• What is cursor?

The oracle engine uses a work area for its internal processing in order to execute an SQL statement. This work area is private to SQL's operations and is called cursor.

The data that is stored in the cursor is called the **Active Data Set.** The size of the cursor in memory is the size required to hold the number of rows in the **Active Data Set**

• How to Use Cursor?

In PL/SQL block SELECT statement can not return more than one row at a time. So Cursor use to some group of rows (more than one row) for implementing certain logic to get all the group of records.

• Classification of CURSORS

1. **Implicit Cursor** (Internal Cursor)
2. **Explicit Cursor** (User-defined Cursor)

1. PL/SQL Implicit Cursor

- Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement.
- Programmers cannot control the implicit cursors and the information in it.
- Implicit cursor scope you can get information from cursor by using session attributes until another SELECT statement or DML statement execute.

- Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted.
- **SQL cursor** has the attributes like %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT.

- **Implicit Cursor Attributes**

Cursor Attribute	Cursor Variable	Description
%ISOPEN	SQL%ISOPEN	Oracle engine automatically open the cursor If cursor open return TRUE otherwise return FALSE.
%FOUND	SQL%FOUND	If SELECT statement return one or more rows or DML statement (INSERT, UPDATE, DELETE) affect one or more rows If affect returnTRUE otherwise return FALSE. If not execute SELECT or DML statement return NULL.
%NOTFOUND	SQL%NOTFOUND	IS the logical opposite of %FOUND. It returns TRUE, If an INSERT, UPDATE or DELETE affected no rows, or a SELECT INTO statement returns no rows. Otherwise returns FALSE.
%ROWCOUNT	SQL%ROWCOUNT	Return the number of rows affected by a SELECT statement or DML statement (INSERT, UPDATE, DELETE). If not execute SELECT or DML statement return NULL.

Implicit Cursor Example

EMP_NO	EMP_NAME	EMP_DEPT	EMP_SALARY
1	JEMS	Web Developer	45000
2	SUNIL	Program Developer	38000
3	AADITYA	Program Developer	34000
4	SHREYA	Web Developer	42000

Now above employee information table update the employee name 'AADITYA' department 'Program Developer' changes to 'Web Developer'.

```
SQL>BEGIN
        UPDATE emp_information SET emp_dept='Web Developer'
                WHERE emp_name='AADITYA';
        IF SQL%FOUND THEN
                dbms_output.put_line('Updated - If Found');
        END IF;
        IF SQL%NOTFOUND THEN
                dbms_output.put_line('NOT Updated - If NOT Found');
        END IF;
        IF SQL%ROWCOUNT>0 THEN
                dbms_output.put_line(SQL%ROWCOUNT||' Rows Updated');
        ELSE
                dbms_output.put_line('NO Rows Updated Found');
        END;
```

Output :

```
Updated - If Found
1 Rows Updated
```

- **USER DEFINE RECORD**

Database columns and tables have similar attributes, which you can use to ease maintenance.

1. **%TYPE**
2. **%ROWTYPE**

1. **%TYPE :** the %TYPE attribute provides the datatype of a variable or database column. This is particularly useful when declaring variables that will hold database values.
   Syntax : variablename tablename.columnname%type;
   Example :**My_title books.title%TYPE;**

To declare a variable named my_title having the same datatype as column title,we use dot notation and the %TYPE attribute.

Declaring my_title with %TYPE has two advantages.

- we need not know the exact datatype of title.
- If we change the database definition of title(make it a longer character string, for example) the datatype of my_title changes accordingly at runtime.

2. %ROWTYPE :

Attribute provides a record type that represents a row in a table. The record can store an entire row of data selected from the table or fetched from a cursor or cursor variable.

Columns in a row and corresponding fields in a record have the same name and datatype.

Syntax : variablename tablename%rowtype;
Example :dept_rec dept%ROWTYPE;

declare record name dept_rec. its fields have the same names and datatype as the columns in the dept table.

You use dot notation to reference fields as the following example shows:\

My_deptno := dept_rec.deptno;

If you declare a cursor that retrieves the last name, salary, and job title of employee, you can use %ROWTYPE to declare a record that store the same information as follows:

```
DECLARE
CURSOR c1 IS SELECT ename,sal,job FROM emp;
Emp_rec c1%ROWTYPE;

FETCH c1 into emp_rec;
```

The value in the ename column of the emp table is assigned to the ename field of emp_rec, the value in the sal column is assigned to the sal field.

2. Explicit Cursor

Explicit Cursor which is managed by user itself calls explicit cursor.

User itself to declare the cursor, open cursor to reserve the memory and populate data, fetch the records from the active data set one at a time, apply logic and last close the cursor.

You can not directly assign value to an explicit cursor variable you have to use expression or create subprogram for assign value to explicit cursor variable.

Step for Using Explicit Cursor :

1. Declare cursor
2. Open cursor
3. Loop
4. Fetch data from cursor
5. Exit loop
6. Close cursor

Step for Using Explicit Cursor

1. **Declare cursor**
   syntax: CURSOR cursor_name IS SELECT STATEMENT;
   Example:
   CURSOR EMP_DEPT IS
   SELECT * FROM emp_information;
2. **Opening Cursor**
   Syntax : **OPEN cursor_name;**
   Example : OPEN EMP_DEPT
3. **Loop**
   Loop iterate until ROW not found. Once found loop exit control goes next statement (outside loop).
4. **Fetching data from cursor**
   Using FETCH statement you can fetch CURSOR data into explicit variable. Fetching the cursor involves accessing one row at a time.
   Syntax : FETCH cursor_name INTO variable1,Variable2..;
   Example : FETCH EMP_DEPT INTO EMP_NO,EMP_NAME;
5. **Exit loop**
6. **Closing Cursor** :Closing the cursor means releasing the allocated memory.
   Syntax : CLOSE cursor_NAME;
   Example : CLOSE EMP_DEPT;

Example1 :

Select * from customers;

```
+----+---------- +-----+ -----------       +----------+
| ID | NAME |   AGE |   ADDRESS       | SALARY |
+----+---------- +-----+ -----------       +----------+
| 1 | Ramesh   | 32 |   Ahmedabad   | 2500.00 |
| 2 | Khilan   | 25 |   Delhi           | 2000.00 |
| 3 | kaushik  | 23 |   Kota            | 2500.00 |
| 4 | Chaitali | 25 |   Mumbai        | 7000.00 |
| 5 | Hardik   | 27 |   Bhopal        | 9000.00 |
| 6 | Komal    | 22 |   MP              | 5000.00 |
+----+---------- +---+   -----------+      --------------+
```

```
DECLARE
        c_id customers.id%type;
        c_name customers.name%type;
        c_addr customers.address%type;
        CURSOR c_customers is
        SELECT id, name, address FROM customers;
BEGIN
        OPEN c_customers;
        LOOP
                FETCH c_customers into c_id, c_name, c_addr;
                dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
                EXIT WHEN c_customers%notfound;
        END LOOP;
CLOSE c_customers;
END;
/
```

Output :

```
1  Ramesh  Ahmedabad
2  Khilan  Delhi
3  kaushik  Kota
4  Chaitali  Mumbai
5  Hardik  Bhopal
6  Komal  MP
PL/SQL procedure successfully completed.
```

Example 2:

Following emp_information table

EMP_NO	EMP_NAME	EMP_DEPT	EMP_SALARY
1	JEMS	Web Developer	45000
2	SUNIL	Program Developer	38000
3	AADITYA	Program Developer	34000
4	SHREYA	Web Developer	42000

Now above employee information table update the employee name 'Aaditya' department 'Program Developer' update to 'Web Developer'.

```
DECLARE
        cursor c is select * from emp_information
        where emp_name='AADITYA';
        tmp emp_information%rowtype;
BEGIN
        OPEN c;
        Loop exit when c%NOTFOUND;
                FETCH c into temp;
                update emp_information set tmp.emp_dept='Web Developer'
                where tmp.emp_name='Saulin';
```

```
        END Loop;
IF c%ROWCOUNT>0 THEN
        dbms_output.put_line(SQL%ROWCOUNT||' Rows Updated');
ELSE
        dbms_output.put_line('NO Rows Updated Found');
END IF;
CLOSE c;
END;
/
```

OUTPUT :

1 Rows Updated
PL/SQL procedure successfully completed.

- ## PL/SQL Cursors For Loop

PL/SQL cursor FOR loop has one great advantage of loop continued until row not found. In sometime you require to use explicit cursor with FOR loop instead of use **OPEN, FETCH, and CLOSE statement.**
FOR loop iterate repeatedly and fetches rows of values from database until row not found.
For loop variable is automatically declare as a rowtype.

Explicit Cursor FOR LOOP Example
following one emp_information table:

EMP_NO	EMP_NAME	EMP_DEPT	EMP_SALARY
1	JEMS	Web Developer	45000
2	SUNIL	Program Developer	38000
3	AADITYA	Program Developer	34000
4	SHREYA	Web Developer	42000

Display employee number wise first two employee details emp,

Example Code

```
SQL>set serveroutput on
SQL>
DECLARE
        cursor c is
        select * from emp_information
        where emp_no <=2;
BEGIN
        FOR tmp IN c LOOP
                dbms_output.put_line('EMP_No:    '||tmp.emp_no);
                dbms_output.put_line('EMP_Name:  '||tmp.emp_name);
                dbms_output.put_line('EMP_Dept:  '||tmp.emp_dept);
                dbms_output.put_line('EMP_Salary:'||tmp.emp_salary);
        END Loop;
```

```
END;
```
OUTPUT
SQL>
```
EMP_No:  1
EMP_Name:  JEMS
EMP_Dept:  Web Developer
EMP_Salary:45k

EMP_No:  2
EMP_Name:  SUNNIL
EMP_Dept:  Program Developer
EMP_Salary:38k
```

PL/SQL procedure successfully completed.

- ## PL/SQL Parameterized Cursor

PL/SQL Parameterized cursor pass the parameters into a cursor and use them in to query.
PL/SQL **Parameterized cursor define only datatype of parameter and not need to define it's length or size.**
Default values are assigned to the Cursor parameters. and scope of the parameters are locally.
Parameterized cursors are also saying static cursors that can pass parameter value when cursors are opened.
Following example introduce the parameterized cursor. following emp_information table,

EMP_NO	EMP_NAME	EMP_DEPT	EMP_SALARY
1	JEMS	Web Developer	45000
2	SUNIL	Program Developer	38000
3	AADITYA	Program Developer	34000
4	SHREYA	Web Developer	42000

Example Code
Cursor display employee information from emp_information table whose emp_no four (4).
parameter_cursor_demo.sql
```
SQL>set serveroutput on
SQL>
DECLARE
        cursor c(no number) is
        select * from emp_information
        where emp_no = no;
BEGIN
        FOR tmp IN c(&no) LOOP
        dbms_output.put_line('EMP_No:  '||tmp.emp_no);
        dbms_output.put_line('EMP_Name:  '||tmp.emp_name);
        dbms_output.put_line('EMP_Dept:  '||tmp.emp_dept);
        dbms_output.put_line('EMP_Salary:'||tmp.emp_salary);
```

```
        END Loop;
END;   /
```

OUTPUT :

SQL> enter value for no : 4

EMP_No: 4
EMP_Name: SHREYA
EMP_Dept: Web Developer
EMP_Salary: 42k

PL/SQL procedure successfully completed.

Important key point you must remember

Scopes of the parameters are locally

You can assign default value to a cursor parameter.