Haleigh Novak
Due: July 10, 2015
CS 496 - Project Three

Project: Three Part One

1. Data Description

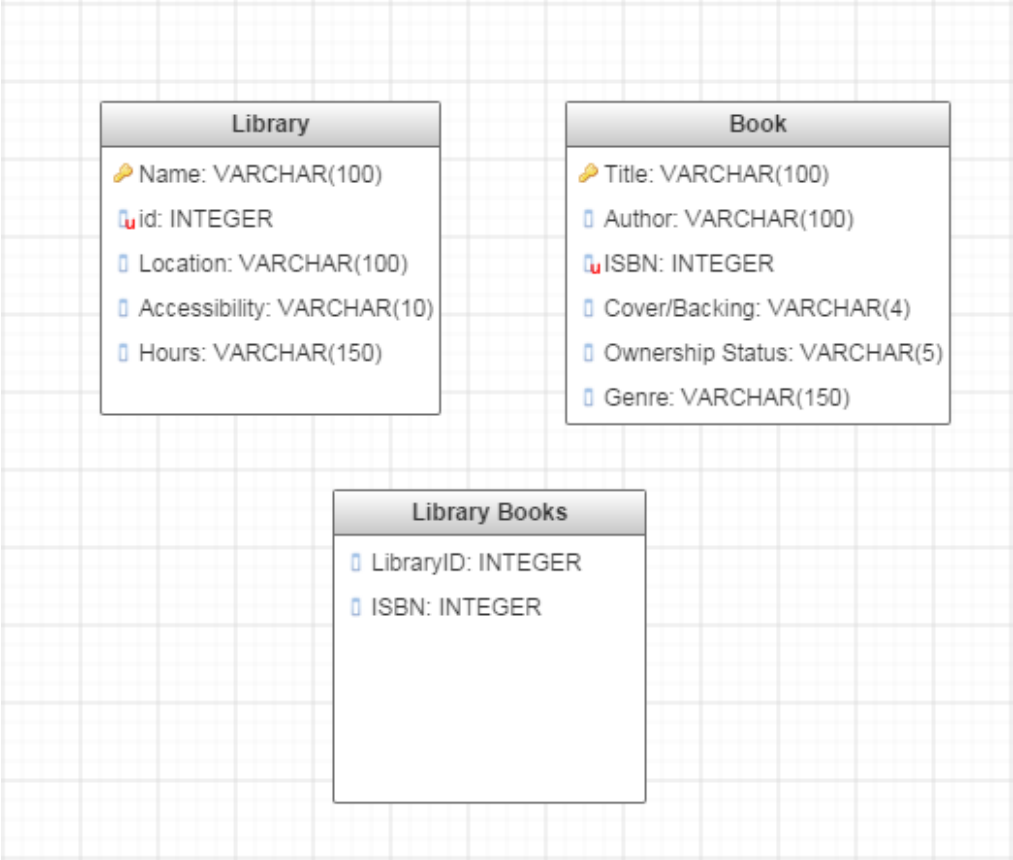
I am going to continue with my library theme, so the two database entities [and databases which have the same name] that I will model in this API are: 'Library' and 'Book' which will be related through 'Library Book'.

The "Library" database will contain 'Library' entities. While a 'Library' entity will contain the following properties: name, location (street address, city, state, zip), and accessibility (personal or public); all of which will be required. 'Library' will also have an optional property hours. 'Name' will be a string representing either the publicly known (Google-able) building name or for a personal library something that the user will remember. 'Location' will be a string address, preferably in the format of: "### Street City State Zipcode". 'Accessability' will be in the format of a radio button (where one option must be chosen), with "Personal" and "Public". 'Hours' will also be an optional string (more specifically for a public library than a personal one) to allow for user notification of the library's hours of operation and 'id' will be the unique numerical 'Library' identifier.

The "Book" database will contain 'Book' entities. While a 'Book' entity will contain the following properties: title, author, isbn, cover/backing (hard or soft), ownership status (owned or want), lib, and an optional genre (Mystery, Fiction , Fantasy, Horror, Sci-Fi, Non-Fiction, Textbook, Childrens, Teen, Adult, Romance, and Other). 'Title', and 'Author' will be strings representing the 'Book' title and author. 'ISBN' will be a unique numerical value representing the 11 or 13 digit international standard book number (typically located by the bar code that is scanned at purchase or checkout on the book). 'Cover/Backing' will be represented in the format of a radio button (where one option must be chosen) with the options being "Hard" [for a hard cover / leather bound book] and "Soft" [for a paperback book]. 'Ownership Status' will be represented in the same fashion with "Owned" representing a book currently in the 'Library' and "Want" representing a book that is needed / wanted in the 'Library'. 'Genre' will be optional and will have the general options of: Mystery, Fiction , Fantasy, Horror, Sci-Fi, Non-Fiction, Textbook, Childrens, Teen, Adult, Romance, and Other; with more than one capable of being selected.

Haleigh Novak
Due: July 10, 2015
CS 496 - Project Three

2. Data Model



"Library" and "Book" will both be databases, "Library Book" will be the connecting piece between the two of them. I do almost everything in the VARCHAR data type because it is user entered and for some fields there are limited choices; 'Accessibility' ("Public" or "Personnel"), 'Cover/Backing' ("Hard" or "Soft"), and 'Ownership Status' ("Owned" Want"); these are shown in the comments for the field. I did this because radio buttons and checkboxes while usable in my web application are unsupported as a database entry and because this will provide a simple way to poll/sort the database for these fields as well. 'ISBN' is a unique key for 'Book' because the ISBN is in the real world unique to the book, whereas 'Title' is the primary key because most people know the title of a book better than the ISBN. 'Genre' and 'Hours' are allowed to be NULL because they are optional fields for each database. 'Hours' is a VARCHAR data type as well because hours of operation are displayed differently and I find the most convenient to be something along the lines of: "Open Mon - Wed 9am to 5 pm" because this concisely states the hours in minimal textual room. 'id' in 'Library' is the unique key because for "Library Book" I needed a unique key from each of the other two tables ('ISBN' serves this purpose for 'Book'). This will allow for correct matching of 'Library' and 'Book' objects.

3. Comparison: My Non-Relational Database vs Other (Spreadsheet)

For this I turned to Google to find another good example of a non-relational database: I found one of the most popular results to be spreadsheets. Spreadsheets are actually very similar to what I have created here, there is one type of data ['Book'] that needs to be fully functional as is, but also be able to relate to another type of data ['Library'] (also fully functional alone) without creating dependencies. Within each implementation [spreadsheets and my own] there is one specific value taken from each type, which is used to relate the data wither mathematically or for viewing. In Excel the mathematical functions are something similar to " =AVERAGE(A1, B1, C1, ...) " thus indicating that the 1 column contains the significant/unique data necessary for relation; similarly in my implementation there are 'ISBN' and 'id', which each data type instance contains one or the other. These are related in yet another type 'Library Book', which in the afor mentioned example would be the cell that contains " =AVERAGE(A1, B1, C1, ...) ". The differences between the implementations are: [not including that mine is still in outline state] in Excel the cells are given a framework of numerically/alphabetically sequential names that are unenforced if required, whereas my implementation is given a basic framework of specifically (descriptively) named data entry positions which when required will be enforced; as well as that my implementation allows for only two data types to be related whereas Excel gives infinite relational possibilities. Though the latter difference is a matter of implementation, my implementation is focused on relating two types as per the assignment specifications, this could be easily modified to include more types of data. The similarities include: the relations of data use specific key values to correspond to various types of data and their 'lesser' values. Overall my implementation is really just a more descriptive, specific version of excel designed for two relatable data types.