Experiment No: 07

Aim:

To write and execute SQL programs that allows enforcement of business rules with database triggers.

Problem Statement:

Using the relation schema established in Experiment – 02, 03, 05 create and execute SQL programs that allow enforcement of business rules with database triggers.

Author : Pravesh Mukesh Dholwani
Semester : 5 CSE
Roll No. : 108
Date : 10-OCT-21

Queries &Outcomes:

==========

****

Query 1: Write SQL code to compile and execute a trigger - UPDATE_CUST_BALANCE_TRG that will update the BALANCE in the CUSTOMER table when a new LINE record is entered. (Assume that the sale is a credit sale.) The BALANCE in CUSTOMER is 0 when customer does not have any invoice to his credit. Test the trigger, using the following new LINE record: 1806, 5, PP101', 18, 5.87.

```
SQL> SELECT * FROM LINE WHERE INV_NUM= 1006;

SQL> SELECT * FROM INVOICE WHERE INV_NUM= 1006;

SQL> SELECT * FROM CUSTOMER WHERE C_CODE = 10014;

SQL>INSERT INTO LINE VALUES(1006,5,'PP101',10,5.87);

SQL>SELECT * FROM CUSTOMER WHERE C_CODE = 10014;
/*
Create a table - SALARY_CHANGES with following composition -

      OP_TYPE VARCHAR2 10 REQUIRED
      OP_DATE DATE  DEFAULT SYSDATE
      OP_TIME CHAR  -9   DEFAULT 'HH:MI:SS' FROM SYSTIMESTAMP
      OLD_SAL NUMBER (8,2)
```

```
       NEW_SAL NUMBER (8,2)
       FID - NUMBER 4 – REQUIRED
*/
*********************************************************************************
CREATE OR REPLACE TRIGGER UPDATE_CUST_BALANCE_TRG
       AFTER INSERT ON LINE
       FOR EACH ROW
BEGIN
       UPDATE CUSTOMER SET BALANCE=BALANCE+(:NEW.L_PRICE*:NEW.L_UNITS)
              WHERE C_CODE=(SELECT C_CODE FROM INVOICE
                     WHERE INV_NUM=:NEW.INV_NUM);
END;
/


SELECT * FROM LINE WHERE INV_NUM=1006;


   INV_NUM      L_NUM P_COD    L_UNITS    L_PRICE
---------- ---------- ----- ---------- ----------
      1006          1 MC001          3       6.99
      1006          2 JB012          1     109.92
      1006          3 CH10X          1       9.95
      1006          4 HC100          1     256.99

SELECT * FROM INVOICE WHERE INV_NUM=1006;


   INV_NUM     C_CODE INV_DATE
---------- ---------- ---------
      1006      10014 17-JAN-20


SELECT * FROM CUSTOMER WHERE C_CODE=10014;


C_CODE LNAME      FNAME      C_AREA    C_PHONE    BALANCE
------ ---------- ---------- ------ ---------- ----------
 10014 Johnson    Bill          615    2455533          0

INSERT INTO LINE(INV_NUM,L_NUM,P_CODE,L_UNITS,L_PRICE)
       VALUES(1006,5,'PP101',10,5.87);
```

1 row created.

```
SELECT * FROM CUSTOMER WHERE C_CODE=10014;


C_CODE LNAME      FNAME      C_AREA    C_PHONE    BALANCE
------ ---------- ---------- ------ ---------- ----------
 10014 Johnson    Bill          615    2455533       58.7
```

```
CREATE TABLE SALARY
(
       OP_TYPE VARCHAR2(10) NOT NULL,
       OP_DATE DATE DEFAULT SYSDATE,
       OP_TIME CHAR(9) DEFAULT TO_CHAR(SYSDATE,'hh24:mi:ss'),
       OLD_SAL NUMBER(8,2),
       NEW_SAL NUMBER(8,2),
       EID NUMBER(4) NOT NULL
);
```

Table created.

****

Query 2: Write SQL code to compile and execute a trigger - SALARY_CHANGE_TRG, which will monitor DML operations on SALARY attribute of EMPP table and will add a record in SALARY_CHANGES table for each row affected by the DML statement. Test the trigger by performing following DML operations on EMPP.

Add : 7121, Melody Malvankar, SYSDATE, 80000, Asst. Professor

Add : 7122, Kalpak Gundappa, SYSDATE, 45000, Research Asst.

Modify : SALARY = SALARY+ 2500 for ENO >= 7121

Remove : ENO = 7122;

SQL> ALTER TRIGGER SALARY_CHANGE_TRG DISABLE; Trigger altered.

SELECT COUNT(*) FROM EMPP; [17 Tuples]

SELECT COUNT(*) FROM SALARY_CHANGES; [00 Tuples]

ALTER TRIGGER SALARY_CHANGE_TRG ENABLE;

```
INSERT INTO EMPP
       VALUES (7121, 'Melody Malvankar', SYSDATE, 'Asst. Professor, 80000); THE
INSERT ENTRY IS LOGGED IN SALARY_CHANGES TABLE
```

1 row created.

```
********************************************************************************

SELECT COUNT(*) FROM EMPP;


  COUNT(*)
----------
        19

SELECT COUNT(*) FROM SALARY_CHANGES;

  COUNT(*)
----------
         0

CREATE OR REPLACE TRIGGER SALARY_CHANGE_TRG
       BEFORE UPDATE OR DELETE OR INSERT OF SALARY ON EMPP
       FOR EACH ROW
BEGIN
       IF INSERTING THEN
              INSERT INTO SALARY_CHANGES(OP_TYPE,NEW_SAL,EID)
                     VALUES('INSERT',:NEW.SALARY,:NEW.EID);
              DBMS_OUTPUT.PUT_LINE('RECORD ENTRY ADDED TO SALARY_CHANGES');
       ELSIF UPDATING THEN
              INSERT INTO SALARY_CHANGES(OP_TYPE,OLD_SAL,NEW_SAL,EID)
                     VALUES('UPDATE',:OLD.SALARY,:NEW.SALARY,:NEW.EID);
              DBMS_OUTPUT.PUT_LINE('RECORD ENTRY ADDED TO SALARY_CHANGES');
       ELSE
              INSERT INTO SALARY_CHANGES(OP_TYPE,OLD_SAL,EID)
                     VALUES('DELETE',:OLD.SALARY,:OLD.EID);
              DBMS_OUTPUT.PUT_LINE('RECORD ENTRY ADDED TO SALARY_CHANGES');
       END IF;
END;
/

Trigger created.


INSERT INTO EMPP
       VALUES(7121,'Melody Malvankar',SYSDATE,'Asst. Professor',80000);
RECORD ENTRY ADDED TO SALARY_CHANGES
```

```
INSERT INTO EMPP
       VALUES(7122,'Kalpak Gundappa',SYSDATE,'Research Asst.',45000);

RECORD ENTRY ADDED TO SALARY_CHANGES

UPDATE EMPP SET SALARY=SALARY + 2500 WHERE EID>=7121;

RECORD ENTRY ADDED TO SALARY_CHANGES
RECORD ENTRY ADDED TO SALARY_CHANGES

DELETE FROM EMPP WHERE EID=7122;

RECORD ENTRY ADDED TO SALARY_CHANGES

SELECT * FROM SALARY_CHANGES;


OP_TYPE    OP_DATE   OP_TIME      OLD_SAL    NEW_SAL        EID
---------- --------- --------- ---------- ---------- ----------
INSERT     10-OCT-21 18:25:32                 80000       7121
INSERT     10-OCT-21 18:25:53                 45000       7122
UPDATE     10-OCT-21 18:26:47       80000      82500       7121
UPDATE     10-OCT-21 18:26:47       45000      47500       7122
DELETE     10-OCT-21 18:27:35       47500                  7122

CREATE TABLE EMP_SALARY(
       ENO NUMBER(4),
       TOT_SAL NUMBER(8,2)
);

Table created.

ALTER TABLE EMP_SALARY
       ADD CONSTRAINT EMP_SAL_PK_ENO PRIMARY KEY(ENO);

Table altered.

ALTER TABLE EMP_SALARY
       ADD STATUS VARCHAR2(7) DEFAULT 'ON_ROLL';

Table altered.
```

```
INSERT INTO EMP_SALARY(ENO,TOT_SAL)
	SELECT EID,(SALARY*1.25-1200)*0.90 FROM EMPP;

19 rows created.
```

****

Query 3: Write SQL code to compile and execute a trigger - UPDATE_TOT_SAL_TRG, which will monitor DML operations on SALARY attribute of EMPP table and will keep EMP_SALARY table updated with the current total salary of the employee. When a new employee record is added in EMPP, a record in EMP_SALARY is also inserted with appropriate values. When employee salary is changed, the EMP SALARY records for affected employees are updated. When an employee is removed from EMPP, the corresponding record in EMP SALARY is not removed, but the STATUS filed is set to 'RETIRED'.

The TOT SAL is computed as - (SALARY+PERKS-PF_Deductions)-IT_Deductions. PERKS are 25% of SALARY and PF Deductions are fixed at 1200. The IT Deductions are 10% of the cumulative of (Salary, Perks) minus PF_Deductions.

Before testing UPDATE_TOT_SAL_TRG, disable the trigger - SALARY_CHANGE_TRG using the command... ALTER TRIGGER SALARY_CHANGE_TRG DISABLE; (which may be enabled when required)

Test UPDATE TOT_SAL_TRG trigger by performing following DML operations on EMPP -

Add : 7121, Melody Malvankar, SYSDATE, 80000, Asst. Professor

Add : 7122, Kalpak Gundappa, SYSDATE, 45000, Research Asst.

Modify : SALARY = SALARY + 2500 for ENO >= 7121

Remove : ENO=7122;

****

```
CREATE OR REPLACE TRIGGER UPDATE_TOT_SAL_TRG
	BEFORE UPDATE OR INSERT OR DELETE OF SALARY ON EMPP
	FOR EACH ROW
BEGIN
	IF INSERTING THEN

		INSERT INTO EMP_SALARY(ENO,TOT_SAL)
			VALUES(:NEW.EID,(:NEW.SALARY+(:NEW.SALARY*0.25)-1200)-
(((:NEW.SALARY*0.25)+:NEW.SALARY-1200)*0.1));
```

```
	ELSIF UPDATING THEN

		UPDATE EMP_SALARY SET TOT_SAL=(:NEW.SALARY+(:NEW.SALARY*0.25)-1200)-
(((:NEW.SALARY*0.25)+:NEW.SALARY-1200)*0.1)
			WHERE ENO=:OLD.EID;

	ELSE

		UPDATE EMP_SALARY SET STATUS='RETIRED'
			WHERE ENO=:OLD.EID;

	END IF;
END;
/

Trigger created.

INSERT INTO EMPP
	VALUES(7121,'Melody Malvankar',SYSDATE,'Asst. Professor',80000);
1 row created.


INSERT INTO EMPP
	VALUES(7122,'Kalpak Gundappa',SYSDATE,'Research Asst.',45000);

1 row created.



UPDATE EMPP SET SALARY=SALARY + 2500 WHERE EID>=7121;

2 rows updated.

DELETE FROM EMPP WHERE EID=7122;

1 row deleted.

SELECT * FROM EMP_SALARY;

	ENO    TOT_SAL STATUS
---------- ---------- -------
      7102   163732.5 ON_ROLL
```

```
      7101     167670 ON_ROLL
      7103     165420 ON_ROLL
      7104     154620 ON_ROLL
      7107     142245 ON_ROLL
      7105     142245 ON_ROLL
      7106     142245 ON_ROLL
      7108   133582.5 ON_ROLL
      7109     101295 ON_ROLL
      7110      96120 ON_ROLL
      7111      53145 ON_ROLL

       ENO    TOT_SAL STATUS
---------- ---------- -------
      7112      49095 ON_ROLL
      7113      38970 ON_ROLL
      7114   35876.25 ON_ROLL
      7115      32670 ON_ROLL
      7116      32670 ON_ROLL
      7117      35145 ON_ROLL
      7118      27045 ON_ROLL
      7119     181170 ON_ROLL
      7121    91732.5 ON_ROLL
      7122    52357.5 RETIRED

*********************************************************************************
Query 4: Write SQL code to compile and execute a trigger - LINE_INS_UPD_QTY_TRG
that will automatically update the quantity on hand (QTY) for each product sold
after a new LINE row is added.
*********************************************************************************


CREATE OR REPLACE TRIGGER LINE_INS_UPD_QTY_TRG
       AFTER INSERT ON LINE
       FOR EACH ROW
BEGIN
       UPDATE PRODUCT SET QTY=QTY-:NEW.L_UNITS
              WHERE UPPER(P_CODE)=UPPER(:NEW.P_CODE);
END;
/

Trigger created.

SELECT P_CODE,DESCRIPT,QTY FROM PRODUCT
```

```
       WHERE UPPER(P_CODE)='RF100';

P_COD DESCRIPT                              QTY
----- ------------------------------ ----------
RF100 Rat Tail File                          43

SELECT INV_NUM,L_NUM,P_CODE,L_UNITS
       FROM LINE WHERE INV_NUM=1005;

   INV_NUM      L_NUM P_COD    L_UNITS
---------- ---------- ----- ----------
      1005          1 PP101         12

INSERT INTO LINE VALUES(1005,2,'RF100',20,4.99);

1 row created.

SELECT INV_NUM,L_NUM,P_CODE,L_UNITS
       FROM LINE WHERE INV_NUM=1005;

   INV_NUM      L_NUM P_COD    L_UNITS
---------- ---------- ----- ----------
      1005          1 PP101         12
      1005          2 RF100         20

SELECT P_CODE,DESCRIPT,QTY FROM PRODUCT
       WHERE UPPER(P_CODE)='RF100';

P_COD DESCRIPT                              QTY
----- ------------------------------ ----------
RF100 Rat Tail File                          23

CREATE TABLE PRODUCT_T(
       P_CODE CHAR(5),
       DESCRIPT VARCHAR(30),
       QTY NUMBER(4),
       P_MIN NUMBER(3),
       P_PRICE NUMBER(6,2),
       V_CODE NUMBER(5)
);

Table created.
```

```
ALTER TABLE PRODUCT_T
       ADD REORDER NUMBER(4) DEFAULT 0;

Table altered.

INSERT INTO PRODUCT_T(P_CODE,DESCRIPT,QTY,P_MIN,P_PRICE,V_CODE)
       SELECT P_CODE,DESCRIPT,QTY,P_MIN,P_PRICE,V_CODE
       FROM PRODUCT;

22 rows created.

********************************************************************************
Query 5: Write a SQL code to compile and execute the stored procedure - ADD_ITEM,
that will insert an item in ITEMS table with given particulars item code, item
description, invoice date, quantity of purchase, minimum quantity, item price and
supplier code.
********************************************************************************
CREATE OR REPLACE TRIGGER CHECK_REORDER_STATUS_TRG_RL

       AFTER UPDATE ON PRODUCT_T

       FOR EACH ROW

BEGIN

       IF :NEW.QTY<=:NEW.P_MIN THEN

              UPDATE PRODUCT_T SET REORDER=1 WHERE P_CODE=:OLD.P_CODE;

       ELSE

              UPDATE PRODUCT_T SET REORDER=0 WHERE P_CODE=:OLD.P_CODE;

       END IF;

END;

/


Trigger created.
```