University of British Columbia
Department of Computer Science
CPSC 304 2021

Summer Term1

Group Project - Implementation of a Relational Database

Project Title:	University Healthcare Booking System
Project Milestone:	M4

#	Student Name	Student Number	Email Address
1	Angela Li	35098152	angelali6462@gmail.com
2	Justin Jao	19923151	jao.c.justin@gmail.com
3	Xinyu Liu	75876581	xinyuliu910@gmail.com

A short description of the final project, and what it accomplished

Our project models a healthcare booking system that can be utilized by a university to schedule appointments between patients and doctors. Using a relational database management system (RDBMS) structure to organize data, with MySQL and PHP to support the backend programming logic, and HTML and CSS for the front-end design, users can interact with the RDBMS in 4 primary ways: viewing, inserting, updating and deleting records.

Users can view (some can do SELECTION/PROJECTION on) specified records of doctors, patients, medical exams and medical exam records, and some can also specify which attributes of the relations they want to view. The RDBMS provides INSERT, UPDATE and DELETE functionality (such as for doctors or appointments) to edit the records in the database.

Our RDBMS also provides JOIN functionality, allowing users to find patient names or doctor names, or symptoms or supervising nurses, for corresponding appointments (using JOIN between **Appointments, Doctors and Patients,** or **Appointments**, **ConfirmationOfSymptomsInvolvedInAppointment and Nurses**).

The total costs of selected medical exams can also be calculated (AGGREGATION on **MedicalExamFee**).

The NESTED AGGREGATION between **Doctor** and **Appointment** functionality outputs the doctors who have fewer than certain appointments after a certain date.

Finally, users can also query which medical exams need to be taken for certain symptoms (DIVISION between **MedicalExamsIncludedIn** and **ConfirmationOfSymptomsInvolvedInAppointment**).

In line with these query capabilities, the RDBMS is currently set up such that the intended user would be a healthcare administrator who manages the database system.

A description of how your final schema differed from the schema you turned in, why?

Our core design for our RDBMS stayed the same, with most of our relations (and their relationships) remaining unchanged. The primary changes implemented involved slight changes to the schema of each relation.

We added the *Type* attribute to the **MedicalExamIncludedIn** table because it was missing in our initial table, to enable recording of type information in the medical exams.

To enforce proper constraints, we specified that *Dosage* must be a foreign key in our **TakeAndSupervises** table, *Type* must be a foreign key in our **MedicalExamIncludedIn** table, to ensure we do not violate the referential integrity constraint.

We changed the *Date* attribute to *startDateTime* and *endDateTime* in the **Appointments** table because the three attributes are redundant and not convenient to query. This also allowed us to capture appointment times, instead of just dates, all in one attribute.

For the *DoctorID* foreign key in **Appointments**, we changed ON DELETE NO ACTION to ON DELETE CASCADE, so doctors can be deleted, which allows corresponding appointments relating to that doctor to be deleted.

Instead of only using the 3 subclasses in our ISA relationship (**Students**,**GeneralPublic**, **FacultyStaff**), we added back the **Patients** superclass into our relations because we need to reference the superclass to simplify our queries, and we could not do this with just the subclasses.

We also added NOT NULL to some of the attributes, to ensure that records would be identifiable and in conformity with constraints, and we added DEFAULT NULL for Doctor's age, specialization attributes because we want our system to have a default.

We also do not display some information captured in our entire schema, such as information about **Equipment** and **MedicineFees,** we did not add this into our interface because we don't have related queries.

Initially, in our **ConfirmationOfSymptomsInvolvedInAppointment** table, we had one foreign one *symptomName* referencing *SymptomsIdentify (SymptomName)* and one foreign key *PHN* referencing *patients (PHN)* but we changed it to foreign key *SymptomName, PHN* referencing *SymptomsIdentify (SymptomName, PHN)* because *SymptomIdentify* is a weak entity and has a composite primary key, while foreign keys can only reference to all of the keys in the composite primary key.