A.Y. 2023-2024
Class: SE-ITA/B,

Semester: III Subject:

Structured Query Lab

Experiment 7. – Perform triggers on the chosen system.

1. **Aim:** To Perform triggers on the chosen system.

2. **Objective:** Construct triggers to store, manipulate data in database

3. **Outcome:** L303.4: To Write queries in SQL to retrieve any type of information from a database.

4. **Prerequisite:** Understanding of various operations, trigger, with notations and terminologies along with sample syntax.

5. **Requirements:** MySQL Workbench**,**PC, Oracle 11g/SQL Server 2008 R2, Microsoft Word, Internet.

6. **Pre-Experiment Exercise:**

Brief Theory :(To be hand written)

Explain what are triggers with example

7. **Laboratory Exercise**

A. Procedure:

i)Open MySQL command line client using below login credentials:
Password: Lab306b
ii) Use existing database created by you or
iii) Construct your own database
iv) Construct tables for any two to three entities from your chosen case study
v) Insert at least 8 to 10 records for each tables

Syntax to Create Trigger:

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

TRIGGERS(Any 2)

Execute below INSERT Triggers

```
create database Hospital1

use Hospital1
CREATE TABLE employee(
```

```
name varchar(45) NOT NULL,
occupation varchar(35) NOT NULL,
working_date date,
working_hours varchar(10)
);

INSERT INTO employee VALUES

('Robin', 'Scientist', '2020-10-04', 12),
('Warner', 'Engineer', '2020-10-04', 10),
('Peter', 'Actor', '2020-10-04', 13),
('Marco', 'Doctor', '2020-10-04', 14),
('Brayden', 'Teacher', '2020-10-04', 12),
('Antonio', 'Business', '2020-10-04', 11);

select * from employee;
DELIMITER //
Create Trigger before_insert_empworkinghours
BEFORE INSERT ON employee FOR EACH ROW
BEGIN
IF NEW.working_hours < 0 THEN SET NEW.working_hours = 0;
END IF;
END //

INSERT INTO employee VALUES
('Markus', 'Former', '2020-10-08', 14);
INSERT INTO employee VALUES
('Alexander', 'Actor', '2020-10-012', -13)

show triggers;
```

Create Table Workcenters & workcenterstats:

```
 CREATE TABLE WorkCenters (
   id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  capacity INT NOT NULL
);

   CREATE TABLE WorkCenterStats(
   totalCapacity INT NOT NULL
);
```

Creating Trigger:

```
CREATE TRIGGER before_workcenters_insert
BEFORE INSERT
ON WorkCenters FOR EACH ROW
BEGIN
DECLARE rowcount INT;

SELECT COUNT(*)
INTO rowcount
FROM WorkCenterStats;

IF rowcount > 0 THEN
UPDATE WorkCenterStats
SET totalCapacity = totalCapacity + new.capacity;
ELSE
INSERT INTO WorkCenterStats(totalCapacity)
VALUES(new.capacity);
END IF;

END $$

DELIMITER ;
```

First, insert a new row into the WorkCenter table:

```
INSERT INTO WorkCenters(name, capacity)
VALUES('Mold Machine',100);

SELECT * FROM WorkCenterStats;
```

The trigger has been invoked and inserted a new row into the WorkCenterStats table.

```
INSERT INTO WorkCenters(name, capacity)
VALUES('Packing',200);

SELECT * FROM WorkCenterStats;
```

The trigger has updated the total capacity from 100 to 200 as expected.

DELETE Trigger

To destroy the trigger, use **a DROP TRIGGER** statement.
You must specify the schema name if the trigger is not in the default schema:

```
DROP TRIGGER Hospital1.before_workcenters_insert;
```

Update Trigger:

The following is the syntax to create an AFTER UPDATE trigger in MySQL:

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

Create Student Table:

```
CREATE TABLE students(
id int NOT NULL AUTO_INCREMENT,
name varchar(45) NOT NULL,
 class int NOT NULL,
 email_id varchar(65) NOT NULL,
PRIMARY KEY (id)
  );
```

Insert Some Values in a Table:

```
INSERT INTO students (name, class, email_id)
VALUES ('Stephen', 6, 'stephen@javatpoint.com'),
('Bob', 7, 'bob@javatpoint.com'),
('Steven', 8, 'steven@javatpoint.com'),
('Alexandar', 7, 'alexandar@javatpoint.com');

Select * from Students;
```

Create Students_log Table:

```
CREATE TABLE students_log(
 user varchar(45) NOT NULL,
 descreptions varchar(65) NOT NULL
);
```

AFTER UPDATE **trigger that promotes all students in the next class**, i.e., 6 will be 7, 7 will be 8, and so on. Whenever an updation is performed on a single row in the "**students**" table, a new row will be inserted in the **"students_log**" table.

```
DELIMITER $$

CREATE TRIGGER after_update_studentsInfo
AFTER UPDATE
ON students FOR EACH ROW
BEGIN
   INSERT into students_log VALUES (user(),
   CONCAT('Update Student Record ', OLD.name, ' Previous Class :',
   OLD.class, ' Present Class ', NEW.class));
END $$

DELIMITER ;
```

How to call the AFTER UPDATE trigger?

```
UPDATE students SET class = class + 1;
Select * from Students;
Select * from students_log;
```

8. Post Experimental Exercise

A. Questions:

1)What are the conditions under which we should avoid using Triggers

B. Conclusion:

1. Write what was performed in the experiment
2. Mention few applications of what was studied.
3. Write the significance of the studied topic

9. Result/Observation/Program code: Attach all queries executed code with proper Output

C. References:

[1] Elmasri and Navathe, “Fundamentals of Database Systems”, 5th Edition, PEARSON Education.

[2] Korth, Silberchatz, Sudarshan, “Database System Concepts”, 6th Edition, McGraw – Hill

[3] https://www.w3schools.com/sql/sql_default.asp