**USMAN INSTITUTE OF TECHNOLOGY**

**Department of Computer Science**

**CS311 Introduction to Database Systems**

Lab#11

**Objective:**

**-** **DATABASE TRIGGERS (II)**

**Name of Student: \_\_Zubair Shahid\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No: 17B-023-SE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Sec. \_SE-A\_\_\_\_\_\_\_\_\_\_**

**Date of Experiment: \_02-01-2020\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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.

**After Row Trigger: Example**

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;** |

**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;

**Before Row Trigger: Example**

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.

|  |
| --- |
| 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; |

**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.

**Creating DDL Trigger: Example**

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;

**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 };**

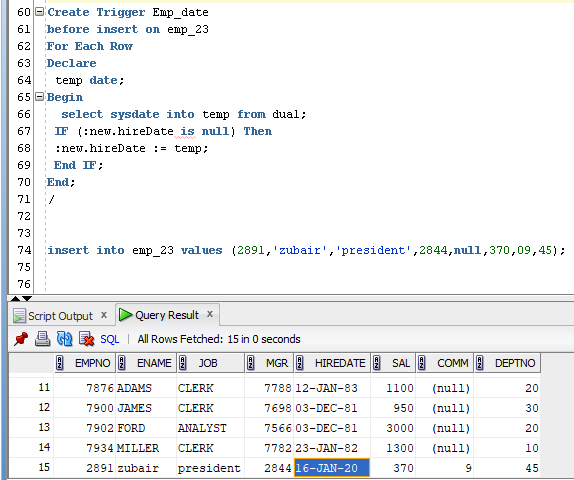
To enable or disable all triggers created on a specific table, use this statement:

**ALTER TABLE table\_name { ENABLE | DISABLE } ALL TRIGGERS;**

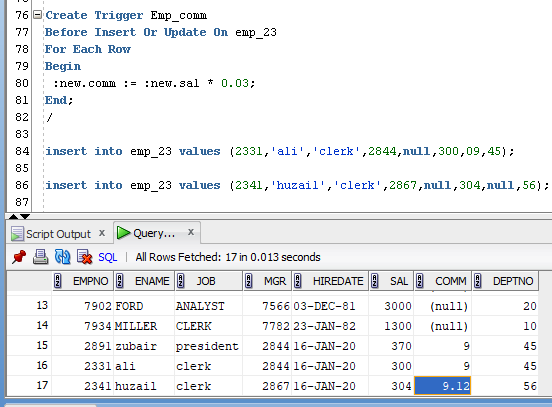
**Exercise:**

1. Create a trigger that fill hiredate column with the current date, If the newly inserted record in employee has null hireDate field.
2. Create a trigger that will fill the commission attribute in Employee table always 3% of the salary attribute.
3. Study INSTEAD OF Triggers and explain it with examples.

Task#1



Task#2



Task#3

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.

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.brand\_approvals for approval before inserting into the production.brands table.

To accomplish this, you create a view called production.vw\_brands for the application to insert new brands. If brands are inserted into the view, an INSTEAD OF trigger will be fired to insert brands into the production.brand\_approvals table.