**数据库专题训练 (2020年秋)**

|  |
| --- |
| **序号：** |

**数据库专题训练**

**实验报告**

**系别：**

**班级：**

**姓名：**

**学号：**

|  |
| --- |
| **成绩：**  **评语：**  **指导教师签字： 日期：** |

目录

[1 Transact-SQL编程 3](#_Toc81742966)

[1.1实验题目 3](#_Toc81742967)

[1.2 实验目的和要求 3](#_Toc81742968)

[1.3 实验步骤 3](#_Toc81742969)

[1.4 实验结果 3](#_Toc81742970)

[1.5 实验总结 8](#_Toc81742970)

[2 存储过程 9](#_Toc81742971)

[2.1实验题目 9](#_Toc81742967)

[2.2 实验目的和要求 9](#_Toc81742968)

[2.3 实验步骤 9](#_Toc81742969)

[2.4 实验结果 1](#_Toc81742970)0

[2.5 实验总结 1](#_Toc81742970)3

[3 触发器 1](#_Toc81742971)4

[3.1实验题目 1](#_Toc81742967)4

[3.2 实验目的和要求 14](#_Toc81742968)

[3.3 实验步骤 14](#_Toc81742969)

[3.4 实验结果 1](#_Toc81742970)5

[3.5 实验总结 1](#_Toc81742970)9

[4 数据库应用系统开发 2](#_Toc81742971)0

[4.1实验题目 2](#_Toc81742967)0

[4.2 实验目的和要求 2](#_Toc81742968)0

[4.3 实验步骤 2](#_Toc81742969)0

[4.4 实验结果 2](#_Toc81742970)1

[4.5 实验总结 38](#_Toc81742970)

# Transact-SQL编程

## 实验题目

掌握tsql的基本语法，并熟悉相关概念。

## 实验目的和要求

1. 掌握常用函数的使用方法。
2. 掌握流程控制语句的使用方法。

## 实验步骤

1. 启动SQL　Server 2005 查询编辑器。
2. 应用转换函数。
3. 应用聚合函数。
4. 应用字符串函数。
5. 应用IF··· ELSE语句。
6. 应用WHILE语句。

## 实验结果

1. 创建一个名为XSCJ的数据库，在XSCJ数据库中创建如下关系表：学生信息表S，课程表C，成绩表SC。

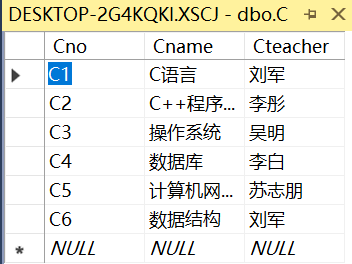
学生基本信息表S：

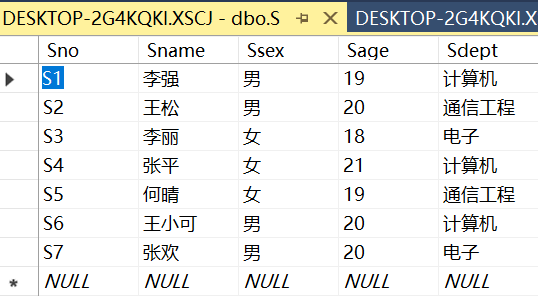
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Sname** | **Ssex** | **Sage** | **Sdept** |
| S1 | 李强 | 男 | 19 | 计算机 |
| S2 | 王松 | 男 | 20 | 通信工程 |
| S3 | 李丽 | 女 | 18 | 电子 |
| S4 | 张平 | 女 | 21 | 计算机 |
| S5 | 何晴 | 女 | 19 | 通信工程 |
| S6 | 王小可 | 男 | 20 | 计算机 |
| S7 | 张欢 | 男 | 20 | 电子 |

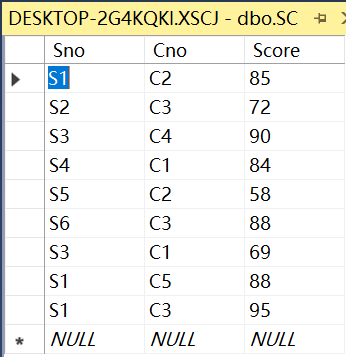
课程信息表C： 学生成绩表SC：

|  |  |  |
| --- | --- | --- |
| **Sno** | **Cno** | **Score** |
| S1 | C2 | 85 |
| S2 | C3 | 72 |
| S3 | C4 | 90 |
| S4 | C1 | 84 |
| S5 | C2 | 58 |
| S6 | C3 | 88 |
| S3 | C1 | 69 |
| S1 | C5 | 88 |
| S1 | C3 | 95 |

|  |  |  |
| --- | --- | --- |
| **Cno** | **Cname** | **Cteacher** |
| C1 | C语言 | 刘军 |
| C2 | C++程序设计 | 李彤 |
| C3 | 操作系统 | 吴明 |
| C4 | 数据库 | 李白 |
| C5 | 计算机网络 | 苏志朋 |
| C6 | 数据结构 | 刘军 |

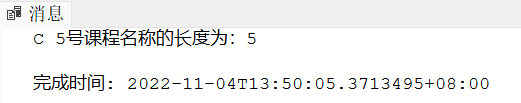






1. 应用转换函数
2. 打开“SQL　Server Manageement Studio” 窗口。
3. 单击“标准”工具栏上的“新建查询”按钮，打开“查询编辑器”窗口。
4. 在窗口内直接输入以下语句，求课程表中课程号为“c5”的课程名称的长度，并输入结果。

|  |
| --- |
| USE XSCJ  DECLARE @length int  SELECT @length=LEN(Cname) FROM C WHERE Cno='c5'  Print 'C 5号课程名称的长度为：'+ CONVERT(varchar(10),@length) |



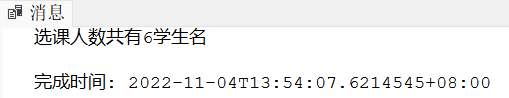
4）单击“SQL编辑器”　工具栏上的“分析”按钮，检查输入的T-SQL语句是否有语法错误。如果有语法错误，则进行修改，直到没有语法错误为止。

5）确保无语法错误后，单击“SQL编辑器”工具栏上的“执行”按钮。

1. 应用聚合函数

　 1）在“查询编辑器”窗口内输入以下语句，统计选课成绩表SC表中的学生人数，并输出结果。

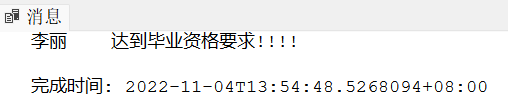
|  |
| --- |
| USE XSCJ  DECLARE @Count int  SELECT @Count = COUNT(distinct Sno) FROM SC  PRINT '选课人数共有'+CAST(@Count AS varchar(10))+'学生名' |



1. 应用IF···ELSE 语句

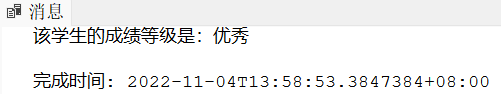
1） 在“查询编辑器”窗口内输入以下语句，查询学号为“S 3” 的学生的平均分是否超过了75分， 若超过则输出“ X X 达到毕业资格要求”， 否则输出 “XX不能按期毕业”。

|  |
| --- |
| USE XSCJ  DECLARE @sno char(6), @sname char(8)  SET @sno ='s3'  IF (SELECT AVG(score) FROM SC WHERE Sno=@sno) >=75  BEGIN  SELECT @sname =Sname FROM S WHERE Sno=@sno  PRINT @Sname+'达到毕业资格要求!!!!'  END  ELSE  PRINT @sname+'XX不能按期毕业！' |



1. 应用CASE语句
   1. 声明变量@cj为float型，变量@str为varchar(60)， @cj赋值90。
   2. 编写程序确定该分数对应的等级，规则如下：大于等于60并小于70为‘及格’，大于等于70并小于80为‘中等’，大于等于80并小于90为‘良好’，大于等于90并小于等于100为‘优秀’，并且分数应在0到100之间，否则就会提示“成绩错误，成绩应在0到100之间”。

|  |
| --- |
| DECLARE @cj float, @str varchar(60)  SET @cj=90  SET @str=  CASE  WHEN @cj>100 or cj<0 THEN ‘成绩输入错误，成绩应在0到100之间’  WHEN @cj>=60 and cj<70 THEN ‘及格’  WHEN @cj>=70 and cj<80 THEN ‘中等’  WHEN @cj>=80 and cj<90 THEN ‘优良’  WHEN @cj>=90 and cj<=100 THEN ‘优秀’  ELSE ‘不及格’  END  PRINT ‘该学生的成绩等级是：’+@str  GO |



1. 综合应用

在XSCJ数据库中，按照如下脚本创建一个用户表，并向其中插入3行记录。然后编写一个程序，能实现以下功能：对于给定变量@hyuser=’stu1’，@hypwd=’111111’，①判断该用户是否存在；②该用户密码是否正确；③用户名和密码都正确则提示成功登录信息，否则提示错误信息。

USE XSCJ

CREATE TABLE hyuser(

hyid int identity(1,1) primary key,

hyname varchar(50) unique,

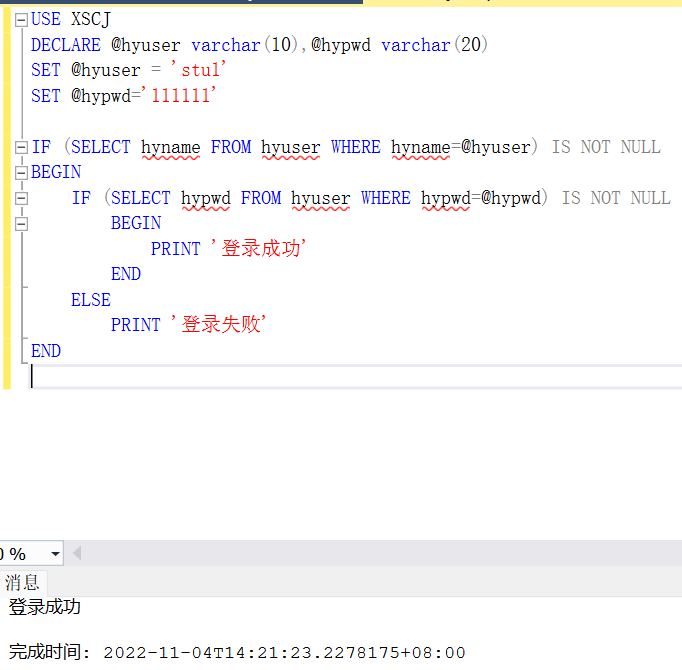
hypwd varchar(50)

)

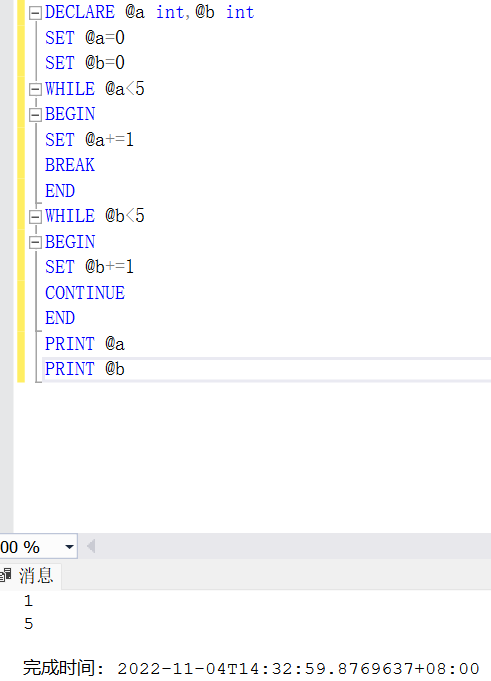
INSERT INTO hyuser(hyname,hypwd) VALUES(‘stu1’, ‘111111’)

INSERT INTO hyuser(hyname,hypwd) VALUES(‘stu2’, ‘222222’)

INSERT INTO hyuser(hyname,hypwd) VALUES(‘stu3’, ‘333333’)

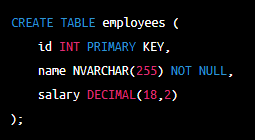


1. 用程序举例说明BREAK语句和CONTINUE语句的区别。



## 实验总结

1. SQL 常用的语法：
2. SELECT 语句：用于查询数据，例如：SELECT \* FROM employees;
3. INSERT 语句：用于插入数据，例如：INSERT INTO employees (id, name, salary) VALUES (1, 'John Doe', 50000);
4. UPDATE 语句：用于更新数据，例如：UPDATE employees SET salary = 55000 WHERE id = 1;
5. DELETE 语句：用于删除数据，例如：DELETE FROM employees WHERE id = 1;
6. CREATE 语句：用于创建数据库对象，例如：



1. ALTER 语句：用于修改数据库对象，例如：ALTER TABLE employees ADD age INT;
2. DROP 语句：用于删除数据库对象，例如：DROP TABLE employees;

除了以上常用的语法，T-SQL 还支持视图、存储过程、触发器、事务等特性，可以提供更丰富的数据库操作能力。

# 存储过程

## 实验题目

掌握tsql存储过程的创建和使用方法。

## 实验目的和要求

1. 目的
   * 掌握存储过程的创建方法。
   * 掌握存储过程的使用方法。
2. 重点
   * 创建无参数的存储过程。
   * 创建有参数的存储过程。
   * 执行存储过程的方法。
   * 查看存储过程的方法。
   * 修改存储过程的方法。
   * 删除存储过程的方法。

## 实验步骤

设定已有一个名为XSCJ的数据库，在XSCJ数据库中创建如下关系表：学生信息表S，课程表C，成绩表SC。

学生基本信息表S：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Sname** | **Ssex** | **Sage** | **Sdept** |
| S1 | 李强 | 男 | 19 | 计算机 |
| S2 | 王松 | 男 | 20 | 通信工程 |
| S3 | 李丽 | 女 | 18 | 电子 |
| S4 | 张平 | 女 | 21 | 计算机 |
| S5 | 何晴 | 女 | 19 | 通信工程 |
| S6 | 王小可 | 男 | 20 | 计算机 |
| S7 | 张欢 | 男 | 20 | 电子 |

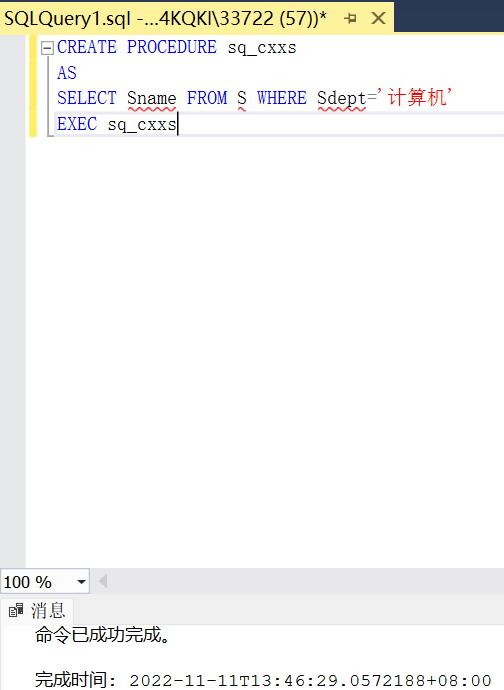
课程信息表C： 学生成绩表SC：

|  |  |  |
| --- | --- | --- |
| **Sno** | **Cno** | **Score** |
| S1 | C2 | 85 |
| S2 | C3 | 72 |
| S3 | C4 | 90 |
| S4 | C1 | 84 |
| S5 | C2 | 58 |
| S6 | C3 | 88 |
| S3 | C1 | 69 |
| S1 | C5 | 88 |
| S1 | C3 | 95 |

|  |  |  |
| --- | --- | --- |
| **Cno** | **Cname** | **Cteacher** |
| C1 | C语言 | 刘军 |
| C2 | C++程序设计 | 李彤 |
| C3 | 操作系统 | 吴明 |
| C4 | 数据库 | 李白 |
| C5 | 计算机网络 | 苏志朋 |
| C6 | 数据结构 | 刘军 |

## 实验结果

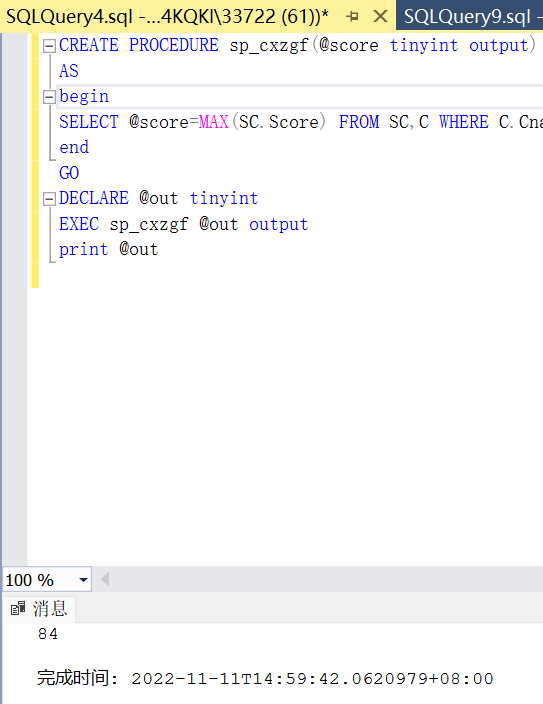
1. 创建一个名为sp\_cxxs的存储过程，要求该存储过程列出计算机系所有学生的姓名。



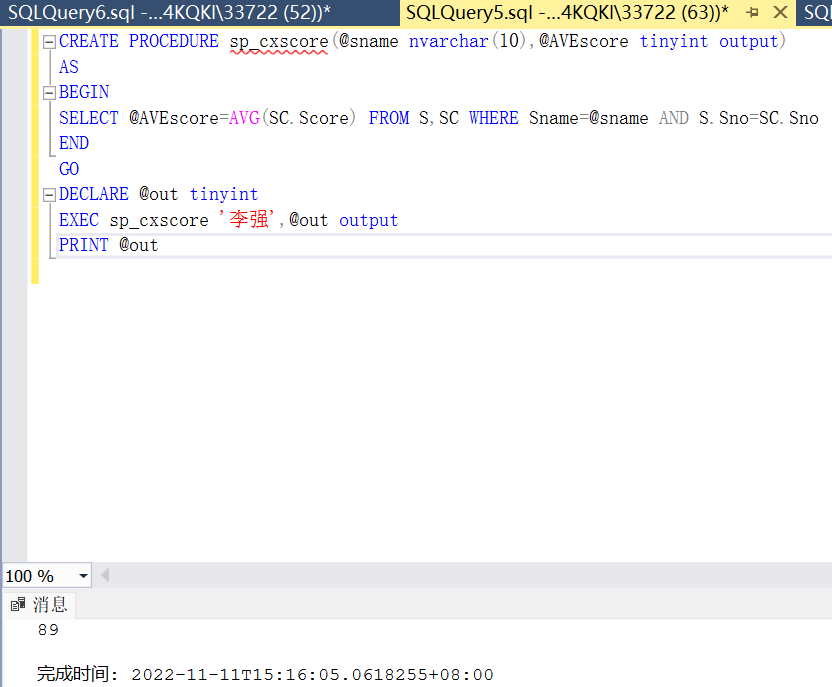
1. 创建一个名为sp\_cxcj的存储过程，要求该存储过程带一个输入参数，用于接收学生姓名。执行该存储过程时，将根据输入的学生姓名列出该生的所修课程名称和分数。



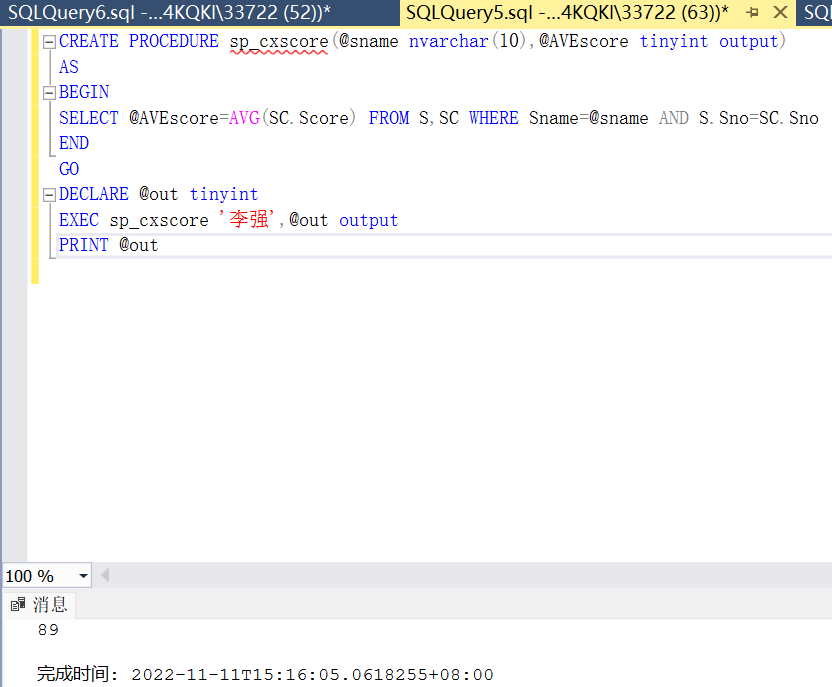
1. 创建一个名为sp\_cxzgf的存储过程，要求该存储过程带一个输出参数，用于返回”C语言”课程的最高分。



1. 创建一个名为sp\_cxscore的存储过程，要求该存储过程带一个输入参数和一个输出参数，输入参数用于接收学生姓名，输出参数用于返回该生的各科平均分。



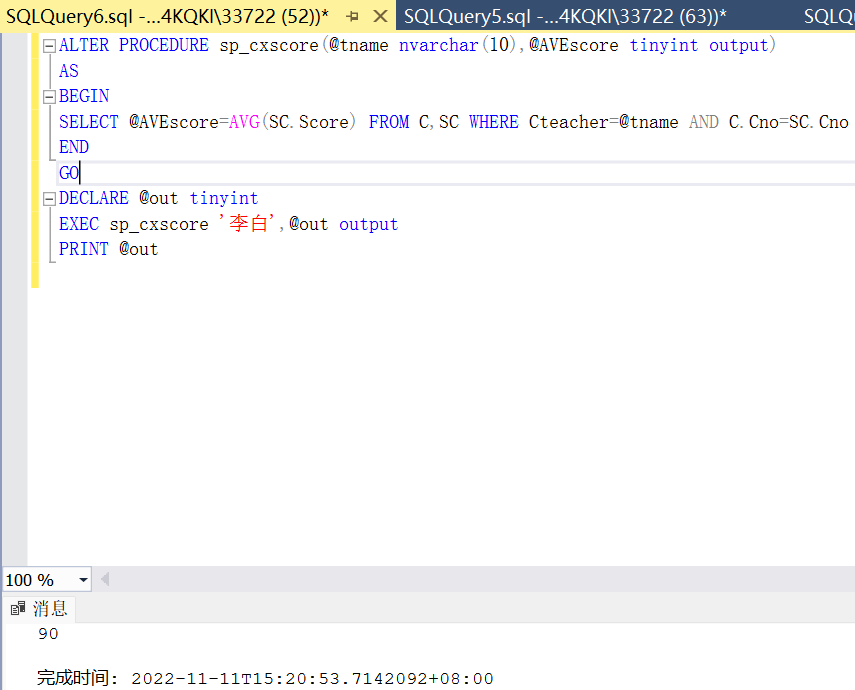
1. 执行题4所创建的存储过程，并打印输出参数的返回值。输入参数为‘李强’。



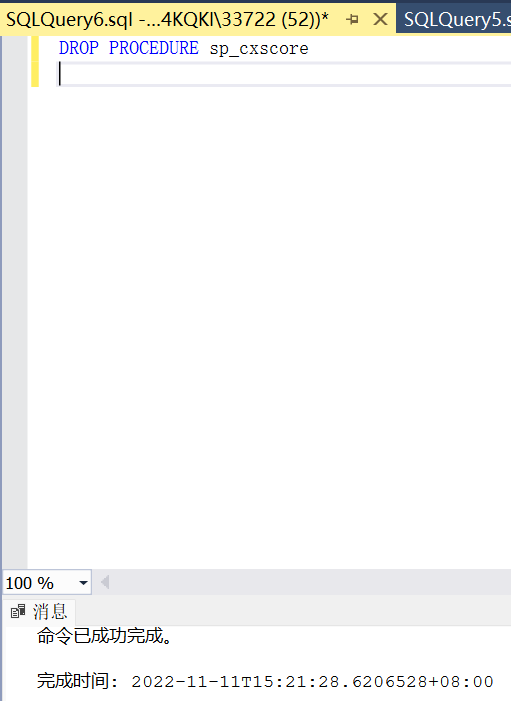
1. 使用系统表查看XSCJ数据库中名为sp\_cxxs的存储过程的定义信息。



1. 修改题4所创建的存储过程，要求新的存储过程带一个输入参数和一个输出参数，输入参数用于接收教师姓名，输出参数用于返回该教师所教课程的平均分。



1. 删除题7所给出存储过程。

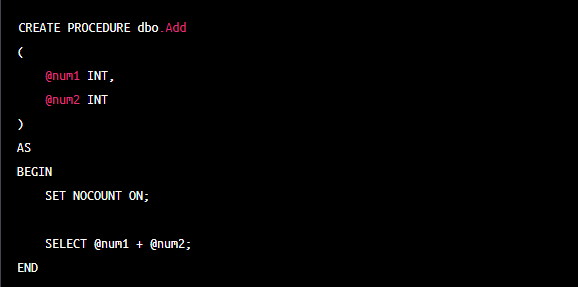


## 实验总结

T-SQL 存储过程的优点包括：

* 1. 提高代码的重用性，可以通过调用同一个存储过程来执行相同的操作，而无需重复编写代码；
  2. 提高代码的安全性，存储过程的代码只能被数据库管理员授权的用户执行；
  3. 提高数据库的性能，存储过程的代码会在数据库服务器端执行，并且会被数据库编译和优化，因此可以提高执行效率。

例如，下面是一个简单的 T-SQL 存储过程示例，该存储过程可以接受两个参数，并返回输入参数的和：



# 3. 触发器

## 实验题目

设定已有一个名为XSCJ的数据库，在XSCJ数据库中创建如下关系表：学生信息表S，课程表C，成绩表SC。根据下述目的和要求及步骤进行实验。

## 实验目的和要求

1. 目的
   * 掌握触发器的分类及工作原理。
   * 掌握触发器的创建方法。
   * 掌握触发器的使用方法。
2. 重点
   1. 创建insert触发器。
   2. 创建delete触发器。
   3. 创建update触发器
   4. 执行触发器的方法。
   5. 查看触发器。
   6. 禁止和启动 触发器。

## 实验步骤

设定已有一个名为XSCJ的数据库，在XSCJ数据库中创建如下关系表：学生信息表S，课程表C，成绩表SC。

学生基本信息表S：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Sname** | **Ssex** | **Sage** | **Sdept** |
| S1 | 李强 | 男 | 19 | 计算机 |
| S2 | 王松 | 男 | 20 | 通信工程 |
| S3 | 李丽 | 女 | 18 | 电子 |
| S4 | 张平 | 女 | 21 | 计算机 |
| S5 | 何晴 | 女 | 19 | 通信工程 |
| S6 | 王小可 | 男 | 20 | 计算机 |
| S7 | 张欢 | 男 | 20 | 电子 |

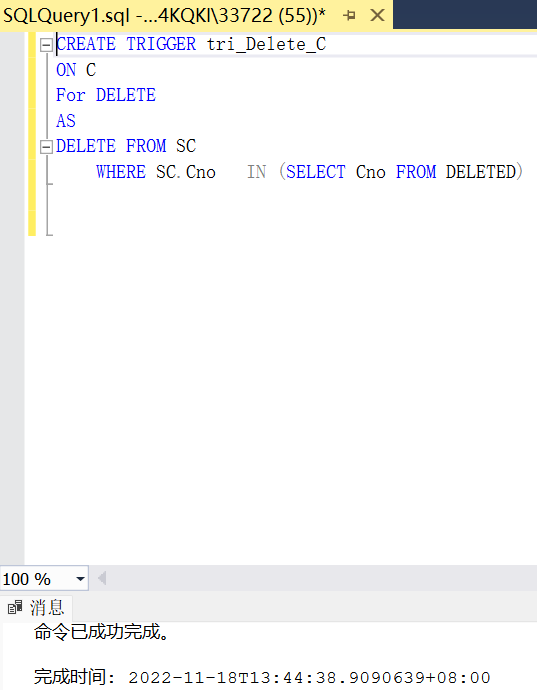
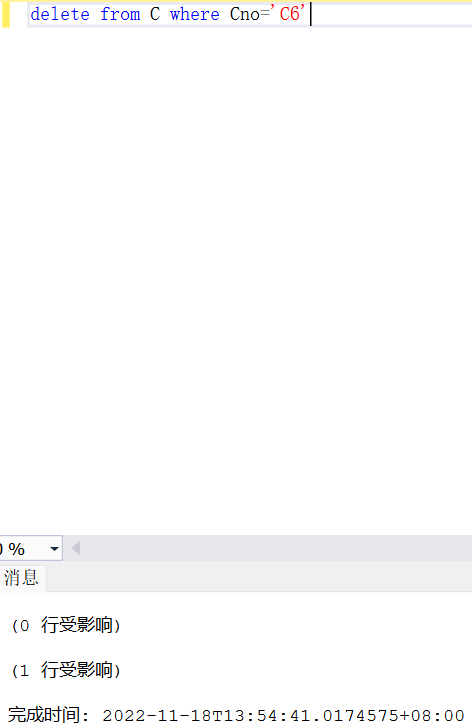
课程信息表C： 学生成绩表SC：

|  |  |  |
| --- | --- | --- |
| **Sno** | **Cno** | **Score** |
| S1 | C2 | 85 |
| S2 | C3 | 72 |
| S3 | C4 | 90 |
| S4 | C1 | 84 |
| S5 | C2 | 58 |
| S6 | C3 | 88 |
| S3 | C1 | 69 |
| S1 | C5 | 88 |
| S1 | C3 | 95 |

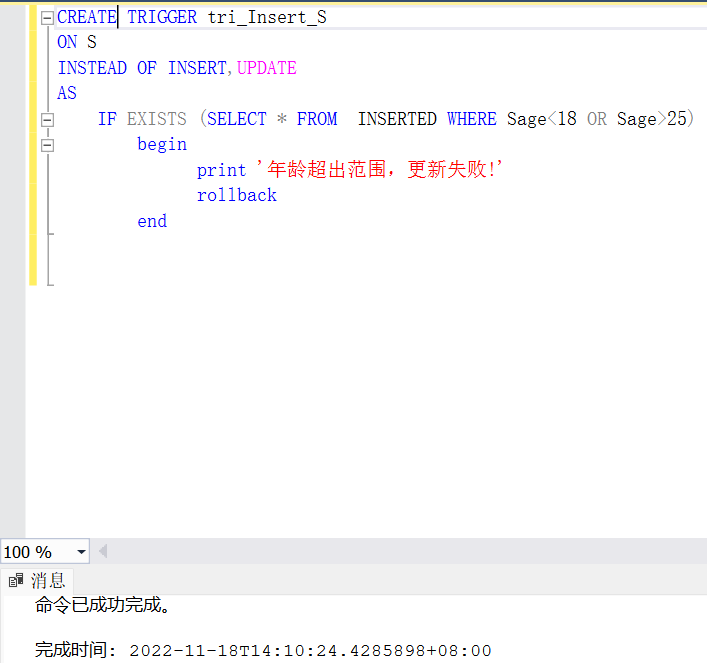
|  |  |  |
| --- | --- | --- |
| **Cno** | **Cname** | **Cteacher** |
| C1 | C语言 | 刘军 |
| C2 | C++程序设计 | 李彤 |
| C3 | 操作系统 | 吴明 |
| C4 | 数据库 | 李白 |
| C5 | 计算机网络 | 苏志朋 |
| C6 | 数据结构 | 刘军 |

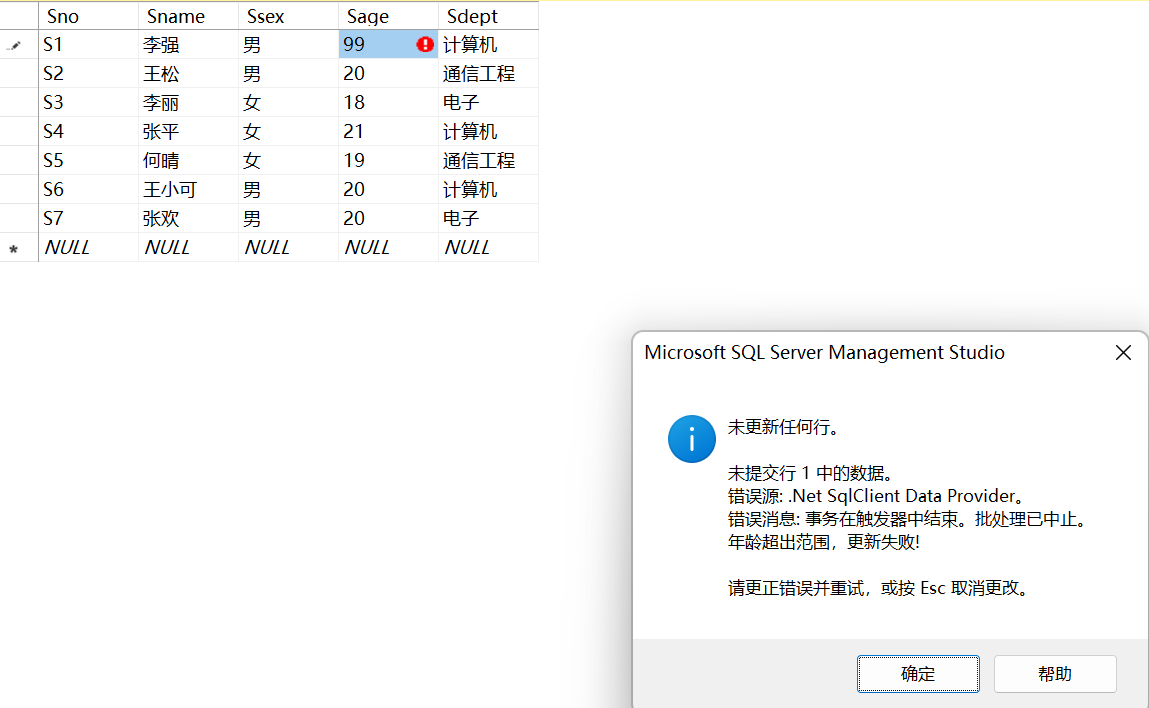
## 实验结果

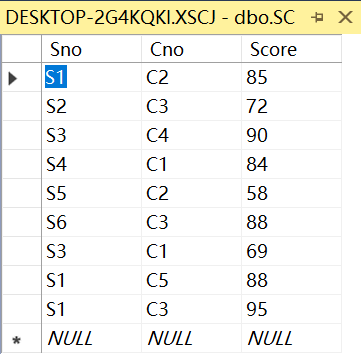
1. 创建一个名为 tri\_Delete\_C的触发器，测试该触发器的执行情况，并给出出实验结果。该触发器的作用为：当在课程表C中删除某门课程时，则所有选修该课程的记录也全部删除。

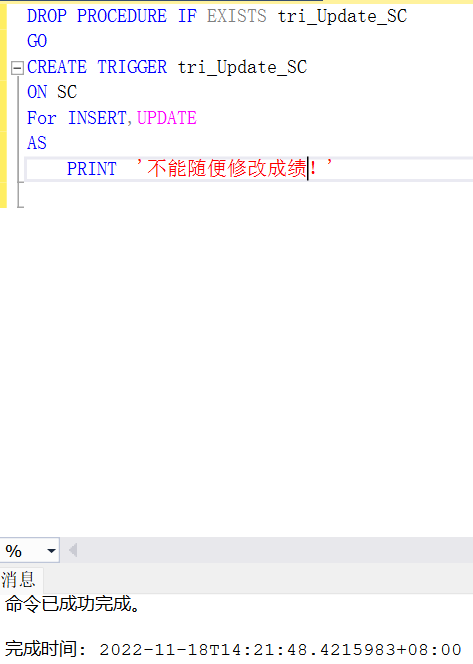


1. 创建一个名为tri\_Insert\_S的触发器，测试该触发器的执行情况，并给出实验结果。当插入的新记录中Sage的值不是18至25之间的数值时，就激活该触发器，撤销该插入操作，并给出错误提示。

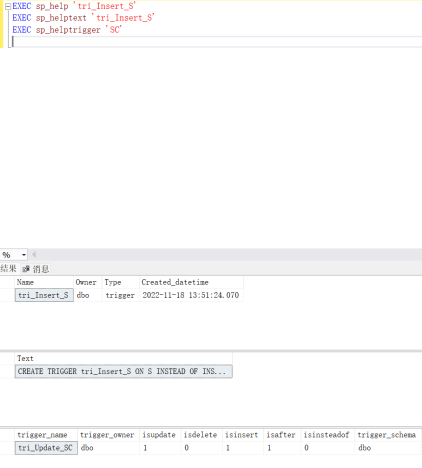




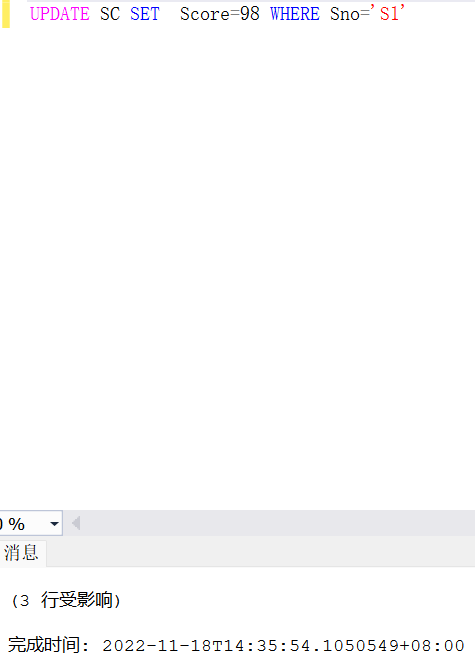
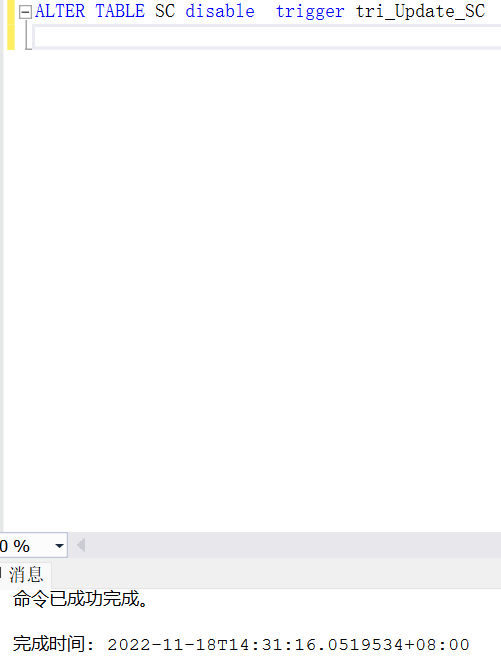
1. 创建一个名为tri\_Update\_SC的触发器，要求：（1）首先判断数据库中是否已经存在名为tri\_Update\_SC的触发器，如果存在，首先删除，再创建。（2）当试图修改SC表中的学生成绩时，给出不能随便修改成绩的信息提示。



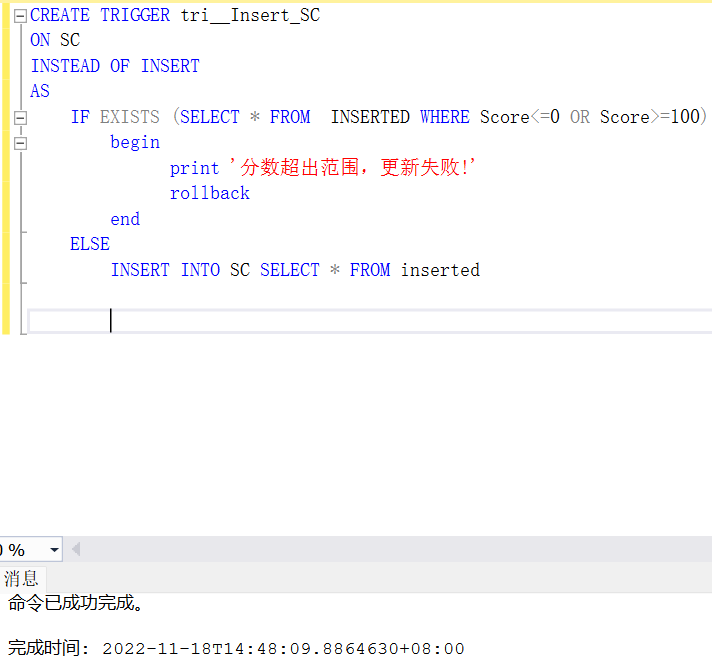
1. 使用系统存储过程sp\_help，sp\_helptext，sp\_helptrigger查看触发器相关信息。给出显示结果。



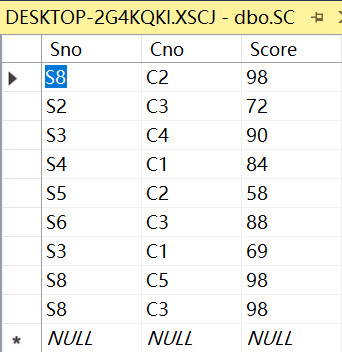
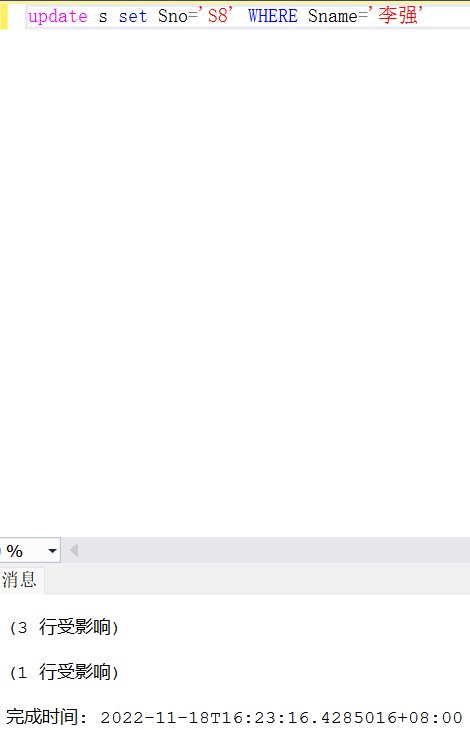
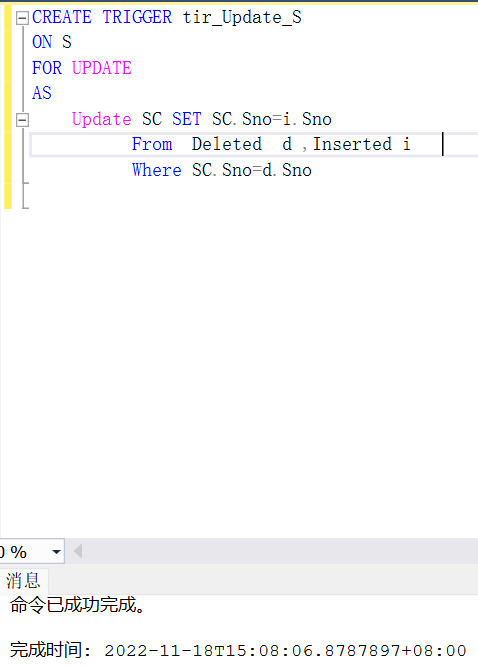
1. 禁用tri\_Update\_SC触发器，并演示禁用该触发器后Update的执行情况。



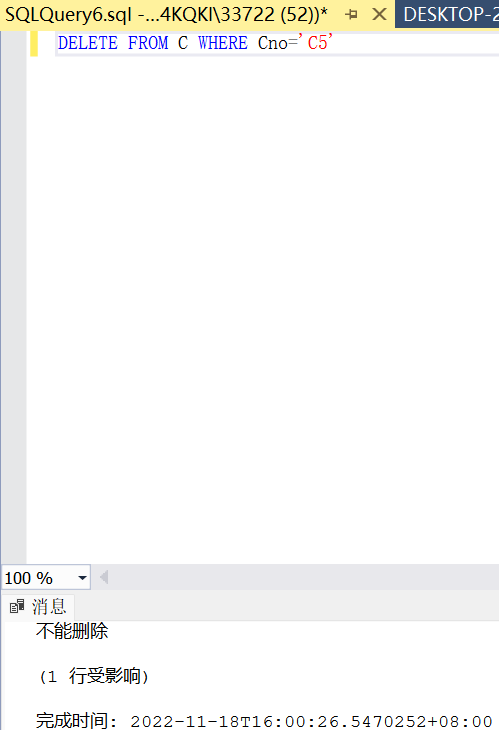
1. 在SC表上创建一个instead of类型的触发器，触发器名称为：tri\_\_Insert\_SC。实现：当向表SC插入记录时检查分数的合理性，分数必须在0到100之间，如果不合理就拒绝插入，并给出信息提示。

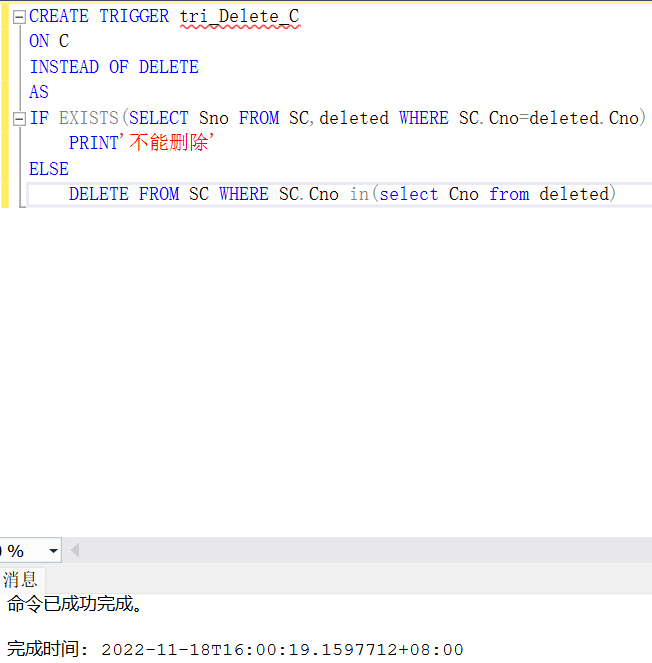


1. 创建一个名为tir\_Update\_S的触发器，实现：当学生基本表S中的学号Sno发生变化时，SC选课记录中对应的学号也发生改变。并通过测试数据验证该触发器的执行情况。



1. 创建一个名为tri\_Delete\_C 的触发器，实现：删除一门课程时候，首先判断该课程有否有人选，如果有人选，则不能删除并通过测试数据验证该触发器的执行情况。

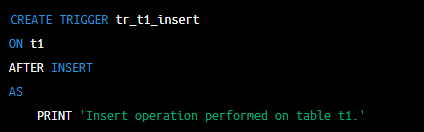




## 实验总结

1. QL Server 2005提供了两种主要机制来强制执行业务规则和数据完整性：约束和触发器。就本质而言，触发器也是一种存储过程，但是是一种特殊类型的存储过程。触发器只要满足一定的条件，就可以触发完成各种简单和复杂的任务，可以帮助我们更好地维护数据库中数据的完整性。
2. 触发器优点
   1. 自动执行：立即被激活，对比其他的一些同类产品不用手动提交。
   2. 能够对数据库中的相关表实现级联更改：基于一个表创建的，但是可以针对多个表进行操作，实现数据库中相关表的级联更改。
   3. 可以实现比CHECK约束更为复杂的数据完整性约束
3. T-SQL 触发器分为两种类型：
   1. DML 触发器：DML 触发器会在数据库对象上执行 INSERT、UPDATE 或 DELETE 操作时触发；
   2. DDL 触发器：DDL 触发器会在数据库对象上执行 CREATE、ALTER 或 DROP 操作时触发。

* 例如，下面是一个简单的 T-SQL 触发器示例，该触发器会在表 t1 上执行 INSERT 操作时触发，并在触发时输出一条提示信息：



上面的代码中，使用了 CREATE TRIGGER 语句来创建一个名为 tr\_t1\_insert 的触发器，该触发器会在表 t1 上执行 INSERT 操作后触发。然后在 AS 子句中，指定了触发时执行的代码，即 PRINT 'Insert operation performed on table t1.'。

# 数据库应用系统开发

## 实验题目

根据下述要求设计一个教务处系统，包含学生、学院和教务处三大部分。

## 实验目的和要求

1. 通过本实验使学生掌握结构化需求分析的方法、过程和相应的文档内容与格式。特别是熟悉数据流程图、数据字典和IPO图三个核心技术的应用。
2. 通过本实验使学生通过掌握选课管理系统的实现。

## 实验步骤

### 步骤1：需求分析------大学生选课管理系统

**1．实验内容说明：**

* 教务处的管理人员录入全校的课程基本信息和本学期的课程授课教师、地点、时间；
* 在学生入学的时候，学院的管理人员录入学生基本信息；
* 学生每学期自己上网登录系统选课，选课成功后信息存入数据库中，学生自己可以查询选课的情况；
* 学生选课不成功的情况有：
  + 所选课程的先修课还没有记录，系统提示“缺先修课，选课失败”；
  + 本学期所选课程的上课时间有冲突，系统提示“上课时间有冲突，选课失败”；
  + 学生一学期所选课程的学分最多不能超18学分
* 学生可以注销所选课程。
* 学院管理员可以查询学生前几学期的选课信息、可以查询课程基本信息、学生基本信息；
* 当学生退学时，由教务处的管理人注销学生基本信息；
* 如果开课之后，学生要求退课，则由教务处的工作人员为学生注销所选课程；
* 允许学生休学，教务处为休学的退学做学籍冻结处理；复学后为其办理解冻处理；
* 每学期教务处为学生办理学期注册手续；没有办理学期注册的学生不能选课；
* 学期末，学院工作人员负责录入学生的成绩。

**实验步骤**

1. 结合实验内容说明，对现有的学生选课系统进行必要的调研，了解基本的工作流程、软件功能、数据需求和界面风格。
2. 分析实验内容说明和调研结果，画出系统的数据流程图。
3. 编写系统的数据字典。
4. 用IPO图描述系统的处理过程。
5. 画出系统ER图。

### 步骤2：系统设计------选课管理系统设计

1. 对实验1的结果进行概要设计和详细设计，画出功能模块图。
2. 对系统的主界面、课程基本信息录入界面、学生选课操作界面、学生选课结果浏览界面进行设计。

进行数据库设计。得到数据库的逻辑结构图。

### 步骤3：选课管理系统编程

同实验1。

根据实验1和实验2对选课管理系统的了解，选用某一高级编程语言实现本系统，并对编码通过注释进行必要的说明。

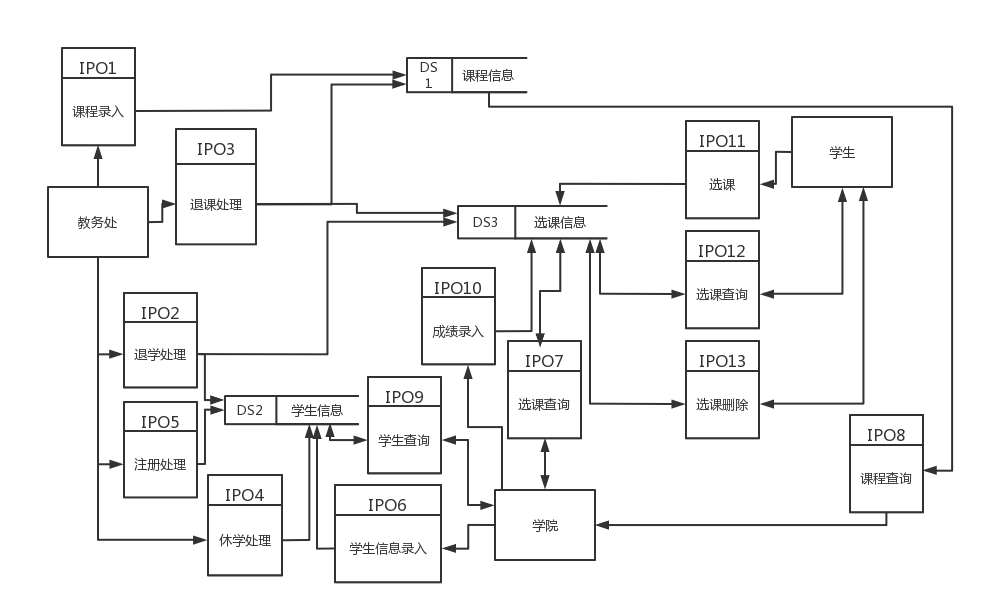
### 步骤4：选课管理系统测试

同实验1。根据实验3编写的选课管理系统，设计测试用例并实现测试。

## 实验结果

### 步骤1：需求分析------大学生选课管理系统

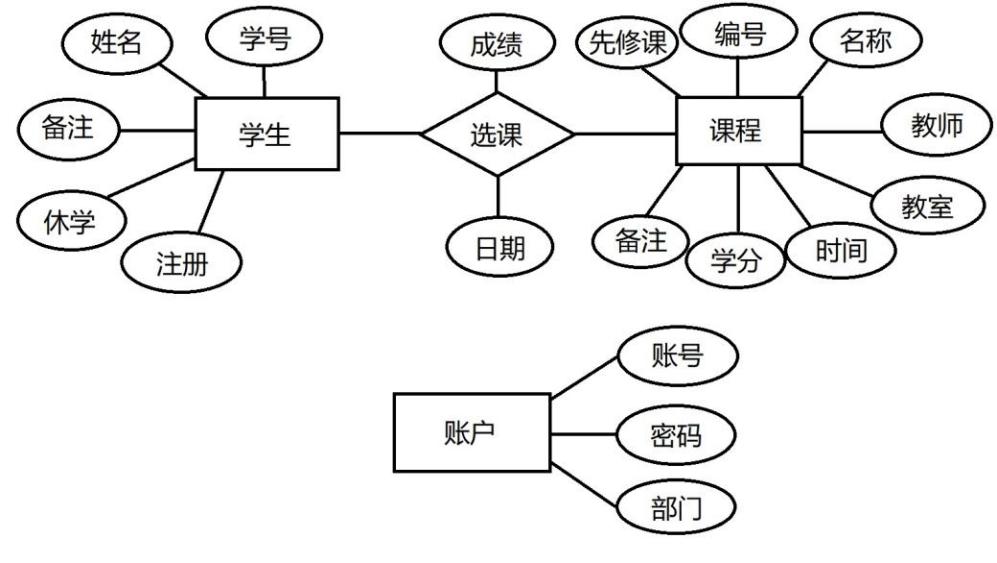
1. 数据流程图



1. 数据字典

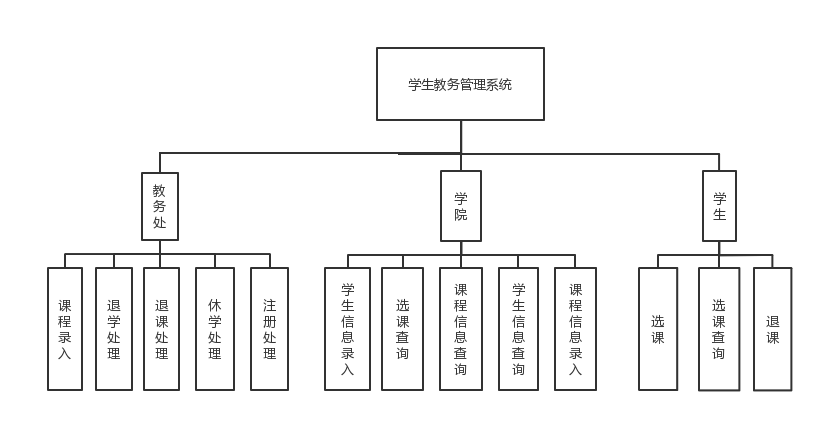
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号：C 名称：课程表 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 名称 | | | | 简称 | | | | 键值 | | | | 类型 | | | | 长度 | | | | 值域 | | | | 初值 | | | | 备注 |
| 课程编号 | | | | ID | | | | P | | | | 整型 | | | |  | | | |  | | | |  | | | |  |
| 课程名称 | | | | NAME | | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  |
| 教室 | | | | CLASSROOM | | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  |
| 教师 | | | | TEACHER | | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  |
| 课程时间 | | | | TIME | | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  |
| 先修课 | | | | PRECOURSE | | | |  | | | | 整型 | | | |  | | | |  | | | |  | | | |  |
| 学分 | | | | CREDIT | | | |  | | | | 整型 | | | |  | | | |  | | | |  | | | |  |
| 备注 | | | | NOTE | | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  |
| 编号：SC 名称：选课表 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 名称 | | | 简称 | | | | 键值 | | | | 类型 | | | | 长度 | | | | 值域 | | | | 初值 | | | | 备注 | |
| 学生编号 | | | SID | | | | P | | | | 整型 | | | |  | | | |  | | | |  | | | |  | |
| 课程编号 | | | CID | | | | P | | | | 整型 | | | |  | | | |  | | | |  | | | |  | |
| 时间 | | | DATE | | | |  | | | | 日期 | | | |  | | | |  | | | |  | | | |  | |
| 分数 | | | SCORE | | | |  | | | | 整型 | | | |  | | | |  | | | |  | | | |  | |
| 编号：ADMINISTRATOR 名称：账户表 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 名称 | 简称 | | | | | 键值 | | | | 类型 | | | | 长度 | | | | 值域 | | | | 初值 | | | | 备注 | | |
| 帐号 | ID | | | | | P | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  | | |
| 密码 | PASSWORD | | | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  | | |
| 部门 | DEPARTMENT | | | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  | | |
| 编号：S 名称：学生表 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 名称 | | 简称 | | | 键值 | | | | 类型 | | | | 长度 | | | | 值域 | | | | 初值 | | | | 备注 | | | |
| 学生编号 | | ID | | | P | | | | 整型 | | | |  | | | |  | | | |  | | | |  | | | |
| 姓名 | | NAME | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  | | | |
| 备注 | | NOTE | | |  | | | | 字符 | | | | 50 | | | |  | | | |  | | | |  | | | |
| 休学 | | SUSPEND | | |  | | | | BIT | | | |  | | | |  | | | |  | | | |  | | | |
| 注册 | | REGISTER | | |  | | | | BIT | | | |  | | | |  | | | |  | | | |  | | | |

1. 系统的ER图。

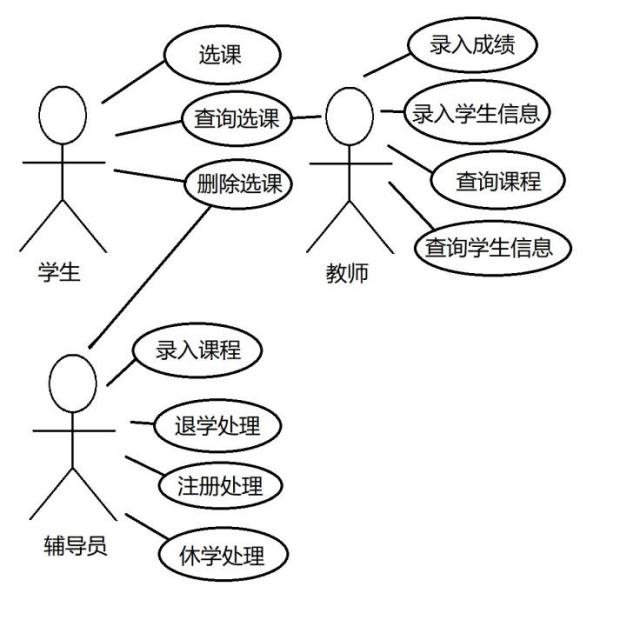


### 步骤2：系统设计------选课管理系统设计

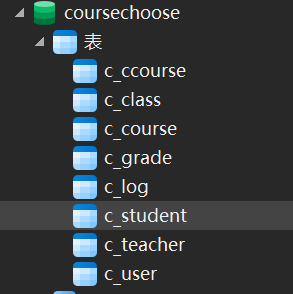
1. 系统功能模块图



1. 用例图

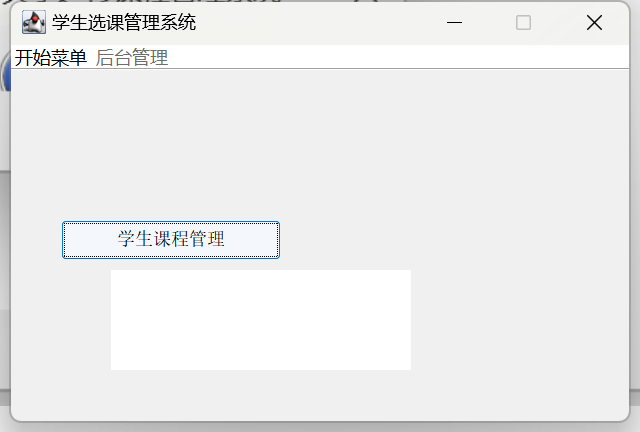


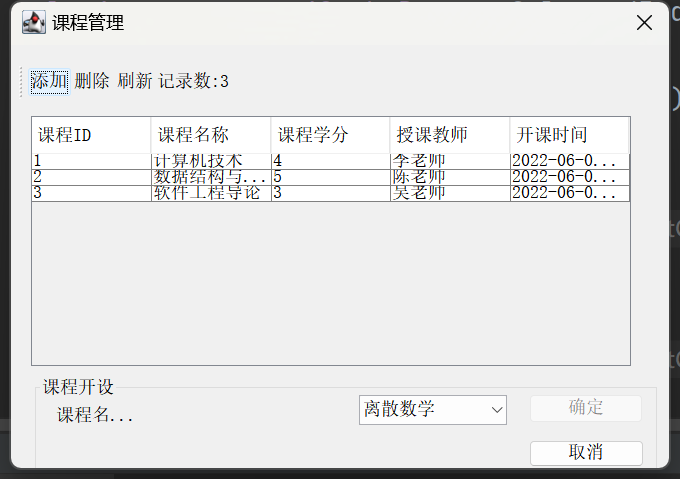
1. 数据库设计（包括表名、字段名、字段类型、字段大小、字段说明）。



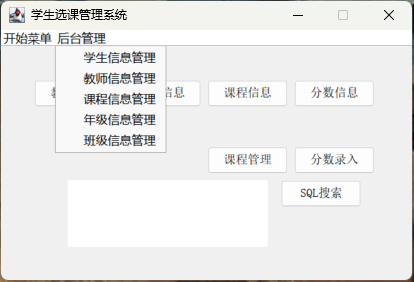
### 步骤3：选课管理系统编程

1. 系统编译无错误，主要功能可以运行通过。（考核方式：验机）
2. 系统主要功能界面。（考核方式：提交文档）
3. 系统主要模块的源代码（包括注释）----（考核方式：提交文档）











**以CCourseDao学生选课类为例**

1. **public** **class** CCourseDao {
2. **private** **int** counts = 0;
4. /\*\*
5. \* @仅仅是添加选课
6. \* @param stu\_id
7. \* @param course\_id
8. \*/
10. **public** **boolean** addCCourse(**int** stu\_id, **int** course\_id, String ccourse\_time) {
12. String sql = "insert into c\_ccourse(stu\_id,course\_id,ccourse\_time) values("
13. + stu\_id + "," + course\_id + ",'" + ccourse\_time + "');";
14. **return** DBConnection.update(sql)==**true**?**true**:**false**;
15. }
17. /\*\*
18. \* @仅仅修改选课
19. \*/
20. **public** **boolean** modifyCCourse(**int** stu\_id, **int** old\_course\_id,**int** new\_course\_id) {
22. String sql = "update  c\_ccourse set course\_id="+new\_course\_id+" where stu\_id="+stu\_id+" and course\_id="+old\_course\_id ;
23. **return** DBConnection.update(sql)==**true**?**true**:**false**;
24. }

27. //获取查询中 的记录集总函数
28. **public** **int** getCounts(){
29. **return** counts;
30. }
32. /\*\*
33. \*
34. \* @根据stu\_id,course\_id 删除选课
35. \* @param stu\_id
36. \* @param course\_id
37. \*/
38. **public** **boolean** deleteCCourse(**int** stu\_id,**int** course\_id){
39. String sql = "delete from c\_ccourse where stu\_id ="+stu\_id+" and course\_id = "+course\_id;
40. **return** DBConnection.update(sql)==**true**?**true**:**false**;
41. }
43. /\*\*
44. \* @添加/修改成绩
45. \* @param stu\_id
46. \* @param course\_id
47. \* @param ccourse\_mark
48. \*/
49. **public** **boolean** modifyCCourseAndMark(**int** stu\_id,**int** course\_id,**int** ccourse\_mark){
50. String sql ="update  c\_ccourse set ccourse\_mark="+ccourse\_mark+" where stu\_id ="+stu\_id+" and course\_id ="+course\_id;
51. **return** DBConnection.update(sql)==**true**?**true**:**false**;
53. }
55. /\*\*
56. \* @是否重复选课
57. \* @param stu\_id
58. \* @param course\_id
59. \* @return true 代表重复
60. \*/
61. **public** **boolean** isCCourseExist(**int** stu\_id, **int** course\_id) {
62. String sql = "select \* from c\_ccourse where stu\_id =" + stu\_id
63. + " and course\_id =" + course\_id;
64. ResultSet rs = DBConnection.query(sql);
65. **try** {
66. **if** (rs.next())
67. **return** **true**;
69. } **catch** (SQLException e) {
70. // TODO Auto-generated catch block
71. e.printStackTrace();
72. }
73. **return** **false**;
75. }
76. /\*\*
77. \* @根据学号获取选课管理模块数据
78. \* @param stu\_id
79. \* @param isNull true 表示 查询成绩为空的列表
80. \* @return 多行记录
81. \*/
82. **public** List<CCourseModel> getListsByStuId(**int** stu\_id,**boolean** isNull){
84. List<CCourseModel> lists = **new** ArrayList<CCourseModel>();
85. String sql ="";
87. **if**(isNull){
88. sql = "select stu\_id,c\_course.course\_id as course\_id,course\_name,ccourse\_mark from c\_course,c\_ccourse where c\_course.course\_id=c\_ccourse.course\_id and  stu\_id ="+stu\_id +" and ccourse\_mark is null";
89. }**else**{
90. sql = "select stu\_id,c\_course.course\_id as course\_id,course\_name,ccourse\_mark from c\_course,c\_ccourse where c\_course.course\_id=c\_ccourse.course\_id and  stu\_id="+stu\_id;
92. }
93. ResultSet rs = DBConnection.query(sql);
94. **try** {
95. **while**(rs.next()){
96. CCourseModel ccm = **new** CCourseModel();
97. ccm.setStu\_id(stu\_id);
98. ccm.setCourse\_id(rs.getInt("course\_id"));
99. ccm.setCourse\_name(rs.getString("course\_name"));
100. ccm.setCcourse\_mark(rs.getInt("ccourse\_mark"));
102. lists.add(ccm);
103. }
104. rs.close();
105. } **catch** (SQLException e) {
106. // TODO Auto-generated catch block
107. e.printStackTrace();
108. }
109. **return** lists;
110. }


114. /\*\*
115. \* @根据id查找对应的名称
116. \* @param course\_id
117. \* @return 课程名称
118. \*/
120. **public** String getNameByCourseId(**int** course\_id) {
121. String course\_name = "";
123. String sql = "select course\_name from c\_course where course\_id="
124. + course\_id;
125. ResultSet rs = DBConnection.query(sql);
126. **try** {
127. **if** (rs.next()) {
128. course\_name = rs.getString("course\_name");
129. }
130. rs.close();
131. } **catch** (SQLException e) {
132. // TODO Auto-generated catch block
133. e.printStackTrace();
134. }
135. **return** course\_name;
137. }

140. **public** List<CCourseModel> getListsByStuId(**int** stu\_id){
141. String sql = "select \* from c\_ccourse where stu\_id="+stu\_id;
142. CourseDao cd = **new** CourseDao();
144. List<CCourseModel> lists = **new** ArrayList<CCourseModel>();

147. **try** {
148. ResultSet rs= DBConnection.query(sql);
149. **while**(rs.next()){
150. CCourseModel ccm=**new** CCourseModel();
151. ccm.setCourse\_id(rs.getInt("course\_id"));
152. ccm.setCourse\_name(cd.getCourseNameByCourseId(rs.getInt("course\_id")));
153. lists.add(ccm);
154. }
155. rs.close();
156. } **catch** (SQLException e) {
157. // TODO Auto-generated catch block
158. e.printStackTrace();
159. }
160. **return** lists;
162. }


166. /\*\*
167. \* @某门课程中，最高最低平均分
168. \* @param course\_id
169. \* @return
170. \*/
171. **public** List<String> getThreeMark(**int** course\_id) {
172. List<String> lists = **new** ArrayList<String>();
174. String sql = "select max(ccourse\_mark) as max\_mark,min(ccourse\_mark) as min\_mark,avg(ccourse\_mark) as avg\_mark from c\_ccourse where course\_id="
175. + course\_id;
176. ResultSet rs = DBConnection.query(sql);
177. **try** {
178. **if** (rs.next()) {
179. lists.add(rs.getString("max\_mark"));
180. lists.add(rs.getString("min\_mark"));
181. lists.add(rs.getString("avg\_mark"));
182. }rs.close();
183. } **catch** (SQLException e) {
184. // TODO Auto-generated catch block
185. e.printStackTrace();
186. }
187. **return** lists;
189. }
191. /\*\*
192. \* @ 某门课程中，优秀率，及格率 以及 男生，女生各多少名
193. \* @param course\_id
194. \* @return 单行数据，List<String> 封装
195. \*/
197. **public** List<String> getGoodRadio(**int** course\_id) {
198. List<String> lists = **new** ArrayList<String>();
199. // 总人数
200. String sql = "select count(\*) as counts from c\_ccourse where course\_id="+ course\_id;
201. ResultSet rs = DBConnection.query(sql);
202. **try** {
203. **if** (rs.next()) {
204. lists.add(rs.getString("counts"));
205. }**else**
206. lists.add("0");
207. //大于 90
209. sql = "select count(ccourse\_mark) as counts from c\_student,c\_ccourse where c\_student.stu\_id=c\_ccourse.stu\_id and ccourse\_mark>=90 and course\_id="+course\_id;
210. rs = DBConnection.query(sql);
212. **if** (rs.next()) {
213. lists.add(rs.getString("counts"));
214. }**else** lists.add("0");
216. // 大于60分人数
217. sql = "select count(ccourse\_mark) as counts from c\_student,c\_ccourse where c\_student.stu\_id=c\_ccourse.stu\_id and ccourse\_mark>=60 and course\_id="
218. + course\_id;
219. rs = DBConnection.query(sql);
220. **if** (rs.next()) {
221. lists.add(rs.getString("counts"));
222. }**else** lists.add("0");
223. // 此门课程男生人数
224. sql = "select count(stu\_sex) as counts from c\_student,c\_ccourse where c\_student.stu\_id=c\_ccourse.stu\_id and stu\_sex='男' and course\_id="+course\_id;
225. rs = DBConnection.query(sql);
227. **if** (rs.next()) {
228. lists.add(rs.getString("counts"));
229. }**else** lists.add("0");
230. rs.close();
231. } **catch** (Exception e) {
232. }
234. **return** lists;
235. }

238. /\*\*
239. \* @根据查询 选课 具体情况(课程id，课程名称，学生id，学生名称，老师id，授课名称，课程成绩)
240. \* @param course\_id 值为-1 按课程id查询
241. \* @param course\_counts\_index 值为-1 根据 course\_id查询
242. \* @return 多行记录，查询模块列表
243. \*/
244. **public** List<CCourseModel> getCCourseSpecificList(**int** course\_id,**int** course\_counts\_index){
245. CourseDao cd = **new** CourseDao();
246. StudentDao sd = **new** StudentDao();
247. TeacherDao td = **new** TeacherDao();
248. CCourseDao ccd = **new** CCourseDao();
249. List<CCourseModel> lists = **new** ArrayList<CCourseModel>();
250. String sql = "";
251. **if**(course\_counts\_index==-1){
252. sql = "select \* from c\_ccourse where course\_id="+course\_id;
254. **try** {
255. ResultSet rs = DBConnection.query(sql);
256. **while**(rs.next()){
257. counts++;
258. CCourseModel ccm = **new** CCourseModel();
259. **int** course\_id\_i = rs.getInt("course\_id");
260. **int** stu\_id\_i = rs.getInt("stu\_id");
261. ccm.setCcourse\_mark(rs.getInt("ccourse\_mark"));
262. **int** teach\_id\_i = cd.getTeachIdByCourseId(course\_id\_i);//老师id
264. ccm.setCourse\_id(course\_id\_i);
265. ccm.setCourse\_name(cd.getCourseNameByCourseId(course\_id\_i));
266. ccm.setStu\_id(stu\_id\_i);
267. ccm.setStu\_name(sd.getStudentNameById(stu\_id\_i));
268. ccm.setTeach\_id(teach\_id\_i);
269. ccm.setTeach\_name(td.getTeacherNameById(teach\_id\_i));
270. lists.add(ccm);
271. }rs.close();
272. } **catch** (SQLException e) {
273. // TODO Auto-generated catch block
274. e.printStackTrace();
275. }

278. }**else** **if**(course\_id==-1){
280. **switch**(course\_counts\_index){
281. **case** 0:
282. sql = "select stu\_id  from c\_student where stu\_id not in  (select stu\_id from c\_ccourse)";
283. **try** {
284. ResultSet rs = DBConnection.query(sql);
285. **while**(rs.next()){
286. CCourseModel ccm = **new** CCourseModel();
287. ccm.setStu\_id(rs.getInt("stu\_id"));
288. ccm.setStu\_name("-1");
289. ccm.setCourse\_id(-1);
290. ccm.setCourse\_name("-1");
291. ccm.setTeach\_id(-1);
292. ccm.setTeach\_name("-1");
293. ccm.setCcourse\_mark(-1);
294. lists.add(ccm);
295. }rs.close();
296. } **catch** (SQLException e) {
297. // TODO Auto-generated catch block
298. e.printStackTrace();
299. }

302. **break**;
303. **case** 1:
305. sql = "select temp.stu\_id,course\_id from ((select stu\_id,count(\*) as counts  from c\_ccourse group by stu\_id having counts=1) as temp),c\_ccourse where c\_ccourse.stu\_id = temp.stu\_id";
307. **try** {
308. ResultSet rs1 = DBConnection.query(sql);
309. **while**(rs1.next()){
310. CCourseModel ccm = **new** CCourseModel();
312. **int** course\_id\_i = rs1.getInt("course\_id");
313. **int** stu\_id\_i = rs1.getInt("stu\_id");
315. ccm.setCourse\_id(course\_id\_i);
316. ccm.setCourse\_name(cd.getCourseNameByCourseId(course\_id\_i));
317. ccm.setStu\_id(stu\_id\_i);
318. ccm.setStu\_name(sd.getStudentNameById(stu\_id\_i));
319. ccm.setCcourse\_mark(ccd.getCCourseMarkByStuIdCourseId(stu\_id\_i, course\_id\_i));
320. //根据课程id 查找对于教师id和姓名
321. ccm.setTeach\_id(cd.getTeachIdByCourseId(course\_id\_i));
322. ccm.setTeach\_name(td.getTeacherNameById(cd.getTeachIdByCourseId(course\_id\_i)));
324. lists.add(ccm);
325. }rs1.close();
326. } **catch** (SQLException e) {
327. // TODO Auto-generated catch block
328. e.printStackTrace();
329. }

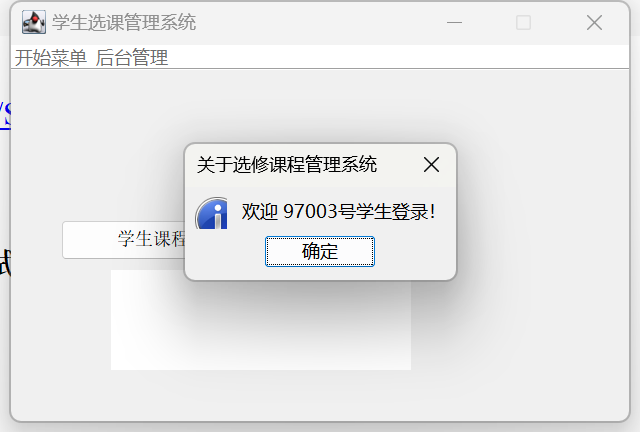
332. **break**;
333. **case** 2:
334. sql = "select temp.stu\_id,course\_id from ((select stu\_id,count(\*) as counts  from c\_ccourse group by stu\_id having counts>=2) as temp),c\_ccourse where c\_ccourse.stu\_id = temp.stu\_id";
336. **try** {
337. ResultSet rs2 = DBConnection.query(sql);
338. **while**(rs2.next()){
339. CCourseModel ccm = **new** CCourseModel();
341. **int** course\_id\_i = rs2.getInt("course\_id");
342. **int** stu\_id\_i = rs2.getInt("stu\_id");
343. ccm.setCourse\_id(course\_id\_i);
344. ccm.setCourse\_name(cd.getCourseNameByCourseId(course\_id\_i));
345. ccm.setStu\_id(stu\_id\_i);
346. ccm.setStu\_name(sd.getStudentNameById(stu\_id\_i));
347. ccm.setCcourse\_mark(ccd.getCCourseMarkByStuIdCourseId(stu\_id\_i, course\_id\_i));
348. //根据课程id 查找对于教师id和姓名
349. ccm.setTeach\_id(cd.getTeachIdByCourseId(course\_id\_i));
350. ccm.setTeach\_name(td.getTeacherNameById(cd.getTeachIdByCourseId(course\_id\_i)));
351. lists.add(ccm);
352. }rs2.close();
353. } **catch** (SQLException e) {
354. // TODO Auto-generated catch block
355. e.printStackTrace();
356. }
358. **break**;
360. **default**:
362. }
364. }
365. **return** lists;
366. }

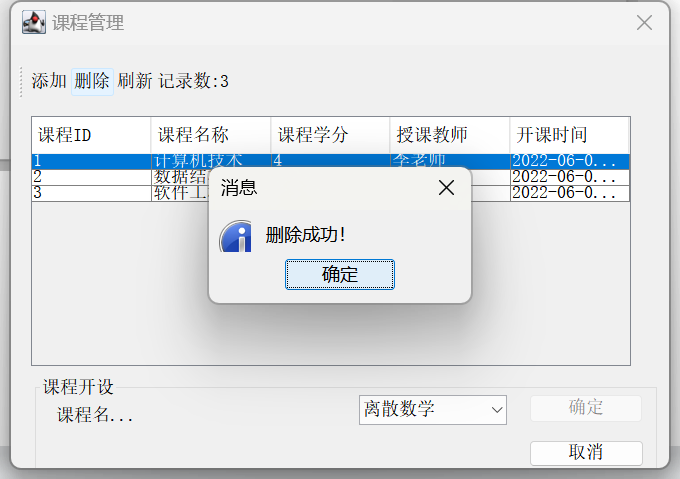
369. /\*\*
370. \* @根据条件，全表模糊查询
371. \* @param type 类型
372. \* @param query\_word 关键字
373. \* @param fuzzy\_query 是否模糊查询
374. \* @return 多行数据，选课模块列表
375. \*/
376. **public** List<CCourseModel> queryCCourseSpecificList(**int** type,String query\_word,**boolean** fuzzy\_query){
377. CourseDao cd = **new** CourseDao();
378. StudentDao sd = **new** StudentDao();
379. TeacherDao td = **new** TeacherDao();
380. List<CCourseModel> lists = **new** ArrayList<CCourseModel>();
381. String sql = "";
382. //如果模糊查询，四表连接查询！慎用。
383. **if**(fuzzy\_query){
384. sql ="select c\_course.course\_id,course\_name,c\_ccourse.stu\_id as stu\_id,stu\_name,c\_course.teach\_id as teach\_id,teach\_name,ccourse\_mark " +
385. "from c\_student,c\_course,c\_ccourse,c\_teacher " +
386. "where   c\_student.stu\_id = c\_ccourse.stu\_id and c\_ccourse.course\_id=c\_course.course\_id and c\_course.teach\_id = c\_teacher.teach\_id " +
387. "and (c\_course.course\_id like '%"+query\_word+"%' or course\_name like '%"+query\_word+"%' or c\_ccourse.stu\_id like '%"+query\_word+"%' or stu\_name like '%"+query\_word+"%'  or c\_course.teach\_id like '%"+query\_word+"%' or teach\_name like '%"+query\_word+"%' or ccourse\_mark like '%"+query\_word+"%')";
388. }**else**{
390. **switch**(type){
391. **case** 0:
392. //按编号
393. sql ="select c\_course.course\_id,course\_name,c\_ccourse.stu\_id as stu\_id,stu\_name,c\_course.teach\_id as teach\_id,teach\_name,ccourse\_mark " +
394. "from c\_student,c\_course,c\_ccourse,c\_teacher " +
395. "where   c\_student.stu\_id = c\_ccourse.stu\_id and c\_ccourse.course\_id=c\_course.course\_id and c\_course.teach\_id = c\_teacher.teach\_id " +
396. "and (c\_course.course\_id like '%"+query\_word+"%' or c\_ccourse.stu\_id like '%"+query\_word+"%')";
398. **break**;
399. **case** 1:
400. sql ="select c\_course.course\_id,course\_name,c\_ccourse.stu\_id as stu\_id,stu\_name,c\_course.teach\_id as teach\_id,teach\_name,ccourse\_mark " +
401. "from c\_student,c\_course,c\_ccourse,c\_teacher " +
402. "where   c\_student.stu\_id = c\_ccourse.stu\_id and c\_ccourse.course\_id=c\_course.course\_id and c\_course.teach\_id = c\_teacher.teach\_id " +
403. "and (course\_name like '%"+query\_word+"%' or stu\_name like '%"+query\_word+"%' or teach\_name like '%"+query\_word+"%')";
404. **break**;
405. **case** 2:
406. sql ="select c\_course.course\_id,course\_name,c\_ccourse.stu\_id as stu\_id,stu\_name,c\_course.teach\_id as teach\_id,teach\_name,ccourse\_mark " +
407. "from c\_student,c\_course,c\_ccourse,c\_teacher " +
408. "where   c\_student.stu\_id = c\_ccourse.stu\_id and c\_ccourse.course\_id=c\_course.course\_id and c\_course.teach\_id = c\_teacher.teach\_id " +
409. "and ccourse\_mark like "+query\_word+"";
410. **break**;
411. **default**:
412. **break**;
414. }
416. }
417. **try** {
418. ResultSet rs = DBConnection.query(sql);
419. **while**(rs.next()){
420. CCourseModel ccm = **new** CCourseModel();
421. counts++;
422. **int** course\_id\_i = rs.getInt("course\_id");
423. **int** stu\_id\_i = rs.getInt("stu\_id");
424. ccm.setCcourse\_mark(rs.getInt("ccourse\_mark"));
425. **int** teach\_id\_i = cd.getTeachIdByCourseId(course\_id\_i);
427. ccm.setCourse\_id(course\_id\_i);
428. ccm.setCourse\_name(cd.getCourseNameByCourseId(course\_id\_i));
429. ccm.setStu\_id(stu\_id\_i);
430. ccm.setStu\_name(sd.getStudentNameById(stu\_id\_i));
431. ccm.setTeach\_id(teach\_id\_i);
432. ccm.setTeach\_name(td.getTeacherNameById(teach\_id\_i));
433. lists.add(ccm);
434. }rs.close();
435. } **catch** (SQLException e) {
436. // TODO Auto-generated catch block
437. e.printStackTrace();
438. }
439. **return** lists;
441. }

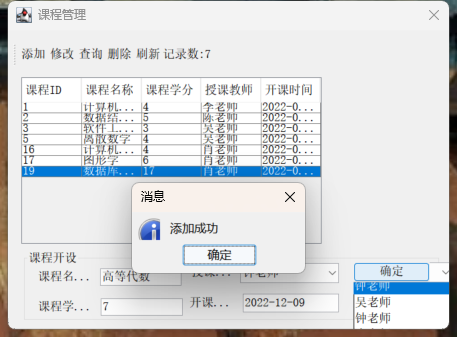
444. /\*\*
445. \* @根据stu\_id和course\_id获取成绩
446. \* @param stu\_id
447. \* @param course\_id
448. \* @return 成绩
449. \*/
451. **public** **int** getCCourseMarkByStuIdCourseId(**int** stu\_id,**int** course\_id){
452. **int** ccourse\_mark = 0;
453. String sql = "select ccourse\_mark from c\_ccourse where stu\_id="+stu\_id+" and course\_id="+course\_id;
454. ResultSet rs = DBConnection.query(sql);
455. **try** {
456. **if**(rs.next()){
458. ccourse\_mark=rs.getInt("ccourse\_mark");
459. }
460. } **catch** (SQLException e) {
461. // TODO Auto-generated catch block
462. e.printStackTrace();
463. }
464. **return** ccourse\_mark;
465. }
466. }

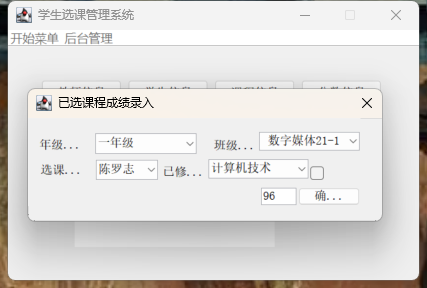
github源码：[PAisOH/SQL\_StudentManager](https://github.com/PAisOH/SQL_StudentManager)

### 步骤4：选课管理系统测试









## 实验总结

1. JDBC（Java DataBase Connectivity）是 Java 语言提供的一种用于访问数据库的接口标准，它定义了一组用于访问数据库的类和接口，可以使 Java 程序通过 JDBC 接口与各种关系型数据库交互。
2. T-SQL（Transact-SQL）是一种用于 Microsoft SQL Server 数据库管理系统的扩展结构化查询语言（SQL）。
3. JDBC 和 T-SQL 是两种不同的技术，但它们可以结合使用。JDBC 可以提供统一的接口来访问数据库，而 T-SQL 可以提供 SQL 语言来操作数据库。因此，通过使用 JDBC 接口来连接 SQL Server 数据库，并通过 T-SQL 语言来执行 SQL 操作，可以让 Java 程序与 SQL Server 数据库进行交互。

例如，下面是一个使用 JDBC 访问 SQL Server 数据库的示例代码，该代码使用 T-SQL 语言来执行一个简单的 SELECT 查询：

