**PL/SQL**

CURSER

When an SQL statement is processed, Oracle creates a memory area known as context area. A cursor is a pointer to this context area. It contains all information needed for processing the statement. In PL/SQL, the context area is controlled by Cursor. A cursor contains information on a select statement and the rows of data accessed by it.

A cursor is used to referred to a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors:

* Implicit Cursors
* Explicit Cursors

1) PL/SQL Implicit Cursors

The implicit cursors are automatically generated by Oracle while an SQL statement is executed, if you don't use an explicit cursor for the statement.

These are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE etc. are executed.

Orcale provides some attributes known as Implicit cursor's attributes to check the status of DML operations. Some of them are: %FOUND, %NOTFOUND, %ROWCOUNT and %ISOPEN.

**For example:**When you execute the SQL statements like INSERT, UPDATE, DELETE then the cursor attributes tell whether any rows are affected and how many have been affected. If you run a SELECT INTO statement in PL/SQL block, the implicit cursor attribute can be used to find out whether any row has been returned by the SELECT statement. It will return an error if there no data is selected.

The following table soecifies the status of the cursor with each of its attribute.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| %FOUND | Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect at least one row or more rows or a SELECT INTO statement returned one or more rows. Otherwise it returns FALSE. |
| %NOTFOUND | Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect no row, or a SELECT INTO statement return no rows. Otherwise it returns FALSE. It is a just opposite of %FOUND. |
| %ISOPEN | It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL statements. |
| %ROWCOUNT | It returns the number of rows affected by DML statements like INSERT, DELETE, and UPDATE or returned by a SELECT INTO statement. |

PL/SQL Implicit Cursor Example

**Create customers table and have records:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** | **SALARY** |
| 1 | Ramesh | 23 | Allahabad | 20000 |
| 2 | Suresh | 22 | Kanpur | 22000 |
| 3 | Mahesh | 24 | Ghaziabad | 24000 |
| 4 | Chandan | 25 | Noida | 26000 |
| 5 | Alex | 21 | Paris | 28000 |
| 6 | Sunita | 20 | Delhi | 30000 |

Let's execute the following program to update the table and increase salary of each customer by 5000. Here, SQL%ROWCOUNT attribute is used to determine the number of rows affected:

**Create procedure:**

1. **DECLARE**
2. total\_rows number(2);
3. **BEGIN**
4. **UPDATE**  customers
5. **SET** salary = salary + 5000;
6. IF sql%notfound **THEN**
7. dbms\_output.put\_line('no customers updated');
8. ELSIF sql%found **THEN**
9. total\_rows := sql%rowcount;
10. dbms\_output.put\_line( total\_rows || ' customers updated ');
11. **END** IF;
12. **END**;
13. /

Output:

6 customers updated

PL/SQL procedure successfully completed.

Now, if you check the records in customer table, you will find that the rows are updated.

1. **select** \* **from** customers;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** | **SALARY** |
| 1 | Ramesh | 23 | Allahabad | 25000 |
| 2 | Suresh | 22 | Kanpur | 27000 |
| 3 | Mahesh | 24 | Ghaziabad | 29000 |
| 4 | Chandan | 25 | Noida | 31000 |
| 5 | Alex | 21 | Paris | 33000 |
| 6 | Sunita | 20 | Delhi | 35000 |

2) PL/SQL Explicit Cursors

The Explicit cursors are defined by the programmers to gain more control over the context area. These cursors should be defined in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

Following is the syntax to create an explicit cursor:

Syntax of explicit cursor

Following is the syntax to create an explicit cursor:

1. **CURSOR** cursor\_name **IS** select\_statement;;

Steps:

You must follow these steps while working with an explicit cursor.

1. Declare the cursor to initialize in the memory.
2. Open the cursor to allocate memory.
3. Fetch the cursor to retrieve data.
4. Close the cursor to release allocated memory.

1) Declare the cursor:

It defines the cursor with a name and the associated SELECT statement.

**Syntax for explicit cursor decleration**

1. **CURSOR** **name** **IS**
2. **SELECT** statement;

2) Open the cursor:

It is used to allocate memory for the cursor and make it easy to fetch the rows returned by the SQL statements into it.

**Syntax for cursor open:**

1. **OPEN** cursor\_name;

3) Fetch the cursor:

It is used to access one row at a time. You can fetch rows from the above-opened cursor as follows:

**Syntax for cursor fetch:**

1. **FETCH** cursor\_name **INTO** variable\_list;

4) Close the cursor:

It is used to release the allocated memory. The following syntax is used to close the above-opened cursors.

**Syntax for cursor close:**

1. **Close** cursor\_name;

PL/SQL Explicit Cursor Example

Explicit cursors are defined by programmers to gain more control over the context area. It is defined in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

Let's take an example to demonstrate the use of explicit cursor. In this example, we are using the already created CUSTOMERS table.

**Create customers table and have records:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** | **SALARY** |
| 1 | Ramesh | 23 | Allahabad | 20000 |
| 2 | Suresh | 22 | Kanpur | 22000 |
| 3 | Mahesh | 24 | Ghaziabad | 24000 |
| 4 | Chandan | 25 | Noida | 26000 |
| 5 | Alex | 21 | Paris | 28000 |
| 6 | Sunita | 20 | Delhi | 30000 |

**Create procedure:**

Execute the following program to retrieve the customer name and address.

1. **DECLARE**
2. c\_id customers.id%type;
3. c\_name customers.**name**%type;
4. c\_addr customers.address%type;
5. **CURSOR** c\_customers **is**
6. **SELECT** id, **name**, address **FROM** customers;
7. **BEGIN**
8. **OPEN** c\_customers;
9. LOOP
10. **FETCH** c\_customers **into** c\_id, c\_name, c\_addr;
11. EXIT **WHEN** c\_customers%notfound;
12. dbms\_output.put\_line(c\_id || ' ' || c\_name || ' ' || c\_addr);
13. **END** LOOP;
14. **CLOSE** c\_customers;
15. **END**;
16. /

Output:

1 Ramesh Allahabad

2 Suresh Kanpur

3 Mahesh Ghaziabad

4 Chandan Noida

5 Alex Paris

6 Sunita Delhi

PL/SQL procedure successfully completed.