Explicit Cursors

An explicit cursor is defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row. We can provide a suitable name for the cursor.  
  
**The General Syntax for creating a cursor is as given below:**

*CURSOR cursor\_name IS select\_statement;*

* *cursor\_name – A suitable name for the cursor.*
* *select\_statement – A select query which returns multiple rows.*

**How to use Explicit Cursor?**

There are four steps in using an Explicit Cursor. 

* DECLARE the cursor in the declaration section.
* OPEN the cursor in the Execution Section.
* FETCH the data from cursor into PL/SQL variables or records in the Execution Section.
* CLOSE the cursor in the Execution Section before you end the PL/SQL Block.

1) Declaring a Cursor in the Declaration Section:

*DECLARE*

*CURSOR emp\_cur IS*

*SELECT \**

*FROM emp\_tbl*

*WHERE salary > 5000;*

      In the above example we are creating a cursor ‘emp\_cur’ on a query which returns the records of all the   
      employees with salary greater than 5000. Here ‘emp\_tbl’ in the table which contains records of all the   
      employees.

2) Accessing the records in the cursor:  
      Once the cursor is created in the declaration section we can access the cursor in the execution   
      section of the PL/SQL program.

**How to access an Explicit Cursor?**

These are the three steps in accessing the cursor.  
1) Open the cursor.  
2) Fetch the records in the cursor one at a time.  
3) Close the cursor.

General Syntax to open a cursor is:

*OPEN cursor\_name;*

General Syntax to fetch records from a cursor is:

*FETCH cursor\_name INTO record\_name;*

OR 

*FETCH cursor\_name INTO variable\_list;*

General Syntax to close a cursor is:

*CLOSE cursor\_name;*

When a cursor is opened, the first row becomes the current row. When the data is fetched it is copied to the record or variables and the logical pointer moves to the next row and it becomes the current row. On every fetch statement, the pointer moves to the next row. If you want to fetch after the last row, the program will throw an error. When there is more than one row in a cursor we can use loops along with explicit cursor attributes to fetch all the records.

Points to remember while fetching a row:

· We can fetch the rows in a cursor to a PL/SQL Record or a list of variables created in the PL/SQL Block.   
· If you are fetching a cursor to a PL/SQL Record, the record should have the same structure as the cursor.  
· If you are fetching a cursor to a list of variables, the variables should be listed in the same order in the fetch statement as the columns are present in the cursor.

General Form of using an explicit cursor is:

*DECLARE*

*variables;*

*records;*

*create a cursor;*

*BEGIN*

*OPEN cursor;*

*FETCH cursor;*

*process the records;*

*CLOSE cursor;*

*END;*

Lets Look at the example below  
  
Example 1:

*1> DECLARE*

*2> emp\_rec emp\_tbl%rowtype;*

*3> CURSOR emp\_cur IS*

*4> SELECT \**

*5> FROM*

*6> WHERE salary > 10;*

*7> BEGIN*

*8>  OPEN emp\_cur;*

*9>  FETCH emp\_cur INTO emp\_rec;*

*10>     dbms\_output.put\_line (emp\_rec.first\_name || '  ' || emp\_rec.last\_name);*

*11>  CLOSE emp\_cur;*

*12> END;*

In the above example, first we are creating a record ‘emp\_rec’ of the same structure as of table ‘emp\_tbl’ in line no 2. We can also create a record with a cursor by replacing the table name with the cursor name. Second, we are declaring a cursor ‘emp\_cur’ from a select query in line no 3 - 6. Third, we are opening the cursor in the execution section in line no 8. Fourth, we are fetching the cursor to the record in line no 9. Fifth, we are displaying the first\_name and last\_name of the employee in the record emp\_rec in line no 10. Sixth, we are closing the cursor in line no 11.

What are Explicit Cursor Attributes?

Oracle provides some attributes known as Explicit Cursor Attributes to control the data processing while using cursors. We use these attributes to avoid errors while accessing cursors through OPEN, FETCH and CLOSE Statements.

When does an error occur while accessing an explicit cursor?

a) When we try to open a cursor which is not closed in the previous operation.  
b) When we try to fetch a cursor after the last operation.

These are the attributes available to check the status of an explicit cursor.

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Return values** | **Example** |
| %FOUND | TRUE, if fetch statement returns at least one row. | Cursor\_name%FOUND |
| FALSE, if fetch statement doesn’t return a row. |
| %NOTFOUND | TRUE, , if fetch statement doesn’t return a row. | Cursor\_name%NOTFOUND |
| FALSE, if fetch statement returns at least one row. |
| %ROWCOUNT | The number of rows fetched by the fetch statement | Cursor\_name%ROWCOUNT |
| If no row is returned, the PL/SQL statement returns an error. |
| %ISOPEN | TRUE, if the cursor is already open in the program | Cursor\_name%ISNAME |
| FALSE, if the cursor is not opened in the program. |

Using Loops with Explicit Cursors:

Oracle provides three types of cursors namely SIMPLE LOOP, WHILE LOOP and FOR LOOP. These loops can be used to process multiple rows in the cursor. Here I will modify the same example for each loops to explain how to use loops with cursors.

**Cursor with a Simple Loop:**

*1> DECLARE*

*2> CURSOR emp\_cur IS*

*3> SELECT first\_name, last\_name, salary FROM emp\_tbl;*

*4> emp\_rec emp\_cur%rowtype;*

*5> BEGIN*

*6> IF NOT sales\_cur%ISOPEN THEN*

*7> OPEN sales\_cur;*

*8> END IF;*

*9> LOOP*

*10> FETCH emp\_cur INTO emp\_rec;*

*11> EXIT WHEN emp\_cur%NOTFOUND;*

*12> dbms\_output.put\_line(emp\_cur.first\_name || ' ' ||emp\_cur.last\_name*

*13> || ' ' ||emp\_cur.salary);*

*14> END LOOP;*

*15> END;*

*16> /*

In the above example we are using two cursor attributes %ISOPEN and %NOTFOUND.   
In line no 6, we are using the cursor attribute %ISOPEN to check if the cursor is open, if the condition is true the program does not open the cursor again, it directly moves to line no 9.   
In line no 11, we are using the cursor attribute %NOTFOUND to check whether the fetch returned any row. If there is no rows found the program would exit, a condition which exists when you fetch the cursor after the last row, if there is a row found the program continues.

We can use %FOUND in place of %NOTFOUND and vice versa. If we do so, we need to reverse the logic of the program. So use these attributes in appropriate instances.

**Cursor with a While Loop:**

Lets modify the above program to use while loop.

*1> DECLARE*

*2> CURSOR emp\_cur IS*

*3> SELECT first\_name, last\_name, salary FROM emp\_tbl;*

*4> emp\_rec emp\_cur%rowtype;*

*5> BEGIN*

*6> IF NOT sales\_cur%ISOPEN THEN*

*7> OPEN sales\_cur;*

*8> END IF;*

*9> FETCH sales\_cur INTO sales\_rec;*

*10> WHILE sales\_cur%FOUND THEN*

*11> LOOP*

*12> dbms\_output.put\_line(emp\_cur.first\_name || ' ' ||emp\_cur.last\_name*

*13> || ' ' ||emp\_cur.salary);*

*15> FETCH sales\_cur INTO sales\_rec;*

*16> END LOOP;*

*17> END;*

*18> /*

In the above example, in line no 10 we are using %FOUND to evaluate if the first fetch statement in line no 9 returned a row, if true the program moves into the while loop. In the loop we use fetch statement again (line no 15) to process the next row. If the fetch statement is not executed once before the while loop the while condition will return false in the first instance and the while loop is skipped. In the loop, before fetching the record again, always process the record retrieved by the first fetch statement, else you will skip the first row.

**Cursor with a FOR Loop:**

When using FOR LOOP you need not declare a record or variables to store the cursor values, need not open, fetch and close the cursor. These functions are accomplished by the FOR LOOP automatically.

**General Syntax for using FOR LOOP:**

*FOR record\_name IN cusror\_name*

*LOOP*

*process the row...*

*END LOOP;*

Let’s use the above example to learn how to use for loops in cursors.

*1> DECLARE*

*2> CURSOR emp\_cur IS*

*3> SELECT first\_name, last\_name, salary FROM emp\_tbl;*

*4> emp\_rec emp\_cur%rowtype;*

*5> BEGIN*

*6> FOR emp\_rec in sales\_cur*

*7> LOOP*

*8> dbms\_output.put\_line(emp\_cur.first\_name || ' ' ||emp\_cur.last\_name*

*9> || ' ' ||emp\_cur.salary);*

*10> END LOOP;*

*11>END;*

*12> /*

In the above example, when the FOR loop is processed a record ‘emp\_rec’of structure ‘emp\_cur’ gets created, the cursor is opened, the rows are fetched to the record ‘emp\_rec’ and the cursor is closed after the last row is processed. By using FOR Loop in your program, you can reduce the number of lines in the program.

**NOTE:** In the examples given above, we are using backward slash ‘/’ at the end of the program. This indicates the oracle engine that the PL/SQL program has ended and it can begin processing the statements.