USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science

CS311 Introduction to Database Systems

Lab#12

Objective:

- DATABASE TRIGGERS (II)

Name of Student: Muhammad Waleed

Roll No: 20B-115-SE Sec. B

Date of Experiment:

Marks Obtained/Remarks: _____

Signature: _____

Creating Row Triggers

Row Triggers. A row trigger is fired each time the table is affected by the triggering statement. For example, if an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement. If a triggering statement affects no rows, a row trigger is not run.

Syntax for creating Row Triggers

CREATE [OR REPLACE] TRIGGER *trigger_name*
Timing event1 [OR *event2* OR *event3*] ON
table_name
FOR EACH ROW
[WHEN *condition*]
PL/SQL block;

This syntax is identical to the syntax for creating statement triggers except following: - FOR EACH ROW: Designates the trigger to be a row trigger
WHEN: Specifies the trigger restriction (This conditional predicate is evaluated for each row to determine whether or not the trigger body is executed.)

Restriction on Row Triggers:

This clause is valid only for DML event triggers, not for DDL or database event triggers.

EXAMPLE 1

After Row Trigger:
A row trigger can be created to keep a running count of data manipulation operations by different users on database tables. If a trigger routine does not have to take place before the triggering operation, create an AFTER row trigger rather than a BEFORE row trigger.

```
CREATE OR REPLACE TRIGGER audit_emp
AFTER DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
IF DELETING THEN
        UPDATE      audit_table     SET    del = del + 1
        WHERE       user_name = user AND table_name = 'EMP'
        AND         column_name IS NULL;
ELSIF INSERTING THEN
        UPDATE      audit_table     SET    ins = ins + 1
        WHERE       user_name = user AND table_name = 'EMP'
```

```
        AND         column_name IS NULL;
ELSIF UPDATING('SAL') THEN
        UPDATE      audit_table     SET    upd = upd + 1
        WHERE       user_name = user AND table_name = 'EMP'
        AND         column_name = 'SAL';
ELSE
        UPDATE      audit_table     SET    upd = upd + 1
        WHERE       user_name = user AND table_name = 'EMP'
        AND         column_name IS NULL;
END IF;
END;
```

```
1  CREATE OR REPLACE TRIGGER audit_emp19b045
2  AFTER DELETE OR INSERT OR UPDATE ON emp19b045
3  FOR EACH ROW
4  BEGIN
5  IF DELETING THEN
6    UPDATE  audit_table19b045   SET   del = del + 1
7    WHERE   user_name = user AND table_name = 'EMP19B045'
8    AND     column_name IS NULL;
9  ELSIF INSERTING THEN
10   UPDATE  audit_table19b045   SET   ins = ins + 1
11   WHERE   user_name = user AND table_name = 'EMP19B045'
12   AND     column_name IS NULL;
13 ELSIF UPDATING('SAL') THEN
14   UPDATE  audit_table19b045   SET   upd = upd + 1
15   WHERE   user_name = user AND table_name = 'EMP19B045'
16   AND     column_name = 'SAL';
17 ELSE
18   UPDATE  audit_table19b045   SET   upd = upd + 1
19   WHERE   user_name = user AND table_name = 'EMP19B045'
20   AND     column_name IS NULL;
21 END IF; END;
```

Script Output | Task completed in 0.042 seconds

```
TRIGGER AUDIT_EMP19B045 compiled
```

AUDIT TABLE BEFORE INSERTION UPDATION AND DELETION

```
33  select * from audit_table19b045;
```

Query Result | SQL | All Rows Fetched: 3 in 0.003 seconds

	USER_NAME	TABLE_NAME	COLUMN_NAME	INS	UPD	DEL	MAX_INS	MAX_UPD	MAX_DEL
1	SCOTT	EMP19B045	(null)	1	1	1	5	5	5
2	SCOTT	EMP19B045	SAL	(null)	6	(null)	(null)	5	(null)
3	SCOTT	EMP19B045	(null)	0	0	0	5	0	0

```
24  Insert into EMP19B045 (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
25  values (7370,'AHTISHAM','MANAGER',7839,to_date('31-DEC-81','DD-MON-RR'),2800,null,20);
```

Script Output | Query Result | Task completed in 0.005 seconds

```
1 rows inserted.
```

```
27  update emp19b045 set sal = 2900 where empno = 7370;
```

Query Result | Script Output | Task completed in 0.006 seconds

```
1 rows updated.
```

```
29  delete from emp19b045 where empno = 7370;
```

Query Result | Script Output | Task completed in 0.006 seconds

```
1 rows deleted.
```

AUDIT TABLE AFTER INSERTION UPDATION AND DELETION

```
34  select * from audit_table19b045;
```

Query Result | SQL | All Rows Fetched: 3 in 0.002 seconds

	USER_NAME	TABLE_NAME	COLUMN_NAME	INS	UPD	DEL	MAX_INS	MAX_UPD	MAX_DEL
1	SCOTT	EMP19B045	(null)	2	1	2	5	5	5
2	SCOTT	EMP19B045	SAL	(null)	7	(null)	(null)	5	(null)
3	SCOTT	EMP19B045	(null)	1	0	1	5	0	0

EXAMPLE 2

Using Old and New Qualifiers

We can create a trigger on the EMP table to add rows to a user table, AUDIT_EMP_VALUES, logging a user's activity against the EMP table. The trigger records the values of several columns both before and after the data changes by using the OLD and NEW qualifiers with the respective column name.

```
CREATE OR REPLACE TRIGGER audit_emp_values
AFTER DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
    INSERT INTO audit_emp_values (user_name, timestamp, id, old_last_name, old_title,
                New_title, old_salary, new_salary)
    VALUES (USER, SYSDATE, :old.empno, :old.ename, :new.ename, :old.job, :new.job,
            :old.sal, :new.sal);
END;
```

```
52  CREATE OR REPLACE TRIGGER audit_emp_val
53  AFTER DELETE OR INSERT OR UPDATE ON empl9b045
54  FOR EACH ROW
55  BEGIN
56          INSERT INTO audit_e045_val (user_name, timestamp, id, old_name, new_name, old_job,
57                                      New_job, old_sal, new_sal)
58          VALUES (USER, SYSDATE, :old.empno, :old.ename, :new.ename, :old.job, :new.job,
59                                 :old.sal, :new.sal);
60  END;
```

Query Result | Script Output | Task completed in 0.045 seconds

```
TRIGGER AUDIT_EMP_VAL compiled
```

```
64  Insert into EMP19B045 (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
65  values (7364,'MURSALEEN','MANAGER',7839,to_date('23-DEC-80','DD-MON-RR'),2800,null,20);
66
67  update empl9b045 set sal = 2900 , job = 'CLERK' where empno = 7364;
```

Query Result | Script Output | Task completed in 0.005 seconds

```
1 rows inserted.
1 rows updated.
```

```
69  delete from empl9b045 where empno = 7364;
```

Query Result | Script Output | Task completed in 0.005 seconds

```
1 rows deleted.
```

```
71  select * from audit_e045_val;
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.007 seconds

	USER_NAME	TIMESTAMP	ID	OLD_NAME	NEW_NAME	OLD_JOB	NEW_JOB	OLD_SAL	NEW_SAL
1	SCOTT	28-JAN-22	(null)	(null)	MURSALEEN	(null)	MANAGER	(null)	2800
2	SCOTT	28-JAN-22	7364	MURSALEEN	MURSALEEN	MANAGER	CLERK	2800	2900
3	SCOTT	28-JAN-22	7364	MURSALEEN	(null)	CLERK	(null)	2900	(null)

EXAMPLE 3

Before Row Trigger:

To restrict the trigger action to those rows that satisfy a certain condition, provide a WHEN clause. Create a trigger on the EMP table to calculate an employee's commission when a row is added to the EMP table or an employee's salary is modified.
The NEW qualifier does not need to be prefixed with a colon in the WHEN clause.

Lab No. 12

```
CREATE OR REPLACE TRIGGER derive_commission_pct
BEFORE INSERT OR UPDATE OF SAL ON emp
FOR EACH ROW
WHEN (new.job = 'SALESMAN')
BEGIN
    IF INSERTING THEN    :new.comm. := 0;
    ELSE       /* UPDATE of salary */
        IF :old.comm. IS NULL THEN
                :new.comm. := 0;
        ELSE
                :new.comm. := :old.comm. + (:new.sal / :old.sal);
        END IF;
    END IF;
END;
```

```
77  CREATE OR REPLACE TRIGGER derive_comm_pct
78  BEFORE INSERT OR UPDATE OF SAL ON empl9b045
79  FOR EACH ROW
80  WHEN (new.job = 'SALESMAN')
81  BEGIN
82        IF INSERTING THEN   :new.comm := 0;
83        ELSE    /* UPDATE of salary */
84    IF :old.comm IS NULL THEN
85      :new.comm := 0;
86    ELSE
87      :new.comm := :old.comm + (:new.sal / :old.sal);
88    END IF;
89       END IF; END;
```

Script Output | Task completed in 0.039 seconds

```
TRIGGER DERIVE_COMM_PCT compiled
```

```
92  Insert into EMP19B045 (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
93  values (7399,'ASAD','SALESMAN',7839,to_date('22-DEC-81','DD-MON-RR'),1600,200,20);
```

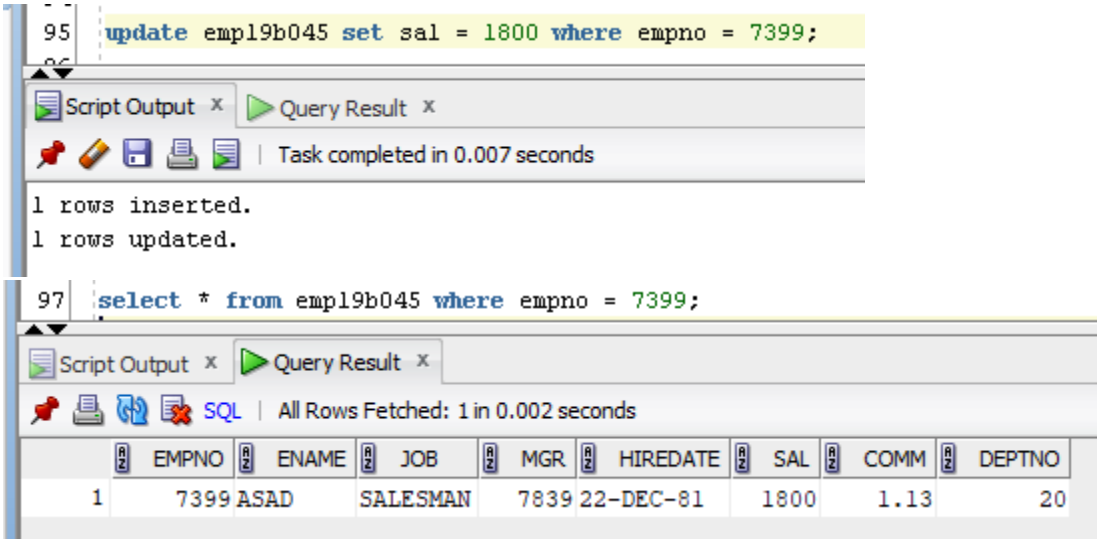
Script Output | Task completed in 0.006 seconds

```
1 rows inserted.
```

```
95  select * from empl9b045 where empno = 7399;
```

Script Output | Query... | SQL | All Rows Fetched: 1 in 0.004 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7399	ASAD	SALESMAN	7839	22-DEC-81	1600	0	20



DDL Triggers

Oracle allows you to define triggers that will fire when DDL statements are executed. Simply put, DDL is any SQL statement used to create or modify a database object such as a table or an index. DL triggers include the following types of triggers:

- BEFORE CREATE and AFTER CREATE triggers fire when a schema object is created in the database or schema.
- BEFORE ALTER and AFTER ALTER triggers fire when a schema object is altered in the database or schema.
- BEFORE DROP and AFTER DROP triggers fire when a schema object is dropped from the database or schema.

EXAMPLE 4

Creating DDL Trigger:

This trigger will insert the respective information in the table *"schema_audit"* such as the date when the DDL is executed, username who executed the DDL, type of database object created, name of the object given by the user at the time of its creation and the type of DDL into the table which we created earlier.

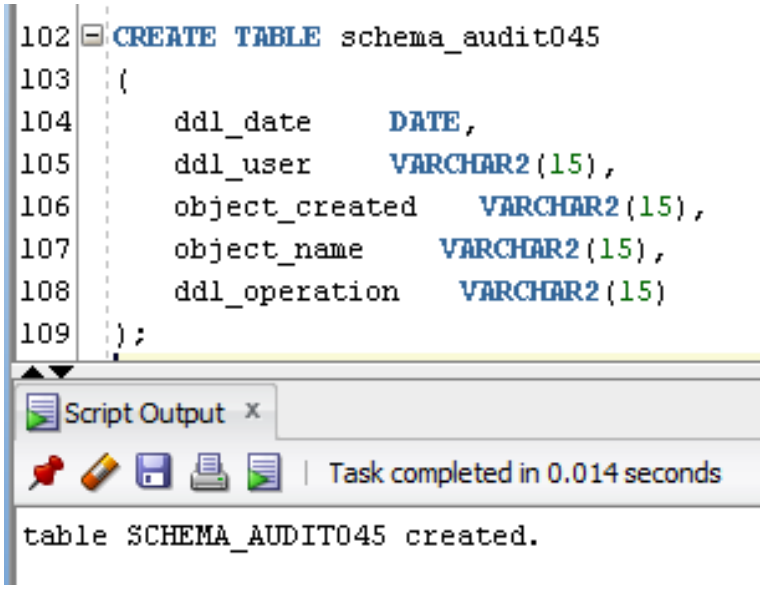
Creating an Audit Table:

```
CREATE TABLE schema_audit
(
    ddl_date            DATE,
```

```
    ddl_user            VARCHAR2(15),
    object_created      VARCHAR2(15),
    object_name         VARCHAR2(15),
    ddl_operation       VARCHAR2(15)
);
```

Creating Trigger:

```
CREATE OR REPLACE TRIGGER hr_audit_tr
AFTER DDL ON SCHEMA
BEGIN
    INSERT INTO schema_audit VALUES
    (
      sysdate,
      sys_context('USERENV','CURRENT_USER'),
      ora_dict_obj_type,
      ora_dict_obj_name,
      ora_sysevent
    );
END;
```



```
112 CREATE OR REPLACE TRIGGER hr_audit045_tr
113  AFTER DDL ON SCHEMA
114 BEGIN
115     INSERT INTO schema_audit045 VALUES
116     (
117       sysdate,
118       sys_context('USERENV','CURRENT_USER'),
119       ora_dict_obj_type,
120       ora_dict_obj_name,
121       ora_sysevent
122     );
123 END;
124
```

Script Output | Task completed in 0.028 seconds

```
TRIGGER HR_AUDIT045_TR compiled
```

```
125 DESCRIBE EMP19b045;
```

Script Output | Task completed in 0.008

```
DESCRIBE EMP19b045
Name      Null     Type
--------- -------- ------------
EMPNO     NOT NULL NUMBER(4)
ENAME     NOT NULL VARCHAR2(10)
JOB                VARCHAR2(9)
MGR                NUMBER(4)
HIREDATE           DATE
SAL                NUMBER(7,2)
COMM               NUMBER(7,2)
DEPTNO    NOT NULL NUMBER(7,2)
```

```
127 Alter table emp19b045
128 modify EMPNO NUMBER(5);
```

Script Output | Task completed in 0.032 seconds

```
table EMP19B045 altered.
```

```
130  DESCRIBE EMP19b045;
```

Script Output | Task completed in 0.007 seconds

```
DESCRIBE EMP19b045
Name      Null     Type
--------- -------- ------------
EMPNO     NOT NULL NUMBER(5)
ENAME     NOT NULL VARCHAR2(10)
JOB                VARCHAR2(9)
MGR                NUMBER(4)
HIREDATE           DATE
SAL                NUMBER(7,2)
COMM               NUMBER(7,2)
DEPTNO    NOT NULL NUMBER(7,2)
```

```
132  select * from schema_audit045;
```

Query Result | SQL | All Rows Fetched: 111 in 0.018 seconds

	DDL_DATE	DDL_USER	OBJECT_CREATED	OBJECT_NAME	DDL_OPERATION
28	28-JAN-22	SCOTT	TABLE	EMP19B045	ALTER

Trigger Enabling and Disabling

By default, the CREATE TRIGGER statement creates a trigger in the enabled state. To create a trigger in the disabled state, specify DISABLE. Creating a trigger in the disabled state lets you ensure that it compiles without errors before you enable it.

Some reasons to temporarily disable a trigger are:

- The trigger refers to an unavailable object.
- You must do a large data load, and you want it to proceed quickly without firing triggers.
- You are reloading data.

To enable or disable a single trigger, use this statement:

ALTER TRIGGER [schema.]trigger_name { ENABLE | DISABLE };

To enable or disable all triggers created on a specific table, use this statement:

ALTER TABLE table_name { ENABLE | DISABLE } ALL TRIGGERS;

Exercise:

1. Create a trigger that fill hiredate column with the current date, If the newly inserted record in employee has null hireDate field.

```
5  Create or replace Trigger HireDate19B045
6  Before Insert On Emp19b045
7  For Each Row
8  Declare
9   tempHiredate date;
10 Begin
11  Select sysdate into tempHiredate from dual;
12  IF (:new.hireDate is null) Then
13  :new.hireDate := tempHiredate;
14  End IF;
15 End;
```

Script Output | Task completed in 0.037 seconds

```
TRIGGER HIREDATE19B045 compiled
```

VERIFIY

```
16 --Exercise 1 VERIFICATION
17 insert into emp19B045(empno,ename,job,mgr,HIREDATE,sal,comm,deptno)
18 values(8495,'KHALID','CLERK', 7839,NULL,500,null,20);
```

Script Output | Task completed in 0.058 seconds

```
1 rows inserted.
```

```
20 select * from emp19b045 where empno = 8495;
```

Script Output | Query... | SQL | All Rows Fetched: 1 in 0.004 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	8495	KHALID	CLERK	7839	28-JAN-22	500	(null)	20

2. Create a trigger that will fill the commission attribute in Employee table always 3% of the salary attribute.

```
22  --EXERCISE 2
23  Create Trigger Commision19B045
24  Before Insert On Emp19B045
25  For each row
26  Begin
27    IF (:new.comm is null) Then
28   :new.comm := :new.sal * 0.03;
29    End IF;
30  End;
```

Script Output

Task completed in 0.035 seconds

```
TRIGGER COMMISION19B045 compiled
```

VERIFY

```
32  --Exercise 2 VERIFICATION
33  insert into emp19B045(empno,ename,job,mgr,hiredate,sal,COMM,deptno)
34  values(8555,'NASIR','CLERK', 7839,null,500,NULL,20);
```

Script Output

Task completed in 0.006 seconds

```
1 rows inserted.
```

```
32  --Exercise 2 VERIFICATION
33  insert into emp19B045(empno,ename,job,mgr,hiredate,sal,COMM,deptno)
34  values(8555,'NASIR','CLERK', 7839,null,500,NULL,20);
35
36  select * from emp19b045 where empno = 8555;
```

Query Result

SQL | All Rows Fetched: 1 in 0.004 seconds

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	8555	NASIR	CLERK	7839	28-JAN-22	500	15	20

3. Study INSTEAD OF Triggers and explain it with examples.

An INSTEAD OF trigger is a trigger that allows you to skip an INSERT, DELETE, or UPDATE statement to a table or a view and execute other statements defined in the trigger instead. The actual insert, delete, or update operation does not occur at all.

In other words, an INSTEAD OF trigger skips a DML statement and execute other statements.

A typical example of using an INSTEAD OF trigger is to override an insert, update, or delete operation on a view. Suppose, an application needs to insert new brands into the production. brands table. However, the new brands should be stored in another table called production.