Name: Vishal Shashikant Salvi.

UID: 2019230069

Batch: C

Class: SE Comps

Experiment No 8

Aim: Practicing Triggers.

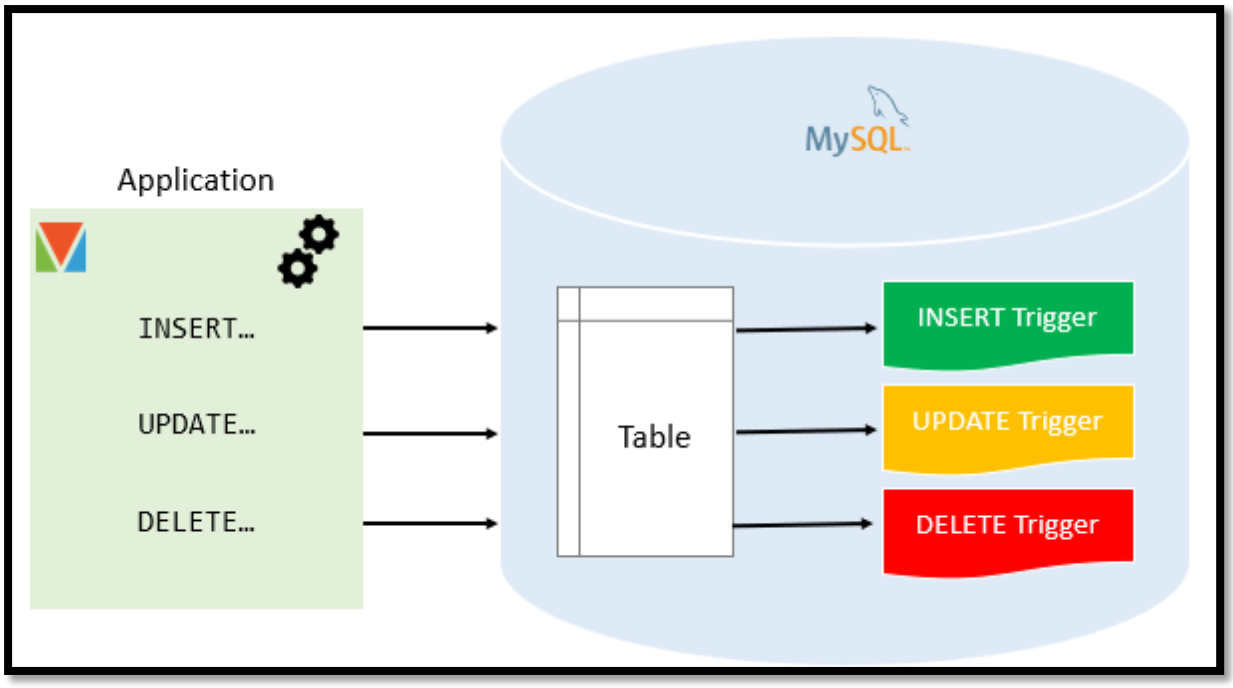
Theory:

In MySQL, a trigger is a stored program invoked automatically in response to an event such as insert, update, or delete that occurs in the associated table. For example, you can define a trigger that is invoked automatically before a new row is inserted into a table.

MySQL supports triggers that are invoked in response to
the INSERT, UPDATE or DELETE event.
The SQL standard defines two types of triggers: row-level triggers and statement-level triggers.

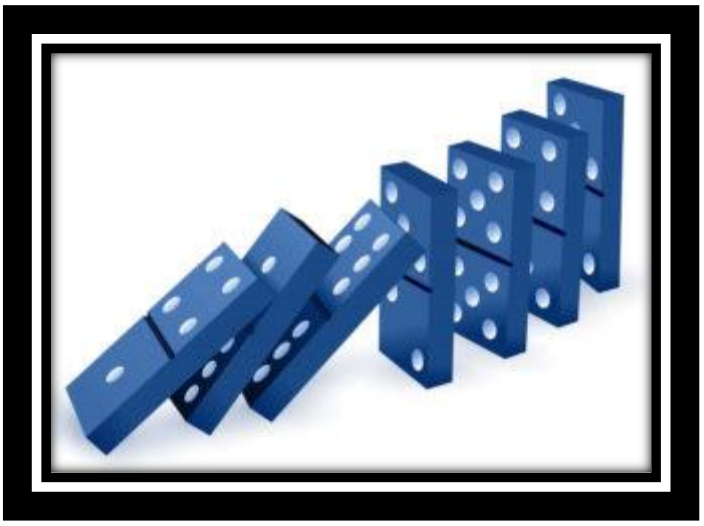
- A row-level trigger is activated for each row that is inserted, updated, or deleted. For example, if a table has 100 rows inserted, updated, or deleted, the trigger is automatically invoked 100 times for the 100 rows affected.
- A statement-level trigger is executed once for each transaction regardless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

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



What is a Trigger?

Triggers are the SQL codes that are automatically executed in response to certain events on a particular table. These are used to maintain the integrity of the data. A trigger in SQL works similar to a real-world trigger. For example, when the gun trigger is pulled a bullet is fired. We all know this, but how this is related to Triggers in SQL? To understand this let's consider a hypothetical situation.



John is the marketing officer in a company. When a new customer data is entered into the company's database, he has to send the welcome message to each new customer. If it is one or two customers John can do it manually, but what if the count is more than a thousand? Well in such scenario triggers come in handy.

Creating a Trigger

We now require two triggers:

- When a record is INSERTed into the blog table, we want to add a new entry into the audit table containing the blog ID and a type of 'NEW' (or 'DELETE' if it was deleted immediately).
- When a record is UPDATEd in the blog table, we want to add a new entry into the audit table containing the blog ID and a type of 'EDIT' or 'DELETE' if the deleted flag is set.

Note that the changetime field will automatically be set to the current time.

Each trigger requires:

1. A **unique name**. I prefer to use a name which describes the table and action, e.g. blog_before_insert or blog_after_update.
2. The **table** which triggers the event. A single trigger can only monitor a single table.
3. **When the trigger occurs**. This can either be BEFORE or AFTER an INSERT, UPDATE or DELETE. A BEFORE trigger must be used if you need to modify incoming data. An AFTER trigger must be used if you want to reference the new/changed record as a foreign key for a record in another table.
4. The **trigger body**; a set of SQL commands to run. Note that you can refer to columns in the subject table using OLD.col_name (the previous value) or NEW.col_name (the new value). The value for NEW.col_name can be changed in BEFORE INSERT and UPDATE triggers.

Syntax and Example

Now let me break down this syntax and explain each and every part in detail.

```
1  Create Trigger Trigger_Name
2  (Before | After)  [ Insert | Update | Delete]
3  on [Table_Name]
4  [ for each row | for each column ]
5  [ trigger_body ]
```

- **Create Trigger**
  These two keywords are used to specify that a trigger block is going to be declared.
- **Trigger_Name**
  It specifies the name of the trigger. Trigger name has to be unique and shouldn't repeat.
- **(Before | After)**
  This specifies when the trigger will be executed. It tells us the time at which the trigger is initiated, i.e, either before the ongoing event or after.

- Before Triggers are used to update or validate record values before they're saved to the database.
- After Triggers are used to access field values that are set by the system and to effect changes in other records. The records that activate the after trigger are read-only. We cannot use After trigger if we want to update a record because it will lead to read-only error.
- **[Insert | Update | Delete]**
  These are the DML operations and we can use either of them in a given trigger.
- **on [Table_Name]**
  We need to mention the table name on which the trigger is being applied. Don't forget to use **on** keyword and also make sure the selected table is present in the database.
- **[for each row | for each column]**

1.
    1. Row-level trigger gets executed before or after any column value of a row changes
    2. Column Level Trigger gets executed before or after the specified column changes

- **[trigger_body]**
  It consists of queries that need to be executed when the trigger is called.

So this was all about a simple trigger. But we can also create a nested trigger that can do multi-process. Also handling it and terminating it at the right time is very important. If we don't end the trigger properly it may lead to an infinite loop.

The basic trigger syntax is:

```
CREATE
   TRIGGER `event_name` BEFORE/AFTER INSERT/UPDATE/DELETE
   ON `database`.`table`
   FOR EACH ROW BEGIN
                    -- trigger body
                    -- this code is applied to every
                    -- inserted/updated/deleted row
   END;
```

We require two triggers — AFTER INSERT and AFTER UPDATE on the blog table. It's not necessary to define a DELETE trigger since a post is marked as deleted by setting it's deleted field to true.

Example for Trigger:

In the below trigger, we are trying to calculate the percentage of the student as soon as his details are updated to the database.

```
CREATE TRIGGER sample_trigger
before INSERT
ON student
FOR EACH ROW
SET new.total = new.marks/6;
```

Here the "**NEW"** keyword refers to the row that is getting affected.

Operations in Triggers

We can perform many operations using triggers. Some may be simple and some may be a little complex, but once if we go through the query its easy to understand.

- **DROP A Trigger**

```
DROP TRIGGER trigger name;
```

- **Display A Trigger**

The below code will display all the triggers that are present.

```
SHOW TRIGGERS;
```

The below code will display all the triggers that are present in a particular database.

```
SHOW TRIGGERS
IN database_name;
```

Example:

```
SHOW TRIGGERS IN edureka;
```

In the above example, all the triggers that are present in the database named Edureka will be displayed.

We also look at some major variants of the triggers that is Before insert and After insert. We have already seen a trigger in the example. But with the help of the table lets see how exactly this works.

Advantages of triggers

- Triggers provide another way to check the integrity of data.
- Triggers handle errors from the database layer.
- Triggers give an alternative way to run scheduled tasks. By using triggers, you don't have to wait for the scheduled events to run because the triggers are invoked automatically *before* or *after* a change is made to the data in a table.
- Triggers can be useful for auditing the data changes in tables.

  - Forcing **security** approvals on the table that are present in the database
  - Triggers provide another way to check the **integrity of data**
  - **Counteracting invalid** exchanges
  - Triggers **handle errors** from the database layer
  - Normally triggers can be useful for **inspecting the data** changes in tables
  - Triggers give an alternative way to run s**cheduled tasks**. Using triggers, we don't have to wait for the scheduled events to run because the triggers are invoked automatically before or after a change is made to the data in a table

Disadvantages of triggers

- Triggers can only provide extended validations, not all validations. For simple validations, you can use the NOT NULL, UNIQUE, CHECK and FOREIGN KEY constraints.
- Triggers can be difficult to troubleshoot because they execute automatically in the database, which may not invisible to the client applications.
- Triggers may increase the overhead of the MySQL Server.

  - Triggers can only provide extended **validations**, i.e, not all kind validations. For simple validations, you can use the NOT NULL, UNIQUE, CHECK and FOREIGN KEY constraints
  - Triggers may increase the **overhead** of the database
  - Triggers can be difficult to **troubleshoot** because they execute automatically in the database, which may not invisible to the client applications

Managing MySQL triggers

- Create triggers – describe steps of how to create a trigger in MySQL.
- Drop triggers – show you how to drop a trigger.
- Create a BEFORE INSERT trigger – show you how to create a BEFORE INSERT trigger to maintain a summary table from another table.
- Create an AFTER INSERT trigger – describe how to create an AFTER INSERT trigger to insert data into a table after inserting data into another table.
- Create a BEFORE UPDATE trigger – learn how to create a BEFORE UPDATE trigger that validates data before it is updated to the table.
- Create an AFTER UPDATE trigger – show you how to create an AFTER UPDATE trigger to log the changes of data in a table.
- Create a BEFORE DELETE trigger – show how to create a BEFORE DELETE trigger.
- Create an AFTER DELETE trigger – describe how to create an AFTER DELETE trigger.
- Create multiple triggers for a table that have the same trigger event and time – MySQL 8.0 allows you to define multiple triggers for a table that have the same trigger event and time.
- Show triggers – list triggers in a database, table by specific patterns.

Example 1:

Triggers on student Data

Show Database:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| test               |
| vishal             |
| world              |
+--------------------+
8 rows in set (0.01 sec)
```

Use Database:

```
mysql> use vishal;
Database changed
```

Create table student:

```
mysql> CREATE TABLE STUDENTS (
    -> ID INT NOT NULL,
    -> NAME VARCHAR (20) NOT NULL,
    -> AGE INT NOT NULL,
    -> ADDRESS CHAR (25),
    -> MARKS INT NOT NULL,
    -> PRIMARY KEY (ID)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> DESCRIBE STUDENTS;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| ID      | int(11)     | NO   | PRI | NULL    |       |
| NAME    | varchar(20) | NO   |     | NULL    |       |
| AGE     | int(11)     | NO   |     | NULL    |       |
| ADDRESS | char(25)    | YES  |     | NULL    |       |
| MARKS   | int(11)     | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Insert into table:

```
mysql> INSERT INTO STUDENTS VALUES(1,"Vishal Salvi",20,"Jogeshwari",450);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO STUDENTS VALUES(2,"Sandesh Kadge",21,"Dadar",400);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO STUDENTS VALUES(3,"Manish Bhagat",19,"Kurla",350);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO STUDENTS VALUES(4,"Rohit Lad",20,"Andheri",300);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO STUDENTS VALUES(5,"Rupesh Gosavi",21,"Malad",425);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO STUDENTS VALUES(6,"Siddhesh",20,"Virar",425);
Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENT;
Empty set (0.01 sec)

mysql> select * from STUDENTS;
+----+---------------+-----+------------+-------+
| ID | NAME          | AGE | ADDRESS    | MARKS |
+----+---------------+-----+------------+-------+
|  1 | Vishal Salvi  |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge |  21 | Dadar      |   400 |
|  3 | Manish Bhagat |  19 | Kurla      |   350 |
|  4 | Rohit Lad     |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi |  21 | Malad      |   425 |
|  6 | Siddhesh      |  20 | Virar      |   425 |
+----+---------------+-----+------------+-------+
6 rows in set (0.00 sec)
```

Create Sample Trigger:

```
mysql> CREATE TRIGGER Sample_Trigger
    -> before Insert
    -> ON STUDENTS
    -> FOR EACH ROW
    -> SET new.MARKS = new.Marks+100;
Query OK, 0 rows affected (0.02 sec)
```

Insert new value:

```
mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(7,"Rohit Vengurlekar",20,"Jogeshwari",300);
Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENTS;
+----+-------------------+-----+------------+-------+
| ID | NAME              | AGE | ADDRESS    | MARKS |
+----+-------------------+-----+------------+-------+
|  1 | Vishal Salvi      |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge     |  21 | Dadar      |   400 |
|  3 | Manish Bhagat     |  19 | Kurla      |   350 |
|  4 | Rohit Lad         |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi     |  21 | Malad      |   425 |
|  6 | Siddhesh          |  20 | Virar      |   425 |
|  7 | Rohit Vengurlekar |  20 | Jogeshwari |   400 |
+----+-------------------+-----+------------+-------+
7 rows in set (0.00 sec)
```

```
mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(8,"Harshal Parab",21,"Vashi",200);
Query OK, 1 row affected (0.00 sec)

mysql> select * from STUDENTS;
+----+-------------------+-----+------------+-------+
| ID | NAME              | AGE | ADDRESS    | MARKS |
+----+-------------------+-----+------------+-------+
|  1 | Vishal Salvi      |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge     |  21 | Dadar      |   400 |
|  3 | Manish Bhagat     |  19 | Kurla      |   350 |
|  4 | Rohit Lad         |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi     |  21 | Malad      |   425 |
|  6 | Siddhesh          |  20 | Virar      |   425 |
|  7 | Rohit Vengurlekar |  20 | Jogeshwari |   400 |
|  8 | Harshal Parab     |  21 | Vashi      |   300 |
+----+-------------------+-----+------------+-------+
8 rows in set (0.00 sec)

mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(8,"Shivam Pawar",20,"Virar",200);
ERROR 1062 (23000): Duplicate entry '8' for key 'PRIMARY'
mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(9,"Shivam Pawar",20,"Virar",200);
Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENTS;
+----+-------------------+-----+------------+-------+
| ID | NAME              | AGE | ADDRESS    | MARKS |
+----+-------------------+-----+------------+-------+
|  1 | Vishal Salvi      |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge     |  21 | Dadar      |   400 |
|  3 | Manish Bhagat     |  19 | Kurla      |   350 |
|  4 | Rohit Lad         |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi     |  21 | Malad      |   425 |
|  6 | Siddhesh          |  20 | Virar      |   425 |
|  7 | Rohit Vengurlekar |  20 | Jogeshwari |   400 |
|  8 | Harshal Parab     |  21 | Vashi      |   300 |
|  9 | Shivam Pawar      |  20 | Virar      |   300 |
+----+-------------------+-----+------------+-------+
9 rows in set (0.00 sec)

mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(10,"Kunal Nalawade",20,"Panvel",300);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(10,"Kunal Nalawade",20,"Panvel",300);
Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENTS;
+----+------------------+-----+------------+-------+
| ID | NAME             | AGE | ADDRESS    | MARKS |
+----+------------------+-----+------------+-------+
|  1 | Vishal Salvi     |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge    |  21 | Dadar      |   400 |
|  3 | Manish Bhagat    |  19 | Kurla      |   350 |
|  4 | Rohit Lad        |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi    |  21 | Malad      |   425 |
|  6 | Siddhesh         |  20 | Virar      |   425 |
|  7 | Rohit Vengurlekar |  20 | Jogeshwari |   400 |
|  8 | Harshal Parab    |  21 | Vashi      |   300 |
|  9 | Shivam Pawar     |  20 | Virar      |   300 |
| 10 | Kunal Nalawade   |  20 | Panvel     |   400 |
+----+------------------+-----+------------+-------+
10 rows in set (0.00 sec)

mysql> create table Final_marks(total_marks int);
Query OK, 0 rows affected (0.03 sec)

mysql> create trigger calculate
    -> after insert
    -> on STUDENTS
    -> for each row
    -> insert into Final_marks values(new.total_marks);
ERROR 1054 (42S22): Unknown column 'total_marks' in 'NEW'
mysql> select * from Final_marks;
Empty set (0.00 sec)

mysql> describe Final_marks;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| total_marks | int(11) | YES  |     | NULL    |       |
+-------------+---------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

```
mysql> insert into Final_marks values(new.total_marks);
ERROR 1054 (42S22): Unknown column 'new.total_marks' in 'field list'
mysql> insert into Final_marks values(new.MARKS);
ERROR 1054 (42S22): Unknown column 'new.MARKS' in 'field list'
mysql> create trigger calculate
    -> after insert
    ->
    -> on STUDENTS
    -> for each row
    -> insert into Final_marks values(new.MARKS);
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(11,"Rahul Bhosale",19,"Mira-Road",300);
Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENTS;
+----+------------------+-----+------------+-------+
| ID | NAME             | AGE | ADDRESS    | MARKS |
+----+------------------+-----+------------+-------+
|  1 | Vishal Salvi     |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge    |  21 | Dadar      |   400 |
|  3 | Manish Bhagat    |  19 | Kurla      |   350 |
|  4 | Rohit Lad        |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi    |  21 | Malad      |   425 |
|  6 | Siddhesh         |  20 | Virar      |   425 |
|  7 | Rohit Vengurlekar |  20 | Jogeshwari |   400 |
|  8 | Harshal Parab    |  21 | Vashi      |   300 |
|  9 | Shivam Pawar     |  20 | Virar      |   300 |
| 10 | Kunal Nalawade   |  20 | Panvel     |   400 |
| 11 | Rahul Bhosale    |  19 | Mira-Road  |   400 |
+----+------------------+-----+------------+-------+
11 rows in set (0.00 sec)
```

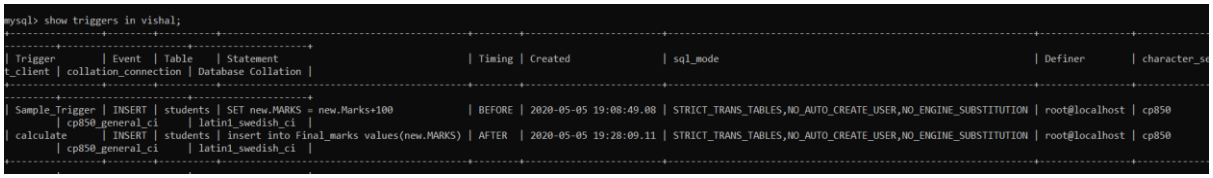
```
mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(12,"Omkar Bavkar",20,"Vasai",100);
Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENTS;
+----+-------------------+-----+------------+-------+
| ID | NAME              | AGE | ADDRESS    | MARKS |
+----+-------------------+-----+------------+-------+
|  1 | Vishal Salvi      |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge     |  21 | Dadar      |   400 |
|  3 | Manish Bhagat     |  19 | Kurla      |   350 |
|  4 | Rohit Lad         |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi     |  21 | Malad      |   425 |
|  6 | Siddhesh          |  20 | Virar      |   425 |
|  7 | Rohit Vengurlekar |  20 | Jogeshwari |   400 |
|  8 | Harshal Parab     |  21 | Vashi      |   300 |
|  9 | Shivam Pawar      |  20 | Virar      |   300 |
| 10 | Kunal Nalawade    |  20 | Panvel     |   400 |
| 11 | Rahul Bhosale     |  19 | Mira-Road  |   400 |
| 12 | Omkar Bavkar      |  20 | Vasai      |   200 |
+----+-------------------+-----+------------+-------+
12 rows in set (0.00 sec)

mysql> INSERT INTO STUDENTS(ID,NAME,AGE,ADDRESS,MARKS)VALUES(13,"Tanay Bagayatkar",19,"Vasai",400);
Query OK, 1 row affected (0.01 sec)

mysql> select * from STUDENTS;
+----+-------------------+-----+------------+-------+
| ID | NAME              | AGE | ADDRESS    | MARKS |
+----+-------------------+-----+------------+-------+
|  1 | Vishal Salvi      |  20 | Jogeshwari |   450 |
|  2 | Sandesh Kadge     |  21 | Dadar      |   400 |
|  3 | Manish Bhagat     |  19 | Kurla      |   350 |
|  4 | Rohit Lad         |  20 | Andheri    |   300 |
|  5 | Rupesh Gosavi     |  21 | Malad      |   425 |
|  6 | Siddhesh          |  20 | Virar      |   425 |
|  7 | Rohit Vengurlekar |  20 | Jogeshwari |   400 |
|  8 | Harshal Parab     |  21 | Vashi      |   300 |
|  9 | Shivam Pawar      |  20 | Virar      |   300 |
| 10 | Kunal Nalawade    |  20 | Panvel     |   400 |
| 11 | Rahul Bhosale     |  19 | Mira-Road  |   400 |
| 12 | Omkar Bavkar      |  20 | Vasai      |   200 |
| 13 | Tanay Bagayatkar  |  19 | Vasai      |   500 |
+----+-------------------+-----+------------+-------+
13 rows in set (0.00 sec)
```

Show Triggers:



```
mysql> show triggers in vishal;
+----------------+--------+----------+-------------------------------------------+--------+----------
------------+-----------------------------------------------------------+----------------+------
---------------+----------------------+----------------------+
| Trigger        | Event  | Table    | Statement                                 | Timing | Created
             | sql_mode                                                | Definer        | chara
cter_set_client | collation_connection | Database Collation |
+----------------+--------+----------+-------------------------------------------+--------+----------
------------+-----------------------------------------------------------+----------------+------
---------------+----------------------+----------------------+
| Sample_Trigger | INSERT | students | SET new.MARKS = new.Marks+100             | BEFORE | 2020-05-05
 19:08:49.08 | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION | root@localhost | cp850
                | cp850_general_ci     | latin1_swedish_ci  |
| calculate      | INSERT | students | insert into Final_marks values(new.MARKS) | AFTER  | 2020-05-05
 19:28:09.11 | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION | root@localhost | cp850
                | cp850_general_ci     | latin1_swedish_ci  |
+----------------+--------+----------+-------------------------------------------+--------+----------
------------+-----------------------------------------------------------+----------------+------
---------------+----------------------+----------------------+
2 rows in set (0.00 sec)
```

```
mysql> show triggers in vishal;
    -> $$
+---------------+--------+----------+----------------------------------------------------------------+--------+------------------------+---------------------------------------------
-------------------+----------------+----------------------+----------------------+--------------------+
| Trigger       | Event  | Table    | Statement                                                      | Timing | Created                | sql_mode
                   | Definer        | character_set_client | collation_connection | Database Collation |
+---------------+--------+----------+----------------------------------------------------------------+--------+------------------------+---------------------------------------------
-------------------+----------------+----------------------+----------------------+--------------------+
| calculate     | INSERT | students | insert into Final_marks values(new.MARKS)                      | AFTER  | 2020-05-05 19:28:09.11 | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_EN
GINE_SUBSTITUTION | root@localhost | cp850                | cp850_general_ci     | latin1_swedish_ci  |
| before_update | UPDATE | students | BEGIN
INSERT INTO UPDATE_LOG
SET action = 'update',
MARKS=OLD.MARKS,
CHANGEDAT = NOW();
END | BEFORE | 2020-05-05 20:25:28.13 | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION | root@localhost | cp850                | cp850_general_ci     | latin1_swedish_ci  |
| T2            | UPDATE | students | INSERT INTO LOG VALUES('UPDATE',CURRENT_TIMESTAMP,OLD.MARKS); | AFTER  | 2020-05-05 20:31:14.45 | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_EN
GINE_SUBSTITUTION | root@localhost | cp850                | cp850_general_ci     | latin1_swedish_ci  |
| T3            | UPDATE | students | INSERT INTO LOG VALUES('UPDATE',CURRENT_TIMESTAMP,OLD.MARKS); | AFTER  | 2020-05-05 20:36:37.83 | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_EN
GINE_SUBSTITUTION | root@localhost | cp850                | cp850_general_ci     | latin1_swedish_ci  |
+---------------+--------+----------+----------------------------------------------------------------+--------+------------------------+---------------------------------------------
-------------------+----------------+----------------------+----------------------+--------------------+
4 rows in set (0.00 sec)
```

Update:

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER before_update
    -> BEFORE UPDATE ON STUDENTS
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO UPDATE_LOG
    -> SET action = 'update',
    -> MARKS=OLD.MARKS,
    -> CHANGEDAT = NOW();
    -> END$$
Query OK, 0 rows affected (0.02 sec)
```

Drop Trigger:

```
mysql> DROP TRIGGER Sample_Trigger;
    -> $$
Query OK, 0 rows affected (0.01 sec)
```

Example 2:

create a small example database for a blogging application. Two tables are required:

- **`blog`: stores a unique post ID, the title, content, and a deleted flag.**
- **`audit`: stores a basic set of historical changes with a record ID, the blog post ID, the change type (NEW, EDIT or DELETE) and the date/time of that change.**

Create table blog:

```
mysql> CREATE TABLE `blog` (
    -> `id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
    -> `title` text,
    -> `content` text,
    -> `deleted` tinyint(1) unsigned NOT NULL DEFAULT '0',
    -> PRIMARY KEY (`id`),
    -> KEY `ix_deleted` (`deleted`)
    -> ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='Blog posts';
Query OK, 0 rows affected (0.03 sec)
```

Create table audit

```
mysql>
mysql> CREATE TABLE `audit` (
    -> `id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
    -> `blog_id` mediumint(8) unsigned NOT NULL,
    -> `changetype` enum('NEW','EDIT','DELETE') NOT NULL,
    -> `changetime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    -> PRIMARY KEY (`id`),
    -> KEY `ix_blog_id` (`blog_id`),
    -> KEY `ix_changetype` (`changetype`),
    -> KEY `ix_changetime` (`changetime`),
    -> CONSTRAINT `FK_audit_blog_id` FOREIGN KEY (`blog_id`) REFERENCES `blog` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
    -> ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.04 sec)
```

Create Trigger:

```
mysql> CREATE
    -> TRIGGER `blog_after_insert` AFTER INSERT
    -> ON `blog`
    -> FOR EACH ROW BEGIN
    ->
    -> IF NEW.deleted THEN
    -> SET @changetype = 'DELETE';
    -> ELSE
    -> SET @changetype = 'NEW';
    -> END IF;
    ->
    -> INSERT INTO audit (blog_id, changetype) VALUES (NEW.id, @changetype);
    ->
    ->     END$$
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> DELIMITER $$
mysql>
mysql> CREATE
    -> TRIGGER `blog_after_update` AFTER UPDATE
    -> ON `blog`
    -> FOR EACH ROW BEGIN
    ->
    -> IF NEW.deleted THEN
    -> SET @changetype = 'DELETE';
    -> ELSE
    -> SET @changetype = 'EDIT';
    -> END IF;
    ->
    -> INSERT INTO audit (blog_id, changetype) VALUES (NEW.id, @changetype);
    ->
    ->     END$$
Query OK, 0 rows affected (0.03 sec)
```

Insert values:

```
mysql> INSERT INTO blog (title, content) VALUES ('Article One', 'Initial text.');
Query OK, 1 row affected (0.01 sec)

mysql> select * from blog;
+----+-------------+---------------+---------+
| id | title       | content       | deleted |
+----+-------------+---------------+---------+
|  1 | Article One | Initial text. |       0 |
+----+-------------+---------------+---------+
1 row in set (0.00 sec)
```

```
mysql> select * from audit;
+----+---------+------------+---------------------+
| id | blog_id | changetype | changetime          |
+----+---------+------------+---------------------+
|  1 |       1 | NEW        | 2020-05-05 22:25:44 |
+----+---------+------------+---------------------+
1 row in set (0.00 sec)
```

Update Blog

```
mysql> UPDATE blog SET content = 'Edited text' WHERE id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from blog;
+----+-------------+-------------+---------+
| id | title       | content     | deleted |
+----+-------------+-------------+---------+
|  1 | Article One | Edited text |       0 |
+----+-------------+-------------+---------+
1 row in set (0.00 sec)

mysql> select * from audit;
+----+---------+------------+---------------------+
| id | blog_id | changetype | changetime          |
+----+---------+------------+---------------------+
|  1 |       1 | NEW        | 2020-05-05 22:25:44 |
|  2 |       1 | EDIT       | 2020-05-05 22:26:24 |
+----+---------+------------+---------------------+
2 rows in set (0.00 sec)
```

```
mysql> UPDATE blog SET deleted = 1 WHERE id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from blog;
+----+-------------+-------------+---------+
| id | title       | content     | deleted |
+----+-------------+-------------+---------+
|  1 | Article One | Edited text |       1 |
+----+-------------+-------------+---------+
1 row in set (0.00 sec)

mysql> select * from audit;
+----+---------+------------+---------------------+
| id | blog_id | changetype | changetime          |
+----+---------+------------+---------------------+
|  1 |       1 | NEW        | 2020-05-05 22:25:44 |
|  2 |       1 | EDIT       | 2020-05-05 22:26:24 |
|  3 |       1 | DELETE     | 2020-05-05 22:26:48 |
+----+---------+------------+---------------------+
3 rows in set (0.00 sec)
```

Conclusion:

Thus, 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.