Nick Carrozza

CMPT220 Milestone for Project 2

RDBMS for Club Management: Implementation through SQL and Java

Abstract:

Included in this report is a comprehensive explanation and elaboration of my Project for Software Development I, a Club Management Relational Database Implemented through SQL and Java. In its entirety, his report includes an introduction, system description, requirements, literature survey, user manual, and conclusion.

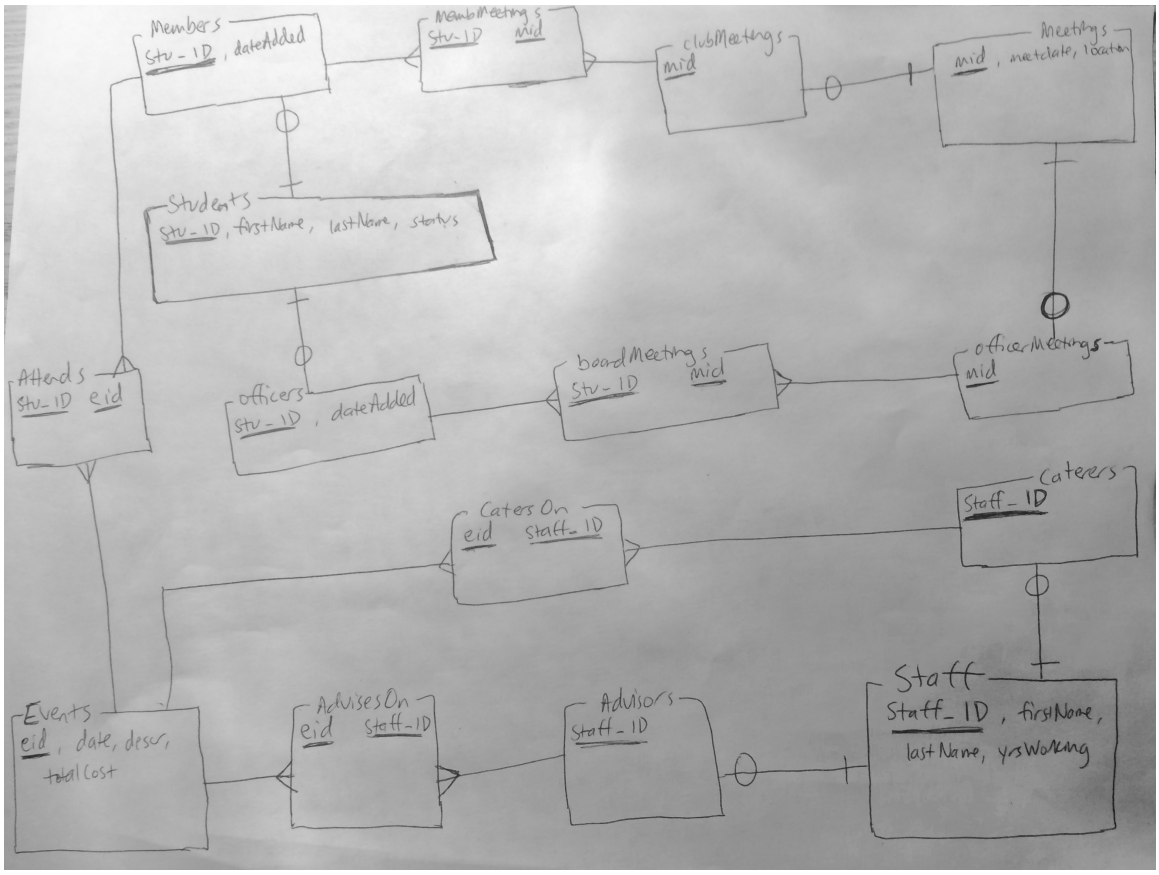
Introduction:

The motivation of this project is to address 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 while 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, which will be outlined and explained in this report.

System Description:

Much thought was given 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. However, this particular application will focus primarily on the insertion of data in a database, utilizing the INSERT INTO SQL command.

ER Diagram:



The ER diagram (crows feet notation) as shown above gives a complete view of what it is these clubs and organizations would benefit from keeping track of, and accurately depicts the relationships between the entities in the database. 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, as present in the ER. These are evident in the functional dependencies of the database as outlined below:

Students.stu_id → firstName, lastName, status

Oficers.stu_id → dateAdded

Members.stu_id → dateAdded

boardMeetings.stu_id, boardMeetings.mid →

membMeetings.stu_id, membMeetings.mid →

Meetings.mid → meetdate, location

OfficerMeetings.mid →

clubMeetings.mid →

Events.eid → date, descr, totalCost

Attends.stu_id, Attends.eid →

advisesOn.eid, advisesOn.staff_id →

catersOn.eid, catersOn.staff_id →

staff.staff_id → firstName, lastName, yrsWorking

caterers.staff_id →

advisors.staff_id →

Using Postgres as my platform, I've written a comprehensive SQL script with CREATE TABLE statements for each table in the database. Included in these declarations are necessary foreign key constraints as appropriate which protect the referential integrity of the data being entered. The user interface, written with Java, provides the user with a series of prompts asking them which entity they'd like to provide information on. Here, they must enter the correct data type of information being asked, otherwise they will be prompted to "try again" as invalid data was entered. This was implemented for each prompt in each entity via try/catch statements, as well as a while loop with a boolean test condition to determine whether or not correct input was entered. There are four main entities to choose from: Students, Events, Staff, and Meetings. Student, Staff, and Meetings each have two entity subtypes, Members and Officers for Students, Caterers and Advisors for Staff, and Club Meetings and Officer Meetings for Meetings. 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. The system first asks for the piece of data that is the primary key of the table. This must be done to ensure that the user enters *something* to serve as the unique identifier for that record in the table. 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. With this, it was necessary to develop a way for the user to enter information to make an entry to one of the associative entities in the database. This is handled with the "Student Details" and "Staff Details" prompt options. Even with only these two options, it is possible to access all six associative entities in the database. This is the case since two possibilities stem from Student Details, being Member or Officer, with two possibilities stemming from each Member and Officer. These each include two associative entities as possibilities from this point, boardMeetings and helpsOn for Officers and membMeetings and attends for Members. The same conditional navigation exists for the Staff Details portion of the system, with Caterer or Advisor as two possibilities stemming from Staff Details. From here, only one associative entity is possible for each one as consistent with the database schema. For caterers, the catersOn table must be updated, and for advisors the advisesOn table must me updated.

Requirements:

The implementation of this system uses Java for the user interface as well as SQL for the back end database application. The Java file for this project was written in Intellij IDEA where it was compiled and tested. The database was written using Postgres where the SQL script for the database is executed. This script must be executed in Postgres prior to running the Java application so that the database exists and stands ready to be updated with information when the user navigates the Java interface.

Literature Survey:

Many database applications exists today that users may choose to implement in their organizations. Popular applications like WordPress allow for everyday users to create databases based on the type of information they want to store. What makes my system unique is its emphasis on user-friendliness and the fact that it caters to a specific type of data management. My system has the purpose of providing an application to a unique niche, which not may not necessarily be appropriate for all types of management needs. Another popular application for data storage is MS Access, which is commonly used even in the professional workplace. In general, when a relational database in implemented in a platform whether its Postgres, MySQL, SQL Server, etc., there is a corresponding degree of control over the data, which has many positives and negatives. For a database developer, this degree of control is highly favorable, whereas for the common user, this may be overwhelming. These common applications (WordPress, Access) allow the common, everyday user to interact with the database, just as my system does.

User Manual:

1. At the prompt "You may identify an entity to add a record to in the database:" enter "Student", "Event", "Staff" or "Meeting" to choose which entity to add to the database. You may also choose to update either "Student Details" or "Staff Details". The system will accept both upper and lower case spellings of each input.

2. The system will prompt you to enter a valid “ID” based on the entity previously entered. This value must be in the form of an integer. If you fail to enter an integer, the system will prompt you to re-enter the value until a valid input data type is recorded.
3. From here, follow the system prompts to enter information based on the entity or details type outlined in Step 1. Note that entering the incorrect data type for any of the fields indicated at each prompt will inform you to re-enter until a valid entry is recorded.
4. Once all the information is correctly entered, the system will display the message that your entry has been successfully inserted into the database.

Conclusion:

Overall, the effectiveness of the system is illuminated with the use of a complicated database design implemented through a very user-friendly application. With a database schema and Java application being used in unison to allow this system to come to fruition, the system incorporates many different aspects of programming and systems design across two very relevant fields, database systems and software development. This system proves that a rather complicated system can be presented to be available to the everyday user, something that is evident in many different software applications in everyday life.