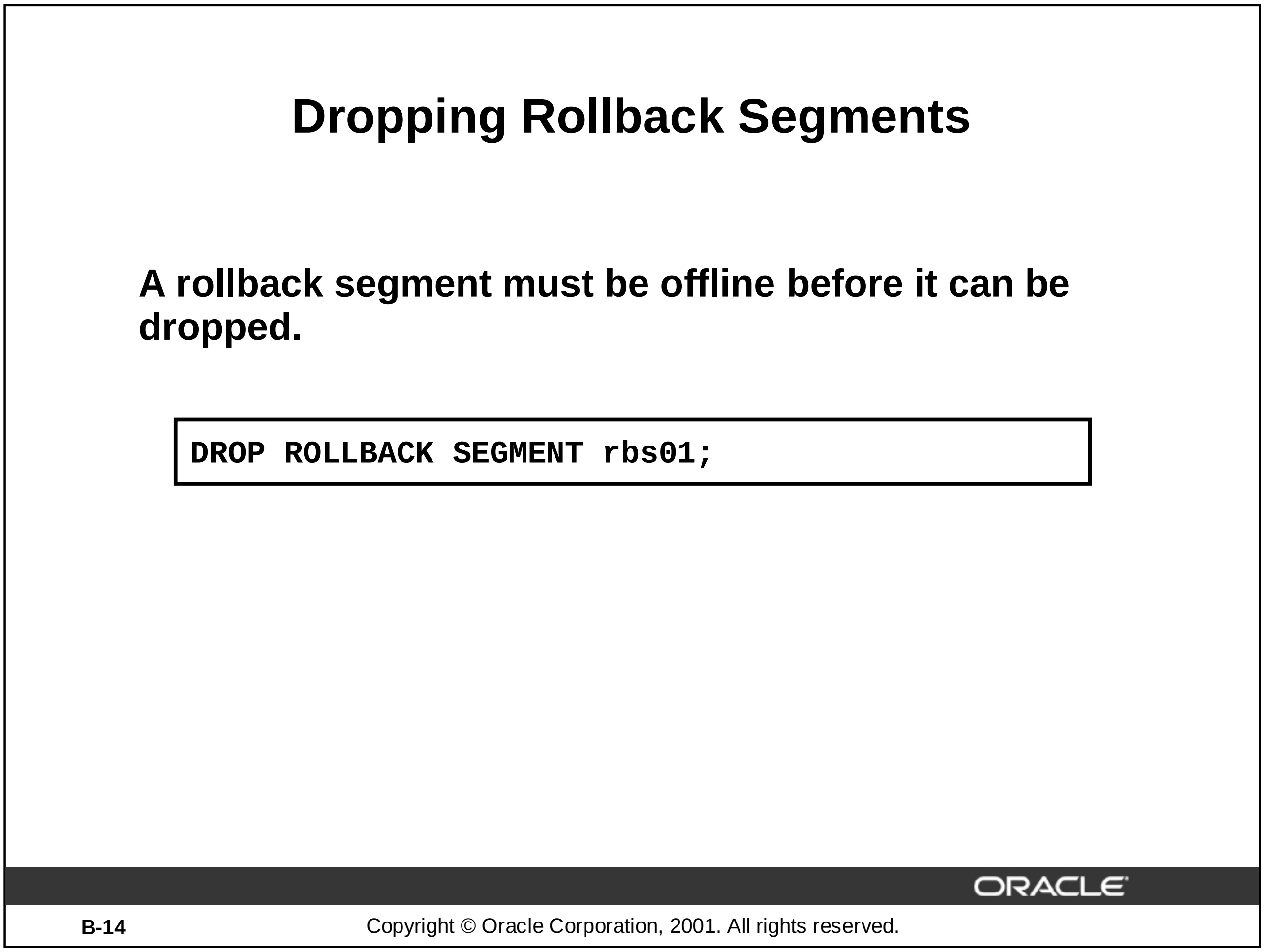
- To prevent new transactions from using a rollback segment
- If the rollback segment needs to be dropped

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, the status of the rollback segment is set to `PENDING OFFLINE`, as seen from the dynamic performance view, `V$ROLLSTAT`. 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.

```
DROP ROLLBACK SEGMENT rbs01;
```

B-14

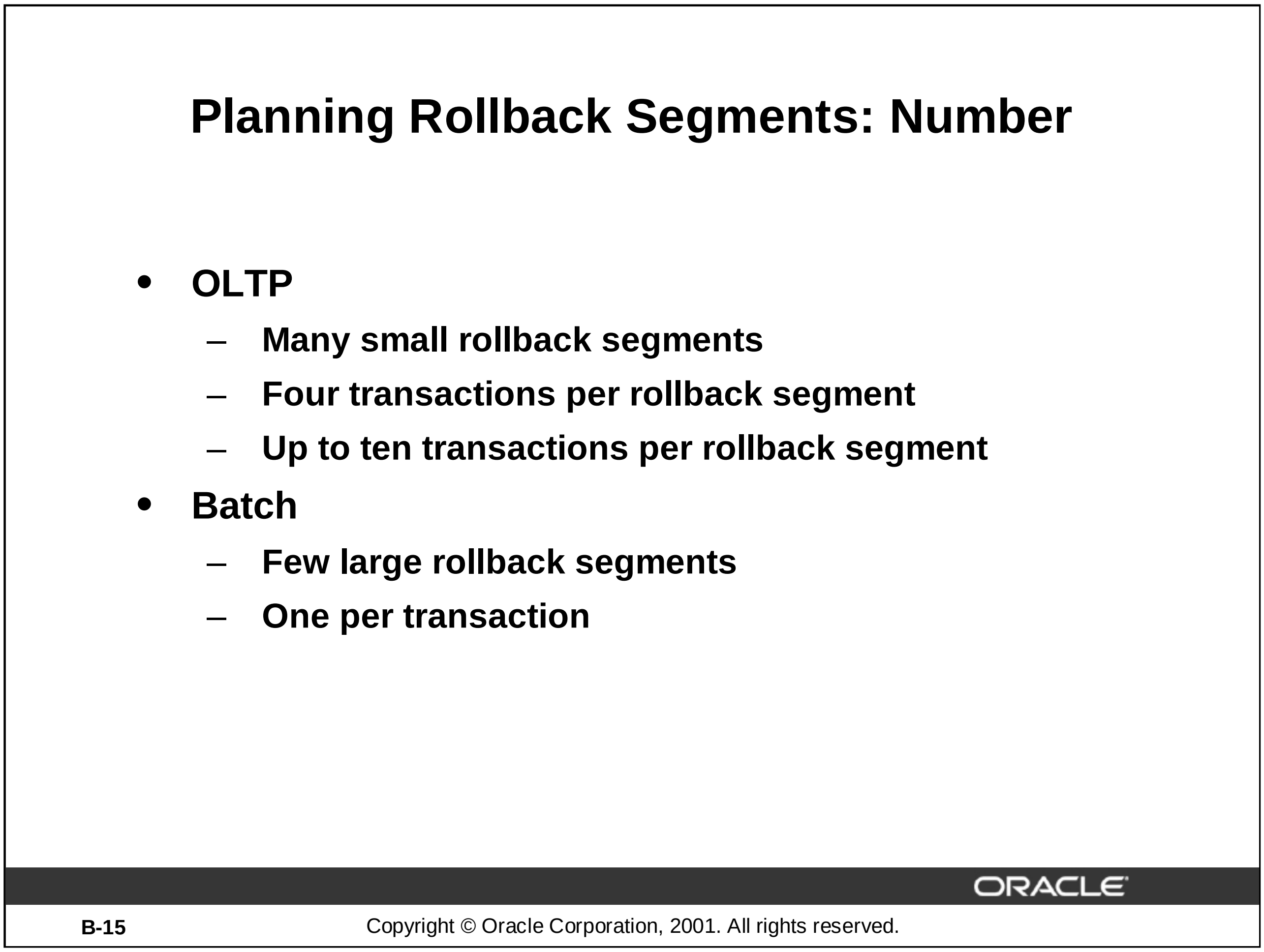
Dropping Rollback Segments

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.



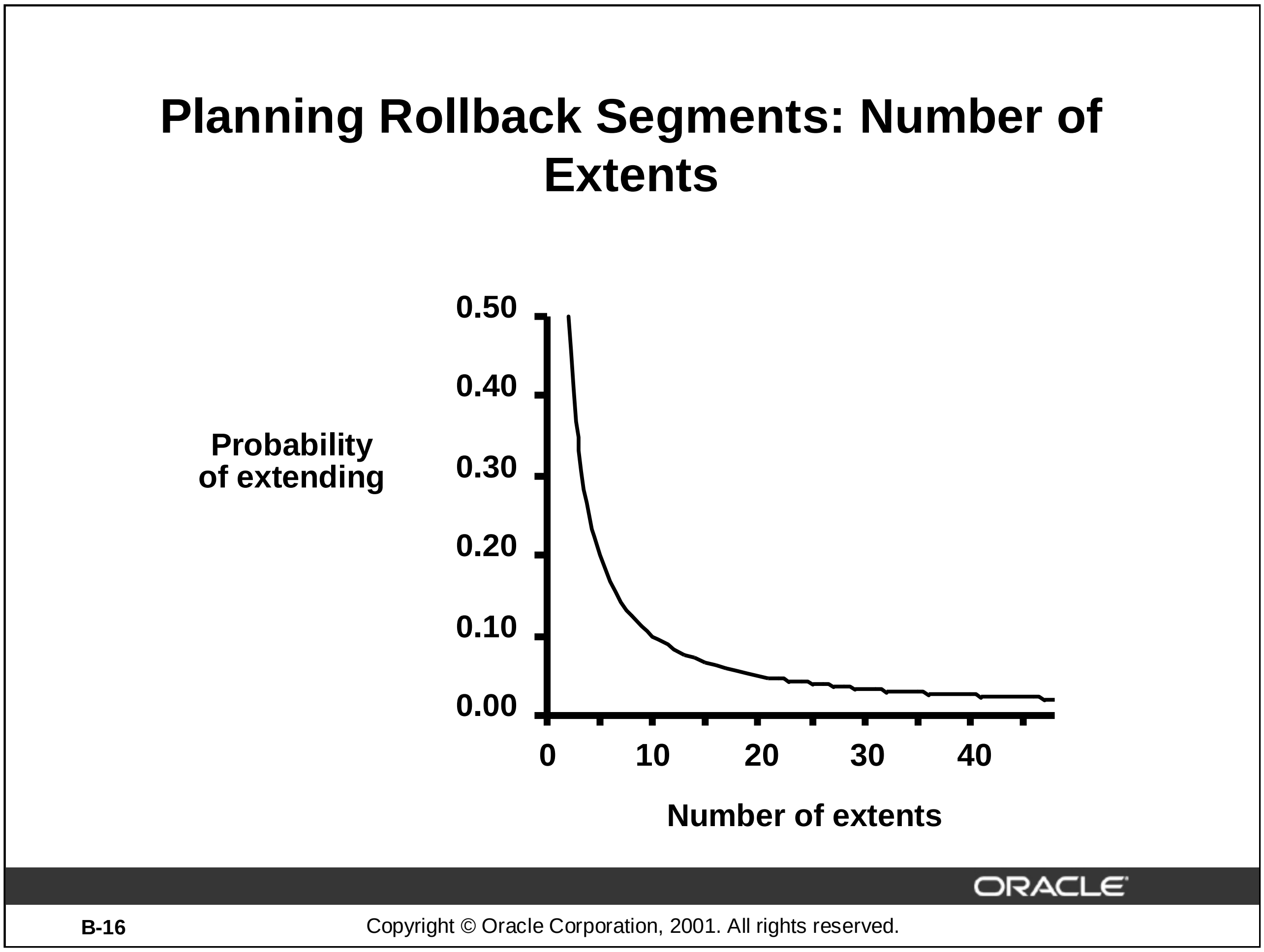
Planning Rollback Segments: Number

- 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

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 needs to 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. Hence, in a batch environment, allow for the growth of the rollback segments by creating them in large tablespaces.



Planning Rollback Segments: Number of Extents

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 being performed (insert, update, delete, and so on)
- The actual data being processed

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

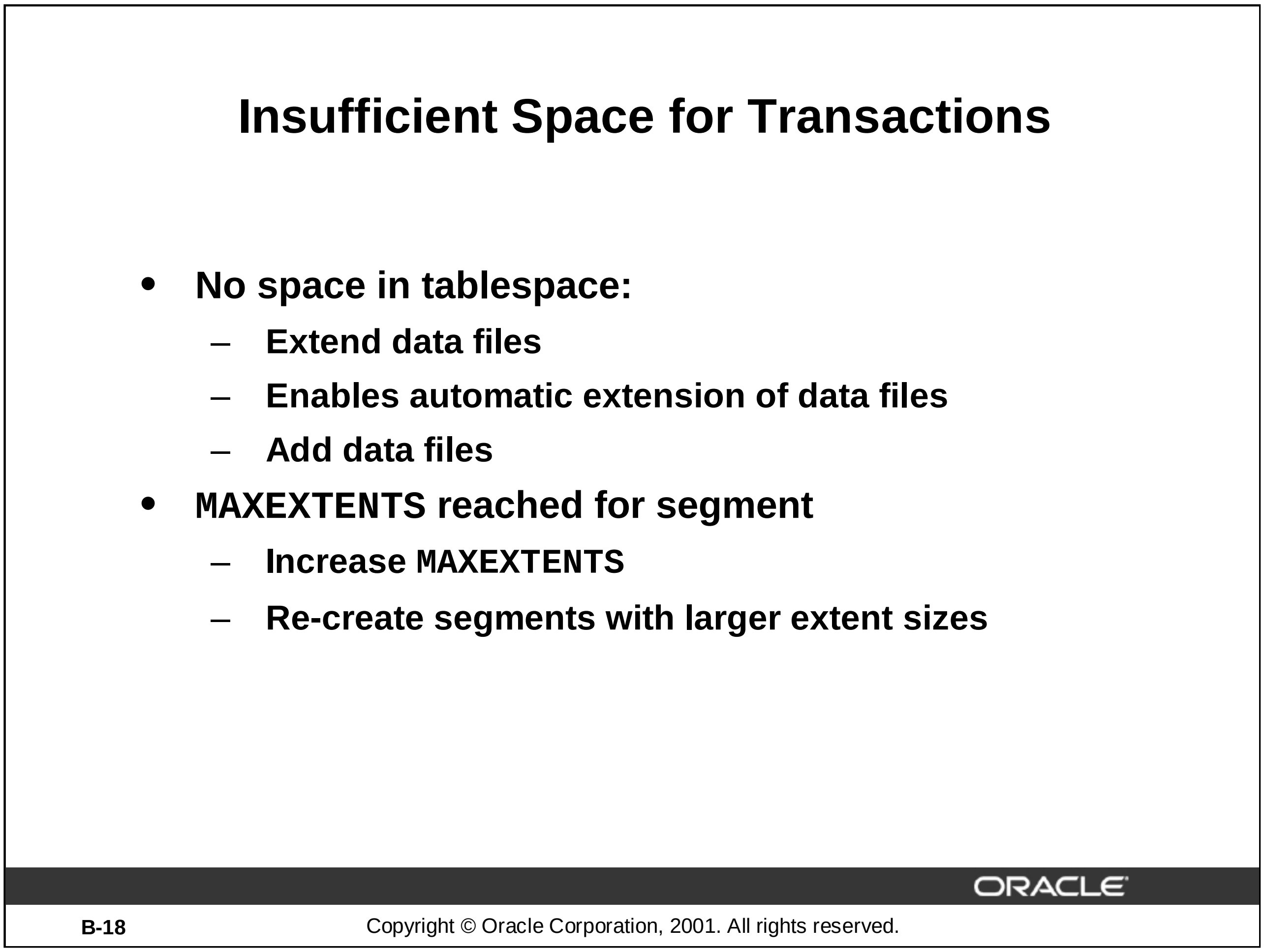
You can estimate the size of the rollback segment by running the longest transaction expected and checking the size of the rollback segment or the amount of rollback generated. The courses *Oracle8i: Performance Tuning* and *Oracle8i: SQL Statement Tuning* discuss monitoring statistics.

Number of Extents

It has been found that 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 error
- Blocking session
- Error in taking tablespace offline



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**

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).

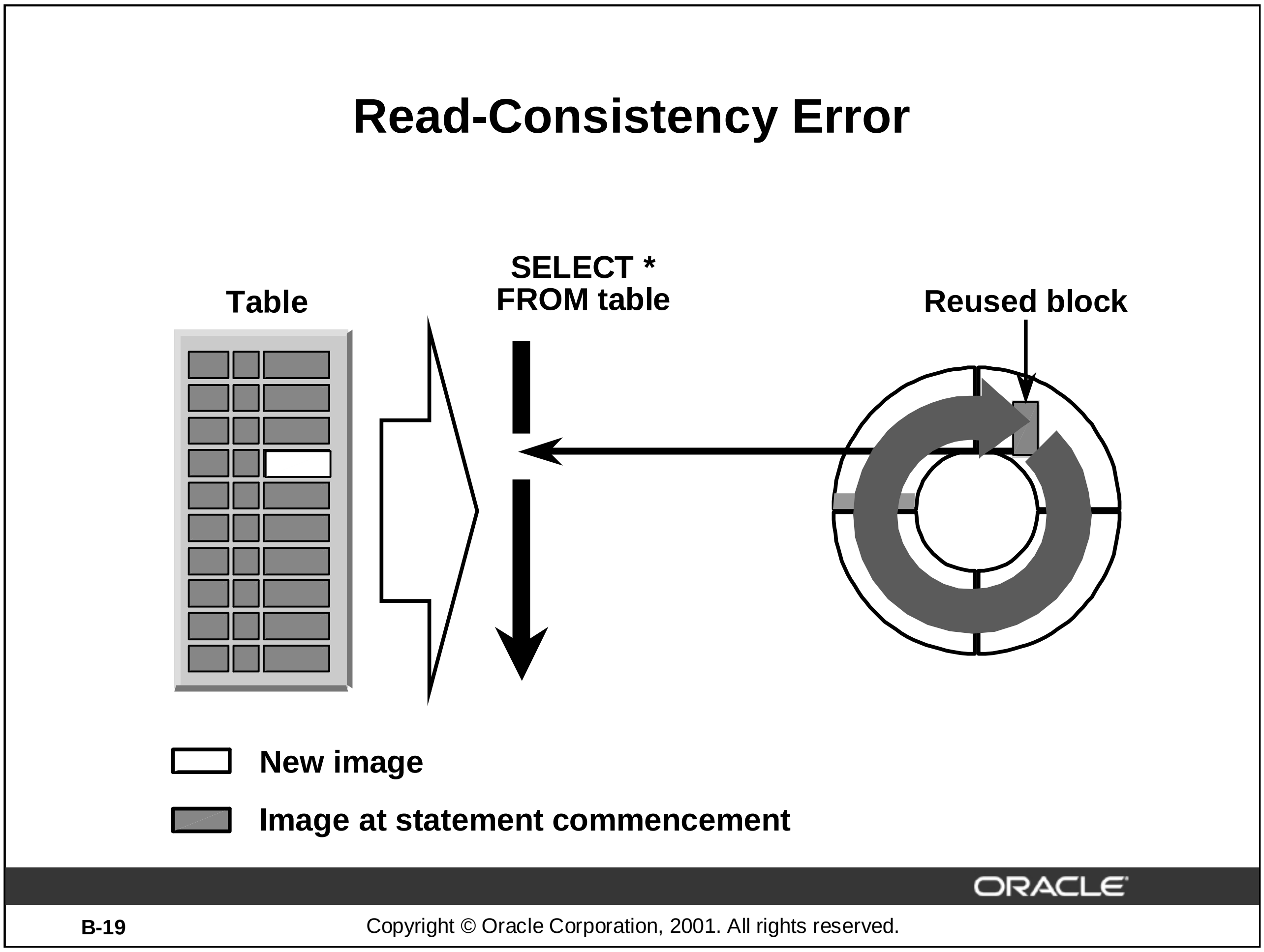
Solution

If the tablespace does not have free space, increase the space available by:

- 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 Error

Table

SELECT *
FROM table

Reused block

□ New image

▨ Image at statement commencement

B-19

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.

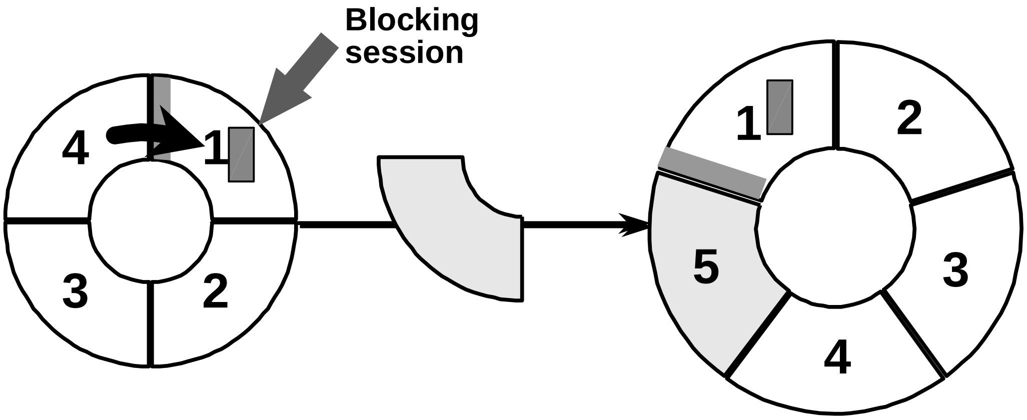
Solution

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 that these errors cannot be avoided by increasing `MAXEXTENTS`.

Blocking Session



□ Existing extent

▨ New extent

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)

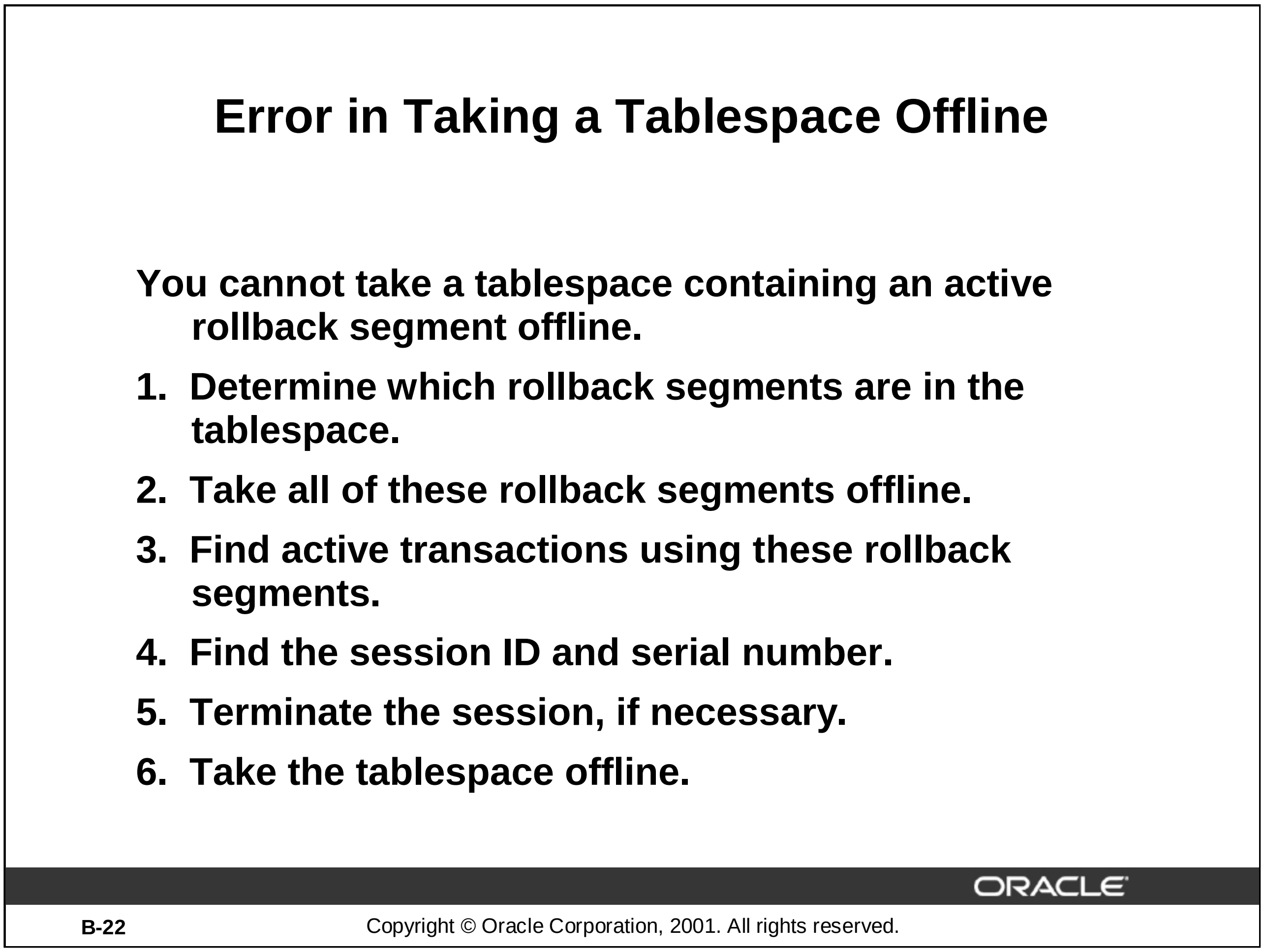
Solution

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.



Error in Taking a Tablespace Offline

You cannot take a tablespace containing an active rollback segment offline.

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.

B-22

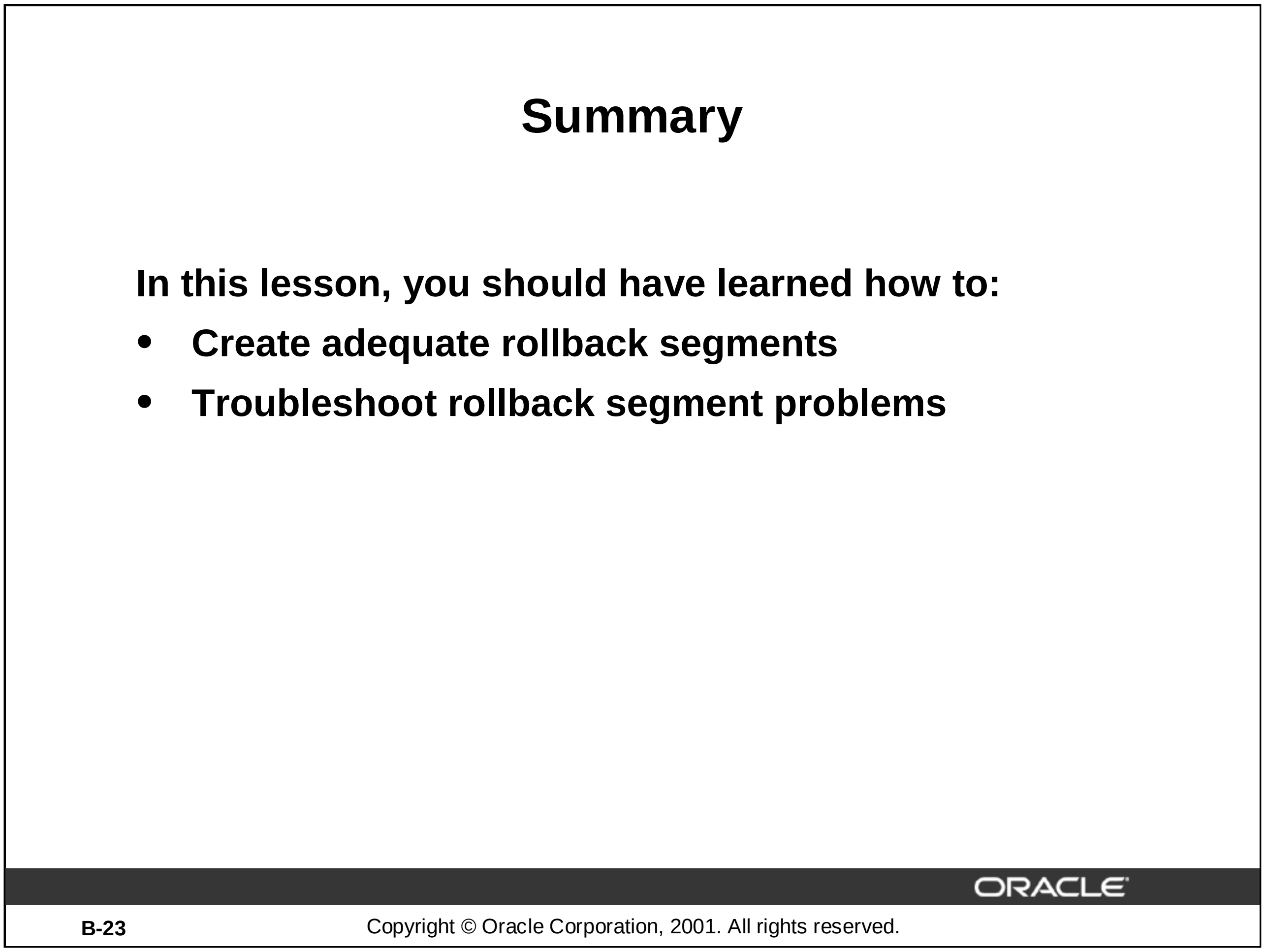
Problem Diagnosis and Resolution

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 user name and session information.
5. Kill the session or have the user end the transaction.
6. Take the tablespace offline.



Summary

In this lesson, you should have learned how to:

- Create adequate rollback segments
- Troubleshoot rollback segment problems

B-23

C

Practice Solutions for SQL*Plus

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 cache
d Shared Pool

Answer: B

3 Which two of the following statements are true about the Shared Pool?

a The shared Pool consists of the Library Cache, Data Dictionary Cache, Shared SQL area, Java Pool, and Large Pool.
b The Shared Pool is used to store the most recently executed SQL statements and the most recently used data.
c The Shared Pool is used for object that can be shared globally.
d The Library Cache consist of the Shared SQL and Shared PL/SQL areas.

Answer: B 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 cache
d Shared Pool

Answer: D

5 The primary purpose of the Redo Log Buffer Cache is to record all changes to the database data blocks.

a True
b False

Answer: True

6 The PGA is a memory region that contains data and control information for multiple server processes or multiple background processes.

a True
b False

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 one of the following processes is available when an Oracle instance is started.

a User process
b Background process
c Server process
d System process

Answer: B

8 Identify five mandatory background processes.

____
____
____
____
____

Answer: DBWR, LGWR, PMON, SMON, CKPT

9 Which one of the following memory areas is used to cache the data dictionary information?

a Database Buffer Cache
b PGA
c Redo log buffer cache
d Shared Pool

Answer: D

10 Match the process with its task.

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

Answer: E,C,D,B,A

11 The physical structure of an Oracle database consists of control files, data files, and redo log files.

a True
b False

Answer: True

12 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)

13 Identify the components of an Oracle server.

____

____

Answer: Oracle instance, Oracle database,

14 Identify the components of an Oracle instance.

____

____

Answer: SGA, Background processes

15 Identify three file types that make up an Oracle database

____

____

____

Answer: Datafiles, Control files, Redo log files

Practice 2: Solutions

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

```
Oracle Universal Installer
Oracle Database Configuration Assistant
Password File Utility
SQL*Plus
Oracle Enterprise Manager
```

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

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
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
Database mounted.
Database opened.
```

Practice 2: Solutions (continued)

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

```
SQL> COLUMN owner           FORMAT a5;
SQL> COLUMN table_name      FORMAT a20;
SQL> COLUMN tablespace_name FORMAT a15;
SQL> SELECT owner,table_name,tablespace_name
  2  FROM   dba_tables
  3  WHERE  rownum < 10;


OWNER TABLE_NAME           TABLESPACE_NAME
----- -------------------- ---------------
SYS   FILE$                SYSTEM
SYS   BOOTSTRAP$           SYSTEM
SYS   SEG$                 SYSTEM
SYS   ICOL$                SYSTEM
SYS   PROXY_ROLE_DATA$     SYSTEM
SYS   IND$                 SYSTEM
SYS   CDEF$                SYSTEM
SYS   OBJ$                 SYSTEM
SYS   CCOL$                SYSTEM
9 rows selected.
SQL> EXIT
```

Practice 2: Solutions (continued)

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

```
SQL> @$HOME/STUDENT/LABS/lab02_03.sql
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SET PAGESIZE 999
SQL> SPOOL para.lst;
SQL> SHOW parameter

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------
O7_DICTIONARY_ACCESSIBILITY          boolean     FALSE
active_instance_count                integer
aq_tm_processes                      integer     0
archive_lag_target                   integer     0
.
.
.
```

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

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

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

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

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

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

```
SQL> !rm $HOME/ADMIN/PFILE/orapw$ORACLE_SID
SQL> !orapwd file=$HOME/ADMIN/PFILE/orapw$ORACLE_SID password=oracle
entries=5
SQL> !chmod 660 $HOME/ADMIN/PFILE/orapw$ORACLE_SID
```

Practice 3: Solutions

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

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

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

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> CREATE SPFILE='$HOME/ADMIN/PFILE/spfile$ORACLE_SID.ora'
  2  FROM    PFILE='$HOME/ADMIN/PFILE/init$ORACLE_SID.ora';
File created.
```

Practice 3: Solutions (continued)

3 From the operating system, view the SPFILE.

```
SQL> !more $HOME/ADMIN/PFILE/spfile$ORACLE_SID.ora

*.background_dump_dest='/u01/home/db01/ADMIN/BDUMP'
*.compatible='9.0.0'
*.control_files='/u01/home/db01/ORADATA/u01/ctrl01.ctl'
*.core_dump_dest='/u01/home/db01/ADMIN/CDUMP'
*.db_block_size=4096
*.db_cache_size=4M
*.db_domain='world'
*.db_name='db01'
*.global_names=TRUE
*.instance_name='db01'
*.java_pool_size='0'
*.max_dump_file_size='10240'
*.remote_login_passwordfile='exclusive'
*.service_names='db01'
*.shared_pool_size=8M
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS'
*.user_dump_dest='/u01/home/db01/ADMIN/UDUMP'
```

4 Connect as user SYS AS SYSDBA, 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   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
Database mounted.
Database opened.
```

Practice 3: Solutions (continued)

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

```
INSERT INTO regions VALUES ( 5, ‘Mars’ );
```

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

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT
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
Database mounted.
SQL> ALTER DATABASE OPEN READ ONLY;
Database altered.
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'
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
```

Practice 3: Solutions (continued)

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

```
INSERT INTO regions VALUES ( 5, ‘Mars’ );
```

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

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

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

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

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

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

HR SESSION
SQL> ROLLBACK;
Rollback complete.
SQL> EXIT;
ERROR:
ORA-01089: immediate shutdown in progress - no operations are permitted
Disconnected from Oracle9i Enterprise Edition Release 9.0.0.0.0 - Beta
With the Partitioning option
JServer Release 9.0.0.0.0 - Beta (with complications)

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

Practice 3: Solutions (continued)

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

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

HR SESSION
SQL> CONNECT hr/hr
Connected.

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

HR SESSION
SQL> SELECT *
  2  FROM   regions;
REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
SQL> CONNECT hr/hr
ERROR:
ORA-01035: ORACLE only available to users with RESTRICTED SESSION
privilege
Warning: You are no longer connected to ORACLE.

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

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, 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 table is gathered from the control file and memory.
- **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 Find a list of the data dictionary views.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT table_name
  2  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
.
.
.
```

Practice 5: Solutions (continued)

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

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

```
SQL> SELECT name
  2  FROM   v$database;
NAME
---------
DB01
SQL> SELECT instance
  2  FROM   v$thread;
INSTANCE
----------------
db01
SQL> SELECT value
  2  FROM   v$parameter
  3  WHERE  name ='db_block_size';
VALUE
--------------------------------
4096
```

9 List the name and size of the data files, online redo log files, and the name of the control files.

Hint: Query the Dynamic Views `V$DATAFILE`, `V$LOGFILE`, and `V$CONTROLFILE`.

```
SQL> SELECT name
  2  FROM   v$datafile;
NAME
-------------------------------------------------------------------
/u01/home/db01/ORADATA/u01/system01.dbf
/u01/home/db01/ORADATA/u02/undotbs.dbf
/u01/home/db01/ORADATA/u03/users01.dbf
/u01/home/db01/ORADATA/u03/indx01.dbf
/u01/home/db01/ORADATA/u02/sample01.dbf
/u01/home/db01/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 data files that make up SYSTEM tablespace.

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

FILE_NAME
---------------------------------------
/u01/home/db01/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
----------------
           42700
SQL> SELECT sum(bytes)/1024 "used space in KB"
  2  FROM   dba_segments;
used space in KB
----------------
          131992
```

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
  2  FROM   dba_users;
USERNAME                       CREATED
------------------------------ ---------
SYS                            16-APR-01
SYSTEM                         16-APR-01
OUTLN                          16-APR-01
DBSNMP                         16-APR-01
ORDSYS                         16-APR-01
ORDPLUGINS                     16-APR-01
MDSYS                          16-APR-01
HR                             16-APR-01
OE                             16-APR-01
9 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$CONTROLFILE or V$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 *
  2  FROM   v$controlfile;
STATUS  NAME
------- ------------------------------------
        /u01/home/db01/ORADATA/u01/ctrl01.ctl
```

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

```
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
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.
```

Practice 6: Solutions (continued)

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

Hints:

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

```
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 $HOME/ORADATA/u02/ctrl02.ctl
SQL> STARTUP
ORACLE instance started.
Database mounted.
Database opened.
SQL> SELECT name
  2  FROM   v$controlfile;
NAME
--------------------------------------
/u01/home/db01/ORADATA/u01/ctrl01.ctl
/u01/home/db01/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 View `V$CONTROLFILE_RECORD_SECTION`.

```
SQL> SELECT records_total
  2  FROM   v$controlfile_record_section
  3  WHERE  type = 'DATAFILE';

RECORDS_TOTAL
-------------
           40
```

Practice 7: Solutions

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

Hints:

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

```
SQL> SELECT member
  2  FROM   v$logfile;
MEMBER
-------------------------------------
/u01/home/db01/ORADATA/u03/log02a.rdo
/u01/home/db01/ORADATA/u03/log01a.rdo
SQL> SELECT group#, members
  2  FROM   v$log;
    GROUP#    MEMBERS
---------- ----------
         1          1
         2          1
```

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

Hints:

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

```
SQL> SELECT log_mode
  2  FROM   v$database;
LOG_MODE
------------
NOARCHIVELOG

SQL> SELECT archiver
  2  FROM   v$instance;
ARCHIVE
-------
STOPPED
```

Practice 7: Solutions (continued)

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

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

Verify the result.

Hints:

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

```
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 *
  2  FROM   v$logfile;

GROUP# STATUS  TYPE    MEMBER
------ ------- ------- ------------------------------------
     2         ONLINE  /u01/home/db01/ORADATA/u03/log02a.rdo
     1 STALE   ONLINE  /u01/home/db01/ORADATA/u03/log01a.rdo
     1 INVALID ONLINE  /u01/home/db01/ORADATA/u04/log01b.rdo
     2 INVALID ONLINE  /u01/home/db01/ORADATA/u04/log02b.rdo
```

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

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

Verify the result.

Hints:

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

```
SQL> ALTER DATABASE ADD
  2     LOGFILE GROUP 3(
  3                      '$HOME/ORADATA/u03/log03a.rdo',
  4                      '$HOME/ORADATA/u04/log03b.rdo'
  5                    ) SIZE 1024K;
Database altered.
SQL> COLUMN GROUP# FORMAT 99
SQL> COLUMN MEMBER FORMAT a40
SQL> SELECT *
  2  FROM   v$logfile;
GROUP# STATUS  TYPE    MEMBER
------ ------- ------- ----------------------------------------
     2         ONLINE  /u01/home/db01/ORADATA/u03/log02a.rdo
     1 STALE   ONLINE  /u01/home/db01/ORADATA/u03/log01a.rdo
     1 INVALID ONLINE  /u01/home/db01/ORADATA/u04/log01b.rdo
     2 INVALID ONLINE  /u01/home/db01/ORADATA/u04/log02b.rdo
     3         ONLINE  /u01/home/db01/ORADATA/u03/log03a.rdo
     3         ONLINE  /u01/home/db01/ORADATA/u04/log03b.rdo
6 rows selected.
SQL> SELECT group#, members
  2  FROM   v$log;
GROUP#    MEMBERS
------ ----------
     1          2
     2          2
     3          2
```

Practice 7: Solutions (continued)

5 Remove the redo log group created in step 4.

Hints:

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

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.

SQL> SELECT group#, members
  2  FROM   v$log;
GROUP#    MEMBERS
------ ----------
     1          2
     2          2
     3          2

SQL> ALTER DATABASE DROP LOGFILE GROUP 3;
Database altered.
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. (Because we cannot resize log files, we have to add new logs and drop the old.)

Hints:

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

```
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
  8  ;
Database altered.
SQL> SELECT group#, status
  2  FROM   v$log;

GROUP# STATUS
------ ----------------
     1 ACTIVE
     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)

```
SQL> ALTER DATABASE DROP LOGFILE GROUP 1, GROUP 2;
Database altered.
SQL> SELECT group#, bytes
  2  FROM   v$log;

GROUP#      BYTES
------ ----------
     3    1048576
     4    1048576
```

Practice 8: Solutions

1 Create permanent tablespaces with the following names and storage:

a `DATA01` data dictionary managed.

b `DATA02` locally managed with uniform sized extents (Ensure that every used extent size in the tablespace is a multiple of 100 KB.)

c `INDX01` locally managed with uniform sized extents of 4K
(Enable automatic extension of 500 KB when more extents are required with a maximum size of 2 MB.)

d `RONLY` for read-only tables with the default storage. DO NOT make the tablespace read only at this time.

Display the information from the data dictionary.

```
SQL> CREATE TABLESPACE data01
  2  DATAFILE '$HOME/ORADATA/u04/data01.dbf' SIZE 2M
  3  EXTENT MANAGEMENT DICTIONARY;
Tablespace created.
SQL> CREATE TABLESPACE data02
  2  DATAFILE '$HOME/ORADATA/u03/data02.dbf' SIZE 1M
  3  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 100K;
Tablespace created.
SQL> CREATE TABLESPACE indx01
  2  DATAFILE '$HOME/ORADATA/u02/indx01.dbf' SIZE 1M
  3  AUTOEXTEND ON NEXT 500K MAXSIZE 2M
  4  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4K;
Tablespace created.
SQL> CREATE TABLESPACE ronly
  2  DATAFILE '$HOME/ORADATA/u01/ronly01.dbf' SIZE 1M;
Tablespace created.
```

Practice 8: Solutions (continued)

```
SQL> COLUMN name FORMAT a50
SQL> SET LINESIZE 80
SQL> SET PAGESIZE 999
SQL> SELECT name, bytes, create_bytes
  2  FROM   v$datafile;
NAME                                             BYTES CREATE_BYTES
-------------------------------------------- --------- ------------
/u01/home/db01/ORADATA/u01/system01.dbf      131072000            0
/u01/home/db01/ORADATA/u02/undotbs.dbf        31457280            0
/u01/home/db01/ORADATA/u03/users01.dbf         5242880            0
/u01/home/db01/ORADATA/u03/indx01.dbf          5242880            0
/u01/home/db01/ORADATA/u02/example01.dbf       5242880            0
/u01/home/db01/ORADATA/u01/querydata01.dbf     1048576            0
/u01/home/db01/ORADATA/u04/data01.dbf          2097152      2097152
/u01/home/db01/ORADATA/u03/data02.dbf          1048576      1048576
/u01/home/db01/ORADATA/u02/indx01.dbf          1048576      1048576
/u01/home/db01/ORADATA/u01/ronly01.dbf         1048576      1048576
10 rows selected.
```

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
---------------------------------------- ---------- ------------
/u01/home/db02/ORADATA/u03/data02.dbf       1536000      1048576
```

Practice 8: Solutions (continued)

3 Relocate tablespace `INDX01` to subdirectory `u06`.

```
SQL> ALTER TABLESPACE indx01 OFFLINE;
Tablespace altered.
SQL> SELECT name, status
  2  FROM   v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/u01/home/db01/ORADATA/u01/system01.dbf            SYSTEM
/u01/home/db01/ORADATA/u02/undotbs.dbf             ONLINE
/u01/home/db01/ORADATA/u03/users01.dbf             ONLINE
/u01/home/db01/ORADATA/u03/indx01.dbf              ONLINE
/u01/home/db01/ORADATA/u02/sample01.dbf            ONLINE
/u01/home/db01/ORADATA/u01/querydata01.dbf         ONLINE
/u01/home/db01/ORADATA/u04/data01.dbf              ONLINE
/u01/home/db01/ORADATA/u03/data02.dbf              ONLINE
/u01/home/db01/ORADATA/u02/indx01.dbf              OFFLINE
/u01/home/db01/ORADATA/u01/ronly01.dbf             ONLINE
10 rows selected.
SQL> !mv $HOME/ORADATA/u02/indx01.dbf $HOME/ORADATA/u06/indx01.dbf
SQL> ALTER TABLESPACE indx01
  2     RENAME DATAFILE
  3     '$HOME/ORADATA/u02/indx01.dbf' TO
  4     '$HOME/ORADATA/u06/indx01.dbf';
Tablespace altered.
SQL> ALTER TABLESPACE indx01 ONLINE;
Tablespace altered.
```

Practice 8: Solutions (continued)

```
SQL> SELECT name, status
  2  FROM   v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/u01/home/db01/ORADATA/u01/system01.dbf            SYSTEM
/u01/home/db01/ORADATA/u02/undotbs.dbf             ONLINE
/u01/home/db01/ORADATA/u03/users01.dbf             ONLINE
/u01/home/db01/ORADATA/u03/indx01.dbf              ONLINE
/u01/home/db01/ORADATA/u02/sample01.dbf            ONLINE
/u01/home/db01/ORADATA/u01/querydata01.dbf         ONLINE
/u01/home/db01/ORADATA/u04/data01.dbf              ONLINE
/u01/home/db01/ORADATA/u03/data02.dbf              ONLINE
/u01/home/db01/ORADATA/u06/indx01.dbf              ONLINE
/u01/home/db01/ORADATA/u01/ronly01.dbf             ONLINE
10 rows selected.
```

Practice 8: Managing Tablespaces and Data Files

4 Create a table in tablespace RONLY. Make tablespace RONLY read-only. Attempt to create an additional table. Drop the first created table. What happens and why?

```
SQL> CREATE TABLE table1 ( x CHAR(1))
  2  TABLESPACE ronly;
Table created.
SQL> ALTER TABLESPACE ronly READ ONLY;
Tablespace altered.
SQL> SELECT name, enabled, status
  2  FROM   v$datafile;
NAME                                               ENABLED    STATUS
-------------------------------------------------- ---------- ------
/u01/home/db01/ORADATA/u01/system01.dbf            READ WRITE SYSTEM
/u01/home/db01/ORADATA/u02/undotbs.dbf             READ WRITE ONLINE
/u01/home/db01/ORADATA/u03/users01.dbf             READ WRITE ONLINE
/u01/home/db01/ORADATA/u03/indx01.dbf              READ WRITE ONLINE
/u01/home/db01/ORADATA/u02/example01.dbf           READ WRITE ONLINE
/u01/home/db01/ORADATA/u01/querydata01.dbf         READ ONLY  ONLINE
/u01/home/db01/ORADATA/u04/data01.dbf              READ WRITE ONLINE
/u01/home/db01/ORADATA/u03/data02.dbf              READ WRITE ONLINE
/u01/home/db01/ORADATA/u06/indx01.dbf              READ WRITE ONLINE
/u01/home/db01/ORADATA/u01/ronly01.dbf             READ ONLY  ONLINE
10 rows selected.
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 datafile. Verify it.

```
SQL> DROP TABLESPACE ronly INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
SQL> SELECT *
  2  FROM   v$tablespace;
    TS# NAME         INC
------- ------------ ---
      0 SYSTEM       YES
      1 UNDOTBS      YES
      3 USERS        YES
      4 INDX         YES
      5 SAMPLE       YES
      2 TEMP         YES
      6 QUERY_DATA   YES
      7 DATA01       YES
      8 DATA02       YES
      9 INDX01       YES
10 rows selected.
SQL> !ls $HOME/ORADATA/u01/*
/u01/home/db01/ORADATA/u01/ctrl01.bak
/u01/home/db01/ORADATA/u01/querydata01.dbf
/u01/home/db01/ORADATA/u01/ctrl01.ctl
/u01/home/db01/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='$HOME/ORADATA/u05'
SCOPE=MEMORY;
System altered.
SQL> CREATE TABLESPACE data03
  2  DATAFILE SIZE 5M;
Tablespace created.
SQL> SELECT *
  2  FROM   v$tablespace;
    TS# NAME          INC
------- ------------- ---
      0 SYSTEM        YES
      1 UNDOTBS       YES
      3 USERS         YES
      4 INDX          YES
      5 SAMPLE        YES
      2 TEMP          YES
      6 QUERY_DATA    YES
      7 DATA01        YES
      8 DATA02        YES
      9 INDX01        YES
     11 DATA03        YES
11 rows selected.
SQL> !ls $HOME/ORADATA/u05
ora_data03_xg17n9nd.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
SQL> CONNECT system/manager
Connected.
SQL> CREATE TABLE emp ( empno    NUMBER(4),
  2                     ename    VARCHAR2(30),
  3                     job      VARCHAR2(9),
  4                     mgr      NUMBER(4),
  5                     hiredate DATE,
  6                     sal      NUMBER(7,2),
  7                     comm     NUMBER(7,2),
  8                     deptno   NUMBER(2) )
  9     TABLESPACE data01
 10     STORAGE ( INITIAL 100K
 11               NEXT 100K
 12               PCTINCREASE 0
 13               MINEXTENTS 8
 14               MAXEXTENTS 10 );
Table created.
SQL> CREATE TABLE fragment1( a NUMBER )
  2     TABLESPACE data01
  3     STORAGE( INITIAL 10K );
Table created.
SQL> CREATE TABLE dept ( deptno NUMBER,
  2                      dname  VARCHAR2(15),
  3                      loc    VARCHAR2(20) )
  4     TABLESPACE data01
  5     STORAGE( INITIAL 50K
  6              NEXT 50K );
Table created.
SQL> CREATE TABLE fragment2( a NUMBER )
  2     TABLESPACE data01
  3     STORAGE ( INITIAL 8K );
Table created.
```

Practice 9: Solutions (continued)

```
SQL> CREATE TABLE big_emp ( empno NUMBER(4),
  2                         ename VARCHAR2(30) )
  3     TABLESPACE data01
  4     STORAGE ( INITIAL 1M
  5               NEXT 1M
  6               MAXEXTENTS 10 );
Table created.
SQL> CREATE INDEX i_e_empno
  2     ON emp(ename)
  3     TABLESPACE indx01
  4     STORAGE ( INITIAL 50K
  5               NEXT 50K );
Index created.
SQL> DROP TABLE fragment1;
Table dropped.
SQL> DROP TABLE fragment2;
Table dropped.
```

Practice 9: Solutions (continued)

2 Identify the different types of segments in the database.

```
SQL> CONNECT system/manager
Connected.
SQL> SELECT DISTINCT segment_type
  2  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.

```
SQL> CONNECT system/manager
Connected.
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> CONNECT system/manager
Connected.
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
-------------------------------------
/u01/home/db01/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
SAMPLE                                  1    2555904    2555904
INDX                                    1    5120000    5120000
INDX01                                  1     917504     917504
QUERY_DATA                              1     983040     983040
SYSTEM                                  1    4943872    4943872
UNDOTBS                                15   24903680    6750208
USERS                                   1    5177344    5177344
10 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 (SELECT 1
  5                     FROM   dba_free_space f
  6                     WHERE   s.tablespace_name=f.tablespace_name
  7                     HAVING  max(f.bytes) > s.next_extent)
  8  ;

SEGMENT_NAME SEGMENT_TYPE TABLESPACE_NAME NEXT_EXTENT
------------ ------------ --------------- -----------
BIG_EMP      TABLE        DATA01              1048576
```

Practice 10: Solutions

1 Connect as user SYS, and list the undo segments in tablespace UNDOTBS.

```
SQL> CONNECT / AS SYSDBA
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$
8 rows selected.
```

2 Create undo tablespace UNDO2, size 15M, in $HOME/oradata/u03. List the rollback segments in tablespace UNDO2.

```
SQL> CREATE UNDO TABLESPACE undo2
  2  DATAFILE '$HOME/ORADATA/u03/undo2.dbf' size 15M;
Tablespace created.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDO2';
SEGMENT_NAME
--------------------
_SYSSMU9$
_SYSSMU10$
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
8 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 (department_id, department_name)
  2  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 for the instance.

```
SQL> ALTER SYSTEM SET undo_tablespace='UNDO2' SCOPE=BOTH;
System altered.
```

5 As SYS drop tablespace UNDOTBS. What happened? Why?

```
SQL> DROP TABLESPACE undotbs 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.

```
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
8 rows selected.
SQL> SELECT a.usn,a.name,b.status
  2  FROM   v$rollname a, v$rollstat b
  3  WHERE  a.name IN ( SELECT segment_name
  4                     FROM   dba_segments
  5                     WHERE  tablespace_name = 'UNDOTBS'
  6                   )
  7  AND a.usn = b.usn;

  USN NAME                                       STATUS
----- -------------------------------------- ---------------
    2 _SYSSMU2$                                  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;
```

8 In the session connected as SYS drop tablespace UNDOTBS. What happened? Why?

```
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? Why?

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 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:

Table	Column	Data Type and Size
CUSTOMERS	CUST_CODE	VARCHAR2(3)
	NAME	VARCHAR2(50)
	REGION	VARCHAR2(5)
ORDERS	ORD_ID	NUMBER(3)
	ORD_DATE	DATE
	CUST_CODE	VARCHAR2(3)
	DATE_OF_DELY	DATE

You have been informed that in the table ORDERS, rows will be inserted without a value for DATE_OF_DELY, and it will be updated when the order is fulfilled. Use tablespace USERS. You can use the default storage settings.

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE TABLE customers ( cust_code VARCHAR2(3),
  2                           name      VARCHAR2(50),
  3                           region    VARCHAR2(5)   )
  4  TABLESPACE users;
Table created.
SQL> CREATE TABLE orders ( 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> CONNECT system/manager
Connected.
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 table `ORDERS`.

```
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, but with MINEXTENTS=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
  5     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 CHECKPOINT 1000;
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 the indexes, naming them CUST_NAME_IDX and CUST_REGION_IDX, respectively, and placing them in the appropriate tablespaces.

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.

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE INDEX cust_name_idx
  2     ON customers(name)
  3     TABLESPACE indx01;
Index created.
SQL> CREATE BITMAP INDEX cust_region_idx
  2    ON system.customers(region)
  3    TABLESPACE indx01;
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.
```

3 Note the files and blocks used by the extents by CUST_REGION_IDX index.

Hint: Use the view DBA_EXTENTS to get this information.

```
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
```

Practice 12: Solutions (continued)

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.

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
```

b Query the table NUMBERS 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
```

Practice 12: Solutions (continued)

5 c Create B-tree indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively, and check the total sizes of the indexes. Put the indexes in tablespace INDX01.

Hint: Check the total blocks allocated to the extents from DBA_SEGMENTS.

```
SQL> CREATE INDEX numb_oe_idx
  2    ON numbers(odd_even)
  3    TABLESPACE indx01;
Index created.
SQL> CREATE INDEX numb_no_idx
  2    ON numbers(no)
  3    TABLESPACE indx01;
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 Create bitmap indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively, and check the total sizes of the indexes. Put the indexes in tablespace INDX01.
What can you conclude about the relationship between cardinality and sizes of the two types of indexes?

Hint: The existing indexes need to be dropped before creating the new indexes. Now re-execute the query to check the sizes of the indexes.

```
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 indx01;
Index created.
SQL> CREATE BITMAP INDEX numb_no_idx
  2     ON numbers(no)
  3     TABLESPACE indx01;
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
```

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 the script `lab13_01.sql`. Run the script to create the constraints.

```
SQL> @$HOME/STUDENT/LABS/lab13_01.sql
```

2 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
------------------------- ---------- - --------------- ----------
CUSTMOERS_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.
```

Practice 13: Solutions (continued)

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', 'CUSTOMERS')
  7           AND    owner='SYSTEM'
  8           AND    constraint_type in ('P','U')
  9         )
 10  ;

INDEX_NAME                     TABLE_NAME UNIQUENES
------------------------------ ---------- ---------
CUSTOMERS_CUST_CODE_PK         CUSTOMERS  NONUNIQUE
ORDERS_ORD_ID_PK               ORDERS     UNIQUE
```

3 Insert two records with the following values into the `PRODUCTS` table:

PRODUCT_ID	PRODUCT_DESCRIPTION	LIST_PRICE
4000	Unix Monitor	3620
4000	NT Monitor	2400

```
SQL> INSERT INTO system.products
  2    VALUES(4000,'UNIX Monitor',3620);
1 row created.
SQL> INSERT INTO system.products
  2    VALUES(4000,'NT Monitor', 2400);
1 row created.
SQL> COMMIT;
Commit complete.
```

Practice 13: Solutions (continued)

4 Enable the unique constraint on the PRODUCT table. Was it successful? Why or why not?

```
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 that new rows added to the table do not violate the constraint on the PRODUCT 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
```

Practice 13: Solutions (continued)

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
```

Practice 13: Solutions (continued)

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
```

c Use the ROWIDs 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 ( SELECT row_id
  4                    FROM   exceptions
  5                    WHERE  table_name='PRODUCTS'
  6                  )
  7  ;

ROWID              PROD_CODE DESCRIPTION
------------------ ---------- ------------------------------
AAABSgAAIAAAAASAAA       4000 UNIX Monitor
AAABSgAAIAAAAASAAB       4000 NT Monitor
```

Practice 13: Solutions (continued)

6 **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'
  6                  )
  7  ;
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? Roll back the changes.

```
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? Why or why not?

```
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 Run the lab14_01.sql script to create user Jeff. Enable password management by running script @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql.

```
SQL> @$HOME/STUDENT/LABS/lab14_01.sql
SQL> @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql
```

2 Try to change the password of 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 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$
ERROR:
ORA-28000: the account is locked
```

6 Using data dictionary view DBA_USERS verify user Jeff is locked. Unlock the account for the 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
OUTLN           OPEN
ORDSYS          OPEN
MDSYS           OPEN
OE              OPEN
HR              OPEN
ORDPLUGINS      OPEN
DBSNMP          OPEN
JEFF            LOCKED(TIMED)
10 rows selected.

SQL> ALTER USER jeff
  2     ACCOUNT UNLOCK;
User altered.
SQL> CONNECT jeff/super1$
Connected.
```

7 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.
```

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 USERS and INDX tablespaces.

Hint: Ensure that the temporary tablespace temp is assigned. Use the lab15_01.sql script to grant Bob the ability to create sessions.

```
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.
```

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? Why?

```
SQL> connect bob/crusader;
Connected.
SQL> ALTER USER bob
  2  TEMPORARY TABLESPACE users;
ALTER USER bob
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

b As `Bob`, change his password to `SAM`.

```
SQL> connect bob/crusader;
Connected.
SQL> ALTER USER bob
  2  IDENTIFIED BY sam;
User altered.
```

Practice 15: Solutions (continued)

6 As SYSTEM, remove Bob's quota on his default tablespace.

```
SQL> CONNECT system/manager
Connected.
SQL> ALTER USER bob QUOTA 0 ON users;
User altered.
```

7 Remove Emi's account from the database.

Hint: Because Emi owns tables, you need to use the CASCADE option.

```
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 SYSTEM, create user Emi and give her the capability to log on to the database and create objects in her schema.

```
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** Connect as Emi, and create tables using the script lab16_02a.sql to create the tables CUSTOMERS and ORDERS.

```
SQL> CONNECT emi/abcd12;
Connected.
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 CUSOMTERS table. Verify that records have been inserted.

```
SQL> CONNECT system/manager;
Connected.
SQL> INSERT INTO emi.customers
  2     SELECT *
  3     FROM   system.customers;
9 rows created.
SQL> SELECT * FROM emi.customers;

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 SYSTEM give Bob the ability to select from Emi's CUSTOMERS table. What happens and why?

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT select ON emi.customers TO bob;
GRANT select ON emi.customers TO bob
                    *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Practice 16: Solutions (continued)

3 Reconnect as Emi and give Bob the ability to select from Emi's CUSTOMERS table. Also, enable Bob to give the select capability to other users. Examine the data dictionary views that record these actions.

```
SQL> CONNECT emi/abcd12;
Connected.
SQL> GRANT select ON customers
  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 hiearchy   FORMAT a3
SQL> SELECT *
  2  FROM   dba_tab_privs
  3  WHERE  grantee='BOB';

GRANTEE  OWNER    TABLE_NAME GRANTOR  PRIVILEGE  GRA HIE
-------- -------- ---------- -------- ---------- --- ---
BOB      EMI      CUSTOMERS  EMI      SELECT     YES NO
```

4 Create user Trevor with the capability to log on to the database.

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE USER trevor IDENTIFIED BY "abcd1?";
User created.
SQL> GRANT create session TO trevor;
Grant succeeded.
```

Practice 16: Solutions (continued)

5 **a** As Bob, enable Trevor to access Emi's CUSTOMERS table. Give Bob the new password sam.

```
SQL> CONNECT bob/olink
ERROR:
ORA-28001: the password has expired
Changing password for bob
New password:
Retype new password:
Password changed
Connected.
SQL> GRANT select ON emi.customers TO trevor;
Grant succeeded.
```

b As Emi, remove Bob's privilege to read Emi's CUSTOMERS table.

```
SQL> CONNECT emi/abcd12;
Connected.
SQL> REVOKE select ON customers FROM bob;
Revoke succeeded.
```

c As Trevor, query Emi's CUSTOEMRS table. What happens and why?

```
SQL> CONNECT trevor/abcd1?;
Connected.
SQL> SELECT *
  2  FROM emi.customers;
FROM emi.customers
         *
ERROR at line 2:
ORA-00942: table or view does not exist
```

Practice 16: Solutions (continued)

6 a Enable `Emi` to create tables in any schema. As `Emi`, create the table `ORDERS` in `Bob`'s schema as a copy of `EMI.ORDERS`. What happened and why?

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT create any table TO emi;
Grant succeeded.
SQL> CONNECT emi/abcd12
Connected.
SQL> CREATE TABLE bob.orders
  2  AS
  3  SELECT *
  4  FROM   orders;
FROM   orders
       *
ERROR at line 4:
ORA-01536: space quota exceeded for tablespace 'USERS'
```

b As `SYSTEM`, examine the data dictionary view `DBA_TABLES` to check the result.

```
SQL> CONNECT system/manager
Connected.
SQL> SELECT owner, table_name
  2  FROM   dba_tables
  3  WHERE  table_name IN ('CUSMTERS', 'ORDERS');

OWNER    TABLE_NAME
-------- ----------
OE       ORDERS
SYSTEM   ORDERS
EMI      ORDERS
```

7 Enable `Emi` to start up and shut down the database without the ability to create a new database.

```
SQL> CONNECT sys/oracle 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 CLUSTER       NO
RESOURCE   CREATE INDEXTYPE     NO
RESOURCE   CREATE OPERATOR      NO
RESOURCE   CREATE PROCEDURE     NO
RESOURCE   CREATE SEQUENCE      NO
RESOURCE   CREATE TABLE         NO
RESOURCE   CREATE TRIGGER       NO
RESOURCE   CREATE TYPE          NO
8 rows selected.
```

2 Create a role called DEV, which enables a user to create a table, create a view and select from Emi's CUSTOMERS 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 customers TO dev;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> GRANT dev TO bob;
Grant succeeded.
```

Practice 17: Solutions (continued)

3 **a** Assign the RESOURCE and DEV roles to Bob, but make only the RESOURCE role to be 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.
```

4 Bob needs to check the undo segments that are currently used by the instance. Connect as Bob and list the undo segments used.

```
SQL> CONNECT bob/sam
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
_SYSSMU9$
_SYSSMU10$
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
9 rows selected.
```

Practice 17: Solutions (continued)

5 As `SYSTEM`, try to create a view `CUST_VIEW` on `Emi`’s `CUSTOMERS` table. What happens and why?

```
SQL> connect system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2     SELECT *
  3     FROM   emi.customers;
   FROM   emi.customers
              *
ERROR at line 3:
ORA-01031: insufficient privileges
```

6 As user `Emi` grant select on customers to `SYSTEM`. As `SYSTEM` try to create view `CUST_VIEW` on `Emi`’s `CUSTOMERS` table. What happens and why?

```
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers TO system;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2     SELECT *
  3     FROM   emi.customers;

View created.
```

Practice 18: Solutions

1 Check the database and the national character set.

```
SQL> CONNECT sys/oracle 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
.
.
.
```

Practice 18: 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 modifiee.
SQL> SELECT sysdate
  2  FROM   dual;

SYSDATE
-----------
29-AVR-2001
```