Data Base Project

“Library Management System”

Instructor’s Name

Dr. Osama Saffarini

Student's Name:

Ammar Twair

202012280

Deliver Date

3rd/Jan/2023

Introduction:

This project is made to handle and manage a library where members could borrow, reserve, and search for books in an efficient way.

The database stores information about each book, including title, author/s, category, publisher and available copies. It also manages the borrowing operation by storing when a book was borrowed and when it was or should be returned.

The library also applies fines for members who didn't return their books in time, therefore the database manages payments of these fines.

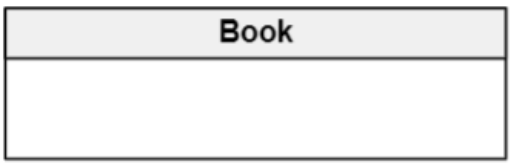
Requirements:

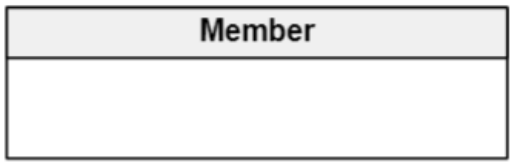
1. Each book has a unique id number.
2. Members can search for books by title, author, category and publication year.
3. A book can have multiple authors.
4. There may be several copies of the same book owned by the library.
5. Members could borrow books, and the system would store the date they borrowed the book.
6. Library staff can see information about the borrowing operations, registration and status of a member.
7. Members could reserve a book to borrow later if all of its copies are borrowed by other members.
8. Fines are applied after 7 days of borrowing a book if not returned, and members can pay these fines.

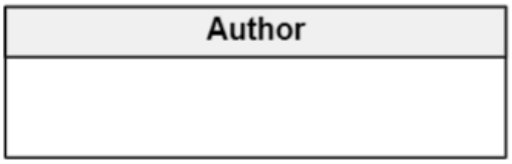
Designing Process:

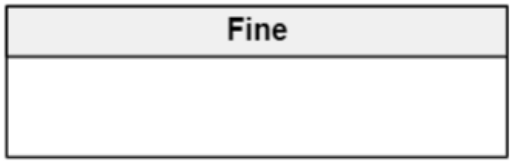
- At first, we can guarantee that we have these 4 essential entities:

Book, Author, Member, Fine

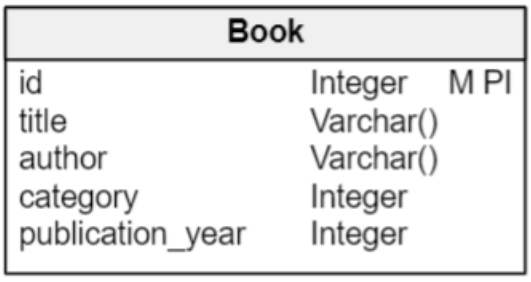


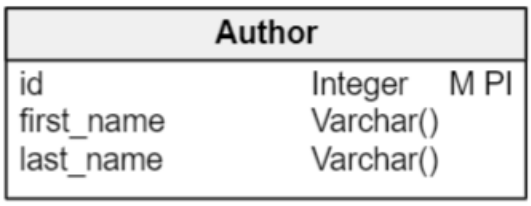




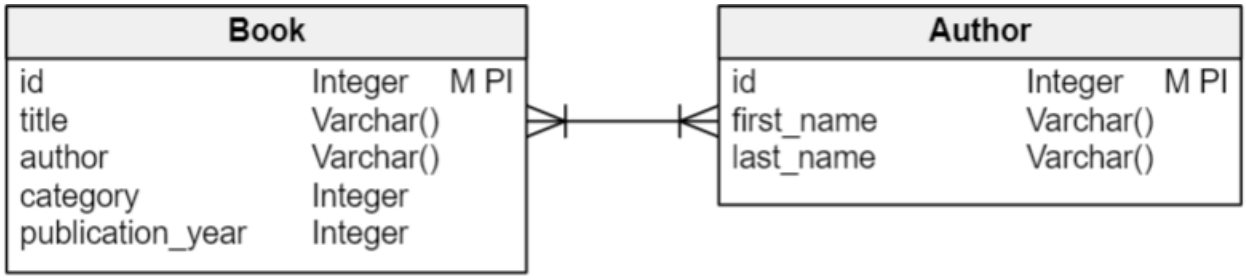


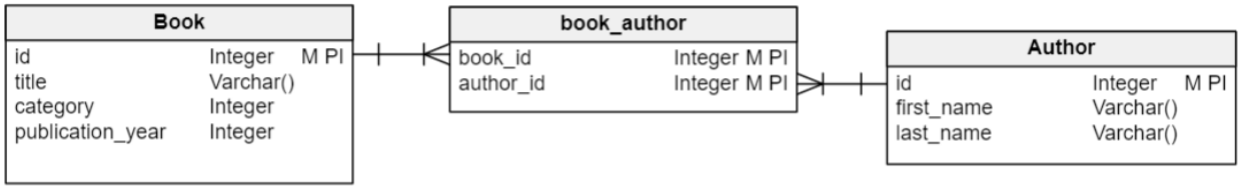
- A book should have a unique id, and could be searched by its author, title, category, and publication year as the first and second requirements suggest. An author has an id, a first name, and a last name.



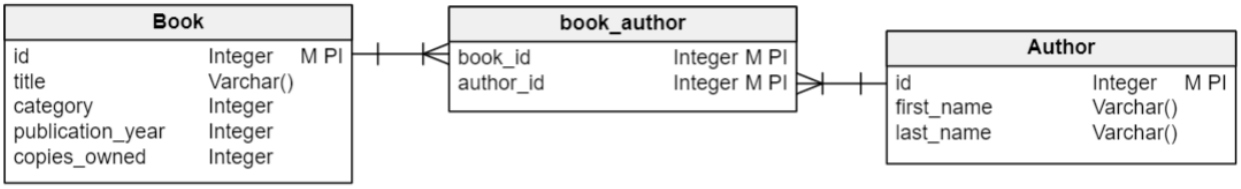


- A book could be written by multiple authors as the third requirement suggests and it's trivial that an author could write multiple books, so the relationship between book and author is many-to-many, which means we have to break it into one-to-many relationships by adding a table between them called Book_Author.



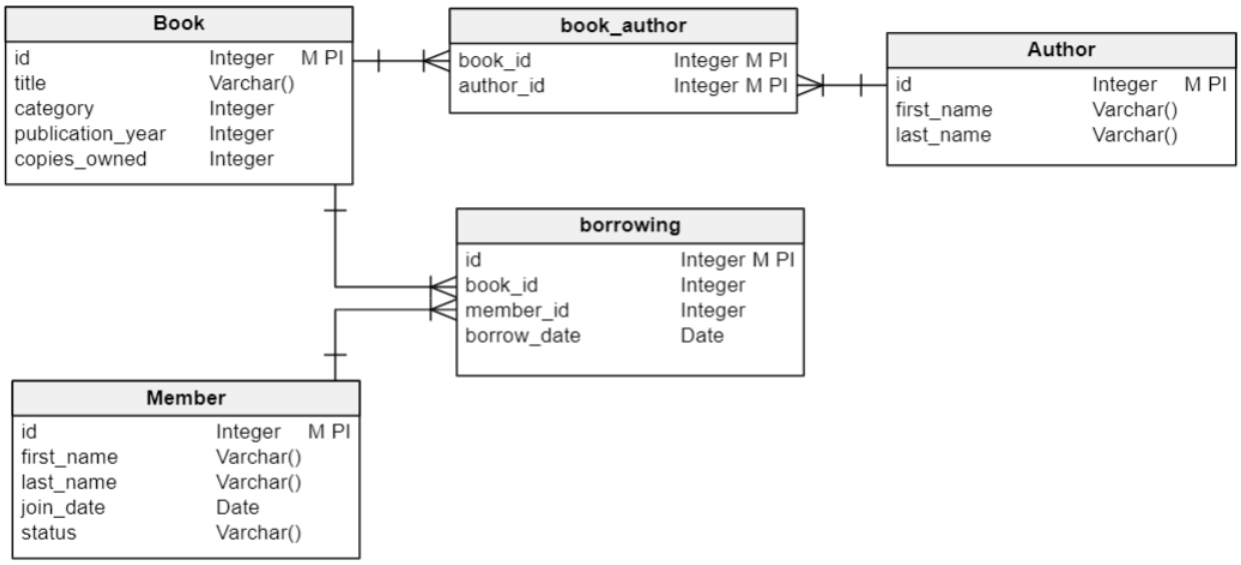


- The library could have several copies of the same book as the fourth requirement suggests, so we need an attribute for copies owned by the library.

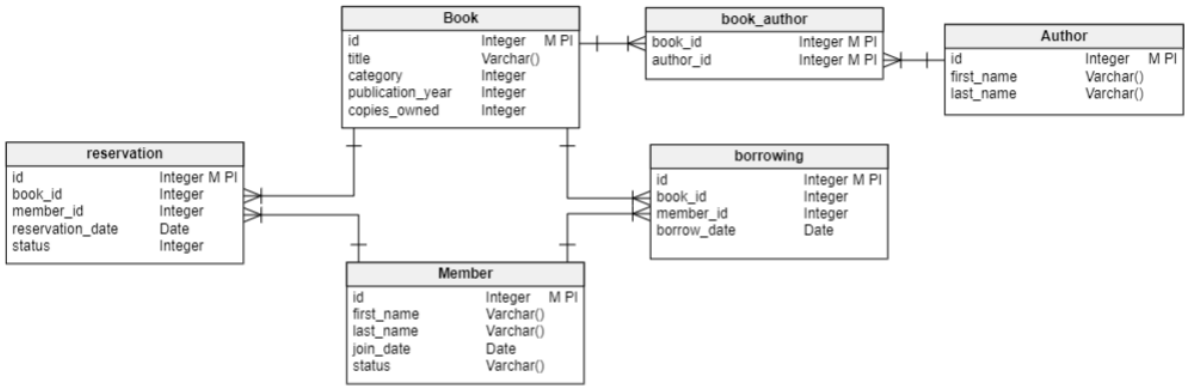


- Members should be able to borrow books as the fifth requirement suggests, which is a relationship between ‘member’ and ‘book’ tables, each member could borrow several books, and each book could be borrowed by several members, so it’s a many-to-many relationship which we need to break into two one-to-many relationships using table ‘borrowing’ which will store also the date in which member borrowed a book.

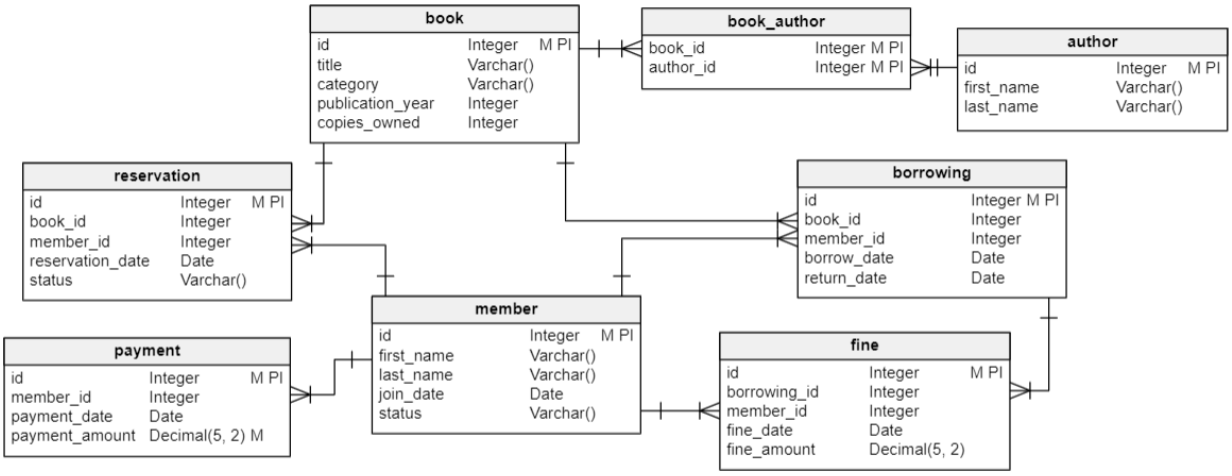
- Library staff must be able to see information about members as the sixth requirement suggests: id, first name, last name, when they’ve joined and the status of their membership. These are the attributes of the ‘member’ table.



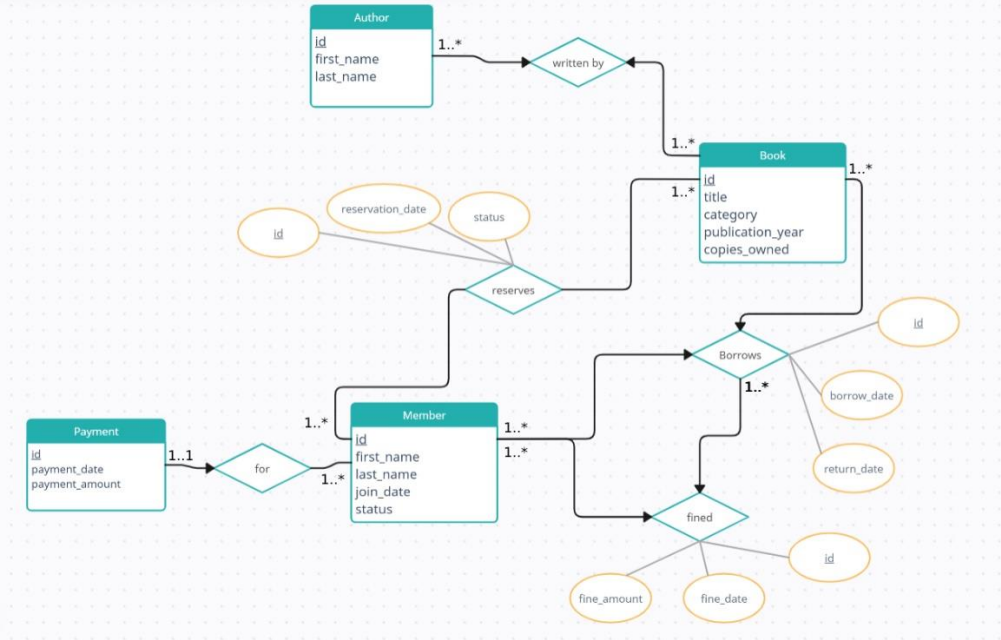
- If there isn’t an available copy of a book that a member wants to borrow, he could ask to reserve that book so that he borrows it as soon as it becomes available as the seventh requirement suggests. This is a relation between ‘book’ and ‘member’, a book could be reserved by multiple members, and a member could reserve multiple books so it’s many-to-many relationship, we need to break it using ‘reservation’ table.



- If a member didn’t return the book after 7 days of borrowing, a fine will be applied to him as the final requirement suggests. For that we need two tables, ‘fine’ table to store the information about the fine that a member has got for a particular borrowing operation, so it’s a relationship between ‘member’ and ‘borrowing’ tables, and ‘payment’ table to store members payments, and we need an extra attribute for borrowing table which is return date to store when a member returns a book.
- There is no need to assign a payment to a specific fine, as we assume in this library that the fine is a fixed amount which doesn’t increase with time, so a member can just do the payment for whatever fine he has which would decrease the total fine on him.



ER Diagram:



Relational Schema:

Book (<u>id</u>, title, category, publication_year, copies_owned)

Author (<u>id</u>, first_name, last_name)

Book_Author (<u>book_id</u>, <u>author_id</u>)

Member (<u>id</u>, first_name, last_name, join_date, status)

Reservation (<u>id</u>, book_id, member_id, reservation_date, status)

Borrowing (<u>id</u>, book_id, member_id, borrow_date, return_date)

Fine (<u>id</u>, borrowing_id, member_id, fine_date, fine_amount)

Payment (<u>id</u>, member_id, payment_date, payment_amount)

Normalization process:

- ## For book table:

  Book (id, title, category, publication_year, copies_owned)

  Functional dependencies:

  Id -> title , id-> category , id-> publication_year, id-> copies_owned

  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id.

- ## For author table:

  Author (id, first_name, last_name)

  Functional dependencies:

  Id -> first_name , id-> last_name

  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id

- ## For book_author table:

  Book_Author (book_id, author_id)

  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id

- For member table:

  Member (id, first_name, last_name, join_date, status)

  Functional dependencies:

  Id -> first_name , id-> last_name, id->join_date, id-> status

  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id

- For reservation table:

  Reservation (id, book_id, member_id, reservation_date, status)

  Functional dependencies:

  Id -> book_id , id-> member_id, id->reservation_date, id-> status

  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id

- For borrowing table:

  Borrowing (id, book_id, member_id, borrow_date, return_date)

  Functional dependencies:

  Id -> book_id , id-> member_id, id->borrow_date, id->return_date

  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id

- For fine table:

  Fine (id, borrowing_id, member_id, fine_date, fine_amount)

  Functional dependencies:

  Id -> borrowing_id , id-> member_id, id->fine_date, id->fine_amount

  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id

- For payment table:

  Payment (id, member_id, payment_date, payment_amount)

  Functional dependencies:

  Id -> member_id , id-> payment_date, id->payment_amount

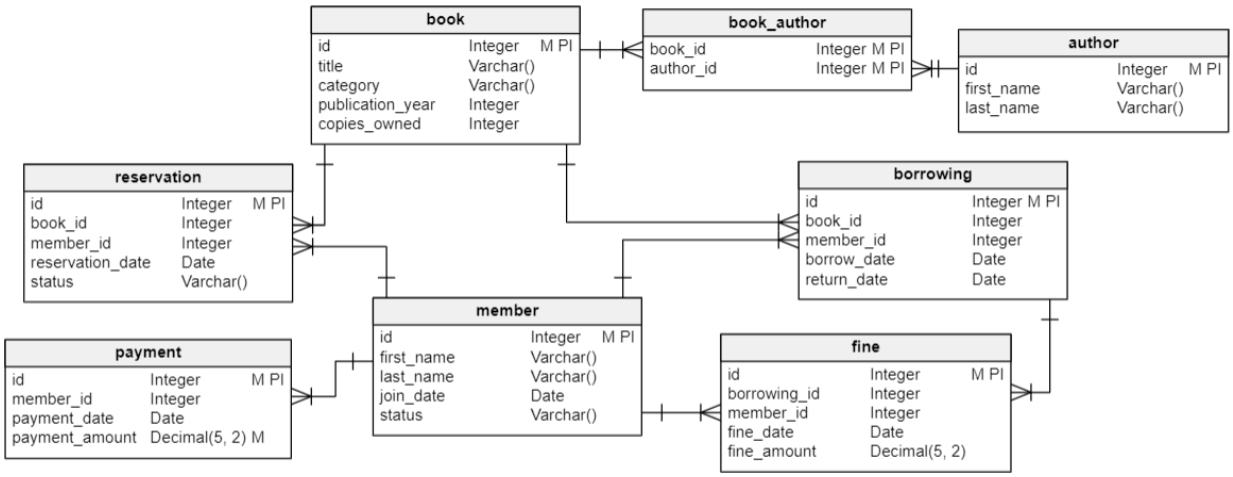
  1NF is satisfied, there is no multi valued attributes.

  2NF is satisfied, there is no partial dependencies on the primary key.

  3NF is satisfied, there is no transitive dependencies.

  Boyce codd is satisfied, because when A->B , A is the super key which is id

Each of the 8 tables is normalized, there for we can create the database safely as presented in the schema



DDL statements using oracle:

create table book (

id number primary key,

title varchar(255),

category varchar(255),

publication_year date,

copies_owned number

);

create table author(

id number primary key,

first_name varchar(255),

last_name varchar(255)

);

```
create table book_author(
book_id number references book(id),
author_id number references author(id),
constraint pk_book_author primary key (book_id,author_id)
);
create table member(
id number primary key,
first_name varchar(255),
last_name varchar(255),
join_date date,
status varchar(255)
);
create table reservation(
id number primary key,
book_id references book(id),
member_id references member(id),
reservation_date date,
status varchar(255)
);
```

```
create table borrowing(
id number primary key,
book_id references book(id),
member_id references member(id),
borrow_date date,
return_date date
);
create table fine(
id number primary key,
borrowing_id references borrowing(id),
member_id references member(id),
fine_date date,
fine_amount number
);
create table payment(
id number primary key,
member_id references member(id),
payment_date date,
payment_amount number
);
```