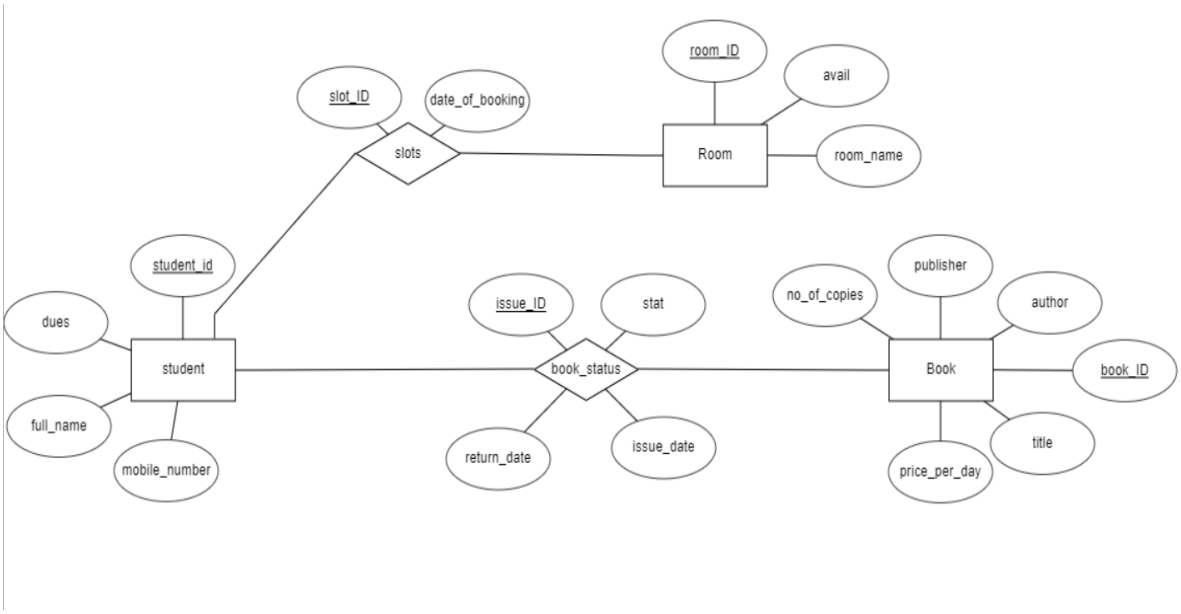
Documentation

This pdf serves as the documentation of our project of Database Systems – Library Management System. We have worked together to build a typical library database with records of various books and added functionalities to book rooms based on different slots (timings) available in a day. Also, the library calculates pending dues for students who have not returned their issued books within one week of time and alert them accordingly.

System Requirement Specifications

1. MySQL
2. Node.JS

ER Diagram



Schema Design

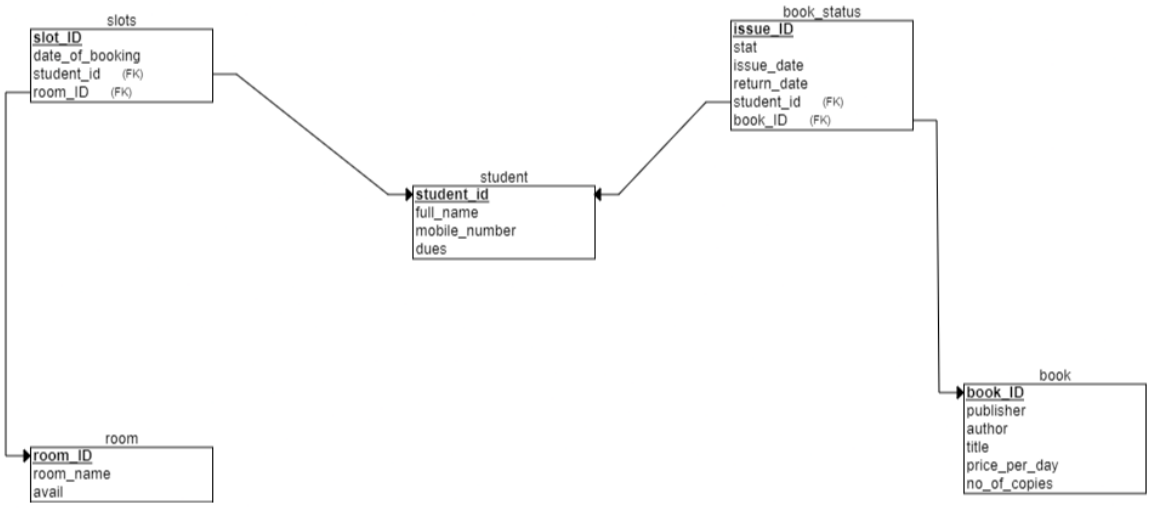
student(<u>student_id</u>, full_name, mobile_number, dues)

book(<u>book_ID</u>, author, publisher, tiltle, price_per_day, no_of_copies)

book_status(<u>issue_ID</u>, student_id, book_ID, stat, issue_date, return_date)

slots(<u>slot_ID</u>, student_id, room_ID, date_of_booking)

room(<u>room_ID</u>, room_name, avail)



Based on the above ER diagram and relationship schema, we have 3 entities with 2 relations connected with each other in the form of tables. This logical schema design has been implemented as it is in the form of physical schema on MySQL. We have tried to reduce the redundancy as much as possible through our database design. Each table has only one primary attribute to uniquely identify each of its entries and to reduce inconsistencies.

1. Student – Student table has the student ids as its primary key for the whole table where we store the full name and mobile number for the students. We also calculate the dues for the students where it is calculated from the day 1 when the book is issued.
2. Book – Book table has the book ids as the primary key. We store the title, publisher and author for each book. Each book has a different price per day which is used to calculate dues of a student as and when he or she issues that book. We also have the 'no_of_copies' attribute for the book relation to enable multiple students issuing the same book.
3. Book_status – We have used this relation between the students and the books table to keep track of all records of issued and returned books. The relation is one is to many from students to books as one student can issue any number of books and books can also be issued multiple times by the 'no_of_copies' attribute.
4. Room – We have used different room ids as the primary key for this table with different rooms like – Brainstorming, Club, Departments, etc. A Boolean attribute helps us to check the availability of the rooms.
5. Slots – This relation helps us to link the room and student relation with each other for the booking functionality. This again is a many to many relation as any number of students can book any number of rooms at a time but not simultaneously. (Consistent and concurrent implementation of database)

Data Normalization

We have maintained the Boyce Codd Normal Form (BCNF) in our database design over all the tables. As all the relations have a maximum of one primary key and it is ensured that each relation has its own primary key, i.e. uniquely identifiable attribute, we have achieved 3rd Normal Form.

1st Normal form is ensured by not having any multivalued attribute in our database. We only allow single valued attributes as an entry in our database.

2nd Normal form is ensured by not having any partial functional dependency in our tables/relations. All non-prime attributes should be fully functionally dependent on the primary key or candidate key.

3rd Normal form is ensured by not having any transitive dependency among the non-prime attributes of an entity. This is again maintained by only having one attribute as the primary key for each table.

BCNF is further ensured by having only the primary key attributes on the LHS of any functional dependency of the relations in the database.