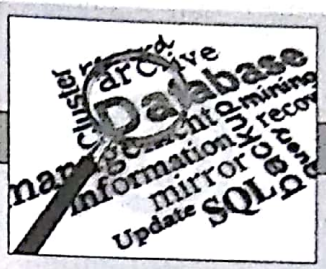
Chapter 32



CURSOR MANAGEMENT

CHAPTER OBJECTIVES

In this chapter you will learn:

- Concept and uses of Cursor Management
- Types of Cursors like Implicit Cursor and Explicit Cursor
- Use of general Cursor Attributes
- Concept and applications of Implicit Cursor Handling
- Concept and applications of Explicit cursor Handling
- Examples of Cursor FOR Loop
- Passing of parameters to Cursors

32.1 INTRODUCTION TO CURSOR

The oracle Engine uses a work area to execute SQL statements and store information. A cursor is a PL/SQL construct that allows us to name these work areas, and to access their stored information. The data that is stored in the cursor is called the Active Data Set.

Example: When a user fires a select statement as:

SELECT empno, ename, job, sal FROM emp WHERE deptno = 10;

All the rows returned by the query are stored in the cursor at the Server and will be as displayed at the client end.

Types of Cursors

There are two types of cursors:

- Implicit Cursor
- Explicit Cursor

Implicit Cursor : Implicit Cursors are declared by PL/SQL implicitly for all SQL statements. They are opened and managed by Oracle engine internally. So there is no need to open and manage by the users, these are operations, which are performed automatically.

Explicit Cursor : Explicit cursors are user-defined cursors for processing of multiple records returned by a query. Explicit Cursors are declared explicitly, along with other identifiers to be used in a block, and manipulated through specific statements within the block's executable actions. These are user-defined cursors, defined in the DECLARE section of the PL/SQL block. The user-defined cursor needs to be opened, before the reading of the rows can be done, after which the cursor is closed. Cursor marks the current position in an active set.

32.1.1 General Cursor Attributes

Whenever any cursor is opened and used, the oracle engine creates a set of four system variables, which keeps track of the current status of a cursor. These cursor variables can be accessed and used in a PL/SQL block. Both Implicit and Explicit cursor have four attributes.

They are described as:

Attributes	Description
%ISOPEN	It returns TRUE if cursor is open, FALSE otherwise.
%FOUND	It returns TRUE if record was fetched successfully from the opened cursor, and FALSE otherwise.
%NOTFOUND	It returns TRUE if record was not fetched successfully and FALSE otherwise.
%ROWCOUNT	It returns number of records processed from cursor.

32.2 IMPLICIT CURSOR HANDLING

Implicit Cursors named "SQL" are declared by PL/SQL implicitly for all SQL statements. Implicit cursors attributes can be used to access information about the status of last insert, update, delete or single-row select statements. This can be done by preceding the implicit cursor attribute with the cursor name (i.e. SQL). The value of the cursor attributes always refers to the most recently executed SQL statement, wherever the statement appears.

Implicit Cursor Attributes

Attributes	Description
SQL%ISOPEN	It is always FALSE because Oracle automatically closes an Implicit cursor after executing its SQL statement.
SQL%FOUND	It is TRUE if the most recently executed DML statement was successful.
SQL%NOTFOUND	It is TRUE if the most recently executed DML statement was not successful.
SQL%ROWCOUNT	It returns the number of rows affected by an INSERT, UPDATE, DELETE, or single row SELECT statement.

Implicit Cursor attributes are used in procedural statements but not in SQL statements.

Use of SQL%NOTFOUND Attribute

SQL%NOTFOUND evaluates to TRUE if an INSERT, UPDATE, or DELETE affected no rows or a SELECT INTO returned no rows. Otherwise%SQLNOTFOUND evaluates to FALSE.

Example 32.1: ***Consider a PL/SQL code to display a message to check whether the record is deleted or not.***

Solution:


```
BEGIN
        DELETE EMP WHERE EMPNO=&EMPNO;
        IF SQL%NOTFOUND THEN
              DBMS_OUTPUT.PUT_LINE('RECORD NOT DELETED');
        ELSE
              DBMS_OUTPUT.PUT_LINE('RECORD DELETED');
        END IF;
END;
```

Use of SQL%FOUND Attribute

SQL%FOUND is the logical opposite of SQL%NOTFOUND. Until a SQL data manipulation statement is executed, SQL%FOUND evaluates to NULL. SQL%FOUND evaluates to TRUE if an INSERT, UPDATE, or DELETE affects one or more rows or a SELECT INTO returns one or more rows. Otherwise, SQL%FOUND evaluate to FALSE.

Example 32.2: ***Consider a PL/SQL code to display a message to check whether the record is deleted or not with the use of SQL%FOUND.***

Solution:

```
BEGIN
        DELETE EMP WHERE EMPNO=&EMPNO;
        IF SQL%FOUND THEN
              DBMS_OUTPUT.PUT_LINE('RECORD DELETED');
        ELSE
              DBMS_OUTPUT.PUT_LINE('RECORD IS NOT DELETED');
        END IF;
END;
```

Use of SQL%ROWCOUNT Attribute

It returns the number of rows affected by an INSERT, UPDATE, DELETE, or single row SELECT statement.

Example 32.3: ***Consider a PL/SQL code to display a message to give the number of records deleted by the delete statement issued in a PL/SQL block.***

Solution:

```
DECLARE
        N NUMBER;
BEGIN
        DELETE EMP WHERE DEPTNO=&DEPTNO;
        N:=SQL%ROWCOUNT;
        DBMS_OUTPUT.PUT_LINE('TOTAL NUMBER OF RECORD DELETED' || N);
END;
```

32.3 EXPLICIT CURSOR HANDLING

The following steps are involved in using an explicit cursor and manipulating data in its active set:

- Declare a cursor mapped to a SQL select statement that retrieves data for processing.
- Open the cursor.
- Fetch data from the cursor one row at a time into memory variables.
- Process the data held in the memory variables as required using a loop.
- Exit from the loop after processing is complete.
- Close the cursor.

Cursor Declaration

During declaration of a cursor, cursor name is given and also defines the structure of the query to be performed within it. The CURSOR statement is used to declare explicit cursor.

Syntax:

```
CURSOR <cursor-name> IS <select statement>;
```

Whereas <select statement> includes most of the usual clauses, but INTO clause is not allowed.

Opening a Cursor

Here query execution is done. It also binds any variables that are referenced. After opening the cursor the rows returned by the query are available for fetching.

Syntax:

```
OPEN <cursor-name>;
```

This statement is used within the executable actions of the block. It also establishes an active set of rows. When cursor is opened it points to the first row in the active set.

Fetching of individual rows

After the cursor is opened the current row is loaded into variables. The current row is the row at which the cursor is currently pointing. The retrieval of data into PL/SQL variables is done through **FETCH** statement.

Syntax:

```
FETCH <cursor-name> INTO <variables>;
```

For each column value returned by the query associated with the cursor, there must be a corresponding variable in the INTO list. Also their data types must be compatible. Each FETCH causes the cursor to move its pointer to the next row in the active set, and hence each FETCH will access a different row returned by the query. Fetch operation is illustrated in figure 32.1.

It explicitly closes the cursor, allowing it to be reopened, if required. This means that an active set can be re-established several times.

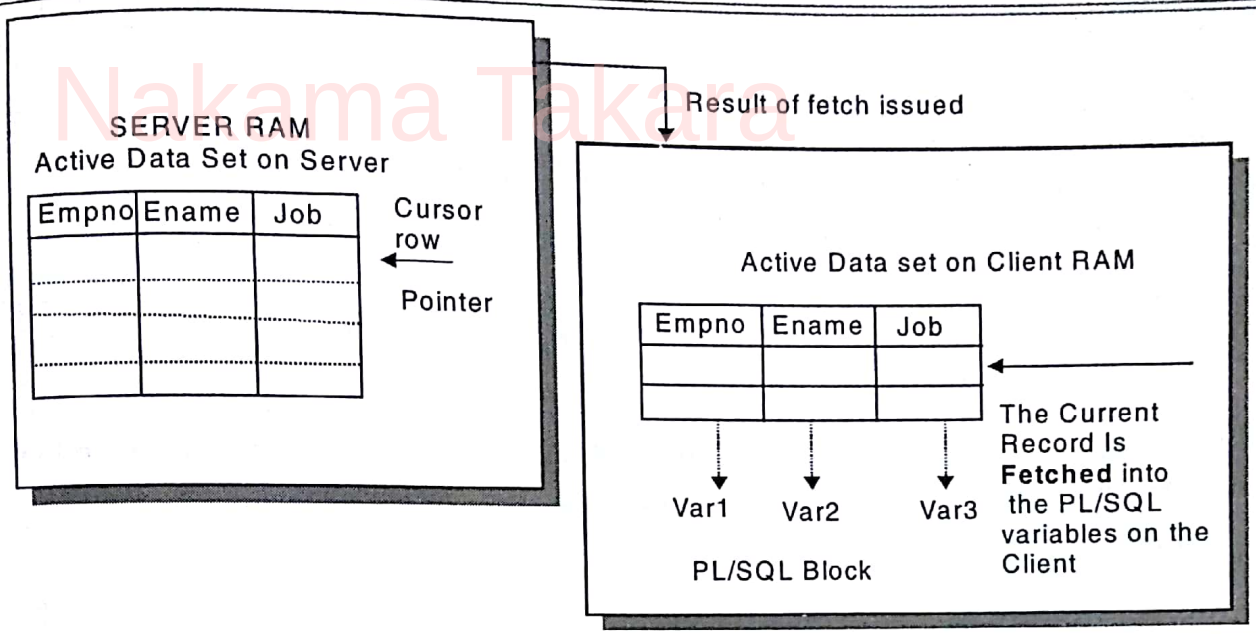


Figure 32.1 Fetch Operation

Syntax:

```
Close <cursor-name>;
```

By closing a cursor a working set of rows are released that were produced by last Opening of the cursor. It is then possible to reopen the cursor and it estavlishes a fresh working set.

Explicit Cursor Attributes

Each cursor that the user defines has four attributes. These are:

Attributes	Description
%ISOPEN	Evaluates to TRUE, if the cursor is opened, otherwise evaluates to FALSE.
%FOUND	Evaluates to TRUE when last fetch succeeded.
%NOTFOUND	Evaluates to TRUE, if the last fetch failed, i.e. no more rows are left.
%ROWCOUNT	Returns the number of rows fetched

There are four attributes of explicit cursor for obtaining status information about a cursor. When appended to the cursor name these attributes let the user access useful information about the execution of a Multi row query.

Example 32.4: ***Consider a PL/SQL code to display the empno, ename, job of employees of department number 10.***

Solution:

```
DECLARE
 CURSOR C1 IS SELECT EMPNO, ENAME, JOB FROM EMP  WHERE
 DEPTNO=10;
 REC C1%ROWTYPE;  /*rec is a row type variable for cursor c1 record,
                                    containing empno, ename and job*/

BEGIN
```

```
OPEN C1;
LOOP
FETCH C1 INTO REC;
EXIT WHEN C1%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('EMPNO '||REC.EMPNO);
DBMS_OUTPUT.PUT_LINE('ENAME '||REC.ENAME);
DBMS_OUTPUT.PUT_LINE('JOB '||REC.JOB);
END LOOP;
CLOSE C1;
END;
```

Example 32.5: ***Consider a PL/SQL code to display the employee number and name of top 5 highest paid employees.***

Solution:

```
DECLARE
        EMPNAME EMP.ENAME%TYPE;
        EMPSAL EMP.SAL%TYPE;
        CURSOR TEMP1 IS SELECT ENAME, SAL FROM EMP ORDER BY
 SAL DESC;
BEGIN
        OPEN TEMP1;
        LOOP
        FETCH TEMP1 INTO EMPNAME, EMPSAL;
        EXIT WHEN TEMP1%ROWCOUNT>5 OR TEMP1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(EMPNAME || EMPSAL);
        END LOOP;
        CLOSE TEMP1;
END;
```

Example 32.6: ***Consider a PL/SQL code to increase the salary of employee according to following rule:***

Salary of department number 10 employees increased by 1000.

Salary of department number 20 employees increased by 500.

Salary of department number 30 employees increased by 800.

Store the employee number, old salary and new salary in a table temp having three columns empno, old and new.

Solution:

```
DECLARE
        OLDSAL NUMBER;
        NEWSAL NUMBER;
        CURSOR C1 IS SELECT * FROM EMP;
        REC C1%ROWTYPE;
BEGIN
        OPEN C1;
```

```
    LOOP
        FETCH C1 INTO REC;
        EXIT WHEN C1%NOTFOUND;
        SELECT SAL INTO OLDSAL FROM EMP WHERE
        EMPNO=REC.EMPNO;
        IF REC.DEPTNO=10 THEN
            UPDATE EMP SET SAL=SAL+1000 WHERE
            EMPNO=REC.EMPNO;
            SELECT SAL INTO NEWSAL FROM EMP WHERE
            EMPNO=REC.EMPNO;
        END IF;
        IF REC.DEPTNO=20 THEN
        UPDATE EMP SET SAL=SAL+500 WHERE
        EMPNO=REC.EMPNO;
        SELECT SAL INTO NEWSAL FROM EMP WHERE
EMPNO=REC.EMPNO;
    END IF;
    IF REC.DEPTNO=30 THEN
        UPDATE EMP SET SAL=SAL+800 WHERE
        EMPNO=REC.EMPNO;
        SELECT SAL INTO NEWSAL FROM EMP WHERE
    EMPNO=REC.EMPNO;
    END IF;
    INSERT INTO TEMP(EMPNO,OLD,NEW)
    VALUES(REC.EMPNO,OLDSAL,NEWSAL);
END LOOP;
CLOSE C1;
END;
```

Example 32.7: ***Consider a PL/SQL code to calculate the total and percentage of marks of the students in four subjects from table result, which has the following structure:***

(rno,s1,s2,s3,s4,total,percentage).

The rollno and marks in each subject are stored in the database, PL/SQL code calculate the total and percentage of each student and update the database.

Solution:

```
DECLARE
    T NUMBER;
    PER NUMBER;
    CURSOR C1 IS SELECT * FROM RESULT;
    REC C1%ROWTYPE;
BEGIN
    OPEN C1;
    LOOP
        FETCH C1 INTO REC;
```

```
        EXIT WHEN C1%NOTFOUND;
        T:=REC.S1+REC.S2+REC.S3+REC.S4;
        PER:=T/4;
        UPDATE RESULT SET TOTAL=T, PERCENTAGE=PER WHERE
    RNO=REC.RNO;
    END LOOP;
    CLOSE C1;
END;
```


32.4 CURSOR FOR LOOP

The Cursor FOR Loop implicitly declares its loop index as a record of type %ROWTYPE, opens a cursor, repeatedly fetches rows of the values from the active set into fields in the record, then closes the cursor when all rows have been processed or when the EXIT command is encountered.

The syntax for the FOR Loop is:

```
FOR  <variable name> IN <cursor_name> LOOP
     <statements>;
END LOOP;
```

It is a machine defined loop exit i.e. when all the values in the FOR construct are exhausted looping stops.

A cursor for loop automatically does the following:

- ✦ Implicitly declares its loop index or variable_name as a %rowtype record.
- ✦ Opens a cursor.
- ✦ Fetches a row from the cursor for each loop iteration.
- ✦ Closes the cursor when all rows have been processed.

Example 32.8: ***Consider a PL/SQL code to display the empno, ename, job of employees of department number 10 with CURSOR FOR Loop statement.***

Solution:

```
DECLARE
    CURSOR C1 IS SELECT EMPNO, ENAME, JOB FROM EMP WHERE
    DEPTNO=10;
BEGIN
    FOR REC IN C1 LOOP
        DBMS_OUTPUT.PUT_LINE('EMPNO '||REC.EMPNO);
        DBMS_OUTPUT.PUT_LINE('ENAME '||REC.ENAME);
        DBMS_OUTPUT.PUT_LINE('JOB '||REC.JOB);
    END LOOP;
END;
```

Example 32.9: ***Consider a PL/SQL code to display the employee number and name of top 5 highest paid employees with CURSOR FOR LOOP statement.***

Solution:


```
DECLARE
        CURSOR TEMP1 IS SELECT ENAME, SAL FROM EMP ORDER BY
 SAL DESC;
BEGIN
        FOR REC IN TEMP1 LOOP
                EXIT WHEN TEMP1%ROWCOUNT>5;
                DBMS_OUTPUT.PUT_LINE(REC.ENAME || REC.SAL);
        END LOOP;
END;
```

32.5 CURSORS WITH PARAMETERS

Commercial applications require that the query, which, defines the cursor, be generic and the data that is retrieved from the table be allowed to change according to need. Oracle recognizes this and permits the creation of a parameterized cursor prior opening. Parameters allow values to be passed to a cursor when it is opened, and used within a query when it executes. Using parameters explicit cursor may be opened more than once in a block, returning a different working set each time. The parameters can be either a constant or a variable.

Syntax to declare parameterized cursor:

```
CURSOR cursor_name (variable_name datatype) IS
        <SELECT statement...>
```

Syntax to open a Parameterized cursor and passing values to the cursor:

```
OPEN cursor_name (value/variable/expression);
```

NOTE: The scope of cursor parameters is local to that cursor, which means that they can be referenced only within the query declared in the cursor declaration. Each parameter in the declaration must have a corresponding value in the open statement.

Example 32.10: ***Consider a PL/SQL code to calculate the total salary of first n records of emp table. The value of n is passed to cursor as parameter.***

Solution:

```
DECLARE
        E NUMBER(5);
        S NUMBER(5):=0;
        CURSOR C1(N NUMBER) IS SELECT SAL FROM EMP  WHERE
 ROWNUM<=N;
BEGIN
        OPEN C1(&N);
        LOOP
                FETCH C1 INTO E;
```

```
            EXIT WHEN C1%NOTFOUND;
            S:=S+E;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE(S);
        CLOSE C1;
END;
```


With Cursor FOR Loop

```
DECLARE
        S NUMBER(5):=0;
        CURSOR C1(N NUMBER) IS SELECT SAL FROM EMP WHERE
        ROWNUM<=N;
BEGIN
        FOR REC IN C1(&N) LOOP
            S:=S+REC.SAL;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE(S);
 END;
```

Flash Back

The oracle Engine uses a work area to execute SQL statements and store information. A cursor is a PL/SQL construct that allows us to name these work areas, and to access their stored information. The data that is stored in the cursor is called the Active Data Set.

There are two types of cursors: Implicit Cursor and Explicit Cursor. Implicit Cursors are declared by PL/SQL implicitly for all SQL statements. They are opened and managed by Oracle engine internally. So there is no need to open and manage by the users, these are operations are performed automatically. Explicit cursors are user-defined cursors for processing of multiple records returned by a query. Explicit Cursors are declared explicitly, along with other identifiers to be used in a block, and manipulated through specific statements within the block's executable actions.

Whenever any cursor is opened and used, the oracle engine creates a set of four system variables, which keeps tack of the current status of a cursor. These are:

Attributes	Description
%ISOPEN	It returns TRUE if cursor is open, FALSE otherwise.
%FOUND	It returns TRUE if record was fetched successfully from the opened cursor, and FALSE otherwise.
%NOTFOUND	It returns TRUE if record was not fetched successfully and FALSE otherwise.
%ROWCOUNT	It returns number of records processed from cursor.

Implicit Cursors named "SQL" are declared by PL/SQL implicitly for all SQL statements Implicit cursors attributes can be used to access information about the status of last insert, update, delete or single-row select statements.

The following steps are involved in using an explicit cursor and manipulating data in its active set:

- Declare a cursor mapped to a SQL select statement that retrieves data for processing.
- Open the cursor.
- Fetch data from the cursor one row at a time into memory variables.
- Process the data held in the memory variables as required using a loop.
- Exit from the loop after processing in complete.
- Close the cursor.

The Cursor FOR Loop implicitly declares its loop index as a record of type %ROWTYPE, opens a cursor, repeatedly fetches rows of the values from the active set into fields in the record, then closes the cursor when all rows have been processed or when the EXIT command is encountered.

Oracle also permits the creation of a parameterized cursor prior opening. Parameters allow values to be passed to a cursor when it is opened, and used within a query when it executes. Using parameters explicit cursor may be opened more than once in a block, returning a different working set each time.

REVIEW QUESTIONS

1. What is the importance of Cursor? What are its types?
2. What are the Cursor attributes? Explain the importance of each?
3. How we can handle the implicit cursors?
4. What are the steps to be followed to handle the user defined cursors?
5. What is the importance of Cursor For Loop? How it simplifies the operations? Explain with suitable examples.
6. How we can pass the parameters to a cursor? Explain with examples.

HANDS ON SESSION

1. What is wrong with the following Cursor declaration?

   CURSOR C1 (pempno IN number(4)) IS
   SELECT empno, ename From emp where empno= pempno;

   (a) The INTO clause is missing.
   (b) Inside the cursor definition, the variable should be preceded with a colon :pempno;
   (c) IN cannot be specified in the cursor definition.
   (d) The size of the data type should not be specified in the cursor definition.

2. Consider the following PL/SQL Block. What is the value of V_COUNT when the block is executed and no rows are deleted.

```
DECLARE
V_COUNT          NUMBER;
BEGIN
```

```
DELETE FROM EMP WHERE EMPNO < 0;
V_COUNT := SQL%ROWCOUNT;
END;
```

(a) NULL (b) TRUE (c) 0 (d) FALSE

3. In a PL/SQL block, what is the value of SQL%NOTFOUND before Executing any DML statement.
   (a) NOTFOUND (b) TRUE (c) NULL (d)FALSE
4. Which of the following statement requires an explicit cursor if processing more than one row.
   (a) SELECT (b) UPDATE (c) DELETE (d) INSERT
5. Which commands are allowed inside a PL/SQL block? Choose all that apply.
   (a) TRUNCATE (b) DELETE (c) SAVEPOINT (d) ALTER TABLE
6. Which of the following cursor attributes are invalid ? Choose two.
   (a) %NOTOPEN (b) %NOTFOUND (c) %FOUND (d) %OPEN
7. If c1 is the cursor that is opened in a PL/SQL block, which attribute will have the number of rows fetched so far ?
   (a) C1%COUNT (b) C1%ROWCOUNT
   (c) COUNT %C1 (d) C1%COUNTROW
8. Which of the following statements are true about implicit cursors.
   (a) Implicit cursors are used for SQL statements that are not named.
   (b) Developer should use implicit cursors with great care.
   (c) Implicit cursors are used in Cursor-for loop to handle data processing
   (d) Implicit cursors are no longer a feature in Oracle.
9. Which of the following is not a feature of CURSOR-FOR loop?
   (a) Record type declaration.
   (b) Opening and Parsing of SQL statements.
   (c) Fetches records from cursor.
   (d) Requires EXIT condition to be defined.
10. Which line in the following statement will produce an error ?
    (a) CURSOR action_cursor is (b) SELECT name, rate, action
    (c) INTO action_record (d) FROM action_table
    (e) There is no error in this code

Solution Keys

1. (d)	2. (c)	3. (c)	4. (a)	5. (b&c)	6.(a&d)	7. (b)	8. (a)	9. (d)
10. (c)								

●●●