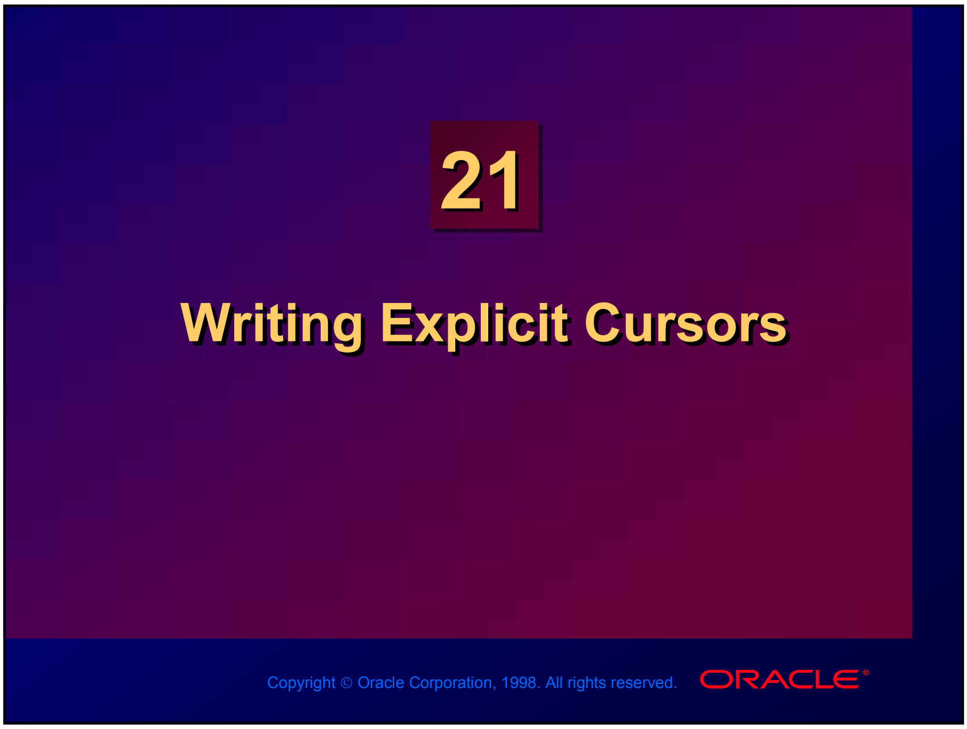
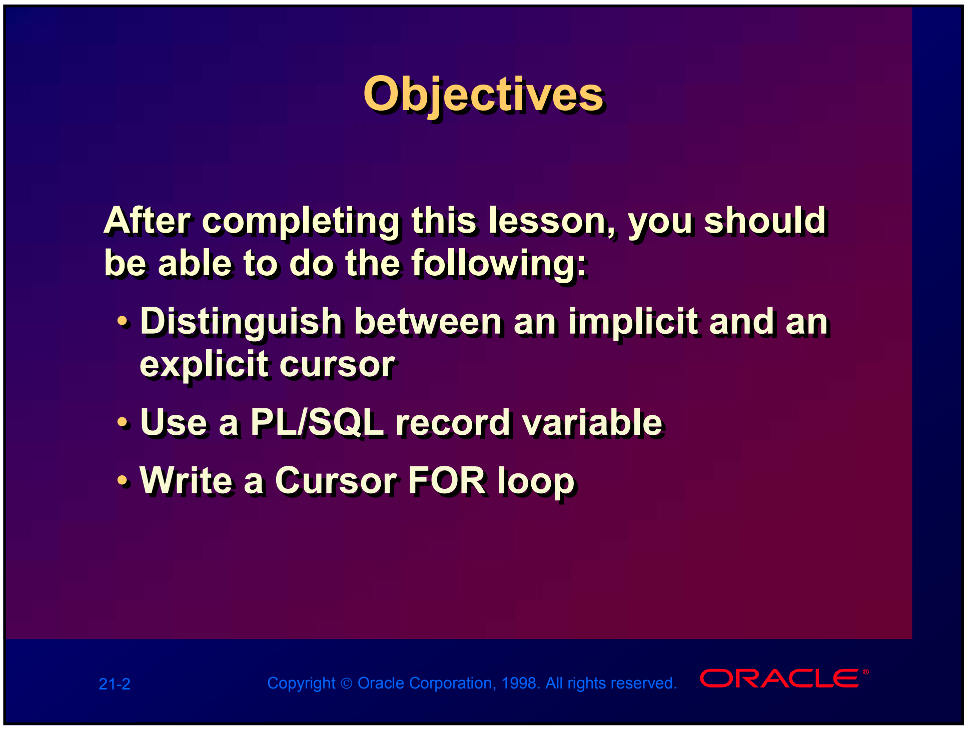


21

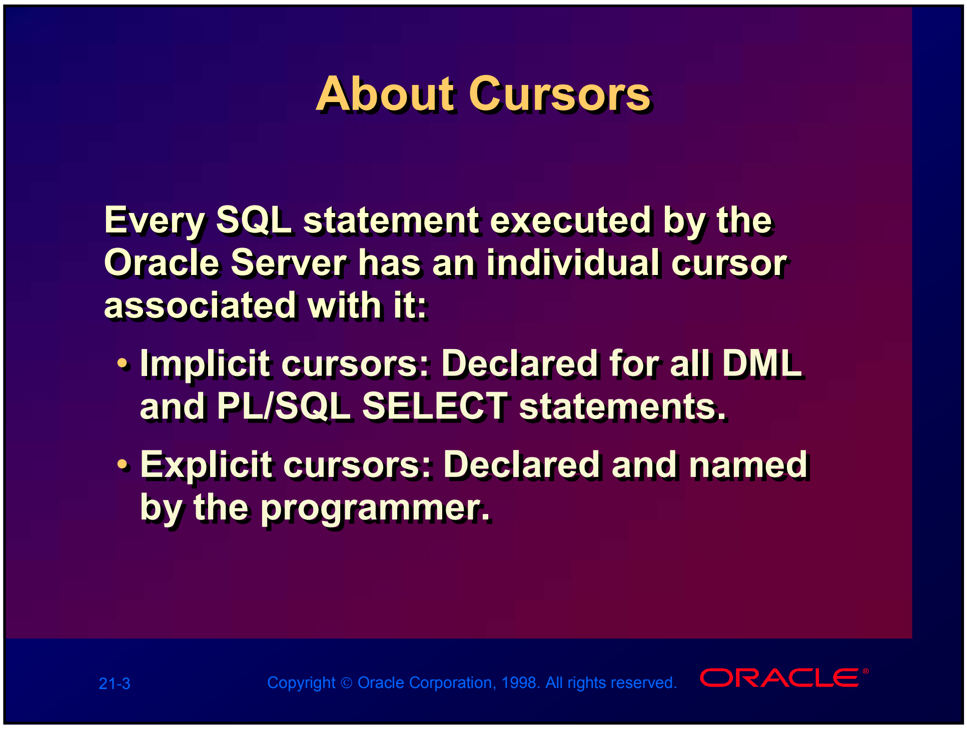
Writing Explicit Cursors



Objectives

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

- Distinguish between an implicit and an explicit cursor
- Use a PL/SQL record variable
- Write a Cursor FOR loop



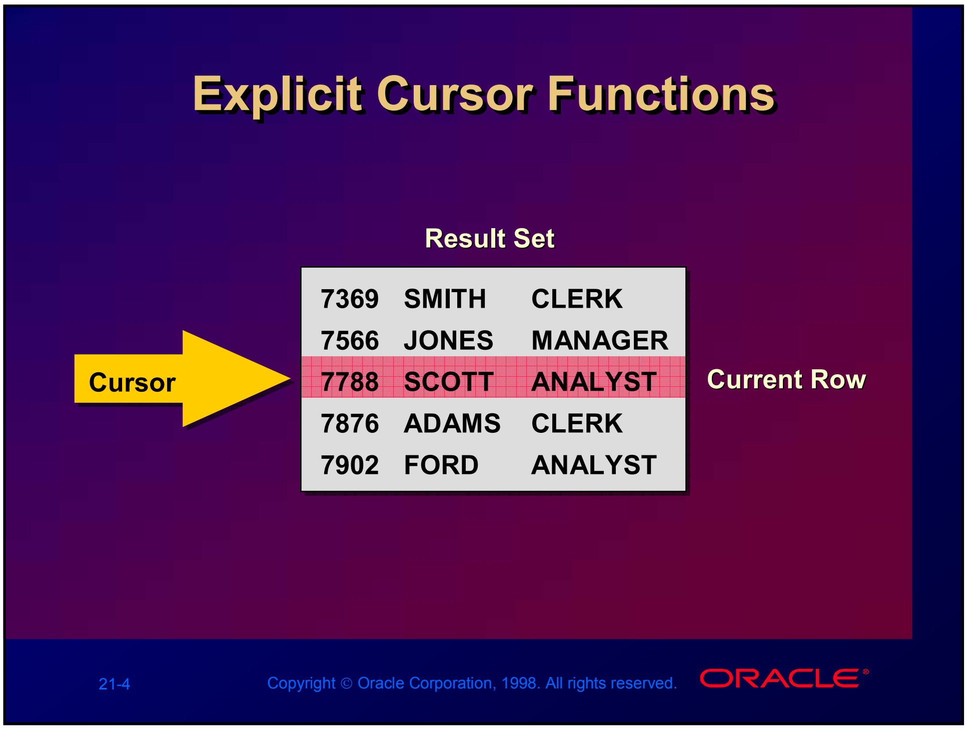
About Cursors

Every SQL statement executed by the Oracle Server has an individual cursor associated with it:

- Implicit cursors: Declared for all DML and PL/SQL SELECT statements.
- Explicit cursors: Declared and named by the programmer.

21-3

ORACLE®



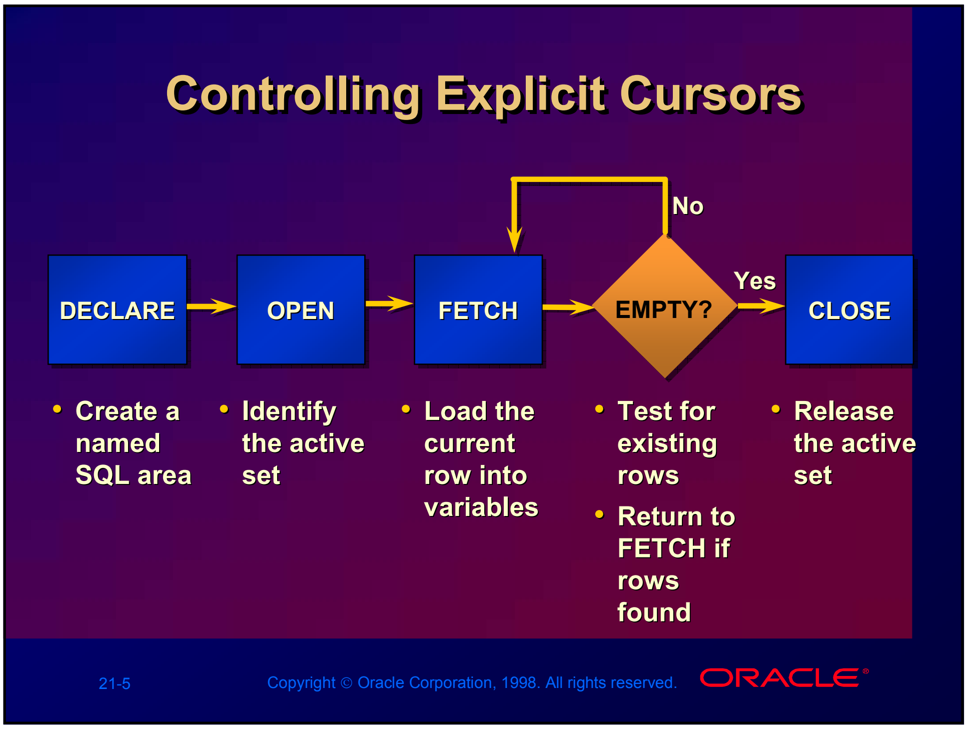
Explicit Cursor Functions

Result Set

7369	SMITH	CLERK
7566	JONES	MANAGER
7788	SCOTT	ANALYST
7876	ADAMS	CLERK
7902	FORD	ANALYST

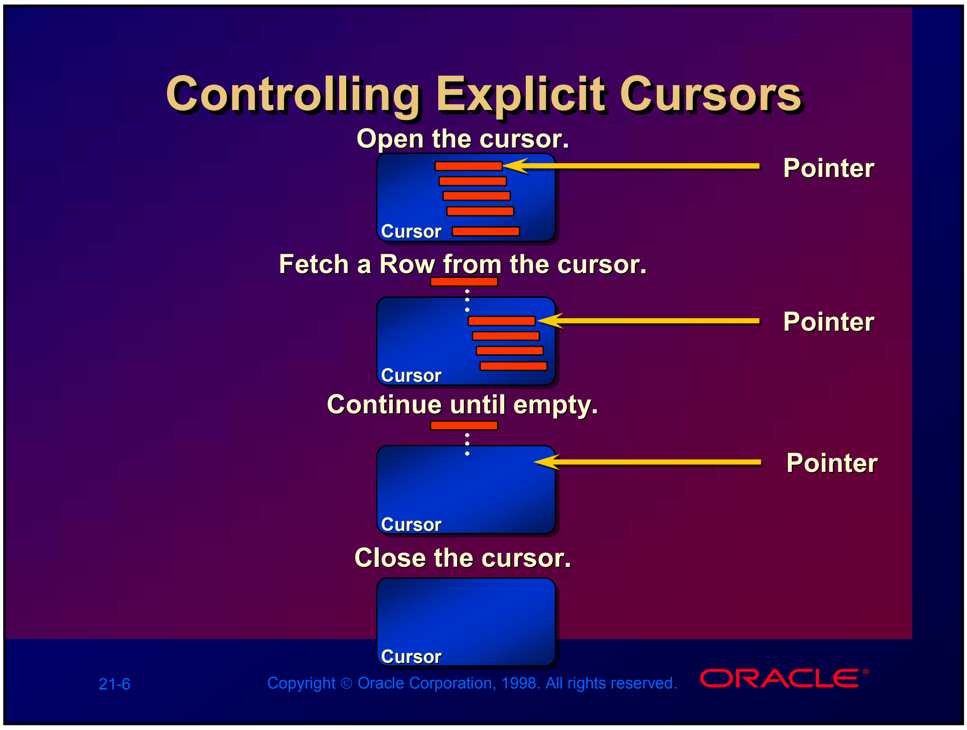
21-4

ORACLE®



Controlling Explicit Cursors

- Create a named SQL area
- Identify the active set
- Load the current row into variables
- Test for existing rows
- Return to FETCH if rows found
- Release the active set



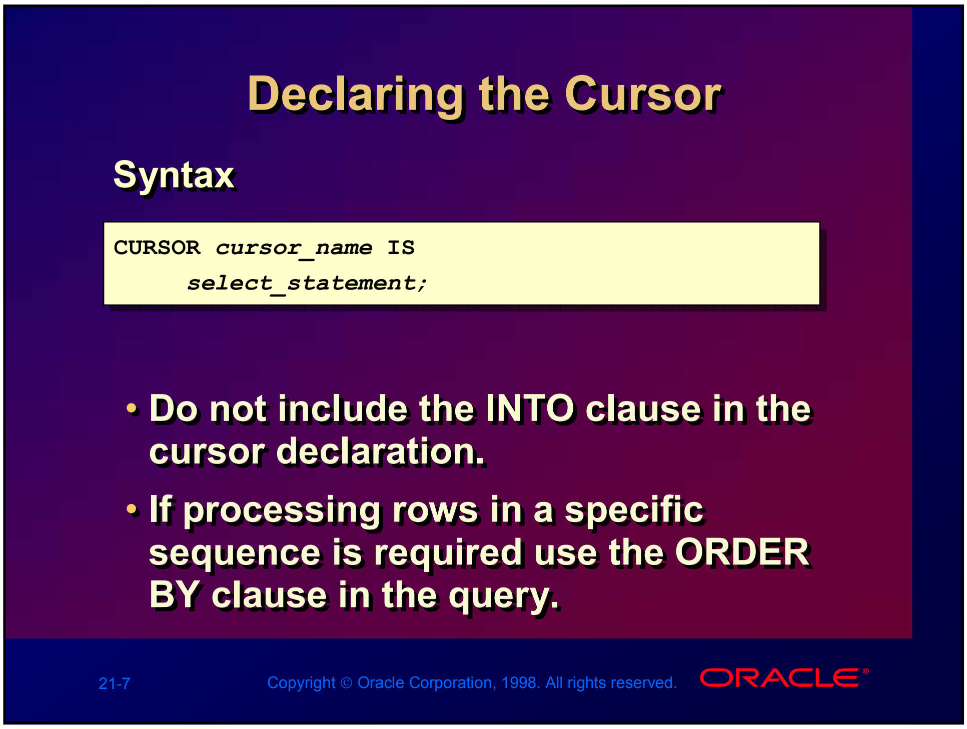
Controlling Explicit Cursors

Open the cursor.

Fetch a Row from the cursor.

Continue until empty.

Close the cursor.

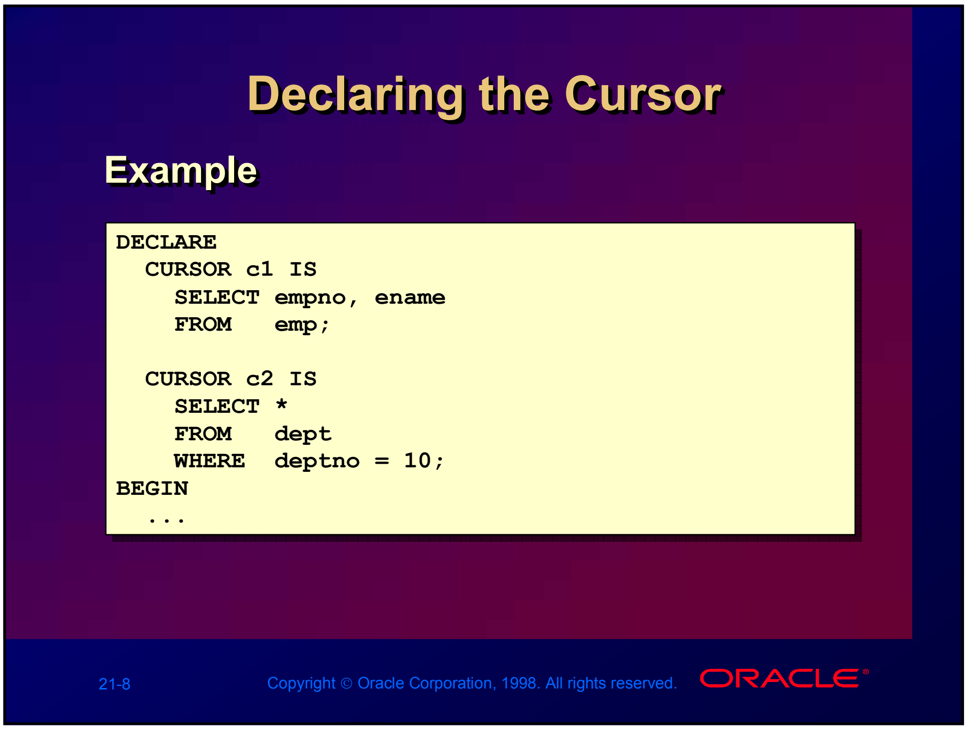


Declaring the Cursor

Syntax

```
CURSOR cursor_name IS
     select_statement;
```

- Do not include the INTO clause in the cursor declaration.
- If processing rows in a specific sequence is required use the ORDER BY clause in the query.

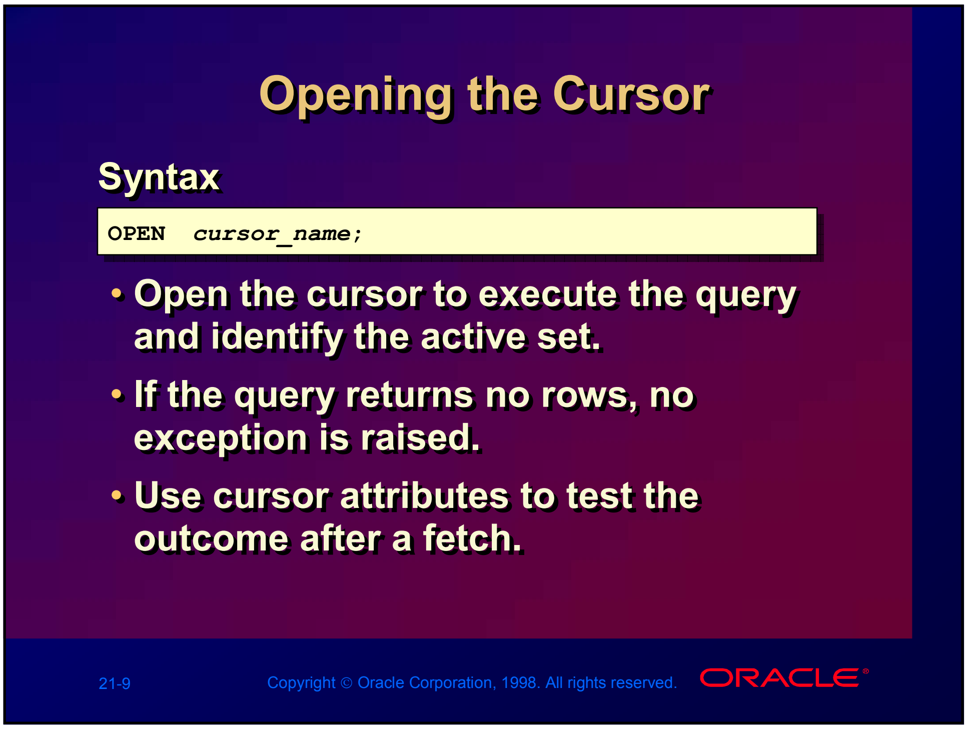


Declaring the Cursor

Example

```
DECLARE
  CURSOR c1 IS
    SELECT empno, ename
    FROM   emp;

  CURSOR c2 IS
    SELECT *
    FROM   dept
    WHERE  deptno = 10;
BEGIN
  ...
```



Opening the Cursor

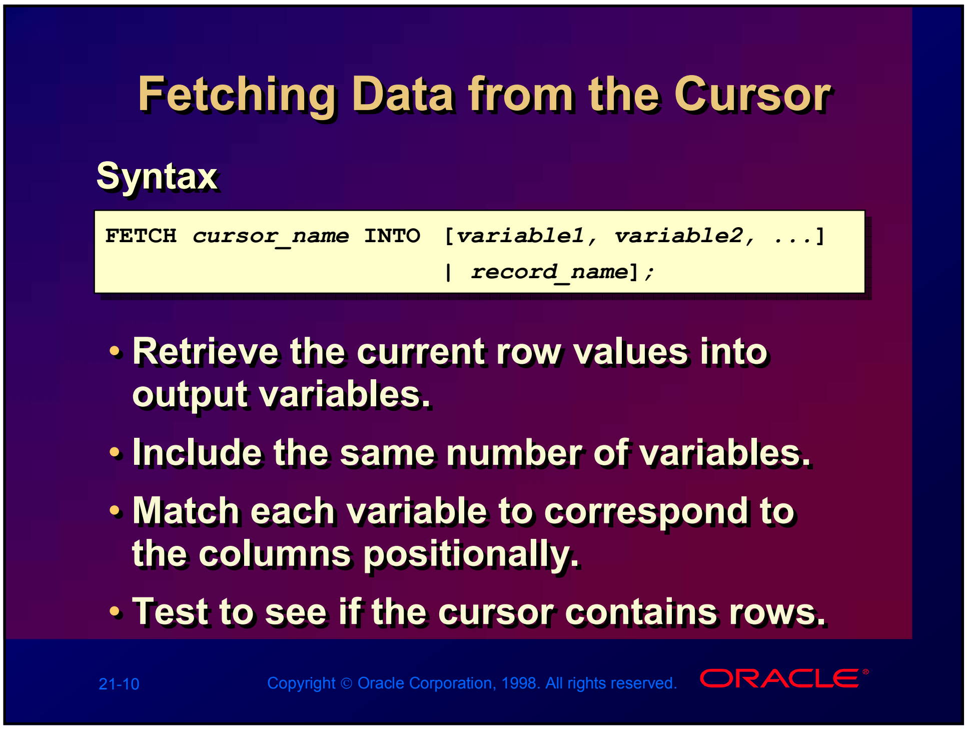
Syntax

```
OPEN  cursor_name;
```

- Open the cursor to execute the query and identify the active set.
- If the query returns no rows, no exception is raised.
- Use cursor attributes to test the outcome after a fetch.

21-9

ORACLE®



Fetching Data from the Cursor

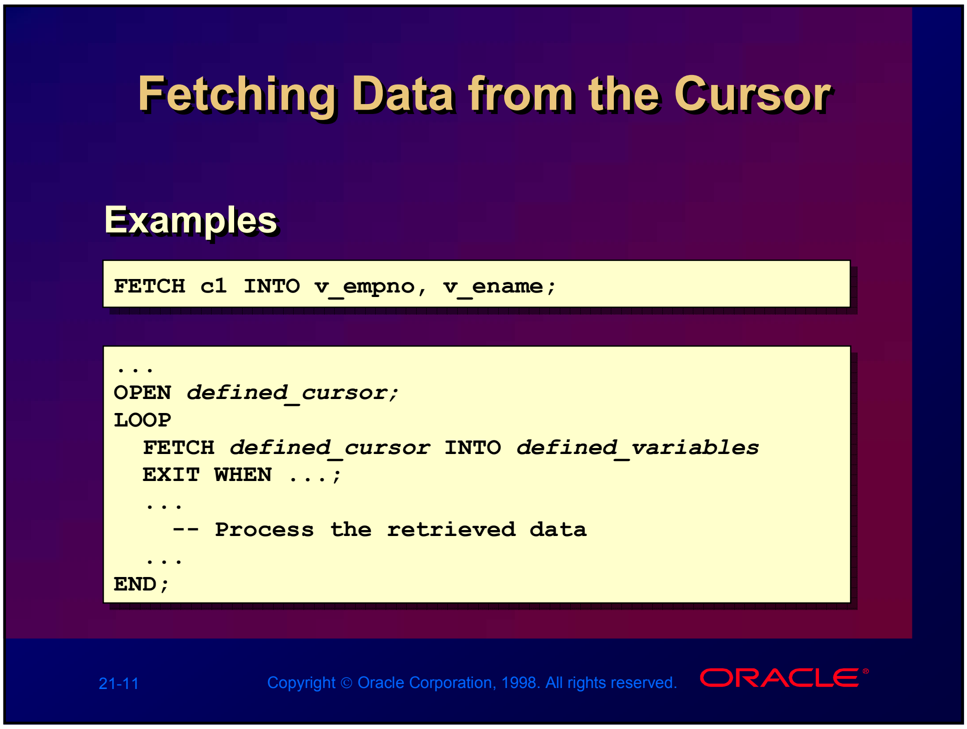
Syntax

```
FETCH cursor_name INTO  [variable1, variable2, ...]
                        | record_name];
```

- Retrieve the current row values into output variables.
- Include the same number of variables.
- Match each variable to correspond to the columns positionally.
- Test to see if the cursor contains rows.

21-10

ORACLE®

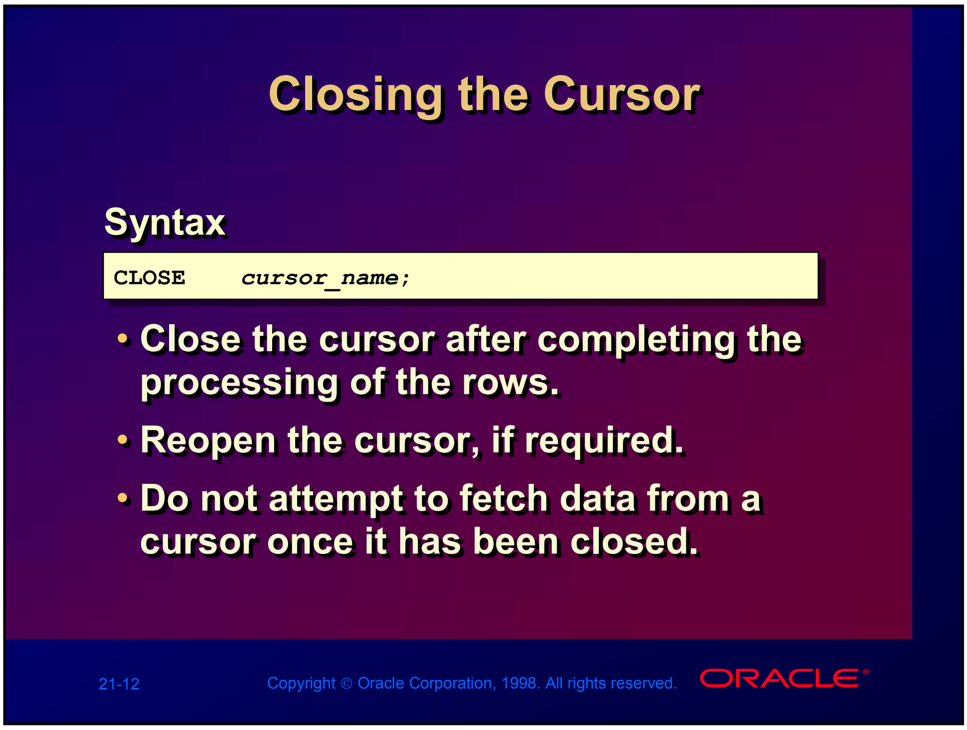


Fetching Data from the Cursor

Examples

```
FETCH c1 INTO v_empno, v_ename;
```

```
...
OPEN defined_cursor;
LOOP
  FETCH defined_cursor INTO defined_variables
  EXIT WHEN ...;
  ...
    -- Process the retrieved data
  ...
END;
```

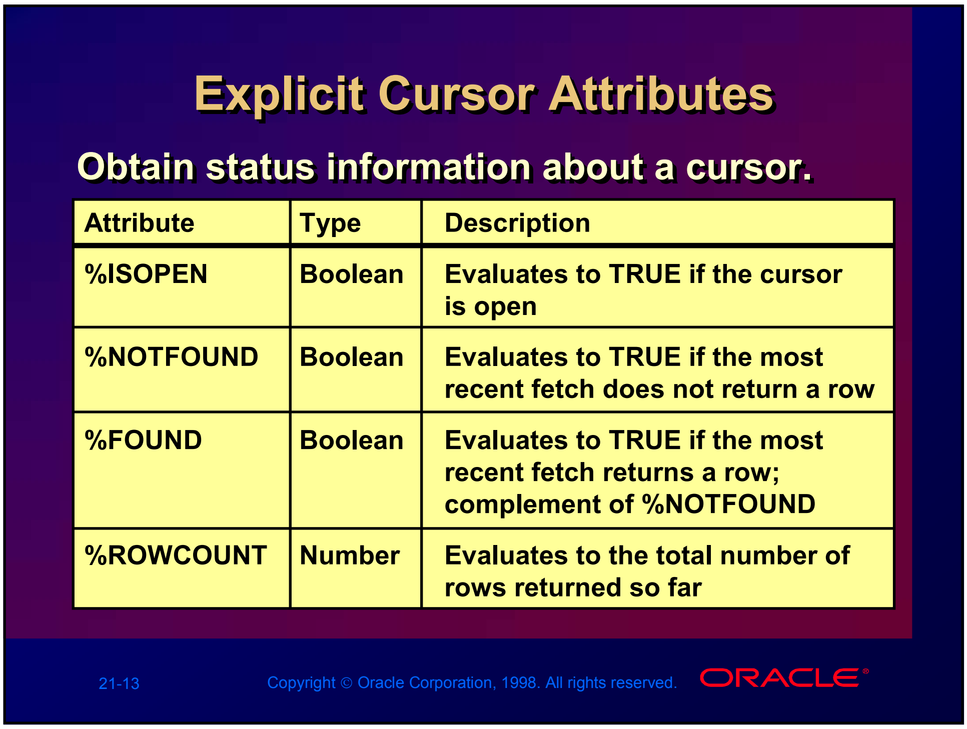


Closing the Cursor

Syntax

```
CLOSE    cursor_name;
```

- Close the cursor after completing the processing of the rows.
- Reopen the cursor, if required.
- Do not attempt to fetch data from a cursor once it has been closed.



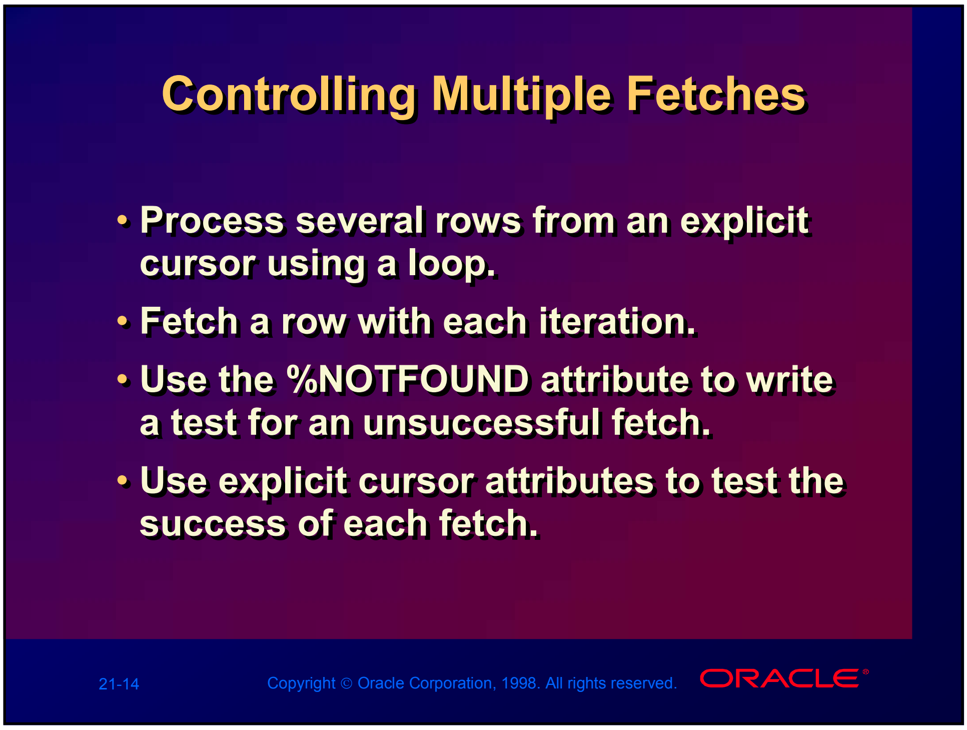
Explicit Cursor Attributes

Obtain status information about a cursor.

Attribute	Type	Description
%ISOPEN	Boolean	Evaluates to TRUE if the cursor is open
%NOTFOUND	Boolean	Evaluates to TRUE if the most recent fetch does not return a row
%FOUND	Boolean	Evaluates to TRUE if the most recent fetch returns a row; complement of %NOTFOUND
%ROWCOUNT	Number	Evaluates to the total number of rows returned so far

21-13

ORACLE®

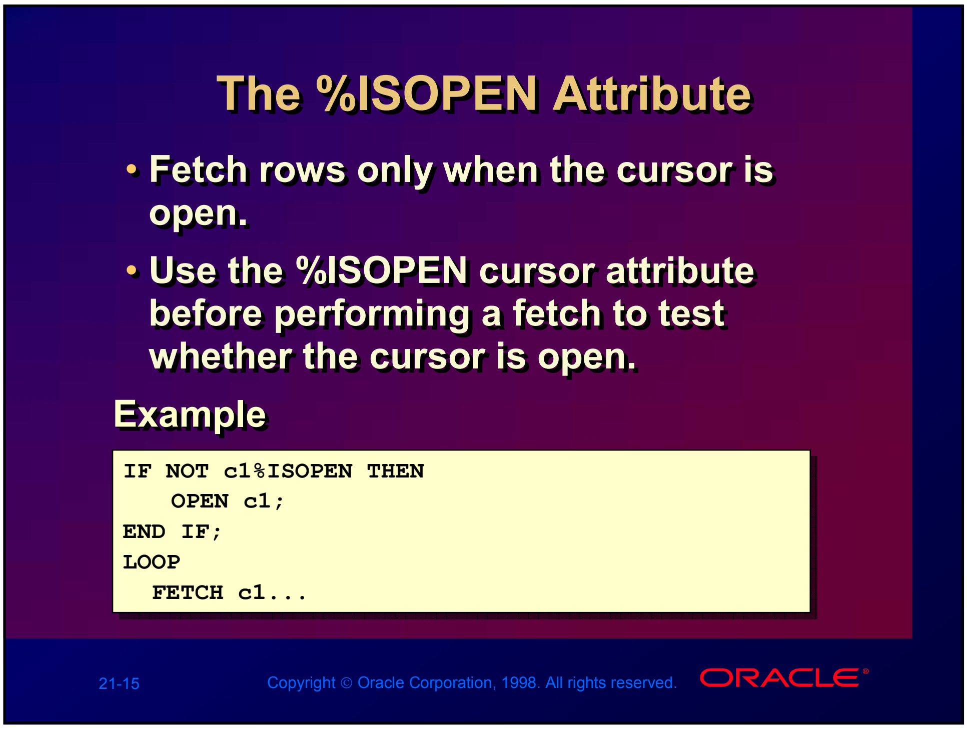


Controlling Multiple Fetches

- Process several rows from an explicit cursor using a loop.
- Fetch a row with each iteration.
- Use the %NOTFOUND attribute to write a test for an unsuccessful fetch.
- Use explicit cursor attributes to test the success of each fetch.

21-14

ORACLE®



The %ISOPEN Attribute

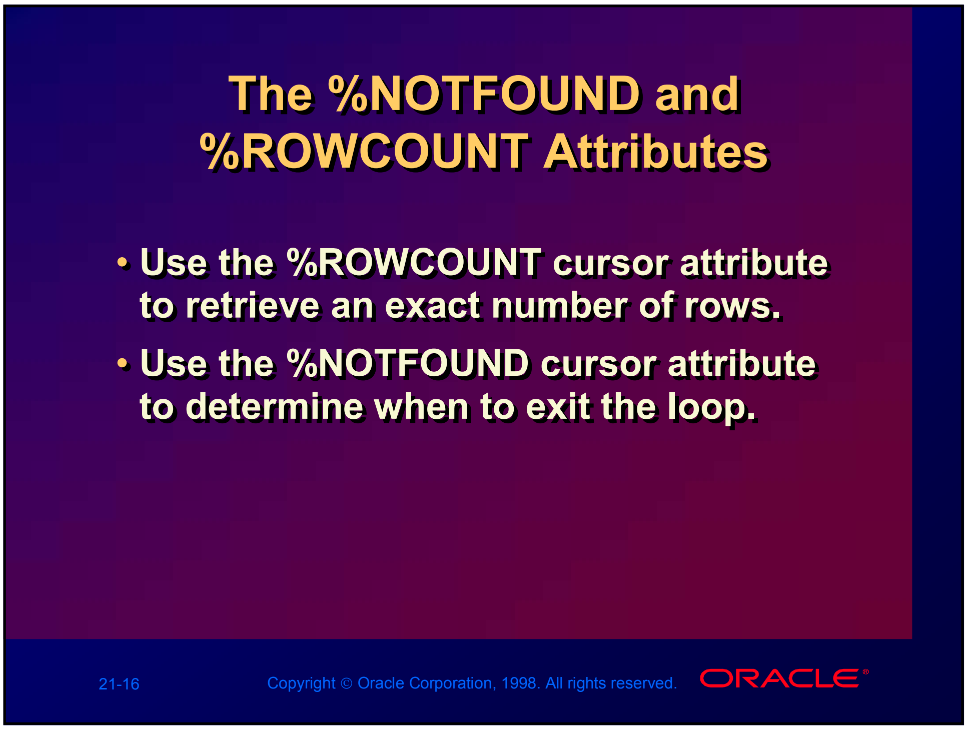
- Fetch rows only when the cursor is open.
- Use the %ISOPEN cursor attribute before performing a fetch to test whether the cursor is open.

Example

```
IF NOT c1%ISOPEN THEN
   OPEN c1;
END IF;
LOOP
  FETCH c1...
```

21-15

ORACLE®

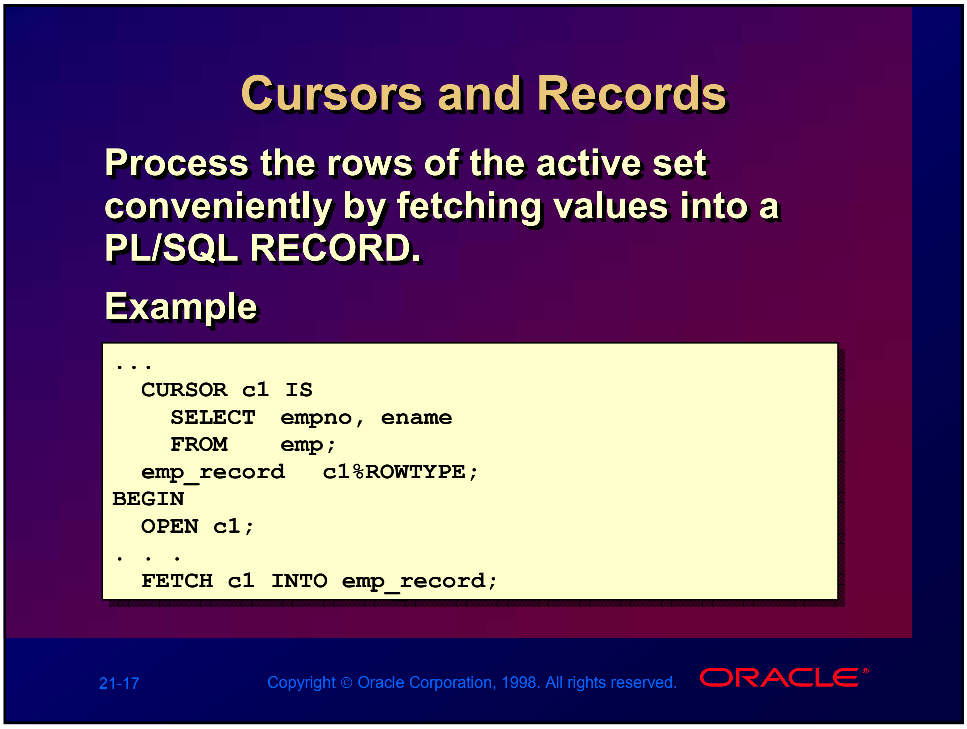


The %NOTFOUND and %ROWCOUNT Attributes

- Use the %ROWCOUNT cursor attribute to retrieve an exact number of rows.
- Use the %NOTFOUND cursor attribute to determine when to exit the loop.

21-16

ORACLE®



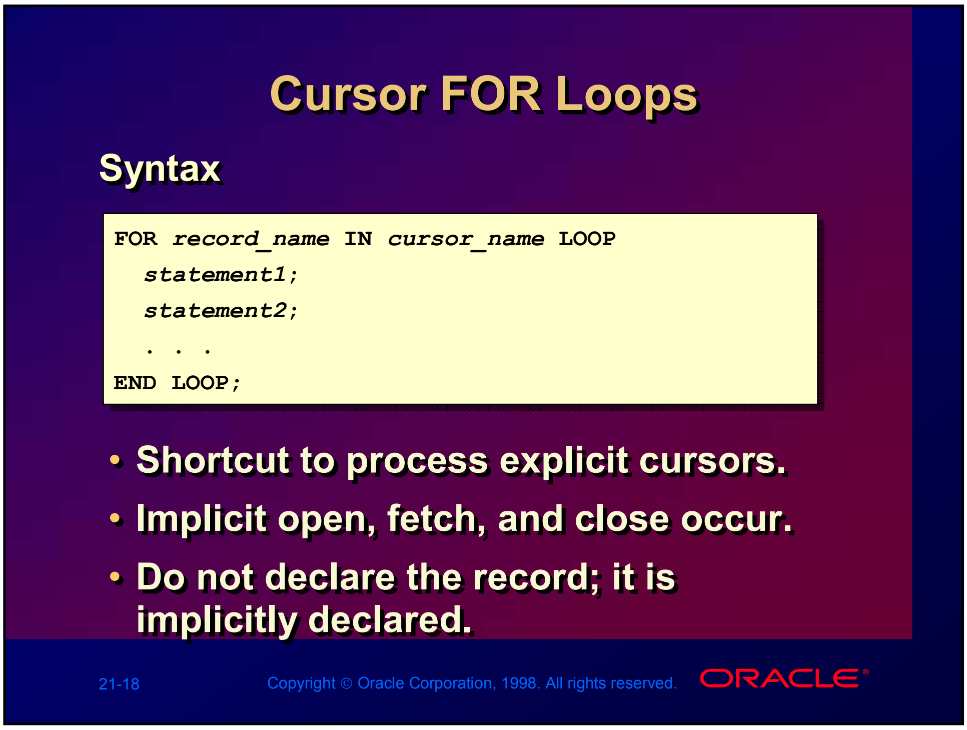
Cursors and Records

Process the rows of the active set conveniently by fetching values into a PL/SQL RECORD.

Example

```
...
  CURSOR c1 IS
    SELECT  empno, ename
    FROM    emp;
  emp_record   c1%ROWTYPE;
BEGIN
  OPEN c1;
. . .
  FETCH c1 INTO emp_record;
```

21-17



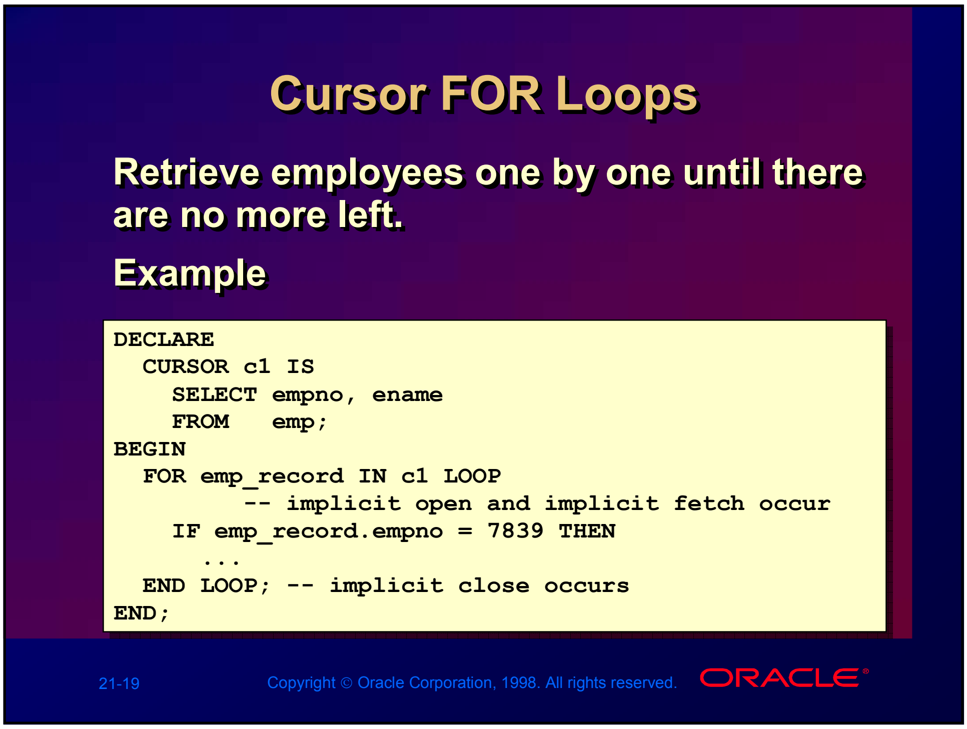
Cursor FOR Loops

Syntax

```
FOR record_name IN cursor_name LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

- Shortcut to process explicit cursors.
- Implicit open, fetch, and close occur.
- Do not declare the record; it is implicitly declared.

21-18



Cursor FOR Loops

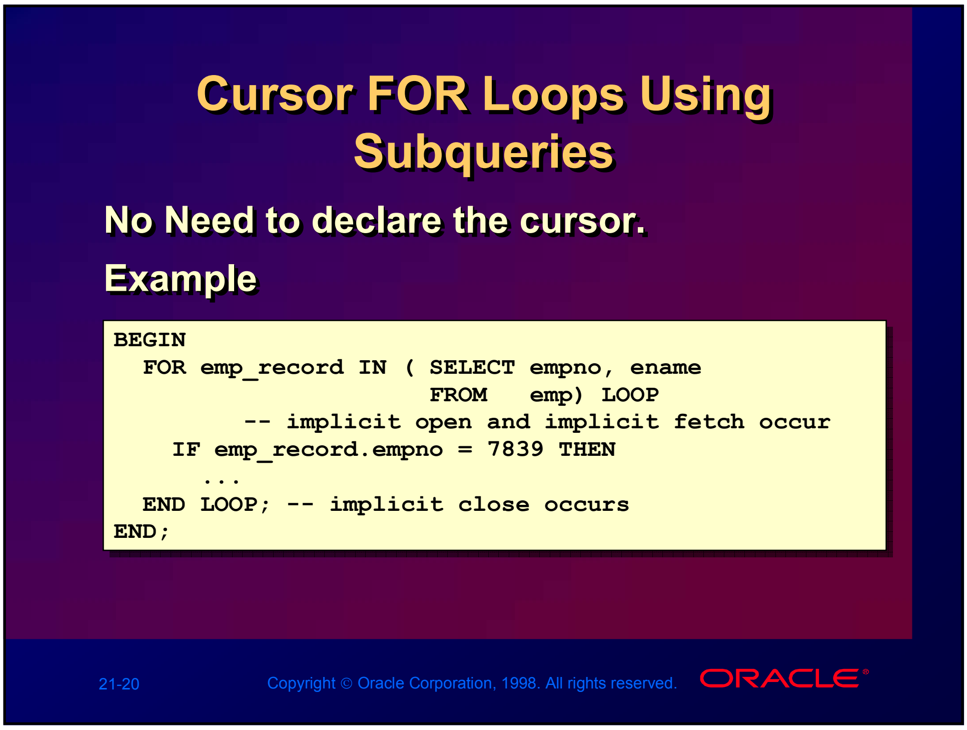
Retrieve employees one by one until there are no more left.

Example

```
DECLARE
  CURSOR c1 IS
    SELECT empno, ename
    FROM   emp;
BEGIN
  FOR emp_record IN c1 LOOP
         -- implicit open and implicit fetch occur
    IF emp_record.empno = 7839 THEN
      ...
  END LOOP; -- implicit close occurs
END;
```

21-19

ORACLE®



Cursor FOR Loops Using Subqueries

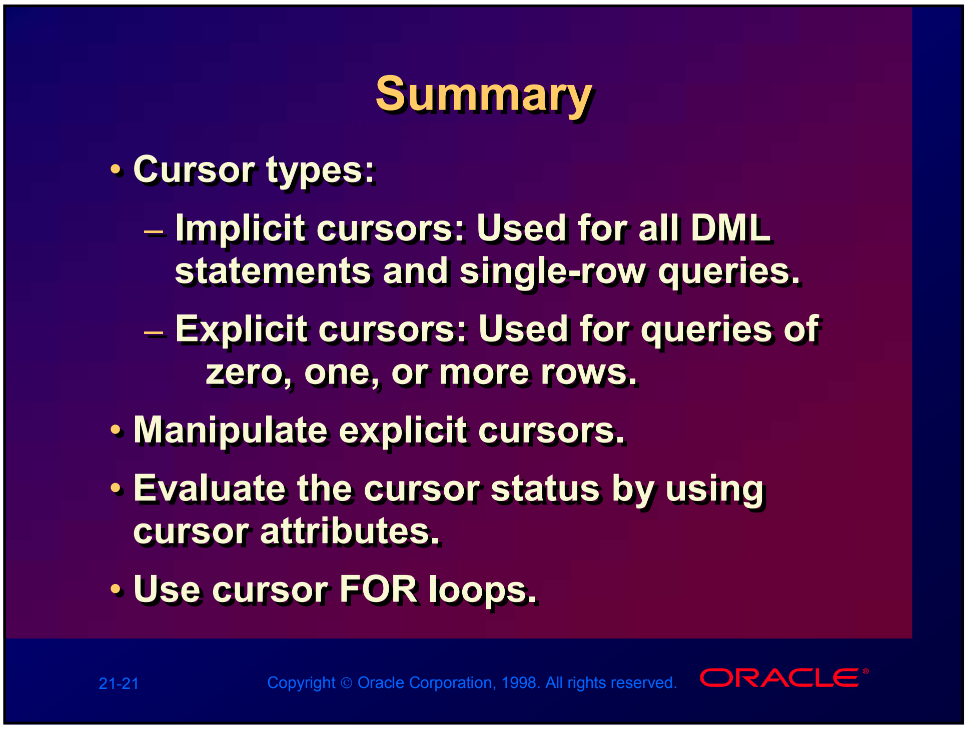
No Need to declare the cursor.

Example

```
BEGIN
  FOR emp_record IN ( SELECT empno, ename
                      FROM   emp) LOOP
         -- implicit open and implicit fetch occur
    IF emp_record.empno = 7839 THEN
      ...
  END LOOP; -- implicit close occurs
END;
```

21-20

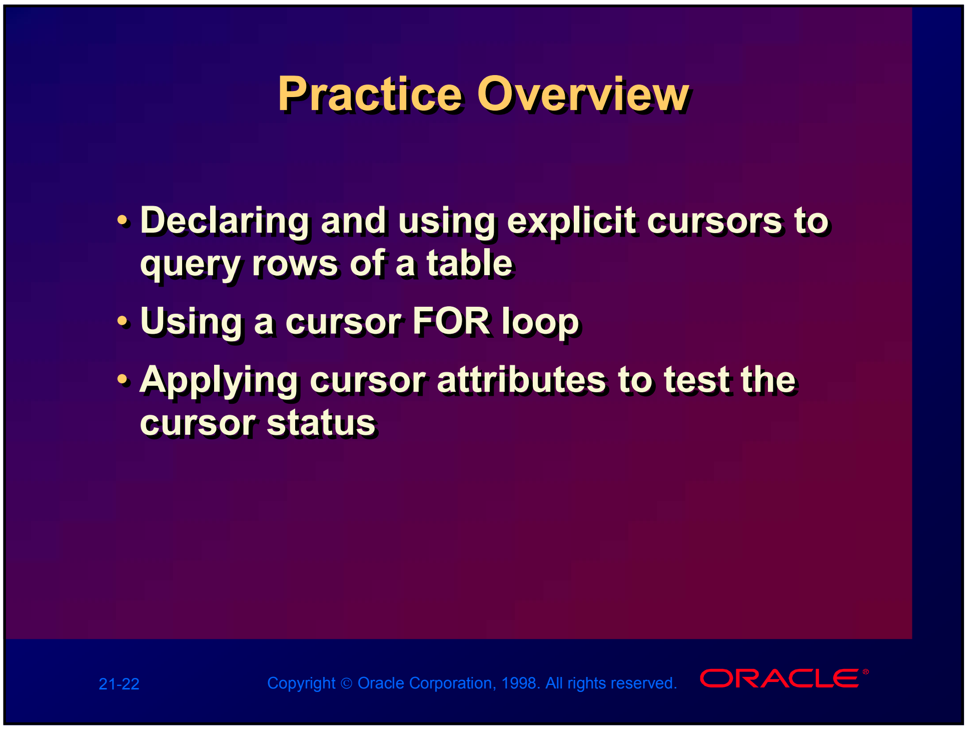
ORACLE®



Summary

- Cursor types:
  - Implicit cursors: Used for all DML statements and single-row queries.
  - Explicit cursors: Used for queries of zero, one, or more rows.
- Manipulate explicit cursors.
- Evaluate the cursor status by using cursor attributes.
- Use cursor FOR loops.

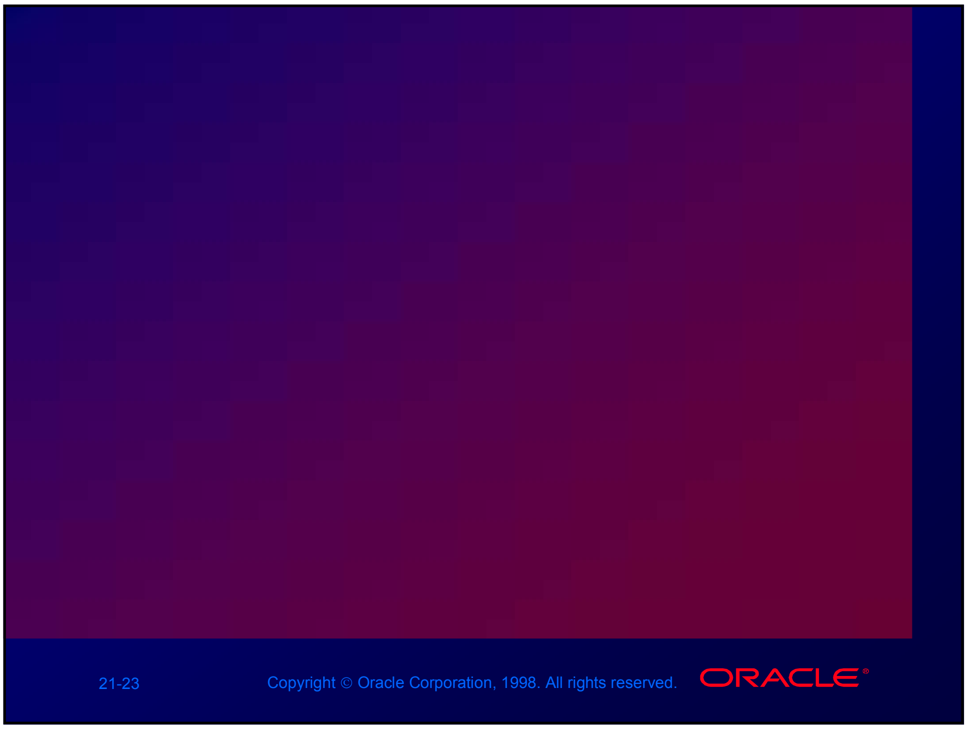
21-21



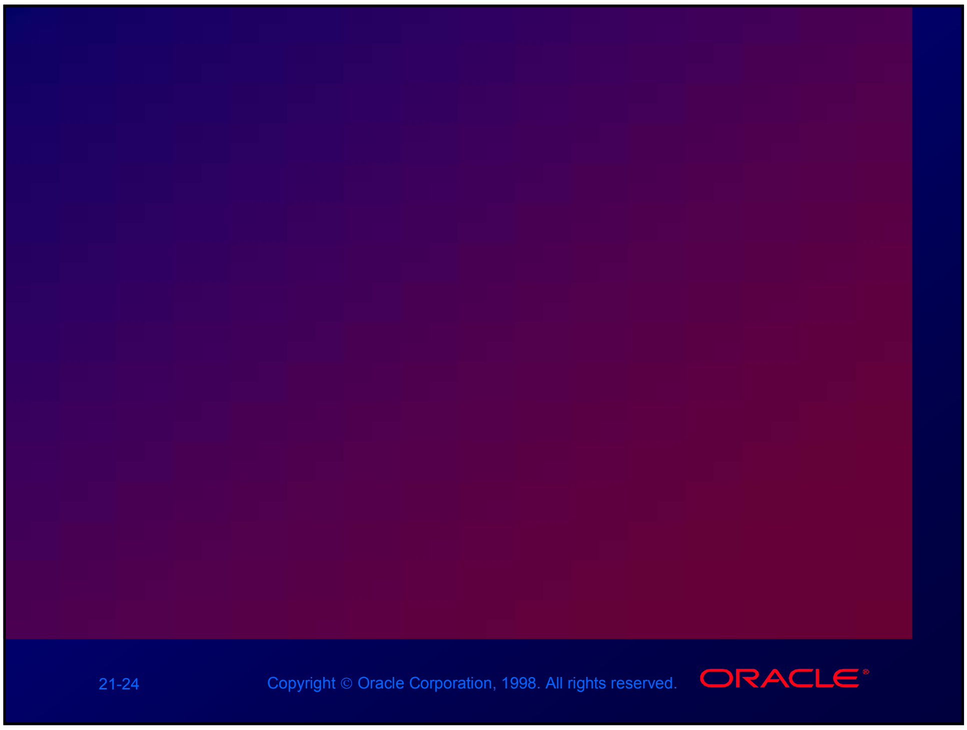
Practice Overview

- Declaring and using explicit cursors to query rows of a table
- Using a cursor FOR loop
- Applying cursor attributes to test the cursor status

21-22



21-23



21-24