e-Library App

SQL and Relational Database Simulation

Project's Objectives

Objective:

Design a database for an e-library app that manages multiple libraries, each with its own collection of books in varying quantities. Users can borrow available books or place holds on books that are currently checked out. The system should track book inventory by library, manage user borrowing and holds, and maintain accurate records of book availability and user transactions.

Features & Limitation:

- The application manages multiple libraries, each with a unique collection of books available in varying quantities for borrowing.
- The database should store information on each library's diverse book collection, including titles, authors, available quantities and categories (basically handled by application).
- Users can register on the e-library platform, allowing them to borrow books, place holds, and manage their accounts.
- Loan and Hold System (basically handled by application)
    - 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

Designing the Database

Mission Statement

- Provide an e-library application database that hosts multiple libraries with diverse collection of books with varying quantities available for borrowing
- Users can borrow available books from any library within the app
- Users can place hold of non-available books (books that are currently being borrowed) as a reservation system

Table Structure

- The libraries table is required to store multiple libraries to be managed
- The books table is required to store information of the book independently
- The book_collection table will store each library's diverse book collection
- The users table stores user that registered on the e-library platform
- Both loan_status and hold_status tables store status for loan and hold transaction
- The loan_transaction stores transactional data in regards to the borrowing transaction

- The hold_reserve stores transactional data in regards to the hold/reservation transaction

Table Name	Description
libraries	Store information about libraries that managed by the app
users	Store about the users registered to the app
books	Store information about details of the books including titles, authors, and categories
book_collection	Store information about book collections available in respected libraries including quantities
loan_status	Store information about the loan status
hold_status	Store information about the hold status
loan_transaction	Store information about the book borrowing transaction including loan dates, due dates, and return dates
hold_reserve	Store information about the book hold reservation including hold dates

Field Name & Data Type

- **Libraries** store information of libraries that is managed by the app. Every field requires a not null value and primary key are auto-incremental

libraries		
library_id	**serial**	**PK**
library_name	varchar(255)	not null
phone	varchar(20)	not null
address	varchar(1000)	not null

- **Users** store information users registered to the app. Every field but last_name requires a not null value and primary key are auto-incremental

users		
user_id	**serial**	**PK**
first_name	varchar(255)	not null
last_name	varchar(255)	
email	varchar(255)	not null, unique
phone	varchar(255)	not null, unique
address	varchar(255)	not null
user_password	varchar(255)	not null

- **Books** store detail information about the book. Only crucial information such as title and author are requires not null value, as in real life situation other fields are often could not be retrieved. Primary key are auto-incremental

books		
book_id	**serial**	**PK**
title	varchar(1000)	not null
author	varchar(1000)	not null
isbn	bigint	

publication_date	date	
publisher	varchar(1000)	
genre	varchar(1000)	

- **book_collection** store information about each library's book collection. The collecion_id hold information about book, its whereabouts, its quantity, and whether is available for borrowing or not. All the field requires not null value. Quantity must be at least 0. Primary key are auto-incremental

book_collection		
collection_id	**serial**	**PK**
library_id	int	not null
book_id	int	not null
quantity	int	not null, check >= 0
is_available	int	not null

- **loan_status** store constraint of loan status for loan_transaction table.

loan_status		
status_id	**int**	**PK**
status	varchar(100)	not null

- **hold_status** store constraint of loan status for hold_reserve table

hold_status		
status_id	**int**	**PK**
status	varchar(100)	not null

- **loan_transaction** store information of the transactional data of loan or borrowing transaction. Other than return_date for it may be are still being borrowed, the fields are required to be not null. Primary key are auto-incremental

loan_transaction		
loan_id	**serial**	**PK**
collection_id	int	not null
user_id	int	not null
loan_date	date	not null
return_date	date	
status_id	int	not null

- **hold_reserve** store information of the reservation or hold placement when the book is currently not available to borrow. Other than hold_date as the it still may be reserved, the fields are required to be not null. Primary key are auto-incremental

hold_reserve		
hold_id	**serial**	**PK**
collection_id	int	not null
user_id	int	not null
hold_start	date	not null
hold_end	date	
status_id	int	not null

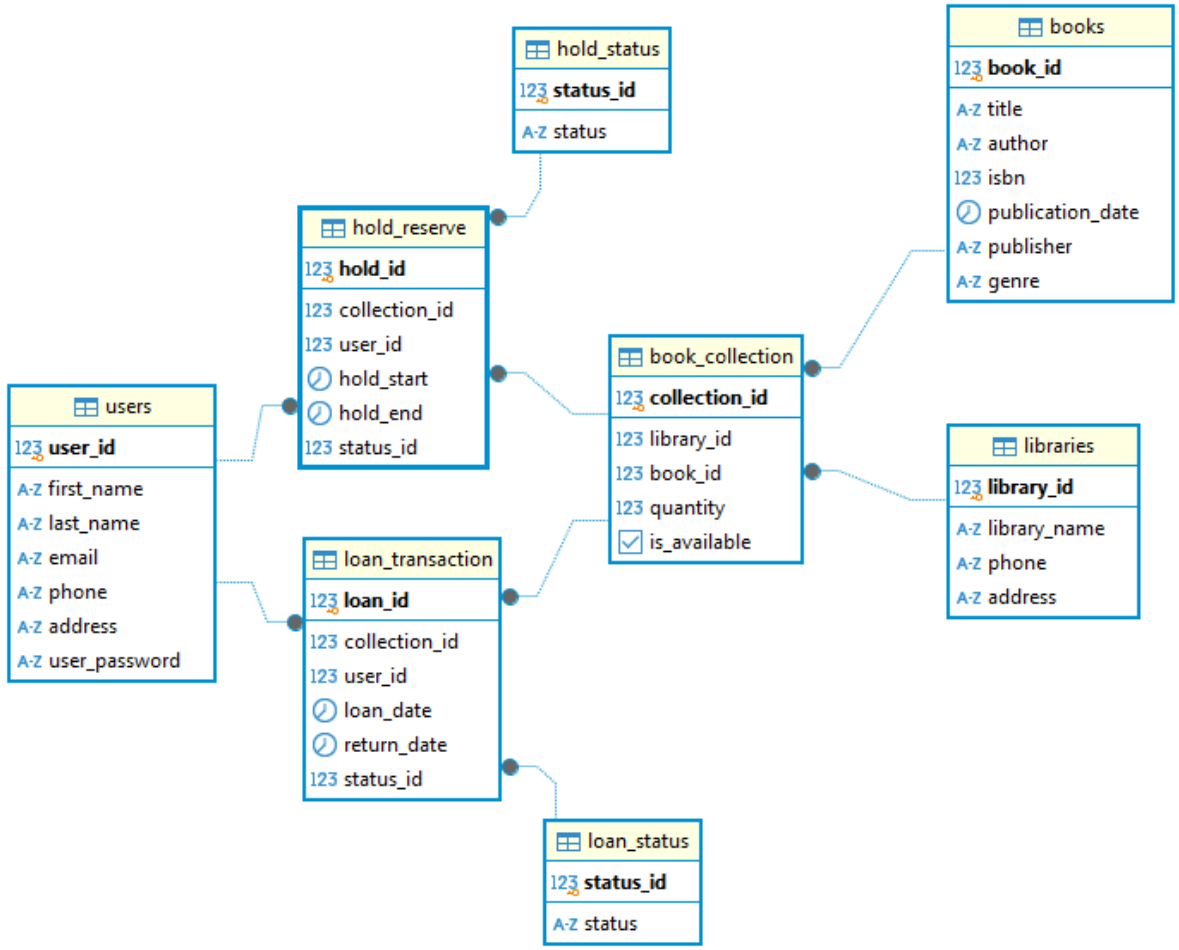
Table Relationship

	libraries	users	books	book_coll	loan_stat	hold_stat	loan_tran	hold_res
libraries				1:N				
users							1:N	1:N
books				1:N				
book_coll	N:1		N:1				1:N	1:N
loan_stat							1:N	1:N
hold_stat								
loan_tran		N:1		N:1		N:1		
hold_res		N:1		N:1		N:1		

Description of the table:

- **libraries – book collection**: 1 library id can be used in multiple collection id, but not vice versa.
- **users – loan_transaction**: 1 user id can do many loan transaction, but not vice versa.
- **users – hold_reverse**: 1 user id can do many hold reserve, but not vice versa.
- **book – book_collection**: 1 book id can be used in multiple collection id, but not vice versa.
- **loan_status – loan_transaction**: 1 loan status id can be used in multiple loan transaction, but not vice versa.
- **hold_status – hold_reserve**: 1 hold status id can be used in multuple hold placement, but not vice versa.

Entity Relational Diagram



Implementing the Design

POSTGRESQL and DDL

```
CREATE TABLE libraries(
    library_id SERIAL PRIMARY KEY,
    library_name VARCHAR(255) NOT NULL,
    phone VARCHAR(20) NOT NULL UNIQUE,
    address VARCHAR(1000) NOT NULL
);

CREATE TABLE users(
    user_id SERIAL PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255),
    email VARCHAR(255) NOT NULL UNIQUE,
    phone VARCHAR(20) NOT NULL UNIQUE,
    address VARCHAR(1000) NOT NULL,
    user_password VARCHAR (20) NOT NULL
);

CREATE TABLE books(
    book_id SERIAL PRIMARY KEY,
    title VARCHAR(1000) NOT NULL,
    author VARCHAR(1000) NOT NULL,
```

```
    isbn BIGINT,
    publication_date DATE,
    publisher VARCHAR(1000),
    genre VARCHAR(1000)
);

CREATE TABLE book_collection(
    collection_id SERIAL PRIMARY KEY,
    library_id INT NOT NULL,
    book_id INT NOT NULL,
    quantity INT NOT NULL CHECK (quantity >= 0),
    is_available BOOLEAN NOT NULL,
    CONSTRAINT fk_book_coll FOREIGN KEY (book_id) REFERENCES books
(book_id),
    CONSTRAINT fk_lib_coll FOREIGN KEY (library_id) REFERENCES libraries
(library_id)
);

CREATE TABLE loan_status(
    status_id INT PRIMARY KEY,
    status VARCHAR (100) NOT NULL UNIQUE
);


CREATE TABLE hold_status(
    status_id INT PRIMARY KEY,
    status VARCHAR(100) NOT NULL UNIQUE
);


CREATE TABLE loan_transaction(
    loan_id SERIAL PRIMARY KEY,
    collection_id INT NOT NULL,
    user_id INT NOT NULL,
    loan_date DATE NOT NULL,
    return_date DATE,
    status_id INT NOT NULL,
    CONSTRAINT fk_coll_loan FOREIGN KEY (collection_id) REFERENCES
book_collection (collection_id),
    CONSTRAINT fk_user_loan FOREIGN KEY (user_id) REFERENCES users
(user_id),
    CONSTRAINT fk_loan_status FOREIGN KEY (status_id) REFERENCES
loan_status (status_id)
);

CREATE TABLE hold_reserve(
    hold_id SERIAL PRIMARY KEY,
    collection_id INT NOT NULL,
    user_id INT NOT NULL,
    hold_start DATE NOT NULL,
    hold_end DATE,
    status_id INT NOT NULL,
    CONSTRAINT fk_coll_hold FOREIGN KEY (collection_id) REFERENCES
book_collection (collection_id),
    CONSTRAINT fk_user_hold FOREIGN KEY (user_id) REFERENCES users
(user_id),
    CONSTRAINT fk_hold_status FOREIGN KEY (status_id) REFERENCES
hold_status (status_id)
);
```

Generating Dummy Data

Generating Dummy Data

Please note that Python code will be attached as link: https://shorturl.at/dqq3Q

Libraries table
In the predefining process, the libraries table requires district names which derive from area table (see references). After consideration, the area table, however, is later not included in the report as it is not necessarily needed for the library is being electronic. The libraries dummy data is generated using Python.

Users table
Every field in users table are generated using Python.

Books table
Books dataset was gathered from kaggle (see reference). However, as it did not have the categories, the categories has to be added later. The categories were defined before assigning it to the data by using Python.
In a real case scenario, it would be better if the author, publisher, and categories had their own master table and id.

Book_collection table
Book collection table stores book information on its respective libraries. The data then generated by generating books for each library and then merging it into one collection table to get the collection id.

Loan_status table
Loan status value was created directly through PostgreSQL with below query:

```
INSERT INTO loan_status(
    VALUES
        (1,'Active'),
        (2,'Returned')
);
```

Hold_status table
Hold status value was created directly through PostgreSQL with below query:

```
INSERT INTO hold_status(
    VALUES
        (1,'Active'),
        (2,'Loaned'),
        (3,'Cancelled'),
        (4,'Expired')
);
```

Loan_transaction table
Loan_transaction table generated by randomly selecting collection id and generating loan date and return date. If collection id has been added to the list, then the date is to be taken after the return date. The rules binding here are:

- 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.

Although for this cases where users can only borrow 2 books at a time are may not represented in the data yet and it have to be done through the app.

Hold_reserve table
Hold_reserve table generated by randomly selection collection id that is available in loan transaction table. The rules are binding here where:

- 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

Although for this cases where users can only hold 2 books at a time are may not represented in the data yet and it have to be done through the app.

Input Data to Database

The data is input by importing using Dbeaver.

Reference:

- https://github.com/alifbint/indonesia-38-provinsi
- https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks