Library Management System Architecture

Design Report

To the

The University of Texas at Dallas

Submitted by:

Shobhika Panda

Net Id-**sxp150031**

The University of Texas at Dallas

Dallas – 75252

March, 2016

1. Design Decisions

1.1 Technologies used

a. **Node.js** is used to develop server side web applications. If you are not using Node.js then you are continuously shuttling JSON back and forth. That is the reason why I chose Node.js, which helps you to use Javascript and jQuery throughout your application. It helps re-use my model, reduce the chance of bugs, complexity of code and size of the application.

b. **MYSQL** because it is open source and highly scalable. It is much lighter than Oracle.

c. **Bootstrap** is the framework used for front-end designing. This framework can be used on all devices, therefore it is quite scalable and easy to use.

d. **JAVA JDBC** to clean up data and put it into data tables from csv files using Hashing.

1.2 Data Clean Up

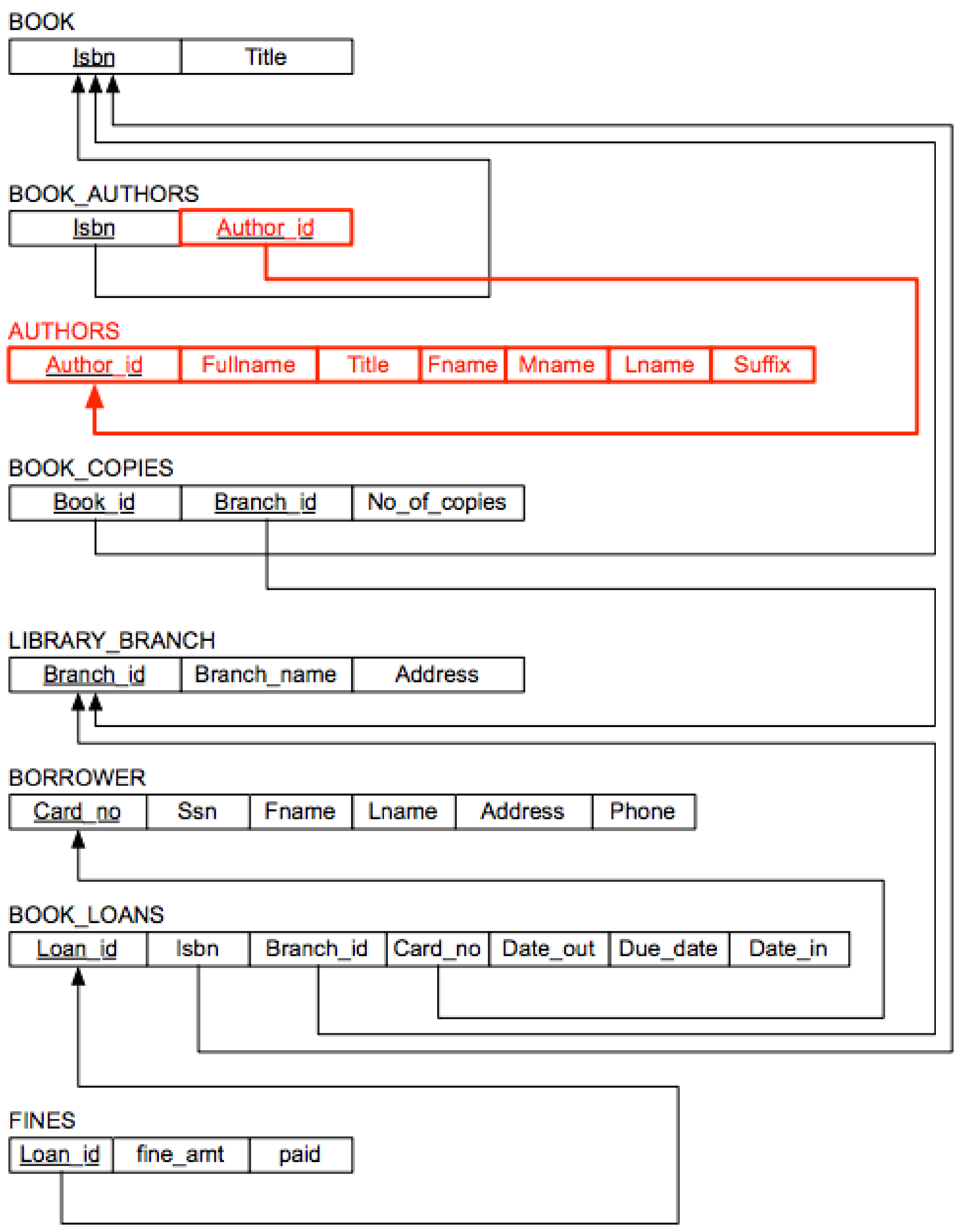
a. In the CreateTables.java, I have inserted data into the three tables book, book_authors and authors based on my hashing logic.

b. I created three hashmap namely <bookmap>, <authormap> and <bookauthor>. I split by tab and read booksnew.csv using java buffered reader. I put ISBN10 as key and title as value into <bookmap>. For creating <authormap>, there will be multiple authors for a ISBN so I split them by comma and put authorname into the hashmap <authormap> as key and auto-increment index as value<author_id> which would ultimately be the author_id in the table. In the <bookauthor> map I have taken ISBN as key and author full name as value. So that based on each author's name, author's id will be generated and stored in the book_authors and authors table. I maintained a prefix and suffix list to store it as author's title in the authors table. Based on keyset values I inserted all the rows into book, authors and book_authors table. The full name of authors was split into title, first name and middle name based on comma separated values and put into a string array.

c. Created and Inserted other tables using JDBC prepared statements and csv reader.

1.3 Schema

a. I used the schema that was provided by the instructor.

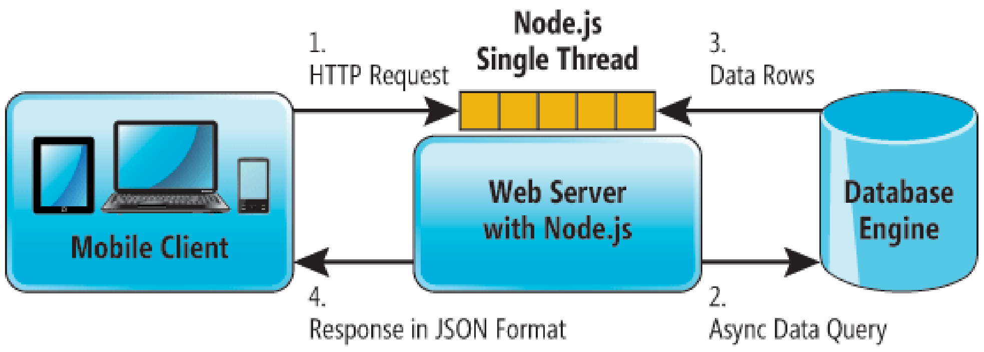


1.4 Directory Structure, Server Modules

a. Node/node_modules/body_parser,cookie_parser,express,multer,mysql
b. Node/routes/dbconnect.js,dbaddBorrower.js,dbCheckin,dbCalculateFine.js...etc
c. Node/views/index.html
d. Node/public/home.css
e. Package.JSON
f. Server.js

1.5 Node Server Design

a. Back-end is independent RESTFUL API following MVC design pattern.
b. I have one controller Server.js which handles all requests.
c. For every functionality, there is one borrower model and for Check-in/Check-out there is another example dbCalculateFine.js, dbAddBorrower.js, dbCheckin.js.
d. It is event-model based rather than thread model. It handles multiple requests and Input/Output operations asynchronously.



1.6 Assumptions

a. OnClick of Check-in/ Check-out/Calculate Fines/Pay buttons will update the database in the backend and show an alert message.
b. All Show/Search Buttons will produce the output in the front end in the form of data tables.

2. References

- E-learning notes