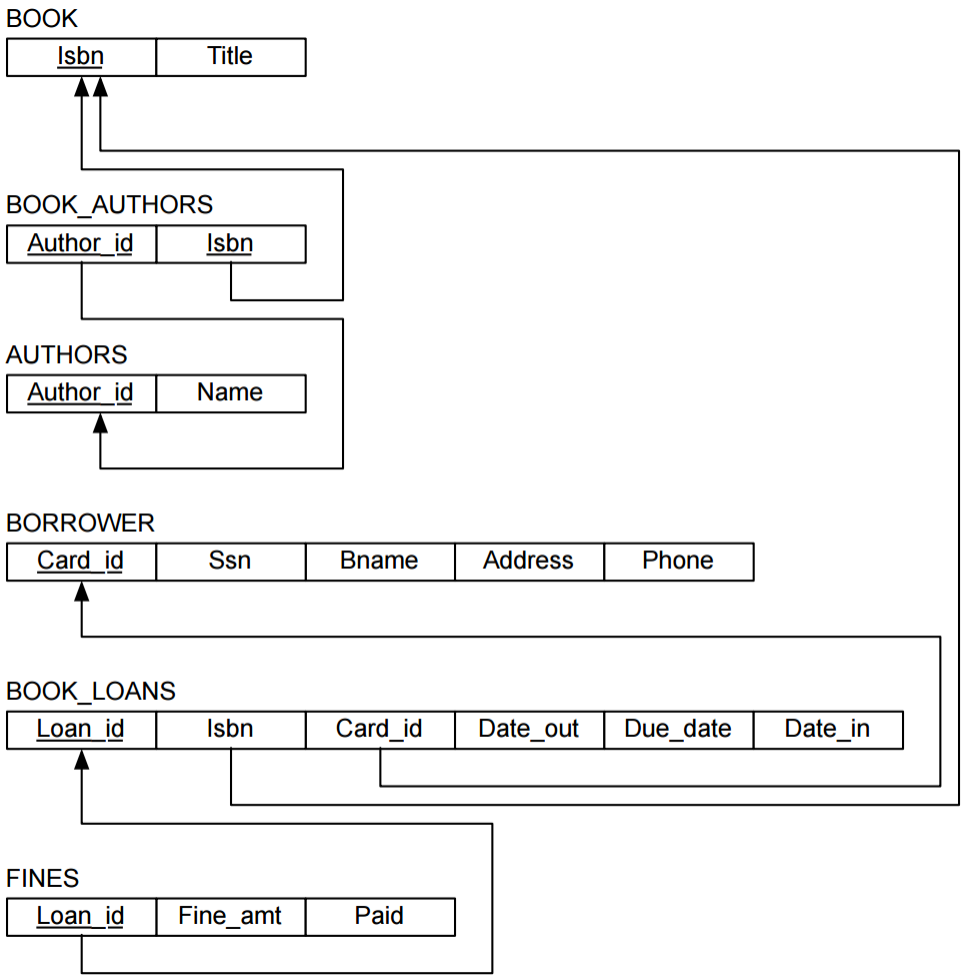
DATABASE DESIGN
CS6360.003
Project#1

Library Management System

Rammurthy Mudimadugula
RXM163730

Introduction:
A Library Management System is a system where a user can checkin/checkout, check availability of books and pay fines for late checkin of books. Users of the system are understood to be librarians (not book borrowers).

Schema:



The schema for the library database is derived from the schema provided in the project description. In addition to the above fields additional fields are added to implement certain functionalities. Following are the additional columns which are added:

1. BOOK Table: A ‘checkedout’ column is added to track whether a book is available or not for the checkout. Additional fields (like pages, title, publisher etc.,) which are obtained from the data provided are also added.

2. BORROWER Table: Additional fields (like state, email etc.,) obtained from the data are added.

Assumptions:
Following assumptions are made in design of this Library Management System.

1. For the search functionality, I have assumed that when a user provides ISBN for a book the remaining fields are discarded in search query. Only title and author are included for the search for partial input. We can search by either title or author or both title and author. Partial string matching is implemented for providing efficient search results. For search, at least one field is expected. If no search criteria is provided, appropriate error message is shown to the user.
2. Payment of fines is assumed to be full. No partial payment of fines is accepted.
3. Book ids are taken from ISBN10 field provided in the data (although 13 digit ISBN is also provided).
4. A user can checkout only maximum of three books at a time.
5. After due date a fine of $0.25 is added per day for late checkins.

System Architecture:
To implement this system, I have used Web Architecture. I have written RESTFul APIs for the implementation of the individual functionalities. I have also developed a web GUI for the ease of use for the librarian.

I have used MySQL as my persistent database storage. REST APIs are developed in NodeJS using a driver interface to connect to MySQL server. I have used Bootstrap to develop a UI to make api calls through browser.

To insert the data into the database, firstly I have created a database in the MySQL. I have wrote DB Migrations to create tables in the database. I have used ‘csvtojson’ NPM module to parse the data given. I have then inserted this parsed data into the appropriate tables.

I have used HAPI framework to create a server. I have then defined routes to individual feature that needs to be implemented. For implementing search, I’m passing the search parameters as query params. These query parameters are then used to perform search on the database. Partial input is accepted and appropriate string matching is implemented to provide useful search results.

For checkin and checkout of a book, the feature is implemented as a HTTP POST request. The borrower card number and the isbn of the book are passed as request body so that these attributes can be accessed on the server side to make appropriate changes to the database. An attribute ‘checkedout’ is added to the books table so that whenever a book is checked in or checked out, this attribute is changed appropriately to indicate the availability of the book. A success message or appropriate error message is shown to the user as a result of this request.

Adding a new borrower to the system is implemented as a HTTP POST request. All the required details to insert a new borrower is passed in the request body. After inserting borrower to the system, borrower id (library card number) is given as response to the request.

For the updating the fines daily, I have provided a HTTP POST request. I have provided a button in the UI which fires this api call so that all fines will be updated. This api call also checks whether a new record in book loans table is past due date and inserts a new record the fines table. Payment of fine is PUT request which just flips a column which indicates that the fines are paid. In addition to this, I have also provided an API call to get total outstanding fines for a borrower. If no fines exist, $0 is provided as the response.