SQL Project

Project Description and Requirements

This SQL programming project involves the creation of a database host application that interfaces with a SQL database that implements a Library Management System. Assume that the users of the system are librarians (not book borrowers).

This is an individual (not group) project. All work, design, and coding must be done by you.

Due Date: April 22, 11:59pm

Programming Language(s) and Frameworks

You may use either Java or C# to build your host application. You may use either MySQL or Microsoft SQL Server as your database. If you would like to use any other languge you must obtain approval from the TA that they are able to evaluate the language of your submission.

Schema

The schema for the library database is derived from (but NOT the same as) the library schema in the textbook (p.113 Figure 4.6). The actual schema used for this project is provided in this folder. You are permitted to modify or augment this schema provided that your system adheres to the written requirements. As long as it supports the documented functionality, you may add any features you deem useful.

Data

- Baseline data to initialize your database is provided in this folder. There is no initial data in the BOOK_LOANS table.
- The use cases executed for grading will be based upon this data.
- Data is provided in plain text (ascii) files. It is your task to map these onto the schema and tables, i.e. there is not one "load file" per table.

Submission

You will be required to submit the following files:

- All application source code, including any build files (e.g. make, ant, maven, etc.)
- An SQL schema creation file that includes appropriate all appropriate CREATE TABLE commands and (potentially) ALTER TABLE commands and/or CREATE TRIGGER commands to implement any integrity constraints.
- Table data import file(s) that include includes either (a) a sequence of INSERT INTO commands, or (b) A file will load directives and the .dat on which it operates
- A 1-2 page written description of your system that includes (a) user instructions for librarians, (b) design decisions and justifications, (c) technical dependencies (software libraries, software versions, etc.).
- All files must be zipped together into a single file. This file should be named ***<net-id>*_cs6360S14_002.zip**, where ***<net-id>*** is your Net ID.
  Example: **cid021000_cs6360S14_002.zip**

Grading

- You will be required to demonstrate your application for the TA. If you are unable to bring a laptop computer to demonstrate your app, please let me know as soon as possible so I can make alternate arrangements.
- Each student will have 10-15 minutes to demonstrate their system and execute the test cases provided at grading time.
- A sign-up mechanism will be made available to reserve a specific time for evaluation. As a courtesy to the TAs and those waiting behind you, **please be on time for your scheduled slot**.

Notice: Requirements will change before the due date. One objective of this project is the ability to accomodate requirement changes after beginning a database project. There will be at least one schema change, one functional requirement change, and one data change.

The follow are a summary of your functional requirements:

1) GUI [25 points]

All interface with the Library (queries, updates, deletes, etc.) must be done from a graphical user interface of your original design. Your GUI application will interface with the Library database via an appropriate MySQL connector. Initial database creation and population may be done from command line or other admin tool.

2) Book Search and Availability [25 points]

Using your GUI, be able to search for a book, given any combination of **book_id**, **title**, and/or **author_name**. Your query should support substring matching. You should then display:

- book_id
- title
- author(s)
- branch_id
- Number of copies at each branch
- Number of available copies at each branch

Hint: For Available number, you must subtract the number of active BOOK_LOANS(i.e. COUNT(*)) from **No_of_copies** for a branch.

3) Book Loans [25 points]

Checking Out Books

- Using your GUI, be able to check out a book, given the combination of BOOK_COPIES(**book_id**, **branch_id**) and BORROWER(**Card_no**), i.e. create a new tuple in BOOK_LOANS. The **date_out** should be today's date. The **due_date** should be 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 (i.e. create new BOOK_LOANS tuple) should *fail* and return a useful error message.
- If the number of BOOK_LOANS for a given book at a branch already equals the No_of_copies (i.e. There are no more book copies available at your library_branch), then the checkout should *fail* and return a useful error message.

Checking In Books

- Using your GUI, be able to check in a book. Be able to locate BOOK_LOANS tuples by searching on any of book_id, Card_no, and/or any part of BORROWER name. Once located, provide a way of selecting one of potentially multiple results and a button (or menu item) to check in (i.e. enter a value for **date_in** in corresponding BOOK_LOANS tuple).

4) Borrower Management [25 points]

- Using your GUI, be able to create new borrowers in the system.
- All name and address attributes are required to create a new account (i.e. value must be not null).
- You must devise a way to automatically generate new **card_no** primary keys for each new tuple that uses a compatible format with the existing borrower IDs.
- Borrowers are allowed to possess exactly one library card. If a new borrower is attempted withe same fname, lname, and address, then your system should *reject* and return a useful error message.

MySQL + Java Resources

Java Connector Download

https://dev.mysql.com/downloads/connector/j/

java.sql.* API documentation

http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html

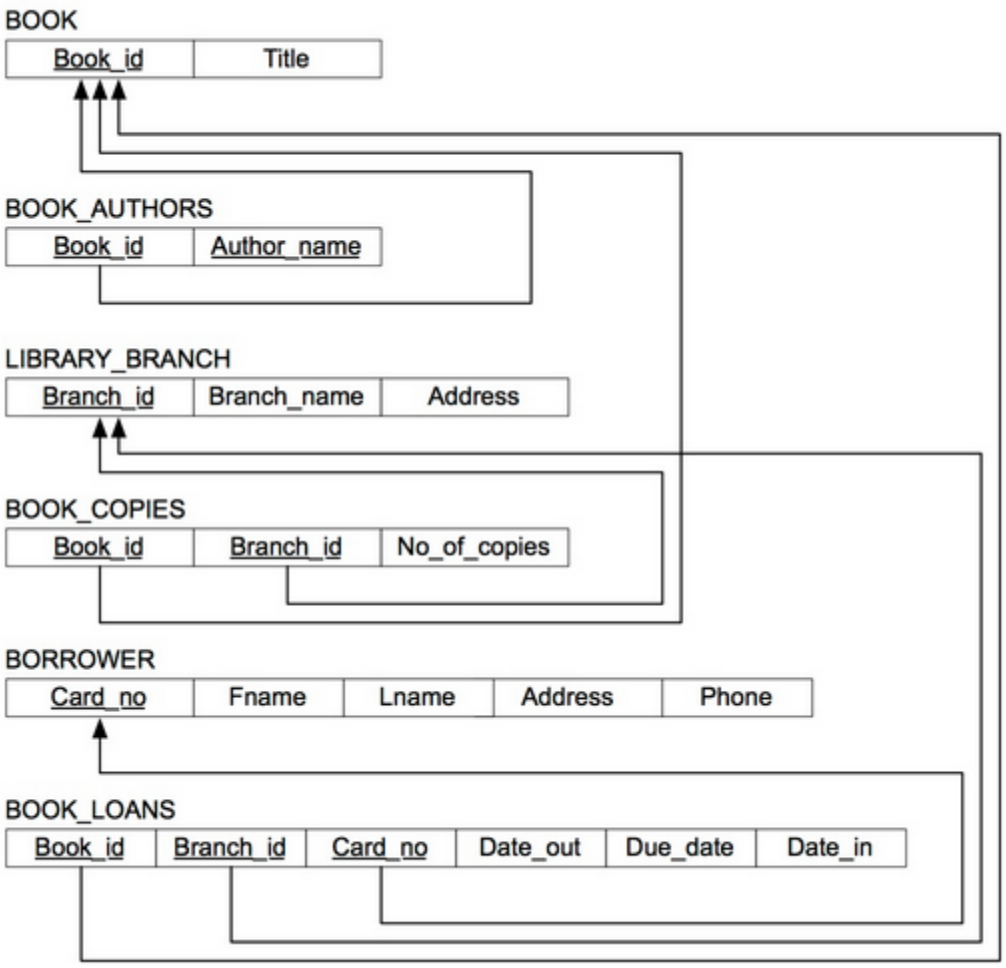
Data Files

Library_data.zip

This file contains ascii format, tab delimited csv data.

- book_copies.csv
- books_authors.csv
- library_branch.csv
- borrowers.csv
-

Database Schema



- 

  Java SQL stub code example

  Example stub code used in class lecture

  DBTest.java