**DBMS Practical No: 7**

**# Write a PL/SQL Code block using Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).**

**Step 1: Connect to Your Oracle Database**

Ensure you have access to an Oracle database and a tool like SQL\*Plus or SQL Developer to execute SQL and PL/SQL code.

**Step 2: Create Sample Tables**

For demonstration purposes, let's create two sample tables - "employees" and "departments."

**sql**

**-- Create "employees" table**

**CREATE TABLE employees (**

**employee\_id NUMBER PRIMARY KEY,**

**first\_name VARCHAR2(50),**

**last\_name VARCHAR2(50),**

**salary NUMBER,**

**created\_date DATE**

**);**

**-- Create "departments" table**

**CREATE TABLE departments (**

**department\_id NUMBER PRIMARY KEY,**

**department\_name VARCHAR2(50)**

**);**

**-- Insert sample data**

**INSERT INTO employees VALUES (1, 'John', 'Doe', 50000, TO\_DATE('01-JAN-2022', 'DD-MON-YYYY'));**

**INSERT INTO departments VALUES (1, 'HR');**

**COMMIT;**

**Step 3: Create Row-Level BEFORE Trigger**

In this step, we'll create a Row-Level BEFORE trigger that updates the "created\_date" column when a new employee is inserted.

**sql**

**-- Row-Level BEFORE Trigger**

**CREATE OR REPLACE TRIGGER before\_insert\_employee**

**BEFORE INSERT ON employee**

**FOR EACH ROW**

**BEGIN**

**:NEW.created\_date := SYSDATE;**

**END;**

**/**

This trigger sets the "created\_date" to the current date and time for each new employee during the insertion.

**Step 4: Create Row-Level AFTER Trigger**

Now, let's create a Row-Level AFTER trigger that logs changes in employee salaries in a separate table named "salary\_changes" after a salary update occurs.

**sql**

**-- Row-Level AFTER Trigger**

**CREATE OR REPLACE TRIGGER after\_update\_salary**

**AFTER UPDATE OF salary ON employee**

**FOR EACH ROW**

**BEGIN**

**IF :NEW.salary > :OLD.salary THEN**

**INSERT INTO salary\_changes (employee\_id, old\_salary, new\_salary, change\_date)**

**VALUES (:OLD.employee\_id, :OLD.salary, :NEW.salary, SYSDATE);**

**END IF;**

**END;**

**/**

This trigger records salary changes in the "salary\_changes" table after a salary update.

**Step 5: Create Statement-Level BEFORE Trigger**

In this step, we'll create a Statement-Level BEFORE trigger that checks if the user executing the delete statement is 'HR\_ADMIN.' If not, it raises an application error to prevent the delete operation.

**sql**

**-- Statement-Level BEFORE Trigger**

**CREATE OR REPLACE TRIGGER before\_delete\_employees**

**BEFORE DELETE ON employee**

**DECLARE**

**BEGIN**

**IF USER != 'HR\_ADMIN' THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Only HR\_ADMIN can delete employees.');**

**END IF;**

**END;**

**/**This trigger enforces a rule that only 'HR\_ADMIN' can delete employees.

**Step 6: Create Statement-Level AFTER Trigger**

Lastly, let's create a Statement-Level AFTER trigger that logs information about newly created departments in a table named "department\_audit" after an insert operation.

**Sql**

**-- Statement-Level AFTER Trigger**

**CREATE OR REPLACE TRIGGER after\_create\_department**

**AFTER INSERT ON departments**

**DECLARE**

**BEGIN**

**INSERT INTO department\_audit (department\_name, operation\_type, audit\_date)**

**VALUES ('New Department', 'INSERT', SYSDATE);**

**END;**

**/**

This trigger records information about new departments in the "department\_audit" table after an insert operation.

**Step 7: Execute Triggers**

You don't need to execute triggers manually; they are automatically triggered when the associated events occur. For example:

* **The "before\_insert\_employee" trigger will run when you insert a new employee.**
* **The "after\_update\_salary" trigger will run when you update an employee's salary.**
* **The "before\_delete\_employees" trigger will run when you attempt to delete an employee (assuming you're not 'HR\_ADMIN').**
* **The "after\_create\_department" trigger will run when you insert a new department.**

These triggers will automatically execute when the respective events occur in your database. You can verify their behavior by performing relevant actions, such as inserting new data or updating records in the "employees" and "departments" tables.