COMP9120 Relational Database Systems

Tutorial Week 8: DB Application Programming

In this lab, we will look at database application programming using Java and develop a small JDBC client program which queries our University database that we have used already in the last tutorials. First ensure that you have created that database. If you have not done so already, create this schema by downloading it from Canvas and running it on PostgreSQL.

We will be using PostgreSQL JDBC to communicate with PostgreSQL database. You may want to check the online documentation at

https://jdbc.postgresql.org/documentation/head/index.html

Exercise 1. Establishing Database Connections

In this first step, we just want to make sure that we can successfully compile the client program and establish a first connection to your database.

1. Download W8_Java.zip to your Desktop from Canvas

2. Right click the zip file > Extract Here. This creates a folder that includes W8_Java in its name.
3. Start Eclipse
4. Click on Browse and select Desktop when prompted for a workspace by Eclipse.
5. In Eclipse, click on File > Import > General > Existing Projects into Workspace > Next.
6. Click Browse with the "Select root directory" radio button selected.
7. Browse to the folder you created in step 2, and click OK.
8. Make sure the Tute8 checkbox is checked.
9. Click Finish.
10. You should now have the Tute8 project successfully imported on the Project Explorer (Ask your tutor if you can't see the project explorer).
11. Now browse to the JDBCclient.java file on the package explorer and replace “unikey” and
“password” with your UNIKEY and PostgreSQL password, as shown in below:

```
//Connection parameters – Enter your login and Password here
private final String userid = “y22s1c9120_UNIKEY”;
private final String passwd = “password”;
private final String myHost = “soit-db-pro-2.ucc.usyd.edu.au”;
```

12. Now Execute and Run the program by clicking on the Run menu and then clicking on Run.

Note: Be sure to save source code files after making any changes.

If everything works fine, you should see the message "You are successfully connected to PostgreSQL server." printed on the screen after a short while, followed by a list of course information, similar to the following picture. The query used to produce this output can be found in the listUnits() method of the JDBCclient class.

```
You are successfully connected to PostgreSQL server.
COMP5046 - Statistical Natural Language Processing (6cp) 2010-S1
COMP5138 - Database Management Systems (6cp) 2006-S2
COMP5138 - Database Management Systems (6cp) 2010-S1
COMP5338 - Advanced Data Models (6cp) 2006-S1
COMP5338 - Advanced Data Models (6cp) 2006-S2
INFO1003 - Introduction to IT (6cp) 2006-S1
INFO1003 - Introduction to IT (6cp) 2006-S2
INFO2005 - Database Management Introductory (3cp) 2004-S2
INFO2120 - Database Systems I (6cp) 2006-S1
INFO2120 - Database Systems I (6cp) 2009-S1
INFO2120 - Database Systems I (6cp) 2010-S1
INFO3005 - Organisational Database Systems (3cp) 2005-S1
INFO3404 - Database Systems II (6cp) 2008-S2
```

Exercise 2. Read-Only Database Access

Next, we want to fill the JDBC skeleton program with some life. You shall extend it with a method, which uses the JDBC PreparedStatement and ResultSet classes to dynamically query your university database.

a) Extend the existing listUnits() function so that it also lists for each unit the name of the faculty member who was teaching it.

b) Extend the existing listUnits() function so that it takes a parameter 'name' and only lists the courses taught by a given faculty member.

Exercise 3. Adding a new query

Now try adding a new method from scratch. We wish to be able to obtain a student's transcript in the Student Registration System.

a) In pgAdmin, write and check a query to get the details of all units completed by a student. This should include details for uosCode, uosName, credits, semester, year, grade.

b) Based upon the listUnits() method, write a method listTranscript(int studentID) that performs the same query as you just wrote and uses cursor (ResultSet) to print out a student's transcript.

c) Create the stored procedure by executing the script below in pgAdmin, (it includes the query required in 3 (b)):

```
CREATE OR REPLACE FUNCTION listTranscript (studentID INTEGER) RETURNS REFCURSOR
AS $$
```

```
DECLARE
        vCursor REFCURSOR;
BEGIN
        OPEN vCursor FOR
        SELECT uosCode, uosName, credits, semester, year, grade
        FROM Transcript JOIN UnitOfStudy USING(uosCode)
        WHERE studId=studentID
        ORDER BY uosCode,year,semester;
        RETURN vCursor;
END; $$ LANGUAGE plpgsql;
```

d) (Trickier) Update your listTranscript method to execute your stored procedure instead of using the query directly. Note you will need to use a CallableStatement instead of a PreparedStatement.