Report

Final Project

Author: Amata ANANTAPRAYOON,
Fartun JAMA,
Student ID: aa224iu, fj222jy
Semester: HT19
Course: Database Theory
Course code: 2DV513

Contents

1 Idea

We are thinking to create a console application which allows a user to store, edit or delete data in the database. The users of this are the librarian managing the DBMS. Some of the functionality is that users can add, delete and edit books data in the DBMS. Also can add, delete and edit member details in the DBMS. Also manage the loan details of a book when a member borrows a specific book.

The problem that we are trying to find a solution for is library system where members of the library can borrow and return books from the library. The members who borrow the books have 3 weeks to return the borrowed book. The library will have an application so they could manage the books in the library, also, library's member. To be able to manage the books that the members are borrowing whether a member returns the borrowed book in time or not. The application also offer another functionality such as, 1) getting the borrowed book that will expire in a given date 2) getting a best reader which is a list of members that have borrow the most books from the library 3) getting a best book which is a list of books that have been borrowed the most.

In the DBMS we are going to have a schema called Library which will have the following 4 tables:

1. Member
2. Book
3. Stock
4. LoanDetails

2 Logical model

The figure below presents the E/R diagram of our schema

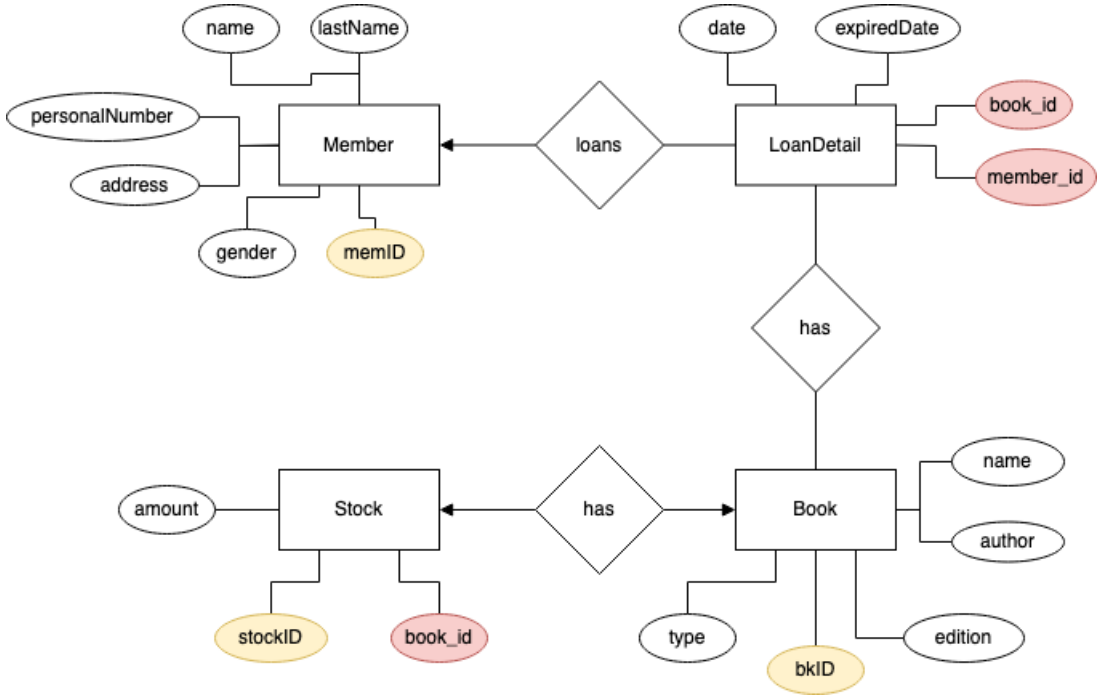


Figure 1: E/R diagram

The member entity set has the following attributes:

- memID (primary key)- the unique id the member has in the library system
- Name - the name of the person
- Last name - the person's last name
- Personal number - the personal number of the person
- Gender - female/male
- Address - the address of the person

The loan details entity set has the following attributes:

- member_id (foreign key): the id of the member that is borrowing the book
- book_id (foreign key): the book that is being borrowed
- date: the date that the book was borrowed
- expireDate: the date that the borrowing time expires

The book entity set has the following attributes:

- bkID (primary key): the id of the book
- author: the author of the book
- name: name of the book

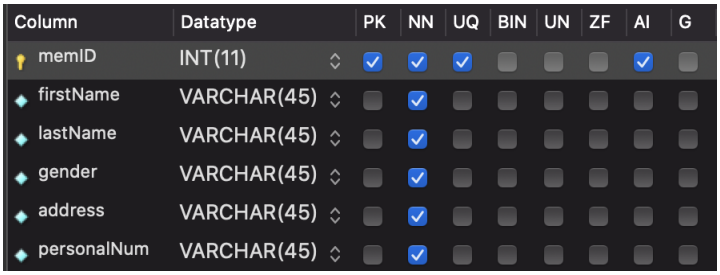
- edition: the edition of the book e.g first edition
- type: the type of the book e.g novel, horror

The stock entity set has the following attributes:

- stockID (primary key): the id of the book
- book_id (foreign key): the id of the book the stock is referring to
- amount: the amount of the book that is left in the library

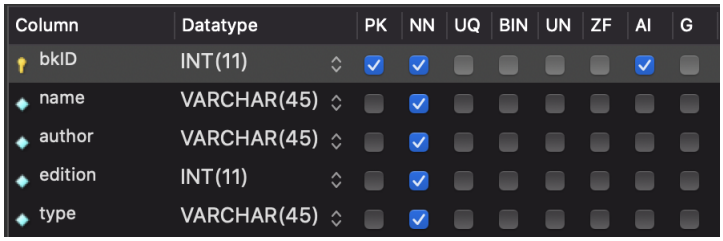
3 Design in SQL

We have translated our E/R diagram into the following tables:



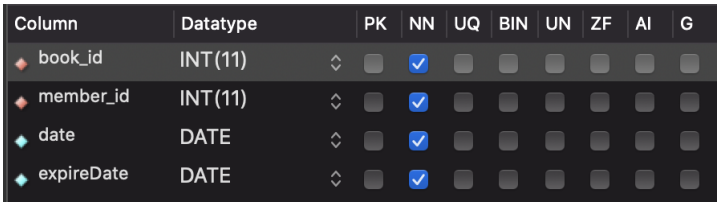
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
memID	INT(11)	✓	✓	✓				✓	
firstName	VARCHAR(45)		✓						
lastName	VARCHAR(45)		✓						
gender	VARCHAR(45)		✓						
address	VARCHAR(45)		✓						
personalNum	VARCHAR(45)		✓						

Figure 2: Member Table



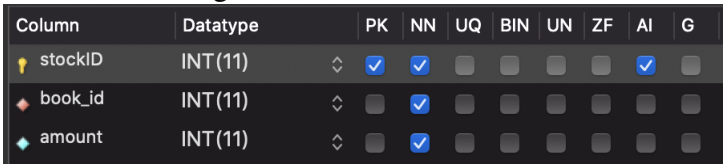
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
bkID	INT(11)	✓	✓					✓	
name	VARCHAR(45)		✓						
author	VARCHAR(45)		✓						
edition	INT(11)		✓						
type	VARCHAR(45)		✓						

Figure 3: Book Table



Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
book_id	INT(11)		✓						
member_id	INT(11)		✓						
date	DATE		✓						
expireDate	DATE		✓						

Figure 4: LoanDetails Table



Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
stockID	INT(11)	✓	✓					✓	
book_id	INT(11)		✓						
amount	INT(11)		✓						

Figure 5: Stock Table

Each of the tables have primary key except for the loan details table, which we thought it was not really necessary for us to have. The *LoanDetails* table has foreign keys *member_id* which is used to connect it to *Member* table using its primary key *memID*. Also *LoanDetails* has *book_id* foreign key which links to the book table through its primary key *bkID*. The stock table has a foreign key *book_id* which links to the book table through the primary key of book table.

The *Book* table has a relationship with *LoanDetails* and *Stock* table, when a member borrows a book from the library, a loan details is created in the database specifying all the attributes the loan details has. The member has 3 weeks from the day the loan has been made to return the book to the library. Also the stock table is updated, for instance if the book that is borrowed had 5 copies it will be updated to 4 copies left. When a book has a stock amount 0, a member can not make loan on the book. When a member returns a book, the loan details for that book is deleted and the stock of the returned book is updated.

The figure below shows how our database looks like after translating our design to collections in SQL. The four tables Member, LoanDetails, Book and Stock. Also *bookstock*, a view that we have created which we are going to discuss about in the next section.

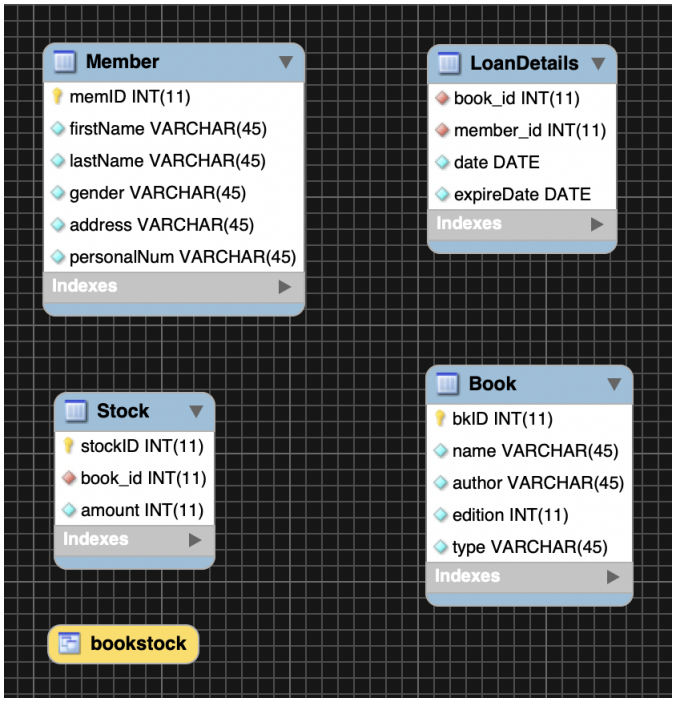


Figure 6: Library System

4 SQL queries

In the application when fetching data that has to do with member we take personal number as an input. For instance, if you want to delete/edit an existing member's details, the program will ask you to give the person's personal number. if member's id is exist then the application will pass the member's information to farther process. If the member's id is not exists, a error message is prompts to the user. The same thing happened with managing book. The only difference is the key which is a weak entity (name , edition). For instance, to delete/edit an existing book's details, the program will ask you to give the name and the edition of the book.

When we first run the application we create a view called *bookStock*, the program first checks if the view already exist if it does, we first drop the view then a new view is created. We then use the *bookStock* view to handle the stock queries inside the application.

```
1  Use Library;
2  CREATE VIEW bookStock As
3     select *
4     from Book
5     join Stock on Book.bkID = Stock.book_id;
```

The query below gets the details of a member who has borrowed a book that's about to expire. The query first gets a given expireDate, then by doing join on LoanDetails table and Member table it gets all the details of member/s that borrowed a book that has the given expire date. We have made this query to make it easier for librarians to notify members when their borrowed books are about to expire.

```
1  Use Library;
2  Select *
3     From Member
4     JOIN LoanDetails on Member.memID = LoanDetails.member_id
5  where expireDate in (
6        select expireDate
7        from LoanDetails
8        where expireDate = '2020-01-31'
9        )
10
```

The query below gets the members that have borrowed the most books from the library. It selects concat of the members first name and last name as MemberName, the count(memID) as numberOFLoans from the Member, by joining member and loan detals table, the grpup it by MemberName , that have numberOfLoans greater than zero, then order by numberOfLoans DESC listing the members that have borrowed the most books to least.

```
Use Library;
Select concat(firstName, ' ', lastName) as MemberName,
count(memID) as numberOfLaons
from Member
JOIN loanDetails on loanDetails.member_id = Member.memID
GROUP BY MemberName
HAVING numberOfLaons > 0
ORDER BY numberOfLaons DESC
```

The query below gets the books that have been borrowed the most. It is similar to the query above and lists the books that have been borrowed the most to the least borrowed books.

```
Use Library;
Select name as BookName,author as bookAuthor,
type as bookeType, edition as bookEd,
count(bkID) as numberOfLaons
from Book
JOIN LoanDetails on LoanDetails.book_id = Book.bkID
GROUP BY BookName,bookAuthor, bookeType,bookEd
HAVING numberOfLaons > 0
ORDER BY numberOfLaons DEsC
```

The query below gets the borrowed book based on a given personal number. The query takes a given personal number of a member and it fetches all the books this specific member have borrowed.

```
USE Library;
Select bkId
From LoanDetails
JOIN Member ON LoanDetails.memberId = Member.id
WHERE personalNum = '9901030000';
```

All the queries that are mentioned above and some other queries that are not mention here are implemented as functions in the application. The queries are presented as alternatives to the user when they run the program.

5 Implementation

Check the *README.md*

6 Supplemental video

Youtube Link: https://youtu.be/6u2YZrYoFDY