



UNIVERSITY INSTITUTE OF ENGINEERING

Advanced Database Management System Experiment 7

23CSP-333

Submitted To: Faculty Name: Er. Alok Kumar **Submitted By:**

Name:

Himanshu Gupta UID: 23BCS10889 Section: KRG - 2B

Semester: 5th

Aim:

Medium Level:

To design a trigger on the student table that prints the inserted or deleted row to the console whenever an insert or delete operation occurs.

Hard Level:

To design a trigger on the tbl_employee table that logs every insertion or deletion into an audit table tbl_employee_audit with a message containing employee name and timestamp.

Procedure:

Medium Level (Student Table Trigger)

- 1. Create a trigger function fn_student_audit() that handles INSERT and DELETE operations.
- 2. Use RAISE NOTICE inside the function to display the newly inserted (NEW) or deleted (OLD) row.
- 3. Attach the trigger function to the student table for INSERT and DELETE operations.
- 4. Test the trigger by inserting and deleting student records.

Hard Level (Employee Audit Trigger)

- 1. Create two tables: tbl employee and tbl employee audit.
- 2. Write a trigger function audit_employee_changes() that inserts a message into tbl_employee_audit whenever an employee is added or deleted.
- 3. The message should include the employee name and the current timestamp.
- 4. Attach the trigger function to tbl_employee for AFTER INSERT OR DELETE operations.
- 5. Test the trigger by inserting and deleting employee records, then check the audit log.

Code:

Medium Level: Student Trigger

```
CREATE TABLE student (
   id SERIAL PRIMARY KEY,
   name VARCHAR(100),
   age INT,
   class INT
);

CREATE OR REPLACE FUNCTION
fn_student_audit()
RETURNS TRIGGER
LANGUAGE plpgsql AS
$$
BEGIN
   IF TG OP = 'INSERT' THEN
```

```
Discover. Learn. Empower.
    RAISE NOTICE 'Inserted Row ->
ID: %, Name: %, Age: %, Class: %',
       NEW.id, NEW.name, NEW.age,
NEW.class:
    RETURN NEW;
  ELSIF TG OP = 'DELETE' THEN
    RAISE NOTICE 'Deleted Row ->
ID: %, Name: %, Age: %, Class: %',
       OLD.id, OLD.name, OLD.age,
OLD.class;
    RETURN OLD;
  END IF;
  RETURN NULL;
END;
$$;
CREATE TRIGGER trg student audit
AFTER INSERT OR DELETE
ON student
FOR EACH ROW
EXECUTE FUNCTION
fn_student_audit();
INSERT INTO student(name, age, class)
```

Hard Level: Employee Audit Trigger

DELETE FROM student WHERE name

VALUES ('Himanshu', 18, 10);

= 'Himanshu';

```
CREATE TABLE tbl_employee (
emp_id SERIAL PRIMARY KEY,
emp_name VARCHAR(100) NOT NULL,
emp_salary NUMERIC
);

CREATE TABLE tbl_employee_audit (
sno SERIAL PRIMARY KEY,
message TEXT
);

CREATE OR REPLACE FUNCTION audit_employee_changes()
RETURNS TRIGGER
LANGUAGE plpgsql AS
$$
BEGIN
IF TG_OP = 'INSERT' THEN
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
Discover. Learn. Empower.
    INSERT INTO tbl employee audit(message)
    VALUES ('Employee name ' || NEW.emp name || ' has been added at ' || NOW());
    RETURN NEW;
  ELSIF TG OP = 'DELETE' THEN
    INSERT INTO tbl employee audit(message)
    VALUES ('Employee name ' || OLD.emp name || ' has been deleted at ' || NOW());
    RETURN OLD:
  END IF;
  RETURN NULL;
END;
$$:
CREATE TRIGGER trg employee audit
AFTER INSERT OR DELETE
ON tbl employee
FOR EACH ROW
EXECUTE FUNCTION audit employee changes();
INSERT INTO tbl employee(emp name, emp salary)
VALUES ('Himanshu', 50000);
DELETE FROM tbl employee
WHERE emp name = 'Himanshu';
```

Output:

SELECT * FROM tbl employee audit;

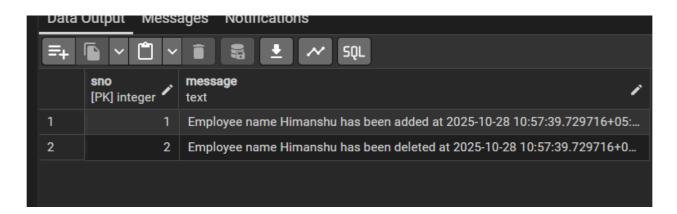
```
Data Output Messages Notifications

NOTICE: Inserted Row -> ID: 1, Name: Himanshu, Age: 18, Class: 10

NOTICE: Deleted Row -> ID: 1, Name: Himanshu, Age: 18, Class: 10

DELETE 1

Query returned successfully in 52 msec.
```





Conclusion:

This experiment demonstrated the use of **PostgreSQL triggers** to automate database actions and maintain audit logs.

- In the **medium-level problem**, the trigger displayed inserted or deleted student rows immediately in the console, providing real-time feedback.
- In the **hard-level problem**, the trigger recorded every insertion and deletion of employees into an audit table with timestamps, ensuring traceability and data integrity.

Overall, triggers help in automating tasks, auditing changes, and maintaining consistent and reliable database operations without manual intervention.