CSE562 Project: Library Database System

Junsong Huang
Yi Jing
50292448
50295580

ABSTRACT

In the modern society, different organizations need powerful systems to store and manage information. Then, the database system starts to play this important role. This report intends to demonstrate our database design and implementation for a library. We analyzed the design of our relational tables and the system's performances. This project is based on the relational database. But we also migrated all data to NoSQL database system to examine its special characteristics and performances.

1 INTRODUCTION

Libraries are the places for people to obtain knowledge, enrich experiences or even enjoy a peaceful moment. Universities and governments provide libraries for students and locals. Thus, to offer great experiences for people and organized the libraries efficiently, it's of vital importance for libraries to have a well designed library database system. The database system we designed is specialized for school libraries. The problem the database system plan to solve is managing a large amount of books. In the meantime, there are many students who want to borrow the book. Storing important information using the Excel would be problematic when the amount of information becomes increasingly larger.(for example, it is hard to figure out the time in which the student returned the book or whether a certain book is still in stock now, or not. If not, who borrow it) With the help of the database system, librarians are capable of managing the book inventory and taking records of student borrowing. The reason why a database is in demand is that there are many relations needed to be maintained. In addition, millions of books(Take UB as an example) and thousands of students' information can be stored in database safely compared with an excel file. When it comes to an Excel file, in a certain table, if a tuple is changed, the attribute in other sheet referencing this sheet remains the same. Thus, it may cause data inconsistency if the administrator fail to update the referencing sheet. In the database, there exists many constraints which excel cannot handle smoothly. Secondly, if we want to query something from many tables, it is easier for the IT developer to write sql other than create many sheet to "vlookup", not to say that the query result can be expressed in a website, so when a student and the administrator can get access to the Internet, they can do the action they want. However, we cannot combine the excel with the web page.

2 TARGET USER

The administrators and students will use this database. The administer is responsible for administering the database. When new books are purchased by the library, the administrator needs to insert the new book into the table which keeps records of books if this book was the first time brought to this library. When a book is destroyed, the administrator need to delete this book. Moreover, the administrator is expected to maintain the return list to remind the student to return the book on time. For students, when they need to borrow a book, he can login in that system to check if this book exist and available in this library, if a book is available, he can borrow it. After borrowing a certain book, if he forget the return date, he can use the system to check the return deadline and take his time.

3 LIST OF RELATIONS AND SCHEMA

3.1 User Table

The attributes for user table are shown as follows:

- `user_id`: Primary key, Integer, not null
- `user_email`: Varchar(40), not null
- `user_password`: Varchar(40), not null
- `type`: Integer, not null

`use_id` is the primary key which can identify one tuple in this relation. In the login page, the user should enter the unique `user_email` and the corresponding password `use_password` to login. The system will check the back-end database to find that whether the user email and the password are matched or not. Every user must have an email. Because when user want's to reset the password, he can find a way to change it. The type attribute indicates the identity of the user. 0 is student, 1 is administrator.

3.2 Student Table

The attributes for student table are shown as follows:

- `student_id`: Primary key, Integer, not null
- `student_name`: Varchar(40), not null
- `grade`: Integer, not null
- `gender`: Varchar(10), not null
- `user_id`: Integer, not null, foreign key

The `student_id` serves as the primary key. Given a `student_id`, we can retrieve the student's basic information. The `user_id` is a foreign key referencing User($user_id$). With a tuple in the student table, we can trace back to the original User table.

3.3 Administrator Table

The attributes for administrator table are shown as follows:

- `admin_id`: Primary key, Integer, not null
- `admin_name`: Varchar(40), not null
- `user_id`: Integer, not null, foreign key

3.4 Classification Table

The attributes for classification table are shown as follows:

- `classification_id`: Primary key, Integer, not null
- `classification_name`: Varchar(40)

- `classification_intro`: Varchar(100)

Each book would be classified to a category. To achieve BCNF, we project classification out to an independent table. The classification id is unique, so if we have the classification id, we can get the classification name and its introduction.

3.5 Book Table

The attributes for book table are shown as follows:

- `book_id`: Primary key, Integer, not null
- `book_name`: Varchar(40), not null
- `author`: Varchar(40), not null
- `classification_id`: Integer, foreign key
- `status`: Varchar(10), (available, borrowed)

In this table, the primary key is the `book_id`. Because it is unique, it can represent a tuple in this table. Although, the book name may be the same. `classification_id` is the foreign key referencing classification table(`classification_id`). If the `classification_id` in the classification table is deleted, it will execute cascade delete. The status attribute would represent the availability of this book and it only has two values. "Available" indicates the book is available, and "borrowed" shows the book is unavailable.

3.6 Borrowing List Table

The attributes for borrowing list table are shown as follows:

- `borrowing_id`: Primary key, Integer, not null
- `student_id`: Integer, not null, foreign key
- `book_id`: Integer, not null, foreign key
- `isReturned`: Integer, (0,1)
- `borrowing_time`: Date

`student_id` is referencing Student table. Through this attribute, we can learn who borrowed this book. If the student id is deleted from the Student table, the records in borrowing list should take no action. Because the administrator would not be able to find the student since the student has already graduated or transferred to another school. However, if the book was removed from the inventories, this record would be deleted. `isReturned` represents whether a certain book is returned or not. 0 is not returned while 1 shows returned.

3.7 Return List Table

The attributes for return list table are shown as follows:

- `return_id`: Primary key, Integer, not null
- `student_id`: Integer, not null, foreign key
- `book_id`: Integer, not null, foreign key
- `borrowing_id`: Integer, foreign key
- `return_time`: Date

This table is similar to the borrow list table. However, all records would be maintained no matter what happened to the student or book.

4 WEB INTERFACE

We divide the user into two roles: the administrator and the student. The privileges of two roles are described in figure 1. We will deliver a complete web project including the HTML file, the java code which can connect to our MySQL server. On the web page, different roles have different operations. And the function button will direct the user to the corresponding operation page. After completing the modifications, the submit button would post the updated result to the back-end database. Final modifications would be made there.

Table 1: Adding book page

Operations	Enter Details
book name	(text to be filled)
author	(text to be filled)
classification id	(text to be filled)
	add(button)

Table 2: Administrator main page

book id	book name	author	classification id	status	edit	delete
value	value	value	value	value	edit(button)	delete(button)

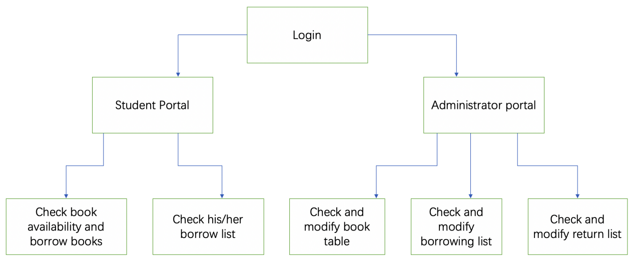
Login
Student Portal
Administrator portal
Check book availability and borrow books
Check his/her borrow list
Check and modify book table
Check and modify borrowing list
Check and modify return list

Figure 1: Web Interface Structure

4.1 Administrator

All books' information would be displayed in the administrator page. Thus, there's no need to set a new function to display all records. And in this page, two portal link will lead us to the borrowing list and return list.

4.1.1 Add Book. The manipulation table is shown in 1 table. After the administrator fill in the blanks and click submit, a new book would be inserted into the book table.

4.1.2 Books manipulation. In table 2, we can click edit or delete button to manipulate our records. The edit button will lead us to a page which is similar to the adding book page. In this page, the administrator can change the information of the book, While by clicking the delete button, the record for that book will be removed from the database.

4.1.3 Borrowing list manipulation. The button "go to borrow list" will take the administrator to the borrowing list page. And all borrow records would be displayed here. In table 3, clicking the delete button will delete this record in this table and when students return books, the administrator can click the return button to change the

Table 3: Borrowing list page

book name	student name	is returned	borrow time	delete	return
value	value	value	value	delete(button)	return(button)

Table 4: Return list page

book name	student name	return time	delete
value	value	value	delete(button)

Table 5: Student main page

(enter book name)	
submit	reset
List all books	
view my borrow list	
view my return list	
logout	

Table 6: Student query page

book id	book name	author	isBorrowed	Borrow
1	Java	Tom	available	borrow(button)
2	Database System	Peter	borrowed	borrow(button)

status of books into "available".

4.1.4 Return list manipulation. In table 4, the button "go to return list" will take the administrator to the return list page. When student returned the book, a return record would be added to this list. The administrator can only delete the return records.

5 STUDENT

Students can query the book in the library and search the book he/she wants and borrow it and check his borrowing status(which book he borrowed).

5.0.1 Student main page. In the main page in table 5, the student can find a blank space to enter the name of a book. By clicking the submit button, the query result will appear in the redirected page. And students can check his/her own borrow list. Then, a "List all books" button will lead the student to the page with all books.

5.0.2 Student query page. In the query page which is similar to the administrator main page shown in table 6, the student can click the borrow button to borrow the book. And this action will modify the records in the database. Then, this book would be unable to borrow by others.

6 DATA SOURCES AND FINAL PROJECT DELIVERY

We can not get the appropriate and suitable data from the Internet , so we will create our own data considering all of the situations and conditions.
We will deliver a complete web page including the html(form), the java code which can connect to our sql server. On the webpage, we will show different operations based on different roles of users(student who borrows the book and administrator who manipulate and maintain the database) as shown in the previous section.

7 FUNCTION DEPENDENCY ANALYSIS

7.1 User Table

Dependencies:

- `user_id` -> `user_email`
- `user_id` -> `user_password`

7.2 Student Table

Dependencies:

- `student_id` -> `student_name,grade,gender,user_id`
- `user_id` -> `student_name,student_id,gender,grade`

Both student id and user id can be super keys of the student table. Thus, our table satisfies BCNF.

7.3 Administrator Table

Dependencies:

- `admin_id` -> `admin_name`
- `user_id` -> `admin_name,admin_id`

Both admin id and user id can be super keys of the administrator table. Thus, this table satisfies BCNF.

7.4 Classification Table

Dependencies:

- `classification_id` -> `classification_name`

Classification id is our only super key. Thus, this table satisfies BCNF.

7.5 Book Table

Dependencies:

- `book_id` -> `book_name,author,classification_id,status`

Book id is the super key. Thus, this table satisfies BCNF.

7.6 Borrowing List Table

Dependencies:

- `borrowing_id` -> `student_id,book_id,isReturned,borrowing_time`
- `student_id,book_id,borrowing_time` -> `borrowing_id,isReturned`

Elements of left hand side from both dependencies can be super keys of the book table. The reason why the elements of left hand side from the second dependency can form a super key is that in certain time, a person borrowed a book is a unique action. Thus, this table satisfies BCNF.

7.7 Return List Table

Dependencies:

- `return_id -> student_id,return_time,book_id,borrowing_id`
- `borrowing_id, -> book_id,student_id`

The first dependency satisfy BCNF, because student id is the super key. However, the borrowing id is not. Hence, should split the table. Our new return table would be:

- `return_id`: Integer, not null, primary key
- `return_time`: Date, not null
- `borrowing_id`: Integer, not null, foreign key referencing the borrow list table.

The second table exists. Then, our return table can reference the borrowing list table. As a result, the return table satisfies BCNF.

8 QUERY EXECUTION ANALYSIS

8.1 First Scenario

If we want to select the total number of book in a certain classification.First, we write the sql like this:

- `select classi_name,count(*)`
- `from classi inner join book`
- `on classi.classi_id = book.classi_id`
- `group by classi_name`

The total time is 00:00:00:176 in the simulation.However, if we use "where" instead of the inner join:

- `select classi_name,count(*)`
- `from classi, book`
- `where classi.classi_id = book.classi_id`
- `group by classi_name`

The total time is 00:00:00:109. The data are very few. Only when the quantity of data is very large, we can see the improvement easily. Using inner join takes more time than "where" in some situation. However, if we exchange the tables (make book inner join classification), it takes less time.

8.2 Second Scenario

If we want to get how many times a student(we want to get his name) borrowed book in a chosen year.

- `with temp1(student_id)`
- `as`
- `(select student_id from borrow_list`
- `where year(borrowing_time)='2020')`
- `select count(*) ,student_name,borrow_list.student_id`
- `from student,temp1`
- `where student.user_id=temp1.student_id`
- `group by temp1.student_id`

The time cost is 00:00:00:453. Most of the time consumption is data transferring time, with almost zero execution time.

- `select count(*) ,student_name,borrow_list.student_id`
- `from student,borrow_list`
- `where student.user_id=borrow_list.student_id`
- `group by borrow_list.student_id`

The time cost is 00:00:00:031. All time consumption is execution time. If the data is large, we can use the first method because it takes less time on execution. It selects the record on the year first, so it saves time to do the Cartesian product of the two tables. However, it takes more time to do the transfer. Thus, when the data quantity is few, we prefer the second method.

9 COMPARISON BETWEEN RELATIONAL DATABASE AND NOSQL DATABASE

9.1 The Advantages and Disadvantages Of Storing The Data In the Relational Database

A relational database is consisting of a group of associated two-dimensional tables. And it must follow ACID characteristics. Two-dimensional tables are easy for people to understand due to their similarities to the logic world. From the perspective of the entire system, although there are many more tables associated, each table is independent, which maintains the simplicity of the whole system. What's more, the use of SQL language can save efforts for upper-level business developers because they don't have to focus on logical level. Last but not the least, special design and rich integrity dramatically reduces the probability of data redundancy and data inconsistency.

However, relational has its own bottleneck. Repeatedly query a fixed table, resulting in repeated waste of resources. In addition, the query which requires joining two tables, resulting in extended execution time, reduced server performance and user experience. And to some extent, multiple foreign keys and joining different tables caused by the relational characteristic increase the complexity of the system. Finally, high concurrent read and write requests are big challenges for traditional relational databases' disk I/O.

9.2 The Advantages and Disadvantages Of Storing The Data In the NoSQL Database

NoSQL database stores elements based on key-value pairs. And the structure is not fixed. Thanks to this characteristic, each tuple would not have to obey certain fixed structures, which can save time and space overheads and these tuples can add its own key-value pairs according to its needs. And NoSQL database has advantages for high performance concurrent reading and writing scenarios and quick query in the massive data. In our website, many student will add the borrow record or return record to the system. Because for the relational database, the index will be updated when there is a "write" record. On the contrary, there is no need to update the index for the Nosql.

Stemming from few constraints, the NoSQL Database won't offer "where" queries based on attributes. And it's difficult to reflect the integrity of the design. When storing comparatively simple data, the NoSQL databases have better performance, while the relational databases are good at handling complex data. The nosql doesn't support the transaction, so there is no "rollback" in the nosql,which will cause data inconsistency and wrong.

10 CONTRIBUTION

Junsong Huang: 50%
Yi Jing: 50%