華中科技大学

课 程 设 计 报 告

题目： 图书管理系统的设计与实现

课程名称： 数据库课程设计

专业班级： 物联网 1501

学　　号： U201514881

姓　　名： 赵浩东

指导教师： 李瑞轩

报告日期： 2017.6.26

计算机科学与技术学院

目　　录

1 系统设计背景.....1
2 系统需求分析.....2
2.1 功能需求.....2
2.2 数据完整性需求.....3
3 数据库设计.....4
3.1 概念设计.....4
3.2 逻辑结构设计.....5
4 详细设计.....7
4.1 用户登陆页面.....7
4.2 图书、用户及借用信息的添加.....8
4.3 图书、用户及借用信息的删除.....9
4.4 图书、用户及借用信息的修改.....10
4.5 数据查询.....11
5 课程设计小结.....12
6 实验练习.....13
实验 1 基本表的创建、数据插入.....13
实验 2. 数据查询.....16
实验 3. 数据修改、删除.....17
实验 4. 视图的操作.....18
实验 5. 库函数，授权的控制.....19
实验 6. 数据库的备份、恢复.....20
附录（课设代码）.....24

1 系统设计背景

随着现代化图书馆书籍的数目与种类不断增加，丰富的资源在给读者更多的选择的同时却带来信息检索的不便。读者在实际科研工作中，我们常常会因为少量文献的漏查而导致大量的重复工作。图书管理员也面临着管理大量丛书力不从心的困境。如何科学、高效地管理图书已成为当今建立图书馆面临的首要问题，而建立信息化的图书管理系统是一所现代化图书馆不可避免的一个环节。

本次课程设计旨在运用数据库课程所学知识，结合软件开发的有关技术，建立一套高效的面向用户与图书管理员的图书管理系统。

2 系统需求分析

2.1 功能需求

添加操作：在执行操作时首先都要先检查是否已经输入以及数据库中是否已经存在输入的数据，如果存在会报错，程序返回上个页面，不执行添加操作。

删除操作：考虑到了字段在不同表中可能会同时存在的情况，比如：删除图书分类时候该分类下有书或者删除用户时该用户有借书记录，这样的情况下程序会给出提醒并跳转到该分类下的所有图书查看或者是该用户的全部借书查看。直到这种关联取消掉之后才允许进行删除这样的危险操作。

查询操作：考虑到了模糊查询的实现，SQL 语句中用“like”，还有组合查询，通过对用户输入情况的判断来确定 SQL 语句的最终形式，比如用户查询时并未输入图书名字，就是名字匹配任意字符，那么程序给数据库的 SQL 语句就会是 name like ‘%’。组合查询的实现也是程序判断用户所做的选择或者输入，在 SQL 语句中实现的。

超期统计：假定借书超期的限制是 20 天，借书记录中的所有数据的借书时间如果不在当前服务器时间减去 20 天和当前服务器时间之间，那么这条借书记录就属于超期记录了。由用户借书记录中的用户编号到用户表中查找该用户。

发送消息：往消息表中添加数据，该数据包含用户名，用户超期的图书，超期时间，罚金等相关警告信息。当用户正常登陆主页后，程序自动检查消息表中是否存在该用户的记录，如果有要给出提示，但是没有强制用户必须查看。

数据库备份和还原操作：程序只要告诉 SQL 服务器执行备份和还原 SQL 语句就可以实现，在还原时程序还在使用系统数据库，此时执行还原操作会因为没有排它使用权出现错误，所以应告诉 SQL 服务器使用另外的一个数据库，程序中用的是：use master。还原操作时用 FSO 组件首先检查备份数据库文件是否存在，如果不存在则给出提示回到上个页面。所有条件都允许后则执行还原操作，如果数据库数据比较多，备份和还原的时间都可能会比较长。

2.2 数据完整性需求

2.2.1 实体完整性

数据库包含图书与用户两个实体，其中表 book（图书）的主码为 bID，user（用户）的主码为 ID。当插入一条记录或对主码列进行操作时，关系数据库管理系统将按照实体完整性规则自动进行检查：

（1） 主码是否唯一，如不唯一则拒绝插入（NO ACTION）。

（2） 主码列是否存在空值，若存在则拒绝插入或修改。

2.2.2 参照完整性

表 borrow（借用）中外码为表 book 中的 bID 以及表 user 中的 ID。

表 return（归还）中的外码与表 borrow 的一致。

若破坏完整性，则进行违约处理。

3 数据库设计

3.1 概念设计

图书馆管理系统，总共含有两个实体：图书，读者，之间的关系如下图的 E-R 图 3-1 所示：

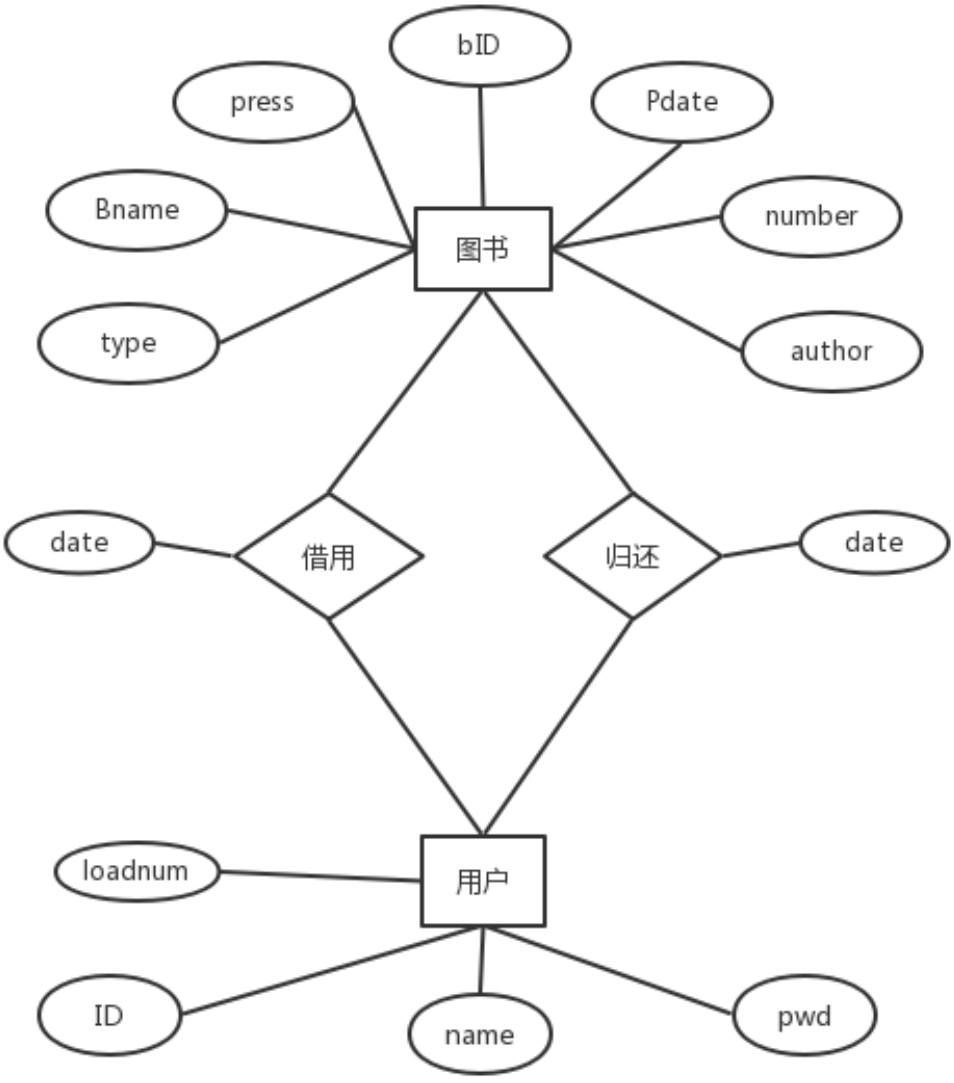

图 3-1 图书管理系统 E-R 图

3.2 逻辑结构设计

表 3.2.1 管理员

字段	类型	长度	必填字段	允许空格	备注
ID	int				id 主键
name	char	50	是	否	名字
pwd	char	50	是	否	密码

表 3.2.2 图书信息

字段	类型	长度	必填字段	允许空值	备注
id	int			否	id 主键
booktype	char	50	是	否	图书类别
publishing	char	50	是	否	出版社
bookmoney	int	20	是	否	货币
pdate	日期时间		是	否	出版时间
bookname	char	50	是	否	图书名称
num	int	8	是	否	图书数量

表 3.2.3 图书类别

字段	类型	长度	必填字段	允许空值	备注
id	int				id 主键
booktype	char	50	是	否	图书类别

表 3.2.4 借出信息

字段	类型	长度	必填字段	允许空值	备注
id	int			否	id 主键
bid	int	50	是	否	图书编号
time	日期时间				借书时间
uid	int	20	是	否	用户编号

表 3.2.5 消息表

字段	类型	长度	必填字段	允许空值	备注
id	int			否	id 主键
content	char	500	是	否	消息内容
uid	int	20	是	否	用户编号

表 3.2.6 用户表

字段	类型	长度	必填字段	允许空值	备注
id	int			否	id 主键
name	char	50	是	否	名字
pwd	char	50	是	否	密码
lock	是/否				是否锁定
loan_num	int	20	是	否	借书数量

4 详细设计

4.1 用户登陆页面

该页面用于用户登陆，包括游客，用户和管理员。成功登陆则取得用户的session 用于以后的操作权限认定。程序执行时首先判断用户类型，如果是游客则对 session 赋值为游客类型，然后直接进入主页；如果不是游客则执行查询语句，看数据库中是否存在这样的用户名和密码，根据选择用户类型的不同在不同的表中进行操作，成功则进入主页，否则提示输入错误。

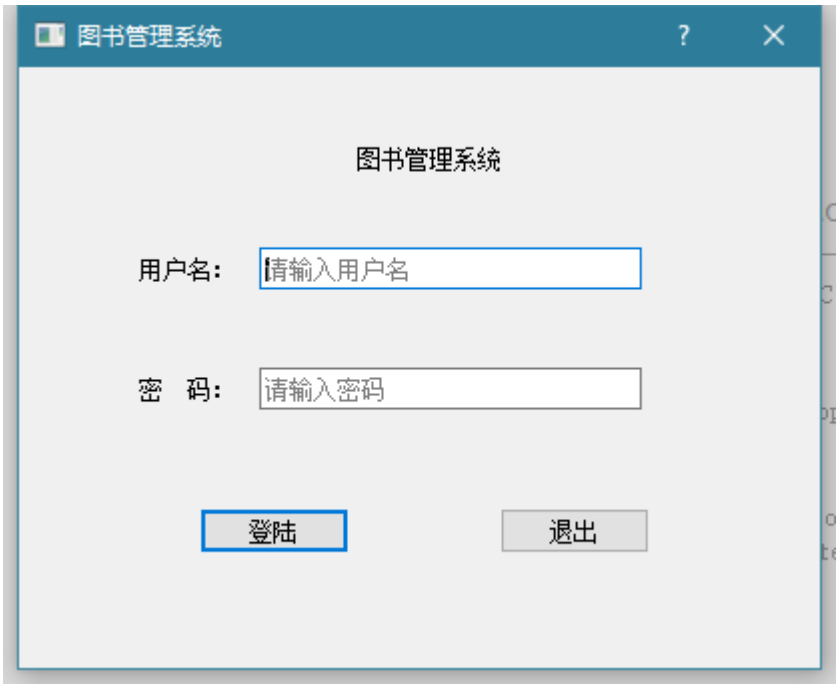

图 4.1.1 用户登陆

由唯一超级管理员账号登陆，可进入管理界面。如图 4.1.2 所示。左上角下拉框选择需要显示的表，点击“显示”按钮即可显示。

图 4.1.2 管理界面

4.2 图书、用户及借用信息的添加

实现表的添加功能。当管理员需要添加某条记录时，通过下拉框选择表显示并点击右边“插入记录”的按钮即可添加新的一行，考虑到输入的方便性图书分类和出版时间都可以进行选择，无须手工输入。添加操作实际上是将数据写入数据库，写入新图书信息之前首先检查图书名是否重复，重复则报错返回上个页面，输入正确而且数据库中原本没有这样的数据则执行写入操作。如图 4.1.3 所示。

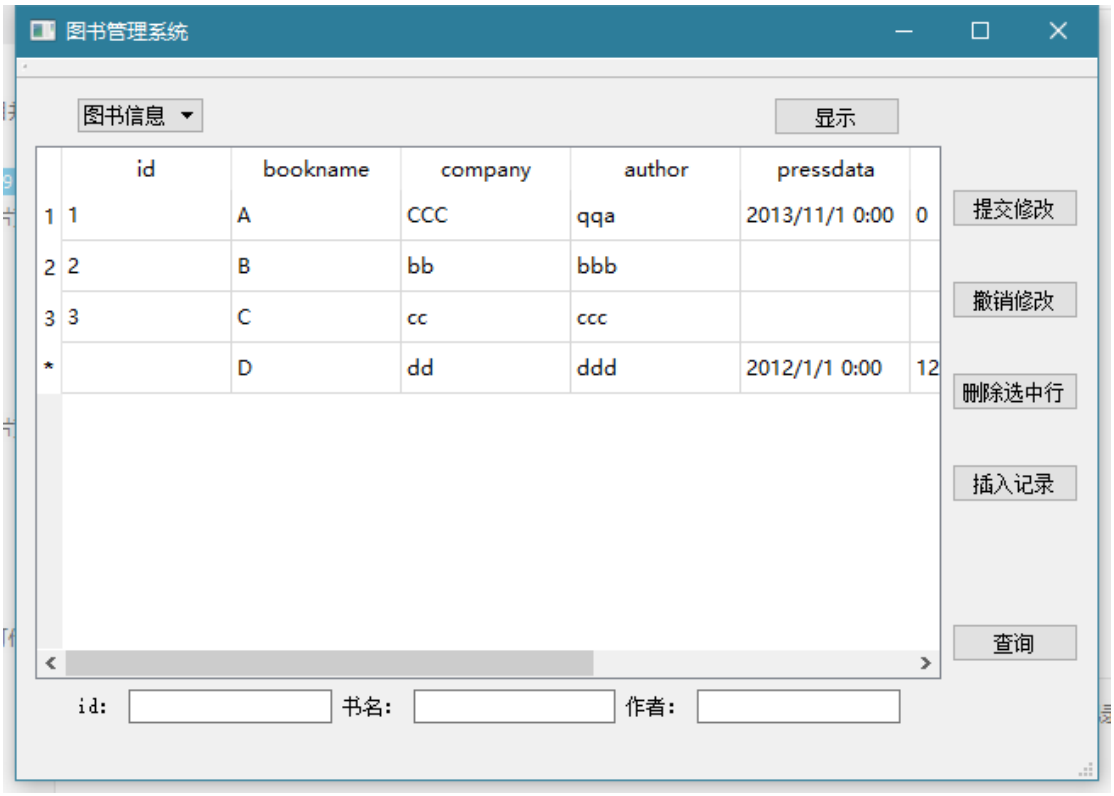

图 4.2.1 插入记录

4.3 图书、用户及借用信息的删除

选中任意表中的一行并点击右侧“删除选中行”的按钮，即可实现删除某条记录的操作。当删除时，会弹出对话窗口决定是否确认删除。如图 4.1.4 所示。

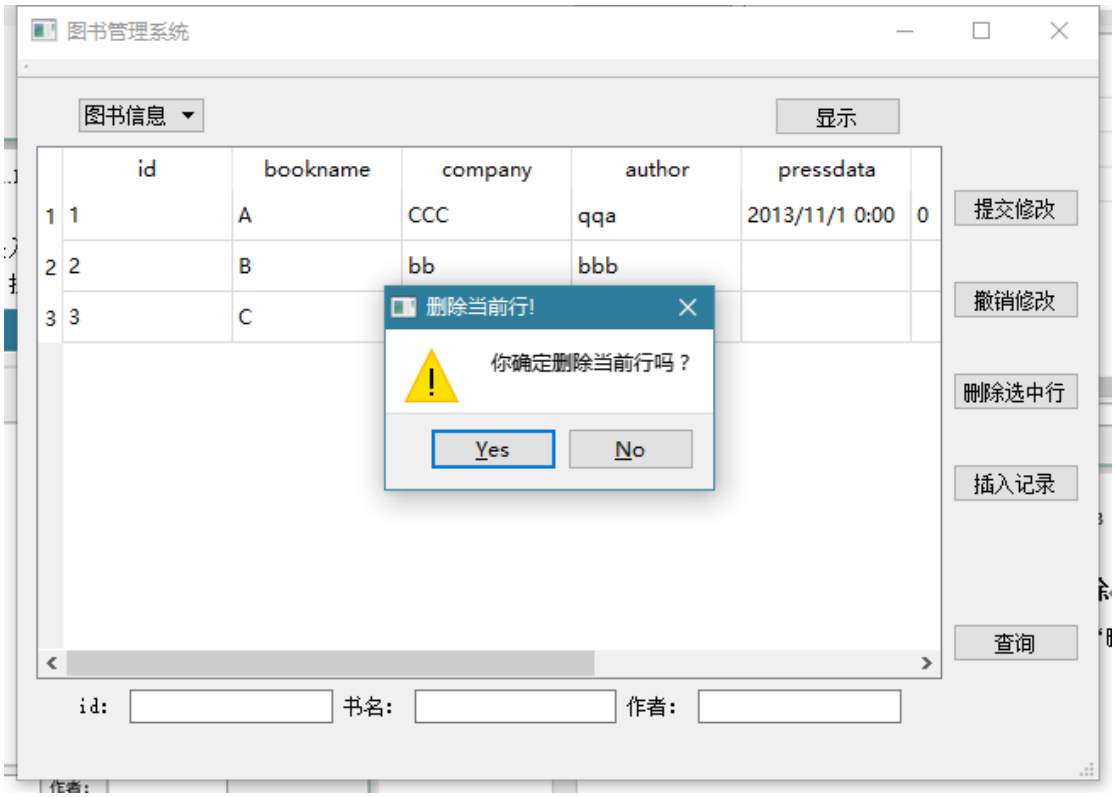

图 4.3.1 删除记录

4.4 图书、用户及借用信息的修改

直接点击单元格即可修改表格内容，修改完毕后点击右侧“提交修改”按钮即可将修改提交至数据库，若在提交之前点击“撤销修改”按钮即可撤销修改操作，注意：若修改已经提交，则无法撤销！如图 4.4.1 所示。

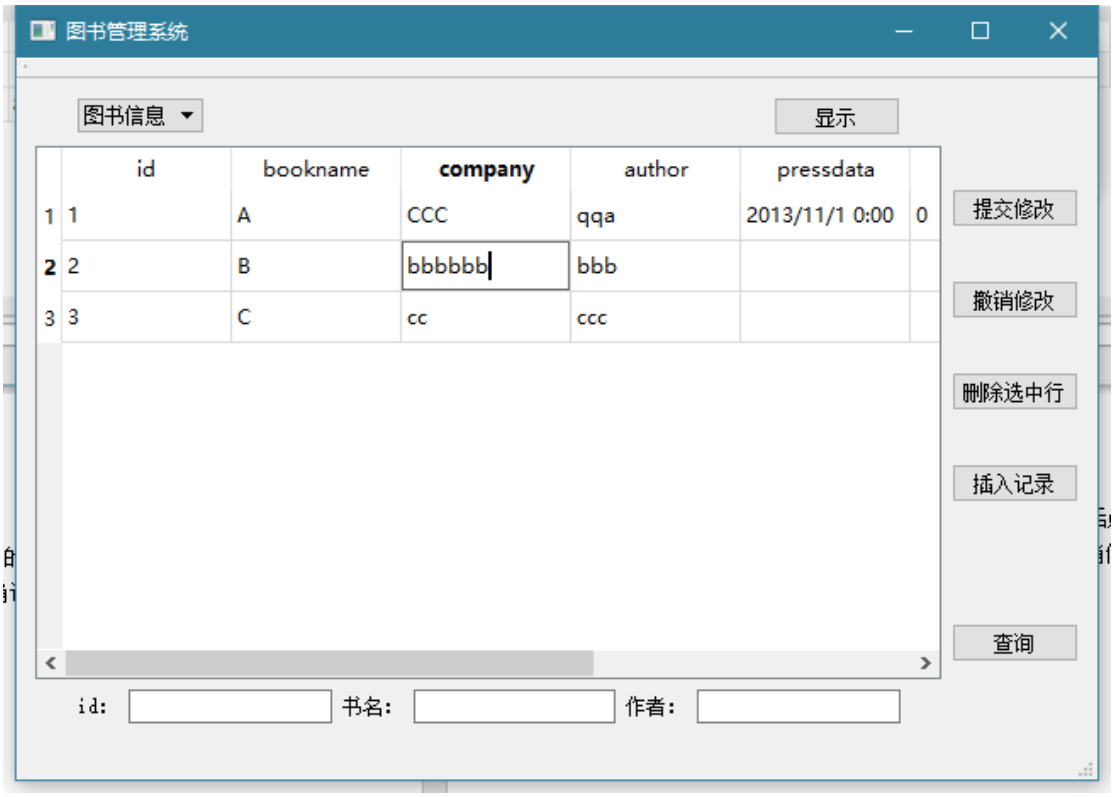

图 4.4.1 修改表格

4.5 数据查询

选择需要执行查询操作的表格，在下方输入框内输入查询条件即可完成查询。

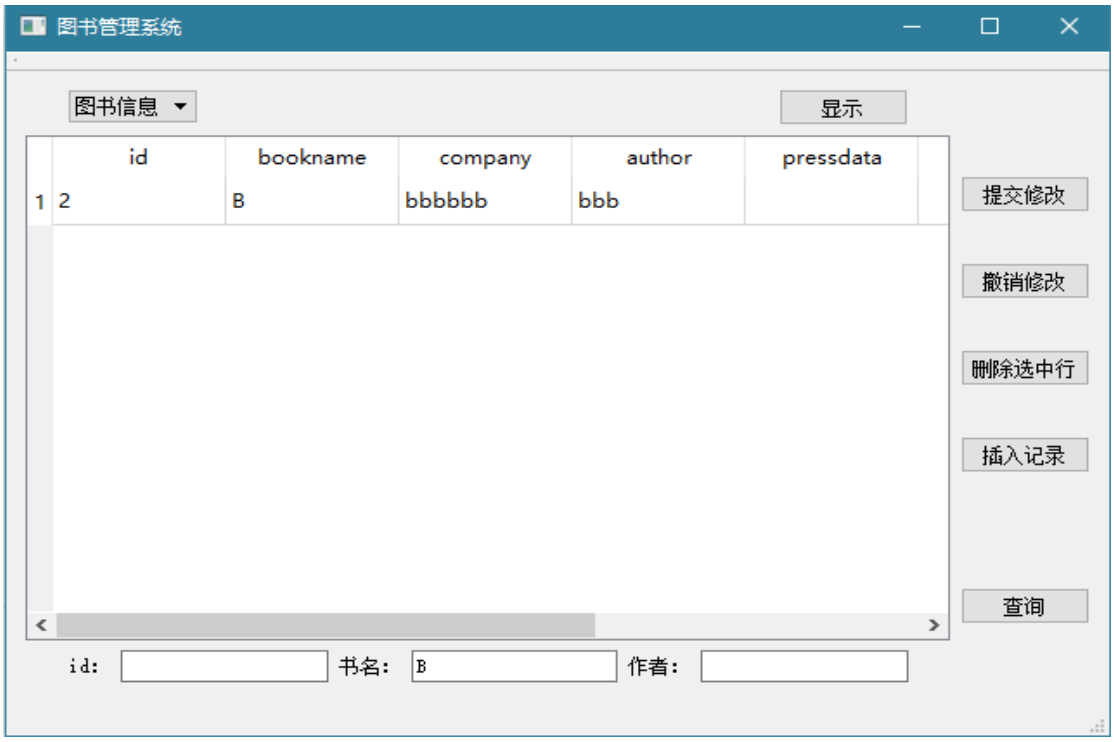

图 4.5.1 查询操作

5 课程设计小结

这次的课程设计前前后后持续了近一个月的时间，收获很多，对自己不足的感受也很是深刻。一开始其实是很不熟悉数据库的，包括环境的安装和配置等等都是从入门搞起，一点一点的学习和向同学了解。当然在这个过程中也学习到了非常多的东西，如 SQL 语句的应用等。

对于本次的课程设计，实现了用户使用的方便以及管理员管理的方便，在界面上力求做到简单实用，在操作方面尽量做到满足用户使用的习惯，但由于自己的水平有限和时间的不足，在很多方面还不完善，有些功能没有在全局上把握好，比如 SQL 语句的执行效果也存在效率不高的问题,关于图书管理的所有操作集中在一个页面实现，出现了一些混淆，程序更改，测试时很不好等。

在整个课程设计的实践中，我学到很多有用的知识，也积累了不少宝贵的开发经验。这也是我第一次使用 Qt 与 C++进行开发，对于 Qt 的使用也是从零开始一点一点学起，初步了解 GUI 的设计基本流程以及相关的思路，通过不断摸索 Qt 包含的库来更加方面地进行数据库开发。本设计通过自己的努力，基本满足了图书管理的基本需要，在今后的学习当中将继续完善此系统，使其功能更加强大，满足用户的更多需求。

建立一个数据库是一个庞大而复杂的问题，数据库课程设计让我们更进一步的了解了其中的知识。数据库的学习还有很长的路，如果想真正了解熟悉这门课程，需要的是不断的深入学习和不断的练习

6 实验练习

实验 1 基本表的创建、数据插入

(1) 建立教学管理中的三个基本表：

Students（S＃，SNAME，AGE，SEX） 学生（学号，姓名，年龄，性别）

Courses（C＃，CNAME，SCORE，PC＃） 课程（课程号，课程名，学分，先行课号）

SC（S＃，C#，GRADE）选修（学号，课程号，成绩）

(2) 用 INSERT 命令输入数据。

S1	LU	20	M
S2	YIN	19	M
S3	XU	18	F
S4	QU	18	F
S5	PAN	14	M
S6	DONG	24	M

表 7 基本表 Students 的数据

C1	数学	4	NULL
C2	英语	8	NULL
C3	数据结构	4	C1
C4	数据库	3.5	C3
C5	网络	4	C1

表 8 基本表 Courses 的数据

S# / C#	S1	S2	S3	S4	S5	S6
C1	85	90	89	84	88	87
C2	73	NULL	86	82	75	85
C3	88	80			90	NULL
C4	89	85		NULL	92	88
C5	73	NULL				87

表 9 基本表 SC 的数据（空格为未选修）

(1) SQL 语句如下：

```
USE ems;
GO
CREATE TABLE Students(
    S# CHAR(9)PRIMARY KEY,
    SNAME CHAR(20)UNIQUE,
    AGE SMALLINT,
    SEX CHAR(2));

CREATE TABLE Courses(
    C# CHAR(4)PRIMARY KEY,
    CNAME CHAR(40)NOT NULL,
    SCORE SMALLINT,
    PC# CHAR(4));

CREATE TABLE SC(
    S# CHAR(9),
    C# CHAR(4),
    GRADE SMALLINT,
    PRIMARY KEY(S#,C#),
    FOREIGN KEY(S#)REFERENCES Students(S#),
    FOREIGN KEY(C#)REFERENCES Courses(C#));
```

执行后可在管理系统中查看已建立三张表 Students、Courses、SC。

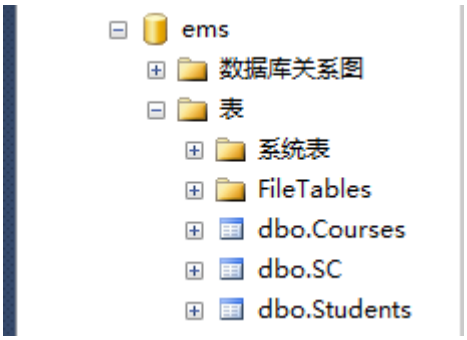

图 6.1

(2) 通过 Microsoft SQL Sever Management Studio 可进行可视化插入数据

	C#	CNAME	PC#	SCORE
▶	C1	数学 ...	*NULL*	4
	C2	英语 ...	*NULL*	8
	C3	数据结构 ...	C1	4
	C4	数据库 ...	C3	3.5
	C5	网络 ...	C1	4
*	*NULL*	*NULL*	*NULL*	*NULL*

图 6.2

	S#	C#	GRADE
▶	S1	C1	85
	S1	C2	73
	S1	C3	88
	S1	C4	89
	S1	C5	73
	S2	C1	90
	S2	C2	*NULL*
	S2	C3	80
	S2	C4	85
	S2	C5	*NULL*
	S3	C1	89
	S3	C2	86
	S4	C1	84
	S4	C2	82
	S4	C4	*NULL*
	S5	C1	88
	S5	C2	75
	S5	C3	90
	S5	C4	92
*	*NULL*	*NULL*	*NULL*

图 6.3

	S#	SNAME	AGE	SEX
▶	S1	LU	20	M
	S2	YIN	19	M
	S3	XU	18	F
	S4	QU	18	F
	S5	PAN ...	14	M
	S6	DONG ...	24	M
*	NULL	NULL	NULL	NULL

图 6.4

实验 2. 数据查询

(1) 列出选修课程号为 C2 的学生学号与姓名。

(2) 检索选修课程名为“数学”的学生学号与姓名。

(3) 检索没有选修 C2 课程的学生姓名与年龄。

(4) 检索选修全部课程的学生姓名。

SQL 语句如下：

```
USE ems;
GO
SELECT Students.S#,SNAME
FROM Students,SC
WHERE Students.S#=SC.S# AND SC.C#='C2'

SELECT Students.S#,SNAME
FROM Students,SC,Courses
WHERE Students.S#=SC.S# AND SC.C#=Courses.C# AND Courses.CNAME='数学'

SELECT SNAME,AGE
FROM Students
WHERE NOT EXISTS
    (SELECT *
     FROM SC
     WHERE S#=Students.S# AND C#='C2')

SELECT SNAME
```

```
FROM Students
WHERE NOT EXISTS
    (SELECT *
     FROM Courses
     WHERE NOT EXISTS
        (SELECT *
         FROM SC
         WHERE S#=Students.S#
            AND C#=Courses.C#));
```

执行结果如下：

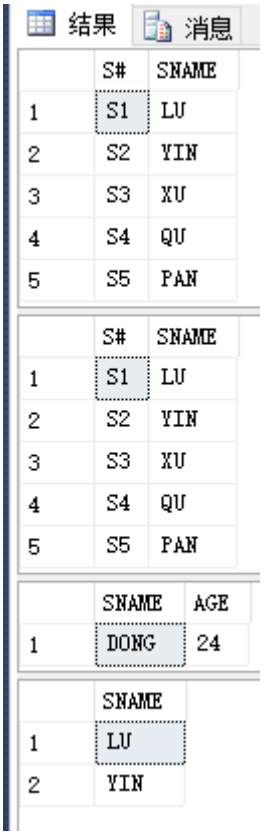

图 6.5

实验 3. 数据修改、删除

(1) 把 C2 课程的非空成绩提高 10%。

(2) 在 SC 表中删除课程名为“数学”的成绩所对应的元组。

(3) 在 S 和 SC 表中删除学号为 S6 的所有数据

SQL 语句如下：

```
USE ems;
GO
```

```
UPDATE SC
SET GRADE=GRADE*1.1
WHERE C#='C2';

DELETE FROM SC
WHERE C# IN
    (SELECT SC.C#
     FROM Courses,SC
     WHERE SC.C#=Courses.C# AND Courses.CNAME='数学');

DELETE FROM Students
WHERE Students.S#='S6';
DELETE FROM SC
WHERE SC.S#='S6';
```

实验 4. 视图的操作

(1) 建立男生学生的视图，属性包括学号，姓名，选修课程名和成绩。

(2) 在男生视图中查询平均成绩大于 80 分的学生学号和姓名。

SQL 语句如下：

(1)
```
USE ems;
GO
CREATE VIEW MStudents
AS SELECT Students.S#,Students.SNAME,Courses.CNAME,SC.GRADE
     FROM Students,SC,Courses
     WHERE Students.S#=SC.S#
     AND SC.C#=Courses.C# AND Students.SEX='M';
```

(2)
```
USE ems;
GO
SELECT S#,SNAME
FROM MStudents
GROUP BY S#,SNAME
HAVING AVG(GRADE)>=80;
```

执行结果如下：

	S#	SNAME	CNAME	GRADE
▶	S1	LU	数学 ...	85
	S1	LU	英语 ...	73
	S1	LU	数据结构 ...	88
	S1	LU	数据库 ...	89
	S1	LU	网络 ...	73
	S2	YIN	数学 ...	90
	S2	YIN	英语 ...	*NULL*
	S2	YIN	数据结构 ...	80
	S2	YIN	数据库 ...	85
	S2	YIN	网络 ...	*NULL*
	S5	PAN ...	数学 ...	88
	S5	PAN ...	英语 ...	75
	S5	PAN ...	数据结构 ...	90
	S5	PAN ...	数据库 ...	92
*	*NULL*	*NULL*	*NULL*	*NULL*

图 6.6

结果 消息

	S#	SNAME
1	S1	LU
2	S2	YIN
3	S5	PAN

图 6.7

实验 5. 库函数，授权的控制

(1) 计算每个学生选修课程的门数、平均成绩。

(2) 建立一个合法的用户，将 SC 表的查询权限授予该用户。

(3) 使用 GRANT 语句，把对基本表 students、Courses、SC 的使用权限授予其他用户。

SQL 语句如下：

(1) SELECT S#, COUNT(C#)个数,AVG(GRADE)平均成绩
FROM SC
GROUP BY SC.S#

(2) USE ems;
GO
GRANT SELECT ON SC TO qwez

(3) USE ems;

```
GO
GRANT ALL PRIVILEGES
ON Students
TO PUBLIC;

GRANT ALL PRIVILEGES
ON SC
TO PUBLIC;

GRANT ALL PRIVILEGES
ON courses
TO PUBLIC;
```

实验 6. 数据库的备份、恢复

(1) 使用完全备份将你的实验数据库备份到 U 盘或硬盘。
(2) 删除你所建立的数据库。
(3) 恢复你的数据库。
(4) 在恢复后的数据库上撤销你建立的基本表和视图。

启动 Management Studio，登录到要增加备份设备的服务器，打开服务器对象，右击新建，在弹出菜单中选择备份设备选项，弹出备份设备对话框，如图 6.8 所示。在设备名称中输入设备的名称，选择设备类型，可以选择磁带；如果选择文件名，则表示使用磁盘做备份，点击确定按钮，新的备份设备即可创建。

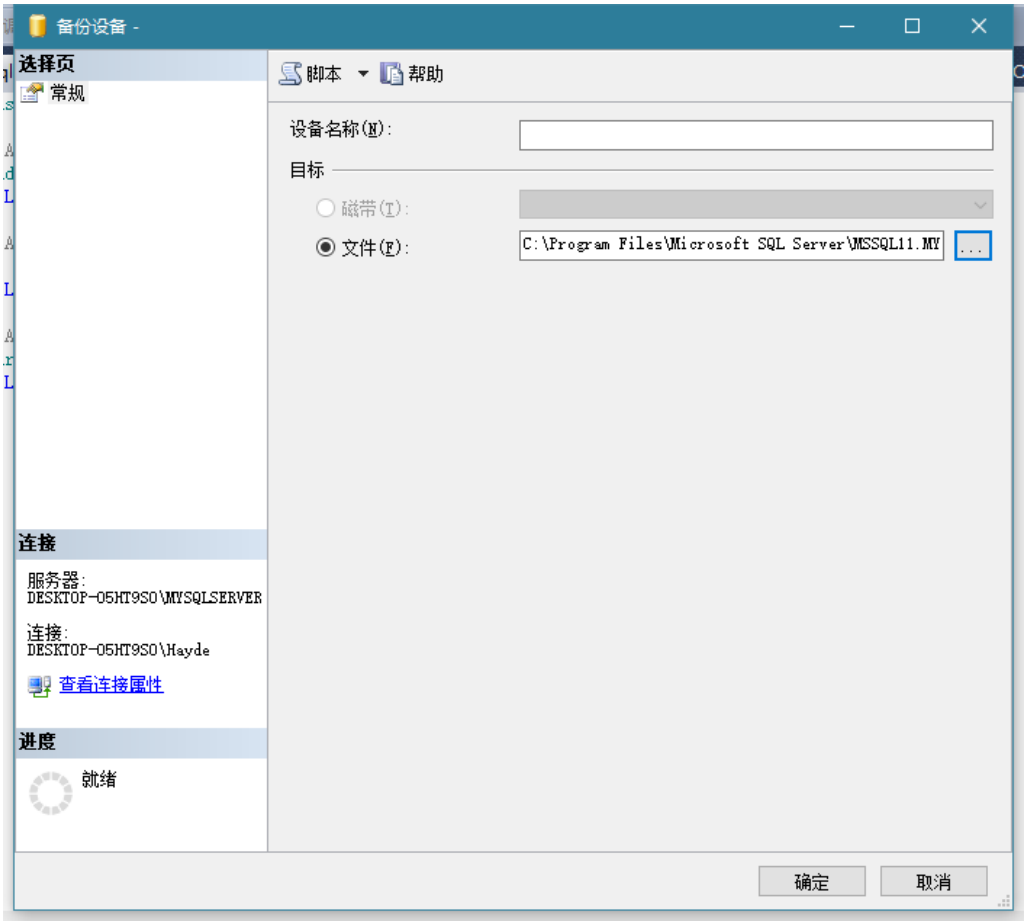

图 6.8

启动 Management Studio，登录到要增加备份设备的服务器，打开数据库文件夹，右击要进行备份的数据库，在弹出菜单中选择任务，再选择备份，弹出备份数据库对话框，如图 6.9 所示。在常规页中的备份选项栏中选择备份类型，单击添加按钮选择备份设备，弹出 如图 40 所示对话框，在此可以选择文件名或备份设备。在选项页中，若选择追加到现有备份集，则将备份内容添加到当前备份；若选择覆盖所有现有备份集，则将原备份覆盖。所有设置完成后，点击确定按钮即可开始备份。

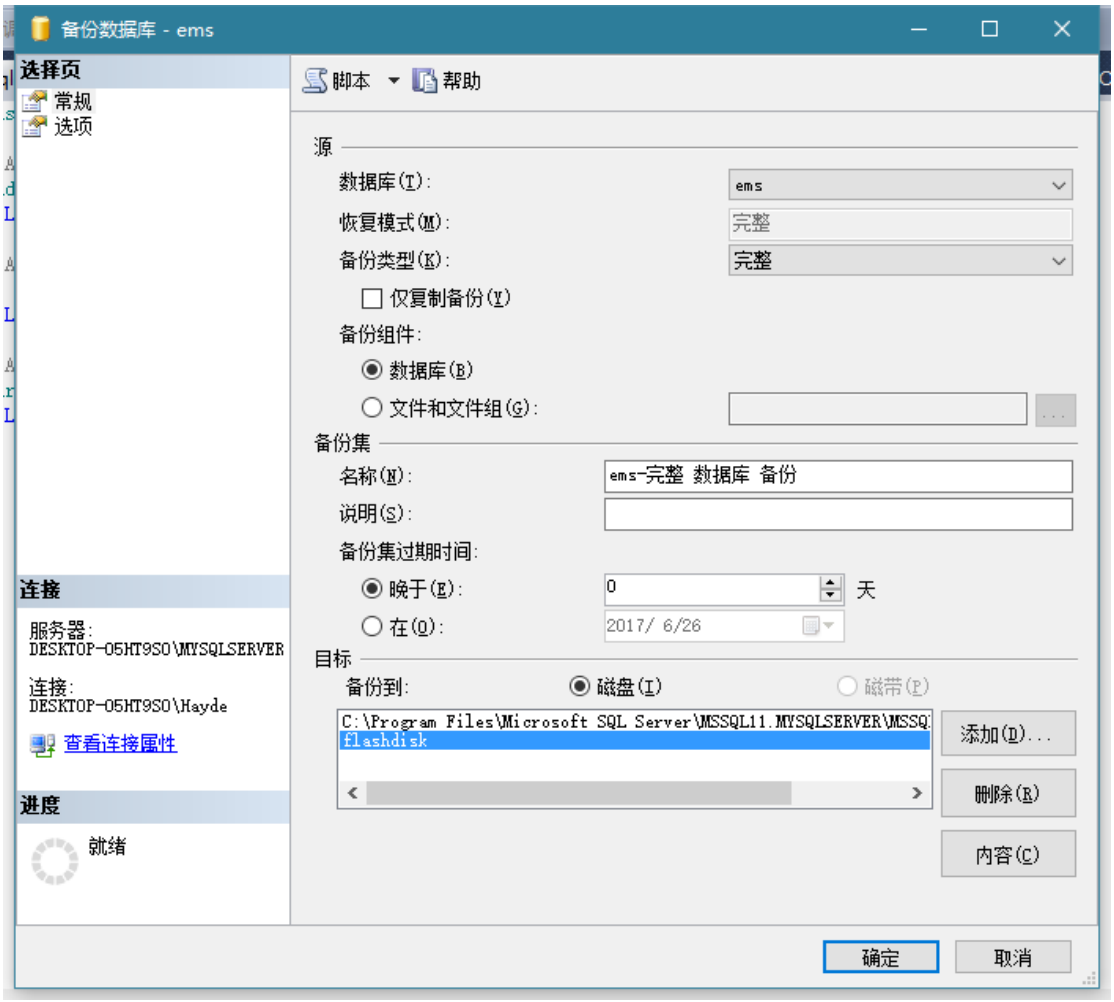

图 6.9

启动 Management Studio，单击要登录的数据库服务器。右击选择还原数据库，弹出还原数据库对话框，如图 6.10 所示。在常规页中的还原目标选项栏中选择要恢复的数据库，在还原源中选择相应的备份类型（恢复与备份时的类型可以不同），在还原的备份集中，选择要恢复的数据库（已经备份的数据库在此均有选项）和使用的备份，缺省情况下使用最近的一次备份。单击选项页设置其他选项。点击确定按钮即可开始恢复数据库。

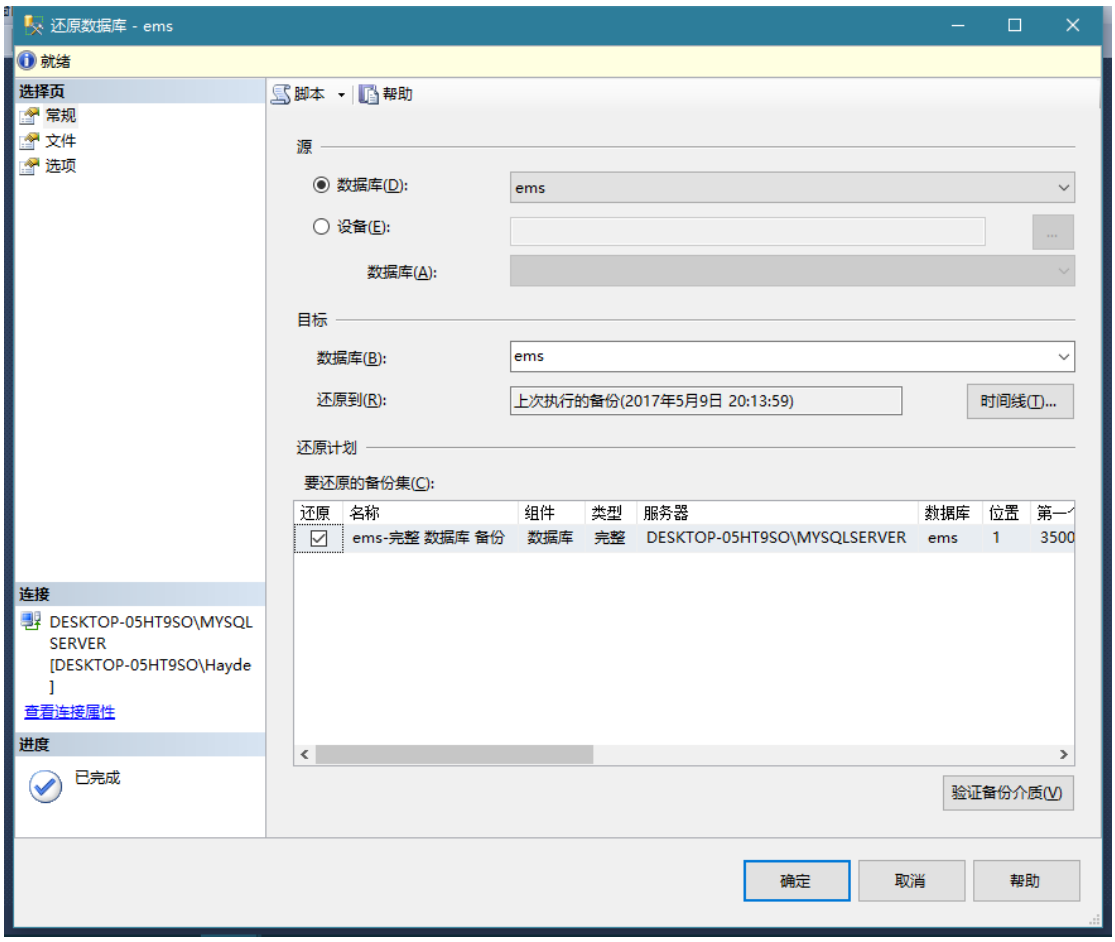
还原数据库 - ems
就绪
选择页
常规
文件
选项
脚本
帮助
源
数据库(D):
ems
设备(E):
数据库(A):
目标
数据库(B):
ems
还原到(R):
上次执行的备份(2017年5月9日 20:13:59)
时间线(T)...
还原计划
要还原的备份集(C):
还原
名称
组件
类型
服务器
数据库
位置
第一
ems-完整 数据库 备份
数据库
完整
DESKTOP-05HT9SO\MYSQLSERVER
ems
1
3500
连接
DESKTOP-05HT9SO\MYSQLSERVER
[DESKTOP-05HT9SO\Hayde
]
查看连接属性
进度
已完成
验证备份介质(V)
确定
取消
帮助

图 6.10

附录（课设代码）

logindialog.h

```
#ifndef    LOGINDIALOG_H
#define    LOGINDIALOG_H

#include <QDialog>

namespace Ui {
class LoginDialog;
}

class LoginDialog : public QDialog
{
    Q_OBJECT

public:
    explicit LoginDialog(QWidget *parent = 0);
    ~LoginDialog();

private slots:
    void on_loginBtn_clicked();

private:
    Ui::LoginDialog *ui;
};

#endif // LOGINDIALOG_H
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
```

```
#include <QSqlTableModel>
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void newFile();     // 新建操作
    bool maybeSave(); // 判断是否需要保存
    bool save();          // 保存操作
    bool saveAs();       // 另存为操作
    bool saveFile(const QString &fileName); // 保存文件
private slots:
    void on_submitBtn_clicked();
    void on_ShowBtn_clicked();
    void on_revertBtn_clicked();
    void on_deletBtn_clicked();

    void on_insertBtn_clicked();

    void on_comboBox_activated(const QString &arg1);

    void on_selectBtn_clicked();

private:
    Ui::MainWindow *ui;
   QSqlTableModel *model;
};

#endif // MAINWINDOW_H
```

logindialog.cpp

```
#include "logindialog.h"
#include "ui_logindialog.h"
#include <QMessageBox>
LoginDialog::LoginDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::LoginDialog)
{
    ui->setupUi(this);
    setWindowTitle("图书管理系统");
}

LoginDialog::~LoginDialog()
{
    delete ui;
}

void LoginDialog::on_loginBtn_clicked()
{
    if(ui->usrLineEdit->text().trimmed() == tr("zhd") &&
            ui->pwdLineEdit->text() == tr("123456"))
        {
            accept();
        }
    else
         {
            QMessageBox::warning(this, tr("Waring"),
                                 tr("user name or password error!"),
                                 QMessageBox::Yes);
            ui->usrLineEdit->clear();
            ui->pwdLineEdit->clear();
            ui->usrLineEdit->setFocus();
        }
}
```

main.cpp

```
#include "mainwindow.h"
#include <QApplication>
#include "logindialog.h"
#include <QSqlDatabase>
#include <QSqlError>
#include <QMessageBox>
#include <QtSql>
#include <QtGui>
#include <QString>
#include <QSqlQueryModel>
#include "ui_mainwindow.h"

void OpenDatabase() //打开数据库
{
    QSqlDatabase db=QSqlDatabase::addDatabase("QODBC");
    db.setDatabaseName(QString("DRIVER={SQL SERVER};"
                               "SERVER=%1;" //服务器名称
                               "DATABASE=%2;"//数据库名
                               "UID=%3;"          //登录名
                               "PWD=%4;"          //密码
                               ).arg("DESKTOP-
05HT9SO\\MYSQLSERVER")
                          .arg("BM")
                          .arg("ZHD")
                          .arg("123456")
                          );
    if (!db.open())
    {
        QMessageBox::critical(0, qApp->tr("Cannot open database"),
                              db.lastError().databaseText(),
QMessageBox::Cancel);

    }
}
```

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    LoginDialog dlg;
        if (dlg.exec() == QDialog::Accepted)
        {
            w.show();
            //打开数据库
            OpenDatabase();

            return a.exec();
        }
        else return 0;
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
#include <QPushButton>
#include <QFileDialog>
#include <QTextStream>
#include <QSqlError>
#include <Qstring>
#include <QTableView>
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    model = new QSqlTableModel(this);
    ui->userlabel->setVisible(false);
    ui->userNamelineEdit->setVisible(false);
```

```
    setWindowTitle("图书管理系统");
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_ShowBtn_clicked()
{
    QString text;
    model = new QSqlTableModel(this);
    text=ui->comboBox->currentText();
    if(!QString::compare(text,"图书信息"))
        model->setTable("books");
    else if(!QString::compare(text,"用户信息"))
        model->setTable("users");
    else if(!QString::compare(text,"借还记录"))
        model->setTable("bookbrowse");
        model->setEditStrategy(QSqlTableModel::OnManualSubmit);
        model->select();
    ui->tableView->setModel(model);
}

void MainWindow::on_submitBtn_clicked()
{

        model->database().transaction(); //开始事务操作
        if (model->submitAll()) {
            model->database().commit(); //提交
        } else {
            model->database().rollback(); //回滚
            QMessageBox::warning(this, tr("tableModel"),
                                 tr("数据库错误: %1")
```

```
                                        .arg(model->lastError().text()));
        }
}

void MainWindow::on_revertBtn_clicked()
{

    model->revertAll();
}

void MainWindow::on_deletBtn_clicked()
{
    //获取选中的行
        int curRow = ui->tableView->currentIndex().row();

        //删除该行
        model->removeRow(curRow);

        int ok = QMessageBox::warning(this,tr("删除当前行!"),tr("你确定"
                                                        "删除当前行吗？
"),

QMessageBox::Yes,QMessageBox::No);
        if(ok == QMessageBox::No)
        {
            model->revertAll(); //如果不删除，则撤销
        }
        else model->submitAll(); //否则提交，在数据库中删除该行
}

void MainWindow::on_insertBtn_clicked()
{
    int rowNum = model->rowCount(); //获得表的行数
        model->insertRow(rowNum); //添加一行
```

```
}

void MainWindow::on_comboBox_activated(const QString &arg1)
{
    if(!QString::compare(arg1,"用户信息"))
    {
        ui->booklabel->setVisible(false);
        ui->bookNamelineEdit->setVisible(false);
        ui->authorlabel->setVisible(false);
        ui->authorNamelineEdit->setVisible(false);

        ui->bookNamelineEdit->clear();
        ui->authorNamelineEdit->clear();

        ui->userlabel->setVisible(true);
        ui->userNamelineEdit->setVisible(true);
    }
    else if(!QString::compare(arg1,"图书信息"))
    {
        ui->userlabel->setVisible(false);
        ui->userNamelineEdit->setVisible(false);

        ui->userNamelineEdit->clear();

        ui->booklabel->setVisible(true);
        ui->bookNamelineEdit->setVisible(true);
        ui->authorlabel->setVisible(true);
        ui->authorNamelineEdit->setVisible(true);
    }
     else if(!QString::compare(arg1,"借还记录"))
    {
        ui->userlabel->setVisible(false);
        ui->userNamelineEdit->setVisible(false);
        ui->booklabel->setVisible(false);
        ui->bookNamelineEdit->setVisible(false);
```

```
        ui->authorlabel->setVisible(false);
        ui->authorNamelineEdit->setVisible(false);

        ui->bookNamelineEdit->clear();
        ui->authorNamelineEdit->clear();
        ui->userNamelineEdit->clear();
    }
}

void MainWindow::on_selectBtn_clicked()
{
    QString sid = ui->idlineEdit->text();
    QString bookname = ui->bookNamelineEdit->text();
    QString authorname = ui->authorNamelineEdit->text();
    QString username = ui->userNamelineEdit->text();
    int nid = sid.toInt();
    if(!sid.isEmpty())
    model->setFilter(QString("id = %1").arg(nid));
    if(!bookname.isEmpty())
    model->setFilter(QString("bookname = '%1'").arg(bookname));
    if(!authorname.isEmpty())
    model->setFilter(QString("author = '%1'").arg(authorname));
    if(!username.isEmpty())
    model->setFilter(QString("username = '%1'").arg(username));
    //显示结果
    model->select();
}
```