哈尔滨工业大学

<<数据库系统>>

实验报告一

(2020 年度春季学期)

姓名：	张景润
学号：	1172510217
学院：	计算机学院
教师：	程思瑶

实验一：数据库系统开发

一、 实验目的

熟练掌握 MSQL 基本命令、SQL 语言以及用 C 在语言编写的 MySQl 操作程序上，学习简单数据库系统的设计方法，包括数据库概要设计和逻辑设计。

二、 实验环境

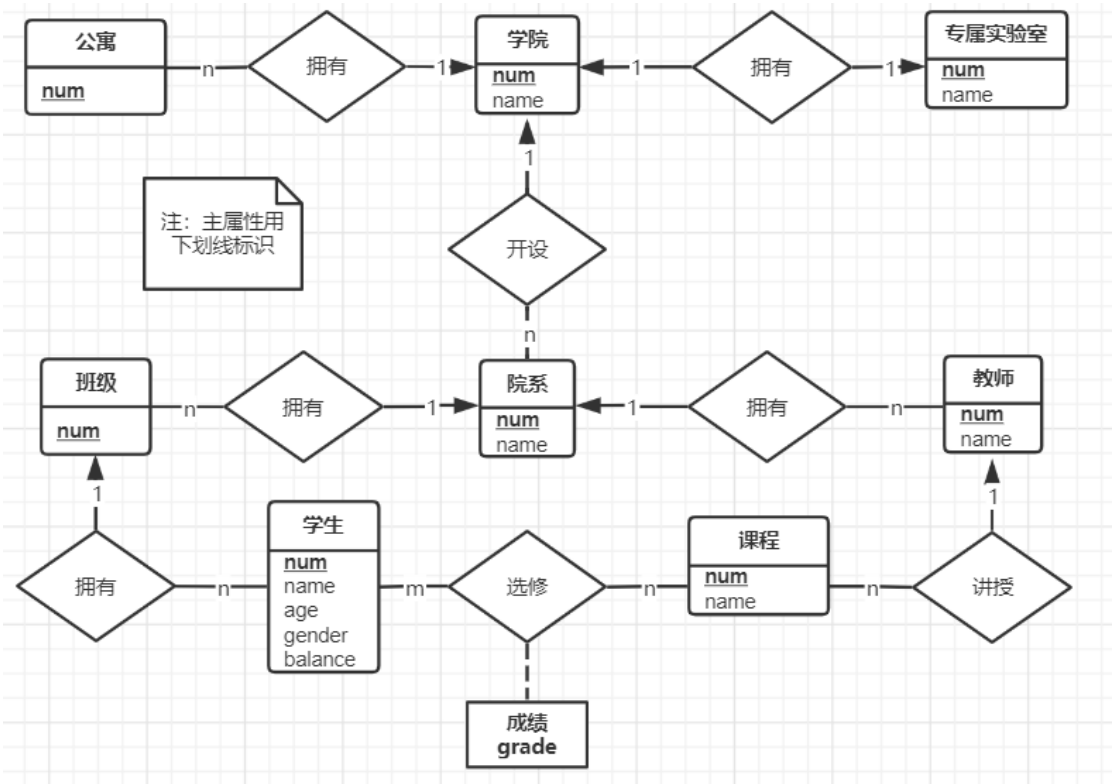
Windows XP 操作系统、MySQL 关系数据库管理系统、Microsoft Visual C++编译器或 MinGW 编译器。本实验可采用 C、C++、JAVA、PHP 等。

三、 实验过程

3.1 系统介绍

系统主要功能是记录学生选课等信息。主要实体是学生、课程以及教师，主要联系是选课等。共有 8 个实体，9 个联系。下图是具体的 ER 图：

其中，学院和专属实验室是一对一联系，学院对院系是一对多联系，学生和课程是多对多联系。学生与课程的联系“选修”带有属性成绩 grade。

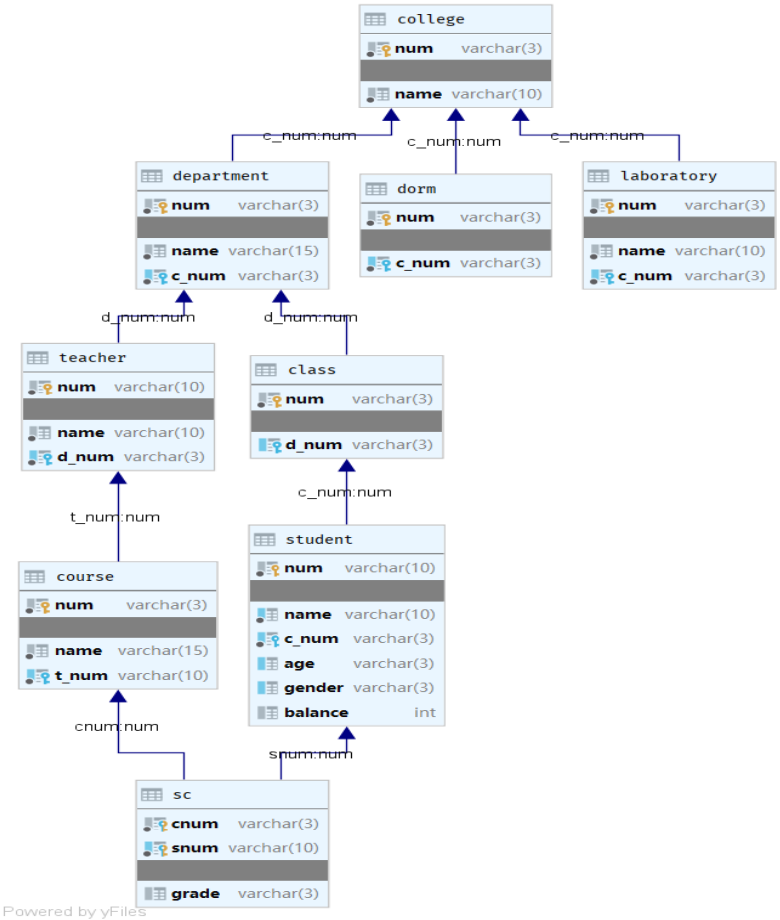


关系表如下表（主码用下划线标注，外码在描述中标注）：

关系	描述
college(num, name)	学院：学院号，学院名
department(num, name, c_num)	院系：院系号，院系名以及院系所在

	的学院号（外键）
dorm(num, c_num)	公寓：公寓号，以及公寓所属的学院号（外键）
laboratory(num, name, c_num)	学院专属实验室：实验室编号，实验室名，以及实验室所属的学院号（外键）
teacher(num, name, d_num)	教师：教师号，教师名以及教师所在的院系号（外键）
class(num, d_num)	班级：班级号，以及班级所在的院系号（外键）
course(num, name, t_num)	课程：课程号，课程名，以及开设课程的教师号（外键）
student(num, name, c_num, age, gender, balance)	学生：学号，姓名，学生所在的班级号（外键），年龄，性别和账户余额
sc(cnum, snum, grade)	选课表：课程号（外键），学号（外键）以及对应的成绩

pycharm 内置的数据库管理模块生成的结构图如下：



3.2 完成部分

系统功能包括：插入与删除；连接、嵌套以及分组查询；建立视图与索引；所有附加功能，事务管理与触发器；以及十分良好的界面交互体验。

插入与删除。插入与删除已经设计实现了完整性约束：如实体完整性中的插入删除空值前会提示，插入重复值前也会警告；参照完整性中的的删除一个被引用的外键前会提示，插入一个未被定义的外键引用也会给予警告（具体表现是学生表和选课表中）。

连接、嵌套与分组。使用 SQL 语句进行了较为全面的连接查询、嵌套查询和分组查询，其中查询体现了分组和 Having 语句。

视图与索引。为了隐藏部分数据库表的细节并展示给用户特定的表属性，实验实现了为常用的查询创建索引的功能；同时还可以为常用的属性（非主键）建立索引。

事务管理。使用 pymysql 程序实现了事务管理，保证了转账功能的原子性、一致性、隔离性与持久性。

触发器。使用 pymysql 程序实现了触发器。设计了几条语句，当对数据修改的时候，就会自动执行。实验中，我使用触发器来保证选课表 SC 的参照完整性。当插入一个选课记录时，就会执行触发器（执行条件），动作是判断相应课程和学生是否存在（触发器的动作）；当删除一个课程记录时，同样执行触发器（执行条件），动作是判断是否存在选课记录中的课程号等于被删除的课程号（触发器的动作）。

3.3 完整性约束

插入与删除的基本语句结构如下：

```
insert into student(num,name,c_num) values('1172510217','张景润','001')
delete from student where num='1172510217'
```

要保证完整性约束，就要保证实体完整性约束和参照完整性约束。

实体完整性约束，主码属性必须是非空且唯一的。实验中，在插入与删除学生表和课程表之前，必须保证信息不为空；且在插入新的信息的时候，必须保证插入的主属性在原表中不存在。如果违反以上要求，则需要弹出警示信息，停止下一步的操作。

参照完整性约束，在参照关系中任意元组在特定属性上的取值必然等于被参照关系中某个元组在特定属性上的取值。实验中，在插入选课表 SC 时，必须保证新插入的选课信息中的学号和课程号已经在学生表和课程表中存在。如果不存在，则违反参照完整性约束，弹出警示信息，停止下一步的操作；在删除学生表 Student 信息时，必须保证要删除的学生信息未被选课表引用。若正在

被引用，则违反参照完整性约束，弹出警示信息，停止下一步的操作；类似地，在删除课程表信息时，也要进行相同的逻辑判断。

通用代码逻辑是，在执行插入与删除之前，先构造查询语句，查询对应项是否存在。然后根据查询结果进行相应的处理：报错警示或者正常进行。

3.4 创建视图

创建视图的语句如下：

```
create view cs_student as select num, name from student where c_name='计算机'
```

出于安全考虑，SQL 允许通过查询来定义“虚关系”，从而向用户隐藏特定的数据。创建的视图比逻辑模型更符合用户的直觉。

实验中为计算机学院的各个院系建立视图，如自然语言处理专业学生视图，视听觉信号处理专业学生视图等。

3.5 触发器

实验中，使用 pymysql 程序实现了触发器。设计了几条语句，当对数据修改的时候，就会自动执行。

可以使用触发器来保证选课表 SC 的参照完整性。当插入一个选课记录时，就会执行触发器（执行条件），动作是判断相应课程和学生是否存在（触发器的动作）；当删除一个课程记录时，同样执行触发器（执行条件），动作是判断是否存在选课记录中的课程号等于被删除的课程号（触发器的动作）。

由于触发器是作用在每一条记录上的，因此触发器的设计一定要简单与高效，否则会极大地占用处理时间。

3.6 事务管理

为学生设置属性 balance，用来表示饭卡余额。不同学生之间可以进行转账操作。而转账操作就是一个很好的事务管理的例子。

事务管理要求事物具有原子性、一致性、隔离性与持久性。若在执行原子的操作语句的过程中，遇到了异常或错误，就会执行回退操作。

当学生自己给自己转账时，提示转账失败；当用户余额小于转账金额时，提示余额不足；当我们在实验中为转账加入一个模拟异常，用于表示不可预测的系统内部错误时，提示转账失败。之后，失败的转账都会进行回退操作，余额恢复到转账操作前，保证事务管理的原子性。

四、 实验结果

4.1 范式分析

满足第一范式要求 1NF。由于数据库模式中的每一个关系 R 中的每一个属

性的值域都是不可分的简单数据项的集合，所以设计的数据库模式满足第一范式要求。

满足第二范式要求 2NF。对于数据库模式的每一个关系 R，其中的每一个非键属性都完全函数依赖于 R 的候选码，所以设计的数据库模式满足第二范式要求。下表是详细分析：

关系表	非键属性	候选码	完全函数依赖于候选码
sc	grade	(snum, cnum)	是
student	name, c_num, age, gender, balance	num	是
course	name, t_num	num	是
teacher	name, d_num	num	是
class	d_num	num	是
department	name, c_num	num	是
dorm	c_num	num	是
laboratory	name, c_num	num	是
college	name	num	是

满足第三范式要求 3NF。对于数据库模式中的每一个关系 R，其中的每一个非键属性都不传递地依赖于任何候选码，所以设计的数据库模式满足第三范式要求。下表是详细分析：

关系表	非键属性	候选码	非键属性不传递函数依赖于候选码
sc	grade	(snum, cnum)	是
student	name, c_num, age, gender, balance	num	是
course	name, t_num	num	是
teacher	name, d_num	num	是
class	d_num	num	是
department	name, c_num	num	是
dorm	c_num	num	是
laboratory	name, c_num	num	是
college	name	num	是

满足 BC 范式要求 BCNF。对于数据库模式中的每一个关系 R，其中的每一个函数依赖 X->Y，则 X 必为候选码，所以设计的数据库模式满足 BC 范式要求。下表是详细分析：

关系表	函数依赖 X->Y	候选码	X 为候选码

sc	(snum, cnum)->grade	(snum, cnum)	是
student	num->name; num->c_num num->age; num->gender num->balance	num	是
course	num ->name; num ->t_num	num	是
teacher	num->name; num->d_num	num	是
class	num->d_num	num	是
department	num->name; num->c_num	num	是
dorm	num->c_num	num	是
laboratory	num->name; num->c_num	num	是
college	num->name	num	是

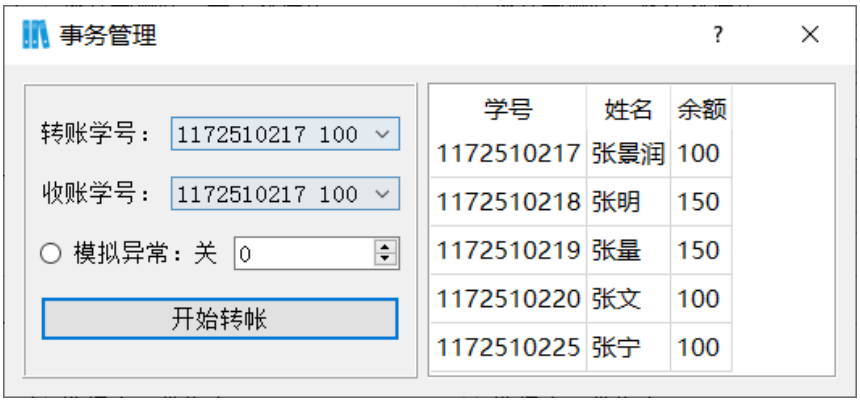
4.2 运行整体界面

整体风格。界面采用菜单栏，使得功能简洁明了；采用下拉选项框，使得用户输入尽可能少同时保证系统的健壮性；逻辑设计合理，通过选项框“触发器开关”打开触发器，使得程序整体可以使用触发器逻辑，从而保证系统的参照完整性约束。

综合界面如下表所示：许多功能使用下拉选项框实现，保证了程序的健壮性。主界面包括所有的基本功能和附加功能：触发器（以选项框形式展现）。

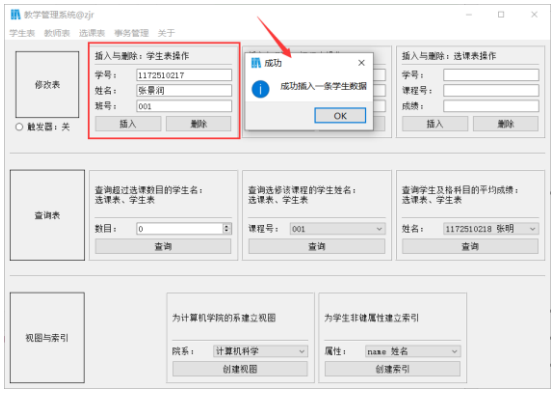
第二个附加功能事务管理的界面如下图所示（通过点击菜单栏中的事务管

理实现）。使用了下拉选项框 combobox 来简化用户输入，同时保证了系统的健壮性；在界面右侧添加表格，展示转账的余额结果动态变化，更加清晰明了。

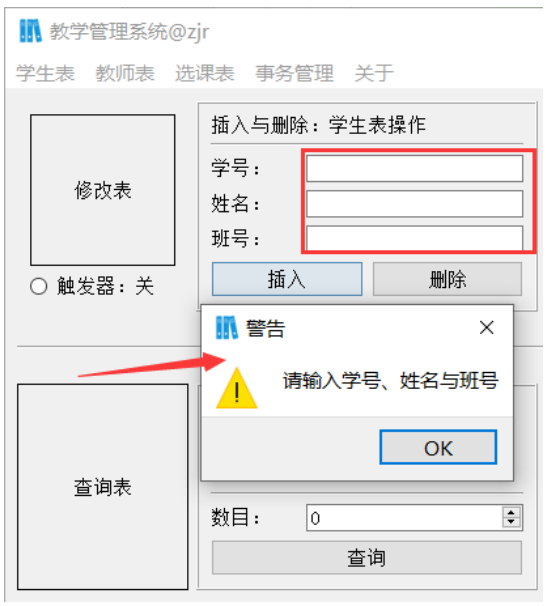


4.3 插入与删除

插入或删除正常值。学生表中插入记录'1172510217','001','张景润'，结果插入成功；删除学号'1172510217'，结果删除成功，如下图所示（弹窗）。



插入或删除空值。



插入重复值或删除不存在的值（即重复删除）

4.4 连接、嵌套、分组查询与创建视图和索引

连接查询、Having 语句与聚集函数。pymysql 构造的查询语句如下：

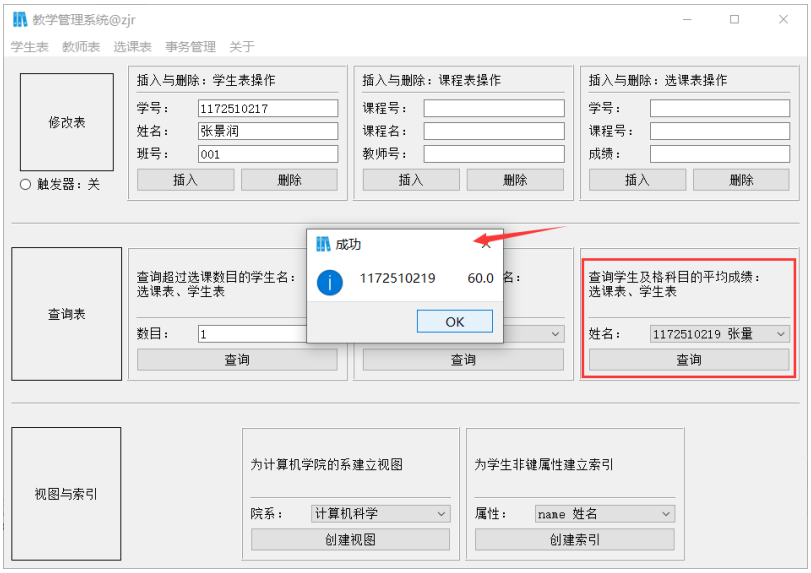
```
select student.name from student, sc where student.num = sc.snum group by student.num HAVING count(*) > %s
```

。查询界面与结果截图如下：

嵌套查询与聚集函数。pymysql 构造的查询语句如下：

```
select snum, avg(grade) from sc where snum in (select num from student where name = %s) and grade >= 60 group by snum
```

。查询界面与结果截图如下：



创建视图。由于创建一个已经存在的同名视图会失败，所以创建视图前，应该提示该视图是否被创建。pymysql 构造的创建视图语句如下：

create view ' + view_name + ' as select num,name,c_num from student where c_num in (select **c_num** from class where d_num in (select **d_num** from department where c_num="001" and name=%s)。创建视图界面以及重复创建视图界面如下：

创建索引。由于创建一个已经存在的同名索引会失败，所以创建视图前，应该提示该视图是否被创建。pymysql 构造的创建视图语句如下：

'create index ' + index + '_index on student(' + index + ' desc) '。创建索引界面以及重复创建索引界面如下：

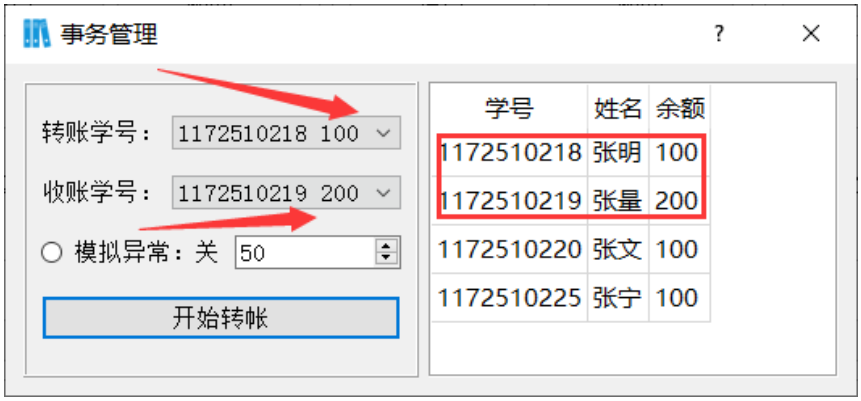
4.5 附加功能：事务管理

该功能通过点击菜单栏“事务管理”中的开启转账可以实现。

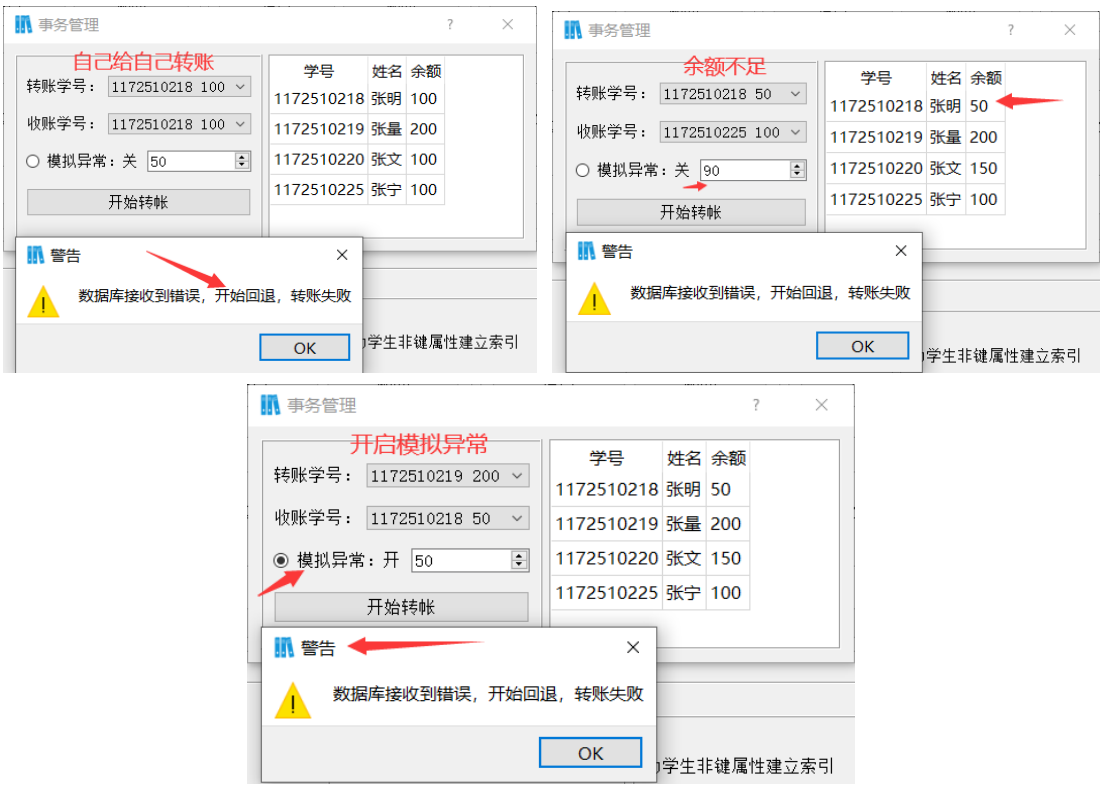
学生表中的属性 balance 用于表示饭卡余额，该虚拟余额可以用于给其余同学转账，该转账可以使用事务管理逻辑（原子性、持久性等）。

事务管理中的异常情况有：自己给自己转账，转账数目大于余额，转帐过程中遇到了其他不可抗异常因素（实验中用模拟异常代替，可以开启模拟异常）。

正常转账界面如下图（转帐前张明、张量余额为 150，转账金额为 50）.可以看到转账后，张明余额为 100，张量余额为 200，说明转账成功。



异常转账界面如下图（自己给自己转账、余额不足、出现随机异常）。



4.6 附加功能：触发器

该功能可以通过点击选项框“开启触发器”实现。

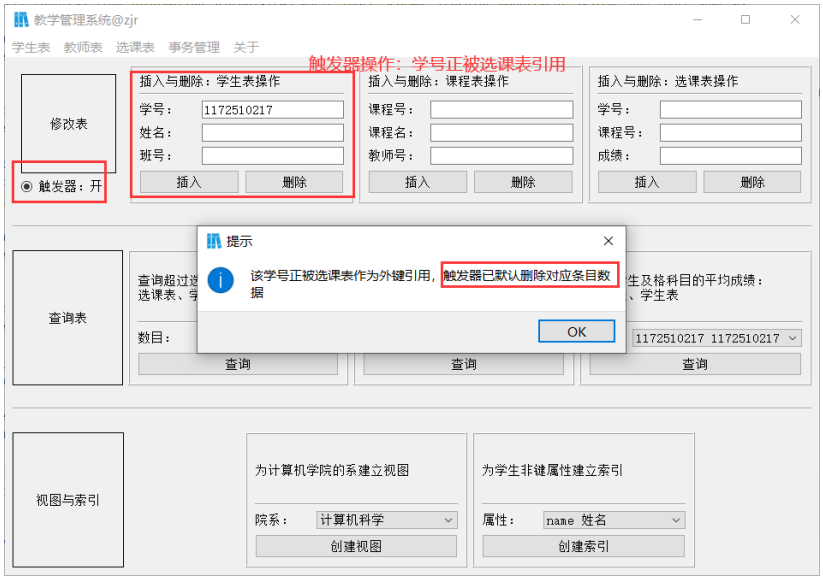
该功能用于实现选课表、学生表与课程表的参照完整性，即插入选课表信息前，执行触发器动作，判断相关主键是否存在；删除学生表或课程表信息前，执行触发器动作，判断相应主键是否被选课表作为外键引用。

情况一：插入一条选课信息，其中学号与课程号在相应表中不存在。在开启触发器前，操作不可以完成；开启触发器之后，操作可以完成，因为执行了触发器动作：插入对应表项，属性值设为默认值，如图（插入选课信息‘1172510220’, ‘010’, ‘90’，其中学号与课程号均不存在）。对应触发器语句为

```
create trigger stu_delete_trigger
    before delete on student for each row
begin
    IF old.num in (select snum from sc) THEN
        delete from sc where snum = old.num;
    END IF;
end;
```

情况二：删除一条学生信息，其中该学生作为选课表的外键被引用。在开启触发器前，操作不可以完成；开启触发器后，操作可以完成，因为执行了相关动作：删除所有引用的表项，如图（删除学生信息‘1172510217，其中学号正被引用）。对应触发器语句为：

```
create trigger sc_trigger
    before insert on sc for each row
begin
    IF new.snum not in (select num from student) THEN
     insert into student(num, name) values (new.snum, new.snum);
    END IF;
    IF new.cnum not in (select num from course) THEN
        insert into course(num, name) values (new.cnum, new.cnum);
    END IF;
end;
```

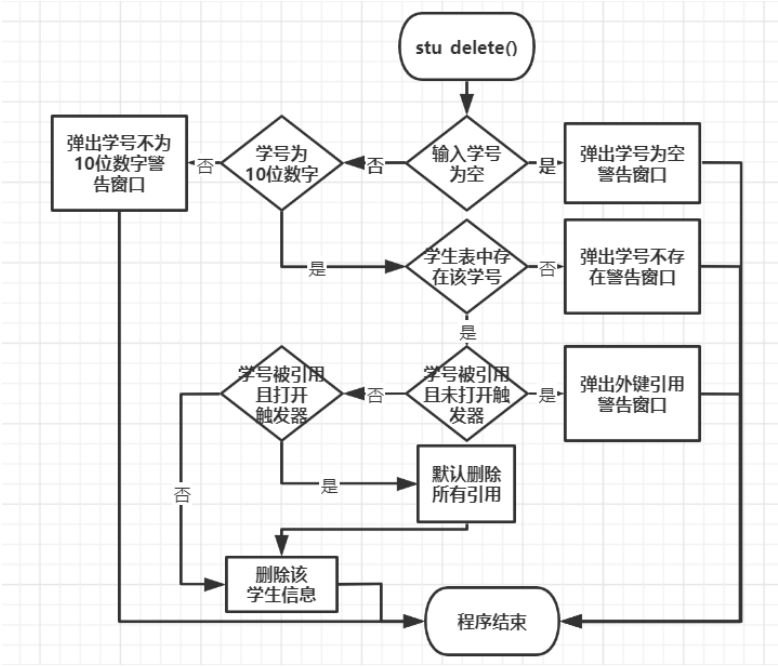


4.7 关键代码及流程图

删除学生信息的关键代码如下图所示：

```
num, trigger_on = ui.stu_num.text(), ui.add_trigger.isChecked()
if not num:
    QMessageBox.warning(self, '警告', '学号为空')
elif not num.isdigit() or len(num) != 10:
    QMessageBox.warning(self, '警告', '学号只可为10位数字')
else:
    query = 'select * from student where num=%s'
    if not cur.execute(query, [num]):
        QMessageBox.warning(self, "删除异常", "该学号不存在，请重新输入")
    elif cur.execute('select * from sc where snum =%s', [num]) and not trigger_on:
        QMessageBox.warning(self, "删除异常", "该学号正被选课表作为外键引用，请尝试删除对应条目")
    else:
        if cur.execute('select * from sc where snum=%s', [num]) and trigger_on:
            QMessageBox.information(self, '提示', '该学号正被选课表作为外键引用，触发器已默认删除对应条目数据')
        QMessageBox.information(self, '成功', '成功删除一条学生数据')
        query = 'delete from student where num=%s'
        cur.execute(query, [num])
        con.commit()  # 修改数据时，需要commit操作
        self.update_s_name_combobox()
```

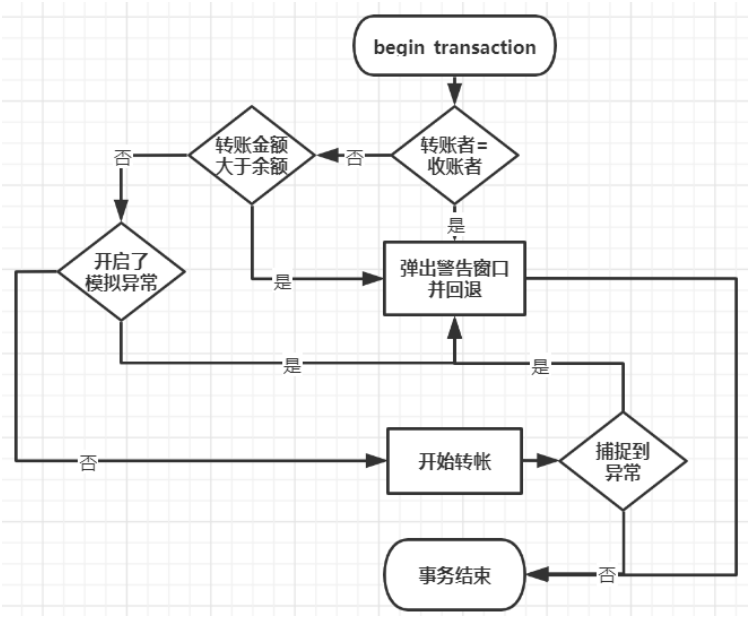
代码流程图如下图所示：



事务管理的关键代码如下图所示：

```
is_checked = self.ui.add_exception.isChecked()
con.begin()  # 开启事务
try:
    sender, s_balance = self.ui.sender_nums.currentText().split()
    receiver, r_balance = self.ui.receivers_nums.currentText().split()
    transfer_num = self.ui.transfer_num.text()
    if sender == receiver or int(s_balance) < int(transfer_num) or is_checked:
        raise Exception
    else:
        QMessageBox.information(self, '提示', '正在转帐')
        cur.execute('update student set balance=balance-' + (transfer_num) + ' where num = %s', sender)
        cur.execute('update student set balance=balance+' + transfer_num + ' where num = %s', receiver)
except Exception as e:
    QMessageBox.warning(self, '警告', '数据库接收到错误，开始回退，转账失败')
    con.rollback()
else:
    con.commit()
    QMessageBox.information(self, '提示', '转账成功')
self.__update_num()
```

代码流程图如下图所示：



五、 实验心得

5.1 遇到的问题

实体完整性约束。在执行修改数据库操作之前，要对输入的数据进行一定的检查。如插入学生表的元素不可以是空的或者是重复元素。这些逻辑可以通过数据库抛出的异常捕捉来实现或者通过提前查询来实现。

参照完整性约束。在实验中，插入选课表之前，需要对待插入的元素的学号和课程号进行判断，若相应表项已经存在，则可以正常执行插入操作，否则驳回用户的插入请求。

如何避免视图、索引和触发器的重复创建。重名的视图、索引或者触发器创建会使 pymysql 抛出异常。因此，我们需要在创建之前，添加相应的查询逻辑进行判断是否已经存在待创建内容。

5.2 解决问题

实体完整性约束实现。在执行数据库插入操作之前，添加处理逻辑：判断相应数据是否为空及是否为合法数据类型；并执行相应数据库查询操作，判断相应数据是否已经存在。相应 python 代码如下

```
if not num or not name or not c_num:
    QMessageBox.warning(self, '警告', '请输入学号、姓名与班号')
elif not num.isdigit() or len(num) != 10 or not c_num.isdigit() or len(c_num) != 3:
    QMessageBox.warning(self, '警告', '学号为 10 位数字，班号为 3 位数字')
else:
    query = 'select * from student where num=%s'
```

```
if cur.execute(query, [num]):
    QMessageBox.warning(self, '插入异常', '该学号已存在，请重新输入')
```

参照完整性约束实现。同样在执行数据库插入操作之前，添加处理逻辑：执行数据库查询操作，判断参照其他表的外码的元素是否在其他表中存在。若不存在，弹窗警告用户的操作；若存在，则可以正常插入。

避免视图、索引和触发器的重复创建。在执行创建操作之前，执行相应的查询操作。由于视图、索引和触发器的保存位置比较特殊（在基本数据库中 information_schema 中），所以需要构造特殊地查询操作，具体查询如下表：

类型	位置 information_schema	查询
视图	VIEWS	select *count*(*) from information_schema.VIEWS where TABLE_SCHEMA="teaching_management_system" and TABLE_NAME=%s
索引	INNODB_INDEXES	select *count*(*) from information_schema.INNODB_INDEXES where NAME=%s
触发器	TRIGGERS	select *count*(*) from TRIGGERS where TRIGGER_NAME=%s