CSC 352 / 452: Database Programming

assignment #5 (60 Points)

**CSC 352/452-501: Due on Wednesday, 8/9/2017 at 11:59PM**

**CSC 352/452-510: Due on Thursday, 8/10/2017 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 procedure headers (names, parameters, or data types).
* 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.
* 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) (0 Point)**

The DPUCC\_USER table stores information about user accounts.

DPUCC\_USER(

**DPUCC\_ID**,

PASSWORD,

LAST\_SUCCESSFUL\_LOGIN\_TIME,

LAST\_FAILED\_LOGIN\_TIME,

FAILED\_PASSWORD\_ATTEMPT\_COUNT,

IS\_ACCOUNT\_LOCKED\_OUT,

LAST\_ACCOUNT\_LOCKED\_OUT\_TIME,

LAST\_PASSWORD\_CHANGED\_TIME);

DPUCC\_ID: Primary Key

FAILED\_PASSWORD\_ATTEMPT\_COUNT: the number of consecutive failed password attempts

IS\_ACCOUNT\_LOCKED\_OUT: 'Y' – the account is locked out

'N' – the account is not locked out

Create and populate the DPUCC\_USER table as described below.

CREATE TABLE DPUCC\_USER

( DPUCC\_ID NUMBER PRIMARY KEY,

PASSWORD VARCHAR2(30) NOT NULL,

LAST\_SUCCESSFUL\_LOGIN\_TIME DATE,

LAST\_FAILED\_LOGIN\_TIME DATE,

FAILED\_PASSWORD\_ATTEMPT\_COUNT NUMBER,

IS\_ACCOUNT\_LOCKED\_OUT CHAR NOT NULL

CONSTRAINT CK\_DPUCC\_USER CHECK

(IS\_ACCOUNT\_LOCKED\_OUT IN ('Y','N')),

LAST\_ACCOUNT\_LOCKED\_OUT\_TIME DATE,

LAST\_PASSWORD\_CHANGED\_TIME DATE);

/

INSERT INTO DPUCC\_USER

VALUES(82001, '99CpsBTKpN1',TO\_DATE('02-APR-2015 14:05:08', 'DD-MON-YYYY HH24:MI:SS'),

NULL, 0, 'N', NULL, TO\_DATE('02-DEC-2014 13:05:08', 'DD-MON-YYYY HH24:MI:SS'));

INSERT INTO DPUCC\_USER

VALUES(82002, 'ZWNWnQJT901', TO\_DATE('02-DEC-2014 16:15:01', 'DD-MON-YYYY HH24:MI:SS'),

NULL, 0, 'N', NULL, TO\_DATE('02-NOV-2014 11:11:18', 'DD-MON-YYYY HH24:MI:SS'));

INSERT INTO DPUCC\_USER

VALUES(82003, 'gc88Wmvpx81', TO\_DATE('01-APR-2015 19:15:08', 'DD-MON-YYYY HH24:MI:SS'),

TO\_DATE('05-MAY-2015 21:45:18', 'DD-MON-YYYY HH24:MI:SS'), 1, 'N', NULL, TO\_DATE('30-JAN-2015 23:01:01', 'DD-MON-YYYY HH24:MI:SS'));

INSERT INTO DPUCC\_USER

VALUES(82004, 'KcxweSYg5551', TO\_DATE('03-JAN-2015 14:12:33', 'DD-MON-YYYY HH24:MI:SS'),

TO\_DATE('06-MAY-2015 09:12:22', 'DD-MON-YYYY HH24:MI:SS'), 5, 'Y', TO\_DATE('06-MAY-2015 09:12:22', 'DD-MON-YYYY HH24:MI:SS'), NULL);

INSERT INTO DPUCC\_USER

VALUES(82005, 'CDYe44BBXd11', TO\_DATE('22-MAR-2015 05:22:18', 'DD-MON-YYYY HH24:MI:SS'),

NULL, 0, 'N', NULL, NULL);

INSERT INTO DPUCC\_USER

VALUES(82006, 'vhSDHMDg6661', TO\_DATE('07-FEB-2015 04:00:08', 'DD-MON-YYYY HH24:MI:SS'),

NULL, 0, 'N', NULL, TO\_DATE('01-FEB-2015 04:35:01', 'DD-MON-YYYY HH24:MI:SS'));

COMMIT;

**2) (30 Points)**

Based on the DPUCC\_USER table created in (1), create a procedure for validating user login.

The procedure header is

CREATE OR REPLACE PROCEDURE check\_dpucc\_user\_login

(

in\_dpucc\_id NUMBER,

in\_password VARCHAR2,

out\_code OUT VARCHAR2

)

(You cannot change the procedure header. You will receive 0 points if you use a different procedure header. You will receive 0 points if you submit more than one procedure. No DBMS\_OUTPUT statement is needed in the procedure.)

**An account will be locked after 5 consecutive failed login attempts.**

A password is case-sensitive.

The logical steps are as follows;

1. The value of in\_dpucc\_id is not in the DPUCC\_ID column of the DPUCC\_USER table.

You assign 'err\_bad\_id' to the out\_code parameter.

1. The value of in\_dpucc\_id is in the DPUCC\_ID column of the DPUCC\_USER table, but the corresponding account is locked out (IS\_ACCOUNT\_LOCKED\_OUT = 'Y').

You assign 'err\_locked\_out' to the out\_code parameter.

1. The value of in\_dpucc\_id is in the DPUCC\_ID column of the DPUCC\_USER table and the value of in\_password matches the corresponding password in the table. The corresponding row in the DPUCC\_USER table will be updated:

* LAST\_SUCCESSFUL\_LOGIN\_TIME <== SYSDATE
* FAILED\_PASSWORD\_ATTEMPT\_COUNT <== 0
* LAST\_FAILED\_LOGIN\_TIME <== NULL

You assign 'ok' to the out\_code parameter.

1. The value of in\_dpucc\_id is in the DPUCC\_ID column of the DPUCC\_USER table, but the value of in\_password does not match the corresponding password in the table. You update the corresponding row in the DPUCC\_USER table and the out\_code parameter:

* FAILED\_PASSWORD\_ATTEMPT\_COUNT <==

FAILED\_PASSWORD\_ATTEMPT\_COUNT + 1

* LAST\_FAILED\_LOGIN\_TIME <== SYSDATE
* Case 1: FAILED\_PASSWORD\_ATTEMPT\_COUNT != 5
* You assign 'err\_bad\_pwd' to the out\_code parameter.
* Case 2: FAILED\_PASSWORD\_ATTEMPT\_COUNT = 5
* IS\_ACCOUNT\_LOCKED\_OUT <== 'Y'
* LAST\_ACCOUNT\_LOCKED\_OUT\_TIME <== SYSDATE
* You assign 'err\_bad\_pwd\_5' to the out\_code parameter.

Hints: UPDATE DPUCC\_USER SET …… WHERE DPUCC\_ID = in\_dpucc\_id;

out\_code := ……;

You need to test your procedure with different parameters in a **PL/SQL block**.

**3) (30 Points)**

Based on the DPUCC\_USER table created in (1), create a procedure to change the password for a given user ID.

* A password is case-sensitive.
* A password must be between 8 and 15 characters in length.
* A password must include at least one uppercase alphabetic character (A-Z), one lowercase alphabetic character (a-z), and one numeric character (0-9).
* You need to update the **PASSWORD** and **LAST\_PASSWORD\_CHANGED\_TIME** columns if the password can be reset. (Oracle SYSDATE function returns the current date and time.)

The procedure header is

CREATE OR REPLACE PROCEDURE check\_dpucc\_user\_password

(

in\_dpucc\_id NUMBER,

in\_current\_password VARCHAR2,

in\_new\_password VARCHAR2,

out\_code OUT VARCHAR2

)

(You cannot change the procedure header. You will get a zero point if a different procedure header is used. You will receive 0 points if you submit more than one procedure. No DBMS\_OUTPUT statement is needed in the procedure.)

The logical steps are as follows;

1. The value of in\_dpucc\_id is not in the DPUCC\_ID column of the DPUCC\_USER table. The password cannot be reset. You assign 'err\_bad\_id' to the out\_code parameter.
2. The current password is not correct. The password cannot be reset. You assign 'err\_bad\_curr\_pwd' to the out\_code parameter.
3. The new password is the same as the current password. The password cannot be reset. You assign 'err\_bad\_new\_pwd\_1' to the out\_code parameter.
4. The new password is too long or too short. The password cannot be reset. You assign 'err\_bad\_new\_pwd\_2' to the out\_code parameter.
5. The new password does not include an uppercase alphabetic character. The password cannot be reset. You assign 'err\_bad\_new\_pwd\_3' to the out\_code parameter.
6. The new password does not include a lowercase alphabetic character. The password cannot be reset. You assign 'err\_bad\_new\_pwd\_4' to the out\_code parameter.
7. The new password does not include a numeric character. The password cannot be reset. You assign 'err\_bad\_new\_pwd\_5' to the out\_code parameter.
8. The password can be reset. You update the corresponding row in the DPUCC\_USER table:

* PASSWORD <== in\_new\_password
* LAST\_PASSWORD\_CHANGED\_TIME <== SYSDATE

You assign 'ok' to the out\_code parameter.

Hint: To test a string for alphabetic and numeric characters, you may use the REGEXP\_LIKE function.

*(REGEXP\_LIKE is similar to the LIKE condition, except REGEXP\_LIKE performs regular expression matching instead of the simple pattern matching performed by LIKE. This condition evaluates strings using characters as defined by the input character set.)*

Examples:

1) SELECT COUNT(\*) INTO v\_n FROM DUAL WHERE REGEXP\_LIKE(v\_1, '[A-Z]');

v\_n = 0: v\_1 does not include an uppercase alphabetic character.

2) SELECT COUNT(\*) INTO v\_n FROM DUAL WHERE REGEXP\_LIKE(v\_1, '[a-z]');

v\_n = 0: v\_1 does not include a lowercase alphabetic character.

3) SELECT COUNT(\*) INTO v\_n FROM DUAL WHERE REGEXP\_LIKE(v\_1, '[0-9]');

v\_n = 0: v\_1 does not include a numeric character.

You need to test your procedure with different parameters in a **PL/SQL block**.

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