Cursor

Cursor is temporary working area which is used to retrieve and process one or more rows in data. Oracle associates every SELECT statement with a cursor to hold the query information in this context area.

Types of Cursor

There are two types of cursors – implicit and explicit cursor.

- **Implicit cursor:** Oracle automatically (implicit) controls or processes the information of SQL statement executed. In this process, the user is unaware of implicit cursor. Oracle automatically performs the OPEN, FETCH, and CLOSE operations.
- **Explicit cursor:** Explicit cursor is used for the query that returns more than one row of data. These cursors are explicitly declared in the DECLARE section of the PL/SQL block. This declaration allows to sequentially process each row of data as the cursor returns it. In explicit cursor DECLARE,OPEN,FETCH,and CLOSE operations are done by the programmer.

The process of working with an explicit cursor:

- Declare: The cursor is initialised into temporary memory area.
- Open: The cursor is opened which is declared, and the temporary memory area is allotted.
- Fetch: Cursor which is declared and opened can now retrieve rows from data.
- Close: The CLOSE statement disables the cursor, and releases the temporary memory area.

The status of the cursor for each of these attributes are defined in the below table.

Attributes	Return Value	Example
%FOUND	The return value is TRUE, if the DML statements like INSERT, DELETE and UPDATE affect at least one row and if SELECT... INTO statement return at least one row.	SQL%FOUND

	The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE do not affect row and if SELECT... INTO statement do not return a row.	

%NOTFOUND	The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE at least one row and if SELECT... INTO statement return at least one row.	SQL%NOTFOUND
	The return value is TRUE, if a DML statement like INSERT, DELETE and UPDATE do not affect even one row and if SELECT... INTO statement does not return a row.	
%ROWCOUNT	Return the number of rows affected by the DML operations INSERT, DELETE, UPDATE, SELECT	

Simple Cursor

```
Declare
A item%rowtype;
cursor c1 is
select * from item ;
begin
open c1 ;
loop
fetch c1 into A;
exit when c1%notfound ;
 dbms_output.put_line(A.itemid|| ' ' ||A.name);
end loop;
close c1;
end;
```

Cursor Using loops

Cursor using simple loop

```
Declare
A item%rowtype;
```

```
cursor c1 is
select * from item ;
begin
      open c1 ;
 loop
      fetch c1 into A;
      exit when c1%notfound ;
       dbms_output.put_line(A.itemid|| ' ' ||A.name);
    end loop;
   close c1;
end;
```

Cursor using while loop

```
declare
 cursor c1 is select * from std;
  a std%rowtype;
begin
   open c1;
   fetch c1 into a;
  dbms_output.put_line(a.rollno ||'  ' ||a.name ||' ' || a.address||'  ' ||a.mno);
while c1%found loop
  fetch c1 into a;
  exit when not c1%found;
  dbms_output.put_line(a.rollno ||'  ' ||a.name ||' ' || a.address||'  ' ||a.mno);
 end loop;
 close c1;
end;
```

Cursor using for loop

```
declare
   cursor c1 is select * from std;
    a std%rowtype;
  begin
    for a in c1 loop
       dbms_output.put_line(a.rollno ||'  ' ||a.name ||' ' || a.address||'  '
||a.mno);
     end loop;
  end;
```

Cursor with parameter

The cursor can be declared with the parameter or without the parameter. It can have any number of parameters as per requirement.Cursors with Parameters are used to work with particular data. Parameters are used to create reusable and adaptable code. Explicit cursors may be declared with parameters. The

parameter contains a variable and its datatype. The parameter can have a default value associated with a variable.

Following cursor is initialized with dept id and access department name

```
      declare
 cursor c1(c number) is select deptname from dept where depid=c;
  a dept.deptname%type;
  n number;
Begin
      n:=&n;
      open c1(n);
     fetch c1 into a;
   dbms_output.put_line(a);
 close c1;
end;
```

Check deptid is exist then insert employee record into emp table (using Cursor)

```
declare
   cursor c1(d number) is select dept_id from dept where dept_id=&de;
  de number;
  eid number;
  n varchar2(30);
  s number;
begin
   eid:=&eid;
    n:='&n';
    s:=&s;
   open c1(de);
     fetch c1 into de;
       if c1%found then
         insert into emp values(eid,n,s,de);
   else
       dbms_output.put_line('dept number' || de ||'does NOT exist');
   end if;
  close c1;
end;
```

Cursor within cursor

```
declare
  a std%rowtype;
  b marks%rowtype;
  cursor c1 is select * from std;
   cursor c2 is select * from marks;
    Begin
```

```
      open c1;
       open c2;
  dbms_output.put_line('Rollno'
||chr(9)||'Name'||chr(9)||'Address'||chr(9)||'Subject'||chr(9)||'Subject'||chr(9)
||'Subject'||chr(9)||'Percentage');
      loop
      fetch c1 into a;
        fetch c2 into b;
         exit when (c1%notfound);
          exit when (c2%notfound);
    dbms_output.put_line(a.rollno || '  ' ||a.name  ||' ' ||a.address||'  '||b.sub1||'   '
||b.sub2||' ' ||b.sub3||'   '||b.per);
  end loop;
    close c2;
    close c1;
   end;
```

Cursor with Join

```
declare
e emp.empid%type;
n emp.empname%type;
d dept.deptname%type;
  cursor c1 is select e.empid,d.deptname,e.empname from emp e join dept d on
e.depid=d.depid;
     Begin
        open c1;
      loop
       fetch c1 into e,d,n;
        exit when (c1%notfound);
        dbms_output.put_line(e||'  ' ||n||'  '|| d);
      end loop;
    end;
```

Cursor with Sub Quries

```
        declare
           cursor c1 is select ename,salary from emp where salary =(select
        max(salary) from emp where salary <(select max(salary) from emp));
            a emp.ename%type;
            b emp.salary%type;
          begin
            open c1;
            fetch c1 into a,b;
                      dbms_output.put_line(a ||'  ' ||b);
             close c1;
```

```
end;
```

Cursor with Procedure

Cursors are particularly useful in stored procedures. They allow you to use only one query to accomplish a task that would otherwise require several queries. However, all cursor operations must execute within a single procedure. A stored procedure cannot open, fetch, or close a cursor that was not declared in the procedure. Cursors are undefined outside the scope of the stored procedure.

```
create or replace procedure emp_proc
as
   cursor c1 is select *from emp;
   a emp%rowtype;
  begin
    open c1;
      loop
       fetch c1 into a;
       exit when (c1%notfound);
              dbms_output.put_line(a.eid ||'    ' ||a.ename || '  ' || a.salary);
     end loop;
     close c1;
  end;
/
```

Procedure is created
Then call /execute procedure
Exec emp_proc