CSCI 4125/5125 Course Project
Data Models and Database Systems
Spring 2022
Course Project
Phase 5: PL/SQL & JDBC

Due: Sunday, 4/24 @ 11:59pm
Reading: Silberschatz Chapter 5

Submission Guidelines:
1. This assignment is worth 100 points for all students.

2. It is your responsibility to make sure all files are readable and submitted on time.

Submission:

- Part A (PL/SQL): submit a single .sql file containing all three triggers for a total of 60 points.
- Part B (JDBC): submit a Java source code file that loads data into the timecard table for a total of 40 points.
    - Submit a Java source code file that manipulates query results for +20 points extra credit.

Part A. PL/SQL Triggers (60 points).

Your task is to write three triggers that will maintain the number of beds a nurse is monitoring. While you are not required to use DBMS_OUTPUT.PUT_LINE, I highly recommend using it for testing and debugging your code. Note, this is an example of a computed column that we saw in Chapter 6: E-R Modeling. First, add a column to the Nurse table to store this value using the following command:

```
ALTER TABLE Nurse ADD BedsMonitored NUMBER DEFAULT 0;
```

Trigger #1 (15 points). Write a trigger that will fire when you INSERT a row into the Bed table. This trigger will check the value of BedsMonitored for the corresponding Nurse's ID. If Nurse.BedsMonitored is less than 2, then a nurse can monitor more beds. If Nurse.BedsMonitored is equal to 2, then the nurse is too busy so your trigger will cancel the INSERT and display a custom error message stating that the Nurse is too busy. Also, be sure to handle the inserts for bed where the NurseID is NULL. One way I suggest doing this is too to handle the `NO_DATA_FOUND EXCEPTION` (i.e., `WHEN NO_DATA_FOUND THEN`).

You can test your trigger with the following:

- Delete all records from Bed: **`DELETE FROM Bed;`**

- Run your .sql script containing your INSERT statements for the Bed table.

- The insert for Bed B28 should produce an error message that looks like:

```
INSERT INTO bed VALUES('B28', 1224, 'Long-Term Care', 'P11', 'N04')
Error report -
ORA-20201: Nurse is too busy!
ORA-06512: at "JAYSTUDENT.ADDBED", line 15
ORA-04088: error during execution of trigger 'JAYSTUDENT.ADDBED'
```

- The query SELECT * FROM Nurse should return the following output:

```
ID  NAME                          SALARY SUP BEDSMONITORED
--- ------------------------- ---------- --- -------------
N01 John Nicklow                  120000                 0
N03 Chris Summa                    88000 N01             2
N05 Vassil Roussev                 90000 N01             0
N09 Tamjid Hoque                   88000 N01             1
N02 James Wagner                   75000 N05             1
N04 Ben Samuel                     79000 N03             2
N06 Sam Hoyt                       82000 N03             0
N07 Ted Holmberg                   81000 N05             0
N08 Matt Toups                     80000 N09             0
N10 Hyunguk Yoo                    80000 N09             0
```

- The Bed table should only have 29 records in it since the insert for B28 was rejected.

Trigger #2 (15 points). Write a trigger that will fire when a user attempts to DELETE one or more rows from BED. This trigger will update the Nurse.BedsMonitored for the corresponding Nurse's ID by decreasing the value by one each time a row is removed from BED.

You can test your trigger with the following:

- Choose a Bed to delete. Ex: `DELETE FROM Bed WHERE BedNumber = 'B07'`
- `SELECT * FROM Nurse` should return the following output if you delete B07:

```
ID  NAME                         SALARY SUP BEDSMONITORED
--- ------------------------- ---------- --- -------------
N01 John Nicklow                 120000                  0
N03 Chris Summa                   88000 N01              2
N05 Vassil Roussev                90000 N01              0
N09 Tamjid Hoque                  88000 N01              0
N02 James Wagner                  75000 N05              1
N04 Ben Samuel                    79000 N03              2
N06 Sam Hoyt                      82000 N03              0
N07 Ted Holmberg                  81000 N05              0
N08 Matt Toups                    80000 N09              0
N10 Hyunguk Yoo                   80000 N09              0
```

Trigger #3 (30 points). Write a trigger that fires when the nurse ID for a bed record is changed (i.e., an UPDATE). This trigger will update the Nurse.BedsMonitored for both the previous Nurse and the new Nurse using their respective ID's. If the existing Nurse.BedsMonitored value for the new Nurse in the update is equal to 2, then the nurse is too busy so your trigger will cancel the UPDATE and display an error message stating that the Nurse is too busy (similar to the first trigger).

You can test your trigger with the following:

- Update a bed's NurseID for a nurse that does not monitor too many beds. Here's one query you can use, and the corresponding content from the Nurse table.

```
UPDATE Bed
SET NurseID = 'N02'
WHERE BedNumber = 'B01';

SELECT * FROM Nurse;

ID  NAME                         SALARY SUP BEDSMONITORED
--- ------------------------- ---------- --- -------------
N01 John Nicklow                 120000                  0
N03 Chris Summa                   88000 N01              2
N05 Vassil Roussev                90000 N01              0
N09 Tamjid Hoque                  88000 N01              0
N02 James Wagner                  75000 N05              2
N04 Ben Samuel                    79000 N03              2
N06 Sam Hoyt                      82000 N03              0
N07 Ted Holmberg                  81000 N05              0
N08 Matt Toups                    80000 N09              0
N10 Hyunguk Yoo                   80000 N09              0
```

- Update a bed’s NurseID for a nurse that is already monitoring the max number of beds allowed. Here’s one query you can use and the corresponding error message.

```
UPDATE Bed
SET NurseID = 'N02'
WHERE BedNumber = 'B02';

Error report -
ORA-20201: Nurse is too busy!
ORA-06512: at "JAYSTUDENT.UPDATEBED", line 15
ORA-04088: error during execution of trigger 'JAYSTUDENT.UPDATEBED'
```

- Update a bed’s NurseID that switches the nurse’s monitoring the bed. Here’s one query you can use, and the corresponding content from the Nurse table.

```
UPDATE Bed
SET NurseID = 'N05'
WHERE BedNumber = 'B09';

SELECT * FROM Nurse;

ID  NAME                          SALARY SUP BEDSMONITORED
--- ------------------------- ---------- --- -------------
N01 John Nicklow                  120000                 0
N03 Chris Summa                    88000 N01             2
N05 Vassil Roussev                 90000 N01             1
N09 Tamjid Hoque                   88000 N01             0
N02 James Wagner                   75000 N05             1
N04 Ben Samuel                     79000 N03             2
N06 Sam Hoyt                       82000 N03             0
N07 Ted Holmberg                   81000 N05             0
N08 Matt Toups                     80000 N09             0
N10 Hyunguk Yoo                    80000 N09             0
```

Part B. JDBC

4. Loading Data using Prepared Statements (40 points)

The goal of this part is to load data using SQL insert commands executed in your Java application. You are required to use prepared statements for this part. Since one advantage of prepared statements is to avoid SQL injection attacks, you cannot use a variable number of parameters. For that reason, we will only insert data for one table, Timecard.

i. Connect to the Oracle database we have been using in the course with your username and password.

ii. Using your JDBC connection, issue the command **`DELETE FROM Timecard;`**. This will delete all the records in the table; it is unnecessary to drop and create the table. Note that this table should not be referenced by any foreign keys so you should not have referential integrity errors.

iii. Read the data line by line from the corresponding CSV file you used earlier in the project. You should be able to reuse your code or the example code I posted from earlier in the project.

iv. In the for loop where you read the file line by line, execute an INSERT statement to insert the data from that line using prepared statements. Using prepared statements and splitting each line on the comma will simplify this task.

v. Close your database connection.

5. Manipulating Query Results (20 points, extra credit)

The goal of this part is to retrieve data using SQL select queries executed in your Java application and then manipulate the query results in Java. In other words, you will retrieve records from your database and use Java to process the data for what would normally be fairly straightforward SQL queries. Your Java application should contain the following:

i. Connect to the Oracle database we have been using in the course with your username and password.

ii. Issue the following select queries for your corresponding database:

   - **`SELECT * FROM Nurse;`**

iii. Using the result set from only the query in the previous step and Java, print the results for the following.

   - Calculate the average salary for all nurses.

iv. Issue the following two select queries:

   - **`SELECT * FROM Physician;`**
   - **`SELECT * FROM Timecard;`**

v. Using the result set from the queries in the previous step and Java, print the following for your corresponding database:

   - Find the names of all physicians who have not submitted a timecard.

vi. Close your database connection.