Assignment No .6

Title of Assignment: Cursors :(All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)

Write a PL/SQL block of code using parameterized Cursor that will merge the data available in the newly created table N_Roll Call with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

Note: Instructor will frame the problem statement for writing PL/SQL block using all types of Cursors in line with above statement

Course Objective:

Implement PL/SQL Code block for given requirements

Course Outcome:

C306.4 Implement PL/SQL Code block for given requirements

Software Required: - Mysql

Theory: -

PL/SQL Cursor

Summary: in this tutorial, we will introduce you to **PL/SQL cursor**. You will learn step by step how to use a cursor to loop through a set of rows and process each row individually.

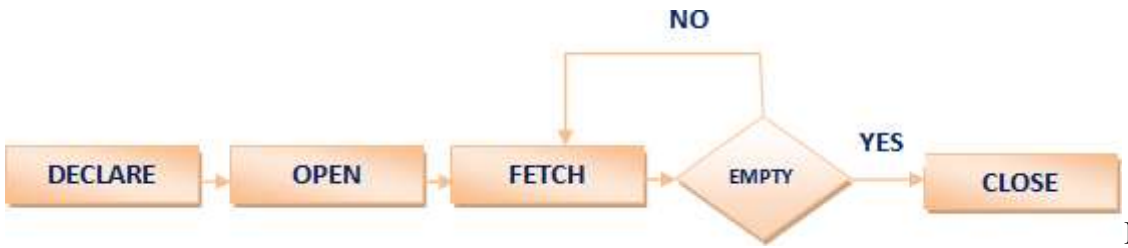
Introducing to PL/SQL Cursor

When you work with Oracle database, you work with a complete set of rows returned from an SQL SELECT statement. However the application in some cases cannot work effectively with the entire result set, therefore, the database server needs to provide a mechanism for the application to work with one row or a subset of the result set at a time. As the result, Oracle created PL/SQL cursor to provide these extensions.

A PL/SQL cursor is a pointer that points to the result set of an SQL query against database tables.

Working with PL/SQL Cursor

The following picture describes steps that you need to follow when you work with a PL/SQL cursor:



PL/SQL Cursor

Declaring PL/SQL Cursor

To use PL/SQL cursor, first you must declare it in the declaration section of PL/SQL block or in a package as follows:

cursor_name CURSOR

- First, you declare the name of the cursor after the keyword. The name of the cursor can have up to 30 characters in length and follows the naming rules of identifiers in PL/SQL. It is important to note that cursor's name is not a variable so you cannot use it as a variable such as assigning it to other cursor or using it in an expression. The parameter1,parameter2… are optional elements in the cursor declaration. These parameters allow you to pass arguments into the cursor. The RETURN return_specification is also an optional part.

- Second, you specify a valid SQL statement that returns a result set where the cursor points to.
- Third, you can indicate a list of columns that you want to update after the FOR UPDATE OF. This part is optional so you can omit it in the CURSOR declaration.

PL/SQL Cursor Attributes

These are the main attributes of a PL/SQL cursor and their descriptions.

Attribute	**Description**
cursor_name%FOUND	returns TRUE if record was fetched successfully by cursor cursor_name
cursor_name%NOTFOUND	returns TRUE if record was not fetched successfully by cursor cursor_name
cursor_name%ROWCOUNT	returns the number of records fetched from the cursor cursor_name at the time we test %ROWCOUNT attribute
cursor_name%ISOPEN	returns TRUE if the cursor cursor_name is open

Explicit Cursors

An **explicit cursor** is defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row. We can provide a suitable name for the cursor.

General Syntax for creating a cursor is as given below:

CURSOR cursor_name IS select_statement;

- *cursor_name – A suitable name for the cursor.*
- *select_statement – A select query which returns multiple rows.*
- General Syntax to open a cursor is:
- *OPEN cursor_name;*
- General Syntax to fetch records from a cursor is:
- *FETCH cursor_name INTO record_name;*
- OR

  FETCH cursor_name INTO variable_list;
- General Syntax to close a cursor is:
- *CLOSE cursor_name;*

- When a cursor is opened, the first row becomes the current row. When the data is fetched it is copied to the record or variables and the logical pointer moves to the next row and it becomes the current row. On every fetch statement, the pointer moves to the next row. If you want to fetch after the last row, the program will throw an error. When there is more than one row in a cursor we can use loops along with explicit cursor attributes to fetch all the records.

FOR LOOP

CursorSyntax

The syntax for the CURSOR FOR LOOP in Oracle/PLSQL is:

FOR record_index in

cursor_nameLOOP

{...statements...

}END LOOP;

Parameters or Arguments

record_index:The index of the

record.

cursor_name:The name of the cursor that you wish to fetch records from.

statements:The statements of code to execute each pass through the CURSOR FOR LOOP.

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 its 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
1	Forbs ross
2	marks jems
3	Saulin
4	Zenia Sroll

Example Code

Cursor display employee information from emp_information table whose emp_no four (4).

```
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);END Loop;
CLOSE
c;END;
/
```

SQL>@parameter_cursor_demo

EMP_No: 4

EMP_Name: Zenia

Conclusion: We have implemented all types of Cursors successfully.

Activity to be Submitted by Students

Write PL/SQL code to display Employee details using Explicit Cursors

Write PL/SQL code in Cursor to display employee names and salary.