Library Database Management System

OREM 7353 Project Report

Authored by Team 4: Kendall Whitbeck, Alex Xia

Table of Contents

[Part 1: Introduction and Literature Work 4](#_Toc133043517)

[Introduction 4](#_Toc133043518)

[Existing Literature 4](#_Toc133043519)

[Part 2: DBMS Design 5](#_Toc133043520)

[DBMS Description 5](#_Toc133043521)

[Constraints 6](#_Toc133043522)

[Entity-Relationship (ER) Diagram without Normalization 6](#_Toc133043523)

[Business Rules 6](#_Toc133043524)

[Entity-Relationship (ER) Diagram 7](#_Toc133043525)

[Library Data Instances 7](#_Toc133043526)

[Part 3: Normalization 10](#_Toc133043527)

[Dependency Diagrams 12](#_Toc133043528)

[Queries 14](#_Toc133043529)

[Insert 14](#_Toc133043530)

[Delete 15](#_Toc133043531)

[Update & Set 16](#_Toc133043532)

[Select, Sum & Coalesce 17](#_Toc133043533)

[Add Constraint: Check Loan Medium Type 18](#_Toc133043534)

[Views 19](#_Toc133043535)

[Available Book Copies View 19](#_Toc133043536)

[Loan Details View 20](#_Toc133043537)

[Overdue Loans View 20](#_Toc133043538)

[Member Loans View 21](#_Toc133043539)

[Conclusion 21](#_Toc133043540)

[References 22](#_Toc133043541)

[Appendix A – SQL Code 23](#_Toc133043542)

Table of Figures

[Figure 1: Entity Relationship Diagram for the Library DBMS (without Normalization) 7](#_Toc133043543)

[Figure 2: Book Relation Instance 8](#_Toc133043544)

[Figure 3: eBook Relation Instance 8](#_Toc133043545)

[Figure 4: Audiobook Relation Instance 8](#_Toc133043546)

[Figure 5: Book Copy Relation Instance 8](#_Toc133043547)

[Figure 6: eBook Copy Relation Instance 9](#_Toc133043548)

[Figure 7: Audiobook Copy Relation Instance 9](#_Toc133043549)

[Figure 8: Member Relation Instance 9](#_Toc133043550)

[Figure 9: Hold Relation Instance 10](#_Toc133043551)

[Figure 10: Loan Relation Instance 10](#_Toc133043552)

[Figure 11: Entity Relationship Diagram for the Library DBMS (with Normalization) 12](#_Toc133043553)

[Figure 12: Dependency Diagrams for the Library DBMS 14](#_Toc133043554)

[Figure 13: SQL Code Inserting Data into the Book Relation 14](#_Toc133043555)

[Figure 14: Book Relation Before Inserting Data 15](#_Toc133043556)

[Figure 15: Book Relation After Inserting Data 15](#_Toc133043557)

[Figure 16: SQL Code Deleting Data 15](#_Toc133043558)

[Figure 17: Book Relation After Deleting Data 15](#_Toc133043559)

[Figure 18: SQL Code Updating the Member Relation 16](#_Toc133043560)

[Figure 19: Member Relation Before Updating 16](#_Toc133043561)

[Figure 20: Member Relation After Updating 17](#_Toc133043562)

[Figure 21: SQL Code Counting the Library’s Total Number of Fantasy Item Copies 17](#_Toc133043563)

[Figure 22: Relation Showing the Library’s Total Number of Fantasy Item Copies 17](#_Toc133043564)

[Figure 23: SQL Code Adding the Check Loan Medium Type Constraint 18](#_Toc133043565)

[Figure 24: Loan Relation Properties Before Implementing Check Loan Medium Type Constraint 18](#_Toc133043566)

[Figure 25: Loan Relation Properties After Implementing Check Loan Medium Type Constraint 19](#_Toc133043567)

[Figure 26: Available Book Copies View 20](#_Toc133043568)

[Figure 27: Loan Details View 20](#_Toc133043569)

[Figure 28: Overdue Loans View 20](#_Toc133043570)

[Figure 29: Member Loans View 21](#_Toc133043571)

# Part 1: Introduction and Literature Work

## Introduction

Our team has chosen to design and implement a library Database Management System (DBMS) to improve the efficiency of library resource management and provide users with timely and sufficient information. The modern-day library must be prepared to process large amounts of data, and an automated, cloud-based, high-performing DBMS is essential to keeping costs down, improving user satisfaction, and promoting a happy and educated community. Our DBMS will provide library managers and their users with a digitally-centralized location for storing and managing information on physical books, eBooks, audiobooks, and other resources. It will also streamline cataloging and classification, provide advanced searching functionality, and enable the procurement of digital media, thereby increasing a library’s total catalog without incurring the additional cost of physical storage.

By creating an online library DBMS that applies modern DBMS principles, we aim to improve the visibility and accessibility of a given library’s catalog. This will make it easier for users to search and access the library’s resources, regardless of location, while also providing accessibility to more users. Users can obtain access to library databases organized by our DBMS via subscription, a library card, or enrollment as in the case of a university library. We hope that our database will provide accessibility to a larger range of users and simplify the process for those who are looking to utilize eBooks and other library resources.

## Existing Literature

There are numerous existing database management systems used in libraries and bibliographic databases worldwide, such as the Library of Congress Integrated Library System, the WorldCat database, and Biblio. These systems have several similarities with our proposed database, such as cataloging and classification, circulation, patron management, and reporting. However, our database will have advanced searching functionality via the use of Artificial Intelligence (AI) and fuzzy logic. Combining existing AI algorithms, such as OpenAI, with fuzzy logic can enhance the capabilities of intelligent systems by allowing them to process and reason with uncertain or imprecise data. Fuzzy logic is a mathematical framework that deals with the management of uncertainty and imprecision in decision-making processes, making it an ideal tool for AI applications. By incorporating fuzzy logic algorithms into OpenAI-powered systems, the library DBMS can understand and respond to natural language queries more accurately and provide more nuanced and relevant responses. This combination can lead to more intelligent and intuitive systems that are effective in a wide range of applications. Many electronic library search systems, such as the Library of Congress Online Catalog and the WorldCat database, use fuzzy search to help users find books and other materials. Fuzzy search can also be used to search for titles with typos or misspellings, and authors with variant spellings of their names. However, it is still difficult for OpenAI access systems to search for books through descriptions. In the future, we hope to develop and utilize the capability to search by description based on existing fuzzy search capabilities.

Additionally, our database will be cloud-based, making it easily accessible from any device, reducing the need for on-site hardware, and enabling data sharing among different libraries.

# Part 2: DBMS Design

## DBMS Description

Some of the tables to be included in the library DBMS design are as follows.

1. Book table: This table contains information about each physical book in the library's collection, such as its book ID, International Standard Book Number (ISBN), title, author, publisher, publication year, genre, and the number of copies available.
   1. Primary key: book ID.
2. eBook table: This table contains information about each electronic book (eBook) in the library's collection, such as its eBook ID, title, author, publisher, Digital Object Identifier (DOI), publication year, genre, and the number of copies available.
   1. Primary key: eBook ID.
3. Audiobook table: This table contains information about each audiobook in the library's collection, such as its audiobook ID, title, author, publisher, International Standard Audiovisual Number (ISAN), publication year, genre, and the number of copies available.
   1. Primary key: audiobook ID.
4. Book Copy table: This table contains information about each copy of a physical book in the library's collection, such as its book copy ID, status, and allowable loan period.
   1. Primary key: book copy ID.
   2. Foreign key: book ID
5. eBook Copy table: This table contains information about each copy of an eBook in the library's collection, such as its eBook copy ID, status, and allowable loan period.
   1. Primary key: eBook copy ID.
   2. Foreign key: eBook ID
6. Audiobook Copy table: This table contains information about each copy of an audiobook in the library's collection, such as its audiobook copy ID, status, and allowable loan period.
   1. Primary key: audiobook copy ID.
   2. Foreign key: audiobook ID.
7. Member table: This table contains information about library members, such as their first and last name, address, contact details, membership type, and membership status.
   1. Primary key: member ID.
8. Hold table: This table contains information about each hold transaction, such as the hold ID, the member who placed the hold, the item put on hold, the date the hold was made, and the status of the hold.
   1. Primary key: hold ID.
   2. Foreign keys: member ID, book ID, eBook ID, and audiobook ID.
9. Loan table: This table contains information about each loan transaction, such as the loan ID, the date the item was borrowed, the due date for the item, and the date the loan was return.
   1. Primary key: loan ID.
   2. Foreign keys: member ID, book copy ID or eBook copy ID or audiobook copy ID.

### Constraints

The rules governing the library DMBS schema are as follows:

1. Unique IDs specific to the DBMS are utilized as the primary keys for each medium, e.g., “book\_id,” “ebook\_id,” and “audiobook\_id,” due to the following:
   1. Books can have an ISBN but are not required to.
   2. eBooks can have a DOI but are not required to.
   3. Audiobooks can have an ISAN but are not required to.
2. ISBN is not applicable to eBooks and audiobooks, and refers to ISBN-13, specifically.
3. Loan transactions must have non-null values for the Member ID.
4. Loan transactions must each reference exactly one media item. Thus, each loan transaction must have a non-null value for exactly one of the following three medium IDs and must have null for the other two: book ID, eBook ID, audiobook ID.
5. For each medium, a copy (e.g., copy of book, copy of eBook, or copy of audiobook) can only exist if the respective entity exists (e.g., book, eBook, or audiobook).

## Entity-Relationship (ER) Diagram without Normalization

The Entity Relationship (ER) Diagram was generated using MySQL Workbench 8.0 Community Edition (MySQL).

### Business Rules

The Business Rules inform the constitution of the ER Diagram and the multiplicities of the relationships. The Business Rules for the library DBMS are as follows:

1. A book, eBook, or audiobook can exist in the library database with zero or infinite copies.
2. A copy of a book, eBook, or audiobook may only be on a single loan at a time.
3. A copy of a book, eBook, or audiobook may exist in the library catalog without ever having been loaned.
4. A copy of a book, eBook, or audiobook has no limit to the number of times it can be loaned.
5. A book, eBook, or audiobook has no limit to the number of holds placed upon it over time or concurrently.
6. Any unique hold or loan may be associated with exactly one member.
7. A member may exist without any loans or holds.
8. A member may place multiple concurrent holds, with the limit being subject to membership type and status.
9. A member may borrow multiple concurrent loans, with the limit being subject to membership type and status.
10. A hold applies to the book, eBook, or audiobook entity and not any one specific copy.

### Entity-Relationship (ER) Diagram

Diagram

Description automatically generated

Figure 1: Entity Relationship Diagram for the Library DBMS (without Normalization)

## Library Data Instances

The library data instances, by relation, are as follows:

Graphical user interface, text, application

Description automatically generated

Figure 2: Book Relation Instance

Graphical user interface, text, application

Description automatically generated

Figure 3: eBook Relation Instance

Graphical user interface, text, application

Description automatically generated

Figure 4: Audiobook Relation Instance

Table

Description automatically generated

Figure 5: Book Copy Relation Instance

Table

Description automatically generated

Figure 6: eBook Copy Relation Instance

Table

Description automatically generated

Figure 7: Audiobook Copy Relation Instance

Table

Description automatically generated

Figure 8: Member Relation Instance

Table

Description automatically generated

Figure 9: Hold Relation Instance

Table

Description automatically generated

Figure 10: Loan Relation Instance

# Part 3: Normalization

Normalized Relations The goal of normalizing a database schema is to reduce data redundancy, improve data integrity, and ensure that it adheres to the rules of each normal form. The 3NF tables were created to achieve these goals. Here's a detailed explanation of why the tables were divided into their current structure in the 3NF:

Author, Publisher, and Genre tables: These tables were created to eliminate data redundancy and maintain data integrity. In the original schema, the author, publisher, and genre information were repeated across the Book, eBook, and Audiobook tables. By creating separate tables for each of these entities and linking them with foreign keys, we avoid redundancy and ensure that any update, insertion, or deletion operation can be performed consistently.

Book, eBook, and Audiobook tables: These tables hold information specific to each medium. They have been structured to contain only data directly relevant to their primary keys. This ensures that all the attributes in each table are fully functionally dependent on the primary key, satisfying 1NF and 2NF requirements.

Book Copy, eBook Copy, and Audiobook Copy tables: These tables were created to separate information about the individual copies of each medium from the general information about the medium itself. This separation helps to maintain data integrity and ensures that each table contains only data relevant to its primary key. This satisfies the 2NF requirement.

Member table: This table contains information about library members. It is structured to maintain data integrity and follows the rules of 1NF, 2NF, and 3NF, ensuring that all attributes are fully functionally dependent on the primary key.

Hold table: This table captures information about hold transactions. It includes foreign keys for member\_ID and the medium IDs (book\_ID, eBook\_ID, audiobook\_ID). By structuring the table this way, we maintain data integrity and ensure that hold transactions only reference valid members and media items.

Loan table: This table contains information about loan transactions. It has foreign keys for member\_ID and the medium copy IDs (book\_copy\_ID, eBook\_copy\_ID, audiobook\_copy\_ID). This structure ensures that each loan transaction has a non-null value for the Member ID and exactly one non-null value for the medium copy IDs, as per the constraints specified. This maintains data integrity and ensures that loan transactions only reference valid members and media items.

By dividing the schema into these 3NF tables, we have reduced data redundancy, improved data integrity, and ensured that the schema adheres to the rules of the first three normal forms. This optimized schema will lead to better performance, easier maintenance, and fewer data anomalies. The normalized ER diagram is shown below:

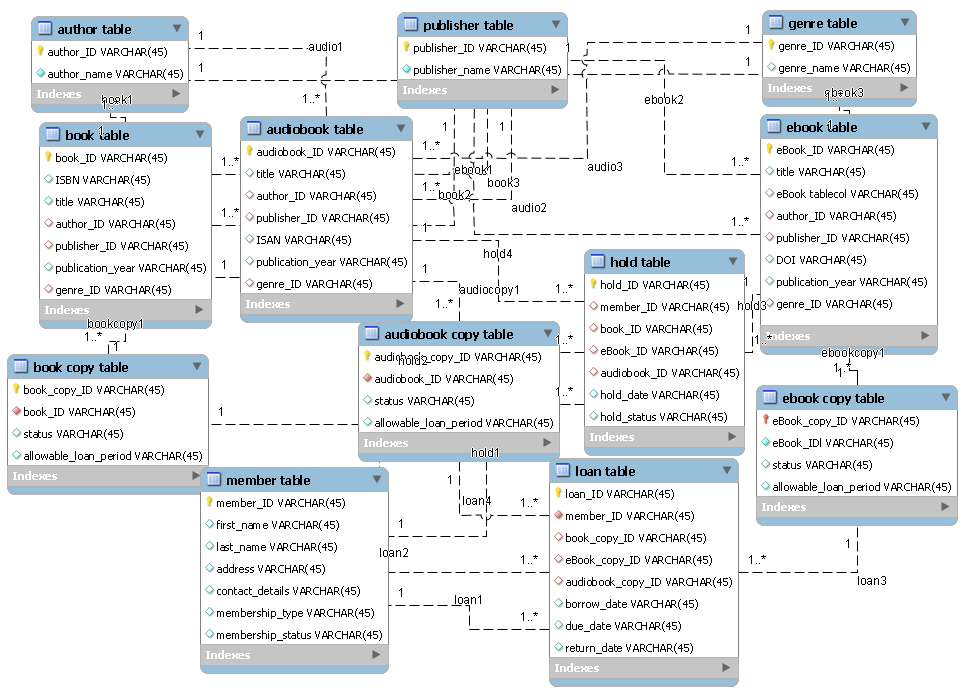


Figure 11: Entity Relationship Diagram for the Library DBMS (with Normalization)

## Dependency Diagrams

The tables in the database schema are related in the following ways:

* Author
  + Has many Books, eBooks, and Audiobooks.
* Publisher
  + Has many Books, eBooks, and Audiobooks.
* Genre
  + Has many Books, eBooks, and Audiobooks.
* Book
  + Belongs to one Author, Publisher, and Genre.
  + Has many Book Copies.
  + Has many Holds.
* eBook
  + Belongs to one Author, Publisher, and Genre.
  + Has many eBook Copies.
  + Has many Holds.
* Audiobook
  + Belongs to one Author, Publisher, and Genre.
  + Has many Audiobook Copies.
  + Has many Holds.
* Book Copy
  + Belongs to one Book.
  + Has many Loans.
* eBook Copy
  + Belongs to one eBook.
  + Has many Loans.
* Audiobook Copy
  + Belongs to one Audiobook.
  + Has many Loans.
* Member
  + Has many Holds.
  + Has many Loans.
* Hold
  + Belongs to one Member.
  + Belongs to one Book, eBook, or Audiobook (mutually exclusive).
* Loan
  + Belongs to one Member.
  + Belongs to one Book Copy, eBook Copy, or Audiobook Copy (mutually exclusive).

Overall, the relationships between the tables in the database schema help ensure data integrity and consistency by ensuring that each transaction references valid entities in the parent and child tables. The Dependency Diagram for the Library DBMS schema is as follows:

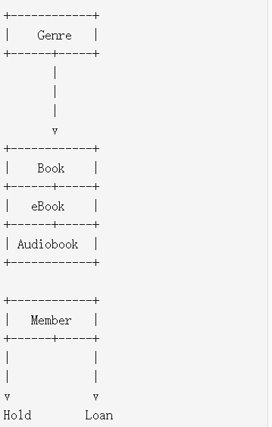
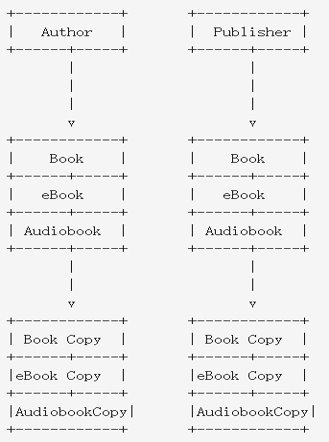


Figure 12: Dependency Diagrams for the Library DBMS

## Queries

### Insert

Many INSERT commands were used to populate the database. The following illustrates how the Book Relation was initially populated:



Figure 13: SQL Code Inserting Data into the Book Relation

Please see the appendix for the rest of the SQL code inserting data into the other relations in the library database. The following figure shows the Book relation before inserting data:



Figure 14: Book Relation Before Inserting Data

The following figure shows the Book relation after inserting data:

Graphical user interface, text, application, email

Description automatically generated

Figure 15: Book Relation After Inserting Data

Observe that the 10 rows of data present in the SQL code have now populated the Book relation.

### Delete

The following illustrates how the Book Relation was initially populated:



Figure 16: SQL Code Deleting Data

The following figure shows the Book relation after deleting Harry Potter and the Sorcerer’s Stone:

Graphical user interface, text, application, chat or text message

Description automatically generated



Figure 17: Book Relation After Deleting Data

Observe that the row associated with Harry Potter and the Sorcerer’s Stone is no longer present in the Book relation.

### Update & Set

The UPDATE and SET commands were used to update some of the member addresses. The code is as follows:



Figure 18: SQL Code Updating the Member Relation

The following figure shows the relation before the update:

Graphical user interface

Description automatically generated with low confidence



Figure 19: Member Relation Before Updating

The following figure shows the relation after the update:

Table

Description automatically generated



Figure 20: Member Relation After Updating

### Select, Sum & Coalesce

More complex SQL queries can be utilized to fulfill more specific use cases. For example, one may wish to know how many copies the Library has in the Fantasy genre, broken out by type. The following SQL code using SELECT, COALESCE, and SUM commands demonstrates how this can be done:



Figure 21: SQL Code Counting the Library’s Total Number of Fantasy Item Copies

The following figure shows the resulting relation:



Figure 22: Relation Showing the Library’s Total Number of Fantasy Item Copies

Note the use of the COALESCE command. This is critical for handling cases where any of the medium types do not have a single entity in the Fantasy genre; this is the case for the Audiobook relation. As such, the total Audiobooks and the total items would return null if COASLESCE were not used.

### Add Constraint: Check Loan Medium Type

The Check Loan Medium Type constraint ensures that exactly one of the three Media Copy ID foreign key attributes in the Loan relation can be non-null, while the other two must be null. This constraint was created via the following SQL code:



Figure 23: SQL Code Adding the Check Loan Medium Type Constraint

The following figure shows the Loan relation properties before implementing the Check Loan Medium Type constraint:

Text

Description automatically generated

Figure 24: Loan Relation Properties Before Implementing Check Loan Medium Type Constraint

The following figure shows the Loan relation properties after implementing the Check Loan Medium Type constraint:

Text

Description automatically generated



Figure 25: Loan Relation Properties After Implementing Check Loan Medium Type Constraint

Note the inclusion of the highlighted sixth constraint, ‘check\_loan\_medium\_type,’ at the bottom of the image. Please see the appendix for the SQL code producing the view.

## Views

The following four views are examples of views that one might find beneficial in using the library DBMS schema:

1. Available Book Copies
2. Loan Details
3. Overdue Loans
4. Member Loans

### Available Book Copies View

The Available Book Copies view displays all book copies available for loan. The view was created by joining elements from the Book table with “Available” copies from the Book Copy table. The resulting table is as follows:

Graphical user interface, text, application, table

Description automatically generated

Figure 26: Available Book Copies View

Please see the appendix for the SQL code producing the view.

### Loan Details View

The Loan Details view displays all loan records with their corresponding book, eBook, or audiobook titles. The view was created by joining non-null elements from the Loan table with elements from the corresponding media tables. The resulting table is as follows:



Figure 27: Loan Details View

Please see the appendix for the SQL code producing the view.

### Overdue Loans View

The Overdue Loans view displays all overdue loans assuming the current date is 2022-02-20. The view was created by joining non-null elements from the Loan table with elements from the Member table and elements from the corresponding media tables. The resulting table is as follows:

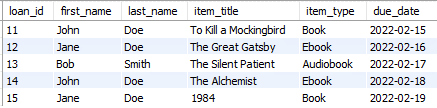


Figure 28: Overdue Loans View

Please see the appendix for the SQL code producing the view.

### Member Loans View

The Member Loans view displays all the loans made by a specific member; in this case, member ID = 7 or Jane Doe. The view was created by joining non-null elements from the Loan table with elements from the Member table and elements from the corresponding media tables. The resulting table is as follows:

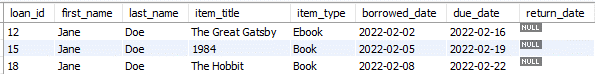


Figure 29: Member Loans View

Please see the appendix for the SQL code producing the view.

# Conclusion

In conclusion, our team has designed and proposed a library database management system (DBMS) that aims to improve the efficiency of library resource management, promote a happy and educated community, and increase user satisfaction. The proposed DBMS is cloud-based, allowing users to access it from any device, and includes advanced searching functionality via the integration of artificial intelligence and fuzzy logic.

The system contains tables such as book, eBook, audiobook, book copy, eBook copy, audiobook copy, member, hold, and loan. While there are similarities between our system and other bibliographic databases, our DBMS offers additional functionality, making it more intelligent and intuitive. We hope that our database will provide accessibility to a larger range of users and simplify the process for those who are looking to utilize eBooks and other digital library resources, ultimately leading to a more informed and educated society.

# References

1. Couchbase. (2022, January 13). Fuzzy matching concepts and applications. Retrieved from <https://blog.couchbase.com/fuzzy-matching-concepts-and-applications/>
2. Hindawi. (2019). Practical Applications of Fuzzy Logic. [Online]. Available at: <https://www.hindawi.com/journals/mpe/2019/7515124/>. Accessed on 23 March 2023.
3. MySQL Workbench 8.0 Community Edition [Computer software]. (2018). Retrieved from <https://dev.mysql.com/downloads/workbench/>.
4. Biblio. "9780747532743 - Harry Potter and the Philosopher's Stone (Harry Potter) by J. K. Rowling." Biblio.com, Biblio, [www.biblio.com/9780747532743](http://www.biblio.com/9780747532743).
5. Library of Congress Integrated Library System. Library of Congress, 2022, <https://ils.loc.gov/>.
6. OCLC. (2023). WorldCat Database. OCLC. https://www.worldcat.org/

# Appendix A – SQL Code

The following embedded SQL files contain the SQL code used for the Library DBMS:

1. 
2. 
3. 
4. 
5. 
6. 