Design and Optimization of an Interoperable Hospital Database System

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Ahnaf Khan
Spring, 2021

Technical Project Team Members
Khan, Ahnaf

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature Ahnaf Khan Date 5/6/2021
Ahnaf Khan

Approved Nada Basit Date 4/23/2021
Nada Basit, Department of Computer Science

Design and Optimization of an Interoperable Database System

Introduction

A modern issue affecting electronic healthcare databases has to deal with how critical and sensitive healthcare data/information (such as patient records) is stored, managed, retrieved, and (most importantly) shared, especially at the level of hospital systems. According to HealthCatalyst, there are a multitude of issues that currently plague these healthcare databases (Cardon, 2018). One is that although these databases have somewhat facilitated easier management of data, it does little to erase the fact that there is still an overwhelming amount of raw data. Attempting to make sense of all this raw data that can come from a variety of sources (i.e. lab test results, patient health info, dietary info, employee info etc.), without some form of structure or analytic tools can become quite challenging. Another such issue has to deal with the fact that different hospital systems set up and structure their database systems differently from one another, which negatively impacts sharing of data in a number of different ways. In more technical terms, how a healthcare organization decides to structure its entire database system puts a heavy precedent on numerous different factors/qualities; some of these include security of the files (along with other kinds of associated information and data), quality of the file retrieval system, and efficiency with which these files can be updated.

A commonly overlooked issue is interoperability, which refers to the ease and extent to which two or more disparate applications, components, or database systems can exchange and use each other's data and information. Sharing of hospital data across disparate hospital systems is of course an important challenge to tackle, as there are many instances where sharing is necessary such as, for example, when patients are present at different hospitals, when employees get transferred to different systems, and when sharing of necessary data and equipment becomes

vital during extraordinary times, e.g. a pandemic (like the current COVID-19 pandemic). However, depending on how different the hospital database systems are (in terms of its structure, data components, user classes), it can be quite challenging to convert from one database format to another. It's in this conversion process that many critical issues can come up, such as issues with security of file transfers/conversion, ensuring user access privileges are roughly the same across both (or more) systems, issues with data redundancy (which takes up storage space in the system), and so forth. All of these factors in combination add up to the difficulty of achieving interoperability (Bhartiya et al., 2016).

Over the years, there have been many solutions and attempts at improving interoperability (among other factors plaguing such databases) through various research studies and articles. One such example that piqued my (as well as my research advisor's) interest was one where a few Nigerian researchers try to improve the usability of a local hospital database system in Nigeria, and then generalize their findings. In this study, Amaechi et al. (2018) found that the current structure was not very conducive to smooth and secure data storage and retrieval among discrete user classes (such as patients, doctors, nurses, administration, etc.)[8]. In their study, they proposed a new, more automated system that they believe concretely define user classes and will enable more secure data and information upload, retrieval, and updating, providing an entity relationship (ER) diagram for their new proposed system along with corresponding tables.

The aim for my research is as follows: (1) determine whether or not the database information and tables provided by Amaechi et al. (2018) follow the rules of standard normal forms of proper database design using 3NF and BCNF tests (2) upon necessary completion of these standard normalization procedures on these tables for the design of their database,

determine whether or not the database is potentially interoperable using guidelines detailed by the research study by Bhartiya et al.[3] , and (3) develop a web-frame/mockup of a simple mobile/web application that can be used to interface with this database. Such an app could potentially be used by healthcare personnel in places like Nigeria. These aims are for the purpose of improving the interoperability of the proposed healthcare database system by introducing a usable means/platform through which data can safely and efficiently be shared across organizations.

Methods

1. *Normalization*

Normalization of table schemas ensures that there is consistency and no redundancy in the tables, which can save a lot of unnecessary wasted storage space and time. I followed 3NF and BCNF, both of which are concrete procedures for normalizing tables. After 3NF and BCNF, it was determined that the tables provided by Amaechi et al. were already normalized, and did not require any further changes. Results of normalization are visualized in Figure 1. These tables were then converted to an ER Diagram for better visualization as seen in Figure 2.

Table 1. Patients Record			Table 2. Patients Triage		List of FD's
patients_id	PK		triage_id	PK	*Table 1*
patients_firstname			patients_id	FK	patients_id --> patients_firstname
patients_lastname			nurse_id	FK	patients_id --> patients_lastname
age			blood_pressure		patients_id --> age
marital_status			heart_beat		patients_id --> marital_status
gender			sugar_level		patients_id --> gender
LGA			height		patients_id --> LGA
state			weight		...all of table 1 essentially (except FK)
home_address			time_of_reg		
resident_address			comment		
phone_no			fee	PK	*Table 2*
email					triage_id implies everything not labelled FK or PK
next_of_kin_id	FK				
comment					*Table 3*
sympton_sickness					test_id implies everything not labelled FK
passport					
patint_type					*Table 4*
passport					bill_id implies everyhting not labelled FK
time_of_reg					
admin					
Table 3. Patient Test			Table 4. Patients' Bill		
test_id	PK		bill_id	PK	
test_name			patients_id	FK	
test_description	Null		doctor_charge		
patients_id	FK		medicine_charge		
lab_scientist_id	FK		room_charge		
doctor_id	FK		operation_charge		
date_time			no_of_days		
return_date			nursing_charge		
test_result			advance		
comment	Null		health_card		
fee			lab_charge		
			bill		

Figure 1. Normalization of healthcare data sharing portal tables

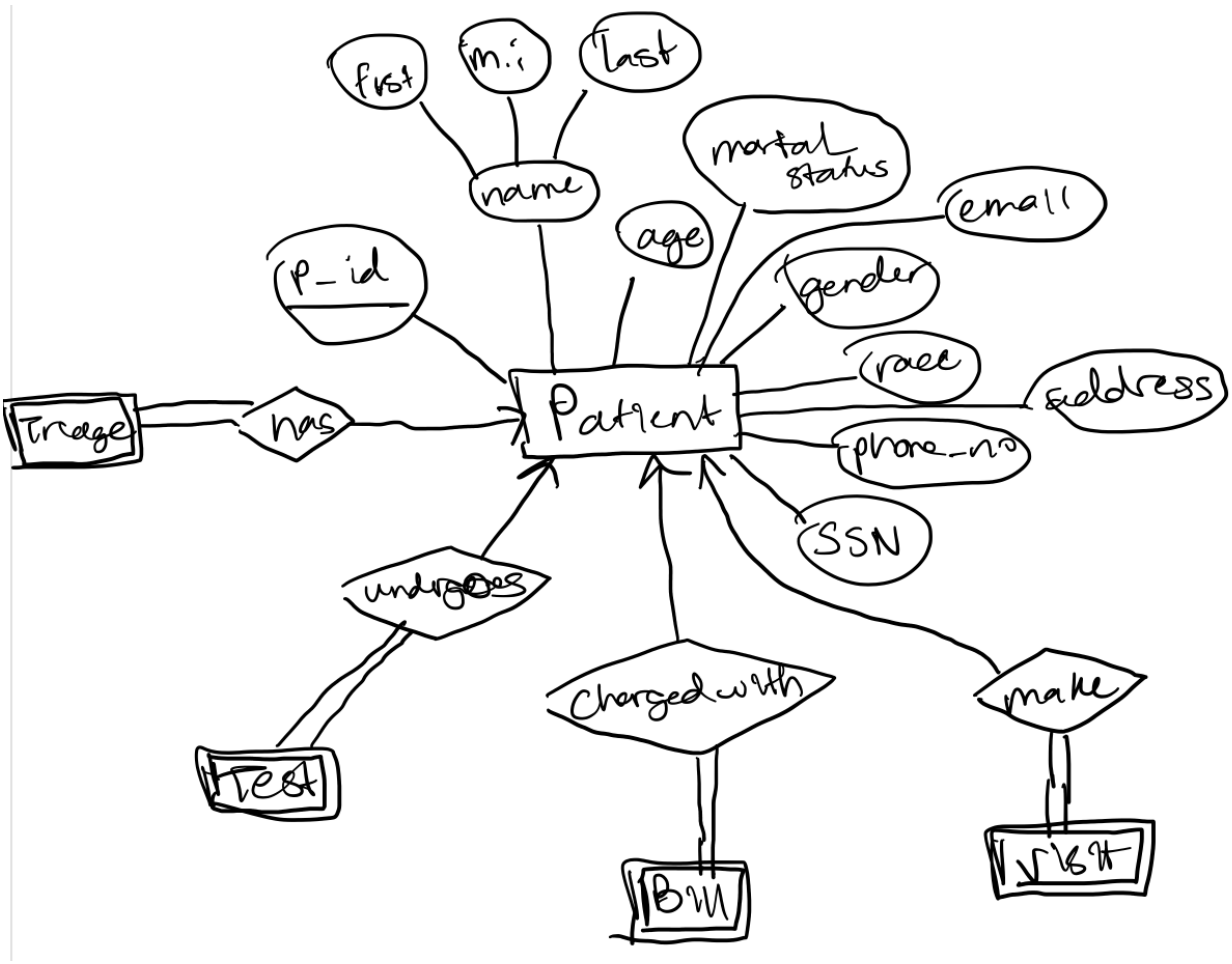

Figure 2. ER Diagram

2. *Google Cloud Firestore*

Hosting of the actual database itself was done via Google Cloud Firestore, which is a cloud-hosted NoSQL database that, according to Google Cloud, is simple enough for rapid prototyping yet scalable and flexible enough to grow to any size. The choice for this particular host was made out of a need for simplicity and ease of setting up the database. In Firestore, data is stored in “collections”, which can be thought of as tables in relational databases, with “documents” being synonymous to individual lines in the table, where each document can be referenced by a unique ID usually auto-generated. However, a critical difference between

Firestore and other kinds of traditional, relational databases is that you can add collections within documents. Hence, for my database set up I decided to set up one overall collection called "patient-record", which would store all the basic patient information that is the most critical information, and within that would be other collections relating to triage, test, bill, and visit information – which are not as crucial as the basic patient record information such as name, age, DOB, etc. Figure 3 shows what the basic database looks like in Google Firestore platform.

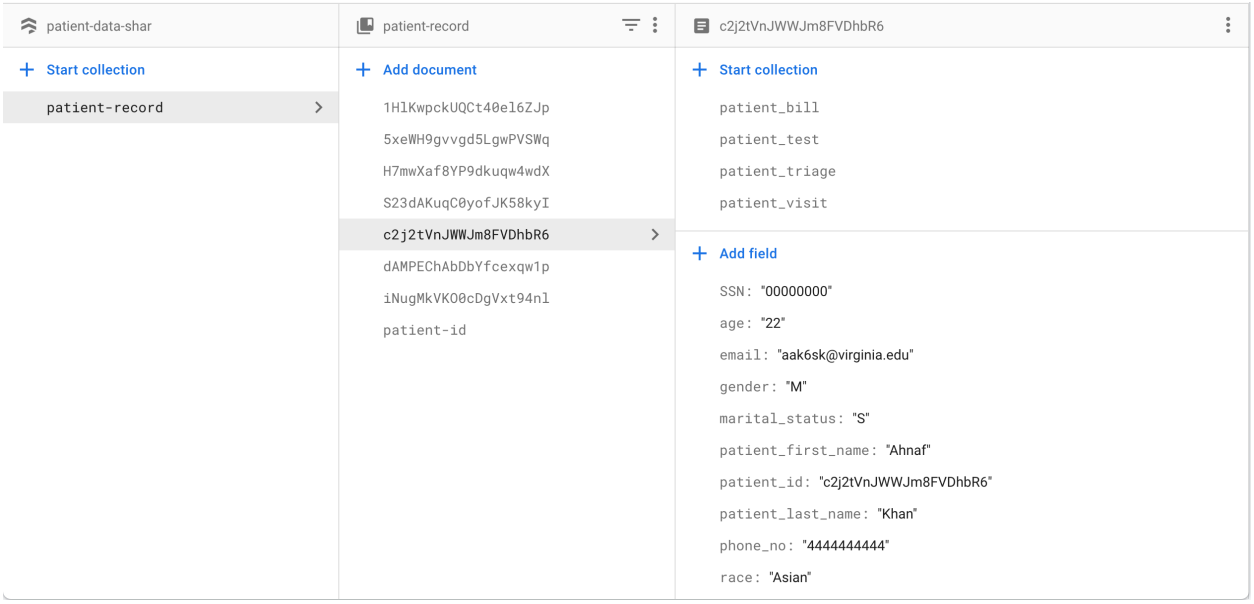

Figure 3. Patient record database

3. *Database Usage and Interfacing*

The final step was implementing a means through which one could interface and interact with this database. To do this, I followed the Cloud Firestore Documentation on Google Firebase website to implement the two most basic functions of adding and updating information to the collection via Python in VSCode. To add a patient, user must run the python file (located here: https://github.com/khancon/patient-health-recods/blob/main/hospital-record.py) and type in 'Add' when it asked to either "Add or Update". Then, a series of fields will be outputted asking

for input for the various fields represented in each of the collections for an individual patient, some of which can be left blank. Add functionality is displayed in Figure 4.

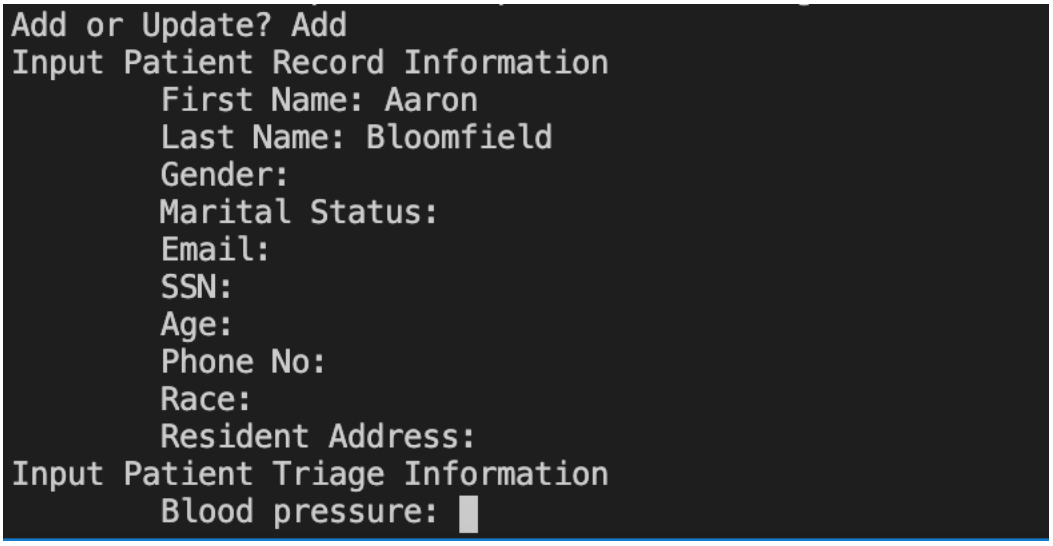

Figure 4. Add patient info functionality sample (more fields exist not shown here)

The other functionality implemented was update, which is used to update the information of an existing patient. Updating requires a valid ID of a patient that exists within the patient-records collection. Then user will be prompted to enter the collection name within which they would like to update data, the actual field they would like to update, and then finally the updated value of the field. Update functionality is displayed in Figure 5.

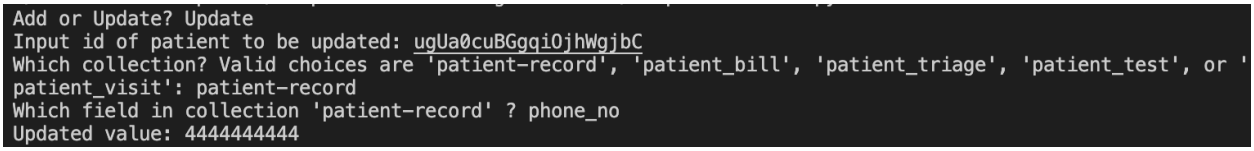

Figure 5. Update patient info functionality sample.

Discussion

The chief goal of interoperability in healthcare is to facilitate the seamless inter-exchange of health-related information amongst caregivers and hospitals for clinical-decision making (Iroju et al., 2013). However, such a goal is still yet to be fully actualized. This is especially true in the United States, as different hospital systems can utilize database formats and processes that differ from one format to the next, hindering full interoperability. As a result, attaining interoperability comes with many challenges, with limited solutions.

This paper offers a solution that can potentially improve interoperability across databases -- a centralized database/tool that individual hospitals can interact with to add, update, and retrieve patient information. Such a tool can provide a means through which hospitals can securely share information with each other, as no such thing currently exists on a widespread scale. As of now, this solution is just a back-end component only limited to add and update patient info functionalities as demonstrated above; further iterations will require addition of other functionalities, such as retrieving patient info with valid ID, secure sign-in and authentication via organization sign up, a front-end that users can interact with, etc. Due to time constraints, these other features could not be implemented.

Usage of such a centralized system could offer immense benefits to all participating health organizations. Particularly, individual organizations wouldn't have to spend too much time or energy worrying over how to transmit their data securely and efficiently to other organizations. Additionally, organizations don't have to continuously update/change the protocols they use for processing/deciphering data they receive from other organizations, as data can be transmitted in formats that differ from one hospital to the next. In fact, usage of such a centralized tool could potentially eliminate any need for having such protocols in the first place,

as the tool itself would be responsible for securely sharing data in formats that all participating organizations, in theory, could process efficiently. Medical professionals can simply log onto this system to add/update patient info. This would especially be helpful in circumstances where patients, who aren't registered with a specific health organization, can have their vital health information accessed in a timely manner to deliver effective care. Alternatively, there can be business logic directly implemented into a hospital's database system that automatically uploads/updates patient info to this central server tool. Having such could make the system run even more efficiently so that medical professionals aren't faced with redundant tasks, and instead, can focus more of their time and energy towards actually caring for their patients.

Nonetheless, an issue that is immediately raised in context to this solution is the idea of security. If hospitals, especially those in the US, are to adopt such a centralized system, there needs to be strict compliance with Health Insurance Portability and Accountability Act (HIPAA), which federally mandates the complete privacy of health-related information. Hence, in order for this system to work, all participating hospital organizations would need to agree to a set of common standards and protocols that they would all abide by, particularly in regards to preserving data safety when uploading/changing patient information to this central server. Additionally, a point of concern could arise from the non-relational nature of the Google Cloud Firestore database. While very efficient in uploading and storing data, there is no way to set hard parameters and rules/schemas for uploading data to the patient-record collection. In other words, anyone could practically set whatever fields they want in the collection. This could definitely lead to some huge issues in the future if there aren't concrete rules set early on. Such issues will need to be resolved in order for this centralized server tool to work as a means of improving interoperability across the board.

References

Amaechi, J. C., Valerian, A. C., & Sixtus, N. E. (2018). Design and Implementation of a Hospital Database Management System (HDMS) for Medical Doctors. International Journal of Computer Theory and Engineering , 10 (1), 1–6. doi:10.7763/ijcte.2018.v10.1190

Bhartiya, S., Mehrotra, D., & Girdhar, A. (2016). Issues in Achieving Complete Interoperability while Sharing Electronic Health Records. Procedia Computer Science , 78 , 192–198. doi:10.1016/j.procs.2016.02.033

Cardon, D. (2018, October 17). The Healthcare Database: Purposes, Strengths, and Weaknesses. Retrieved from https://www.healthcatalyst.com/insights/healthcare-database-purposes-strengths-weaknes ses

Iroju, Olaronke & Soriyan, Abimbola & Gambo, Ishaya & Olaleke, J.. (2013). Interoperability in Healthcare: Benefits, Challenges and Resolutions. International Journal of Innovation and Applied Studies. 3. 2028-9324.