School of Computer Science

COMP9120 Database Management Systems

Tutorial Week 7: DB Application Programming

In this lab, we will look at database application programming using Python and develop a small Python 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 haven't done so already, create this schema by downloading it from Canvas and running it on PostgreSQL.

We will be using **Psycopg2** to communicate with PostgreSQL database. You may want to check the online documentation of Psycopg2 at the https://www.psycopg.org/docs/ Specifically, the following pages will be particularly useful.

https://www.psycopg.org/docs/usage.html

https://www.psycopg.org/docs/connection.html

https://www.psycopg.org/docs/cursor.html

Exercise 1. Establishing Database Connections

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

1. Download W7_Python.zip to your Desktop from Canvas
2. Right click the zip file > Extract Here. This creates a folder that includes W7_Python in its name.
3. Right click on the file PythonClient.py and click on edit with PythonWin (or any other available editor that can edit Python). This is the python code that interacts with the database, and which you'll be extending in this tutorial.
    a. Replace the word "unikey" with your UNIKEY in the variable "userid" eg: if you unikey is abcd1234 then the value of userid should be set to "y25s1c9120_abcd1234". Replace the word "password" with your PostgreSQL password.

```
# Connection parameters – Enter your login and Password here
myHost = "awsprddbs4836.shared.sydney.edu.au"
userid = "y25s1c9120_unikey"
passwd = "password"
```

    b. Be sure to **save** source code files after making any changes.
4. Press Windows + R on your keyboard, type in cmd in the textbox that pops up, then click OK.
5. cd into the directory where your PythonClient.py file is located.
6. Type in the following at the command prompt to install psycopg2:

```
pip install psycopg2-binary
```

7. Type in the following at the command prompt to run the python code:
   `python PythonClient.py`

If everything works fine, you should see the message "You are successfully connected to the database." 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 PythonClient class.

```
You are successfully connected to the database.
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 skeleton program with some life. You shall extend it with a method which uses the DB-API classes to dynamically query your database.

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

b) Extend the existing listUnits method 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.

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 def listTranscript(self, studentID) that performs the same query as you just wrote and uses cursor 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
TABLE(puosCode CHAR(8), puosName VARCHAR(40), pcredits INTEGER, psemester
CHAR(2), pyear INTEGER, pgrade VARCHAR(2)) AS $$
BEGIN
RETURN QUERY
        SELECT uosCode, uosName, credits, semester, year, grade
        FROM Transcript JOIN UnitOfStudy USING(uosCode)
        WHERE studId=studentID
```

```
        ORDER BY uosCode,year,semester;
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 the callproc method of cursor.