Chapter 18

CURSOR HANDLING

- What is a cursor?
- When do we need explicit cursor?
- Handling explicit cursor
- Cursor FOR loop
- Sample program
- Implicit cursor
- Cursor attributes
- Input arguments to cursor
- FOR UPDATE and CURRENT OF clauses

What is a cursor?

Oracle uses a work area to execute SQL commands and store processing information. PL/SQL allows you to access this area through a name using a cursor.

Cursors that you use in PL/SQL are of two types:

- Implicit cursor
- Explicit cursor

Implicit Cursor

PL/SQL declares an implicit cursor for every DML command, and queries that return a single row. The name of the implicit cursor is *SQL*. You can directly use this cursor without any declaration. We will see more about this later in this chapter.

Explicit Cursor

PL/SQL's implicit cursor can handle only single-row queries. If you ever need to select more than one row using SELECT in PL/SQL then you have to use explicit cursor.

When do we need an explicit cursor?

SELECT command in PL/SQL block can retrieve only one row. If SELECT command retrieves no row then NO_DATA_FOUND exception will be raised. If SELECT retrieves more than one row then TOO_MANY_ROWS exception occurs.

So, a SELECT command will succeed only when it retrieves a single row. The reason for this is; SELECT command copies the values of columns that it retrieved into variables. If multiple rows are retrieved then multiple values for each column are to be copied to a single variable and that creates the problem.

```
declare
      v_ccode  varchar2(5);
      v_fee    number(5);
begin

      select   ccode,fee  into   v_ccode, v_fee
      from  courses
      where  duration > 25;

end;
```

SELECT command in the above example will raise TOO_MANY_ROWS exception if more than one course is having duration more than 25.

An explicit cursor is the solution to the problem. A cursor can store a collection of records retrieved by a query. Then it allows us to fetch one record from cursor at a time and thereby enabling to process all the records in the cursor.

As you can see in figure 1, SELECT retrieves rows from database into cursor. Cursor stores the collection of record retrieved by SELECT. Then the program can fetch one row at a time from cursor and apply the required process to it.

SELECT command given at the time of declaring the cursor is used to retrieve the data from database. Records in the cursor will be fetched one at a time using FETCH statement, which fetches the data from current row of the cursor and copies the data into variables.

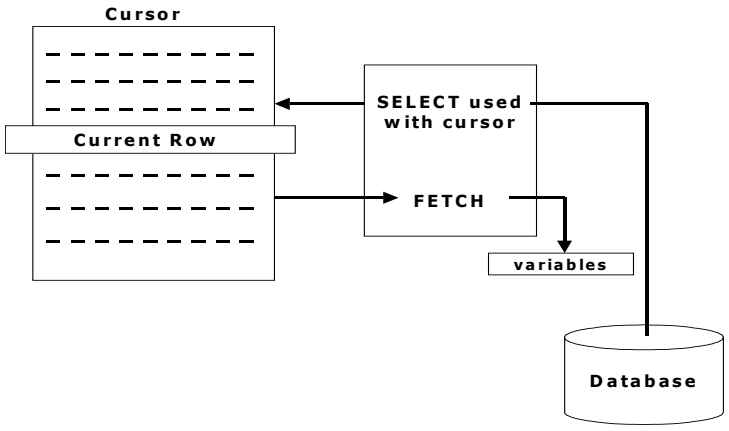

Figure 1: An explicit cursor.

Handling explicit cursor

Explicit cursor is a name used to refer to an area where you can place multiple rows retrieved by SELECT. You must use an explicit cursor whenever you have to use a multi-row query in PL/SQL.

The following are the steps required to create and use an explicit cursor:

- Declare the cursor in Declare section
- Open the cursor using OPEN statement in Executable part
- Fetch one row at a time using FETCH statement.
- Close the cursor after all the records in the cursor are fetched and processed by using CLOSE.

The following section will discuss each step in detail.

Declaring a cursor

A cursor is declared in Declare section using CURSOR statement. At the time of declaration the name of the cursor and the associated SELECT statement are mentioned.

```
CURSOR cursor_name [(parameter[, parameter]...)]
         IS select_statement
         [FOR UPDATE [OF column,column, . . . ];
```

The following example shows how to declare a cursor.

```
declare
  cursor course_det is
     select ccode, name, fee
     from   courses;
begin

end;
```

COURSE_DET is the name of the cursor, which will be populated with the rows retrieved by the given SELECT at the time of opening the cursor.

We will discuss more about parameters in declaration of cursor in the section "Input Arguments".

The following example declares a cursor that takes course code and number of batches of the course in the ascending order of number of batches.

Opening a cursor using OPEN statement

OPEN statement is used to execute the SELECT command associated with the cursor and place the rows retrieved by the query into cursor.

```
OPEN  cursor_name  [(input arguments)];
```

Cursor_name is the name of the cursor that is to be opened.

Input_arguments are the values to be passed to the parameters of the cursor. More on this later in the section "Input Arguments".

The following statement opens the cursor COURSE_DET and places the rows retrieved by the query into the cursor.

```
declare
  cursor course_det is
     select ccode, name, fee
     from   courses;

begin
     open course_det;

end;
```

Fetching rows using FETCH statement

Once cursor is opened using OPEN statement, cursor has a set of rows, which can be fetched using FETCH statement. FETCH statement takes the data of the current row in the cursor and copies the values of the columns into variables given after INTO keyword.

```
FETCH  cursor_name INTO  variable-1, variable-2, . . .;
```

For each column in the cursor there should be a corresponding variable in FETCH statement. Also make sure the data types of variables and corresponding columns are matching.

The following snippet demonstrates how to fetch and copy data from current row of the cursor to variables given after INTO.

```
declare
  cursor course_det is
     select ccode, name, fee
     from   courses;

     v_ccode  courses.ccode%type;
     v_name   courses.name%type;
     v_fee    courses.fee%type;
```

```
begin
     open course_det;
     loop
          fetch  course_det  into  v_ccode, v_name, v_fee;
          . . .

     end loop;

end;
```

FETCH statement is to be used inside the loop to repeatedly fetch rows from the cursor. The process of fetching should stop once all rows of the cursor are fetch (reached end of cursor).

The following code will show how to exit cursor when cursor is completely processed.

```
     loop
          fetch  course_det  into  v_ccode, v_name, v_fee;
          exit   when course_det%notfound;

     end loop;
```

NOTFOUND attribute of the cursor returns TRUE when previous FETCH doesn't successfully read a row from cursor.

We will discuss more about attributes of explicit and implicit cursors later in his chapter.

Closing a cursor using CLOSE command

CLOSE statement is used to close cursor after the cursor is processed. Closing a cursor would release the resources associated with cursor.

```
CLOSE cursor_name;
```

The following example closes COURSE_DET cursor:

```
declare
begin
   open ..
   loop
        ...

   end loop;

  close course_det;

end;
```

Cursor For Loop

In order to process a cursor, you can use cursor FOR loop to automate the following steps.

- Opening cursor
- Fetching rows from the cursor
- Terminating loop when all rows in the cursor are fetched
- Closing cursor

The following is the syntax of cursor for loop. This for loop is specifically meant to process cursors.

```
   FOR rowtype_variable IN  cursor_name
   LOOP
            Statements;
   END LOOP;
```

rowtype_variable is automatically declared by cursor for loop. It is of ROWTYPE of the cursor. It has columns of the cursor as fields. These fields can be accessed using *rowtype_variable.fieldname.*

The following example shows the process involved without using cursor for loop and using for loop.

```
DECLARE
   cursor  courses_cursor   is
        select  ccode,fee
        from   courses;
BEGIN
   -- open cursor
   open courses_cursor;

   loop
        fetch  courses_cursor into v_ccode, v_fee;
        -- if previous fetch failed then exit loop
        -- NOTFOUND attribute of the cursor return true if
        -- previous fetch has failed.
        exit when  courses_cursor%notfound;

        -- process the record fetched from cursor
        . . .
   end loop;

  close  courses_cursor;
END;
```

The same program can also be written using cursor for loop as follows:

```
declare
   cursor  courses_cursor   is
        select  ccode,fee
        from   courses;
begin
      -- cursor is opened and one row at a time is fetched
      -- loop is automatically terminated if all records are fetched


      for rec in  courses_cursor
      loop
        -- process the record fetched from cursor
        if  rec.fee > 5000 then
             -- do something here
        end if;

      end loop;

end;
```

The following are the important steps in the above program:

- Cursor COURSES_CURSOR is automatically opened by cursor for loop.
- REC is declared automatically by cursor for loop as:

  REC courses_cursor%ROWTYPE;

  But REC is available only inside the cursor for loop. It contains the same columns as the cursor. In order to access a column of the current row of the cursor, in the cursor for loop, use the format:

  rowtype_variable.columnname
- Statements in the cursor for loop are executed once for each row of the cursor. And for each row of the cursor, row is copied into rowtype_variable.
- Loop is terminated once end of cursor is reached. And cursor is closed.

The following section will summarize what we have seen so far by showing how to write a simple program using cursor.

Sample program

Let us now put all pieces together and write a complete program that makes use of cursor.

The sample program is to update FEE of each course based on the following table.

No. of students	Percentage of change
> 10	10% increase
> 20	15% increase
Otherwise	5% decrease

```
/* Author : P.Srikanth
   Date   : 22-10-2001
   Place  : Vizag.
   Purpose: Sample cursor program to illustrate how to handle a cursor*/

declare
    cursor  courses_cursor  is
      select c.ccode, count(*) count
      from   batches b, students s
      where  c.ccode = b.ccode and b.bcode = s.bcode
      group  by c.ccode;

 v_per number(5,2);

 begin
         -- rec is automatically declared
         for rec in courses_cursor
         loop
            if  rec.count > 20 then
                  v_per:= 1.15;
            elsif  rec.count > 10 then
                  v_per := 1.10;
            else
                  v_per := 0.90;
            end if;

            -- update row in the table

            update courses set fee = fee * v_per
            where  ccode = rec.ccode;

         end loop;

end;
/
```

The above program is used to declare a cursor that takes course code and no. of students joined into that course using a SELECT that joins BATCHES and STUDENTS table. Depending upon the no. of students joined into each course, it updates the course fee of the course.

It uses cursor for loop to take one row at a time from COURSES_CURSOR and updates the FEE of COURSES table after the process.

Implicit cursor

Oracle implicitly opens a cursor to process each SQL statement not associated with an explicitly declared cursor. You can refer to this cursor using the name SQL.

You cannot use the `OPEN`, `FETCH`, and `CLOSE` statements with SQL cursor. But, you can use cursor attributes to get information about the most recently executed SQL statement.

The following example shows how to use implicit cursor to know whether the most recent UPDATE has updated any rows or not.

```
declare

begin
      update . . .
      if  SQL%NOTFOUND then
           statements;
      end if;
end;
```

NOTFOUND is an attribute of implicit cursor that returns true if previous UPDATE command has not affected any row.

Cursor attributes

Cursor attributes allow you to get information regarding cursor. For example, you can get the number of rows fetched so far from a cursor using ROWCOUNT attribute and whether a row is fetched or not using FOUND attribute.

Use the following syntax to access cursor attributes:

```
cursor_name%Attribute
```

Explicit cursor attributes

The following is the list of available cursor attributes and their meaning for explicit cursor.

Attribute	What it returns?
NOTFOUND	True, if previous fetch failed.
FOUND	True, if previous fetch succeeded.
ROWCOUNT	Number of rows fetched from cursor so far.
ISOPEN	True, if cursor is currently open

Table 1: Explicit Cursor Attributes.

Implicit cursor attributes

Cursor attributes do not have the same meaning for both explicit and implicit cursors. The following table shows the meaning of cursor attributes with implicit cursor.

Attribute	What it returns?
NOTFOUND	True, if previous DML operation didn't affect any row.
FOUND	True, if previous DML operation affected any row.
ROWCOUNT	Number of rows affected by the most recent DML operation.

Table 2: Implicit Cursor Attributes.

The following example shows how to use ROWCOUNT attribute with implicit cursor to know how many rows were updated with most recent UPDATE command.

```
begin
        update courses set   fee = fee & 1.1
        where  duration > 25;

        /*  if more than 5 rows are effected
            then rollback updation */

        if   SQL%ROWCOUNT  >  5 then
             rollback;
        else
             commit;
        end if;
end;
```

The following is how you use cursor attributes with explicit cursors. Attribute NOTFOUND returns true if previous FETCH statement couldn't fetch any row.

```
LOOP
        fetch  courses_cursor into  v_ccode, v_fee;

        /* exit loop if previous FETCH failed */

        exit  when    students_cursor%NOTFOUND;

       /* process the record fetched */
END LOOP;
```

In the above example EXIT is executed when NOTFOUND attribute of cursor CORUSES_CURSOR returns TRUE.

Input arguments to cursor

Input arguments are the values that are passed to cursor at the time of opening the cursor. These values are passed to parameters that are declared in cursor and they are used while executing the query.

The following example will illustrate the process.

```
declare
        cursor  batch_cursor ( pccode varchar2) is
        select * from batches where ccode = pccode;
```

```
begin

       -- open the cursor and get all batches of course oracle.
       open  batch_cursor('ora');
       -- process the cursor

       close batch_cursor;

       -- open the same cursor but with a different course code
       open  batch_cursor('vb');

       -- process the cursor, which contains details of batches of vb

      close batch_cursor;
end;
```

Cursor BATCH_CURSOR is declared to take a parameter or input argument – PCCODE. The parameter is used to retrieve details of the batches of the given course.

At the time of opening the cursor, the value for parameter PCCODE is to be passed by enclosing the value within parentheses. The value is passed to PCCODE, which is then used in the query of the cursor.

The advantage of the input arguments, as you can see in the example above, is that you can use the same cursor to represent different sets of records at different points of time. In the above example, first cursor contains the batches of Oracle course then that cursor is closed and reopened with course code VB to take the details of batches of VB.

In case of cursor for loop, values to input arguments are passed at the time of using cursor in for loop as follows:

```
for rec in batch_cursor('ora')
  loop
              ...
  end loop;
```

Note: *The size of data type of input arguments should not be given. That means, we should give only VARCHAR2 and not VARCHAR2 (10).*

FOR UPDATE and CURRENT OF

By default, Oracle locks rows while manipulating the rows. But it is possible to override default locking by using FOR UPDATE option of SELECT command.

FOR UPDATE option clause can be used with SELECT while declaring cursor to lock all records retrieved by cursor to make sure they are not locked by others before we update or delete them. As Oracle automatically locks rows for you, FOR UPDATE clause is required only when you want to lock rows ahead of update or delete - at the time of opening cursor.

CURRENT OF clause with UPDATE and DELETE commands can be used to refer to the current row in the cursor.

***Note**: FOR UPDATE must be given if you want to use CURRENT OF clause to refer to current row in the cursor.*

The following example shows how to use FOR UPDATE OF and CURRENT OF clauses:

```
DECLARE
   CURSOR course_details  IS
       SELECT ccode, name, fee from courses
       FOR UPDATE OF fee;
   ...
BEGIN
    OPEN course_details;

    LOOP
       FETCH  course_details INTO...
       ...
      /* Update the current record in the cursor
            course_details */

       UPDATE courses SET fee =  new_fee
          WHERE CURRENT OF course_details;
    END LOOP;

    CLOSE   course_details;

END;
```

Summary

A cursor is always used by PL/SQL to execute single-row queries and DML command. But, in order to use multi-row query, you have to use an explicit cursor. An explicit cursor contains a row-set, which is retrieved by multi-row query. Implicit cursor is used to get information about the most recent DML operation.

Cursor FOR loop is used to open, fetch rows until end of cursor, and close the cursor.

Input arguments of the cursor can be used to pass values to cursor at the time of opening cursor so that these values are used by SELECT command of the cursor. FOR UPDATE clause is used to override default locking and CURRENT OF is used to refer to current record in the cursor.

Exercises

1. Which attribute is used to find out how many rows were fetched from cursor so far.
2. Can we use ISOPEN attribute with implicit cursor.
3. How can we know whether the most recent DML operation has affected any row?
4. How do you declare an input arguments for the cursor and how do you pass value to it?
5. What is the use of CURRENT OF clause in DELETE and UPDATE commands?
6. Create table called COURSE_DETAILS with the columns:

   Course Code, No. of batches completed, Total amount collected so far