Database Management Systems Laboratory

Assignment A6

Date of Submission: 15-09-2020

Title: CURSORS

Roll number: 31301

Name: Aboli Marathe

Class: TE-3

Batch: K-3

PICT, Pune

ASSIGNMENT 6

TITLE CURSORS

DATE

PROBLEM STATEMENT Write a PL/SQL block of code using parameterized cursor, that will merge the data available in the newly created table N_EmpId with the data available in the table O_EmpId IF the data in the first table already exists in the second table then data should be skipped.

LEARNING OBJECTIVES
- To understand the types of cursors
- To understand how to use cursors with PL\SQL block

LEARNING OUTCOMES Students will be able to:
- Implement diff types of cursors
- Use PL\SQL

H/W & S/W REQUIREMENTS SQL package, 64-bit Linux Fedora, Monitor, Keyboard, Mouse

THEORY

- PL/SQL stands for Procedural Language / Structured Query Language. PL/SQL offers a set of procedural commands, organized with blocks that complement & extend the reach of SQL.

- Blocks in PL/SQL:
  It is a block-structured language. A PL/SQL block is defined by the keywords DECLARE, BEGIN, EXCEPTION and END, which break up the block in 3 sections-

↳ Declarative
↳ Executable
↳ Exception Handling

Eg
```
BEGIN
    dbms_op.put_line ("Hello Wo
END;
```

CURSORS

- A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.
- There are 2 types of cursors in PL/SQL:

a) Implicit cursors-

These are created by default when DML statements like INSERT, UPDATE and DELETE statements are executed. Also created when a SELECT statement that returns just one row is executed.

b) Explicit cursors

They must be created when you are executing a SELECT statement that returns more than one row.

Eg
```
DECLARE var_rows number(5);
BEGIN
    UPDATE employee
    Set salary = salary + 1000.
    IF SQL% NOTFOUND THEN
     dbms_op. put_line('None salary upda
    ELSIF SQL%FOUND THEN
     var_rows:= SQL%ROWCOUNT.
     dbms op. put_line('Salaries fo' || var_ro
          'employees are updated');
    END IF;
END;
```

Conclusion We have successfully studied and implemented the Concept of cursors in PL/SQL and applied it in real world applications.

CODE AND OUTPUTS

```
SQL> create table O_EmpId( id INTEGER not null, name varchar(25) not null);

Table created.

SQL> create table N_EmpId( id INTEGER not null, name varchar(25) not null);

Table created.
```

```
SQL> select * from O_EmpId;

        ID NAME
---------- -------------------------
         1 Aboli
         2 Sarah
         3 Bulu
         4 Lily

SQL> insert into N_EmpId(id, name) values (1,'Aboli');

1 row created.

SQL> insert into N_EmpId(id, name) values (3,'Bulu');

1 row created.

SQL> select * from N_EmpId;

        ID NAME
---------- -------------------------
         1 Aboli
         3 Bulu
```

```
SQL> declare
  2  newcid int;
  3  newname varchar(25);
  4  oldcid int;
  5  oldname varchar(25);
  6  cursor newcursor(n int) is select * from N_EmpId where id=n;
  7  cursor oldcursor is select * from O_EmpId;
  8  begin
  9  open oldcursor;
 10  loop
 11  fetch oldcursor into newcid, newname;
 12  exit when oldcursor%notfound;
 13  open newcursor(newcid);
 14  fetch newcursor into oldcid, oldname;
 15  if newcursor%notfound then
 16  insert into N_EmpId values(newcid,newname);
 17  end if;
 18  close newcursor;
 19  end loop;
 20  close oldcursor;
 21  end;
 22  /

PL/SQL procedure successfully completed.
```

Alternate Code using MySQL procedures.

```
create procedure cur_merge()

begin

declare oldemp_id int;

declare oldemp_name varchar(15);

declare newemp_id int;

declare newemp_name varchar(15);

declare common int;


declare finished integer default 0;


declare oldcursor cursor for select * from old_emp;

declare newcursor cursor for select * from new_emp where id = common;


declare continue handler for not found set finished =1;
```

```
open oldcursor;

for_loop : loop

set finished=0;

fetch oldcursor into oldemp_id , oldemp_name;

set common = oldemp_id;

if finished = 1 then
leave for_loop;
end if;

open newcursor;

fetch newcursor into newemp_id , newemp_name;

if finished = 1 then
insert into new_emp values (oldemp_id, oldemp_name);
end if;

close newcursor;
end loop for_loop;
close oldcursor;
end;
```

```
SQL> select * from O_EmpId;

        ID NAME
---------- ------------------------
         1 Aboli
         2 Sarah
         3 Bulu
         4 Lily

SQL> select * from N_EmpId;

        ID NAME
---------- ------------------------
         1 Aboli
         3 Bulu
         2 Sarah
         4 Lily
```