Group 114 - Project Step 4 Draft

Team: Yash Sankanagouda, Jim Landers
Project: Ebook Library Database

Website Link Here
(Must be connected to OSU network to view)

Feedback by the peer reviewers:
Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Yes, it appears the tables match the schema and sample data. They are present in the UI.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

There doesn't appear to be a filter (you could easily add one for employees or customer names just to sort them via name).

Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.

Every table has the opportunity to add data, thus filling the INSERT need.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

Yes, the checkout has an insertion for M:M with foreign keys from Books, Employee, Member, etc.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Yes, there are DELETE statements for each table.

Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Members, Employee, and Checkout all have editable (UPDATE) entries.

Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

There doesn't appear to be a nullable FK relationship? Might be wrong here? But you could do like a self-checkout as NULL employee!

Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.

You might want to add a title so it just doesn't say "Bootstrap Demo" on the tab. Looks really solid.

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Each table has a page.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

There are no apparent filters or searches. There is maybe an opportunity for dorp-downs in Books for Author and in Chekout for Book and Author?

Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.

Every table has an INSERT statement.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

Yes, there is insertion to the Checkout intersection table which includes the FK for Books, Memebrs and Employyes.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete

the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

There are Delete statements/UI component in all of the tables.

Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Employees and Members have update/edit statements/UI components.

Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

There are no apparent Nullable FK relationships.

Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.

AuthorID in Books, Member Id and Employee Id in Checkouts should not be filed for the user to enter the PK, but to select it somehow. Other than that, I like the use of Bootstrap! (The title property of the website should be set to your project's name).

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

The UI shows data for every table on its HTML table page. In the DML, there are select queries written for each of these tables.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

No, none of the select queries utilize a search or filter. Authors and Books are good candidates for this type of search/filter to implement.

Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.

The UI displays an insert/add option for for every table on its HTML table page.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

It appears that you are unable to add the Author_id FK when inserting a new book. I see no other way for this FK to be added.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

There are two tables with a delete feature and one of them is on the intersection table. However, upon clicking on the delete button there is no confirmation pop up or anything to confirm that the feature is set up.

Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Yes, there are two tables with an update feature. However, the button does not bring up any kind of update form for the user to interact with so there is no way to confirm that the feature is set up.

Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

None of the relationships are nullable. Perhaps you could make the employee id nullable for checking out books.

Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.

The title that appears in the browser tab currently says 'Bootstrap Demo'. I would recommend changing this to match the project title or the name of each page you are on.

○ ·Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

○

○ o The UI does utilize a select for each table in the schema, assuming that the tables displayed will be implemented with SELECT later on.

○

○ ·Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?

○

○ o It does not look like a search or a filter with a dynamically populated list will be implemented. Maybe add a search function to the books and members table in the actual implementation?

○

○ ·Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.

○

○ o Insert is implemented on every table.

○

○ ·Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

○

○ o Inserting at the checkouts table should function properly given the FKs.

○

○ ·Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

○

○ o The DELETE if correctly implemented should remove things from an M:M relationship (the checkouts conjunction table) correctly.

○

○ ·Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

○

○ o The edit button provides for the ability to update.

○

○ ·Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.
○
○ o According to the ERD and the PDF, there is no NULLable relationship. Maybe try to make employee_id in checkouts to optional.
○
○ ·Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.
○
○ o The UI looks really clean. Two things for pointers: add the search functionality to books and make employees in checkouts to a NULLable entity.

T.A Feedback:

Good work group 114, but your DML is the same as your DDL. Your DML needs to include queries such as insert/select/update/delete

---

Actions based on the feedback and updates to the draft version:

- Implement a search/filter feature on the website
- Update books to properly reflect a NULLable relationship. (This was already implemented, but some parts of this document needed to be updated to reflect it.)
- Implement the update/delete feature on the website for the table, this should delete all the appropriate values (including FKs)
- Change the title from ‘bootstrap demo’ to Ebook Library
- Make sure the DML includes queries such as insert/select/update/delete

1. **What technology are you using (e.g. NodeJS/Flask Python)**
   We are using NodeJS.
2. **What works**
   The website (should) have a functioning select, insert, update, and delete operation implemented into the website.
3. **What doesn’t work**
   Not all queries and code has been written for every operation which needs to be functional so far.
4. **Where/why you are blocked**
   No hard blockers, just need to spend more time on the project currently.

Overview:
Our project will be a front end web application connected to a NodeJS server with a PostgreSQL database. The web application is meant to only be used by administrators, so it will have full access to view data held inside each table existing inside the database.

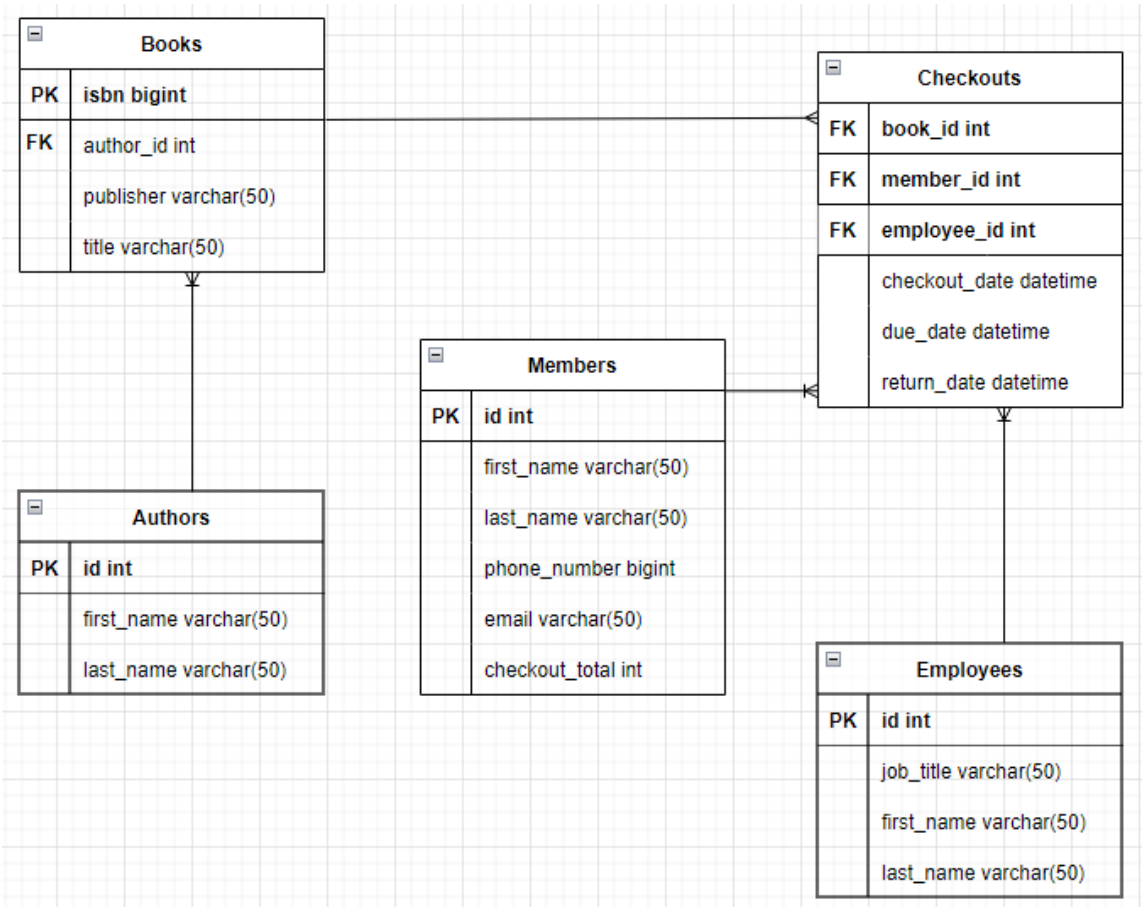
This library only checks out digital ebooks, which means there is no overhead of keeping track of multiple copies of the same book. However, certain county laws and regulations require members to physically visit the building in order to have an employee verify their membership and check out the book onto their device. The library checks out hundreds of books per month (400 total) to its 200 members with 20 employees working. The library checks out ebooks with a copy which is set to expire after one month. The library needs a more efficient database to see which members have currently outstanding ebook loans on their accounts and to better track their ebooks.

Database Outline:

1. Books
    a. isbn: bigint, not null, primary_key
    b. title: varchar, not null, primary_key
    c. author: int, foreign_key (from authors.id)
    d. publisher: varchar, not null
    e. Relationship: a 1:M relationship with checkouts. A digital book can be checked out many times.
2. Authors
    a. id: int, auto_increment, not null, primary_key
    b. first_name: varchar, not null
    c. last_name: varchar, not null
    d. Relationship: a 1:M relationship with Books where an author can have written at least one or multiple book IPs.
3. Members
    a. id: int, auto_increment, not null, primary_key
    b. first_name: varchar, not null
    c. last_name: varchar, not null
    d. phone_number: bigint, not null
    e. email: varchar, not null
    f. checkout_total: int, not null
4. Employees

    a. id: int, auto_increment, not null, primary_key
    b. job_title: varchar, not null
    c. first_name: varchar, not null
    d. last_name: varchar, not null
    e. Relationship: a 1:M relationship with checkouts. An employee can have multiple checkouts authorized.
5. Checkouts (Intersection table)
    a. book_isbn: int, not null, foreign_key (from books.isbn)
    b. member_id: int, not null, foreign_key(from members.id)
    c. employee_id: int, not null, foreign_key(from locations.id)
    d. checkout_date: datetime, not null
    e. due_date: datetime, not null
    f. return_date: datetime, not null
    g. Relationship: This table acts as a M:M intersection between books and members. A book can be checked out to one or many users, and a user can check out one or many books. Another M:M relationship is depicted here in how an employee can authorize the checkout of many books to a member, but also a member could have their books checked out by many different employees. This relationship would look like a very tangled web if it were to be drawn out.

Entity-Relationship Diagram:



Sample Data

Authors		
id	first_name	last_name
1	J. K.	Rowling
2	J. R. R.	Tolkien
3	J. D.	Salinger
4	E. B.	White

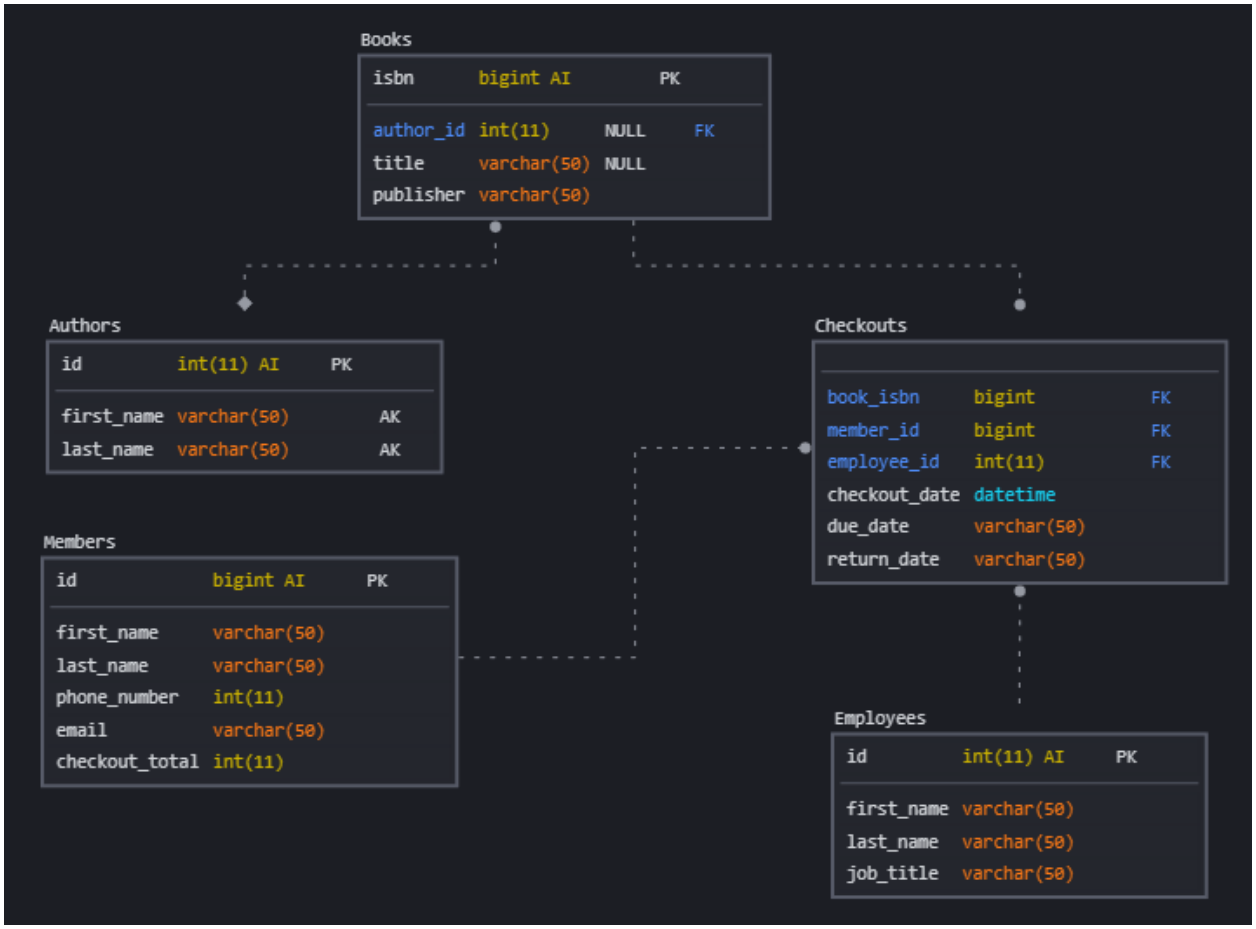
Books			
isbn	author_id	title	publisher
9780747532699	1	Harry Potter and the Philosopher's Stone	Bloomsbury
9780439064873	1	Harry Potter and the Chamber of Secrets	Bloomsbury
9780547928227	2	The Hobbit	Houghton Mifflin Harcourt
9787543321724	3	The Catcher in the Rye	Little, Brown and Company
9780062658753	4	Charlotte's Web	Harper & Brothers

Members					
id	first_name	last_name	phone_number	email	checkout_total
1	John	Smith	5031231234	jsmith@gmail.com	0
2	John	Carter	1231231234	johncarter@gmail.com	0
3	Eliza	Smith	8946548426	esmith@gmail.com	0
4	George	Washington	7913455469	presidentgeorge@gmail.com	0
5	Abraham	Lincoln	9711223000	abelincoln@gmail.com	0

Employees			
id	first_name	last_name	job_title
1	George	Allenson	Janitor
2	Paul	Robinson	Director
3	Thomas	Song	IT Director
4	Lea	Irving	Librarian
5	Bob	Trock	Librarian

Checkouts					
book_isbn	member_id	employee_id	checkout_date	due_date	return_data
9780547928227	1	1	2/16/2023	3/2/2023	2/17/2023
9780747532699	1	1	2/16/2023	3/2/2023	2/17/2023
9780062658753	2	4	2/16/2023	3/2/2023	2/17/2023
9780062658753	1	5	2/16/2023	3/2/2023	2/17/2023
9780062658753	3	1	2/16/2023	3/2/2023	2/17/2023

Schema


Books
isbn bigint AI PK
author_id int(11) NULL FK
title varchar(50) NULL
publisher varchar(50)
Authors
id int(11) AI PK
first_name varchar(50) AK
last_name varchar(50) AK
Checkouts
book_isbn bigint FK
member_id bigint FK
employee_id int(11) FK
checkout_date datetime
due_date varchar(50)
return_date varchar(50)
Members
id bigint AI PK
first_name varchar(50)
last_name varchar(50)
phone_number int(11)
email varchar(50)
checkout_total int(11)
Employees
id int(11) AI PK
first_name varchar(50)
last_name varchar(50)
job_title varchar(50)