|  |  |  |
| --- | --- | --- |
| **Academic Year:** 2024-25 | **Year:** Second Year | **Semester:** II |
| **PRN No.: 1012412079** | **Name:** Ratnajeet Patil | |
| **Subject:** Database Management System | | |
| **Assignment No.**: 8 |  | |
| **Date:** |  | |

**Lab Assignment: 08**

**Title: Basic Trigger for Audit Logs:**

**Write and Execute Triggers to perform following kind of operations:**

**Theory:**

What is Trigger?

A trigger in a database is a stored procedure that automatically executes when a specified event occurs on a table or view.

Row Level and Statement Level Trigger

1. Row-Level Trigger

* Executes once for each row affected by the triggering event.
* Useful when you want to inspect or manipulate data row-by-row.
* Uses FOR EACH ROW clause.
* Accesses NEW and OLD row data.
* If you insert 5 users, the trigger runs 5 times, once per user.

2. Statement-Level Trigger

* Executes once per SQL statement, regardless of how many rows are affected.
* Default if FOR EACH ROW is not specified.
* Does not have access to individual NEW or OLD row values.
* If you insert 5 users, the trigger runs once **total**

**Execution:**

1. **Create a trigger that logs new user registrations into a separate audit table.**  
   *Table*: users(id, name, email)  
   *Audit Table*: user\_log(id, name, email, action, log\_time)
2. **Write a trigger to ensure that any newly inserted employee has a default department set if not specified.**  
   *Table*: employees(id, name, department\_id)
3. **Create a trigger that automatically calculates and inserts a loyalty point entry when a new order is placed.**  
   *Table*: orders(order\_id, user\_id, amount)  
   *Loyalty Table*: points(user\_id, earned\_points)
4. **Write a trigger that stores the old salary of an employee into a salary history table before it gets updated.**  
   *Table*: employees(id, name, salary)  
   *History Table*: salary\_history(emp\_id, old\_salary, change\_date)
5. **Create a BEFORE UPDATE trigger if anyone tries to update salary of employees.**  
   *Table*: employees(id, name, salary)
6. **Design a trigger that automatically updates the last\_modified field whenever a row is updated.**  
   *Table*: documents(id, title, content, last\_modified)
7. **Create a trigger that saves deleted customer records into a backup table, along with the date when deletion occurs.**  
   *Table*: customers(id, name, email)  
   *Backup Table*: deleted\_customers(id, name, email, deleted\_at)
8. **Write an AFTER DELETE trigger that removes all orders linked to a deleted user.**  
   *Tables*: users(id), orders(order\_id, user\_id)
9. **Create a trigger that logs the deletion of product entries including name and reason.**  
   *Table*: products(id, name)  
   *Log Table*: product\_deletion\_log(product\_id, name, reason, log\_time)

****

****

****

**FAQs:**

1. What are the different types of triggers?

Triggers are classified based on:

A. Event Type:

* BEFORE INSERT
* AFTER INSERT
* BEFORE UPDATE
* AFTER UPDATE
* BEFORE DELETE
* AFTER DELETE

B. Execution Level:

* Row-Level Trigger: Executes once per row affected.
* Statement-Level Trigger: Executes once per SQL statement.

1. How do you create a trigger in MySQL?

1. CREATE TRIGGER trigger\_name

2. [BEFORE | AFTER] [INSERT | UPDATE | DELETE]

3. ON table\_name

4. FOR EACH ROW

5. BEGIN

6. -- trigger logic

7. END;

8.

1. What is the purpose of OLD and NEW in a trigger?

OLD: Refers to the existing row before the operation (used in UPDATE or DELETE).

NEW: Refers to the new row being inserted or updated (used in INSERT or UPDATE).

1. Can triggers be used for audit purposes?

Triggers can log changes to a separate audit table, helping track:

* Who modified data
* When it was modified
* What was changed

1. What are the restrictions on triggers?

* Cannot call COMMIT or ROLLBACK inside a trigger.
* Cannot create or drop tables dynamically inside triggers.
* Recursive trigger calls may be limited or disallowed by default.
* Triggers cannot accept parameters.
* In MySQL, each table can have only one BEFORE and one AFTER trigger per action.