Program #4: Database-driven Web Application

Due Dates:

<table>
<thead>
<tr>
<th>Category</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Members</td>
<td>November 20th, 2018, at the beginning of class</td>
</tr>
<tr>
<td>Draft E-R Diagram</td>
<td>November 27th, 2018, at the beginning of class</td>
</tr>
<tr>
<td>Final Product</td>
<td>December 4th, 2018, at the beginning of class</td>
</tr>
</tbody>
</table>

Designed by Terrence Lim and Bailey Nottingham

Overview: In this assignment, you will build a database-driven web information management system from ground up. We will give you an application domain to work on, your goal is to design the underlying database and define the application functionalities you will provide with the database, and implement this application as a web-based system.

Assignment: In this assignment you are to implement a three-tier client-server architecture.

1. **Database Back-End**, which runs the Oracle DBMS on aloe.cs.arizona.edu. Your job is to design the database relational schema, create tables and populate your tables with some initial data. We are requiring that you create an E-R diagram, analyze the FDs of each table and apply table normalization techniques to your schema to justify that your schema satisfies 3NF, and, if possible, BCNF.

2. **The business logic and data processing layer**, which is the middle tier that runs on an application server (lectura.cs.arizona.edu) running the Maven web server. We’ll be using the Spring framework for Java. Spring provides an MVC (Model–View–Controller) format to design this layer of the application. This layer helps you create a controller which will allow you to handle GET, POST, PUT and DELETE requests made by the client. The response generation may involve accessing, from within the controller, the back-end Oracle database you have created. The demo provided (see below) demonstrates how the HTML page and the controller would interact (Controller-View interaction).

3. **Web Front-End**, which is the client user-interface. You need to design web pages appropriately to handle all the required functionalities. Your client application can run on a CS lab workstation, or on any machine that can tunnel into lectura.

Application Domain: The problem description for the project is as follows:

You are tasked with creating an application for the Oklahoma University Community Hospital (OUCH).

Data stored for each patient includes: PID, last name, first name, gender, DOB, address, and contact number.

Each doctor has DID, last name, first name, DOB, status, department ID, office number. In reality, one doctor can belong to multiple departments depending on his/her duty, but each doctor belongs to only one department and holds one office in this hospital. Not all doctors are tenured doctor of the hospital that their status can be one if trainee, tenured, or visiting.

The hospital has its own internal pharmacy, so there are pharmacists with pharmacist ID, last name, first name, DOB, and office number. And, pharmacists maintain their own data table to keep records of subscribed medicines. This table holds pharmacists ID, patient ID, medicine name, and the number of days to take the medicine.

(Continued...)
There are also nurses who are governing patients’ hospitalized rooms. The stored data includes nurse ID, last name, first name, DOB, room number in charge. One nurse is assigned to work for only one room, but there can be multiple nurses per room. There are three types of rooms, which are single, double, and five people room. One nurse working for a single room, two for double, and three for five people room.

There are, of course, people who are running the hospital business, who are staffs of the hospital. (For the simplicity of the project) There are only two business departments, which are registration and finance. Each staff data holds employee ID, last name, first name, DOB salary, department ID, office number (assume there are only two offices one for each), job title, gender, and contact number. Receptionists at registration office keep records of patients’ appointment such as PID, appointment date, and appointment number with their EID. Cashiers keeps track of PID, amount due, due date, payment status, payment date, also, with their EID.

OUCH is quite a large hospital with three different buildings for different purposes. Building L holds doctor’s offices, pharmacy, medical departments, and emergency room. Building I holds registration office and finance office. Building M holds rooms only for hospitalization. The data for buildings only holds building name and address.

The room number increments by 1 and there are 300 rooms in total of three buildings. A building L has 170 rooms, building I has 2 rooms, and building M has 128 rooms. (Read above carefully for the constraints of the rooms). Again, one office per doctor, there are single, double, five people room, and 5 or more employees are working in each business office. The room table holds room number, building name, room type (doctor’s office/ registration office/finance office/hospitalization room/pharmacy/emergency room), and capacity.

The hospital also holds records of each patient’s treatment record, which includes: PID, appointment number, reason of visit, date of visit, initial hospitalized date, expected discharge date, discharged date, hospitalized room number, treatment method, and assigned doctor ID. Each patient is assigned with only one doctor, but one doctor can have a multiple patients. If a patient needs to be hospitalized, he/she will be assigned to one room. This means that there will be patients who leave on the same day of visit. In such case, date of visit, expected and actual discharged dates should be the same.

There are, also, department data with department id, name, building name, and department office number.

This description does not describe every detail. These are the essentials; we expect that your team will create logical and conceptual designs that incorporate these features, at minimum. You are free to add additional details that you feel are appropriate.

For each table you create, you need to populate a reasonable number of tuples in order to test your queries. Some data basics are provided in the application domain description; the rest are left for you to determine, based on your needs. (What is ‘reasonable’ is difficult to define; a few dozen tuples per relation certainly would be; just a handful per relation may not provide sufficient variety.)

We realize that you are not a hospital administrator, but you have dealt with various health care systems in your life. Hopefully, you have enough experience that this problem domain and description make sense. If you have questions, please ask, and the TAs will help you clear things up.
**Required functionalities:** Within the framework provided above, your system is expected to perform examples of the following operations:

1. **Record insertion:** Your application should support inserting a new data record via web interface.
2. **Record deletion:** Your application should support deleting an existing data record via web interface.
3. **Record update:** Your application should support updating an existing data record via web interface.
4. **Queries:** Your application should support querying your database via the web interface for the problem description given above. You are required to implement three provided queries as well as at least one query of your own design. Details are provided below.

Specifically, the web interface should enable users to:

1. Add, update, or delete a patient, doctor, nurse, pharmacist, and staff. When adding, the user cannot leave ID, first and last name, and DOB empty. However, other records can be left as empty and be updated later. When updating, the user is allowed to update everything except ID and DOB. When deleting, the entire row needs be deleted.
2. Add or update patient treatment records. When adding PID, appointment number, reason of visit, and date of visit cannot be left as blank. When updating, everything can be updated except the PID, appointment number, reason of visit, and the date of visit.
3. Add or update cashier records, such as the status and payment date. When adding, only the payment date can be left as blank. The update must not change the PID where as other records should be allowed to modify including the EID.
4. Update department room number and building name from the department table. In fact, anything in this table can be modified and updated except the DID.

Here are the queries that your application should be able to answer:

1. When patients visit the hospital, they all must first check-in with the receptionist. Then, the receptionist searches the patient from the patient record with their first name, last name, and DOB. Write a query that lookup and print the found patient record. The query result should display PID, Pname (first name + last name), gender, DOB, the most recent date of visit, the reason for the last visit, treatment method, and the assigned doctor during the previous visit.
2. Write a query that displays the result of search by user who searched the list of doctors based on the department. The user will input the department name, then the result should print Doctor’s name (first + last name), office number, and the building name.
3. List the patients who are currently hospitalized and expected to be hospitalized more than 5 days with an outstanding fees to be paid. Print the PID, Pname (first name + last name), number of expected hospitalized days, room number, and total amount due.
4. One additional non–trivial query of your own design, with these restrictions: There are three tables not being used by the previously asked queries: “Staff,” “Receptionist,” and “Pharmacist.” These tables are listed to make the project as realistic as possible (while keeping the scale in a smaller size) that you’ll be seeing these three tables nearly in any real-life hospital management systems. However, the reason for these tables have not been asked in the previous queries is because we want you to come up with your own creative query to create relationships between the tables that are not directly related to each other. The “Staff” and “Receptionist” tables are directly related to each other with a FK where as the “Pharmacist” table isn’t. Therefore, we would like you to construct your own query to create a relationship between the three tables. You are more than welcome to use as many tables as possible in order to make a relationship. And, this query should not be trivial (for example, simply selecting everything from a table); your queries should be constructed to answer questions that real users would be expected to ask.

(Continued...)
A Simple Demo Application: To speed up the development, we’ve provided a simple demo application. The demo contains a simple web page with a simple interface. Please read the short documentation file that accompanies it to see how to run it. We also demonstrated this demo the day the assignment was handed out.

The demo package (with documentation!) named program4-files.zip is in the class directory (/home/cs460/fall18/) on lectura. You should download, install, and run this demo yourself soon because (a) the Spring framework is probably new to you, and (b) running it will help you get familiar with the components and directory structure that you are going to use for this assignment.

Working in Groups: In industry, such a project is usually the work of multiple developers, because it involves several different components. Good communication is a vital key to the success of the project. This assignment provides such an opportunity for teamwork. Therefore, we are accepting team sizes of two, three, or four members. Because we want this class to include at least one team project, working alone is not permitted.

You will need to agree on a reasonable workload distribution plan. More importantly, you need to come up with a well-formed design at the beginning. This will minimize conflicts and debugging effort in the actual implementation.

Hand In: Here are the ‘deliverables’ for each of the assignment’s three due dates:

1. Team Composition: By the first due date (see the top of the front page of this handout), one member of your team must create a PRIVATE post on Piazza using the “program 4” tag with the names of the members of your team. Failure to do so by the start of class on this date will cost your team the points listed in the Grading Criteria section (below).

2. E–R Diagram: As stated above, in the Assignment section, your team will need to create an E–R diagram that describes your database design. Before the second due date, your team will need to prepare a draft of your E–R diagram and a member of your team will need to submit it through turnin. The purpose of this requirement is to allow the TAs to review your schema and make suggestions for improvement. The sooner you create your design and discuss it with the TAs, the more time you will have to refine your final E-R diagram. If TAs need further explanation of your E–R Diagram, they’ll send out an email to make an appointment to have an additional meeting.

3. Final Product: On or before the third due date, a member of your team must submit a .tar file of your well-documented application program file(s) via turnin to the folder cs460p4. The tar file should contain all of the following:
   (a) The source code for your application. The structure of it should follow that of the demo application (see below).
   (b) A subdirectory called “doc”, containing a PDF document including these sections in this order:
      i. Conceptual database design: Final ER diagram along with your design rationale and any necessary high-level text description of the data model (e.g., constraints or anything not able to show in the ER diagram but is necessary to help people understand your database design).
      ii. Logical database design: Converting an ER schema into a relational database schema. Show the schemas of the tables resulted in this step.
      iii. Normalization analysis: Show the FDs of all your tables and justify why your design adheres to 3NF.
      iv. Query description: Describe your created query. Specifically, what question is it answering? What is the significance of including such a query in the system?
   (c) A ReadMe.txt describing:
      i. How the class staff can operate your website to see the required functionalities
      ii. The workload distribution among team members (that is, which people were responsible for which parts of the project?).

In addition, each team should schedule a time slot (~15 minutes) to meet with a TA and demonstrate your system. Closer to the third due date, we will let you know how to sign up.
Grading Criteria: Total: 100 points

1. Team Composition (1st due date): 5

2. Complete E–R Diagram Draft (2nd due date): 20

3. Final Submission (3rd due date):
   (a) Coding / Implementation: 60
       • Documentation 10
       • Style and organization 10
       • Record insertion: 5
       • Record deletion: 5
       • Record update: 10
       • Record query: 10
       • web front-end: 10
   (b) Database design: 15
       • Final E–R diagram: 5
       • Normalization analysis: 10

Grading Notes:

1. Unless we receive verifiable complaints about inadequate contributions, each team member will receive the same score.

2. We won’t put much weight at all on the appearance of the pages; concern yourselves with the web page functionality instead. The main point of the assignment is the DB design.

Late days: Late days can be used on this assignment, but only on the third due date. How many a team has to use is determined as follows: Team members total their remaining late days, and divide by the number of members in the team (integer division), producing the number of late days the team has available, to a max of two days late. (Justification: The TAs need to get grading done soon after the due date, and you need time to study for your final exams.)

For example, a team whose three members have 1, 1, and 3 late days remaining have \( \frac{1+1+3}{3} = 1 \) late day to use to use, if needed.

5