SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(IoT and Cybersecurity Including BCT)

COURSE NAME : 19SB504 DATABASE MANAGEMENT SYSTEMS

III YEAR / V SEMESTER

Unit II- **SQL**

Topic :Cursors, Stored Procedures, Triggers

Cursor in SQL

A cursor is a temporary workstation that is allocated by the database server during the execution of a statement.

It is a database object that allows us to access data of one row at a time. This concept of SQL is useful when the user wants to update the rows of the table one by one.

The cursor in SQL is the same as the looping technique of other programming languages. The collection of tuples held by the cursor is known as the active set.

In SQL database systems, users define the cursor using DECLARE statement and take the SELECT statement as the parameter, which helps in returning the set of rows.

Types of Cursor in SQL

Following are the two types of Cursor in Structured Query Language:

1. Implicit Cursor
2. Explicit Cursor

Implicit Cursor

These types of cursors are generated and allocated by the SQL server when the system performs INSERT, DELETE, and UPDATE operations on SQL queries.

This cursor is also referred to as the default cursor in SQL.

An implicit cursor is also created by the system when the SELECT query selects the single row.

Explicit Cursor

These types of cursors are created by the user using the SELECT query.

An explicit cursor holds multiple records but processes a single row at a time. It uses the pointer, which moves to another row after reading one row.

It is basically used for gaining extra control over the temporary workstation.

Life Cycle of Cursor

The life cycle of the cursor is described into the following five stages:

1. Declare a Cursor
2. Open Cursor
3. Fetch Data from Cursor
4. Close Cursor Connection
5. Deallocate cursor

1. Declare a Cursor

First, we have to declare the cursor by using the following SQL syntax:

1.DECLARE Cursor_Name **CURSOR FOR** Select_Statement;

2. Open Cursor

It is the second stage that opens the cursor for storing the data retrieved from the result set.

SYNTAX

OPEN Cursor_Name;

3. Fetch Cursor

It is the third stage in the cursor life cycle that fetches the rows for performing the insertion, deletion, and updation operations on the currently active tuple in the cursor.

Following are the six options that are used in syntax for fetching data from the cursor:

1. **FETCH FIRST FROM** Cursor_Name;
2. **FETCH LAST FROM** Cursor_Name;
3. **FETCH NEXT FROM** Cursor_Name;
4. **FETCH PRIOR FROM** Cursor_Name;
5. **FETCH ABSOLUTE** n **FROM** Cursor_Name;
6. **FETCH RELATIVE** n **FROM** Cursor_Name;

4. Close Cursor

It is the fourth stage in the process of the cursor. When we complete the work with the cursor, we have to close the cursor in this stage. We can close the cursor in SQL by using the following query:

CLOSE Cursor_Name;

5. Deallocate Cursor

It is the last and fifth stage of the cursor life cycle. In this part, we have to erase the definition of the cursor and discharge all the system resources combined with the cursor.

Basic Type of Cursor

Following are the four basic types of cursor in Structured Query Language:

1. STATIC Cursor
2. Forward Only cursor
3. KEYSET Driven Cursor
4. Dynamic Cursor

Stored Procedure

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Stored Procedure Syntax

CREATE PROCEDURE *procedure_name*
AS
sql_statement
GO;

Execute a Stored Procedure Syntax

```
EXEC procedure_name;
```

Stored Procedure Example

The following SQL statement creates a stored procedure named "SelectAllCustomers" that selects all records from the "Customers" table:

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;

EXEC SelectAllCustomers;
```

Triggers in SQL

- ✓ A trigger is a set of SQL statements that reside in system memory with unique names.
- ✓ It is a specialized category of stored procedure that is called automatically when a database server event occurs.
- ✓ Each trigger is always associated with a table.
- ✓ A **trigger is called a special procedure** because it cannot be called directly like a stored procedure.
- ✓ The key distinction between the trigger and procedure is that a trigger is called automatically when a data modification event occurs against a table.
- ✓ A stored procedure, on the other hand, must be invoked directly.

Characteristics that distinguish triggers from stored procedures

- We cannot manually execute/invoked triggers.
- Triggers have no chance of receiving parameters.
- A transaction cannot be committed or rolled back inside a trigger.

Syntax of Trigger

CREATE TRIGGER schema.trigger_name
ON table_name
AFTER {**INSERT**, **UPDATE**, **DELETE**}
[NOT **FOR** REPLICATION]
AS
{
SQL_Statements
}

schema: It is an optional parameter that defines which schema the new trigger belongs to.

trigger_name: It is a required parameter that defines the name for the new trigger.

table_name: It is a required parameter that defines the table name to which the trigger applies. Next to the table name, we need to write the AFTER clause where any events like INSERT, UPDATE, or DELETE could be listed.

NOT FOR REPLICATION: This option tells that SQL Server does not execute the trigger when data is modified as part of a replication process.

SQL_Statements: It contains one or more SQL statements that are used to perform actions in response to an event that occurs.

When we use triggers?

Triggers will be helpful when we need to execute some events automatically on certain desirable scenarios.

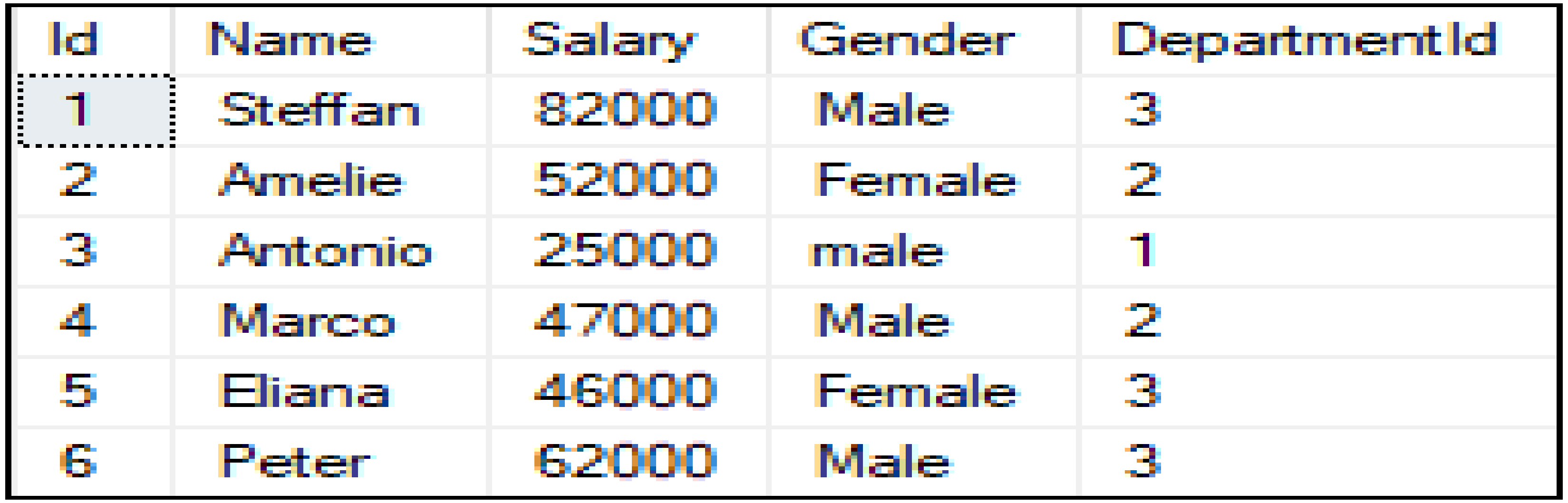
Example of Trigger in SQL Server

Let us understand how we can work with triggers in the SQL Server. We can do this by first creating a table named '**Employee**' using the below statements:

```
CREATE TABLE Employee
(
  Id INT PRIMARY KEY,
  Name VARCHAR(45),
  Salary INT,
  Gender VARCHAR(12),
  DepartmentId INT
)
```

```
INSERT INTO Employee VALUES (1,'Steffan', 82000, 'Male', 3),
(2,'Amelie', 52000, 'Female', 2),
(3,'Antonio', 25000, 'male', 1),
(4,'Marco', 47000, 'Male', 2),
(5,'Eliana', 46000, 'Female', 3)
```

```
1.SELECT * FROM Employee;
```



Id	Name	Salary	Gender	DepartmentId
1	Steffan	82000	Male	3
2	Amelie	52000	Female	2
3	Antonio	25000	male	1
4	Marco	47000	Male	2
5	Eliana	46000	Female	3
6	Peter	62000	Male	3

Any Query????

Thank you……

Any Query????

Thank you……