Experiment No. 9

TITLE

Write and execute suitable database triggers considering row-level and statement-level triggers.

PROBLEM STATEMENT

Create and implement row-level and statement-level triggers in SQL to monitor and log actions performed on the employees table.

OBJECTIVE

- Understand the use and purpose of SQL triggers.
- Differentiate between row-level and statement-level triggers.
- Implement triggers to monitor INSERT, UPDATE, and DELETE operations.
- Log audit data into a separate table.
- Test trigger functionality using sample data.

PLATFORM REQUIRED

- Operating System: Windows or Linux
- Software/Tools: MySQL / Oracle SQL / PostgreSQL / SQL Developer

THEORY

A trigger is a stored procedure that automatically executes when an event (INSERT, UPDATE, DELETE) occurs on a table.

Types of Triggers:

- **Row-level trigger**: Executes once for each row affected.
- **Statement-level trigger**: Executes once for the entire SQL operation, regardless of row count.

STEP BY STEP ALGORITHM

1. Create base tables: employees and audit_log.
2. Insert sample data into employees.
3. Create a row-level trigger to log updates to salary.
4. Create a statement-level trigger to log general updates.
5. Perform UPDATE operations to test triggers.
6. Query the audit_log to verify trigger actions.

SQL CODE

Step 1: Create Tables

```
CREATE TABLE employees (
  emp_id INT PRIMARY KEY,
  name VARCHAR(50),
  salary DECIMAL(10,2),
  dept VARCHAR(30)
);

CREATE TABLE audit_log (
  log_id INT AUTO_INCREMENT PRIMARY KEY,
  emp_id INT,
  action VARCHAR(100),
  log_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Step 2: Insert Data

```
INSERT INTO employees VALUES (1, 'Alice', 50000, 'HR');
INSERT INTO employees VALUES (2, 'Bob', 60000, 'IT');
```

Step 3: Row-Level Trigger

```
CREATE TRIGGER trg_row_salary_update
AFTER UPDATE ON employees
FOR EACH ROW
WHEN (OLD.salary <> NEW.salary)
BEGIN
  INSERT INTO audit_log(emp_id, action)
  VALUES (OLD.emp_id, 'Salary updated');
END;
```

Step 4: Statement-Level Trigger

```
CREATE TRIGGER trg_statement_update
AFTER UPDATE ON employees
```

```
BEGIN
  INSERT INTO audit_log(emp_id, action)
  VALUES (NULL, 'Employees table updated');
END;
```

Step 5: Perform Updates

```
UPDATE employees SET salary = salary + 1000 WHERE emp_id = 1;
UPDATE employees SET dept = 'Finance' WHERE emp_id = 2;
```

Step 6: View Audit Logs

```
SELECT * FROM audit_log;
```

QUESTIONS

1. What is the difference between row-level and statement-level triggers?
2. Can a trigger call another trigger?
3. What is the importance of the WHEN clause in a trigger?
4. Explain what happens when multiple rows are updated in a single statement.
5. When would you use a BEFORE vs AFTER trigger?
6. What are some practical use cases for triggers in real-world systems?

CONCLUSION

This assignment helps understand the concept of database triggers and their real-world use in auditing and automation. It highlights how row-level triggers log individual record changes and statement-level triggers log event-level operations. Through practical execution, the learner gains insight into when and how to use triggers effectively.