中国地质大学（武汉）

报告名称：　数据库原理 C 实验报告

班　　级：　231223

学　　号：　20221000983

姓　　名：　陈子逸

目录

实验 1-4

一、实验目的与要求

实验目的

本实验旨在进行数据基本查询，包括：

1. 设计并实现单表查询 SQL 语句。
2. 设计并实现分组统计查询语句。
3. 设计并实现针对自身的连接查询。
4. 设计并实现多表的连接查询。

实验要求

1. 掌握 SQL 程序设计基本规范：熟练运用 SQL 语言实现数据基本查询，包括单表查询、分组统计查询和连接查询。
2. 能按照 SQL 程序设计规范编写 SQL 查询语句并调试通过。

二、实验代码

2.1 单表查询

选择表中的若干列

```
SELECT Sno, Sname, Sdept
FROM STUDENT;
```

选择表中的若干元组

```
SELECT DISTINCT Sno
FROM SC;
```

对查询结果排序

```
SELECT Sno, Grade
FROM SC
WHERE Cno = '3'
ORDER BY Grade DESC;
```

2.2 连接查询（自身连接）

1. 查询其它系中比自控系（用‘AC’表示自控系）某一学生年龄小的学生姓名：

```
SELECT Sname
FROM STUDENT
WHERE Sdept <> 'AC' AND Sage < (
    SELECT Sage
    FROM STUDENT
    WHERE Sname = '张四'
);
```

2. 查询其它系中比 CS 系某一学生年龄小的学生姓名（对于所有 CS 系学生）：

```
SELECT Sname
FROM STUDENT
WHERE Sage < (
    SELECT Sage
    FROM STUDENT
    WHERE Sdept = 'CS'
) AND Sdept <> 'CS';
```

2.3 连接查询（多表连接）

1. 检索没有获得奖学金、同时至少有一门课程成绩在 80 分以上的学生信息，包括学号、姓名和专业：

```
SELECT DISTINCT STUDENT.Sno, Sname, Sdept
FROM STUDENT, SC
WHERE STUDENT.Sno = SC.Sno AND 奖学金 = 0 AND Grade > 80;
```

2. 求每个课程的选课人数：

```
SELECT Cname, COUNT(*)
FROM SC, COURSE
WHERE SC.Cno = COURSE.Cno
GROUP BY Cname;
```

3. 查询“数据库原理”成绩在 80 分以上（含 80 分）的学生学号：

```
SELECT STUDENT.Sno
FROM STUDENT, SC, COURSE
WHERE STUDENT.Sno = SC.Sno AND SC.Cno = COURSE.Cno AND Cname = "数据库
" AND Grade >= 80;
```

2.4 分组统计查询

1. 使用集函数统计每个课程的选课人数：

```
SELECT COUNT(Sno)
FROM SC
GROUP BY Cno;
```

2. 查询有 3 门以上课程（包括 3 门）不及格的学生学号：

```
SELECT Sno
FROM SC
WHERE Grade < 60
GROUP BY Sno
HAVING COUNT(*) > 3;
```

2.5 更新表信息

对成绩得过满分(100 分)的学生，如果没有获得奖学金，将其奖学金设为 1000 元：

```
UPDATE STUDENT
SET 奖学金 = 1000
WHERE Sno IN (
    SELECT Sno
    FROM SC
    WHERE Grade = 100
);
```

2.6 视图

定义学生成绩得过满分(100 分)的课程视图 AAA，包括课程号、名称和学分：

```
CREATE VIEW EZ_Course AS
SELECT DISTINCT COURSE.Cno, Cname, Ccredit
FROM COURSE, SC
WHERE COURSE.Cno = SC.Cno AND Grade = 100;
```

2.7 创建表

```
CREATE TABLE [student]( [name] varchar(10), [number] int, [class]
varchar(10), [EnterTime] int, [GPA] int, [password] VARCHAR)
```

三、运行结果

3.1 单表查询

单表查询结果如下：

选择表中的若干列

	Sno	Sname	Sdept
1	201215121	明明	CS
2	201215122	张四	CS
3	201215123	李三	MA
4	201215125	陈晨	IS

选择表中的若干元组

	Sno
1	201215121
2	201215122
3	201215123
4	201215125

对查询结果排序

	Sno	Grade
1	201215121	88
2	201215122	80

3.2 连接查询（自身连接）

查询其它系中比自控系（用‘AC’表示自控系）某一学生年龄小的学生姓名

	Sname
1	张四
2	李三

查询其它系中比 CS 系某一学生年龄小的学生姓名（对于所有 CS 系学生）：

	Sname
1	李三

3.3 连接查询（多表连接）

检索没有获得奖学金、同时至少有一门课程成绩在 80 分以上的学生信息，包括学号、姓名和专业

	Sno	Sname	Sdept
1	201215122	张四	CS

求每个课程的选课人数：

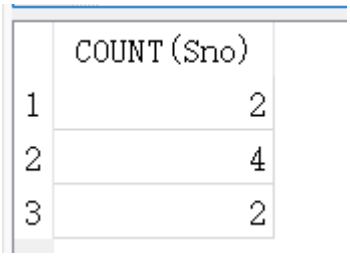
	Cname	COUNT(*)
1	信息系统	2
2	数学	4
3	数据库	2

查询“数据库原理”成绩在 80 分以上（含 80 分）的学生学号：

	Sno
1	201215121

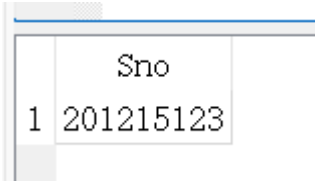
3.4 分组统计查询

使用集函数统计每个课程的选课人数：



	COUNT(Sno)
1	2
2	4
3	2

查询有 3 门以上课程（包括 3 门）不及格的学生学号：



	Sno
1	201215123

四、实验总结

通过本次实验，我掌握了 SQL 语句中基本的查询操作，包括单表查询、连接查询和分组统计查询等，熟悉了如何进行多表连接查询、GROUP BY 进行数据分组统计等技巧。同时，通过对各种查询语句的调试，进一步加深了对 SQL 语言的理解，并提高了编写、调试 SQL 查询语句的能力。

五、问题解决

5.1 问题一：SQL 语句正确但仍然报错

5.1.1 解决方法

在尝试更换英文标点符号后，问题得到了解决。可能是输入时使用了中文标点，导致 SQL 语句解析错误。

5.2 问题二：无法直接修改外码

5.2.1 解决方法

在使用 DB Browser for SQLite 时，如果表的外码是其他表的主码，则无法直接修改。需要使用 SQL 语言进行修改。

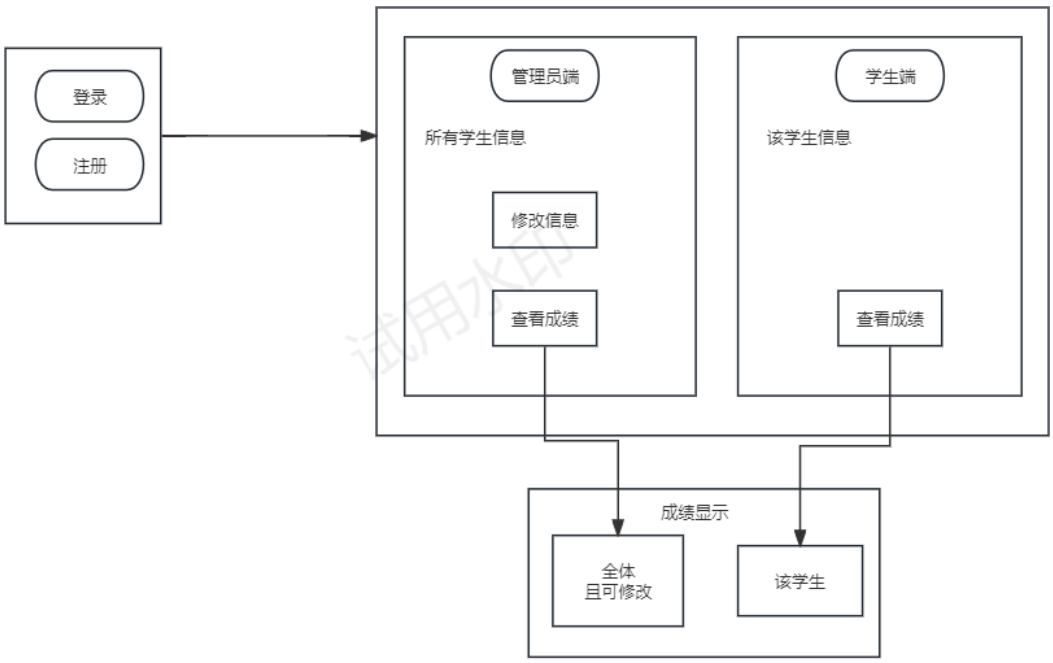
实验5

一、 实验目的

在qt上使用c和sql语言实现一个基于sqlite的数据库学生管理系统，保存了学生的基本信息、成绩、课程信息，并包含注册、登录功能。分为管理员端和学生端，管理员能够查看所有同学信息，添加、修改、删除学生信息以及成绩信息；学生端能够查看自己的信息以及成绩，还实现了成绩可视化工具让同学更直观地看到成绩高低。

本实验用的数据库结构是上课时代码示例用的结构，即student、sc、course，在这个基础上增加了奖学金（做作业时加的），以及密码（用于登录功能）。

二、 系统框架



（试用水印为wps自带）

对于系统管理员，需要实现更改信息的功能，这需要和学生区分开，所以在登录界面就设计了登录后的不同跳转，相当于将学生和管理员的系统分开设计。

三、 功能实现

3.1 qt与数据库连接

Qt提供了一系列的库函数来支持sql语言以及数据库的操作

```
bool opendatabase()
{
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
```

```
    db.setDatabaseName("C:\\Users\\wwwcz\\Documents\\Tencent
Files\\1637150660\\FileRecv\\sc.db");  //平时 debug 正常用
    //mydb.setDatabaseName("./student.db"); //release 用
    if (!db.open()) {
        QMessageBox::critical(nullptr, QObject::tr("Cannot open database"),
                              QObject::tr("Unable to establish a database connection.\n"
                                          "This example needs SQLite support. Please read "
                                          "the Qt SQL driver documentation for information how "
                                          "to build it.\n\n"
                                          "Click Cancel to exit."), QMessageBox::Cancel);
        return false;
    }
    return true;
}
```

这段代码中addDatabase("QSQLITE")确定了数据库的类型为sqlite，db.setDatabaseName读取数据库文件所在的地址找到数据库，然后db.open()打开它。

3.2 sql操作

查询：

对全体：

信息：

```
select Sname as 姓名 ,Sno as 学号,Ssex as 性别 ,Sdept as 学院, 奖学金
from student
```

成绩：

```
select Sname as 姓名 ,student.Sno as 学号,Cname as 课程,Course.Cno as 课程号
,Grade as 成绩
from student,sc,course
where student.sno = sc.sno and sc.cno =course.cno
```

对该学生：

信息：

```
select Sname as 姓名 ,Sno as 学号,Ssex as 性别 ,Sdept as 学院, 奖学金
from student where Sname='"+stuName+"'
```

成绩：

```
select Sname as 姓名 ,student.Sno as 学号,Cname as 课程,Course.Cno as 课程号
,Grade as 成绩
from student,sc,course
where student.sno = sc.sno and sc.cno =course.cno and Sname='"+Sname+"'
```

插入：

```
Insert into student
values('"+Sno+"','"+Sname+"','"+Ssex+"','"+Sage+"','"+Sdept+"','"+奖学金
+"','"+password+"')
```

修改：

```
update student set number ='"+Sno+"','"+Sname+"','"+Ssex+"','"+Sdept+"','"+奖学金+"','"+password+"'
where name='"+Sname+"'
```

排序：

```
order by Sno
order by AVG(grade)
order by '奖学金'
```

3.3 界面设计

Qt使用信号与槽来进行程序与界面的交互，简单来说就是当你点击一个按钮后会执行哪一段程序。

主要背景Widget

在style sheet中填入

```
QMainWindow {
background-image: url(:/new/prefix1/image/background.jpg);
background-repeat: no-repeat;
background-position: center;
}

QPushButton {
background-color: rgba(255, 255, 255, 150); // 设置半透明白色背景
}
```

能够添加图片资源，在背景中展示你喜欢的图片

交互按钮pushButton

作为触发事件，点进去后运行槽（函数）的内容，通常可以实现跳转、查询等各种功能。

数据输入LinEdit

输入的内容可以被读取，比如：

```
find=ui->lineEdit_findname->text();
```

数据库查询显示tableView

```
    QSqlQueryModel *modell=new QSqlQueryModel;
    QString find,sql0;
    find=ui->lineEdit_findname->text();
    sql0="select Sname as 姓名 ,Sno as 学号,Ssex as 性别 ,Sdept as 学院, 奖学金  from student  where Sname LIKE '%"+find+"%'";
    modell->setQuery(sql0);
```

```
ui->tableView->setModel(modell);
```

使用一个QSqlQueryModel对象来实现查询显示，ui->tableView->setModel(modell);

Ui界面
总体的可视化设计都要靠ui文件，一个ui文件管理一个界面，我这里设计了登录、管理员、修改信息、删除信息、学生、成绩的界面，这样分开设计能够便于程序的独立性，不过相应的cpp和h文件也要多增加几个，让整体可读性高一些。
界面的控件通过connect()函数来与程序连接。

3.4 成绩可视化

需要使用qtcharts的插件，在qt maintence tool里面下载。然后调用里面的库函数

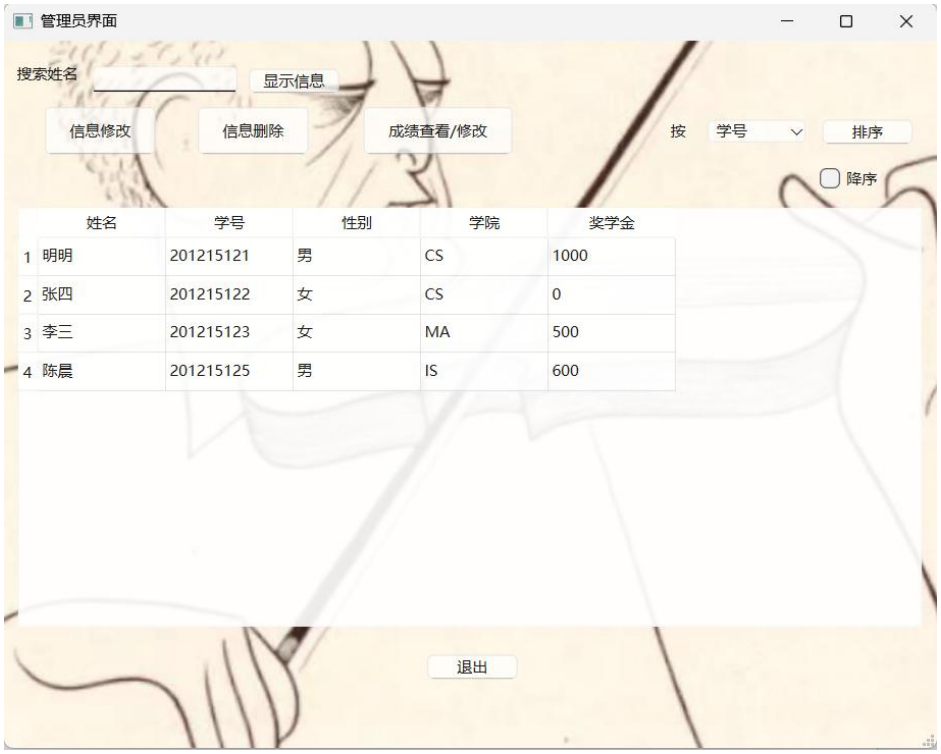
```
// 创建图表对象
QChart *chart = new QChart();
chart->addSeries(series);
chart->setTitle("学生成绩分布");
chart->setAnimationOptions(QChart::SeriesAnimations);

// 创建坐标轴
QBarCategoryAxis *axisX = new QBarCategoryAxis();
axisX->append(courseNames);  // 使用课程名称作为 X 轴的标签
chart->setAxisX(axisX, series);
chart->setAxisY(new QValueAxis(), series);

// 创建图表视图并显示图表
QChartView *chartView = new QChartView(chart);
chartView->setRenderHint(QPainter::Antialiasing);
ui->verticalLayout->addWidget(chartView); //
```

四、 运行结果

4.1 登录

可以看到其中有管理员身份的选项，通过这个可以将学生与管理员分开。也包含了基本的用户名、密码输入，登录注册的选项。

4.2 管理员界面



管理员包含全部学生信息的显示，单个学生信息的查询，和修改、删除，全部学生的成绩查看/修改，排序功能。
也可以使用名字搜索，使用了sql语言中的like实现模糊搜索。

4.2 学生信息修改界面

提供了添加和修改的选项，能够将student表的所有信息都填入。

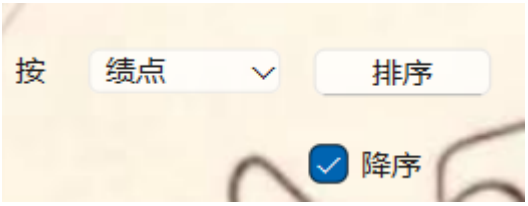
4.3 学生信息删除界面

提供了删除的选项。

4.4 学生成绩查看/修改界面

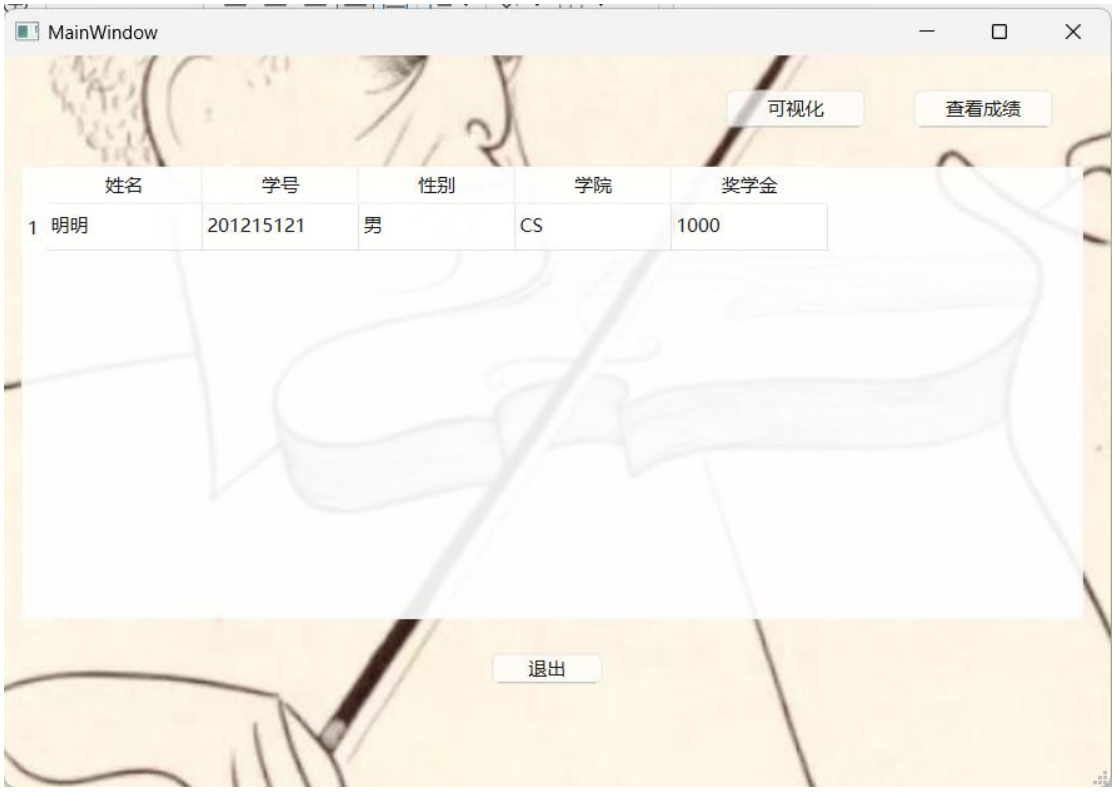
这里可以查看所有课程成绩信息，以及提供了添加和删除的功能，只需要填右边的栏，再点击添加或删除即可。

4.5 排序操作



可以按照学号、奖学金、绩点三种方式来排序，也支持降序排序。

4.6 学生界面



可以看到学生界面基本只能查看，不能修改，而且会根据刚刚登录的学生姓名来锁定查询的学生，不会出现查询到别的学生的情况。

点击**可视化**后：

使用qtcharts来显示的表格通过vertical layout的形式显示在界面中。

查看成绩：

这个画面其实与管理员是共用一个ui的，但是qt非常贴心地提供了setvisible这个函数来隐藏界面内容，所以可以显示与管理员不一样的内容，削减了修改成绩的功能。

五、 实验结论与心得

本实验，主要实现了一个基于qt和sql库的学生管理系统，使用sqlite库，包含了登录注册模块，查询模块，添加删除模块，可视化模块。总体上将管理员和学生分开，实现功能的独立，也便于系统的维护与管理。

在设计过程中，我先通过qt提供的sql案例学习，了解了Model/View框架，和连接数据库的基方法，但是这样还远不能达到能够自主编写的程度。于是我上网搜索相关视频，在csdn上查看每个小部分的内容如何实现，并结合git上的源码进行学习，大概了解了具体原理，然后再自主进行编写。

设计的主要难点一开始集中在将ui、程序、数据库连接起来，在编写过后才真正学会了其中的联系。几个程序的协作使用到了不同的类，在类中包含其他类型的对象来实现界面跳转等功能。多文件编译、sql连接、查询的内容会出现很多报错内容，不断发现程序错误再改正是比较漫长的过程。好在qt的信息与槽结构事实上节省了很多通信上的问题，结构也很独立且简单，所以没有特别困难的bug要解决。

建立学生管理系统这个学习过程对我的自主学习能力有不小提升。不过程序依然有非常多需要改善的地方，也有更多可以改进的地方，期待今后这个实验的内容可以在实际中应用上。