Designing Data Table For Loan Systems Library Books

Woka Aditama

In this series of articles, I'm going to share what I've learned and go through the process of creating a Relational Database using MySQL (and MySQL Server) from the very beginning, and hopefully clarify things enough on the way that you can get started building your own and feel confident that you won't accidentally delete all the data!

What is a Relational Database?

According to Oracle, a relational database is "a type of database that stores and provides access to data points that are related to one another". OK, sounds good!

We can create, read, update and delete (the basic functions of any database) the information in our relational database using a Relational Database Management System (RDBMS). Example of RDBMSs include Oracle, Microsoft SQl Server, MySQL, and PostgreSQL, among many others. Each of these have their pros and cons (and like everything coding-adjacent, their online hyper-partisans), and SQL is not implemented in exactly the same way in each of them. The concepts are the same, but the syntax and keywords may be slightly different, so it is not usually possible to use SQL code written for PostgreSQL in Microsoft SQL Server, for example, without making some modifications.

Designing a Database

As always, it will help us to walk through a real-life example here. Let's design a database based on our own requirements!

Defining the requirements

1. Mission Statement

Given the description, features, and limitations for an e-library application, your first mission is to create a mission statement for the project. This Mission Statement will guide you in determining the necessary objects for the database.

. The application oversees multiple libraries, each hosting a diverse collection of books with varying quantities available for borrowing. Users can borrow or place holds on books (when the book is not immediately available for borrowing).

Below are the key points and requirements for the e-library database system:

● Manages multiple libraries

The application manages multiple libraries, each housing a diverse collection of books with varying quantities available for borrowing.

● Book Collection

○ The database needs to store information about the diverse collection of books, including titles, authors, and available quantities.

○ To make searching easier for users, books are also divided into categories such as: self-improvement, biography, Fantasy, Romance, Science Fiction, etc.

● User Registration

Users can register on the e-library platform. Registered users can interact with the platform by borrowing books, placing holds, and managing their account.

● Loan and Hold System

○ Users can borrow books from any library in this application if the book is available.

○ The loan period is 2 weeks. Users can return books earlier than the due date

○ Books will be automatically returned when they exceed the due date

○ Users can only borrow 2 books at a time

○ The platform keeps track of loan transactions, including loan dates, due dates, and return dates.

○ Users can place holds on books that are currently unavailable.

○ The library maintains a hold queue, and when a book becomes available, it can be borrowed by the customer at the front of the queue. Additionally, if a customer doesn't borrow a held book within one week, the book is released for other users to borrow.

○ Users can only hold 2 books at the same time

2. Creating Table Structures

Determine the objects needed for this database. These objects can eventually become tables. After identifying the tables and their descriptions, specify the fields and keys for each table. Start by determining candidate keys, then choose the primary key among these candidates.

3. Determine Table Relationships

Next, determine the relationships between the tables that have been created. Explan with the type of relationship (one-to-one, one-to-many, or many-to-many) and the foreign key.

4. Determine Business Rules

Apply Business rules via constraints for the required fields. For example:

Tabel Book:

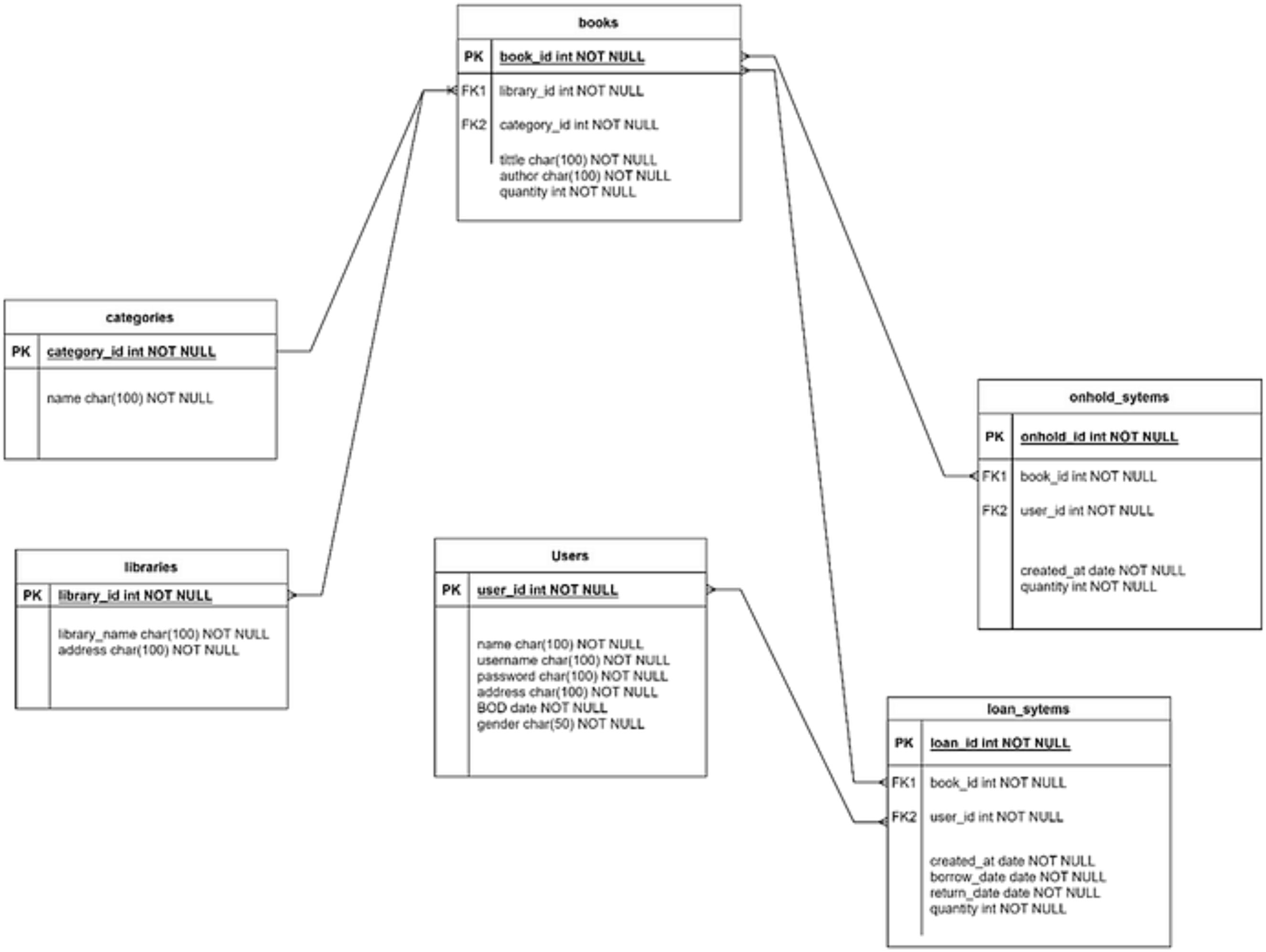
- tittle: NOT NULL

- author: NOT NULL

- quantity: NOT NULL, CHECK(quantity >= 0)

5. Implementing The Design

The result of this Database Design is an Entity Relationship Diagram (ERD). After creating the ERD, implement the ERD results into the database using PostgreSQL and Data Definition Language (DDL).



Creating Dummy Dataset

In this report, we outline the process of generating dummy datasets for a loan book system using Python. The purpose of creating these datasets is to simulate data for analysis and testing purposes.

Steps Taken:

1. Data Generation Functions:

We utilized the Faker library in Python to generate fake data for various tables in the loan book system.

Separate functions were created to generate data for the Books, Users, Categories, Libraries, and Loan_Systems tables.

2. Generating Dummy Data:

For each table, we determined the required fields and used Faker to generate realistic data.

Randomization techniques such as random IDs, names, addresses, dates, and quantities were used to create diverse datasets.

3. Saving Data to CSV:

After generating data for each table, we saved the data to CSV files using Python’s built-in CSV module.

Each CSV file corresponds to a specific table in the loan book system.

Overview of Generated Datasets:

Books.csv: Contains information about books such as title, author, library ID, category ID, and quantity.

Users.csv: Includes user details such as name, username, password, address, date of birth, and gender.

Categories.csv: Provides a list of categories with unique IDs and names.

Libraries.csv: Contains information about libraries, including library ID, name, and address.

Loan_Systems.csv: Simulates loan records with details like user ID, book ID, creation date, borrow date, return date, on-hold status, and quantity.

you can check the code here :

Learning How To Analyze

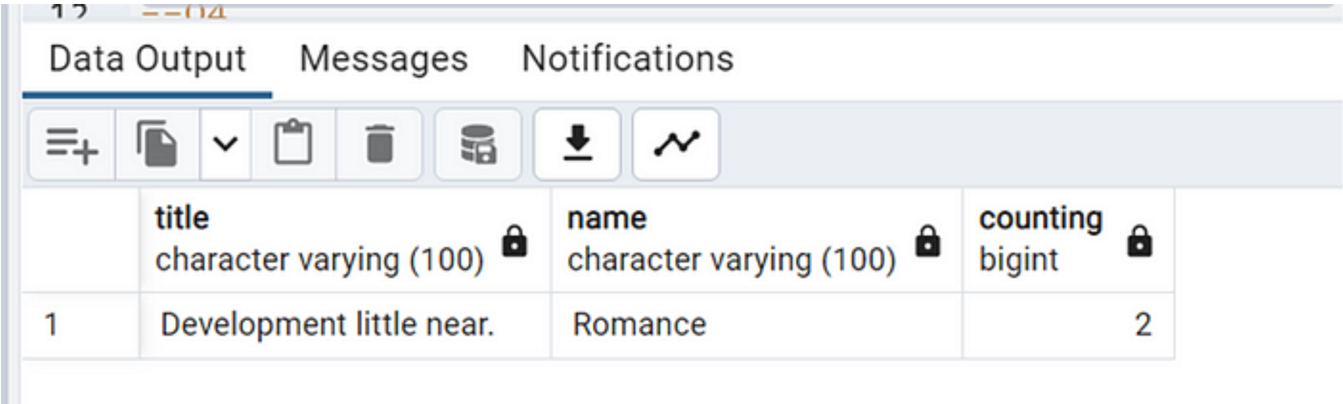
Five question for analysis:

1. What is most category books being borrowed?

This question helps decision makers to know what kind of book category can be added to library collection based on borrower's interest.

SELECT new_join.title, c.name, COUNT(*) AS counting FROM (SELECT * FROM loan_systems as ls LEFT JOIN books as b ON ls.book_id=b.book_id)

as new_join LEFT JOIN categories as c ON c.category_id=new_join.category_id GROUP BY new_join.title, c.name ORDER BY COUNT(*) DESC LIMIT 1;



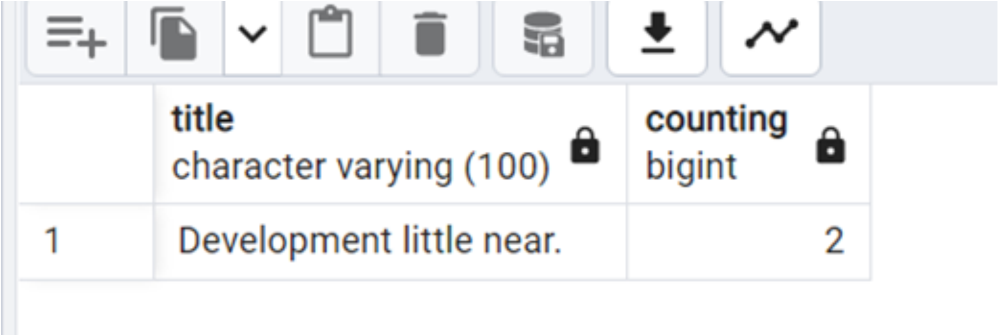
	title character varying (100)	name character varying (100)	counting bigint
1	Development little near.	Romance	2

Romance could be a recommended category to be added more in the future for library collections.

2. What is most books being borrowed?

This question helps decision makers to know what kind of book title can be added to the library collection based on the borrower's interest.

SELECT b.title, COUNT(*) AS counting FROM loan_systems as ls LEFT JOIN books as b ON ls.book_id=b.book_id GROUP BY b.title ORDER BY COUNT(*) DESC LIMIT 1;



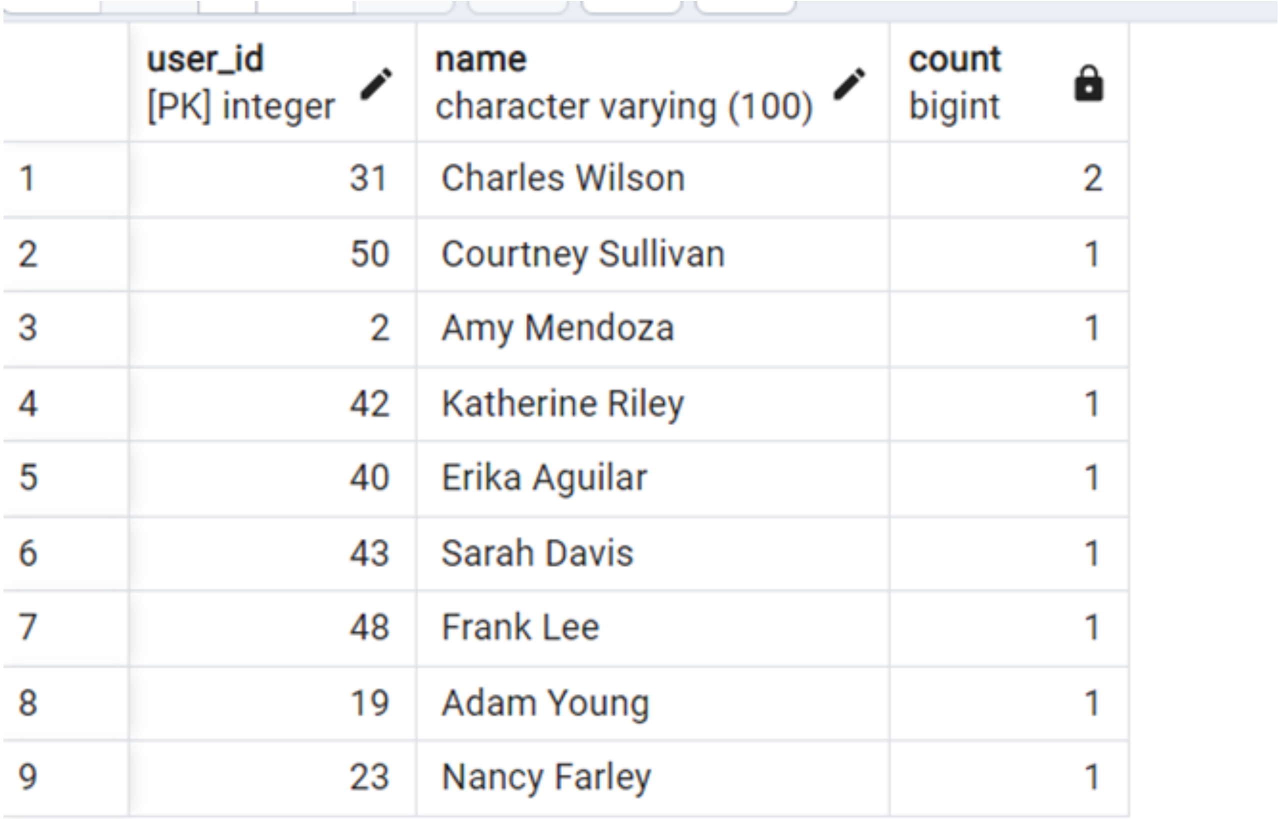
	title character varying (100)	counting bigint
1	Development little near.	2

Development little near might be favorurite so far borrowed 2 unit.

3. What is user's most borrowing book?

This question helps decision-makers identify the profile of customers who have a big interest in library books.

SELECT u.user_id, u.name, COUNT(*) FROM loan_systems as ls LEFT JOIN users as u ON ls.user_id=u.user_id GROUP BY u.user_id, u.name ORDER BY COUNT(*) DESC;



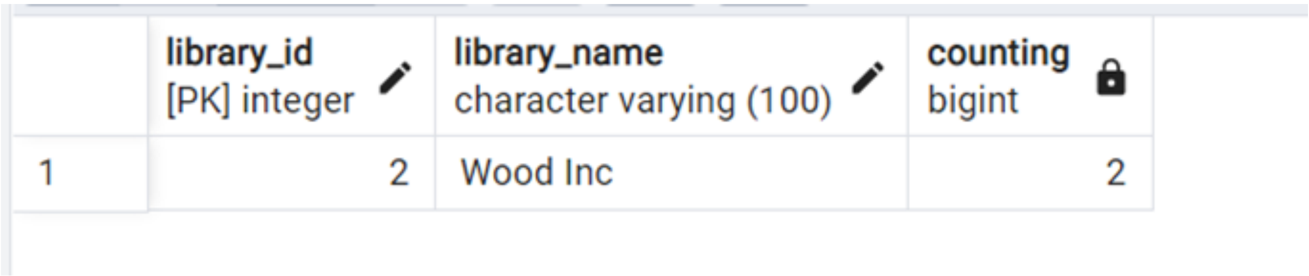
	user_id [PK] integer	name character varying (100)	count bigint
1	31	Charles Wilson	2
2	50	Courtney Sullivan	1
3	2	Amy Mendoza	1
4	42	Katherine Riley	1
5	40	Erika Aguilar	1
6	43	Sarah Davis	1
7	48	Frank Lee	1
8	19	Adam Young	1
9	23	Nancy Farley	1

Charles Wilson can be explored more from their background and profile as one who is indicated to have a big interest in library collections.

4. What library having most books category collection?

In library management, knowing this insight helps to set benchmarks and standards of capacity for all libraries.

SELECT lb.library_id, lb.library_name, COUNT(DISTINCT new_join2.category_id) AS counting FROM (SELECT new_join.* FROM (SELECT * FROM loan_systems as ls LEFT JOIN books as b ON ls.book_id=b.book_id) as new_join LEFT JOIN categories as c ON c.category_id=new_join.category_id) as new_join2 LEFT JOIN libraries as lb ON new_join2.library_id=lb.library_id GROUP BY lb.library_id, lb.library_name ORDER BY COUNT(*) DESC LIMIT 1;



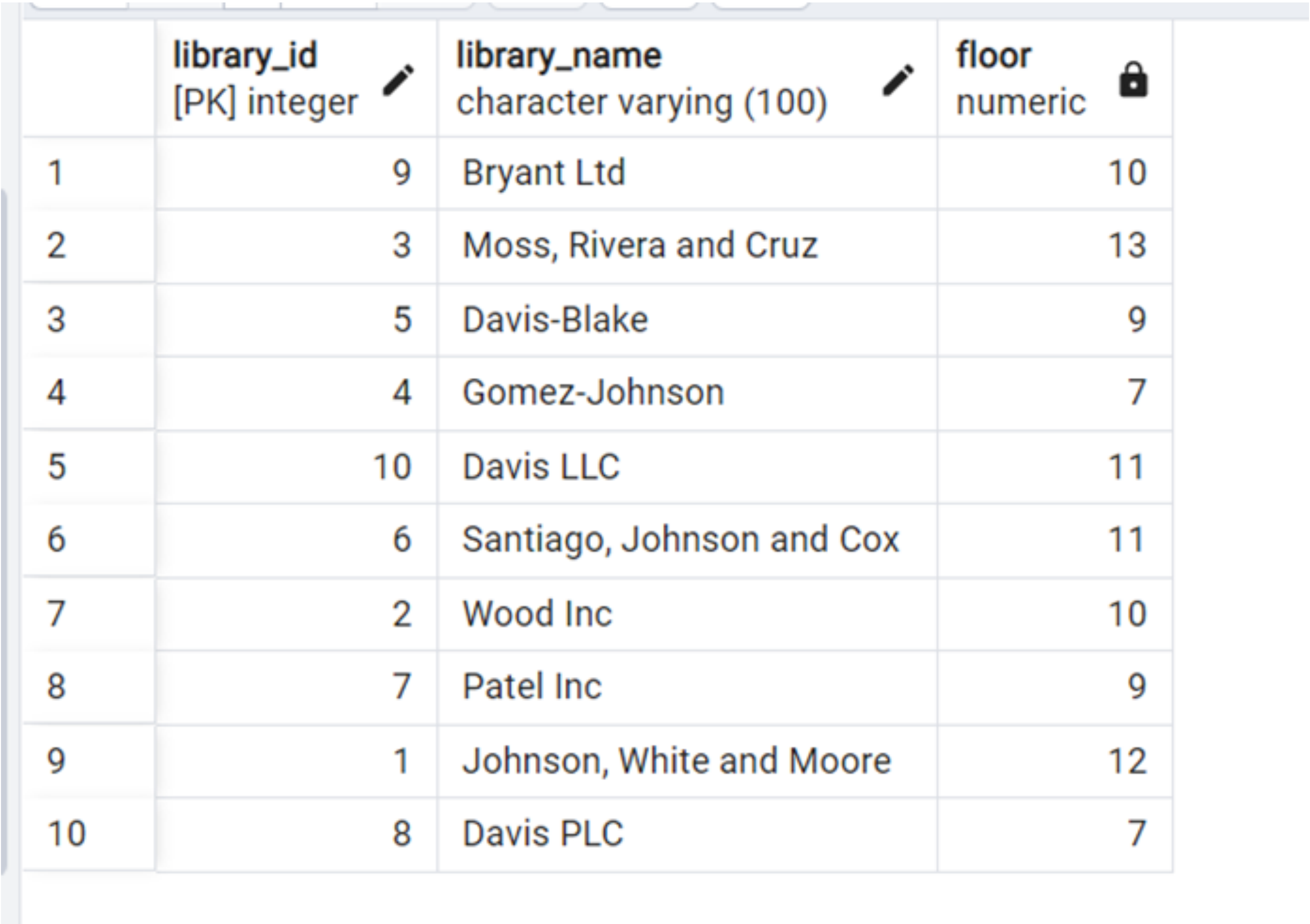
	library_id [PK] integer	library_name character varying (100)	counting bigint
1	2	Wood Inc	2

Decision makers can use Wood Inc. library specifications as an example if they have a plan to add a new library.

5. How much is the Average quantity of available books?

In library management, knowing this insight helps to set benchmarks and standards of capacity for all libraries.

SELECT lb.library_id, lb.library_name, FLOOR(AVG(b.quantity)) FROM libraries as lb LEFT JOIN books as b ON lb.library_id= b.library_id GROUP BY lb.library_id, lb.library_name;



	library_id [PK] integer	library_name character varying (100)	floor numeric
1	9	Bryant Ltd	10
2	3	Moss, Rivera and Cruz	13
3	5	Davis-Blake	9
4	4	Gomez-Johnson	7
5	10	Davis LLC	11
6	6	Santiago, Johnson and Cox	11
7	2	Wood Inc	10
8	7	Patel Inc	9
9	1	Johnson, White and Moore	12
10	8	Davis PLC	7

Decision-makers can use this data to set a minimum standard for library specifications in order to accommodate more collections of books in the future.

REFERENCES

https://github.com/woka20/damn_a_lot_code_syntax/tree/main