Name : Vedant V Mehta
Class : SE Computers
UID : 2018130028
Class : SE COmputers

Aim – To implement tiggers in Mysql

Theory -

A MySql trigger is a set of SQL statements stored in the database.

A SQL trigger is a "special type" of stored procedure. A stored procedure are executed when they are called by user whereas, Triggers is not called by user.
A trigger is attached to a table so when a insert, update or delete event occurs with that table the MySql trigger is fired.
Trigger can be executed after as well as before insert, update or delete.

Any logic so as to keep check on the data being updated or inserted or tried to be deleted from the table is first checked by firing a trigger.
Therefore Triggers are never called by users but only through operations. There are different types of triggers for different operations.
Triggers in Mysql is a concept which has a variety of appilcations like security, constant check and many more.

Various Types -

1. Before insert
2. After insert
3. Before Update
4. After Update
5. Before Delete
6. After Delete

Creating Trigger Body

Create Trigger trigger_name trigger_time(Before or After) trigger_event(Update or Delete or Insert)
On table_name

For Each Row(So that even if an operation is perfomed on a single row the trigger gets fired)
Begin
....trigger_body...
End;

Example -

```
DELIMITER //

CREATE TRIGGER tr_AfterInsertEmp
AFTER INSERT on Emp(base table
name) FOR EACH ROW
BEGIN

INSERT into emp_audit(auditing table) VALUES(null,concat('A row is inserted into employee table at ', date_formate(now(), '%d-%m-%y %h:%i:%s %p'))); END //
DELIMITER ;
```

New and Old keyword in Mysql -

1.) Mysql provides us two magical or virtual tables called NEW and OLD.

2.) When we insert a row in a table then the row is also inserted in NEW table 3.) When we delete a row from a table the that row is inserted in OLD table.

Example using the new table after inserting an entry into the base table-

```
DELIMITER //
CREATE TRIGGER tr_AfterInsertEmp1
AFTER INSERT ON Emp
FOR EACH ROW
BEGIN
DECLARE emp_id int; set
emp_id = new.id;
INSERT INTO emp_audit VALUES(null, concat("A row is inserted in employee table with id: ", emp_id, " at", date_formate(now(), "%d-%m-%y :
%h:%i:%s %p")));
END //
DELIMITER ;
```

Example using old table after deleting an entry from the base table-

```
DELIMITER //
CREATE TRIGGER tr_AfterDeleteEmp
AFTER DELETE ON emp
FOR EACH ROW
BEGIN
DECLARE emp_id int; set
emp_id = old.id;
INSERT INTO emp_audit VALUES(null, concat("A row is deleted
from employee table with id: ", emp_id, " at", date_formate(now(),
"%d-%m-%y :
%h:%i:%s %p")));
END // DELIMITER
;
```

Example using old and new for table after updating the

base table - DELIMITER //

```
CREATE TRIGGER tr_AfterUpdateEmp
AFTER UPDATE ON Emp FOR
EACH ROW
BEGIN
DECLAREinserted_emp_name varchar(50); DECLARE deleted_emp_name
varchar(50); SET inserted_emp_name = new.name;
SET deleted_emp_name = old.name;
INSERT INTO emp_audit VALUES(null, concat("A name",
deleted_emp_name, "replaced with name", inserted_emp_name,
"at", date_formate(now(), "%d-%m-%y :%h:%i:%s %p")));
END //
DELIMITER ;
```

Note – For "before" trigger it works as that when the new data is inserted into the base table then before insertion into the base table it first gets inserted into the NEW table then its checked for its validity if not valid then it's rejected from the NEW table itself and hence protecting the base table data. Similarly its for deletion when the data from the base table is tried to be deleted, then it's checked at that point only that if deletion is allowed or not. If yes then the data is deleted and moved to the OLD table.

Example for using the before trigger-

```
DELIMITER //
CREATE TRIGGER tr_BeforeInsertEmp
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
DECLARE emp_age int;
SET emp_age = new.age;
IF emp_age<18
THEN SIGNAL SQLSTATE '45000' SET MESSAGE.TEXT = "You are not eligible";
END IF;
END //
DELIMITER ;
```

DROPPING OF TRIGGER -

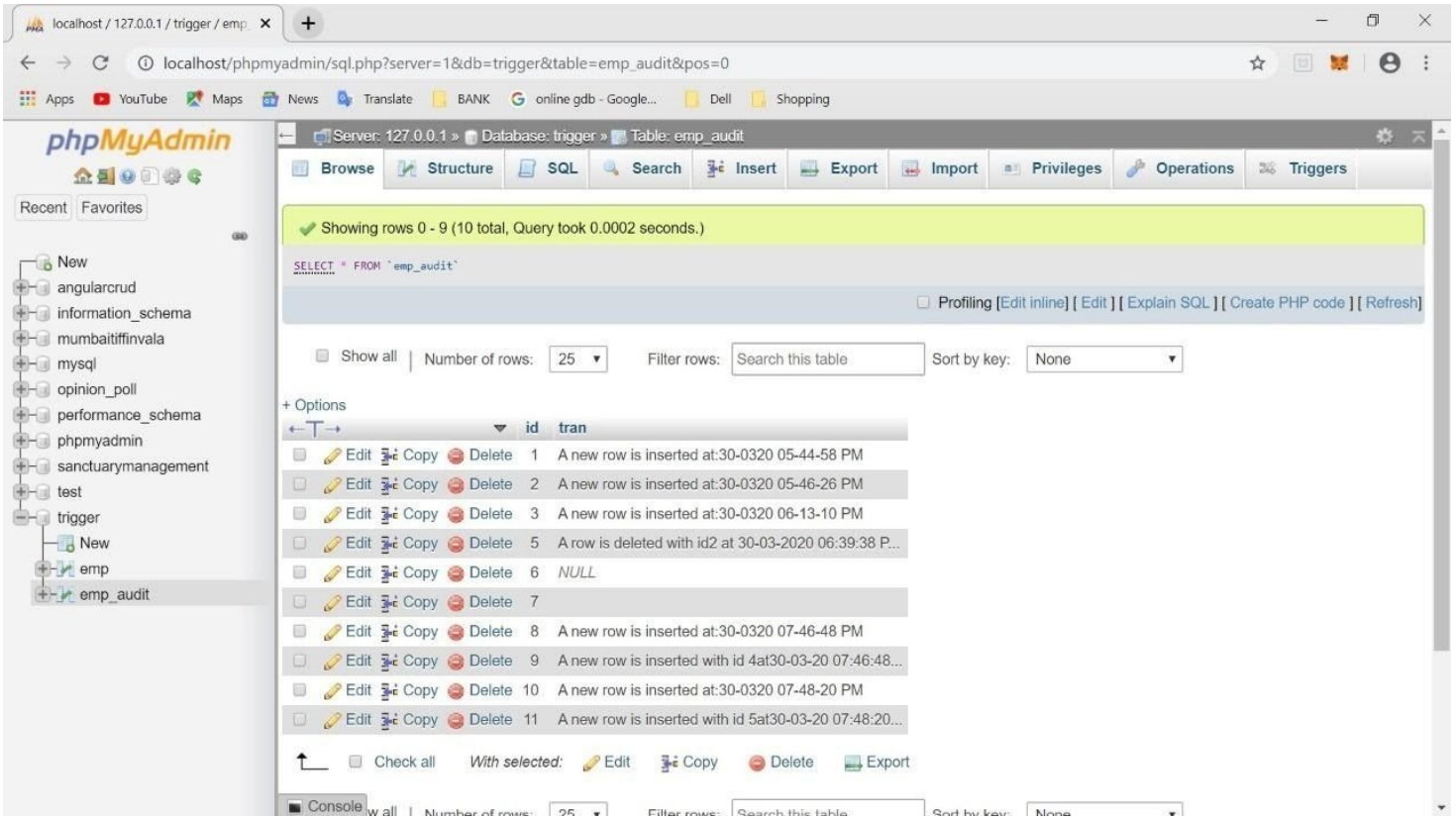
```
DROP TRIGGER trigger_name;
```

Advantages of triggers -

1. SQL trigger provide an alternative to check the integrity of data.
2. SQL trigger can catch errors in business logic in the database layer.
3. SQL trigger are scheduled therfore one dosen't have to wait for checking the new data or checking before deleting old data
4. SQL trigger help to provide security to data in tables by maintaining a copy of every change seperately in the so called audit table for future reference.

Disadvantages of triggers -

1. Triggers can provide only extended validations. For simple validations one needs to use check constraints, not null constraints etc.
2. Triggers are difficult to troubleshoot since they are fired automatically in the Mysql server.
3. Triggers may increase overheads of mysql server.



Conclusion – After completing the above experiment, I have understood how to use triggers and their advantage in terms of protections and automation.