| Ex.No.7 | CURSOR |
|---|---|

## AIM

To implement cursors in DBMS for efficient row-by-row data retrieval and manipulation.

## CREATING A TABLES

```
SQL> create table emp1 (id number(6), name varchar2 (50), basic
number(10,2));

Table created.

SQL> create table cust (id number(6), name varchar2(50), address
varchar2(100));

Table created.
```

## INSERTING VALUES INTO TABLE

```
SQL> insert into emp1 values(1, 'kabesh', 5000);

1 row created.

SQL> insert into emp1 values(2, ‘kamalesh', 6000);

1 row created.

SQL> insert into emp1 values(3, 'sanjay', 7000);

1 row created.

SQL> insert into cust values(101, 'karthik', 'Namakkal');
```

```
1 row created.

SQL> insert into cust values(102, 'jegan', 'Salem');

1 row created.

SQL> insert into cust values (103, 'kavin', 'Erode');

1 row created.

SQL> commit;

Commit complete.
```

**IMPLICIT CURSOR**

**EXAMPLE 1**

```
SQL> declare
       totalrows number(2);
       begin
       update emp1 set basic = basic + 500;

       if sql%notfound then
          dbms_output.put_line('No emp1loyees updated.');
         elsif sql%found then
         totalrows := sql%rowcount;
         dbms_output.put_line(totalrows || ' emp1loyees updated.');
      end if;
  end;
  /
3 emp1loyees updated.

PL/SQL procedure successfully completed.
```

**EXAMPLE 2**

```
SQL> declare
```

```
       deleted number(2);
      begin
       delete from emp1 where basic < 5500;

       if sql%notfound then
          dbms_output.put_line('no emp1loyees deleted.');
       else
          deleted := sql%rowcount;
         dbms_output.put_line(deleted || ' emp1loyees deleted.');
      end if;
  end;
  /
No emp1loyees deleted.

PL/SQL procedure successfully completed.
```

**EXAMPLE 3**

```
SQL> declare
       total_inserted number(2);
    begin
       insert into customers values (104, 'iyyappan', 'perundurai');
       insert into customers values (105, 'praveen', 'erode');

       total_inserted := sql%rowcount;
       dbms_output.put_line(total_inserted || ' customers inserted.');
   commit;
   end;
   /
1 customers inserted.

PL/SQL procedure successfully completed.
```

**EXPLICIT CURSOR**

**EXAMPLE 1**

```
SQL> declare
```

```
      c_id  cust.id %type;
      c_name cust.name%type;
      c_addr cust.address%type;

      cursor c_cust is
      select id, name, address from customers;
   begin
      open c_cust;
      loop
        fetch c_cust into c_id, c_name, c_addr;
        exit when c_cust%notfound;

        dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
     end loop;
     close c_cust;
  end;
  /
101 Karthik Namakkal
102 Jegan Salem
103 Kavin Erode

PL/SQL procedure successfully completed.
```

**EXAMPLE 2**

```
SQL> declare
      e_id  emp1.id%type;
      e_name   emp1.name%type;
      e_basic emp1.basic%type;

      cursor emp1_cursor is
         select id, name, basic from emp1l68 where basic > 5500;

   begin
     open emp1_cursor;
    loop
        fetch emp1_cursor into e_id, e_name, e_basic;
```

```
         exit when emp1_cursor%notfound;

         dbms_output.put_line('id: ' || e_id || ', name: ' || e_name || ',
basic: ' || e_basic);
      end loop;
     close  emp1_cursor;
  end;
  /
ID: 1, Name: kamalesh, Basic: 6500
ID: 2, Name: sanjay, Basic: 7500

PL/SQL procedure successfully completed.
```

**<u>EXAMPLE 3</u>**

```
SQL> declare
       c_id   customers.id%type;
       c_name customers.name%type;
       c_addr customers.address%type;

       cursor t_customers is
          select id, name, address from customers where address = 'salem';

   begin
      open t_customers;
      loop
         fetch t_customers into c_id, c_name, c_addr;
         exit when t_customers%notfound;

         dbms_output.put_line('customer name: ' || c_name || ', address: ' ||
c_addr);
      end loop;
      close t_customers;
  end;
  /
Customer Name: Jegan, Address: Salem

PL/SQL procedure successfully completed.
```

| CONTENTS | MARKS ALLOTED | MARKS OBTAINED |
|---|---|---|
| Aim,Algorithm,SQL,PL/SQL | 30 | |
| Execution and Result | 20 | |
| Viva | 10 | |
| Total | 60 | |

**RESULT :**

Achieved controlled and optimized data processing using cursors, enabling complex operations with improved precision.