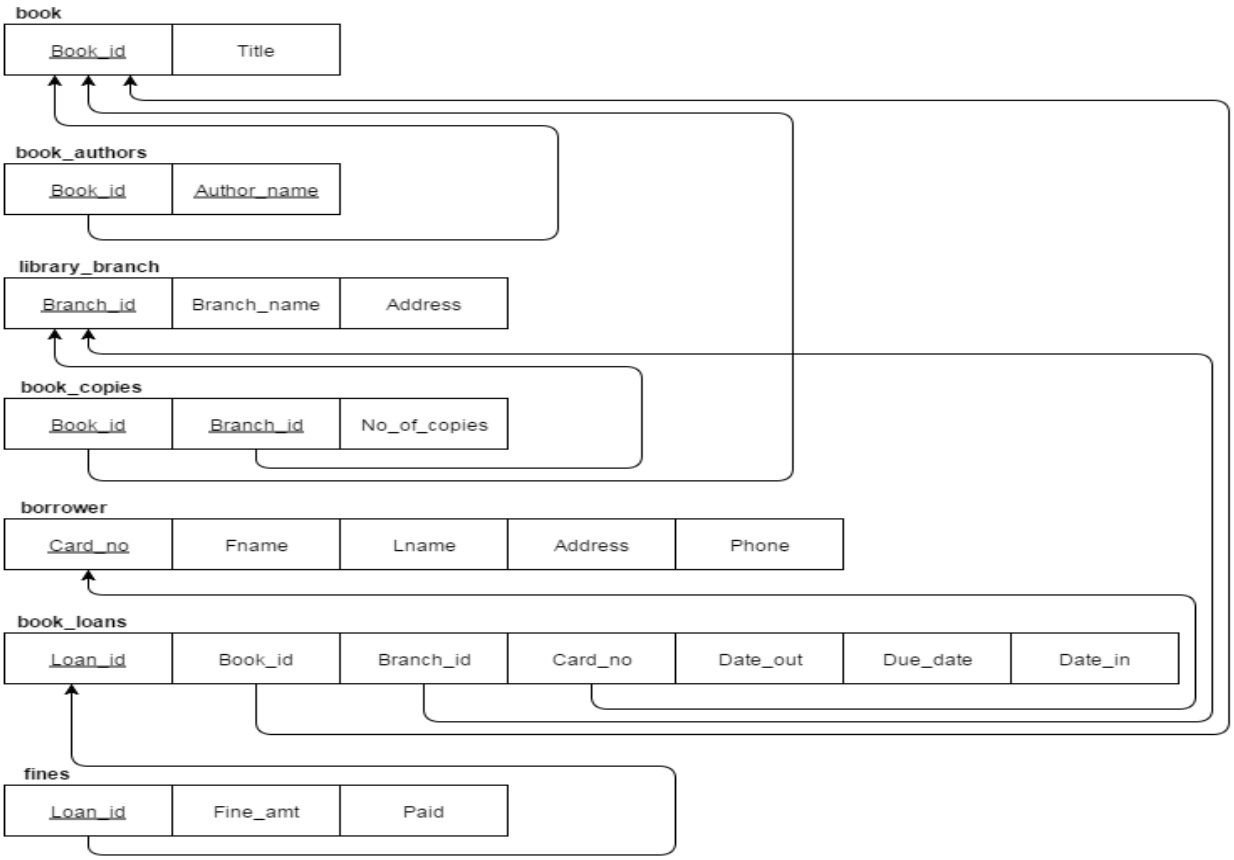
Description

1. Languages and Frameworks

   I used To build your host application you may implement either a native GUI application or a web interface. My applications are programmed with Python with framework Flask. I used MySQL Server as my database. And I use html to design a web interface.

2. Schema

   The schema is shown below:



3. Data

   There are 5 csv data for this project, they are:
   book_copies.csv;
   book_loans.csv;
   books3.csv
   borrowers.csv
   library_branch.csv

4. Function:

1) GUI

I designed a webpage interface with the Library with html. Initial database creation and population could be done from command line or other admin tool.

2) Book Search and Availability

My application is able to search for a book, given any combination of book_id, title, and/or Author(s) with the display: book_id; title; author(s); branch_id; How many copies are owned by a specified branch; Book availability at each branch (i.e. How many copies not already checked out?).

3) Book Loans

a. Checking Out Books

My GUI is able to check out a book, given the combination of BOOK_COPIES(book_id, branch_id) and BORROWER(Card_no), and to generate a new unique primary key for loan_id. The date_out is today's date. The due_date is 14 days after the date_out.

Each BORROWER is permitted a maximum of 3 BOOK_LOANS. If a BORROWER already has 3 BOOK_LOANS, then the checkout will fail and return a useful error message. And if the number of BOOK_LOANS for a given book at a branch already equals the No_of_copies, then the checkout will fail and return a useful error message.

b. Checking In Books

My GUI is able to check in a book and to locate BOOK_LOANS tuples by searching on any of book_id, Card_no, and/or any part of BORROWER name. Once located, it can select one of potentially multiple results and a button to check in.

4) Borrower Management

My GUI is able to create new borrowers in the system with all name and address attributes are required to create a new account. It can automatically generate new card_no primary keys for each new tuple that uses a compatible format with the existing borrower IDs. Besides Borrowers are allowed to possess exactly one library card. If a new borrower is attempted withe same fname, lname, and address, then my system will reject and return a useful error message.

5) FINES

I created a new table FINES with the primary key loan_id is also a foreign key that references BOOK_LOANS(loan_id), the fine_amt attribute which is a dollar amount, the paid attribute which is an integer 0/1
that indicates whether a fine has been paid, and the est_pay attribute which indicates the money that the estimate fine of books that are not returned. The Fines are assessed at a rate of $0.25/day.
There are two scenarios for late books:
a. Late books that have been returned — the fine will be [(the difference in days between the due_date and date_in) * $0.25].
b. Late book that are still out — the estimated fine will be [(the difference between the due_date and TODAY) * $0.25].
If a row already exists in FINES for a particular late BOOK_LOANS record, then
a. If paid ==0, do not create a new row, only update the fine_amt if different than current value.
b. If paid == 1, do nothing.
Finally, I provided a mechanism for librarians to enter payment of fines, and do not allow payment of a fine for books that are not yet returned.
My GUI could display the Fines should be grouped by card_no and the fines should provide a mechanism to filter out previously paid fines (either by default or choice).

The running instruction is in the readme file.

Gaoyang Ye
Net-id: gxy140830