COMS W4111 - Introduction to Databases, Fall 2011
Project 1, Part 3 (Final Part)

(Worth 50% of overall Project 1 grade)

Your Team

You will do Part 3 (the final part) of Project 1 with the same team as for Parts 1 and 2. If your team partner dropped the class and you did **not** submit a contingency plan with your Part 1 submission, then, unfortunately, you will still have to complete the project yourself. If your team partner dropped the class and you did submit a contingency plan with your Part 1 submission, then you are welcome to switch to this reduced version of your project.

What you need to do for Part 3 of Project 1

As you recall from Part 1, you had two options for Part 3 of the project: you could either follow the **Web Front-End Option** or the **Expanded-Design Option**. If you stated in Part 1 that you would follow the Expanded-Design Option, it is OK for you to change your mind now and follow the Web Front-End Option for Part 3; however, if you did **not** submit a plan for the Expanded-Design Option when you submitted Part 1 of the project, then you do **not** have a choice now and you will have to follow the Web Front-End Option.

Important notes:

- You **cannot use grace days for the Web Front-End Option**; you can use grace days as usual for the Expanded-Design Option.
- You should make sure that you loaded sufficient data into your database to show off all functionality of your application.
- You can make (hopefully small) changes to the SQL schema that you created in Part 2. If for some strange reason you feel the need to make any radical changes, please check with your project mentor ahead of time to avoid any last-minute surprises.
- Please recall that the **Oracle servers** that you should use are **w4111b.cs.columbia.edu** and **w4111c.cs.columbia.edu.** Note that the **web servers** for this project are running on different machines, namely, **w4111a.cs.columbia.edu** and **w4111d.cs.columbia.edu** (see below).

Web Front-End Option

If you are following the Web Front-End option, you will finish building the application that you proposed in Part 1, on top of the database that you created in Part 2. For the final evaluation of Project 1, both team members will meet with your project mentor together on **Thursday November 17** or **Friday November 18**. Your project mentor will contact you shortly to schedule a 15-minute meeting for one of those days.

Your application can be written in Java or PHP (your choice), and should have a web interface. Specifically, you have a choice of two technologies for your project:

- **Option 1**: Java servlets, using Java only and JDBC for database connectivity.
- **Option 2**: PHP, using PHP only.

The web front-end for your application can be HTML forms in combination with Java servlets or PHP (pick one). The less your web site looks like it is interacting with a relational database, the better. At the very least, the user should be completely shielded from having to type anything resembling SQL.

Important notes:

- *Unix option:* If you are doing your project on Unix, your project should be **developed and run on a CS account**. If you decide to develop your application using Java, you may run the Resin application server (see below) on any **clic.cs.columbia.edu machine**, but you should remember **to shut down the Resin server before logging out.** Instead, if you decide to develop your application using PHP, please **use the class web servers** (see below) for development and testing, **not** the http://www.cs.columbia.edu or http://www1.cs.columbia.edu web servers.
- *Windows option:* If you are doing your project on Windows, please read the instructions below to install Apache Tomcat, if you are using Java, or PHP and Apache, if you are using PHP, on your Windows computer.
- *Mac option*: If you are doing your project on a Mac, please ignore the rest of the instructions and read the Instructions for Java & JDBC on Mac.

Option 1a: Java and JDBC, on Unix

If you choose Option 1 on Unix for your project, you should implement your application using Java servlets. Your implementation will consist of:

1. Java servlet and/or JSP,
2. Servlet engine (Resin), and
3. Oracle backend (your database)

The application you write will use Java servlets, which receive web page requests, interact with the database server by using JDBC, and return HTML. A very friendly short course on Java database programming using JDBC is at http://developer.java.sun.com/developer/onlineTraining/Database/JDBCShortCourse/contents.html. A JDBC database access tutorial is at http://java.sun.com/docs/books/tutorial/jdbc/basics/gettingstarted.html.

You may use Java as your language for CGI scripts. This option is permissible, but will result in extremely slow response times, since the web server will need to load a Java VM for every invocation of your Java program. Instead, follow the directions below to install Resin, a second-tier server. **Note:** Using Java directly for CGI is an unsupported alternative, so we discourage you from following it.

The following are directions for installing your Java second-tier server, **Resin**, assuming that you use bash as shell (if you use other shells, please use an appropriately modified version of this; please contact your project mentor if you have questions):

1. **Install Resin:** Download Resin at http://www.caucho.com/download/resin-3.0.23.tar.gz, save it in your home directory, and then extract/untar the file, as follows.

   ~$tar zxvf resin-3.0.23.tar.gz
   ~$cd resin-3.0.23
   ~/resin-3.0.23$./configure
   ~/resin-3.0.23$make
   ~/resin-3.0.23$make install

2. **Configure Resin:** Add the following lines to your **.bashrc** or **.profile** file, or equivalent:

   export RESIN_HOME=<directory where you installed resin>
   export CLASSPATH=$RESIN_HOME/lib/resin.jar:$RESIN_HOME/lib/servlet.jar:$CLASSPATH

export JAVA_HOME=/usr/lib/jvm/java-6-sun
~$cd $RESIN_HOME/conf

Note: If the **servlet.jar** file is missing from the $RESIN_HOME/lib/ directory, please download this file from here.

Edit file **resin.conf** to change the value of port from 8080 to the port on which your host (server) will accept incoming connections (http requests). **Your team's port number will be 5090 + (team-number)*2**. To find out your team number, check the CourseWorks grade page for the class.

You should also remove or mask the following line (what follows is a masking example):

<cluster> <!-- < srun id="" host="127.0.0.1" port="6802"/ > -- > < /cluster >.

3. **Start the server:**

   ~$cd resin-3.0.23/bin
   ~/resin-3.0.23/bin$sh httpd.sh &

4. **Create servlets:** You should put your servlets in the resin-3.0.23/webapps/ROOT/WEB-INF/classes directory. Then, you will be able to access your servlets at http://localhost:portnumber/Javafile.

   For example, you can create a Java file as resin-3.0.23/webapps/ROOT/WEB-INF/classes /HelloServlet.java, and then access it as http://helsinki.clic.cs.columbia.edu:7791/HelloServlet, where "helsinki.clic.cs.columbia.edu" is the machine where your Resin server is running and 7791 is the port number where your Resin server is listening.

5. **Stop the server:** Do ps -U <account-name> and kill the first (i.e., lowest-numbered) process that says JAVA next to it.

   ~$ps –U <account-name>
   ~$kill <process ID> (the lowest-numbered process that says JAVA next to it)

Important note: You must stop your Resin server by following the last step above after you're done testing, etc. and before you logout from the clic.cs.columbia.edu machine where you were working.

Web.xml file and sample Java program

A **web.xml** file is required to provide configuration and deployment information for the web components that comprise a web application. Place the **web.xml** file in the directory resin-3.0.23/webapps /ROOT/WEB-INF/, where **resin-3.0.23** is the directory where you installed Resin.

A sample Java program that accesses one of our Oracle servers is at http://www.cs.columbia.edu/~biliris /cs4111/Proj1-3/sample.java. The program assumes that it is stored in the **resin-3.0.23/webapps /ROOT/WEB-INF/classes** directory and the compiled class is in the same directory, where **resin-3.0.23** is the directory where you installed Resin. The web.xml file for this sample Java file can be found at http://www.cs.columbia.edu/~biliris/cs4111/Proj1-3/web.xml.txt. Please rename the file to web.xml. An HTML file to invoke the above Java program should be placed in the directory **resin-3.0.23/webapps /ROOT/.** The HTML form action should resemble the following line:

<form name = input action = sample>

Option 1b: Java and JDBC, on Windows

If you choose Option 1 on Windows for your project, you should install Apache Tomcat on your Windows computer. To install and configure Apache Tomcat, please refer to our instructions.

Option 2a: PHP, on UNIX

If you choose Option 2 on Unix for your project, you should implement your application using PHP.

Important note 1: Always use the class web servers, **w4111a.cs.columbia.edu** and **w4111d.cs.columbia.edu** (rather than www.cs.columbia.edu or www1.cs.columbia.edu), for all your development and testing related to this project.

Your implementation of your application using PHP will consist of:

- PHP source code, which will provide all database connectivity, and
- Oracle backend (your database).

The application you write will use PHP, which is executed by the web server when a PHP page is requested by a browser. The PHP source code both connects to the backend database and outputs information (HTML) to the web server, which is in turn transmitted for display on the client.

For information on using PHP with Oracle, see the PHP OCI8 manual at http://us3.php.net/oci8. Documentation for PHP can be found at http://www.php.net/docs.php as well as through the numerous books and websites related to PHP.

Important note 2: If you pursue this option, please follow these steps from your CS account as soon as possible to check that your CS account is properly configured to work with our web servers:

- $ cd html
- $ cp ~ks2677/html/test_w4111.php ./
- From your web browser, type: http://w4111a.cs.columbia.edu/~(yourUNI)/test_w4111.php
- If the browser displays a message "User Name: SCOTT," then your CS account is properly configured. Otherwise, please email Pranjal (pg2354@columbia.edu) right away so that he can configure your account.

Please check our document with useful PHP tips.

Important Note on PHP Security

If you choose to use PHP, please read this section carefully and follow our guidelines to secure your database account.

The PHP source files in your html directory are executed by the web server when accessed by a client. This means that the HTML produced by your PHP files is sent to the client, not the PHP source code. This is a good thing, in terms of security, since some of your PHP files will need to include your database account information (i.e., id, password, db name).

Your password, however, is not totally secure. Anyone with a CS account can still access your files by

logging into a CS department machine and examining your directory. To help mitigate any security risks, please do the following:

1. Make sure that your Oracle password (which you should have changed) is not the same as your UNI or CS password, or the password for any other account.

2. Within your html directory, create a subdirectory <new_directory> for this project. While logged in and in your html directory, execute:
chmod 711 <new_directory>
This will prevent anyone other than you from obtaining a listing of this directory.

3. Create your project without using the filenames index.html or index.php. Use somewhat more cryptic, yet useful filenames (e.g., john_db_index.php instead of index.php). The server looks for these index.* files by default, which is nice, but the permissions change in Step 2 cannot prevent CS users from copying a file in your directory whose name they know. You can still test your project by typing the filename in your browser's address box.

Option 2b: PHP, on Windows

If you choose Option 2 on Windows for your project, you should install Apache and PHP, configured with the OCI8 extension, on your Windows computer. To install and configure Apache and PHP, please refer to our instructions. Please also check our document with useful PHP tips.

Expanded-Design Option

If you are following the Expanded-Design Option, you need to follow the expansion plans that you outlined in Part 1, and:

1. **Extend your E/R diagram** from Part 1 to include the entity sets and relationship sets—and all associated real-world constraints—for your expanded design.

2. **Extend your SQL schema** from Part 2 on one of our Oracle servers to include the mapping of all new entity sets and relationship sets, following the directions in Part 2 of the project on how to specify **constraints** in your expanded SQL design.

3. **Add tuples** to your new SQL relations on the Oracle server, following the directions in Part 2 of the project.

What to submit and when

Web Front-End Option

You do not need to submit anything for this (final) part of Project 1 if you are following the Web Front-End Option. Instead, students will meet with the project mentor on either **Thursday November 17** or **Friday November 18**.

Your project mentor will email you shortly to schedule a 15' meeting for either day. (If you haven't received an email from your project mentor by **Wednesday November 9**, please contact your mentor immediately after that day.) During the meeting with your project mentor, you will show your mentor your application using a regular web browser:

1. ***Unix option:*** You should have your application up and running so that you and your project

mentor can access it over the web from the CS department simply by typing a URL in a regular browser. Your project mentor will either be running Mozilla/Firefox over Linux or Windows, or Internet Explorer over Windows. (You may use JavaScript for some of the browser control, but make sure it is portable and works under Mozilla/Firefox over Linux and Windows, and Internet Explorer on Windows.) **Note:** The TA room only has Linux machines just as the clic.cs.columbia.edu machines.
Windows option: You should bring your laptop to your meeting with your mentor, to be able to run the application with your mentor.

2. Your project mentor should be able to interact with your application and access the functionality that you specified in Part 1, over the database that you created for Part 2.

3. The project mentor might ask to look at your code during your meeting. You do not need to submit anything to the project mentor ahead of time, but you might have to show your code during the meeting.

4. Your web interface does not need to be fancy. (See Grading below.) However, **you should not force users to type SQL**. The less your web site looks like it is interacting with a relational database, the better. At the very least, the user should be completely shielded from having to type anything resembling SQL. Most interactions should involve some sort of input values in addition to the user pressing a *Submit* button. Whenever possible, input values should be specified using menus, radio buttons, checkboxes, scrollers, etc. Text input boxes may also be appropriate.

5. Sophisticated error handling is not necessary; however your web site definitely should not "lock up" regardless of how the user chooses to interact with it.

6. Your database should contain (at least) the data that you entered for Part 2. You can, of course, add extra tuples to your tables if you want to make interaction with your application more interesting and revealing.

7. Your grade will suffer considerably if your application is not running properly when you access it from your project mentor's machine, if you are following the Unix option, or on your machine, if you are following the Windows option. It is your responsibility to ensure that your application is not dependent on a specific platform, if you follow the Unix option, and that it is up and running when you meet with your project mentor.

8. You should have a number of example interactions prepared so that you can use your meeting time efficiently. The more you can impress your project mentor during the 15-minute meeting, the better your grade is likely to be, so come to the meeting prepared.

Expanded-Design Option

If you follow the Expanded-Design Option, your submission deadline is **Thursday November 17, at 5 p.m. EDT**.

1. Create a directory named **<your-UNI>-proj1part3**, where you should replace **<your-UNI>** with the Columbia UNI of one teammate (for example, if the teammate's UNI is **abc123**, then the directory should be named **abc123-proj1part3**).

2. If you have your expanded E/R diagram in electronic form, please generate a PDF version of it and name the resulting file **expanded-er.pdf**. Place this file inside the **<your-UNI>-proj1part3** directory. If you cannot generate a PDF file with your E/R diagram, please submit a hard-copy version of the diagram to your project mentor before the submission deadline.

3. After extending your SQL schema for Part 2, to account for your expanded design (see above),

and populating the new SQL tables as required (see above), generate the dump file **expdat.dmp** using Oracle's export utility **exp** and place it inside the **<your-UNI>-proj1part3** directory. See Part 2 of Project 1 for instructions on how to use this utility.

4. Generate a **plain-text file** called **queries.txt** inside the **<your-UNI>-proj1part3** directory, with three "interesting" SQL queries over your expanded database. Include a sentence or two per query explaining what the query is supposed to compute; the goal is simply to help us better understand your application.

5. *(Unix option only)* **Tar** and **gzip** the **<your-UNI>-proj1part3** directory, to generate **a single file <your-UNI>-proj1part3.tar.gz** (containing the expanded-er.pdf file (unless you submitted a hard-copy of the expanded E/R diagram to your project mentor), as well as the expdat.dmp and queries.txt files), which is the file that you will submit.

6. *(Windows option only)* Zip the **<your-UNI>-proj1part3** directory, to generate **a single file <your-UNI>-proj1part3.zip** (containing the expanded-er.pdf file (unless you submitted a hard-copy of the expanded E/R diagram to your project mentor), as well as the expdat.dmp and queries.txt files), which is the file that you will submit.

7. Login to CourseWorks at https://courseworks.columbia.edu/ and select the site for our class.

8. Select "Class Files" and then "Post File."

9. Enter **"Project 1, Part 3"** in the "Title" field and select your **<your-UNI>-proj1part3.tar.gz** file (from Unix) or your **<your-UNI>-proj1part3.zip** file (from Windows) in the "File" field using the "Browse" button.

10. Select **folder "Project 1, Part 3" (under "Shared Folders")** in the "Post File To" field.

11. Enter the names of the members of the team and their UNI in the "Comments" field.

12. Hit the "Submit" button.

Grading for Part 3

Web Front-End Option: Your grade for Part 3 of Project 1 will be a function of how well your application (which should be up and running) matches your specification that you submitted as Part 1, of how well you have incorporated any feedback that your project mentor has given you, and of how well you have followed the guidelines above. Your project mentor has kept a copy of your specification and of his comments. Your grade will **not** be influenced by how fancy the web-based user interface to your application is. It is sufficient and perfectly fine for this interface to be plain and simple as long as it supports the functionality that you indicated earlier, following the guidelines above about not having to type SQL commands, not "locking up" on unexpected input, etc.

Expanded-Design Option: Your grade for Part 3 of Project 1 will be a function of how well you have incorporated any feedback that your project mentor has given you, and the following factors:

1. **Quality of your expanded E/R diagram:** We will evaluate how well your expanded E/R diagram implements your plans for the Expanded-Design Option from Part 1, and how well your expanded E/R diagram models your application, including how well you modeled any relevant real-world constraints.

2. **Quality of your expanded SQL schema and implementation on Oracle:** We will evaluate how well you mapped your expanded E/R diagram, including constraints, into a SQL schema on Oracle, using the techniques that we covered in class.

3. **Quality of your expanded constraint handling:** We will evaluate how well you managed to capture real-world constraints of your expanded design through primary key, foreign key, unique, and attribute- and tuple-based CHECK constraints.

4. **Quality of the expanded real-world (or at least realistic) data** that you loaded into the expanded database on Oracle.