CS032 2008 Lab 07
Database Lab

1 Introduction

In this lab you are going to learn how to use a database directly and integrated with applications written in Java and PHP. You are going to connect and manipulate data in a PostgreSQL database that we have set up for you.

This lab consists of two distinct parts. First, you are going to connect directly to the database and execute SQL queries in your shell. Afterwards, you are going to write a simple application with front end that allows the user to enter search criteria, and back end that connects to the database to execute search queries and passes the results to the front end. For the second part we provide you with two versions of the front end - a Swing-based GUI and an HTML web page. You can choose whether to use Java or PHP to implement the back end of a desktop or web-based application.

2 Part I: Direct Interaction with a Database

To connect to the database type in your shell:

```
psql -h cs32 -U cs32student -d cs32lab
```

```
-h stands for the host name;
-U stands for username
-d stand for the database name.
```

You are going to be prompted for the password: Password: csci320.

Create table:

```
CREATE TABLE students (student_id integer,
                       name       text,
                       box_n      integer,
                       year       integer);
```

You have to ensure that the data entered in the table is valid. For example, the student id column should contain unique id number for each student. To set this constraint on the student id column:

```
ALTER TABLE students ADD CONSTRAINT constr1 UNIQUE (student_id);
```

You could also define the constraint when you created the table by adding the word 'unique' after the type of the column:

```
CREATE TABLE students (student_id integer unique,
                       name       text,
                       box_n      integer,
                       year       integer);
```

To make sure you created the table correctly and see its definition type in:

```
\d students
```

To insert data in the table:

```
INSERT INTO students VALUES (1, 'jsmith', 1000, 2008);
```

To view the data:

```
SELECT * from students;
```

Repeat the above INSERT statement to insert information for more students. For example:

```
INSERT INTO students VALUES (2, 'aronald', 1020, 2010);
INSERT INTO students VALUES (3, 'dfred', 1030, 2009);
INSERT INTO students VALUES (2, 'fmillur', 1010, 2008);
INSERT INTO students VALUES (4, 'jsmith', 1015, 2008);
```

CHECKPOINT 1

Notice that it didn't allow you to insert two entries with the same student ID number.

Say for example you did a mistake when you inserted dfred's box number and you want to correct it. To update his information execute the following statement:

```
UPDATE students SET box_n = 1010 WHERE student_id = 3;
```

To view the change:

```
SELECT box_n from students WHERE student_id = 3;
```

To search for all students with name jsmith, graduating in 2008:

```
SELECT * from students WHERE year = 2008 AND name='jsmith';
```

CHECKPOINT 2: Show to a TA that you can execute simple SELECT and UPDATE SQL queries.

To delete the table:

```
DROP TABLE students;
```

3 Part II: Database Interaction in Java

For this part of the lab we provide you with a Swing-based GUI and stencil code in Java that you will have to complete to set up the back end of the application that communicates with the database. For this purpose Java provides the java.sql library. In particular you are going to use java.sql.Connection, java.sql.Statement, java.sql.DriverManager and java.sql.ResultSet. You can look at the javadocs for detailed description of these classes. You are also going to write simple SQL queries that you are going to embed and execute in the Java code. For further help with the SQL queries you can look at the PostgreSQL tutorial at http://www.postgresql.org/docs/8.3/static/queries.html.

You can copy the support code from /course/cs032/pub/lab07. To set up the project in Eclipse you have to import two external jar files: /course/cs032/lib/swing-layout.jar and /course/cs032/lib/postgresql.jar.

The methods that you will have to fill in are in the DBLabFormHandler.java class. It contains methods that handle openning and closing connection to the database, creating empty table, inserting data into the table and searching. These methods are called within the actionPerformed() methods associated with each button of the GUI. To give you a better idea how the GUI interacts with the back end, here's a small part of the DBLabGUI implementation:

```
private DBLabFormHandler my_handler;
// ...
jButton2.setText("Add");

jButton2.addActionListener(new java.awt.event.ActionListener() {
// This method is called when the Add button, called jButton2, is pressed.
   public void actionPerformed(java.awt.event.ActionEvent evt) {
       String name, student_id, box_n;
       Object year;

       // Get the entered values from the GUI text field and combo box.
       name = jTextField2.getText();
       student_id = jTextField1.getText();
       box_n = jTextField3.getText();
       year = jComboBox1.getSelectedItem();

       // Call the insert method of the handler, passing it the data.
       my_handler.handleAddStudentInfo(name, student_id, box_n, year);
   }
});
```

4 How to connect to PostgreSQL Database

To connect to a database within your java code you need an URL, username and password. These information has already been set up for you in the DBLabFormHandler class. All you need to do to establish the connection is to write the following lines of code in the initDB() method within a try/catch block:

```
try {
        Class.forName("org.postgresql.Driver");
} catch (ClassNotFoundException e) {
        e.printStackTrace();
}
// The url, username and pass are given to you.
Connection connection = DriverManager.getConnection(url, username, pass);
Statement stmt = connection.createStatement();
```

Go ahead and complete the initDB() method. **CHECKPOINT 3**

5 How to execute SQL queries in Java

To execute an update to the database, i.e. SQL statement that does not return result set, simply call executeQuery() to the instance of class Statement, passing it a String for the query. For example:

```
private Statement stmt;
// ...
String sql = "CREATE TABLE students VALUES (...);";
stmt.executeUpdate(sql);
```

To execute SELECT query and get the result data:

```
ResultSet rs;
String sql = "SELECT * from students;";
rs = stmt.executeQuery(sql);
```

```
// To print the results:
while ( rs.next() ) {
       int cur_id = rs.getInt("student_id");
       String cur_name = rs.getString("name");
       int cur_boxn = rs.getInt("box_n");
       int cur_year = rs.getInt("year");

       System.out.println("cur_name = " + cur_name);
       System.out.println("cur_student id = " + cur_id);
       System.out.println("cur_box number = " + cur_boxn);
       System.out.println("cur_year = " + cur_year);
}
rs.close();
```

Don't forget to call the close() method of the ResultSet once you are done.

To display the results in the GUI call the setModel() method of the JTable (created in the DBLabGUI class), which is going to hold the results. You have to pass the method a Vector that holds the data and a Vector with the column names. Here's how you can do that:

```
Vector<Vector> data = new Vector<Vector>(); // A 2D vector that holds vectors, representing the rows
Vector<String> columnNames = new Vector<String>(); // A vector with the column names
// ...
jtable.setModel(new javax.swing.table.DefaultTableModel(data, columnNames));
```

Go ahead and complete the handleAddStudentInfo() method. **CHECKPOINT 4**

Go ahead and complete the handleSearchStudentInfo() method. **CHECKPOINT 5**

6 How to Close the Database Connection

Simply call the close() method to the instances of Connection and Statement within a try/catch block:

```
Connection conn;
Statement stmt;
// ...
stmt.close();
conn.close();
```

Go ahead and complete the closeDB() method. **CHECKPOINT 6**

7 Part III: SQL interaction with PHP

WARNING: This is NOT a PHP tutorial. If you have no idea what PHP or HTML is, we strongly suggest you do the Java part of the lab. This part will assume basic knowledge and interaction using PHP.

In this section you will learn how to interact with a database using a simple php script and php-html forms. The process is quite simple - the queries that you learned about in Part I will be executed in your script. To get started create a directory in /web/dweb-devel/html/cs32lab/ with your username. In that directory, create a file called cs32lab.php. Open that file for editing.

7.1 The Header

This is the standard HTML header.

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />

<title>Database Lab - CS032</title>

</head>
```

To start your php script, after the header, type in

```
<?php
```

7.2 Connecting to a database

Connecting to a database in PHP is simple. For this lab, the database set up is PostgreSQL, so you will need specific syntax for that library. To connect to the database, at the beginning of your php script you should have the following line:

```
 pg_connect('host=cs32 dbname=cs32lab user=cs32student password=csci320')
 or die("Could not connect to server");
```

Now, close the PHP section by typing `?>`, and close the HTML header by typing `</html>`. Open up your PHP script in a web browser. It should be in dweb-devel.cs.brown.edu/cs32lab/¡username¿/cs32lab.php. The script should show nothing at all, most importantly it shouldn't say "Couldn not connect". If so, you've reached **CHECKPOINT 7**

7.3 Forms

You will need to write several forms in your script. A form is similar to a function call. Each of the inputs is a paramater and the function that's called is the action. This is an example of a simple form echoed out in PHP.

```
echo "
<form method='post' action='cs32lab.php'>

<input type='hidden' name='task' value='begin'>

<input type='submit' value='Go'>

</form> ";
```

This form will load the script cs32lab.php with a parameter of 'task' set to 'begin'. To retrieve this parameter in your php script you can do the following:

```
 $param1 = $_POST['task'];
```

This will put the value from 'task' into the new variable param1. You can use this to check different task values. For example `if ($param1 == 'begin') {}...` will check the value of param1. You will have to write several of these forms throughout your php script.

7.4 Queries in PHP

Queries in PHP are the same as the command line queries you executed in part I. Only difference is, the actual query will be stored in a string variable, and then executed using a specific php function. Here's an example of how to execute a query:

```
$query = "SELECT * FROM students";
$result = pg_query($query) or die("Something bad happened, I'm crying");
```

Retrieving results from queries is also simple. The resulting table is returned as a series of associative arrays. Here's how to go about retrieving data from your query. Assuming the query has already been executed, and its result stored in `$result`...

```
while ($row = pg_fetch_assoc($result))
{
$S_ID = $row['student_id'];
echo $S_ID."<br/>";
}
```

This will echo (or display) all the fields associated with student_id in each row in your students table.

7.5 Writing cs32lab.php

Once again, this part assumes some familiarity with HTML and PHP. If you're not familiar with it, but you still want to do this, expect to spend more than 90 minutes on this lab. You are to write several forms, that will call the same function (link to cs32lab.php), with different variables. When everything is written, this is the database-oriented functionality expected from your script:

- Some kind of a simple search through your database, with ability to display the results
- Ability to remove results that have been searched for
- Ability to add a new row to your table

Once you're done with this, you've reached **CHECKPOINT 8**.

8 Checkpoints

1. 10 points TA Initials: _____
   - Table with data created successfully.

2. 10 points TA Initials: _____
   - Student can execute SELECT and UPDATE queries.

Java Implementation

3. 10 points TA Initials: _____
   - Connecting successfully to the database.

4. 30 points TA Initials: _____
   - handleAddStudentInfo() inserts successfully data.

5. 30 points TA Initials: _____
   - handleSearchStudentInfo() works.

6. 10 points TA Initials: _____
   - Connection closed successfully.

PHP Implementation

7. 10 points TA Initials: _____
   - Connecting successfully to the database.

8. 70 points TA Initials: _____
   - Everything works!.