TRIGGERS

Triggers:

→A MySQL **trigger** is a stored program (with queries) which is executed automatically to respond to a specific event such as insertion, updation or deletion occurring in a table.

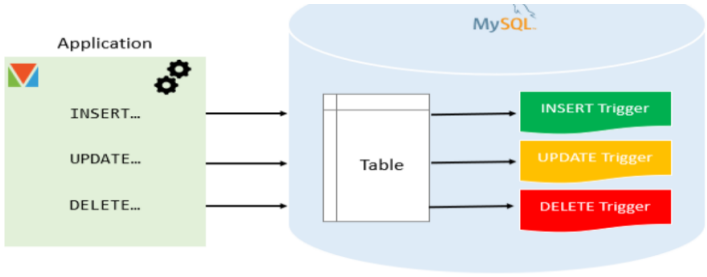
Event: the event that invokes the trigger. Events are INSERT, UPDATE,DELETE.

The SQL standard defines two types of triggers: row-level triggers and statement-level triggers.

- A row-level trigger is activated for each row that is inserted, updated, or deleted. For example, if a table has 100 rows inserted, updated, or deleted, the trigger is automatically invoked 100 times for the 100 rows affected.
- A statement-level trigger is executed once for each transaction regardless of how many rows are inserted, updated, or deleted.

Note: MySQL supports only row-level triggers. It doesn't support statement-level triggers.

Types of triggers in MySQL



Note: If you have multiple statements in the **trigger_body,** you have to use the BEGIN END block and change the default delimiter.

1.Before Update Trigger:

The keyword BEFORE indicates the trigger action time. In this case, the trigger activates before each row updated into the table.

Syntax:

Single Statement:

```
CREATE TRIGGER trigger_name
BEFORE UPDATE
ON table_name FOR EACH ROW
trigger_body
```

Multiple Statements:

```
DELIMITER $$

CREATE TRIGGER trigger_name
     BEFORE UPDATE
     ON table_name FOR EACH ROW
BEGIN
     -- statements
END$$

DELIMITER ;
```

2.After Update Trigger:

The keyword AFTER indicates the trigger action time. In this case, the trigger activates after each row updated into the table.

Syntax:

Single Statement:

```
CREATE TRIGGER trigger_name
AFTER UPDATE
ON table_name FOR EACH ROW
trigger_body
```

Multiple Statements:

```
DELIMITER $$

CREATE TRIGGER trigger_name
     AFTER UPDATE
     ON table_name FOR EACH ROW
BEGIN
     -- statements
END$$

DELIMITER ;
```

3.Before Insert Trigger:

Syntax:

Single Statement:

```
CREATE TRIGGER trigger_name
     BEFORE INSERT
     ON table_name FOR EACH ROW
trigger_body;
```

Multiple Statements:

```
DELIMITER $$

CREATE TRIGGER trigger_name
     BEFORE INSERT
     ON table_name FOR EACH ROW
BEGIN
     -- statements
END$$

DELIMITER ;
```

4.After Insert Trigger

Syntax:

Single Statement:

```
CREATE TRIGGER trigger_name
    AFTER INSERT
    ON table_name FOR EACH ROW
        trigger_body
```

Multiple Statements:

```
DELIMITER $$

CREATE TRIGGER trigger_name
     AFTER INSERT
     ON table_name FOR EACH ROW
BEGIN
     -- statements
END$$

DELIMITER ;
```

5.Before Delete Trigger

Syntax:

Single Statement:

```
CREATE TRIGGER trigger_name
    BEFORE DELETE
    ON table_name FOR EACH ROW
trigger_body
```

Multiple Statements:

```
DELIMITER $$

CREATE TRIGGER trigger_name
     BEFORE DELETE
     ON table_name FOR EACH ROW
BEGIN
     -- statements
END$$

DELIMITER ;
```

6.After Delete Trigger

Syntax:

Single Statement:

```
CREATE TRIGGER trigger_name
    AFTER DELETE
    ON table_name FOR EACH ROW
trigger_body;
```

Multiple Statements:

```
DELIMITER $$

CREATE TRIGGER trigger_name
     AFTER DELETE
     ON table_name FOR EACH ROW
BEGIN
     -- statements
END$$

DELIMITER ;
```

Creation:

Syntax:

create trigger **trigger_name**
BEFORE INSERT/AFTER INSERT/BEFORE UPDAT
/AFTER UPDATE/BEFORE DELETE/AFTER DELETE
ON **Table_name**
FOR EACH ROW SET **Trigger_body**

DROP:
To delete(drop) the trigger table.
Note: If you drop a table, any triggers for the table are also dropped.
Syntax:

DROP TRIGGER Trigger_name;

Show Trigger:
The SHOW TRIGGERS statement shows all triggers. The following illustrates the basic syntax of the SHOW TRIGGERS statement.
Syntaxes:

show tiggers returns all triggers in all databases	`SHOW TRIGGERS;`
To show all triggers in a specific database, you specify the database name after the FROM or IN keyword like this	`SHOW TRIGGERS FROM database_name;` `SHOW TRIGGERS IN database_name;`
To find triggers according to a pattern, you use the LIKE clause	`SHOW TRIGGERS FROM database_name LIKE 'pattern';`
To find triggers that match a condition, you use the WHERE clause	`SHOW TRIGGERS FROM database_name WHERE search_condition;`

Example:
Base Table creation:

```
create table student(
sid int primary key,sname varchar(20), marks int);
insert into student values(1001,'raam',23);
insert into student values(1002,'sita',26);
insert into student values(1003,'lakki',28);
insert into student values(1004,'feroz',27);
insert into student values(1005,'urva',25);
insert into student values(1007,'maruthy',24);
insert into student values(1006,'murthy',30);
create table student_audit(
sid int,sname varchar(20),marks int,
changedate datetime default null,
action varchar(50) default null
);
```

Trigger creation:

Before update

```
create trigger before_student_update
    before update on student
    for each row
    insert into student_audit
    set action='update',
    sid=old.sid,
   sname=old.sname,
   marks=old.marks,
    changedate=NOW();
```

After Insert

```
delimiter |
CREATE TRIGGER before_student_insert BEFORE INSERT ON student
         FOR EACH ROW
         BEGIN
           insert into student_audit SET marks = new.marks+2,
        sid=new.sid,
        sname=new.sname,
        changedate=NOW();
         END;
        |
        delimiter ;
```

Execution & checking:

```
update student set marks=24 where sid=1005;
insert into student values(1009,'grp2',35);
select * from student;
select * from student_audit;
```
