Using lab computers for Project 2

From the lab machines, you can access the course database cs564instr as follows.

```
psql -h postgres.cs.wisc.edu cs564instr
```

There is a schema with your login available for you. You should be able to create tables and views in that schema.

So let's say your login is <cs564student>.

Then you should be able to create a table like:

```
create table <cs564student>.t1(a integer, b integer);
```

or a view like

```
create view <cs564student>.sales as select * from hw2.sales;
```

Tables for part1 are available in hw2 schema.

The following is not necessary but could be a convenience.

You can set schema search path to default to your schema as follows

```
set search_path to <cs564student>, hw2, public;
```

`show search_path;` will show you the schema search path.

For part 2, you'd need JDBC. Please follow the following instructions to use jdbc correctly on the lab machines.

For security reasons, the postgres service is only available from within the CS network so you need to connect to it from a CS computer (or from inside CS VPN).

Since you must connect using kerberos and SSL, you'll need to use java 1.8 version found by default in /usr/lib/java on the lab computers.

You need to download the JDBC postgres jar file. The lab only had luck with an older version (Download `postgresql-8.4-703.jdbc4.jar` from https://jdbc.postgresql.org/download.html). The jar file is also available in ~shatdal/data directory for your convenience.

You also need a login.config file with these contents:

```
pgjdbc {
      com.sun.security.auth.module.Krb5LoginModule required useTicketCache=true
debug=true;
```

};

You may simply copy this file from ~shatdal/data and put it in your working directory.

Here is a sample test program that could be used to get started.

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;


public class pgtest{
       /**
        * @param args
        * @throws Exception
        */
       public static void main(String[] args) throws Exception {
            String url = "jdbc:postgresql://stampy.cs.wisc.edu/cs564instr?
sslfactory=org.postgresql.ssl.NonValidatingFactory&ssl";
            Connection conn = DriverManager.getConnection(url);

            Statement st = conn.createStatement();

       ResultSet rs = st.executeQuery("select count(*) from hw2.sales”);
       while (rs.next()) {
           System.out.print("Row returned: ");
           System.out.println(rs.getInt(1));
       }
       // close up shop
       rs.close();
       st.close();
       conn.close();
     }
   }
```

Of course you'll need to use the proper schema prefixes for the table names. Note that connecting to postgres.cs.wisc.edu does NOT work with JDBC; you need to use it's real name "stampy.cs.wisc.edu".

Don't forget to compile and run with the following flags from the directory containing your class, your jdbc4.jar file *and* your login.config file:

Here is an example (using pgtest.java above):
$ javac pgtest.java
$ java -Djava.security.auth.login.config=login.config -classpath ./
postgresql-8.4-703.jdbc4.jar:. pgtest

However, if you update the CLASSPATH, you can avoid the extra typing. You can also create an alias “javasec” so that you don't have to type in the long security information in command line. Here is the addition to .bash_profile. I kept the jar in ~/jdbc.

```
export CLASSPATH=~shatdal/jdbc/postgresql-8.4-703.jdbc4.jar:.
alias javasec='java -Djava.security.auth.login.config=login.config'
```

With the above changes, you should be able to compile and run.

$ javac pgtest.java
$ javasec pgtest
Debug is true storeKey false useTicketCache true useKeyTab false doNotPrompt false ticketCache is null isInitiator true KeyTab is null refreshKrb5Config is false principal is null tryFirstPass is false useFirstPass is false storePass is false clearPass is false
Acquire TGT from Cache
Principal is shatdal@CS.WISC.EDU
Commit Succeeded

Row returned: 421570