Bookstore Management Database Project Report

4/28/24

CS 4347.002 Database Systems

Rishika Narala, Thompson Pham, Omar Hussain, Manvir Chakal

Prepared for:

Professor Jalal Omer

Table of Contents

1. Introduction

In the evolving landscape of retail, the effectiveness of inventory management significantly impacts business success. For a bookstore where inventory ranges widely in genres, volumes, authors, price, and quantity available, managing such diversity is difficult. This project introduces a database system to streamline operations in a bookstore. The system aims to provide real-time updates on inventory, facilitate order processing, and enhance customer and employee ease-of-use.

Problem Statement

The primary challenge addressed by this database system is the real-time management of inventory in a bookstore. Without a centralized database, it is cumbersome for both employees and customers to ascertain the availability of specific books, leading to missed sales opportunities and customer dissatisfaction. As the bookstore expands in inventory, maintaining data consistency and concurrency becomes increasingly complex. The proposed database system leverages primary and foreign keys to ensure data accuracy and reduce redundancy.

Target Users

The database system caters to several user groups, each with specific roles and needs:

Customers: They can access the database via the bookstore's online platform to browse available books, place orders, and check book availability. This direct access enhances the shopping experience by providing customers with up-to-date information on the bookstore's offerings.

Employees: This group is responsible for the day-to-day management of the database. Their tasks include updating inventory, processing orders, and maintaining accurate product information.

Report Overview

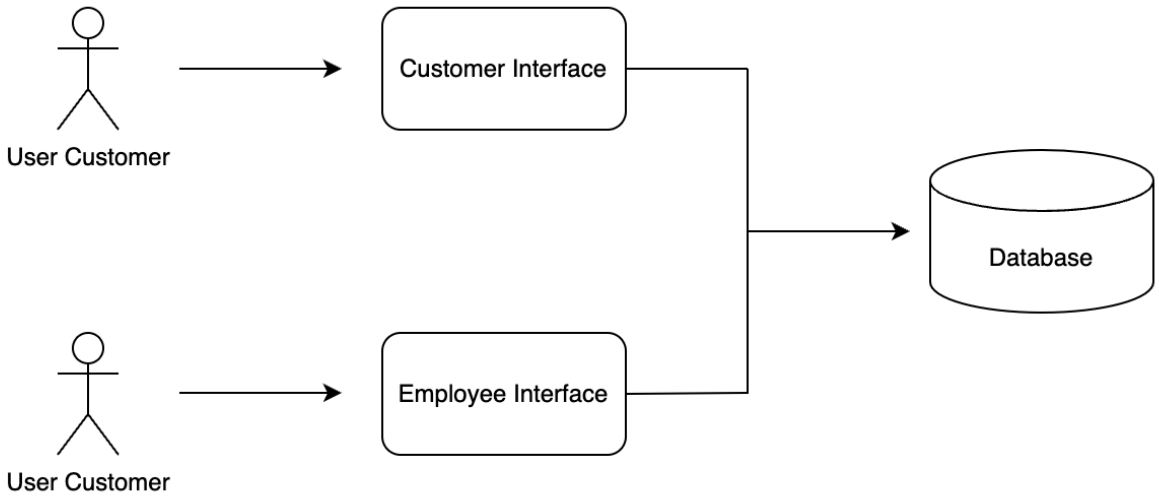
During this report we will go through eight sections:

- System requirements: Describes the system architecture through a context diagram and lists the interface requirements. It also details the functional and non-functional requirements essential for the database system.

- Conceptual design of the database: Presents the complete Entity-Relationship (ER) model, data dictionary, and business rules associated with the ER model.
- Logical database schema: Discusses the translation of the ER diagram into a logical database schema, including SQL statements for schema construction, referential constraints, expected database operations, and estimated data volumes.
- Functional dependencies and database normalization: Analyzes functional dependencies for each relation and illustrates the normalization process up to the Third Normal Form (3NF).
- Database system: Provides instructions for installing and invoking the system.
- Suggestions on database tuning: Offers recommendations for optimizing the database, focusing on index structures, database design, and query performance.
- User Application Interface: Describes the development and functionality of the system's user interface and how it facilitates user interaction with the database.
- Conclusions and future work: Summarizes the project's outcomes and explores potential future enhancements to further improve the system's efficiency and scalability.

2. System Requirements

System Architecture Diagram



Interface Requirements

User Registration and Login - Users will be able to create accounts, login to existing ones, and change passwords of existing ones.

- Registration Form
- Login Form
- Change Password Form

Browsing Books - Users will be able to search for and browse through books.

- Search Form (with filters title and genre)

Placing Orders - Users will be able to place and delete orders.

- Place Order Form
- Delete Order Form

Inventory Management - Users will be able to update the details of books, add new books, and delete books.

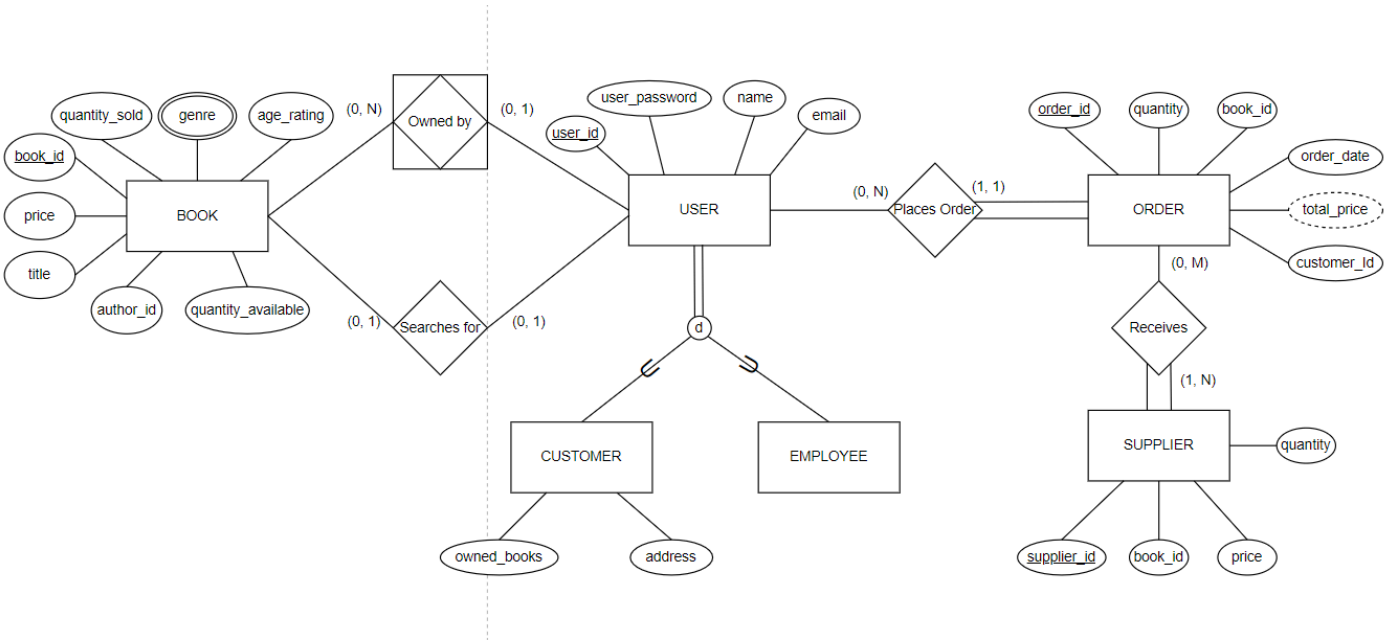
- Adding New Books to Inventory Form
- Updating Book Details Form
- Deleting Books Form

Functional and Non-Functional Requirements

- Users must be able to create accounts with userID, password, name, and email.
- Users must be able to add salary to account if they are an employee and add address and owned books if they are a customer.
- Users must be able to search for books by title or genre.
- Users must be able to see the list of books that match their query.
- Users (employees) must be able to update book inventory details, including adding new books, updating existing books, and removing existing books from the inventory.
- Users must be able to place orders for books.
- Users (employees) must be able to delete orders for books.
- Users must be able to login using their userID and password.
- Users must be able to change their password using their userID and existing password.
- Users must be able to see accurate information after querying.
- The system must immediately update the database based on any information it receives from the users.

3. Conceptual Design

Complete Enhanced Entity Relationship Model



Data Dictionary and Constraints

Relation	Primary/Foreign Keys	Attribute Description/Purpose/Data Type	Constraints
User	Primary Key: user_id - int (10) ● Identifying register for each user (each user type would have a specific starting sequence to differentiate between users)	user_password - varchar (20) ● Password user uses to login (along with email address) name - varchar (30) ● Full name of the user email - varchar (50) ● Email address user uses to login (along with user_password) Subclasses: *Customer* ● user_id - linked to “user_id” under	None

		User ● owned_books - Null String List ○ List of name of books that customer has previously placed an order for ● address - varchar (100) ○ Address of customer used to deliver placed orders *Employee* ● user_id - linked to “user_id” under User ● *Salary* - int ○ Given amount employee earns	
Order	Primary Key: order_id - int (10) ● Identifying register for each order Foreign Key: customer_id - int (10) ● Identifying register for each customer book_id - int (10) ● Identifying register for each book	quantity - int (5) ● Number of books placed in order total_price - double (10) ● Total sum of price of books order_date - date(MM-DD-YYYY) ● Date of which the order was placed	When primary key order_id is deleted, the following actions will take place for foreign keys customer_id and book_id: - Set null - Cascade

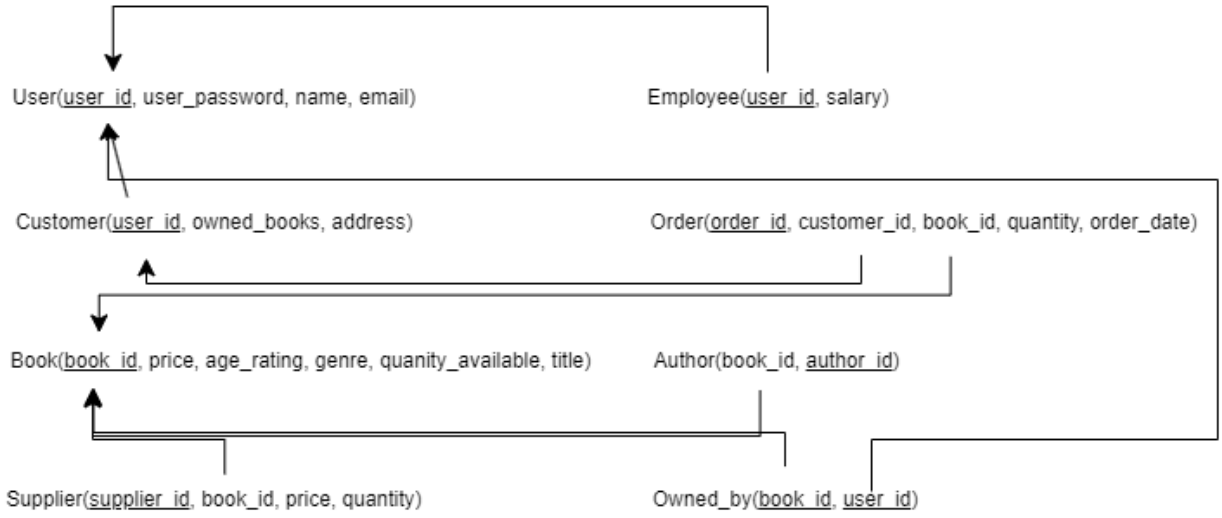
Supplier	Primary Key: supplier_id - int (10) ● Identifying register for each supplier Foreign Key: book_id - int (10) ● Identifying register for each book	price - double (10) ● The price tag of each book quantity - int (5) ● Number book(s) bought from the supplier	When primary key supplier_id is deleted, the following actions will take place for foreign key book_id: - Set null - Cascade
Book	Primary Key: book_id - int (10) ● Identifying register for each book	Price - int ● Cost of each book age_rating - int ● The appropriate age range for the intended audience of the book genre - varchar (20) ● Categorizes books to assist users as they search quantity_available - int ● Tracks the number of copies available for each book quantity _sold - int ● Records the total number of copies each book sold title - varchar (100) ● Allows users to search for the name of a book book_id - int (10) ● Identifies each book	None
Owned_ By	Foreign Keys: book_id - int (10) ● Identifying register for each book user_id - int (10) ● Identifying register for each user (each user type would have a specific starting	None	When primary keys user_id for the User relation and book_id for the Book relation are deleted, the following actions will take place for foreign keys book_id and user_id: - Restrict - Cascade

	sequence to differentiate between users)		

4. Logical Database Schema

General Summary
“Bookstore” (Overarching name of our database)

Relational Database Schema



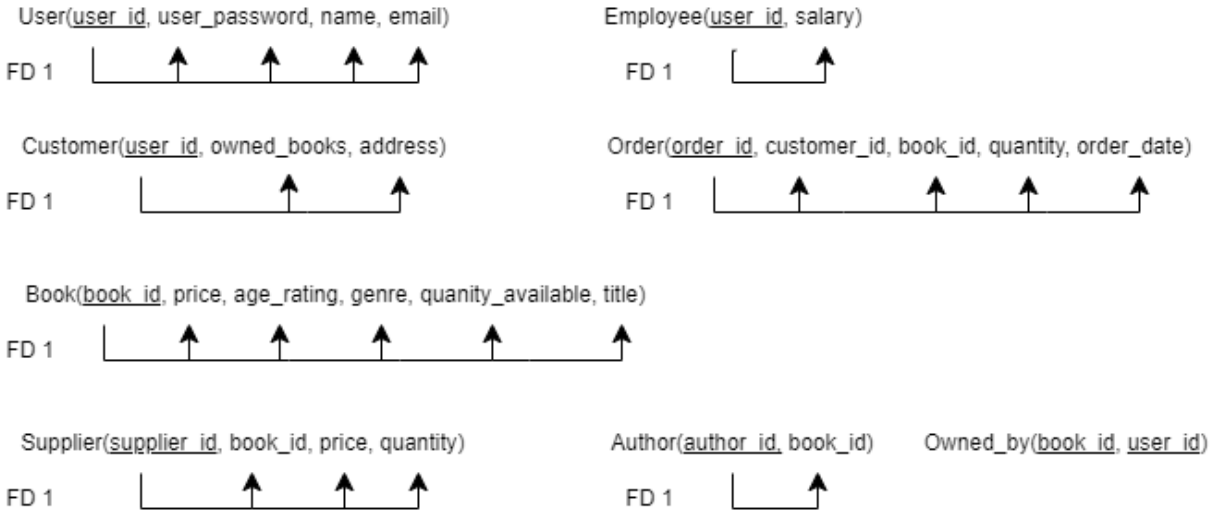
SQL Statements used to Construct Schema

```
-- Create a new database named 'bookstore'
CREATE DATABASE bookstore;

-- Select the 'bookstore' database for use
USE bookstore;

-- Create the 'User' table
CREATE TABLE User (
    user_id         int     NOT NULL,
    user_password       varchar(20),
    name            varchar(30),
    email           varchar(50),
    CONSTRAINT UserPK PRIMARY KEY(user_id)
);

-- Create the 'Customer' table
CREATE TABLE Customer (
    user_id int NOT NULL,
    owned_books varchar(255) NULL,
    address varchar(100),
    CONSTRAINT CustomerPK PRIMARY KEY (user_id),
```

```
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE
);

-- Create the 'Employee' table
CREATE TABLE Employee (
    user_id int NOT NULL,
    salary int,
    CONSTRAINT EmployeePK PRIMARY KEY (user_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE
);

-- Create the 'Book' table
CREATE TABLE Book (
    book_id         int     NOT NULL,
    price           int,
    age_rating      int,
    genre           varchar(20),
    quantity_available  int,
    title           varchar(100),
    CONSTRAINT BookPK PRIMARY KEY(book_id)
);

-- Create the 'Orders' table
CREATE TABLE Orders (
    order_id int NOT NULL,
    customer_id int,
    book_id int,
    quantity int,
    order_date DATE,
    CONSTRAINT OrderPK PRIMARY KEY (order_id),
    FOREIGN KEY (customer_id) REFERENCES Customer(user_id) ON DELETE SET
NULL ON UPDATE CASCADE,
   FOREIGN KEY (book_id) REFERENCES Book(book_id) ON DELETE SET NULL ON
UPDATE CASCADE
);

-- Create the 'Supplier' table
CREATE TABLE Supplier (
    supplier_id int NOT NULL,
    book_id int,
    price double,
    quantity int,
    CONSTRAINT SupplierPK PRIMARY KEY (supplier_id),
```

```
    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON DELETE SET NULL ON
UPDATE CASCADE
);

-- Create the 'Author' table
CREATE TABLE Author(
    book_id     int NOT NULL,
    author_id   int NOT NULL,
    CONSTRAINT AuthorPK PRIMARY KEY (author_id),
    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON DELETE RESTRICT ON
UPDATE CASCADE
);

-- Create the 'Owned_By' relation table
CREATE TABLE Owned_By(
    book_id     int NOT NULL,
    user_id     int NOT NULL,
    CONSTRAINT OwnedByPK PRIMARY KEY (book_id, user_id),
    FOREIGN KEY (book_id) REFERENCES Book(book_id) ON DELETE RESTRICT ON
UPDATE CASCADE,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE RESTRICT ON
UPDATE CASCADE
);
```

5. Functional Dependencies and Database Normalization



1. User Table

- Original Structure:
  - Primary Key: user_id
  - Attributes: user_password, name, email, owned_books, address (for Customer subclass), salary (for Employee subclass)
- Issues Identified:
  - Subclass attributes (owned_books, address) in the User table do not relate to all users.
- Normalization Steps:
  - Separate Customer and Employee into their own tables linked by user_id.
- Final Structure:
  - User Table: user_id (PK), user_password, name, email
  - Customer Table: user_id (FK), owned_books, address
  - Employee Table: user_id (FK), salary

2. Order Table

- Original Structure:
  - Primary Key: order_id
  - Foreign Keys: customer_id, book_id
  - Attributes: quantity, total_price, order_date
- Issues Identified:
  - Total_price potentially a calculated field.
- Normalization Steps:
  - Remove total_price if it is calculated from quantity and book price.
- Final Structure:

- Order Table: order_id (PK), customer_id (FK), book_id (FK), quantity, order_date

3. Supplier Table

- Original Structure:
  - Primary Key: supplier_id
  - Foreign Key: book_id
  - Attributes: price, quantity
- Issues Identified:
  - None. This table is already in 3NF.
- Normalization Steps:
  - None required.
- Final Structure:
  - Supplier Table: supplier_id (FK), book_id (FK), price, quantity

4. Book Table

- Original Structure:
  - Primary Key: book_id
  - Attributes: Price, age_rating, genre, quantity_available, quantity_sold, title, author_id
- Issues Identified:
  - Potential many-to-many relationship between books and authors.
- Normalization Steps:
  - Create a Book_Author table if a book can have multiple authors.
- Final Structure:
  - Book Table: book_id (PK), Price, age_rating, genre, quantity_available, title
  - Author Table: book_id (FK), author_id (FK)

5. Owned_By Table

- Original Structure:
  - Foreign Keys: book_id, user_id
- Issues Identified:
  - None. This table is already in 3NF.
- Normalization Steps:
  - None required.
- Final Structure:
  - Owned_By Table: book_id (FK), user_id (FK)

The SQL code in the previous phase matches to the given 3NF normalization as shown above.

6. Database System

Installation and Invocation

1. Installation:
    - Download and install XAMPP from Apache Friends.
    - Run the XAMPP Control Panel as administrator and start the Apache and MySQL modules.
2. Database Setup:
    - Copy all files from file.zip into C:\xampp\htdocs, overwriting existing files.
    - Import the create.sql file via the phpMyAdmin interface at http://localhost/phpmyadmin/.
    - Import the load.sql file as well if you need to import sample data.

Using the System

- Access the Database Management System (DBMS):
    - Navigate to http://localhost/home.html for user operations.
    - For administrative tasks, access http://localhost/phpmyadmin/.

7. Suggestions on Database Tuning

Indexing:

- Create indexes on frequently searched fields such as user_id, book_id, and order_id to speed up query times.

Database Design:

- Use partitioning to manage large datasets, especially for historical order data and user logs.
- Normalize the database to 3NF to eliminate redundant data and reduce update anomalies.

Query Optimization:

- Avoid using SELECT * in queries; instead, specify the column names to reduce the load.
- Use JOINs instead of sub-queries where applicable to improve performance.

8. User Application Interface

Development Tools:

- Front-end: HTML forms styled with CSS for user interaction.
- Back-end: PHP scripts to handle form submissions and interact with the MySQL database.

Interface Design:

- Forms for adding, updating, and deleting records for users, books, orders, and suppliers.
- Validation implemented to ensure data integrity before submission to the database.

User Interaction:

- Users can register, login, browse books, place orders, and manage their profiles through intuitive web forms.
- Employees can manage inventory, process orders, and update book details through dedicated forms.

9. Conclusions and Future Work

Conclusions:

- The project successfully integrates a robust database system to manage a bookstore's operations effectively.
- Through normalization and thoughtful interface design, the system ensures data integrity and ease of use.

Future Work:

- Implement additional features like advanced analytics for business insights.
- Introduce mobile applications to provide more accessibility.
- Enhance security measures to safeguard user data and transactions.
- Incorporate machine learning algorithms that give personalized recommendations.

10. References

B. Fajardo, “How to Use Associative Entities in Relational Databases,” *Medium*, Dec. 13, 2019. https://medium.com/@BryanFajardo/how-to-use-associative-entities-in-relational-databases-4456a2c71cda

R. A S, “ER Diagrams in DBMS: Entity Relationship Diagram Model,” *Simplilearn*, May 23, 2023. https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms

“Mapping an E-R Diagram to a Relational DBMS,” *Open Textbooks for Hong Kong*, Aug. 27, 2015. https://www.opentextbooks.org.hk/ditatopic/25280

“Learn All About Relational Database Normalization In 3 Steps,” *RedSwitches*, Aug. 10, 2023. https://www.redswitches.com/blog/relational-database-normalization/

A. Khan, “How to Connect MySQL Database with PHP (A Developer’s Guide),” *The Official Cloudways Blog*, Jun. 29, 2019. https://www.cloudways.com/blog/connect-mysql-with-php/

“Foreign Key Constraint,” *CockroachDB Docs*. https://www.cockroachlabs.com/docs/stable/foreign-key

11. Appendix

Files.zip - https://drive.google.com/file/d/1_Ge9U5EDKAbfiwVKFZOTb6JDWkL-HCUJ/view?usp=sharing