PL/SQL Cursors (Implicit, Explicit, For Loop, Parameterized Cursor)

#### **What is Cursor?**

Cursor is the work area which Oracle reserves for internal processing of SQL statements. This work area is private for oracles reserved are called cursor.

Cursor area also saying session cursor. because session cursor store information until the session end. Both way you can manage session cursor either implicit cursor or explicit cursor.

Using procedural statement you can get any information using session attribute.

#### **How to Use Cursor**

In PL/SQL block SELECT statement can not return more than one row at a time. So Cursor use to some group of rows (more than one row) for implementing certain logic to get all the group of records.

#### **Classification of CURSORS**

Cursors can be classified as:

1. Implicit Cursor or Internal Cursor : Manage for Oracle itself or internal process itself.
2. Explicit Cursor or User-defined Cursor : Manage for user/programmer or external processing.

#### **Implicit Cursor**

Oracle uses implicit cursors for its internal processing. Even if we execute a SELECT statement or DML statement Oracle reserves a private SQL area in memory called cursor.

Implicit cursor scope you can get information from cursor by using session attributes until another SELECT statement or DML statement execute. [Read More...](http://www.way2tutorial.com/plsql/plsql_implicit_cursor.php)

#### **Explicit Cursor**

Explicit Cursor which are construct/manage by user itself call explicit cursor.

User itself to declare the cursor, open cursor to reserve the memory and populate data, fetch the records from the active data set one at a time, apply logic and last close the cursor.

You can not directly assign value to an explicit cursor variable you have to use expression or create subprogram for assign value to explicit cursor variable.

Step for Using Explicit Cursor :

1. Declare cursor
2. Open cursor
3. Loop
4. Fetch data from cursor
5. Exit loop
6. Close cursor

### PL/SQL Implicit Cursor

Oracle uses implicit cursors for its internal processing. Even if we execute a SELECT statement or DML statement Oracle reserves a private SQL area in memory called cursor.

Implicit cursor scope you can get information from cursor by using session attributes until another SELECT statement or DML statement execute.

#### **Implicit Cursor Attributes**

Following are implicit cursor attributes,

|  |  |  |
| --- | --- | --- |
| Cursor Attribute | Cursor Variable | Description |
| %ISOPEN | SQL%ISOPEN | Oracle engine automatically open the cursor If cursor open **return TRUE** otherwise **return FALSE.** |
| %FOUND | SQL%FOUND | If SELECT statement return one or more rows or DML statement (INSERT, UPDATE, DELETE) affect one or more rows If affect **return TRUE** otherwise **return FALSE.** If not execute SELECT or DML statement **return NULL.** |
| %NOTFOUND | SQL%NOTFOUND | If SELECT INTO statement return no rows and fire no\_data\_found PL/SQL exception before you can check SQL%NOTFOUND. If not affect the row **return TRUE** otherwise **return FALSE.** |
| %ROWCOUNT | SQL%ROWCOUNT | Return the number of rows affected by a SELECT statement or DML statement (insert, update, delete). If not execute SELECT or DML statement **return NULL.** |

#### **Syntax**

cursor\_attribute ::=

{

cursor\_name |

cursor\_variable\_name |

:host\_cursor\_variable\_name

}

% {FOUND | ISOPEN | NOTFOUND | ROWCOUNT}

Explanation :

cursor\_name : cursor\_name identifies the current scope which are previously declared.

cursor\_variable\_name : cursor variable or parameter identifies the current scope which are previously declared.

host\_cursor\_variable\_name : host\_cursor\_variable\_name must be prefixed with a colon. Host cursor variable datatype must be compatible with the PL/SQL cursor variable.

#### **Implicit Cursor Example**

Following one emp\_information table:

|  |  |  |  |
| --- | --- | --- | --- |
| EMP\_NO | EMP\_NAME | EMP\_DEPT | EMP\_SALARY |
| 1 | Forbs ross | Web Developer | 45k |
| 2 | marks jems | Program Developer | 38k |
| 3 | Saulin | Program Developer | 34k |
| 4 | Zenia Sroll | Web Developer | 42k |

Now above employee information table update the employee name 'Saulin' department 'Program Developer' update to 'Web Developer'.

#### **Example Code**

*implicit\_cursor.sql*

SQL>set serveroutput on

SQL>edit implicit\_cursor

BEGIN

UPDATE emp\_information SET emp\_dept='Web Developer'

WHERE emp\_name='Saulin';

IF SQL%FOUND THEN

dbms\_output.put\_line('Updated - If Found');

END IF;

IF SQL%NOTFOUND THEN

dbms\_output.put\_line('NOT Updated - If NOT Found');

END IF;

IF SQL%ROWCOUNT>0 THEN

dbms\_output.put\_line(SQL%ROWCOUNT||' Rows Updated');

ELSE

dbms\_output.put\_line('NO Rows Updated Found');

END;

/

#### **Example Result**

**SQL>@implicit\_cursor**  
Updated - If Found  
1 Rows Updated  
  
PL/SQL procedure successfully operation.

### PL/SQL Explicit Cursor

Explicit Cursor which are construct/manage by user itself call explicit cursor.

User itself to declare the cursor, open cursor to reserve the memory and populate data, fetch the records from the active data set one at a time, apply logic and last close the cursor.

You can not directly assign value to an explicit cursor variable you have to use expression or create subprogram for assign value to explicit cursor variable.

#### **Step for Using Explicit Cursor**

1. Declare cursor

Declare explicit cursor has this syntax,

CURSOR cursor\_name [ parameter ] RETURN return\_type;

CURSOR cursor\_name [ parameter ] [ RETURN return\_type ]

IS SELECT STATEMENT;

Declaring explicit cursor example,

CURSOR c RETURN EMP\_DEPT%ROWTYPE; -- Declare c

CURSOR c IS -- Define c,

SELECT \* FROM emp\_information; -- all row return type

CURSOR c RETURN EMP\_DEPT%ROWTYPE IS -- Define c,

SELECT \* FROM emp\_information; -- repeating return type

1. Opening Explicit Cursor

DECLARE block you are already declare CURSOR now you can OPEN CURSOR by using following way, and allocate some reserve area for process database query.

OPEN cursor\_name [( cursor\_parameter )];

1. Loop

Loop iterate until ROW not found. Once found loop exit control goes next statement (outside loop).

1. Fetching data from cursor

Using FETCH statement you can fetch CURSOR data into explicit variable.

FETCH cursor\_name INTO variable;

1. Exit loop
2. Closing Explicit Cursor

This way you can close opened CURSOR.

CLOSE cursor\_name [( cursor\_parameter )];

Following emp\_information table having employee information, now we update information using Explicit Cursor,

|  |  |  |  |
| --- | --- | --- | --- |
| EMP\_NO | EMP\_NAME | EMP\_DEPT | EMP\_SALARY |
| 1 | Forbs ross | Web Developer | 45k |
| 2 | marks jems | Program Developer | 38k |
| 3 | Saulin | Program Developer | 34k |
| 4 | Zenia Sroll | Web Developer | 42k |

Now above employee information table update the employee name 'Saulin' department 'Program Developer' update to 'Web Developer'.

#### **Example Code**

*explicit\_cursor.sql*

SQL>set serveroutput on

SQL>edit explicit\_cursor

DECLARE

cursor c is select \* from emp\_information

where emp\_name='bhavesh';

tmp emp\_information%rowtype;

BEGIN

OPEN c;

Loop exit when c%NOTFOUND;

FETCH c into tmp;

update emp\_information set tmp.emp\_dept='Web Developer'

where tmp.emp\_name='Saulin';

END Loop;

IF c%ROWCOUNT>0 THEN

dbms\_output.put\_line(SQL%ROWCOUNT||' Rows Updated');

ELSE

dbms\_output.put\_line('NO Rows Updated Found');

END IF;

CLOSE c;

END;

/

#### **Example Result**

**SQL>@explicit\_cursor**  
1 Rows Updated  
  
PL/SQL procedure successfully completed.

### PL/SQL Cursors For Loop

PL/SQL cursor FOR loop has one great advantage of loop continued until row not found. In sometime you require to use explicit cursor with FOR loop instead of use OPEN, FETCH, and CLOSE statement.

FOR loop iterate repeatedly and fetches rows of values from database until row not found.

#### **Explicit Cursor FOR LOOP Example**

following one emp\_information table:

|  |  |  |  |
| --- | --- | --- | --- |
| EMP\_NO | EMP\_NAME | EMP\_DEPT | EMP\_SALARY |
| 1 | Forbs ross | Web Developer | 45k |
| 2 | marks jems | Program Developer | 38k |
| 3 | Saulin | Program Developer | 34k |
| 4 | Zenia Sroll | Web Developer | 42k |

Display employee number wise first two employee details emp,

#### **Example Code**

*cursor\_for\_loop.sql*

SQL>set serveroutput on

SQL>edit cursor\_for\_loop

DECLARE

cursor c is select \* from emp\_information

where emp\_no <=2;

tmp emp\_information%rowtype;

BEGIN

OPEN c;

FOR tmp IN c LOOP

FETCH c into tmp;

dbms\_output.put\_line('EMP\_No: '||tmp.emp\_no);

dbms\_output.put\_line('EMP\_Name: '||tmp.emp\_name);

dbms\_output.put\_line('EMP\_Dept: '||tmp.emp\_dept);

dbms\_output.put\_line('EMP\_Salary:'||tmp.emp\_salary);

END Loop;

CLOSE c;

END;

/

#### **Example Result**

**SQL>@cursor\_for\_loop**  
EMP\_No:    1  
EMP\_Name:  Forbs ross  
EMP\_Dept:  Web Developer  
EMP\_Salary:45k  
  
EMP\_No:    2  
EMP\_Name:  marks jems  
EMP\_Dept:  Program Developer  
EMP\_Salary:38k  
  
PL/SQL procedure successfully completed.

### PL/SQL Parameterized Cursor

PL/SQL Parameterized cursor pass the parameters into a cursor and use them in to query.

PL/SQL Parameterized cursor define only datatype of parameter and not need to define it's length.

Default values is assigned to the Cursor parameters. and scope of the parameters are locally.

Parameterized cursors are also saying static cursors that can passed parameter value when cursor are opened.

Following example introduce the parameterized cursor. following emp\_information table,

|  |  |  |  |
| --- | --- | --- | --- |
| EMP\_NO | EMP\_NAME | EMP\_DEPT | EMP\_SALARY |
| 1 | Forbs ross | Web Developer | 45k |
| 2 | marks jems | Program Developer | 38k |
| 3 | Saulin | Program Developer | 34k |
| 4 | Zenia Sroll | Web Developer | 42k |

#### **Example Code**

Cursor display employee information from emp\_information table whose emp\_no four (4).

*parameter\_cursor\_demo.sql*

SQL>set serveroutput on

SQL>edit parameter\_cursor\_demo

DECLARE

cursor c(no number) is select \* from emp\_information

where emp\_no = no;

tmp emp\_information%rowtype;

BEGIN

OPEN c(4);

FOR tmp IN c(4) LOOP

dbms\_output.put\_line('EMP\_No: '||tmp.emp\_no);

dbms\_output.put\_line('EMP\_Name: '||tmp.emp\_name);

dbms\_output.put\_line('EMP\_Dept: '||tmp.emp\_dept);

dbms\_output.put\_line('EMP\_Salary:'||tmp.emp\_salary);

END Loop;

CLOSE c;

END;

/

#### **Example Result**

**SQL>@parameter\_cursor\_demo**  
EMP\_No:    4  
EMP\_Name:  Zenia Sroll  
EMP\_Dept:  Web Developer  
EMP\_Salary:  42k  
  
PL/SQL procedure successfully completed.

Important key point you must remember

1. Scope of the parameters are locally
2. You can assign default value to a cursor parameter.