Julian Savini: savini.j@northeastern.edu

Maxwell Rizzuto: rizzuto.m@northeastern.edu

Beckett Sanderson: sanderson.b@northeastern.edu

CSV Readers' Library

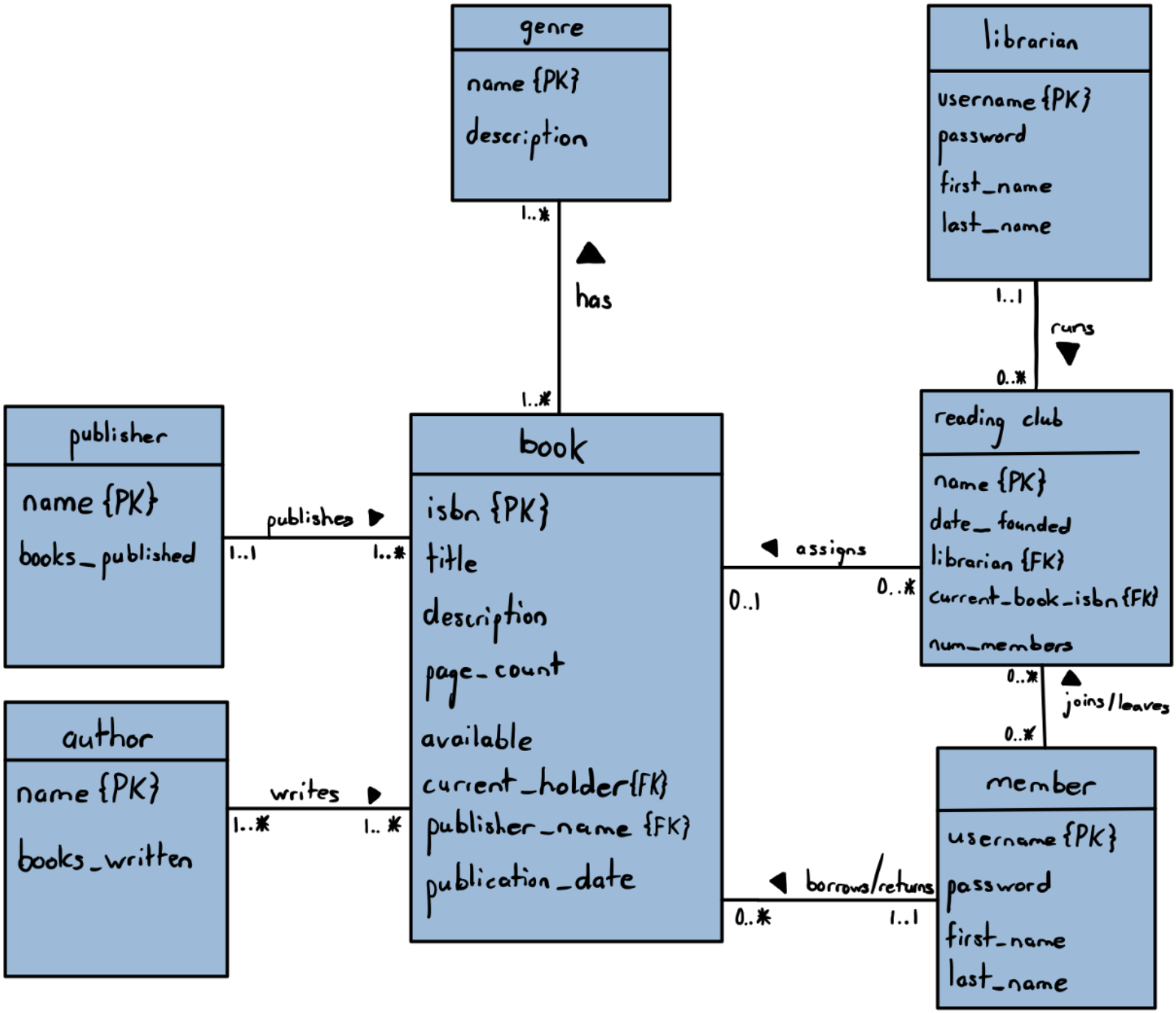
README Steps

1) Have Spyder (Python IDE) and mySQL workbench downloaded
2) Read in the mySQL dump file (library_db_dump.sql)
3) Open "library.py" in the Python interpreter of your choice
4) Change the host, username, password in lines 46-48 to match your connection
5) Run "library.py"
6) Explore library database and its features (see User Flow section for feature options) using some of the following data examples:
    a) Librarian (login)
        i) username: "testUser", password: "pass1234"
    b) Member (login)
        i) username: "testUser", password: "pass1234"
    c) Book
        i) ISBN: "9573860998237", Title: "A Study in Scarlet"
    d) Author
        i) Name: "Arthur Conan Doyle", Books Written: 2
    e) Reading Club
        i) Name: "Husky Book Club"

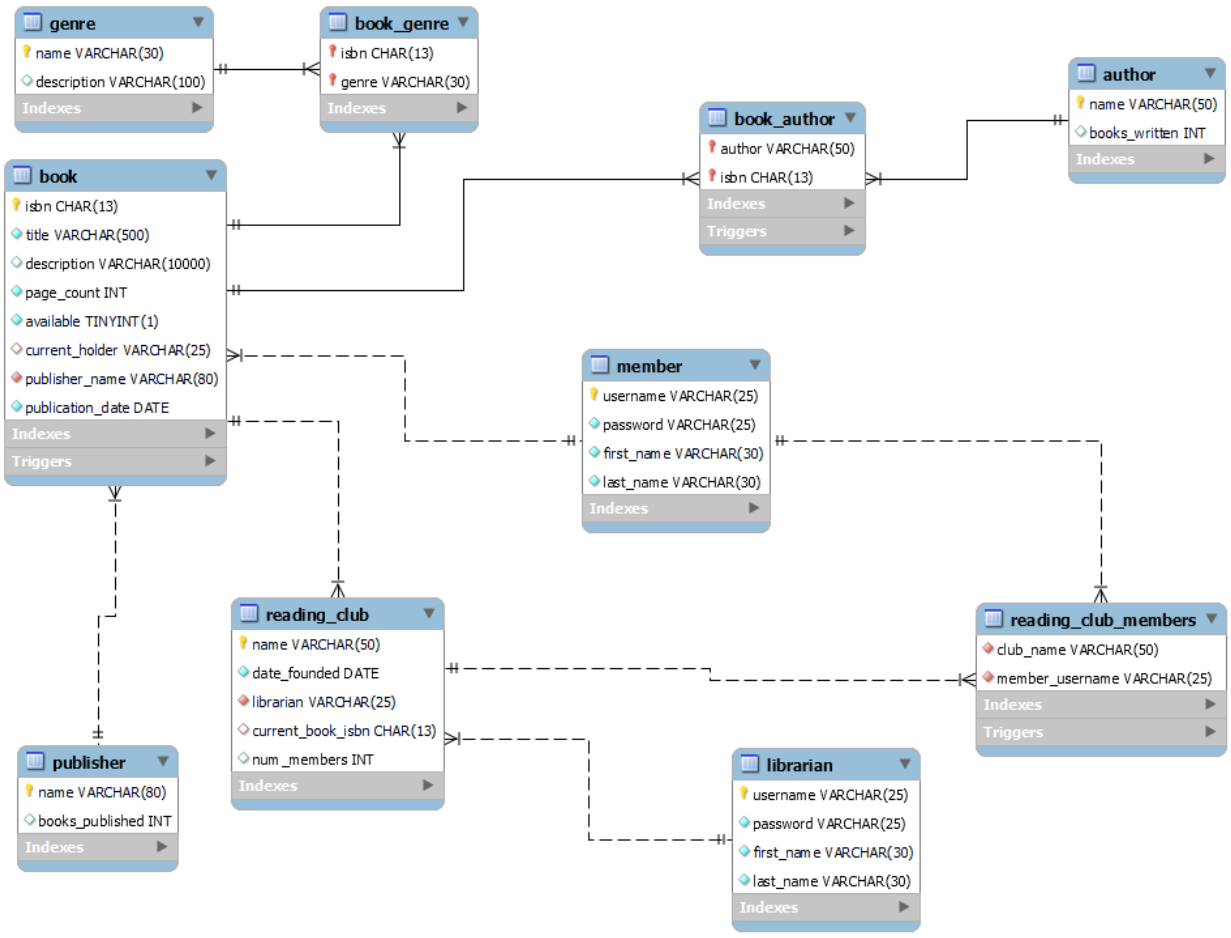
Technical Specifications

The application uses Spyder (a Python IDE) and mySQL. In Spyder, pymysql is downloaded so the IDE can connect to the database.

Conceptual Design



Logical Design



User Flow

Choose LIBRARIAN or MEMBER login

1. LIBRARIAN
    1.1. Enter USERNAME and PASSWORD until you enter valid login info
    1.2. Choose CREATE BOOK CLUB, MODIFY BOOK, ADD BOOK, REMOVE BOOK, ASSIGN NEW BOOK, or LOGOUT
        1.2.1. CREATE BOOK CLUB
            1.2.1.1. Enter the name of the book club to be created
            1.2.1.2. Enter the ISBN of the assigned book for the book club
        1.2.2. MODIFY BOOK
            1.2.2.1. Choose whether to ADD GENRE, REMOVE GENRE, CHANGE DESCRIPTION, or BACK
                1.2.2.1.1. ADD GENRE

1.2.2.1.1.1. Enter the ISBN of the book to modify
1.2.2.1.1.2. Enter the genre to add to the book
1.2.2.1.2. REMOVE GENRE
1.2.2.1.2.1. Enter the ISBN of the book to modify
1.2.2.1.2.2. Enter the genre to remove from the book
1.2.2.1.3. CHANGE DESCRIPTION
1.2.2.1.3.1. Enter the ISBN of the book to modify
1.2.2.1.3.2. Enter a new description for the input book
1.2.2.1.4. BACK
1.2.2.1.4.1. Returns back to 1.2 selection screen
1.2.3. ADD BOOK
1.2.3.1. Enter the ISBN for the new book
1.2.3.2. Enter the title for the new book
1.2.3.3. Enter the page count for the new book
1.2.3.4. Enter the author for the new book
1.2.3.5. Enter the genre for the new book
1.2.3.6. Enter the publisher for the new book
1.2.3.7. Enter the publication date for the new book
1.2.4. REMOVE BOOK
1.2.4.1. Enter the ISBN of the book to remove
1.2.5. ASSIGN NEW BOOK
1.2.5.1. Enter the name of the book club to assign the book to
1.2.5.2. Enter the ISBN of the new book for the club
1.2.6. LOGOUT
1.2.6.1. Program closes

2. MEMBER
2.1. Enter CREATE or LOGIN depending on if member already has a login with the library
2.1.1. CREATE
2.1.1.1. Enter the username for the new account
2.1.1.2. Enter the password for the new account
2.1.1.3. Enter the member's first name
2.1.1.4. Enter the member's last name
2.1.2. LOGIN
2.1.2.1. Enter USERNAME and PASSWORD until you enter valid login info
2.2. Choose BORROW BOOK, RETURN BOOK, FILTER BOOKS, JOIN BOOK CLUB, LEAVE BOOK CLUB, or LOGOUT
2.2.1. BORROW BOOK
2.2.1.1. Enter the ISBN of the book to borrow

2.2.2. RETURN BOOK
2.2.2.1. Enter the ISBN of the book to return
2.2.3. FILTER BOOKS
2.2.3.1. Choose whether to filter by PAGE COUNT, TITLE, GENRE, AUTHOR, or BACK
2.2.3.1.1. PAGE COUNT
2.2.3.1.1.1. Enter the maximum number of pages the book can have
2.2.3.1.1.2. Enter the minimum number of pages the book can have
2.2.3.1.2. TITLE
2.2.3.1.2.1. Enter a title to filter by
2.2.3.1.3. GENRE
2.2.3.1.3.1. Enter the genre to filter by
2.2.3.1.4. AUTHOR
2.2.3.1.4.1. Enter the author to filter by
2.2.3.1.5. BACK
2.2.3.1.5.1. Returns back to 2.2 selection screen
2.2.4. JOIN BOOK CLUB
2.2.4.1. Enter the name of the book club to join
2.2.5. LEAVE BOOK CLUB
2.2.5.1. Enter the name of the book club to leave
2.2.6. LOGOUT
2.2.6.1. Program closes

Lessons Learned

We built our own database in sql. We read a csv file to get the values of our database, leading us to learn the hard way how difficult it can be to read data into SQL that's obtained online. We learned how to use pymysql and perform CRUD operations in SQL using Python. We learned how to effectively filter inputs from the user. We created many procedures to filter through, append, delete data. We added triggers to update tables when other operations are run. Overall, we gained a deeper understanding of the concepts we covered in class. We gained a lot of practical experience in how to work with a database in Spyder.

One key insight we learned is that to create a database front end you need to be considerate of all possible user inputs. There are many potential ways a user can potentially falter inputting text, so we had to consider all the ways a user could make an error and keep the code as accommodating as possible and can redirect them if they fail to type something comprehensible. Our other insight was about how tedious creating options for a user can be. By the time we

reached the end of our front end code we had several layers of loops and if/else statements so a method to efficiently apply these layers is key.

We contemplated many methods of how we would want to get information from the user. We considered getting multiple inputs simultaneously, but ultimately I believe our method of retrieving information was the most sensible. Since we are receiving input from the user which is then stored as a variable, it is reasonable to ask for many singular statements, even if it isn't the most efficient. Given the constraints of the data, we did not have to really reconsider the design of our database.

All of the code in both Spyder and mySQL workbench run successfully.

Future Work

Since our project reads in a csv of google books, the planned uses of the database are very limited. If we wanted to read in other csv files containing information on a collection of books, we would either have to make sure the csv has the same features as the csv we used, or we would have to modify our backend and frontend code. However, if we were able to successfully complete either of those tasks, we could read in other libraries and have users complete their checkout process via our application.

There are a couple areas we could improve to add functionality. Although none of us have experience in this field, web design could greatly benefit our application. We have very tedious code which has to request the user multiple statements. However, if we created a website, we could have the user input their username and password simultaneously, have them select a tab for the action they would like to commit, and get their final input after they selected said tab. Another area we would have liked to work more on is error prevention. Currently, we need a user to input a certain statement in order for a procedure to run. We would like the program to be more forgiving if a user types something relatively close to what is being asked. This could be done by using web design and utilizing some sort of button or tab system. Finally, we could improve functionality by modifying how we filter by title. It would be optimal functionality if the user could type a title similar to a book, and still receive that as a suggested book in the output.