National Teachers College

629 J Nepomuceno, Quiapo, Manila, 1001 Metro Manila Bachelor
of Science in Information Technology

Digital Library Information System Enhancing Learning and Education for SDG 4

Presented by:
Castillo, Jake
Lawrence
Mamayson, John Michael
Malazarte, Kevince Lee
Soriano, Mark Leeroy

Presented to:
Justin Louise R.
Neypes

I. Introduction

1.1 Project Overview and UN SDG Target

The **Digital Library Information System (DLIS)** is a database project made to help the public and community libraries give **equal access to learning materials**. The system helps organize books, manage borrowing and returning, track available items in real time, and create reports for better library management.

This project goal is supporting the **United Nations Sustainable Development Goal (SDG) 4 – Quality Education**, targeting these:

- **SDG 4.1:** Provide fair and inclusive education for everyone
- **SDG 4.3:** Increase access to affordable learning materials
- **SDG 4.5:** Reduce differences in access to education

By managing books and tracking how resources are used, the system helps improve community learning, especially for people who have limited access to formal education.

1.2 Problem Statement

Many libraries struggle to manage their learning resources in an organized and reliable way. Book records are often incomplete or outdated, making it hard to know which materials are available or missing. Borrowing and return transactions are sometimes not recorded properly, which causes confusion for both librarians and users.

In addition, libraries have difficulty monitoring which books are frequently used and which resources are rarely accessed. Without accurate data, it is hard to improve library services or decide what new learning materials are needed by the community.

The proposed **Digital Library Information System** solves these problems by providing a clear and organized database for managing books, users, and transactions. This system helps ensure better access to educational resources and supports quality education within the community.

II. Requirements & Analysis

2.1. Functional Requirements and Non-Functional Requirements (List of features, e.g., FR1, FR2)

Functional Requirements (FR)

ID	Requirement	Alignment
FR1	The system loads more than 50 starting records from the SQL file. These records include books, members, categories, inventory, and borrowing data. The data is added using several **INSERT** statements in the SQL file.	DDL/DML
FR2	The system uses basic DBMS features included in the file. These include **foreign keys** to connect books with categories and borrowings with books and members, **CHECK constraints** to make sure the published year is 1800 or later and that copies are not negative, and a **trigger** called ***trg_no_negative_inventory*** to prevent negative inventory values.	Core Concepts
FR3	The SQL file includes a **stored procedure** called ***borrow_book***. It handles the full borrowing process by starting a transaction, locking the needed records, showing error messages when needed, and saving the changes. This helps make sure the transaction follows **ACID rules**.	Finals Concepts
FR4	The system provides **three SDG-related reports** using SQL views: ***v_category_demand***, ***v_most_borrowed***, and ***v_overdue_members***. These views are included in the file and are used to generate reports from the data.	DQL/Reporting

Non-Functional Requirements (NFR)

ID	Requirement	Metric
NFR1	The database stops invalid data by using built-in rules like **CHECK constraints**, **FOREIGN KEYS**, and a **trigger** that prevents inventory from going negative. The stored procedure also shows easy-to-understand error messages using **SIGNAL**.	Error messages should be informative.
NFR2	The SQL objects in the file are organized into separate parts: **tables, constraints, stored procedures, and views**. The names are clear, like ***v_category_demand*** and ***borrow_book***, which makes the database easy to understand and maintain.	Use of comments and logical VIEW definitions.
NFR3	The views use **simple joins** and **indexed columns** like ***book_id***, ***member_id***, and ***category_id***, which helps the three reports run quickly even when there are many records.	Reports should return results in under 1 second on standard hardware with test data.

2.2. Data Requirements (Description of input data structure and

size) Minimum record count: **≥50 records across all tables**

Input data includes:

- Members (20)
- Books (35)
- Categories (10)
- Inventory for all books
- Borrowings (25+ sample

transactions) Data must follow validation

rules:

- NOT NULL, CHECK, and FK constraints
- Unique ISBN constraints

2.3. Schema Normalization Analysis: Justify the level of normalization achieved (3NF/BCNF) for the most complex tables, listing functional dependencies.

Books Table (Most complex)

Functional Dependencies

- **book_id → title, author, category_id, isbn**
- **isbn → title, author** (ISBN uniquely identifies a book)
- **category_id → category_name** (through Categories table)

1NF

- All attributes atomic
- No repeating groups

2NF

- No partial dependencies (PK is a single attribute)

3NF

- No transitive dependencies inside the table

Conclusion:

- The books table is in BCNF because every determinant is a candidate key.

Categories Table

Functional

Dependencies

- category_id → category_name

1NF

- All attributes atomic
- No repeating groups

2NF

- No partial dependencies (PK is a single attribute)

3NF

- No transitive dependencies inside the table

Conclusion:

- The categories table is in BCNF because every determinant is a candidate key.

Inventory Table

Functional

Dependencies

- **inventory_id → book_id, total_copies, available_copies**
- **book_id → total_copies, available_copies** (if one-to-one)

1NF

- All attributes atomic
- No repeating groups

2NF

- No partial dependencies (PK is a single attribute)

3NF

- No transitive dependencies inside the table

Conclusion:

- Inventory table is in BCNF because every determinant is a candidate key.

Members Table

Functional

Dependencies

- member_id → full_name, address, date_registered

1NF

- All attributes atomic
- No repeating groups

2NF

- No partial dependencies (PK is a single attribute)

3NF

- No transitive dependencies inside the table

Conclusion:

- Members table is in BCNF because every determinant is a candidate key.

Borrowings Table

Functional Dependencies

- borrow_id → member_id, book_id, date_borrowed, due_date, date_returned

1NF

- All attributes atomic
- No repeating groups

2NF

- No partial dependencies (PK is a single attribute)

3NF

- No transitive dependencies inside the table

Conclusion:

The borrowing table is in BCNF because every determinant is a candidate key.

III. Design Specification

3.1. Core DBMS Concepts Used (The Five):

For each of the required DBMS concepts, include a section detailing:

- **Justification:** Why was this specific concept chosen (e.g., *Why was a TRIGGER used instead of a CHECK constraint for this business rule?*).
- **Implementation Details**: Provide the exact SQL code snippet for the concept (Stored Procedure, Trigger, VIEW, etc.)

Stored Procedure

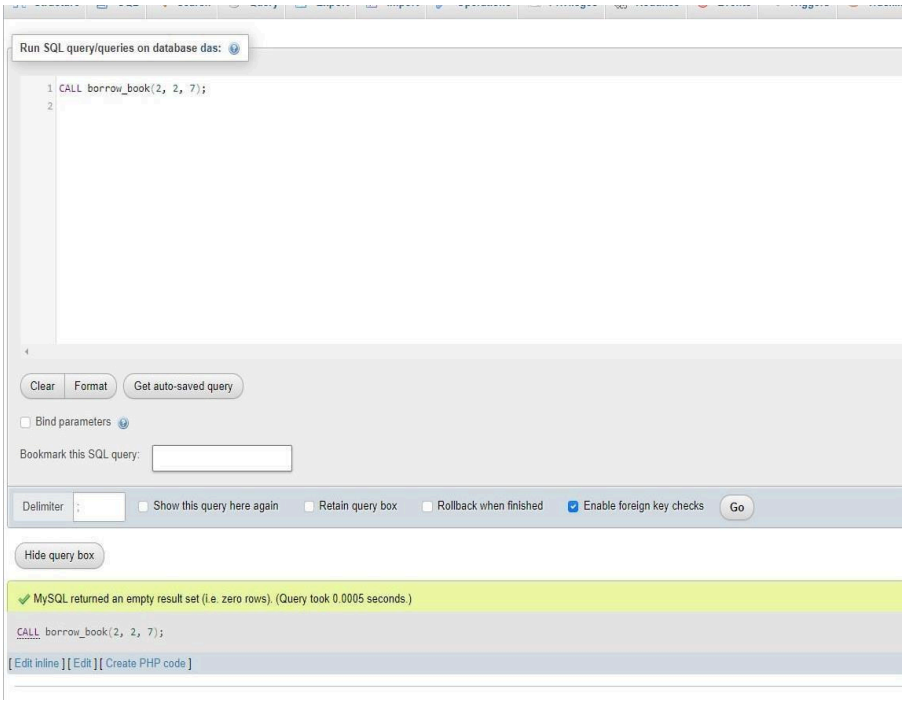
CODE:

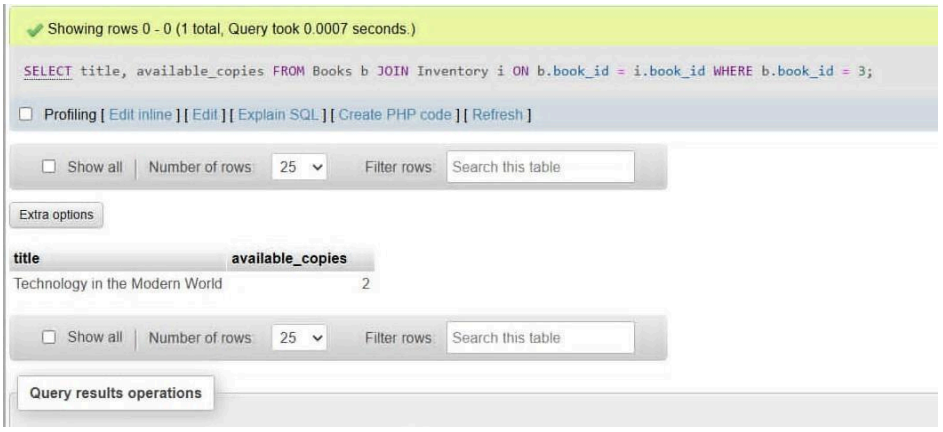
```
DELIMITER $$

CREATE PROCEDURE borrow_book (
   IN p_member_id INT,
   IN p_book_id INT,
   IN p_days INT
)
BEGIN
   INSERT INTO borrowings (member_id, book_id, borrow_date, due_date)
   VALUES (
      p_member_id,
      p_book_id,
      CURDATE(),
      DATE_ADD(CURDATE(), INTERVAL p_days DAY)
   );
END$$

CREATE PROCEDURE return_book (
   IN p_borrowing_id INT
)
BEGIN
   UPDATE borrowings
   SET return_date = CURDATE()
   WHERE borrowing_id = p_borrowing_id
   AND return_date IS NULL;
END$$

DELIMITER ;
```

A **stored procedure** is a prepared SQL code that is saved inside the database, allowing it to be reused many times.
Instead of writing the same SQL query repeatedly, you can save it as a stored procedure and simply call it whenever you need to run that operation.

Stored procedures help make database work **faster** because the code is already compiled and ready to run. They also make the system **more organized** since complex queries are stored in one place instead of being

scattered in the application.
In addition, stored procedures help **reduce errors** and can improve **security**, because users only need permission to run the procedure rather than access the raw tables directly.





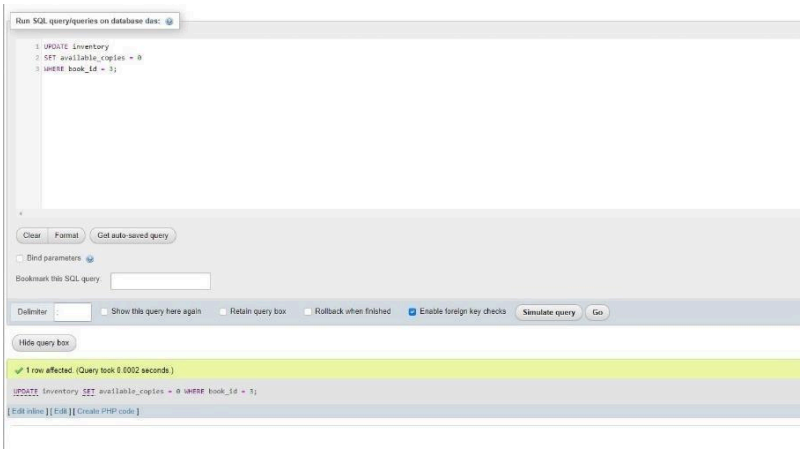
Rollback

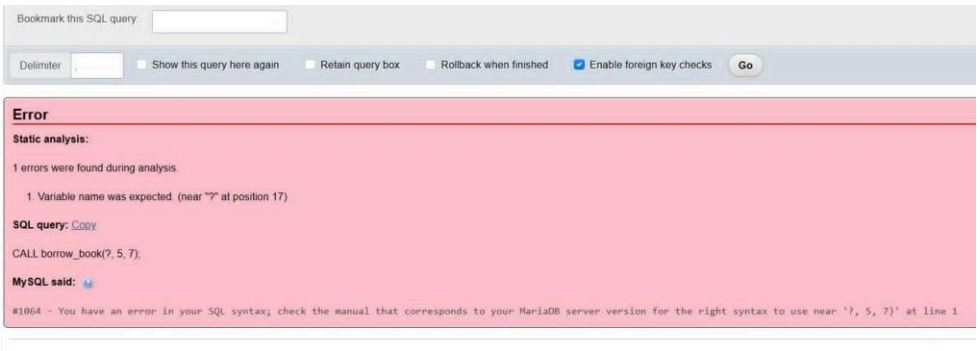
CODE:

START TRANSACTION;

CALL borrow_book(2, 5, 7);

-ROLLBACK;





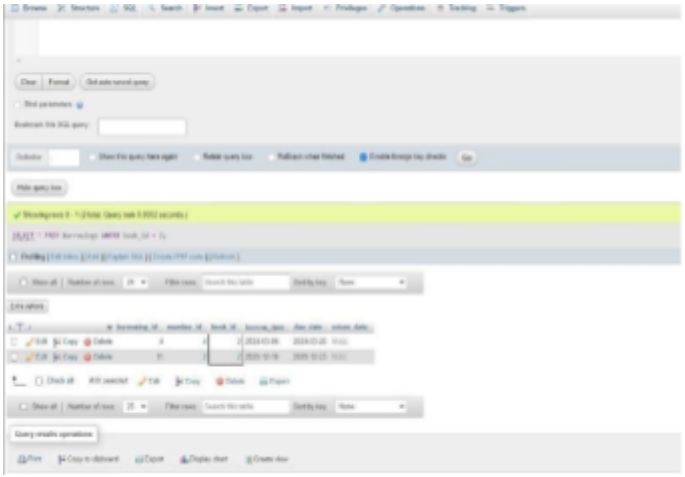
The image shows two SQL operations in a database interface like phpMyAdmin—one successful and one with an error. The first query updates the Inventory table, setting available_copies to 0 for the book with book_id = 5. It runs successfully and affects one row, showing that the system correctly processed the update.

The second query tries to run the borrow_book stored procedure with parameters 5 and 7, probably for a book ID and member ID. However, it fails because of a syntax error. The system reports an “Unclosed quote” near 5, which means the query might be missing proper formatting, like quotation marks or parentheses. This error stops the procedure from running, showing how SQL enforces strict rules to keep the data safe and correct.

Trigger

CODE

```
DELIMITER $$

CREATE TRIGGER trg_before_borrow
BEFORE INSERT ON borrowings
FOR EACH ROW
BEGIN
   DECLARE copies INT;
   SELECT available_copies INTO copies
   FROM inventory
   WHERE book_id = NEW.book_id;

   IF copies <= 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Book not available';
   END IF;
END$$

CREATE TRIGGER trg_after_borrow
AFTER INSERT ON borrowings
FOR EACH ROW
BEGIN
   UPDATE inventory
   SET available_copies = available_copies - 1
   WHERE book_id = NEW.book_id;
END$$

CREATE TRIGGER trg_after_return
AFTER UPDATE ON borrowings
FOR EACH ROW
BEGIN
   IF OLD.return_date IS NULL AND NEW.return_date IS NOT NULL THEN
      UPDATE inventory
      SET available_copies = available_copies + 1
      WHERE book_id = NEW.book_id;
   END IF;
END$$

DELIMITER ;
```

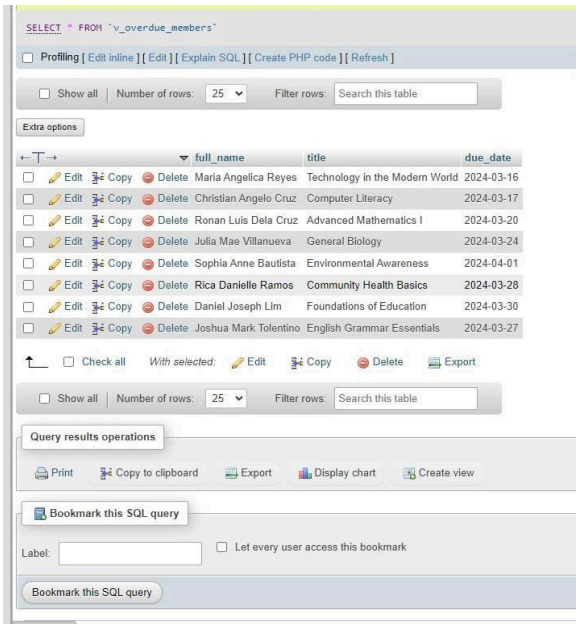


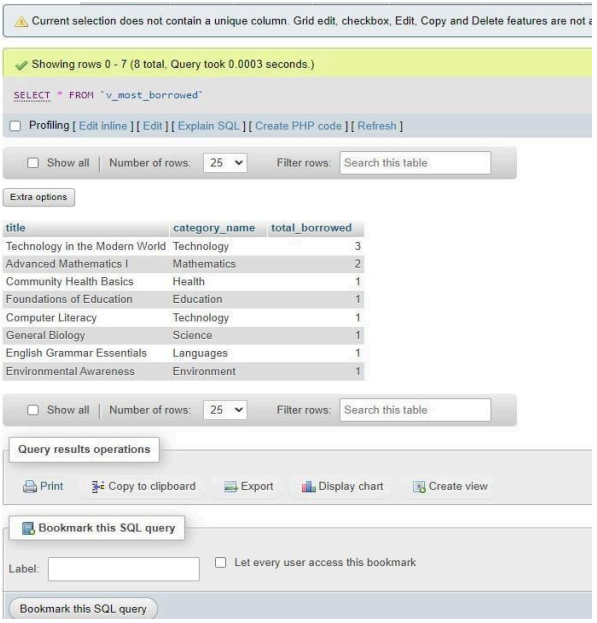
The database system for the Library Information System demonstrates strong integrity controls through its use of both successful data retrieval and explicit constraint enforcement. An essential function involves tracking loaned materials, such as the successful execution of a query that fetched all borrowing records for book_id = 5, which showed that member_id = 12 borrowed the book on 2024-03-13, and the **NULL** value in the date_returned column confirmed the book was still on loan. Crucially, the system protects itself against invalid operations by rejecting an UPDATE statement that attempted to set available_copies = -1 in the Inventory table, returning the error message **"Inventory cannot go negative"**. This protective mechanism, likely implemented via a database trigger or constraint, is vital for maintaining the logical consistency and accuracy of the library's stock records.

Views

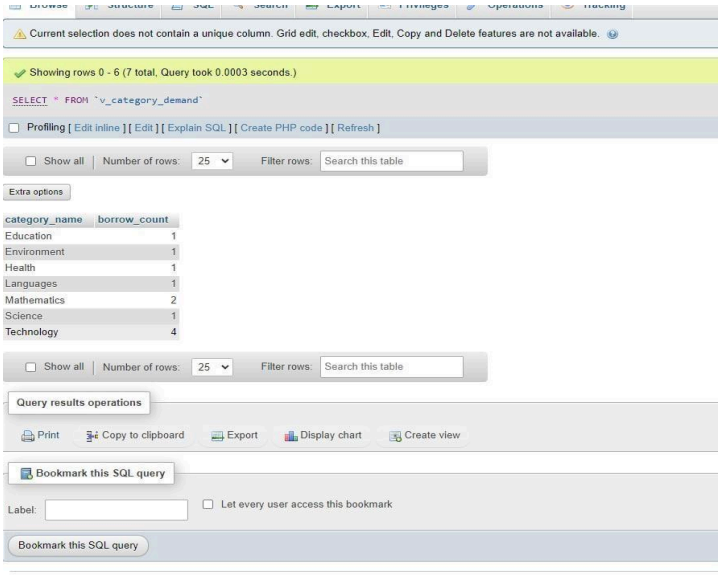
CODE:

```
CREATE VIEW v_most_borrowed AS
SELECT
    b.title,
    c.category_name,
    COUNT(br.borrowing_id) AS total_borrowed
FROM borrowings br
JOIN books b ON br.book_id = b.book_id
JOIN categories c ON b.category_id = c.category_id
GROUP BY b.book_id;

CREATE VIEW v_category_demand AS
SELECT
    c.category_name,
    COUNT(br.borrowing_id) AS borrow_count
FROM borrowings br
JOIN books b ON br.book_id = b.book_id
JOIN categories c ON b.category_id = c.category_id
GROUP BY c.category_name;

CREATE VIEW v_overdue_members AS
SELECT
    m.full_name,
    b.title,
    br.due_date
FROM borrowings br
JOIN members m ON br.member_id = m.member_id
JOIN books b ON br.book_id = b.book_id
WHERE br.return_date IS NULL
AND br.due_date < CURDATE();
```





hii

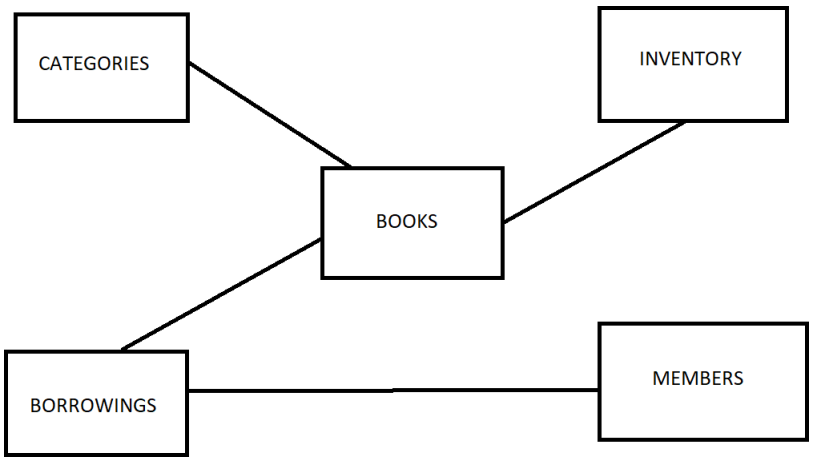


The image provided showcases examples of how SQL views are used to organize and summarize data within a Library Information System. This approach is highly effective for monitoring usage, managing inventory, and supporting data-driven decision-making, which aligns with the goal of SDG 4: Quality Education by promoting access to learning resources. These SQL views collectively streamline the process for librarians and administrators to monitor borrowing trends, manage physical inventory, and make informed decisions based on actual usage patterns.

3.2. ER Diagram: Include the Conceptual, Logical, and Physical ERD showing all entities, attributes, primary keys, foreign keys, and cardinalities.

Conceptual ERD:

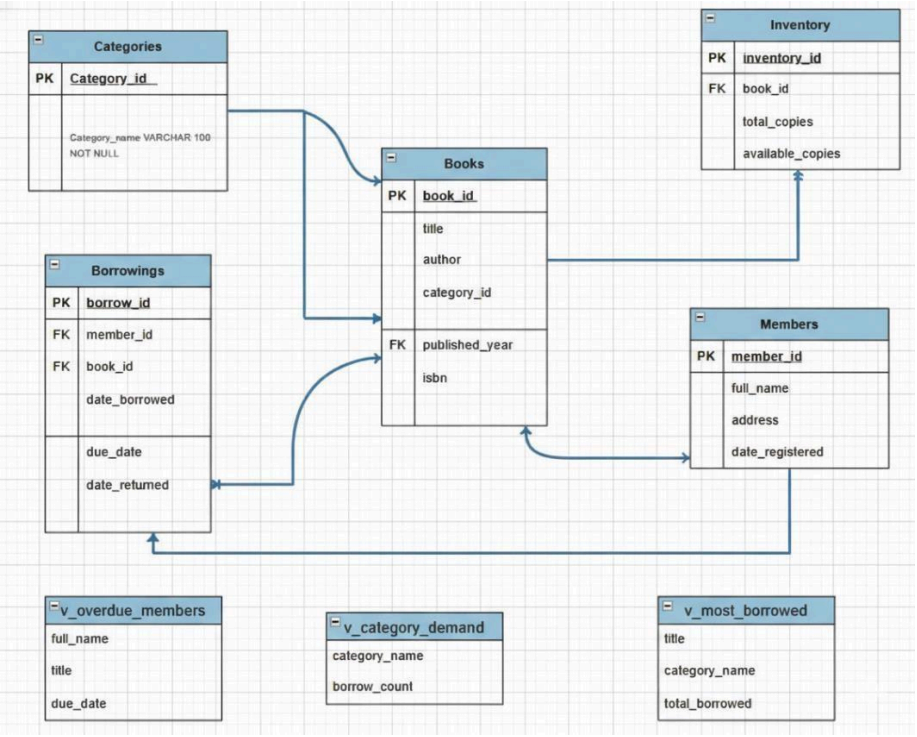
Entity Diagram Relationship



Logical ERD: In the **Logical ERD**, all **primary keys** and **foreign keys** have been included. For example:

- **Books table:** book_id (PK), title, author, published_year, category_id (FK)
- **Members table:** member_id (PK), name, contact_info
- **Borrowings table:** borrow_id (PK), book_id (FK), member_id (FK), borrow_date, return_date

This ensures that each table has a unique identifier and proper relationships between tables are maintained.



Physical ERD:

The Physical Entity-Relationship Diagram (ERD) represents the physical structure of a digital library system aligned with SDG 4: Quality Education. It is designed to manage core operations such as book borrowing, inventory tracking, member registration, and category-based analytics.

Core Tables and Their Roles:

- `members` – Stores personal details of registered library users, including full name, address, and registration date.
- `books` – Contains metadata for each book, such as title, author, category, publication year, and ISBN.
- `categories` – Definesthe genre or classification of books, linked to the `books` table via `category_id`.
- `inventory` – Tracks the total and available copies of each book, ensuring real-time monitoring of stock levels.
- `borrowings` – Logs every borrowing transaction, linking members to books and recording borrow, due, and return dates.

Views for Reporting and Insights:

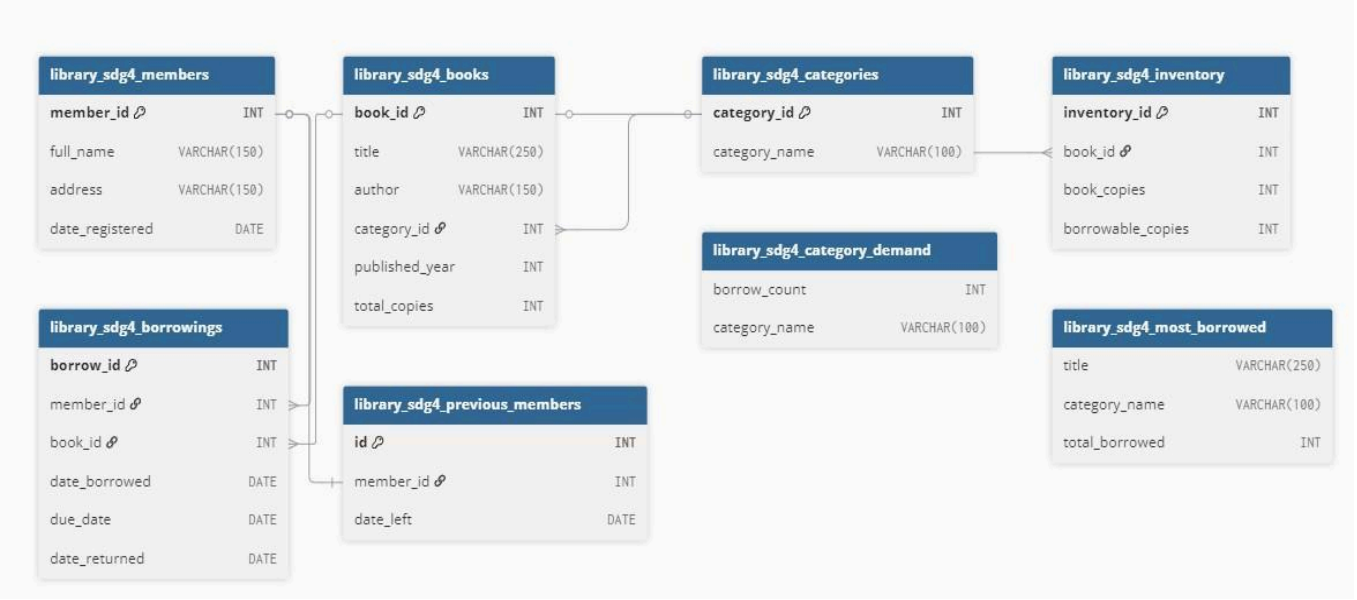
- `v_category_demand` – Aggregates borrow counts per category to identify high-demand genres.
- `v_most_borrowed` – Lists the most frequently borrowed books, useful for acquisition planning.
- `v_overdue_members` – Flags members with overdue books, supporting follow-ups and accountability.

Relationships and Flow:

- `borrowings` connects to both `members` and `books` via foreign keys, enabling traceability of each transaction.
- `books` link to `categories` for classificationand to `inventory` for availabilitytracking.
- Views are derived from these base tables to support decision-making and performance monitoring.

Purpose and Impact:

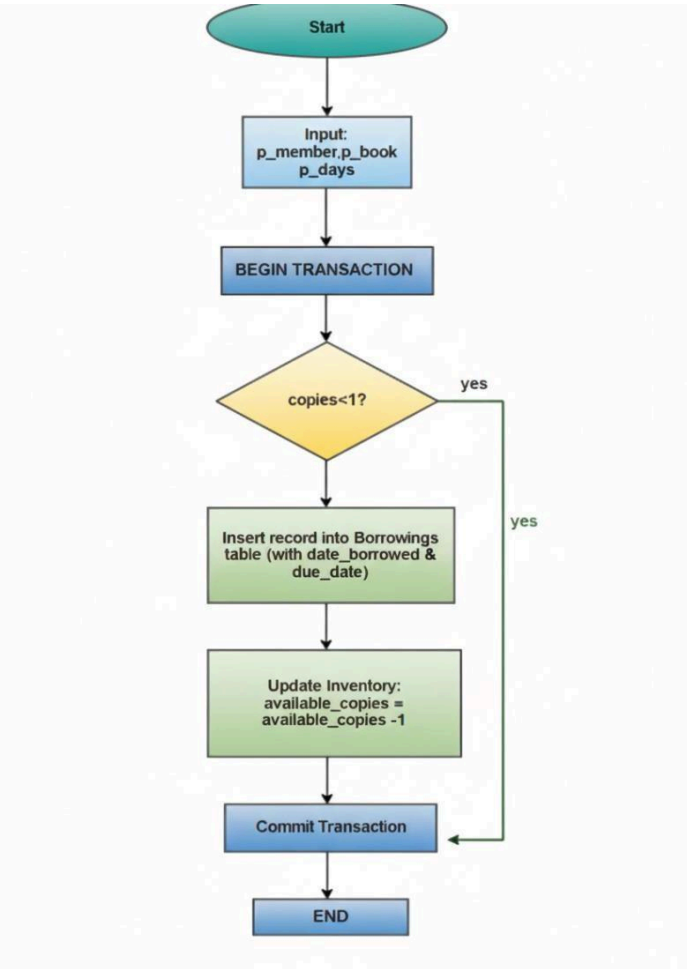
This ERD helps the library run more efficiently and supports access to learning materials. It allows librarians to make decisions based on data, track which books are used most, manage inventory, and make sure books are returned on time. By doing this, it supports SDG 4: Quality Education.



3.3. Transaction Flowchart: Include the Flowchart for the system's core transactional Stored Procedure (FR3), highlighting the BEGIN TRANSACTION and COMMIT/ROLLBACK points.

The flowchart shows the borrowing process handled by the ***borrow_book*** stored procedure. The system receives the member ID, book ID, and number of borrowing days, and then starts a transaction to make sure all steps are done safely together. It checks if the requested book has available copies using a locked selection to prevent other changes at the same time. If no copies are available, the system stops and shows an error. If copies are available, the system adds a record to the Borrowings table and reduces the number of available copies in the Inventory by one. Finally, the transaction is committed, saving all changes permanently and keeping the data accurate. This process ensures that borrowing is safe, consistent, and free from errors or data conflicts.

Flowchart:



IV. Conclusion and Contributions

5.1 Conclusion

The **Digital Library Information System** successfully demonstrates important DBMS principles while enhancing learning and education to support **SDG 4 – Quality Education**. The system includes:

- A fully normalized database schema (3NF/BCNF)
- Transactions that follow ACID rules
- Enforcement of data integrity using triggers and foreign keys
- SDG-focused reports and analytics, such as most borrowed books and popular categories
- Organized, modular design with professional version control

By streamlining library operations, improving access to learning materials, and providing data-driven insights, the system addresses real-world challenges in community libraries and promotes **inclusive, equitable, and quality education**.

5.2 Individual Contributions

This section details each team member's assigned module or script, showing their specific role in developing the **Digital Library Information System**. Each contribution supports the system's goal of enhancing learning, improving library management, and promoting SDG 4.

MEMBERS:

CASTILLO, JAKE LAWRENCE - Normalization 3NF/BCNF
MAMAYSON, JOHN MICHAEL - Complex DQL
MALAZARTE, KEVINCE LEE – TRIGGER, STORED PROCEDURE
SORIANO, MARK LEEROY - ERD