Database Design (Group 16A)

Nikola (nk465) & Ethan (et365)

In the rapidly-evolving world of healthcare, efficient patient management systems are essential to ensure timely and high-quality care. As part of Group 16A, we were tasked with designing the "Patient" interface, a crucial component of such a system. To tackle the complexity of the project, we developed a comprehensive database with four different tables: Bookings, Doctors, Patients, and Logs. In this report, we provide a detailed analysis of our design choices, including table and column names, variable types, and PK and FK selection to provide best normalization. I (Nikola) will provide insights on the Doctors and Logs DBs, while Ethan will delve into the Bookings and Patients DBs.

DB Design Diagram

The 'Doctors' table stores information about the medical professionals associated with the clinic. It contains the following attributes:

* DoctorID (PK): A unique identifier for each doctor in the system, as their name and specialty may not be unique and hence cannot be a PK.
* Name: The name of the doctor, stored as a VARCHAR(15) to accommodate most names' length.
* PhoneNumber: The doctor's contact number, stored as a VARCHAR with a length of 12 due to the UK's phone format - XXXXX XXXXXX. This attribute is not allowed to be null in case of emergencies.
* Background: An optional field that stores any important background information about the doctor. This field can be null in most cases and has a VARCHAR(255) type. I choose a length of 255 because that is the SQL standard for a longer attribute.
* Speciality: The doctor's area of expertise which cannot be null. It will help with department filtration in the future.

For example, the PhoneNumber field might be used in an emergency, and the Background field could store information about a doctor's previous surgeries or research projects. The Speciality field will allow for easier filtering of doctors based on their area of expertise.

The ‘Logs’ table records users’ actions within the interface, providing the administrators with valuable information on how the system is used. It contains the following attributes:

* LogID (PK): A unique identifier for each log in the system, ensuring no two entries’ PKs are identical.
* PatientID: An identifier connecting the ‘Logs’ table with the ‘Patients’ table, allowing administrators to track user activity by patient.
* Timestamp: A crucial attribute that records the date and time of each user action, aiding in problem detection and system maintenance.
* Action: This column stores a human-readable description of the users’ actions, such as logging in or booking an appointment. Its type is VARCHAR(255), providing ample space for a detailed description.

All the attributes in the ‘Logs’ table are required and cannot be null. The LogID and PatientID are essential for connecting user actions to the relevant patients and ensuring the uniqueness of each log entry. The Timestamp is crucial for tracking the sequence of events within the system and identifying potential issues. Finally, the Action attribute provides insight into how the interface is used and is critical for administrators to understand user behavior.

The ‘Patients’ stores information about registered users that can be used by both the system and the doctors to aid the patient. It contains the following attributes:

* PatientID(PK): A unique identifier for every patient that exists in the system, given that any other attributes can be common between patients.
* FirstName: The first name of the patient, stored as a VARCHAR(20) to accommodate any and all names. This field cannot be null.
* Surname: Likewise to FirstName, the surname is also stored as a VARCHAR(20) to allow for patients with a longer surname. This field can’t be null.
* Gender: The gender of the patient is a VARCHAR(20) to accommodate the options a patient can enter, consisting of “male”, “female”, “other” and “prefer not to say”. This field also can’t be null.
* Age: Age is an INT(3) that has a check statement ensuring the age is less than 130. The reasoning is because a patient cannot have an age over 999. By limiting the patient age to 3 digits, we can restrict the patient’s entry to prevent incorrect data. It can’t be null.
* Phone Number: Phone number is stored as a VARCHAR(12) due to the fact that UK phone numbers are stored as 11 length numbers. VARCHAR is required to accommodate the space. Since a phone number may be used to inform the patient, it can’t be null.
* DoctorID(FK): The DoctorID is a foreign key that relates it to the doctors database, allowing a patient to be linked with a doctor. DoctorID in the doctors table is a serial integer, so the one in patients has to be as well to connect. Since it’s a foreign key, it cannot be null.
* Details: The details attribute is stored as a VARCHAR(100), since although text may be stored, it is unlikely to be a vast amount of data. Since it is optional, Details can be null.
* Messages: Messages is a Text field that stores information about services that the patient uses inside of the system, e.g., create a booking, changing doctors. Given that it could hold a vast amount of data, TEXT is the best fit. Since the patient may not use the system outside of registration, it can be null.

The ‘Bookings’ table stores information about appointments that patients book with their doctor to discuss their medical issue. It contains the following attributes:

* BookingID(PK): BookingID the unique identifier of every booking recorded. Primary keys can be null.
* PatientID(FK): This key is associated with the Patients table, allowing a patient to be associated with a booking. Since is a foreign key, it has the same data type as key it matches in the other table: INT and cannot be null.
* DoctorID(FDK): DoctorID is another foreign key that links ‘Bookings’ and ‘Doctors’ together. The data type is INT. Since it is a foreign key, it can’t be null.
* Time: Since the booking will occur at a certain time, it is crucial that it is recorded. As a result, the data type used here is ‘Time’ which cannot be null.
* Date: Likewise, to Time, the date of the appointment is also crucial to the system. The date is hence stored as Date. The data type is not null/
* Detail and Prescription: After a booking has occurred, the doctor may provide details and prescribe prescriptions for the user. Since the information is unlikely to be long, VARCHAR(250) provides enough characters for the doctor to ulitise. Since they are both optional, they can both be null.

There is also a UNIQUE constraint to ensure that both doctors and patients can’t have a booking at the same time.