PL/SQL Cursors and Triggers

Gautam Kumar

March 17, 2021

Cursors

- When an SQL statement is processed, Oracle creates a memory area known as context area. A **cursor** is a pointer to this context area.
- In other words, a cursor is a temporary work area created in the system memory when a SQL statement is executed.
- PL/SQL allows the programmer to control the context area through the cursor.
- A cursor holds the rows (one or more) returned by a SQL statement.
- Cursor is used to referred to a program to fetch and process the rows returned by the SQL statement, one at a time.
- There are two types of cursors:

    1. Implicit Cursors
    2. Explicit Cursors

Implicit Cursors

- Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement.

- Programmers cannot control the implicit cursors and the information in it.

- Oracle provides some attributes known as Implicit cursor's attributes to check the status of DML operations. Some of them are: %FOUND, %NOTFOUND, %ROWCOUNT and %ISOPEN.

Cursor Attributes

Attribute	Description
%FOUND	Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect at least one row or more rows or a SELECT INTO statement returned one or more rows. Otherwise it returns FALSE.
%NOTFOUND	Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect no row, or a SELECT INTO statement return no rows. Otherwise it returns FALSE. It is a just opposite of %FOUND.
%ISOPEN	It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL statements.
%ROWCOUNT	It returns the number of rows affected by DML statements like INSERT, DELETE, and UPDATE or returned by a SELECT INTO statement.

Implicit Cursor Example (customers table)

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allahabad	20000
2	Suresh	22	Kanpur	22000
3	Mahesh	24	Ghaziabad	24000
4	Chandan	25	Noida	26000
5	Alex	21	Paris	28000
6	Sunita	20	Delhi	30000

Query: Increase salary of each customer by 5000

Query: Increase salary of each customer by 5000

```
1.  DECLARE
2.     total_rows number(2);
3.  BEGIN
4.     UPDATE  customers
5.     SET salary = salary + 5000;
6.     IF sql%notfound THEN
7.        dbms_output.put_line('no customers updated');
8.     ELSIF sql%found THEN
9.        total_rows := sql%rowcount;
10.       dbms_output.put_line( total_rows || ' customers updated ');
11.    END IF;
12. END;
13. /
```

Output:

6 customers updated

PL/SQL procedure successfully completed.

Result of *select * from customers;*

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allahabad	25000
2	Suresh	22	Kanpur	27000
3	Mahesh	24	Ghaziabad	29000
4	Chandan	25	Noida	31000
5	Alex	21	Paris	33000
6	Sunita	20	Delhi	35000

Explicit Cursors

- The Explicit cursors are defined by the programmers to gain more control over the context area.
- These cursors should be defined in the declaration section of the PL/SQL block.
- It is created on a SELECT statement which returns more than one row.
- Syntax to create an explicit cursor: CURSOR cursor_name IS select_statement;
- Working with an explicit cursor includes the following steps −

  1. Declaring the cursor for initializing the memory
  2. Opening the cursor for allocating the memory
  3. Fetching the cursor for retrieving the data
  4. Closing the cursor to release the allocated memory

Working with an explicit cursor

1. Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement.

For example − **CURSOR c_customers IS SELECT id, name, address FROM customers;**

2. Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows − **OPEN c_customers;**

3. Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows − **FETCH c_customers INTO c_id, c_name, c_addr;**

4. Closing the Cursor

Closing the cursor means releasing the allocated memory. For example − **CLOSE c_customers;**

Syntax of creating Explicit Cursor

1. <cursor_variable declaration>
2. DECLARE CURSOR <cursor_name> IS <SELECT statement>
3. BEGIN OPEN <cursor_name>;
4. FETCH <cursor_name> INTO <cursor_variable>; . . CLOSE <cursor_name>;
5. END;

- In the above syntax, 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.

Example to illustrate the concepts of explicit cursors

```
1.  DECLARE
2.   c_id customers.id%type;
3.   c_name customer.name%type;
4.   c_addr customers.address%type;
5.   CURSOR c_customers IS
6.          SELECT id, name, address FROM customers;
7.  BEGIN
8.      OPEN c_customers;
9.      LOOP
10.     FETCH c_customers into c_id, c_name, c_addr;
11.         EXIT WHEN c_customers%notfound;
12.         dbms_output.put_line(c_id || ' ' || c_name || ' ‘ || c_addr);
13.     END LOOP;
14.     CLOSE c_customers;
15. END;
16.  /
```

Output:

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

PL/SQL procedure successfully completed.

PL/SQL Trigger

- Trigger is invoked by Oracle engine automatically whenever a specified event occurs. Trigger is stored into database and invoked repeatedly, when specific condition match.

- Triggers are stored programs, which are automatically executed or fired when some event occurs.

- Triggers are written to be executed in response to any of the following events.

    1. A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).

    2. A database definition (DDL) statement (CREATE, ALTER, or DROP).

    3. A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Classification of Triggers

- Classification based on the **timing**
  - BEFORE Trigger: It fires before the specified event has occurred.
  - AFTER Trigger: It fires after the specified event has occurred.
  - INSTEAD OF Trigger: It is used when any DML event is going to occur on the complex view
- Classification based on the **level**
  - STATEMENT level Trigger: It fires one time for the specified event statement.
  - ROW level Trigger: It fires for each record that got affected in the specified event. (only for DML)
- Classification based on the **Event**
  - DML Trigger: It fires when the DML event is specified (INSERT/UPDATE/DELETE)
  - DDL Trigger: It fires when the DDL event is specified (CREATE/ALTER)
  - DATABASE Trigger: It fires when the database event is specified (LOGON/LOGOFF/STARTUP/SHUTDOWN)

Creating a trigger:

```
1.  CREATE [OR REPLACE ] TRIGGER trigger_name          -- It creates or replaces an existing trigger with the trigger_name.
2.  {BEFORE | AFTER | INSTEAD OF }  -- This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
3.  {INSERT [OR] | UPDATE [OR] | DELETE}               -- This specifies the DML operation.
4.  [OF col_name]                                      -- This specifies the column name that would be updated.
5.  ON table_name                                      --This specifies the name of the table associated with the trigger.
6.  [REFERENCING OLD AS o NEW AS n]                    --This allows you to refer new and old values for various DML statements
7.   [FOR EACH ROW]                                    --This specifies a row level trigger
8.  WHEN (condition)                  --This provides a condition for rows for which the trigger would fire.
9.  DECLARE
10.         Declaration-statements
11. BEGIN
12.         Executable-statements
13. EXCEPTION
14.         Exception-handling-statements
15. END;
```

Trigger Example: **Student** table

Student

Student_id	Name	Address	Marks
1	Billie	NY	220
2	Eilish	London	190
3	Ariana	Miami	180

Query: To create a **trigger** that will add 100 marks to each new row of the *Marks* column whenever a new student is inserted to the table.

Example of Triger

```
1. CREATE TRIGGER Add_marks
2. BEFORE
3. INSERT
4. ON Student
5. FOR EACH ROW
6. SET new.Marks = new.Marks + 100;
```

The ***new*** keyword refers to the row that is getting affected.

Output:
Trigger created.

Query for inserting a new student

After creating the trigger, write the **query for inserting a new student** in the database.

```
INSERT INTO Student(Name, Address, Marks) VALUES('Alizeh', 'Maldives', 110);
```

To see the final output the query would be:

```
SELECT * FROM Student;
```

Student

Student_id	Name	Address	Marks
1	Billie	NY	220
2	Eilish	London	190
3	Ariana	Miami	180
4	Alizeh	Maldives	210

Enabling and Disabling Triggers

- To enable or disable the trigger, an ALTER (DDL) statement needs to be given for the trigger that disable or enable it.
  - Syntax for enabling/disabling the triggers.
    - ALTER TRIGGER <trigger_name> [ENABLE|DISABLE];
    - ALTER TABLE <table_name> [ENABLE|DISABLE] ALL TRIGGERS;

Advantages of Triggers

1. Triggers provide a way to check the integrity of the data. When there is a change in the database the triggers can adjust the entire database.

2. Enforces referential integrity

3. Event logging and storing information on table access

4. Imposing security authorizations

5. Preventing invalid transactions

Disadvantages of Triggers

- Triggers may be difficult to troubleshoot as they execute automatically in the database. If there is some error then it is hard to find the logic of trigger because they are fired before or after updates/inserts happen.

- The triggers may increase the overhead of the database as they are executed every time any field is updated.

References

1. https://tutorialink.com/dbms/cursors.dbms

2. https://www.tutorialspoint.com/plsql/plsql_cursors.htm

3. https://www.guru99.com/triggers-pl-sql.html

Questions

1. When a DML statement is executed, in which cursor attributes, the outcome of the statement is saved?

2. Why is %ISOPEN always false for an implicit cursor?

3. Explain the difference in the execution of triggers and stored procedures?

4. Explain the difference between Triggers and Constraints?