Programming Assignment 2

CSE 132A Fall 2015

Due: by 10:00pm on Friday, Dec 4 (electronically)

The goal of the second assignment is the development of a Java program that follows the money trail by calculating the transitive closure of a database relation that shows fund transfers between accounts.

As discussed in class, the transitive closure of a relation cannot be expressed in SQL without extension with recursion. For this reason you need the added expressive power of a general-purpose programming language, like Java. The project will also introduce you to the standard Java Database Connectivity (JDBC) interface for building Java applications on top of databases. This is an individual assignment. The usual criteria of academic integrity apply.

Computing Environment

- Your application will be built using Sqlite.
- The programming language will be Java.
- You will use JDBC to connect to the database.
  Find the JDBC driver for Sqlite, as well as documentation, at https://bitbucket.org/xerial/sqlite-jdbc.
  Follow the download link (https://bitbucket.org/xerial/sqlite-jdbc/downloads) and select
  sqlite-jdbc-3.8.11.2.java by xerial.

Database Schema Recall our schema of a bank:

customer (*name:* `string`, *credit:* `integer`)
account (*no:* `string`, *balance:* `float`)
depositor (*cname:* `string`, *ano:* `string`)

where depositor.*cname* and depositor.*ano* are foreign keys referencing customer, respectively account. As usual, primary keys are underlined. Add to these a table

transfer (*src:* `string`, *tgt:* `string`, *timestamp:* `date`, *amount:* `float`)

where *src* and *tgt* are foreign keys referencing account. Rows in this table state that a fund transfer has taken place at the given time, transfering the given amount from source account *src* to target account *tgt*.

The graph The transfer relation induces a relationship between customers, *funds (A,B)*, according to which customer A funds customer B if and only if A is the depositor of an account acc_A and B is the depositor of an account acc_B, such that there is a tuple in relation transfer whose *src* attribute is acc_A and whose target attribute is acc_B.

This connection between customers can be modeled as a directed graph in which customers are represented by nodes, and a tuple (c_1, c_2) in *funds* is an edge from node c_1 to node c_2.

The transitive closure You will compute the table

$$influence(from, to)$$

which records that customer *to* can be reached from customer *from* in the graph along a path of one or more edges. In other words, table *influence* is the transitive closure of relationship *funds*.

Requirements

- You will implement the computation of the transitive closure using semi-naive evaluation.
- Your Java program must create and fill table *influence* using the information in table *transfer*.
- You may create any other auxiliary helper tables you might need. For example, as shown in class, you will need a Delta table to detect termination of the computation and to use in the semi-naive evaluation. Any extra tables must be created from your Java program. Please make sure to clean up after your program, dropping at the end the *Delta* table as well as any other auxiliary tables you have created.
- Make sure that your program can be run repeatedly without requiring manual dropping of tables. To this end, **start the program by dropping the *influence* table and end it by dropping all auxiliary tables.** This will help you in development and will allow us to re-run your program on different inputs. We cannot intervene manually in 200 submissions and you risk losing the points for the affected tests.
- You must avoid as much as possible transferring data between your Java program and the database server. Recall that in real deployment scenarios, the Java client resides on a different machine than the database server so communication is expensive. This means that data manipulation should be done exclusively using SQL commands executed on the server, rather than by transferring data to the client and processing there. In other words, the client side should be used for control only, not for computation. Leave the heavy lifting to the SQL server, this is what it was designed for.

  In particular, store the output of your SQL queries directly at the server, without first pulling their tuples to the Java client. The sole exception is the query checking the emptiness of the Delta table; this query is allowed to send back to the client a *single tuple per iteration*.

  The result of your computation will therefore *not* be printed from your Java program, but must be simply left in the *influence* table at the server.

What to turn in Turn in a single self-contained Java source file that we can compile without needing any other file from you, and uses the JDBC driver for Sqlite. Assume that this driver is in the same directory as the source file.

1. The file should be named 'FTM_<ID>.java' where <ID> is your student ID. For example, if your student ID is A01234567, then you should submit a single file named FTM_A01234567.java.
2. Dont put your class into any java package.
3. The test will run on the TAs database where the *customer, account, depositor* and *transfer* tables will have been created for you. **So do not put create table statements in your submission, with the sole exception of the *influence* table that you do need to create!**
4. We will not be editing your source file, so the username and password for the sqlite account need to be given as arguments to the program, **not hardcoded into it!**

5. Before turning in, make sure your file compiles and runs properly on your machine using the following commands (use your own user name and password for the arguments):

   Test commands for Linux/Mac (In Windows, just change : to ;):

   javac -cp ./sqlite-jdbc-3.8.11.2.jar:. FTM_<your ID>.java
   java -cp ./sqlite-jdbc-3.8.11.2.jar:. FTM_<your ID> <your user name> <your password>

6. Include as the first line of each file, a line with your name and student ID commented out.

7. If you resubmit, change the filename to include the resubmission number as follows:

   FTM_<your ID>.java.<resubmission number>.

   The first resubmission has number 1.

8. Turn in by email to `deutsch@eng.ucsd.edu`, as a file attachment. Your email should have the subject **cse132a-pa2**.[1]

9. There will be an automatic email receipt, contact us if none arrives.

[1] Please, no variations on this subject line. For pa1, some of you forgot the -, others inserted spaces, others forgot the pa's number at the end. All these variations end up requiring manual intervention and risk loss of your submission.