Nick Carrozza

CMPT220 Milestone for Project 2

RDBMS for Club Management: Implementation through SQL and Java

This project addresses a common issue that many everyday users face on a daily basis when they interact with various types of software systems. In summary, this particular project focuses on the implementation of a relational database and allowing the everyday user to interact successfully with a, rather complicated and intricate, database. In an effort to bring this to fruition, I've developed a clear plan for the schema, relationships, and identities of the various entities I introduce in my club management database. What follows is my progress thus far and what I plan to do with what I have currently.

My first step in creating this project began with some thought in identifying what it takes to have a fully comprehensive and meaningful database that clubs and organizations can use to organize the various amounts of data they have available to them. This led to the realization that there are several strong entities that must be considered for *any* club or organization, regardless of the type. These include members, advisors, events, and meetings. It makes sense that these are all things that a given club or organization would benefit from keeping track of. Various queries could be written based on information contained in these tables, such as the number of meetings held during a particular time period, or the title of a given officer, etc.

Perhaps just as interesting, the next step was to generate an ER diagram that gives a complete view of what it is these clubs and organizations would benefit from keeping track of. In an effort to satisfy the rules of database normalization, I've developed a schema that is seemingly in 3NF, if not BCNF, and have determined necessary associative entities between the strong entities identified earlier (please refer to the attached ER diagram for a complete overview of this).

Using Postgres as my platform, I've began writing CREATE TABLE statements for my database according to the relationships and keys outlined in the ER diagram. I think, for testing purposes, it would make sense to insert some test information into the database using SQL in Postgres. My ultimate goal is to utilize Java for this, allowing the user to enter information via prompts by System.out rather than having to actually write an "insert into" SQL statement much like a database manager would. My vision includes a Java program that prompts the user to select the type on information to enter which, from the back end, would be one of the strong entities identified earlier. Members, advisors, meetings, and events would all be candidates for a "type" or "class" of information that the user could enter information for. If the user were to select type "member" for example, they would then be prompted to enter all the pieces of data necessary to create a record in that table of the database. However, this must be done in a very particular manner. It would make sense to first ask for the piece of data that is the primary key of the table. This must be done to ensure that the user (1) enters *something* to serve as the unique identifier for that record in the table, and (2) ensures that this piece of information is currently non-existent in that table. What could make this especially difficult is the notion of satisfying NOT NULL and UNIQUE constraints I have identified in the CREATE TABLE statements. Once a valid primary key is entered, the Java program will then prompt the user to fill out the remaining set(s) of data necessary to provide a complete record in that table consistent with what is outlined in the corresponding CREATE TABLE statement for that table.

In terms of coding style, and whether or not to use an object-oriented approach vs. a functional approach, I'm leaning towards avoiding heavily using objects for the Java implementation. The reason for this is that I think it may overcomplicate my intentions. My initial plan is to create instances of the Scanner object to read the information the user types in, and then take that information and place it accordingly into a SQL INSERT INTO statement. I'm considering the use of methods for the different types of information the user could enter, although it's possible a conventional if-else statement could provide the same utility.

A few outstanding things need to be addressed for the successful implementation of this project. Most importantly, I have not yet linked Postgres to the program I write and compile Java in, IntelliJ IDEA. I have confidence that this can be done successfully. Once this is implemented, I can begin writing the Java program necessary to allow the user to enter relevant information. Another important aspect of this would be ensuring that, when relevant information is entered by the user, all tables that relate to that parent table are accurately updated and all changes adhere to the well known ACID properties. This may be very difficult to accomplish, and it's something I hope to work towards as I progress in implementing this system.