

Assignment No.6



Title :- Write a PL-SQL block for parameterized cursor.

problem statement :- Write a PL-SQL block of code using parameterized cursor, that will merge the data available in new created table.

Objective :-

- To understand the types of cursor
- To understand how to use cursor with PL-SQL block.

SQL package :- SQL package

Learning objective :-

- To understand & implement types of cursor with PL-SQL block code.

Learning outcomes:-

Implement PL-SQL block code

Implement types of cursor

Theory :-

PL-SQL stands for procedural language / structured query language. PL-SQL offers a set of procedural commands.

• Block in SQL :-

i) Declarative :- statement that declare variable, constants & other code elements which can be used within code

ii) Executable :- statement that are run when block is executed.

iii) Exception handling :- a specially structured section you can use to catch or trap.

eg.

Begin

```
DBMS_OUTPUT.put-line ("Hello world");
```

END;

• Cursor :-

A cursor is temporary work area created in system memory when a SQL statement is executed.

A cursor contains information on select statement and rows of data accessed by it.

The temporary work area is used to store the data retrieved from the database & manipulate this data.

A cursor can hold more than one row, but can process only one row at a time.

There are two types of PL-SQL cursor :-

i) Implicit cursor :-

These are created by default when DML statement like insert, update & delete statements are executed. They are also created when select statement that return just one row is executed.

ii) Explicit cursor :-

They must be created when you are executing a select statement that return more than one row. Even though the cursor stores multiple records, only one record can be processed at a time, which is called as current row.

e.g.

```
Declare var-rownumber(5);
```

```
Begin
```

```
update employee
```

```
set salary = salary + 1000;
```

```
IF SQL%NOTFOUND THEN
```

```
dbms-output-line('None of salaries updated');  
put
```

```
else SQL%NOTFOUND THEN
```

```
var-row := SQL%ROWCOUNT;
```

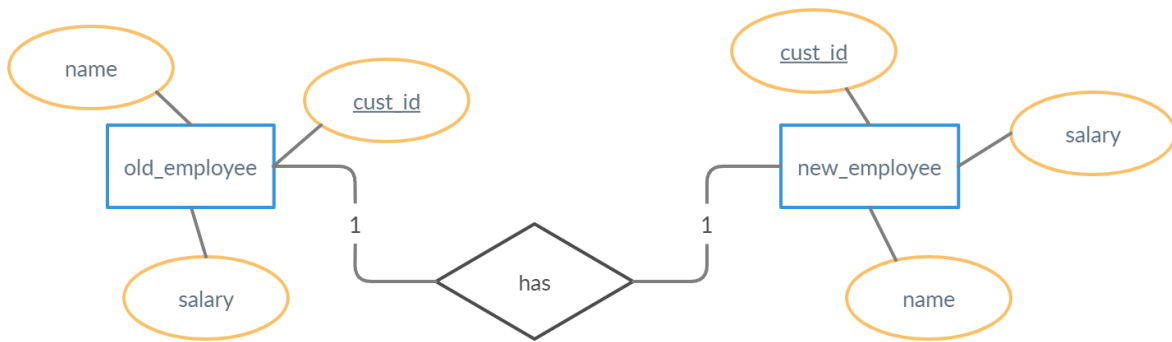
```
dbms-output-line('Salaries for ' || var-row || ');  
put
```

```
END IF;
```

```
END;
```

Conclusion :- Thus we implemented PL-SQL cursor in this assignment.

ER Diagram:-



Code:-

```
create database OurCursor;
```

Assignment No :- 6 Roll No:-31384

```
use OurCursor;

create table O_employee(id int primary key auto_increment,name varchar(230),salary int);
create table N_employee(id int primary key auto_increment,name varchar(230),salary int);
insert into O_employee values(1,"Swapnil",2000);
insert into O_employee values(2,"Ganesh",3000);
insert into O_employee values(3,"rahul",4000);
insert into N_employee values(3,"rahul",4000);
```

```
select * from O_employee;
select * from N_employee;
```

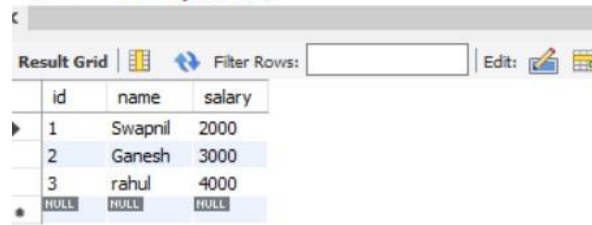
```
call MyCursor;
```

```
DELIMITER //
create procedure MyCursor()
begin
declare id1 int;
declare ename1 varchar(30);
declare salary1 int;
declare exit_loop boolean;
declare cur2 cursor for select id,name,salary from O_employee;
declare continue handler for not found set exit_loop=true;
open cur2 ;
l2:loop fetch cur2 into id1,ename1,salary1;
select id1,ename1,salary1;
insert into N_employee values(id1,ename1,salary1) on duplicate key update
id=id1,name=ename1,salary=salary1;
if exit_loop then leave l2;
end if;
end loop l2;
close cur2;
end//DELIMITER ;
```

Output:-

Old Employee table after inserting three values manually and before calling procedure:-

```
12
13 • select * from O_employee;
14 • select * from N_employee;
15 • call MyCursor;
```

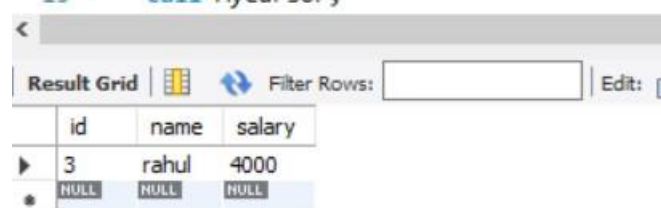


	id	name	salary
▶	1	Swapnil	2000
	2	Ganesh	3000
	3	rahul	4000
*	NULL	NULL	NULL

O_employee 9 ×

New employee after inserting values manually and before calling procedure :-

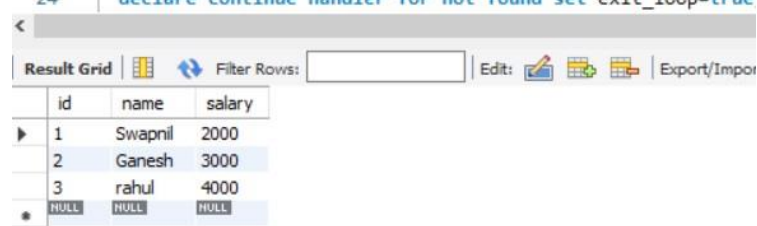
```
13 • select * from O_employee;
14 • select * from N_employee;
15 • call MyCursor;
```



	id	name	salary
▶	3	rahul	4000
*	NULL	NULL	NULL

New employee table after calling procedure (Since Rahul record is duplicate so its skipped as per assignment requiremnet) :-

```
23 declare cur2 cursor for select id,name,salary from O_empl
24 declare continue handler for not found set exit_loop=true
```



	id	name	salary
▶	1	Swapnil	2000
	2	Ganesh	3000
	3	rahul	4000
*	NULL	NULL	NULL

N_employee 15 ×

ure :-