COMP2016 Database Management

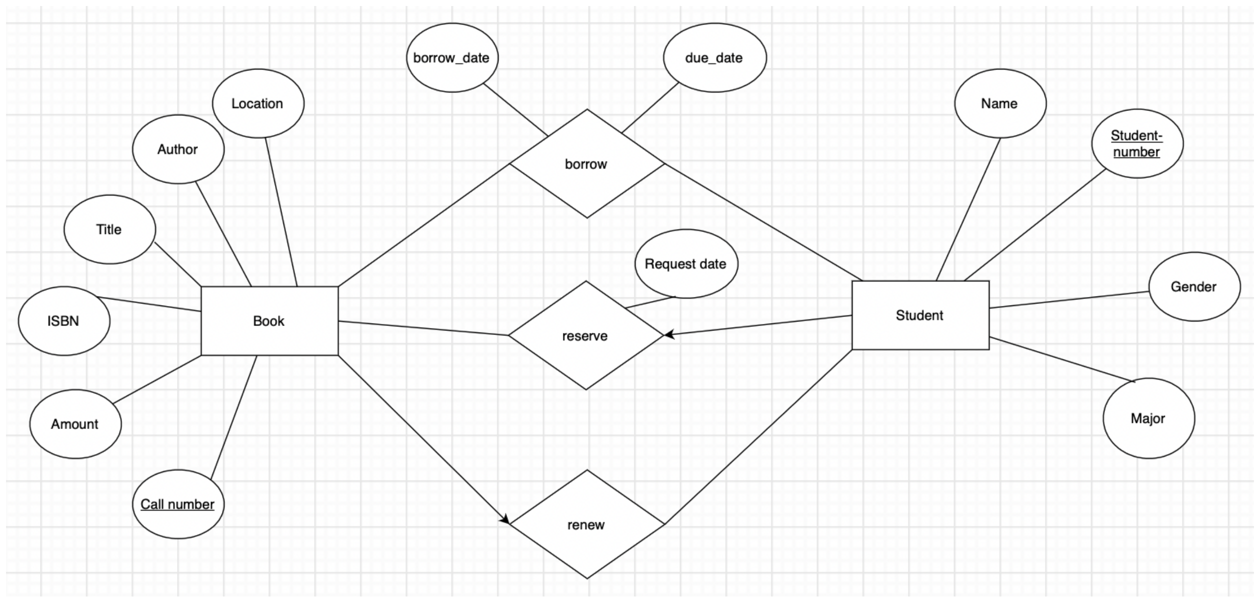
Database for an University Library

Group 20

Chan Cheuk Him 20202326
Chan Chi Hin Jonathan 20202288
Wong Tin Yau 21219443

Table of Contents

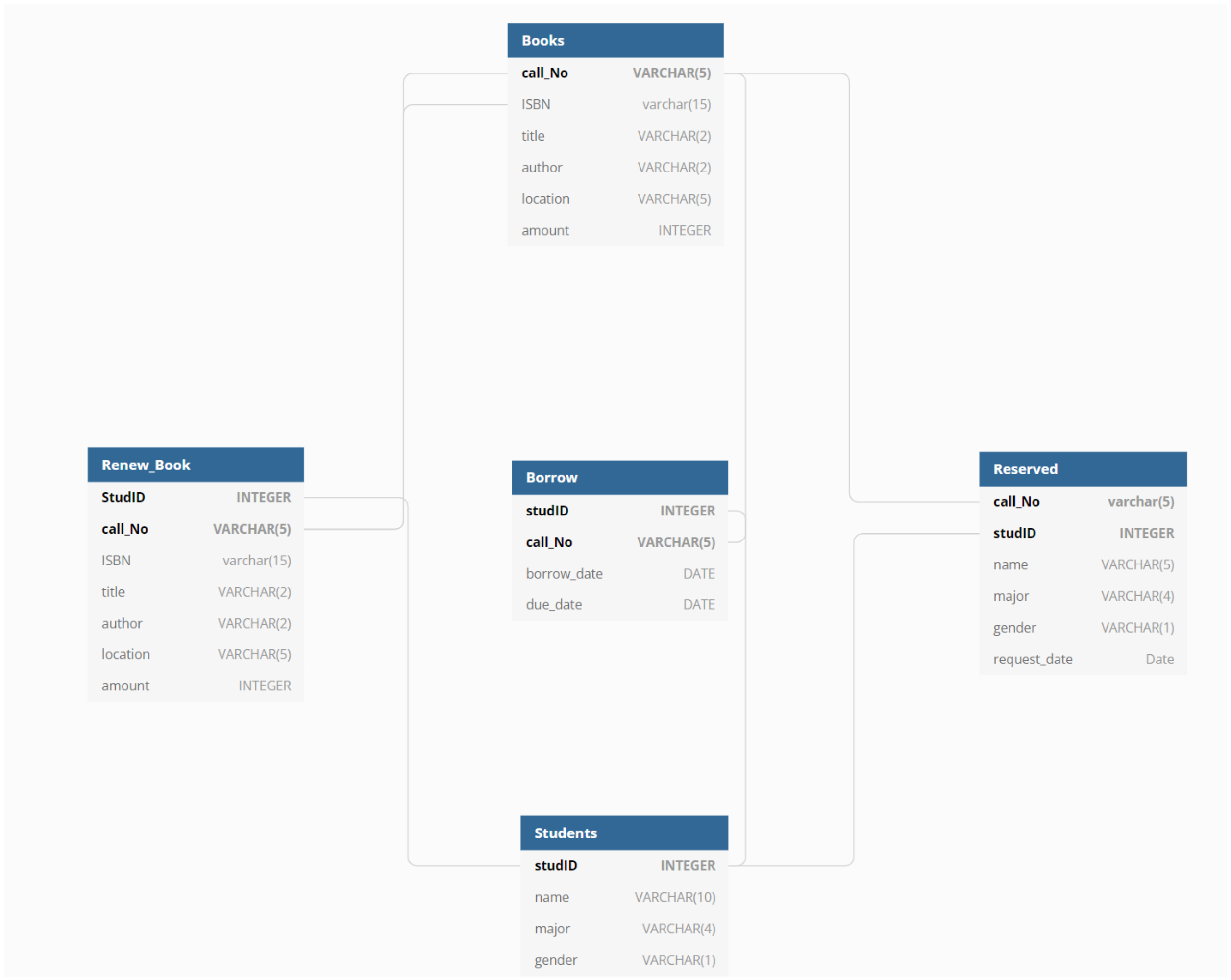
Entity Relationship Diagram



- **Explanation of the ER Design**:
  We have 2 entities(Student and Book) and 3 relationships (borrow, reserve and renew). We regard the call number and student number as the primary keys of Book entity and Student entity respectively.

  The cardinality constraints of the relationships are: Many-to-Many for borrow (A student can borrow/return many books while a book can be borrowed/returned by many students with its amount) , Many-to-One for reserve (A student can reserve at most one book while a book can be reserved by many students) , One-to-Many for renew (A student can renew many books while A book can only renewed by one particular student).

Table Schemes



Normalization

- **Normal Forms**
  For table Books, It is in 1st Normal Form since it does not include any multivalued attributes. Every attribute value is atomic. It is also in 2nd Normal Form since every non-key attribute is fully functionally dependent on the primary key(call_No). There are no partial dependencies. It is in 3rd Normal Form since Books does not have any transitive dependencies.

  For table Students, It is in 1st Normal Form since it does not include any multivalued attributes. Every attribute value is atomic.
  It is in 2nd Normal Form since every non-key attribute is fully functionally dependent on the primary key(studID). There are no partial dependencies. It is in 3rd Normal Form since Students do not have any transitive dependencies.

For table Borrow, It is not in 1st Normal Form since it could have multivalued attributes on call_No and studID.

For table Renew_Book, It is not in 1st Normal Form since it could have multivalued attributes on call_No or studID.

For table Reserved, It is not in 1st Normal Form since it could have multivalued attributes on call_No.

- **Normalization applied**
  In our model, we didn’t apply any Normalization on our design.

Source Code of the Java Program

```
import java.awt.GridLayout;
import java.awt.TextField;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;
import javax.swing.*;
import java.util.Properties;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.JSchException;
import com.jcraft.jsch.Session;

/**
* This is a University Library manager to support: (1) Search a book (2) Borrow a book
(3) Return a book (4) Renew a book (5) Reserve a book
*
* @author  Chan Cheuk Him  20202326 || Wong Tin Yau 21219443 || Chan Chi Hin Jonathan
20202288

*/
public class PJ {

   Scanner in = null;
   Connection conn = null;
   // Database Host
   final String databaseHost = "orasrv1.comp.hkbu.edu.hk";
   // Database Port
   final int databasePort = 1521;
   // Database name
   final String database = "pdborcl.orasrv1.comp.hkbu.edu.hk";
   final String proxyHost = "faith.comp.hkbu.edu.hk";
   final int proxyPort = 22;
   final String forwardHost = "localhost";
   int forwardPort;
   Session proxySession = null;
   boolean noException = true;
   // JDBC connecting host
   String jdbcHost;
   // JDBC connecting port
   int jdbcPort;

   String[] options = { // if you want to add an option, append to the end of
                        // this array
           "search a book", "borrow a book", "return a book",
           "renew a book", "reserve a book","exit" };


/**
    * Get YES or NO. Do not change this function.
    *
    * @return boolean
    */
     boolean getYESorNO(String message) {
       JPanel panel = new JPanel();
       panel.add(new JLabel(message));
       JOptionPane pane = new JOptionPane(panel, JOptionPane.QUESTION_MESSAGE,
JOptionPane.YES_NO_OPTION);
       JDialog dialog = pane.createDialog(null, "Question");
       dialog.setVisible(true);
       boolean result = JOptionPane.YES_OPTION == (int) pane.getValue();
       dialog.dispose();
       return result;
   }

   /**
    * Get username & password. Do not change this function.
    *
    * @return username & password
    */
   String[] getUsernamePassword(String title) {
       JPanel panel = new JPanel();
       final TextField usernameField = new TextField();
       final JPasswordField passwordField = new JPasswordField();
       panel.setLayout(new GridLayout(2, 2));
       panel.add(new JLabel("Username"));
       panel.add(usernameField);
       panel.add(new JLabel("Password"));
       panel.add(passwordField);
       JOptionPane pane = new JOptionPane(panel, JOptionPane.QUESTION_MESSAGE,
JOptionPane.OK_CANCEL_OPTION) {
           private static final long serialVersionUID = 1L;

           @Override
           public void selectInitialValue() {
               usernameField.requestFocusInWindow();
           }
       };
       JDialog dialog = pane.createDialog(null, title);
       dialog.setVisible(true);
       dialog.dispose();
       return new String[] { usernameField.getText(), new
String(passwordField.getPassword()) };
   }

   /**
    * Login the proxy. Do not change this function.
    *
    * @return boolean
    */
   public boolean loginProxy() {
       if (getYESorNO("Using ssh tunnel or not?")) { // if using ssh tunnel
           String[] namePwd = getUsernamePassword("Login cs lab computer");
           String sshUser = namePwd[0];
           String sshPwd = namePwd[1];
           try {
               proxySession = new JSch().getSession(sshUser, proxyHost, proxyPort);
               proxySession.setPassword(sshPwd);
               Properties config = new Properties();
               config.put("StrictHostKeyChecking", "no");
               proxySession.setConfig(config);
               proxySession.connect();
               proxySession.setPortForwardingL(forwardHost, 0, databaseHost,
databasePort);
               forwardPort =
Integer.parseInt(proxySession.getPortForwardingL()[0].split(":")[0]);
           } catch (JSchException e) {
               e.printStackTrace();
               return false;
           }
```

```
            jdbcHost = forwardHost;
            jdbcPort = forwardPort;
        } else {
            jdbcHost = databaseHost;
            jdbcPort = databasePort;
        }
        return true;
    }

    /**
     * Login the oracle system. Change this function under instruction.
     *
     * @return boolean
     */
    public boolean loginDB() {
        String username = "f0202326";//Replace e1234567 to your username
        String password = "f0202326";//Replace e1234567 to your password

        /* Do not change the code below */
        if(username.equalsIgnoreCase("e1234567") ||
password.equalsIgnoreCase("e1234567")) {
            String[] namePwd = getUsernamePassword("Login sqlplus");
            username = namePwd[0];
            password = namePwd[1];
        }
        String URL = "jdbc:oracle:thin:@" + jdbcHost + ":" + jdbcPort + "/" + database;

        try {
            System.out.println("Logging " + URL + " ...");
            conn = DriverManager.getConnection(URL, username, password);
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    /**
     * Show the options. If you want to add one more option, put into the
     * options array above.
     */
    public void showOptions() {
```

```
    System.out.println("Please choose following option:");
    for (int i = 0; i < options.length; ++i) {
        System.out.println("(" + (i + 1) + ") " + options[i]);
    }
}

/**
 * Run the manager
 */
public void run() {
    while (noException) {
        showOptions();
        String line = in.nextLine();
        if (line.equalsIgnoreCase("exit"))
            return;
        int choice = -1;
        try {
            choice = Integer.parseInt(line);
        } catch (Exception e) {
            System.out.println("This option is not available");
            continue;
        }
        if (!(choice >= 1 && choice <= options.length)) {
            System.out.println("This option is not available");
            continue;
        }
        if (options[choice - 1].equals("search a book")) {
            printBookByNo();
        } else if (options[choice - 1].equals("borrow a book")) {
            borrow();
        } else if (options[choice - 1].equals("return a book")) {
            Return();
        } else if (options[choice - 1].equals("renew a book")) {
            renew();
        } else if (options[choice - 1].equals("reserve a book")) {
            reserve();
        } else if (options[choice - 1].equals("exit")) {
            break;
        }
    }
}
```

```
   /**
    * Print out the infomation of a flight given a flight_no
    *
    */
   private void printBookInfo(String ISBN) {
       try {
           Statement stm = conn.createStatement();
           String sql = "SELECT * FROM Books WHERE ISBN = '" + ISBN + "'";
           ResultSet rs = stm.executeQuery(sql);
           String checkAmount ="0";
           if (!rs.next())
               return;

            // Check if the book amount = 0.
           if(rs.getString(6).equals(checkAmount)) {
               System.out.println("Sorry, The book is not available now.");
               return;
           }

           // if amount != 0 , Show details.
           String[] books = { "Call-number", "ISBN", "Title", "Author", "Location",
"Amount" };
           for (int i = 0; i < 6; ++i) { // flight table 6 attributes
               try {
                   System.out.println(books[i] + " : " + rs.getString(i + 1)); //
attribute
                                                                                // id
                                                                                //
starts
                                                                                // with
                                                                        // 1
               } catch (SQLException e) {
                   e.printStackTrace();
               }
           }
       } catch (SQLException e1) {
           e1.printStackTrace();
           noException = false;
       }
   }

   /**
```

```
  * List all Books ISBN in the database.
  */
 private void listAllBooks() {
     System.out.println("All Books in the database now:");
     try {
         Statement stm = conn.createStatement();
         String sql = "SELECT ISBN FROM books";
         ResultSet rs = stm.executeQuery(sql);

         int resultCount = 0;
         while (rs.next()) {
             System.out.println(rs.getString(1));
             ++resultCount;
         }
         System.out.println("Total " + resultCount + " book(s).");
         rs.close();
         stm.close();
     } catch (SQLException e) {
         e.printStackTrace();
         noException = false;
     }
 }

 /**
  * Select out a Book according to the ISBN.
  */
 private void printBookByNo() {
     listAllBooks();
     System.out.println("Please input the ISBN to print info:");
     String line = in.nextLine();
     line = line.trim();
     if (line.equalsIgnoreCase("exit"))
         return;

     printBookInfo(line);
 }

 private void borrow() {
     try {
         Statement stm = conn.createStatement();
         System.out.println("Please input student ID");
         String studentID= in.nextLine();
```

```
        System.out.println("Please input ISBM");
        String ISBN= in.nextLine();
        String sql = "SELECT * FROM Books WHERE ISBN = '" + ISBN + "'";
        ResultSet rs = stm.executeQuery(sql);
        int checkAmount=0;
        if(!rs.next()){
            return;
        }
        if(rs.getInt(6)<=checkAmount) {
            System.out.println("The book is not available now.");
            return;
        }
        rs.close();

        String sql2 = "(SELECT COUNT(*) FROM Borrow WHERE StudID =
'"+studentID+"')";
        ResultSet rs2 = stm.executeQuery(sql2);
        if(!rs2.next()){
            return;
        }
        if(rs2.getInt(1)>2){
            System.out.println("The amount of books haven't returned yet is more
than 2");
            return;
        }
        rs2.close();



        String sql3="SELECT COUNT(*) FROM Borrow WHERE StudID = '"+studentID+"' AND
(to_date('22-Apr-22', 'DD-MM-YY') - borrow_date) > 28";
        ResultSet rs3 = stm.executeQuery(sql3);
        if(!rs3.next()){
            return;
        }
        if(rs3.getInt(1)>0){
            System.out.println("There are"+rs3.getInt(1)+" books borrowed is
overdue.");
            return;
        }
        rs3.close();
```

```
            String sql5="SELECT call_No FROM BOOKS WHERE ISBN='"+ISBN+"'";
            ResultSet rs5 = stm.executeQuery(sql5);
            if(!rs5.next()){
                return;
            }
            String call_No = rs5.getString(1);
            rs5.close();

            String sql5a = "SELECT COUNT(*) FROM Borrow WHERE StudID = '"+studentID+ "'
AND call_No = '" + call_No +"'";
            ResultSet rs5a = stm.executeQuery(sql5a);
            if(!rs5a.next()){
                return;
            }
            if(rs5a.getInt(1)>0){
                System.out.println("You have already borrow this book. ");
                return;
            }
            rs5a.close();



            String sql6 = " SELECT R1.studID FROM Reserved R1, Reserved R2 WHERE
R1.call_No = '" + call_No + "' AND R1.request_date < R2.request_date" ;
            ResultSet rs6 = stm.executeQuery(sql6);
            if(!rs6.next()){
                String sql8="Insert into Borrow Values( " +studentID +" ,'" +
call_No+"','22-Apr-22','20-May-22')";
                stm.executeQuery(sql8);

                System.out.println("You have successfully borrowed the book: ") ;
                System.out.println("========================================");
                printBookInfo(ISBN);
                return;
            }else if((rs6.getInt(1) != Integer.parseInt(studentID))) {
                System.out.println("Someone is reserved before.");
                return;
            }
            else if (rs6.getInt(1) == Integer.parseInt(studentID)){
                String sql7 = "DELETE FROM Reserved Where studID = " + studentID ;
                stm.executeQuery(sql7);
```

```
               String sql8="Insert into Borrow Values( "+studentID+" ,'" +
call_No+"','22-Apr-22','20-May-22')";
               stm.executeQuery(sql8);

               System.out.println("You have successfully borrowed the book: ") ;
               System.out.println("========================================");
               printBookInfo(ISBN);

           }
           rs6.close();
           stm.close();

       } catch (SQLException e) {
           e.printStackTrace();
           noException = false;
       }
   }



   private void Return(){
       try {
           Statement stm = conn.createStatement();
           System.out.println("Please input student ID");
           String studentID= in.nextLine();
           String sql = "SELECT * FROM Borrow WHERE studID= " + studentID ;
           ResultSet rs = stm.executeQuery(sql);
           if(!rs.next()){
               return;
           }
           System.out.println("StudentID \t call-number \t Borrow Date \t\t\t\t Due
Date");

System.out.println("------------------------------------------------------------------
------");
           do{
               System.out.print(rs.getString(1)+ " \t ");
               System.out.print(rs.getString(2)+" \t\t\t");
               System.out.print(rs.getString(3)+"\t\t");
               System.out.print(rs.getString(4)+"\t");
               System.out.println();
           }while (rs.next());
           System.out.println("Please input book's Call-Number to return.");
           String call_No=in.nextLine();
           String sql2="DELETE FROM BORROW WHERE call_No='"+call_No+"' AND
studID="+studentID;
           stm.executeQuery(sql2);

           rs.close();
           stm.close();
           System.out.println("You have successfully returned this book!!");

       } catch (SQLException e) {
           e.printStackTrace();
           noException = false;
       }
   }

   private void renew() {
       try {
           Statement stm = conn.createStatement();
           System.out.println("Please input student ID");
           String studentID= in.nextLine();
           String sql = "SELECT * FROM Borrow WHERE studID= " + studentID ;
           ResultSet rs = stm.executeQuery(sql);
           if(!rs.next()){
               System.out.println("You did not borrow any book.");
               return;
           }
           System.out.println("StudentID \t call-number \t Borrow Date \t\t\t\t Due
Date");

System.out.println("-------------------------------------------------------------------
------");
           do {

               System.out.print(rs.getString(1)+" \t ");
               System.out.print(rs.getString(2)+" \t\t\t");
               System.out.print(rs.getString(3)+"\t\t");
               System.out.print(rs.getString(4)+"\t");
               System.out.println();

           }while(rs.next());
```

```
            String sql2 ="SELECT COUNT(*) FROM Borrow WHERE StudID = '" +studentID+ "'
AND (to_date('22-Apr-22', 'DD-MM-YY') - borrow_date) > 28";
            ResultSet rs2 = stm.executeQuery(sql2);
            if(!rs2.next()){
                return;
            }
            if(rs2.getInt(1)>0){
                System.out.println("There are "+ rs2.getInt(1) +" books borrowed is
overdue.");
                return;
            }
            rs2.close();

            System.out.println("Please input the call-number of the books you want to
renew");
            String call_No=in.nextLine();

            String sql3= "SELECT COUNT(*) FROM Renew_Book WHERE call_No = " + "'" +
call_No + "' AND studID = " + studentID ;
            ResultSet rs3 = stm.executeQuery(sql3);
            if(!rs3.next()){
                return;
            }
            if(rs3.getInt(1)>0){
                System.out.println("This book had been renewed before.");
                return;
            }
            rs3.close();


            String sql4= "SELECT COUNT(*) FROM Borrow WHERE call_No = " + "'" +call_No
+"' AND studID = " + studentID +  " AND (due_date - to_date('22-Apr-22', 'DD-MM-YY'))
> 14";
            ResultSet rs4 = stm.executeQuery(sql4);
            if(!rs4.next()){
                return;
            }
            if(rs4.getInt(1)>0){
                System.out.println("This book is only allowed to renew only during the
2nd half of its borrow period.");
                return;
            }
```

```
           rs4.close();

           String sql5 = " SELECT COUNT(*) FROM Reserved WHERE call_No = '" + call_No
+ "'";
           ResultSet rs5 = stm.executeQuery(sql5);
           if(!rs5.next()){
               return;
           }
           if(rs5.getInt(1) >0) {
               System.out.println("This book has been reserved.");
               return;
           }
           rs5.close();


           String sql6 = " Update borrow  Set due_date = " + "(select due_date from
borrow where call_No = '" +call_No + "' AND studID = " + studentID + ") +  INTERVAL
'14' DAY Where call_No = '" +  call_No + "' AND studID = " +studentID ;
           stm.executeQuery(sql6);

           String sql7 = "Insert into renew_book (studID ,call_no ) VALUES ( " +
studentID + ",'"  + call_No+ "')";
           stm.executeQuery(sql7);
           stm.close();
           System.out.println("You have successfully renewed this book.");



       } catch (SQLException e) {
           e.printStackTrace();
           noException = false;
       }
   }

   private void reserve() {
       //amount of book > 0
       //Not borrowed by this student
       //this student don't have another reservation
       try {
       Statement stm = conn.createStatement();
       System.out.println("Please input student ID");
       String studentID= in.nextLine();
```

```
        System.out.println("Please input call- number ");
        String call_No= in.nextLine();


        String sql = "SELECT * FROM Books WHERE call_No = '" + call_No + "'";
        ResultSet rs = stm.executeQuery(sql);

        //First Checking
        int checkAmount=0;
        if(!rs.next()){
            return;
        }
        if(rs.getInt(6)>checkAmount) {
            System.out.println("The book is now available. No reservation is
required.");
            return;
        }
        rs.close();

        //Second Checking
        String sql2 = "SELECT count(*) FROM Borrow WHERE StudID = '"+studentID+ "' AND
call_No= '" + call_No + "'";
        ResultSet rs2 = stm.executeQuery(sql2);
        if(!rs2.next()){
            return;
        }
        if(rs2.getInt(1) >0){
            System.out.println("This book is already borrowed by you.");
            return;
        }
        rs2.close();

        //Third checking
        String sql3="SELECT COUNT(*) FROM reserved WHERE StudID = '"+studentID+"'";
        ResultSet rs3 = stm.executeQuery(sql3);
        if(!rs3.next()){
            return;
        }
        if(rs3.getInt(1)>0){
            System.out.println("Multiple reservations are not allowed");
            return;
        }
        rs3.close();

        //Make reservation
        String sqlsp1="select * from students where StudID="+studentID;
        //String sqlsp2="select * from books where ISBN="+ISBN;

        //Select all attributes from students table
        ResultSet SPrs1 = stm.executeQuery(sqlsp1);
        if(!SPrs1.next())
            return;
        String[] heads = { "Student ID", "Student Name", "Department", "Gender"};
        for (int i = 0; i < 4; ++i) {  // Students table have 4 attributes
                System.out.println(heads[i] + " : " + SPrs1.getString(i + 1));
                heads[i] = SPrs1.getString(i+1);
        }




        System.out.println("Call No :"+call_No);


System.out.println(call_No+"','"+heads[0]+"','"+heads[1]+"','"+heads[2]+"','"+heads[3]
+"'");

        String sql4 = "insert into reserved
values('"+call_No+"',"+heads[0]+",'"+heads[1]+"','"+heads[2]+"','"+heads[3]+"','22-APR
-22')";
        ResultSet rs4 = stm.executeQuery(sql4);
        if(!rs4.next()) {
            System.out.println("Failed");
            return;
        }else
            System.out.println("Succeed to reserve!");

        rs4.close();
        stm.close();

        }catch (SQLException e) {
            e.printStackTrace();
            noException = false;
        }
    }
```

```java
    /**
     * Close the manager. Do not change this function.
     */
    public void close() {
        System.out.println("Thanks for using this manager! Bye...");
        try {
            if (conn != null)
                conn.close();
            if (proxySession != null) {
                proxySession.disconnect();
            }
            in.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    /**
     * Constructor of flight manager Do not change this function.
     */
    public PJ() {
        System.out.println("Welcome to use this manager!");
        in = new Scanner(System.in);
    }

    /**
     * Main function
     *
     * @param args
     */
    public static void main(String[] args) {
        PJ manager = new PJ();
        if (!manager.loginProxy()) {
            System.out.println("Login proxy failed, please re-examine your username and
password!");
            return;
        }
        if (!manager.loginDB()) {
            System.out.println("Login database failed, please re-examine your username
and password!");
            return;
        }
        System.out.println("Login succeed!");
        try {
            manager.run();
        } finally {
            manager.close();
        }
    }
}
```

Source Code for SQL Commands and Triggers

Table Creation :

```
PROMPT DROP TABLES;

DROP TABLE Students CASCADE CONSTRAINT;
DROP TABLE Renew_Book CASCADE CONSTRAINT;
DROP TABLE Reserved CASCADE CONSTRAINT;
DROP TABLE Borrow CASCADE CONSTRAINT;
DROP TABLE Books CASCADE CONSTRAINT;

CREATE TABLE Students (
   studID INTEGER,
   name VARCHAR(10),
   major VARCHAR(4),
   gender VARCHAR(1),
   PRIMARY KEY(studID)
)

CREATE TABLE Books(
   call_No VARCHAR(5),
   ISBN varchar(15),
   title VARCHAR(2),
   author VARCHAR(2),
   location VARCHAR(5),
   amount INTEGER,
   PRIMARY KEY (call_No)
)

CREATE TABLE Renew_Book (
   StudID INTEGER,
   call_No VARCHAR(5),
   ISBN varchar(15),
   title VARCHAR(2),
   author VARCHAR(2),
   location VARCHAR(5),
   amount INTEGER,
   PRIMARY KEY(call_No)
)

CREATE TABLE Reserved(
   call_No varchar(5),
   studID INTEGER,
   name VARCHAR(5),
   major VARCHAR(4),
   gender VARCHAR(1),
   request_date Date,
   PRIMARY KEY (studID)
)

CREATE TABLE Borrow(
   studID INTEGER,
   call_No VARCHAR(5),
   borrow_date  DATE,
   due_date DATE,
   PRIMARY KEY (studID, call_No)
)

ALTER TABLE Renew_Book DROP CONSTRAINT stud_CONST

ALTER TABLE Renew_Book ADD CONSTRAINT stud_CONST
FOREIGN KEY (StudID) REFERENCES Students(StudID);

ALTER TABLE Reserved ADD CONSTRAINT book_CONST
FOREIGN KEY (call_No) REFERENCES Books(call_No);

ALTER TABLE Borrow ADD CONSTRAINT stud_CONST1
FOREIGN KEY (StudID) REFERENCES Students(StudID);
ALTER TABLE Borrow ADD CONSTRAINT book_CONST1
```

```
FOREIGN KEY (call_No) REFERENCES Books(call_No);

COMMIT;
```

Record Insertion:

```
PROMPT INSERT BOOKS TABLE;

INSERT INTO BOOKS VALUES('A0000','0-306-40615-1','AA','XX','S1E01',0);
INSERT INTO BOOKS VALUES('B0000','0-306-40615-2','BB','YY','S2E02',0);
INSERT INTO BOOKS VALUES('C1111','0-306-40615-3','CC','ZZ','D1E11',2);
INSERT INTO BOOKS VALUES('B0001','0-306-40615-4','DD','UU','G1E00',2);
INSERT INTO BOOKS VALUES('A1111','0-306-40615-5','EE','VV','B1E00',2);
INSERT INTO BOOKS VALUES('D0101','0-306-40615-6','FF','WW','B2E11',1);
INSERT INTO BOOKS VALUES('E0000','0-306-40615-7','GG','PP','X0E22',0);
INSERT INTO BOOKS VALUES('E0100','0-306-40615-8','HH','QQ','X0E21',2);
INSERT INTO BOOKS VALUES('E0111','0-306-40615-9','II','RR','X0E44',0);

PROMPT INSERT Students TABLE;
INSERT INTO Students VALUES(12345678,'A','Comp','M');
INSERT INTO Students VALUES(11111111,'B','Math','M');
INSERT INTO Students VALUES(22222222,'C','Comm','F');
INSERT INTO Students VALUES(33333333,'D','Comm','F');
INSERT INTO Students VALUES(44444444,'E','Comp','M');
INSERT INTO Students VALUES(55555555,'F','Comm','M');
INSERT INTO Students VALUES(66666666,'G','Math','F');
INSERT INTO Students VALUES(77777777,'H','Comp','M');

PROMPT INSERT BORROW TABLE;
INSERT INTO BORROW VALUES(11111111,'D0101','24-MAR-2022','21-APR-2022');
INSERT INTO BORROW VALUES(55555555,'A1111','23-MAR-2022','20-APR-2022');
INSERT INTO BORROW VALUES(22222222,'B0000','31-MAR-2022','12-MAY-2022');
INSERT INTO BORROW VALUES(11111111,'A0000','1-APR-2022','29-APR-2022');
INSERT INTO BORROW VALUES(33333333,'A0000','3-APR-2022','1-MAY-2022');
INSERT INTO BORROW VALUES(11111111,'B0000','3-APR-2022','15-MAY-2022');
INSERT INTO BORROW VALUES(44444444,'C1111','4-APR-2022','16-MAY-2022');
INSERT INTO BORROW VALUES(44444444,'A0000','6-APR-2022','4-MAY-2022');

INSERT INTO BORROW VALUES(33333333,'C1111','6-APR-2022','4-MAY-2022');
INSERT INTO BORROW VALUES(33333333,'A1111','6-APR-2022','4-MAY-2022');
INSERT INTO BORROW VALUES(33333333,'B0001','6-APR-2022','4-MAY-2022');
INSERT INTO BORROW VALUES(44444444,'D0101','10-APR-2022','8-MAY-2022');
INSERT INTO BORROW VALUES(33333333,'D0101','10-APR-2022','8-MAY-2022');
INSERT INTO BORROW VALUES(44444444,'A1111','14-APR-2022','12-MAY-2022');
INSERT INTO BORROW VALUES(55555555,'C1111','18-APR-2022','16-MAY-2022');
INSERT INTO BORROW VALUES(22222222,'E0111','19-APR-2022','17-MAY-2022');
INSERT INTO BORROW VALUES(11111111,'E0000','20-APR-2022','18-MAY-2022');
INSERT INTO BORROW VALUES(44444444,'B0001','21-APR-2022','19-MAY-2022');

PROMPT INSERT RESERVED TABLE;
INSERT INTO Reserved (studID,call_No, request_date)
Values('12345678','A0000','20-APR-22');
INSERT INTO Reserved (studID,call_No, request_date)
Values('66666666','E0000','22-APR-22');

PROMPT INSERT Renew_book TABLE;
INSERT INTO Renew_book (studID,call_No) Values('22222222','B0000');
INSERT INTO Renew_book (studID,call_No) Values('11111111','B0000');
INSERT INTO Renew_book (studID,call_No) Values('12345678','C1111');
```

BORROW_BOOK Trigger:

```
-- Borrow Trigger (Decrement the amount attribute of table Books)
CREATE OR REPLACE TRIGGER BORROW_BOOK
AFTER INSERT OR UPDATE ON Borrow
FOR EACH ROW
DECLARE
   cnt INTEGER;
BEGIN
   SELECT Amount INTO cnt FROM Books
   WHERE call_No = :new.call_No;
   cnt := cnt - 1;
   UPDATE Books
   SET Amount = cnt
       WHERE call_No = :new.call_No;
END;
/
```

RETURN_BOOK Trigger:

```

-- Return Trigger (Increment the amount attribute of table Books)
CREATE OR REPLACE TRIGGER RETURN_BOOK
```

```
AFTER DELETE OR UPDATE ON Borrow
FOR EACH ROW
DECLARE
   cnt INTEGER;
BEGIN
   SELECT Amount INTO cnt FROM Books
   WHERE call_No = :old.call_No;
   cnt := cnt + 1;
   UPDATE Books
   SET Amount = cnt
       WHERE call_No = :old.call_No;
END;
/

COMMIT;
SET AUTOCOMMIT ON
```