数据库系统开发实验报告

选课管理

1. 实验方法

（1） 开发环境

操作系统	Windows10
数据库管理系统	SQL Server 2019
数据模型设计工具	PowerDesigner16.5
应用程序开发工具	Microsoft Visual Studio 2017
框架	.Net FrameWork 4.7.1
编程语言	C#

（2） 数据模型设计与建立

对实验要求进行基本分析后，可知该系统应包括四个基本实体：学生 Student、教师 Teacher、课程种类 Course、课程 Class。

实体之间的联系关系为：学生-课程（多对多联系）、课程种类-课程（一对多联系）、教师-课程（一对多联系）。

为实现学生与课程间的多对多联系，设置单独实体“Selected_Class”存储学生选择课程的信息。

为实现提交与临时保存功能，故设置单独实体“Submited”，记录学生选课信息的提交状态，保存已提交的学生 ID，若在提交后再次临时保存，则变为未提交状态，从该表中移除。

各实体的基本信息表设计如下所示：

学生表 Student

属性名	属性含义	类型	备注
Student_ID	学生 ID	Int	主键
Name	姓名	Varchar(64)	

已提交学生表 Submited

属性名	属性含义	类型	备注
Student_ID	学生 ID	Int	主键，外键

课程种类表 Course

属性名	属性含义	类型	备注
Course_ID	课程种类 ID	Int	主键
Name	课程名	Varchar(64)	
Credit	学分	Int	

教师表 Teacher

属性名	属性含义	类型	备注
Teacher_ID	教师 ID	Int	主键
Name	姓名	Varchar(64)	

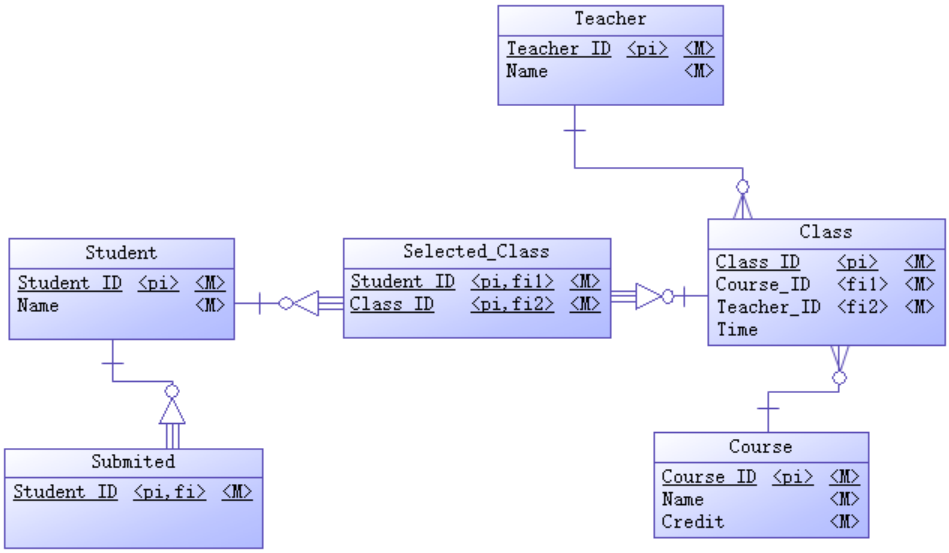
课程表 Class

属性名	属性含义	类型	备注
Class_ID	课程 ID	Int	主键
Course_ID	种类 ID	Int	外键 1
Teacher_ID	教师	Int	外键 2
Time	授课时间	Varchar(256)	

选课表 Selected_Class

属性名	属性含义	类型	备注
Student_ID	学生 ID	Int	外键，联合主键属性之一
Class_ID	课程 ID	Int	外键，联合主键属性之一

如此设计可以完整地存储学生选课系统的信息与不同实体间的关系，并在后续的应用程序开发中实现相应的功能。根据以上设计，使用 PowerDesigner16.5 构建逻辑模型，形成 ERD 图如下所示：



保存逻辑模型文件为 LogicalDataModel.ldm。

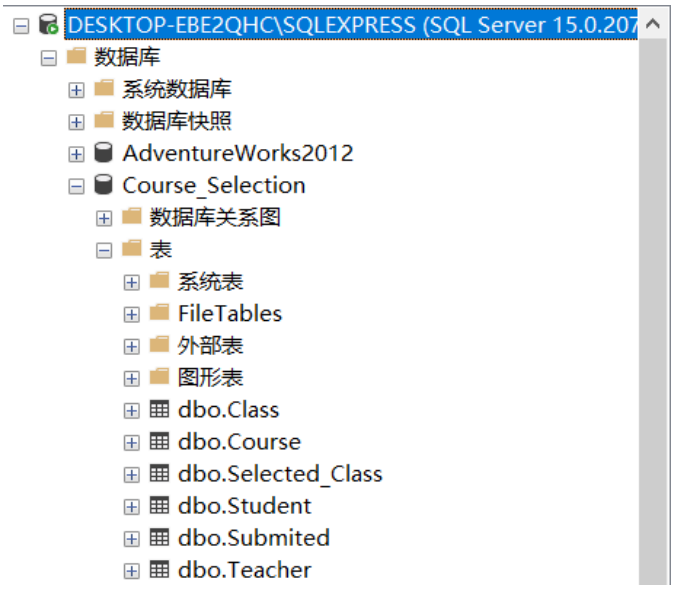
使用 PowerDesigner 将逻辑模型转化为物理模型，对应 DBMS 为 Microsoft SQL Server 2012，保存物理模型文件为 PhysicalDataModel.pdm。

使用该物理模型生成对应 DBMS 的 SQL 命令行代码，保存文件为 crebas.sql。

综上，完成了数据模型的构建，逻辑模型文件 LogicalDataModel.ldm、物理模型文件 PhysicalDataModel.pdm、 SQL 命令行文件 crebas.sql 均保存在附件文件夹内。

（3） 数据库生成与基本数据录入

使用系统管理员登录名连接 SQL Server，新建数据库 Course_Selection。打开 crebas.sql 文件，可以在查询编辑框内查看具体代码。切换当前数据库至 Courese_Selection，执行该代码，Course_Selection 数据库即可构建此前设计好的数据库对象。构建完成后，该数据库内表如下图所示：



此时各表仍为空，需要向其中手工录入部分信息，下面给出录入信息的命令行语句，信息的直观示意表格可在附录中查看。

```
insert into Student values(1001,'张三');
insert into Student values(1002,'李四');
insert into Student values(1003,'王五');
insert into Teacher values(2001,'陈品如');
insert into Teacher values(2002,'曹兰');
insert into Teacher values(2003,'岑永锋');
```

```
insert into Teacher values(2004,'常仪');
insert into Teacher values(2005,'蔡勇');
insert into Teacher values(2006,'崔嘉敏');
insert into Teacher values(2007,'柴立');
insert into Teacher values(2008,'成可');
insert into Teacher values(2009,'池临泽');
insert into Course values(1,'大学计算机',2);
insert into Course values(2,'程序设计基础',2);
insert into Course values(3,'线性代数',2);
insert into Course values(4,'数值分析',2);
insert into Course values(5,'最优化方法',2);
insert into Course values(6,'数学分析',3);
insert into Course values(7,'大学物理',3);
insert into Course values(8,'数据结构',3);
insert into Course values(9,'数据库原理',3);
insert into Class values(101,1,2001,'周一，1-3节');
insert into Class values(102,2,2001,'周二，4-5节');
insert into Class values(103,2,2002,'周二，6-7节');
insert into Class values(104,3,2003,'周一，4-5节');
insert into Class values(105,3,2004,'周一，6-7节');
insert into Class values(106,4,2004,'周一，8-10节');
insert into Class values(107,5,2003,'周二，8-10节');
insert into Class values(108,6,2005,'周三，1-3节');
insert into Class values(109,6,2003,'周三，8-10节');
insert into Class values(110,7,2006,'周三，4-5节');
insert into Class values(111,7,2007,'周三，6-7节');
insert into Class values(112,8,2002,'周四，4-5节');
insert into Class values(113,8,2008,'周四，6-7节');
insert into Class values(114,9,2009,'周五，1-3节');
```

在新建查询内输入并执行以上语句，即可完成对基本信息的录入。

（4） 数据库应用程序的设计与实现

使用 Visual Studio 构建 Visual C# 窗体应用项目 Course_Selection，并对需要实现的功能进行分析，确定所有功能通过一个窗体 Form1 实现，并设计窗口中可视化基本控件列表如下：

控件类型	控件名称	控件说明
文本框 TextBox	tbxStudentID	输入学号，获取学号信息，同时通过文本框文本改变事件，在其余可视化控件中显示学生各项信息。
	tbxStudentName	只读文本框，显示学生姓名

	tbxSubmited	只读文本框，显示学生提交状态
	tbxNumber	只读文本框，显示已选总课程数
	tbxCredit	只读文本框，显示已选总学分
数据表格 DataGrid	dataGrid1	已选课程列表，只读
	dataGrid2	可选课程列表，只读
按钮 Button	btnDelete	选中已选课程列表中某行后点击，可以将该课程从原列表移到可选课程列表中。
	btnAdd	选中可选课程列表中某行后点击，可以根据已选课程信息判断是否将选中课程从原列表移到已选课程列表中。
	btnSave	根据已选课程列表修改数据库，并标记该学生为未提交状态。
	btnSubmit	根据已选总课程数和总学分判断是否符合要求，符合要求则根据列表修改数据库，并标记该学生为已提交状态。

通过以上可视化空间，实现了用户与选课系统的交互接口，接下来说明与数据库连接相关的控件，通过回答实验要求中的问题的形式展开。

问题 1：使用哪种数据提供程序？

本次实验采用 OLE DB .NET Framework 数据提供程序,其接口名为 SQLNCLI11。

问题 2：使用的数据连接对象是哪一个？连接对象是如何建立的？最后生成的连接对象中的连接字符串是什么？代表什么含义？

数据连接对象为 oleDbConnection，在建立第一个数据适配器时通过数据适配器设置向导界面第一页新建连接后弹出的添加连接界面建立该对象，数据源选择 Microsoft SQL Server (OLE DB)，服务器名选择 DESKTOP-EBE2QHC\SQLEXPRESS，身份验证选择 Windows 身份验证，连接到的数据库选择 Course_Selection。连接对象的连接字符串如下所示：

Provider=SQLNCLI11;Data Source=DESKTOP-EBE2QHC\SQLEXPRESS;Integrated Security=SSPI;Initial Catalog=Course_Selection

表明用于连接的提供者名为 SQLNCLI11（OLE DB 访问接口），连接的 SQL Server 服务器名称为 DESKTOP-EBE2QHC\SQLEXPRESS，使用当前 Windows 账户凭据进行身份验证，连接的数据库为 Course_Selection。

问题 3：使用的数据适配器对象是什么？其中的查询或更新语句是什么？如果有参数则参数是如何处理的？

共建立 4 个数据适配器对象，列表显示如下：

对象名	使用说明
Adapter_Update	用于提交或保存操作时更新数据库
Adapter_Selected	用于录入学生学号后将对应已选课程信息写入对应的数据表 Class_Selected 中，列名用英文表示，该数据表用于提交或保存时对数据库进行更新。
Adapter_UnSelected	用于录入学生学号后将对应可选课程信息写入对应的数据表 Class_UnSelected 中，列名用英文表示
Adapter_IsSubmited	用于修改学生提交状态

接下来分别展示各适配器的语句和参数情况

Adapter_Update

删除语句:

```
DELETE FROM Selected_Class
WHERE    (Student_ID = ?)
```

参数情况：1 个参数，需要学号填充，通过显示设置参数的 Value 值建立，示例语句如下，

```
Adapter_Update.DeleteCommand.Parameters[0].Value = Convert.ToInt32(this.SID);
```

插入语句：

```
INSERT INTO Selected_Class
(Student_ID, Class_ID)
VALUES   (?,?)
```

参数情况：2 个参数，需要学号和课程号填充，通过显示设置参数的 Value 值建立，示例语句如下，

```
Adapter_Update.InsertCommand.Parameters[0].Value = Convert.ToInt32(this.SID);
Adapter_Update.InsertCommand.Parameters[1].Value = Convert.ToInt32(CID);
```

Adapter_Selected

选择语句：

```
SELECT  Class.Class_ID, Course.Name, Course.Credit, Teacher.Name AS
TeacherName, Class.[Time]
FROM      Class INNER JOIN
           Course ON Class.Course_ID = Course.Course_ID INNER JOIN
           Teacher ON Class.Teacher_ID = Teacher.Teacher_ID
WHERE   (Class.Class_ID IN
               (SELECT  Class_ID
                FROM       Selected_Class
                WHERE    (Student_ID = ?)))
```

参数情况：1 个参数，需要学号填充，通过显示设置参数的 Value 值建立，示例语句如下，

```
Adapter_Selected.SelectCommand.Parameters[0].Value=this.SID;
```

Adapter_UnSelected

选择语句：

```
SELECT  Class.Class_ID, Course.Name, Course.Credit, Teacher.Name AS
TeacherName, Class.[Time]
FROM      Class INNER JOIN
           Course ON Class.Course_ID = Course.Course_ID INNER JOIN
           Teacher ON Class.Teacher_ID = Teacher.Teacher_ID
WHERE   (Class.Class_ID NOT IN
               (SELECT  Class_ID
                FROM       Selected_Class
                WHERE    (Student_ID = ?)))
```

参数情况：1 个参数，需要学号填充，通过显示设置参数的 Value 值建立，示例语句如下，

```
Adapter_UnSelected.SelectCommand.Parameters[0].Value = this.SID;
```

Adapter_IsSubmited

删除语句:

```
DELETE FROM Submited
WHERE   (Student_ID = ?)
```

参数情况：1 个参数，需要学号填充，通过显示设置参数的 Value 值建立，示例语句如下，

```
Adapter_IsSubmited.DeleteCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
```

插入语句：

```
INSERT INTO Submited
(Student_ID)
VALUES  (?)
```

参数情况：1 个参数，需要学号填充，通过显示设置参数的 Value 值建立，示例语句如下，

```
Adapter_IsSubmited.InsertCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
```

问题 4：使用的数据集对象是什么？数据集中有哪些数据表？数据表是由哪些适配器对象生成的？（或采用其它方法）。

使用的数据集对象为 dataSet11，数据集中有两个数据表，其具体信息如下，

表对象名	来源适配器对象	说明
Class_Selected	Adapter_Selected	学生已选课程，在录入学号后会由适配器填充初始行信息，添加选课会在该表中插入行，删除选课会在该表中删除对应行，在提交或保存时会将该表中各行的信息录入数据库。
Class_UnSelected	Adapter_UnSelected	学生可选课程，在录入学号后会由适配器填充初始行信息，添加选课会在该表中删除对应行，删除选课会在该表中插入行。

两个数据表的列名均为 Class_ID、Name、Credit、TeacherName、Time，可完整表示课程列表信息。

综上，完成了对应用程序基本控件说明，由于本应用程序全部功能均在一个窗体 Form1 中实现，为了方便之后对各功能实现思路和代码的说明，对 Form1 各手动添加修改的成员做简单介绍，各成员列表如下

类型	名称	简要说明
成员变量	number_selected	int 型，保存已选课程数，减少对数据表的操作
	credit_selected	int 型，保存已选总学分，减少对数据表的操作
	SID	String 型，保存学号
	SNAME	String 型，保存姓名
	submited	int 型，保存提交状态，0 表示未提交，1 表示已提交，减少对数据库的操作
成员函数	tbxStudentID_TextChanged	由文本框控件 tbxStudentID 文本内容改变事件调用，根据录入学号赋值 SID，查询获得 SName 和 submited 值，并调用数据适配器填充数据集，以此在 dataGrid 控件中显示该学生初始已选课程列表和可选课程列表。
	Update_Text	在各成员函数中均被调用，根据当前控件和成员变量状况修改成员变量值，并将其显示在各文本框控件中。
	btnDelete_Click	由按钮控件 btnDelete 点击调用，获取已选课程列表中选中行的信息，在 Class_Selected 数据表中删除对应行，并在 Class_UnSelected 数据表中插入对应行。
	btnAdd_Click	由按钮控件 btnAdd 点击调用，获取可选课程列表中选中行的信息，在 Class_UnSelected 数据表中删除对应行，并在 Class_Selected 数据表中插入对应行。
	btnSave_Click	由按钮控件 btnSave 点击调用，将选课信息临时保存到数据库，并将该学生标记为未提交状态。调用适配器 Adapter_Update 从数据库表 Selected_Class 中删除原选课记录，并将 Class_Selected 数据表中的记录的已选信息逐行插入选课表，若当前学生为已提交状

		态，则调用适配器 Adapter_IsSubmited 在数据库表 Submited 中删除对应记录，并修改成员变量 submited 值。
	btnSubmit_Click	由按钮控件 btnSubmit 点击调用，将选课信息提交到数据库，并将该学生标记为提交状态。首先根据成员变量判断是否符合提交要求，若符合则调用适配器从数据库选课表中删除原选课记录，并将 Class_Selected 数据表中的记录的已选信息逐行插入选课表，若当前学生为未提交状态，则调用适配器 Adapter_IsSubmited 在数据库表 Submited 中插入对应记录，并修改成员变量 submited 值。

接下来开始介绍实现各功能的思路与代码的详细说明：

功能 1：录入学号，自动显示学生姓名。

涉及控件：tbxStudentID、tbxStudentName、oleDbConnection、dataSet11

涉及函数：tbxStudentID_TextChanged、Update_Text

实现思路：通过 tbxStudentID 控件的文本改变事件函数实现，在文本 tbxStudentID 框内输入学号，判断其符合数字格式后，自动将其赋值给窗体对象 Form1 的私有变量 SID，并新建 OleDbCommand 查询命令（因为只涉及单个值的查询，所以不单独设置适配器，如果要显示复杂学生信息，可以在此处修改），对 Student 表进行查询，若存在该学生则可获得学生姓名，并将查询结果赋值给 Form1 的私有变量 Sname 和 tbxStudentName.Text，即可在 tbxStudentName 中显示姓名。对学生提交状态表 Submited 做类似的查询，即可获得相应的提交状态，并将相应的值赋给私有变量 submited

主要代码：

仅展示函数 tbxStudentID_TextChanged 中涉及该功能的部分

```
int temp;
bool result = int.TryParse(tbxStudentID.Text, out temp);
if (!result)
    return;
this.SID = tbxStudentID.Text;
//初始化数据表
dataSet11.Clear();
```

```
            //获取姓名
            String strSelectName = "SELECT Name FROM Student WHERE Student_ID = " +
this.SID;
            OleDbCommand cmdSelect = new OleDbCommand(strSelectName,
this.oleDbConnection);
            this.oleDbConnection.Open();
            if (cmdSelect.ExecuteNonQuery() != -1)
            {
                this.oleDbConnection.Close();
                //清空文本框
                tbxStudentName.Text = "";
                tbxCredit.Text = "";
                tbxNumber.Text = "";
                tbxSubmited.Text = "";
                return;
            }
            this.SName = cmdSelect.ExecuteScalar().ToString();
            this.oleDbConnection.Close();
            tbxStudentName.Text = this.SName;

            //获取提交情况
            String strSelectSubmit = "SELECT Student_ID FROM Submited WHERE Student_ID
= " + this.SID;
            cmdSelect = new OleDbCommand(strSelectSubmit, this.oleDbConnection);
            this.oleDbConnection.Open();
            if (cmdSelect.ExecuteNonQuery() != -1)
                this.submited = 0;
            else
                this.submited = 1;
            this.oleDbConnection.Close();
```

功能 2：在可选择课程列表上方显示已选择的课程列表，可以从已选择的课程列表中删除已选择的课程。

涉及控件：dataGrid1、btnDelete、oleDbConnection、Adapter_Selected、dataSet11.Class_Selected、dataSet11.Class_UnSelected、

涉及函数：tbxStudentID_TextChanged、btnDelete_Click、Update_Text

实现思路：该功能由两部分实现，分别是显示已选课程和从已选课程中删除某课程并将其移动到可选课程表中。设置可视化控件 dataGrid1 的属性 DataSource 为 dataSet11,属性 DataMember 为 Class_Select,即可将该表数据绑定到数据表 dataSet11.Class_Selected。而该数据表的初始行信息由 Adapter_Selected 映射产生，故可在录入学号信息后调用其 Fill 成员函数来填充数据表，进而在

dataGrid1 中显示已选课程列表。删除功能首先通过 dataGrid1 获取选定的要删除的行的序号 CID，然后将该行在 dataSet11.Class_Selected 中的列信息复制到 dataSet11.Class_UnSelected 的新行对象 row 的列属性中，并在 dataSet11.Class_UnSelected 中插入 row，然后依据行序号 CID 在 dataSet11.Class_Selected 中删除对应行。

主要代码：

从数据库中获取并显示原已选课程列表的代码在函数 tbxStudentID_TextChanged 中实现，实现该功能的代码部分如下：

```
    //获取已选课程信息
Adapter_Selected.SelectCommand.Parameters[0].Value = Convert.ToInt32(this.SID);
Adapter_Selected.Fill(this.dataSet11);
```

从已选课程列表中删除当前所选课程的代码在 btnDelete_Click 中实现，代码如下：

```
private void btnDelete_Click(object sender, EventArgs e)
{
    //获取行序号
    int CID = this.dataGrid1.CurrentRowIndex;
    //建立新行对象
    DataRow row = dataSet11.Class_UnSelected.NewRow();
    row["Class_ID"]=dataSet11.Class_Selected[CID]["Class_ID"];
    row["Name"] = dataSet11.Class_Selected[CID]["Name"];
    row["Credit"] = dataSet11.Class_Selected[CID]["Credit"];
    row["TeacherName"] = dataSet11.Class_Selected[CID]["TeacherName"];
    row["Time"] = dataSet11.Class_Selected[CID]["Time"];
    //删除原表对应行
    dataSet11.Class_Selected[CID].Delete();
    //插入新行
    dataSet11.Class_UnSelected.Rows.InsertAt(row, 0);
    //使数据表接受更新操作，防止重复删除导致的序号错误
    dataSet11.Class_Selected.AcceptChanges();
    dataSet11.Class_UnSelected.AcceptChanges();
    //更新已选课程数和已选学分信息
    Update_Text();
}
```

功能 3：显示可选择的课程列表（不包含已选择的课程），可以按规则选择 3-5 门课程，选择的课程自动从可选课程列表中移至已选课程列表。

涉及控件：dataGrid2、btnAdd、oleDbConnection、Adapter_Selected、dataSet11.Class_Selected、dataSet11.Class_UnSelected、

涉及函数：tbxStudentID_TextChanged、btnAdd_Click、Update_Text

实现思路：为保证未提交时选课的自由度（即允许当前选课状态不符合规则，毕竟每个学生初始的选课状态均不符合规则），对该功能做简化，将总学分和课程数的规则检测放在提交功能中，该功能仅保留对重复选择同一种课程的检测。该功能由两部分实现，分别是显示可选课程和从可选课程中删除某课程并将其添加到已选课程表中。设置可视化控件 dataGrid2 的属性 DataSource 为 dataSet11，属性 DataMember 为 Class_UnSelect，即可将该表数据绑定到数据表 dataSet11.Class_UnSelected。而该数据表的初始行信息由 Adapter_UnSelected 映射产生，故可在录入学号信息后调用其 Fill 成员函数来填充数据表，进而在 dataGrid2 中显示已选课程列表。添加功能首先通过 dataGrid2 获取选定的要删除的行的序号 CID 和 Name 属性，然后与 dataSet11.Class_Selected 数据表每行进行比较，若某行的 Name 属性与待添加的相同，则说明已选过此种课程，不可重复选择，弹出提示框并拒绝此次添加操作，否则进行下一步，将该行在 dataSet11.Class_UnSelected 中的列信息复制到 dataSet11.Class_Selected 的新行对象 row 的列属性中，并在 dataSet11.Class_Selected 中插入 row，然后依据行序号 CID 在 dataSet11.Class_UnSelected 中删除对应行。

主要代码：

从数据库中获取并显示原可选课程列表的代码在函数 tbxStudentID_TextChanged 中实现，现该功能的代码部分如下：

```
    //获取可选课程信息
Adapter_UnSelected.SelectCommand.Parameters[0].Value = Convert.ToInt32(this.SID);
Adapter_UnSelected.Fill(this.dataSet11);
```

从可选课程列表中删除当前所选课程并添加到已选课程列表中的代码在 btnAdd_Click中实现，代码如下：

```
private void btnAdd_Click(object sender, EventArgs e)
{
    //获取行序号
    int CID = this.dataGrid2.CurrentRowIndex;
    //验证是否重复选择同类课程
    int limit = dataSet11.Class_Selected.Count;
    for(int i=0;i<limit;i++)
    {
        if(dataSet11.Class_Selected[i]["Name"].ToString()==
            dataSet11.Class_UnSelected[CID]["Name"].ToString())
        {
            MessageBox.Show("不可重复选择同一项课程","提示" );
            return;
        }
    }
```

```
    //建立新行对象
    DataRow row = dataSet11.Class_Selected.NewRow();
    row["Class_ID"] = dataSet11.Class_UnSelected[CID]["Class_ID"];
    row["Name"] = dataSet11.Class_UnSelected[CID]["Name"];
    row["Credit"] = dataSet11.Class_UnSelected[CID]["Credit"];
    row["TeacherName"] = dataSet11.Class_UnSelected[CID]["TeacherName"];
    row["Time"] = dataSet11.Class_UnSelected[CID]["Time"];
    //删除原表对应行
    dataSet11.Class_UnSelected[CID].Delete();
    //插入新行
    dataSet11.Class_Selected.Rows.InsertAt(row, 0);
    //使数据表接受更新操作，防止重复删除导致的序号错误
    dataSet11.Class_Selected.AcceptChanges();
    dataSet11.Class_UnSelected.AcceptChanges();
    //更新已选课程数和已选学分信息
    Update_Text();
}
```

功能 4：显示当前学生选择的课程数量、总学分。

涉及控件：tbxCredit、tbxNumber、tbxSubmited、dataSet11.Class_Selected

涉及函数：Update_Text

实现思路：设置窗体对象 Form1 私有变量 number_selected 和 credit_selected，分别代表当前已选课程的总数和总学分，因为这两个变量可能因为多种操作变化，而其值由仅依赖于数据表 dataSet11.Class_Selected，所以单独设置函数 Update_Text 修改变量值，并将其在对应的可视化控件中显示，同时需要根据变量 submited 值对提交状态文本框 tbxSubmited 的内容。每次调用 Update_Text 函数，number_selected 的值修改为 dataSet11.Class_Selected 的行数，credit_selected 值修改为 dataSet11.Class_Selected 中 Credit 列的数值之和。该函数在其他所有函数中均需要被调用。

主要代码：

实现该功能的函数 Update_Text 代码如下所示：

```
private void Update_Text()
{
    //获取已选课程数
    this.number_selected = dataSet11.Class_Selected.Rows.Count;
    //获取已选课程学分
    object sumObject = dataSet11.Class_Selected.Compute("sum(Credit)", "True");
    if (sumObject == System.DBNull.Value)
        this.credit_selected = 0;
    else
```

```
            this.credit_selected = Convert.ToInt32(sumObject);
        //将课程数和课程学分显示到相应的文本框控件中
        tbxCredit.Text = this.credit_selected.ToString();
        tbxNumber.Text = this.number_selected.ToString();
        //根据变量submited值修改提交状态的文本框内容
        if (this.submited == 0)
            tbxSubmited.Text = "未提交";
        else
            tbxSubmited.Text = "已提交";
    }
```

功能 5：可以临时保存学生选择的课程信息，如果符合选课规则（3-5 门、学分：8-12 学分），则可提交选课结果。

涉及控件：Adapter_Update、Adapter_IsSubmited、btnSave、btnSubmit、dataSet11.Class_Selected

涉及函数：btnSave_Click、btnSubmit_Click、Update_Text

实现思路：该功能由两个子功能组成，分别为临时保存和提交。两个子功能主要作用类似，都需要将数据库 Selected_Class 表中原选课信息删除，再将 dataSet11.Class_Selected 中的选课信息结合当前学生学号写入数据库中。区别在于提交功能首先需要根据已选课程数和总学分是否符合要求来决定是否对数据库做出修改，还有两个子功能在修改选课表 Selected_Class 后需要对应修改学生提交状态信息，比如原未提交的学生在成功提交后需要修改提交状态为已提交，原已提交的学生在临时保存后需要修改提交状态为未提交。对数据库 Selected_Class 表的插入与删除操作通过数据适配器 Adapter_Update 进行，删除操作显式修改单一参数的 Value 值后即可执行，而插入操作在建立连接并设置第一个参数 Value 值为 SID 后仍需要遍历 dataSet11.Class_Selected 数据表的每行，提取每行 Class_ID 列作为插入语句第二个参数 Value 值，并多次执行插入语句直到遍历完成，才可以关闭连接。提交功能的前置操作仅需要对 number_selected 和 credit_selected 做出范围判断即可。对于提交状态的修改操作，则是根据当前功能和提交状态，使用数据适配器 Adapter_IsSubmited 对数据库 Submited 表做出相应修改，并修改变量 submited 值

主要代码：

临时保存功能在函数 btnSave_Click 中实现，代码如下：

```
private void btnSave_Click(object sender, EventArgs e)
{
    DialogResult dr = MessageBox.Show("确定保存选课信息（会覆盖原有提交信息）？", "提示", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
    if (dr == DialogResult.No)
```

```
                return;

            //从数据库中删除原选课记录
            Adapter_Update.DeleteCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
            Adapter_Update.DeleteCommand.Connection.Open();
            Adapter_Update.DeleteCommand.ExecuteNonQuery();
            Adapter_Update.DeleteCommand.Connection.Close();

            //根据已选课列表向数据库中插入选课记录
            int limit = dataSet11.Class_Selected.Count;
            Adapter_Update.InsertCommand.Connection.Open();
            Adapter_Update.InsertCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
            for (int i=0;i<limit;i++)
            {
                String CID = dataSet11.Class_Selected[i]["Class_ID"].ToString();
                Adapter_Update.InsertCommand.Parameters[1].Value =
Convert.ToInt32(CID);
                Adapter_Update.InsertCommand.ExecuteNonQuery();
            }
            Adapter_Update.InsertCommand.Connection.Close();

            //根据原提交状态判断是否要修改提交状态
            if (this.submited == 1)
            {
                Adapter_IsSubmited.DeleteCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
                Adapter_IsSubmited.DeleteCommand.Connection.Open();
                Adapter_IsSubmited.DeleteCommand.ExecuteNonQuery();
                Adapter_IsSubmited.DeleteCommand.Connection.Close();
                this.submited = 0;
            }
            MessageBox.Show("保存成功", "提示");

            //刷新提交状态文本框内容
            Update_Text();
        }
```

提交功能在函数 btnSubmit_Click 中实现，代码如下：

```
        private void btnSubmit_Click(object sender, EventArgs e)
        {
            //判断当前已选课程数和总学分是否符合要求
            if (this.number_selected < 3 || this.number_selected > 5 ||
                this.credit_selected < 8 || this.credit_selected > 12)
```

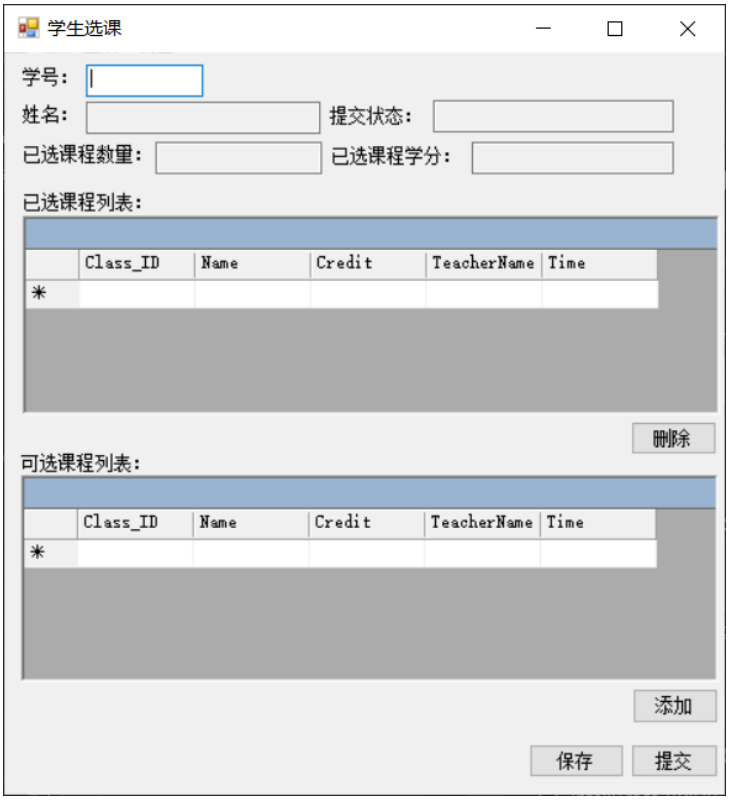
```
            {
                MessageBox.Show("不符合要求，提交失败", "提示");
                return;
            }

            //从数据库中删除原选课记录
            Adapter_Update.DeleteCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
            Adapter_Update.DeleteCommand.Connection.Open();
            Adapter_Update.DeleteCommand.ExecuteNonQuery();
            Adapter_Update.DeleteCommand.Connection.Close();

            //根据已选课列表向数据库中插入选课记录
            int limit = dataSet11.Class_Selected.Count;
            Adapter_Update.InsertCommand.Connection.Open();
            Adapter_Update.InsertCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
            for (int i = 0; i < limit; i++)
            {
                String CID = dataSet11.Class_Selected[i]["Class_ID"].ToString();
                Adapter_Update.InsertCommand.Parameters[1].Value =
Convert.ToInt32(CID);
                Adapter_Update.InsertCommand.ExecuteNonQuery();
            }
            Adapter_Update.InsertCommand.Connection.Close();

            //根据原提交状态判断是否要修改提交状态
            if (this.submited==0)
            {
                Adapter_IsSubmited.InsertCommand.Parameters[0].Value =
Convert.ToInt32(this.SID);
                Adapter_IsSubmited.InsertCommand.Connection.Open();
                Adapter_IsSubmited.InsertCommand.ExecuteNonQuery();
                Adapter_IsSubmited.InsertCommand.Connection.Close();
                this.submited = 1;
            }
            MessageBox.Show("提交成功", "提示");

            //刷新提交状态文本框内容
            Update_Text();

        }
```

以上代码的详细情况可在附件项目文件夹 Course_Selection 中 Form1.cs 文件中查看。

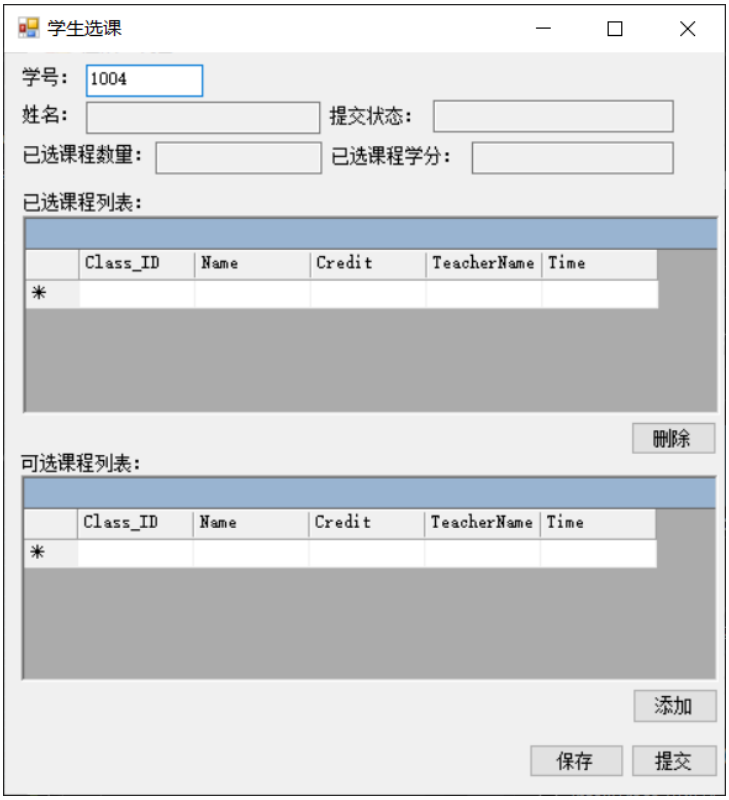
综上，实现了对数据库应用程序整体和各功能的构建。

2. 实验结果

（1） 程序初始界面如下图所示



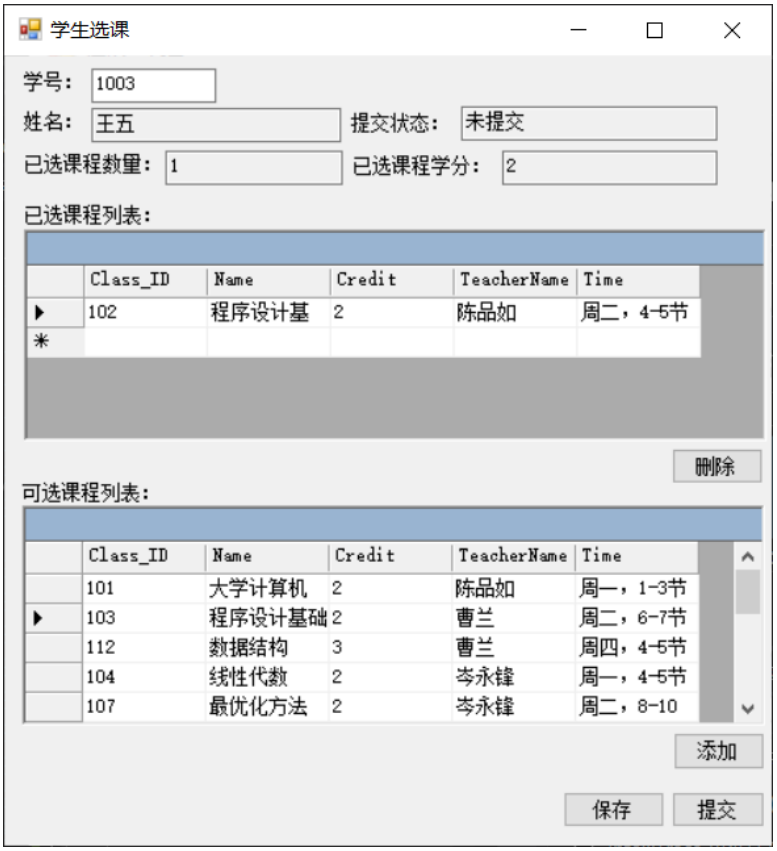
（2） 输入学号，若不存在该学号，则无反应



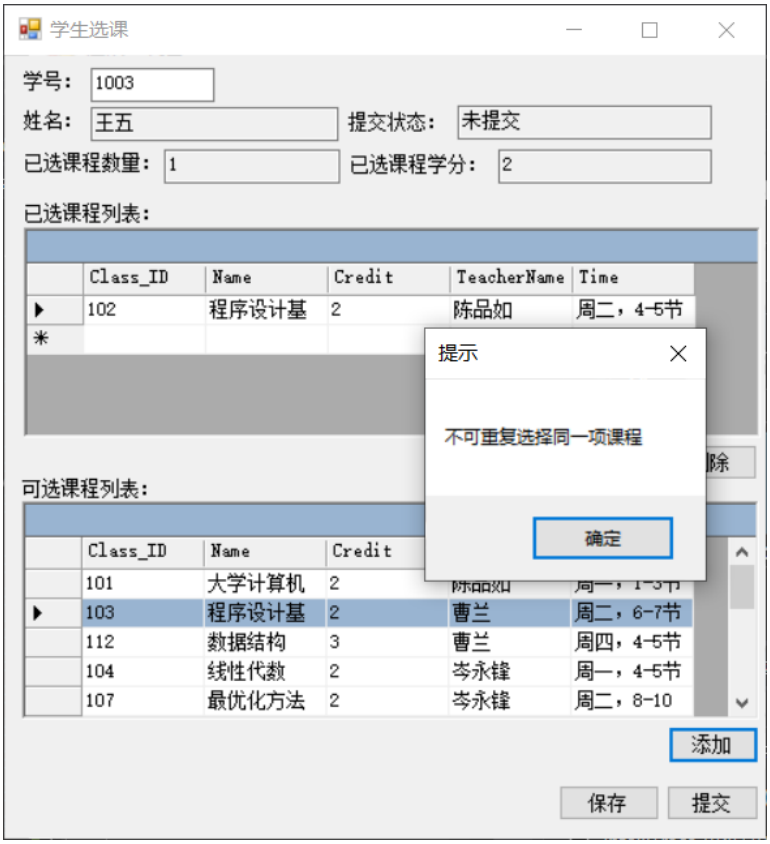
若存在该学号，则显示该学生各类信息，以下分别是两个学号对应的显示界面

（3） 添加课程前界面如下图所示，并在可选课程列表中选择某项

点击添加，界面变化如下



添加同一种课程，弹出提示

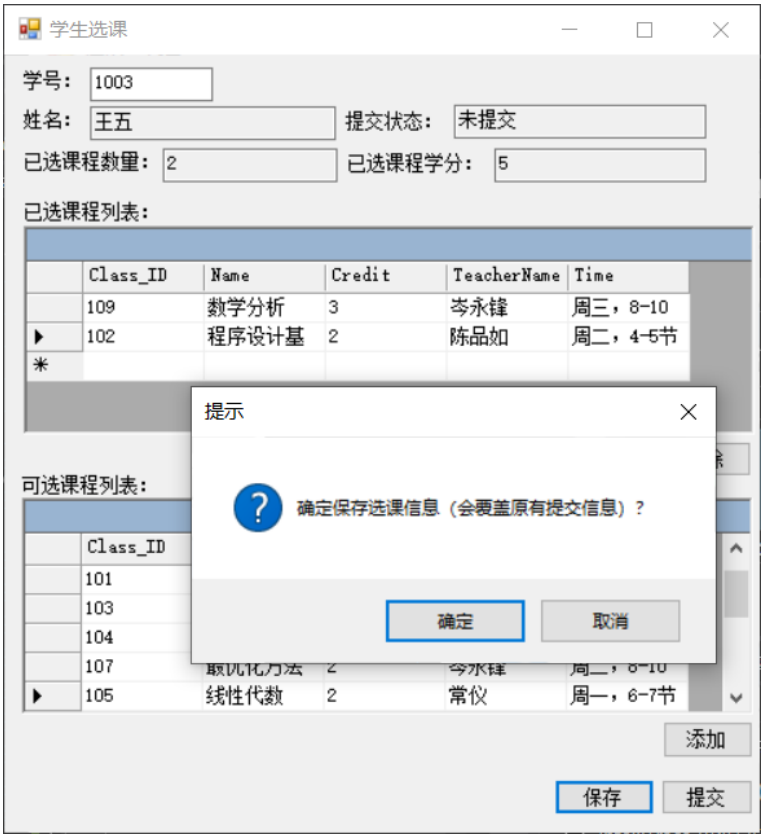


（4） 删除课程前界面如下所示，并选择已选课程列表中某项

点击删除，界面变化如下

（5） 保存前界面如下图所示

点击保存，弹出提示，点击确认



关闭并重新打开程序，选择同一学号，界面信息与保存前相同

（6） 点击提交，课程数量和学分不符合要求，弹出提示

修改已选课程列表，再次点击提交，弹出提示

提交状态发生改变

关闭并重新打开程序，选择同一学号，界面信息与保存前相同。

3. 实验总结

（1） 问题分析

最初设计中为实现查询学生姓名和提交状态功能专门设计了一个数据适配器和数据表，但实际上这两个信息较为简单，且查询只需要简单语句，并且在整个程序运行过程中使用不多，故仅用 OleDbCommand 新建临时命令对数据源查询，并将查询值保存到成员变量中。

最初设计的中有第三个数据表专门存储已选课程列表，在后续的添加删除操作中不做改变，在最终提交或保存时用于检索数据库选课表来删除原有记录。这种多余设计的原因是开始并不熟悉数据适配器机制，只会使用 OleDbCommand，对于适配器的插入删除语句使用僵硬，在构建项目的过程中逐渐熟悉了适配器操作，并直接配置了适配器插入删除语句，将数据表数量减少为 2 个，并提高了保存或提交步骤中删除操作效率。

（2） 后续改进

对于提交与保存时对数据库表的插入操作，本程序选择对数据表逐行处理，多次调用适配器插入语句。依据官方文档，InsertCommand 存在批处理属性，但并未作出详细说明，网络上也未找到相关案例，故未对此处进行优化，后续可进一步查阅相关资料对此作出改进,或者考虑修改更新数据库的方式，显式设置 Adapter_Select 的 Update 语句，以此做到批量处理效果。

在本实验中仅用文本方式保存课程时间，未能对课程冲突的情况加以判断。可以考虑构建课程表功能，可视化显示学生选择的课程在各时间段上的信息，并对对课程时间冲突的情况加以判断并给出反馈。

在本实验中多个数据适配器仅仅使用了选择语句从数据库中读取记录，不涉及修改，所以可以考虑使用 DataReader 实现类似功能，提高程序运行速度。

本程序页面 UI 均为最简化表示，可以考虑在各个方面美化和修饰。

4. 附录

手工录入数据库基础信息表：

录入学生信息如下：

Student_ID	Name
1001	张三
1002	李四
1003	王五

录入教师信息如下：

Teacher_ID	Name
2001	陈品如
2002	曹兰
2003	岑永锋
2004	常仪
2005	蔡勇
2006	崔嘉敏
2007	柴立
2008	成可
2009	池临泽

录入课程种类信息如下：

Course_ID	Name	Credit
1	大学计算机	2
2	程序设计基础	2
3	线性代数	2
4	数值分析	2
5	最优化方法	2
6	数学分析	3
7	大学物理	3
8	数据结构	3
9	数据库原理	3

录入课程信息如下

Class_ID	Course_ID	Teacher_ID	Time
101	1	2001	周一，1-3 节
102	2	2001	周二，4-5 节
103	2	2002	周二，6-7 节
104	3	2003	周一，4-5 节
105	3	2004	周一，6-7 节
106	4	2004	周一，8-10 节
107	5	2003	周二，8-10 节

108	6	2005	周三，1-3 节
109	6	2003	周三，8-10 节
110	7	2006	周三，4-5 节
111	7	2007	周三，6-7 节
112	8	2002	周四，4-5 节
113	8	2008	周四，6-7 节
114	9	2009	周五，1-3 节