K. Abhishek
TCOA76

DBMSL

Assignment No. : 7

Aim

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_RollCall 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.

Objective

Learning the concept of cursor in PL/SQL.

Theory

Cursor

For the processing of any SQL statement, database needs to allocate memory. This memory is called context area. The context area is a part of PGA(Process Global Area) and is allocated on the oracle server.
A cursor is associated with this work area used by ORACLE, for multi row queries. A cursor is a handle or pointer to the context area. The cursor allows to process contents in the context area row by row.

Types of cursors :

i) Implicit cursor - It's defined by ORACLE implicitly. ORACLE defines implicit cursor for every DML statement.
ii) Explicit cursor - These are user-defined cursors which are defined in the declaration section of the PL/SQL block.

Syntax:

```
DECLARE
    Cursor cursor_name IS select_statement or query;
BEGIN
    Open cursor_name;
Fetch cursor_name into list_of_variables;
    Close cursor_name;
END;
```

Explicit cursor attributes

i) %FOUND - It is boolean attribute. It returns TRUE if the previous fetch returns a row and FALSE if it doesn't.

ii) %NOTFOUND - It is boolean attribute. It returns FALSE if the previous fetch returns a row and TRUE if it doesn't. This is often used as the exit condition for the fetch loop.

iii) %ISOPEN - This attribute is used to determine whether or not the associated cursor is open. If so it returns TRUE otherwise FALSE.

Cursor Fetch loops

i) Simple loop

```
LOOP
    Fetch cursorname into list of variables;
EXIT WHEN cursor_name %NOTFOUND
    sequence_of_statements;
END LOOP;
```

ii) While loop

```
FETCH cursorname INTO list of variables;
WHILE cursorname % FOUND LOOP
    sequence_of_statements;
    FETCH cursorname INTO list of variables;
END LOOP;
```

iii) Cursor For loop

```
FOR variable_name IN cursorname LOOP
  -- an implicit fetch is done here.
  -- process the fetch records.
  Sequence_of_statements;
END LOOP;
```

PL/SQL Parameterized Cursor

It passes the parameters into a cursor and use them into query.
It defines only data type of parameter and not need to define its length.

```
DECLARE
  cursor cursorname (parameter_list)
  IS
   cursor query;
END;
```

Conclusion

Thus, we learned concept of cursor in PL/SQL.

```
 > create table o_rollcall(roll_no int,namevarchar(20),address
varchar(20));
Query OK, 0 rows affected (0.28 sec)

 > create table n_rollcall(roll_no int,namevarchar(20),address
varchar(20));
Query OK, 0 rows affected (0.27 sec)

 > insert into o_rollcall values('1','Hitesh','Nandura');Query OK, 1
row affected (0.05 sec)

 > insert into o_rollcall values('2','Piyush','MP');Query OK, 1
row affected (0.06 sec)

 > insert into o_rollcall values('3','Ashley','Nsk');Query OK, 1
row affected (0.05 sec)

 > insert into o_rollcall values('4','Kalpesh','Dhule');Query OK, 1
row affected (0.05 sec)

 > insert into o_rollcall values('5','Abhi','Satara');Query OK, 1
row affected (0.04 sec)
 > delimiter //
 > create procedure p3(in r1 int)
    -> begin
    -> declare r2 int;
    -> declare exit_loop boolean;
    -> declare c1 cursor for select roll_no from o_rollcallwhere
roll_no>r1;
    -> declare continue handler for not found setexit_loop=true;
    -> open c1;
    -> e_loop:loop
    -> fetch c1 into r2;
    -> if not exists(select * from n_rollcall whereroll_no=r2)
```

```
    -> then
    -> insert into n_rollcall select * from o_rollcall where
roll_no=r2;
    -> end if;
    -> if exit_loop
    -> then
    -> close c1;
    -> leave e_loop;
    -> end if;
    -> end loop e_loop;
    -> end
    -> //
Query OK, 0 rows affected (0.00 sec)


 > call p3(3);
-> //
Query OK, 0 rows affected (0.10 sec)


 > select * from n_rollcall;
    -> //
+---------+---------+---------+
| roll_no | name    | address |
+---------+---------+---------+
|       4 | Kalpesh | Dhule   |
|       5 | Abhi    | Satara  |
+---------+---------+---------+
2 rows in set (0.00 sec)


 > call p3(0);
    -> //
Query OK, 0 rows affected (0.22 sec)


 > select * from n_rollcall;
    -> //
```

```
+---------+---------+---------+
| roll_no | name    | address |
+---------+---------+---------+
|       4 | Kalpesh | Dhule   |
|       5 | Abhi    | Satara  |
|       1 | Hitesh  | Nandura |
|       2 | Piyush  | MP      |
|       3 | Ashley  | Nsk     |
+---------+---------+---------+
5 rows in set (0.00 sec)

 > insert into o_rollcall values('6','Patil','Kolhapur');
    -> //
Query OK, 1 row affected (0.04 sec)

 > call p3(4);
    -> //
Query OK, 0 rows affected (0.05 sec)

 > select * from n_rollcall;
    -> //
+---------+---------+----------+
| roll_no | name    | address  |
+---------+---------+----------+
|       4 | Kalpesh | Dhule    |
|       5 | Abhi    | Satara   |
|       1 | Hitesh  | Nandura  |
|       2 | Piyush  | MP       |
|       3 | Ashley  | Nsk      |
|       6 | Patil   | Kolhapur |
+---------+---------+----------+
6 rows in set (0.00 sec)
```