# 学籍管理系统设计报告

**1，需求分析**

**1.1信息要求**

本系统针对学籍管理，主要涉及学生信息，课程信息，教学计划信息，学生课程成绩信息，班级教师信息（一个班的某门课由哪个老师所教），班级专业信息等信息。具体如下：

班级专业信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 含义 |
| Class | char(10) | 是 | 班级 |
| Major | char(10) |  | 专业 |

学生信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 含义 |
| Sno | char（10） | 是 | 学号 |
| Sname | char（20） |  | 姓名 |
| Ssex | char（10） |  | 性别 |
| Birthday | char（10） |  | 生日 |
| Class | char（10） |  | 班级 |

教师信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 含义 |
| Tname | char（10） | 是 | 姓名 |
| Tsex | char（2） |  | 性别 |
| Tage | smallint |  | 年龄 |

课程信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 含义 |
| Cno | char（10） | 是 | 课程号 |
| Cname | char（50） |  | 课程名 |
| Ccredit | float |  | 学分 |

班级教师信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 含义 |
| Class | char（10） | 是 | 课程号 |
| Cno | char（10） | 课程名 |
| Tname | char（10） |  | 学分 |

学生课程成绩信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 含义 |
| Sno | char（10） | 是 | 学号 |
| Cno | char（10） | 课程名 |
| Grade | float |  | 成绩 |
| BK | char（4） |  | 考试性质 |

教学计划信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 含义 |
| Major | char（10） | 是 | 专业 |
| Cno | char（10） | 课程号 |
| Cp | char（10） |  | 课程性质 |
| Semster | Smallint |  | 学期 |

**1.2处理要求：**

本系统主要满足两类用户的要求，分别是管理员和学生。

管理员用户：

①录入学生信息

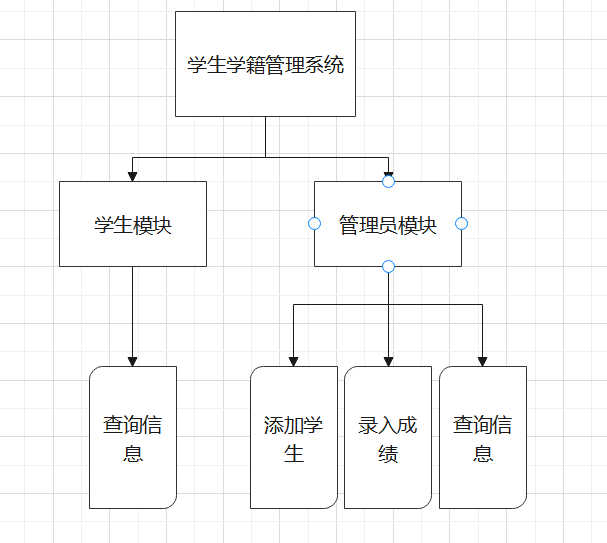
②查询学生信息、某学生授课教师及其学生所选课程信息

③录入和查询学生成绩信息

④查询快要被开除的学生

学生用户：查询信息

功能模块如下图所示：



**1.3安全性和完整性要求。**

1>实体完整性

2>参照完整性

Student(Class)参照了CM(Class)

CT(Class)参照了CM(Class)

CT(Cno)参照了Course(Cno)

SC(Sno)参照了Student(Sno)

SC(cno)参照了Course(Cno)

Schedule(Major)参照了CM(Major)

Schedule(Cno)参照了Course(Cno)

3>用户定义的完整性

学生表的性别只能为男或女

学生表的年龄在0到100之间

课程表的学分只能大于0

选课表的成绩只能在0到100之间

选课表的考试性质只能在初考和补考之间选择

教学计划的课程性质只能在必修和选修之间选择

教学计划的学期只能在0到8之间选择

**2，概念结构设计**

经分析知，学籍管理系统包含学生，教师，课程，班级四个实体

学生具有属性：学号，姓名，性别，生日，班级

其中学号是主码，班级是外码

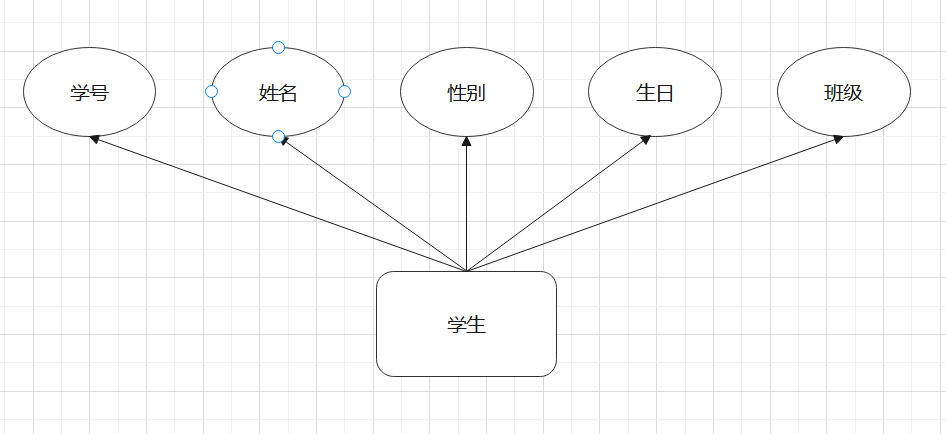
教师具有属性：姓名，性别，年龄

课程具有属性：课程号，课程名，学分

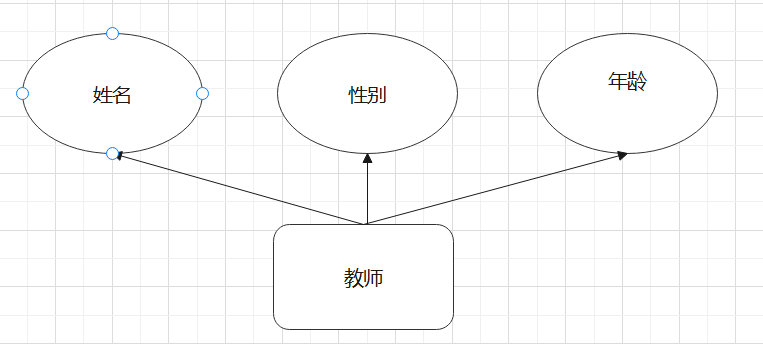
班级具有属性：班级，专业

各个实体的局部ER图如下：

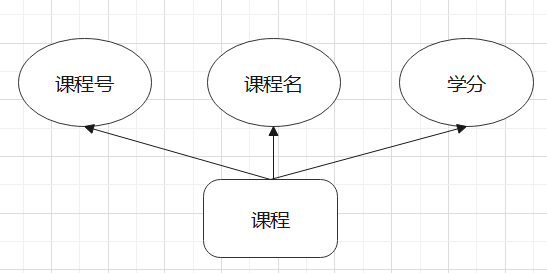
①学生：



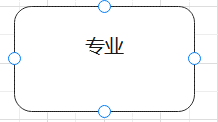
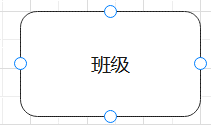
②教师：



③课程：



④班级： ⑤专业：



分析需求：

学校有若干专业，每个专业每年招若干个班，每个班有若干学生

每个专业有自己的教学计划，规定了该专业相关课程的性质（必修或选修）以及授课学期；例如，数据库课程对计算机专业为必修、在大三上学期，但对数学专业可能为选修、在大三下学期，而中文专业可能不学这门课

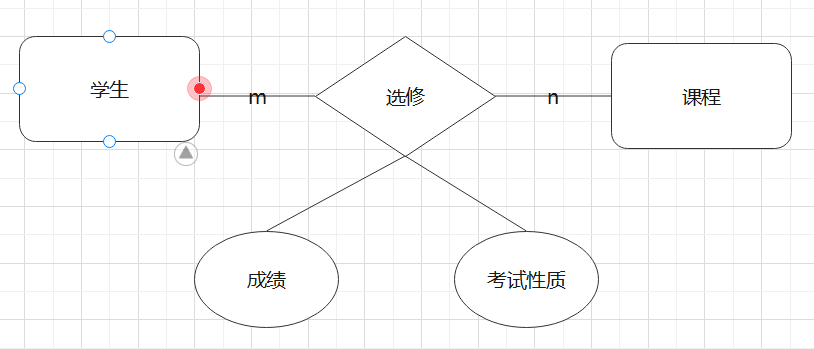
一位教师可以给多个班带课，但不能给一个班带多门课

一门课程最多允许学生一次补考；学生达到如下条件之一的被开除：不及格必修课累计达10学分、或不及格选修课累计达15学分

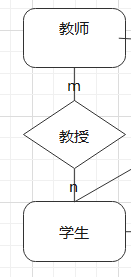
假设一个班级的某一门课只由一个老师代课，老师的姓名不重复。

根据上述信息可以推导出以下ER图：

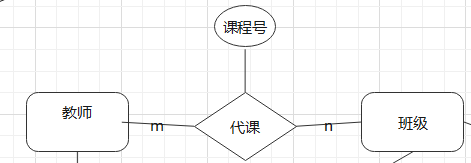
①学生和课程之间的ER图：



②学生和教师之间的ER图



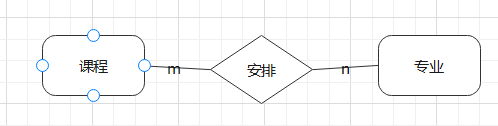
③教师和班级之间的ER图



④班级和专业之间的ER图



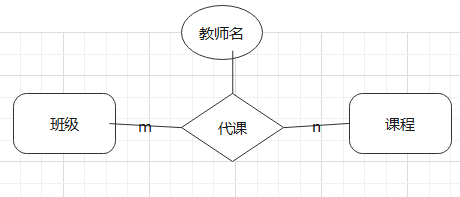
⑤专业和课程之间的ER图



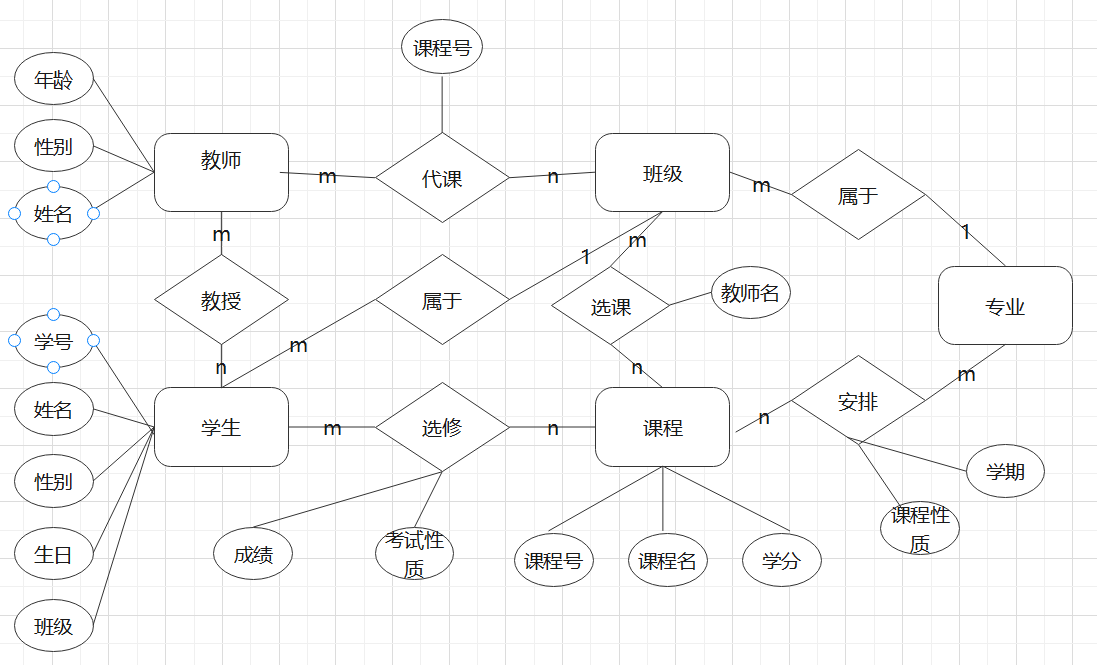
⑥学生和班级之间的ER图



⑦班级和教师之间的ER图



总的ER图为：



**3，逻辑结构设计**

**3.1根据ER图向关系数据模型的转换规则，将上述ER图转换为关系模式如下：**

班级专业（班级，专业），码为班级

学生（学号，姓名，性别，生日，班级），码为学号

课程（课程号，课程名，学分），码为课程号

教师（姓名，性别，年龄），码为姓名

选课（学号，课程号，成绩，考试性质），码为(学号，课程号)

班级课程教师（班级，课程，教师），码为(班级，教师)

教学计划（专业，课程号，课程性质，学期），码为(专业，课程号)

**3.2由分析知，每个模式的数据依赖分别为：**

班级专业（班级->专业）

学生（学号->姓名，学号->性别，学号->生日，学号->班级）

课程（课程号->课程名，课程号->学分）

教师（姓名->性别，姓名->年龄）

选课（(学号，课程号)->成绩，(学号，课程号)->考试性质）

班级课程教师（(班级，课程)->教师，(教师，班级)->课程）

教学计划（（专业，课程号）->课程性质，(专业，课程号)->学期）

**3.3分析上述各个模式的码及其依赖可知：**

班级关系模式属于4NF，因为其是二目关系。

学生关系模式最低属于BCNF，因为其每一个依赖都包含码。

课程关系模式最低属于BCNF，因为其每一个依赖都包含码。

教师关系模式最低属于BCNF，因为其每一个依赖都包含码。

选课关系模式最低属于BCNF，因为其每一个依赖都包含码。

班级课程教师关系模式属于3NF，不属于4NF因为不存在非主属性对码的传递依赖和部分依赖，但(教师，班级)->课程依赖不包含码，所以不是BCNF。

教学计划关系模式最低属于BCNF，因为其每一个依赖都包含码。

**3.4用户子模式设计。**

①管理员查看快要挂科的学生，建立如下两个视图：

不及格必修选课记录表(学号，课程号，学分)，表示选课表中不及格必修课程的记录以及学分。

不及格选修选课记录表(学号，课程号，学分), 表示选课表中不及格选修课程的记录以及学分。

②管理员查看学生所选课程详情，建立如下视图：

学生所选课程详情表(学号，课程名，课程性质，学期，学分，成绩)，记录某学生所选课程详情。

**4，功能及实现**

选用的DBMS为sqlserver，根据以上的思路建立好符合完整性要求的模式，使用javax.swing和java.awt辅助设计前台应用。

**功能一：**录入一位学生，应包含学号、姓名、性别、出生年月、班级等信息

相应的数据库语句：

insert into student values(?,?,?,?,?)，其中？从左到右依次设为Sno,Sname,Ssex,Birthday,Class。

成功插入界面：

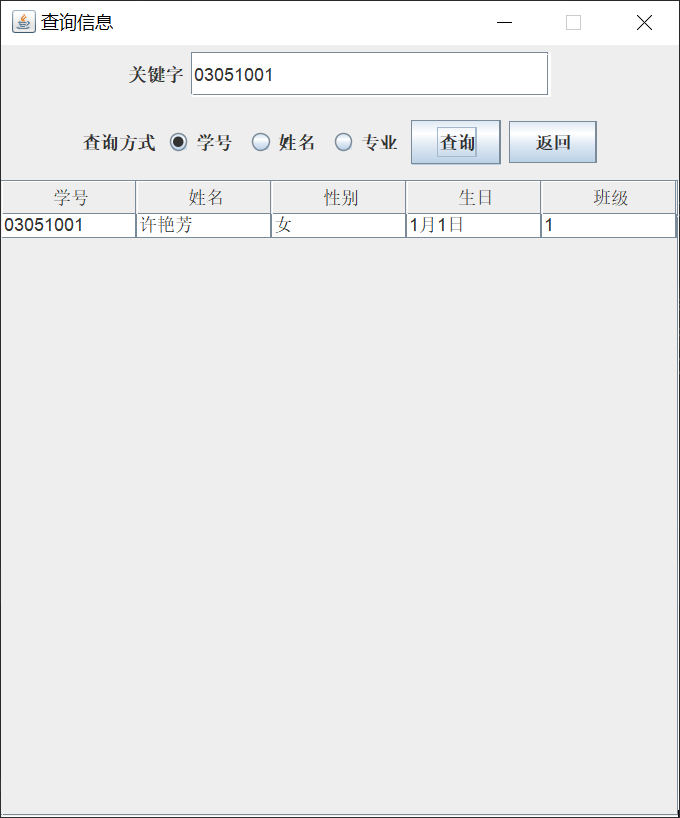


学号已存在时会插入失败

**功能二：**按学号、姓名、专业三种方式查询学生基本信息

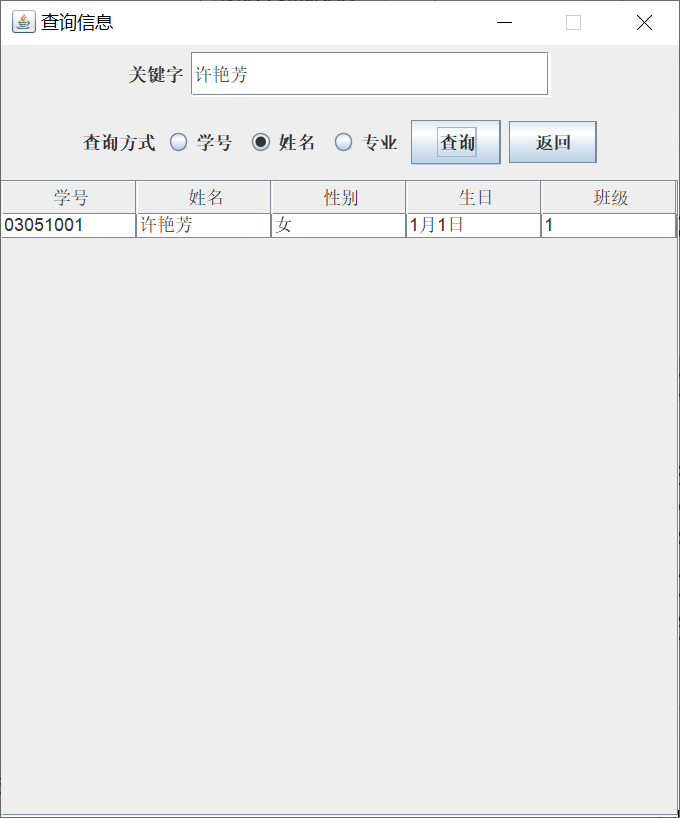
①学号：

相应的数据库查询语句：select \* from Student where Sno = ?



②姓名：

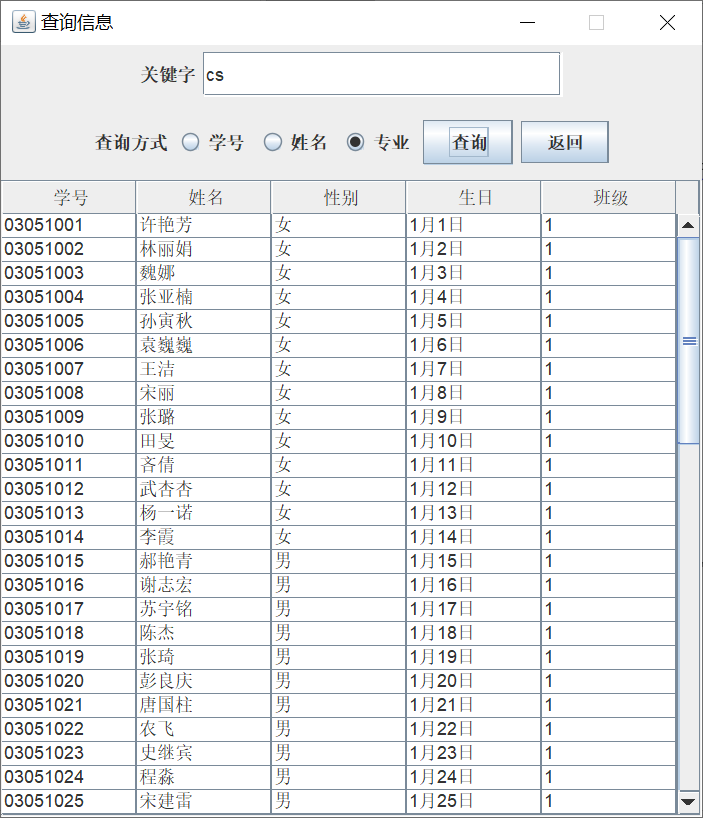
相应的数据库查询语句：select \* from Student where Sname = ?



③专业：

相应的数据库查询语句：select \* from Student where Class in " +

"(select Class from CM where major = ?)



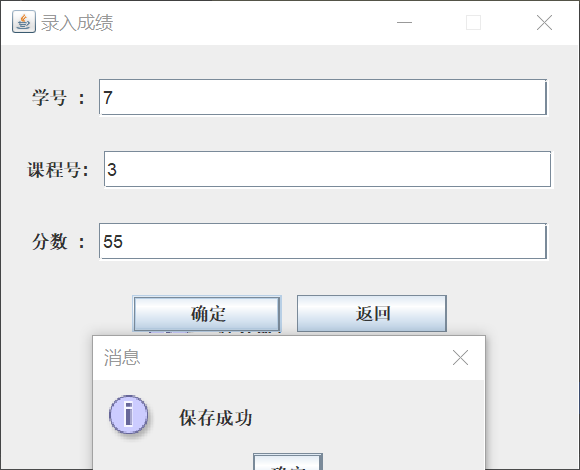
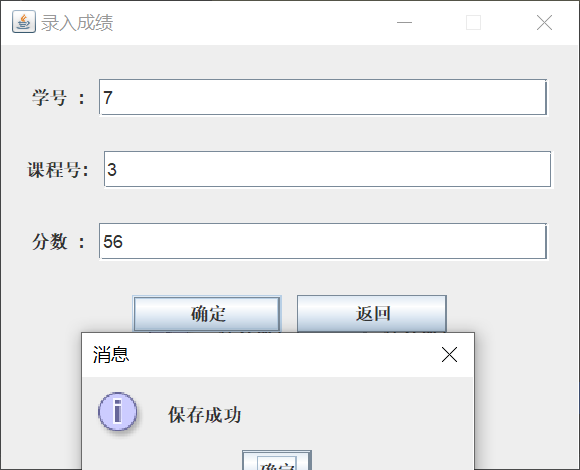
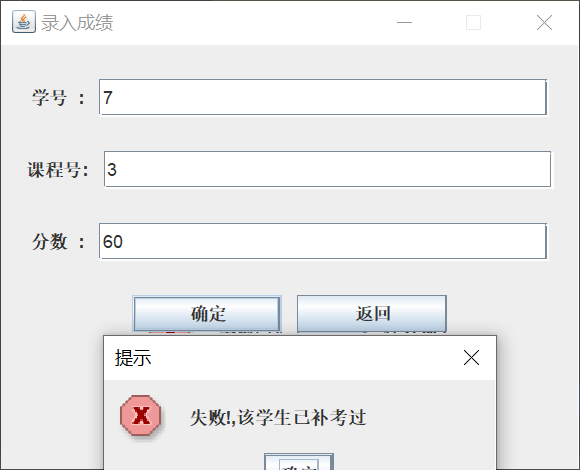
**功能三：**录入一位学生一门课的成绩

结合需求每位同学只允许补考一次，这里我先在选课表中查找该学号和课程号，语句为：select \* from SC where Sno = ? and Cno = ?再分下面三种情况：

①无该条记录或考试性质为null，则该学生为初考，插入记录并将考试性质设为初考，sql语句为: insert into SC(Sno,Cno,Grade) values (?,?,?,?)

②有该条记录，考试性质为初考，则更新该条记录的成绩，考试性质置为补考，sql语句：update SC set Grade = ?,BK = ? where Sno=? and Cno=?

③有该条记录，考试性质为补考，则拒绝执行。

**功能四：**查询一位学生所修的课程、性质（必修或选修）、学期、学分及成绩；查询他的必修课平均成绩、所有课程平均成绩（平均成绩应按学分加权）

①查询一位学生所修的课程、性质（必修或选修）、学期、学分及成绩。

根据3.4的分析，建立视图的sql语句为：

CREATE VIEW SC\_INFO as

select Sno,Cname,Cp,Semster,Ccredit,Grade from Course,SC,Schedule

where Course.Cno=SC.Cno and

Schedule.Cno = SC.Cno and

Schedule.Major in (select Major from Student,CM

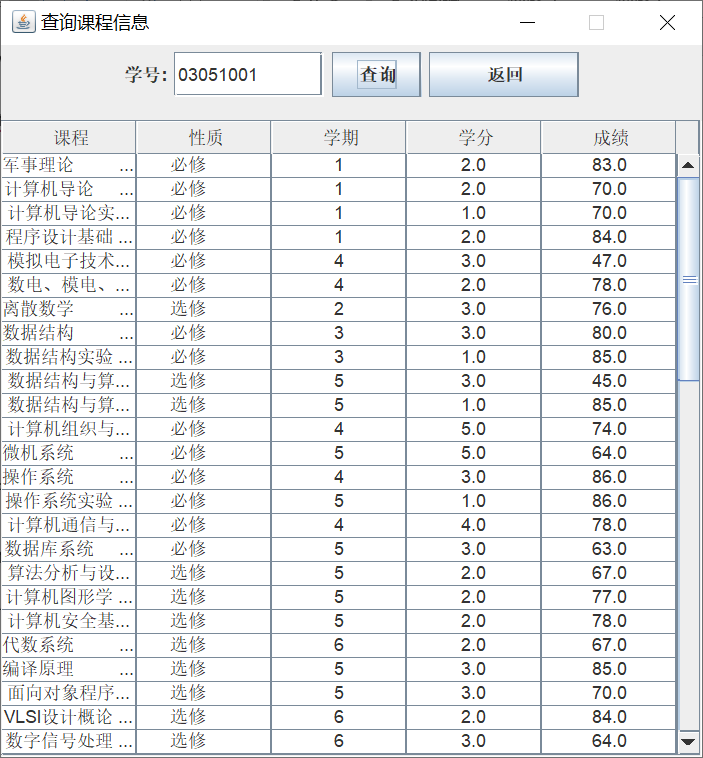
where Student.Class=CM.CLass and

Sno = SC.Sno)

视图建立好后，再在该视图中查询，sql查询语句为：

select \* from SC\_INFO where Sno = ？

查询结果如下图：



②查询他的必修课平均成绩、所有课程平均成绩

可以利用上一步生成的视图SC\_INFO，查询必修平均成绩的sql语句如下：

select sum(Grade\*Ccredit)/sum(Ccredit) from SC\_INFO

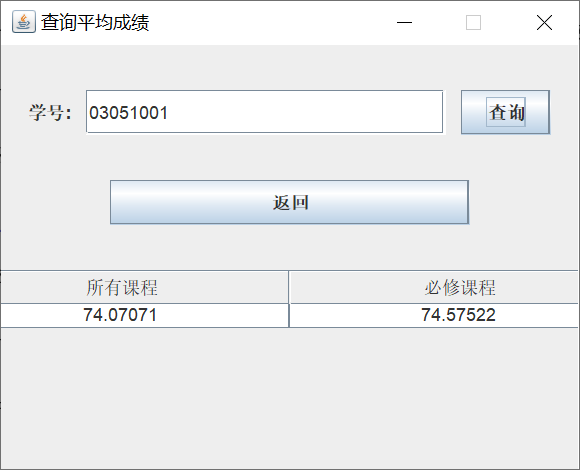
where Cp='必修' and

Sno = ?

查询所有课平均成绩的sql语句如下：

select sum(Grade\*Ccredit)/sum(Ccredit) from SC\_INFO

where Sno = ?



**功能五：**查询一位学生被哪些教师教过课

考虑利用学号查询该学生的班级和所选课程，再用Class和Cno在CT表中查出教师姓名，Cno在课程表中查询出Cname，用Tname在教师表中查出教师详情，相应的sql语句为：

select CT.Tname,Cname,Tsex,Tage from Teacher,CT,Course where CT.Tname=Teacher.Tname and

CT.Cno=Course.Cno and

CT.Cno in (select Cno from SC where Sno = ?) and

Class = (select Class from Student where Sno = ?)



**功能六：**查询快要被开除的学生（距被开除差3学分之内）

根据3.4分析，创建不及格必修选课记录视图(学号，课程号，学分)的sql语句：

CREATE VIEW LOW\_COMPUL\_SC(Sno,Cno,Ccredit) as

select Sno,SC.Cno,Ccredit from SC,Course where Grade<60 and

SC.Cno=Course.Cno and

SC.Cno in (select Cno from Schedule where Major = (select Major from CM,Student where Sno = SC.Sno and Cm.CLass = Student.Class) and

Cp = '必修')

创建不及格选修选课记录视图(学号，课程号，学分)的sql语句：

CREATE VIEW LOW\_NCOMPUL\_SC(Sno,Cno,Ccredit) as

select Sno,SC.Cno,Ccredit from Course,SC where Grade<60 and

SC.Cno=Course.Cno and

SC.Cno in (select Cno from Schedule where Major = (select Major from CM,Student where Sno = '3' and Cm.CLass = Student.Class) and

Cp = '选修')

接着，只需将LOW\_COMPUL\_SC按学号成组，选择出学分之和在7到10之间的学号，利用学号查询出个人信息

将LOW\_NCOMPUL\_SC按学号成组，选择出学分之和在12到15之间的学号，利用学号查询出个人信息

将以上两者联合即为快被开除的学生信息。相应的sql语句如下：

select \* from Student where Sno in(select Sno from LOW\_COMPUL\_SC

GROUP BY Sno having SUM(Ccredit)>=7 and SUM(Ccredit)<10)

union select \* from Student where Sno in(select Sno from LOW\_NCOMPUL\_SC

GROUP BY Sno having SUM(Ccredit)>=12 and SUM(Ccredit)<15)



**5，遇到的问题及解决办法**

**①**刚开始时我将Tname一项也加在SC表中，后来发现这样数据冗余太大了，而且也不方便后期扩展，考虑到班级，教师和课程之间的关系紧密，所以我最终将他们三个组成了一张表CT。

**②**处理专业这个属性时发现无论放到哪张表里都会有冗余，所以最终我将班级和专业组成一张表，班级为主码，这样符合实际情况。

**③**在处理需求**一门课程最多允许学生一次补考**时，我设计了考试性质这一属性，表示该门选课是初考还是补考，并且将其作为每次录入成绩时都要填写的，后来考虑到客观情况，第一次录入就是初考，第二次录入就是补考，于是修改了相关逻辑。

**④**在查询快被开除的学生信息时，发现过程过于复杂，于是考虑建立一个不及格选课记录视图，将学号，课程号和学分都放到一个视图里，然后使用集函数处理就简单了许多。

**⑤**查询一位被哪些教师教过课时，刚开始我只返回了教师姓名，别的信息都没有，后来发现这样过于简洁和单调，于是我修改了相关语句，加入了所教课程，性别，年龄等信息，这样才满足实际使用需求。

**⑥**在查询一位学生的必修和选修课加权平均成绩时，一开始我直接将多个表连接查询，比较复杂，效率也可能不高，后来发现可以直接使用上一步建立的视图SC\_INFO，这样查询语句就简单了很多。

**6，总结**

通过这次数据库的设计，我对sql语言有了更加深刻的了解，对数据库系统的流程有了深刻的体会。一个完善的数据库系统不是一蹴而就的，要经过谨慎的需求分析、概念结构设计、逻辑结构设计等步骤，容不得一丝疏忽。尤其是需求分析阶段，必须全面、准确了解用户的实际要求，并进行分析，如果需求分析没做好，那么将对后面的步骤产生很多负面影响，甚至无法进行下去。

这次实验也让我懂得了模块化设计的重要性，可以化繁为简，将前台应用和数据库分离，这样在发现错误时可以快速的定位到错误的地方，而且方便后来的扩展。在处理某个复杂的需求时，可以先创建合适的视图，再对视图进行操作，这样将复杂的过程分解成两步相对简单的操作，思路清晰且不易出错。

数据库应用系统设计完成成功运行之后并不代表任务就结束了，在数据库系统运行过程中必须不断地对其进行评价、调整与修改。