Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

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:

```
create trigger [trigger_name]

[before | after]

{insert | update | delete}
```

on [table_name]

[for each row]

[trigger_body]

Explanation of syntax:

1. create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.
2. [before | after]: This specifies when the trigger will be executed.
3. {insert | update | delete}: This specifies the DML operation.
4. on [table_name]: This specifies the name of the table associated with the trigger.
5. [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
6. [trigger_body]: This provides the operation to be performed as trigger is fired

BEFORE and AFTER of Trigger:

BEFORE triggers run the trigger action before the triggering statement is run. AFTER triggers run the trigger action after the triggering statement is run.

Example:

Given Student Report Database, in which student marks assessment is recorded. In such schema, create a trigger so that the total and percentage of specified marks is automatically inserted whenever a record is insert.

Here, as trigger will invoke before record is inserted so, BEFORE Tag can be used.

Suppose the database Schema –

```
mysql> desc Student;

+-------+-------------+------+-----+---------+----------------+

| Field | Type        | Null | Key | Default | Extra          |

+-------+-------------+------+-----+---------+----------------+

| tid   | int(4)      | NO   | PRI | NULL    | auto_increment |

| name  | varchar(30) | YES  |     | NULL    |                |

| subj1 | int(2)      | YES  |     | NULL    |                |
| subj2 | int(2)      | YES  |     | NULL    |                |
| subj3 | int(2)      | YES  |     | NULL    |                |
| total | int(3)      | YES  |     | NULL    |                |
| per   | int(3)      | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

SQL Trigger to problem statement.

```
create trigger stud_marks
before INSERT
on
Student
for each row
set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total * 60 / 100;
```

Above SQL statement will create a trigger in the student database in which whenever subjects marks are entered, before inserting this data into the database, trigger will compute those two values and insert with the entered values. i.e.,

```
mysql> insert into Student values(0, "ABCDE", 20, 20, 20, 0, 0);
Query OK, 1 row affected (0.09 sec)


mysql> select * from Student;
+-----+-------+-------+-------+-------+-------+------+
| tid | name  | subj1 | subj2 | subj3 | total | per  |
+-----+-------+-------+-------+-------+-------+------+
| 100 | ABCDE |    20 |    20 |    20 |    60 |   36 |
+-----+-------+-------+-------+-------+-------+------+
1 row in set (0.00 sec)
```

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:

1. They are not compiled.
2. It is not possible to find and debug the errors in triggers.
3. If we use the complex code in the trigger, it makes the application run slower.
4. Trigger increases the high load on the database system.

Examples of triggers:

1) **After Delete**

```
create table employee7
(
eid int primary key,
salary int
);

insert into employee7 values(110,15000),
(220,30000),(330,23000);

select * from employee7;
```

```
CREATE TABLE SalaryBudget(
  total int
);//

INSERT INTO SalaryBudget(total)
SELECT SUM(salary)
FROM employee7;
//

select * from employee7;
select * from SalaryBudget;

CREATE TRIGGER after_salaries_delete1
AFTER DELETE
ON employee7 FOR EACH ROW
UPDATE SalaryBudget
SET total = total - old.salary;



DELETE FROM employee5
WHERE eid = 220;

select * from employee5;
select * from SalaryBudget;
```

2) Before Insert:

```
create table student
(
RNO int,
SName varchar(15),
age int,
city varchar(15)
);

insert into student values(1,"Raj",20,'Thane');
insert into student values(2,"Ram",21,'Pune');
insert into student values(3,"Rahul",18,'Mumbai');
insert into student values(8,'XYZ',-20,'Pune');

select * from student;

delimiter //
create trigger stud_age_verify
before insert
on student
for each row
if new.age<0 then set new.age=0;
end if;//

insert into student values(4,"Tony",19,'Pune');
(5,"Siya",-21,'Pune');

select * from student;

show triggers;

alter trigger stud_age_verify disable;
```