Oracle Triggers: Practice Questions and Answers

1. What is a trigger in Oracle? List the types of triggers available in Oracle databases.

Answer:

A trigger in Oracle is a stored procedure that is automatically executed in response to certain events on a particular table or view.

Types of triggers in Oracle:

- DML Triggers: Before/After INSERT, UPDATE, DELETE operations.
- Instead-of Triggers: Specifically for views.
- System/Event Triggers: Triggered by database/system events such as logon, shutdown.

2. Write the basic syntax to create a BEFORE INSERT trigger.

Answer:

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE INSERT ON table_name
FOR EACH ROW
BEGIN
    -- Trigger logic here
END;
/
```

3. What is the difference between statement-level and row-level triggers?

Answer:

- Statement-level triggers: Executed once for the triggering event, irrespective of the number of rows affected.
- Row-level triggers: Executed for each row that is affected by the triggering statement. Uses :OLD and :NEW pseudo-records.

4. Write a BEFORE UPDATE trigger to log updates on an employee table into a log table.

Answer:

```
CREATE OR REPLACE TRIGGER log_employee_updates

BEFORE UPDATE ON EMPLOYEES

FOR EACH ROW

BEGIN

   INSERT INTO EMP_LOGS (EMP_ID, UPDATE_DATE, UPDATED_BY)

   VALUES (:OLD.EMP_ID, SYSDATE, USER);

END;

/
```

5. Write a trigger that prevents deletion of records in a table named ORDERS if the status is 'PENDING'.

Answer:

```
CREATE OR REPLACE TRIGGER prevent_pending_deletions

BEFORE DELETE ON ORDERS

FOR EACH ROW

BEGIN

   IF :OLD.STATUS = 'PENDING' THEN

      RAISE_APPLICATION_ERROR(-20001, 'Cannot delete orders with status PENDING.');

   END IF;

END;

/
```

6. What is a mutating table error, and how can you avoid it in a trigger?

Answer:

A mutating table error occurs when a trigger tries to query or modify the table that caused the trigger to fire. This is common with row-level triggers.

Avoidance techniques:

- Use compound triggers.

- Use an autonomous transaction to handle changes.

- Use a temporary table to store intermediate results.

7. Write a compound trigger to maintain an audit trail for the PRODUCTS table.

Answer:

```
CREATE OR REPLACE TRIGGER product_audit_compound
FOR INSERT OR UPDATE OR DELETE ON PRODUCTS
COMPOUND TRIGGER
   TYPE audit_rec IS RECORD (
      operation VARCHAR2(10),
      product_id NUMBER,
      old_value VARCHAR2(100),
      new_value VARCHAR2(100),
      timestamp TIMESTAMP
   );
   audit_table audit_rec;
   BEFORE EACH ROW IS
   BEGIN
      IF INSERTING THEN
         audit_table.operation := 'INSERT';
         audit_table.new_value := :NEW.PRODUCT_NAME;
      ELSIF UPDATING THEN
         audit_table.operation := 'UPDATE';
         audit_table.old_value := :OLD.PRODUCT_NAME;
```

```
            audit_table.new_value := :NEW.PRODUCT_NAME;

        ELSIF DELETING THEN

            audit_table.operation := 'DELETE';

            audit_table.old_value := :OLD.PRODUCT_NAME;

        END IF;

        audit_table.timestamp := SYSTIMESTAMP;

    END BEFORE EACH ROW;

    AFTER STATEMENT IS

    BEGIN

        INSERT INTO PRODUCTS_AUDIT VALUES (audit_table.*);

    END AFTER STATEMENT;

END;

/
```

8. Explain the differences between a trigger and a stored procedure.

Answer:

- A trigger is event-driven and automatically executed in response to specific events. A stored procedure is explicitly invoked by the user.
- Triggers are tied to tables or views, while stored procedures can exist independently.
- Triggers are not directly invoked, whereas stored procedures can be called multiple times with varying parameters.

9. Create a database-level trigger to log user login/logout events.

Answer:

```
CREATE OR REPLACE TRIGGER log_user_activity

AFTER LOGON OR LOGOFF ON DATABASE

BEGIN
```

```
    IF USERENV('SESSIONID') IS NOT NULL THEN
        INSERT INTO DB_ACTIVITY_LOG (USERNAME, EVENT_TYPE, EVENT_TIMESTAMP)
        VALUES (USER, SYS_CONTEXT('USERENV', 'EVENT'), SYSTIMESTAMP);
    END IF;
END;
/
```

10. Write an INSTEAD OF INSERT trigger for a view.

Answer:

```
CREATE OR REPLACE TRIGGER insert_employee_details_view
INSTEAD OF INSERT ON EMPLOYEE_DETAILS_VIEW
FOR EACH ROW
BEGIN
    INSERT INTO EMPLOYEES (EMP_ID, NAME) VALUES (:NEW.EMP_ID, :NEW.NAME);
    INSERT INTO DEPARTMENTS (DEPT_ID, EMP_ID) VALUES (:NEW.DEPT_ID, :NEW.EMP_ID);
END;
/
```