Trigger in SQL

In this article, you will learn about the trigger and its implementation with examples.

A **Trigger** in Structured Query Language is a set of procedural statements which are executed automatically when there is any response to certain events on the particular table in the database. Triggers are used to protect the data integrity in the database.

In SQL, this concept is the same as the trigger in real life. For example, when we pull the gun trigger, the bullet is fired.

To understand the concept of trigger in SQL, let's take the below hypothetical situation:

Suppose Rishabh is the human resource manager in a multinational company. When the record of a new employee is entered into the database, he has to send the 'Congrats' message to each new employee. If there are four or five employees, Rishabh can do it manually, but if the number of new Employees is more than the thousand, then in such condition, he has to use the trigger in the database.

Thus, now Rishabh has to create the trigger in the table, which will automatically send a 'Congrats' message to the new employees once their record is inserted into the database.

The trigger is always executed with the specific table in the database. If we remove the table, all the triggers associated with that table are also deleted automatically.

In Structured Query Language, triggers are called only either before or after the below events:

1. **INSERT Event:** This event is called when the new row is entered in the table.
2. **UPDATE Event:** This event is called when the existing record is changed or modified in the table.
3. **DELETE Event:** This event is called when the existing record is removed from the table.

Types of Triggers in SQL

Following are the six types of triggers in SQL:

1. **AFTER INSERT Trigger**
   This trigger is invoked after the insertion of data in the table.
2. **AFTER UPDATE Trigger**
   This trigger is invoked in SQL after the modification of the data in the table.
3. **AFTER DELETE Trigger**
   This trigger is invoked after deleting the data from the table.
4. **BEFORE INSERT Trigger**
   This trigger is invoked before the inserting the record in the table.
5. **BEFORE UPDATE Trigger**
   This trigger is invoked before the updating the record in the table.
6. **BEFORE DELETE Trigger**
   This trigger is invoked before deleting the record from the table.

Syntax of Trigger in SQL

⇧ SCROLL TO TOP

```
rigger_Name
```

```
[ BEFORE | AFTER ]  [ Insert | Update | Delete]
ON [Table_Name]
[ FOR EACH ROW | FOR EACH COLUMN ]
AS
Set of SQL Statement
```

In the trigger syntax, firstly, we have to define the name of the trigger after the CREATE TRIGGER keyword. After that, we have to define the BEFORE or AFTER keyword with anyone event.

Then, we define the name of that table on which trigger is to occur.

After the table name, we have to define the row-level or statement-level trigger.

And, at last, we have to write the SQL statements which perform actions on the occurring of event.

Example of Trigger in SQL

To understand the concept of trigger in SQL, first, we have to create the table on which trigger is to be executed.

The following query creates the **Student_Trigger** table in the SQL database:

```
CREATE TABLE Student_Trigger
(
Student_RollNo INT NOT NULL PRIMARY KEY,
Student_FirstName Varchar (100),
Student_EnglishMarks INT,
Student_PhysicsMarks INT,
Student_ChemistryMarks INT,
Student_MathsMarks INT,
Student_TotalMarks INT,
Student_Percentage  );
```

The following query shows the structure of the**Student_Trigger** table:

```
DESC Student_Trigger;
```

Output:

⇧ SCROLL TO TOP

Field	Type	NULL	Key	Default	Extra
Student_RollNo	INT	NO	PRI	NULL	
Student_FirstName	Varchar(100)	YES		NULL	
Student_EnglishMarks	INT	YES		NULL	
Student_PhysicsMarks	INT	YES		NULL	
Student_ChemistryMarks	INT	YES		NULL	
Student_MathsMarks	INT	YES		NULL	
Student_TotalMarks	INT	YES		NULL	
Student_Percentage	INT	YES		NULL	

The following query fires a trigger before the insertion of the student record in the table:

```
CREATE TRIGGER Student_Table_Marks
BEFORE INSERT
ON
Student_Trigger
FOR EACH ROW
SET new.Student_TotalMarks = new.Student_EnglishMarks + new.Student_PhysicsMarks + new.Student_ChemistryMarks + new.Student_Maths
new.Student_Percentage = ( new.Student_TotalMarks / 400) * 100;
```

The following query inserts the record into Student_Trigger table:

```
INSERT INTO Student_Trigger (Student_RollNo, Student_FirstName, Student_EnglishMarks, Student_PhysicsMarks, Student_ChemistryMarks, St
```

To check the output of the above INSERT statement, you have to type the following SELECT statement:

```
SELECT * FROM Student_Trigger;
```

Output:

Student_RollNo	Student_FirstName	Student_EnglishMarks	Student_PhysicsMarks	Student_chemistryMarks	Student_MathsMa
201	Sorya	88	75	69	92

Advantages of Triggers in SQL

Following are the three main advantages of triggers in Structured Query Language:

1. SQL provides an alternate way for maintaining the data and referential integrity in the tables.
2. Triggers helps in executing the scheduled tasks because they are called automatically.
3. They catch the errors in the database layer of various businesses.
4. They allow the database users to validate values before inserting and updating.

Disadvantages of Triggers in SQL

Following are the main disadvantages of triggers in Structured Query Language:

⇧ SCROLL TO TOP

ompiled.