Cursors

* To process a SQL statement, PL/SQL opens a work area called a context area.
* PL/SQL uses this area to execute SQL statements and store processing information
* A PL/SQL construct called ‘Cursor’ allows you to name a context area, access its information and in some cases, control its processing

**Explicit Cursors**

Defined by the user to keep track of which row is being processed, when a query returns multiple rows

**Defining a Cursor**

* A cursor is defined in the declarative part of the PL/SQL block by naming it and associating it with a query

CURSOR <cursorname> IS

<SELECT statement>;

* Example

CURSOR emp\_cur IS

SELECT empno, ename, job, sal

FROM emp;

* A Cursor can be manipulated using
  + OPEN
  + FETCH
  + CLOSE
* Cursor must be declared before it can be referenced using the OPEN, CLOSE or FETCH statements

**The OPEN Statement**

* Initializes or opens a cursor
* Cursor must be opened before any rows are returned by the query

OPEN <cursorname>

Example --

OPEN emp\_cur;

**The FETCH Statement**

* Can be executed repeatedly until all rows have been retrieved

FETCH <cursorname> INTO var1, …, varN;

OR

FETCH <cursorname> INTO record\_variable;

* Example

FETCH emp\_cur INTO mrec;

**The CLOSE Statement**

* Closes the cursor and makes the active set undefined

CLOSE <cursorname>;

* Example

CLOSE emp\_cur;

* Once a cursor is closed, it can be reopened by using the OPEN statement

**Attributes of Explicit Cursors**

* Every cursor has four attributes that can be used to access the cursor’s context area
  + %NOTFOUND
  + %FOUND
  + %ROWCOUNT
  + %ISOPEN
* To use these attributes, simple append them to the name of the cursor
* %NOTFOUND
  + evaluates to TRUE if last FETCH failed because no more rows were available
  + evaluates to FALSE if last FETCH returned a row
* %FOUND
  + evaluates to TRUE if last FETCH returned a row
  + evaluates to FALSE if last FETCH failed because no more rows were available
* %ROWCOUNT
  + returns the number of rows FETCHed from the active set so far
* %ISOPEN
  + evaluates to TRUE if an explicit cursor is open
  + evaluates to FALSE if an explicit cursor is closed

Examples of Cursor –

**1) To transfer names and sal of employees from emp table where sal >= 2500 in table try1**

create table try1

(no number,

ename varchar2(50),

sal number);

Declare

Cursor cf is

select ename,sal

from emp

where sal >= 2500;

M cf%rowtype;

N number;

Begin

Open cf;

N := 0;

Loop

Fetch cf into M;

Exit when cf%notfound;

N := cf%rowcount;

Insert into try1

values(N,M.ename,M.sal);

End Loop;

Close cf;

End;

**Cursor FOR Loop**

* Implicitly declares its loop index as a record of %ROWTYPE,
* Implicitly opens the cursor
* Repeatedly fetches rows of values from the active set into fields in the record
* Implicitly closes the cursor when all rows have been processed or the loop is exited
* The statements in the loop construct are executed once for each row that satisfies the query associated with the cursor name
* Cursor FOR loop is used to simplify coding
* **No need of --**

1)Open cursor

2)Fetch

3)Exit

4)Close cursor

**To show records where salary is > 2000**

Declare

Cursor cf is

select \*

from emp

where sal >= 2000;

Begin

For mrec in cf

Loop

dbms\_output.put\_line(mrec.ename||' '||mrec.sal||' '||mrec.deptno);

End Loop;

End;

/

# For Loops using sub queries

*No need of declaring cursor.*

A private cursor within an anonymous block can be created.

To show names of employees who have job MANAGER.

begin

for MREC in (select \* from emp)

Loop

if MREC.job = 'MANAGER' then

dbms\_output.put\_line('Name is ' ||MREC.ename);

end if;

END LOOP;

end;

**Parameterized Cursor**

The same cursor can be reopened and closed with different active sets.

declare

cursor cf **(pjob emp.job%type)**

is

select empno,ename,job,sal

from emp

where job = **pjob**;

M cf%rowtype;

Begin

open cf**('ANALYST');**

LOOP

FETCH CF INTO M;

EXIT WHEN CF%NOTFOUND;

dbms\_output.put\_line('Name of ' || M.job || ' is ' || M.ename);

end loop;

**close cf;**

open cf**('CLERK');**

LOOP

FETCH CF INTO M;

EXIT WHEN CF%NOTFOUND;

dbms\_output.put\_line('Name of ' || M.job || ' is ' || M.ename);

end loop;

**close cf;**

END;

/

**Implicit Cursors**

* + - Automatically defined and opened, by Oracle, to process each SQL statement
    - most recently opened context area is referred to as a ‘SQL%’ cursor

**Attributes of Implicit Cursors**

* Although OPEN, CLOSE and FETCH statements cannot be used to manipulate the SQL% cursor, the attributes can be used to access its context area
* Attributes evaluate to NULL, before the cursor is opened automatically
* **The following four cursor attributes can be used to access the SQL% cursor’s context area**
  + **SQL%NOTFOUND**
  + **SQL%FOUND**
  + **SQL%ROWCOUNT**
  + **SQL%ISOPEN**
* **SQL%NOTFOUND**
  + evaluates to TRUE if an INSERT, UPDATE or DELETE statement affected no rows, else it evaluates to FALSE
* **SQL%FOUND**
  + logical opposite of SQL%NOTFOUND
  + evaluates to TRUE if an INSERT, UPDATE or DELETE affected one or more rows, else it evaluates to FALSE
* **SQL%ROWCOUNT**
  + returns the number of rows affected by an INSERT, UPDATE or DELETE statement
* **SQL%ISOPEN**
  + Oracle automatically closes an implicit cursor after executing its associated SQL statement
  + For an implicit cursor SQL%ISOPEN always evaluates to FALSE

**Example of Implicit Cursors**

Create table newemp

As

Select \* from emp;

Begin

Delete from newemp

where ename = '&name';

If **SQL%Found** Then

dbms\_output.put\_line('Record found and it is deleted');

End If;

If **SQL%NotFound** Then

dbms\_output.put\_line('No record is present of the given name.');

End If;

End;

/

-- After successfully deleting any record please give the **rollback** command.

10) Implicit Cursor for rowcount

Declare

c number := 0;

Begin

Update NewEmp

set sal = sal + 500

where deptno = &deptno;

**/\*If no record is updated since the deptno supplied is wrong then giving**

**the customised error message.\*/**

If **SQL%Rowcount** = 0 then

dbms\_output.put\_line('No records are updated since the department number entered is not in the table.');

End if;

**/\*To prevent sal to be updated where deptno is > 3 \*/**

If SQL%RowCount > 3 then

**Rollback;**

dbms\_output.put\_line('Cannot update since there are more than 3 employees.');

End If;

If SQL%RowCount <= 3 then

c := SQL%RowCount;

dbms\_output.put\_line(c || ' records updated.');

**Commit;**

End If;

End;

**Cursor used for Updating actual rows of the table:**

**Aka For Update …… Where Current Of**

Drop Table NewEmp;

Create Table NewEmp

As

Select \* from Emp;

Declare

cursor cf is select \* from NewEmp **For Update**;

M cf%RowType;

Begin

Open cf;

Loop

Fetch Cf into M;

Exit When Cf%NotFound;

If M.Sal Between 800 and 1000 Then

M.Sal := M.Sal + 7000;

ElsIf M.Sal Between 1001 and 2000 Then

M.Sal := M.Sal + 5500;

ElsIf M.Sal Between 2001 and 3000 Then

M.Sal := M.Sal + 2000;

ElsIf M.Sal > 3000 Then

M.Sal := M.Sal + 1000;

End If;

Update NewEmp

Set Sal = M.Sal

**Where Current of Cf;**

End Loop;

Close cf;

commit;

End;