Database Programming with PL/SQL

Using Explicit Cursor Attributes

Objectives

This lesson covers the following objectives:

- Define a record structure using the `%ROWTYPE` attribute
- Create PL/SQL code to process the rows of an active set using record types in cursors
- Retrieve information about the state of an explicit cursor using cursor attributes

Purpose

One of the reasons to use explicit cursors is that they give you greater programmatic control when handling your data. This lesson discusses techniques for using explicit cursors more effectively.

- Cursor records enable you to declare a single variable for all the selected columns in a cursor.
- Cursor attributes enable you to retrieve information about the state of your explicit cursor.

Cursors and Records

The cursor in this example is based on a SELECT statement that retrieves only two columns of each table row. What if it retrieved six columns .. or seven, or eight, or twenty?

```
DECLARE
  v_emp_id          employees.employee_id%TYPE;
  v_last_name       employees.last_name%TYPE;
  CURSOR emp_cursor IS
    SELECT employee_id, last_name
      FROM employees
      WHERE  department_id = 30;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor
    INTO v_emp_id, v_last_name;
  ...
```

Cursors and Records (cont.)

This cursor retrieves whole rows of EMPLOYEES. Messy and long-winded, isn't it?

```
DECLARE
  v_emp_id         employees.employee_id%TYPE;
  v_first_name     employees.first_name%TYPE;
  v_last_name      employees.last_name%TYPE;
  ...
  v_department_id employees.department_id%TYPE;
  CURSOR emp_cursor IS
    SELECT * FROM employees
      WHERE  department_id = 30;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor
     INTO v_emp_id, v_first_name, v_last_name ...
          v_department_id;
  ...
```

Cursors and Records (cont.)

Compare the following snippets of code. What differences do you see?

```
DECLARE
  v_emp_id         ...;
  v_first_name     ...;
  ...
  v_department_id ...:
  CURSOR emp_cursor IS
    SELECT * FROM employees
    WHERE  department_id =30;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor
    INTO v_emp_id, v_first_name,
    ... v_department_id;
  ...
```

```
DECLARE
  CURSOR emp_cursor IS
    SELECT * FROM employees
    WHERE  department_id = 30;
  v_emp_record
    emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor
    INTO v_emp_record;
  ...
```

Cursors and Records (cont.)

The code on the right uses %ROWTYPE to declare a record structure based on the cursor. A record is a composite data type in PL/SQL.

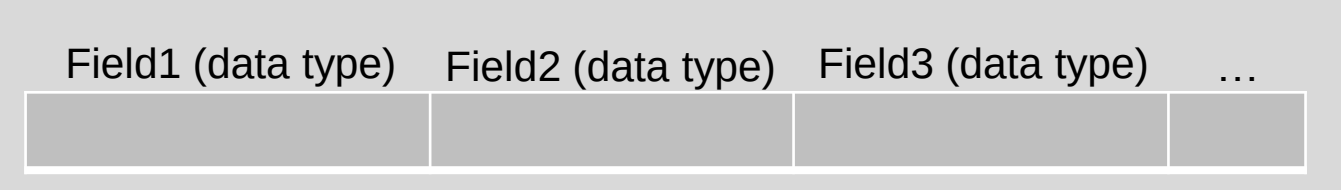
Variables

```
DECLARE
  v_emp_id          ...;
  v_first_name      ...;
  ...
  v_department_id ...:
  CURSOR emp_cursor IS
   SELECT * FROM employees
   WHERE  department_id = 30;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor
    INTO v_emp_id, v_first_name,
    ... v_department_id;
  ...
```

Records

```
DECLARE
  CURSOR emp_cursor IS
   SELECT * FROM employees
   WHERE  department_id = 30;
  v_emp_record
     emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor
    INTO v_emp_record;
  ...
```

Structure of a PL/SQL Record



A record is a composite data type, consisting of a number of fields each with their own name and data type.

You reference each field by dot-prefixing its field-name with the record-name.

`%ROWTYPE` declares a record with the same fields as the cursor on which it is based.

Structure of `cursor_name%ROWTYPE`

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name, salary FROM employees
      WHERE  department_id = 30;
  v_emp_record emp_cursor%ROWTYPE;
...
```

v_emp_record.employee_id	v_emp_record.last_name	v_emp_record.salary	…
100	King	24000	

Cursors and %ROWTYPE

%ROWTYPE is convenient for processing the rows of the active set because you can simply fetch into the record.

```
DECLARE
  CURSOR emp_cursor IS
    SELECT * FROM employees
    WHERE  department_id = 30;
  v_emp_record   emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO v_emp_record;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_emp_record.employee_id|| ' - '
||v_emp_record.last_name);
    END LOOP;
  CLOSE emp_cursor;
END;
```

Cursors and %ROWTYPE: Another Example

```
DECLARE
  CURSOR emp_dept_cursor IS
    SELECT first_name, last_name, department_name
      FROM employees e, departments d
      WHERE  e.department_id = d.department_id;
  v_emp_dept_record       emp_dept_cursor%ROWTYPE;
BEGIN
  OPEN emp_dept_cursor;
  LOOP
    FETCH emp_dept_cursor INTO v_emp_dept_record;
    EXIT WHEN emp_dept_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_emp_dept_record.first_name
      ||' - '||v_emp_dept_record.last_name
      ||' - '||v_emp_dept_record.department_name);
  END LOOP;
  CLOSE emp_dept_cursor;
END;
```

Explicit Cursor Attributes

As with implicit cursors, there are several attributes for obtaining status information about an explicit cursor. When appended to the cursor variable name, these attributes return useful information about the execution of a cursor manipulation statement.

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

%ISOPEN Attribute

You can fetch rows only when the cursor is open. Use the %ISOPEN cursor attribute before performing a fetch to test whether the cursor is open. %ISOPEN returns the status of the cursor: TRUE if open and FALSE if not.

Example:

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

%ROWCOUNT and %NOTFOUND Attributes

Usually the %ROWCOUNT and %NOTFOUND attributes are used in a loop to determine when to exit the loop.

Use the %ROWCOUNT cursor attribute for the following:

- To process an exact number of rows
- To count the number of rows fetched so far in a loop and/or determine when to exit the loop

%ROWCOUNT and %NOTFOUND Attributes (cont.)

Use the %NOTFOUND cursor attribute for the following:

- To determine whether the query found any rows matching your criteria
- To determine when to exit the loop

Example of %ROWCOUNT and %NOTFOUND

This example shows how you can use %ROWCOUNT and %NOTFOUND attributes for exit conditions in a loop.

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

Explicit Cursor Attributes in SQL Statements

You cannot use an explicit cursor attribute directly in an SQL statement. The following code returns an error:

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, salary FROM employees
    ORDER BY SALARY DESC;
  v_emp_record  emp_cursor%ROWTYPE;
  v_count       NUMBER;
BEGIN
  OPEN emp_cursor;
  LOOP
   FETCH emp_cursor INTO v_emp_record;
   EXIT WHEN emp_cursor%NOTFOUND;
   INSERT INTO top_paid_emps
    (employee_id, rank, salary)
   VALUES
    (v_emp_record.employee_id, emp_cursor%ROWCOUNT,
     v_emp_record.salary);
...
```

Terminology

Key terms used in this lesson included:

- %ISOPEN
- %NOTFOUND
- Record
- %ROWCOUNT
- %ROWTYPE

Summary

In this lesson, you should have learned how to:

- Define a record structure using the `%ROWTYPE` attribute
- Create PL/SQL code to process the rows of an active set using record types in cursors
- Retrieve information about the state of an explicit cursor using cursor attributes