ARDBMS UNIT 4

PL/SQL Cursors

PL/SQL uses **cursors** to retrieve and process multiple rows from a query. There are two types of cursors: **Implicit Cursors** and **Explicit Cursors**.

1. Cursors in PL/SQL

1.1 Implicit Cursors

PL/SQL automatically creates an implicit cursor whenever a SELECT statement (that returns a single row) or DML (INSERT, UPDATE, DELETE) is executed.

Example:

```pl
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM employees;
    DBMS_OUTPUT.PUT_LINE('Total Employees: ' || v_count);
END;
```

1.2 Explicit Cursors

Explicit cursors are created when a query returns multiple rows, and you want to process them row by row.

Steps to use Explicit Cursor:

1. **Declare** the cursor using CURSOR cursor_name IS query;
2. **Open** the cursor using OPEN cursor_name;
3. **Fetch** the data row by row using FETCH cursor_name INTO variable;
4. **Close** the cursor using CLOSE cursor_name;

```
DECLARE
    CURSOR emp_cursor IS SELECT employee_id, first_name FROM employees;
    v_id employees.employee_id%TYPE;
    v_name employees.first_name%TYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_id, v_name;
        EXIT WHEN emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('ID: ' || v_id || ' Name: ' || v_name);
    END LOOP;
    CLOSE emp_cursor;
END;
```

2. Cursor Attributes

PL/SQL provides built-in cursor attributes to check the state of the cursor.

Attribute	Description
%FOUND	Returns TRUE if the last fetch returned a row
%NOTFOUND	Returns TRUE if the last fetch did not return a row
%ROWCOUNT	Returns the number of rows fetched so far
%ISOPEN	Returns TRUE if the cursor is open

3. Cursor FOR Loop

The FOR loop simplifies the use of cursors by automatically opening, fetching, and closing them.

```
BEGIN
   FOR emp_record IN (SELECT employee_id, first_name FROM employees) LOOP
      DBMS_OUTPUT.PUT_LINE('ID: ' || emp_record.employee_id || ' Name: ' ||
emp_record.first_name);
   END LOOP;
END;
```

4. SELECT...FOR UPDATE & WHERE CURRENT OF Clause

The SELECT...FOR UPDATE locks the selected rows to prevent other transactions from modifying them until the transaction is committed or rolled back.

The WHERE CURRENT OF clause is used to update or delete the row that was last fetched.

```
DECLARE
   CURSOR emp_cursor IS SELECT employee_id, salary FROM employees FOR UPDATE;
   v_id employees.employee_id%TYPE;
BEGIN
   OPEN emp_cursor;
   LOOP
      FETCH emp_cursor INTO v_id;
      EXIT WHEN emp_cursor%NOTFOUND;
      UPDATE employees SET salary = salary * 1.10 WHERE CURRENT OF emp_cursor;
   END LOOP;
   CLOSE emp_cursor;
END;
```

5. Cursor with Parameters

Parameterized cursors allow passing values dynamically at runtime.

```
DECLARE
   CURSOR emp_cursor (dept_id NUMBER) IS SELECT employee_id, first_name FROM
employees WHERE department_id = dept_id;
   v_id employees.employee_id%TYPE;
   v_name employees.first_name%TYPE;
BEGIN
   OPEN emp_cursor(10);
   LOOP
      FETCH emp_cursor INTO v_id, v_name;
      EXIT WHEN emp_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('ID: ' || v_id || ' Name: ' || v_name);
   END LOOP;
   CLOSE emp_cursor;
END;
```

6. Cursor Variables (REF CURSOR)

Cursor variables (REF CURSOR) allow dynamic query execution and are useful in procedures/functions.

```
DECLARE
   TYPE emp_cursor_type IS REF CURSOR;
   emp_cursor emp_cursor_type;
   v_id employees.employee_id%TYPE;
   v_name employees.first_name%TYPE;
BEGIN
   OPEN emp_cursor FOR SELECT employee_id, first_name FROM employees;
   LOOP
      FETCH emp_cursor INTO v_id, v_name;
      EXIT WHEN emp_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('ID: ' || v_id || ' Name: ' || v_name);
   END LOOP;
   CLOSE emp_cursor;
END;
```

7. PL/SQL Records

A **record** is a composite datatype that groups related data into a single unit.

7.1 Types of Records

1. Table-based Records

Automatically inherit the structure of a table.

```
DECLARE
   emp_rec employees%ROWTYPE;
BEGIN
   SELECT * INTO emp_rec FROM employees WHERE employee_id = 101;
   DBMS_OUTPUT.PUT_LINE('Name: ' || emp_rec.first_name || ' Salary: ' ||
emp_rec.salary);
END;
```

2. Cursor-based Records

Takes structure from an explicit cursor.

```
DECLARE
   CURSOR emp_cursor IS SELECT employee_id, first_name FROM employees;
   emp_rec emp_cursor%ROWTYPE;
BEGIN
   OPEN emp_cursor;
   FETCH emp_cursor INTO emp_rec;
   CLOSE emp_cursor;
   DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_rec.first_name);
END;
```

3. User-defined Records

Manually defined using TYPE.

```
DECLARE
   TYPE emp_type IS RECORD (id NUMBER, name VARCHAR2(50));
   emp_rec emp_type;
BEGIN
   emp_rec.id := 101;
   emp_rec.name := 'John Doe';
   DBMS_OUTPUT.PUT_LINE('ID: ' || emp_rec.id || ' Name: ' || emp_rec.name);
END;
```