

数据库工程作业

要求:

- 1. 完成一个小型的数据库信息管理系统（或部分功能），并填写工程作业报告；程序和报告请在规定时间之内上传。
- 2. 开发模式（B/S 或 C/S）、开发高级语言任选，后台数据库使用大型数据库管理系统（SQL Server、Oracle、MySQL 等），不要使用桌面数据库。
- 3. 报告中所列举的四种操作，每种操作举一个例子即可。
- 4. 作业成绩按照报告中的标准评分，程序只实现报告中涉及的部分即可。
- 5. 作业完成后，请将工程作业报告和程序打包提交给助教老师，并联系助教老师进行系统说明和演示，回答相关问题。

工程作业报告

1. 项目信息（10 分）

学号	2111460	姓名	张洋	专业	信息安全
项目名称	学生信息管理系统				
必备环境	Windows 11 SQL server 2022 JRE/JDK 环境, 并加载 JDBC 驱动 Microsoft SQL Server Management Studio 19 Eclipse Java Oxygen				
系统主要功能简介（4 分）	本系统设计为一个简易的学生信息管理系统，可以针对学生基本信息、选课信息、专业信息等相关事项进行管理。 本次工程作业只对其中的部分功能予以实现，实现的功能包括： （1）管理员登录界面验证：需要在图形化界面输入相应的正确 SQL 登录名和登录密码方可进入管理系统，否则弹窗提示用户名或者密码错误 （2）删除操作：课程信息删除 （3）插入操作：新生信息录入 （4）更新操作：转专业信息更新 （5）查询操作：选课情况信息查询				
系统主要页面	截 3-4 个页面即可				

截图
(6
分)





2. 系统配置 (10 分)

说明	(2 分) 请说明系统配置情况 (后台数据库, 高级语言); (8 分) 请使用连接串连接高级语言和数据库, 并分析字符串的各个部分。	
配置步骤	DBMS	1. 后台数据库使用 SQL Server 2022, 所使用管理系统为 Microsoft SQL Server Management Studio 19

2 分		2. 在 SQL Server 配置管理器中，配置 SQL Native Client 11.0 的客户端协议，将其 TCP/IP 协议设置为启用，并设置默认端口 1433		
		3. 在 SSMS 中创建登录名 zy，设置密码为 123456，同时为其分配服务器角色为：public 以及 sysadmin		
	高级语言	1. 前台图形化界面使用 JAVA 编写，需要 JKD/JRE 环境，所使用集成环境为 Eclipse Java Oxygen		
		2. 需要将 JDBC 驱动 mssql-jdbc-12.2.0.jre11.jar 加载至引用库		
		3. 部分图形化界面开发，利用了 Eclipse Java Oxygen 的 Window Builder 插件		
连接串分析 (6 分)	序号	名称	功能说明	取值
	1	Driver Name	指向一个用户要使用的数据库的驱动程序	"com.microsoft.sqlserver.jdbc.SQLServerDriver"
	2	dbURL	指向要访问的数据库名	"jdbc:sqlserver://localhost:1433;DatabaseName=stu_mng"
	3.	userName	SQL server 身份验证的登录名	"sa"
	4.	userPassword	SQL server 身份验证的密码	"123456"
连接串代码 (截屏) (2 分)	<pre> public class Home { public static Connection cn; public static Statement st; //Statement 用于在已经建立数据库连接的基础上，向数据库发送要执行的 SQL 语句。 public static ResultSet rs; //ResultSet 用于接收数据库所有查询记录，并对其内容予以显示 public static int a; public static boolean openDB() { boolean joinFlag; try { joinFlag = true; Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver"); cn = DriverManager.getConnection("jdbc:sqlserver://localhost:1433;" + "DatabaseName=homework;" + "user=sa;" + "password=123456;" + "trustServerCertificate=true;"); cn.setCatalog("homework"); System.out.println("数据库连接成功"); st = cn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY); return joinFlag; } catch (SQLException sqlEx) { System.out.println(sqlEx.getMessage()); } } } </pre>			
备注				

3. 数据库设计 (14 分)

说明	(10 分) 按照数据表的创建顺序，依次给出所涉及数据表的信息，其中参照字段以“(字段 1, 字段 2, ……, 字段 n)”的形式给出，被参照字段以“表名(字段 1, 字段 2, ……, 字段 n)”的形式给出； (4 分) 一般 DBMS 都可以为数据库生成关系图，请将该图片截屏并粘贴到表格中。				
数据表 (10)	创建顺序	数据表名称	主键	参照属性	被参照表及属性
	1	course	courseno	无	无

	2	Major	majorno	无	无
	3	Student	stuno	stumajor	Major (majorno)
	4	college	collegeno	无	无
	5	Teacher	teacherno	teachercollege	College (collegeno)
	6	coursechoice	(stu, course)	stu	Student (stuno)
				course	Course (courseno)
	7	teach	(teacher, course)	teacher	Teacher (teacherno)
course				Course (courseno)	
关系图 (4)	<pre>graph TD student((student)) --- 1:M major((major)) student --- 1:M coursechoice((coursechoice)) major --- 1:M college((college*)) coursechoice --- 1:M course((course)) course --- 1:M teach((teach)) teach --- 1:M teacher((teacher*))</pre> <p>The diagram illustrates the following tables and their attributes:</p> <ul style="list-style-type: none">student: stuno (PK), stuname, stumajormajor: majorno (PK), majorname, majorstunumcollege *: collegeno (PK), collegename, college_majorcoursechoice: stu (FK), course (FK), scorecourse: courseno (PK), coursename, coursestumax, coursetimeteach: teacher (FK), teach_classteacher *: teacherno (PK), teachername, teacher_college <p>Relationships (1:M):</p> <ul style="list-style-type: none">student to majorstudent to coursechoicemajor to collegecoursechoice to coursecourse to teachteach to teacher				
	备注				

4. 含有事务应用的删除操作（13分）

说	（1分）简要说明该操作所要完成的功能：
---	---------------------

明	<p>(2分) 该操作会涉及的表(必须含有两张或两张以上的关系表,同时以“表名”的形式给出)</p> <p>(1分) 表连接涉及字段描述(描述方式为“表1.属性=表2.属性”)</p> <p>(1分) 删除条件涉及的字段描述(以“表名.属性=?”形式给出)</p> <p>(4分) 实现该操作的关键代码(高级语言、SQL),截图即可;(其中如果删除语句中不包含任何形式的事务应用将扣除3分)</p> <p>(4分) 如何执行该操作,按所述方法能够正常演示程序则给分。</p>	
功能描述 (1分)	<p>通过输入课程编号(即courseno)删除课程表(即course)中的课程信息</p> <p>情况一:若该课程不在course表中,不能删除,事务回滚</p> <p>情况二:若该课程在course表中,同时也在选课情况表(即coursechoice)和老师教课表(即teach)中,则先将coursechoice表和teach表中相关信息删除,再删除course表中相关信息</p> <p>情况三:若该课程在course表中,而不在coursechoice表中,则先将teach表中相关信息删除,再删除course表中的相关信息</p> <p>没有异常的情况下,事务commit;如果出现异常,事务rollback</p> <p>对于情况二的处理,之所以采用这种处理方式,是因为事务添加在了触发器里,如果没有这个触发器,由于coursechoice表中有相关行依赖于所要删除的course表中的行,如果执行相关删除,由于外键约束,删除操作将无法进行;而添加了含有事务的触发器之后,触发器采用instead of方式添加,instead of delete将使得相关delete不被执行,而去执行自定义的触发器内的操作。触发器内,设置操作为先删除coursechoice表中所有相关行,再删除course表相关行,进而使删除操作可以进行,最终证明实现删除功能的触发器发挥了作用。</p>	
涉及的表 (2分)	<p>course(courseno , coursename , coursestumax , coursetime)</p> <p>coursechoice(stu, course, score)</p> <p>teach(teacher, teach_class)</p>	
表连接涉及字段 (1分)	<p>@courseno_deleted in (select course from coursechoice)</p> <p>coursechoice.course=@courseno_deleted</p> <p>teach.teach_class=@courseno_deleted</p> <p>course.courseno=@courseno_deleted</p>	
删除条件 字段描述 (1分)	字段	规则
	coursechoice.course teach.teach_class	<p>如果 coursechoice 表中存在满足条件 coursechoice.course=@courseno_deleted 的行,先把这些行删除,再去 course 表中删除 teach.teach_class=@courseno_deleted course.courseno=@courseno_deleted 的行;</p> <p>如果 coursechoice 表中不存在满足条件 coursechoice.course=@courseno_deleted 的行, 且 course 表中存在满足条件 course.courseno=@courseno_deleted 的行, 则直接去 teach 表和 course 表中删除这些行;</p>

	course.courseno	<p>如果 coursechoice 表中不存在满足条件 coursechoice.course=@courseno_deleted 的行， 且 course 表中也不存在满足条件 course.courseno=@courseno_deleted 的行， 则不执行任何操作</p> <p>如果出现异常，例如用户输入的@courseno_deleted 为空等，则 事务回滚</p>
<p>代码 (4 分)</p>	<p>(截屏)</p> <p>Sql 代码:</p> <pre> create trigger delete_course on course instead of delete as begin transaction delete_cour save transaction delete_cour declare @courseno_deleted char(10) declare @course_name_deleted char(20) declare @coursestimax_deleted int declare @coursestime_deleted char(20) set @courseno_deleted=(select courseno from deleted) set @course_name_deleted=(select course_name from deleted) set @coursestimax_deleted=(select coursestimax from deleted) set @coursestime_deleted=(select coursestime from deleted) if(@courseno_deleted in (select course from coursechoice)) --若被删除的课程现正有学生选课，不能删除，事务回滚 begin delete from coursechoice where coursechoice.course=@courseno_deleted delete from teach where teach.teach_class=@courseno_deleted delete from course where course.courseno=@courseno_deleted end else begin delete from teach where teach.teach_class=@courseno_deleted delete from course where course.courseno=@courseno_deleted end if @@ERROR <> 0 begin rollback transaction delete_cour end else begin commit transaction delete_cour end </pre>	

这段代码中的整个触发器 (create trigger...as) 就是一个包含事务应用的删除操作。在这个触发器内部, 先执行了 `begin transaction delete_cour` 开始了一个事务, 随后保存了一次即将执行的 `delete course` 操作点。在接下来的代码中, 会根据被删除课程是否有学生选课来判断是否能够进行删除操作, 若不能, 代码会先执行“删除选课表 (coursechoice) 中选了该门课程的所有记录和教师教课的记录”, 再执行“删除课程表 (course) 中该门课程”。如果可以删除, 则只需要执行“教师教课记录和删除课程表中该门课程”操作。最后, 使用 `commit` 或 `rollback` 来提交或撤消整个事务的操作, 实现数据库操作的原子性。

Java 代码:

```
// 删除课程操作
public static void decourse(int coursens) throws SQLException{
    openDB();
    Statement stmt =cn.createStatement();
    try{
        cn.setAutoCommit(false);
        stmt = cn.createStatement();
        int a=stmt.executeUpdate("delete from course where courseno='" + coursens + "'");
        cn.commit(); //提交JDBC事务
        cn.setAutoCommit(true); // 恢复JDBC事务的默认提交方式
        if(a==1){
            JOptionPane.showMessageDialog(null, "删除课程成功", "成功", JOptionPane.INFORMATION_MESSAGE);
        }
        else{
            JOptionPane.showMessageDialog(null, "删除课程失败", "失败", JOptionPane.INFORMATION_MESSAGE);
        }
        stmt.close();
    }
    catch (Exception exc) {
        cn.rollback(); //回滚JDBC事务
        exc.printStackTrace();
        stmt.close();
    }
}
```


	<pre> class deListener implements ActionListener{ public void actionPerformed(ActionEvent e) { String STIQ; int rc=dtm.getRowCount(); for(int i=0;i<rc;i++){ dtm.removeRow(0); } String s =numgot.getText(); int key=Integer.parseInt(s); try { Home.decourse(key); //更新列表 String Tim="select * from 学生选课信息"; if(Home.query(Tim)){ System.out.println(Tim); try{ while(Home.rs.next()){ String XueShengBianHao=Home.rs.getString("学生编号"); String XueShengXingMing=Home.rs.getString("学生姓名"); String KeChengBianMa=Home.rs.getString("课程编号"); String KeChengMingCheng=Home.rs.getString("课程名称"); String KeChengChengJi=Home.rs.getString("课程成绩"); Vector v=new Vector(); v.add(XueShengBianHao); v.add(XueShengXingMing); v.add(KeChengBianMa); v.add(KeChengMingCheng); v.add(KeChengChengJi); dtm.addRow(v); //dtm是显示信息的表格 } } catch(Exception eTIQ){ System.out.println("初始化表格失败!"); } } //更新列表结束 } catch (SQLException e1) { e1.printStackTrace(); } } </pre>
<p>程序 演示 (4 分)</p>	<p>初始界面:</p>

课程信息删除

课程信息删除

课程代码：

删除课程

返回主菜单

学生学号	学生姓名	课程编号	课程名称	修课成绩
02	李四	..2004	数据库	...90
03	张三	..2005	C++	..

情况一：若该课程不在course表中，不能删除，事务回滚

课程信息删除

课程信息删除

课程代码：

删除课程

返回主菜单

学生学号	学生姓名	课程编号	课程名称	修课成绩
------	------	------	------	------

失败

删除课程失败

确定

情况二：若该课程在course表中，同时也在选课情况表(即coursechoice)和老师教课表(即

teach) 中，则先将coursechoice表和teach表中相关信息删除，再删除course表中相关信息

课程信息删除

课程信息删除

课程代码：

2004

删除课程

返回主菜单

学生学号	学生姓名	课程编号	课程名称	修课成绩
------	------	------	------	------

成功

删除课程成功

确定

课程信息删除

课程信息删除

课程代码：

2004

删除课程

返回主菜单

学生学号	学生姓名	课程编号	课程名称	修课成绩
03	张三	2005	C++	

情况三：若该课程在course表中，而不在coursechoice表中，则先将teach表中相关信息删

除，再删除course表中的相关信息
数据库中存在一个课程编号为 2007 的课程，删除 2007 后的界面为



备注 该可视化窗口的设计中，显示的并不是全部的课程，而是所有有学生选择的课程

5. 触发器控制下的添加操作（20 分）

说明	(1 分) 简要说明该操作所要完成的功能; (2 分) 简要说明该触发器所要完成的功能 (1 分) 该操作会涉及的表 (以“表名”的形式给出)。 (2 分) 该操作输入数据以及输入数据应该满足的条件, 如: 数值范围、是否为空; (6 分) 实现该操作的关键代码 (高级语言、SQL), 截图即可; (8 分) 如何执行该操作, 按所述方法能够正常演示程序则给分。	
功能描述 (1 分)	进行新生信息的录入, 录入新生的学号、姓名、专业 在触发器中添加事务, 检查输入的新生学号是否已经存在: 如果新生学号已经存在, 则不能插入 如果新生学号不存在, 且新生专业不存在, 则不能插入 如果新生学号存在, 且新生专业存在, 则执行插入, 并将对应专业人数增加 如果执行过程中出现异常, 例如输入的姓名为空等情况, 则不执行插入	
触发器描述 (2 分)	通过输入 stuno, stuname, stumajor 插入新生信息 1. 若该 stuno 已经存在于当前的 stu 表的 stuno 中, 则证明不是新生, 不能插入, 事务回滚 2. 若该 stuno 不存在于当前的 stu 表的 stuno 中, 且 stumajor 也不存在于当前的 major 表的 majorno 中, 则该新生专业错误, 不能插入, 事务回滚 3. 若该 stuno 不存在于当前的 stu 表的 stuno 中, 且 stumajor 存在于当前的 major 表的 majorno 中, 则正常插入, 并对所对应专业的专业人数进行维护, 即 majorstunum+1	
涉及的表 (1 分)	student(stuno, stuname, stumajor); major(majorno, majorname, majorstunum);	
输入数据 (2 分)	字段	规则
	student. stuno	如果 student 表中存在 满足条件 student. stuno = @stuno_insert 的行, 则不能执行插入, 事务回滚; 如果 student 表中不存在 满足条件 student. stuno = @stuno_insert 的行,
	student. stuname	且 major 表中不存在 满足条件 major. stumajor= @stumajor_insert 的行, 则不能执行插入, 事务回滚; 如果 coursechoice 表中不存在 满足条件 coursechoice. course=@courseno_deleted 的行, 且 major 表中存在 满足条件
	student. majorstunum	major. stumajor= @stumajor_insert 的行, 则执行插入, 并且 major 的 majorstunum 增加 1 如果出现异常, 例如用户输入的@stuname_insert 为空等, 则事务回滚

插入 操作 源码 (3 分)	<p>Java 代码:</p> <pre> class insertListener implements ActionListener{ public void actionPerformed(ActionEvent e) { String STIQ; int rc=dtm.getRowCount(); //***** t=dtm.getRowCount(); //插入前的表格行数 //***** for(int i=0;i<rc;i++){ dtm.removeRow(0); } String s1 =xuehaogot.getText(); String s2 =xingminggot.getText(); String s3 =zhuanwegot.getText(); try { Home.insertstu(s1,s2,s3); //更新列表 String Tim = "select * from 学生专业情况"; if(Home.query(Tim)){ try{ while(Home.rs.next()){ String XueShengBianHao=Home.rs.getString("学生编号"); //getString("")中双引号里的是表格的列的名字 String XueShengXingMing=Home.rs.getString("学生姓名"); String ZhuanYeMingCheng=Home.rs.getString("专业名称"); String ZhuanYeBianHao=Home.rs.getString("专业编号"); String ZhuanYeRenShu=Home.rs.getString("专业人数"); Vector v=new Vector(); v.add(XueShengBianHao); v.add(XueShengXingMing); v.add(ZhuanYeMingCheng); v.add(ZhuanYeBianHao); v.add(ZhuanYeRenShu); dtm.addRow(v); //dtm是显示信息的表格 } //***** k=dtm.getRowCount(); //插入后的表格行数 //***** if(t != k){ JOptionPane.showMessageDialog(null,"录入新生成功", "成功", JOptionPane.INFORMATION_MESSAGE) } else{ JOptionPane.showMessageDialog(null,"录入新生失败", "失败", JOptionPane.INFORMATION_MESSAGE) } } catch(Exception eTIQ){ System.out.println("初始化表格失败!"); } } //更新列表结束 } catch (SQLException e1) { e1.printStackTrace(); } } </pre>	

	<pre> //新生录入操作 public static void insertstu(String s1,String s2,String s3) throws SQLException{ openDB(); Statement stmt =cn.createStatement(); try{ cn.setAutoCommit(false); stmt = cn.createStatement(); int a=stmt.executeUpdate("insert into student values('"+s1+"','"+s2+"','"+s3+"');"); cn.commit(); //提交JDBC事务 cn.setAutoCommit(true); // 恢复JDBC事务的默认提交方式 stmt.close(); } catch (Exception exc) { cn.rollback(); //回滚JDBC事务 exc.printStackTrace(); stmt.close(); } } </pre>
触发器源代码 (3分)	<p>(截屏)</p> <p>Sql 代码:</p>

```

create trigger insert_student on student
instead of insert
as
begin transaction insert_stu
save transaction insert_stu

    declare @stuno_insert char(10)
    declare @stuname_insert char(20)
    declare @stumajor_insert char(10)
    set @stuno_insert=(select stuno from inserted)
    set @stuname_insert=(select stuname from inserted)
    set @stumajor_insert=(select stumajor from inserted)

    if(@stuno_insert in (select stuno from student))
        --若该生学号已存在于student表中
        begin
            print 'student exist'
            rollback transaction insert_stu
        end
    else
        --若该生学号未存在于student表中
        if(@stumajor_insert not in(select majorno from major))
            --若该生专业未存在于major表中
            begin
                print 'majorno not exist'
                rollback transaction insert_stu
            end
        else
            --若该生专业存在于major表中
            begin
                --在student表中插入该生信息
                insert into student
                values(@stuno_insert,@stuname_insert,@stumajor_insert)

                --维护major表中的majorstunum信息
                update major
                set major.majorstunum=major.majorstunum+1
                where major.majorno=@stumajor_insert
            end
        end
    if @@ERROR<>0
        begin
            rollback transaction insert_stu
        end
    else
        begin
            commit transaction insert_stu
        end
end

```

程序
演示

说明：不违背触发器能够执行插入操作。
初始界面：

(4
分)

新生信息录入

新生信息录入

新生学号:

新生姓名:

专业编号:

新生录入

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	.. 计算机	...1003	181
02	李四	.. 密码学	...1001	32
03	张三	.. 密码学	...1001	32
04	王五	.. 物联网	...1002	32
05	张洋	.. 信息安全	...1000	50
06	哈哈	.. 信息安全	...1000	50
1	啦啦	.. 物联网	...1002	32

若该 stuno 不存在于当前的 stu 表的 stuno 中,且 stumajor 存在于当前的 major 表的 majorno 中,则正常插入,并对所对应专业的专业人数进行维护,即 majorstunum+1

新生信息录入

新生信息录入

新生学号:

07

新生姓名:

111

专业编号:

1003

新生录入

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	.. 计算机	...1003	182
02	李四	.. 密码学	...1001	32
03	张三	.. 密码学	...1001	32
04	王五	.. 物联网	...1002	32
05	成功			0
06				0
07				82
1				2

成功

录入新生成功

确定

录入前编号为 1003 的专业人数为 181, 录入后变为 182

新生信息录入

新生信息录入

新生学号：

07

新生姓名：

111

专业编号：

1003

新生录入

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	..计算机	..1003	182
02	李四	..密码学	..1001	32
03	张三	..密码学	..1001	32
04	王五	..物联网	..1002	32
05	张洋	..信息安全	..1000	50
06	哈哈	..信息安全	..1000	50
07	111	..计算机	..1003	182
1	啦啦	..物联网	..1002	32

说明：违背触发器要求，不能够执行插入操作，系统报错。
初始界面：

新生信息录入

新生信息录入

新生学号：

新生姓名：

专业编号：

新生录入

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	..计算机	..1003	181
02	李四	..密码学	..1001	32
03	张三	..密码学	..1001	32
04	王五	..物联网	..1002	32
05	张洋	..信息安全	..1000	50
06	哈哈	..信息安全	..1000	50
1	啦啦	..物联网	..1002	32

1. 若该 stuno 已经存在于当前的 stu 表的 stuno 中，则证明不是新生，不能插入，事务回滚

新生信息录入

新生信息录入

新生学号：

新生姓名：

专业编号：

新生录入

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	计算机	1003	181
02	李四	密码学	1001	32
03	张三	密码学	1001	32
04	王五	物联网	1002	32
05	失败			0
06				0
1				2

失败

录入新生失败

确定

2. 若该 stuno 不存在于当前的 stu 表的 stuno 中，且 stumajor 也不存在于当前的 major 表的 majorno 中，则该新生专业错误，不能插入，事务回滚
插入不存在的 major

新生信息录入

新生信息录入

新生学号：

新生姓名：

专业编号：

新生录入

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	计算机	1003	181
02	李四	密码学	1001	32
03	张三	密码学	1001	32
04	王五	物联网	1002	32
05	失败			0
06				0
1				2

失败

录入新生失败

确定

备注

6. 存储过程控制下的更新操作（18分）

说明	(1分) 简要说明该操作所要完成的功能; (1分) 简要说明该存储过程所要完成的功能; (2分) 说明该操作涉及操作的表(必须包含两张或两张以上的关系表,以“表名形式”描述) (1分) 表连接涉及字段描述(描述方式为“表1.属性=表2.属性”) (2分) 该操作会修改字段(以“表名.字段名”的形式给出),以及修改规则,如新数值的计算方法、在何种条件下予以修改等; (6分) 实现该操作的关键代码(高级语言、SQL),截图即可; (5分) 如何执行该操作,按所述方法能够正常演示程序则给分。	
功能描述 (1分)	完成转专业同学的专业信息更新 输入申请转专业的学生的学号以及其所要转入的专业 如果学号存在且目标专业与该生当前专业不同,则修改该生的专业信息,并将其转出专业的专业人数属性减一,将其转入专业的专业人数属性加一	
存储过程功能描述 (1分)	存储过程的两个参数为输入 学生学号@applicantid 和 目标专业@goalmajor 分别对应该学生的学号 stuno 以及申请转入的专业的编号 majorno 形参@oldmajor 赋值为@applicantid 对应学生的当前的 stumajor 形参@newmajor 赋值为@goalmajor 若输入的目标专业@newmajor 与当前专业@oldmajor 相同,则转专业失败 否则该生转专业成功,维护转入专业和转出专业的人数变化	
涉及的关系表 (2分)	student(stuno, stuname, stumajor); major(majorno, majorname, majorstunum);	
表连接涉及字段 (1)	student.stumajor = major.majorno	
更改字段 (2分)	字段	规则
	student.stuno	如果 student 表中不存在满足条件 student. stuno = @applicantid 的行, 则不能执行插入; 如果 student 表中存在满足条件

	<div>student.stumajor</div> <div>major.majorno</div> <div>major.majorstunum</div>	<div>student. stuno = @applicantid 的行， 但是该行满足条件 student.stumajor = @newmajor，即要转入的专业和原专业相同，则不能执行插入； 如果 student 表中存在满足条件 student. stuno = @applicantid 的行， 且该行不满足条件 student.stumajor = @newmajor， 则执行插入, 并且 major 表中满足条件 major.majorno = @newmajor 的行的 majorstunum 增加 1 major.majorno = @oldmajor 的行的 majorstunum 减少 1</div>
更新 代码 (3 分)	<div>(截屏)</div> <pre>class updateListener implements ActionListener{ public void actionPerformed(ActionEvent e) { String STIQ; int rc=dtm.getRowCount(); for(int i=0;i<rc;i++){ dtm.removeRow(0); } String s1 =xuehaogot.getText(); String s2 =newmajorgot.getText(); try { Home.updatestu(s1,s2); //更新列表 String Tim = "select * from 学生专业情况排序"; if(Home.query(Tim)){ try{ while(Home.rs.next()){ String XueShengBianHao=Home.rs.getString("学生编号"); //getString("") 中双引号里的是表格的列的名字 String XueShengXingMing=Home.rs.getString("学生姓名"); String ZhuanYeMingCheng=Home.rs.getString("专业名称"); String ZhuanYeBianHao=Home.rs.getString("专业编号"); String ZhuanYeRenShu=Home.rs.getString("专业人数"); Vector v=new Vector(); v.add(XueShengBianHao); v.add(XueShengXingMing); v.add(ZhuanYeMingCheng); v.add(ZhuanYeBianHao); v.add(ZhuanYeRenShu); dtm.addRow(v); //dtm是显示信息的表格 } } catch(Exception eTIQ){ System.out.println("初始化表格失败！"); } } //更新列表结束 } } catch (SQLException e1) { e1.printStackTrace(); }</pre>	
创建 存储	<div>(截屏)</div>	

过程
源码
(3
分)

```
create procedure change_major_proc
    @applicantid char(10) output, @goalmajor char(10) output
as
begin
    declare @oldmajor char(10)
    declare @newmajor char(10)
    set @oldmajor=(select major.majorno
                    from student,major
                    where student.stumajor=major.majorno
                    and student.stuno=@applicantid)
    set @newmajor=@goalmajor

    if(@oldmajor<>@newmajor)
        begin
            --若转出专业与转入专业不同，则本次转专业操作有效
            --更新该生的专业信息
            update student
            set student.stumajor=(select majorno
                                   from major
                                   where major.majorno=@newmajor)
            where student.stuno=@applicantid

            --更新转出班级及转入班级的人数
            update major
            set major.majorstunum=major.majorstunum-1
            where major.majorno=(select majorno
                                   from major
                                   where major.majorno=@oldmajor)

            update major
            set major.majorstunum=major.majorstunum+1
            where major.majorno=(select majorno
                                   from major
                                   where major.majorno=@newmajor)
        end
    end
```

存储

(截屏)

过程
执行
源码
(1
分)

```
//转专业信息更新操作
public static void updatestu(String s1,String s2) throws SQLException{//s1是学生学号,s2是新专业编号
    openDB();
    CallableStatement stmt =cn.prepareCall("{call change_major_proc(?,?)}");
    try{
        cn.setAutoCommit(false);
        stmt.setNString(1,s1);
        stmt.setNString(2,s2);
        stmt.registerOutParameter(1, Types.INTEGER);
        stmt.registerOutParameter(2, Types.INTEGER);
        a=0;
        a=stmt.executeUpdate();
        cn.commit();//提交JDBC事务
        cn.setAutoCommit(true);// 恢复JDBC事务的默认提交方式

        if(a==1){
            JOptionPane.showMessageDialog(null,"转专业信息更新成功", "成功", JOptionPane.INFORMATION_MESSAGE);
        }
        else{
            JOptionPane.showMessageDialog(null,"转专业信息更新失败", "失败", JOptionPane.INFORMATION_MESSAGE);
        }
        stmt.close();
    }
    catch (Exception exc) {
        cn.rollback();//回滚JDBC事务
        exc.printStackTrace();
        stmt.close();
    }
}
```

说明：不违背存储过程，能够执行更新操作
初始界面：

学生学号	学生姓名	专业名称	专业编号	专业人数
05	张洋	信息安全	1000	50
06	哈哈	信息安全	1000	50
02	李四	密码学	1001	32
03	张三	密码学	1001	32
04	王五	物联网	1002	32
1	啦啦	物联网	1002	32
01	嘻嘻	计算机	1003	182
07	111	计算机	1003	182

程序
演示
(2
分)

把学号为 01 的学生从专业编号为 1003 转到专业编号为 1000

转专业信息更新

转专业信息更新

学生学号：

01

新专业编号：

1000

更新专业信息

返回主菜单

成功

i

转专业信息更新成功

确定

专业人数

初始专业编号为 1003 的人数为 182，转专业后变为 181，
初始专业编号为 1000 的人数为 50，转专业后变为 51

转专业信息更新

转专业信息更新

学生学号：

01

新专业编号：

1000

更新专业信息

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	..信息安全 ..	1000	51
05	张洋	..信息安全 ..	1000	51
06	哈哈	..信息安全 ..	1000	51
02	李四	..密码学 ..	1001	32
03	张三	..密码学 ..	1001	32
04	王五	..物联网 ..	1002	32
1	啦啦	..物联网 ..	1002	32
07	111	..计算机 ..	1003	181

说明：违背存储过程，系统报错；
初始界面：

转专业信息更新

转专业信息更新

学生学号：

新专业编号：

更新专业信息

返回主菜单

学生学号	学生姓名	专业名称	专业编号	专业人数
01	嘻嘻	.. 信息安全 ..	1000	51
05	张洋	.. 信息安全 ..	1000	51
06	哈哈	.. 信息安全 ..	1000	51
02	李四	.. 密码学 ..	1001	32
03	张三	.. 密码学 ..	1001	32
04	王五	.. 物联网 ..	1002	32
1	啦啦	.. 物联网 ..	1002	32
07	111	.. 计算机 ..	1003	181

1. 如果 student 表中不存在满足条件 student. stuno = @applicantid 的行，则不能执行插入；

转专业信息更新

转专业信息更新

学生学号：

11

新专业编号：

1000

更新专业信息

返回主菜单

失败

转专业信息更新失败

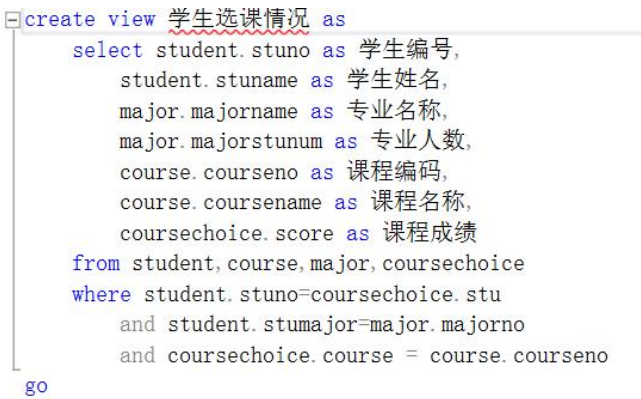
确定

学生学号	学生姓名	专业名称	专业编号	专业人数
------	------	------	------	------

程序
演示
(2
分)

	<p>2. 如果 student 表中存在满足条件 student. stuno = @applicantid 的行，但是该行满足条件 student. stumajor = @newmajor，即要转入的专业和原专业相同，则不能执行插入；学号 01 的学生专业为 1000，如果让其转入专业编号为 1000 的专业，则显示转专业失败</p> 
备注	<p>该窗口中的各行，是按照专业编号进行排序的，以方便查看转专业后的变化情况。具体实现办法是，先正常地在 student、major 两个表上创建视图“学生专业情况”，再在“学生专业情况”这个视图上再创建另一个视图“学生专业情况排序”，从“学生专业情况”视图中获取数据，并按照指定顺序显示。</p>

7. 含有视图的查询操作（15 分）

说明	<p>(1 分) 简要说明该操作所要完成的功能;</p> <p>(1 分) 简要说明建立的该视图的功能;</p> <p>(2 分) 简要说明该操作涉及的关系数据表 (以 “表名” 的形式给出)</p> <p>(1 分) 简要说明表连接涉及的字段 (以 “表 1. 属性=表 2. 属性”)</p> <p>(6 分) 实现该操作的关键代码 (高级语言、SQL), 截图即可;</p> <p>(4 分) 如何执行该操作, 按所述方法能够正常演示程序则给分。</p>
操作功能描述 (1 分)	<p>完成学生选课情况信息的查询</p> <p>不输入任何内容, 点击查询按钮, 可以查询到所有学生的所有选课情况</p> <p>如果只输入学生学号, 点击查询按钮, 可以查询该生的所有选课情况 (若该生存在)</p> <p>如果只输入课程名称, 点击查询按钮, 可以查询该课程的所有被选择情况 (若该课程存在)</p> <p>如果同时输入学生学号和个人信息, 点击查询按钮, 可以该生选择该课程的选课情况 (若该生存在, 该课程存在, 且该生选择了该课程)</p>
视图功能描述 (1 分)	<p>在 student、major、course、coursechoice 四个表上创建视图, 视图中的信息包括:</p> <p>学生编号 (来自 student)、学生姓名 (来自 student)、专业名称 (来自 major)、专业人数 (来自 major)、课程编码 (来自 course)、课程名称 (来自 course)、课程成绩 (来自 coursechoice)</p> <p>只有选了课的学生的信息才会出现在视图中</p> <p>只有有学生的专业的信息才会出现在视图中</p> <p>只有有学生选的课的信息才会出现在视图中</p>
涉及的关系表 (2 分)	<p>student(stuno, stuname, stumajor)</p> <p>major(majorno, majorname, majorstunum)</p> <p>course(courseno, coursename, coursestunum, coursetime)</p> <p>coursechoice(stu, course, score)</p>
表连接字段 (1 分)	<p>student.stuno=coursechoice.stu</p> <p>student.stumajor=major.majorno</p> <p>coursechoice.course = course.courseno</p>
创建视图代码 (3 分)	<p>(截屏)</p>  <pre> create view 学生选课情况 as select student.stuno as 学生编号, student.stuname as 学生姓名, major.majorname as 专业名称, major.majorstunum as 专业人数, course.courseno as 课程编码, course.coursename as 课程名称, coursechoice.score as 课程成绩 from student, course, major, coursechoice where student.stuno=coursechoice.stu and student.stumajor=major.majorno and coursechoice.course = course.courseno go </pre>
查询	<p>(截屏)</p>

代码
(3
分)

```
//为查询按钮加事件-----
queryBtn.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        System.out.println("查询按钮加事件");
        String STIQ;
        int rc=dtm.getRowCount();
        for(int i=0;i<rc;i++){
            dtm.removeRow(0);
        }
    }
});
//-----
if(stunumgot.getText().equals("")&&cnamegot.getText().equals("")){//两者都为空，显示全部信息
    STIQ="select * from 学生选课情况";
}
else if(stunumgot.getText().equals("")&&!cnamegot.getText().equals("")){//学号栏为空，课程栏不为空，显示课程相关信息
    STIQ="select * from 学生选课情况 where 课程名称 = '" + cnamegot.getText() + "'";
}
else if(!stunumgot.getText().equals("")&&cnamegot.getText().equals("")){//学号栏不为空，课程栏为空，显示学生相关信息
    STIQ="select * from 学生选课情况 where 学生编号 = '" + stunumgot.getText() + "'";
}
else{//学号栏不为空，课程栏不为空，显示学生课程相关信息
    STIQ="select * from 学生选课情况 where 学生编号 = '" + stunumgot.getText() + "' and 课程名称 = '" + cnamegot.getText() + "' ";
}
System.out.println(STIQ);
if(Home.query(STIQ)){
    try{
        while(Home.rs.next()){
            String XueShengBianHao=Home.rs.getString("学生编号"); //getString("")中双引号里的是表格的列的名字
            String XueShengXingMing=Home.rs.getString("学生姓名");
            String ZhuanYeMingCheng=Home.rs.getString("专业名称");
            String ZhuanYeRenShu=Home.rs.getString("专业人数");
            String KeChengBianMa=Home.rs.getString("课程编码");
            String KeChengMingCheng=Home.rs.getString("课程名称");
            String KeChengChengJi=Home.rs.getString("课程成绩");

            Vector v=new Vector();
            v.add(XueShengBianHao);
            v.add(XueShengXingMing);
            v.add(ZhuanYeMingCheng);
            v.add(ZhuanYeRenShu);
            v.add(KeChengBianMa);
            v.add(KeChengMingCheng);
            v.add(KeChengChengJi);

            dtm.addRow(v); //dtm是显示信息的表格
        }
    } catch (Exception eT) {}
}

//查询操作
public static boolean query(String sqlString) {
    openDB();
    try {
        rs = null;
        System.out.println(sqlString);
        rs = st.executeQuery(sqlString);
    } catch (Exception Ex) {
        System.out.println("sql exception+ " + Ex);
        return false;
    }
    return true;
}
```

程序
演示
(4
分)

初始界面:

学生修课情况查询

学生学号:

课程名称:

查 询

返回主菜单

学生编号	学生姓名	专业名称	专业人数	课程编码	课程名称	课程成绩
01	晒晒	信息安全	51	2004	计组	60
02	李四	密码学	32	2001	python	59
02	李四	密码学	32	2002	算法	80
02	李四	密码学	32	2003	数据结构	50
03	张三	密码学	32	2000	数据库	70
03	张三	密码学	32	2005	C++	100

查询学号为 02 的学生选课情况

学生选课情况查询

学生学号：

02

课程名称：

查 询

返回主菜单

学生选课情况查询

学生编号	学生姓名	专业名称	专业人数	课程编码	课程名称	课程成绩			
02	李四	...	密码学	...	32	2001python	...	59	
02	李四	...	密码学	...	32	2002	算法	...	80
02	李四	...	密码学	...	32	2003	数据结构	...	50

查询数据库课程的选课情况

学生选课情况查询

学生学号：

课程名称：

数据库

查 询

返回主菜单

学生选课情况查询

学生编号	学生姓名	专业名称	专业人数	课程编码	课程名称	课程成绩		
03	张三	...	密码学	...	32	2000数据库	...	70

查询学号为 02 的学生选择“python”课程的情况

学生选课情况查询

学生学号：

02

课程名称：

python

查 询

返回主菜单

学生选课情况查询

学生编号	学生姓名	专业名称	专业人数	课程编码	课程名称	课程成绩		
02	李四	...	密码学	...	32	2001python	...	59

查询学号为 02 的学生选择“数据库”课程的情况（查询不存在）

	<div><div>学生选课情况查询</div><div><div>学生选课情况查询</div><div><div>学生学号：<input type="text" value="02"/></div><div>课程名称：<input type="text" value="数据库"/></div><div>查 询</div><div>返回主菜单</div></div><table><tr><th>学生编号</th><th>学生姓名</th><th>专业名称</th><th>专业人数</th><th>课程编码</th><th>课程名称</th><th>课程成绩</th></tr><tr><td colspan="7"></td></tr></table></div></div>	学生编号	学生姓名	专业名称	专业人数	课程编码	课程名称	课程成绩							
学生编号	学生姓名	专业名称	专业人数	课程编码	课程名称	课程成绩									
备注	该视图按照学生学号进行排序														