CSC 352 / 452: Database Programming

assignment #6 (60 Points)

**CSC 352/452-501: Due on Tuesday, 8/15/2017 at 11:59PM**

**CSC 352/452-510: Due on Wednesday, 8/16/2017 at 11:59PM**

Unless prior arrangements are made, homework turned in late will not be accepted. However, homework turned in within 24 hours late will be graded at 50% credit.

**If there is a syntax error anywhere in your program, you will receive 0 points for the program.**

* Please read the assignment carefully. You will receive 0 points if you use different tables (names, columns, or data types) or trigger headers.
* Please note that only TEXT files will be accepted. All other file types (e.g., DOC, DOCX, RTF, PDF, JPG, or ZIP) will be rejected. In D2L, only the most recent submission is kept.
* The autonomous transactions are not required.
* Please review your assignment file before submitting it to make sure you have the correct one. It is your responsibility to upload the correct assignment file.

**1) (30 points)**

Employees may move to different departments. We want to keep track of the departments where each employee has been. To do so we create a new table TAB\_EMP\_DEPT\_HIST that keeps track of such history.

TAB\_EMP\_DEPT\_HIST(EMPLOYEE\_ID, EMPLOYEE\_NAME, OLD\_DEPARTMENT\_NAME, NEW\_DEPARTMENT\_NAME, OPERATION, EFFECTIVE\_DATE);

Write a trigger EMP\_DEPT\_HIST\_TRG that monitors the EMPLOYEE table as follows.

* When a row (record) is inserted into the EMPLOYEE table, the trigger automatically inserts a row (record) into the TAB\_EMP\_DEPT\_HIST table **in any situations**.
* The OLD\_DEPARTMENT\_NAME field is always 'X'.
* If the new DEPARTMENT\_ID is not NULL, find the new department name from the DEPARTMENT table based on the new DEPARTMENT\_ID.

If the new DEPARTMENT\_ID is NULL, the NEW\_DEPARTMENT\_NAME field will be 'X'.

* The OPERATION field is always'INSERT'.
* When an employee changes his/her department (the old DEPARTMENT\_ID is not equal to the new DEPARTMENT\_ID), the trigger automatically inserts a row (record) into the TAB\_EMP\_DEPT\_HIST table. (If both the old DEPARTMENT\_ID and new DEPARTMENT\_ID are NULL (from NULL department to NULL department), the trigger does not insert a row (record) into the TAB\_EMP\_DEPT\_HIST table.)
* If the old DEPARTMENT\_ID is not NULL, find the old department name from the DEPARTMENT table based on the old DEPARTMENT\_ID.

If the old DEPARTMENT\_ID is NULL, the OLD\_DEPARTMENT\_NAME field will be 'X'.

* If the new DEPARTMENT\_ID is not NULL, find the new department name from the DEPARTMENT table based on the new DEPARTMENT\_ID.

If the new DEPARTMENT\_ID is NULL, the NEW\_DEPARTMENT\_NAME field will be 'X'.

* The OPERATION field is always'UPDATE'.
* The SYSDATE can be used in the EFFECTIVE\_DATE column.
* You can assume that the insert/update statements do not violate the integrity constraints between the DEPARTMENT and EMPLOYEE tables.
* No temporary table/view/procedure/function is allowed in your trigger.
* You can only use the DEPARTMENT, EMPLOYEE, and TAB\_EMP\_DEPT\_HIST tables in your trigger. You will receive 0 points if you use a different table (e.g., different table names, column names, or data types) in your trigger.

Step 1) (0 point) Create the TAB\_EMP\_DEPT\_HIST table,

CREATE TABLE TAB\_EMP\_DEPT\_HIST

(

EMPLOYEE\_ID NUMBER(4) NOT NULL,

EMPLOYEE\_NAME VARCHAR2(50) NOT NULL,

OLD\_DEPARTMENT\_NAME VARCHAR2(100) NOT NULL,

NEW\_DEPARTMENT\_NAME VARCHAR2(100) NOT NULL,

OPERATION VARCHAR2(50) NOT NULL,

EFFECTIVE\_DATE DATE NOT NULL

);

Step 2) Create the trigger EMP\_DEPT\_HIST\_TRG.

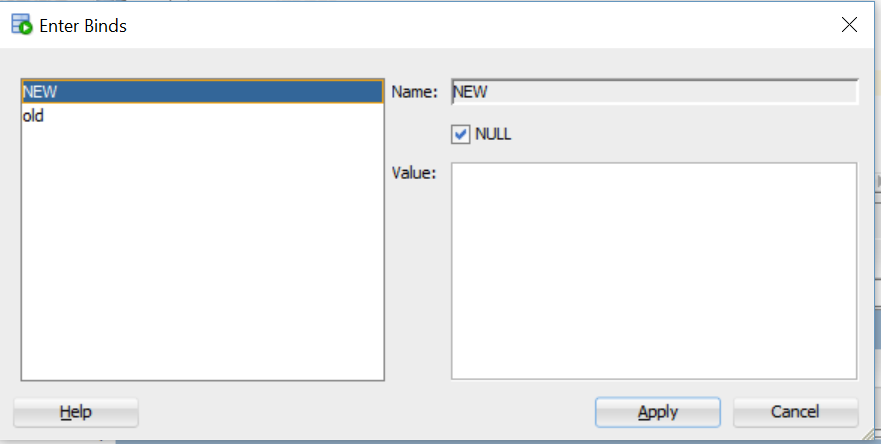
You will receive 0 points if you use a different trigger name.

Step 3) Test your trigger.

You need to create/run some test cases to check your trigger. You do not need to submit your test cases.

Q & A)

Q: I keep getting the following prompt when I try to use :OLD and :NEW when doing this week's homework. Does this have to do with my system settings or do I need to enter something on this screen?



A: You should always click the 2nd button or F5 to run your PL/SQL program. You can find that from lecture note 1, page 8.

**2) (30 points)**

Create a trigger called EMP\_MIN\_SAL\_TRG on the EMPLOYEE table. When an INSERT or UPDATE statement is issued against the EMPLOYEE table, the trigger is fired to ensure that the value of the SALARY column meets **the criteria in the TAB\_MINIMUM\_SALARY table** in any situations. (For example, you can find that the minimum salary for a programmer is 800 from the TAB\_MINIMUM\_SALARY table. Your trigger ensures that the salary for a programmer in the EMPLOYEE table is greater than or equal to 800 in any situations.)

Step 1) (0 point) Create a table TAB\_MINIMUM\_SALARY as follows.

CREATE TABLE TAB\_MINIMUM\_SALARY

(

JOB\_TITLE VARCHAR2(100) PRIMARY KEY,

MINIMUM\_SALARY NUMBER(7, 2) NOT NULL

);

Step 2) (0 point) Populate the TAB\_INIMUM\_SALARY table as follows.

INSERT INTO tab\_minimum\_salary VALUES ('BUSINESS ANALYST', 2800);

INSERT INTO tab\_minimum\_salary VALUES ('CHIEF ACCOUNTANT', 2900);

INSERT INTO tab\_minimum\_salary VALUES ('DATABASE ADMINISTRATOR', 2800);

INSERT INTO tab\_minimum\_salary VALUES ('PRESIDENT', 4800);

INSERT INTO tab\_minimum\_salary VALUES ('PROGRAMMER', 800);

INSERT INTO tab\_minimum\_salary VALUES ('PUBLIC ACCOUNTANT', 2400);

INSERT INTO tab\_minimum\_salary VALUES ('REPORTING ANALYST', 2000);

INSERT INTO tab\_minimum\_salary VALUES ('SALES CONSULTANT', 1500);

INSERT INTO tab\_minimum\_salary VALUES ('SALES EXECUTIVE', 2800);

INSERT INTO tab\_minimum\_salary VALUES ('SALES REPRESENTATIVE', 2000);

INSERT INTO tab\_minimum\_salary VALUES ('TEST ANALYST', 1500);

INSERT INTO tab\_minimum\_salary VALUES ('VICE PRESIDENT', 3800);

INSERT INTO tab\_minimum\_salary VALUES ('X', 800);

COMMIT;

Step 3) Create the trigger EMP\_MIN\_SAL\_TRG.

* The TAB\_MINIMUM\_SALARY table is read-only. Your trigger **cannot** modify any rows in the TAB\_MINIMUM\_SALARY table.
* You must get the minimum salaries from the TAB\_MINIMUM\_SALARY table in your trigger.
* The job is not case sensitive (e.g., PROGRAMMER = Programmer).
* Hard coding, except the string 'X', is not allowed in your trigger (e.g., IF UPPER (job\_title) = 'PROGRAMMER' THEN v\_min\_sal = 800 …).
* If the job title cannot be found from the TAB\_MINIMUM\_SALARY table, the job title is considered as “X”. (e.g., the job title “TBA” is not in the TAB\_MINIMUM\_SALARY table, you need to check whether the salary is equal to or greater than the minimum salary for UPPER(job\_title) = 'X'.)
* If the salary is equal to or greater than the minimum salary of the corresponding job, your trigger does not change anything.
* If the salary is less than the minimum salary of the corresponding job title, your trigger increases the salary to the minimum salary of the corresponding job title.
* No temporary table/view/procedure/function is allowed in your trigger.
* To avoid a **mutating table error**, please take a look at examples on page 8 of class handout 8. (Hint: you cannot use some INSERT/UPDATE statements to modify the EMPLOYEE table in your trigger.)
* You will receive 0 points if you use a different table (e.g., different table names, column names, or data types) in your trigger.
* You will receive 0 points if you use a different trigger name.
* If you modified the EMPLOYEE table created in Assignment #1, please delete and re-populate it.

Step 4) Test your trigger.

You need to create/run some test cases to check whether the values of the SALARY column in the EMPLOYEE table meet the criteria in the TAB\_MINIMUM\_SALARY table **in any situations**. You do not need to submit your test cases.

**Please submit a text file containing all the source codes to D2L by the due date.**