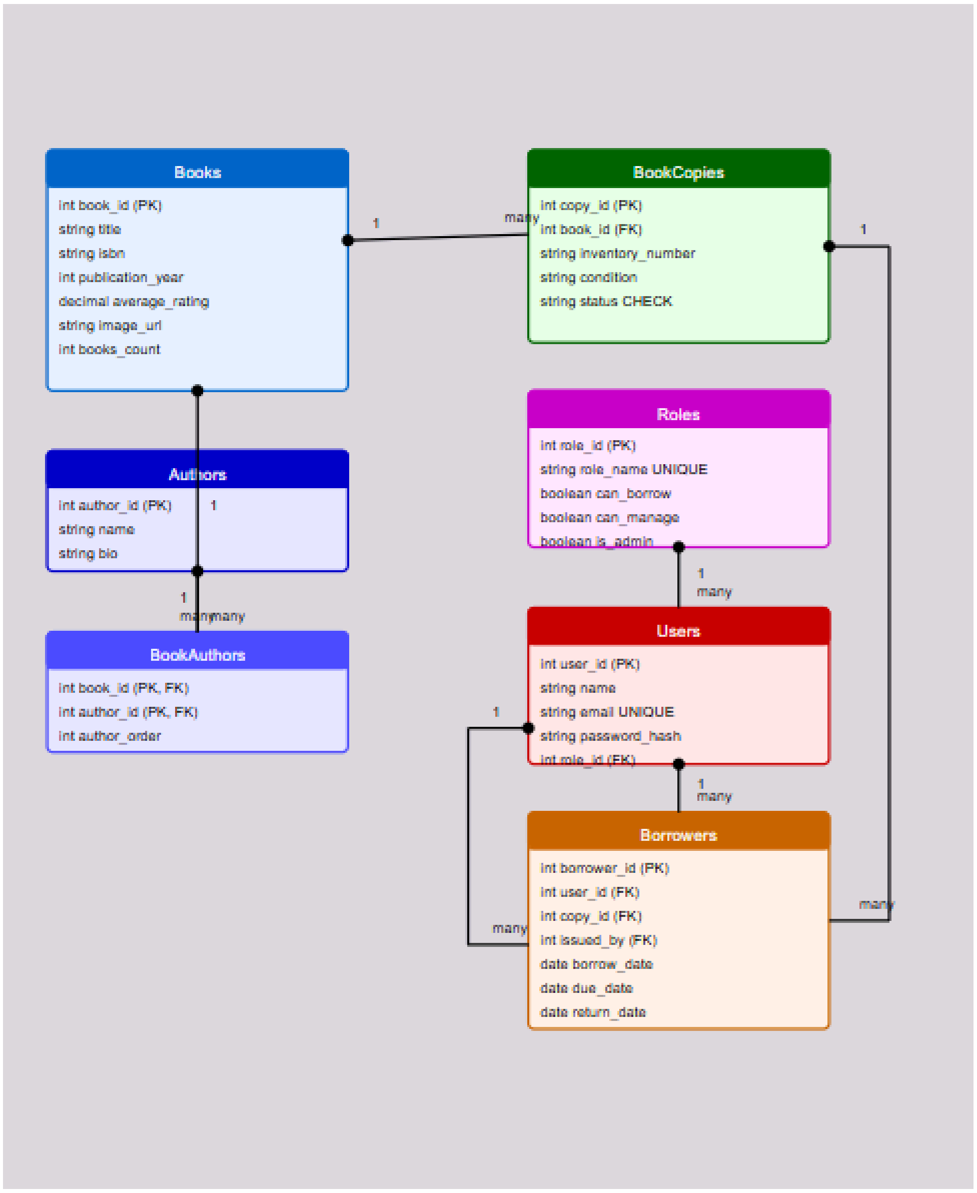


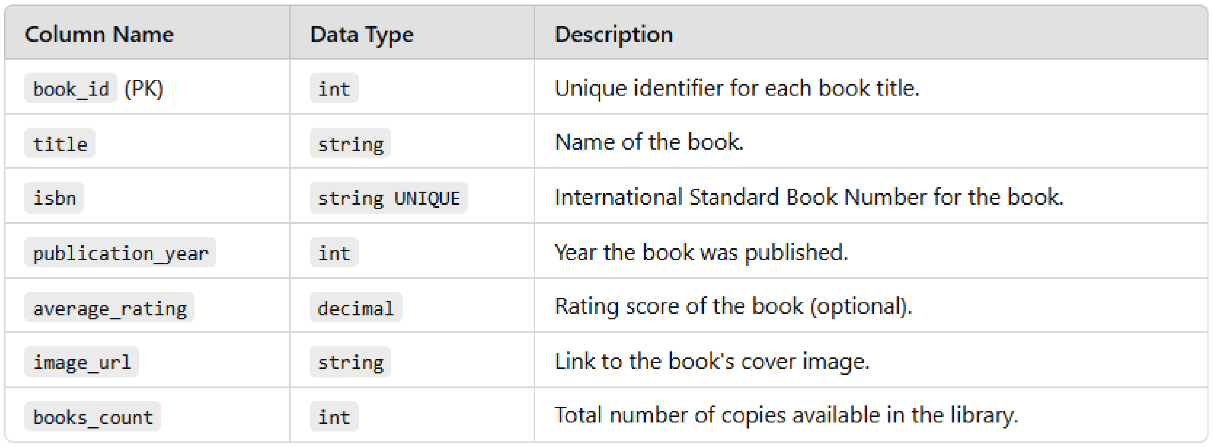
Detailed Explanation of the Enhanced Library System ERD with Role-Based Access Control

The provided **Enhanced Library System ERD (Entity-Relationship Diagram)** has been designed to effectively manage books, users, and borrowing transactions, incorporating a robust **role-based access control (RBAC)** mechanism. This ensures that different user types (normal users, librarians, and administrators) have appropriate permissions. Below is an in-depth analysis of each table, relationships, constraints, and potential improvements.

---

1. Books Table (Core Table)

The **Books** table represents unique book titles available in the library.



Column Name	Data Type	Description
`book_id` (PK)	`int`	Unique identifier for each book title.
`title`	`string`	Name of the book.
`isbn`	`string UNIQUE`	International Standard Book Number for the book.
`publication_year`	`int`	Year the book was published.
`average_rating`	`decimal`	Rating score of the book (optional).
`image_url`	`string`	Link to the book's cover image.
`books_count`	`int`	Total number of copies available in the library.

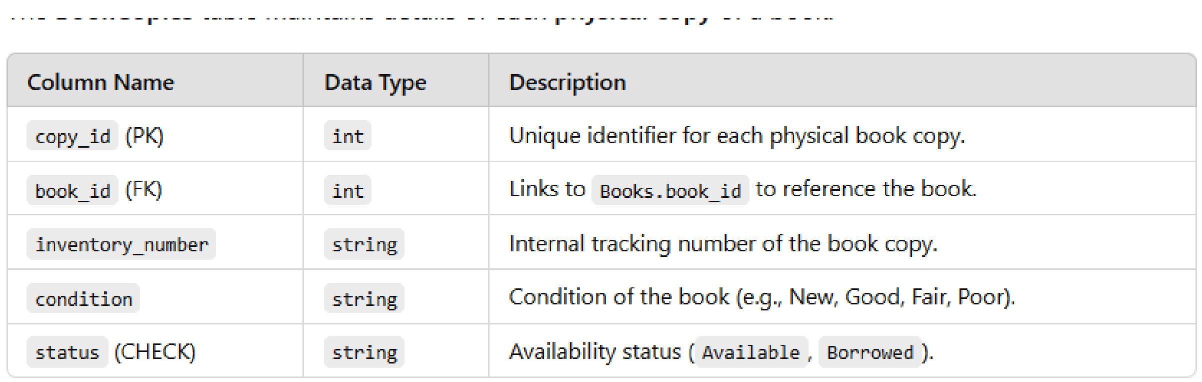
✅ **Key Considerations**:

- A single book entry can have **multiple copies**, tracked separately in the `BookCopies` table.
- The **ISBN should be UNIQUE** to prevent duplication errors.
- **Book rating and image URL** enhance user experience but are **not critical** for core operations.

---

2. BookCopies Table (Individual Copies of Books)

The **BookCopies** table maintains details of each **physical copy** of a book.



The BookCopies table maintains details of each physical copy of a book.

Column Name	Data Type	Description
`copy_id` (PK)	`int`	Unique identifier for each physical book copy.
`book_id` (FK)	`int`	Links to `Books.book_id` to reference the book.
`inventory_number`	`string`	Internal tracking number of the book copy.
`condition`	`string`	Condition of the book (e.g., New, Good, Fair, Poor).
`status` (CHECK)	`string`	Availability status (`Available`, `Borrowed`).

✅ **Key Considerations**:

- Ensures **accurate inventory tracking** per copy.

- The **condition** field helps in maintaining book quality.
- **Status constraint** ensures valid values (e.g., CHECK constraint to prevent invalid status entries).
- **Each copy_id is unique**, even if multiple copies of the same book exist.

---

3. Authors Table (Normalized Author Information)

This table **normalizes** author information, avoiding redundancy.

Column Name	Data Type	Description
`author_id` (PK)	`int`	Unique identifier for each author.
`name`	`string`	Author's full name.
`bio`	`string`	Short biography of the author (optional).

✅ **Key Considerations**:

- A **book can have multiple authors** (handled via `BookAuthors`).
- Author **details are stored separately** to eliminate **data duplication**.

---

4. BookAuthors Table (Junction Table for Many-to-Many Relationship)

Since a book can have **multiple authors**, and an author can write **multiple books**, we use a junction table.

Column Name	Data Type	Description
`book_id` (PK, FK)	`int`	References `Books.book_id`.
`author_id` (PK, FK)	`int`	References `Authors.author_id`.
`author_order`	`int`	Defines the order of authorship (e.g., primary author = 1).

✅ **Key Considerations**:

- Enables **many-to-many** author-book relationships.
- Maintains **author order**, important for books with multiple contributors.

5. Roles Table (Role-Based Access Control)

This table **manages user roles** with specific permissions.

Column Name	Data Type	Description
`role_id` (PK)	`int`	Unique identifier for each role.
`role_name` (UNIQUE)	`string`	Name of the role (`Member` , `Librarian` , `Admin`).
`can_borrow`	`boolean`	Determines if users in this role can borrow books.
`can_manage`	`boolean`	Determines if users in this role can **issue/return** books.
`is_admin`	`boolean`	Determines if users in this role have **full CRUD permissions**.

✅ **Key Considerations**:

- Ensures **flexibility** by defining **who can do what**.
- Easily **expandable** for **future roles** (e.g., "Guest" with read-only access).
- Uses **boolean flags** for efficient access control.

6. Users Table (Library Users)

This table **stores all users**, including members, librarians, and admins.

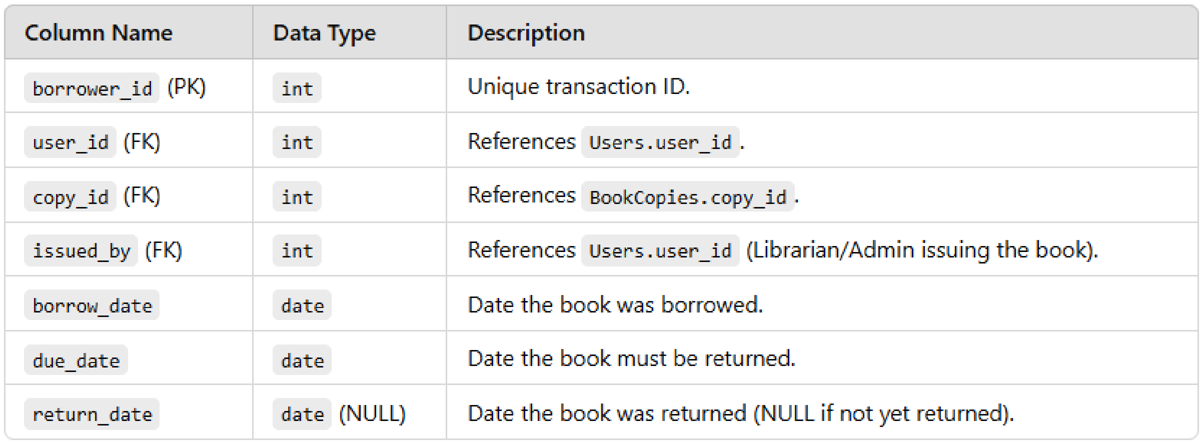
Column Name	Data Type	Description
`user_id` (PK)	`int`	Unique identifier for each user.
`name`	`string`	User's full name.
`email` (UNIQUE)	`string`	Unique email for login.
`password_hash`	`string`	Securely stored password hash.
`role_id` (FK)	`int`	References `Roles.role_id` to assign permissions.

✅ **Key Considerations**:

- **Passwords should be securely hashed** (never stored in plaintext).
- **Role-based access ensures security**.
- **Email must be unique** to prevent duplicate user accounts.

7. Borrowers Table (Borrowing Transactions)

Tracks book borrowing and returning activities.



Column Name	Data Type	Description
`borrower_id` (PK)	`int`	Unique transaction ID.
`user_id` (FK)	`int`	References `Users.user_id`.
`copy_id` (FK)	`int`	References `BookCopies.copy_id`.
`issued_by` (FK)	`int`	References `Users.user_id` (Librarian/Admin issuing the book).
`borrow_date`	`date`	Date the book was borrowed.
`due_date`	`date`	Date the book must be returned.
`return_date`	`date` (NULL)	Date the book was returned (NULL if not yet returned).

✅ **Key Considerations**:

- Tracks **who issued** the book (Librarian/Admin).
- The **return_date is NULL until the book is returned**.
- Allows **reporting of overdue books**.

Relationships in the Schema

1. **Books** → **BookCopies** (1:M): One book has many copies.
2. **Books** → **BookAuthors** (1:M) → **Authors** (M:1): A book has many authors, and an author has written many books.
3. **Users** → **Borrowers** (1:M): A user can have multiple borrow transactions.
4. **BookCopies** → **Borrowers** (1:M): A book copy can be borrowed multiple times.
5. **Users** → **Roles** (1:M): A user has one role, but a role can be assigned to many users.

Possible Improvements

✅ **Adding Fines/Penalties Table**:

- Track overdue fines.

- Add fields for `fine_amount`, `paid_status`, `payment_date`.

✅ **Adding Reservations Table**:

- Allows users to reserve books before borrowing.
- Tracks `reservation_date` and `expiration_date`.

✅ **Adding Notifications Table**:

- Send reminders for due/overdue books.

✅ **Expanding Book Details**:

- Add `genre`, `publisher`, `page_count`, `summary`.
