Department of Computer Science and Engineering

Information Systems and Databases

Lab 09 : Application Development and Transactions

At this point, you should already have the Bank's sample database created in your account on the **db.tecnico.ulisboa.pt** server. If you need to create the Bank database, you can do so following the instructions in Lab 1.

Part I – Simple Query and expansion into HTML

Each Python script you run will require your credentials to access the database. Instead of having to add them to every CGI script file you can keep them in a single file and import them into every other file.

The following code is a Python script that stores your credentials.

```
#!/usr/bin/python3

IST_ID = 'ist1XXXXX'
host = 'db.tecnico.ulisboa.pt'
port = 5432
password = 'XXXXXXXX'
db_name = IST_ID

credentials = 'host={} port={} user={} password={} dbname={}'.format(host, port, IST_ID,
password, db_name)
```

1. Save this code as a file named **login.py** in your local machine.

2. Replace the values for the variables **IST_ID** and **password** with your own login credentials on Postgres (username of Fénix, and the password from **psql_reset**)

3. Locate the folder "~/web" in you home on **sigma.tecnico.ulisboa.pt**. The "~/web" folder is where you should place your Python scripts.

4. Move or copy **login.py** into your "~/web" folder on **sigma.tecnico.ulisboa.pt.**

5. In the command line in Sigma, go to the web/ folder and run the command:

   ```
   chmod 755 login.py
   ```

   to give it execution permissions.

Make sure that your Python CGI scripts that connect to the database now import the **login.py** and create the connection from the credentials as follows:

```
import login
...
connection = psycopg2.connect(login.credentials)
...
```

The following code is a Python script that connects to the database and executes an SQL query to get all accounts and then presents them in an HTML table.

```
#!/usr/bin/python3
import psycopg2

import login

print('Content-type:text/html\n\n')
print('<html>')
print('<head>')
print('<title>Lab 09</title>')
print('</head>')
print('<body>')

connection = None
try:
        # Creating connection
        connection = psycopg2.connect(login.credentials)
        print('<p>Connected to Postgres with: {}.</p>'.format(login.credentials[:51]))
        cursor = connection.cursor()

        # Making query
        sql = 'SELECT * FROM account;'
        print('<p>{}</p>'.format(sql))
        cursor.execute(sql)
        result = cursor.fetchall()
        num = len(result)

        # Displaying results
        print('<p>{} records retrieved:</p>'.format(num))
        print('<table border="5">')
        print('<tr><td>account_number</td><td>branch_name</td><td>balance</td></tr>')
        for row in result:
                print('<tr>')
                for value in row:
                # The string has the {}, the variables inside format() will replace the {}
                        print('<td>{}</td>'.format(value))
                print('</tr>')
        print('</table>')

        #Closing connection
        cursor.close()

        print('<p>Connection closed.</p>')
        print('<p>Test completed successfully.</p>')
except Exception as e:
        print('<h1>An error occurred.</h1>')
        print('<p>{}</p>'.format(e))
finally:
        if connection is not None:
                connection.close()
```

6. Save this code as file named **test.cgi** in your local machine.

7. Move or copy the **test.cgi** into your "~/web" folder on **sigma.tecnico.ulisboa.pt**.

8. In the command line in Sigma, go to the web/ folder and run the command:

   ```
   chmod 755 *.cgi
   ```

   to give all cgi files execution permissions.

9. Open the web browser on your local machine and access the address::

   http://web2.tecnico.ulisboa.pt/ist1xxxxx/test.cgi

   where **ist1xxxxx** should be replaced by the name of the user.

10. An HTML page with all the accounts should appear

11. Right-click on the browser window and access the HTML code of the page you are viewing (select "View Source" or "View page source"). Compare this HTML with the contents of the **test.cgi** script. What are the differences?

Parte II – Updating the Database

Let us now create a page that modifies the balance of an account. For this purpose, we will use the code shown in the figure below.

1. Save the **accounts.cgi** file on your local machine.

2. Move or copy the **accounts.cgi** file to your "~/web" folder on the **sigma.tecnico.ulisboa.pt**, server.

3. Give the file execution permissions

   ```
   chmod 755 *.cgi
   ```

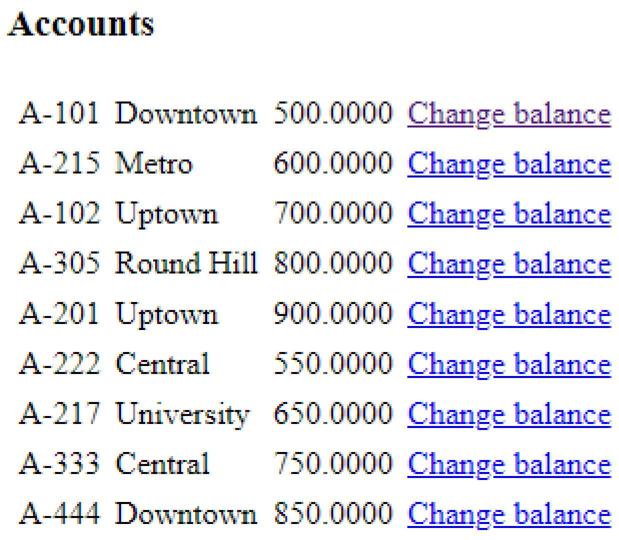
4. Using your browser, access the page:

   http://web2.tecnico.ulisboa.pt/ist1xxxxx/accounts.cgi

Code for **accounts.cgi**

```
#!/usr/bin/python3
import psycopg2

import login

print('Content-type:text/html\n\n')
print('<html>')
print('<head>')
print('<title>Lab 09</title>')
print('</head>')
print('<body>')
print('<h3>Accounts</h3>')

connection = None
try:
        # Creating connection
        connection = psycopg2.connect(login.credentials)
        cursor = connection.cursor()

        # Making query
        sql = 'SELECT account_number, branch_name, balance FROM account;'
        cursor.execute(sql)
        result = cursor.fetchall()
        num = len(result)

        # Displaying results
        print('<table border="0" cellspacing="5">')
        for row in result:
                print('<tr>')
                for value in row:
                # The string has the {}, the variables inside format() will replace
the {}
                        print('<td>{}</td>'.format(value))
                print('<td><a href="balance.cgi?account_number={}">Change
balance</a></td>'.format(row[0]))
                print('</tr>')
        print('</table>')

        # Closing connection
        cursor.close()
except Exception as e:
        # Print errors on the webpage if they occur
        print('<h1>An error occurred.</h1>')
        print('<p>{}</p>'.format(e))
finally:
        if connection is not None:
                connection.close()
```

5. A list of accounts should appear in the browser (with a "Change balance" link for each one.)



Accounts

A-101	Downtown	500.0000	Change balance
A-215	Metro	600.0000	Change balance
A-102	Uptown	700.0000	Change balance
A-305	Round Hill	800.0000	Change balance
A-201	Uptown	900.0000	Change balance
A-222	Central	550.0000	Change balance
A-217	University	650.0000	Change balance
A-333	Central	750.0000	Change balance
A-444	Downtown	850.0000	Change balance

6. Right-click the browser window, access the HTML code of the page. Compare it with the contents of the **accounts.cgi** script. In particular, notice how the "Change balance" links are constructed.

The following Python script will create a form to submit the new balance for a given account.

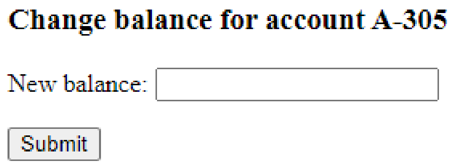
```
#!/usr/bin/python3
import cgi

form = cgi.FieldStorage()

account_number = form.getvalue('account_number')

print('Content-type:text/html\n\n')
print('<html>')
print('<head>')
print('<title>Lab 09</title>')
print('</head>')
print('<body>')

# The string has the {}, the variables inside format() will replace the {}
print('<h3>Change balance for account {}</h3>'.format(account_number))

# The form will send the info needed for the SQL query
print('<form action="update.cgi" method="post">')
print('<p><input type="hidden" name="account_number"
value="{}"/></p>'.format(account_number))
print('<p>New balance: <input type="text" name="balance"/></p>')
print('<p><input type="submit" value="Submit"/></p>')
print('</form>')

print('</body>')
print('</html>')
```

7. Notice how the code **account_number = form.getvalue('account_number')** is being used to get the account number value. Where does this account number come from?

8. Save the code above to a **balance.cgi** file on your local machine. Move or copy the **balance.cgi** file to your "~/web" folder on **sigma.tecnico.ulisboa.pt** and return to the page:

   http://web2.tecnico.ulisboa.pt/ist1xxxxx/accounts.cgi

9. Click on the "Change balance" link on the right side of the account **A-305**. The following input form should appear:


Change balance for account A-305

New balance:

Let us now create a PHP script that receives data from this form and updates the account balance. For this purpose, we will use the following code:

```
#!/usr/bin/python3
import psycopg2, cgi

import login

form = cgi.FieldStorage()
#getvalue uses the names from the form in previous page
account_number = form.getvalue('account_number')
balance = form.getvalue('balance')

print('Content-type:text/html\n\n')
print('<html>')
print('<head>')
print('<title>Lab 09</title>')
print('</head>')
print('<body>')

connection = None
try:
        # Creating connection
        connection = psycopg2.connect(login.credentials)
        cursor = connection.cursor()

        # Making query
        sql = 'UPDATE account SET balance = %s WHERE account_number = %s;'
        data = (balance, account_number)
        # The string has the {}, the variables inside format() will replace the {}
        print('<p>{}</p>'.format(sql % data))
        # Feed the data to the SQL query as follows to avoid SQL injection
        cursor.execute(sql, data)

        # Commit the update (without this step the database will not change)
        connection.commit()

        # Closing connection
        cursor.close()
except Exception as e:
        # Print errors on the webpage if they occur
        print('<h1>An error occurred.</h1>')
        print('<p>{}</p>'.format(e))
finally:
```

10. Save the code above on a file named **update.cgi** on your local machine. Move or copy the **update.cgi** file to your "~/web" folder on the **sigma.tecnico.ulisboa.pt**, server. Give the file execution permissions (chmod 755 *.cgi), and using your browser, access the page:

http://web2.tecnico.ulisboa.pt/ist1xxxxx/accounts.cgi

11. Click on the link "Change balance" for account **A-305**. The form should appear.

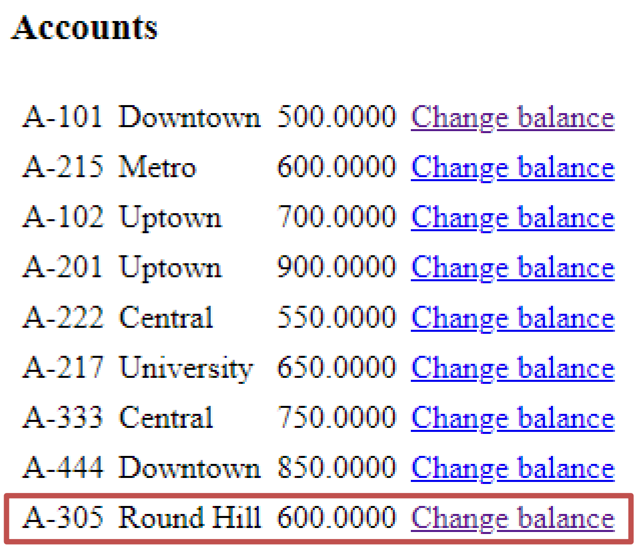
12. Complete the form, indicating the new balance for account **A-305** (for example: 600).

13. Press the “submit” button. The UPDATE statement that was executed on the database should appear in the browser.

14. You can now verify that the balance of the account A-305 has been updated. Access the address:

http://web2.tecnico.ulisboa.pt/ist1xxxxx/test.cgi

15. An HTML table with all accounts should appear in the browser.



Accounts

A-101	Downtown	500.0000	Change balance
A-215	Metro	600.0000	Change balance
A-102	Uptown	700.0000	Change balance
A-201	Uptown	900.0000	Change balance
A-222	Central	550.0000	Change balance
A-217	University	650.0000	Change balance
A-333	Central	750.0000	Change balance
A-444	Downtown	850.0000	Change balance
A-305	Round Hill	600.0000	Change balance

16. Confirm that the A-305 account has the new balance you entered on the form.

Part III – Concurrent Updates and Transactions

1. Open a Command Line Interface console (**pqsl**) for the Postgres database system and use the “Bank” database.

2. Write a query to display the data of the accounts of the customer Cook

3. The Cook customer now wants to transfer **500€** from his account **A-102** into the **A-101** account. Start a transaction with the **START TRANSACTION** command;

4. Write and execute the statement that puts **500€** in the account **A-101**.

5. Check the account balances of the customer Cook. At this point how much money does the Cook customer have in the bank?

6. Keeping the transaction open, open a second Command Line Interface terminal and connect to the same database.

7. On the new terminal, run again the query that returns the balances the accounts of the customer Cook. How much money does the Cook customer have in the bank after all?

8. On the first command line, where the transaction is running, write and execute an UPDATE SQL statement that withdraws **500€** from the account A-102.

9. On the second terminal, run the query that checks the account balances of the customer Cook once again.

10. On the first command line terminal, confirm the transaction (execute a **COMMIT** statement)

11. On the second command line terminal, check Cook customer account balances again. Confirm that the results of the transaction are now visible.

Part IV – Implementing Transactions in Python

1. Double check your Python scripts to see how they work with the transaction mechanism.

2. The customer *Cook* has just refilled **25€** of fuel at a service station and will pay by ATM (account **A-101**). At the same time, his wife, who has an ATM card for the same account, will withdraw **50€** from an ATM machine on the other side of town.

3. Open a command line interface for the database and start a transaction (issue a **START TRANSACTION** command)

4. Open a browser window and access the **accounts.cgi** script created before:

   http://web2.tecnico.ulisboa.pt/ist1xxxxx/accounts.cgi

5. In the first transaction, withdraw **25€** from the account (use an Cook statement).

6. In the browser, withdraw **50€** from the account. What happens when you try to do this? Why?

7. Suppose that, at the gas station, the power just went down, and the connection is dropped. The payment that was being made will be rolled back by the database due to a connection timeout. To simulate this behaviour, cancel this transaction explicitly by executing the **ROLLBACK** command.

8. At the same time as point 7, note what happens in the browser. How do you explain this phenomenon?

9. Check the account balances of the customer Cook once again. What do you observe?