VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018

A PROJECT REPORT
ON
SMART LIBRARY MANAGER

BY

RATHAN H V (4SF16CS124)

RAVIGANESH M (4SF16CS125)

In the partial fulfillment of the requirement for V Sem. B. E. (CSE)

DBMS LABORATORY WITH MINI PROJECT

Under the guidance of

Mrs. Sadhana
Asst.Prof, Dept. of CSE

Department of Computer Science & Engineering
SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
Adyar, Mangaluru-575007

2018-19

SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
(Affiliated to Visvesvaraya Technological University, BELAGAVI)
Adyar, Mangaluru – 575007

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**SMART LIBRARY MANAGER"** is submitted in partial fulfillment for the requirement of V sem. B. E. (Computer Science & Engineering), "DBMS LABORATORY WITH MINI PROJECT" during the year 2018 – 19 is a result of bonafide work carried out by

RATHAN H V	**4SF16CS124**
RAVIGANESH M	**4SF16CS125**

……
Mrs. Sadhana
Asst.Prof. Dept. of CS&E
SCEM, Mangaluru

……
Dr. J V Gorabal
HOD, Dept. of CS&E
SCEM, Mangaluru

Signature of the Examiners

1. ……..

2. …….

ACKNOWLEDGEMENT

With regards to the project " Smart Library Manager " we are grateful to all the hours of workmanship involved by our acquaintance's for developing a pleasant software for creating and maintaining a database with reference to library .This project was completed not only by a team of two ,but it included encouragement and guidance from invisible sources . Even though a project can be completed by a single mind , we are thankful to a large number of well-wishers.

We would like to express our deep gratitude to our beloved Head of Compter Science Department Dr. J V Gorbal, and our project guide Mrs . Sadhana ,

Asst.Prof , Department of Computer Science and engineering, without whom this project would not have been completed successfully . We owe a great debt to our beloved principal , Dr. R Srinivasa Rao Kunte and our beloved chairman sir, Dr . Manjunath Bhandary .

ABSTRACT

The project titled “Smart Library Manager ” is a Library management software for monitoring and controlling the transactions in a library .The project is developed using java as the front end, which mainly focuses on basic operations in a library such as adding new member, new books, and updating new information, searching books and members and facility to borrow and The project consists of a login facility for the administrator who is the only one allowed to enter and authorize data and transactions .Our software allows us to provide unique details to all the borrowers, it also provides the facility for efficient searching capabilities throughout the database .Administrator has the authority to view details regarding all the transactions that were initiated by borrowers . This software provides a convenient interface for the users . The data is foolproof and is well organized for easy analysis during retrieval .

INDEX

CHAPTER 1

INTRODUCTION

DBMS stands for Database Management System . DBMS is a collection of interrelated data and set of programs to store and access those data in an easy and effective manner. Database systems are basically developed for large amount of data , there are two things that require optimization: Storage of data and Retrieval of data . According to principles of database systems, the data is stored in such a way that it acquires lot less space as the redundant data has been removed before storage.

A major problem faced by library administrator is to keep track of borrower details, book details ,transaction details . This problem has inspired us to develop a Library Management System . It is very difficult to manually maintain borrower , book and transaction details. Hence we develop a java based application to solve this problem. This project can be used by the administrator to manage the borrower details and also manage the particulars of the borrowers. GUI is user friendly and is very easy to access and understand . It reduces lot of manual maintenance .We develop our Smart Library Manager using Oracle 10g and Java(NetBeansIDE).

1.1 About Oracle Database

Oracle Database commonly referred to as Oracle RDBMS is multi model database management system produced and marketed by Oracle Corporation. Originally developed in 1977 by Lawrence Ellison and other developers, Oracle DB is one of the most trusted and widely used relational database engines.

The system is built around a relational database framework in which data objects may be directly used by users (or an application front end) through structured query language(SQL). Oracle is a fully scalable relational database architecture and is often used by global enterprises, which manage and process data across wide and local area networks. The Oracle database has its own network component to allow communications across networks .

1.2 About JAVA

Java is a general purpose computer programming language that is concurrent, class-based, object oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers “write once, run anywhere” meaning that compiled java code can be run on all platforms that support java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture .

1.3 About NetBeans

NetBeans is an integrated development environment (IDE) for Java . NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Microsoft Windows , macOS , Linux and Solaris . In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML and Javascript .

CHAPTER 2

DESIGN

Design is the first step in the development of any engineering product or system. Design is the planning that lays the basics for making of every object or system. The design process involves developing several models of the system at different levels of abstraction. As a design is decomposed, errors and omission in the earlier stage are discovered. These feedbacks allow the earlier design of the model to be improved.

Software design is still a complicated process. Starting from a set of requirement, usually in natural language and informal design is prepared. Coding commences and the design is modified as the system is implemented. It is a model of a real world system that has many participating entities and relationships.

2.1 ER-Diagram

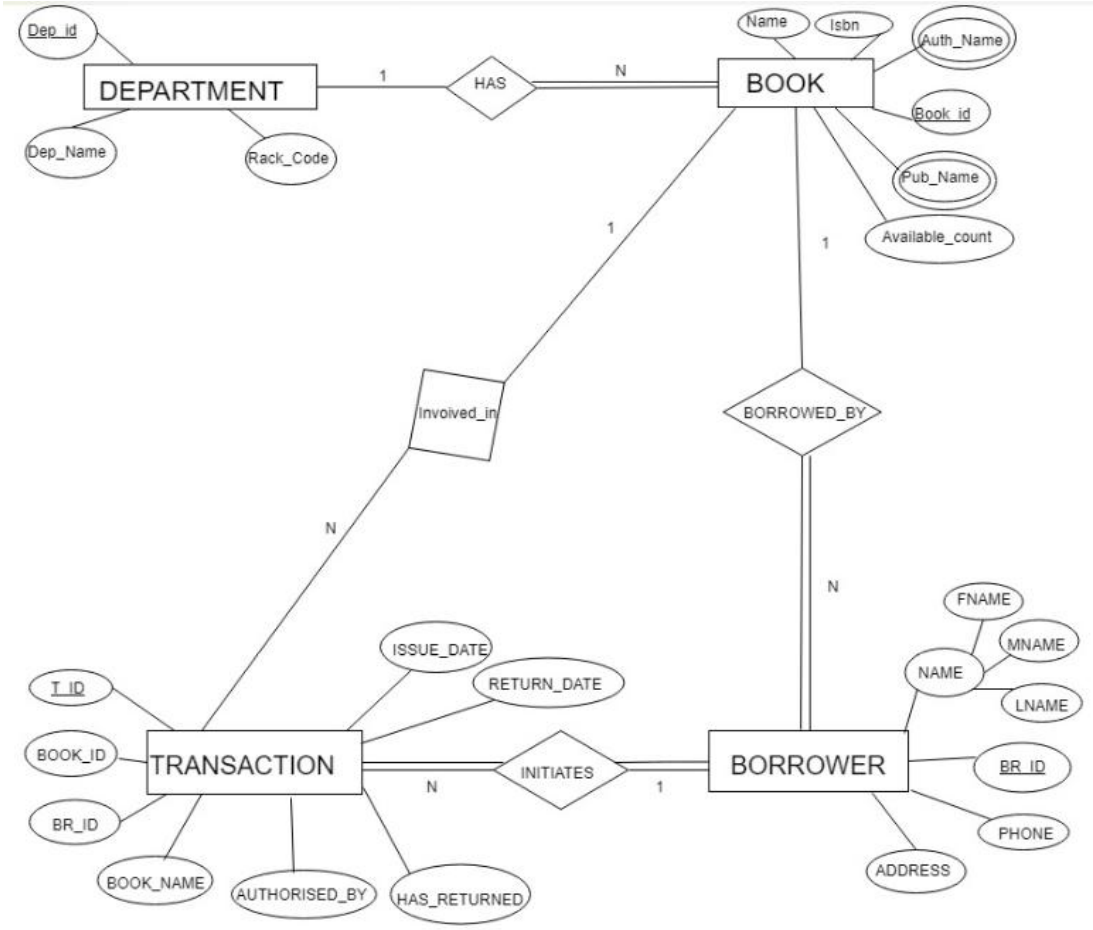


Figure 2.1 : ER-diagram

2.2 Relational Schema

STEP 1: MAPPING OF REGULAR ENTITY TYPES

DEPARTMENT

dep_id	dep_name	rack_code

BOOK

book_id	isbn	name	dep_id	available_count

BORROWER

br_id	fname	mname	lname	phone	address

TRANSACTION

t_id	br_id	book_id	authorised_by	has_returned	issue_date	return_date

AUTHOR

auth_name	book_id

PUBLISHER

pub_name	book_id

Figure: 2.2.1 Mapping of Regular Entity

STEP 2: MAPPING OF WEAK ENITY TYPES

In this step, we draw the schema for all weak entities and add primary key of owner entity as foreign key to the weak entity. There are no weak entities in this project and hence this step is skipped.

STEP 3: MAPPING OF BINARY 1:1 RELATION TYPES

In this step we map the 1:1 binary relationships and the primary key of partially participating entity is made as foreign key in the entity with total participation in this relation. Since we donot have any binary 1:1 relationships, we skip this step.

STEP 4: MAPPING OF 1: N RELATION TYPES

In this step, we map binary 1:N relationships and the primary key of the entity on 1 side is made as foreign key in the entity present on N side of relationship. In this project there is a 1:N relationship between DEPARTMENT and BOOK entities and between BORROWER and TRANSACTION . Hence we make DEP_ID as foreign key in BOOK entity and we make BR_ID as foreign key in TRANSACTION . Schema after step 4 is shown in figure 2.2.2.

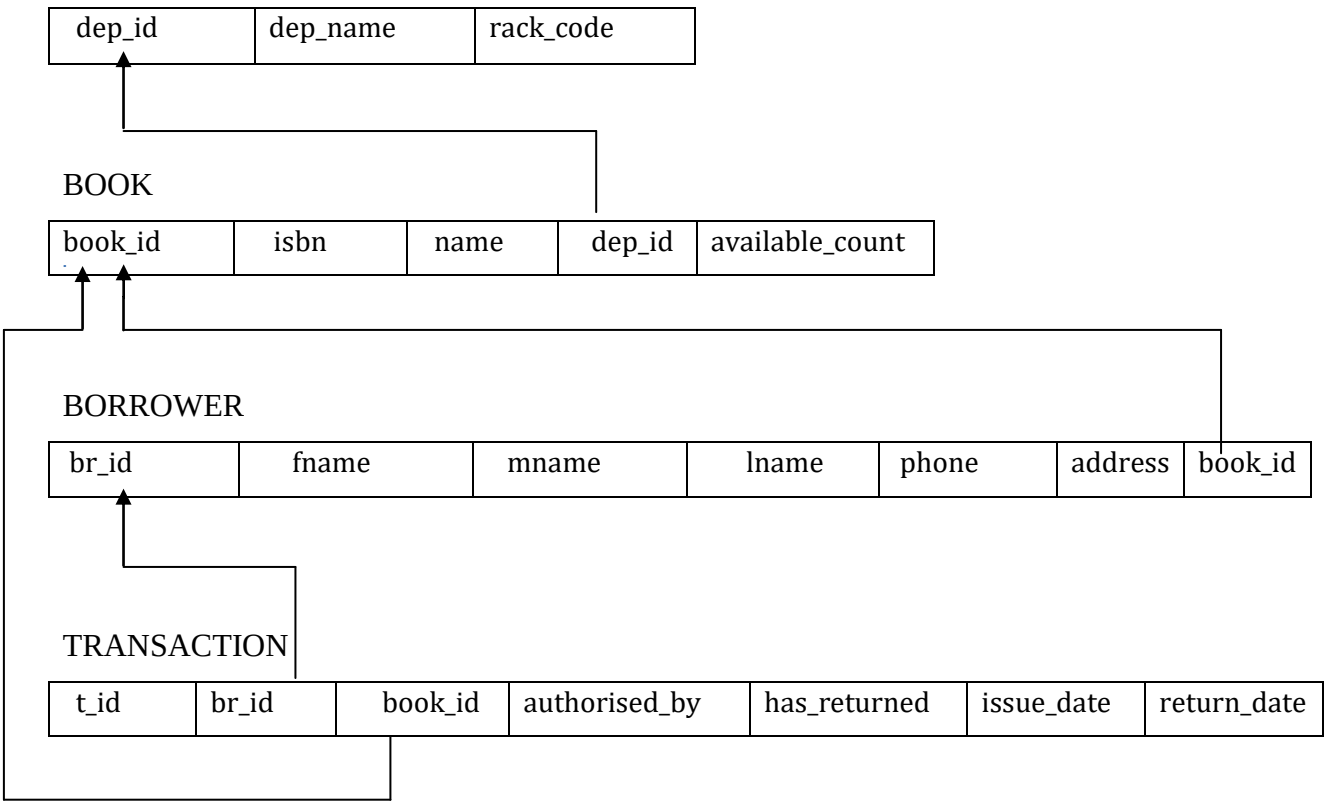


Figure: 2.2.2 : Mapping of 1:N Relation Type

STEP 5: MAPPING OF M: N RELATION TYPES

In this step, we map Binary M:N relationship. We create a new table and add the primary keys of the participating entities as foreign keys in this new table. Since we do not have any binary M:N relationships, we skip this step.

STEP 6: MAPPING OF MULTIVALUED ATTRIBUTES

In this step, we map the multivalued attributes. This is done by creating a new table and adding the multivalued attribute and making the entity to which the multivalued attribute belongs as foreign key in new table. This combination of foreign key and multivalued attribute is made as primary key in the new table . In this project , we the entity BOOK has a multivalued attribute auth_name and pub_name. Hence we create two new tables and add key attribute book_id as foreign key in respective tables. After Step 6 is shown in figure 2.2.3

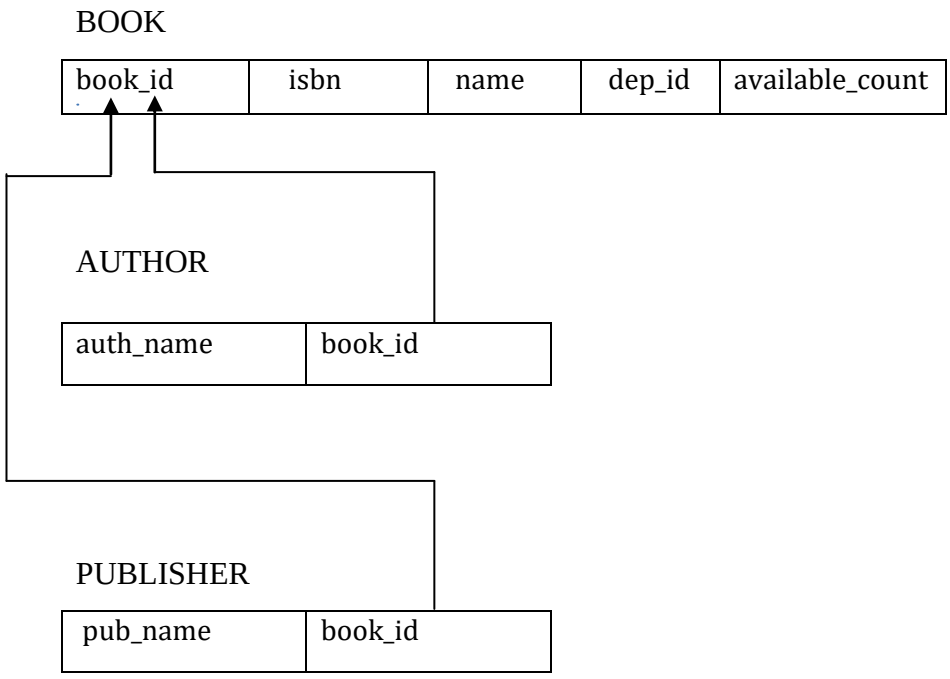


Figure 2.2.3 : Multivalued Attribute

STEP 7: MAPPING OF N-ARY RELATION TYPES

In this step, we map n-ary relationships for n > 2. Since there are no n-ary relationships in this project, we skip this step.

2.3 Schema diagram

The schema diagram of our project is shown in the figure 2.3

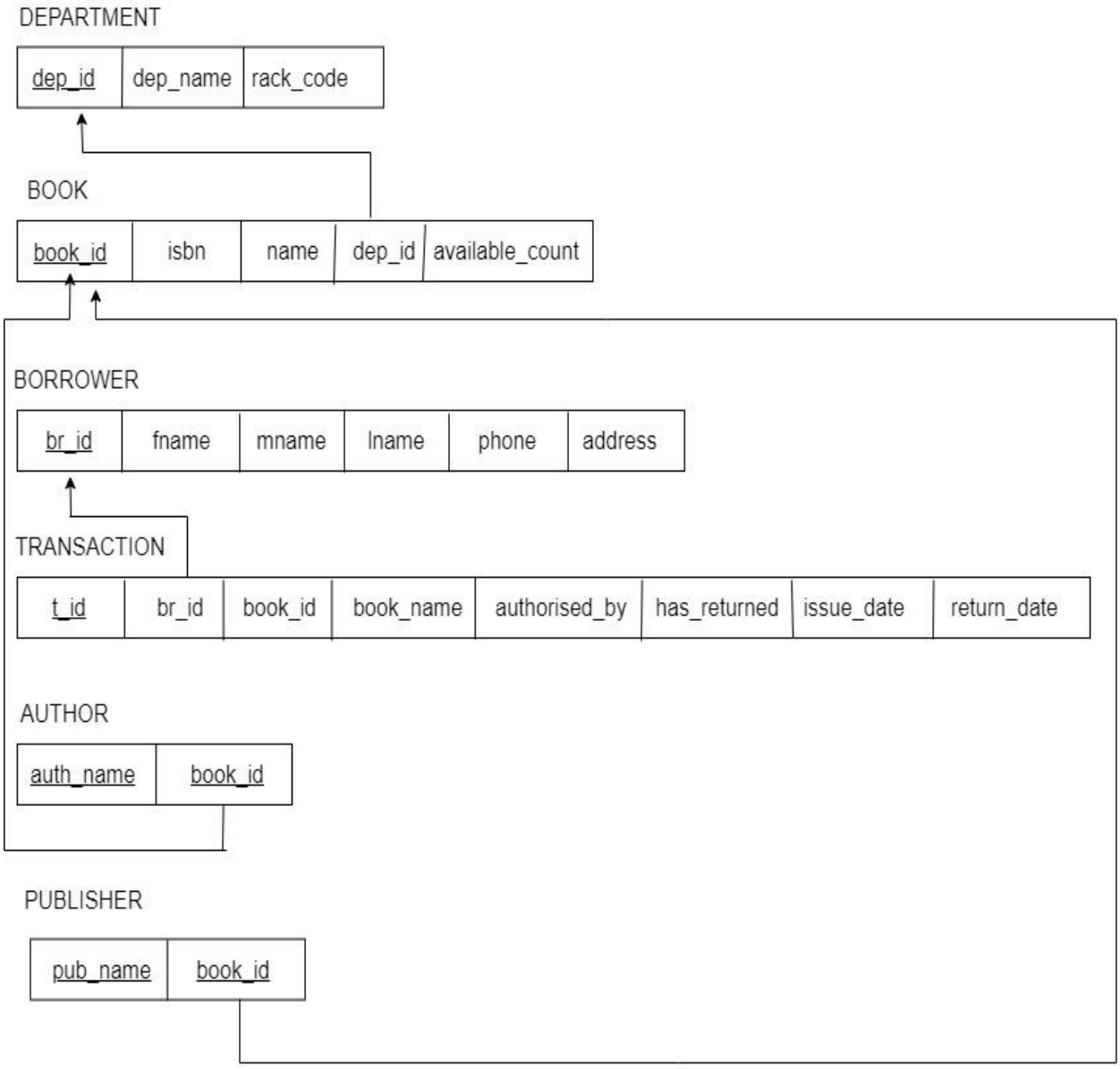
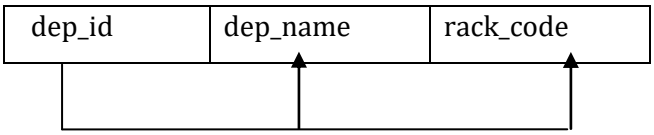


Figure: 2.3 : Schema Diagram of Library Management System

2.4 Functional Dependencies

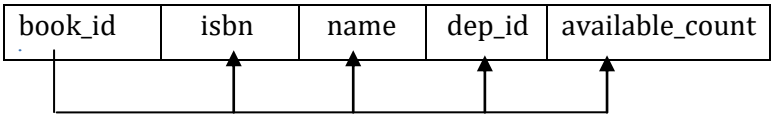
The functional dependencies are shown below in Figure 2.4.1

DEPARTMENT



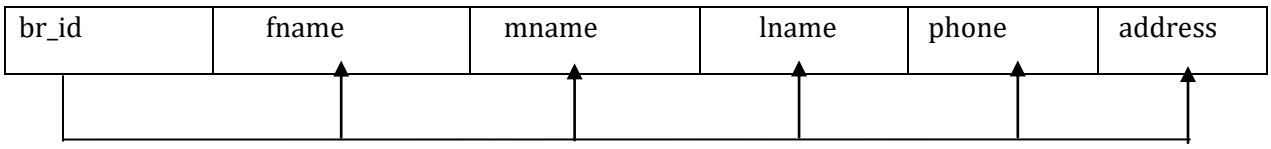
FD1: dep_id = {dep_name,rack_code }

BOOK



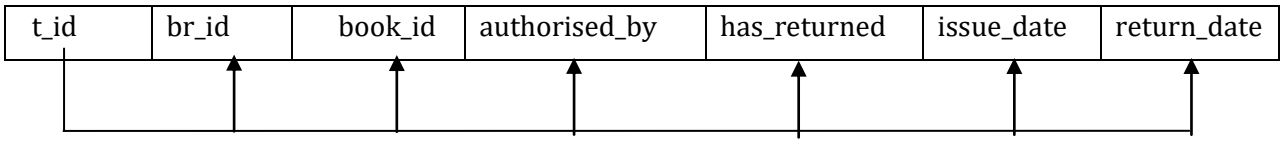
FD2: book_id = {isbn,name,dep_id,available_count }

BORROWER



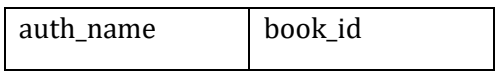
FD3: br_id= {fname, mname, lname,phone,address}

TRANSACTION



FD4: t_id = {br_id,book_id,authorized_by,has_returned,issue_date,return_date }

AUTHOR



PUBLISHER

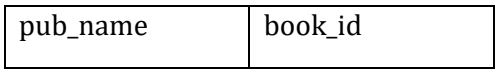


Figure: 2.4.1 : Functional Dependencies

2.5 Normalization

First Normal Form :

It states that the domain of an attribute must include only atomic values and the values of any attribute in a tuple must be single value from the domain of that attribute. 1NF only permits single atomic values. The First Normal form is shown above in figure 2.5.1. All the above relations are in 1NF because all the attributes are atomic .

Second Normal Form :

A relation schema is in 2NF if it is 1NF and every nonprime attribute A in R is fully functionally dependent on the Primary key of R. All the relations are already in 2NF because every non key attribute of the relation is fully functionally dependent on the primary key attribute.

Third Normal Form :

A relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key. All the relations are already in 3NF because there is no attribute which is transitively dependent on primary key attribute.

It is a normal form that is used in normalizing a database design to reduce the duplication of data and ensure referential integrity by ensuring that all the attributes in a table are determined only by the candidate keys of that relation and not by any non-prime attribute.

The schema after normalization is shown in Figure 2.5.1

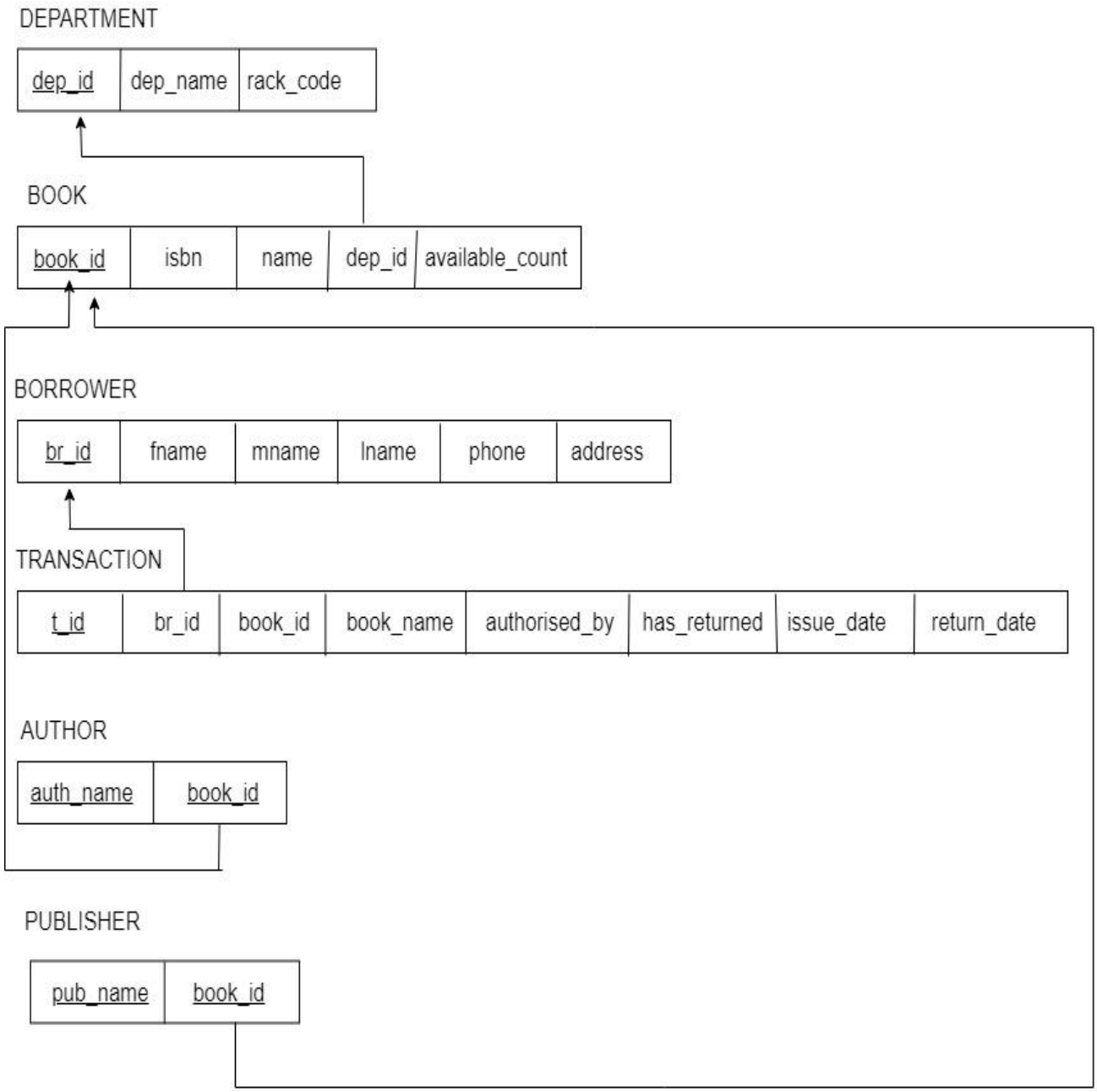


Figure: 2.5.1 : Normalized Schema

CHAPTER 3

Implementation

3.1 Requirements Specifications

Hardware specifications

Processor: Intel Core i5 or equivalent

Memory: 4GB DDR3 RAM

Disk space: 10GB

Keyboard: Standard

Mouse

Software specifications

Platform: Windows 8 or higher

Front end tool: Java Back

end tool: Oracle 10g

3.2 CREATING TABLE

Department Table

CREATE TABLE DEPARTMENT(DEP_ID VARCHAR2(20),CONSTRAINT DEP199ID PRIMARY KEY(DEP_ID),DEP_NAME VARCHAR2(20) NOT NULL,RACK_CODE VARCHAR2(10) NOT NULL);

NAME	TYPE
DEP_ID	VARCHAR2(20)
DEP_NAME	VARCHAR2(20)
RACK_CODE	VARCHAR2(20)

Table 3.2.1 : Department Table

Book Table

CREATE TABLE BOOK(BOOK_ID VARCHAR2(20),CONSTRAINT BOK99ID PRIMARY KEY(BOOK_ID),ISBN VARCHAR2(20) NOT NULL,NAME VARCHAR2(20) NOT NULL,DEP_ID VARCHAR2(20)AVAILABLE_COUNT NUMBER(38),CONSTRAINT DEP895ID FOREIGN KEY(DEP_ID) REFERENCES DEPARTMENT);

NAME	TYPE
BOOK_ID	VARCHAR2(20)
ISBN	VARCHAR2(20)
NAME	VARCHAR2(20)
DEP_ID	VARCHAR2(20)
AVAILABLE_COUNT	NUMBER(38)

Table 3.2.2 : Book Table

Borrower Table

CREATE TABLE BORROWER(BR_ID VARCHAR2(20),FNAME VARCHAR2(20) NOT NULL,MNAME VARCHAR2(20),LNAME VARCHAR2(20),PHONE NUMBER(38) NOT NULL,ADDRESS VARCHAR2(20),CONSTRAINT PK890 PRIMARY KEY(BR_ID));

NAME	TYPE
BR_ID	VARCHAR2(20)
FNAME	VARCHAR2(20)
MNAME	VARCHAR2(20)
LNAME	VARCHAR2(20)
PHONE	NUMBER(38)
ADDRESS	VARCHAR2(20)

Table 3.2.3 : Borrower Table

Transaction Table

CREATE TABLE TRANSACTION(T_ID VARCHAR2(20),CONSTRAINT PK4494 PRIMARY KEY(T_ID),BR_ID VARCHAR2(20)REFERENCES BORROWER,BOOK_ID VARCHAR2(20) REFERENCES BOOK,BOOK_NAME VARCHAR2(20),AUTHORISED_BY VARCHAR2(20) NOT NULL,HAS_RETURNED VARCHAR2(20),ISSUE_DATE TIMESTAMP,RETURN_DATE TIMESTAMP);

NAME	TYPE
T_ID	VARCHAR2(20)
BR_ID	VARCHAR2(20)
BOOK_ID	VARCHAR2(20)
BOOK_NAME	VARCHAR2(20)
AUTHORISED_BY	VARCHAR2(20)
HAS_RETURNED	VARCHAR2(20)
ISSUE_DATE	TIMESTAMP
RETURN_DATE	TIMESTAMP

Table 3.2.4 : Transaction Table

Author Table

CREATE TABLE AUTHOR(AUTH_NAME VARCHAR2(20),BOOK_ID VARCHAR2(20),CONSTRAINT AU99T FOREIGN KEY(BOOK_ID) REFERENCES BOOK ,CONSTRAINT AUTH299 PRIMARY KEY (AUTH_NAME,BOOK_ID));

NAME	TYPE
AUTH_NAME	VARCHAR2(20)
BOOK_ID	VARCHAR2(20)

Table 3.2.5 : Author Table

Publisher Table

CREATE TABLE PUBLISHER(PUB_NAME VARCHAR2(20),BOOK_ID VARCHAR2(20),CONSTRAINT AUTH399 FOREIGN KEY(BOOK_ID) REFERENCES BOOK,CONSTRAINT PUBLI99 PRIMARY KEY(PUB_NAME,BOOK_ID));

NAME	TYPE
PUB_NAME	VARCHAR2(20)
BOOK_ID	VARCHAR2(20)

Table 3.2.6 : Publisher Table

3.3 FUNCTIONALITY

3.3.1 Connecting to Database

```
void connectionDB(){
      try{
         String url="jdbc:oracle:thin:@localhost:1521:XE";
         con=DriverManager.getConnection(url,"system","raviganesh123");
      }
      catch(Exception e){
         System.out.println(e);
      }
   }
```

3.3.2 Inserting values into table

```
String AUTH_NAME=jTextField1.getText();
String BOOK_ID=jTextField2.getText();
connectionDB();
String sql="insert into AUTHOR(AUTH_NAME,BOOK_ID)values(?,?)";
ps=(OraclePreparedStatement) con.prepareStatement(sql);
ps.setString(1,AUTH_NAME);
ps.setString(2,BOOK_ID);
ps.executeQuery();
JOptionPane.showMessageDialog(null,"Inserted successfully");
jTextField1.setText("");
jTextField2.setText("");
con.close();
ps.close();
```

3.3.3 Deleting values from table

```
String  BOOK_ID = jTextField2.getText();
        connectionDB();
        String sql = "delete from AUTHOR where BOOK_ID=?";
        try{
            ps = (OraclePreparedStatement) con.prepareStatement(sql);
            ps.setString(1,BOOK_ID);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(null,"deleted successfully");
            jTextField1.setText("");
            jTextField2.setText("");
            con.close();
        }
         catch(Exception e){
            JOptionPane.showMessageDialog(null, "Error in Deletion");
            System.out.println(e);
        }
```

3.3.4 Updating the table values

```
String AUTH_NAME=jTextField1.getText();
      String BOOK_ID =jTextField2.getText();
      connectionDB();
      String sql="UPDATE AUTHOR set AUTH_NAME =?  , BOOK_ID=? WHERE
BOOK_ID=?";
      try{
         ps = (OraclePreparedStatement) con.prepareStatement(sql);
         ps.setString(1,AUTH_NAME);
         ps.setString(2,BOOK_ID);
         ps.setString(3,BOOK_ID);
         ps.executeUpdate();
         JOptionPane.showMessageDialog(null,"updated successfully");
         jTextField1.setText("");
         jTextField2.setText("");
         con.close();

      }
      catch(Exception e){
         System.out.println("error update:"+e);
      }
```

3.3.5 Trigger to Delete Transaction record when Borrower record is deleted

```
CREATE OR REPLACE TRIGGER TRG800  BEFORE DELETE ON BORROWER
FOR EACH ROW
BEGIN
 DELETE FROM TRANSACTION WHERE BR_ID=:OLD.BR_ID;
END;
```

3.3.6 Trigger to Delete Book record when Department record is deleted

```
CREATE OR REPLACE TRIGGER TRG900 BEFORE DELETE ON DEPARTMENT
FOR EACH ROW
BEGIN
DELETE FROM BOOK  WHERE DEP_ID=:OLD.DEP_ID;
END;
```

3.3.7 Trigger to Delete Author ,Publisher ,Transaction record when Book record is deleted

```
CREATE OR REPLACE TRIGGER TRG1091 BEFORE DELETE ON BOOK
FOR EACH ROW
BEGIN
DELETE FROM AUTHOR  WHERE BOOK_ID=:OLD.BOOK_ID;
DELETE FROM PUBLISHER  WHERE BOOK_ID=:OLD.BOOK_ID;
DELETE FROM TRANSACTION  WHERE BOOK_ID=:OLD.BOOK_ID;
END;
```

3.3.8 Trigger to Update available book count when book is borrowed by borrower

```
CREATE OR REPLACE TRIGGER TGRBOOK  BEFORE INSERT ON TRANSACTION
FOR EACH ROW
BEGIN
UPDATE BOOK  SET AVAILABLE_COUNT =AVAILABLE_COUNT-1
WHERE BOOK_ID=:NEW.BOOK_ID;
END;
```

3.3.9 Trigger to Update available book count when book is returned by borrower

```
CREATE OR REPLACE TRIGGER TGRBOOK1  BEFORE DELETE ON TRANSACTION
FOR EACH ROW
BEGIN
UPDATE BOOK  SET AVAILABLE_COUNT =AVAILABLE_COUNT+1
WHERE BOOK_ID=:OLD.BOOK_ID;
END;
```

3.3.10 Stored procedure to delete Transaction details

```
CREATE OR REPLACE PROCEDURE PROC1(TID IN VARCHAR2)
AS
BEGIN
DELETE FROM TRANSACTION  WHERE T_ID=TID;
END;
```

Stored procedure call for Transaction table

```
String  T_ID = jTextField1.getText();
      connectionDB();
      String sql = "{call proc1(' "+T_ID+" ')}";
      try
      {
         ps = (OraclePreparedStatement) con.prepareStatement(sql);
         ps.executeUpdate();
         JOptionPane.showMessageDialog(null,"deleted successfully");
         jTextField1.setText("");
         jTextField2.setText("");
         jTextField3.setText("");
         jTextField4.setText("");
         jTextField5.setText("");
         jTextField6.setText("");
         con.close();
      }
      catch(Exception e)
      {
         JOptionPane.showMessageDialog(null, "Error in Deletion");
         System.out.println(e);
      }
```

CHAPTER 4

Result

4.1 :Login Page

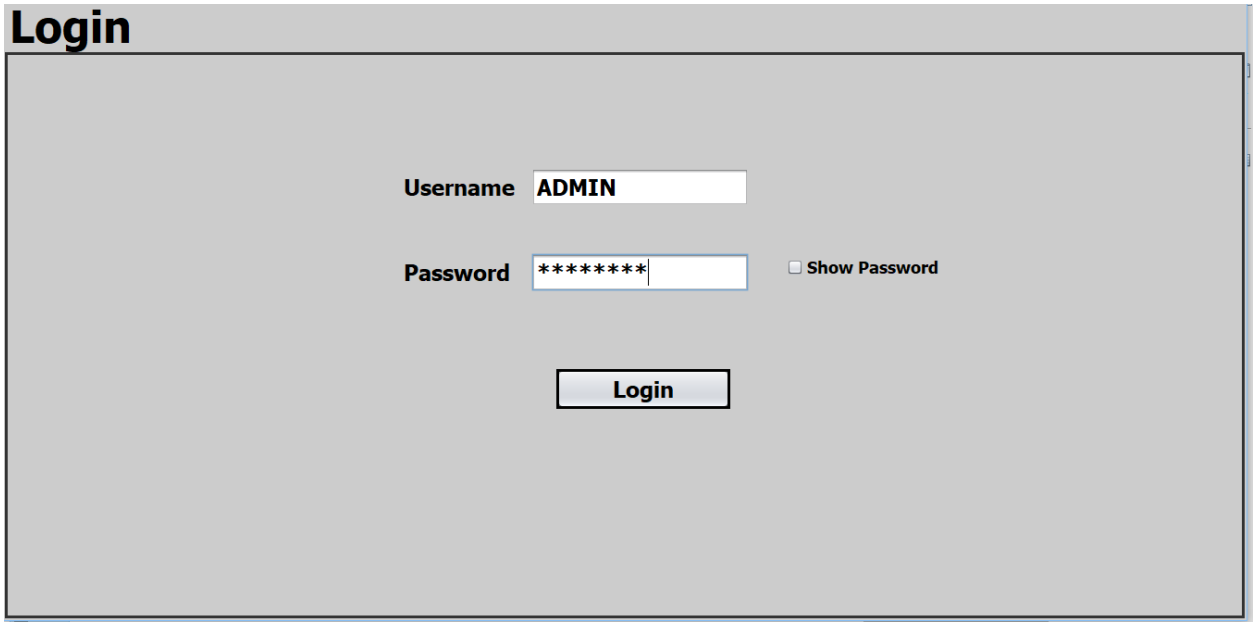


Figure 4.1

The page shown in figure 4.1 is used to login to the application.

4.2 :Home Page

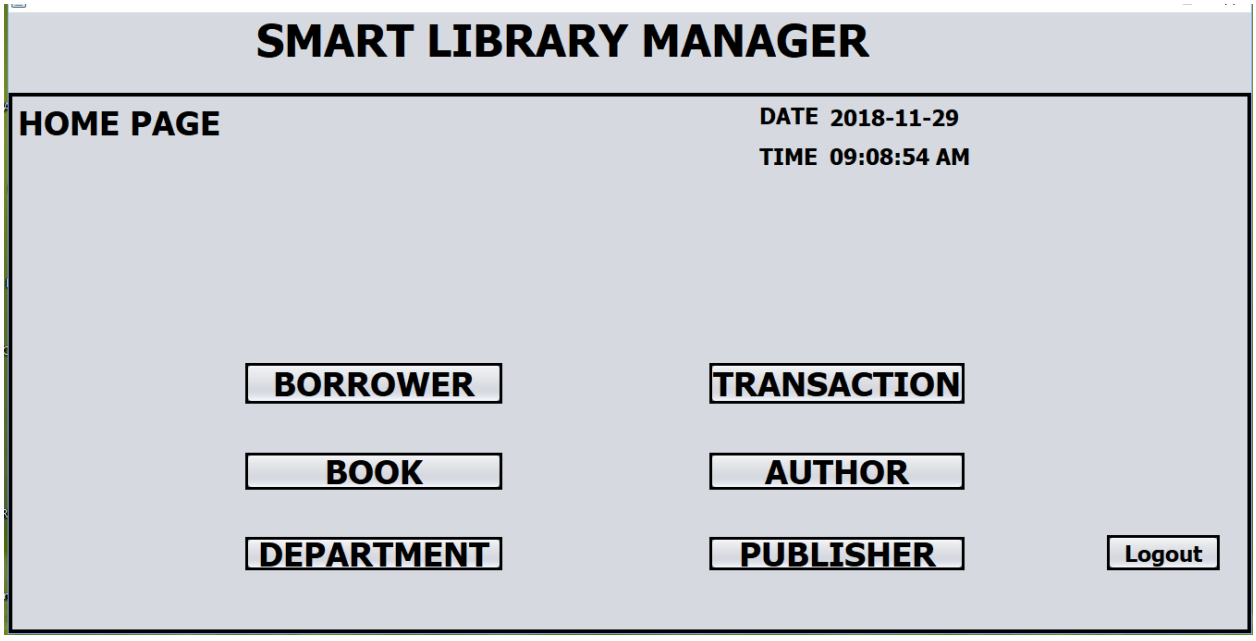


Figure 4.2

The page shown in figure 4.2 is used as a home page.

4.3 :Department Page

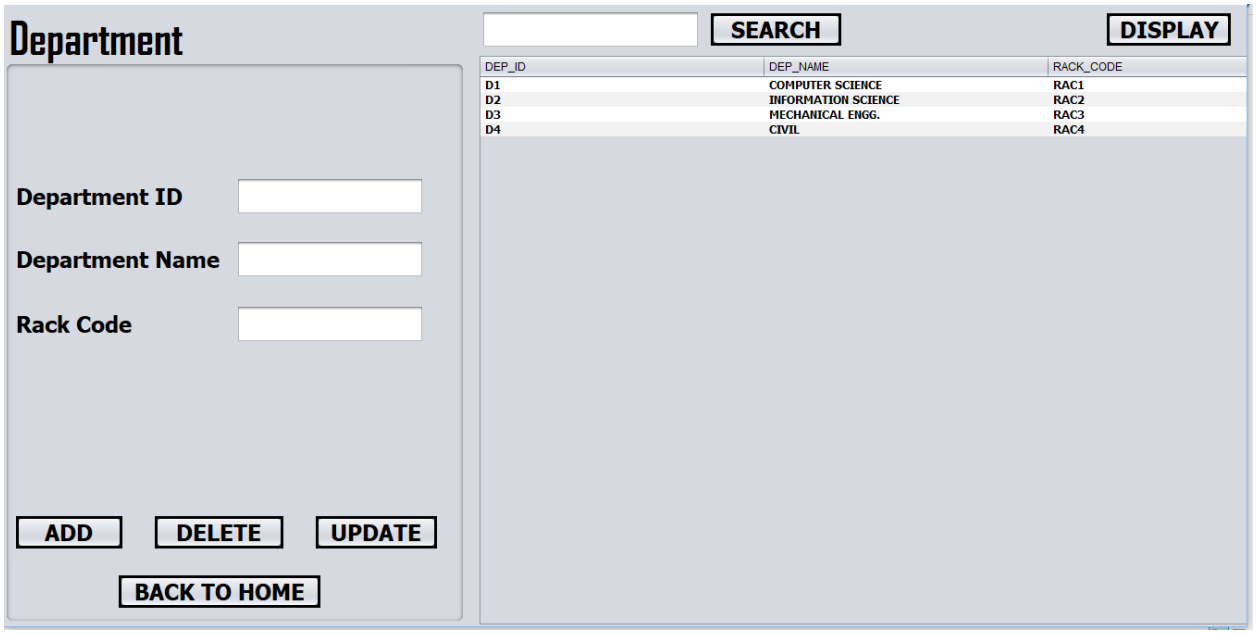


Figure 4.3

The page shown in figure 4.3 is used to insert ,update ,delete and display the department details

4.4 :Book Page

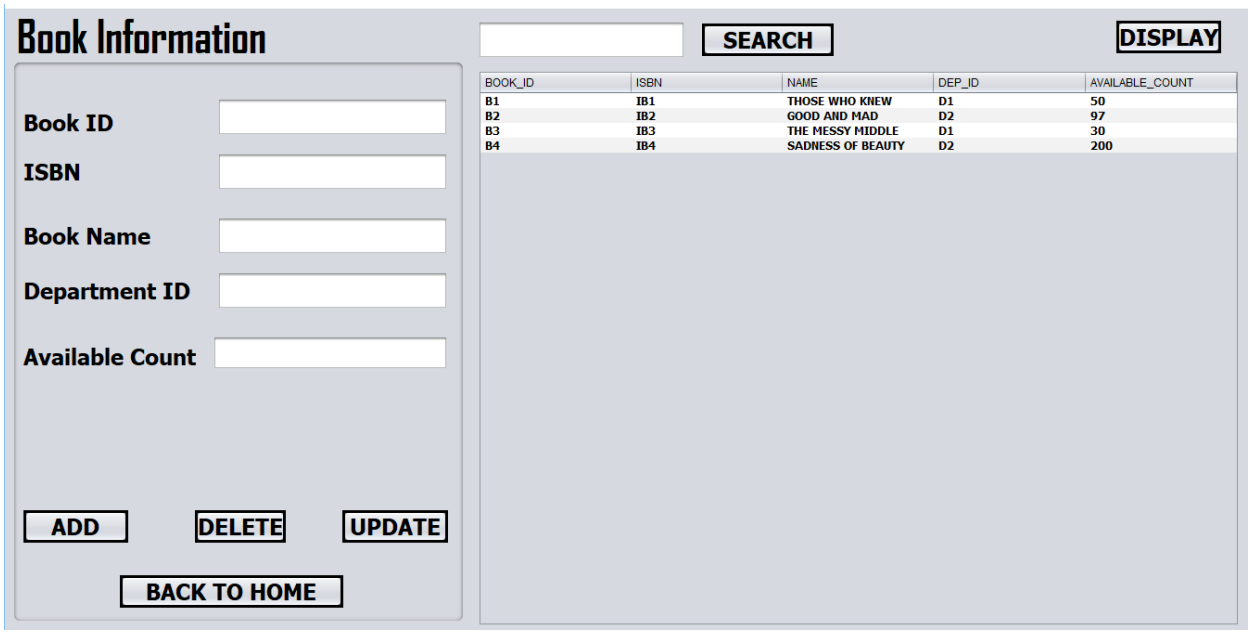


Figure 4.4

The page shown in figure 4.4 is used to insert ,update ,delete and display the book details.

4.5 :Author Page

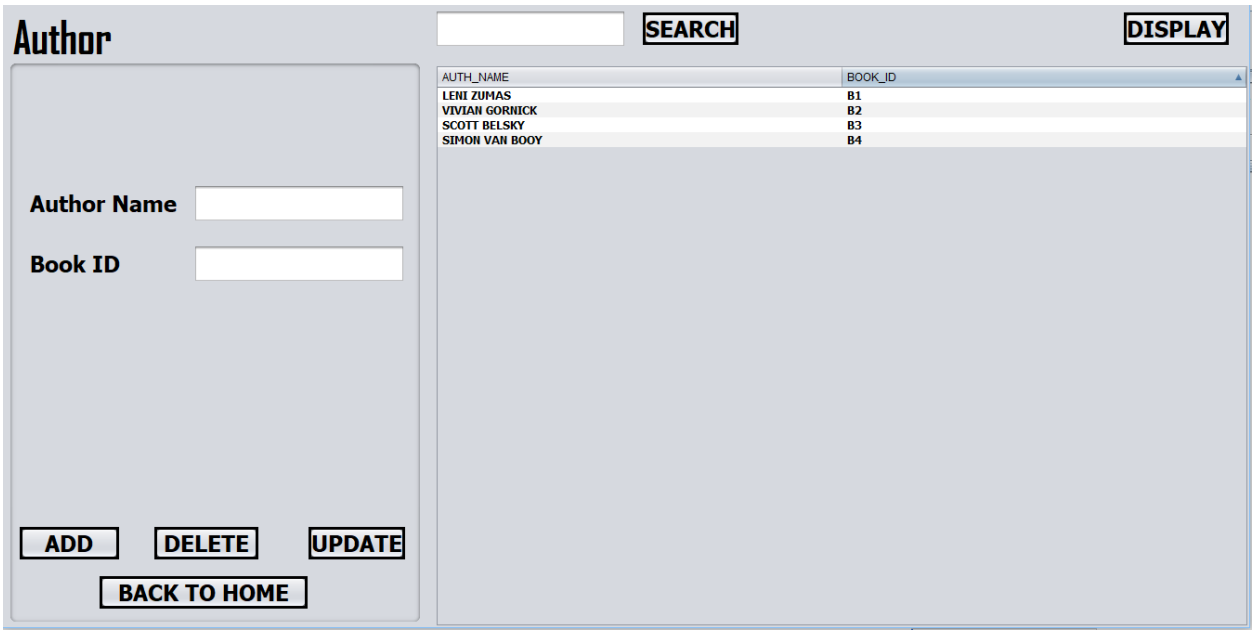


Figure 4.5

The page shown in figure 4.5 is used to insert ,update ,delete and display the author details.

4.6 :Publisher Page

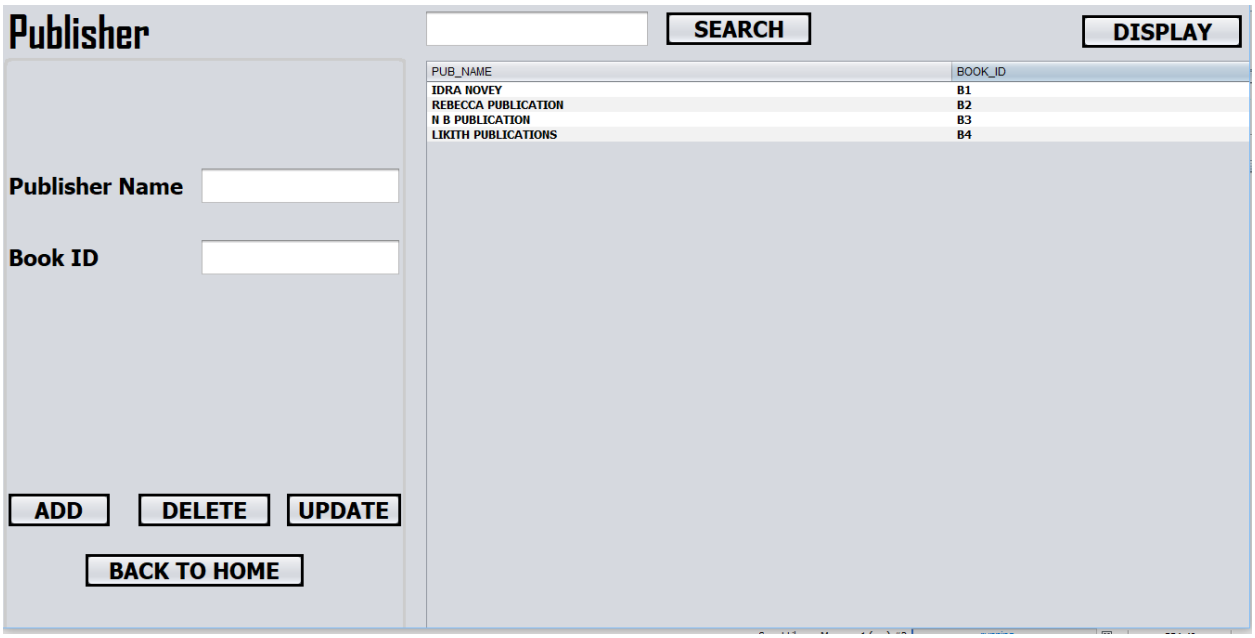


Figure 4.6

The page shown in figure 4.6 is used to insert ,update ,delete and display the publisher details.

Figure 4.7 :Borrower Page

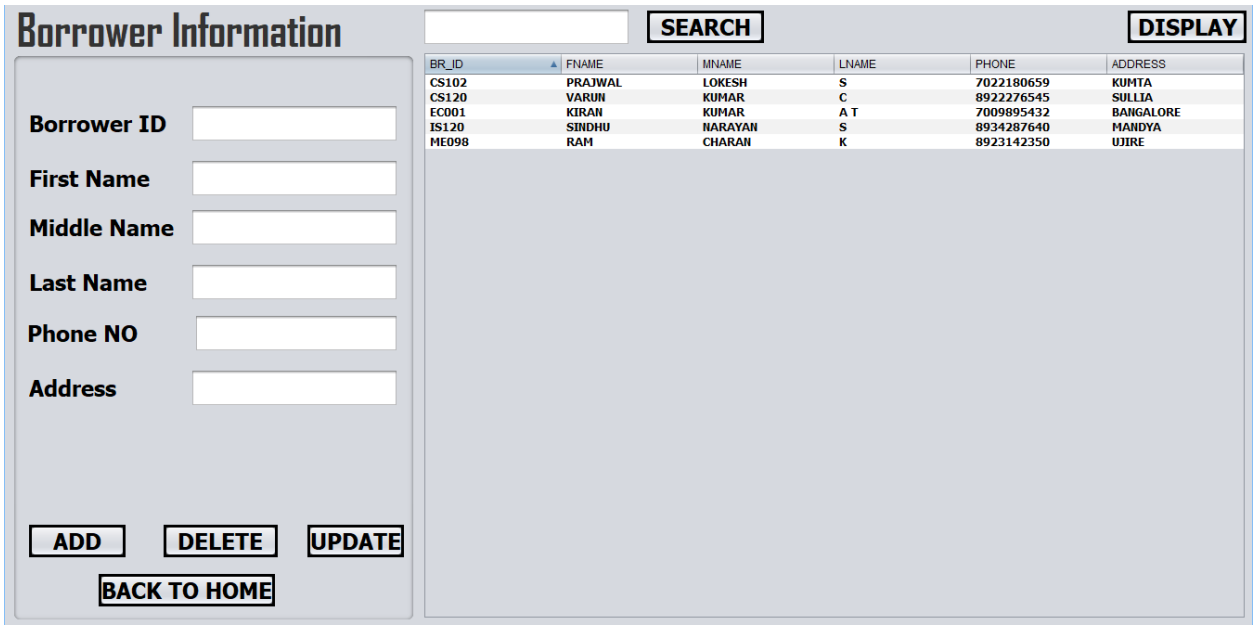


Figure 4.7

The page shown in figure 4.7 is used to insert ,update ,delete and display the borrower details.

Figure 4.8 : Transaction Page

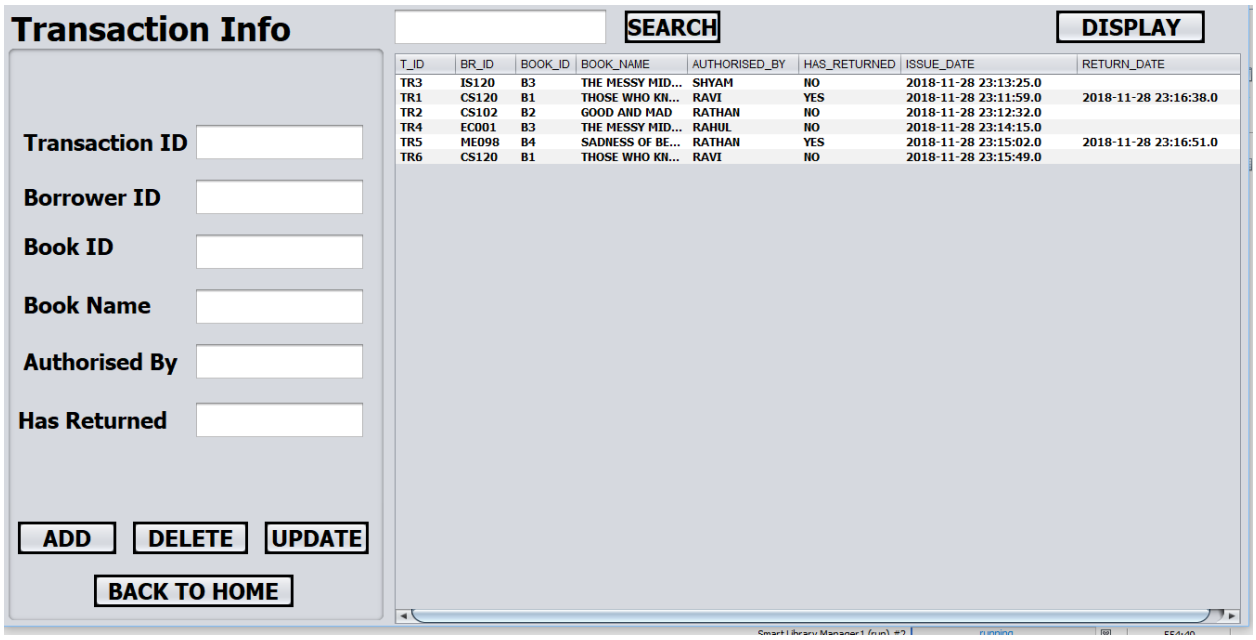


Figure 4.8

The page shown in figure 4.8 is used to insert ,update ,delete and display the transaction details.

CHAPTER 5

Conclusion

While developing this software a conscious effort has been made to develop a system package using available resources tools and technique while making the system an eye has been kept out to ensure that the user interface is user friendly as cost effective as flexible as possible . And we hope that the system will be acceptable to any user and it will adequately meet his/her requirements .As in any system development process where there are a number of shortcomings ,there have been some shortcomings in the development of this system also . The project is still under modification .

References

[1] Oracle

https://en.wikipedia.org/wiki/Oracle_Database

[2] Java

https://en.wikipedia.org/wiki/Java_(programming_language)

[3] NetBeans

https://en.wikipedia.org/wiki/NetBeans

[4] Database Systems Models , Languages , Design and Application Programming ,Ramez Elmasri and Shamkant B . Navathe , 6th Edition , Pearson