LUT University

Basic of Database System — Final Project
Library Management System

Group Number	7	
Group Member	Zhuoyi Cao	000205083
	Cheng Xiao	000205999
	Zitong Di	000211048

Introduction

I. Motivation for choosing a particular domain

Libraries are an important source of knowledge, information and entertainment for students, scholars and readers of all ages. It can also be said to be a kind of resource collection, books and users in the borrowing of a wide range of information, including a lot of information management, now there are many libraries are the initial start to use, there is no establishment of the corresponding book management data system, but the use of manual calculation, transcription to carry out, data processing workload, easy to make mistakes and data loss. So we think it may be a challenging task for libraries to manage a large number of books.

II. The general description of the domain

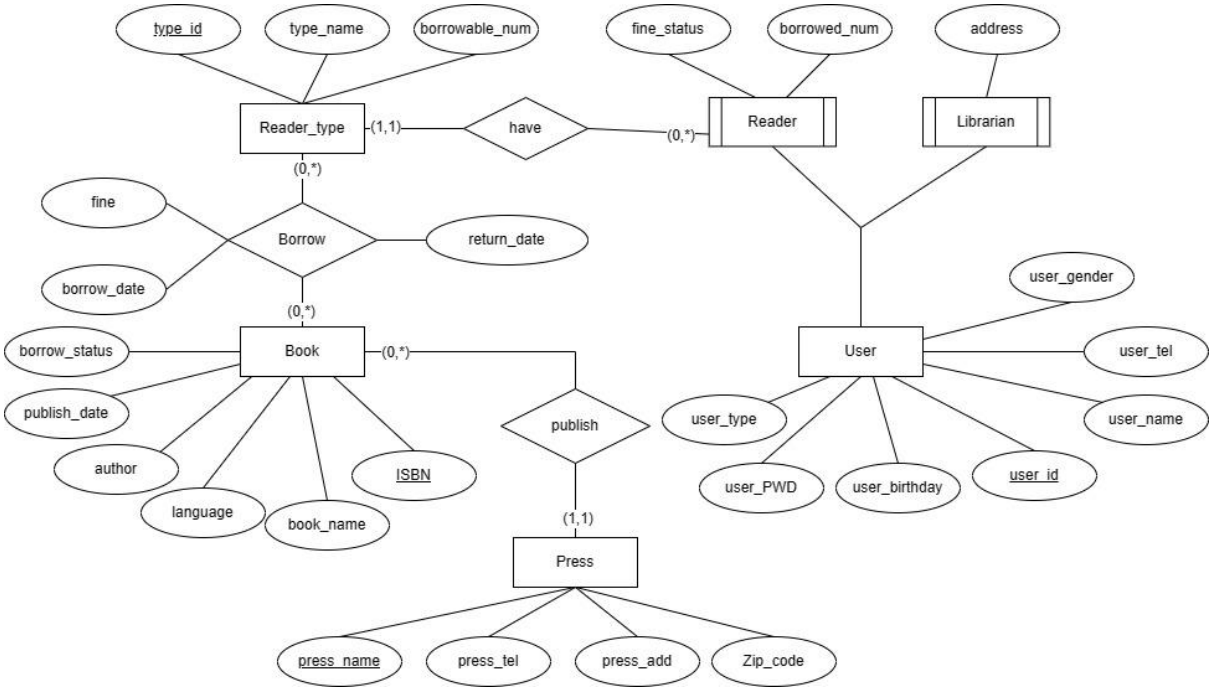
Library management systems (LMS) can automate and rationalize many processes in libraries and enable librarians and patrons to access and utilize resources efficiently. Therefore, the motivation for choosing this area was to create a reliable, user-friendly, simple and practical LMS that would benefit both libraries and their users.

III. Elicit the business rules of the domain

The library includes various entities such as books, users, press, etc. Each of these entities has specific relationships and restrictions. Among them users have their own username and password to log in to the library management system, users contain two categories, one for readers and one for librarians. Readers contain attributes such as ID, name, telephone number, gender, date of birth, fine status, and number of books checked out. Readers also have different types to determine the number of books they can borrow. Librarians contain attributes such as ID, name, gender, date of birth, telephone number and address. Books contain attributes such as ISBN, name, author, publication date, language, and borrowing status. A reader can borrow many books, and books can be checked out by many readers. It is also necessary to record the time when the book is borrowed and the return time. If the book is overdue, the reader should pay a fine based on the number of days overdue. Press contains attributes such as the publisher's name, telephone number, address, and Zip code. A press can publish many books, and a book can be

published by only one press. The library management system needs to manage all these relationships and restrictions while ensuring the smooth operation of the library.

The ERD of the real-world problem



The relational model, transformed from the ERD

User(user_id, user_name, user_gender, user_birthday, user_tel, user_PWD, user_type)
Librarian (user_id, address)
Reader (user_id, fine_status, borrowed_num, type_id(FK))
Reader_type(type_id(PK), type_name, borrowable_num)
Book(ISBN(PK), book_name, author, publish_date, press_name(FK))
Borrow(reader_id(FK), ISBN(FK), fine, borrow_date,
return_date)
Press(press_name(PK), press_tel, press_add, Zip_code)

Name the certain improvements that you have applied to the DB structure, and at which level (design or coding)

Normalization

We normalized two tables from 2NF to 3NF

1.

Reader (<u>user_id</u>, fine_status, borrowed_num, type_id, type_name, borrowable_num)
F = {(user_id) → (fine_status, borrowed_num, type_id, type_name, borrowable_num),
(type_id) → (type_name, borrowrable_num)}

The table we designed at the beginning is shown above, and it is clear that it has a transitive dependency, so this is 2NF. So that we need to transform it into 3NF to eliminate the transitive dependency.

Here are the results of our normalization:

Reader (<u>user_id</u>, fine_status, borrowed_num, type_id)
F = {(user_id)→ (fine_status, borrowed_num, type_id)}

Reader_type(<u>type_id</u>, type_name, borrowable_num)
F = {(type_id) → (type_name, borrable_num)}

2.

Book(ISBN, book_name, author, publish_date, press_name, press_tel, press_add, Zip_code)
F = {(ISBN) → (book_name, author, publish_date, press_name, press_tel, press_add, Zip_code),
(press_name) → (press_tel, press_add, Zip_code)}

The table we designed at the beginning is shown above, and it is clear that it has a transitive dependency, so this is 2NF. So that we need to transform it into 3NF to eliminate the transitive dependency.

Here are the results of our normalization:

Book(ISBN, book_name, author, publish_date, press_name)
F = {(ISBN) → (book_name, author, publish_date, press_name)}

Press(press_name, press_tel, press_add, Zip_code)
F = {(press_name) → (press_tel, press_add, Zip_code)}

Index

The database itself adds indexes to the primary and foreign keys of each table. In addition, we also added indexes for the columns book_name, author, and book_lang(book language). This is because we have designed to allow readers to find and filter books by book name, book author, and book language. So these three columns are often involved in where clause or group by clause. So we create indexes for them to increase the efficiency of search.

```
1  CREATE INDEX name_index ON book (book_name);
2  CREATE INDEX author_index ON book (author);
3  CREATE INDEX lang_index ON book (book_lang);
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 75 msec.
```

View

We have created a view named Unreturned Books. The main information stored in this view is information about books that have not been returned and information about readers who have not returned the book, as well as borrowing information, such as whether a fine was incurred, when it was borrowed and when it was returned, etc. This makes it easier for administrators to manage unreturned books more clearly.

Query Query History

```
1  CREATE VIEW Unreturned_Books(user_id, user_name, ISBN, book_name, press_name, borrow_date, return_date,fine)
2  AS
3  SELECT reader.user_id, users.user_name, book.ISBN, book.book_name, book.press_name, borrow.borrow_date, borrow.return_date, borrow.fine
4  FROM borrow, book, reader, users
5  WHERE borrow.ISBN=book.ISBN AND book.borrow_status = 'F' AND reader.user_id = borrow. user_id AND users.user_id = reader.user_id
```

Data Output Messages Notifications

	user_id character varying (10)	user_name character varying (50)	isbn character varying (50)	book_name character varying (200)	press_name character varying (100)
1	207004	gigi	1-4987-6014-7	Software Test Attacks to Break Mobile and Embedded Devices	Florence : Taylor & Francis Gro
2	207005	tella	1-78899-978-9	Mastering Kubernetes : master the art of container management by using the power of Kuberne...	Birmingham ; Mumbai : Packt
3	207006	coco	90-272-7141-0	Increased empiricism : recent advances in Chinese linguistics	Philadelphia : John Benjamins

Brief description of the application/program that you develop to interact with the database

I. Which programming language have you chosen?

We have chosen Java and SQL as our programming languages.

Java is widely used in developing management systems because of its platform independence, security, and scalability. It can run on multiple operating systems and devices, making it accessible to a wider audience. Additionally, Java has a vast library of open-source tools and frameworks that make development easier and more efficient.

SQL is used in conjunction with Java to manage data in the system. It provides a standardized way to access and manipulate data in relational databases. SQL allows users to retrieve data efficiently and write complex queries to perform data analysis. This makes it an essential component of any management system where data is crucial.

II. A list of the SQL queries implemented in the program.

1. SELECT * FROM users JOIN reader ON users.user_id = reader.user_id WHERE users.user_id = '"+id+"'
2. SELECT * FROM reader JOIN users ON reader.user_id = users.user_id WHERE "+state+"= '"+search_name+"'
3. SELECT * FROM book WHERE "+state+"= '"+search_name+"'
4. SELECT * FROM book WHERE "+state+" like '%"+search_name+"%';
5. UPDATE reader SET fine_status = 'T' where user_id='"+userid+"';
6. DELETE FROM borrow WHERE user_id='"+userid+"';
7. INSERT INTO book(isbn, book_name, author, press_name, publish_date, book_lang, borrow_status) VALUES('"+isbn+"',

'"+name+"', '"+author+"',
'"+press+"','"+pd+"','"+language+"','"+bstatus+"')

8. INSERT INTO press(press_name) VALUES('"+press+"')
9. DELETE FROM book WHERE isbn = '"+bnum+"'

III. What kind of end-user requirements are you meeting with these SQL queries? Write the imagined scenarios (per the chosen domain and its business rules) or questions from the end-user perspective

1. Users can log in to the Library Management System based on their user type, user id and the correct login password, otherwise they cannot log in. It ensures the security of the system to some extent.
2. If the user is a reader, the reader can view the information of all books. Also, he can search for the book he needs based on the book name, book author and book language.
3. The reader can also borrow books, depending on the availability of books and the number of books that the reader can borrow and the number that have been borrowed. For example, if the status of the book the reader wants to borrow is currently F, this means that the book has been borrowed and cannot be borrowed again. Also if the reader has borrowed more books than he can borrow, the reader is still unable to borrow.
4. When a reader successfully borrows a book, the program will automatically update the data in the borrow database. And the number of books borrowed by the reader will be increased accordingly.
5. Readers can also view their personal information, such as fine status, user id, password, etc.
6. If the user is a librarian, first of all, the librarian can also search books by book name, book author, and book language to make it easier to manage them.
7. The librarian can also edit the incoming records of books, such as adding books and deleting books.

8. The librarian can also adjust the reader's information to a certain extent, for example, after the reader pays the fine, the reader's fine status will be changed from F to T, and the system will automatically delete the corresponding borrowing record.

Notes:

Due to the database connection problem (the system is connected to our own database's password and port number, etc.), our exe file cannot be run on other computers after export. So that means we only uploaded the source code and the running video.
We're sorry about that, but we've made a lot of attempts to do that.