Laboratory 5
SQL Triggers

Goal

The objective of this practice is to write some triggers in SQL to run on an Oracle database. To connect to a Database you can follow the guides related to the first lab practice. This lab can be done in Oracle SQL Developer or in Oracle APEX online without any difference. The triggers must be created using the functionality that allows the execution of SQL code.

Useful SQL statements

- Delete a trigger:

```
DROP TRIGGER triggerName;
DROP TRIGGER "triggerName";
```

- Update of an existing trigger (instead of delete and recreate it):

```
CREATE OR REPLACE TRIGGER triggerName …
```

- Display defined triggers:

```
SELECT trigger_name, triggering_event, table_name, status,
Description, action_type, trigger_body
FROM user_triggers
```

- Disable a trigger:

```
ALTER TRIGGER triggerName DISABLE;
```

- Display trigger errors:

```
SELECT * FROM USER_ERRORS;
```

Suggestions

Before doing each exercise, load the related tables by executing the scripts *script_db_es1.sql, script_db_es2.sql*.

To create a trigger, pay attention to the syntax and to the following issues:

- assign a proper name to the variables avoiding keywords like MIN, MAX, ...
- declare different variables on different lines and not on the same line delimited by a comma

```
MyVarOne NUMBER;
MyVarTwo NUMBER;
MyVarThree VARCHAR2(16)
```

- terminate the statements with ; character and assign new values to the variables with :=, e.g.

```
UPDATE tableName
SET varname=newvalue
WHERE column=:NEW.attribute;

IF A<3 OR A=3 THEN
        MyVar := 'Three';
ELSE
IF A>3 AND A<5 THEN
        MyVar := 'Four';
ELSE
        MyVar := 'Other';
END IF;
END IF;
```

Before starting this practice we suggest you to delete any existing trigger, which could affect the outcome of the exercises.

Remark. Pay attention because copying and pasting SQL code lines from the practice text in PDF format into the Browser Web may generate errors due to invalid character encoding or conversion, such as *ora-00911: invalid character*.

1 Exercise 1

The following relations are given (primary keys are underlined; optional attributes are denoted with *):

- IMP(EMPNO, DEPTNO, ENAME, JOB, SAL)
- DIP(DEPTNO, DNAME, LOC, MINSAL, MAXSAL)

Write the trigger which manages the update of the DNAME attribute on the DIP table. When the DNAME attribute changes from 'ACCCOUNTING' to 'SALES', the wage (SAL attribute) for all employees, who work in the corresponding DEPTNO, is increased by 100.

Procedure:

- Create the database using script *create_db_es1.sql*.
- Create the trigger, eventually by means of a script.
- Verify the content of IMP and DIP tables.
- Modify the department name 'ACCOUNTING':

```
UPDATE DIP SET DNAME = 'SALES' WHERE DNAME = 'ACCOUNTING';
```

- Verify the content of the IMP and DIP tables.

2 Exercise 2

The following relations are given (primary keys are underlined; optional attributes are denoted with *):

- IMP(EMPNO, ENAME, JOB, SAL)
- SUMMARY(JOB, NUM)

In the SUMMARY table, the NUM attribute species the number of employees in the IMP table who perform the same job. Write the triggers to guarantee the consistency between IMP and SUMMARY tables when:

- A new record is inserted in the IMP table.
- The value of JOB in the IMP table is updated.

Create the database using script *create_db_es2.sql*.

Solutions

1 Exercise 1

Table before trigger action

IMP table

EMPNO	DEPTNO	ENAME	JOB	SAL
7000	10	SMITH	CLERK	850
7010	10	SCOTT	ANALYST	1600
7020	10	BLAKE	SALESMAN	1400
7030	10	SMITH	MANAGER	2500
7040	20	SMITH	CLERK	800
7050	20	SCOTT	ANALYST	1600
7060	30	ADAMS	CLERK	900
7070	30	JAMES	CLERK	1000
7080	40	ALLEN	CLERK	850

DIP table

DEPTNO	DNAME	LOC	MINSAL	MAXSAL
10	ACCOUNTING	NEW YORK	100	2500
20	RESEARCH	DALLAS	150	3000
30	SALES	CHICAGO	120	2500
40	OPERATIONS	BOSTON	200	2100

Code

Trigger

```
CREATE OR REPLACE TRIGGER UP_SAL
AFTER UPDATE OF DNAME ON DIP
FOR EACH ROW
WHEN (OLD.DNAME = 'ACCOUNTING' AND NEW.DNAME='SALES')
BEGIN
-- updating salary of the employees of the changed department
UPDATE IMP
SET SAL = SAL+100
WHERE DEPTNO = :OLD.DEPTNO;
END;
```

Update statement

```
UPDATE DIP SET DNAME = 'SALES' WHERE DNAME = 'ACCOUNTING';
```

Table after trigger action

IMP table

EMPNO	DEPTNO	ENAME	JOB	SAL
7000	10	SMITH	CLERK	950
7010	10	SCOTT	ANALYST	1700
7020	10	BLAKE	SALESMAN	1500
7030	10	SMITH	MANAGER	2600
7040	20	SMITH	CLERK	800
7050	20	SCOTT	ANALYST	1600
7060	30	ADAMS	CLERK	900
7070	30	JAMES	CLERK	1000
7080	40	ALLEN	CLERK	850

DIP table

DEPTNO	DNAME	LOC	MINSAL	MAXSAL
10	SALES	NEW YORK	100	2500
20	RESEARCH	DALLAS	150	3000
30	SALES	CHICAGO	120	2500
40	OPERATIONS	BOSTON	200	2100

2 Exercise 2

Table before trigger action

IMP table

EMPNO	ENAME	JOB	SAL
1	VERDI	SECRETARY	800
2	ROSSI	BANKER	900
3	BIANCHI	BANKER	1100

SUMMARY table

JOB	NUM
SECRETARY	1
BANKER	2

Code

Trigger to manage the insertion of a new record into the IMP table

```
CREATE OR REPLACE TRIGGER INS_IMP
AFTER INSERT ON IMP
FOR EACH ROW
DECLARE
N NUMBER;
M NUMBER;
BEGIN
-- checking if there are other employees with the same job
SELECT COUNT(*) INTO N
FROM SUMMARY
WHERE JOB = :NEW.JOB;
IF (N=0) THEN
-- this is the first employee with this job
INSERT INTO SUMMARY (JOB, NUM)
VALUES (:NEW.JOB, 1);
ELSE
-- there is at least another employee with the same job
SELECT NUM INTO M
FROM SUMMARY
WHERE JOB = :NEW.JOB;
UPDATE SUMMARY
SET NUM = M+1
WHERE JOB = :NEW.JOB;
END IF;
END;
```

Insert statement

```
INSERT INTO IMP(EMPNO, ENAME, JOB, SAL) VALUES (4, 'NERI', 'DELIVERER', 750);
```

Table after trigger action

IMP table

EMPNO	ENAME	JOB	SAL
1	VERDI	SECRETARY	800
2	ROSSI	BANKER	900
3	BIANCHI	BANKER	1100
4	NERI	DELIVERER	750

SUMMARY table

JOB	NUM
SECRETARY	1
BANKER	2
DELIVERER	1

Code

Trigger to manage the update of the JOB field on the IMP table

```
CREATE OR REPLACE TRIGGER UPD_IMP
AFTER UPDATE OF JOB ON IMP
FOR EACH ROW
DECLARE
N NUMBER;
M NUMBER;
BEGIN
-- count how many employees have the new job
SELECT COUNT(*) INTO N
FROM SUMMARY
WHERE JOB = :NEW.JOB;
-- increment the number of employees for the new job
IF (N=0) THEN
-- the inserted employee is the first employee for the new job
INSERT INTO SUMMARY (JOB, NUM)
VALUES (:NEW.JOB, 1);
ELSE
-- there is at least one other employee for the new job
UPDATE SUMMARY
SET NUM = NUM+1
WHERE JOB = :NEW.JOB;
END IF;
SELECT NUM INTO M
FROM SUMMARY
WHERE JOB = :OLD.JOB;
IF (M =1) THEN
-- there was only an employee for the old job. Delete the record from
-- SUMMARY table
DELETE FROM SUMMARY
WHERE JOB = :OLD.JOB;
ELSE
-- decrement NUM in the corresponding record of SUMMARY table
UPDATE SUMMARY
SET NUM = NUM-1
WHERE JOB = :OLD.JOB;
END IF;
END;
```

Update statement

```
UPDATE IMP SET JOB = 'DELIVERER' WHERE EMPNO = 2;
```

Table after trigger action

IMP table

EMPNO	ENAME	JOB	SAL
1	VERDI	SECRETARY	800
2	ROSSI	DELIVERER	900
3	BIANCHI	BANKER	1100
4	NERI	DELIVERER	750

SUMMARY table

JOB	NUM
SECRETARY	1
BANKER	1
DELIVERER	2