语言课程设计实验报告

> 姓名：牛佳飞　　学号：57119203　　专业：网络空间安全
> 起始日期：2021.7.1　　结束日期：2021.7.31

一、设计任务

题目三：图书室管理系统

该系统需创建和管理以下信息：1、书籍信息：书名、书目编号、作者名、出版日期、出版社、库存册数、登记号数据集；2、每册书的登记信息：登记号、是否借出、借阅日期、借书证号。

系统功能要求如下：

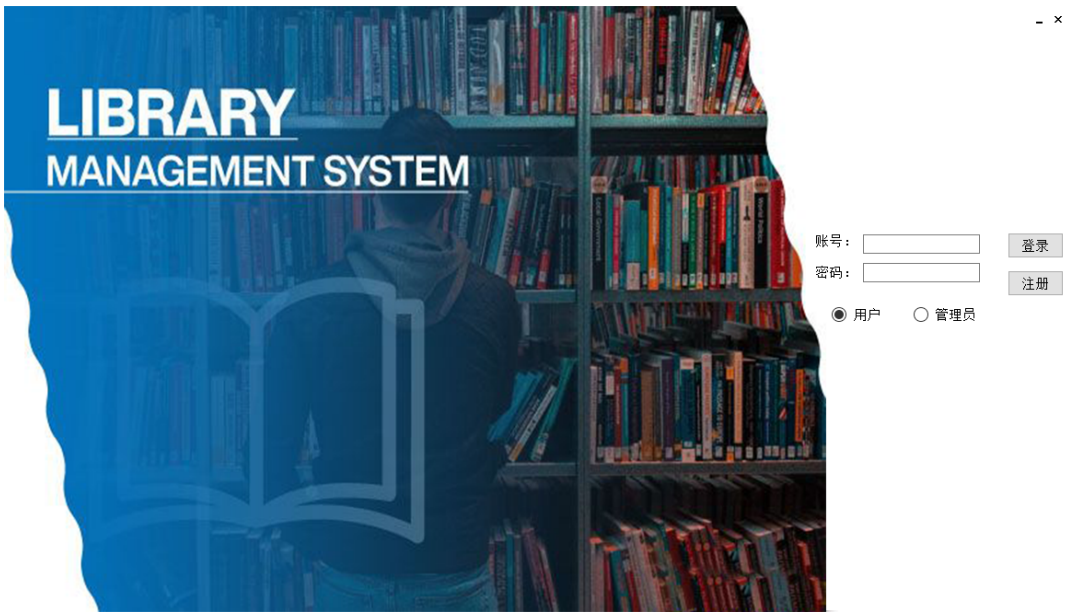
1. 创建和管理描述每本书籍的对象；
2. 创建和管理描述每册书登记信息的对像；
3. 增加和删除书籍；
4. 借书操作：读者提供书籍信息（书名或书目编号），检查该书籍是否可借（有没有没借出的登记号），可借时将某一登录号设置为借出，填入借书证号和借出日期；
5. 还书操作：根据书籍名先找到书，然后找到欲还书的登记号并修改为可借，同时删除借书证信息；
6. 基本查询功能；
7. 数据文件读写：文件中包含所有书籍信息、每个书籍的登记信息等数据；
8. 基本信息显示：1）所有书籍信息显示；3）特定书籍的借阅信息（已借出或可借）；
9. 可选功能提升：根据登记号直接还书操作等；

二、设计实现的方法

1. **系统主要功能**

（1）登录系统：

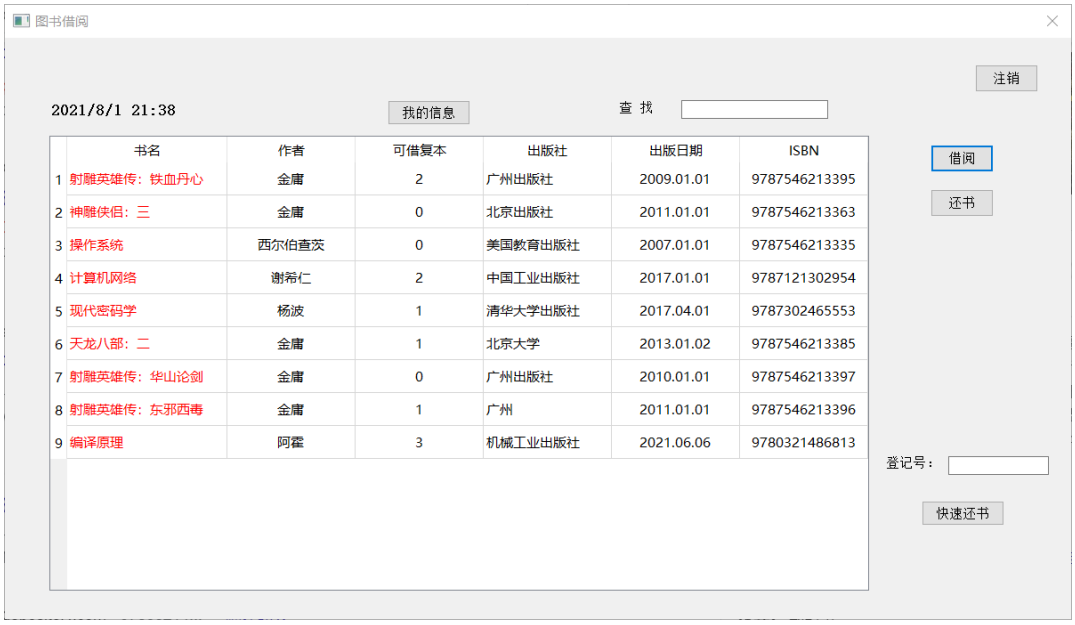
- 登录
- 注册
- 记录登录时间



（2）用户系统：

- 查询功能（匹配表格任意字段，模糊搜索）

- 选中表格项目借书&还书
- 右下角根据登记号快速还书
- 查询用户个人信息（用户名、上次登录时间、借阅记录等）
- 注销，返回登录界面

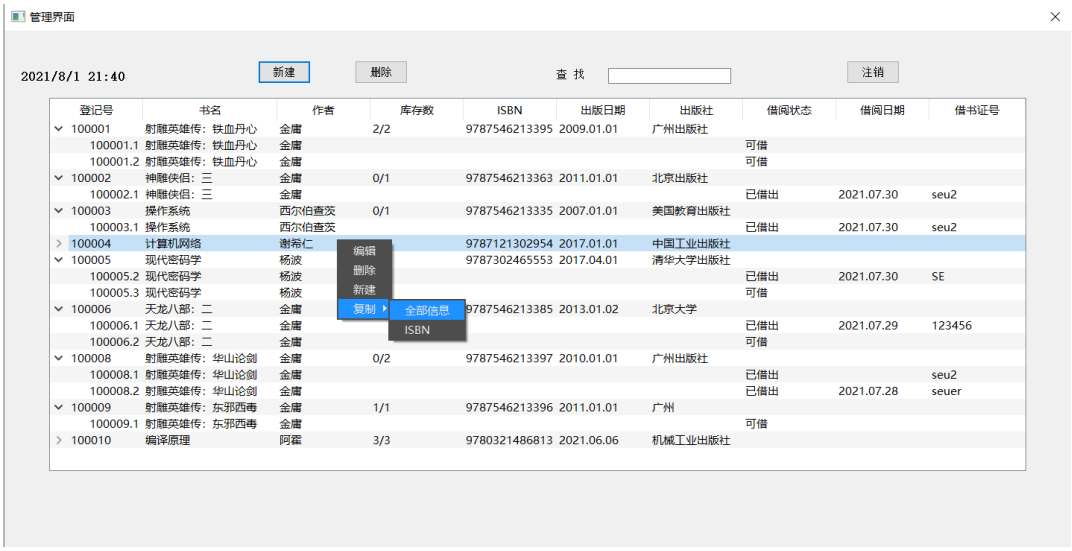


	书名	作者	可借复本	出版社	出版日期	ISBN
1	射雕英雄传：铁血丹心	金庸	2	广州出版社	2009.01.01	9787546213395
2	神雕侠侣：三	金庸	0	北京出版社	2011.01.01	9787546213363
3	操作系统	西尔伯查茨	0	美国教育出版社	2007.01.01	9787546213335
4	计算机网络	谢希仁	2	中国工业出版社	2017.01.01	9787121302954
5	现代密码学	杨波	1	清华大学出版社	2017.04.01	9787302465553
6	天龙八部：二	金庸	1	北京大学	2013.01.02	9787546213385
7	射雕英雄传：华山论剑	金庸	0	广州出版社	2010.01.01	9787546213397
8	射雕英雄传：东邪西毒	金庸	1	广州	2011.01.01	9787546213396
9	编译原理	阿霍	3	机械工业出版社	2021.06.06	9780321486813

	登记号	书名	ISBN	借阅状态	借阅时间
1	100004.1	计算机网络	9787121302954	未还	2021.08.01 21:39:03
2	100001.1	射雕英雄传：铁血丹心	9787546213395	已还	2021.07.30 19:08:19
3	100003.1	操作系统	9787546213335	未还	2021.07.30 09:53:09
4	100002.1	神雕侠侣：三	9787546213363	未还	2021.07.30 09:51:29

(3) 管理员系统

- 查找功能（匹配树形视图的任意字段，模糊搜索）
- 右键菜单功能
- 对**一本书籍**的增、删、改
- 对**同种所有书籍**的增、删、改
- 复制书籍信息
- 注销，返回登录界面

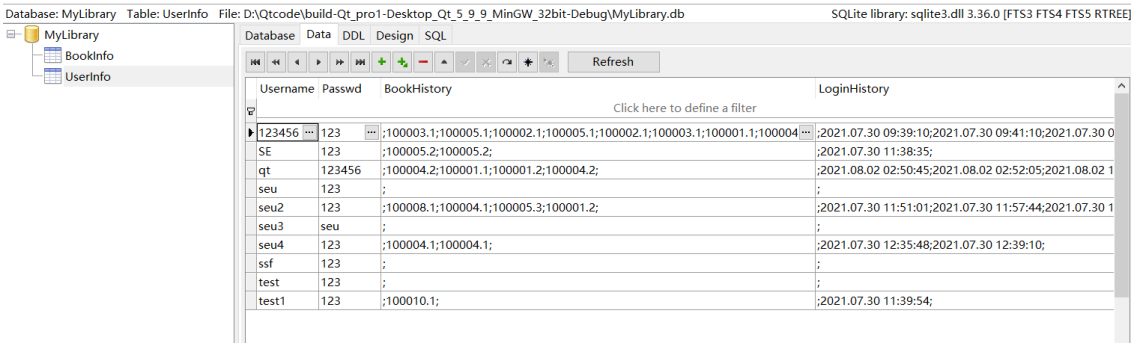


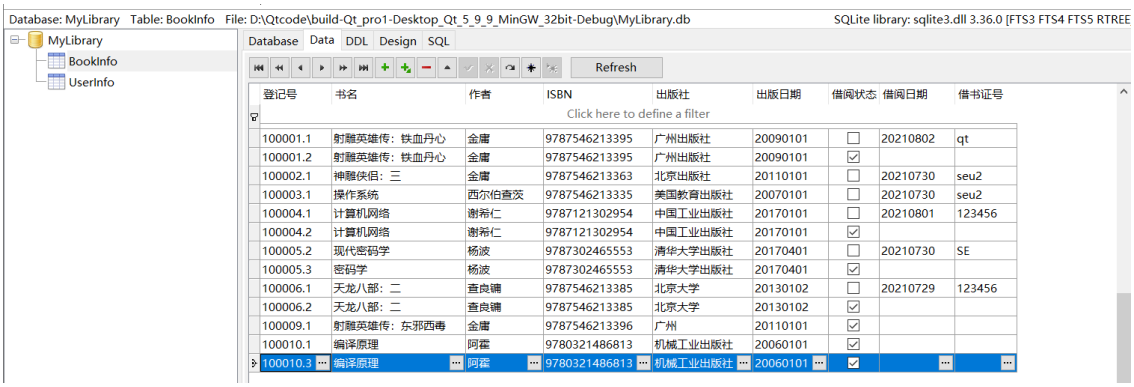
2. **对设计任务的理解和系统分析**

（1）题目要求有一本书**自身信息**的描述对象和书的**管理信息**的描述对象，所以要为它们单独定义一个类，此后都是围绕这两个基本元素展开的。

同时这两个类之间又不是孤立的，可以考虑在管理信息类创建一个书籍信息类的对象，又考虑到每本书具有唯一的登记号，所以应该在管理信息类中创建一个书籍信息类的对象数组（向量）。

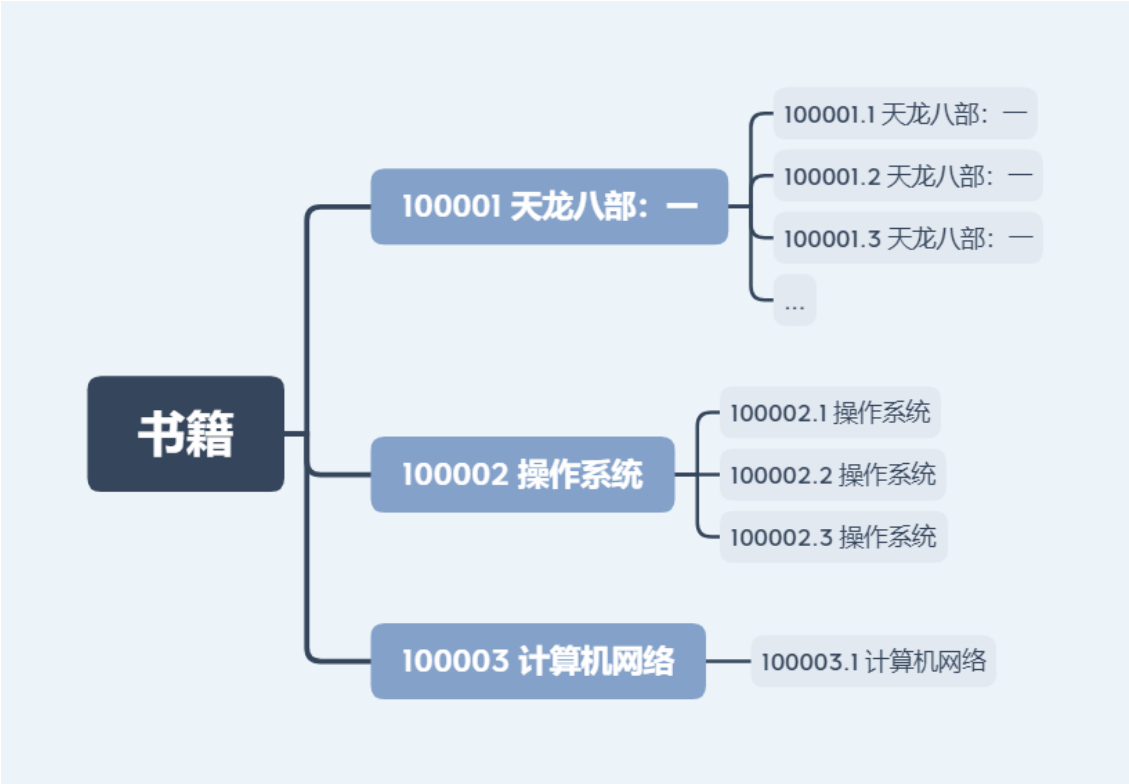
（2）题目中提到了大量的**增、删、查、改**操作，为了方便管理和操作，采用了Qt自带的数据库**SQLite**。Qt对SQL语句进行了不同程度的封装，使用比较灵活。既可以直接使用SQL语句进行交互，也可以使用封装好的model进行操作。





虽然数据库使得信息管理更加方便，但将数据库读出的信息保存到有结构的类当中，还需要自己去实现比较复杂的算法。

因为整个程序的数据结构主要是一个就像管理界面展示的**树形结构**，数据库中是**线性的信息**（所有数据地位相同），需要自己设计算法产生父结点，添加对应的子结点，将线性数据结构转化为非线性数据结构。



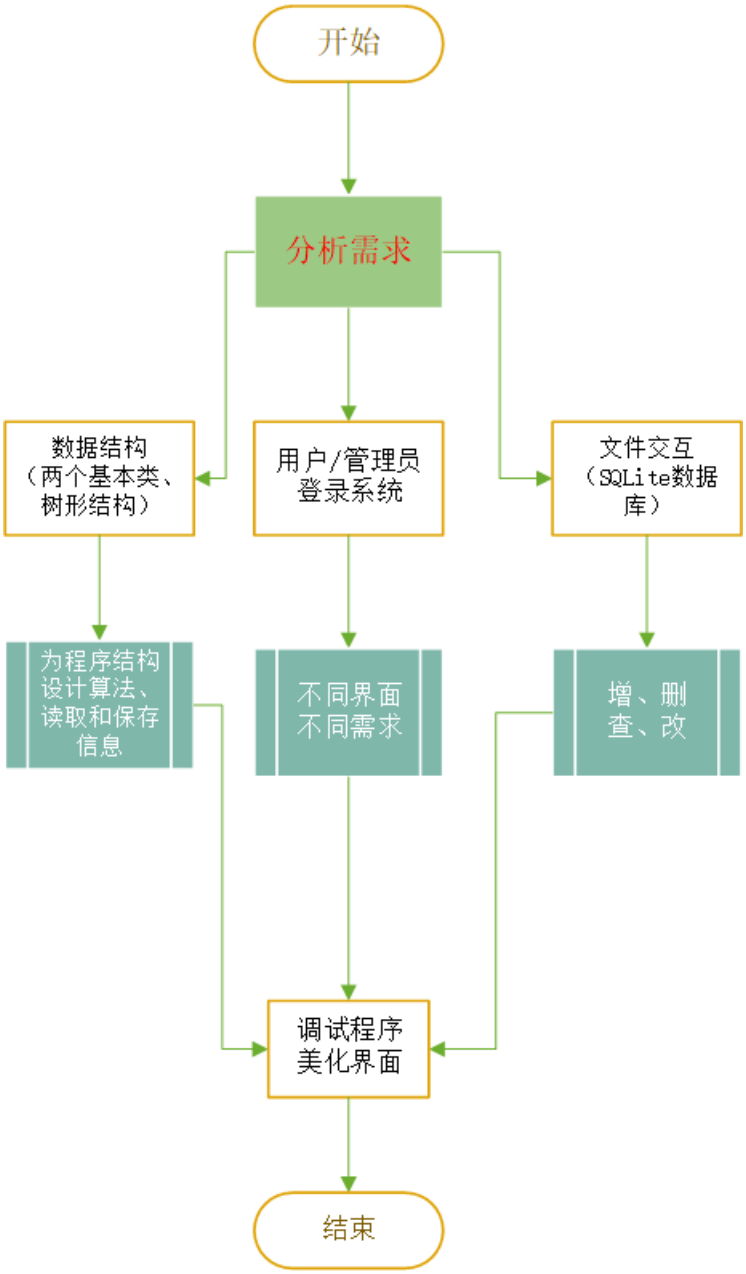
（3）信息的显示：

- 用户系统：用户并不关心书籍具体的唯一登记号等管理信息，所以为用户展现的只有书籍本身的一些信息，近似于线性结构，所以采用了简单实用`QTableView`**表格视图。**
- 管理系统：管理员需要对整个程序的数据结构拥有控制权，所以为了直观展示并且匹配程序的数据结构，采用了`QTreeView`**树形视图**，相对`QTableView`编程的复杂度大大提高，增、查、删、改等操作既要考虑父结点，还要考虑每一项子结点，以及特殊结点（既是父结点，又是子结点，即该种书一共只有一本的情况）

（3）除了题目中描述的功能外，为了使程序更完整、更灵活、更贴近实际，将**用户和管理员系统**分离，实现了一个较为完整的登录--注册--用户/管理员系统。

（4）在完成系统功能的基础上，对界面做了**美化处理**，如登录界面的无边框和背景处理，但美化的同时也会带来一些自带属性或控件不可用，如拖动、关闭等，需要自己重新实现这些功能。

流程图：



三、关键代码说明

1. **两个基本类：**

```
/**
  *  @brief     书籍的描述对象
  *  @file      book.h
  *  @author    57119203牛佳飞
  *  @version   5.0(版本号)
  *  @date      2021-07-20
  */
#ifndef BOOK_H
#define BOOK_H
# include<qstring.h>
#include<QDate>

class Book
{
public:
    Book();
    QString m_strBookName; //书名
    QString m_strISBN; //书目编号，即ISBN号
```

```
    QString m_strAuthor; //作者
    QDate m_dtPublicationDate; //出版日期
    QString m_strPublishingHouse; //出版社
    QString m_strRegistrationNumber; //登记号
};
```

```
/**
  *  @brief     书籍的管理信息对象
  *  @file      bookinfo.h
  *  @author    57119203牛佳飞
  *  @version   5.0(版本号)
  *  @date      2021-07-20
  */
#ifndef BOOKINFO_H
#define BOOKINFO_H
#include"book.h"
#include<QDate>
#include<QVector>
#include<QMap>
typedef struct
{
    Book m_Book; //书籍自身信息
    bool m_bIsBorrowed; //借阅状态
    QDate m_dtBorrowingDate; //借阅日期
    QString m_strLibraryCardNumber; //借书证号
}g_BookInfo; //描述书籍本身的信息和它的管理信息

class BookInfo
{
public:
    QVector<g_BookInfo> m_BookInfo;  //创建一个元素为包含书自身信息和管理信息的向量
    QString m_strRegistrationNumber; //登记号
    int m_nInventory; //总库存
    int m_nRealInventory; //实际库存
    BookInfo();
};
#endif // BOOKINFO_H
```

2. **主窗口-----连接数据库**

连接数据库是运行程序后的第一步，数据库的两个表（`UserInfo`、`BookInfo`）将整个程序紧密地联系起来。

```
/* 连接数据库 */
void MainWindow::ConnectDatabase()
{
    QSqlDatabase g_dbAccount=QSqlDatabase::addDatabase("QSQLITE"); //建立和SQlite数据库的连接
    g_dbAccount.setDatabaseName("MyLibrary.db"); //选择数据库
    if(g_dbAccount.open()) //打开数据库
    {
        qDebug()<<"连接成功";
        QMessageBox::about(NULL,"提示","数据库连接成功");
    }
    else
    {
        QMessageBox::about(NULL,"提示","数据库连接失败");
```

```
        exit(0);
    }
}
```

3. **主窗口-----用数据库初始化保存书籍信息和管理信息的全局向量`g_vBookInfo`。**

   全局向量`g_vBookInfo`在程序中至关重要，`g_vBookInfo`的每个元素是一种书的对象，同时每个元素又包含多本同种书组成的一个向量。它将数据库中的线性数据读取并保存为树形结构，更好地适应程序的要求。

   因为数据库中是默认按照登记号排序的，所以初始化时要根据前一项和后一项的登记号前缀是否相同，来判断它们是否是同一种书，同时还要对这同一种书提取信息，共同维护一个父结点，最终形成了多个父结点（`g_vBookInfo[i]`）下多个子结点（`g_vBookInfo[i].m_BookInfo[j]`）的树形结构。此外，循环读取时还需要考虑到一些特殊情况，例如第一个和最后一个元素，看是否能归并到一般情况，以及具体向量元素的数据类型和数据库字段的数据类型之间的转换。

```
/* 从数据库读取信息，初始化程序的数据结构 */
void MainWindow::InitializeBookInfo()
{

    g_vBookInfo.clear();
    QSqlQuery t_query;
    t_query.prepare("SELECT * FROM BookInfo");  //查询数据库
    t_query.exec();

    int index=0; //记录读取到的信息，用于特殊处理第一条
    BookInfo t_tmp; //暂存读取到的信息
    while(t_query.next())
    {

        QString t_strRegistrationNumber=t_query.value("登记
号").toString().section('.',0,0); //登记号小数点前的数字，用来标识同种书籍
        g_BookInfo t_BookInfo; //暂存读取到的信息
        t_BookInfo.m_Book.m_strISBN=t_query.value("ISBN").toString();
        t_BookInfo.m_Book.m_strAuthor=t_query.value("作者").toString();
        t_BookInfo.m_Book.m_strBookName=t_query.value("书名").toString();
        t_BookInfo.m_Book.m_strPublishingHouse=t_query.value("出版
社").toString();

 t_BookInfo.m_Book.m_dtPublicationDate=QDate::fromString(t_query.value("出版
日期").toString(),"yyyyMMdd");
        t_BookInfo.m_Book.m_strRegistrationNumber=t_query.value("登记
号").toString();
        t_BookInfo.m_strLibraryCardNumber=t_query.value("借书证
号").toString();
        t_BookInfo.m_bIsBorrowed=t_query.value("借阅状态").toBool();
        t_BookInfo.m_dtBorrowingDate=QDate::fromString(t_query.value("借阅日
期").toString(),"yyyyMMdd");
        if(index!=0&&t_strRegistrationNumber!=t_tmp.m_strRegistrationNumber)
//添加新的父结点项
        {
            g_vBookInfo.append(t_tmp);
            t_tmp.m_nInventory=0;
            t_tmp.m_nRealInventory=0;
            t_tmp.m_BookInfo.clear();
        }
        if(t_BookInfo.m_bIsBorrowed==1) //可借，将实际库存+1
```

```
            t_tmp.m_nRealInventory++;
        t_tmp.m_nInventory++;  //总库存++
        t_tmp.m_BookInfo.append(t_BookInfo); //父结点项添加子结点
        t_tmp.m_strRegistrationNumber=t_strRegistrationNumber; //更新用于判断同种书籍的登记号前缀
        index++;
    }
    g_vBookInfo.append(t_tmp); //上面的循环中由于循环条件会漏掉最后一个结点，这里单独处理
}
```

4. **主窗口-----登录**

先过滤一些非法输入的情况，如用户名和密码有空值等等。再判断是否是管理员的特殊账号`admin`，如果不是，再到数据库中查找获取的用户名和密码，只有同时匹配时，才能判定登录成功。登录成功后，创建相应的子窗口，并将当前用户名传递给子窗口。

```
/* 登录 */
void MainWindow::on_LoginButton_clicked()
{
    QString t_strUsername=ui->UsernameEdit->text(); //获取用户名
    QString t_strPasswd=ui->PasswdEdit->text(); //获取密码
    if(ui->AdminModeButton->isChecked()) //查看选择的登录模式
    {
        if(t_strUsername=="admin"&&t_strPasswd=="admin") //检查管理员用户名和密码
        {
            QMessageBox::information(this, "登录提示","登录成功，欢迎"+t_strUsername+"!");
            AdminMode t_AdminWindow(t_strUsername); //新建管理系统的窗口，并将当前用户名传给新窗口
            this->hide(); //隐藏主窗口
            t_AdminWindow.exec();
        }
        else
            QMessageBox::information(this, "登录提示","用户名或密码错误，请重新输入！");
    }
    else
    {

        QSqlTableModel *t_model=new QSqlTableModel; //构造model查询数据库
        t_model->setTable("UserInfo");
        t_model->setFilter(QString("Username='%1' and Passwd='%2'")
                           .arg(t_strUsername).arg(t_strPasswd));
        t_model->select();
        if( t_model->rowCount()) //如果行数不为0，则说明查到该记录，用户名和密码匹配且正确
        {
            QSqlQuery sqlQuery;
            sqlQuery.exec(QString("UPDATE UserInfo SET LoginHistory=LoginHistory||'%1' WHERE Username='%2'").arg(QDate::currentDate().toString("yyyy.MM.dd")+" "+QTime::currentTime().toString("hh:mm:ss")+";")
                          .arg(t_strUsername));
            QMessageBox::information(this, "登录提示","登录成功，欢迎"+t_strUsername);
```

```
            UserMode t_UserWindow(t_strUsername); //新建用户系统的窗口，并将当前
用户名传给新窗口
            this->hide(); //隐藏主窗口
            t_UserWindow.exec();
        }
        else
            QMessageBox::information(this, "登录提示","用户名或密码错误，请重新输
入！");
    }
}
```

5. **主窗口-----注册新用户**

首先过滤掉一些非法输入的情况，如输入存在空值，或者两次密码不匹配等等。如果输入合法，在数据库中查找获取到的用户名判断是否已经存在该用户名，如果发现不冲突，则把用户名和密码添加到数据库中，成功注册。

```
/* 注册 */
void Register::on_RegisterButton_clicked()
{
    QString t_strUsername=ui->UsernameEdit->text(); //获取用户名
    if(t_strUsername=="") //用户名为空，报错
    {
        QMessageBox::information(this, "注册提示","请输入正确格式的用户名和密
码！");
        return ;
    }
    QString t_strPasswd;
    if(ui->PasswdEdit_1->text()==ui->PasswdEdit_2->text()&&ui->PasswdEdit_1-
>text()!="") //当两次密码输入相同且不为空时，允许注册
        t_strPasswd=ui->PasswdEdit_1->text();
    else
    {
        QMessageBox::information(this, "注册提示","两次输入密码不一致，请重新输
入！");
        ui->PasswdEdit_1->clear();
        ui->PasswdEdit_2->clear();
        return;
    }
    QSqlTableModel *t_model=new QSqlTableModel;
    qDebug()<<t_strUsername;
    t_model->setTable("UserInfo");
    t_model->setFilter(QString("Username='%1'").arg(t_strUsername)); //查找是
否已存在该用户名
    t_model->select();
    if(t_model->rowCount()> 0)
    {
        QMessageBox::information(this, "注册提示","用户已存在");
    }
    else //如果不存在该用户名，则允许进行注册
    {
        QString t_strCmd=QString("insert into UserInfo
values('%1','%2',';',';',';')").arg(t_strUsername) .arg(t_strPasswd); //将用
户名和密码添加到数据库中
        QSqlQuery t_query;
        if(t_query.exec(t_strCmd))
            QMessageBox::information(this, "注册提示","注册成功");
```

```
        else
            QMessageBox::information(this, "注册提示","注册失败");
        this->close(); //关闭注册窗口
    }
}
```

6. **主窗口-----实现窗口拖动**

   自定义鼠标左键按下、松开、移动的鼠标事件，实现无边框窗口的拖动效果。

```
/* 实现窗口拖动 */
void MainWindow::mouseMoveEvent(QMouseEvent *event) //获取鼠标左键按下时移动的路径
{
    if (m_bPressed)
        move(event->pos() - m_point + pos());
}

void MainWindow::mouseReleaseEvent(QMouseEvent *event) //松开鼠标左键
{
    Q_UNUSED(event);
    m_bPressed = false;
}

void MainWindow::mousePressEvent(QMouseEvent *event) //按下鼠标左键
{
    if (event->button() == Qt::LeftButton)
    {
        m_bPressed = true;
        m_point = event->pos();
    }
}
```

7. **主窗口-----绘制背景图**

```
/* 为主窗口添加背景 */
void MainWindow::paintEvent(QPaintEvent *event)//设置背景图片
{
    QPainter painter(this); //创建画家，指定绘图设备
    QPixmap pix; //创建QPixmap对象
    pix.load(":/Resource/background.png"); //加载图片
    painter.drawPixmap(0,0,this->width(),this->height(),pix); //绘制背景图
}
```

8. **用户界面-----初始化表格视图**

   用户并不关心书籍具体的登记号、数量等管理信息，他们更关心书籍自身的信息，所以只需根据全局变量`g_vBookInfo`中的父结点（`g_vBookInfo[i]`）和任一子结点中书籍的自身信息初始化表格。

   此外还对表格做出一些样式设置，如居中对齐、设置只读、颜色等等。

```
/* 初始化表格视图 */
void UserMode::InitializeTable()
{
    QStandardItemModel *model = new QStandardItemModel();
    model->setColumnCount(6); //设置列数
```

```
    model->setHeaderData(0,Qt::Horizontal,"书名"); //设置列头
    model->setHeaderData(1,Qt::Horizontal,"作者");
    model->setHeaderData(2,Qt::Horizontal,"可借复本");
    model->setHeaderData(3,Qt::Horizontal,"出版社");
    model->setHeaderData(4,Qt::Horizontal,"出版日期");
    model->setHeaderData(5,Qt::Horizontal,"ISBN");
    for(int i=0;i<g_vBookInfo.size();++i)
    {
        model->setItem(i,0,new
QStandardItem(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strBookName)); //插入数据
        model->setItem(i,1,new
QStandardItem(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strAuthor));
        model->setItem(i,2,new
QStandardItem(QString::number(g_vBookInfo[i].m_nRealInventory)));
        model->setItem(i,3,new
QStandardItem(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strPublishingHouse));
        model->setItem(i,4,new
QStandardItem(g_vBookInfo[i].m_BookInfo[0].m_Book.m_dtPublicationDate.toStri
ng("yyyy.MM.dd")));
        model->setItem(i,5,new
QStandardItem(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strISBN));
        model->item(i,0)->setForeground(QBrush(QColor(255, 0, 0))); //设置第0
列红色字体
        model->item(i,1)->setTextAlignment(Qt::AlignCenter); //设置居中样式
        model->item(i,2)->setTextAlignment(Qt::AlignCenter);
        model->item(i,4)->setTextAlignment(Qt::AlignCenter);
        model->item(i,5)->setTextAlignment(Qt::AlignCenter);
    }
    ui->tableView->setModel(model);//为表格应用model
    ui->tableView->horizontalHeader()->setDefaultAlignment(Qt::AlignCenter);
//设置表头居中
    ui->tableView->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);//所有列都扩展自适应宽度，填充充满整个
屏幕宽度
    ui->tableView->horizontalHeader()->setSectionResizeMode(0,
QHeaderView::Fixed);//对第0列单独设置固定宽度
    ui->tableView->setColumnWidth(0, 180);//设置固定宽度
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);//设置
选中模式为选中行
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);//设置
表格只读
}
```

9. **用户界面-----查找**

因为用户界面采用的是表格视图，查找也比较方便，直接按行列遍历整个表格，查找匹配字段即可。显示搜索结果可以通过设置该行隐藏与否实现，如果该行包含匹配字段，就显示该行，否则隐藏该行。

```
/* 用户界面查找 */
void UserMode::on_FindEdit_textChanged(const QString &arg1)
{

    if(ui->FindEdit->text()=="") //查找框为空时，不进行对列的隐藏
    {
        for(int i=0;i<ui->tableView->model()->rowCount();i++)
            ui->tableView->setRowHidden(i,false);
```

```
    }
    else
    {
        QString str=ui->FindEdit->text();  //获取查找框内容
        str.remove(QRegExp("\\s")); //去除空格
        for(int i=0;i<ui->tableView->model()->rowCount();i++) //遍历表格每一行
        {
            ui->tableView->setRowHidden(i,true); //隐藏该列
            QString t_strTmp=""; //临时字符串,保存每一格的内容
            //提取信息
            QAbstractItemModel *model=ui->tableView->model();
            QModelIndex index;
            for(int j=0;j<ui->tableView->model()->columnCount();j++) //遍历该
行的每一列
            {
                index=model->index(i,j);
                t_strTmp+=model->data(index).toString();
            }
            t_strTmp.remove(QRegExp("\\s")); //去除每一格内容中的空格
            if(t_strTmp.contains(str,Qt::CaseSensitive)) //如果t_strTmp包含要
查找的字符串，则查找成功
                ui->tableView->setRowHidden(i,false);//取消隐藏该列
        }
    }
}
```

10. **用户界面-----选中表格借书**

    因为每本书籍都以登记号作为唯一标识，所以需要获取表格选中的书籍信息，特别是登记号。之后的借阅操作就是通过登记号查询数据库，修改记录的一些字段的值（借阅状态、借阅日期、借书证号）。

    为了和用户个人信息界面联系，此处还需要在借阅之后保存用户的借阅信息（包括书名、ISBN、借阅时间等等）。

```
/* 用户界面还书 */
void UserMode::on_BorrowButton_clicked()
{

    int t_nSelectedRow= ui->tableView->currentIndex().row(); //获取当前选中行的
下标
    QAbstractItemModel *model = ui->tableView->model ();
    QModelIndex index = model->index(t_nSelectedRow,5);//获取选中行的索引
    QVariant data = model->data(index);//选中行第一列的内容
    QString t_strSelectedItem=data.toString();
    QSqlTableModel *t_model=new QSqlTableModel;
    t_model->setTable("BookInfo");
    t_model->setFilter(QString("ISBN='%1' and 借阅状态='1'
").arg(t_strSelectedItem)); //根据ISBN号和借阅状态查询数据库
    t_model->select();
    if( t_model->rowCount()>0) //查找结果大于0，说明的确有此书被借出
    {
        QSqlRecord record = t_model->record(0); //获取数据库中的该条记录
        record.setValue("借阅状态", 0); //重设借阅状态为不可借
        record.setValue("借阅日期",QDate::currentDate().toString("yyyy-MM-
dd").replace(QRegExp("-"), "")); //设置借阅日期
        record.setValue("借书证号",m_strCurrentUser); //设置借书证号
        m_strSelectRegNum=record.value("登记号").toString(); //获取该书的登记号
```

```
        t_model->setRecord(0, record);
        t_model->submitAll(); //更新数据库
        QSqlQuery sqlQuery;
        //更新数据库中该用户的借书历史
        sqlQuery.exec(QString("UPDATE UserInfo SET
BookHistory=BookHistory||'%1' WHERE
Username='%2'").arg(m_strSelectRegNum+";").arg(m_strCurrentUser));
        //更新数据库中该用户的借阅时间
        sqlQuery.exec(QString("UPDATE UserInfo SET
BorrowTime=BorrowTime||'%1' WHERE
Username='%2'").arg(QDate::currentDate().toString("yyyy.MM.dd")+"
"+QTime::currentTime().toString("hh:mm:ss")+";").arg(m_strCurrentUser));
        MainWindow::InitializeBookInfo(); //更新全局书籍信息
        InitializeTable(); //重新初始化当前表格视图
        QMessageBox::information(this, "借阅提示","借阅成功，书籍登记
号："+m_strSelectRegNum);
    }
    else
    {
        QMessageBox::information(this, "借阅提示","此书已被借完！");
    }
}
```

11. **用户界面-----选中表格还书**

    与借书类似，需要获取到选中书籍的登记号，以此来查询数据库，修改记录的借阅状态、借阅日期和借书证号字段的值。

```
/* 还书 */
void UserMode::on_ReturnButton_clicked()
{
    int t_nSelectedRow= ui->tableView->currentIndex().row();//获取当前选中行的
下标
    QAbstractItemModel *model = ui->tableView->model ();
    QModelIndex index = model->index(t_nSelectedRow,5);//获取选中行的索引
    QVariant data = model->data(index); //选中行第一列的内容
    QString t_strSelectedItem=data.toString();
    QSqlTableModel *t_model=new QSqlTableModel;
    t_model->setTable("BookInfo");
    t_model->setFilter(QString("ISBN='%1' and 借书证号='%2'
").arg(t_strSelectedItem).arg(m_strCurrentUser));  //根据ISBN号和借阅状态查询数
据库
    t_model->select();
    qDebug()<<t_model->rowCount();
    if( t_model->rowCount()>0) //查找结果大于0，说明的确有此书被借出
    {
        QSqlRecord record = t_model->record(0);//获取数据库中的该条记录
        record.setValue("借阅状态", 1);  //重设借阅状态为可借
        record.setValue("借阅日期","");  //清空借阅日期
        record.setValue("借书证号","");  //清空借书证号
        t_model->setRecord(0, record);
        t_model->submitAll(); //更新数据库
        MainWindow::InitializeBookInfo(); //更新全局书籍信息
        InitializeTable(); //重新初始化当前表格视图
        QMessageBox::information(this, "借阅提示","归还成功！");

    }
```

```
    else
    {
        QMessageBox::information(this, "借阅提示","未查询到借阅记录，还书失败！");
    }
}
```

12. **用户界面-----快速还书**

    根据从文本框直接获取的登记号，查询数据库，修改记录的借阅状态、借阅日期和借书证号字段的值。

```
/* 快速还书 */
void UserMode::on_FastReturnButton_clicked()
{
    QString t_strRegistrationNumber=ui->ReturnBookEdit->text(); //获取文本框中的登记号
    if(t_strRegistrationNumber=="") //确认是否为空
    {
        QMessageBox::information(this, "借阅提示","请输入登记号！");
    }
    QSqlTableModel *t_model=new QSqlTableModel;
    t_model->setTable("BookInfo"); //根据登记号和借书证号查询数据库
    t_model->setFilter(QString("登记号='%1' and 借书证号='%2'").arg(t_strRegistrationNumber).arg(m_strCurrentUser));
    t_model->select();
    if( t_model->rowCount()>0) //查找成功
    {
        QSqlRecord record = t_model->record(0);
        record.setValue("借阅状态", 1); //重设借阅状态为可借
        record.setValue("借阅日期",""); //清空借阅日期
        record.setValue("借书证号",""); //清空借书证号
        t_model->setRecord(0, record);
        t_model->submitAll(); //更新数据库
        MainWindow::InitializeBookInfo(); //更新全局书籍信息
        InitializeTable(); //重新初始化当前表格视图
        QMessageBox::information(this, "借阅提示","归还成功！");
    }
    else
    {
        QMessageBox::information(this, "借阅提示","未查询到借阅记录，还书失败！");
    }
}
```

13. **用户界面-----个人信息**

    用户界面的个人信息会触发一个“我的信息“新窗口，主要通过从用户界面传递进来的用户名，查询数据库，通过利用分号分隔筛选查询到的登录时间、借阅历史和借阅时间，利用这些数据初始化表格和文本框。

```
/* 初始化表格 */
void UserInfo::Initialize()
{
    QSqlQuery sqlQuery;
    sqlQuery.exec(QString("SELECT LoginHistory FROM UserInfo WHERE Username='%1'").arg(m_strCurrentUser)); //查询用户登录历史
    sqlQuery.next();
```

```
    QString
t_strLoginTime=sqlQuery.value(0).toString().section(";",-3,-3);//提取上次登录的
日期和时间
    ui->LastLoginTime->setText(t_strLoginTime);
    ui->LastLoginTime->setReadOnly(true);
    QStandardItemModel *model = new QStandardItemModel();
    model->setColumnCount(5); //设置表格列数为5:
    sqlQuery.exec(QString("SELECT BookHistory FROM UserInfo WHERE
Username='%1'").arg(m_strCurrentUser)); //查询书籍借阅历史
    sqlQuery.next();
    QString t_strBookHistory=sqlQuery.value(0).toString();
    sqlQuery.exec(QString("SELECT BorrowTime FROM UserInfo WHERE
Username='%1'").arg(m_strCurrentUser)); //查询书籍借阅时间
    sqlQuery.next();
    QString t_strBorrowTime=sqlQuery.value(0).toString();

    QStringList t_strBookHistoryList = t_strBookHistory.split(";"); //根据分号
分隔多个借阅记录
    QStringList t_strBorrowTimeList=t_strBorrowTime.split(";");

    model->setHeaderData(0,Qt::Horizontal,"登记号");
    model->setHeaderData(1,Qt::Horizontal,"书名");
    model->setHeaderData(2,Qt::Horizontal,"ISBN");
    model->setHeaderData(3,Qt::Horizontal,"借阅状态");
    model->setHeaderData(4,Qt::Horizontal,"借阅时间");

    int t_nRowCount=t_strBookHistoryList.size()-2; //除去最开始的分号前的空白和最
后面的分号之后的空白
    for(int i=0;i<t_nRowCount;++i)
    {
        model->setItem(i,0,new
QStandardItem(t_strBookHistoryList[t_nRowCount-i])); //倒序插入，把时间近的插入到
靠前的列
        sqlQuery.exec(QString("SELECT *FROM BookInfo WHERE 登记号
='%1'").arg(t_strBookHistoryList[t_nRowCount-i]));
        sqlQuery.next();
        model->setItem(i,1,new QStandardItem(sqlQuery.value(1).toString()));
//设置表格的内容
        model->setItem(i,2,new QStandardItem(sqlQuery.value(3).toString()));
        if(sqlQuery.value(6).toBool()==1)
            model->setItem(i,3,new QStandardItem("已还"));
        else
            model->setItem(i,3,new QStandardItem("未还"));
        model->setItem(i,4,new
QStandardItem(t_strBorrowTimeList[t_nRowCount-i]));
        model->item(i,0)->setForeground(QBrush(QColor(255, 0, 0))); //设置第0
列字的颜色
        model->item(i,0)->setTextAlignment(Qt::AlignCenter); //设置对齐
        model->item(i,2)->setTextAlignment(Qt::AlignCenter);
        model->item(i,3)->setTextAlignment(Qt::AlignCenter);

    }
    ui->RecentBorrowTable->setModel(model);//为表格应用model
    ui->RecentBorrowTable->horizontalHeader()-
>setDefaultAlignment(Qt::AlignCenter);//表头居中对齐
    ui->RecentBorrowTable->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);//所有列都扩展自适应宽度，填充充满整个
屏幕宽度
```

```
    ui->RecentBorrowTable->setColumnWidth(0, 100);//设置固定宽度
    ui->RecentBorrowTable-
>setSelectionBehavior(QAbstractItemView::SelectRows);//设置选中模式为选中行
    ui->RecentBorrowTable-
>setEditTriggers(QAbstractItemView::NoEditTriggers);//设置表格只读
}
```

14. **管理员界面-----初始化树形视图**

因为`g_vBookInfo`中已经保存了从数据库中读取的信息，并且具有树状结构，所以这里可以直接对`g_vBookInfo`进行遍历，将对应的父结点提取作为树形视图的一级结点，将父结点对应的子结点作为一级结点下的二级结点。还需要注意对一些字段进行判断，例如当借阅状态==1时，说明这本书当前可借，之后的借阅日期和借书证号直接插入空值。

```
/* 初始化树形视图 */
void AdminMode::InitializeTree()
{
    MainWindow::InitializeBookInfo(); //更新全局书籍信息
    ui->treeView->header()->setDefaultAlignment(Qt::AlignCenter);  //设置居中
    ui->treeView->header()->setStretchLastSection(true);  //设置其他列自适应列宽
    ui->treeView->setEditTriggers(QTreeView::NoEditTriggers);            //单
元格不能编辑
    ui->treeView->setSelectionBehavior(QTreeView::SelectRows);           //一
次选中整行
    ui->treeView->setAlternatingRowColors(true);                    //每间隔一行
颜色不一样，当有qss时该属性无效

    m_Model = new QStandardItemModel(ui->treeView);
    //设置表头
    m_Model->setHorizontalHeaderLabels(QStringList()<<QStringLiteral("登记
号") << QStringLiteral("书名")<<QStringLiteral("作者")<<QStringLiteral("库存
数")<<QStringLiteral("ISBN")
                                       <<QStringLiteral("出版日期")
<<QStringLiteral("出版社")<<QStringLiteral("借阅状态")<<QStringLiteral("借阅日
期")<<QStringLiteral("借书证号"));      //设置列头
    for(int i=0;i<g_vBookInfo.size();++i) //添加一级结点，即父结点
    {
        QList<QStandardItem*> t_itemsLv1;
        QVector<QString> t_vItem0;  //暂存每一列的信息
        t_vItem0.append(g_vBookInfo[i].m_strRegistrationNumber);
        t_vItem0.append(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strBookName);
        t_vItem0.append(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strAuthor);

 t_vItem0.append(QString::number(g_vBookInfo[i].m_nRealInventory)+"/"+QStrin
g::number(g_vBookInfo[i].m_nInventory));
        t_vItem0.append(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strISBN);

 t_vItem0.append(g_vBookInfo[i].m_BookInfo[0].m_Book.m_dtPublicationDate.toS
tring("yyyy.MM.dd"));

 t_vItem0.append(g_vBookInfo[i].m_BookInfo[0].m_Book.m_strPublishingHouse);

        for(int p=0;p<3;++p)
            t_vItem0.append(""); //因为是父结点项，后面的3列（书籍的具体借阅信息）都
不需要添加内容
        QStandardItem* t_item0=new QStandardItem(t_vItem0[0]);
        t_itemsLv1.append(t_item0);
```

```
        for(int m=1;m<t_vItem0.size();++m)
        {
            QStandardItem* t_item=new QStandardItem(t_vItem0[m]);
            t_itemsLv1.append(t_item);
        }
        t_vItem0.clear(); //清空暂存变量
        m_Model->appendRow(t_itemsLv1); //插入一个完整的一级结点到树形视图中

        for(int j=0;j<g_vBookInfo[i].m_BookInfo.size();++j) //添加二级结点
        {


            QList<QStandardItem*> t_itemsLv2;
            QVector<QString> t_vItem;

 t_vItem.append(g_vBookInfo[i].m_BookInfo[j].m_Book.m_strRegistrationNumber)
;

 t_vItem.append(g_vBookInfo[i].m_BookInfo[j].m_Book.m_strBookName);
            t_vItem.append(g_vBookInfo[i].m_BookInfo[j].m_Book.m_strAuthor);

            for(int i=0;i<4;i++)  //因为是二级结点，中间的一些列（书籍自身的具体信
息，在父结点可以体现）不用填写
                t_vItem.append("");
            if(g_vBookInfo[i].m_BookInfo[j].m_bIsBorrowed==1) //判断每一本书籍
的借阅状态
            {
                t_vItem.append("可借");
                t_vItem.append("");
            }
            else
            {
                t_vItem.append("已借出");

 t_vItem.append(g_vBookInfo[i].m_BookInfo[j].m_dtBorrowingDate.toString("yyy
y.MM.dd"));
            }

 t_vItem.append(g_vBookInfo[i].m_BookInfo[j].m_strLibraryCardNumber);

            for(int m=0;m<t_vItem.size();++m)
            {
                QStandardItem* t_item=new QStandardItem(t_vItem[m]);
                t_itemsLv2.append(t_item);
            }
            t_vItem.clear(); //清空暂存变量
            t_item0->appendRow(t_itemsLv2); //插入一个完整的二级结点到树形视图中
        }
    }

    ui->treeView->setModel(m_Model); //为树形视图应用上面得到的model
    ui->treeView->setColumnWidth(1,180); //设置第一列宽度（书名那一列）
    setContextMenuPolicy(Qt::CustomContextMenu);            //设置treeView支持
右键弹出菜单
    connect(this,SIGNAL(customContextMenuRequested(const QPoint &)),this,
            SLOT(slotCustomContextMenu(const QPoint &)));  //连接点击右键信号与
槽函数
```

```
}
```

15. **管理员界面-----获取选中条目的信息**

    由于树形视图比较复杂，这里把获取选中条目单独作为一个函数，供其他函数调用。

```
/* 获取当前选中条目的信息 */
void AdminMode::on_treeView_clicked(const QModelIndex &index)
{

 m_SelectBook.m_strRegistrationNumber=index.sibling(index.row(),0).data().to
String(); //获取该行第0列的内容

 m_SelectBook.m_strBookName=index.sibling(index.row(),1).data().toString();
//获取该行第1列的内容，下面类似
    m_SelectBook.m_strAuthor=index.sibling(index.row(),2).data().toString();
    m_SelectBook.m_strISBN=index.sibling(index.row(),4).data().toString();

 m_SelectBook.m_dtPublicationDate=index.sibling(index.row(),5).data().toDate
();

 m_SelectBook.m_strPublishingHouse=index.sibling(index.row(),6).data().toStr
ing();
}
```

16. **管理员界面-----创建新书籍**

    新建一本书和新建多本同种书统一在一起，可以通过调节数量控制。难点是新建的书籍是全新的还是已有的。如果全新的，那么登记号可以从.1开始自动递增，直到满足数量要求；如果是已有的，需要去将已有的登记号保存下来，新建的时候跳过这些登记号，避免重复。

```
/* 创建新书籍，当数量为1时，相当于创建单本书籍；大于1时，批量创建 */
void CreateItem::on_SaveButton_clicked()
{
//确保输入的新书籍的各项基本信息不为空
    if(ui->RegNumEdit->text()==""||ui->BookNameEdit->text()==""||ui-
>AuthorEdit->text()==""|| \
            ui->ISBNEdit->text()==""||ui->PubHouseEdit->text()==""||ui-
>PubDateEdit->text()=="")
    {
        QMessageBox::information(this, "提示","存在非法输入，创建失败！");
        return;
    }
    else
    {   //获取各项信息
        QString t_strRegNum=ui->RegNumEdit->text();
        QString t_strBookName=ui->BookNameEdit->text();
        QString t_strAuthor=ui->AuthorEdit->text();
        QString t_strISBN=ui->ISBNEdit->text();
        QString t_strPubHouse=ui->PubHouseEdit->text();
        QString t_strPubDate=ui->PubDateEdit->text();
        int t_nInventory=ui->InventoryEdit->value();
        QSqlQuery sqlQuery1,sqlQuery2;
        sqlQuery1.exec(QString("SELECT 登记号 FROM BookInfo WHERE
ISBN='%1'").arg(t_strISBN)); //根据ISBN查询登记号
        sqlQuery1.next();
```

```
        sqlQuery2.exec(QString("SELECT COUNT(登记号) AS nums FROM BookInfo
WHERE ISBN='%1'").arg(t_strISBN)); //根据ISBN查询登记号的数量
        sqlQuery2.next();
        int t_nSize=sqlQuery2.value(0).toInt();
        if(t_nSize==0) //查询结果为0，说明数据库中没有这类书，是“新书”，登记号可以从.1
开始
        {
            for(int i=0;i<t_nInventory;++i)
            {
                //向数据库中插入数据，登记号从.1开始递增
                QString t_strCmd=QString("INSERT INTO BookInfo
VALUES('%1','%2','%3','%4','%5','%6',1,'','')").arg(t_strRegNum+'.'+QString:
:number(i+1)) .arg(t_strBookName).arg(t_strAuthor)\

.arg(t_strISBN).arg(t_strPubHouse).arg(t_strPubDate);
                QSqlQuery sqlQuery;
                if(! sqlQuery.exec(t_strCmd))
                {
                    QMessageBox::information(this, "提示","创建失败！");
                    break;
                }
                if(i==t_nInventory-1)
                    QMessageBox::information(this, "提示","创建成功！");
            }
        }
        else//数据库中有记录，说明新建的是“旧书”，登记号不能从.1开始自动递增
        {
            QVector<QString> t_vCurrentRegNum;//保存该种书当前已有的登记号
            for(int i=0;i<t_nSize;++i)
            {

 t_vCurrentRegNum.append(sqlQuery1.value(0).toString().section(".",1,1)); //
把已有的登记号提取出，以便去重
                sqlQuery1.next();
            }
            int t_nCount=1;//遍历，去和已有登记号比对
            int t_nRealCount=0; //统计可用登记号的个数
            QVector<QString> t_vInsertRegNum;//保存可以新建的登记号
            while(1)
            {
                int t_nFlag=1; //查看该登记号是否已有
                for(int i=0;i<t_vCurrentRegNum.size();++i)
                {
                    if(QString::number(t_nCount)==t_vCurrentRegNum[i])
                    {
                        t_nFlag=0;
                        break;
                    }
                }
                if(t_nFlag==1) //找到了新的登记号
                {
                    t_vInsertRegNum.append(QString::number(t_nCount)); //保存
新的可用登记号
                    t_nRealCount++;
                }
                t_nCount++;
                if(t_nRealCount==t_nInventory)
                    break;
```

```
            }
            for(int i=0;i<t_nInventory;++i) //利用新的可用登记号，向数据库中插入新
记录
            {
                QString t_strCmd=QString("INSERT INTO BookInfo
VALUES('%1','%2','%3','%4','%5','%6',1,'','')").arg(t_strRegNum+'.'+t_vInser
tRegNum[i]) .arg(t_strBookName).arg(t_strAuthor)\

.arg(t_strISBN).arg(t_strPubHouse).arg(t_strPubDate);
                QSqlQuery sqlQuery;
                if(! sqlQuery.exec(t_strCmd))
                {
                    QMessageBox::information(this, "提示","创建失败！");
                    break;
                }
                if(i==t_nInventory-1)
                    QMessageBox::information(this, "提示","创建成功！");
            }

        }
        this->close();
    }
}
```

17. **管理员界面-----右键菜单选项**

    因为树形视图相比表格结构比较复杂，管理和操作也增加了难度。这里采用了右键菜单（编辑--删除--新建--复制（全部信息，ISBN）），针对当前选中的书籍进行操作。

```
/* 右键菜单实现 */
void AdminMode::slotCustomContextMenu(const QPoint &point)
{
    if(ui->treeView->hasFocus())    //使鼠标在treeView里右键才弹出菜单
    {
        QMenu menu;
        QString qss = "QMenu{color:#E8E8E8;background:#4D4D4D;margin:2px;}\
                QMenu::item{padding:3px 20px 3px 20px;}\
                QMenu::indicator{width:13px;height:13px;}\
                QMenu::item:selected{color:#E8E8E8;border:0px solid
#575757;background:#1E90FF;}\
                QMenu::separator{height:1px;background:#757575;}";      //设
置样式表
                menu.setStyleSheet(qss);
    //给菜单设置样式

        menu.addAction( QStringLiteral("编辑"), this, SLOT(slotEditItem()));
     //设置菜单
        menu.addAction(QStringLiteral("删除"), this, SLOT(slotDeleteItem()));
        menu.addAction(QStringLiteral("新建"), this, SLOT(slotCreateItem()));

        QAction* actionParent = menu.addAction(QStringLiteral("复制"));    //
父菜单
        QMenu* subMenu = new QMenu(&menu);  //子菜单
        subMenu->addAction(QStringLiteral("全部信息"), this,
SLOT(slotCopyBookInfo()));
        subMenu->addAction(QStringLiteral("ISBN"), this,
SLOT(slotCopyBookISBN()));
```

```
        actionParent->setMenu(subMenu); //应用子菜单
        menu.exec(this->mapToGlobal(point));//应用右键菜单
    }
}
```

18. **管理员界面-----查找**

    树形视图的查找无法像表格视图那样简单遍历，也无法通过隐藏行来实现搜索结果的效果。查阅资料后，发现设置代理model进行关键词过滤可以实现。因为是树形视图中所有行所有列进行匹配，所以这里的过滤列参数设置为-1，代表所有列都参与过滤。

```
/* 管理员界面查找 */
void AdminMode::on_FindEdit_2_textChanged(const QString &arg1)
{
    QString t_strFind=ui->FindEdit_2->text(); //获取查找文本框的内容

    m_ProxyModel = new QSortFilterProxyModel; //设置代理model
    m_ProxyModel->setSourceModel(m_Model); //设置源model
    m_ProxyModel->setFilterKeyColumn(-1);//设置过滤列为所有列
    ui->treeView->setModel(m_ProxyModel);//应用到树形视图
    QRegExp regExp(t_strFind, Qt::CaseInsensitive, QRegExp::FixedString);
    m_ProxyModel->setFilterRegExp(regExp);//对关键词进行查找过滤
}
```

19. **管理员界面-----删除书籍**

    删除书籍包括删除同种所有书籍和删除单本书籍，根据当前选中的是一级结点还是二级结点区分。如果当前选中行的ISBN信息为空，说明是二级结点，只需根据具体登记号查询数据库，删除单条记录即可；如果选中行的ISBN信息不为空，说明是一级结点，为了删除同种所有书籍，可以用ISBN号查询数据库，删除查找结果中的所有的记录，实现批量删除。

```
/* 删除书籍 */
void AdminMode::slotDeleteItem()
{
    int t_nWarning = QMessageBox::warning(this,tr("删除"),tr("你确定删除当前书籍
吗？"),QMessageBox::Yes,QMessageBox::No);
    if(t_nWarning==QMessageBox::No)
        return;
    else
    {
        QString deleteSql="";
        if(m_SelectBook.m_strISBN=="")//如果ISBN为空，说明是二级子结点，选中和删除
的是单本书籍
        {
            deleteSql = QString ("DELETE FROM BookInfo WHERE 登记号 =
%1").arg(m_SelectBook.m_strRegistrationNumber);
        }
        else//否则，选中的是一级父结点，删除同种所有书籍
        {
            deleteSql = QString ("DELETE FROM BookInfo WHERE ISBN =
%1").arg(m_SelectBook.m_strISBN);
        }

        // 执行插入的sql语句
        QSqlQuery query;
        query.exec (deleteSql);
        QMessageBox::information(this, "提示","删除成功");
```

```
        MainWindow::InitializeBookInfo();
        InitializeTree(); //重新初始化树形视图
    }
}
```

20. **管理员界面-----编辑单本书籍**

    先判断各字段输入值是否合法，之后通过获取当前选中书籍的登记号查询数据库，用查询到的信息初始化编辑框。由于是单本书籍，故所有基本信息和管理信息都可以修改，同时需要判断借阅状态是否为可借，如为可借，借阅日期和借书证号编辑框变为不可编辑状态。

```
/* 保存编辑后的信息 */
void EditItem::on_SaveButton_clicked()
{
    //确保各项输入非空
    if(ui->RegNumEdit->text()==""||ui->BookNameEdit->text()==""||ui-
>AuthorEdit->text()==""|| \
            ui->ISBNEdit->text()==""||ui->PubHouseEdit->text()==""||ui-
>PubDateEdit->text()=="")
    {
        QMessageBox::information(this, "提示","存在非法输入，编辑失败！");
        return;
    }
    else
    {
        //从控件获取各字段的值
        QString t_strRegNum=ui->RegNumEdit->text();
        QString t_strBookName=ui->BookNameEdit->text();
        QString t_strAuthor=ui->AuthorEdit->text();
        QString t_strISBN=ui->ISBNEdit->text();
        QString t_strPubHouse=ui->PubHouseEdit->text();
        QString t_strPubDate=ui->PubDateEdit->text();
        bool t_bBorrowState=ui->BorrowState->currentIndex();
        QString t_strBorrowDate="";
        QString t_strLibraryCard="";

        if(t_bBorrowState==0)//借出状态下再设置借阅日期和借书证号
        {
            t_strBorrowDate=ui->BorrowDateEdit->text();
            t_strLibraryCard=ui->LibraryCard->text();
            QSqlQuery sqlQuery;
            //更新数据库中该书的借阅日期
            sqlQuery.exec(QString("UPDATE UserInfo SET
BookHistory=BookHistory||'%1' WHERE
Username='%2'").arg(t_strRegNum+";").arg(t_strLibraryCard));
            //更新数据库中该书的借书证号
            sqlQuery.exec(QString("UPDATE UserInfo SET
BorrowTime=BorrowTime||'%1' WHERE
Username='%2'").arg(QDate::fromString(t_strBorrowDate,"yyyyMMdd").toString("
yyyy.MM.dd")+";").arg(t_strLibraryCard));

        }
        //更新数据库中该书的其他字段
        QString t_strCmd=QString("UPDATE BookInfo SET 登记号='%1',书名='%2',作
者='%3',ISBN='%4',出版社='%5',出版日期='%6',借阅状态='%7',借阅日期='%8',借书证号
='%9' WHERE 登记号= '%10'").arg(t_strRegNum)
.arg(t_strBookName).arg(t_strAuthor)\
```

```
.arg(t_strISBN).arg(t_strPubHouse).arg(t_strPubDate).arg(t_bBorrowState).arg
(t_strBorrowDate).arg(t_strLibraryCard).arg(m_strSelectedEditItem);
        QSqlQuery sqlQuery;
        if(! sqlQuery.exec(t_strCmd))
        {
            QMessageBox::information(this, "提示","编辑失败！");
        }
        else
        {
            QMessageBox::information(this, "提示","编辑成功！");
        }
    }
    this->close();
}
```

21. **管理员界面-----编辑同种所有书籍**

    先判断各字段输入值是否合法，之后通过ISBN号查询数据库，对查询到的多条结果统一修改。因为是同种所有书籍，主要批量修改书籍的基本信息（登记号、书名、作者、ISBN号等等）。

```
/* 保存编辑后的信息 */
void EditBook::on_SaveButton_clicked()
{
    //确保书籍各项基本信息输入非空
    if(ui->RegNumEdit->text()==""||ui->BookNameEdit->text()==""||ui-
>AuthorEdit->text()==""|| \
            ui->ISBNEdit->text()==""||ui->PubHouseEdit->text()==""||ui-
>PubDateEdit->text()=="")
    {
        QMessageBox::information(this, "提示","存在非法输入，编辑失败！");
        return;
    }
    else
    {
        //获取各字段的值
        QString t_strRegNum=ui->RegNumEdit->text();
        QString t_strBookName=ui->BookNameEdit->text();
        QString t_strAuthor=ui->AuthorEdit->text();
        QString t_strISBN=ui->ISBNEdit->text();
        QString t_strPubHouse=ui->PubHouseEdit->text();
        QString t_strPubDate=ui->PubDateEdit->text();
        QVector<QString> m_vRegNum;//保存编辑前的登记号
        QSqlQuery sqlQuery;
        //根据选中书籍的ISBN号查询登记号
        sqlQuery.exec(QString("SELECT 登记号 FROM BookInfo WHERE
ISBN='%1'").arg(m_strSelectedEditISBN));
        while(sqlQuery.next())
        {
            m_vRegNum.append(sqlQuery.value(0).toString()); //获取编辑前的登记
号
        }
        for(int j=0;j<m_vRegNum.size();++j)
        {
            //将新的登记号前缀和旧的登记号后缀组合，更新数据库
            sqlQuery.exec(QString("UPDATE BookInfo SET 登记号='%1' WHERE 登记
号='%2'").arg(t_strRegNum+'.'+m_vRegNum[j].section('.',1,1))
```

```
                            .arg(m_vRegNum[j]));
        }
        //更新书籍的其他字段
        QString t_strCmd=QString("UPDATE BookInfo SET 书名='%1',作者
='%2',ISBN='%3',出版社='%4',出版日期='%5' WHERE ISBN=
'%6'").arg(t_strBookName).arg(t_strAuthor)\

.arg(t_strISBN).arg(t_strPubHouse).arg(t_strPubDate).arg(m_strSelectedEditIS
BN);
        if(! sqlQuery.exec(t_strCmd))
        {
            QMessageBox::information(this, "提示","编辑失败！");
        }
        else
        {
            QMessageBox::information(this, "提示","编辑成功！");
        }

    }
    this->close();
}
```

四、测试报告

1. **登录**

   前置条件：

   - 错误的/不存在的普通用户账号
   - 正确的普通用户账号和密码
   - 正确的管理员账号（`admin`）和密码（`admin`）

   操作：

   - 注册账号
   - 以用户身份登录
   - 以管理员身份登录

   预期结果：

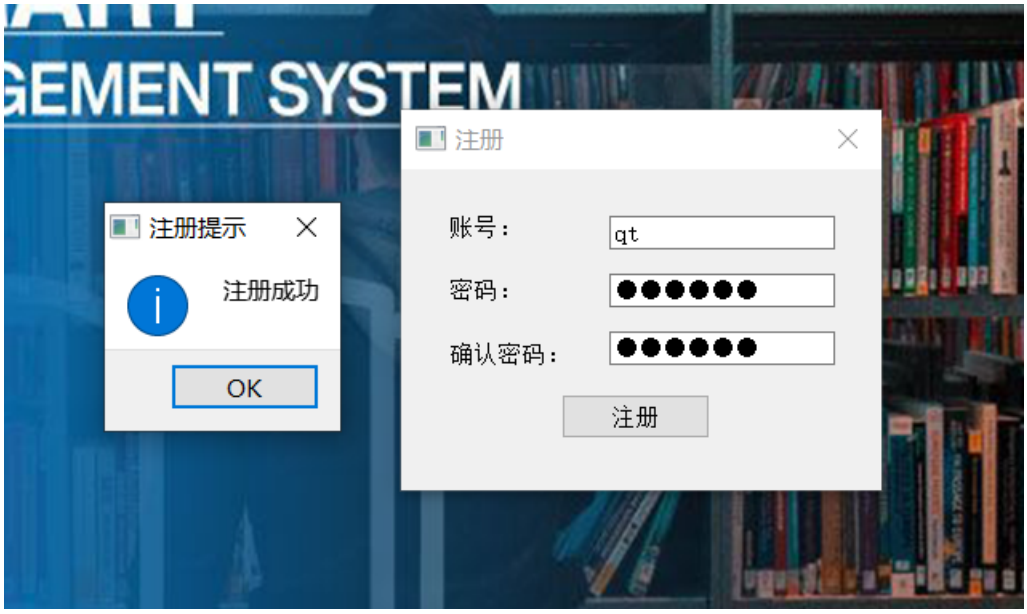
   - 当用户账号不存在时，登录错误
   - 当输入正确的用户名（普通用户/管理员）和密码时登录成功

   测试结果：

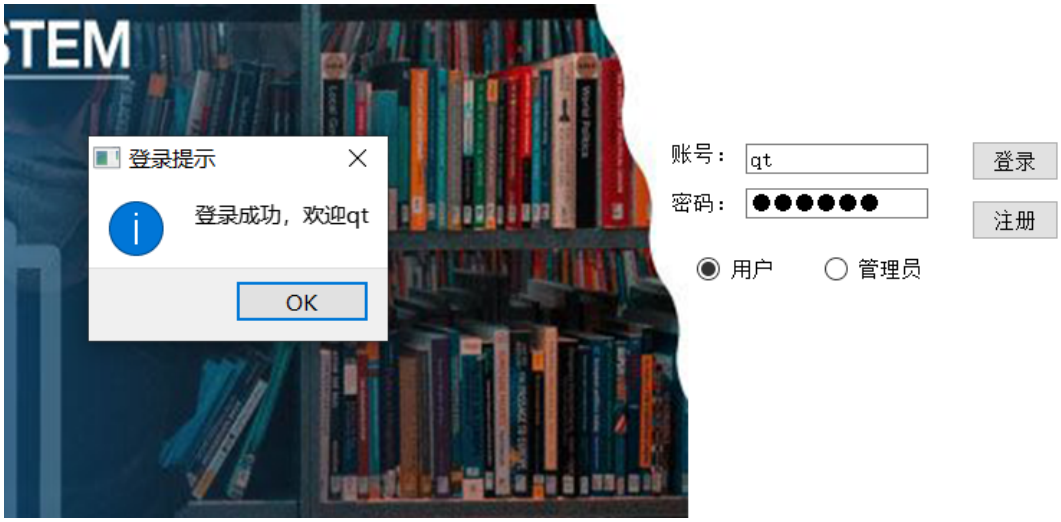
   - 登录失败：

   - 注册成功：



- 普通用户登录成功：



- 管理员登录成功：

2. **借阅：**

   前置条件：

   - 正常的已登录的普通用户账号
   - 已被借阅完的书籍
   - 可借的书籍

   操作：选中表格项，借阅图书

   预测结果：

- 可借的书籍正常借阅
- 已被借阅完的书籍提示借完

测试结果：

- 借阅已经借完的书籍：

- 借阅可借的书籍：

3. **还书：**

前置条件：

- 已借书的用户

操作：

- 通过选中表格还书
- 通过右下角登记号快速还书

预测结果：

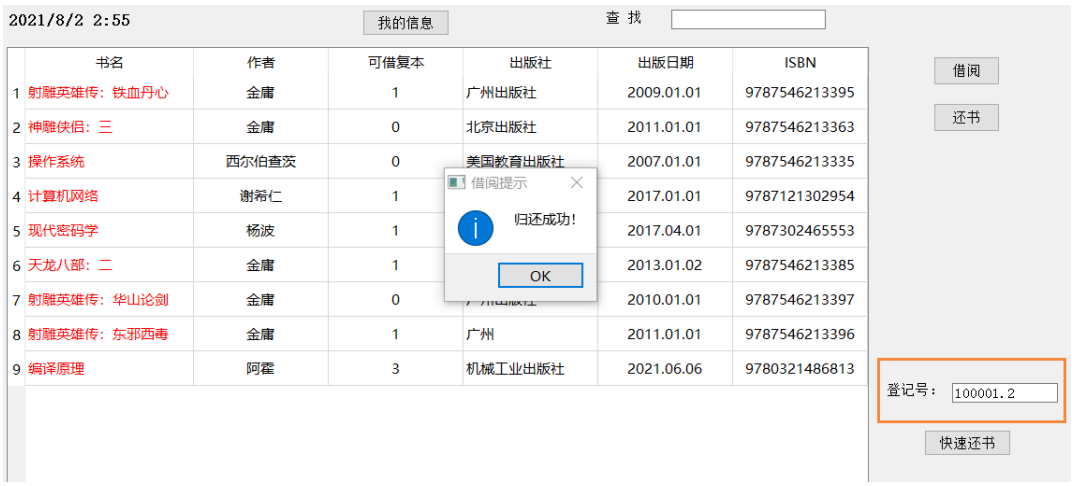
- 归还借阅且未还的书成功
- 归还未曾借阅的书失败
- 找不到登记号

测试结果：

- 正常归还：

- 归还未借过的图书：

- 利用登记号，快速还书：



- 查看我的信息：

	登记号	书名	ISBN	借阅状态	借阅时间
1	100004.2	计算机网络	9787121302954	已还	2021.08.02 03:03:29
2	100001.2	射雕英雄传：铁血丹心	9787546213395	已还	2021.08.02 03:02:50
3	100001.1	射雕英雄传：铁血丹心	9787546213395	未还	2021.08.02 03:02:46
4	100004.2	计算机网络	9787121302954	已还	2021.08.02 02:56:28

4. **用户界面查找书籍**

   前置条件：

   - 登录用户系统

操作：在查找文本框输入文字

预期结果：

- 若查找成功，则显示匹配到的书籍条目
- 若查找失败，搜索结果为空

测试结果：

- 查找成功：

- 查找失败：

5. **新建书籍**

前置条件：

- 成功登录管理员系统

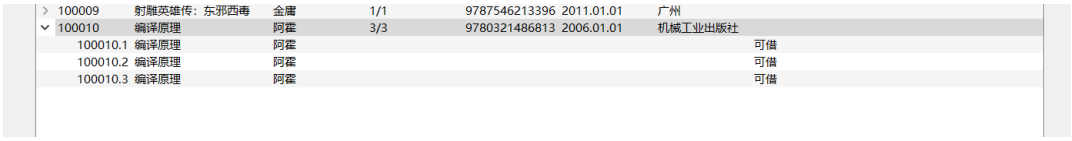
操作：打开右键菜单----新建

预期结果：

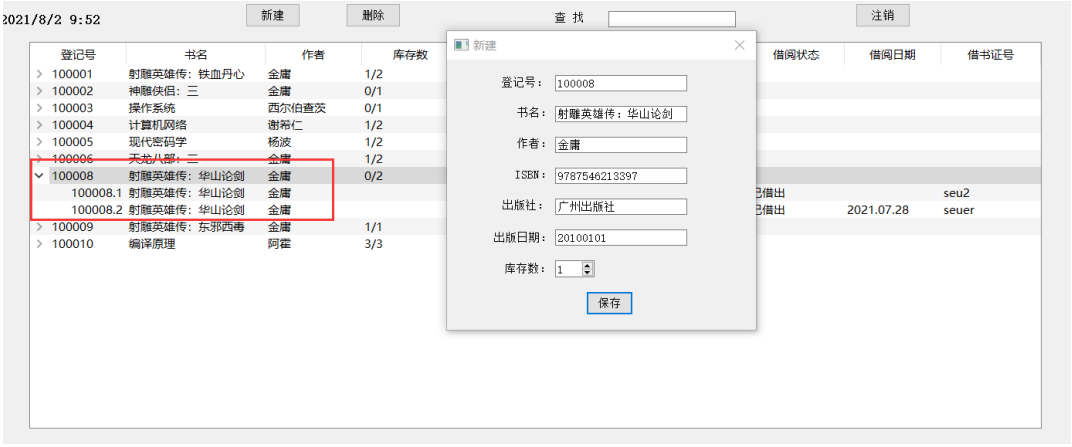
- 各项信息完整时新建成功
- 缺少信息时创建失败

测试结果：

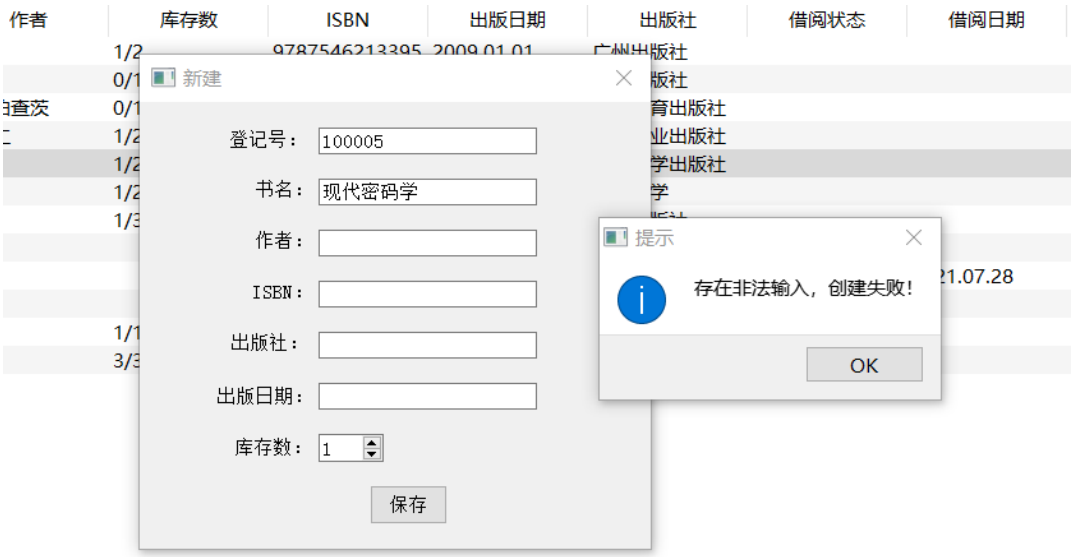
- 创建多本同种新书籍成功



- 新增单本已有书籍



- 信息输入不完整，新建失败



6. **编辑书籍**

前置条件：

- 成功登录管理员系统

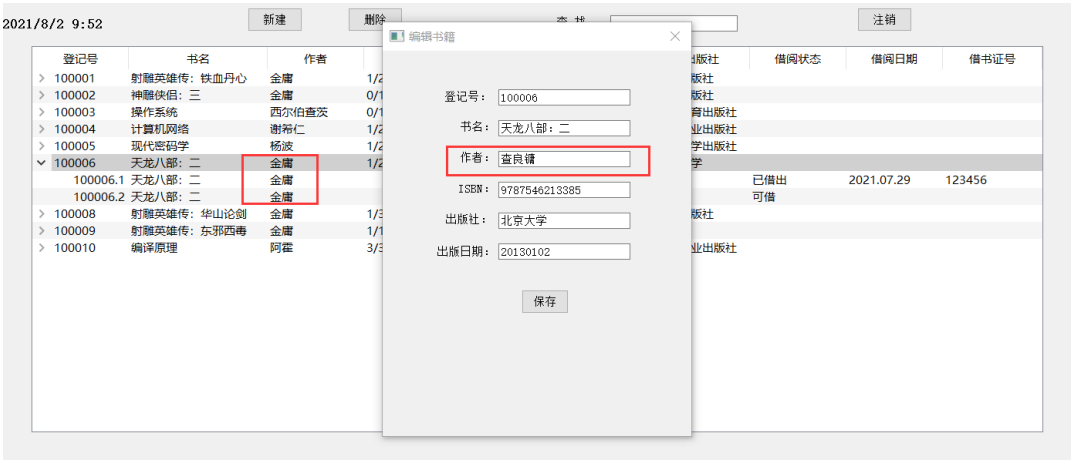
操作：选中书籍结点（父结点或子结点均可以）-----右键菜单-----编辑

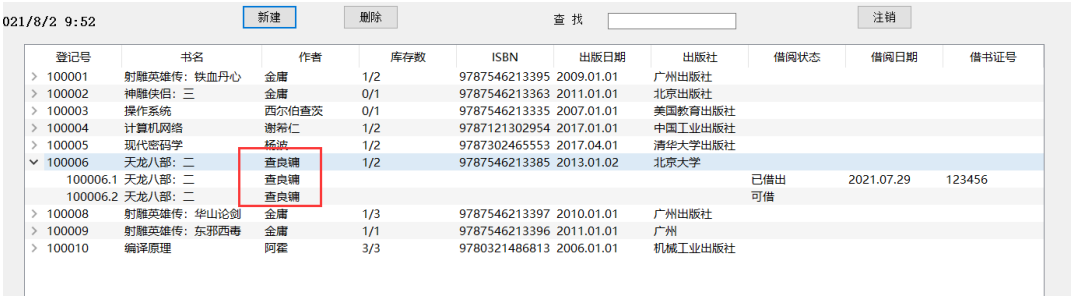
预期结果：

- 各项信息输入完整时编辑成功
- 缺少信息时编辑失败

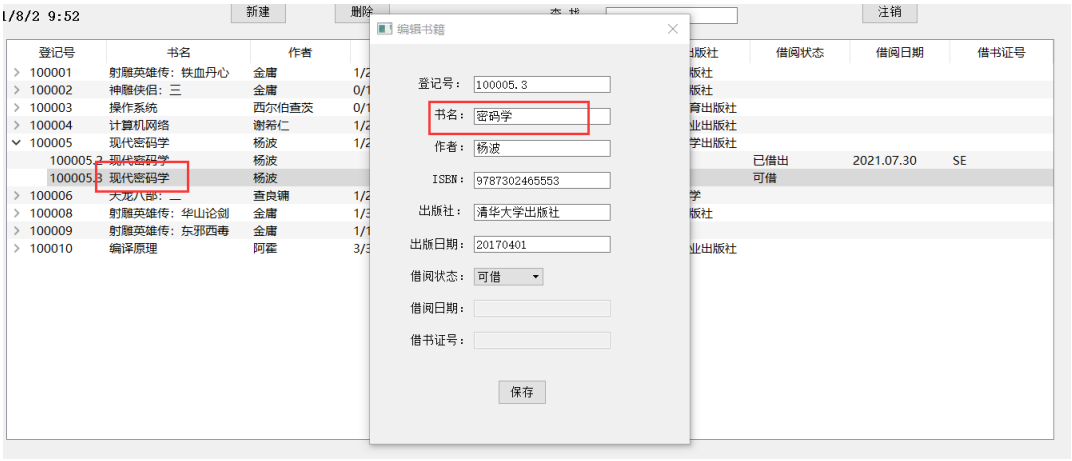
测试结果：

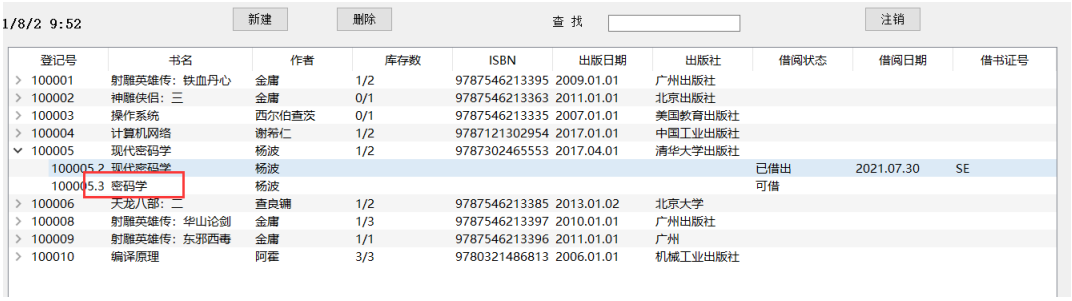
- 成功编辑多本同种书籍：



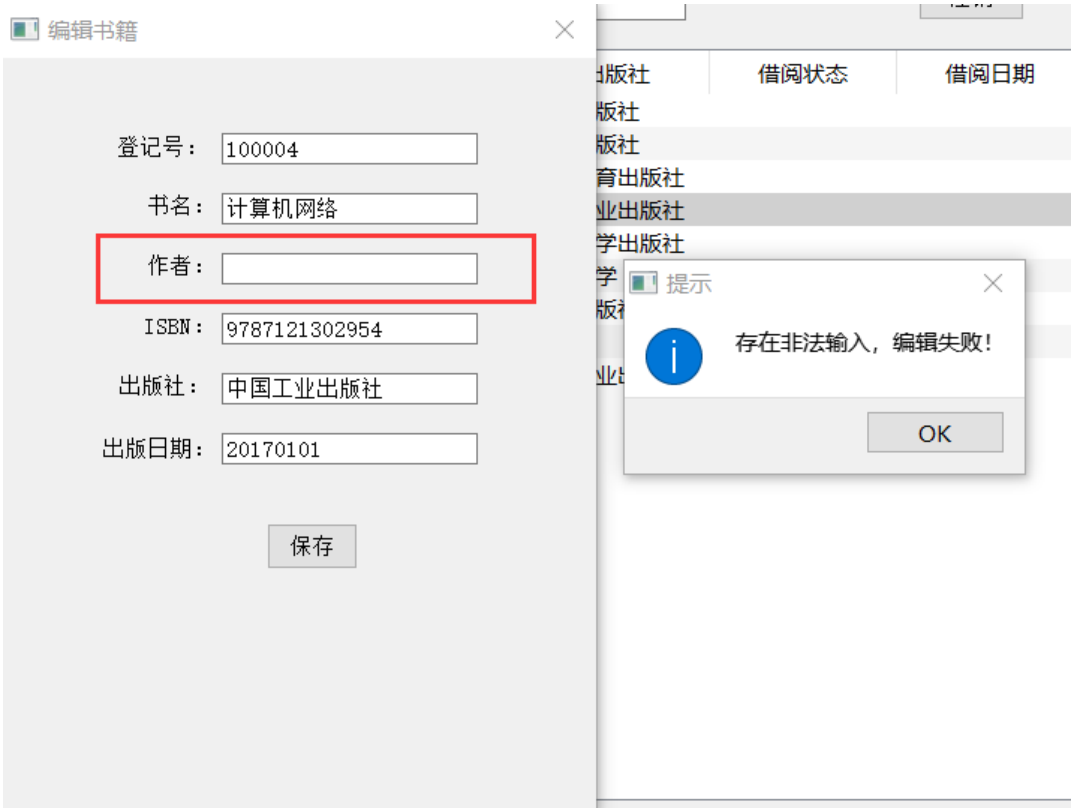


- 成功编辑单本书籍：





- 缺少信息编辑失败



7. **管理员界面查找书籍**

   前置条件：

   - 成功登录管理员系统

   操作：在查找文本框输入文字

   预期结果：

   - 若查找成功，则显示匹配到的书籍条目
   - 若查找失败，搜索结果为空

   测试结果：

   - 查找成功：

   - 查找失败：

8. **删除书籍**

   前置条件：

- 登录管理员系统

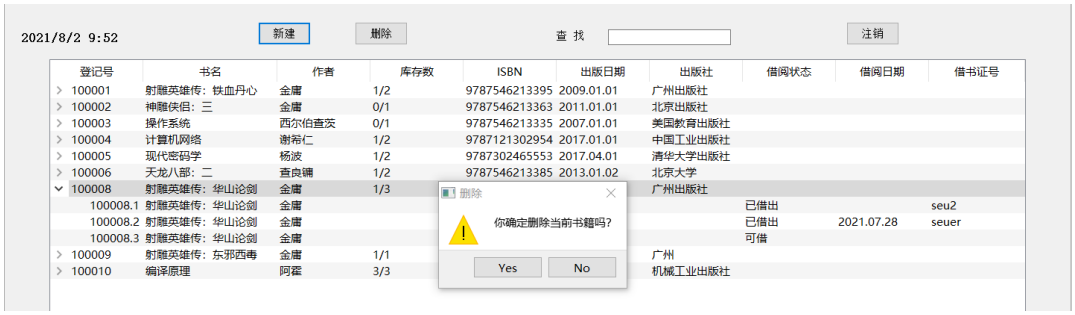
操作：选中书籍结点（父结点或子结点均可以）-----右键菜单-----删除

预期结果：

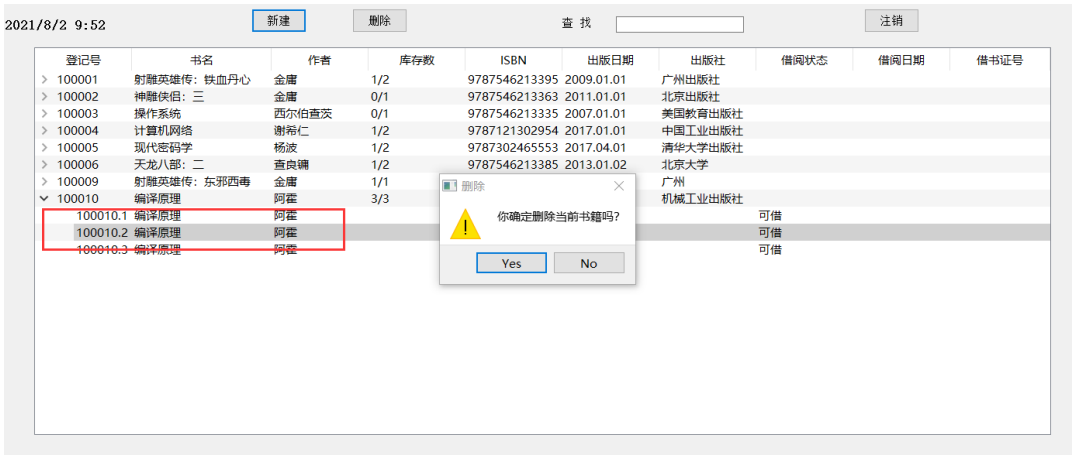
- 选中父结点，成功删除同种所有书籍
- 选中子结点，成功删除单本书籍

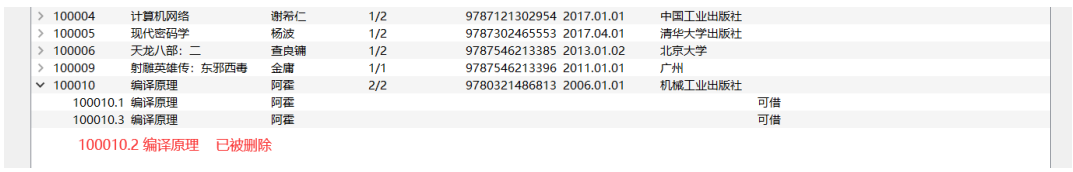
测试结果：

- 成功删除同种所有书籍：



- 成功删除单本书籍：





9. **复制书籍信息**

前置条件：

- 登录管理员系统

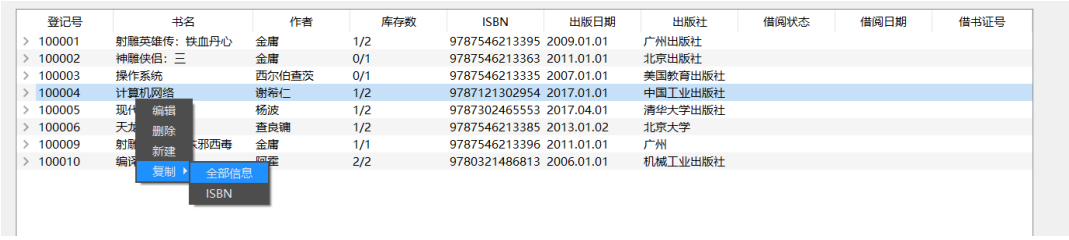
操作：选中书籍结点（父结点或子结点均可以）-----右键菜单-----复制-----全部信息/ISBN

预期结果：

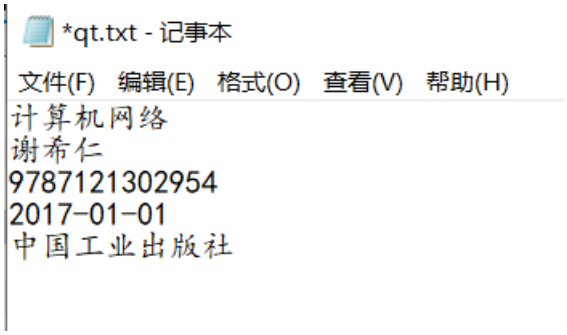
- 成功复制全部信息
- 成功复制ISBN号

测试结果：

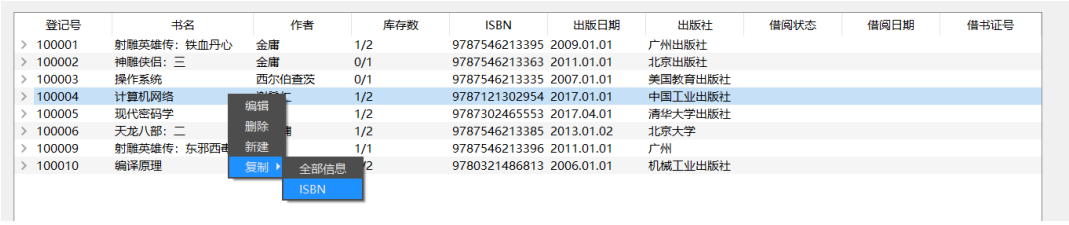
- 成功复制所有信息：



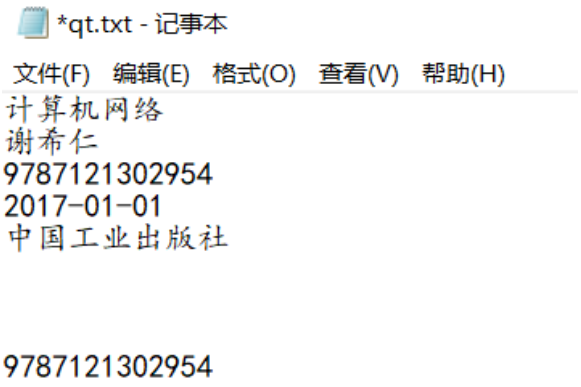
粘贴到记事本：



- 成功复制ISBN号：



粘贴到记事本：



五、暑期学校学习小结

1. 这次补修暑期学校语言程序设计使用的是Qt，相比之前在原学院大一时候用的MFC，深深体会到了Qt的**简单便捷**以及它的**强大功能**，据了解，很多常用的软件比如WPS、YY语音、Mathematic的GUI等等都是基于Qt的。

   总的来说，这次程序设计遇到语法不会的问题确实有，但是只是少部分（数据库和树形视图那部分），而且大部分简单的语法可以直接通过F1查看Qt自带的帮助手册解决。

   更多遇到的问题还是已知代码的**逻辑问题**，这也是与编程形影不离的一大bug。为此，我采用了**设断点**、利用**qDebug**输出一些提示性语句、**主动输出**可能出错的变量等方法，解决了程序中的大部分bug。

2. 这次Qt大作业我尝试使用了一些之前未接触过的技术，比如**数据库**。由于我们还未学习数据库相关理论课程，所以很多知识都要现学现用，在网络上查找有用的资料。但是由于**SQLite**数据库比较小众，相关可视化软件和教程的资料也较少，只能自己参考其他数据库摸索。但是由于SQL语句的通用性，在Qt中实际操作的时候还是比较方便。

3. 除了实现程序的基本功能外，界面的**美观程度**也是一个重要的方面。我查找了关于窗口无边框化的资料，但也发现美化的同时提升了编程的复杂度，需要自定义实现鼠标拖动、窗口最大最小化、关闭等功能。
4. 整个项目有1000多行代码，整个项目的**模块化组织**、**规范命名**和**注释**就显得尤为重要，如果能够坚持这种良好的**编程习惯**将会大大提高编程的效率，避免不必要的浪费和错误。
5. 在编写程序的过程中，我还体会到了**亲身实践**和**积累**的重要性。知识通过实际应用才能进一步掌握并实现其价值。有些算法想的时候感觉简单，实际做起来发现一些细节地方没有处理好从而产生错误，例如**数组下标越界**、**全局变量的组织**等等。计算机编程语言博大精深，仅就C++来看，我目前所掌握的只不过是冰山一角，因此一些常用的算法或功能需要自己去积累，在加深理解的基础上达到掌握的程度。学习本身就是一个不断积累的过程，在以后的学习生活中都应该不断地学习，通过学习来锻炼自己、提高自己。
6. 感谢王老师从我们大二以来的辛苦付出以及老师课下的帮助，老师的讲解使得数据结构和C++不再变得那么抽象，反而充满了乐趣。我相信，在未来的学习以及科研生涯中，**C++**以及学到的**数据结构**和**编程思想**一定会作为一个强大的工具常伴我左右，助我一臂之力。