GALGOTIAS UNIVERSITY

School of Computing Science and Engineering

Program: BCA - IOP

Course Code: BCAS3031

Course Name: PL/SQL & Cursors and Triggers

Dr. T. Poongodi

Associate Professor

- A Cursor is a pointer to the context area.
- Oracle creates context area for processing an SQL statement which contains all information about the statement.
- PL/SQL allows the programmer to control the context area through the cursor.
- A cursor holds the rows returned by the SQL statement.
- The set of rows the cursor holds is referred as active set.
- These cursors can also be named so that they can be referred from another place of the code.

- Implicit Cursor
- Explicit Cursor
- Cursor Attributes
- FOR Loop Cursor statement

The cursor is of two types.

- Implicit Cursor
- Explicit Cursor

Implicit Cursor

- Whenever any DML operations occur in the database, an implicit cursor is created that holds the rows affected, in that particular operation.
- These cursors cannot be named and, hence they cannot be controlled or referred from another place of the code.
- We can refer only to the most recent cursor through the cursor attributes.

Explicit Cursor

- Programmers are allowed to create named context area to execute their DML operations to get more control over it.
- The explicit cursor should be defined in the declaration section of the PL/SQL block, and it is created for the 'SELECT' statement that needs to be used in the code.

Steps that involved in working with explicit cursors.

1.Declaring the cursor

- Declaring the cursor simply means to create one named context area for the 'SELECT' statement that is defined in the declaration part.
- The name of the context area is same as the cursor name.

2.Opening Cursor

- Opening the cursor will instruct the PL/SQL to allocate the memory for this cursor.
- It will make the cursor ready to fetch the records.

3. Fetching Data from the Cursor

- The 'SELECT' statement is executed and the rows fetched is stored in the allocated memory. These are now called as **active sets**.
- Fetching data from the cursor is a record-level activity that means we can access the data in a record-by-record way.
- Each fetch statement will fetch one active set and holds the information of that particular record.
- This statement is same as 'SELECT' statement that fetches the record and assigns to the variable in the 'INTO' clause.

4. Closing the Cursor

Once all the record is fetched now, close the cursor so that the memory allocated to this context area will be released.

```
DECLARE
CURSOR <cursor_name> IS <SELECT statement^>
<cursor_variable declaration>
BEGIN
OPEN <cursor_name>;
FETCH <cursor_name> INTO <cursor_variable>;
.
.
CLOSE <cursor_name>;
END;
```

- The declaration part contains the declaration of the cursor and the cursor variable in which the fetched data will be assigned.
- The cursor is created for the 'SELECT' statement that is given in the cursor declaration.
- In execution part, the declared cursor is opened, fetched and closed.

Cursor Attributes

Both Implicit cursor and the explicit cursor has certain attributes that can be accessed. These attributes give more information about the cursor operations.

Cursor Attribute	Description
%FOUND	It returns the Boolean result 'TRUE' if the most recent fetch operation fetched a record successfully, else it will return FALSE.
%NOTFOUND	This works oppositely to %FOUND it will return 'TRUE' if the most recent fetch operation could not able to fetch any record.
%ISOPEN	It returns Boolean result 'TRUE' if the given cursor is already opened, else it returns 'FALSE'
%ROWCOUNT	It returns the numerical value. It gives the actual count of records that got affected by the DML activity.

Example 1: How to declare, open, fetch and close the explicit cursor.

Project all the employee's name from emp table using a cursor. Use cursor attribute to set the loop to fetch all the record from the cursor.

```
DECLARE
CURSOR sample IS SELECT emp_name FROM emp;
lv_emp_name emp.emp_name%type;

BEGIN
OPEN sample;

LOOP
FETCH sample INTO lv_emp_name;
IF sample%NOTFOUND
THEN
EXIT;
END IF;
Dbms_output.put_line('Employee Fetched:'||lv_emp_name);
END LOOP;

Dbms_output.put_line('Total rows fetched is'||sample%R0WCOUNT);
CLOSE sample;
END;
/
```

Output

Employee Fetched:BBB

Employee Fetched:XXX

Employee Fetched:YYY

Total rows fetched is 3

Code Explanation:

Code line 2: Declaring the cursor sample for statement 'SELECT emp_name FROM emp'.

Code line 3: Declaring variable lv_emp_name.

Code line 5: Opening the cursor sample.

Code line 6: Setting the Basic loop statement to fetch all the records in the 'emp' table.

Code line 7: Fetches the sample data and assign the value to lv_emp_name.

Code line 9: Using the cursor attribute '%NOTFOUND' to find whether all the record in the cursor is fetched. If fetched then it will return 'TRUE' and control will exit from the loop, else the control will keep on fetching the data from the cursor and print the data.

Code line 11: EXIT condition for the loop statement.

Code line 12: Print the fetched employee name.

Code line 14: Using the cursor attribute '%ROWCOUNT' to find the total number of records that got affected/fetched in the cursor.

Code line 15: After exiting from the loop the cursor is closed and the memory allocated is set free.

FOR Loop Cursor statement

- "FOR LOOP" statement can be used for working with cursors.
- Give the cursor name instead of range limit in the FOR loop statement so that the loop will work from the first record of the cursor to the last record of the cursor.
- The cursor variable, opening of cursor, fetching and closing of the cursor will be done implicitly by the FOR loop.

```
DECLARE
CURSOR <cursor_name> IS <SELECT statement>;
BEGIN
  FOR I IN <cursor_name>
  LOOP
  .
  .
  END LOOP;
END;
```

- The declaration part contains the declaration of the cursor.
- The cursor is created for the 'SELECT' statement that is given in the cursor declaration.
- In execution part, the declared cursor is setup in the FOR loop and the loop variable 'I' will behave as cursor variable in this case.

Project all the employee name from emp table using a cursor-FOR loop.

```
DECLARE
CURSOR sample IS SELECT emp_name FROM emp;
BEGIN
FOR lv_emp_name IN sample
LOOP
Dbms_output.put_line('Employee Fetched:'||lv_emp_name);
END LOOP;
END;
/
```

Employee Fetched:BBB

Employee Fetched:XXX

Employee Fetched:YYY

Code Explanation:

- Code line 2: Declaring the cursor sample for statement 'SELECT emp_name FROM emp'.
- Code line 4: Constructing the 'FOR' loop for the cursor with the loop variable lv_emp_name.
- Code line 5: Printing the employee name in each iteration of the loop.
- Code line 8: Exit the loop

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 customer.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;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
  END LOOP;
  CLOSE c_customers;
END;
/
```

1	Ramesh	Ahmedabad
2	Khilan	Delhi
3	kaushik	Kota
4	Chaitali	Mumbai
5	Hardik	Bhopal
6	Komal	MP

PL/SQL procedure successfully completed.

```
SQL> select * from student1;

    ROLLNO SNAME                AGE COURSE
---------- -------------------- ---------- --------------------
        11 Anu                  20 bca
        12 Anjali               20 bcaiop
        13 Ankur                 19 bcaiop
        14 Ankit                19 bcaiop
        15 Nitin                19 bcaiop

SQL>
```

Run SQL Command Line

```
PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> DECLARE
  2  CURSOR student_cursor IS SELECT sname FROM Student1 ;
  3  snm Student1.sname%type;
  4  BEGIN
  5  OPEN student_cursor;
  6  IF student_cursor%ISOPEN=FALSE then
  7  dbms_output.put_line('Cannot open cursor');
  8  ELSE
  9  LOOP
 10  FETCH student_cursor INTO snm;
 11  IF student_cursor%NOTFOUND then
 12  Exit;
 13  END IF;
 14  dbms_output.put_line('' ||snm);
 15  END LOOP;
 16  dbms_output.put_line('Total Records: ' ||student_cursor%rowcount);
 17  CLOSE student_cursor;
 18  END IF;
 19  END;
 20  /
Anu
Anjali
Ankur
Ankit
Nitin
Total Records: 5

PL/SQL procedure successfully completed.

SQL> select * from student1;

    ROLLNO SNAME                     AGE COURSE
---------- -------------------- ---------- --------------------
        11 Anu                          20 bca
        12 Anjali                       20 bcaiop
        13 Ankur                        19 bcaiop
        14 Ankit                        19 bcaiop
        15 Nitin                        19 bcaiop

SQL>
```

```
SQL> DECLARE
  2  CURSOR student_cursor IS SELECT sname FROM Student1 ;
  3  snm Student1.sname%type;
  4  BEGIN
  5  OPEN student_cursor;
  6  IF student_cursor%ISOPEN=FALSE then
  7  dbms_output.put_line('Cannot open cursor');
  8  ELSE
  9  LOOP
 10   FETCH student_cursor INTO snm;
 11   IF student_cursor%NOTFOUND then
 12   Exit;
 13   END IF;
 14   dbms_output.put_line('' ||snm);
 15   END LOOP;
 16   dbms_output.put_line('Total Records: ' ||student_cursor%rowcount);
 17   CLOSE student_cursor;
 18   END IF;
 19   END;
 20   /
```

Anu
Anjali
Ankur
Ankit
Nitin
Total Records: 5

PL/SQL procedure successfully completed.

```
SQL> create table products(product_name varchar2(20),
list_price number(6,2));

Table created.

SQL> insert into products values('Mobile',6000);

1 row created.

SQL> insert into products values('Watch',7000);

1 row created.
```

```
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_price );
  END LOOP;
END;
```

```
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;
```

Run SQL Command Line

```
1 row created.

SQL> DECLARE
  2    CURSOR c_product
  3    IS
  4      SELECT
  5          product_name, list_price
  6      FROM
  7          products
  8      ORDER BY
  9          list_price DESC;
 10  BEGIN
 11    FOR r_product IN c_product
 12    LOOP
 13      dbms_output.put_line( r_product.product_name || ': $' ||  r_product.list_price );
 14    END LOOP;
 15  END;
 16  /
Watch: $7000
Mobile: $6000

PL/SQL procedure successfully completed.

SQL> BEGIN
  2    FOR r_product IN (
  3          SELECT
  4              product_name, list_price
  5          FROM
  6              products
  7          ORDER BY list_price DESC
  8      )
  9    LOOP
 10       dbms_output.put_line( r_product.product_name ||
 11          ': $' ||
 12          r_product.list_price );
 13    END LOOP;
 14  END;
 15  /
Watch: $7000
Mobile: $6000

PL/SQL procedure successfully completed.

SQL>
```


Thank You