Thomas Madden

CPSC 408: Database Management

Professor Rene German

18 May 2018

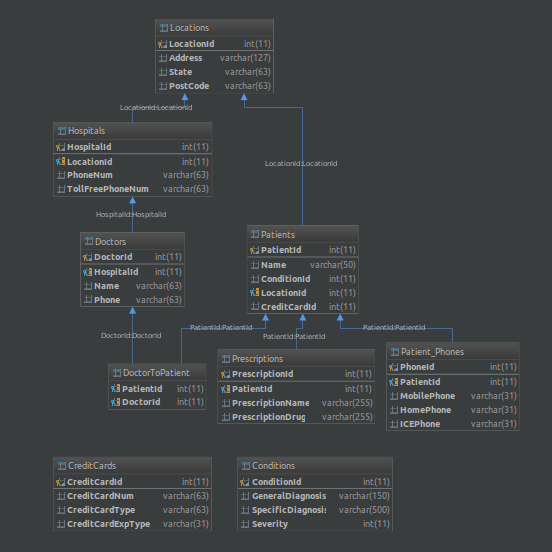
HospitalHelper

Currently, the database systems used by hospitals, while functional, tend to be disorganized and make it abnormally difficult to insert, update, and delete data, displaying it only in a tabular form which is unhelpful for doctors wishing to visualize their information and make decisions on that visualization. HospitalHelper, when developed to later stages with more diverse hospital information, would be designed to allow doctors to see a patient’s data in reference to the past, as well as in reference to other patients from all over the country. For this project, I have completed the beginning stages of this undertaking, using the same tabular scheme to build off of for the future. Currently, the project would be more helpful for hospital staff than doctors, because it is mostly demographic and clerical information, and for this reason it is my belief that HospitalHelper could help hospital staff be more organized in retaining patient information, especially in monitoring and managing prescriptions. Also, by looking at the demographics of a patient, a doctor is better able to understand the patient’s background.

Currently there are countless applications which accomplish the same goals as Hospital Helper 1.0, because just about every hospital needs to keep track of its data somehow, and that is usually done through a database system, but through the advancement of the product, HospitalHelper can be expanded to analyze medical data more completely rather than just displaying it. One example of this is at CHOC Medical Hospital, where there is a project going on to visualize patient data, leveraging artificial intelligence and nationwide data from CERNER Medical Group to better enable doctors to compare patients’ conditions with each other and make informed decisions. Eventually, this is the goal of HospitalHelper as well, but such goals are not accomplished in a couple months. To accomplish a more efficient and usable system, such applications must use some form of database system, either MySQL, SQL Server in general, or even NoSQL, because of the necessity of having organized data, which can be efficiently queried and updated. Also, this kind of solution would have extensive statistical analysis for doctors to view and manipulate. The impacts of such efforts are boundless, because they could potentially save thousands of lives if done correctly, not to mention saving hospitals in the U. S. an inordinate amount of money.

As for this particular program (HospitalHelper 1.0), I developed the front end/GUI using Java Swing and the back end using a MySQL Database. My Java Swing application connected to MySQL using JDBC, and utilized certain core functions from the DatabaseMetadata class as well, in order to check what tables and views existed in the database. The application was organized into Panels - a PatientPanel to organize Patient demographic information, a DoctorPanel to view doctor information, and a HospitalPanel to engage with hospital data. Each of these Panels have an ArrayList of “DataUpdates” which contains all recent updates which will be loaded to the server on pressing the “Save Changes” button. There are several other utility Panels, however - a Panel to instantiate the application (HomePanel), a Panel for inserting data, a Panel for grouping hospitals by State, a Panel for grouping Patients by State, and a Panel for the top bar visible at any point when using the application.

Those panels used for grouping data used the COUNT(\*) aggregate function to count the number of patients and hospitals in each state, and they also give the user the opportunity to filter by a certain state using a SELECT-FROM-WHERE statement on a View which contains the basic information for those tables. And finally, the Main and DataManager classes are utility classes used to create all the tables and generate fake data. Some of the data was generated using Faker, a library for data generation, and some other data specific to hospitals, like prescriptions, conditions, and diagnoses, were generated using an online tool, downloaded to a file called “specific\_data.csv”. The DataGenerator.php (which uses Faker) automatically loads data to a file called “demographic\_data.csv.” Also, when data is exported from any of the three main panels (it exports the main, important information which is contained in the Views), it automatically saves to one of three file names: “patient\_data\_[timestamp].csv”, “doctor\_data\_[timestamp].csv”, and “hospital\_data\_[timestamp].csv,” and it does so using the Apache Commons CSV library. (The timestamp is provided so that the user can export data multiple times and it will show when it was exported, rather than just overwriting the same file again and again).

 Here is my schema diagram, with keys.

As you can see from the schema diagram, the majority of the data is Patient information, such as mobile phone numbers, addresses (with city, state, and ZIP codes), prescriptions, and diagnoses. This data provides a more holistic view of the patient’s demographics and background, as well as that patient’s present condition. Statistically, the project also demonstrates the grouping of patients and hospitals by state, which could be useful for a hospital system to know if comparing patients with one another because of the differences in background and ethnicity that can come from being in two different states. Ideally, the data would be spread out over the entire U.S., but that is not necessarily the case, and the GROUP BY aids in doctors being able to adequately analyze data demographically (if not with empirical health data in this particular application). Also, users looking for a hospital near them could search for hospitals in their current state, and find relevant ones.

**Functionality:**

HospitalHelper has many different functions. It can be sorted by any of its columns by clicking on the column header, updated when double clicked and changed (but the database is only updated when the user clicks the “Save Changes” button!), deleted from with the “Delete Selected Row” button, and inserted into with the form provided on clicking the “Insert Data” button. The data inserted only actually updates the database if it satisfies integrity constraints - that is, the requirements previously placed on the database are not broken by the new data. Otherwise, an error message prints in the Standard Out Console. Also, the ID field is not editable by the user because of its uniqueness and the many ways a user could try to break that integrity constraint. On top of that, users can group by state to see how many patients or hospitals are in a certain state, and then filter to a certain state name by clicking “Filter To State” after that and typing in the full name of the state he/she wants to filter to. Under the Doctors tab, a user can filter to a given hospital to only see doctors from that Hospital. This would be crucial for a hospital’s staff team to organize their records and manage things, and seemed like a valuable feature to implement. As aforementioned, each of these three views can be exported to CSV files as well.

**Conclusion:**

Overall, HospitalHelper 1.0 is the beginning of a database system which could revolutionize the medical field by integrating medical data with demographic data, thereby allowing doctors to more adequately understand the patient’s condition and background. Also, with some improvements, it could be helpful to compare patients with similar backgrounds, ethnicities, and situations in general, to compare variables never examined before in medicine. Also, HospitalHelper could be improved to have data visualizations for blood pressure, overall condition, and pain level so that the doctor can glance at a picture of the condition and have a general understanding of where the patient is coming from. With a more organized, patient-centered approach, doctors have a much more grounded perspective when they give diagnoses/treatments, and decisions will be made on the basis of a more thorough understanding of that individual, rather than simply on medical knowledge.

**Demo Instructions:**

1. Make sure there exists a database “cpsc\_408” accessible on localhost port 3308
2. Ensure there is a folder called “assets” with the file “demographic\_data.csv” and “specific\_data.csv” in it (the first one is created by Faker, from DataGenerator.php, and the second was created from an online tool). The “assets” folder should be the working directory for the project.
3. Ensure there is a folder called “schemas” in the “assets” folder with all these names:
   1. condition\_mysql.sql
   2. creditcards.sql
   3. doctor\_data\_view.sql
   4. doctor\_mysql.sql
   5. doctor\_patient\_mysql.sql
   6. hospital\_data\_view.sql
   7. hospitals.sql
   8. location\_mysql.sql
   9. patient\_data\_view.sql
   10. patient\_mysql.sql
   11. patient\_phone\_mysql.sql
   12. prescriptions\_mysql.sql
4. Ensure there is “commons-csv-1.1.jar” in the folder as well, compiled on project compilation. It will be used for exporting CSV files.
5. Run the project! I used Intellij as an IDE for the project in general.