ORACLETUTORIAL.COM

PL/SQL Cursor FOR LOOP

Summary: in this tutorial, you will learn how to use the PL/SQL cursor `FOR LOOP` statement to fetch and process every record from a cursor.

Introduction to PL/SQL cursor FOR LOOP statement

The cursor `FOR LOOP` statement is an elegant extension of the numeric `FOR LOOP` (https://www.oracletutorial.com/plsql-tutorial/plsql-for-loop/) statement.

The numeric `FOR LOOP` (https://www.oracletutorial.com/plsql-tutorial/plsql-for-loop/) executes the body of a loop once for every integer value in a specified range. Similarly, the cursor `FOR LOOP` executes the body of the loop once for each row returned by the query (https://www.oracletutorial.com/oracle-basics/oracle-select/) associated with the cursor.

A nice feature of the cursor `FOR LOOP` statement is that it allows you to fetch every row from a cursor without manually managing the execution cycle i.e., `OPEN`, `FETCH`, and `CLOSE`.

The cursor `FOR LOOP` implicitly creates its loop index as a record (https://www.oracletutorial.com/plsql-tutorial/plsql-record/) variable with the row type in which the cursor returns and then opens the cursor.

In each loop iteration, the cursor `FOR LOOP` statement fetches a row from the result set into its loop index. If there is no row to fetch, the cursor `FOR LOOP` closes the cursor.

The cursor is also closed if a statement inside the loop transfers control outside the loop, e.g., `EXIT` and `GOTO` (https://www.oracletutorial.com/plsql-tutorial/plsql-goto/), or raises an exception (https://www.oracletutorial.com/plsql-tutorial/plsql-raise/).

The following illustrates the syntax of the cursor `FOR LOOP` statement:

```
FOR record IN cursor_name
LOOP
```

```
    process_record_statements;
END LOOP;
```

1) record

The `record` is the name of the index that the cursor `FOR LOOP` statement declares implicitly as a `%ROWTYPE` record variable of the type of the cursor.

The `record` variable is local to the cursor `FOR LOOP` statement. It means that you can only reference it inside the loop, not outside. After the cursor `FOR LOOP` statement execution ends, the `record` variable becomes undefined.

2) cursor_name

The `cursor_name` is the name of an explicit cursor that is not opened when the loop starts.

Note that besides the cursor name, you can use a `SELECT` (https://www.oracletutorial.com/oracle-basics/oracle-select/) statement as shown below:

```
FOR record IN (select_statement)
LOOP
    process_record_statements;
END LOOP;
```

In this case, the cursor `FOR LOOP` declares, opens, fetches from, and closes an implicit cursor. However, the implicit cursor is internal; therefore, you cannot reference it.

Note that Oracle Database automatically optimizes a cursor `FOR LOOP` to work similarly to a `BULK COLLECT` query. Although your code looks as if it fetched one row at a time, Oracle Database fetches multiple rows at a time and allows you to process each row individually.

PL/SQL cursor FOR LOOP examples

Let's look at some examples of using the cursor `FOR LOOP` statement to see how it works.

A) PL/SQL cursor FOR LOOP example

The following example declares an explicit cursor and uses it in the cursor `FOR LOOP` statement.

```
DECLARE
  CURSOR c_product
  IS
    SELECT
        product_name, list_price
    FROM
        products
    ORDER BY
        list_price DESC;
BEGIN
  FOR r_product IN c_product
  LOOP
    dbms_output.put_line( r_product.product_name || ': $' ||  r_product.list_pric
  END LOOP;
END;
```

In this example, the `SELECT` statement of the cursor retrieves data from the `products` table. The `FOR LOOP` statement opened, fetched each row in the result set, displayed the product information, and closed the cursor.

B) Cursor FOR LOOP with a SELECT statement example

The following example is equivalent to the example above but uses a query in a cursor `FOR LOOP` statement.

```
BEGIN
  FOR r_product IN (
        SELECT
            product_name, list_price
        FROM
            products
```

```
        ORDER BY list_price DESC
    )
  LOOP
    dbms_output.put_line( r_product.product_name ||
      ': $' ||
      r_product.list_price );
  END LOOP;
END;
```

In this tutorial, you have learned how to use the PL/SQL cursor `FOR LOOP` to fetch data from a cursor.