THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學
FACULTY OF ENGINEERING
工程學院
WHERE CONCEPTS BECOME REALITY
Department of Computing
電子計算學系

COMP2411

DATABASE SYSTEMS

GROUP PROJECT

Library Management System (LMS)

Second Stage

2022-2023 (Semester One)

Lec001 Grp 14
HAO Jiadong 20084595d
WANG Hao 20076279d
JIANG Zheng 21095995d

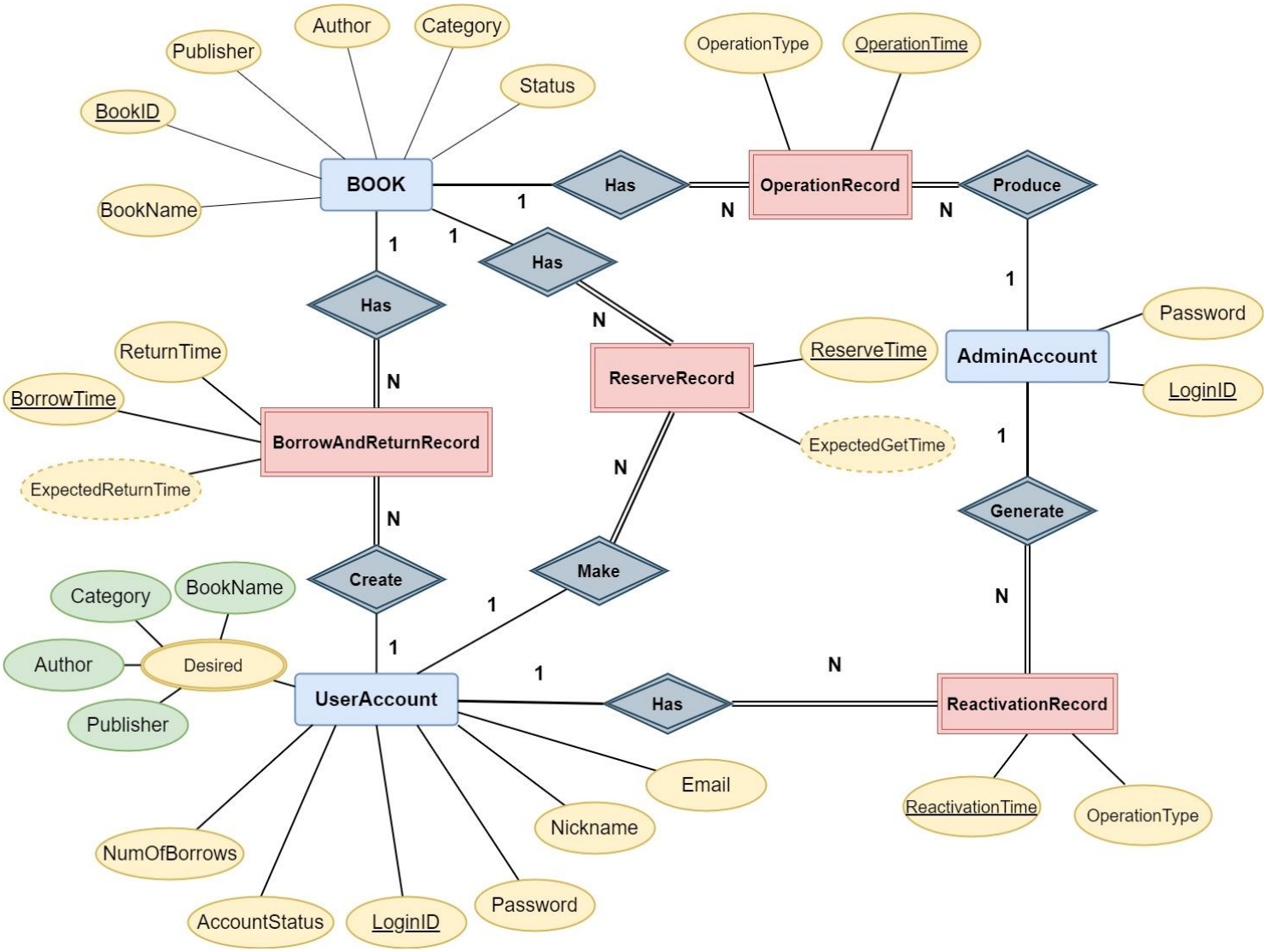
Words count:2459
Pages: 14

1. Overview

In this project, our group developed a user-friendly library management system which benefits both users and administrators of a library. We chose java as our basic programming language and used a lot of database knowledge learnt in the lectures such as ER Diagram, SQL, Integrity Constrains and so on to ensure the proper operation of our app. We not only perfectly fulfilled all the basic requirements, but also did lots of extensions based on our assumptions, which will be discussed in detail in the **User Guide**.

2. ER diagram design

After much thought and discussion, we finally decide to slightly modify the original ER diagram, relational schema and assumptions to improve the performance of our system, which are shown below:



3. Relational Schema

1.BOOK

BOOKID	BOOKNAME	AUTHOR	CATEGORY	PUBLISHER	STATUS

2.BORROW_AND_RETURN_RECORD

BOOKID	LOGINID	BORROWTIME	RETURNTIME	EXPECTEDRETURNTIME

3.RESERVED_RECORD

LOGINID	BOOKID	RESERVEDTIME	EXPECTEDRETURNTIME

4.USER_ACCOUNT

LOGINID	PASSWORD	NICKNAME	ACCOUNTSTATUS	NumOFBorrows	EMAIL

5.BOOK_DESIRED

LOGINID	BOOKNAME	PUBLISHER	CATEGORY	AUTHOR

6.ADMIN_ACCOUNT

LOGINID	PASSWORD

7.OPERATION_RECORD

BOOKID	LOGINID	OPERATIONTIME	OPERATIONTY

8.REACTIVATION_RECORD

LoginID_ADMIN	LoginID_USER	REACTIVATIONTIME	OPERATIONTYPE

*Tables & Attributes Explanation

1. BOOK
   A. BOOKID: Primary key to distinguish each book (even two identical books have different IDs).
   B. STATUS: A categorical value to represent the current status of a book: 0(Available), 1(Borrowed), 2(Reserved)

2. BORROW_AND_RETURN_RECORD
   A. BorrowTime: Primary key of this relation together with the "BOOKID", "LOGINID". If the primary key does not contain "BORROWTIME", there may be cases where the same user borrows the same book multiple times, and the records cannot be distinguished.
   B. ExpectedReturnTime: Derived attribute, which can be calculated from "BORROWTIME" plus 30 days.
   C. ReturnTime: On creating a record, the value of this field is set to be NULL. When a borrower returns the book, this field will be filled with the return time.

3. RESERVED_RECORD
   A. ReservedTime: Primary key of this relation together with the "BOOKID", "LOGINID". If the primary key does not contain "RESERVEDTIME", there may be cases where the same user reserves the same book multiple times, and the records cannot be distinguished.
   B. ExpectedGetTime: Derived attribute, which can be calculated from "RESERVEDTIME" plus three days.

4. USER_ACCOUNT
   A. AccountStatus: Store a categorical value of 0(Normal), or 1(Deactivated).
   B. NumOfBorrows: Store a sum of books the user currently holds.

5. BOOK_DESIRED
   After a user clicks the "desire" button of a certain book, the information of that book together with the user's LoginID will be stored in this table.

6. OPERATION_RECORD
   A table to record the admin's operations on a book or a user, such as updates on the information, uploading a new book, deleting an existing book, reactivate or deactivate a user.

7. REACTIVATION_RECORD
   A table to record the admin's reactivation operations on the users.

4. Description of LMS functions

1.Admin

Once logging in, an admin has 5 types of operations.

A. An admin can **reactivate** or **deactivate** a user account manually if needed.

B. An admin can **modify, upload, or delete a book.**

C. An admin can view the **analysis report** (The popular books or category and so on).

D. An admin can **look up the returning and borrowing records** of books.

E. An admin can **look up the logs of admin operations** (operations on users and books).

2.User

2.1 Search book

A user can **create his own account** to log in to the system.

In searching interface, a user can **search** books by name, author, category, publisher or all above**.** Once he selected a book, he can **borrow, reserve, and desire a book.**

2.2 Borrow, reserve and desire

A. In the user's personal interface, he can view all books borrowed, reserved and desired. He can also **return** a book he borrowed, **pick up** a book he reserved or **cancel the reservation**, or **cancel the desired book**.

B. For borrowing and reserving, a user **can only reserve and/or borrow three books at the same time.** A user can **borrow or reserve** a book only if that book now **has available copies**.

2.3 Email Notification

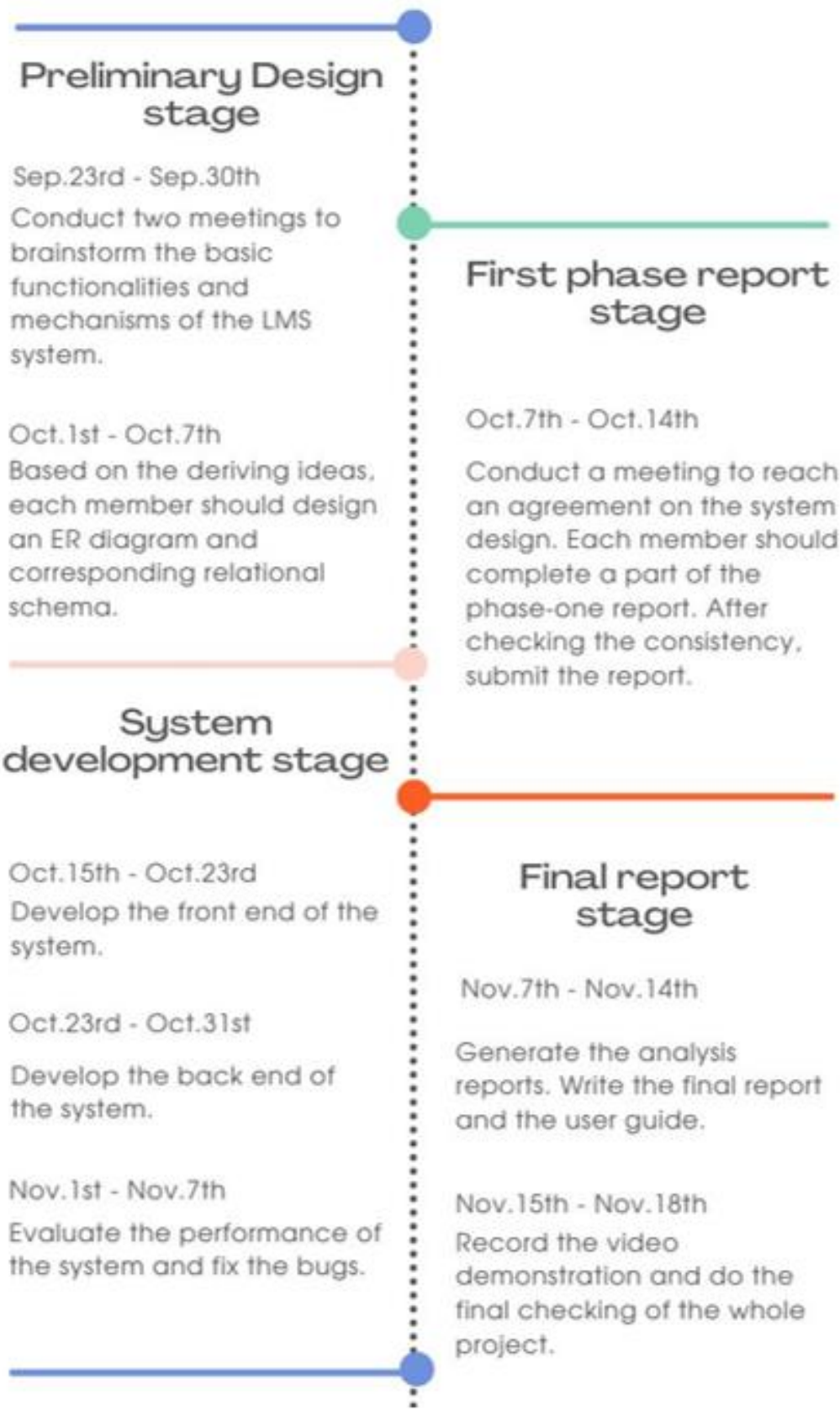
A. Once borrowed a book, the user must return it in one month. If he failed to return the book on time, his account will be **deactivated** by the system automatically. A **reminding email** will be sent to the registered email one week before the return deadline.

B. Once reserved a book, the user must pick it up in three days, otherwise, the reservation is invalid. **A reminding email** will be sent to the registered email one day before the reservation deadline.

C. For desiring, a user can only desire a book if the book has **NO available copies** currently (otherwise, he can directly borrow it). Once a desired book becomes available, the user will **receive an email.**

***(Note that for simplicity of demonstration and testing purpose, we purposely shorten the time interval for sending the reminding email, which means that if a user **doesn't return a borrowed book or pick up a reserved book in 1 minute**(simulating borrowing and reserving reminder before the deadline), he will receive a **reminding email**, if a user **doesn't return a borrowed book in 2 minutes**(simulating failing to return a book on time), he will receive an email and his account will be deactivated.)

5. Project Schedule



6. Code explanation of the main functions

In this part we briefly introduce the key feature code implementation.

6.1 Initial Oracle Database

Code Block 1

```
if(isExists("ADMIN_ACCOUNT")) conn.executeUpdate("drop table ADMIN_ACCOUNT");
if(isExists("BOOK")) conn.executeUpdate("drop table BOOK");
if(isExists("BOOK_DESIRED")) conn.executeUpdate("drop table BOOK_DESIRED");
if(isExists("BORROW_AND_RETURN_RECORD")) conn.executeUpdate("drop table
BORROW_AND_RETURN_RECORD");
if(isExists("OPERATION_RECORD")) conn.executeUpdate("drop table OPERATION_RECORD");
if(isExists("REACTIVATION_RECORD")) conn.executeUpdate("drop table REACTIVATION_RECORD");
if(isExists("RESERVED_RECORD")) conn.executeUpdate("drop table RESERVED_RECORD");
if(isExists("USER_ACCOUNT")) conn.executeUpdate("drop table USER_ACCOUNT");
```

Code Block 2

```
"create table USER_ACCOUNT" +
    "(" +
    "   LoginID      VARCHAR2(30) not null," +
    "   Password     VARCHAR2(30) not null,"
    "   Nickname     VARCHAR2(30)," +
    "   AccountStatus NUMBER      not null," +
    "   NumOfBorrows  NUMBER      not null,"
    "   Email        VARCHAR2(30)," +
    "   constraint USER_ACCOUNT_pk" +
    "      primary key (LoginID)" +
    ")"
```

```
"create table BOOK" +
    "(" +
    "   BookID   VARCHAR2(30) not null," +
    "   BookName  VARCHAR2(50) not null," +
    "   Author   VARCHAR2(50)," +
    "   Category  VARCHAR2(50)," +
    "   Publisher VARCHAR2(50)," +
    "   Status   NUMBER," +
    "   constraint BOOK_pk" +
    "      primary key (BookID)" +
    ")"
```

```
"create table BOOK_DESIRED" +
    "(" +
    "   LoginID  VARCHAR2(30) not null," +
    "   BookName  VARCHAR2(50) not null," +
    "   Author   VARCHAR2(50)," +
    "   Category  VARCHAR2(50)," +
    "   Publisher VARCHAR2(50)," +
    "   constraint BOOK_DESIRED_pk" +
    "      primary key (LoginID, BookName, Publisher)" +
    ")"
```

```
"create table RESERVED_RECORD" +
    "(" +
    "   LoginID      VARCHAR2(30) not null," +
    "   BookID       VARCHAR2(30) not null," +
    "   ReservedTime   TIMESTAMP(0) not null," +
    "   ExpectedGetTime TIMESTAMP(0) not null,"
+
    "   constraint Reserved_PK1" +
    "      primary key (BookID, LoginID,
ReservedTime)" +
    ")"
```

```
"create table BORROW_AND_RETURN_RECORD" +
    "(" +
    "   BookID          VARCHAR2(30) not null," +
    "   LoginID         VARCHAR2(30) not null," +
    "   BorrowTime       TIMESTAMP(0) not null," +
    "   ReturnTime       TIMESTAMP(0)," +
    "   ExpectedReturnTime TIMESTAMP(0) not null," +
    "   constraint PK1 " +
    "      primary key (BookID, LoginID, BorrowTime)" +
    ")"
```

```
"create table OPERATION_RECORD" +
    "(" +
    "   BookID       VARCHAR2(30) not null," +
    "   LoginID_ADMIN VARCHAR2(30) not null," +
    "   OperationTime  TIMESTAMP(0) not null," +
    "   OperationType  VARCHAR2(30) not null," +
    "   constraint OPERATION_RECORD_pk " +
    "      primary key (LoginID_ADMIN, BookID,
OperationTime)" +
    ")"
```

```
"create table REACTIVATION_RECORD" +
    "(" +
    "   LoginID_ADMIN   VARCHAR2(30) not null," +
    "   LoginID_USER    VARCHAR2(30) not null," +
    "   OperationType  VARCHAR2(30) not null," +
    "   ReactivationTime TIMESTAMP(0) not null," +
    "   constraint REACTIVATION_RECORD_pk" +
    "      primary key (LoginID_ADMIN,
LoginID_USER, ReactivationTime)" +
    ")"
```

```
'create table ADMIN_ACCOUNT " +
    "(" +
    "   LoginID  VARCHAR2(30),"+
    "   Password VARCHAR2(30),"+
    "   constraint ADMIN_ACCOUNT_pk" +
    "      primary key (LoginID)" +
    ")"
```

Code Block 3

```
"INSERT INTO USER_ACCOUNT VALUES ('1','1','Jay',0,0,'571532474@qq.com')");
"INSERT INTO USER_ACCOUNT VALUES ('2','2','wh1223',0,0,'973459859@qq.com')");
"INSERT INTO USER_ACCOUNT VALUES ('Testing1','2','Testing1',0,2,'1@qq.com')");
"INSERT INTO USER_ACCOUNT VALUES ('Testing2','2','Testing2',0,2,'2@qq.com')");
"INSERT INTO USER_ACCOUNT VALUES ('Testing3','2','Testing3',0,3,'3@qq.com')");
"INSERT INTO USER_ACCOUNT VALUES ('5','5','HAO Jiadong',0,0,'20084595d@connect.polyu.hk')");
"INSERT INTO USER_ACCOUNT VALUES ('4','4','WANG Hao',0,0,'20076279d@connect.polyu.hk')");

"INSERT INTO ADMIN_ACCOUNT VALUES ('admin','123')");
"INSERT INTO BOOK VALUES ('001','1984','George Orwell','anti-utopia','Thacker and Warburg',0)");
"INSERT INTO BOOK VALUES ('002','For Whom the Bell Tolls','Ernest Miller Hemingway','Love','The Brick Church Chapel',0)");
"INSERT INTO BOOK VALUES ('003','To the Lighthouse','Virginia Woolf','Modernist','Hogarth Press',0)");
"INSERT INTO BOOK VALUES ('004','An American Tragedy','Theodore Dreiser','novel','Horace Liveright',0)");
"INSERT INTO BOOK VALUES ('005','The Heart Is a Lonely Hunter','Carson McCullers','novel','Houghton Mifflin',0)");
"INSERT INTO BOOK VALUES ('006','The Heart Is a Lonely Hunter','Carson McCullers','novel','Houghton Mifflin',0)");
"INSERT INTO BOOK VALUES ('007','The Heart Is a Lonely Hunter','Carson McCullers','novel','Sylvia Beach',0)");
"INSERT INTO BOOK VALUES ('008','The Heart Is a Lonely Hunter','Carson McCullers','novel','Sylvia Beach',0)");
"INSERT INTO BOOK VALUES ('009','1984','George Orwell','anti-utopia','Thacker and Warburg',0)");
"INSERT INTO BOOK VALUES ('010','1984','George Orwell','anti-utopia','Thacker and Warburg',0)");
"INSERT INTO BOOK VALUES ('011','The Great Gatsby','Fitzgerald','tragedy','Charles Scribner Sons',0)");
"INSERT INTO BOOK VALUES ('012','The Great Gatsby','Fitzgerald','tragedy','Charles Scribner Sons',0)");
"INSERT INTO BOOK VALUES ('013','The Great Gatsby','Fitzgerald','tragedy','Charles Scribner Sons',0)");
"INSERT INTO BOOK VALUES ('014','The Great Gatsby','Fitzgerald','tragedy','Charles Scribner Sons',0)");
"INSERT INTO BOOK VALUES ('015','Ulysses','James Joyce','novel','Sylvia Beach',0)");
"INSERT INTO BOOK VALUES ('016','Lolita','Vladimir Nabokov','legendary drama','Olympia Press',0)");
"INSERT INTO BOOK VALUES ('017','Lolita','Vladimir Nabokov','legendary drama','Olympia Press',0)");
"INSERT INTO BOOK VALUES ('018','Lolita','Vladimir Nabokov','legendary drama','Olympia Press',0)");
"INSERT INTO BOOK VALUES ('019','Lolita','Vladimir Nabokov','legendary drama','Shanghai Translation Press',0)");
"INSERT INTO BOOK VALUES ('020','Lolita','Vladimir Nabokov','legendary drama','Shanghai Translation Press',0)");
"INSERT INTO BOOK VALUES ('021','Brave New World','Aldous Huxley','Science Fiction','Chatto & Windus',0)");
"INSERT INTO BOOK VALUES ('022','The Sun Also Rises','Ernest Hemingway','novel','Charles Scribner Sons',0)");
"INSERT INTO BOOK VALUES ('023','The Sun Also Rises','Ernest Hemingway','novel','Charles Scribner Sons',0)");

"INSERT INTO BOOK VALUES ('024','The Sun Also Rises','Ernest Hemingway','novel','Jonathan Kapp',1)");
"INSERT INTO BOOK VALUES ('025','AI 2041','Lee, Kai-Fu','Computing & Information Technology','Crown Publishing Group',1)");
"INSERT INTO BOOK VALUES ('026','AI 2041','Lee, Kai-Fu','Computing & Information Technology','Crown Publishing Group',1)");
"INSERT INTO BOOK VALUES ('027','AI 2041','Lee, Kai-Fu','Computing & Information Technology','Crown Publishing Group',1)");
"INSERT INTO BOOK VALUES ('028','Lolita','Vladimir Nabokov','legendary drama','Shanghai Translation Press',1)");
"INSERT INTO BOOK VALUES ('029','Lolita','Vladimir Nabokov','legendary drama','Shanghai Translation Press',1)");
"INSERT INTO BOOK VALUES ('030','Lolita','Vladimir Nabokov','legendary drama','Shanghai Translation Press',1)");
"INSERT INTO BOOK VALUES ('031','The Three-Body Problem','Liu Cixin','Science fiction','Chongqing Press',1)");
"INSERT INTO BOOK VALUES ('032','The Three-Body Problem','Liu Cixin','Science fiction','Chongqing Press',1)");
"INSERT INTO BOOK VALUES ('033','The Three-Body Problem','Liu Cixin','Science fiction','Chongqing Press',1)");
"INSERT INTO BOOK VALUES ('034','Village Teacher','Liu Cixin','Science fiction','Xin Hua Press',1)");
"INSERT INTO BOOK VALUES ('035','Village Teacher','Liu Cixin','Science fiction','Xin Hua Press',1)");
"INSERT INTO BOOK VALUES ('036','Village Teacher','Liu Cixin','Science fiction','Xin Hua Press',1)");


"INSERT INTO RESERVED_RECORD VALUES ('Testing1','024', TO_TIMESTAMP('2022-11-15 19:58:03','YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2022-1-18 19:58:03','YYYY-MM-DD HH24:MI:SS'))");
"INSERT INTO RESERVED_RECORD VALUES ('Testing1','025', TO_TIMESTAMP('2022-11-16 19:58:03','YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2022-1-19 19:58:03','YYYY-MM-DD HH24:MI:SS'))");
"INSERT INTO RESERVED_RECORD VALUES ('Testing2','026', TO_TIMESTAMP('2022-11-17 19:58:03','YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2022-1-20 19:58:03','YYYY-MM-DD HH24:MI:SS'))");
"INSERT INTO BOOK_DESIRED VALUES ('Testing1','AI 2041', 'Lee, Kai-Fu','Computing & Information Technology','Crown Publishing Group')");
"INSERT INTO BOOK_DESIRED VALUES ('Testing2','AI 2041', 'Lee, Kai-Fu','Computing & Information Technology','Crown Publishing Group')");
"INSERT INTO BOOK_DESIRED VALUES ('Testing3','AI 2041', 'Lee, Kai-Fu','Computing & Information Technology','Crown Publishing Group')");
"INSERT INTO BORROW_AND_RETURN_RECORD VALUES ('030','Testing1', TO_TIMESTAMP('2022-11-15 19:58:03','YYYY-MM-DD HH24:MI:SS'),null, TO_TIMESTAMP('2022-12-15 19:58:03','YYYY-MM-DD HH24:MI:SS'))"
"INSERT INTO BORROW_AND_RETURN_RECORD VALUES ('031','Testing2', TO_TIMESTAMP('2022-11-16 19:58:03','YYYY-MM-DD HH24:MI:SS'),null, TO_TIMESTAMP('2022-12-16 19:58:03','YYYY-MM-DD HH24:MI:SS'))"
"INSERT INTO BORROW_AND_RETURN_RECORD VALUES ('032','Testing2', TO_TIMESTAMP('2022-11-17 19:58:03','YYYY-MM-DD HH24:MI:SS'),null, TO_TIMESTAMP('2022-12-17 19:58:03','YYYY-MM-DD HH24:MI:SS'))"
"INSERT INTO BORROW_AND_RETURN_RECORD VALUES ('033','Testing3', TO_TIMESTAMP('2022-11-18 19:58:03','YYYY-MM-DD HH24:MI:SS'),null, TO_TIMESTAMP('2022-12-18 19:58:03','YYYY-MM-DD HH24:MI:SS'))"
```

As blocks 1,2,3, these SQL statements implement the function of initialization. When a user successfully logs in with an Oracle account:

(1) We will check if there are eight tables in the database that we are going to create,

and if there are duplicate table names, delete them and create them again.

(2) Insert the initial information into the eight tables (book information, loan information, user account, administrator account).

Initial User account	Password
1	1
2	2
Testing1	2
Testing2	2
Testing3	2
5	5
4	4

Initial Admin account	Password
admin	123

6.2 Search book

Code Block 4

```
query = "SELECT BOOKNAME,AUTHOR,PUBLISHER,CATEGORY,COUNT(*) FROM BOOK " +
 " WHERE BOOKNAME LIKE '%" + searchVal + "%'" + "OR AUTHOR LIKE '%" + searchVal +
"%'" +
 "OR CATEGORY LIKE '%" + searchVal + "%'"+ "OR PUBLISHER LIKE '%" + searchVal + "%'" +
 "GROUP BY BOOKNAME,AUTHOR,PUBLISHER,CATEGORY";
```

As block 4, these SQL statements implement the function of user searching books by "All fields". After a user selects the searching field to be "All" using the dropdown list and click the "Search" button

(1) We will select all the information of a book if one of its fields contains the search key word. To achieve this goal, we use SQL "LIKE" to find the matches.

(2) We group the result by BOOKNAME, AUTHOR, PUBLISHER and CATEGORY so that identical books will only appear once on the display table.

Code Block 5

```
query = "SELECT BOOKNAME,AUTHOR,PUBLISHER,CATEGORY, COUNT(*) FROM BOOK  WHERE
" + searchField.toUpperCase() + " LIKE '%" + searchVal + "%'"
+ "GROUP BY BOOKNAME,AUTHOR,PUBLISHER,CATEGORY";
```

As block 5, these SQL statements implement the function of user searching books by a certain field(i.e., book name, publisher, author or category). After a user selects a specific searching field using the dropdown list and click the "Search" button

(1) We will select all the information of a book if that selected field contains the searching key word. To achieve this goal, we use SQL "LIKE" to find the matches.

(2) Similarly, we group the result by BOOKNAME, AUTHOR, PUBLISHER and

CATEGORY so that identical books will only appear once on the display table.

6.3 Deactivate a user account

Code Block 6

```
QuartzManager.addJob("Check borrowing/returning and reserving records regularly", RootJob.class, "0/15
* * * * ?", "This is a description");

Oracle_Login.jobs.add(new Jobs(jobId,"SwapJob.class",remindingTime,mailDescription));

String sql = String.format("select * from BORROW_AND_RETURN_RECORD where BookID='%s' and
LoginID='%s'",BookId,LoginID);

String sqlUser = String.format("update USER_ACCOUNT set AccountStatus=1 where LoginID='%s' ",LoginID);
```

(1) Every 15 seconds, the system will call the RootJob.class to traverse the "BORROW_AND_RETURN_RECORD" table to find whether there are new jobs to add to a job waiting list.

(2) Inside the RootJob.class, we add a deactivation job to the job list. After "remindingTime" which is 2 minutes, the job will be executed, which means that the email with the content of "mailDescription" will be sent to the user whose account is going to be deactivated if the system find that the user hasn't returned the book at that moment.

(3) After 2 minutes, before the execution of the job, we use a SQL statement to check whether the user returned this book or not. If the user hasn't returned the book at that moment, we will send him the email and deactivate his account.

(4) Used to deactivate the user account by setting "AccountStatus" to 1.

6.4 User account operations (Borrow, Desire, Reserve, Return)

1.Borrowing

Code Block 7

```
query = "SELECT * FROM BOOK WHERE BOOKNAME = '" + bookName + "' AND PUBLISHER = '" +
publisher + "' AND AUTHOR = '" + author + "' AND CATEGORY = '" + category + "' AND STATUS = '2'" ;

query = "SELECT * FROM BOOK WHERE BOOKNAME = '" + bookName + "' AND PUBLISHER = '" +
publisher + "' AND AUTHOR = '"+ author + "' AND CATEGORY = '" + category + "' AND STATUS = '0'"

query = "SELECT NUMOFBORROWS FROM USER_ACCOUNT WHERE LOGINID = '" + Initial.ID + "'";

query="UPDATE USER_ACCOUNT SET NUMOFBORROWS = '"+ currentBorrows + "' WHERE
LOGINID = '"+ Initial.ID +"'";

query = "INSERT INTO BORROW_AND_RETURN_RECORD
(BOOKID,LOGINID,BORROWTIME,EXPECTEDRETURNTIME) VALUES('" + bookID + "','" + Initial.ID + "','"
+"TIMESTAMP \'" + ts + "\'," + "TIMESTAMP \'" + tsLater + "\')";
```

As block 7, these SQL statements implement the function of user borrowing books. After a user selects a book from the "Search" page and decides to borrow it:

(1) we first determine whether the user has made a reservation (status=2) or has not made a reservation (status=0). If a reservation is made, we will delete the

information in the "RESERVED_RECORD" table (change it to checked out).

(2) In addition, since users cannot check out more than three books at the same time, we also verify this in the column "NumberOfBorrows".

(3) Finally insert the borrowed book information into the table "BORROW_AND_RETURN_RECORD" and update the book status to 1 (checked out).

2. Desiring

Code Block 8

```
query = String.format("SELECT * FROM BOOK_DESIRED WHERE LoginID='%s' and BookName='%s' and Author='%s' and Category='%s' and Publisher='%s'",Initial.ID,bookName,author,category,publisher);
query = "SELECT * FROM BOOK WHERE BOOKNAME = '" + bookName + "' AND PUBLISHER = '" + publisher + "' AND AUTHOR = '"+ author + "' AND CATEGORY = '" + category + "' AND STATUS = '0'" ;
query = String.format("INSERT INTO BOOK_DESIRED (LoginID,BookName,Author,Category,Publisher) VALUES('%s','%s','%s','%s','%s')", Initial.ID,bookName,author,category,publisher);
```

As block 8, these SQL statements implement the function of user desiring books. After a user selects a book from the "Search" page and decides to desire it:

(1) Determine if there are available copies of the book selected by the user (the user can DESIRE the book when and only when there are no available copies of the book, i.e., it has all been checked out or reserved. When there are available copies of the book in users' desire lists, the system will send an email alert).

(2) When the book information does not exist in the user's original desires list, insert the book information into the table "BOOK_DESIRED ".

3. Reserving

Code Block 9

```
query = "SELECT * FROM BOOK WHERE BOOKNAME = '" + bookName + "' AND PUBLISHER = '" + publisher + "' AND AUTHOR = '" + author + "' AND CATEGORY = '" + category + "' AND STATUS = 0"
query = "SELECT NUMOFBORROWS FROM USER_ACCOUNT WHERE LOGINID = '" + Initial.ID + "'";
query="UPDATE USER_ACCOUNT SET NUMOFBORROWS = '"+ currentBorrows + "' WHERE LOGINID = '"+ Initial.ID +"'";
query="UPDATE BOOK SET STATUS = '2' WHERE BOOKID = \'"+bookID+"\'";
query = "INSERT INTO RESERVED_RECORD (BOOKID,LOGINID,ReservedTime,ExpectedGetTime) VALUES('" + bookID + "','" + Initial.ID + "'," +"TIMESTAMP \'" + ts + "\'," + "TIMESTAMP '" + tsLater + "\')";
```

As block 9, these SQL statements implement the function of user reserving books. After a user selects a book from the "Search" page and decides to reserve it:

(1) If there are still copies of the book, first determine whether the user has borrowed more than 3 books, if less than 3 books, update "NUMOFBORROWS".

(2) After successful reservation, the status of the book is updated to "2" (for reserved) and inserted into the "RESERVED_RECORD" table, the user needs to pick up the book within 3 days or the reservation will be cancelled.

4. Return

Code Block 10

```
query = "UPDATE BORROW_AND_RETURN_RECORD SET RETURNTIME = TO_TIMESTAMP('" + sd1 + "','YYYY-MM-DD HH24:MI:SS') WHERE BOOKID = '" +
bookID + "' AND LOGINID = '" + Initial.ID + "' AND BORROWTIME = TO_TIMESTAMP('" + time +"','YYYY-MM-DD HH24:MI:SS')";

query="UPDATE BOOK SET STATUS = '0' WHERE BOOKID = \'"+bookID+"\'";

query="UPDATE USER_ACCOUNT SET NUMOFBORROWS = '"+ currentBorrows + "' WHERE LOGINID = '"+ Initial.ID +"'";

query = String.format("DELETE FROM BOOK_DESIRED WHERE LoginID='%s' and BookName='%s' and Author='%s' and Category='%s' and Publisher='%s'",
Initial.ID,bookname,author,category,publisher);

query = "DELETE FROM RESERVED_RECORD WHERE BOOKID = '" + bookID + "' AND LOGINID = '" + Initial.ID + "' AND RESERVEDTIME =
TO_TIMESTAMP('" + time +"','YYYY-MM-DD HH24:MI:SS')";
```

As block 10, these SQL statements implement the function of user return books / cancel desiring/ cancel reservations. After a user selects a book record from the "My borrowing" page and decides to return/cancel it:

(1) Return: Update the value of "return time" from "null" to the current time and reduce the number of books borrowed by one for this user.

(2) Cancel Reservations: Delete the reservation record from the "RESERVED_RECORD" table and update number of borrows for the user.

(3) Cancel Desiring: Delete the corresponding book record from the "BOOK_DESIRED" table.

6.5 Notification (desire, return)

1. Send an email to users who desired a book which turned to be available

Code Block 11

```
ResultSet desireRS = oracleDB.executeQuery(String.format("select * from BOOK_DESIRED
where BookName='%s' and Author='%s' and Category='%s' and Publisher='%s' ",BookName, Author, Category,
Publisher));
while (desireRS.next()) {
   String loginId = desireRS.getString("LoginID");
   ResultSet userRS = oracleDB.executeQuery(String.format("select * from USER_ACCOUNT where LoginID='%s' ",
loginId));
   if (userRS.next()) {
      String email = userRS.getString("Email");
      String subject = String.format("One Desired Book Available!", BookName);
      String description = String.format("The book '%s' you desired is now available, you can borrow it through our
system!<br>", BookName);
      description += String.format("BookName: %s <br>", BookName);
      description += String.format("Author: %s <br>", Author);
      description += String.format("Category: %s <br>", Category);
      description += String.format("Publisher: %s <br>", Publisher);
      description += "PAO YUE-KONG LIBRARY<br>";
      description += sd1; //DATE
      new EmailControll().SendLibraryEmail(email, subject, description);
   }
```

As block 11, these SQL statements implement the function that when a book becomes available, the users who desired that book would receive an email.

(1) Based on the BOOKID of the returned book, we can easily get the information (book name, author, category, and publisher) of that book.

(2) In the table "DESIRE_RECORD", we use such information as a condition to

search if there is any user who desires such a book. If found, we send him an email using the “EmailCOntroll().SendLibraryEmail()” function.

2. When a book is due soon (for testing: 1 minute later), the user who hasn’t returned it will receive an email.

Code Block 12

```
QuartzManager.addJob("Check borrowing/returning and reserving records regularly", RootJob.class, "0/15 * * * * ?", "This is a description");

Oracle_Login.jobs.add(new Jobs(jobId,"SwapJob.class",returnTime,mailDescription));

String sql = String.format("select * from BORROW_AND_RETURN_RECORD where BookID='%s' and LoginID='%s'",BookId,LoginID);
```

The logic is basically the same as account deactivation.

(1) Every 15 seconds, the system will call the RootJob.class to traverse the “BORROW_AND_RETURN_RECORD” table to find whether there are new jobs to add to a job waiting list.

(2) Inside the RootJob.class, we add a sending-email job to the job list. After “remindingTime” which is 1 minute, the job will be executed, which means that the email with the content of “mailDescription” will be sent to the user who should be reminded.

(3) After 1 minute, before the execution of the job, we use a SQL statement to check whether the user returned this book or not. If the user hasn’t returned the book at that moment, we will send him the email.

6.6 Generate Analysis report

Code Block 13

```
String query = "SELECT " + selectedItem + ", SUM(SUM1) AS Total " +"FROM
(SELECT " + selectedItem + ", COUNT(*) AS SUM1 FROM BORROW_AND_RETURN_RECORD a, BOOK b WHERE
a.BOOKID = b.BOOKID GROUP BY " + selectedItem +
" UNION ALL " +
"SELECT " + selectedItem + ", COUNT(*) FROM RESERVED_RECORD a, BOOK b WHERE a.BOOKID = b.BOOKID GROUP
BY " + selectedItem +
" UNION ALL " +
"SELECT " + selectedItem + ", COUNT(*) FROM BOOK_DESIRED GROUP BY " + selectedItem + ")" +
" GROUP BY " + selectedItem +  " ORDER BY Total DESC";
```

As block 13, these SQL statements are used to generate the analysis report by listing the most popular book name, author, category and publisher (measured by the total number of borrowings, desiring and reserving records).

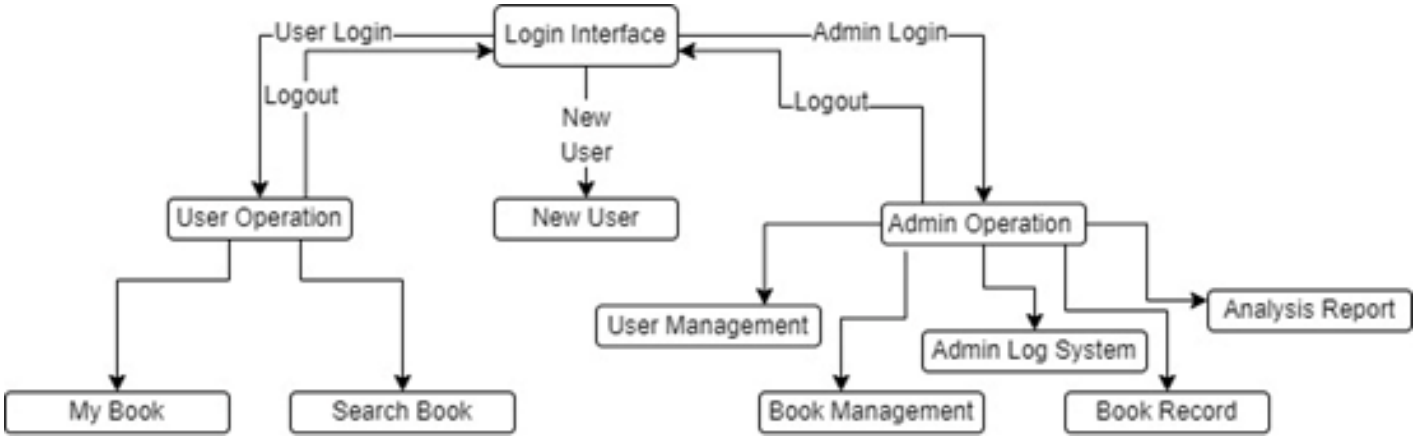
(1) The variable “selectedItem” represents the data field the analysis is focused on (i.e., book name, author category or publisher).

(2) We adopt a nested query. First, we group the books with the same value in their “selectedItem” column in the tables “BORROW_AND_RETURN”, “RESERVED_RECORD” and “BOOK_DESIRED” respectively and count how

many records are there for each specific “selectedItem” value in each table. Then we group the three tables results together and order them in descending order to get the final results of the popularity.

(For more information, see the word document "Generating Analysis Reports".)

7. Overview of LMS interface relationships



(For more information, see the word document "User Guide".)

8. Technics used

IDE: IntelliJ
Programming Language: JAVA, SQL
Database: Oracle
JDK: OpenJDK 19.0.1
Front end tool: Swing GUI Designer
Online software source code hosting service platform: Git Hub
Diagrams drawing: diagrams.net

9. Reflection

1. In the process of writing the code, we encountered numerous difficulties. For example, how to use java to connect to the COMP oracle server through SSH, how to solve the problem of exceeding the thread limit when accessing the database and so on. Those problems let us realize the fact that in practical work, the most difficult tasks and most troublesome problems can't be solved only with knowledge from books. What a good programmer needs to do is definitely not just how to get good

grades in the examinations, but to improve the ability to analyze and solve problems through experiences gained from a lot of practice. So, in the future life, we will cherish the opportunities to do a project, by constantly summarizing and reflecting on the work, we can improve our coding skills.

2. This experience let us realize that one person's ability is limited, only through effective cooperation can maximize the efficiency of the whole team. Frequent communication, meetings, and work reports are necessary. There were many times we found that two people were writing the same function at the same time because of lack of communication, which was a huge waste of time and energy for each of us and slowed down the project.
When assigning tasks, we should make detailed records and time planning, and the deadline should be accurate to the hour. Only when every member of the team is united, cooperative and enthusiastic can the team achieve the best performance.

10. Conclusion

This project gave us a deeper understanding of real-life database operations. Taking the "Library Management System" as an example, we completed three main tasks: data initialization, data classification, and data presentation, and successfully applied our knowledge in real-life situations.