Ex.No.: 13		WORKING WITH TRIGGER
Date:	11/10/24	TRIGGER

DEFINITION

A trigger is a statement that is executed automatically by the system as a side effect of a modification to the database. The parts of a trigger are,

- **Trigger statement**: Specifies the DML statements and fires the trigger body. It also specifies the table to which the trigger is associated.

- **Trigger body or trigger action**: It is a PL/SQL block that is executed when the triggering statement is used.

- **Trigger restriction**: Restrictions on the trigger can be achieved

The different uses of triggers are as follows,

- *To generate data automatically*
- *To enforce complex integrity constraints*
- *To customize complex securing authorizations*
- *To maintain the replicate table*
- To audit data modifications

TYPES OF TRIGGERS

The various types of triggers are as follows,

- **Before**: It fires the trigger before executing the trigger statement.

- **After**: It fires the trigger after executing the trigger statement
- .
- **For each row**: It specifies that the trigger fires once per row
- .
- **For each statement**: This is the default trigger that is invoked. It specifies that the trigger fires once per statement.

VARIABLES USED IN TRIGGERS

- :new

- :old

Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
CREATE OR REPLACE TRIGGER Prevent_Parent_deletion
BEFORE DELETE ON PARENT
FOR EACH ROW
DECLARE
    child_count NUMBER;
BEGIN
  SELECT COUNT(*) into child_count From child
                  Whou Parent_id
If child_count > 0 Then RAISE_APPLICATION_ERROR
END;
```

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE TABLE SampleTable (
    id number(5) PRIMARY KEY,
    name varchar(50) NULL,
    email varchar2(100) UNIQUE );
CREATE OR REPLACE TRIGGER Check_duplicate_email
BEFORE Insert OR Update on Sampletable
FOR EACH ROW
DECLARE
    duplicate_count NUMBER *
BEGIN
    SELECT COUNT(*) INTO duplicate_count
 END IF;
END;
```

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
CREATE OR REPLACE TRIGGER restrict_total_sales
BEFORE INSERT ON SALES
FOR EACH ROW
BEGIN
   IF (SELECT SUM(amount) FROM Sales)+ : new amount>10000
   RAISE_APPLICATION_ERROR (-20002), 'Total exceeds
                                      threshold.');
 END IF;'
END;
```

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE OR REPLACE TRIGGER .log_salary_changes
AFTER UPDATE OF SALARY ON EMPLOYEES
 FOR EACH ROW
  BEGIN
INSERT INTO EmployeeAudit VALUES (audit_seq.NEXTVAL,
     :OLD.emp_id ,:OLD:Salary, :New.Salary, SYSDATE),'

  END ;'
```

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
CREATE OR REPLACE TRIGIGER record_user_activity
AFTER INSERT OR UPDATE OR DELETE ON Employees For
                                        EACH ROW
                    |
BEGIN
  INSERT INTO Audit log VALUES(audit_seq.NEXTVAL,
CASE WHEN INSERTING THEN 'INSERT' WHEN UPDATING
                              THEN 'UPDATE'
'Employees', NVL(:OLD.emp_id, :NEW:emp_id), SYSDATE, USER);
END;
```

Program 7

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE TABEL Sales(
      Sale_id NUMBER PRIMARY KEY,
amount NUMBER (10,2),
      running_total number (10,2)
      );
CREATE OR REPLACE TRIGIGER UPDATE_running total
FOR EACH ROW
BEGIN
SELECT NVL(MAX(running_total,0)+ :NEW.amount
              INTO :NEW.running );
END;
```

Program 8

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE OR REPLACE TRIGGIER Validate_stock_before_order
BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
    IF :New.order_quantity > (SELECT stock_quantity
                              FROM items
    WHERE item_id = :NEW.item_id;
    END IF;
    END;
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	