Home assignment #7

Due date 6/1/2022, 8:00 PM

Note, all the triggers we create in hw are for exercise. It is not good practice to bind a table with many none-sense triggers. So, once you have done with all the codes for submission, it is better to drop the trigger, such as: DROP TRIGGER TRIGGER\_name;

Q1. (25 points) statement level Trigger.

There will be typically four steps for completing a trigger related question. The first step, before working on the trigger, we will create a log table, here for Q1, it is called Empl\_Log as below:

CREATE TABLE Empl\_log(

Updated\_Date DATE default SYSDATE,

Updated\_By Varchar2 (15) default USER,

Action Varchar2 (30)

);

Normally, the default value of Updated\_Date will be SYSDATE, Updated\_by will be the current login name “USER”. You are free to define the string of action,

The second step requires you to create a statement level trigger named Empl\_delete on table Employees (either before or after, up to you). Whenever a command of delete on the table Employees is executed, the trigger will be fired, and add one row into the table Empl\_log.

After having successfully compiled the trigger, your program will run a DELETE commands to test the trigger as the third step.

DELETE employees where employee\_id = 105;

In the forth step, your submission should include the output of Empl\_log table, that displays the new record in the log table. It will be a good practice, remember to rollback after you get the display of the results:

-- rollback;

\*\* you may notice that the “rollback” will cancel the deletion command on table employees, also delete the correspondent record in the Empl\_log table.

Q2. (25 points) Row level Trigger.

This question Q2 is similar to question #1 in this hw7, but it will be “for each row”.

It requires to create a row level trigger named Empl\_Del\_Row on table Employees.

You will create a table called Empl\_Del\_log to store the info from this trigger. Whenever there is a command of delete on the table employees, the trigger will be fired, and add one record into the table Empl\_Del\_log for each record deleted.

Below are the code for creating a new log table called Empl\_Del\_log

-- DROP table Empl\_Del\_log;

CREATE TABLE Empl\_Del\_log (

Old\_Empno number (6),

OLD\_fname Varchar2 (20),

OLD\_lname varchar2 (25),

OLD\_sal number (8, 2),

OLD\_mgrno number (6),

Updated\_Date DATE,

Updated\_By Varchar2 (15),

Action Varchar2 (30)

);

Similar as Q1, the second step requires you to create a row level trigger named Empl\_Del\_row on table Employees. Whenever there is a command of delete on the table employees, the trigger will be fired, and it will add one record into the table Empl\_Del\_log for each record deleted.

After having successfully compiled the trigger, your program will run a DELETE commands.

delete Employees where manager\_id = 103;

In the forth step, your submission should include the output of Empl\_Del\_log table, that displays the new record in the log table. It will be a good practice, remember to rollback after you get the display of the results:

-- rollback;

Q3. (25 points) Conditional Predicates Row level Trigger. This question is based on the departments table.

First step, please build a log table Dept\_log as below:

CREATE TABLE Dept\_log(

OLD\_Deptno number (4),

NEW\_Deptno number (4),

OLD\_Deptname Varchar2 (30),

NEW\_Deptname Varchar2 (30),

OLD\_MgrID number (6),

NEW\_MgrID number (6),

OLD\_LocID number (4),

NEW\_LocID number (4),

Updated\_Date DATE,

Updated\_By Varchar2 (15),

Action Varchar2 (25)

);

To monitor the table Departments, we will create a conditional predicates row-level trigger called Dept\_Change. Whenever there is a command of insert, or delete, or update/change on column of manager\_id or location ID, the trigger will be fired, and one row will be inserted into the table Dept\_log.

Once you have successfully built the trigger, you are required to test your trigger using the following commands, display the correspondent output.

Column OLD\_Deptname format A10

Column NEW\_Deptname format A10

SELECT \* from Dept\_log ;

INSERT INTO departments VALUES (290, 'Test', null, 1700);

UPDATE departments set manager\_id = 103 WHERE department\_id = 290;

UPDATE departments set location\_id = 1800 WHERE department\_id = 290;

DELETE departments where department\_id = 290;

SELECT \* from Dept\_log ;

rollback;

Q4. Package (25 point).

Please be patient when you read this question. This question requires you to edit, reorganize the codes from a PL/SQL block environment into package module, then invoke the functions defined in package.

Following codes are answers for Q3. hw 6. In a PL/SQL block, we have defined two (nested, or say local) functions with the same name as Dept\_Head\_FName with different parameter names and data types. Both functions have one IN parameter. The first function use deptno (department\_id) as its IN parameter, its data type is number. The second function has dept\_name (department\_name) as its IN parameter, the data type is varchar2. Both functions will return the full name of the head/manager of that department.

Declare

Dpt\_id employees.department\_id%TYPE := 60;

dept\_name varchar2 (30) := 'it' ;

Dpt\_head\_fname varchar2 (47);

Function Dept\_Head\_FName

( Deptno IN number)

RETURN varchar2

IS

Dept\_Head\_fullname varchar2 (47);

BEGIN

SELECT first\_name ||' '|| last\_name INTO Dept\_Head\_fullname

FROM employees WHERE employee\_id =

(select manager\_id from departments where department\_id = Deptno);

RETURN Dept\_Head\_fullname;

END Dept\_Head\_FName ;

Function Dept\_Head\_FName

( Dept\_name IN varchar2)

RETURN varchar2

IS

Dept\_Head\_fullname varchar2 (47);

BEGIN

SELECT first\_name ||' '|| last\_name INTO Dept\_Head\_fullname

FROM employees WHERE employee\_id =

(select manager\_id from departments

where upper (department\_name) = upper (Dept\_name));

RETURN Dept\_Head\_fullname;

END Dept\_Head\_FName ;

BEGIN

Dpt\_head\_fname := Dept\_Head\_FName ( Dpt\_id);

DBMS\_OUTPUT.PUT\_LINE( 'For the department with ID '|| dpt\_id ||':'||

CHR (10)|| 'its dept head name is: '|| Dpt\_head\_fname || '.');

Dpt\_head\_fname := Dept\_Head\_FName ( dept\_name );

DBMS\_OUTPUT.PUT\_LINE( 'For the department with name '|| Dept\_name ||':'||

CHR (10)|| 'its dept head name is: '|| Dpt\_head\_fname || '.');

END;

/

The above functions will exist only in the life of that block. To store then share these functions for the future use, a good practice will change the codes into package module.

In your program, define a package named Pkg\_Dept\_Head. In the specification section, define two functions with the same name as Dept\_Head\_Fname, similar as in the provided codes.

In the package body, you will provide the function programs to return the right value for these functions. In the executive section of the body, you do not need to write any code for initializing. It can be simply as

BEGIN

null;

END;

or skipped the “begin null; ”, only with “end; ”.

After having successfully compiled the package specification and its body, write a PL/SQL block. It will call the function twice. One uses 60 as department\_id. The second time, uses ‘IT’ as actual parameter value. Your program will print out the results.