

6

Writing Explicit Cursors

ORACLE

Objectives

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

- **Distinguish between an implicit and an explicit cursor**
- **Discuss when and why to use 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**

Explicit Cursor Functions

Table

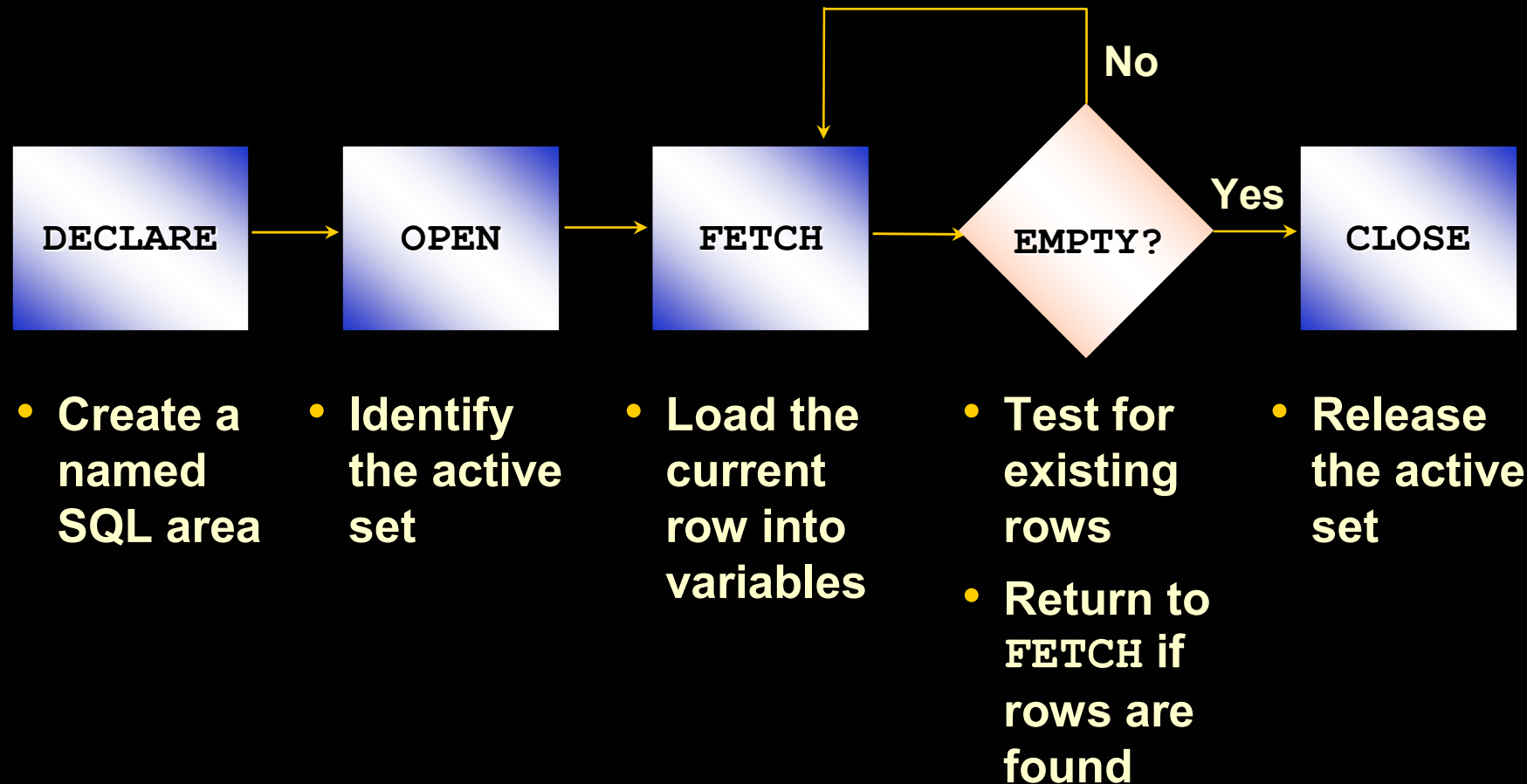
100	King	AD_PRES
101	Kochhar	AD_VP
102	De Haan	AD_VP
.	.	.
.	.	.
.	.	.
139	Seo	ST_CLERK
140	Patel	ST_CLERK
.	.	.

Active set



Cursor

Controlling Explicit Cursors



Controlling Explicit Cursors

1. **Open the cursor**
2. **Fetch a row**
3. **Close the Cursor**

1. Open the cursor.

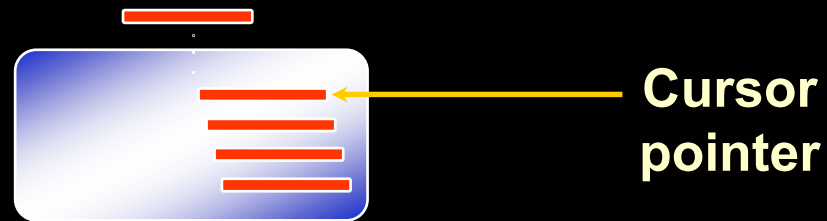


**Cursor
pointer**

Controlling Explicit Cursors

1. Open the cursor
2. **Fetch a row**
3. Close the Cursor

2. Fetch a row using the cursor.

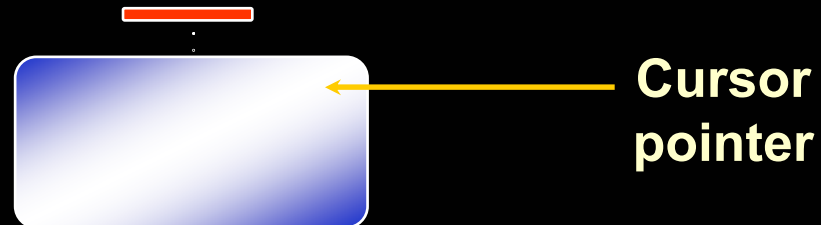


Continue until empty.

Controlling Explicit Cursors

1. Open the cursor
2. Fetch a row
3. Close the Cursor

3. 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 emp_cursor IS
    SELECT employee_id, last_name
    FROM   employees;

  CURSOR dept_cursor IS
    SELECT *
    FROM   departments
    WHERE  location_id = 170;
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.**

Fetching Data from the Cursor

Syntax:

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

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

Fetching Data from the Cursor

Example:

```
LOOP
  FETCH emp_cursor INTO v_empno,v_ename;
  EXIT WHEN ...;
  ...
  -- Process the retrieved data
  ...
END LOOP;
```

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 after 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

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 emp_cursor%ISOPEN THEN
    OPEN emp_cursor;
END IF;
LOOP
    FETCH emp_cursor...
```


Controlling Multiple Fetches

- **Process several rows from an explicit cursor using a loop.**
- **Fetch a row with each iteration.**
- **Use explicit cursor attributes to test the success of each fetch.**

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.

Example

```
DECLARE
    v_empno  employees.employee_id%TYPE;
    v_ename  employees.last_name%TYPE;
    CURSOR emp_cursor IS
        SELECT employee_id, last_name
        FROM   employees;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_empno, v_ename;
        EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)
                               || '   ' || v_ename);
    END LOOP;
    CLOSE emp_cursor;
END ;
```

Cursors and Records

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

```
DECLARE
  CURSOR emp_cursor IS
    SELECT  employee_id, last_name
    FROM    employees;
  emp_record  emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
    ...
```

emp_record	
employee_id	last_name

100	King
-----	------

Cursor FOR Loops

Syntax:

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

- The cursor FOR loop is a shortcut to process explicit cursors.
- Implicit open, fetch, exit, and close occur.
- The record is implicitly declared.

Cursor FOR Loops

Print a list of the employees who work for the sales department.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT last_name, department_id
    FROM   employees;
BEGIN
  FOR emp_record IN emp_cursor LOOP
    -- implicit open and implicit fetch occur
    IF emp_record.department_id = 80 THEN
      ...
    END LOOP; -- implicit close occurs
END;
/
```

Cursor FOR Loops Using Subqueries

No need to declare the cursor.

Example:

```
BEGIN
  FOR emp_record IN (SELECT last_name, department_id
                      FROM   employees) LOOP
    -- implicit open and implicit fetch occur
    IF emp_record.department_id = 80 THEN
      ...
    END LOOP; -- implicit close occurs
END;
```


Summary

In this lesson you should have learned to:

- **Distinguish 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**

Practice 6 Overview

This practice covers the following topics:

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

Topik Latihan Bab 6

Meliputi topic berikut :

- Mendeklarasikan dan menggunakan explicit cursors untuk menampilkan baris data pada tabel
- Menggunakan cursor FOR loop
- Menggunakan atribut cursor untuk menguji status cursor

Latihan 6

1. Jalankan skrip berikut ini untuk membuat table baru dengan nama TOP_DOGS yang menyimpan gaji (salary) dari pegawai

```
CREATE TABLE top_dogs  
( salary      NUMBER(8,2));
```

2. Buat program PL/SQL untuk menyimpan data n pegawai yang memiliki gaji tertinggi. Misal n diinputkan nilainya = 5, berarti menampilkan 5 pegawai yang memiliki gaji tertinggi.

Keterangan :

- Gunakan perintah DEFINE untuk menyimpan nilai n , lewatkan nilainya pada blok PL/SQL block melalui variable substitusi &SQL*Plus.
- Jika dua pegawai memiliki gaji yang sama, maka nilai gaji cukup disimpan sekali saja.
- Simpan nilai gaji dalam table TOP_DOGS.
- Uji program dengan memasukkan nilai n yang tidak umum misal : $n = 0$ atau n lebih dari jumlah pegawai dalam table EMPLOYEES. Kosongkan isi table sebelum diisi dengan n ilai gaji tertinggi.

3. Buat program PL/SQL untuk menampilkan data pegawai (last name, salary, dan MANAGER ID) yang bekerja di departemen tertentu, tentukan nomer departemennya, letakkan dalam variable substitusi.

Jika gaji pegawai kurang dari 5000 dan jika manager ID = 101 atau 124, tampilkan pesan <<last_name>> Gaji perlu dinaikkan. Jika tidak, tampilkan pesan <<last_name>> Gaji tidak perlu naik.

Keterangan :

Ketik : SET ECHO OFF supaya tidak perlu menampilkan kode PL/SQL setiap kali menjalankan program