

Brandon Sheehan

Professor Ahmadnia

CSE 5720

10 December 2023

## Medical System Database

### Introduction

The database that I constructed was a medical database to store information about appointments, doctors, patients, and medical history. Medical databases are extremely important and are standard in the medical industry. Before the construction of medical databases, medical records were stored physically in offices and hospitals. This was not an effective way to store or access information. Databases revolutionized ease of access and convenience for storing these important records. Medical information is protected by a variety of HIPAA guidelines. This is important when considering the design of a database because all of the information stored within the database must be held to a much higher level of security. The objective of my database was to replicate a simple hospital records system that could hold information about doctors on staff, patients, medical history of each patient, and appointment information for each patient. It would become very important to effectively plan the execution of the database before creating it. For the implementation of the database, I utilized MySQL database management system.

## Implementation

Before getting the database up and running, I had to sufficiently plan how I wanted to structure each table. I knew that it would be important to store each of the entities separately so that there were no transitive dependencies on one another. I eventually decided to make four separate tables that held information about patients, doctors, medical history, and appointment information respectively. This made the most logical sense to separate the information this way and allow the tables to access one another with certain foreign key constraints.

After deciding on the general structure of the tables, it was important to decide what kind of information should be stored within each of the tables. My goal was to keep the design relatively simple so that it did not become too bloated with information. For the patient table I decided upon patient id, first name, last name, gender, address, phone number, and date of birth. It seemed that this was enough information to correctly identify each patient within the database. The patient id was an integer value used as the primary key to the patient table. The rest of the columns were varchar variables set up to hold strings.

### Patient Table

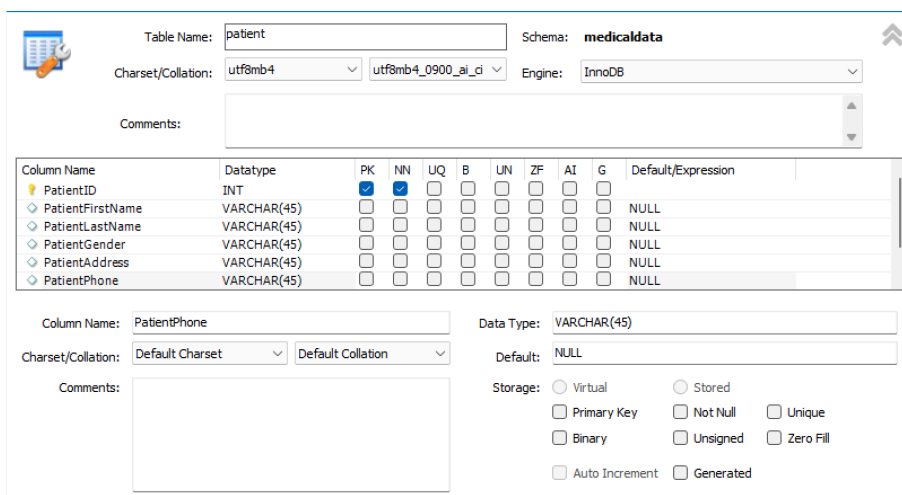


Table Name:  Schema: **medicaldata**

Charset/Collation:   Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
⚡ PatientID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
🔍 PatientFirstName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔍 PatientLastName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔍 PatientGender	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔍 PatientAddress	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
🔍 PatientPhone	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Name:  Data Type:

Charset/Collation:

Default:

Comments:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

For the doctor table, I included doctor id, first name, last name, practice address, phone number, and specialty. The primary key was set to doctor id as an integer value in order to correctly identify each doctor. The rest of the values were set to varchar values to hold strings as with the patient table.

## Doctor Table

Table Name:  Schema: **medicaldata**

Charset/Collation:   Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
DoctorID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
DoctorFirstName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
DoctorLastName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
DoctorAddress	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
DoctorPhoneNumber	VARCHAR(25)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
DoctorSpecialty	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Name:

CharSet/Collation:

Comments:

Data Type:

Default:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

For the medical history and appointments tables, I applied the same design philosophy. The medical history table included history id, family conditions, vaccines, surgeries, allergies, and patient id as a foreign key constraint. The history id was the primary key for the table. The patient id was referenced from the patient table to connect the medical history to each patient. With regards to the appointments table, I included an appointment id, appointment time,

appointment date, appointment location, diagnosis, treatment plan. I also included patient id and doctor id as foreign key constraints that allow me to reference what patient and doctor will be at each appointment.

## Medical History Table

Table Name:  Schema: **medicaldata**

Charset/Collation:   Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
FamilyConditions	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Vaccines	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Surgeries	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Allergies	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
PatientID	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Name:  Data Type:

Charset/Collation:

Comments:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

## Appointment Table

Table Name:  Schema: **medicaldata**

Charset/Collation:   Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
AppointmentLocation	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
PatientID	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
DoctorID	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Diagnosis	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
TreatmentPlan	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Name:  Data Type:

Charset/Collation:

Comments:

Default:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Using foreign keys, I was able to effectively connect the information from each table in a way that made sense. I scanned the design for any extraneous information that could be removed. After doing so, it was time to finalize the design and test the database with some sample data. I used a sql query in order to populate each table with information. For each table, I inserted six rows of data in order to ensure that the tables were functioning as I intended.

### Tables with Sample Data

Limit to 1000 rows

```
1 SELECT * FROM patient;
```

PatientID	PatientFirstName	PatientLastName	PatientGender	PatientAddress	PatientPhone	PatientDOB
1	Brandon	Sheehan	Male	1374 Tolstoy Way	9519619930	11/17/1999
2	George	Jefferson	Male	5734 Ranch Way	9519419950	12/01/1987
3	Bella	West	Female	8585 Washington St	9098784040	01/01/2001
4	Jason	Welch	Male	1834 Diagon Alley	9674035555	12/25/2000
5	Regina	Smith	Female	7755 Lincoln Ave	9095550000	04/03/1950
6	Brandon	Sheehan	Male	9090 Central Ave	9519619970	6/17/1967
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Limit to 1000 rows

```
1 SELECT * FROM doctor;
```

DoctorID	DoctorFirstName	DoctorLastName	DoctorAddress	DoctorPhoneNumber	DoctorSpecialty
1	James	Pitt	6060 Hopkins Ave	9519619970	Cardiology
2	John	Hernandez	3456 Main St	9519319950	Urology
3	Samantha	Suarez	9091 Washington St	9097784040	Trauma
4	George	Welch	1864 Diagon Alley	9674045555	OBGYN
5	Regina	Long	7655 Lincoln Ave	9095541000	Dermatology
6	Jonathan	Bates	9010 Central Ave	9519319670	Pediatrics
NULL	NULL	NULL	NULL	NULL	NULL

**Query Editor**

```
SELECT * FROM medicalhistory;
```

Limit to 1000 rows

---

**Result Grid**

HistoryID	FamilyConditions	Vaccines	Surgeries	Allergies	PatientID
1	Colon Cancer	Up to date	Wrist Fracture	Peanuts	1
2	Alzheimers	Up to date	None	Pollen	2
3	Skin Cancer	flu shot	None	Pineapple	3
4	Depression	covid vaccine	Femur Fracture	Tree Nuts	4
5	Anxiety	Up to date	Brain surgery	None	5
6	Addiction	small pox	None	Mold	6
* NULL	NULL	NULL	NULL	NULL	NULL

[illegible]

After executing the queries, it was clear that the tables were working as I had intended for them to do so. All values were being filled into the correct column in each table. I made sure when inserting values into the table that there were no null values.

## Results

The implementation of my medical database was not without problems. The first issue that I faced was deciding which attributes I wanted to assign foreign keys too. Primary keys are pretty straightforward because most of the time they are just an id number. With foreign keys, I had to decide which table I wanted to have reference to. It made the most sense to include foreign keys in the appointment table that referenced the doctor and patient table respectively in order to give access to them. The other foreign key I had to use was in the medical history table. I included the patient id as a foreign key constraint in order to assign a patient to a certain medical history. Most of the other aspects of creating the database were relatively straightforward. Creating the tables and executing queries is simple once you understand the relationships between each table. The most difficult part initially is understanding how each entity will be related to one another within your design.

## Conclusion

Through the design of this medical database, I was able to gain a lot of insight into how to carry out the process of creating a database. It was difficult initially to go from a small concept into a full database. Through taking small steps and planning what I wanted to include within the scope of the project. My main problems came from conceptually understanding the relationship between entities in each table. If the entities do not interact with each other properly, then your design is ultimately flawed. I spent a lot of time thinking about how I would set up these

relationships to ensure a smooth operation of the database. Once that was clear, the set up of each table became extremely easy because all the groundwork had been done. After completing the database, I can see how important it is to plan the execution before you actually start doing it. ER Diagrams and relational models are very important to understanding how you can turn a concept into a functioning database.