Data Base Team Reflection

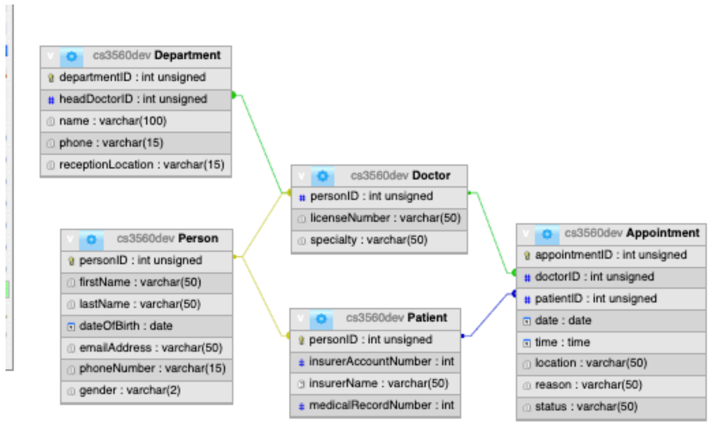
The team explored some options to store the data for our system, and we decided to use the DB4Free online database which is the administration of MySQL. The database was an excellent resource to get used to the SQL-type database. All the group members can access the same data after the database teams create mock data. However, the team encountered a couple of issues with this database. It takes a long access time to fetch data. Also, it has restrictions on accessing data within a certain period.

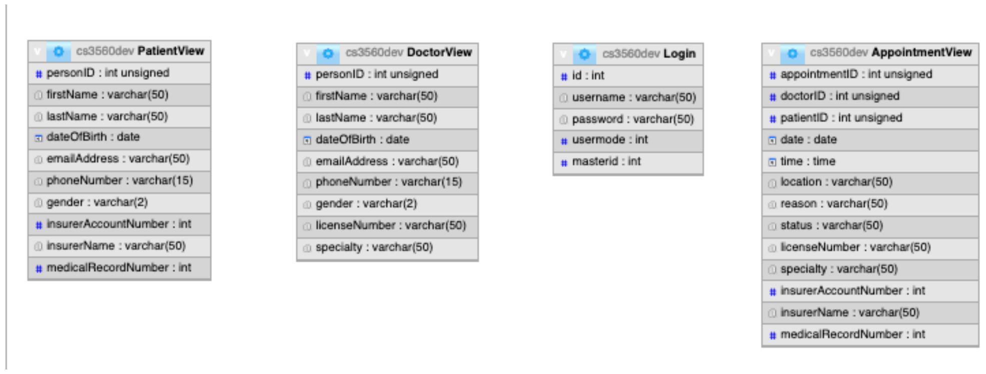
They are setting up a database schema to make five classes: person, doctor, patient, appointment, and department. In the system, the team designed the person as a supper class for the doctor and patient sub classes. The person and patients must have the same personID to realize this relationship in the database. When the new patient or doctor class is created, the API calls the SQL statement to create a new row in the person table with the data provided by the user. Then, the API fetches the new personID and crates the new patient or doctor row in the database with an inheritance relationship. The process promises the new person and patient or doctor data share the same primary key in the database tables to represent the inheritance relationship in the database.

To show the patient and doctor data in a person table, it is required to combine person data and patient or doctor data in each table. Creating an inner join table between a person and a patient or doctor table prepares all the necessary information for the user. The team made the patient and doctor view to consolidate the person and the patient or doctor attributes. The person class joins into the patient or doctor class.

The same idea is applied to create an appointment view. The opponent view needs all the patient, doctor, and appointment information. The database team decided on three steps. The first step combines the person and patient tables to create the patient view. The second step connects the person and doctor tables to create the doctor view. The third step is consolidating the patient view, doctor view, and appointment table to generate the appointment view.

It has a lot of inner joins between tables and views, and it was challenging to organize tables and views in the database. After some tries and errors, the team successfully created the appointment view, which shows all the required information to check appointments for users, both the doctors and the patients.





Looking back on this project, it was certainly helpful and saved time to design diagrams and database schema to set up data in the database. Creating complex tables and views as appointment views needs a blueprint to organize efficiently. Through this group project, the database team and other group members benefit from designing diagrams before implementing codes.