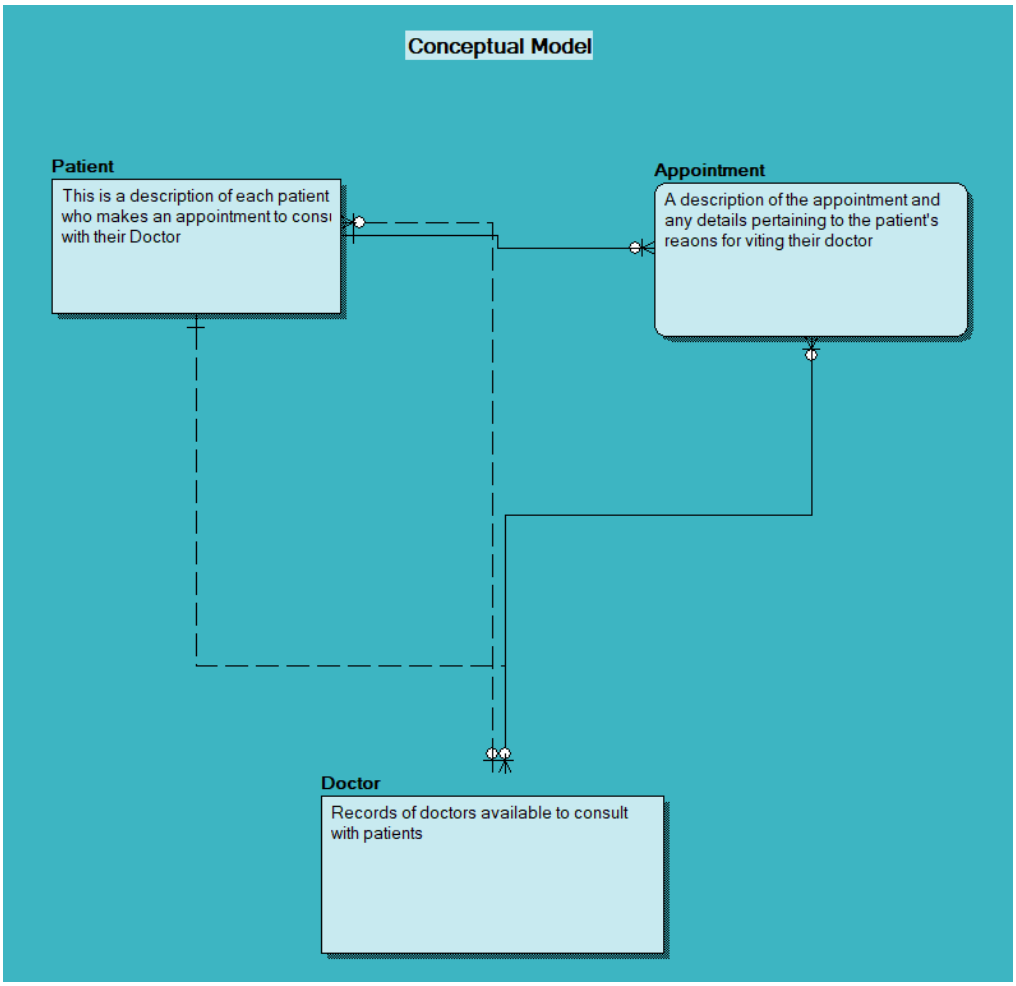
KING GYASI
DATA MODELING – SB CLINIC - 2.24.2025

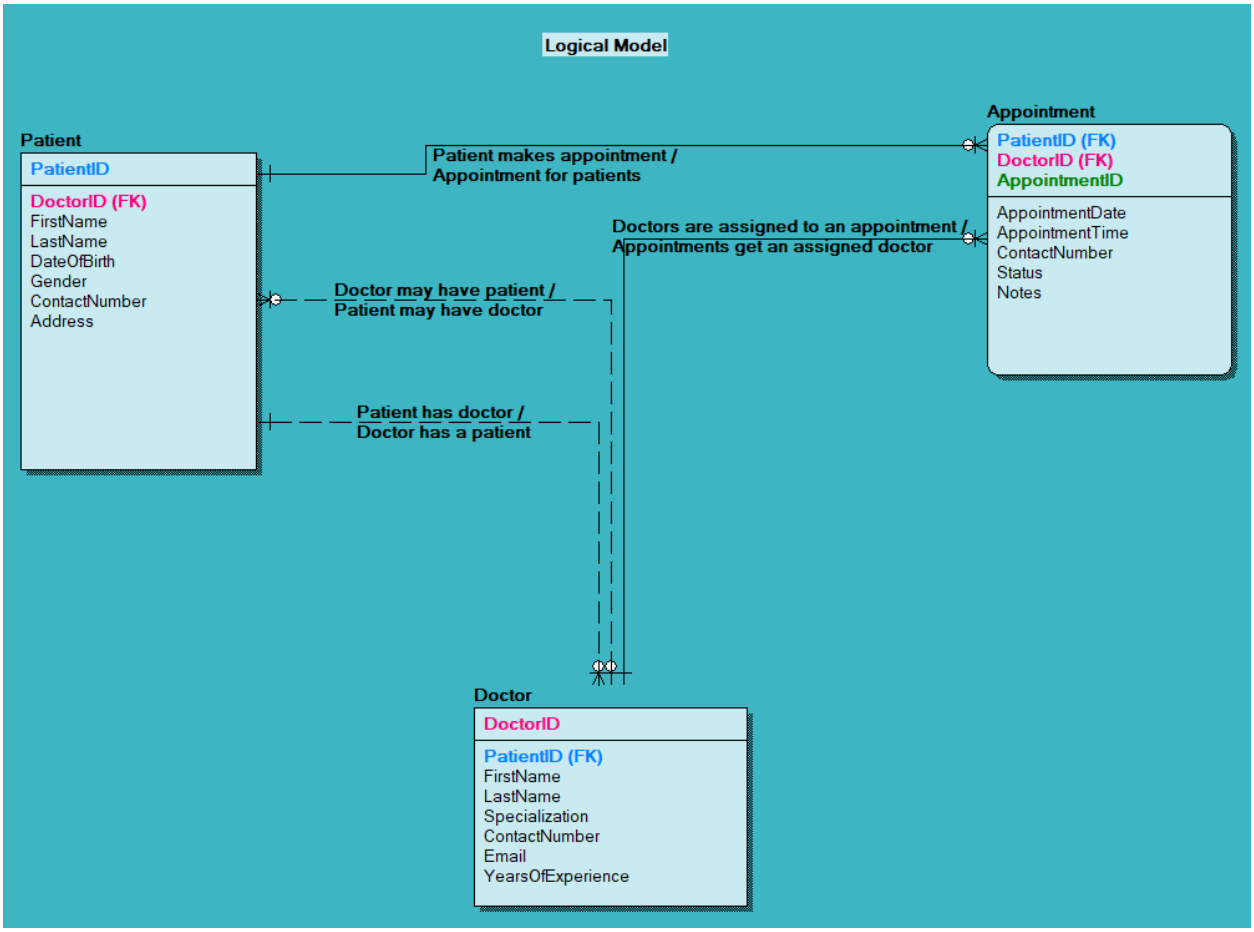
**DOMAIN DESCRIPTION**

A clinic database is an interesting example because it highlights the essential components needed to efficiently manage patient care, scheduling, and medical staff. I was naturally drawn to this domain/industry because of my background. The 3 entities chosen were 'Patient', 'Appointment' and 'Doctor'. The **Patient** entity ensures that personal details and contact information are recorded for proper identification and communication. The **Appointment** entity is crucial for tracking visits, managing doctor availability, and maintaining a history of patient consultations. The **Doctor** entity helps in organizing medical professionals by their specialization, experience, and contact details. Key attributes like contact information are essential for communication between patients, doctors, and the clinic, ensuring smooth operations, timely reminders, and effective patient management. This structured approach is designed to improve clinic efficiency, reduce scheduling conflicts, and enhance patient care.

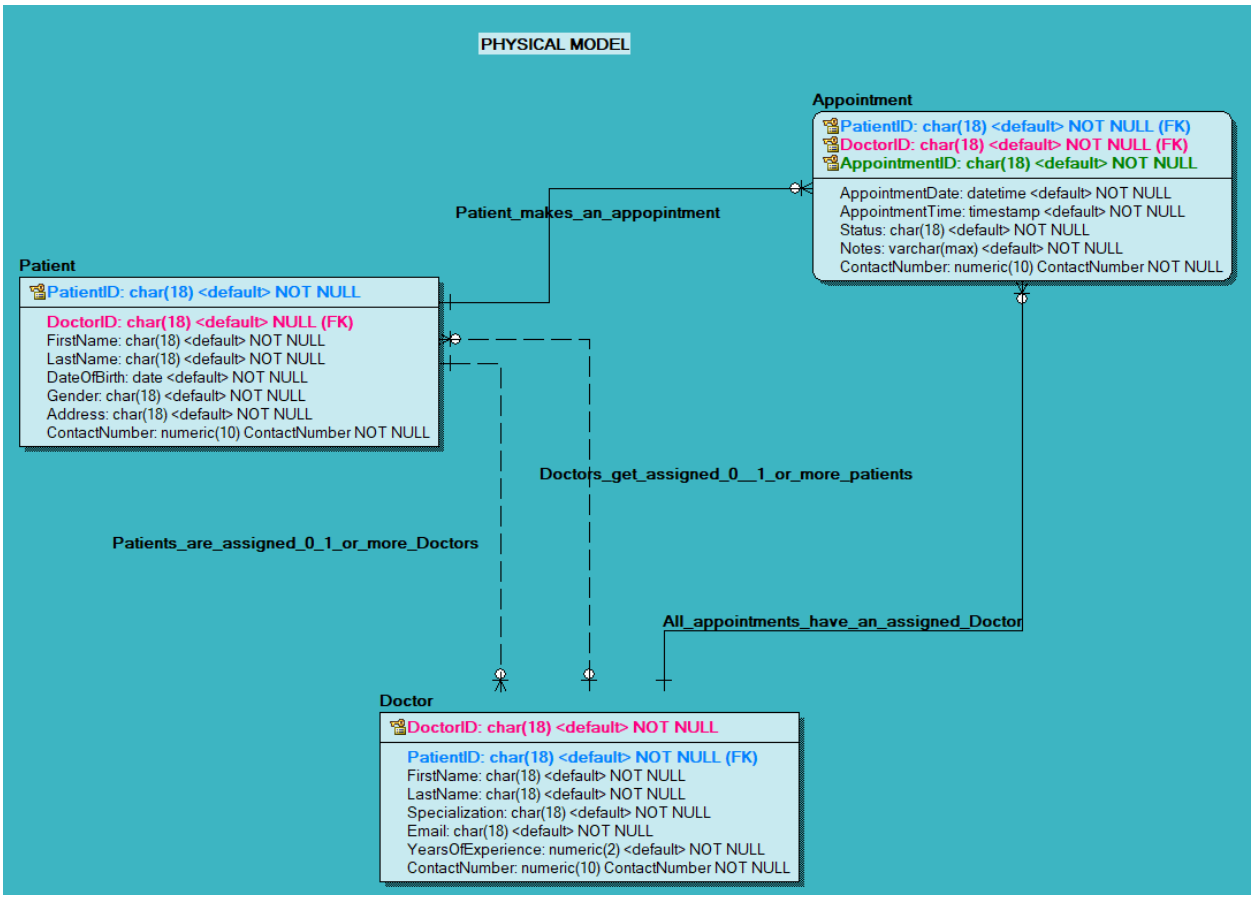
**DATA MODELING**



KING GYASI
DATA MODELING – SB CLINIC - 2.24.2025



KING GYASI
DATA MODELING – SB CLINIC - 2.24.2025



KING GYASI
DATA MODELING – SB CLINIC - 2.24.2025

# DISCUSSION OF DESIGN DECISIONS AND CHALLENGES

**Conceptual:**

The **Patient** entity is linked to the **Appointment** entity, as each patient can have multiple appointments. The **Doctor** entity connects to **Appointment**, ensuring that each appointment is assigned to a specific doctor. Additionally, the **Doctor** entity is indirectly related to **Patient** entity through appointments, as doctors provide medical care to patients during their scheduled visits. This structure allows the clinic to efficiently manage their data; tracking and managing patient history, doctor assignments, and appointment schedules.

**Logical:**

In this database, primary keys (PKs) and foreign keys (FKs) were carefully chosen to establish clear relationships between the 3 entities (patient, appointment, and doctor).

PatientID (PK) in Patient uniquely identifies each patient, while DoctorID (FK) links each patient to their primary doctor, wherever applicable.

AppointmentID (PK) in Appointment ensures each appointment is uniquely identified, while PatientID (FK) and DoctorID (FK) establish relationships with the respective patient and doctor involved in the appointment.

DoctorID (PK) in Doctor uniquely identifies each doctor. The presence of PatientID (FK) in this entity enables the (optional) modeling of a doctor-patient assignment separately.

This schema allows the clinic to efficiently track patient history, doctor assignments, and scheduled appointments while maintaining data integrity.

Physical:

Within the physical model, all attributes under the **Patient, Appointment, and Doctor** entities are set to **NOT NULL** because the client emphasized the importance of capturing complete and accurate data for effective clinic management. The process of ensuring that critical inputs such as patient and doctor details, as well as appointment-specific information like notes (describing the reason for the visit) are always recorded helps maintain data integrity, prevent incomplete records, and enhance patient care. By enforcing this constraint, the system guarantees that every appointment is properly

documented, every patient is linked to the correct doctor, and no essential information is missing, ultimately leading to efficient operations and better healthcare outcomes.

Challenges:

A key challenge design challenge was linking the doctors to both appointment and patient entity. Mainly when it comes to ensuring the correct **one-to-many relationship** between **Doctors and Patients** (and its reverse relationship) while maintaining proper **one-to-many relationships** with **Appointments**. Since a doctor can have zero, one or multiple patients, and a patient can see zero, one or multiple doctors over time, storing this relationship directly in the **Patient** or **Doctor** table can lead to **data redundancy** and **complexity**. Instead, using the **Appointment** table as a bridge ensures that each interaction between a doctor and a patient is properly tracked.