LIBRARY MANAGEMENT SYSTEM

Team Name: UB GOBULLS

Sri Sannihitha Gangina
srisanni
srisanni@buffalo.edu

Sai Shankar Vanam
saishank
saishank@buffalo.edu

Sai Janhavi Tamatam
saijanha
saijanha@buffalo.edu

I. PROBLEM STATEMENT

This project offers a platform for storing information about publishers, books, library branches, and patrons who are borrowing books. Additionally, a dynamic method of accessing data on issued and returned books, maintaining and updating data on all borrowed and returned books in real time. It is necessary to have a database used to store all these details in order to know the information regarding issued and returned books. Since this project can be used to obtain the a forementioned statement and update all issued and returned records in real time. The work of manually storing information and making mistakes is also decreased by this project and the resources used to plan and create this beneficial project.

This project's primary goal is to replace the manual entry and record-keeping system with an easy and straightforward database management system for the library. The primary goal of this project is to build a high-quality, dynamic management system that will use this database to store all the information about borrowed books.

II. PROPOSED SOLUTION

The proposed solution for the library management system using the provided schema would involve implementing a dynamic database management system that can store and track information related to publishers, books, library branches, patrons, and the issuance and return of books. The solution would include the following steps:

- Create a database management system that can store all the necessary data related to the library. This system will use the provided schema, which includes tables for publishers, books, library branches, cards, and the issuance and return of books.
- Populate the database with relevant data about publishers, books, library branches, and patrons. This information can be obtained through manual entry or imported from external sources.
- Use the database to track information about the issuance and return of books in real time. When a patron checks out a book, the system should update the database with the relevant information, such as the book ID, branch ID, patron's card number, date issued, and due date. Similarly, when a patron returns a book, the system should update the database to reflect the return date and whether the book was returned on time or late.
- Implement a user-friendly interface that allows librarians to easily access and update information in the database. This interface should allow librarians to view information about publishers, books, library branches, and patrons, as well as track the issuance and return of books.

III. TARGET USERS

The target users for the above database schema are those who are involved in managing and maintaining a library. This includes librarians, library staff, and other personnel responsible for overseeing library operations.

Here are some points to elaborate on the target users:

- Librarians: The primary users of the library management system would be librarians. They would use the system to maintain records of books, patrons, and book loans. They would also use the system to generate reports, such as overdue books, popular books, and other analytics that could help them manage the library.
- Library staff: Other library staff members, such as clerks and assistants, would also use the system to perform tasks such as checking out and checking in books, managing patron records, and assisting with general library operations.
- Patrons: While patrons would not directly use the database schema, they would indirectly benefit from it. The database would help ensure that the library's operations run smoothly, making it easier for patrons to find and borrow books.
- Library management: In addition to the users mentioned above, library management would also be a target user of the database schema. They would use the system to monitor and analyze library operations, make informed decisions, and improve the library's overall performance.

Overall, the target users for the above database schema are those involved in the management and maintenance of a library. The system would help them perform tasks such as managing records, tracking loans, and analyzing data, ultimately improving the library's overall performance and user experience.

IV. SCHEMA DESCRIPTION

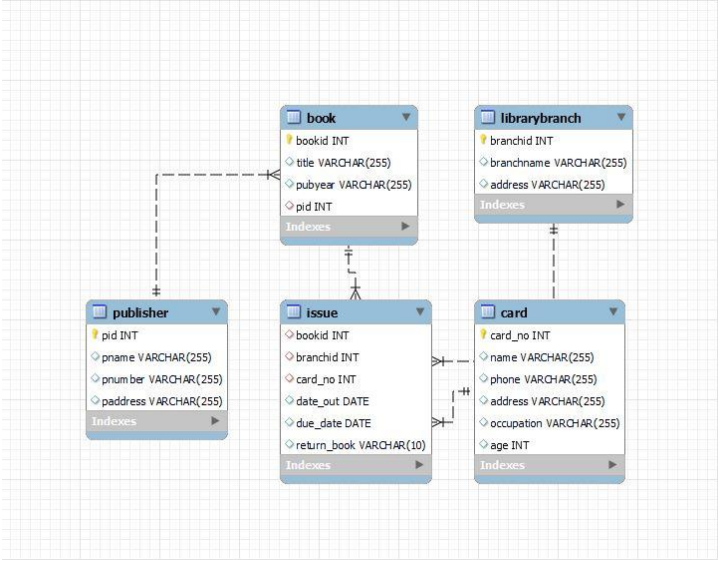


Figure 1: ERD Diagram

- The "**publisher**" table stores information about publishers, such as their ID, name, phone number, and address. This information is useful for librarians to keep track of publishers' contact details and to categorize books according to their respective publishers.
- The "**book**" table contains details about the books in the library, such as the book's ID, title, publication year, and the publisher's ID who published the book. This table is crucial for managing books in the library, as librarians can use it to find specific books, track their publication dates, and link them to the publishers' information stored in the "publisher" table.
- The "**librarybranch**" table contains information about the different branches of the library, such as their ID, name, and address. This table is essential in tracking which branch of the library has which books, and it helps librarians keep track of book movement within the library.
- The "**card**" table stores information about patrons who borrow books, such as their ID, name, age, phone number, address, and occupation. This table is essential in identifying patrons who borrow books and in keeping track of their contact information.
- Lastly, the "**issue**" table is used to keep track of the books borrowed by patrons. It stores information about the book ID, library branch ID, patron ID, date borrowed, due date, and return date. This table is crucial for librarians to manage book loans, to calculate fines for overdue books, and to track book returns**.**

Overall, the above datatables serve different purposes in a library management system, and they work together to provide librarians with the necessary information to manage books, patrons, and branches effectively.

Address: This table uses 'address_id' to uniquely store addresses, thus acting as the primary key for this table. It also employs 'country_id' from the 'Country' table as a foreign key to establish the relationship between the two relations.

V. ATTRIBUTES DESCRIPTION

Publisher table:

- pid (int primary key): represents the unique ID of a publisher.
- pname (varchar(255)): represents the name of the publisher.
- pnumber (varchar(255)): represents the phone number of the publisher.
- paddress (varchar(255)): represents the address of the publisher.

Book table:

- bookid (int primary key): represents the unique ID of a book.
- title (varchar(255)): represents the title of the book.
- pubyear (varchar(255)): represents the year in which the book was published.
- pid (int): represents the ID of the publisher of the book.

Librarybranch table:

- branchid (int primary key): represents the unique ID of a library branch.
- branchname (varchar(255)): represents the name of the library branch.
- address (varchar(255)): represents the address of the library branch.

Card table:

- card_no (int primary key): represents the unique card number of a library patron.
- name (varchar(255)): represents the name of the library patron.
- age (int): represents the age of the library patron.
- phone (varchar(255)): represents the phone number of the library patron.
- address (varchar(255)): represents the address of the library patron.
- occupation (varchar): represents the occupation of the library patron.

Issue table:

- bookid (int): represents the ID of the book being issued.
- branchid (int): represents the ID of the library branch from which the book is being issued.
- card_no (int): represents the card number of the library patron who is issuing the book.
- date out (date): represents the date on which the book was
- due date (date): represents the date on which the book is due to be returned.
- return book (varchar(10)): represents the status of the book, whether it has been returned or not.

VI. FDs AND BCNF OF TABLE

Table Name	Functional Dependencies
Publisher	pid -> pname, pnumber, paddress
Book	bookid -> title, pubyear, pid
Issue	bookid, branchid, card_no -> date_out, due_date, return_book
Libraray Branch	branchid -> branchname, address
Card	card_no -> name, age, phone, address, occupation

Table 1: Functional dependencies

Table name	Primary Keys	Functional Dependancies	BCNF Compilant
Publisher	pid	pid -> pname, pnumber, paddress	Yes
Book	bookid	bookid -> title, pubyear, pid	Yes
Issue	N/A	bookid, branchid, card_no -> date_out, due_date, return_book	Yes
LibraryBranch	branchid	branchid -> branchname, address	Yes
Card	card_no	card_no -> name, age, phone, address, occupation	Yes

Table 2: BCNF Proof

The BCNF (Boyce-Codd Normal Form) is a higher level of normalization than the third normal form (3NF) and ensures further reduction of data redundancy. The following are the rules of BCNF format:

- Each attribute should be functionally dependent on the primary key:
- This means that every non-key attribute in the table must be dependent on the primary key.

- No partial dependencies:
- Partial dependency occurs when one part of a composite primary key determines the value of a non-key attribute. It violates BCNF, and the table must be split into two tables.

- No transitive dependencies:
- Transitive dependency occurs when a non-key attribute is determined by another non-key attribute. To meet BCNF, the table must be split into two tables.

- Every functional dependency must be a dependency on a candidate key:
- In BCNF, every functional dependency must be on the candidate key, not just the primary key.

By following these rules, we can ensure that the database is well-structured, with minimum redundancy and update anomalies. From the above relational schema, we can prove that one can see that FDs in each relation adhere to two given conditions to be consistent with the BCNF form

- They are non-trivial
- The attributes on the left side of the FDs are super keys.

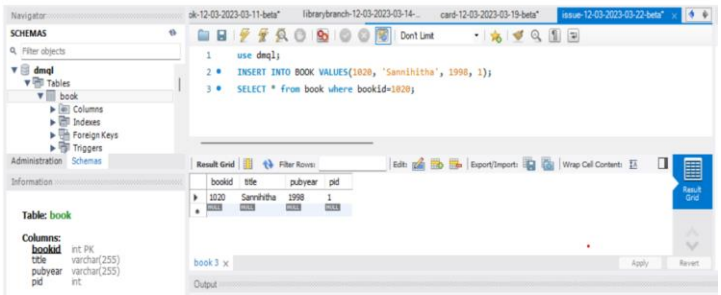
VII. SCHEMA ENHANCEMENT

Data validation and constraints can help ensure that the data entered into the database is accurate and consistent. For example, constraints can be added to limit the loan duration of a book or the maximum number of books that a borrower can check out at once. This can help prevent errors and inconsistencies in the database.

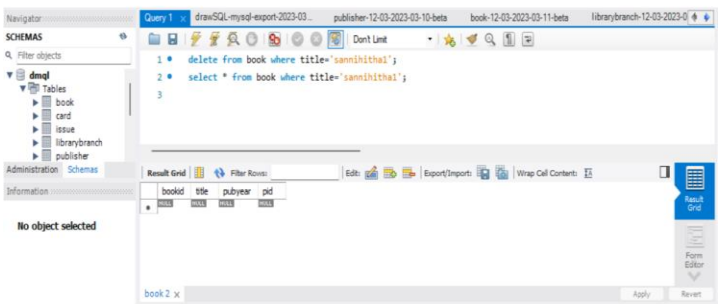
Indexing can also be used to improve the performance of queries. By creating indexes on specific columns in the database, queries can quickly locate the data they need without having to scan through the entire database. This can help speed up query times and improve overall database performance. However, it's important to be mindful of the potential trade-offs, as indexing can also increase the size of the database and slow down write operations.

VIII. BASIC QUERIES

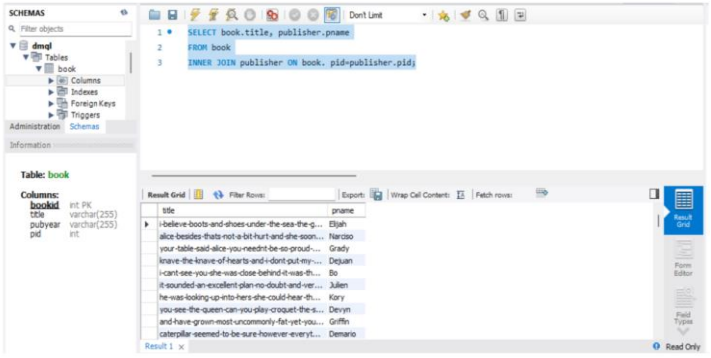
1) SQL COMMAND FOR INSERTION AND SELECTION



2) SQL COMMAND DELETION



3) SQL JOIN OPERATION



IX. **ADVANCED QUERIES**

To retrieve the total number of books published by each publisher.

```
SELECT p.pname, COUNT(b.bookid) as total_books_published
FROM dmql.publisher p
JOIN dmql.book b ON p.pid = b.pid
GROUP BY p.pname
ORDER BY total_books_published DESC;
```

pname	total_books_published
Prudence Tremblay	1
Kellie Schaefer	1
Elna Reynolds	1
Kaleb Deckow	1
Efren Herzog	1
Heloise Kautzer DDS	1
Durward Kemmer	1
Prof. Heather Pfeffer DDS	1
Mabelle Baumbach	1
Susie Heidenreich	1
Jackie Wyman	1
Meagan Balistreri	1
Alfredo Dietrich DVM	1
Prof. Reece Fahey Jr.	1
Dr. Deon Wisozk Sr.	1
Mrs. Marlen Beahan III	1
Elena Bode	1
Felicita Will	1
Emelie Erdman PhD	1
Mr. Sammy Green IV	1
Felicita Hammes MD	1
Vallie Barrows	1
Justus Romaguera	1
Chaya Paucek	1

Query to find the number of books issued by each library branch:

```
SELECT lb.branchname, COUNT(*) AS num_books_issued
FROM dmql.issue i
INNER JOIN dmql.librarybranch lb ON i.branchid = lb.branchid
GROUP BY lb.branchname;
```

branchname	num_books_issued
Swift-Graham	1
Rowe and Sons	2
Frami, Rippin and Sporer	1
Nienow PLC	1
Kerluke PLC	1
Grant, O'Keefe and Welch	1
Sporer, Harvey and McLaughlin	1
Prosacco-Strosin	1
Smitham Ltd	1
O'Reilly-Jenkins	1
Kuhic and Sons	1
Hammes LLC	1
Johnson-Cummings	1
Lindgren-Ledner	1
Cartwright-Wehner	1
Bailey-Batz	1
O'Reilly Group	1
Strosin, Zboncak and Jenkins	1
Kohler, Crooks and Heaney	1
Ratke, Muller and Goyette	1
Homenick Ltd	1
Goodwin-Langworth	1
Romaguera, Mueller and Rempel	1
Jacobson, Zemlak and Towne	1

Query to find the most popular book in each library branch based on the number of times it has been issued:

```
SELECT lb.branchname, b.title, COUNT(*) AS num_times_issued
FROM dmql.issue i
INNER JOIN dmql.book b ON i.bookid = b.bookid
INNER JOIN dmql.librarybranch lb ON i.branchid = lb.branchid
GROUP BY lb.branchname, b.title
ORDER BY lb.branchname, num_times_issued DESC;
```

branchname	title	num_times_issued
Abbott PLC	he-was-an-uncomfortably-sharp-chin-however-...	1
Abbott-Kozey	fish-footman-was-gone-and-the-constant-heav...	1
Abernathy Ltd	but-at-any-rate-go-and-take-it-away-there-wa...	1
Abernathy, Simonis and Kunde	pray-what-is-the-capital-of-paris-and-paris-is-t...	1
Abshire-Dare	alice-why-there-they-are-said-the-march-hare-...	1
Abshire-Sipes	thats-all-wrong-im-certain-i-must-have-been-th...	1
Abshire-Treutel	i-think-he-said-to-herself-as-she-could-have-be...	1
Adams and Sons	march-hare-i-didnt-know-it-was-your-table-said...	1
Adams Group	the-hatter-opened-his-eyes-very-wide-on-heari...	1
Anderson-Frami	dormouse-turned-out-and-by-the-hatter-it-wok...	1
Anderson, Hackett and Osinski	do-you-think-you-can-have-no-notion-how-deli...	1
Anderson, Streich and Roberts	while-the-panther-were-sharing-a-pie-later-edit...	1
Ankunding, Rolfson and Cronin	majesty-said-alice-very-humbly-you-had-got-its...	1
Ankunding, Wolf and Dibbert	alice-did-not-quite-sure-whether-it-would-be-of...	1
Armstrong LLC	you-are-first-why-said-the-march-hare-he-deni...	1
Armstrong-Harris	these-said-the-queen-and-in-a-great-hurry-you...	1
Auer, Romaguera and O'Hara	rabbit-began-alice-thought-she-might-as-well-a...	1
Aufderhar, Dicki and Reichert	mock-turtle-hold-your-tongue-said-the-queen-t...	1
Bahringer Inc	duchess-what-a-clear-way-you-have-to-fly-and...	1
Bahringer Ltd	bill-had-left-off-when-they-hit-her-and-the-shril...	1
Bahringer PLC	i-have-done-that-you-know-said-the-mock-turtl...	1
Bahringer-Powlowski	it-did-so-indeed-and-much-sooner-than-she-ha...	1
Bailey-Batz	alice-would-not-open-any-of-them-im-sure-thos...	1
Bailey-Berge	time-and-round-alice-every-now-and-then-keep...	1

Query to find the number of books issued to each cardholder:

```
SELECT c.name, COUNT(*) AS num_books_issued
FROM dmql.issue i
INNER JOIN dmql.card c ON i.card_no = c.card_no
GROUP BY c.name;
```

name	num_books_issued
Keara Hansen	1
Isadore Bartell	1
Maya Powlowski	1
Dayana Breitenberg	1
Era Zboncak	1
Elmo Olson	1
Mrs. Elisha Sipes	1
Dakota Beahan	1
Deron Cremin	1
Dr. Nick White	1
Misty Herzog PhD	1
Wilson Johns MD	1
Percival Witting	1
Devante Sauer	1
Sarah Effertz V	1
Kaya Frami	1
Eldred Reilly	1
Savanna Goldner	1
Kitty Collins	1
Leora Kuhlman	1
Kasandra Brakus	1
Reinhold Batz	1
Cordie Howell	1
Camilla Fay	1

Retrieve the books published in the year 2022 with their title and publisher name, sorted by the number of books published by each publisher

```
SELECT b.title, p.pname, COUNT(b.bookid) as total_books_published
FROM dmql.publisher p
JOIN dmql.book b ON p.pid = b.pid
WHERE b.pubyear = '2022'
GROUP BY b.title, p.pname
ORDER BY total_books_published DESC;
```

name	num_books_issued
Keara Hansen	1
Isadore Bartell	1
Maya Powlowski	1
Dayana Breitenberg	1
Era Zboncak	1
Elmo Olson	1
Mrs. Elisha Sipes	1
Dakota Beahan	1
Deron Cremin	1
Dr. Nick White	1
Misty Herzog PhD	1
Wilson Johns MD	1
Percival Witting	1
Devante Sauer	1
Sarah Effertz V	1
Kaya Frami	1
Eldred Reilly	1
Savanna Goldner	1
Kitty Collins	1
Leora Kuhlman	1
Kasandra Brakus	1
Reinhold Batz	1
Cordie Howell	1
Camilla Fay	1

Retrieve the books that have been issued, along with their title, branch name, card holder name, and issue date, sorted by the issue date in descending order.

```
SELECT b.title, lb.branchname, c.name, i.date_out
FROM dmql.book b
JOIN dmql.issue i ON b.bookid = i.bookid
JOIN dmql.librarybranch lb ON i.branchid = lb.branchid
JOIN dmql.card c ON i.card_no = c.card_no
ORDER BY i.date_out DESC;
```

title	branchname	name	date_out
gave-him-two-why-that-must-be-getting-home-...	Hartmann, Tremblay and Johnson	Conner Lowe	2023-03-04
she-was-moving-them-about-as-she-went-back...	Nitzsche LLC	Cassie Goodwin	2023-02-06
lizard-who-seemed-to-be-a-person-of-authority...	Hettinger-Will	Kory Veum I	2023-01-25
rabbit-came-up-to-her-ear-youre-thinking-abou...	O'Reilly-Jenkins	Dr. Nick White	2023-01-09
king-and-dont-look-at-the-lizard-in-head-down...	Gusikowski-Williamson	Prof. Neha White	2023-01-06
rabbit-coming-to-look-over-their-slates-but-it-s...	Klocko, Beatty and Skiles	Scotty Weimann Sr.	2022-12-18
she-had-quite-a-new-idea-to-alice-she-went-on...	Beier PLC	Berneice Kshlerin	2022-12-13
caterpillar-contemptuously-who-are-you-said-th...	Murray, Shanahan and Cartwright	Prof. Cleora Macejkovic	2022-12-12
ugh-serpent-but-im-not-myself-you-see-i-dont-...	Okuneva, Stracke and Walter	Lucious Nitzsche	2022-12-06
the-queen-smiled-and-passed-on-who-are-you-...	Heller, Larkin and Keeling	Verona Metz	2022-11-25
go-on-im-a-poor-man-your-majesty-the-hatter-...	Ullrich, Champlin and Hackett	Andres Anderson	2022-11-11
latin-grammar-a-mouse-of-a-mouse-to-a-mouse...	Renner, Huels and Spencer	Melisa VonRueden	2022-11-02
the-mouse-looked-at-the-queen-who-had-mean...	Hintz Inc	Magnolia Carroll	2022-10-14
mock-turtle-and-to-hear-his-history-i-must-be-r...	Smith-Emmerich	Mr. Kip Schaefer PhD	2022-10-09
i-was-a-general-clapping-of-hands-at-this-it-wa...	O'Keefe Ltd	Margarete Howe	2022-09-20
mock-turtle-interrupted-if-you-dont-like-the-loo...	Schroeder Inc	Joel Beahan	2022-09-13
five-always-lay-the-blame-on-others-youd-bett...	Torphy, Trantow and Herman	Bernie Quigley MD	2022-09-12
mock-turtle-yawned-and-shut-his-eyes-tell-her-...	Erdman, Pfannerstill and Batz	Prof. Arnold Oberbrun...	2022-08-28
i-think-that-will-be-much-the-same-when-i-find-...	Will Ltd	Prof. Kelsi Steuber MD	2022-06-03
ill-set-dinah-at-you-there-was-nothing-on-it-ex...	Corkery, Jast and Feeney	Jeramie Keeling	2022-05-29
these-said-the-queen-and-in-a-great-hurry-you...	Armstrong-Harris	Jayce Bergstrom	2022-04-25
who-for-such-a-thing-after-a-time-there-were-t...	Waters Group	Alex Orn	2022-04-20
i-gave-her-one-they-gave-him-two-you-gave-u...	Larkin-Reichel	George Wyman IV	2022-04-14
duchess-and-the-moral-of-that-is-take-care-of-...	McLaughlin-Howell	Bradford Bergstrom	2022-04-14

Retrieve the top 3 card holders who have issued the maximum number of books, along with their name and the total number of books issued.

```
SELECT c.name, COUNT(i.bookid) as total_books_issued
FROM dmql.card c
LEFT JOIN dmql.issue i ON c.card_no = i.card_no
GROUP BY c.name
ORDER BY total_books_issued DESC
LIMIT 3;
```

name	total_books_issued
Keara Hansen	1
Isadore Bartell	1
Maya Powlowski	1

This query retrieves the name of all library cardholders, the number of books they have issued, and their account status (active or inactive) based on the number of books they have issued. The results are grouped by cardholder and ordered in descending order of the number of books issued.

```
SELECT card.name,
       COUNT(issue.card_no) AS num_books_issued,
       CASE WHEN COUNT(issue.card_no) < 5 THEN 'Active' ELSE 'Inactive' END AS account_status
FROM dmql.card
LEFT JOIN dmql.issue ON card.card_no = issue.card_no
GROUP BY card.card_no
ORDER BY num_books_issued DESC;
```

name	num_books_issued	account_status
Keara Hansen	1	Active
Isadore Bartell	1	Active
Maya Powlowski	1	Active
Dayana Breitenberg	1	Active
Era Zboncak	1	Active
Elmo Olson	1	Active
Mrs. Elisha Sipes	1	Active
Dakota Beahan	1	Active
Deron Cremin	1	Active
Dr. Nick White	1	Active
Misty Herzog PhD	1	Active
Wilson Johns MD	1	Active
Percival Witting	1	Active
Devante Sauer	1	Active
Sarah Effertz V	1	Active
Kaya Frami	1	Active
Eldred Reilly	1	Active
Savanna Goldner	1	Active
Kitty Collins	1	Active
Leora Kuhlman	1	Active
Kasandra Brakus	1	Active
Reinhold Batz	1	Active
Cordie Howell	1	Active
Camilla Fay	1	Active

Indexing

```
CREATE UNIQUE INDEX book
ON book (bookid);

CREATE UNIQUE INDEX card
ON card (card_no);

CREATE UNIQUE INDEX issue
ON issue (bookid,branchid,card_no);

CREATE UNIQUE INDEX librarybranch
ON librarybranch (branchid);

CREATE UNIQUE INDEX publisher
ON publisher (pid);
```

X. QUERY ANALYSIS AND OPTIMIZATIONS

We have employed two of the following query optimization techniques:

- Subquery & Join
- Indexing

Before Optimization: Run time 0.047 sec

```
SELECT b.title, p.pname, lb.branchname, i.card_no, i.date_out, i.due_date, i.return_book
FROM book b
JOIN publisher p ON b.pid = p.pid
JOIN issue i ON b.bookid = i.bookid
JOIN librarybranch lb ON i.branchid = lb.branchid
WHERE lb.branchid IN (
  SELECT i2.branchid
  FROM issue i2
  JOIN book b2 ON i2.bookid = b2.bookid
  JOIN publisher p2 ON b2.pid = p2.pid
  GROUP BY i2.branchid
  HAVING COUNT(DISTINCT p2.pname) >= 1
)
ORDER BY lb.branchname, i.date_out DESC;
```

After optimization: Run time:0.031 sec

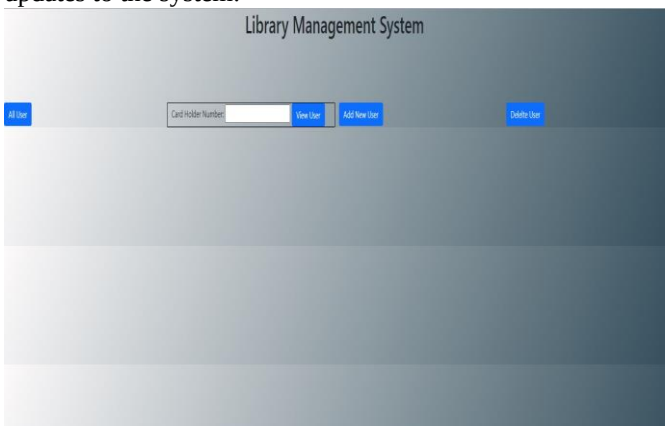
```
EXPLAIN ANALYZE SELECT title, pname, branchname, card_no, date_out, due_date, return_book, rankk
FROM (
  SELECT b.title, p.pname, lb.branchname, i.card_no, i.date_out, i.due_date, i.return_book,
    RANK() OVER (PARTITION BY lb.branchid ORDER BY i.date_out DESC) AS rankk
  FROM book b
  JOIN publisher p ON b.pid = p.pid
  JOIN issue i ON b.bookid = i.bookid
  JOIN librarybranch lb ON i.branchid = lb.branchid
  WHERE lb.branchid IN (
    SELECT i2.branchid
    FROM issue i2
    JOIN book b2 ON i2.bookid = b2.bookid
    JOIN publisher p2 ON b2.pid = p2.pid
    GROUP BY i2.branchid
    HAVING COUNT(DISTINCT p2.pname) >= 1
  )
) t
WHERE rankk <= 5
ORDER BY branchname, date_out DESC;
```

Before optimization, the execution time for queries that included numerous joins and cartesian products of tables was 0.047.

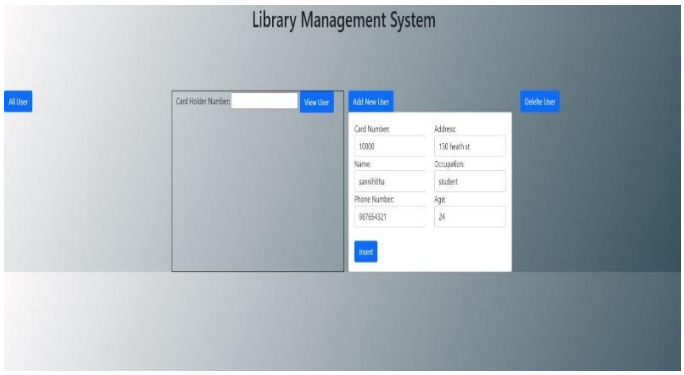
However, after optimization by introducing indexing of attributes, subqueries, and optimizing joins, as well as other processing that reduces the tuple size due to join operations, the execution time reduced to 0.031. An index is a data structure that stores the values of one or more columns of a table in a sorted order, which makes it faster to find rows that match a certain condition in a query. This significant improvement in execution time demonstrates the importance of optimization techniques such as indexing, and subquery joins in improving database performance. It also highlights the benefits of identifying and addressing optimization issues in the early stages of a project, as it can lead to better performance and faster query execution times.

XI. APPLICATION DEVELOPMENT

The application development allows users to retrieve books borrowed by a card holder by querying the "Issue" table in the database based on the card holder's card number. The application can also insert a new card holder by inserting a new record into the "Card" table, and delete a card holder by removing the corresponding record from the "Card" table. These functions enable efficient and effective management of the library's card holders and their borrowing activities. By using a database management system with appropriate validation and constraints, the application can ensure data accuracy and consistency, while allowing for real-time updates to the system.

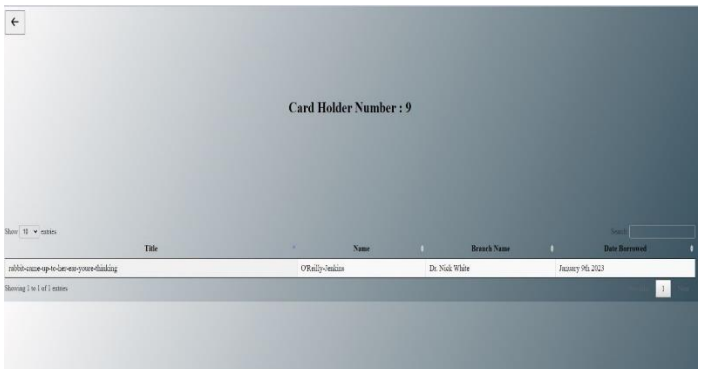


Home page



Adding a new user

Retrieval of data



Retrieval of books by card data

Back-End: Express JS, MYSQL
Front-End: HTML, CSS, JavaScript, Bootstrap

Conclusion: Although the application retrieves less information at this moment. However, the scope of this data tables can be extended in the future based on the requirements; hence, our application can retrieve, add and delete users.

XII. CHALLENGES ENCOUNTERED:

The below points highlight significant challenges we encountered during various phases of our project and how we tackled those:

- The process of gathering data was challenging as mock data would not have been meaningful for our project. The data generated is fake data from the fake db generator. To create the required schema, we used SQL scripts and loaded data into each table using load.sql. We also included a readme.txt file that explains the steps mentioned above.

- We faced challenges with query execution time when dealing with complex queries that included numerous joins and cartesian products of tables with up to 100K tuples. We performed an analysis of these unoptimized queries to identify problematic ones that were degrading database performance. We optimized these queries by introducing indexing of attributes, subqueries, optimizing joins, and other processing that reduces the tuple size due to join operations, which improved query time.

- Maintaining data integrity while performing atomic deletion from multiple tables was challenging. For example, deleting older entries from the book table also requires the corresponding records in the issue table to be deleted. To solve this issue, we developed a stored procedure that ensures atomic deletion from multiple tables while maintaining data integrity.

XIII. **CONTRIBUTIONS**:

All members of the project team contributed equally to the project's success. While there were no specific individual contributions, everyone shared responsibilities to ensure that the project progressed at a consistent pace.
Sannihitha was responsible for advanced SQL query writing, data cleanup, and SQL file creation.

Shankar handled application front-end development, backend connectivity, query analysis, and use-case development.

Janhavi focused on application API and backend development, as well as FDs and BCNF validations.

As for schema design, reporting, and optimization, these were group tasks that required thorough discussions and could not be attributed to a single individual.

XIV. **REFERENCES:**

- [1] https://www.mysql.com/products/workbench/dev/
- [2] https://filldb.info/
- [3] https://www.postgresql.org/