COMP 3150 Presentation Script

[General description of the project] ~ 30 seconds

The purpose of this project is to demonstrate our understanding of database design and implementation by creating a database management system for a chain of hospitals.

Following the requirements of the project description we were to design a database with entities and their relationships to one another. We were required to submit ER diagrams and database schemas as milestones for this project to show our progression to the final working database.

The final submission is a fully functional database that manages all of the hospitals information such as doctors, nurses, patients, and pharmacy bills.

[Description of requirements for the database]

First, let me briefly go over the requirements as you can see in our Entity Relationship Diagram. This diagram has been slightly updated since our first milestone to fix any issues we had. As you can see, the requirements of the database were to have 3 hospitals where each hospital can admit a maximum of 10 patients and has 3 doctors and 5 nurses on staff. The hospitals can order medication from any of the three available pharmacies. Patients can have multiple doctors treating them for various conditions, but only one primary doctor. Nurses can care for a maximum of two patients.

One interesting feature we added was the Stocks relationship where each pharmacy stocks various drugs with various markups. This allows the hospital to order a specific drug at whichever pharmacy is selling it for cheapest. This also makes our database more realistic because pharmacies only have a limited amount of each drug, so if the hospital wanted to order a large amount of some medication, they might need to order from more than one pharmacy.

 [Description of the ER diagram and Schema design]

For the second milestone, Shieanne and I put all the entities and relationships from our ER diagram into this table, and I worked on normalizing it into 3rd normal form. In this table, the underlined attributes are part of the primary key, and the italicized attributes indicate foreign keys. You might notice a few changes from the ER diagram here. First, we had to have a separate WorksAt relationship for Doctors and Nurses. We also needed to make Location a separate entity. Finally, the AdmittedTo relationship has HospitalID, PatientID, Admitted and Discharged dates all as part of the primary key to allow a single patient to be re-admitted into the same hospital more than once.

Shieanne transformed this table into the SQL queries that we submitted for our second milestone, and I helped her with that, especially with making all the keys work correctly.

The derived attributes you see on the side are not actually a part of the schema; we did not want our database to be redundant, so we would derive them through SQL queries. At the time of filming, I am not quite done writing all the queries yet, but these are a couple of examples for the total cost of a patient’s stay based on the number of admission days, and the total cost of a drug being ordered from a specific pharmacy.

To enforce integrity constraints, Shieanne and I used trigger queries which replace insertion. You can see here that on insert, we first check some condition, and only if it is true, the data will be inserted into the table. This is an example of a more complex one that I did, which checks that the hospital actually has room to admit a patient, and that the patient isn’t already admitted, because we do not want to admit a patient when they are already admitted in the hospital, nor do we want a patient to be admitted to two hospitals at the same time, because they cannot be in two different place at once. I am also working on a file of TestQueries which will let you test these constraints. I am not done yet but I will try to write as many as possible and document them in comments to make them clear. For example, this one tests my Admit trigger - you can see that we’re trying to insert a bunch of patients on the same date range to fill up the hospital, and if you run this query you will see that the last patient is not inserted because the hospital reaches maximum capacity at that point.

[Description of the SQL data and how to navigate the files]

Once we implemented the schema in SQL, we began to add the necessary data and using bulk insert we inserted it into the database. However, we discovered that the bulk insert feature is incompatible with Azure Data Studio so we used a python program to generate the SQL insert code from our CSV files. In our attached zip file you will find folders titled Queries and TestData, in the Queries folder is all of the queries needed to setup our database on your own system. We suggest you first run the DropQuery file to ensure there are no duplicates in your database that will conflict with the rest of the project. Once this runs, run the EMRSCHEMA file, then followed by the BulkInsert or InsertQuery file. If you’re using Azure we suggest using the InsertQuery file, and with SSMS we suggest using BulkInsert. However, please note that with BulkInsert you must change the path names in the file to the path names related to the csv files.

Now that you have created and populated the database with information, running the SelectQuery will display different views and tables with important information.

[Description of the GUI]

Welcome to OneTouch EMR, a system designed to ensure that hospitals run efficiently without error. There are five views which display relevant information to the user and allows a smooth operation within the hospital. Some features include allowing the ordering of medication from registered pharmacies, viewing and paying outstanding bills, admitting, relocating and discharging patients from the hospital or to another, tracking which staff are responsible for each patient and much more. The system starts by the user logging in. Once it verifies which user type and individual you are it displays the relevant data. There are three user types in total Hospital administrator, Doctor and Nurse and each user is given specific permissions to view and/or make changes to information in the system.

The GUI was designed using HTML, python and Django, however with complications connecting our database to the Django project the views seen in the video are using sample data inserted into the SQLite database that came with the installation of Django. It was populated only for demonstration purposes of the goal for our end product.