CS5200
PROJECT REPORT

Team: YangLWanXZengX

Team Members: Lianrui Yang, Xinyu Wan, Xiangyu Zeng

Contents

Top Level Description

This database will provide services for all users of a library.

In a library, a book has its book_id, ISBN, book name, author, availability and publisher id which belong to its publisher. When a user borrows a book, availability of the borrowed book will become “Not Available”. When this book is returned, “Available” will be assigned to its availability. Each publisher has a unique publisher_id. It also has a name, a contact phone number. A book can only be provided by one publisher and a publisher can provide many books and many kinds of books to the library.

Users of the library can borrow books. Each user has a unique user_id and a name. Users can borrow at most 5 books. Once a user is borrowing a book, he/she will generate a record which contains user_id, book_id, starting_time, estimated_End_time. If the user returns any book late, a penalty (True/False) will be generated based on its real_end_time.

A user can reserve study rooms from the library from 9:00 to 21:00. A user can hourly reserves any time duration. For example, user “20220101” can reserve from 12:00 to 21:00. Each room contains tables and seats. Some of them also have screens (or TV). Tables and seats have their own unique id. A screen has a unique screen_id, its manufacturer and a serial number.

Because some students want to print their document in the library. The library also provides some printing services. Students can provide their electronic documents to these printers so that they can be printed. Each electronic document has their own document id, a document name. When a document is printed, the system will eliminate its record.

Functionalities Provided by This Database:

(We called the people who are using this system as Administrators)

1. Administrators can create/delete/update/read books and borrow record of books;
    1) Administrators can help users borrow and return books by updating the borrow records;
    2) Administrators can track books by their information (attributes). (i.e. tracking book by book name, ISBN, book id)
    3) Administrators can add/delete books due to the changing in the library;
2. Administrators can create/delete users;
    1) Administrators can register users (add them to database) or take them away from database (delete them);
    2) The system will track if the user is in the database in several procedures such as borrow book, add(print) document and all functionalities that need the authority as a user;
3. Administrators can create room record and track the devices associated with study rooms;
    1) Administrators can help users reserve rooms in a specific time slot;

    2) Administrators can check if one study room exists and help users track study rooms by seats, screens and tables. On the contrary, could track seats, screens and tables by room as well by different procedures;
4. Administrators can create/delete records to documents;
    1) Administrators could upload (add record to database) user's documents and print (delete record from database) them.
5. Administrators can create new publisher

Technology

The front-end development utilized the pymysql library to connect to the database server. Basic functions are also developed for management purposes. The core functions used the interfaces provided by the procedures stored in the library database, and some extra work such as the formalization of the output result and the validation of the input data are done by python. Several algorithms such as the slide puzzle have also been utilized to facilitate the correctness of the logic.

We used SQL based database language. Although according to [1], the performance of NoSQL is significantly better than SQL, SQL still has some advantages. Because SQL existed for a long time, community support of SQL is more robust than NoSQL support. As a result, the development time of this database would be shorter.
We completed and improved the functionality we assumed in the proposal. Internal logic is self-consistent.

Specifically, we use procedures to create, read, update, delete.
We use procedures to implement all the functionalities due to which procedures can have input and output parameters whereas functions can have only input parameters. And the procedure can return 0 or n values.

Software, Language and Library

SQL, MySQL Workbench (Database Construction Part)
Python, Pycharm (Front End Construction Part)
MacOS, Terminal
This project is fully conducted on MacOS. Python (pymysql) provides most front end functions.

UML Diagram

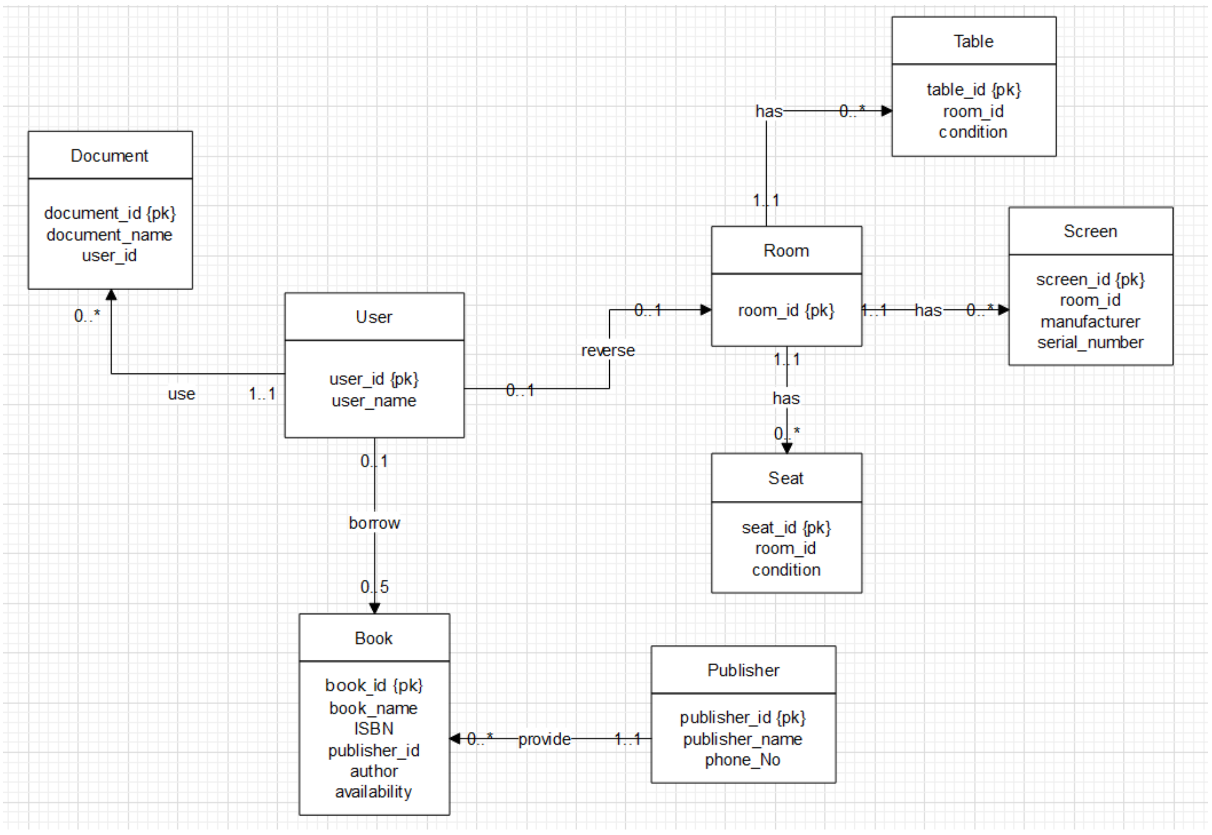

Figure 1: UML Diagram

We have changed our UML due to our improvement to the requirements and design for the system. Figure 1 shows our latest concept of our system. It varies from our original proposal with much more clear relationships, logical development and a wide variety of functionalities.

Reverse Engineer

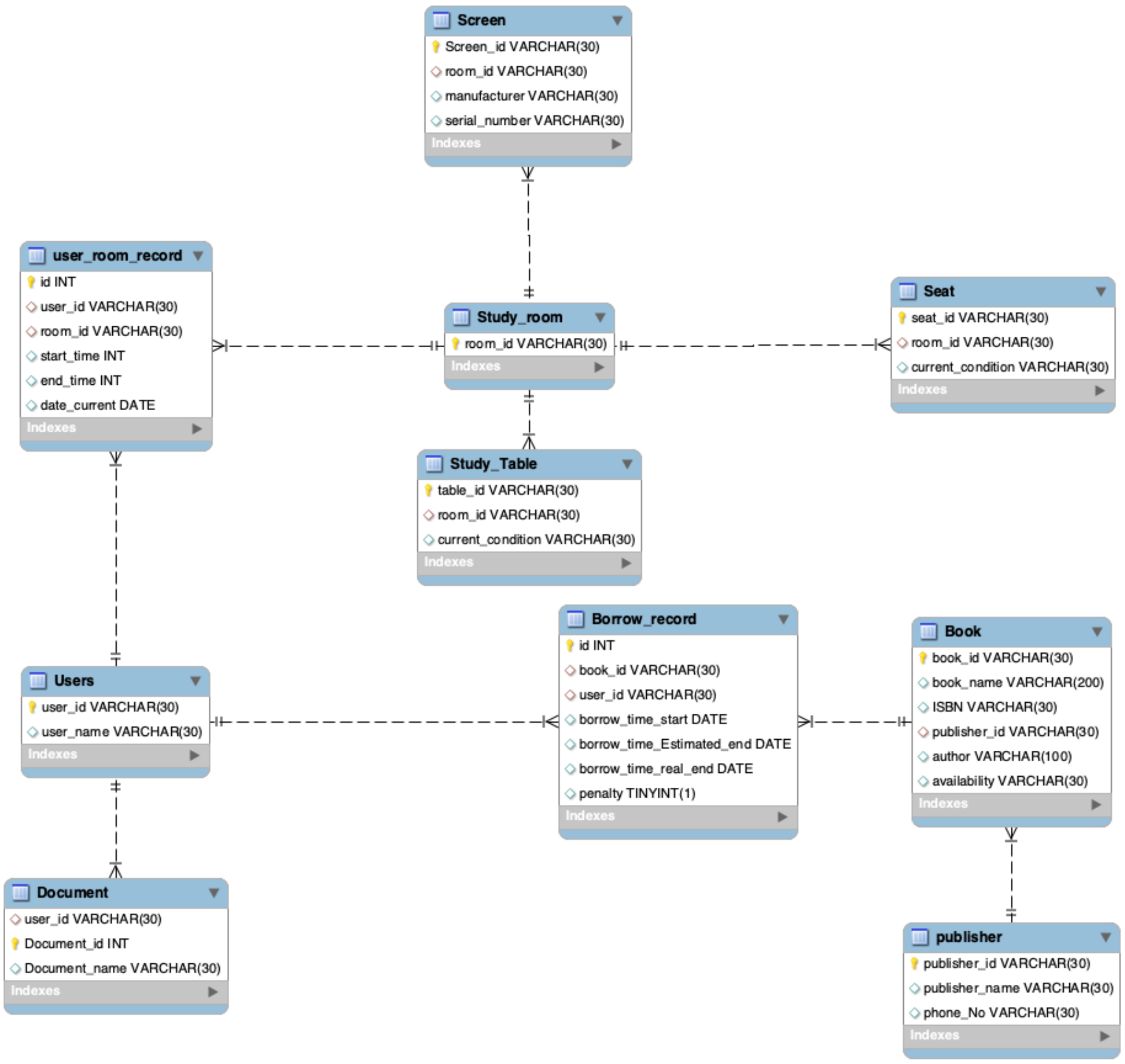

Figure 2: Reverse Engineering Logical Design Diagram of Library Database

As shown in Figure 2, the Library database contains 10 entities. Within these 10 entities, 8 of them are strong entities and 2 of them (user_room_record, Borrow_record) are weak entities.

User Flow

Figure 3 illustrates user flow of our database. You could run **python db.py** and connect to the database server, then choose functions according to the hints that are provided by **db.py**.

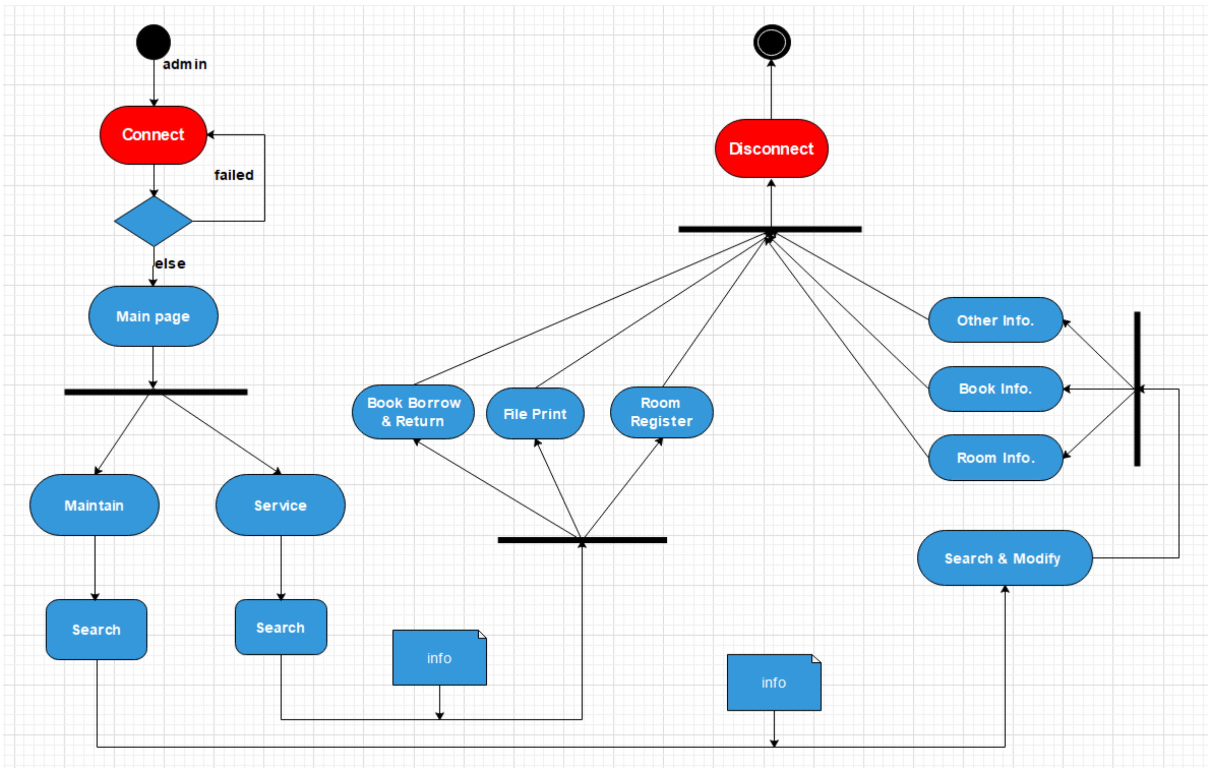

Figure 3: User flow diagram

Readme

Readme Refers to the README.md file in the zip.

Lessons Learned

1. For some tables we didn't know which ones should be the primary key or partial primary key at the beginning. Then we did a little research about it. We solved it by involving auto increment. So, everytime administrator creates a record, the system will generate an unique auto increment number as the primary key. It's unique because it won't be influenced by the behavior of delete[2][3].

   Because we have 3 teammates in this group, how to cooperate with each other becomes a significant problem. Since we divided the Library database into 3 parts which are: Constructing tables, Constructing procedures/functions/triggers, Constructing front end (in Python), we need a protocol to guarantee that codes programmed respectively can work together.

2. Time management is another challenge for us. Logically, we should firstly finish creating tables. Then, procedures need to be done. Finally, the user interface (front end) will be programmed based on functions provided by the database. Unfortunately, if we follow this procedure, this project will consume a large amount of time since the three of us cannot work parallely. We created a simple but effective method to guarantee parallelism. Firstly, we designed an UML and all of us should work based on this diagram. Next, we negotiated functions that the database should provide and designed an interface between MySQL and Python. This interface boosted our work significantly.

3. We designed 2 plans to store records of users reserving study rooms. The first plan which is not applied in this project is shown in Table 1 where room_id_value is the reserved room id, date_value is the date that the room is reserved. user_id_value under 9:00-10:00 means this user reserves the study room between 9:00 to 10:00. This plan will work because we only allow users to reserve a study room hour by hour.

Room	Date	9:00-10:00	…	20:00-21:00
room_id_value	date_value	user_id_value	…	user_id_value

Table 1: Plan A of user_room_record table

Obviously, this plan has some tremendous drawbacks. When a user tends to reserve multiple hours, the only way to reserve more than one hour is updating the tuple multiple times. This design does not allow parallelism and does not quite make sense. To solve these problems, we negotiated another plan as shown in Table 2 which is applied in this project.

Room	Date	User ID	Start Time	End Time
room_id_value	date_value	user_id_value	start_timeValue	end_timeValue

Table 2: Plan B of user_room_record table

In Table 2, only start time and end time are recorded in one tuple. Because each user, room, time combination will have a unique tuple, Plan B meets the third normal form. Furthermore, this design allows users to reserve multiple hours by 1 operation. The only drawback is that an algorithm is required to guarantee that a time slot of a specific room cannot be reserved by multiple users.

4. When we discussed and designed how to achieve our functionalities, we listed all the functionalities we needed and finally we decided to use Procedures to implement them all. This is because Functions can have only input parameters whereas Procedures can have input and output parameters. And the procedure can return zero or n values. So Procedures are more suitable for our system[4].

5. In consideration of the case sensitive of book names, we involved fuzzy search into that procedure[5]. It could significantly reduce the errors the administrator will make in the actual implementation.

6. Also, we received a lot of useful advice about how to design and manage our system from professor Durant. We really appreciate her for her patience and time.

Future Works

This system can be improved in the following aspects:

1. More advanced Graphical User Interface can be implemented on the front end, we could use React.js and divide different functions into different pages which would be easier to be implemented by the user.
2. We could add the login/logout flow for the users in the future. So users could borrow/return/track books and reserve rooms etc. all by themselves. This will cut down the workload for administrators.
3. We could add a function for users to reserve rooms online and visualize the availability of the time slot. Users can reserve the study room in the next one week. And the max reservation of each user within a week is twice.
4. We can subdivide users to teachers, students and guests. Each type of user should have different/hierarchy authorities.
5. We can adjust the stock every year by analyzing the record according to the frequency each book was borrowed. If certain books were very popular and were borrowed frequently, we can raise their stock, vice versa.

References

[1] Benymol Jose, Sajimon Abraham, *Performance analysis of NoSQL and relational databases with MongoDB and MySQL*, Materials Today: Proceedings, Volume 24, Part 3, 2020, Pages 2036-2043, ISSN 2214-7853,
[2] https://www.w3schools.com/sql/sql_autoincrement.asp
[3] https://www.geeksforgeeks.org/sql-auto-increment/
[4]https://tutorialslink.com/Articles/Difference-between-Stored-Procedures-and-Functions-in-SQL-Functions-vs-Stored-Procedures/1625
[5] https://blog.pythian.com/fuzzy-search-sql-server-part-1/