Analytical and Technical Specification for a Library Management System

Section 1: Strategic Framework and System Mandate

1.1. Executive Summary and Project Context

This project's objective is to design, document, and implement a relational database for a Library Management System. The system is built upon five core relations: Author, Book, BookCopy, Member, and Loan. This model provides a logically sound foundation for inventory tracking, search capabilities, and data integrity. This approach was necessary to implement the project's required "more comprehensive features" and ensure the system is scalable.

1.2. Functional Requirements (FR) Analysis

Functional requirements define what the system must do. The five requirements below are the operational mandate for this system.

- **FR1: Book Management:** Librarians must be able to add, update, and remove book titles from the library's catalog. For each title, they must also manage the inventory of individual physical copies, including adding new copies or marking existing ones as lost or under maintenance.
- **FR2: Member Management:** Librarians must be able to register new library members, update their information, and retrieve their complete borrowing history, including current and past loans.
- **FR3: Loan Management:** The system must track individual physical copies of books. A member can check out a specific, available copy. The system must record the checkout date, calculate a due date (e.g., 21 days), and record the actual return date. The status of the specific copy must be updated to 'On Loan'.
- **FR4: Search and Discovery:** All users must be able to search the catalog for books by title or author. Search results must display the book's metadata and a real-time summary of its inventory, such as '2 copies owned, 1 available'.
- **FR5: Reporting:** Librarians must be able to generate a report of all currently overdue book copies. This report must include the book's title, the member's name, and the loan's due date.

The functional requirements, particularly FR3 and FR4, necessitate a data model that can track individual physical items and report on their availability. This is achieved through the use of distinct Book and BookCopy entities, which separate a book's metadata from its physical instances.

1.3. Non-Functional Requirements (NFR) Analysis

Non-functional requirements define the system's quality attributes and operational constraints on how it should operate. They are the blueprint for a reliable, consistent, and usable system.

- **NFR1 (Data Integrity):** The database must enforce business rules to prevent invalid data. This includes database checks to prevent a book copy from being marked as returned before it was borrowed and to restrict status fields to predefined values ('Available', 'On Loan', 'Maintenance').

- **NFR2 (Data Consistency):** All data must remain consistent. This is achieved using foreign key constraints to maintain referential integrity. For example, deleting an author must be handled to avoid orphaning book records, using constraints like ON DELETE RESTRICT or ON DELETE SET NULL.
- **NFR3 (Usability):** The command-line interface (CLI) must provide clear and actionable feedback. This includes confirmation messages for successful operations ("Checkout successful for copy #123") and informative error messages for failures ("Error: This copy is currently on loan.").

These NFRs translate directly into technical implementations. NFR1 is implemented using CHECK constraints. NFR2 is implemented using FOREIGN KEY definitions with ON DELETE and ON UPDATE clauses. NFR3, while part of the application layer, relies on the database returning predictable errors so the CLI can provide meaningful feedback. By defining these rules, the database itself enforces data integrity, creating a more resilient system.

Section 2: Conceptual Architecture and Data Modeling

2.1. Justification of the Five-Entity Model

Justifying the entities is crucial, especially the separation of the Book and BookCopy concepts. This distinction is the core of the design.

- **Book Entity:** This entity stores title-level information. It holds metadata that is common to all copies of a particular title, such as title, isbn, and publication_year, and its relationship to an Author. An ISBN, for example, identifies a specific edition of a work.
- **BookCopy Entity:** This entity represents an individual, physical copy that the library owns. Each record corresponds to a unique, loanable copy of a book. Its attributes, such as copy_id (Primary Key) and status (e.g., 'Available', 'On Loan'), are specific to that individual copy.

The introduction of the BookCopy entity allows the system to manage inventory at the individual item level. A member borrows a specific, barcoded copy of a book from the shelf. This distinction impacts the entire system.

The first consequence of this change affects the Loan entity. A loan must be associated with a physical item, so the Loan table's foreign key must be copy_id, not book_id. This is a necessary consequence of the model.

This change also enables advanced functional requirements. With loans tied to a copy_id, the system can update a copy's status to 'On Loan', fulfilling FR3. By joining the Book and BookCopy tables, the system can also count total and available copies for any title, fulfilling FR4.

Finally, this model ensures scalability. The architecture supports future features like tracking a copy's physical condition or shelf location. The Book/BookCopy separation is a foundational design choice that improves logic, enables functionality, and supports future growth.

2.2. Analysis of Relationships, Cardinality, and Modality

Relationships are defined by their cardinality (how many) and modality (optional or mandatory). These rules model the library's business logic.

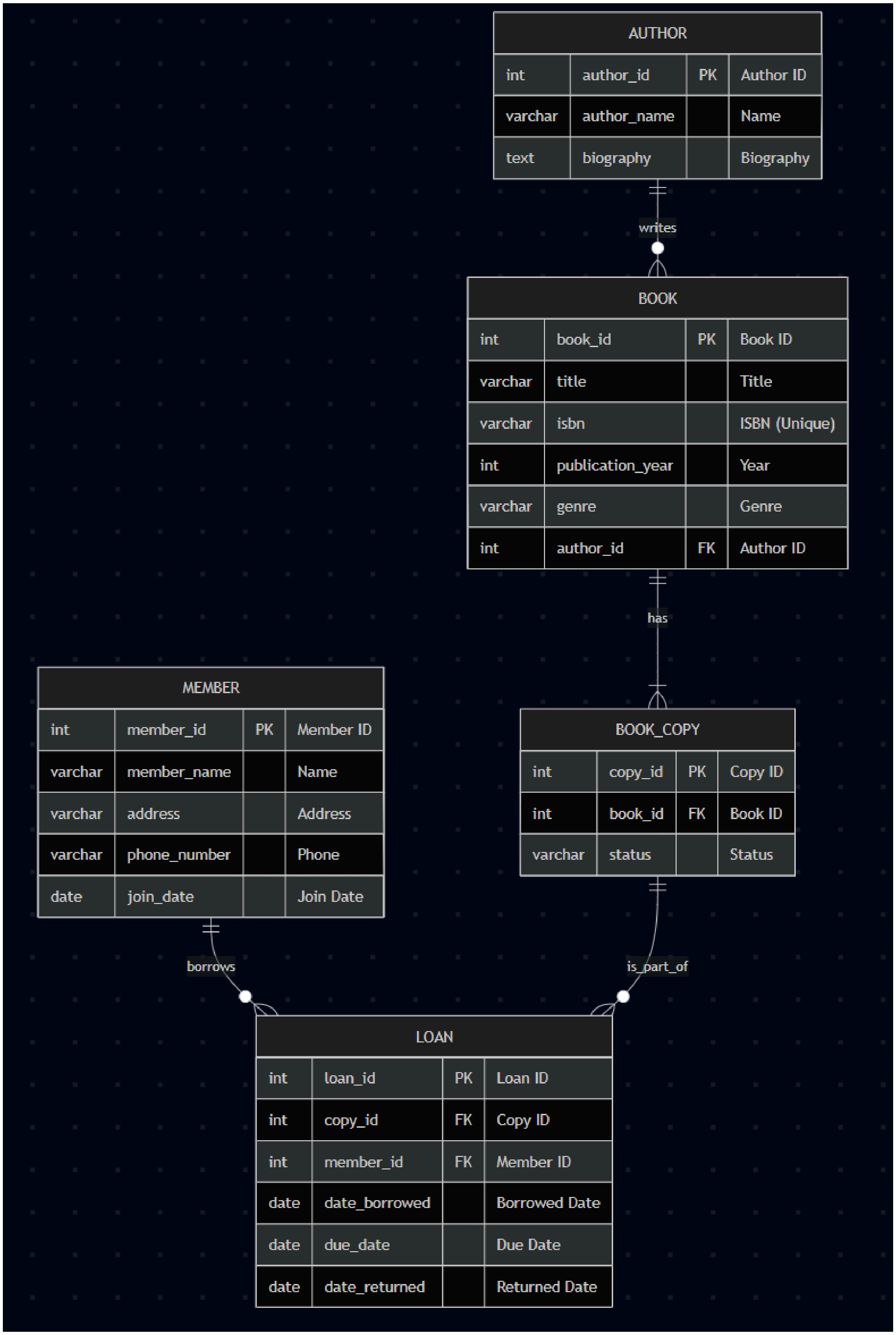
- **Book to BookCopy:** A **one-to-many** relationship. One Book can have many BookCopy records. Each BookCopy must belong to exactly one Book. This models the library owning multiple copies of one title.
- **Member to Loan:** A **one-to-many** relationship. One Member can have zero or many Loan records. Each Loan must be associated with exactly one Member.
- **BookCopy to Loan:** A **one-to-many** relationship. Over time, one BookCopy can have many Loan records. At any moment, however, it can have at most one active loan (where date_returned is NULL). Each Loan record is for exactly one BookCopy.

2.3. Formal Use Case Scenarios

Use cases bridge the gap between requirements and design, showing how actors will interact with the system.

- **Use Case 1: Librarian Adds New Inventory**
  - **Actor:** Librarian
  - **Description:** The librarian adds a new book title (e.g., 'Neuromancer') to the catalog. The system creates a record in the Book table. The librarian then specifies the number of physical copies being added. The system creates that number of records in the BookCopy table, links them to the Book record, and sets their status to 'Available'.
- **Use Case 2: Member Searches for a Book**
  - **Actor:** Member
  - **Description:** A member searches for "Dune". The system queries the database, joining Book and BookCopy. It returns the book's metadata and an inventory summary, such as: "3 copies owned, 1 available for loan."
- **Use Case 3: Member Checks Out a Book Copy**
  - **Actor:** Member (facilitated by Librarian via CLI)
  - **Description:** A member wants to check out 'Dune'. The librarian finds an available copy_id. The system validates the copy's status is 'Available'. It then creates a new record in the Loan table, linking the member_id and copy_id. The system also updates the BookCopy record's status to 'On Loan'.

2.4. ER diagram



AUTHOR

int	author_id	PK	Author ID
varchar	author_name		Name
text	biography		Biography

BOOK

int	book_id	PK	Book ID
varchar	title		Title
varchar	isbn		ISBN (Unique)
int	publication_year		Year
varchar	genre		Genre
int	author_id	FK	Author ID

MEMBER

int	member_id	PK	Member ID
varchar	member_name		Name
varchar	address		Address
varchar	phone_number		Phone
date	join_date		Join Date

BOOK_COPY

int	copy_id	PK	Copy ID
int	book_id	FK	Book ID
varchar	status		Status

LOAN

int	loan_id	PK	Loan ID
int	copy_id	FK	Copy ID
int	member_id	FK	Member ID
date	date_borrowed		Borrowed Date
date	due_date		Due Date
date	date_returned		Returned Date

Section 3: Formal Relational Schema and Integrity Constraints

3.1. Detailed Schema Definitions

The five tables in the relational schema are defined below. The definitions include columns, data types, and constraints that implement the business rules and non-functional requirements.

1. Author

- author_id: INT, PRIMARY KEY, AUTO_INCREMENT - Uniquely identifies each author.
- author_name: VARCHAR(255), NOT NULL - The author's name.
- biography: TEXT - A biography of the author.

2. Book

- book_id: INT, PRIMARY KEY, AUTO_INCREMENT - Uniquely identifies each book title.
- title: VARCHAR(255), NOT NULL - The book's title.
- isbn: VARCHAR(13), UNIQUE - The unique ISBN for the book edition.
- publication_year: INT - The year of publication.
- genre: VARCHAR(100) - The book's genre.
- author_id: INT, FOREIGN KEY REFERENCES Author(author_id) ON DELETE SET NULL - Links to Author. If an author is deleted, the book record remains but the author link is set to NULL (NFR2).

3. BookCopy

- copy_id: INT, PRIMARY KEY, AUTO_INCREMENT - Uniquely identifies each physical copy.
- book_id: INT, NOT NULL, FOREIGN KEY REFERENCES Book(book_id) ON DELETE CASCADE - Links to Book. If a book title is deleted, all its physical copies are also deleted (NFR2).
- status: VARCHAR(20), NOT NULL, CHECK (status IN ('Available', 'On Loan', 'Maintenance', 'Lost')) - Enforces NFR1 by ensuring the status is always one of the four valid values.

4. Member

- member_id: INT, PRIMARY KEY, AUTO_INCREMENT - Uniquely identifies each member.
- member_name: VARCHAR(255), NOT NULL - The member's name.
- address: VARCHAR(255) - The member's address.
- phone_number: VARCHAR(15) - The member's phone number.
- join_date: DATE, NOT NULL, DEFAULT (CURRENT_DATE) - The date the member joined, defaults to the current date.

5. Loan

- loan_id: INT, PRIMARY KEY, AUTO_INCREMENT - Uniquely identifies each loan.
- copy_id: INT, NOT NULL, FOREIGN KEY REFERENCES BookCopy(copy_id) ON DELETE RESTRICT - Links to the loaned BookCopy. A copy cannot be deleted if it has a loan history (NFR2).
- member_id: INT, NOT NULL, FOREIGN KEY REFERENCES Member(member_id) ON DELETE

RESTRICT - Links to the Member. A member cannot be deleted if they have a loan history.

- date_borrowed: DATE, NOT NULL - The date the copy was borrowed.
- due_date: DATE, NOT NULL - The date the copy is due.
- date_returned: DATE, NULL - The date the copy was returned; NULL for active loans.
- CHECK (date_returned IS NULL OR date_returned >= date_borrowed) - Enforces NFR1 by ensuring a book cannot be returned before it was borrowed.

3.2. Normalization Analysis

The proposed five-table schema is in **Third Normal Form (3NF)**. The design follows normalization principles to reduce data redundancy and prevent data anomalies.

- **First Normal Form (1NF):** All table cells hold atomic values, and there are no repeating groups.
- **Second Normal Form (2NF):** There are no partial dependencies, as all non-key attributes depend on the entire single-column primary key.
- **Third Normal Form (3NF):** There are no transitive dependencies, as no non-key attribute depends on another non-key attribute.

Adhering to 3NF ensures data integrity and makes the database more efficient and maintainable.