Assignment #3

CS 348 - Spring 2015

Due on Saturday, July 25, 2015, 9 AM

For instructions on how to submit your assignment check the course website.

Question 1.

Consider the following Java class structure, where each comented line will be replaced by the contents of the files described below.

```
// 9_imports.txt

public class JDBCExample {
  public static void main(String[] args) throws Exception {

    // 1_driver.txt

    // 2_connect.txt

    // 3_autocommit.txt

    // 4_create.txt

    // 5_insert.txt

    // 6_procedure.txt

    // 7_select.txt

    // 8_close.txt
  }
}
```

Write the code described in each of the following files:

1_driver.txt – the file contains **a single line** with the Java code for loading the PostgreSQL driver.

2_connect.txt – the file contains **a single line** with the Java code for connecting to the a database having:

- host : **localhost**
- port : **5432**
- database name : **postgres**
- username : **postgres**
- password : **postgres**

3_autocommit.txt – the file contains **a single line** with the Java code for setting the auto commit to **false** for the established connection.

4_create.txt – the file contains all the code necessary (including variable declarations) to perform **a single transaction** to define the tables *emp*, *dept* and *works*. The SQL statements for creating these tables are listed below and available on the course website. Your code **must use** a *Statement* object and make **3 calls** to the *executeUpdate* method.

```
CREATE TABLE emp
  (
     eid    NUMERIC(9, 0) PRIMARY KEY,
     ename  VARCHAR(30),
     age    NUMERIC(3, 0),
     salary NUMERIC(10, 2)
  );

CREATE TABLE dept
  (
     did       NUMERIC(2, 0) PRIMARY KEY,
     dname     VARCHAR(20),
     budget    NUMERIC(10, 2),
     managerid NUMERIC(9, 0) REFERENCES emp(eid)
  );

CREATE TABLE works
  (
     eid      NUMERIC(9, 0) REFERENCES emp,
     did      NUMERIC(2, 0) REFERENCES dept,
     pct_time NUMERIC(3, 0),
     PRIMARY KEY(eid, did)
  );
```

5_insert.txt – the file contains all the code necessary (including variable declarations) to perform **a single transaction** to populate the *emp* table with the tuples given in the file **emp.txt** available on the course website. The file will be given as a command-line argument to the class (*args[0]*). It is the only command-line argument used. Each tuple in the file must be inserted as a separate insert statement (i.e. your SQL insert statement must use placeholders in order to perform batch insertions). Your code **must use** a *PreparedStatement* object.

6_procedure.txt – the file contains all the code necessary (including variable declarations) to perform **a single transaction** to store the function *getnames(real)* in the database. The SQL statement for creating the function is listed below and available on the course website. Your code **must use** a *Statement* object.

```
CREATE FUNCTION getnames(minsalary real)
RETURNS refcursor AS
$BODY$
DECLARE mycurs refcursor;
BEGIN
  OPEN mycurs FOR
  SELECT DISTINCT ename
  FROM            emp
  WHERE           salary >= minsalary
  ORDER BY        ename ASC;
  RETURN mycurs;
END
$BODY$
LANGUAGE plpgsql;
```

7_select.txt – the file contains all the code necessary (including variable declarations) to perform **a single transaction** to call the function *getnames(real)* with the parameter 39000 and list the returned tuples on the standard out (*System.out*). To call the function your code **must use** a *CallableStatement* object.

8_close.txt – the file contains **a single line** with the Java code for closing the database connection. Closing statements and result sets **should have already been done** in the file where they were declared and used.

9_imports.txt – the file contains **all the imports necessary for the class to run**.

Some additional considerations:

- The class described above must be a runnable Java class once the content of each file will be copied in its designated place. The process of copying the files will be done automatically. It is your duty to ensure that the class can be executed. Therefore follow the specifications to the letter and test the class using the website resources before you submit it. The execution test will be worth half the marks for this exercise.
- Exception testing, while important, is not the focus of this exercise. The *throws Exception* is sufficent at this point. However you must make sure that with correct parameters (existing database, user, password, proper file path etc.) your class works.
- Do not submit any code in addition to what is required. You are, however, encouraged to take the application futher to learn more about accessing a database using JDBC.

Question 2.

Part I. We call a transaction that only reads database object a **read-only** transaction, otherwise the transaction is called a **read-write** transaction. Give brief answers to the following questions:

1. When does lock thrashing occur?
2. If the database system *has not reached* the the thrashing point and the number of *read-write transactions* is increased, will the database throughput increase or decrease? (answer using one word)
3. If the database system *has reached* the the thrashing point and the number of *read-write transactions* is increased, will the database throughput increase or decrease? (answer using one word)
4. If the number of *read-only transactions* is increased, will the database throughput increase or decrease? (answer using one word)
5. List three ways of tuning your system to increase transaction throughput.

Part II. Consider the following schema:

Suppliers(<u>*sid*: `integer`</u>, *sname*: `string`, *address*: `string`)
Parts(<u>*pid*: `integer`</u>, *pname*: `string`, *color*: `string`)
Catalog(<u>*sid*: `integer`, *pid*: `integer`</u>, *cost*: `real`)

The *Catalog* relation lists the prices charged for parts by *Suppliers*.
For each of the following transactions, **state the SQL isolation level** that you would use and explain **in one sentence** why you chose it.

1. A transaction that adds a new part to a suppliers catalog.
2. A transaction that increases the price that a supplier charges for a part.
3. A transaction that determines the total number of items for a given supplier.
4. A transaction that shows, for each part, the supplier that supplies the part at the lowest price.

Question 3.

Part I. Consider a relation R with five attributes $ABCDE$.
You are given the following dependencies: $A \rightarrow B$, $BC \rightarrow E$, and $ED \rightarrow A$.

1. List all keys for R (as groups of attributes separated by comma, e.g. A, BC, DEF).
2. Is R in 3NF? Explain your answer in one sentence.
3. Is R in BCNF? Explain your answer in one sentence.

Part II. Consider the attribute set $R = ABCDEGH$ and the FD set

$$F = AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G$$

1. For each of the listed attribute sets corresponding to your **section number**, do the following:
   (i) Compute the set of dependencies that hold over the set and write down a minimal cover.
   (ii) Name the strongest normal form that is not violated by the relation containing these attributes.
   (iii) Decompose it into a collection of BCNF relations if it is not in BCNF.

$$Section\ 1 : (a) ABC, (c) ABCEG, (e) ACEH$$

$$Section\ 2 : (a) ABC, (b) ABCD, (d) DCEGH$$

2. Is the following decomposition of $R = ABCDEG$, with the same set of dependencies F, is (a) dependency-preserving? (b) lossless-join?

$$Section\ 1 : AB, BC, ABDE, EG$$

$$Section\ 2 : ABC, ACDE, ADG$$