LAB 10 SQL Triggers

OBJECTIVE(S)

- Learn about SQL Triggers

SQL TRIGGER

A trigger is a set of SQL statements associated with a table that are executed (or fired) when an event associated with the table occurs such as INSERT, UPDATE, DELETE. It can be invoked before or after the event. If, however, a statement that does not use the INSERT, UPDATE, or DELETE command is used to manipulate data (e.g. the TRUNCATE command), the triggers associated with the table are not invoked.

A trigger can be considered as a special type of stored procedure. The main difference between triggers and stored procedures is that a trigger is automatically called when a data modification event occurs in a table whereas a stored procedure must be called explicitly. Furthermore, multiple triggers for the same event and action time can be defined.

Advantages of SQL Triggers

- Alternative way to check the integrity of data.
- Can be used to catch errors in business logic in the database layer.
- Provide an alternative way to run scheduled tasks since the triggers are invoked automatically *before* or *after* a change is made to the data in the tables.
- Used to audit the changes of data in the tables.

Disadvantages of SQL Triggers

- May increase the overhead of the database server.
- Makes it difficult to figure out what happens in the database layer since triggers are invoked and executed invisible from the client applications.

CREATING A TRIGGER

The CREATE TRIGGER command is used to create a trigger in MySQL.

- ```
  DELIMITER $$
  CREATE TRIGGER trigger_name BEFORE/AFTER INSERT/UPDATE/DELETE
      ON tb_name
      FOR EACH ROW
      BEGIN
          statement(s);
      END$$
  DELIMITER ;
  ```
  ```

For instance, consider the following trigger that is invoked when a change is made to the *students* table. For that purpose, let us first define a new table that logs all the changes.

```
CREATE TABLE students_audit (
        auditID int PRIMARY KEY AUTO_INCREMENT,
        studentID int,
        lname varchar(50),
        action varchar(20));
```

We can now define a trigger that logs these changes.

```
DELIMITER $$
CREATE TRIGGER before_students_update
        BEFORE UPDATE ON students
        FOR EACH ROW
        BEGIN
                INSERT INTO students_audit
                        SET action = 'update',
                        studentID = OLD.studentID,
                        lname = OLD.lname;
        END$$
DELIMITER ;
```

Note the two keywords, **OLD** and **NEW**. In an UPDATE trigger, OLD refers to the row before it is updated and NEW refers to the row after it is updated. In an INSERT trigger, however, only NEW is allowed, while in a DELETE trigger, only OLD is allowed as there is no new row.

We can view all the triggers in the database by:

- **SHOW TRIGGERS;**

REMOVING A TRIGGER

The following syntax is used in order to remove an existing trigger.

- **DROP TRIGGER** tb_name.trigger_name;

Note that triggers also need to be removed (i.e. deleted and recreated) if we wish to modify them.

TASK

- Create an AFTER INSERT trigger that inserts the username, current date and time, and department name into a log table every time a new department is created.

LAB ASSIGNMENT

1. Create a BEFORE DELETE trigger that maintains a log of the username of the user performing the deletion, the ID and name of the employee being deleted, and the timestamp of when the action is performed every time an employee is removed from the database.

SUBMISSION GUIDELINES

- Take a screenshot of each task. Ensure that all screenshots have a white background and black text. You can alter the background and text colors through the properties of the MySQL command line client.
- Place all the screenshots in a single word file labeled with Roll No and Lab No. e.g. **'cs181xxx_Lab12'**
- Convert the file into PDF.
- Submit the PDF file at LMS
- **-100%** policies for plagiarism.