

NAME :- Manish Shashikant Jadhav

UID :- 2023301005.

BRANCH :- Comps -B. BATCH: B.

**EXPERIMENT 9: To examine integrity of database using Triggers.**

**SUBJECT :- DBMS (DATABASE MANAGEMENT SYSTEM)**

## 1. What is trigger in the database? Why are triggers used in databases?

A trigger is a set of instructions that are automatically executed, or "triggered," in response to certain events on a particular table or view. These events can include operations such as INSERT, UPDATE, DELETE, and sometimes even SELECT. Triggers are defined to execute a specified action, such as modifying data in other tables, enforcing business rules, or logging changes.

Here are some key points about triggers and their use in databases:

**Event-Driven Execution:** Triggers are event-driven, meaning they are executed in response to specific events occurring in the database. For example, a trigger may be set to execute after an INSERT operation on a table.

### Types of Triggers:

**BEFORE Triggers:** These triggers execute before the triggering event (e.g., BEFORE INSERT).

**AFTER Triggers:** These triggers execute after the triggering event (e.g., AFTER UPDATE).

**INSTEAD OF Triggers:** These triggers replace the triggering event. For example, instead of performing the default action for an INSERT, an INSTEAD OF trigger can be defined to perform a different action.

### Use Cases:

**Enforcing Business Rules:** Triggers can be used to enforce business rules by checking and validating data before it is inserted, updated, or deleted.

**Referential Integrity:** Triggers can help maintain referential integrity by automatically updating related tables when changes are made to a primary table.

**Logging and Auditing:** Triggers can be employed to log changes made to the database, providing an audit trail for tracking modifications.

**Automated Calculations:** Triggers can perform calculations or generate derived data based on changes in other data.

### Benefits:

**Consistency:** Triggers help ensure data consistency and integrity by automatically enforcing rules and actions.

**Automation:** They automate repetitive tasks, reducing the need for manual intervention and minimizing the chances of human error.

**Security:** Triggers can contribute to security by logging changes and facilitating auditing.

**Considerations:**

**Performance:** Poorly designed triggers can impact performance, so it's important to consider the efficiency of trigger logic.

**Complexity:** Overuse of triggers or complex trigger logic can make the database schema and behavior more difficult to understand and maintain.

- Create a table Uni\_Audit

```
CREATE TABLE Uni_Audit (
    audit_id INT AUTO_INCREMENT PRIMARY KEY,
    table_name VARCHAR(50) NOT NULL,
    pk_values VARCHAR(255) NOT NULL,
    action_type ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,
    old_values VARCHAR(255) NOT NULL,
    change_timestamp TIMESTAMP NOT NULL
);
```

2. Write any five triggers for the university database using MySQL.

1. **AFTER INSERT** : This trigger creates an audit trail by logging information about the newly inserted instructor record in the Uni\_Audit table after the actual insertion takes place. The trigger is using the NEW keyword to access the values of the columns in the new row being inserted.

```
DELIMITER //
CREATE TRIGGER audit_instructor_create_trigger
AFTER INSERT ON instructor
FOR EACH ROW
BEGIN
    INSERT INTO Uni_Audit (table_name, pk_values, action_type, old_values, change_timestamp)
    VALUES ('instructor', NEW.ID, 'INSERT',
            CONCAT('name: ', NEW.name, ', dept_name: ', NEW.dept_name, ', salary: ', NEW.salary), NOW());
END;

DELIMITER //
```

## DBMS EXPERIMENT NO. 9

```
13 • INSERT INTO instructor(ID, name, dept_name, salary) values('10000','Ram','Comp. Sci.', 60000);
14 • SELECT * FROM Uni_Audit;
```

audit_id	table_name	pk_values	action_type	old_values	change_timestamp
1	instructor	10000	INSERT	name: Ram, dept_name: Comp. Sci., salary: 60000.00	2023-11-11 22:38:39
*	NULL	NULL	NULL	NULL	NULL

**2. BEFORE INSERT :** This trigger creates an audit trail by logging information about the new section record in the Uni\_Audit table before the actual insertion takes place. The trigger is using the NEW keyword to access the values of the columns in the new row being inserted.

```
DELIMITER //
```

```
CREATE TRIGGER before_insert_section
```

```
BEFORE INSERT ON section
```

```
FOR EACH ROW
```

```
BEGIN
```

```
INSERT INTO Uni_Audit(table_name, pk_values, action_type, old_values, change_timestamp)
```

```
VALUES ('section', NEW.course_id, 'INSERT',
```

```
CONCAT('Course ID: ', NEW.course_id, ', Section ID: ', NEW.sec_id, ', Semester: ',
```

```
NEW.semester, ', Year: ', NEW.year),NOW());
```

```
END;
```

```
//
```

```
DELIMITER //
```

```
49 • INSERT INTO section
50 values("PHY-101", 3, "Fall", 2010, "Watson", 100, "C");
51 • SELECT * FROM Uni_Audit;
```

audit_id	table_name	pk_values	action_type	old_values	change_timestamp
1	instructor	10000	INSERT	name: Ram, dept_name: Comp. Sci., salary: 60000.00	2023-11-11 22:38:39
2	section	PHY-101	INSERT	Course ID: PHY-101, Section ID: 3, Semester: Fall, Year: 2010	2023-11-11 22:41:28
*	NULL	NULL	NULL	NULL	NULL

**3. AFTER DELETE :** This trigger captures the old values of the deleted row in the 'course' table and inserts them into the Uni\_Audit table. The trigger execution time (change\_timestamp) is also recorded.

```
DELIMITER //
CREATE TRIGGER after_delete_course
AFTER DELETE ON course
FOR EACH ROW
BEGIN
    -- Insert old values into Uni_Audit
    INSERT INTO Uni_Audit (table_name, pk_values, action_type, old_values, change_timestamp)
    VALUES ('course', OLD.course_id, 'DELETE',
            CONCAT('title: ', OLD.title, ', dept_name: ', OLD.dept_name, ', credits: ', OLD.credits), NOW());
END //
DELIMITER ;
```

```
29 • DELETE FROM course WHERE course_id = 'CS-101';
30 • SELECT * FROM Uni_Audit;
```

Result Grid						
audit_id	table_name	pk_values	action_type	old_values	change_timestamp	
1	instructor	10000	INSERT	name: Ram, dept_name: Comp. Sci., salary: 60000.00	2023-11-11 22:38:39	
2	section	PHY-101	INSERT	Course ID: PHY-101, Section ID: 3, Semester: Fall, Year: 2010	2023-11-11 22:41:28	
3	course	CS-101	DELETE	title: Intro. to Computer Science, dept_name: Comp. Sci., credits: 4	2023-11-11 22:46:20	
*	NULL	NULL	NULL	NULL	NULL	

**4. BEFORE DELETE :** This trigger captures the old values of the deleted row in the 'student' table and inserts them into the Uni\_Audit table. The trigger execution time (change\_timestamp) is also recorded.

```
DELIMITER //
CREATE TRIGGER before_delete_student
BEFORE DELETE ON student
FOR EACH ROW
BEGIN
    -- Capture old values and insert into Uni_Audit
    INSERT INTO Uni_Audit (table_name, pk_values, action_type, old_values, change_timestamp)
    VALUES ('student', OLD.ID, 'DELETE',
            CONCAT('name: ', OLD.name, ', dept_name: ', OLD.dept_name, ', tot_cred: ', OLD.tot_cred),
            NOW());
END;
//
DELIMITER ;
```

```
11 • DELETE FROM student WHERE ID = '12345';
12 • SELECT * FROM Uni_Audit;
```

Result Grid						
audit_id	table_name	pk_values	action_type	old_values	change_timestamp	
1	instructor	10000	INSERT	name: Ram, dept_name: Comp. Sci., salary: 60000.00	2023-11-11 22:38:39	
2	section	PHY-101	INSERT	Course ID: PHY-101, Section ID: 3, Semester: Fall, Year: 2010	2023-11-11 22:41:28	
3	course	CS-101	DELETE	title: Intro. to Computer Science, dept_name: Comp. Sci., credits: 4	2023-11-11 22:46:20	
4	student	12345	DELETE	name: Shankar, dept_name: Comp. Sci., tot_cred: 32	2023-11-11 22:48:55	
*	NULL	NULL	NULL	NULL	NULL	

**5. BEFORE UPDATE :** This trigger captures the old values of the 'instructor' table before an update and inserts them into the Uni\_Audit table, including the primary key (ID), action type ('UPDATE'), old values, and the timestamp of the change

```
DELIMITER //
CREATE TRIGGER before_update_instructor
BEFORE UPDATE ON instructor
FOR EACH ROW
BEGIN
    INSERT INTO Uni_Audit (table_name, pk_values, action_type, old_values, change_timestamp)
    VALUES ('instructor', OLD.ID, 'UPDATE',
            CONCAT('name: ', OLD.name, ', dept_name: ', OLD.dept_name, ', salary: ', OLD.salary),
            NOW());
END; //
```

```
26 • UPDATE instructor
27   SET salary = 75000
28   WHERE ID = '10101';
29 • SELECT * FROM Uni_Audit;
```

Result Grid						
Filter Rows:						
	audit_id	table_name	pk_values	action_type	old_values	change_timestamp
▶	1	instructor	10000	INSERT	name: Ram, dept_name: Comp. Sci., salary: 60000.00	2023-11-11 22:38:39
	2	section	PHY-101	INSERT	Course ID: PHY-101, Section ID: 3, Semester: Fall, Year: 2010	2023-11-11 22:41:28
	3	course	CS-101	DELETE	title: Intro. to Computer Science, dept_name: Comp. Sci., credits: 4	2023-11-11 22:46:20
	4	student	12345	DELETE	name: Shankar, dept_name: Comp. Sci., tot_cred: 32	2023-11-11 22:48:55
	5	instructor	10101	UPDATE	name: Srinivasan, dept_name: Comp. Sci., salary: 65000.00	2023-11-11 22:51:34
*	NULL	NULL	NULL	NULL	NULL	NULL

**Conclusion:** Hence by completing this experiment I came to know about triggers in database.