Triggers

Introduction

A Trigger can be considered as an Event Driven Stored Procedure. We cannot call a trigger explicitly. Triggers are called automatically on the happening of an event such as After Insert, Before Update etc.

Contd…

There is two MySQL extension to triggers 'OLD' and 'NEW'. OLD and NEW are not case sensitive.

-Within the trigger body, the OLD and NEW keywords enable you to access columns in the rows affected by a trigger

- In an INSERT trigger, only NEW.col_name can be used.

- In a UPDATE trigger, you can use OLD.col_name to refer to the columns of a row before it is updated and NEW.col_name to refer to the columns of the row after it is updated.

- In a DELETE trigger, only OLD.col_name can be used; there is no new row.

MySQL trigger syntax

```
CREATE TRIGGER trigger_name trigger_time
   trigger_event
 ON table_name
 FOR EACH ROW
 BEGIN
 ...
 END;
```

BEFORE INSERT

```
mysql> CREATE TABLE people (age INT, name varchar(150));
```

Next we will define the trigger. It will be executed before every INSERT statement for the people table:

```
mysql> delimiter //
mysql> CREATE TRIGGER agecheck BEFORE INSERT ON people FOR EACH ROW IF NEW.age < 0 THEN SET NEW.age = 0; END IF;//
Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

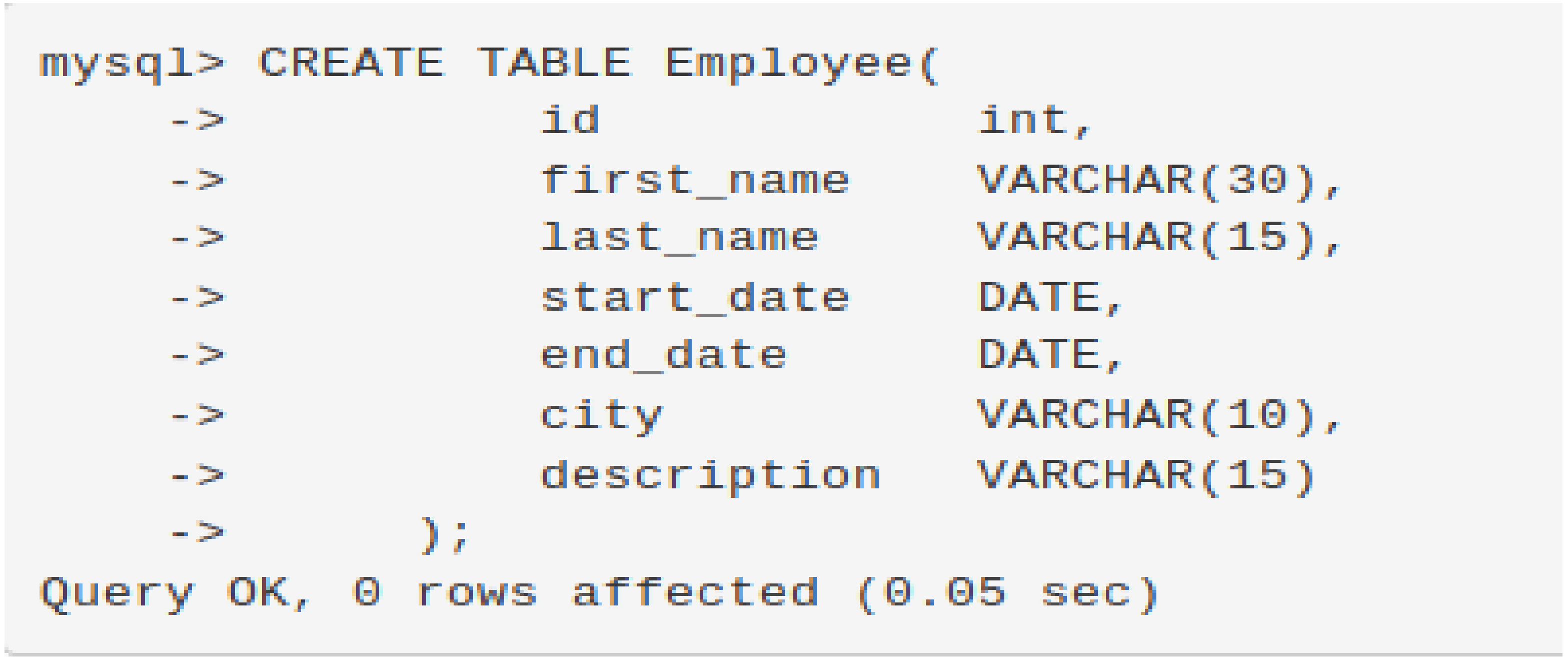
Contd...

We will insert two records to check the trigger functionality.

```
mysql> INSERT INTO people VALUES (-20, ‘Sid’), (30, ‘Josh’);
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql> SELECT * FROM people;
+——-+——-+
| age | name |
+——-+——-+
| 0    | John   |
| 30   | Ram    |
+——-+——-+
2 rows in set (0.00 sec)
```

After Insert

Mysql Trigger After Insert fired the trigger after you perform an insert a records or rows to the table.



```
mysql> CREATE TABLE Employee(
    ->            id             int,
    ->            first_name     VARCHAR(30),
    ->            last_name      VARCHAR(15),
    ->            start_date     DATE,
    ->            end_date       DATE,
    ->            city           VARCHAR(10),
    ->            description    VARCHAR(15)
    ->        );
Query OK, 0 rows affected (0.05 sec)
```

Contd….

```
+------+-----------+----------+-----------+-----------+---------+------------+
| id| first_name | last_name | start_date | end_date| city|description |
+------+-----------+----------+-----------+-----------+---------+------------+
| 1 | Girish| Tewari| 2008-12-25 | 2010-06-25|Nainital |Programmer  |
| 2 |Komal|Choudhry| 2007-11-22|2010-04-21 |Meerut|Programmer  |
| 3 |Mohan|Singh|2006-10-12|2007-05-12|Lucknow|Programmer  |
+------+-----------+----------+-----------+-----------+---------+------------+
3 rows in set (0.00 sec)
```

Contd…

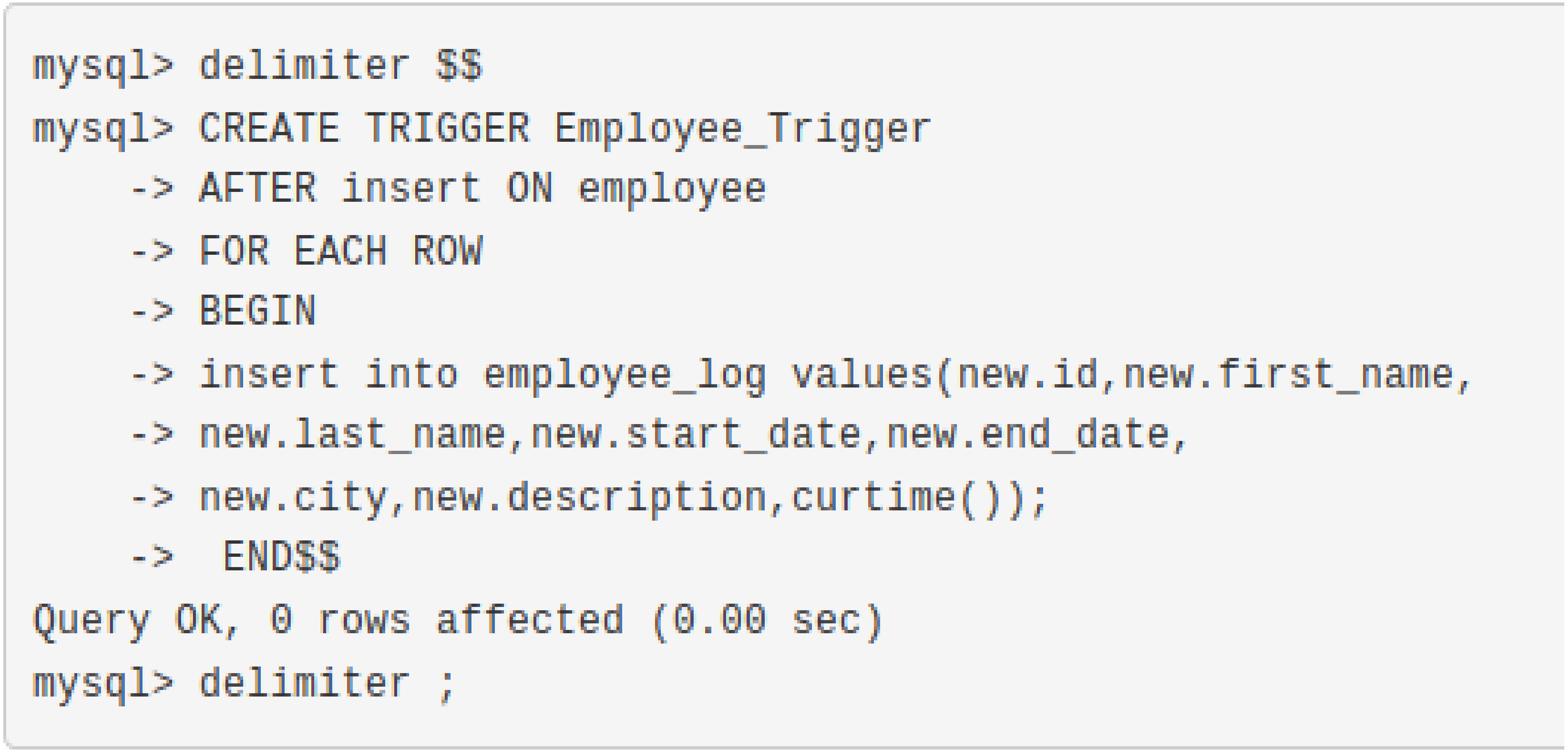
```
mysql> CREATE TABLE Employee_log(id int,first_name
   varchar(50),last_name varchar(50),start_date
   date,end_date date,city varchar(50),description
   varchar(50), Lasinserted Time
        );

mysql> select * from employee_log;
Empty set (0.02 sec)
```

Contd….

Create Trigger:-

The Create Trigger create a trigger 'Employee_Trigger' on table 'employee'. The 'After insert trigger' fired the trigger after you performed a insert operation on table.

Contd….



```
mysql> delimiter $$
mysql> CREATE TRIGGER Employee_Trigger
    -> AFTER insert ON employee
    -> FOR EACH ROW
    -> BEGIN
    -> insert into employee_log values(new.id,new.first_name,
    -> new.last_name,new.start_date,new.end_date,
    -> new.city,new.description,curtime());
    ->  END$$
Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
```

Contd....

Query to insert into employee table:-

The insert into add the records or rows into the table 'employee'.

```
mysql> insert into employee values
(4,'Amit','Kumar','20081225','2010120','Lucknow','Programmer');
```

Table that has been modified after update query executes is Employee_log:-

Query to view Employee_log table:-

```
mysql> select * from employee_log;
```

Contd….

```
+---+-------+----+-------+------+-------+--------+------+
| id| first_name|last_name| start_date|end_date |city|description |
    Lasinserted |
+------+-----------+----------+-----------+-----------+---------+------------
|4 | Amit| Kumar|2008-12-25| 2010-12-03|Lucknow|Programmer |
    14:55:31    |
+------+-----------+----------+-----------+-----------+---------+------------
```

AFTER UPDATE

```
mysql> select * from employee;
+------+------------+-----------+------------+------------+---------+------------+
| id   | first_name | last_name | start_date | end_date   | city    |
  description |
+------+------------+-----------+------------+------------+---------+------------+
|    1 | Girish     | Tewari    | 2008-12-25 | 2010-06-25 | Nainital |
  Programmer  |
|    2 | Komal      | Choudhry  | 2007-11-22 | 2010-04-21 | Meerut   |
  Programmer  |
|    3 | Mahendra   | Singh     | 2006-10-12 | 2007-05-12 | Lucknow  |
  Programmer  |
+------+------------+-----------+------------+------------+---------+------------+
3 rows in set (0.00 sec)
```

Contd...

```
mysql> CREATE TABLE Employee_log(
    ->          user_id      VARCHAR(15),
    ->          description   VARCHAR(100)
    ->      );

mysql> select * from employee_log;
```

Output:-

```
Empty set (0.00 sec)
```

Contd..

Create Trigger:-

The Create Trigger create a Employee_Trigger on table 'employee'. The Trigger 'Employee_Trigger' fired automatically after update operation is performed on table 'employee'. Further,the update records is inserted on table 'Employee_log'.

Contd..

```
mysql> delimiter $$
mysql> CREATE TRIGGER Employee_Trigger
    -> AFTER UPDATE ON employee
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT into Employee_log
    -> (user_id, description)VALUES (user(),
    -> CONCAT('Id with ',NEW.id,' is modified ',
    -> ' from ',OLD.start_date, ' to ', NEW.start_date));
    ->  END$$
Query OK, 0 rows affected (0.00 sec)
```

Contd..

```
mysql> delimiter ;
```

uery to update employee table:-

```
mysql> update employee set start_date='20061231';
```

Table that has been modified after update query executes is Employee_log:-

Query to view Employee_log table:-

```
mysql> select * from employee_log;
```

Output:-

```
+---------------+------------------------------------------+
| user_id       | description                              |
+---------------+------------------------------------------+
| root@localhost | Id with 1 is modified  from 2008-12-24 to 2006-
   12-31 |
| root@localhost | Id with 2 is modified  from 2008-12-24 to 2006-
   12-31 |
| root@localhost | Id with 3 is modified  from 2008-12-24 to 2006-
   12-31 |
+---------------+------------------------------------------+
3 rows in set (0.00 sec)
```

Mysql Trigger After Delete

Create table:-

```
mysql> CREATE TABLE Employee(
    ->          id             int,
    ->          first_name    VARCHAR(30),
    ->          last_name     VARCHAR(15),
    ->          start_date    DATE,
    ->          end_date      DATE,
    ->          city          VARCHAR(10),
    ->          description   VARCHAR(15)
    ->      );
Query OK, 0 rows affected (0.05 sec)
```

After Delete

Create Trigger:-

Drop Trigger is used to delete trigger Employee_Trigger from database, in case if there is any trigger existing on this name. The 'Employee_Triger' fired trigger after you perform a delete operation on table 'employee'. The deleted records from table 'employee' copy into a table 'employee_log'.

Contd...

```
drop trigger if exists Employee_Trigger;
delimiter $$
CREATE TRIGGER Employee_Trigger
AFTER delete ON employee
FOR EACH ROW
BEGIN
insert into employee_log values(old.id,old.first_name,
old.last_name,old.start_date,old.end_date,
old.city,old.description,curtime());
 END$$
delimiter ;
```

Query to delete record from employee table:-

We run a delete query in table employee where id is '4'.

```
mysql> delete from employee where id =4;
```

Contd...

Table that has been modified after delete query executes is Employee_log:-

Query to view Employee_log table:-

```
mysql> select * from employee_log;
```

Output:-

```
+------+------------+-----------+------------+------------+---------+-------------+-------------+
| id   | first_name | last_name | start_date | end_date   | city    | description |
  Lasinserted |
+------+------------+-----------+------------+------------+---------+-------------+-------------+
|    4 | Amit       | Kumar     | 2008-12-25 | 2010-12-03 | Lucknow | Programmer  |
  15:42:38    |
+------+------------+-----------+------------+------------+---------+-------------+-------------+
1 row in set (0.00 sec)
```

BEFORE UPDATE

```
mysql> SELECT * FROM STUDENT_MARKS;
+------------+-----------------+------+------+------+------+------+-------
   +-----------+-------+
|STUDENT_ID | NAME| SUB1 | SUB2 | SUB3 | SUB4 | SUB5 | TOTAL |
   PER_MARKS | GRADE |
+------------+-----------------+------+------+------+------+------+-------
   +-----------+-------+
|1 | Steven King      |    0 |    0 |    0 |    0 |    0 |     0 |      0.00 |       |
|2 | Neena  Kochhar   |    0 |    0 |    0 |    0 |    0 |     0 |      0.00 |
   |
|3 | Lex  De Haan     |    0 |    0 |    0 |    0 |    0 |     0 |      0.00 |       |
|4 | Alexander Hunold | 0 |    0 |    0 |    0 |    0 |     0 |      0.00 |       |
+------------+-----------------+------+------+------+------+------+--
```

Contd…

```
mysql> UPDATE STUDENT_MARKS SET SUB1 = 54,
   SUB2 = 69, SUB3 = 89, SUB4 = 87, SUB5 = 59
   WHERE STUDENT_ID = 1;
Query OK, 1 row affected (0.05 sec)
```

Contd...

```
Mysql> DELIMITER //
Mysql> CREATE TRIGGER mark_u
BEFORE UPDATE
ON student_marks FOR EACH ROW
BEGIN
SET NEW.TOTAL = NEW.sub1 + NEW.sub2 ;
SET NEW.PER_MARKS = NEW.TOTAL/2;
IF NEW.PER_MARKS >=90 THEN
SET NEW.GRADE = 'EXCELLENT';
ELSEIF NEW.PER_MARKS>=75 AND NEW.PER_MARKS<90 THEN
SET NEW.GRADE = 'VERY GOOD';
ELSEIF NEW.PER_MARKS>=60 AND NEW.PER_MARKS<75 THEN
SET NEW.GRADE = 'GOOD';
ELSEIF NEW.PER_MARKS>=40 AND NEW.PER_MARKS<60 THEN
SET NEW.GRADE = 'AVERAGE';
ELSE SET NEW.GRADE = 'NOT PROMOTED';
END IF;
END //
Mysql> DELIMITER ;
```

Contd...

Now check the STUDENT_MARKS table with updated data. The trigger show you the updated records in 'stu_log'.

```
mysql> SELECT * FROM STUDENT_MARKS;
+------------+------------------+------+------+------+------+------+-------+-----------+-------+
| STUDENT_ID | NAME             | SUB1 | SUB2 | SUB3 | SUB4 | SUB5 | TOTAL | PER_MARKS | GRADE |
+------------+------------------+------+------+------+------+------+-------+-----------+-------+
|          1 | Steven King      |   54 |   69 |   89 |   87 |   59 |   358 |     71.60 | GOOD  |
|          2 | Neena  Kochhar   |    0 |    0 |    0 |    0 |    0 |     0 |      0.00 |       |
|          3 | Lex  De Haan     |    0 |    0 |    0 |    0 |    0 |     0 |      0.00 |       |
|          4 | Alexander Hunold |    0 |    0 |    0 |    0 |    0 |     0 |      0.00 |       |
+------------+------------------+------+------+------+------+------+-------+-----------+-------+
4 rows in set (0.00 sec)
```

Advantage of a stored procedure over a database trigger

1. Stored procedures can accept parameters and can return values. Triggers can neither accept parameters nor return values. A Trigger is dependent on a table and the application has no control to not fire a trigger when not needed. On the other hand, a stored procedure can be called as needed.

2.Firing of a stored procedure can be controlled whereas on the other hand trigger will get fired whenever any modification takes place on the table.

3. Stored procedure is a set of pre-compiled SQL statements, executed when it is called in the program.

Triggers are similar to stored procedure except it is executed automatically when any operations are occurred on the table.

4. Procedure runs only when one call them manually whereas a trigger runs when there is any activity (insert,update,delete) on table on which the trigger is written.

Difference between Stored Procedure and Trigger

1) Triggers do not accept parameters where as procedures can accept parameters.
2) A trigger is executed implicitly by the Database engine. Where as, a procedure is executed explicitly by invoking a call to the procedure.
3) A trigger is event-driven. Where as, a procedure is not event-driven.
4) A trigger is attached to a specific table. Where as, procedures are not attached to any table.