Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2024), B.Sc. in CSE (Day)

Lab Report NO: 01

Course Title: Database

Course Code: CSE 209 Section: 231_D1

Assignment Name: Trigger and Function

Student Details

	Name	ID
1.	Promod Chandra Das	231002005

Submission Date : 17.12.2024

Course Teacher's Name : Dr. Faiz Al Faisal

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

1. TITLE OF THE ASSIGNMENT NAME :

Trigger and Function

2. OBJECTIVES :

1. **Understand Database Automation**
   - Learn how to create **triggers** that respond to specific database events (e.g., INSERT, UPDATE, DELETE).
   - Understand how **functions** can encapsulate reusable logic that triggers can invoke.
2. **Enhance Data Integrity**
   - Automatically enforce rules and constraints (e.g., keeping logs, preventing invalid operations).
   - Ensure that critical business processes are executed without manual intervention.
3. **Practice Advanced SQL Concepts**
   - Gain proficiency in procedural SQL by writing functions.
   - Understand how triggers and functions work together to automate backend processes.
4. **Implement Real-World Use Cases**
   - Simulate scenarios where triggers and functions are used for:
     - Auditing changes in data (e.g., stock updates, salary changes).
     - Ensuring consistency across tables (e.g., cascading updates, validations).
     - Automating notifications or alerts.
5. **Demonstrate Practical Skills**
   - Showcase the ability to design a database with advanced features.
   - Highlight skills useful in real-world database administration and application development.

1. Schema for Employees and Orders

Assuming a simplified schema for Employees and Orders:

Employees Table:

```
CREATE TABLE Employees (
    EmployeeID SERIAL PRIMARY KEY,
    EmployeeName VARCHAR(100) NOT NULL
);
```

Orders Table:

```
CREATE TABLE Orders (
    OrderID SERIAL PRIMARY KEY,
    EmployeeID INT NOT NULL,
    OrderAmount DECIMAL(10, 2) NOT NULL,
    OrderDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

---

2. Create Backup Table

We will create a backup table for Orders named Backup_OrdersTBL.

```
CREATE TABLE Backup_OrdersTBL (
    BackupID SERIAL PRIMARY KEY,
    ActionType VARCHAR(10) NOT NULL, -- Indicates INSERT, UPDATE, or DELETE
    OrderID INT,
    EmployeeID INT,
    OrderAmount DECIMAL(10, 2),
    OrderDate TIMESTAMP,
    ChangeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- When the action occurred
);
```

---

3. Create the Trigger Function

The trigger function will log changes from the Orders table into the Backup_OrdersTBL.

```
CREATE OR REPLACE FUNCTION BackupOrdersFN()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        INSERT INTO Backup_OrdersTBL (ActionType, OrderID, EmployeeID, OrderAmount,
    OrderDate)
        VALUES ('INSERT', NEW.OrderID, NEW.EmployeeID, NEW.OrderAmount,
    NEW.OrderDate);

    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO Backup_OrdersTBL (ActionType, OrderID, EmployeeID, OrderAmount,
    OrderDate)
        VALUES ('UPDATE', NEW.OrderID, NEW.EmployeeID, NEW.OrderAmount,
    NEW.OrderDate);
```

```
    ELSIF (TG_OP = 'DELETE') THEN
        INSERT INTO Backup_OrdersTBL (ActionType, OrderID, EmployeeID, OrderAmount,
    OrderDate)
        VALUES ('DELETE', OLD.OrderID, OLD.EmployeeID, OLD.OrderAmount,
    OLD.OrderDate);
    END IF;

    RETURN NULL; -- Triggers of type AFTER return NULL
END;
$$ LANGUAGE plpgsql;
```

---

4. Create the Trigger

The trigger will invoke the BackupOrdersFN() function after any INSERT, UPDATE, or DELETE operation on the Orders table.

```
CREATE TRIGGER BackupOrdersTRG
AFTER INSERT OR UPDATE OR DELETE ON Orders
FOR EACH ROW
EXECUTE FUNCTION BackupOrdersFN();
```

---

5. Demonstrate the Trigger

Insert Data into Orders

```
INSERT INTO Employees (EmployeeName) VALUES ('John Doe');
INSERT INTO Orders (EmployeeID, OrderAmount) VALUES (1, 100.50);
```

Verify the Backup Table

```
SELECT * FROM Backup_OrdersTBL;
```

Expected Output:

BackupID	ActionType	OrderID	EmployeeID	OrderAmount	OrderDate	ChangeDate
1	INSERT	1	1	100.50	2024-12-15 11:00:00	2024-12-15 11:00:00

Update Data in Orders

```
UPDATE Orders SET OrderAmount = 120.75 WHERE OrderID = 1;
```

Verify the Backup Table

```
SELECT * FROM Backup_OrdersTBL;
```

Expected Output:

BackupID	ActionType	OrderID	EmployeeID	OrderAmount	OrderDate	ChangeDate
1	INSERT	1	1	100.50	2024-12-15 11:00:00	2024-12-15 11:00:00
2	UPDATE	1	1	120.75	2024-12-15 11:00:00	2024-12-15 11:05:00

Delete Data from Orders

```
DELETE FROM Orders WHERE OrderID = 1;
```

Verify the Backup Table

```
SELECT * FROM Backup_OrdersTBL;
```

Expected Output:

BackupID	ActionType	OrderID	EmployeeID	OrderAmount	OrderDate	ChangeDate
1	INSERT	1	1	100.50	2024-12-15 11:00:00	2024-12-15 11:00:00
2	UPDATE	1	1	120.75	2024-12-15 11:00:00	2024-12-15 11:05:00
3	DELETE	1	1	120.75	2024-12-15 11:00:00	2024-12-15 11:10:00

---

6. Explanation

1. Backup Table (Backup_OrdersTBL):
    - Stores a record of changes (INSERT, UPDATE, DELETE) from the Orders table.

2. Trigger Function (BackupOrdersFN):
    - Captures the type of operation (TG_OP) and logs the respective details.
3. Trigger (BackupOrdersTRG):
    - Ensures every change to the Orders table is logged in the backup table.

- **DISCUSSION :**

Triggers and functions in databases enable automation, enforce business rules, and maintain data integrity. Triggers act on events like INSERT or UPDATE, while functions encapsulate reusable logic. Together, they streamline workflows, such as logging changes or validating data. Proper design ensures efficiency, but overuse can lead to performance issues and complexity.