TRIGGERS

Q. 1 Compare and contrast triggers and procedures.
Ans. Both, triggers and procedures are named PL/SQL blocks that are stored in the database schema. The differences between trigger and procedures are as follows:
(i)   Procedures need to be invoked explicitly whereas triggers are invoked automatically when a triggering event take place.
(ii) Procedures can take place parameters whereas triggers can not.

Q.2 Compare and contrast triggers and constraints.
Ans. Both, triggers and constraints can be used to maintain database integrity. The differences between the two are as follows:
(i)   Triggers affect only those row insertions that take place after the trigger is created whereas a constraints can be made to affect all rows in the table, even if they existed before the constraints was created.
(ii) Triggers can be used to define custom constraints, which might not be possible through inbuilt constraints.
(iii) Apart from enforcing constraints, triggers can also be used for database replication, automatic column value generation etc.

Q.3 When do you create are INSTEAD OF triggers?
Ans. An INSTEAD OF trigger is one that Oracle fires instead of executing the triggering statement. Thus triggers are created on views rather than tables. Thus when we want to enforce certain functionality on views, we need to create INSTEAD OF triggers.

Q.4 Find the errors from the following PL/SQL code and rewrite the corrected code underlining the correction made:

```
CREATE ASWELLAS REPLACE TRIGGER DEPT_UP
AFTER UPDATE ON DEPT FOR EVERY ROW
DECLARE
V_num  NUMBER(3);
BEGIN
              SELECT COUNT(*) INTO V_num FROM emp WHERE Deptno = '101';
IF V_num >5
                     Raise_Application _Error (-20001, 'Cannot exceed 5');
END;
```

Ans :

```
CREATE OR REPLACE TRIGGER DEPT_UP
AFTER UPDATE ON DEPT FOR EACH ROW
DECLARE
V_num  NUMBER(3);
BEGIN
              SELECT COUNT(*) INTO V_num FROM emp WHERE Deptno = 101;
IF V_num >5 THEN
                     Raise_Application _Error (-20001, 'Cannot exceed 5');
END IF;
END;
```

Q.5 What is a trigger? Name two types of triggers available in PL/SQL.
Ans : A trigger is a stored procedure that defines an action that the database should take when some database related event (such as insert, update, delete) occurs. Thus, database trigger is a set of PL/SQL statements that executes each time an event (such as insert, update, delete) occurs on the database.

Two types of triggers available in PL/SQL are-

1. Row Level Triggers
2. Statement Level Triggers

Q.6 Write the difference between a cursor and a trigger.
Ans :
A cursor is a temporary memory area that stores the result of execution of an SQL statement known as result set, whereas a trigger is a PL/SQL block that is invoked automatically when a triggering SQL statement executes.

A cursor is used to process the result of SQL query row by row and a trigger is used to enforce, a constraints/ specific functionality linked to the execution of a DML statement.

Q.7 Create a trigger that raises an error if a user attempts to delete a row from the employee table.
Ans.
```
CREATE OR REPLACE TRIGGER trg
BEFORE DELETE ON emp
FOR EACH ROW
BEGIN
            RAISE_APPLICATION_ERROR (-20000, 'CAN NOT DELETE');
END;
```

Q.8 What are mutating tables and constraining tables? How are these used by row level triggers?
Ans. Mutating table is the one that is currently being modified by a DML statement, whereas a constraining table is the one that defines the domain for the values of a foreign key column of the mutating table.
A trigger can not read from or modify any of the values of the mutating table but can not read from or modify the non key values of the constraining table.

Q.9 Write statements to do following:
(i) Delete trigger chkCost
(ii) List the names of triggers that you created.
(iii) Disable trigger dispCost.
(iv) Enable trigger highCost
(v) You created a trigger, which was not compiled successfully. Write statement to view the list of errors.

Ans.
```
(i)   DROP TRIGGER chkCost;
(ii)  SELECT TRIGGER_NAME FROM USER_TRIGGERS;
(iii) ALTER TRIGGER dispCost DISABLE;
(iv)  ALTER TRIGGER highCost ENABLE;
(v)   SHOW ERRORS <Trigger name>
```

Q.10 Explain following trigger parts with example.
(i) Triggering type
(ii) Triggering event

(iii) Trigger restriction
(iv) Trigger body

Ans.
(i) **Triggering type** : Trigger type can be BEFORE or AFTER and decides whether is the trigger fired before or after the triggering statement.
(ii) **Triggering event**: Triggering event can be INSERT or UPDATE or DELETE or a combination of all, and decides which DML statements fire the trigger.
(iii) **Trigger restriction**: The condition written with the WHEN clause, which must evaluate to true before if the trigger is to be fired defines the trigger restriction.
(iv) **Trigger body** : The PL/SQL code associated with the trigger that is executed when the triggering event occurs is referred to as trigger body.

Q. 11 Consider the table definitions:
INVENTORY (itemcode, category, name, stock, unitprice, reorder_level)
BILL (billno, itemcode, saledate, unitsold)

Create a trigger that reduces stock by the number of units sold each time a purchase is made (i.e. a record is added to the table BILL) provided there is enough stock in the inventory. The trigger should not allow the insertion if this condition does not hold good.
Ans.
```
CREATE OR REPLACE TRIGGER trg
AFTER INSERT ON BILL
FOR EACH ROW
DECLARE
                Nstock  INVENTORY.stock%TYPE;
BEGIN
                SELECT stock INTO Nstock FROM INVENTORY
 WHERE itemcode = :NEW.itemcode;
IF :NEW.unitsold<Nstock THEN
        UPDATE INVENTORY SET stock = stock - : NEW.unitsold
WHERE itemcode = :NEW.itemcode;
        ELSE
                        RAISE_APPLICATION_ERROR (-20000, ‘NOT ENOUGH STOCK’);
                END IF;
END;
```

Q.12 Create a trigger that prints the change in salary every time salary of an employee is changed.
Ans.
```
CREATE OR REPLACE TRIGGER trg
BEFORE DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (:NEW.empno>0)
DECLARE
                SAL_DIFF NUMBER;
BEGIN
                SAL_DIFF : = :NEW.sal - :OLD.sal;
                DBMS_OUTPUT.PUT (‘DIFFERENCE’ || SAL_DIFF);
END;
```

Q. 13 Create a trigger that displays the number of employees after every delete in emp table.
Ans.
```
CREATE OR REPLACE TRIGGER trg
```

```
AFTER DELETE ON emp
FOR EACH ROW
DECLARE
                N  NUMBER;
BEGIN
                SELECT COUNT(*) INTO N FROM emp;
                DBMS_OUTPUT.PUT (‘THERE ARE NOW’ || N || ‘EMPLOYEES’);
END;
```

Q. 14 Write PL/SQL code to create two statement level triggers B_D_Flight and A_D_Flight before and after DELETE statement respectively on the table Flight which displays the message ‘Ready for Deletion’ and ‘Records Deleted’ respectively.

Ans :

```
CREATE OR REPLACE TRIGGER B_D_Flight
        BEFOR DELETE ON Flight
        BEGIN
                DBMS_OUTPUT.PUT_LINE (‘Ready for Deletion’);
END;

CREATE OR REPLACE TRIGGER A_D_Flight
        AFTER DELETE ON Flight
        BEGIN
                DBMS_OUTPUT.PUT_LINE (‘Records Deleted’);
END;
```

Q.15 : Differentiate between Row –level and Statement level triggers in PL/SQL.

OR

Q. Differentiate between Row –level and Statement level triggers in PL/SQL. Give example of each trigger.

Ans : The row level triggers fire once for every single row processed by the DML statement (Insert/Update/Delete). If the DML statement (Insert/Update/Delete) for which the trigger has been defined, affects 20 rows then the row level trigger will get executed twenty times i.e. once for each row.

The statement level trigger is fired once on behalf of triggering statement, regardless of the number of rows affected in the table. If the DML statement (Insert/Update/Delete) for which the trigger has been defined, affects 20 rows then the statement level trigger will get executed only once.

A row level trigger is identified by the FOR EACH ROW clause in the CREATE TRIGGER command. The omission of FOR EACH ROW clause in the CREATE TRIGGER command, makes the trigger the statement level trigger.

Example of Row Level Trigger –

```
CREATE OR REPLACE TRIGGER DEPT_UP
AFTER UPDATE ON DEPT FOR EACH ROW
DECLARE
V_num  NUMBER(3);
BEGIN
        SELECT COUNT(*) INTO V_num FROM emp WHERE Deptno = 101;
IF V_num >5 THEN
        Raise_Application _Error (-20001, ‘Cannot exceed 5’);
END IF;
END;
```

Example of Row Level Trigger –

```
CREATE OR REPLACE TRIGGER B_D_Flight
        BEFOR DELETE ON Flight
```

```
          BEGIN
                    DBMS_OUTPUT.PUT_LINE ('Ready for Deletion');
END;
```

Q.16 Write PL/SQL code to create trigger to display the HELLO message before insert operation on EMPLOYEE table.
Ans :

```
          CREATE OR REPLACE TRIGGER Insert_Emp
                    BEFOR INSERT ON EMPLOYEE
                    FOR EACH ROW
                    BEGIN
                              DBMS_OUTPUT.PUT_LINE ('HELLO');
          END;
```